A Novel Evolutionary Higher Order Neural Network for pattern Classification

Sibarama Panigrahi [1], Sreetoma Pandey [2], Ria Singh [3]
Department of Computer Science and Engineering, MITS, Rayagada, Odisha, India

Abstract

Over the last few decades pattern classification, pattern matching and mathematical function approximation were predominately performed using various neural networks (NNs). Compared to traditional NNs, higher order neural networks (HONNs) have several unique characteristics, including: 1) stronger approximation property; 2) faster convergence; 3) greater storage capacity; and 4) higher fault tolerance capability. In this paper, a novel evolutionary HONN especially the Pi-Sigma network (PSN) is used for pattern classification. The PSN is trained using a strictly greedy chemical reaction optimization algorithm. In contrast to basic CRO algorithms, in this CRO algorithm the reactant size (population size) is remained fixed throughout all the iteration, which makes it easier to implement. The performance of proposed methodology for pattern classification is evaluated through three well-known real world classification problems from UCI machine learning data library. The results obtained from the proposed method for classification is compared with results obtained by applying the two most popular variants of differential evolution algorithm (DE/rand/1/bin and DE/best/1/bin). It is observed that the proposed method provides better classification accuracy than that of other methods.

1. Introduction

Classification is the process of assigning an object into a predefined group or class of objects based on the number of attributes of that object. It is one of the most frequently encountered decision making tasks of human activity. Traditionally, statistical procedures such as discriminant analysis were widely used to perform pattern classification. However, the effectiveness of these methods depends to a large extent on the various assumptions or conditions under which the models are developed and thus, these procedures work well only when the underlying assumptions are satisfied. Users must have a good knowledge of both data properties and model capabilities before the models can be successfully applied. Considering the above pitfalls several classifiers using various data mining and computational intelligence methods have been developed.

Out of various types of classifiers neural network based classifiers are most popular and predominantly found in the literature [1]. Despite of advantageous features of HONN models over traditional NN models, only few papers were found in the literature for pattern classification using HONN models [2-4]. Therefore, in this paper the class of HONNs and in particular Pi-Sigma Networks (PSNs) has been studied. The PSNs were introduced by Shin and Ghosh [4]. The PSNs have addressed several difficult tasks such as zeroing polynomials [5] and polynomial factorization [6] more effectively than traditional feed-forward neural networks (FFNNs). Moreover, PSN employ less number of weights than other HONNs, but still manage to incorporate the capability of first order HONN indirectly. Therefore, in this paper Pi-Sigma Network is considered for pattern classification.

The rest of this paper is organized as follows. Section-2 briefly describes the background related to architecture and mathematical model of PSN; chemical reaction optimization; and differential evolution. The method used for classification using an evolutionary PSN is explained in Section-3. Experimental results are presented in section-4. And finally conclusion are described in Section-5.

2. Preliminaries

2.1. Pi-Sigma Network

Pi–Sigma Network (PSN) is a special type of feed forward higher order neural network that calculates the product of sum of the input components and passes it to a nonlinear function. The network architecture of PSN is shown in Figure 1. It consists of a single hidden layer with summing units and an output layer of product units. The weights connecting the input and hidden layer are adapted during the training process, while those connecting the neurons of the hidden layer to the output layer are fixed to one and they are not trainable. Such a network topology with only one layer of trainable weights drastically reduces the training time [2, 7]. Moreover, the product units of PSN gives higher order capabilities which increase its computational power. This is because; the product units enable to expand the input space into higher dimensional space, thus easily separates nonlinearly separable classes to linear separable. Thus, PSN provides nonlinear

decision boundaries offering a better classification capability than the linear neuron (Guler and Sahin, 1994). In addition, Shin and Ghosh (1991) also suggested that PSNs not only offers better classification over a broad class of problems but also requires less memory and need at least two orders of magnitude less number of computations as compared to MLP for similar performance level.

Consider a PSN with NOIN (number of input neurons), NOHN (number of hidden neurons) and one output neuron. The number of hidden neurons in the hidden layer defines the order of a PSN. For a NOHNth order PSN the number of trainable weights is NOIN × NOHN considering each summing unit is associated with NOIN weights. The output of the PSN is computed by making product of the output of NOHN hidden units and passing it to a nonlinear function, which is defined as follows:

$$Y = \sigma\left(\prod_{j=1}^{NOHN} h_j\right)$$

Where σ is a nonlinear transfer function and h_j is the output of j^{th} hidden unit which is computed by making sum of the products of each input (x_i) with the corresponding weight (w_{ij}) between i^{th} input and j^{th} hidden unit. The output of hidden unit is computed as follows:

$$h_j = \sum_{i=1}^{NOIN} (w_{ij} x_i)$$

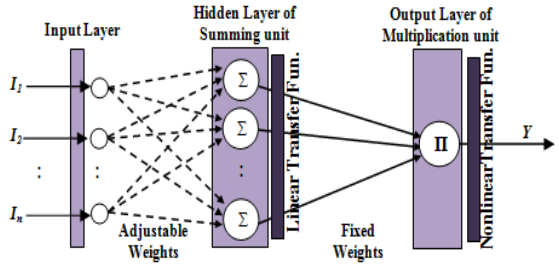

Figure 1: Architecture of a Typical Pi-Sigma Network

3. Proposed Method

The efficiency of any supervised neural network depends on the algorithm used for its training. The objective of any supervised Pi-Sigma network training is to minimize the error between the approximation by the HONN and the target output. For this the optimal weight set of a HONN must be obtained. The optimal weight set of a HONN can be obtained by using either gradient or evolutionary learning algorithms. The objective function of HONN training is going to be a multimodal search problem, since it depends on number of parameters.

Algorithm 1 (CRO-HONNT)

Set the iteration-counter i=0

/*Randomly Initialize the ReacNum of Reactants from a uniform distribution [U;L]: $P^i=\{R_1^i, R_2^i, R_3^i \ldots, R_{ReacNum}^i\}$, with $R_j^i = \{W_{j,1}^i, \ldots\ldots, W_{j,D}^i\}$ for j=1,2,3..... ReacNum, D=length of each Reactant (NOIN×NOHN), $W_{j,k}^i = k^{th}$ atom of j^{th} reactant in i^{th} iteration representing a weight of PSN.

for j=1 to ReacNum
  Calculate the enthalpy e(R_j)
end of for
While (termination criteria is not satisfied) **do**
  begin
  for j=1 to ReacNum
    // perform all reaction over the reactants of P^i
    Get $rand_1$ randomly in an interval [0, 1]
    if $rand_1 \leq 0.5$
      Get $rand_2$ randomly in an interval [0, 1]
      if $rand_2 \leq 0.5$
        Decomposition (R_j);
      else
        Redox1(R_j)
      end of if
    else
      Get $rand_3$ randomly in an interval [0, 1]
      if $rand_3 \leq 0.33$
        Select the best reactant R_k($R_k \neq R_j$)
        Synthesis (R_j, R_k)
      else if $rand_3 \leq 0.66$
        Select another reactant R_k ($R_k \neq R_j$) randomly
        Displacement(R_j, R_k);
      else
        Select another reactant R_k($R_k \neq R_j$) randomly
        Redox2(R_j, R_k)
      end of if
    end of if
    Apply strictly greedy Reversible Reaction for increased enthalpy to update reactants
  end of for
  Set the iteration counter i=i+1
end of while
Use the reactant having best enthalpy as the optimal weight set of PSN and perform classification.

Therefore, the gradient based training algorithms often suffer from several shortcomings, including: 1) easily getting trapped to local minima; 2) have slow convergence properties; 3) training performance is sensitive to initial values of its parameters. Due to these disadvantages, research on different optimization techniques that are dedicated to HONN training is still required. There are many optimization techniques such as differential evolution (DE) [8], genetic algorithm (GA) [9], particle swarm optimization (PSO) [10], ant

colony optimization (ACO) [11], a bee colony optimization (BCO) [12], an evolutionary strategy (ES) [13], quantum inspired algorithms (QEA) [14], chemical reaction optimization (CRO) [15-17] etc. can be used for HONN training. Chemical reaction optimization (CRO) is a new optimization technique, inspired by the nature of chemical reactions. CRO has demonstrated excellent performance in solving many engineering problems such as the mining classification rules [18], quadratic assignment problem [15] etc. Therefore, in this paper a novel strictly greedy CRO algorithm has been used to train PSN and further used for classification. The algorithm is described as follows.

The proposed training algorithm operates in three phases: initialization phase, iteration phase and final phase. The initial phase assigns the value to initial parameters like termination criterion, length of reactants/molecules, ReacNum and generates initial reactants. The iteration phase simulates the reaction processes. The reactions may be monomolecular or bimolecular. For monomolecular reaction, Decomposition and Redox1 reactions are considered; and for bimolecular reactions three types of reactions such as: Synthesis, Displacement and Redox2 are considered. The reaction types are chosen considering both intensification and diversification. Moreover, a strictly greedy reversible reaction is used to update the reactants. All the reactions have been elaborated in the following subsequent subsections. In final phase the reactant having best enthalpy is used as the optimal solution (i.e. optimal weight set of a PSN). The pseudo-code of the proposed method is explained in Algorithm 1.

3.1 Reactant Encoding

A set of real numbers are used to represent one reactant, with each reactant corresponding to a weight set of the PSN. The length of a reactant depends on the number of input and hidden neurons of the PSN (i.e. NOIN×NOHN).

3.2 Enthalpy of A Reactant

Each reactant is associated with some enthalpy. As each reactant represents a weight set of the PSN, the mean square error (MSE) on the train set is considered as enthalpy. The lower the value of enthalpy the better the reactant is. The MSE is defined as follows:

$$MSE = \frac{\sum_{i=1}^{NOP} (Y_i - T_i)^2}{NOP}$$

Where Y_i and T_i are the output of PSN and target for i^{th} train pattern.

3.3 Chemical Reactions

3.3.1 Monomolecular reactions

In monomolecular reactions only one reactant takes part in the reaction and one product is produced by modifying one atom of the reactant. These reactions assist in intensification of the solution by making local search. In our algorithm monomolecular reactions are performed with a probability of 50%, there by glorifying the chances to obtain a better solution around the current solution. Two monomolecular reactions are considered such as: Decomposition and Redox1.

3.3.1.1 Decomposition Reaction

In this reaction a randomly selected atom of the reactant takes a value from the range [U, L]. Consider a reactant $R_j=\{W_{j,1},W_{j,2}\ldots\ldots,W_{j,D}\}$ with $W_{j,x}$ ($x \in [1,n]$) be an atom of the reactant-j. The pseudo-code of the decomposition reaction is described in Algorithm-2.

Algorithm 2 (Decomposition(R_j))
Input: A reactant R_j
Duplicate R_j to produce R_1
Select an atom x ($x \in$ [1, n]) randomly.
$W_{1,x}=L+ rand() \times (U-L)$
Where rand() is a random number generated randomly from a range [0, 1].
Output: A new reactant R_1

3.3.1.2 Redox1 Reaction

It is similar to decomposition reaction except that the rate of reaction(λ)used in this algorithm is obtained in a self adaptive manner. The pseudo-code is described in Algorithm-3.

Algorithm 3 (Redox1(R_j))
Input: A reactant R_j
Duplicate R_j to produce R_1
Select a point x ($x \in$ [1:n]) randomly
$W_{1,x} =L+ \lambda_t \times (U-L)$
Where $\lambda_{t+1}= 4\ \lambda_t\ (1- \lambda_t)$ With $\lambda_0 \in [0,1]-\{0, 0.25, 0.5, 0.75, 1.0\}$
Output: A new reactant R_1

3.3.2 Bimolecular reactions

Here two reactants $R_j=\{W_{j,1},\ldots\ldots,W_{j,D}\}$ and $R_k=\{W_{k,1},W_{k,2}\ \ldots\ldots,W_{k,D}\}$ will take part in the reaction. These reactions help in diversification of the solution by generating a new solution that is significantly different from the current solution. These reactions occur with a probability of 50%. Following types of bimolecular reactions are used.

3.3.2.1 Synthesis Reaction

In this reaction one reactant is produced due to reaction between a reactant and the best reactant of the iteration.

Algorithm 4 (Synthesis (R_j,R_k))

Input: Two reactants R_j, R_k

$R_1=R_J+\lambda\times(R_k-R_J)$

Where λ is a random number between [-0.25;1.25]

Output: A new reactant R_1

3.3.2.2 Displacement Reaction

Two solutions R_1 and R_2 are obtained from reaction between two reactants R_j and R_k.

Algorithm 5 (Displacement (R_j, R_k))

Input: Two reactants R_j, R_k

$R_1=\lambda_t\times R_j+\lambda_t\times(1-R_k)$

$R_2=\lambda_t\times R_k+\lambda_t\times(1-R_j)$

Where λ_t is initialized to a random number [0,1]and is updated in the following manner every time this reaction reoccurs (t=number of time the reaction occurs).

$\lambda_{t+1}=2.3(\lambda_t)^{2\sin(\Pi\lambda t)}$

Output: Two reactants R_1 and R_2

3.3.2.3 Redox2 Reaction

Algorithm 6 (Redox2 (R_j,R_k))

Input: Two reactants R_j, R_k

$R_1=R_j+\lambda\times(R_k-R_j)$

Where the rate of reaction λ=rand(0,1) random number between [0;1].

Output: A new reactant R_1

This reaction is similar to that of synthesis reaction, but here, the rate of reaction is obtained from a range of [0-1].

3.3.3 Reactant update

Every monomolecular or bimolecular reaction is followed by a strictly greedy reversible reaction to update the reactants. In the strictly greedy reversible reaction, for a monomolecular reaction the product produced replaces the worst reactant of the reactant set for better enthalpy; and for a bimolecular reaction if two products are produced, best product replace the worst reactant for better enthalpy. Thus the number of reactants of the population remains same throughout the reaction process. The pseudo-code of the strictly greedy reversible reaction is elaborated in algorithm 7.

Algorithm 7 (Strictly greedy Reversible Reaction ())

For Monomolecular Reactions

Let R_j under goes monomolecular reaction to produce R_1

If enthalpy(R_1)<enthalpy(worst(R)))

Replace R_{worst} by R_1

end of if

For Bimolecular Reactions

If R_j and R_k under goes reaction to produce R_1 and R_2

If enthalpy (R_1) < enthalpy(R_{worst})

Replace R_{worst} by R_1

end of if

end of if

4. Experimental Setup and Simulation Results

The simulations in this paper were carried out on a system with Intel ® core(TM) 2Duo E7500 CPU, 2.93 GHz with 2GB RAM and implemented using SCILAB5.4.1. All ANNs are trained using proposed CRO, DE/rand/1/bin and DE/best/1/bin with population size (reactant size) 50 and initial value of each chromosome (representing a ANN weight-set) is initialized to uniform distributed random values drawn from a range [-1, 1].

4.1. Performance Measure

Correct classification percentage is used as performance measure, which is computed as follows:

Correct classification

$$\text{Correct Classification}(\%)=\frac{\sum_{i=1}^{NOP}C_i}{NOP}$$

Where NOP is number of test patterns (NOP/2); C_i- the coefficient representing the correctness of the classification of the i^{th} testing pattern which is determined as follows:

$$C_i=\begin{cases}1, & \text{when } Y_i=1 \text{ and } T_i=1\\ 1, & \text{when } Y_i=-1 \text{ and } T_i=-1\\ 0, & \text{Otherwise}\end{cases}$$

Where Y_i and T_i are the output of PSN and target for i^{th} test pattern.

4.1. Data Sets and Simulation Results

For experimental analysis three binary classification problems (Sonar, Breast Cancer Wisconsin, Haberman's Survival) are considered from the well known UCI machine learning data library.

For the Sonar problem the task is to train a PSN to distinguish between sonar signals bounced off a metal cylinder (mine) and those bounced off a roughly

cylindrical rock. In this experiment the dataset contains 208 samples obtained by bouncing sonar signals off a metal cylinder and a rock at various angles and under various conditions. There exist 111 samples obtained from mines and 97 samples obtained from rocks. Each pattern consists of 60 real numbers in the range [0.0, 1.0]. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The trained PSNs have one unit in the middle layer with 60–1–1 architecture. For comparative performance analysis, the results obtained from 100 independent simulations using the proposed method (using proposed strictly greedy CRO algorithm), DE/rand/1/bin and DE/best/1/bin methods are shown in Table 1 where mean, St.Dev, Min and Max represents the mean, standard deviation, minimum and maximum classification accuracy of the 100 independent simulations.

Table 1 Classification Accuracy (%) for Sonar problem.

Method	Mean	St.Dev.	Min	Max
Proposed Method	**77.88**	**3.64**	**56.67**	**84.13**
DE/rand	73.81	4.24	52.88	81.73
DE/best	73.35	4.34	53.36	80.77

For the breast cancer wisconsin problem the task is to train a PSN to discriminate between benign and malignant based on ten attributes. In the original dataset 2 represents benign (367 patterns) and 4 represents malignant (332 patterns) which are converted to -1 and 1 respectively for our simulation. 100 independent simulations were carried out and the mean, standard deviation, min and max values of the obtained results are shown in Table 2. Note that PSNs with 10-1-1 architectures are trained with 50% of total samples and tested with other 50%.

Table 2 Classification Accuracy (%) for breast cancer wisconsin problem.

Method	Mean	St.Dev.	Min	Max
Proposed Method	**75.15**	**6.75**	**52.44**	**85.67**
DE/rand	71.46	8.06	52.15	83.38
DE/best	73.59	7.94	49.86	84.53

Haverman's survival dataset contains cases from study conducted on the survival of patients who had undergone surgery for breast cancer. For the Haberman's survival problem the task is to train a PSN to predict whether the patient will survive more than 5 years or not based on three attributes. The dataset contains 306 patterns with three attributes each and a class level -1 for less than 5 years and 1 for more than five years of survival. 100 independent simulations were carried out and the mean, standard deviation, min and max values of the obtained results are shown in Table 3. Note that PSNs with 3-1-1 architectures are trained with 50% of total samples and tested with other 50%.

Table 3 Classification Accuracy (%) for Haverman's survival problem.

Method	Mean	St.Dev.	Min	Max
Proposed Method	**70.66**	**4.51**	**43.14**	**73.20**
DE/rand	68.99	9.71	30.72	72.55
DE/best	68.25	9.36	28.76	**73.20**

It can be observed from the results that the proposed methodology provides better classification accuracy for the three problems considered.

5. Conclusion

In this paper, we have studied evolutionary HONN models especially; the Pi–Sigma network for pattern classification. A novel strictly greedy chemical reaction optimization algorithm is developed for its training. For comparative performance analysis of the proposed method three real world binary classification problems are considered. It is found that the strictly greedy CRO algorithm along with PSN provides better classification accuracy than the two most popular variants of differential evolution algorithm i.e. DE/rand/1/bin and DE/best/1/bin. In future more efficient training algorithm for PSN can be developed to improve the classification accuracy.

6. References

[1] G. P. Zhang, "Neural Networks for Classification: A Survey", *IEEE Transaction on Systems, Man, and Cybernetics- Part C: Applications and Reviews*, vol. 30, no. 3, pp. 451-462, 2000.

[2] Y. Shin and J. Ghosh, "Efficient higher-order neural networks for classification and function approximation", *In: International Journal on Neural Systems*, vol. 3, pp.323–350, 1992.

[3] M. G. Epitropakis, V. P. Plagianakos, M. N. Vrahatis, "Hardware-friendly Higher-Order Neural Network Training using Distributed Evolutionary Algorithms", *Applied Soft Computing*, vol. 10, pp. 398-408, 2010.

[4] Y. Shin, J. Ghosh, "The pi–sigma network: An efficient higher-order neural network for pattern classification and function approximation", *International Joint Conference on Neural Networks*, 1991.

[5] D. S. Huang, H. H. S. Ip, K. C. K. Law and Z. Chi, "Zeroing polynomials using modified constrained neural network approach", *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 721–732, 2005.
[6] S. Perantonis, N. Ampazis, S. Varoufakis and G. Antoniou, "Constrained learning in neural networks: Application to stable factorization of 2-d polynomials", *Neural Processing Letter*, vol.7, no. 1, pp. 5–14, 1998.
[7] Y. Shin and J. Ghosh, "Realization of Boolean functions using binary pi-sigma networks", *in: C. H. Dagli, S. R. T. Kumara, Y. C. Shin (Eds.), Intelligent Engineering Systems through Artificial Neural Networks, ASME Press*, pp. 205–210, 1991.
[8] R. Storn and K.Price, "Differential evolution- A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, vol. 11, no.4, pp. 341-359, 1997.
[9] D. Goldberg, "Genetic Algorithms in Search", *Optimization and Machine Learning. Reading*, MA:Addison-Wesley, 1989.
[10] J. Kennedy, R. C.Eberhart and Y.Shi, "Swarm intelligence", *San Francisco, CA:Morgan Kaufmann*, 2001.
[11] K. Socha and M. Doringo, "Ant colony optimization for continuous domains", Europian Journal of Operation Research, vol. 185, no. 3, pp. 1155-1173, 2008.
[12] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim and M. Zaidi, "The bees algorithm- A novel tool for complex optimization problems", *in IPROMS Oxford, U.K.: Elsevier*, 2006.
[13] H.G. Beyer and H.P. Schwefel, "Evolutionary Strategies: A Comprehensive introduction", *Nat. Comput.*, vol. 1, no. 1, pp. 3-52, 2002.
[14] K. H. Han and J.H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization", *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 580–593,2002.
[15] A. Y. S. Lam and V. O. K. Li, "Chemical-Reaction-inspired metaheuristic for optimization", *IEEE Transactionson on Evolutionary Computation*, vol. 14, no.3, pp. 381–399, 2010.
[16] A.Y.S. Lam, "Real-Coded Chemical Reaction Optimization", *IEEE Transaction on Evolutionary Computation*, vol. 16, no. 3,pp. 339-353, 2012.
[17] B. Alatas, "ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization", *Expert Systems with Applications*, vol. 38, pp. 13170–13180, 2011.
[18] B. Altas, "A novel chemistry based metaheuristic optimization method for mining of classification rules", *Expert Systems with Applications*, vol. 39, pp. 11080-11088, 2012.
[19] T. K. Truong, K. Li and Y. Xu, "Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem", *Applied Soft Computing*, vol. 13, pp. 1774-1880, 2013.
[20] J.J.Q. Yu, A.Y.S. Lam and V.O.K. Li, "Evolutionary Artificial Neural Network based on chemical reaction optimization", *in: IEEE Congress on Evolutionary Computation (CEC)*, pp. 2083–2090, 2011.