A Modified Differential Evolution Algorithm trained Pi-Sigma Neural Network for Pattern Classification

Sibarama Panigrahi, Ashok Kumar Bhoi, Yasobanta Karali

Abstract— In this paper a modified differential evolution (DE) algorithm trained Pi-Sigma network (PSN) is used for classification. The used DE algorithm is a modification of traditional DE/rand/1/bin algorithm and novel mutation as well as crossover strategies are followed considering both exploration and exploitation. 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) and Chemical Reaction Optimization (CRO) algorithm. It is observed that the proposed method provides better classification accuracy than that of other methods.

Index Terms—Differential Evolution, Higher Order Neural Network, Pi-Sigma Network, Classification.

I. INTRODUCTION

Classification is the process of assigning objects in a collection to one of the predefined target categories or classes. The goal of classification is to accurately predict the categorical value of an object based on its number of observed attributes (pattern). Many problems in engineering, business, science, industry, and medicine can be treated as classification problems. The classification task is a two-step process such as: Classifier Building and Classifier Testing. In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier finds relationships between the values of the predictors and the values of the target by analyzing or "learning from" a training set made up of attributes and their associated class labels. These relationships are summarized in a classifier. In the second step the obtained classifier is used for classification on a different data set in which the class assignments are unknown, to predict the class of the patterns.

Traditionally, statistical procedures were widely used for pattern classification. However, the effectiveness of these methods depends on various assumptions under which the models are developed and prior knowledge regarding both data properties and model capabilities. Considering the above pitfalls several classifiers using various data mining and computational intelligence methods like rule induction, fuzzy rule induction, decision trees, neural networks (NNs) have been developed.

Out of various types of classifiers neural network based classifiers were predominantly found in the literature [1]. However, Compared to traditional NNs, higher order neural networks (HONNs) have several unique characteristics, including: 1) stronger approximation with faster convergence property; 2) greater storage capacity; and 3) higher fault tolerance capability. However, the major drawback of HONNs is the exponential growth in number of weights with the increase in order of the network. But, PSNs are special type of feed forward HONN model which have the capability of higher order neural networks and at the same time uses less number of weights. Despite of advantageous features of PSNs over traditional NN models and other HONN models, only few papers were found in the literature for pattern classification using HONN models [2]–[5]. Therefore, in this paper the class of 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 [6] and polynomial factorization [7] more effectively than traditional feed-forward neural networks (FFNNs).

The rest of this paper is organized as follows. Section-2 briefly describes the background related to architecture and mathematical model of PSN; 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 conclusions are described in Section-5.

II. PRELIMINARIES

A. Pi-Sigma Network

Pi–Sigma Network (PSN) is a feed forward higher order neural network consisting of a single hidden layer. The hidden layer has summing units where as the output layer has product units. The weights connecting the input and hidden layer are obtained during the training process and weights connecting the neurons of the hidden layer to the output layer are fixed to one. It has a linear activation function at hidden layer and nonlinear transfer function at output layer. Thus the PSN calculates the product of sum of the inputs and corresponding weights and pass it through a nonlinear function. Such a network topology with only one layer of trainable weights drastically reduces the training time [2], [8]. The network architecture of PSN is shown in Figure 1. Additionally, the product units of PSN gives higher order capabilities by expanding 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

Manuscript Received November, 2013.

Sibarama Panigrahi, Department of Computer Science and Engineering, MIRC Lab, MITS Engineering College, Rayagada, Odisha, India.
Ashok Kumar Bhoi, Department of Computer Science and Engineering, VSSUT, Burla, Sambalpur, India.
Yasobanta Karali, Department of Computer Science and Engineering, VSSUT, Burla, Sambalpur, India.

the linear neuron.

Consider a PSN with NOIN (number of inputs), 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 activation function and h_j is the output of jth hidden unit which is computed by summing the products of each input (x_i) with the corresponding weight (w_{ij}) between ith input and jth hidden unit. The output of hidden unit is computed as follows:

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

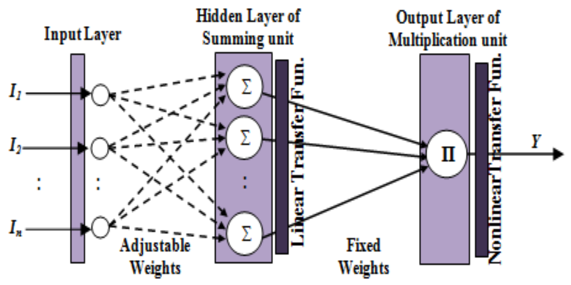

Figure 1: Architecture of a Typical Pi-Sigma Network.

B. Differential Evolution

The differential evolution (DE) algorithm is a simple and efficient stochastic direct search method for global optimization. It was introduced several years ago (1997) [9]. Since then it has been upgraded intensively in recent years [10]. It has various advantages such as: ability to find global minimum of a non-differentiable, nonlinear and multimodal function, parallelizability and good convergence properties. Compared to most other EAs, DE is much simpler to implement. Although particle swarm optimization (PSO) is also very easy to code, the performance of DE and its variants outperforms the PSO variants over a wide variety of problems as has been indicated by studies like [11]-[12] and the CEC competition series. The two most popular variants of DE are DE/best/1/bin and DE/rand/1/bin. The major difference between these two lies in the selection of base vector for perturbation. In DE/best/1/bin the base vector is the best solution of the current population whereas in DE/rand/1/bin the base vector is selected randomly. The conventions used above is DE/a/b/c, where 'DE' stands for 'differential evolution', 'a' represents the base vector to be perturbed, 'b' represents number of difference vectors used for perturbation of 'a' and 'c' represents the type of crossover used (bin: binary, exp: exponential). Interested reader may go through [9]-[10] to have a detail description regarding DE algorithm and its variants. Every differential evolution algorithm operates in following steps:

Step 1: Problem and algorithm parameter initialization.

Step 2: Initialize the initial population and calculate the fitness of each chromosome/individual.

Step 3: Apply Mutation operator to generate the mutant vector.

Step 4: Apply crossover (binary or exponential) between the target vector and mutant vector to generate the trial vector.

Step 5: Perform Selection between trial vector and target vector.

Step 6: If termination criteria is satisfied go to step-7 otherwise go to step-3.

Step 7: Use the best individual as the solution of the problem.

III. METHODOLOGY

Being a supervised neural network, the objective of Pi-Sigma network training is to minimize the error between the approximation by the PSN and the target output. For this the optimal weight set of a PSN must be obtained. The optimal weight set of a PSN can be obtained by using either gradient or evolutionary learning algorithms.

The objective of PSN is going to be a multimodal search problem, the optimization techniques using evolutionary methods is a better choice. There are many optimization techniques such as differential evolution (DE) [9], genetic algorithm (GA) [13], particle swarm optimization (PSO) [14], ant colony optimization (ACO) [15], a bee colony optimization (BCO) [16], an evolutionary strategy (ES) [17], quantum inspired algorithms (QEA) [18], chemical reaction optimization (CRO) [19]-[20] etc. can be used for PSN training. In this paper a modified differential evolution algorithm is been used.

The method used in this paper is explained in algorithm-1. An attempt has been made to combine the advantage of DE/rand/1 (diversification property) and DE/best/1 (intensification property) by overcoming the shortcomings of both the algorithms. Taking these facts into consideration to overcome the limitation of slow convergence but reliable DE/rand/1 we use an explorative yet greedy variant of DE/rand/1/bin mutation strategy with novel mutation and crossover strategies. The crossover probability (Cr) is generated randomly (within a range [0-1], and regenerated if It is beyond the range) from a cauchy distribution with location parameter=0.6 and scale parameter 0.1.

Algorithm 1

```
Set the gen-counter g=0
/*Randomly Initialize the population of PopSize individuals: $P_g=\{C^1_g, C^2_g, C^3_g, \ldots, C^{PopSize}_g\}$, with $C^i_g=\{W^{i,1}_g, \ldots, W^{i,D}_g\}$ for i=1,2,3.....NP, D=length of each chromosome, $W^{i,k}_g$=kth gene of ith individual in gth generation representing a weight of PSN.
Evaluate the fitness of each individual
While (termination criteria is not satisfied)
    %for each individual chromosome ($C^i_g$) in the population
    for i=1 to PopSize
        Select three individuals ($I_1$, $I_2$, $I_3$) and such that $I_1 \neq I_2 \neq I_3 \neq i$
        Sort the three select individuals
        Set $r_1$=best individual out of $I_1$, $I_2$, $I_3$
            $r_2$= second best individual out of $I_1$, $I_2$, $I_3$
            $r_3$= worst individual out of $I_1$, $I_2$, $I_3$
        Generate a scale factors $F_i$ with mean
      % Mutation Step
      % Generate scale factor $F_i$ =gaussianrnd(0.5,0.1), is a random number generated randomly from gaussian distribution with mean 0.5 and standard deviation 0.1.
      $MV_g = C^{r1}_g + F_i*(C^{r2}_g - C^{r3}_g)$
      % Generate Cross over Probability
        $Cr_i$ =cauchyrnd(0.6,0.1), is a random number generated randomly from Cauchy distribution with location parameter 0.7 and scale parameter 0.1. It is regenerated if the random number falls put of the range [0-1].
      for x=1 to D
        if rand(0,1)< $Cr_i$
          $TV^{k,x}_g = MV^{k,x}_g$
        else
          $TV^{k,x}_g = W^{i,x}_g$
        end of if
      end of for
      % Selection Step
      % Fitness of a chromosome is -1×RMSE on train set
    if fitness(TV) > fitness($C_g^i$)
      $C^i_{g+1}$= TV
    else
      $C^i_{g+1} = C^i_g$
    end of if
  end of for
  Set the generation counter g=g+1
end of while
Use the reactant having best fitness (least RMSE) as the optimal weight set of PSN and perform classification.
```

For crossover probability instead of normal or uniform distribution, Cauchy distribution is considered because it diversifies the solution more as compared to traditional normal or uniform distribution. The scale parameter (F) is generated randomly from a Gaussian distribution with mean=0.5 and standard deviation=0.1. If the number generated is out of the range [0-2] it is regenerated.

IV. 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 MATLAB. 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].

A. Performance Measure

Classification percentage is used as performance measure, which is computed as follows:

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

Where NOP is number of test patterns (NOP/2); C_i- the coefficient representing the correctness of the classification of the ith 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 ith test pattern.

B. Datasets and Simulation Results

For experimental analysis three binary classification problems such as: Sonar, Breast Cancer Wisconsin and Haberman's Survival which are widely used classification problems.

Table I. Classification Accuracy (%) for Sonar problem.

Method	Mean	St.Dev.	Min	Max
Proposed	**79.24**	**3.24**	**60.08**	**85.57**
CRO	77.88	3.64	56.67	84.13
DE/rand/1	73.81	4.24	52.88	81.73
DE/best/1	73.35	4.34	53.36	80.77

The Sonar problem consists of 208 samples/patterns (111 samples obtained from mines and 97 samples obtained from rocks.) with each pattern consists of 60 attributes representing energy with in a particular frequency bands. The task is to train a PSN to make a distinction between sonar signals bounced off a metal cylinder (mine) and those bounced off a roughly cylindrical rock based on the 60 observed attributes. 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, CRO algorithm [5], 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 on test set (50% of the total patterns) of the 100 independent simulations.

For the breast cancer wisconsin problem the task is to train a PSN to distinguish between benign and malignant based on ten attributes. The dataset consists of 699 patterns 367 patterns and 332 patterns with class labels benign and malignant respectively. 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 II. Classification Accuracy (%) for breast cancer wisconsin problem.

Method	Mean	St.Dev.	Min	Max
Proposed	**76.42**	**5.28**	**52.44**	**87.10**
CRO	75.15	6.75	52.44	85.67
DE/rand/1	71.46	8.06	52.15	83.38
DE/best/1	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. The task is to predict whether the patient will survive more than 5 years or not based on three attributes. The dataset contains 306 patterns. 100 independent simulations were carried out and the mean, standard deviation, min and max values of correct classification on test set (50% of total samples) are shown in Table 3.

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

Method	Mean	St.Dev.	Min	Max
Proposed	**72.92**	**2.32**	**56.21**	**73.20**
CRO	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

V. CONCLUSION

In this paper, we have studied differential evolution based Pi–Sigma network for pattern classification. A modified differential evolution algorithm trained PSN is used for classification. For comparative performance analysis of the proposed method three real world binary classification problems are considered. It is found that the proposed method provides better classification accuracy than chemical reaction optimization and the two most popular variants of differential evolution algorithm i.e. DE/rand/1/bin and DE/best/1/bin trained PSN for all the three datasets considered.

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, 2000, pp. 451-462.
2. Y. Shin and J. Ghosh, "Efficient higher-order neural networks for classification and function approximation", In: International Journal on Neural Systems, vol. 3, 1992, pp.323–350.
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, 2010, pp. 398-408.
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. S. Panigrahi, S. Pandey, R. Singh, "A Novel Evolutionary Higher Order Neural Network for Pattern Classification", International Journal of Engineering Research and Technology, vol. 2, no. 9, 2013, pp.2561-2566.
6. 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, 2005, pp. 721–732.
7. 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, 1998, pp. 5–14.
8. 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, 1991, pp. 205–210.
9. 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, 1997, pp. 341-359.
10. S. Das, P. N. Suganthanam, "Differential Evolution: A Survey of the state-of-the-Art", IEEE Transaction on Evolutionary Computation, vol. 15, no.1, 2011, pp. 4-31.
11. S. Das, A. Abraham, U. K. Chakraborty, A. Konar, "Differential evolution using a neighbourhood based mutation operator", IEEE Transaction on Evolutionary Computation, vol. 13, no. 3, 2009, pp. 526-553.
12. S. Rahnamayan, H. R. Tizhoosh, M. M. A. Salama, "Opposition based differential evolution", IEEE Transaction on Evolutionary Computation, vol. 12, no.1, 2008, pp. 64-79.
13. D. Goldberg, "Genetic Algorithms in Search", Optimization and Machine Learning. Reading, MA:Addison-Wesley, 1989.
14. J. Kennedy, R. C.Eberhart and Y.Shi, "Swarm intelligence", San Francisco, CA:Morgan Kaufmann, 2001.
15. K. Socha and M. Doringo, "Ant colony optimization for continuous domains", Europian Journal of Operation Research, vol. 185, no. 3, 2008, pp. 1155-1173.
16. 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.
17. H.G. Beyer and H.P. Schwefel, "Evolutionary Strategies: A Comprehensive introduction", Nat. Comput., vol. 1, no. 1, 2002, pp. 3-52.
18. K. H. Han and J.H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization", IEEE Transactions on Evolutionary Computation, vol. 6, 2002, pp. 580–593.
19. A. Y. S. Lam and V. O. K. Li, "Chemical-Reaction-inspired metaheuristic for optimization", IEEE Transactionson on Evolutionary Computation, vol. 14, no.3, 2010, pp. 381–399.
20. A.Y.S. Lam, "Real-Coded Chemical Reaction Optimization", IEEE Transaction on Evolutionary Computation, vol. 16, no. 3, 2012,pp. 339-353.