Hamming Clustering: Improving Generalization in Binary Neural Networks

M. Muselli
Istituto per i Circuiti Elettronici, Consiglio Nazionale delle Ricerche
via De Marini, 6 - 16149 Genova, Italy

1 Introduction

An important reason that gives rise to current increasing interest in neural networks is undoubtedly their ability to generalize the information contained in a given set of patterns. In fact, in a large number of real applications a proper neural network is able to provide correct outputs for patterns not belonging to its training set.

The comprehension of this phenomenon has been the object of many papers because of its practical consequences (Denker et al., 1987; Baum & Haussler, 1989). With regard to classification problems it has been recently observed that simple statistical methods, like nearest neighbor algorithm, show a high generalization ability in many real situations (Bottou & Vapnik, 1992). Consequently, corresponding functional dependences between inputs and outputs have some locality properties that can be used to accelerate and/or improve the learning process.

If we limit ourselves to binary neural networks, then a locality property can be formulated in the following way: *it is more likely that patterns at a low Hamming distance belong to the same class (*i.e. *have the same outputs).* In the following section a new method, called *Hamming Clustering* (HC), is presented, that allows the construction of a given neural network in such a way to satisfy this locality property.

This technique can be directly inserted in a particular constructive method, such as the upstart algorithm (Frean, 1990) or the procedure of sequential learning (Marchand et al., 1990). Moreover, the Hamming clustering considerably reduces the number of connections in the input layer by neglecting redundant weights.

2 The Hamming Clustering

Consider the basic operation of a constructive method, that is the addition of a neuron to the hidden layer. This neuron has binary inputs, coded by the values -1 and $+1$, and provides, through a proper activation function, a binary output. Let P^+ and P^- be the sets of input patterns for which the hidden unit must give output $+1$ and -1 respectively.

HC groups the patterns of P^+ (or equivalently P^-) which are closer according to the Hamming distance, and leads to the formation of clusters in the input space. In this way an output is naturally assigned to the patterns that are not contained in the training set.

For sake of simplicity let us denote with the symbols '+' and '−' the two binary states (corresponding to the integer values -1 and $+1$); with this notation the input pattern $\boldsymbol{x} = (+1, -1, -1, +1, -1)^t$ is equivalent to the string $+--+-$ (t indicates the transpose operation). Then we use the term *template* for denoting a string of binary components which can also contains *don't care* symbols, coded by the character '0'. By expanding the don't care symbols of a template, we obtain a set of patterns called the *equivalent set* of that template.

Two important properties can be easily verified:

1. A template has a direct correspondence with a logic AND among the pattern components. Such an operation gives output $+1$ only for the patterns contained in the equivalent set.

2. The construction of a threshold (or window) neuron that performs the AND associated to a given template is straightforward.

HC proceeds by extending and subdividing clusters of templates; every cluster contains templates having don't care symbols in the same locations. The method initially considers only one cluster containing all the patterns of P^+ and iterates the following two actions:

Extension: Each binary component in the cluster is replaced one at a time by the don't care symbol and the corresponding number of conflicts with the patterns of P^- is computed.

Subdivision: The binary component with the minimum number of conflicts is considered. If this number is greater than zero, then the cluster is subdivided in two subsets. The former contains the templates which do not lead to conflicts (with the selected binary component replaced by the don't care symbol) while the latter is formed by the remaining templates (unchanged).

These two actions are then applied to the resulting clusters and the procedure is iterated until no more extension could be done. It is straightforward to construct a neural network that provides output $+1$ for the patterns contained in the equivalent sets of the final templates and output -1 for all the remaining patterns.

However, more compact neural networks could exists; in order to find a configuration of this kind we insert HC into a training algorithm for single neuron. The following definition introduces a possible approach: if a pattern of P^+ belongs to the equivalent set of a given template, then we say that this template *covers* that pattern. Furthermore, we use the term *covering* for denoting the number of patterns of P^+ covered by the templates of a given cluster.

With these definitions a practical method for the integration between HC and a training algorithm for single neuron follows these steps:

1. Starting from the training set (P^+, P^-) perform the Hamming clustering and reach the final clusters of templates.

2. Choose the cluster having maximum covering and consider the binary components of this cluster (leave out don't care symbols).

3. Construct a new training set from P^+ and P^-, in which only the selected components are kept.

4. Generate the corresponding neuron and remove the connections corresponding to the neglected components (by zeroing their weights).

3 Tests and Results

In order to analyze in detail the performances of HC we have considered a high number of different training sets characterized by the nearest-neighbor correlation function (Andree et al., 1993):

$$\Gamma = \Gamma(P^+, P^-) = \frac{N_{\rm eq} - N_{\rm diff}}{n2^{n-1}}$$

where $N_{\rm eq}$ ($N_{\rm diff}$) is the number of pattern pairs at Hamming distance 1 having equal (different) output. The term $n2^{n-1}$ (n is the number of inputs) is a normalization factor.

A correct definition of the correlation function Γ requires that the training set contains all the possible patterns of length n. Consequently we have always $-1 \leq \Gamma \leq +1$.

With a proper algorithm we have constructed, in the case $n = 8$, fifty training sets for every value of Γ in the range $[-1, 0.75]$ with step 0.05. The main object of this test is the generalization ability; so we have imposed that the number of input patterns with corresponding output $+1$ is equal to the number of patterns with output -1. In such a way a random classifier has exactly probability 0.5 of giving a right answer. It can be verified that a training set of this kind cannot have $\Gamma > 0.75$.

Half of patterns in each training set has been then eliminated and used for the construction of the corresponding test set. Fig. 1 shows the average percentage of patterns in the test sets correctly classified by the following algorithms for each value of Γ:

1. Hamming clustering (HC)

2. Modified procedure of sequential learning with training algorithm for window neuron (WTA) (Muselli, 1992)

3. Same as 2 plus Hamming clustering (WTA + HC)

The constructive method used in 2 and in 3 has been mainly chosen for its high velocity in the construction of neural networks. The average number of connections in the resulting networks are also reported in fig. 1.

From the simulation results we can deduce the following two facts:

1. HC presents a better generalization ability only for $\Gamma > 0$; in fact, these values of the correlation function characterize training sets with the required locality property. Moreover, a simple operation can change the sign of Γ in a given training set (Andree et al., 1993), increasing the application range of HC.

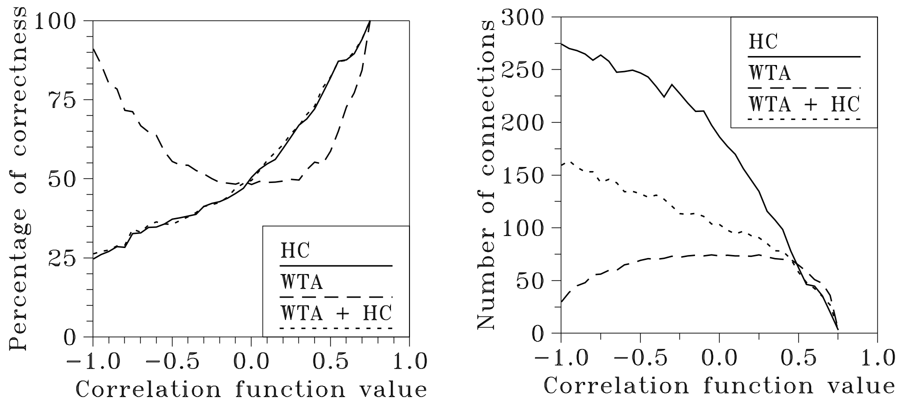

Figure 1: Average percentage of correctness and average number of connections for neural networks corresponding to several values of the correlation function Γ.

2. The union of HC and WTA leads to more compact neural networks, maintaining at the same time a good generalization ability. This can be very important from an implementative point of view.

References

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

Baum, E. B., & Haussler, D. (1989) What size net gives valid generalization?. *Neural Computation,* **1,** 151–160.

Bottou, L., & Vapnik, V. (1992) Local learning algorithms. *Neural Computation,* **4,** 888–900.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., & Hopfield, J. (1987) Large automatic learning, rule extraction, and generalization. *Complex Systems,* **1,** 877–922.

Frean, M. (1990) The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation,* **2,** 198–209.

Marchand, M., Golea, M., & Rujan, P. (1990) A convergence theorem for sequential learning in two-layer perceptrons. *Europhysics Letters,* **11,** 487–492.

Muselli, M. (1992) Training window neurons through the solution of linear equations. In *Artificial Neural Networks, 2,* **vol. 1,** 157–160. Eds.: Aleksander, I., and Taylor, J. Amsterdam: North Holland.