Training Neural Networks with {-1,1} Weights by Genetic Algorithm

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

Abstract: The author previously reported an experimental result of evolutionary reinforcement learning of binary neural network controllers. In the previous study, the controller was trained by Evolution Strategy. In this study, the author experimentally applies Genetic Algorithm, instead of ES, and the results were compared between GA and ES. In both studies, the same Acrobot control task is utilized, and the same three-layer feedforward neural network is adopted. The difference lies in the training algorithm. The findings from this study are (1) GA trained the controller better than ES ($p<.01$), (2) increasing the population size, rather than the number of generations, improved performance more in GA ($p < .01$), and (3) the optimal number of hidden units for the binary MLP was 128 among the choices of 16, 32, 64, 128 and 256, which was consistent with the previous study using ES.

Keywords: evolutionary algorithm, genetic algorithm, binary neural network, neuroevolution, reinforcement learning.

1. Introduction

The author previously reported an experimental result of evolutionary reinforcement learning of neural network controllers for the Acrobot task [1]. In the previous study, the weights of connections between neurons are not real numbers but binary {-1,1}, and Evolution Strategy (ES) [2,3] was adopted as the training algorithm. In this paper, the author reports another experimental result for the same training task where Genetic Algorithm (GA) [4,5] is adopted. The topology of the neural network, the activation function, and the control method for the Acrobot system using the neural network are kept the same as in the previous study. The training result obtained through GA is compared with that obtained through ES. Both GA and ES suit for handling binary genotypes; however, differences in the reproduction methods lead to distinct characteristics in the solution search process. While ES excels in local search, GA is more effective in global search. By comparing the experimental results of ES and GA, this study investigates which type of search—local or global—is more effective for the given training task.

2. Acrobot Control Task

The Acrobot control task provided at OpenAI Gym was employed in the previous study [1]. This study employs the same task. Figure 1 shows a screenshot of the system. The webpage for this system describes as follows[1]; *The system consists of two links connected linearly to form a chain, with one end of the chain fixed. The joint between the two links is actuated. The goal is to apply torques on the actuated joint to swing the free end of the linear chain above a given height while starting from the initial state of hanging downwards.*

Let p_y denote the height of the free end of the linear chain, where the minimum (maximum) value of p_y is 0.0 (1.0) as shown in Figure 2. The goal of the task is originally to achieve $p_y \geq 0.5$, and an episode is finished

[1] https://www.gymlibrary.dev/environments/classic_control/acrobot/

when the goal is achieved or the time step reaches to a preset limit. In this study, the goal is changed so that the free end of the linear chain is kept as high as possible (i.e., let the value of p_y as greater as possible) throughout an episode, where an episode consists of 200 time steps. Besides, the author changed the system so that (i) the control task starts with the state shown in Figure 2(a) where p_y=0.0, and (ii) the applicable torque to the actuated joint is continuous within [-1.0, 1.0] while the torque is originally discrete (either of -1, 0 or 1).

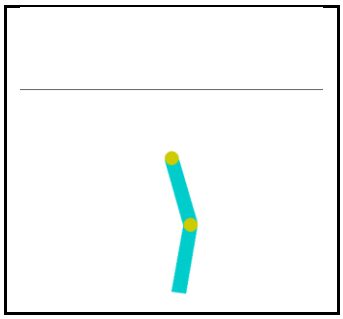
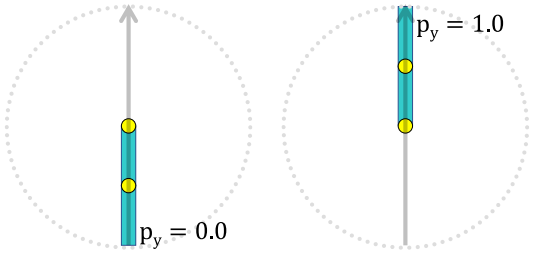


Figure 1: Acrobot system[1].



(a) initial state (b) best state

Figure 2: Initial and best states.

In each step, the controller observes the current state and then determines the action. An observation obtains $\cos(\theta_1)$, $\sin(\theta_1)$, $\cos(\theta_2)$, $\sin(\theta_2)$, and the angular velocity of θ_1 and θ_2, where θ_1 is the angle of the first joint and θ_2 is relative to the angle of the first link[1]. The ranges are $-1.0 \leq \cos(\theta_1), \sin(\theta_1), \cos(\theta_2), \sin(\theta_2) \leq 1.0$, $-4\pi \leq$ angular velocity of $\theta_1 \leq 4\pi$, and $-9\pi \leq$ angular velocity of $\theta_2 \leq 9\pi$ respectively.

In this study, the author defines the fitness of a neural network controller as shown in eq. (1). In eq. (1), $p_y(t)$ denotes the height p_y at each time step t. The fitness score is larger as $p_y(t)$ is larger for more time steps. Thus, a controller fits better as it can keep the free end of the linear chain as higher as possible.

$$\text{Fitness} = \frac{1}{200}\sum_{t=1}^{200} p_y(t) \tag{1}$$

3. Neural Networks with Binary Connection Weights

This study employs a feedforward neural network, known as a multilayer perceptrons (MLP), as the controller to the Acrobot system. Figure 3 illustrates the topology of the network. A single hidden layer is included, and the connection weights are not real numbers but binary, i.e., either -1 or 1. The unit activation function is the hyperbolic tangent (tanh) so that the network outputs real numbers within the range [-1.0, 1.0]. The same network was employed in the previous study; the feedforward calculations are the same as those

described in [1]. In both of this study and the previous one, the MLP serves as the policy function: action(t) = F(observation(t)). The input layer consists of three units (N=6 in Figure 3), each corresponding to the values obtained by an observation. The output layer comprises one unit (L=1 in Figure 3), and its output value is applied as the torque to the pendulum system.

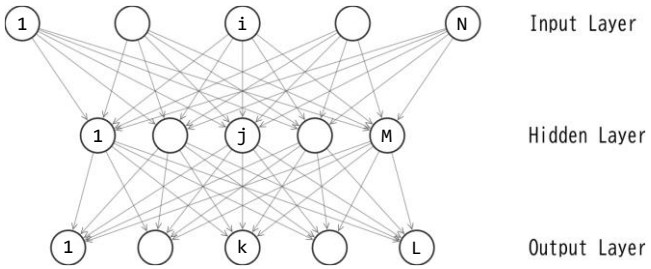


Figure 3. Topology of the MLP.

4. Training of Binary Neural Networks by Genetic Algorithm

The MLP depicted in Figure 3 includes M+L unit biases and NM+ML connection weights, resulting in a total of M+L+NM+ML parameters. Let D represent the quantity M+L+NM+ML. For this study, the author sets N=6 and L=1, leading to D=8M+1. The training of this perceptron is essentially an optimization of the D-dimensional binary vector. Let $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ denote the D-dimensional vector, where each x_i corresponds to one of the D parameters in the perceptron. In this study, each x_i is a binary variable, $x_i \in \{-1,1\}$. By applying the value of each element in $\mathbf{x}$ to its corresponding connection weight or unit bias, the feedforward calculations can be processed.

In the previous study, the binary MLP was trained using ES. In this study, the same MLP was trained using GA. Both of ES and GA treat $\mathbf{x}$ as a chromosome (a genotype vector) and applies evolutionary operators to manipulate it. The fitness of $\mathbf{x}$ is evaluated based on eq. (1), which is the same as in the previous study [1]. Figure 4 illustrates the GA process. In Step 1, D-dimensional binary vectors $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^P$ are randomly initialized where P denotes the population size. A larger value of P promotes exploration more. In Step 2, binary values in each vector $\mathbf{y}^p$ $(p = 1,2,\ldots,P)$ are applied to the MLP and the MLP controls the Acrobot system for a single episode with 200 time steps. The fitness of $\mathbf{y}^p$ is then evaluated with the result of the episode. In Step 3, the loop of evolutionary training is finished if a preset condition is met. A simple example of the condition is the maximum number of fitness evaluations. Elites are the best E vectors among all offsprings evaluated so far, where E denotes the size of elite population. To locally search around good solutions found so far, elite vectors are kept as parent candidates for the crossover. The elite population is empty at first. In Step 4, members in the elite population, $\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^E$, are updated. Step 5 consists of the crossover and the mutation. In Step 5.1, new P offspring vectors are produced by applying the crossover operator to the union of the elite vectors $\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^E$ and the vectors $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^P$ in the current population. A single crossover with two parents produces two new offsprings, where the two parents are randomly selected from the union of $\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^E$ and $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^P$. To produce new P offspring vectors, the crossover is performed $P/2$ times. Each vector in the union of $\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^E$ and $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^P$ can be selected as a parent two or more times (several vectors in the union may not be selected any time). The new P offspring vectors replace the population $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^P$. Crossover operators applicable to binary chromosomes are a single-point crossover, a multi-point crossover and the uniform crossover. The uniform crossover is applied in this study. In Step 5.2, each element in the new offspring vectors is bit-flipped under

the mutation probability pm. A greater pm promotes explorative search more.

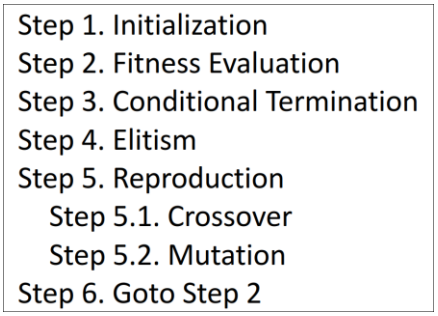


Figure 4. Process of Genetic Algorithm.

4. Experiment

In the previous study using ES, the number of fitness evaluations included in a single run was set to 10,000 [1]. The number of new offsprings generated per generation was either of (a) C=10 and (b) C=50. The number of generations for each case of (a) or (b) was 1,000 and 200 respectively. The total number of fitness evaluations were 10 × 1,000 = 10,000 for (a) and 50 × 200 = 10,000 for (b). The experiments in this study, using GA for the training of binary MLPs, employ the same configurations. The hyperparameter configurations for GA are shown in Table 1. The mutation probability pm is set to 0.01. The values of pm were adjusted based on preliminary experiments. The mutation probability pm is set to $1/D$ where D denotes the length of a binary genotype vector.

Table 1. GA Hyperparameter Configurations.

Hyperparameters	**(a)**	**(b)**
Number of offsprings (C)	10	50
Generations	1,000	200
Fitness evaluations	10×1,000=10,000	50×200=10,000
Number of elites (E)	10×0.2=2	50×0.2=10
Mutation probability (pm)	$1/D$	$1/D$

In both of the two studies, each binary connection weight is either -1 or 1. In the previous study [1], the author tested five variations for the number of hidden units: 16, 32, 64, 128 and 256. The same variations are tested in this study. A binary MLP with either of 16, 32, 64, 128 or 256 hidden units underwent independent training 11 times, which is also consistent with the previous study [1].

Table 2 presents the best, worst, average, and median fitness scores of the trained MLPs across the 11 runs. Each of the two hyperparameter configurations (a) and (b) in Table 1 was applied. A larger value is better in Table 2 with a maximum of 1.0. In order to investigate which of the two configurations (a) or (b) is superior, the Wilcoxon signed-rank test was applied to the 20×2 data points presented in Table 2. This test revealed that configuration (b) is better than configuration (a) with a statistical significance ($p < .01$). Thus, increasing the population size and reducing the number of generations is more favorable than the opposite approach. This result is consistent with that in the previous study [1], which indicates that increasing the population size to promote broader exploration is more effective for training a binary MLP controller using evolutionary algorithms than increasing the number of generations to enhance local search.

Table 2. Fitness Scores among 11 Runs using GA.

	M	Best	Worst	Average	Median
	16	0.384	0.355	0.368	0.369
	32	0.417	0.385	0.408	0.410
(a)	**64**	0.436	0.385	0.415	0.420
	128	0.449	0.415	0.429	0.423
	256	0.449	0.415	0.429	0.423
	16	0.398	0.368	0.386	0.387
	32	0.425	0.396	0.416	0.416
(b)	**64**	0.443	0.418	0.427	0.427
	128	0.451	0.419	0.438	0.438
	256	0.462	0.425	0.443	0.444

Next, the author examines whether there is a statistically significant difference in the performances among the five MLPs with the different numbers of hidden units M. For each M of 16, 32, 64, 128 and 256, the author conducted 11 runs using the configuration (b), resulting in 11 fitness scores. The Wilcoxon rank sum test was applied to the 11×5 data points. The results showed that,

(1) M=16 was significantly worse than any of M=32, 64, 128 and 256 ($p < .01$),
(2) M=32 was significantly worse than any of M=64, 128 and 256 ($p < .01$), and
(3) M=64 was significantly worse than any of M=128 and 256 ($p < .01$).

M=128 was worse than M=256 but the difference was not statistically significant ($p > .05$). Therefore, from the perspective of the trade-off between performance and memory size, the most desirable number of hidden units was found to be 128. This result is also consistent with the findings of previous study [1]. The results of the two studies show that (1) if the number of hidden units is too small, the control capability of the binary MLP is insufficient, and (2) if the number of hidden units is too large, the balance between memory usage and performance will not be optimal.

Figure 5 presents learning curves of the best, median, and worst runs among the 11 runs where the number of hidden units is 128 and the configuration (b) was applied. Note that the horizontal axis of these graphs is logarithmic. The learning curves show similar shapes to those shown in the previous study (Figure 6 in [1]), i.e., the performance of the binary MLP in controlling the Acrobot system increased similarly along with the number of evaluations, regardless of using ES or GA as the training algorithm.

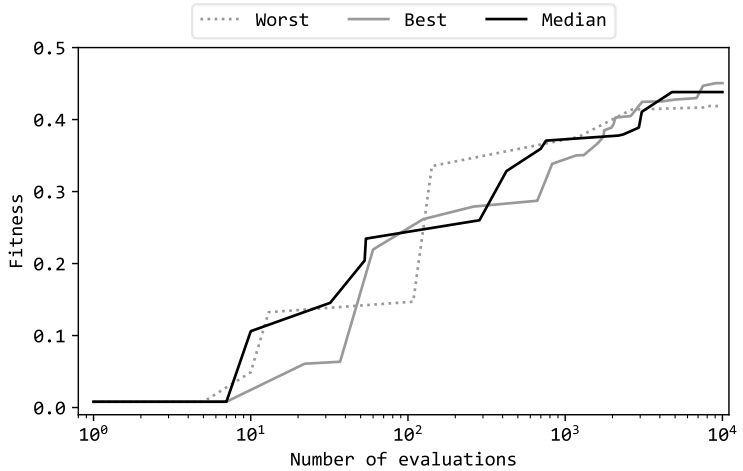
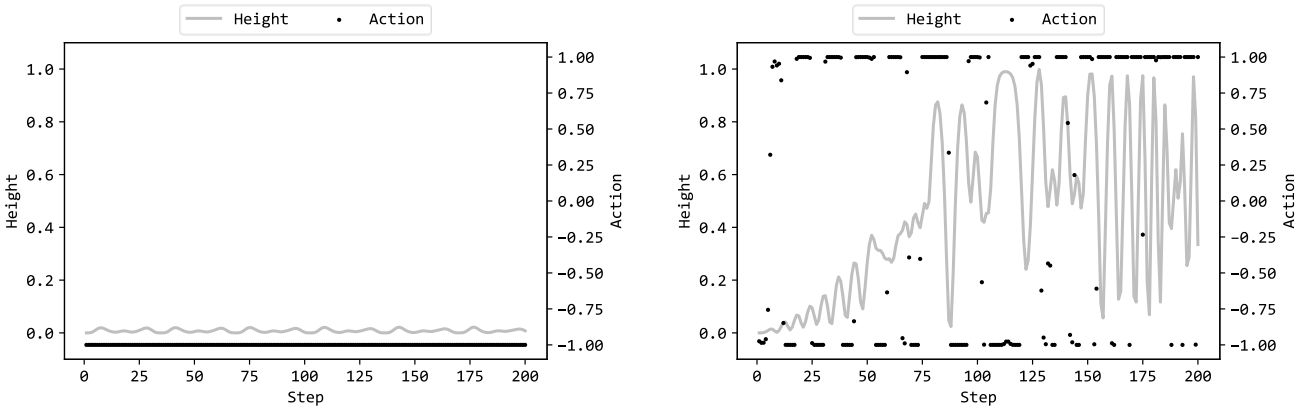


Figure 5. Learning curves of binary MLP with 128 hidden units.

Figure 6(a) illustrates the actions by the MLP and the heights $p_y(t)$ in the 200 steps prior to training, while Figure 6(b) displays the corresponding actions and heights after training. In this scenario, the MLP employed 128 hidden units, and the configuration (b) was utilized. These figures also exhibit a similarity to Figure 7 in the previous report [1], i.e., the MLP trained using GA controlled the Acrobot system similarly to the MLP trained using ES. Supplementary videos[2,3] are provided which demonstrate the motions of the chains controlled by the MLP with 128 hidden units.



(a) Before the training. (b) After the training.

Figure 6. MLP actions and errors in an episode.

5. Comparison of GA and ES

The control performance scores of the binary MLP trained using GA are shown in Table 2 in this paper, while the corresponding scores obtained using ES were presented in the previous paper [1]. Table 3 is a citation of the previous table. Tables 2 and 3 reveal that the values in Table 2 are generally higher. For example, for the configuration (b) and M=128, the median score of the MLP trained using GA (ES) is 0.438 (0.433) according to Table 2 (3), respectively. This indicates that GA trained the MLP more effectively than ES. To test the statistical significance of this difference, the Wilcoxon rank sum test was applied to the data in Tables 2 and 3. It was found that, for both of the configurations (a) and (b), the scores of the MLP trained using GA were significantly higher than those trained with ES ($p < .01$). Given that GA excels at exploration compared to ES, this finding suggests that exploration is more important than local search for this evolutionary training task. This is consistent with the finding described above that increasing the population size rather than the number of generations is more suitable for this task.

6. Conclusion

In this study, Genetic Algorithm was applied to the reinforcement learning of a neural network controller for the Acrobot task, where the connection weights in the neural network are not real numbers but binary (-1 or 1). The findings from this study are summarized as follows:

(1) Prior to training, the MLP was almost unable to move the Acrobot chains, but after training, the MLP successfully swung the chains and raised them to as high a position as possible.

[2] http://youtu.be/Fjsd8OHCaE4
[3] http://youtu.be/0MD1qlRDm0I

Table 3. Fitness Scores among 11 Runs using ES (cited from [1]).

	M	Best	Worst	Average	Median
	16	0.391	0.330	0.363	0.356
	32	0.421	0.376	0.401	0.401
(a)	**64**	0.432	0.245	0.395	0.412
	128	0.440	0.385	0.424	0.428
	256	0.448	0.305	0.418	0.426
	16	0.386	0.347	0.362	0.360
	32	0.419	0.377	0.396	0.394
(b)	**64**	0.440	0.394	0.416	0.413
	128	0.448	0.426	0.435	0.433
	256	0.448	0.426	0.438	0.438

(2) The control performance scores of the MLP trained using GA were compared with those of the MLP trained using ES, and it was found that the GA-trained MLP performed better ($p < .01$). Furthermore, it was also found that increasing the population size, rather than the number of generations, improved performance in GA ($p < .01$). These results suggest that, for the evolutionary training task in this study, broader exploration is more effective in finding better solutions than local search.

(3) The optimal number of hidden units for the binary MLP was found to be 128 among the choices of 16, 32, 64, 128 and 256. This result, obtained using GA, is consistent with the previous result obtained by ES.

The author plans to further apply and evaluate other evolutionary algorithms to the same task and compare the performance.

References

[1] Okada, H. (2024). Training neural networks with {-1,1} weights by evolution strategy. viXra.org. https://vixra.org/pdf/2410.0101v1.pdf

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

[3] Beyer, H.G., & Schwefel, H.P. (2002). Evolution strategies: a comprehensive introduction. Journal Natural Computing, 1(1), 3–52.

[4] Goldberg, D.E., Holland, J.H. (1988). Genetic algorithms and machine learning. Machine Learning 3, 95-99. https://doi.org/10.1023/A:1022602019183

[5] Sastry, K., Goldberg, D., & Kendall, G. (2005). Genetic algorithms. in Burke, E.G. and Kendall, G. (eds.), Search methodologies: Introductory tutorials in optimization and decision support techniques, 97–125.