An Improved Differential Evolution Algorithm and Its Application to Large-Scale Artificial Neural Networks

Tae Jong Choi and Chang Wook Ahn

Department of Electrical and Computer Engineering, Sungkyunkwan University, 2066, Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-Do, Republic of Korea

cwan@skku.edu

Abstract. A new differential evolution (DE) algorithm is presented in this paper. The proposed algorithm monitors the evolutionary progress of each individual and assigns appropriate control parameters depends on whether the individual is successfully evolved or not. We conducted the performance evaluation on CEC 2014 benchmark problems and confirmed that the proposed algorithm outperformed than the conventional DE algorithm. In addition, we apply the proposed DE algorithm as an optimization technique of training large scale multilayer perceptron. We conducted the performance evaluation on an artificial neural network that has approximately 1,000 weights and confirmed again that the proposed algorithm performed better than the conventional DE algorithm. As a result, we proposed a new DE algorithm that has better optimization performance for solving large-scale global optimization problems.

1. Introduction

Training artificial neural networks[1,2] is considered as a multimodal continuous optimization problem. Generally, Back-propagation algorithm is used for this purpose. However, this algorithm has a disadvantage that it has the higher probability to get stuck in local minima. Some researchers attempt to devise a method to overcome this problem by using another optimization technique such as evolutionary algorithms[3-6].

In this paper, we propose a new differential evolution (DE) algorithm and apply this algorithm to training large-scale artificial neural networks as an optimization technique. DE algorithm has three control parameters and these control parameters affect its optimization performance significantly. However, it is the laborious task to apply the trial-and-error approach to find appropriate control parameters depends on the structure of artificial neural networks.

Therefore, the proposed DE algorithm automatically finds and tunes appropriate control parameters. More specifically, the proposed algorithm monitors the evolutionary progress of each individual. If an individual is successfully evolved to the next generation, then the individual is assigned new control parameters tuned by the Gaussian distribution. If an individual is failed to evolve, then the individual is assigned new control parameters tuned by the Cauchy distribution.

We conducted the performance evaluation on CEC 2014 benchmark problems and confirmed that the proposed algorithm outperformed than the conventional DE algorithm. In addition, we apply the proposed DE algorithm as an optimization technique of training large scale multilayer perceptron. We conducted the performance evaluation on an artificial neural network that has approximately 1,000

weights and confirmed again that the proposed algorithm performed better than the conventional DE algorithm.

2. Background

2.1. Artificial Neural Networks

Generally, Back-propagation algorithm[1,2] is used for training artificial neural networks. First, the output value generated from the input layer, the hidden layer, and the output layer is compared with the target value. And adjust the weights from the output layer to the hidden layer through the differential by the difference that occurs. We then adjust the weights by repeating this process up to the input layer. In this process, the chain-rule of the differential is used. Back-propagation algorithm, however, has disadvantages. A typical example is that there is a high probability of convergence to local optima and cannot escape from the saddle-point.

There are studies that use evolutionary algorithms as a way to overcome these shortcomings. Evolutionary algorithms do not need to consider differentiability and the above mentioned disadvantages disappear. For this reason, some researchers have been studying artificial neural network training methods using evolutionary algorithms. However, training methods of artificial neural networks using evolutionary algorithms also have disadvantages. In particular, it is considered that evolutionary algorithms are not suitable for learning artificial neural networks with large-scale weights. The reason for this is that it requires a lot of computational cost to apply vast amounts of data to existing evolutionary algorithms.

Recent study has confirmed that evolutionary algorithms can perform well similar to Back-propagation algorithm through hypothesis that each individual need not use all training data. In other words, unlike previous methods in which individuals learn all training data, this condition can be mitigated by reducing the computational cost and achieving the same level of performance as Back-propagation algorithm.

2.2. Differential Evolution

DE algorithm [7,8,9] contains NP individuals. $\overrightarrow{X_i^G} = \{x_{i,1}^G, x_{i,2}^G, \dots, x_{i,D}^G\}$ denotes a target vector of D components in G generation. Target vector acts as a parent individual in evolutionary algorithms. $\overrightarrow{U_i^G} = \{u_{i,1}^G, u_{i,2}^G, \dots, u_{i,D}^G\}$ denotes a trial vector. Trial vector acts as a child individual. And, $\overrightarrow{V_i^G} = \{v_{i,1}^G, v_{i,2}^G, \dots, v_{i,D}^G\}$ denotes a mutant vector. Mutant vectors are used to generate trial vectors with target vectors.

The DE algorithm consists of mutation, crossover, and selection operators. The mutant operator generates a mutant vector based on three donor vectors selected randomly in the population. This is as follows.

$$\overrightarrow{V_i^G} = \overrightarrow{X_{r_1}^G} + F \cdot \left(\overrightarrow{X_{r_2}^G} - \overrightarrow{X_{r_3}^G}\right) \quad (1)$$

where $r_1, r_2, r_3 \in \{1,2,\dots,NP\}$, $r_1 \neq r_2 \neq r_3 \neq i$. F is scaling factor, which is one of the control parameters of DE algorithm. In the crossover operator, a trial vector is generated based on a target vector and a mutant vector. This is as follows.

$$u_{i,j}^G = \begin{cases} v_{i,j}^G, & if\ rand_j[0,1) \leq CR\ or\ j \equiv jrand_i \\ x_{i,j}^G, & otherwise \end{cases} \quad (2)$$

where $jrand_i \in \{1,2,\dots,D\}$. $jrand_i$ ensures that at least one of the components of the trial vector is composed of the components of the mutant vector. CR is crossover rate, which is one of the control parameters of DE algorithm. Selection operator selectes the individuals to be populated in the $G+1$ generation population This is as follows.

$$\overrightarrow{X_i^{G+1}} = \begin{cases} \overrightarrow{U_i^G}, & if\ f\left(\overrightarrow{U_i^G}\right) \le f\left(\overrightarrow{X_i^G}\right) \\ \overrightarrow{X_i^G}, & otherwise \end{cases} \quad (3)$$

where f is an objective function. The DE algorithm executes these operators until the termination condition is satisfied.

3. Improved Differential Evolution

In DE algorithm, the more likely an individual has the appropriate control parameters, the more likely the individual is to evolve successfully. In other words, successfully evolved individuals are likely to have the appropriate control parameters. Using this observation, we propose a new adaptive parameter control method. The proposed method monitors the selection operator that which individuals succeeded in evolution and which individuals failed. The successfully evolved individuals tunes their control parameters using the Gaussian distribution, a short tailed distribution, and the failed individuals tunes their control parameters using the Cauchy distribution, a long tailed distribution. Through this process, the following effects can be obtained. First, even if an individual has been successfully evolved, it searches for a new value in the neighbor of its control parameters and applies it. This helps to find a new value that are better than the current control parameters. Next, an individual that has failed to evolve uses a large step to find and apply a new value that is far from its control parameters. This helps individuals which fail to evolve to find a new control parameter. The adaptive parameter control method of the proposed algorithm is as follows.

$$\overrightarrow{F_i^{G+1}} = \begin{cases} rand_{G_i}\left(\overrightarrow{F_i^G}, 0.1\right), & if\ \overrightarrow{X_i^{G+1}} \neq \overrightarrow{X_i^G} \\ rand_{C_i}\left(\overrightarrow{F_i^G}, 0.1\right), & otherwise \end{cases} \quad (4)$$

$$\overrightarrow{CR_i^{G+1}} = \begin{cases} rand_{G_i}\left(\overrightarrow{CR_i^G}, 0.1\right), & if\ \overrightarrow{X_i^{G+1}} \neq \overrightarrow{X_i^G} \\ rand_{C_i}\left(\overrightarrow{CR_i^G}, 0.1\right), & otherwise \end{cases} \quad (5)$$

where $rand_G(\mu, \sigma^2)$ and $rand_C(x_0, \gamma)$ denote the Gaussian distribution and the Cauchy distribution, respectively. Then, truncate the control parameters as follows. If $\overrightarrow{F_i^{G+1}}$ is lower than 0.1, $\overrightarrow{F_i^{G+1}} = 0.1$, else if $\overrightarrow{F_i^{G+1}}$ is bigger than 1.0, $\overrightarrow{F_i^{G+1}} = 1.0$. If $\overrightarrow{CR_i^{G+1}}$ is lower than 0, $\overrightarrow{CR_i^{G+1}} = 0$, if $\overrightarrow{CR_i^{G+1}}$ is bigger than 1.0, $\overrightarrow{CR_i^{G+1}} = 1.0$. Due to page limitation, we omit the analysis of the proposed algorithm.

4. Experiment and Result

4.1. CEC2014 Benchmark Problems

We performed the performance evaluation of the proposed algorithm in the CEC 2014 benchmark problems [10] with the conventional DE algorithm. Here, the control parameters of the conventional DE algorithm are: $F = 0.5$, $CR = 0.5$. A total of 50 performance evaluations were performed independently. We used the following settings.

- Population Size: $NP = 100$
- Function Evaluation Counter: $10000 \times D \times NP = 3.0\text{E}7$

Table 1 shows the experimental results. The algorithms that perform better in this table are shown in bold. As we can see in this table, we can confirm that the proposed algorithm has better optimization performance than the conventional DE algorithm. The proposed algorithm outperformed or exceeded the conventional DE algorithm with the exception of 5 problems for a total of 30 problems. Therefore, it is experimentally confirmed that the adaptive parameter control of the proposed algorithm can improve the performance of the DE algorithm.

Table 1. The result of performance evaluation in CEC2014 benchmark problems, $D = 30$

	Func.	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
IDE	Avg.	**2.21E+05**	**0.00E+00**	**0.00E+00**	**4.48E+01**	**2.04E+01**	**5.44E+02**	1.28E-03	**0.00E+00**	**3.70E+01**	**8.96E-01**
	Std. Dev.	**1.69E+05**	**0.00E+00**	**0.00E+00**	**3.64E+01**	**5.66E-02**	**2.13E+00**	3.28E-03	**0.00E+00**	**1.56E+01**	**1.20E+00**
DE	Avg.	9.42E+07	1.58E-09	6.71E-05	7.07E+01	2.09E+01	5.71E+02	**3.19E-14**	9.27E+01	1.78E+02	2.74E+03
	Std. Dev.	1.91E+07	6.80E-10	2.64E-05	5.06E-01	5.12E-02	1.60E+00	**5.12E-14**	7.85E+00	9.26E+00	2.00E+02
	Func.	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
IDE	Avg.	**3.66E+03**	**6.58E-01**	**2.77E-01**	3.13E-01	**6.89E+00**	**1.09E+01**	**3.16E+03**	**6.85E+01**	**4.71E+00**	**2.48E+01**
	Std. Dev.	**2.64E+02**	**8.91E-02**	**4.93E-02**	7.84E-02	**1.12E+00**	**3.00E-01**	**3.08E+03**	**2.49E+02**	**1.05E+00**	**3.70E+01**
DE	Avg.	6.60E+03	2.06E+00	3.92E-01	**2.81E-01**	1.62E+01	1.27E+01	2.30E+06	5.22E+03	7.85E+00	1.64E+02
	Std. Dev.	1.91E+02	2.43E-01	4.52E-02	**4.75E-02**	9.51E-01	1.43E-01	7.68E+05	3.04E+03	1.09E+00	1.95E+01
	Func.	F_{21}	F_{22}	F_{23}	F_{24}	F_{25}	F_{26}	F_{27}	F_{28}	F_{29}	F_{30}
IDE	Avg.	**4.35E+02**	**8.25E+01**	**3.15E+02**	2.28E+02	**2.04E+02**	**1.00E+02**	4.03E+02	8.49E+02	**9.41E+02**	**2.48E+03**
	Std.Dev.	**5.11E+02**	**6.09E+01**	**0.00E+00**	4.56E+00	**8.62E-01**	**0.00E+00**	3.55E+01	3.35E+01	**1.36E+02**	**1.08E+03**
DE	Avg.	1.14E+05	1.72E+02	**3.15E+02**	**2.23E+02**	2.22E+02	**1.00E+02**	**3.35E+02**	**8.29E+02**	2.85E+03	2.86E+03
	Std.Dev.	3.95E+04	5.85E+01	**0.00E+00**	**1.24E+00**	3.52E+00	**0.00E+00**	**3.65E+01**	**2.41E+01**	5.36E+02	6.33E+02

4.2. Training Multilayer Perceptron

We performed the performance evaluation of the proposed algorithm in the training of the multilayer perceptron with the conventional DE algorithm. The dataset we used is CANCER [11]. The CANCER dataset is divided into a total of 30 attributes and two classes. There are 569 instances in CANCER, of which 75% is used as training data and the remaining 25% is used as test data. Here, the control parameters of the conventional DE algorithm are: $F = 0.5$, $CR = 0.5$. A total of 10 performance evaluations were performed independently. We used the following settings.

- Network Structure: [30-20-10-2], [30-10-2]
- Population Size: $NP = 10$ for [30-20-10-2], $NP = 100$ for [30-10-2]
- Function Evaluation Counter: 500 for [30-20-10-2], 1000 for [30-10-2]

In the network structure of [30-20-10-2], each individual has a total of 620 + 210 + 22 = 852 chromosomes including bias. In the network structure of [30-10-2], each individual has a total of 310 + 22 = 332 chromosomes including bias. Therefore, the training multilayer perceptron problem is a large scale optimization problem. Table 2 shows the experimental results. The algorithms that perform better in this table are shown in bold. As we can see in this table, we can confirm that the proposed algorithm has superior classification performance compared to the conventional DE algorithm.

Table 2. The result of performance evaluation in the training of the multilayer perceptron

	Algorithm	Network Structure	Population Size	Result	
				Avg.	Std.Dev.
CANCER	IDE	30-20-10-2	10	**3.16.E-01**	**4.61.E-04**
	DE	30-20-10-2	10	5.00.E-01	1.41.E-01
CANCER	IDE	30-10-2	100	**4.08.E-01**	**1.85.E-05**
	DE	30-10-2	100	4.44.E-01	1.29.E-01

5. Conclusion

In this paper, we propose an improved DE algorithm and conduct the performance evaluation through CEC2014 benchmark problems. As a result of the performance evaluation, we confirmed that the overall optimization performance was improved compared with the conventional DE algorithm. We applied the proposed algorithm to the optimization technique of the training multilayer perceptron. In particular, we used a multilayer perceptron with a size of weights approaching 1,000. The performance evaluation shows that the classification performance is improved as compared with the conventional DE algorithm. As a result, we propose an improved DE algorithm that is useful for large scale optimization problems. We extend the proposed algorithm to develop algorithms that can be used to the optimization technique of training Deep learning as a future work.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2015R1D1A1A02062017). Correspondence should be addressed to Dr. Chang Wook Ahn.

References

[1] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

[2] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. No. ICS-8506. CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, 1985.

[3] Ilonen, Jarmo, Joni-Kristian Kamarainen, and Jouni Lampinen. "Differential evolution training algorithm for feed-forward neural networks." Neural Processing Letters 17.1 (2003): 93-105.

[4] Du, Ji-Xiang, et al. "Shape recognition based on neural networks trained by differential evolution algorithm." Neurocomputing 70.4 (2007): 896-903.

[5] Slowik, Adam, and Michal Bialko. "Training of artificial neural networks using differential evolution algorithm." 2008 Conference on Human System Interactions. IEEE, 2008.

[6] Slowik, Adam. "Application of an adaptive differential evolution algorithm with multiple trial vectors to artificial neural network training." IEEE Transactions on Industrial Electronics 58.8 (2011): 3160-3167.

[7] Storn, Rainer, and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." Journal of global optimization 11.4 (1997): 341-359.

[8] Storn, Rainer, and Kenneth Price. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Vol. 3. Berkeley: ICSI, 1995.

[9] Price, Kenneth, Rainer M. Storn, and Jouni A. Lampinen. Differential evolution: a practical approach to global optimization. Springer Science & Business Media, 2006.

[10] Liang, J. J., B. Y. Qu, and P. N. Suganthan. "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization." Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2013).

[11] Street, W. Nick, William H. Wolberg, and Olvi L. Mangasarian. "Nuclear feature extraction for breast tumor diagnosis." IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology. International Society for Optics and Photonics, 1993.