On the Problem of Local Minima in Backpropagation

Marco Gori and Alberto Tesi

Abstract—**Supervised learning in multilayered neural networks (MLN's) has been recently proposed through the well-known backpropagation (BP) algorithm. This is a gradient method that can get stuck in local minima, as simple examples can show. In this paper, some conditions on the network architecture and the learning environment, which ensure the convergence of the BP algorithm, are proposed. It is proven in particular that the convergence holds if the classes are linearly separable. In this case, the experience gained in several experiments shows that MLN's exceed perceptrons in generalization to new examples.**

Index Terms—**Backpropagation, learning environment, linearly separable classes, multilayered networks, pattern recognition.**

I. Introduction

SUPERVISED learning in multilayered networks (MLN's) can be accomplished thanks to backpropagation (BP) [20], [26], [33]. Its application to several different subjects [26] and, particularly, to pattern recognition [4], [7], [9], [21], [28], [31] has demonstrated the power of this algorithm. A significant contribution to the understanding of these successful results has been given in [8], [16], [17], and [23]. In particular, in [8], [16], and [17], it has been proved that two-layered MLN's (i.e., only one hidden layer) can produce surfaces for discriminating any nonlinearly separable classes. Although BP is only valid for feedforward networks, it has also been used for learning in synchronous and asynchronous recurrent networks [11], [27].

In spite of these important properties, two major criticisms are commonly raised against BP. The first is that it is computationally expensive. For example, a problem of speech recognition such as learning the plosive set can take many days on ordinary workstations. Several researchers are currently involved in making neural network implementations efficient by using both VLSI silicon and optical technologies [14], [35]. Some significant results have already been obtained, and there are likely to be further improvements in the near future. The second criticism refers to some theoretical considerations. BP is actually a gradient method, and therefore, there is no guarantee at all that the absolute minimum can be reached. In spite of this theoretical remark, researchers involved with BP applications know that this is not a very serious problem. BP often leads to the global minimum, or it at least makes it possible to meet practical stopping criteria.

To the best of our knowledge, the only general analysis concerning the problem of local minima in MLN's has been recently published in [1]. Under the hypothesis that the neurons are linear, it has been proved that only one minimum exists and that the other points in which the gradient is null are saddle points. Although this analysis is interesting, it does not seem easy to generalize it to actual architectures based on nonlinear neurons. In [30], for the particular case of perceptron-like nets (no hidden units), the global convergence of BP has been proved for linearly separable patterns for the case of threshold-penalty LMS cost functions.

This paper proposes a new theoretical framework of BP in order to identify some of its limitations as a general learning procedure and the reasons for its success in several experiments of pattern recognition. The first important conclusion is that in spite of what is sometimes claimed, examples can be found in which BP gets stuck in local minima. We propose a simple example in which BP can get stuck during gradient descent without having learned the entire training set, and we guarantee the existence of a solution with null cost. Other researchers, e.g., [3], [5], [29] have proposed different kind of examples. From a theoretical point of view, these results could lead us to believe that BP is not a very reliable learning procedure. There is no clear evidence for affirming that the gradient descent would be successful (see pp. 257–261 of [24]). However, both our experience with the algorithm in different applications, especially in speech recognition [11], and the experience of several other researchers [4], [7], [9], [21], [28], [31] suggests that there must be some specific reasons for the algorithm's successful application in various problems of pattern recognition.

The main result of this paper is that some conditions on the network architecture and the learning environment have been found, and these allow BP to reach the optimal solution. The conditions are only related to the shape of the error surface and not to any particular learning algorithm. They rely on the concept of delta error [26] and make it possible to establish that as long as they hold, BP converges to the absolute minimum. Our analyses are summarized by two theorems. The first one establishes general conditions on fully connected networks and represents a basical result for more particular analyses. Unfortunately, the theorem's conditions appear to be difficult to apply in practice. The second theorem is based on more restrictive hypotheses, but it proposes a very straightforward and practical result. Basically, it states that when applied to linearly separable patterns, BP does not get stuck in local minima. This is a classical condition that was also adopted by Rosenblatt [25] to prove the convergence of the perceptron learning algorithm. Although in this case a perceptron would separate the entire training set, the generalization on new patterns obtained with feedforward nets is significantly better [12], [21]. Moreover, problems that involve linearly separable

Manuscript received March 20, 1990; revised June 11, 1991.

This research was supported by MURST and by CNR Grant 90.01530.CT01. Recommended for acceptance by R. De Mori.

The authors are with the Dipartimento di Sistemi e Informatica, Universitá di Firenze, Firenze, Italy. IEEE Log Number 9104528.

patterns are also of practical interest, and feedforward nets have often been applied in this case.

This paper is organized as follows. In the next section, we propose a brief review of BP, whose main purpose is to define the concepts and symbols involved in our theoretical framework. In Section III, we illustrate the basic results of our paper, which are proved in the Appendix, whereas in Section IV, we analyze some examples of local minima. We discuss our results in Section V and draw some final conclusions in Section VI.

II. Learning in Multilayered Networks with BP

In this section, we define the formalism adopted throughout the paper and give a brief review of BP. Some details on the derivation of the algorithm and other general considerations can be found in several publications [16], [20], [26], [33], [34]. Here, we propose a vectorial formulation of the basic equations, which is particularly useful for our analysis of the convergence of BP.

Basically, the problem of learning in MLN's is to find a set of weights that 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 [26].

- *Network* $\mathcal{N}$. It has a multilayered architecture (see Fig. 1). This means that all neurons are grouped in sets called layers.[1] With reference to index l, we distinguish among the input layer ($l = 0$), the output layer ($l = L$), and hidden layers ($0 < l < L$). The number of neurons per layer is denoted by $n(l)$. Each neuron of layer l is referred to by its index $i(l)$, $i(l) = 1, \ldots, n(l)$.
  When pattern "t" is presented at the input, for each neuron, we consider

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

  The activation is computed by propagating forward the outputs of the neurons of the previous level as follows:[2]

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

  where $w_{i(l),j(l-1)}$ denotes the weight of the link between the neurons $i(l), j(l-1)$, 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 $f(\cdot) : R \to [\underline{d}, \bar{d}]$ is a C^2 function with a positive first derivative. This function must satisfy the following conditions, which are useful in our analysis on local minima:

$$\begin{aligned} \lim_{a\to\infty} \frac{f'(a)}{\bar{d} - f(a)} &= \bar{k}_1 \qquad \lim_{a\to\infty} \frac{-f''(a)}{\bar{d} - f(a)} = \bar{k}_2 \\ \lim_{a\to-\infty} \frac{f'(a)}{f(a) - \underline{d}} &= \underline{k}_1 \qquad \lim_{a\to-\infty} \frac{f''(a)}{f(a) - \underline{d}} = \underline{k}_2 \end{aligned} \tag{4}$$

  where $\underline{k}_1, \bar{k}_1, \underline{k}_2, \bar{k}_2$ are real positive constants, and $f'(\cdot), f''(\cdot)$ denote the first and second derivatives, respectively. For example, a squash function [26] satisfies these hypotheses.

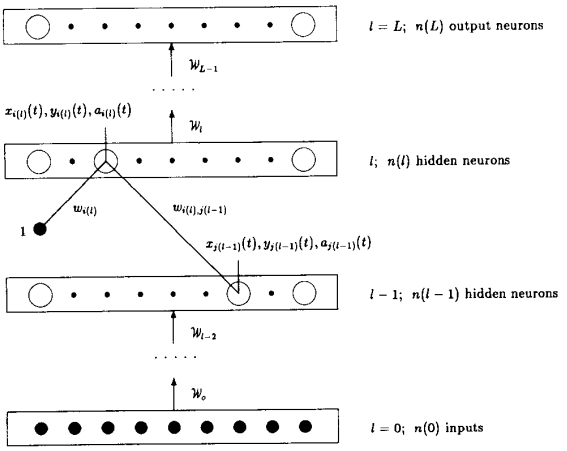


Fig. 1. Network architecture and symbol definitions.

- *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 $\mathcal{N}$:

$$\mathcal{L}_e \doteq \Big\{ ((X_o(t), D(t)), X_o(t) \in R^{n(o)}, D(t) \in R^{n(L)}, t = 1, \ldots, T \Big\}.$$

  $X_o(t)$ is the input pattern, and $D(t)$ is its corresponding target vector. Each component of $D(t)$ belongs to the interval $[\underline{d}, \bar{d}]$. All the targets are collected in the matrix $\mathcal{D} \doteq [D(1), \ldots, D(T)]' \in R^{T,n(L)}$.

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

$$E_T \doteq \frac{1}{2} \sum_{t=1}^{T} E_t = \frac{1}{2} \sum_{t=1}^{T} \sum_{j=1}^{n(L)} [d_j(t) - x_{j(L)}(t)]^2. \tag{5}$$

Learning $\mathcal{L}_e$ with the network $\mathcal{N}$ means finding out a set of weights that minimize the cost index E_T. Several numerical methods have been proposed to solve this problem. First- and second-order methods can be distinguished. Basically, first-order methods rely on gradient descent. Weights are computed by the well-known gradient descent scheme[3]

$$w_{ij}^{(r+1)} = w_{ij}^{(r)} - \epsilon \frac{\partial E_T}{\partial w_{ij}} \tag{6}$$

[1] BP can also be derived for networks in which only an order relationship among neurons is assumed [10]. However, this more general class of nets does not seem to be of any practical interest.

[2] In the sequel, both layer and pattern indices may be omitted for simplicity's sake.

[3] The same relationship holds for the thresholds, which can be considered weights connected to inputs clamped at "1" (see Fig. 1).

where "r" denotes the index of the iteration process. Numerous variants have recently been proposed in the literature. Weight updating has been suggested after each pattern instead of accumulating the gradient for the whole learning environment [26]. This updating scheme is referred to as *pattern mode*, whereas the classical gradient descent scheme (6) is defined as *batch mode*. Some researchers have also investigated weight updating done after a subset of the whole learning environment has been presented. Normally, these subsets contain at least one pattern per class. Throughout this paper, we refer to this weight updating scheme as *block mode*. Much attention has been paid to the adaptive selection of the coefficient ϵ (learning rate) [19]. Another commonly used variant is that of adding a term in (6) (momentum term), which produces a low-pass filtering effect in order to reduce abrupt gradient changes [26]. Second-order methods are based on hessian matrix approximation. Successful results have been obtained with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [32] and the diagonal approximation of the hessian matrix. Most of the experiments have been carried out by using first-order methods based on a very quick algorithm, which is referred to as BP, that computes the gradient $\partial E_t/\partial w_{ij}$ for all the weights. This method has been proposed in different forms by several researchers. Its origins have been traced to the field of the optimal control theory (see [6]) by le Cun [22]. However, the actual scientific diffusion of BP is primarily due to the parallel processing distributed (PDP) group.

Here, we briefly review the algorithm for gradient computation by proposing a vectorial formulation that turns out to be very useful in our analysis on local minima. The following is a definition of the concepts and symbols involved:

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

$$\mathcal{X}_l \doteq [X_{1(l)} \cdots X_{n(l)}\ \Pi] \in R^{T,n(l)+1},\ 0 \le l \le L \quad (7)$$

where $\Pi \doteq [1 \cdots 1]' \in R^T$ is used to deal with biases.
3) $w_{i(l),j(l-1)}$ is the weight that connects neuron $j(l-1)$ with neuron $i(l)$ (see Fig. 1). The associated matrix $\mathcal{W}_{l-1} \in R^{n(l),n(l-1)}$ is referred to as the *weight layer matrix*. The symbol Ω denotes the weight space.
4) We assume $y_{i(l)}(t) \doteq \partial E_t/\partial a_{i(l)}(t)$, the vector $Y_{i(l)} \doteq [y_{i(l)}(1) \cdots y_{i(l)}(T)]'$ is called the *delta trace*, and the *delta layer trace* is the matrix $\mathcal{Y}_l \doteq [Y_{1(l)} \cdots Y_{n(l)}] \in R^{T,n(l)}$. We denote by $\mathcal{S}_l^Y \subset R^{T,n(l)}$ the set of all the $\mathcal{Y}_l$ generated by varying the weights in Ω.
5) Let $\alpha(\cdot) : \Omega \to R^+$ be a generic function. Let us assume that $\forall \epsilon > 0$, there is a region contained in Ω such that, for all its points, the condition $\alpha < \epsilon$ holds. This is concisely denoted as $\alpha \to 0$ or as α is "zero."
6) All the patterns of the learning environment $\mathcal{L}_e$ are divided into C classes, where each is composed of T_c patterns ($c = 1, \ldots, C$). The patterns belonging to the same class have the same target.
7) We assume that there is no connection that jumps a layer. Therefore, 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 $g(j(l-1), i(l))$ is given by $\partial E_T/\partial w_{i(l),j(l-1)}$ if $j(l-1) \le n(l-1)$ and $\partial E_T/\partial w_{i(l)}$ if $j(l-1) = n(l-1)+1$ (bias term gradient contribution).

With these definitions, the gradient matrix can be computed as follows

$$\mathcal{G}_{l-1} = \mathcal{X}'_{l-1}\mathcal{Y}_l, \quad (8)$$

where $\mathcal{X}'_{l-1}$ is the transpose of $\mathcal{X}_{l-1}$. 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 ((2),(3), *forward step*). Matrix $\mathcal{Y}_l$, $l = 1, \ldots, L$ can be computed by the following equations:

$$\begin{cases} \tilde{\mathcal{Y}}_L = \mathcal{X}_L - \mathcal{D} & \text{(9a)} \\ \tilde{\mathcal{Y}}_l = \mathcal{Y}_{l+1}\mathcal{W}_l \quad l = L-1, \ldots, 1 & \text{(9b)} \end{cases}$$

where the generic element of $\tilde{\mathcal{Y}}_l$ is $\tilde{y}_{i(l)}(t) \doteq y_{i(l)}(t)/f'(a_{i(l)}(t))$. Therefore, from (9a) we directly compute $\mathcal{Y}_L$ and then go backward to compute $\mathcal{Y}_l$ until $l = 1$ ((9b), *backward step*). The gradient computation for all the M weights of the network and for all the T patterns takes $O(M \cdot T)$ time, versus $O(M^2 \cdot T)$ of a direct gradient computation based on gradient's definition. As a matter of fact, this dramatic reduction of complexity makes it possible to explore the application to many interesting practical problems.

In the last few years, several researchers have applied gradient algorithms based on BP (with numerous variants) to the solution of pattern recognition, communication, and control problems. As already mentioned, the diffusion in the scientific community of BP is primarly due to PDP's research group. They consider this procedure as a challenge to the problems raised in the famous book *Perceptrons* [24]. They state that BP provides a direct generalization of the perceptron learning procedure and can be applied to arbitrary networks with multiple layers and feedback among layers. This claim has been discussed by several researchers in order to find out what practical conditions are necessary for learning arbitrary functions. It has been pointed out that practical problems can still arise for learning predicates having an infinite order, such as connectedness or other topological predicates. Finite order predicates (e.g., symmetry) can also create practical problems because of the exponential explosion of the weights. However, the most significant criticism is that BP is a hill-climbing method that cannot be considered to be reliable in practice because the gradient can get stuck in local minima. The authors of BP do not provide any theoretical answer to this, but they claim that it is reliable in practice (see p. 361 of [26]).

In the following section, we propose a formal analysis of the problem of local minima that supports PDP's claims under certain conditions (PR hypotheses) inspired by pattern recognition problems. In Section IV, we discuss some examples of suboptimal learning that show the limits of BP for general mapping problems.

III. Conditions Ensuring Optimal Learning

In this section, we begin by analyzing the problem of minima of the cost function (5). As already mentioned in the introduction, we are mainly interested in analyzing the surface attached to E_T. Knowledge on this surface, particularly on stationary points, gives straightforward information on the behavior of gradient algorithms like batch mode BP. In the following, we say that BP converges to the optimal solution, provided that no local minima exist for the cost function. The knowledge on the surface attached to E_T is useful for whatever learning algorithm is used. For example, it also gives significant insights to pattern and block weight updating modes.

All our considerations rely on the hypothesis that the entire training set can be learned exactly. We also assume that $\underline{d}$ and $\bar{d}$ are the target values. This hypothesis implies the existence of only asymptotic optimal solutions. Using the formalism introduced in Section II for the vectorial formulation of BP (see point 5), we concisely state these assumptions as follows:

- There exists at least a set of weights for which $E_T \rightarrow 0$.

This condition has been implicitly assumed in many pattern recognition applications since it is quite normal to use "0% error rate" as a stopping criterion (see e.g., [7], [21]). Moreover, as some recent results [8], [16], [17] have shown, networks with just one hidden layer are universal interpolators in that they can map "arbitrary functions," provided that a sufficient number of neurons is chosen in the hidden layer.

Our analysis focuses on stationary points. We investigate what happens when the gradient of E_T is "zero," in order to discover what configurations cause learning algorithms to get stuck. All our conclusions are carried out by considering $y_{i(l)}(t)$ for each neuron $i(l)$ and each pattern t. We provide some conditions for ensuring that whenever the gradient gets stuck, delta layer matrices $\mathcal{Y}_l$, $l = 1, \ldots, L$ satisfy $\mathcal{Y}_l \rightarrow 0$. Then, we prove (see Lemma 3) that all the configurations $\mathcal{Y}_L \rightarrow 0$ are either saddle points or the absolute minimum ($E_T \rightarrow 0$). As a result, any condition guaranteeing that $\mathcal{Y}_L \rightarrow 0$ also guarantees the convergence to the optimal solution ($E_T \rightarrow 0$). The following theorem gives a first general answer to the problem of local minima.

Theorem 1: Gradient descent leads to the absolute minimum ($E_T \rightarrow 0$) if the MLN and its associated learning environment satisfy the following PR1 hypotheses:

1) $n(l+1) \leq n(l), \quad l = 1, \ldots, L-1$ (pyramidal hypothesis).
2) The weight layer matrices W_l, $l = 1, \cdots, L-1$ are full row rank matrices.
3) The following relationship between the kernel of matrix $\mathcal{X}_o'$ and set $\mathcal{S}_1^Y$ holds

$$Ker[\mathcal{X}_o'] \cap \mathcal{S}_1^Y = \{\varnothing\}. \tag{10}$$

Proof: See the Appendix. □

In the hypotheses of this theorem, batch mode BP cannot get stuck in local minima. Once the gradient method converges, it does reach the absolute minimum with "null cost." At this point, it is worth making a few remarks concerning the hypotheses of the theorem. First, we notice that the pyramidal assumption does not involve the input layer ($l = 0$). This hypothesis appears as a natural consequence of the task accomplished by the neurons in the network since if a hidden layer gets closer to the output, the information is more compressed. Moreover, this structure is often adopted in many experiments [4], [7], [9], [11], [21], [28], [31]. Second, the hypothesis concerning the rank of the weight layer matrices $\mathcal{W}_l$ is quite reasonable, as has also been pointed out in [1]. The PR1.3 hypothesis involves both network and learning environment conditions. Unfortunately, it is quite hard to understand its practical meaning since it requires the knowledge of $\mathcal{S}_1^Y$, i.e., the set of all the $\mathcal{Y}_1$ generated by varying the weights in Ω. As a particular case, the PR1.3 hypothesis holds when all patterns are linearly independent since, in this case, $Ker[\mathcal{X}_o] = \{\varnothing\}$. This is formally stated by the following corollary.

Corollary 1: The gradient descent leads to the absolute minimum ($E_T \rightarrow 0$) if the following conditions hold:

1) The PR1.1 and PR1.2 network hypotheses hold.
2) $Rank[\mathcal{X}_o] = T$ (linearly independent patterns). □

It is worth mentioning that in this case, the PR1.3 hypothesis only involves the learning environment. This makes hypothesis verification very straightforward. On the other hand, when the patterns are linearly independent, the number of input neurons $n(0)$ cannot be smaller than T. This is a very serious practical restriction because the number of input neurons limits the cardinality of the learning environment.

In order to discover meaningful conditions with a straightforward, practical interpretation, we propose modeling the data to be learned. A classical data model that also fits practical situations in pattern recognition assumes that the patterns are linearly separable. With a few more assumptions concerning network architecture, we can prove that the result stated in Theorem 1 still holds by assuming this data model. This is formally stated by the following theorem.

Theorem 2: The gradient descent leads to the absolute minimum ($E_T \rightarrow 0$) if the network and the learning environment satisfy the following PR2 hypotheses:

- Network:
  1. The network has only one hidden layer ($L = 2$).
  2. The network has C outputs, where C is the number of classes.
  3. Full connections are assumed from the input to the hidden layer. The hidden layer is divided into C sublayers $H_1, \ldots, H_c, \ldots, H_C$, and connections are only permitted from any sublayer to the associated output (see Fig. 2). The sublayer H_c contains $n_c(1)$ neurons referred to by the index $i_c(1)$.
- Output Coding: Exclusive coding is used for the output, i.e., if pattern t belongs to class c, $(c = 1, \ldots, C)$; then

$$D(t) = [\underline{d}, \ldots, \bar{d}, \ldots, \underline{d}]'.$$

- Learning Environment: All the patterns of $\mathcal{L}_e$ are linearly separable, i.e., for each class c, a vector $A_c \in R^{n(0)+1}$ exists such that

$$A_c' X_o^e(t) > 0 \text{ for all } t \text{ in class } c$$

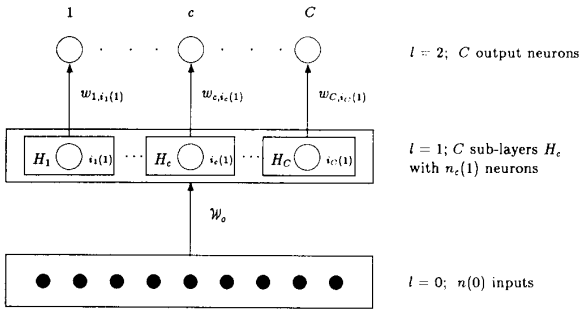


Fig. 2. Network architecture according to PR2 hypotheses.

$$\text{and } A_c' X_o^e(\tau) < 0 \quad \forall \tau \text{ not in class } c \tag{11}$$

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

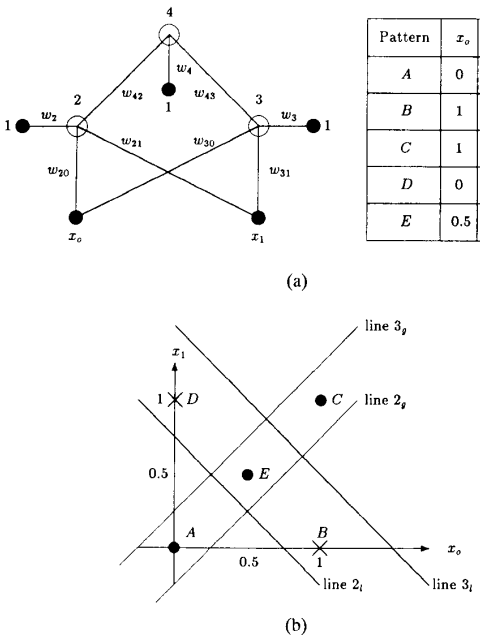
Proof: See the Appendix. □

Some remarks concerning this result are worth mentioning. First of all, compared with Theorem 1, more hypotheses are assumed on the network architecture. Only one hidden layer is taken into consideration, and some additional constraints are put on the network connectivity (hyp. 3). This kind of architecture is also adopted for proving the interpolation capabilities of MLN's in [8], [16], [17]. Second, the hypothesis of linearly separable patterns suggests that a comparison with Rosenblatt's perceptron [25] be made. It is well known that this hypothesis is also sufficient for guaranteeing the convergence of the δ rule [25] to configurations where all the patterns are correctly classified. As discussed in Section V, this must not lead us to conclude that in experiments of patterns recognition with linearly separable patterns, perceptrons, and MLN's are equivalent. MLN's exceed perceptrons, in general, on test sets [12], [21].

IV. Some Examples of Local Minima

Recent research has shown that the convergence of BP can sometimes be very critical. In some regions of the weight space, this may depend on the flatness of the cost surface where the algorithm is obviously slow. Moreover, it has been proved that BP can get stuck in local minima. When using batch weight updating, this means that a suboptimal solution can be found.

Brady *et al.* [5] have proposed several examples of local minima concerning linearly separable patterns. These minima even hold for one-layered networks. However, a quick glance at them makes it clear that these examples only hold with target values that are different from the asymptotic values $\underline{d}, \bar{d}$. If we adopt asymptotic target values, as assumed in our theorems, this kind of minima no longer holds. In addition, this kind of counterexample seems to get increasingly artificial as the target values approximate $\underline{d}, \bar{d}$. In [5], the authors conclude that BP is unable to separate where *Perceptrons* succeeds in doing so. Nevertheless, as we have proved, the opposite conclusion can be reached if we take $\underline{d}$ and $\bar{d}$ as target values or if one uses threshold-penalty LMS cost functions [30]. Sontag and Sussman [29] propose an example of local minimum in single-layered networks that is different from the ones proposed by Brady *et al.* in that asymptotic values of the squash function are used as target values. Sontag and Sussman consider a nonlinearly separable set of patterns that cannot be learned completely by the network used. Recently, Blum [3] has shown that sigmoidal networks for the XOR boolean function present a manifold of local minima. These local minima are obtained when the weights between the input layer and the hidden layer all are null. We propose a new kind of example where, although a nonlinearly separable set of patterns could be learned exactly by the network chosen, BP can get stuck in local minima. In this example, all of the hypotheses in Theorem 2 are satisfied, except the condition of linearly separable patterns.



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

Fig. 3. (a) Network and learning environment of the example; (b) separation lines of the local and global minima configurations.

Example

We consider the learning environment and the network reported in Fig. 3(a). Compared with the XOR predicate, we only need to learn the additional point "E." Depending on the initial weights, the gradient can get stuck in points where the cost is far from being null. Experimental evidence of this fact is given in Fig. 4, where both the cost and the gradient are reported as a function of the variable r that parameterizes the line that connects, in the weight space, the absolute minimum ($r = 0$) to a local minimum ($r = 5$). 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. 3(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

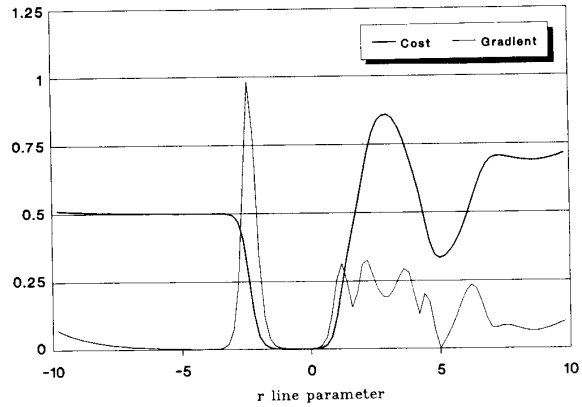


Fig. 4. Cost and gradient along a line connecting the global and the local minima.

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 [10], 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 be reached. With reference to the simplified notations of Fig. 3(a), the global contribution of the patterns to $\partial E_T/\partial w_{20}$ and $\partial E_T/\partial w_{21}$ is

$$\begin{aligned}\frac{\partial E_T}{\partial w_{20}} &= y_2(B)+y_2(C)+\frac{1}{2}y_2(E)\\ \frac{\partial E_T}{\partial w_{21}} &= y_2(C)+y_2(D)+\frac{1}{2}y_2(E).\end{aligned} \tag{12}$$

The BP rule makes it possible to compute y_2 as

$$y_2 = f'(a_2)w_{42}y_4. \tag{13}$$

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 (13) implies $y_2(B)=y_2(D)$, where $a_2(B)=a_2(D)$ as well. From (12), it follows that $(\partial E_T/\partial w_{20}, \partial E_T/\partial w_{21})$ has a direction that 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 the BP rule, it follows that the gradient components $\partial E_T/\partial w_{42}$ and $\partial E_T/\partial w_{43}$ are null if

$$(x_4(E)-\underline{d})+(x_4(D)-\bar{d})+(x_4(B)-\bar{d})=0. \tag{14}$$

TABLE I
REPORT OF SOME PATTERN RECOGNITION APPLICATIONS WITH MLN'S

Publication	Application	Architecture
Bourlard-Wellekens [4]	Phoneme Recognition	1118-200-50
Cosi-Bengio-De Mori [7]	Phoneme Recognition	400-20-10
Elman-Zipser [9]	Phoneme Recognition	320-6-3
le Cun [21]	Handwritten Digit Recognition	256-12-10
Sejnowsky [28]	Text to Speech	203-80-26

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

Some other examples of local minima can be found in [13].

V. Discussion

Although the analysis carried out in Section III refers to the classical quadratic cost function, it can be extended to other cost functions as well. Let us assume

$$E_T=\sum_{t=1}^{T}\sum_{j=1}^{n(L)} l(x_j(t)-d_j(t)) \tag{15}$$

where $l(\cdot)$ is a generic nonnegative C^2 function, such that

$$l(\alpha)=0, \qquad l'(\alpha)=0 \;\Leftrightarrow\; \alpha=0 \tag{16}$$

where $'$ stands for differentiation. It can be shown that all the results reported in Section III still hold when using a function like (15) [12]. The use of cost functions that differ from the quadratic one has already been investigated (see e.g., [15], where Minkowsky-r cost functions are used). They match conditions (16) except for the case where $r=1$.

We are confident that Theorem 2 explains the reasons of the BP's success in several pattern recognition experiments. Many successful experiments reported in the literature probably deal with quasi-linearly separable patterns. We have experimental evidence to state that the presence of a slight class overlapping, which is typical of many experiments of pattern recognition, does not significantly affect the convergence of BP [12]. A quite 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 that are themselves sufficient to discriminate all the patterns. Table I shows a list of recently published papers that 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 [21], where it is explicitely stated that the patterns are linearly separable in experiments of handwritten digit recognition with a 16×16 input grid. In these experiments, the entire training set was succesfully 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 preprocessing. We found that the dimensions of the input grid play a significant role

since the preprocessed patterns turn out to be linearly separable when "high" resolution is used. For example, 600 handwritten characters, equally distributed in classes A, B, and C, were found to be linearly separable by using a resolution as low as $6 * 8$ pixels. Although single-layered nets also separated the training set, we found that MLN's perform significantly better in generalization to new examples. In the above mentioned experiment, we found that adding a hidden layer of six neurons to a net having 48 $(6 * 8)$ inputs and three outputs (A, B, and C with exclusive coding) allows us to decrease the recognition error from 3.21% (no hidden layer) to 2.53% (with hidden layer) [12]. The same conclusions are reported for handwritten digit recognition in [21] by le Cun, who also noticed a significant reduction of the overtraining phenomenon in the presence of hidden layers.

Unfortunately, in other practical cases, the patterns are not linearly separable. In these situations, one can apply Theorem 1, even if the PR1.3 hypothesis is quite difficult to verify. In order to tackle these cases, 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 light of our results, it comes out that many recent variations to the BP scheme may turn out to be particularly useful for difficult nonlinearly separable patterns. Many researchers have found it useful to learn huge learning environments block by block. For example, in experiments of speaker independent speech recognition, it is common practice to begin learning a few speakers and then increase the training set when a stop criterion has been met (see e.g., [31]). 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, whereas more successful results were obtained with the technique of gradually increasing the training set.

As a final remark, we point out that our analysis is independent of the learning algorithm used. As a particular case, our results give a straightforward implication on the convergence of batch mode BP. On the other hand, a weight updating modality, such as pattern or block mode, departs to some extent from a pure gradient descent. This departure depends on both the learning rate and the number of patterns of the learning environment. As pointed out in [26], with a "small enough" learning rate, these weight updating modalities lead to the same result as a true gradient descent. In some cases, this departure cannot be ignored, and the analysis proposed in this paper no longer holds.

VI. Conclusions

In this paper, we analyze the problem of learning in MLN's and develop some conditions that guarantee the absence of local minima in BP. The main result is that for linearly separable patterns, the convergence to optimal solutions is ensured by using batch mode BP. Moreover, in these cases, BP exceeds *Perceptron* in generalization to new examples. This result explains the success of BP in several problems of speech and image recognition.

When the patterns are not linearly separable, the situation is more complicated because of the possible presence of local minima, as shown by a simple example. Although the proposed analysis cannot be easily applied to nonlinearly separable problems, we are confident that it opens the door for more detailed investigations on this problem.

Appendix

Some preliminary lemmas are useful in order to prove Theorems 1 and 2.

On the hypothesis that on the input layer $(l = 0)$, $Ker[\mathcal{X}_o']$ is disjont from the space $\mathcal{S}_1^Y$, in the first lemma, it is proved that if the gradient for this layer is arbitrarily close to zero ("zero") on a certain region of the weight space, then the delta layer matrix $\mathcal{Y}_1 \to 0$. When the PR1 hypotheses holds, in the second lemma, it is proved that if the gradient for the first layer is "zero" on a compact region Ω of the weight space, then the same is true for $\mathcal{Y}_L$. Moreover, it is shown that this property still holds even if the compactness hypothesis of the set Ω is removed. Said another way, it turns out that the "zero" of $\mathcal{Y}_L$ implies the "zero" of the gradient on each layer, i.e., the gradient descent only gets stuck at the points for which $\mathcal{Y}_L$ is "zero." This condition holds for perfect matching $(E_T \to 0)$ as well as for mismatching configurations in which there are outputs that saturate to a wrong target for some patterns. In Lemma 3, it is shown that among all the points for which $\mathcal{Y}_L$ is "zero," only those whose cost is also "zero" correspond to attractive points for the gradient descent. Said another way, the points for which $\mathcal{Y}_L$ is "zero" and the cost E_T is finite do not represent local minima. As a result, Theorem 1 is a straightforward consequence of Lemmas 2 and 3.

In order to prove Theorem 2, we begin by exploiting the PR2 hypotheses concerning network architecture and output coding. In Lemma 4, we study $\mathcal{Y}_1$'s sign. Consequently, in Lemma 5, we prove that for any stationary point, it is $\mathcal{Y}_1 \to 0$ under the assumption of linearly separable patterns. In Lemma 6, we prove that the stationary points, for which the weights connecting the neurons of at least a hidden sublayer with the corresponding output all are null, are not local minima. As a result, Theorem 2 follows directly from Lemma 6 and Theorem 1.

Lemma 1: Let us consider the input layer $(l = 0)$ and assume that $Ker[\mathcal{X}_o'] \cap \mathcal{S}_1^Y = \{\varnothing\}$. Then, for an arbitrary real constant $\epsilon > 0$, there exists $\delta = \delta(\epsilon) > 0$ such that

$$\forall i(1), j(0) \; |\frac{\partial E_T}{\partial w_{i(1),j(0)}}| < \delta \to \|\mathcal{Y}_1\|_\infty < \epsilon. \tag{17}$$

Proof: According to the BP rule, the gradient can be written as

$$\mathcal{G}_o = \mathcal{X}_o' \mathcal{Y}_1. \tag{18}$$

We observe that $\mathcal{Y}_1 \in \mathcal{S}_1^Y$ only if the linear system (18) is solvable, i.e.

$$Rank[\mathcal{X}_o'] = Rank[\mathcal{X}_o' | \mathcal{G}_o]. \tag{19}$$

From condition $Ker[\mathcal{X}_o'] \cap \mathcal{S}_1^Y = \{\varnothing\}$, it follows that $\mathcal{Y}_1 = 0$ is the unique solution of (18) when $\mathcal{G}_o = 0$. Hence, the proof

of the lemma is a direct consequence of condition (19) and of the continuity of linear system's solutions. □

Lemma 2: Let us assume that the PR1 hypotheses hold, and let $\epsilon > 0$ be an arbitrary positive real number. Then, for a compact region Ω of the weight space, there exists $\delta = \delta(\epsilon) > 0$ such that

$$\forall i(1), j(0) \; |\frac{\partial E_T}{\partial w_{i(1),j(0)}}| < \delta \rightarrow \|\mathcal{Y}_L\|_\infty < \epsilon. \qquad (20)$$

Proof: The compactness hypothesis implies that there exists $d \in R$ such that $0 < d \leq f'(a_{i(l)}(t))$ in Ω. According to the BP backward step, we have

$$\tilde{\mathcal{Y}}_l = \mathcal{Y}_{l+1}\mathcal{W}_l \qquad l = 1, \cdots, L-1 \qquad (21)$$

where $\tilde{\mathcal{Y}}_l \in R^{T,n(l)}$ and $\tilde{y}_{t,i(l)} \doteq y_{i(l)}(t)/f'(a_{i(l)}(t))$. From the PR1.1 and PR1.2 hypotheses, it follows that

$$\mathcal{Y}_{l+1} = \tilde{\mathcal{Y}}_l\mathcal{W}_l'[\mathcal{W}_l\mathcal{W}_l']^{-1} \qquad l = 1, \cdots, L-1. \qquad (22)$$

We then have

$$\begin{aligned}\|\mathcal{Y}_{l+1}\|_\infty &\leq \|\tilde{\mathcal{Y}}_l\|_\infty \cdot \|\mathcal{W}_l'[\mathcal{W}_l\mathcal{W}_l']^{-1}\|_\infty \\ &\leq R_l\|\mathcal{Y}_l\|_\infty \qquad l = 1, \cdots, L-1 \qquad (23)\end{aligned}$$

where $R_l \doteq \|\mathcal{W}_l'[\mathcal{W}_l\mathcal{W}_l']^{-1}\|_\infty/d$. If we choose $\delta_1(\epsilon)$ satisfying Lemma 1, then the proof of the lemma easily follows by selecting

$$\delta(\epsilon) = \frac{\delta_1(\epsilon)}{R_1 \cdots R_{L-1}}. \qquad (24)$$

□

Remark 1: Suppose that for the learning environment $\mathcal{L}_e$, the gradient gets arbitrarily close to zero only asymptotically in the weight space (for example, this happens for the symmetric squash function with target values $+1, -1$). We can show that the conclusions of Lemma 2 still hold. Let $\left\{\mathcal{L}_e^{(n)},\ n = 1, \ldots\right\}$ be a sequence of learning environments defined by the same inputs and by targets that are arbitrarily close to those of $\mathcal{L}_e$ when $n \rightarrow \infty$. Lemma 2 can be applied for each $\mathcal{L}_e^{(n)}$ with reference to a suitable compact set $\Omega^{(n)}$. Because of the cost and gradient regularity, it can be shown [10] that Lemma 2 is also true for $\mathcal{L}_e$.

Lemma 3: Let us consider a configuration for which $\mathcal{Y}_L \rightarrow 0$ 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 $f(a_{i(L)}(\hat{t}))$ for the output neuron $i(L)$, which has no bias (i.e., $w_{i(L)} = 0$). Consequently, at least a neuron $j(L-1)$ exists such that

$$\frac{\partial^2 E_T}{\partial w_{i(L),j(L-1)}^2} < 0. \qquad (25)$$

Proof: The BP relationship

$$y_{i(L)}(t) = f'(a_{i(L)}(t))(f(a_{i(L)}(t)) - d_{i(L)}(t)) \qquad (26)$$

states that for pattern $\hat{t}$, $\mathcal{Y}_L \rightarrow 0$ implies that $f'(a_{i(L)}(\hat{t})) \rightarrow 0$. The contribution of pattern $\hat{t}$ to (25) can be written as follows:

$$\frac{\partial^2 E_{\hat{t}}}{\partial w_{i(L),j(L-1)}^2} = f^2(a_{j(L-1)}(\hat{t}))\frac{\partial^2 E_{\hat{t}}}{\partial a_{i(L)}^2(\hat{t})} \qquad \forall j(L-1) \qquad (27)$$

where the term $\frac{\partial^2 E_{\hat{t}}}{\partial a_{i(L)}^2(\hat{t})}$ can be computed directly, as follows:

$$\frac{\partial^2 E_{\hat{t}}}{\partial a_{i(L)}^2(\hat{t})} = f''(a_{i(L)}(\hat{t}))(f(a_{i(L)}(\hat{t}) - d_i(\hat{t})) + f'^2(a_{i(L)}(\hat{t})). \qquad (28)$$

From (4), it is easy to verify that the following relationship holds:

$$\frac{\partial^2 E_{\hat{t}}}{\partial a_{i(L)}^2(\hat{t})} = \begin{cases} \rightarrow & -\frac{\bar{k}_2}{\bar{k}_1}(\bar{d} - \underline{d})f'(a_{i(L)}(\hat{t})) < 0 \quad \text{if } d_{i(L)}(\hat{t}) = \bar{d} \\ \rightarrow & -\frac{\underline{k}_2}{\underline{k}_1}(\bar{d} - \underline{d})f'(a_{i(L)}(\hat{t})) < 0 \quad \text{if } d_{i(L)}(\hat{t}) = \underline{d}. \end{cases} \qquad (29)$$

Moreover, a similar analysis can be carried out for the other patterns

$$\frac{\partial^2 E_t}{\partial a_{i(L)}^2(t)} = \begin{cases} \rightarrow & \left(1 + \frac{\bar{k}_2}{\bar{k}_1^2}\right)f'^2(a_{i(L)}(t)) > 0 \quad \text{if } d_{i(L)}(t) = \bar{d} \\ \rightarrow & \left(1 + \frac{\underline{k}_2}{\underline{k}_1^2}\right)f'^2(a_{i(L)}(t)) > 0 \quad \text{if } d_{i(L)}(t) = \underline{d}. \end{cases} \qquad (30)$$

It can be easily checked that we cannot have $f(a_{j(L-1)}(\hat{t})) = 0$, $\forall j(L-1)$ because the activations of all the output units would be null, and this is not possible since $\mathcal{Y}_L \rightarrow 0$. As a result, at least a neuron $j(L-1)$ exists such that

$$\begin{aligned}\frac{\partial^2 E_T}{\partial w_{i(L),j(L-1)}^2} &= \sum_{\sigma=1}^{T}\frac{\partial^2 E_\sigma}{\partial w_{i(L),j(L-1)}^2} \\ &= \sum_{\sigma=1}^{T} f^2(a_{j(L-1)}(\sigma))\frac{\partial^2 E_\sigma}{\partial a_{i(L)}^2(\sigma)} \qquad (31)\end{aligned}$$

is negative since from (29) and (30), it follows that $\frac{\partial^2 E_t}{\partial w_{i(L),j(L-1)}^2}$ has order higher than $\frac{\partial^2 E_{\hat{t}}}{\partial w_{i(L),j(L-1)}^2}$, and therefore, the conclusion of the lemma easily follows. □

Remark 2: We can remove the hypothesis that neuron $i(L)$ has no bias ($w_{i(L)} = 0$) simply by observing that

$$\frac{\partial^2 E_{\hat{t}}}{\partial w_{i(L)}^2} = \frac{\partial^2 E_{\hat{t}}}{\partial a_{i(L)}^2(\hat{t})}.$$

Therefore, the lemma follows from (30) in this case, as well.

Remark 3: Lemma 3 implies that "$\mathcal{Y}_L = 0$" mismatching configurations cannot represent local minima. It is quite obvious that the result of the previous lemma can be readily extended to different and more complicated mismatch configurations. Apart from its use in the proofs for Theorems 1 and 2, Lemma 3 gives itself an interesting explanation of the fact that BP is able to escape when the output neurons are saturated to erroneous values. For example, this happens when the initial weights of the algorithm are too high. Even though the algorithm gets very slow, it is still able to escape from these configurations.

Proof of Theorem 1: If the PR1 hypotheses hold, then Lemma 2 indicates that the points for which the gradient is "zero" are the same for which $\mathcal{Y}_L$ is "zero." From Lemma 3, it follows that only the points whose cost is "zero" represent attractive points for the gradient descent. □

Lemma 4: Let us consider a network whose architecture matches the PR2 hyphoteses. Partitioning matrix $\mathcal{Y}_1$ as

$$\mathcal{Y}_1 = [\mathcal{Y}_1^1|\mathcal{Y}_1^2|\cdots|\mathcal{Y}_1^c|\cdots|\mathcal{Y}_1^C] \in R^{T,n(1)} \tag{32}$$

where $\mathcal{Y}_1^c \in R^{T,n_c(1)}$, then its generic column $Y_{i_c(1)} \in R^T$ gets the following "sign" structure:

$$sign[Y_{i_c(1)}] = \alpha_{i_c}[-e_1^1| - e_1^2|\cdots|e_1^c|\cdots| - e_1^C]' \tag{33}$$

where $e_1^c \doteq [1\ 1\ \cdots\ 1] \in R^{T_c}$, T_c is the number of patterns per class, and

$$\begin{aligned} \alpha_{i_c} &= \pm 1 \quad \text{if} \quad w_{c,i_c(1)} \neq 0 \\ \alpha_{i_c} &= 0 \quad \text{if} \quad w_{c,i_c(1)} = 0. \end{aligned} \tag{34}$$

Proof: The analysis on the surface of $E_T(w_{ij})$ does not limit the ordering of patterns in $\mathcal{L}_e$. Hence, w.l.o.g. we assume that the patterns of the same class are contiguous in the learning environment. For the output layer, the following relationship holds:

$$\begin{aligned} y_{i(2)}(t) &= f'(a_{i(2)}(t))(x_{i(2)}(t) - d_{i(2)}(t)), \quad \forall i(2) \\ &= 1, \ldots, c, \ldots, C. \end{aligned} \tag{35}$$

Because of the exclusive coding hypothesis, if pattern t belongs to class c, then $d_c(t) = \bar{d}$; otherwise, $d_c(t) = \underline{d}$. It readily follows that

$$\begin{aligned} y_c(t) &< 0 \qquad \text{if } t \text{ belongs to class } c \\ y_c(t) &> 0 \qquad \text{if } t \text{ does not belong to class } c. \end{aligned} \tag{36}$$

According to the backward relationship of BP

$$y_{i_c(1)}(t) = f'(a_{i_c(1)}(t))w_{c,i_c(1)}y_c(t) \tag{37}$$

it follows that $y_{i_c(1)}(t)$ gets the same sign for all the patterns of a given class. As a result, the thesis follows from (36) and (37). □

Lemma 5: Let us consider a network and a learning environment that match the PR2 hyphoteses. Then

$$\mathcal{X}_o'\mathcal{Y}_1 = 0 \tag{38}$$

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

Proof: If the patterns are linearly separable, C hyperplanes exist that are defined by the vectors $\left\{\hat{A}_c \in R^{n(0)+1},\ c = 1, \ldots, C\right\}$ such that

$$sign[\hat{A}_c'\mathcal{X}_o'] = [-e_1^1| - e_1^2|\cdots|e_1^c|\cdots| - e_1^C]. \tag{39}$$

The solutions $\mathcal{Y}_1$ of (38) must necessarily satisfy the following equations:

$$\hat{A}_c'\mathcal{X}_o'\mathcal{Y}_1 = 0 \qquad \forall\, c = 1, \ldots, C$$

and, therefore, for each class c and for each neuron $i_c(1)$ of the hidden sublayer H_c, the following scalar equality must hold:

$$\left(\hat{A}_c'\mathcal{X}_o'\right)Y_{i_c(1)} = 0.$$

Because of (39) and (33) of Lemma 4, it easily follows that $\mathcal{Y}_1 = 0$ is the unique solution of (38). □

Lemma 6: Let the PR2 hypotheses hold, and let us assume the existence of a stationary point whose weights connecting the neurons of hidden sublayer H_c to corresponding output c all are null, i.e.

$$w_{c,i_c(1)} = 0, \qquad \forall i_c(1) \in H_c. \tag{40}$$

Then, this stationary point is not a local minimum.

Proof: Condition (40) implies that for $i(2) = c$, we get

$$y_{i(2)}(t) = f'(w_c)(f(w_c) - d_{i(2)}(t)) \neq 0 \tag{41}$$

where w_c is the bias for output c, and $d_c(t)$ is the target for pattern t.

From the network hypotheses, it follows that the hessian matrix has the following block-diagonal structure:

$$\mathcal{H} = \text{diag}[\mathcal{H}_1, \cdots, \mathcal{H}_c, \cdots, \mathcal{H}_C] \tag{42}$$

where $\mathcal{H}_c \in R^{(n(0)+1)n_c(1)+n_c(1)+1,(n(0)+1)n_c(1)+n_c(1)+1}$ is the submatrix associated to the subnetwork defined by the input layer, hidden sublayer H_c, and output c (see Fig. 2). This submatrix is partitioned as follows:

$$\mathcal{H}_c = \begin{bmatrix} \mathcal{H}_c(1,1) & \mathcal{H}_c(1,0) \\ \mathcal{H}_c(0,1) & \mathcal{H}_c(0,0) \end{bmatrix} \tag{43}$$

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

We observe that condition (40) implies that the delta trace $Y_{i_c(1)} \in R^T$, $\forall i_c(1) \in H_c$, is identically null. Hence, from BP equations, we obtain that $\mathcal{H}_c(0,0)$ is the null matrix. Now, the generic element of $\mathcal{H}_c(0,1)$ has the following expression:

$$\frac{\partial^2 E_T}{\partial w_{i_c(1),j(0)}\partial w_{i(2),i_c(1)}} = \sum_{t=1}^{T} x_{j(0)}(t)f'(a_{i_c(1)}(t))y_{i(2)}(t) \tag{44}$$

where $i(2) = c$ and $j(0)$ denotes the generic input. Hence, any subcolumn $\mathcal{H}_c^v(0,1) \in R^{n(0)+1}$ of $\mathcal{H}_c(0,1)$ can be written as

$$\mathcal{H}_c^v(0,1) = \mathcal{X}_o' \begin{bmatrix} f'(a_{i_c(1)}(1))y_c(1) \\ \cdots \\ f'(a_{i_c(1)}(T))y_c(T) \end{bmatrix} \doteq \mathcal{X}_o' \hat{Y}_{c(2)} \quad (45)$$

where $\hat{Y}_{c(2)} \in R^T$ has the sign structure explained by the right side of (39). From Lemma 5 and condition (41), we get that $\mathcal{H}_c^v(0,1)$ is not identically null. Therefore, matrix $\mathcal{H}_c$ has the following structure:

$$\mathcal{H}_c = \begin{bmatrix} P & D' \\ D & \varnothing \end{bmatrix} \quad (46)$$

where matrix P is assumed positive definite, and D is not the null matrix. By applying Sylvester's theorem (see p. 104 of [2]), it can be easily obtained that $\mathcal{H}_c$ has both positive and negative eigenvalues, and hence, from (42), the proof of the lemma directly follows. □

Remark 4: We notice that the stationary points assumed in Lemma 6 can actually exist. For example, let us consider a linearly separable set and a one-output network with symmetric squash functions that satisfies the PR2 hypotheses. In this case, the weight configuration having all the weights and biases null is a stationary point.

Proof of Theorem 2: From Lemma 6, it follows that if the PR2 hypotheses hold, the stationary points, having all null the weights connecting the neurons of any hidden sublayer to the corresponding output, are not local minima. Therefore, only the case in which the weights connecting each hidden sublayer with the corresponding output are not all null remains to be considered. In this case, Theorem 2 easily follows from a direct application of Theorem 1 because PR2 network conditions satisfy the PR1.1 and PR1.2 hypotheses, and the PR2 learning environment and output coding conditions satisfy the PR1.3 hypothesis because of Lemma 5. □

REFERENCES

[1] P. Baldi and K. Hornik , "Neural networks and principal component analysis: Learning from examples and local minima," *Neural Networks*, vol. 2, pp. 53–58, 1989.

[2] R. Bellman, *Introduction to Matrix Analysis*. New York: McGraw-Hill, 1974 (2nd ed.).

[3] E. K. Blum, "Approximation of Boolean functions by sigmoidal networks: Part I: XOR and other two-variable functions," *Neural computation*, vol. 1, pp. 532–540, 1989.

[4] H. Bourlard and C. Wellekens, "Speech pattern discrimination and multi-layered perceptrons," *Comput. Speech Language*, vol. 3, pp. 1–19, 1989.

[5] M. L. Brady, R. Raghavan, and J. Slawny, "Back-propagation fails to separate where perceptrons succeed," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 665–674, 1989.

[6] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*. Waltham, MA: Blaisdell, 1969.

[7] P. Cosi, Y. Bengio, and R. De Mori, "Phonetically-based multi-layered networks for vowel classification," *Speech Commun.*, vol. 9, no. 1, pp. 15–29, Feb. 1990.

[8] G. Cybenko, "Approximation by superpositions of a single sigmoidal function," *Math. Contr. Signal Syst.*, vol. 3, pp. 303–314, 1989.

[9] L. Elman and D. Zipser, "Learning the hidden structure of the speech," *J. Acoust. Soc. Amer.*, vol. 83, no. 4, 1988.

[10] M. Gori, "Apprendimento con supervisione in reti neuronali," *Ph.D. thesis*, Univ. di Bologna, Feb. 1990.

[11] M. Gori, Y. Bengio, and R. De Mori, "BPS: A learning algorithm for capturing the dynamical nature of speech," in *Proc. IEEE-IJCNN* (Washington DC), 1989, pp. 417–423, vol. II.

[12] M. Gori, G. Soda, and A. Tesi, "Optimal learning in experiments of handwritten character recognition," *DSI-RT 32/90*, Univ. di Firenze, 1990.

[13] M. Gori and A. Tesi, "Some examples of local minima during learning with backpropagation," in *Proc. Parallel Architectures Neural Networks* (Vietri sul Mare, Italy), May 1990.

[14] H. P. Graf, L. D. Jackel, and W. E. Hubbard, "VLSI implementation of a neural network model," *IEEE Comput.*, vol. 21, no. 3, Mar. 1988.

[15] S. J. Hanson and D. J. Burr, "Minkowsky-r backpropagation: Learning in connectionist models with non-Euclidean error signals," in *Proc. NIP87* (Denver, CO), 1988.

[16] R. Hecht-Nielsen, "Theory of backpropagation neural network," in *Proc. IEEE-IJCNN89* (Washinghton DC), 1989, pp. 593–605, vol. I.

[17] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators, " *Neural Networks*, vol. 2, pp. 359–366, 1989.

[18] W. Y. Huang and R. P. Lippman, "Neural net and traditional classifiers," in *Proc. NIP87* (Denver, CO), 1988, pp. 387–396.

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

[20] Y. le Cun, "Learning processes in an asymmetric threshold network," in *Disordered Systems and Biological Organization* (F. Soulie, E. Bienenstock, and G. Weisbuch, Eds.). Les Houches, France: Springer-Verlag, 1986, pp. 233–340.

[21] ——,"Generalization and network design strategies," in *Proc. Conectionism in Perspective*. New York: Elsevier, North Holland, 1989.

[22] ——,"A theoretical framework for backpropagation", in *Proc. 1988 Conectionist Models Summer Sch.* (D. Touresky, G. Hinton, and T. Sejnowski, Eds.). San Mateo, CA: Morgan Kauffmann, 1988, pp. 21–28.

[23] R. P. Lippman, "An introduction to computing with neural nets," *IEEE Acoust. Speech Signal Processing Mag.*, pp. 4–22, 1987.

[24] M. L. Minsky and S. A. Papert, *Perceptrons—Expanded Edition*. Cambridge, MA: MIT Press, 1988.

[25] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanism*. Washington DC: Spartan, 1962.

[26] 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*. Cambridge, MA: MIT Press, 1986.

[27] F. J. Pineda , "Generalization of back-propagation to recurrent neural networks," *Phys. Rev. Lett.*, vol. 59, no. 19, 1987.

[28] T. J. Sejnowsky and C. R. Rosemberg, "Parallel networks that learn to pronounce english text," *Complex Syst.*, vol. 1, pp. 145–168.

[29] E. D. Sontag and H. J. Sussman, "Backpropagation can give to spurious local minima even for networks without hidden layers, *Complex Syst.*, vol. 3, pp. 91–106, 1989.

[30] ——, "Backpropagation separates when perceptrons do," in *Proc. IEEE-IJCNN89* (Washington DC), 1989, pp. 639–642, vol. I.

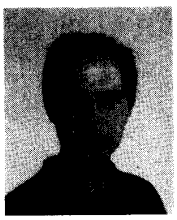
[31] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 37, no. 3, 1989.

[32] R. L. Watrous, "Learning algorithm for connectionist networks: Applied gradient methods for nonlinear optimization," in *Proc. First IEEE Int. Conf. Neural Networks* (San Diego, CA), 1987, pp. 619–627.

[33] P. J. Werbos, "Back-propagation: Past and future," in *Proc. IEEE Int. Conf. Neural Networks*. New York: IEEE Press, 1988, pp. 343–353.

[34] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[35] S. F. Zornetzer, J. L. Davis, and C. Lau, Eds., *An Introduction to Neural and Electronic Networks*. New York: Academic, 1990.



Marco Gori was born in Pistoia, Italy, on April 10, 1957. He received the Laurea in electronic engineering from Universitá di Fierenze, Italy, in 1984 and the Ph. D. degree in 1990 from Universitá di Bologna, Italy.

After receiving the Laurea, for two years, he was involved in designing microprocessor-based real-time systems for industrial applications. In 1990, he joined the Dipartmento di Sistemi e Informatica of Universitá di Fierenze, where he is currently a researcher in the area of computer science. His research interests include parallel processing, pattern recognition, and neural networks.

Alberto Tesi received the Laurea in electronic engineering from Universitá di Fierenze, Italy, in 1984. In 1989, he received the Ph. D. degree from Universitá di Bologna, Italy.

From 1984 to 1986, he worked as a system engineer at Autostrade s.p.a. In 1990, he joined the Dipartmento di Sistemi e Informatica of the Universitá di Firenze, where he is currently a researcher in the area of Automatic Control and System Theory. His research interests are mainly in linear and nonlinear systems analysis, robustness, and optimization.