Hierarchical Learning Using Neural Networks

Lluís Garrido
Dept. ECM
Univ. de Barcelona
garrido@ecm.ub.es

Sergio Gómez
DEIM
Univ. Rovira i Virgili
sgomez@etse.urv.es

Jaume Roca
Dept. ECM
Univ. de Barcelona
roca@ecm.ub.es

Abstract

We address the problem of training and prediction with Neural Networks in those situations where the data to predict possess a natural hierarchical structure. We perform backpropagation with a specialized objective function that takes into account this structure and show that it betters the results of the non-hierarchical learning.

Keywords: Artificial Neural Networks.

1 Introduction

It is widely known that Neural Networks are well suited for learning and prediction purposes [1][2]. The most standard setting uses a so-called feed-forward network [3] where the units are organized in layers, each of them connected only to the units that form the outer layers.

In the training phase the net is presented with pairs of input and desired output values corresponding to a set of patterns

$$(\boldsymbol{x}_a, \boldsymbol{z}_a), \qquad a = 1, \ldots, p. \tag{1}$$

Here $\boldsymbol{x}_a \in \mathbb{R}^n$ is used to feed the n cells of the input layer. For each of the p patterns the network produces an answer in the output layer $\boldsymbol{o}(\boldsymbol{x}_a) \in \mathbb{R}^m$ that we would like to be as close as possible to the correct result $\boldsymbol{z}_a$. The *learning problem* consists thus in adjusting the synaptic weights ω_{ij} of the network connections in such a way that it minimizes the overall error for the complete set of patterns.

The most commonly used objective function for this purpuse is the quadratic error function

$$E_Q = \sum_{a=1}^{p} (\boldsymbol{o}(\boldsymbol{x}_a) - \boldsymbol{z}_a)^2. \tag{2}$$

This turns the learning problem into a standard optimization problem. The backpropagation algorithm [1] can now be invoked which basically performs an iterative process where the ω_{ij} are progressively adjusted following a gradient descent of (2) in the space of weights. Once the training is finished the performance of the network can be tested by presenting it with new patterns not seen so far and checking how the results predicted by the net resemble the correct ones.

One of the properties of the cost function (2) is that it weights alike the errors of each of the units in the output layer. This is a natural behavior when all output cells can be attributed the same "importance". There are situations, however, where the correctnes of the result relies heavily only on the values of a subset of the output cells, while the other cells are used to qualify or modify slightly the big picture drawn by the first set. For this sort of problems it is reasonable to expect that a more specialized cost function —one that takes into account the hierarchical role of each of the output units— will be able to produce more accurate results than a "general-purpose" one such as (2). In this letter we consider one such cost function, describe its main properties, and test its performance in the problem of learning binary addition.

Note that the kind of hierarchy we are addressing is not the one which is commonly found in the literature (see *e.g.* [4, 5]): we are not grouping the patterns in families, families within families, and so on. Instead, we consider a hierarchical structure of the output units for all patterns in the training set.

2 A hierarchical cost function

Let us consider the objective function defined by

$$E_N = \sum_{i=1}^{N} a_i, \tag{3}$$

where the coeficients a_i are defined in terms of the error functions e_i acording to the recursive rule

$$\begin{aligned} a_1 &= e_1, \\ a_i &= e_i + (1 - e_i)\, a_{i-1}, \qquad (i > 1). \end{aligned} \tag{4}$$

The errors e_i need not correspond to individual cells but rather to each one of the subsets of cells that share the same hierarchical value. Thus, we are considering here N different hierarchies, e_1 being the error associated with the most important group of cells. Furthermore, the precise form how the e_i are computed in terms of the errors for its individual cells will not be relevant for the ongoing discussion, although a natural choice may be a standard quadratic cost function for each of them.

We will also assume that the e_i have a finite range of variation (this is the standard situation when using non-linear neuron activation functions such as sigmoids), which we set for convenience between 0 and 1.

The objective function E_N was first introduced by Moscato and Klasnogor [6] within a Simulated Annealing approach using a learning by example scheme. Our aim here is to show that E_N has also all the right properties to be used by a Neural Network trained with the backpropagation algorithm.

In fact, even before solving the recursive relations (4) we can already get the flavor of the hierarchical structure of (3) with the help of the following property

$$\begin{aligned} &E_N(e_1, \ldots, e_{i-1}, 1, e_{i+1}, \ldots, e_N) \\ &\quad = E_{i-1}(e_1, \ldots, e_{i-1}) + N - i + 1. \end{aligned} \tag{5}$$

This implies that, in a situation where the i-th hierarchy group is assigned a very large error ($e_i = 1$) the cost function becomes independent of the values $e_{i+1}, \ldots, e_N$, meaning that E_N will become insensitive of the correctnes of these higher groups. The total error will only lower if the error e_i is lowered in the first place.

Solving the recursive relations (4) we get

$$a_i = e_i + \sum_{j=1}^{i-1} e_j \prod_{k=j+1}^{i} (1 - e_k), \tag{6}$$

so the cost function (3) can be rewritten as

$$E_N = \sum_{i=1}^{N} e_i + \sum_{i=1}^{N-1} e_i \sum_{j=1}^{N} \prod_{k=i+1}^{j} (1 - e_k). \tag{7}$$

Computation of the derivatives $\partial E_N / \partial e_i$ is a necessary step in order to apply the backpropagation algorithm with this objective function. In addition, it will also provide more information on the analytic structure of E_N. We get the following expressions

$$\begin{aligned} \frac{\partial E_N}{\partial e_1} &= 1 + \sum_{j=2}^{N} \prod_{k=2}^{j} (1 - e_k), \\ \frac{\partial E_N}{\partial e_i} &= \left(1 + \sum_{j=i+1}^{N} \prod_{k=i+1}^{j} (1 - e_k) \right) \\ &\qquad \times \prod_{l=1}^{i-1} (1 - e_l), \qquad (1 < i < N) \\ \frac{\partial E_N}{\partial e_N} &= \prod_{l=1}^{N-1} (1 - e_l). \end{aligned} \tag{8}$$

Here we see again the hierarchical behavior of the cost function: the dependence of E_N on e_i tends to vanish if any of the previous errors $e_1, \ldots, e_{i-1}$ is still very large, *i.e.*

$$\frac{\partial E_N}{\partial e_i} \longrightarrow 0 \qquad \text{for} \quad e_j \to 1, \quad \text{with} \quad j < i \tag{9}$$

Another important property of E_N is the fact that all its derivatives are positive definite in the range of variation of the $\boldsymbol{e} = (e_k)$,

$$\begin{aligned} \frac{\partial E_N}{\partial e_1} &> 0, \qquad \forall\, \boldsymbol{e} \in [0,1]^N \\ \frac{\partial E_N}{\partial e_i} &> 0, \qquad \forall\, \boldsymbol{e} \in [0,1)^N \end{aligned} \tag{10}$$

which implies that E_N possesses only one (absolute) minimum, located at $\boldsymbol{e} = \boldsymbol{0}$, as it should be.

3 Learning binary addition

Here we will use the above objective function for training a Neural Network to compute the addition of pairs of numbers in a binary representation. This example, which has also been addressed via a Simulated Annealing approach [7], fits well in the kind of problems where the output of the network can be categorized: between two output cells it is clearly more important the one representing a higher order bit, since failing to predict it correctly will produce a larger error in the predicted sum. So, in this example, it seems natural to consider as many hierarchical classes as there are cells in the output, the top hierarchy being assigned to the highest order bit.

We have trained a net to perform binary addition of pairs of 4-bit numbers. This amounts to $2^4 \cdot 2^4 = 256$ possible sums. We used for the training a subset of 64 randomly chosen sums. The net performance was tested by asking it to predict the remaining 192 sums.

The network arquitecture contained an input layer of 8 units, two hidden layers of 17 and 10 units respectively, and an output layer of 5 units. For every selection of training and testing subsets we performed backpropagation one time with the above hierarchical function E_5 (5 hierarchical levels) and a second time with the (non-hierarchical) quadratic cost function E_Q.

The results confirmed the intuitive idea that the hierarchical function should do better. We have illustrated it with the help of the so-called *confusion matrix*. This is just a plot where each sum tested by the net is represented by a point whose coordinates are given by the correct and the predicted values for the sum. Good predictions would thus generate highly diagonal confusion matrices. Figures 1 and 2 are plots of the confusion matrix for a typical run corresponding E_5 and E_Q, respectively. The points are actually represented by circles of variable sizes. The size of a circle is proportional to the multiplicity of the point it represents (*i.e*, it is proportional to the number of patterns that share the same correct and predicted sum). It can be noticed that the hierarchical confusion matrix (figure 1) is notably more regular and shows less dispersion around the diagonal than the non-hierarchical one (figure 2).

A complementary view is provided by the histograms in figures 3 and 4 which display the number of patterns that have failed their prediction by a given amount. Again, the hierarchical case (figure 3) clearly beats the non-hiearchical one (figure 4) by showing a less dispersed shape. Notice also an awkward feature in figure 4 where errors tend to pile together in even number (*i.e.* it is much more common to fail the prediction by 2, 4,..., than by 1, 3,...). This is probably a consequence that it is quite simple for the net to learn the lowest order bit of the sum, as it only depends on the lowest order bits of the two input numbers. So, in the non-hierarchical case this bit is almost always correct although it is actually the less important one.

Acknowledgements

This work has been supported in part by a CICYT contract AEN99-0483.

References

[1] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back-propagating errors, *Nature* **323** (1986) 533.

[2] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representations by error propagation. In *Parallel Distributed Processing*, D. E. Rumelhart and J. L. McClelland (eds.), MIT Press, Vol. 1, Cambridge, MA (1986) 318.

[3] F. Rosenblatt, *Principles of neurodynamics*, Spartan, New York (1962).

[4] C. Cortes, A. Krogh and J. A. Hertz, Hierarchical associative networks, *J. Phys. A* **20** (1987) 4449.

[5] H. Gutfreund, Neural networks with hierarchically correlated patterns, *Phys. Rev. A* **37** (1988) 570.

[6] P. Moscato and N. Krasnogor, Learning, Hierarchies and Hints, (unpublished).

[7] P. Carnevali and S. Patarnello, Exhaustive thermodynamical analysis of boolean networks, *Europhys. Lett.* **4** (1987) 1199.

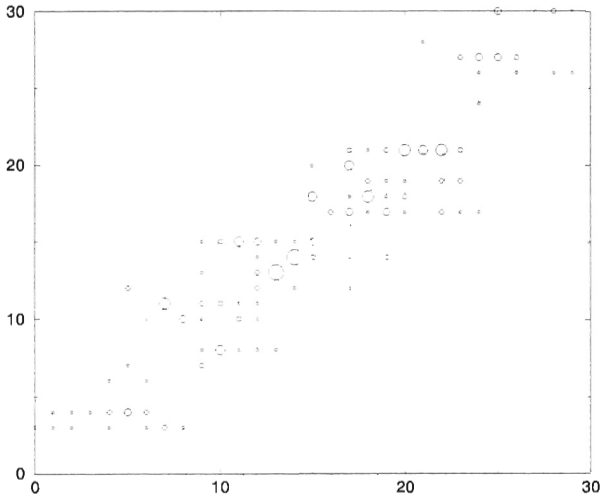

Figure 1: Confusion matrix for hierarchical (E_5) cost function

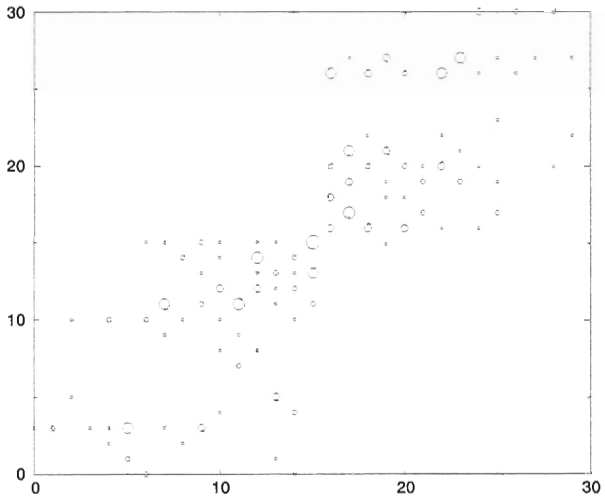

Figure 2: Confusion matrix for non-hierarchical (E_Q) cost function

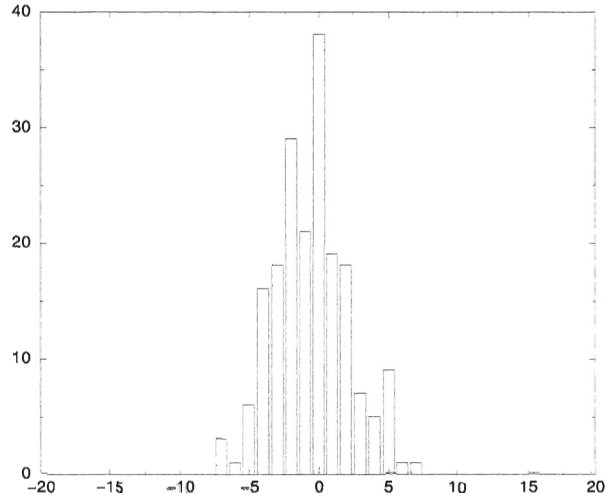

Figure 3: Histogram for hierarchical (E_5) cost function

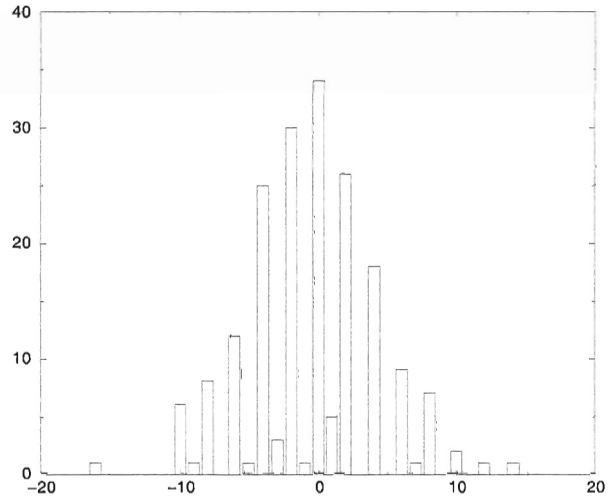

Figure 4: Histogram for non-hierarchical (E_Q) cost function