IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 3, 2019, accepted September 23, 2019, date of publication September 30, 2019,
date of current version October 10, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2944682*

# Training an Approximate Logic Dendritic Neuron Model Using Social Learning Particle Swarm Optimization Algorithm

**SHUANGYU SONG[1], XINGQIAN CHEN[1], CHENG TANG[1], SHUANGBAO SONG[1], ZHENG TANG[1], AND YUKI TODO[2], (Member, IEEE)**

[1]Faculty of Engineering, University of Toyama, Toyama 930-8555, Japan
[2]Faculty of Electrical and Computer Engineering, Kanazawa University, Kanazawa 920-1192, Japan

Corresponding author: Yuki Todo (yktodo@ec.t.kanazawa-u.ac.jp)

This work was supported by the JSPS KAKENHI under Grant JP19K12136.

**ABSTRACT** With the rapid development of artificial neural networks, recent studies have shown that dendrites play a vital role in neural computations. In this study, we propose a dendritic neuron model called the approximate logic dendritic neuron model (ALDNM) to solve classification problems. The ALDNM can be divided into four layers: the synaptic layer, the dendritic layer, the membrane layer, and the soma body. Considering the limitation of the back-propagation (BP) algorithm, we employ a heuristic optimization called the social learning particle swarm optimization algorithm (SL-PSO) to train the ALDNM. In order to investigate the effectiveness of SL-PSO for training the ALDNM, we compare this training method with BP and four other typical heuristic optimization methods. Moreover, the proposed ALDNM is also compared with seven classifiers to verify its performance. The experimental results and statistical analysis on four classification problems indicate that the proposed ALDNM trained by SL-PSO can provide a competitive performance for solving the classification problems. It is worth emphasizing that the structure of the trained ALDNM can be greatly simplified owing to the unique pruning operations. Furthermore, the simplified ALDNM for a specific problem can be converted into a corresponding logic circuit classifier for a fast classification.

**INDEX TERMS** dendritic neuron model, heuristic optimization, classification, pruning, logic circuit.

## I. INTRODUCTION

The human cerebral cortex is surprisingly complex. It consists of approximately 100 billion neurons and $10^{15}$ interconnections. A typical neuron structure is divided into three parts: the cell body (soma), an axon, and many dendrites. The neuron receives signals via specialized connections called synapses and sends signals to its cell body. Then, signals cross from the axon to another neuron. In 1943, Warren MuCulloch and Walter Pitts creatively proposed a computational neuron model called the McCulloch-Pitts model (M-P model) by mimicking the functionality of a biological neuron [1]. This model is considered to be the origin of artificial neural networks (ANNs) and lays the foundation for neural network models. However, this model contains only a nonlinear term on the cell body, and the nonlinear mechanisms on the dendrites are completely ignored [2].

With the development of neurobiology, the important role of dendritic structures in neural computation has aroused the great interest of researchers [2]–[6]. To investigate the interaction between synaptic signals (inhibitory and excitatory), Koch, Poggio, and Torre proposed a dendritic neuron model called the $\delta$ cell model [7], [8]. Subsequent experiments provided strong support for Kochs model. For example, Taylor et al. [9] demonstrated a key role for postsynaptic dendritic processing in neuronal computation by studying direction-selective ganglion cells. Another example is that Segev [10] investigated the sound grounds for computing dendrites in the auditory brain stem. However, the $\delta$ cell model is incapable of reducing the redundant branches of the trees when solving all given tasks, which means that the dendritic structure is fixed [11]. In [12], Legensen and Maass exploited the competition between dendritic branches

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

to enable a single neuron to obtain nonlinear computational capabilities. However, this model still cannot solve nonlinear separated problems, such as the simplest XOR problem.

The development of biological neurology in recent years has revealed the importance of neuronal pruning, including axon pruning [13] and dendritic pruning [14]. Neuronal pruning is the process of removing redundant connections between neurons in the brain. This process allows fewer connections and changes to ensure that the brain is more able to focus in-depth on complex tasks. In our previous works, based on these biophysical phenomenon, a variety of neuron models were proposed to deal with several real-world problems, including credit-risk evaluation [15], breast cancer diagnosis [16], financial time series prediction [17], and the diagnosis of liver disorders [18].

For an artificial neuron model, the training process is an extremely important aspect. The goal of training is to minimize classification error by finding the optimal combination of parameters [19]. The back-propagation (BP) algorithm is a traditional gradient-based algorithm that utilizes the gradient information of the error function to adjust the weight of the neurons in the negative gradient direction. It has been widely used in training neuron models [20], [21]. However, the BP algorithm is extremely dependent on initial conditions, which leads to some shortcomings such as high probability of local minima entrapment [22]–[24] and a difficult learning rate setting [25]. Although the BP algorithm has been well applied in our previous research, the scale of these problems is considered small, and larger-scale problems are not well explored. As the complexity of the classification problems increases, the shortcomings of the BP algorithm have severely limited the ability of these dendritic neuron models. This motivates us to find a more promising learning algorithm.

In recent years, heuristic optimization methods have received great attention from researchers due to their powerful performance for solving many practical problems [26]–[29]. Specifically, recent literature has shown the advantages of using heuristic optimization methods to train ANNs. For example, in [30], a particle swarm optimization (PSO) algorithm is adopted to train ANNs to predict water levels in a river in Hong Kong. Mirjalili et al. investigated the efficiency of a biogeography-based optimization (BBO) algorithm in training a multilayer perceptron (MLP) [31]. In [32], the authors conducted a comprehensive study of randomized algorithms for training ANNs. Additional research in [33] employed a states of matter search (SMS) for training a dendritic neuron model. These successful applications motivate our attempts to investigate the efficiencies of heuristic optimization methods in training neuron models.

In this study, a novel neuron model, namely ALDNM, is proposed for solving the classification problems. In detail, ALDNM is made up of a synaptic layer, a dendritic layer, a membrane layer, and a soma (cell body). Considering the limitations of the BP algorithm and the superiority of the heuristic optimization algorithm, especially the promising performance of the recently proposed SL-PSO [34] for solving complex and high-dimensional problems, SL-PSO is employed as the learning method. Four classification datasets are used to examine the effectiveness of the ALDNM and the training method employed. The experimental results indicate that SL-PSO is the most promising method to train the ALDNM, compared with BP and four other typical heuristic optimization methods. Moreover, when compared with seven traditional classifiers, the proposed model ALDNM trained by SL-PSO can also provide very competitive results. Finally, the highlights of the proposed ALDNM are also investigated and analyzed. A trained ALDNM can be simplified by neuronal structure pruning, including synaptic pruning and dendritic pruning. Further, a simplified ALDNM can even be converted into a logical circuit classifier. This logical circuit classifier only contains digital competitor, NOT gate, AND gate, and OR gate. It is worth noting that if the logical circuit classifier is implemented in hardware, the classification speed of this classifier will be greatly improved.

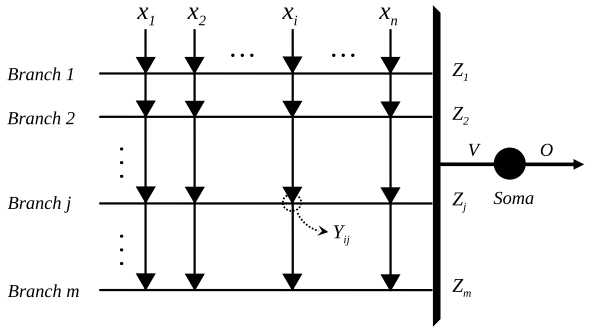



**FIGURE 1.** The structure of the ALDNM.

The remainder of this paper is organized as follows: Section 2 describes the architecture of the ALDNM. Section 3 introduces the learning method, neuronal structure pruning, and logic circuit transformation process. The experimental study is described in Section 4. Finally, we draw conclusions in Section 5.

## II. MATERIALS

### A. APPROXIMATE LOGIC DENDRITIC NEURON MODEL

Inspired by the biological neuron model and the dendritic mechanism, we proposed a neuron model called the ALDNM in this study. The ALDNM consists of four layers: a synaptic layer, a dendritic layer, a membrane layer, and a soma body. Fig. 1 shows the architecture of the ALDNM, where $n$ denotes the input number and $m$ represents the number of branches, and thus, the total number of synapses is $m * n$. The synaptic layer receives incoming signals from the previous neuron and processes a sigmoid function for the received signals. Then an AND operation is performed between synapses on each branch. All the dendritic branches of the dendritic layer are connected to the membrane layer, and the interaction between these branches corresponds to a logic OR operation. Finally, the soma processes a nonlinear computation on the signals from the previous layer.

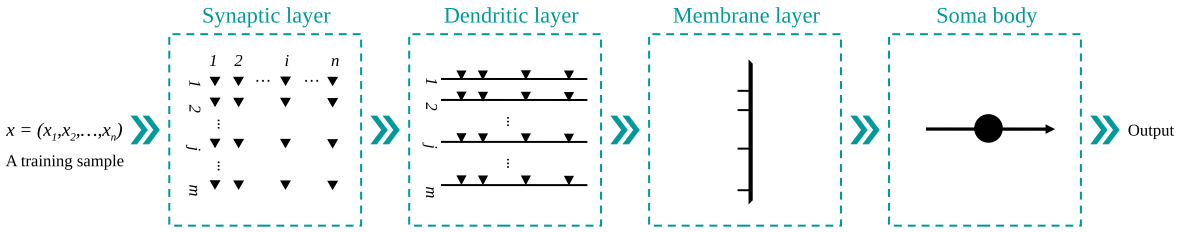



**FIGURE 2.** An example of the classification process of the ALDNM.

#### 1) SYNAPTIC LAYER

This layer denotes the synaptic connection organization from a neuron to a post neuron; its signal transfer is feed-forward. Whether the synapse is excited or inhibited depends on a certain specific ion that can cause a change in synaptic potential. The sigmoid function is adopted to process the synaptic connections. The connection function of the synaptic layer from the $i$th ($i = 1, 2, , n$) input to the $j$th ($j = 1, 2, , m$) branch is shown as follows:

$$Y_{ij} = \frac{1}{1 + e^{-c(w_{ij}x_i - q_{ij})}}, \tag{1}$$

where $x_i$ is the synaptic input, and ranges in $[0, 1]$. $c$ represents a constant parameter. $w_{ij}$ and $q_{ij}$ denote connection parameters that need to be adjusted during the learning process.

#### 2) DENDRITIC LAYER

It has been known that multiplication operations exist in neurons to process neural information [35]. For each branch, the dendritic layer executes a multiplicative operation on the synaptic connections. Since the synaptic signals of the dendritic layer are nearly binary, the operation can be replaced by the logic AND operation. The output of the $j$th branch can be expressed as follows:

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

#### 3) MEMBRANE LAYER

Each branch of the dendritic layer is connected to the membrane layer. This layer executes the summation operation on the results of all branches. This operation can be replaced by the logic OR operation if the signals are binary. Then, the result of the membrane layer is sent to the last layer. The output equation of the membrane layer is formulated as follows:

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

#### 4) SOMA BODY

The soma (cell body) is the last layer of the neuron model. It performs a nonlinear computation on the received result. The neuron will fire if the input signal exceeds a predefined threshold. The calculation can be expressed as follows:

$$O = \frac{1}{1 + e^{-c_{soma}(V - \gamma)}}, \tag{4}$$

where $\gamma$ is the threshold constant. $V$ represents the output of the membrane layer. $c_{soma}$ is a constant parameter. $O$ represents the final output of the soma.

To explain the structure of ALDNM more clearly, we provide an example to show the classification process of the ALDNM. Figure 2 shows the operation steps and the basic components of each layer. The operation steps of ALDNM are described as follows.

**Step 1**: In the synaptic layer, m branches cross with input (n dimensions) to form $m * n$ synaptic connections (▼). Each synaptic connection produces an output by (1). Then, the $m*n$ outputs are transmitted to the dendritic layer.

**Step 2**: In the dendritic layer, each branch receives the corresponding n inputs. The output of each branch is obtained by (2). The generated m outputs are transmitted to the membrane layer.

**Step 3**: The membrane layer processes the outputs of m branches by (3). Then it sends the result to the soma layer.

**Step 4**: In the soma body, the sigmoid function is used to process the output of the membrane layer to obtain the classification result.

The numbers of inputs and outputs for each layer of ALDNM are summarized in Table 1. The inputs of each layer come from the outputs of the previous layer.

**TABLE 1.** The number of inputs and outputs for each layer.

| Layer name | Num. of inputs | Num. of outputs |
|---|---|---|
| Synaptic layer | n | m*n |
| Dendritic layer | m*n | m |
| Membrane layer | m | 1 |
| Soma body | 1 | 1 |

### B. CONNECTION STATES

Initially, the parameters ($w_{ij}$ and $q_{ij}$) in (1) are random values in the range $[-1.5, 1.5]$, which means that all synaptic connections are random states. $\theta_{ij} = q_{ij}/w_{ij}$ is used to represent the threshold of the synapse, which actually represents the center of the sigmoid function on the x-axis. Depending on the different values of $w_{ij}$ and $q_{ij}$, the connection states are divided into six cases. After further analyzing the relationship between the input values and the output values of these six cases, the connection states can be classified into four types: direct connection (●), inverse connection (■), constant 1 connection (①), and constant 0 connection (⓪). Fig. 3 illustrates the four types of connection states.

To explain how a synapse connects a dendritic branch in four states, the four function types of connection states

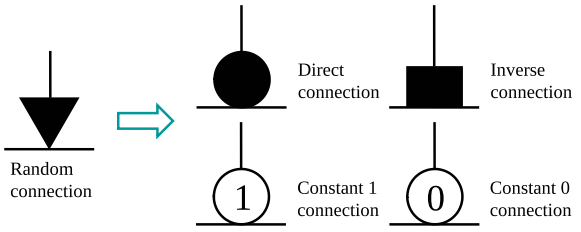



**FIGURE 3.** The four types of connection states.

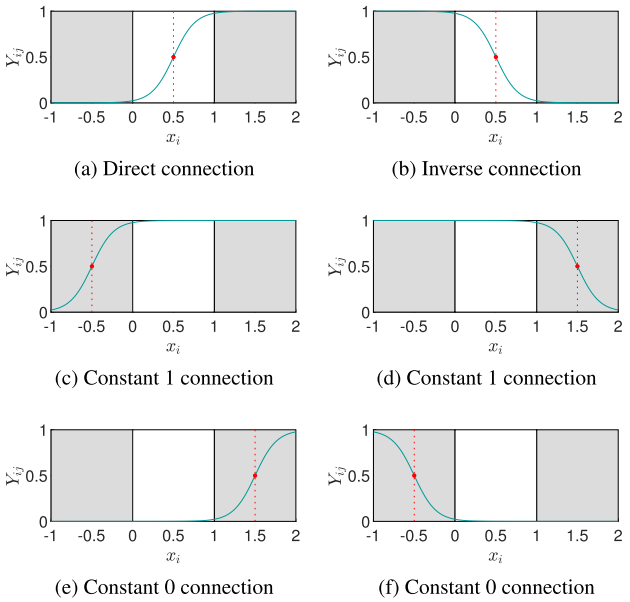



**FIGURE 4.** Six cases of connection states.

are plotted in Fig. 4. The description can be summarized as follows:

Case (a): $0 < q_{ij} < w_{ij}$, for example, $w_{ij} = 1$, $q_{ij} = 0.5$, and $\theta_{ij} = 0.5$. In this case, an input greater than $\theta$ leads to a high output, and an input less than $\theta_{ij}$ leads to a low output. Therefore, this case is defined as a direct connection.

Case (b): $w_{ij} < q_{ij} < 0$, for example, $w_{ij} = -1$, $q_{ij} = -0.5$, and $\theta_{ij} = 0.5$. Contrary to case (a), an input greater than $\theta_{ij}$ leads to a low output, and an input less than $\theta_{ij}$ leads to a high output. This case is called an inverse connection.

Case (c): $q_{ij} < 0 < w_{ij}$, for example, $w_{ij} = 1$, $q_{ij} = -0.5$, and $\theta_{ij} = -0.5$. The output is always 1 regardless of how the input changes. This case is called a constant 1 connection.

Case (d): $q_{ij} < w_{ij} < 0$, for example, $w_{ij} = -1$, $q_{ij} = -1.5$, and $\theta_{ij} = 1.5$. The output of this case is also always 1. Thus, this case is defined as a constant 1 connection.

Case (e): $0 < w_{ij} < q_{ij}$, for example, $w_{ij} = 1$, $q_{ij} = 1.5$, and $\theta_{ij} = 1.5$. In this case, the output is always 0 regardless of how the input changes. This case is called a constant 0 connection.

Case (f): $w_{ij} < 0 < q_{ij}$, for example, $w_{ij} = -1$, $q_{ij} = 0.5$, and $\theta_{ij} = -0.5$. Similarly, this case is a constant 0 connection because the output of this connection is always 0.

Obviously, in the interval [0, 1], the outputs in case (c) and case (d) are always high outputs, while the outputs in case (e) and case (f) are always low outputs. This finding means that these connection states have little impact on the final results of classification problems. In other words, ALDNM can discard these unnecessary connection cases, only the direct connection and the inverse connection play crucial roles in the structure of the ALDNM.

## III. METHODOLOGY

### A. TRAINING METHOD

The characteristics of the training algorithm have a great impact on the capabilities of the neural network model [36], [37]. In our previous works [15]–[18], the BP algorithm was proven an effective training method to train dendritic neuron models. However, when solving more complex tasks, the BP algorithm suffers from its shortcomings and greatly limits the computation capacity of dendritic neuron models [38]. Therefore, finding a high-performance training algorithm to train the ALDNM becomes extremely necessary.

PSO is a widely used swarm intelligence algorithm introduced by Kennedy and Eberhart [39]. Although PSO has been well applied to many optimization problems, its performance on multilocal optima and high-dimensional problems is still unsatisfactory [40]. To address this issue, Chen et al. proposed an SL-PSO algorithm by introducing social learning mechanisms [41], [42] into a classical PSO algorithm. The swarm update mechanism of SL-PSO is completely different from those of traditional PSO variants. Each particle in classical PSO learns from global best solutions and personal best solutions, which is based on historical information. On the contrary, each particle in SL-PSO can learn from any better particle in a current swarm.

SL-PSO firstly initializes $m$ particles of the swarm $P(t)$ in a random way, where $t$ represents the generation index. Then, the algorithm randomly initializes a behavior vector $X_i(t)$ for each particle $i$. After that, the algorithm enters the main loop until the termination condition is reached. Fig. 5 illustrates the entire optimization process of SL-PSO.

According to Fig. 5, there are three main steps between generation $t$ and generation $t + 1$, including fitness evaluation, swarm sorting and behavioral learning. First, the fitness of each particle in the current swarm is evaluated. Then, the swarm is sorted in descending order according to the fitness value, assuming it is a minimum optimization problem. Finally, each particle learns from particles with better fitness in the current swarm. Note that, during the optimization process, the best particle will not be updated and the worst particle will not be learned by other particles.

Like social learning mechanisms among social animals, a particle will learn the behaviors of different particles. The learning rule for $j$th dimension of particle $i$ in the $t$th generation is expressed as:

$$X_{i,j}(t+1) = \begin{cases} X_{i,j}(t) + \bigtriangleup X_{i,j}(t+1) & \text{if } p_i(t) \leqslant P_i^L, \\ X_{i,j}(t) & \text{otherwise,} \end{cases} \quad (5)$$

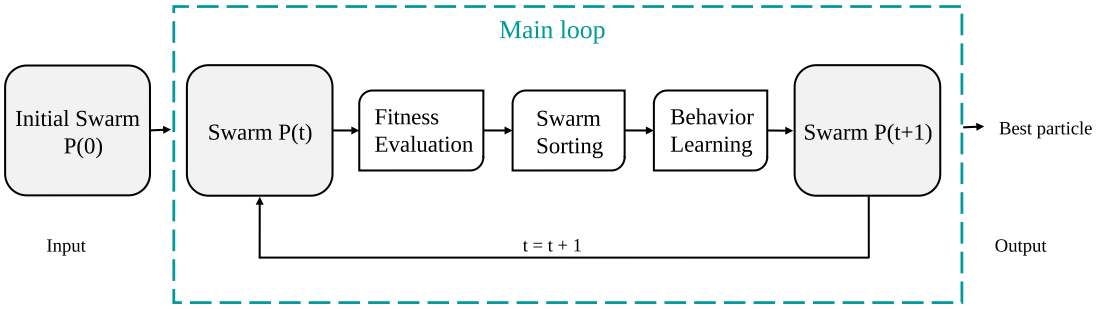



**FIGURE 5.** Main components of the SL-PSO.

where $\Delta X_{i,j}(t+1)$ represents the behavior correction,with i $\in$ $\{1, 2, ..., m\}$ and j $\in$ $\{1, 2, ..., n\}$. $X_{i,j}(t)$ shows the $j$th dimension of behavior vector. $P_i^L$ is the learning probability corresponding to each particle $i$, and $p_i$ indicates a randomly generated probability. In detail, $\Delta X_{i,j}(t+1)$ is calculated as follows:

$$\begin{aligned}\Delta X_{i,j}(t+1) = {} & r_1(t) \cdot \Delta X_{i,j}(t) \\ & + r_2(t) \cdot (X_{k,j}(t) - X_{i,j}(t)) \\ & + r_3(t) \cdot \epsilon \cdot (\bar{X}_j(t) - X_{i,j}(t)),\end{aligned} \tag{6}$$

where $\epsilon$ is the influence factor. $r_1(t)$, $r_2(t)$, and $r_3(t)$ are random coefficients in the interval [0, 1].

In the above equation, the behavior correction $\Delta X_{i,j}(t+1)$ contains three parts, namely, $\Delta X_{i,j}(t)$, $X_{k,j}(t) - X_{i,j}(t)$ and $\bar{X}_j(t) - X_{i,j}(t)$. The first part is the behavior modification of $t$th generation. In the second part, $X_{i,j}(t)$ imitates $X_{k,j}(t)$. Specifically, $k$ is independent for each dimension $j$, which means that particle $i$ may learn from different particles in the current swarm. In the last part, each particle $i$ learns from the mean behavior of all particles. $\bar{X}_j(t)$ is calculated as follows:

$$\bar{X}_j(t) = \frac{\sum_{i=1}^{m} X_{i,j}}{m}. \tag{7}$$

Specifically, SL-PSO uses dimension-dependent parameter control strategies to control swarm size $m$, learning probability $P_i^L$, and impact coefficient $\epsilon$. The swarm size $m$ is given by

$$m = M + \left\lfloor \frac{n}{G} \right\rfloor, \tag{8}$$

where $M$ is the base size that is preset to 100. $n$ represents the dimension of the behavior vector, and $G$ is a constant parameter equal to 70. The learning probability $P_i^L$ mentioned above is calculated as:

$$P_i^L = \left(1 - \frac{i-1}{m}\right)^{\alpha \cdot log(\lceil \frac{n}{M} \rceil)}, \tag{9}$$

where $\alpha$ is a positive coefficient and is set to 5. This indicates that the worse the fitness of the particle is, the higher the learning probability will be. The last parameter $\epsilon$ can be calculated as:

$$\epsilon = \beta \times \frac{n}{M}, \tag{10}$$

where $\beta$ is a small constant and is set to 0.01.

Due to these dimension-dependent parameters ($m$, $P_i^L$, and $\epsilon$), a good balance between convergence and diversity is achieved. Extensive experiments in previous papers have been executed to verify the performance of SL-PSO. Specifically, SL-PSO has achieved excellent results in dealing with high-dimensional problems.

To adopt SL-PSO in training the ALDNM, the first step is the problem representation. The goal of training the ALDNM with SL-PSO is to find a set of parameter values to obtain the highest classification accuracy. Each synaptic connection has two parameters ($w$ and $q$). Thus, the number of parameters that need to be adjusted can be expressed as:

$$N = 2 \times n \times m, \tag{11}$$

where $n$ denotes the input number and $m$ represents the number of branches. Each particle in the SL-PSO is represented by a vector, which is shown as follows:

$$\begin{aligned}\mathbf{X} &= (\mathbf{W}, \mathbf{Q}) \\ &= (w_{1,1}, w_{1,2}, \cdots, w_{n,m}, q_{1,1}, q_{1,2}, \cdots, q_{n,m}).\end{aligned} \tag{12}$$

The direct goal of the training process is to make the difference between the actual output and the ideal output smaller. Thus, the mean square error (MSE) of the ALDNM is employed as the fitness function of SL-PSO. The MSE is computed as follows:

$$fitness = \frac{1}{2S} \sum_{s=1}^{S} (T_s - O_s)^2, \tag{13}$$

where $T_s$ and $O_s$ represent the ideal output and actual output, respectively. $S$ denotes the number of training samples.

In order to explain the training process more clearly, we present the flowchart of the training process in Fig. 6. The flowchart contains three main parts: training set, SL-PSO, and ALDNM. Their roles are described as follows.

- Training set: providing training samples for the ALDNM.
- SL-PSO: updating particles to get the best fitness. The particles of SL-PSO are represented by (12). The fitness function of a particle is calculated by (13). When the stopping criterion is met, the SL-PSO outputs the particle with the best fitness value.
- ALDNM: calculating the actual outputs of all samples as shown in Fig. 2. Then, the value of MSE calculated in (13) is returned to SL-PSO as the fitness of a particle.

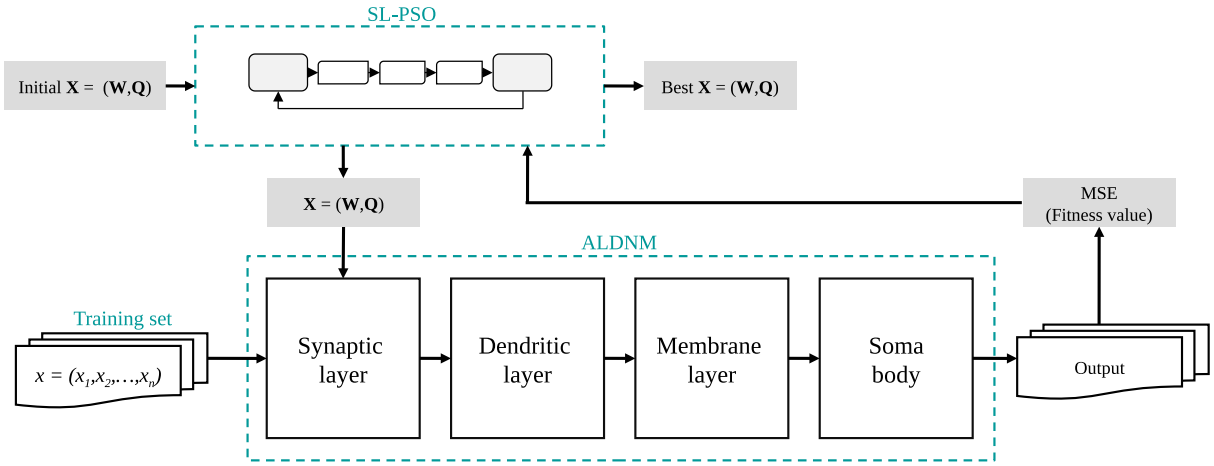



**FIGURE 6.** The flowchart of the training process.

### B. NEURONAL STRUCTURE PRUNING

After the training process, the trained ALDNM has a set of determined parameters $w_{ij}$s and $q_{ij}$s. Then, the neuronal structure pruning operations can be performed on it to eliminate unnecessary synapses and branches, depending on the four connection states described above. The pruning process can be divided into two steps: synaptic pruning and dendritic pruning.

a) synaptic pruning: in the dendritic layer, the operation between synapses is multiplication. For a constant 1 connection, the output result is always 1. Multiplying any value by 1 will produce itself, which suggests that the constant 1 connection has no effect on the result of the branch. The synapse of this connection needs to be omitted from the branch.

b) dendritic pruning: the output result is always 0 if it is a constant 0 connection. Any number multiplied by 0 is equal to 0. Thus if a constant 0 connection occurs, the output result of this branch will always be equal to 0. This dendritic branch should be removed from the dendrites.

After the structure pruning process is completed, the structure of the model becomes simpler than before. Fig. 7 illustrates an example of the pruning process for a trained model. The original structure contains two branches with six synaptic connections. After pruning the neural structure, it retains only one branch with two synaptic connections. For each specific problem, the ALDNM can generate a corresponding unique and simple structure. In a sense, it has an implicit feature selection mechanism.

### C. TRANSFORMING THE ALDNM INTO A LOGICAL CIRCUIT

Through the neuronal structure pruning process, the ALDNM retains only the direct connections and the inverse connections, forming a unique simple topology. Then, the simplified ALDNM can be further transformed into a logic circuit. Only comparators, NOT gates, AND gates, and OR gates are used in the logic circuit, as shown in Fig. 8. For the synaptic layer, a direct connection can be equivalent to a comparator, and a reverse connection can be replaced by a comparator coupled with a NOT gate. In the dendritic layer, a branch with several connections is equivalent to an AND gate. A membrane layer connected by several branches is equivalent to an OR gate. The cell body is a simple nonlinear mapping and is equivalent to a single wire. Through these transformations, an ALDNM can be transformed into a logic circuit classifier. As the equivalent classifier of the ALDNM, it has a very high classification speed because it does not have floating-point computation but only needs logical operations.

## IV. EXPERIMENTAL STUDY

This section presents the experiments to verify the performance of the proposed model. All algorithms in this study are implemented in Python and C languages. All experiments are performed on a Linux 64-bit system with a Core-i5 CPU, 3.4 GHz, and 8 GB memory.

In our experiments, four classification datasets are used to verify the performance of the ALDNM trained by SL-PSO. These four datasets are Wine, Climate model simulation crashes (CMSC), Wisconsin diagnostic breast cancer (WDBC), and Ionosphere. All datasets of these classification problems are collected from the UCI machine learning repository [43], and the details of four datasets are summarized in Table 2. The Wine dataset includes the chemical composition analysis results of wines in specific regions of Italy. The origin of the wine can be inferred from the chemical composition. This dataset contains 178 records, each with 13 attributes. The CMSC is used to predict climate model simulation outcomes given climate model parameter values. It contains a total of 540 parameter value combinations, each with 18 values. All attribute values of this dataset are scaled in the interval [0, 1]. The simulation outcomes are represented by 0 (success) and 1 (failure). The WDBC dataset was provided by Dr. William H. Wolberg et al. from the University of Wisconsin for breast cancer diagnosis. The features are obtained by computing the digitized image of

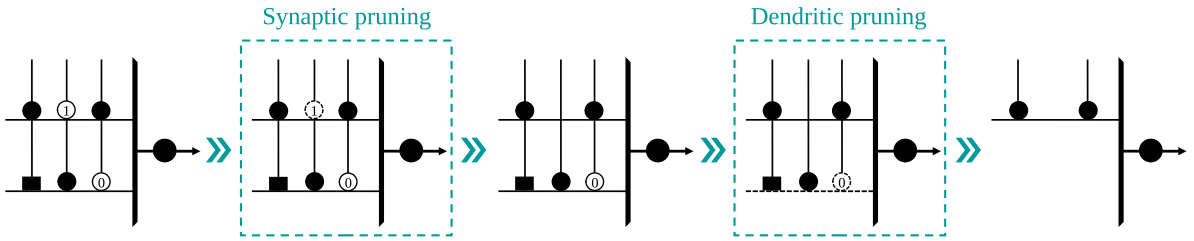



**FIGURE 7.** Two steps of the structure pruning operation: synaptic pruning and dendritic pruning.

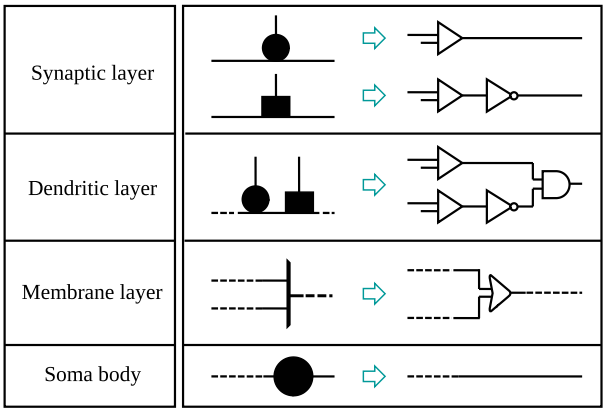



**FIGURE 8.** An example of each layer and its equivalent logical circuit component.

**TABLE 2.** The details of four datasets.

| Dataset | Num. of classes | Num. of features | Num. of samples |
|---|---|---|---|
| Wine | 2 | 13 | 178 |
| CMSC | 2 | 18 | 540 |
| WDBC | 2 | 30 | 569 |
| Ionosphere | 2 | 34 | 351 |

breast mass. This dataset includes records from 569 samples, each with 30 feature items. The diagnosis results are divided into two categories: benign (B) and malignant (M). The Ionosphere dataset records the Ionosphere data collected by Johns Hopkins University. The goal is to determine the type of radar return by analyzing the free electrons in the ionosphere. Returns can be classified as g (good) and b (bad). The dataset contains 351 cases, each with 34 continuous attributes within $[-1, 1]$. It should be noted that, since the proposed ALDNM is a binary classifier, all of the employed test classification problems are binary. When dealing with multiclass classification problems, we can use some existing multiclass classification techniques, such as the One-vs-One strategy [45] and the One-vs-All strategy [46], to extend ALDNM for these problems. This issue deserves our future investigation.

According to our previous studies [16], [24], [33], the constant paramaters $c$, $c_{soma}$, and $\gamma$ of ALDNM are set to 5, 5, and 0.5, respectively. Each dataset is randomly split into two subsets in one experiment. One subset is used for training, and the other one is used for testing. The proportions of each subset are set to 50%∼50%. For each dataset, the split operation is performed 30 times, forming 30 pairs of subsets. After running the proposed model, 30 experimental results are obtained. To fit the input of the ALDNM, all features are normalized in the interval [0, 1].

### A. COMPARATIVE STUDY WITH THE BP ALGORITHM

The BP algorithm uses gradient information to minimize the function error, which is completely based on mathematical concepts. In our previous works, the BP algorithm was applied to the benchmark datasets such as XOR, Iris, Glass, and Cancer. The number of features in these benchmark problems is less than 10. However, the number of features in the datasets in this experiment is greater than 10 (13, 18, 30, and 34). More features mean an increase in parameters, which is a challenge for the performance of BP algorithm.

In this section, SL-PSO is compared to the BP algorithm in training the ALDNM. For a fair comparison between SL-PSO and BP, the maximum function evaluation number of SL-PSO is 20000, and the maximum number of iterations of BP is 10000. Even in this setup, the BP algorithm still spends more memory and training time than SL-PSO. The experimental results, including the mean square error (MSE), accuracy and p-value, are provide in Table 3.

From Table 3, it is clear that the MSEs of SL-PSO are smaller than those of BP on all benchmark problems. It implies that SL-PSO is more powerful than BP in training the ALDNM. By comparing the accuracy rate of SL-PSO with that of BP, it is clear that the results obtained by SL-PSO are better than those obtained by BP. The Wine dataset is the simplest problem among the four classification problems. BP provided a competitive result for this dataset. However, BP showed poor classification accuracy on the other three problems. The Wilcoxon signed-ranks test is utilized in our experiments to detect significant differences between SL-PSO and BP in term of accuracy rate. In Table 3, SL-PSO is the control algorithm. All p-values of the four problems are smaller than 0.05, indicating that SL-PSO outperforms BP significantly in classification accuracy. Moreover, Fig. 9 exhibits box-and-whisker diagrams of classification accuracy for two training algorithms on four benchmark problems. SL-PSO has higher medians and minimums on all benchmark problems. Moreover, SL-PSO has a shorter intentional range in all benchmark problems, indicating that SL-PSO has a more stable performance of training ability.

To further evaluate the performance of two training algorithms, the average convergence curves of two algorithms for four benchmark problems are demonstrated in Fig. 10.

**TABLE 3.** The experimental results of SL-PSO and BP on four benchmark problems.

| Dataset | Algorithm | MSE | Accuracy(%) | $R^+$ | $R^-$ | p-value |
|---|---|---|---|---|---|---|
| Wine | SL-PSO | 2.56e-03±4.76e-04 | 98.54±0.83 | - | - | - |
| | BP | 2.18e-02±3.29e-02 | 95.47±7.80 | 407.5 | 57.5 | **0.000026** |
| CMSC | SL-PSO | 2.03e-02±4.26e-03 | 92.96±1.59 | - | - | - |
| | BP | 1.33e-01±1.58e-01 | 69.95±36.55 | 435.5 | 29.5 | **0.00001** |
| WDBC | SL-PSO | 2.58e-02±3.92e-03 | 94.36±1.34 | - | - | - |
| | BP | 1.63e-01±9.73e-03 | 63.27±2.29 | 465 | 0 | **0.000001** |
| Ionosphere | SL-PSO | 2.96e-02±4.61e-03 | 92.39±1.73 | - | - | - |
| | BP | 2.75e-01±1.08e-02 | 35.93±2.54 | 465 | 0 | **0.000001** |

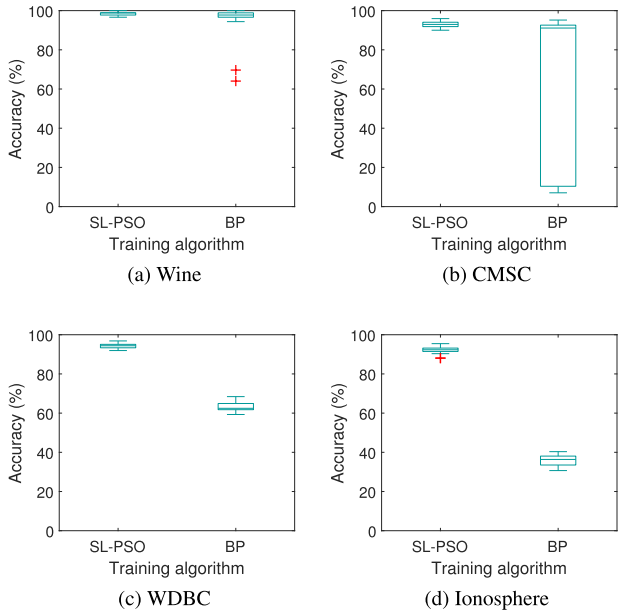



**FIGURE 9.** The boxplot graph of accuracy for four problems.

The convergence curve reflects the stability and efficiency of an algorithm. It is quite clear that SL-PSO provides a much faster convergence rate than BP on each classification problem. Specifically, the convergence curves of BP are almost horizontal on two problems (WDBC and Ionosphere) with more than 30 features, which means that BP is invalid on these datasets. These results show that SLPSO has a obvious advantage over BP in training the ALDNM.

### B. COMPARATIVE STUDY WITH OTHER HEURISTIC ALGORITHMS

In this subsection, we compare the performance of the SL-PSO algorithm with four typical heuristic optimization methods. Two of the four algorithms are the classical PSO [40] and its variant multiswarm particle swarm optimization (MSPSO) [47], which focus on improving the topology of the classical PSO. The remaining two algorithms are the state-of-the-art genetic algorithm (GA) [48] and differential evolution (DE) [49]. For a fair comparison, the parameter settings of all algorithms are based on studies in the literature [40], [47], [50], [51]. Table 4 gives the details of the parameter settings.

Table 5 lists the results achieved by these heuristic optimization algorithms for the four classification problems.

**TABLE 4.** The parameter settings of compared heuristic optimization algorithms.

| Algorithm | Parameter | Value | Description |
|---|---|---|---|
| PSO | $w$ | 0.5 | Inertia factor |
| | $c_1$ | 1.5 | Cognitive constant |
| | $c_2$ | 1.5 | Social constant |
| | $N$ | 50 | Population size |
| MSPSO | $w$ | 0.5 | Inertia factor |
| | $c_1$ | 1.5 | Cognitive constant |
| | $c_2$ | 1.5 | Social constant |
| | $M$ | 5 | Number of subpopulation |
| | $N$ | 10 | Subpopulation size |
| GA | $p_c$ | 0.9 | Crossover rate |
| | Crossover type | Single point | |
| | $p_m$ | 0.1 | Mutation rate |
| | Selection | Roulette wheel | |
| | $N$ | 50 | Population size |
| | $N_e$ | 10 | Number of elites |
| DE | $CR$ | 0.9 | Crossover rate |
| | $F$ | 0.5 | Scaling factor |
| | $N$ | 50 | Population size |

**TABLE 5.** The classification results of SL-PSO and other heuristic optimization algorithms.

| Algorithm | Accuracy (%) | | | |
|---|---|---|---|---|
| | Wine | CMSC | WDBC | Ionosphere |
| SL-PSO | **98.54 ± 0.83** | **92.96 ± 1.59** | **94.36 ± 1.34** | **92.39 ± 1.73** |
| PSO | 98.01 ± 1.44 | 91.57 ± 1.69 | 92.74 ± 1.64 | 87.48 ± 4.11 |
| MSPSO | 97.75 ± 1.51 | 91.48 ± 1.62 | 90.56 ± 2.32 | 78.54 ± 4.53 |
| GA | 98.46 ± 1.03 | 91.74 ± 1.75 | 92.63 ± 1.71 | 88.33 ± 3.42 |
| DE | 98.20 ± 1.15 | 91.68 ± 1.67 | 93.27 ± 1.49 | 88.96 ± 3.68 |

It is clear that SL-PSO achieves the highest classification accuracy rate for each classification problem among five algorithms. The Friedman test is used to detect differences between multiple groups. Here, we use it to determine whether there is a statistical difference between SL-PSO and other algorithms. The statistical results are listed in Table 6, where SL-PSO is the control algorithm. The ranking values obtained by the Friedman test can evaluate the performance of each algorithm. A smaller ranking value indicates a better performance. As shown in Table 6, SL-PSO achieves the smallest ranking value 1.825. In addition, since the unadjusted p-values ignore the familywise error rare [52] in a multiple comparison. A common post-hoc procedure called Bonferroni-Dunn procedure is applied to obtain adjusted p-value expressed as $p_{Bonf}$. In Table 6, all $p_{Bonf}$ values are less than the significance level of 0.05, which indicates that there is a statistical difference between SL-PSO and other algorithms. Therefore, we can conclude that SL-PSO provides the

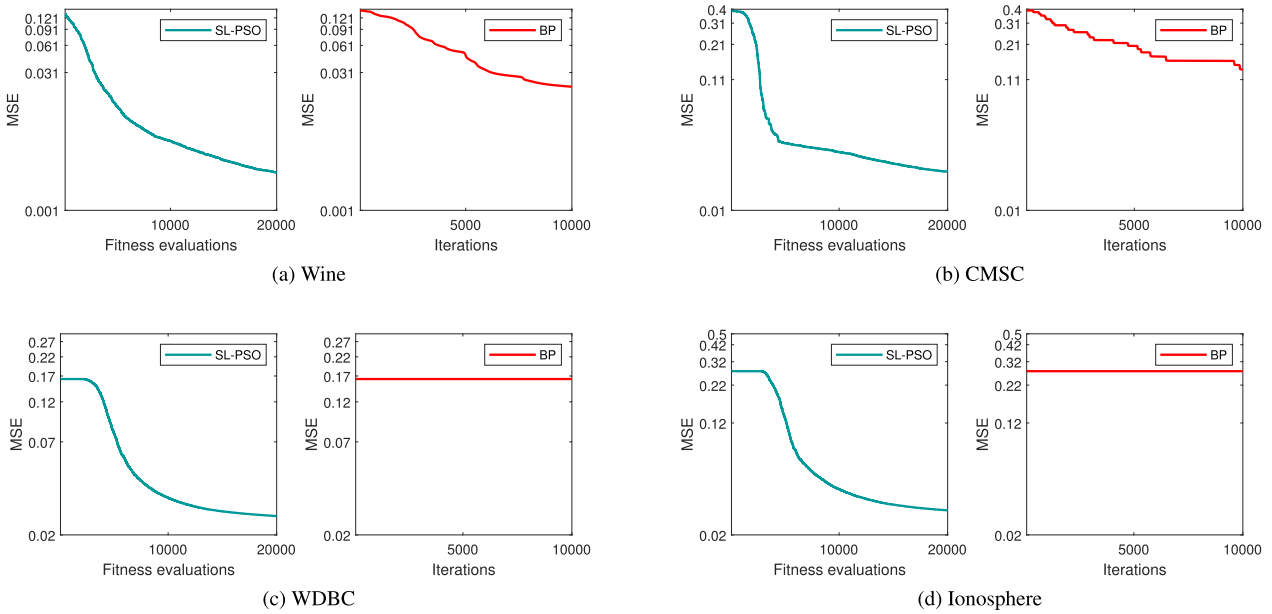



**FIGURE 10.** The convergence curves of two training algorithms for four problems.

**TABLE 6.** Statistical analysis of SL-PSO and other heuristic algorithms.

| Algorithm | Ranking | Unadjusted $p$ | $p_{Bonf}$ |
|---|---|---|---|
| SL-PSO | 1.825 | - | - |
| PSO | 3.2583 | 0 | **0** |
| MSPSO | 4.1083 | 0 | **0** |
| GA | 2.9417 | 0 | **0** |
| DE | 2.8667 | 0 | **0.000001** |

most powerful performance in training the ALDNM among these heuristic optimization methods.

To further investigate the differences among these algorithms in training the ALDNM, the average convergence curves for four benchmark problems are plotted in Fig. 11. As shown in Fig. 11, although other heuristic optimization methods converge faster than SL-PSO at the beginning, they trap into local minima quickly. Additionally, the MSEs obtained by SL-PSO are the smallest among the five algorithms. It suggests that SL-PSO is a highly suitable algorithm for training the ALDNM.

The reasons for the excellent performance of SL-PSO in training the ALDNM can be explained as follows. SL-PSO contains a social learning mechanism, which maintains a better balance of exploitation and exploration. Thus, the issues of prematurity and falling into local minima can be avoided. In addition, SL-PSO has a parameter control strategy dependent on dimension, which can improve the robustness for optimization problems of different dimensions.

### C. COMPARATIVE STUDY WITH OTHER CLASSIFIERS

All the above experimental results indicate that SL-PSO is a promising algorithm in training the ALDNM. In addition to comparison with other heuristic algorithms, we also compare the ALDNM with seven other widely used classifiers to investigate the performance of ALDNM training

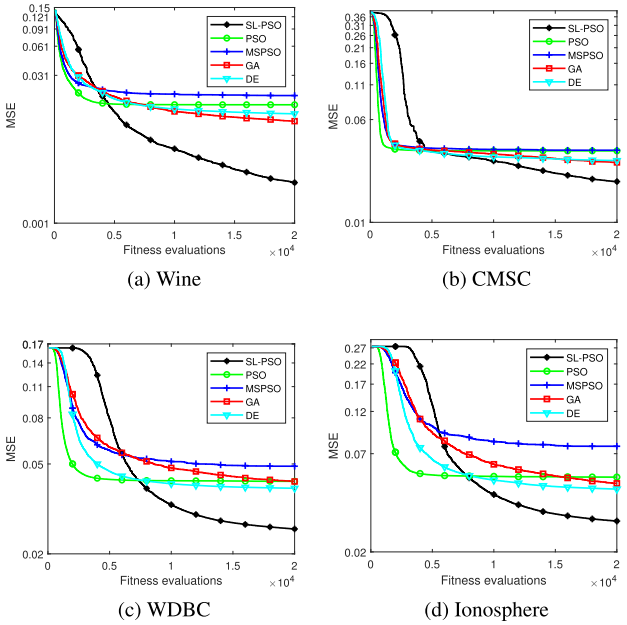



**FIGURE 11.** The convergence curves of SL-PSO and other heuristic optimization algorithms.

by SL-PSO. The compared classifiers are the k-nearest neighbors algorithm (k-NN) [53], linear SVM [44], decision tree [54], random forest [55], MLP [20], naive Bayes (NB) [56], and quadratic discriminant analysis (QDA) [57]. The parameters of these classifiers are set according to the recommendation of the scikit-learn library [58]. The same subsets for four classification problems are implemented on each classifier 30 times. The classification accuracies of all classifiers on the four classification problems are summarized in Table 7.

**TABLE 7. Comparison of the accuracy of different classifiers for four problems.**

| Classifier | Accuracy (%) | | | |
|---|---|---|---|---|
| | Wine | CMSC | WDBC | Ionosphere |
| ALDNM | **98.54±0.83** | 92.96±1.59 | 94.36±1.34 | **92.39±1.73** |
| k-NN | 97.57±1.42 | 92.07±1.76 | 96.48±0.86 | 83.75±2.37 |
| linear SVM | 67.19±2.67 | 91.51±1.66 | 90.63±1.49 | 71.27±4.93 |
| Decision Tree | 93.60±3.35 | 90.07±1.74 | 92.50±1.65 | 88.67±3.04 |
| Random Forest | 95.06±2.68 | 91.51±1.66 | 93.52±1.38 | 88.18±2.91 |
| MLP | 97.38±1.24 | 92.46±2.16 | **96.49±0.89** | 86.00±7.61 |
| NB | 97.04±1.49 | **93.81±1.54** | 93.12±1.35 | 87.88±2.47 |
| QDA | 97.94±1.48 | 91.47±1.64 | 95.25±1.10 | 83.71±9.49 |

**TABLE 8. Statistical analysis of ALDNM in comparison with other classifiers.**

| Algorithm | Ranking | Unadjusted $p$ | $p_{Bonf}$ |
|---|---|---|---|
| ALDNM | 2.6833 | - | - |
| k-NN | 3.8375 | 0.000262 | **0.001837** |
| linear SVM | 7.2625 | 0 | **0** |
| Decision Tree | 5.7125 | 0 | **0** |
| Random Forest | 5.1958 | 0 | **0** |
| MLP | 3.2542 | 0.071054 | 0.497377 |
| NB | 3.8792 | 0.000156 | **0.001091** |
| QDA | 4.175 | 0.000002 | **0.000017** |

From Table 7, we can see that the ALDNM achieves the best classification accuracies on the Wine and Ionosphere datasets. On the CMSC dataset, the classification accuracy obtained by the ALDNM is second only to NB. On the WDBC datasets, although the ALDNM does not achieve the best classification accuracy, it also achieves competitive result compared with other classifiers.

In order to further determine the significant differences between the accuracies of eight classifiers, the Friedman test is also employed here. The statistical results of the Friedman test are summarized in Table 8. Except for the MLP, all unadjusted p-values of the other classifiers are smaller than 0.05. This indicates that the ALDNM trained by SL-PSO is significantly superior to k-NN, SVM, decision tree, random forest, NB, and QDA. The adjusted p-value ($p_{Bonf}$) of MLP is 0.497377. This value indicates that there is no significant difference between SL-PSO and MLP. On the other hand, according to Table 8, it can be seen that the ALDNM has the smallest ranking value of 2.6833. MLP has the second smallest ranking value 3.2542, and the linear SVM has the largest ranking value 7.2625. It can be concluded that the ALDNM is the most promising classifier on the four classification problems, MLP is the second best classifier, and the linear SVM is the worst classifier.

The comparison results show the powerful performance of ALDNM for the classification problems. The advantages of SL-PSO described above make it suitable for training the ALDNM, especially when a real-world problem is complex. The ALDNM has a unique neural topology and utilizes nonlinear computing on the neuron dendrites, which is also an important reason why the ALDNM can acquire a high performance in the classification problems.

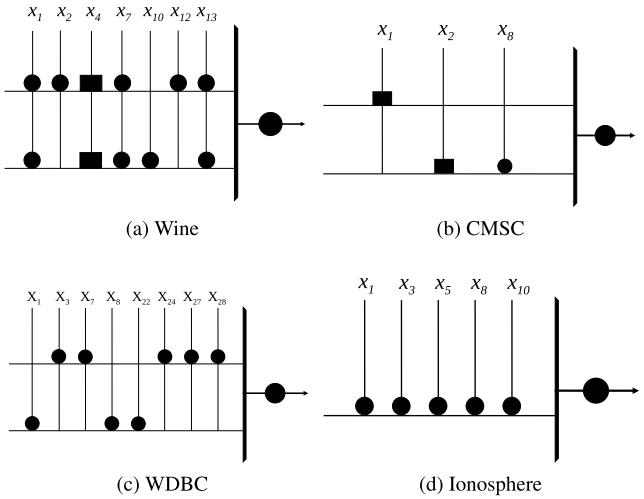



**FIGURE 12. The final simplified structures for the four benchmark problems.**

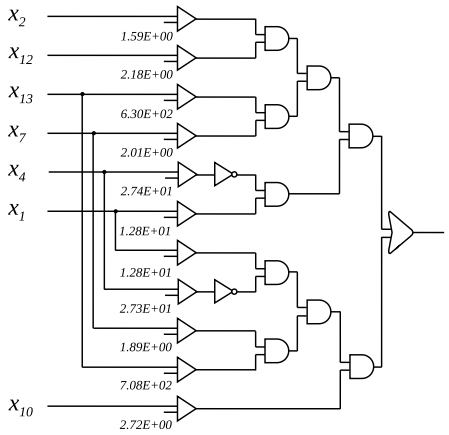
**TABLE 9. Results of original structures and simplified structures on four datasets.**

| Dataset | Num. of features | | Num. of branches | | Accuracy(%) | |
|---|---|---|---|---|---|---|
| | Original | Simplified | Original | Simplified | Original | Simplified |
| Wine | 13 | 7 | 26 | 2 | 98.88 | 100 |
| CMSC | 18 | 3 | 36 | 2 | 95.93 | 94.07 |
| WDBC | 30 | 8 | 60 | 2 | 94.74 | 91.23 |
| Ionosphere | 34 | 5 | 68 | 1 | 92.05 | 91.48 |

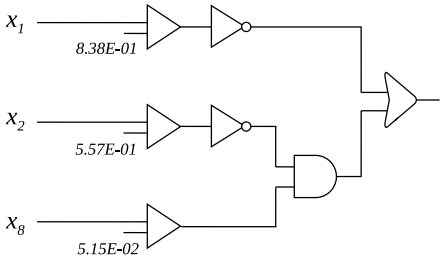
### D. LOGICAL CIRCUIT APPROXIMATION TRANSFORMATION

As introduced above, a trained ALDNM can perform two unique pruning operations (synaptic pruning and dendritic pruning) to obtain a more simplified dendritic structure. Then, a corresponding logic circuit classifier can be generated according to the simplified structure. We apply the pruning operations on a trained ALDNM for each dataset. Table 9 shows the results of the ALDNM before and after simplification on four datasets. From Table 9, it can be seen that the number of features is much less than the original structure for each classification dataset. This means that effective features are preserved, while unnecessary features are discarded. Furthermore, the number of branches of all classification problems is also greatly reduced compared to the previous ones. Among these simplified structures, three models retain two dendritic branches, and one model retains only one dendritic branch. The structure of each benchmark problem is greatly simplified by the pruning operations.

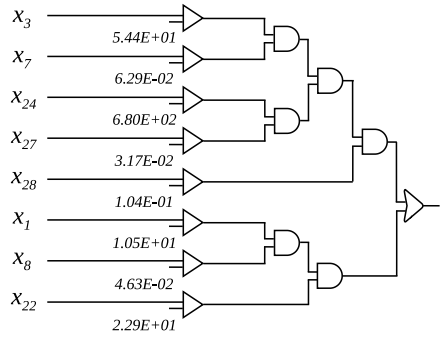
In order to have a more detailed explanation of the process of transforming the simplified ALDNM into an equivalent logic circuit, the final simplified structures and logic circuits for the four benchmark problems are expressed in Fig. 12 and Fig. 13, respectively. According to Fig. 12 (a), the classification result of the Wine classification problem is ultimately determined by 11 synaptic connections (9 direct connections and 2 inverse connections). More surprisingly, there are only three synaptic connections in the final structure of the CMSC



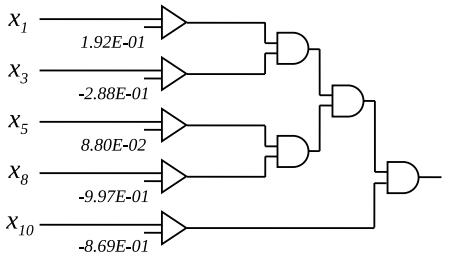

(a) Wine




(b) CMSC




(c) WDBC




(d) Ionosphere

**FIGURE 13.** **The logic circuits for the four benchmark problems.**

classification problem. The most surprising thing is that Ionosphere, which originally had 34 features, ended up with only five synaptic connections on one dendritic branch. Figure 13 shows the corresponding logic circuits of each simplified ALDNM for classification. As shown in Fig. 13, all logical circuits are composed of basic circuit units, including Comparators, NOT gates, AND gates, and OR gates. These circuits are easy to be realized in hardware and have the characteristics of fast computation speed and low cost. In particular, the simplified ALDNM and its corresponding logic circuit provide some insights into how to conclude the classification results. As illustrated in Table 9, although the logic circuits are much simplified compared to the original structures, the classification accuracy rate is not greatly sacrificed. Therefore, it is promising to convert complex classification problems into equivalent logic circuits.

## V. CONCLUSION

Based on the dendritic structure of neurons, a variety of neuron models have been proposed and used to solve many real-world problems. In this study, a novel neural model called an approximate logic dendritic neuron model was proposed. This model is made up of four layers, namely, a synaptic layer, a dendritic layer, a membrane layer, and a soma body. In our previous studies, a BP algorithm was used as a training algorithm and proved to be effective when the number of weights in neuron models is relatively small. However, on the high-dimensional classification problem, the BP algorithm shows the disadvantage of being easily trapped in local minima and slow convergence, which limits the performance of the ALDNM. As an algorithm with excellent performance on optimizing high-dimensional problems, SL-PSO was adopted to train the ALDNM in this study. The experimental results show that SL-PSO can acquire a superior performance, better than the BP algorithm and four other typical heuristic optimization algorithms.

In addition, we also compared the proposed ALDNM trained by SL-PSO with seven other widely used classifiers. The experimental results proved its superiority on these datasets. It is worth emphasizing that the ALDNM owns the ability to simplify its structure, which can remove redundant synaptic connections and dendritic branches. In fact, the simplification process can be regarded as a feature selection mechanism for each specific classification problem. Furthermore, the simplified structures can be transformed into corresponding logic circuits, which are composed of a small number of circuit elements. The experimental results on the four datasets indicate that the logic circuits can maintain high classification accuracies on the four datasets. Moreover, these logic circuits have the merit of being easy to implement in hardware with fast computation and low cost.

In the future, we intend to apply the proposed ALDNM to more complex classification problems to verify its effectiveness. Moreover, extending ALDNM to solve multiclass classification problems is worth further investigating.

## REFERENCES

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[2] M. London and M. Häusser, "Dendritic computation," *Annu. Rev. Neurosci.*, vol. 28, pp. 503–532, Jul. 2005.

[3] F. Rosenblatt, "The perceptron—A perceiving and recognizing automaton project para," Cornell Aeronaut. Lab., Buffalo, NY, USA, Tech. Rep. 85-460-1, 1957.

[4] R. Vega, G. Sanchez-Ante, L. E. Falcon-Morales, H. Sossa, and E. Guevara, "Retinal vessel extraction using lattice neural networks with dendritic processing," *Comput. Biol. Med.*, vol. 58, pp. 20–30, Mar. 2015.

[5] F. Arce, E. Zamora, C. Fócil-Arias, and H. Sossa, "Dendrite ellipsoidal neurons based on k-means optimization," *Evolving Syst.*, vol. 10, no. 3, pp. 381–396, 2018.

[6] H. Sossa, F. Arce, E. Zamora, and E. Guevara, "Morphological neural networks with dendritic processing for pattern classification," in *Proc. Adv. Topics Comput. Vis., Control Robot. Mechatronics*, V. O. Vergara, M. Nandayapa, and I. Soto, Eds. Springer, 2018, pp. 27–47.

[7] C. Koch, T. Poggio, and V. Torre, "Retinal ganglion cells: A functional interpretation of dendritic morphology," *Philos. Trans. Roy. Soc. London B, Biol. Sci.*, vol. 298, no. 1090, pp. 227–263, 1982.

[8] C. Koch, T. Poggio, and V. Torre, "Nonlinear interactions in a dendritic tree: Localization, timing, and role in information processing," *Proc. Nat. Acad. Sci. USA*, vol. 80, no. 9, pp. 2799–2802, 1983.

[9] W. R. Taylor, S. He, W. R. Levick, and D. I. Vaney, "Dendritic computation of direction selectivity by retinal ganglion cells," *Science*, vol. 289, no. 5488, pp. 2347–2350, 2000.

[10] I. Segev, "Sound grounds for computing dendrites," *Nature*, vol. 393, no. 6682, p. 207, 1998.

[11] A. Destexhe and E. Marder, "Plasticity in single neuron and circuit computations," *Nature*, vol. 431, no. 7010, p. 789, 2004.

[12] R. Legenstein and W. Maass, "Branch-specific plasticity enables self-organization of nonlinear computation in single neurons," *J. Neurosci.*, vol. 31, no. 30, pp. 10787–10802, 2011.

[13] L. K. Low and H.-J. Cheng, "Axon pruning: An essential step underlying the developmental plasticity of neuronal connections," *Phil. Trans. Roy. Soc. B, Biol. Sci.*, vol. 361, no. 1473, pp. 1531–1544, 2006.

[14] T. Kanamori, M. I. Kanai, Y. Dairyo, K.-I. Yasunaga, R. K. Morikawa, and K. Emoto, "Compartmentalized calcium transients trigger dendrite pruning in drosophila sensory neurons," *Science*, vol. 340, no. 6139, pp. 1475–1478, 2013.

[15] Y. Tang, J. Ji, S. Gao, H. Dai, Y. Yu, and Y. Todo, "A pruning neural network model in credit classification analysis," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–22, 2018, Art. no. 9390410.

[16] Z. Sha, L. Hu, Y. Todo, J. Ji, S. Gao, and Z. Tang, "A breast cancer classifier using a neuron model with dendritic nonlinearity," *IEICE Trans. Inf. Syst.*, vol. 98, no. 7, pp. 1365–1376, 2015.

[17] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, and Z. Tang, "Financial time series prediction using a dendritic neuron model," *Knowl.-Based Syst.*, vol. 105, pp. 214–224, Aug. 2016.

[18] T. Jiang, S. Gao, D. Wang, J. Ji, Y. Todo, and Z. Tang, "A neuron model with synaptic nonlinearities in a dendritic tree for liver disorders," *IEEJ Trans. Elect. Electron. Eng.*, vol. 12, no. 1, pp. 105–115, 2017.

[19] R. D. A. Araújo, A. L. Oliveira, and S. Meira, "A morphological neural network for binary classification problems," *Eng. Appl. Artif. Intell.*, vol. 65, pp. 12–28, Oct. 2017.

[20] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[21] N. Zhang, "An online gradient method with momentum for two-layer feedforward neural networks," *Appl. Math. Comput.*, vol. 212, pp. 488–498, Jun. 2009.

[22] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 1, no. 1, pp. 76–86, Jan. 1992.

[23] G. Magoulas, M. Vrahatis, and G. Androulakis, "On the alleviation of the problem of local minima in back-propagation," *Nonlinear Anal.*, vol. 30, no. 7, pp. 4545–4550, 1997.

[24] J. Ji, S. Gao, J. Cheng, Z. Tang, and Y. Todo, "An approximate logic neuron model with a dendritic structure," *Neurocomputing*, vol. 173, pp. 1775–1783, Jan. 2016.

[25] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Biol. Cybern.*, vol. 59, nos. 4–5, pp. 257–263, 1988.

[26] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.

[27] S. Song, J. Ji, X. Chen, S. Gao, Z. Tang, and Y. Todo, "Adoption of an improved PSO to explore a compound multi-objective energy function in protein structure prediction," *Appl. Soft Comput.*, vol. 72, pp. 539–551, Nov. 2018.

[28] J. Ji, S. Song, C. Tang, S. Gao, Z. Tang, and Y. Todo, "An artificial bee colony algorithm search guided by scale-free networks," *Inf. Sci.*, vol. 473, pp. 142–165, Jan. 2019.

[29] F. Arce, E. Zamora, H. Sossa, and R. Barrón, "Differential evolution training algorithm for dendrite morphological neural networks," *Appl. Soft Comput.*, vol. 68, pp. 303–313, Jul. 2018.

[30] K. W. Chau, "Particle swarm optimization training algorithm for ANNS in stage prediction of Shing Mun river," *J. Hydrol.*, vol. 329, nos. 3–4, pp. 363–367, 2006.

[31] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Let a biogeography-based optimizer train your multi-layer perceptron," *Inf. Sci.*, vol. 269, pp. 188–209, Jun. 2014.

[32] L. Zhang and P. N. Suganthan, "A survey of randomized algorithms for training neural networks," *Inf. Sci.*, vol. 364, pp. 146–155, 2016.

[33] J. Ji, S. Song, Y. Tang, S. Gao, Z. Tang, and Y. Todo, "Approximate logic neuron model trained by states of matter search algorithm," *Knowl.-Based Syst.*, vol. 163, pp. 120–130, Jan. 2018.

[34] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Inf. Sci.*, vol. 291, pp. 43–60, Jan. 2015.

[35] F. Gabbiani, H. G. Krapp, C. Koch, and G. Laurent, "Multiplicative computation in a visual neuron sensitive to looming," *Nature*, vol. 420, no. 6913, pp. 320–324, 2002.

[36] N. Karayiannis and A. N. Venetsanopoulos, *Artificial Neural Networks Learning Algorithms, Performance Evaluation, and Applications*, vol. 209. New York, NY, USA: Springer, 2013.

[37] H. Sossa and E. Guevara, "Efficient training for dendrite morphological neural networks," *Neurocomputing*, vol. 131, pp. 132–142, May 2014.

[38] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–614, Feb. 2018.

[39] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proc. IEEE Int. Conf. neural Netw.*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[40] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evol. Comput.*, vol. 25, no. 1, pp. 1–54, 2016.

[41] J. Fisher, "The opening of milkbottles by birds," *Brit. Birds*, vol. 42, no. 11, pp. 347–357, 1949.

[42] T. R. Zentall, "Imitation in animals: Evidence, function, and mechanisms," *Cybern. Syst.*, vol. 32, nos. 1–2, pp. 53–96, 2001.

[43] A. Asuncion and D. Newman, "UCI machine learning repository," 2007.

[44] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[45] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognit.*, vol. 44, no. 8, pp. 1761–1776, 2011.

[46] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[47] W. Ye, W. Feng, and S. Fan, "A novel multi-swarm particle swarm optimization with dynamic learning strategy," *Appl. Soft Comput.*, vol. 61, pp. 832–843, Dec. 2017.

[48] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 4, pp. 656–667, Apr. 1994.

[49] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[50] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artif. Intell. Rev.*, vol. 12, no. 4, pp. 265–319, 1998.

[51] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution—An updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, Apr. 2016.

[52] J. Derrac, S. García, D. Molina, F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.

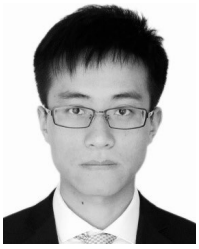
[53] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[54] R. Lior, *Data Mining With Decision Trees: Theory and Applications*, vol. 81. Singapore: World Scientific, 2014.
[55] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[56] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Mach. Learn.*, vol. 29, nos. 2–3, pp. 103–130, 1997.
[57] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Proc. IEEE signal Process. Soc. Workshop Neural Netw. Signal*, Aug. 1999, pp. 41–48.
[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

**SHUANGYU SONG** received the B.E. degree from Huaiyin Normal University, Huai'an, China, in 2013, and the M.S. degree from the University of Toyama, Toyama, Japan, in 2017, where he is currently pursuing the Ph.D. degree. His current research interests include artificial neural networks, evolutionary computation, and swarm intelligent algorithms.

**XINGQIAN CHEN** received the B.S. degree from the Nanjing University of Science and Technology, Nanjing, China, in 2014, and the M.S. degree from the University of Toyama, Toyama, Japan, in 2019, where she is currently pursuing the Ph.D. degree. Her main research interests include computational intelligence, artificial neural networks, and bioinformatics.

**CHENG TANG** received the B.E. degree from the Jinling Institute of Technology, Nanjing, China, in 2017. He is currently pursuing the M.S. degree with the University of Toyama, Toyama, Japan. His main research interests include evolutionary computation and swarm intelligent algorithms.



**SHUANGBAO SONG** received the B.E. degree from Soochow University, Suzhou, China, in 2013, and the M.S. degree from the University of Toyama, Toyama, Japan, in 2017, where he is currently pursuing the Ph.D. degree. His main research interests include computational intelligence, bioinformatics, and artificial neural networks.

**ZHENG TANG** received the B.S. degree from Zhejiang University, Zhejiang, China, in 1982, and the M.S. and D.E. degrees from Tsinghua University, Beijing, China, in 1984 and 1988, respectively, where he was an Instructor with the Institute of Microelectronics, from 1988 to 1989. From 1990 to 1999, he was an Associate Professor with the Department of Electrical and Electronic Engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined Toyama University, Toyama, Japan, where he is currently a Professor with the Department of Intellectual Information Systems. His current research interests include intellectual information technology, neural networks, and optimizations.

**YUKI TODO** received the B.S. degree from Zhejiang University, Zhejiang, China, in 1983, the M.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 1986, and the D.E. degree from Kanazawa University, Kanazawa, Japan, in 2005. From 1987 to 1989, she was an Assistant Professor with the Institute of Microelectronics, Shanghai Jiaotong University, Shanghai, China. From 1989 to 1990, she was a Research Student with Nagoya University, Nagoya, Japan. From 1990 to 2000, she was a Senior Engineer with Sanwa Newtech Inc., Miyazaki, Japan. From 2000 to 2011, she was with the Tateyama Systems Institute, Toyama, Japan. In 2012, she joined Kanazawa University, where she is currently an Associate Professor with the School of Electrical and Computer Engineering. Her current research interests include multiple-valued logic, neural networks, and optimization.