Binary Rule Generation via Hamming Clustering

Marco Muselli, *Member*, *IEEE*, and Diego Liberati

Abstract—The generation of a set of rules underlying a classification problem is performed by applying a new algorithm called Hamming Clustering (HC). It reconstructs the AND-OR expression associated with any Boolean function from a training set of samples. The basic kernel of the method is the generation of clusters of input patterns that belong to the same class and are close to each other according to the Hamming distance. Inputs which do not influence the final output are identified, thus automatically reducing the complexity of the final set of rules. The performance of HC has been evaluated through a variety of artificial and real-world benchmarks. In particular, its application in the diagnosis of breast cancer has led to the derivation of a reduced set of rules solving the associated classification problem.

Index Terms—Rule generation, Hamming clustering, knowledge discovery, Boolean function approximation, generalization.

1 Introduction

The target of classical expert systems in the field of artificial intelligence has been to emulate the human reasoning by inferring appropriate rules from a given knowledge base containing a complete set of basic procedural relations [1], [2]. The construction of this knowledge base requires a long and complex work (years of men power) of interdisciplinary experts [3], [4]. Although the resulting systems can achieve a good degree of efficacy in several contexts, their efficiency level is very low that even the bare execution often requires excessive computational cost.

Thus, it would be important to reconstruct a knowledge base typical of the expert system approach, starting only from a finite set of samples pertaining to the problem at hand. To achieve this goal, usually called *rule generation*, a wide variety of methods have been proposed in the literature; they can be subdivided into two broad classes denoted as *rule extraction* and *rule induction*.

The former includes all the techniques [5], [6] that reconstruct a set of understandable rules by analyzing a model trained with a specific learning algorithm. Typical examples are methods for extracting rules from artificial neural networks [6], [7], [8], [9], [10], [11], [12], [13]; in these cases, the accuracy can decrease when passing from the trained connectionist model to the associated rule set. In order to avoid this drawback, proper architectures and learning techniques have been introduced, so as to simplify the task of knowledge extraction. The combination of neural networks and fuzzy logic systems yields the so-called "neuro-fuzzy" hybrid models [14], [15], [16], [17], which represent a good alternative for rule generation.

- *M. Muselli is with the Istituto per i Circuiti Elettronici, Consiglio Nazionale delle Ricerche, via De Marini, 6, 16149 Genova, Italy. E-mail: muselli@ice.ge.cnr.it.*
- *D. Liberati is with the Centro di Studio sulle Tecnologie dell'Informatica e dell'Automazione, Consiglio Nazionale delle Ricerche, Politecnico di Milano, piazza Leonardo da Vinci, 32, 20133 Milano, Italy. E-mail: liberati@elet.polimi.it.*

Manuscript received 4 Feb. 1999; revised 1 June 2001; accepted 24 Sept. 2001. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 109138.

Specific methods have also been proposed to recover a set of understandable rules from decision trees [18], [19], [20], [21]; here, the target can be achieved with less effort, since decision trees already present knowledge in a convenient structured form.

A complementary approach is followed by rule induction techniques, which generate the desired set of rules directly from the given collection of samples. To this class of techniques belong methods developed in the area of *Inductive Logic Programming* (ILP) [22], [23], [24], [25]; they aim at constructing a logic program associated with a set of rules expressed in first-order logic. Specific heuristics, such as the *separate-and-conquer* procedure [24], are usually employed to produce an initial redundant collection of rules. The problem of overfitting is then avoided by adopting proper pruning techniques, whose action greatly affects the final accuracy. An interesting review of available pruning algorithms for ILP and, more generally, for rule induction techniques is contained in [26].

Recently, a new class of methods belonging to the rule induction paradigm has been proposed [27], [28], which rely on standard techniques for the synthesis of digital circuits [29], [30], [31], [32]. In fact, any binary classification problem can be solved by a proper Boolean function f, which can be always written in Disjunctive Normal Form (DNF), i.e., a logical sum of AND operations [33]. Each of these AND operations can be viewed as an `if-then` rule, whose antecedents and consequent are its inputs and output, respectively. The combination of these rules univocally defines the unknown function f.

Since the goal of methods for the synthesis of digital circuits 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 the rules for any binary classification problem. Unfortunately, classical methods for Boolean function reconstruction do not care about the output assigned to input patterns not belonging to the given training set. Their target is to obtain the simplest AND-OR expression that is able to satisfy all the input-output relations provided initially.

1041-4347/02/$17.00 © 2002 IEEE

To overcome this drawback proper heuristic solutions have been introduced in the MINI method [32], giving rise to the R-MINI algorithm [28], best suited for rule induction. However, the new iterative approach can increase the execution time of the overall procedure. On the other hand, the computational burden required by other available methods [29], [30], [33] that extensively examine the whole input space, prevents their employment in the treatment of most real-world applications.

In this paper, we propose the adoption of a new technique, called Hamming Clustering (HC), for rule induction in classification problems with binary inputs. It is essentially a method for the synthesis of digital circuits. It is able to achieve performances comparable to those of artificial neural networks, in terms of both efficiency and efficacy, at least when the structure is logically analogous to usual real problems in a physical context (otherwise, the preprocessing method described in [34] is needed). HC allows a direct implementation on a physical support [35] since it does not require variable weights with floating point precision [36], [37].

Theoretical results [35] ensure that HC has a polynomial computational cost $O(n^2cs + nc^2)$, where n is the dimension of the input space, s is the size of the given training set, and c is the total number of AND ports in the resulting digital circuit. This upper bound can be lowered if sorting techniques are used throughout the procedure. Also, the amount of memory required is small and shows a polynomial asymptotic behavior $O(n(c+s))$.

Classification problems with discrete or real-valued inputs can also be addressed by HC through the employment of proper coding [1, chap. 2]. For example, in the widely used thermometer code a real variable is specified by a string of bits with fixed length. The number of positive values, always shifted to the left, is proportional to the relative magnitude of the continuous variable itself. With this coding, the ordering of the real axis is preserved, thus allowing a simple direct interpretation of the AND expressions generated by HC; each of them is equivalent to the intersection of inequalities fixing thresholds for the antecedent real inputs.

The performances of HC have been tested through simulations on three artificial benchmarks concerning the reconstruction of AND-OR expressions (possibly in the presence of noise). Furthermore, the well known Monk's problems [38], as well as real-world data sets extracted by the StatLog [39] and the UCI [40] repositories have been examined, comparing the performances of HC with those of other classification algorithms. In particular, the Wisconsin Breast Cancer Database [41] has been analyzed and a set of rules underlying the diagnostic process has been generated.

The structure of this paper is as follows: A general description of the proposed method is presented in Section 2, while Section 3 contains a detailed analysis of the functioning of HC. Section 4 and Section 5 report the results of the simulations performed on the above mentioned artificial and real-world benchmarks. Finally, Section 6 draws some conclusions about the work presented here.

2 The Procedure of Hamming Clustering

Let S be a training set containing s samples (x_j, y_j), $j = 1, \ldots, s$, associated with a single output binary classification problem. The input patterns x_j have n Boolean components, denoted with x_{ji}, $i = 1, \ldots, n$. The integer values -1 and $+1$ will be used in the following to code the two binary states; with this choice, the input patterns $x \in \{-1, +1\}^n$ are placed at the vertices of an n-dimensional hypercube centered at the origin of the space $\mathbb{R}^\propto$.

To simplify the notations, a binary vector will often be written in string form by substituting symbols "+" and "−" to integers $+1$ and -1; in this way, the vector $(-1, -1, +1, -1)$ and the string '$--+-$' refer to the same pattern.

We can associate with our classification problem a Boolean function $f : \{-1, +1\}^n \to \{-1, +1\}$, which has generated the samples of the given training set, possibly in the presence of noise. Such a function f subdivides the vertices of the hypercube $\{-1, +1\}^n$ into two subsets H^+ and H^-, according to the corresponding output:

$$\begin{aligned} H^+ &= \{x \in \{-1, +1\}^n \mid f(x) = +1\} \\ H^- &= \{x \in \{-1, +1\}^n \mid f(x) = -1\}. \end{aligned} \tag{1}$$

It follows that $H^+ \cup H^- = \{-1, +1\}^n$ and $H^+ \cap H^- = \emptyset$.

In a similar way, the training set S can be subdivided into two subsets S^+ and S^-:

$$\begin{aligned} S+ &= \{x_j \in \{-1, +1\}^n \mid (x_j, y_j) \in S, y_j = +1\} \\ S^- &= \{x_j \in \{-1, +1\}^n \mid (x_j, y_j) \in S, y_j = -1\}. \end{aligned}$$

Hence, we have by definition $S^+ \cup S^- = S$.

Since f is a Boolean function, the class for these input patterns are univocally determined, but, in many real-world problems, the training set is generated in the presence of noise. Due to this effect, S can contain *ambiguous* input patterns, whose associated output is not unique. In this case, we say that the training set S is not *consistent* and $S^+ \cap S^- \neq \emptyset$.

The main goal of the procedure of *Hamming Clustering* (HC) is to generate, within a reasonable computational time, a consistent set of rules synthesizing the knowledge embedded in the underlying Boolean function f. This technique proceeds by pattern: A sample of the training set is randomly chosen at each iteration and one or more clusters in the input space are generated starting from that sample and assigned to the same class. Since the method can subsequently work on samples belonging to different classes, rule generation is based on a particular kind of competitive learning.

HC can be viewed as an algorithm for the construction of digital circuits; therefore, its behavior will be analyzed by adopting the same formalism employed to describe classical techniques [31]. Let us denote with I_n the set $\{1, 2, \ldots, n\}$ of the first n positive integers; the following definition plays a basic role:

Definition 1. *Let $x \in \{-1, +1\}^n$ be an input pattern and $L \subset I_n$ a subset of indices of its components having size $l = |L|$. The set $C_l(x, L)$ given by:*

$$C_l(x, L) = \{z \in \{-1, +1\}^n \ : \ z_i = x_i \ \text{ for every } i \notin L\}$$

will be called l-cube (or simply cube) with vertex x and generator L.

As one can note, the elements of the set $C_l(x, L)$ are exactly 2^l and are placed at the vertices of an l-dimensional hypercube in the input space (from which the name l-cube introduced in the previous definition). In particular, we have the following extreme cases:

$$C_0(x, \emptyset) = x, \quad C_n(x, I_n) = \{-1, +1\}^n.$$

An l-cube $C_l(x, L)$ can have different notations according to the vertex x chosen (among the possible 2^l), but each of them contains the same generator L. The string form of an l-cube is obtained by putting the *don't care* symbol "0" in the positions associated with the elements of the generator L. For example, in the case $n = 4$, if $x = '+--+'$ and $L = \{2, 4\}$, we have:

$$C_2(x, L) = '+0-0' = \left\{ \begin{matrix} + & - & - & - \\ + & - & - & + \\ + & + & - & - \\ + & + & - & + \end{matrix} \right\}, \tag{2}$$

for which

$$\begin{aligned} C_2(x, L) &= C_2('+--', L) = C_2('++--', L) \\ &= C_2('++-+', L) = '+0-0'. \end{aligned}$$

The well-known definition of Hamming distance $d_H(x, z)$ between two input patterns x and z (given by the number of different bits),

$$d_H(x, z) = \sum_{i=1}^{n} |x_i - z_i|/2, \tag{3}$$

can then be easily extended to the case of two cubes [31].

Definition 2. *The Hamming distance between the cubes $C_l(x, L)$ and $C_k(z, K)$ is given by*

$$d_H(C_l(x, L), C_k(z, K)) = \sum_{i \notin L \cup K} |x_i - z_i|/2\,. \tag{4}$$

In the case $L = K = \emptyset$, (4) gives the usual definition (3) employed for the strings of binary values. If either L or K is empty, we obtain the relation for the Hamming distance between an input pattern and a cube.

As is known, with any l-cube $C_l(x, L)$ can be associated an AND operation that provides output $+1$ only when the input patterns contained in $C_l(x, L)$ are presented. It is sufficient to perform the logical product of the components having index $i \notin L$, possibly preceded by the NOT operation if $x_i = -1$. For example, the logical product $x_1 \bar{x}_3$ can be associated with the cube $'+0-0'$; it can be easily seen that this operation provides output $+1$ only for the input patterns listed in (2).

This property is exploited by every classical method for the synthesis of digital circuits, and also by HC, to obtain the resulting set of rules. During the training process, cubes in the input space are generated by employing as vertices samples randomly chosen in the training set. The corresponding generator L is determined by adopting proper criteria that tend to improve the generalization ability of the final form. In this way, two collections of cubes C^+ and C^- which approximate the sets H^+ and H^- associated with the unknown Boolean function f are constructed.

HAMMING CLUSTERING

1. Choose at random a sample $(\boldsymbol{x}, y)$ in the training set S.
2. Build a cube having a vertex in $\boldsymbol{x}$.
   Remove $(\boldsymbol{x}, y)$ from S.
   If the construction is not complete, go to Step 1.
3. Prune the set(s) of cubes generated.

Fig. 1. General procedure followed by Hamming Clustering.

The rationale of this approach is based on the following property, which is found in many real-world classification problems; the smaller is the Hamming distance between two input patterns, the greater is the probability that they belong to the same class. As one can note, the construction of cubes in the input space is equivalent to the generation of clusters of patterns that are close according to the Hamming distance. The adoption of proper criteria allows us to further emphasize this aspect; for this reason, the proposed technique has been named Hamming Clustering. When the property above is not verified, a simple preprocessing of the input patterns allows the successful application of HC [34]. Note, however, that in case of continuous data, other clustering algorithms [42], [43], [44], [45] may lead to better results; the fuzzy approach may also be adopted to improve the final result [16], [46], [47].

The set of rules to be generated is not unique, if, as usual, the whole truth table of f is not completely specified in S. Furthermore, there can exist ambiguous input patterns in the training set, which are normally removed from S before starting the generation of the AND-OR expression. Since the procedure followed by HC expands each class in a competitive way, the proposed algorithm can also tackle binary classification problems with nonconsistent training sets.

3 Algorithm Description

The algorithm followed by HC is reported in Fig. 1; at Step 1, an input pattern x is randomly chosen in the training set S and is subsequently used at Step 2 for the construction of clusters (cubes) of vertices of the hypercube $\{-1, +1\}^n$. These steps are repeatedly executed and form the basic iteration of the method. A final pruning phase (Step 3) has the aim of selecting from C^+ and C^- the most significant cubes to be employed in the construction of the resulting DNF expression.

The computational kernel of HC is given by the generation of clusters having a vertex in the input patterns x selected at Step 1. Suppose without loss of generality that $x \in S^+$; since the sets C^+ and C^- are constructed in an incremental way, their separation is maintained if the l-cube $C_l(x, L)$ generated in the current iteration satisfies the following two conditions:

METHOD FOR THE GENERATION OF CUBES IN HC

1. Compute the r^- Hamming distances d_j, for $j = 1, \ldots, r^-$, between $\boldsymbol{x}$ and the patterns $\boldsymbol{x}_j \in R^-$. Let D_j be the subset of I_n containing the d_j indices of the components that differ in $\boldsymbol{x}$ and in $\boldsymbol{x}_j$:
   $D_j = \{i \in I_n \mid x_i \neq x_{ji},\ \boldsymbol{x}_j \in R^-\}$
   for $j = 1, \ldots, r^-$.
2. Compute the c^- Hamming distances d_j, for $j = r^- + 1, \ldots, r^- + c^-$, between $\boldsymbol{x}$ and the cubes $C_{k_j}(\boldsymbol{u}_j, K_j) \in C^-$. Let D_j be the subset of $I_n \setminus K_j$ containing the d_j indices of the components that differ in $\boldsymbol{x}$ and in $\boldsymbol{u}_j$:
   $D_j = \{i \in I_n \setminus K_j \mid x_i \neq u_{ji},\ C_{k_j}(\boldsymbol{u}_j, K_j) \in C^-\}$
   for $j = r^- + 1, \ldots, r^- + c^-$.
3. Extract a subset $L \subset I_n$ such that $D_j \setminus L \neq \emptyset$ for every $j = 1, \ldots, r^- + c^-$.
   The l-cube $C_l(\boldsymbol{x}, L)$ satisfies conditions (6).

Fig. 2. Procedure followed by Hamming Clustering for the generation of a cube having a vertex in the input pattern x.

$$C_l(x, L) \cap S^- = \emptyset,$$
$$C_l(x, L) \cap C_k(u, K) = \emptyset \quad \text{for every } C_k(u, K) \in C^-.$$

The first condition can possibly be relaxed if the training set S is not consistent ($S^+ \cap S^- \neq \emptyset$). In this case, we can accept that $C_l(x, L) \cap S^-$ contains a subset of ambiguous input patterns, that is

$$\begin{aligned} &C_l(x, L) \cap S^- \subset S^+, \\ &C_l(x, L) \cap C_k(u, K) = \emptyset \quad \text{for every } C_k(u, K) \in C^-. \end{aligned} \tag{5}$$

Note that these two conditions are not disjoint; in fact, if C^- is not empty, every cube $C_k(u, K) \in C^-$ has common elements with S^-. To avoid a redundant check of consistency let us consider the sets B^+ and B^- containing the input patterns covered by the clusters in C^+ and C^-:

$$B^+ = \{z \in C_k(u, K) \ : \ C_k(u, K) \in C^+\},$$
$$B^- = \{z \in C_k(u, K) \ : \ C_k(u, K) \in C^-\}$$

Furthermore, let us denote with R^+ and R^- the nonambiguous subsets of S^+ and S^- which do not overlap with the elements of C^+ and C^-, respectively:

$$R^+ = S^+ \setminus (S^- \cup B^+), \qquad R^- = S^- \setminus (S^+ \cup B^-).$$

Condition (5) can easily be written in terms of R^- and B^-

$$C_l(x, L) \cap R^- = \emptyset, \qquad C_l(x, L) \cap B^- = \emptyset. \tag{6}$$

The procedure employed by HC for the generation of a cube having a vertex in $x \in S^+$ is reported in Fig. 2, where we have denoted with x_j, $j = 1, \ldots, r^-$, the jth element of the set R^-, and with $C_{k_j}(u_j, K_j)$, $j = r^- + 1, \ldots, r^- + c^-$, the cubes included in C^-.

At Step 1, the Hamming distances d_j between the input pattern x previously chosen and the elements $x_j \in R^-$ are computed. The sets D_j, with $j = 1, \ldots, r^-$, contain the indices of the components having different value in x and in x_j. A similar task is performed at Step 2 to obtain the Hamming distances d_j, for $j = r^- + 1, \ldots, r^- + c^-$, between x and the cubes $C_{k_j}(u_j, K_j) \in C^-$. The associated sets of indices D_j are directly determined by picking the integers $i \in I_n \setminus K_j$ for which $x_i \neq u_{ji}$.

The construction of an l-cube $C_l(x, L)$ verifying conditions (6) is performed at Step 3 by finding a generator L which satisfies the constraints

$$D_j \setminus L \neq \emptyset \quad \text{for every } j = 1, \ldots, r^- + c^-. \tag{7}$$

The correctness of this approach is ensured by the following theorem, whose proof is contained in [35].

Theorem 1. *The l-cube $C_l(x, L)$, with $x \in S^+$, verifies (6) if and only if the generator L is such that $D_j \setminus L \neq \emptyset$, for every $j = 1, \ldots, r^- + c^-$.*

Note that, as a consequence of (7), if a set D_j contains only one element, that index must not be included in the generator L of the cube to be constructed. The wide freedom in the choice of the generator L allows us to define additional criteria that can influence the generalization ability of the resulting set of rules. Two examples of these criteria are the following:

1. *Largest Cube (LC) criterion*. The size of the generator L chosen at Step 3 of Fig. 2 must be maximum. In this way, the target of HC is to construct a minimal AND-OR expression for the unknown Boolean function, likewise classical methods for digital synthesis. A near-optimal greedy procedure that implements this criterion is presented in Fig. 3. Again the input pattern x is supposed to have a corresponding output $y = +1$.
2. *Maximum-covering Cube (MC) criterion*. The number of samples of S included in the cube to be generated must be maximum. This criterion privileges the generality of induced rules with respect to their simplicity. Best results in terms of accuracy are obtained with this criterion; it can be implemented through the greedy procedure shown in Fig. 4.

Through the repeated execution of Steps 1-2 of HC (Fig. 1), the construction of two sets of cubes C^+ and C^- is

CLUSTER GENERATION WITH THE LARGEST CUBE CRITERION

1. Set $L = I_n$, $l = n$, $J = \{1, 2, \ldots, r^- + c^-\}$.
2. If an index $i \in L$ is the only element of some D_j with $j \in J$, go to Step 3,
   else find the index $i \in L$ that is included in the greatest number of sets D_j.
3. Remove from J all the indices j for which $i \in D_j$. Remove i from L and set $l = l - 1$. If $J \neq \emptyset$, go to Step 2.

Fig. 3. Greedy procedure followed by the largest cube criterion for the generation of clusters.

CLUSTER GENERATION WITH THE MAXIMUM COVERING CUBE CRITERION

1. Construct the sets $D_j^+ \subset I_n$ containing the indices of the components that differ in $\boldsymbol{x}$ and in the elements $\boldsymbol{x}_j \in R^+$:
   $D_j^+ = \{i \in I_n \mid x_i \neq x_{ji},\ \boldsymbol{x}_j \in R^+\}$
   for $j = 1, \ldots, r^+$. Set $L = \emptyset$ and $l = 0$.
2. Let J be the set of the indices $i \in I_n \setminus L$ that lead to $(l+1)$-cubes $C_{l+1}(\boldsymbol{x}, L \cup \{i\})$ satisfying condition (7). If $J = \emptyset$ the construction of the l-cube $C_l(\boldsymbol{x}, L)$ is concluded.
3. Find the index $i \in J$ that is included in the greatest number of sets D_j^+ associated with the input patterns $\boldsymbol{x}_j \in R^+$. Add i to L, set $l = l + 1$ and go to Step 2.

Fig. 4. Greedy procedure followed by the maximum covering cube criterion for the generation of clusters.

achieved. In general, they contain redundant elements that can be removed without changing the behavior of the resulting set of rules. Thus, a pruning phase can be useful to optimize the complexity of the final form.

To better describe the main pruning methods employed in HC, we introduce the following definition:

Definition 3. *The set Q given by*

$$Q = \{x \in C_l(u, L) \mid (x, y) \in S\}$$

will be called cover set *of the l-*cube *$C_l(u, L)$. The number q of input patterns contained in Q will be named* covering *of $C_l(u, L)$.*

To minimize the number of rules generated by HC, we select in C^+ and C^- a reduced (possibly minimum) subset of clusters that satisfies all the input-output pairs contained in the given training set S. To this end, we choose the cubes having maximum covering and in suborder those with greater dimension. Two different rules have been adopted in our simulations to perform this selection: the minimal pruning and the threshold pruning.

Minimal pruning. The easiest way to reduce the size of C^+ and C^- is to find a minimum subset of cubes that correctly classify all the input patterns in S. Several techniques have been proposed in the literature to perform this task in the synthesis of digital circuits [30], [31], [32]. A simple suboptimal greedy procedure extracts the clusters with maximum covering one at a time. At each extraction, only the samples of S not included in the cubes already selected are considered for the construction of the cover sets. Breaks are tied by examining the original coverings.

Threshold pruning. In some classification problems, it can be advantageous to maintain in C^+ and C^- only the cubes whose covering exceeds a fixed threshold τ. Whereas this approach can misclassify some input patterns of S, which are included in clusters with low covering, the resulting generalization ability may increase. This pruning method is similar to that employed in C4.5 to avoid rules with low information content [19]. In our simulations, we have always used threshold pruning with $\tau = 9$.

An example can help us understand the behavior of HC in a practical situation. Suppose we have to reconstruct the Boolean function $f(x) = x_1\overline{x_3} + x_2x_4$ starting from the following two set of input patterns

$$S^+ = \begin{Bmatrix} - & + & - & + \\ - & + & + & + \\ + & - & - & - \\ + & + & - & + \end{Bmatrix}, \quad S^- = \begin{Bmatrix} - & - & - & + \\ - & + & - & - \\ + & - & + & + \\ + & + & + & - \end{Bmatrix}$$

obtained through a noiseless sampling of f in the domain $\{-1, +1\}^4$. To simplify the description, consider the case where only the cubes in C^+ have to be generated through the procedure of Fig. 1; consequently, the sample x at Step 1 is randomly chosen only within S^+. Furthermore, during the whole procedure we have $C^- = \emptyset$, $c^- = 0$, and $R^- = S^-$ (since the training set is consistent); there is no need to take into account the distances between the selected pattern x and the cubes of C^-.

Assume that the LC criterion is employed and only one cube is generated at any iteration. If the input pattern $x = {}'+--- {}' \in S^+$ is chosen at the first iteration, the Hamming distances d_j become equal $d_1 = d_2 = d_3 = d_4 = 2$ and are associated with the following sets D_j:

$$\begin{aligned} D_1 &= \{1, 4\}\ , \quad D_2 = \{1, 2\}\ , \\ D_3 &= \{3, 4\}\ , \quad D_4 = \{2, 3\}. \end{aligned} \tag{8}$$

The application of the procedure in Fig. 3 initially sets $L = J = \{1, 2, 3, 4\}$; since each of these indexes scores the same number of occurrences (two) in the sets D_j, with $j \in J$, the choice of i at Step 2 is performed at random. Suppose we select $i = 3$; the execution of Step 3 will lead to $L = \{1, 2, 4\}$ and $J = \{1, 2\}$. Then, a valid generator $L = \{2, 4\}$ is obtained through the elimination of the index 1, which is included in all the remaining sets D_j. In this way we have determined the 2-cube $C_2({}'+--- {}', \{2, 4\}) = {}' +0-0 {}'$ which will be inserted in C^+. The sets B^+ and R^+ become after this change:

$$B^+ = \begin{Bmatrix} + & - & - & - \\ + & - & - & + \\ + & + & - & - \\ + & + & - & + \end{Bmatrix}, \ R^+ = \begin{Bmatrix} - & + & - & + \\ - & + & + & + \end{Bmatrix}.$$

Now, suppose the input pattern $x = {}'-+-+ {}'$ is chosen at second iteration; the Hamming distances d_j and the corresponding sets D_j are consequently

$$\begin{aligned} &d_1 = d_2 = 1\ , \quad d_3 = d_4 = 3\ , \quad D_1 = \{2\}\ , \\ &D_2 = \{4\}\ , \quad D_3 = \{1, 2, 3\}\ , \quad D_4 = \{1, 3, 4\}. \end{aligned}$$

The application of the LC criterion therefore leads to the subsequent elimination of the indices two and four from the initial generator $L = \{1, 2, 3, 4\}$, since they are the only elements of D_1 and D_2, respectively. Thus, the 2-cube $C_2({}'-+-+ {}', \{1, 3\}) = {}' 0+0+ {}'$ is generated and we have

$$C^+ = \begin{Bmatrix} + & 0 & - & 0 \\ 0 & + & 0 & + \end{Bmatrix},\ B^+ = \begin{Bmatrix} - & + & - & + \\ - & + & + & + \\ + & - & - & - \\ + & - & - & + \\ + & + & - & - \\ + & + & - & + \\ + & + & + & + \end{Bmatrix},\ R^+ = \emptyset$$

If we continue with the generation of new cubes having vertices in $'-+++'$ and in $'++-+'$, we can create the undesired cluster $'-0+0'$, thus yielding the following final set C^+

$$C^+ = \begin{Bmatrix} + & 0 & - & 0 \\ 0 & + & 0 & + \\ - & 0 & + & 0 \end{Bmatrix}.$$

The application of the minimal pruning allows to reconstruct the exact DNF expression for the unknown function $f(x)$; as a matter of fact, the first selection extracts the cube $'0+0+'$ that has covering $q=3$. After the removal of elements $'-+-+'$, $'-+++'$, and $'++-+'$ from S^+, we note that the remaining pattern $'+---'$ is only covered by the cluster $'+0-0'$. The pruned set C^+ includes therefore the two cubes $'0+0+'$ and $'+0-0'$, which yield the correct AND-OR expression $x_1\overline{x_3} + x_2x_4$ for $f(x)$.

The application of the MC criterion simplifies the execution of HC. Suppose again the pattern $x = '+---'$ is selected at the first iteration with $R^+ = S^+$ and $B^+ = \emptyset$. The sets D_j in (8) will be produced; then, the first step of the procedure in Fig. 4 requires the computation of the sets D_j^+ containing the indices of the components that differ in x and in the elements $x_j \in R^+$. We have, in our example,

$$D_1^+ = \{1,2,4\}\ ,\ D_2^+ = \{1,2,3,4\}\ ,\ D_3^+ = \emptyset\ ,\ D_4^+ = \{2,4\}$$

Now, we can directly build the set $J = \{1,2,3,4\}$ of the indices that can be added to the generator $L = \emptyset$ without violating conditions (7). Since elements 2 and 4 are included in all the three nonempty sets D_j^+, one of them is randomly selected to be inserted in L; suppose we take $L = \{2\}$. A second execution of these steps gives $J = \{4\}$ and allows us to obtain the generator $L = \{2,4\}$ corresponding to the cube '$+0-0$,' which will be inserted in C^+. The sets B^+ and R^+ are again as shown in (9).

If at the second iteration of HC the pattern $x = '-+-+'$ is chosen, the application of the procedure in Fig. 4 produces the generator $L = \{1,3\}$ and the cube $'0+0+'$. After its insertion in C^+ we obtain $R^+ = \emptyset$ and the construction is complete.

Note that, in this case, no spurious clusters have been generated and the pruning phase is not necessary; this may be a signal of a more general result. In fact, simulations have shown that in most of the real-world situations examined the MC criterion leads to a higher accuracy with respect to the LC criterion. For this reason, MC has been employed in all the tests presented in the next section.

4 Artificial Benchmarks

In order to assess the ability of HC in generating predefined set of rules starting from a collection of samples, two different tests on artificial benchmarks have been performed. The first of them deals with the reconstruction of a randomly generated Boolean function, also taking into account the effect of noise in the acquisition of the training set. The second one deals with the well-known Monk's problems [38], where given classification rules are to be identified on the basis of a reduced set of samples.

In the following sections, the collections of rules obtained for each test are presented and discussed, together with the computational cost involved. Comparisons with other methods for the synthesis of digital circuits, such as the procedure of Quine-McCluskey (QMC) [29], [33] and the fast algorithm ESPRESSO (ESP) [31], or with other rule extraction techniques, such as the method described in [8], are also reported. All the execution times refer to a DEC ALPHAStation 500/400 running under operating system DIGITAL UNIX 4.0A.

4.1 Boolean Function Reconstruction

A Boolean function has been generated as the logical sum of a predetermined number h of AND operations. Each of them is built by randomly choosing n_i, $i = 1,\ldots,h$, of the total n inputs, possibly negated with probability 0.5. The whole truth table is obtained, thus resulting in 2^n input-output pattern pairs.

By choosing $n = 10$ and $h = 10$, the following AND-OR expression has been produced

$$\begin{aligned} f(x) = {} & \bar{x}_2\bar{x}_3x_8\bar{x}_9x_{10} + \bar{x}_2\bar{x}_3x_5\bar{x}_6x_7 + \bar{x}_1x_3\bar{x}_4x_6\bar{x}_8\bar{x}_9 \\ & + \bar{x}_1x_2x_4x_5\bar{x}_7\bar{x}_8\bar{x}_{10} + x_1\bar{x}_2\bar{x}_4x_6\bar{x}_9 + \bar{x}_1\bar{x}_7\bar{x}_9\bar{x}_{10} \\ & + x_1\bar{x}_2x_3\bar{x}_5x_6x_{10} + \bar{x}_2x_4x_5x_6\bar{x}_7x_8\bar{x}_{10} + x_2\bar{x}_3\bar{x}_7 \\ & + x_1\bar{x}_2\bar{x}_4\bar{x}_6\bar{x}_7\bar{x}_9\bar{x}_{10}. \end{aligned} \tag{10}$$

Three experiments are then performed. In the first one, all of the examples are fed into HC, aiming to reconstruct the unknown Boolean function $f(x)$. The execution of HC with minimal pruning on the whole truth table, containing 1,024 examples, has produced the following AND-OR expression:

$$\begin{aligned} g_1(x) = {} & \bar{x}_2\bar{x}_3x_8\bar{x}_9x_{10} + \bar{x}_2\bar{x}_3x_5\bar{x}_6x_7 + \bar{x}_1x_3\bar{x}_4x_6\bar{x}_8\bar{x}_9 \\ & + \bar{x}_1x_2x_4x_5\bar{x}_7\bar{x}_8\bar{x}_{10} + x_1\bar{x}_2\bar{x}_4x_6\bar{x}_9 + \bar{x}_1\bar{x}_7\bar{x}_9\bar{x}_{10} \\ & + x_1\bar{x}_2x_3\bar{x}_5x_6x_{10} + \bar{x}_2x_4x_5x_6\bar{x}_7x_8\bar{x}_{10} + x_2\bar{x}_3\bar{x}_7 \\ & + \bar{x}_2\bar{x}_4\bar{x}_7\bar{x}_9\bar{x}_{10}. \end{aligned} \tag{11}$$

As one can see, the logical products in (10) and (11) are identical, except for the last one. In fact, HC has been able to find a valid expression for the last AND operation, which is even simpler than the original one, taking advantage from the overlap with other logical products. This shows the ability of HC to perform rule simplification.

Such a totally correct result has been obtained in only 0.15 seconds of CPU time. QMC and ESP generate the same Boolean function (11) within an equivalent execution time.

In the second experiment, only half of the 1,024 examples have been fed into HC, thus obtaining the following $g_2(x)$ with 11 AND operations

$$\begin{aligned} g_2(x) = {} & \bar{x}_2\bar{x}_3x_8\bar{x}_9x_{10} + \bar{x}_2\bar{x}_3x_5\bar{x}_6x_7 + \bar{x}_1x_3\bar{x}_4x_6\bar{x}_8\bar{x}_9 \\ & + \bar{x}_1x_2x_4\bar{x}_7\bar{x}_8\bar{x}_{10} + x_1\bar{x}_2\bar{x}_4x_6\bar{x}_9 + \bar{x}_1\bar{x}_7\bar{x}_9\bar{x}_{10} \\ & + \bar{x}_3x_5x_6\bar{x}_7x_8\bar{x}_{10} + \bar{x}_2x_3x_4x_6\bar{x}_7x_8\bar{x}_{10} \\ & + x_1\bar{x}_2x_3\bar{x}_5x_6x_{10} + x_2\bar{x}_3\bar{x}_7 + \bar{x}_2\bar{x}_4\bar{x}_7\bar{x}_9\bar{x}_{10}. \end{aligned} \tag{12}$$

It can be seen that seven logical products are exactly reconstructed and $x_1\bar{x}_2\bar{x}_4\bar{x}_6\bar{x}_7\bar{x}_9\bar{x}_{10}$ is again correctly reduced to the same simpler expression. On the other side, the fourth logical product $\bar{x}_1x_2x_4x_5\bar{x}_7\bar{x}_8\bar{x}_{10}$ of f is wrongly reduced to $\bar{x}_1x_2x_4\bar{x}_7\bar{x}_8\bar{x}_{10}$ because the information contained in the training set is incomplete.

Finally, the eighth AND operation $\bar{x}_2x_4x_5x_6\bar{x}_7x_8\bar{x}_{10}$ of f is covered by the second, the eighth, and the ninth logical products of $g_2(x)$. The total number of errors in the 512 patterns not used in the training phase is only seven, corresponding to a percentage of 1.3 percent. As a comparison, QMC and ESP generate 26 and 57 AND operations, leading to a corresponding error percentage of 10 percent and 30 percent, respectively.

In the third experiment, the output is randomly negated in 26 (5 percent) of the 512 examples included in the training set, simulating the effect of noise. The execution of HC with threshold pruning generates a Boolean function $g_3(x)$ with 34 logical products, three of which are contained in the expression (10). The percentage of correct output for the whole truth table of $f(x)$ is about 95 percent. The computational time required for the execution of HC is only 0.13 seconds.

The error percentages of QMC and ESP remains almost the same as in the previous experiment, whereas the complexity of the digital circuits generated increases to 42 and 72 logical products, respectively. Remember that the target of ESP is to obtain within a reduced execution time a DNF expression satisfying all the samples in the training set; consequently, it looks for a suboptimal reconstruction of the unknown Boolean function without caring about generalization. Better results are obtained by QMC, which analyzes all the consistent AND operations, thus requiring a higher computational cost. Here, the pruning method plays a crucial role, having to choose among a high number of logical products; again, the standard technique is not able to recover the loss of information involved in the last two experiments.

4.2 Monk's Problems

A classical set of benchmarks, widely used in the artificial intelligence literature, is Monk's problems [38]. These are discrete classification problems defined in a "robot" domain, described by the following six attributes:

HEAD-SHAPE	$\in$ {ROUND, SQUARE, OCTAGON}
BODY-SHAPE	$\in$ {ROUND, SQUARE, OCTAGON}
IS-SMILING	$\in$ {YES, NO}
HOLDING	$\in$ {SWORD, BALLOON, FLAG}
JACKET-COLOR	$\in$ {RED, YELLOW, GREEN, BLUE}
HAS-TIE	$\in$ {YES, NO}.

With this characterization, 432 different robots can be obtained.

The aim of each proposed benchmark is the determination of a general classification rule starting from a limited set of samples. The first two Monk's problems are noise-free, whereas in the third case, the outputs of the training set can undergo small changes from their correct value. This last test can therefore provide a measure of the robustness of HC.

Since the inputs are discrete, an only-one code has been applied to allow the generation of an AND-OR expression. The resulting binary input patterns contain 15 components, each of which is associated with the presence of a particular attribute in the corresponding robot.

4.2.1 Monk #1

The first Monk's problem is defined by the following rule:

$$\begin{aligned} & \text{if (HEAD-SHAPE} = \text{BODY-SHAPE} \\ & \quad \text{or JACKET-COLOR} = \text{RED)} \\ & \text{then OUTPUT} = \text{TRUE}. \end{aligned} \tag{13}$$

The execution of HC on the training set containing 124 input-output pairs requires 0.06 seconds of CPU time and leads to the following four expressions:

$$\begin{aligned} & \text{JACKET-COLOR} = \text{RED} \\ & \text{HEAD-SHAPE} = \text{BODY-SHAPE} = \text{OCTAGON} \\ & \text{HEAD-SHAPE} = \text{BODY-SHAPE} = \text{SQUARE} \\ & \text{HEAD-SHAPE} = \text{BODY-SHAPE} = \text{ROUND}. \end{aligned}$$

As one can note, they are exactly equivalent to the rule (13). As a comparison, the method extracting rules from neural networks described in [8] obtained 12 more hidden units, each representing a different rule, besides the four above.

4.2.2 Monk #2

The second Monk's problem has a more complex formulation:

$$\begin{aligned} & \text{if (exactly two of the six attributes} \\ & \quad \text{have their } first \text{ value)} \\ & \text{then OUTPUT} = \text{TRUE}. \end{aligned} \tag{14}$$

It can be easily seen that the logical complexity of this rule is equivalent to that of the parity problem [48]. In this case, patterns with low Hamming distance have different output with high probability; for this reason HC is in great trouble in finding the underlying rule (14).

In spite of this, HC generates an AND-OR expression containing 32 logical products, less redundant than the result achieved by [8] with 43 hidden rules.

4.2.3 Monk #3

Finally, the third Monk's problem is described by the following rule:

$$\begin{aligned} & \text{if ((JACKET-COLOR} = \text{GREEN} \\ & \quad \text{and HOLDING} = \text{SWORD)} \\ & \quad \text{or (JACKET-COLOR} = \text{not BLUE} \\ & \quad \text{and BODY-SHAPE} = \text{not OCTAGON))} \\ & \text{then OUTPUT} = \text{TRUE}. \end{aligned} \tag{15}$$

TABLE 1
Accuracy and Complexity of the Rule Sets Obtained by HC, C4.5, and RAMP on the StatLog Benchmark

Classification Problem	Evaluation Method	Accuracy			Number of Rules			#Att./Rule	
		HC	C4.5	RAMP	HC	C4.5	RAMP	HC	C4.5
HEART	9 fold on 270	0.463	0.781	—	30.6	11.4	—	4.81	2.68
AUSTRALIAN	10 fold on 690	0.129	0.155	0.139	55.6	11.5	31.2	3.60	2.76
DIABETES	12 fold on 768	0.250	0.270	0.272	43.2	9.4	55.1	1.99	2.58
VEHICLE	9 fold on 846	0.307	0.266	0.289	128.6	26.1	55.6	4.82	4.03
GERMAN	10 fold on 1000	0.743	0.985	—	57.2	21.1	—	5.33	2.77
SEGMENT	10 fold on 2310	0.052	0.040	0.033	64.5	28.0	35.6	4.02	3.94
DNA	2000 Tr - 1186 Tst	0.061	0.076	0.049	57.0	34.0	50.0	6.36	4.47
SATIMAGE	4435 Tr - 2000 Tst	0.144	0.150	0.126	318.0	80.0	146.0	7.99	5.41
LETTER	15000 Tr - 5000 Tst	0.135	0.132	0.108	1155.0	570.0	1175.0	9.22	7.64
SHUTTLE	43500 Tr - 14500 Tst	0.0003	0.001	0.001	23.0	20.0	21.0	3.44	3.14

From the total 432 examples, 122 are selected randomly, 5 percent of whom are misclassified, simulating the action of noise in the acquisition of the training set.

The Boolean function found by HC contains 11 logical products and achieves a generalization error of 3 percent, partially recovering the effect of noise and reaching the same performance of the best neural methods tested in [38].

5 Results on Real-World Classification Problems

The performances of HC, in terms of accuracy and complexity, have also been analyzed through a series of experiments on real-world classification problems. Tests contained in the well-known StatLog benchmark [39] have been considered, so as to compare our method with many other pattern recognition techniques, among which rule generation algorithms, such as C4.5 [19], [20] and the algorithm RAMP described in [28], [49], which is based on a similar approach.

Two further experiments deal with real-world databases included in the UCI repository [40]: the Mushroom problem, where the edibility of 23 species of gilled mushrooms in the Agaricus and Lepiota family is to be assessed, and the Wisconsin Breast Cancer database [41], where breast cancer is to be diagnosed on the basis of nine integer features. These two experiments aim at evaluating the significance of the rules generated by HC, also in comparison with other available techniques.

5.1 StatLog Data

Table 1 shows the performances, in terms of accuracy and complexity, obtained by HC on the StatLog benchmark. As a comparison, the corresponding results achieved by two other rule generation techniques, C4.5 and RAMP, are also reported; for other classification methods, see [39]. The StatLog benchmark includes 10 classification problems that raise from different application areas and involve a number of samples starting from a minimum of 270 (Heart) to a maximum of 68,000 (Shuttle).

The accuracy of each classification method (third column of Table 1) is measured by computing the error rate obtained on a separate test set, except for the Heart and the German problems, where a cost matrix has to be applied for weighting misclassified patterns. The complexity of the final classifier is given by the number of rules involved (fourth column) and by the number of attributes used for each rule (last column). This last value is not available for the method RAMP.

Performances on problems with a relatively small training set size (up to 2,310) are evaluated by averaging the results scored on a cross-validation set. Benchmarks involving a higher number of samples are analyzed by counting the number of errors performed on a separate test set containing patterns not considered in the learning phase; the training and the test set for each problem are fixed over the three considered methods, thus providing a common environment for the benchmark. The second column of Table 1 specifies the method employed for computing accuracy and complexity.

Only four problems (Heart, Australian, Diabetes, and German) has a binary output, thus allowing a direct application of the HC algorithm. Since HC is not yet suited for treating multioutput classifiers, the remaining six benchmarks are analyzed by obtaining a separate set of rules for each output value. The final decision on a given pattern is then taken by picking the class associated with the rule at minimum Hamming distance; random choices are employed to break ties. Although this procedure allows HC to deal with multioutput problems, it may lead to poor performances; in particular, the complexity of the resulting rule set may become high, since rules pertaining to different outputs are replicated.

It can be observed that in seven out of 10 cases, HC achieves a better accuracy value with respect to C4.5. However, it requires a higher number of rules to describe every classification problem since, by construction, each rule cannot misclassify patterns belonging to the training set. This prevents the construction of rules with high covering that score a nonnull error rate in the learning phase. For the same reason, the number of attributes per rule is generally higher for HC, although the difference is rarely greater than two. RAMP outperforms both HC and C4.5 when dealing with multiclass training sets, but obtains a lower accuracy than HC in the Australian, the Diabetes and the Shuttle databases (results concerning the Heart and the German problems are not available).

5.2 Mushroom Data Set

From the UCI repository, we have derived a real-world benchmark concerning the classification of 23 species of gilled mushrooms in the Agaricus and Lepiota family. Each species

must be identified as edible or poisonous upon knowledge of 22 nominal variables, which give raise to a binary input pattern containing 111 components after the application of the only-one code. One thousand of the total 8,124 samples (4,208 of which are edible) in the dataset have been fed into HC to obtain the resulting set of rules, which has subsequently been tested on the remaining 7,124 input patterns.

The error rate achieved within one second of CPU time is 0.10 percent lower than that obtained with the HILLARY algorithm [50] reported in the UCI description and slightly better than the corresponding result shown in [27], which refers to a technique derived from ESPRESSO [31].

The digital circuit built by HC contains only two AND ports associated with the following rules which hold for the recognition of edible mushrooms:

Rule 1 (Covering 4,000):
Odor: NOT (creosote foul pungent),
Gill-color: NOT (buff),
Stalk-surface-below-ring: NOT (scaly),
Ring-type: NOT (large), and
Spore-print-color: NOT (green).

Rule 2 (Covering 2,728):
Odor: NOT (creosote foul pungent),
Gill-color: NOT (buff),
Ring-type: NOT (large none),
Spore-print-color: NOT (green), and
Population: NOT (clustered several).

The number of tests necessary to check these rules is 10, slightly more than the eight needed in the two rules shown by Hong [28]. On the other side, the covering of the cubes built by HC is higher.

The application of C4.5 to the same benchmark produced 12 rules, which can be reduced to the following four:

Rule 1 (Covering 2,414):
Odor: none,
Stalk-surface-below-ring: smooth, and
Ring-number: one.

Rule 2 (Covering 800):
Odor: (almond,anise).

Rule 3 (Covering 528):
Ring-number: two, and
Spore-print-color: white.

Rule 4 (Covering 456):
Odor: none, and
Stalk-surface-below-ring: fibrous.

Also in this case, the error rate was 0.10 percent. As one can note, the rules obtained by C4.5 are simpler but present a lower covering; as a consequence they can lead to a poor characterization of edible mushrooms.

A ten-fold cross validation on the mushroom benchmark performed with HC has led to a perfect recognition of the two classes, scoring a null error rate. An average error rate of 0.025 percent was obtained when using C4.5.

5.3 Winsconsin Breast Cancer Database

A classical real-world classification problem is the Winsconsin Breast Cancer Database [40], [41], where breast cancer is to be diagnosed on the basis of nine features, assuming integer values in the range $1, \ldots, 10$: clump thickness (x_1), uniformity of cell size (x_2), uniformity of cell shape (x_3), marginal adhesion (x_4), single epithelial cell size (x_5), bare nuclei (x_6), bland chromatin (x_7), normal nucleoli (x_8), and mitoses (x_9). The available input-output pairs are 699, of which 458 (65.5 percent) are benign and 241 (34.5 percent) malignant.

Since 16 input patterns include missing data, the corresponding pairs have been removed. The remaining 683 samples have been subdivided into training (372 patterns) and test set (311 patterns) according to the proportion used in [51]. Since the application of HC requires binary inputs, a thermometer code with length nine has been used to translate the integer values in the database, thus obtaining input patterns with 81 binary components.

In [35], we have shown the generalization ability of HC in correctly classifying previously unseen input patterns. In particular, the error rate obtained through a ten-fold cross validation is below 3 percent, which is comparable with analogous results obtained with other rule generation techniques [6], [12], [52]. Here, we are interested in verifying the capability of HC to explicitly extract a set of rules underlying the problem of diagnosing breast cancer.

The execution of HC with minimal pruning yields the generation of six rules, two of which describe only one example each. The CPU time required to obtain this result is 0.25 seconds. The four significant rules producing a benign diagnosis are:

$$\begin{aligned}
&x_1 \leq 8 \text{ AND } x_4 \leq 7 \text{ AND } x_5 \leq 5 \text{ AND } x_6 \leq 2 \\
&\quad \text{AND } x_7 \leq 9 \text{ AND } x_8 \leq 8 \text{ AND } x_9 \leq 3 \\
&x_1 \leq 8 \text{ AND } x_3 \leq 2 \text{ AND } x_4 \leq 7 \text{ AND } x_5 \leq 5 \\
&\quad \text{AND } x_6 \leq 3 \text{ AND } x_8 \leq 8 \\
&x_3 = 1 \text{ AND } x_4 \leq 9 \text{ AND } x_8 \leq 8 \\
&x_1 \leq 8 \text{ AND } x_4 \leq 7 \text{ AND } x_5 \leq 5 \text{ AND } x_6 \leq 4 \\
&\quad \text{AND } x_7 \leq 2 \text{ AND } x_8 \leq 8 \text{ AND } x_9 \leq 3.
\end{aligned} \tag{16}$$

The direct application of these four rules produces a generalization error of 4.5 percent (14 misclassified patterns), corresponding to a total error on the whole dataset of about 2 percent.

The employment of C4.5 allows to reduce the number of rules needed, but the total error increases to 2.78 percent. The rules generated by C4.5 for a benign diagnosis are the following two:

$$\begin{aligned}
&x_2 \leq 3 \text{ AND } x_6 \leq 2 \\
&x_1 \leq 4 \text{ AND } x_2 \leq 3 \text{ AND } x_3 \leq 3.
\end{aligned}$$

We remind that the higher complexity of the rule set generated by HC follows from the constraint of scoring a null error rate on the samples of the training set.

It is interesting to consider a parallel result of Drago and Ridella [53] concerning a measure of the relative input saliency obtained via the application of the method described in [54]. The joint use of the two more relevant features, namely the sixth and the first, allow them to achieve an error percentage of 3.95 percent. It is important to note that this result is obtained through the training on the whole data set. Furthermore, the resulting classifier is linear.

The search for a similar rule among the four listed in (16) produces two simple disjoint quantitative thresholds for the variables $x_1 \leq 8$ and $x_6 \leq 4$. This leads to a generalization error of 7.3 percent when the learning is performed on the training set only. However, this single rule is not the best possible one; the addition of the three more used other conditions in the four found rules (16), $x_4 \leq 9$, $x_5 \leq 5$, and $x_8 \leq 8$, yields an error percentage of 4.4 percent. As a comparison, the best rule between the two generated by C4.5 scores a total error of 5.9 percent.

6 Conclusions

Hamming Clustering (HC) has been applied to a variety of different benchmarks, illustrating its capabilities of extracting the rules underlying a classification problem from a training set of samples. In the reconstruction of Boolean functions HC has been able to achieve a simplified DNF expression, partially recovering the action of noise possibly affecting the acquisition of the input-output pairs.

In the application to the Monk's problems HC obtains in two of the three cases a consistent set of rules and a generalization error comparable to that of the best neural methods tested in [38]. Only in the second problem, which is parity-like, HC yields only partially correct results.

Simulations on real-world classification problems extracted from the StatLog database point out the wide applicability of HC and the good generalization ability exhibited by the set of rules built, particularly in the cases with two output classes. This is probably due to the MC criterion for the construction of cubes, which tries to maximize the covering of each cluster, instead of simply minimizing the complexity of the resulting DNF expression. On the other hand, this can increase the number of attributes involved in each rule.

The same effect is also produced by the constraint of having a null training error when constructing each cube. To satisfy this constraint, it is necessary to lower the dimension of the cluster by adding new inputs with fixed values, which leads to more specialized rules with a higher number of antecedents. As a further consequence of this approach, the size of the rule set generated by HC increases: the results shown in Table 1 for the StatLog benchmark point out these effects.

The application of HC to the Mushroom data set has led to the generation of a reduced set of rules for the recognition of edible mushrooms, which present the highest covering among the rules available in the literature. Validation techniques show also the high accuracy associated with DNF expressions produced by HC. In the Wisconsin Breast Cancer Database, the execution of HC has allowed us to obtain a reduced set of rules solving the corresponding classification problem and outlining the relevant factors to be considered in the medical diagnosis.

These performances arise from the capability of HC to build the DNF expression of any Boolean function. The possible drawback deriving from the necessity of having binary inputs has been overcome by adopting suitable coding. Any logical products can be easily associated with an `if-then` rule, thus characterizing HC also as an expert system. Despite this difficult task of extracting intelligible rules, the computational cost of HC is extremely low.

We are currently developing a new version of the HC algorithm, which is able to treat in a natural way multioutput classification problems, thus allowing the generation of rules associated with two or more output values. At the same time, the possible benefits deriving from loosing the constraint of having a null training error will be analyzed; this can improve the performances of HC when dealing with real world classification problems characterized by a high level of noise.

Acknowledgments

The authors wish to thank the anonymous reviewers for their valuable comments that improved this paper.

References

[1] S.I. Gallant, *Neural Networks Learning and Expert Systems.* Cambridge, Mass.: MIT Press, 1993.
[2] D. Liberati, "Expert Systems: The State of the Art," *The Ligand Quarterly,* vol. 8, pp. 606-611, 1989.
[3] B. Buchanan and E. Shortliffe, *Rule-Based Expert Systems.* Reading, Mass.: Addison-Wesley, 1984.
[4] J. McDermott, "R1: The Formative Years," *AI Magazine,* vol. 2, pp. 21-29, 1981.
[5] R. Andrews, J. Diederich, and A. Tickle, "A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks," *Knowledge-Based Systems,* vol. 8, pp. 373-389, 1995.
[6] I. Taha and J. Ghosh, "Symbolic Interpretation of Artificial Neural Networks," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, pp. 448-463, 1999.
[7] R.M. Goodman, C.M. Higgins, J.W. Miller, and P. Smyth, "Rule-Based Neural Networks for Classification and Probability Estimation," *Neural Computation,* vol. 4, pp. 781-804, 1992.
[8] K.-P. Huber and M.R. Berthold, "Building Precise Classifiers with Automatic Rule Extraction," *Proc. IEEE Int'l Conf. Neural Networks,* pp. III-1263-1268, 1995.
[9] L.M. Fu, *Neural Networks in Computer Intelligence.* McGraw-Hill, 1994.
[10] G.G. Towell and J.W. Shavlik, "The Extraction of Refined Rules Form Knowledge-Based Neural Networks," *Machine Learning,* vol. 13, pp. 71-101, 1993.
[11] R. Setiono, H. Liu, "Symbolic Representation of Neural Networks," *Computer,* vol. 29, pp. 71-77, 1996.
[12] R. Setiono, "Extracting m-of-n Rules from Trained Neural Networks," *IEEE Trans. Neural Networks,* vol. 11, pp. 512-519, 2000.
[13] M. Ishikawa, "Rule Extraction by Successive Regularization," *Neural Networks,* vol. 13, pp. 1171-1183, 2000.
[14] C.T. Lin and C.S.G. Lee, "Neural Network-Based Fuzzy Logic Control and Decision System," *IEEE Trans. Computers,* vol. 40, pp. 1320-1326, 1991.
[15] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On Fuzzy Modeling Using Fuzzy Neural Networks with Back-Propagation Algorithm," *IEEE Trans. Neural Networks,* vol. 3, pp. 801-806, 1992.
[16] P.K. Simpson, "Fuzzy Min-Max Neural Network—Part 2: Clustering," *IEEE Trans. Fuzzy Systems,* vol. 1, pp. 32-45, 1993.
[17] M. Setnes, "Supervised Fuzzy Clustering for Rule Extraction," *IEEE Trans. Fuzzy Systems,* vol. 8, pp. 416-424, 2000.
[18] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees.* Belmont: Wadsworth, 1994.
[19] J.R Quinlan, *C4.5: Programs for Machine Learning.* San Francisco: Morgan Kaufmann, 1994.
[20] J.R. Quinlan, "Generating Production Rules from Decision Trees," *Proc. 10th Int'l Joint Conf. Artificial Intelligence,* pp. 304-307, 1987.
[21] G. Pagallo, "Learning DNF by Decision Trees," *Proc. 11th Int'l Joint Conf. Artificial Intelligence,* pp. 639-644, 1989.
[22] S. Muggleton, *Inductive Logic Programming.* New York: Academic Press, 1992.

[23] S. Muggleton and L. De Raedt, "Inductive Logic Programming: Theory and Methods," *J. Logic Programming,* vols. 19/20, pp. 629-679, 1994.
[24] J.R. Quinlan and R.M. Cameron-Jones, "Induction of Logic Programs: Foil and Related Systems," *New Generation Computing,* vol. 13, pp. 287-312, 1995.
[25] P.R.J. van der Laag and S.-H. Nienhuys-Cheng, "Completness and Properness of Refinement Operators in Inductive Logic Programming," *J. Logic Programming,* vol. 34, pp. 201-225, 1998.
[26] J. Fürnkranz, "Pruning Algorithms for Rule Learning," *Machine Learning,* vol. 27, pp. 139-171, 1997.
[27] J.V. Jaskolski, "Construction of Neural Network Classification Expert Systems Using Switching Theory Algorithms," *Proc. Int'l Joint Conf. Neural Networks,* pp. I-1-6, 1992.
[28] S.J. Hong, "R-MINI: An Iterative Approach for Generating Minimal Rules from Examples," *IEEE Trans. Knowledge and Data Eng.,* vol. 9, pp. 709-717, 1997.
[29] H.W. Gschwind and E.J. McCluskey, *Design of Digital Computers.* New York: Springer-Verlag, 1975.
[30] T. Downs and M.F. Schultz, *Logic Design with Pascal.* New York: Van Nostrand Reinhold, 1988.
[31] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis.* Hingham, Mass.: Kluwer Academic Publishers, 1984.
[32] S.J. Hong, R.G. Cain, and D.L. Ostapko, "MINI: A Heuristic Approach for Logic Minimization," *IBM J. Research and Development,* vol. 18, pp. 443-458, 1974.
[33] D.L. Dietmeyer, *Logical Design of Digital Systems (third ed.).* Boston, Mass.: Allyn and Bacon, 1988.
[34] M. Muselli, "Predicting the Generalization Ability of Neural Networks Resembling the Nearest-Neighbor Algorithm," *Proc. Int'l Joint Conf. on Neural Networks (IJCNN 2000),* pp. I-27-33, 2000.
[35] M. Muselli and D. Liberati, "Training Digital Circuits with Hamming Clustering," *IEEE Trans. Circuit and Systems—I: Fundamental Theory and Applications,* vol. 47, pp. 513-527, 2000.
[36] C. Mead, *Analog VLSI and Neural Systems.* Reading, Mass.: Addison-Wesley, 1989.
[37] F.N. Sibai and S.D. Kulkarni, "A Time-Multiplexed Reconfigurable Neuroprocessor," *IEEE Micro,* vol. 17, pp. 58-65, 1997.
[38] S.B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, K. De Jong, S. Dzeroski, S.E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang, "A Performance Comparison of Different Learning Algorithms," Technical Report CMU-CS-91-197,: Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1991.
[39] *Machine Learning, Neural, and Statistical Classification.* D. Michie, D. Spiegelhalter, and C. Taylor, eds., London: Ellis-Horwood, 1994.
[40] C.J. Merz and P.M. Murphy, "UCI Repository of Machine Learning Databases" http://www.ics.uci.edu/~mlearn/MLRepository.html, Irvine, Dept. of Information and Computer Science, Univ. of California, Irvine, 1996.
[41] O.L. Mangasarian and W.H. Wolberg, "Cancer Diagnosis via Linear Programming," *SIAM News,* vol. 23, pp. 1-18, 1990.
[42] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys,* vol. 31, pp. 264-323, 1999.
[43] R.M. Gray, "Vector Quantization," *IEEE ASSP Magazine,* vol. 1, pp. 4-29, 1984.
[44] T. Kohonen, *Self-Organization and Associative Memory,* third ed. Berlin: Springer-Verlag, 1989.
[45] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," *Advances in Neural Information Processing Systems 7,* G. Tesauro, D. S. Touretzky, and T. K. Leen, eds., Cambridge, MA: MIT Press, pp. 625-632, 1995.
[46] G. Carpenter, S. Grossberg, and D.B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks,* vol. 4, pp. 759-771, 1991.
[47] N.B. Karayiannis and J.C. Bezdek, "An Integrated Approach to Fuzzy Learning Vector Quantization and Fuzzy C-means Clustering," *IEEE Trans. Fuzzy Systems,* vol. 5, pp. 622-628, 1997.
[48] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distribute Processing,* D.E. Rumelhart and J.L. McClelland, eds., pp. 318-362, Cambridge, Mass.: MIT Press, 1986.
[49] S.J. Hong, "Use of Contextual Information for Feature Ranking and Discretization," *IEEE Trans. Knowledge and Data Eng.,* vol. 9, pp. 718-730, 1997.
[50] W. Iba, J. Wogulis, and P. Langley, "Trading Off Simplicity and Coverage in Incremental Concept Learning," *Proc. Fifth Int'l Conf. Machine Learning,* pp. 73-79, Ann Arbor, Mich.: Morgan Kaufmann, 1988.
[51] J. Zhang, "Selecting Typical Instances in Instance-Based Learning," *Proc. Ninth Int'l Machine Learning Conf.,* pp. 470-479, Aberdeen, Scotland: Morgan Kaufmann, 1992.
[52] R. Setiono, "Generating Concise and Accurate Classification Rules for Breast Cancer Diagnosis," *Artificial Intelligence in Medicine,* vol. 18, pp. 205-219, 2000.
[53] G.P. Drago and S. Ridella, "Pruning with Interval Arithmetic Perceptron," *Neurocomputing,* vol. 18, pp. 229-246, 1998.
[54] J.M. Steppe and K.W. Bauer, "Improved Feature Screening in Feedforward Neural Networks," *Neurocomputing,* vol. 13, pp. 47-58, 1996.

Marco Muselli received the MSc degree in electronic engineering from the University of Genoa, Italy, in 1985. In 1988, he joined the Institute for Electronic Circuits of the Italian National Research Council in Genoa, where he is currently a principal scientist. His research interests include neural networks, machine learning, global optimization, mathematical statistics, and probability theory. His activity is mainly centered on the development of new efficient training methods for connectionist systems and on the theoretical analysis of their convergence properties. He is a member of the IEEE.

Diego Liberati received the PhD degree in electronic engineering, summa cum laude, from the Milano Institute of Technology, in February 1983. Since 1984, he is with the Italian National Research Council, where he is currently a chief scientist. In 1988, the Italian Ministry for Scientific Research awarded him the first research doctorate in Biomedical Engineering, Honoris Causa. He has been a secretary of the Biomedical Engineering Society within the Italian Electrical and Electronic Engineering Association (and Milano prize laureate in 1988), chairman of the Scientific Committee for the Conferences on Information and Control Technologies in Health Systems sponsored by the Italian Control Association, past president of the Milano region within the Saint Vincent de Paul Charity and of the Milano chapter within the International Movement of Catholic Intellectuals, visiting scientist at Rockefeller University, New York University, University of California, and the International Computer Science Institute. He has directed joint projects granted by transnational private (e.g., Hewlett Packard on artificial neural networks in industrial process monitoring) and public (e.g., European Union on nonlinear analysis of brain plasticity) institutions. Having taught signal processing, mathematical modeling, microelectronics, lab instrumentation, systems, and control theory in european universities. He has also mentored dozens of pupils toward and beyond their doctorate. His main scientific interest in information and communication technologies for life are related to measurement analysis, model identification, and synthesis of emulations in natural and artificial complex systems, whose scientific and ethical aspects involving the human beings he loves to discuss in his papers and books.

▷ **For more information on this or any computing topic, please visit our Digital Library at** http://computer.org/publications/dilb.