Learning in Feed-Forward Artificial Neural Networks II

Lluís A. Belanche Muñoz
Universitat Politècnica de Catalunya, Spain

INTRODUCTION

Supervised Artificial Neural Networks (ANN) are information processing systems that adapt their functionality as a result of exposure to input-output examples. To this end, there exist generic procedures and techniques, known as *learning rules*. The most widely used in the neural network context rely in derivative information, and are typically associated with the Multilayer Perceptron (MLP). Other kinds of supervised ANN have developed their own techniques. Such is the case of Radial Basis Function (RBF) networks (Poggio & Girosi, 1989). There has been also considerable work on the development of *adhoc* learning methods based on evolutionary algorithms.

BACKGROUND

The problem of learning an input/output relation from a set of examples can be regarded as the task of approximating an unknown function from a set of data points, which are possibly sparse. Concerning approximation by classical feed-forward ANN, these networks implement a parametric approximating function and have been shown to be able of representing generic classes of functions (as the continuous or integrable functions) to an arbitrary degree of accuracy. In general, there are three questions that arise when defining one such parameterized family of functions:

1. What is the most adequate parametric form for a given problem?
2. How to find the best parameters for the chosen form?
3. What classes of functions can be represented and how well?

The most typical problems in a ANN supervised learning process, besides the determination of the learning parameters themselves, include (Hertz, Krogh & Palmer, 1991), (Hinton, 1989), (Bishop, 1995):

1. The possibility of getting stuck in *local optima* of the cost function, in which conventional non-linear optimization techniques will stay forever. The incorporation of a global scheme (like multiple restarts or an annealing schedule) is surely to increase the chance of finding a better solution, although the cost can become prohibitedly high. A feed-forward network has multiple equivalent solutions, created by weight permutations and sign flips. Every local minima in a network with a single hidden layer of h_1 units has $s(h_1)=h_1!2^{h_1}$ equivalent solutions, so the chances of getting in the basin of attraction of one of them are reasonable high. The complexity of the error surface –especially in very high dimensions– makes the possibility of getting trapped a real one.
2. Long *training times*, *oscillations* and network *paralysis*. These are features highly related to the specific learning algorithm, and relate to bad or too general choices for the parameters of the optimization technique (such as the learning rate). The presence of saddle points—regions where the error surface is very flat—also provoke an extremely slow advance for extensive periods of time. The use of more advanced methods that dynamically set these and other parameters can alleviate the problem.
3. *Non-cumulative* learning. It is hard to take an already trained network and re-train it with additional data without losing previously learned knowledge.
4. The *curse of dimensionality*, roughly stated as the fact that the number of examples needed to represent a given function grows exponentially with the number of dimensions.
5. Difficulty of finding a *structure* in the training data, possibly caused by a very high dimension or a distorting pre-processing scheme.

6. Bad *generalization*, which can be due to several causes: the use of poor training data or attempts to extrapolate beyond them, an excessive number of hidden units, too long training processes or a badly chosen regularization. All of them can lead to an *overfitting* of the training data, in which the ANN adjusts the training set merely as an *interpolation* task.
7. Not amenable to *inspection*. It is generally arduous to interpret the knowledge learned, especially in large networks or with a high number of model inputs.

LEARNING IN RBF NETWORKS

A Radial Basis Function network is a type of ANN that can be viewed as the solution of a high-dimensional curve-fitting problem. Learning is equivalent to finding a surface providing the best fit to the data. The RBF network is a two-layered feed forward network using a linear transfer function for the output units and a radially symmetric transfer function for the hidden units. The computation of a hidden unit is expressed as the composition of two functions, as:

$$F_i(\boldsymbol{x}) = \{g(h(\boldsymbol{x},\boldsymbol{w}_i)),\ \boldsymbol{w}_i \in R^n\},\quad \boldsymbol{x}\in R^n \qquad (1)$$

with the choice $h(\boldsymbol{x},\boldsymbol{w}_i)= ||\boldsymbol{x}-\boldsymbol{w}_i||/\theta$ (or other distance measure), with $\theta>0$ a smoothing term, plus an activation g which very often is a monotonically decreasing response from the origin. These units are localized, in the sense that they give a significant response only in a neighbourhood of their centre $\boldsymbol{w}_i$. For the activation function a Gaussian $g(z)=exp(-z^2/2)$ is a preferred choice.

Learning in RBF networks is characterized by the separation of the process in two consecutive stages (Haykin, 1994), (Bishop, 1995):

1. Optimize the free parameters of the hidden layer (including the smoothing term) using only the $\{\boldsymbol{x}\}_i$ in D. This is an *unsupervised* method that depends on the *input sample distribution*.
2. With these parameters found and frozen, optimize the $\{c_i\}_i$, the hidden-to-output weights, using the full information in D. This is a *supervised* method that depends on the given *task*.

There are many ways of optimizing the hidden-layer parameters. When the number of hidden neurons equals the number of patterns, each pattern may be taken to be a center of a particular neuron. However, the aim is to form a representation of the probability density function of the data, by placing the centres in only those regions of the input space where significant data are present. One commonly used method is the *k-means algorithm* (McQueen, 1967), which in turn is an approximate version of the maximum-likelihood (ML) solution for determining the location of the means of a mixture density of component densities (that is, maximizing the likelihood of the parameters with respect to the data). The Expectation-Maximization (EM) algorithm (Duda & Hart, 1973) can be used to find the exact ML solution for the means and covariances of the density. It seems that EM is superior to k-means (Nowlan, 1990). The set of centres can also be selected randomly from the set of data points.

The value of the smoothing term can be obtained from the clustering method itself, or else estimated a posteriori. One popular heuristic is:

$$\theta = \frac{d}{\sqrt{2M}} \qquad (2)$$

where d is the maximum distance between the chosen centers and M is the number of centers (hidden units). Alternatively, the method of *Distance Averaging* (Moody and Darken, 1989) can be used, which is the global average over all Euclidean distances between the center of each unit and that of its nearest neighbor.

Once these parameters are chosen and kept constant, assuming the output units are linear, the (square) error function is quadratic, and thus the hidden-to-output weights can be fast and reliably found iteratively by simple gradient descent over the quadratic surface of the error function or directly by solving the minimum norm solution to the over determined least-squares data fitting problem (Orr, 1995).

The whole set of parameters of a RBF network can also be optimized with a global gradient descent procedure on all the free parameters at once (Bishop, 1995), (Haykin, 1994). This brings back the problems of local minima, slow training, etc, already discussed. However, better solutions can in principle be found, because the unsupervised solution focuses on esti-