K

Knowledge Discovery with Artificial Neural Networks

Juan R. Rabuñal Dopico
University of A Coruña, Spain

Daniel Rivero Cebrián
University of A Coruña, Spain

Julián Dorado de la Calle
University of A Coruña, Spain

Nieves Pedreira Souto
University of A Coruña, Spain

INTRODUCTION

The world of Data Mining (Cios, Pedrycz & Swiniarrski, 1998) is in constant expansion. New information is obtained from databases thanks to a wide range of techniques, which are all applicable to a determined set of domains and count with a series of advantages and inconveniences. The Artificial Neural Networks (ANNs) technique (Haykin, 1999; McCulloch & Pitts, 1943; Orchad, 1993) allows us to resolve complex problems in many disciplines (classification, clustering, regression, etc.), and presents a series of advantages that convert it into a very powerful technique that is easily adapted to any environment. The main inconvenience of ANNs, however, is that they can not explain what they learn and what reasoning was followed to obtain the outputs. This implies that they can not be used in many environments in which this reasoning is essential.

This article presents a hybrid technique that not only benefits from the advantages of ANNs in the data-mining field, but also counteracts their inconveniences by using other knowledge extraction techniques. Firstly we extract the requested information by applying an ANN, then we apply other Data Mining techniques to the ANN in order to explain the information that is contained inside the network. We thus obtain a two-levelled system that offers the advantages of the ANNs and compensates for its shortcomings with other Data Mining techniques.

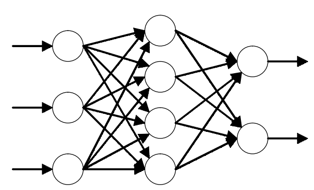
BACKGROUND

Ever since artificial intelligence appeared, ANNs have been widely studied. Their generalisation capacity, and their inductive learning convert them into a very robust technique that can be used in almost any domain. An ANN is an information processing technique that is inspired on neuronal biology and consists of a large amount of interconnected computational units (neurons), usually in different layers. When an input vector is presented to the input neurons, this vector is propagated and processed in the network until it becomes an output vector in the output neurons. Figure 1 shows an example of neural network with 3 inputs, 2 outputs and three layers with 3, 4 and two neurons in each one.

The ANNs have proven to be a very powerful tool in a lot of applications, but they present a big problem: their reasoning process cannot be explained, i.e. there is no clear relationship between the inputs that are presented to the network and the outputs it produces. This means that ANNs cannot be used in certain domains, even though several approaches and attempts to explain their behaviour have tried to solve this problem.

The reasoning of ANNs, and the rules extraction that explains their functioning, were explained in various ways. One of the first attempts established an equivalence between ANNs and fuzzy rules (Benítez, Castro, & Requena, 1997; Buckley, Hayashi, & Czogala, 1993; Jang & Sun, 1992), obtaining only theoretical solutions. Other works were based on the individual analysis of each

Figure 1. Example of Artificial Neural Network



neuron in the network, and its connection with the consecutive neurons. Towell and Shavlik (1994) in particular see the connections between neurons as rules, and Andrews and Geva (1994) uses networks with functions that allow a clear identification of the dominant inputs.

Other approaches are the RULENEG (Rule-Extraction from Neural Networks by Step-wise Negation) (Pop, Hayward, & Diederich, 1994) and TREPAN (Craven, 1996) algorithms. The first approach however modifies the training set and therefore loses the generalisation capacity of the ANNs. The TREPAN approach is similar to decision tree algorithms such as CART (Classification and Regression Trees) or C4.5, which turns the ANN into a MofN (M-of-N) decision tree.

The DEDEC (Decision Detection) algorithm (Tickle, Andrews, Golea, & Tickle, 1998) extracts rules by finding minimal information sufficient to distinguish, from the neural network point of view, between a given pattern and all other patterns. The DEDEC algorithm uses the trained ANN to create examples from which rules can be extracted. Unlike other approaches, it also uses the weight vectors of the network to obtain an additional analysis that improves the extraction of rules. This information is then used to direct the strategy for generating a (minimal) set of examples for the learning phase. It also uses an efficient algorithm for the rule extraction phase. Based on these and other already mentioned techniques, Chalup, Hayward and Diedrich (1998); Visser, Tickle, Hayward and Andrews (1996); and Tickle, Andrews, Golea and Tickle (1998) also presented their solutions.

The methods for the extraction of logical rules, developed by Duch, Adamczak and Grabczewski (2001), are based on multilayer perceptron networks with the MLP2LN (Multi-Layer Perceptron converted to Logical Network) method and its constructive version C-MLP2LN. MLP2LN consists in taking a multilayer and already trained perceptron and simplify it in order to obtain a network with weights 0, +1 or -1. C-MLP2LN acts in a similar way. After this process, the dominant rules are easily extracted, and the weights of the input layer allow us to deduct which parameters are relevant.

More recently, genetic algorithms (GAs) have been used to discover rules in ANNs. Keedwell, Narayanan and Savic (2000) use a GA in which the chromosomes are rules based on value intervals or ranges applied to the inputs of the ANN. The values are obtained from the training patterns.

The most recent works in rules extraction from ANNs are presented by Rivero, Rabuñal, Dorado, Pazos and Pedreira (2004) and Rabuñal, Dorado, Pazos and Rivero (2003). They extract rules by applying a symbolic regression system, based on Genetic Programming (GP) (Engelbrecht, Rouwhorst & Schoeman, 2001; Koza, Keane, Streeter, Mydlowec, Yu, & Lanza, 2003; Wong & Leung, 2000), to a set of *inputs / outputs* produced by the ANN. The set of network *inputs / produced outputs* is dynamically modified, as explained on this paper.

ARCHITECTURE

This article presents an architecture in two levels for the extraction of knowledge from databases. In a first level, we apply an ANN as Data Mining technique; in the second level, we apply a knowledge extraction technique to this network.

Data Mining with ANNs

Artificial Neural Networks constitute a Data Mining technique that has been widely used as a technique for the extraction of knowledge from databases. Their training process is based on examples, and presents several advantages that other models do not offer:

- A high generalisation level. Once ANNs are trained with a training set, they produce outputs (close to desired or supposed outputs) for inputs that were never presented to them before.
- A high error tolerance. Since ANNs are based on the successive and parallel interconnection between many processing elements (neurons), the output of the system is not significantly affected if one of them fails.
- A high noise tolerance.

All these advantages turn ANNs into the ideal technique for the extraction of knowledge in almost any domain. They are trained with many different training algorithms. The most famous one is the backpropagation algorithm (Rumelhart, Hinton & Williams, 1986), but many other training algorithms are applied according to the topology of the network and the use that is given to it. In the course of recent years, Evolutionary Computation techniques such as Genetic Algorithms (Holland, 75) (Goldberg, 89) (Rabuñal, Dorado, Pazos, Gestal, Rivero & Pedreira, 2004a; Rabuñal, Dorado, Pazos, Pereira & Rivero, 2004b) are gaining ground, because they correct the defects of other training algorithms, such as the tendency to generate local minimums or to overtrain the network.

Even so, and in spite of these algorithms that train the network automatically (and even search for the topology of the network), ANNs present a series of defects that make them useless in many application fields. As we already said, their main defect is the fact that in general they are not interpretable: once an input is applied to the