Building Neural and Logical Networks with Hamming Clustering

Marco Muselli
Istituto per i Circuiti Elettronici - CNR
via De Marini, 6 - 16149 Genova, Italy

Abstract

The solution of binary classification problems is obtained by employing a new learning method, called Hamming Clustering (HC). It is able to build in a constructive way a two-layer perceptron with binary weights, which can be easily implemented by means of conventional logical ports.

This technique generalizes the information contained in the given training set by combining input patterns that are close each other according to the Hamming distance. The output class is assigned in a competitive way, thus allowing the treatment of ambiguous samples.

The application of HC to the signal prediction in genomic sequences shows its ability to determine regularities in complex problems.

1 Introduction

Many real-world classification problems belonging to different application fields can be described through Boolean variables. In these cases the employment of general methods, like the back-propagation algorithm [1], which are expressly developed for the treatment of real inputs, increases the complexity of the problem and leads to higher execution times that limit the applicability range of the considered approach.

It is thus important to develop suitable training algorithms for binary classification problems, which exploit their peculiar characteristics to make easier the learning process and to improve the properties of the resulting connectionist model [2, 3]. Particular attention must also be devoted to the implementation on physical supports; the need of dealing with only two-state variables can indeed reduce the complexity of the devices to be realized.

A new learning technique of this kind is the procedure of Hamming Clustering (HC): it generates two-layer perceptrons which can be directly implemented through a combinatorial logical network, containing only AND, OR and NOT ports. The learning task 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 element. Since the method can subsequently work on samples belonging to different classes, the construction of the logical network is based on a particular kind of competitive learning.

During the training HC is able to determine redundant inputs, which do not affect the behavior of the resulting configuration. A final pruning

phase allows a further reduction of the network complexity and can yield a significant improvement in the corresponding generalization ability.

Some theoretical results [4] 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, whereas c is the number of hidden units in the resulting neural network. Also the amount of memory required is small and shows an asymptotic behavior $O(n(c+s))$.

The application of HC to the prediction of relevant signals in genomic sequences shows the good performances of the networks generated. Furthermore, it is straightforward to extract from these two-layer perceptrons a collection of templates that code the behavior of the underlying classifier. This property suggests a possible application of HC as a method for extracting understandable rules from a given set of samples. The analysis of this aspect is the subject of a complementary publication [5].

2 The Hamming Clustering procedure

Consider a training set S containing s samples $(\mathbf{x}_j, y_j)$, $j = 1, \ldots, s$, generated by a Boolean function $f : \{-1, +1\}^n \longrightarrow \{-1, +1\}$, eventually in the presence of noise. The n components x_{ji}, $i = 1, \ldots, n$, of the input patterns $\mathbf{x}_j$ and the corresponding output y_j are binary-valued, coded by the integers -1 and $+1$. Such a function f subdivides the vertices of the hypercube $\{-1, +1\}^n$ into two subsets H^+ and H^-, according to the corresponding output:

$$H^+ = \{\mathbf{x} \in \{-1, +1\}^n \mid f(\mathbf{x}) = +1\}, \quad H^- = \{-1, +1\}^n \setminus H^+ \tag{1}$$

In the same way, S can be subdivided into two subsets S^+ and S^-:

$$S^+ = \{\mathbf{x}_j \in \{-1, +1\}^n \mid (\mathbf{x}_j, y_j) \in S,\ y_j = +1\}, \quad S^- = S \setminus S^+$$

Note that $H^+ \cap H^- = \emptyset$, but $S^+ \cap S^- = \emptyset$ only if the training set S is *consistent*, i.e. it does not contain *ambiguous* input patterns $\mathbf{x}$, for which both $(\mathbf{x}, +1)$ and $(\mathbf{x}, -1)$ belong to S.

The procedures of *Hamming Clustering* (HC) determine the unknown function f or, equivalently, the sets H^+ and H^-, by combining the elements of $\{-1, +1\}^n$ in a competitive way. The technique adopted by HC can be better described by denoting with I_n the set $\{1, 2, \ldots, n\}$ of the first n positive integers and by introducing the following definition.

Definition 1 *The set* $C_l(\mathbf{x}, L) = \{\mathbf{z} \in \{-1, +1\}^n \mid z_i = x_i \text{ for every } i \notin L\}$, *where* $\mathbf{x} \in \{-1, +1\}^n$ *and* $L \subset I_n$ *has size* $l = |L|$, *will be called* l-cube *(or simply* cube*) with* vertex $\mathbf{x}$ *and* generator L.

The well known definition of Hamming distance between two binary vectors can then be easily extended to the case of two cubes:

Definition 2 *The Hamming distance* d_H *between an* l*-cube* $C_l(\mathbf{x}, L)$ *and a* k*-cube* $C_k(\mathbf{z}, K)$, *with* $0 \leq l, k \leq n$, *is given by* $d_H = \sum_{i \notin L \cup K} |x_i - z_i|/2$.

HAMMING CLUSTERING

1. Choose at random a sample $(\mathbf{x}, y)$ in the training set S.

2. Build one or more cubes having a vertex in $\mathbf{x}$. Remove $(\mathbf{x}, y)$ from S. If the construction is not complete, go to Step 1.

3. Prune the set(s) of cubes generated.

Figure 1: General procedure followed by Hamming Clustering.

The goal of HC is to obtain from the training set S two collection of cubes C^+ and C^-, which approximate the sets H^+ and H^-. A two-layer perceptron that provides output $+1$ for all the input patterns belonging to C^+ (C^-) can be directly realized. It is sufficient to build a layer of hidden threshold units performing AND operations that provide output $+1$ only when the input patterns contained in each cube $C_l(\mathbf{x}, L) \in C^+$ (C^-) are presented. The two-layer perceptron is completed by a final neuron performing a logical OR among the output of the hidden units. Such a configuration will be called *single-class HC network*.

A different architecture can be employed to realize a classification similar to that performed by the nearest-neighbor algorithm. This digital network will be called *majority HC network* and is formed by two consequent blocks. The first of them performs the computation of the Hamming distances between the input pattern and each of the elements belonging to C^+ and C^-, while the second one chooses the class containing the cube having minimum Hamming distance. Both the single-class HC and the majority HC network contain only binary weights and can be directly implemented by using only logical AND, OR and NOT ports [4].

The algorithm followed by HC is reported in Fig. 1; Steps 1 and 2 are repeatedly executed and form the basic iteration of the method.

2.1 Cluster generation

The computational kernel of HC is given by the generation of cubes having a vertex in the sample $(\mathbf{x}, y)$ previously selected (at Step 1 of Fig. 1). Suppose without loss of generality that $\mathbf{x} \in S^+$: the separation of the sets C^+ and C^- is ensured if each of the l-cubes $C_l(\mathbf{x}, L)$ generated in the current iteration satisfies the following two conditions:

$$C_l(\mathbf{x}, L) \cap S^- \subset S^+ \,, \qquad C_l(\mathbf{x}, L) \cap C^- = \emptyset \tag{2}$$

The procedure employed by HC for the generation of cubes having a vertex in $\mathbf{x} \in S^+$ is reported in Fig. 2. Minor changes are necessary if the training set S is not consistent. The wide freedom in the choice of the generator L

METHOD FOR THE GENERATION OF CUBES IN HC

1. Compute the s^- Hamming distances d_j between $\mathbf{x}$ and the patterns $\mathbf{x}_j \in S^-$ by applying Def. 2. Let D_j be the subset of I_n containing the d_j indices of the components that differ in $\mathbf{x}$ and in $\mathbf{x}_j$:

$$D_j = \{i \in I_n \mid x_i \neq x_{ji},\ \mathbf{x}_j \in S^-\} \quad , \quad \text{for } j = 1, \ldots, s^-.$$

2. Extract a subset $L \subset I_n$ such that $L \cap D_j \neq D_j$ for every $j = 1, \ldots, s^-$. The l-cube $C_l(\mathbf{x}, L)$, where $l = |L|$, satisfies conditions (2).

3. For every cluster $C_k(\mathbf{u}, K) \in C^-$, if $C_l(\mathbf{x}, L) \cap C_k(\mathbf{u}, K) \neq \emptyset$, remove from L an index $i \notin K$ such that $x_i \neq u_i$.

Figure 2: Procedure followed by HC for the generation of an l-cube.

allows us to define additional criteria that can influence the generalization ability of the resulting logical network. Two of them are:

Largest cube criterion: it requires that the size of the generator L chosen at Step 2 is maximum. To this end the index $i \in L$ that is included in the greatest number of sets D_j associated with the input patterns $\mathbf{x}_j \in S^-$ is repeatedly removed from L until conditions (2) are satisfied.

Maximum covering cube criterion: it maximizes the number of samples of S included in the current cluster and is mostly employed in the construction of majority HC networks. This criterion needs the construction of the sets $D_j^+ \subset I_n$ containing the indices of the components that differ in $\mathbf{x}$ and in the elements $\mathbf{x}_j \in S^+$. Then L is built by choosing the indices which are contained in the greatest number of sets D_j^+.

2.2 Pruning

In general the collection of cubes C^+ and C^- generated by Steps 1-2 of HC contain redundant elements which can be removed without changing the behavior of the resulting network. Thus, a pruning phase can be used to optimize the complexity of the final configuration. To describe the main pruning methods employed in HC we introduce the following definition:

Definition 3 *The set* $Q = \{\mathbf{x} \in \{-1, +1\}^n \mid (\mathbf{x}, y) \in S,\ \mathbf{x} \in C_l(\mathbf{u}, L)\}$ *will be called* cover set *of the* l*-cube* $C_l(\mathbf{u}, L)$*. The number* q *of input patterns contained in* Q *will be named* covering *of* $C_l(\mathbf{u}, L)$*.*

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. This can be realized by employing a near-optimal greedy procedure that extracts the clusters with maximum covering one at a

time. In this way the resulting collections C^+ and C^- cannot contain redundant clusters.

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. Normally, the value of τ is proportional to the maximum covering $q_{\max}$ found in the set of clusters to be pruned (e. g. $\tau = 0.1 \cdot q_{\max}$).

3 Application of HC to the signal prediction in genomic sequences

The properties of the two-layer perceptrons generated by HC have been tested on the problem of locating regulatory signals in genomic sequences [6]. In particular, HC has been applied to the identification of poly(A) signals on an inhomogeneous set of human DNA sequences. It is known that a fixed six base pattern such as AATAAA does not determine univocally the presence of a poly(A) signal; consequently, it is interesting to establish if a greater number of adjacent bases is sufficient to predict its precise position.

To this end we have considered a window containing a fixed number b of locations that slides on the genomic sequence and captures corresponding bases. Since HC works on binary input patterns, a four bit coding for the DNA bases has been adopted. In this way it is possible to associate every cube generated by HC with a template which can be used for the recognition of poly(A) signals.

For each value of $b = 6, 7, \ldots, 12$ about 1000 sequences containing poly(A) signals have been extracted by the EMBL database [7]; half of them (S^+) have been employed by HC to perform the training of binary neural networks together with a high number of base sequences corresponding to gene regions that do not contain the target signal (S^-). The resulting training sets S contain more than 100,000 samples, thus preventing the application of time-consuming classification methods. On the contrary, the low computational cost of HC allows us to obtain within a reasonable time a complete set of templates to be used for the analysis of new patterns.

Tab. 1 presents the results of the classification performed on ten genomic sequences randomly extracted from the test set. For each of them it is reported the accession number in the EMBL database, the total number of bases, the number of poly(A) signals found experimentally and the number of possible targets located by HC. An asterisk near this last value denotes that one target has been missed by the resulting network.

As one can note, the number of presumed poly(A) signals decreases with the window length b; this shows that the flanking region plays an important role as supposed. In several test sequences the redundancy of the prediction vanishes when $b = 9$ or $b = 10$; consequently, these are the optimal window lengths to be used in the recognition of these signals.

Genomic sequence			Window length						
Accession n.	N. bases	N. poly(A)	6	7	8	9	10	11	12
d12676	2329	1	14	6	4	3	2	3	6
j00184	2685	1	1	1	1	1	1	1	1
j05582	4139	1	4	2	1	1	1	1	1
l11706	3250	1	2	2	1	1	1	1	2
m26679	3079	1	8	6	3	2	1	1	3
m33518	6349	1	6	3	2	2	2	2	2
m59040	1794	1	9	6	2	1	2	*2	3
m63154	1621	2	6	5	2	2	3	3	5
m69245	1503	1	8	6	1	2	1	*1	4
m91083	1539	1	1	1	1	1	1	1	1

Table 1: Number of presumed poly(A) signals obtained by HC.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning internal representations by error propagation. In *Parallel Distribute Processing*, D. E. Rumelhart and J. L. McClelland, eds., Cambridge, MA: MIT Press, 1986, 318–362.

[2] D. L. Gray and A. N. Michel, A training algorithm for binary feedforward neural networks. *IEEE Transactions on Neural Networks*, **3** (1992), 176–194.

[3] H. M. A. Andree, G. T. Barkema, W. Lourens, A. Taal, and J. C. Vermeulen, A comparison study of binary feedforward neural networks and digital circuits. *Neural Networks*, **6** (1993), 785–790.

[4] M. Muselli and D. Liberati, Training digital circuits with Hamming Clustering. Submitted to *IEEE Transactions on Circuit and Systems*, (1998).

[5] M. Muselli and D. Liberati, Inferring understandable rules through digital synthesis. Accepted as a contribution to *WIRN'99 - The 11-th Italian Workshop on Neural Nets*, (1999).

[6] E. E. Snyder and G. D. Stormo, Identification of coding regions in genomic DNA sequences: An application of dynamic programming and neural networks. *Nucleic Acid Research*, **21** (1993), 607–613.

[7] C. M. Rice, R. Fuchs, D. G. Higgins, P. J. Stoehr and G. N. Cameron, The EMBL data library. *Nucleic Acid Research*, **21** (1993), 2967–2971.