A survey on algorithms for training artificial neural networks

Technical Report · September 2008

CITATIONS
26

READS
1,168

2 authors:

Ioannis Livieris
CORE INNOVATION CENTER
88 PUBLICATIONS **1,098** CITATIONS
SEE PROFILE

P. E. Pintelas
University of Patras
183 PUBLICATIONS **6,451** CITATIONS
SEE PROFILE

TECHNICAL REPORT

A SURVEY ON ALGORITHMS FOR TRAINING ARTIFICIAL NEURAL NETWORKS

I.E. Livieris[1,†] P. Pintelas[1,2]

No. TR08-01

University of Patras
Department of Mathematics
GR-265 04 Patras, Greece.
http://www.math.upatras.gr/

---

[1]University of Patras, Department of Mathematics, GR-265 04, Patras, Hellas.
[2]Educational Software Development Laboratory, University of Patras, GR-265 04 Patras, Hellas.
e-mail: livieris|pintelas@math.upatras.gr
† corresponding author

Technical Report
No. TR08-01

A survey on algorithms for training artificial neural networks

I.E. Livieris P. Pintelas

September 2008

Abstract. Literature review corroborates that artificial neural networks are being successfully applied in a variety of regression and classification problems. Due of their ability to exploit the tolerance for imprecision and uncertainty in real-world problems and their robustness and parallelism, artificial neural networks have been increasingly used in many applications. It is well-known that the procedure of training a neural network is highly consistent with unconstrained optimization theory and many attempts have been made to speed up this process. In particular, various algorithms motivated from numerical optimization theory [14, 34] have been applied for accelerating neural network training. Moreover, commonly known heuristics approaches such as momentum [26, 49] or variable learning rate [40, 41] lead to a significant improvement. In this technical report we compare the performance of classical gradient descent methods and examine the effect of incorporating into them a variable learning rate and an adaptive nonmonotone strategy. We perform a large scale study on the behavior of the presented algorithms and identify their possible advantages. Additionally, we propose two modifications of two well-known second order algorithms aiming to overcome the limitations of the original methods.

keywords. artificial neural networks, optimization algorithms, adaptive nonmonotone strategy, backpropagation algorithms with adaptive learning rate, self-scaled quasi-Newton algorithms, scaled conjugate gradient algorithms.

Contents

1 Introduction

The area of artificial neural networks has been extensively studied and has been widely used in many applications of artificial intelligence. Due to their excellent capability of self-learning and self-adapting, they attracted much interest from the scientific community. The problem of training a neural network can be formulated as the minimization of an error function that depends on the connection weights of the network [49]. However this optimization problem has the disadvantage of being non-linear, non-convex, possessing multitudes of local minima and has broad flat regions adjoined with narrow step ones. To overcome these difficulties several optimization algorithms and modifications have been applied [19, 26, 30, 40, 41, 49] for this problem and their performance significantly varies with the problem to solve.

In the neural network literature a classical approach for training a neural network is the application of gradient-based algorithms such as the backpropagation which was introduced by Rumelhart et al. [49]. In particular, backpropagation algorithm minimizes the error function using the steepest descent direction. Although the error function is rapidly decreasing along the negative gradient direction, unfortunately backpropagation is usually very inefficient and unreliable [18] due to the morphology of the error surface. In addition, this is also because its performance is also depended on parameters which have to be heuristically specified by the user, because no theoretical basis for choosing them exists [30]. The values of these parameters are often crucial for the success of the algorithm, thus the designer is required to arbitrarily select parameters such as initial weights, network topology and a learning rate value. It has been established by many researchers [10, 19, 40, 41] that very small variations in these values can make the difference between good, average or bad performance and the proper choice of them constitutes one of the subjects of major concern in the literature [19]. To provide faster and more stable convergence several variations and modifications of backpropagation have been developed.

Various heuristic techniques have been proposed including the adaptation of a momentum term [26, 49] or a variable learning rate [26, 54]. More specifically, in [29, 41] have been proposed two popular techniques for dynamically evaluating the learning rate without any heuristic factor or any additional function and gradient evaluations. The first one was based on the Barzilai and Borwein algorithm [5] which adapts the learning rate containing second order information without evaluating the Hessian matrix; while the second one is based on estimates of the Lipschitz constant, exploiting local information from the error surface and the subsequent weights [29]. Numerical evidences [29, 40, 41] have shown that backpropagation with adaptive learning rate algorithms are robust and have good average performance for training neural networks.

Furthermore, another burst of research for improving the convergence properties of backpropagation is relying on the well established unconstrained optimization theory, for neural network training. Specifically, a variety of second-order optimization methods were suggested for improving the efficiency of the minimization error process. Methods that use second-order information such us the conjugate gradient [15, 24, 44] and quasi-Newton methods [25, 34] constitute a popular substitutional choice. Conjugate gradient methods are well suited for

large scale neural networks due to their simplicity, their convergence properties and their very low memory requirement. Instead of using the negative gradient direction, they use a linear combination of the previous search direction and the current gradient which produces generally faster and more stable convergence than backpropagation algorithm. In the literature there is a variety of conjugate gradient methods [8, 30] that have been intensively used for neural network training in several applications [11, 39, 51]. Quasi-Newton methods are usually regarded as the most sophisticated algorithms constituting an alternative to conjugate gradient algorithms for fast neural network training. They define the search direction building up an approximation of the Hessian matrix requiring additional curvature information. These methods are usually criticized because sometimes the approximations of the Hessian matrix during the training process may come close to singular or badly scaled and as a consequence they might produce inaccurate results or even stalling. Several attempts [3, 35, 36, 37, 57] have been proposed to adjust the size of the Hessian approximation by introducing scaling strategies. These strategies combined with nonmonotone line search made it possible to define global and superlinear convergent [57] and significantly improve the performance of the original methods for neural network training.

In this work, we briefly summarize classical gradient descent algorithms for neural network training. Our long term aim is to perform a large scale study on the behavior of the presented algorithms using different network architectures and identify their possible advantages and limitations. To be more precise, all algorithms have been evaluated not only on training classical feedforward neural networks but also and on recurrent neural networks which include feedback loops and are capable of processing temporal patterns. This architecture constitutes an elegant way of increasing the capacity of feed-forward neural networks to deal with more complex data sets and problems [39]. Although they can be trained in a way similar to the backpropagation networks [55, 56] such training requires a great deal of computation. In this contribution, we also propose two modifications of two well-known second order algorithms. Our first proposed algorithm combines the spectral conjugate gradient of Birgin and Martínez [8] with an adaptive nonmonotone strategy [29], exploiting the efficiency of the spectral conjugate gradient algorithm and the tuning strategy for the nonmonotone learning horizon. While the second one revises the adaptive nonmonotone self-scaling BFGS algorithm [38] using an alternative scaling factor introduced in [3]. These algorithms aim to overcome the limitations of the original methods and provide faster and more stable training.

The remainder of this paper is organized as follows. In Section 2 we introduce a brief overview of the optimization theory which form the foundation of the neural network training. In Section 3 we give a short description of the first and second order algorithms and their modifications for neural network training. Additionally we present two proposed variations of the original second order methods and explore their advantages. Experimental results are reported in Section 4 to evaluate the performance of presented algorithms in a variety of classical artificial intelligence benchmarks. Finally, Section 5 presents our concluding remarks and proposals for future work.

2 Background and Notations

To simplify the formulation of training a neural network throughout this article, we use the following notations based on its weights. Let us consider a multi-layer neural network with L layers where the l-th layer contains N_l neurons, for $l = 1, \ldots, L$. The network operation is based on the following equations, representing the input net_j^l and output o_j^l of the j-th neuron in the l-th layer

$$net_j^l = \sum_{i=1}^{N_{l-1}} w_{ij}^{l-1,l} o_i^{l-1}, \qquad o_j^l = f(net_j^l), \tag{1}$$

where $f(\cdot)$ is an activation function. The weights from the j-th neuron at the $l-1$-th layer to the j-th neuron at the l-th layer are denoted by $w_{ij}^{l-1,l}$. The above formulation defines the weight vector as a point in the N-dimensional Euclidean space $\mathbb{R}^N$, where N denotes the total number of connection weights, that is $w = (w_1, w_2, \ldots, w_n)^T$. For training a neural network, our aim is to find the optimal weights $w^* = (w_1^*, w_2^*, \ldots, w_n^*)^T$ which minimize the error function, namely,

$$w^* = \min_{w \in \mathbb{R}^N} E(w) \tag{2}$$

where E is defined as the sum of the square difference between the actual output value $o_{j,p}^l$ at the j-th output layer neuron of the output layer L for pattern p and the target output value $t_{j,p}$, that is

$$E(w) = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{N_l} \left(o_{j,p}^L - t_{j,p} \right)^2, \tag{3}$$

Actually finding such a minimizer is equivalent to finding an optimal set of connection weights such that the errors of the network output are minimized. In order to perform the minimization of the error function (3) we iteratively update the weights using the following iterative formula

$$w^{k+1} = w^k + \eta_k d_k \tag{4}$$

where k is the current iteration usually called *epoch*, $w^0 \in \mathbb{R}^N$ is a given initial point, $\eta_k \in \mathbb{R}$ is the learning rate and $d_k \in \mathbb{R}^N$ is the search direction.

2.1 Monotone Line Search

Traditional gradient-descent algorithms determine a search direction d_k using gradient information and then search along that direction for determining the learning rate η_k satisfying the *Wolfe conditions*:

$$\begin{aligned} E(w^k + \eta_k d_k) - E(w^k) &\leq \sigma_1 \eta_k \nabla E(w^k) d_k, & (5) \\ \langle \nabla E(w^k + \eta_k d_k), d_k \rangle &\geq \sigma_2 \langle \nabla E(w^k), d_k \rangle & (6) \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product and $0 < \sigma_1 < \sigma_2 < 1$. The first inequality (5) called *Armijo's condition* [4] guarantees that the error function is sufficiently reduced at each epoch. While, the second inequality (6) called *curvature condition*, prevents the learning rate from being too small. In general, the strategy based on the above conditions provide an efficient and effective way to ensure that the error function is globally reduced sufficiently.

2.2 Nonmonotone Line Search

Unfortunately, even when an algorithm has been proved to be globally convergent [13] using Wolfe's conditions (5)-(6), there is no guarantee that the method would be efficient in the sense that it may be trapped in a local minimum and never jump out to a better one under ill conditions [17], such as poorly initialized weights in the case of neural networks. To alleviate this situation Grippo et al. proposed a nonmonotone learning strategy [20] that exploits the accumulated information with regard to the M most recent values of the error function.

$$E(w^k + \eta_k d_k) - \max_{0 \le j \le M} E(w^{k-j}) \leq \sigma_1 \eta_k \nabla E(w^k) d_k, \tag{7}$$

$$\langle \nabla E(w^k + \eta_k d_k), d_k \rangle \geq \sigma_2 \langle \nabla E(w^k), d_k \rangle \tag{8}$$

where M is a positive integer, named *nonmonotone learning horizon*. It has been shown [22] that the *nonmonotone Wolfe's conditions* (7)-(8) allows for an increase in the function values, without affecting the global convergence properties. The philosophy behind nonmonotone strategies is that, many times, the first choice of a trial point by a minimization algorithm hides a lot of wisdom about the problem structure and that such knowledge can be destroyed by the decrease imposition. Many authors [20, 41, 42] suggested that the selection of the parameter M is crucial for the efficiency of the algorithm and it heavily depends on the problem. Motivated by Caychy's method [4, 10], in [40] has been proposed an elegant way for adapting the constant M throughout the local estimation of the Lipschitz constant, namely

$$M^k = \begin{cases} M_{k-1} + 1, & \text{if } \Lambda^k < \Lambda^{k-1} < \Lambda^{k-2}, \\ M_{k-1} - 1, & \text{if } \Lambda^k > \Lambda^{k-1} > \Lambda^{k-2}, \\ M_{k-1}, & \text{otherwise} \end{cases} \tag{9}$$

where Λ^k is the local estimation of the Lipschitz constant at the k-th iteration

$$\Lambda^k = \frac{\|\nabla E(w^k) - \nabla E(w^{k-1})\|}{\|w^k - w^{k-1}\|} \tag{10}$$

Based on their numerical experiments [29, 40] the authors concluded that the dynamic adaptation of M using (9) results better or equally good as the average performance taken for any predefined value for M.

Conclusively, we point out that both monotone and nonmonotone learning strategies can be incorporated in any training algorithm we will analyze in the following section. Moreover,

it is worth noticing that the conditions (6) and (8) are generally not needed because the use of backtracking strategy avoids very small learning rates [40]. Next, we will present a typical framework of a neural network training algorithm.

Algorithm 1 Algorithm for Training an Artificial Neural Network

Step 1: Initiate w^0, σ_1, σ_2, $k = 0$ and the error goal ϵ.

Step 2: **If** $((\nabla E(w^k) = 0)$ or $(E(w^k) < \epsilon))$ **then** stop.

Step 3: Compute descent direction d_k using backpropagation, quasi-Newton or conjugate gradient direction.

Step 4: Find a step length η_k using the following line search. For $0 < \sigma_1 \leq \sigma_2 < 1$ at each iteration, choose the step length η_k, using monotone (5) or nonmonotone (7) strategy.

Step 5: Update the weights

$$w^{k+1} = w^k + \eta_k d_k$$

Step 6: Set $k = k + 1$ goto Step 2.

3 Algorithms for Training Neural Networks

After the discovery of the backpropagation algorithm [49] several variants and modifications of this method were developed in order to provide faster convergence and are divided in two categories. The first category includes the heuristic techniques of which stand out the adaptation of a momentum term or a variable learning rate in the backpropagation framework; the second category includes the application of other numerical optimization methods, such as conjugate gradient and quasi-Newton methods.

3.1 Backpropagation Method and Its Variants

From the point of view of optimization, the backpropagation algorithm can be viewed essentially as a gradient-type method using the steepest descent direction [17].

$$d_k = -\nabla E(w^k) \tag{11}$$

Hence we can rely on the well established theory of gradient-related minimization algorithms for accelerating its convergence properties [19].

3.1.1 Momentum Backpropagation

An improved version of backpropagation was proposed in [26, 49] which consists of adding a *momentum term* in the steepest descent direction as follows:

$$w^{k+1} = w^k - (1 - m)\eta_k \nabla E(w^k) + m(w^k - w^{k-1}) \tag{12}$$

where m is the momentum constant. This extra term is generally interpreted to avoiding oscillations in the error surface and stacking in a swallow local minimum. The main drawback of this version is that an unfortunate choice of the momentum constant could significantly slow the convergence. For example, if m is set to a comparatively large value, gradient information from previous epochs is more influential than the current gradient information in updating the weights. One solution is to dynamically modify the learning rate, however, in practice, this approach sometimes proves ineffective and leads to instability or saturation.

3.1.2 Variable Learning Backpropagation

Another modification of backpropagation is the adaptation of a different *learning rate* at each epoch, exploiting local information from the error surface. Adaptive learning rate algorithms [4, 5, 10, 54] is a popular class of first order algorithms that try to overcome the inherent difficulty of choosing the proper learning rate for each problem at each epoch. They attempt to avoid oscillations and, at the same time, maximizing the length of the minimization step by controlling the magnitude of the changes in the weight states during the learning process. The key feature of the following two approaches are that they are simple and they require low storage and inexpensive computations.

The first approach was based on the Cauchy's method [4, 10] suggests that the value of the learning rate η_k is closely related with Lipschitz constant L, which depends on the morphology of the error surface. However, during the training process neither the value of the Lipschitz constant nor the morphology of error surface is known a priori. To alleviate this situation a variant of backpropagation method has been proposed [40] for training neural networks which employs the learning rate using a local estimation of the Lipschitz constant Λ_k (10) for gradient descent methods. Thus the variable learning rate at each epoch is defined by

$$\eta_k = \frac{1}{2\Lambda_k} = \frac{\|w^k - w^{k-1}\|}{2\|\nabla E(w^k) - \nabla E(w^{k-1})\|} \tag{13}$$

The selection of this choice constitutes in that whenever the error surface has steep regions Λ_k is large, and a small learning rate is selected. On the other hand when the error surface has flat regions, Λ_k is small, thus a large learning rate is selected to accelerate the convergence.

3.1.3 Barzilai and Borwein Backpropagation

An alternative approach was proposed in [5] where Barzilai and Borwein introduced a formula to dynamically evaluate the learning rate at each iteration. The motivation for this selection constitutes in proving a two-point approximation to the secant equation underlying quasi-Newton methods [43]. In particular, the learning rate is updated using the following formula:

$$\eta_k = \frac{\langle s_{k-1}, s_{k-1}\rangle}{\langle s_{k-1}, y_{k-1}\rangle} \tag{14}$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product in $\mathbb{R}^N$, $s_{k-1} = w^k - w^{k-1}$ and $y_{k-1} = \nabla E(w^k) - \nabla E(w^{k-1})$. Barzilai and Borwein [5] presented a convergence analysis in the two-dimensional

case quadratic case, establishing, for that particular case R-superlinear convergence. Later, Raydan [48] extended their work embedding the nonmonotone strategy of Grippo et al. [20] in the Barzilai and Borwein gradient method to prove global convergence. Furthermore, Raydan had also proposed an acceptance criterion for the Barzilai and Borwein step (14) to avoid having the learning rate very small or too large. In addition, another use of his criterion was to rule out the uphill directions and keep the sequence of learning rate $\{\eta_k\}_{\infty}^{k=0}$ uniformly bounded during the training process. Numerical experiments in [41] showed that the adaptation of the Barzilai and Borwein formula (14) in backpropagation algorithm is valuable in neural network training since their proposed method escapes local minima and flat regions, whereas the classical method is trapped.

It is worth noticing that in both mentioned approaches [40, 41] for adapting the learning rate the authors concluded based on their numerical results that the selection of a dynamic learning rate provides increased rate of convergence and guarantees the stability and the efficiency of the training process.

3.2 Conjugate Gradient Methods

Conjugate gradient methods [15, 23, 24, 44] are among the most commonly and efficient used methods for large scale optimization problems [17] due to their speed and simplicity. The main idea for determining the search direction in equation (4) is the linear combination of the negative gradient vector at the current iteration with the previous search direction. The way to determine the search direction can be expressed as follows:

$$\begin{array}{rcl} d_0 & = & -\nabla E(w^0) \\ d_k & = & -\nabla E(w^k) + \beta_{k-1} d_{k-1} \end{array} \tag{15}$$

Conjugate gradient methods differ in their way of defining the multiplier β_{k-1}. The most famous approaches were proposed by Fletcher-Reeves (FR) [15] and Polak-Ribière (PR) [44].

$$\beta_k^{FR} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \quad \text{and} \quad \beta_k^{PR} = \frac{(g_k - g_{k-1})^T g_k}{g_{k-1}^T g_{k-1}} \tag{16}$$

where $g_k = \nabla E(w)$. The conjugate gradient methods using Fletcher-Reeves update were shown to be globally convergent [1]. However, the corresponding methods using Polak-Ribière update are generally more efficient even without satisfying the global convergence property. A possible explanation for this has been presented by Powell [45]. In particular, based on his practical experience, he suggested that the Fletcher-Reeves method shows a propensity to take short steps remote from the solution which leads to slow convergence and sometimes even stalling. On the other hand, the Polak-Ribière method may not suffer from this disadvantage.

Moreover an important issue of conjugate gradient algorithms is that when the search direction (15) fails to be a descent direction, we restart the algorithm using the negative gradient direction to guarantee convergence. A more sophisticated and popular restarting

criterion has been introduced by Powell [45] which consists of testing if there is very little orthogonality left between the current gradient and the previous one, namely

$$\|g_{k+1}g_k\| \geq 0.2\|g_{k+1}\|^2 \tag{17}$$

Later, Shanno and Phua [50] and Birgin and Martínez [8] have also suggested algorithms based on the use of restarts. Nevertheless, there is also a worry with restart algorithms that restarts may bee triggered too often, therefore degrading the overall efficiency of the method.

In general, conjugate gradient methods constitute a very popular choice for training large neural networks and their convergence properties and performance has been investigated by many researchers [11, 27, 30, 39, 53].

3.2.1 Adaptive Nonmonotone Spectral Conjugate Gradient

In a more recent work, Birgin and Martínez [8] introduced a spectral conjugate gradient method embedding the spectral gradient in the conjugate gradient framework in the following way:

$$\begin{array}{rcl} d_0 & = & -\nabla E(w^0) \\ d_k & = & -\theta_k \nabla E(w^k) + \beta_{k-1} d_{k-1}, \end{array} \tag{18}$$

where θ_k is the inverse of the Rayleigh quotient defined by

$$\theta_k = \frac{s_{k-1}^T s_{k-1}}{y_{k-1}^T s_{k-1}} \tag{19}$$

which implies that this choice contains second order information without estimating the Hessian matrix. Using a geometric interpretation for the quadratic function minimization, Birgin and Martínez [8] suggested the following expressions for defining β_k in equation (18):

$$\beta_k^{SFR} = \frac{\theta_k g_k^T g_k}{\theta_{k-1} g_{k-1}^T g_{k-1}} \quad \text{and} \quad \beta_k^{SPR} = \frac{\theta_k y_k g_{k+1}}{\theta_{k-1} g_k^T g_k} \tag{20}$$

If $\theta_k = \theta_{k-1} = 1$ these are the classic Fletcher-Reeves [15] and Polak-Ribière formulas [44] respectively. The learning rate is automatically adapted at each epoch according to Shanno's and Phua's technique [50] using the conjugate gradient values and the learning rate of the previous epoch.

$$\eta_k = \begin{cases} \frac{1}{\|g_0\|_2} & \text{if } k = 0, \\ \frac{\eta_{k-1}\|d_{k-1}\|}{\|d_k\|} & \text{otherwise} \end{cases} \tag{21}$$

where d_k is the conjugate gradient direction, d_{k-1} is the previous one, η_{k-1} is the previous learning rate and g_0 is the initial gradient. The restarting procedure considered by Birgin and Martínez [8] consists of testing if the angle between the current direction d_{k+1} and the gradient g_k is not accurate enough. Therefore, when:

$$d_{k+1}g_{k+1} \leq -10^{-3}\|d_{k+1}\|_2\|g_{k+1}\|_2 \tag{22}$$

then the algorithm is restarted using the spectral gradient direction $d_k = -\theta_k \nabla E(w^k)$. An obvious advantage of the spectral conjugate gradient algorithms is that since the steepest descent direction has been proven to be inefficient [18], it is not wise to restart the algorithm with the negative gradient direction. Namely, in the case the algorithm is restarted such a choice will not necessarily provide an effective result in the efficiency of the method. Instead the spectral conjugate gradient algorithm restarts using the more effective [48] spectral gradient direction exploiting second order information without estimating the Hessian matrix. Since their appearance, spectral conjugate gradient have been intensively used in a large area of applications [6, 7, 58] and have been the object of several modifications due to their efficiency and stability properties.

Motivated by the efficiency and the advantages of spectral conjugate gradient algorithms, we propose an *adaptive spectral conjugate gradient algorithm* for training neural networks which is based on the combinations of the frameworks presented in [8, 39, 51]. In particular, we supply the algorithm of Birgin and Martínez [8] with the tuning strategy for the nonmonotone learning horizon (9) introduced in [40]. Our proposed algorithm exploits the ideas of scaling the gradient by means of the Raydan's spectral parameter [48] and the adaptive nonmonotone learning horizon. We have also used the restarting criterion proposed in [8] because it is associated with the spectral gradient choice for restarts. A high level description of our algorithms is presented as follows:

Algorithm 2 Adaptive Nonmonotone Spectral Conjugate Gradient

Step 1: Initiate w^0, $\sigma_1 \in (0,1)$, $k = 0$, $d_0 = -\nabla E(w^0)$ and the error goal ϵ.

Step 2: **If** $((\nabla E(w^k) = 0)$ or $(E(w^k) < \epsilon))$ **then** stop;

Step 3: Adapt the nonmonotone learning horizon M using (9).

Step 4: Calculate the learning rate η_k using (21).

Step 5: Find a step length η_k using the following line search. For $\sigma_1, \sigma_2 \in (0,1)$ at each iteration, choose the step length satisfying

$$\begin{aligned} E(w^k + \eta_k d_k) - \max_{0 \le j \le M} E(w^{k-j}) &\le \sigma_1 \eta_k \nabla E(w^k) d_k \\ \langle \nabla E(w^k + \eta_k d_k), d_k \rangle &\ge \sigma_2 \langle \nabla E(w^k), d_k \rangle \end{aligned}$$

Step 6: Update the weights

$$w^{k+1} = w^k + \eta_k d_k$$

Step 7: (i) Compute the spectral quotient θ_{k+1} (19).
(ii) Apply Raydan's acceptance criterion [48].

Step 8: (i) Compute the descent direction using (20):

$$d_{k+1} = -\theta_{k+1} \nabla E(w^{k+1}) + \beta_{k+1} d_k$$

(ii) Apply the restart criterion of Birgin and Martínez (22)

Step 9: Set $k = k + 1$. and goto Step 3.

3.3 Quasi-Newton Methods

Quasi-Newton methods [25, 34] constitute the most efficient and powerful second-order methods for neural network training with respect to the training precision. Quasi-Newton search directions provide an attractive alternative to Newton's direction requiring only to satisfy the secant equation $B_{k+1}s_k = y_k$ where $s_k = w^{k+1} - w^k$ and $y_k = \nabla E(w^{k+1}) - \nabla E(w^k)$. The approximation of the of Hessian matrix is estimated directly by a symmetric positive definite matrix B_k which is updated iteratively maintaining its positive definiteness and preservation of symmetry using a relevant line search [34]. Thus, the search direction (4) takes the following form:

$$d_k = -B_k^{-1} g_k \tag{23}$$

where g_k is the gradient of the error function and the approximation of the Hessian matrix, B_k is calculated using

$$B_{k+1} = \tau_k \left[B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \phi_k u_k u_k^T \right] + \frac{y_k y_k^T}{y_k^T s_k} \tag{24}$$

where $s_k = w^{k+1} - w^k$, $y_k = \nabla E(w^{k+1}) - \nabla E(w^k)$, $u_k = (s_k^T B_k s_k)^{1/2} \left(\frac{y_k}{s_k^T y_k} - \frac{B_k s_k}{s_k^T B_k s_k} \right)$ and τ_k and ϕ_k are the scaling and the updating parameters, respectively. If $\tau_k = 1$ then class (24) reduces to the classical unscaled Broyden family of Hessian approximation updates. In this case, although $\phi_k \in [0, 1)$, the iteration scheme (4) converges globally and q-superlinearly for convex functions under appropriate conditions [9, 34]. However only the update with $\phi_k = 0$ which is the popular the Broyden-Fletcher-Goldfarb-Shanno update (BFGS)

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{25}$$

has been shown to be efficient and useful in practice. Additionally, it has been suggested by many researchers [3, 9, 46, 59] that the performance worsens as ϕ_k increases from 0 to 1.

Nevertheless, even if this method is globally convergent, it suffers from the large eigenvalues in the Hessian approximation matrix of the objective function (3). This means that some approximations of the Hessian matrix during the training process may come close to singular or badly scaled and as a consequence they might produce inaccurate results or even stalling. To overcome this difficulty a new technique has been first proposed by Oren [36] and Oren and Luenberger [37] motivated from the analysis of quadratic objective functions [33], by tuning the scalar τ_k using as scaling factor:

$$\tau_k = \frac{y_k^T s_k}{s_k^T B_k s_k} \tag{26}$$

Subsequently, there have been proposed several scaling approaches [3, 35, 57] for defining τ_k in order to dynamically scale the Hessian approximations of error function when they have large eigenvalues. However, the first numerical experiments [35] where very disappointing. In particular, Nocedal and Yuan [35] showed that the best self-scaling BFGS algorithm of

Oren and Luenberger [37] performs badly compared to the classical BFGS algorithm when applied with inexact line search to a simple quadratic function of two variables.

Later, Al-Baali [2] determined conditions on τ_k and ϕ_k that ensure the global and superlinear convergence for scaling algorithms with inexact line search under the additional condition that

$$\tau_k \leq 1. \tag{27}$$

Condition (27) is motivated by the fact that the eigenvalues of the bracketed matrix in (24) can be reduced if $\tau_k < 1$ and hence smaller eigenvalues are introduced in B_{k+1} if the eigenvalues of B_k are large. Numerical evidences [2, 3] presented that some members of the Broyden family (24), including BFGS, was improved substantially and concluded that these scaled methods are superior to the original quasi-Newton methods. Recently, Yin and Du [57] extended the work of Oren and Luenberger [37] and based on the self-correcting property of the Hessian approximation, they established global convergence for nonconvex functions with nonmonotone line search. Their method has been used for training recurrent neural networks combined with an adaptive nonmonotone learning in [38] proving some promising results.

3.3.1 Adaptive Self-Scaling Nonmonotone BFGS (Revised)

Motivated by the works presented in [3, 38, 57] we revise the *adaptive self-scaling nonmonotone BFGS algorithm* [38] incorporating an alternative scaling factor [3]

$$\tau_k = \min\left\{1, \frac{y_k^T s_k}{s_k^T B_k s_k}\right\} \tag{28}$$

As suggested by Al-Baali [2, 3], the scaling factor (28) has been shown that it converges to 1 if the BFGS method is used with a learning rate of 1 [12]. Because in the limit this self-scaled BFGS algorithm reduces to the unscaled BFGS algorithm if superlinear convergence is obtained, the latter property becomes less important. Therefore, we prefer using the scaling factor (28), which does not destroy the properties of scaling whenever $\tau_k < 1$. Namely, only when the approximation of the Hessian matrix B_k can be indefinite or singular the scaling factor scale B_k in order to keep its eigenvalues within a suitable range, in other case the scaling factor is equal to 1. Additionally, Yin and Du [57] suggested that the substitution of scaling factor (26) by (28) doesn't destroy the global convergence property.

There has been no theoretical proof for which scaling factor (26)-(28) is superior, nevertheless based on our numerical experiments in Section 4, we believe that the scaling factor (28) provides better, faster and more stable convergence. Our approach seems to overcome the limitations caused by monotone line search and exploits the mentioned benefit of the usage of scaling factor (28). To this end we summarize the new training algorithm.

Algorithm 3 Adaptive Self-Scaling Nonmonotone BFGS (Revised)

Step 1: Initiate w^0, $\sigma_1 \in (0,1)$, $k=0$ and the error goal ϵ.

Step 2: **If** $((\nabla E(w^k)=0)$ or $(E(w^k)<\epsilon))$ **then** stop;

Step 3: Update the Hessian approximation matrices B_k using the self-scaling BFGS updating:

$$B_{k+1} = \tau_k \left[B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \right] + \frac{y_k y_k^T}{y_k^T s_k}, \quad \text{where } \tau_k = \min\left\{1, \frac{y_k^T s_k}{s_k^T B_k s_k}\right\}$$

Step 4: Compute the descent direction:

$$d_k = -B_k^{-1} g_k$$

Step 5: Adapt the nonmonotone learning horizon M using (9).

Step 6: Find a step length η_k using the following line search. For $\sigma_1, \sigma_2 \in (0,1)$ at each iteration, choose the step length satisfying

$$\begin{aligned} E(w^k + \eta_k d_k) - \max_{0\le j\le M} E(w^{k-j}) &\le \sigma_1 \eta_k \nabla E(w^k) d_k \\ \langle \nabla E(w^k + \eta_k d_k), d_k \rangle &\ge \sigma_2 \langle \nabla E(w^k), d_k \rangle \end{aligned}$$

Step 7: Update the weights

$$w^{k+1} = w^k + \eta_k d_k$$

Step 8: Set $k = k+1$ goto Step 2.

4 Experimental Results

In the following section we will present experimental results in order to study and evaluate the performance of the most commonly used first and second order described algorithms in three classical artificial intelligence problems. In particular, we investigate the performance of the first order methods of backpropagation (BP) [49], Barzilai and Borwein backpropagation (BBP) [41] using the modification of Raydan [48] and variable learning backpropagation (BPVS) [40] and compare them with their respectively nonmonotone versions, namely the momentum backpropagation (MBP) [26, 49], the nonmonotone Barzilai and Borwein backpropagation (NMBBP) [41] and the nonmonotone variable learning backpropagation (NMBPVS) [40]. Additionally, we compare the performance of the BFGS [17] and the adaptive self-scaling nonmonotone BFGS (ASCNMBFGS) [38] with the revised adaptive self-scaling nonmonotone BFGS (ASCNMBFGS$^+$). And finally we evaluate the performance of the conjugate gradient (CG) [17] and the adaptive nonmonotone conjugate gradient (ANMCG) [39] with the adaptive nonmonotone scaled conjugate gradient (ANMSCG). Note that all conjugate gradient methods have been implemented using the Polak-Ribière update. For

all algorithms the heuristic parameters were set as $\eta_0 = 0.1$, $\sigma_1 = 10^{-4}$, $\sigma_2 = 0.5$, $m = 0.9$ and $3 \leq M^k < 15$ for all experiments which provide us the best overall results. The implementation has been carried out on a processor Pentium-IV computer (3.2MHz, 1Gbyte RAM) using Matlab version 7 and the Matlab Neural Network Toolbox.

The performance of each algorithm was tested in two different neural networks architectures, i.e Feedforward Neural Networks (FFNN) and Elman's Recurrent Neural Network (ERNN). All networks have received the same sequence of input patterns and the initial weights were initiated using the Nguyen-Widrow method [32]. The selection of this method results in distributing the initial weights at the hidden layer in such a way that it is more likely that each input pattern will cause a hidden neuron to learn efficiently, accelerating convergence and preventing premature saturation.

For each test problem, we present a table summarizing the performance of the algorithms for simulations that reached solution within a predetermined limit of epochs. The parameters used in all tables are as follows: *Min* the minimum number of epochs, *Max* the maximum number of epochs, *Mean*, the mean value of epochs, *s.d.* the standard deviation and *Succ.* the simulation succeeded out of 100 trials within predetermined error limit. If an algorithm fails to converge within the epoch limit, it is considered that it fails to train the neural network but its results are not included in the experimental analysis of the algorithm.

4.1 Parity-N Problem

The n-bit parity problem can be viewed as a generalization of the well-known XOR (eXclusive-OR) problem. Namely, the task is to train a neural network to produce the sum mod 2. It is usually used for evaluating algorithm performances and has been regarded as a famous challenging problem, that is well known to be a hard problem of many local minima. Here we consider two instances of the problem: parity-2 and parity-5. All networks were based on hidden neurons of hyperbolic tangent activations and on a linear output neuron. For parity-2 we have used neural networks with 2 hidden nodes while for parity-5 with 7 hidden nodes. The stopping criterion is set to $E \leq 10^{-2}$ within the limit of 1000 and 2000 epochs for each instance of the problem respectively.

4.1.1 Parity-2

Table 1 summarizes the average performance of the presented algorithms for the problem parity-2. In general quasi-Newton algorithms exhibit the best performance since they report the least average number of epochs to converge and they demonstrate the highest success rate in case of training a feedforward neural network. Among them, ASCNMBFGS$^+$ significantly outperforms the other two quasi-Newton algorithms in terms of training performance. As regards conjugate gradient algorithms, ANMSCG exhibits the highest possibility of successful training and demonstrates the least average number of epochs in contrast to the other conjugate gradient algorithms. Furthermore, it is worth noticing that ANMSCG exhibits the highest possibility (69%) of successful training a recurrent neural network, outperforming

	FFNN					ERNN				
Algorithms	Min	Mean	Max	s.d.	Succ.	Min	Mean	Max	s.d.	Succ.
BP	9	138.1	794	136.7	43%	31	223.5	988	173.2	42%
MBP	19	172.7	777	155.4	44%	48	246.9	806	187.1	44%
BBP	2	15.9	96	13.9	54%	4	65.7	864	123.7	54%
NMBBP	2	14.5	128	17.2	56%	4	53.1	943	109.8	56%
BPVS	4	22.8	255	35.6	53%	4	98.8	900	159.6	52%
NMBPVS	4	22.6	244	34.1	55%	6	93	841	146.4	53%
BFGS	2	13.7	407	54.6	54%	3	33.7	419	75.2	49%
ASCNMBFGS	2	9.2	40	5.5	56%	6	37.6	173	46	56%
ASCNMBFGS$^+$	2	8.2	79	10.1	61%	4	32.2	211	49.3	63%
CG	5	27.7	168	22.8	54%	13	95.6	487	122.2	58%
ANMCG	5	28.1	197	26.4	54%	14	105.1	618	147.3	59%
ANMSCG	3	13.6	123	16.3	58%	4	69.9	467	96.1	69%

Table 1: Comparative results of average performance for the Parity-2.

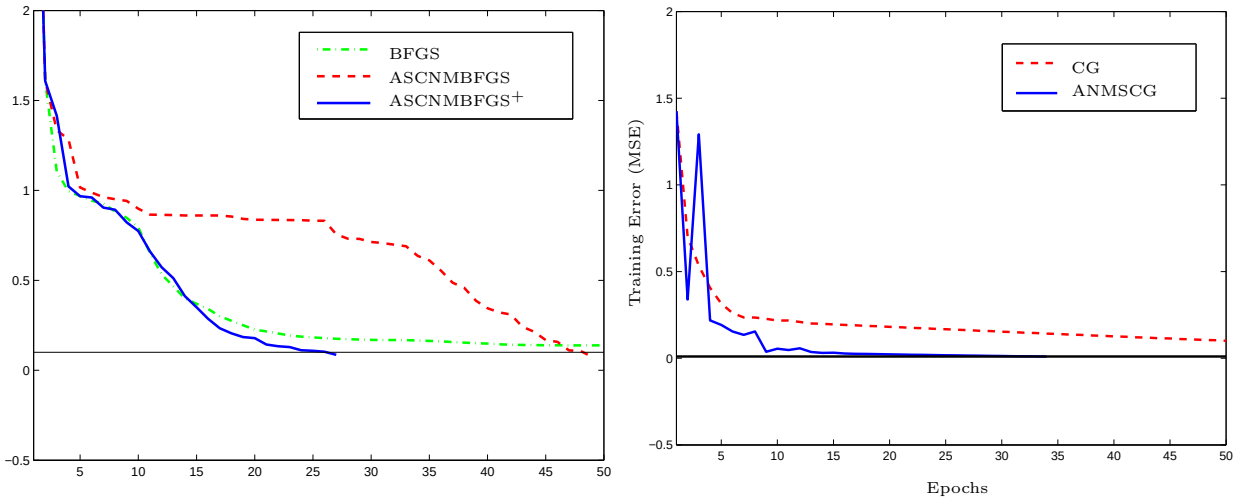

Figure 1: Performance comparison for training a recurrent neural network using quasi-Newton (left) and conjugate gradient (right) algorithms for problem parity-2.

even quasi-Newton algorithms. A performance comparison of the quasi-Newton and conjugate gradient algorithms is presented in Figure 1, demonstrating the superiority of the two proposed algorithms. As regards first order algorithms, Barzilai and Borwein backpropagation algorithms require the least amount of computation measurement in the average number of epochs and moreover their results are similar with those of the second order algorithms. It seems that the Barzilai and Borwein step (14) significantly improves and has a greater effect in the efficiency of the backpropagation algorithm in contrast to its all other accelerating modifications.

4.1.2 Parity-5

	FFNN					ERNN				
Algorithms	Min	Mean	Max	s.d.	Succ.	Min	Mean	Max	s.d.	Succ.
BP	346	940.7	1903	393.8	68%	334	769.1	1901	352.6	71%
MBP	389	978.3	1999	377.9	60%	293	780.3	1975	337	77%
BBP	39	218.2	1036	230.7	93%	69	239.1	1246	223.5	94%
NMBBP	32	230.5	1954	316.3	95%	51	251.4	1961	379.2	96%
BPVS	51	261.2	1552	251	92%	56	271	1933	297.7	93%
NMBPVS	60	289.7	1897	333.5	94%	63	280.7	1874	326.2	94%
BFGS	18	75.2	724	109.6	84%	20	52.0	244	37.7	82%
ASCNMBFGS	21	123.3	608	137.1	87%	31	83.4	310	65.6	84%
ASCNMBFGS$^+$	19	74.0	337	68.5	86%	21	48.8	222	38.7	82%
CG	77	268.5	1593	274.8	96%	61	250.2	1221	284.3	96%
ANMCG	80	237.4	724	167.3	98%	62	221.7	1232	228.4	97%
ANMSCG	31	140.5	847	153.5	100%	33	97.6	419	76.6	100%

Table 2: Comparative results of average performance for the Parity-5.

In Table 2 are presented the simulation results and the performance comparison of the presented algorithms for the problem parity-5. Regarding the quasi-Newton and the conjugate gradient algorithms ASCNMBFGS$^+$ and ANMSCG exhibit the best performance, respectively. These algorithms demonstrated the highest success rates, as well as the least average number of epochs to converge in contrast to the classical algorithms. Furthermore, it is worth mentioning that the adaptation of a variable learning rate in backpropagation algorithm provides a significant improvement in the efficiency of the backpropagation algorithm (Figure 2); while the use of a momentum term appears to be the worse of the adaptive techniques investigated in terms of training performance. In particular the performances of the backpropagation algorithms with adaptive learning rate exhibit very good probability of successful training (92%-96%). Moreover, their success rate are similar with the corresponding success rate of the second order algorithms. The rigorous analysis from the simulations results demonstrated that the nonmonotone strategy improves the efficiency and the effectiveness of the all algorithms. In addition, in some cases, nonmonotone methods exhibit equal

fast or even faster convergence than the corresponding monotone methods. Characteristic examples of the effect of the nonmonotone strategy on quasi-Newton and backpropagation algorithms are presented in Figure 2.

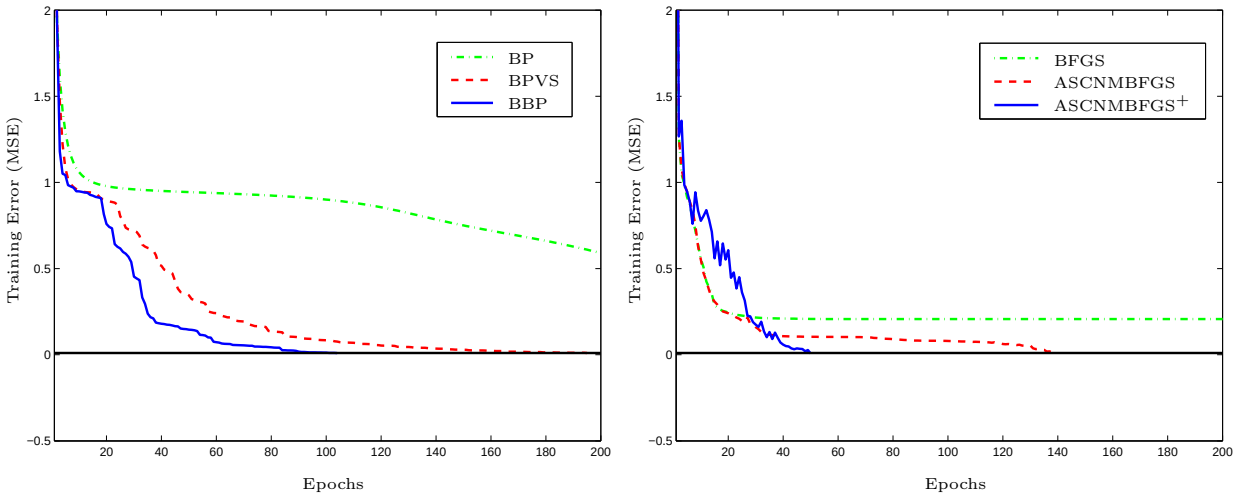

Figure 2: Performance comparison for training a recurrent neural using backpropagation algorithms with adaptive learning rate (left) and quasi-Newton algorithms (right) for problem parity-5.

4.2 Alphabetic Font Learning Problem

The problem of letter recognition is one of the oldest problems for evaluating the performance of a neural network training algorithm. Each letter has been defined in terms of binary values on a grid of size 5×7. Each network is based on 30 hidden neurons of logistic activations and on 26 linear output neurons. Training is considered successful when $E \leq 10^{-1}$ within the predetermined limit of 2000 epochs. However, due to the large size of the network quasi-Newton algorithms couldn't be applied for this specified topology.

Table 3 presents the performance comparison of the presented algorithms for the alphabetic font problem. All algorithms exhibit excellent probability (100%) of successful training for both network architectures. Thus, computational cost is probably the most appropriate indicator for measuring the efficiency of the algorithms. ANMSCG significantly outperforms all algorithms in terms of the average number of epochs. In particular, it requires approximately 7 times less average number of epochs in contrast to the classical conjugate gradient algorithm which implies an improvement of order of 84%. Thus ANMSCG provides much faster and more stable training, as it is also demonstrated in Figure 3. Regarding first order algorithms, Barzilai and Borwein backpropagation algorithms demonstrate 34% to 97% less average number of epochs in contrast to all other first order algorithms.

	FFNN					ERNN				
Algorithms	Min	Mean	Max	s.d.	Succ.	Min	Mean	Max	s.d.	Succ.
BP	518	651.8	776	52.2	100%	292	438.6	572	56.8	100%
MBP	531	643.3	746	48.2	100%	296	434.2	558	54.2	100%
BBP	12	15.1	20	2.1	100%	8	13.0	20	2.1	100%
NMBBP	11	13.7	18	1.3	100%	8	11.8	16	1.7	100%
BPVS	16	23.7	31	2.8	100%	12	18.9	25	2.5	100%
NMBPVS	16	23.7	33	3.0	100%	12	18.9	25	2.4	100%
BFGS	*	*	*	*	*	*	*	*	*	*
ASCNMBFGS	*	*	*	*	*	*	*	*	*	*
ASCNMBFGS$^+$	*	*	*	*	*	*	*	*	*	*
CG	71	83.9	97	6.0	100%	52	67.2	84	6.5	100%
ANMCG	73	83.0	96	4.9	100%	49	62.7	76	5.6	100%
ANMSCG	10	13.1	17	1.7	100%	7	11.4	15	1.4	100%

Table 3: Comparative results of average performance for the alphabetic font problem.

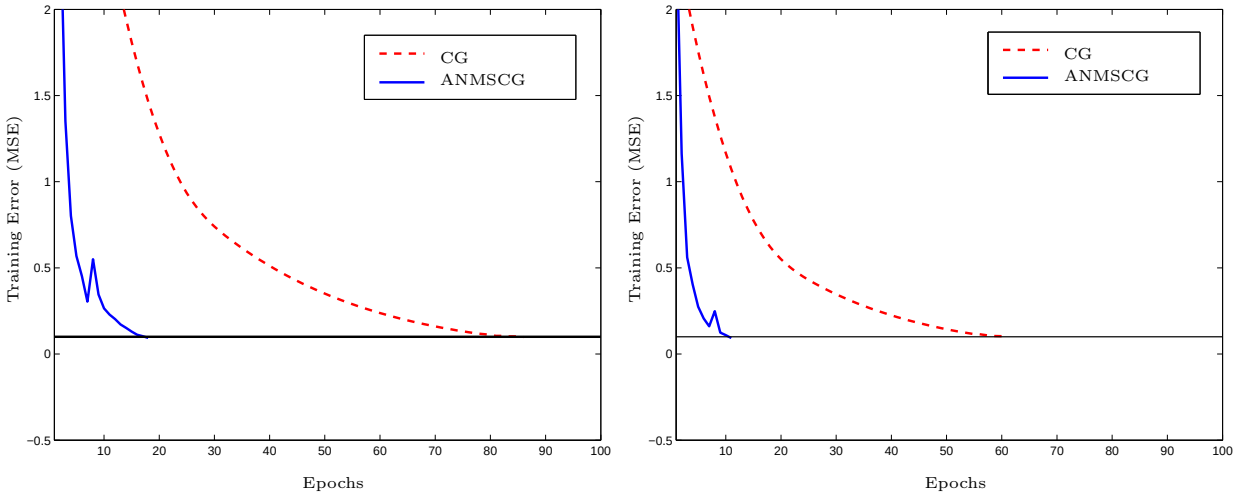

Figure 3: Performance comparison of CG and ASCNMSC for training a feedforward (left) and a recurrent neural network (right) for the alphabetic font recognition problem.

4.3 Monk's problems

The MONK's problem [52] is a collection of three binary classification problems relying on the artificial robot domain, in which robots are described by six different attributes. These benchmarks are made of a numeric base of examples and of a set of symbolic rules. Monk's problem is presented with three variants:

(1) Monk-1 consists of 124 patterns which were selected randomly from the data set for training, while the remaining 308 were used for the generalization testing.

(2) Monk-2 comprise by 169 randomly selected examples from the data set for training, while the rest were used for testing.

(3) Monk-3 problem consists of 122 patterns for training and the remaining 310 patterns were used for testing.

In our simulations, we have used the same number of neurons in the hidden layer as those found in [52].

4.3.1 Monk-1 Problem

	FFNN					ERNN				
Algorithms	Min	Mean	Max	s.d.	Succ.	Min	Mean	Max	s.d.	Succ.
BP	0	-	0	-	0%	0	-	0	-	0%
MBP	0	-	0	-	0%	0	-	0	-	0%
BBP	43	142.7	367	67.0	100%	37	146.7	584	97.6	98%
NMBBP	27	90.3	206	40.0	100%	25	91.1	414	62.2	100%
BPVS	80	389.8	933	195.0	98%	90	381.2	975	219.3	95%
NMBPVS	70	372.8	845	182.5	99%	70	379.0	898	214.9	96%
BFGS	19	60.2	897	88.9	99%	22	63	618	74.5	99%
ASCNMBFGS	13	31.4	83	10.1	100%	15	32.3	69	10.9	100%
ASCNMBFGS$^+$	11	20.7	29	3.6	100%	11	20.7	37	5.2	100%
CG	125	292.4	630	113.7	100%	70	266.5	681	127.7	100%
ANMCG	126	248.8	500	89.4	82%	71	237.1	611	106.0	86%
ANMSCG	26	71.1	166	24.1	100%	20	69.1	279	42.0	100%

Table 4: Comparative results of average performance for the Monk-1 problem.

Table 4 summarizes the simulation results for the Monk-1 problem. In both network architectures backpropagation and momentum backpropagation failed to converge within the pretermitted error limit. Quasi-Newton algorithms exhibit the best performance since they demonstrate excellent success rate (100%) and they require the least average number of epochs. It worth noting that the ASCNMBFGS$^+$ outperforms ASCNMBFGS, since the adaptation of scaling factor (28) instead of (26), implies a reduction of order 35% in the

average number of epochs for both network architectures. In general, conjugate gradient algorithms exhibit the worst performance among the second order algorithms. However, ANMSCG significantly outperforms the rest of the conjugate gradient algorithms since it demonstrates approximately 70.8% to 76.5% less average number of epochs. As regards the variants of backpropagation, we point out that the adaptation of a variable learning rate has a significant improvement in the efficiency of backpropagation especially the adaptation of the Barzilai and Borwein step (14). Additionally, it is worth noticing that the use of nonmonotone strategy has a significant effect on all algorithms, thus in Figure (4) we compare the performances of BBP and NMBBP and the performances of BFGS and ASCNMBFGS$^+$ with respect to the number of epochs.

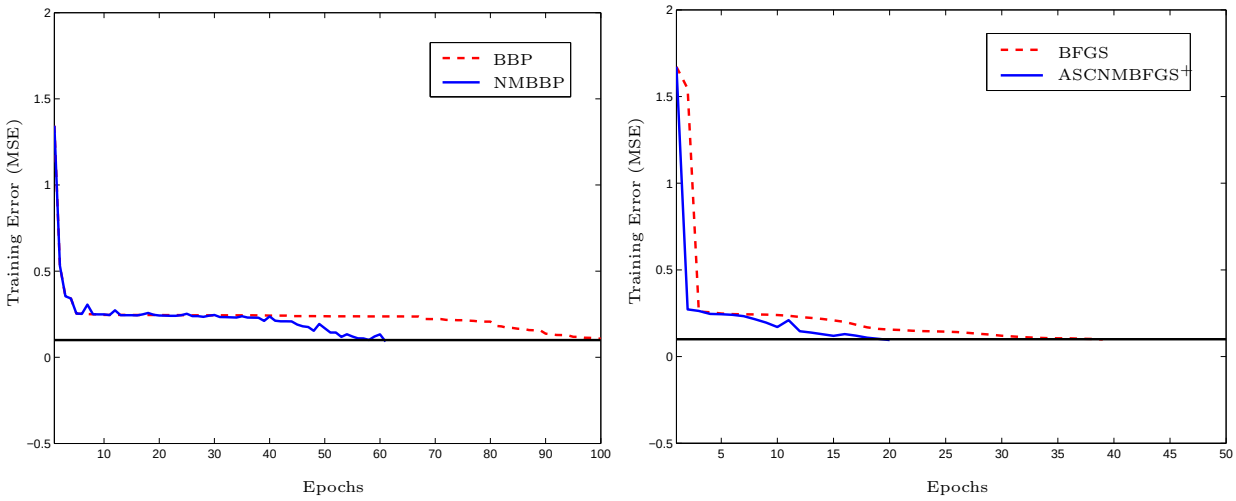

Figure 4: Performance comparison of BBP and NMBBP (left) and BFGS and ASCNMBFGS$^+$ (right) for training a recurrent neural for problem Monk-1.

4.3.2 Monk-2 Problem

In Table 5 are presented the simulation results for the Monk-2 problem. As regards the training success ANMSCG demonstrates the highest possibility of successful training for both network architectures. It is worth noticing that only the self-scaled BFGS algorithms required less average number of epochs but they also report less training success. Regarding quasi-Newton algorithms, ASCNMBFGS$^+$ demonstrated the best performance in terms of training success and average number of epochs. As regards the first order algorithms, the results have shown that the application of a variable learning rate significantly improves the performance of backpropagation algorithm. In particular the probability of successful training of the backpropagation algorithms with adaptive learning rate is 61% to 92%; while backpropagation and momentum backpropagation failed to converge in all simulations. For this very difficult problem nonmonotone strategy appears to have a significant effect on the performance of almost all algorithms. For example ANMSCG exhibits 60.8% to 76.5% less

	FFNN					ERNN				
Algorithms	Min	Mean	Max	s.d.	Succ.	Min	Mean	Max	s.d.	Succ.
BP	0	-	0	-	0%	0	-	0	-	0%
MBP	0	-	0	-	0%	0	-	0	-	0%
BBP	68	272.7	851	171.2	84%	42	245.1	996	196.7	85%
NMBBP	39	179.6	986	149.6	91%	46	156.3	642	131.7	92%
BPVS	174	528.2	999	207.1	63%	118	518.5	981	195.8	61%
NMBPVS	160	509.7	950	218.0	64%	131	495.0	993	206.3	64%
BFGS	32	113.5	972	172.3	77%	23	107.9	665	146.2	70%
ASCNMBFGS	23	46.2	358	38.9	78%	20	44.0	112	18.9	79%
ASCNMBFGS$^+$	16	30.4	121	14.5	80%	11	30.1	186	21.0	79%
CG	110	477.7	960	221.6	89%	186	498.4	923	211.4	88%
ANMCG	123	323.5	645	132.5	89%	165	372.7	688	151.4	88%
ANMSCG	36	127.2	321	76.1	97%	29	117.0	320	64.5	94%

Table 5: Comparative results of average performance for the Monk-2 problem.

average number of epochs than CG while the probability of successful training improved by 6% to 8%. Similar is the improvement for the Barzilai and Borwein backpropagation where the probability of successful training is improved by 7%. It is noticeable that some algorithms exhibit better average performance for training a recurrent neural neural which is a much more difficult network to be trained.

4.3.3 Monk-3 Problem

	FFNN					ERNN				
Algorithms	Min	Mean	Max	s.d.	Succ.	Min	Mean	Max	s.d.	Succ.
BP	191	707.8	974	204.4	38%	276	736.7	983	171.5	55%
MBP	149	710.4	998	202.7	39%	237	720.0	987	179.5	49%
BBP	14	46.3	94	17.7	100%	13	39.7	88	16.1	100%
NMBBP	10	31.1	59	10.6	100%	12	25.8	64	8.3	100%
BPVS	31	115.4	278	52.5	100%	30	100.0	255	42.3	100%
NMBPVS	31	107.7	250	47.7	100%	30	100.1	286	46.5	100%
BFGS	6	16.2	28	3.9	100%	8	15.9	34	4.1	100%
ASCNMBFGS	8	15.0	31	4.0	100%	6	13.8	34	3.9	100%
ASCNMBFGS$^+$	6	10.9	19	2.5	100%	5	10.2	17	2.0	100%
CG	45	99.6	232	32.1	100%	30	88.6	154	26	100%
ANMCG	35	99.3	187	31.4	100%	33	88.2	150	25	100%
ANMSCG	9	27.0	55	9.0	100%	9	24.4	44	7.6	100%

Table 6: Comparative results of average performance for the Monk-3 problem.

The results for the easiest variant of Monk's problems in Table 6 present that all algorithms except BP and MBP exhibit excellent (100%) probability of successful training for both network architectures. Once again quasi-Newton algorithms exploit their advantage of using additional curvature information exhibiting the least average number of epochs to converge. It is worth noticing that ASCNMBFGS$^+$ and ANMSCG outperform the classical quasi-Newton and conjugate gradient algorithms respectively in terms of average number of epochs (Figure 5). As regards first order algorithms, Barzilai and Borwein backpropagation algorithms demonstrated the least average number of epochs.

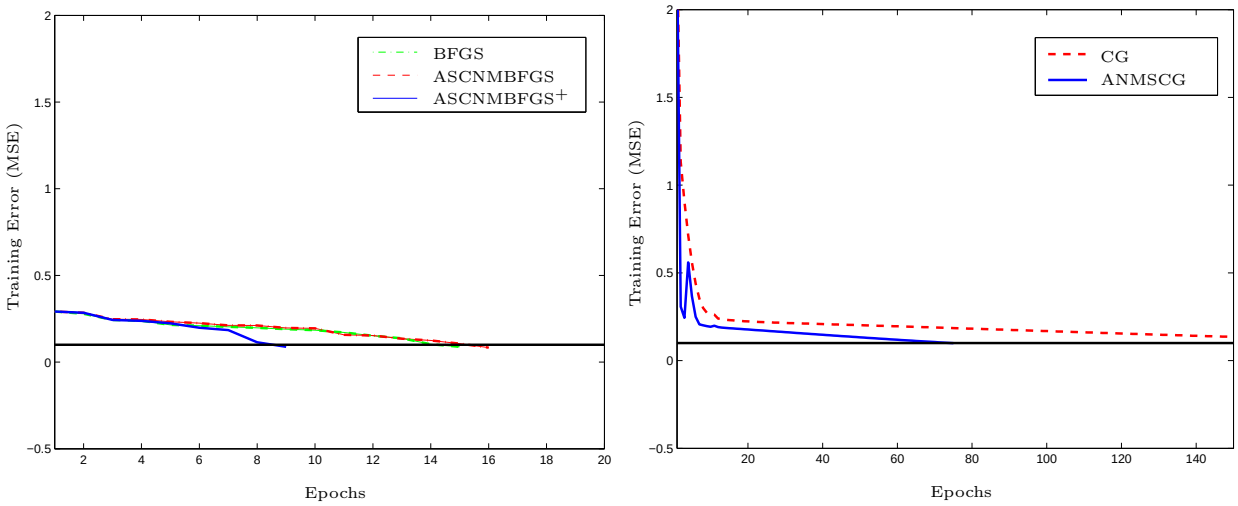

Figure 5: Performance comparison for training a recurrent neural for problem Monk-3 of quasi-Newton (left) and conjugate gradient algorithms (right).

4.3.4 Generalization Performance

Generalization performance is a decisive factor for the selection a training algorithm. Various techniques have been proposed in order to avoid overfitting and improve generalization performance [21, 28] and the results reported in the literature show that these techniques may help in many practical applications. However, their success depends on users choices, and on fine tuning or optimizing problem-dependent heuristic parameters [21, 47] such as the initial weights or the size of the training set.

As shown in Table 7 the generalization performance of all algorithms is heavily depended on the network architecture. As we expected all algorithms exhibit much better generalization results in case of training a recurrent neural network. We point out that ANMSCN is an excellent generalizer since it manages to correct classify most of the input patterns for every variant of Monk's problems. Regarding the generalization capability of quasi-Newton algorithms, it is worth noticing that ASCNMBFGS$^+$ outperforms the rest of the quasi-Newton algorithms, exhibiting equal or better generalization performance. Moreover, the results

	FFNN			ERNN		
Algorithm	Monk-1	Monk-2	Monk-3	Monk-1	Monk-2	Monk-3
BP	-	-	100%	-	-	100%
MBP	-	-	100%	-	-	100%
BBP	94.0%	95.7%	100%	96.0%	98.9%	100%
NMBBP	96.0%	99%	100%	98.0%	100%	100%
BPVS	93.9%	96.8%	100%	97.9%	98.6%	100%
NMBPVS	94.9%	96.9%	100%	97.9%	98.6%	100%
BFGS	97.0%	94.7%	100%	96.9%	98.4%	100%
ASCNMBFGS	99.0%	96.4%	100%	99.0%	100%	100%
ASCNMBFGS$^+$	99.0%	95.7%	100%	99.0%	100%	100%
CG	94.0%	96.8%	100%	94.0%	98%	100%
ANMCG	96.0%	96.8%	100%	94.7%	100%	100%
ASCNMSC	96.5%	97.3%	100%	96.7%	100%	100%

Table 7: Results of generalization in the Monk-1, Monk-2 and Monk-3 problem.

show that the incorporation of nonmonotone strategy in any training algorithm provides a significant improvement of its generalization performance.

5 Conclusion and Future Work

In this technical report, we investigated the performance of two proposed algorithms comparing them with various descent gradient methods and their modifications for training two different types of neural networks in a variety of classical artificial intelligence problems.

In general, self-scaled BFGS algorithms exhibit the best training performance and conjugate gradient the best generalization performance. As regards the revised adaptive self-scaling nonmonotone BFGS, the numerical experiments reveal that the application of the scaling factor (28) instead of the scaling factor (26) has a greater effect in the efficiency of classical BFGS algorithm combined with the adaptive nonmonotone strategy.

Regarding conjugate gradient algorithms, we noticed that the adaptive spectral conjugate gradient algorithm demonstrates much better possibility of successful training, requires less computational cost. Moreover, it leads to very good quality solutions in the sense that final weight vectors provide on the average improved generalization capability with no need for fine-tuning problem-dependent heuristic parameters such as the learning rate. Thus, we conclude that scaling the gradient by means of scaling parameter [5] and incorporating an adaptive nonmonotone strategy is worthwhile.

Furthermore, we point out that the performance of the second order algorithms especially of the conjugate gradient is heavily moderated by the use of the line search. The use of different and more advanced line search techniques [11, 13, 31, 34] will have a significant improvement in the efficiency of these algorithms [34].

Simulations results also pointed out that the nonmonotone strategy and the adaptation of different learning rate have a crucial affect on the practical behavior of backpropagation and conjugate gradient algorithms. As a result, we observed a significant improvement in the computational efficiency of classical algorithms, more stable learning and therefore a greater possibility of good performance.

Our future work is concentrated on performing a study on the training performance of the presented algorithms and mostly on the generalization ability of the trained networks on problems from the area of medicine and bioinformatics. Another interesting idea is to perform an investigation on the combination of neural network training algorithms with dimensional reducing techniques [16] for improving the generalization performance of the trained networks.

References

[1] M. Al-Baali. Descent property and global convergence of the Fletcher-Reeves method with inexact line search. *IMA Journal of Numerical Analysis*, 5:121–124, 1985.

[2] M. Al-Baali. Analysis of a family self-scaling quasi-Newton methods. *Computational Optimization and Applications*, 9:191–203, 1998.

[3] M. Al-Baali. Numerical experience with a class of self-scaling quasi-Newton algorithms. *Journal of Optimization Theory*, 96:533–553, 1998.

[4] L. Armijo. Minimization of functions having Lipschitz continuous partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[5] J. Barzilai and J.M. Borwein. Two point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

[6] L. Bello and M. Raydan. Convex constrained optimization for the seismic reflection tomography problem. *Journal of Applied Geophysics*, 62:158–166, 2007.

[7] E.G. Birgin, I.E. Chambouleyron, and J.M. Martínez. Optimization problems in the estimation of parameters of thin films and the elimination of the influence of the substrate. *Journal of Computational and Applied Mathematics*, 152:35–50, 2003.

[8] E.G. Birgin and J.M. Martínez. A spectral conjugate gradient method for unconstrained optimization. *Applied Mathematics and Optimization*, 43:117–128, 1999.

[9] R. Byrd, D. Liu, and J. Nocedal. On the behavior of Broyden's class of quasi-Newton methods. *SIAM Journal of Optimization*, 6:1025–1039, 1992.

[10] A. Cauchy. Methode generale pour la resulution des systemes d'equations simultanees. *Comp. Rend. Acad. Sci. Paris*, pages 536–538, 1847.

[11] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. *IEEE Proceedings*, 139(3):301–310, 1992.

[12] M. Contreras and R.A. Tapia. izing the bfgs and dfp updates: Numerical study. *Journal of Optimization Theory and Applications*, 78:93–108, 1993.

[13] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. PA: SIAM, Philadelphia, 1996.

[14] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, New York, first edition, 1987.

[15] R. Fletcher and C.M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

[16] I.K. Fodor. A survey of dimension reduction techniques. Technical Report LLNL, UCRL-ID-148494, Center for Applied Scientic Computing, Lawrence Livermore National Laboratory, June 2002.

[17] P. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, 1981.

[18] M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

[19] L. Grippo. A class of unconstrained minimization methods for neural network training. *Optimization Methods and Software*, 4:135–150, 1994.

[20] L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for newtons method. *SIAM Journal of Numerical Analysis*, 23(4):707–716, 1986.

[21] A. Gupta and S.M. Lam. Weight decay backpropagation for noisy data. *Neural Networks*, 11:1127–1137, 1998.

[22] J. Han, J. Sun, and W. Sun. Global convergence of non-monotone descent methods for unconstrained optimization problems. *Journal of Computational and Applied Mathematics*, 146:89–98, 2002.

[23] M.R. Hestenes. Iterative methods for solving nonlinear equations. *NAML Report No.52-9*, 1951.

[24] M.R. Hestenes and E. Stiefel. Methods for conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[25] H.Y Huang. A unified approach to quadratically convergent algorithms for function minimization. *Journal of Optimization Theory and Applications*, pages 405–423, 1970.

[26] R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[27] E.M. Johansson, F.U. Dowla, and D.M. Goodman. Backpropagation learning for multi-layer feed-forward neural networks using the conjugate gradient method. *International Journal of Neural Systems*, 2(4):291–301, 1992.

[28] G.N. Karystinos and D.A. Pados. On overfitting, generalization, and randomly expanded training sets. *IEEE Transactions on Neural Networks*, 11:1050–1057, 2000.

[29] G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis. Effective backpropagation with variable stepsize. *Neural Networks*, 10:69–82, 1997.

[30] M.F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[31] J.J. Moré and D.J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software*, 20:286–307, 1994.

[32] D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural network by choosing initial values of adaptive weights. *Biological Cybernetics*, 59:71–113, 1990.

[33] J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, 1:199–242, 1992.

[34] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[35] J. Nocedal and Y. Yuan. Analysis of a self-scaling quasi-Newton method. *Mathematical Programming*, 61:19–37, 1993.

[36] S.S. Oren. *Self-scaling variable metric algorithms for unconstrained minimization*. PhD thesis, Stanford University, California, USA, 1972.

[37] S.S. Oren and D.G. Luenberger. Self-scaling variable metric (SSVM) algorithms, Part I: Criteria and sufficient conditions for scaling a class of algorithms. *Management Science*, 20:845–862, 1974.

[38] C.C. Peng and G.D. Magoulas. Adaptive self-scaling non-monotone BFGS training algorithm for recurrent neural networks. In *Conference in Numerical Analysis*, pages 113–117, 2007.

[39] C.C. Peng and G.D. Magoulas. Adaptive nonmonotone conjugate gradient training algorithm for recurrent neural networks. In 19^{th} *IEEE International Conference on Tools with Artificial Intelligence*, pages 374–381, 2008.

[40] V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis. Determing nonmonotone strategies for effective training of multi-layer perceptrons. *IEEE Transactions on Neural Networks*, 13(6):1268–1284, 2002.

[41] V.P. Plagianakos, D.G. Sotiropoulos, and M.N. Vrahatis. Automatic adaptation of learning rate for backpropagation neural networks. In N.E. Mastorakis, editor, *Recent Advantages in Circuits and Systems*, pages 337–341, Singapure, 1998.

[42] V.P. Plagianakos, M.N. Vrahatis, and G.D. Magoulas. Nonmonotone methods backpropagation training with adaptive learning rate. In *Procedding of the IEEE International Joint Conference on Neural Networks*, volume 3, pages 1762–1767, Washingon D.C., 1999.

[43] E. Polak. *Optimization: Algorithms and Consistent Approximations*. Springer-Veglag, New York, 1997.

[44] E. Polak and G. Ribière. Note sur la convergence de methods de directions conjuguees. *Revue Francais d'Informatique et de Recherche Operationnelle*, 16:35–43, 1969.

[45] M.J.D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12:241–254, 1977.

[46] M.J.D. Powell. How bad are the BFGS and DFP methods when the objective function is quadratic. *Mathematical Programming*, 34:34–47, 1986.

[47] L. Prechelt. Automatic early stopping using cross validation: Quantifying the criteria. *Neural Networks*, 11:761–767, 1998.

[48] M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal of Optimization*, 7:26–33, 2007.

[49] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 318–362, Cambridge, Massachusetts, 1986.

[50] D.F. Shanno and K.H. Phua. Minimization of unconstrained multivariate functions. *ACM Transactions on Mathematical Software*, 2:87–94, 1976.

[51] D.G. Sotiropoulos, A.E. Kostopoulos, and T.N. Grapsa. A spectral version of Perry's conjugate gradient method for neural network training. In *Proceedings of the* 4^{th} *GRACM Congress on Computational Mechanics*, University of Patras, 2002.

[52] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Džeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CS-91-197, Pittsburgh, PA, 1991.

[53] M. Towsey, D. Alpsan, and L. Sztriha. Training a neural network with conjugate gradient methods. In *Proceedings of the ICNN'95*, pages 373–378, Perth, Western Australia, 1995.

[54] T.P. Vogl, J.K. Mangis, J.K. Rigler, W.T. Zink, and D.L. Alkon. Accelerating the convergence of the backpropagation method. *Biological Cybernetics*, 59:257–263, 1988.

[55] P.J. Werbos. Backpropagation through time: What it does and how to do it. In *Proceedings of the IEEE*, volume 78, pages 1550–1560, 1990.

[56] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[57] H.X. Yin and D.L. Du. The global convergence of self-scaling BFGS algorithm with nonmonotone line search for unconstrained nonconvex optimization problems. *Acta Mathematica Sinica*, 23(7):1233–1240, 2007.

[58] N. Zeev, O. Savasta, and D. Cores. Non-monotone spectral projected gradient method applied to full waveform inversion. *Geophysical Prospecting*, 54:525–534, 2006.

[59] Y. Zhang and R.P. Tewarson. Quasi-Newton algorithms with updates from the preconvex part of broyden's family. *IMA Journal of Numerical Analysis*, 8:487–509, 1988.