# OFFER: Off-Environment Reinforcement Learning

**Kamil Ciosek** **Shimon Whiteson**
Department of Computer Science, University of Oxford

## Abstract

*Policy gradient* methods have been widely applied in reinforcement learning. For reasons of safety and cost, learning is often conducted using a simulator. However, learning in simulation does not traditionally utilise the opportunity to improve learning by adjusting certain *environment variables* – state features that are randomly determined by the environment in a physical setting but controllable in a simulator. Exploiting environment variables is crucial in domains containing *significant rare events* (SREs), e.g., unusual wind conditions that can crash a helicopter, which are rarely observed under random sampling but have a considerable impact on expected return. We propose *off environment reinforcement learning* (OFFER), which addresses such cases by simultaneously optimising the policy and a *proposal distribution* over environment variables. We prove that OFFER converges to a locally optimal policy and show experimentally that it learns better and faster than a policy gradient baseline.

## Introduction

When applying *reinforcement learning* (RL) to physical systems, a major issue is the cost and risk of running trials, e.g., when learning a control policy for a robot. Hence, learning is often performed using a simulator, to which off-line RL (i.e., sample-based planning) can be applied. Although this is cheaper and safer than physical trials, the computational cost of each trial can still be considerable. It is therefore important to develop algorithms that minimise this cost. *Policy gradient* methods (Sutton et al. 2000) are popular in such settings as they cope well with continuous action spaces, which often occur in physical systems such as robots.

However, existing policy gradient methods do not exploit an important opportunity presented by simulators: the chance to adjust certain *environment variables*, i.e., state features that cannot be controlled in a physical setting but are (stochastically) determined by the environment. For example, if we learn to fly a helicopter under varying wind conditions (Koppejan and Whiteson 2011), we cannot control the wind in physical trials but can easily do so in simulation.

A conventional application of policy gradient methods to such settings is not robust to *significant rare events* (SREs), i.e., it fails any time there are rare events that substantially affect expected performance. For example, some rare wind conditions may increase the risk of crashing the helicopter. Since crashes are so catastrophic, avoiding them is key to maximising expected performance, even though the wind conditions contributing to the crash occur only rarely. In such cases, the conventional approach does not see such rare events often enough to learn an appropriate response.

In this paper, we propose a new policy gradient method called *off-environment reinforcement learning* (OFFER) that aims to address this deficiency. The main idea is to couple the *primary optimisation* of the policy with the *secondary optimisation* of a proposal distribution governing the environment variables. Since environment variables can be controlled in simulation, we are free to sample from the proposal distribution when generating data for the primary optimisation. Thanks to importance sampling, the primary optimisation can retain unbiased gradient estimates. Just as *off-policy* RL learns about a target policy while using a behaviour policy, *off-environment* RL learns about a target environment (the true distribution over environment variables) while generating data from another environment (the proposal distribution). By learning a proposal distribution that minimises the variance in the gradient estimate used during primary optimisation, OFFER can automatically discover and focus on the SREs that any robust policy must address.

We show that OFFER is guaranteed to converge to a locally optimal policy. In addition, we present empirical results on several tasks showing that it greatly outperforms existing policy gradient methods in the presence of significant rare events.

## Related Work

Our approach is related to existing work on *adaptive importance sampling* (Ahamed, Borkar, and Juneja 2006; Frank, Mannor, and Precup 2008; Desai and Glynn 2001), which also seeks to optimise proposal distributions. However, most work focuses on *Markov reward processes*, i.e., the evaluation of a fixed policy. By contrast, our work considers how to learn a good proposal distribution for a policy that is itself changing.

As in our work, Frank, Mannor, and Precup (2008) consider a full *Markov decision process* and aim to learn using a proposal distribution that takes rare events into account. However, they assume prior knowledge of what the signif-

icant rare events are as well as access to a simulator with environment variables that directly control the probability of such rare events. By contrast, our work seeks to automatically discover significant rare events and the proposal distributions that generate them. For example, we may know that a helicopter becomes unstable in a tornado but not know a priori (1) whether such an event is important for distinguishing between policies, (2) how to make our simulator generate a tornado if it is significant, or (3) what a good proposal distribution for learning is. Our approach is designed to work without such prior knowledge.

Paul et al. (2016) consider a similar setting to ours but in the context of Bayesian optimisation, in which case environment variables must be marginalised out using Bayesian quadrature. Here, we consider a policy gradient approach, which is typically more effective in high-dimensional tasks.

Variance reduction in policy gradient methods has also been well studied. Most work focuses on subtracting an appropriate baseline (Peters and Schaal 2006; Williams 1992). Some work also considers importance sampling for variance reduction (Glynn 1990; Nakayama, Goyal, and Glynn 1994) but uses only hard-coded proposal distributions, whereas our approach learns them automatically.

Our work is also related to *safe reinforcement learning* (García and Fernández 2015), which also aims to identify risky states. Many such methods aim to optimise a risk-averse objective (Bertsekas and Rhodes 1971; Heger 1994; Malfaz and Salichs 2011), whereas we aim to more efficiently optimise a risk-neutral objective (expected return). Other methods aim to constrain exploration to avoid risky states (Gehring and Precup 2013), whereas we learn in a safe simulator and thus seek proposal distributions that visit such states *more* often, if they are significant to expected return.

## Background

We begin by formalising the problem setting and reviewing existing methods.

### Problem Setting

We model the decision-making task as a *Markov decision process* (MDP), in which taking an action $a_t \in A$ in state $s_t \in S$ at time $t$ generates a reward whose expected value is $r(s_t, a_t)$ and a transition to a next state $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. We assume access to an MDP simulator in the form of a *trajectory model* that generates sequences of samples:

$$\tau = (s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, r_3, \ldots, s_N, a_N, r_N),$$

where $s_1$ is sampled from the distribution over initial states $p_1(s_1)$ and each action $a_t$ is sampled from a stochastic policy $\pi_\theta(a_t|\phi(s_t))$ parameterised by a vector $\theta$; $\phi(s_t)$ is a function mapping $s_t$ to a vector of real-valued features. In the sequel, we write $\pi_\theta(a_t|s_t)$ for brevity. We assume $\pi_\theta$ is a twice differentiable function of $\theta$. $T$ is the set of all possible trajectories, i.e., $T = \{\tau : \exists\theta.\, p_{\pi_\theta}(\tau) > 0\}$, where $p_{\pi_\theta}(\tau) = p_1(s_1)\prod_{t=1}^{N} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$.

In addition, we assume access to a vector of *environment variables* $\psi$ (e.g., coefficients of friction, wind velocities) that affect state transition probabilities. Note that, while we can control $\psi$ when running the simulator, the policy we ultimately deploy cannot.

In this paper, we model environment variables by supposing that states in the simulator are sampled, not from $p_1(s_1)$ and $p(s_{t+1}|s_t, a_t)$, but from *proposal distributions* $f_1^\psi(s_1)$ and $f^\psi(s_{t+1}|s_1, a_t)$, which are parameterised by $\psi$. We can set $\psi$ such that $p_1 = f_1^\psi$ and $p = f^\psi$ and thus sample trajectories from the original MDP, or we can set it otherwise and thus sample trajectories from altered transition dynamics.

The goal in this setting is find a $\theta$ that maximises the total expected return:

$$J_\theta = \int_S \rho^\pi(s) \int_A \pi_\theta(a|s) r(s,a) da ds,$$

where the improper distribution $\rho^\pi(s') = \int_S \sum_{n=1}^{\infty} \gamma^{n-1} p_1(s) p(s \to s', n, \pi) ds$ and:

$$p(s \to s', n, \pi) = \sum_{\tau \in \text{Traj}(n,s,s')} \prod_{t=1}^{n} \int p(s_{t+1}|s_t, a_t)\pi(a_t|s_t) da_t,$$

where $\text{Traj}(n, s, s')$ is the set of all possible trajectories of length $n$ beginning with $s$ and ending with $s'$.

In the next section, we propose an algorithm that learns both $\theta$ and $\psi$ in parallel.

### Existing methods

*Policy gradient* (PG) methods (Sutton et al. 2000; Silver et al. 2014; Degris, Pilarski, and Sutton 2012; Deisenroth et al. 2013) use gradient ascent to directly optimise $\theta$. PG methods are appealing in many settings because they cope well continuous action spaces.

Sutton et al. (2000) showed that the gradient of $J_\theta$ with respect to $\theta$ can be written as:

$$\nabla_\theta J_\theta = \mathrm{E}_{s\sim\rho^\pi,\, a\sim\pi_\phi}\left[\nabla_\theta \log \pi_\theta(a|s) \underbrace{(Q^\pi(s,a) - b(s))}_{u(s,a)}\right], \tag{1}$$

where $b(s)$ a *baseline function*. Given a trajectory $\tau$ sampled from $p_1(s_1)$ and $p(s_{t+1}|s_t, a_t)$, we can estimate the gradient as:

$$\widehat{\nabla}_\theta J_\theta(\tau) = \sum_{t=1}^{N} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{u}_t(s_t, a_t), \tag{2}$$

where $\hat{u}_t(s_t, a_t)$ is an estimate of $u_t(s_t, a_t)$. Setting $\hat{u}_t(s_t, a_t) = \sum_{i\geq t} \gamma^{i-t} r_i - b$, where $b$ is a baseline (Peters and Schaal 2006), yields the REINFORCE method (Williams 1992) summarised in Algorithm 1.

Setting $\hat{u}_t(s_t, a_t) = r_i + \gamma\phi(s_{i+1})^\top w - \phi(s_i)^\top w$, i.e., the TD-error computed using TD($\lambda$) policy evaluation (Sutton 1988), yields an *actor-critic* method (Degris, Pilarski, and Sutton 2012) summarised in Algorithm 2.

Algorithm 2 provides an unbiased estimator for $Q(s, a) - V(s)$ (Bhatnagar et al. 2009, Lemma 3), where $V$ is the

**Algorithm 1** CRITIC-REINFORCE($\tau$)

▷ Traj. $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_N, a_N, r_N)$
$b_i \leftarrow$ compute baselines
**for** $i = 1 \ldots N$ **do**
    $\hat{u}(s_i, a_i) \leftarrow (\sum_{t=i}^{N} \gamma^{t-i} r_t) - b_i$
**end for**
**return** $\hat{u}$

**Algorithm 2** CRITIC-AC($\tau$)

▷ Traj. $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_N, a_N, r_N)$
▷ TD($\lambda$) learning algorithm
$\hat{u}(s_1, a_1) \leftarrow r_1 + \gamma \phi(s_1)^\top w$
**for** $i = 2 \ldots N$ **do**
    $\delta_i \leftarrow r_i + \gamma \phi(s_i)^\top w - \gamma \phi(s_{i-1})^\top w$
    $\hat{u}(s_i, a_i) \leftarrow \delta_i$
    $e \leftarrow \gamma \lambda e + \phi(s_{i-1})$
    $w \leftarrow \beta \delta_i e$
**end for**
**return** $\hat{u}$

value function of the current policy, thus estimating $u(s, a)$ with $b(s) = V(s)$.

Convergence of PG using Algorithm 2 is guaranteed if the critic's representation is *compatible* with the policy representation (Sutton et al. 2000).

## Off-Environment Reinforcement Learning

In this section, we propose *off-environment reinforcement learning* (OFFER) for coping with significant rare events by exploiting environment variables. The main idea is to interleave two different optimisation steps. The *primary optimisation* step performs an importance-weighted policy gradient update, adjusting $\theta$ to improve the policy's expected return. The *secondary optimisation* performs a gradient descent step on the proposal distribution, adjusting $\psi$ to reduce the variance of the gradient estimate used during primary optimisation. For example, in the helicopter domain, OFFER can optimise the control policy while simultaneously optimising the wind conditions used to evaluate that policy. Algorithm 3 summarises OFFER; the rest of this section provides the details of primary and secondary optimisation.

**Algorithm 3** OFFER()

**while** not converged **do**
    $\tau \leftarrow$ sample trajectory using $\pi_\theta$ and $f_\Psi$
    ▷ $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_N, a_N, r_N)$
    $\Delta\theta \leftarrow$ PRIMARY-OPTIMISATION($\tau$, $\theta$, $\psi$)
    $\theta \leftarrow \theta + \Delta\theta$
    $\Delta\psi \leftarrow$ SECONDARY-OPTIMISATION($\tau$, $\psi$)
    $\psi \leftarrow \psi + \Delta\psi$
**end while**

### Primary Optimisation

The primary optimisation performs a policy gradient update that adjusts $\theta$ to improve expected return. We first derive the gradient and then discuss an appropriate update rule.

To properly estimate the gradient, we must modify (2) to take into account the fact that trajectories are sampled from $f$ rather than $p$. To do so, we simply apply importance sampling to the policy gradient update (Glynn 1990):

$$\nabla_\theta J_\theta = \mathrm{E}_{\tau \sim p_\pi(\tau)} \left[ \sum_{t=1}^{N} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) Q^\pi(s_t, a_t) \right] \tag{3}$$
$$= \mathrm{E}_{\tau \sim f_\pi(\tau)} \left[ \frac{p_\pi(\tau)}{f_\pi(\tau)} \sum_{t=1}^{N} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) Q^\pi(s_t, a_t) \right],$$

where $f_\pi(\tau) = f_1(s_1) \prod_{t=1}^{N} f(s_{t+1}|s_t, a_t) \pi(a_t|s_t)$. Given a concrete sampled trajectory $\tau$, we can then estimate the gradient as,

$$\hat{\nabla}_\theta J_\theta = \frac{p_1(s_1)}{f_1(s_1)} \prod_{t=1}^{N} \frac{p(s_{t+1}|s_t, a_t)}{f(s_{t+1}|s_t, a_t)} \sum_{t=1}^{N} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) Q^\pi(s_t, a_t). \tag{4}$$

We now consider how to use (4) to update $\theta$. A naive approach would be to use the standard policy gradient update $\Delta\theta = \alpha \hat{\nabla}_\theta J_\theta$. However, doing so is problematic in any setting characterised by significant rare events.

To see why, first note that the magnitude of the update depends on the magnitude of the gradient, which in turn depends on the magnitude of the rewards. Now consider two MDPs that are identical save that in one MDP, the rewards have been multiplied by a large constant. Clearly, for a given $\theta$, the magnitude of $\Delta\theta$ should be the same in both MDPs. However, since $\hat{\nabla}_\theta J_\theta$ will be different, the two MDPs will require different values of the learning rate $\alpha$.

We now show that, in the presence of SREs, the same problem can arise within a single MDP. We need the following definitions.

**Definition 1.** *A* partition $P = \{E_1, E_2, \ldots, E_{|P|}\}$ *divides the set of possible trajectories* $T$ *into* events*,* $E_i$.

For example, a simple partition $P = \{E_{\text{NE}}, E_{\text{RE}}\}$ involves just normal and rare events.

**Definition 2.** *A policy feature* $\theta_i$ *is* salient *with respect to an event* $E$ *under partition* $P$ *if* $\exists \tau \in E. \{\hat{\nabla}_\theta J_\theta(\tau)\}_i \neq 0$.

Note that different policy features can be salient for different subsets of events, e.g., $\theta_i$ might be salient only for $E_{RE}$ while $\theta_j$ might be salient for both $E_{RE}$ and $E_{NE}$. Now suppose that different events have different reward scales, e.g., trajectories in $E_{RE}$ trigger rewards of a larger order of magnitude than those in $E_{NE}$. Then, different policy features will need different learning rates, just as in the example with two MDPs. Consequently, the ability to adaptively set the learning rate separately for each policy feature becomes an essential, not merely desirable, characteristic of the learning algorithm in our setting.

To meet this requirement, we employ a stochastic variant of Newton's method (Furmston and Barber 2012; Spall 2000). In our setting, the Hessian $H$ equals $\nabla_\theta \nabla_\theta^\top J_\theta$. This can be estimated from a sampled trajectory, yielding $\hat{H}(\tau)$.

Furthermore, we can maintain a mean of the Hessians estimated over time: $\hat{H}_M(\tau'_1, \ldots, \tau'_N) = \sum_{i=1}^{N} \frac{1}{N} \hat{H}(\tau'_i)$. The gradient update is then:

$$\Delta\theta = \alpha \, \mathrm{diag}(\hat{H}_M(\tau_1, \ldots, \tau_{n+1}))^{-1} \nabla_\theta \widehat{J_\theta(\tau_{n+1})}.$$

To avoid the prohibitive cost of inverting the full Hessian, only its diagonal is used, which corresponds to performing Newton's method on each coordinate separately. As a result, the computational complexity is the same as that of vanilla gradient descent.

Algorithm 4 summarises the resulting primary optimisation algorithm, which calls either Algorithm 1 or 2 as a subroutine. The diagonal variant of Newton's method that Algorithm 4 implements can be seen as an automatic way of setting learning rates for vanilla gradient descent (Furmston and Barber 2012). Consequently, it addresses the problem described above: the estimate of the $i$-th entry of the vector $\hat{\nabla}_\theta J_\theta$ and the $i$-th diagonal entry of $\hat{H}_M$ both scale with $\hat{u}$ (the output of Algorithms 1 and 2), which scales with the rewards. Hence, if some policy features are salient for different subsets of events, then the effects on the update cancel in line 11 of Algorithm 4.

**Algorithm 4** PRIMARY-OPTIMISATION($\tau$, $\theta$,$\psi$)

1: ▷ Traj. $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_N, a_N, r_N)$
2: ▷ Critic
3: $\hat{u} \leftarrow$ CRITIC-REINFORCE($\tau$) or CRITIC-AC($\tau$)
4:
5: ▷ Actor
6: $w \leftarrow \frac{p_1(s_1)}{f_1(s_1)} \prod_{t=1}^{N} \frac{p(s_{t+1}|s_t,a_t)}{f(s_{t+1}|s_t,a_t)}$ ▷ Importance sampling
7: $\hat{\nabla}_\theta J_\theta \leftarrow w \sum_{i=1}^{N} \gamma^i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{u}_i$
8: $\hat{H} \leftarrow w \sum_{i=1}^{N} \gamma^i \mathrm{diag}(\nabla_\theta \nabla_\theta^\top \log \pi_\theta(a_i|s_i)) \hat{u}_i$
9: $\hat{H}_M \leftarrow \frac{i-1}{i} \hat{H}_M + \frac{1}{i} \hat{H}$
10:
11: **return** $\alpha_i \hat{H}_M^{-1} \hat{\nabla}_\theta J_\theta$

We use Newton's method because it has already been shown to work well with policy gradient methods (Furmston and Barber 2012). However, other adaptive learning rate approaches such as ADAM (Kingma and Ba 2014) could also be considered. Nevertheless, as we discuss below, using ADAM would complicate the secondary optimisation.

## Secondary Optimisation

The goal of secondary optimisation is to find a proposal distribution that best facilitates primary optimisation. To this end, we propose to minimise the variance of the gradient estimate followed during primary optimisation. Such variance is known to be a key contributor to slow learning in policy gradient methods (Peters and Schaal 2006; Williams 1992; Glynn 1990; Nakayama, Goyal, and Glynn 1994). By minimising this variance, we expect OFFER to discover proposal distributions that generate significant rare events more often than in the original task. Since such events contribute substantially to expected return, doing so makes it easier to estimate the gradient of the expected return.

We start by defining the covariance:

$$\begin{aligned} C &= \mathrm{Cov}_{\tau \sim f^\psi} \left[ \frac{1}{f^\psi(\tau)} \underbrace{p(\tau) \sum_{t=1}^{N} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) u_t}_{h(\tau)} \right] \\ &= \mathrm{Cov}_{\tau \sim f^\psi} \left[ \frac{1}{f^\psi(\tau)} h(\tau) \right] \\ &= \mathrm{E}_{\tau \sim f^\psi} \left[ \left( \frac{h(\tau)}{f^\psi(\tau)} \right) \left( \frac{h(\tau)}{f^\psi(\tau)} \right)^\top \right] - \\ &\qquad \left( \mathrm{E}_{\tau \sim f^\psi} \left[ \frac{h(\tau)}{f^\psi(\tau)} \right] \right) \left( \mathrm{E}_{\tau \sim f^\psi} \left[ \frac{h(\tau)}{f^\psi(\tau)} \right] \right)^\top . \end{aligned} \tag{5}$$

Since we are interested in minimising our uncertainty about each of the partial derivatives that make up the gradient, rather than the correlations between them, we consider only the diagonal of $C$. Furthermore, since we have no a priori reason to think one partial derivative is more important than another, we define the trace of $C$ as our objective:

$$\min_\psi \mathrm{trace}(C) \, . \tag{6}$$

Note that, if we treat $\hat{H}_M$ as a constant (which is approximately true for large $N$), then $\Delta\theta$, as computed by Newton's method, is a linear function of $\hat{\nabla}_\theta J_\theta$. Hence, minimising the covariance of $\hat{\nabla}_\theta J_\theta$ also minimises the covariance of $\Delta\theta$. By contrast, in ADAM, $\Delta\theta$ is a nonlinear function of $\hat{\nabla}_\theta J_\theta$ that divides by the square root of the second moment. Hence, using ADAM for primary optimisation would necessitate a more complex secondary objective.

To solve (6) by gradient descent, we must evaluate the gradient of the trace with respect to $\phi$. The gradient with respect to the second term in (5) is zero because,

$$\nabla_\psi \, \mathrm{E}_{\tau \sim f^\psi(\tau)} \left[ \frac{h(\tau)}{f^\psi(\tau)} \right] = 0.$$

Hence, we focus on the gradient of the trace of the first term:

$$\begin{aligned} &\nabla_{\psi_j} \, \mathrm{trace} \left( \mathrm{E}_{\tau \sim f^\psi} \left[ \left( \frac{h(\tau)}{f^\psi(\tau)} \right) \left( \frac{h(\tau)}{f^\psi(\tau)} \right)^\top \right] \right) \\ &\quad = \sum_i \nabla_{\psi_j} \int \left( \frac{h_i(\tau)}{f^\psi(\tau)} \right)^2 f^\psi(\tau) d\tau \\ &\quad = \sum_i \nabla_{\psi_j} \int \frac{h_i(\tau)^2}{f^\psi(\tau)} d\tau \\ &\quad = \sum_i \int \left( \nabla_{\psi_j} \frac{1}{f^\psi(\tau)} \right) h_i(\tau)^2 d\tau \\ &\quad = -\sum_i \int \frac{1}{f^\psi(\tau)^2} \left( \nabla_{\psi_j} f(\tau) \right) h_i(\tau)^2 d\tau \\ &\quad = -\sum_i \int \frac{1}{f^\psi(\tau)^3} \left( \nabla_{\psi_j} f(\tau) \right) h_i(\tau)^2 f^\psi(\tau) d\tau \\ &\quad = \mathrm{E}_{\tau \sim f^\psi} \left[ -\sum_i \frac{1}{f^\psi(\tau)^3} \left( \nabla_{\psi_j} f(\tau) \right) h_i(\tau)^2 \right] . \end{aligned}$$

Consequently:

$$\nabla_{\psi_j} \operatorname{trace}(C) = \mathrm{E}_{\tau \sim f^\psi}\left[ -\sum_i \frac{1}{f^\psi(\tau)^3} \left(\nabla_{\psi_j} f(\tau)\right) h_i(\tau)^2 \right],$$

which can be estimated from a trajectory $\tau$ as follows:

$$\hat{\nabla}_{\psi_j} \operatorname{trace}(C) = \sum_i -\frac{1}{f^\psi(\tau)^3} \left(\nabla_{\psi_j} f(\tau)\right) \hat{h}_i(\tau)^2, \quad (7)$$

where $\hat{h}_i(\tau) = p(\tau) \sum_{t=1}^N \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{u}_t$.

Unlike the primary optimsation, the secondary optimisation can be performed using any variant of stochastic gradient descent. Since our estimate of the gradient (7) relies on only one trajectory, we employ ADAM, which smooths estimates across previous trajectories. Algorithm 5 summarises the secondary optimisation.

**Algorithm 5** SECONDARY-OPTIMISATION($\tau$, $\psi$)

▷ Traj. $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_N, a_N, r_N)$
▷ ADAM algorithm
$\hat{\nabla}_{\psi_j} \operatorname{trace}(C) \leftarrow \frac{1}{m} \sum_{\tau \in S} \sum_i -\frac{1}{f^\psi(\tau)^3} \left(\nabla_{\psi_j} f(\tau)\right) h_i(\tau)^2$
$m \leftarrow \beta_1 m + (1-\beta_1)\hat{\nabla}_{\psi_j} \operatorname{trace}(C)$
▷ $\odot$ is an element-wise product
$v \leftarrow \beta_2 v + (1-\beta_2)\hat{\nabla}_{\psi_j} \operatorname{trace}(C) \odot \hat{\nabla}_{\psi_j} \operatorname{trace}(C)$
$\hat{m} \leftarrow m/(1-\beta_1^i)$
$\hat{v} \leftarrow v/(1-\beta_2^i)$

▷ Division and square root are element-wise
**return** $\alpha' \hat{m}/\sqrt{\hat{v}+\epsilon}$

## Convergence Guarantee

In this section, we show that OFFER has convergence guarantees similar to those of existing policy gradient methods (Peters and Schaal 2008). In particular, we use existing results to establish a corollary showing that $\theta$ converges to a local optimum of $J$.

**Corollary 1.** *If the following assumptions hold:*

1. *$f$ is non-zero everywhere;*
2. *$J$ is convex over some restricted domain $B$ containing the local optimum $\theta^\star$;*
3. *All iterates $\theta^{(1)}, \ldots$ belong to $B$;*
4. *Robbins-Monro conditions (Spall 2000, Condition C.1') hold for the learning rate $\alpha_i$;*
5. *At each iteration $k$, there exist $\delta, \rho > 0$ such that $\mathrm{E}\left[\|\hat{\nabla}_\theta J_\theta{}^{(k)}\|^{2+\delta}\right] \leq \rho$;*
6. *$J$ has a uniformly bounded third derivative in $B$;*
7. *At each iteration, $\hat{H}_M^{(k)}$ is diagonal with positive entries*[1]*;*
8. *At each iteration $k$, there exist $\delta, \rho > 0$ such that $\mathrm{E}\left[\|(\hat{H}_M^{(k)})^{-1}\|^{2+\delta}\right] \leq \rho$;*
9. *If we are using a critic, then it is compatible (Sutton et al. 2000) with the policy, i.e., $\nabla_\theta u(s,a) = (\nabla_\theta \pi(s,a)) \frac{1}{\pi(s,a)}$,*

*then OFFER (Algorithm 3) converges almost surely to the local minimum $\theta^\star$. Moreover, $\psi$ converges to a local minimum of the optimisation problem in* (6)*.*

*Proof.* First we show $\mathrm{E}\left[\hat{\nabla}_\theta J_\theta\right] = \nabla_\theta J_\theta$. For the REINFORCE-based critic, this follows from the derivation of (3), i.e., importance sampling does not introduce bias (Owen 2013, Chapter 9). For the actor-critic, the same reasoning together with Assumption 9 implies that the gradient involving the approximate critic is the same in expectation as the true gradient (Sutton et al. 2000, Theorem 2). Since this condition holds at each iteration, even though $\psi$ and the distribution the expectation is taken with respect to changes, we satisfy condition C.0′ from (Spall 2000), which requires that we use a sequence of unbiased estimators.

Moreover, Assumption 7 implies that, at each iteration $k$, for each coordinate $i$, $\operatorname{sgn}(\{\nabla_\theta J_\theta(\theta_k)\}_i) = \operatorname{sgn}(\{\hat{H}_M^{-1} \nabla_\theta J_\theta(\theta_k)\}_i)$. Furthermore, Assumption 2 implies that the iterates are, of necessity, bounded. Together, these facts imply the lemma "Sufficient Conditions for C.5 and C.7" from (Spall 2000), which shows that certain conditions relating to the Hessian estimate are satisfied.

Given this lemma, we can instantiate the main theorem (Spall 2000, Theorem 1b-2SG), which says that a general second-order stochastic optimisation scheme, of which the primary optimisation of Algorithm 4 is a special case, converges given a set of conditions that we have either just shown or which follow directly from our assumptions.

The secondary optimisation converges because eventual convergence of $\theta$ renders the secondary optimisation problem stationary, at which point ADAM's convergence guarantee applies (Kingma and Ba 2014, Theorem 4.1).

□

Note that Corollary 1 holds for any choice of $f^\psi$, not just the ones learned by OFFER: a poor choice of $\psi$ will only slow convergence, not prevent it.

## Experiments

In this section, we empirically compare OFFER to a policy gradient baseline (primary optimisation only) on variants of the mountain car task, as well as a simulated robot arm. We describe the experimental setup only briefly; complete details are in the supplementary material. Both domains have penalties rather than rewards, so lower is better on all plots. All results are averaged over 48 runs.

### Mountain Car

We modified the mountain car benchmark task by adding a third state feature (besides position and velocity): a boolean variable indicating whether a rare event occurs in the given episode. With probability 0.01, this boolean is true, in which

[1] Existing techniques (Spall 2000) can be used to show consistency for non-diagonal Hessians as well. In this work, we opt for the diagonal Hessian for efficiency reasons.

case the landscape is shifted to the right, changing when the agent should go forwards or backwards, and penalties are multiplied by 1000. In our setup, $\theta$ consists of the parameters of a standard tile-coding function approximator, and $\psi$ is a single scalar that determines the probability of a rare event.

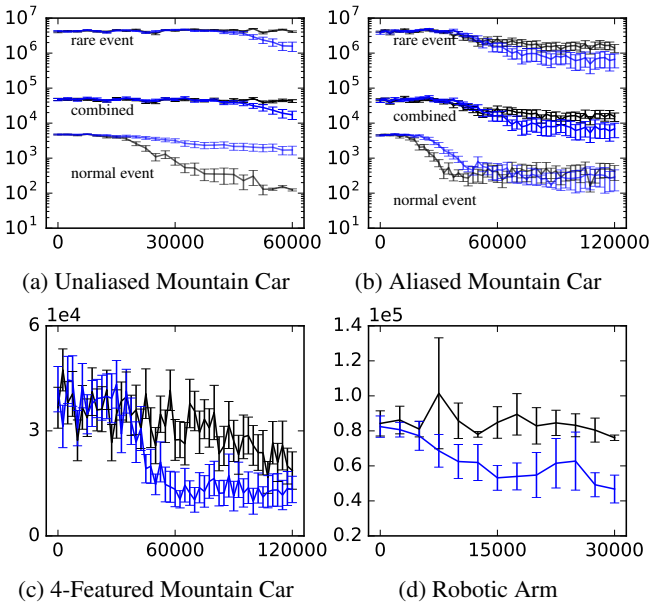
We first consider an *unaliased* setting (Figure 1a), in which the agent has separate features for normal and rare events, i.e., the tile coding is duplicated, with each feature activated only for normal or rare events, not both. Hence, the agent can in principle learn effective policies for both events, but must learn a proposal distribution that allocates data appropriately to them.

The lines in the middle of the plot show the average performance of OFFER (blue) compared to the policy gradient baseline (black). OFFER substantially improves performance (note the log scale on the $y$-axis). In fact, the baseline shows negligible learning, confirming that optimising the proposal distribution can be key to successful learning when significant rare events are present. Note that, unlike in previous work (Frank, Mannor, and Precup 2008), OFFER discovered a good proposal distribution on its own.

The lines at the top and bottom of Figure 1a show the performance of the same learning runs split out into just rare events and just normal events, respectively, i.e., the middle lines are weighted averages of the top and bottom lines. These lines show that the baseline algorithm is actually learning, but its progress is limited to the normal events that dominate its data but contribute only negligibly to expected return. By contrast, OFFER learns a proposal distribution that samples rare events with a probability of approximately 0.8, enabling it to efficiently learn how to handle such events. It learns more slowly on the normal events but has correctly concluded that doing so is worth the tradeoff.

Next, we consider an *aliased* setting (Figure 1b) in which the actor uses the same features in $\theta$ for both normal and rare events, i.e., it cannot directly observe a rare event or condition its behaviour on it. Unlike in the unaliased setting, it cannot learn separate policies for the two events but must learn a single policy that balances between the two, with the correct balance depending critically on both the probability and relative scale of penalties for rare events. Here, OFFER again performs much better. It learns a proposal distribution assigning a probability of about 0.9 to rare events, enabling it to more quickly learn to cope with such events. Furthermore, though initially slower, it quickly catches up to the baseline even on the normal events. Because the two events are somewhat similar, knowledge gained in the rare events transfers to the normal events via the shared features.

Finally, we consider a variant of mountain car (Figure 1c) with a more complex representation of $\psi$, which is now a vector of four features, yielding a bigger challenge for secondary optimisation. Two features control the probability of *trigger events*, with the rare event occurring only if both are triggered. The other two features control the probability of *slippery* and *jittery* conditions, which affect the task but not as dramatically as a rare event. The results show that OFFER again outperforms the baseline, and can thus effectively optimise the proposal distribution even when doing so requires setting multiple features correctly. The learned proposal distribution gives high values to the trigger events to maximise the probability of rare events, while the probabilities of slippery and jittery conditions vary per run.




(a) Unaliased Mountain Car (b) Aliased Mountain Car

(c) 4-Featured Mountain Car (d) Robotic Arm

Figure 1: *Average per-episode penalty on the orig. MDP (lower is better) for OFFER (blue) and baseline PG (black).*

## Robotic Arm

We also applied OFFER to a simulated robotic-arm control task (Figure 1d) that was recently proposed specifically to model significant rare events (Paul et al. 2016). An off-the-shelf kinematics simulator (Cully et al. 2015) models an arm with three joints, each of which can be set by the agent, whose goal is to get the end effector to a specific location. The task is episodic and in each step the agent moves each joint by at most 30% of the allowed movement range. The penalty the agent receives is proportional to the distance of the end effector to the target location. Moreover, there is a wall near the arm and the agent incurs a large penalty if the arm hits it. Usually, the arm is far from the wall (normal event) but occasionally it is near to it (rare event). OFFER learns a safe policy that avoids the wall even when it is nearby, whereas the baseline is unable to solve the task.

## Conclusions & Future Work

We proposed *off-environment reinforcement learning* (OFFER), which speeds convergence of policy gradient methods by optimising the proposal distribution from which environment variables are set. We prove that OFFER converges to a locally optimal policy. Furthermore, empirical results in multiple tasks show that OFFER learns better and faster than a policy gradient baseline in settings that are characterised by significant rare events. In future, we plan to apply OFFER to deterministic policy gradient methods (Silver et al. 2014) as well as to the generalised helicopter hovering task (Koppejan and Whiteson 2011), which is known to be characterised by significant rare events.

## References


Ahamed, T. I.; Borkar, V. S.; and Juneja, S. 2006. Adaptive importance sampling technique for markov chains using stochastic approximation. *Operations Research* 54(3):489–504.

Bertsekas, D. P., and Rhodes, I. B. 1971. On the minimax feedback control of uncertain dynamic systems. In *Decision and Control, 1971 IEEE Conference on*, 451–455. IEEE.

Bhatnagar, S.; Sutton, R.; Ghavamzadeh, M.; and Lee, M. 2009. Natural actor–critic algorithms. *Automatica*.

Cully, A.; Clune, J.; Tarapore, D.; and Mouret, J.-B. 2015. Robots that can adapt like animals. *Nature* 521(7553):503–507.

Degris, T.; Pilarski, P. M.; and Sutton, R. S. 2012. Model-free reinforcement learning with continuous action in practice. In *American Control Conference (ACC), 2012*, 2177–2182. IEEE.

Deisenroth, M. P.; Neumann, G.; Peters, J.; et al. 2013. A survey on policy search for robotics. *Foundations and Trends in Robotics* 2(1-2):1–142.

Desai, P. Y., and Glynn, P. W. 2001. Simulation in optimization and optimization in simulation: a markov chain perspective on adaptive monte carlo algorithms. In *Proceedings of the 33nd conference on Winter simulation*, 379–384. IEEE Computer Society.

Frank, J.; Mannor, S.; and Precup, D. 2008. Reinforcement learning in the presence of rare events. In *Proceedings of the 25th international conference on Machine learning*, 336–343. ACM.

Furmston, T., and Barber, D. 2012. A unifying perspective of parametric policy search methods for markov decision processes. In *Advances in Neural Information Processing Systems*, 2717–2725.

García, J., and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16:1437–1480.

Gehring, C., and Precup, D. 2013. Smart exploration in reinforcement learning using absolute temporal difference errors. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 1037–1044. International Foundation for Autonomous Agents and Multiagent Systems.

Glynn, P. W. 1990. Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM* 33(10):75–84.

Heger, M. 1994. Consideration of risk in reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, 105–111.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koppejan, R., and Whiteson, S. 2011. Neuroevolutionary reinforcement learning for generalized control of simulated helicopters. *Evolutionary intelligence* 4(4):219–241.

Malfaz, M., and Salichs, M. A. 2011. Learning to avoid risky actions. *Cybernetics and Systems* 42(8):636–658.

Nakayama, M. K.; Goyal, A.; and Glynn, P. W. 1994. Likelihood ratio sensitivity analysis for markovian models of highly dependable systems. *Operations Research* 42(1):137–157.

Owen, A. B. 2013. *Monte Carlo theory, methods and examples*.

Paul, S.; Ciosek, K.; Osborne, M. A.; and Whiteson, S. 2016. Alternating optimisation and quadrature for robust reinforcement learning. *CoRR* abs/1605.07496.

Peters, J., and Schaal, S. 2006. Policy gradient methods for robotics. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2219–2225. IEEE.

Peters, J., and Schaal, S. 2008. Reinforcement learning of motor skills with policy gradients. *Neural Networks* 21(4):682–697.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *Proceedings of The 31st International Conference on Machine Learning*, 387–395.

Spall, J. C. 2000. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE transactions on automatic control* 45(10):1839–1853.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1057–1063.

Sutton, R. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3:9–44.

Williams, R. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.