M2AN. MATHEMATICAL MODELLING AND NUMERICAL ANALYSIS
- MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE

T. PHAM DINH
S. WANG
A. YASSINE

# Training multi-layered neural network with a trust-region based algorithm

*M2AN. Mathematical modelling and numerical analysis - Modélisation mathématique et analyse numérique*, tome 24, n° 4 (1990), p. 523-553

<http://www.numdam.org/item?id=M2AN_1990__24_4_523_0>

© AFCET, 1990, tous droits réservés.

L'accès aux archives de la revue « M2AN. Mathematical modelling and numerical analysis - Modélisation mathématique et analyse numérique » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM
*Article numérisé dans le cadre du programme*
*Numérisation de documents anciens mathématiques*
http://www.numdam.org/

MATHEMATICAL MODELLING AND NUMERICAL ANALYSIS
MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE
(Vol. 24, n° 4, 1990, p. 523 à 553)

# TRAINING MULTI-LAYERED NEURAL NETWORK WITH A TRUST-REGION BASED ALGORITHM (*)

T. PHAM DINH ([1,2]), S. WANG ([1,3]), A. YASSINE ([1,2])

Communicated by F. ROBERT

Abstract. — *In this paper, we first show how the problem of training a neural network is modelized as an optimization problem ; and the generally used training algorithm. Then we propose a new algorithm based on a trust-region technique which is very efficient for non-convex optimization problems. Experimental results show that the new algorithm is much faster and robust compared with GBP. It makes the design of neural net architecture much less problem-dependent.*

Résumé. — *Dans ce papier, nous allons présenter d'abord la modélisation du problème d'apprentissage par un réseau de neurones en un problème d'optimisation et l'algorithme d'apprentissage le plus utilisé (GBP). Ensuite, nous proposerons un nouvel algorithme basé sur la technique de région de confiance qui est très efficace pour des problèmes d'optimisation non convexe. Les résultats expérimentaux que nous donnerons montrent que le nouvel algorithme est beaucoup plus rapide et robuste par rapport à GBP. Et en plus, il rend la conception des réseaux beaucoup plus indépendante du problème particulier traité.*

## 1. INTRODUCTION

Recent years have seen explosive interests in neural networks, a new paradigm of information processing system (See Fogelman-Soulie, Gallinari, Le Cun & Thiria 1987, Rhumelhart & McClelland 1986, Wang, Yé & Robert 1988).

A neural network is a network of highly interconnected neuronlike subsystems that are dynamically coupled and exhibit useful computational properties via their collective behavior. There are many models of neural

(*) Received in December 1988.

([1]) Laboratoire TIM 3/INPG, 46, avenue Félix Viallet, 38031 Grenoble, France.
([2]) Équipe Optimisation
([3]) Équipe de Calcul Parallèle.

M² AN Modélisation mathématique et Analyse numérique 0764-583X/90/04/523/31/$ 5.10
Mathematical Modelling and Numerical Analysis © AFCET Gauthier-Villars

networks, some of them have been studied over the last 40 years. The recent resurgence of interests in neural networks is due to the rapid development of hardware and to the discovery of new algorithms which leads to the appearance of new models.

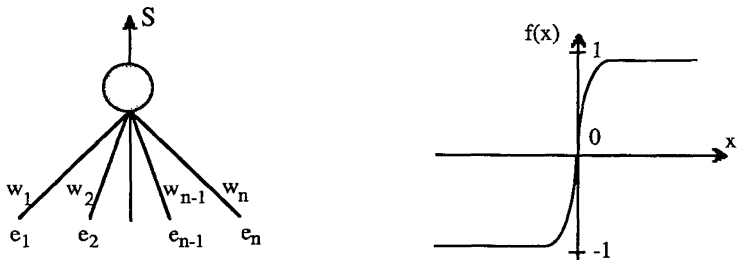
The model of multilayered neural network is commonly considered as a powerful connectionist model for the tasks such as constraint satisfaction and nonlinear classification. The example that most authors refer to for showing its advantage over other single-layered models is the learning problem of the exclusive-or relation (xor), because this elementary relation is characterized by separating four non-linearly separable points on a plane.

The learning (or training) algorithm for the multilayered network, called gradient back propagation, has been found by Hinton, Rumelhart (Rhumelhart & McClelland 1986) and by Le Cun (Le Cun 1987) independently. It is based on the method of steepest descent for the solution of optimization problems. The objective function here depends on the whole set of connection weights and on the desired input-output associations (patterns) to be realized. It is the total error committed by the network and measured in euclidean space. The goal of the optimization is to find a set of appropriate weights so that the objective function reaches its minimum. However, one would expect to diminish the objective function until a very small value, which means in practice that the network is able to do the desired input-output associations.

Here in the following, we present the mathematical modeling of the learning problem and the learning algorithm.

### 1.1. Gradient back propagation algorithm (GBP)

We consider an automata network, structured on successive layers, with the elements illustrated on the figure 1. From now on, we call them cells.




$S_i = f(A_i)$ where
$A_i = \sum_i w_i * e_i$ : Total input
$f(x) = (1-\exp(-\alpha x))/(1+\exp(-\alpha x))$

**Figure 1. — Elementary cell where $e_i$ are its inputs, $w_i$ are the input weights and $\alpha$ is the parameter determining the sigmoidal function $f$.**

Figure 2 gives an example of such a network. It receives inputs on the first layer (index 0) and produces the outputs on the last layer (index $N$). It is assumed that a connection between any two cells can only go from a lower-layer cell to a higher-layer cell and that no intra-layered connection is allowed. Each cell $i$ of the network updates its output by :

$$x_i = f(A_i) \quad \text{with} \quad A_i = \sum_j w_{ij} * x_j \tag{1}$$

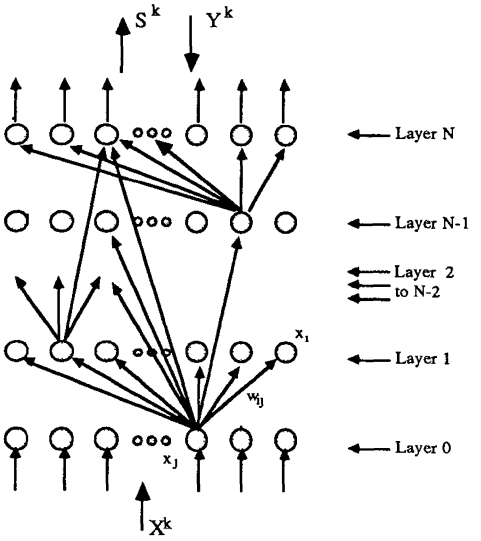



**Figure 2. — A general scheme of multilayered network.**

where $w_{ij}$ is the connection weight from cell $j$ to cell $i$ and $f$ is a sigmoidal function (as shown in *fig.* 1), and the sum is calculated over all the (lower-layer) cells $j$ which are connected to cell $i$.

$$S_i = f(A_i) \qquad \text{where}$$

$$A_i = \sum_i w_i * e_i : \quad \text{Total input}$$

$$f(x) = (1 - \exp(-\alpha x)/(1 + \exp(-\alpha x))$$

Suppose that there are $K$ input-output associations to be realized $(X^k, Y^k)$, for $X^k \in \mathbb{R}^n$, $Y^k \in \mathbb{R}^m$ and $k = 1, \ldots, K$. Each input vector $X^k$ is taken as states for the cells in layer 0, and the output $S^k$ a vector composed of the cell states in the last layer is calculated by applying the formula (1) to all the cells in each successive layer until the last one. The corresponding error of the network is defined as

$$C^k(W, X^k, Y^k) = \|S^{(k)} - Y^{(k)}\|^2 . \tag{2}$$

And the global error function is defined as the sum of all $C^k$.

$$C(W) = \sum_k C^k(W, X^k, Y^k) . \tag{3}$$

It is this error function that serves as the objective function to be minimized.

In the GBP algorithm, the gradient method is applied to each error function $C^k$, one by one following a pre-defined or random order.

If we note for the $k$-th association $(X^k, Y^k)$ a set of new variables $y_i$ defined as

$$y_i = \frac{\partial C^k}{\partial A_i} \tag{4}$$

the gradient of $C^k$ with respect to the connection weight $w_{ij}$ can be easily written as

$$\frac{\partial C^k}{\partial w_{ij}} = \frac{\partial C^k}{\partial A_i} * \frac{\partial A_i}{\partial w_{ij}} = y_i * x_j . \tag{5}$$

The formula (4) and (5) provide a convenient way to calculate the gradient of $C^k$. The gradient back-propagation algorithm can be shown through three phases :

1° Supply the input $X^k$, then calculate the output of the network $S^k$ by forward propagation of the states :

If $x_i$ is on the layer 0 : $x_i = X_i^k$

Otherwise $x_i = f(A_i)$

$$A_i = \sum_j w_{ij} x_j .$$

2° Supply the desired output $Y^k$. Calculate the gradient $y_i$ by the back-propagation :

For the output cells :

$$y_s = 2(S_s^k - Y_s^k) * f'(A_s) .$$

From layer $N - 1$ to 1 :

$$y_i = f'(A_i) * \sum_j w_{ji} y_j .$$

3° Modify the weights by gradient method :

$$w_{ij}^k = w_{ij}^{(k-1)} - \lambda_h y_i x_j$$

where $\lambda_h$ is a sequence of positive real numbers (converging to zero) which defines the length of forward step that the weight vector will be incremented in the direction of the gradient.

The algorithm is called gradient back propagation because the computation of the gradient is done in a way similar to the propagation of input states but in the opposite direction.

The discovery of this algorithm has answered the question of existence of multilevel learning algorithms for perceptron-like networks, put forth by Minsky and Papert (Minsky & Papert 1969). The algorithm makes it possible for the multilayered network to solve very complicated matching problems, while the algorithm itself is simple. But it also has some serious drawbacks.

1) The GBP is, from strategic point of view, too dependent on a single direction at each step, due to the nature of the gradient method. With the fixed forward factors $\lambda_h$ which can not be optimal, the GBP algorithm may cause serious problems such as misleading the searching process when $\lambda_h$ are relatively large or a very slow learning process otherwise.

2) The performance of the algorithm depends heavily on the empirical choice of learning parameters, network architecture and initial weights. These characteristics of networks, especially on what is concerned with the architecture (the distribution of cells over different layers, connections and weights), are determined by the nature of the problems to be treated (Wang 1988, Stéphane, Schereiber & Wang 1988). Thus a deep study of the problem should be made before an "appropriate" net architecture can be worked out. In other word, the design of network is very problem-dependent.

3) The algorithm aims mainly to find a minimum of the objective function, no special effort is made for finding the solution with a good neighborhood properties.

There are many ways to improve the algorithm. By experience, we feel that it is necessary to use some more sophisticated techniques in order to accelerate the searching of minimum and at the same time to have a good eigen-property for the objective function. Stéphane, Schereiber and Wang (Stéphane, Schereiber & Wang 1988) show the importance of network stability which has direct relation to the eigen-property.

Trust-Region (TR) algorithm has been developed in order to improve the learning algorithm. It is a good algorithm for the minimization problem of non-convex function (as for our cases). It consists in successively approximating the objective function by a quadratic form and reducing the function via the direction that minimizes the quadratic form. The scale of the region on which the function is approximated is determined dynamically regarding

the "quality" of the approximation. Since the quadratic form is a second order approximation depending on the gradient and the Hessian (in practice, it is an approximation of Hessian) of the objective function, the minimum found in this way has thus good second order properties.

In the section 2 of the paper we will present the TR algorithm in its original form, the theoretical results on the convergence and other properties of the algorithm. In section 3, we will discuss our adaptation of the algorithm to the learning problem. We will give the practical algorithms for each computing stage in the TR algorithm. The experimental results and the comparisons with the GBP algorithm will be given in the section 4.

## 2. TRUST REGION METHOD IN UNCONSTRAINED MINIMIZATION

We consider the unconstrained problem :

$$\min \{f(x) : x \in \mathbb{R}^n\} \qquad (P)$$

where $f$ is a function from $\mathbb{R}^n$ to $\mathbb{R}$. We denote by $g_k$ the gradient of $f$ at $x_k$ and by $H_k$ the Hessian of $f$ at $x_k$ or an approximation of it, which therefore will be called "quasi-Hessian". The main purpose of this section is to describe and analyze a technique for the solution of this problem.

The approach we shall present is well known (Moré 1983, Sorensen 1981). It is appropriately called a model trust region approach in which the step to a new iterate is obtained by minimizing a local quadratic model to the objective function over a restricted spherical region centered about the current iterate. The diameter of this region is expanded and contracted in a controlled way based upon how well the local model predicts behavior of the objective function. It is possible to control the iteration in the way so that convergence is forced from any starting value assuming reasonable conditions on the objective function.

In fact we shall present some very strong convergence properties for this method in § 2.4). There it is shown that one can expect (but not ensure) that the iteration will converge to a point which satisfies the second order necessary conditions for a minimum.

Trus region method computes a consequence of iterates by solving at each step a quadratic problem with an euclidean norm constraint

$$\min \{q_k(d) : \|d\| \leq \delta_k\} \qquad (P_k)$$

where $q_k$ is a quadratic approximation of the variation of $f$ at $x_k$ defined by :

$$q_k(d) = \langle g_k, d\rangle + 1/2 \langle H_k d, d\rangle .$$

The strictly positive number $\delta_k$ is the trust radius.

In trust region algorithms the computation of gradient of the objective function is required. The use of first order information leads in general to the first order stationary point. By incorporating the second order information $H_k = \nabla^2 f(x_k)$ these algorithms may satisfy the second order necessary conditions for $(P)$. In this case the trust region method can be regarded as modified Newton's method applied to finding a zero gradient of objective function. It can be described as follows :

Computing the direction $d_k$, solution to $(P_k)$, we can easily check the quality of the local approximation $q_k(d)$ and hence take the appropriate decision :

* If the approximation is satisfactory, then the solution of $(P_k)$ yields a new iterate, and the trust radius is increased.

* In the opposite case the iterate is unchanged, in addition the trust radius is decreased until $q_k$ yields a satisfactory approximation inside the trust region (which necessarily occurs for small radii since the gradient is supposed to be exactly known and the first order term in $q_k$ becomes dominant if $\|g_k\| \neq 0$).

Of course this general scheme can be implemented in many different ways. The quality of the approximation is generally examined through the following quantity, called "quality coefficient" :

$$r_k = \frac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)} . \tag{6}$$

The numerator represents the actual reduction of $f$ when we move from $x_k$ to $x_k + d_k$, the denominator represents the predicted reduction according to the quadratic approximation. Thus, in a trust region algorithm, the main source of computation effort, apart from the function evaluation required, is the work on a problem of the form $(P_k)$ to determine the step from the current iterate.

Trust region algorithms differ in their strategies for approximately solving $(P_k)$.

### 2.1. An abstract algorithm

Let us describe an abstract algorithm at the beginning :

1. Let $x_0 \in \mathbb{R}^n$, $\delta_0 > 0$ and $H_0$ be given
2. For $k = 0, 1, 2, \ldots$
   *a*) Compute $g_k = \nabla f(x_k)$, if $g_k = 0$, $x_k$ will be taken as an optimal solution. Stop ! (See § 2.3).

*b*) Determine a solution $d_k$ to problem $(P_k)$ and compute the quality coefficient $r_k$ via (6).
*c*) Update the iterate.
*d*) Update the trust radius and the quasi-Hessian and go to step 2.

## 2.2. Computing TR iteration

In the following five sub-sections we are going to discuss the last three points *a*), *b*) and *c*) of the above abstract algorithm (Gay 1981, Sorensen 1982, Moré 1983, Denis & Schnabel 1983).

### 2.2.1. *Update of the trust radius :*

The trust radius is updated according to the following rule : Let $0 < \mu < \eta < 1$ and $0 < \gamma_1 < \gamma_2 < 1 < \gamma_3$ be specified constants.

1. If $r_k \leq \mu$ then $\delta_{k+1} = \Delta \in [\gamma_1 \delta_k, \gamma_2 \delta_k]$
2. If $r_k \leq \eta$ then $\delta_{k+1} \in [\gamma_2 \delta_k, \delta_k]$ else $\delta_{k+1} \in [\delta_k, \gamma_3 \delta_k]$.

### 2.2.2. *Update of the iterate :*

In trust region method, the updating of the iterate is usually governed by a parameter $s$ such that $0 < s < 0.25$ which must be kept constant throughout the iteration. The rule is the following :

1. If $r_k < s$ then $x_{k+1} = x_k$
2. If $r_k \geq s$ then $x_{k+1} = x_k + d_k$.

Note that a significant decrease of the function is demanded to allow the algorithm to move from $x_k$ to $x_k + d_k$.

### 2.2.3. *Update of the quasi Hessian $H_k$ :*

In this section we will give two algorithms for updating $H_k$. Suppose $\gamma_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, $s_k = x_{k+1} - x_k$ and $H_0$ has been initialized as a symmetric matrix.

(i) Formula of rank 1 (DFP)

$$H_{k+1} = H_k + \alpha_k \cdot u_k \cdot u_k^t$$

where $\alpha_k = -1/s_k^t \cdot (H_k s_k - \gamma_k)$ and $u_k = H_k s_k - \gamma_k$.

(ii) Formula of rank 2 (BFGS)

$$H_{k+1} = H_k + \alpha_k \cdot u_k \cdot u_k^t + \beta_k \cdot v_k \cdot v_k^t$$

where $\alpha_k = 1/s_k^t \gamma_k$, $u_k = \gamma_k$ and $\beta_k = -1/s_k^t \cdot H_k s_k$, $v_k = H_k s_k$.

It has been proved (Minoux 1983, Gill & Murray 1972) that if $f$ is quadratic and $A = \nabla^2 f(x)$ is positive definite, the $H_k$ calculated by the above formula converges to $\nabla^2 f(x^*)$.

2.2.4. *The local problem and its properties :*

The local problem is reduced to that of minimizing a quadratic form inside a sphere :

$$\min \{ \langle g, d \rangle + 1/2 \langle d, Hd \rangle : \|d\| \leq \delta \} \qquad \text{(LP)}$$

where $g$ is a $n$-vector, $H$ is a symmetric matrix and $\delta$ is a positive number.

Since the constraint set $\{d \in \mathbb{R}^n : \|d\| \leq \delta\}$ is compact, the problem (LP) has a solution. If in addition $H$ is positive definite then there is uniqueness of solution to (LP).

Here our purpose is to give a complete discussion of the theoretical aspects of problem (LP) and to set out the nature of the computational difficulties that may be met.

LEMMA 2.1 (Gay 1981, Sorensen 1982) : *$d^*$ is a solution to* (LP), *if and only if there is $\mu \geq 0$ such that :*

(i) $(H + \mu I)$ *positive semidefinite*

(ii) $(H + \mu I)\, d^* = -g$

(iii) $\|d^*\| \leq \delta$ *and* $\mu(\|d^*\| - \delta) = 0$. □

The solution of (LP) is straightforward if (LP) has no solution on the boundary of $\{d \in \mathbb{R}^n : \|d\| \leq \delta\}$. In fact (LP) has no solution $d$ with $\|d\| = \delta$ if and only if $H$ is positive definite and $\|H^{-1} g\| < \delta$. The nonnegative scalar $\mu$ in Lemma 2.1 is called the Lagrange multiplier associated with the constraint $\|d\|^2 \leq \delta^2$.

It is worth noting that (LP) represents an interesting case of nonconvex optimization problem whose complete characterization of solution can be pointed out.

Now let us define a function $\phi$ on

$$\{\mu \in \mathbb{R} : H + \mu I \text{ non singular}\}$$

by

$$\phi(\mu) = \|d(\mu)\|$$

where $d(\mu)$ is solution of $(H + \mu I)\, d = -g$.

Denote by $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ the eigenvalues of $H$ and by $u_1, u_2, \ldots, u_n$ their corresponding eigenvectors. We note $\mathcal{N}(H - \lambda_1 I)$ the null space of $H - \lambda_1 I$ (or the eigenspace relative to $\lambda_1$), and $J_1 = \{i : \lambda_i = \lambda_1\}$.

It is clear that the solution of problem (LP) is closely related to the nonlinear equation $\phi(\mu) = \delta$ for $\mu$ in $]-\lambda_1, +\infty[$. More precisely this is the case where (LP) has a solution on the boundary of its constraint set and there is $\mu > \max(0, -\lambda_1)$ in Lemma 2.1. In this case the algorithm of Hebden (which shall be presented below) is known to be reliable and efficient for solving $\phi(\mu) - \delta = 0$.

The so called "hard case" corresponds to the situation where the coefficient $\mu$ in Lemma 2.1 must be equal to $-\lambda_1$.

Let us consider now the solution of the equation

$$(H + \mu I)\, d = -g \quad \text{for} \quad \mu \geq -\lambda_1 \tag{7}$$

using the eigensystem of $H$. We obtain that if $\mu > \lambda_1$ then the solution to (7) is defined by :

$$u_i^t\, d = \frac{-u_i^t\, g}{(\lambda_i + \mu)} \quad \text{for} \quad i = 1, \ldots, n\,.$$

Therefore $\phi(\mu) = \|d(\mu)\| = \left[\sum_{i=1}^{n} \frac{(u_i^t\, g)^2}{(\lambda_i + \mu)^2}\right]^{1/2}$.

The function $\phi(\mu)$ is positive and strictly decreasing in $]-\lambda_1, +\infty[$ with $\lim_{m \to \infty} \phi(\mu) = 0$, then the equation $\phi(\mu) = \delta$ has at most one solution in this interval.

LEMMA 2.2

1° *If $g$ is perpendicular to $\mathcal{N}(H - \lambda_1 I)$ then problem* (7) *with $\mu \geq -\lambda_1$ has always a solution.*

2° *Problem* (7) *with $\mu = -\lambda_1$ has a solution if and only if $g$ perpendicular to $\mathcal{N}(H - \lambda_1 I)$.*

*Moreover the solution set to problem* (7) *with $\mu = -\lambda_1$ is $d^* + \mathcal{N}(H + \mu I)$ where $d^*$ is defined by :*

$$u_i^t\, d^* = 0 \qquad \text{if} \quad i \in J_1$$
$$u_i^t\, d^* = -u_i^t\, g/(\lambda_i + \mu) \qquad \text{otherwise.}\ \square$$

See Appendix A for the proof.

The following results (which are consequence of the previous ones) are helpful in understanding the algorithm described in the sequel for solving (LP).

LEMMA 2.3 : *Suppose that $g \neq 0$ in* (LP) *(See Lemma 2.4 for the case $g = 0$).*

1° *If $\lambda_1 > 0$ (i.e. $H$ is positive definite) then*

(i) *If $\|H^{-1} g\| \leq \delta$ then $d = -H^{-1} g$ is the only solution to* (LP) *($\mu = 0$ in Lemma 2.1 and $\phi(\mu) - \delta = \|d(\mu)\| - \delta \leq 0$).*

(ii) *Otherwise $\|H^{-1} g\| > \delta$ and $\phi(0) - \delta = \|d(0)\| - \delta > 0$, therefore there is an unique solution $\mu > 0$ to $\phi(\mu) = \delta$.*

2° *If* $\lambda_1 = 0$ *(i.e. H is only positive semidefinite) then :*

(i) *If* $\| (H - \lambda_1 I)^+ g \| \leq \delta$ *and* $(H - \lambda_1 I)(H - \lambda_1 I)^+ g = g$ *(this equality occurs if and only if g is perpendicular to* $\mathcal{N}(H - \lambda_1 I)$ *according to Lemma 2.2) then every* $d = -(H - \lambda_1 I)^+ g + u$, *with* $u \in \mathcal{N}(H - \lambda_1 I)$ *and* $\|d\|^2 = \| (H - \lambda_1 I)^+ g \|^2 + \|u\|^2 \leq \delta^2$ *is a solution to problem* (LP), ($\mu = 0$ *in Lemma 2.1).*

(ii) *If* $\| (H - \lambda_1 I)^+ g \| > \delta$ *then* $\phi(\mu) - \delta = 0$ *admits the unique solution in* $]-\lambda_1, +\infty[$.

3° *If* $\lambda_1 < 0$ *then :*

(i) *If* $\| (H - \lambda_1 I)^+ g \| \leq \delta$ and $(H - \lambda_1 I)(H - \lambda_1 I)^+ g = g$ *(this equality occurs if and only if g is perpendicular to* $\mathcal{N}(H - \lambda_1 I)$ *according to Lemma 2.2) then every* $d = -(H - \lambda_1 I)^+ g + u$, *with* $u \in \mathcal{N}(H - \lambda_1 I)$ *and* $\|d\|^2 = \| (H - \lambda_1 I)^+ g \|^2 + \|u\|^2 = \delta^2$ *is a solution to problem* (LP), ($\mu = -\lambda_1 > 0$ *in Lemma 2.1)*

(ii) *If* $\| (H - \lambda_1 I)^+ g \| > \delta$ *then* $\phi(\mu) - \delta = 0$ *admits the unique solution in* $]-\lambda_1, +\infty[$, ($\mu > -\lambda_1 > 0$ *in Lemma 2.1).*

4° *If* $\lambda_n \leq 0$ *(i.e. the quadratic function* $q(d)$ *is concave) then* (LP) *has only solution on the boundary of* $\{d : \|d\| \leq \delta\}$ *unless g and H are null (Rockafellar 1970).*

(i) *If* $\lambda_1 = 0$ *then* $H = 0$. *In this case the solution set of* (LP) *is :*

$$\begin{array}{ll} \{-(g/\|g\|)\delta\} & \text{if } g \neq 0 \\ \{d : \|d\| \leq \delta\} & \text{if } g = 0 . \end{array}$$

*(See also Lemma 2.4)*

(ii) *If* $\lambda_1 < 0$ *then we apply the same results as in 3°.* □

The situation where $g = 0$ in (LP) is closely related to the variational spectral theory (Lancaster 1969).

LEMMA 2.4 : *If* $g = 0$ *in problem* (LP) *then the solution set of* (LP) *is :*

(i) $\{0\}$ *if* $\lambda_1 > 0$ ;

(ii) $\{d \in \mathcal{N}(H - \lambda_1 I) : \|d\| \leq \delta\}$ *if* $\lambda_1 = 0$ ;

(iii) $\{d \in \mathcal{N}(H - \lambda_1 I) : \|d\| = \delta\}$ *if* $\lambda_1 < 0$ *(in this case* $\mu = -\lambda$ *in Lemma 2.1).* □

*Proof:* We can use the results above or those concerning variational properties of the spectral theory (Lancaster 1969).

2.2.5. *Algorithms for the local problem* (LP) *:*

Three algorithms will be given for solving (LP) in different cases. The first two, Hebden and Projection methods, are generally applied when $H$ is

positive semidefinite, and the third one using curvature information is used to deal with the "hard cases" that correspond to 2° (i) and 3° (ii) in Lemma 2.3.

In the hard cases, except that the eigensystem of $H$ is easily obtainable (See Lemma 2.2) recognizing each of the situations 2° (i) and 3° (ii) in Lemma 2.3 may also be time consuming because of the computation of $(H - \lambda_1 I)^+ g$. Fortunately there is a completely acceptable alternative due to (Shultz, Schnabel & Byrd 1985) (See § 2.2.5) *c*). Their inexpensive approximate solution technique appears to perform as well as the more expensive exact method in practice. This third algorithm requires the computation of the smallest eigenvalue $\lambda_1$ of $H$ and an eigenvector of $H$ corresponding to $\lambda_1$.

a) *Hebden algorithm*

Let us describe now this algorithm for the solution problem (LP) in the case where the nonlinear equation : $\phi(\mu) - \delta = 0$, with :

$$\phi(\mu) = \left[ \sum_{i=1}^{n} \left( \frac{u_i^t g}{\lambda_i + \mu} \right)^2 \right]^{1/2}$$

admits an unique root in $]-\lambda_1, +\infty[$.

(Reinsch 1971) and (Hebden 1973) observed independently that in order to solve (7) the fact which should be taken into account is that the function $\phi^2(\mu)$ is a rational function in $\mu$ with second order poles on a subset of the negatives of the eigenvalues of $H$ (See (7)).

The Newton's method which is based on a local linear approximation to $\phi(\mu)$ is then not likely to the best method for solving (7) because the rational structure of $\phi^2(\mu)$ is ignored. Instead, an interation for solving (7) can be derived based upon a local rational approximation to $\phi$. The iteration is obtained by requiring $\phi^*(\mu) = \gamma/(\alpha + \mu)$ to satisfy $\phi^*(\mu) = \phi(\mu)$, $\phi^{*\prime}(\mu)\phi'(\mu)$ where we take $\mu$ as the current approximation to the root $\mu^*$. This approximation is then improved by solving for an $\hat{\mu}$ that satisfies $\phi^*(\hat{\mu}) = \delta$. The resulting iteration is

$$\mu_{k+1} = \mu_k + \left( \frac{\phi(\mu_k)}{\phi'(\mu_k)} \right) * \left( \frac{\delta - \phi(\mu_k)}{\delta} \right). \tag{8}$$

In fact Hebden's algorithm can be viewed as Newton's algorithm applied to the equation

$$\psi(\mu) = \frac{1}{\delta} - \frac{1}{\phi(\mu)} = 0 \quad \text{for} \quad \mu \in ]-\lambda_1, +\infty[ .$$

The local rate convergence of this iteration is quadratic but the most

important feature of (8) is that usually the number of iterations required to produce an acceptable approximation of $\mu^*$ is very small because the iteration is based upon the rational structure of $\phi^2$. Iteration (8) can be implemented without explicit knowledge of the eigensystem of $H$. This important observation which is due to (Hebden 1973) makes it possible to implement (8) merely by solving linear system with $(H + \mu I)$ as coefficient matrix. This is easy to see since

$$\phi(\mu) = \| d(\mu) \|$$

and

$$\phi'(\mu) = - (1/\phi(\mu))\, d(\mu)^t\, (H + \mu \mathrm{I})^{-1}\, d(\mu)$$

where $(H + \mu I)\, d(\mu) = - g$.

Therefore the Hebden's algorithm can be described as follows : Let $\mu_0 > 0$ with $H + \mu_0 I$ positive definite, and $\phi(\mu_0) > 0$.
$k = 0, 1, 2, \ldots$ until convergence

1) Factor $(H + \mu_k + I) = R^T R$.
2) Solve $R^T R p = - g$.
3) Solve $R^T q = p$.
4) Let $\mu_{k+1} = \mu_k + \left[ \frac{\| p \|^2}{\| q \|} \right] \left( \frac{\| p \| - \delta}{\delta} \right)$.

$k = k + 1$ and return to 1).

In this algorithm $R^T R$ is the Cholesky factorization of $(H + \mu I)$ with $R$ upper triangular. The function $\psi$ is convex and strictly decreasing on $]- \lambda_1, + \infty[$.
This fact was discovered by (Reinsch 1971) and follows from the expression of $\phi(\mu)$. It implies that Newton's method started from $\mu_0 \in ]- \lambda_1, + \infty[$ with $\psi(\mu_0) > 0$ (i.e. $\phi(\mu_0) - \delta > 0$) produces a monotonically increasing sequence converging to the solution of $\phi(\mu) - \delta = 0$.

b) *Projection methods*

i) Projected gradient method :

Let $H$ be a $n * n$ symmetric positive semidefinite and $g$ a vector in $\mathbb{R}^n$. We note

$$f(x) = 1/2 \langle x, Hx \rangle + \langle g, x \rangle$$

as the corresponding convex quadratic function.

Consider now the following convex optimization problem :

$$\alpha = \mathrm{Inf} \{ f(x) : x \in C \} \qquad (\mathscr{P})$$

where $C$ is a closed convex set in $\mathbb{R}^n$. The solution set to $(\mathscr{P})$ is denoted by $S$.

We shall use the following characterization of solution to ($\mathscr{P}$) (Ekeland & Temam 1974) : $x^*$ is solution to ($\mathscr{P}$) if and only if

$$x^* = \text{Proj}_C\,(x^* - \rho\,\nabla f(x^*)) \quad \text{for some} \quad \rho > 0\,.$$

where $\text{Proj}_C$ denotes (orthogonal) projection operator on $C$ :

$$x = \text{Proj}_C\,(y) \Leftrightarrow \|y - x\| = \min\{\|y - u\| : u \in C\}\,.$$

The projected gradient method for solving ($\mathscr{P}$) is then defined as :

Starting with an arbitrary $x_0$ in $C$, we set

$$x_{k+1} = \text{Proj}_C\,(x_k - \rho_k\,\nabla f(x_k)) \tag{9}$$

where the sequence of positive scalars $(\rho_k)$ are chosen in order to assure the convergence of $(x_k)$ to a solution to ($\mathscr{P}$).

We shall give a particular interesting choice of $(\rho_k)$ which is based on the contraction property. For other choices, see (Auslender 1976).

It is clear that if $\rho_k = \rho$ for every $k$ then (9) is exactly the fixed point iteration relative to $F_\rho(x) = \text{Proj}_C\,(x - \rho\,\nabla f(x))$. Our choice is :

$$\rho_k = \rho^* = 2/(\lambda_1 + \lambda_n)\,.$$

The convergence result concerning this method is given in the Appendix B.

*Remarks :*

1. The projected gradient algorithm is practically interesting when the operator $\text{Proj}_C$ is easy to calculate. For instance if

$$C = \{x \in \mathbb{R}^n : \|x - y\| \leq r\}\,,\ C = \prod_{i=1}^{n} [a_i, b_i] \text{ or}$$

$$C = \mathbb{R}^n_+ = \{x \in \mathbb{R}^n : x \geq 0\}\,.$$

2. To obtain the convergence of this algorithm, we could use its following version :

For $y_0$ arbitrarily chosen, we construct the sequence $(y_k)$ by :

$$y_{k+1} = y_k + \lambda_k(\text{Proj}_C\,(y_k - \rho_k\,\nabla f(y_k)) - y_k)$$

where $\lambda_k$ is such that $y_{k+1}$ be a solution of the problem :

$$f(y_{k+1}) = \min\,\{f(y : y \in [y_k, \text{Proj}_C\,(y_k - \rho_k\,\nabla f(y_k))])\}\,.$$

ii) Primal dual or Uzawa algorithm :

After introducing the duality (relative to the primal problem ($\mathscr{P}$)) with the help of Lagrangian $L(x, \mu)$, we define the dual problem ($\mathscr{D}$) and

describe the main relations between solutions of primal and dual problems. Uzawa algorithm is effectively based on this duality.

For the generality of the duality theory and the details of the results presented here, one can see (Ekeland & Teman 1974, Laurent 1972).

Let us suppose that the closed convex set $C$ be defined analytically by :

$$C = \{x \in \mathbb{R}^n : f_i(x) \leq 0 , \quad i = 1, ..., m\}$$

where $f_i, i = 1, ..., m$, is convex continuous function on $\mathbb{R}^n$. If we define the vector-valued function $\phi$ as :

$$\phi(x) = (f_1(x), ..., f_m(x))$$

for every $x$ in $\mathbb{R}^n$. Then $C$ can be written as

$$C = \{x \in \mathbb{R}^n : \phi(x) \leq 0\} .$$

We define the Lagrangian $L(x, \mu)$ on $\mathbb{R}^n \times \mathbb{R}^m_+$ by

$$L(x, \mu) = f(x) + \langle \mu, \phi(x) \rangle$$

where $\mu = (\mu_i)$ and $\mathbb{R}^m_+ = \{\mu \in \mathbb{R}^m : \mu \geq 0\}$.

It is clear that $L(x, \mu)$ is convex (resp. concave or rather affine) in $x$ (resp. in $\mu$) for $\mu$ fixed (resp. $x$ fixed). It follows that the function $g(\mu) = \inf \{L(x, \mu) ; x \in \mathbb{R}^n\}$ is concave.

The dual problem $(\mathscr{D})$ is then defined by :

$$\beta = \sup \{g(\mu) : \mu \in \mathbb{R}^m_+\} . \qquad (\mathscr{D})$$

An element $(x^*, \mu^*) \in \mathbb{R}^n \times \mathbb{R}^m_+$ is called a saddle point of the Lagrangian $L(x, \mu)$ if

$$L(x^*, \mu) \leq L(x^*, \mu^*) \leq L(x, \mu^*) \qquad (*)$$

for every $(x, \mu) \in \mathbb{R}^n \times \mathbb{R}^m_+$.

Note $S(\mu)$ for each $\mu \in \mathbb{R}^m_+$ the solution set to the following problem :

$$g(\mu) = \inf \{L(x, \mu) : x \in \mathbb{R}_n\} . \qquad (\mathscr{P}_\mu)$$

The Uzawa algorithm can be described as follow : Starting with an arbitrary $\mu_0$ in $\mathbb{R}^m_+$ we define $x_0$ as an element in $S(\mu_0)$. If $\mu_k$ is known we take $x_k$ in $S(\mu_k)$ and define $\mu_{k+1}$ by :

$$\mu_{k+1} = \underset{\mathbb{R}^m_+}{\text{Proj}} (\mu_k + \rho_k \phi(x_k))$$

where $\rho_k$ is chosen in $]0, \rho^*[$, being a constant depending on $H$ (See § 2.2.5) *b*)

The theoretical results about the method is given in Appendix C. There we can see that Uzawa algorithm solves primal and dual problems at the same time. It is worth noting that Uzawa algorithm is projected gradient method applied to the dual problem ($\mathscr{D}$).

*Remarks*

i) $x = \underset{\mathbb{R}^n_+}{\text{Proj}}(y)$ is given by a : $x_i = y_i$ if $y_i \geq 0$, 0 if $y_i \leq 0$, $i = 1, \ldots, n$.

ii) In trust region method, the closed convex set $C$ is of the form $C = \{x \in \mathbb{R}^n : 1/2 \|x\|^2 \leq 1/2 \cdot \delta^2\}$ ; then we have $m = 1$ and $S(\mu) = \{x \in \mathbb{R}^n : (H + \mu I)x = -g\}$. It follows that $S(\mu)$ is singleton if $H$ is positive definite or $\mu > 0$.

Like Hebden method, the method of Uzawa works well only when the matrix $H$ is at last positive semidefinite.

c) *The most negative curvature method*

The two methods stated above for solving the problem (LP) are very efficient when the matrix $H$ is positive definite. If $H$ is indefinite or only positive semidefinite it should be preferable to use the following strategy due to (Shultz, Schnabel & Byrd 1985) for the solution of problem (LP) in the hard case ($m = -\lambda_1$ in Lemma 2.1 & 2.3)

1° Taking $\alpha \in ]-\lambda_1, c \cdot \max(|\lambda_1|, \lambda_n)]$, where $c > 1$ is a constant which depends on each problem.
2° Solving $p = -(H + \alpha I)^{-1} g$.
3° If $\|p\| > \delta$, then apply the Hebden method to $H := H + \alpha I$ in order to find a direction $d$.
4° If $\|p\| = \delta$, then $d = p$.
5° If $\|p\| < \delta$, then $d = p + \xi u_1$, where $u_1$ is an eigenvector of $H$ corresponding to $\lambda_1$ and $\xi$ is chosen such that : $\|d\| = \delta$ and sign $(\xi)$ = sign $(u_1^T p)$.

Compared to Hebden algorithm, the only modification introduced by the method of negative curvature is at step 5) where we should proceed as if $\mu = -\lambda_1$ in Lemmas 2.1 & 2.3.

The direction $d$ obtained by this method gives as good a decrease of quadratic model as a direction of sufficient negative curvature and satisfies the practical conditions 1 and 2 that will be given in § 2.4).

### 2.3. Convergence tests

We can use the following termination criteria in trust region algorithms :

* First order necessary condition test :

$$\text{If } \nabla f(x_k) = 0 \quad \text{then stop .}$$

* Second order sufficient condition : when the first order necessary condition is satisfied we can stop the algorithms or proceed to verify second order sufficient condition as follows :

(i) If $H_k$ is not positive definite then the algorithm continues (See Lemma 2.4, for the solution of (LP) in the case where $g = 0$).

(ii) If $\nabla^2 f(x_k)$ is positive definite then stop. Otherwise $H_k = \nabla^2 f(x_k)$ and we restart the algorithm from the step *d*) (See § 2.1).

### 2.4. Convergence results of the TR method

We end this section with some well-known convergence results on the TR algorithm.

THEOREM 2.5. (Moré 1983, Sorensen 1983, Sorensen & Moré 1981) : *If the function f is differentiable, bounded below on* $\mathbb{R}^n$, *and if* $\nabla f$ *is uniformly continuous then :*

$$\lim_{k \to +\infty} \|\nabla f(x_k)\| = 0 .$$

THEOREM 2.6 : *If the function f is twice continuously differentiable and bounded below on* $\mathbb{R}^n$, *and if* $\nabla^2 f$ *is bounded on the level set* $\{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ *then trust region algorithms with* $H_k = \nabla^2 f(x_k)$ *possess the following properties :*

1° $\lim_{k \to +\infty} \|\nabla f(x_k)\| = 0.$

2° *If* $\{x_k\}$ *is bounded then there is a limit point* $x^*$ *with* $\nabla^2 f(x^*)$ *positive semidefinite.*

3° *If* $x^*$ *is an isolated limit point of* $\{x_k\}$ *then* $\nabla^2 f(x^*)$ *positive semidefinite.*

4° *If* $\nabla^2 f(x^*)$ *is non-singular for some limit point* $x^*$ *of* $\{x_k\}$ *then*

a) $\nabla^2 f(x^*)$ *is positive definite ;*

b) $\lim x_k = x^*$ *and there exists a* $\varepsilon > 0$ *and integer K such that* $\delta_k > \varepsilon$ *for all* $k > K$ ;

c) *The convergence is superlinear.*

Besides the superlinear convergency, the zero of gradient and the positive definiteness of Hessian are also very important properties of the algorithm which encourage us to adapt the algorithm to our learning problem. Although in practice, we use an approximation instead of computing the Hessian during the optimization, these properties are still present.

In the above two theorems, we have an implicit assumption that every step in the TR algorithm can be exactly carried. For example, $d_k$ is supposed to be exactly solved. Apparently, it cannot be always true in the practice. What conditions every calculated direction should satisfy in order to preserve the above convergence properties ?

Before giving the convergence conditions, we define an additional notation : let $d(g, H, \delta)$ stand for a calculated direction (depending on $g$, $H$ and $\delta$),

$$\text{pred } (g, H, \delta) = - \langle g, d(g, H, \delta)\rangle - 1/2 \langle d(g, H, \delta), Hd(g, H, \delta)\rangle .$$

Our conditions that a step selection strategy may satisfy are :

**Condition 1**

There exists $c_1$, $s_1 > 0$ so that $\forall g \in \mathbb{R}^n$, $\forall H \in \mathbb{R}^{n\times n}$ symmetric et $\forall \delta > 0$ then :

$$\text{pred } (g, H, \delta) \geq c_1 . \|g\| . \min (\delta, s_1 . \|g\|/\|H\|) .$$

**Condition 2**

There exists $c_2 > 0$ so that $\forall g \in \mathbb{R}^n$, $\forall H \in \mathbb{R}^{n\times n}$ symmetric and $\forall d > 0$ then :

$$\text{pred } (g, H, \delta) \geq c_2 . (-\lambda_1(H)) . \delta^2 ,$$

here $\lambda_1$ is the most smallest eigenvalue of $H$.

**Condition 3**

If the matrix $H$ is positive definite and $\|H^{-1}g\| \leq \delta$ then :

$$d(g, H, \delta) = - H^{-1} g .$$

THEOREM 2.7 : *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be twice continuously differentiable and bounded from below, and let $H(x) = \nabla^2 f(x)$ satisfy $\|H(x)\| \leq \beta_1$ for all $x \in \mathbb{R}^n$. Suppose that an practical TR algorithm is applied to $f(x)$, starting from $x_0 \in \mathbb{R}^n$, generating a sequence $\{x_k\}$, $x_k \in \mathbb{R}^n$, $k = 1, 2, \ldots$,* then :

1° *If $d(g, H, \delta)$ satisfies Condition 1 and $\|H_k\| \leq \beta_2$ for all k, then $\nabla f(x_k)$ converges to 0 (first order stationary point convergence).*

2° *If $d(g, H, \delta)$ satisfies Condition 1 and 3, $H_k = H(x_k)$ for all k, $H(x)$ is Lipschitz continuous with constant L, and $x^*$ is a limit point of $\{x_k\}$ with $H(x^*)$ positive definite, then $x_k$ converges q-quadratically to $x^*$.*

3° *If $d(g, H\delta)$ satisfies Condition 1 and 2, $H_k = H(x_k)$ for all k, $H(x)$ is uniformly continuous, and $\{x_k\}$ converges to $x^*$, then $H(x^*)$ is positive semidefinite (second order stationary point convergence, with 1.)*

### 3. ADAPTED TR LEARNING ALGORITHM

In this section, we will present the practical TR based algorithm utilized in our experiments.

In applying the TR algorithm to the learning problem by a multilayered network, we need some special consideration in programming, because the method needs the gradient of the objective function with respect to all the connection weights instead of the gradient with respect to all the total inputs (to the cells) as in the GBP algorithm, and the objective function should often be evaluated for sets of temporal connection weights.

In order to maintain the same notation, we still use $f$ to represent the function to be minimized, that means $f(W) = C(W)$, where $W$ can be viewed as a vector of connection weights. Our practical algorithm is as follows :

0° *Initialization :*

$W^{(0)}$: The weight set which is initialized arbitrarily between $-1.0$ and 1.0.

$$\begin{aligned} \delta_0 &> 0 : \text{ initial trust radius.} \\ \varepsilon &> 0 : \text{ "zero" for general tests ;} \\ \varepsilon_g &> 0 : \text{ zero for } \|g\| ; \\ \varepsilon_f &> 0 : \text{ zero for } f ; \end{aligned}$$

$H_0$: A small-value symmetric matrix, or if possible, taking $H_0$ as $H(W_0)$ the Hessian of the objective function at $W_0$. Evaluating the smallest eigenvalue $\lambda_1$ and one corresponding eigenvector, $v_1$ ; $k = 0$ ; iteration counter.

1° Calculating $f_k = f(W^{(k)})$, $g_k = \nabla f(W^{(k)})$.

2° If $\|g_k\| \leq \varepsilon_g$ or $f_k \leq \varepsilon_f$ then $W^{(k)}$ is our solution ; stop.

3° Calculating $d_k$ :

If $\lambda_1 > 0$, solve $H_k d = -g_k$ ;

$$\text{if } (\|d\| \leq \delta_k) \text{ then } d_k = d ;$$

else using Hebden or Projection method to find a $\mu > 0$ so that the solution of $(H_k + \mu I) d = -g_k$ satisfies $|\,\|d\| - \delta_k| \leq \varepsilon$, then $d_k = d$ ;
else $(*\lambda_1 \leq 0*)$ then $\alpha_k = -\lambda_1 + 0.0001$ ; solving $p_k = -(H_k + \alpha_k I)^{-1} g_k$.
if $(\|p_k\| > \delta_k)$ then apply the Hebden method to $H = H_k + \alpha_k I$ in order to find

$$a\mu > 0 \text{ so that } \left\| (H_k + (\mu + \alpha_k) I)^{-1} g_k \right\| = \delta_k$$

then

$$d_k = -\ (H_k + (\mu + \alpha_k)\, I)^{-1}\, g_k\ ;$$

else if $(\|p_k\| = \delta_k)$ then $d_k = p_k$.

else if $(\|p_k\| < \delta_k)$ then $d_k = p_k + \xi u_1$, where $\xi$ is chosen so that $\|d_k\| = \delta_k$ and

$$\text{sign}\ (\xi) = \text{sign}\ (u_1^T p_k)\ .$$

4° Let $r_k = \dfrac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)}$.

If $(r_k \geq 0.1)$ then $W^{(k+1)} = W^{(k)} + d_k$ else $W^{(k+1)} = W^{(k)}$.

5° If $r_k \leq 0.25$ then $\delta_{k+1} = \delta_k/2$
else if $(r_k \geq 0.75)$ then $\delta_{k+1} = 2\,\delta_k$
else $\delta_{k+1} = \delta_k$.

6° Updating $H_{k+1}$ by the algorithms in 3.2) formula of rank 1, or of rank 2. Using any practical method to calculate the smallest eigenvalue and a corresponding eigenvector of $H_{k+1}$.

7° $k = k + 1$ and return to step 1°).

## 4. EXPERIMENTAL RESULTS

Three experiments are described below. The first one shows a comparison of average performance between the GBP algorithm and the TR algorithm for two different structures of network. The second one is to show the advantage of the TR algorithm with regard to the "hard" cases for GBP algorithm, although these hard cases are not frequent events when connection weights are initialized randomly. The third one shows that the performance of the TR algorithm is much less dependent on the network architecture and more robust.

The testing problem is the learning problem of xor relation. For GBP algorithm, the order of pattern presentation is fixed and the learning rate has been initialized or fixed depending on the purpose of the tests. The sigmoidal function for each cell is not identical, the alfa value defining each of the functions has been initialized between 0.5 and 1.5. When there is an initialization of the connection weights, it has always been taken between $-1$ and 1.

### 4.1. Average performance comparison

The two networks figure 3 (xor r 211) and figure 4 (xor_r_241) have fixed structures as shown in the following pictures. The "blank" cells on

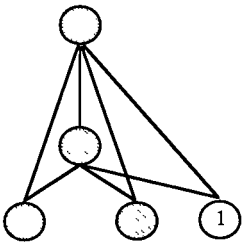



**Figure 3. — xor_r_211, network for xor with 2 input cells, 1 intermediate cell, and 1 output cell.**

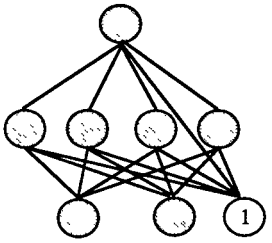



**Figure 4. — xor_r_241 : network for xor with 2 input cells, 4 intermediate cells, and 1 output cell.**

these pictures are those who serve as threshold cells with 1 as their states. Every other cell should be connected to it except for those of input.

The connection weights are initialized 100 times for each network. For each set of initial weights, the GBP and TR are applied to improve the weights. The maximum number of training iterations is fixed to be 1 000. For the GBP method, the learning rate is initialized between 0 and 0.4 for each new set of weights. The two figures (5 & 6) show the average evolution of the objective function in each net. Only the results within 100 iterations are shown. It is remarked that the function has rapidly converged to zero using TR algorithm in both cases, but only the second network xor_r_241 with (too) many connections has reached a comparable performance using GBP algorithm, training the net xor_r_211 with GBP has failed since the total error stays approximately at 1.0 (at least until 1 000 learning iterations).

### 4.2. Comparison Between Particular Nets

Two fixed networks xor_S 1 and xor_S_2 shown below (*fig.* 7 & 8) are among the "hard" cases for GBP. The learning rate for GBP has been chosen to be 0.2. None of the two nets can be trained in less than 1 000 iterations by GBP.

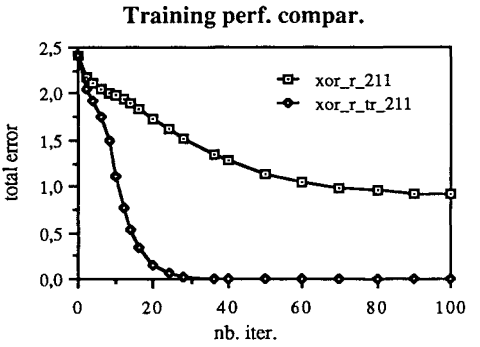



**Figure 5. — Average performance of the two algorithms over 100 randomly initialized weight sets for the network in figure 3 (xor_r_211). The legend xor_r_tr_211 corresponds to the same net but trained by TR algo.**

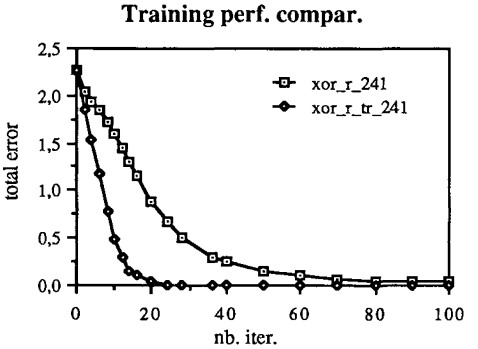



**Figure 6. — Average performance of the two algorithms over 100 randomly initialized weight sets for the network in figure 4 (xor_r_241). The legend xor_r_tr_241 corresponds to the same net but trained by TR algo.**

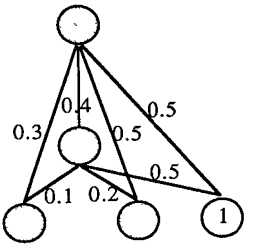



**Figure 7. — xor_S_1, a 2_1_1 net with given initial weights.**

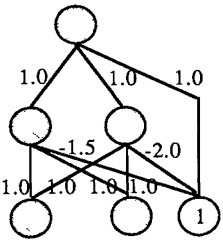



**Figure 8. — xor_S_1, a 2 2_1 net with given initial weights.**

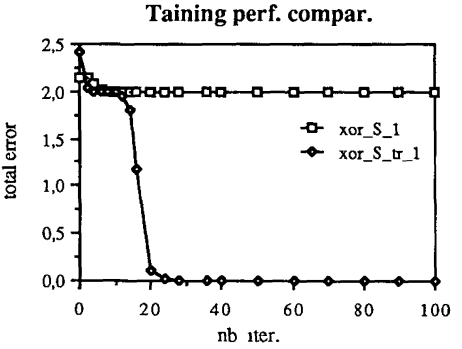



**Figure 9. — xor_S_1 corresponds to the net in figure 7 trained by GBP algorithm, xor S_tr_1 corresponds to the net in figure 7 trained by TR algorithm.**

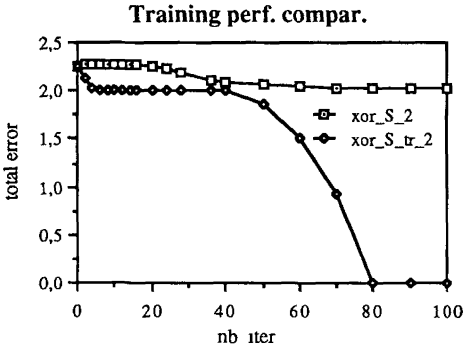



**Figure 10. — xor_S_1 corresponds to the net in figure 8 trained by GBP algorithm, xor_S_tr_1 corresponds to the net in figure 8 trained by TR algorithm.**

Note : The "hard" cases for GBP observed in these two examples are not rare when the number of connections is relatively small compared to the number of patterns and a majority of weights have a same sign. It is remarked that the TR algorithm may also meet difficulties (*fig.* 10), but it has eventually converged while there has been no hope for GBP algorithm.

### 4.3. Structure independence

Many studies on the multilayered network now concern with the architecture of network, because GBP is a slow and very inefficient algorithm. In the real application, we have to work out many "intelligent" net structures (especially the receiving field of hidden cell) to facilitate the

learning and to favorite certain generalization tasks. This is somewhat paradoxal with the saying that GBP can learn automatically what a network is asked to do. A more efficient algorithm is needed for training multilayered networks because for many application problems we can not have a priori much information about their nature which allows us to design special networks.

The following example shows the robustness of our new algorithm with respect to net structure. It concerns about the realization of a vectorial boolean function by a 6_3_3 network. The function $F$ is defined as

$$F : \{-1, 1\}^6 \to \{-1, 1\}^3$$

$$F(x_1, x_2, x_3, x_4, x_5, x_6)^t = (\bar{x}_1 x_2 + \bar{x}_2 x_1, \bar{x}_3 x_4 + \bar{x}_4 x_3, \bar{x}_5 x_6 + \bar{x}_6 x_5)^t .$$

It is composed of 3 xor functions. There are, in total, 64 associations to be realized.

From the experience we know that if we use GBP algorithm, one of the best choices for the net architecture is as shown in figure 11. This is a natural choice because each output cell depends upon only 2 input cells and it does not need any information from other input cells. The intermediate cells are equally "attributed to" each of the three parts in order to make each part be able to deal with the nonlinear separateness of the xor problem.

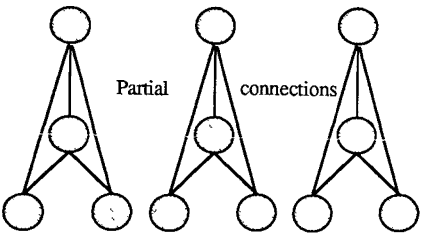



**Figure 11. — A 6 3 3 net composed of three 2_1_1 nets can realize $F$ with both algorithms.**

In this network there is not any connection between each output cell and any irrelevant input cell. However, in the real application we may not get enough information for us to determine this "intelligent" net. We may chose a network with full inter-layer connection as shown in figure 12. We are going to see that the TR based algorithm works well for both networks but GBP does not.

In designing the figure 11 (also *fig.* 12) we have ignored the threshold cell (blank cell in the previous net figures) and (of course) all the connections issued from this cell because of the space constraint. Another reason

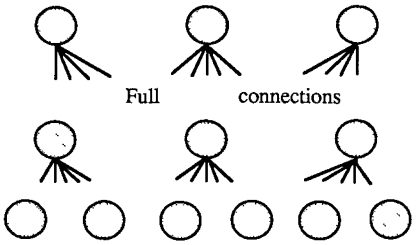



**Figure 12. — A 6_3_3 net with full inter-layered connection. It is very difficult to be trained be GBP but easily by TR algorithm.**

allowing us to do so is that it has become a convention that each non-input cell should be connected to a special cell of state 1, where the connection weight acts as threshold of the previous one.

Experimental results show that if the weights in the network of figure 11 are not too "badly" initialized, we can train this net by GBP to realize the function although the training times may be very long.

We have done the same experiments as in § 4.1 for both networks. The statistical result for the first network is quite the same as shown in figure 5, because the network is composed of three nets with the same architecture as in figure 3. Here we given only the performance result (*fig.* 13) about the second network. It is seen that there is no chance for it to be trained by GBP.

Although the first net is also a good choice for TR algorithm, the result in figure 13 shows that it is not indispensable.

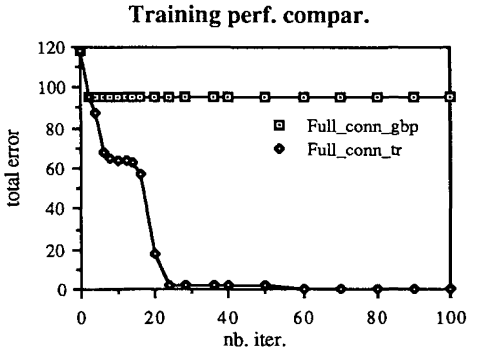



**Figure 13. — Full_conn_gbp : correspond to the net in figure 12 trained by GBP algorithm. Full conn_tr : correspond to the net in figure 12 trained by TR algorithm.**

## DISCUSSION AND CONCLUSION

During the experiments, we have remarked that :

1) If a network has many connections and if it is far from "saturation", the GBP algorithm is better in convergence speed, because in this case the dimension of the "decision space" is relatively high and the gradient calculated at each training step can give good decent to the objective function. Since the time for each learning step is much less for GBP (only up to one fifteenth of that for TR), it may be said that in this case GBP is better. But in real applications, a network may be asked to store very complicated (in terms of nonlinearity) input-output associations and to store as many as possible, in such situation GBP fails.

2) When GBP works, TR works too. When GBP does not work, the TR may still work unless all the weights of a net are initialized as almost the same value. The TR algorithm has a higher capacity to deal with the "hard" initial conditions.

3) In many cases, TR algorithm makes the Hessian of the objective function positive definite at the end of training. This means that the solution has good eigen-properties.

4) For any architecture of networks, the TR algorithm can always give a good solution. Due to the efficient use of the second order information, the TR algorithm can overcome many difficulties encountered by GBP in the choice of network architecture.

The TR algorithm can considerably reduce the human intervention in training the neural networks, and make the so called "automatic learning" (by neural net) more significant.

Alghough the use of the TR algorithm is time-consuming in each learning step, it can be compensated by a spectacular reduction in the total number of learning steps. The calculation of the Hessian (second order information) can be efficiently replaced by approximations as in our experiments. The real drawback of the TR algorithm is that it needs a great amount of physical memory as for any optimization method which uses the second order information. It is the memory for storing the Hessian matrix whose number of the elements is up to a power of two of the total number of connection weights.

This drawback can partially be made up by an efficient utilization of middle-size network as it is promised by the algorithm. We think that a completely satisfactory solution to the problem may be found in a good compromise between the use of TR principle and the choice of a class of intermediate variables (less numerous) to which the TR algorithm is

applied. There are many theoretical and practical problems in the "passage" between the intermediate variables and the "real" weights. This will be the subject of our next paper.

## APPENDIX A : PROOF TO LEMMA 2.2

It suffices to prove 2°.

If $\mu = -\lambda_1$ then the range of $H + \mu I$ is Lin $\{u_i : i \notin J_1\}$ which is equal to $\mathcal{N}(H - \lambda_1 I)^{\perp}$. Statement 2° is then immediate.

It is easy to verify that $d^*$ is a solution to (7) with $\mu = -\lambda_1$. It follows that the solution set of problem (7) is $d^* + \mathcal{N}(H - \lambda_1 I)$. Because $d^* \in \mathcal{N}(H - \lambda_1 I)^{\perp}$, $d^*$ is a solution of minimum norm to problem (7) with $\mu = -\lambda_1$ :

$$\|d^*\| = \min \{\|d\| : d \in d^* + \mathcal{N}(H - \lambda_1 I)\} .$$

In terms of pseudo-inverse (Lancaster 1969, Stewart 1973) we can write :

$$d^* = -(H - \lambda_1 I)^{+} g .$$

## APPENDIX B : CONVERGENCE OF PROJECTED GRADIENT METHOD

Here is the theoretical result on the convergence of $(x_k)$ defined at § 2.2.5) *b*) i) for $\rho_k = \rho$. It is obtained through the study of contraction property of $F_\rho$.

THEOREM B.1 : *Suppose H be positive definite. Let* $\lambda_1$ *(resp.* $\lambda_n$*) be the smallest (resp. largest) eigenvalue of H. Then* (*P*) *admits an unique solution and we have*

(i) $$\|F_\rho(x) - F_\rho(x)\| \leq q(\rho)\|x - y\|$$

*where* $q(\rho) = \text{Max } \{|1 - \rho\lambda_1|, |1 - \rho\lambda_n|\}$.

(ii) $$q(\rho) < 1 \quad \text{if and only if} \quad 0 < \rho < 2/\lambda_n .$$

*Moreover the optimal value* $\rho^*$ *of* $\rho$ *(i.e.* $\rho^*$ *minimizes* $q(\rho)$ *on* $]0, 2/\lambda_n[$*) is*

$$\rho^* = 2/(\lambda_1 + \lambda_n)$$

*and*

$$q^* = q(\rho^*) = (\lambda_n - \lambda_1)/(\lambda_n + \lambda_1) .$$

(iii) *Finally the sequence* $(x_k)$ *defined by (9) with* $\rho_k = \rho \in ]0, 2/\lambda_n[$ *for every k converges to the solution of* (*P*). □

*Proof:* It is well known that if $H$ is positive definite then $f(x)$ is strictly convex and

$$\lim_{\|x\| \to +\infty} f(x) = +\infty .$$

It follows that $(P)$ admits an unique solution (Cea 1971). Since the operator $\text{Proj}_C$ is non expansive (Auslender 1976), we have

$$\begin{aligned} \|F_\rho(x) - F_\rho(x)\| &\leq \|\text{Proj}_C\,(x - \rho\,\nabla f(x)) - \text{Proj}_C\,(y - \rho\,\nabla f(y))\| \\ &\leq \|x - \rho(Hx + g) - y + \rho(Hy + g)\| \\ &= \|(I - \rho H)(x - y)\| \leq \|I - \rho H\|\,\|x - y\| . \end{aligned}$$

Or $\|I - \rho H\| = \text{Max}\,\{|1 - \rho\lambda| : \lambda\}$ being an eigenvalue of $H$ (Gastinel 1966) it follows that

$$\|I - \rho H\| = \text{Max}\,\{|1 - \rho\lambda_1|, |1 - \rho\lambda_n|\}$$

because the function $\lambda \mapsto |1 - \rho\lambda|$ is convex. By simple calculation we obtain (ii). The last property (iii) is simple consequence of the contraction of operator $F_\rho$ for $\rho \in ]0, 2/\lambda_n[$. □

**APPENDIX C : THEORETICAL RESULTS ABOUT THE UZAWA METHOD (Cea 1971, Ekeland & Teman 1974, Laurent 1972)**

THEOREM C.1

(i) $\alpha \geq \beta$

(ii) *If* $(x^*, \mu^*)$ *is a saddle point of* $L(x, \mu)$ *then* $x^*$ *is a solution to* $(\mathscr{P})$ *and* $\mu^*$ *is a solution to* $(\mathscr{D})$ *and* $\alpha = \beta$.

(iii) *If there is* $x_0$ *such that* $\phi(x_0) < 0$ *then* $\alpha = \beta$ *and* $x^*$ *is a solution to* $(\mathscr{P})$ *if and only there is* $\mu^* \in \mathbb{R}^m_+$ *such that* $(x^*, \mu^*)$ *is a saddle point of* $L(x^*, \mu^*)$.

(iv) *If* $H$ *is positive definite or* $C$ *is bounded then* $S$ *is nonempty.* □

COROLLARY C.2 : *If there is* $x_0$ *such that* $\phi(x_0) \leq 0$ *then* $x^*$ *is a solution to* $(\mathscr{P})$ *if and only if there is* $\mu^* \in \mathbb{R}^m_+$ *such that*

(i) $f(x^*) + \langle \mu^*, \phi(x^*)\rangle = \inf\,\{f(x) + \langle \mu^*, \phi(x)\rangle : x \in \mathbb{R}^n\}$

(ii) $\phi(x^*) \leq 0$ and $\langle \mu^*, \phi(x^*)\rangle = 0$

*such an* $\mu^*$ *is necessarily solution to* $(\mathscr{D})$. □

COROLLARY C.3 : (i) *If* $\mu^*$ *is a solution to* $(\mathscr{D})$ *then* $S(\mu^*)$ *contains the solution set* $S$ *to* $(\mathscr{P})$. *More precisely we have*

$$S = \{x^* \in S(\mu^*) : \phi(x^*) \leq 0,\ \langle \mu^*, \phi(x^*)\rangle = 0\} .$$

(ii) *If H is positive definite then* $S(\mu)$ *is non empty and in fact reduced to a single point*

$$S(\mu) = \{s(\mu)\}. \quad \square$$

THEOREM C.4 : *If A is positive definite and if there is* $x_0$ *such that* $\phi(x_0)$ *then*

(i) $x_k = s(\mu_k)$.

(ii) $g$ *is differentiable and* $\nabla g(\mu) = \phi(s(\mu))$.

(iii) *The sequence* $(\mu_k)$ *converges to a solution* $\mu^*$ *to* $(\mathscr{D})$ *and the sequence* $(x_k)$ *converges to a solution* $x^*$ *to* $(\mathscr{P})$. $\square$

## REFERENCES


A. AUSLENDER (1976), *Optimisation, méthodes numériques*. Masson, Paris.

J. CEA (1971), *Optimisation : Théories et algorithmes*. Dunod.

A. R. CONN, N. GOULD & Ph. TOINT (1986), *Testing a class of methods for solving minimization problems with simple bounds on the variables*. Report n° 86-3, University of Waterloo.

J. E. DENNIS, R. B. SCHNABEL (1983), *Numerical methods for unconstrained optimization and nonlinear equations*. Printice-Hall.

I. S. DUFF, J. NOCEDAL & J. K. REID (1987), *The use linear programming for solution of sparse sets of nonlinear equations*. SIAM J. Sci. Stat. Comput. vol. 8, N° 2, pp. 99-108.

I. EKELAND & R. TEMAM (1974), *Analyse Convexe et problèmes variationnels*. Dunod, Gauthier-Villars.

R. FLETCHER (1980), *Practical Methods of Optimization*, vol. 1, John Wiley, New York.

F. FOGELMAN-SOULIE, P. GALLINARI, Y. LE CUN, S. THIRIA (1987), *Automata networks and artificial intelligence*. In F. Fogelman-Soulie, Y. Robert, M. Tchuente (Eds.), Computing on automata networks, Manchester University Press.

N. GASTINEL (1966), *Analyse numérique linéaire*. Hermann, Paris.

D. M. GAY (1981), *Computing optimal constrained steps*. SIAM J. Sci. Stat. Comput. 2, pp. 186-197.

P. E. GILL & W. MURRAY (1972), *Quasi-Newton methods for unconstrained optimization*. The Journal of the Institute of Mathematics and its Applications, vol. 9, pp. 91-108.

P. E. GILL, W. MURRAY & M. H. WRIGHT (1981), *Practical Optimization*. Academic Press.

M. D. HEBDEN (1973), *An algorithm for minimization using exact second derivatives*. Atomic Energy Research Establishment report T.P. 515, Harwell, England.

S. KANIEL & A. DAX (1979), *A modified Newton's method for unconstrained minimization.* SIAM J. Num. Anal., pp. 324-331.

P. LANCASTER (1969), *Theory of Matrix.* Academic Press, New York and London.

P. J. LAURENT (1972), *Approximation et Optimisation.* Hermann, Paris.

Y. LE CUN (1987), *Modèles connectionnistes de l'apprentissage.* Thèse de doctorat, Université de Paris VI.

MINOUX (1983), *Programmation Mathématique.* Tome 1, Dunod.

M. MINSKY & S. PAPERT (1969), *Perceptrons.* Cambridge, MA : MIT Press.

J. J. MORÉ (1978), *The Levenberg-Marquart algorithm : implementation and theory.* Lecture Notes in Mathematics 630, G. A. Waston, ed., Springer-Verlag, Berlin-Heidelberg-New York, pp. 105-116.

J. J. MORÉ (1983), *Recent developments in algorithm and software for Trust Region Methods.* Mathematical Programming, The State of the Art, Springer, Berlin, pp. 258-287.

J. J. MORÉ & D. C. SORENSEN (1979), *On the use of directions of negative curvature in a modified Newton method.* Math. Prog. 16, pp. 1-20.

J. J. MORÉ & D. C. SORENSEN (1981), *Computing a trust region step.* Argonne National Laboratory report, Argonne, Illinois.

H. MUKAI & E. POLAK (1978), *A second order method for unconstrained optimization.* J.O.T.A. vol. 26, pp. 501-513.

J. P. PENOT & A. ROGER, *Updating the spectrum of a real matrix.* Mathematics of Computation.

M. J. D. POWELL (1975), *Convergence properties of a class of minimization algorithms.* O. L. Mangazarian, R. R. Meyer, S. M. Robinson Editors, Nonlinear programming *2* pp. 1-27, Academic press, New York.

REINSCH (1967), *Smoothing by spline functions.* Numer. Math. 10, 177-183.

REINSCH (1971), *Smoothing by spline functions II.* Numer. Math. 16, 451-454.

D. E. RHUMELHART & J. C. MCCLELLAND (1986) (Eds.), *Parallel Distributed Processing.* Cambridge, MA : MIT Press.

F. ROBERT & S. WANG (1988), *Implementation of a Neural Network on a Hypercube F.P.S. T20.* Proceeding of IFIP WG 10.3 Working Conference on Parallel Processing. Pisa : Italy, 25-27 April. North-Holland.

R. T. ROCKAFELLAR (1970), *Convex Analysis.* Princeton University Press, Princeton, New Jersey.

A. ROGER (1987), *Mise à jour du spectre d'une matrice symétrique.* Rapport de recherche SNEA (P), n° AR/87-970.

S. ROUSSET, A. SCHREIBER & S. WANG (1988), *Modélisation et simulation connexionniste de l'identification des visages en contexte.* Le système FACENET RR 742 -M-. IMAG Grenoble.

G. A. SHULTZ, R. B. SCHNABEL & R. H. BYRD (1985), *A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties.* SIAM Journal on Numerical Analysis 22, pp. 47-67.

G. A. SHULTZ, R. B. SCHNABEL & R. H. BYRD (1988), *Approximate solution of the trust region problem by minimization over two-dimensional subspaces Mathematical Programming.* Vol. 40, pp. 247-263, North-Holland.

D. C. SORENSEN (1982), *Newton's method with a model trust region modification.* SIAM J. Numer. Anal. vol. 19, n°2, pp. 409-426.

G. W. STEWART (1973), *Introduction to matrix computation.* Academic Press, New York.

S. WANG (1988), *Implementation of threshold automata networks with multilayers on a Hypercube F.P.S. T20.* RR 725 -M-. IMAG, Grenoble.

S. WANG, H. YÉ & F. ROBERT (1988), *A PNML neural network for isolated words recognition.* Proceedings of nEuro '88. First european conference on neural network, 6-9 Juin 1988 : Paris.

Y. YUAN (1984), *An example of only linear convergence of trust region algorithms for nonsmooth optimization.* IMA Journal of Numerical Analysis 4, pp. 327-335.

Y. YUAN (1985), *On the superlinear convergence of a trust region.algorithm for nonsmooth optimization.* Mathematical Programming, vol. 3, pp. 269-285. North-Holland.