Learning without Local Minima in Radial Basis Function Networks

Monica Bianchini, Paolo Frasconi, *Student Member, IEEE*, and Marco Gori, *Member, IEEE*

***Abstract*— Learning from examples plays a central role in artificial neural networks (ANN). However, the success of many learning schemes is not guaranteed, since algorithms like Backpropagation (BP) may get stuck in local minima, thus providing sub-optimal solutions. For feedforward networks, the theoretical results reported in [5,6,15,20] show that optimal learning can be achieved provided that certain conditions on the network and the learning environment are met. A similar investigation is put forward in this paper for the case of networks using radial basis functions (RBF) [10,14]. The analysis proposed in [6] is extended naturally under the assumption that the patterns of the learning environment are separable by hyperspheres. In that case, we prove that the attached cost function is local minima free with respect to all the weights. This provides us with some theoretical foundations for a massive application of RBF in pattern recognition.**

***Keywords*— Backpropagation, multilayered networks, pattern recognition, radial basis functions.**

I. Introduction

In the last few years there has been renewed interest in artificial neural networks which have been used for several different applications. For example, in pattern recognition the remarkable experience gained by numerous experiments suggests that ANNs are very promising tools. However, in order to exploit the potential learning capabilities of these techniques, a more thorough knowledge of the learning algorithms seems really necessary. Many of these algorithms essentially deal with the optimization of functions that, in pattern recognition, may be based on several thousand variables. The rich literature on optimization algorithms still has to be exploited in a suitable way in order to deal effectively with such functions. There is, however, no doubt that the shape of those cost functions deeply affects the success of any candidate algorithm. The presence of stationary points, and particularly of local minima, may severely limit the effectiveness of any neural network learning scheme. Although many suggestions have been given for avoiding local minima (see e.g.: [18,21]), their presence represents a serious problem from the computational point of view. For local minima free cost functions just simple gradient descent algorithms allow us to discover optimal solutions with a limited computational burden. This motivates the research of conditions for guaranteeing the absence of local minima. To some extent, such conditions give us an idea of the difficulties of the problem we are dealing with.

In the case of feedforward networks, analyses on optimal learning have been carried out by numerous researchers in the attempt to find examples of local minima and conditions which imply local minima free error surfaces. A general analysis on that problem can be found in [6] which also ensures that the error surfaces are local minima free in the case of *pyramidal networks*.

There have also been attempts to establish straightforward sufficient conditions for guaranteeing the absence of local minima. Roughly speaking BP convergence is guaranteed for *many input* and *many hidden* networks. In [6], the local minima analysis is specialized for the case of linearly separable patterns. Linear separability is likely to hold for patterns represented by "many coordinates." This hypothesis does not require constraints on the network architecture and particularly on the number of hidden units. As a consequence, a good generalization can be attained on the test set. The limitation, of course, is with the condition itself, which can be met in some problems of pattern recognition, but certainly not in general. Moreover, the use of hidden units for dealing with linearly separable patterns may seem unnecessary, since a one-layer network suffices for separating the learning set[1].

Poston *et al.* [15] have shown that the absence of local minima is guaranteed for networks having one hidden layer and as many hidden units as patterns. Yu [20] independently obtained an interesting result that contains Poston's result as a particular case. In practice, the conditions they assume obviously lead to nets with "many hiddens." In the case of "many hidden networks," the absence of local minima can be ensured without making assumptions about geometrical or statistical properties of data. Unfortunately, the resulting architectures are not likely to generalize very well to new examples because of the large number of trainable parameters. It is worth mentioning that the condition assumed in [20] is slightly less restrictive than the one proposed in [15], since it establishes that the number of hidden units required for ensuring local minima free error surface is equal to the number of different inputs. Although it seems unlikely to have coincident inputs in practical applications, Yu's analyses may also be useful when considering "clustering" of inputs, at least when those clusters are nearly reduced to single points.

Recently, many researchers have used RBF for a wide range of applications and have also proposed comparisons to feedforward nets (see e.g.: [10]). To the best of our

The authors are with Dipartimento di Sistemi e Informatica, Università di Firenze (Italy).

[1] As pointed out in [9], however, the generalization to new examples is better for networks with a hidden layer.

knowledge, however, no analyses have been carried out on the problem of local minima for these networks. In this paper we investigate the process of learning with RBF, in terms of the associated cost function's shape. When learning from examples without prior knowledge, that shape gives some indications on the difficulty of the problem to be solved, no matter what kind of learning algorithm is used. There are many different approaches for adjusting the weights of an RBF. Basically, we can distinguish between *hybrid* and *global* learning. With hybrid schemes, first, *self-organization* takes place at the hidden layer and, second, *LMS* optimization is performed at the output layer [10] [2]. Global learning simply deals with the optimization of the cost function associated with network and learning environment. We mainly focus on global learning because it is better suited for a theoretical analysis, but we also establish some intriguing connections is has to hybrid learning schemes.

In order to find the stationary points of the cost function, we compute its gradient in terms of vectorial equations. Afterwards, we introduce some hypotheses regarding the network and learning environment which, to some extent, are dual with respect to the *PR2* (for pattern recognition) hypotheses that were assumed in [6]. The network architecture has one hidden layer of locally-tuned processing units fully-connected with the inputs. Each output is instead connected to its own group of hidden units. As in [6] we assume the existence of a zero cost solution for the learning problem. The patterns of the learning environment are assumed to be separable by hyperspheres. This means that all positive examples must belong to regions bounded by hyperspheres whereas all eventual negative examples, which do not belong to the assumed classes, must be in the complementary domain. The main result of the paper is that the cost function is local minima free under these assumptions.

Moreover, we relate the computation of RBF to Kohonen's Learning Vector Quantization (LVQ) [8]. This comparison suggests that the selected updating of hidden neurons based on the distance to the current pattern is likely to improve the chances of finding optimal solutions, and also theoretically supports the use of hybrid learning. As stressed in [19], a good weight initialization plays a crucial role in avoiding local minima, and our analysis suggests that the self-organization step may also be useful from this point of view.

II. A Unified Vectorial Formulation of Learning in Radial Basis Functions

In this section, we define the formalism adopted throughout the paper and report some general results which will turn out to be useful in the following. We consider particularly, multilayered architectures with a hidden layer of *locally-tuned* units [10] and an output layer of *ordinary* processing units [16] [3]. The following three entities need to be defined: a network $\mathcal{N}$, a learning environment $\mathcal{L}_e$ (set of data used for learning), and a cost index E_T [16].

1) Network $\mathcal{N}$.

It has a multilayered architecture with one hidden layer[4]. With reference to index l, we distinguish between 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 $n(l)$. Each neuron of layer l is referred to by its index $i(l)$: $i(l)=1,\ldots,n(l)$. When the 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 previous level neurons[5]. As already mentioned, two different kinds of neurons are considered: *Locally-tuned processing units* in the hidden layer, and *ordinary processing units* in the output layer. Depending on the kind of neuron, the following processing is performed:

$$\begin{aligned} a_{i(1)}(t) &= w_{i(1)} + \frac{1}{\sigma^2_{i(1)}} \sum_{j(0)=1}^{n(0)} (x_{j(0)}(t) - w_{i(1),j(0)})^2 \\ &= w_{i(1)} + \frac{\|X_0(t) - W_{i(1)}\|^2}{\sigma^2_{i(1)}}, \\ a_{i(2)}(t) &= w_{i(2)} + \sum_{j(1)=1}^{n(1)} w_{i(2),j(1)} x_{j(1)}(t) \\ &= w_{i(2)} + W'_{i(2)} X_1(t). \end{aligned} \tag{2}$$

The output of neuron $i(l)$ is related to the activation as follows:

$$\begin{aligned} x_{i(1)} &= \tilde{f}(a_{i(1)}) \doteq \exp(-a_{i(1)}), \\ x_{i(2)} &= f(a_{i(2)}) \doteq 1/(1+\exp(-a_{i(2)})). \end{aligned} \tag{3}$$

In the above equations $w_{i(l),j(l-1)}$ denotes the weight of the link between the neurons $j(l-1)$ and $i(l)$, and

$$X_l(t) \doteq [x_{1(l)}(t), \ldots, x_{n(l)}(t)]',$$
$$W_{i(l)} \doteq [w_{i(l),1(l-1)}, \ldots, w_{i(l),n(l-1)}]'.$$

$w_{i(l)}$ denotes the threshold of ordinary and locally-tuned neurons and $\sigma_{i(l)}$ defines the "width" of the gaussians used as locally-tuned processing units.

Remark: Unlike the RBF proposed in [10], equations (2) include a bias term. Since

$$\begin{aligned} x_{i(1)} &= \exp(-a_{i(1)}) = \exp\Big(-\frac{\| X_0(t) - W_{i(1)} \|^2}{\sigma^2_{i(1)}} - w_{i(1)}\Big) \\ &= \exp(-w_{i(1)}) \exp\Big(-\frac{\| X_0(t) - W_{i(1)} \|^2}{\sigma^2_{i(1)}}\Big), \end{aligned} \tag{4}$$

[2] In many cases, after self-organization has taken place at the hidden layer, the optimization with respect to all the parameters may turn out to be better than simple *LMS* for the weights of the output layer.

[3] The RBF networks proposed in [10] have a linear output. The assumption made in this paper, however, does not change the essence of the analysis, which can also be carried out under the hypothesis of linear outputs.

[4] This restriction on the number of hidden layers is motivated by recent research on approximation techniques (see e.g.: [12]).

[5] In the sequel, both layer and pattern indices may be omitted for the sake of simplicity.

the contribution of hidden neuron $i(1)$ to output neuron $j(2)$ turns out to be

$$[w_{j(2),i(1)} \exp(-w_{i(1)})] \exp\Big(-\frac{\| X_0(t) - W_{i(1)} \|^2}{\sigma^2_{i(1)}}\Big).$$

By denoting $\hat{w}_{j(2),i(1)} \doteq [w_{j(2),i(1)} \exp(-w_{i(1)})]$, the RBF under consideration becomes exactly the one assumed in [10].

2) Learning Environment $\mathcal{L}_e$.

We use a set of supervised data. It is convenient to think of that set as a collection of T input/output pairs

$$\mathcal{L}_e \doteq \Big\{ (X_0(t), D(t)) \, , X_0(t) \in R^{n(0)}, \\ D(t) \in R^{n(2)}, \qquad t = 1, \ldots, T \Big\}.$$

$X_0(t)$ is the input pattern and $D(t)$ is its corresponding target. Each component of $D(t)$ is either $\underline{d}$ or $\overline{d}$ (asymptotical targets).

3) Cost Index.

Given $\mathcal{L}_e$, the input-output data fitting is measured by means of the cost function

$$\begin{aligned} E_T(w_{i,j}; \mathcal{N}, \mathcal{L}_e) &\doteq \frac{1}{2} \sum_{t=1}^{T} E_t \\ &= \frac{1}{2} \sum_{t=1}^{T} \sum_{j(2)=1}^{n(2)} [d_{j(2)}(t) - x_{j(2)}(t)]^2. \end{aligned} \tag{5}$$

In order to understand the basic result proposed in this paper, some more symbols have to be introduced which allow us to deal with a compact vectorial formulation of the problem.

1. $X_{i(l)} \doteq [x_{i(l)}(1), \ldots, x_{i(l)}(T)]' \in R^T$ 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 *layer output trace*. It is kept in the matrix

$$\mathcal{X}_l \doteq [X_{1(l)} \cdots X_{n(l)}] \ \in R^{T, n(l)}, \quad 0 \le l \le 2. \tag{6}$$

3. $w_{i(l),j(l-1)}$ is the weight that connects neuron $j(l-1)$ to neuron $i(l)$. The resulting matrix $\mathcal{W}_{l-1} \in R^{n(l), n(l-1)}$ is referred to as *layer weight matrix*. The symbol Ω denotes the weight space.
4. Let $y_{i(l)}(t) \doteq \partial E_t / \partial a_{i(l)}(t)$. We call *delta trace* the vector $Y_{i(l)} \doteq [y_{i(l)}(1), \ldots, y_{i(l)}(T)]' \in R^T$ and *layer delta trace* the matrix $\mathcal{Y}_l \doteq [Y_{1(l)} \cdots Y_{n(l)}] \ \in R^{T, n(l)}$. We denote by $\mathcal{S}_l^{\mathcal{Y}} \subset R^{T, n(l)}$ the set of all the $\mathcal{Y}_l$ generated by varying the weights in Ω.
5. We assume that there is no connection which skips a layer. Therefore, for the weights connecting layers 0 to 1, the gradient can be represented by a matrix $\mathcal{G}_1 \in R^{n(0)+2, n(1)}$, whose generic element is

$$g_{j(0),i(1)} = \begin{cases} \frac{\partial E_T}{\partial w_{i(1),j(0)}} & \text{if } \ j(0) \le n(0); \\ \frac{\partial E_T}{\partial \sigma_{i(1)}} & \text{if } \ j(0) = n(0) + 1; \\ \frac{\partial E_T}{\partial w_{i(1)}} & \text{if } \ j(0) = n(0) + 2. \end{cases}$$

Analogous relationships hold for $\mathcal{G}_2$, which, instead, belongs to $R^{n(1)+1, n(2)}$, as there is no row for differentiation with respect to the gaussian width term.

On the basis of these definitions, the gradient of (5) can be written in a very compact vectorial formula. Thanks to the hypothesis concerning the network architecture, the computation of the gradient can be performed by using the elegant BP style[6].

Gradient Computation.

- For the output layer, the use of BP computing scheme makes it possible to determine the stationary points as follows:

$$\mathcal{G}_{i(2)} = 0 \ \ \Rightarrow \hat{\mathcal{X}}_1' Y_{i(2)} = 0, \quad i(2) = 1, \ldots, n(2), \tag{7}$$

being $\hat{\mathcal{X}}_1 \doteq [X_{1(1)} \cdots X_{n(1)} \ \ \Pi] \ \in R^{T, n(1)+1}$, and $\Pi \doteq [1, \ldots, 1]' \in R^T$. Notice that in this case we can even obtain the matrix $\mathcal{G}_2 = \hat{\mathcal{X}}_1' \mathcal{Y}_2$ directly.
- For *locally-tuned* hidden neurons, the use of backpropagation rule leads to:

$$\mathcal{G}_{i(1)} = 0 \ \ \Rightarrow \tilde{\mathcal{X}}'_{0,i(1)} Y_{i(1)} = 0, \quad i(1) = 1, \ldots, n(1), \tag{8}$$

where

$$\tilde{\mathcal{X}}'_{0,i(1)} \doteq \begin{bmatrix} -2 \frac{\mathcal{X}_0' - \mathcal{M}_T'(W_{i(1)})}{\sigma^2_{i(1)}} \\ -2 \frac{\mathcal{A}'_{i(1)}}{\sigma_{i(1)}} \\ \Pi \end{bmatrix} \tag{9}$$

and $\mathcal{M}_T()$ is the *T-matrix-replica operator* which creates a matrix with T rows all equal to the argument $W'_{i(1)}$. In the above equation,

$$\mathcal{A}_{i(1)} \doteq [A_{i(1)}(1) \cdots A_{i(1)}(T)] \ \in R^T,$$

where $A_{i(1)}(t) \doteq \frac{\|X_0(t) - W_{i(1)}\|^2}{\sigma^2_{i(1)}}, t = 1, \ldots, T$, derives from differentiation with respect to gaussian widths, and $\Pi \doteq [1, \ldots, 1]' \in R^T$ derives from biases.

III. Optimal Learning for Patterns Separated by Hyperspheres

In this section we analyze the above stated problem of learning from examples as the optimization of the cost function (5). Basically, we deal with *global learning* and *batch mode* weight updating. The weight configurations determined by such a scheme are defined by equations (7) and (8), stating that the gradient of cost function (5) is null. As discussed further on, that learning scheme is somewhat related to *pattern mode* weight updating and, also, with *hybrid learning schemes*. Since we assume a multilayered RBF architecture, the results established in [6] can be extended conveniently. We focus, in particular, on all the

[6] It is worth mentioning that such a computation is not necessarily the most useful scheme to be used for learning. This is particularly clear for RBF where "centers and widths" are determined with very efficient self-organization schemes [10]. Notice, however, that we are interested in performing an analysis on the shape of the cost surface and, mainly, in determining the stationary points of the cost, no matter what learning algorithm is used.

weight configurations associated with $\mathcal{Y}_l \to 0, l = 1, 2$, and provide a natural extension of Theorems 1 and 2 proven in [6]. All the results in this section hold for a network $\mathcal{N}$ having enough capacity for the given learning environment $\mathcal{L}_e$, i.e. we assume the existence of one null cost solution.

Theorem 1 Cost function $E_T(w_{i,j}; \mathcal{N}, \mathcal{L}_e)$ has no local minima if the network $\mathcal{N}$ and its associated learning environment $\mathcal{L}_e$ satisfy the following *PR1-bis* hypotheses:

1. $n(2) \leq n(1)$ (pyramidal hypothesis);
2. the weight layer matrix $\mathcal{W}_1$ is full row rank;
3. the following relationships between the kernel of matrices $\tilde{\mathcal{X}}'_{0,i(1)}$ and $\mathcal{S}^{\mathcal{Y}}_{i(1)}$ hold

$$\mathrm{Ker}[\tilde{\mathcal{X}}'_{0,i(1)}] \cap \mathcal{S}^{\mathcal{Y}}_{i(1)} = \{\emptyset\}, \quad \forall\, i(1) = 1, \ldots, n(1). \quad (10)$$

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 reaches the absolute minimum with "null cost." At this point, it is worth making a few remarks concerning the hypotheses of the theorem. Like Theorem 1 in [6], the first two assumptions are easy to understand and are quite reasonable. The first one typically holds for pattern recognition problems, in which hidden and output layers contain input representations that are progressively more and more compressed when going towards the outputs. As also Baldi and Hornik [1] pointed out, the second hypothesis is not very restrictive and holds with probability one for random matrices. However, the problem with *PR1-bis.2* assumption is that it must be checked throughout the learning process. The third hypothesis is the hardest to check, since it requires the knowledge of $\mathcal{S}^{\mathcal{Y}}_{i(1)}$, i.e. the set of all the $Y_{i(1)} \in R^T$ generated by varying the weights in Ω.

In order to discover meaningful conditions, with a straightforward practical interpretation, we need to put forward assumptions about data. In the following theorem we require that the input patterns be separated by hyperspheres.

Theorem 2 Cost function $E_T(w_{i,j}; \mathcal{N}, \mathcal{L}_e)$ has no local minima if the network $\mathcal{N}$ and the learning environment $\mathcal{L}_e$ satisfy the following *PR2-bis* hypotheses:

- Network.
  1. the network has C outputs, where C is the number of classes;
  2. 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 $i_c(1)$.
- Output coding.
  Exclusive coding is used for the output, i.e., if pattern t belongs to class c, $c = 1, \ldots, C$, then

$$d_i(t) = \begin{cases} \overline{d} & \text{if } i = c, \\ \underline{d} & \text{otherwise.} \end{cases} \quad (11)$$

- Learning environment.
  All the patterns of $\mathcal{L}_e$ are separated by hyperspheres, i.e. for each class c, with $c = 1, \ldots, C$, there exists a pair $(\mathcal{C}'_c, r_c)'$, $\mathcal{C}'_c = \big(c_c(1), c_c(2), \ldots, c_c(n(0))\big)' \in R^{n(0)}$, $r_c \in R^+$, such that

$$\begin{array}{ll} \| X_0(t) - \mathcal{C}_c \| - r_c \leq 0, & \forall\, t \text{ in class c}, \\ \| X_0(t) - \mathcal{C}_c \| - r_c > 0, & \forall\, t \text{ not in class c}, \end{array} \quad (12)$$

  where $\mathcal{C}_c$ and r_c are center and radius of the hypersphere, respectively.

Proof: see the Appendix. □

Some remarks are worth mentioning concerning this result.

1) Network architecture: As already noticed for feedforward nets, network assumption *PR2-bis.2* is quite reasonable in practice [6]. Plaut and Hinton [13] showed that this kind of architecture learns faster than one with fully-connected layers. Jacobs *et al.* [7] considered network assumption *PR2-bis.2* as a first step towards the conception of modular architectures that are usually well-suited for good generalization.

2) Output coding: It is quite easy to realize that Theorem 2 also holds for networks with only one output, and *positive examples* belonging to a region delimited by a given hypersphere. In this case, the class c is simply coded by 1, whereas $\overline{c}$ is coded by 0. This extension also holds in the case of C exclusively coded classes and a set of *negative examples* which do not belong to the assumed classes.

3) Beyond hyperspheres? When choosing more general processing units for the hidden layer, one may wonder if the results established in Theorem 2 also hold for different separation surfaces. It can be shown that this is indeed the case for more general RBF in which the locally-tuned processing units follow the equation

$$a_{i(1)}(t) = (X_0(t) - W_{i(1)})' Q_{i(1)} (X_0(t) - W_{i(1)}) + w_{i(1)}, \quad (13)$$

being $Q_{i(1)} \in R^{n(0),n(0)}$ a symmetric matrix associated with neuron $i(1)$. In that case, following exactly the proving scheme drawn in Lemma 5, it can be shown that the cost function (5) is local minima free if the patterns are separated by a general quadratic surface. This extension is related to *input preprocessing* which allows us to obtain complex separation surfaces with just one layer network (linear machines) (see e.g.: [11], pp. 29-30). Notice that the quadratic preprocessing for linear machines is *not equivalent* to the RBF studied in this paper for at least two reasons. First, with linear machines and quadratic preprocessing, the quadratic separation among patterns of different classes is a necessary condition for determining their parameters. Consequently, when the separation condition does not hold, no solution can be found, whereas with RBF the condition established by Theorem 2 *is only sufficient* and, therefore, solutions can be found even when the separation condition is not met. Second, the generalization to new examples is generally different. With multilayered architectures like RBF, the prior knowledge on a

given problem can be exploited much better than for linear machines and this may lead to significant improvements in generalization.

4) LMS-Threshold Functions: The analysis put forward in this paper assumes quadratic cost functions having "asymptotical targets" which force the learning algorithms towards asymptotical weights. However, in practice this may limit to the efficiency of a learning algorithm. The use of *LMS-Threshold* cost functions [17] removes this kind of limitation, and allows us to guarantee the absence of local minima while ensuring finite weight solutions (see e.g. [4,17], for the case of feedforward nets).

5) Forcing limited hyperspheres: In many cases, the optimization algorithms for the cost (5) may discover hyperspheres with "large" σ. In the limit, very large values for σ produce discrimination surfaces very similar to those obtained with ordinary processing units. If besides classes discrimination one is interested in rejecting "negative" input patterns (i.e., to get nearly zero outputs for patterns that do not belong to any valid class), then separation with smaller hyperspheres allows us to better exploit the advantages of locally tuned processing units. A possible way of restricting the input regions mapped to valid classes is to provide negative examples during learning. Unfortunately, in most relevant applications defining the set of negative examples is nearly impossible. We can however force a prior constraint on the radii of the hyperspheres and learn under such a constraint. We can easily demonstrate that Theorem 2 also holds in this case. It suffices to assume $\sigma = B/(1+\exp(-\sigma_p))$ and consider the cost as a function of σ_p. In so doing, σ is constrained into the interval $(0, B)$ that can be properly chosen with some prior knowledge about the problem to be solved.

6) Relationships with hybrid learning: It is well-known that pattern mode weight updating departs to some extent from exact gradient descent of cost function E_T. However, the adoption of "small" learning rates[7] arbitrarily reduces the difference with respect to batch mode [16]. Following the gradient descent scheme in pattern mode, the weights of the locally-tuned processing units are updated according to

$$W_{i(1)}^{(r+1)} = W_{i(1)}^{(r)} + \frac{2\epsilon y_{i(1)}(t)}{\sigma_{i(1)}^2}(X_0(t) - W_{i(1)}^{(r)}), \quad \forall\, i(1) = 1, \ldots, n(1),\ \ t = 1, \ldots, T, \tag{14}$$

where ϵ is a learning rate, $W'_{i(1)} \doteq [w_{i(1),1(0)}, \ldots, w_{i(1),n(0)}]'$ and r is the iteration index. This equation is essentially the same as the one used for performing the updating of codebook vectors in Learning Vector Quantization (LVQ) [8]. In particular, $\frac{2\epsilon y_{i(1)}(t)}{\sigma_{i(1)}^2}$ plays the role of coefficient $\alpha(t)$ and is consistent with the requirement of being asymptotically null. The basic difference, however, is that in BP optimization schemes each pattern of the learning environment affects each vector $W_{i(1)}$ (codebook vector), whereas with LVQ just the patterns closest to $X_0(t)$ in the Euclidean metric are taken into account (one vector in LVQ1, two vectors in LVQ2 and LVQ3). In practice, however, in BP optimization schemes the updating of the codebook vectors depends strongly on the distance $||X_0(t) - W_{i(1)}||$. For a given pattern $X_0(t)$, all codebook vectors $W_{i(1)}$ such that $x_{i(1)}(t) \approx 0$, in practice, are not updated since $y_{i(1)} \approx 0$ in that case. One more significant difference is that BP optimization typically leads to distributed representations, whereas the LVQ solution produces codevectors that locally model clusters of points.

These remarks also suggest that hybrid learning as described in [10] may turn out to be less affected by suboptimal solutions. In that case the self-organization step provides a first tuning of the codebook vectors such that they go away from each other. In so doing, in practice, the number of codebook vectors reacting to a given pattern decreases. The resulting effect is that of linking the reaction of processing units to specific patterns. As a result, for a given processing unit the optimization takes place in a sub-set of the learning environment. Hence, hybrid learning performs a sort of *divide and conquer* and is likely to be faster than ordinary BP optimization beginning from random weights. The conclusion is that in many practical cases hybrid learning can be very successful, particularly if BP optimization is used as second step instead of simple *LMS* used for learning $\mathcal{W}_1$.

7) Beyond $\mathcal{Y}_1$'s sign rule: Lemma 4 plays a very crucial role in the proof of Theorem 2. Basically it describes $\mathcal{Y}_1$ in terms of the signs of its components. The sign structure proposed by Lemma 4 holds for any point of the weight space. Theorem 2 is based on that description of $\mathcal{Y}_1$ and assumes no other knowledge on nonlinear map $\mathcal{Y}_1(\Omega)$. Theorem 2 could be improved by a more accurate description of that map.

IV. Conclusions

In this paper we have given some analyses on the problem of learning in RBF independently of the learning algorithm. Using the proposed vectorial formalism, it is shown that when patterns are separated by hyperspheres, or more generally by quadratics, the cost function associated with radial basis function is local minima free. Notice that our conditions are only sufficient and are based on an approximate knowledge of the map $\mathcal{Y}_1(\Omega)$ that derives from the assumption of asymptotical targets, typical of pattern recognition problems. However, the theoretical conclusions reported in [3] indicate that the use of non-asymptotical targets, commonly used in function approximation, introduces additional spurious local minima. It is worth mentioning that even with local minima free cost functions, most learning algorithms may be trapped into a plateau due to the saturation of either locally-tuned processing units or ordinary squashing units. The analysis proposed in the paper allows us also to establish some intriguing comparisons to hybrid learning. Basically, hybrid learning performs a sort of divide and conquer, with respect to the learning envi-

[7] The term "small" is strictly related to the dimension of the learning environment.

ronment, that is likely to improve optimal convergence.

Acknowledgments

This research was partially supported by MURST 40%.

Appendix

In the sequel, the track of the proofs given in [6] will be followed, as almost all results obtained there can be extended to RBF using the stated vector notation. The basic ideas used in proving Theorem 2 are quite simple. The assumed data structure allows us to conclude that $\mathcal{Y}_1 \to 0$ (Lemma 5). This conclusion relies on $\mathcal{Y}_1$'s sign rule that holds under the assumption of asymptotical targets (Lemma 4). As already mentioned, this assumption limits our analysis to pattern recognition. By using the *PR2-bis* network hypotheses, $\mathcal{Y}_2 \to 0$ follows directly from $\mathcal{Y}_1 \to 0$, unless the weights of the last layer are null. In this special case however, using Lemma 6, we prove that these configurations are not local minima. Finally, the condition $\mathcal{Y}_2 \to 0$ implies, according to Lemma 3, that only the configurations for which $E_T \to 0$ are minima. Some preliminary lemmas need to be claimed from [6] and slightly restated. Lemma 5 and 6, instead, are proven under the new assumption on RBF and patterns separated by hyperspheres.

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

$$\forall\, i(1), j(0) \qquad |\tfrac{\partial E_T}{\partial w_{i(1),j(0)}}| < \delta \quad \Rightarrow \quad \| Y_{i(1)} \|_\infty < \epsilon. \tag{15}$$

Proof: see [6]. □

Lemma 2 Let us assume that the *PR1-bis* 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

$$\forall\, i(1), j(0) \quad |\tfrac{\partial E_T}{\partial w_{i(1),j(0)}}| < \delta \quad \Rightarrow \quad \| \mathcal{Y}_2 \|_\infty < \epsilon. \tag{16}$$

Proof: see [6]. Note that, in this case, the compactness hypothesis implies

$$\tilde{f}'(a_{i(1)}(t)) < \tilde{d} < 0, \quad 0 < d < f'(a_{i(2)}(t)). \tag{17}$$

As a result, we need to choose $d_{\min} = \min\{d, |\tilde{d}|\}$ in order to extend the proof of the lemma. □

Lemma 3 Let us consider a configuration for which $\mathcal{Y}_2 \to 0$ and the cost $E_T \neq 0$. For this configuration there exist $i(2)$ and $j(1)$ such that

$$\frac{\partial^2 E_T}{\partial w^2_{i(2),j(1)}} < 0. \tag{18}$$

Proof: see [6]. □

Remark: The proof that is given in [6] assumes that there is output/target matching for all the patterns except $\hat{t}$, for which there is a mismatch between the target $d_{i(2)}(\hat{t})$ and the output $f(a_{i(2)}(\hat{t}))$. The scheme of this proof however, can easily be extended to all the configurations such that $\mathcal{Y}_2 \to 0$ and $E_T \neq 0$.

Theorem 1 : Proof.

If the *PR1-bis* hypotheses hold, then Lemma 2 indicates that the points for which the gradient is "zero" are the same for which $\mathcal{Y}_2$ is "zero." From Lemma 3, it follows that only the points whose cost is "zero" represent attractive points for the gradient descent (minima of the cost). □

Lemma 4 Let us consider a network whose architecture matches *PR2-bis* hypotheses and assume also exclusive coding for the C classes. Partitioning the vector $Y_{i_c(1)}$ as

$$Y_{i_c(1)} = \left[Y^1_{i_c(1)} | Y^2_{i_c(1)} | \cdots | Y^c_{i_c(1)} | \cdots | Y^C_{i_c(1)} \right] \in R^T, \tag{19}$$

we get the following "sign" structure

$$\mathrm{sign}[Y_{i_c(1)}] = \alpha_{i_c} \left[-e^1_1 | - e^2_1 | \cdots | e^c_1 | \cdots | - e^C_1 \right]', \tag{20}$$

where $e^c_1 \doteq [1, 1, \ \ldots \ 1] \in R^{T_c}$, T_c is the number of patterns per class, and

$$\begin{array}{lll} \alpha_{i_c} = \pm 1 & \text{if} & w_{c,i_c(1)} \neq 0, \\ \alpha_{i_c} = 0 & \text{if} & w_{c,i_c(1)} = 0. \end{array} \tag{21}$$

Proof: see [6]. Notice that, in this case, the derivative of locally-tuned output function is always negative. □

Lemma 5 Let us consider a network $\mathcal{N}$ and a learning environment $\mathcal{L}_e$ that match the *PR2-bis* hypotheses. Then

$$\tilde{\mathcal{X}}'_{0,i(1)} Y_{i(1)} = 0, \qquad i(1) = 1, \ldots, n(1) \tag{22}$$

only admits the solution $Y_{i(1)} = 0$.

Proof: If the patterns are separable by hyperspheres, then (12) holds. As a result, for each $W_{i(1)} \in R^{n(0)}$, there exist C vectors $\Phi'_c(W_{i(1)}) = (\tilde{\Phi}'(W_{i(1)}), 1, \beta(W_{i(1)}))' \in R^{n(0)+2}$, where

$$\tilde{\Phi}(W_{i(1)}) = \frac{2(W_{i(1)} - \mathcal{C}_c)}{\sigma_{i(1)}}, \qquad c = 1, \ldots, C \tag{23}$$

and

$$\beta(W_{i(1)}) = \frac{2}{\sigma^3_{i(1)}} [r^2_c - \| \mathcal{C}_c - W_{i(1)} \|^2], \qquad c = 1, \ldots, C \tag{24}$$

such that

$$\begin{aligned} \mathrm{sign}[\Phi'_c \tilde{\mathcal{X}}'_{0,i(1)}] &= \mathrm{sign}\Big[-\frac{2}{\sigma^2_{i(1)}} \Big\{ \tilde{\Phi}' \left(\mathcal{X}'_0(t) - \mathcal{M}'_T(W_{i(1)}) \right) \\ &\quad + \frac{\| X_0(t) - W_{i(1)} \|^2}{\sigma_{i(1)}} \Big\} + \beta \Big] \\ &= \left[-e^1_1 | - e^2_1 | \cdots | e^c_1 | \cdots | - e^C_1 \right]. \end{aligned} \tag{25}$$

The solution $Y_{i(1)}$ of equation (22) must necessarily satisfy the following equations

$$\Phi'_c \tilde{\mathcal{X}}'_{0,i(1)} Y_{i(1)} = 0, \qquad \forall\, c = 1, \ldots, C. \tag{26}$$

Hence, for each class c and for each neuron $i_c(1)$ of the hidden sub-layer H_c, the following equality must hold

$$\Phi'_c \tilde{\mathcal{X}}'_0 Y_{i_c(1)} = 0 \tag{27}$$

Because of (25) and (20) in Lemma 4, it easily follows that $Y_{i_c(1)} = 0, \ \forall\, i_{c(1)} = 1, \ldots, n_{c(1)}$ and $c = 1, \ldots, C$. Consequently $Y_{i(1)} = 0$ is the unique solution of (22). □

Lemma 6 Let *PR2-bis* hypotheses hold and let us assume the existence of a stationary point such that the weights connecting the neurons of the hidden sub-layer H_c to the corresponding output c are all null, i.e.

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

Then, this stationary point is not a local minimum.

Proof: Condition (28) implies that for $i(2) = c$ we get

$$y_{i(2)}(t) = f'(w_c)(f(w_c) - d_c(t)) \neq 0, \tag{29}$$

where w_c is the bias for output c, $d_c(t)$ is the target for pattern t and f is the squash-function of output neurons. From the network hypotheses it follows that the Hessian matrix $\mathcal{H}$ has the following block-diagonal structure

$$\mathcal{H} = \mathrm{diag}[\mathcal{H}_1, \cdots, \mathcal{H}_c, \cdots, \mathcal{H}_C], \tag{30}$$

where $\mathcal{H}_c \in R^{(n(0)+2)n_c(1)+n_c(1)+1,(n(0)+2)n_c(1)+n_c(1)+1}$ is the sub-matrix associated to the sub-network defined by the input layer, hidden sub-layer H_c, and output c. This sub-matrix is partitioned as follows

$$\mathcal{H}_c = \begin{bmatrix} \mathcal{H}_c(1,1) & \mathcal{H}_c(1,0) \\ \mathcal{H}_c(0,1) & \mathcal{H}_c(0,0) \end{bmatrix}, \tag{31}$$

where

$$\mathcal{H}_c(1,1) \in R^{n_c(1)+1,n_c(1)+1}$$

and

$$\mathcal{H}_c(0,0) \in R^{(n(0)+2)n_c(1),(n(0)+2)n_c(1)}$$

are generated by the weights connecting the hidden units to the outputs and the inputs to the hiddens respectively, while

$$\mathcal{H}_c(0,1) = \mathcal{H}_c(1,0)' \in R^{(n(0)+2)n_c(1),n_c(1)+1}$$

represents the cross-contribution of these weights.

We observe that condition (28) 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. The generic element of $\mathcal{H}_c(0,1)$ has the following expression

$$\frac{\partial^2 E_T}{\partial w_{i_c(1),j(0)} \partial w_{i(2),i_c(1)}} = 2 \sum_{t=1}^{T} \frac{(x_{j(0)}(t) - w_{i_c(1),j(0)})}{\sigma^2_{i_c(1)}} \tilde{f}'(a_{i_c(1)}(t)) y_{i(2)}(t), \tag{32}$$

where $i(2) = c$, $j(0)$ denotes the generic input and $\tilde{f}$ is the locally-tuned output function. Hence, any sub-column $\mathcal{H}^v_c(0,1) \in R^{n(0)+2}$ of $\mathcal{H}_c(0,1)$ can be written as

$$\mathcal{H}^v_c(0,1) = \tilde{\mathcal{X}}'_{0,i(1)} \begin{bmatrix} \tilde{f}'(a_{i_c(1)}(1)) y_c(1) \\ \cdots \\ \tilde{f}'(a_{i_c(1)}(T)) y_c(T) \end{bmatrix} \doteq \tilde{\mathcal{X}}'_{0,i(1)} \tilde{Y}_c, \tag{33}$$

where $\tilde{Y}_c \in R^T$ has the sign structure explained by the right side of equation (25). From Lemma 5 and condition (28) we get that $\mathcal{H}^v_c(0,1)$ is not identically null. Therefore, the matrix $\mathcal{H}_c$ has the following structure

$$\mathcal{H}_c = \begin{bmatrix} P & D' \\ D & \emptyset \end{bmatrix}, \tag{34}$$

where P is assumed positive definite and D is not the null matrix. By applying Sylvester's theorem (e.g. see [2] page 104) it can be obtained that $\mathcal{H}_c$ has both positive and negative eigenvalues. Hence the proof of the lemma directly follows from (30). □

Theorem 2 : Proof.

From Lemma 6 it follows that, if the *PR2-bis* hypotheses hold, the stationary points defined by the condition of zero weight for the connections between any hidden sub-layer to the corresponding output are not local minima. Therefore, we only have to examine the case in which the weights connecting each hidden sub-layer to the corresponding output are not all null. In this case, Theorem 2 easily follows from a direct application of Theorem 1 because *PR2-bis* network conditions satisfy *PR1.1-bis* and *PR1.2-bis* hypotheses, and *PR2-bis* learning environment and output coding conditions satisfy *PR1.3-bis* hypothesis. □

References

[1] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, pp. 53–58, 1989.

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

[3] M. Brady, R. Raghavan, and J. Slawny, "Backpropagation fails to separate where perceptrons succeed," *IEEE Transactions on Circuits and Systems*, vol. 36, pp. 665–674, 1989.

[4] P. Frasconi and M. Gori, "Backpropagation for linearly separable patterns: a detailed analysis," in *IEEE International Conference on Neural Networks*, (S. Francisco CA), pp. 1818–1822, IEEE Press, 1993.

[5] P. Frasconi, M. Gori, and A. Tesi, "Successes and failures of backpropagation: a theoretical investigation," in *Progress in Neural Networks* (O. Omidvar, ed.), Ablex Publishing. (in press).

[6] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-14, no. 1, pp. 76–86, 1992.

[7] R. Jacobs, M. Jordan, and A. Barto, "Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks," tech. rep., COINS, 1990.

[8] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[9] Y. Le Cun, "Generalization and network design strategies," in *Connectionism in Perspective*, Elsevier Publishers, 1989.

[10] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.

[11] N. Nilsson, *Learning Machines*. McGraw-Hill, 1965.

[12] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[13] D. Plaut and G. Hinton, "Learning set of filters using back-propagation," *Computer Speech and Language*, vol. 2, pp. 35–61, 1987.

[14] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[15] T. Poston, C. Lee, Y. Choie, and Y. Kwon, "Local minima and backpropagation," in *International Joint Conference on Neural Networks*, (Seattle WA), pp. 173–176, IEEE Press, 1991.

[16] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing* (D. Rumelhart and J. McClelland, eds.), vol. 1, ch. 8, pp. 318–362, Cambridge: MIT Press, 1986.

[17] E. Sontag and H. J. Sussman, "Backpropagation separates when perceptrons do," in *International Joint Conference on Neural Networks*, (Washington DC), IEEE Press, 1989.

[18] S. Wang and C. H. Hsu, "Terminal attractor learning algorithms for backpropagation neural networks," in *International Joint Conference on Neural Networks*, (Singapore), pp. 183–189, IEEE Press, November 1991.

[19] L. Wessels and E. Barnad, "Avoiding false local minima by proper initialization of connections," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 899–905, 1992.

[20] X. Yu, "Can backpropagation error surface not have local minima?," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 1019–1020, 1992.

[21] M. Zak, "Terminal attractors in neural networks," *Neural Networks*, vol. 2, pp. 259–274, 1989.

Monica Bianchini Monica Bianchini received the Laurea in Mathematics from Università di Firenze in 1989. After receiving the Laurea, for two years, she was involved in designing vectorial and parallel software for solving differential equations. She is currently a Ph.D. candidate at Dipartimento di Sistemi e Informatica (Università di Firenze). Her research interests include neural networks, optimization and numerical methods for differential equations.

Paolo Frasconi (S '91) received the degree in electronic engineering from Università di Firenze, Italy in 1990. Since 1991 he is with Dipartimento di Sistemi e Informatica, Firenze, where he is currently a Ph.D. candidate in Computer Science. In 1992 he was a visiting scholar in the Department of Brain and Cognitive Science at M.I.T. His research interests include neural networks, speech recognition, and parallel computing.

Marco Gori (S '89 – M '91) received the "Laurea" in electronic engineering from the University of Florence, Italy, in 1984, and the Ph.D. degree in 1990 from the Università di Bologna, Italy.

After receiving the Laurea, for two years, he was involved in designing microprocessor-based real-time systems for industrial applications. While working towards the Ph.D. degree, he was with the Dipartimento di Sistemi e Informatica (Università di Firenze) and with the School of Computer Science (McGill University, Montréal). In 1990, he joint the Dipartimento di Sistemi e Informatica (Università di Firenze), where he is currently an Associate Professor of Computer Science.

His main research interests are in parallel processing, speech recognition, and neural networks. Dr. Gori is a member of the program committee of the Italian Workshop on Neural Networks and of the Italian Workshop of Neural Networks for Speech Processing.