Continuous Tactical Optimism and Pessimism

Kartik Bharadwaj
Robert Bosch Centre for Data Science and Artificial Intelligence
Department of Computer Science and Engineering, Indian Institute of Technology Madras
Chennai, India
cs20s020@smail.iitm.ac.in

Chandrashekar Lakshminarayanan
Robert Bosch Centre for Data Science and Artificial Intelligence
Department of Computer Science and Engineering, Indian Institute of Technology Madras
Chennai, India
chandrashekar@cse.iitm.ac.in

Balaraman Ravindran
Robert Bosch Centre for Data Science and Artificial Intelligence
Department of Computer Science and Engineering, Indian Institute of Technology Madras
Chennai, India
ravi@cse.iitm.ac.in

ABSTRACT

In the field of reinforcement learning for continuous control, deep off-policy actor-critic algorithms have become a popular approach due to their ability to address function approximation errors through the use of pessimistic value updates. However, this pessimism can reduce exploration, which is typically seen as beneficial for learning in uncertain environments. Tactical Optimism and Pessimism (TOP) proposed an actor-critic framework that dynamically adjusts the degree of optimism used in value learning based on the task and learning stage. However, their fixed bandit framework acts as a hyper-parameter for each task. We need to consider two hyperparameters: the number of arms and arm values. To simplify this problem, we consider learning the degree of optimism β while training the agent in the environment. We demonstrate that this approach outperforms other methods that use a fixed level of optimism in a series of continuous control tasks in Walker2d-v2 and HalfCheetah-v2 environments, and can be easily implemented in various off-policy algorithms. We call our algorithm: cTOP or continuous TOP.

KEYWORDS

Reinforcement Learning, Bandits, Optimism, Pessimism, SPSA

1 INTRODUCTION

Reinforcement learning (RL) has been increasingly successful, particularly with the use of deep neural networks for value function approximation. One of the main challenges preventing the widespread use of actor-critic methods [14, 23] in control tasks is their low sample efficiency. Despite recent progress [12, 13], these methods still require millions of interactions with the environment to achieve satisfactory performance on moderately complex control problems. This means that deploying these algorithms can be prohibitively expensive in systems where collecting samples is costly. Another challenge arises due to the use of function approximators that can introduce positive bias in value computation. This can cause an overestimation of the expected reward, which might result in the exploration of states and actions that would not otherwise be explored. However, without a proper understanding of the nature of the overestimation, such exploration can be risky. There are two opposing views on addressing this tension in the literature on RL approaches to continuous-control problems: one seeks to correct the overestimation. At the same time, the other side argues that being optimistic can be helpful in encouraging exploration. Alternatively, RL agents can benefit by varying their optimism and pessimism based on the task they are trying to accomplish. For instance, in the exploration-exploitation trade-off, increasing optimism allows the agent to explore more, as they are more likely to take actions that have not been tried before. In contrast, increasing pessimism makes it more likely for the agent to choose actions that have proven to be effective in the past. Moreover, in some tasks, pessimistic agents are more likely to avoid taking risky actions whose consequences can be severe, while optimistic agents are more risk-seeking likely to take risks in the hope of obtaining a larger reward. Also, varying optimism and pessimism can be helpful in dealing with non-stationary environments, where the true value of states and actions may change over time. By adapting to the changing environment, the agent can maintain good performance and avoid getting stuck in obsolete policies. Tactical Optimism and Pessimism (TOP) [17] hypothesize that the degree of action-value estimation bias and subsequent efficacy of an optimistic strategy depends on the environment, the learning stage, and the overall context in which a learner is embedded. Therefore, they propose to view optimism/pessimism as a spectrum and investigate procedures that actively move along that spectrum during the learning process. They measure two forms of uncertainty that arise during learning: aleatoric uncertainty and epistemic uncertainty, and aim to control their effects. TOP acknowledges the inherent uncertainty in the level of estimation bias present and estimates the optimal approach on the fly by formulating the optimism/pessimism dilemma as a $k-$armed bandit problem. However, deciding arm values and the number of arms depends on each environment, and finding an optimal set of arms becomes more of a hyper-parameter search. In this work, we propose learning the degree of optimism/pessimism while the agent interacts online with the environment.

2 RELATED WORK

The recent success in deep reinforcement learning has been attributed to improvements to off-policy actor-critic algorithms. Specifically, the Deterministic Policy Gradient (DPG) [20] method, which underpins this work, is based on a deterministic policy. The shape of the policy gradient is particularly interesting in that it does not need integration throughout the action space, which means that it may require fewer samples to learn. DDPG [14] combines the deterministic policy gradient [20] with off-policy learning using neural

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

networks architecture based on DQN [16]. This effort leads to an off-policy actor-critic architecture in which the actor's gradients rely solely on derivatives via the trained critic. This indicates that boosting the critic's evaluation immediately improves the actor gradients. C51 [3] has demonstrated that the distribution over returns, the expectation of which is the value function, obeys a distributional Bellman equation. Estimating the distribution was sufficient to attain state-of-the-art scores on the Atari 2600 benchmarks. Notably, this approach provides these benefits by directly enhancing updates for the critic.

Optimistic algorithms are built upon the principle of "optimism in the face of uncertainty" (OFU). They operate by maintaining a set of statistically plausible models of the world and selecting actions to maximize the returns in the best plausible world. Such algorithms were first studied in the context of multi-armed bandit problems [6], and went on to inspire numerous algorithms for reinforcement learning.

The off-policy exploration strategy used in OAC [9] is designed to optimize the upper confidence bound on the critic, which is derived from an epistemic uncertainty estimate on the Q-function obtained using the bootstrap method [18]. Unfortunately, due to the difficulties associated with maintaining low estimation error when using function approximation, attempts to establish an upper bound on the true value function have had limited success.

Recently, there has been increasing evidence in support of the efficacy of adaptive algorithms [19]. Agent57 [1] is the first agent to outperform the human baseline for all 57 games in the Arcade Learning Environment [4]. Based on the game state, Agent57 uses a meta-controller to adaptively adjust its exploration strategies parameterized by exploration rate and discount factor, thereby controlling the *exploration/exploitation trade-off*.

Tactical Optimism and Pessimism (TOP) [17] integrates DPG with distributional value estimate, just like in D4PG [2]. D4PG uses a categorical distribution, whereas TOP uses two critics and a quantile representation. TOP models the trade-off between optimism and pessimism as a discrete multi-armed bandit problem. While optimism can help with exploration if there is a considerable estimating error, a more pessimistic strategy may be required to stabilize learning. Furthermore, both techniques have resulted in algorithms that are backed by substantial empirical data. TOP seeks to unify these seemingly contrary views by postulating that the respective contributions of optimism and pessimism elements can change depending on the environment. Unlike TOP, which uses $k-$armed bandits, our approach is to learn the degree of optimism/pessimism β to actively vary the level of optimism/pessimism in its value estimates.

3 OFF-POLICY REINFORCEMENT LEARNING

Reinforcement learning considers the framework where an agent interacts with its environment with the goal of learning to maximize its cumulative reward. Ideally, an environment is cast as a Markov Decision Process (MDP), formally defined as a tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the space of possible actions, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. For a given policy π, the return $Z_\pi = \sum_t \gamma^t r_t$, is a random variable describing the sum of discounted rewards, starting at state s_t, and observed along one episode obtained by unrolling policy π until some time horizon T, potentially infinite. γ determines the priority of short-term rewards. Given a set of policies parameterized by θ, $\{\pi_\theta : \theta \in \Theta\}$, the goal is to update θ so as to maximize the expected return, or discounted cumulative reward, $J(\theta) = \mathbb{E}_\pi[\sum_t \gamma^t r_t] = \mathbb{E}[Z_\pi]$.

Actor-critic algorithms provide a solution for maximizing the expected cumulative reward in Deep RL by using two separate components: the actor and the critic. The actor, represented by the policy π, is trained to make decisions that maximize the expected return. The critic, in the form of a value function, evaluates the actions of the policy by predicting the expected return under the current policy, $Q_\pi(s, a) \coloneqq \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}[Z_t | s_t = s, a_t = a]$. In continuous control, parameterized policies π_θ are updated by taking the gradient of the expected return $\nabla_\theta J(\theta)$. In the actor-critic framework, the policy can be updated using gradient ascent through the deterministic policy gradient algorithm [20]:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi[\nabla_a Q_\pi(s, a)|_{a=\pi(s)} \nabla_\theta \pi_\theta(s)] \tag{1}$$

In Q-learning, the action-value function can be learned using temporal difference learning [22], [25], an update rule based on the Bellman equation [5]. The Bellman equation is a fundamental relationship between the value of a state-action pair (s, a) and the value of the next state-action pair (s', a'):

$$Q_\pi(s, a) = r + \gamma \mathbb{E}_{s', a'}[Q_\pi(s', a')], a' \sim \pi(s') \tag{2}$$

For large continuous state space, the value can be estimated with a differentiable function approximator $Q_\phi(s, a)$, with parameters ϕ. In deep Q-learning [16], the network is updated by using temporal difference learning with a secondary frozen target network $Q'_\phi(s, a)$ to maintain a fixed target y over multiple updates:

$$y = r + \gamma Q'_\phi(s', a'), a' \sim \pi'_\theta(s'), \tag{3}$$

where actions are selected from a target action network π'_θ. The weights of the target networks are periodically updated to match the weights of the current network by some proportion τ at each time step as:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta' \tag{4}$$

This update can be applied in an off-policy fashion, sampling random mini-batches of transitions from an experience replay buffer [15].

4 TACTICAL OPTIMISM & PESSIMISM (TOP)

When operating in an RL environment online, the agent following policy π tends to overestimate the value of $Q(s, a)$ because of the max operator used in the Q-learning update. In contrast to the method presented in [14], [12] adopts a strategy of training two Q-functions instead of one and chooses the smaller Q-value between the two to determine the target Q-value in the Bellman error loss function. Although it is possible to minimize multiple Q-functions, this approach can increase computational expenses. Instead, the TOP technique relies on adaptive optimism in uncertain situations.

4.1 Modeling uncertainty

TOP represents two types of uncertainty: *aleatoric uncertainty* and *epistemic uncertainty*

Aleatoric Uncertainty is characterized as the inherent randomness of the environment that cannot be explained regardless of the agent's comprehension of the task. To account for this type of uncertainty, TOP employs distributional RL [3, 10, 11], TOP learns the full return distribution $\mathbb{Z}^\pi(s,a)$, for policy π and state-action pair (s,a), instead of the expected return, $Q^\pi(s,a) = \mathbb{E}[Z^\pi(s,a)]$, $Z^\pi(s,a) \sim \mathbb{Z}^\pi(\cdot|s,a)$. $Z^\pi(s,a)$ denotes the return random variable of policy π and state-action (s,a). The distribution $\mathbb{Z}^\pi(s,a)$ captures the aleatoric uncertainty.

Epistemic uncertainty, on the other hand, arises from the agent's lack of knowledge about the environment due to insufficient experience. TOP measures the extent to which an optimistic belief about the return differs from a pessimistic belief. Epistemic uncertainty is modeled as:

$$\mathbb{Z}^\pi(s,a) = \bar{Z}(s,a) + \epsilon\sigma(s,a) \tag{5}$$

Drawing inspiration from [11], TOP represents the return distribution $\mathbb{Z}^\pi(s,a)$ using a *quantile approximation* with k quantiles. Quantiles $q^k(s,a)$ are learned using a deep neural network parameterized by ϕ. Epistemic uncertainty is measured using two quantile functions $q_1^k(s,a)$ and $q_2^k(s,a)$, parameterized by ϕ_1 and ϕ_2 respectively. Values of mean $\bar{Z}(s,a)$ and $\sigma(s,a)$ is approximated as follows:

$$\bar{q}^k(s,a) = \frac{1}{2}(q_1^k(s,a)+q_2^k(s,a)) \quad \bar{\sigma}^k(s,a) = \sqrt{\sum_{i=1}^{2}(q_i^k(s,a) - \bar{q}^k(s,a))^2} \tag{6}$$

4.2 Learning the Critic and Actor

Using the quantile estimates from Equation 6, belief distribution $\tilde{\mathbb{Z}}^\pi(s,a)$ of the random return variable $Z^\pi(s,a)$ is defined as:

$$q_{\tilde{Z}^\pi(s,a)} = q_{\bar{Z}(s,a)} + \beta q_\sigma(s,a) \tag{7}$$

Belief distribution $\tilde{\mathbb{Z}}^\pi(s,a)$ is considered to be optimistic for $\beta \geq 0$ and pessimistic when $\beta < 0$. TOP dynamically adjusts the degree of optimism β during training using Exponential Weighted Forecasting Algorithm [8]. By replacing $\epsilon \sim \mathcal{N}(0,1)$ with β in Equation 5, belief distributions is non-Gaussian.

Critic: TOP uses the quantiles $\tilde{q}^k$ of the belief distribution $\tilde{\mathbb{Z}}^\pi(s,a)$ from Equation 7 as a target for both estimates of $\tilde{\mathbb{Z}}_1^\pi(s,a)$ and $\tilde{\mathbb{Z}}_2^\pi(s,a)$. The temporal difference error for each $\tilde{\mathbb{Z}}_i^\pi(s,a)$ is defined as: $\delta_i^{(j,k)} := r + \gamma\tilde{q}^j - q_i^k \in \{1,2\}$ where (j,k) ranges over all possible quantiles. Similar to QR-DQN [11], we learn the distributional critics by taking the gradient of the Huber loss $\mathcal{L}_{Huber}$ evaluated at each distributional TD error $\delta_i^{(j,k)}$.

Actor: The actor is trained to maximize the expected value $\tilde{Q}(s,a)$ under the belief distribution $\tilde{\mathbb{Z}}_2^\pi(s,a)$. The expected value $\tilde{Q}(s,a)$ can be calculated by taking the average of quantiles $\tilde{q}^k$ for state-action (s,a): $\tilde{Q}(s,a) = \frac{1}{K}\sum_{k=1}^{K}\tilde{q}^k(s,a)$. Then, the actor update follows the standard DPG gradient:

$$\Delta\theta \propto \nabla_a \tilde{Q}(s,a)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s) \tag{8}$$

$$\Delta\phi_i \propto \sum_{1\leq j,k\leq K} \nabla_{\phi_i}\mathcal{L}_{Huber}(\delta_i^{(j,k)}) \tag{9}$$

4.3 Learning the degree of optimism as a $k-$armed bandit problem

It would be beneficial for an agent to vary its degree of optimism or pessimism β as the reward distribution varies across environments. TOP casts the problem of choosing β as a $k-$armed bandit framework to adjust its degree of optimism. Before the start of every episode m, the agent picks arm d_m from a set of D arms, each taking a discrete value $\{\beta_d\}_{d=1}^D$. These arms follow a distribution $\mathbf{p}_m \in \delta_D$. Bandit feedback is of the form $f_m = R_m - R_{m-1}$, which tells us the absolute level of performance associated with selecting an arm. Weights of the $k-$arms are learned using the Exponential Weighted Average Forecasting algorithm [8]:

$$w_{m+1}(d) = \begin{cases} w_m(d) + \eta\frac{f_m}{\mathbf{p}_m(d)} & \text{if } d = d_m \\ w_m(d) & \text{otherwise} \end{cases} \tag{10}$$

5 OUR APPROACH

5.1 Learning β using gradient-based algorithms

Rather than choosing β as a hyper-parameter, we want to learn it. Many optimization algorithms assume a setting where information about the gradient of the loss function with respect to the parameters being optimized is available. We can update β for each episode m or for each sample from the replay buffer. Updating β has a direct effect on the exploration strategy and sample efficiency. To update β for each sample from the replay buffer, we would need the gradient signal characteristic of the potential update to the value β, which will lead to faster convergence towards optimal action-value estimates. Since we cannot define an action-values distribution in continuous state-action pairs, it is challenging to reliably update β. Similar to updating β for each sample from the replay buffer, updating β for each episode m is challenging as well. The non-stationarity of the action-values estimates also adds to the difficulty of finding a reliable gradient estimate to update β. In essence, it may not be possible to obtain reliable knowledge of the relationship between β and $Q(s,a)$, implying that the gradient-based algorithms may be either infeasible or undependable. Moreover, the cost of achieving effective convergence depends not only on the number of required iterations but also on the cost per iteration, which is usually higher in gradient-based algorithms. Additionally, the convergence rates when using function approximators such as neural networks may not accurately reflect practical convergence rates in limited sample sizes.

One workaround to update β for each episode m is to use absolute feedback $f_m = R_m - R_{m-1}$. While this does not tell us anything about the relative feedback of using a specific value of β over time, f_m serves as a good proxy, especially in our case, where the domain of $\beta \in \mathbb{R}$, and where we need to find the optimal β in 1M timesteps. Hence, current gradient-based algorithms act as a bottleneck to update β. Instead, we use gradient-free optimization algorithms like SPSA which we describe in the next subsection.

5.2 Simultaneous perturbation Stochastic Approximation (SPSA)

Simultaneous Perturbation Stochastic Approximation (SPSA)[21] is a general recursive optimization algorithm that is specifically useful when information associated with the gradient vector of the loss function is not available or is too resource intensive to compute. Instead, gradient-free algorithms similar to SPSA are based on approximating the gradient formed from noisy measurements of the loss function.

Contrary to the finite-differences method where evaluation is performed on only one shifted component of the learnable parameter vector, SPSA approximates the gradient by evaluating the loss function at perturbed values of the original learnable parameter: A random noise is added to every component of the original parameter vector.

Let $L(\theta)$ be a differentiable loss function where θ is an n-dimensional vector. There exists a θ^* at which $\frac{\partial L}{\partial \theta} = 0$. SPSA starts with an initial parameter vector $\hat{\theta}_0$. Its update rule uses the stochastic approximation scheme given by:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - \eta_k \hat{g}_k(\hat{\theta}_k) \tag{11}$$

where $\hat{\theta}_k$ is the $k-$th feasible solution, $\eta_k \in \mathbb{R}$ is the learning rate, and $g_k \in \mathbb{R}$ is an iterative direction, a stochastic estimate of the gradient.

Let $y(\theta) = L(\theta) + \epsilon$, where ϵ is some perturbation of the output. We can estimate the gradient at each timestep as follows:

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{L(\hat{\theta}_k + c_k\Delta_k) - L(\hat{\theta}_k - c_k\Delta_k)}{2c_k\Delta_{ki}}, \tag{12}$$

where c_k is a positive number and $\Delta_k = (\Delta_{k_1}, \Delta_{k_2}, ..., \Delta_{k_n})^T$ is a perturbation vector.

In general, gradient-free stochastic algorithms have convergence characteristics similar to those of gradient-based stochastic algorithms, such as Robbins-Monro stochastic approximation (R-M SA), while only requiring measurements of the loss function. The main benefit of such algorithms is that they don't need a functional connection between the parameters being optimized and the loss function being minimized, which is necessary for gradient-based algorithms.

5.3 Learn β using SPSA1

Our goal is the following unconstrained optimization to find the degree of optimism/pessimism β that maximizes expected return $J(\theta)$:

$$\text{(P)} \max_{\beta \in \mathbb{R}} J(\theta) \tag{13}$$

There are two versions of approximating g_m: One-sided and Two-sided. One-sided gradient approximations involve measuring $L(\beta_m)$ and $L(\beta_m + \text{perturbation})$, whereas two-sided gradient approximations measures $L(\beta_m \pm \text{perturbation})$. Before episode m begins, we perturb β_m to estimate the gradient. We use one-sided gradient approximation as it is relatively cheap to compute compared to two-sided gradient approximations. Hence, our gradient estimate is:

$$g_m = L(\beta_m + c_m \cdot \Delta_m)\frac{\Delta_m}{c_m} \tag{14}$$

Algorithm 1 Algorithm1: cTOP

Initialize critic networks Q_{ϕ_1}, Q_{ϕ_2}, and actor π_θ
Initialize target networks $\phi_1' \leftarrow \phi_1, , \phi_2' \leftarrow \phi_2, \theta' \leftarrow \theta$
Initialize replay buffer and $\beta_0 \sim \mathcal{U}([-1, 1])$
for episode in $m = 1, 2, \ldots$ **do**
  Initialize episode reward $R_m \leftarrow 0$
  Perturb β_m as $\beta_m \leftarrow \beta_m + \Delta_m$
  for time step $t = 1, 2, \ldots$ **do**
    Select noisy action $a_t = \pi_\theta(s_t) + \epsilon, \epsilon \sim \mathcal{N}(0, 1)$
    Obtain $r_{t+1}, s_{t+1} = \text{Env}(a_t)$
    Add to total reward $R_m \leftarrow R_m + r_{t+1}$
    Store transition $\mathcal{B} \leftarrow B \cup \{(s_t, a_t, r_{t+1}, s_{t+1})\}$
    Sample N transitions $\mathcal{T} = (s, a, r, s')_{n=1}^N \sim \mathcal{B}$
    UpdateCritics($\mathcal{T}, \beta_m, \theta', \phi_1', \phi_2'$)
    if $t\%b$ **then**
      UpdateActor($\mathcal{T}, \beta_m, \theta, \phi_1, \phi_2$)
      Update: $\phi_i' : \phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i', i \in \{1, 2\}$
      Update: $\theta' : \theta' \leftarrow \tau\theta + (1-\tau)\theta$
    end if
  end for
  $f_m = R_m - R_{m-1}$
  SPSAUpdate(β_m, f_m)
end for

Algorithm 2 : SPSAUpdate

Error Buffer $\mathcal{B}_e$ = deque(maxlen=5)
$\eta = 0.1$, decay rate $\alpha = 1 - \eta$
function UPDATE(β_m, f_m)
  $\mathcal{B}_e \leftarrow \mathcal{B}_e \cup f_m$
  $f_{normalized}$ = Normalize($\mathcal{B}_e$)
  $g_m = \epsilon_m * f_{normalized}$
  if m > 1 **then**
    $\beta_m \leftarrow \alpha\beta_m$
  end if
  $\beta_m \leftarrow \beta_m + \eta g_m$
  $m \leftarrow m + 1$
end function

where $\Delta_m \sim \mathcal{N}(0, 1)$. In our case, the feedback is our loss function L. Our feedback mechanism is the same as TOP:

$$L_{\text{sb}} = R_{m+1} - R_m \tag{15}$$

One of the challenges in directly using SPSA is that the learning rate η_m must decay over the entire course of the agent's training experience to satisfy convergence properties. This means we would need to specify how many episodes M the agent will experience in its training lifetime. M now becomes a random variable as it is dependent on different environments. To mitigate this problem, we use Exponential Weighted Averaging over the recent five feedback samples. Algorithm1 describes the skeleton with changes marked in red made to the TOP[17] framework. Algorithm2 shows the SPSA update mechanism.

6 EXPERIMENTS

Table 1: Average reward over five seeds on Mujoco tasks, trained for 1M time steps. ± values denote one standard deviation across trials. Values within one standard deviation of the highest performance are listed in bold

Task	*cTOP*	*TOP-TD3*	*TD3*
Hopper-v2	3630 ± 120	**3695 ± 152**	3367 ± 136
Walker2d-v2	**5672 ± 447**	5408 ± 87	4529 ± 360
HalfCheetah-v2	**13148 ± 438**	13076 ± 357	12321 ± 657

The key question we would like to answer is whether learning the degree of optimism/pessimism β help, rather than pre-specifying the arm values when framed as a multi-armed bandit problem. We test this hypothesis by running experiments on three state-based continuous control tasks from the Mujoco framework [24] via OpenAI gym [7]. We use TD3 and TOP-TD3 as baselines. TD3 uses the default hyperparameters configuration. TOP-TD3 uses the arm settings $\{-1, 0\}$, with $\beta = -1$ corresponding to a pessimistic arm and $\beta = 0$ corresponds to average of critics' quantiles. Hyperparameters were kept constant for all environments. Each algorithm was trained for one million steps and repeated for five random seeds. Our results displayed in Table 1 show cTOP outperforming TOP-TD3 and TD3 for Walker2d-v2 and HalfCheetah-v2, while marginally underperforming for Hopper-v2.

7 CONCLUSION

We empirically demonstrate that learning the degree of optimism β while training the agent is often helpful across tasks. To vary the degree of optimism, previous off-policy RL algorithms either rely on a fixed value of optimism or may have to pre-specify the arm values when framed as a multi-armed bandit problem. We show that cTOP adaptively updates its degree of optimism while training the agent in order to achieve maximum return.

One limitation of our algorithm is stability. While cTOP achieves a better average return for Walker2d-v2 and HalfCheetah-v2, there is a marginal drop in performance for Hopper-v2. We believe this is due to the non-decaying nature of EWMA on the learning rate. This might lead to noisy updates, which could result in divergence from the optimal action-value function. A natural extension will be to use meta-gradients to learn β robustly.

REFERENCES

[1] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. 2020. Agent57: Outperforming the Atari Human Benchmark. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 507–517. https://proceedings.mlr.press/v119/badia20a.html

[2] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. 2018. Distributed Distributional Deterministic Policy Gradients. In *6th International Conference on Learning Representations, ICLR 2018, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, Vancouver, BC, Canada. https://openreview.net/forum?id=SyZipzbCb

[3] Marc G. Bellemare, Will Dabney, and Rémi Munos. 2017. A Distributional Perspective on Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 449–458. https://proceedings.mlr.press/v70/bellemare17a.html

[4] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2012. The Arcade Learning Environment: An Evaluation Platform for General Agents. *CoRR* abs/1207.4708 (2012). arXiv:1207.4708 http://arxiv.org/abs/1207.4708

[5] Richard Bellman. 1957. *Dynamic Programming* (1 ed.). Princeton University Press, Princeton, NJ, USA.

[6] Ronen I. Brafman and Moshe Tennenholtz. 2003. R-Max - a General Polynomial Time Algorithm for near-Optimal Reinforcement Learning. *J. Mach. Learn. Res.* 3 (mar 2003), 213–231.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540 http://arxiv.org/abs/1606.01540

[8] Nicolo Cesa-Bianchi and Gabor Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press. https://doi.org/10.1017/CBO9780511546921

[9] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. 2019. Better Exploration with Optimistic Actor Critic. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32.

[10] Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. 2018. Implicit Quantile Networks for Distributional Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 1096–1105. https://proceedings.mlr.press/v80/dabney18a.html

[11] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. 2018. Distributional Reinforcement Learning with Quantile Regression. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, Article 353, 10 pages.

[12] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 1587–1596. https://proceedings.mlr.press/v80/fujimoto18a.html

[13] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. [n.d.]. Soft Actor-Critic Algorithms and Applications. ([n. d.]). http://arxiv.org/abs/1812.05905

[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. http://arxiv.org/abs/1509.02971

[15] Long-Ji Lin. 1992. *Reinforcement Learning for Robots Using Neural Networks*. Ph.D. Dissertation. USA.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[17] Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. 2021. Tactical Optimism and Pessimism for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*.

[18] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29.

[19] Jack Parker-Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts. 2020. Effective Diversity in Population Based Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 33. 18050–18062.

[20] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on Machine Learning (Proceedings of Machine Learning Research, 1)*. PMLR, Bejing, China, 387–395. https://proceedings.mlr.press/v32/silver14.html

[21] James C. Spall. [n.d.]. An Overview of the Simultaneous Perturbation Method for Efficient Optimization. https://www.jhuapl.edu/spsa/PDF-SPSA/Spall_An_Overview.PDF

[22] Richard S. Sutton. 1988. Learning to Predict By the Methods of Temporal Differences. *Machine Learning* 3, 1 (August 1988), 9–44. http://www.cs.ualberta.ca/~sutton/papers/sutton-88.pdf

[23] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller (Eds.), Vol. 12. MIT Press. https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf

[24] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

[25] Christopher John Cornish Hellaby Watkins. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation. King's College, Cambridge, UK. http://www.cs.rhul.ac.uk/

~chrisw/new_thesis.pdf