A Comparative Study of DE, GA and ES for Evolutionary Reinforcement Learning of Neural Networks in Pendulum Task

Hidehiko Okada
Faculty of Information Science and Engineering
Kyoto Sangyo University
Kyoto, Japan
hidehiko@cc.kyoto-su.ac.jp

***Abstract*—Reinforcement learning of neural networks requires gradient-free algorithms as labeled training data are not available. Evolutionary algorithms are well-suited for this purpose since they do not rely on gradients. However, the success of training neural networks with evolutionary algorithms is contingent on the careful selection of appropriate algorithms, given the numerous algorithmic variations available. In this study, the author evaluates the efficacy of Differential Evolution (DE), Genetic Algorithm (GA), and Evolution Strategy (ES) for the reinforcement learning of neural networks, utilizing a pendulum control task. The experimental results indicate that DE exhibits statistically significant superiority over GA and ES. While GA performs better than ES, this difference is not statistically significant. The study highlights DE's ability to effectively balance between exploratory and exploitative search, adapting to the problem at hand. Based on these findings, it is suggested that an algorithm possessing such characteristics is better suited for evolutionary reinforcement learning of neural networks.**

Keywords—Differential Evolution, Genetic Algorithm, Evolution Strategy, neuroevolution, reinforcement learning

I. Introduction

Evolutionary algorithms [1] provide a viable approach for reinforcement learning of neural networks, as they do not rely on gradients. Q-learning [2], a popular reinforcement learning method, requires obtaining the reward r(t) for action a(t) at state s(t), to determine the next action a(t+1), where t denotes the time step. However, evolutionary algorithms only necessitate evaluating the reward after the completion of an episode, negating the need for designing appropriate rewards for all state-action pairs. However, given the plethora of algorithmic variations available, it is crucial to carefully select the most appropriate algorithm for successful neural network training. Previous work by the author [3,4] evaluated Genetic Algorithm (GA) [5] and Evolution Strategy (ES) [6]. In this paper, the author expands on the earlier work by evaluating Differential Evolution (DE) [7] and comparing the performance of the three algorithms.

II. Pendulum Task

In this study, the OpenAI Gym pendulum balancing task[1,2] was utilized. The objective of this task is to swing the pendulum up, maintaining it in an upright position. The author modified the system such that the control task initiates with the pendulum in a downward position (Fig. 1(a)) and the goal is to make and keep the pendulum upright (Fig. 1(b)). Furthermore, the author adjusted the system such that the control task starts with the pendulum having zero angular velocity.

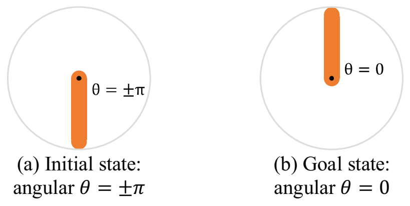

θ = ±π
θ = 0
(a) Initial state: angular $\theta = \pm\pi$
(b) Goal state: angular $\theta = 0$

Fig. 1. OpenAI Gym pendulum balancing task[1,2]

Each episode of the task consists of 200 timesteps, with the controller observing the current state and selecting the appropriate action in each step. The observation comprises cos(θ), sin(θ), and the angular velocity of the pendulum. The action corresponds to the torque applied to the pendulum, with a torque range of [-2.0, 2.0]. The controller needs to swing the pendulum such that the force of gravity assists in increasing the angular velocity to a level sufficient for the pendulum to climb over.

The author defines the fitness of a neural network controller as follows.

$$\mathrm{Fitness} = \frac{1}{200}\sum_{t=1}^{200}(1-\mathrm{Error}(t)) \quad (1)$$

$$\mathrm{Error}(t) = \frac{|\theta(t)|}{\pi} \quad (2)$$

θ(t) represents the angular position of the pendulum at each time step t. The initial state is such that Error(t)=|±π|/π=1, resulting in 1-Error(t)=0. Conversely, in the goal state, Error(t)=0/π=0, so that 1-Error(t)=1. The fitness score increases with a decrease in Error(t) for a larger number of time steps. Therefore, a controller performs better if it can quickly stabilize the pendulum in the upright position and maintain it for a longer duration.

III. Neural Networks

The controller in this study is implemented using a three-layered feedforward neural network, also known as a multilayer perceptron (MLP). Fig. 2 illustrated the topology of the MLP.

1. https://gym.openai.com/envs/Pendulum-v0/
2. https://github.com/openai/gym/blob/master/gym/envs/classic_control/pendulum.py

The hyperbolic tangent (tanh) function is adopted as the activation function for the MLP units, which produces an output value within the range of [-1.0, 1.0].

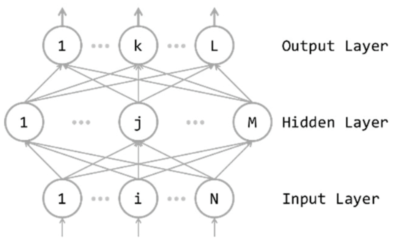


Fig. 2. Neural network topology in this study.

The MLP serves as the policy function, which maps the observation of the current state to an action, i.e., action(t) = F(observation(t)). The input layer of the MLP consists of three units that receive the values of cos(θ), sin(θ), and the angular velocity, respectively. The output layer has one unit that generates the torque applied to the pendulum. To ensure that the torque falls within the range [-2.0, 2.0], the output value is multiplied by 2.0.

IV. Evolutinary Training of Neural Networks

The MLP depicted in Fig. 2 is composed of M+L units and NM+ML connections, resulting in a total of M+L+NM+ML parameters. The dimensionality of the parameter space is denoted by D=M+L+NM+ML. The training of the MLP is equivalent to optimizing a D-dimensional real vector, represented by $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$, where each x_i corresponds to one of the D parameters in the MLP.

Neuroevolution, the process of training neural networks using evolutionary algorithms, has been widely researched [8]. In this study, the D-dimensional vector $\boldsymbol{x}$ representing the parameters of the MLP is optimized using either DE, GA, or ES. These algorithms treat $\boldsymbol{x}$ as a chromosome and apply evolutionary operators to it. The fitness of $\boldsymbol{x}$ is evaluated using (1) and (2).

V. Experimant

The MLP in this study has 3 input units (N) and 1 output unit (L). The number of hidden units (M) is set to 8, 16, or 32, which is consistent with previous work [3,4]. Therefore, the dimensionality of the parameter vector (D) is 41, 81, or 161 for M=8, 16, or 32, respectively.

In this experiment, DE reproduces offsprings by the rand/1/bin method [9]. DE adaptively balances explorative and exploitive search to the optimization problem at hand because it reproduces offspring solutions by utilizing the difference between two solutions in the current population. In the initial generation, the difference is large and becomes adaptively smaller as the number of generations increases. On the other hand, GA reproduces offspring through crossover and mutation, where the blend crossover [10] for continuous chromosomes is adopted, promoting exploration more than exploitation. ES reproduces offspring by perturbation within a fixed step size, promoting exploitation more than exploration. Thus, this study compares three types of search: GA that emphasizes global search, ES that emphasizes local search, and ES that balances global and local searches.

Table 1 presents the hyperparameter configurations for the algorithms, which were determined empirically in preliminary experiments. In Table 1, setting (a) is designed for experiments with 10 offsprings per generation and 500 generations, and setting (b) is designed for experiments with 50 offsprings per generation and 100 generations. The total number of fitness evaluations was 50,000 for each trial in both cases.

TABLE I. Hyperparameters of DE, GA and ES

Algorithm	Hyperparameter	Value
DE	#Parents	5
	Step size	[-1.0, 1.0]
GA	α for blend crossover	0.5
	#Elites	(a)2 (b)10
	Mutation probability	1/D
DE	Scaling factor (F)	(a)0.1 (b)0.2
	Crossover rate (CR)	(a)0.5 (b)0.9

Each of the three algorithms was applied 11 times under the same settings, yielding 11 fitness scores for the best solution for each setting. With three variations of the number of hidden units (8, 16, 32) and two variations of population sizes (10, 50), a total of six combinations of settings were tested. Thus, 66 fitness scores were obtained for each algorithm (11 runs × 6 settings). These data were used to compare the performance of the three algorithms.

VI. Result

Tables 2-4 presents the fitness scores obtained from the 11 runs under the same settings for DE, GA and ES, respectively. Tables 3 and 4 are cited from previous papers by the author [3,4].

TABLE II. Fitness Scores by Differential Evolution

(a) 10 offsprings, 500 generations.

Units	Best	Median	Worst
8	0.832	0.825	0.608
16	0.830	0.828	0.731
32	0.832	0.824	0.796

(b) 50 offsprings, 100 generations.

Units	Best	Median	Worst
8	0.832	0.828	0.801
16	0.832	0.822	0.812
32	0.831	0.819	0.783

TABLE III. Fitness Scores by Genetic Algorithm [3]

(a) 10 offsprings, 500 generations.

Units	Best	Median	Worst
8	0.833	0.827	0.609
16	0.834	0.830	0.605
32	0.834	0.832	0.628

(b) 50 offsprings, 100 generations.

Units	Best	Median	Worst
8	0.817	0.768	0.737
16	0.816	0.769	0.732
32	0.803	0.776	0.742

TABLE IV. Fitness Scores by Evolution Strategy [4]

(a) 10 offsprings, 500 generations.

Units	Best	Median	Worst
8	0.829	0.612	0.520
16	0.833	0.823	0.579
32	0.832	0.831	0.613

(b) 50 offsprings, 100 generations.

Units	Best	Median	Worst
8	0.833	0.825	0.583
16	0.833	0.832	0.612
32	0.833	0.831	0.586

By comparing these tables, the following findings are observed.

(1) The best scores are mostly comparable among DE, GA, and ES, ranging from 0.830 to 0.834. Notably, when using setting (b), GA yields significantly lower scores, ranging from 0.803 to 0.817. These findings suggest that GA may require a greater number of generations than offspring to reach optimal performance.

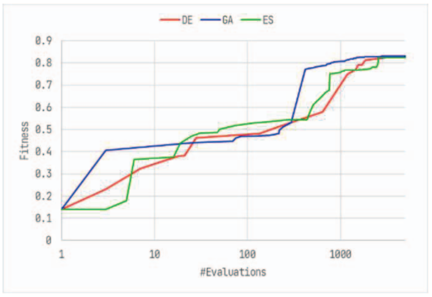
(2) The median scores achieved by DE are close to the best scores. ES shows a similar trend, except for setting (a) and M=8 where the median score is 0.612 and the best score is 0.829. In contrast, for GA, the differences between the median and best scores are more significant than those of DE and ES, particularly with setting (b). These findings again suggest that GA may require a greater number of generations than offsprings.

(3) DE exhibits the highest worst scores of the three methods, whereas ES shows the smallest. This suggests that DE is more resilient to the influence of randomly generated initial solutions, enabling it to robustly search for better solutions. In contrast, ES appears to be more sensitive to the quality of the initial solutions.

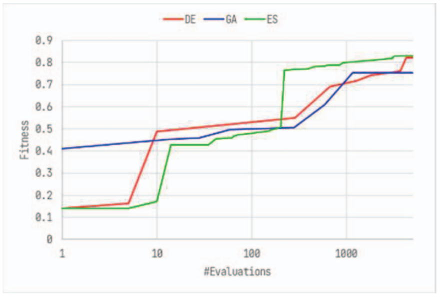
The results indicate that DE outperforms both GA and ES. To confirm the statistical significance of this difference, the author conducted a Wilcoxon signed-rank sum test, revealing that DE is significantly superior to both GA and ES (p=3.25E-4 and p=1.75E-3, respectively). While GA performs better than ES, the difference is not statistically significant (p=0.0938).

Fig. 3 shows the learning curves for DE, GA, and ES, representing the median results of 11 runs. The curves for DE, GA, and ES exhibit similar shapes, with most showing two distinct stages of learning. Initially, fitness scores remained relatively flat, starting at around 0.1-0.2 and persisting for approximately 10 evaluations. Subsequently, scores increased rapidly within 10-50 evaluations, reaching around 0.4-0.5. After this, the rate of improvement became more gradual, eventually reaching around 0.6. Finally, fitness scores increased rapidly again at around 500 evaluations, reaching values of approximately 0.75-0.8.

Initially, the torque output from the MLPs was independent of the pendulum state, often remaining fixed at either the maximum (2.0) or the minimum (-2.0) values. As a result, the pendulum was never lifted above the horizontal position. At a fitness score of approximately 0.4-0.5, the MLPs were able to initiate swinging motion, but unable to maintain the pendulum in an upright position, causing it to rotate instead. Finally, when the fitness score reached around 0.8, the MLPs were capable of both initiating and maintaining pendulum swings in an upright position. Supplementary videos showcasing the pendulum's behavior by the trained/untrained MLPs are available.[4]



(a) 10 offsprings, 500 generations.



(b) 50 offsprings, 100 generations.

Fig. 3. Learning curves for MLP with 16 hidden units.

VII. Conclution

This study compares the performance of DE, GA, and ES in evolutionary reinforcement learning of multilayer perceptrons for balancing a pendulum. The experiment includes six configurations with varying numbers of hidden units in the MLP (8, 16, or 32) and the number of offsprings reproduced by each evolutionary algorithm (10 or 50). The learning curves showed that all algorithms trained the MLPs in two stages. Results from the statistical test revealed that DE exhibited statistically significant superiority over GA and ES. While GA performed better than ES, this difference was not statistically significant. These findings suggest that an algorithm that can balance exploration and exploitation adaptively may be better suited for evolutionary reinforcement learning of neural networks.

To expand on this study, future work will involve evaluating and comparing additional evolutionary algorithms by implementing them on the same task.

Acknowledgment

The author conducted this study as an official researcher of Kyoto Sangyo University.

References

[1] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, 1996.

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

[3] H. Okada, "An evolutionary approach to reinforcement learning of neural network controllers for pendulum tasks using genetic algorithms," Inter. J. of Scientific Research in Computer Science and Engineering, Vol. 11, Issue 1, pp. 40–46, 2023.

[4] H. Okada, "Evolutionary reinforcement learning of neural network controller for pendulum task by evolution strategy," Int. J. of Scientific Research in Computer Science and Engineering, Vol. 10, Issue 3, pp. 13–18, 2022.

[5] H. P. Schwefel, "Evolution strategies: a family of non-linear optimization techniques based on imitating some principles of organic evolution," Annals of Operations Research, Vol. 1, pp. 165–167, 1984.

[6] D. E. Goldberg, and J. H. Holland, "Genetic algorithms and machine learning," Machine Learning, Vol. 3, No. 2, pp. 95–99, 1988.

[7] R. Storn, and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," J. of Global Optimization, Vol. 11, pp. 341–359, 1997.

[8] F. Dario, P. Dürr, and C. Mattiussi. "Neuroevolution: from architectures to learning." Evolutionary Intelligence, Vol. 1, pp. 47–62, 2008.

[9] R. Storn, K. Price, and J. A. Lampinen, Differential Evolution – A Practical Approach to Global Optimization, Springer, 2005.

[10] L. J. Eshelman, and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," Foundations of Genetic Algorithms, Vol. 2, pp.187–202, 1993.

4. https://www.cc.kyoto-su.ac.jp/~hidehiko/csce2023/