Learning in Feed-Forward Artificial Neural Networks I

Lluís A. Belanche Muñoz
Universitat Politècnica de Catalunya, Spain

INTRODUCTION

The view of artificial neural networks as adaptive systems has lead to the development of ad-hoc generic procedures known as *learning rules*. The first of these is the Perceptron Rule (Rosenblatt, 1962), useful for single layer feed-forward networks and linearly separable problems. Its simplicity and beauty, and the existence of a *convergence theorem* made it a basic departure point in neural learning algorithms. This algorithm is a particular case of the Widrow-Hoff or *delta* rule (Widrow & Hoff, 1960), applicable to continuous networks with no hidden layers with an error function that is quadratic in the parameters.

BACKGROUND

The first truly useful algorithm for feed-forward multilayer networks is the *backpropagation* algorithm (Rumelhart, Hinton & Williams, 1986), reportedly proposed first by Werbos (1974) and Parker (1982). Many efforts have been devoted to enhance it in a number of ways, especially concerning speed and reliability of convergence (Haykin, 1994; Hecht-Nielsen, 1990). The backpropagation algorithm serves in general to compute the gradient vector in all the first-order methods, reviewed below.

Neural networks are trained by setting values for the network parameters $\underline{\boldsymbol{w}}$ to minimize an error function $E(\underline{\boldsymbol{w}})$. If this function is quadratic in $\underline{\boldsymbol{w}}$, then the solution can be found by solving a linear system of equations (e.g. with Singular Value Decomposition (Press, Teukolsky, Vetterling & Flannery, 1992)) or iteratively with the delta rule. The minimization is realized by a variant of a *gradient descent* procedure, whose ultimate outcome is a local minimum: a $\underline{\boldsymbol{w}}^*$ from which any infinitesimal change makes $E(\underline{\boldsymbol{w}}^*)$ increase, that may not correspond to one of the global minima. Different solutions are found by starting at different initial states. The process is also perturbed by round-off errors. Given $E(\underline{\boldsymbol{w}})$ to be minimized and an initial state $\underline{\boldsymbol{w}}^0$, these methods perform for each iteration the updating step:

$$\underline{\boldsymbol{w}}^{i+1}=\underline{\boldsymbol{w}}^{i}+\alpha_i\boldsymbol{u}^i \qquad (1)$$

where $\boldsymbol{u}^i$ is the *minimization direction* (the direction in which to move) and $\alpha_i \in R$ is the *step size* (how far to make a move in $\boldsymbol{u}^i$), also known as the *learning rate* in earlier contexts. For convenience, define $\Delta\underline{\boldsymbol{w}}^i=\underline{\boldsymbol{w}}^{i+1}-\underline{\boldsymbol{w}}^i$. Common stopping criteria are:

1. A maximum number of presentations of D (*epochs*) is reached.
2. A maximum amount of computing time has been exceeded.
3. The evaluation has been minimized below a certain tolerance.
4. The gradient norm has fallen below a certain tolerance.

LEARNING ALGORITHMS

Training algorithms may require information from the objective function only, the gradient vector of the objective function or the Hessian matrix of the objective function:

- *Zero-order* training algorithms make use of the objective function only. The most significant algorithms are *evolutionary algorithms*, which are global optimization methods (Goldberg, 1989).
- *First-order* training algorithms use the objective function and its gradient vector. Examples are *Gradient Descent*, *Conjugate Gradient* or *Quasi-Newton* methods, which are all local optimization methods (Luenberger, 1984).
- *Second-order* training algorithms make use of the objective function, its gradient vector and its Hessian matrix. Examples are *Newton's method*

and the *Levenberg-Marquardt algorithm*, which are local optimization methods (Luenberger, 1984).

First-order methods. The *gradient* $\nabla E\underline{\boldsymbol{w}}$ of an s-dimensional function is the vector field of first derivatives of $E(\underline{\boldsymbol{w}})$ w.r.t. $\underline{\boldsymbol{w}}$,

$$\nabla E\underline{\boldsymbol{w}}= \left(\frac{\partial E(\underline{\mathbf{w}})}{\partial w_1},\ldots,\frac{\partial E(\underline{\mathbf{w}})}{\partial w_s}\right) \qquad (2)$$

Here $s=dim(\underline{\boldsymbol{w}})$. A linear approximation to $E(\underline{\boldsymbol{w}})$ in an infinitesimal neighbourhood of an arbitrary point $\underline{\boldsymbol{w}}^i$ is given by:

$$E(\underline{\boldsymbol{w}}) \approx E(\underline{\boldsymbol{w}}^i)+\nabla E\underline{\boldsymbol{w}}(\underline{\boldsymbol{w}}^i)\cdot(\underline{\boldsymbol{w}}-\underline{\boldsymbol{w}}^i) \qquad (3)$$

We write $\nabla E\underline{\boldsymbol{w}}(\underline{\boldsymbol{w}}^i)$ for the gradient $\nabla E\underline{\boldsymbol{w}}$ evaluated at $\underline{\boldsymbol{w}}^i$. These are the first two terms of the Taylor expansion of $E(\underline{\boldsymbol{w}})$ around $\underline{\boldsymbol{w}}^i$. In *steepest* or *gradient* descent methods, this local gradient alone determines the minimization direction $\boldsymbol{u}^i$. Since, at any point $\underline{\boldsymbol{w}}^i$, the gradient $\nabla E\underline{\boldsymbol{w}}(\underline{\boldsymbol{w}}^i)$ points in the direction of fastest increase of $E(\underline{\boldsymbol{w}})$, an adjustment of $\underline{\boldsymbol{w}}^i$ in the negative direction of the local gradient leads to its maximum decrease. In consequence the direction $\boldsymbol{u}^i= -\nabla E\underline{\boldsymbol{w}}(\underline{\boldsymbol{w}}^i)$ is taken.

In conventional steepest descent, the step size α_i is obtained by a *line search* in the direction of $\boldsymbol{u}^i$: how far to go along $\boldsymbol{u}^i$ before a new direction is chosen. To this end, evaluations of $E(\underline{\boldsymbol{w}})$ and its derivatives are made to locate some nearby local minimum. Line search is a move in the chosen direction $\boldsymbol{u}^i$ to find the minimum of $E(\underline{\boldsymbol{w}})$ along it. For this one-dimensional problem, the simplest approach is to proceed along $\boldsymbol{u}^i$ in small steps, evaluating $E(\underline{\boldsymbol{w}})$ at each sampled point, until it starts to increase. One often used method is a divide-and-conquer strategy, also called Brent's method (Fletcher, 1980):

1. Bracket the search by setting three points $a<b<c$ along $\boldsymbol{u}^i$ such that $E(a\boldsymbol{u}^i)>E(b\boldsymbol{u}^i)<E(c\boldsymbol{u}^i)$. Since E is continuous, there is a local minimum in the line joining a to c.
2. Fit a parabola (a quadratic polynomial) to a,b,c.
3. Compute the minimum μ of the parabola in the line joining a to c. This value is an approximation of the minimum of E in this interval.
4. Set three new points a,b,c out of μ and the two points among the old a,b,c having the lowest E. Repeat from 2.

Although it is possible to locate the nearby global minimum, the cost can become prohibitedly high. The line search can be replaced by a fixed step size α, which has to be carefully chosen. A sufficiently small α is required such that $\alpha\nabla E\underline{\boldsymbol{w}}(\boldsymbol{u}^i)$ is effectively very small and the expansion (3) can be applied. A too large value might cause to overshoot or lead to divergent oscillations and a complete breakout of the algorithm. On the other hand, very small values translate in a painfully slow minimization. In practice, a trial-and-error process is carried out.

A popular heuristic is a historic average of previous changes to exploit tendencies and add inertia to the descent, accomplished by adding a so-called *momentum* term $\beta_i\Delta\underline{\boldsymbol{w}}^{i-1}$, where $\Delta\underline{\boldsymbol{w}}^{i-1}$ is the previous weight update (Rumelhart, Hinton & Williams, 1986). This term helps to avoid or smooth out oscillations in the motion towards a minimum. In practice, it is set to a constant value $\beta\in(0.5,1)$. Altogether, for steepest descent, the update equation (1) reads:

$$\underline{\boldsymbol{w}}^{i+1}=\underline{\boldsymbol{w}}^i+\alpha_i\boldsymbol{u}^i+\beta\Delta\underline{\boldsymbol{w}}^{i-1} \qquad (4)$$

where $\boldsymbol{u}^i=-\nabla E\underline{\boldsymbol{w}}(\underline{\boldsymbol{w}}^i)$ and $\Delta\underline{\boldsymbol{w}}^{i-1}=\underline{\boldsymbol{w}}^i-\underline{\boldsymbol{w}}^{i-1}$. This method is very sensitive to the chosen values for α_i and β, to the point that different values are required for different problems and even for different stages in the learning process (Toolenaere, 1990). The inefficiency of the steepest descent method stems from the fact that both $\boldsymbol{u}^i$ and α_i are somewhat poorly chosen. Unless the first step is chosen leading straight to a minimum, the iterative procedure is very likely to wander with many small steps in zig-zag. Therefore, these methods are quite out of use nowadays. A method in which both parameters are properly chosen is the *conjugate gradient*.

Conjugate Gradient. This minimization technique (explained at length in Shewchuck, 1994) is based on the idea that a new direction $\boldsymbol{u}^{i+1}$ should not spoil previous minimizations in the directions $\boldsymbol{u}^i,\underline{\boldsymbol{u}}^{i-1},\ldots,\underline{\boldsymbol{u}}^1$. This is the case if we simply choose $\boldsymbol{u}^i=-\underline{\boldsymbol{g}}^i$, where $\underline{\boldsymbol{g}}^i=\nabla E\underline{\boldsymbol{w}}(\underline{\boldsymbol{w}}^i)$, as was found above for steepest descent. At most points on $E(\underline{\boldsymbol{w}})$, the gradient does *not* point directly towards the minimum. After a line minimization, the new gradient $\underline{\boldsymbol{g}}^{i+1}$ is orthogonal to the line