A Brief Survey on Actor-Critic Methods

DDA4230 Final Project

Abstract

This literature review offers a comprehensive analysis of actor-critic methods in model-free reinforcement learning, a key area combining policy-based and value-based approaches. These methods have gained prominence for their effective balance of exploration and exploitation tasks and reduced variance during training. The review traces the evolution of actor-critic algorithms from foundational concepts to advanced variations incorporating deep learning. It critically examines various algorithms, highlighting their strengths and weaknesses. Additionally, the review addresses the practical challenges and theoretical limitations of these methods, providing insights into their real-world applicability while also providing open questions for potential future research in this area.

1 Introduction

Reinforcement Learning (RL) is a branch of machine learning and artificial intelligence which aims to create systems capable of learning optimal behaviors through iterative interactions with their environment. This domain of artificial intelligence is pivotal for developing agents that can learn and improve over time using a trial-and-error approach, applicable across a spectrum of applications from robotics to software-based agents in natural language processing and multimedia [1].

Actor-critic methods are one of the most important concepts that are central to the advances in deep reinforcement learning (DRL). It merges the benefits of policy-based and value-based approaches within a cohesive framework. These methods have been instrumental in propelling DRL to the forefront of AI research, enabling learning in complex scenarios such as video games and robotics, where traditional algorithms falter [1]. The intrinsic balance of exploration and exploitation tasks, coupled with the management of variance during training, proves that actor-critic methods are a potent tool for a variety of RL challenges [1].

The development of actor-critic methods has been marked by significant evolution, from the foundational structures to contemporary variations that integrate deep learning [2][3]. This literature review aims to trace this advancement, offering a comprehensive dive into the actor-critic paradigm, its iterations, and its practical implementations. It aims to present a critical examination of the strengths, limitations, and applications of various actor-critic algorithms.

The rest of the paper is structured as follows: Section 2 introduces the foundational actor and critic components, setting the stage for the concepts discussed in the paper. Section 3 explores the various variations of actor-critic methods, detailing how each is structured and functions in different contexts. Section 4 then provides a discussion comparing these variations, focusing on their strengths and weaknesses in diverse scenarios as well as the challenges and limitations of implementing actor-critic methods, both in practice and theory. The paper concludes with Section 5, summarizing key insights from the review, reflecting on the current state of actor-critic methods in reinforcement learning, and suggesting avenues for future research.

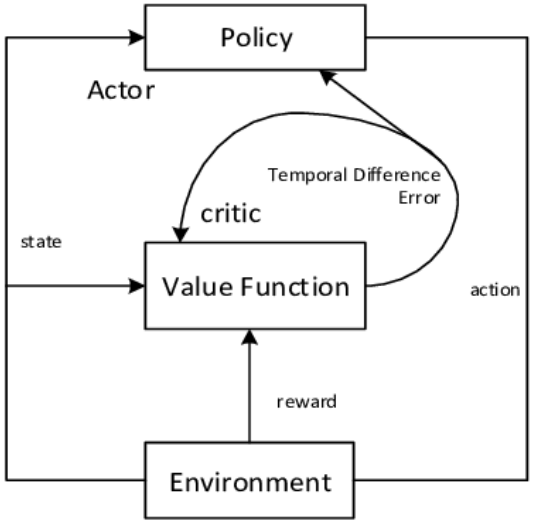

Figure 1: Illustration of the Actor-Critic Framework

2 Base actor-critic

2.1 Foundation of actor-critic

In Reinforcement Learning (RL), an agent learns to make decisions through interactions with its environment. The agent decides what action a (or a_t) to take depending on the policy π (or π_t), which is the strategy for deciding actions based on the current state. The state s is decided by the environment depending on the previous state and the action taken, it is the agent's current situation or environment. The state-value $V(s)$ and state-action value $Q(s, a)$ functions estimate the expected return from a state or state-action pair, respectively.

Actor-critic methods in RL utilize a dual-component framework. The actor component makes decisions based on the policy π [4]. In contrast, the critic evaluates these actions by estimating the value function $V(s)$, often using methods like Temporal Difference (TD) learning [5]. This integrated approach allows for a dynamic balance between exploring actions (actor) and providing feedback for policy improvement (critic) [6].

A simple diagram for the general actor-critic framework is illustrated in Figure 1. The actor receives the current state from the environment and selects an action based on its policy. This action is then applied to the environment, which in response provides a new state and an associated reward. The critic, utilizing the value function, assesses the action by calculating the loss, such as the Temporal Difference (TD) Error which is based on the difference between predicted and actual rewards. This error is then used to update both the policy, through the actor, and the value function itself, ensuring that the actor's decisions are continually refined for future encounters with the environment.

2.2 The actor model

The actor model in the actor-critic model will try to update the current policy with the guidance of the state or state-action value functions provided by critic model. The actor model is responsible for learning a policy $\pi(a|s;\theta)$ that maps states to a probability distribution over actions. We set the policy to be parameterized by θ, which will be updated by the actor in an effort to maximize the expected return.

The expected return $J(\theta)$, when following policy π, is the expected sum of discounted rewards:

$$J(\theta) = \mathbb{E}_{\pi(\theta)}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right], \tag{1}$$

where γ is the discount factor and R_t is the reward at time t. The actor's policy parameters are updated by performing gradient ascent on $J(\theta)$:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta_t), \tag{2}$$

where α represents the learning rate.

For the base actor-critic algorithm, the actor's parameter update utilizes a one-step return and incorporates the value function estimated by the critic as a baseline. The estimated value function $\hat{V}(s, w)$, parameterized by w, serves to reduce the variance of the actor's update step:

$$\theta_{t+1} = \theta_t + \alpha \left(r_t + \gamma \hat{V}(S_{t+1}, w) - \hat{V}(S_t, w) \right) \nabla_\theta \log \pi_\theta(a_t | S_t). \tag{3}$$

This value function $\hat{V}(s, w)$ is learned by the critic model and is crucial for guiding the actor towards more rewarding actions.

2.3 The critic model

The critic model in the actor-critic framework estimates the value function, which will serve as an evaluator for the policy executed by the actor. This value function can be either a state-value function $V(s; w)$ or an action-value function $Q(s, a; w)$, both will parameterized by w.

The critic's objective is to minimize the loss between the estimated value function and the actual return. The loss function, often referred to as the Temporal Difference (TD) error, is defined as:

$$\delta_t = r_t + \gamma \hat{V}(S_{t+1}, w) - \hat{V}(S_t, w), \tag{4}$$

where δ_t is the TD error at time t, r_t is the reward, γ is the discount factor, and $\hat{V}(s, w)$ is the estimated value function, parameterized by w. The TD error reflects the difference between the predicted value of the current state and the estimated value of the next state, adjusted by the received reward.

The critic updates its value function parameters w by applying gradient descent to minimize the TD error:

$$w_{t+1} = w_t + \beta \delta_t \nabla_w \hat{V}(S_t, w), \tag{5}$$

where β is the learning rate for the critic. This update process enables the critic to make more accurate predictions of future rewards, which in turn, provides more informative feedback to the actor's policy.

2.4 Base actor-critic algorithm

Algorithm 1 Base actor-critic algorithm

Initialize the policy parameters θ and value function parameters w randomly
for each episode **do**
    Initialize s, the first state of the episode
    for $t = 0, 1, \ldots, T-1$ **do**
        Sample $a \sim \pi(a|s, \theta)$
        Take action a and observe s', r
        Compute delta $\delta \leftarrow r + \gamma \hat{V}(s', w) - \hat{V}(s, w)$
        Update Critic: $w \leftarrow w + \beta \delta \nabla_w \hat{V}(s, w)$
        Update Actor: $\theta \leftarrow \theta + \alpha \delta \nabla_\theta \log \pi(a|s, \theta)$
        $s \leftarrow s'$

3 Variations of actor-critic methods

3.1 Advantage Actor-Critic methods

Advantage Actor-Critic Methods, encompassing both Asynchronous Advantage Actor-Critic (A3C) and its synchronous counterpart, Advantage Actor-Critic (A2C), is a variation of the base actor-critic

method that uses a core concept of the advantage function to guide policy updates. A3C, notable for its asynchronous update mechanism, utilizes multiple agents operating in parallel environments [7], thereby enhancing exploration and robustness of the learning process. In contrast, A2C synchronizes the updates across all agents, leading to a more stable and consistent training experience. Both methods incorporate entropy in their policy updates [7] to promote exploration and mitigate the risk of premature convergence to sub-optimal policies.

The update steps for the actor and critic is the same for A2C and A3C. For both update steps the method makes use of the advantage function, defined as:

$$A = r + \gamma \hat{V}(s', w) - \hat{V}(s, w) \tag{6}$$

Then both models are updated as below,

Actor Update in A2C:

$$\theta_{t+1} = \theta_t + \alpha A \nabla_\theta \log \pi(a|s, \theta) + \eta \nabla_\theta H(\pi(\cdot|s, \theta)) \tag{7}$$

where $H(\pi(\cdot|s, \theta))$ is the entropy of the policy.

Critic Update in A2C::

$$w_{t+1} = w_t + \beta \nabla_w A^2 \tag{8}$$

The entropy term $H(\pi(\cdot|s, \theta))$ in the A2C and A3C algorithm is calculated to encourage exploration by the policy. It is defined as the sum over all actions of the negative product of the policy's probability of selecting an action and the logarithm of that probability:

$$H(\pi(\cdot|s, \theta)) = -\sum_a \pi(a|s, \theta) \log \pi(a|s, \theta) \tag{9}$$

This entropy term is added to the policy gradient, ensuring that the policy avoids premature convergence to a deterministic behavior and maintains a level of stochasticity in its action selection.

These updates lead to more stable and efficient learning in A2C and A3C compared to the base actor-critic method. The general algorithm for A2C is exactly the same as base actor-critic method, with the update steps for the actor and critic model being consistent with the mathematical notations above.

For the A3C method the general algorithm is given as follows:

Algorithm 2 A3C (Asynchronous Advantage Actor-Critic) Algorithm

Initialize global policy parameters θ and global value function parameters w randomly
Initialize multiple agents with their own set of parameters θ' and w'
for each agent **do**
    for each episode **do**
        Initialize s, the first state of the episode
        for $t = 0, 1, \ldots, T-1$ **do**
            Sample $a \sim \pi(a|s, \theta')$
            Take action a and observe s', r
            Compute advantage $A \leftarrow r + \gamma \hat{V}(s', w') - \hat{V}(s, w')$
            Accumulate gradients w.r.t. θ' and w'
            $s \leftarrow s'$
        Perform asynchronous update of global θ and w using accumulated gradients
        Update local parameters $\theta' \leftarrow \theta$, $w' \leftarrow w$

Both advantage actor-critic methods offer unique advantages in reinforcement learning. Other than that the incorporation of the advantage function, the entropy term leads to more efficient learning and better exploration. A2C, with its synchronous update approach, is efficient in utilizing computational resources like GPUs while A3C capitalizes on parallel, asynchronous actor-learners, which accelerates training speed and enhances exploration efficiency. However, both methods face challenges in balancing exploration with exploitation and require careful tuning of hyperparameters.

A2C can be sensitive to hyperparameter settings, impacting its exploration efficiency. A3C, while offering a more robust policy due to its multiple independent agents, introduces complexity in implementation, particularly with asynchronous updates.

The Advantage Actor-Critic methods each brings unique strengths to reinforcement learning. A2C's synchronous updates and computational efficiency make it well-suited for controlled, resource-limited environments, while A3C's asynchronous approach accelerates learning and enhances exploration through parallel actor-learners.

3.2 Soft Actor-Critic (SAC) method

The Soft Actor-Critic (SAC) algorithm is an off-policy, model-free reinforcement learning approach that integrates the maximum entropy framework. The main difference of this method is that it integrates the entropy term directly in the objective function as well as being an off-policy method. This method aims to simultaneously maximize expected return and entropy, promoting a balance between efficient exploration and exploitation [8].

SAC's objective function can be described as:

$$J(\pi) = \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[\sum_t \gamma^t (r(s_t,a_t) + \tau H(\pi(\cdot|s_t)))\right], \tag{10}$$

where $H(\pi(\cdot|s_t))$ represents the entropy of the policy at state s_t, and τ is the temperature parameter that balances the importance of entropy versus reward.

The updates for the actor and critic model in SAC are given by:

Actor Update in SAC:

$$\theta_{t+1} = \theta_t + \alpha\nabla_\theta \left(\log \pi(a_t|s_t,\theta)(Q_\theta(s_t,a_t) - V(s_t))\right) \tag{11}$$

Critic Update in SAC:

$$w_{t+1} = w_t + \beta \left(V(s_{t+1}) - \tau \log \pi(a_{t+1}|s_{t+1},\theta)\right) \tag{12}$$

The algorithm for SAC is given below:

Algorithm 3 Soft Actor-Critic (SAC) Algorithm

Initialize policy parameters θ, value function parameters w, and temperature parameter τ
for each episode **do**
    Initialize s, the first state of the episode
    for $t = 0, 1, \ldots, T-1$ **do**
        Sample $a_t \sim \pi(a|s,\theta)$
        Take action a_t, observe reward r_t and next state s'
        Update actor: $\theta_{t+1} = \theta_t + \alpha\nabla_\theta \left(\log \pi(a_t|s_t,\theta)(Q_\theta(s_t,a_t) - V(s_t))\right)$
        Update critic: $w_{t+1} = w_t + \beta \left(V(s_{t+1}) - \tau \log \pi(a_{t+1}|s_{t+1},\theta)\right)$
        $s \leftarrow s'$

SAC's incorporation of entropy regularization promotes better exploration, leading to more robust policy learning, especially in environments with high-dimensional, continuous action spaces. The off-policy nature of SAC also allows for more efficient use of past experiences, enhancing sample efficiency. However, SAC can be even more complex in terms of hyperparameter tuning, especially in determining the optimal balance between exploration (entropy) and exploitation (reward) since the entropy term is integrated directly into the objective function instead of the update steps.

SAC presents a novel approach in the actor-critic algorithm family, with its distinctive emphasis on entropy maximization for improved exploration and policy robustness. Its off-policy nature and efficient use of past experiences make it a compelling choice for complex reinforcement learning tasks.

3.3 Deep Deterministic Policy Gradient (DDPG) method

Deep Deterministic Policy Gradient (DDPG) is an off-policy actor-critic method that uniquely blends ideas from Deep Q-Networks (DQN) and deterministic policy gradients. Unlike the base actor-critic and its variations like A2C and A3C, which typically employ stochastic policies, DDPG operates with a deterministic policy in continuous action spaces [9]. This deterministic approach, combined with techniques like experience replay and target networks [9], distinguishes DDPG from other actor-critic methods.

The objective function in DDPG follows a Q-learning style update for the critic and a policy gradient update for the actor. The critic's objective is to approximate the Q-function, while the actor aims to maximize the Q-value predicted by the critic. The updates can be described mathematically as:

Actor Update in DDPG:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \pi(s_t, \theta) \nabla_a Q(s_t, a, w)|_{a=\pi(s_t,\theta)} \tag{13}$$

Critic Update in DDPG:

$$w_{t+1} = w_t + \beta \nabla_w \left(Q(s_t, a_t, w) - (r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}, \theta), w'))^2 \right) \tag{14}$$

The general algorithm for DDPG is given below:

Algorithm 4 DDPG (Deep Deterministic Policy Gradient) Algorithm

Initialize the policy parameters θ and Q-function parameters w randomly
Initialize target policy parameters θ' and target Q-function parameters w'
Initialize replay buffer $\mathcal{D}$
for each episode **do**
    Initialize s, the first state of the episode
    for $t = 0, 1, \ldots, T-1$ **do**
        Sample action $a_t \sim \pi(a|s_t, \theta)$ according to the current policy
        Take action a_t and observe reward r_t and next state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
        Sample a random minibatch of transitions from $\mathcal{D}$
        Compute target value $y = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}, \theta'), w')$
        Update Critic: $w \leftarrow w + \beta \nabla_w \left(Q(s_t, a_t, w) - y \right)^2$
        Update Actor: $\theta \leftarrow \theta + \alpha \nabla_\theta \pi(s_t, \theta) \nabla_a Q(s_t, a, w)|_{a=\pi(s_t,\theta)}$
        Update target networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
                                $w' \leftarrow \tau w + (1-\tau)w'$
        $s_t \leftarrow s_{t+1}$

A few advantages of DDPG are that it is particularly effective in environments with continuous action spaces, where its deterministic policy allows for precise action selection. Additionally, the algorithm's use of experience replay and target networks helps stabilize the training process. However, some disadvantages faced by DDPG include its tendency for limited exploration due to its deterministic nature, which can lead to getting trapped in local optima. They also suffer from sample inefficiency and require careful hyperparameter tuning to optimize performance.

DDPG stands out for its application in environments that demand precise and continuous control actions. Its deterministic policy gradient approach, combined with stability mechanisms borrowed from DQN, provides a powerful tool for solving complex reinforcement learning problems.

3.4 Twin-Delayed Deep Deterministic Policy Gradient (TD3) method

Twin-Delayed Deep Deterministic Policy Gradient (TD3) is an advanced off-policy actor-critic method that enhances the Deep Deterministic Policy Gradient (DDPG) framework. TD3 addresses DDPG's overestimation of Q-values by introducing three key improvements: Double Q-Learning, Delayed Policy Updates, and Target Policy Smoothing [6],[10]. Double Q-Learning employs two critic networks to provide a more conservative Q-value estimation, reducing bias. Delayed Policy Updates slow down the frequency of policy updates compared to the critic updates, enhancing the

stability of learning. Target Policy Smoothing adds noise to the action chosen by the target policy, improving robustness.

The TD3 algorithm's mathematical updates are as follows:

Actor Update in TD3:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \pi(s_t, \theta) \nabla_a Q(s_t, a, w)|_{a=\pi(s_t,\theta)} \tag{15}$$

Critic Update in TD3:

$$w_{t+1} = w_t + \beta \nabla_w \min_{i=1,2} (Q_i(s_t, a_t, w) - (r_t + \gamma Q_i(s_{t+1}, \pi(s_{t+1}, \theta), w'))^2) \tag{16}$$

Below is the general algorithm for the TD3 method:

Algorithm 5 TD3 (Twin-Delayed Deep Deterministic Policy Gradient) Algorithm

Initialize the policy parameters θ and two sets of Q-function parameters w_1, w_2 randomly
Initialize target policy parameters θ' and target Q-function parameters w_1', w_2'
Initialize replay buffer $\mathcal{D}$
for each episode **do**
    Initialize s, the first state of the episode
    for $t = 0, 1, \ldots, T-1$ **do**
        Sample action $a_t \sim \pi(a|s_t, \theta)$ according to the current policy and add noise
        Take action a_t and observe reward r_t and next state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
        Sample a random minibatch of transitions from $\mathcal{D}$
        Compute target value $y = r_t + \gamma \min_{i=1,2} Q_i(s_{t+1}, \pi_{\text{target}}(s_{t+1}, \theta'), w_i')$
        Update Critics: For $i = 1, 2$,
        $w_i \leftarrow w_i + \beta \nabla_{w_i} \left(Q_i(s_t, a_t, w_i) - y\right)^2$
        if $t \mod d$ **then**
            Update Actor: $\theta \leftarrow \theta + \alpha \nabla_\theta \pi(s_t, \theta) \nabla_a Q_1(s_t, a, w_1)|_{a=\pi(s_t,\theta)}$
        Update target networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
                                $w_i' \leftarrow \tau w_i + (1-\tau) w_i'$ for $i = 1, 2$
        $s_t \leftarrow s_{t+1}$

Advantages of TD3 include its reduced overestimation bias and improved stability in learning, especially in environments with continuous action spaces. However, the complexity introduced by twin critics and the delayed update mechanism can increase computational demands and slow down learning.

TD3 represents a significant step forward in reinforcement learning for continuous control tasks, offering a more stable and reliable policy learning process by mitigating DDPG's overestimation bias.

3.5 Actor-Critic TRPO Method

Actor-Critic TRPO is an on-policy algorithm that primarily addresses the challenge of large policy updates [6][11], which can destabilize learning in traditional actor-critic methods. The key distinction of TRPO lies in its implementation of a trust region, which effectively constrains the policy updates to ensure that the new policy is not too far from the old one. This constraint is quantified using the Kullback-Leibler (KL) divergence. It optimizes a surrogate objective function under the trust region constraint, which restricts the extent of divergence between the new and old policies.

For TRPO, the surrogate objective function for the policy (actor) is given by:

$$L^{TRPO}(\theta) = \mathbb{E}_t \left[\frac{\pi_\theta(a_t|S_t)}{\pi_{\theta_{old}}(a_t|S_t)} A^{TRPO}(S_t, a_t) \right], \tag{17}$$

where $\pi_\theta(a_t|S_t)$ is the policy under the current parameters θ, $\pi_{\theta_{old}}(a_t|S_t)$ is the policy under the old parameters θ_{old}, and $A^{TRPO}(S_t, a_t)$ is the advantage function at time t. The expectation $\mathbb{E}_t$ denotes the average over a finite batch of samples.

The actor's parameter update in TRPO is performed using a trust region optimization approach. The objective is to maximize the surrogate objective function while ensuring that the KL divergence between the new and old policies remains within a predefined limit. This can be represented as:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta L^{TRPO}(\theta_t) \quad \text{s.t.} \quad D_{KL}(\pi_{\theta_{old}}(\cdot|S_t)||\pi_{\theta_t}(\cdot|S_t)) \leq \delta, \tag{18}$$

The critic in TRPO updates its value function parameters w by minimizing the mean-squared error of the value function estimate:

$$w_{t+1} = w_t + \beta \nabla_w \left(V(S_t, w) - r_t\right)^2, \tag{19}$$

A general algorithm for TRPO actor critic method is as follows:

Algorithm 6 TRPO (Trust Region Policy Optimization) Algorithm

Initialize the policy parameters θ and value function parameters w randomly
Initialize the old policy parameters θ_{old}
for each episode **do**
    Initialize s, the first state of the episode
    for $t = 0, 1, \ldots, T-1$ **do**
        Sample $a \sim \pi(a|s, \theta)$
        Take action a and observe s', r
        Compute advantage estimate $A^{TRPO}(s, a)$
        Compute surrogate objective $L^{TRPO}(\theta)$
        Update Critic: $w \leftarrow w - \beta \nabla_w \left(V(s, w) - r\right)^2$
        Update Actor: Perform constrained optimization to find θ_{t+1}
        $\theta_{old} \leftarrow \theta$
        $s \leftarrow s'$

TRPO's main strength lies in its robust management of large policy updates, making it highly suitable for both continuous and discrete action spaces. The algorithm's trust region approach ensures balanced and stable policy updates, avoiding instability while allowing significant learning progress. However, this method faces challenges such as computational intensity due to its use of second-order optimization methods, making it less practical for problems with large state or action spaces.Additionally, while the trust region constraint stabilizes training, it can also slow down the convergence process.

TRPO's innovative approach to managing policy updates through the trust region concept and KL divergence constraint makes it adept at handling complex environments where maintaining a balance between exploration and exploitation is crucial.

3.6 Actor-Critic PPO Method

Proximal Policy Optimization (PPO) is an on-policy algorithm that builds upon the ideas of TRPO, aiming to simplify and improve the efficiency of policy optimization in reinforcement learning. PPO stands out for its novel approach to handling policy updates, striking a balance between complexity and performance [6]. Unlike TRPO, which uses a complex second-order method to maintain the trust region, PPO achieves a similar objective with a clipped surrogate objective function, simplifying the optimization process while still controlling the size of policy updates.

The surrogate objective function for PPO is defined as follows:

$$L^{PPO}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right], \tag{20}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|S_t)}{\pi_{\theta_{old}}(a_t|S_t)}$ is the probability ratio, A_t is the advantage function at time t, and ϵ is a hyperparameter defining the clipping range. The expectation $\mathbb{E}_t$ indicates averaging over a finite batch of samples.

The actor (policy) updates in PPO are performed via gradient ascent on this clipped objective:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta L^{PPO}(\theta_t), \tag{21}$$

The critic (value function) updates in PPO are similar to those in TRPO and base actor-critic methods:

$$w_{t+1} = w_t + \beta \nabla_w \left(V(S_t, w) - r_t\right)^2, \tag{22}$$

The general algorithm for PPO actor critic method is as follows:

Algorithm 7 PPO (Proximal Policy Optimization) Algorithm

Initialize the policy parameters θ and value function parameters w randomly
for each episode **do**
    Initialize s, the first state of the episode
    for $t = 0, 1, \ldots, T-1$ **do**
        Sample $a \sim \pi(a|s, \theta)$
        Take action a and observe s', r
        Compute advantage estimate $A^{PPO}(s, a)$
        Compute surrogate objective $L^{PPO}(\theta)$
        Update Critic: $w \leftarrow w + \beta \nabla_w \left(V(s, w) - r\right)^2$
        Update Actor: $\theta \leftarrow \theta + \alpha \nabla_\theta \min\left(r_t(\theta) A^{PPO}(s, a), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) A^{PPO}(s, a)\right)$
        $s \leftarrow s'$
    $\theta_{old} \leftarrow \theta$

PPO's main advantage is its ease of implementation and efficiency compared to TRPO. The clipped objective function simplifies the optimization while still effectively constraining the policy updates, making PPO suitable for a wide range of applications. However, PPO may sometimes exhibit reduced sample efficiency and require careful tuning of its clipping parameter ϵ to balance exploration and exploitation.

PPO's simplicity and effectiveness in handling policy updates have made it a popular choice in reinforcement learning tasks, offering a practical alternative to more complex algorithms like TRPO. PPO's balance of performance and simplicity makes it a significant tool for advancing reinforcement learning applications.

4 Discussion

4.1 Comparison and evaluation of actor critic methods

In this section, we compare the key actor-critic methods we have discussed in this review, which are A2C/A3C, SAC, DDPG, TD3, TRPO, and PPO. We will focus on their policy types, suitability for different action spaces, and computational costs.

A2C and A3C (Advantage Actor-Critic): Both methods are on-policy and excel in discrete action spaces. A2C, being synchronous, offers more stable updates, while A3C's asynchronous updates provide faster learning but increased complexity. They are computationally less demanding than some of the more advanced methods.

SAC (Soft Actor-Critic): An off-policy method effective in continuous action spaces, SAC incorporates entropy into the objective function, promoting exploration. It balances sample efficiency with computational demand, making it suitable for complex environments.

DDPG: An off-policy method optimized for continuous action spaces. It is known for precise control but suffers from limited exploration and high computational costs due to its use of replay buffers and target networks.

TD3: An advancement of DDPG, TD3 is off-policy, suitable for continuous spaces. It addresses the overestimation bias of DDPG, offering more stable learning at a slightly higher computational cost due to its twin-critic architecture.

TRPO: An on-policy method applicable in both discrete and continuous spaces. TRPO's trust region approach for policy updates ensures stability but at a high computational cost due to its second-order optimization methods.

PPO: An on-policy algorithm similar to TRPO but with a simpler implementation. PPO is effective in both discrete and continuous spaces and strikes a balance between computational efficiency and policy update stability.

Table 1: Comparison of Actor-Critic Methods

Method	Policy Type	Computational Cost	Action Space
A2C/A3C	On-Policy	Moderate	Discrete
SAC	Off-Policy	Relatively High	Continuous
DDPG	Off-Policy	High	Continuous
TD3	Off-Policy	Relatively High	Continuous
TRPO	On-Policy	Very High	Both
PPO	On-Policy	Moderate to High	Both

In summary, A2C/A3C and SAC can both provide efficient learning in their respective domains, while DDPG and TD3 offer precise control in continuous spaces but with higher computational needs. TRPO provides robust policy updates across different spaces but is computationally intensive. PPO emerges as a balanced alternative, offering simplicity and efficiency, making it widely applicable.

4.2 Challenges and limitations of actor critic methods

Despite their advancements, actor-critic methods face several challenges and limitations. One significant challenge is the balance between exploration and exploitation, particularly in complex environments. Additionally, these methods often require extensive hyperparameter tuning, which can be a resource-intensive process. Another limitation is their computational cost, especially for methods that use second-order optimization techniques like TRPO. These methods can also suffer from stability issues during training, making them less reliable in certain scenarios. Furthermore, the implementation details, including network architectures and loss functions, significantly influence the performance of these algorithms, adding to the complexity of their effective application. Future research could focus on addressing these limitations to enhance the efficiency and robustness of actor-critic methods in more diverse and challenging environments.

5 Conclusion

Actor-critic methods and their variations represent a significant advancement in reinforcement learning, addressing complex decision-making in diverse environments. These methods, including DDPG, TRPO, PPO, and others, have contributed to precise control and robust policy updates in both continuous and discrete action spaces. Despite their strengths, challenges like balancing exploration and exploitation, computational demands, hyper-parameter tuning, and stability issues still persist. Some research has been done to address these problems [12][13] but future research can be more directed towards overcoming these limitations, enhancing efficiency, and adapting these methods for broader real-world applications, from autonomous systems to financial modeling, indicating a promising direction for the field of reinforcement learning.

References

[1] Deep Reinforcement Learning: A Brief Survey. (2017, November 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/8103164

[2] Fan, Z. (2019, March 4). Hybrid Actor-Critic Reinforcement Learning in Parameterized Action Space. arXiv.org. https://arxiv.org/abs/1903.01344

[3] Merdivan, E. (2019, July 2). Modified Actor-Critics. arXiv.org. https://arxiv.org/abs/1907.01298

[4]: A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. (2012, November 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/6392457

[5]: Tesauro, G. (1995). Temporal Difference Learning and TD-Gammon. Retrieved from https://cling.csd.uwo.ca/cs346a/extra/tdgammon.pdf

[6]: Dutta, D., & Upreti, S. R. (2022, July 12). A survey and comparative evaluation of actor-critic methods in process control. The Canadian Journal of Chemical Engineering, 100(9), 2028–2056. https://doi.org/10.1002/cjce.24508

[7] Mnih, V. (2016, February 4). Asynchronous Methods for Deep Reinforcement Learning. arXiv.org. https://arxiv.org/abs/1602.01783

[8] Haarnoja, T. (2018, January 4). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv.org. https://arxiv.org/abs/1801.01290

[9] Lillicrap, T. P. (2015, September 9). Continuous control with deep reinforcement learning. arXiv.org. https://arxiv.org/abs/1509.02971

[10] Zhang, Z., Li, X., An, J., Man, W., & Zhang, G. (2020, December 29). Model-Free Attitude Control of Spacecraft Based on PID-Guide TD3 Algorithm. International Journal of Aerospace Engineering. https://doi.org/10.1155/2020/8874619

[11] Schulman, J. (2015, February 19). Trust Region Policy Optimization. arXiv.org. https://arxiv.org/abs/1502.05477

[12] Andrychowicz, M. (2020, October 2). What Matters for On-Policy Deep Actor-Critic Methods? A Large-Scale Study. OpenReview. https://openreview.net/forum?id=nIAxjsniDzg

[13] Fujimoto, S. (2018, February 26). Addressing Function Approximation Error in Actor-Critic Methods. arXiv.org. https://arxiv.org/abs/1802.09477