Computing, Performance and Communication Systems (2026)
Clausius Scientific Press, Canada
DOI: 10.23977/cpcs.2026.100101
ISSN 2371-8889 Vol. 10 Num. 1

A Review of Hyperparameter Tuning Methods for Reinforcement Learning——Taking DQN, PPO, and A3C Algorithms as Examples

Hongyuan Liu

School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China
3102772267@qq.com

Keywords: Reinforcement Learning; Hyperparameter Tuning; DQN; PPO; A3C; Parameter Setting Guide

Abstract: The performance of reinforcement learning algorithms is highly dependent on hyperparameter configuration. Unreasonable hyperparameter settings can easily lead to training non-convergence, slow convergence, or poor policy performance. This paper takes three mainstream algorithms in the field of deep reinforcement learning, namely DQN (Deep Q-Network), PPO (Proximal Policy Optimization), and A3C (Asynchronous Advantage Actor-Critic), as the research objects. It systematically sorts out the key hyperparameters and core action mechanisms of each algorithm, and summarizes the general hyperparameter tuning rules and targeted strategies through literature research. To verify the effectiveness of the tuning methods, small-scale comparative experiments are designed to test the convergence performance and final policy effect of different hyperparameter configurations in classic Gym environments. Based on the experimental results and literature analysis, a practical parameter setting guide for beginners is extracted to reduce the application threshold of reinforcement learning algorithms. The research shows that reasonable hyperparameter tuning can increase the algorithm convergence speed by more than 30% and the final performance by about 15%, among which the learning rate, exploration strategy-related parameters, and regularization coefficients have the most significant impact on algorithm performance.

1. Introduction

As an important branch of machine learning, reinforcement learning learns optimal decision-making strategies through the interaction between agents and the environment, and has achieved breakthrough progress in many fields such as game playing, robot control, and autonomous driving. Different from supervised learning, the training process of reinforcement learning has characteristics such as non-stationarity and sample dependence, making the algorithm extremely sensitive to the selection of hyperparameters. Studies have shown that 68% of industrial implementation failure cases of reinforcement learning are caused by improper hyperparameter configuration, of which 32% directly lead to policy collapse. Therefore, hyperparameter tuning has become a key link for the successful application of reinforcement learning algorithms.

DQN, PPO, and A3C are the three most representative algorithms in the field of deep reinforcement learning, covering three core frameworks: value-based, policy-based, and actor-critic. DQN first combines deep learning with Q-learning to solve the problem of processing high-dimensional state spaces; PPO achieves stable policy updates through trust region constraints and has become the mainstream choice for industrial applications; A3C uses asynchronous multi-threaded parallel training to improve training efficiency and convergence stability. Although the core mechanisms of the three algorithms are different, hyperparameter tuning has always been a core requirement for their performance improvement.

Current research on hyperparameter tuning of reinforcement learning mostly focuses on a single algorithm or general tuning framework, lacking systematic sorting and targeted comparison of mainstream algorithms. At the same time, existing research results are not user-friendly for beginners and difficult to provide clear and operable parameter setting guidance. Based on this, this paper carries out the following work: first, sort out the key hyperparameters and action mechanisms of DQN, PPO, and A3C algorithms; second, summarize the general hyperparameter tuning rules and targeted tuning strategies for each algorithm; third, verify the effectiveness of tuning methods through small-scale experiments; fourth, extract a practical parameter setting guide for beginners. The research results of this paper can provide direct technical references for beginners of reinforcement learning and lay a foundation for hyperparameter tuning research in related fields.

2. Related Work

The core goal of hyperparameter tuning is to find the optimal hyperparameter combination within a given search space to maximize algorithm performance. Traditional hyperparameter tuning methods include grid search, random search, etc. Grid search finds the optimal solution by traversing all parameter combinations, but it is inefficient and difficult to apply to high-dimensional parameter spaces; random search performs better than grid search in most cases with lower computational cost, but lacks guided exploration of the parameter space.

With the in-depth research, adaptive tuning methods have gradually become mainstream, such as Bayesian optimization, Hyperband, Population-Based Training (PBT), etc. Bayesian optimization constructs a probabilistic model based on prior experimental results to guide subsequent parameter selection, significantly improving tuning efficiency; Hyperband combines random search with early stopping strategy to achieve efficient tuning under limited computational resources; PBT dynamically adjusts hyperparameters by training multiple models in parallel, balancing exploration and exploitation, and has been proven to have excellent performance in reinforcement learning tuning. Research by Eimer et al. shows that automatic tuning methods are 10 times cheaper than manual search and can achieve better performance in most environments.

In terms of tuning research for specific algorithms, tuning for DQN mainly focuses on the optimization of exploration rate strategies and experience replay mechanism-related parameters; tuning for PPO focuses on the balance of core parameters such as clipping coefficients and regularization coefficients; tuning for A3C mostly revolves around the number of asynchronous threads and learning rate decay strategies. These studies provide an important theoretical basis for the work of this paper, but lack systematic comparison and integration of the three mainstream algorithms, making it difficult to form a unified tuning cognition and guidance plan.

3. Sorting Out Key Hyperparameters of Mainstream Reinforcement Learning Algorithms

The three algorithms DQN, PPO, and A3C are based on different theoretical frameworks, and the types and action mechanisms of their key hyperparameters are significantly different. This section will sort out the core hyperparameters of each algorithm separately, clarifying their physical meanings and impacts on algorithm performance.

3.1. Key Hyperparameters of DQN Algorithm

The core innovations of DQN lie in the experience replay and target network mechanisms, and its key hyperparameters can be divided into four categories: general optimization parameters, experience replay-related parameters, exploration strategy parameters, and target network update parameters[1].

General optimization parameters mainly include learning rate and batch size. The learning rate controls the update step size of Q-network parameters; an excessively large learning rate can easily lead to training oscillation and non-convergence, while an excessively small one results in extremely slow convergence. The default learning rate of DQN is usually set to 0.00025, which needs to be appropriately reduced in complex environments. The batch size determines the number of samples for each parameter update; an excessively small batch size leads to large variance in gradient estimation, while an excessively large one increases computational cost and reduces update frequency, with common values of 32 or 64.

Experience replay-related parameters include replay memory size and sampling strategy parameters. The replay memory is used to store experience samples of agent-environment interaction, and its size is usually set to 1 million samples to ensure sample diversity. In the prioritized experience replay variant, it is also necessary to set the priority coefficient α and the importance sampling coefficient β. α controls the influence degree of priority (value range 0~1), and β is used to balance bias and variance (usually linearly increasing from 0.4 to 1.0).

The main exploration strategy parameter is the exploration rate ε in the ε-greedy strategy. Its initial value is usually 1.0 (full exploration), and then linearly decays to 0.1 with a decay step of about 1 million steps to achieve a smooth transition from exploration to exploitation.

Target network update parameters include update frequency and discount factor γ. The target network is updated every 10,000 steps to be consistent with the behavior network, so as to reduce the fluctuation of target Q values. The discount factor γ controls the weight of future rewards, with a value range of 0~1, usually set to 0.99. An excessively large γ leads to unstable reward accumulation, while an excessively small one makes the agent lack long-term planning ability.

3.2. Key Hyperparameters of PPO Algorithm

PPO achieves trust region policy optimization by clipping the probability ratio, and its key hyperparameters can be divided into four categories: policy update parameters, advantage estimation parameters, regularization parameters, and general optimization parameters.

The core of policy update parameters is the clipping coefficient ε, which is used to limit the amplitude of policy updates and avoid training instability caused by sudden policy changes. The default value of ε is 0.2; it can be appropriately increased to 0.3 in continuous action environments (such as robot control) and reduced to 0.1 in discrete action environments. In addition, the number of policy update epochs (ppo_epochs) and batch size per epoch are also crucial, usually set to 3~10 update epochs, and the batch size per epoch is adjusted to 512~2048 according to environmental complexity[2].

The main advantage estimation parameter is the discount factor λ of the advantage function, which is used to balance temporal difference error and Monte Carlo estimation, with a value range of 0.9~0.99. A larger λ results in larger variance but smaller bias in advantage estimation, with a default setting of 0.95.

Regularization parameters include entropy regularization coefficient and value function coefficient. The entropy regularization coefficient is used to encourage policy exploration and avoid premature convergence to local optima, with a value range of 0.01~0.05; a larger coefficient is required in complex environments. The value function coefficient is used to balance policy loss and value loss, usually set to 0.5.

General optimization parameters include learning rate and gradient clipping threshold. The learning rate of PPO is usually set to 3e-4 with a linear decay strategy; the gradient clipping threshold is used to prevent gradient explosion, usually set to 0.5 or 1.0.

3.3. Key Hyperparameters of A3C Algorithm

The core advantage of A3C lies in asynchronous multi-threaded training, and its key hyperparameters can be divided into four categories: parallel training parameters, learning rate-related parameters, regularization parameters, and reward processing parameters.

Parallel training parameters mainly refer to the number of asynchronous threads, usually set to 4~8. Too few threads make it difficult to exert parallel advantages, while too many increase computational resource consumption and lead to parameter update conflicts. The trajectory length (t_max) of each thread is also crucial, controlling the frequency of parameter updates after each thread collects samples, usually set to 20~50 steps[3].

Learning rate-related parameters include initial learning rate and decay strategy. The initial learning rate of A3C is usually set to 0.001 with an exponential decay strategy, decaying to 0.96 times the original every 100,000 steps to achieve stable convergence.

The main regularization parameter is the entropy regularization coefficient, which, similar to PPO, is used to balance exploration and exploitation, with a value range of 0.01~0.02. In addition, A3C usually adopts a gradient clipping strategy, with a clipping threshold set to 40.0 to avoid gradient explosion.

Reward processing parameters mainly refer to the reward normalization coefficient. By standardizing rewards to a distribution with a mean of 0 and a standard deviation of 0.01, the variance of policy updates is reduced and convergence speed is improved.

4. Summary of Hyperparameter Tuning Rules

Based on literature research and practical tuning experience, hyperparameter tuning follows the basic principles of "core first, secondary second; global first, local second; stability first, optimization second". This section will discuss from two aspects: general tuning rules and targeted tuning strategies for each algorithm.

4.1. General Tuning Rules

The learning rate is the core hyperparameter of all reinforcement learning algorithms with the highest tuning priority. It is recommended to adopt a tuning method of "logarithmic scale search + decay strategy". The initial search range is set to 1e-4~1e-2, and then fine-tuned after determining the approximate optimal range. For most algorithms, adopting a linear or exponential decay learning rate strategy can significantly improve convergence stability[4].

Balancing exploration and exploitation is a key goal of reinforcement learning tuning. For the ε-greedy strategy, a strategy of "initial high exploration, gradual decay" should be adopted, and the decay rate should be adjusted according to environmental complexity: fast decay can be used in simple environments, while slow decay is required in complex environments to ensure sufficient exploration. For the entropy regularization strategy, the coefficient should be dynamically adjusted according to the policy entropy during training: increase the coefficient when the entropy is too low (insufficient exploration) and decrease it when the entropy is too high (slow convergence).

Data preprocessing has a significant impact on tuning effects. Unconditionally performing state space normalization and reward scaling can effectively improve convergence speed and performance. State normalization adopts standardization with a mean of 0 and a standard deviation of 1, and reward scaling uses clip operation to limit the range within [-10, 10] to avoid interference from extreme rewards on training. It should be noted that rewards can only be scaled, not translated as a whole, otherwise the nature of the return function will be changed.

The recommended tuning order follows the priority of "learning rate → exploration strategy parameters → regularization coefficients → other secondary parameters". The control variable method is adopted, adjusting only one parameter at a time, and determining the optimal parameter value by comparing the convergence speed and final performance of the training curve. After preliminary tuning in small-scale datasets or simple environments, fine-tuning is performed in large-scale datasets or complex environments.

4.2. Targeted Tuning Strategies for Each Algorithm

The tuning focus of DQN lies in the exploration rate strategy and experience replay mechanism. For the exploration rate ε, in visual input environments such as Atari, it is recommended to adopt a linear decay strategy, decaying from 1.0 to 0.01 with a decay step of 2 million steps; in simple discrete action environments, the final ε can be set to 0.1 with the decay step reduced to 500,000 steps. For the experience replay mechanism, uniform sampling can be used in simple environments, while prioritized experience replay is recommended in complex environments with α set to 0.6 and β linearly increasing. The target network update frequency can be adjusted according to environmental stability: the update frequency can be appropriately reduced in stable environments (such as updating every 15,000 steps) and increased in unstable environments (such as updating every 5,000 steps).

The tuning focus of PPO lies in the balance between the clipping coefficient ε and regularization coefficients. In continuous action environments (such as robot gait control), ε is recommended to be set to 0.2~0.3 to ensure sufficient policy exploration; in discrete action environments, ε is set to 0.1~0.2 to avoid overly conservative policy updates. There is an interactive influence between the entropy regularization coefficient and the value function coefficient, so joint tuning is recommended: increase the entropy regularization coefficient when convergence is slow in the early training stage, and increase the value function coefficient when performance fluctuates greatly in the late training stage. The number of policy update epochs is recommended to be adjusted according to the batch size: reduce the number of update epochs (3~5 epochs) when the batch size is large, and increase it (5~10 epochs) when the batch size is small.

The tuning focus of A3C lies in the number of asynchronous threads and the learning rate decay strategy. The number of threads should be adjusted according to hardware resources: set to 4 when the number of CPU cores is small, and 8 when the number of cores is sufficient. Too many threads will not significantly improve performance but increase resource consumption. The learning rate decay strategy needs to match the number of threads: the decay rate can be appropriately accelerated when the number of threads is large to avoid parameter update conflicts, and slowed down when the number of threads is small to ensure sufficient convergence. Reward normalization is the key to stable training of A3C, and it is recommended to adopt standardization in all environments, especially in sparse reward environments, which can significantly improve the convergence probability.

5. Small-Scale Experimental Verification

To verify the effectiveness of the above tuning rules and strategies, small-scale comparative experiments are designed. Three classic Gym environments, namely CartPole-v1, MountainCar-v0, and HalfCheetah-v4, are selected to test the performance differences of DQN, PPO, and A3C algorithms under default parameters and tuned parameters respectively.

5.1. Experimental Setup

The experimental environments cover three types: simple discrete action (CartPole-v1), medium-difficulty discrete action (MountainCar-v0), and complex continuous action (HalfCheetah-v4) to comprehensively verify the generality of the tuning strategy. The experimental parameter settings are as follows: the default parameters of DQN adopt the standard configuration in the literature, and the tuned parameters are adjusted according to the strategy in Section 4.2; the default parameters of PPO and A3C adopt the default configuration of the Stable Baselines3 library, and the tuned parameters are also optimized based on the strategy proposed in this paper. Each experiment sets 5 random seeds, runs 1 million steps, and records three evaluation indicators: average reward, convergence steps, and final performance[7].

5.2. Experimental Results and Analysis

In the CartPole-v1 environment, the convergence steps of DQN after tuning are reduced from 250,000 steps to 170,000 steps, with a convergence speed improvement of 32%, and the final average reward is increased from 470 to 495, with a performance improvement of 5.3%. The key to tuning is to reduce the final value of the exploration rate ε from 0.1 to 0.05 and adopt prioritized experience replay (α=0.6, β linearly increasing).

In the MountainCar-v0 environment, the convergence probability of PPO after tuning is increased from 60% to 100%, the convergence steps are reduced from 300,000 steps to 210,000 steps, with a convergence speed improvement of 30%, and the final average reward is increased from -120 to -95, with a performance improvement of 20.8%. The core of tuning is to adjust the clipping coefficient ε from 0.2 to 0.25, increase the entropy regularization coefficient from 0.01 to 0.03, and adopt a linear learning rate decay strategy.

In the HalfCheetah-v4 environment, the final average reward of A3C after tuning is increased from 4500 to 5200, with a performance improvement of 15.6%, and the convergence steps are reduced from 400,000 steps to 280,000 steps, with a convergence speed improvement of 30%. The key to tuning is to increase the number of threads from 4 to 8, adjust the learning rate decay coefficient from 0.96 to 0.98, and perform standardization on rewards.

The experimental results show that the hyperparameter tuning strategy proposed in this paper can effectively improve the convergence speed and final performance of the three algorithms, and the improvement effect is more significant in medium-difficulty and complex environments. At the same time, the experiments also verify the decisive influence of core hyperparameters (learning rate, exploration strategy parameters, regularization coefficients) on algorithm performance, which is consistent with the tuning rules mentioned earlier.

6. Practical Parameter Setting Guide for Beginners

Based on literature research and experimental results, this paper extracts recommended parameter configurations of DQN, PPO, and A3C algorithms in different environment types for beginners, and provides tuning steps and precautions to form an operable practical guide.

6.1. Recommended Parameter Configurations

For simple discrete action environments (such as CartPole-v1, Acrobot-v1): Recommended parameters for DQN are learning rate 0.00025, batch size 32, replay memory size 1 million, ε initially 1.0 linearly decaying to 0.05 (decay steps 1 million), target network update frequency 10,000 steps, γ=0.99; Recommended parameters for PPO are learning rate 3e-4 (linear decay), ε=0.15, ppo_epochs=5, batch size 1024, λ=0.95, entropy regularization coefficient 0.02, value function coefficient 0.5; Recommended parameters for A3C are number of threads 4, initial learning rate 0.001 (exponential decay, ×0.96 every 100,000 steps), t_max=20, entropy regularization coefficient 0.01, reward standardization, gradient clipping threshold 40.0.

For medium-difficulty discrete action environments (such as MountainCar-v0, LunarLander-v2): Recommended parameters for DQN are learning rate 0.0001, batch size 64, prioritized experience replay (α=0.6, β linearly increasing from 0.4 to 1.0), ε initially 1.0 linearly decaying to 0.01 (decay steps 2 million), target network update frequency 5,000 steps, γ=0.99; Recommended parameters for PPO are learning rate 2e-4 (linear decay), ε=0.25, ppo_epochs=8, batch size 2048, λ=0.98, entropy regularization coefficient 0.03, value function coefficient 0.5; Recommended parameters for A3C are number of threads 6, initial learning rate 0.001 (exponential decay, ×0.98 every 100,000 steps), t_max=30, entropy regularization coefficient 0.015, reward standardization, gradient clipping threshold 40.0.

For complex continuous action environments (such as HalfCheetah-v4, Hopper-v4): Recommended parameters for DQN (it is recommended to use DQN variants such as DDPG) are learning rate 0.0001, batch size 64, replay memory size 1 million, ε initially 1.0 exponentially decaying to 0.1 (γ=0.995), target network update frequency 10,000 steps, γ=0.99; Recommended parameters for PPO are learning rate 3e-4 (linear decay), ε=0.3, ppo_epochs=10, batch size 4096, λ=0.99, entropy regularization coefficient 0.05, value function coefficient 0.5; Recommended parameters for A3C are number of threads 8, initial learning rate 0.0008 (exponential decay, ×0.98 every 100,000 steps), t_max=50, entropy regularization coefficient 0.02, reward standardization, gradient clipping threshold 40.0.

6.2. Tuning Steps and Precautions

The recommended tuning steps follow the following four steps: Step 1, Environment Verification and Preprocessing: visualize the environment, verify the rationality of states and rewards, and perform state normalization and reward scaling; Step 2, Baseline Testing: run experiments with recommended parameters and record the training curve as the baseline; Step 3, Core Parameter Tuning: tune one by one according to the order of learning rate → exploration strategy parameters → regularization coefficients using the control variable method; Step 4, Fine Optimization: adjust secondary parameters and verify the stability of the parameter combination.

Precautions: First, real-time monitor the reward curve, loss function, and policy entropy during training. If the reward curve fluctuates sharply, researchers or practitioners should prioritize checking the learning rate and gradient clipping parameters; if the algorithm exhibits slow convergence, they should prioritize adjusting the exploration strategy-related parameters; if the final policy performance is poor, they should prioritize optimizing the regularization coefficients. Second, avoid adjusting multiple parameters at the same time, which will make it impossible to locate key influencing factors. Third, attach importance to the reproducibility of parameters and record the parameter configuration and random seeds of all experiments. Fourth, when resources are limited, automatic tuning tools such as Ray Tune's PBT algorithm can be used to improve tuning efficiency[5].

7. Conclusion and Future Work

This paper systematically sorts out the key hyperparameters of three mainstream reinforcement learning algorithms: DQN, PPO, and A3C, summarizes the general hyperparameter tuning rules of "core first, secondary second; global first, local second; stability first, optimization second", and proposes targeted tuning strategies for each algorithm. The effectiveness of the tuning strategy is verified through small-scale experiments. The experimental results show that reasonable hyperparameter tuning can increase the algorithm convergence speed by more than 30% and the final performance by about 15%. Based on the experimental results and literature analysis, a practical parameter setting guide for beginners is extracted, covering different difficulty levels of environment types, reducing the application threshold of reinforcement learning algorithms.

Future research can be carried out in the following aspects: first, expand the experimental scope to verify the generality of the tuning strategy in more complex environments; second, combine automatic tuning algorithms to develop lightweight tuning tools for beginners; third, conduct in-depth research on the interaction mechanism between hyperparameters to establish a more accurate tuning model[6].

References

[1] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 (2015): 529-533.

[2] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[3] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. PmLR, 2016.

[4] Eimer, Theresa, Marius Lindauer, and Roberta Raileanu. "Hyperparameters in reinforcement learning and how to tune them." International conference on machine learning. PMLR, 2023.

[5] Liaw, Richard, et al. "Tune: A research platform for distributed model selection and training." arXiv preprint arXiv:1807.05118 (2018).

[6] Wang, Linnan, Rodrigo Fonseca, and Yuandong Tian. "Learning search space partition for black-box optimization using monte carlo tree search." Advances in Neural Information Processing Systems 33 (2020): 19511-19522.

[7] Huang, Shengyi, et al. "Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms." Journal of Machine Learning Research 23.274 (2022): 1-18.