A memoryless BFGS neural network training algorithm

M.S. Apostolopoulou, D.G. Sotiropoulos, I.E. Livieris and P. Pintelas

***Abstract*—We present a new curvilinear algorithmic model for training neural networks which is based on a modifications of the memoryless BFGS method that incorporates a curvilinear search. The proposed model exploits the nonconvexity of the error surface based on information provided by the eigensystem of memoryless BFGS matrices using a pair of directions; a memoryless quasi-Newton direction and a direction of negative curvature. In addition, the computation of the negative curvature direction is accomplished by avoiding any storage and matrix factorization. Simulations results verify that the proposed modification significantly improves the efficiency of the training process.**

***Index Terms*—Neural networks, memoryless BFGS, negative curvature direction, curvilinear search.**

I. Introduction

Mathematically, the standard "training" problem of a neural network reduces to finding a set of weights w to minimize the error function $E(w)$, defined as the sum of the squares of the errors in the outputs [21]. Traditional gradient-based algorithms, update the weights using the following iterative formula

$$w_{k+1} = w_k + \eta_k p_k$$

where k is the current iteration usually called *epoch*, $w_0 \in \mathbb{R}^n$ is a given starting point, η_k is a stepsize (or learning rate) with $\eta_k > 0$ and p_k is a descent search direction, i.e., $g_k^T p_k < 0$. Moreover, the gradient can be easily obtained by means of back propagation of errors through the network layers. In the literature, there have been proposed many suggestions [5], [9] to define the search direction p_k and most of them use second order information. The most elaborated directions are the limited memory quasi-Newton direction [14], [18] where the search direction is defined by building up a Hessian approximation using curvature information from the most recent iterations.

Recently, it has been proposed a method that exploits the eigenstructure of the memoryless BFGS matrices without using storage and matrix factorization [2]. Consequently, a direction of negative curvature can be computed analytically avoiding the storage of any matrix. Therefore, the method is well-suited for large scale problems, such as training neural networks. Motivated by [2], we propose a curvilinear scheme which is based on a modification of a memoryless quasi-Newton method for training neural networks. The proposed algorithm utilizes a pair of directions; a memoryless quasi-Newton direction and a direction of negative curvature, i.e., directions d such that $d^T \nabla^2 E(w) d < 0$, and it is based on the following iterative form

$$w_{k+1} = \begin{cases} w_k + \eta_k p_k, & \text{if } B_k \text{ is positive definite;} \\ w_k + \eta_k^2 p_k + \eta_k d_k, & \text{otherwise} \end{cases}$$

where p_k is a memoryless quasi-Newton direction, d_k is a direction of negative curvature and B_k is the memoryless BFGS Hessian approximation. When B_k is positive definite, the proposed iterative scheme is the standard linesearch procedure (see [3], [19]). In different case, the iterative scheme searches along the curvilinear path $w_{k+1} = w_k + \eta_k^2 p_k + \eta_k d_k$ which was first proposed by Moré and Sorensen [15]. The proposed method preserves the strong convergence properties provided by the quasi-Newton direction when B_k is positive definite. Additionally, it exploits the nonconvexity of the error surface through the computation of the negative curvature direction without using any storage and matrix factorization.

Notations. Throughout the paper $\|\cdot\|$ denotes the Euclidean norm and n the dimension of the error function. We indicate that a matrix A is positive definite by $A > 0$. The gradient of the error function is denoted by $g_k = \nabla E(w^k)$.

II. Properties of the memoryless BFGS matrices

The memoryless BFGS matrices are computed based on the L-BFGS philosophy [14], [18] using information from the most recent iteration. Given an initial matrix $B_0 = (1/\theta)I$, $\theta \in \mathbb{R}\backslash\{0\}$, and the BFGS formula

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

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$, the resulting minimal memory BFGS update scheme takes the form

$$B_{k+1} = \frac{1}{\theta_{k+1}} I - \frac{s_k s_k^T}{\theta_{k+1} s_k^T s_k} + \frac{y_k y_k^T}{s_k^T y_k}. \tag{2}$$

Additionally, it is known that the inverse of B_{k+1} is given by the following expression [18]

$$B_{k+1}^{-1} = \theta_{k+1} I - \theta_{k+1} \frac{y_k s_k^T + s_k y_k^T}{s_k^T y_k} + \frac{s_k s_k^T}{s_k^T y_k} \tag{3}$$

In our approach we consider the case where the scalar parameter θ is defined as $\theta_{k+1} = \left(s_k^T s_k\right) / \left(s_k^T y_k\right)$ which is the spectral parameter of Barzilai and Borwein [4].

*L*emma 1 ([2]) Let the symmetric memoryless BFGS matrix defined in (2). Then, the characteristic polynomial of $B_{k+1} \in \mathbb{R}^{n \times n}$ has the general form

M. S. Apostolopoulou is with the Department of Mathematics University of Patras, Greece. E-mail: msa@math.upatras.gr

D. G. Sotiropoulos is with the Department of Informatics Ionian University, Greece. E-mail: dgs@ionio.gr

I. E. Livieris is with the Department of Mathematics University of Patras, Greece. E-mail: livieris@upatras.gr

P. Pintelas is with the Department of Mathematics University of Patras, Greece. E-mail: pintelas@math.upatras.gr

978-1-4244-3760-3/09/$25.00 © 2009 IEEE

$$p(\lambda) = \left(\lambda - \frac{1}{\theta_{k+1}}\right)^{n-2} \left(\lambda^2 - \frac{a_k}{\theta_{k+1}}\lambda + \frac{1}{\theta_{k+1}^2}\right), \quad (4)$$

where $a_k = 1 + \theta_{k+1}\frac{y_k^T y_k}{s_k^T y_k}$. Moreover, if $a_k > 2$, then $\lambda_1 < 1/\theta_{k+1} < \lambda_n$, where λ_1 and λ_n are the smallest and largest eigenvalues of B_{k+1}, respectively.

The parameter a_k is bounded from below by 2, since

$$a_k = 1 + \theta_{k+1}\frac{y_k^T y_k}{s_k^T y_k} = 1 + \frac{\|s_k\|^2\|y_k\|^2}{(s_k^T y_k)^2} = 1 + \frac{1}{cos^2\phi} \geq 2,$$

where ϕ is the angle between s_k and y_k. If $a_k > 2$, the extreme eigenvalues can be computed by solving the quadratic equation $\lambda^2 - (a_k/\theta_{k+1})\lambda + 1/\theta_{k+1}^2 = 0$. In contrast, if $a_k = 2$, then the characteristic polynomial is reduced to $p(\lambda) = (\lambda - 1/\theta_{k+1})^n$; thus $B_{k+1} = (1/\theta_{k+1})I$.

*P*roposition 1 ([2]) Let Λ be the set of eigenvalues of B_{k+1} with opposite signs. Then, for any $\lambda \in \mathbb{R} \setminus \Lambda$, the matrix $(B_{k+1} + \lambda I)$ is invertible and its inverse can be expressed by the following closed-form

$$(B_{k+1} + \lambda I)^{-1} = \frac{1}{\gamma}\sum_{i=0}^{2} (-1)^i \gamma_i(\lambda)\,(B_{k+1})^i \quad (5)$$

where the quantities $\gamma = (1/\theta_{k+1} + \lambda)(\lambda^2 + a_k\lambda/\theta_{k+1} + 1/\theta_{k+1}^2)$, $\gamma_2 = 1$, $\gamma_1 = \lambda + (a_k+1)/\theta_{k+1}$ and $\gamma_0 = \lambda^2 + (a_k+1)\lambda/\theta_{k+1} + (a_k+1)/\theta_{k+1}^2$ are functions of λ.

For determining the eigenvector corresponding to the smallest eigenvalue of B_{k+1}, we consider the following cases.

Case 1: If the smallest eigenvalue of B_{k+1} is distinct, then the corresponding eigenvector is computed by applying a single step of the inverse iteration. Given a non-zero starting vector u_0, inverse iteration generates a sequence of vectors u_i, generated recursively by the formula

$$u_i = \left(B - \hat{\lambda} I\right)^{-1} \frac{u_{i-1}}{\|u_{i-1}\|}, \quad i = 1, 2, \ldots$$

where $\hat{\lambda} = \lambda + \epsilon$, λ is a distinct eigenvalue of B and $\epsilon \to 0^+$. The sequence of iterates u_i converges to an eigenvector associated with an eigenvalue closest to $\hat{\lambda}$. Moreover, if this particular eigenvalue λ is known exactly, this method converges in a single iteration [12]. Hence, using Lemma 1 and Proposition 1 and after some simple algebraic computations, the expression for the eigenvector is defined by $u_1 = \hat{u}_1/\|\hat{u}_1\|$, where

$$\begin{aligned}\hat{u}_1 &= \sum_{i=0}^{2} (-1)^i \gamma_i(\hat{\lambda})\,(B_{k+1})^i \frac{u}{\gamma(\hat{\lambda})} \\ &= -\gamma_u(\hat{\lambda})\,u + \gamma_{us}(\hat{\lambda})\,s_k - \gamma_{uy}(\hat{\lambda})\,y_k, \end{aligned} \quad (6)$$

with $\hat{\lambda} = -\lambda_1 + \epsilon$, $u = u_0/\|u_0\|$ and the coefficients are

$$\gamma_u(\hat{\lambda}) = \left[1 - \gamma_1(\hat{\lambda})\,\theta_{k+1} + \gamma_0(\hat{\lambda})\,\theta_{k+1}^2\right] / \left[\gamma(\hat{\lambda})\,\theta_{k+1}^2\right],$$
$$\gamma_{us}(\hat{\lambda}) = \left\{\left[1 - \gamma_1(\hat{\lambda})\,\theta_{k+1}\right] s_k^T u + \theta_{k+1} y_k^T u\right\} / \left[\gamma(\hat{\lambda})\,\theta_{k+1}^2 s_k^T s_k\right],$$
$$\gamma_{uy}(\hat{\lambda}) = \left\{\left[1 - \gamma_1(\hat{\lambda})\,\theta_{k+1} + a_k\right]\theta_{k+1} y_k^T u - s_k^T u\right\} / \left[\gamma(\hat{\lambda})\,\theta_{k+1}^2 s_k^T y_k\right].$$

Case 2: If the smallest eigenvalue of B_{k+1} is multiple, then from Lemma 1 we have that $a_k = 2$ and $B_{k+1} = (1/\theta_{k+1})I$. Thus, using the eigendecomposition of B it follows that $B = U\Lambda U^T$, where $U = I$ and $\Lambda = \text{diag}\,(\lambda_1, \lambda_1, \ldots, \lambda_1)$. It is easy to verify that an eigenvector corresponding to λ_1 is $u_1 = e_1 = (1, 0, \ldots, 0)^T$.

III. Curvilinear Memoryless BFGS Algorithm

A. Computation of the descent pair of directions

At this point, we recall that our new proposed curvilinear scheme uses a pair of directions; a quasi-Newton direction [19] which is defined as

$$p_{k+1} = \begin{cases} -B_{k+1}^{-1} g_{k+1}, & B_{k+1} > 0; \\ -g_{k+1}, & \text{otherwise.} \end{cases} \quad (7)$$

where B_{k+1}^{-1} is defined in equation (3) and a direction of negative curvature [15] which is calculated by

$$d_{k+1} = \begin{cases} 0, & B_{k+1} > 0; \\ -\text{sgn}(u_1^T g_{k+1})u_1, & \text{otherwise,} \end{cases} \quad (8)$$

where u_1 is a normalized eigenvector corresponding to the most negative eigenvalue of B_{k+1}. We note that both directions must be *descent* directions. In a given iterate w_k, the pair (p_k, d_k) is assumed to be a sufficient descent pair of directions if $\{p_k\}_{k=0}^{\infty}$ and $\{d_k\}_{k=0}^{\infty}$ are bounded and satisfy the following conditions [8]:

*C*ondition 1: $g_k^T p_k = 0$ implies $g_k = 0$ and $p_k = 0$.
*C*ondition 2: $g_k^T p_k \to 0$ implies $g_k \to 0$ and $p_k \to 0$.
*C*ondition 3: $d_k^T B_k d_k \to 0$ implies $\min(\lambda_1, 0) \to 0$ and $d_k \to 0$, where λ_1 is the smallest eigenvalue of B_k.

Conditions 1 and 2 are the standard ones for quasi-Newton directions. They ensure that $p_k \neq 0$ and $g_k \neq 0$ are not orthogonal and do not become nearly so too rapidly. Condition 3 ensures that d_k contains information related to the smallest eigenvalue of the approximate Hessian.

When the vectors s_k and y_k are linearly independent, i.e. $a_k > 2$, then, the quasi-Newton direction is computed using relation (7). Additionally, the negative curvature direction is computed by means of relations (6) and (8).

In case where the vectors s_k and y_k are collinear, i.e., $a_k = 2$, then the quasi-Newton direction is reduced to

$$p_{k+1} = \begin{cases} -\dfrac{g_{k+1}}{\theta_{k+1}}, & B_{k+1} > 0; \\ -g_{k+1}, & \text{otherwise.} \end{cases} \quad (9)$$

B. Model training algorithm `CM-BFGS`

At this point, we present a high level description of our proposed algorithm based on the Armijo procedure.

Step 1: Initiate w_0, $0 < c_1 < c_2 < 1$, Err and $\epsilon \to 0$; set $k = 0$.
Step 2: If $(E(w_k) < Err)$ or $(\|\nabla E(w_k)\|_2 < \epsilon)$ terminate; else compute the eigenvalues λ_i of B_k.
Step 3: If $\lambda_1 > 0$ then
(a) Compute p_k; set $d_k = 0$ and $\eta_k = 1$.
(b) Find $\eta_k > 0$ such that

$$E(w_k + \eta_k p_k) \leq E(w_k) + c_1\eta_k g_k^T p_k$$

Step 4: Else if $\lambda_1 \leq 0$ then

(a) Set $p_k = -g_k$ and compute the normalized eigenvector u_1; set $d_k = -\text{sgn}\left(u_1^T g_k\right) u_1$ and $\eta_k = 1$.

(b) Find $\eta_k > 0$ such that

$$E(w_k + \eta_k^2 p_k + \eta_k d_k) \leq E(w_k) + c_2 \eta_k \left(g_k^T d_k + \frac{1}{2}\lambda_1\right)$$

Step 5: Update the weights

$$w_{k+1} = \begin{cases} w_k + \eta_k p_k, & \text{if } \lambda_1 > 0; \\ w_k + \eta_k^2 p_k + \eta_k d_k, & \text{otherwise} \end{cases}$$

Step 6: Compute g_{k+1}, $s_k = w_{k+1} - w_k$ and $y_{k+1} = g_{k+1} - g_k$; if $|s_k^T y_k| > 10^{-6}\|s_k\|\|y_k\|$, update the vector pair $\{s_k, y_k\}$.

Step 7: Set $k = k+1$ and goto Step 2.

Remarks: In Step 2, the computation of the eigenvalues is based on Lemma 1. In Step 3(a), p_k is computed either by relation (7) or by relation (9) (that is, in case $a_k > 2$ or $a_k = 2$, respectively). In Step 4(a), if $a_k > 2$, then d_k is computed using relation (6), in contrast we set $d_k = -\text{sgn}(g_{k+1}^{(1)})(1, 0, \ldots, 0)^T$. Finally, in Step 6 we skip the update in case $|s_k^T y_k| \leq 10^{-6}\|s_k\|\|y_k\|$ to ensure that B_k is well defined.

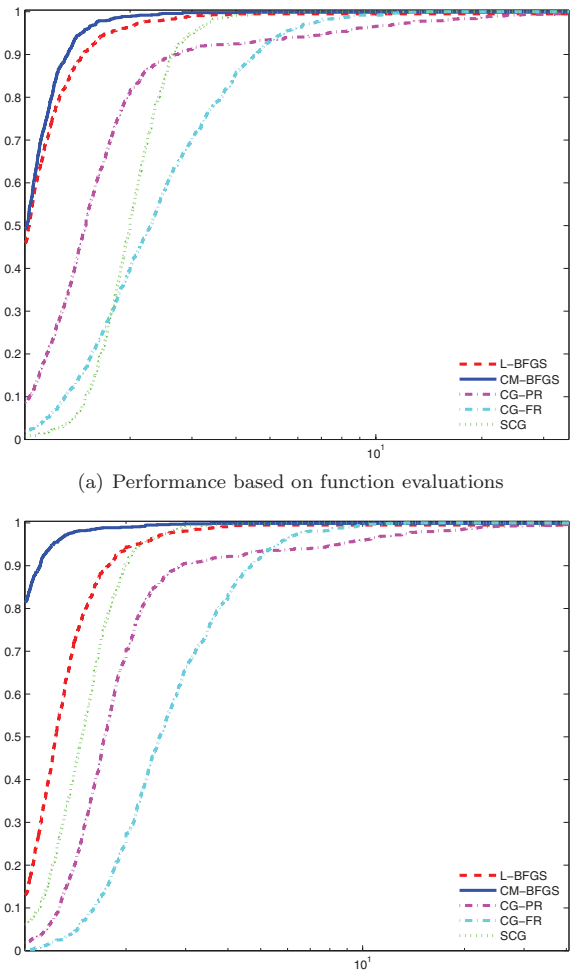
IV. Experimental Results

In the following section we will evaluate the performance of our proposed method in six famous benchmarks acquired by the UCI Repository of Machine Learning Databases [16]. In the following subsections we briefly describe each problem and present the performance comparison between our proposed curvilinear memoryless BFGS (`CM-BFGS`) with the linesearch memoryless BFGS (`L-BFGS`) [18], [14] the conjugate gradient methods of Polak-Ribiére (`CG-PR`) and Fletcher-Reeves (`CG-FR`) and with the scaled conjugate gradient (`SCG`).

The parameters in `CM-BFGS` were set as $\sigma_1 = \sigma_2 = 10^{-4}$ and for the rest algorithms we have used the standard parameters as in [18] for all experiments. All networks have received the same sequence of input patterns and the initial weights were initiated using the Nguyen-Widrow method [17]. For evaluating classification accuracy we have used the standard procedure called *k-fold cross-validation* [13]. We have also used the Libopt environment [7] to create the performance profiles proposed by Dolan and Moré [6] to present perhaps the most complete information in terms of functions and gradient evaluations. All simulations have been carried out on a processor Pentium-IV dual core computer (2.0MHz, 1Gbyte RAM) running Linux operating system.

A. Breast cancer problem

The first benchmark concerns the diagnosis of breast cancer malignancy. The data have been collected from 683 patients from the University of Wisconsin each having 9 attributes and a class label (malignant or benign tumor). We have used neural networks with 2 hidden layers of 4 and 2 neurons respectively, as suggested in the PROBEN1 benchmark collection [20]. The termination criterion is set to $E \leq 0.02$ within the limit of 2000 epochs and all networks were tested using 10-fold cross validation.



(a) Performance based on function evaluations

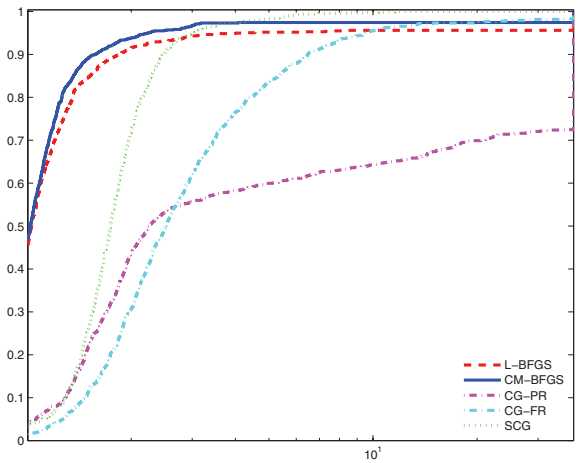
(b) Performance based on gradient evaluations

Fig. 1. Log_{10} scaled performance profile for number of function and gradient evaluations for the breast cancer problem.

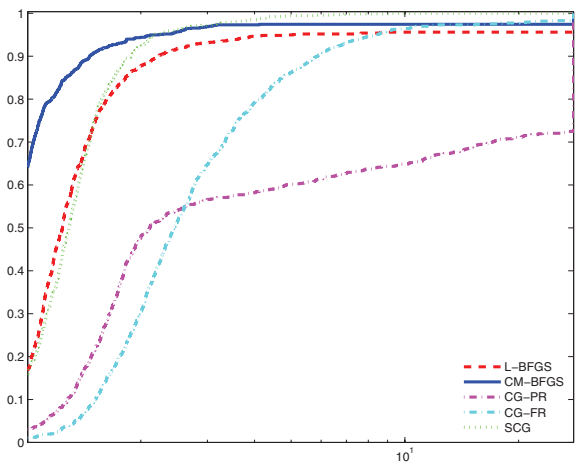
In Figure 1 are presented the performance profiles for the breast cancer problem, investigating the efficiency and robustness of each training method. As regards the function evaluations metric, `CM-BFGS` and `L-BFGS` are the top performers with `CM-BFGS` exhibiting slightly better efficiency. Furthermore, the performance profile relative to the gradient evaluation metric reports that `CM-BFGS` is much more robust and more efficient than all other training methods.

B. Diabetes problem

The aim of this real-world classification task is to decide when a Pima Indian female is diabetes positive or not. We have used neural networks with 2 hidden layers of 4 neurons each and an output layer of 2 neurons [20]. The termination criterion is set to $E < 0.14$ within the limit of 2000 epochs and all networks were tested using 10-fold cross validation.



(a) Performance based on function evaluations



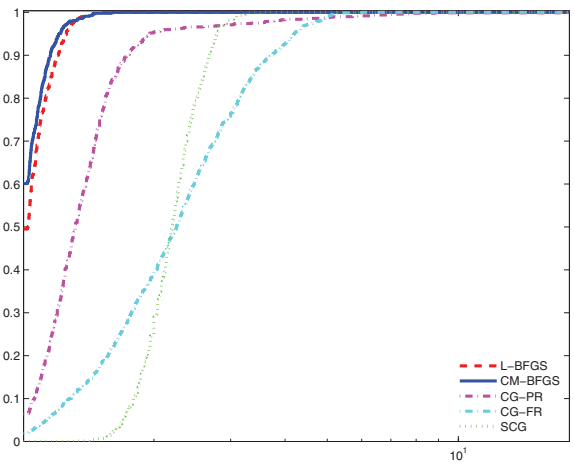
(b) Performance based on gradient evaluations

Fig. 2. Log_{10} scaled performance profile for number of function and gradient evaluations for the diabetes problem.

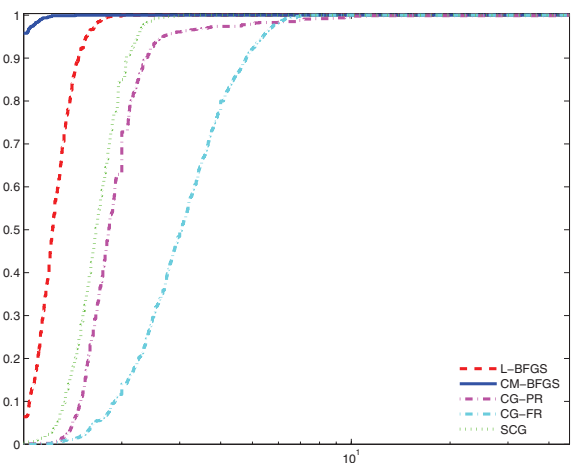
Figure 2 presents the performance profiles for both performance metrics for the diabetes problem. In Figure 2(a) we demonstrate that `CM-BFGS` slightly outperforms `L-BFGS` and exhibits much better performance comparatively to the conjugate gradient methods, in terms of function evaluations. Additionally, Figure 2(b) reports that relative to the gradient evaluations metric, our proposed method is the most efficient and robust method outperforming all other methods.

C. SPECT Heart Problem

This dataset contains data instances derived from cardiac Single Proton Emission Computed Tomography (SPECT) images from the University of Colorado [16]. This is also a binary classification task, where patients heart images are classified as normal or abnormal. The class distribution has 55 instances of the abnormal class (20.6%) and 212 instances of the normal class (79.4%). The network architectures for this medical classification problem constitute of 1 hidden layer with 6 neurons and an output layer of 1 neuron [22]. The termination criterion is set to $E \leq 0.1$ within the limit of 1000 epochs.



(a) Performance based on function evaluations



(b) Performance based on gradient evaluations

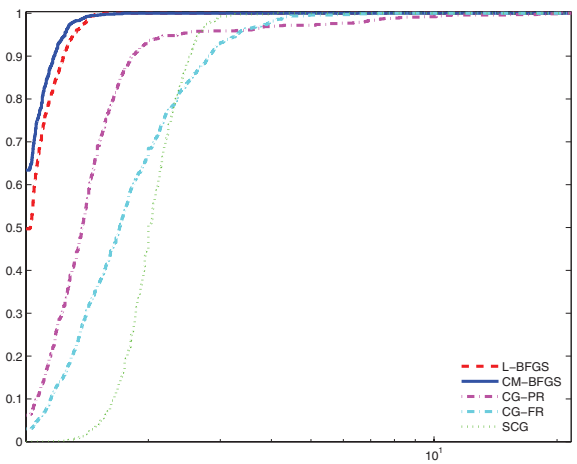
Fig. 3. Log_{10} scaled performance profile for number of function and gradient evaluations for the spect heart problem.

Figure 3 presents the performance profiles for the SPECT heart problem. Regarding the function evaluations metric, we can observe that `CM-BFGS` is slightly more robust than `L-BFGS` and significantly outperforms all conjugate gradient methods. However, the performance profile relative to the gradient evaluation metric reports that `CM-BFGS` is much more robust and efficient comparatively to all training methods.

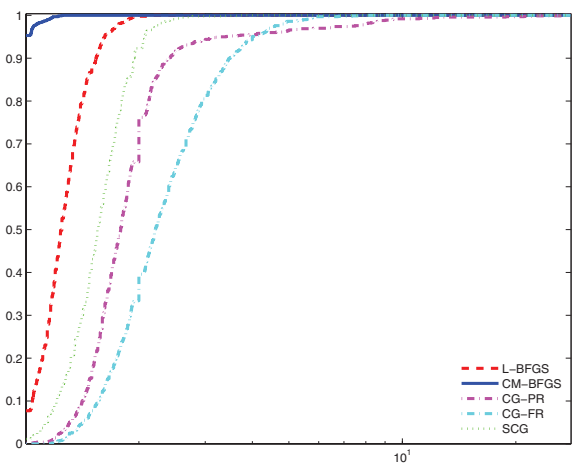
D. Australian credit approval problem

Australian Credit Approval dataset contains all the details about credit card applications. This dataset is interesting because the data varies and has mixture of attributes which is continuous, nominal with small numbers of values, and nominal with larger numbers of values. We have used

neural networks with 2 hidden layers consist of 16 and 8 neurons and an output layer of 2 neurons [20] which provide us the best generalization results. The termination criterion is set to $E \leq 0.1$ within the limit of 1000 epochs and all networks were tested using 10-fold cross validation.



(a) Performance based on function evaluations



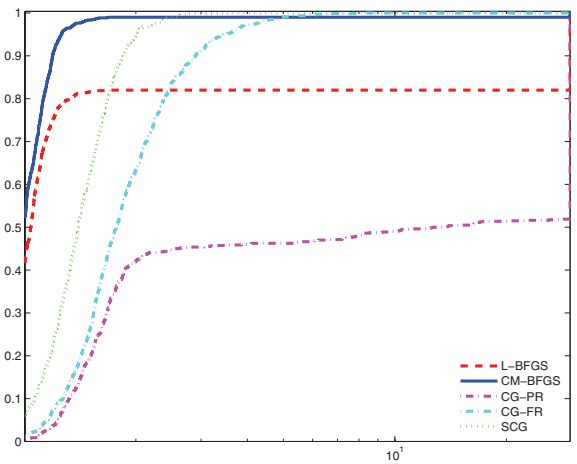
(b) Performance based on gradient evaluations

Fig. 4. Log_{10} scaled performance profile for number of function and gradient evaluations for the Australian credit approval problem.

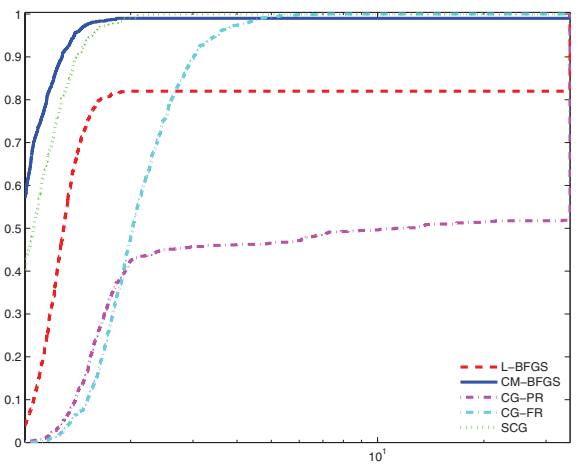
Figure 4 illustrates the performance profiles for the Australian credit approval problem investigating the efficiency and robustness of each training method. Figure 4(a) reports that `CM-BFGS` is most robust and efficient method relative to the function evaluations metric since its curve lies above the curves of all other methods. Additionally in Figure 4(b), it is interesting to observe that `CM-BFGS` is the top performer, significantly outperforming all other training methods.

E. Escherichia coli problem

This problem is based on a drastically imbalanced dataset of 336 patterns and concerns the classification of the *E. coli* protein localization patterns into eight localization sites. The network architectures constitute of 1 hidden layer with 16 neurons and an output layer of 8 neurons [1]. The termination criterion is set to $E \leq 0.02$ within the limit of 2000 epochs and all networks were tested using 4-fold cross-validation [11].



(a) Performance based on function evaluations



(b) Performance based on gradient evaluations

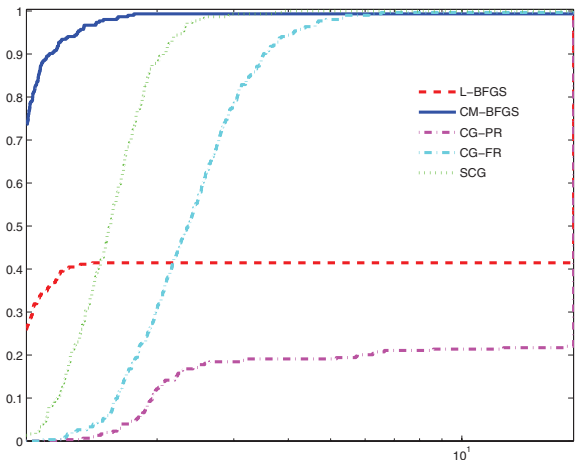
Fig. 5. Log_{10} scaled performance profile for number of function and gradient evaluations the escherichia coli problem.

In Figure 5 are presented the performance profiles for both performance metrics for the escherichia coli problem. In general, the best overall performance relative to both performance metrics was obtained by `CM-BFGS` which significantly outperforms all other training methods, especially in terms of gradient evaluations.

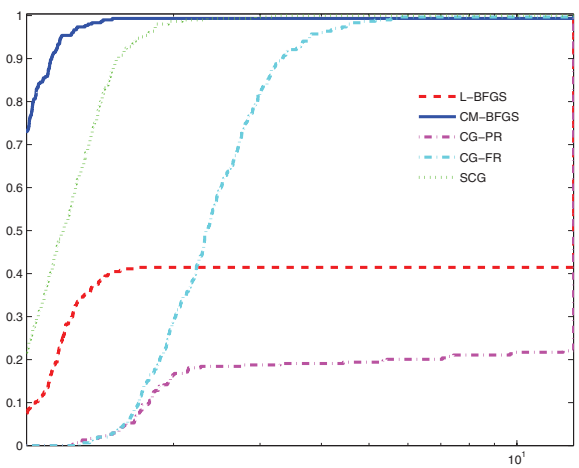
F. Yeast problem

This problem is similar with the Escherichia coli problem, namely the determination of the cellular localization of the yeast proteins. Saccharomyces cerevisiae (yeast) is the simplest Eukaryotic organism. We have used neural

networks with 1 hidden layer of 16 neurons and an output layer of 10 neurons [1]. The termination criterion is set to $E < 0.05$ within the limit of 2000 epochs and all networks were tested using 10-fold cross validation [10].



(a) Performance based on function evaluations



(b) Performance based on gradient evaluations

Fig. 6. Log_{10} scaled performance profile for number of function and gradient evaluations the yeast problem.

Figure 6 illustrates the performance profiles of each training method for the yeast problem. Similar observations can be made with the previous problem. More specifically, `CM-BFGS` is the most robust and efficient method since it exhibits the top performance outperforming all other training methods relative to both performance metrics.

V. Conclusions

In this work, we have proposed a new curvilinear method for training neural networks which is based on the analysis of the eigenstructure of the memoryless BFGS matrices. The proposed method preserves the strong convergence properties provided by the quasi-Newton direction while simultaneously it exploits the nonconvexity of the error surface through the computation of the negative curvature direction without using any storage and matrix factorization. Based on the fact that the algorithm uses only inner products and vector summations, the proposed method is suitable for training large scale neural networks. Our numerical experiments have shown that the proposed method outperforms other popular training methods on famous benchmarks.

References

[1] A.D. Anastasiadis, G.D. Magoulas, and M.N. Vrahatis. New globally convergent training scheme based on the resilient propagation algorithm. *Neurocomputing*, 64:253–270, 2005.

[2] M.S. Apostolopoulou, D.G. Sotiropoulos, and P. Pintelas. Solving the quadratic trust-region subproblem in a low-memory BFGS framework. *Optimization Methods and Software*, 23(5):651–674, 2008.

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

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

[5] R. Battiti. First and second order methods for learning: between steepest descent and Newton's method. *Neural Computation*, 4:141–166, 1992.

[6] E. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[7] J.Ch. Gilbert and X. Jonsson. LIBOPT-An enviroment for testing solvers on heterogeneous collections of problems - version 1. CoRR, abs/cs/0703025, 2007.

[8] D. Goldfarb. Curvilinear path steplength algorithms for minimization which use directions of negative curvature. *Mathematical Programming*, 18:31–40, 1980.

[9] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, MA, 1991.

[10] P. Horton and K. Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. In 4^{th} *International Conference on Intelligent Systems for Molecular Biology*, pages 109–115, 1996.

[11] P. Horton and K. Nakai. Better prediction of protein cellular localization sites with the k Nearest Neighbors classifier. In *Intelligent Systems in Molecular Biology*, pages 368–383, 1997.

[12] I. Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 39:254–291, 1997.

[13] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, pages 223–228. AAAI Press and MIT Press, 1995.

[14] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization methods. *Mathematical Programming*, 45:503–528, 1989.

[15] J.J. Moré and D. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, 16:1–20, 1979.

[16] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, 1994.

[17] 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.

[18] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematical Computing*, 35(151):773–782, 1980.

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

[20] L. Prechelt. PROBEN1-A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultt fr Informatik, University of Karlsruhe, 1994.

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

[22] M. Zhang and P. Wong. Genetic programming for medical classification: a program simplification approach. *Genetic Programming and Evolvable Machines*, 9:229–255, 2008.