ISSN 2616-5775 Vol. 6, Issue 5: 113-122, DOI: 10.25236/AJCIS.2023.060516

Approximate Logical Dendritic Neuron Model Based on Selection Operator Improved Differential Evolution Algorithm

Ning Zhang*, **Yubao Yan**

College of Computer and Artificial Intelligence, Changzhou University, Changzhou, Jiangsu, 213100, China

**Corresponding author: zhangning19951218@126.com*

Abstract: *In order to solve the problem that the classification effect of approximate logic dendritic neuron model is largely limited by the training effect of learning algorithm. The differential evolution algorithm of the population evolution algorithm is selected as the training algorithm of the model. Differential evolution algorithm is a branch of population evolution algorithm, which has the advantages of good robustness and easy implementation. But it also has the disadvantage that it is easy to fall into the stagnation of evolution. In order to solve this key problem, several differential evolution algorithms are investigated. Finally, we noticed that a differential evolution algorithm improved by selection operator can better solve the problem of algorithm evolution stagnation. This paper studies the training of logic dendritic neuron model using the differential evolution algorithm improved by this selection operator. In order to evaluate the performance of the algorithm training model, four representative data sets were used for experiments. When comparing the classification effect, particle swarm optimization algorithm, traditional differential evolution algorithm and genetic algorithm are selected as the comparison experiment.*

Keywords: *artificial neural network, approximate logic dendritic neuron model, population evolution algorithm, differential evolution algorithm, classification*

1. Introduction

Since the invention of the artificial neural network, as the research on it has deepened, the artificial neural network has been widely used to solve complex problems such as classification, function approximation, and prediction [1]. In 1943, Warren MuCulloch and Walter Pitts creatively proposed a computational neuron model that mimics the structure and function of biological neurons, which is called the M-P model[2]. Since this model was invented, it has been widely used as a computing unit for neural network computing. The disadvantage is that the oversimplified generalization of the biological neuron structure by the M-P model makes the model unable to effectively solve the nonlinear classification problem[3].

In recent years, academic research on biological neurology and biophysics has become increasingly in-depth. The role of dendritic neurons in neural computing has been increasingly noticed[4]. Individual neurons with specific dendritic structures can perform specific brain functions, including memory, learning, and other cognitive behaviors[5]. Due to a large number of studies on the structure of neurons, Koch et al. proposed a delta-like cell model with a dendrite structure based on the above studies. The model was used to analyze the interaction between excitatory and inhibitory inputs to neurons[6,7]. This model was validated by several biological experiments[8]. However, the dendrite structure is highly dynamic, and Koch's model cannot simulate the plasticity of this neuron structure. The biological neural structure in reality can adopt different dendrite structures when dealing with different affairs. In this aspect, the model fails to simulate the real biological neuron structure well[9].

In terms of simulating the plasticity of the dendritic neural structure, researchers have proposed a variety of plastic structures based on the pyramidal nerve cells of the brain [10-13]. Legenstein and Maass proposed a single neuron model with a dendritic structure. Such a model is designed considering spike-time-dependent plasticity and branch-strength reinforcement learning principles[14]. The researchers verified the feasibility of the model with a mathematical proof and a simple feature binding problem. But the disadvantage of this model is that it cannot solve nonlinear classification problems[15].

Recently, researchers proposed an approximate logic neuron model (Approximate Logic Dendritic Neuron Model, ALDNM). At present, the ALDNM model has been applied to solve some real-world problems, such as computer-aided diagnosis and financial forecasting[16,17]. In order to prove the effectiveness of the learning mechanism of the ALNM model, researchers proposed an unsupervised learning rule training model to solve the two-dimensional selection problem. Experimental results prove the effectiveness of the model[18]. In addition, the model has neural pruning functions that other models do not have, including dendrite pruning and synapse pruning[19].

Differential evolution algorithm is a simple and efficient global optimization algorithm. It was first proposed by Storn and Price in 1995[20]. Since the differential evolution algorithm was invented, it is suitable for solving various optimization problems. These optimization problems include continuous optimization, discrete optimization, constrained optimization, and unconstrained optimization[21-24]. Although the performance of the differential evolution algorithm is superior, its accuracy is still greatly affected by mutation operators, crossover operators, parameter control and selection operators. There are a large number of studies on differential evolution algorithms that have made a lot of contributions to the improvement of algorithm performance in various aspects.

Most studies on differential evolution algorithms focus on the improvement of mutation operators and parameter control. Zhiqiang Zeng et al. proposed an improved differential evolution algorithm for selection operator optimization[25]. The proposer compared 58 benchmark functions with 6 differential evolution algorithms, and the simulation results showed that the proposed selection operator significantly improved the performance of the differential evolution algorithm.

This article will use the differential evolution algorithm optimized by the selection operator mentioned above to train the ALDNM model, and use the CMSC, Glass, Wine and wisconsin breast cancer data sets to test the classification effect of the algorithm. The classification results were compared with the ALDNM model trained by particle swarm optimization, traditional differential evolution algorithm and genetic algorithm.

2. Approximate Logic Dendritic Neuron Model

In this section, the article will introduce the structure and principle of the Approximate Logical Neuron Model (ALDNM). Inspired by the biological neuron model and dendrite mechanism, the researchers proposed a neuron model called ALDNM. ALDNM consists of four layers: synaptic layer, dendritic layer, membranous layer, and cell body. Figure 1 shows the structure of ALDNM, where n represents the number of inputs and m represents the number of branches, thus, the total number of synapses is m*n. The synaptic layer receives the incoming signal from the previous neuron and processes the sigmoid function of the received signal. A logical AND operation is then performed between the synapses on each branch. All dendritic branches of the dendritic layer are connected to the membranous layer, and the interrelationship between these branches can be regarded as a logical OR operation. Finally, the cell body performs nonlinear computations on the signal from the previous layer. After outputting the result, the model completes a calculation as a whole.

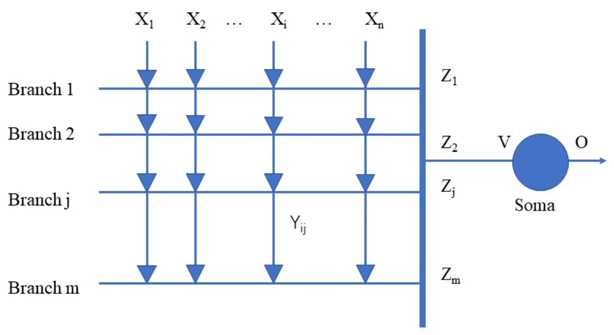

Figure 1: Structure of ALNM model

2.1. Synaptic Layer

This layer represents the organization of synaptic connections from the previous neuron to the next neuron, and its signal transmission is feed-forward. Whether a synapse is excited or inhibited depends on changes in specific ionic synaptic potentials. The sigmoid function was used to process 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 as follows:

$$Y_{i,m} = (1 + e^{-k(w_{i,m} * x_i - q_{i,m})})^{-1} \tag{1}$$

The k in the formula is a user-defined parameter, which is set to 5 in this experiment. x_i represents the signal input to the synaptic layer, and these data will be normalized to [0,1]. Y_(i,m) represents the output of the i-th local synapse on the m-th branch of the dendrite. w_(i,m) and q_(i,m) represent the connection parameters that need to be adjusted during the learning process. Figure 2 is a schematic diagram of the synaptic layer structure.

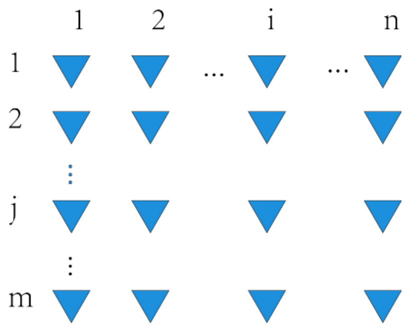

Figure 2: Synaptic layer

2.2. Dendritic layer

There are multiplication operations in the process of neuron processing neural information [26]. For each branch, the dendritic layer performs a multiplication operation on the synaptic connections. Since synaptic signaling at the dendritic layer is almost binary, logical AND operations can be used instead. The output of the jth branch is shown in formula (2). Figure 3 is a schematic diagram of the dendritic layer structure.

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

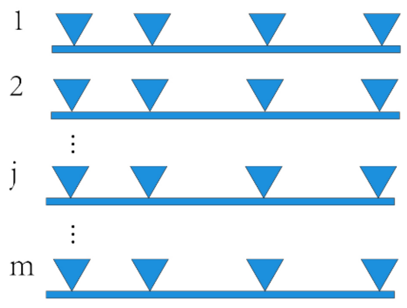

Figure 3: Dendritic layer

2.3. Membrane layer

Each branch of the dendritic layer is connected to the membrane layer. This layer performs a sum operation on the results of all branches. Because the signals are binary, this operation can be replaced by a logical OR operation. Then, the results of the film layer are sent to the final layer. The output formula of the film layer is shown in formula (3). Figure 4 is a schematic diagram of the film layer structure.

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

Figure 4: Membrane layer

2.4. Soma body

The soma body is the last part of the neuron model. It performs non-linear computations on the results it receives. If the input signal exceeds a predefined threshold, the neuron will be activated. The calculation formula of this layer is shown in formula (4).

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

In that formula, γ represents the threshold constant. V represents the output result of the membrane layer. c_{soma} represents a fixed parameter. O indicates the final output of the cell body layer. Figure 5 is a schematic diagram of the cell body structure.

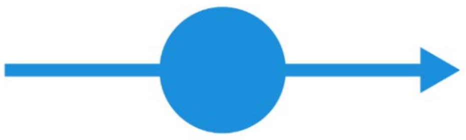

Figure 5: Soma body

In order to explain the structure of ALDNM more clearly, an example is provided here to represent the ALDNM classification process. Figure 6 shows the operation steps and basic components of each layer. The operation steps of ALDNM are described below.

Step 1: At the synaptic layer, m branches intersect with the input (n-dimensional), forming m*n synaptic connections (▼). Each synaptic connection produces the output of formula (1). Then, m*n outputs are transmitted to the dendritic layer.

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

Step 3: The membrane layer processes the output of m branches through formula (3). The result is then sent to the soma body.

Step 4: In the soma, use the sigmoid function to process the output of the membrane layer to obtain the classification result.

The number of inputs and outputs per layer of ALDNM is summarized in Table 1. The input of each layer comes from the output of the previous layer.

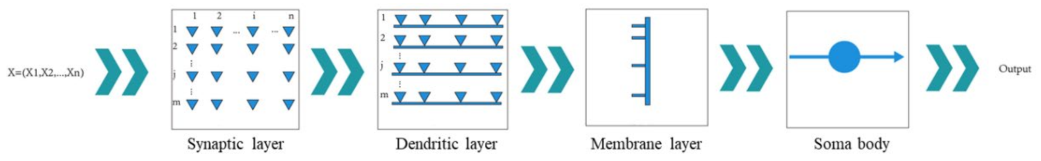

Figure 6: A classification progress example of ALDNM

Table 1: The number of inputs and outputs of each layer of ALDNM

Name of layer	Nunmber of input	Number of output
Synaptic layer	n	m*n
Dendritic layer	m*n	m
Membrane layer	m	1
Soma	1	1

3. Selection operator improved differential evolution algorithm

Differential evolution algorithm is a population-based optimization algorithm. A population contains multiple individuals, and each individual is represented by a vector. The differential evolution algorithm continuously improves individuals in the iterative process, and each individual evolves continuously in the iterative process to finally reach the best solution. The differential evolution algorithm mainly includes three operation links: mutation operator, crossover operator and selection operator. The mutation operation selects multiple individuals from the population, and generates mutant individuals through the mutation operator. Crossover applies a crossover operator to mutation vectors and parent vectors to generate trial vectors. The selection operator selects a vector from the test vector and the parent vector to survive to the next generation.

At present, the selection operator used by most differential evolution algorithms is the greedy selection operator. It has been found that the greedy selection operator is effective most of the time, but when the individual stagnates (stagnation can be understood as the individual cannot be improved after a certain number of iterations), it is likely that the individual will fall into a local optimum. If greedy selection is still used at this time, it will be difficult for the individual to jump out of the local optimum. Therefore, a new selection operator is needed to help the individual jump out of the local optimum when it is stagnant.

As a global optimization algorithm, the differential evolution algorithm is mostly improved on the mutation operator and parameter control. The improvement method proposed by Zhiqiang Zeng et al. is to improve the selection operator of the general differential evolution algorithm.

The operation logic of this selection operator is: when the individual is not in a stagnant state, the proposed selection operator operates in the same way as the traditional selection operator, that is, the optimal vector is selected from the test vector and the parent vector to survive to the next generation; when the individual is at a standstill, the algorithm selects from the three previously selected candidate vectors. The first candidate vector is the optimal vector selected from the trial vectors among all the discarded parent vectors, the second candidate vector is the suboptimal vector selected from the trial vectors among all the discarded parent vectors, and the third candidate vectors are randomly selected from all successfully updated solutions[25].

This improved algorithm prevents the evolution process of the algorithm from falling into an evolutionary stagnation state, and enables the algorithm to obtain more optimized results.

4. Training methods

As we all know, training algorithms have a large impact on the capabilities of neural network models. As a relatively novel population algorithm, the differential evolution algorithm has the advantages of good robustness and convenient use, but at the same time it has the defect that it is easy to fall into evolutionary stagnation. The differential evolution algorithm for selection operator optimization mentioned above can effectively prevent the algorithm from falling into an evolutionary stagnation state by setting candidate variables in advance.

The differential evolution algorithm for selection operator optimization first initializes m individuals of the group P (t) in a random manner, where t represents the current iteration. Then, during each iteration, a mutation vector is generated by a mutation operation. Once the mutation vector is obtained, use the crossover operator on the parent vector and the mutation vector to obtain the trial vector. After the trial vector is obtained, and the algorithm is not stagnant, a better vector can be selected from the parent vector and the test vector through the selection operator, so that the vector survives to the next generation. When the algorithm detects that the number of generations that the individual stops evolving reaches a certain value, the algorithm considers that the current evolution has stagnated. At this time, the optimal vector is selected from the three candidate vectors prepared before and the algorithm is re-optimized for iteration.

To train the ALDNM using the selection operator optimization differential evolution algorithm in the experiment, the first step is the representation of the problem. The goal of training a model with an algorithm is to find the most appropriate parameter values to achieve the highest classification accuracy. Since each synaptic connection has two parameters (w and q), the number of parameters that need to be adjusted by the algorithm can be expressed as formula (5).

$$\mathrm{N} = 2 * \mathrm{n} * \mathrm{m} \tag{5}$$

Where n represents the number of inputs and m represents the number of branches. Therefore, each individual of the differential evolution algorithm optimized by the selection operator can be expressed as a vector as shown in formula (6).

$$\begin{aligned} \mathrm{X} &= (\mathrm{W}, \mathrm{Q}) \\ &= (w_{1,1}, w_{1,2}, \cdots, w_{n,m}, q_{1,1}, q_{1,2}, \cdots, q_{n,m}) \end{aligned} \tag{6}$$

The direct purpose of training is to make the gap between the actual output and the ideal output smaller. Therefore, the mean square error of the dendritic neuron model can be directly used as the fitness function of the selection operator optimization differential evolution algorithm. It can be expressed as formula (7).

$$\text{fitness} = \frac{1}{2S}\sum_{s=1}^{S}(T_s - O_s)^2 \tag{7}$$

In formula (7), T_s and O_s represent ideal output and actual output respectively. S represents the number of training samples.

In order to express the training process more clearly, we show the flow chart of the entire training in Figure 7. The flowchart consists of three main parts: the training set, the differential evolution algorithm for selection operator optimization, and ALDNM. The role of these three parts in the experiment process is as follows.

Training set: Provides training samples for ALDNM.

Differential evolution algorithm for operator optimization: Iteratively update individual values to obtain the best fitness function value. The individual selection operator optimized by the differential evolution algorithm is represented by formula (6). The fitness function of the individual is calculated by formula (7). When the number of iterations reaches the preset target value, the algorithm stops iterating, and at this time returns the parameter values obtained from the optimal individual and outputs them to ALDNM.

ALDNM: Calculate the actual output of all samples, as shown in Figure 6. Then, the value calculated in formula (7) is returned to the differential evolution algorithm for selection operator optimization as the fitness function value of the individual.

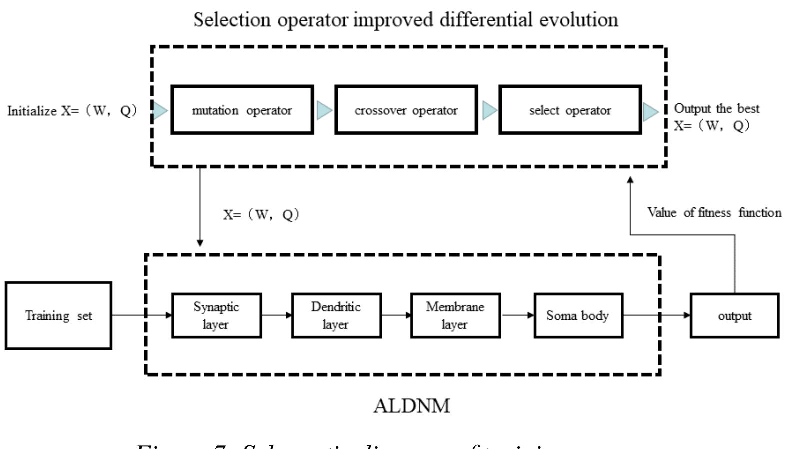

Figure 7: Schematic diagram of training process

5. Experimental Research

This section presents the experiments to verify the performance. The algorithms in this research are all written in python. The experiment runs on a Windows system with a core i7 processor and 16G memory.

In this experiment, four classic data sets were selected to verify the classification performance of the dendritic neuron model optimized by the selection operator. The four datasets are Wine, Climate modelsimulation crashes (CMSC), Wisconsin diagnostic breast cancer (WDBC), and Glass. These four dataset classification problems are all from the UCI Machine Learning Repository. The details of the four datasets are summarized in Table 2.

The wine dataset includes chemical composition analyzes of red wines produced in specific regions of Italy. The specific origin of red wine can be analyzed from the chemical composition contained in the wine. This dataset contains 178 records, each with 13 attributes. CMSC is used to predict the value of climate model simulation results for a given climate model parameter. It contains a total of 540 parameter value combinations, each with 18 values. All attribute values for this dataset are scaled on the interval [0, 1]. Simulation results are represented by 0 (success) and 1 (failure). The WDBC dataset was provided by Dr. William H. Wolberg, University of Wisconsin, et al. for breast cancer diagnosis. These features are obtained by computing digitized images of breast masses. The dataset includes records from 569 samples, each with 30 feature items. The diagnosis results are divided into benign (B) and malignant (M) categories. The Glass dataset includes various parameters of different types of glass. Depending on the parameters, the type of glass can be deduced. The Glass dataset includes 214 sample records, and each sample contains 9 attributes.

Since the proposed ALDNM is a binary classifier, the above data sets are treated as a binary classification problem in the data processing stage in order to be used for ALDNM testing. The topic of ALDNM solving multi-classification problems is worthy of more exploration in the future.

Table 2: Details of the four datasets used in the experiments

Name of dataset	Num.of classes	Num.of featrues	Num.of samples
Wine	2	13	178
CMSC	2	18	540
WDBC	2	30	569
Glass	2	9	214

According to a large number of researchers' previous studies, the constant parameters c, c_{soma} and γ of ALDNM are set to 5, 5 and 0.5, respectively. Each dataset is randomly divided into two parts, one for training and one for testing. The ratio of the two subsets is 50%:50%. Performing 30 operations on each dataset formed 30 pair subsets for experiments. Since each random operation to divide the dataset is performed independently, the 30 pairs of subsets are different. After running the proposed model, 30 experimental results were obtained. To satisfy the input of ALDNM, all eigenvalues are normalized in the interval [0, 1]. The number of iterations for all training algorithms is set to 100 generations.

6. Comparison with other heuristic algorithms

Table 3: Parameter settings of three comparison algorithms

Name of algorithms	Name of parameters	Value of parameters
	ω	0.5
Pso	c_1	1.5
	c_2	1.5
	N	50
	p_c	0.9
	Crossover type	Single point
Ga	p_m	0.1
	Selection	Roulette wheel
	N	50
	N_e	10
	CR	0.9
De	F	0.5
	N	50

This section discusses the training effect of the differential evolution algorithm for operator optimization and other three typical heuristic algorithms on the dendritic neuron model. The three algorithms are Particle Swarm Optimization (PSO) [27], Genetic Algorithm (GA) [28] and traditional Differential Evolution (DE) [29]. The parameter settings of these three algorithms are shown in Table 3.

Table 4 lists the classification results of the four heuristic optimization algorithms. Obviously, except that the classification accuracy rate of the CMSC data set is not much different from the other three algorithms, the classification accuracy rate of the operator optimization algorithm on the other three data sets has reached the highest.

Table 4: Classification accuracy of four algorithms

	CMSC	Glass	Wine	WDBC
Trail algorithm	0.914815∓0.02963	**0.920561∓0.051402**	**0.988764∓0.011236**	**0.955715∓0.018571**
pso	0.920371∓0.035185	0.901869∓0.060748	0.926967∓0.073033	0.952858∓0.021428
de	0.912963∓0.024074	0.915888∓0.046729	0.988764∓0.011236	0.954286∓0.022857
ga	**0.925926∓0.02963**	0.920561∓0.042056	0.915731∓0.08427	0.955715∓0.021429

To further investigate the differences between these algorithms, Figure 8 lists the average convergence curves for the four benchmark problems. Since the experiment used MSE as the fitness function, so the curve of fitness function is the curve of MSE. The comparison of the convergence curves on four datasets is shown in Figure 8.

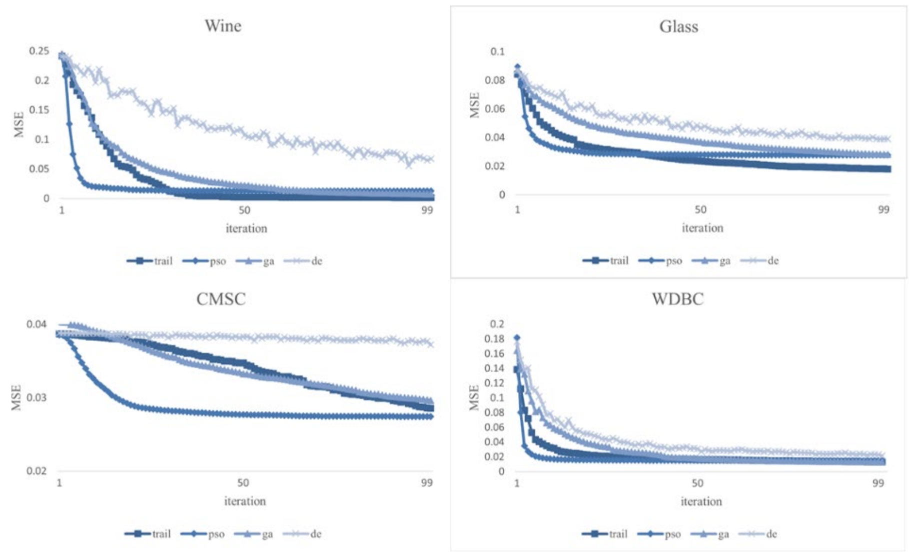

Figure 8: Comparison of the MSE curve of the differential evolution algorithm training with other three heuristic algorithms

It is not difficult to see that when the number of iterations reaches about 20 times, the MSE of the differential evolution algorithm training with operator optimization is close to the minimum value, while the training results of other heuristic algorithms are still not stable when the number of iterations reaches the maximum value set in the experiment down. Therefore, it is obvious that the convergence speed of the differential evolution algorithm training using the selection operator optimization on the wine data set is significantly faster than that of other heuristic algorithm training.

A comparison of the convergence curves on the glass dataset is shown in Figure 8. The convergence curve of the objective function trained by the differential evolution algorithm for selection operator optimization reaches a relatively stable state roughly after 50 iterations. Although the convergence curve of the objective function of particle swarm training has reached a stable state before 50 generations, the minimum value of the convergence is far from the effect achieved by the differential evolution algorithm of selective operator optimization. Therefore, it can be seen that in the same number of iterations, the differential evolution algorithm with operator selection optimization achieves a smaller minimum value than the backpropagation algorithm, so it can also be concluded that the differential evolution algorithm trained with operator selection optimization. The convergence speed is significantly faster than the convergence speed of backpropagation algorithm training.

Since the objective function value of the CMSC data set can reach less than 0.05 at the beginning, the change range of the convergence curve on this data set is not very obvious. Although the change range of the convergence curve with the increase of the number of iterations is not obvious, it can still be seen that whether it is the initial function value or the function value after the preset number of training iterations, the training of the differential evolution algorithm with operator optimization is selected. The effect is better than the other three heuristic algorithms. Therefore, it can be seen that in the same number of iterations, the differential evolution algorithm with operator selection optimization achieves a smaller minimum value than the backpropagation algorithm, so it can also be concluded that the differential evolution algorithm trained with operator selection optimization The convergence speed is significantly faster than the other three traditional heuristic algorithm training convergence speed.

It is not difficult to see that when the number of iterations reaches about 20 times, the fitness function trained by the operator-optimized differential evolution algorithm is close to the minimum value, while the training results of the other three heuristic algorithms have already reached the minimum value when the number of iterations reaches 50 generations. The convergence curve of the objective function reaches a stable value, but when the number of iterations reaches the preset maximum value, the minimum value achieved by the differential evolution algorithm for operator optimization is better than the minimum value trained by the other three heuristic algorithms. Therefore, it can still be concluded that the convergence speed of the differential evolution algorithm training using the selection operator optimization is significantly faster than that of the other three heuristic algorithm training.

The excellent performance of the differential evolution algorithm for selecting operator optimization in training ALDNM can be explained as follows. The three candidate vectors reserved by the differential evolution algorithm for selection operator optimization during normal operation ensure that the algorithm can be restarted when it is stagnant. This mechanism effectively prevents the optimization process from stagnating.

7. Conclusion

The neuron model proposed based on the structure of dendritic neuron cells has been applied to solve many real-world problems. In this field of research, a new model called the approximate logistic dendritic neuron model was proposed. The model consists of four layers, the synaptic layer, the dendritic layer, the membrane layer, and the soma body. In this study, ALDNM is trained by using the differential evolution algorithm optimized by selection operator, which is excellent in solving high-dimensional problems. The experimental results show that the differential evolution algorithm for operator optimization can achieve better performance, and it is superior to three typical heuristic optimization algorithms, PSO, GA, and DE, in terms of classification accuracy and the minimum value of the convergence curve.

In future research, we intend to adopt ALDNM to solve more complex classification problems to verify its classification efficiency. In addition, the development of ALDNM to solve multi-classification problems is also a field worthy of exploration.

References

[1] Yegnanarayana B. Artificial neural networks. PHI Learning Pvt. Ltd., 2009.
[2] McCulloch W S, Pitts W. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 1943, 5: 115-133.
[3] Rosenblatt F. The perceptron - a perceiving and recognizing automaton. 1957.
[4] London M, Häusser M. Dendritic computation. Annu. Rev. Neurosci., 2005, 28: 503-532.
[5] Niell C M, Meyer M P, Smith S J. In vivo imaging of synapse formation on a growing dendritic arbor. Nature neuroscience, 2004, 7(3): 254-260.
[6] Koch C, Poggio T, Torre V. Retinal ganglion cells: a functional interpretation of dendritic morphology. Philosophical Transactions of the Royal Society of London, 1982, 298(1090):227.
[7] Koch C, Poggio T, Torre V. Nonlinear interactions in a dendritic tree: localization, timing, and role in information processing. Proceedings of the National Academy of Sciences, 1983, 80(9):2799-2802.
[8] W.R. Taylor, S. He, W.R. Levick, D.I. Vaney, Dendritic computation of direction selectivity by retinal ganglion cells, Science (5488) (2000) 2347–2350.
[9] Segev I. Sound grounds for computing dendrites. Nature, 1998, 393(6682):207-8.
[10] H.C. Dringenberg, B. Hamze, A. Wilson, W. Speechley, M.-C. Kuo, Heterosynaptic facilitation of in vivo thalamocortical long-term potentiation in the adult rat visual cortex by acetylcholine, Cerebral Cortex17 (4) (2006) 839–848.
[11] A. Losonczy, J.K. Makara, J.C. Magee, Compartmentalized dendritic plasticity and input feature

storage in neurons,.Nature 452 (7186) (2008) 436.
[12] Sjostrom P J, Rancz E A, Roth A, et al. Dendritic excitability and synaptic plasticity. Physiological Reviews, 2008, 88(2):769-840.
[13] Makara J K, Losonczy A, Wen Q, et al. Experience-dependent compartmentalized dendritic plasticity in rat hippocampal CA1 pyramidal neurons. Nature Neuroscience, 2009, 12(12):1485-7.
[14] Legenstein R, Maass W. Branch-specific plasticity enables self-organization of nonlinear computation in single neurons. Journal of Neuroscience, 2011, 31(30):10787-10802.
[15] Costa Rui. One Cell to Rule Them All, and in the Dendrites Bind Them. Frontiers in Synaptic Neuroscience, 2011, 3:5.
[16] Sha Z, Lin H U, Todo Y, et al. 14-A Breast Cancer Classifier Using a Neuron Model with Dendritic Nonlinearity. 2016. 1365–1376.
[17] Zhou T, Gao S, Wang J, et al. Financial time series prediction using a dendritic neuron model. Knowledge-Based Systems, 2016, 105(aug.):214-224.
[18] Todo Y, Tamura H, Yamashita K, et al. Unsupervised learnable neuron model with nonlinear interaction on dendrites. Neural Networks, 2014, 60:96-103.
[19] Ji J, Gao S, Cheng J, et al. An approximate logic neuron model with a dendritic structure. Neurocomputing, 2015, 173(P3):1775-1783.
[20] Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 1997, 11(4): 341.
[21] Liu Q, Du S, Wyk B, et al. Double-layer-clustering differential evolution multimodal optimization by speciation and self-adaptive strategies. Information Sciences, 2021, 545(1):465-486.
[22] Hameed A, Aboobaider B, Mutar M, et al. A new hybrid approach based on discrete differential evolution algorithm to enhancement solutions of quadratic assignment problem. International Journal of Industrial Engineering Computations, 2020, 11(1): 51-72.
[23] Xu B, Zhang H, Zhang M, et al. Differential evolution using cooperative ranking-based mutation operators for constrained optimization. Swarm and Evolutionary Computation, 2019, 49: 206-219.
[24] Cheng J, Pan Z, Liang H, et al. Differential evolution algorithm with fitness and diversity ranking-based mutation operator. Swarm and Evolutionary Computation, 2021, 61: 100816.
[25] Zeng Z, Zhang M, Chen T, et al. A new selection operator for differential evolution algorithm. Knowledge-Based Systems, 2021, 226:107150.
[26] Gabbiani F, Krapp H G, Koch C, et al. Multiplicative computation in a visual neuron sensitive to looming. Nature. vol. 420, no. 6913, pp. 320–324, 2002.
[27] Bonyadi M R, Michalewicz Z. Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. Evolutionary Computation, 2017, 25(1):1-54.
[28] Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems Man & Cybernetics, 2002, 24(4):656-667.
[29] Storn R, Price K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, 1997, 11(4):341-359.