# Off-Policy Actor-Critic

**Thomas Degris** THOMAS.DEGRIS@INRIA.FR
Flowers Team, INRIA, Talence, ENSTA-ParisTech, Paris, France

**Martha White** WHITEM@CS.UALBERTA.CA
**Richard S. Sutton** SUTTON@CS.UALBERTA.CA
RLAI Laboratory, Department of Computing Science, University of Alberta, Edmonton, Canada

## Abstract

This paper presents the first actor-critic algorithm for off-policy reinforcement learning. Our algorithm is online and incremental, and its per-time-step complexity scales linearly with the number of learned weights. Previous work on actor-critic algorithms is limited to the on-policy setting and does not take advantage of the recent advances in off-policy gradient temporal-difference learning. Off-policy techniques, such as Greedy-GQ, enable a target policy to be learned while following and obtaining data from another (behavior) policy. For many problems, however, actor-critic methods are more practical than action value methods (like Greedy-GQ) because they explicitly represent the policy; consequently, the policy can be stochastic and utilize a large action space. In this paper, we illustrate how to practically combine the generality and learning potential of off-policy learning with the flexibility in action selection given by actor-critic methods. We derive an incremental, linear time and space complexity algorithm that includes eligibility traces, prove convergence under assumptions similar to previous off-policy algorithms, and empirically show better or comparable performance to existing algorithms on standard reinforcement-learning benchmark problems.

The reinforcement learning framework is a general temporal learning formalism that has, over the last few decades, seen a marked growth in algorithms and applications. Until recently, however, practical online methods with convergence guarantees have been restricted to the *on-policy* setting, in which the agent learns only about the policy it is executing.

In an *off-policy* setting, on the other hand, an agent learns about a policy or policies different from the one it is executing. Off-policy methods have a wider range of applications and learning possibilities. Unlike on-policy methods, off-policy methods are able to, for example, learn about an optimal policy while executing an exploratory policy (Sutton & Barto, 1998), learn from demonstration (Smart & Kaelbling, 2002), and learn multiple tasks in parallel from a single sensorimotor interaction with an environment (Sutton et al., 2011). Because of this generality, off-policy methods are of great interest in many application domains.

The most well known off-policy method is Q-learning (Watkins & Dayan, 1992). However, while Q-Learning is guaranteed to converge to the optimal policy for the tabular (non-approximate) case, it may diverge when using linear function approximation (Baird, 1995). Least-squares methods such as LSTD (Bradtke & Barto, 1996) and LSPI (Lagoudakis & Parr, 2003) can be used off-policy and are sound with linear function approximation, but are computationally expensive; their complexity scales quadratically with the number of features and weights. Recently, these problems have been addressed by the new family of gradient-TD (Temporal Difference) methods (e.g., Sutton et al., 2009), such as Greedy-GQ (Maei et al., 2010), which are of linear complexity and convergent under off-policy training with function approximation.

All action-value methods, including gradient-TD methods such as Greedy-GQ, suffer from three important limitations. First, their target policies are deterministic, whereas many problems have stochastic optimal policies, such as in adversarial settings or in partially observable Markov decision processes. Second, finding the greedy action with respect to the action-

Appearing in *Proceedings of the 29$^{th}$ International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

value function becomes problematic for larger action spaces. Finally, a small change in the action-value function can cause large changes in the policy, which creates difficulties for convergence proofs and for some real-time applications.

The standard way of avoiding the limitations of action-value methods is to use policy-gradient algorithms (Sutton et al., 2000) such as actor-critic methods (e.g., Bhatnagar et al., 2009). For example, the natural actor-critic, an on-policy policy-gradient algorithm, has been successful for learning in continuous action spaces in several robotics applications (Peters & Schaal, 2008).

The first and main contribution of this paper is to introduce the first actor-critic method that can be applied off-policy, which we call Off-PAC, for Off-Policy Actor–Critic. Off-PAC has two learners: the actor and the critic. The actor updates the policy weights. The critic learns an off-policy estimate of the value function for the current actor policy, different from the (fixed) behavior policy. This estimate is then used by the actor to update the policy. For the critic, in this paper we consider a version of Off-PAC that uses GTD($\lambda$) (Maei, 2011), a gradient-TD method with eligibitity traces for learning state-value functions. We define a new objective for our policy weights and derive a valid backward-view update using eligibility traces. The time and space complexity of Off-PAC is linear in the number of learned weights.

The second contribution of this paper is an off-policy policy-gradient theorem and a convergence proof for Off-PAC when $\lambda = 0$, under assumptions similar to previous off-policy gradient-TD proofs.

Our third contribution is an empirical comparison of Q($\lambda$), Greedy-GQ, Off-PAC, and a soft-max version of Greedy-GQ that we call Softmax-GQ, on three benchmark problems in an off-policy setting. To the best of our knowledge, this paper is the first to provide an empirical evaluation of gradient-TD methods for off-policy control (the closest known prior work is the work of Delp (2011)). We show that Off-PAC outperforms other algorithms on these problems.

## 1. Notation and Problem Setting

In this paper, we consider Markov decision processes with a discrete state space $\mathcal{S}$, a discrete action space $\mathcal{A}$, a distribution $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, where $P(s'|s, a)$ is the probability of transitioning into state $s'$ from state $s$ after taking action $a$, and an expected reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ that provides an expected reward for taking action $a$ in state $s$ and transitioning into $s'$. We observe a stream of data, which includes states $s_t \in \mathcal{S}$, actions $a_t \in \mathcal{A}$, and rewards $r_t \in \mathbb{R}$ for $t = 1, 2, \ldots$ with actions selected from a fixed behavior policy, $b(a|s) \in (0, 1]$.

Given a termination condition $\gamma : \mathcal{S} \to [0, 1]$ (Sutton et al., 2011), we define the value function for $\pi : \mathcal{S} \times \mathcal{A} \to (0, 1]$ to be:

$$V^{\pi,\gamma}(s) = \mathrm{E}\left[r_{t+1} + \ldots + r_{t+T} | s_t = s\right] \ \forall s \in \mathcal{S} \quad (1)$$

where policy $\pi$ is followed from time step $t$ and terminates at time $t + T$ according to $\gamma$. We assume termination always occurs in a finite number of steps.

The action-value function, $Q^{\pi,\gamma}(s, a)$, is defined as:

$$Q^{\pi,\gamma}(s, a) = \sum_{s' \in \mathcal{S}} P(s'|s, a)[\mathcal{R}(s, a, s') + \gamma(s')V^{\pi,\gamma}(s')] \quad (2)$$

for all $a \in \mathcal{A}$ and for all $s \in \mathcal{S}$. Note that $V^{\pi,\gamma}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)Q^{\pi,\gamma}(s, a)$, for all $s \in \mathcal{S}$.

The policy $\pi_{\mathbf{u}} : \mathcal{A} \times \mathcal{S} \to [0, 1]$ is an arbitrary, differentiable function of a weight vector, $\mathbf{u} \in \mathbb{R}^{N_{\mathbf{u}}}$, $N_{\mathbf{u}} \in \mathbb{N}$, with $\pi_{\mathbf{u}}(a|s) > 0$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$. Our aim is to choose $\mathbf{u}$ so as to maximize the following scalar objective function:

$$J_\gamma(\mathbf{u}) = \sum_{s \in \mathcal{S}} d^b(s)V^{\pi_{\mathbf{u}},\gamma}(s) \quad (3)$$

where $d^b(s) = \lim_{t \to \infty} P(s_t = s|s_0, b)$ is the limiting distribution of states under $b$ and $P(s_t = s|s_0, b)$ is the probability that $s_t = s$ when starting in $s_0$ and executing $b$. The objective function is weighted by $d^b$ because, in the off-policy setting, data is obtained according to this behavior distribution. For simplicity of notation, we will write $\pi$ and implicitly mean $\pi_{\mathbf{u}}$.

## 2. The Off-PAC Algorithm

In this section, we present the Off-PAC algorithm in three steps. First, we explain the basic theoretical ideas underlying the gradient-TD methods used in the critic. Second, we present our off-policy version of the policy-gradient theorem. Finally, we derive the forward view of the actor and convert it to a backward view to produce a complete mechanistic algorithm using eligibility traces.

### 2.1. The Critic: Policy Evaluation

Evaluating a policy $\pi$ consists of learning its value function, $V^{\pi,\gamma}(s)$, as defined in Equation 1. Since it is often impractical to explicitly represent every

state $s$, we learn a linear approximation of $V^{\pi,\gamma}(s)$: $\hat{V}(s) = \mathbf{v}^\top \mathbf{x}_s$ where $\mathbf{x}_s \in \mathbb{R}^{N_\mathbf{v}}$, $N_\mathbf{v} \in \mathbb{N}$, is the feature vector of the state $s$, and $\mathbf{v} \in \mathbb{R}^{N_\mathbf{v}}$ is another weight vector.

Gradient-TD methods (Sutton et al., 2009) incrementally learn the weights, $\mathbf{v}$, in an off-policy setting, with a guarantee of stability and a linear per-time-step complexity. These methods minimize the $\lambda$-weighted mean-squared projected Bellman error:

$$\text{MSPBE}(\mathbf{v}) = ||\hat{V} - \Pi T_\pi^{\lambda,\gamma} \hat{V}||_D^2$$

where $\hat{V} = X\mathbf{v}$; $X$ is the matrix whose rows are all $\mathbf{x}_s$; $\lambda$ is the decay of the eligibility trace; $D$ is a matrix with $d^b(s)$ on its diagonal; $\Pi$ is a projection operator that projects a value function to the nearest representable value function given the function approximator; and $T_\pi^{\lambda,\gamma}$ is the $\lambda$-weighted Bellman operator for the target policy $\pi$ with termination probability $\gamma$ (e.g., see Maei & Sutton, 2010). For a linear representation, $\Pi = X(X^\top D X)^{-1} X^\top D$.

In this paper, we consider the version of Off-PAC that updates its critic weights by the GTD($\lambda$) algorithm introduced by Maei (2011).

### 2.2. Off-policy Policy-gradient Theorem

Like other policy gradient algorithms, Off-PAC updates the weights approximately in proportion to the gradient of the objective:

$$\mathbf{u}_{t+1} - \mathbf{u}_t \approx \alpha_{u,t} \nabla_\mathbf{u} J_\gamma(\mathbf{u}_t) \tag{4}$$

where $\alpha_{u,t} \in \mathbb{R}$ is a positive step-size parameter. Starting from Equation 3, the gradient can be written:

$$\begin{aligned}\nabla_\mathbf{u} J_\gamma(\mathbf{u}) &= \nabla_\mathbf{u} \left[ \sum_{s \in \mathcal{S}} d^b(s) \sum_{a \in \mathcal{A}} \pi(a|s) Q^{\pi,\gamma}(s,a) \right] \\ &= \sum_{s \in \mathcal{S}} d^b(s) \sum_{a \in \mathcal{A}} [\nabla_\mathbf{u} \pi(a|s) Q^{\pi,\gamma}(s,a) \\ &\qquad + \pi(a|s) \nabla_\mathbf{u} Q^{\pi,\gamma}(s,a)\ ]\end{aligned}$$

The final term in this equation, $\nabla_\mathbf{u} Q^{\pi,\gamma}(s,a)$, is difficult to estimate in an incremental off-policy setting. The first approximation involved in the theory of Off-PAC is to omit this term. That is, we work with an approximation to the gradient, which we denote $\mathbf{g}(\mathbf{u}) \in \mathbb{R}^{N_\mathbf{u}}$, defined by

$$\nabla_\mathbf{u} J_\gamma(\mathbf{u}) \approx \mathbf{g}(\mathbf{u}) = \sum_{s \in \mathcal{S}} d^b(s) \sum_{a \in \mathcal{A}} \nabla_\mathbf{u} \pi(a|s) Q^{\pi,\gamma}(s,a) \tag{5}$$

The two theorems below provide justification for this approximation.

**Theorem 1** (Policy Improvement)**.** *Given any policy parameter* $\mathbf{u}$*, let*

$$\mathbf{u}' = \mathbf{u} + \alpha\, \mathbf{g}(\mathbf{u})$$

*Then there exists an* $\epsilon > 0$ *such that, for all positive* $\alpha < \epsilon$*,*

$$J_\gamma(\mathbf{u}') \geq J_\gamma(\mathbf{u})$$

*Further, if* $\pi$ *has a tabular representation (i.e., separate weights for each state), then* $V^{\pi_{\mathbf{u}'},\gamma}(s) \geq V^{\pi_\mathbf{u},\gamma}(s)$ *for all* $s \in \mathcal{S}$*.*

(Proof in Appendix).

In the conventional on-policy theory of policy-gradient methods, the policy-gradient theorem (Marbach & Tsitsiklis, 1998; Sutton et al., 2000) establishes the relationship between the gradient of the objective function and the expected action values. In our notation, that theorem essentially says that our approximation is exact, that $\mathbf{g}(\mathbf{u}) = \nabla_\mathbf{u} J_\gamma(\mathbf{u})$. Although, we can not show this in the off-policy case, we can establish a relationship between the solutions found using the true and approximate gradient:

**Theorem 2** (Off-Policy Policy-Gradient Theorem)**.** *Given* $\mathcal{U} \subset \mathbb{R}^{N_\mathbf{u}}$ *a non-empty, compact set, let*

$$\begin{aligned}\tilde{\mathcal{Z}} &= \{\mathbf{u} \in \mathcal{U} \mid \mathbf{g}(\mathbf{u}) = 0\} \\ \mathcal{Z} &= \{\mathbf{u} \in \mathcal{U} \mid \nabla_\mathbf{u} J_\gamma(\mathbf{u}) = 0\}\end{aligned}$$

*where* $\mathcal{Z}$ *is the true set of local maxima and* $\tilde{\mathcal{Z}}$ *the set of local maxima obtained from using the approximate gradient,* $\mathbf{g}(\mathbf{u})$*. If the value function can be represented by our function class, then* $\mathcal{Z} \subset \tilde{\mathcal{Z}}$*. Moreover, if we use a tabular representation for* $\pi$*, then* $\mathcal{Z} = \tilde{\mathcal{Z}}$*.*

(Proof in Appendix).

The proof of Theorem 2, showing that $\mathcal{Z} = \tilde{\mathcal{Z}}$, requires tabular $\pi$ to avoid update overlap: updates to a single parameter influence the action probabilities for only one state. Consequently, both parts of the gradient (one part with the gradient of the policy function and the other with the gradient of the action-value function) locally greedily change the action probabilities for only that one state. Extrapolating from this result, in practice, more generally a local representation for $\pi$ will likely suffice, where parameter updates influence only a small number of states. Similarly, in the non-tabular case, the claim will likely hold if $\gamma$ is small (the return is myopic), again because changes to the policy mostly affect the action-value function locally.

Fortunately, from an optimization perspective, for all $\mathbf{u} \in \tilde{\mathcal{Z}} \backslash \mathcal{Z}$, $J_\gamma(\mathbf{u}) < \min_{\mathbf{u}' \in \mathcal{Z}} J_\gamma(\mathbf{u}')$, in other words,

$\mathcal{Z}$ represents all the largest local maxima in $\tilde{\mathcal{Z}}$ with respect to the objective, $J_\gamma$. Local optimization techniques, like random restarts, should help ensure that we converge to larger maxima and so to $\mathbf{u} \in \mathcal{Z}$. Even with the true gradient, these approaches would be incorporated into learning because our objective, $J_\gamma$, is non-convex.

### 2.3. The Actor: Incremental Update Algorithm with Eligibility Traces

We now derive an incremental update algorithm using observations sampled from the behavior policy. First, we rewrite Equation 5 as an expectation:

$$\begin{aligned}
\mathbf{g}(\mathbf{u}) &= \mathrm{E}\left[\sum_{a\in\mathcal{A}} \nabla_{\mathbf{u}}\pi(a|s)Q^{\pi,\gamma}(s,a)\middle| s\sim d^b\right] \\
&= \mathrm{E}\left[\sum_{a\in\mathcal{A}} b(a|s)\frac{\pi(a|s)}{b(a|s)}\frac{\nabla_{\mathbf{u}}\pi(a|s)}{\pi(a|s)}Q^{\pi,\gamma}(s,a)\middle| s\sim d^b\right] \\
&= \mathrm{E}\left[\rho(s,a)\psi(s,a)Q^{\pi,\gamma}(s,a)\middle| s\sim d^b, a\sim b(\cdot|s)\right] \\
&= \mathrm{E}_b\left[\rho(s_t,a_t)\psi(s_t,a_t)Q^{\pi,\gamma}(s_t,a_t)\right]
\end{aligned}$$

where $\rho(s,a) = \frac{\pi(a|s)}{b(a|s)}$, $\psi(s,a) = \frac{\nabla_{\mathbf{u}}\pi(a|s)}{\pi(a|s)}$, and we introduce the new notation $\mathrm{E}_b[\cdot]$ to denote the expectation implicitly conditional on all the random variables (indexed by time step) being drawn from their limiting stationary distribution under the behavior policy. A standard result (e.g., see Sutton et al., 2000) is that an arbitrary function of state can be introduced into these equations as a baseline without changing the expected value. We use the approximate state-value function provided by the critic, $\hat{V}$, in this way:

$$\mathbf{g}(\mathbf{u}) = \mathrm{E}_b\left[\rho(s_t,a_t)\psi(s_t,a_t)\left(Q^{\pi,\gamma}(s_t,a_t) - \hat{V}(s_t)\right)\right]$$

The next step is to replace the action value, $Q^{\pi,\gamma}(s_t,a_t)$, by the off-policy $\lambda$-return. Because these are not exactly equal, this step introduces a further approximation:

$$\mathbf{g}(\mathbf{u}) \approx \widehat{\mathbf{g}(\mathbf{u})} = \mathrm{E}_b\left[\rho(s_t,a_t)\psi(s_t,a_t)\left(R_t^\lambda - \hat{V}(s_t)\right)\right]$$

where the off-policy $\lambda$-return is defined by:

$$\begin{aligned}
R_t^\lambda = r_{t+1} &+ (1-\lambda)\gamma(s_{t+1})\hat{V}(s_{t+1}) \\
&+ \lambda\gamma(s_{t+1})\rho(s_{t+1},a_{t+1})R_{t+1}^\lambda
\end{aligned}$$

Finally, based on this equation, we can write the forward view of Off-PAC:

$$\mathbf{u}_{t+1} - \mathbf{u}_t = \alpha_{u,t}\rho(s_t,a_t)\psi(s_t,a_t)\left(R_t^\lambda - \hat{V}(s_t)\right)$$

The forward view is useful for understanding and analyzing algorithms, but for a mechanistic implementation it must be converted to a backward view that does not involve the $\lambda$-return. The key step, proved in the appendix, is the observation that

$$\mathrm{E}_b\left[\rho(s_t,a_t)\psi(s_t,a_t)\left(R_t^\lambda - \hat{V}(s_t)\right)\right] = \mathrm{E}_b\left[\delta_t\mathbf{e}_t\right] \quad (6)$$

where $\delta_t = r_{t+1} + \gamma(s_{t+1})\hat{V}(s_{t+1}) - \hat{V}(s_t)$ is the conventional temporal difference error, and $\mathbf{e}_t \in \mathbb{R}^{N_{\mathbf{u}}}$ is the eligibility trace of $\psi$, updated by:

$$\mathbf{e}_t = \rho(s_t,a_t)\left(\psi(s_t,a_t) + \lambda\mathbf{e}_{t-1}\right)$$

Finally, combining the three previous equations, the backward view of the actor update can be written simply as:

$$\mathbf{u}_{t+1} - \mathbf{u}_t = \alpha_{u,t}\delta_t\mathbf{e}_t$$

The complete Off-PAC algorithm is given above as Algorithm 1. Note that although the algorithm is written in terms of states $s$ and $s'$, it really only ever needs access to the corresponding feature vectors, $\mathbf{x}_s$ and $\mathbf{x}_{s'}$, and to the behavior policy probabilities, $b(\cdot|s)$, for the current state. All of these are typically available in large-scale applications with function approximation. Also note that Off-PAC is fully incremental and has per-time step computation and memory complexity that is linear in the number of weights, $N_{\mathbf{u}} + N_{\mathbf{v}}$.

With discrete actions, a common policy distribution is the Gibbs distribution, which uses a linear combination of features $\pi(a|s) = \frac{e^{\mathbf{u}^\mathsf{T}\phi_{s,a}}}{\sum_b e^{\mathbf{u}^\mathsf{T}\phi_{s,b}}}$ where $\phi_{s,a}$ are state-action features for state $s$, action $a$, and where $\psi(s,a) = \frac{\nabla_{\mathbf{u}}\pi(a|s)}{\pi(a|s)} = \phi_{s,a} - \sum_b \pi(b|s)\phi_{s,b}$. The state-action features, $\phi_{s,a}$, are potentially unrelated to the feature vectors $\mathbf{x}_s$ used in the critic.

**Algorithm 1** The Off-PAC algorithm

Initialize the vectors $\mathbf{e}_v$, $\mathbf{e}_u$, and $\mathbf{w}$ to zero
Initialize the vectors $\mathbf{v}$ and $\mathbf{u}$ arbitrarily
Initialize the state $s$
For each step:
  Choose an action, $a$, according to $b(\cdot|s)$
  Observe resultant reward, $r$, and next state, $s'$
  $\delta \leftarrow r + \gamma(s')\mathbf{v}^\mathsf{T}\mathbf{x}_{s'} - \mathbf{v}^\mathsf{T}\mathbf{x}_s$
  $\rho \leftarrow \pi_{\mathbf{u}}(a|s)/b(a|s)$
  Update the critic (GTD($\lambda$) algorithm):
    $\mathbf{e}_v \leftarrow \rho\left(\mathbf{x}_s + \gamma(s)\lambda\mathbf{e}_v\right)$
    $\mathbf{v} \leftarrow \mathbf{v} + \alpha_v\left[\delta\mathbf{e}_v - \gamma(s')(1-\lambda)(\mathbf{w}^\mathsf{T}\mathbf{e}_v)\mathbf{x}_s\right]$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha_w\left[\delta\mathbf{e}_v - (\mathbf{w}^\mathsf{T}\mathbf{x}_s)\mathbf{x}_s\right]$
  Update the actor:
    $\mathbf{e}_u \leftarrow \rho\left[\frac{\nabla_{\mathbf{u}}\pi_{\mathbf{u}}(a|s)}{\pi_{\mathbf{u}}(a|s)} + \gamma(s)\lambda\mathbf{e}_u\right]$
    $\mathbf{u} \leftarrow \mathbf{u} + \alpha_u\delta\mathbf{e}_u$
  $s \leftarrow s'$

## 3. Convergence Analysis

Our algorithm has the same recursive stochastic form as the off-policy value-function algorithms

$$\mathbf{u}_{t+1} = \mathbf{u}_t + \alpha_t(h(\mathbf{u}_t, \mathbf{v}_t) + M_{t+1})$$

where $h : \mathbb{R}^N \to \mathbb{R}^N$ is a differentiable function and $\{M_t\}_{t\geq 0}$ is a noise sequence. Following previous off-policy gradient proofs (Maei, 2011), we study the behavior of the ordinary differential equation

$$\dot{\mathbf{u}}(t) = \mathbf{u}(h(\mathbf{u}(t), \mathbf{v}))$$

The two updates (for the actor and for the critic) are not independent on each time step; we analyze two separate ODEs using a two timescale analysis (Borkar, 2008). The actor update is analyzed given fixed critic parameters, and vice versa, iteratively (until convergence). We make the following assumptions.

(A1) The policy viewed as a function of $\mathbf{u}$, $\pi_{(\cdot)}(a|s) : \mathbb{R}^{N_\mathbf{u}} \to (0, 1]$, is continuously differentiable, $\forall s \in \mathcal{S}, a \in \mathcal{A}$.

(A2) The update on $\mathbf{u}_t$ includes a projection operator, $\Gamma : \mathbb{R}^{N_\mathbf{u}} \to \mathbb{R}^{N_\mathbf{u}}$, that projects any $\mathbf{u}$ to a compact set $\mathcal{U} = \{\mathbf{u} \mid q_i(\mathbf{u}) \leq 0, i = 1, \ldots, s\} \subset \mathbb{R}^{N_\mathbf{u}}$, where $q_i(\cdot) : \mathbb{R}^{N_\mathbf{u}} \to \mathbb{R}$ are continuously differentiable functions specifying the constraints of the compact region. For $\mathbf{u}$ on the boundary of $\mathcal{U}$, the gradients of the active $q_i$ are linearly independent. Assume the compact region is large enough to contain at least one (local) maximum of $J_\gamma$.

(A3) The behavior policy has a minimum positive value $b_{\min} \in (0, 1]$: $b(a|s) \geq b_{\min}\ \forall s \in \mathcal{S}, a \in \mathcal{A}$

(A4) The sequence $(\mathbf{x}_t, \mathbf{x}_{t+1}, r_{t+1})_{t\geq 0}$ is i.i.d. and has uniformly bounded second moments.

(A5) For every $\mathbf{u} \in \mathcal{U}$ (the compact region to which $\mathbf{u}$ is projected), $V^{\pi,\gamma} : S \to \mathbb{R}$ is bounded.

**Remark 1:** It is difficult to prove the boundedness of the iterates without the projection operator. Since we have a bounded function (with range $(0, 1]$), we could instead assume that the gradient goes to zero exponentially as $\mathbf{u} \to \infty$, ensuring boundedness. Previous work, however, has illustrated that the stochasticity in practice makes convergence to an unstable equilibrium unlikely (Pemantle, 1990); therefore, we avoid restrictions on the policy function and do not include the projection in our algorithm

Finally, we have the following (standard) assumptions on features and step-sizes.

(P1) $||\mathbf{x}_t||_\infty < \infty,\ \forall t$, where $\mathbf{x}_t \in \mathbb{R}^{N_\mathbf{v}}$

(P2) Matrices $C = E[\mathbf{x}_t \mathbf{x}_t^\mathsf{T}]$, $A = E[\mathbf{x}_t(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\mathsf{T}]$ are non-singular and uniformly bounded. $A$, $C$ and $E[r_{t+1}\mathbf{x}_t]$ are well-defined because the distribution of $(\mathbf{x}_t, \mathbf{x}_{t+1}, r_{t+1})$ does not depend on $t$.

(S1) $\alpha_{v,t}, \alpha_{w,t}, \alpha_{u,t} > 0,\ \forall t$ are deterministic such that $\sum_t \alpha_{v,t} = \sum_t \alpha_{w,t} = \sum_t \alpha_{u,t} = \infty$ and $\sum_t \alpha_{v,t}^2 < \infty$, $\sum_t \alpha_{w,t}^2 < \infty$ and $\sum_t \alpha_{u,t}^2 < \infty$ with $\frac{\alpha_{u,t}}{\alpha_{v,t}} \to 0$.

(S2) Define $H(A) \doteq (A + A^\mathsf{T})/2$ and let $\lambda_{\min}(C^{-1}H(A))$ be the minimum eigenvalue of the matrix $C^{-1}H(A)$[1]. Then $\alpha_{w,t} = \eta\alpha_{v,t}$ for some $\eta > \max(0, -\lambda_{\min}(C^{-1}H(A)))$.

**Remark 2:** The assumption $\alpha_{u,t}/\alpha_{v,t} \to 0$ in (S1) states that the actor step-sizes go to zero at a faster rate than the value function step-sizes: the actor update moves on a slower timescale than the critic update (which changes more from its larger step sizes). This timescale is desirable because we effectively want a converged value function estimate for the current policy weights, $\mathbf{u}_t$. Examples of suitable step sizes are $\alpha_{v,t} = \frac{1}{t}$, $\alpha_{u,t} = \frac{1}{1+t\log t}$ or $\alpha_{v,t} = \frac{1}{t^{2/3}}$, $\alpha_{u,t} = \frac{1}{t}$. (with $\alpha_{w,t} = \eta\alpha_{v,t}$ for $\eta$ satisfying (S2)).

The above assumptions are actually quite unrestrictive. Most algorithms inherently assume bounded features with bounded value functions for all policies; unbounded values trivially result in unbounded value function weights. Common policy distributions are smooth, making $\pi(a|s)$ continuously differentiable in $\mathbf{u}$. The least practical assumption is that the tuples $(\mathbf{x}_t, \mathbf{x}_{t+1}, r_{t+1})$ are i.i.d., in other words, Martingale noise instead of Markov noise. For Markov noise, our proof as well as the proofs for GTD($\lambda$) and GQ($\lambda$), require Borkar's (2008) two-timescale theory to be extended to Markov noise (which is outside the scope of this paper). Finally, the proof for Theorem 3 assumes $\lambda = 0$, but should extend to $\lambda > 0$ similarly to GTD($\lambda$) (see Maei, 2011, Section 7.4, for convergence remarks).

We give a proof sketch of the following convergence theorem, with the full proof in the appendix.

**Theorem 3** (Convergence of Off-PAC)**.** *Let $\lambda = 0$ and consider the Off-PAC iterations with GTD(0)[2] for the critic. Assume that (A1)-(A5), (P1)-(P2) and (S1)-(S2) hold. Then the policy weights, $\mathbf{u}_t$, converge to $\hat{\mathcal{Z}} = \{u \in \mathcal{U} \mid \widehat{\mathbf{g}(\mathbf{u})} = 0\}$ and the value function weights, $\mathbf{v}_t$, converge to the corresponding TD-solution with probability one.*

**Proof Sketch:** We follow a similar outline to the two timescale analysis for on-policy policy gradient

[1] Minimum exists as all eigenvalues real-valued (Lemma 4)

[2] GTD(0) is GTD($\lambda$) with $\lambda = 0$, not the different algorithm called GTD(0) by Sutton, Szepesvari & Maei (2008)

actor-critic (Bhatnagar et al., 2009) and for nonlinear GTD (Maei et al., 2009). We analyze the dynamics for our two weights, $\mathbf{u}_t$ and $\mathbf{z}_t{}^\mathsf{T} = (\mathbf{w}_t{}^\mathsf{T}\mathbf{v}_t{}^\mathsf{T})$, based on our update rules. The proof involves satisfying seven requirements from Borkar (2008, p. 64) to ensure convergence to an asymptotically stable equilibrium. ∎

## 4. Empirical Results

This section compares the performance of Off-PAC to three other off-policy algorithms with linear memory and computational complexity: 1) Q($\lambda$) (called Q-Learning when $\lambda = 0$), 2) Greedy-GQ (GQ($\lambda$) with a greedy target policy), and 3) Softmax-GQ (GQ($\lambda$) with a Softmax target policy). The policy in Off-PAC is a Gibbs distribution as defined in section 2.3.

We used three benchmarks: mountain car, a pendulum problem and a continuous grid world. These problems all have a discrete action space and a continuous state space, for which we use function approximation. The behavior policy is a uniform distribution over all the possible actions in the problem for each time step. Note that Q($\lambda$) may not be stable in this setting (Baird, 1995), unlike all the other algorithms.

The goal of the mountain car problem (see Sutton & Barto, 1998) is to drive an underpowered car to the top of a hill. The state of the system is composed of the current position of the car (in $[-1.2, 0.6]$) and its velocity (in $[-.07, .07]$). The car was initialized with a position of -0.5 and a velocity of 0. Actions are a throttle of $\{-1, 0, 1\}$. The reward at each time step is $-1$. An episode ends when the car reaches the top of the hill on the right or after 5,000 time steps.

The second problem is a pendulum problem (Doya, 2000). The state of the system consists of the angle (in radians) and the angular velocity (in $[-78.54, 78.54]$) of the pendulum. Actions, the torque applied to the base, are $\{-2, 0, 2\}$. The reward is the cosine of the angle of the pendulum with respect to its fixed base. The pendulum is initialized with an angle and an angular velocity of 0 (i.e., stopped in a horizontal position). An episode ends after 5,000 time steps.

For the pendulum problem, it is unlikely that the behavior policy will explore the optimal region where the pendulum is maintained in a vertical position. Consequently, this experiment illustrates which algorithms make best use of limited behavior samples.

The last problem is a continuous grid-world. The state is a 2-dimensional position in $[0, 1]^2$. The actions are the pairs $\{(0.0, 0.0), (-.05, 0.0), (.05, 0.0), (0.0, -.05), (0.0, .05)\}$, representing moves in both dimensions. Uniform noise in $[-.025, .025]$ is added to each action component. The reward at each time step for arriving in a position $(p_x, p_y)$ is defined as: $-1 + -2(\mathcal{N}(p_x, .3, .1) \cdot \mathcal{N}(p_y, .6, .03) + \mathcal{N}(p_x, .4, .03) \cdot \mathcal{N}(p_y, .5, .1) + \mathcal{N}(p_x, .8, .03) \cdot \mathcal{N}(p_y, .9, .1))$ where $\mathcal{N}(p, \mu, \sigma) = e^{-\frac{(p-\mu)^2}{2\sigma^2}} / \sigma\sqrt{2\pi}$. The start position is $(0.2, 0.4)$ and the goal position is $(1.0, 1.0)$. An episode ends when the goal is reached, that is when the distance from the current position to the goal is less than 0.1 (using the L1-norm), or after 5,000 time steps. Figure 1 shows a representation of the problem.

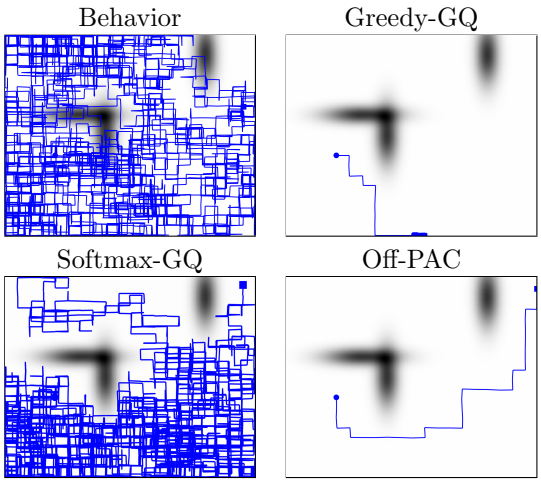
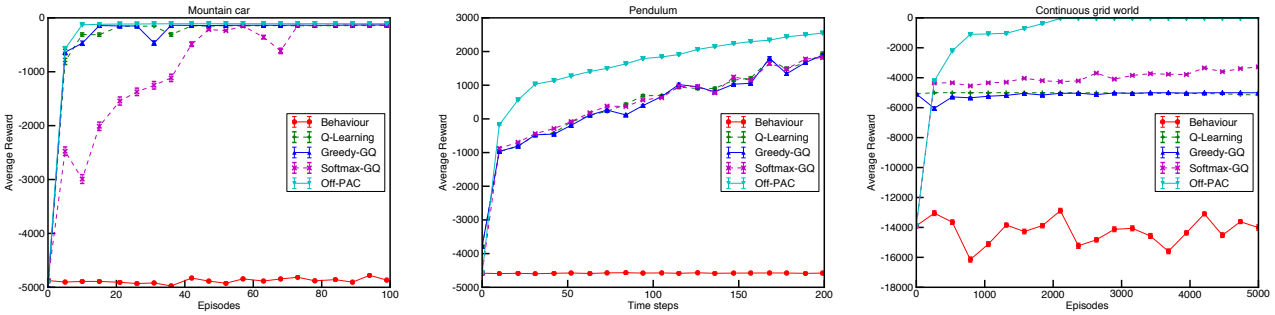



*Figure 1.* Example of one trajectory for each algorithm in the continuous 2D grid world environment after 5,000 learning episodes from the behavior policy. Off-PAC is the only algorithm that learned to reach the goal reliably.

The feature vectors $\mathbf{x}_s$ were binary vectors constructed according to the standard tile-coding technique (Sutton & Barto, 1998). For all problems, we used ten tilings, each of roughly $10 \times 10$ over the joint space of the two state variables, then hashed to a vector of dimension $10^6$. An addition feature was added that was always 1. State-action features, $\psi_{s,a}$, were also $10^6 + 1$ dimensional vectors constructed by also hashing the actions. We used a constant $\gamma = 0.99$. All the weight vectors were initialized to 0. We performed a parameter sweep to select the following parameters: 1) the step size $\alpha_v$ for Q($\lambda$), 2) the step-sizes $\alpha_v$ and $\alpha_w$ for the two vectors in Greedy-GQ, 3) $\alpha_v$, $\alpha_w$ and the temperature $\tau$ of the target policy distribution for Softmax-GQ and 4) the step sizes $\alpha_v$, $\alpha_w$ and $\alpha_u$ for Off-PAC. For the step sizes, the sweep was done over the following values: $\{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, \ldots, .5, 1.\}$ divided by 10+1=11, that is the number of tilings plus 1. To compare TD methods to gradient-TD methods, we also used $\alpha_w = 0$. The temperature parameter, $\tau$, was chosen from $\{.01, .05, .1, .5, 1, 5, 10, 50, 100\}$ and $\lambda$ from $\{0, .2, .4, .6, .8, .99\}$. We ran thirty runs




| | | Mountain car | | | | | Pendulum | | | | | Continuous grid world | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\alpha_w$ | $\alpha_u, \tau$ | $\alpha_v$ | $\lambda$ | Reward | $\alpha_w$ | $\alpha_u, \tau$ | $\alpha_v$ | $\lambda$ | Reward | $\alpha_w$ | $\alpha_u, \tau$ | $\alpha_v$ | $\lambda$ | Reward |
| Behavior: | final | *na* | *na* | *na* | *na* | $-4822 \pm 6$ | *na* | *na* | *na* | *na* | $-4582 \pm 0$ | *na* | *na* | *na* | *na* | $-13814 \pm 127$ |
| | overall | *na* | *na* | *na* | *na* | $-4880 \pm 2$ | *na* | *na* | *na* | *na* | $-4580 \pm .3$ | *na* | *na* | *na* | *na* | $-14237 \pm 33$ |
| Q($\lambda$): | final | *na* | *na* | .1 | .6 | $-143 \pm .4$ | *na* | *na* | .5 | .99 | $1802 \pm 35$ | *na* | *na* | .0001 | 0 | $-5138 \pm .4$ |
| | overall | *na* | *na* | .1 | 0 | $-442 \pm 4$ | *na* | *na* | .5 | .99 | $376 \pm 15$ | *na* | *na* | .0001 | 0 | $-5034 \pm .2$ |
| Greedy-GQ: | final | .0001 | *na* | .1 | .4 | $-131.9 \pm .4$ | 0 | *na* | .5 | .4 | $1782 \pm 31$ | 0.05 | *na* | 1.0 | .2 | $-5002 \pm .2$ |
| | overall | .0001 | *na* | .1 | .2 | $-434 \pm 4$ | .0001 | *na* | .01 | .4 | $785 \pm 11$ | 0 | *na* | .0001 | 0 | $-5034 \pm .2$ |
| Softmax-GQ: | final | .0005 | .1 | .1 | .4 | $-133.4 \pm .4$ | 0 | .1 | .5 | .4 | $1789 \pm 32$ | .1 | 50 | .5 | .6 | $-3332 \pm 20$ |
| | overall | .0001 | .1 | .05 | .2 | $-470 \pm 7$ | .0001 | .05 | .005 | .6 | $620 \pm 11$ | .1 | 50 | .5 | .6 | $-4450 \pm 11$ |
| Off-PAC: | final | .0001 | 1.0 | .05 | 0 | $-108.6 \pm .04$ | .005 | .5 | .5 | 0 | $2521 \pm 17$ | 0 | .001 | .1 | .4 | $-37 \pm .01$ |
| | overall | .001 | 1.0 | .5 | 0 | $-356 \pm .4$ | 0 | .5 | .5 | 0 | $1432 \pm 10$ | 0 | .001 | .005 | .6 | $-1003 \pm 6$ |

*Figure 2.* Performance of Off-PAC compared to the performance of Q($\lambda$), Greedy-GQ, and Softmax-GQ when learning off-policy from a random behavior policy. Final performance selected the parameters for the best performance for the last 10% of the run, whereas the overall performance was over all the runs. The plots on the top show the learning curve for the best parameters for the final performance. Off-PAC had always the best performance and was the only algorithm able to learn to reach the goal reliably in the continuous grid world. Performance is indicated with the standard error.

with each setting of the parameters.

For each parameter combination, the learning algorithm updates a target policy online from the data generated by the behavior policy. For all the problems, the target policy was evaluated at 20 points in time during the run by running it 5 times on another instance of the problem. The target policy was *not* updated during evaluation, ensuring that it was learned only with data from the behavior policy.

Figure 2 shows results on three problems. Softmax-GQ and Off-PAC improved their policy compared to the behavior policy on all problems, while the improvements for Q($\lambda$) and Greedy-GQ is limited on the continuous grid world. Off-PAC performed best on all problems. On the continuous grid world, Off-PAC was the only algorithm able to learn a policy that reliably found the goal after 5,000 episodes (see Figure 1). On all problems, Off-PAC had the lowest standard error.

## 5. Discussion

Off-PAC, like other two-timescale update algorithms, can be sensitive to parameter choices, particularly the step-sizes. Off-PAC has four parameters: $\lambda$ and the three step sizes, $\alpha_v$ and $\alpha_w$ for the critic and $\alpha_u$ for the actor. In practice, the following procedure can be used to set these parameters. The value of $\lambda$, as with other algorithms, will depend on the problem and it is often better to start with low values (less than .4). A common heuristic is to set $\alpha_v$ to 0.1 divided by the norm of the feature vector, $\mathbf{x}_s$, while keeping the value of $\alpha_w$ low. Once GTD($\lambda$) is stable learning the value function with $\alpha_u = 0$, $\alpha_u$ can be increased so that the policy of the actor can be improved. This corroborates the requirements in the proof, where the step-sizes should be chosen so that the slow update (the actor) is not changing as quickly as the fast inner update to the value function weights (the critic).

As mentioned by Borkar (2008, p. 75), another scheme that works well in practice is to use the restrictions on the step-sizes in the proof and to also subsample updates for the slow update. Subsampling updates means only updating every $\{tN, t \geq 0\}$, for some $N > 1$: the actor is fixed in-between $tN$ and $(t+1)N$ while the critic is being updated. This further slows the actor update and enables an improved value function estimate for the current policy, $\pi$.

In this work, we did not explore incremental *natural*

actor-critic methods (Bhatnagar et al., 2009), which use the *natural gradient* as opposed to the conventional gradient. The extension to off-policy natural actor-critic should be straightforward, involving only a small modification to the update and analysis of this new dynamical system (which will have similar properties to the original update).

Finally, as pointed out by Precup et al. (2006), off-policy updates can be more noisy compared to on-policy learning. The results in this paper suggest that Off-PAC is more robust to such noise because it has lower variance than the action-value based methods. Consequently, we think Off-PAC is a promising direction for extending off-policy learning to a more general setting such as continuous action spaces.

## 6. Conclusion

This paper proposed a new algorithm for learning control off-policy, called Off-PAC (Off-Policy Actor-Critic). We proved that Off-PAC converges in a standard off-policy setting. We provided one of the first empirical evaluations of off-policy control with the new gradient-TD methods and showed that Off-PAC has the best final performance on three benchmark problems and consistently has the lowest standard error. Overall, Off-PAC is a significant step toward robust off-policy control.

## 7. Acknowledgments

This work was supported by MPrime, the Alberta Innovates Centre for Machine Learning, the Glenrose Rehabilitation Hospital Foundation, Alberta Innovates—Technology Futures, NSERC and the ANR MACSi project. Computational time was provided by Westgrid and the Mésocentre de Calcul Intensif Aquitain.

**Appendix: See http://arXiv.org/abs/1205.4839**

### References

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30–37. Morgan Kaufmann.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., Lee, M. (2009). Natural actor-critic algorithms. *Automatica 45*(11):2471–2482.

Borkar, V. S. (2008). *Stochastic approximation: A dynamical systems viewpoint*. Cambridge Univ Press.

Bradtke, S. J., Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning 22*:33–57.

Delp, M. (2010). Experiments in off-policy reinforcement learning with the GQ($\lambda$) algorithm. Masters thesis, University of Alberta.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural computation 12*:219–245.

Lagoudakis, M., Parr, R. (2003). Least squares policy iteration. *Journal of Machine Learning Research 4*:1107–1149.

Maei, H. R., Sutton, R. S. (2010). GQ($\lambda$): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the Third Conf. on Artificial General Intelligence*.

Maei, H. R. (2011). *Gradient Temporal-Difference Learning Algorithms*. PhD thesis, University of Alberta.

Maei, H. R., Szepesvári, C., Bhatnagar, S., Precup, D., Silver, D., Sutton, R. S. (2009). Convergent temporal-difference learning with arbitrary smooth function approximation. *Advances in Neural Information Processing Systems 22*:1204–1212.

Maei, H. R., Szepesvári, C., Bhatnagar, S., Sutton, R. S. (2010). Toward off-policy learning control with function approximation. *Proceedings of the 27th International Conference on Machine Learning*.

Marbach, P., Tsitsiklis, J. N. (1998). Simulation-based optimization of Markov reward processes. Technical report LIDS-P-2411.

Pemantle, R. (1990). Nonconvergence to unstable points in urn models and stochastic approximations. *The Annals of Probability 18*(2):698–712.

Peters, J., Schaal, S. (2008). Natural actor-critic. *Neurocomputing 71*(7):1180–1190.

Precup, D., Sutton, R.S., Paduraru, C., Koop, A., Singh, S. (2006). Off-policy learning with recognizers. *Neural Information Processing Systems 18*.

Smart, W.D., Pack Kaelbling, L. (2002). Effective reinforcement learning for mobile robots. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pp. 3404–3410.

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

Sutton, R. S., McAllester, D., Singh, S., Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems 12*.

Sutton, R. S., Szepesvári, Cs., Maei, H. R. (2008). A convergent $O(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems 21*, pp. 1609–1616.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, Cs., Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 993–1000.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*.

Watkins, C. J. C. H., Dayan, P. (1992). Q-learning. *Machine Learning 8*(3):279–292.