Learning Algorithm with Nonlinear Map Optimization for Neural Network

Yuki Hoshino¹ and Kenya Jin’no²

¹ Graduate School of Nippon Institute of Technology
4-1 Gakuendai, Miyashiro, Minami-SAITAMA,
Saitama 345 8501, Japan
E-mail: 2188012@sstu.nit.ac.jp

² Tokyo City University
1-28-1 Tamazutsumi, Setagaya-ku,
Tokyo 158 8557, Japan
E-mail: kjinno@tcu.ac.jp

Abstract

Recently, machine learning has been attracting attention. Machine learning is mainly realized by the learning of artificial neural networks. Various learning methods have been proposed; however, the learning methods are based on gradient methods. On the other hand, swarm intelligence (SI) algorithms have been attracting attention in the optimization field. Generally speaking, SI algorithms have a large computation cost. Therefore, there are few cases of SI algorithms being applied to machine learning. In this paper, we propose a novel learning algorithm for an artificial neural network which applies our proposed nonlinear map optimization (NMO) method. NMO consists of some simple particles which are driven by a simple nonlinear map. NMO can be classified as an SI algorithm. However, it has only a small computation cost. Therefore, NMO can be applied to a learning algorithm for an artificial neural network. In this paper, we introduce NMO, and a small learning simulation is carried out to confirm the performance of our learning method.

1. Introduction

Machine learning is currently applied to various fields and is attracting great attention[1][2]. The basis of machine learning is the learning of layer-type artificial neural networks (LNNs)[2][3]. A method based on the gradient method is mainly used for such learning. On the other hand, swarm intelligence (SI) algorithms[4][5] have been attracting attention in the optimization field. SI algorithms can search for an optimal value of a black-box problem without gradient information of the evaluation function. SI has the ability to search for the optimal value of an evaluation function without being trapped at a local minimum[5]. If we can utilize this capability of SI, we can construct an efficient learning algorithm. However, SI has a large computation cost because it consists of multiple computational units. Therefore, there are few cases of SI algorithms being applied to machine learning[6]. In such a situation, we recently proposed a novel SI algorithm named nonlinear map optimization (NMO)[7][8]. NMO consists of some simple particles which are driven by a simple nonlinear map. NMO can explore solutions with few particles; therefore, its computation cost is smaller than that of conventional SI algorithms[8]. For this reason, we consider using NMO for the learning of artificial neural networks. In this study, we apply NMO to machine learning, and we examine its learning performance. We determine a suitable number of neurons in the middle layer of an LNN.

2. Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) is one of the most popular LNN structures. It is usually interconnected in a feed forward manner[3]. Figure 1 shows an example MLP, which consists of an input layer, an intermediate layer, and an output layer. The operation of the MLP is described by Eq. (1).

![Figure 1: Three-layer MLP](image)

\[
\begin{align*}
x &= f(input) \\
y &= f(W^1x) \\
z &= f(W^2y)
\end{align*}
\] (1)

Here, input denotes an input vector. x, y, and z are the output vectors of the input layer, intermediate layer, and output layer, respectively. W¹ and W² denote the connection
coefficient matrices between the input layer and the intermediate layer, and between the intermediate layer and the output layer, respectively.

An MLP can represent various input/output relationships by appropriately setting the values of the connection coefficients between these neurons and the bias value of each neuron. In order to realize an arbitrary input/output relationship with an MLP, it is necessary to set appropriate connection coefficient matrices. For this purpose, learning involves setting connection coefficient matrices that minimize the evaluation function value using the error between the desired output vector and the actual output vector as shown in Eq. (2)[9].

$$E(W^1, W^2) = \sum_{p=1}^{P}(d^p - z^p)^t(d^p - z^p)$$  \hspace{1cm} (2)

Here $d^p$ is the $p$th desired output vector for the $p$th input vector, and $z^p$ is the output vector for the $p$th input vector. $P$ denotes the number of desired learning combinations. $t$ represents transpose.

In order to minimize the evaluation value, it is necessary to learn the input/output relationship based on previously given data[1][2][9]. Various learning methods have been proposed to effectively learn the connection weights of an MLP[1]. A method based on the gradient method is mainly used for such learning. Backpropagation (BP)[9] is one of the most well known learning algorithms for an MLP based on the gradient method.

The learning performance of BP is excellent. For this reason, deep learning, which has attracted attention in recent years, is also fundamentally based on a BP algorithm[1][2]. However, since BP is based on a gradient method, there is a possibility that parameters will be trapped at local minima during learning[9]. In general, an MLP has a large number of parameters, that is, connection weights, so there are many local minima in the evaluation function[1]. If we can establish a learning method which is not trapped by local minima, the MLP may be able to exhibit better performance.

3. Nonlinear Map Optimization

The learning algorithm of an MLP can be regarded as a nonlinear optimization algorithm. Various nonlinear optimization algorithms have been proposed. SI algorithms[4][5] are one such type of nonlinear optimization algorithms and include the ant colony algorithm[10], artificial bee algorithm[11], firefly algorithm[12], cuckoo search[13], and particle swarm optimization[14][15]. Such an SI algorithm can find a feasible solution in a short time[5]. The reason why an SI algorithm has a large computation cost is because of the need to search for a solution with many search individuals. Generally, a number of individuals greater than or equal to the number of search dimensions of the evaluation function is required in an SI algorithm. In such a situation, we propose a new SI algorithm that can satisfactorily search for the optimal solution of an evaluation function even if the number of individuals is small. The search individuals of the new SI algorithm are driven by a simple nonlinear map[7][8] which is a kind of circle map. The system is named, nonlinear map optimization (NMO).

NMO is described by the following equation[7].

$$\theta_{ij}^{t+1} = \begin{cases} \theta_{ij}^t + \gamma_{ij} \left[ x_{ij}^t - p_{ij}^t \right] + \frac{\pi}{2} - \cos^{-1}(\epsilon_c) \\ + \frac{\pi}{2} - \cos^{-1}(\epsilon_c), \text{if } 0 < \sin \theta_{ij}^t \cos \theta_{ij}^t < \epsilon_c \\ \theta_{ij}^t + \gamma_{ij} \frac{|x_{ij}^t - p_{ij}^t|}{R_{ij}(t)}, \text{otherwise} \end{cases}$$  \hspace{1cm} (3)

Here $x_{ij}$ and $\theta_{ij}$ denote the $j$th-dimensional location of the $i$th search individual in the $t$th iteration and the $j$th-dimensional rotation angle of the $i$th search individual in the $t$th iteration, respectively. $R_{ij}(t)$ represents the search range. $p_{ij}^t$ denotes the $j$th-dimensional coordinate of the $i$th search individual at the location giving the smallest objective function value up to the $t$th iteration. $\gamma_{ij}$ denotes a parameter which controls the response of the nonlinear map.

In order to search for a suitable connection coefficient matrix, each dimension of each search individual is made to correspond to a connection coefficient element. The search point of the search individual is updated by Eq. (3). The search performance of NMO depends on the parameters, namely, $\gamma$, $\epsilon_c$, and $R_{ij}^t$. Based on our preliminary experiments, in our simulations, we set the parameters as follows.

$$\begin{align*} \gamma &= 0.795 + \frac{(i+j) \mod D}{D} + 0.01 \\ \epsilon_c &= 0.01 \\ R_{ij}^t &= 5 \text{ (constant)} \end{align*}$$  \hspace{1cm} (4)

These parameter values are empirical values. By using the NMO, we search for an optimal value of the evaluation function as shown in Eq. (2).

4. Numerical Simulations

We carry out some numerical simulations to clarify the learning ability of an MLP by the NMO algorithm. In order to confirm the ability of the MLP learning method with a small problem, we learn EXOR with a two-input one-output MLP. Equation (2) is used to calculate the error for each input vector. In each simulation, there are four kinds of input vector; therefore, the sum of each error is applied to the evaluation. The initial value of each element of the connection weight matrices $W^1$ and $W^2$ is set in the interval $[-1, +1]$. In our simulations, the hyperbolic tangent function is applied to the function $f(\cdot)$ in Eq. (1). The input of each hyperbolic
tangent function is normalized by the number of input signals to each neuron.

First, in order to determine the influence of the number of search individuals on the search, we carry out some simulations in which only the number of searched individuals is changed under the same conditions. Figure 2 shows typical results of one trial. The horizontal axis denotes the iteration and the vertical axis denotes the error, which corresponds to the evaluation value. The number of neurons in the intermedi-

ute layer for the case in Fig. 2 is five. The blue curve denotes the case of three search individuals, the red curve denotes the case of six search individuals, and the green curve denotes the case of nine search individuals.

In these simulations, all trials converged to a suitable input/output relation. Figure 2 indicates that the convergence speed is faster when the number of individuals is nine. However, the amount of computation is proportional to the number of individuals. That is, when the number of individuals doubles, the amount of calculation also doubles. Comparing the cases of three and nine individuals, although the convergence speed for nine individuals is fast, it does not become fast in proportion to the number of individuals. We confirmed that NMO can search for a suitable solution very quickly even if it has only three search individuals.

Next, we consider the case where the number of neurons in the intermediate layer is fixed. In this simulation, the numbers of neurons in the input and output layers are fixed. Namely, the number of neurons in the input layer is two and the number of neurons in the output layer is one. Assuming that the number of neurons in the intermediate layer is \( M \), the number of connection coefficients is \( 4M + 1 \) when including the bias for each neuron. Therefore, the dimension of the evaluation function is increased when the number of neurons in the intermediate layer is increased. For the case that the number of intermediate-layer neurons is increased, we determine whether NMO can search for a suitable solution. Figures 3 and 4 show typical simulation results of one trial when the number of neurons in the intermediate layer is changed for the cases of three and nine individuals, respectively. The horizontal axis denotes the iteration and the vertical axis denotes the error, which corresponds to the evaluation value. In Figs.

Figure 2: Learning result based on the number of search individuals

Figure 3: Learning result for each number of neurons in the intermediate layer (three search individuals)

Figure 4: Learning result for each number of neurons in the intermediate layer (nine search individuals)

3 and 4, the blue curve denotes the case of five neurons in the intermediate layer, the red curve denotes the case of 10 neurons in the intermediate layer, and the green curve denotes the case of 15 neurons in the intermediate layer. As the number of neurons in the intermediate layer increases, the number of search dimensions increases. The results shown in Figs. 3 and 4 indicate that NMO can search for a suitable solution quickly even if the number of neurons in the intermediate layer is increased.

Finally, we carry out a numerical simulation to compare learning by NMO and learning by BP, which is one of the most popular learning algorithms for an MLP. Figure 5 illustrates the learning results. The horizontal logarithmic axis denotes the iteration and the vertical axis denotes the error, which corresponds to the evaluation value. In Fig. 5, the red curve represents the learning error of the BP method, and the other three curves denote the learning error of the NMO method for the cases of three, six, and nine search individuals. One hundred trials were done. Each result denotes the average of the 100 trials. We consider that the error converged to 0
when it became 0.001 or less. These results show the average results for the trials in which the learning is successful.
Note that the horizontal axis is logarithmic. Therefore, the convergence speed using NMO is extremely fast.
Therefore, we can conclude that the learning by NMO is considerably better than the learning by BP.

Figure 5: Comparison of learning by NMO and learning by BP

5. Conclusions
In this paper, we proposed a learning algorithm for MLP based on the NMO method. This method consists of some simple particles which are driven by a simple nonlinear map. NMO requires only a few search individuals for the search and has only a small computation cost. Therefore, it can be applied to a learning algorithm for an artificial neural network. Based on some numerical simulations of the learning input/output relation of EXOR, we confirmed the performance of learning by NMO. As a result, we clarified that the NMO method can learn a relationship more quickly than by BP.

Acknowledgements
This work was supported by JSPS KAKENHI Grant-in-Aid for Challenging Exploratory Research Number 16K14271 and Grant-in-Aid for Scientific Research(C) Number: 15K06077.

References
[1] K. Saito: Deep Learning to Make from Scratch, O’Reilly Japan, Ohm-sha, 2016 (in Japanese).
[2] T. Odaka: Machine Learning and Deep Learning, Ohm-sha, 2016 (in Japanese).
[3] M. L. Minsky and S. A. Papert: Perceptrons, MIT Press, 1969.
[4] G. Beni and J. Wang: Swarm intelligence in cellular robotic systems, Proc. NATO Advanced Workshop on Robots and Biological Systems, June 1989.
[5] E. Bonabeau and M. Dorigo and G. Theraulaz: Swarm Intelligence, Oxford University Press, 1999.
[6] A. Chakraborty and A. K. Kar: Swarm intelligence: A review of algorithms, Nature-Inspired Computing and Optimization, pp. 475-494, 2017.
[7] K. Jin’no: Nonlinear map model optimization, IEICE Tech. Rep. NLP2017-95, January 2018.
[8] S. Ishikawa and K. Jin’no: On a nonlinear map optimization, Proc. NCSP2018, pp. 379-382, March 2018.
[9] D. E. Rumelhart and G. E. Hinton and R. J. Williams: Learning representations by back-propagating errors, Nature 323 (6088), pp. 533-536, 1986.
[10] M. Dorigo and T. Stützle: Ant Colony Optimization, MIT Press, 2004.
[11] D. Karaboga: An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[12] X. S. Yang: Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2008.
[13] X. S. Yang and S. Deb: Cuckoo search via lévy flights, Proc. World Congress on Nature & Biologically Inspired Computing, Dec. 2009.
[14] J. Kennedy and R. Eberhart: Particle swarm optimization, Proc. IEEE 1995 International Conference on Neural Networks, pp. 1942-1948, Nov. 1995.
[15] M. Clerc: Particle Swarm Optimization, Wiley-ISTE, 2006.