Accuracy Increase on Evolving Product Unit Neural Networks via Feature Subset Selection

Antonio J. Tallón-Ballesteros[1,2(✉)], José C. Riquelme[2], and Roberto Ruiz[1]

[1] Area of Computer Science, Pablo de Olavide University, Seville, Spain
ajtalbal@upo.es
[2] Department of Languages and Computer Systems,
University of Seville, Seville, Spain

Abstract. A framework that combines feature selection with evolutionary artificial neural networks is presented. This paper copes with neural networks that are applied in classification tasks. In machine learning area, feature selection is one of the most common techniques for preprocessing the data. A set of filters have been taken into consideration to assess the proposal. The experimentation has been conducted on nine data sets from the UCI repository that report test error rates about fifteen percent or above with reference classifiers such as C4.5 or 1-NN. The new proposal significantly improves the baseline framework, both approaches based on evolutionary product unit neural networks. Also several classifiers have been tried in order to illustrate the performance of the different methods considered.

1 Introduction

Many techniques addressing the classification problem [1] have been presented by the machine learning community. Depending on the nature of the algorithm we can distinguish, among others, neural networks, rule-based classifiers and decision trees. Neural networks models play an important role in pattern recognition [13]. The possible inputs to an Artificial Neural Network (ANN) could be extremely large in the context of many practical problems. There may be some redundancy among different inputs. A huge number of inputs to an ANN increase its size and thus require more training data and longer training times in order to achieve reasonable generalization ability. Pre-processing is often needed to reduce the number of inputs to an ANN.

This paper aims at improving the accuracy and getting simpler neural models with a lower number of inputs and, if possible, containing a lower number of hidden neurons. The kind of the neural networks that are in the scope of the current work are feed-forward neural networks composed by product units in the hidden layer. Basically, the training of the models is carried out by an evolutionary programming algorithm [4]. More concretely we utilise a framework following a master-slave approach, where the master distributes a configuration to slave processes. A preliminary study of this base approach is described in [16].

Now, the novel ingredient is a preprocessing phase prior to the training of the classification models.

The remainder of this paper is organised as follows: Sect. 2 describes some concepts about the training of Product Unit Neural Networks (PUNNs), the experimental design distribution and feature selection; Sect. 3 presents proposal; Sect. 4 details the conducted experimentation; then Sect. 5 shows and analyzes the results obtained; finally, Sect. 6 states the concluding remarks.

2 Methodology

2.1 Product Unit Neural Networks and Training Procedure

Among the different types of neural network architectures, the most popular are feed-forward ones. Within this kind, single hidden-layer networks are very powerful due to their universal approximation property. Multiplicative ANNs [17] contain nodes that multiply their inputs instead of adding them. This class of networks comprises such types as sigma-pi networks and product unit neural networks. The latter type was introduced by R. Durbin and D. Rumelhart [5] and is the study object of the current paper. The training of the neural networks is performed by means of an evolutionary programming algorithm to simultaneously learn the architecture and weights of the PUNN classification model. The neural network topology is a three-layer architecture, with k (number of features of the problem at hand) nodes in the input layer, m ones and a bias one in the hidden layer and a number of nodes equals to the number of classes minus one in the output layer. The m value is determined by the training algorithm. The transfer function of each node in the hidden and output layers is the identity function. We have considered a standard soft-max activation function, associated with the g network model with J classes, given by:

$$g_j(\mathbf{x}) = \frac{exp\ f_j(\mathbf{x})}{\sum_{j=1}^{J} exp\ f_j(\mathbf{x})} \quad j = 1, ..., J \tag{1}$$

where $f_j(\mathbf{x})$ is the output of node j for pattern $\mathbf{x}$ and $g_j(\mathbf{x})$ is the probability that this pattern belongs to class j. Given a training set $D = (x_i, yi)\ i = 1, ..., N$, a function of cross-entropy error is used to evaluate a network g with the instances of a problem, which is reflected in the following expression:

$$l(g) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{J} (y_i^j ln(g_j(\mathbf{x}_i))) \tag{2}$$

and substituting g_j defined in (2),

$$l(g) = -\frac{1}{N} \sum_{i=1}^{N} \left(-\sum_{j=1}^{J} y_i^j f_j(\mathbf{x_i}) + ln(\sum_{j=1}^{J} exp\ f_j(\mathbf{x}_i)) \right) \tag{3}$$

where y_i^j is the target value for class j with pattern $\mathbf{x_i}$ ($y_i^j = 1$ if $\mathbf{x_i} \in$ class j and $y_i^j = 0$ otherwise), $f_j(\mathbf{x_i})$ is the output value of the neural network for the output neuron j with pattern $\mathbf{x_i}$. Observe that soft-max transformation produces probabilities that sum to one and therefore the outputs can be interpreted as the conditional probability of class membership. Thus, the number of nodes in the output layer is equal to the number of classes minus one in the problem. Since the EA objective is to minimize the chosen error function, a fitness function is used in the form $A(g) = (1 + l(g))^{-1}$.

The main issues about the evolutionary training algorithm are briefly explained next. The search begins with a random initial population and, for each iteration, the population is modified using a population-update algorithm founded on parametric and structural mutations. The algorithm loops are repeated until the maximum number of generations, in each case, is reached or until the best individual or the population mean fitness does not improve during $gen - without - improving$ (20 in this paper) generations. The population is subjected to the operations of replication and mutation. More details and common parameter values of the algorithm are found in [16]. Crossover is not used due to its potential disadvantages in evolving artificial networks. With these properties the algorithm falls into the class of evolutionary programming.

2.2 Experimental Design Distribution

The starting framework of the current work is named Experimental Design Distribution (EDD) and follows a master-slave programming model. The master process prepares a base configuration that is distributed to all the slave processes that update the received configuration. Depending on the identity of the slave the task to be performed is different in the sense that may act on a concrete parameter doing a specific operation with a single value of the base configuration. Next, each process of every type runs the training algorithm described in the previous subsection using the proper (base/updated) configuration. The advantage of this framework is that a single configuration file and the number of slaves to be spawned is required. EDD is able to distribute two or three parameters over a maximum of eight computing nodes, that is, each process is mapped to one processor that is used in a exclusive way. We may have one master and seven slave processes. This is the first approach published in [16]. There we came to the empirical conclusion that is very useful to distribute three parameters. From the eight configurations of that proposal, the configurations that do not reduce the number of generations get better results. This fact motivates us to only consider hereinafter the first four configurations of the approach delivering three parameters among the processing system. In other words, it is just the same that asserting that the maximum number of neurons in the hidden layer and the parameter value associated with the parametrical mutation are distributed among four computation nodes.

2.3 Feature Selection

Feature selection may be defined as the problem of picking up a subset of features that are necessary and sufficient to describe the target concept [9]. A taxonomy of the feature selection algorithms may be based on the attribute evaluation measure: depending on the type (filter or wrapper technique) or on the way that features are evaluated (individual or subset evaluation). The filter model relies on general characteristics of the data (such as consistency, correlation and distance) to assess and select feature subsets without involving any data mining algorithm. The wrapper model requires a predetermined mining algorithm and uses its performance as evaluation criterion. This paper pays attention to feature subset selection implemented as filters. In this context, it is a fact that two kinds of features are generally perceived as being unnecessary: features that are irrelevant to the target concept, and features that are redundant given other features. BIRS (Best Incremental Ranked Subset [15]) method was proposed in a previous work to obtain relevant features and to remove redundancy. These features selected are considered as input variables to the network models that we get in this paper via EDD framework. Since BIRS belongs to a hybrid category, the selection process does not follow the typical paths and is divided into two phases: in the first one, features are evaluated individually, providing a ranking based on a criterion; in phase two, a feature subset evaluator is applied to a certain number of features in the previous ranking according to a search strategy. BIRS can use any evaluator in the two stages. In the current contribution, BIRS uses as a subset evaluator CFS (Correlation-based Feature Selection [6]) and CNS (CoNSistency based measure [10] -that are established on correlation and consistency concepts- at the second phase, and SOAP (Selection Of Attributes by Projection [14]) measure and the own subset evaluator at the first phase as a ranking evaluator. Thus, in the experiments, spBI_CNS indicates that SOAP is utilised as an individual measure in the first part of BIRS, and CNS is employed as a subset evaluator in the second part. In the same way, cfBI_CFS denotes that CFS evaluator will be used in both part of the BIRS algorithm.

3 Proposal Description

The current paper introduces *Experimental Design Distribution with Feature Selection* (EDDFS) framework, a combination of some FS methods, one by one independently, with EDD. First of all, some feature selectors are applied in an independent way to the training set of all data sets in order to obtain a list of attributes, for each of them, that it is considered for training and test phases. In this way, two subsets (training and test subset) are generated, where only most relevant features are included. It is important to remark that the feature selection is performed only with training data; the test subset has exactly the same features as the reduced training set. These subsets are taken as input to the evolutionary algorithm. EDDFS methodology operates with four feature selectors. As a result of the FS stage, a list of relevant features is obtained with each of the FS methods for each data set. The EDDFS properties are the

Table 1. Configurations of the EDD (baseline) and EDDFS frameworks

Framework								
	EDD				EDDFS			
$\sharp Configuration$	1	2	3	4	$1'$	$2'$	$3'$	$4'$
$\sharp Neurons$ (**neu**)	neu	$neu+1$	neu	$neu+1$	neu'	$neu'+1$	neu'	$neu'+1$
$\sharp Gener.$ (**gen**)	gen	gen	gen	gen	gen'	gen'	gen'	gen'
α_2	1	1	1.5	1.5	1	1	1.5	1.5

following: (a) PUNNs have been utilised, with a number of neurons in the input layer 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 selected features; and the number of nodes in the output layer equal to the number of classes minus one because a soft-max type probabilistic approach has been used; (b) four experiments have been performed for each problem, where two different values have been used for -associated with the residual of the updating expression of the output-layer weights- and the number of neurons in the hidden layer; (c) it employs similar terminology to aforementioned EDD; (d) four different configurations (1', 2', 3' and 4') are applied to subsets obtained with each of the selectors, for each data set. The parameters of each configuration are neu, gen and α_2. The first two ones take specific values depending on the data set and the last one depends on the configuration number (1', ...). Table 1 shows the main aspects of both EDD and EDDFS configurations.

4 Experimentation

Table 2 summarizes the data sets employed. All of them have been downloaded from the *University of California at Irivine* (UCI) repository [2]. Since we are using neural networks, all nominal variables have been converted to binary ones; due to this, sometimes the number of inputs is greater than the number of features. Also, the missing values have been replaced in the case of nominal variables by the mode or, when concerning continuous variables, by the mean, taking into account the full data set. These data sets have in common that present error rates in test phase about 15 % or above with reference classifiers such as C4.5 or 1-NN without feature selection. The number of samples in the training and test sets ensues from the splitting of the data sets following a experimental design via a cross validation technique called $hold-out$ that consists of dividing the data into two sets: a training and a test set. In our case, the sizes of the training and test sets are three and one quarters of the number of patterns in the problem, respectively; these percentages are similar to those used in [11]. More exactly, we have utilised a stratified holdout where the two sets are stratified [7] so that the class distribution of the samples in each set is approximately the same as in the original data set.

Table 2. Summary of the data sets and specific parameter values for EDD and EDDFS frameworks

Data set	Size	Train.	Test	Feat.	Inp.	Cl.	neu; gen	neu'; gen'
Breast	286	215	71	9	15	2	9; 500	7; 500
Heart	270	202	68	13	13	2	6; 500	4; 25
Hepatitis	155	117	38	19	19	2	3; 100	3; 100
Parkinsons	195	146	49	23	22	2	6; 500	3; 500
Pima	768	576	192	8	8	2	3; 120	3; 120
Promoter	106	80	26	58	114	2	11; 500	5; 300
Waveform	5000	3750	1250	40	40	3	3; 500	3; 500
Winequality-red	1599	1196	403	11	11	6	6; 300	4; 300
Yeast	1484	1112	372	8	8	10	11; 500	11; 500

Regards to EDD methodology, the concrete values of *neu* and *gen* parameters depend on the data set and are shown in the eighth last columns of Table 2. The decision about the number of neurons in the hidden-layer is a very difficult task in the scope of neural networks; we have done a preliminary study splitting the training set in two stratified sets with three and one quarters on the patterns and exploring the range [2–12] for the number of hidden neurons. Concerning the number of generations, we have defined three kinds of values: small (100–120), medium (300) and large (500). In EDDFS, again there are two parameters: *neu* and *gen*, whose values, neu' and gen', are defined for each data set.The assignment of values in EDD is not trivial, but now in EDDFS this decision is more difficult because there are four FS methods and the values are common for all of them. Kwak and Choi [8] have also considered this idea. The problem of finding the best architectures in neural networks that employ input feature selection remains unsolved. In most of cases the number of neurons is defined by us with a lower value than model EDD. There is no heuristics to guide this process, so we have used values at a guess. Regards the number of generations, the values are the same than in previous methodology except in the case of Promoter and Heart data sets. In the former, the dimensionality reduction is very important and the generation number has been change to medium value. In the latter the algorithms converges soon and a very small value (25) is utilised.

Table 3 depicts the methods used in the experimentation regarding the data preparation stage by means feature selection. There are four ones with and one without feature selection that belong respectively to EDDFS (the current proposal) and EDD frameworks. Last column defines an abbreviated name for each of them that is employed in next sections.

As previously mentioned, four FS methods have been applied to each data set. Table 4 illustrates, for each data set, the number of inputs of the original train set (see column labelled $F0$) and those that have been obtained with the different feature selectors (see columns labelled F1-4) along with the reduction

Table 3. List of filters based on feature subset selection employed in the empirical study

Feature selector name	Search method	Subset evaluator	Framework	Abbreviation
–	*None*	*None*	*EDD*	$F0$
spBI_CFS	*spBI*	*CFS*	*EDDFS*	$F1$
cfBI_CFS	*cfBI*	*CFS*	*EDDFS*	$F2$
spBI_CNS	*spBI*	*CNS*	*EDDFS*	$F3$
cnBI_CNS	*cnBI*	*CNS*	*EDDFS*	$F4$

percentage in the inputs of each selector compared to the original data set. Last row shows the average of the number of inputs or reduction percentage of the test bed for each experimented method on this paper. The reduction percentage of the number of inputs is defined as:

$$Reduction\_of\_Inputs(\%) = \left(1 - \frac{Inputs(Fi)}{Inputs(F0)}\right) 100 \quad i = 1, ..., 4 \tag{4}$$

where i is the FS method index and $Inputs(j)$ represents the number of inputs of a given data set with method j. In all cases, FS methods successfully decreased the data dimensionality by selecting, in mean, much less than the half of the original features. Precisely, the number of selected features fluctuates between a quarter and a third of the original features. F2 method achieves a reduction percentage, on average, of 63.34 % (from 27.78 to 6.56 features in average), which is the highest overall average value obtained. Individually, Promoter data set has the highest reduction rate, above a 92 % in all cases.

Table 4. Number and reduction (%) of inputs with EDD (baseline) and EDDFS frameworks

Data set	Inputs					Reduction (%)			
	$F0$	$F1$	$F2$	$F3$	$F4$	$F1$	$F2$	$F3$	$F4$
Breast	15	4	4	2	2	73.33	73.33	86.67	86.67
Heart	13	7	7	8	9	46.15	46.15	38.46	30.77
Hepatitis	19	10	10	11	5	47.37	47.37	42.11	73.68
Parkinsons	22	5	5	7	6	77.27	77.27	68.18	72.73
Pima	8	3	3	4	5	62.50	62.50	50.00	37.50
Promoter	114	7	7	8	7	93.86	93.86	92.98	93.86
Waveform	40	14	14	15	15	65.00	65.00	62.5	62.50
Winequality – red	11	5	5	8	8	54.55	54.55	27.27	27.27
Yeast	8	5	4	7	7	37.50	50.00	12.50	12.50
Average	27.78	6.67	6.56	7.78	7.11	61.95	63.34	53.41	55.28

We follow the guidelines pointed out by J. Demšar [3] to perform nonparametric statistical tests. Iman-Davenport and Bonferroni-Dunn (Dunn, 1961) tests have been performed. The critical difference (CD) for Bonferroni-Dunn test 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

This section depicts the results obtained, measured in accuracy in the test set or in the test subset depending on that feature selection has been considered or not. First of all, we present the results obtained with EDD and EDDFS. After that, a statistical analysis compares EDD versus EDDFS to determine whether there are significant differences between applying or not feature selection with PUNN. Next, a second experiment compares, for each feature selector, the best mean values obtained with the current proposal to other classifiers using the same reduced data sets. Hence, in regard to EDD, the results have been extracted from next subsection for F1-4 methods.

5.1 Results Applying EDD and EDDFS

The results obtained by applying the EDD framework [16] are presented, along with those obtained with EDDFS. In the case of EDD, there were 8 configurations, denoted in the following way: 1, 2, ... 8. As already mentioned, this paper only deals with the first four configurations. In EDDFS, the four existing configurations are 1', ..., 4'. Table 5 shows the mean and standard deviation (SD) of the test accuracies for each data set for a total of 30 runs. From the analysis of the data, it can be concluded, from a purely descriptive point of view, that the EDDFS framework obtains best results for all data sets. Always, the SD reduction with EDDFS is clear and it expresses more homogeneous results compared to EDD.

Statistical Analysis. In this subsubsection we compare EDD and EDDFS methodologies by means of nonparametric statistical tests. To determine whether there are significant differences we apply an Iman-Davenport test. It compares the average ranks of the algorithms, where a low rank value indicates a good algorithm performance and a high value a bad algorithm performance. The average ranks of all methods, without (F0) and with FS (F1-4) are 4.78, 2.33, 2.56, 3.06 and 2.28, respectively. According to Iman-Davenport test results, since the statistic $F_F = 6.10$ is higher than the critical value at $(F(4, 32) = 2.67)$ the null-hypothesis is rejected. Therefore, we apply a post-hoc Bonferroni-Dunn test that compares a number of methods with a control method, by determining whether the average ranks differ by at least the CD. In our case, we make a comparison of the methods that employ FS (F1-4) versus the control method (F0) that does not use FS. CDs obtained by Bonferroni-Dunn test are 1.86 (at $\alpha = 0.05$)

Table 5. Results obtained in the test-bed by means of EDD and EDDFS frameworks

Data set	Filter	$Mean \pm SD$			
		$Config.$			
		$1/1'$	$2/2'$	$3/3'$	$4/4'$
Breast	$F0$	63.85 ± 3.81	63.00 ± 3.24	64.27 ± 3.89	63.43 ± 3.80
	$F1$	70.84 ± 1.92	70.93 ± 1.59	70.18 ± 1.77	70.00 ± 1.92
	$F2$	70.84 ± 1.92	70.93 ± 1.59	70.18 ± 1.77	70.00 ± 1.92
	$F3$	69.20 ± 0.48	69.10 ± 0.35	69.06 ± 0.25	69.06 ± 0.25
	$F4$	69.20 ± 0.48	69.10 ± 0.35	69.06 ± 0.25	69.06 ± 0.25
Heart	$F0$	75.93 ± 2.40	75.83 ± 3.27	76.23 ± 2.48	76.03 ± 3.50
	$F1$	76.23 ± 1.86	76.47 ± 2.12	75.93 ± 2.33	77.50 ± 2.01
	$F2$	76.23 ± 1.86	76.47 ± 2.12	75.93 ± 2.33	77.50 ± 2.01
	$F3$	76.08 ± 2.50	75.98 ± 2.30	76.47 ± 2.01	75.59 ± 2.37
	$F4$	77.40 ± 2.10	76.76 ± 2.09	77.89 ± 2.49	77.99 ± 1.79
Hepatitis	$F0$	84.47 ± 4.49	85.52 ± 4.67	84.47 ± 4.55	84.29 ± 5.33
	$F1$	88.77 ± 2.49	88.77 ± 2.93	88.95 ± 2.80	89.91 ± 2.59
	$F2$	88.77 ± 2.49	88.77 ± 2.93	88.95 ± 2.80	89.91 ± 2.59
	$F3$	89.04 ± 2.40	89.30 ± 2.49	89.56 ± 2.13	89.47 ± 2.85
	$F4$	86.67 ± 1.53	86.67 ± 1.82	86.40 ± 1.40	86.32 ± 1.45
Parkinsons	$F0$	79.66 ± 5.01	78.16 ± 4.77	78.98 ± 4.05	79.32 ± 4.76
	$F1$	79.66 ± 2.37	79.32 ± 2.32	79.93 ± 2.15	79.05 ± 2.83
	$F2$	79.66 ± 2.37	79.32 ± 2.32	79.93 ± 2.15	79.05 ± 2.83
	$F3$	80.27 ± 4.23	81.84 ± 4.20	80.61 ± 3.07	80.14 ± 3.90
	$F4$	78.03 ± 1.16	80.07 ± 2.82	78.78 ± 1.98	79.66 ± 3.24
Pima	$F0$	77.33 ± 2.36	78.61 ± 1.88	76.96 ± 1.67	77.69 ± 1.79
	$F1$	79.54 ± 0.90	79.49 ± 0.79	79.60 ± 0.87	79.89 ± 0.92
	$F2$	79.54 ± 0.90	79.49 ± 0.79	79.60 ± 0.87	79.89 ± 0.92
	$F3$	75.48 ± 1.42	75.19 ± 1.26	75.00 ± 1.17	75.31 ± 1.51
	$F4$	78.42 ± 1.09	78.76 ± 1.13	78.73 ± 1.06	78.71 ± 1.47
Promoter	$F0$	59.74 ± 9.30	58.21 ± 9.67	60.51 ± 10.00	55.51 ± 10.03
	$F1$	84.48 ± 3.97	84.62 ± 3.78	83.20 ± 3.97	82.94 ± 4.12
	$F2$	84.48 ± 3.97	84.62 ± 3.78	83.20 ± 3.97	82.94 ± 4.12
	$F3$	67.43 ± 5.77	67.05 ± 5.11	66.15 ± 5.56	67.94 ± 5.74
	$F4$	75.89 ± 4.39	75.51 ± 4.45	76.41 ± 3.74	76.53 ± 4.55
Waveform	$F0$	81.43 ± 2.10	82.78 ± 0.64	82.05 ± 1.64	84.32 ± 1.73
	$F1$	84.97 ± 1.13	86.54 ± 0.48	84.92 ± 0.98	86.30 ± 0.95
	$F2$	84.97 ± 1.13	86.54 ± 0.48	84.92 ± 0.98	86.30 ± 0.95
	$F3$	85.39 ± 1.41	85.78 ± 0.74	85.20 ± 1.14	86.37 ± 0.84
	$F4$	84.87 ± 0.93	86.75 ± 0.57	85.55 ± 1.21	85.66 ± 0.80

(Continued)

Table 5. *(Continued)*

Data set	Filter	$Mean \pm SD$			
$Winequality-red$	$F0$	61.03 ± 1.30	60.80 ± 1.25	61.16 ± 1.20	60.98 ± 1.42
	$F1$	61.67 ± 1.10	61.49 ± 0.99	61.21 ± 1.11	61.15 ± 1.30
	$F2$	61.67 ± 1.10	61.49 ± 0.99	61.21 ± 1.11	61.15 ± 1.30
	$F3$	61.70 ± 1.06	61.54 ± 1.10	61.75 ± 1.01	61.66 ± 1.05
	$F4$	61.70 ± 1.06	61.54 ± 1.10	61.75 ± 1.01	61.66 ± 1.05
$Yeast$	$F0$	59.18 ± 1.17	59.62 ± 1.27	58.50 ± 1.74	59.18 ± 1.83
	$F1$	59.82 ± 1.22	59.53 ± 1.36	58.95 ± 1.14	59.72 ± 1.46
	$F2$	54.86 ± 1.39	54.37 ± 1.16	54.16 ± 1.56	54.84 ± 1.35
	$F3$	60.10 ± 1.41	60.36 ± 1.16	59.93 ± 1.85	60.20 ± 1.46
	$F4$	60.10 ± 1.41	60.36 ± 1.16	59.93 ± 1.85	60.20 ± 1.46
$Average$	$F0$	71.40	71.39	71.46	71.20
	$F1$	76.22	76.35	75.87	76.27
	$F2$	75.67	75.78	75.34	75.73
	$F3$	73.85	74.02	73.75	73.97
	$F4$	74.70	75.06	74.94	75.09

and 1.67 (at $\alpha = 0.10$). The ranking difference with F0 are 2.45 with F1, 2.22 with F2, 1.72 with F3 and 2.50 with F4. Thus, there are significant differences between EDD applying each of the FS methods and without FS. The statistical tests points out that PUNN performance improves significantly pre-processing the data set with any of the FS methods employed in this paper. Summarising, EDDFS improves significantly EDD. However, F4 is better from the point of view of ranking difference, sharing the statistical significance level with F1 and F2.

5.2 Results Obtained with State-of-the-art Classifiers

Now, a comparison is performed between EDDFS and other machine learning algorithms. These methods are C4.5 [12], k-nearest neighbours (k-NN), -where k is 1-, SVM and PART. Since, C4.5, 1-NN, SVM and PART are implemented in Weka tool [18], we have used the same cross-validation, thus the same instances in each of the partitions, that in the first experiment. Regarding the parameters, the algorithms have been run with Weka default values. We have reported in Table 6 the results without and with FS for each data set and algorithm. From a purely descriptive analysis of the results, we can assert the following. Taking into account the data set without any FS the SVM and EDD algorithm achieves the best result in four out of nine data sets; C4.5 classifiers get once the highest accuracies. In average, SVM has the better accuracy (74.95 %), followed by EDD (72.21 %) and the remaining algorithms range from 67.28 % to 69.32 %. Applying FS, it can be concluded that the EDD method (EDDFS, the current

Table 6. Results obtained in nine data sets for several classifiers with and without feature subset selection

Data set	Filter	C4.5	1-NN	SVM	PART	EDD
Breast	$F0$	70.42	64.79	64.79	69.01	64.27
	$F1$	69.01	70.42	66.20	71.83	70.93
	$F2$	69.01	70.42	66.20	71.83	70.93
	$F3$	69.01	70.42	64.79	69.01	69.20
	$F4$	69.01	70.42	64.79	69.01	69.20
Heart	$F0$	70.59	73.53	76.47	73.53	76.23
	$F1$	73.53	73.53	76.47	77.94	77.50
	$F2$	73.53	73.53	76.47	77.94	77.50
	$F3$	73.53	75.00	76.47	75.00	76.47
	$F4$	72.06	75.00	76.47	75.00	77.99
Hepatitis	$F0$	84.21	86.84	89.47	81.58	85.52
	$F1$	84.21	89.47	86.84	84.21	89.91
	$F2$	84.21	89.47	86.84	84.21	89.91
	$F3$	89.47	92.11	89.47	86.84	89.56
	$F4$	89.47	84.21	89.47	84.21	86.67
Parkinsons	$F0$	71.43	77.55	75.51	75.51	79.66
	$F1$	75.51	79.59	75.51	77.55	79.93
	$F2$	75.51	79.59	75.51	77.55	79.93
	$F3$	75.51	81.63	75.51	75.51	81.84
	$F4$	79.59	79.59	75.51	81.63	80.07
Pima	$F0$	74.48	73.96	78.13	74.48	78.61
	$F1$	76.04	74.48	77.60	76.04	79.89
	$F2$	76.04	74.48	77.60	76.04	79.89
	$F3$	69.79	71.88	73.96	72.92	75.48
	$F4$	74.48	67.19	78.65	74.48	78.76
Promoter	$F0$	69.23	65.38	88.46	53.85	60.51
	$F1$	73.08	57.69	84.62	80.77	84.62
	$F2$	73.08	57.69	84.62	80.77	84.62
	$F3$	76.92	61.54	76.92	80.77	67.94
	$F4$	80.77	57.69	84.62	76.92	76.53
Waveform	$F0$	74.8	68.96	86.24	76.88	84.32
	$F1$	74.40	75.36	86.88	77.04	86.54
	$F2$	74.40	75.36	86.88	77.04	86.54
	$F3$	74.88	74.88	87.12	79.92	86.37
	$F4$	74.40	76.64	87.12	79.68	86.75

(Continued)

Table 6. *(Continued)*

Data set	Filter	C4.5	1-NN	SVM	PART	EDD
$Winequality-red$	$F0$	53.85	49.88	59.55	51.36	61.16
	$F1$	50.87	48.88	59.80	52.11	61.67
	$F2$	50.87	48.88	59.80	52.11	61.67
	$F3$	50.12	49.63	58.81	52.85	61.75
	$F4$	50.12	49.63	58.81	52.85	61.75
$Yeast$	$F0$	54.84	48.39	55.91	56.72	59.62
	$F1$	53.49	48.92	54.03	54.84	59.82
	$F2$	54.30	45.97	53.76	51.08	54.86
	$F3$	54.03	49.46	54.84	54.30	60.36
	$F4$	54.03	49.46	54.84	54.30	60.36
$Average$	$F0$	69.32	67.70	74.95	68.10	72.21
	$F1$	70.02	68.71	74.22	72.48	76.76
	$F2$	70.11	68.38	74.19	72.06	76.21
	$F3$	70.36	69.62	73.10	71.90	74.33
	$F4$	71.55	67.76	74.48	72.01	75.34

contribution) obtains the best result for five out of nine data sets; SVM yield the highest performance for two data sets and finally 1-NN and PART report the best accuracy once. Furthermore, the EDD reports the highest mean accuracy (76.76 %) followed by the SVM method (74.48 %). Both statements confirm the best behaviour of the product units. EDDFS with F1 is competitive with SVM. According to the results, the application of a preprocessing for SVM is not needed because the behaviour is better than with the reduced data sets; nevertheless, the performance of EDDFS overcomes SVM.

6 Conclusions

This paper presented a framework called EDDFS that combines FS with EDD in the context of product unit neural networks trained with an evolutionary programming approach. The models obtained with the proposal have the advantages that are more accurate and more simple, bearing in mind the number of inputs and/or the number of nodes in the hidden-layer. An empirical study on nine UCI classification problems, that present test error rates about a 15 percent or above with C4.5 or 1-NN classifiers, has been performed to compare EDDFS and EDD methodologies. Nonparametric statistical tests have been applied and the main conclusions achieved are as follows. The FS methods help to improve significantly the accuracy of the models with product units in all cases, although with three out of the four ones the performance is higher in terms of level significance. In regard to the comparison with other classifiers, EDDFS is better than SVM.

Acknowledgments. This work has been partially subsidized by TIN2011-28956-C02-02 and TIN2014-55894-C2-R projects of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P11-TIC-7528 project of the "Junta de Andalucía" (Spain).

References

1. Aggarwal, C.C.: An introduction to data classification. In: Data Classification: Algorithms and Applications, p. 1 (2014)
2. Bache, K., Lichman, M.: UCI machine learning repository online (2015)
3. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
4. Downing, K.L.: Intelligence Emerging: Adaptivity and Search in Evolving Neural Systems. MIT Press, Cambridge (2015)
5. Durbin, R., Rumelhart, D.: Products units: a computationally powerful and biologically plausible extension to backpropagation networks. Neural Comput. **1**(1), 133–142 (1989)
6. Hall, M.A.: Correlation-based feature selection for discrete and numeric class machine learning. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML), pp. 359–366. Morgan Kaufmann, San Francisco (2000)
7. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), vol. 14, pp. 1137–1145 (1995)
8. Kwak, N., Choi, C.-H.: Input feature selection for classification problems. IEEE Trans. Neural Netw. **13**(1), 143–159 (2002)
9. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining, vol. 454. Springer Science & Business Media, New York (2012)
10. Liu, H., Setiono, R.: A probabilistic approach to feature selection - a filter solution. In: Proceedings of the thirteenth International Conference on Machine Learning (ICML), pp. 319–327, Italy. Morgan Kaufmann (1996)
11. Prechelt, L.: Proben1: A set of neural network benchmark problems and benchmarking rules (1994)
12. Quinlan, J.R.: C4.5: Programming for machine learning. Morgan Kauffmann, San Francisco (1993)
13. Rojas, R.: Neural Networks: A Systematic Introduction. Springer Science & Business Media, New York (2013)
14. Ruiz, R., Riquelme, J.C., Aguilar-Ruiz, J.S.: Projection-based measure for efficient feature selection. J. Intell. Fuzzy Syst. **12**(3–4), 175–183 (2002)
15. Ruiz, R., Riquelme, J.C., Aguilar-Ruiz, J.S.: Incremental wrapper-based gene selection from microarray expression data for cancer classification. Pattern Recogn. **39**(12), 2383–2392 (2006)
16. Tallón-Ballesteros, A.J., Gutiérrez-Peña, P.A., Hervás-Martínez, C.: Distribution of the search of evolutionary product unit neural networks for classification. arXiv preprint (2012). arxiv:1205.3336
17. Wang, J.: On the trainability, stability, representability, and realizability of artificial neural networks. Ph.D. thesis, Case Western Reserve University (1991)
18. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Data Management Systems. Morgan Kaufmann, San Francisco (2005)