# TRAINING A NEURAL NETWORK WITH CONJUGATE GRADIENT METHODS

Michael Towsey[1], Dogan Alpsan[1] and Laszlo Sztriha[2].
1. Dept of Biophysics, United Arab Emirates University, PO Box 17666, Al Ain, UAE.
2. Dept of Paediatrics, United Arab Emirates University, PO Box 17666, Al Ain, UAE.

*ABSTRACT*

*This study investigates the use of several variants of conjugate gradient (CG) optimisation and line search methods to accelerate the convergence of an MLP neural network learning two medical signal classification problems. Much of the previous work has been done with artificial problems which have little relevance to real world problems and results on real world problems have been variable. The effectiveness of CG compared to standard backpropagation (BP) depended on the degree to which the learning task required finding a global minimum. If learning was stopped when the training set had been learned to an acceptable degree of error tolerance (the typical pattern classification problem), standard BP was faster than CG and did not display the convergence difficulties usually attributed to it. If learning required finding a global minimum (as in function minimisation or function estimation tasks), CG methods were faster but performance was very much dependant on careful selection of 'tuning' parameters and line search. This requirement for meta-optimisation was more difficult for CG than for BP because of the larger number of parameters.*

## 1. Introduction

Standard back-propagation (BP) remains a popular supervised learning algorithm to train feed-forward neural networks. However, it is a simple first order gradient descent method with poor convergence properties. There have been a number of recent reports describing the use of second order optimisation methods, such as conjugate gradient (CG), to accelerate the convergence of BP, but the results have been mixed. For example, in comparing percentage of trials that converged on a global minimum, Fletcher-Reeves CG was not as good as standard BP on the XOR task but was better than standard BP on two function estimation tasks [14]. Aylward et al [5] report faster learning by CG on a function estimation problem but slower learning (compared to standard BP) on a task involving classification of handwritten numerals.

In this study, we demonstrate that the effectiveness of CG compared to standard BP depends on the degree to which the task demands converging on a global minimum. For a pattern classification problem which does not require finding the exact global minimum of the error function, standard BP may well be faster than CG and have good generalisation. However if the exact global minimum must be found, as in function estimation problems, then CG methods are much superior, although attaining optimum performance requires fine tuning of parameters just as does standard BP. This study has the added interest that the comparisons are performed on two real-world medical tasks. Performance comparisons on artificial tasks such as XOR and higher order parity problems have doubtful relevance to real world problems.

## 2. Conjuate Gradient and its Variants

There are a wide variety of CG methods from which to choose. They differ in their computation of both step direction and stepsize. Traditional CG employs either the Fletcher-Reeves or Polak-Ribiere step direction, with restarts every $n$ iterations (where $n$ = number of weights in the network) and exact line searches. For a full exposition of CG, see Gill et al. [9] and Shanno [13]. As stated above, results with traditional CG in neural network applications have been mixed. Attempts to improve the efficiency of CG include altered restart conditions [8, 11] and inexact line searches. Use of Beale's algorithm with Powell restarts showed promising results in training neural nets on function estimation problems [14].

A class of methods known as memoryless BFGS computes the step direction using a formula derived from quasi-Newton algorithms but without storing the Hessian explicitly. These methods work well with inexact line searches, thus reducing computational burden. Shanno reported good results with three scaled versions of BFGS on function minimisation problems [13]. When applied to a neural net pattern classification problem, memoryless BFGS without scaling or restarts required four times more iterations than traditional CG but total computation time was less [7]. The performances of standard CG and BFGS were similar on parity problems of parity greater than 5 [10].

Exact line searches are computationally expensive and indeed unnecessary where the objective function is not a quadratic. Inexact line searches can be

0-7803-2768-3/95/$4.00 © 1995 IEEE

achieved simply by terminating the search when the minimum is located within a larger interval of uncertainty. However it is more usual to apply one or both of the following tests before terminating a line search [9];

$$f_k - f_{k+1} \geq -\mu \, \alpha_k \, g_k \, d_k \qquad (1)$$

and $$|d_k \, g_{k+1}| \leq -\eta \, g_k \, d_k \qquad (2)$$

where $f_k - f_{k+1}$ is the change in the error function on the $k$+1th step, g the gradient vector, d the direction vector and $\alpha$ the stepsize. Eqn (1) ensures a sufficient decrease in the value of the objective function and eqn (2) ensures not too small a stepsize by requiring sufficient decrease in the directional gradient. $\mu$ and $\eta$ are tuning parameters whose values, in practice, have a big effect on learning performance. Gill et al [9] suggest an alternative to eqn (2) which requires only function evaluations rather than the more time consuming gradient calculations;

$$| f(w_k+\alpha d_k) - f(w_k+v d_k) |/(\alpha-v) \leq -\eta \, g_k \, d_k \quad (3)$$

where $0 \leq v < \alpha$.

Since most of the computational burden in CG involves line search, Møller [10] introduced an algorithm which computes the stepsize analytically. His method has the attractive feature of having no arbitrary tuning parameters whose values are critical to performance. Møller reports it to be twice as fast as traditional CG and memoryless BFGS on parity problems.

## 3. Methods

### 3.1. Task 1

Task 1 was to classify Brainstem Auditory Evoked Potential (BAEP) signals into one of two classes, corresponding to the presence or absence of a sound evoked waveform following auditory stimulus. The raw BAEP's were processed as described in [1] to give temporal input signals of 50 values normalised between 0 and 1. A training set of 60 signals and test set of 261 signals were prepared as described in [2].

### 3.2. Task 2

Task 2 was to classify single channel segments of EEG into two classes, according to the presence or absence of artifact. This is part of a longer term study to automate the process of artifact avoidance for quantitative EEG. Thirty epochs (each of 1s duration, 19 channels and 256 Hz sampling) containing a representative selection of acceptable and artifactual EEG were selected from each of ten recordings from ten children aged 7 to 11 years (5 normal, 5 diagnosed with hemiplegia). A further sub-selection of 1103 signals was made from the Fp1, Fz, T3 and C3 channels of the above 300 epochs. Artifactual signals were restricted to those having artifacts of at least 0.2 s duration or longer. A training set of 100 signals was randomly selected and the remaining 1003 signals were used as a test set. In this task, the input vector consisted of signal features, not the temporal signal. Signals features included the mean, the range and selected spectral components giving an input with 15 components between 0 and 1.

### 3.3. The Net and Training

A 3 layer feed-forward net with 2 output units representing the two classes was used for both tasks. For task 1, the net had 50 input units and 6 hidden layer units. For task 2, the net had 15 input units and 8 hidden layer units. These hidden layer sizes were the minimum required to give good performance on each task. Other conditions were as in [3] except that error was backpropagated for all signals, learned or not, at every iteration. Weights were updated in batch mode. Each experiment (combination of algorithm and tuning parameters) was repeated 10 times and used to calculate average performance indices.

Two training termination conditions were compared using the same set of tuning parameters for each; when *all* training signals had been learned within an error tolerance of 0.2 *for each output unit* and when every element of the gradient vector had an absolute value less than $10^{-5}$. In either case, training was terminated if the number of iterations exceeded 10,000 and average performance indices were calculated using only those trials which learned the training set or converged on a global minimum. It should be noted that in this problem there was no single global minimum but rather a large number of possibilities to reduce the value of the error function to zero.

### 3.4. Algorithms

We investigated seven CG algorithms for this study. Abbreviations (in parentheses) are added for ease of later discussion. CG algorithms are described in the literature using somewhat different names and notations. To avoid ambiguity, we refer to the equation numbers in Shanno [13] wherever applicable.

1) Polak-Ribiere version (Shanno eqn (7)) of traditional CG. (PR-CG)

2) Beales's CG (Shanno eqn (28). (B-CG)

3) Memoryless BFGS without scaling, using Shanno's eqn (20). (BFGS-CG)

4) Memoryless BFGS with Fletcher's scaling, using Shanno's eqns (20) and (45). (F-BFGS-CG)

5) Memoryless BFGS with Shanno's scaling, using Shanno's eqn (26). (S-BFGS-CG)

6) Memoryless BFGS with Shanno's scaling, using Shanno's eqns (26), (34), (38) and (39) and including restarts when Powell's inequality condition [14], $|g_{k+1} g_k| \geq 0.2 \, ||g_{k+1}||^2$, is true. (SH-CG)

7) Møller's auto-scaling CG algorithm [10] without line searches. (MLS-CG)

### 3.5. Line Search

The speed of CG methods depends critically on line search efficiency. Unfortunately the best search algorithm appears to be task dependent [5]. We experimented with library versions [12] of two line minimisation algorithms, Golden Section and Brent's method which combines quadratic interpolation (using function evaluations only) with golden section. Since Brent's method proved to be 2 to 3 times faster, we report results using it alone.

The above seven algorithms were combined with different conditions to terminate line search. Exact line searches (ELS) were terminated when the minimum had been located to a fractional tolerance of $10^{-6}$. Inexact line searches (ILS) were terminated as follows;
(ILS1):- terminated when minimum located to a fractional tolerance of 0.05.
(ILS2):- terminated when eqn (1) true.
(ILS3):- terminated when eqns (1) and (2) both true.
(ILS4):- terminated when eqns (1) and (3) both true.

In addition to the constants in eqns (1) to (3), another important tuning parameter in the line search is the initial value of $\alpha$ to initialise bracketing. We attempted to find the best value for this parameter for each combination of CG and line search.

## 4. Results

### 4.1. Learning Speed

For both tasks, maximum learning speed of standard BP was obtained with high momentum. Although it could quickly learn the training set, standard BP could not find a global minimum in less than 10,000 iterations. In separate trials using the same parameter values as in Table 1, standard BP without momentum required an average of 170,555 iterations to converge on a global minimum in task 1.

The results for CG in Tables 1 & 2 represent the best in terms of speed from hundreds of trials. For both tasks, the shortest time to learn the training set was less than for standard BP without momentum, but no algorithm was faster than standard BP with momentum. Speed differences between algorithms became more apparent when tested for their ability to converge on a global minimum. Fastest convergence time on both tasks was achieved by SH-CG with ILS1 and ILS2, although the speed was not faster than standard PR-CG with simple inexact line searches (ILS1). The least satisfactory algorithms, both in speed and stability were unscaled BFGS-CG and Fletcher scaled F-BFGS-CG, both of which use Shanno's eqn (20) to determine step direction.

### 4.2. Generalisation

For task 1, the best generalisation was achieved by standard BP with large stepsize (0.4) and zero momentum. Use of second order information to modify the direction of steepest descent (using momentum or CG directions) resulted in poor generalisation. In the case of task 2, all algorithms yielded 78 - 79% generalisation. We did not detect 'overlearning' as the network searched for a global minimum after learning the training set. On the contrary, most algorithms yielded slightly better generalisation after converging on a global minimum.

### 4.3. Line Search

As expected, the use of exact searches slowed learning by increasing computational complexity. However the simple expedient of increasing the bracketing uncertainty (ILS1) yielded greatest increase in speed. ILS2 had mixed results, reducing the number of line search iterations but increasing the number of CG iterations. The net effect on learning speed varied with the algorithm. ILS3 increased computational complexity and therefore training times, without necessarily reducing CG iterations. These trends are clearly seen in PR-CG and SH-CG. ILS4 reduced line search computation in some cases and not in others and had a variable effect on the number of CG iterations.

Møller's self-scaling algorithm was somewhat slower than most other modified CG methods, despite not having to perform line searches. In task 2, it showed a tendancy to become trapped in local minima.

## 5. Discussion

Our results suggest that speed comparisons between standard BP and CG depend on the degree to which the task requires closing in on a global minimum. If training is stopped when the training set is learned, the ability of an algorithm to converge on a minimum is not tested, particularly when the error tolerance is set to a high level (0.2) as in this study and others. This would account for the task dependence of relative learning speeds reported in some studies [5, 14]. Note that a pattern classification task becomes a function minimisation task when the error tolerance is set to a small value approaching zero.

For both of our classification tasks, none of the CG methods could learn the training set in a shorter time than standard BP with momentum. Their reduced number of iterations was more than offset by greater computational complexity.

**TABLE 1 - Comparative performance of optimisation algorithms on BAEP task (task 1).**

Performance values represent the mean and SD of converged simulations. Line Search:- For meaning of codes see METHODS. No. con:- number of simulations converged within 10,000 iterations, out of 10 trials differing in the initial random weights. Time:- average CPU time (minutes) required to reach the convergence condition. Gen:- generalisation, % of test set correctly classified at convergence. *m=0.0, m=0.9* refer to momentum value.

| | | Convergence when training set learned | | | | Convergence when gradient = zero | | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | Line Search | No con | Time (min) | Iterations | % Gen | No con | Time (min) | Iterations | % Gen |
| stdBP m=*0.0* | - | 10 | 1.0 | 216 ±26 | 81.5±1.4 | 0 | - | 10,000 | - |
| stdBP *m=0.9* | - | 10 | 0.5 | 107 ±11 | 77.9±1.9 | 0 | - | 10,000 | - |
| PR-CG | ELS | 9 | 1.0 | 33 ± 8 | 75.6±2.1 | 9 | 1.9 | 58 ±15 | 77.2 ±2.5 |
| | ILS1 | 10 | 0.8 | 31 ± 7 | 76.7±1.0 | 10 | 1.5 | 61 ±19 | 78.7 ±1.4 |
| | ILS2 | 10 | 0.7 | 40 ± 8 | 77.2±1.8 | 10 | 1.8 | 88 ±17 | 78.0 ±0.9 |
| | ILS3 | 10 | 0.9 | 38 ± 8 | 77.5±1.7 | 10 | 2.2 | 87 ±12 | 79.1 ±1.2 |
| B-CG | ILS3 | 10 | 0.9 | 47 ±15 | 76.3±2.1 | 10 | 1.9 | 87 ±20 | 77.7 ±1.5 |
| BFGS-CG | ILS1 | 9 | 0.7 | 32 ± 6 | 76.9±2.5 | 9 | 1.7 | 65 ±16 | 78.0 ±2.3 |
| | ILS2 | 7 | 0.9 | 45 ±25 | 76.6±1.8 | 7 | 3.7 | 181 ±79 | 76.5 ±1.8 |
| F-BFGS-CG | ILS3 | 10 | 0.8 | 30 ±15 | 77.1±1.6 | 10 | 2.6 | 103 ±40 | 77.9 ±1.7 |
| | ILS4 | 10 | 0.8 | 31 ±15 | 76.9±1.6 | 10 | 2.4 | 97 ±28 | 76.9 ±1.6 |
| S-BFGS-CG | ILS2 | 10 | 0.7 | 36 ± 8 | 77.7±2.0 | 9 | 1.2 | 57 ± 9 | 78.5 ±1.8 |
| | ILS3 | 9 | 0.9 | 29 ± 6 | 77.5±1.5 | 9 | 1.6 | 54 ± 8 | 78.2 ±1.5 |
| SH-CG | ELS | 10 | 1.0 | 31 ±12 | 77.2±1.9 | 10 | 1.7 | 53 ±12 | 76.9 ±2.6 |
| | ILS1 | 10 | 0.7 | 28 ± 6 | 77.5±1.8 | 10 | 1.3 | 52 ± 8 | 78.1 ±1.2 |
| | ILS2 | 10 | 0.7 | 35 ± 7 | 77.0±1.8 | 10 | 1.3 | 65 ±18 | 77.9 ±1.4 |
| | ILS3 | 10 | 0.9 | 34 ± 7 | 77.7±1.0 | 10 | 1.5 | 61 ±11 | 78.9 ±1.4 |
| | ILS4 | 10 | 0.8 | 30 ± 8 | 77.2±1.5 | 10 | 1.5 | 60 ±11 | 77.5 ±1.8 |
| MLS-CG | - | 10 | 1.0 | 46 ± 8 | 77.2±1.2 | 10 | 2.2 | 104 ±10 | 78.4 ±1.7 |

**TABLE 2 - Comparative performance of optimisation algorithms on EEG task (task 2).**
(caption as for Table 1)

| | | Convergence when training set learned | | | | Convergence when gradient = zero | | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | Line Search | No con | Time (min) | Iterations | % Gen | No con | Time (min) | Iterations | % Gen |
| stdBP *m=0.0* | - | 10 | 3.4 | 808 ±116 | 79.0±1.3 | 0 | - | 10,000 | - |
| stdBP *m=0.9* | - | 10 | 1.0 | 239 ± 39 | 78.6±0.5 | 0 | - | 10,000 | - |
| PR-CG | ELS | 8 | 3.3 | 118 ± 63 | 78.4±1.5 | 8 | 5.7 | 199 ±146 | 78.5 ±1.8 |
| | ILS1 | 10 | 2.4 | 115 ± 61 | 77.8±1.4 | 10 | 3.5 | 159 ± 66 | 78.1 ±1.5 |
| | ILS2 | 10 | 2.7 | 160 ± 55 | 78.6±1.6 | 10 | 5.1 | 281 ± 61 | 78.5 ±1.0 |
| | ILS3 | 9 | 2.7 | 98 ± 26 | 77.6±1.0 | 9 | 4.3 | 148 ± 38 | 78.3 ±1.0 |
| B-CG | ILS2 | 9 | 2.5 | 146 ± 51 | 78.0±1.6 | 9 | 4.2 | 229 ± 60 | 78.4 ±1.4 |
| BFGS-CG | ILS1 | 9 | 2.4 | 112 ± 75 | 77.6±1.4 | 9 | 4.5 | 193 ± 81 | 78.4 ±1.1 |
| | ILS2 | 7 | 4.8 | 277 ±402 | 77.8±0.4 | 7 | 13.3 | 725 ±339 | 77.9 ±1.2 |
| F-BFGS-CG | ILS3 | 10 | 5.0 | 154 ±132 | 77.8±2.0 | 10 | 7.5 | 235 ±133 | 77.6 ±1.4 |
| | ILS4 | 8 | 6.9 | 317 ±413 | 77.3±1.9 | 8 | 10.6 | 482 ±413 | 77.0 ±2.3 |
| S-BFGS-CG | ILS2 | 9 | 3.2 | 176 ±184 | 78.4±1.5 | 9 | 4.0 | 111 ±167 | 78.3 ±1.6 |
| | ILS3 | 9 | 3.9 | 118 ± 55 | 77.9±1.7 | 9 | 5.6 | 171 ± 54 | 78.1 ±1.8 |
| SH-CG | ELS | 10 | 2.7 | 94 ± 43 | 78.2±1.4 | 10 | 4.2 | 147 ± 39 | 78.7 ±0.8 |
| | ILS1 | 10 | 2.4 | 106 ± 51 | 78.1±1.6 | 10 | 3.6 | 161 ± 54 | 78.3 ±1.3 |
| | ILS2 | 9 | 2.3 | 124 ± 80 | 78.4±2.0 | 9 | 3.3 | 173 ± 87 | 78.5 ±1.3 |
| | ILS3 | 10 | 3.8 | 94 ± 48 | 78.3±1.4 | 10 | 5.6 | 142 ± 48 | 78.6 ±1.1 |
| | ILS4 | 10 | 2.8 | 115 ± 39 | 78.3±0.9 | 10 | 4.8 | 204 ± 57 | 78.5 ±0.8 |
| MLS-CG | - | 7 | 2.9 | 115 ± 40 | 77.7±1.4 | 7 | 5.3 | 281 ± 58 | 78.1 ±1.0 |

The failure of standard BP to converge on a global minimum within 10,000 iterations is due to its fixed stepsize, which means weight changes become extremely small as the gradient approaches zero. By contrast, CG methods were able to converge quickly to a global minimum.

The performance of CG methods depends upon the formula used for the step direction and the line search algorithm. Numerous formulae for step direction have been proposed which are all roughly equivalent where the objective function approximates a quadratic. Of the many CG methods tested in [13], SH-CG was substantially more efficient than the others on a variety of artificial function minimisation problems. Likewise in our study, SH-CG performed very well on both tasks, while S-BFGS-CG performed very well on task 1. Shanno also reported that use of Powell restarts improved the performance of all the methods he investigated, but noted dramatic improvements when used with SH-CG. In our case, Powell restarts were sometimes useful but the improvement was never dramatic.

Since line-search accounts for most of the computational burden in CG, it requires careful attention. Inexact searches were clearly superior but the more careful line search methods (ILS2 and 3) did not outperform ILS1. Furthermore, the effectiveness of eqns (1) and (2) depended critically on values of $\mu$ and $\eta$. Shanno set $\mu$ to a small value (0.0001) in all cases, so that $\eta$ became the critical tuning parameter. Although in theory, problems can arise if $\mu > \eta$, in our tasks we obtained good results with a larger $\mu$. In short, the use of ILS1 was an effective means to achieve inexact line searches and obviated the need for tuning $\mu$ and $\eta$.

The dependence of generalisation on stepsize and momentum observed in task 1 may be peculiar to the task. We have not observed this phenomenon with artificial problems nor in task 2. Note that in task 1, the training set was expert selected, whereas in task 2 the training set was randomly selected. In a previous study [2], an expert selected training set gave better generalisation than a randomly selected training set. Other reports suggest that the effect of an optimisation algorithm on generalisation is very much task dependent [6, 7]. An interesting conclusion from these studies and ours is that generalisation can vary between algorithms and even between different trials of the same algorithm despite all of them finding a global minimum. That is, better generalisation is not the result of finding a better (lower) minimum.

One cannot ignore the time and difficulty involved in the fine tuning of an optimisation problem (sometimes called *meta-optimisation*). Although standard BP is frequently criticised for having arbitrary tuning parameters, in fact CG methods have a more difficult tuning requirement. Reference has already been made to the best choice of line search algorithm being problem dependent [5]. In addition, the initial choice of $\alpha$ to initialise bracketing and the values assigned to the constants in eqns (1) to (3) require fine tuning. Since in practice, the choice of algorithm could well depend on its ease of implementation, Møller's self-scaling algorithm (MLS-CG) deserves particular attention. Although comparatively slow on our tasks, the lack of any requirement for fine tuning could make it the method of choice for quick implementation in problems demanding convergence to a global minimum.

## 6. Concluding Remarks

The classification of BAEP signals in task 1 is difficult because the signals are frequently contaminated by muscle artifacts. The classification of EEG segments in task 2 proved to be even more difficult due to the highly variable nature of acceptable EEG, as well as the wide variety and subtle nature of possible artifacts. Even human experts have difficulty with these tasks as evidenced by the high percentage (approximately 20%) of disagreement between them [1]. Nevertheless these tasks are typical biomedical signal classification problems and appropriate ones with which to compare standard BP and CG optimising algorithms. The results suggest that for pattern classification tasks with moderate error tolerance, standard BP may well be good enough and indeed better than more sophisticated second order methods. The convergence problems frequently cited for back-propagation only became apparent in our study when the nature of the task was changed to one of finding a global minimum as in a function minimisation or function estimation task. Even when it is necessary to find the global minimum, we have found that first order adaptive stepsize algorithms can be faster than the best of conjugate gradient [4].

With regard to the requirement for meta-optimisation, standard BP was easier to tune than the various CG algorithms because it had only two tuning parameters. However the lack of any requirement for fine tuning of Møller's self-scaling algorithm more than compensated for its slower learning speed and make it the method of choice for fast implementation.

To conclude, conjugate gradient optimisation methods do not offer an easy or automatic solution to the training of feed-forward neural networks but they can greatly improve performance when thoughtfully applied.

## 7. References

[1] D. Alpsan and O. Ozdamar, "Auditory brainstem evoked potential classification for threshold detection by neural networks. I. Network design, similarities between human-expert and network classification, feasibility", *Automedica* **15:** 67-82, 1992a.

[2] D. Alpsan and O. Ozdamar, "Auditory brainstem evoked potential classification for threshold detection by neural networks. II. Effects of input coding, training set size and composition and network size on performance", *Automedica* **15:** 83-93, 1992b.

[3] D. Alpsan, M. Towsey, O. Ozdamar, A. Tsoi and D. Ghista, "Determining hearing threshold from brain stem evoked potentials - Optimising a neural network to improve classification performance", *IEEE Eng in Med and Biol Magazine* **13:** 465-471, 1994.

[4] A. Alpsan, M. Towsey, O. Ozdamar, A. Tsoi and D. Ghista, 'Efficacy of Modified Backpropagation and Optimisation Methods on a Real-World Medical Problem", *Neural Networks* In press, 1995.

[5] S. Aylward, D. St.Clair, W. Bond, B. Flachsbart and A. Rigler, "One-dimensional search strategies for conjugate gradient training of backpropagation neural networks", *Proc Artificial Neural Networks in Eng (ANNIE '92) Conf. St. Louis, MO*, **2:** 197-202, 1992.

[6] E. Barnard and J. Holm, "Optimisation for training neural nets', *Neurocomputing* **6:** 19-30, 1994.

[7] R. Battiti and G. Tecchiolli, 'Learning with first, second and no derivatives: A case study in high energy physics", *Neuro-computing* **6:** 181-206, 1994.

[8] E. Beale, in F.A. Lootsma (Ed.), *Numerical Methods for Nonlinear Optimization*, 39-43, Academic Press, London, UK, 1972.

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

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

[11] M. Powell, 'Restart procedures for the conjugate gradient method", *Math Programming,* **12:**, 241-254, 1977.

[12] W. Press, B. Flannery, S. Teukolsky and W. Vetterling, *Numerical Recipes in Pascal - the art of scientific computing* Cambridge University Press., Cambridge, UK, 1989.

[13] D. Shanno, "Conjugate gradient methods with inexact searches", *Mathematics of Operations Research* **3:** 244-256, 1978.

[14] P. van der Smagt, 'Minimisation methods for training feedforward neural networks", *Neural Networks* **7:** 1-11, 1994.