Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2006) 384–397
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Extracting rules from multilayer perceptrons in classification problems: A clustering-based approach

Eduardo R. Hruschka[a,*], Nelson F.F. Ebecken[b]

[a]Graduate Program in Computer Science, Catholic University of Santos (UniSantos), R. Carvalho de Mendonça, 144, CEP 11.070-906, Santos, SP, Brazil
[b]COPPE/Federal University of Rio de Janeiro, CEP 21.945-970, CP 68.506, Rio de Janeiro, Brazil

Received 8 April 2005; received in revised form 1 August 2005; accepted 31 December 2005
Communicated by A. Abraham
Available online 24 May 2006

Abstract

Multilayer perceptrons adjust their internal parameters performing vector mappings from the input to the output space. Although they may achieve high classification accuracy, the knowledge acquired by such neural networks is usually incomprehensible for humans. This fact is a major obstacle in data mining applications, in which ultimately understandable patterns (like classification rules) are very important. Therefore, many algorithms for rule extraction from neural networks have been developed. This work presents a method to extract rules from multilayer perceptrons trained in classification problems. The rule extraction algorithm basically consists of two steps. First, a clustering genetic algorithm is applied to find clusters of hidden unit activation values. Then, classification rules describing these clusters, in relation to the inputs, are generated. The proposed approach is experimentally evaluated in four datasets that are benchmarks for data mining applications and in a real-world meteorological dataset, leading to interesting results.

Keywords: Rule extraction from neural networks; Clustering; Genetic algorithms

1. Introduction

Neural networks have been successfully applied to solve data mining problems in several domains. In this sense, multilayer perceptrons (MPs) may achieve high classification accuracy, but the knowledge acquired by such neural networks is usually incomprehensible for humans [13]. This fact can be a major obstacle in data mining applications, in which human-interpretable patterns describing the data, like symbolic rules or other forms of knowledge structure, are important [37]. Therefore, many methods have been developed to alleviate the lack of explanation of neural network (NN) models.

Neural networks (NNs) learn by adjusting their connection weights, which somehow reflect the statistical properties of the data [17]. Thus, the knowledge acquired by a NN is codified on its connection weights, which in turn are associated to both its architecture and activation functions [2]. In this context, the process of knowledge acquisition from NNs usually implies the use of algorithms based on the values of either connection weights or hidden unit activations. The algorithms designed to perform such task are generally called *algorithms for rule extraction from neural networks*. The task of rule extraction from NNs is a computationally hard problem [23], and heuristics have been developed to overcome its combinatorial complexity [69]. In our work, a clustering genetic algorithm (CGA) is employed for rule extraction from MPs. The proposed method is based on the hidden unit activation values and consists of two main steps. First, the CGA is employed to find clusters of hidden unit activation values. Then, these clusters are translated into logical rules.

Andrews et al. [2] suggested a classification scheme for rule extraction algorithms. The proposed scheme is based on four aspects: (i) form and quality of the extracted rules; (ii) necessity of specific neural network training algorithms; (iii) complexity of the rule extraction algorithm; (iv) translucency of the neural network. According to this

*Corresponding author.
E-mail addresses: erh@unisantos.br (E.R. Hruschka), nelson@ntt.ufrj.br (N.F.F. Ebecken).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.12.127

scheme, our method provides *If…Then* propositional rules and it does not require any specific MP training algorithm. In addition, it can be applied in classification problems involving discrete and continuous attributes. The rule extraction algorithm complexity is based on the employed CGA. As far as the *translucency* of the NN is concerned, there are three approaches: *decompositional*, *pedagogical and eclectic*. *Decompositional* approaches involve rule extraction at the level of hidden and output units, which are mapped in a binary form. *Pedagogical* approaches try to map inputs directly into outputs, using machine-learning techniques. In our work, hidden unit activation expressions are employed to get classification rules by means of a CGA. Thus, our approach can be classified as *eclectic*, because it is based on both decompositional and pedagogical approaches.

The remainder of the paper is organized as follows. Section 2 situates the proposed method in the context of related work. Section 3 describes the CGA, which is applied to extract rules from MPs trained in classification problems. In Section 4, we present empirical results in four datasets that are benchmarks for data mining (Iris Plants, Wisconsin Breast Cancer, Australian Credit Approval and Pima Indians Diabetes) as well as in a real-world meteorological dataset. Finally, Section 5 concludes our work.

2. Related work

Several methods for rule extraction from NNs have been proposed in the literature, showing the increasing importance of this issue in several domains. Under this perspective, this section provides a brief description of several rule extraction methods. To do so, we follow a chronological order, considering the original work of each author. Then, we present our proposed method, comparing it with similar ones described in the literature.

In 1988, Gallant [19] proposed the first approach to understand the knowledge encoded by NNs. In [19], each NN node represents a conceptual entity, e.g. the input, hidden and output units represent symptoms, diseases, and treatments, respectively. This approach provides NN understanding, but requires available domain knowledge [46]. Also, since redundant rules can appear in the rule extraction process, this method presents the inconvenient necessity of establishing subjective criteria for choosing the appropriate rules [4]. In a later work, Gallant [20] developed the concepts relating NN learning with expert systems.

In 1992, McMillan et al. [43] described a neural network called *RuleNet*, which learns symbolic rules in array manipulation domains. The RuleNet was employed to natural language processing.

In 1993, Towell and Shavlik [68] showed how to use NNs for rule refinement. The developed algorithm was called SUBSET, which is based on the analysis of the weights that make a specific neuron active (considering binary activations). All methods published until 1992 are based on this concept, i.e. in a *decompositional* approach. Also in [68], the authors described the MofN algorithm, which provides more precise and understandable rules than the SUBSET. Thrun [67] described the *validity interval analysis* (VIA) approach, which is based on the propagation of activation values through the network, from the inputs to the outputs and vice versa. Activation ranges are used to prove rule correctness, and the procedure does not require any specific training method for back-propagation NNs. Fu [18] presented a hybrid system that combines symbolic and connectionist inferences. The initial NN architecture is dependent of available domain knowledge and a dataset is employed to refine the rules codified on the NN model. In this sense, the KT Algorithm [16] extracts the refined rules from the trained NN. Kane and Milgram [35] developed a rule extraction method based on numeric and logic-numeric units.

In 1994, Craven and Shavlik [7] explored the fact that trained NNs can be effectively questioned, answering questions about situations in which specific instances are described by a particular class. The rule extraction process is visualized as a learning task, whose main goal is to learn the function computed by the trained NN, providing conjunctive and MofN [68] rules. Sethi and Yoo [55] proposed a method for converting the connection weights in symbolic representations, and showed how to apply it to understand and approximate NNs.

In 1995, Alexander and Mozer [1] developed a rule extraction method, based on connection weights, that supposes activation functions showing approximately Boolean behavior. Andrews et al. [2] presented an important survey of the main approaches developed for rule extraction from NNs until 1995.

In 1996, Narazaki et al. [47] proposed a method to reorganize the knowledge distributed in NNs and to extract approximate fuzzy classification rules. The developed methodology is focused on the analysis of the function learned by a NN. Omlin and Giles [48] proposed a method to insert knowledge in recurrent NNs, showing that these are capable of reviewing the inserted rules. Sethi and Yoo [54] developed a rule extraction method based on the connection weights. Craven and Shavlik [6,61] proposed the TREPAN Algorithm, which does not require neither specific architectures nor particular NN learning algorithms, to extract symbolic representations from NNs. The authors use a classification tree to perform this task. Herrmann [26] described an algorithm to extract rules from logic NNs. Taha and Gosh developed three rule extraction algorithms [64,65] and also a methodology to evaluate the extracted rules [66]. Lu et al. [41] proposed an approach for rule extraction from NNs based on the clustering of hidden unit activation values.

In 1997, Setiono [56] described a method based both on the work of Lu et al. [41] and on an algorithm that eliminates NN redundant connections. Duch et al. [10] proposed two rule extraction algorithms that depend on a

special NN training scheme, which tends to provide NNs formed by few connections. Benítez et al. [5] developed a method to interpret NN models by means of fuzzy rules, showing an equality relation between three-layer perceptrons and fuzzy rule based systems. In this sense, the NN model and the rules provide the same precision rates, but the method is restricted to one-output unit NN models. Weijters et al. [70] proposed an algorithm that trains NNs and automatically extracts classification rules. The algorithm is called BP-SOM and it is based on the clustering, performed by Kohonen networks, of hidden unit activations. Setiono and Liu [59] presented a system to extract oblique decision rules from NNs trained in classification problems. Their simulation results show that the system extracts compact, comprehensible, and high accuracy rules.

In 1998, Das and Mozer [8] proposed a NN learning algorithm that tends to provide a discrete representation, more suitable for rule extraction. In principle, the proposed method was supposed to be used for recurrent networks, but the authors argue that it is possible to employ the same method in other NNs. Duch et al. [9] developed a method to extract fuzzy rules from NNs whose weights belong to the set $\{-1, 0, +1\}$. In addition, the authors proposed a NN training algorithm that allows most weights to take the null value, in a way that the minimal number of rules is obtained. Ludermir [42] suggested a rule extraction process for Boolean networks. In [49,52], mechanisms that compile a NN into an equivalent set of fuzzy rules are proposed. Genetic algorithms are employed to find an appropriate structure of the fuzzy model equivalent with the NN.

In 1999, Duch et al. [12] presented a methodology to extract logical rules. The rules are generated by adding hidden neurons in MPs. In [11] the authors describe how to optimize the extracted rules, converting traditional rules into fuzzy ones. Vahed and Omlin [69] addressed the extraction of knowledge from recurrent NNs trained to behave like deterministic finite-state automata.

In 2000, Duch et al. [13] described a methodology to extract optimized crisp and fuzzy rules, providing interesting results. Keedwell et al. [37] developed a system in which a genetic algorithm is used to search for rules in the NN input space. Setiono and Leow [57] presented the FERNN, a fast method for extracting rules from trained NNs. The method is based on the relevance of the hidden units, considering their information gains. Hayashi et al. [24] reported the results of two rule extraction methods [59,60]) applied to the diagnosis of hepatobiliary disorders. Palade et al. [51] presented a method of rule extraction from NNs that is based on interval propagation across the network, using a procedure of inverting a NN.

Recently, Garcez et al. [21] presented a method to extract non-monotonic rules from NNs formed by discrete input units. Palade et al. [50] developed a method for extracting crisp rules out of NNs, as well as they elaborate on how to transform a NN into a set of fuzzy rules. Snyders and Omlin [63] compared the performance of symbolic rules extracted from NNs trained with and without adaptive bias, giving empirical results for a molecular biology problem. Jiang et al. [34] proposed a method that combines NNs and rule learning. The proposed algorithm utilizes a NN ensemble as the front-end process, which generates abundant training instances for the back-end rule learning process. Setiono et al. [58] presented an approach for extracting rules from NNs trained in regression problems.

In summary, most of the approaches described in the literature have basically two motivations. On the one hand, some authors noticed the need for simplification of neural networks to facilitate the rule extraction process, and are in favor of using specialized training schemes and architectures to perform such task. The assumption underlying these approaches is that neural networks can help the extraction of interesting rules. On the other hand, some papers have proposed algorithms mainly intended to clarify the knowledge encoded in previously trained NNs. This work is mainly focused on the problem of extracting rules from previously trained NNs. In order to elucidate our motivation to adopt such approach, let us suppose a practical situation, e.g. credit card fraud detection, in which a particular NN model has provided better accuracy results than other classifiers. In that situation, it would be interesting to extract rules that *explain* the knowledge acquired by the corresponding neural network model, because comprehensibility is an important issue in data mining applications. In addition, some methods for rule extraction from trained neural networks only require information about the connection weights, whereas others (like the one presented in this paper) also need the training set.

Our work on rule extraction from trained NNs is based on the work of Lu et al. [41], which basically depends on the clustering of hidden unit activation values. The algorithm proposed by Lu et al. [41], called RX Algorithm, computes the following main steps:

(1) Processing training instances through the NN, computing their corresponding hidden unit activation values.
(2) Applying a clustering algorithm in the values obtained in (1), i.e. a clustering algorithm is applied in a transformed space, where the variables (attributes) correspond to the hidden units.
(3) Enumerating the discretized (by means of the obtained clusters) activation values and computing the network outputs. Generate rules that describe the network outputs in terms of the discretized hidden unit activation values.
(4) For each hidden unit, enumerate the input values that lead to them and generate a set of rules to describe the hidden unit discretized values in terms of the inputs.
(5) Merging the sets of rules obtained in steps (3) and (4) to get rules that relate the inputs to the outputs.

The Modified RX Algorithm [30] represents an improvement of its predecessor [41]. The amelioration is based on the concept that instances of a particular class tend to have

similar activation patterns, which, in turn, are generally different from the activation patterns of the instances of other classes. Therefore, the Modified RX Algorithm [30] extracts rules separately for each class. Under a slightly different perspective, this approach allows including information about known classes in an unsupervised technique, making possible to find rules that describe the involved sub-classes. In addition, it also eliminates the extra effort to establish the class corresponding to each cluster. In other words, the clustering process can be improved and the rule extraction process is simplified—because steps (3) and (4) are no more necessary. In summary, the Modified RX Algorithm conceptually consists of two main steps. Initially, a clustering algorithm is applied in a dataset formed by hidden unit activation values (independently in the activations corresponding to the instances of each class). Then, the obtained clusters are translated into logical rules that have the following general form:

$$\textit{If } \{v^1_{\min} \leqslant a_1 \leqslant v^1_{\max} \text{ and} \ldots \text{and } v^n_{\min} \leqslant a_n \leqslant v^n_{\max}\} \text{ then class } C_j. \quad (1)$$

where a_is are the hidden unit activation expressions (as a function of the input units), v^is are the maximum or minimum activation values, obtained from the clusters of each class C_j, and n is the number of hidden units. Such rules provide a way of relating domain regions to the classes, through the clustering of the activation values.

The application of a clustering algorithm in the hidden unit activation values is an interesting approach, because it is simpler to extract rules in a space transformed by these units, which are feature detectors. Hidden units are feature detectors because they try to map classes, which in the original space may not be linearly separable, into the hidden unit activation space, where classes are likely to be linearly separable [25]. From the geometrical point of view, these logical rules correspond to a division of the feature space with hyperplanes perpendicular to the axes, into areas with symbolic names. If the classes are correctly separated with such hyperplanes, accurate logical description of the data is possible and worthwhile [13]. In addition, since linear transformations are usually adopted for scaling the input data, our methodology tends to provide good results in terms of the comprehensibility of the extracted rules, because in such cases the hidden unit activation expressions are also linear functions of the original attributes (input variables). More specifically, each hidden unit (in the first layer) of a MP can create a hyperplane, because if the input transformations are linear, then the activation functions are written by

$$a_k = \sum_{j=1}^{j=m} w_{kj} x_j, \quad (2)$$

where a_k is the kth hidden unit, m is the number of attributes, w_{kj} are the synaptic weights, and x_js are the attributes. Thus, although NNs can map non-linear functions, the rule extraction process is performed in a space where the attributes were linearly transformed by the connection weights. In practice, the use of linear transformations for scaling the inputs is very common, but we believe that it is also possible to get satisfactory results in other cases, approximating non-linear functions by linear ones.

The main disadvantage of the Modified RX Algorithm [30] involves the lack of optimization of the clustering process, mainly in relation to the number of clusters, which must be defined a priori. Consequently, neither the number nor the quality (in terms of accuracy) of the extracted rules is optimized. In order to circumvent these problems, we propose to employ a genetic algorithm, especially designed for clustering problems, that allows optimizing both the number and the quality of the extracted rules. The next section describes the CGA used for rule extraction from MPs trained in classification problems.

3. Clustering Genetic Algorithm (CGA)

Clustering is a task in which one seeks to identify a finite set of categories (clusters) to describe a given data set, both maximizing homogeneity within each cluster and heterogeneity among different clusters. In other words, instances that belong to the same cluster should be more similar to each other than instances that belong to different clusters. Thus, it is necessary to devise means of evaluating the similarities among instances. This problem is usually tackled indirectly, i.e. distance measures are used to quantify the dissimilarity between two instances, in such a way that more similar instances have smaller dissimilarity measures. Several dissimilarity measures can be employed for clustering tasks, such as the commonly used Euclidean and Manhattan distances [36].

There are three main types of clustering techniques: overlapping, hierarchical and partitioning. This work considers that clustering involves the partitioning of a set $\mathbf{X}$ of instances into a collection of mutually disjoint subsets $\boldsymbol{C}_i$ of $\mathbf{X}$. Formally, let us consider a set of N instances $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ to be clustered, where each $\mathbf{x}_i \in \Re^\rho$ is an attribute vector consisting of ρ real-valued measurements. The instances must be clustered into non-overlapping groups $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_k\}$ where k is the number of clusters, such that:

$$\boldsymbol{C}_1 \cup \boldsymbol{C}_2 \cup \ldots \cup \boldsymbol{C}_k = \mathbf{X},\ \boldsymbol{C}_i \neq \varnothing,$$
$$\text{and } \boldsymbol{C}_i \cap \boldsymbol{C}_j = \varnothing \text{ for } i \neq j. \quad (3)$$

Many methods described in the literature assume that k is given by the user [36]. These methods search for k clusters according to a predefined criterion. Doing so, the number of ways (NW) of classifying N instances into k clusters is [40]:

$$\mathrm{NW}(N,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^N. \quad (4)$$

Thus, there are a large number of possible partitions even for moderate N and k (e.g. $NW(100,5) \approx 56.6 \times 10^{67}$), and the complete enumeration of every possible partition is simply not possible [14]. In other words, it is not easy to find the best partitioning even assuming that k is known. Indeed, this is rarely the case in practice. A usual approach is to run a clustering algorithm several times and, based on the obtained results, choose the value for k that provides the most *natural* clustering. This strategy assumes domain knowledge and usually has the disadvantage of searching for the best solution in a small subset of the search space. Consequently, these methods have, in general, low probabilities of success. Another alternative involves optimizing k according to numeric criteria. In this case, k is unknown and the number of ways of grouping N instances into k clusters, considering S different scenarios (each one resulting from a different k), is [40]:

$$\sum_{S} NW(\mathrm{N}, \mathrm{k}). \tag{5}$$

The problem of finding an optimal solution to the partition of N data into k clusters is NP-complete [53] and, provided that the number of distinct partitions of N instances into k clusters increases approximately as $k^N/k!$, attempting to find a globally optimum solution is usually not computationally feasible [3]. This difficulty has stimulated the search for efficient approximation algorithms. Genetic algorithms are widely believed to be effective on NP-complete global optimization problems and they can provide good sub-optimal solutions in reasonable time [53]. Under this perspective, the Clustering Genetic Algorithm (CGA), whose main steps are summarized in Fig. 1, was introduced in [32] and is here reviewed. In [32], randomly generated initial populations were used in order to show the efficacy of the CGA operators. In the current work, we use a more suitable methodology (to be described in Section 3.5) to set up the initial population.

3.1. Encoding scheme

The CGA [32] is based on a simple encoding scheme. Let us consider a dataset formed by N instances. Then, a genotype is an integer vector of $(N+1)$ positions. Each position corresponds to an instance, i.e., the ith position (gene) represents the i-th instance, whereas the last gene represents the number of clusters (k). Thus, each gene has a value over the alphabet $\{1,2,3,\ldots,k\}$. For instance, in a dataset composed of 20 instances, a possible genotype is:

Genotype 1 : 22345123453321454552 5.

In this case, five instances {1,2,7,13,20} form the cluster whose label is 2. The cluster whose label is 1 has 2 instances {6,14}, and so on.

Standard genetic operators are usually not suitable for clustering problems for several reasons [15,32]. First, the encoding scheme presented above is naturally redundant, i.e., the encoding is *one-to-many*. In fact, there are $k!$ different genotypes that represent the same solution. For example, there are 5! different genotypes that correspond to the same clustering solution represented by *Genotype* 1. Some of them are:

Genotype 2 : 33451234514432515113 5,
Genotype 3 : 44512345125543121224 5,
Genotype 4 : 55123451231154232335 5,
Genotype 5 : 11234512342215343441 5.

Thus, the size of the search space the genetic algorithm has to search is much larger than the original space of solutions. This augmented space may reduce the efficiency of the genetic algorithm. In addition, the redundant encoding also causes the undesirable effect of casting context-dependent information out of context under the standard crossover, i.e., equal parents can originate different offspring. For example, if *Genotype* 2 was crossed-over with *Genotype* 3, the resulting genotypes should be equal to their parents, since they represent the same solution to the clustering problem. Unfortunately, however, that rarely happens. In order to make this situation more easily observed, let us apply a two-point crossover, indicated by two vertical lines, to the following genotypes, which represent the same clustering solution:

Genotype 6 : 2|**2222**|111113333344444 4,
Genotype 7 : 4|**4444**|333335555511111 4.

The genotypes of the resulting offspring are given by:

Offspring 1 : 2**4444**111113333344444 4,
Offspring 2 : 4**2222**333335555511111 5.

These offspring are different from their parents; that is, they represent different clustering solutions. In addition, *offspring 2* represents five clusters, whereas their parents represent just four. An alternative to solve these problems is to employ a renumbering algorithm [28,29,31]. However, this is not enough, because when traditional genetic algorithms are employed in clustering problems, they usually just manipulate gene values without taking into account their connections with other genes. Indeed, the interconnections among gene values (represented by cluster labels) constitute the genuine optimization goal in clustering problems. For this reason, the development of genetic operators specially designed to clustering problems has been investigated [15,32]. In this context, the CGA operators proposed in [32] are of particular interest since

1. Initialize a population of random genotypes;
2. Evaluate each genotype according to its silhouette (Section 3.4);
3. Apply a linear normalization (ranking);
4. Select genotypes by proportional selection;
5. Apply crossover and mutation;
6. Replace the old genotypes by the ones formed by step 5;
7. If convergence is attained, stop; else, go to step 2.

Fig. 1. Main steps of the CGA [32].

they operate on constant length chromosomes. These cluster-oriented (i.e. context-sensitive) operators are described in the next sections.

3.2. Crossover operator

The CGA crossover operator combines clustering solutions coming from different genotypes. It works in the following way. First, two genotypes (**G1** and **G2**) are selected. Then, assuming that **G1** represents k_1 clusters, the CGA randomly chooses $c \in \{1, 2, \ldots, k_1\}$ clusters to copy into **G2**. The unchanged clusters of **G2** are maintained and the changed ones have their instances allocated to the corresponding nearest clusters (according to their centroids, i.e. mean vectors). In this way, an offspring **G3** is obtained. The same procedure is employed to get an offspring **G4**, but now considering that the changed clusters of **G2** are copied into **G1**. To illustrate this procedure, let us consider the following two genotypes, in which the last gene (number of clusters) is not represented:

G1 – 1123245125432533424,

G2 – 1212332124423221321.

For example, let us assume that clusters 2 and 3 of **G1** (below in boldface) were randomly selected:

G1 – 11**232**45125**432533**424.

When these clusters are copied into **G2**, they modify clusters {1,2,3} of **G2**, whereas cluster 4 is not modified:

G2 – 12 **232** 321 **24432** 2 **33** 3 **2** 1.

The underlined positions, which correspond to those genes indirectly affected by clusters 2 and 3 of **G1**, are now changed to 0 (zero):

G3 – 0023200024432033020.

The genes set equal to zero will then be placed into the nearest clusters (according to their centroids) already encoded by **G3**. The same procedure is used to get an offspring **G4**, except for the selected clusters to be copied into **G1**. These clusters are now the clusters of **G2** initially changed for deriving offspring **G3**, namely {1,2,3}. Note that, although the crossover operator produces offspring usually formed by a number of clusters that is neither smaller nor larger than the number of clusters of their parents, this operator is able to increase or decrease the number of clusters, as shown in the example above.

3.3. Mutation operators

Two operators for mutation are used in the CGA. The first operator works only on genotypes that encode more than two clusters. It eliminates a randomly chosen cluster, placing its instances into the nearest remaining clusters (according to their centroids). The second operator divides a randomly selected cluster into two new ones. The first cluster is formed by the instances closer to the original centroid, whereas the other cluster is formed by those instances closer to the farthest instance from the centroid.

3.4. Objective function

The objective function is based on the silhouette [36]. To explain it, let us consider an instance i belonging to cluster **A**. So, the average dissimilarity of i to all other instances of **A** is denoted by $a(i)$. Now let us take into account cluster **B**. The average dissimilarity of i to all instances of **B** will be called $d(i,\mathbf{B})$. After computing $d(i,\mathbf{B})$ for all clusters $\mathbf{B} \neq \mathbf{A}$, the smallest one is selected, i.e. $b(i) = \min d(i,\mathbf{B})$, $\mathbf{B} \neq \mathbf{A}$. This value represents the dissimilarity of i to its neighbor cluster, and the silhouette $s(i)$ is given by

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}. \tag{6}$$

It is easy to verify that $-1 \leqslant s(i) \leqslant 1$. Thus, the higher $s(i)$ the better the assignment of instance i to a given cluster. In addition, if $s(i)$ is equal to zero, then it is not clear whether the instance should have been assigned to its current cluster or to a neighboring one [14]. Finally, if cluster **A** is a singleton, then $s(i)$ is not defined and the most neutral choice is to set $s(i) = 0$ [36]. The objective function is the average of $s(i)$ over $i = 1, 2, \ldots, N$ and the best clustering is achieved when its value is maximized.

The calculation of distances between two instances represents the main computational cost of the CGA. Therefore, the computational cost of CGA is estimated as $O(N^2)$, where N is the number of instances to be clustered.

3.5. Initial population

Randomly generated initial populations were used in [32] to show the efficacy of the CGA operators. In the present work, in order to accelerate the CGA convergence, we use a methodology similar to the one described in [36] to set up the initial population. The initial clustering process is based on the selection of representative instances. The first selected instance is the most centrally located in the set of instances. Subsequently, at each step another instance is selected by means of the following steps:

1. Consider an instance i that has not been selected.
2. Consider a non-selected instance j and calculate the difference between its dissimilarity D_j with the most similar previously selected instance and its dissimilarity $d(j, i)$ with instance i.
3. If the difference is positive, j contributes to the decision to select i and $C_{ji} = \max\{D_j - d(j, i), 0\}$ is calculated.
4. Calculate the total gain obtained by selecting instance i: $\sum_j C_{ji}$
5. Choose the not yet selected instance i that maximizes the total gain calculated in step 4.

This process is continued until r representative instances have been found. In fact, our algorithm stops when the total gain is not positive anymore, because in this case the selection of instance i is not appropriate. The algorithm developed in [36] has a second phase to improve the set of representative instances. This is done by considering all pairs of instances (i,h) for which instance i has been selected and instance h has not. Basically, it is determined what effect is obtained on the value of the clustering when a swap is carried out, i.e. when instance i is no longer selected as a representative instance, but instance h is. As all potential swaps are considered, this process may not be suitable for data mining applications. In addition, we are not ultimately searching for representative instances, but for good partitions, which will be translated into classification rules. Indeed, our method aims at searching for the best clustering in a space where it is more likely to be found, and this process is carried out by the CGA.

After selecting the representative instances (*cluster seeds*) the initial population is formed considering that the non-selected instances are clustered according to their proximity with the representative ones. Considering that r representative instances are selected, the first genotype represents two clusters, the second genotype represents three clusters,..., and the last one represents r clusters. Thus, the initial populations are formed by (r-1) genotypes, which, in turn, represent the *starting point* for the genetic search.

3.6. Selection and settings

The genotypes corresponding to each generation are selected according to the roulette wheel strategy [22], which does not admit negative objective function values. For this reason, a constant equal to one is summed up to the objective function before the selection procedure takes place. It is important to mention that the objective function values reported in the description of our experiments are the original ones (defined in Section 3.4). The highest fitness genotype is always copied into the succeeding generation. In addition, the CGA does not employ crossover and mutation probabilities, i.e., the operators are necessarily applied to some selected genotypes after the selection procedure is performed. Particularly, 50% of the selected genotypes are crossed-over, 25% are mutated by operator 1 and 25% are mutated by operator 2. The sensitivity of the CGA in relation to these parameters will be addressed in a future work.

4. Experimental evaluation

The proposed method was evaluated by means of experiments in five datasets. The first case studied is a pedagogical example, which shows how our method works. To do so, we used the Iris Plants dataset, which is a well-known data mining benchmark. In the sequel, we describe experiments performed in three datasets that are also data mining benchmarks — Wisconsin Breast Cancer, Australian Credit Approval, and Pima Indians Diabetes — and in a real-world meteorological dataset. The benchmark datasets are available at the UCI Repository [44]. As previously mentioned, our work is mainly focused on the problem of extracting rules from previously trained neural networks. Thus, several neural networks were first trained. Then, our rule extraction method was applied in the best achieved MP model. This way, we simulated real applications for our method, whose preliminary empirical results were published in [27,33].

In the pedagogical experiment, all instances were employed for training, whereas in the other datasets (Wisconsin Breast Cancer, Australian Credit Approval, Pima Indians Diabetes and Meteorological) we used a cross-validation method [25,62], in which 60% of the data were used for training, 20% for validation (to stop training), and 20% for testing. Stratified samples were used in all the experiments, i.e. the class proportion was maintained in the training, validation and test datasets. Considering MPs architectures, we used [0,1] and [−1,1] linear transformations for scaling the input data, the hyperbolic tangent and the logistic functions in the hidden units, and three functions in the output units: logistic, hyperbolic tangent and linear. In each dataset, we trained 40 MPs architectures, varying the number of hidden units from 1 to 5. We just employed one-hidden layer neural networks, with standard connections. The MPs were trained by a backpropagation algorithm, for several learning rates and momentum terms. The initial weights were randomly chosen from the range $[-0.3, +0.3]$.

Considering the CGA, we used the Euclidean and Manhattan distances [36] in order to calculate the dissimilarities among the instances (choosing the one that provides the *best* rule set in terms of accuracy) and the CGA was stopped after 2000 generations.

4.1. Iris Plants dataset—pedagogical illustration

This dataset is formed by 150 instances. There are three classes (Setosa, Versicolour and Virginica), each one represented by 50 instances. The class Setosa is linearly separable from the others, whereas the classes Versicolour and Virginica are not linearly separable. Four attributes (sepal and petal length and width) describe each instance. However, we did not employ these original attributes in this pedagogical illustration. Instead, sepal and petal areas were used as attributes (variables). The sepal area is obtained by multiplying the sepal length by the sepal width and the petal area is calculated in an analogous way. These *artificial* attributes provide classification results very similar to those obtained by means of the original ones. In addition, the sepal and petal areas allow plotting interesting graphs, facilitating the understanding of the main concepts underlying our method.

First, in Section 4.1.1, we illustrate a typical clustering task, applying the CGA in all instances. Then, in Section 4.1.2, we elucidate the process of rule extraction from

trained MPs. By comparing the results obtained in these sections, we show how the hidden units activation space can facilitate the extraction of interesting patterns (in this case classification rules), showing situations in which the proposed rule extraction method can be useful. It is important to observe that only in this pedagogical example we do not extract rules separately for each class, i.e. we compare the results obtained by the CGA in the original space of attributes (sepal and petal areas) with those achieved in the hidden units activation space. Thus, as a secondary contribution, these results help visualizing the potential of a hybrid system, in which a MP transforms the attribute space, favoring the extraction of better classification rules (some related works about this issue were addressed in Section 2).

4.1.1. *Applying the CGA in the dataset formed by petal and sepal areas*

This section reports the results of the CGA application in the Iris instances, considering the sepal and petal areas as attributes. The class labels were not used in the clustering process. The CGA employed a population formed by 72 genotypes (the number of genotypes is determined automatically, according to the method described in Section 3.5). Our *best* solution (Fig. 2) was obtained after 68 generations, using the Manhattan distance. The corresponding objective function value (OFV) was equal to 0.49, and 88% of the instances were correctly classified. The *right* solution for this problem (i.e. the clusters formed by the instances of the three known classes) is represented in Fig. 3. The CGA did not find this solution because its corresponding OFV is equal to 0.39, less than the OFV found by the CGA, i.e. 0.49. It happens because there are overlapping clusters. In this context, the next section shows how the hidden unit activation space facilitates the recognition of each class.

4.1.2. *Applying the CGA in the hidden units activation space*

This section illustrates how our method for rule extraction from MPs works. Basically, MPs can transform the original space of attributes into another one (hidden unit activation space), where accurate classification rules, which describe the knowledge encoded in a trained MP,

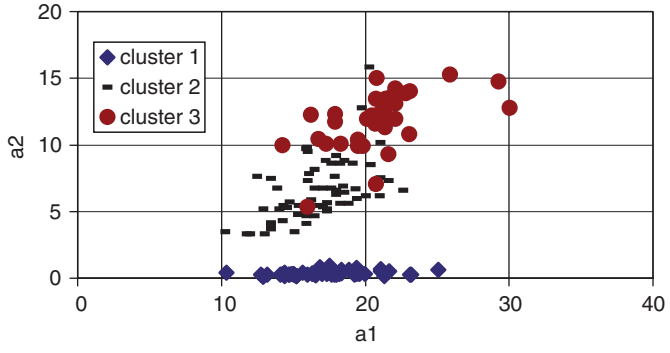


Fig. 2. Clustering using the Manhattan distance.

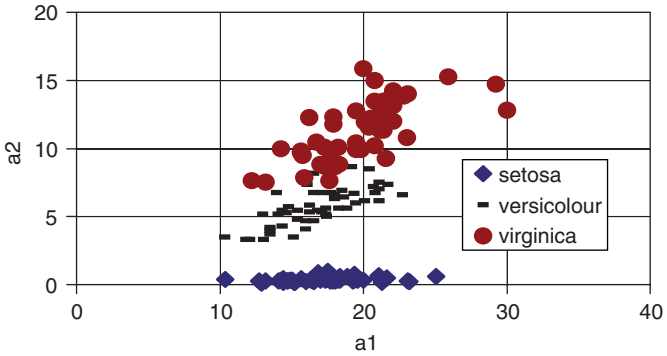


Fig. 3. Right clustering provided by the known classes.

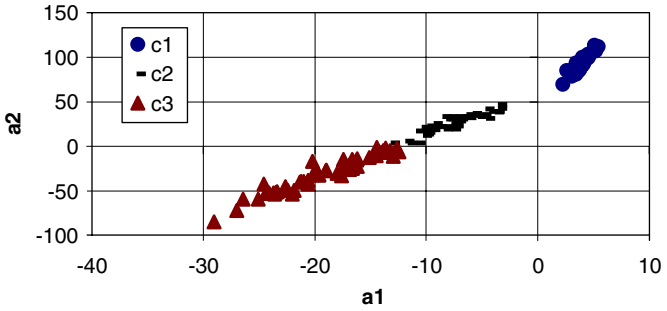


Fig. 4. Clusters of activation values.

can be extracted. In this way, it is possible to elucidate why our method for rule extraction from MPs can provide good results.

Our best obtained MP model is formed by linear $[-1,+1]$ transformations for scaling the input data, two hyperbolic hidden units, and two logistic output units. After 15,000 epochs of training, 98% of the instances are correctly classified. The hidden unit activation values for each training instance are computed according to the following expressions:

$$a_1 = 0.22A_s - 2.16A_p + 0.84, \quad (7)$$

$$a_2 = 3.12A_s - 11.91A_p + 41.99, \quad (8)$$

where A_s and A_p are the sepal and petal areas, respectively. The CGA is then applied in the dataset formed by these activation values. The *best* solution, whose OFV is 0.72, was found in the first generation. Fig. 4 shows the corresponding clusters, which are translated into the following rules:

If ($2.27 \leqslant a_1 \leqslant 5.43$ and $69.64 \leqslant a_2 \leqslant 113.09$) then C1—Setosa.
If ($-13.09 \leqslant a_1 \leqslant -3.46$ and $3.47 \leqslant a_2 \leqslant 46.54$) then C2—Versicolour.
If ($-29.04 \leqslant a_1 \leqslant -12.46$ and $-84.56 \leqslant a_2 \leqslant -1.14$) then C3—Virginica.

These rules classify correctly 96% of the instances, providing an *explanation* of the *knowledge* encoded in the

neural network model. Under a different perspective, these rules can also be viewed as an approximation of the function learned by the corresponding MP, which classified 98% of the instances correctly.

The CGA application in the activation values provided better results than in the space formed by the sepal and petal areas. As previously mentioned, it happens because it is simpler to extract patterns in the space transformed by the hidden units. In the next sections, we describe the application of the proposed method in its full form, extracting rules separately for each class and using samples for training, testing, and validation.

4.2. Wisconsin Breast Cancer dataset

The instances of this dataset are formed by nine attributes (A, B,…I) and an associated class label (benign or malignant). The classes are not linearly separable. The total number of instances is 699 (458 benign and 241 malignant), of which 16 have a single missing feature. We removed those 16 instances and used the remaining ones in our experiments. Three stratified samples were randomly extracted from the original database: 60% to train MPs and consequently to extract rules, 20% for validation (to stop training), and 20% for testing. Thus, our training set is formed by 409 instances (266 benign and 143 malignant). The validation and test sets are formed by 137 instances (89 benign and 48 malignant). Our best MP is formed by nine linear [0,1] input units, two logistic hidden units, and two logistic output units. This model correctly classifies 97.81% and 97.08% of the validation and test instances respectively, and the hidden unit activation expressions are:

$$a_1 = 8.34 - 0.51\mathrm{A} - 0.23\mathrm{B} - 0.40\mathrm{C} - 0.43\mathrm{D} - 0.04\mathrm{E} - 0.61\mathrm{F} - 0.12\mathrm{G} - 0.24\mathrm{H} - 0.27\mathrm{I}, \quad (9)$$

$$a_2 = -5.52 + 0.36\mathrm{A} - 0.23\mathrm{B} + 0.15\mathrm{C} + 0.12\mathrm{D} + 0.10\mathrm{E} + 0.19\mathrm{F} + 0.23\mathrm{G} + 0.12\mathrm{H} + 0.32\mathrm{I}. \quad (10)$$

Expressions 9 and 10 are used to calculate the hidden unit activation values for each training instance. Then, the activation values are separated according to their class label, and the CGA is applied to find clusters that correspond to sub-classes. The Euclidean distance provided better results than the Manhattan distance. Considering the benign class, the CGA found 81 representative instances (80 genotypes) and converged, after 128 generations (OFV = 0.81), into a solution formed by two clusters: C4 (260 instances) and C5 (6 instances). Considering the malignant class, the CGA found 70 representative instances and converged, after 66 generations (OFV = 0.52), into a solution formed by two clusters: C6 (99 instances) and C7 (44 instances). The following rules describe these clusters:

If $\{-3.43 \leqslant a_1 \leqslant 5.49$ and $-4.17 \leqslant a_2 \leqslant -0.02\}$ then C4 (benign).
If $\{-8.14 \leqslant a_1 \leqslant -5.06$ and $-0.97 \leqslant a_2 \leqslant 2.58\}$ then C5 (benign).
If $\{-19.44 \leqslant a_1 \leqslant -7.30$ and $0.21 \leqslant a_2 \leqslant 7.18\}$ then C6 (malignant).
If $\{-7.69 \leqslant a_1 \leqslant 0.54$ and $-1.88 \leqslant a_2 \leqslant 2.74\}$ then C7 (malignant).

These rules classify correctly 89.78% and 85.40% of the validation and test sets, respectively. An important consideration involves the cluster C5, which contains only 1.47% of the training instances. This proportion is a measure analogous to the *support of a rule*, used to evaluate the interestingness of association rules [39]. In this context, some methodologies consider that a rule is interesting if its support is greater than the user specified minimum support [38]. Let us here suppose that the minimum support is equal to 5%. In this case, the rule described by C5 may be considered not interesting and the following rules would be obtained:

If $\{-3.43 \leqslant a_1 \leqslant 5.49$ and $-4.17 \leqslant a_2 \leqslant -0.02\}$ then benign.
If $\{-19.44 \leqslant a_1 \leqslant -7.30$ and $0.21 \leqslant a_2 \leqslant 7.18\}$ then malignant.
If $\{-7.69 \leqslant a_1 \leqslant 0.54$ and $-1.88 \leqslant a_2 \leqslant 2.74\}$ then malignant.

These rules classify correctly 97.81% and 96.35% of the validation and test instances, respectively, whereas the corresponding rates for the MP model are 97.81% and 97.08%. In other words, the rule obtained from cluster C5 is too general, i.e. it covers a relatively large percentage of cases from the *wrong* class. If a *default* class is considered, our best result is obtained by the two rules of the malignant class. These rules correctly classify 98.54% and 97.81% of the validation and test sets, respectively. Thus, we believe that such rules do not significantly improve the classification results. Furthermore, rules based on a *default class* are less comprehensible (in this case they only describe the malignant instances).

Duch et al. [13] reported several empirical results regarding this dataset. The experiments described in [13] were performed considering a 10-fold cross-validation process, providing average correct classification rates (ACCRs) that vary from 92.7% to 99%. It suggests that our results are comparable to the best ones described in the literature. These comparisons may be biased by the use of different samples. However, in our experiments, the ACCR in the validation set was similar to the one achieved in the test set, suggesting that the rules have good generalization capability. In addition, Duch et al. [13] also argue that the stratified 10-cross validation test is difficult to perform since rules have to be extracted many times from different neural network models. Moreover, if this process is employed, it is impossible not only to present a single set of rules, but also to compare rules extracted by different methods. Under this perspective, the results described in

the literature can be considered as benchmarks to evaluate the efficacy of our method.

In several data mining tasks, the extracted rules are evaluated not only in terms of accuracy, but also in terms of their comprehensibility. In this sense, the complexity of the extracted rules mainly depends on the architecture of the trained neural network. In practical applications, optimized neural networks are usually available. Considering our experiments, we believe that it would be possible to get simpler rules, using more sophisticated neural network optimization techniques.

4.3. Australian credit approval database

This dataset concerns about credit card applications and it contains 690 instances—307 of people who were granted credit (class zero) and 383 of people who were not granted credit (class one)—formed by 14 attributes (A, B,..., N) and the class label. There are six continuous attributes and eight categorical ones. Our best MP correctly classifies 87.68% and 76.82% of the validation and test instances respectively. This model is formed by 14 linear [0,1] input units, one logistic hidden unit, and two logistic output units. The corresponding hidden unit activation expression is:

$$a_1 = -0.58\mathrm{A} - 0.02\mathrm{B} + 0.089\mathrm{C} + 1.23\mathrm{D} + 0.3\mathrm{E} - 0.086\mathrm{F} + 0.026\mathrm{G} + 7.02\mathrm{H} + 3.27\mathrm{I} + 0.034\mathrm{J} - 0.02\mathrm{K} - 0.065\mathrm{L} - 0.003\mathrm{M} + 0.00007\mathrm{N} - 8.88. \quad (11)$$

Considering the class *zero*, the CGA used 113 genotypes and found the *best* clustering in the first generation (OFV = 0.67), suggesting that the genetic search started in an exceptionally good initial population. The corresponding obtained clusters are here called C8 (170 instances) and C9 (59 instances). For the class *one*, the CGA employed 86 genotypes and found the *best* clustering after 28 generations (OFV = 0.75), corresponding to clusters C10 (169 instances) and C11 (14 instances). This solution provides the following rule set:

If $(-9.86 \leqslant a_1 \leqslant -2.83)$ then C8—class zero.
If $(-2.65 \leqslant a_1 \leqslant 7.24)$ then C9—class zero.
If $(-2.23 \leqslant a_1 \leqslant 9.27)$ then C10—class one.
If $(-9.87 \leqslant a_1 \leqslant -4.01)$ then C11—class one.

In the validation set, these rules provide an ACCR equal to 7.19%, whereas in the test set just 3.6% of the instances are correctly classified. These low ACCRs can be explained by examining the rules. Considering the two rules of class *zero*, a_1 belongs to the range [−9.86,7.24], whereas in the two rules of class *one* it belongs to the range [−9.87, 9.27]. Thus, both classes are modeled by very similar activation values, i.e. the set of rules is too general. However, if only clusters C8 (contains 74% of the *class zero* instances) and C10 (contains 92% of *class one* instances) are considered, one gets ACCRs equal to 86.33% and 76.26% in the validation and test sets respectively. Default rules have not significantly improved classification results. In this sense, if cluster C8 is used as *default,* the ACCRs in the validation and test sets are equal to 87.05% and 76.97%, respectively. Considering cluster C10 as default, ACCRs equal to 86.33% and 76.62% are obtained in the validation and test sets, respectively. In summary, the best obtained rules are:

If $(-9.86 \leqslant a_1 \leqslant -2.83)$ then *class zero.*
If $(-2.23 \leqslant a_1 \leqslant 9.27)$ then *class one*.

These rules describe 82% of the instances in the training set, classifying correctly 86.33% and 76.26% of the validation and test sets, respectively, whereas the corresponding rates for the MP model are 87.68% and 76.82%. Several results for this dataset are reported in [45], in which the ACCRs vary from 79.30% to 86.90%, suggesting that our method provided good results.

4.4. Pima Indians dataset

This is a very difficult classification problem [45]. The dataset has 768 instances—500 represent negative conditions for diabetes (class zero) and 268 represent positive conditions (class one). Each instance is formed by eight attributes (A,B,...,H) and the class label. The best obtained MP correctly classifies 70.78% and 70.13% of the validation and test instances respectively. This model is formed by eight linear $[-1,+1]$ input units, two logistic hidden unit and two logistic output units. The hidden unit activation values were calculated by expressions 12 and 13.

$$a_1 = 44.61 - 0.43\mathrm{A} - 0.17\mathrm{B} + 0.04\mathrm{C} + 0.16\mathrm{D} - 0.03\mathrm{E} - 0.51\mathrm{F} - 7.60\mathrm{G} + 0.02\mathrm{H}, \quad (12)$$

$$a_2 = 47.34 - 0.15\mathrm{A} - 0.14\mathrm{B} + 0.01\mathrm{C} - 0.03\mathrm{D} + 0.04\mathrm{E} - 0.54\mathrm{F} - 2.59\mathrm{G} - 0.68\mathrm{H}. \quad (13)$$

The CGA used 151 genotypes and found the *best* clustering after 63 generations (OFV = 0.62), corresponding to clusters C12 (298 instances) and C13 (2 instances) for class *zero*. Considering the class *one*, the CGA employed 81 genotypes and found the *best* clustering after 51 generations (OFV = 0.44), corresponding to clusters C14, C15 and C16, formed by 78, 75, and 7 instances, respectively. The achieved solution corresponds to the following rule set:

If $\{-13.19 \leqslant a_1 \leqslant 36.09$ and $-41.09 \leqslant a_2 \leqslant 21.40\}$ then C12 (*zero*).
If $\{-39.07 \leqslant a_1 \leqslant -23.71$ and $2.08 \leqslant a_2 \leqslant 6.90\}$ then C13 (*zero*).
If $\{-15.65 \leqslant a_1 \leqslant 11.28$ and $-40.76 \leqslant a_2 \leqslant -11.06\}$ then C14 (*one*).
If $\{-4.21 \leqslant a_1 \leqslant 28.00$ and $-19.64 \leqslant a_2 \leqslant 5.02\}$ then C15 (*one*).

If $\{-24.20 \leqslant a_1 \leqslant -11.09$ and $-6.65 \leqslant a_2 \leqslant 6.89\}$ then C16 (*one*).

These rules provide ACCRs equal to 12.98% and 11.69% in the validation and test sets respectively. However, if class *one* is taken as default, the rule set classifies correctly 66.88% and 66.23% of the validation and test sets, respectively, whereas the corresponding rates for the MP model are 70.78% and 70.13%. Several results for this dataset are described in [45], in which the ACCRs vary from 67.60% to 77.70%. The results achieved by means of a default rule were satisfactory, but this example also allows foreseeing situations in which our proposed method may have problems to extract the knowledge encoded in the neural network model. Fig. 5 shows the hidden unit activation values, where one can see overlapping clusters, which prevents the CGA from finding good classification rules. In other words, as already observed in [17,47,68], distributed representations make the rule extraction process difficult.

4.5. Meteorological dataset

This real-world dataset was collected at the International Airport of Rio de Janeiro, and it is formed by 17,443 instances. Each instance is described by 38 attributes and the class label, which represents weather conditions. Instances with missing values were eliminated, remaining 1226 instances. There are seven involved classes: normal, cloudy, dry fog, wet fog, drizzle, rain, and lightning, described by 302, 34, 125, 512, 67, 152 and 34 instances respectively. Several MPs were trained in order to get satisfactory results in terms of the average error rate. The process of training MPs indirectly allowed performing feature selection. In summary, the *best* MP is formed by six $[-1,+1]$ linear input units, four logistic hidden units and seven logistic output units. The selected attributes were: x_1(visibility), x_2(dew temperature), x_3(pressure), x_4(wet bulb temperature), x_5(precipitation) and x_6(air relative humidity). After 638 training epochs, this model correctly classifies 78.78% and 78.37% of the validation and test instances, respectively. The hidden unit activation expressions are:

$$a_1 = -22.02 + 0.0125x_1 + 0.0671x_2 + 0.0026x_3 - 0.0067x_4 - 0.00535x_5 - 0.03x_6, \quad (14)$$

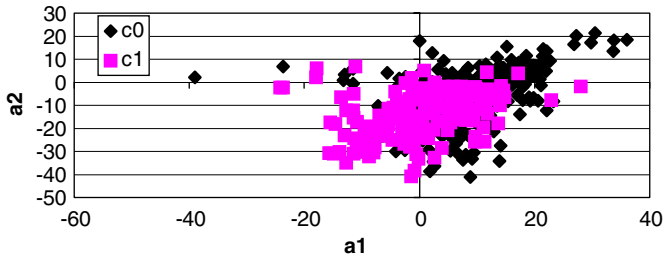


Fig. 5. Hidden unit activation values—Pima Indians Diabetes dataset.

$$a_2 = -28.33 - 0.0014x_1 + 0.4571x_2 - 0.0029x_3 - 0.059x_4 + 0.02827x_5 + 0.44636x_6, \quad (15)$$

$$a_3 = 17.08 - 0.0081x_1 + 0.0771x_2 + 0.00034x_3 + 0.0215x_4 - 0.0026x_5 - 0.15636x_6, \quad (16)$$

$$a_4 = 10.02 - 0.0015x_1 + 0.1228x_2 + 0.00809x_3 + 0.0077x_4 - 0.0297x_5 - 0.15848x_6. \quad (17)$$

Wet fog conditions are very important to airport operation and, consequently, are their corresponding classification rules. Therefore, we focused on obtaining rules to describe conditions in which wet fog occurs. Two clusters (OFV = 0.46, 53 generations) of activation values were found to this class. The obtained clusters are formed by 104 and 202 instances, respectively corresponding to the following rules:

If $\{-22.8 \leqslant a_1 \leqslant -11.5$ and $0.3 \leqslant a_2 \leqslant 12.7$ and $0.3 \leqslant a_3 \leqslant 9.4$ and $-3.3 \leqslant a_4 \leqslant 2.5\}$ then wet fog.
If $\{-13.0 \leqslant a_1 \leqslant 0.7$ and $-1.9 \leqslant a_2 \leqslant 17.1$ and $-8.2 \leqslant a_3 \leqslant 1.8$ and $-10.0 \leqslant a_4 \leqslant 1.3\}$ then wet fog.
Else another weather condition.

These rules classify correctly 72.18% and 71.37% of the validation and test sets, respectively, whereas the corresponding rates for the MP model are 78.78% and 78.37%. Considering the complexity involved in meteorological phenomena, good rules were extracted from the MP model.

4.6. Summary of empirical results

We evaluated our method for rule extraction from MPs in four datasets that are benchmarks for data mining methods and in a real-world meteorological dataset. Experiments in the Iris Plants dataset elucidated how the proposed method works. Our evaluations were focused on the ACCRs of the extracted rules, and Table 1 summarizes these results. Considering the benchmark datasets (Iris Plants, Australian Credit Approval, Pima Indians Diabetes and Wisconsin Breast Cancer), our method provided results comparable to the best ones described in the literature. The rules extracted from the MP trained in the meteorological dataset were also interesting, mainly in the light of the complexity involved in meteorological phenomena. In general, the ACCRs of the extracted rules were similar to those achieved by the corresponding MPs, thus suggesting good approximations of the functions learned by MPs. In addition, in our experiments we did not employ sophisticated techniques to optimize MPs training. In this sense, better MP models may provide better rules. Experiments in the Pima Indians Diabetes dataset showed an example of a problematic situation that can be faced by our method. In this case, distributed representations complicated the rule extraction process, as it was already observed in other works [17,47,68].

Table 1
Average correct classification rates

Dataset	Multilayer perceptrons (MPs)	Rules (CGA)	Literature
Iris Plants	98.00%*	96.00%*	95.7–100% [13]
Wisconsin Breast Cancer	97.81%^{v} 97.08%^{t}	97.81%^{v} 96.35%^{t}	92.7–99% [13]
Australian Credit Approval	87.68%^{v} 76.82%^{t}	86.33%^{v} 76.26%^{t}	79.3–86.9% [45]
Pima Indians Diabetes	70.78%^{v} 70.13%^{t}	66.88%^{v} 66.23%^{t}	67.6–77.7% [13,45]
Meteorological	78.78%^{v} 78.37%^{t}	72.18%^{v} 71.37% ^{t}	—

^{v}validation set; ^{t}test set; *training set.

5. Conclusions

Neural networks usually provide high classification accuracy. However, the knowledge acquired by such models is generally incomprehensible for humans. This fact is a major obstacle in data mining applications, in which ultimately understandable patterns (like classification rules) are very important. Therefore, many algorithms for rule extraction from neural networks have been developed. This paper described a method that employs a CGA to extract rules from MPs trained in classification problems. The rule extraction algorithm basically consists of two steps. First, the CGA is applied to find clusters of hidden unit activation values. Then, logical rules that describe these clusters, in relation to the inputs, are generated.

The CGA automatically optimizes the number of clusters of activation patterns, which, in turn, are translated into logical rules. Consequently, our method allows optimizing the number of extracted rules. This characteristic is important, because hundreds of logical rules produced by some algorithms may provide an opaque description of the data and, therefore, are not more comprehensible than any black-box classification system. The proposed method requires neither a particular MP training algorithm nor a specific MP architecture. In addition, since linear transformations are usually adopted for scaling the input data, our methodology tends to provide good results in terms of the comprehensibility of the extracted rules, because in such cases the hidden unit activation expressions are also linear functions of the original attributes (input variables). However, we believe that it is also possible to get reasonable results in other cases, approximating non-linear functions by linear ones.

We evaluated the efficacy of the proposed method by means of experiments in four benchmark datasets (Iris Plants, Wisconsin Breast Cancer, Australian Credit Approval and Pima Indians Diabetes), and in a real-world meteorological dataset. The experiments performed in the benchmark datasets showed that our method provided results comparable to the best ones found in the literature. The rules extracted from the MP trained in the meteorological dataset were also interesting, mainly in the light of the complexity involved in meteorological phenomena. In general, the extracted rules provided good approximations of the functions learned by MPs. Our future work will focus on the assessment of the proposed method in applications in which optimized MP models are available.

Acknowledgments

We are grateful to the Brazilian Research Agencies CNPq, FAPESP, and FAPERJ for their financial support. We would also like to thank Dr. Ricardo J. G. B. Campello and Dr. Leandro N. de Castro for their valuable suggestions on making Section 3 more readable.

References

[1] J.A. Alexander, M.C. Mozer, Template-based algorithm for connectionist rule extraction, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems, vol. 7, MIT Press, Cambridge, MA, 1995.

[2] R. Andrews, J. Diederich, A.B. Tickle, A. Survey, Critique of techniques for extracting rules from trained artificial neural networks, Knowledge Based Sys. 8 (6) (1995) 373–389.

[3] P. Arabie, L.J. Hubert, An Overview of Combinatorial Data Analysis, in: P. Arabie, L.J. Hubert, G. DeSoete (Eds.), Clustering and Classification, World Scientific, Singapore, 1999 (Chapter 1).

[4] R. Baron, Knowledge extraction from neural networks: A survey, in: Report no. 94-17, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 1994.

[5] J.M. Benítez, J.L. Castro, I. Requena, Are artificial neural networks black boxes?, IEEE Trans. Neural Networks 8 (5) (1997) 1156–1164.

[6] M.W. Craven, J.W. Shavlik, Extracting tree-structured representations of trained networks, Advances in Neural Information Processing Systems, vol. 8, MIT Press, Cambridge, MA, 1996, pp. 24–30.

[7] M.W. Craven, J.W. Shavlik, Using sampling and queries to extract rules from trained neural networks, in: Proceedings of the 11th International Conference on Machine Learning, San Francisco CA, 1994.

[8] S. Das, M. Mozer, Dynamic on-line clustering and state extraction: an approach to symbolic learning, Neural Networks 11 (1) (1998) 53–64.

[9] W. Duch, R. Adamczak, K. Grabczewski, Extraction of logical rules from training data using backpropagation networks, Neural Process. Lett. 7 (1998) 1–9.

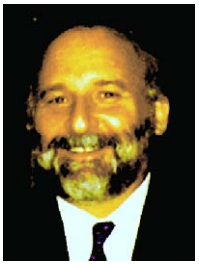
[10] W. Duch, R. Adamczak, K. Grabczewski, M. Ishikawa, H. Ueda, Extraction of crisp logical rules using constrained backpropagation networks, in: Proceedings of the European Symposium on Artificial Neural Networks, Bruge, 1997.

[11] W. Duch, R. Adamczak, K. Grabczewski, Optimization of Logical Rules Derived by Neural Procedures, in: Proceedings of the International Joint Conference on Neural Networks, Washington. DC, July, 1999.

[12] W. Duch, R. Adamczak, K. Grabczewski, G. Zal, Hybrid neural-global minimization method of logical rule extraction, J. Adv. Comput. Intell. 3 (1999) 348–356.
[13] W. Duch, R. Adamczak, K. Grabczwski, A new methodology of extraction, optimization and application of crisp and fuzzy logical rules, IEEE Trans. Neural Networks 11 (2) (2000) 1–31.
[14] B.S. Everitt, S. Landau, M. Leese, Cluster Analysis, Arnold Publishers, London, 2001.
[15] E. Falkenauer, Genetic Algorithms and Grouping Problems, Wiley, New York, 1998.
[16] L. Fu, Rule generation from neural networks, IEEE Trans. Syst. Man Cybern 24 (8) (1994) 1114–1124.
[17] L. Fu, Neural Networks in Computer Intelligence, first ed, McGraw-Hill, New York, USA, 1994.
[18] L. Fu, Knowledge-based connectionism for revising domain theories, IEEE Trans. Syst. Man Cybern. 23 (1) (1993) 173–182.
[19] S.I. Gallant, Connectionist expert systems, Commun. ACM 31 (2) (1988) 152–169.
[20] S.I. Gallant, Neural Network Learning and Expert Systems, MIT Press, Cambridge, MA, 1994.
[21] A.S.D. Garcez, K. Broda, D.M. Gabbay, Symbolic knowledge extraction from trained neural networks: a sound approach, Artif. Intell. 125 (2001) 155–207.
[22] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Longman, 1989.
[23] M. Golea, On the complexity of rule-extraction from neural networks and network-querying, in: Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop, University of Sussex, Brighton, UK, 1996, pp. 51–59.
[24] Y. Hayashi, R. Setiono, K. Yoshida, A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders, Artif. Intell. Med. 20 (2000) 205–216.
[25] S.S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, Englwood Eliffs, NJ, 1998.
[26] C. Hermann, A. Their, Backpropagation for neural DNF- and CNF-networks, Technical Report, FG Intellektik, TH Darmstadt, Germany, 1996.
[27] E.R. Hruschka, N.F.F. Ebecken, A clustering genetic algorithm for extracting rules from multilayer perceptrons trained in classification problems. In: A. Zanasi, et al. (ed.), Data Mining, vol. III, Southampton,UK, 2002. pp. 451–460.
[28] E.R. Hruschka, N.F.F. Ebecken, A clustering genetic algorithm for extracting rules from supervised neural network models in data mining tasks, Int. J. Comput. Syst. Signals (IJCSS) 1 (1) (2000) 17–29 (Special Issue: Knowledge Discovery from Structured and Unstructured Data).
[29] E.R. Hruschka, N.F.F. Ebecken, Applying a clustering genetic algorithm for extracting rules from a supervised neural network, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Como, Italy, 2000.
[30] E.R. Hruschka, N.F.F. Ebecken, Rule extraction from neural networks: modified RX algorithm, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Washington DC, USA, 1999.
[31] E.R. Hruschka, N.F.F. Ebecken, Using a clustering genetic algorithm for rule extraction from artificial neural networks, in: Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, San Antonio, TX, USA, 2000.
[32] E.R. Hruschka, N.F.F. Ebecken, A genetic algorithm for cluster analysis, Intell. Data Anal. 7 (1) (2003) 15–25.
[33] E.R. Hruschka, N.F.F. Ebecken, Rules from supervised neural network in data mining tasks. In: J.M. Abbe, J.I. da Silva Filho, (eds.), Frontiers in Artificial Intelligence and Applications, vol. 71, Amsterdam, 2001, pp. 84–100.
[34] Y. Jiang, Z. Zhou, Z. Chen, Z., Rule learning based on neural network ensemble, in: Proceedings of the International Joint Conference on Neural Networks, Honolulu, 2002, pp. 1416–1420.
[35] R. Kane, M. Milgram, Extraction of semantic rules from trained neural networks, in: Proceedings of the ICNN, San Francisco, CA, 1993.
[36] L. Kaufman, P. J. Rousseeuw, Finding Groups in Data—An Introduction to Cluster Analysis, Wiley Series in Probability and Mathematical Statistics, 1990.
[37] E. Keedwell, A. Narayanan, D. Savic, Creating rules from trained neural networks using genetic algorithms, Int. J. Comput. Syst. Signals (IJCSS), 1 (1) (2000) 30–42.
[38] B. Liu, W. Hsu, Ma Yiming, Pruning and summarizing the discovered associations, in: Proceedings of the ACM International Conference on Knowledge Discovery & Data Mining, San Diego, CA, USA, 1999, pp. 125–134.
[39] B. Liu, W. Hsu, S. Chen, Ma Yiming, Analyzing the Subjective Interestingness of Association Rules, IEEE Intell. Syst. (2000) 47–55.
[40] G.L. Liu, Introduction to Combinatorial Mathematics, McGraw Hill, New York, 1968.
[41] H. Lu, R. Setiono, H. Liu, Effective data mining using neural networks, IEEE Trans Knowledge Data Eng 8 (6) (1996) 957–961.
[42] T.B. Ludermir, Extracting Rules from Feedforward Boolean Neural Networks, in: Proceedings of the V Brazilian Symposium on Neural Networks, 1998, pp. 61–65.
[43] C. McMillan, M.C. Mozer, P. Smolenski, Rule Induction through Integrated Symbolic and Subsymbolic Processing, in: J.E. Moody, S.J. Hanson, R.P. Lippmann (Eds.), Advances in Neural Processing Systems, vol. IV, Morgan Kaufmann Publishers, Los Allos, 1992, pp. 969–976.
[44] C. J. Merz, P. M. Murphy, UCI repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine. www.ics.uci.edu.
[45] G. Nakhaeizadeh, U. Kressel, S. Keh, et al., Dataset descriptions and results, in: D. Michie, D.J. Spiegelhalter, C.C. Taylor (Eds.), Machine Learning, Neural and Statistical Classification, Ellis Horwood Series in Artificial Intelligence, Bookcraft, Midsomer Norton, 1994, pp. 131–174.
[46] H. Narazaki, I. Shigaki, T. Watanabe, A Method for Extracting Approximate Rules from Neural Network, in: Proceedings of the Fourth IEEE International Conference on Fuzzy Systems and Second International Fuzzy Engineering Symposium, Yokohama, Japan, 1995.
[47] H. Narazaki, T. Watanabe, M. Yamamoto, Re-organizing knowledge in neural networks: an explanatory mechanism for neural networks in data classification problems, IEEE Trans. Syst. Man Cybern. 26 (1) (1996) 107–117.
[48] C.W. Omlin, C.L. Giles, Rule revision with recurrent neural networks, IEEE Trans Knowledge Data Eng 8 (1) (1996) 183–188.
[49] V. Palade, S. Bumbaru, M.G. Negoita, A method for compiling neural networks into fuzzy rules using genetic algorithms and hierarchical approach, in: Proceedings of the Second IEEE International Conference on Knowledge-Based Intelligent Electronic Systems 2 (1998), pp. 353–358.
[50] V. Palade, D. Neagu, R.J. Patton, Interpretation of trained neural networks by rule extraction, in: Bernd Reusch (Ed.), Lecture Notes in Computer Science (LNCS 2206), Computational Intelligence: Theory and Applications, Springer, Berlin, 2001, pp. 152–161.
[51] V. Palade, D. Neagu, G. Puscasu, Rule extraction from neural networks by interval propagation, in: Proceedings of the Fourthh IEEE International Conference on Knowledge-Based Intelligent Engineering Systems, Brighton-UK, 2000, pp. 217–220.
[52] V. Palade, M. G. Negoita, V. Ariton, Genetic algorithm optimization of knowledge extraction from neural networks, in: Proceedings of the Sixth IEEE International Conference on Neural Information Processing (ICONIP99), vol. 2, Perth, Australia, 1999, pp. 752–758.
[53] Y. Park, M. Song, A genetic algorithm for clustering problems, in: Proceedings of the Genetic Programming Conference, University of Wisconsin, July, 1998.
[54] I. Sethi, J. Yoo, Multi-valued logic mapping of neurons in feed-forward networks, Eng Intell. Syst. 4 (4) (1996) 243–253.

[55] I. Sethi, J.H. Yoo, Symbolic approximation of feedforward neural networks, in: E.S. Gelsema, L.N. Kanal (Eds.), Pattern Recognition in Practice vol. IV, North-Holland, Amsterdam, 1994, pp. 313–324.
[56] R. Setiono, Extracting rules from neural networks by pruning and hidden-unit splitting, Neural Computation 9 (1) (1997) 205–225.
[57] R. Setiono, K. Leow, FERNN: an algorithm for fast extraction of rules from neural networks, Appl. Intell. 12 (1–2) (2000) 15–25.
[58] R. Setiono, W.K. Leow, J.M. Zuarada, Extraction of rules from artificial neural networks for nonlinear regression, IEEE Trans. Neural Networks 13 (3) (2002) 564–577.
[59] R. Setiono, H. Liu, Neurolinear: from neural networks to oblique decision rules, Neurocomputing 16 (3) (1997) 225–235.
[60] R. Setiono, H. Liu, Symbolic representation of neural networks, IEEE Comput 29 (3) (1996) 71–77.
[61] J. Shavlik, An overview of research at wisconsin on knowledge-based neural networks, in: Proceedings of the International Conference on Neural Networks, Washington, DC, 1996, pp. 65–69.
[62] M. Smith, Neural networks for statistical modeling, first ed, International Thomson Computer Press, USA, 1996.
[63] S. Snyders, C. Omlin, Rule Extraction from Knowledge-Based Neural Networks with Adaptive Inductive Bias, in: Proceedings of the Eighth International Conference on Neural Information Processing (ICONIP), vol. 1, 2001, pp. 143–148.
[64] I. Taha, J. Ghosh, Symbolic interpretation of artificial neural networks, IEEE Trans. Knowledge Data Eng. 11 (3) (1999) 448–463.
[65] I. Taha, J. Ghosh, Three techniques for extracting rules from feedforward networks, Intelligent Engineering Systems Through Artificial Neural Networks, vol. 6, ASME Press, New York, 1996.
[66] I. Taha, J. Ghosh, Evaluation and ordering of rules extracted from feedforward networks, in: Proceedings of the IEEE International Conference on Neural Networks, 1997, pp. 221–226.
[67] S. Thrun, Extracting provably correct rules from artificial neural networks, Technical Report IAI-93-5, Institut für Informatik III, Universität Bonn, 1993.
[68] G.G. Towell, J. Shavlik, Extracting refined rules from knowledge-based neural networks, Mach. Learn. 13 (1993) 71–101.
[69] A. Vahed, C.W. Omlin, Rule extraction from recurrent neural networks using a symbolic machine learning algorithm, in: Proceedings of the Sixth International Conference on Neural Information Processing, Perth, Australia, 1999.
[70] T. Weijters, A.V.D. Bosh, J.V.D. Herik, Intelligible neural networks with BP_SOM, in: Proceedings of the Ninth Dutch Conference on Artificial Intelligence, 1997, pp. 27–36.

Eduardo Raul Hruschka received his B.Sc. degree in Civil Engineering from Federal University of Paraná, Brazil, in 1995, and his M.Sc. and Ph.D. degrees in Computational Systems from Federal University of Rio de Janeiro, Brazil, in 1998 and 2001, respectively. He is currently assistant professor at Catholic University of Santos (UniSantos), Brazil. His main research interest is data mining, with particular emphasis on evolutionary algorithms, artificial neural networks, clustering algorithms, feature selection, and missing values imputation.



Nelson Francisco Favilla Ebecken is Professor of Computational Systems at COPPE/UFRJ, the Engineering Graduated Center of Federal University of Rio de Janeiro. His research focuses on basic methodologies for modeling and extracting knowledge from data and their application across different disciplines. He develops and integrates ideas and computational tools from statistics and information theory with artificial intelligence paradigms.