EUROPHYSICS LETTERS 14 February bis 1992

Europhys. Lett., 18 (2), pp. 95-100 (1992)

The Offset Algorithm: Building and Learning Method for Multilayer Neural Networks.

D. MARTINEZ and D. ESTÈVE

Laboratoire d'Automatique et d'Analyse des Systèmes
7 av. du Col. Roche, 31077 Toulouse, France

(received 22 April 1991; accepted in final form 8 January 1992)

PACS. 02.70 – Computational techniques.
PACS. 87.10 – General, theoretical, and mathematical biophysics (inc. logic of biosystems, quantum biology and relevant aspects of thermodynamics theory, cybernetics, and bionics).

Abstract. – A general method for building and training a multilayer neural network functioning as a parity machine is proposed. It is composed of two basic steps: a growth step in which two hidden layers are built and a pruning step in which any redundant units are removed. A perceptron-type algorithm is used to learn the connection strengths in order to minimize a classification error. The first hidden layer is built by adding units as they are needed, until the zero error convergence is achieved. We then show that the problem of mapping these internal representations onto the desired output is the n-parity problem. So, the second hidden layer is built by a geometrical design procedure with no learning. The used pruning process can remove sometimes all the units of the second hidden layer. The final architecture can then have one or two hidden layers.

1. Introduction.

The Perceptron algorithm can realize a dichotomy in the case of a linear separation [1]. An arbitrary dichotomy can be realized by a neural architecture with hidden layers. However, designing a multilayer architecture for a certain classification is an important practical problem. This can be trivally solved if one hidden unit is allocated for each pattern to learn (Grand Mother solution). In fact, this solution is not capable of generalizing with new inputs. Furthermore, it has been shown that the generalization properties of a network are most pronounced for a minimal architecture [2]. Conventional neural architectures are static because back-propagation learning is limited to a synaptic weight values modification [3]. So, in practice, the network architecture is selected by a trial and test procedure. The number of units and the number of layers are modified until the generalization performances obtained are satisfactory. This procedure tends to be time-consuming and it is used for small problems (few inputs and few units) but is unacceptable for complex problems. It is therefore better if the architecture building can be automated and incorporated into the learning procedure. The neural net can be grown by adding and removing units until the net works as desired. Recently, in the same spirit, several attemps have been proposed [4-7]. The Tiling algorithm [4, 5] lets you obtain several layers. Each hidden layer has two kinds of hidden unit: a

master unit obtained by minimizing an error and ancillary units obtained in order to have «faithful» classes. The Golea and Marchand approach [7] and Upstart algorithm [6] builds an unconventional architecture like a binary tree. Each hidden unit can make two kinds of error. Each kind of error is then treated separately by one unit. In our approach, the neural architecture is a conventional multilayer network and only one type of unit and one type of error are considered. Moreover, after the growth step that builds two hidden layers, a pruning step is used to remove redundant units in the second layer.

2. The Offset strategy.

Consider a layered network made of binary units which can take a state -1 or $+1$. The activation function of such a unit k is

$$S_k = \operatorname{sgn}\left(\sum_{j=1}^{n} W_{kj} S_j\right), \tag{1}$$

where W_{kj} is the synaptic weight value between unit k and a unit j of the preceding layer. S_j is the state of unit j. $j = 1 \ldots n$ if the preceding layer has n units. We want to map M patterns $\xi^\mu \in \{-1, +1\}^m$ onto a desired output (target) $t^\mu \in \{-1, +1\}$. We have $\mu = 1 \ldots M$ and $M \leqslant 2^m$. On presentation of pattern ξ^μ, the Perceptron algorithm alters the weights W_j^μ of the output unit if the output o^μ is different from the target t^μ. The Perceptron algorithm converges if classes are linearly separable but does not if the problem is not. However, a simple extension called Pocket algorithm [8] makes it possible to find a good solution in the nonlinearly separable case. This consists of running the Perceptron algorithm but keeping in memory the set of weights that remains the longest without being modified. It has been shown that the probability of having the set of weights which produces the smallest number of errors is close to one when the training time increases. We have chosen to minimize the error E defined as the number of misclassified patterns. For unit k, we can write

$$E(k) = \frac{1}{4} \cdot \sum_{\mu=1}^{M} (t_k^\mu - o_k^\mu)^2. \tag{2}$$

We denote by o_k^μ the output and by t_k^μ the target of unit k for an input pattern ξ^μ (o_k^μ and $t_k^\mu = \pm 1$). Let $f(t_k^\mu, o_k^\mu)$ be a certain measure of distance between t_k^μ and o_k^μ for the pattern ξ^μ and hidden unit k. Our strategy is to offset this error $f(t_k^\mu, o_k^\mu)$ in building a new unit $k+1$. Targets for unit $k+1$ are given by

$$t_{k+1}^\mu = f(t_k^\mu, o_k^\mu). \tag{3}$$

Like $t_{k+1}^\mu \in \{-1, +1\}$, an example of such a measure can be

$$t_{k+1}^\mu = f(t_k^\mu, o_k^\mu) = t_k^\mu \oplus o_k^\mu, \tag{4}$$

where $\oplus$ is the *exclusive disjunction function*. Hence, the input patterns that made errors for unit k are mapped onto the target $+1$ and the other patterns (properly classified by unit k) are mapped onto the target -1. So, unit $k+1$ must be active for the errors of unit k.

Note that this error measure f (eq. (4)) can be decomposed by means of Boolean algebra in two measures f_1 and f_2 such as $f(t_k^\mu, o_k^\mu) = f_1(t_k^\mu, o_k^\mu) \vee f_2(t_k^\mu, o_k^\mu)$, with $f_1(t_k^\mu, o_k^\mu) = \neg\, t_k^\mu \wedge o_k^\mu$ and $f_2(t_k^\mu, o_k^\mu) = t_k^\mu \wedge \neg\, o_k^\mu$, where $\wedge$, $\vee$, $\neg$ correspond to the *conjunction*, the *disjunction*, and the *complementation*. For example, the Upstart algorithm uses f_1 for «wrongly on» errors and f_2 for «wrongly off» errors (see [6]).

Based on a successive error correction in the hidden layer, the growth procedure is simple:

first, a hidden unit is built, connected to all the input units, and trained with the original mapping: $\xi^\mu \to t^\mu$. If the problem is not linearly separable, the Pocket algorithm will give a weight set which will produce a certain number of errors. In this case, a new hidden unit is built to offset the errors of the preceding unit. To make this, the training set is modified by eq. (4).

However, after training, unit $k+1$ can also produce errors on the modified mapping: $\xi^\mu \to t^\mu_{k+1}$. So, a new unit $(k+2)$ is built to offset the errors of unit $k+1$. This growth procedure terminates when the last building unit does not produce any errors.

3. Two criteria for convergence.

1) The Offset algorithm will converge after a finite number of units in the first hidden layer.

Proof. Let $E_{(k)}$ be the number of errors of unit k and $E_{(k+1)}$ the number of errors of unit $k+1$. Let ξ^μ be an error pattern for unit k, *i.e.* $o^\mu_k \neq t^\mu_k$. Like $\xi^\mu \in \{-1, +1\}^m$, it is always possible to isolate the pattern ξ^μ by an $(m-1)$-dimensional hyperplane. The weight values for this are

$$W_{j(k+1)} = \xi^\mu_j \quad \text{and} \quad W_{0(k+1)} = -\sum_j (\xi^\mu_j)^2 . \tag{5}$$

With these weights, the output of unit $k+1$ is $+1$ for the pattern ξ^μ only and -1 for the others. We therefore have: $E_{(k+1)} = E_{(k)} - 1$. Given that the Pocket algorithm is then used for unit $k+1$ to reduce the number of errors $E_{(k+1)}$, we will have $E_{(k+1)} \leqslant E_{(k)} - 1$.

2) Suppose that n units have been built in the first hidden layer by the growth process. Hence, $E_{(n)} = 0$. To the M input patterns ξ^μ $(\mu = 1 \ldots M)$, there corresponds M internal representations $o^\mu = (o^\mu_1, o^\mu_2 \ldots o^\mu_k \ldots o^\mu_n)$ in the first hidden layer. The problem is now to map these internal representations o^μ onto the targets $t^\mu = \pm 1$. The original Boolean function has been now mapped into a new function defined by the internal representations.

Theorem 1. If n is the number of units built by the Offset algorithm in the first hidden layer, then the problem of mapping the internal representations o^μ onto the target t^μ is the n-parity problem.

Proof. Recall that with the Offset strategy, the units are constructed in order to respond $+1$ for the preceding unit errors. Targets for the last building unit are obtained by $t^\mu_n = t^\mu_{n-1} \oplus o^\mu_{n-1}$. Targets for unit $n-1$ are obtained by the same way: $t^\mu_{n-1} = t^\mu_{n-2} \oplus o^\mu_{n-2}$.

Like targets for the first unit are given by the mapping problem $(t^\mu_1 = t^\mu)$, we can write $t^\mu_n = o^\mu_1 \oplus o^\mu_2 \oplus \ldots \oplus o^\mu_{n-1} \oplus t^\mu$.

If unit n does not produce any errors $(o^\mu_n = t^\mu_n)$ we therefore have

$$t^\mu = o^\mu_1 \oplus o^\mu_2 \oplus \ldots \oplus o^\mu_{n-1} \oplus o^\mu_n . \tag{6}$$

4. Geometrical building of the second hidden layer.

The Offset algorithm therefore reduces any Boolean function to a parity problem of dimension n. The problem is now to map the internal representations onto the target, in other words solve the n-parity problem.

Theorem 2. A perceptron with one hidden layer of n units can solve a parity dichotomy on an n-dimensional hypercube. The network architecture and the synaptic weights values can be determined geometrically.

Proof. Let $(x_1 \ldots x_n)$ be a vector, with $x_i = \pm 1$. $\sum x_i = 2k - n$ define a hyperplane. All vectors with exactly k bits «on» are lying in this hyperplane. We construct $n - 1$ hyperplanes for $k = 1 \ldots n - 1$. In order to have all the k bits «on» vectors lying in between two hyperplanes, the $n - 1$ hyperplanes are shifted in two opposite directions: $\sum x_i = 2k - n \pm 1$. We obtain n different hyperplanes. All patterns with k bits «on» are lying in between two hyperplanes. Let a pair of units be denoted by u_1 and u_2, associated to the pair of preceding hyperplanes. All patterns with k bits «on» will give rise to the output $+1$ for u_1 and -1 for u_2. Moreover, all other patterns with no k bits «on» will give rise to the same output for u_1 and u_2. So, the weight from u_1 to the output unit is set to $+1$ and from u_2 to the output unit is set to -1. The threshold of the output unit is set at 0.5.

5. Removing of redundant units.

Figure 1 shows the network with two hidden layers obtained at the end of construction. The first hidden layer maps an arbitrary dichotomy onto an n-parity problem. This network is known as a parity machine, *i.e.* a feedforward network whose output is the parity of the internal representations [9]. In our network, the second layer solves the n-parity problem. Note that the storage capacity of a parity machine has received a lot of attention in the recent literature, namely in [9, 10].

Let p be the number of distinct internal representations and n the number of units in the first hidden layer. Two outcomes can be distinguished:

1) if $p = 2^n$, we have the entire n-parity problem. To solve it, the second hidden layer is necessary, so no units are redundant.

2) If $p < 2^n$, we have just a part of the n-parity problem. The building network can solve the problem but it is quite possible that units in the second hidden layer may be redundant. In that case, a very simple pruning procedure is used: a hidden unit in the second hid-

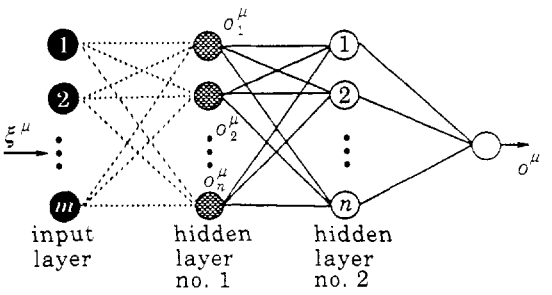

Fig. 1.

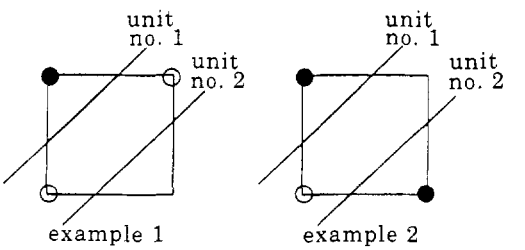

Fig. 2.

Fig. 1. – Network with two hidden layers built by the Offset algorithm before pruning step. —— Non-adaptive connections, – – – adaptive connections.

Fig. 2. – Two examples of three distinct internal representations. Both examples are linearly separable but two units have been built to solve the parity problem. Then two line segments correspond to the two units in the second hidden layer. In the first example, unit 2 will be removed. So the network will have only one hidden layer. In the second example, no unit will be removed. ● Odd internal representation, ○ even internal representation.

den layer can be removed if the internal representations remain «faithful» after removing. This means that two input patterns of different target must not be associated with the same internal representation. In fig. 2, the second example shows that in certain cases, the elimination procedure used is not optimum. However, we noted that in practice this pruning procedure reduced the network size considerably.

6. Simulations.

We used 1000 iterations of pocket algorithm for simulations, however the weight value variation is not exactly the training perceptron rule. A version of perceptron learning given by Frean makes it possible in practice to accelerate the search for an optimum weight set [6].

Random mapping. – In this problem, patterns of m bits and targets are generalized at random with 50% probability. This is a difficult problem due to lack of correlation between the inputs. For m up to 9, the number of hidden units is indicated in fig. 3. Each network's size is an average over 25 runs on a different training set. We can see that in practice the pruning procedure reduces the network's size considerably. The networks are smaller than those produced by the Tiling algorithm or the Golea and Marchand approach and approximately equal to those produced by the Upstart algorithm (see results shown in [4-7]).

Iris data. – The Iris data classification problem, used by Fisher in 1936, is still today a standard example for testing pattern recognition methods. The data set consists of three classes, four continuous features and 150 patterns, 50 for each classes [11]. Note that the criterion 1 for Offset algorithm convergence is not verified for multivalued inputs. However, a data normalization, such as $\xi^\mu \cdot \xi^\mu = 1$, allows one to ensure Offset convergence in the presence of nonbinary inputs [12]. The mean number of hidden units found after the pruning step was 3.41 ± 1.01 which is similar to the one determined by back-propagation network [13]. The generalization performances are estimated with the leaving-one-out method: 0.0 error on training set and 0.033 cross-validation error obtained with Offset are better than 0.017 error on training set and 0.033 cross-validation error obtained in [13] with Back-propagation learn-

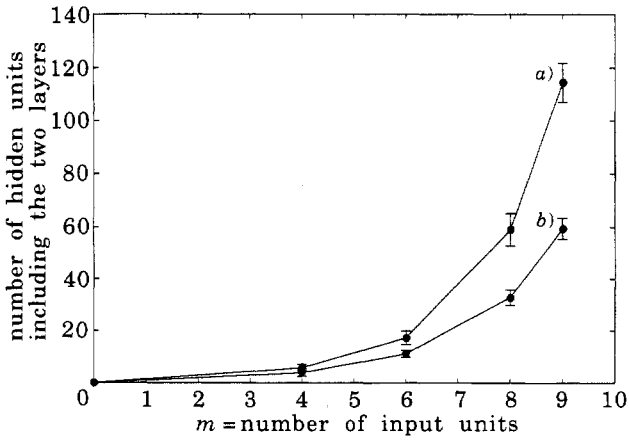

Fig. 3. – Random mapping, mean number of hidden units built by the Offset algorithm before (a)) and after (b)) the pruning step. There are 2^m patterns.

ing. However, in opposition to the «Offset» network, the Back-propagation network was trained with not normalized input data.

7. Conclusion.

We have demonstrated the basic steps of a new algorithm, called the Offset algorithm, for building and training multilayer networks. The Offset algorithm differs from other building algorithms based on an error strategy, such as Tiling, Upstart algorithms or Golea and Marchand approach, for several reasons. Firstly, the Offset strategy considers a single type of unit and a single type of error, so errors are dealt with as a whole. Furthermore, after the growth step, a pruning step removes any redundant hidden units in the second layer. This pruning step is very simple but not the optimum solution in some cases. However, we obtained in practice a reduction in the size of the network of approximately 40% on random mappings, and we are currently working to optimize pruning procedure without making it too compute-intensive. The size of networks built by the Offset algorithm is smaller than with the Tiling algorithm or Golea and Marchand approach and is practically the same as that obtained with Upstart, even when using a nonoptimum pruning process. Our results therefore seem promising. Presently, an hardware implementation of the Offset algorithm with standard logic gates is in design phase. The network implementation is greatly facilitated by the algorithm simplicity and the use of neurons with binary output.

* * *

This study was carried out within the scope of research done at LAAS-SIP on automatic detection for automotive applications. We should particularly like to thank our partners in the European programmes DRIVE and PROMETHEUS.

REFERENCES

[1] MINSKY M. and PAPERT S., *Perceptrons* (MIT Press, Cambridge) 1969.
[2] BAUM E. B. and HAUSSLER D., *Neural Computation*, **1** (1989) 151.
[3] RUMELHART D. E., HINTON G. E. and WILLIAMS R. J., *Nature*, **323** (1986) 533.
[4] MÉZARD M. and NADAL J., *J. Phys. A*, **22** (1989) 2191.
[5] NADAL J., *Int. J. Neural Syst.*, **1** (1989) 55.
[6] FREAN M., *Neural Computation*, **2** (1990) 198.
[7] GOLEA M. and MARCHAND M., *Europhys. Lett.*, **12** (1990) 205.
[8] GALLANT S. I., *Proceedings of the VIII International Conference on Pattern Recognition* (1986), p. 849; *IEEE Trans. Neural Networks*, **1** (1990) 179.
[9] MITCHISON G. J. and DURBIN R. M., *Biol. Cybern.*, **60** (1989) 345.
[10] BARKAI E., HANSEL D. and KANTER I., *Phys. Rev. Lett.*, **65** (1990) 2312; BARKAI E. and KANTER I., *Europhys., Lett.*, **14** (1991) 107.
[11] FISHER R., *Ann. Eugenics*, **7** (1936) 179.
[12] MARTINEZ D., *Dealing with multivalued inputs for Offset algorithm*, in preparation.
[13] WEISS S. M. and KAPOULEAS I., *International Joint Conference on Artificial Intelligence* (1989), p. 781.