Rule Generation Methods Based on Logic Synthesis

Marco Muselli
Italian National Research Council, Italy

INTRODUCTION

One of the most relevant problems in artificial intelligence is allowing a synthetic device to perform inductive reasoning, i.e. to infer a set of rules consistent with a collection of data pertaining to a given real world problem. A variety of approaches, arising in different research areas such as statistics, machine learning, neural networks, etc., have been proposed during the last 50 years to deal with the problem of realizing inductive reasoning.

Most of the developed techniques build a black-box device, which has the aim of solving efficiently a specific problem generalizing the information contained in the sample of data at hand without caring about the intelligibility of the solution obtained. This is the case of connectionist models, where the internal parameters of a nonlinear device are adapted by an optimization algorithm to improve its consistency on available examples while increasing prediction accuracy on unknown data.

The internal structure of the nonlinear device and the training method employed to optimize the parameters determine different classes of connectionist models: for instance, multilayer perceptron neural networks (Haykin, 1999) consider a combination of sigmoidal basis functions, whose parameters are adjusted by a local optimization algorithm, known as back-propagation. Another example of connectionist model is given by support vector machines (Vapnik, 1998), where replicas of the kernel of a reproducing kernel Hilbert space are properly adapted and combined through a quadratic programming method to realize the desired nonlinear device.

Although these models provide a satisfactory way of approaching a general class of problems, the behavior of synthetic devices realized cannot be directly understood, since they generally involve the application of nonlinear operators, whose meaning is not directly comprehensible. Discriminant analysis techniques as well as statistical nonparametric methods (Duda, Hart, & Stork., 2001), like *k*-nearest-neighbor or projection pursuit, also belong to the class of black-box approaches, since the reasoning followed by probabilistic models to perform a prediction cannot generally be expressed in an intelligible form.

However, in many real world applications the comprehension of this predicting task is crucial, since it provides a direct way to analyze the behavior of the artificial device outside the collection of data at our disposal. In these situations the adoption of black-box techniques is not acceptable and a more convenient approach is offered by *rule generation methods* (Duch, Setiono, & Zurada, 2004), a particular class of machine learning techniques that are able to produce a set of intelligible rules, in the if-then form, underlying the real world problem at hand.

Several different rule generation methods have been proposed in the literature: some of them reconstruct the collection of rules by analyzing a connectionist model trained with a specific optimization algorithm (Setiono, 2000; Setnes, 2000); others generate the desired set of rules directly from the given sample of data. This last approach is followed by algorithms that construct decision trees (Hastie, Tibshirani, & Friedman, 2001; Quinlan, 1993) and by techniques in the area of Inductive Logic Programming (Boytcheva, 2002; Quinlan & Cameron-Jones, 1995).

A novel methodology, adopting proper algorithms for logic synthesis to generate the set of rules pertaining to a given collection of data (Boros, Hammer, Ibaraki, & Kogan, 1997; Boros et al., 2000; Hong, 1997; Sanchez, Triantaphyllou, Chen, & Liao, 2002; Muselli & Liberati, 2000), has been recently proposed and forms the subject of the present chapter. In particular, the general procedure followed by this class of methods will be outlined in the following sections, analyzing in detail the specific implementation followed by one of these techniques, Hamming Clustering (Muselli & Liberati, 2002), to better comprehend the peculiarities of the rule generation process.

BACKGROUND

Any logical combination of simple conditions can always be written as a Disjunctive Normal Form (DNF) of binary variables, each of which takes into account the fulfillment of a particular condition. Thus, if the inductive reasoning to be performed amounts to making a binary decision, the optimal set of **if-then** rules can be

associated with a Boolean function *f* that assigns the most probable output to every case.

Since the goal of methods for logic synthesis is exactly the determination of the DNF for a Boolean function *f*, starting from a portion of its truth table, they can be directly used to generate a set of rules for any pattern recognition problem by examining a finite collection of examples, usually called *training set*. To allow the generalization of the information contained in the sample at hand, a proper logic synthesis technique, called Hamming Clustering (HC) (Muselli & Liberati, 2000; Muselli & Liberati, 2002), has been developed. It proceeds by grouping together binary strings with the same output value, which are close among them according to the Hamming distance.

Theoretical results (Muselli & Liberati, 2000) ensure that HC has a polynomial computational cost $O(n^2cs+nc^2)$, where n is the number of input variables, s is the size of the given training set, and c is the total number of AND ports in the resulting digital circuit. A similar, more computationally intensive, methodology has been proposed by Boros et al. (2000).

Every method based on logic synthesis shows the two following advantages:

- It generates artificial devices that can be directly implemented on a physical support, since they are not affected by problems connected with the precision used when numbers are stored.
- It determines automatically the significant inputs for the problem at hand (feature selection).

MAIN THRUST

A typical situation, where inductive reasoning has to be performed, is given by *pattern recognition problems*. Here, vectors $\boldsymbol{x} \in \Re^n$, called *patterns*, have to be assigned to one of two possible classes, associated with the values of a binary output y, coded by the integers 0 and 1. This assignment task must be consistent with a collection of m examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, m$, called *training set*, obtained by previous observations for the problem at hand. The target is to retrieve a proper binary function $g(\boldsymbol{x})$ that provides the correct answer $y = g(\boldsymbol{x})$ for most input patterns $\boldsymbol{x}$.

Solving Pattern Recognition Problems Through Logic Synthesis

Inductive reasoning occurs if the form of the target function g is directly understandable; a possible way of achieving this result is to write g as a collection of intelligible rules in the **if-then** form. The conditions included in the **if** part of each rule act on the input variables contained in the vector $\boldsymbol{x}$; consequently, they have different form depending on the range of values assumed by the component x_j of the vector $\boldsymbol{x}$ to which they refer. Three situations can be devised:

1. **Continuous Variables:** x_j varies within an interval $[a,b]$ of the real axis; no upper bound on the number of different values assumed by x_j is given.
2. **Discrete (Ordered) Variables:** x_j can assume only the values contained in a finite set; typically, the first positive integers are considered with their natural ordering.
3. **Nominal Variables:** x_j as for discrete variables x_j can assume only the values contained in a finite set, but there is no ordering relationship among them; again, the first positive integers are usually employed for the values of x_j.

Binary variables are considered as a particular case of nominal variables, but the values 0 and 1 are used instead of 1 and 2.

Henceforth, only threshold conditions of the kind $x_j < c$, being c a real number, will be considered for inclusion in the **if** part of a rule, when x_j is a continuous or a discrete variable. On the other hand, a nominal variable $x_j \in \{1,2,\ldots,k\}$ will participate in g only through membership conditions, like $x_j \in \{1,3,4\}$. Separate conditions are composed only by means of AND operations, whereas different rules are applied as if they were linked by an OR operator.

As an example, consider the problem of analyzing the characteristics of clients buying a certain product: the average weekly expense x_1 is a continuous variable assuming values in the interval [0,10000], whereas the age of the client x_2 is better described by a discrete variable in the range [0,120]. His/her activity x_3 gives an example of nominal variable; suppose to consider only four categories: farmer, worker, employee, and manager, coded by integers 1, 2, 3, and 4, respectively. A final binary variable x_4 is associated with the gender of the client (0 = male, 1 = female). With this notation, two rules for the problem at hand can assume the following form:

if $x_1 > 300$ AND $x_3 \in \{1,2\}$ **then** $y = 1$ (he/she buy the product)

if $x_2 < 20$ AND $x_4 = 0$ **then** $y = 0$ (he/she does not buy the product)

Note that $x_3 \in \{1,2\}$ refers to the possibility that the client is a farmer or a worker, whereas $x_4 = 0$ (equivalent to $x_4 \in \{0\}$) means that the client must be a male to verify the conditions of the second rule.