Making Policy Gradient Estimators for Softmax Policies More Robust to Non-stationarities

Shivam Garg
Department of Computing Science
University of Alberta
Edmonton, AB
sgarg2@ualberta.ca

Samuele Tosatto
Department of Computing Science
University of Alberta
Edmonton, AB
tosatto@ualberta.ca

Yangchen Pan*
Noah's Ark Lab
Huawei
Edmonton, AB
pan6@ualberta.ca

Martha White†
Department of Computing Science
University of Alberta
Edmonton, AB
whitem@ualberta.ca

A. Rupam Mahmood†
Department of Computing Science
University of Alberta
Edmonton, AB
armahmood@ualberta.ca

Abstract

Policy gradient (PG) estimators are ineffective in dealing with softmax policies that are sub-optimally saturated, which refers to the situation when the policy concentrates its probability mass on sub-optimal actions. Sub-optimal policy saturation may arise from a bad policy initialization or a sudden change, i.e. a non-stationarity, in the environment that occurs after the policy has already converged. Unfortunately, current softmax PG estimators require a large number of updates to overcome policy saturation, which causes low sample efficiency and poor adaptability to new situations. To mitigate this problem, we propose a novel policy gradient estimator, which we call as the *alternate estimator*, for softmax policies. This new estimator utilizes the bias in the critic estimate and the noise present in the reward signal to escape the saturated regions of the policy parameter space. We establish these properties by analyzing this estimator in the tabular bandit setting, and testing it on non-stationary reinforcement learning environments. Our results demonstrate that the alternate estimator is significantly more robust to policy saturation compared to the regular variant, and can be readily adapted to work with different PG algorithms and function approximation schemes.

(The full version of this paper is available at https://arxiv.org/abs/2112.11622.)

Keywords: Policy gradient, softmax policies, policy saturation, non-stationary environments.

Acknowledgements

The authors gratefully acknowledge funding from the Canada CIFAR AI Chairs program, the Reinforcement Learning and Artificial Intelligence (RLAI) laboratory, the Alberta Machine Intelligence Institute (Amii), and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

*Work done while at UofA.
†CIFAR AI Chair, Alberta Machine Intelligence Institute (Amii).

1 Introduction

Policy gradient (PG) algorithms aim to optimize a sequential decision making problem defined on a set of parameterized policies: the policy parameters are optimized using standard stochastic gradient-based update to maximize the net reward received by the agent. PG methods have been successfully deployed in a variety of real world tasks such as robotics (Mahmood, et al., 2018) and large scale simulated problems (Berner et al., 2019). For discrete action tasks, policies are typically represented using a categorical distribution parameterized by a softmax function. However, softmax policies have some inherent issues. Even with access to the true gradients, they can be slow in responding to non-stationarity and their performance heavily depends on the initialization of the policy parameters (Mei et al. 2020). Both these problems arise from an issue, which we call *sub-optimal policy saturation*.

Sub-optimal policy saturation refers to the situation when the policy places a high probability mass on sub-optimal actions. An agent with a saturated policy will not be able to explore other actions and may continue to remain in the sub-optimal region if an appropriate measure is not taken. Sub-optimal policy saturation arises in multiple scenarios: (1) Non-stationarity: in a constantly changing environment, what was once an optimal strategy may no longer work well. (2) Pre-training / transfer learning: deep reinforcement learning (RL) systems are often pre-trained on different tasks before being used on the main task, and the subtle differences in the structure of these tasks might lead to a bad policy initialization which is sub-optimally saturated. And (3) Stochastic updates: PG updates suffer from high variance and therefore, it is possible for a policy to become saturated on sub-optimal actions during the course of learning.

PG methods with softmax policies are particularly susceptible to policy saturation. Entropic regularization (Peters et al., 2010) is a popular approach to mitigate this issue: it works by making the policy more explorative. However, entropic regularization introduces additional terms in the objective, and therefore the resulting optimal policy can be different from the original one. We take a different approach to address this issue. Instead of augmenting the optimization objective with entropy, we introduce a PG estimator that inherently helps to escape the sub-optimally saturated regions.

Our proposed estimator is a simple yet effective approach for dealing with sub-optimally saturated policies thereby making PG algorithms more robust. The classic likelihood ratio estimator for softmax policies, which we call as the *regular estimator*, takes a frustratingly large amount of experience to escape sub-optimally saturated policy regions. The regular estimator produces near zero gradients at saturation as both its expectation and variance, even with reward noise, are vanishingly small at those regions. In contrast, our proposed estimator, which we call the *alternate estimator*, has a non-zero variance in the same scenario that can be utilized to escape the sub-optimal regions. Further, the alternate estimator naturally utilizes the bias in the critic estimate to increase the policy's entropy, thereby encouraging exploration. As the critic estimate improves, this effect reduces, allowing the policy to saturate towards the optimal actions.

2 Preliminaries

In RL, the decision making task is described using a Markov decision process (MD) $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mu, p, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R} \subset \mathbb{R}$ represent the sets of states, actions, and rewards; $\mu \in \Delta(\mathcal{S})$ is the start state distribution; $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$ is the transition dynamics; and $\gamma \in [0, 1]$ is the discount factor. (The object $\Delta(\mathcal{X})$ denotes the set of all possible probability distributions over the set $\mathcal{X}$.) The agent maintains a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that describes its interaction with the environment. This interaction results in the episode $\{S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T\}$, where $S_0 \sim \mu$, $A_t \sim \pi(\cdot|S_t)$, and $S_{t+1}, R_{t+1} \sim p(\cdot, \cdot|S_t, A_t)$ for $t \in \{0, \ldots, T-1\}$ with T being the (possibly random) episode termination length. The agent uses such interactions to learn a policy that maximizes the expected return $\mathcal{J} = \mathbb{E}_\pi[\sum_0^{T-1} \gamma^t R_{t+1}]$. PG methods provide one way to accomplish this task. We now describe these methods in two typical settings.

Gradient Bandits: In bandits the agent picks an action A_t from a discrete action set $\mathcal{A}$ at each timestep, and obtains a reward $R_t \sim p(\cdot|A_t)$. The goal is to maximize the expected immediate reward $\mathcal{J} := \mathbb{E}_\pi[R_t] =: r_\pi$. Gradient bandit algorithms maximize $\mathcal{J}$ using gradient ascent. Given a softmax policy $\pi_{\boldsymbol{\theta}}(a) = e^{\theta_a} / \sum_{b \in \mathcal{A}} e^{\theta_b}$, where θ_a is the action preference corresponding to action a, the gradient $\nabla \mathcal{J}$ (Sutton & Barto, 2018) is given by $[\nabla_{\boldsymbol{\theta}} \mathcal{J}_\pi]_i = r(a_i) \cdot \pi(a_i) \cdot (1 - \pi(a_i))$, where $r(a) := \mathbb{E}[R|a]$ is the reward function. However, the agent usually does not have access to r for all actions at the same time and must resort to using sample based gradient estimators $\hat{\mathbf{g}}$ which satisfy $\nabla_{\boldsymbol{\theta}} \mathcal{J}_\pi = \mathbb{E}_{A \sim \pi; R \sim p(\cdot|A)}[\hat{\mathbf{g}}(A, R)]$. And the typical choice is to use what we call the *regular estimator*

$$[\hat{\mathbf{g}}^{\text{REG}}(A, R)]_a := (R - b)(\mathbb{I}(A = a) - \pi(a)), \tag{1}$$

where $\mathbb{I}$ is the indicator function and b is the agent's estimate of the average reward r_π.

Policy Gradient: In an episodic MDP, the PG objective is $\mathcal{J} = \mathbb{E}_\pi[\sum_0^{T-1} \gamma^t R_{t+1}]$, and the policy gradient is given by

$$\nabla \mathcal{J} = \sum_{s \in \mathcal{S}} \nu_\pi(s) \sum_{a \in \mathcal{A}} \nabla \pi(a|s) q_\pi(s, a). \tag{2}$$

where $\nu_\pi(s) := \sum_{k=0}^{\infty} \gamma^k \mathbb{P}(S_k = s)$ is the state occupancy measure under policy π.

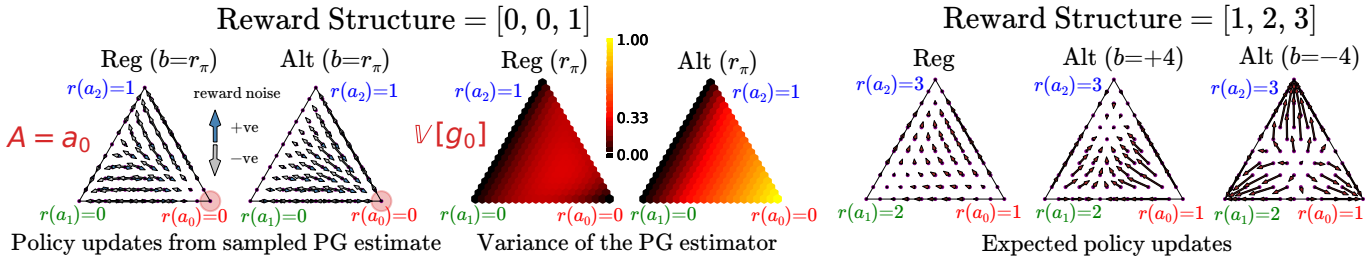

Figure 1: Policy updates and the variance for PG estimators on 3-armed bandits plotted on the probability simplex. The policy-update plots were drawn by updating the policy using the corresponding stochastic PG estimator, assuming that action a_0 was taken and a noise of constant magnitude was added to the reward. The blue and grey arrows correspond to the policy change direction for $+ve$ and $-ve$ reward noises. The variance plots show the heatmap for the zeroth element of the gradient estimator. Last three plots show policy updates using expectation of the estimators.

3 Warmup: Alternate Gradient Bandits

In this section, we derive the alternate PG estimator for the bandit setting using vector notation. For a k-armed bandit problem, let $\boldsymbol{\pi}, \mathbf{r} \in \mathbb{R}^k$ denote the policy and the reward vectors with $[\boldsymbol{\pi}]_k = \pi(a_k)$ and $[\mathbf{r}]_k = r(a_k)$. Using vectors, the expectations of scalar quantities become vector inner products: $\mathcal{J}_\pi \equiv r_\pi = \mathbb{E}_\pi[R] = \sum_a \pi(a) r(a) = \boldsymbol{\pi}^\top \mathbf{r}$. Further, the softmax policy can be written as $\boldsymbol{\pi} = e^{\boldsymbol{\theta}} / (\mathbf{1}^\top e^{\boldsymbol{\theta}})$, where the vector $e^{\boldsymbol{\theta}} \in \mathbb{R}^k$ is defined as $[e^{\boldsymbol{\theta}}]_a := e^{\theta_a}$. And $\nabla_{\boldsymbol{\theta}} \boldsymbol{\pi} = (\mathbf{I} - \boldsymbol{\pi}\, \mathbf{1}^\top) \operatorname{diag}(\boldsymbol{\pi})$. Hence, the regular policy gradient estimator can be derived as follows:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J} = \nabla_{\boldsymbol{\theta}} (\boldsymbol{\pi}^\top \mathbf{r}) = (\mathbf{I} - \boldsymbol{\pi}\, \mathbf{1}^\top) \operatorname{diag}(\boldsymbol{\pi})\, \mathbf{r} = \mathbb{E}_{A \sim \pi}[r(A)(\mathbf{e}_A - \boldsymbol{\pi})] = \mathbb{E}_{A \sim \pi, R \sim p(\cdot|A)}[(R - r_\pi)(\mathbf{e}_A - \boldsymbol{\pi})], \tag{3}$$

where $\mathbf{g}^{\text{REG}}(A, R) := (R - r_\pi)(\mathbf{e}_A - \boldsymbol{\pi})$ is the regular gradient bandit estimator and $\mathbf{e}_A \in \mathbb{R}^k$ represents the basis vector. The approximate version of it, as given before in Eq. 1, is $\hat{\mathbf{g}}^{\text{REG}}(A, R) = (R - b)(\mathbf{e}_A - \boldsymbol{\pi})$, where b is an estimate of r_π. Also note that this approximate estimator remains unbiased, i.e. $\nabla \mathcal{J} = \mathbb{E}[\hat{\mathbf{g}}^{\text{REG}}(A, R)]$.

The Alternate Gradient Bandit Estimator is derived by using an alternate form of the true gradient:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{J} &= (\mathbf{I} - \boldsymbol{\pi}\, \mathbf{1}^\top) \operatorname{diag}(\boldsymbol{\pi})\, \mathbf{r} = \left(\operatorname{diag}(\mathbf{r}) - \boldsymbol{\pi}\, \mathbf{r}^\top\right) \boldsymbol{\pi} = \operatorname{diag}(\mathbf{r})\, \boldsymbol{\pi} - \boldsymbol{\pi}^\top \mathbf{r}\, \boldsymbol{\pi} = \left(\operatorname{diag}(\mathbf{r}) - \boldsymbol{\pi}^\top \mathbf{r}\, \mathbf{I}\right) \boldsymbol{\pi} \\ &= \mathbb{E}_{A \sim \pi, R \sim p(\cdot|A)}[(R - r_\pi)\, \mathbf{e}_A], \end{aligned} \tag{4}$$

where $\mathbf{g}^{\text{ALT}}(A, R) := (R - r_\pi)\, \mathbf{e}_A$ is the *alternate* gradient bandit estimator. Using a baseline b to estimate the average reward r_π, gives the following estimator: $\hat{\mathbf{g}}^{\text{ALT}}(A, R) = (R - b)\, \mathbf{e}_A$.

4 Properties of the Alternate Gradient Bandits Estimator

We now discuss how the alternate estimator utilizes the reward noise and the bias in the critic estimate to escape saturated regions in the policy space.

Alternate Estimator Utilizes Reward Noise: Whenever the policy is saturated, i.e. it places a high probability mass on some actions, the expected gradient becomes close to zero, and consequently the policy weights are not updated. Further, for the regular estimator, the variance at these "corners" of the probability simplex is also zero. Therefore, there is neither any gradient signal nor any noise, and the agent is unable to escape the sub-optimal region. In contrast, the alternate estimator, despite being zero in expectation, has a non-zero variance and therefore the agent can utilize the reward noise to escape the sub-optimal region. We illustrate this point with Figure 1 (left and middle) and an accompanying example:

Example 1. *Consider a 3-armed bandit with $\mathcal{A} = \{a_0, a_1, a_2\}$, the expected reward $\mathbf{r} = [0\ 0\ 1]^\top$, and the policy $\boldsymbol{\pi} = [1\ 0\ 0]^\top$. The rewards are perturbed with a normally distributed noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$, so that upon picking action A, the agent receives the reward $r(A) + \epsilon$. Since $r_\pi = 0$, $R - r_\pi = 0 + \epsilon - 0 = \epsilon$. And because the agent samples $A = a_0$ at each timestep, we obtain*

$$\begin{aligned} \mathbf{g}^{REG}(A, R) &= (R - r_\pi)(\mathbf{e}_A - \boldsymbol{\pi}) = \epsilon \cdot \left([1\ 0\ 0]^\top - [1\ 0\ 0]^\top\right) = [0\ 0\ 0]^\top, \text{ and} \\ \mathbf{g}^{ALT}(A, R) &= (R - r_\pi)\, \mathbf{e}_A = \epsilon \cdot [1\ 0\ 0]^\top = [\epsilon\ 0\ 0]^\top. \end{aligned}$$

This example shows that at a sub-optimal corner, the agent with the alternate estimator effectively does a random walk until it escapes that region and can choose other actions to get a non-zero reward signal. Further, doing a random walk at the optimal corner is not problematic, since there is an attractive gradient signal as soon as the agent moves away from this corner. The following propositions formalize these points.

Proposition 1. *Define $\mathcal{I}_c := \{a \mid r(a) = c\}$ for some constant $c \in \mathbb{R}$. Assume that $\exists c$ such that $\mathcal{I}_c \neq \emptyset$ and that the policy is saturated on the actions in the set $\mathcal{I}_c$: $\sum_{a\in\mathcal{I}_c} \pi(a) = 1$. Then the expected policy gradient for softmax policies is zero: $\nabla_{\boldsymbol{\theta}}\mathcal{J} = \mathbb{E}[\mathbf{g}^{REG}(A,R)] = \mathbb{E}[\hat{\mathbf{g}}^{REG}(A,R)] = \mathbb{E}[\mathbf{g}^{ALT}(A,R)] = \mathbf{0}$.*

Proposition 2. *Let $\sigma(a)^2 := \mathbb{V}[R|A=a]$ be the variance of the reward corresponding to action a. Assume that the policy is saturated on the action c, i.e., $\pi(c) = 1$. Then, the variance of the regular PG estimator (with or without a baseline) is zero: $\mathbb{V}[\mathbf{g}^{REG}(A)] = \mathbb{V}[\hat{\mathbf{g}}^{REG}(A,R)] = \mathbf{0}$. Whereas, the variance of the alternate PG estimator is non-zero: $\mathbb{V}[\mathbf{g}^{ALT}(A,R)] = \sigma(c)^2\,\mathbf{e}_c$.*

Although, the above example and the propositions require the policy to lie at the boundary of the probability simplex which is unsatisfiable for softmax policies, using continuity arguments, we can still reason that for the regular estimator, both the expected gradient and its variance vanish in the proximity of the simplex boundary, whereas the alternate estimator will have non-zero variance. In addition to this, the stochastic update for the alternate estimator is much higher than that for the regular estimator. This can be seen from Figure 1 (left): look at the length of the update arrows near the bottom-right corner on the simplex.

Alternate Estimator Utilizes its Biasedness: The alternate estimator can also utilize the bias in the critic estimate b to escape saturation. To see this, consider a baseline $b \neq r_\pi$. Then the alternate estimator becomes biased: $\nabla_{\boldsymbol{\theta}}\mathcal{J} \neq \mathbb{E}[\hat{\mathbf{g}}^{\text{ALT}}(A,R)] = \mathbb{E}[(R-b)\,\mathbf{e}_A] = \boldsymbol{\pi}\odot(\mathbf{r} - b\mathbf{1})$. Interestingly, this biased update $\mathbb{E}[\hat{\mathbf{g}}^{\text{ALT}}(A,R)]$ is not the gradient of any function. Therefore, in order to understand its behavior, we look at its fixed point and under what conditions it acts as an attractor or a repellor. Figure 1 (right) illustrates this fixed point and its behavior based on how the baseline is initialized for one specific instance of a bandit problem. We now state a theorem that formalizes this property. Let $r_1 \leq r_2 \leq \cdots \leq r_k$ represent the true rewards for the bandit problem. Let $\pi_t = e^{\boldsymbol{\theta}^{(t)}}/\mathbf{1}^\top e^{\boldsymbol{\theta}^{(t)}}$ represent the softmax policy at timestep t. And let the action preference vector $\boldsymbol{\theta}^{(t)}$ be updated using $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \alpha\,\mathbb{E}[\hat{\mathbf{g}}^{\text{ALT}}(A,R)]$ for $\alpha > 0$.

Lemma 2.1. ***Fixed points of the biased gradient bandit update.*** ***(1)*** *If there exists an $n \in \{1,2,\ldots,k-1\}$ such that $r_1 \leq \cdots \leq r_n < b < r_{n+1} \leq \cdots \leq r_k$, then $\mathbb{E}_\pi[\hat{\mathbf{g}}^{ALT}(A,R)]$ is never equal to zero.* ***(2)*** *If $b = r(a)$ for at least one action, then $\mathbb{E}_\pi[\hat{\mathbf{g}}^{ALT}(A,R)] = 0$ at any point on the face of the probability simplex given by $\sum_{a\in\mathcal{I}_b}\pi(a) = 1$ with $\mathcal{I}_b := \{a|r(a) = b\}$.* ***(3)*** *If $b < r_1$ or $b > r_k$, then $\mathbb{E}_\pi[\hat{\mathbf{g}}^{ALT}(A,R)] = 0$ at a point π^* within the simplex boundary given by $\pi^*(a) = \frac{1}{r(a)-b}\left(\sum_{c\in\mathcal{A}}\frac{1}{r(c)-b}\right)^{-1},\ \forall a \in \mathcal{A}$.*

Theorem 1. ***Nature of the fixed point $\boldsymbol{\pi}^*$.*** *Assume that $\pi_t \neq \pi^*$. If the baseline is pessimistic, i.e. $b < r_1$, then for any $\alpha > 0$, the fixed point π^* acts as a repellor: $D_{KL}(\pi^*\|\pi_{t+1}) > D_{KL}(\pi^*\|\pi_t)$, where D_{KL} is the KL-divergence. And if the baseline is optimistic, i.e. $b > r_k$, then given a sufficiently small positive stepsize α the fixed point π^* acts as an attractor: $D_{KL}(\pi^*\|\pi_{t+1}) < D_{KL}(\pi^*\|\pi_t)$.*

The above theorem illustrates that with an optimistic baseline, an agent using the alternate estimator is updated towards a more uniform distribution π^*. Therefore, if the agent were stuck in a sub-optimal corner of the probability simplex, an optimistic baseline would make its policy more uniform and encourage exploration. (On the flip side, with a pessimistically initialized baseline, the alternate estimator can pre-maturely saturate towards a sub-optimal corner; see Figure 1 (right).) And even though π^* is different from the optimal policy, the agent with an alternate estimator can still reach the optimal policy, because as the agent learns and improves its baseline estimate, the alternate PG estimator becomes asymptotically unbiased. And hopefully by this time, the agent has already escaped the saturated policy region.

5 Alternate PG Estimator for MDPs

The alternate gradient estimator can be readily extended to MDPs, where it enjoys properties analogous to the bandit case. For a given state s, let $\boldsymbol{\pi}(\cdot|s) \in \mathbb{R}^{|\mathcal{A}|}$ be the vector with $[\boldsymbol{\pi}(\cdot|s)]_a = \pi(a|s)$. Similarly, define the action value vector $[\mathbf{q}_\pi(s,\cdot)]_a = q_\pi(s,a)$. The softmax policy for MDPs then becomes $\pi(a|s) = e^{[\boldsymbol{\theta}_\mathbf{w}(s)]_a}/\sum_{b\in\mathcal{A}} e^{[\boldsymbol{\theta}_\mathbf{w}(s)]_b}$, where $\boldsymbol{\theta}_\mathbf{w} : \mathcal{S} \to \mathbb{R}^{\mathcal{A}}$ denotes the action preference vector, parameterized by $\mathbf{w}$, and $[\boldsymbol{\theta}_\mathbf{w}(s)]_a$ is its ath element. In practice, $\boldsymbol{\theta}_\mathbf{w}$ could be implemented using, say, a neural network. For brevity, we will drop $\mathbf{w}$, i.e. $\boldsymbol{\theta} \equiv \boldsymbol{\theta}_\mathbf{w}$. To obtain the results that follows, we analytically worked out the gradient of the action likelihood for the softmax policy (similar to what we did for the bandit case; see Eqs. 3 and 4), which gives us

$$\nabla_\mathbf{w}\mathcal{J}_\pi = \mathbb{E}\Big[\underbrace{\nabla_\mathbf{w}\log\pi(A|S)(q_\pi(S,A) - v_\pi(S))}_{=:\mathbf{g}^{\text{REG}}(S,A)}\Big] = \mathbb{E}\Big[\underbrace{\nabla_\mathbf{w}[\boldsymbol{\theta}(S)]_A(q_\pi(S,A) - v_\pi(S))}_{=:\mathbf{g}^{\text{ALT}}(S,A)}\Big]. \tag{5}$$

Note that even though the regular and the alternate PG estimators are equal in expectation, in general they are not equal for an arbitrary state-action pair: $\mathbf{g}^{\text{REG}}(S,A) \neq \mathbf{g}^{\text{ALT}}(S,A)$. Further, it is straightforward to adapt the alternate estimator given in Eq. 5 to work with different PG methods such as REINFORCE, Actor-Critic, TRPO, or PPO.

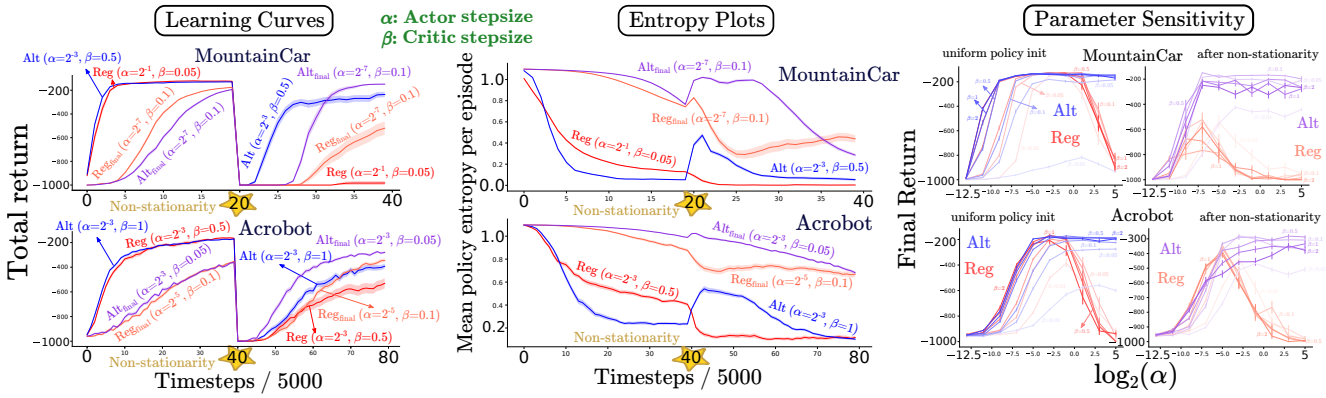

Figure 2: Performance of online Actor-Critic on MountainCar (200k timesteps) and Acrobot (400k timesteps). Learning curves are for the best performing stepsize configurations at the time non-stationarity was introduced and at the final timestep. Entropy plots show entropy of the policy on the exact states encountered during each episode. The sensitivity plots show the mean performance during the last 5000 timesteps. All the results were averaged over 50 runs.

6 Experiments on Non-stationary Environments

In this section, we demonstrate that the alternate estimator is able to effectively handle non-stationarity in an environment, whereas the regular estimator fails to do so. We trained the online Actor-Critic algorithm (Sutton and Barto, 2018) to solve the MountainCar and the Acrobot control tasks, with linear function approximation (using tile-coding). To induce non-stationarity in either task, we switched the `left` and `right` actions after half-time.

Figure 2 (left) shows the learning curves for the best parameter configuration. For each estimator, we selected two sets of parameter values: which performed best right before the non-stationarity hit and another which performed best at the end of the experiment (denoted by a `final` in the subscript). For MountainCar, the alternate estimator is superior to the regular estimator for both sets of stepsizes. Remarkably, the best performing parameter set for regular (red curve) at timestep 100k was unable to recover from the non-stationarity; whereas alternate (blue curve), despite having similar performance as regular at 100k timestep, was able to recover. On Acrobot, the difference in performance is still there but relatively smaller. We attribute the superior performance of the alternate estimator to the bias in the critic estimate. For instance, in MountainCar at 100k timesteps, the critic would have converged to predict a return of about -200. But when the non-stationarity hit, the agent would have started receiving returns much lower than -200. This means that at that time, the critic estimate became optimistic and encouraged exploration by pushing the policy towards a more uniform distribution. Figure 2 (middle) corroborates this point: the policy entropy for alternate (but not for regular) jumps right around the timestep when the non-stationarity hit. Figure 2 (right) shows the parameter sensitivity plots for these tasks.

7 Conclusions

We proposed an alternate policy gradient estimator for softmax policies that, as we demonstrated theoretically and empirically, effectively utilizes the reward noise and the bias in the critic to escape sub-optimally saturated regions in the policy space. Our analysis, conducted on multiple bandit and MDP tasks, suggests that this estimator works well with different PG algorithms and different function approximation schemes. The alternate estimator makes existing PG methods more viable for non-stationary problems, and by extension for many practical real-life control tasks.

References

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R. (2019). Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680.*

Mahmood, A. R., Korenkevych, D., Vasan, G., Ma, W., Bergstra, J. (2018). Benchmarking reinforcement learning algorithms on real-world robots. *Conference on Robot Learning.*

Mei, J., Xiao, C., Dai, B., Li, L., Szepesvári, C., Schuurmans, D. (2020). Escaping the Gravitational Pull of Softmax. *Advances in Neural Information Processing Systems.*

Peters, J., Mulling, K., Altun, Y. (2010). Relative entropy policy search. *AAAI Conference on Artificial Intelligence.*

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