Approximation techniques for neuromimetic calculus

Vincent Vigneron, Claude Barret

Approximation techniques for neuromimetic calculus

Vincent Vigneron, Claude Barret
Université d'Evry - CEMIF
40 rue du Pelvoux, CE 1455
91020 Evry Courcouronnes Cedex, France
vvigne,cbarret@iup.univ-evry.fr

Abstract

Approximation Theory plays a central part in modern statistical methods, in particular in Neural Network modeling. These models are able to approximate a large amount of metric data structures in their entire range of definition or at least piecewise. We survey most of the known results for networks of neurone-like units. The connections to classical statistical ideas such as ordinary Least Squares *are emphasized.*

Notations

Small boldface letters are used to denote vectors (e.g. $\boldsymbol{u}$), the letter 'T' to denote transposition, and $\|\boldsymbol{x}\|$ to denote the Euclidean norm of a vector $\boldsymbol{x}$. Also, subscripts are used for vector-component indexing (e.g. u_i) and parenthesis for time-indexing (e.g. $\boldsymbol{v}(t)$). All vectors are column vectors. The symbol $\mathbb{E}[\cdot]$ will be used for averages over the set of patterns.

1. Introduction

For recent years, the task of learning from examples has been considered in many cases to be equivalent to multivariate function approximation, that is, to the problem of approximating a smooth function from sparse data, the examples. The interpretation of an approximation scheme in terms of networks has also been extensively discussed [13, 12, 18].

We have now a rather good understanding of simple neural networks, which consist of two sort of objects : the processing units ('neurons, cells') and the weighted connections between the units. The formers make simple computation (summation, thresholding), the latters produce input values for these computations.

For a given task, building a network requires to choose

- the network topologies : number of inputs, connectivity, etc.
- the connection values with respect to the task to learn, through the minimization of an error function E. This error is function of the adaptative parameters in the network, which can conveniently group together into a single weight vector $\boldsymbol{x}$ with components $x_1, \ldots, x_p$.

This highlights the need to optimize the networks in order to archieve the best generalization [2, page 332]. The problem of minimizing continuous functions of many variables is one which has been widely studied, and many of the conventional approaches to this problem are directly applicable to the training of neural networks.

In this article we review several of the most important practical algorithms. We investigate in section 2, some of the simplest of them in more details, and discuss their limitations. We then describe a number of heuristic modifications to gradient descent which aim to improve their performance. Next, in section 3, we review an important extension of conventional optimization algorithms based on the concept of *Ordinary Differential Equation*. Section 4 addresses the problem of supervised learning in layered Neural Network with linear units and includes an analysis of the effect of noise on training algorithms. We survey most of the known results on linear networks. A good familiarity with linear algebra and basic calculus on the part of the reader should be sufficient to follow the paper.

There are many standard monographs which cover linear/non-linear optimization techniques, including Polak [14], Gill & *al.* [8], Fletcher [6]. Most of the algorithms which are described here are ones which have been found to have good performance in a wide range of applications. However, different algorithms will perform best on different problems and it is therefore not possible to recommend a single optimization algorithm.

2. The problem of local minimization

2.1 The gradient rule

For example, consider a function f depending of the variables $(x_1, \ldots, x_n)$. Let X be the variables vector (x_j).

Proposition 1 *If f is two times derivable, a condition for x^* to be a local minimum of f is*

(i) $\frac{\partial f}{\partial x}(x^*) = 0$

(ii) $\frac{\partial^2 f}{\partial x^2}(x^*) > 0$ *(or, in the case this derivative is negative, that the next derivative be non-null and positive).*

■

Proof
Indeed, if x^* is a minimum, we have :

(i) $f(x) - f(x^*) > 0$ for $x \neq x^*$ but close to x^*, and then

$$\frac{f(x) - f(x^*)}{x - x^*} \begin{cases} < 0 & \text{if } x < x^* \\ > 0 & \text{if } x > x^*. \end{cases}$$

Thus, $\lim_{x \to x^*} \frac{f(x) - f(x^*)}{x - x^*} = 0.$

(ii) development of Taylor-Lagrange in the vicinity of x^* gives

$$f(x) = f(x^*) + \frac{\partial f}{\partial x}(x^*)(x - x^*) + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(x^* + \theta(x - x^*))(x - x^*)^2,$$

where $\theta \in]0, 1[$.

Since $\frac{\partial f}{\partial x}(x^*) = 0$, the sign of $f(x) - f(x^*)$ is the same that $\frac{\partial^2 f}{\partial x^2}(x^* + \theta(x - x^*))(x - x^*)^2$ which, for $x - x^* \ll 1$, is the same to that of $\frac{\partial^2 f}{\partial x^2}(x^*)$. Thus $\frac{\partial^2 f}{\partial x^2}(x^*) \geq 0$. It is easy to prove that these conditions are also sufficient. We observe also that $\frac{\partial f}{\partial x}(x) < 0$ if $x < x^*$, $\frac{\partial f}{\partial x}(x) > 0$ if $x > x^*$. Then $\frac{\partial f}{\partial x}(x)$ and $(x - x^*)$ have the same sign that x in the vicinity of x^*, *i.e*

$$(x - x^*)\frac{\partial f}{\partial x}(x) > 0.$$

Hence it seems natural to try to attain x^* from $x^{(0)}$, with the successive approximations

$$x^{(t+1)} = x^{(t)} - \gamma \frac{\partial f}{\partial x}(x^{(t)}), \qquad (1)$$

where γ is a positive "*gain*" parameter. □

Example 2.1 Some real function *Given the real function $f(x) = e^{(x^2)}$, $\frac{\partial f}{\partial x}(x) = xe^{(x^2)}$, then*

$$\begin{aligned} x^{(t+1)} &= (1 - \gamma e^{(\frac{x^2}{2})})x^{(t)} \\ &= x^{(0)} \prod_{k=1}^{t} (1 - \gamma e^{(\frac{x^2}{2})}) \end{aligned}$$

It can be easily seen that, for $0 < \gamma < 2e^{-\frac{x^{(0)2}}{2}}$, the terms series $|x^{(t)}|$ decrease toward zero. △

Suppose now that $x^{(0)} = x^* + \epsilon$. Since $\frac{\partial f}{\partial x}(x^*) = 0$:

$$x^{(1)} = x^{(0)} - \gamma \frac{\partial f}{\partial x}(x^{(0)}) = x^* + \epsilon - \gamma \frac{\partial^2 f}{\partial x^2}(x^* + \theta\epsilon)\epsilon.$$

Then:

$$|x^{(1)} - x^*| = |\epsilon||1 - \gamma \frac{\partial^2 f}{\partial x^2}(x^* + \theta\epsilon)|$$

which deviation is far smaller than $|\epsilon|$ for small values of γ, as $\frac{\partial^2 f}{\partial x^2}(x^* + \theta\epsilon)$ is positive for small values of $|\epsilon|$. Then, x converge toward x^*. The Eq.1 we have rapidly examined define the well-known "gradient rule". These results can be easily extended to the multivariate case.

Proposition 2 *$\boldsymbol{x}^*$ is a local minimum of f if and only if*

(i) $\frac{\partial f}{\partial x_i}(\boldsymbol{x}^*) = 0, \forall i$

(ii) the matrix of second partial derivatives is definite positive, i.e $\forall h = (h_1, \ldots, h_n) \neq \mathbf{0}$:

$$\sum_{i,j} \frac{\partial^2 f}{\partial x_j x_i}(\boldsymbol{x}^*) h_i h_j > 0.$$

■

Proof
The item (i) is trivial. (ii) is solved using a Taylor-Lagrange development at the second order in the vicinity of $\boldsymbol{x}^*$ [5]:

$$\begin{aligned} f(x) = f(x^*) + \sum_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}^*)(x_i - x_i^*) + \\ \frac{1}{2}\sum_{i,j} \frac{\partial^2 f}{\partial x_j x_i}(\boldsymbol{x}^* + \theta(\boldsymbol{x} - \boldsymbol{x}^*))(x_i - x_i^*)(x_j - x_j^*) \end{aligned}$$

where $\theta \in]0,1[$. As previously, it is easy to observe if $\boldsymbol{x}$ is close to $\boldsymbol{x}^*$:

$$\sum_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}^*)(x_i - x_i^*) > 0,$$

which can also be written

$$\boldsymbol{\nabla} f(\boldsymbol{x}) \cdot (\boldsymbol{x} - \boldsymbol{x}^*) > 0,$$

by denoting $\boldsymbol{\nabla} f(\boldsymbol{x})$ the vector $[\frac{\partial f}{\partial x_1}(\boldsymbol{x}), \ldots \frac{\partial f}{\partial x_n}(\boldsymbol{x})]^T$.

Indeed, developping $\frac{\partial f}{\partial x_i}$ yields:

$$\frac{\partial f}{\partial x_i}(\boldsymbol{x}) = \sum_j \frac{\partial^2 f}{\partial x_j x_i}(\boldsymbol{x}^* + \theta_i(\boldsymbol{x} - \boldsymbol{x}^*))(x_i - x_i^*), \theta_i \in]0,1[$$

Hence,

$$\begin{aligned}\boldsymbol{\nabla} f(\boldsymbol{x}) \cdot (\boldsymbol{x} - \boldsymbol{x}^*) &= \sum_i \left(\sum_j \frac{\partial^2 f}{\partial x_j x_i}(\boldsymbol{x}^* + \theta_i(\boldsymbol{x} - \boldsymbol{x}^*)) \right) (x_i - x_i^*)(x_j - x_j^*) \\ &= \sum_{i,j} \frac{\partial^2 f}{\partial x_j x_i}(\boldsymbol{x}^* + \theta(\boldsymbol{x} - \boldsymbol{x}^*))(x_i - x_i^*)(x_j - x_j^*)\end{aligned}$$

which is stricly positive if $\boldsymbol{x}$ is close enough to $\boldsymbol{x}^*$, according to Proposition 2.(ii). We can then use the same type of algorithm in one dimension and propose the following algorithm:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \gamma \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)}). \tag{2}$$

From Eq.2, we obtain:

$$\begin{aligned}\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^*\|^2 &= \left(\boldsymbol{x}^{(t)} - \boldsymbol{x}^* - \gamma \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)})\right) \cdot \\ &\qquad \left(\boldsymbol{x}^{(t)} - \boldsymbol{x}^* - \gamma \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)})\right) \\ &= \|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\|^2 - 2\gamma \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)}) \cdot \\ &\quad (\boldsymbol{x}^{(t)} - \boldsymbol{x}^*) + \gamma^2 \|\boldsymbol{\nabla} f(\boldsymbol{x}^{(t)})\|^2\end{aligned}$$

For small γ, the term with γ^2 is negligible with respect to the term with γ which is negative. Thus, the distance between $\boldsymbol{x}^{(t)}$ and $\boldsymbol{x}^*$ is decreasing with t. Moreover, the function $f(\boldsymbol{x}^{(t)})$ is diminishing:

$$f(\boldsymbol{x}^{(t+1)}) = f\left(\boldsymbol{x}^{(t)} - \gamma \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)})\right).$$

A Taylor development gives :

$$\begin{aligned}f(\boldsymbol{x}^{(t+1)}) &= f(\boldsymbol{x}^{(t)}) - \gamma \|\boldsymbol{\nabla} f(\boldsymbol{x}^{(t)})\|^2 \\ &\quad + \frac{\gamma^2}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_j x_i}(\boldsymbol{x}^{(t)} + \theta\gamma \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)})) \cdot \\ &\qquad \frac{\partial f(\boldsymbol{x}^{(t)})}{\partial x_i} \frac{\partial f(\boldsymbol{x}^{(t)})}{\partial x_j}\end{aligned}$$

Once more, we have proved that, for small γ values, the term with γ^2 is negligible. f is decreasing again. □

We note that such gradient descent is reminescent of the Robbins-Monroe procedure for finding the zero of a regression function.

Example 2.2 *This can be illustrated with a two-variables exemple [7]. Consider the function* $f(\boldsymbol{x}) = e^{x_1^2 + 8x_2^2}$. *The derivatives of* $f(\cdot)$ *with respect to* $\boldsymbol{x}$ *yield* $(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}) = (2x_1 f(\boldsymbol{x}), 16x_2 f(\boldsymbol{x}))$. *Then, Eq.2 gives*

$$\begin{cases} x_1^{(t+1)} &= x_1^{(t)}(1 - 2\gamma f(\boldsymbol{x})) \\ x_2^{(t+1)} &= x_2^{(t)}(1 - 16\gamma f(\boldsymbol{x})) \end{cases}$$

Clearly, if $16\gamma f(\boldsymbol{x}^{(0)})$ *is in* $]0,2[$, *then* $\boldsymbol{x}^{(t)}$ *converge to* $\boldsymbol{x}^* = 0$. *Moreover the decreasing behaviour is different:* $x_1^{(t)}$ *decrease as* $(1-2\gamma f(\boldsymbol{x}))^t$, *by contrast to* $x_2^{(t)}$ *which decrease as* $(1 - 16\gamma f(\boldsymbol{x}))^t$, *which is faster.* △

2.2 Newton method

The algorithms which are described in this paper involve taking a sequence of steps through parameter space: we must decide the direction in which to move and then how far to move in that direction. With simple gradient descent, the direction of each step is given by the local negative gradient of some error function f and the step size is determined by an arbitrary learning parameter. We might expect that a better procedure would be to move along the direction of the negative gradient to find the point at wich f is minimized. This procedure is referred to as line search we now consider in the following.

As it has been seen in the previous section, various convergence speeds can occur depending on $\frac{\partial f}{\partial x_i}$. Suppose $\boldsymbol{\Delta} f(\boldsymbol{x})$ is the second derivative matrix, i.e. $(\frac{\partial^2 f(\boldsymbol{x}^{(t)})}{\partial x_j x_i})$. $\boldsymbol{\Delta} f(\boldsymbol{x})$ is a symetric definite positive matrix, whose eigenvalues are all real and positive. Let $\lambda_1 > \lambda_2 > \ldots > \lambda_n$ the eigenvalues of $\boldsymbol{\Delta} f(\boldsymbol{x}^*)$ and $[\phi_1, \phi_2, \ldots, \phi_n]$ the orthogonal basis associated.

If we choose the starting point of the gradient algorithm near $\boldsymbol{x}^*$ but in the direction of ϕ_i, we obtain:

$$\boldsymbol{x}^{(0)} = \boldsymbol{x}^* + \epsilon\phi_i, \quad \epsilon \ll 1.$$

At the next step:

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} - \gamma \boldsymbol{\nabla} f(\boldsymbol{x}^* + \epsilon\phi_i).$$

Let us compute $\boldsymbol{x}^{(1)}$ with a second-order Taylor development (see [7]). Neglecting the term with γ^2, we get:

$$\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^*\|^2 = \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|^2 - 2\gamma \boldsymbol{\nabla} f(\boldsymbol{x}^* + \epsilon\phi_i)\epsilon\phi_i.$$

If ϵ is small enough, we have:

$$\boldsymbol{\nabla} f(\boldsymbol{x}^* + \epsilon\phi_i) \approx \boldsymbol{\nabla} f(\boldsymbol{x}^*) + \boldsymbol{\Delta} f(\boldsymbol{x}^*)\epsilon\phi_i = \boldsymbol{\Delta} f(\boldsymbol{x}^*)\epsilon\phi_i.$$

hence, for small ϵ and γ, and using the definition of eigendecomposition, $\boldsymbol{\Delta} f(\boldsymbol{x}^*) \cdot \phi_i = \lambda_i \phi_i$, we obtain:

$$\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^*\|^2 \approx \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|^2 - 2\gamma\epsilon^2\lambda_i\|\phi_i\|^2,$$

then, since $\|\phi_i\|^2 = 1$:

$$\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^*\|^2 \approx \epsilon^2(1 - 2\gamma\lambda_i).$$

It is clear that $\boldsymbol{x}^{(t)}$ converge towards $\boldsymbol{x}^*$ with a speed in proportion of the eigenvalue associated with the chosen direction. In particular, in the case of strong disparity between the eigenvalues, it can lead to a very bad convergence.

To assure a "good" rate of convergence, it is desirable to choose a modified gradient algorithm, the *Newton method.*

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \gamma \boldsymbol{\Delta} f(\boldsymbol{x}^{(t)})^{-1} \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)}), \qquad (3)$$

which, in the vicinity of $\boldsymbol{x}^*$, behaves in a homogeneous manner in all the directions.

2.3 Inexact gradient and Newton method

Newton method is computational prohibitive since it would require $\mathcal{O}(n^3)$ operations to inverse the Hessian matrix (n being the number of parameters). Alternative approaches, known as *quasi-Newton*, build up an approximation to the inverse Hessian over a number of steps.

The quasi-Newton approach involves generating a sequence of matrices M which represent increasingly accurate approximations to the inverse Hessian $(\boldsymbol{\Delta} f)^{-1}$ using only information on the first derivatives of the error function. From the Newton formula (3), we see that the weight vectors at steps t and $t+1$ and related to the corresponding gradients by

$$\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)} = -\boldsymbol{\Delta} f(\boldsymbol{x}^{(t)})^{-1}(\boldsymbol{\nabla} f(\boldsymbol{x}^{(t+1)}) - \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)}))$$

which is known as the quasi-Newton condition. The approximation M of the inverse Hessian is computed so as to satisfy this condition also. The two most commonly used update formulae are the *Davisdon-Fletcher-Powell* (DFP) and the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) procedures. Full description approach is beyond the scope of this paper, but it is well established that DFP or BFGS are extremely robust with respect to gradient errors. In practice, these approaches work better when $\boldsymbol{x}^{(0)}$ is sufficiently close to the problem solution, but may not work well otherwise.

Proposition 3 *Let δ^* be the Newton direction and $\tilde{\delta}$ the direction computed by the inexact gradient $g^{(t)} = (\boldsymbol{\nabla} f(\boldsymbol{x}^{(t+1)}) - \boldsymbol{\nabla} f(\boldsymbol{x}^{(t)}))$. At step 0, $\tilde{\delta}$ is a rotation of δ^* through the angle θ_δ.*

Proof
Define a quadratic function $f(x) = \frac{1}{2}\boldsymbol{x}^T A \boldsymbol{x}$ where A is a positive definite matrix defined by

$$A = \frac{1}{2}\begin{pmatrix} \zeta + \zeta^{-1} & \zeta - \zeta^{-1} \\ \zeta - \zeta^{-1} & \zeta + \zeta^{-1} \end{pmatrix}.$$

Select $\mu \in [0; 1]$ and define $\boldsymbol{x}^{(0)} = \frac{\sqrt{2}}{2}\begin{pmatrix} 1 - \mu\zeta \\ 1 + \mu\zeta \end{pmatrix}$. Then, we have the following vectors:

$$\begin{aligned} f(\boldsymbol{x}^{(0)}) &= \frac{\epsilon}{2}(1 + \mu^2), \boldsymbol{\nabla} f(\boldsymbol{x}^{(0)}) \\ &= (A + A^T)\boldsymbol{x}^{(0)} = \frac{\sqrt{2}}{2}\begin{pmatrix} \zeta - \mu \\ \mu + \zeta \end{pmatrix} \end{aligned}$$

and $\boldsymbol{\Delta} f(\boldsymbol{x}^{(0)}) = A$.

Using these definitions, one can easily establish that the Newton direction is

$$\begin{aligned} \delta^* &= -\boldsymbol{\Delta} f(\boldsymbol{x}^{(0)})^{-1} \boldsymbol{\nabla} f(\boldsymbol{x}^{(0)}) \\ &= -\frac{\sqrt{2}}{2}\begin{pmatrix} 1 - \zeta\mu \\ 1 + \mu\zeta \end{pmatrix} = -\boldsymbol{x}^{(0)}, \end{aligned}$$

hence, $\boldsymbol{x}^{(0)} + \delta^*$ is the exact solution $\forall\zeta$. However, the quasi-Newton direction actually computed using the inexact gradient $g^{(0)}$ is[1]:

$$\begin{aligned} \tilde{\delta} = -A^{-1}g^{(0)} &= \frac{1}{2}\begin{pmatrix} \zeta + \zeta^{-1} & -\zeta + \zeta^{-1} \\ -\zeta + \zeta^{-1} & \zeta + \zeta^{-1} \end{pmatrix}\frac{\sqrt{2}}{2}\begin{pmatrix} -\zeta - \mu \\ \mu + \zeta \end{pmatrix} \\ &= -\frac{\sqrt{2}}{2}\begin{pmatrix} -1 - \zeta\mu \\ \mu\zeta - 1 \end{pmatrix}. \end{aligned}$$

Notice that $\|\boldsymbol{\nabla} f(\boldsymbol{x}^{(0)})\| = \|g^{(0)}\|$ and $\|\delta^*\| = \|\tilde{\delta}\|$. More specifically, $g^{(0)}$ can be considered as a rotation of

[1] Initializing the procedure using the identity matrix corresponds to taking the first step in the direction of the negative gradient. At each step of the algorithm, the direction Mg is guaranteed to be a descent direction, since the matrix M is positive definite.

$\nabla f(\boldsymbol{x}^{(0)})$ through an angle $\theta_g = \cos^{-1}\left(\frac{\mu^2-\zeta^2}{\mu^2+\zeta^2}\right)$ while $\tilde{\delta}$ can be considered as a rotation of δ^* through an angle $\theta_\delta = \cos^{-1}\left(\frac{\mu^2\zeta^2-1}{\mu^2\zeta^2+1}\right)$.

Further, it comes that the angle between $\tilde{\delta}$ and $\nabla f(\boldsymbol{x}^{(0)})$ is:

$$\eta = \cos^{-1}\left(\frac{(\mu^2-\zeta^2)(\mu^2\zeta^2-1)-4\mu^2\zeta^2}{(\mu^2+\zeta^2)(\mu^2\zeta^2+1)}\right)$$

Notice that as ζ approaches zero, the matrix A becomes successively more ill-conditioned and errors in $g^{(t)}$ are magnified by successively greater factors when used in the computation of the search direction $\tilde{\delta}$. □

2.4 The gradient descent method

It could seem that what the previous algorithm solve any minimisation problem of a regular fonction. In pratice, Δf and ∇f are not easily/directly computable. Sometimes, in the neural networks theory, ∇f is not easily computable but, we can observe easily a random variable, say $G(\boldsymbol{\xi}, \boldsymbol{x})$, whose average value is ∇f:

$$\mathbb{E}[G(\boldsymbol{\xi}, \boldsymbol{x})] = \nabla f.$$

Let us examine the algorithm :

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \gamma G(\boldsymbol{\xi}, \boldsymbol{x}^{(t)}). \tag{4}$$

One can immediatly note that the condition $\gamma \to 0$ is necessary[2] to stabilize the algorithm on $\boldsymbol{x}^*$. Suppose $\gamma = \gamma^{(t)}$ decrease towards zero. Eq.4 is

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \gamma^{(t+1)} G(\boldsymbol{\xi}^{(t+1)}, \boldsymbol{x}^{(t)}). \tag{5}$$

Suppose also t and T fixed, but with large values:

$$\begin{aligned}\boldsymbol{x}^{(t+T)} &= \boldsymbol{x}^{(t)} - \sum_{k=1}^{T} \gamma^{(t+k)} G(\boldsymbol{\xi}^{(t+k)}, \boldsymbol{x}(t+k-1)) \\ &= \boldsymbol{x}^{(t)} - (\sum_{j=1}^{T} \gamma^{(t+j)}) \cdot \\ &\quad \frac{1}{\sum_{j=1}^{T} \gamma^{(t+j)}} \sum_{k=1}^{T} \gamma^{(t+k)} G(\boldsymbol{\xi}^{(t+k)}, \boldsymbol{x}(t+k-1)).\end{aligned}$$

[2] Except if the variance $\mathbb{V}[G(\boldsymbol{\xi}, \boldsymbol{x})] \to \mathbf{0}$, which never occurs.

For very small values $\gamma^{(t+k)}$ and if the law of $G(\cdot)$ is not very sensitive to slow variations of $\boldsymbol{x}$, we have (by approximation):

$$\begin{aligned}\boldsymbol{x}^{(t+T)} &\approx \boldsymbol{x}^{(t)} - (\sum_{j=1}^{T} \gamma^{(t+j)}) \cdot \\ &\quad \frac{1}{\sum_{j=1}^{T} \gamma^{(t+j)}} \sum_{k=1}^{T} \gamma^{(t+k)} G(\boldsymbol{\xi}^{(t+k)}, \boldsymbol{x}^{(t)}).\end{aligned}$$

However, if $\sum_{j=1}^{T} \gamma^{(t+j)} \gg 1$ (for large T) and the variables $G(\boldsymbol{\xi}^{(t+k)}, \boldsymbol{x}^{(t)})$ are independent, we can apply the Large Numbers law:

$$\frac{1}{\sum_{j=1}^{T} \gamma^{(t+j)}} \sum_{k=1}^{T} \gamma^{(t+k)} G(\boldsymbol{\xi}^{(t+k)}, \boldsymbol{x}^{(t)}) \approx \mathbb{E}[G(\boldsymbol{\xi}, \boldsymbol{x})] = \nabla f.$$

This can be shown by examining the variance of the deviation $\boldsymbol{x}^{(t+T)} - \boldsymbol{x}^{(t)}$:

$$\frac{1}{\sum_{j=1}^{T} \gamma^{(t+j)^2}} \sum_{k=1}^{T} \gamma^{(t+k)^2} \mathbb{V}[G(\boldsymbol{\xi}^{(t+k)}, \boldsymbol{x}^{(t)})].$$

which is very small if $\sum_{k=1}^{\infty} \gamma^{(k)^2}$ converge. Thus, assuming (i) $\sum_{k=1}^{\infty} \gamma^{(k)} = \infty$ and (ii) $\sum_{k=1}^{\infty} \gamma^{(k)^2} < \infty$, the algorithm in Eq.5 converge [5]. In other words, We are assured of convergence, if the learning rate parameter γ is made to decrease at each step of the algorithm in accordance with the requirements of the theorem [11]. These can be satisfied by choosing $\gamma^{(k)} \propto \frac{1}{t}$ although such a choice leads to very slow convergence[3] and very long computation times.

Example 2.3 *Here, we demonstrates the minimization of the following function in* $\mathbb{R}^2$*:*

$$J(\boldsymbol{x}) = (p + x_1^2 - x_2^2)^2(\frac{1}{p} - x_1^2) + p(1-x_2)^2$$

where p is a positive parameter. Because of the way the curvature bends around the origin, such function is notoriously difficult in optimization examples because of the slow convergence with which most methods exhibit when trying to solve this problem. This point is due to the ill-conditioned Hessian matrix. This function has a unique minimum at the point $\boldsymbol{x} = [\frac{1}{p}, 1]$ where $f(\boldsymbol{x}) = 0$. We demonstrate in Fig.1 a number of techniques for its minimization starting at the point $\boldsymbol{x} = [-2; 0, 5]$. BFGS and Newton find easily the right

[3] In most artcle, for practical reasons, a constant value of γ is often used as this generally leads to (seemingly) better results, but guarantee of convergence is lost.

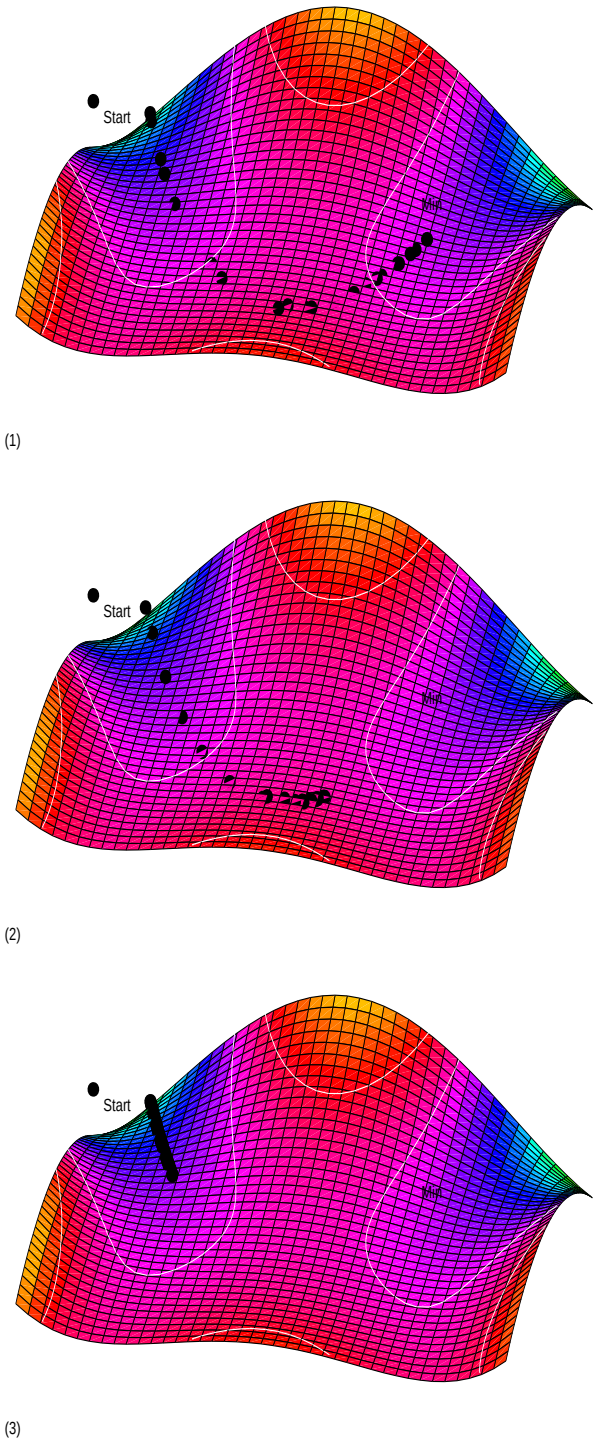

Figure 1: $\mathbb{R}^2$ Function minimization. (1) BFGS method (25 iterations), (2) Newton method (40 iterations), (3) Steepest Descent method with constant step (no convergence in 250 iterations).

search direction (Fig.1.1) while in constrast the Steepest Descent fail to find the minimum (Fig.1.3). △

An important advantage of the sequential approach over "*batch*" methods arises if there is a high degree of redundant information in the data set [2]. The gradient descent method updates the weigths after each pattern presentation, and so will be unaffected by the replication of data.

3. Stochastic approximation algorithms

The results of §2.4 can be generalized in order to study other algorithms proposed in the Theory of Neural Networks [17]. By contrast, $G(\boldsymbol{\xi}, \boldsymbol{x})$ is no more the opposite of a "*random gradient*" but more generaly a "*random function*" of $\boldsymbol{x}$. this function denoted $h(\boldsymbol{x})$ is supposed regular, valued in $\mathbb{R}^n$. We have:

$$\mathbb{E}[G(\boldsymbol{\xi}, \boldsymbol{x})] = h(\boldsymbol{x}).$$

3.1 Ordinary Differential Equation

The Eq.4 can also be written

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \gamma^{(k)} G(\boldsymbol{\xi}^{(k)}, \boldsymbol{x}^{(t)}) + {\gamma^{(k)}}^2 \mathcal{O}(\boldsymbol{\xi}^{(k)}, \boldsymbol{x}^{(t)}).$$

where $\mathcal{O}(\cdot)$ is an error term (negligeable in most of the cases) and $\gamma^{(k)}$ such that $\sum_{k=1}^{\infty} \gamma^{(k)} = \infty, \sum_{k=1}^{\infty} {\gamma^{(k)}}^2 < \infty$. Hence,

$$\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)} \approx \gamma^{(t+1)} G(\boldsymbol{\xi}^{(t+1)}, \boldsymbol{x}^{(t)}).$$

Simple induction shows that:

$$\boldsymbol{x}^{(t+T)} - \boldsymbol{x}^{(t)} \approx \left(\sum_{k=1}^{T} \gamma^{(t+k)} \right) h(\boldsymbol{x}), \qquad (6)$$

which shows that the variation on $\boldsymbol{x}$ between t and $t+T$ si proportional to $h(\boldsymbol{x})$.

Let us introduce a new scale of time:

$$u^{(t)} = (\sum_{k=0}^{t} \gamma^{(k)}),$$

then $\lim_{t \to \infty} u^{(t)} = \infty$. Suppose also that

$$\chi(u^{(t)}) = \boldsymbol{x}^{(t)}$$

i.e. at time $u^{(t)}$, χ take the value $\boldsymbol{x}^{(k)}$. The equation 6 gives:

$$\chi(u^{(t+T)}) - \chi(u^{(t)}) \approx (u^{(t+T)} - u^{(t)}) h(\chi(u^{(t)})).$$

i.e. in a simpler form:

$$\frac{\partial \chi}{\partial u} = h(\chi(u^{(t)})). \tag{7}$$

The behaviour of $\boldsymbol{x}$ can be compared with the solution of a ("rescaled") differential equation. this equation is usually called the *Ordinary Differential Equation* (ODE) [5].

The particularity of the ODE is that for t very large, the law of $\boldsymbol{x}$ is uniformly close to the law of the ODE solution. Moreover, the trajectory of $\boldsymbol{x}$ is close to the trajectory of the ODE solution. In conclusion, the study of these algorithms is comparable to the associated ODE.

3.2 Lyapounov function and attractor

In the case of a gradient descent, the associated ODE is:

$$\frac{\partial \boldsymbol{x}}{\partial u} = h(\boldsymbol{x}) = -\boldsymbol{\nabla} f(\boldsymbol{x}),$$

for simplicity, let us take again $\boldsymbol{x}$. The local minima $\boldsymbol{x}^*$ of f are constant solutions. Moreover, such a point $\boldsymbol{x}^*$ is an attractor. Indeed, the differentiation of f for a solution $\boldsymbol{x}$ of the ODE is given by the *chain rule*:

$$\frac{\partial f}{\partial u}(\boldsymbol{x}) = \frac{\partial f}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial u} = \boldsymbol{\nabla} f(\boldsymbol{x}) h(\boldsymbol{x}) = -\|\boldsymbol{\nabla} f(\boldsymbol{x})\|^2.$$

f decrease along the trajectories of the ODE. $\boldsymbol{x}(u)$ converge toward a point $\boldsymbol{x}^*$. More generaly, it would be useful to caracterise the points $\boldsymbol{x}^*$, which verify necessarily:

$$h(\boldsymbol{x}^*) = 0.$$

Let $\boldsymbol{x}^*$ be a zero of $h(\cdot)$. $\Gamma(\boldsymbol{x}^*)$ is a domain of attraction for $\boldsymbol{x}^*$ if, when $\boldsymbol{x}^{(0)}$ is in $\Gamma(\boldsymbol{x}^*)$:

(i) $\boldsymbol{x}^{(t)}$ stay in $\Gamma(\boldsymbol{x}^*)$ and $\lim_{t\to\infty} = \boldsymbol{x}^*$.

(ii) When $\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|$ is small, $\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\|$ is small $\forall t$.

Such points are called "*attractors*" for Eq.6. However, zeros can not be easily shown to be attactors for most functions $h(\cdot)$. It can be done indirectly using a Lyapounov function: it is a fuction V, continuously differentiable and definite on a vicinity Γ of $\boldsymbol{x}^*$ such that

(i) $V(\boldsymbol{x}^*) = 0$ and $V(\boldsymbol{x}) > 0$ if $\boldsymbol{x} \neq \boldsymbol{x}^*$.

(ii) If Γ is not bounded, $\lim_{\boldsymbol{x}\to\infty} V(\boldsymbol{x}) = \infty$.

(iii) $\boldsymbol{\nabla} V(\boldsymbol{x}) \cdot h(\boldsymbol{x})$ if $\boldsymbol{x} \neq \boldsymbol{x}^*$.

We have the following result

Proposition 4 *If such a Lyapounov function exists,* Γ *is an attractor domain for* $\boldsymbol{x}^*$. ■

Proof
Let $\psi(u) = V(\boldsymbol{x}(u))$, the derivative of ψ is using again chain-rule:

$$\frac{\partial \psi}{\partial u}(u) = \boldsymbol{\nabla} V(\boldsymbol{x}(u)) h(\boldsymbol{x}(u)) < 0,$$

and $\frac{\partial \psi}{\partial u}(u)$ is non-zero except in $\boldsymbol{x}^*$. Hence, ψ is a strictly decreasing function, and in the case Γ is not bounded, the limit of $\psi(u)$ exists when $u \to \infty$. This limit is zero and $\boldsymbol{x}(u) \to \boldsymbol{x}^*$ or this limit is strictly positive which is impossible.

Indeed, in this case $\boldsymbol{x}(u)$ is moving in a crown around $\boldsymbol{x}^*$, but not so close to $\boldsymbol{x}^*$:

$$\begin{aligned} &\exists a > 0 \text{ s.t. } \|\boldsymbol{x}(u) - \boldsymbol{x}^*\| \geq a \\ &\Rightarrow \exists b > 0 \text{ s.t. } \boldsymbol{\nabla} V(\boldsymbol{x}(u)) h(\boldsymbol{x}(u)) < -b \\ &\Rightarrow \psi(\boldsymbol{x}(0)) - bt \geq \psi(t). \end{aligned}$$

Moreover,

$$\forall c > 0, \exists d \text{ s.t. } \|\boldsymbol{x}(u) - \boldsymbol{x}^*\| < d \Rightarrow V(\boldsymbol{x}) < c.$$

Since ϕ is a decreasing function, if $\boldsymbol{x}(0)$ is near $\boldsymbol{x}^*$, $V(\boldsymbol{x}(u))$ stay small, i.e. $\boldsymbol{x}(u)$ is close to $\boldsymbol{x}^*$.

□

Example 3.1 *Let us consider the differential equation in* $\mathbb{R}^2$ *[7]*

$$\begin{cases} \frac{\partial x_1}{\partial u} &= -2x_1(e^{(x_1^2+x_2^2)} - e^{x_2}) \\ \frac{\partial x_2}{\partial u} &= -2x_2(e^{(x_1^2+x_2^2)} - e^{x_1}) \end{cases}$$

which can be written:

$$\frac{\partial \boldsymbol{x}}{\partial u} = -\boldsymbol{\nabla} f(\boldsymbol{x}) - g(\boldsymbol{x}),$$

with $f(\boldsymbol{x}) = e^{(x_1^2+x_2^2)}$, $\boldsymbol{\nabla} f = [\frac{\partial f}{\partial \boldsymbol{x}_1}, \frac{\partial f}{\partial \boldsymbol{x}_2}]$ *and* $g(\boldsymbol{x}) = (2x_1 e^{x_2}, 2x_2 e^{x_1})$. *We have:*

$$\begin{aligned} \frac{\partial \|f(\boldsymbol{x}) - 1\|^2}{\partial u} &= 2(f(\boldsymbol{x}) - 1)[\boldsymbol{\nabla} f(\boldsymbol{x})^T(-\boldsymbol{\nabla} f(\boldsymbol{x}) - g(\boldsymbol{x}))] \\ &= 2(f(\boldsymbol{x}) - 1)[-\|\boldsymbol{\nabla} f(\boldsymbol{x})\|^2 - \boldsymbol{\nabla} f(\boldsymbol{x})^T g(\boldsymbol{x}))]. \end{aligned}$$

Since $(f(\boldsymbol{x}) - 1) > 0$, *we just have to verify that*

$$\boldsymbol{\nabla} f(\boldsymbol{x})^T g(\boldsymbol{x}) > 0.$$

It is easy to show that:

$$\nabla f(\boldsymbol{x})^T g(\boldsymbol{x}) = 4f(\boldsymbol{x})[x_1^2 e^{x_2} + x_2^2 e^{x_1}] > 0$$

Then, $V(\boldsymbol{x}) = \|f(\boldsymbol{x}) - 1\|^2$ is a Lyapounov function for this differential equation, the point $(0,0)$ is the only attractor of $\mathbb{R}^2$ since $V(\boldsymbol{x}) \to \infty$ when $\boldsymbol{x} \to \infty$. $\triangle$

3.3 Remarks concerning constant-step algorithms

For an algorithm such that the learning rate γ in Eq.8 is constant:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \gamma G(\boldsymbol{\xi}^{(k)}, \boldsymbol{x}^{(t)}) \tag{8}$$

we cannot talk of real convergence since the variance of $\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}$ stays around γ^2. For very small values of γ, the behaviour of this algorithm is comparable the one of an algorithm with decreasing rate. The difference is in the case $\boldsymbol{x}^*$ is an ODE attractor, then $\boldsymbol{x}^{(t)}$ don't converge toward $\boldsymbol{x}^*$ but is rapidly oscillating around $\boldsymbol{x}^*$. Under the effect of a succession of hasards, it can escape the vicinity of $\boldsymbol{x}^*$. These phenomenons are not under the scope of this article and won't be detailled here. The results with constant-step algorithms should be considered valuable only in the mean term, not at the long term.

In practice, those algorithms are the most often used for the following reason : if under some effects, the objective (*i.e.* the attractor $\boldsymbol{x}^*$) is slowly moving through time, a constant-step algorithm is capable to follow it without much difficulties. By contrast a decreasing step algorithm which can loose its target when γ is not so small.

4. Linear calculus, optimization or regularization

A prominent feature of modern Artificial Neural Network classifiers is the nonlinear aspects of neural computation. So why bother with linear networks ? Nonlinear computations are obviously crucial but, by focusing on these arguments we miss subtle aspects of dynamic, structure and organization that arise in the network during training. Furthermore, general results in the nonlinear case are rare or impossible to derive analytically. One often forgets by instance that when learning starts with small random initial weights the networks *is operating in its linear regime*[4] [1]. Finally, the study of linear networks leads to some interesting questions and paradigms which could not have been guessed by advance.

In classical statistical discriminant analysis by example, to discriminate between two pattern classes, one uses a discriminant function

$$g(\boldsymbol{x}) = w^T \boldsymbol{x} + w_0, \tag{9}$$

where,

- input vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ in d-dimensions are given with a corresponding set of m-dimensional target vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{x}_N$.
- w_0, $\boldsymbol{w} = (w_1, \ldots, w_d)$ are the weights of the Fisher discriminant fonction (DF)

  $$w = S^{-1}\left(\bar{x}^{(1)} - \bar{x}^{(2)}\right), \quad w_0 = -\frac{1}{2} w^T \left(\bar{x}^{(1)} + \bar{x}^{(2)}\right)$$

- $x_j^{(i)}$ is the j-th pattern vector from the i-th class,
- where the sample covariance matrix is

  $$S = \frac{1}{n_1 + n_2 - 2} \sum_{i=1}^{2} \sum_{j=1}^{n_i} \left(x_j^{(i)} - \bar{x}^{(i)}\right)\left(x_j^{(i)} - \bar{x}^{(i)}\right)^T$$

  and $\bar{x}^{(i)} = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j^{(i)}$ a mean sample vector.

When one omits the covariance matrix S, one has the Euclidean distance Classifier [4]. The neural network can itself form linear discriminant hyperplane in a high-dimensional feature space and discriminate complicated objects. Gallinari *et al.* [3] and Koford [10] show that when $n_1 = n_2$, training of cost function with f linear (which become *ADALINE*, a prototype of modern SLP) leads to a weight vector w equivalent to the weights of standard Fisher DF, which is asymptotically optimal when classes are gaussian with common covariance matrix. Raudys [16] indicated that if $n_1 = n_2$ and with whitened data such that $\mathbb{E}[\boldsymbol{x}] = \mathbb{E}[\boldsymbol{y}] = 0$) [15], after the first back-propagation learning step in a batch mode of the SLP, one obtains the weights equivalent to the *Euclidean Distance Classifier*. Lets onsider a two layered linear network which computes the linear function $\boldsymbol{y} = \zeta\xi\boldsymbol{x} = A\boldsymbol{x}$ (Fig.2).

As usual, we assume that a set of d-dimensional input patterns/vectors $x_1, x_2, \ldots, x_N$ is given with a correponding set of m-dimensional target vectors $y_1, \ldots, y_N$. The patterns are whitened (*i.e.* $\mathbb{E}[\boldsymbol{x}] = \mathbb{E}[\boldsymbol{y}] = 0$) [15]. The problem of linear regression can be stated in the following manner : *find an $d \times m$ matrix A which minimizes the L_2 loss function*

$$L_2 = \text{trace}[(\boldsymbol{y} - A\boldsymbol{x})^T(\boldsymbol{y} - A\boldsymbol{x})],$$

[4] Several authors defend the idea that even when training is completed several units in the networks are operating in their linear range

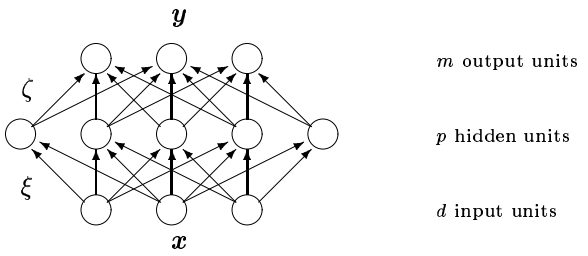

Figure 2: The basic Neural Network with an $d - p - m$ architecture comprising one input layer, one hidden layer and one output layer with n, p, m units respectively.

given $\boldsymbol{y}$ and $\boldsymbol{x}$. A minimum will always exist, since L_2 is a convex differentiable function. Learning algorithms consist in slowly altering the connection weights to archieve this minimization. L_2 is continuous, differentiable and bounded below by zero and therefore it must reach its minimum for a matrix satisfaying $A\Sigma_{xx} = \Sigma_{yx}$, where $\Sigma_{xx} = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T]$ and $\Sigma_{yx} = \mathbb{E}[\boldsymbol{y}\boldsymbol{x}^T]$ are the variance-covariance matrix. In the case where Σ_{xx} is positive definite, the unique optimal A is given by

$$A = \Sigma_{yx}\Sigma_{xx}^{-1}. \tag{10}$$

The derivative of L_2 with respect to the weights A is

$$\nabla L_2(A) = (2A\Sigma_{xx} - 2\Sigma_{yx}).$$

Thus, the gradient descent learning rule can be expressed as $A^{(k+1)} = A^{(k)} - \eta\nabla L_2(A)$, where η is the constant learning rate. After the first learning iteration for the weights A, one obtains

$$\begin{aligned} A^{(1)} &= A^{(0)} - \eta\nabla L_2(A) = A^{(0)} - \eta(2A^{(0)}\Sigma_{xx} - 2\Sigma_{yx}) \\ &= A^{(0)}(I - 2\eta\Sigma_{xx}) - (I - (I - 2\eta\Sigma_{yx})) \\ &= A^{(0)}G_{xx} - (I - G_{yx}), \end{aligned}$$

where $G_{xx} = I - 2\eta\Sigma_{xx}$ and $G_{yx} = I - 2\eta\Sigma_{yx}$. Now we will analyse the changes of the weight vector in the second and following iterations :

$$\begin{aligned} A^{(2)} &= A^{(1)} - \eta(2A^{(1)}\Sigma_{xx} - 2\Sigma_{yx}) \\ &= \left(A^{(0)}G_{xx} - (I - G_{yx})\right)(I - 2\eta\Sigma_{xx}) - (I - G_{yx}) \\ &= A^{(0)}G_{xx}^2 - (I - G_{yx})(I + G_{xx}), \end{aligned}$$

and further, simple induction shows that

$$A^{(k)} = A^{(0)}G_{xx}^k - \Sigma_{yx}\Sigma_{xx}^{-1}(I - G_{yx}^k)$$

which can also be written

$$A^{(k)} = A^{(0)}(I - 2\eta\Sigma_{xx})^k - \Sigma_{yx}\Sigma_{xx}^{-1}\left(I - (I - 2\eta\Sigma_{xx})^k\right). \tag{11}$$

By definition matrix Σ_{xx} is not singular, so it has an inverse. *Stopped learning* stands in $k \ll \infty$. Using the first terms of the expansion $(I - 2\eta\Sigma_{xx})^k = I - 2k\eta\Sigma_{xx} + \frac{k(k-1)}{2}(2\eta)^2\Sigma_{xx}^2 - \dots$ in (11) for small η and k and results in

$$A^{(k)} \approx A^{(0)} - 2\eta k(I - \eta(k-1)\Sigma_{xx})(I + A^{(0)}\Sigma_{xx})$$

Further $(I - \beta\Sigma_{xx})^{-1} = I + \beta\Sigma_{xx} + \dots$ with supposition that η and k are small gives

$$\begin{aligned} A^{(k)} \approx A^{(0)} - 2\eta k(\Sigma_{yx} + A^{(0)}\Sigma_{xx}) \times \\ \times (I - \eta(k-1)\Sigma_{xx})^{-1}. \end{aligned} \tag{12}$$

When the prior weights are very small one shall assume $A^{(0)} = 0$. Then we obtain from (12) :

$$A^{(k)} = -2\eta k\lambda\Sigma_{yx}\left(\Sigma_{xx} + \lambda I\right)^{-1}, \tag{13}$$

with $\lambda = -\frac{1}{\eta(k-1)}$. Equations (13) and (10) are equivalent when k, the number of iterations, increase. Compare with the definition of the *ridge* estimate $\hat{A}(\alpha) = \Sigma_{yx}(\Sigma_{xx} + \alpha I)^{-1}$, $\alpha \geq 0$ proposed by Hoerl and Kennard [9], the iterative scheme of a linear Neural Network is equivalent to computing the LS ridge estimate of the covariance $(\Sigma_{xx} + \lambda I)$ instead of the usual Σ_{xx}. Of course $\hat{A}^{(0)}$ is the ordinary LS estimate.

Our analysis links *Stopped Learning* in linear neural networks to classical statistical techniques in general, and *Ridge estimation* in particular.

5. Epilog

This article is the occasion to familiarize with the "use" of optimisation and approximation techniques in neural networks computation. Even propositions are not strongly precise and don't pretend to be exhaustive, the methods and results constitute the essential of these well-known algorithms. The use of these algorithms overtakes the neuromimetic context.

Much debate has occured over past years between advocates of linesearch techniques and devotees of gradient descent. Some arguments center upon simplicity

or elegance which are non-scientific ones. More tangible are arguments such as linesearch codes are cited as superior with respect to scale invariance and with respect to cost-per-iteration, while others are often regarded as superior for nonconvex problems. However, when computing a search direction with a linesearch method, a very small amount of error in the computed gradient may result in a computed search almost diametrically opposite the desired direction. Although the example presented in §2.3 page 4 represents a extreme case of bad conditioned matrix, linesearch algorithms are nevertheless in principe highly vulnerable to the slightest inaccuracies or noise in gradient evaluations.

References

[1] P. Baldi and K. Hornik. Learning in linear neural networks : A survey. *IEEE Transactions on Neural Networks*, 6(4), July 1995.

[2] C. Bishop. *Neural networks for pattern recognition.* Clarendon Press, 1885.

[3] L. Bottou and P. Gallinari. A framework for the cooperation of learning algorithms. In *Neural Information Processing Systems*, pages 781–788, San Mateo, 1991. Morgan Kaufmann.

[4] O. Duda and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, second edition, 1973.

[5] M. Duflo. *Méthodes récursives aléatoires.* Masson, 1990.

[6] R. Fletcher. *Practical methods in optimization.* John Wiley, New-York, second edition, 1987.

[7] J. Fort. Bases mathématiques pour les réseaux de neurones artificiels. unpublished, 1994. Chapitre 4.

[8] P. Gill, W. Murray, and M. Wright. *Practical optimisation.* Academis Press, 1981.

[9] A. Hoerl and R. Kennard. Ridge regression : Biased estiomation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

[10] J. Koford and G. Groner. The use of an adaptative threshold element to design a linear optimal pattern classifier. *IEEE Trans. on Inf. Theory*, 12(1):42–50, 1966.

[11] Z. luo. On the convergence of the lms algorithm with adaptative larning rate for linear feedforward networks. *Neural Computation*, 3(2):226–245, 1991.

[12] J. Moody. The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems. In S. H. J. Moody and R. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 847–854, Paolo Alto, CA, 1989. Morgan Kaufmann.

[13] T. Poggio and F. Girosi. A theory of networks for approximation and learning. A.I. memo 1140, Massachusetts Institute of Technology, 1989.

[14] E. Polak. *Computational Methods in Optimisation: a unified approch.* Academis Press, New-York, 1971.

[15] C. Rao and H. Toutengurg. *Linear Models. Least Squares and Alternatives.* Springer series in Statistics. Springer, Berlin, 1996.

[16] S. Raudys and T. Cibas. Regularization by early stopping in single layer perceptron training. In J. V. C. Von der Malsburg, W. von Seelen and B. Sendhoff, editors, *Proceedings Int. Conf. on Artificial Neural Networks*, pages 77–82. Springer, 1995.

[17] V. Vigneron. *Méthodes d'apprentissage statistiques et problèmes inverses. Applications à la spectrographie.* Thèse de doctorat, Université d'Evry, May 1997.

[18] H. White. Learning in artificial neural networks: a statistical perspective. *Neural Computation*, 1:425–464, 1989.