

Matrix Algebras in Quasi-Newtonian Algorithms for Optimal Learning in Multi-Layer Perceptrons

Carmin Di Fiore - Stefano Fanelli - Paolo Zellini

Dipartimento di Matematica
Università di Roma "Tor Vergata"
Via della Ricerca Scientifica
00133 Roma (Italy)

Abstract

In this work the authors implement in a Multi-Layer Perceptron (MLP) environment a new class of Quasi-Newtonian (QN) methods. The algorithms proposed in the present paper use in the iterative scheme of a generalized BFGS-method a family of matrix algebras, recently introduced for optimal preconditioning. This novel approach allows to construct methods having an $O(n \log_2 n)$ complexity.

Numerical experiences compared with the performances of the best QN-algorithms known in the literature confirm the effectiveness of these new optimization techniques.

1. Introduction

In [1],[8] two variants of QN-BFGS algorithms [4] were introduced to perform optimal learning in a MLP-network. Both methods, respectively named OSS and OSSV, were essentially memory-less modifications of the classical iterative procedure to evaluate the Hessian matrix or its inverse.

The main advantage of OSS-OSSV's modified scheme is to reduce the number of operations per step from $O(n^2)$ to $O(n)$ (being n the number of neurons), by maintaining a suitable amount of Second Order information.

On the other hand, by the very nature of the memory-less approach, the number of steps k^* to obtain a satisfactory approximation of the Hessian matrix (or equivalently of its inverse) can be extremely large for some classes of learning problems (see [8] for details). However, as underlined before, since the alternative classical BFGS method requires $O(n^2)$ operations per step, by denoting with k^{**} the corresponding number of iterations of the BFGS procedure, the inequality:

$$k^*O(n) \ll k^{**}O(n^2)$$

is always satisfied in general. In order to improve the efficiency of BFGS methods, this paper takes into account a preconditioning scheme inspired to some preliminary ideas developed in [15].

An effective preconditioning technique to reduce the number of steps of iterative methods for positive definite linear systems $A\mathbf{x} = \mathbf{b}$, consists in determining the preconditioner in an algebra L of matrices simultaneously diagonalized by a fast transform (FFT, trigonometric or Hartley-type [2],[5],[7],[9]), structurally close to A . If L_A denotes the best L -approximation of A in the Frobenius norm, this technique guarantees, for particular classes of matrices A , a superlinear rate of convergence of conjugate gradient methods for the system preconditioned by L_A (see [3],[7],[9]).

The principal aim of this paper is to show some preliminary results concerning the implementation of a *new class of QN-methods with memory*, named LQN, involving suitable L_A approximations and thus requiring $O(n \log_2 n)$ operations per step.

Since the amount of Second Order information utilized by LQN is considerably stronger than in OSS-OSSV, we expect that, by indicating with k^{***} the number of steps required by LQN to obtain the desired precision, the inequality:

$$k^{***}O(n \log_2 n) < k^*O(n)$$

can be verified in several cases.

The preliminary numerical experiences illustrated in Section 5 are in fact rather encouraging and confirm that the innovative methods are effective and have a strong flexibility.

Thus, they deserve further investigations (see [6]).

2. Optimal approximation by matrix algebras

Let Q be a $n \times n$ unitary matrix, i.e. $Q^H = Q^{-1}$. Set

$$L = \{Qd(\mathbf{z})Q^H : \mathbf{z} \in \mathbf{C}^n\}$$

where $d(\mathbf{z}) = \text{diag}(z_i, i=1, \dots, n)$, $\mathbf{z} = [z_1 \dots z_n]^T$. For an arbitrary matrix $A \in \mathbf{C}^{n \times n}$ consider the minimum problem

$$\min_{X \in L} \|X - A\|_F \quad (1)$$

where $\|\cdot\|_F$ is the Frobenius norm, i.e. the norm induced by the inner product $(X, Y) = \sum \bar{x}_{ij} y_{ij}$ on $\mathbf{C}^{n \times n}$.

We know from preconditioning literature (beginning with T.Chan [3]) that there exists a unique matrix L_A in L solving (1); this is called the best least squares (l.s.) fit to A from the space L . Here below are listed some properties of L_A [7].

$$L_A = Qd(\mathbf{z}_A)Q^H, \quad [\mathbf{z}_A]_i = (Q\mathbf{e}_i)^H A Q\mathbf{e}_i; \quad (2)$$

$$\mathbf{z}_{\mathbf{xy}^T} = d(Q^H \mathbf{x}) Q^T \mathbf{y}, \quad \mathbf{x}, \mathbf{y} \in \mathbf{C}^n; \quad (3)$$

$$L_A = \sum_{k=1}^n [B^{-1} \mathbf{c}]_k J_k, \quad b_{ij} = (J_i, J_j), \quad c_i = (J_i, A), \quad i, j = 1, \dots, n; \quad (4)$$

$$L_{\alpha A + \beta B} = \alpha L_A + \beta L_B, \quad \alpha, \beta \in \mathbf{C}, \quad A, B \in \mathbf{C}^{n \times n}; \quad (5)$$

$$A^H = A \Rightarrow (L_A)^H = L_A \quad \text{and}$$

$$\min \lambda(A) \leq \lambda(L_A) \leq \max \lambda(A); \quad (6)$$

$$A \text{ and } J_k \text{ real} \Rightarrow L_A \text{ real}, \quad (7)$$

where $\{J_k\}_{k=1, \dots, n}$ is a basis for L and $\lambda(X)$ denotes the generic eigenvalue of X .

We are interested in cases where Q defines a fast discrete transform (see Sections 3 and 4), i.e. when the matrix-vector products $Q\mathbf{z}$ and $Q^T \mathbf{z}$ are computable in $O(n \log_2 n)$ arithmetic operations. In the following are reported some examples of such particular matrices $Q = (q_{ij})_{i,j=1, \dots, n}$ with a description of the corresponding matrix algebras L (other examples can be found in [2],[5],[7],[9]).

$$\begin{aligned} & \sqrt{1/n} \omega^{(i-1)(j-1)}, \quad C_1 = \{X: XP_1 = P_1 X\}; \\ & \sqrt{1/n} \rho^{i-1} \omega^{(i-1)(j-1)}, \quad C_{-1} = \{X: XP_{-1} = P_{-1} X\}; \\ & \sqrt{2/(n+1)} \sin(ij\pi/(n+1)), \quad \tau = \{X: XM = MX\}; \\ & \sqrt{1/n} (\cos(2(i-1)(j-1)\pi/n) \pm \sin(2(i-1)(j-1)\pi/n)), \\ & \quad H = C_1^S + JP_1 C_1^{SK}; \\ & \sqrt{1/n} (\cos((i-1)(2j-1)\pi/n) \pm \sin((i-1)(2j-1)\pi/n)), \\ & \quad K = C_{-1}^S + JP_{-1} C_{-1}^{SK} \quad [5]; \\ & \sqrt{2/n} \sin((2i-1)(2j-1)\pi/(2n)) \quad j=1, \dots, (n-1)/2', \\ & \quad (-1)^{j-1} \sqrt{1/n} \quad j=(n+1)/2 \quad (n \text{ odd}), \\ & \sqrt{2/n} \cos((2i-1)(2j-1)\pi/(2n)) \quad j \in (n+3)/2 \vee, \dots, n, \\ & \quad \mu = C_{-1}^S + JC_{-1}^S \quad [5],[7]. \end{aligned}$$

Here, $\rho = e^{-in/n}$, $\omega = \rho^2$, $C_{\pm 1}^S = \{X \square C_{\pm 1} : X^T = X\}$, $C_{\pm 1}^{SK} = \{X \square C_{\pm 1} : X^T = -X\}$, $J = (\delta_{i, n+1-j})_{i,j=1, \dots, n}$

$$P_\varepsilon = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & & 1 & \\ \varepsilon & 0 & 0 & 0 \end{bmatrix}$$

and $M = P_0 + P_0^T$.

C_1 is the space of *circulant* matrices, while the matrix Q defining C_1 is the well known discrete Fourier transform F . The matrices Q defining τ , H and K , denoted, respectively, by S , Q_+ and Q_- , are known as the discrete sine, Hartley and skew-Hartley transforms.

If L is closed under transposition and $A = A^T$, then $L_A = (L_A)^T$ (by the uniqueness of L_A). If $L = C_{\pm 1}$ this remark leads to a useful formula for $(C_{\pm 1})_A$, alternative to (2) [6]:

$$A = A^T \Rightarrow (C_{\pm 1})_A = Q_{\pm} d(\mathbf{w}_A) Q_{\pm}^T, \quad (8)$$

where $\mathbf{w}_A = (1/2) R_{\pm} \sum_{k=1}^n d(Q_{\pm}^T A \mathbf{e}_k) Q_{\pm}^T \mathbf{e}_k$, $R_+ = I + JP_1$ and $R_- = I + J$; observe that

$$\begin{aligned} \mathbf{w}_{\mathbf{xx}^T} &= (1/2) R_{\pm} d(Q_{\pm}^T \mathbf{x}) Q_{\pm}^T \mathbf{x} \quad \text{and} \\ \mathbf{w}_{\mathbf{xy}^T + \mathbf{yx}^T} &= R_{\pm} d(Q_{\pm}^T \mathbf{x}) Q_{\pm}^T \mathbf{y}, \quad \mathbf{x}, \mathbf{y} \in \mathbf{C}^n. \end{aligned} \quad (9)$$

One can also have some information about the quality of the approximation to A of L_A , $L = C_1, C_{-1}, H, K, \mu$. In particular, if $A = A^T$, then $H_A (K_A, \mu_A)$, is a better fit than $(C_1)_A ((C_{-1})_A)$, because the latter matrix is symmetric and thus belongs to $H (K, \mu)$. Moreover, if $A = A^T = JAJ$, then the following result holds [7]:

$$\|\mu_A - A\|_F \leq \|K_A - A\|_F.$$

3. The methods BFGS, OSS-OSSV and alternative strategies

Let $f: \mathbf{R}^n \rightarrow \mathbf{R}$ be and consider the following *basic* algorithms **1a** and **1b** for the minimization of f .

Given $\mathbf{x}_0 \in \mathbf{R}^n$, define: $\mathbf{g}_0 = \nabla f(\mathbf{x}_0)$, \mathbf{s}_0 by **(1a)** $A_0 \mathbf{s}_0 = -\mathbf{g}_0$ or by **(1b)** $\mathbf{s}_0 = -B_0 \mathbf{g}_0$, A_0 and B_0 real symmetric positive definite (r.s.p.d.) matrices, and $\lambda_0 =$ suitable initial guess.

Then, for $k=0, 1, \dots$, define:

$$\begin{aligned} * \quad \mathbf{p}_k &= \lambda_k \mathbf{s}_k, \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \mathbf{p}_k, \\ \text{if } f(\mathbf{x}_{k+1}) &> \varphi_2 f(\mathbf{x}_k): \lambda_k = \varphi_3 \lambda_k \text{ and go to } *, \\ \mathbf{g}_{k+1} &= \nabla f(\mathbf{x}_{k+1}), \\ \mathbf{y}_k &= \mathbf{g}_{k+1} - \mathbf{g}_k, \\ \mathbf{s}_{k+1} &\text{ by} \\ \mathbf{(1a)} \quad A_{k+1} \mathbf{s}_{k+1} &= -\mathbf{g}_{k+1}, \text{ where} \\ A_{k+1} &= \tilde{A}_k + (1/\mathbf{y}_k^T \mathbf{p}_k) \mathbf{y}_k \mathbf{y}_k^T - \\ & \quad (1/\mathbf{p}_k^T \tilde{A}_k \mathbf{p}_k) \tilde{A}_k \mathbf{p}_k \mathbf{p}_k^T \tilde{A}_k, \end{aligned} \quad (10)$$

or by

$$\begin{aligned} \mathbf{(1b)} \quad \mathbf{s}_{k+1} &= -B_{k+1} \mathbf{g}_{k+1}, \text{ where} \\ B_{k+1} &= \tilde{B}_k - (1/\mathbf{y}_k^T \mathbf{p}_k) (\tilde{B}_k \mathbf{y}_k \mathbf{p}_k^T + \mathbf{p}_k \mathbf{y}_k^T \tilde{B}_k) + \\ & \quad (1 + \mathbf{y}_k^T \tilde{B}_k \mathbf{y}_k / \mathbf{y}_k^T \mathbf{p}_k) (1/\mathbf{y}_k^T \mathbf{p}_k) \mathbf{p}_k \mathbf{p}_k^T, \quad (11) \\ \lambda_{k+1} &= \varphi_1 \lambda_k. \end{aligned}$$

Here $\{\tilde{A}_k\}$ and $\{\tilde{B}_k\}$ are two sequences of r.s., possibly p.d. matrices, built from the sequences $\{A_k\}$ and $\{B_k\}$ or arbitrary, and φ_i are fixed real numbers such that $\varphi_2 \geq 1$, $0 < \varphi_3 < 1$, $\varphi_1 > 1$.

The algorithms **1a** and **1b** are two examples of *secant* methods because the matrices A_{k+1} and B_{k+1}^{-1} in (10),(11) solve the secant equation $X\mathbf{p}_k = \mathbf{y}_k$. Notice that if $B_0 = A_0^{-1}$ and $\tilde{B}_k = \tilde{A}_k^{-1}$, then, by the Sherman-Morrison-Woodbury formula [4, chapter 8], $B_{k+1} = A_{k+1}^{-1}$, that is, the procedures **1a** and **1b** coincide. In particular, for $\tilde{A}_k = A_k$ and $\tilde{B}_k = B_k$, **1a** and **1b** yield the well known BFGS method [4] while, for $\tilde{A}_k = \tilde{B}_k = I$, they yield two "memory-less" methods named OSS-OSSV [1],[8].

The matrices A_{k+1} and B_{k+1} in (10),(11) are obviously r.s.. A necessary and sufficient condition for A_{k+1} (B_{k+1}) to be p.d. is that $\mathbf{y}_k^T \mathbf{p}_k > 0$, provided that \tilde{A}_k (\tilde{B}_k) is p.d. (see [4]). Thus, if $\mathbf{y}_k^T \mathbf{p}_k < 0$, the line search \mathbf{s}_{k+1} proposed by **1a** (**1b**) may be an increasing direction. In this case, one could redefine \mathbf{s}_{k+1} as a decreasing direction and go on. However, since \tilde{A}_k (\tilde{B}_k) can be indefinite, we proceed as follows.

If the proposed \mathbf{s}_{k+1} and λ_{k+1} are such that $f(\mathbf{x}_{k+1} + \lambda_{k+1} \mathbf{s}_{k+1}) > f(\mathbf{x}_{k+1})$, then (without checking the reason, λ_{k+1} too large or/and \mathbf{s}_{k+1} increasing direction) redefine \mathbf{s}_{k+1} as a *sure* decreasing direction by

$$\mathbf{s}_{k+1} = -\mathbf{g}_{k+1} \text{ (algorithms 1ag and 1bg)}$$

or, alternatively, when $\{\tilde{A}_k\}$ and $\{\tilde{B}_k\}$ are p.d., by

$$\tilde{A}_k \mathbf{s}_{k+1} = -\mathbf{g}_{k+1} \text{ (algorithm 1a\tilde{g}) and}$$

$$\mathbf{s}_{k+1} = -\tilde{B}_k \mathbf{g}_{k+1} \text{ (algorithm 1b\tilde{g}) ,}$$

and then do backtracking along this direction.

The adaptive reset of the line search adopted in algorithms **1ag**, **1bg** (see also the end of Section 4), is similar to the procedure described in [12] and was exploited in [8] to introduce OSSV as an improvement of OSS [1]. In the latter method the direction \mathbf{s}_{k+1} is restarted to $-\mathbf{g}_{k+1}$ every n steps.

If the sequence $\{\tilde{A}_k\}$ ($\{\tilde{B}_k\}$) is built from the sequence $\{A_k\}$ ($\{B_k\}$) - \tilde{A}_k and \tilde{B}_k may be suitable, for instance least squares, approximations of A_k and B_k -, then an interesting alternative to the basic methods **1a** and **1b** can be introduced by defining, for any k ($k \geq -1$), \mathbf{s}_{k+1} by

$$(2a) \tilde{A}_{k+1} \mathbf{s}_{k+1} = -\mathbf{g}_{k+1}$$

or by

$$(2b) \mathbf{s}_{k+1} = -\tilde{B}_{k+1} \mathbf{g}_{k+1}.$$

The basic algorithms **2a** and **2b** so obtained may require in general more iterations to converge than **1a** and **1b** because they are not secant methods. However, each step of the former algorithms might be cheaper in terms of computation time (see

Section 4). A possible decreasing direction for **2a** and **2b**, in an adaptive reset of the line search, is

$$\mathbf{s}_{k+1} = -\mathbf{g}_{k+1} \text{ (algorithms 2ag and 2bg)}.$$

4. Updating the best l.s. fit to the Hessian approximation

By (6) of Section 2, if A is a real symmetric matrix, then L_A , i.e. the best l.s. fit to A from $L = \{Qd(\mathbf{z})Q^H: \mathbf{z} \in \mathbb{C}^n\}$, Q unitary, is hermitian and such that:

$$\min \lambda(A) \leq \lambda(L_A) \leq \max \lambda(A).$$

In particular L_A is p.d. whenever A is p.d..

This result suggests the choices $\tilde{A}_k = L_{A_k}$ and $\tilde{B}_k = L_{B_k}$ in the basic algorithms **1a**, **1b**, **2a** and **2b** of the previous Section. In particular, the new algorithms **1a** and **1b** so obtained appear as a sort of compromise between the BFGS and the OSS-OSSV methods, in which, respectively, the proposed direction \mathbf{s}_{k+1} depends and does not depend *explicitly* upon the old "Hessian approximation" A_k ($=B_k^{-1}$). All the new methods are referred here as LQN.

From now on we assume $\tilde{A}_k = L_{A_k}$ and $\tilde{B}_k = L_{B_k}$ in (10) and (11). We also assume that L is spanned by real matrices so that, if A_k (B_k) is r.s. then also L_{A_k} (L_{B_k}) and therefore A_{k+1} (B_{k+1}) in (10) ((11)) are r.s. (see (7) in Section 2).

Six possible new minimization methods follow from the arguments of Section 3:

$$\text{I. 1ag for } \tilde{A}_k = L_{A_k}, \quad \text{I'. 1bg for } \tilde{B}_k = L_{B_k},$$

$$\text{II. 1a\tilde{g} for } \tilde{A}_k = L_{A_k + X_k}, \quad \text{II'. 1b\tilde{g} for } \tilde{B}_k = L_{B_k + Y_k},$$

III. **2ag** for $\tilde{A}_{k+1} = L_{A_{k+1}}$ with A_{k+1} as in (10), and

III'. **2bg** for $\tilde{B}_{k+1} = L_{B_{k+1}}$ with B_{k+1} as in (11).

The matrix X_k (Y_k) in II (II') is null if L_{A_k} (L_{B_k}) is p.d.; otherwise it is chosen such that \tilde{A}_k (\tilde{B}_k) is p.d.. For example $X_k = \alpha_k I$ ($Y_k = \beta_k I$) with α_k (β_k) zero or sufficiently large.

Notice that since the matrices L_{A-1} and L_A^{-1} are generally not equal, the apostrophized methods differ from the non apostrophized ones.

We show now that each one of the methods I, I', II, II', III, III' requires at each step the computation of two transforms involving the matrix Q plus $O(n)$ arithmetic operations (a.o.); thus, if Q defines a fast discrete transform (for instance Q is one of the matrices listed in Section 2), then $O(n \log_2 n)$ a.o. are sufficient to perform a step. Since the BFGS and the OSS-OSSV methods require, respectively, $O(n^2)$ and $O(n)$ a.o. per step, in order to evaluate the competitiveness of the new LQN methods one should study their rate of convergence. See

Section 5 where some experimental results are reported, and [6].

By the linearity (5) of L , the identities (10) and (11) yield, respectively,

$$\mathbf{z}_{A_{k+1}} = \mathbf{z}_{A_k} + (1/\mathbf{y}_k^T \mathbf{p}_k) \mathbf{z}_{\mathbf{y}_k \mathbf{y}_k^T} - \\ (1/\mathbf{p}_k^T L_{A_k} \mathbf{p}_k) \mathbf{z}_{L_{A_k} \mathbf{p}_k} (\bar{L}_{A_k} \mathbf{p}_k)^T$$

and

$$\mathbf{z}_{B_{k+1}} = \mathbf{z}_{B_k} - (1/\mathbf{y}_k^T \mathbf{p}_k) (\mathbf{z}_{L_{B_k} \mathbf{y}_k} \mathbf{p}_k^T + \mathbf{z}_{\mathbf{p}_k} (\bar{L}_{B_k} \mathbf{y}_k)^T) + \\ (1 + \mathbf{y}_k^T L_{B_k} \mathbf{y}_k / \mathbf{y}_k^T \mathbf{p}_k) (1/\mathbf{y}_k^T \mathbf{p}_k) \mathbf{z}_{\mathbf{p}_k} \mathbf{p}_k^T$$

(see the definition (2) of Section 2). Then the expression for $\mathbf{z}_{\mathbf{x} \mathbf{y}^T}$ in (3) leads to the following relations which imply that $L_{A_{k+1}}$ ($L_{B_{k+1}}$) is computable from L_{A_k} (L_{B_k}) in $O(n \log_2 n)$ a.o..

$$\mathbf{z}_{A_{k+1}} = \mathbf{z}_{A_k} + (1/\lambda_k) (1/\mathbf{y}_k^T \mathbf{s}_k) |Q^H \mathbf{y}_k|^2 - \\ (1/\mathbf{z}_{A_k}^T |Q^H \mathbf{s}_k|^2) d(|\mathbf{z}_{A_k}|^2) |Q^H \mathbf{s}_k|^2, \quad (12)$$

$$\mathbf{z}_{B_{k+1}} = \mathbf{z}_{B_k} - \\ (1/\mathbf{y}_k^T \mathbf{s}_k) [2 \operatorname{Re}(d(\mathbf{z}_{B_k}) d(Q^H \mathbf{y}_k) Q^T \mathbf{s}_k) - \\ (\lambda_k + \mathbf{z}_{B_k}^T |Q^H \mathbf{y}_k|^2 / \mathbf{y}_k^T \mathbf{s}_k) |Q^H \mathbf{s}_k|^2]. \quad (13)$$

Here $|\mathbf{z}|^2$ is the vector whose i th entry is $|z_i|^2$.

Moreover, one can write formulas for the directions \mathbf{s}_{k+1} in **1a** and **2a**. These are, respectively,

$$Q^H \mathbf{s}_{k+1} = -d(\mathbf{z}_{A_k}^{-1}) Q^H \mathbf{g}_{k+1} + \\ [-(\lambda_k + (\mathbf{z}_{A_k}^{-1})^T |Q^H \mathbf{y}_k|^2 / \mathbf{y}_k^T \mathbf{s}_k) (\mathbf{s}_k^T \mathbf{g}_{k+1} / \mathbf{y}_k^T \mathbf{s}_k) + \\ (\mathbf{z}_{A_k}^{-1})^T d(Q^H \mathbf{g}_{k+1}) Q^T \mathbf{y}_k / \mathbf{y}_k^T \mathbf{s}_k] Q^H \mathbf{s}_k + \\ [\mathbf{s}_k^T \mathbf{g}_{k+1} / \mathbf{y}_k^T \mathbf{s}_k] d(\mathbf{z}_{A_k}^{-1}) Q^H \mathbf{y}_k, \quad (14) \\ Q^H \mathbf{s}_{k+1} = -d(\mathbf{z}_{A_{k+1}}^{-1}) Q^H \mathbf{g}_{k+1} \quad (15)$$

where \mathbf{z}^{-1} denotes the vector whose i th entry is z_i^{-1} . By simply replacing, in (14), $\mathbf{z}_{A_k}^{-1}$ with \mathbf{z}_{B_k} , and, in (15), $\mathbf{z}_{A_{k+1}}^{-1}$ with $\mathbf{z}_{B_{k+1}}$, we also obtain the formulas representing the directions \mathbf{s}_{k+1} in the algorithms **1b** and **2b**. All these formulas state that *the new direction \mathbf{s}_{k+1} is computable from the old one, \mathbf{s}_k , in $O(n \log_2 n)$ a.o..*

Now, by using (12), (13), (14), (15) one is able to give an explicit form to the algorithms I, I', II, II', III, III'. Only I, III are described here in detail. I' and III' are obtained analogously. In the algorithm II, notice that the choice $\tilde{A}_{k+1} = L_{A_{k+1}} + X_{k+1}$ is equivalent to the redefinition of $\mathbf{z}_{A_{k+1}}$ in (12) as a positive vector:

$$\mathbf{z}_{\tilde{A}_{k+1}} = \mathbf{z}_{A_{k+1}} + \mathbf{v}, \quad \mathbf{v} \in \mathbf{R}^n \text{ such that } (\mathbf{z}_{\tilde{A}_{k+1}})_i > 0, \forall i.$$

An identical remark holds for II'. For more details on I', III', II, II' see [6].

The algorithms I, III are shown with the same adaptive reset of the line search adopted in [8]. For sake of simplicity $A_0 = I$ and then $L_{A_0} = I$ ($\mathbf{z}_{A_0} = \mathbf{e}$

$[1 \ 1 \ \dots \ 1]^T$), i.e. the first line search direction is $\mathbf{s}_0 = -\nabla f(\mathbf{x}_0)$. However, if \mathbf{x}_0 is in the neighbourhood of a minimum for f , it could be more convenient to choose A_0 as an "approximation" of the Hessian $\nabla^2 f(\mathbf{x}_0)$ in order to increase the quasi-newtonian properties of the methods. In this case, it is useful to know that, if Q is one of the matrices listed in Section 2 and if A_0 is *structured* (for example A_0 is a low-rank perturbation of a Toeplitz plus Hankel matrix), then the complexity of the computation of \mathbf{z}_{A_0} is at most $O(n \log_2 n)$ [7],[9] – as the complexity of the generic step – while it is $O(n^2)$ for an arbitrary A_0 .

Given $\mathbf{x}_0 \in \mathbf{R}^n$, define: $\mathbf{g}_0 = \nabla f(\mathbf{x}_0)$, $\mathbf{z}_{A_0} = \mathbf{e}$, $Q^H \mathbf{s}_0 = -Q^H \mathbf{g}_0$, $\mathbf{s}_0 = -\mathbf{g}_0$, $sdg = \text{FALSE}$, $\lambda_0 =$ suitable initial guess. Then, for $k=0, 1, \dots$, define: $\mathbf{p}_k = \lambda_k \mathbf{s}_k$, $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$, IF $f_{k+1} > 1.05 f_k$ THEN (IF sdg THEN $(\mathbf{z}_{A_k} = \mathbf{e}, \quad Q^H \mathbf{s}_k = -Q^H \mathbf{g}_k, \quad \mathbf{s}_k = -\mathbf{g}_k, \quad sdg = \text{FALSE})$

REPEAT

$$\lambda_k = 0.7 \lambda_k,$$

$$\mathbf{p}_k = \lambda_k \mathbf{s}_k,$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$$

UNTIL NOT $f_{k+1} > 1.05 f_k$,

$$\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1}),$$

$$\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k,$$

IF $f_{k+1} \leq f_k$ AND $\mathbf{p}_k^T \mathbf{y}_k \neq 0$ THEN

$$(12),$$

$$(14) \text{ or } (15),$$

$$\mathbf{s}_{k+1} = Q(Q^H \mathbf{s}_{k+1}), \quad sdg = \text{TRUE}, \quad \lambda_{k+1} = 1.05 \lambda_k)$$

ELSE

$$(\mathbf{z}_{A_{k+1}} = \mathbf{e}, \quad Q^H \mathbf{s}_{k+1} = -Q^H \mathbf{g}_{k+1}, \quad \mathbf{s}_{k+1} = -\mathbf{g}_{k+1},$$

$$sdg = \text{FALSE}, \quad \lambda_{k+1} = 0.7 \lambda_k).$$

Here sdg (\mathbf{s} different from gradient) is a boolean variable and $f_k = f(\mathbf{x}_k)$.

If the matrix Q is not real, then the algorithms I, I', II, II', III, III' use complex arithmetic to generate real vectors. In the particular cases $q_{ij} = (1/|n|) \omega^{(i-1)(j-1)}$ and $q_{ij} = (1/|n|) \rho^{i-1} \omega^{(i-1)(j-1)}$ (see Section 2) one could rewrite the methods so that only real operations are needed, by using the analogs of formulas (12), (13), (14), (15) derived from (5),(8),(9) of Section 2 [6].

4. Numerical results

The following tables illustrate the number of steps required by the OSSV and by the LQN methods I (III), $Q = Q_+, Q_-, S, F$, to verify the inequality $\|\square f(\mathbf{x}_k)\|_{\square} < 10^{-t}$, $t=3, 4, 5, 6$, where $f(\mathbf{x})$ is the error function of a I-H-O network for learning a set of functions $\mathbf{v} = \mathbf{v}(\mathbf{u})$, $\mathbf{u} \in \mathbf{R}^1$, $\mathbf{v} \in \mathbf{R}^0$, and \mathbf{x} is the vector

of weights ($\lambda_0=0.1$, $[\mathbf{x}_0]_i$ random in $[-1,1]$). The benchmarks considered are the same selected in [8],[10] (see [8] for other references). Performances of LQN-methods are clearly competitive in these preliminary experiences.

2-2-1, XOR				
	10^{-3}	10^{-4}	10^{-5}	10^{-6}
OSSV	84	86	88	90
Q_+	41	45	48	51
Q_-	59	63	65	68
S	42	47	53	200
F	49	53	60	62

8-16-3, $v_1(\mathbf{u})=(u_1u_2+u_3u_4+u_5u_6+u_7u_8)/4$, $v_2(\mathbf{u})=(u_1+u_2+u_3+u_4+u_5+u_6+u_7+u_8)/8$, $v_3(\mathbf{u})=(1-v_1(\mathbf{u}))^{1/2}$ (50 random points in $(0,1)^8$)				
	10^{-3}	10^{-4}	10^{-5}	
OSSV	469	1675	7038	
Q_+	452	1797	8551	
Q_-	447	1745	6041	
S	338	1082	6223	
F	373	1102	7533	

1-2-1, $v(u)=.5\sin(\pi u/5)+.5\sin^2(\pi u/10)$ ($i/21, i=1,\dots,20$)				
	10^{-3}	10^{-4}	10^{-5}	10^{-6}
OSSV	69	144	4022	5840
Q_+	41 (68)	57 (73)	1943	2826
Q_-	42 (59)	45 (308)	2283	2686
S	43 (64)	61 (154)	2357	5111
F	42 (69)	52 (75)	3243	5523

1-2-1, $v(u)=3.95u(1-u)$ ($i/51, i=1,\dots,50$)				
	10^{-3}	10^{-4}	10^{-5}	10^{-6}
OSSV	40	43	47	50
Q_+	39	41	50	53
Q_-	89	292	409	502
S	108	110	937	1041
F	33	39	41	43

References

- [1] R.Battiti, *First and second-order methods for learning: between steepest descent and Newton's methods*, Neural Computation, 4(1992), pp.141-166.
- [2] E.Bozzo and C.Di Fiore, *On the use of certain matrix algebras associated with discrete trigonometric transforms in matrix displacement decomposition*, SIAM J.Matrix Anal. Appl., 16(1995), pp.312-326.
- [3] T.Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Stat. Comput., 9(1988), pp.766-771.
- [4] J.E.Dennis and R.B.Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
- [5] C.Di Fiore, *Matrix algebras and displacement decompositions*, SIAM J. Matrix Anal. Appl., to appear.
- [6] C.Di Fiore, S.Fanelli and P.Zellini, *Matrix algebras in secant methods for unconstrained minimization*, in preparation.

[7] C.Di Fiore and P.Zellini, *Matrix algebras in optimal preconditioning*, submitted to Linear Algebra Appl., 1999.

[8] S.Fanelli, P.Paparo and M.Protasi, *Improving performances of Battiti-Shanno's quasi-newtonian algorithms for learning in feed-forward neural networks*, in Proc. 2nd Australian and New Zealand Conference on Intelligent Information Systems, Brisbane, pp.115-119,1994.

[9] T.Kailath and V.Olshevsky, *Displacement structure approach to discrete trigonometric transform based preconditioners of G.Strang type and of T.Chan type*, Calcolo, 33(1996), pp.191-208.

[10] A.A.Minai and R.D.Williams, *Back-propagation heuristic: a study of the extended delta bar delta algorithm*, in IJCNN, San Diego, Vol.1, pp.595-600, 1990.

[11] M.Møller, *A scaled conjugate gradient algorithm for fast supervised learning*, Neural Networks, 6(4) (1993), pp.525-533.

[12] T.P.Vogl, J.K.Mangis, A.K.Rigler, W.T.Zink and D.L.Alkon, *Accelerating the convergence of the back-propagation method*, Biological Cybernetics, 59(1988), pp.257-263.