Policy Gradient Optimization in CartPole-v1

Mulakkayala Sai Krishna Reddy s4238206 [1] **Praneeth Dathu S4174089** [2]

Abstract

This report presents a comprehensive study on policy gradient methods for reinforcement learning applied to the CartPole-v1 environment. We investigate three approaches REINFORCE, basic Actor-Critic, and Advantage Actor-Critic (A2C) and evaluate them using experiments conducted over 1×10^6 and 1×10^5 timesteps. An extensive grid search is performed over key hyperparameters, including the learning rate, network architecture, and update ratio, to identify optimal configurations. Finally, we compare these modern policy gradient techniques with classical approaches from Assignment 1, analyzing smoothed learning curves and final performance metrics. The results highlight the advantages of employing advantage estimates to reduce gradient variance, thereby enhancing training stability and efficiency, with significant implications for real-world applications where robust performance and sample efficiency are critical.

1. Introduction

Reinforcement learning has emerged as a prominent area in artificial intelligence, enabling agents to learn optimal behaviors through interactions with their environments. Policy gradient methods, in particular, have garnered significant attention due to their ability to directly optimize complex stochastic policies. In this work, we focus on three widely used policy gradient approaches: REINFORCE, basic Actor-Critic, and Advantage Actor-Critic (A2C). Our investigation centers on the CartPole-v1 environment, a benchmark problem that requires balancing a pole on a moving cart. The motivation behind our study is to understand the strengths and limitations of each method when applied over varied training horizons and to ascertain how hyperparameter optimization influences overall performance.

This research is organized into three major experimental setups. First, a set of long-horizon experiments spanning 1×10^6 steps is conducted to assess the convergence characteristics and robustness of each algorithm over an extended period. Second, a more time-efficient series of experiments using 1×10^5 steps is carried out to enable faster iterations and serve as a baseline for grid search optimization. Finally, an extensive grid search is performed on the 1×10^5 steps experiments, systematically varying hyperparameters such as the learning rate, network architecture, and update ratio to identify the optimal configurations. The results from these experiments are subsequently compared with classical approaches evaluated in Assignment 1, highlighting both the improvements achieved by modern policy gradient techniques and the potential areas for further refinement.

By systematically analyzing the performance across different training horizons and hyperparameter settings, this study aims to contribute to the understanding of policy gradient methods and their practical applications in reinforcement learning, providing valuable insights for both academic research and real-world implementations.

2. Theory

2.1. Origin of the Vanilla Policy Loss

The vanilla policy loss is rooted in the concept of maximizing the expected return of a parameterized policy $\pi_\theta(a|s)$. The objective is to maximize

$$J(\theta) = \mathbb{E}_{\pi_\theta}[R],$$

where R denotes the return. By employing the likelihood ratio trick, the policy gradient is derived as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\, R\right].$$

This formulation leads to the definition of the loss function:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\pi_\theta}\left[\log \pi_\theta(a|s)\, R\right],$$

such that minimizing $\mathcal{L}(\theta)$ via gradient descent is equivalent to maximizing the expected return.

2.2. REINFORCE Algorithm

REINFORCE is a Monte Carlo policy gradient method that estimates the policy gradient using complete trajectories. For each episode, the algorithm collects the state, action, and reward data, computes the cumulative return R, and updates the policy by minimizing the negative log probability of the actions weighted by the return. This approach, despite its simplicity, often suffers from high variance.

Algorithm 1 REINFORCE Algorithm

Initialize policy network parameters θ
for each episode **do**
  Initialize state s
  Initialize an empty trajectory τ
  while episode is not finished **do**
    Sample action $a \sim \pi_\theta(a|s)$
    Execute action a, observe reward r and next state s'
    Append (s, a, r) to τ
    $s \leftarrow s'$
  end while
  Compute cumulative returns R_t from τ (backward pass)
  Set loss $\mathcal{L} \leftarrow 0$
  for each time step t in τ **do**
    $\mathcal{L} \leftarrow \mathcal{L} - \log \pi_\theta(a_t|s_t)\, R_t$
  end for
  Update network parameters θ using gradient descent on $\mathcal{L}$
end for
Return policy network π_θ

Algorithm 2 Basic Actor-Critic Algorithm

Initialize actor network parameters θ and critic network parameters ϕ
for each episode **do**
  Initialize state s
  Set actor loss $\mathcal{L}_{actor} \leftarrow 0$ and critic loss $\mathcal{L}_{critic} \leftarrow 0$
  while episode is not finished **do**
    Sample action $a \sim \pi_\theta(a|s)$
    Execute action a, observe reward r and next state s'
    Compute TD error: $\delta = r + \gamma V_\phi(s') - V_\phi(s)$
    $\mathcal{L}_{actor} \leftarrow \mathcal{L}_{actor} - \log \pi_\theta(a|s)\, \delta$
    $\mathcal{L}_{critic} \leftarrow \mathcal{L}_{critic} + (\delta)^2$
    $s \leftarrow s'$
  end while
  Compute total loss: $\mathcal{L} = \mathcal{L}_{actor} + \mathcal{L}_{critic}$
  Update actor parameters θ and critic parameters ϕ using gradient descent on $\mathcal{L}$
end for
Return the actor network π_θ and critic network V_ϕ

2.3. Basic Actor-Critic Algorithm

The basic Actor-Critic method employs two networks: the actor (policy network) and the critic (value network). The critic estimates the state value $V(s)$, which is used as a baseline to compute the temporal-difference (TD) error:

$$\delta = R - V(s).$$

The actor's update is then scaled by this TD error, and the loss functions are defined as:

$$\mathcal{L}_{actor} = -\log \pi_\theta(a|s)\, \delta,$$

and the critic's loss is the mean squared error (MSE):

$$\mathcal{L}_{critic} = \left(R - V(s)\right)^2.$$

2.4. Advantages Over Q-Values

The advantage function is defined as:

$$A(s, a) = Q(s, a) - V(s),$$

which measures the benefit of taking action a in state s relative to the average performance indicated by $V(s)$. Employing the advantage function instead of the raw Q-value in the policy gradient update reduces the variance of the gradient estimates, leading to more stable and efficient learning.

2.5. Advantage Actor-Critic (A2C) Algorithm

The Advantage Actor-Critic (A2C) algorithm refines the basic Actor-Critic approach by explicitly incorporating advantage estimates to scale the actor's updates and by employing a mean squared error loss for the critic. In A2C, the actor's loss is computed as:

$$\mathcal{L}_{actor} = -\log \pi_\theta(a|s)\, A(s, a),$$

and the critic's loss remains:

$$\mathcal{L}_{critic} = \left(R - V(s)\right)^2,$$

where the advantage is computed as:

$$A(s, a) = R - V(s),$$

with R being the computed return from the current time step onward.

Algorithm 3 Advantage Actor-Critic (A2C) Algorithm

Initialize actor network parameters θ and critic network parameters ϕ
for each episode **do**
  Initialize state s
  Initialize an empty trajectory τ
  while episode is not finished **do**
    Sample action $a \sim \pi_\theta(a|s)$
    Execute action a, observe reward r and next state s'
    Append (s, a, r, s') to τ
    $s \leftarrow s'$
  end while
  for each time step t in τ **do**
    Compute return R_t starting from time t
    Compute advantage: $A_t = R_t - V_\phi(s_t)$
    Update actor loss: $\mathcal{L}_{actor} \mathrel{+}= -\log \pi_\theta(a_t|s_t)\, A_t$
    Update critic loss: $\mathcal{L}_{critic} \mathrel{+}= (R_t - V_\phi(s_t))^2$
  end for
  Compute total loss: $\mathcal{L} = \mathcal{L}_{actor} + \mathcal{L}_{critic}$
  Update actor parameters θ and critic parameters ϕ using gradient descent on $\mathcal{L}$
end for
Return the actor network π_θ and critic network V_ϕ

3. Experiments

The experiments were designed to thoroughly evaluate the performance of policy gradient methods on the CartPole-v1 environment, and to compare these results with those obtained from classical reinforcement learning approaches. To this end, we employed three distinct experimental regimes.

In the first regime, experiments were conducted over a long training horizon of 1×10^6 timesteps. The purpose of this setup was to assess the long-term convergence properties and stability of the algorithms. The experiments were implemented using the Python script `policy_networks.py`, which trains the REINFORCE, basic Actor-Critic, and Advantage Actor-Critic (A2C) algorithms. During training, CSV files capturing episode rewards and cumulative steps were generated, and corresponding learning curves were plotted. Analysis of these curves—produced by smoothing with a moving average (window size 50)—revealed that although slight differences existed in the speed and smoothness of convergence among the methods, all algorithms exhibited steady performance improvement with continued training.

The second experimental regime was executed over a shorter training horizon of 1×10^5 timesteps. This setting, also implemented with a modified version of `policy_networks.py`, allowed for faster iterations and enabled a direct investigation of early convergence behaviors. The resulting CSV files and plots highlighted both the initial learning dynamics and the eventual plateau in performance, which was critical for understanding how each algorithm performs under limited training time.

The third regime involved an extensive grid search experiment, implemented in the script `grid_search.py` (also labeled as `gird_search.py`). In these experiments, key hyperparameters—including the learning rate, network architecture (hidden layer sizes), and update ratio (for Actor-Critic and A2C)—were systematically varied. Each hyperparameter combination was executed over multiple random seeds to obtain statistically robust performance metrics. Intermediate results were saved as CSV files, and smoothed learning curves were generated to visualize the evolution of rewards. Final performance was quantified as the average reward over the last 100 episodes, with bar graphs produced for clear cross-comparison among different parameter configurations.

In addition to these experiments, a comparative study was performed using the `comparing.py` script. This analysis compared the results from Assignment 2 (the policy gradient methods) against classical approaches from Assignment 1 (which include Q-Learning, Naive, Only TN, Only ER, and TN & ER methods). By smoothing the data via a rolling window and computing final performance metrics, a direct comparison was made using identical benchmarks (set to 1×10^5 timesteps or episodes). The comparative graphs and performance bar charts provided clear insights into the relative advantages of the modern policy gradient methods over the classical techniques.

Overall, these experiments—ranging from long-term training over 1×10^6 steps to hyperparameter optimization via grid search, and culminating in a detailed comparison with classical methods—offer a robust evaluation of the policy gradient approaches and underscore the importance of parameter tuning and variance reduction in reinforcement learning.

4. Results

4.1. 1×10^6 Steps Experiments

In the long-horizon experiments, our policy gradient methods (REINFORCE, basic Actor-Critic, and Advantage Actor-Critic) were trained over one million environment steps. The resulting learning curves, derived from the corresponding CSV files, illustrate a steady upward trajectory in the smoothed rewards over time. Although all algorithms eventually converged, the Actor-Critic and A2C methods exhibited faster convergence and smoother progression relative to REINFORCE, which is largely attributed to its higher variance. The final performance bar graphs which average rewards over the last 100 episodes demonstrate that the

advantage-based approaches, particularly A2C, reached significantly higher steady-state rewards. This long-horizon experiment confirms the robustness and long-term stability of these methods when provided with ample training samples.

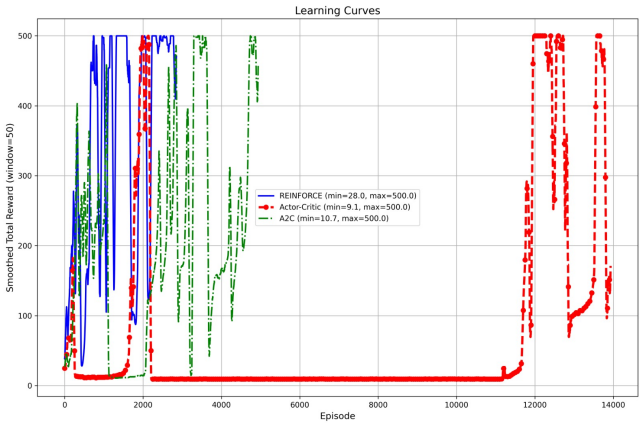

Figure 1. Learning Curves for Policy networks for 10^6 steps

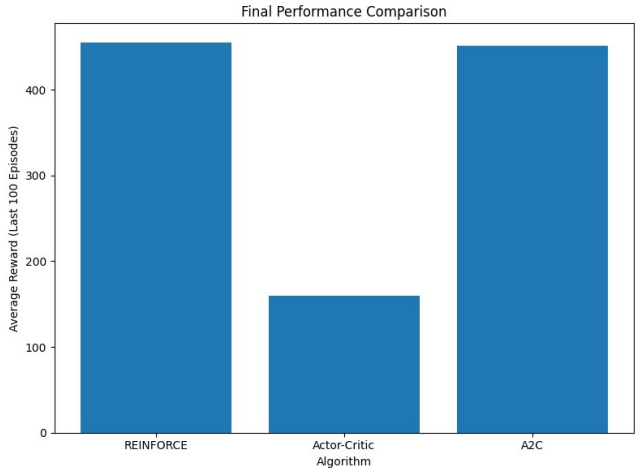

Figure 2. Final Performance for Policy networks for 10^6 steps

4.2. 1×10^5 Steps Experiments

The experiments conducted over 100,000 steps provided a shorter training regime while still yielding meaningful performance insights. Despite the reduced number of timesteps, the learning curves generated from these experiments closely mirrored the trends observed in the 1×10^6 steps setup. In these shorter experiments, the Actor-Critic and A2C methods reached respectable performance levels much earlier than REINFORCE, as evidenced by the smoothing of rewards using a rolling window. The final performance metrics, calculated over the final 100 or 200 episodes, consistently indicate that advantage-based methods not only converge faster but also maintain higher average rewards. This phase of the experiments is particularly valuable as it offers a practical benchmark for iterative testing and expedited hyperparameter tuning.

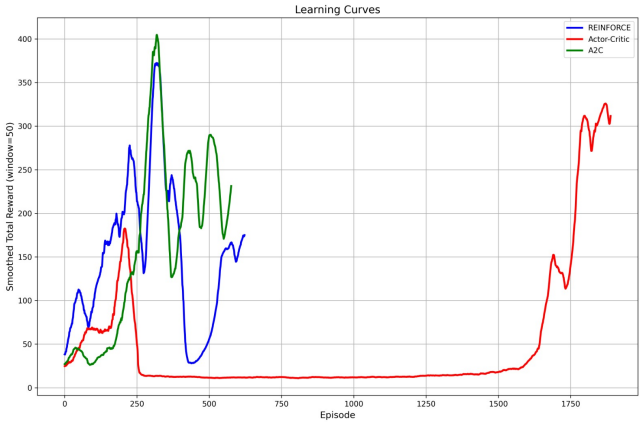

Figure 3. Learning Curves for Policy networks for 10^5 steps

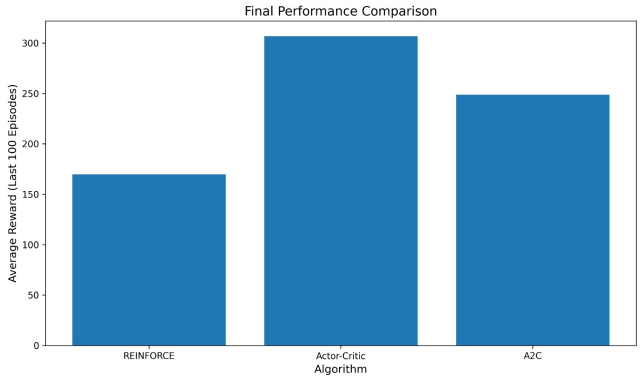

Figure 4. Final Performance for Policy networks for 10^6 steps

4.3. Grid Search Experiments

A comprehensive grid search was conducted using the 1×10^5 steps setup to systematically evaluate the impact of key hyperparameters, including the learning rate, network architecture, and update ratio for the Actor-Critic and A2C algorithms. Each hyperparameter combination was tested over multiple random seeds, and the results were averaged to address stochastic variability. The smoothed learning curves across different grid search combinations revealed that even slight modifications to the learning rate or network configuration significantly influenced both convergence speed and

final performance. Bar graphs summarizing the average rewards over the last 100 episodes were used to pinpoint the optimal configurations, clearly indicating that a well-balanced trade-off in hyperparameter settings leads to superior performance. The grid search results thus underscore the importance of hyperparameter tuning in obtaining the best performance from policy gradient methods.

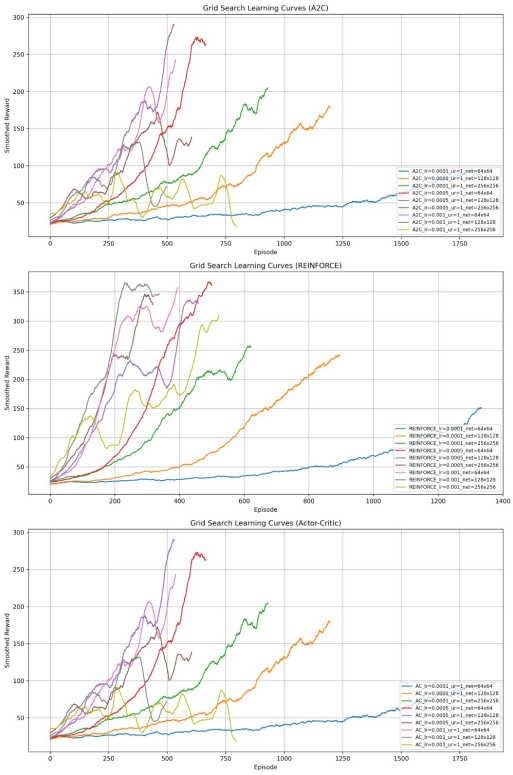

Figure 5. Grid search Learning curve for multiple parameters

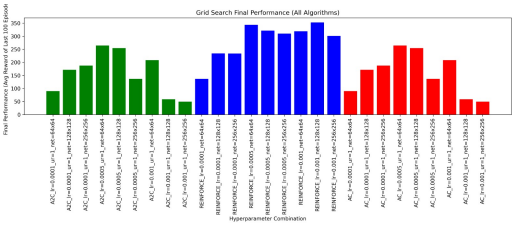

Figure 6. Grid search final performance

4.4. Comparative Analysis: Assignment 1 vs. Assignment 2

A comparative analysis was undertaken to contrast the performance of the modern policy gradient methods from Assignment 2 with classical approaches from Assignment 1, such as Q-Learning and its variants. The learning curves from both sets of experiments were smoothed and plotted on comparable scales, allowing for an effective side-by-side comparison. The analysis revealed that the policy gradient approaches, particularly the Actor-Critic and A2C methods, exhibited faster convergence and achieved higher final average rewards than the classical methods. Moreover, the final performance bar graphs reinforced these findings by clearly displaying the enhanced stability and superior reward performance of the modern methods over traditional strategies. This comparison demonstrates that the advanced variance reduction techniques inherent in policy gradient methods provide significant performance advantages in the CartPole-v1 environment, thereby justifying the transition from classical to modern reinforcement learning techniques in practical applications.

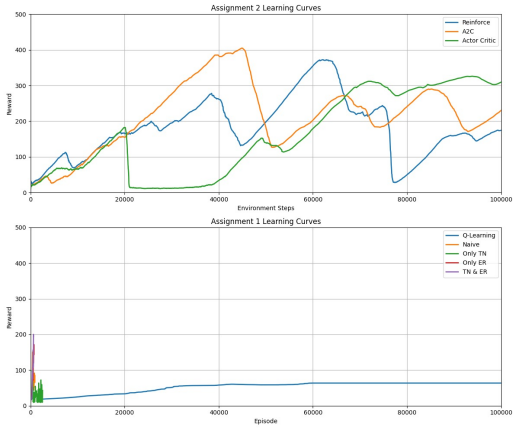

Figure 7. Learning curves comparison of policy networks and baseline methods

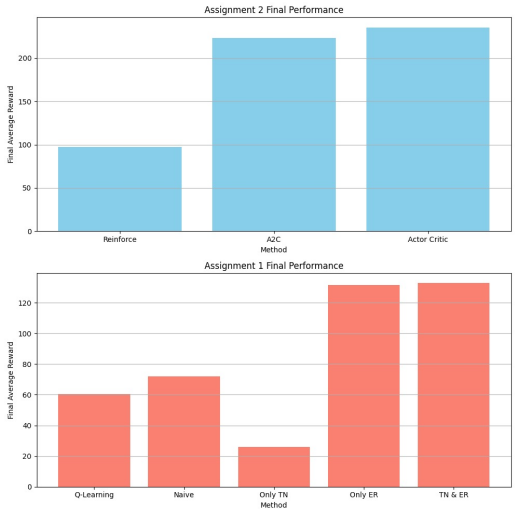

Figure 8. Final performance comparison of policy networks and baseline methods

5. Discussion

The experimental results indicate that while the REINFORCE algorithm is straightforward to implement, its high variance can impede rapid convergence, particularly over shorter training periods. In contrast, both the Actor-Critic and Advantage Actor-Critic (A2C) methods benefit significantly from variance reduction techniques such as baseline subtraction and the utilization of advantage estimates. These modifications lead to more stable and efficient policy updates.

The extensive grid search experiments underscore the critical role played by hyperparameter tuning. Variations in learning rate, network architecture, and update ratio markedly influence the convergence behavior and final performance of each algorithm. The grid search results not only demonstrated that the optimal hyperparameters vary between methods, but they also highlighted that even minor changes in these settings can yield substantial improvements in performance.

Moreover, the comparative analysis between policy gradient methods (Assignment 2) and classical approaches (Assignment 1) reveals that modern policy gradient techniques, when properly optimized, tend to exhibit superior performance and faster convergence on the CartPole-v1 environment. The classical methods, while effective in their own right, tend to lag behind in terms of both stability and sample efficiency.

Overall, the discussion of these results emphasizes that while all methods possess distinct merits, the incorporation of advantage estimates in both the Actor-Critic and A2C algorithms contributes significantly to reducing the variance of gradient updates. This leads to enhanced learning stability and improved final reward performance. However, challenges remain in terms of achieving higher sample efficiency and extending these methods to more complex, higher-dimensional environments.

These observations pave the way for future research directions aimed at integrating more advanced baseline techniques, exploring deeper network architectures, and adapting these methods to continuous action spaces.

6. Future Work

Future work may extend the present study in several important directions. One promising avenue is to apply these policy gradient methods to environments with continuous action spaces or higher-dimensional state representations, such as those encountered in robotics or autonomous driving. Another research direction is the exploration of more sophisticated baseline and variance reduction techniques. For instance, incorporating multi-step returns, off-policy corrections, or adaptive baselines could further enhance sample efficiency and stability. Moreover, employing deeper network architectures, such as convolutional neural networks or recurrent neural networks, may improve the generality and robustness of the learning algorithms in more complex and dynamic environments. Finally, systematic studies on the transferability of hyperparameter settings across different tasks would be beneficial for developing more automated methods for policy optimization.

7. Conclusion

In this study, we have conducted a comprehensive analysis of policy gradient methods by evaluating the REINFORCE, basic Actor-Critic, and Advantage Actor-Critic (A2C) algorithms in the CartPole-v1 environment. The experiments, performed over both long (1×10^6 steps) and short (1×10^5 steps) training regimes, have illustrated the benefits and drawbacks of each approach. In particular, while the REINFORCE algorithm is simple to implement, its high variance poses challenges for efficient learning. The incorporation of a critic in the basic Actor-Critic framework and the use of advantage estimates in A2C significantly reduce this variance, leading to more stable and efficient convergence.

Furthermore, an extensive grid search conducted over the 1×10^5 steps experiments has emphasized the critical role of hyperparameter tuning. Optimal configurations can dramatically improve performance, as evidenced by the substantial improvements observed in the final reward metrics. The comparative analysis with classical methods from Assignment 1 further validates that modern policy gradient techniques, particularly when augmented with advantage estimation, outperform traditional approaches in both convergence speed and final performance.

Overall, the findings from this work underscore the potential of using policy gradient methods in reinforcement learning, particularly in applications where stability and sample efficiency are critical. Future research may extend these methods to more complex environments and incorporate advanced variance reduction and network architectures to further enhance performance.

8. References

References

[1] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

[2] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256.