Chapter 9
An Extended Approach of a Two-Stage Evolutionary Algorithm in Artificial Neural Networks for Multiclassification Tasks

Antonio J. Tallón-Ballesteros[1], César Hervás-Martínez[2], and Pedro A. Gutiérrez[2]

[1] Department of Languages and Computer Systems,
University of Seville,
Reina Mercedes Avenue, Seville, 41012 Spain
[2] Department of Computer Science and Numerical Analysis,
University of Córdoba,
Campus of Rabanales, Albert Einstein Building, Córdoba, 14071 Spain
atallon@us.es

Abstract. This chapter considers a recent algorithm to add broader diversity at the beginning of the evolutionary process and extends it to sigmoidal neural networks. A simultaneous evolution of architectures and weights is performed with a two-stage evolutionary algorithm. The methodology operates with two initial populations, each one containing individuals with different topologies which are evolved for a small number of generations, selecting the half best individuals from each population and combining them to constitute a single population. At this point, the whole evolutionary cycle is applied to the new population. This idea was previously proposed by us for product unit neural networks, and we now extend to sigmoidal neural networks. The experimentation has been carried out on twelve data sets from the UCI repository and two complex real-world problems which differ in their number of instances, features and classes. The results have been contrasted with nonparametric statistical tests and show that our proposal significantly improves the test accuracy of the models with respect to the obtained ones with a standard methodology based on a single population. Moreover, the new proposal is much more efficient than other methods developed previously by us.

Keywords: Artificial neural networks, Sigmoidal units, Product units, Evolutionary algorithms, Classification, Population diversity.

1 Introduction

The diversity issue is very important to avoid a premature convergence of evolutionary algorithms (EAs) and to reach solutions with good quality. At the beginning of the algorithm is suitable a diverse population and a more condensed one at the end of the search [1]. In this way, the algorithm combines the two key ideas exploration and exploitation. The matter of generating various populations is related

I. Jordanov and L.C. Jain (Eds.): Innovations in Intelligent Machines -3, SCI 442, pp. 139–153.
springerlink.com

with the diversity throughout the evolutionary process and has been also discussed previously (for instance in [2]). The number of populations may vary and there is not a common accepted value. A common view of the evolutionary cycle is that diversity enhances the performance of a population by providing more chances for evolution. A homogeneous population offers no advantage for improvement as the entire population is focused in a particular portion of the search space. High diversity does not imply better genetic algorithm performance; this is closely related to the question of exploration versus exploitation, but enforcing diversity in the early phases of evolution ensures a broad exploration of the search space [3].

Briefly, the main approach considered in this chapter diversifies the architecture of the neural network (NN) for classification problems at the beginning of the evolutionary process. Our previous methodology [4], which operates with evolutionary artificial neural networks (EANNs) based on product units [5] in a two-stage EA (TSEA) is extended in the current chapter in order to consider EANNs with sigmoidal units. Our objective is to compare the improved methodology -based on two stages- to the standard one -composed of a single stage with a population- employing sigmoidal units in both cases and also to the previous TSEA methodology that have been employed to date by us for product unit neural networks (PUNNs). A computational cost analysis is performed to determine the efficiency of the standard and improved methodologies that have been applied first time both for sigmoidal neural networks in the current chapter. Also, an efficiency report between product and sigmoidal units applying the improved methodology is presented.

The chapter is organized as follows: Sect. 2 describes some concepts about evolutionary sigmoidal and product unit neural networks, the base EA and TSEA; Sect. 3 introduces our proposal; Sect. 4 details the experimentation; then Sect. 5 shows and analyzes the results obtained; finally, Sect. 6 states the concluding remarks.

2 Methodology

2.1 Evolutionary Artificial Neural Networks Based on Sigmoidal and Product Units

EANNs offer a platform to optimize network performance and architecture simultaneously. Miller et al. [6] proposed that evolutionary computation was a very good candidate to search the space of architectures. Since then, many methods have been developed to evolve artificial neural networks (ANNs), for instance, [7, 8]. Next, we describe EANNs based on sigmoidal and product units. The methodology employed here uses an EA as a tool for simultaneous learning the architecture and weights of the ANN. Other authors have also considered this idea [9, 10]. In previous works we have trained PUNNs [4, 11], but in the current chapter we experiment, moreover, with sigmoidal units.

Fig. 1 shows the scheme of two ANN models for a bi-classification problem with sigmoidal (at part a)) and product units (at part b)). Each one is a *k:m:1* three-layer architecture, that is, k nodes in the input layer, m ones and a bias one in the hidden layer and one node in the output layer.

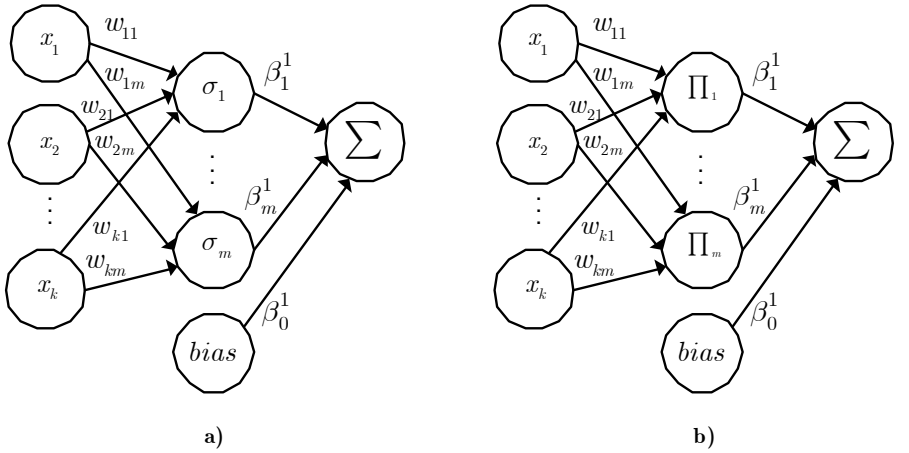

Fig. 1. Scheme of two ANN models with sigmoidal (a) and product (b) units for a bi-classification problem

The transfer function of each node in the hidden and output layers is the identity function. Thus, the functional model obtained by each of the nodes in the output layer with *J* classes is given by Eq. 1.

$$f(x_1, x_2, ..., x_k) = \beta_0^l + \sum_{j=1}^{m} \beta_j^l B_j(\boldsymbol{x}, w_j) \qquad l = 1, 2, ..., J \tag{1}$$

where B_j follows different expressions depending on the type of unit:

- Product unit:

$$B_j(\boldsymbol{x}, w_j) = \prod_{i=1}^{k} x_i^{w_{ji}} \tag{2}$$

- Sigmoidal unit:

$$B_j(\boldsymbol{x}, w_j) = \frac{1}{1 + \exp(-w_{j0} - \sum_{i=1}^{k} w_{ji} x_i)} \tag{3}$$

2.2 Two-Stage Evolutionary Algorithm

We have used a base EA to design the structure and learn the weights of ANN as in [11]. The followed base algorithm in the current chapter, that has also several common points with the EA described in [12], is exactly equal to those described in our previous work [4] for classification tasks. However, it has been generalized to any type of ANNs. We have followed the same guidelines given in our previous work; a detailed explanation can be read in Sect. 2.2 of [4].

```
Program: Two-Stage Evolutionary Algorithm
Data: Training set
Input parameters: gen, neu
Output: Best ANN model
                                        // First Stage
t ← 0
                                        // Population P1
P1(t) ← {ind1, …, ind10000}        // Individuals of P1 have neu nodes in the hidden layer
f1 (P1(t) {ind1, …, ind10000}) ← fitness (P1(t) {ind1, …, ind10000}) // Calculate fitness
P1(t) ← P1(t) {ind1, …, ind10000}     // Sort individuals
P1(t) ← P1(t) {ind1,… ind1000}                  // Retain the 1000 best ones
                                        // Population P2
P2(t) ← {ind1, …, ind10000}     // Individuals of P2 have neu+1 nodes in the hidden layer
f2 (P2(t) {ind1, …, ind10000}) ← fitness (P2(t) {ind1, …, ind10000}) // Calculate fitness
P2(t) ← P2(t) {ind1, …, ind10000}     // Sort individuals
P2(t) ← P2(t) {ind1,… ind1000}                  // Retain the 1000 best ones
for each Pi                  // Evolution of populations P1 and P2 until 0.1*gen generations
  current_generation ← 0
  t ← 0
    while current_generation < 0.1*gen not met do
        Pi(t) {ind901,…ind1000} ← Pi(t) {ind1, …, ind100} // Best 10% replace the worst 10%
        Pi(t+1) ← Pi(t) {ind1, …, ind900}
        Pi(t+1) ← pm (Pi(t+1) {ind1, …, ind90})      // Parametric mutation (10% Pi (t+1))
        Pi(t+1) ← sm (Pi(t+1) {ind91, …, ind900})     // Structural mutation (90% Pi (t+1))
        fi(Pi(t+1) {ind1, .. ind900}) ← fitness (Pi(t+1) {ind1, …, ind900})         // Evaluate
        Pi(t+1) ← Pi(t+1) (ind1, …, ind900) ⋃ Pi(t){ind901, …, ind1000}
        Pi(t+1) ← Pi(t+1){ind1, …, ind1000}                  // Sort individuals
        current_generation ← current_generation + 1
        t ← t + 1
    end while
end for
P(0) ← P1{ind1, …, ind500} ⋃ P2{ind1, …, ind500}   // Individuals of P has [neu, neu+1]
                                                    // nodes in the hidden layer
P(0) ← P(0) {ind1, …, ind1000}                  // Sort individuals by fitness: indi > indi+1
                                        // Second Stage
                                    // Input: gen, neu+1
t ← 0
while stop criterion not met do                        // main loop
      P(t) {ind901,…ind1000} ← P(t) {ind1, …, ind100} // Best 10% replace the worst 10%
      P(t+1) ← P(t) {ind1, …, ind900}
      P(t+1) ← pm (P(t+1) {ind1, …, ind90})      // Parametric mutation (10% P(t+1))
      P(t+1) ← sm (P(t+1) {ind91, …, ind900})     // Structural mutation (90% P(t+1))
      f(P(t+1) {ind1, .. ind900}) ← fitness (P(t+1) {ind1, …, ind900})         // Evaluate
      P(t+1) ← P(t+1) (ind1, …, ind900) ⋃ P(t) {ind901, …, ind1000}
      P(t+1) ← P(t+1){ind1, …, ind1000}            // Sort individuals
      last_generation ← t
      t ← t+1
end while
return best (P(last_generation) {ind1})
```

Fig. 2. Pseudo-code of the TSEA for classification

A specialization of the previous algorithm called TSEA has been introduced in [4] for product units. Basically, the two-stage EA operates with two initial populations, each of one containing individuals with different topologies, evolving them for a small number of generations, selecting the best individuals from each population in the same proportion and combining them to constitute a single population. At this point, the whole evolutionary cycle is applied to the new population. The pseudo-code of the TSEA appears in Fig. 2. The individuals are subjected to the operations of replication and mutation, thus the algorithm falls into the class of evolutionary programming. We do not use the crossover operator due to the permutation problem [7].

3 Proposal Description

The current chapter presents an extended methodology called TSEASig (Two-Stage Evolutionary Algorithm for neural networks with *Sig*moidal units) that is derived from TSEA [4]. The number of neurons in the input layer is equal to the number of variables in the problem; a hidden layer with a number of nodes that depends on the data set to be classified; and the number of nodes in the output layer equal to the number of classes minus one because a softmax-type probabilistic approach has been used. The first stage consists of creating two populations, each one with individuals that present different maximum number of nodes, *neu* and *neu+1*, in the hidden layer, evolving them with equal settings of the remaining parameters of the EA for a small number of generations, *0.1*gen*, selecting the half best individuals from each population and unifying them to constitute a single population. In the second stage, the full evolutionary process is applied to the population. The initial short training improves random individuals and let to explore possible promising areas in two directions, since there are two different populations. After that, individuals with different topologies coexist and the more adapted ones will remain. The parameters are defined as *gen,* the maximum number of generations; and *neu,* the maximum number of nodes in the hidden layer. In the recently cited work, we proposed a TSEA for EANNs with product units. TSEASig is an extension of TSEA that operates with NNs based on sigmoidal units. Fig. 3 depicts the structure of TSEASig.

EDD (experimental design distribution) methodology has been launched by us in [13]. It distributes some parameters of the network topology or of the base EA over some computing nodes. An initial configuration, called base configuration, is defined and it is modified with a new value for one parameter in each of the nodes. The idea is to run the base EA with different configurations. Up to date, EDD has been applied with NNs based on product units. Now, we try out it first time with sigmoidal units and the resulting methodology is named EDDSig, another original contribution of this chapter.

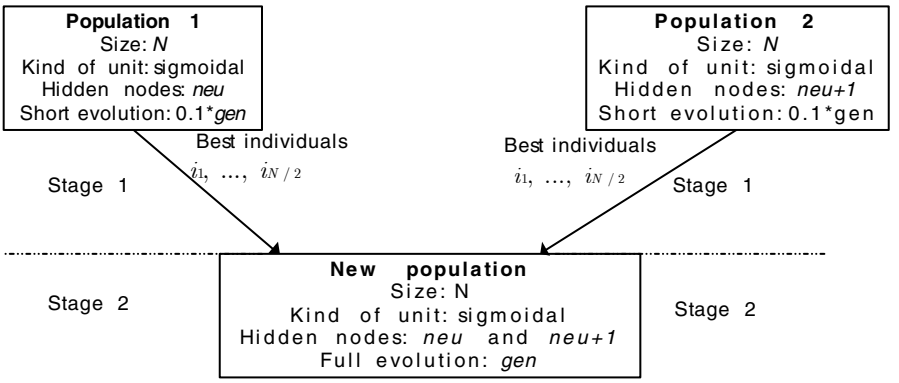

Fig. 3. TSEASig structure

Table 1 presents the description of the TSEA, EDDSig and TSEASig configurations. The *neu* and *gen* and parameters take the values indicated as input to the program. α_2 is associated with the residual of the updating expression of the output-layer weights and the initial value is fixed. For EDDSig, we have considered two configurations that can be compared to the equivalent one of TSEASig. Related to TSEA, we have taken into account the configuration 1*. It is noticeable that TSEA works with PUNNs and the remaining methodologies with sigmoidal units.

Table 1. Description of the TSEA, EDDSig and TSEASig configurations

Methodology	Configuration	Num. of neurons	Num. of neurons in each population	Max. Num. of generations	Max. Num. of generations in each population	α_2
TSEA	1*	-	neu and neu+1	-	0.1*gen	1
EDDSig	1S	neu	-	gen	-	0.5
EDDSig	2S	neu+1	-	gen	-	0.5
TSEASig	1S*	-	neu and neu+1	-	0.1*gen	0.5

4 Experimentation

4.1 Data Sets and Validation Technique

Table 2 summarizes the data sets employed. Most of them are publicly available at the UCI repository [14] and the last two concern complex real-world problems. The following fourteen ones have been used: Statlog (*Australian* credit approval), *Balance* scale, breast *Cancer* Wisconsin, *Heart* disease (Cleveland), *HeartY, Hepatitis*, *Horse* colic, Thyroid disease (allhypo, *Hypothyroid*), Thyroid disease (*Newthyroid*), *Pima*

Indians diabetes, *Waveform* database generator (version 2) and *Yeast* regarding the UCI data sets, and *BTX* and *Listeria* monocytogenes as complex real-world problems. BTX is a multi-class classification environment problem for different types of drinking waters [15]. *Listeria* monocytogenes is a bi-class problem in predictive microbiology [16].

All nominal variables have been converted to binary ones. Also, the missing values have been replaced in the case of nominal variables by the mode or, when concerning continuous variables, by the mean, considering the full data set. The experimental design uses the cross validation technique called stratified hold-out that consists of splitting the data into two sets: training and test set, maintaining the class distribution of the samples in each set approximately equal as in the original data set. Their sizes are approximately *3N/4* and *N/4*, being *N* the number of patterns in the problem. The proportions do not match in BTX [15] and Listeria [17] because the data is prearranged in two sets due to their specific features.

Table 2. Summary of the data sets used

Data set	Total Patterns	Training Patterns	Test Patterns	Features	Inputs	Classes
Australian	690	517	173	14	51	2
Balance	625	469	156	4	4	3
Cancer	699	525	174	10	9	2
Heart	303	227	76	13	26	2
HeartY	270	202	68	13	13	2
Hepatitis	155	117	38	19	19	2
Horse	368	276	92	27	83	2
Hypothyroid	3772	2829	943	29	29	4
Newthyroid	215	161	54	5	5	3
Pima	768	576	192	8	8	2
Waveform	5000	3750	1250	40	40	3
Yeast	1484	1112	372	8	8	10
BTX	63	42	21	3	3	7
Listeria	539	305	234	4	4	2

4.2 Common Parameters of the Different Methodologies and Specific Parameters Depending on the Dataset

Table 3 introduces the common parameters for all datasets related to TSEASig, EDDSig and TSEA methodologies. All values are according with our previous works [4, 18]. In the case of TSEA, since PUNNs are used, data is normalized to avoid input values near to 0, which can produce very large values of the outputs for negative exponents.

Table 3. Common parameters for TSEASig, EDDSig and TSEA

Parameter/Feature	Value	
	TSEASig/EDDSig	TSEA
Population size (*N*)	1000	1000
gen-without-improving	20	20
Interval for the terms/exponents w_{ji}	[-5, 5]	[-5, 5]
Interval for the coefficients β_j^l	[-10, 10]	[-5, 5]
Initial value of α_1	0.5	0.5
Initial value of α_2	0.5	1
Normalization of the input data	[0.1, 0.9]	[1, 2]
Number of nodes in node addition and node deletion operators	[1, 2]	[1, 2]

The values of the *neu* and *gen* parameters depend on the data set and are shown in Table 4. The decision about the number of neurons is a very difficult task in the scope of NNs, but determining the optimal values is a challenge. With respect to the number of generations, we have defined three kinds of values: small (100-150), medium (300), large (500) and very large (1000). Again, the optimal number is unknown; however the algorithm has a stop criterion to avoid evolving up to the maximum number of generations if there is no improvement. We have given values of our choice to the two parameters depending on the complexity of the data set (number of

Table 4. Values of TSEA/EDDSig/TSEASig parameters depending on the data set

Data set	TSEA		EDDSig		TSEASig	
	Num. of neurons in each population (*neu* and *neu+1*)	Max. Num. of generations in each population	Num. of neurons (*neu*)	Max. Num. of generations (*gen*)	Num. of neurons in each population (*neu* and *neu*+1)	Max. Num. of generations in each population
Australian	4 and 5	100	4	100	4 and 5	100
Balance	5 and 6	150	4	300	4 and 5	300
Cancer	2 and 3	100	2	100	2 and 3	100
Heart	3 and 4	300	3	300	3 and 4	300
HeartY	4 and 5	100	4	100	4 and 5	100
Hepatitis	3 and 4	100	3	100	3 and 4	100
Horse	4 and 5	300	4	300	4 and 5	300
Hypothyroid	3 and 4	500	3	500	3 and 4	500
Newthyroid	3 and 4	300	3	300	3 and 4	300
Pima	3 and 4	120	3	100	3 and 4	100
Waveform	3 and 4	500	3	500	3 and 4	500
Yeast	11 and 12	1000	11	1000	11 and 12	1000
BTX	5 and 6	500	5	500	5 and 6	500
Listeria	4 and 5	300	4	300	4 and 5	300

classes, inputs, instances,…). Other times the values are based on previous works [4, 18]. EDDSig and TSEASig values are in concordance to compare the performance of both methodologies. Sometimes, the values differ between methodologies. The initial tests with sigmoidal units for Balance dataset sheds light on that a high number of neurons provokes overfitting.

4.3 Nonparametric Statistical Analysis

We follow the recommendations pointed out by J. Demšar [19] to perform nonparametric statistical tests to determine the statistical significance of the differences in rank observed for each method with all data sets. There are two methods, Friedman and Iman-Davenport tests. The former test is based in χ_F^2 statistic; the null hypothesis states that all algorithms perform equal. The latter test is based of F_F which is a better statistic, derived from χ_F^2. F_F is distributed according to the F-distribution with $(k-1)$ and $(k-1)(N-1)$ degrees of freedom with k algorithms and N datasets. If the null-hypothesis is rejected, we can proceed with a post-hoc test. Nemenyi test has been performed to compare all classifiers to each other. The critical difference (CD) can be computed from critical values, k and N. The considered significance levels have been 0.05 for Iman-Davenport test, and 0.05 and 0.10 for the post-hoc method.

5 Results

First of all, this section presents the results obtained related to the Correct Classification Ratio (CCR) in the test set with TSEA, EDDSig and TSEASig methodologies. After that, a nonparametric statistical analysis compares all of them. Next, an analysis of the computational cost is performed. Finally, we report a summary of the results obtained with a good number of classifiers, from the scope of NNs or classical/modern machine learning.

5.1 Results Applying TSEA, EDDSig and TSEASig

The results obtained by applying TSEA [4] are presented, along with those obtained with EDDSig and TSEASig. In the case of EDDSig two configurations have been considered (1S and 2S). In TSEA, there were two configurations 1* and 2*; however, only one (1*) is considered to make a fair comparison with the remaining methodologies. In TSEASig, the single configuration is 1S*. EDDSig configurations are equivalent to 1S*.

Table 5 shows the mean and standard deviation (SD) of the CCR and the topology for each data set and configuration for a total of 30 runs or iterations. By rows, the best result appears in boldface. The value obtained with TSEASig is in italics if it is better than the two values related to EDDSig. The descriptive analysis of the data reveals that the TSEA methodology obtains best results for eight data sets and TSEASig six times. Usually, TSEASig has lower SD than EDDSig and it expresses more homogeneous results of the former methodology.

Table 5. Results obtained in fourteen data sets with the different configurations related to TSEA, EDDSig and TSEASig methodologies

Data set	Methodologies							
	TSEA		EDDSig				TSEASig	
	1*		1S		2S		1S*	
	Mean±SD	Top.	Mean±SD	Top.	Mean±SD	Top.	Mean±SD	Top.
Australian	**88.11±1.56**	51:[4,5]:1	87.12±1.45	51:4:1	87.46±1.18	51:5:1	86.55±1.43	51:[4,5]:1
Balance	**96.20±1.06**	4:[5,6]:2	94.14±1.87	4:4:2	95.08±1.27	4:5:2	93.93±2.21	4:[4,5]:2
Cancer	**98.74±0.61**	9:[2,3]:1	98.39±0.74	9: 2:1	98.69±0.54	9:3:1	98.31±0.69	9:[2,3]:1
Heart	83.68±2.57	26:[3,4]:1	82.89±2.46	26:3:1	83.02±2.59	26:4:1	***83.85±2.65***	26:[3,4]:1
HeartY	**84.01±3.05**	13:[4,5]:1	83.28±2.92	13:4:1	83.18±2.61	13:5:1	*83.87±2.66*	13:[4,5]:1
Hepatitis	85.26±4,34	19:[3,4]:1	86.22±4.77	19:3:1	85.78±4.12	19:4:1	**86.75±3.41**	19:[3,4]:1
Horse	85.50±2.97	83:[4,5]:1	85.21±2.98	83:4:1	85.40±2.69	83:5:1	***88.04±2.19***	83:[4,5]:1
Hypothyroid	**95.37±0.40**	29:[3,4]:3	94.69±0.39	29:3:3	94.94±0.35	29:4:3	*95.15±0.18*	29:[3,4]:3
Newthyroid	**94.81±0.89**	5:[3,4]:3	94.07±0.75	5:3:3	94.25±1.01	5:4:3	*94.38±0.59*	5:[3,4]:3
Pima	78.63±1.33	8:[3,4]:1	77.62±1.62	8:3:1	78.14±1.60	8:4:1	***78.76±1.39***	8:[3,4]:1
Waveform	84.46±0.92	40:[3,4]:2	85.35±1.45	40:3:2	85.96±1.22	40:4:2	***86.58±1.18***	40:[3,4]:2
Yeast	60.05±1.10	8:[11,12]:9	58.96±1.27	8:11:9	59.24±1.20	8:12:9	***60.33±0.61***	8:[11,12]:9
BTX	**79.68±7.39**	3:[5,6]:6	72.85±7.41	3:5:6	73.01±4.89	3:6:6	*76.34±5.65*	3:[5,6]:6
Listeria	**86.54±1.67**	4:[4,5]:1	85.06±1.13	4:4:1	85.43±0.97	4:5:1	*85.75±0.66*	4:[4,5]:1

5.1.1 Statistical Analysis

Now, we compare TSEA, EDDSig and TSEASig methodologies by means of nonparametric statistical tests. To determine whether there are significant differences we apply an Iman-Davenport test. Since two configurations were run for EDDSig, now we consider the best value of the two mean ones reported in Table 5. The average ranks of the different methodologies are 2.64, 1.78 and 1.57 respectively for EDDSig, TSEASig and TSEA. According to Iman-Davenport test results, since the statistic $F_F = 6.16$ is higher than the critical value $F(2,26) = 3.37$ at $\alpha = 0.05$ the null-hypothesis is rejected. Therefore, we proceed with post-hoc Nemenyi test. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the CD. Table 6 shows the Nemenyi test results where the ranking difference between each different pair and the detected significant difference level have been indicated for more clarity. In a single row, the CD (at $\alpha = 0.05$ and $\alpha = 0.10$) is shown.

Table 6. Pairwise comparisons of the TSEA, EDDSig and TSEASig methodologies by means of a Nemenyi test

	EDDSig	TSEASig	TSEA
EDDSig		0.86º	1.07 *
TSEASig			0.21
$CD_{(\alpha = 0.05)} = 0.89$; $CD_{(\alpha = 0.10)} = 0.78$			
Each filled cell contains the ranking difference between the methods in the row and the column. Also, it is specified if the former method outperforms the latter one at a significance level of 0.05 (*) or 0.10 (º)			

An analysis based upon the results Nemenyi test allow us to state the following. There are significant differences between the TSEASig and EDDSig at $\alpha = 0.10$. Consequently, TSEASig is better than EDDSig. In other words, regarding to ANNs with sigmoidal units, the methodology based in the two-stage algorithm outperforms the standard one based in EDD. Comparing TSEA and TSEASig there are not significant differences. It means that the new approach is competitive with respect to TSEA. We can conclude that the methodology based on the two-stage algorithm is suitable both for sigmoidal and product units. Finally, TSEA is significantly better than EDDSig at $\alpha = 0.05$.

5.1.2 Analysis of Computational Cost

The comparison between TSEA, EDDSig and TSEASig methodologies is completed by means of a computational cost analysis. Experiments have been run in a desktop computer with an Intel Core 2 Quad processor at 2.4GHz and 2GB RAM of physical memory. The acceleration rate of the *i* method with respect to *j* method is given by Eq. 4.

$$Acceleration\_Rate(i, j) = \frac{time(j)}{time(i)} \tag{4}$$

Table 7 reports the time results concerning to the computational cost per iteration measured in seconds (s). The first column specifies the data set name. From second to fifth columns are showed the elapsed time of an iteration with each configuration of the different methodologies. Two last columns depicted the accelerate rates for TSEASig regards to EDDSig and TSEA. Last row contains the average of the values in the column. Since two configurations of EDDSig are equivalent to one of

Table 7. Computational cost and acceleration rates of TSEA, EDDSig and TSEASig

Data set	**Computational cost (s)**				**Acceleration rate**	
	Methodologies				(TSEASig, EDDSig)	(TSEASig, TSEA)
	TSEA	EDDSig		TSEASig		
	1*	1S	2S	1S*		
Australian	303	126	155	191	1.47	1.59
Balance	287	317	370	382	1.80	0.75
Cancer	98	40	51	61	1.49	1.61
Heart	207	114	133	134	1.84	1.54
HeartY	62	34	37	51	1.39	1.22
Hepatitis	36	19	22	25	1.64	1.44
Horse	817	304	378	344	1.98	2.38
Hypothyroid	6503	4525	6090	6694	1.59	0.97
Newthyroid	122	83	101	116	1.59	1.05
Pima	105	63	73	88	1.55	1.19
Waveform	9213	5217	6412	7155	1.63	1.29
Yeast	49320	21983	30349	28538	1.83	1.73
BTX	256	158	174	190	1.75	1.35
Listeria	231	168	198	216	1.69	1.07
Average	4825.71	2367.93	3181.64	3156.07	1.66	1.37

TSEASig, the acceleration rate (TSEASig, EDDSig) is calculated as the sum of the times of 1S and 2S divided by the time of 1S*.

Having a look at the Table 7, we conclude that regarding to sigmoidal units, TSEASig is 1.66 times faster than EDDSig. In the comparison between TSEASig and TSEA, the former is 1.37 times faster than the latter. The empirical times give notice that the proposed methodology, TSEASig, is much more efficient than the previous methodology, TSEA. Moreover TSEASig is faster than EDDSig. The efficiency measures are based on the computation time of an iteration of the whole population throughout the EA running. The speed depends on the own evolutionary process and the number of mathematical operations that are involved to calculate the NN output.

5.2 Results Obtained with a Good Number of Classifiers

Now, a general review is made about the results obtained with another kind of NNs and other machine learning algorithms. In the literature, a huge amount of tests has been carried out with some of the data sets here considered. Our purpose is to view some of the methods that have been tested with some of the data sets dealt with in the current chapter.

Related to NNs, we have reported TSEASig, TSEA, the traditional MLP model [20] with a learning Back-Propagation method (BP); the RBF model [21] with a normalized Gaussian, HMOEN_L2 [22], SONG [23] and CC-EBFNN [24]. As classical or modern machine learning algorithms have been included: C4.5, k-nearest neighbours (k-NN) -with the best accuracy for k in {1, 3, 5, 7, 9}-, PART and SVM [25]. Since, MLP, RBF, C4.5, k-NN, PART and SVM are implemented in Weka tool [26], we have conduced the experiments. The parameters have been set to the default values with the exceptions that we describe; for BP were the following: learning rate $\eta = 0.3$, momentum $\alpha = 0.2$ and the number of epochs was adjusted in each data set. To determine the learning and the momentum we try out a grid search algorithm with values in the range [0, 1] in 0.1 steps. Regarding the topology of the models in the case of MLP and RBF we have considered the default one.

Table 8 includes the results of all classifiers, averaged for 30 runs in non-deterministic algorithms, with each of the data set; the best ones in boldface and in italics the second best ones, as well as the averages of the methods run by us with all data sets. From a purely descriptive analysis of the results, it can be concluded that SVM and RBF obtain the best result for three data sets, TSEASig, MLP, C4.5 and PART for two data sets, and TSEA and k-NN once. There is not one method that performs really well with all data sets; depending on the data set, the best classifier belongs to either the neural networks approach or to the classical/modern machine learning. Furthermore, the TSEA method achieves the highest mean accuracy ($\overline{CCR} = 85.79$), followed by the TSEASig ($\overline{CCR} = 85.61$) and RBF ($\overline{CCR} = 84.42$). The previous statements point out that methodology based on two stages is proper with product or sigmoidal units.

Table 8. Summary of the results in fourteen data sets comparing TSEASig to other methods related to neural networks or classical/modern machine learning approaches

Method	Australian	Balance	Cancer	Heart	HeartY	Hepatitis	Horse
TSEASig	86.55	*93.93*	98.31	83.85	83.87	86.75	*88.04*
TSEA	*88.11*	**96.20**	*98.74*	83.68	84.01	85.26	85.50
HMOEN_L2	-	-	96.30	-	-	80.30	-
MLP	84.10	93.78	97.81	*84.82*	**84.29**	84.73	**88.51**
RBF	75.84	88.27	97.20	**86.75**	83.79	*89.30*	80.47
SONG	-	87.80	97.40	-	-	-	-
CC-EBFNN	-	-	96.67	82.45	-	-	-
C4.5	86.71	83.33	97.13	75.00	*84.21*	84.21	*88.04*
k-NN	85.55	91.67	**98.85**	82.89	83.70	86.84	*88.04*
PART	84.97	85.26	97.13	80.26	82.12	81.58	85.87
SVM	**88.44**	88.46	98.28	82.89	80.37	**89.47**	*88.04*

Method	Hypothyroid	Newthyroid	Pima	Waveform	Yeast	BTX	Listeria
TSEASig	95.15	94.38	**78.76**	86.58	**60.33**	76.34	85.75
TSEA	95.37	94.81	*78.63*	84.46	60.05	*79.68*	*86.54*
HMOEN_L2	-	-	78.50	-	-	-	-
MLP	94.39	97.08	75.94	84.85	*60.11*	54.12	84.49
RBF	92.83	**98.27**	77.34	*87.29*	59.83	**80.95**	83.70
SONG	-	*97.20*	76.40	-	-	-	-
CC-EBFNN	-	-	76.04	-	-	-	-
C4.5	**99.15**	96.30	74.48	76.40	54.84	**80.95**	85.93
k-NN	94.06	94.44	75.00	81.12	48.39	76.19	85.93
PART	*98.83*	92.59	74.48	78.16	56.72	**80.95**	**86.67**
SVM	93.85	88.89	78.13	**88.80**	55.91	61.90	80.74

$\overline{CCR}\ (TSEASig) = 85.61$; $\overline{CCR}\ (TSEA) = 85.79$; $\overline{CCR}\ (MLP) = 83.50$;
$\overline{CCR}\ (RBF) = 84.42$
$\overline{CCR}\ (C4.5) = 83.22$; $\overline{CCR}\ (k-NN) = 83.76$; $\overline{CCR}\ (PART) = 83.26$;
$\overline{CCR}\ (SVM) = 83.16$

6 Conclusions

This chapter aims to tackle multi-classification problems using evolutionary artificial neural networks based on sigmoidal units. We have extended to sigmoidal units a previous methodology for neural networks based on product units based with an EA divided in two phases. The new approach has been called TSEASig. Our basic assumption is that it is convenient to employ a methodology based on a population with more diverse models in terms network architectures and this produces an improvement in efficiency and accuracy.

The TSEASig methodology is applied to solve fourteen classification problems, twelve from the UCI repository and two real-world problems, with a great deal of variety in the number of instances, features and classes. The test results confirm that our approach obtains promising results, achieving a high classification rate level in the data sets at a lower computational cost than EDDSig.

A comparison between TSEASig, EDDSig and TSEA has been carried out by means of nonparametric tests. The test results reveal that there are significant differences between TSEASig and EDDSig. However, significant differences are not present between TSEASig and TSEA; this fact indicates that the methodology in two stages is also suitable for sigmoidal units. According to the above results, our new learning methodology of neural networks, TSEASig, based on sigmoidal units is competitive in accuracy and more efficient than the remaining methodologies. The empirical times give notice that TSEASig is 1.37 times faster than TSEA.

We have also summarized the results obtained with other kinds of neural networks and classical/modern machine learning algorithms. From the analysis of the results we can observe the good performance of our new approach.

Acknowledgments. This chapter has been partially subsidized by TIN2007-68084-C02-02 and TIN2008-06681-C06-03 projects of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain).

References

1. Maaranen, H., Miettinen, K., Mäkelä, M.M.: Quasi-random initial population for genetic algorithms. Computers & Math. with Appl. 47, 1885–1895 (2004)
2. Wang, L., Zheng, D.Z., Tang, F.: An improved evolutionary programming for optimization. In: Proc of the 4th world Congress on Intelligent Control and Automation, vol. 3, pp. 1769–1773. IEEE, Shanghai (2002)
3. Amor, H.B., Rettinger, A.: Intelligent exploration for genetic algorithms: using self-organizing maps in evolutionary computation. In: Proc. of the 2005 Conference on Genetic and Evolutionary Computation, GECCO 2005, pp. 1531–1538. ACM, Washington DC (2005)
4. Tallón-Ballesteros, A.J., Hervás-Martínez, C.: A two-stage algorithm in evolutionary product unit neural networks for classification. Expert Syst. with Appl. 38(1), 743–754 (2011)
5. Durbin, R., Rumelhart, D.: Products units: a computationally powerful and biologically plausible extension to back-propagation networks. Neural Comp. 1(1), 133–142 (1989)
6. Miller, G.F., Toddm, P.M., Hegde, S.U.: Designing neural networks using genetic algorithms. In: Proc. of the 3rd International Conference on Genetic Algorithms, ICGA 1989, pp. 379–384. Morgan Kaufmann, George Mason University, Fairfax, Virginia, USA (1989)
7. Yao, X.: Evolving artificial neural networks. Proc. of the IEEE 87(9), 1423–1447 (1999)
8. Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE Trans. on Neural Netw. 8(3), 694–713 (1997)
9. Yao, X., Liu, Y.: Making use of population information in evolutionary artificial neural networks. IEEE Trans. on Syst., Man and Cybernetics, Part B: Cybernetics 28(3), 417–425 (1998)
10. Azzini, A., Tettamanzi, A.G.B.: A new genetic approach for neural network design, vol. 82, pp. 289–323. Springer, Heidelberg (2008)
11. Martínez-Estudillo, F.J., Hervás-Martínez, C., Gutiérrez, P.A., Martínez-Estudillo, A.C.: Evolutionaryproduct-unit neural networksclassifiers. Neurocomputing 72(1-3), 548–561 (2008)

12. Martínez-Estudillo, A.C., Martínez-Estudillo, F.J., Hervás-Martínez, C., García-Pedrajas, N.: Evolutionary product unit based neural networks for regression. Neural Netw. 19, 477–486 (2006)
13. Tallón-Ballesteros, A.J., Gutiérrez-Peña, P.A., Hervás-Martínez, C.: Distribution of the search of evolutionary product unit neural networks for classification. In: Proc. of the IADIS Internacional Conference on Applied Computing, AC 2007, pp. 266–273. IADIS, Salamanca (2007)
14. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. School of Information and Computer Science. University of California, Irvine (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
15. Hervás, C., Silva, M., Gutiérrez, P.A., Serrano, A.: Multilogistic regression by evolutionary neural network as a classification tool to discriminate highly overlapping signals: Qualitative investigation of volatile organic compounds in polluted waters by using headspace-mass spectrometric analysis. Chemom. and Intell. Lab. Syst. 92, 179–185 (2008)
16. Beuchat, L.R.: Listeria monocytogenes: incidence on vegetables. Food Control 7(4-5), 223–228 (1996)
17. Valero, A., Hervás, C., García-Gimeno, R.M., Zurera, G.: Product unit neural network models for predicting the growth limits of Listeria monocytogenes. Food Microbiol. 24(5), 452–464 (2007)
18. Gutiérrez, P.A., Hervás, C., Lozano, M.: Designing multilayer perceptrons using a guided saw-tooth evolutionary programming algorithm. Soft Computing 14, 599–613 (2010)
19. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
20. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, Oxford (2004)
21. Howlett, R.J., Jain, L.C.: Radial Basis Function Networks 1: Recent Developments in Theory and Applications. Springer, Heidelberg (2001)
22. Goh, C.K., Teoh, E.J., Tan, K.C.: Hybrid multi objective evolutionary design for artificial neural networks. IEEE Trans. on Neural Netw. 19(9), 1531–1548 (2008)
23. Inoue, H., Narihisa, H.: Self-organizing neural grove and its applications. In: Proc. of the International Joint Conference on Neural Networks, IJCNN 2005, vol. 2, pp. 1205–1210. IEEE, Montreal (2005)
24. Tian, J., Li, M., Chen, F.: A hybrid classification algorithm based on coevolutionary EBFNN and domain covering method. Neural Comput. & Applic. 18, 293–308 (2009)
25. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
26. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco (2005)