SUCCESSES AND FAILURES OF BACKPROPAGATION: A THEORETICAL INVESTIGATION

P. Frasconi, M. Gori, and A. Tesi

Dipartimento di Sistemi e Informatica, Universitá di Firenze
Via di Santa Marta 3 - 50139 Firenze - Italy
Tel. (39) 55-4796265 - Telex 580681 UNFING I - Fax (39) 55-4796363
e-mail : marco@ingfi1.cineca.it

1 Introduction

Backpropagation is probably the most widely applied neural network learning algorithm. Backprop's popularity is related to its ability to deal with complex multi-dimensional mappings. In the words of Werbos [56] the algorithm goes "beyond regression". Backprop's theory is related to many disciplines and has been developed by several different research groups. As pointed out by le Cun [38], to some extent, the basic elements of the theory can be traced back to the famous book of Bryson and Ho[9]. A more explicit statement of the algorithm has been proposed by Werbos [56], Parker [43], le Cun [36], and members of the PDP group [44]. Although many researchers have contributed in different ways in the development and proposition of different aspects of Backprop, there is no question that Rumelhart and the PDP group have the credit for the current high diffusion of the algorithm. As Widrow points out in [57], what is actually new with Backprop is the adoption of "squashing" instead of "hard-limiting" nonlinearities. That idea never occurred to neural network researchers throughout the sixties. More importantly, thanks to that idea, algorithms based on optimization of "regular" functions can be used for the learning process. In addition to this nice property, which opened the doors to a different methodology with respect to that of the Perceptron learning algorithm, the

squashing functions allow to deal with decision-like tasks. Attracted by Backprop's interpolation capabilities, mainly because of its possibility of going beyond the bound of linearly-separable patterns, several researchers have used the algorithm for very different problems ranging from automatic control to pattern recognition. Specialized journals, international conferences, as well as more classical journals contain several significant applications of the algorithm in different tasks. Apart from the enthusiasm deriving from the novelty of the discipline, which may sometimes lead to overlooking the experimental results obtained, there are some fields where we already have comparisons with other more classical methods. For example, Waibel *et al.* [54], using TDNN *(Time Delay Neural Networks)*, have obtained results which are slightly better than those obtained with *Hidden Markov Models* (HMM) for problems of phoneme recognition. Unlike these first optimistic results, so far, neural networks cannot compete with HMM for problems like word and sentence recognition, that deal heavily with time.

Many researchers have looked at the powerful interpolation capabilities of Backprop as a mean for learning sophisticated concepts. When looking in this direction one soon becomes interested in finding out what Backprop can, and cannot learn. For example, learning complex boolean functions is far more complicated than solving "ordinary pattern recognition problems". These functions typically involve inputs with a few coordinates, but very complex mappings. As we will show in this chapter, Backprop may fail in discovering the optimal solution for problems of this kind, particularly when dealing with *minimal architectures* [1].

The aim of this chapter is to investigate the reasons for the successes and failures of Backprop with theoretical arguments. This research was motivated by the need to understand the experimental behavior of the algorithm. *Why does it perform well in some problems, and fail in other ones? How is that behavior related to the chosen architecture and the learning environment? What is the role played by the cost function and the squashing nonlinearities used?*

Our belief is that many questions of this kind can be answered, once the Backprop's gradient descent is closely investigated in terms of the shape of the cost function. Such knowledge is fundamental in order to better understand the successes and failures of the gradient descent. As pointed out by numerous researchers, Backprop can be trapped in local minima during gradient descent, and in many of these cases it seems very unlikely that any learning algorithm could perform satisfactorily in terms of computational requirements. The use of powerful parallel architecture, or even of dedicated hardware

[1] We call *minimal* an architecture which uses the minimal number of neurons for solving a given task.

[25, 58] can significantly speed up many practical applications, but we should not ignore the intrinsic limitations of Backprop learning algorithm. These limitations are inherited from the learning by example model itself.

As it will be shown in detail in the following sections, the research on the problem of local minima, which is useful for guaranteeing the convergence to an optimal solution, leads to "large" architectures which typically create problems for attaining a good generalization. This chapter is organized as follows. In the next section we give a review of some contributions on the problem of convergence to the optimal solution. In section 3 we propose a vectorial formulation of Backprop equations which turns out to be very useful for investigating the problem of local minima. In section 4 we prove that, under the hypothesis that the patterns are linearly-separable, no local minima exist in the cost surface. In section 5 we show examples where Backprop gets stuck during gradient descent. The consequences of the proposed results for pattern recognition problems are discussed in section 6. Finally, conclusions are drawn and future research projects are outlined in section 7.

2 What is known on Backprop convergence to optimal solution?

As referred to in the introduction, a lot of efforts have been made on experimental application of Backprop to several fields, ranging from automatic control to pattern recognition. Because of the nonlinearity of the feedforward networks, several aspects of the Backprop learning scheme still remain a "mystery". The convergence of the algorithm is probably the most obscure of those aspects. Since 1986, the date of "Backprop-rediscovery" and more importantly, of its wide diffusion in the scientific community, Rumelhart, Hinton, and Williams [44] tried to understand the convergence of their algorithm. For example, they found out that if all weights start out with equal values — and obviously the solution requires that unequal weights be developed — the network does not learn ([44], p. 330). The "breaking of this symmetry", obtained with random weights, was suggested as a way of escaping from this stationary point. They also found out that local minima can arise also in more complex cases. For example, they proposed learning the task of adding binary words of 2 bits with a Multi Layered Network (MLN) and found out that, using the minimal network configuration, local minima were likely to occur. In order to overcome this problem they suggested using more hidden units than those strictly required by the minimal network, and concluded that with just one more hidden unit the problem of local

minima practically disappeared ([44], p.341-346).

A similar result was obtained by McClelland and Rumelhart [46] when looking closely at a small *identity mapping network* [2]. The net had only 1 input, 1 hidden and one output. When setting the biases of the neurons to 0, and not allowing them to change, a local minimum comes out. Interestingly, the small increase of the number of free parameters due to the biases produces a cost surface with no local minima. As a conclusion to their experiments, Rumelhart *et al.* [46] said "Although our learning results do not *guarantee* that we can find a solution for all solvable problems, our analyses and results have shown that as a practical matter, the error propagation scheme leads to solutions in virtually every case".

On the other hand, other researchers are not convinced of this qualitative explanation. Basically, the investigation on small cases does not allow to conclude on actual applications of Backprop where nets with even 100.000 neurons may be used. In the expanded edition of Perceptrons, Minsky [41] points out that what the PDP group call a "powerful new learning result" is nothing more than a straightforward hill-climbing algorithm. He invites us to look in the actual mathematical structure of Backprop and says: "We have the impression that many people in the connectionist community do not understand that this is merely a particular way to compute a gradient and have assumed instead that back-propagation is a new learning scheme that somehow gets around the basic limitation of hill-climbing."([41], p. 260) [3].

There has been a sort of "folklore" around Backprop convergence. Many researchers, mainly users of the algorithms, have reported no serious troubles and have developed a practical feeling on the design of feedforward nets for solving problems. Basically they soon realized that random weight initialization (with maximum value "not too small and not too large") often leads to the discover of a satisfactory solution. Other researchers, mainly mathematicians or computer scientists in the area of Artificial Intelligence, according to Minsky's remarks on the convergence to optimal solutions, were more skeptical on the actual capabilities of Backprop . Now, about five years later, though the problem of Backprop convergence is still open in many directions, we know several more things about it, and begin to understand the reason for the successes and failures of the algorithm.

[2] An *identity mapping network* has as many inputs as outputs. Its task is that of mapping the inputs onto the outputs (targets equal to inputs) [12, 14].

[3] The criticism raised by Minsky to the Backprop learning scheme also involves the mapping capabilities of feedforward nets. For example, Minsky ([41], p. 265) points out that the net proposed by Rumelhart *et al.* ([44], pp. 340–341) for learning to recognize the symmetry has very serious problems of scaling up. It may happen that the bits needed for representing the weights exceed those needed for recording the patterns themselves!

In order to acquire such knowledge, first we must define clearly what convergence means. If we look at the Backprop weight updating rule as a recurrent equation, we realize that its behavior can be very complex. We are only interested in making this equation asymptotically stable. In so doing the algorithm may be stopped by using several classical criteria. From this point of view the convergence of the algorithm is equivalent to the asymptotic stability of the system describing the weight updating rule. As a result, different rules may lead to very different results. First and second-order methods have been used in several formulations particularly adapted to the problem. The weight updating has been suggested after each pattern instead of accumulating the gradient for the whole learning environment. When using a gradient descent learning scheme Rumelhart *et al.* [44] (p. 330) have suggested the adoption of the *momentum term* which acts like a low pass filter for smoothing the weight changes. Much attention has been paid to the adaptive selection of the *learning rate* [32] and also to second order methods based on different hessian approximations. A problem discovered with second-order methods is that they do not seem to scale up very well with the dimension of the problem [55, 39]. Using an approach more heuristic than formal, Fahlman [15] has developed an algorithm, called *Quickprop*, which is surprisely efficient. The selection of proper numerical techniques for implementing the weight updating rule has a fundamental impact on the actual performance. The previous contributions represent just a first attempt to exploit the rich literature available in the field of numerical methods for optimization. We expect further achievements in the near future on this subject.

Unfortunately, however, as for any nonlinear optimization problem, we do not have "a-priori" guarantees that the numerical solving scheme will approach the optimal solution. The main difficulty is with the "intrinsic shape" of the cost surface which is normally fixed and independent of the learning algorithm. As a result any algorithm must deal with such a surface, and therefore, there are likely to be "easy and difficult" problems, depending on the shape of that surface and not on the learning algorithm. When using weight updating schemes different from batch-mode, however, the shape of the cost surface, though helpful, does not indicate directly what the behavior of the learning algorithm will be. Basically there are at least three different approaches that can be used for predicting the success or failure of the algorithm. First, we can look at the problem of learning from a computational point of view in an attempt to establish its *complexity*. Second, we can analyze the *shape of the cost surface* looking mainly at stationary points with non-optimal cost. Third, we can analyze the behavior of the *recurrent equation* representing the weight updating rule and investigate conditions for the convergence to an optimal solution.

2.1 Some attempts to evaluate the computational complexity

There have been several attempts to evaluate the Backprop computational burden. A first suggestion was provided by Hinton [28] who formulated a conjecture on the convergence time. He established empirically that the learning time on serial machines is very approximately $O(N^3)$ where N is the number of weights in the network [28] [4]. Other researchers have investigated how Backprop scales up w.r.t. the training set size and the predicate order to be learned (e.g. [51, 52, 35]). An interesting analysis on the asymptotical convergence behavior has been proposed by Tesauro [50] who points out that, for large k (epoch number), the error decreases as $1/k$, and that the convergence can never be faster than $1/k$. Other researchers have begun assuming a "pessimistic point of view" based on the belief that the optimization problem to be solved is computationally intractable. Judd, in his doctoral thesis, now published in a book [34] which is conceived for being accessible also for non-specialists in Complexity Theory, after having reformulated the problem of learning in a general framework *(loading problem)*, proposes a collection of theoretical results which basically state that *the loading problem is NP-complete* ([34], p. 51). As pointed out by Baum [3], Judd's conclusions, when analyzed superficially, may lead to very pessimistic conclusions. All Judd's results concern what the author calls the loading problem. In this formulation, it is shown that memorization problems can become intractable if an adversary is allowed to choose the network that we must use to solve them. The network chosen by Judd increases the number of output units for exploring the scaling up behavior. In practice, however, we usually do choose the most appropriate architecture for solving a given task, and therefore, we deal with a problem which is significantly different with respect to that investigated by Judd (loading problem) [3]. Blum and Rivest [6] have studied the computational complexity of the problem of learning in a net with a fixed number of neurons when increasing the number of inputs and the number of patterns. Again, as pointed out by Baum [3], the "negative" result which is found, that states that the problem is NP-complete, does not follow the common practice of "tuning" the network architecture on the problem to be solved.

At the present time, the results found concerning the intractability of learning in feedforward nets are based on assumptions normally not followed during the setting up of the practical experiments. As a consequence, the reasons for Backprop failures in several

[4] Any forward and backward step costs $O(N)$ and is performed for all T patterns. If $T \approx N$, then the cost for each epoch is $O(N^2)$. Moreover, in this evaluation Hinton assumes that the number of epochs needed for the convergence is roughly $O(N)$. On the other hand, we can expect this result to be meaningful provided that the network used is "properly" designed for the learning environment assigned. Obviously, a fixed network with N neurons does not perform in the same way for every task.

problems should be considered carefully. As we will show in this chapter, sometimes, these failures may not be intrinsic with the Backprop learning scheme and can easily be removed (e.g.: see section 5, example 1).

2.2 Studies on the surface attached to the cost

The study of the shape of the cost surface allows us to get an indication of the actual behavior of Backprop. In the case of gradient descent algorithms with batch mode, that shape is a straightforward indicator of successes and failures. Apart from numerical problems due to an approximate implementation of the gradient descent, the absence of local minima guarantees the convergence to the optimal solution.

If we look at the problem of supervised learning in general (see for example the method proposed in the next section), the shape of the cost function depends on several elements. Keeping fixed the pattern of connectivity, the squashing function still plays quite an important role on the cost. Some general requirements deriving from the need to avoid local minima are proposed in section 4. Moreover, the cost function itself may not be necessarily a quadratic one. Basically, what is important with that function is that it must penalize errors with respect to desired values. As a result, different choices of the cost may lead to quite different optimization problems (see next section). Most importantly, the optimization problem to be solved is strongly related to the mapping to be performed by the network. Consequently, the network architecture and the learning environment play a very fundamental role. As already pointed out, the data to be learned are fixed since they are assigned by the problem to be solved, whereas the network designer usually choices the architecture in terms of number of layers, neurons, constraints on weights and so on. These remarks lead to separate the role of parameters like the squashing and cost functions from the architecture and learning environment. When certain requirements are satisfied (e.g.: see section 3), the influence of squashing and cost function is not relevant for the formulation of the learning problem. Although not significant for the learning problem itself, in practice, the actual assumptions of these functions may affect significantly the optimization problem.

Several researchers have investigated the optimization problem derived from the supervised learning scheme typical of Backprop. A first kind of research has been done in an attempt to clearly identify local minima in the cost surface.

Brady *et al.* [8] have found examples where Backprop fails and a Perceptron succeeds. These kind of minima even hold for one-layered network. The analysis of these cases reveals that these minima are related to the squashing and cost functions. For example,

using the functions (for squash and cost) proposed by Rumelhart [44], local minima can arise when assuming "non-asymptotic targets" (e,g.: 0.1, 0.9). As shown in [21], this kind of local minima disappear when choosing "asymptotic values" for the targets. Sontag and Sussman [48] have proposed an example of local minimum in a single-layered network that does not involve the squash, nor the cost functions, but is related with the data assumed (no architectural choices can be done since the input data define the one-layered net). Since the patterns given to the net were not linearly-separable, they could not be learned completely. Examples with local minima for multi-layered networks have been independently found by Blum, Gori, and McInerney [5, 22, 42] and some of them will be discussed in detail in section 5, in the light of the theory proposed in section 4.

Other researchers have proposed more general studies which contribute to understand the shape of the cost surface. Lee *et al.* [40] point out that one of the main reasons for the slow convergence of Backprop is the *premature saturation*. They calculate the probability of saturation in the first learning epochs. The main result is that setting a "small" range of initial weights helps avoiding premature saturation. The results they find give quantitative suggestions for the weight setting in the case of binary input patterns. The same problem has been investigated by le Cun [37] who gives a heuristic criterion for setting up the initial weights. These studies, however, do not investigate the convergence during premature saturation. In [40] it is pointed out that these configurations are related to the mismatching of at least one output target. Gori and Tesi [21] demonstrate that although these configurations are associated with flat surfaces, they do not represent local minima. Therefore, using enough numerical precision, a gradient descent learning algorithm is likely to escape, even though slowly, from these configurations. This fact, which will be briefly discussed in section 6, is one of the main reasons for the success of Backprop .

Beginning from investigation on small examples, Hush *et al.* [31] give some interesting qualitative indications on the shape of the cost surface. As observed by numerous other researchers, they point out that the cost surface is mainly composed of plateaus, that extend to infinity in all directions, and very steep regions. When the number of patterns is "small", they observe "stair steps" in the cost surface, one for each pattern. When increasing the cardinality of the training set, however, the surface becomes smoother. Careful analyses on the shape of the cost surface, also supported by a detailed investigation of an example, are proposed by Gouhara *et al.* [23, 24]. They introduce the concepts of *memory and learning surfaces*. The learning surface is the surface attached to the cost function, whereas the memory surface is the region in the weight space which represents the solution to the problem of mapping the patterns onto the target values. Some in-

teresting relationships between these two concepts allow us to introduce an interesting taxonomy of local minima. One of their main conclusions is that the learning process "has the tendency to descend along the memory surfaces because of the valley-hill shape of the learning surface". They also suggest what the effect of the P and S symmetries [5] [27, 33] is on the shape of the learning surface.

There have also been some attempts to offer sufficient conditions for avoiding local minima. In [1], under the hypothesis that the neurons are linear, it has been proved that only one minimum exists and that the other points where the gradient is null are saddle points. However, as also pointed out by the authors, it does not seem easy to generalize it to actual architectures based on nonlinear neurons. In [49], for the particular case of networks with no hidden layers, the global convergence of Backprop has been proved for linearly-separable patterns for the case of LMS threshold-penalty functions. In [21], under the assumption of quadratic cost and that the targets are equal to the asymptotic values of the squashing functions, the optimal convergence have been proved in the case of linearly-separable patterns also for networks with hidden units. This result will be extended in this chapter to the case of LMS threshold-penalty cost functions. Other extensions concerning the removal of some hypotheses on the network architecture will also be referred to. A limit to the theory proposed for the derivation of these convergence results is that the number of hidden units is not directly taken into account. Poston *et al.* [45] propose a first investigation on the role of such neurons for avoiding local minima. They point out that if there are as many hidden units as patterns then "almost certainly *(i)* a solution exists, and *(ii)* the error function has no local minima." Unfortunately, due to the limitation of space, only a trace of the proofs is reported in [45]. The result is quite interesting, from a theoretical point of view, since it represents a quantitative statement supporting the well-known fact that the problem of local minima can be contracted by using more hidden units ("large nets").

Unfortunately, in practice, Poston's conclusion is not useful, since the design of networks like those suggested by his hypotheses would lead to cumbersome nets which are likely not to generalize to new examples. An attempt to obtain more practical conditions can be found in [17].

[5] P and S symmetries are weight transformations that do not affect the network output. The P symmetry can act on any vector W_i of input weights of a given hidden neuron i. The vectors of the hidden neurons of an assigned layer can be permuted in any order, since their global contribution to the upper layer is not affected at all. The S symmetry acts for symmetric squashing functions such that $f(a) = -f(-a)$. In this case a transformation of the weights can be created which inverses the sign of all the input and output connections of a neuron.

2.3 Optimal convergence analysis based on the weight updating equations

Another method for analyzing the algorithm behavior is that of investigating the weight updating equations. An example of such method is the Rosenblatt's Perceptron convergence proof (see for example [41], Chap. 11). Unlike the analysis on the shape of the cost, this method gives a straightforward indication on the actual behavior of the algorithm used. Said another words, the investigation on different learning modes (pattern mode, block mode, and batch mode) normally proceeds separately for each mode, the weight updating equation being different one each other. To the best of our knowledge, no paper reports research using this method. If compared with the study of the cost surface, this is probably due to the additional difficulty of managing the actual weight updating equation. The choice of parameters like learning rate and momentum in Backprop may affect significantly the convergence, but may also hidden the essence of the problem.

3 Backprop: a brief review and notation

An introduction on Backprop's theory can be found in numerous books and journals, under different points of view [26, 36, 43, 44, 56, 57]. In this section we review Backprop's basic principles and introduce a vectorial formulation of the equations which will turns out to be very useful for investigations on the local minima. Basically, the problem of learning in MLNs is to find a set of weights which minimizes the mismatching between network outputs and target values. This strategy is referred to as supervised learning. More formally, there is a network $\mathcal{N}$, a learning environment $\mathcal{L}_e$ (set of data used for learning), and a cost index E_T [44].

- *Network* $\mathcal{N}$ [6].

  It has a Multi-Layered architecture (see Fig. 1). With reference to index l, we distinguish among the input layer ($l = 0$), the hidden layer ($l = 1$), and the output layer ($l = 2$). The number of neurons per layer is denoted by $n(l)$. In particular, the output layer has one neuron, i.e. $n(2) = 1$. Each neuron of layer l is referred to by its index $i(l)$, $i(l) = 1, \ldots, n(l)$.

[6] In the sequel we will also consider a network with more than one output (see Fig. 3). This more general architecture can be thought of as the union of networks like that in Fig. 1.

When pattern t is presented at the input, for each neuron, we consider

$$\begin{array}{lll} a_{i(l)}(t) & : & \text{neuron } i(l)\text{'s activation} \\ x_{i(l)}(t) & : & \text{neuron } i(l)\text{'s output.} \end{array} \tag{1}$$

The activation is computed by propagating forward the outputs of the neurons of the previous level as follows

$$a_{i(l)}(t) = w_{i(l)} + \sum_{i(l-1)=1}^{n(l-1)} w_{i(l),i(l-1)} x_{i(l-1)}(t), \tag{2}$$

where $w_{i(l),i(l-1)}$ denotes the weight of the link between the neurons $i(l), i(l-1)$ and $w_{i(l)}$ is the threshold (see Fig. 1).
The output of neuron $i(l)$ is related to the activation by a "squash-like" function

$$x_{i(l)} = f(a_{i(l)}), \tag{3}$$

where $f(\cdot) : R \to [\underline{d}, \overline{d}]$ is a C^2 function with a positive first derivative.

- *Learning Environment* $\mathcal{L}_e$.
  We use a set of supervised data for learning. It is convenient to think of it as a collection of T input/output pairs for network $\mathcal{N}$

$$\mathcal{L}_e \doteq \left\{ \left(X_o(t), d(t)\right), X_o(t) \in R^{n(o)}, \quad d(t) \in R, t = 1, \ldots, T \right\}. \tag{4}$$

  $X_o(t)$ is the input pattern and $d(t)$ is its corresponding target. We assume that there are two classes $\mathcal{C}_1$ and $\mathcal{C}_2$, having T_1 and T_2 patterns ($T_1 + T_2 = T$), respectively. If pattern t belongs to class $\mathcal{C}_1$ ($\mathcal{C}_2$), then

$$d(t) = d_1, \qquad (d(t) = d_2),$$

  where $[d_1, d_2] \subset [\underline{d}, \overline{d}]$. All the targets are collected in the vector $\mathcal{D} \doteq [d(1), \ldots, d(T)]' \in R^T$.
  We recall that the patterns of $\mathcal{L}_e$ are *linearly-separable*, if a vector $A \in R^{n(0)+1}$ exists such that

$$\begin{array}{ll} A' X_o^e(t) > 0 & \text{if pattern } t \in \mathcal{C}_1 \\ A' X_o^e(t) < 0 & \text{if pattern } t \in \mathcal{C}_2 \end{array} \tag{5}$$

  where $X_o^e(t) \doteq [X_o'(t)\ 1]'$.

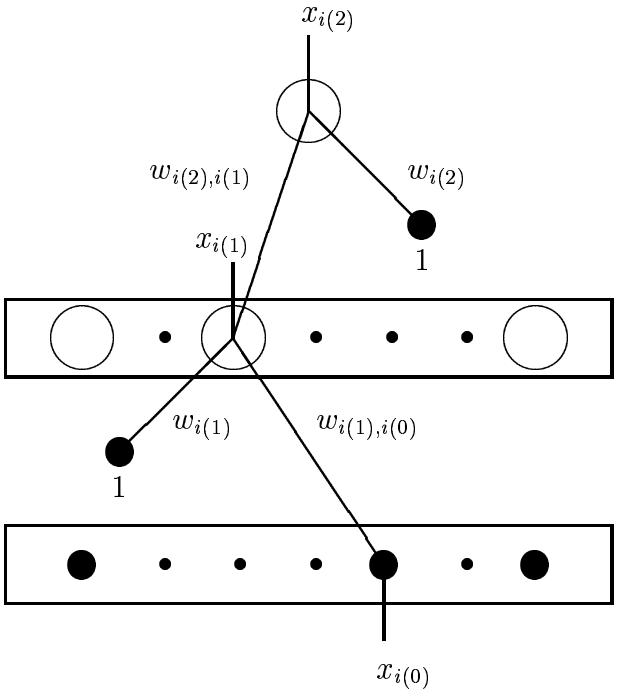


Figure 1: Network architecture and symbol definitions.

- *Cost index.*
  For a given $\mathcal{L}_e$, the input-output data fitting is measured by means of the cost function

$$E_T = \sum_{t=1}^{T_1} l_1(x_{i(2)}(t) - d_1) + \sum_{t=T_1+1}^{T} l_2(x_{i(2)}(t) - d_2) \tag{6}$$

  where $l_k(\cdot) : R \to R$, $k = 1, 2$ are C^2 functions such that

$$\begin{array}{ll} l_1(\alpha) = 0 & \text{if } \alpha \leq 0 \\ l_1(\alpha) > 0, \quad l_1'(\alpha) > 0 & \text{if } \alpha > 0 \\ l_2(\alpha) = 0 & \text{if } \alpha \geq 0 \\ l_2(\alpha) > 0, \quad l_2'(\alpha) < 0 & \text{if } \alpha < 0 \end{array} \tag{7}$$

  where $'$ stands for differentiation with respect to α. In Fig. 2 an example of functions $l_1(\cdot)$ and $l_2(\cdot)$ is reported.

In order to develop a BP vectorial formulation (see also [21]), we need some additional definitions.

1. $X_{i(l)} \doteq [x_{i(l)}(1), \cdots, x_{i(l)}(T)]' \in R^T$ is called *output trace* of neuron $i(l)$. This vector stores the output of neuron $i(l)$ for all the T patterns of the learning environment.

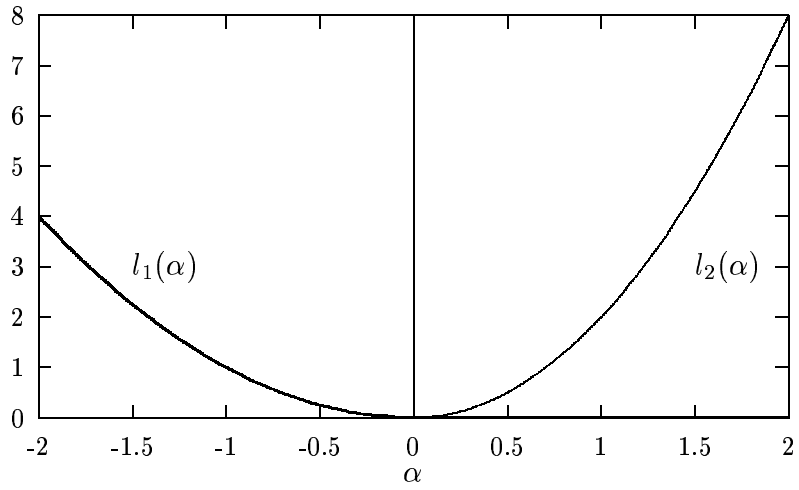


Figure 2: An example of functions $l_1(\cdot)$ and $l_2(\cdot)$.

2. The output trace for all the neurons of a given layer l is called *output layer trace*. It is kept in the matrix

$$\mathcal{X}_l \doteq [X_{1(l)} \cdots X_{n(l)} \ \Pi] \ \in R^{T,n(l)+1} \quad 0 \leq l \leq 2 \tag{8}$$

where $\Pi \doteq [1 \cdots 1]' \in R^T$ is used to deal with biases.

3. $w_{i(l),i(l-1)}$ is the weight which connects neuron $i(l-1)$ to neuron $i(l)$ (see Fig. 1). The associated matrix $\mathcal{W}_{l-1} \ \in R^{n(l),n(l-1)}$ is referred to as *weight layer matrix*, $l = 1, 2$.

4. We assume $y_{i(l)}(t) \doteq \partial E_t / \partial a_{i(l)}(t)$ and we call *delta trace* the vector

$$Y_{i(l)} \doteq [y_{i(l)}(1) \cdots y_{i(l)}(T)]'$$

and *delta layer trace* the matrix $\mathcal{Y}_l \doteq [Y_{1(l)} \cdots Y_{n(l)}] \ \in R^{T,n(l)}$ $(l = 1, 2)$.

5. For weights connecting layers l and $l-1$ the gradient can be represented by a matrix $\mathcal{G}_{l-1} \in R^{n(l-1)+1,n(l)}$, whose generic element $g_{i(l-1),i(l)}$ is given by $\partial E_T / \partial w_{i(l),i(l-1)}$ if $i(l-1) \leq n(l-1)$ and $\partial E_T / \partial w_{i(l)}$ if $i(l-1) = n(l-1)+1$ (bias term gradient contribution).

With these definitions the gradient matrix can be computed as follows

$$\mathcal{G}_{l-1} = \mathcal{X}'_{l-1} \mathcal{Y}_l. \tag{9}$$

Each row t of matrix $\mathcal{X}_{l-1}$ can be computed by feeding the network with pattern t and by propagating forward activations and outputs (eqs. (2), (3), *forward step*). Matrix $\mathcal{Y}_l,\ l = 1, 2$ can be computed by the following equation

$$\tilde{\mathcal{Y}}_1 = \mathcal{Y}_2 \mathcal{W}_1 \tag{10}$$

where the generic element of $\tilde{\mathcal{Y}}_1$ is [7]

$$\tilde{y}_{i(1)}(t) \doteq y_{i(1)}(t)/f'(a_{i(1)}(t)).$$

We can directly compute $\mathcal{Y}_2$ by deriving the cost w.r.t. the output neuron activation, and then go backwards to compute $\mathcal{Y}_1$ — eq. (10), *backward step*. The gradient computation for all the M weights of a network and for all the T patterns takes $O(M \times T)$ time, versus $O(M^2 \times T)$ of a direct gradient computation based on gradient's definition. It is this reduction of complexity which has made it possible to tackle many practical and interesting problems. The gradient computation is actually very efficient. Unlike other naming conventions found in literature, it would be probably better reserving the Backpropagation name only to this efficient gradient computation scheme and not to the learning algorithm used (weight updating scheme). As already pointed out when discussing on computational complexity (Hinton's conjecture), what makes the learning process slow is the gradient descent, whose number of steps for the optimal convergence cannot easily be predicted.

4 Conditions for the absence of local minima

In this section, we analyze the stationary points of the cost function (6). We investigate what happens when the gradient of E_T w.r.t. weights is zero, in order to discover what configurations cause learning algorithms to get stuck. Our considerations all rely upon the following assumption

A.1 *The entire training set can be learned exactly.*

This hypothesis is verified by networks with just one hidden layer provided that a sufficient number of neurons is chosen in the hidden layer [13, 19, 26, 29]. According to more recent research, the perfect mapping of all the training patterns can also be attained by using at least $T-1$ hidden neurons [30]. Moreover, as can be shown in simple examples, the "$T-1$"

[7] These vectorial equations can be easily derived [20] from the ordinary scalar equations for the gradient computation.

bound can be overcome in many cases — e.g. see the network chosen by Rumelhart ([44], pp. 330–334) for implementing the XOR predicate.

Since the analysis of the stationary points does not depend on the ordering of patterns in $\mathcal{L}_e$, w.l.o.g. we assume that the patterns of the same class are contiguous in the learning environment [8]. In this case, the following Lemma shows that the delta trace vector of the output neuron and of the hidden neurons have a special structure.

Lemma 1 *The generic column $Y_{i(1)} \in R^T$ of matrix $\mathcal{Y}_1 \in R^{T,n(1)}$, gets the following "sign" structure*

$$sign\ [Y_{i(1)}] = \alpha_i \left[e_1^1| - e_1^2\right]', \tag{11}$$

where $e_1^k \doteq [1\ 1\ \cdots\ 1] \in R^{T_k}$, $k = 1, 2$. T_k is the number of patterns per class, and

$$\begin{array}{lll} \alpha_i = \pm 1 & if & w_{i(2),i(1)} \neq 0 \\ \alpha_i = 0 & if & w_{i(2),i(1)} = 0. \end{array} \tag{12}$$

Proof:
For the output layer the following relationship holds

$$\begin{array}{ll} y_{i(2)}(t) = f'(a_{i(2)}(t))l_1'(x_{i(2)}(t) - d_1), & \text{if } t \in \mathcal{C}_1 \\ y_{i(2)}(t) = f'(a_{i(2)}(t))l_2'(x_{i(2)}(t) - d_2), & \text{if } t \in \mathcal{C}_2 \end{array} \tag{13}$$

From assumptions (7) it readily follows that

$$\begin{array}{ll} y_{i(2)}(t) \geq 0 & \text{if } \ t \in \mathcal{C}_1 \\ y_{i(2)}(t) \leq 0 & \text{if } \ t \in \mathcal{C}_2. \end{array} \tag{14}$$

According to the backward relationship of Backpropagation,

$$y_{i(1)}(t) = f'(a_{i(1)}(t))w_{i(2),i(1)}y_{i(2)}(t), \tag{15}$$

it follows that $y_{i(1)}(t)$ gets the same sign structure as $y_{i(2)}(t)$ for all the patterns of a given class. As a result, the thesis follows from (14) and (15). ■

The role of linearly-separable patterns is clarified by the following Lemma.

Lemma 2 *If the patterns of the learning environment are linearly-separable then the equation*

$$\mathcal{X}_o'\mathcal{Y}_1 = 0 \tag{16}$$

[8] Of course, this assumption does not change the cost function.

only admits the solution $\mathcal{Y}_1 = 0$.

Proof:

If the patterns are linearly-separable, a hyperplane exists, defined by vector A, such that

$$\text{sign } [A'\mathcal{X}_o'] = \left[e_1^1| - e_1^2\right], \tag{17}$$

where $e_1^k \doteq [1\ 1\ \cdots\ 1] \in R^{T_k}$, $k = 1, 2$. The solutions $\mathcal{Y}_1$ of (16) must necessarily satisfy the following equation

$$A'\mathcal{X}_o'\mathcal{Y}_1 = 0$$

and, therefore, for each neuron $i(1)$ of the hidden layer, the following scalar equality must hold

$$(A'\mathcal{X}_o')\, Y_{i(1)} = 0.$$

From (17) and (11) of Lemma 1, it easily follows that $\mathcal{Y}_1 = 0$ is the unique solution of (16). ■

Before proving the main result, we need the following Lemma concerning a particular structure of stationary points.

Lemma 3 *Let the patterns be linearly-separable and let us assume the existence of a stationary point whose weights connecting the output to the hidden layer all are null, i.e.*

$$w_{i(2),i(1)} = 0, \qquad \forall\ i(1) = 1, \ldots, n(1). \tag{18}$$

Then, this stationary point is not a local minimum.

Proof:

Condition (18) implies that

$$\begin{aligned} y_{i(2)}(t) &= f'(w_{i(2)})l_1'(f(w_{i(2)}) - d_1), \qquad \text{if } t \in \mathcal{C}_1 \\ y_{i(2)}(t) &= f'(w_{i(2)})l_2'(f(w_{i(2)}) - d_2), \qquad \text{if } t \in \mathcal{C}_2 \end{aligned} \tag{19}$$

where $w_{i(2)}$ is the bias for the output neuron.

From smoothness assumption on functions $l_1(\cdot)$ and $l_2(\cdot)$, it follows that the hessian matrix can be evaluated for the stationary points defined by equations (18). The hessian matrix $\mathcal{H}$ can be partitioned as follows

$$\mathcal{H} = \begin{bmatrix} \mathcal{H}(1,1) & \mathcal{H}(1,0) \\ \mathcal{H}(0,1) & \mathcal{H}(0,0) \end{bmatrix}, \tag{20}$$

where $\mathcal{H}(1,1) \in R^{n(1)+1,n(1)+1}$ and $\mathcal{H}(0,0) \in R^{(n(0)+1)n(1),(n(0)+1)n(1)}$ are generated by the hidden-output and input-hidden weights, respectively, and $\mathcal{H}(0,1) = \mathcal{H}(1,0)' \in R^{(n(0)+1)n(1),n(1)+1}$ represents the cross-contribution of these weights.

We observe that condition (18) implies that the delta trace $Y_{i(1)} \in R^T, \ \forall \ i(1) = 1, \ldots, n(1)$, is identically null. Hence, from BP equations we obtain that $\mathcal{H}(0,0)$ is the null matrix. Now, the generic element of $\mathcal{H}(0,1)$ has the following expression

$$\frac{\partial^2 E_T}{\partial w_{i(1),i(0)} \partial w_{i(2),i(1)}} = \sum_{t=1}^{T} x_{i(0)}(t) f'(a_{i(1)}(t)) y_{i(2)}(t), \tag{21}$$

where $i(0)$ denotes the generic input. Hence, any sub-column $\mathcal{H}^v(0,1) \in R^{n(0)+1}$ of $\mathcal{H}(0,1)$ can be written as

$$\mathcal{H}^v(0,1) = \mathcal{X}_o' \begin{bmatrix} f'(a_{i(1)}(1)) y_{i(2)}(1) \\ \cdots \\ f'(a_{i(1)}(T)) y_{i(2)}(T) \end{bmatrix} \doteq \mathcal{X}_o' \hat{Y}_2, \tag{22}$$

where $\hat{Y}_2 \in R^T$ has the sign structure explained by the right side of equation (17). From Lemma 2 and condition (19) we get that $\mathcal{H}^v(0,1)$ is not identically null. Therefore, matrix $\mathcal{H}$ has the following structure

$$\mathcal{H} = \begin{bmatrix} P & D' \\ D & \emptyset \end{bmatrix}, \tag{23}$$

where matrix P is assumed positive definite (on the contrary the proof of the Lemma is trivial) and D is not the null matrix. By applying Sylvester's theorem (see [4], p. 104) it can be easily obtained that $\mathcal{H}$ has both positive and negative eigenvalues, and, hence, the proof of the Lemma directly follows. ■

Remark 1 We notice that the stationary points assumed in Lemma 3 can actually exist. For example, let us consider a linearly-separable set and a network with a symmetric squash function. In this case, the weight configuration having null all the weights and biases is a stationary point. ■

We are now ready to prove following basic result.

Theorem 1 *The cost function* (6) *has an unique minimum* $E_T = 0$ *if the patterns of the learning environment are linearly-separable.*

Proof:

Two cases are dealt with which refer to different kind of stationary points. First, we consider the stationary points for which the weights connecting the output to the hidden layer all are null. In this case, from Lemma 3, it follows that these points are not local minima. Second, we consider all the other stationary points for which the weights connecting the output to the hidden layer are not all null. In this case, because of learning environment's

hypotheses, from Lemma 2, it follows that any stationary point must satisfy $\mathcal{Y}_1 = 0$. According to BP's backward equation

$$y_{i(1)}(t) = f'(a_{i(1)}(t))w_{i(2),i(1)}y_{i(2)}(t),$$

it follows that $y_{i(2)}(t) = 0$ for all the patterns, since, by assumption, there exists at least a weight $w_{i(2),i(1)} \neq 0$. As a result, all the stationary points also satisfy $\mathcal{Y}_2 = 0$, and therefore $E_T = 0$. ■

If the patterns are nonlinearly-separable, then the Theorem 1's conclusion that $E_T = 0$ is the unique minimum does not hold in general. In the light of the theory proposed in this section, we will investigate the case of nonlinearly-separable patterns in section 5, by discussing some examples where the existence of local minima is known.

Theorem 1 is now extended to other classes of MLNs. Moreover, the classical case of quadratic cost with the squashing functions proposed in [44] (p. 329) is investigated. The following three extensions discussed.

1. *Multiple Networks* [9]. They have $n(2) = C$ outputs, where C is the number of classes (see Fig. 3). Full connections are assumed from the input to the hidden layer. The hidden layer is divided into C sub-layers $H_1, \ldots, H_c, \ldots, H_C$ and connections are only permitted from any sub-layer to the associated output. The sub-layer H_c contains $n_c(1)$ neurons referred to by the index $i_c(1)$.
   For this class of networks the exclusive coding is used for the output, i.e. if pattern t belongs to class c, $(c = 1, \ldots, C)$, then the target vector $D(t) \in R^{n(2)}$ has all the entries equal to d_1 apart of the entry $i(2) = c$ given by d_2

   $$D(t) = [d_1, \ldots, d_2, \ldots, d_1]'.$$

   Obviously, denoting by T_c the number of patterns belonging to class c, the cost function turns out to be

   $$E_T = \sum_{i(2)=1}^{n(2)} \left(\sum_{t=1}^{T-T_c} l_1(x_{i(2)}(t) - d_1) + \sum_{t=T-T_c+1}^{T} l_2(x_{i(2)}(t) - d_2) \right) . \qquad (24)$$

2. *Full Rank Pyramidal Networks.* These networks have one hidden layer. $\mathcal{W}_1$ is a full rank matrix and $n(2) \leq n(1)$.

[9] These networks have been introduced by Gori and Tesi in [21].

3. *Cost function with asymptotic targets* ($d_1 = \underline{d}$ and $d_2 = \overline{d}$). The cost function (24) is changed to

$$E_T = \sum_{t=1}^{T} \sum_{i(2)=1}^{n(2)} l(x_{i(2)}(t) - d_{i(2)}(t)), \tag{25}$$

where $d_{i(2)}(t) = d_1$ or $d_{i(2)}(t) = d_2$ and where $l(\cdot)$ is a generic nonnegative C^2 function, such that

$$\begin{array}{llll} l(\alpha) = 0 & l'(\alpha) = 0 & \text{if} & \alpha = 0 \\ l'(\alpha) < 0 & & \text{if} & \alpha < 0 \\ l'(\alpha) > 0 & & \text{if} & \alpha > 0 \end{array} \tag{26}$$

where $'$ stands for differentiation w.r.t. α. We notice that the choice $l(\alpha) = \alpha^2/2$ gives the standard quadratic cost.

In order to state results on optimal convergence, some asymptotical conditions on the squashing functions are needed. The activation function $f(\cdot)$ must satisfy the following conditions

$$\begin{array}{ll} \lim_{a\to\infty} \frac{f'(a)}{\overline{d}-f(a)} = \overline{k}_1 & \lim_{a\to\infty} \frac{-f''(a)}{\overline{d}-f(a)} = \overline{k}_2 \\ \lim_{a\to-\infty} \frac{f'(a)}{f(a)-\underline{d}} = \underline{k}_1 & \lim_{a\to-\infty} \frac{f''(a)}{f(a)-\underline{d}} = \underline{k}_2, \end{array} \tag{27}$$

where $\underline{k}_1, \overline{k}_1, \underline{k}_2, \overline{k}_2$ are real positive constants, and $f'(\cdot), f''(\cdot)$ denote the first and second derivatives, respectively. For example, a squashing function ([44], p. 329) satisfies these hypotheses.

The Multiple Network definition makes it clear that their behavior depends on C one-output independent networks, like the one in Fig. 1. When assuming exclusive coding for the outputs, the convergence result proved in Theorem 1 can be extended to Multiple Networks as follows.

Theorem 2 *The multiple outputs network cost function* (24) *has an unique minimum* $E_T = 0$ *if the patterns of the learning environment are linearly-separable.* ■

A formal proof of this result can be found in [16]

Remark 2 As a further observation we notice that in this case the other possible stationary points are only those whose weights connecting the neurons of hidden sub-layer H_c to corresponding output c all are null, i.e.

$$w_{c,i_c(1)} = 0, \qquad \forall i_c(1) \in H_c. \tag{28}$$

■

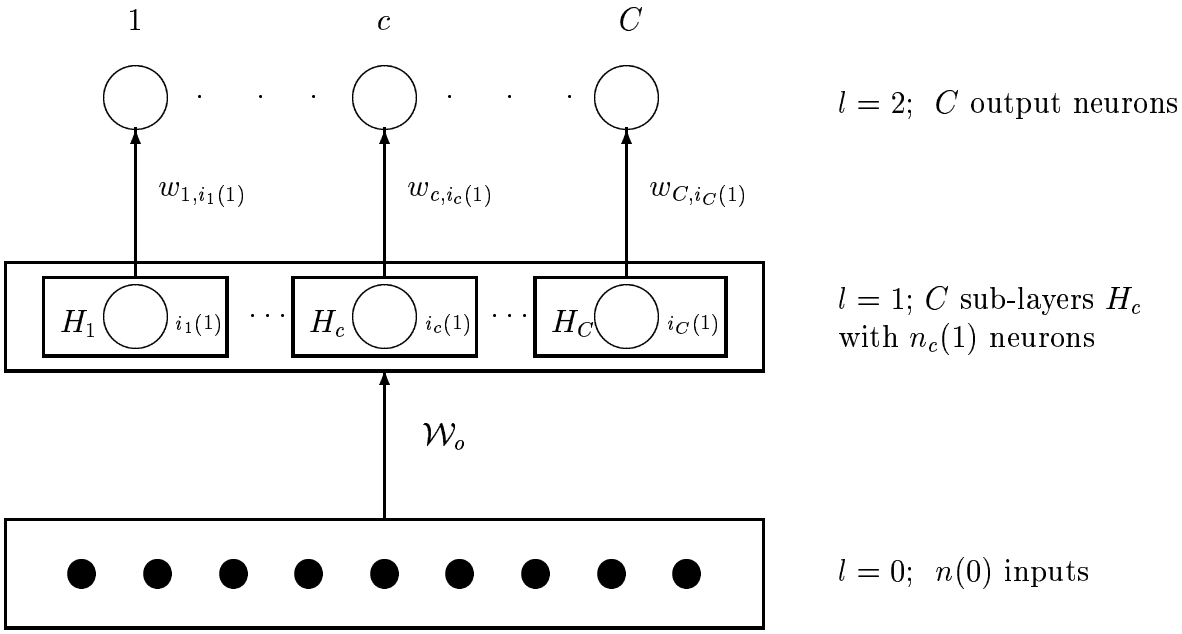


Figure 3: Network architecture with multiple outputs.

Theorem 3 *The quadratic cost function with asymptotic targets of a Multiple Network with squashing functions satisfying assumption (27), has an unique minimum $E_T = 0$ if the patterns of the learning environment are linearly-separable.* ■

The proof of this Theorem does not directly follow from that of Theorem 1, because the presence of asymptotic targets requires a more elaborate analysis. The details of the proof can be found in [21]. It is worth mentioning that, as for Theorem 1, all the weight configurations described in Remark 2 are saddle points.

Theorem 4 *Any LMS threshold-penalty cost function of a Full Rank Pyramidal Network has an unique minimum $E_T = 0$ if the patterns of the learning environment are linearly-separable.* ■

The proof of this result can be found in [16]. This result is somewhat more general than those of Theorems 2 and 3. Apart from the restriction due to the pyramidal assumption [10] no restrictions are placed on the connections between hidden and output layers. Unlike Multiple Networks, however, we do not have, so far, a clear indication of what happens for configurations in which $\mathcal{W}_i$ is not full rank.

[10] This hypothesis is met in many experiments of pattern recognition where the number of hidden units is typically greater than the number of outputs. Going toward the outputs the pattern representation is more compressed.

5 Examples of local minima

In this section we discuss some examples of MLNs where BP can get stuck in local minima. Basically, these examples belong to two different classes, depending on the fact that they are associated with cost and squashing function chosen, or they are intrinsically related to the network and learning environment.

Example 1 The first example is taken from [8], where a single-layered sigmoidal network is considered. The following linearly-separable learning environment is selected for minimizing the quadratic cost

$$\mathcal{L}_e = \Big\{([x_0, x_1]', d)\Big\} = \Big\{([-1, 0]', 0.1), ([1, 0]', 0.9), ([0, 1]', 0.9), ([0, -5]', 0.9)\Big\} \qquad (29)$$

For the sake of simplicity, the explicit dependence of x_o, x_1, and d on t has been omitted. It turns out that there exists a nonseparating local minimum. From the results presented in section 4, it follows that the presence of this minimum is due to the fact that the asymptotic values ($\underline{d}$, $\overline{d}$) are not used as targets as required for the quadratic cost function. If asymptotic values were used, then local minima no longer hold. The reason of the failure can be traced out in the light of the theory proposed in section 4. In particular it is important to consider the $\mathcal{Y}_i(1)$'s sign. In this case Lemma 1 cannot be applied because it uses a particular quadratic cost definition, based on the functions $l_1(\alpha)$ and $l_2(\alpha)$. However, it is quite simple to show that Lemma 1 also holds if we use a standard quadratic cost and asympotical target values (see [21]). The "sign-structure" with $\mathcal{Y}_i(1)$'s is associated with that of $\mathcal{Y}_2$, and depends strictly on the asymptotical target assumption. In [8] Brady *et al.* have proposed other similar examples concerning linearly-separable patterns. The common characteristic in these examples is that the patterns are not of the same "importance" (for example, in eq. (29) the last pattern has a module sensibly greater than the others). It is quite intuitive that for such a set of patterns, a strong learning decision (for instance, asymptotic target values) must be used for BP to work properly. ■

Example 2 In [48] Sontag and Sussman propose an example of local minimum in a single-layered network which has no bias and the symmetric squash function as activation function. The quadratic cost function is minimized with reference to the following linearly-separable learning environment

$$\mathcal{L}_e = \left\{ \begin{array}{lll} ([1,1,1,-1,-1]',1), & ([1,1,-1,1,-1]',1), & ([1,-1,1,-1,1]',1) \\ ([-1,1,1,-1,1]',1), & ([-1,1,1,1,-1]',1), & ([-1,-1,-1,1,1]',1) \\ ([-1,-1,1,-1,1]',1), & ([-1,1,-1,1,-1]',1), & ([1,-1,-1,1,-1]',1) \\ ([1,-1,-1,-1,1]',1), & ([1,1,1,1,1]',1) & \end{array} \right\}. \qquad (30)$$

This example is different with respect to the one previously discussed in that it assumes asymptotical target values. In this case the presence of the local minimum is due to the fact that the chosen network (without bias) is not able to exactly learn the patterns. Hence, it turns out that our assumption (A.1) is no longer satisfied. ■

Example 3 This example considers the sigmoidal network for the XOR boolean function (see Fig. 4*a*) with the quadratic cost function and the standard learning environment (see the first four rows of the table reported in Fig. 4*a*).

Following [5], it can be shown that there is a manifold of local minima given by

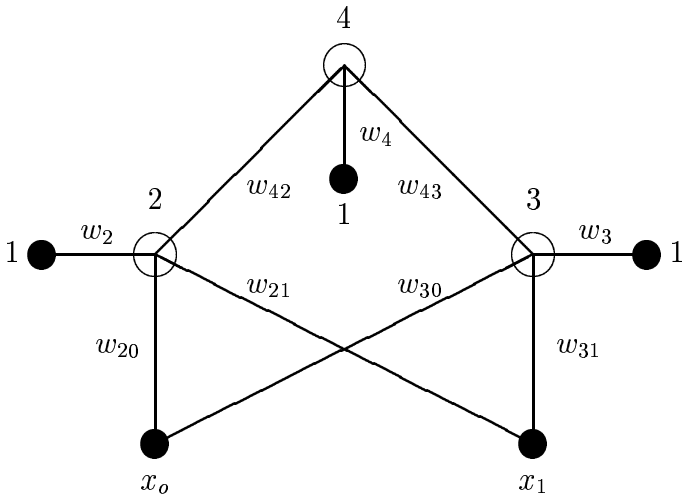
$$\{w_{20} = w_{21} = w_2 = w_{20} = w_{21} = w_2 = 0,\ w_{42} = w_{43},\ w_{42} + w_4 = 0\ \}\ ,$$

with cost $E_T = 0.5$. Such manifold is enlighten in Fig. 5 and Fig. 6, where the cost is plotted as function of two weights at a time. A contour level plot is also shown on the side of the surface, to better clarify the aspect of the manifold. A particular local minimum weight configuration is that having null all the weights. This fact makes it clear how this configuration may be either a saddle point for linearly-separable patterns (see Remark 1), or a local minimum for nonlinearly-separable ones. ■

Example 4 In this example, we consider the same network and cost function of the XOR boolean function of the preceding example, but a different learning environment requiring to learn the additional pattern $E = ([0.5, 0.5]', 0)$ (see Fig. 4*a*). Obviously, also in this case Theorem 3 cannot be applied because the patterns are not linearly-separable. Depending on the initial weights, the gradient can get stuck in points where the cost is far from being null. The presence of these local minima is intuitively related to the symmetry of the learning environment. Experimental evidence of the presence of local minima is given in Fig. 7 and Fig. 8.

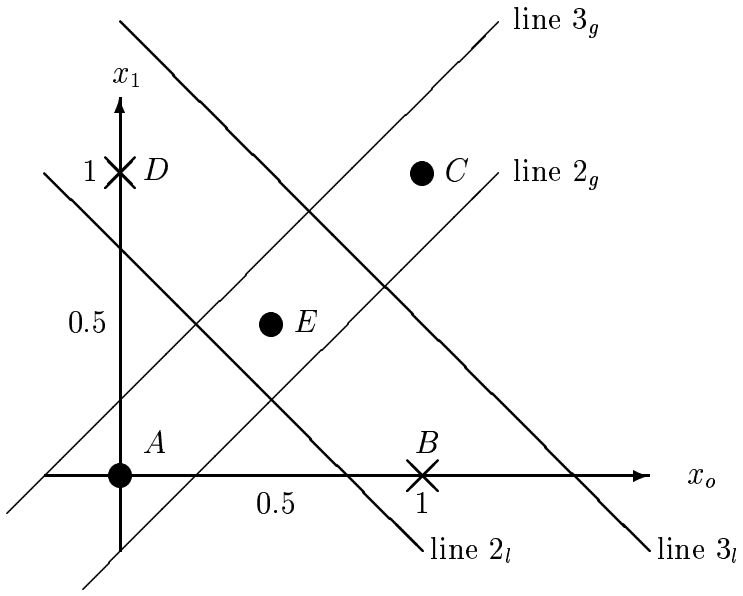
We ran this experiment several times with different initial weights. We found that both these minima are likely to occur no matter which gradient descent algorithm is used.

From a geometrical point of view, these minima are related to the position of the separation lines depicted in Fig. 4*b*. The global minimum is related to configurations $\mathcal{S}_g$ defined by parallel lines 2_g, 3_g, whose directions are identified by vector $[1, 1]'$ ($w_{20} = -w_{21}, w_{30} = -w_{31}$). The local minimum is related to configurations $\mathcal{S}_l$ defined by parallel lines 2_l, 3_l, whose directions are identified by vector $[1, -1]'$ ($w_{20} = w_{21}, w_{30} = w_{31}$). In [20], it is proven that a particular configuration $\mathcal{S}_l$ is a local minimum. Here, we only show that as long as we start from any configuration $\mathcal{S}_l$, the absolute minimum cannot



Pattern	x_o	x_1	Target
A	0	0	0
B	1	0	1
C	1	1	0
D	0	1	1
E	0.5	0.5	0

$-(a)-$



$-(b)-$

Figure 4: (a) : Network and learning environment of example 4. (b) : Separation lines of the local and global minima configurations.

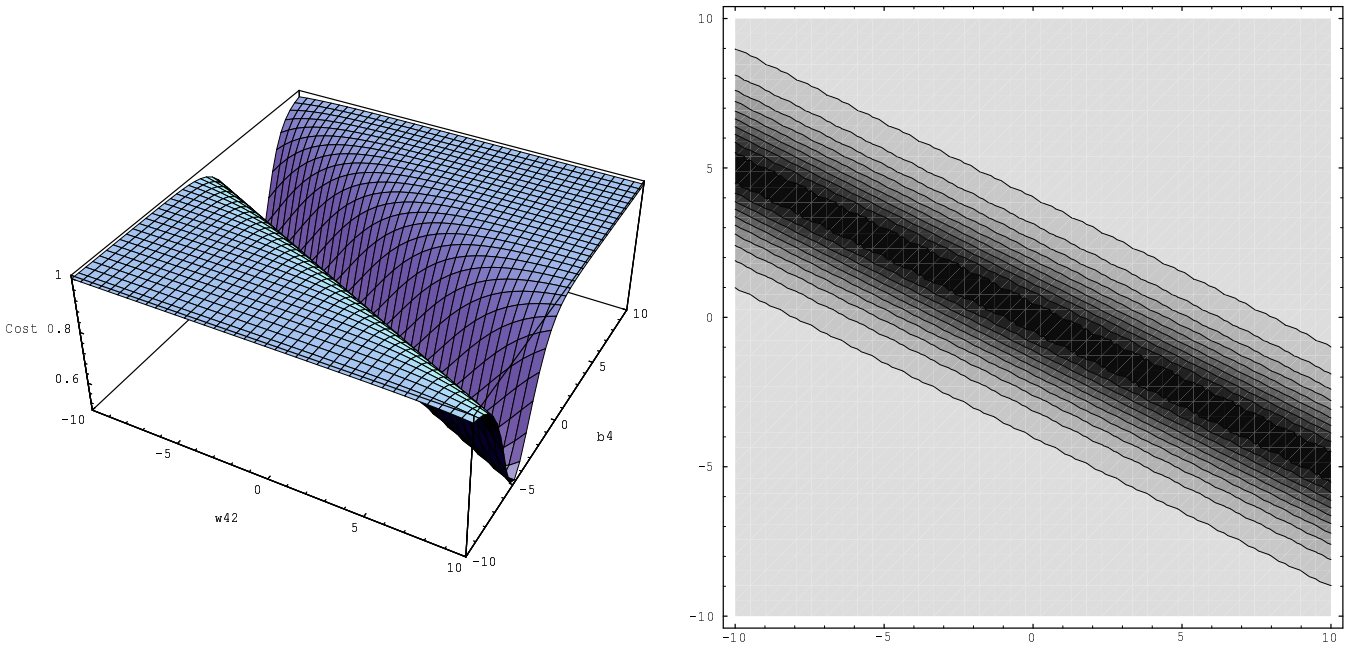


Figure 5: Cost surface as a function of w_{42} and w_4 for example 3 (see text). A contour plot is shown on the right side.

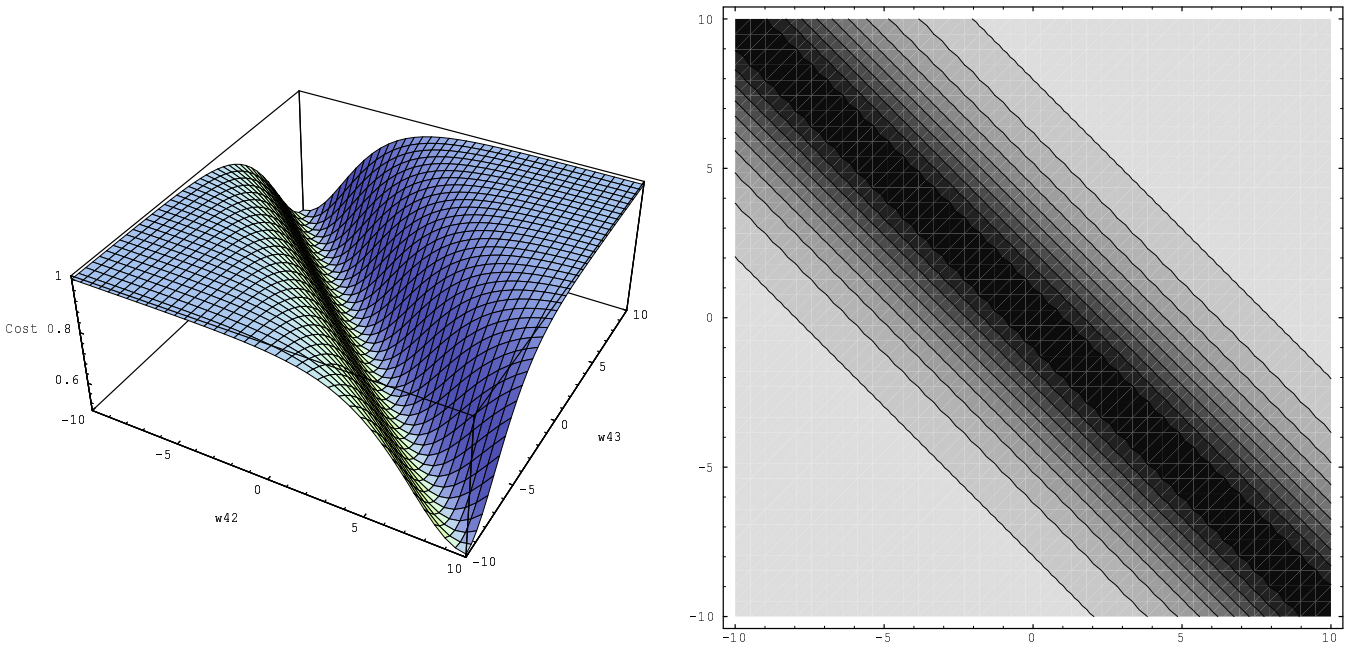


Figure 6: Cost surface as a function of w_{42} and w_{43} for example 3 (see text).

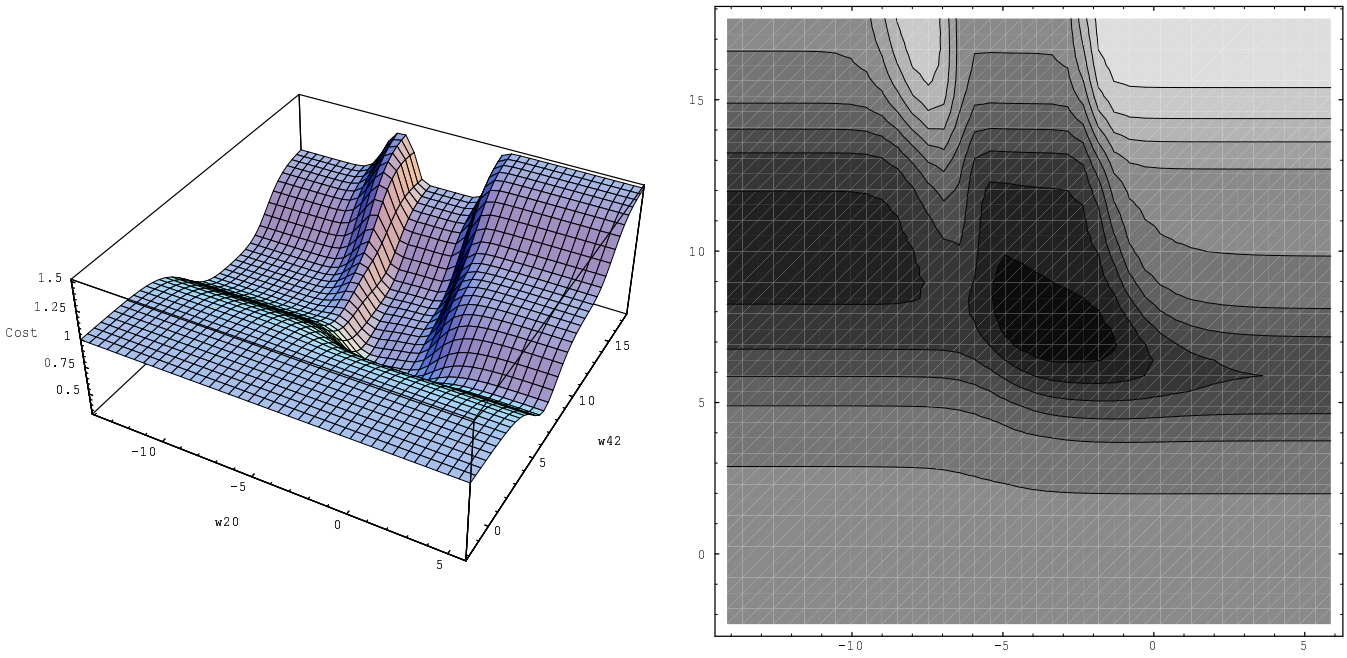


Figure 7: Cost surface as a function of w_{20} and w_{42} for example 4 (see text). On the right side a contour plot is shown.

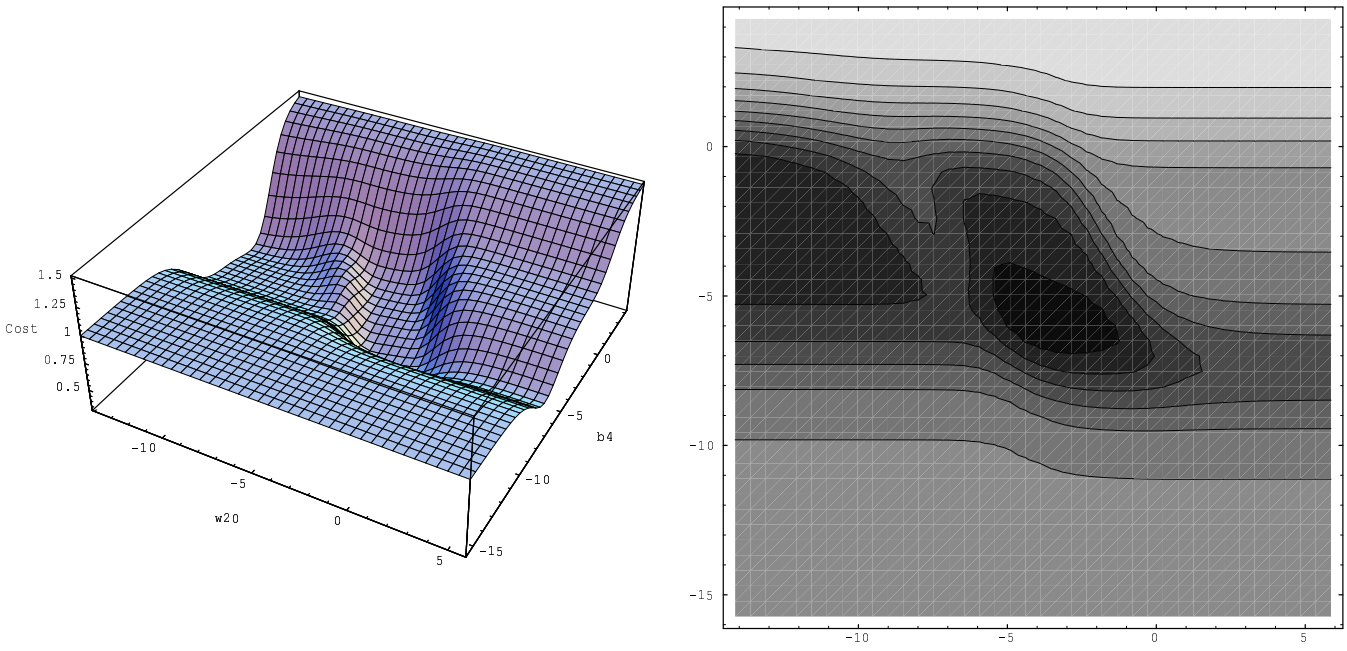


Figure 8: Cost surface as a function of w_{20} and w_4 for example 4 (see text).

be reached. With reference to the simplified notations of Fig. 4*a*, the global contribution of the patterns to $\partial E_T/\partial w_{20}$ and $\partial E_T/\partial w_{21}$ is

$$\begin{array}{l} \frac{\partial E_T}{\partial w_{20}} = y_2(B) + y_2(C) + \frac{1}{2}y_2(E) \\ \frac{\partial E_T}{\partial w_{21}} = y_2(C) + y_2(D) + \frac{1}{2}y_2(E). \end{array} \tag{31}$$

The Backpropagation rule makes it possible to compute y_2 as

$$y_2 = f'(a_2)w_{42}y_4. \tag{32}$$

Because of the symmetry it follows that $a_4(B) = a_4(D)$, i.e. the activation of the neuron 4 is the same for patterns B and D. As a result, $y_4(B) = y_4(D)$ and (32) implies $y_2(B) = y_2(D)$, being $a_2(B) = a_2(D)$, too. From (31) it follows that $(\partial E_T/\partial w_{20}, \partial E_T/\partial w_{21})$ has a direction which is identified by the vector $[1, 1]'$. Hence, the weight updating can only produce parallel translations of line 2_l. The analysis also applies to the separation line 3_l. Therefore, we can conclude that, if we start from any configuration $\mathcal{S}_l$, then the gradient descent only produces parallel translations of separation lines 2_l, 3_l, and that the global minimum (configuration $\mathcal{S}_g$) can never be reached.

We can also predict the actual value of the cost associated with the local minimum. The patterns A and C are in fact learned exactly, whereas the network is unable to recognize the patterns D, E and B, which belong to the region identified by the two separation lines. Because of the configuration's particular geometry, for these patterns the output of the net is exactly the same $(x_4(B) = X_4(D) = X_4(E))$. According to Backpropagation rule, it follows that the gradient components $\partial E_T/\partial w_{42}$ and $\partial E_T/\partial w_{43}$ are null if

$$(x_4(E) - \underline{d}) + (x_4(D) - \overline{d}) + (x_4(B) - \overline{d}) = 0 \tag{33}$$

The conclusion is that the output for these three patterns is $(2\overline{d} + \underline{d})/3$, and the cost is $(\overline{d} - \underline{d})^2/3$. The actual simulation, with $\overline{d} = 1$ and $\underline{d} = 0$, confirms this result. ■

6 Why does Backprop succeed and where?

6.1 It works for linearly-separable patterns

The theory proposed in section 4 extends the conclusions published in [21] for the case of LMS threshold-penalty cost functions. The basic assumption that the patterns must be linearly-separable is somewhat restrictive, but we are confident that it explains the Backpropagation's success in several pattern recognition experiments. Many successful

Publication	Application	Architecture
Bourlard-Wellekens [7]	Phoneme Recognition	1118-200-50
Cosi-Bengio-De Mori [11]	Phoneme Recognition	400-20-10
Elman-Zipser [14]	Phoneme Recognition	320-6-3
le Cun [37]	Handwritten Digit Recognition	256-12-10
Sejnowski [47]	Text to Speech	203-80-26

Table 1: Report of some pattern recognition applications with MLNs.

experiments reported in literature probably deal with quasi linearly-separable patterns. We have experimental evidence to state that the presence of a slight class overlapping, typical of many experiments of pattern recognition, does not significantly affect the convergence of Backpropagation. Quite a common practice is that of using many more inputs than are strictly necessary for discriminating the patterns. This results from the habit of assuming many more inputs than hidden units, since these units extract features which are themselves sufficient to discriminate all the patterns. We may call "many input" net this kind of architecture. Table 1 shows a list of papers recently published which reports different kinds of pattern recognition applications using BP. It is worth mentioning that the number of inputs used largely exceeds the number of hidden units. Moreover, a straightforward confirmation of our belief can be found in [37], where it is explicitly stated that, in experiments of handwritten digit recognition with a 16x16 input grid, the patterns are linearly-separable. In these experiments the entire training set was successfully learned just by using a single-layered network. In order to understand this point better, we investigated the problem of handwritten character recognition in an attempt to establish the role of the input pre-processing. We found that the dimensions of the input grid play a significant role, since the pre-processed patterns turn out to be linearly-separable when "high" resolution is used. For example, 600 handwritten characters, equally distributed in classes A, B, C, were found to be linearly-separable by using a resolution as low as 6x8 pixels. Although single-layered nets also separated the training set, we found that MLNs perform significantly better in generalization to new examples. In the above mentioned experiment, we found that adding a hidden layer of 6 neurons to a net having 48 (6x8) inputs and 3 outputs (A, B, C with exclusive coding) allows us to increase the recognition rate from 3.21% (no hidden layer) to 2.53% (with hidden layer). The same conclusions are reported for handwritten digit recognition in [37] by le Cun, who also noticed a significant reduction of the over-training phenomenon in presence of hidden layers.

6.2 Flat regions

There are flat regions, but Backprop with enough numerical precision does not get stuck there. This is certainly one of the main reasons for the Backpropagation success. As pointed out in section 2, many researchers have seen in the "premature saturation" a serious problem, and have considered it as one of the main reasons for the slow convergence. This is certainly true, but unfortunately, the premature saturation is related to the squashing functions chosen, and basically, with their asymptotic behavior. Some researchers [11] [15] have tried to avoid the typical "squashing saturation" by assuming nonlinear functions which satisfy $f' = f \times (1 - f) + c$, being c a "small" positive constant. In so doing the asymptotical behavior is not a saturation any more. Our experimental conclusion, however, is that this change of the squashing function succeeds in improving the convergence of the algorithm, but does not produce a good generalization to new examples. Basically, we cannot change significantly the "decision role" of the ordinary squashing function. In any case, it seems that in many cases squashing functions with saturation should be used for attaining good generalization. The use of the previous trick could however significantly speed up the convergence time at least during the first learning phase.

The use of a sort of simulated annealing for gradually transforming the nonlinear neuron functions into saturated squashing ones may turn out to be useful. Apart from these remarks concerning the convergence speed up, and studies on proper initial weight setting up to avoid premature saturation, one basic problem of understanding the actual shape of the plateaus associated with saturation. In [21], the following result has been proved which makes it possible to see why a numerical learning algorithm with "enough" numerical precision is expected to succeed in escaping from premature saturation. [12]

Theorem 5 *Let us consider a configuration for which* $\mathcal{Y}_2 \to 0$ *and the cost* $E_T \neq 0$. *In particular, let us assume that there is input-output matching for all the patterns except pattern* $\hat{t}$, *for which there is a mismatch between the target* $d_i(\hat{t})$ *and the output* $f(a_{i(2)}(\hat{t}))$ *for the output neuron* $i(2)$. *Consequently at least one neuron* $i(1)$ *exists such that*

$$\frac{\partial^2 E_T}{\partial w^2_{i(2),i(1)}} < 0. \tag{34}$$

Proof: see [21] ■

[11] We ourselves have experimented the solution of avoiding the "squashing saturation" with significant results concerning the convergence time.

[12] This result holds for asymptotic quadratic cost functions with squashing functions satisfying (27).

This theorem shows that the weight configuration associated with premature saturation are saddles, and therefore, explains why Backprop is likely not to get stuck there.

6.3 Null weights and symmetries

Such configurations may give raise to local minima, that, however, are often not as stable as the absolute minimum. As discussed in section 2, the stationary configurations arising from symmetries were already investigated by Backprop's authors in [44] (pp. 340–341). The conclusions derived by Blum [5], and those arising from the proposed theory for linearly-separable patterns, however, give some more indications on the problem. It comes out that for linearly-separable problems the stationary point associated with null weight is not a local minimum, and therefore does not create very serious convergence problems. Blum's investigation on the XOR net (see section 5) has shown instead, that null weight configurations may be local minima, and therefore may represent an attraction area for learning algorithms. The experiments we have performed however, indicate that the attraction basin is far from being as strong as that of the global minimum. As a result, the Rumelhart *et al.* [46] suggestion of setting up the initial weights with small random values is likely to succeed in most important problems. Suggestions for a proper choice of the magnitude of the random values can be found in [37, 40].

6.4 Going beyond linearly-separable patterns with "many hidden" nets

The basic conclusion of the theory proposed in section 4 is that Backprop converges for linearly-separable patterns. Basically, this conclusion holds for "many input" nets. Unfortunately, in many practical cases, this hypothesis does not hold. What is neglected by the proposed theory is the actual role of the hidden units. The number of hidden neurons is not directly involved in the theorem's proof, apart from the requirement that a null cost solution must exist.

We have recently investigated this problem closely and, using our theoretical framework based on $\mathcal{Y}_{i(l)}$ errors, have found the following result which generalizes that of Poston *et al.* [45]. The condition we found [17] for the guaranteed convergence to an optimal solution is that, for nets having one hidden layer, "enough" units must be chosen such that matrix $\mathcal{X}_1$ is linearly-separable with $\mathcal{P} = 1$ with respect to $\mathcal{W}_0$ matrix [13]. For example, this result allow us to conclude that the XOR predicate is guaranteed to be learned when

[13] $\mathcal{P}$ is the probability that $\mathcal{X}_1$ is linearly-separable associated with the choice of a generic matrix $\mathcal{W}_0$.

using an MLN with at least 3 neurons. In that case, the mapping of the input patterns in the space of the hidden units leads to linearly-separable patterns with $\mathcal{P} = 1$. The main result proposed in [45] instead, is based on the fact $\mathcal{X}_1$ is a full rank matrix and therefore, at least T hidden units are needed for a guaranteed learning to an optimal solution. Noteworthy, in the case of linearly-separable patterns, matrix $\mathcal{X}_1$ is linearly-separable with $\mathcal{P} = 1$.

Our experience with Backprop in complex real nonlinearly-separable patterns is that starting the algorithm from particular initial weights could be convenient. For this reason the investigation of *local attraction* to optimal solutions is a very important issue. In the light of our results, we see that many recent variations to the Backpropagation scheme may turn out to be particularly useful for difficult nonlinearly-separable patterns. For example, many researchers have found it useful to learn huge learning environments block by block. In experiments of speaker independent speech recognition it is common practice to begin learning a few speakers and then increasing the training set when a stop criterion has been met (see e.g. [54]). This procedure is repeated until the whole set of speakers is learned. We have found examples where learning nonlinearly-separable patterns with batch mode BP was very difficult because of the presence of local minima, while more successful results were obtained with the technique of gradually increasing the training set.

7 Conclusions

In this chapter we have proposed an attempt to provide some theoretical reasons for successes and failures of Backprop. The contribution given here concerns mainly the problem of obtaining an optimal convergence of the algorithm. We have reviewed the basic approaches to the problem in an attempt to understand research lines and actual results on the subject. We have seen that the computational theory's approach, based on the study of the complexity of the supervised learning problem, so far, has not given a definite statement concerning the intractability.

The studies on the cost surface, including the one presented here in section 4, have definitely shown that there are theoretical foundations to the experimental success of Backprop. In the case of linearly-separable patterns the algorithm is guaranteed to converge to an optimal solution. The opposite conclusions derived in other papers (see [8]) have been proved to be related to nonasymptotic cost functions. This kind of local minima are somewhat spurious, in that they are related to cost and squashing functions chosen, not to the mapping problem itself. However, a careful choice of these functions does not

suffice to avoid local minima. In section 5, it has been proven that examples can be found where local minima of the cost function are strictly related to the network architecture and the learning environment. Recent research by Poston [45] and [17], however, has shown that there is some theoretical foundation also on the Rumelhart *et al.*'s claims that with enough hidden units local minima become very rare.

However, this theoretical support should not give the novice the impression that the problems related to the convergence of supervised learning algorithm as basically solved. As pointed out by Minsky in [41] (pp..) and Valiant [53] the basic aim of learning is not memorizing patterns, but generalizing to new examples. When dealing with this more general problem, which is actually the only one that is meaningful in practice, there still remains the reduction of what we could call the *learning indetermination principle*. A guaranteed convergence to optimal solutions usually places constraints on the network architecture which makes good generalization to new examples very unlikely. Basically, the more hidden units we use, the higher the probability is to avoid local minima during the gradient descent. On the other hand, that increase of hidden units is likely to reduce the generalization to new examples. From this point of view, there is still the need to obtain learning algorithms capable of discovering optimal solutions with "smaller nets" than those used with Backprop. Although other accurate technical analysis on the shape of the cost surface, and perhaps on the weight updating recurrent equations, may give further interesting results, our belief is that much future research should concentrate on integrating explicit rules with learning by example schemes like Backprop. In so doing the learning scheme is not necessarily called to reinvent acquired knowledge and may be significantly releaved on the computational burden typical of Backprop-like learning procedure (e.g. [18])

Acknowledgments

This research was partially supported by "MURST 40%" and by CNR under grant 90.01530.CT01.

References

[1] P. Baldi and K. Hornik , "Neural Networks and Principal Component Analysis: Learning from Examples and Local Minima," *Neural Networks*, Vol. 2, 1989, pp. 53–58.

[2] E.B. Baum, and D. Haussler, "What Size Net Gives Valid Generalization", *Neural Computation*, Vol. 1, No. 1, 1989, 151–160.

[3] E. B. Baum. Book review: "Neural Network Design and the Complexity of Learning" (J.S. Judd), *IEEE Transactions on Neural Networks*, Vol. 2, No. 1, January 1991, pp. 181–182.

[4] R. Bellman, *Introduction to Matrix Analysis*, McGraw-Hill, Second Edition, New York, 1974.

[5] E. K. Blum, "Approximation of Boolean Functions by Sigmoidal Networks: Part I: XOR and Other Two-Variable Functions", *Neural Networks*, Vol. 1, 1989, pp. 532–540.

[6] A. Blum and R. Rivest, "Training in 3-node neural net is NP-complete", in: D.S. Touresky Ed. *Advance in Neural Information Processing Systems* I, San Mateo, CA: Morgan Kaufman, 1989, pp. 494–501.

[7] H. Bourlard and C. Wellekens, "Speech Pattern Discrimination and Multi-Layered Perceptrons," *Computer Speech and Language*, Vol. 3, 1989, pp. 1–19.

[8] M. L. Brady, R. Raghavan, and J. Slawny,"Back-Propagation fails to Separate Where Perceptrons Succeed," *IEEE Transactions on Circuits and Systems*, Vol. 36, No. 5, 1989, pp. 665–674.

[9] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*, Blaisdell, Waltham Mass, 1969.

[10] Y. Bengio, "Acceleration of Learning in Backpropagation", personal communication.

[11] P. Cosi, Y. Bengio, and R. De Mori, "Phonetically-Based Multi-Layered Networks for Vowel Classification", *Speech Communications*, Vol. 9, No. 1, February 1990, pp. 15–29.

[12] G. Cottrel, P. Munro, and D. Zipser, "Learning Internal Representation of Gray Scale Images: An example of Extensional Programming", in Proc. 9th Annual Cognitive Science Society Conference, Seattle WA, 1987.

[13] G. Cybenko, "Approximation by superpositions of a single sigmoidal function", *Mathematics of Control, Signal and Systems*, Vol. 3, 1989, pp. 303–314.

[14] L. Elman and D. Zipser, "Learning the Hidden Structure of the Speech.", *Journal of Acoustic Society of America*, Vol. 83, No. 4, 1988.

[15] S.E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks", Technical Report CMU-CS-88-162, September 1988.

[16] P. Frasconi, M. Gori, and A. Tesi, "A Technical Note on the Convergence of Backpropagation to Optimal Solution", Technical Report DSI-TR-1/92 Universita' di Firenze, 1992.

[17] P. Frasconi, M. Gori, and A. Tesi, "The Role of Hidden Neurons on the Convergence of Backpropagation", Technical Report DSI-TR-2/92 Universita' di Firenze, 1992.

[18] P. Frasconi, M. Gori, M.Maggini, and G. Soda, "Unified Integration of Explicit Knowledge and Learning by Example in Recurrent Networks", *IEEE Trans. on Knowledge and Data Engineering* (Special Section on Self-Organizing Knowledge and Data Representation in Distributed Environments), 1992.

[19] K. Funahashi, "On the approximate realization of continuous mappings by neural networks", *Neural Networks*, Vol. 2, 1989, pp. 183–192.

[20] M. Gori, "Apprendimento con supervisione in reti neuronali", *PhD Thesis*, Universitá di Bologna, February 1990.

[21] M. Gori, A. Tesi, "On the Problem of Local Minima in Backpropagation", *IEEE Transactions on PAMI*, Vol. 14, No. 1, January 1992, pp. 76–86.

[22] M. Gori and A. Tesi, "Some Examples of Local Minima during Learning with Backpropagation", *Parallel Architectures and Neural Networks*, Vietri sul Mare(IT), May 1990.

[23] K. Gouhara, and Y. Uchikawa, "Memory Surface and Learning Surfaces in Multilayer Neural Networks", Tech. Rep. Naguya University (Japan).

[24] K. Gouhara, N. Kanai, and Y. Uchikawa, "Experimental Learning Surface and Learning Process in Multilayer Neural Networks" Tech. Rep. Naguya University (Japan).

[25] H. P. Graf, L. D. Jackel, and W. E. Hubbard, "VLSI Implementation of a Neural Network Model", *IEEE Computer*, Vol. 21, No. 3, March 1988, pp. 41–49.

[26] R. Hecht-Nielsen, "Theory of Backpropagation Neural Network", *Proceedings of the IEEE-IJCNN89*, Washington D.C., Vol. I, 1989, pp. 593–605.

[27] R. Hecht-Nielsen, "On the algebraic structure of feedforward networks weight space. In R. Eckmiller (Ed.), *Advanced Neural Computers*, Amsterdam, Elsevier Science Publishers B.V., pp. 129–135

[28] G. E. Hinton, "Connectionist Learning Procedures", *Artificial Intelligence*, Vol. 40, 1989, pp. 185–234.

[29] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, Vol. 2, 1989, pp. 359–366.

[30] S.C. Huang and Y.F Huang, "Bounds on the Number of Hidden Neurons in Multi-Layer Perceptrons", *IEEE Transactions on Neural Networks*, Vol. 2, No. 1, January 1991, pp. 47–55.

[31] D. R. Hush, J.M. Sales, and B. Horne, "Error Surfaces for Multi-layer Perceptrons", IEEE-IJCNN91, Seattle 8-12, July 1991, Vol. I, 759–764.

[32] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation", *Neural Networks*, Vol. 1, 1988, pp. 295–307.

[33] F. Jordan, and G. Clement, "Using the Symmetries of Multi-layered Networks to Reduce the Weight Space", Proc. of the IEEE-IJCNN91, Seattle 8-12, July 1991, Vol. II, pp. 391–396.

[34] J.S. Judd, *Neural Network Design and the Complexity of Learning*, The MIT Press, Cambridge, London, 1990.

[35] J.S. Judd ,"Complexity of Connectionist Learning with Various Node Functions", COINS Technical Report 87-60, University of Amherst, Amherst MA.

[36] Y. le Cun, "Learning processes in an asymmetric threshold network," in Fogelman Soulie, E. Bienenstock and G. Weisbuch, Eds., *Disordered Systems and biological organization*, Springer-Verlag, Les Houches, France, 1986, pp. 233–340.

[37] Y. le Cun,"Generalization and Network Design Strategies", in *Proceedings of Connectionism in Perspective*, Elsevier Publishers P. V., North Holland, 1989.

[38] Y. le Cun,"A theoretical framework for Backpropagation", in *Proceedings of the 1988 Connectionist Models Summer School*, D. Touresky, G. Hinton and T. Sejnowski, Eds., San Mateo (CA), Morgan Kauffman, 1988, pp. 21–28.

[39] Y. le Cun, "Improving the Convergence of Backpropagation with Second Order Methods", University of Toronto, Technical Report CRG-TR-88-5.

[40] Y. Lee, S. Oh, and M. W. Kim, "The Effect of Weights on Premature Saturation in Back-Propagation Learning", Proc. of the IEEE-IJCNN91, Seattle 8-12, July 1991, Vol. I, pp. 765–770.

[41] M. L. Minsky and S. A. Papert, *Perceptrons - Expanded Edition*, MIT Press, 1988.

[42] J. M. McInerney, K.G. Haines, S. Biafore, and R. Hecht-Nielsen, "Can Backpropagation error surfaces have non-global minima?", Proc. of the IEEE-IJCNN89, Vol. II, 1989, p. 627.

[43] D.B. Parker, "Learning Logic", Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, April 1985.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing. Exploration in the microstructure of Cognition. Vol. 1: Foundations.*, MIT Press, 1986.

[45] T. Poston, C. Lee, Y. Choie, and Y. Kwon, "Local Minima and Back Propagation", Proc. of the IEEE-IJCNN91, Seattle, July 1991, Vol. II, pp. 173–176.

[46] D.E. Rumelhart, and J.L. McClelland, "Exploration in Parallel Distributed Processing", Vol. 3, MIT Press, 1988.

[47] T. J. Sejnowski and C. R. Rosenberg, "Parallel Networks that learn to pronounce English text," *Complex Systems*, Vol. 1, 1987, pp. 145–168.

[48] E. D. Sontag and H. J. Sussman, "Backpropagation Can Give to Spurious Local Minima Even for Networks without Hidden Layers", *Complex Systems*, Vol. 3, 1989, pp. 91–106.

[49] E. D. Sontag and H. J. Sussman, "Backpropagation separates when perceptrons do", Proc. IEEE-IJCNN89 (Washington DC), Vol. I, 1989, pp. 639–642.

[50] Tesauro G., He Y., "Asymptotic Convergence of Backpropagation" *Neural Computation*, Vol. 1, No. 3, 1989, pp. 382–391.

[51] Tesauro G., "Scaling relationships in backpropagation learning: Dependence on training set size", *Complex Systems*, No. 1, 1987, pp. 367–372.

[52] G. Tesauro and R. Janssens, "Scaling relationships in backpropagation learning: Dependence on predicate order", *Complex Systems*, Vol. 2, 1988, pp. 39–44.

[53] L.G. Valiant, "A Theory of the Learnable", *Communications of the ACM*, Vol. 27, No. 11, 1984, pp. 1134–1142.

[54] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Transactions on ASSP*, Vol. 37, No. 3, 1989, pp. 328–339.

[55] R. L. Watrous, "Learning algorithm for connectionist networks: applied gradient methods for nonlinear optimization", *Proceedings of the first IEEE International Conference on Neural Networks*, S. Diego (CA), 1987, pp. 619–627.

[56] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences", PhD dissertation, Committee on Applied Mathematics, Harvard University, Cambridge, MA, November 1974.

[57] B. Widrow, and M. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation", *Proc. of the IEEE*, Vol. 78, No. 9, pp. 1415–1442.

[58] S. F. Zornetzer, J. L. Davis, and C. Lau, Eds., *An Introduction to Neural and Electronic Networks*, Academic Press, 1990.