Performance Evaluation of Descent CG Methods for Neural Network Training

I.E. Livieris* and P. Pintelas

***Abstract*—Conjugate gradient methods constitute an excellent choice for efficiently training large neural networks since they don't require the evaluation of the Hessian matrix neither the impractical storage of an approximation of it. Despite the theoretical and practical advantages of these methods their main drawback is the use of restarting procedures in order to guarantee convergence, abandoning second order derivative information [31]. In this work, we propose a neural network training algorithm which preserves the advantages of classical conjugate gradient methods and simultaneously avoids the inefficient restarts. Encouraging numerical experiments verify that the presented algorithm provides fast, stable and reliable convergence.**

***Index Terms*—Neural networks, descent spectral conjugate gradient methods, sufficient descent property, truncate strategy.**

I. 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. Nowadays, they have been established as vital components of many systems and are considered as a powerful tool for pattern classification [6].

It is well known that the problem of training a neural network is highly consistent with the unconstrained optimization theory. More analytically, it can be formulated as the minimization of the error function $E(w)$ that depends on the connection weights w of the network, defined as the sum of squares of the errors in the outputs [35]. A traditional way to solve this problem is by an iterative gradient-based training algorithm which generates a sequence of weights $\{w\}_{k=0}^{\infty}$, starting from an initial point $w^0 \in \mathbb{R}^n$ using the recurrence

$$w_{k+1} = w_k + \eta_k d_k \tag{1}$$

where k is the current iteration usually called *epoch*, $\eta_k > 0$ is the learning rate and d_k is a descent search direction, i.e., $g_k^T d_k < 0$. Since the appearance of backpropagation [35] a variety of approaches that use second order derivative related information was suggested for improving the efficiency of the minimization error process, such as the conjugate gradient methods. In general, conjugate gradient methods constitute an excellent choice for efficiently training large neural networks due to their simplicity and their very low memory requirements since they don't require the evaluation of the Hessian matrix neither the impractical storage of an approximation of it. The family of conjugate gradient methods includes a variety of variants with interesting convergence properties and numerical efficiency.

The convergence analysis of these methods has been widely studied by many researchers [10]-[12], [23], [28]-[33] and it is usually based on mild conditions which refer to the Lipschitz and boundedness assumptions and is closely connected with the sufficient descent property

$$g_k^T d_k \le -c\|g_k\|^2 \tag{2}$$

where c is a fixed constant. Hager and Zhang [17] have presented an excellent survey of conjugate gradient methods, with special attention to global convergence properties.

Despite the theoretical and practical advantages of conjugate gradient methods, the main drawback of these methods is the use of restarting procedures in order to guarantee convergence. Nevertheless, there is also a worry with restart algorithms that restart may be triggered too often, abandoning the second-order derivative information; thus degrading the overall efficiency of the method [27]. Recently, Yu et al. [39] motivated by [15], [38] proposed a new class of conjugate gradient methods which guarantee the sufficient descent property independent of the accuracy of the line search, avoiding thereby the restarting procedures. Additionally, they established that their proposed method converges globally for general nonconvex functions under the Wolfe line search conditions [39].

In this work, we evaluate the performance of this class of methods and we propose a new conjugate gradient algorithm for training neural networks. The proposed algorithm preserves the advantages of classical conjugate gradient methods while simultaneously avoids the inefficient restarts.

The remainder of this paper is organized as follows. Section II presents a brief summary of conjugate gradient methods and in Section III are presented the proposed conjugate gradient method and two truncated schemes for neural network training. Experimental results are reported in Section IV and Section V presents our concluding remarks and our proposals for future research.

Notations. Throughout the paper $\|\cdot\|$ denotes the Euclidean norm, $\langle\cdot\rangle$ stands for the inner product and the gradient of the error function is indicated by $\nabla E(w_k) = g_k$.

II. Conjugate Gradient Methods

The basic idea for determining the search direction in conjugate gradient methods is the linear combination of the negative gradient vector at the current iteration with the previous search

*Corresponding Author

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`

direction, that is

$$d_k = \begin{cases} -g_k, & \text{if } k = 0; \\ -g_k + \beta_k d_{k-1}, & \text{otherwise} \end{cases} \tag{3}$$

Conjugate gradient methods differ in their way of defining the scalar parameter β_k. In the literature, there have been proposed several choices for β_k which give rise to distinct conjugate gradient methods.

$$\beta_k^{HS} = \frac{g_k^T y_{k-1}}{y_{k-1}^T d_{k-1}} \quad \text{(Hestenes-Stiefel [18])} \tag{4}$$

$$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2} \quad \text{(Fletcher-Reeves [11])} \tag{5}$$

$$\beta_k^{PR} = \frac{g_k^T y_{k-1}}{\|g_{k-1}\|^2} \quad \text{(Polak-Ribière [30])} \tag{6}$$

$$\beta_k^{CD} = -\frac{\|g_k^T\|}{d_{k-1}^T g_{k-1}} \quad \text{(Conjugate Descent [10])} \tag{7}$$

$$\beta_k^{LS} = \frac{g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}} \quad \text{(Liu-Storey [23])} \tag{8}$$

$$\beta_k^{DY} = \frac{\|g_k\|^2}{y_{k-1}^T d_{k-1}} \quad \text{(Dai-Yuan [8])} \tag{9}$$

$$\beta_k^{P} = \frac{g_k^T (y_{k-1} - s_{k-1})}{y_{k-1}^T d_{k-1}} \quad \text{(Perry [29])} \tag{10}$$

where $s_{k-1} = w_k - w_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$.

Birgin and Martínez [5] introduced the class of spectral conjugate gradient methods embedding the spectral gradient [4] in the conjugate gradient framework, where the search direction is determined as the linear combination of the spectral negative gradient with the previous search direction, namely

$$d_k = \begin{cases} -g_k, & \text{if } k = 0; \\ -\frac{1}{\delta_k} g_k + \beta_k d_{k-1}, & \text{otherwise} \end{cases} \tag{11}$$

where δ_k is a scalar, usually defined as the Rayleigh quotient:

$$\frac{s_{k-1}^T \left[\int_0^1 \nabla^2 E(w_{k-1} + \delta s_{k-1}) d\delta\right] s_{k-1}}{\|s_{k-1}\|^2}$$

i.e.

$$\delta_k = \frac{\langle y_{k-1}, s_{k-1}\rangle}{\langle s_{k-1}, s_{k-1}\rangle} \tag{12}$$

which lies between the minimum and the maximum eigenvalue of the average Hessian $\int_0^1 \nabla^2 E(w_{k-1} + \delta s_{k-1}) d\theta$. The motivation for this selection constitutes in proving a two point approximation to the secant equation underlying Quasi-Newton methods [4]. Using a geometric interpretation for quadratic function minimization Birgin and Martínez suggested the following expression for defining the update parameter β_k in Eq. (11)

$$\beta^{SP} = \frac{g_k^T (y_{k-1} - \delta_k s_{k-1})}{\delta_k y_{k-1}^T d_{k-1}} \tag{13}$$

Clearly, if $\delta_k = 1$ this formula is reduced to the classical formula Eq. (10), introduced by Perry [29]. Along this line, the rest of the spectral conjugate gradient formulas can be presented as follows:

$$\beta_k^{SHS} = \frac{g_k^T y_{k-1}}{\delta_k y_{k-1}^T d_{k-1}} \tag{14}$$

$$\beta_k^{SFR} = \frac{\delta_{k-1}\|g_k\|^2}{\delta_k \|g_{k-1}\|^2} \tag{15}$$

$$\beta_k^{SPR} = \frac{\delta_{k-1} g_k^T y_{k-1}}{\delta_k \|g_{k-1}\|^2} \tag{16}$$

$$\beta_k^{SCD} = -\frac{\delta_{k-1}\|g_k^T\|}{\delta_k d_{k-1}^T g_{k-1}} \tag{17}$$

$$\beta_k^{SLS} = \frac{\delta_{k-1} g_k^T y_{k-1}}{\delta_k d_{k-1}^T g_{k-1}} \tag{18}$$

$$\beta_k^{SDY} = \frac{\|g_k\|^2}{\delta_k y_{k-1}^T d_{k-1}} \tag{19}$$

Unfortunately, even with exact line search conjugate gradient methods fail to generate descent directions. Therefore, the use of a restarts are employed in order to guarantee convergence. A more sophisticated and popular restarting criterion has been introduced by Birgin and Martínez [5] which consists of testing if the angle between the current direction and the gradient is not acute enough, namely

$$d_k^T g_k \leq 10^{-3} \|d_k\|_2 \|g_k\|_2$$

In the literature there have been proposed several restarting criteria [7], [31], [36] with various performances.

III. Descent Conjugate Gradient Methods

In more recent works, Hager and Zhang [15], [16] proposed a modification of the Hestenes-Stiefel formula Eq. (4) by adding the term $-\mu g_k^T d_{k-1}$ with $\mu = 2\|y_{k-1}\|^2/(y_{k-1}^T d_{k-1})^2$ which can guarantee the sufficient descent property Eq. (2) independent of the accuracy of the line search. In particular, they proposed the following choice for the update parameter.

$$\beta^{HZ} = \beta^{HS} - \frac{2\|y_{k-1}\|^2}{(y_{k-1}^T d_{k-1})^2} g_k^T d_{k-1}$$

Later, Yu and Guan [38] motivated by their work proposed the class of descent conjugate gradient methods as follows

$$\beta_k^{DHS} = \beta_k^{HS} - \frac{C\|y_{k-1}\|^2}{(y_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \tag{20}$$

$$\beta_k^{DFR} = \beta_k^{FR} - \frac{C\|g_k\|^2}{\|g_{k-1}\|^4} g_k^T d_{k-1} \tag{21}$$

$$\beta_k^{DPR} = \beta_k^{PR} - \frac{C\|y_{k-1}\|^2}{\|g_{k-1}\|^4} g_k^T d_{k-1} \tag{22}$$

$$\beta_k^{DCD} = \beta_k^{CD} - \frac{C\|g_k\|^2}{(g_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \tag{23}$$

$$\beta_k^{DLS} = \beta_k^{LS} - \frac{C\|y_{k-1}\|^2}{(g_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \tag{24}$$

$$\beta_k^{DDY} = \beta_k^{DY} - \frac{C\|g_k\|^2}{(y_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \tag{25}$$

$$\beta_k^{DP} = \beta_k^{P} - \frac{C\|y_{k-1} - s_{k-1}\|^2}{(y_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \tag{26}$$

Parameter C essentially controls the relative weight between conjugacy and descent and in case $C \geq 1/4$ then the sufficient descent property is satisfied for all the above formulas.

Along this line, Yu et al. [39] introduced a new class of spectral conjugate gradient methods which ensure sufficient descent using any line search. In particular they embedded the spectral gradient [4] in the descent conjugate gradient framework.

$$\beta_k^{DSHS} = \beta_k^{SHS} - \frac{C\|y_{k-1}\|^2}{\delta_k (y_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \quad (27)$$

$$\beta_k^{DSFR} = \beta_k^{SFR} - \frac{C\delta_{k-1}^2 \|g_k\|^2}{\delta_k \|g_{k-1}\|^4} g_k^T d_{k-1} \quad (28)$$

$$\beta_k^{DSPR} = \beta_k^{SPR} - \frac{C\delta_{k-1}^2 \|y_{k-1}\|^2}{\delta_k \|g_{k-1}\|^4} g_k^T d_{k-1} \quad (29)$$

$$\beta_k^{DSCD} = \beta_k^{SCD} - \frac{C\delta_{k-1}^2 \|g_k\|^2}{\delta_k (g_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \quad (30)$$

$$\beta_k^{DSLS} = \beta_k^{SLS} - \frac{C\delta_{k-1}^2 \|y_{k-1}\|^2}{\delta_k (g_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \quad (31)$$

$$\beta_k^{DSDY} = \beta_k^{SDY} - \frac{C\|g_k\|^2}{\delta_k (y_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \quad (32)$$

$$\beta_k^{DSP} = \beta_k^{SP} - \frac{C\|y_{k-1} - s_{k-1}\|^2}{\delta_k (y_{k-1}^T d_{k-1})^2} g_k^T d_{k-1} \quad (33)$$

Obviously, in case $\delta_k = \delta_{k-1} = 1$ then the above formulas will reduce to the corresponding descent conjugate gradient formulas. Subsequently, we will present two truncated schemes for the descent spectral Perry's conjugate gradient method as it was previously presented in [39].

In order to ensure global convergence of the conjugate gradient methods for general nonconvex functions, it has been proposed to truncate β_k^{DSP} by restricting the lower bound of the update parameter of being nonnegative [12]. However by restricting β_k^{DSP} to be nonnegative, results the iterates may differ significantly from those of Eq. (3)-(33) and the convergence speed may be reduced. Thus, Yu et al. [39] proposed two modified schemes in which the lower bound on β_k is dynamically adjusted, in order to make the lower bound small as the iterates converge. In particular, they proposed two variants for selecting the truncated update parameter β^{DSP^+}. The first variant of the truncated scheme uses the update parameter

$$\beta_k^{DSP_1^+} = \max\left\{\beta_k^{DSP}, -\frac{C\|y_{k-1} - \delta_k s_{k-1}\|^2}{\delta (y_{k-1}^T d_{k-1})^2} |g_k^T s_{k-1}|\right\} \quad (34)$$

while the second one is based on a scheme that was first introduced by Hager and Zhang [15], namely

$$\beta_k^{DSP_2^+} = \max\left\{\beta_k^{DSP}, -\frac{1}{\delta_k \|d_{k-1}\| \min\{\xi, \|g_{k-1}\|\}}\right\} \quad (35)$$

where ξ is some positive constant. Notice that when $\|g_{k-1}\|$ tends to zero as k grows, ξ_k tends to $-\infty$ when $\|d_{k-1}\|$ is bounded [15] . In that case $\beta_k^{DSP_2^+}$ will reduce to the descent spectral Perry formula (33).

At this point, we present a high level description of our proposed algorithm for neural network training.

Step 1. Initiate w^0, $0 < \sigma_1 < \sigma_2 < 1$, Err and $\epsilon \to 0$; set $k = 0$.

Step 2. If $(E(w) < Err)$ or $(\|\nabla E(w_k)\|_2 < \epsilon)$ terminate.

Step 3. Compute the descent direction

$$d_k = \begin{cases} -\nabla E(w_k), & \text{if } k = 0; \\ -\frac{1}{\delta_k}\nabla E(w_k) + \beta_k d_{k-1}, & \text{otherwise} \end{cases}$$

where the update parameter is defined by one of (33), (34) or (35).

Step 4. Calculate the learning rate η_k using

$$\eta_k = \begin{cases} \dfrac{1}{\|g_k\|}, & k = 0; \\ \eta_{k-1}\dfrac{\|d_{k-1}\|}{\|d_k\|}, & \text{otherwise} \end{cases}$$

Step 5. Find a step length η_k using the following line search. For $0 < \sigma_1 < \sigma_2 < 1$ at each iteration, choose the step length satisfying the strong Wolfe line search conditions

$$\begin{aligned} E(w_k + \eta_k d_k) - E(w_k) &\leq \sigma_1 \eta_k \langle \nabla E(w_k), d_k \rangle \\ |\langle \nabla E(w_k + \eta_k d_k), d_k \rangle| &\leq -\sigma_2 \langle \nabla E(w_k), d_k \rangle \end{aligned}$$

Step 6. Update the weights

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

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

Remarks: In Step 3, we have used Perry's update parameters which provide us the best overall results. Additionally, since d_k is always a descent direction we avoid the use of a restarting criterion. In Step 4, the learning rate is adaptated using the choice of Shanno and Phua in CONMIN [36] which exploits the conjugate gradient values and the learning rate from the previous epoch.

IV. Experimental Results

In the following section we will present experimental results in order to evaluate the performance of the proposed conjugate gradient algorithm in five famous benchmarks acquired by the UCI Repository of Machine Learning Databases [25]. Subsequently, we briefly describe each problem and the performance comparison between: the descent spectral Perry's conjugate gradient (DSP) with the spectral Perry's conjugate gradient (SP) of Birgin and Martínez [5]. Moreover we test the numerical performance of two different truncation strategies for the update parameter β_k. So we evaluate the two versions of the truncate spectral Perry conjugate gradient (DSP^+): the first one (DSP_1^+) uses the update parameter defined in Eq. (34) and the other one (DSP_2^+) uses the update parameter defined in Eq. (35).

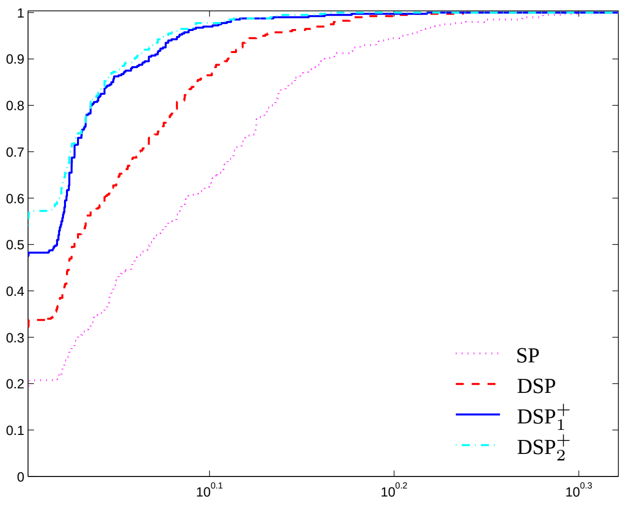
However, the cumulative total for each performance metric over all simulations does not seem to be too informative, since a small number of simulations can tend to dominate these results. For this reason, we use the performance profiles proposed by Dolan and Morè [9] to present perhaps the most complete information in terms of robustness, efficiency and solution quality. The use of profiles present a descriptive measure providing a wealth of information such as solver efficiency, robustness and probability of success in compact form. We have used the Libopt environment [13] for measuring the efficiency and the robustness of our algorithm in terms of CPU time and function/gradient evaluations.

For all algorithms the heuristic parameters were set as $\sigma_1 = 10^{-4}$, $\sigma_2 = 0.5$, $C = 0.5$ and $\xi = 0.01$ for all experiments as in [38]. All networks have received the same sequence of input patterns and the initial weights were initiated using the Nguyen-Widrow method [26]. For evaluating classification accuracy we have used the standard procedure called *k-fold cross-validation* [21]. The implementation has been carried out using Matlab version 7, based on the SCG code of Birgin and Martínez [5] obtained from the web page: *http://www.ime.usp.br/∼egbirgin/*. All simulations have been carried out on a processor Pentium-IV computer (3.2MHz, 1Gbyte RAM) running Linux operating system.

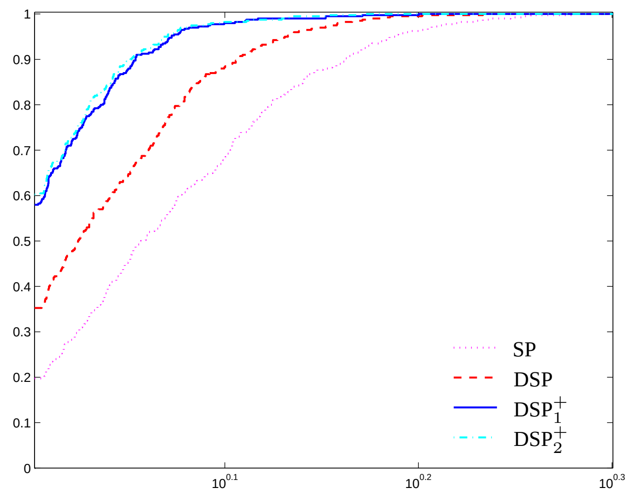
A. SPECT Heart Data Set

This data set contains data instances derived from cardiac Single Proton Emission Computed Tomography (SPECT) images from the University of Colorado [25]. 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 and an output layer of 1 neuron [3], [40]. The termination criterion is set to $E \leq 0.1$ within the limit of 1000 epochs.

In Figure 1 are presented the performance profiles for the SPECT heart problem. Relative to both performance metrics, DSP significantly outperforms SP method, since the curve of the former lies above the curve of later. Furthermore, DSP_1^+ and DSP_2^+ exhibit the highest probability of being the optimal solver with DSP_2^+ exhibiting slightly better performance.



(a) Performance based on CPU time



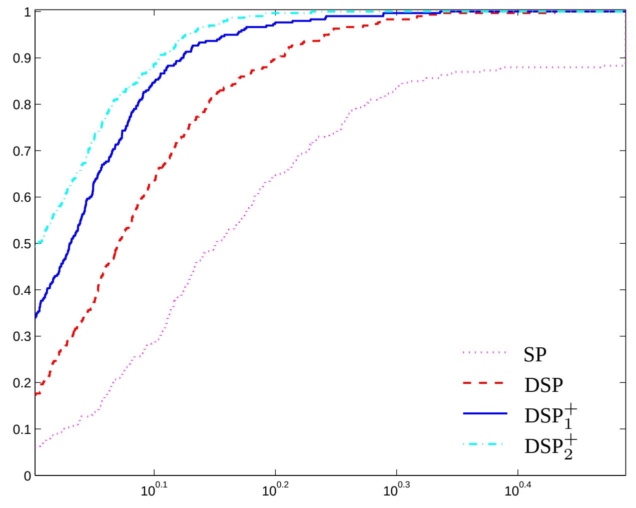
(b) Performance based on function/gradient evaluations

Fig. 1. Log_{10} scaled performance profiles for CPU time and number of function/gradient evaluations for the spect heart problem.

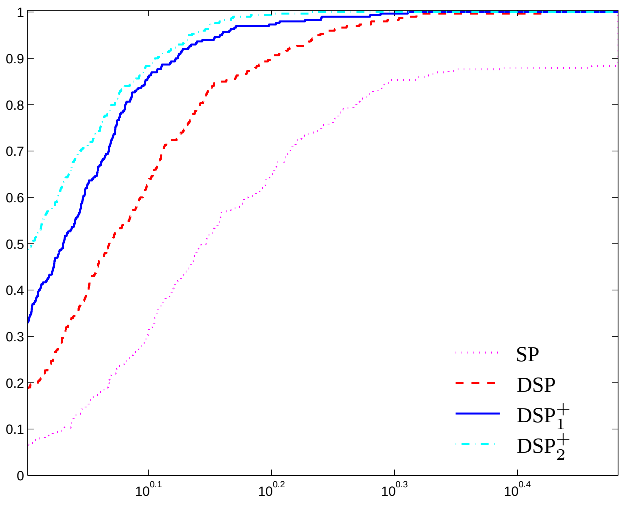
B. Iris Data Set

This benchmark is perhaps the most best known to be found in the pattern recognition literature [25]. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The network architectures constitute of 1 hidden layer with 7 neurons and an output layer of 3 neurons. The termination criterion is set to $E \leq 0.01$ within the limit of 1000 epochs and all networks were tested using 10-fold cross-validation.

Figure 2 presents the performance profiles for the iris problem investigating the performance of each training method. DSP is much more robust and efficient than SP in terms of both performance metrics. Moreover, DSP_2^+ exhibits the best performance, slightly outperforming DSP_1^+.

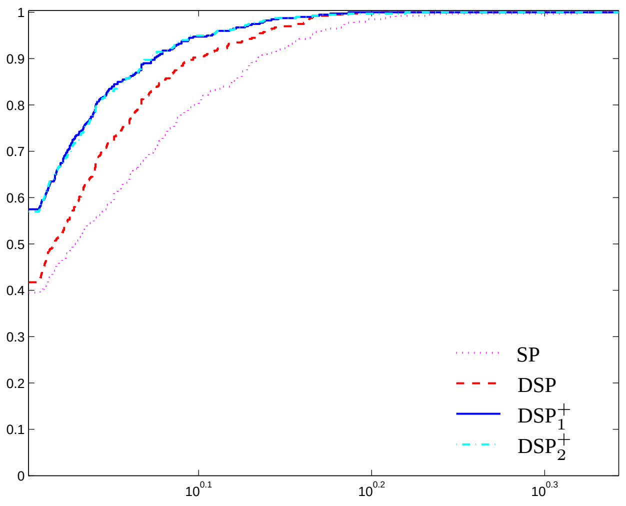


(a) Performance based on CPU time



(b) Performance based on function/gradient evaluations

Fig. 2. Log_{10} scaled performance profiles for CPU time and number of function/gradient evaluations for the iris problem.



(b) Performance based on function/gradient evaluations

Fig. 3. Log_{10} scaled performance profiles for CPU time and number of function/gradient evaluations for the Australian credit approval problem.

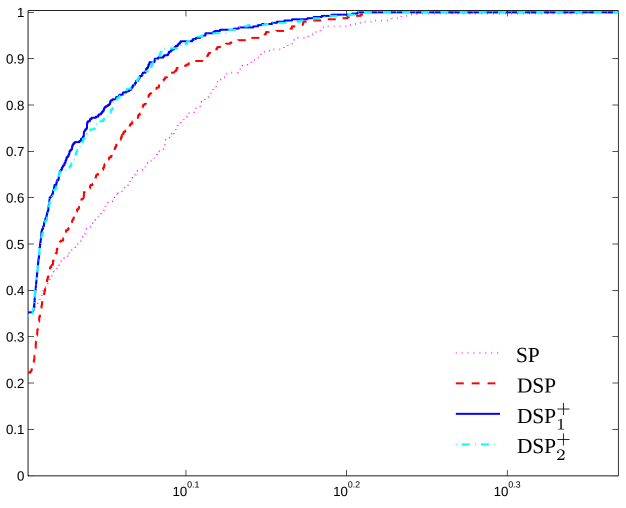
C. *Australian Credit Approval Data Set*

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 layer with 16 and 8 neurons and an output layer of 2 neurons [34]. 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.

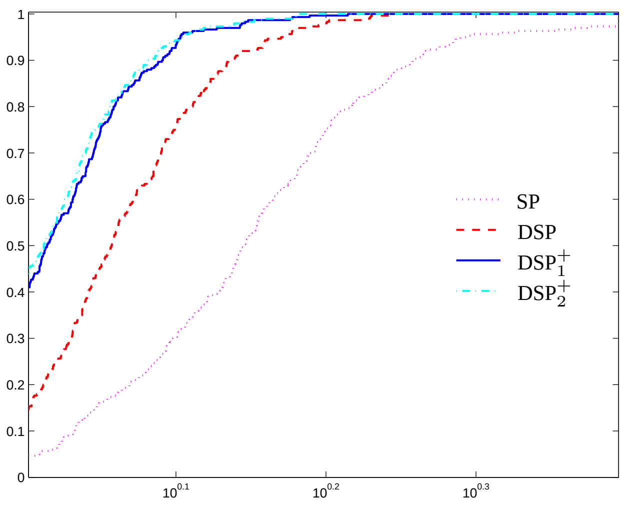
Figure 3 presents the performance profiles for the Australian credit approval problem evaluating the efficiency and robustness of each training method. Regarding all performance metrics DSP_1^+ and DSP_2^+ exhibit the best performance. Moreover, the interpretation in Figure 3 implies that the sufficient descent property led to a significant improvement of DSP, DSP_1^+ and DSP_2^+ over the SP method since the curves of the former lie above the curve of the latter.

D. *Escherichia coli Data Set*

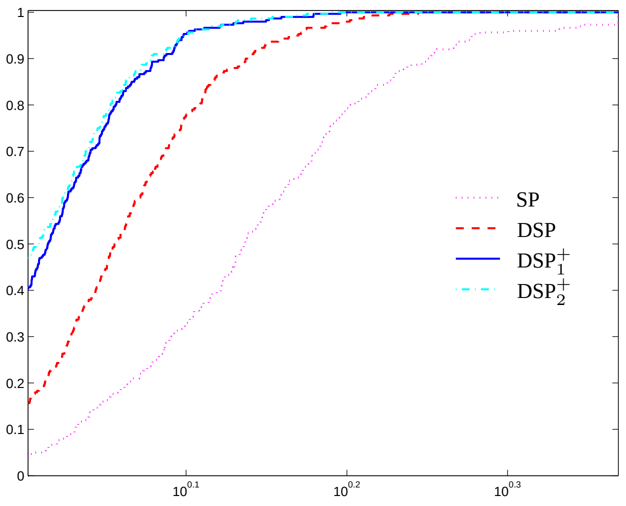
This problem is based on a drastically imbalanced data set of 336 patterns and concerns the classification of the *E. coli* protein localization patterns into eight localization sites. E. coli, being a prokaryotic gram-negative bacterium, is an important component of the biosphere. Three major and distinctive types of proteins are characterized in E. coli: enzymes, transporters and egulators. The largest number of genes encodes enzymes (34%) (this should include all the cytoplasm proteins) followed by the genes for transport functions and the genes for regulatory process (11.5%) [22]. The network architectures constitute of 1 hidden layer with 16 neurons and an output layer of 8 neurons [1]. The termination criterion is



(a) Performance based on CPU time



(a) Performance based on CPU time



(b) Performance based on function/gradient evaluations

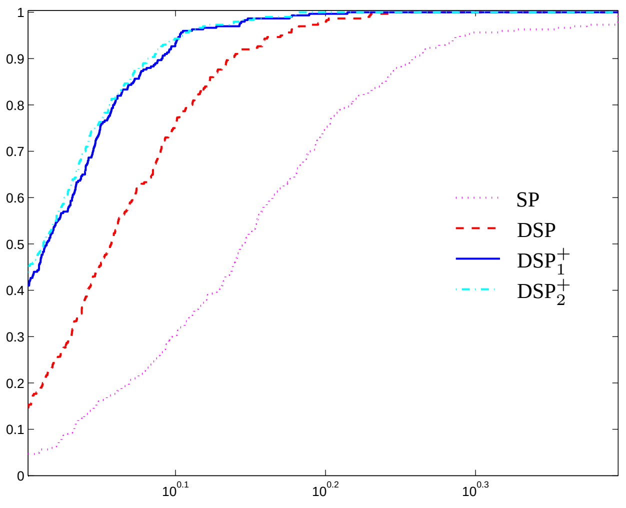
Fig. 4. Log_{10} scaled performance profiles for CPU time and number of function/gradient evaluations for the escherichia coli problem.

set to $E \leq 0.02$ within the limit of 2000 epochs and all neural networks were tested using 4-fold cross-validation [20].

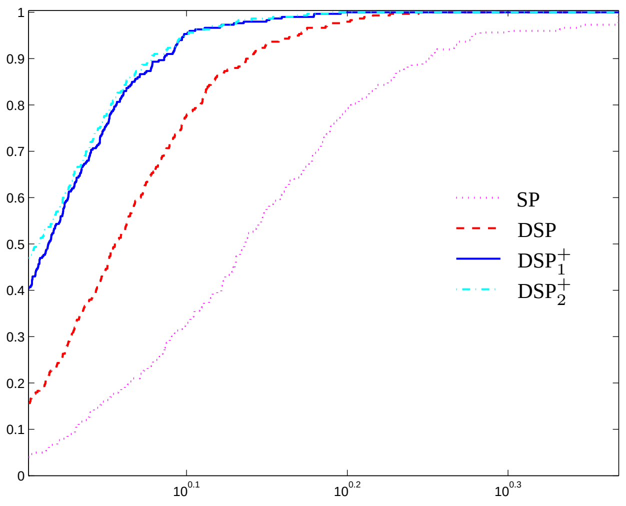
In Figure 4 are presented the performance profiles for both performance metrics for the escherichia coli problem. DSP is much more efficient and robust than SP in terms of CPU time and number of function/gradient evaluations. Furthermore, DSP_1^+ and DSP_2^+ perform similarly and exhibit the best performance since they present the highest probability of being the optimal solver regarding all performance metrics.

E. Yeast Data Set

This problem is similar with the Escherichia coli problem, namely the determination of the cellular localization of the yeast proteins [19]. 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



(a) Performance based on CPU time



(b) Performance based on function/gradient evaluations

Fig. 5. Log_{10} scaled performance profiles for CPU time and number of function/gradient evaluations for the yeast problem.

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 [1].

Figure 5 illustrates the performance profiles of each training method for the yeast problem, investigating the efficiency and robustness of each method. Similar observations can be made with the previous problems. More specifically, DSP significantly outperforms SP, exhibiting better performance regarding both performance metrics while DSP_1^+ and DSP_2^+ exhibit the best performance, reporting the highest probability of being the optimal solver.

V. Conclusions

In this work, we evaluate the performance of a new conjugate gradient algorithm for training neural networks. From the rigorous analysis of the simulation results is strongly demonstrated that the new proposed method overwhelmingly outperforms classical conjugate gradient methods for neural network training, proving more stable, efficient and reliable learning. Moreover, we have explored the performance of two different truncation strategies for the update parameter β_k. We conclude that the truncation may be needed not only to ensure convergence but also from the practical point of viewpoint truncation impedes the convergence of the conjugate gradient methods, as it was also pointed out by Powell [32].

Our future work is concentrated on investigating the effect of incorporating them a nonmonotone strategy [14] and exploring different choices of the coefficient δ_k. Moreover an interesting idea is to perform a large scale study on several conjugate gradient algorithms for neural network training including the SCALCG of Andrei [2] and the CG Descent of Hager and Zhang [15].

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] N. Andrei. Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. *Optimization Methods and Software*, 22:561–571, 2007.

[3] M.S. Apostolopoulou, D.G. Sotiropoulos, I.E. Livieris, and P. Pintelas. A memoryless BFGS neural network training algorithm. In 7^{th} *IEEE International Conference on Industrial Informatics (INDIN'09)*, 2009, to be presented.

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

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

[6] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1995.

[7] A. Buckley and A. LeNir. QN-like variable storage conjugate gradients. *Mathematical Programming*, 27:155–175, 1983.

[8] Y.H. Dai and Y.X. Yuan. A nonlinear conjugate gradient with a strong global convergence properties. *SIAM Journal of Optimization*, 10:177–182, 2000.

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

[10] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, New York, 2nd edition, 1987.

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

[12] J.C. Gilbert and J. Nocedal. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal of Optimization*, 2(1):21–42, 1992.

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

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

[15] W.W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM of Journal Optimization*, 16:170–192, 2005.

[16] W.W. Hager and H. Zhang. CG DESCENT: A conjugate gradient method with guarantee descent and an efficient line search. *ACM Transactions on Mathematical Software*, 32:113–137, 2006.

[17] W.W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific of Journal Optimization*, 2:35–58, 2006.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] P. Liang, B. Labedan, and M. Riley. Physiological genomics of escherichia coli protein families. *Physiological Genomics*, 9:15–26, 2002.

[23] Y. Liu and C. Storey. Efficient generalized conjugate gradient algorithms, part 1: theory. *Journal of Optimization Theory and Applications*, 69:129–137, 1991.

[24] I.E. Livieris and P. Pintelas. A spectral conjugate gradient method with sufficient descent property for neural network training. Technical Report TR08-03, Department of Mathematics, University of Patras, 2008.

[25] 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.

[26] 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.

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

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

[29] A. Perry. A modified conjugate gradient algorithm. *Operational Research*, 26:1073–1078, 1978.

[30] 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.

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

[32] M.J.D. Powell. Nonconvex minimization calculations and the conjugate gradient method. In *Lectures Notes in Mathematics*, volume 1066, pages 122–141. Springer Verlag, Berlin, 1984.

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

[34] 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.

[35] 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.

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

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

[38] G. Yu and L. Guan. New descent nonlinear conjugate gradient methods for large scale unconstrained optimization. Technical report, Department of Scientific Computation and Computer Application, Sun Yat-Sen University, Guangzhou, P.R. China, 2005.

[39] G. Yu, L. Guan, and W. Chen. Spectral conjugate gradient methods with sufficient descent property for large-scale unconstrained optimization. *Optimization Methods and Software*, 23(2):275–293, 2008.

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