INFORMATICA, 2007, Vol. 18, No. 2, 163–186

Neuro-IG: A Hybrid System for Selection and Elimination of Predictor Variables and non Relevant Individuals

Baghdad ATMANI[1,2], Bouziane BELDJILALI[2]
[1]*Leibniz Laboratory, IMAG*
46 Av Felix Viallet, 38031 Cedex Grenoble, France
e-mail: baghdad.atmani@imag.fr
[2]*Department of Computer Science, Faculty of Science, University of Oran*
BP 1524 El M'Naouer 31000 Oran, Algeria
e-mail: baghdad.atmani@univ-oran.dz, bouziane.beldjilali@univ-oran.dz

Received: June 2006

Abstract. In this article we present the general architecture of a hybrid neuro-symbolic system for the selection and stepwise elimination of predictor variables and non-relevant individuals for the construction of a model. Our purpose is to design tools for extracting the relevant variables and the relevant individuals for an automatic training from data. The objective is to reduce the complexity of storage, therefore the complexity of calculation, and to gradually improve the performance of ordering, that is to say to arrive at a good quality training.

Key words: hybrid system, neural network, automatic training, pruning, symbolic system, rule extraction.

1. Introduction

Data mining (Lefebure and Venturi, 2001) is the core part of knowledge discovery in database (KDD) (Fayyad *et al.*, 1996). The KDD usually has to process a huge volume of data in order to extract knowledge units (rules) that are non trivial, potentially useful, significant, and reusable. The KDD is at the cross point of several areas: database technology (Bentayeb and Darmont, 2002; Chaudhuri, 1998; Netz *et al.*, 2000), artificial intelligence (Fujimoto and Nakabayashi, 2003; Liao *et al.*, 1999; Montgomery *et al.*, 1997; Ramesh *et al.*, 2004), machine learning (Freitag, 2000; Ishibuchi *et al.*, 2001; Quinlan, 1993), statistic analysis (Hastie *et al.*, 2001; Fukunaga, 1990; Montgomery *et al.*, 1997), fuzzy logic (Bingchiang *et al.*, 1997; Olaru and Wehenkel, 2003), artificial neural networks (Craven and Shavlik, 1997; Fan and Li, 2002; Lisboa *et al.*, 2002; Lu *et al.*, 1996) and induction decision tree (Breiman *et al.*, 1984; Cantu-Paz and Kamath, 2003; Crémilleux, 2000; Herr *et al.*, 1997; Kohavi and Quinlan, 2002). The KDD process can be described as consisting of the following five steps (Kodratoff, 1997; Lee and Sian, 2001):

- understanding the domain;

- preparing the data set;
- discovering patterns (data mining);
- presenting the discovered patterns;
- and putting the results into use.

Generally, the KDD process is iterative and interactive, and controlled by an expert of the data domain, called the analyst, who is in charge of guiding the extraction process, based upon his objectives and his domain knowledge. The analyst selects and interprets a subset of the knowledge units for building models that will be further considered as knowledge units with a certain plausibility. These units are in turn embedded within a representation formalism to be used by a knowledge-based system. The KDD process is performed within a KDD system that is composed of databases, symbolic and/or numerical data mining modules, and interfaces for interactions with the system. Closing the loop, the knowledge units extracted by the KDD system must be represented in an adequate formalism, so that they may be integrated within the ontology and reused for problem-solving needs in application domains such as agronomy, biology, chemistry, medecine...

The extraction process is based on data mining methods for returning knowledge units from the considered data. Data mining methods can be either symbolic or numerical:

- Symbolic methods include, among others (Zighed *et al.*, 2002; Quinlan, 1986): classification based on decision trees, lattice-based classification, association rule extraction, classification based on rough sets, learning methods, e.g., induction, instance-based learning, explanation-based learning, and database methods based on information retrieval and query answering;
- Numerical methods include (Bologna and Pellegrini, 1996; D'Avila Garcez *et al.*, 2001; Kurfess, 2000), among others: statistics and data analysis, hidden markov models, bayesian networks, neural networks, genetic algorithms;

Within the framework of a medical project, like (Duhamel *et al.*, 2001; Lee *et al.*, 2002; Liao and Lee, 2002) and (Setiono, 1996), we have been submitted a problem concerning diabetic patients, with the purpose of designing an automatic system for identification of the diabetes types. In order to propose a first automatic model of prediction of the various diabetes types, we tried knowledge extraction using several methods of data mining (Denis and Gilleron, 1999; Zighed and Rakotomalala, 2000). We have experimented, using data mining platforms SIPINA-V2.5 (Zighed, 1996) and TANAGRA (Rakotomalala, 2005b), the regression model, discriminating analysis, a multi-layer perceptron, three methods relying upon induction graphs (ID3, C4.5, SIPINA) and the system CHARADE (Ganascia, 1988). Through a comparison of the results of the various methods, it appears that induction graphs, and in particular the SIPINA method, show better rates of classification with a low number of rules (Duhamel *et al.*, 2001; Sebban *et al.*, 2002).

Classification, which involves finding rules that partition a given data set into disjoint groups, is one class of data mining problems. Data items in databases, such as tuples in relational database systems usually represent real world entities. The values of the attributes of a tuple represent the properties of the entity. Classification is the process of finding the common properies among differents entities and classifying them into classes.

The results of the process are often expressed in the form of rules. In the following, we are interested in symbolic and numerical KDD methods based on a classification approach, more precisely on decision trees and neural netwoks.

Decision trees are a widely used technique and occupy a strategic place in the field of machine learning (Hastie *et al.*, 2001). Many empirical comparisons (Lim *et al.*, 2000) show that decision trees often reach performances that are comparable to other supervised methods. Methods based on decision trees are nonparametric; they do not require any assumption on the distribution of the data; they are resistant to input data; their model of prediction is nonlinear and when the training base is large, they show similar properties to closest neighbors algorithms. Finally the strong point which ensures the popularity of decision trees is their ability to produce a simple, directly usable knowledge in the form of rules (Rakotomalala, 2005a). On the other hand, neural netwoks give a lower classification error rate than the decision trees but require longer learning time and the knowledge generated by neural netwoks is not explicitly represented in the form of rules suitable for verification or interpretation by humans (Setiono and Liu, 1995).

Nevertheless the main drawback that can be identified in induction by decision trees is their inability, with the traditional algorithms (e.g., SIPINA, ID3, C4.5, CART, CHAID), to detect the relevant predictor variables and examples. Another weakness is the need for having a sample of training of large size. In addition, as shown by (Castellano and Fanelli, 2000) in the area of selection of variables and non-relevant individuals, the preliminary reduction of the descriptors in strongly disturbed fields significantly improves the performances of the decision trees. In such processes, data pre-processing (Erray, 2001; Jensen *et al.*, 2002; Lee *et al.*, 2002; Zighed *et al.*, 1998) is important and machine learning techniques (Castellano *et al.*, 2002; Duch *et al.*, 1999; Ishibuchi *et al.*, 2001; Nurnberger, 2003) can be used to achieve automated data set preparation: problem of selecting relevant variable and examples.

Our contribution within the framework of this real application is to design a neural network able to learn and prepare the data set (select the relevant variables and examples) starting from pratical examples. After, data selection, data cleaning and data transformation, a multilayered neural network is used to select and stepwise eliminate the predictor variables and the non-relevant individuals before launching the SIPINA method.

This paper is organized as follows. We start by reviewing in Section 2 the classification approach commonly used in machine learning. Our Neuro-IG system is outlined in Section 3. We describe in detail the neural network training and pruning. Finally, Section 4 shows experimental results for checking the validity of the proposed system. Section 5 summarizes the paper and suggest further perspectives of this work.

2. Chosen Approach for Data Mining

Classification is the most widely studied data mining task. The problem with classification from examples is to let this process induce only correct classification procedures from correctly classified data. For example, from a set of data about patients treated for diabetes, correctly stored by physicians, the machine should determine diagnosis rules that

are applicable to new patients in order to determine if they are, or not, insulin dependent. Once validated, these rules can be inserted into a running system. In the classification, each data instance (or database record) belongs to a class, which is indicated by the value of a goal attribute. This attribute can take a small number of discrete values, each of them corresponding to a class. Each instance consists of two parts, namely a set of predictor attribute values and a goal attribute value. The former are used to predict the value of the later.

Note that the predictor attributes should be relevant for predicting the class (goal attribute value) of a data instance. For example, if the goal attribute indicates whether or not a patient has or will develop a certain disease, the predictor attributes should contain medical information relevant for this prediction, and no irrelevant attributes such as the name of the patient. In a context of diabetic patients monitoring (Duhamel *et al.*, 2001; Liao and Lee, 2002; Setiono, 1996), for example, setting up tools for accident detection is not possible without also taking into account the necessary role of the physician. The aim is to design a system for assisted monitoring and diagnosis that will provide specialists with the necessary information for identifying the diabetes type of patients.

Since classification is studied in many different disciplines, there is a wide variety of terminology in use to describe the basic elements of this task. For example, a data instance can be called an example, an object, a case, a record, or a tuple. An attribute can be called a variable or a feature. In this paper we will use mainly the terms tuple and variable. The diabetic patient example will be used all along the paper.

Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ be the population of diabetic patients taken into account for the classification. A variable is associated with this population, called goal variable or class attribute, denoted C.

A class $C(\omega)$ can be associated with every individual ω. The goal variable C takes its values in the set IC of class identifiers.

$$C\colon \Omega \to IC = \{c_1, c_2, \ldots, c_m\}, \quad \omega_i \mapsto C(\omega_i) = c_j.$$

In the example, the aim is diagnosing a possible insulin dependence. This will be designated by a goal variable $C\colon \Omega \to \{1, 2\}$, where class 1 contains all insulin dependent patients (***diabetes of Type 1***), and class 2 the non dependent patients (***diabetes of Type 2***). The objective is to define a function φ for predicting the class C, thus the detection of possible accidents. The determination of the prediction model φ, which is the goal of the classification, is bound to the hypothesis that the values taken by the goal variable C are not at random, but depend upon certain individual situations, called predictor variables that are determined by the expert.

The predictor variables can be qualitative or quantitative. The values of the predictor variables concerning an individual $\omega \in \Omega$ constitute a tuple:

$$X(\omega) = \big(X_1(\omega), X_2(\omega), \ldots, X_p(\omega)\big).$$

So, the predictor variables is a function:

$$X\colon \Omega \to IM,$$

where IM is the set of all possible tuples.

The value taken by $X_j(\omega)$ is called the modality or the value of the variable X_j for ω. We denote by l_j the number of different values taken by the variable X_j. In order to illustrate this form of notations, let us consider the medical problem of the identification of diabetes types. The patient is described, for example, by three predictor variables: Viral Infection, which can take on the values Yes or No; Association with auto-immune illness, which can take on the values Relation or no Relation; and Sex, which can take on the values Feminine or Masculine.

The population Ω of diabetic patients taken account for the training consist of a pairs composed of predictor variables and their corresponding goal values for the prediction model φ of diabetes type.

- $((Yes, Relation, Feminine), Type1or2)$
- $((Yes, NoRelation, Masculine), Type1or2)$
- $((No, Relation, Masculine), Type1or2)$
- ...

In our case the predictor variables are summarized in Table 1.

In the classification task the set of individuals Ω being mined is randomly divided into two mutually exclusive and exhaustive subsets Ω_a and Ω_t, called the training set and the test set. The training set Ω_a is made entirely available to the data mining algorithm, so that the algorithm has access to the values of both predictor variables and the goal variable for each individual in Ω_a. The aim of the data mining algorithm is to discover a relationship φ between the predictor variables and the goal variable using Ω_a.

The discovered relationship φ is then used to predict the class (goal-variable value) of all the individuals in the test set Ω_t. Note that, from the viewpoint of the algorithm, the test set contains unknown-class individuals. Only after a prediction is made the algorithm can have access to the actual class of the just-classified individual. If the class predicted

Table 1

Predictor variables, semantics and values

Predictor Var	**Semantics**	**Values**
X_1	Seniority	> 35; $\geqslant 15$ and < 30; unspecified
X_2	How revealed	Spontaneous; Infectious; Glycemy unbalance; Recent
X_3	Weight	Normal; Skinny; Obese; Overweight
X_4	Viral Infection	Yes; No
X_5	State	Weight loss; No Weight loss
X_6	Association	Relation with auto-immune illness; No relation
X_7	Circumstance of discovery	Diabetic feet; Fortuitous; Infection; Retinopathy; Comas; Inaugural; Cetosic coma
X_8	Asthenia	Yes; No
X_9	Antecedent	Family; Personal; No Antecedent
X_{10}	Sex	Feminine, Masculine

by the algorithm is the same as the actual class of the tuple, this is considered as a correct prediction. If the class predicted by the algorithm is different from the actual class of the tuple, this is a wrong prediction. We denote by Ω_e the set of individuals in Ω_t not correctly classified during the test of the prediction function φ. One of the aims of the data mining algorithm is to maximize the classification accuracy rate in the test set, which is simply the number of correct predictions divided by the total number (correct+wrong) of predictions.

Following (Kodratoff, 1997; Zighed and Rakotomalala, 2000), the discovered relationship φ produced by the machine, also called output of the classification, is not necessarily of a logical nature (Duch *et al.*, 1999), it can take various forms: neural networks, algebraic models, geometric models, decision trees, etc.

The general learning process that our hybrid system applies once to a data set is organized in five stages:

- acquisition and transformation of data;
- data pre-processing using a neural networks (selection of relevant predictor variables and individuals);
- symbolic classification for producing a prediction model;
- validation of the extracted knowledge;
- design of an operational hybrid system for data pre-processing and prediction.

All along this paper, we take as an example the aid to medical diagnosis (Atmani and Beldjilali, 2003).

We define:

Ω_a^{NN} – learning sample for training the neural network;

Ω_e^{NN} – individuals in Ω_t^{NN} not correctly classified during the test of the numeric training by neural network;

Ω_a^{SS} – learning sample for symbolic training by induction graph;

Ω_e^{SS} – individuals in Ω_t^{SS} not correctly classified during the test of the symbolic training by induction graph.

Initially: $\Omega_a^{NN} = \Omega_a$, $\Omega_t^{NN} = \Omega_t$ and $\Omega_e^{NN} = \{\}$.

3. Neuro-IG: Neuro-Induction Graph System

After acquisition and transformation of data, Neuro-IG consists of the following steps:

- numeric training by a neural network and minimization of connections;
- symbolic training by an induction graph and generation of rules;
- validation and generalization.

Fig. 1 summarizes the general diagram of our hybrid system, where as stands for satisfactory training, and ss stands for satisfactory threshold.

First stage:

While $(as > \beta_1)$ *train by the neural network with* Ω_a^{NN}

if $(as \leqslant \beta_1)$ *then launch the validation phase of the neural network with* Ω_t^{NN} *and generate* Ω_e^{NN}.

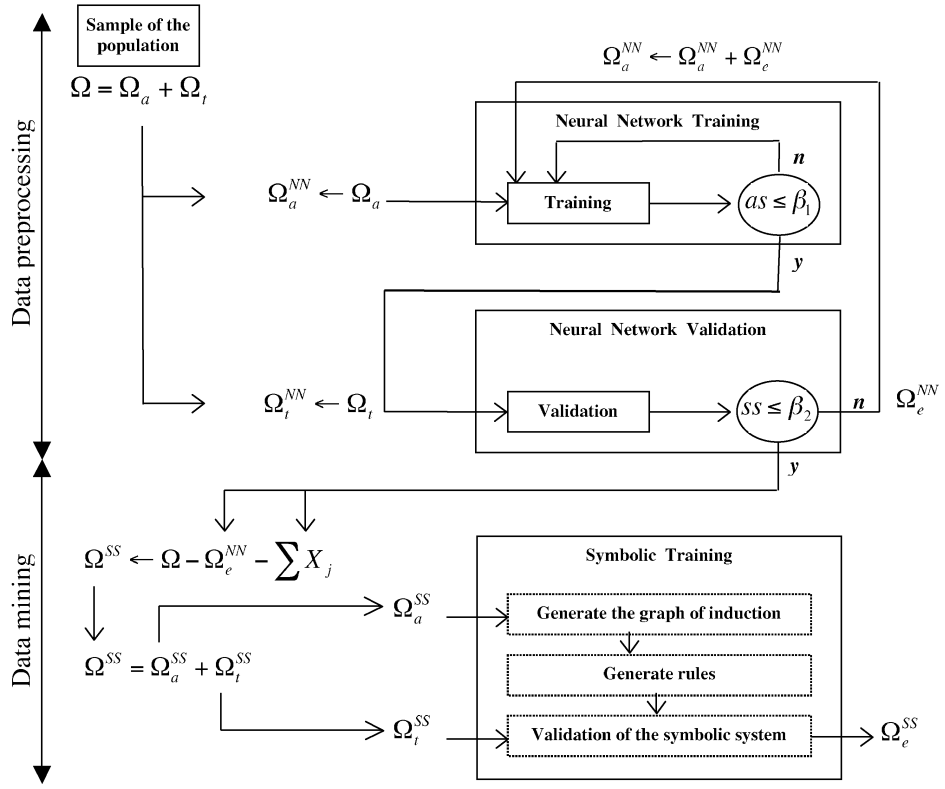

Fig. 1. General diagram of Neuro-IG.

Second stage:

if $(ss > \beta_2)$ *then repeat the first stage while using* $\Omega_a^{NN} \leftarrow \Omega_a^{NN} + \Omega_e^{NN}$,

if $(ss \leqslant \beta_2)$ *then transmit parameters to the symbolic system, while proposing the elimination of the non relevant predictor variable(s)* X_j *and non relevant individual(s) (tuple(s))* Ω_e^{NN}*:* $\Omega^{SS} \leftarrow \Omega - \Omega_e^{NN}$ *without the non relevant pretictor variable(s)* X_j.

We define:

Ω_e – individuals in Ω_t (patients) not correctly classified during the test of the prediction function φ; initially, $\Omega_e = \{\}$.

Ω_a^{NN} – learning sample for training the neural network; initially, $\Omega_a^{NN} = \Omega_a = 70\%$ of Ω.

Ω_e^{NN} – individuals in Ω_t^{NN} not correctly classified during the test of the numeric training by neural network; initially, $\Omega_t^{NN} = \Omega_t = 30\%$ of Ω and $\Omega_e^{NN} = \{\}$.

Ω_a^{SS} – learning sample for symbolic training by induction graph; initially, $\Omega_a^{SS} = 70\%$ of Ω^{SS}, where $\Omega^{SS} = \Omega - \Omega_e^{NN}$ without the non relevant predictor variables X_j.

Ω_e^{SS} – individuals in Ω_t^{SS} not correctly classified during the test of the symbolic training by induction graph; initially, $\Omega_t^{SS} = 30\%$ of Ω^{SS} and $\Omega_e^{SS} = \{\}$.

3.1. *Numeric Training by Neural Network and Minimization of Connections*

The neural network that we use for our experiments is the standard three layer feedforward network. Assume that input tuples in an q-dimensional space are to be classified into two disjoint classes *diabetes of Type 1* and *diabetes of Type 2*. The number of nodes in the input layer correspond to the dimensionality of the input tuples. One output unit with binary encoding is used. The network is trained with value equal to 0 for all tuples in class *diabetes of Type 1* and to 1 for all tuples in class *diabetes of Type 2*.

There is still no clear cut rule to determine the number of hidden nodes to be included in the network. Two different approaches have been proposed to overcome the problem of determining the optimal number of hidden nodes required by a neural network to solve a given problem. The first approach begins with a minimal network and adds more hidden nodes only when they are needed to improve the learning capability of the network (Setiono and Liu, 1995). The second approach begins with an oversized network and then prunes redundant hidden nodes and connections between the layers of the network (Reed, 1993). In this paper, we adopt the second approach since we are interested in finding a network with a small number of hidden nodes as well as the least number of input nodes.

Let w_l^h be the weights for the connections from input node l to hidden node h. Given an input tuple t^i, $i \in \{1, 2, \ldots, a\}$, where a is the number of tuples in the data set Ω_a^{NN}, the activation value of the h-th hidden node is

$$\delta^h = g\Big(\sum_{l=1}^{q}(t_l^i w_l^h) - \Theta^h\Big),$$

where $g(.)$ is an activation function and where Θ^h is the threshold of hidden node h. In our study, we use the hyperbolic tangent function

$$g(x) = \frac{\mathrm{e}^x - \mathrm{e}^{-x}}{\mathrm{e}^x + \mathrm{e}^{-x}},$$

as the activation function for the hidden nodes. Once the activation values of all the hidden nodes haves been computed, the output y^i of the network for input tuple t^i is computed as

$$y^i = \sigma\Big(\sum_{h=1}^{H}(\delta^h v^h)\Big),$$

where v^h is the weight of the connection between hidden node h and the output node, and H is the number of hidden nodes in the network. The activation function used here is the normal sigmoid function,

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}},$$

which yields activation values of the output nodes in the range $[0, 1]$.

A tuple will be correctly classified if $|y^i - y_d^i| \leqslant \beta_1$, where y_d^i is the desired outcome (target) and β_1 is a small positive number. The ultimate objective of the training phase is to obtain a set of weights that make the network classify the input tuples correctly. To measure the classification error, an error function, $E_{app}(w, v)$, is needed so that the training process becomes a process to adjust the weights (w, v) to minimize this function. Furthermore, to facilitate the pruning phase, it is desired to have many with very small values so that they can be set to zero. This is achieved by adding a penalty term $E_{\min}(w, v)$ to the error function.

Generaly, the training phase starts with an initial set of weights (w, v) and iteratively updates the weights to minimize $E_{app}(w, v) + E_{\min}(w, v)$. The gradient descent method can be used for this purpose. The network training is terminated when the gradient of the function is sufficiently small.

For our experiments we follow principles analogous to those of (D'Avila Garcez *et al.*, 2001; Kurfess, 2000; Bologna and Pellegrini, 1996; Setiono and Liu, 1995; Setiono and Leow, 2000) and (Lu *et al.*, 1996) that propose to generate rules from a multilayered network in several stages:

1. A first training permits to determine connections weights of a network possessing only one hidden layer with an arbitrary number of units.
2. By a method of minimization, the resulting network is simplified by eliminating connections with smallest weights, while maintaining the accuracy of the network.
3. A new training is done for the remaining useful connections.
4. The process of optimization stops according to a satisfaction criteria, determined in general with respect to a threshold.
5. Validation of the pruned network which contains only connections that are estimated as relevant.

3.1.1. *The Neural Network Training Algorithm*

One of the most widely used training algorithms is the backpropagation algorithm (Abdi, 1994). Learning in neural network involves modifying the weights of the network in order to minimize a cost function $E_{app}(w, v)+E_{\min}(w, v)$. An appropriate cost function for the classification problems is the cross-entropy function $E_{app}(w, v) = \sum_{i=1}^{a}(y_d^i - y^i)^2$ (Lu *et al.*, 1995; Setiono and Leow, 2000) augmented with a penalty term $E_{\min}(w, v)$ (Reed, 1993; Setiono and Leow, 2000). The training phase is terminated when the norm as of the gradient of the error function $E_{app}(w, v)$ falls bellow a prespecified value β_1 (see Fig. 1). The penalty term $E_{\min}(w, v)$ when minimized pushes the weights values toward the origin of the weight space, and in pratice results in many final weights taking values near or at zero. Network connections with such weights may be removed from the network without sacrificing the network accuracy (Setiono and Leow, 2000). We use a training sample denoted Ω_a^{NN} where all individuals are described by the modalities of their predictor variables, and the value of the goal variable are known.

Initialy $\Omega_a^{NN} = \Omega_a$ and $\Omega_e^{NN} = \{\}$.

1. The sample Ω_a^{NN} is presented to the multilayered neural network.

2. From a completely connected neural network, the function is learned using the back-propagation method (Abdi, 1994), with a specific error function $E_{app}(w, v)$ for creating solution sets.
3. When the convergence to a solution set has been reached, we add a second error term $E_{\min}(w, v)$ for decreasing connections toward zero.
4. Input neurons that have a majority of connections at zero are estimated useless and are proposed for elimination.
5. If $(as > \beta_1)$ then to go to 1 (see Fig. 1).
6. Launch the validation phase of the neural network with Ω_t^{NN}, for the detection of non relevant individuals (later estimated useless and proposed for elimination): verification, on a test sample Ω_t^{NN}, of whether the prediction model denoted φ^{NN} produced by the previous stages is satisfatory. That is to say, whether the quotient $ss = |\Omega_e^{NN}|/|\Omega_t^{NN}|$ is lower than a previously chosen threshold β_2, where Ω_e^{NN} is the set of individuals proposed as non applicable at this stage.
7. If $(ss > \beta_2)$ then $\Omega_a^{NN} \leftarrow \Omega_a^{NN} + \Omega_e^{NN}$ and to go to 1.
8. Send the parameters to the symbolic system with the set of variables and individuals proposed for elimination.

3.1.2. *Error Functions used by the Neural Network*

The general idea is to relax the requirements for a first error function E_{app} in order to facilitate the training, and then to use a second error function $E_{\min}$ allowing the elimination of connections. The back-propagation is supposed to minimize a quadratic error. The elementary error function is given by: $E = (y_d - y)^2$, where y_d and y are respectively the desired outcome (target) and the outcome provided by the network in presence of a given input. One has then two error functions $E_0 = (y)^2$ and $E_1 = (y - 1)^2$, according to whether the target is 0 or 1.

Neural networks techniques have recently been applied to many medical diagnostic (classification) problems (Setiono, 1996). Using the training data set, a network with H hidden units is trained, so as to minimize the cross entropy function (Lu *et al.*, 1995; Reed, 1993; Setiono, 1996):

$$E(w, v) = -\sum_{i=1}^{a} \left(y_d^i \log y^i + (1 - y_d^i) \log(1 - y^i)\right).$$

In the case of binary desired outcomes (yes or no), one can facilitate the training: if the target is 0 and using $E_0(w, v) = -\sum_{i=1}^{a} \log(1 - y^i)$, we can consider that the obtained value is acceptable inside the interval $\left[0, \frac{1}{2}\right]$ and, if the target is 1 and using $E_1(w, v) = -\sum_{i=1}^{a} \log(y^i)$, in the interval $\left]\frac{1}{2}, 1\right]$. This considerably increases the odds to find a solution with zero error (see Fig. 2).

Similarly to these remarks, (Poulard *et al.*, 1991) proposes an error function that is zero on a segment. Fig. 3 represents the general shape of error functions that one can chose.

Among the mathematical constraints imposed by (Poulard *et al.*, 1991) and experimented by (Atmani and Beldjilali, 2003) one can say that the error function, be

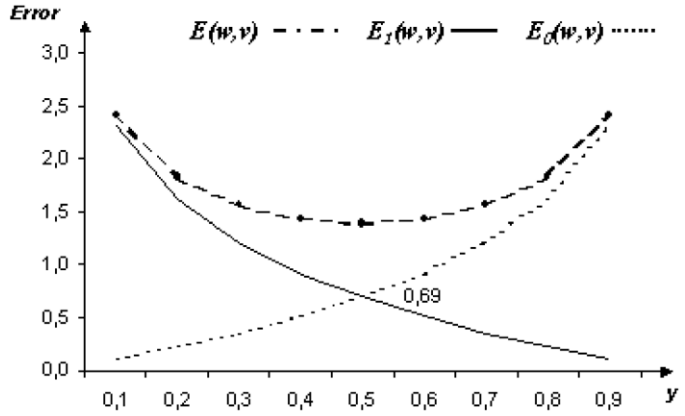

Fig. 2. The general shape of $E(w, v)$, $E_0(w, v)$ and $E_1(w, v)$.

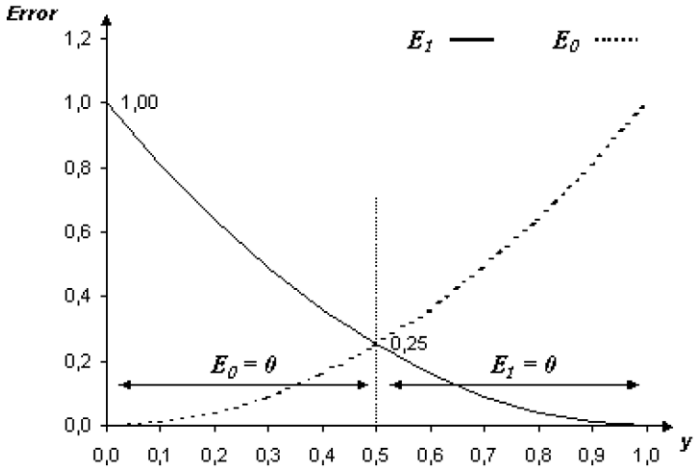

Fig. 3. Zero error function on a segment.

monotonous and increasing for the target 0, be monotonous and decreasing for the target 1, and symmetrical with respect to the $x = \frac{1}{2}$ axis. Following these remarks we propose a simpler function E_{app} that is zero on a segment. In the case where the target is 0, the error function $E_{app}(w, v)$ is the quadratic function defined as follows:

$$y^i \in \left[0, \frac{1}{2}\right] \Rightarrow E_{app}(w, v) = 0,$$

$$y^i \in \left[\frac{1}{2} + \epsilon, 1\right] \Rightarrow E_{app}(w, v) = \sum_{i=1}^{a} \left(\left(\frac{1}{\frac{1}{2} - \epsilon} \right)^2 (y^i - \epsilon)^2 \right), \text{ where } \epsilon \neq 0 \text{ and } \epsilon \lll \frac{1}{2}.$$

In the case where the target is 1 one takes the symmetrical function with respet to $\frac{1}{2}$.

It is rather easy to be convinced that $E_{app}(w, v)$, although much simpler, is an acceptable approximation of $E(w, v)$. Consider for example the case of target 0. The behaviores of $E_{app}(w, v)$ and $E_0(w, v)$ are close enough so that, with an appropriate choise of β_1, using $E_{app}(w, v)$ instead of $E_0(w, v)$ will produce the same solution sets. This can be visualized in Fig. 4.

A symmetric picture would be drawn for target 1, and the same argument holds.

The first stage of the process consists, while minimizing the error function E_{app}, in bringing the network toward an optimal solution. The second stage is the use of this solution set to reduce the number of connections while using $E_{\min}$. The principle is simple: after having converged to a solution set, one looks for the point in the set for which a maximum of connections are at zero.

The error function $E_{\min}$ is a second term which depends only on the values of weights whiches added to the error function E_{app}. Its role is to lower weights toward zero. One can take the error function $E_{\min}$ proposed by (Poulard *et al.*, 1991; Setiono, 1996; Reed, 1993) and tested by (Atmani and Beldjilali, 2003):

$$E_{\min}(w, v) = k \times \big(f(w) + f(v)\big) \quad \text{with } f(w) = \frac{(\eta w)^2}{1 + (\eta w)^2} \quad \text{for all } w \in W,$$

where W is the weights matrix for the connections from input nodes to hidden nodes, $f(v) = \frac{(\eta v)^2}{1+(\eta v)^2}$, for all $v \in V$ where V is the weights matrix of the connection between hidden nodes and the output node and k a constant.

For $k = 1$, $\eta = 0,2$ and $-0.9 < w < +0.9$ the function $f(w)$ chosen for our model is represented in Fig. 5.

For this function to achieve the objective, f must satisfy the following mathematical constraints:

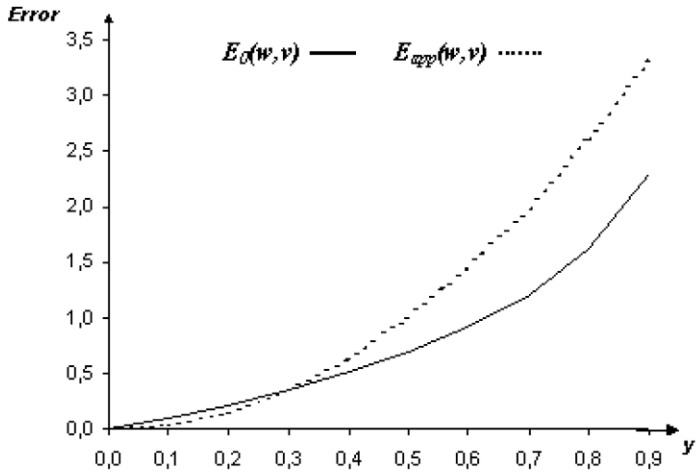

Fig. 4. Comparaison of $E_{app}(w, v)$ and $E_0(w, v)$.

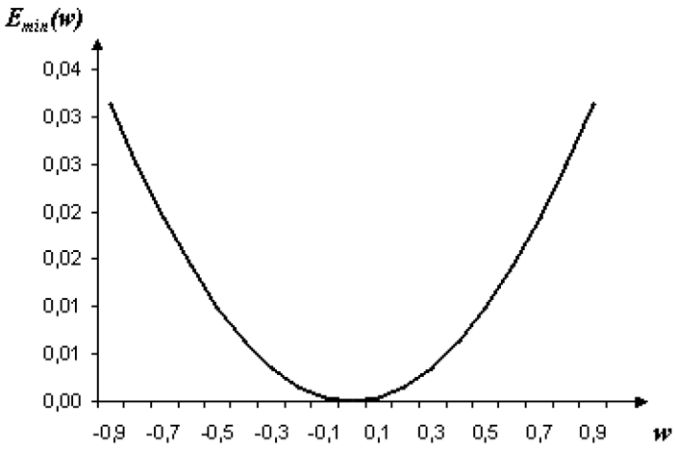

Fig. 5. The function f used to bring weights toward zero.

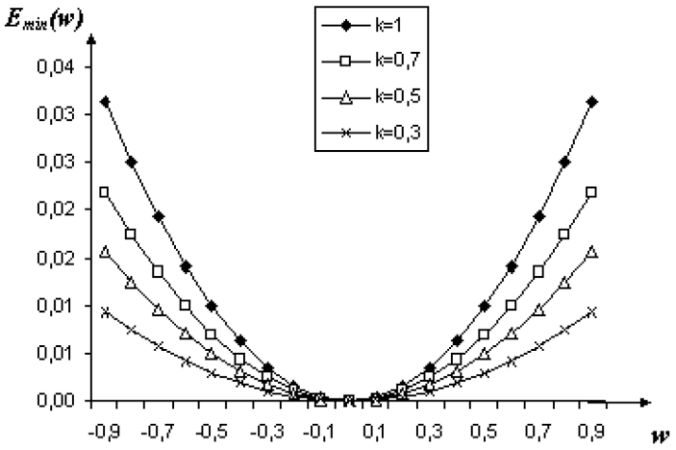

Fig. 6. The role of the constant k in the function $E_{\min}(w, v)$.

- bounded and $f(-\infty) = f(+\infty) = 1$,
- positive, even and $f(0) = 0$.

This method is implemented so that the error function $E_{\min}(w, v)$ is no longer taken into account when the network goes out of the solution set. This is achieved by testing the value of the function $E_{app}(w, v)$ at every iteration. It will then be possible to set the constant k for disconnecting the second function and wait until the network comes back into the solution set. The constant k is equal to $\frac{|E_{app}|}{|W|}$. In this case, the function f must satisfy the first constraint and, according to its definition, f will be bounded by $|E_{app}|$. The role of the constant k in the function $E_{\min}(w, v)$ is represented in Fig. 6. The chosen function f has a good finite development at zero and a good infinite behavior. This explains the requirements of the gradient method, which is optimal with parabolic functions.

3.2. *Symbolic Training by Induction Graph and Rule Generation*

With the help of the sample Ω_e^{NN} produced by the previous phase and by eliminating the predictor variables proposed by the neural network, we start the symbolic treatment for the construction of the graph (SIPINA method) (Zighed, 1996; Zighed and Rakotomalala, 2000):

1. Set the measure of uncertainty.
2. Set parameters: λ, μ and the initial partition S_0.
3. Apply the SIPINA algorithm for going from partition S_i to S_{i+1} and generate induction graph.
4. Generate prediction rules (Rabaseda and Zighed, 1996; Rakotomalala *et al.*, 1999).

The parameters λ, μ, the partitions and all other notions used in this process are introduced by mean of examples and defined in the following paragraphs.

3.2.1. *Definition of a Partition through an Example*

The SIPINA method is a heuristic for the construction of a non arborescent induction graph (Zighed and Rakotomalala, 2000). Its principle consists in performing a succession of stages of fusion and/or splitting of nodes in the graph. Let us suppose that our training sample be composed of 15 diabetic patients belonging to two classes 1 and 2 (Table 2).

The initial partition S_0 includes only one element denoted s_0 that contains the sample, with 10 individuals belonging to class 1 and 5 belonging to class 2. The next partition $S_1 = (s_1, s_2, s_3)$ is generated by the variable X_1. The individuals in node s_i are defined as follows:

$$s_1 = \{\omega \in \Omega_a | X_1(\omega) = 0\}, \quad s_2 = \{\omega \in \Omega_a | X_1(\omega) = 1\} \text{ and } s_3 = \{\omega \in \Omega_a | X_1(\omega) = 2\}.$$

Table 2

Example of training sample

Ω_a	**CLASS**	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
ω_1	1	0	2	2	0	0	0	1	0	2	1
ω_2	1	2	2	0	1	0	0	1	0	2	0
ω_3	1	2	2	0	0	1	0	1	0	2	0
ω_4	1	0	2	0	0	1	0	1	0	2	0
ω_5	1	0	0	2	0	0	0	1	0	2	0
ω_6	1	0	2	0	0	0	0	1	0	2	0
ω_7	1	0	2	0	0	0	0	1	0	2	0
ω_8	1	0	2	0	0	0	0	1	0	2	1
ω_9	1	0	1	0	0	0	0	1	0	2	0
ω_{10}	1	0	1	0	0	0	0	1	0	2	0
ω_{11}	2	1	0	1	1	0	1	6	1	2	0
ω_{12}	2	1	3	1	1	0	1	6	1	2	0
ω_{13}	2	1	0	1	1	1	1	6	1	2	0
ω_{14}	2	1	2	0	0	1	0	6	0	2	0
ω_{15}	2	1	0	1	0	1	1	6	1	2	0

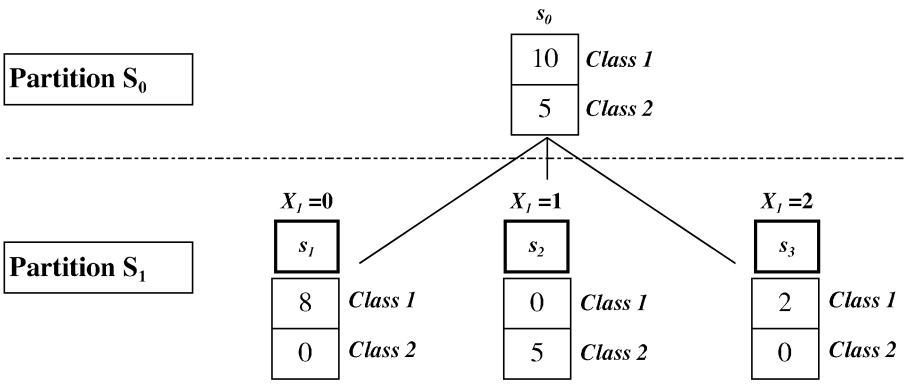

Fig. 7. The construction stages of s_0, s_1, s_2 and s_3.

As in s_0, one distinguishes in s_1, s_2 and s_3 individuals of classes 1 and 2 respectively. Fig. 7 summarizes the construction stages of s_0, s_1, s_2 and s_3.

From partition S_1, the process is iterated looking for a partition S_2 that would be better according to some criteria.

3.2.2. *Measure of Quality of a Partition*

The objective of the SIPINA method is to optimize a criteria τ_λ, called variation of uncertainty, during the transition from S_i to S_{i+1}, defined by $\Delta\tau_\lambda^{(i+1)} = \tau_\lambda^{(S_i)} - \tau_\lambda^{(Si+1)}$.

Let λ be a positive and non zero parameter. For the calculation of $\tau_\lambda^{(S_i)}$, one can use several functions constructed from uncertainty measures, like:

the Shannon entropy

$$\tau_\lambda^{(S_i)} = \sum_{j=1}^{K} \frac{n_{.j}}{n} \Big(- \sum_{i=1}^{m} \frac{n_{ij}+\lambda}{n_{.j}+m\lambda} \log_2 \frac{n_{ij}+\lambda}{n_{.j}+m\lambda} \Big)$$

or the Quadratic entropy

$$\tau_\lambda^{(S_i)} = \sum_{j=1}^{K} \frac{n_{.j}}{n} \Big(- \sum_{i=1}^{m} \frac{n_{ij}+\lambda}{n_{.j}+m\lambda} \Big(1 - \frac{n_{ij}+\lambda}{n_{.j}+m\lambda}\Big) \Big),$$

where:

n_{ij} – size of the population coming from class c_i, which is at node s_j;

$n_{i.}$ – total size of the class c_i;

$n_{.j}$ – total size of node s_j;

n – total size of Ω_a;

m – number of classes;

K – number of nodes s_j.

In our example, the distribution T_1 of individuals for $K = 3$ and $m = 2$ in partition S_1 (Table 2) is represented in Table 3. We can deduce, for $n_{11} = 8$ and $n_{21} = 0$, that the majority class of the node s_1 is c_1.

Setting the parameter λ. The parameter λ of the uncertainty measure controls the construction of the graph by penalizing partitions having many nodes with low population, thus favoring the fusion of such nodes.

The value of λ can be set arbitrarily to 1 or 2, but it can also be determined in an optimal way. The solution proposed by Zighed (Zighed *et al.*, 1998) is to define a node of low size. The value of λ is such that among all possible distributions T_k of μ individuals on m classes we have:

$$\lambda = \max\Big(\lambda(m-1)\frac{2\mu^2 + 2\mu + m\lambda + 2\mu m\lambda}{(\mu + m\lambda)^2(\mu + 1 + m\lambda^2)}\Big).$$

For a simple example $m = 2$ and $\mu = 2$, the optimal value will be $\lambda = 0.61098$.

Setting the value of μ. There exist two strategies to determine the minimal size μ of a node. The first one consists in asking the user to give the minimum number of individuals that every node should include. The second one consists in calculating this number while adopting a statistical view point.

The size of the training sample is $n = n_e + n_c$ where n_e is the number of individuals in the class of examples, and n_c the number of individuals in the class of counter examples. Let s be a terminal node of the induction graph whose total size is $n_s = n_{se} + n_{sc}$. The value of μ for a critical threshold $\alpha_0 = 0.05$ is then

$$\mu = n_s = -\log(\alpha_0) \times \Big(\frac{n}{n_c}\Big).$$

Table 3
The distribution T_1

T_1	s_1	s_2	s_3	$Total$
c_1	$n_{11} = 8$	$n_{12} = 0$	$n_{13} = 2$	$n_{1.} = 10$
c_2	$n_{21} = 0$	$n_{22} = 5$	$n_{23} = 0$	$n_{2.} = 5$
$Total$	$n_{.1} = 8$	$n_{.2} = 5$	$n_{.3} = 2$	$n = 15$

3.2.3. *How to Go from Partition S_i to S_{i+1}*

Let us consider the example of Fig. 7. Partition S_1 possesses three elements s_1, s_2 and s_3. Going from partition S_1 to partition S_2 is done in three phases:

1. **Fusion.** One can note that from S_1 we can generate only one partition by fusion. This phase consists in gathering the nodes belonging to S_1 for generating only one node in S_2 while optimizing the criteria $\tau_\lambda^{(S_2)}$.
   If the gain on the uncertainty $\tau_\lambda^{(S_2)} - \tau_\lambda^{(S_1)}$ is positive, then S_2 is generated. Otherwise, go to phase 2. Note that the fusion is always done between two nodes: if there were three nodes s_1, s_2 and s_3, three partitions could be generated by fusion (two by two) and we would choose the one that maximizes τ_λ.
2. **Fusion-splitting.** As in phase 1, fusions are done between all pairs of nodes. On every node produced by a fusion we search for the best admissible partition by splitting all variable X_j.
   For example, with three nodes in S_1 and three variables, we generate three different partitions for each of the three nodes coming from the fusion in S_2, which gives nine possible partitions. Among all admissible partitions, we then keep those that lead to the best gain on the uncertainty.
3. **Splitting.** On every $s \in S_i$, we look for the best admissible partition by splitting all predictor variables, and we keep the one that optimizes τ_λ.

Fig. 8 summarizes the different phases.

3.2.4. *Generation of Rules*

At the end of the symbolic treatment, we can generate the rules coming from the induction graph. Let us consider the graph of Fig. 8 as if it was a final induction graph, without worrying about checking the details of all calculations that lead to this graph. At that point, we can deduce three prediction rules R_1, R_2 and R_3 that have the form *if* **condition** *then* **conclusion**, as where **condition** is a logical expression in disjunctive-conjunctive form and **conclusion** is the majority class in the node reached by the condition. For example, in Fig. 7, the majority class of s_1 is 8 (class 1), but the majority class of s_2 is 5 (class 2).

R_1:
if $((X_1 = 1)$ *or* $(X_1 = 2))$ *and* $(X_5 = 1)$ *then majority class of* s_9.

R_2:
if $(((X_1 = 1)$ *or* $(X_1 = 2))$ *and* $(X_5 = 0)$ *and* $(X_{10} = 0))$
or $((X_1 = 0)$ *and* $((X_4 = 0)$ *or* $(X_4 = 1))$ *and* $(X_{10} = 0))$
then majority class of s_{11}.

R_3:
if $(((X_1 = 1)$ *or* $(X_1 = 2))$ *and* $(X_5 = 0)$ *and* $(X_{10} = 1))$
or $((X_1 = 0)$ *and* $((X_4 = 0)$ *or* $(X_4 = 1))$ *and* $(X_{10} = 1))$
then majority class of s_{12}.

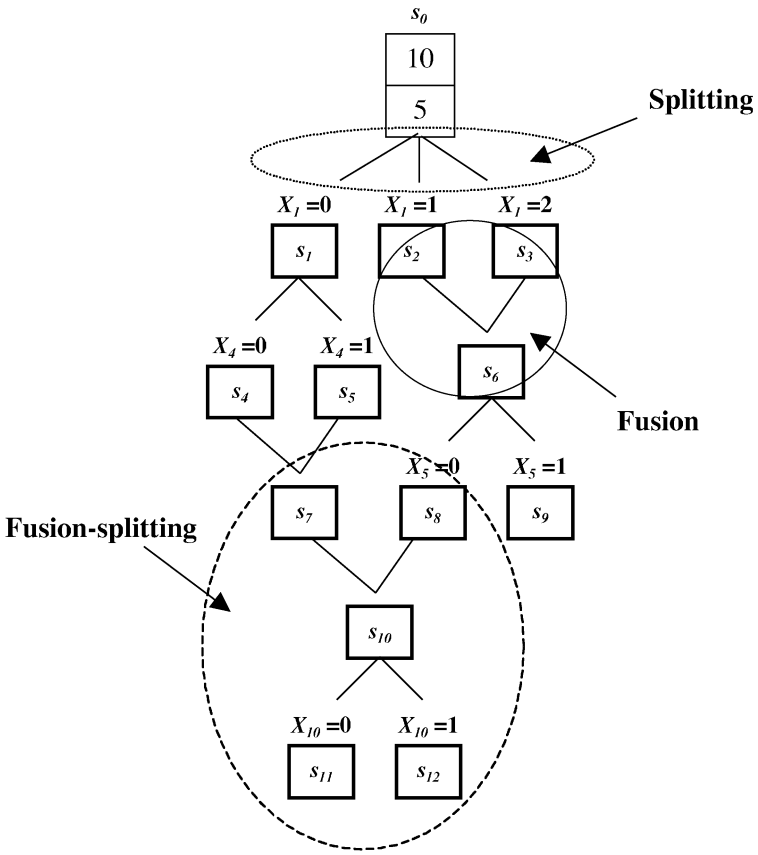

Fig. 8. Going from partition S_i to S_{i+1}.

4. Experimental Results

From an experimental point of view a straightforward manner to evaluate the quality of the training is to match the prediction of the model with the actual values on a sample of the population. This confrontation is summarized in a table called *confusion matrix* (see Table 4). It is possible to synthesize indicators from this table, like the error rate or rate of wrong classification. It is possible to interpret this rate as an average cost of wrong classification when the cost matrix of wrong assignment is the unit matrix; it is also possible to interpret it like an estimator of the probability of carrying out a wrong prediction.

The main interest of the error rate is that it is objective; it is generally used to compare the methods of training on a given problem. To obtain an unbiaised indicator, it is essential not to measure it on the sample which was used to work the model out. To this end, experts often put a sample aside, known as test sample, which is used to evaluate and compare the models.

Table 4
Confusion matrix obtained using the SIPINA method

error rate = 0.0821	$ofType2$	$ofType1$	**Sum**
$ofType2$	835	39	874
$ofType1$	91	496	587
Sum	926	535	1461

Table 5
Generalization

	SIPINA				Neuro-IG		
	No. Var	Ω^{SS}	ξ	Ω_e^{NN}	No. Var	Ω^{SS}	ξ
Diabetes	10	1461	0.0889	122	9	1339	0.0082
Breast	9	699	0.0257	11	9	688	0.0100
Titanic	3	2201	0.2167	96	3	2105	0.1814

We have constructed our medical experimentation basis on 1461 cases drawn from actual medical clinic archives.

Initialization: $|\Omega_a| = 968$ and $|\Omega_t| = 493$

The confusion matrix obtained using the SIPINA method and without neural network data preprocessing is represented in Table 5.

The error rate in our test is $\xi = |\frac{39+91}{1461}| = 0.0889$. We can thus say that by classifying a randomly taken individual in the population, we have 8.89 chances out of 100 to carry out a wrong assignment.

In our Neuro-IG system, a neural network with q input nodes, one hidden layer and one output node were considered. A backpropagation training algorithm was used, starting from initial weights uniformly distributed in $[-1.0, 1.0]$. In all the trials the training was stopped when all the training individuals were learnt, that is when, for each training individual $\omega^i \in \Omega_a^{NN}$, the following condition was met: $|y^i - y_d^i| \leqslant \beta_1$. where y^i represents the output of the neural network and y_d^i is the desired output. In our experiments we chose $\beta_1 = 0.01$ and $\beta_2 = 0.25$.

After several iterations of learning phase and minimization of connections, the neural network arrives at the solution set with a threshold $ss = |\frac{\Omega_e^{NN}}{\Omega_t^{NN}}| = 0.25$. In short, for $\Omega_a^{NN} = 3751$, $\Omega_t^{NN} = 493$ and $\Omega_e^{NN} = 122$, we propose to the symbolic system an optimal data base for training and 122 non classified individuals are proposed for elimination. The neuron that has the maximum of connections at zero is also proposed for elimination. In our case, it is the X_{10} neuron (sex).

Validation is the phase that consists in testing, on sample Ω_t, the rules of prediction generated by our symbolic system. Generalization is the last phase that consists in extending the application of the model to all individuals of the population ($\Omega^{SS} = \Omega_a^{SS} + \Omega_t^{SS}$). To test our system, the generalization has been tested on several data base of larger sizes:

one with 699 individuals containing only continuous variables (breast) and a second one of 2201 individuals containing only discrete variables (titanic). We obtained the results presented in Table 5.

A number of different methods have been proposed to approach the optimal solution to variable selection (Jain and Zongker, 1997). Significant contributions have come from statisticians in the field of pattern recognition, ranging from techniques that find the optimal variable set (Narendra and Fukunaga, 1997) and those that result in a sub-optimal variable set that is near to the optimal solution (Pudil *et al.*, 1994). More recently, some variable selection methods for artificial neural networks have been developed (Mao and Jain, 1995). However, no optimal and generally applicable solution to the variable selection problem exists (Castellano and Fanelli, 2000): some methods are more suitable under certain conditions and some under others, depending on the degree of knowledge about the problem at hand. In a context of diabetic patients monitoring, for example, setting up tools for variable selection is not possible without also taking into account the necessary role of the physician.

Our Neuro-IG system is concerned with the problem of variable and non relevant examples selection using neural networks. In this context, variable selection can be seen as a special case of network pruning, and examples selection as a special case of testing phase of the neural network. The pruning of input nodes is equivalent to removing the corresponding variable from the original predictor variables set. Several pruning procedures for neural networks have been proposed (Reed, 1993), but most of them focus on removing hidden nodes or connections, and they are not directly applicable to prune irrelevant input nodes. Pruning procedures extended to the removal of input nodes were proposed in (Battiti, 1994; Stepps and Bauer, 1996; Setiono, 1996) and (Setiono and Leow, 2000). In the following, to position our Neuro-IG system, we are interested in variable selection using artificial neural networks methods, more precisely in the MIFS filtering method (Battiti, 1994). To compare Neuro-IG and the MIFS method, we have used the system (Rakotomalala, 2005b), in particular four algorithms (ID3, C4.5, CART and Naive Bayes). We obtained the following comparisons results (Table 6).

These experimentations have shown that the new Neuro-IG data preprocessing leave unchanged the classification success, with a success rate of approximately 93%, whereas this reduces the training data base by more than 07% and the graph size by more than 15%, as well as the validation time and the number of descriptive variables. In most of the cases Neuro-IG produces fewer rules with better accuracy. These results show not

Table 6
Comparisons results

	Neuro-IG	Tanagra using the MIFS method			
	ξ	ξ (ID3)	ξ (C4.5)	ξ (CART)	ξ (Naive Bayes)
Diabetes	$0.0082 \simeq 99\%$	0.0130	0.0034	0.0034	0.0185
Breast	$0.0100 \simeq 99\%$	0.0758	0.0358	0.0572	0.0272
Titanic	$0.1814 \simeq 82\%$	0.2240	0.2240	0.2240	0.2240

only that one can generate in an optimal way the rules starting from an induction graph, but also encourage the exploitation of this new technique to optimize the data set size and time. This neural network is now integrated with the SIPINA method in order to have a complete autonomous device for data selection.

5. Conclusion

In this work we have presented the general architecture of our hybrid neuro-symbolic system: neural network and induction graph.

The neural network considers the diagnosis as a classification problem where it is necessary to determine the class where a patient belongs: diabetes type 1 or type 2 in our example. The symbolic step applies size reductions proposed by the neural network to construct the optimal induction graph and to generate rules that achieve the prediction function of the symbolic system.

All along this experimentation (diabetic patients), the same test sample used by the neural network to detect and eliminate the predictor variables and the non relevant individuals, is also used for the validation of the symbolic system. We noted a rate of equal satisfactory answers of 88%. We nevertheless wish to point out that the specific framework of this experiment does not allow full generalization of the prediction function. On the other hand, one can say that it is most representative of the initial population used for the training.

Our contribution to hybrid neuro-symbolic systems belongs to a family of methods that are extensively exploited in the domain of data mining. It provides an experimental setting that permits to optimize the construction of induction graphs and it brings some rigorous answers to the questions of minimality of population sizes and consistency of prediction rules. In addition, to better exploit the data, our system does a preprocessing before starting the development of the prediction model. One can say that we have reached our objective of reduction of complexity of storage, therefore the complexity of calculation.

Finally, it is not yet possible to really talk about generalization of the function elaborated by our experimental system, where the human operator is still in charge of decision making. It is however foreseeable to design a complete cooperation between the neural network and the symbolic system for producing a general prediction function. Instead of choosing as satisfaction criteria a threshold depending on Ω_e^{NN} and Ω_t^{NN}, we could generalize this criteria and have it depend on Ω_e^{NN} and Ω_e^{SS}.

A more balanced cooperation between the neural network and the symbolic process deserves also to be explored. Once the validation is over, it would then be necessary to proceed to the test for comparing and verifing whether the symbolic system reacts correctly to some perturbations.

Acknowledgements

I would like to express my gratitude to Philippe Jorrand who has significantly and decisively contributed to improving the written presentation of this work.

References

Abdi, H. (1994). *Networks of Neurons*. Press Academic of Grenoble.

Atmani, B., and B. Beldjilali (2003). Neural network for the selection and the elimination of the exogenous variables and the non applicable individuals. In *The Second International Workshop on Advanced Computation For Engineering Applications* (*ACEA03*), December 21–22. Electronics Research Institute (ERI), Cairo, Egypt.

D'Avila Garcez, A.S., K. Broda and D.M. Gabbay (2001). Symbolic knowledge extraction from trained neural networks: a sound approach. *Artificial Intelligence*, **125**(1), 155–207.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Networks*, **5**(4), 537–550.

Bentayeb, F., and J. Darmont (2002). Decision tree modeling with relational views. In M.-S. Hacid *et al.* (Eds), *ISMIS 2002*. *LNAI*, vol. 2366. pp. 423–431.

Bologna. G., and C. Pellegrini (1996). Extraction of rules of a PMC network to values of entry continues using units to doorstep. *Acts of Days NSI*.

Bingchiang, J., Y.M. Jeng and T.P. Liang (1997). FILM: a fuzzy inductive learning method for automated knowledge acquisition. *Decision Support Systems*, **21**(2), 61–73.

Breiman, L., J.H. Friedman, R.A. Olshen and C.J. Stone (1984). Classification and regression trees. *Technical Report*, Wadsworth International, Monterey, CA.

Cantu-Paz, E., and C. Kamath (2003). Induction oblique decision tree with evolutionary algorithms. *IEEE Transaction on Evolutionary Computation*, **7**(1), 54–69.

Castellano, G., A.M. Fanelli and C. Mencar (2002). A neuro-fuzzy network to generate human-understandable knowledge from data. *Cognitive Systems Research*, **3**(2), 125–144.

Castellano, G., and A.M. Fanelli (2000). Variable selection using neural network models. *Neurocomputing*, **31**, 1–13.

Chaudhuri, S. (1998). Data mining and database systems. *Data Engineering Bulletin*, **21**(1), 4–8.

Craven, M.W., and J.W. Shavlik (1997). Using neural networks for data mining. *Future Generation Computer Systems*, **13**(2), 211–229.

Crémilleux, B. (2000). Decision tree as a data mining tool. *Computing and Information Systems*, **7**, 91–97.

Denis, F., and R. Gilleron (1999). Apprentissage partir d'exemples. *Technical Report*. University of Lille 3, Grappa.

Duch, W., R. Adamczak and K. Grabczewski (1999). Methodology of extraction, optimization and application of logical rules. In *Intelligent Information Systems VIII Proceedings of the Workshop*, June 14–18, Poland.

Duhamel, A., M. Picavet, P. Devos and R. Beuscart (2001). From data collection to knowledge data discovery: a medical application of data mining. *Studies in Health Technology and Informatics*, **84**, 1329–1333.

Erray, W. (2001). Extraction of knowledge from data, generalized segmentation. In *Memory DEA ECD*. ERIC laboratory, University of Lyons 2.

Fan, Y., and C.J.A.M.E.S. Li (2002). Diagnostic rule extraction from trained feedforward neural networks. *Mechanical Systems and Signal Processing*, **16**(6), 59–67.

Fayyad, U., G.P. Shapiro and P. Smyth (1996). The KDD process for extraction useful knowledge from volumes data. *Communication of the ACM*.

Freitag, D. (2000). Machine learning for information extraction in informal domains. *Machine Learning*, **39**(2–3), 169–202.

Fujimoto, K., and S. Nakabayashi (2003). Applying GMDH algorithm to extract rules from examples. *Systems Analysis Modelling Simulation*, **43**(10), 1311–1319.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA.

Ganascia, J.G. (1988). CHARADE: A study of the learning bias semantics. In *European Conference on Artificial Intelligence*. ECAI, Munich.

Hastie, T., R. Tibshirani and J. Freidman (2001). *The Elements of Statistical Learning – Data Mining, Inference and Prediction*. Springer.

Herr, A., N.I. Klomp and J.S. Atkinson (1997). Identification of bat echolocation calls using a decision tree classification system. *Complexity International*, **4**.

Ishibuchi, H., T. Nakashima and T. Murata (2001). Three-objective genetics-based machine learning for linguistic rule extraction. *Information Sciences*, **136**(1), 109–133.

Jain, A., and D. Zongker (1997). Feature selection: evaluation, application, and small sample performance. *IEEE Trans. Pattern Anal. Mech. Intell.*, **19**(2), 153–158.

Jensen, D., R. Kohavi and M. Sahami (2002). Supervised and unsupervised discretization of continuous attributes. In *Proceeding of 12th International Conference on Machine Learning*.

Kodratoff, Y. (1997). The extraction of knowledge from data, a new topic for the scientific research. *Magazine Electronic READ*.

Kohavi, R., and J. Quinlan (2002). Decision tree discovery. In Klosgen and Zytkow (Eds.), *Handbook of Data Mining and Knowledge Discovery*. pp. 267–276.

Kurfess, F.J. (2000). Neural networks and structured knowledge: rule extraction and applications. *Applied Intelligence*, **12**(1–2), 7–13.

Lee, I.N., S.C. Lee and M.J. Embrechts (2002). Important variable selection techniques with multiple solutions for medical information applications. *Medical Informatics and the Internet in Medicine*, **27**(4), 253–266.

Lee, S.J., and K. Sian (2001). A review of data mining techniques. *MCB UP Ltd*, *Industrial Management and Data Systems*, **11**, 41–46.

Lefebure, R., and G. Venturi (2001). *Data Mining*. EYROLLES, Paris.

Liao, S.C., and I.N. Lee (2002). Appropriate medical data categorization for data mining classification techniques. *Medical Informatics and the Internet in Medicine*, **27**(1), 59–67.

Liao, T.W., Z.H. Zhan and C.R. Mount (1999). An integrated database and expert system for failure mechanism identification: Part I – Automated knowledge acquisition. *Engineering Failure Analysis*, **6**(6), 387–406.

Lim, T., W. Loh and Y. Shih (2000). A comparison of prediction accuracy, complexity and training of thirty-three old and new classification algorithms. *Machine Learning Journal*, **40**, 203–228.

Lisboa, P.J.G., T.A. Etchells and D.C. Pountney (2002). Minimal MLPs do not model the XOR logic. *Neurocomputing*, **48**(1), 1033–1037.

Lu, H., R. Setiono and H. Liu (1995). NeuroRule: A connectionist approach to data mining. in *Proceeding of the 21st VLDB conference*, Zürich, Switzerland. pp. 478–489.

Lu, H., R. Setiono and H. Liu (1996). Effective data mining using neural networks. *IEEE Transaction on Knowledge and Data Engineering*, **8**(6), 957–961.

Mao, J., and A.K. Jain (1995). Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. Neural Networks*, **6**, 296–317.

Montgomery, D., G. Swinnen and K. Vanhoof (1997). Comparison of som AI and statistical classification methods for marketing case. *European Journal of Operational Research*, **103**(1), 312–325.

Narendra, P.M., and K. Fukunaga (1997). A branch and bound algorithm for feature subset selection. *IEEE Trans. Comput.*, **26**(9), 917–922.

Netz, A., S. Chaudhuri, J. Bernhardt and U. Fayyad (2000). Integration of data mining and relational databases. In *26th International Conference on Very Large Data Bases* (*VLDB 00*), Cairo, Egypt. Morgan Kaufmann. pp. 719–722.

Nurnberger, A. (2003). A hierarchical recurrent neuro-fuzzy model for system identification. *International Journal of Approximate Reasoning*, **32**(2), 59–67.

Olaru, C., and L. Wehenkel (2003). A complete fussy decision tree technique. *Fussy Set and Systems*, **138**(2).

Osorio, F.S., and B. Amy (1999). INSS: A hybrid system for constructive machine learning. *Neurocomputing*, **28**(1), 59–67.

Poulard, H., D. Martinez and D. Esteve (1991). Minimization of the number of connections of a network multilayered neuronal at the time of the training. *Report LAAS n91451*.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*.

Quinlan, J.R. (1993). *C4.5*: *Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Rabaseda, S., and D.A. Zighed (1996). Generation and simplification of rules in graphs of induction. In *Acts of the 25th Symposium of the Economic Structures, Econometrics and Data Processing*. p. 7.

Rakotomalala, R. (2005a). Arbres de Décision. *Revue Modulad*, **33**.

Rakotomalala, R. (2005b). TANAGRA: une plate-forme d'expérimentation pour la fouille de données. *Revue*

MODULAD, **32**, 70–85.
Rakotomalala, R., D.A. Zighed and Feschet (1999). *Characterization of Production Rules in a Process of Induction*. Hermes Science Publication, Paris.
Ramesh, A.N., C. Kambhampati, J.R.T. Monson and P.J. Drew (2004). Artificial intelligence in medicine. *Annals of the Royal College of Surgeons of England*, **86**(5), 334–338.
Reed, R. (1993). Pruning algorithms: A survey. *IEEE Transaction on Neural Networks*, **4**, 740–747.
Pudil, P., J. Novovicova and J. Kittler (1994). Floating search methods in feature selection. *Pattern Recognition Lett.*, **15**, 1119–1125.
Sebban, M., I. Mokrousov, N. Rastogi and C. Sola (2002). A data mining approach to spacer oligonucleotide typing of Mycrobacterium tuberculosis. *Bioinformatics*(electronic edition), **18**(2), 235–243.
Setiono, R., and H. Liu (1995). Understanding neural networks via rule extraction. In *Proceeding of the 14th Internatianals Joint Conference one Artificial Intelligence*.
Setiono, R. (1996). Extraction rules from pruned neural networks for breast cancer diagnosis. *Artificial Intelligence in Medicine*, **8**(1), 37–51.
Setiono, R., and W.K. Leow (2000). FERNN: An algorithm for fast extraction of rules from neural networks. *Applied Intelligence*, **12**(1–2), 15–25.
Stepps, J.M., and K.W. Bauer (1996). Improved feature screening in feedforward neural networks. *Neurocomputing*, **13**, 47–58.
Zighed, D.A. (1996). *SIPINA for Windows*, ver 2.5. Laboratory ERIC, University of lyon2.
Zighed, D.A., S. Rabaseda and R. Rakotomalala (1998). FUSINTER: a method for discretization of continuous attributes. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **6**, 307–326.
Zighed, D.A., and R. Rakotomalala (2000). *Graphs of induction, Training and Data Mining*. Hermes Science Publication.
Zighed, D.A., S. Lallich and F. Muhlenbach (2002). Separability index in supervised learning. In *Principles of Data Mining and Knowledge Discovery*, 6th European Conference.

B. Atmani was born in 1967, Mascara, Algeria. He graduated in 1991 from the Department of Computer Science in Oran Algeria, and obtained his master of science degree in the same department, in 1996. He is currently a PhD candidate in the Computer Science Department at the University of Oran. His research interests include knowledge discovery in databases, datamining, feature selection, neural networks, and cellular automata. A substantial part of his recent research was conducted in cooperation with the Laboratory of Informatics of Grenoble, France.

B. Beldjilali received his PhD degree in computer science from the University of Oran Algeria, in 1996. He is a professor in the Computer Science Department at the University of Oran. His research interests include formal specifications, knowledge management, databases, artificial intelligence and automatic learning.

Neuro–IG: hibridinė sistema nereikšmingų kintamųjų ir objektų atrinkimui ir eliminavimui

Baghdad ATMANI, Bouziane BELDJILALI

Šiame straipsnyje yra pateikiama hibridinės neuro-simbolinės sistemos (neuroninių tinklų ir indukcijos grafų) bendra architektūra. Ši sistema skirta nereikšmingų kintamųjų ir objektų atrinkimo ir eliminavimo modelio kūrimui. Pagrindinis tikslas yra sukurti įrankius, išrenkančius reikšmingus kintamuosius ir objektus, o tuo pačiu sukurti automatinio mokymosi sistemą. Siekiama sumažinti atminties naudojimą, skaičiavimų sudėtingumą ir palaipsniui gerinti sistemos mokymosi kokybę.