electronics

MDPI

Article

Validation of Large-Scale Classification Problem in Dendritic Neuron Model Using Particle Antagonism Mechanism

Dongbao Jia [1], Yuka Fujishita [2], Cunhua Li [1], Yuki Todo [3] and Hongwei Dai [1,*]

[1] School of Computer Engineering, Jiangsu Ocean University, Lianyungang 222005, China; dbjia@jou.edu.cn (D.J.); 1987000045@jou.edu.cn (C.L.)
[2] Department of Management, eSOL Co., Ltd., Tokyo 164-8721, Japan; morimiworld-in-4.5jyou@outlook.com
[3] Faculty of Electrical, Information and Communication Engineering, Kanazawa University, Kanazawa-shi 920-1192, Japan; yktodo@ec.t.kanazawa-u.ac.jp
* Correspondence: hwdai@jou.edu.cn

Received: 15 April 2020; Accepted: 1 May 2020; Published: 11 May 2020

Abstract: With the characteristics of simple structure and low cost, the dendritic neuron model (DNM) is used as a neuron model to solve complex problems such as nonlinear problems for achieving high-precision models. Although the DNM obtains higher accuracy and effectiveness than the middle layer of the multilayer perceptron in small-scale classification problems, there are no examples that apply it to large-scale classification problems. To achieve better performance for solving practical problems, an approximate Newton-type method-neural network with random weights for the comparison; and three learning algorithms including back-propagation (BP), biogeography-based optimization (BBO), and a competitive swarm optimizer (CSO) are used in the DNM in this experiment. Moreover, three classification problems are solved by using the above learning algorithms to verify their precision and effectiveness in large-scale classification problems. As a consequence, in the case of execution time, DNM + BP is the optimum; DNM + CSO is the best in terms of both accuracy stability and execution time; and considering the stability of comprehensive performance and the convergence rate, DNM + BBO is a wise choice.

Keywords: neuron model; large-scale classification problem; dendritic neuron model (DNM); learning algorithm; neural network

1. Introduction

Along with the arrival of the era of big data, the third wave of artificial intelligence (AI) has been entered [1]. AI is being advanced for uniting IoT and robotics, not just as a research craze but through technological advances in hardware.

Commercialized AI is used as a system of prediction or classification that has been confirmed with high precision [2]. In general, it is difficult and indispensable to classify high quality data from vast amounts of data that contain all the information. However, if the problem is solved using AI, the cost can be greatly reduced. Moreover, the classification problems also exist in fields other than AI [3–5]. For example, in the field of gamma ray research in astronomy, cosmic rays can be observed through a telescope [6]. Due to the complexity of radiation, measurements are currently made using Monte-Carlo simulation. The ability to classify and detect gamma rays based on a limited set of characteristics would contribute to further improvements in telescopy.

The neural network is a symbolic presence that can deal with nonlinear problems in the third wave of AI [7,8]. In particular, with the deep learning of the middle layer of the multilayer perceptron (MLP) [9,10], which uses nonlinear functions to network, the accuracy is improved with an increase in

the amount of data used for learning. However, as the deepening of middle layer and the number of processed data increase, the computing cost becomes huge. As a consequence, with the characteristics of simple structure and cost saving, research into the dendritic neuron model (DNM) [11–16] is being developed to achieve high-precision models. Unlike other neural networks, the DNM is a model of one neuron, and much research indicates that the DNM has better performance than the MLP in small-scale classification problems [17,18]. Additionally, it is proved that the model, with its excellent metaheuristics, can obtain better classification accuracy by the previous optimization experiment for weight and threshold for small-scale classification problems with the DNM.

In various neuronal models and neural networks, it is necessary to change the weight of bonding strength between neurons in order to reduce the error in the desired output. However, as the neural network with random weights (NNRW) cannot guarantee the convergence of errors to the desired output, it is not considered to be a practical learning algorithm [19,20].

Generally, the learning algorithm is used to solve difficult optimization problems since the weighted learning can be considered as an optimization problem. The most famous learning algorithm in local exploration is the back-propagation (BP) [21–23], which computes the gradient using the chain rule and has the advantage of low learning cost since the installation is simple. In addition, the method of multi-point exploration, which is used as a model to classify according to natural phenomena and laws, is documented [24]. For example, the gravitational search algorithm is a basic learning algorithm to simulate physical phenomena [25–29], biogeography-based optimization (BBO) [30] is generally used for simulating ecological concepts since the accuracy and stability are the most outstanding among the models using representative metaheuristics [31], and some basic learning algorithms can simulate the moving sample population of organisms such as particle swarm optimization (PSO) [32,33] and ant colony optimization. Moreover, as a variant of PSO, the competitive swarm optimizer (CSO) [34,35] is a simplified metaheuristics set that is suitable for both multi-point and local exploration. Compared to the systems hat only conduct multi-point exploration or local exploration, the trap of the local optimal solution and convergence rate can be balanced using the CSO.

Although the DNM has shown higher accuracy and effectiveness than the MLP in small-scale classification problems, there are no examples that apply it to large-scale classification problems [36]. In this paper, the most famous learning algorithm using a gradient descent method with low computing cost, BP; the high classification accuracy algorithm for small-scale classification problems, BBO; and the especially low cost CSO algorithm are highlighted. The DNM is applied to large-scale classification problems with the above three learning algorithms, respectively. Consequently, the learning algorithm with BP is named DNM + BP, and DNM + BBO and DNM + CSO are named in a similar way. As a comparison object, the approximate Newton-type method (ANE)-NNRW is selected since it was applied to the same classification problem in the previous study. The ANE-NNRW is an NNRW based on the forward propagation MLP that ensures the convergence of solutions by an approximate Newton-type method [37].

Therefore, this study verifies the effectiveness of the DNM for large-scale classification problems, very important information for studying the performance of DNM.

2. Model and Learning Algorithm

2.1. Dendritic Neuron Model

The DNM is a model that vests dendrite function to the existing single layer perceptron [38–40] and is composed of four layers. Inputs $x_1, x_2, \ldots, x_n$ in each dendrite are firstly transformed to their corresponding outputs according to four connection instances in the synaptic layer, which possesses a sigmoid function for received inputs. Secondly, all the outputs from the synaptic layer in each dendrite are multiplied as new outputs of the dendrite layer. Thirdly, all the outputs in the dendrite layer are summed to obtain an output of the membrane layer. Finally, this output of the membrane layer is regarded as the input of the soma layer, which utilizes another sigmoid function to calculate the

ultimate result of the DNM. The complete structure of the DNM is shown in Figure 1, and its details are described as follows.

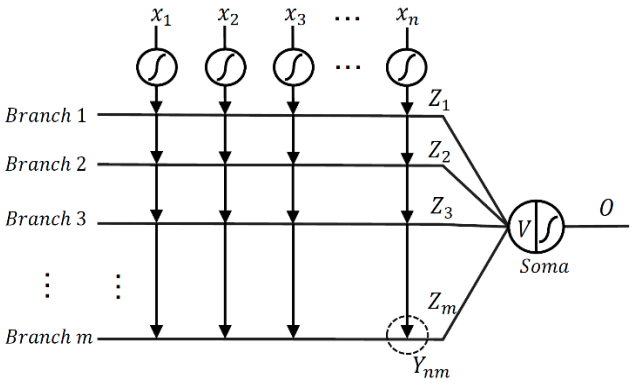

Figure 1. Structure of the dendritic neuron model (DNM).

2.1.1. Synaptic Layer

A synapse connects neurons from a dendrite to another dendrite/axon or the soma of another neural cell. The information flows from a presynaptic neuron to a postsynaptic neuron, which shows feedforward nature. The changes in the postsynaptic potential influenced by ionotropic phenomena determine the excitatory or inhibitory nature of a synapse. The description connecting the ith ($i = 1, 2, \ldots, n$) synaptic input to the jth ($j = 1, 2, \ldots, m$) synaptic layer is given as

$$Y_{ij} = \frac{1}{1 + e^{-k(w_{ij}x_i - \theta_{ij})}} \tag{1}$$

where Y_{ij} is the output from the ith synaptic input to the jth synaptic layer. k indicates a positive constant. x_i manifests the ith input of a synapse and $x_i \in [0, 1]$. Weight w_{ij} and threshold θ_{ij} are the connection parameters to be learned.

According to the values of w_{ij} and θ_{ij}, four types of connection instance are shown in Figure 2, where the horizontal axis indicates the inputs of presynaptic neurons and the vertical one clarifies the output of the synaptic layer. As the range of x is [0, 1], only the conforming part is required to be seen. The four connection instances contain:

Figure 2a,b presents $w_{ij} < 0 < \theta_{ij}$ or $0 < w_{ij} < \theta_{ij}$, where the output is approximately 0 no matter when the input transforms from 0 to 1;

Figure 2c,d presents $\theta_{ij} < 0 < w_{ij}$ or $\theta_{ij} < w_{ij} < 0$, where the output is approximately 1 no matter when the input transforms from 0 to 1;

Figure 2e depicts $0 < \theta_{ij} < w_{ij}$, where the output is proportional to the input no matter when the input transforms from 0 to 1;

Figure 2f depicts the inhibitory connection when $w_{ij} < \theta_{ij} < 0$, where the output is inversely proportional to the input no matter when the input transforms from 0 to 1. It is worth noting that these four connection instances are critical to infer the morphology of a neuron by specifying the positions and synapse types of dendrites.

2.1.2. Dendrite Layer

The dendrite layer shows a multiplicative function of the outputs from synapses at various synaptic layers [41]. A type of multiplicative operation can be achieved due to the nonlinearity of synapses, i.e., constant 0 or 1 connection. That is why a multiplicative operation has been chosen to use in this model when it comes to the dendrite layer. The multiplication is equivalent to the logic and

operation since the values of inputs and outputs of the dendrites correspond to 1 or 0. The output function for the *j*th dendrite branch is expressed as follows:

$$Z_j = \prod_{i=1}^{n} Y_{ij} \tag{2}$$

2.1.3. Membrane Layer

The membrane layer collects the signals from each dendritic branch. The input received from a dendrite branch is calculated with a summation function, which closely resembles a logic or operation. Then, the resultant output is delivered into the next layer to activate the soma body. The output of this layer is formulated as

$$V = \sum_{j=1}^{m} Z_j \tag{3}$$

2.1.4. Soma Layer

At last, the soma layer implements the function of soma body such that the neuron fires if the output from the membrane layer exceeds its threshold. This process is expressed by a sigmoid function used to calculate the ultimate output of the entire model:

$$O = \frac{1}{1 + e^{-k_s(V-\theta_s)}} \tag{4}$$

The parameter k_s is a positive constant, and the threshold θ_s varies from 0 to 1.

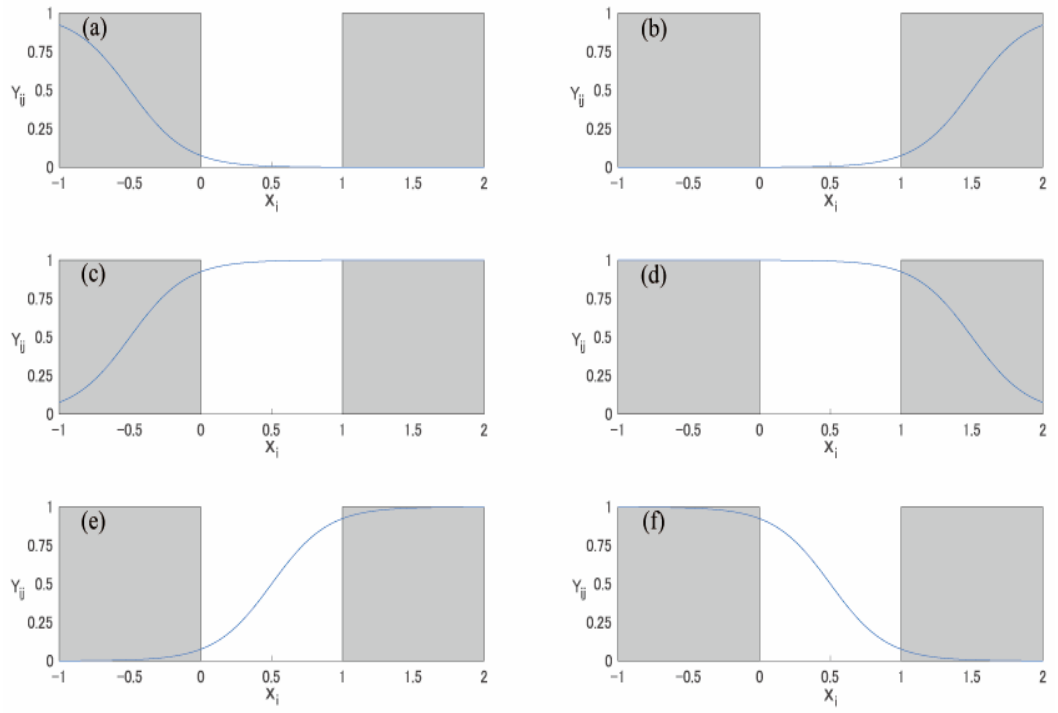

Figure 2. Four types of connection instance in the synaptic layer: (**a**) Constant 0 connection, (**b**) Constant 0 connection, (**c**) Constant 1 connection, (**d**) Constant 1 connection, (**e**) Excitatory connection, and (**f**) Inhibitory connection.

According to the multiplication of each dendrite, the DNM can be used as a neuron model and solve complex problems such as nonlinear problems. In addition, in terms of the activation function of synaptic layer and soma layer, the sigmoid function is used in reference to previous studies.

2.2. Back-Propagation

BP is a point gradient descent learning algorithm that uses chain law to calculate the gradient [42,43]. The construction of the neuron model depends on an effective learning rule. Its learning rule is obtained by the least squared error between the real output vector O and the target output vector T, shown as follows:

$$E = \frac{1}{2}(T - O)^2 \tag{5}$$

The error is decreased by correcting the synaptic parameters w_{ij} and θ_{ij} in the DNM of the connection function during learning. The updated equations are expressed as follows:

$$\begin{aligned} \Delta w_{ij}(t) &= \textstyle\sum_{p=1}^{P} \frac{\partial E_p}{\partial w_{ij}}, \\ \Delta \theta_{ij}(t) &= \sum_{p=1}^{P} \frac{\partial E_p}{\partial \theta_{ij}} \end{aligned} \tag{6}$$

where η represents the learning rate, which is a user-defined parameter, and E_p is the mean square error. Then, the updating rules of w_{ij} and θ_{ij} are computed as follows:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) - \eta \Delta w_{ij}(t), \\ \theta_{ij}(t+1) &= \theta_{ij}(t) - \eta \Delta \theta_{ij}(t) \end{aligned} \tag{7}$$

where t is the number of the learning iteration. In addition, the partial differentials of E_p with regard to w_{ij} and θ_{ij} are defined as follows:

$$\begin{aligned} \frac{\partial E_p}{\partial w_{ij}} &= \frac{\partial E_p}{\partial O_p} \frac{\partial O_p}{\partial V} \frac{\partial V}{\partial Z_j} \frac{\partial Z_j}{\partial Y_{ij}} \frac{\partial Y_{ij}}{\partial w_{ij}}, \\ \frac{\partial E_p}{\partial \theta_{ij}} &= \frac{\partial E_p}{\partial O_p} \frac{\partial O_p}{\partial V} \frac{\partial V}{\partial Z_j} \frac{\partial Z_j}{\partial Y_{ij}} \frac{\partial Y_{ij}}{\partial \theta_{ij}} \end{aligned} \tag{8}$$

The detail parts of the above partial differentials are represented as follows:

$$\begin{aligned} \frac{\partial E_p}{\partial O_p} &= T_p - O_p \\ \frac{\partial O_p}{\partial V} &= \frac{k_s x_i e^{-k_s(w_{ij}x_i - \theta_s)}}{\left(1 + e^{-k_s(w_{ij}x_i - \theta_s)}\right)^2} \\ \frac{\partial V}{\partial Z_j} &= 1 \\ \frac{\partial Z_j}{\partial Y_{ij}} &= \prod_{l=1 and li}^{n} Y_{lj} \\ \frac{\partial Y_{ij}}{\partial w_{ij}} &= \frac{k x_i e^{-k(w_{ij}x_i - \theta_{ij})}}{\left(1 + e^{-k(w_{ij}x_i - \theta_{ij})}\right)^2} \\ \frac{\partial Y_{ij}}{\partial \theta_{ij}} &= \frac{k e^{-k(w_{ij}x_i - \theta_{ij})}}{\left(1 + e^{-k(w_{ij}x_i - \theta_{ij})}\right)^2} \end{aligned} \tag{9}$$

In the calculation of $\Delta w_{ij}(t)$ and $\Delta \theta_{ij}(t)$, the partial differential is obtained from input to output in order and in reverse order.

2.3. Biogeography-Based Optimization

BBO is a metaheuristics model of the speciation, extinction, and geographical distribution in biogeography, whose characteristic takes the habitat as a solution and shares the suitability index variables (SIVs) with other habitats directly [44]. The fitness values of other learning algorithms are expressed as the habitat suitability index (*HSI*), implemented as follows:

1. Current rank of habitat H_i ($i = 1, 2, \ldots, n$) produces the integer spectrum SIV.
2. The *HSI* of each habitat is calculated using the following equation:

$$HSI(H_i) = \frac{1}{2P}\sum_{p=1}^{P}(T_p - O_p)^2 \tag{10}$$

where P is the total number of training samples, T_p is the target vector of the pth sample, and O_p is the actual output vector obtained by H_i.

3. The SIV is randomly selected and immigration to other habitats occurs according to the calculations of the emigration rate μ_i and immigration rate λ_i.

$$\begin{array}{c} \mu_i = \frac{Ei}{m} \\ \lambda_i = I\left(1 - \frac{i}{m}\right) \end{array} \tag{11}$$

where E is the emigration rate, and I is the maximum immigration rate. The case that $E = I = 1$ is considered in BBO, and the relationship between λ and μ is established as the following formula:

$$\lambda_i + \mu_i = E \tag{12}$$

4. For each habitat H_i, the immigrated *HSI* and the probability, Ps_i, that it contains the Sth species of habitat are updated.

$$Ps_i(t + \Delta t) = Ps_i(t)(1 - \lambda_i\Delta t - \mu_i\Delta t) + Ps_{i-1}\lambda_{i-1}\Delta t + Ps_{i+1}\mu_{i+1}\Delta t \tag{13}$$

If t is sufficiently small, the following equation can be approximated:

$$Ps_i = \begin{cases} -(\lambda_i + \mu_i)Ps_i + \mu_{i+1}Ps_{i+1} & i = 0 \\ -(\lambda_i + \mu_i)Ps_i + \lambda_{i-1}Ps_{i-1} + \mu_{i+1}Ps_{i+1} & 1 \le i \le n-1 \\ -(\lambda_i + \mu_i)Ps_i + \lambda_{i-1}Ps_{i-1} & i = n \end{cases} \tag{14}$$

5. Species numbers are varied according to the mutation rate, Pm_i, for non-elite habitats:

$$Pm_i = Pm_{max}\frac{1 - Ps_i}{Ps_{max}} \tag{15}$$

where Ps_{max} is the maximum value of Ps_i, and Pm_{max} is the parameter.

6. Step 2 is returned to for the next iteration. The algorithm does not end until the termination condition is satisfied.

2.4. Competitive Swarm Optimizer

The CSO is a kind of group intelligence that improves PSO to face large-scale classification problems. It is a mechanism for comparing the evaluation results of different particles selected from the population; only the failed particles are learned to update [45]. Therefore, in addition to the number of updated particles being able to be reduced to $2/N$, the excellent solutions in the search do not need to

be saved, and it can be used for efficient search on large-scale classification problems. As with PSO, the individual movement with speed will not be eliminated. The operating steps are shown as follows:

1. For N initial solutions, the particle position $x_i(i = 1, 2, \ldots, N)$ and velocity $v_i(i = 1, 2, \ldots, N)$ of generated particles are calculated.
2. All solutions are evaluated.
3. The kth ($k = 1, 2, \ldots, 2/N$) competition for generation t occurs as follows:
   (a) The non-repeating particles N_{k1} and N_{k2} are randomly selected from the undecided particles.
   (b) The positions of selected particles of N_{k1} and N_{k2} are compared and evaluated to determine the winning particle and the failing particle.
   (c) A velocity $v_{l,k}$ is applied to the position $x_{l,k}$ of the failed particle to make it move.

$$\begin{gathered} v_{l,k}(t+1) = R_1(k,t)v_{l,k}(t) + R_2(k,t)\left[x_{w,k}(t) - x_{l,k}(t)\right] + \varphi R_3(k,t)\left[\overline{x_k}(t) - x_{l,k}(t)\right], \\ x_{l,k}(t+1) = x_{l,k}(t) + v_{l,k}(t+1) \end{gathered} \tag{16}$$

   where $R_1(k, t)$, $R_2(k, t)$, and $R_3(k, t)$ are the random vectors of [0, 1], $\overline{x_k}(t)$ is the average position of the whole particles, and ϕ is the control parameter that sets the degree of influence from the average position, which is recommended as the following conditions in the previous study:

$$\begin{cases} \varphi = 0 & N \le 100 \\ \varphi \in [0.14\log(N) - 0.3,\ 0.27\log(N) - 0.51] & otherwise \end{cases} \tag{17}$$

   (d) Operations (a) through (c) are repeated until all the particles are decided.
4. Step 2 is returned to for the next iteration. The algorithm does not end until the termination condition is satisfied.

3. Experiment

Three classification problems in our experiments are shown in Table 1 below. The most downloaded open data sets in different fields of the UCI Machine Learning Repository are used [46], and the value of the characteristic number does not contain the class. F1 classifies whether the cosmic ray received by the Cherenkov telescope is a gamma ray or not, F2 classifies whether the space shuttle radiator is abnormal, and F3 classifies whether the pixel is skin based on the RGB information of the image. Furthermore, the characteristics of any classification problem are expressed by numbers with no data error that contain negative numbers and decimal numbers. According to the input range of the synaptic layer, each characteristic data set is standardized for the experiment.

Table 1. Details of the classification data sets.

No.	Classification Data Set	Characteristic Number	Total Data
F1	Magic gamma telescope data set	10	19,020
F2	Stat log shuttle data set	9	58,000
F3	Skin segmentation data set	3	245,057

Each classification problem is tested 30 times independently, and the accuracy of the expected output is calculated according to the classification results. The formula of accuracy is shown in the Equation (18) with TP (true positivity), FP (false positive), TN (true negative), and FN (false negative).

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \tag{18}$$

Besides, the mean square error (MSE) is determined as the evaluation function of the solution that is obtained through Equation (5) for DNM + BP, and Equation (19) for DNM + BBO and DNM + CSO:

$$MSE = \frac{1}{2P}\sum_{p=1}^{P}\left(O_p - T_p\right)^2 \tag{19}$$

The termination condition is set to reach the maximum generation number; for DNM + BP, it is 1000; and for DNM+BBO and DNM + CSO, it is 200, according to the ANE-NNRW. The population number of BBO and CSO is 50.

In addition to the partition ratio of F2, which is specified by the data set, the learning data account for 70% and the testing data account for 30%, referring to the previous study of the DNM as shown in Table 2 [18], and the proportion in the ANE-NNRW is shown in Table 3. In order to make the dimensions consistent, the maximum value of m is set based on the maximum value of m of the interlayer in the previous study.

Table 2. Number and proportion of data sets in the DNM.

No.	Learning Data Number (Proportion)	Testing Data Number (Proportion)
F1	13,314 (70%)	5706 (30%)
F2	43,500 (75%)	14,500 (25%)
F3	171,540 (70%)	73,517 (30%)

Table 3. Numbers and proportions of data sets in the approximate Newton-type neural network with random weights (ANE-NNRW).

No.	Learning Data Number (Proportion)	Testing Data Number (Proportion)
F1	19,020 (100%)	1300 (6.83%)
F2	43,500 (75%)	14,500 (25%)
F3	200,000 (81.6%)	45,057 (18.4%)

Moreover, considering the processing load of the DNM, the upper limit of m is 100. Table 4 shows a list of m for the previous study and dimension D and m for this study. Due to the experimental load and time, F1, F2, and F3 use different environments as shown in Table 5.

Table 4. Set value of m in the experiment.

No.	Maximum Value m of ANE-NNRW	Maximum Value m of DNM	Maximum Dimension D of DNM
F1	1200	60	$2 \times 10 \times 60 = 1200$
F2	2000	100	$2 \times 9 \times 100 = 1800$
F3	1200	100	$2 \times 3 \times 100 = 600$

Table 5. Experimental environment of the DNM.

Item	Computing Environment of F1	Computing Environment of F2 and F3
CPU	3.00 GHz Intel(R) Core(TM) i5-8500	3.00 GHz Intel(R) Core(TM) i5-7400
OS	Windows 10 Education	Windows 10 Pro
RAM	16.0 GB	8.00 GB
Software	MATLAB R2018b	MATLAB R2018b

According to the design of experiment, which is a statistical method for the effective analysis of large combinations using orthogonal arrays based on Latin square, DNM + BP, DNM + BBO, and DNM + CSO are conducted under the above conditions, and the number of experiments can be greatly reduced by the relationship between the factors and levels.

Each factor and level will be applied in the orthogonal array of L_{25} (5^6) since this experiment has five factors and five levels. Tables 6 and 7 represent lists of factors and levels used in F1, F2, and F3.

Table 6. Factors and levels of F1.

m	k	k_s	θ_s	η
5	1	1	0.1	0.0001
15	5	5	0.3	0.0005
30	10	10	0.5	0.001
45	15	15	0.7	0.005
60	25	25	0.9	0.01

Table 7. Factors and levels for F2 and F3.

m	k	k_s	θ_s	η
15	1	1	0.1	0.0001
30	5	5	0.3	0.0005
50	10	10	0.5	0.001
75	15	15	0.7	0.005
100	25	25	0.9	0.01

Table 8 applies to the parameters of F1, and Table 9 applies to those of F2 and F3. Moreover, the numbers in Tables 8 and 9 are the combination numbers of the experimental parameters.

Table 8. Parameter combination of F1.

No.	m	k	k_s	θ_s	η
1	5	1	1	0.1	0.0001
2	5	5	5	0.3	0.0005
3	5	10	10	0.5	0.001
4	5	15	15	0.7	0.005
5	5	25	25	0.9	0.01
6	15	1	5	0.5	0.005
7	15	5	10	0.7	0.01
8	15	10	15	0.9	0.0001
9	15	15	25	0.1	0.0005
10	15	25	1	0.3	0.001
11	30	1	10	0.9	0.0005
12	30	5	15	0.1	0.001
13	30	10	25	0.3	0.005
14	30	15	1	0.5	0.01
15	30	25	5	0.7	0.0001
16	45	1	15	0.3	0.01
17	45	5	25	0.5	0.0001
18	45	10	1	0.7	0.0005
19	45	15	5	0.9	0.001
20	45	25	10	0.1	0.005
21	60	1	25	0.7	0.001
22	60	5	1	0.9	0.005
23	60	10	5	0.1	0.01
24	60	15	10	0.3	0.0001
25	60	25	15	0.5	0.0005

Table 9. Parameter combination of F2 and F3.

No.	m	k	k_s	θ_s	η
1	15	1	1	0.1	0.0001
2	15	5	5	0.3	0.0005
3	15	10	10	0.5	0.001
4	15	15	15	0.7	0.005
5	15	25	25	0.9	0.01
6	30	1	5	0.5	0.005
7	30	5	10	0.7	0.01
8	30	10	15	0.9	0.0001
9	30	15	25	0.1	0.0005
10	30	25	1	0.3	0.001
11	50	1	10	0.9	0.0005
12	50	5	15	0.1	0.001
13	50	10	25	0.3	0.005
14	50	15	1	0.5	0.01
15	50	25	5	0.7	0.0001
16	75	1	15	0.3	0.01
17	75	5	25	0.5	0.0001
18	75	10	1	0.7	0.0005
19	75	15	5	0.9	0.001
20	75	25	10	0.1	0.005
21	100	1	25	0.7	0.001
22	100	5	1	0.9	0.005
23	100	10	5	0.1	0.01
24	100	15	10	0.3	0.0001
25	100	25	15	0.5	0.0005

4. Results

By calculating the average accuracy, the optimum parameters used in this experiment are selected as shown in Tables 10–12, respectively.

Table 10. Optimum parameters of the DNM + back-propagation (BP).

No.	m	k	k_s	θ_s	η	Average Accuracy (%)
F1	60	5	1	0.9	0.005	82.40
F2	75	10	1	0.7	0.0005	90.96
F3	75	10	1	0.7	0.0005	83.76

Table 11. Optimum parameters of the DNM + biogeography-based optimization (BBO).

No.	m	k	k_s	θ_s	Average Accuracy (%)
F1	45	1	15	0.3	81.69
F2	100	15	10	0.3	92.40
F3	50	1	10	0.9	97.44

Table 12. Optimum parameters of DNM + competitive swarm optimizer (CSO).

No.	m	k	k_s	θ_s	ϕ	Average Accuracy (%)
F1	60	1	25	0.7	0.05	82.34
F2	75	10	1	0.7	0.05	96.76
F3	50	1	10	0.9	0	95.04

The average accuracy and standard deviation of each problem and learning algorithm are shown in Table 13. It is clearly that DNM + BP in F1, DNM + CSO in F2, and DNM + BBO in F3 have the highest accuracy.

Table 13. Average accuracy and standard deviation of each learning algorithm.

No.	Learning Algorithm	Learning's Average Accuracy (%) ± Standard Deviation	Test's Average Accuracy (%) ± Standard Deviation
F1	DNM + BP	81.91 ± 1.66	81.78 ± 1.60
	DNM + BBO	79.92 ± 1.62	79.65 ± 1.67
	DNM + CSO	80.09 ± 2.10	80.03 ± 2.13
F2	DNM + BP	90.70 ± 3.30	90.68 ± 3.32
	DNM + BBO	91.53 ± 2.93	91.49 ± 2.90
	DNM + CSO	94.53 ± 2.55	94.54 ± 2.58
F3	DNM + BP	80.53 ± 4.83	80.49 ± 4.84
	DNM + BBO	97.72 ± 0.72	97.70 ± 0.75
	DNM + CSO	95.95 ± 0.77	95.98 ± 0.77

The results of the ANE-NNRW as the comparison object are shown in Table 14, and the accuracy of the previous paper is recorded as a percentage [19]. Obviously, the accuracy of the DNM is higher than that of the ANE-NNRW in all problems. They also prove that the DNM has the advantage of high accuracy even if the data required for learning are not as much as for the ANE-NNRW, which corresponds to Tables 2 and 3.

Table 14. Highest accuracy of each problem with the ANE-NNRW.

No.	Learning's Average Accuracy (%)	Test's Average Accuracy (%)
F1	67.16	61.46
F2	79.24	79.16
F3	79.28	79.46

Table 15 summarizes the average execution time and optimum parameter m, which is in deep relation to dimension D. The execution time of the DNM is larger than that of the ANE-NNRW as shown in Table 15 [19].

Table 15. Average execution time and optimum parameter of each learning algorithm.

No.	Learning Algorithm	Average Execution Time (sec)	Optimum Parameter m
F1	ANE-NNRW	27.3	
	DNM + BP	1276.8	60
	DNM + BBO	1629.7	45
	DNM + CSO	2158.4	60
F2	ANE-NNRW	35.89	
	DNM + BP	5485.2	75
	DNM + BBO	12454.0	100
	DNM + CSO	9284.5	75
F3	ANE-NNRW	271.0	
	DNM + BP	6369.1	75
	DNM + BBO	8805.1	50
	DNM + CSO	8795.7	50

Even in DNM + BP, which has the lowest computational cost in the experimental method, the execution time of the 200th generation is at least four times that of the ANE-NNRW. One of the reasons for this is that the DNM is more time-consuming than the MLP, which is also mentioned in the case of small-scale classification problems. Additionally, the ANE-NNRW is a calculation method based on the pure propagation MLP; it is considered that a similar result should appear for this large-scale classification problem. Secondly, the ANE-NNRW is an efficient way to solve the large-scale

problem; the data set segmentation is performed. Besides, the computational cost is less by learning with random weighting to ensure convergence.

According to the average execution time of DNM+BBO and DNM + CSO in F2 and F3, as the optimum parameter m of DNM + BBO is higher than that of DNM + CSO, there is a large difference of more than 3000 seconds between the two methods in F2. Similarly, because the value of m is the same in F3, the difference is about 10 seconds since DNM + BBO has more order of solution updates. In addition, the difference between DNM + BBO and DNM + CSO in F1 can also be considered as the difference between m, but it is not as great as in F2. Therefore, DNM+CSO is better than DNM + BBO in terms of the execution time of the learning algorithm.

On the other hand, for DNM + BP and DNM + CSO, there is a difference of at least 1000 seconds in the execution time between F1 and F2 under the same m, which is caused by the difference in the number of data and features.

As the DNM calculates m and the number of features using Equation (1), it is confirmed that the execution time approximately follows the calculated amount for the model. Consequently, the execution time increases due to the increase in the number of data. In the same problem, it is desirable that a smaller value of m can reduce the time. However, the m of DNM + BP in F2 and F3 obviously shows that the difference in execution time cannot be determined by the number of data alone.

Therefore, for improving the precision, it is necessary to set an upper limit, split data, and process in parallel to reduce the load of a large-scale problem since the DNM varies in precision according to the set parameters.

Furthermore, the upper limit of m is set as 100 in this experiment, but both of the optimum parameters m of DNM + BBO and DNM + CSO in F3 are 50, half of the upper limit, with a high accuracy of classification as shown in Table 13. Therefore, for large-scale classification problems, DNM learning by metaheuristics may not require an extremely large number of m for problems with a small number of features. As a reference for parameter determination in the application of the DNM to large-scale problems, this will be clarified in the future.

The average convergence graph of each generation of MSE obtained by experiments are shown in Figures 3–5. Figure 3a presents the DNM + BBO and DNM + CSO convergence graphs of F1, and Figure 3b presents the DNM + BP convergence graph of F1. In a similar way, the convergence graphs of F2 and F3 are shown in Figures 4 and 5, respectively.

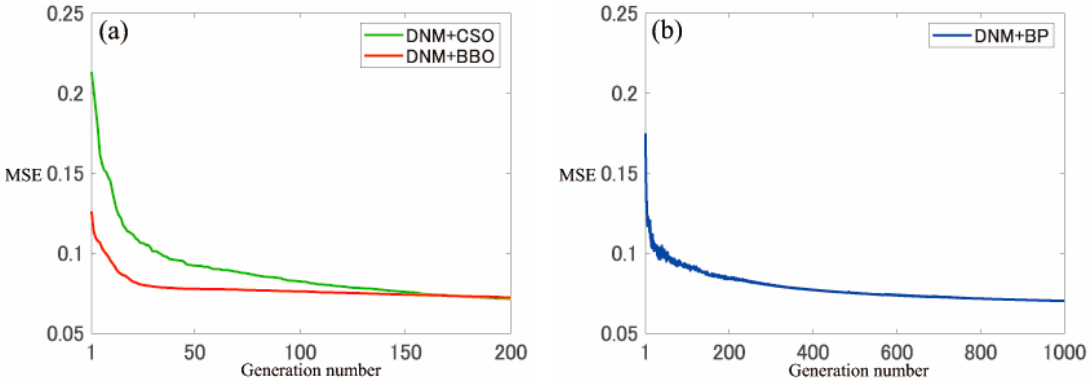

Figure 3. Convergence graphs of F1. (**a**) DNM + BBO and DNM + CSO, (**b**) DNM + BP.

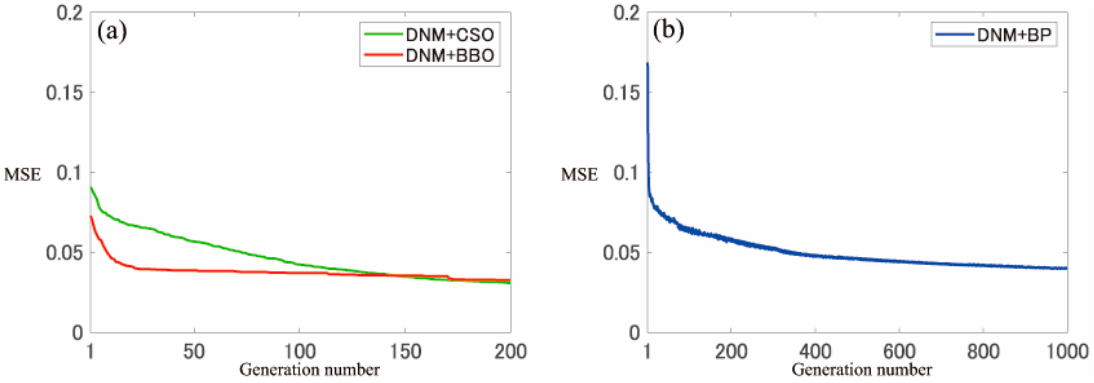

Figure 4. Convergence graphs of F2. (**a**) DNM + BBO and DNM + CSO, (**b**) DNM + BP.

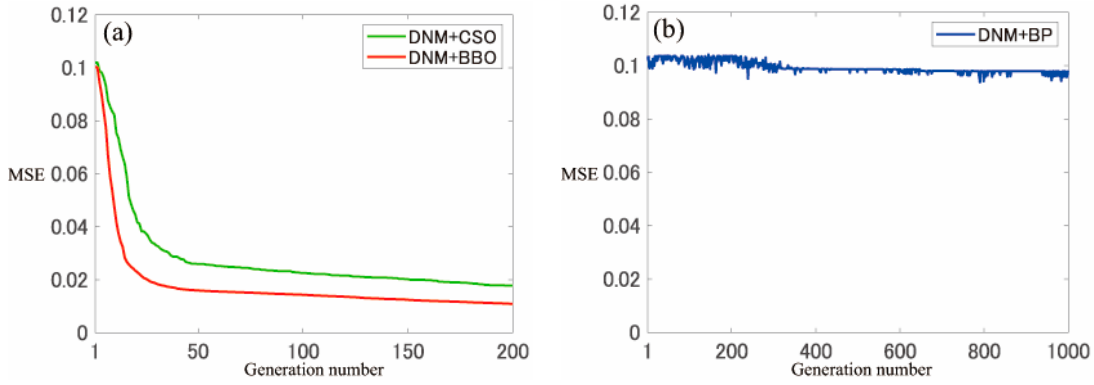

Figure 5. Convergence graphs of F3. (**a**) DNM + BBO and DNM + CSO, (**b**) DNM + BP.

It is clearly that the values of each learning algorithm converge to the end generation number and that DNM + BBO converged first in all cases. In order to reduce the computation, the CSO only updates the solution of particles equal to half of the population numbers. The BBO of the same multi-point search keeps the elite habitat and changes the solution in each generation, and the number of new candidate optimum solutions produced in one generation is larger than that with the CSO. As a new solution is derived from a candidate optimum solution at a certain point in time, the convergence rate of the higher quality solution will be accelerated. Therefore, it is considered that BBO is more suitable than the CSO to obtain a small MSE with a smaller number of generations.

Furthermore, the MSE does not change greatly due to the local solution of F3 in Figure 5b, indicating that BP with a feature that tends to trapped in the local solution has the shortcoming of the problem orientation not being significant compared with that with the multi-point search method, with which it is easy to escape from the local solution.

The stability of each method in F1, F2, and F3 is illustrated by the box-plots of Figures 6–8, respectively. In the case of the minimum MSE of each problem, F1 is DNM + CSO, and F2 and F3 are DNM + BBO. Besides, for the maximum MSE of each problem, F1 is DNM + BBO, and F2 and F3 are DNM + BP.

However, for the comprehensive consideration, it can be seen that DNM + BP in F1, DNM + CSO in F2, and DNM + BBO in F3 record the best stability. Moreover, the average values of MSE for the end conditions in each problem and method are shown in Table 16. It shows that the method with excellent stability in each problem is also superior to other methods in terms of the average MSE.

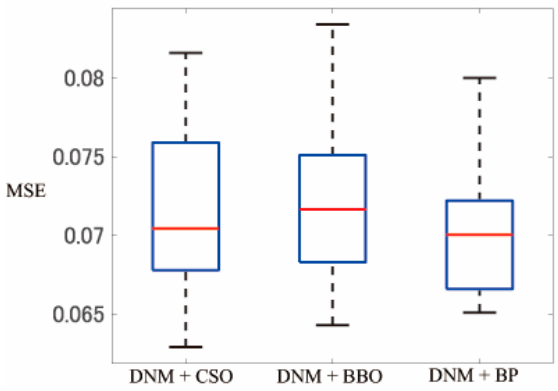

Figure 6. Mean square error (MSE) of each method in F1.

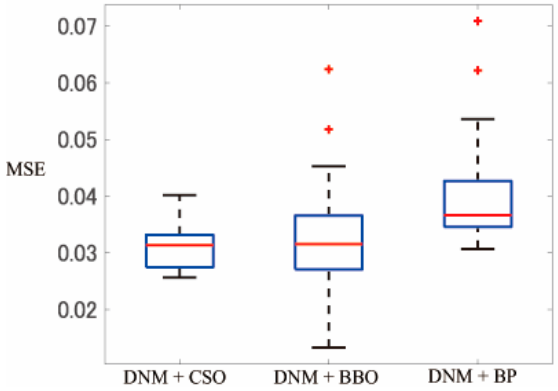

Figure 7. MSE of each method in F2.

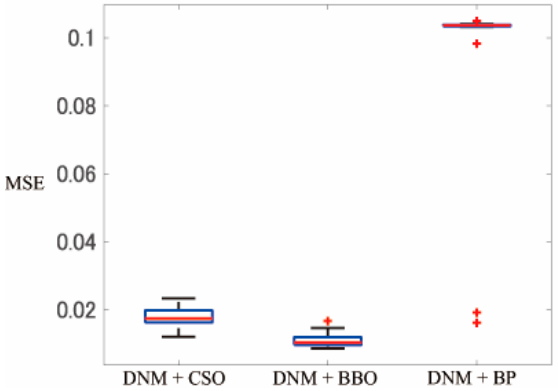

Figure 8. MSE of each method in F3.

Furthermore, the average of standard deviation of accuracy and the standard deviation of each method for the tests are shown in Table 17 below. Although DNM + BP has the best stability in F1, both the average value and standard deviation are the highest, which indicates that DNM + BP has a large deviation according to the different problems. To the contrary, DNM + BBO and DNM + CSO have more stability that is not easily affected by these three problems of this experiment.

Table 16. Average MSE at end condition.

No.	Learning Algorithm	Average MSE
F1	DNM + BP	7.04×10^{-2}
	DNM + BBO	7.24×10^{-2}
	DNM + CSO	7.15×10^{-2}
F2	DNM + BP	4.01×10^{-2}
	DNM + BBO	3.27×10^{-2}
	DNM + CSO	3.09×10^{-2}
F3	DNM + BP	9.78×10^{-2}
	DNM + BBO	1.09×10^{-2}
	DNM + CSO	1.78×10^{-2}

Table 17. Average of standard deviation of accuracy and standard deviation of each method for the tests.

Learning Algorithm	Average of Standard Deviation ± Standard Deviation
DNM + BP	3.25 ± 1.62
DNM + BBO	1.77 ± 1.08
DNM + CSO	1.83 ± 0.95

As a consequence, in terms of the convergence and stability of MSE, it is better to adopt a multi-point search method, especially DNM + BBO with the advantage in terms of the convergence rate.

Figures 9–11 depict the receiver operating characteristic (ROC) of each method in F1, F2 and F3, respectively. Furthermore, the average value of area under curve (AUC) in each problem and method is shown in Table 18.

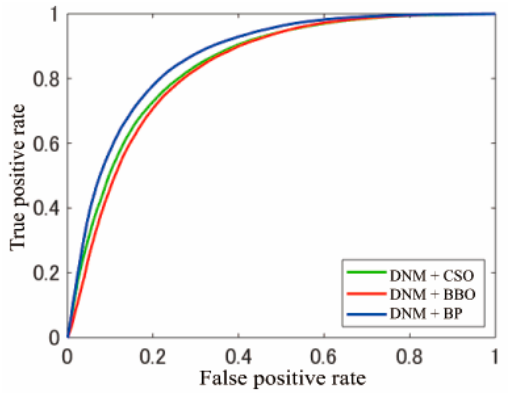

Figure 9. Receiver operating characteristic (ROC) of each method in F1.

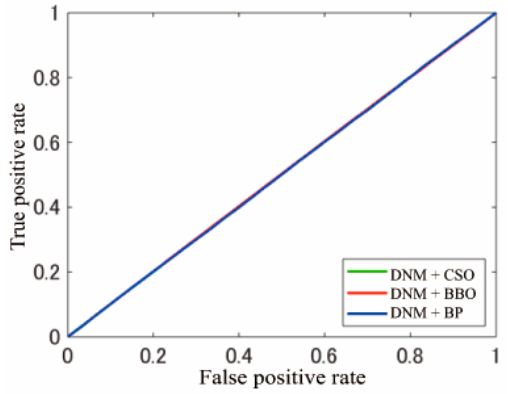

Figure 10. ROC of each method in F2.

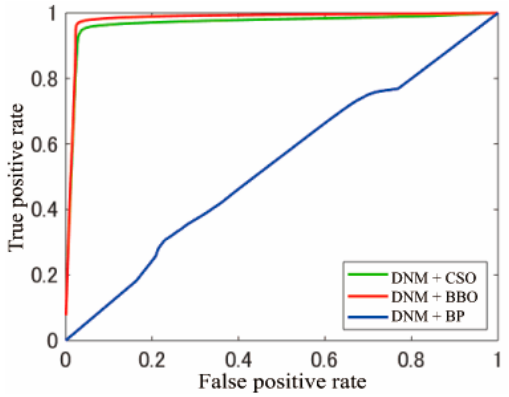

Figure 11. ROC of each method in F3.

Table 18. Average area under curve (AUC) for each problem and method.

No.	Learning Algorithm	Average AUC
F1	DNM + BP	0.868
	DNM + BBO	0.835
	DNM + CSO	0.847
F2	DNM + BP	0.500
	DNM + BBO	0.501
	DNM + CSO	0.501
F3	DNM + BP	0.542
	DNM + BBO	0.981
	DNM + CSO	0.967

According to Figures 9 and 11, DNM + BP in F1 and DNM + BBO in F3 obtain the highest classification accuracies. On the other hand, the three methods overlap on the diagonal line in Figure 10, and the results are similar to those in the case of random classification since the value of AUC is very close to 0.5, as shown in Table 18.

Although DNM + BBO and DNM + CSO differ to some degree in Figures 9 and 11, both of them are convex curves to the upper left. In particular, the AUC of F3 is close to 1, which shows that their classification accuracy is excellent. However, for DNM + BP, the best AUC is a convex curve to the upper left in F1, while F3 is a curve that approximates the diagonal.

On the other hand, in Tables 13 and 18, the difference in accuracy between DNM + BBO and DNM + CSO in F1 and F3 is also reflected in the AUC. To the contrary, even though the difference in accuracy of DNM + BP in F1 and F3 is only about 1%, the AUC is about 0.3. This is because the value of O_p that outputs the error classification result contains many values independent of the set threshold value of DNM + BP in F3. As shown in Figure 5b, the DNM + BP in F3 is trapped in the local solution since it fails to obtain the output with a higher classification accuracy.

As DNM + BP in F3 above, the value of O_p that outputs the error classification result contains many values independent of the set threshold value as shown in Figure 10, and the results are almost arranged on the same diagonal by any method in F2. The data set is considered to be the reason for this.

Moreover, because the output range of the DNM is [0, 1], the upper limit of the classifiable class is 2. In this experiment, in order to classify in the DNM, all of the classes representing the abnormality of F2 are unified as a non-anomaly class. It can be seen that the output with a high classification accuracy is not available since the different data trends are aggregated in the class representing each abnormality. Therefore, depending on the network of the DNM, etc., it is possible to expand the output range of the DNM effectively for improvement.

In addition, the average rank of the methods in F1, F2, and F3 obtained by the Friedman test are shown in Table 19. It is clear that DNM + BP in F1, DNM + CSO in F2, and DNM + BBO in F3 ranked

the highest on average, that there is no difference in the average accuracy for each problem, and that the result of this test is proved to be significant.

Table 19. Average ranks of methods in F1, F2 and F3.

No.	Learning Algorithm	Average Rank
F1	DNM + BP	1.4333
	DNM + BBO	2.4333
	DNM + CSO	2.1333
F2	DNM + BP	2.5
	DNM + BBO	2.0667
	DNM + CSO	1.4333
F3	DNM + BP	2.9
	DNM + BBO	1.1
	DNM + CSO	2

5. Discussion

According to Tables A1–A3 in the appendix, Table 20 shows the standard deviation of the average accuracy of each method. It can be seen that the accuracy of DNM + BP is the most affected by the combination of the parameters in F1 and F2. In the experiment for optimum parameter selection, the stability is slightly poor. However, the result of DNM + BP in F3 shows that it is not affected by the parameters and is trapped in the local solution with stability. In addition, the average accuracy of the DNM + CSO test in F3 is 20.78% of No. 21 as shown in appendix in Table A3, which is the lowest average accuracy among all methods.

Table 20. Standard deviation of average accuracy of each method for the tests.

No.	Learning Algorithm	Average Rank
F1	DNM + BP	15.65
	DNM + BBO	2.04
	DNM + CSO	1.75
F2	DNM + BP	21.15
	DNM + BBO	2.72
	DNM + CSO	2.53
F3	DNM + BP	1.06
	DNM + BBO	4.86
	DNM + CSO	14.49

Therefore, in terms of the stability of parameters, no matter which learning algorithm is used, the accuracy will deviate according to compatibility with the problem and the combination of parameters. Due to the nature of neural networks, it is difficult to predict the accuracy deviation based on parameters and learning algorithms, so a variety of methods should be performed for the experiment.

6. Conclusions

With the arrival of the era of big data, research into high-precision models with simple structures and low cost for addressing complex problems is developing rapidly. As a neuron model, the DNM has been proven to be more accurate than the MLP in small-scale classification problems. This study focused on the application of the DNM in complex problems and verified its effectiveness in large-scale classification problems. The DNM, as the model; BP, the most famous method for using the gradient descent to calculate the cost; BBO, with a high classification accuracy for small-scale problems; and CSO, which has the characteristic of low computational cost, were used as the learning algorithms in this experiment.

The comparison results for the three large-scale classification problems with the ANE-NNRW show that any learning algorithm using the DNM can achieve a higher accuracy than the ANE-NNRW.

However, they lag behind the ANE-NNRW in terms of execution time. In order to improve this situation, it is necessary to parallelize the parts of the DNM and reduce the computing cost.

Moreover, according to the applied three large-scale classification problems, the precision and classification accuracy of each DNM method are different. This experiment compared each learning algorithm in various aspects. In terms of execution time, DNM + BP is the optimum; DNM + CSO is the best to ensure both accuracy stability and short execution time; and considering the stability of comprehensive performance and convergence rate, DNM + BBO is a wise choice. In the future, for seeking stability independent of the problem, we will attempt to expand the output range of the DNM and employ it across a wider range of fields, e.g., Internet of Vehicles [47–50] and complex networks [51–53]. In addition, recent advanced evolutionary algorithms, e.g., chaotic differential evolution [54], can also be an alternative training method for the DNM.

Author Contributions: Conceptualization, D.J., C.L., Y.T. and H.D.; Data curation, D.J. and Y.F.; Formal analysis, C.L. and H.D.; Funding acquisition, D.J., C.L., Y.T. and H.D.; Investigation, H.D.; Methodology, D.J., Y.F., C.L. and Y.T.; Project administration, C.L., Y.T. and H.D.; Resources, D.J., C.L. and H.D.; Software, D.J. and Y.F.; Supervision, D.J., C.L. and H.D.; Validation, D.J., Y.F., C.L., Y.T. and H.D.; Writing—original draft, D.J.; Writing—review & editing, D.J., Y.F., C.L., Y.T. and H.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the JSPS KAKENHI Grant Number JP19K12136, the National Natural Science Foundation of China under Grant 61873105, the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant 19KJB160001, the Lianyungang city Haiyan project Grant 2019-QD-004, the "six talent peaks" high level talent selection and training support project of Jiangsu Province Grant XYDXX-140, and the "521 project" scientific research project support plan of Lianyungang City Grant LYG52105-2018030 and LYG52105-2018040.

Acknowledgments: The authors would like to thank all collaborators for their time and all reviewers for their valuable suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1 shows the average accuracy of 30 replicate experiments for whole parameter combinations in F1. Similarly, Table A2 is for F2, and Table A3 is for F3.

Table A1. Average accuracy of the learning and tests of each learning algorithm in F1.

No.	DNM + BP		DNM + BBO		DNM + CSO	
	Learning (%)	Test (%)	Learning (%)	Test (%)	Learning (%)	Test (%)
1	35.10	35.31	73.24	73.21	76.08	75.96
2	68.83	68.86	78.23	78.16	76.61	76.54
3	35.15	35.19	76.97	76.85	77.70	77.43
4	35.19	35.10	77.38	77.36	77.91	77.74
5	35.18	35.13	76.59	76.52	76.72	76.53
6	82.35	82.23	80.36	80.44	81.62	81.20
7	39.12	39.11	77.43	77.54	77.62	77.39
8	38.99	39.01	76.43	76.35	78.13	77.97
9	51.89	51.75	78.34	78.26	75.89	75.90
10	40.05	39.83	77.56	77.38	77.32	77.16
11	35.08	35.35	80.88	80.93	81.63	81.60
12	68.36	68.51	80.27	80.04	80.35	80.17
13	55.89	55.74	79.21	78.85	79.71	79.52
14	58.37	58.53	78.25	78.25	78.77	78.54
15	45.83	45.72	77.51	77.52	78.09	78.00
16	67.31	67.28	81.71	81.69	81.68	81.36
17	45.20	45.21	77.83	77.65	79.31	79.22
18	74.17	74.29	75.58	75.64	78.78	78.82
19	60.15	60.15	73.17	73.25	76.53	76.52
20	40.71	40.88	75.98	75.86	76.88	76.94
21	35.14	35.21	78.82	78.76	81.49	81.45
22	82.40	82.40	76.66	76.63	79.46	79.58
23	49.69	49.62	75.78	75.58	78.45	78.32
24	65.82	65.97	76.67	76.60	78.62	78.37
25	45.52	45.27	77.34	77.31	79.62	79.56

Table A2. Average accuracy of the learning and tests of each learning algorithm in F2.

No.	DNM + BP		DNM + BBO		DNM + CSO	
	Learning (%)	Test (%)	Learning (%)	Test (%)	Learning (%)	Test (%)
1	78.60	78.59	79.81	79.76	83.67	83.64
2	86.38	86.35	89.05	89.04	96.06	96.10
3	54.30	54.34	90.15	90.11	94.47	94.50
4	23.65	23.66	86.87	86.83	95.63	95.56
5	21.45	21.27	87.43	87.48	94.61	94.58
6	85.11	85.12	86.05	86.09	93.44	93.49
7	67.93	67.79	90.67	90.70	94.80	94.80
8	29.71	29.72	88.15	88.19	95.78	95.81
9	72.49	72.49	88.77	88.84	94.75	94.73
10	60.92	60.80	87.12	87.08	92.82	92.90
11	69.00	68.95	90.75	90.73	96.08	96.05
12	78.79	78.77	92.09	92.07	94.83	94.78
13	76.87	76.99	91.02	91.03	95.29	95.36
14	88.00	87.94	88.87	88.88	93.15	93.17
15	66.77	66.59	89.57	89.51	96.02	96.00
16	78.60	78.59	91.40	91.34	94.76	94.74
17	65.27	65.19	91.88	91.83	95.76	95.74
18	90.97	90.96	88.03	87.99	93.19	93.20
19	85.29	85.31	88.61	88.55	95.33	95.32
20	77.19	77.30	91.47	91.43	96.38	96.37
21	25.21	25.23	92.11	92.13	95.68	95.69
22	89.08	89.19	87.52	87.63	92.48	92.49
23	75.43	75.38	90.42	90.50	95.60	95.58
24	80.27	80.30	92.41	92.40	96.01	96.01
25	66.58	66.59	91.66	91.64	96.23	96.24

Table A3. Average accuracy of the learning and tests of each learning algorithm in F3.

No.	DNM + BP		DNM + BBO		DNM + CSO	
	Learning (%)	Test (%)	Learning (%)	Test (%)	Learning (%)	Test (%)
1	79.25	79.24	78.62	78.60	91.47	91.48
2	79.26	79.22	97.20	97.19	94.64	94.63
3	79.25	79.23	96.36	96.37	94.30	94.30
4	79.25	79.23	95.82	95.82	93.96	93.99
5	79.24	79.27	96.05	96.05	94.22	94.22
6	79.26	79.22	96.10	96.11	93.71	93.74
7	79.25	79.24	96.86	96.86	94.44	94.43
8	79.24	79.26	96.60	96.59	93.81	93.62
9	79.25	79.24	95.59	95.57	93.29	93.30
10	82.20	82.18	95.83	95.82	93.32	93.33
11	79.23	79.29	97.43	97.44	95.05	95.04
12	79.23	79.28	96.97	96.95	93.91	93.93
13	79.26	79.22	95.55	95.56	92.82	92.78
14	79.26	79.22	96.18	96.17	91.66	91.66
15	79.25	79.24	96.62	96.61	93.46	93.49
16	79.25	79.25	90.67	90.66	88.38	88.37
17	79.25	79.23	93.37	93.39	93.02	93.02
18	83.76	83.76	96.55	96.54	92.85	92.84
19	79.25	79.25	96.37	96.37	93.00	93.01
20	79.25	79.25	96.30	96.30	91.23	91.24
21	79.26	79.22	79.70	79.78	20.79	20.78
22	79.24	79.27	96.65	96.65	91.11	91.14
23	79.26	79.22	97.09	97.11	90.89	90.93
24	79.24	79.25	95.20	95.17	91.84	91.82
25	79.23	79.27	92.22	92.20	91.50	91.48

References

1. Gao, S.C.; Zhou, M.C.; Wang, Y.R.; Cheng, J.J.; Yachi, H.; Wang, J.H. Dendritic neural model with effective learning algorithms for classification, approximation, and prediction. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 601–604. [CrossRef] [PubMed]
2. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]

3. Koch, C.; Poggio, T.; Torre, V. Nonlinear interactions in a dendritic tree: Localization, timing, and role in information processing. *Proc. Nat. Acad. Sci. USA* **1983**, *80*, 2799–2802. [CrossRef] [PubMed]
4. Brunel, N.; Hakim, V.; Richardson, M.J. Single neuron dynamics and computation. *Curr. Opin. Neurobiol.* **2014**, *25*, 149–155. [CrossRef] [PubMed]
5. Cazé, R.D.; Jarvis, S.; Foust, A.J.; Schultz, S.R. Dendrites enable a robust mechanism for neuronal stimulus selectivity. *Neural Comput.* **2017**, *29*, 2511–2527. [CrossRef]
6. Almufti, S.; Marqas, R.; Ashqi, V. Taxonomy of bio-inspired optimization algorithms. *J. Adv. Comput. Sci. Technol.* **2019**, *8*, 23–31. [CrossRef]
7. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
8. Gao, S.C.; Song, S.B.; Cheng, J.J.; Todo, Y.; Zhou, M.C. Incorporation of Solvent Effect into Multi-objective Evolutionary Algorithm for Improved Protein Structure Prediction. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2018**, *15*, 1365–1378. [CrossRef]
9. Todo, Y.; Tamura, H.; Yamashita, K.; Tang, Z. Unsupervised learnable neuron model with nonlinear interaction on dendrites. *Neural Netw.* **2014**, *60*, 96–103. [CrossRef]
10. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Let a biogeography based optimizer train your multi-layer perceptron. *Inf. Sci.* **2014**, *269*, 188–209. [CrossRef]
11. Jain, A.K.; Bhasin, S. Tracking control of uncertain nonlinear systems with unknown constant input delay. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 420–425. [CrossRef]
12. Sha, Z.J.; Hu, L.; Todo, Y.; Ji, J.K.; Gao, S.C.; Tang, Z. A breast cancer classifier using a neuron model with dendritic nonlinearity. *IEICE Trans. Inf. Syst.* **2015**, *98*, 1365–1376. [CrossRef]
13. Ji, J.K.; Gao, S.C.; Cheng, J.J.; Tang, Z.; Todo, Y. An approximate logic neuron model with a dendritic structure. *Neurocomputing* **2016**, *173*, 1775–1783. [CrossRef]
14. Jiang, T.; Gao, S.C.; Wang, D.; Ji, J.K.; Todo, Y.; Tang, Z. A neuron model with synaptic nonlinearities in a dendritic tree for liver disorders. *IEEJ Trans. Elect. Electron. Eng.* **2017**, *12*, 105–115. [CrossRef]
15. Zhou, T.L.; Gao, S.C.; Wang, J.; Chu, C.; Todo, Y.; Tang, Z. Financial time series prediction using a dendritic neuron model. *Knowl. Based Syst.* **2016**, *105*, 214–224. [CrossRef]
16. Chen, W.; Sun, J.; Gao, S.C.; Cheng, J.J.; Wang, J.; Todo, Y. Using a single dendritic neuron to forecast tourist arrivals to Japan. *IEICE Trans. Inf. Syst.* **2017**, *100*, 190–202. [CrossRef]
17. He, K.M.; Zhang, X.Y.; Ren, S.Q.; Sun, J. Deep Residual Learning for Image Recognition. *IEEE Conf. Comput. Vis. Pattern Recognit.* **2016**. [CrossRef]
18. Yang, X.; Zhao, B. Optimal neuro-control strategy for nonlinear systems with asymmetric input constraints. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 575–583.
19. Ye, H.L.; Cao, F.L.; Wang, D.H.; Li, H. Building feedforward neural networks with random weights for large scale datasets. *Expert Syst. Appl.* **2018**, *106*, 233–243. [CrossRef]
20. Yu, Y.; Gao, S.C.; Wang, Y.R.; Todo, Y. Global optimum-based search differential evolution. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 379–394. [CrossRef]
21. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; University of California: San Diego, CA, USA, 1985.
22. Widrow, B.; Lehr, M.A. 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *Proc. IEEE* **1990**, *78*, 1415–1442. [CrossRef]
23. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. Available online: https://arxiv.org/abs/1609.04747 (accessed on 16 March 2020).
24. Zhang, Y.; Zhou, P.; Cui, G.M. Multi-model based PSO method for burden distribution matrix optimization with expected burden distribution output behaviors. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 1506–1512. [CrossRef]
25. Jia, D.; Yanagisawa, K.; Ono, Y.; Hirobayashi, K.; Hasegawa, M.; Hirobayashi, S.; Tagoshi, H.; Narikawa, T.; Uchikata, N.; Takahashi, H. Multiwindow nonharmonic analysis method for gravitational waves. *IEEE Access* **2018**, *6*, 48645–48655. [CrossRef]
26. Jia, D.; Yanagisawa, K.; Hasegawa, M.; Hirobayashi, S.; Tagoshi, H.; Narikawa, T.; Uchikata, N.; Takahashi, H. Time-frequency-based non-harmonic analysis to reduce line noise impact for LIGO observation system. *Astron. Comput.* **2018**, *25*, 238–246. [CrossRef]

27. Jia, D.; Dai, H.; Takashima, Y.; Nishio, T.; Hirobayashi, K.; Hasegawa, M.; Hirobayashi, S.; Misawa, T. EEG Processing in Internet of Medical Things Using Non-Harmonic Analysis: Application and Evolution for SSVEP Responses. *IEEE Access* **2019**, *7*, 11318–11327. [CrossRef]
28. Abbott, L.F.; Regehr, W.G. Synaptic computation. *Nature* **2004**, *431*, 796–803. [CrossRef]
29. Jan, Y.N.; Jan, L.Y. Branching out: Mechanisms of dendritic arborization. *Nat. Rev. Neurosci.* **2010**, *11*, 316–328. [CrossRef]
30. Simon, D. Biogeography-based optimization. *IEEE Trans. Evolut. Comput.* **2008**, *12*, 702–713. [CrossRef]
31. Li, R.M.; Huang, Y.F.; Wang, J. Long-term traffic volume prediction based on K-means Gaussian interval type-2 fuzzy sets. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 1344–1351. [CrossRef]
32. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS'95, Nagoya, Japan, 4–6 October 1995; Volume 1, pp. 39–43.
33. Yu, Y.; Gao, S.C.; Wang, Y.R.; Cheng, J.J.; Todo, Y. ASBSO: An Improved Brain Storm Optimization with Flexible Search Length and Memory-based Selection. *IEEE Access* **2018**, *6*, 36977–36994. [CrossRef]
34. Cheng, R.; Jin, Y. A competitive swarm optimizer for large scale optimization. *IEEE Trans. Cybern.* **2014**, *45*, 191–204. [CrossRef] [PubMed]
35. Khan, A.H.; Cao, X.W.; Li, S.; Katsikis, V.N.; Liao, L.F. BAS-ADAM: An ADAM based approach to improve the performance of beetle antennae search optimizer. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 461–471.
36. Guan, S.W.; Wang, Y.P.; Liu, H.Y. A New Cooperative Co-evolution Algorithm Based on Variable Grouping and Local Search for Large Scale Global Optimization. *J. Netw. Intell.* **2017**, *2*, 339–350.
37. Gao, S.C.; Wang, Y.R.; Cheng, J.J.; Inazumi, Y.; Tang, Z. Ant colony optimization with clustering for solving the dynamic location routing problem. *Appl. Math. Comput.* **2016**, *285*, 149–173. [CrossRef]
38. Qian, X.X.; Wang, Y.R.; Cao, S.; Todo, Y.; Gao, S.C. Mr2DNM: A Novel Mutual Information-Based Dendritic Neuron Model. *Comput. Intell. Neurosci.* **2019**, *2019*, 7362931. [CrossRef] [PubMed]
39. Cheng, J.J.; Cheng, J.L.; Zhou, M.C.; Liu, F.Q.; Gao, S.C.; Liu, C. Routing in Internet of Vehicles: A Review. *IEEE Trans. Intell. Transp.* **2015**, *16*, 2339–2352. [CrossRef]
40. Gandhi, R.V.; Adhyaru, D.M. Takagi-Sugeno fuzzy regulator design for nonlinear and unstable systems using negative absolute eigenvalue approach. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 482–493. [CrossRef]
41. Gabbiani, F.; Krapp, H.G.; Koch, C.; Laurent, G. Multiplicative computation in a visual neuron sensitive to looming. *Nature* **2002**, *420*, 320–324. [CrossRef]
42. Khaw, J.F.C.; Lim, B.S.; Lim, L.E.N. Optimal design of neural networks using the Taguchi method. *Neurocomputing* **1995**, *7*, 225–245. [CrossRef]
43. Dong, W.; Zhou, M. Gaussian classifier-based evolutionary strategy for multimodal optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 1200–1216.
44. Mahapatro, S.R.; Subudhi, B.; Ghosh, S. Design of a robust optimal decentralized PI controller based on nonlinear constraint optimization for level regulation: An experimental study. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 187–199. [CrossRef]
45. Roy, P.; Mahapatra, G.S.; Dey, K.N. Forecasting of software reliability using neighborhood fuzzy particle swarm optimization based novel neural network. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 1365–1383.
46. UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 18 March 2020).
47. Cheng, J.J.; Yuan, G.Y.; Zhou, M.C.; Gao, S.C.; Huang, Z.H.; Liu, C. A Connectivity Prediction-based Dynamic Clustering Model for VANET in an Urban Scene. *IEEE Internet Things* **2020**. In Press. [CrossRef]
48. Cheng, J.J.; Yuan, G.Y.; Zhou, M.C.; Gao, S.C.; Liu, C.; Duan, H.; Zeng, Q.T. Accessibility Analysis and Modeling for IoV in an Urban Scene. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4246–4256. [CrossRef]
49. Cheng, J.J.; Yuan, G.Y.; Zhou, M.C.; Gao, S.C.; Liu, C.; Duan, H. A Fluid Mechanics-based Data Flow Model to Estimate VANET Capacity. *IEEE Trans. Intell. Transp.* **2020**. [CrossRef]
50. Cui, J.; Wei, L.; Zhong, H.; Zhang, J.; Xu, Y.; Liu, L. Edge Computing in VANETs-An Efficient and Privacy-Preserving Cooperative Downloading Scheme. *IEEE J. Sel. Areas Commun.* **2020**. [CrossRef]
51. Cheng, J.J.; Qin, P.Y.; Zhou, M.C.; Gao, S.C.; Huang, Z.H.; Liu, C. A Novel Method for Detecting New Overlapping Community in Complex Evolving Networks. *IEEE Trans. Syst. Man Cybern.* **2019**, *49*, 1832–1844. [CrossRef]

52. Cheng, J.J.; Chen, M.J.; Zhou, M.C.; Gao, S.C.; Liu, C.M.; Liu, C. Overlapping Community Change Point Detection in an Evolving Network. *IEEE Trans. Big Data* **2020**, *6*, 189–200. [CrossRef]
53. Sun, J.; Gao, S.C.; Dai, H.W.; Cheng, J.J.; Zhou, M.C.; Wang, J.H. Bi-objective Elite Differential Evolution Algorithm for Multivalued Logic Networks. *IEEE Trans. Cybern.* **2020**, *50*, 233–246. [CrossRef]
54. Gao, S.C.; Yu, Y.; Wang, Y.R.; Wang, J.H.; Cheng, J.J.; Zhou, M.C. Chaotic Local Search-based Differential Evolution Algorithms for Optimization. *IEEE Trans. Syst. Man Cybern.* **2019**. [CrossRef]