Neural Computing & Applications manuscript No.
(will be inserted by the editor)

An advanced active set L-BFGS algorithm for training weight-constrained neural networks

Ioannis E. Livieris

the date of receipt and acceptance should be inserted later

Abstract In this work, a new advanced active set limited memory BFGS (Broyden–Fletcher–Goldfarb–Shanno) algorithm is proposed for efficiently training weight-constrained neural networks, called AA-L-BFGS. The proposed algorithm possesses the significant property of approximating the curvature of the error function with high-order accuracy by utilizing the theoretically advantaged secant condition. Moreover, the global convergence of the proposed algorithm is established provided that the line search satisfies the modified Armijo condition. The presented numerical experiments illustrate the efficiency of the proposed AA-L-BFGS, providing empirical evidence that it significantly accelerates the convergence of the training process.

Keywords Artificial neural networks · constrained optimization · L-BFGS · modified secant equation.

1 Introduction

Artificial Neural Networks (ANNs) constitute intelligent dynamic system models, which have been characterized as probably the most powerful machine learning algorithms for extracting knowledge from complex and ill-defined problems. Their universal approximation ability, as well as their self-learning and self-adapting capability have established them as vital components of decision support systems; Therefore, ANNs have been widely applied in an impressive spectrum of real-world applications [2,9,14,18].

The standard problem of *training* an ANN is the incremental adaptation of connection weights, in order to minimize the measure of difference between the actual output and the desired output of the network for all examples of the training data [27]. Mathematically, the training process can be formulated as the minimization of an error function $E(w)$ which depends on a vector w of n weights of the neural network, namely

$$\min_{w\in\mathbb{R}^n} E(w) \tag{1}$$

Gradient-based training algorithms constitute an elegant choice for dealing with the optimization problem (1) and they produce a sequence of weights $\{w_k\}$ using the iterative recurrence

$$w_{k+1} = w_k + \eta_k d_k, \quad k = 0,1,\ldots,k_{\max} \tag{2}$$

where k is the current iteration (epoch), $k_{\max}$ is the maximum number of iterations, $w_0 \in \mathbb{R}^n$ is the initial vector of weights, $\eta_k > 0$ is a stepsize (learning rate) and d_k is a search direction. Notice that the gradient which constitutes primary importance for these algorithms, can be easily obtained by means of back propagation of errors through the network layers.

Nevertheless, the training process is a significantly challenging optimization problem since the error function $E(w)$ is nonconvex and usually high-dimensional. Additionally, it is characterized by broad flat regions adjoined with narrow step ones and a large number of local minima [22–24]. Therefore, several methodologies have been proposed based on the well-established unconstrained optimization theory in order to efficiently accelerate the convergence of the minimization process while situationally, provide good generalization performance. Karras and Perantonis [15,30] proposed a novel approach based on a Lagrange multiplier for the optimization of the error function. The advantage of the proposed algorithm was the avoidance of zig-zag trajectories in the parameter space since the weights updates in two successive iterations are highly aligned. Another interesting approach for increasing the generalization performance of an

I.E. Livieris
Department of Computer & Informatics Engineering, Technological Educational Institute of Western Greece, Greece, GR 263-34.
E-mail: livieris@teiwest.gr

ANN was the incorporation of nonmonotone learning strategies which attempt to exploit the accumulated information relative to the most recent values of $E(w)$. More sophisticated algorithms exploit second order derivative related information in order to accelerate the efficiency of the training process. Along this line, many limited memory quasi-Newton [3,4,11] and conjugate gradient algorithms [5,17,23] have been proposed which possess strong convergence properties and are computationally superior to state-of-the-art training algorithms.

Recently, Livieris [21] proposed a novel methodology for improving the generalization ability of ANNs based on the application of box constraints on the weights. In other words, the problem of training an ANN is re-formulated as a constrained optimization problem, i.e.

$$\min\{E(w)\,|\,w\in\mathscr{B}\} \tag{3}$$

with

$$\mathscr{B}=\{w\in\mathbb{R}^n : l\leq w\leq u\} \tag{4}$$

where $l\in\mathbb{R}^n$ and $u\in\mathbb{R}^n$ denote the lower and upper bounds on the weights, respectively. The rationale for this strategy aimed at restricting the weights of the trained network from taking large values in order to develop a more stable prediction model which is less likely to overfit the training data and less sensitive to minor variations in the inputs.

Additionally, for evaluating the efficiency of this new type of ANNs, Livieris [21] proposed a new Weight-Constrained Neural Network algorithm (WCNN) which handles the box constraints on the weights utilizing a gradient-projection strategy and exploits the computational efficiency of the L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) matrices. It is worth noticing that this new training methodology has been also evaluated in a variety of real-world problems, providing some interesting and promising results [22]. Additionally, in [26] an improved version of algorithm WCNN (iWCNN) was proposed, which exploits a new scaling factor for defining the initial Hessian approximation utilized in the L-BFGS formula.

Along this line, Livieris and Pintelas [25] presented an Adaptive nonmonotone Active set -weight constrained- Neural Network (AANN) training algorithm based on a conjugate gradient philosophy which consists of two distinct phases. In the first phase, AANN ensures a significant reduction in the error function (3) by efficiently exploiting a property of an active-set estimate, while in the second phase the superlinear convergent CG-DESCENT algorithm is utilized in a lower-dimensional space composed by the weights estimated as non-active ones.

Motivated by the previous research, we propose a new advanced active set limited memory BFGS neural network training algorithm, named AA-L-BFGS. The proposed algorithm exploits the efficient active set identification technique of Facchinei et al. [10] for handling the box constraints on the weights. Two attractive properties of AA-L-BFGS are that it approximates the curvature of the error function $E(w)$ with higher accuracy by utilizing an advanced secant condition and it utilizes line search satisfying an Armijo-type condition. These properties result in considerably reducing the computational cost. Additionally, the global convergence of the proposed algorithm is established under mild conditions. The reported experimental results demonstrate empirical evidence that the proposed algorithm increases the convergence of the training process and provides more stable and reliable prediction models.

The remainder of this paper is organized as follows. In Section 2, we present the proposed weight-constrained neural network training algorithm. Section 3 presents the global convergence analysis of our method. Section 4 presents the numerical experiments utilizing the performance profiles of Dolan and Morè [7]. Finally, Section 5 presents the conclusions and out future research proposals.

Notations. Throughout this paper, the gradient of the error function $E(w)$ is indicated by $g(w)=\nabla E(w)$. The vectors $s_k=w_{k+1}-w_k$ and $y_k=\nabla E(w_{k+1})-\nabla E(w_k)$ represent the evolutions of the current point and of the error function gradient between two successive iterations, where k is the current iteration.

Moreover, a vector $\overline{w}\in\mathscr{B}$ is said to be a stationary point of problem (3) if it satisfies

$$\begin{cases} l_i=\overline{w}_i & \Rightarrow \nabla E_i(\overline{w})\geq 0;\\ l_i<\overline{w}_i<u_i & \Rightarrow \nabla E_i(\overline{w})=0;\\ u_i=\overline{w}_i & \Rightarrow \nabla E_i(\overline{w})\leq 0, \end{cases} \tag{5}$$

where $\nabla E_i(\overline{w})$ is the i-th component of the gradient vector at $\overline{w}$.

2 An advanced active L-BFGS neural network training algorithm (AA-L-BFGS)

In this section, we present the proposed advanced active set L-BFGS algorithm for efficiently training weight-constrained neural networks.

2.1 Advanced L-BFGS update

The process of training a weight-constrained network consists a rather challenging optimization problem since the surface of the error function $E(w)$ is characterized by high complexity and by a number of unhelpful features. More specifically, its dimensionality is often high and the corresponding nonconvex multimodal objective function possess broad flat regions adjoined with narrow step ones and has multitudes of local minima [21,25]. In order to increase the convergence rate of the minimization process, we modify the

L-BFGS updates by approximating with high-accuracy the curvature of the error function through the utilization of a theoretically advanced secant equation.

Firstly, we recall that for quasi-Newton methods, an approximation matrix H_k to the inverse of the Hessian of the nonlinear error function E is updated so that a new matrix H_{k+1} satisfies the following secant condition

$$H_{k+1}y_k = s_k. \tag{6}$$

Livieris and Pintelas [23] expanded condition (6) and proposed a new class of modified secant condition

$$H_{k+1}\widetilde{y}_k = s_k. \tag{7}$$

where

$$\begin{cases} \widetilde{y}_k = y_k + \delta_k \dfrac{\max\{\theta_k,0\}}{s_k^T u}u, \\ \theta_k = 2(E_k - E_{k+1}) + (g_{k+1}+g_k)^T s_k. \end{cases} \tag{8}$$

where $E_k = E(w_k)$, u is any vector satisfying $s_k^T u \neq 0$ and parameter $\delta_k \in \{0,1\}$. Notice that parameter δ_k is utilized to adaptively switch between the standard secant equation (6) and the modified secant equation (7), by setting $\delta_k = 1$ if $\|s_k\| \leq 1$ and setting $\delta_k = 0$, otherwise.

It is worth noticing that if $\|s_k\|$ is sufficiently small then

$$s_k^T(\nabla^2 E(w_{k+1})s_k - y_k) = O(\|s_k\|^3), \tag{9}$$
$$s_k^T(\nabla^2 E(w_{k+1})s_k - \tilde{y}_k) = O(\|s_k\|^4). \tag{10}$$

Clearly, the above equations imply that the modified secant equation (7) is superior to the classical one (6) in the sense that $\tilde{y}_k$ better approximates $\nabla^2 E(w_{k+1})s_k$ than y_k (see [23]).

Motivated by the theoretical advantages of the new secant equation (7), we propose a modification of the L-BFGS direction as follows. Let $\hat{m} = \min\{k, m-1\}$, then given the set of correction vector pairs (s_i, y_i) for $i = k-\hat{m}, \ldots, k-1$, the L-BFGS direction is defined by

$$d_k = -\widetilde{H}_k g_k, \tag{11}$$

where

$$\begin{aligned} \widetilde{H}_{k+1} = {} & (\widetilde{V}_k^T \ldots V_{k-\hat{m}}^T)\widetilde{H}_k^{(0)}(V_{k-\hat{m}}\ldots V_k) \\ & + \widetilde{\rho}_{k-\hat{m}}(\widetilde{V}_k^T \ldots V_{k-\hat{m}+1}^T)s_{k-\hat{m}}s_{k-\hat{m}}^T(V_{k-\hat{m}+1}\ldots V_k) \\ & + \widetilde{\rho}_{k-\hat{m}+1}(\widetilde{V}_k^T \ldots V_{k-\hat{m}+2}^T)s_{k-\hat{m}+1}s_{k-\hat{m}+1}^T(V_{k-\hat{m}+2}\ldots V_k) \\ & + \cdots + \\ & + \widetilde{\rho}_k s_k s_k^T. \end{aligned} \tag{12}$$

where

$$\widetilde{\rho}_k = \frac{1}{\widetilde{y}_k^T s_k} \qquad \text{and} \qquad V_k = I - \widetilde{\rho}_k \widetilde{y}_k s_k^T. \tag{13}$$

Notice that, in order to maintain the positive definiteness of the advanced L-BFGS matrix, the correction pair $(s_k, \widetilde{y}_k)$ is discarded in case the curvature $s_k^T \widetilde{y}_k < 0$ is not satisfied.

Conclusively, we point out that the rationale for utilizing the theoretically advanced secant equation (7) is to approximate with high-accuracy the curvature of the error function. Therefore, increasing the computational efficiency of the training algorithm and the convergence rate of the minimization process.

2.2 Search direction

In the sequel, we give a brief description of the active set identification technique which was originally proposed by Facchinei et al. [10] and it is utilized to define the search direction d_k at each iteration.

Let $\overline{w} \in \mathscr{B}$ be a stationary point of the constrained optimization problem (3). Moreover, let us consider the associated active constrained set

$$\overline{L} = \{i : \overline{w}_i = l_i\} \quad \text{and} \quad \overline{U} = \{i : \overline{w}_i = u_i\} \tag{14}$$

and the index set of free (non-active) variables

$$\overline{F} = \{1,2,\ldots,n\} \backslash (\overline{L} \cup \overline{U}). \tag{15}$$

By utilizing this notation, the condition (5) can be rewritten as

$$\begin{cases} \nabla E_i(\overline{w}) \geq 0, \ \forall i \in \overline{L}, \\ \nabla E_i(\overline{w}) = 0, \ \forall i \in \overline{F}, \\ \nabla E_i(\overline{w}) \leq 0, \ \forall i \in \overline{U}. \end{cases} \tag{16}$$

It is worth mentioning that strict complementarity is said to hold at a stationary point $\overline{w}$, if $\nabla E_i(\overline{w}) > 0$ and $\nabla E_i(\overline{w}) < 0$ in the first and third implication of (16), respectively.

Next, we define the following approximations $L(w)$, $U(w)$ and $F(w)$ to $\overline{L}$, $\overline{U}$ and $\overline{F}$, respectively:

$$\begin{aligned} L(w) &= \{i : w_i \leq l_i + a_i(w)\nabla E_i(w)\}; \\ U(w) &= \{i : w_i \geq u_i + b_i(w)\nabla E_i(w)\}; \\ F(w) &= \{1,2,\ldots,n\} \backslash (L(w) \cup U(w)), \end{aligned} \tag{17}$$

where $a_i(w)$ and $b_i(w)$ be nonnegative continuous and bounded functions defined on $\mathscr{B}$, such that if $w_i = l_i$ or $w_i = u_i$ then $a_i(w) > 0$ or $b_i(w) > 0$, respectively. The following theorem is very useful for presenting that $L(w)$, $U(w)$ and $F(w)$ are "good" estimates of $\overline{L}$, $\overline{U}$ and $\overline{F}$, respectively.

Theorem 1 [10]. *For any feasible w, $L(x) \cap U(x) = \emptyset$. Moreover, if $\overline{w}$ is a stationary point of problem* (3) *where strict complementarity holds, then there exists a neighborhood $N(\overline{w})$ of $\overline{w}$ such that*

$$L(x) = \overline{L}, \quad U(x) = \overline{U}, \quad F(x) = \overline{F}, \qquad \forall w \in N(\overline{w}). \tag{18}$$

For the sake of simplicity, we abbreviate $L(w_k)$, $U(w_k)$ and $F(w_k)$ to L_k, U_k and F_k, respectively. Next, we recall the active set identification technique and briefly describe how to determine a search direction. Let us consider the subspace direction $d_k^{F_k}$ at $w_k \in \mathscr{B}$ defined as the search direction for the inactive variables. Let Z_k be the matrix whose columns are $\{e_i : i \in F_k\}$, where e_i is the i-th column of the identity matrix and $\widetilde{H}_k$ be an approximation of the full space inverse Hessian matrix. Moreover, let $\overline{H}_k$ be an approximation of the Hessian matrix in the subspace, then $\overline{H}_k = Z_k^T \widetilde{H}_k Z_k$. Then, the search direction $d_k = (d_k^{L_k}, d_k^{U_k}, d_k^{F_k})$ is defined by

$$d_{k_i} = \begin{cases} l_i - w_{k_i}, & \forall i \in L_k; \\ u_i - w_{k_i}, & \forall i \in U_k; \\ -\eta_k^* \left(\overline{H}_k g_k^{F_k}\right)_i, & \forall i \in F_k, \end{cases} \tag{19}$$

where d_{k_i} and w_{k_i} are the i-th components of d_k and w_k, respectively and η_k^* is a positive scalar defined by

$$\eta_k^* = \max\left\{ \eta : \begin{array}{l} \eta \leq 1, l_i - w_{k_i} \leq -\eta \left(\overline{H}_k g_k^{F_k}\right)_i \leq u_i - w_{k_i}, \\ \text{with } i \in F(w_k) \end{array} \right\} \tag{20}$$

Hence, by the previous relation, it is not difficult to conclude that

$$w_k + d_k \in \mathscr{B}. \tag{21}$$

2.3 Training algorithm

At this point, we present a high level description of the proposed Advanced Active set L-BFGS (AA-L-BFGS) neural network training algorithm.

Algorithm 1: Advanced Active set L-BFGS (AA-L-BFGS)

Input: w_0 − Initial weights.
$\sigma \in (0,1)$ − Hyper-parameter of modified Armijo line search.
$\mu \in [0,\infty)$ − Hyper-parameter of modified Armijo line search.
$\beta \in (0,1)$ − Hyper-parameter of modified Armijo line search.
$\mathscr{L}_{\min}$ − Lower bound on the estimation of Lipschitz constant.
$\mathscr{L}_{\max}$ − Upper bound on the estimation of Lipschitz constant.
H_0 − Initial approximation of the Hessian matrix.
m − Number of correction vector pairs.
E_G − Error goal.

Output: w_k − Weights of the trained ANN.

[1] Set $k = 0$.
[2] **repeat**
[3] Calculate the error function value E_k and its gradient g_k.
[4] Determine L_k, U_k and F_k using (17).
[5] Determine the search direction d_k using (19).
[6] Calculate the safe-guarded approximation of Lipschitz constant

$$\mathscr{L}_k = \max\left(\mathscr{L}_{\min}, \min\left(\frac{s_k^T \widetilde{y}_k}{\|s_k\|^2}, \mathscr{L}_{\max}\right)\right). \tag{22}$$

[7] Set $\alpha_k = -\dfrac{d_k^T g_k}{\mathscr{L}_k \|d_k\|^2}$ and choose the learning rate η_k to be the largest one in $\{\alpha_k, \alpha_k\beta, \alpha_k\beta^2, \ldots\}$ satisfying the modified Armijo condition

$$E(w_k + \eta_k d_k) - E_k \leq \sigma \eta_k \left[g_k^T d_k - \frac{1}{2}\eta_k \mu \mathscr{L}_k \|d_k\|^2\right] \tag{23}$$

[8] Update the weights $w_{k+1} = w_k + \eta_k d_k$.
[9] Update the correction pairs $(s_i, \widetilde{y}_i)$, with $i = k, \ldots, k - \hat{m} + 1$.
[10] Update $\widetilde{H}_k$ using (12).
[11] Set $k = k + 1$.
[11] **until** ($E_k \leq E_G$ and $l \leq w_k \leq u$).

Let w_k be the current vector of weights. At the each iteration k, the algorithm calculated the error function value E_k and the gradient g_k at point w_k (Step 3). Then, the AA-L-BFGS algorithm determines the index sets L_k and U_k of active weights and the index set F_k of non-active weights (Step 4). In the sequel, AA-L-BFGS algorithm utilizing (19), determines the search direction d_k (Step 5). Notice that each component of the search direction d_{k_i} with $i \in L_k \cup U_k$, relative to the active weights, can be easily computed by $d_{k_i} = l_i - w_{k_i}$ and $d_{k_i} = u_i - w_{k_i}$ for $i \in L_k$ and $i \in U_k$, respectively. The rest components of the search direction d_{k_i} with $i \in F_k$ can be calculated using the advanced L-BFGS Hessian approximation of the subspace defined by the non-active weights. Next, the algorithm calculates the safe-guarded approximation of the Lipschitz constant and performs a line search procedure satisfying the modified Armijo condition (23) to determine the new connection weights w_{k+1} of the network (Steps 6-8). Finally, AA-L-BFGS updates the correction pairs and the Hessian approximation utilizing the advanced L-BFGS update (12) (Steps 9-10).

It is worth noticing that the proposed algorithm AA-L-BFGS has $O(m^2 n)$ complexity.

3 Global convergence analysis

In order to establish the global convergence result for Algorithm 1, we will impose the following assumptions.

Assumption 1 The level set $\Omega = \{w \in \mathbb{R}^n : E(w) \leq E(w_0)\} \cap \mathscr{B}$ is compact.

Assumption 2 In some neighborhood $\mathscr{N} \in \Omega$, E is differentiable and its gradient g is Lipschitz continuous, namely, there exists a constant $\mathscr{L} > 0$ such that

$$\|g(w) - g(\tilde{w})\| \leq \mathscr{L}\|w - \tilde{w}\|, \quad \forall w, \tilde{w} \in \mathscr{N}. \tag{24}$$

Assumption 3 There exist positive scalars c_1 and c_2 such that any matrix $\overline{H}_k$ satisfies

$$c_1\|z\|^2 \leq z^T \overline{H}_k z \leq c_2\|z\|^2, \qquad \forall z \in \mathbb{R}^{|F_k|}, z \neq 0 \tag{25}$$

Notice that since the error function E is bounded below in R^n by zero, it is differentiable and its gradient is Lipschitz continuous [12], Assumptions 1 and 2 always hold.

In the sequel, we will present some Lemmas which are significant for the establishment of global convergence of algorithm AA-L-BFGS.

Lemma 1 [10]. *Suppose that the sequences $\{w_k\}$ and $\{d_k\}$ are generated by Algorithm AA-L-BFGS, then there exists a positive constant γ such that*

$$g_k^T d_k \leq -\gamma\|d_k\|^2. \tag{26}$$

Lemma 2 [10]. *Suppose that the sequences $\{w_k\}$ and $\{d_k\}$ are generated by Algorithm AA-L-BFGS. Then, $d_k = 0$ iff w_k is a stationary point of problem* (3)*.*

Lemma 3 [10]. *Suppose that the sequences $\{w_k\}$ and $\{d_k\}$ are generated by Algorithm AA-L-BFGS. Furthermore, suppose that the subsequences $\{w_k\}_K \to \overline{w}$ and $\{d_k\}_K \to 0$ as $k \to \infty$. Then, $\overline{w}$ is a stationary point of* (3)*.*

Next, making use of Lemmas 1, 2 and 3 we can establish the global convergence theorem for Algorithm AA-L-BFGS whose proof is similar to that of Theorem 2.1 in [10], however we present it here for completeness.

Theorem 2 *Suppose that Assumptions 1 and 3 hold. If the sequence $\{w_k\}$ is generated by Algorithm AA-L-BFGS, then every limit point of this sequence is a stationary point of the problem* (3)*.*

Proof. From relations (21) and (23) and Lemma 1 we have that the sequence of weights $\{w_k\}$ are generated by Algorithm AA-L-BFGS are contained in the compact set Ω. Thus, it immediately follows from Assumption 1 that there exists at least a limit point of this sequence.

If the sequence $\{w_k\}$ is finite with last point $\overline{w}$ then by Lemma 2 we obtain that $\overline{w}$ is a stationary point of (3). Next, we assume that the sequence is infinite. Let

$$K_1 = \{k \,|\, \eta_k = \alpha_k\} \qquad \text{and} \qquad K_2 = \{k \,|\, \eta_k < \alpha_k\}. \tag{27}$$

In the sequel, we consider the following cases:
Case I: If $k \in K_1$, then by Lemma 1 and the modified Armijo line search (23), we have

$$E(w_k + \eta_k d_k) - E_k \leq \sigma\eta_k \left[g_k^T d_k - \frac{1}{2}\eta_k \mu \mathscr{L}_k \|d_k\|^2\right] \tag{28}$$

$$= -\sigma \left(\frac{g_k^T d_k}{\|d_k\|^2}\right) \left(\frac{2+\mu}{2\mathscr{L}_{\max}}\right) g_k^T d_k \tag{29}$$

$$\leq -\sigma\gamma^2 \left(\frac{2+\mu}{2\mathscr{L}_{\max}}\right) \|d_k\|^2. \tag{30}$$

Case II: If $k \in K_2$, then $\eta_k < \alpha_k$, hence $\eta_k/\beta \leq \alpha_k$. Let $\eta = \eta_k\beta^{-1}$, then by the modified Armijo line search (23), we obtain

$$E(w_k + \eta_k d_k) - E_k > \sigma\eta \left[g_k^T d_k - \frac{1}{2}\eta\mu\mathscr{L}_k\|d_k\|^2\right]. \tag{31}$$

Using the mean value theorem on the left-hand side of the above inequality, we have that there exists $\theta_k \in [0,1]$ such that

$$\eta g(w_k + \theta_k\eta d_k)^T d_k > \sigma\eta \left[g_k^T d_k - \frac{1}{2}\eta\mu\mathscr{L}_k\|d_k\|^2\right]. \tag{32}$$

Therefore, $\left(g(w_k + \theta_k\eta d_k) - g(w_k)\right)^T d_k > (\sigma - 1)\, g_k^T d_k - \frac{1}{2}\sigma\eta\mu\mathscr{L}_k\|d_k\|^2$, which together with Assumption 2 and the Cauchy–Schwartz inequality, we obtain

$$\left(\mathscr{L} + \frac{1}{2}\sigma\mu\mathscr{L}_k\right)\eta\|d_k\|^2 > (\sigma - 1)g_k^T d_k. \tag{33}$$

By re-arranging the previous relation, we have

$$\eta_k > \frac{2\beta(\sigma-1)}{2\mathscr{L} + \sigma\mu\mathscr{L}_k} \frac{g_k^T d_k}{\|d_k\|^2}. \tag{34}$$

Utilizing (22), (26) and (34), we can easily obtain

$$\eta_k > \frac{2\beta(1-\sigma)\gamma}{2\mathscr{L} + \sigma\mu\mathscr{L}_{\max}}. \tag{35}$$

Next, by the modified Armijo line search (23) and Lemma 1, we get

$$E(w_k + \eta_k d_k) - E_k \leq \sigma\eta_k \left[g_k^T d_k - \frac{1}{2}\eta_k\mu\mathscr{L}_k\|d_k\|^2\right] \tag{36}$$

$$\leq \frac{2\beta\sigma(\sigma-1)\gamma^2}{2\mathscr{L} + \sigma\mu\mathscr{L}_{\max}}\|d_k\|^2. \tag{37}$$

Since $\{E_k\}$ is decreasing and bounded from below, it immediately follows from (30) and (37) that $\{d_k\} \to 0$ which together with Lemma 3 completes the proof. □

4 Experimental results

In this section, we conduct a series of experiments for evaluating the performance of the proposed training algorithm AA-L-BFGS against AANN and iWCNN. These algorithms were selected since they constitute the only weight-constrained training algorithms proposed in the literature.

All training algorithms were implemented in Matlab 7.6 and evaluated on a laptop (2.4GHz Quad-Core processor, 4GB RAM). The initial weights of all weight-constrained neural networks were defined using the Nguyen-Widrow method [28]. The line search parameters of AA-L-BFGS were set as $\sigma = 10^{-2}$, $\mu = 0.01$ and $\beta = 0.5$ for all experiments [24]. We let $a_i(w) = b_i(w) = 10^{-6}$ and $H_0 = I$ as in [35] while the boundaries for the approximation of Lipschitz constant were set as $L_{\min} = 10^{-3}$ and $L_{\max} = 10^{8}$ as in [24, 32]. The number of correction pairs used in AA-L-BFGS is $m = 7$ and in order to maintain the positive definiteness of the advanced limited memory BFGS matrix (7), a correction pair $\{s_k, y_k\}$ is discarded if the curvature condition

$$s_k^T y_k > 10^{-8}\|\widetilde{y}_k\|^2, \tag{38}$$

is not satisfied [21,20]. Finally, the algorithms AANN and iWCNN were implemented with their default optimized parameter settings [26,25].

For conducting the performance evaluation of the training algorithms, we selected three well-known problems acquired from the UCI Repository [8]: the Escherichia coli problem, the Splice-junction gene sequences problem and the Yeast problem. It is worth mentioning that for reasons of more objective comparison, we decided that all classification problems were adjusted using their original "set-ups" introduced in [1,31,16].

For each benchmark, we performed 100 simulations for all training algorithms utilizing the same initial weights and presented the descriptive statistics including Minimum (min), Mean (mean), Maximum (max) and Standard Deviation (st.d) of CPU time in seconds. Nevertheless, since a small number of simulations tends to dominate the numerical results, the cumulative total for CPU time over all simulations does not seem to be too informative; thus, we also evaluated the performance of each training algorithm using the performance profiles of Dolan and Morè [7]. The utilization of performance profiles demonstrate perhaps the most complete information in terms of efficiency and solution quality and eliminates the influence of a small number of simulations on the evaluation process [7]. The curves in the following figures have the following meaning:

- "AA-L-BFGS" stands for the proposed Advanced Active set L-BFGS algorithm.
- "AANN" stands for Adaptive nonmonotone Active set -weight constrained- Neural Network training algorithm [25].
- "iWCNN" stands for improved Weight-Constrained Neural Network training algorithm [26].

Additionally, in order to investigate the sensitivity of the proposed algorithm AA-L-BFGS to the selection of bounds of the weights, we have selected three different bounds for the weights, namely $[-1,1]$, $[-2,2]$ and $[-5,5]$ as in [21, 22].

4.1 Escherichia coli classification problem

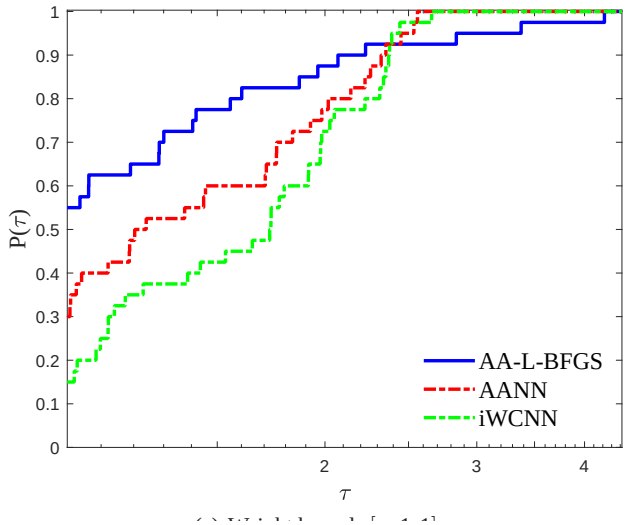
The Escherichia coli (*E. coli*) classification problem consists of 336 patterns and concerns the classification of the protein localization patterns into eight localization sites by employing some measures about the cell i.e cytoplasm (cp), inner-membrane (im), periplasm (pp), inner membrane Uncleavable signal sequence (imU), outer-membrane (om), outer-membrane Lipoprotein (omL), inner-membrane Lipoprotein (imL), inner membrane cleavable signal sequence (imS) [13]. For this imbalanced dataset, we used a neural network with 1 hidden layer of 16 neurons and an output layer of 8 neurons. The error goal E_G was set to 0.01, the maximum number of epochs was set to 1000 and the classification accuracy was measured utilizing 4-fold stratified cross-validation [21].

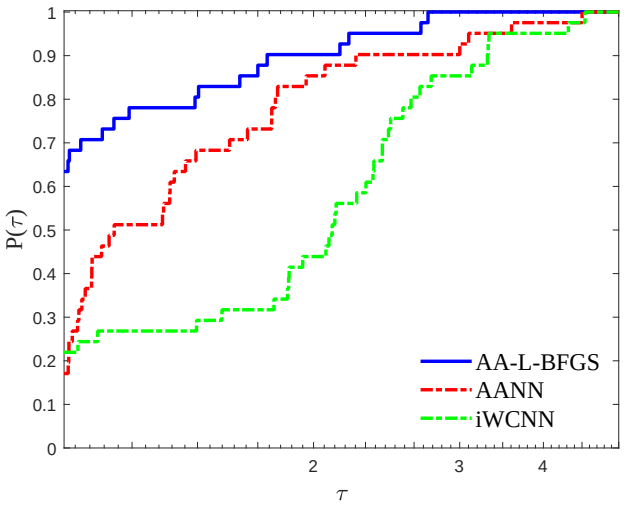
Table 1 presents the descriptive statistics including min, mean, max and st.d of CPU time in seconds for each weight-constrained training algorithm. Clearly, AA-L-BFGS exhibited the best overall performance, relative to all bound on the weights. More specifically, AA-LBFGS reported 1.40, 0.91 and 0.72 mean CPU time for bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$, respectively. In contrast, AANN reported 1.65, 1.03 and 0.8 while iWCNN reported 1.76, 1.22 and 0.8, in the same situations. Moreover, it is worth mentioning, that tighter bounds are most likely for AA-L-BFGS to considerably outperform the rest weight-constrained algorithms in terms of CPU time.

	AA-L-BFGS	AANN	iWCNN	Weight bounds
min	0.56	0.52	1.07	[-1,1]
mean	1.40	1.65	1.76	
max	2.62	2.76	2.78	
st.d	0.60	0.51	0.39	
min	0.61	0.65	0.57	[-2,2]
mean	0.91	1.03	1.22	
max	1.35	1.69	2.15	
st.d	0.19	0.24	0.36	
min	0.49	0.44	0.49	[-5,5]
mean	0.72	0.80	0.80	
max	0.92	1.14	1.03	
st.d	0.13	0.16	0.14	

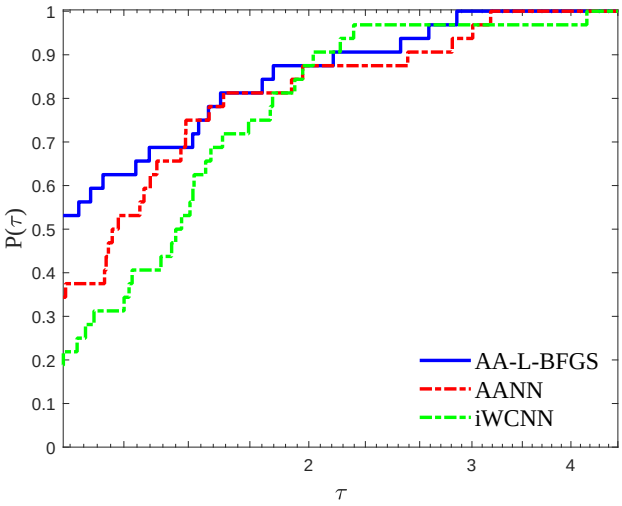
Table 1 CPU time (seconds) of the weight-constrained training algorithms for the Escherichia coli classification problem

Figure 1 presents the performance profiles for the Escherichia coli classification problem, based on CPU time. More specifically, AA-L-BFGS exhibited 55%, 61% and 51% of simulations with the least CPU time, with bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$ on all weights, respectively while AANN exhibited 30%, 17% and 32% of simulations, in the same situations. iWCNN presented the worst performance exhibiting 15%, 22% and 20% of simulations with the least CPU time, with bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$ on all weights, respectively. Therefore, the interpretation of Figure 1 demonstrates that the proposed algorithm AA-L-BFGS exhibits the highest probability of being the optimal training algorithm, regarding all bounds on the weights.





(b) Weight bounds $[-2,2]$



(c) Weight bounds $[-5,5]$

Fig. 1 Log_{10} scaled performance profiles for the Escherichia coli classification problem

4.2 Splice-junction gene sequences classification problem

Splice-junction gene sequences classification problem concerns the identification of exon/intron boundaries (EI), intron/exon boundaries (IE) or neither boundary (N), given a DNA sequence of 60 nucleotides [29]. For this classification problem, we used a neural network with 2 hidden layers of 4 and 2 neurons, respectively and an output layer of 3 neurons. The error goal E_G was set to 10^{-5}, the maximum number of epochs was set to 2000 and the classification accuracy was measured utilizing 10-fold stratified cross-validation [1].

Table 2 presents the descriptive statistics in terms of CPU time, regarding all weight-constrained training algorithms. For weight bounds $[-1,1]$, AA-L-BFGS reported 52.55 ± 29.7 CPU time while AANN and iWCNN reported 61.03 ± 26.4 and 67.42 ± 29.19, respectively. For weight bounds $[-2,2]$, AA-L-BFGS presented 35.88 ± 16.87 CPU time while AANN and iWCNN presented 40.95 ± 31.5 and 40 ± 23.03, respectively. For weight bounds $[-5,5]$, AA-L-BFGS exhibited 27.37 ± 10.67 CPU time while AANN and iWCNN exhibited 39.36 ± 25.76 and 34.91 ± 17.78, respectively. Summarizing, the interpretation of Table 2 highlights that AA-L-BFGS requires 12.38%-30.47% and 10.3%-22.1% less CPU time on average, compared to AANN and iWCNN, respectively.

Figure 2 presents the performance profiles for the Splice-junction gene sequences classification problem Firstly, it is worth noticing that the proposed algorithm AA-L-BFGS reported the best overall performance, regarding all bounds on the weights, since its curves lie on the top. More specifically, AA-L-BFGS trained 62% of simulations with the least CPU time, with bounds $[-1,1]$ on the weights while AANN and iWCNN trained only 32% and 30% of simulations, respectively. For weights bounds $[-2,2]$, AA-L-BFGS reported

52% of simulations with the least CPU time while AANN and iWCNN reported only 32% and 28% of simulations, respectively. Finally, for weights bounds $[-5,5]$, AA-L-BFGS exhibited 65% of simulations with the least CPU time while AANN and iWCNN exhibited only 30% and 10% of simulations, respectively.

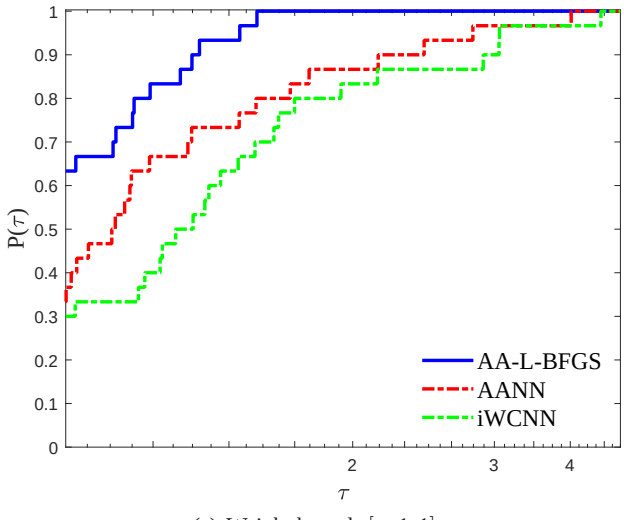
	AA-L-BFGS	AANN	iWCNN	Weight bounds
min	20.00	31.53	27.24	[-1,1]
mean	52.55	61.03	67.42	
max	142.77	149.41	148.08	
st.d	29.70	26.40	29.19	
min	15.27	15.94	15.36	[-2,2]
mean	35.88	40.95	40.00	
max	102.81	138.58	136.24	
st.d	16.87	31.50	23.03	
min	11.33	16.08	17.01	[-5,5]
mean	27.37	39.36	34.91	
max	49.09	134.85	88.02	
st.d	10.67	25.76	17.78	

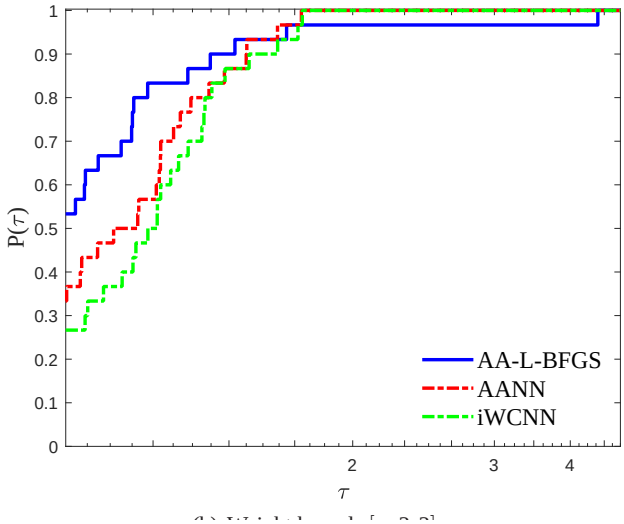
Table 2 CPU time (seconds) of the weight-constrained training algorithms for the Splice-junction gene sequences classification problem

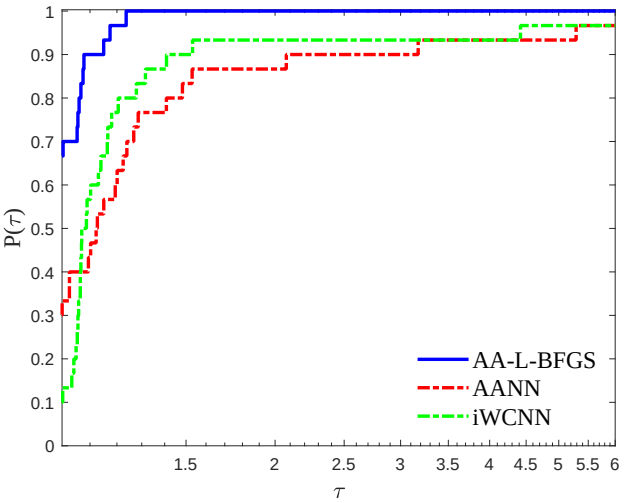
4.3 Yeast classification problem

This imbalanced classification benchmark concerns the determination of the cellular localization of the yeast proteins into ten localization sites [13]. The data consists of 1484 instances each of them having 8 features of real continuous values. For this problem, we used a neural network with 1 hidden layer 16 neurons while the error goal E_G and the maximum number of epochs were set to 0.05 and 1000, respectively. The classification accuracy was measured utilizing 4-fold stratified cross-validation [21].

Table 3 presents the descriptive statistics including min, mean, max and st.d of CPU time in seconds for Yeast classification problem. AA-L-BFGS exhibited 45.00, 19.87 and 18.62 mean CPU time for bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$, respectively while AANN exhibited 50.76, 21.42 and 19.18. Thus, we conclude that AA-L-BFGS requires 2.92%-11.35% less CPU time on average, compared to AANN. Additionally, iWCNN reported 66.23, 23.93 and 20.25 mean CPU time for bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$, respectively which implies that AA-L-BFGS requires 8.03%-32.06% less CPU time on average, compared to iWCNN. Finally, it is worth mentioning, that tighter bounds are most likely for AA-L-BFGS to considerably outperform the classical weight-constrained algorithm AANN and iWCNN, in terms of CPU time.







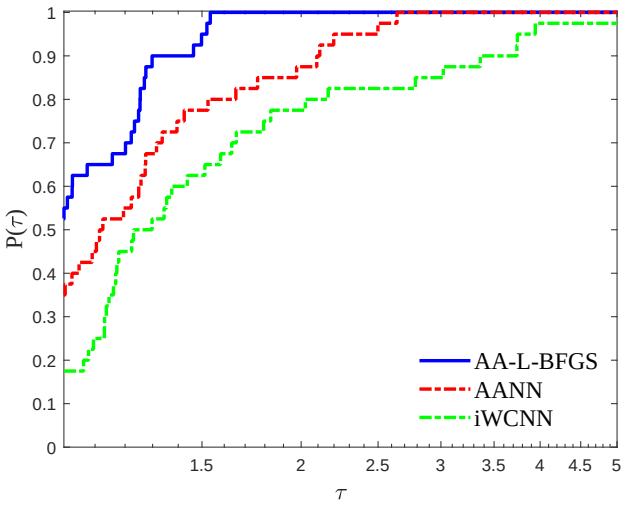
(c) Weight bounds $[-5,5]$

Fig. 2 Log_{10} scaled performance profiles for the Splice-junction gene sequences classification problem

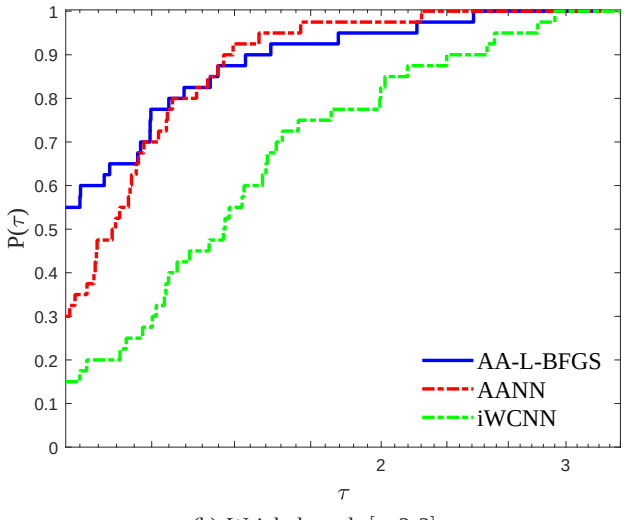
	AA-L-BFGS	AANN	iWCNN	Weight bounds
min	20.04	27.80	30.79	[-1,1]
mean	45.00	50.76	66.23	
max	190.95	165.00	125.96	
st.d	28.03	23.36	35.34	
min	11.82	14.49	16.33	[-2,2]
mean	19.87	21.42	23.93	
max	29.05	28.94	33.24	
st.d	3.80	2.51	4.13	
min	12.63	15.49	13.51	[-5,5]
mean	18.62	19.18	20.25	
max	27.36	23.72	27.36	
st.d	3.21	2.05	3.98	

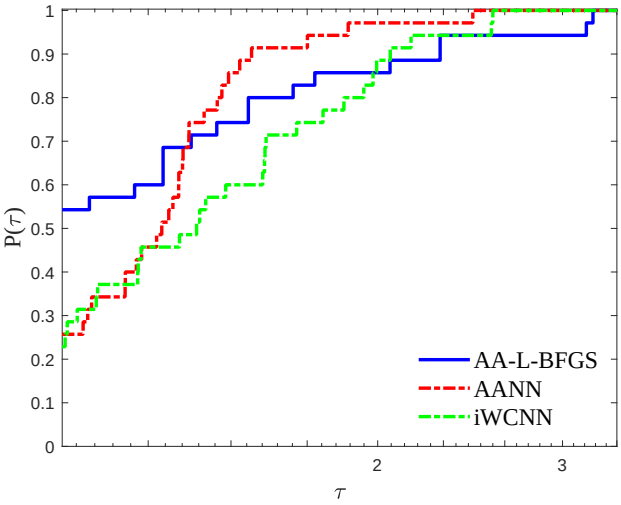
Table 3 CPU time (seconds) of the weight-constrained training algorithms for the Yeast classification problem

Figure 3 illustrates the performance profiles for the Yeast classification problem, relative to all bounds on the weights. Similar, conclusions can be made with the previous classification benchmarks. For weight bounds $[-1,1]$, AA-L-BFGS reported 52% of simulations with the least CPU time, with bounds $[-1,1]$ on the weights while AANN and iWCNN reported only 35% and 18% of simulations, respectively. For weight bounds $[-2,2]$, AA-L-BFGS exhibited 54% of simulations with the least CPU time while AANN and iWCNN exhibited only 30% and 16% of simulations, respectively. For weight bounds $[-5,5]$, AA-L-BFGS presented 54% of simulations with the least CPU time while AANN and iWCNN exhibited only 26% and 24% of simulations, respectively. Summarizing, the proposed training algorithm AA-L-BFGS exhibited the highest probability of being the optimal training algorithm, regarding all bounds on the weights.



(a) Weight bounds $[-1,1]$





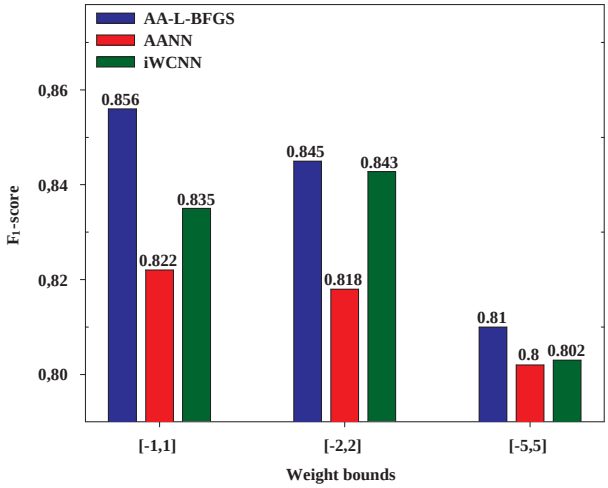
(c) Weight bounds $[-5,5]$

Fig. 3 Log_{10} scaled performance profiles for the Yeast classification problem

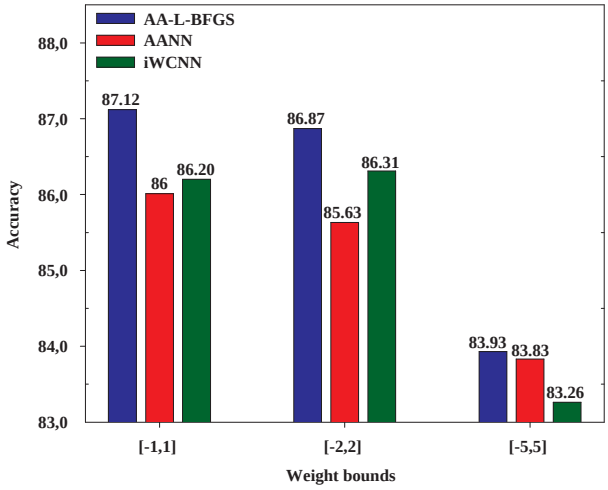
4.4 Generalization performance

In the sequel, we evaluate the classification performance of all weight-constrained training algorithms utilizing the performance metrics: F_1-*score* and *Accuracy*. Additionally, we compare the confusion matrices of all training algorithms, for each benchmark and utilized bounds on the weights. Notice in each case, all training algorithm were initiated with the same initial weights. The confusion matrix provides additional information about classes which are commonly mislabeled one as another, therefore providing a deeper insight to the classification performance of each training algorithm Each row of a confusion matrix represents the instances in a predicted class while each column represents the instances in an actual class.

Figure 4 presents the classification performance of AA-L-BFGS, AANN and iWCNN training algorithms, regarding the Escherichia coli problem. The proposed algorithm AA-L-BFGS exhibited the highest classification accuracy and F_1-score, relative to all bounds on the weights. Additionally, all training algorithms presented the best performance with bounds $[-1,1]$, closely followed by $[-2,2]$. while the classification performance of all algorithms was considerably decreased, for weight bounds $[-5,5]$.



(a) F_1-score



(b) Accuracy

Fig. 4 Classification performance of each weight-constrained neural network training algorithm on Escherichia coli problem

Tables 4, 5 and 6 present the confusion matrices of all training algorithms with weight bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$, respectively. AA-L-BFGS presented the best performance in 5 classes, namely "cp", "pp", "imU", "om" and "omL", regarding the weight bounds $[-1,1]$ and $[-2,2]$ and the best performance in 4 classes, namely "pp", "imU", "om" and "omL", for the weight bounds and $[-5,5]$. Additionally, notice that all training methods exhibit 0% performance the classes "imL" and "imS". It is worth noticing that for the classes "cp", "im" and "pp" all training algorithms exhibited similar performance while for the classes "imU", "om" and "omL" the proposed algorithm considerably outperform the classical weight-constrained training algorithms. Therefore, we concluded that AA-L-BFGS for this imbalanced set provided more efficient and reliable training.

	cp	im	pp	imU	om	omL	imL	imS
cp	141	0	2	0	0	0	0	0
im	5	61	0	11	0	0	0	0
pp	4	1	47	0	0	0	0	0
imU	1	11	0	23	0	0	0	0
om	0	0	3	0	17	0	0	0
omL	0	0	0	0	0	5	0	0
imL	0	0	0	1	0	1	0	0
imS	0	1	1	0	0	0	0	0

AA-L-BFGS

	cp	im	pp	imU	om	omL	imL	imS
cp	141	0	2	0	0	0	0	0
im	5	62	1	9	0	0	0	0
pp	4	2	46	0	0	0	0	0
imU	1	14	0	20	0	0	0	0
om	0	0	4	0	16	0	0	0
omL	0	0	0	0	1	4	0	0
imL	0	0	0	1	0	1	0	0
imS	0	1	1	0	0	0	0	0

AANN

	cp	im	pp	imU	om	omL	imL	imS
cp	141	0	2	0	0	0	0	0
im	5	60	1	11	0	0	0	0
pp	4	1	47	0	0	0	0	0
imU	1	14	0	20	0	0	0	0
om	0	0	4	0	16	0	0	0
omL	0	0	0	0	1	4	0	0
imL	0	0	0	1	0	1	0	0
imS	0	1	1	0	0	0	0	0

iWCNN

Table 4 Confusion matrices of all training algorithms with bounds $[-1,1]$ for Escherichia coli problem

	cp	im	pp	imU	om	omL	imL	imS
cp	141	0	2	0	0	0	0	0
im	5	60	1	11	0	0	0	0
pp	4	0	48	0	0	0	0	0
imU	2	12	0	21	0	0	0	0
om	0	0	4	0	16	0	0	0
omL	0	0	0	0	0	5	0	0
imL	0	0	0	1	0	1	0	0
imS	0	1	1	0	0	0	0	0

AA-L-BFGS

	cp	im	pp	imU	om	omL	imL	imS
cp	141	0	2	0	0	0	0	0
im	5	63	3	6	0	0	0	0
pp	4	2	46	0	0	0	0	0
imU	1	16	0	17	1	0	0	0
om	0	0	5	0	15	0	0	0
omL	0	0	0	0	1	4	0	0
imL	0	0	0	1	0	1	0	0
imS	0	1	1	0	0	0	0	0

AANN

	cp	im	pp	imU	om	omL	imL	imS
cp	141	0	2	0	0	0	0	0
im	5	60	1	11	0	0	0	0
pp	4	1	47	0	0	0	0	0
imU	1	12	0	22	0	0	0	0
om	0	0	4	0	16	0	0	0
omL	0	0	0	0	1	4	0	0
imL	0	0	0	1	0	1	0	0
imS	0	1	1	0	0	0	0	0

iWCNN

Table 5 Confusion matrices of all training algorithms with bounds $[-2,2]$ for Escherichia coli problem

	cp	im	pp	imU	om	omL	imL	imS
cp	139	0	4	0	0	0	0	0
im	5	60	1	11	0	0	0	0
pp	6	0	46	0	0	0	0	0
imU	4	12	0	19	0	0	0	0
om	0	0	5	0	15	0	0	0
omL	0	0	0	0	0	5	0	0
imL	0	0	0	1	0	1	0	0
imS	0	0	1	1	0	0	0	0

AA-L-BFGS

	cp	im	pp	imU	om	omL	imL	imS
cp	140	0	3	0	0	0	0	0
im	8	60	5	4	0	0	0	0
pp	4	1	47	0	0	0	0	0
imU	1	20	0	13	1	0	0	0
om	0	0	5	0	15	0	0	0
omL	0	0	0	0	1	4	0	0
imL	0	0	0	1	0	1	0	0
imS	0	1	1	0	0	0	0	0

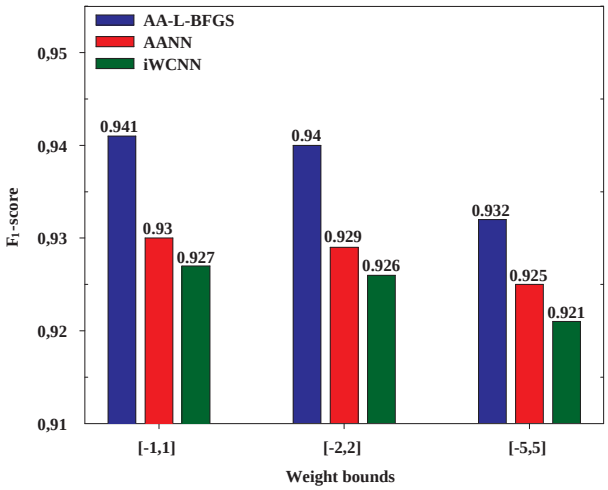
AANN

	cp	im	pp	imU	om	omL	imL	imS
cp	139	0	4	0	0	0	0	0
im	6	61	2	8	0	0	0	0
pp	4	3	45	0	0	0	0	0
imU	1	16	0	17	1	0	0	0
om	0	0	5	0	15	0	0	0
omL	0	0	0	0	1	4	0	0
imL	0	0	0	1	0	1	0	0
imS	0	1	1	0	0	0	0	0

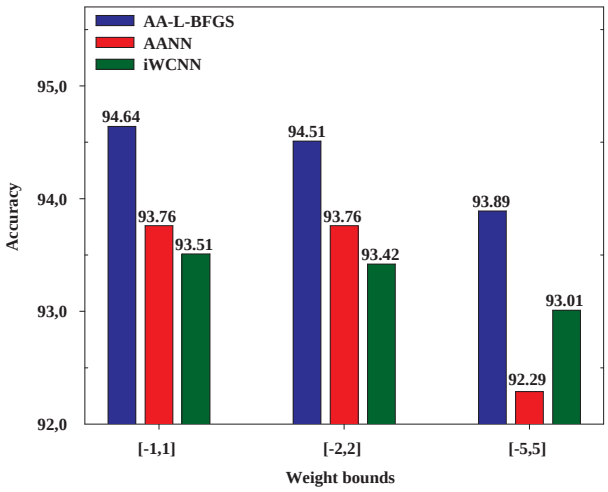
iWCNN

Table 6 Confusion matrices of all training algorithms with bounds $[-5,5]$ for Escherichia coli problem

Figure 5 presents the classification performance of all weight-constrained training algorithms, regarding the Splice-junction gene sequences problem. Clearly, the proposed algorithm AA-L-BFGS reported the highest classification accuracy and the best F_1-score, relative to all bounds on the weights. AA-L-BFGS presented 0.75%-1.6% and 0.007-0.011 greater generalization accuracy and F_1-score, respectively compared to AANN and iWCNN.



(a) F_1-score



(b) Accuracy

Fig. 5 Classification performance of each weight-constrained neural network training algorithm on Splice-junction gene sequences problem

Tables 7, 8 and 9 present the confusion matrices of all training algorithm with bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$, respectively. The proposed training algorithm AA-LBFGS reported the best performance for classes EI and IE, regarding all weight bounds. More specifically, AA-LBFGS presented 95.44%, 95.44% and 94.26% for class EI for weight

bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$, respectively while AANN reported 93.74%, 93.74% and 93.35%, in the same situations. Moreover, AA-LBFGS exhibited 93.62%, 93.49% and 92.71% for class IE for weight bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$, respectively while AANN reported 91.15%, 91.15% and 90.76% in the same situations. In contrast, AA-LBFGS presented slightly worst for class N, compared to both AANN and iWCNN.

	EI	IE	N
EI	732	21	14
IE	24	719	25
N	30	57	1568

AA-L-BFGS

	EI	IE	N
EI	719	29	19
IE	30	700	38
N	41	42	1572

AANN

	EI	IE	N
EI	718	30	19
IE	34	692	42
N	40	42	1573

iWCNN

Table 7 Confusion matrix of all training algorithms with bounds $[-1,1]$ for Genes problem

	EI	IE	N
EI	732	21	14
IE	24	718	26
N	31	59	1565

AA-L-BFGS

	EI	IE	N
EI	719	29	19
IE	30	700	38
N	41	42	1572

AANN

	EI	IE	N
EI	716	32	19
IE	34	692	42
N	41	42	1572

iWCNN

Table 8 Confusion matrix of all training algorithms with bounds $[-2,2]$ for Genes problem

	EI	IE	N
EI	723	30	14
IE	24	712	32
N	32	63	1560

AA-L-BFGS

	EI	IE	N
EI	716	32	19
IE	30	697	41
N	42	50	1563

AANN

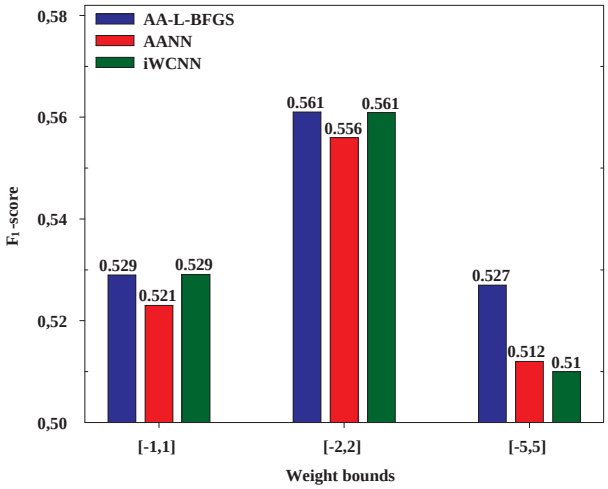
	EI	IE	N
EI	713	35	19
IE	35	688	45
N	41	48	1566

iWCNN

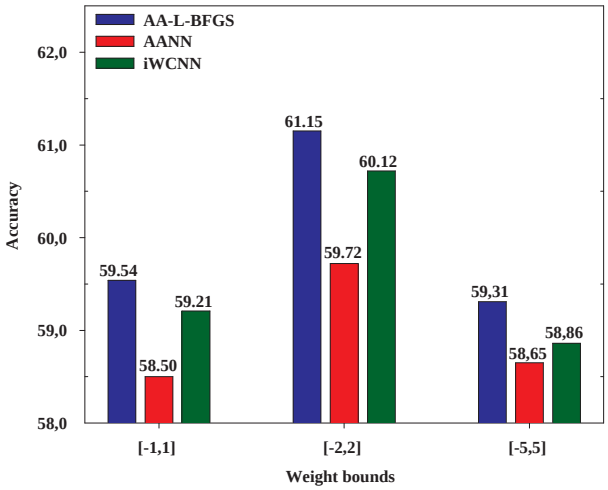
Table 9 Confusion matrix of all training algorithms with bounds $[-5,5]$ for Genes problem

Figure 4 presents the classification performance of AA-L-BFGS, AANN and iWCNN training algorithms, regarding the Yeast problem. Firstly, it is worth mentioning that AA-L-BFGS reported the highest classification accuracy, relative to all bounds on the weights. Furthermore, all training algorithms presented the best performance with bounds $[-2,2]$, while for weight bounds $[-1,1]$ and $[-5,5]$ they exhibited similar classification performance.

Tables 10, 11 and 12 present the confusion matrices, for weight bounds $[-1,1]$, $[-2,2]$ and $[-5,5]$, respectively. Clearly, the proposed training algorithm AA-LBFGS reported the best performance, considerably outperforming AANN and iWCNN. For weight bounds $[-1,1]$, AA-L-BFGS exhibited the best performance for the classes "mit" and "nuc", while for the rest classes, all training methods exhibited the same performance. For weight bounds $[-2,2]$, AA-L-BFGS presented the best performance in 6 and 3 out of 10 classes compared to AANN and iWCNN, respectively. For weight bounds $[-5,5]$, AA-L-BFGS outperformed AANN and iWCNN in 6 classes, i.e. "cyt", "erl", "me2", "me3", "mit", "nuc"; while for the rest classes, all training methods exhibited the same performance.



(a) F_1-score



(b) Accuracy

Fig. 6 Classification performance of each weight-constrained neural network training algorithm on Yeast problem

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	320	0	0	1	0	7	36	99	0	0
erl	0	5	0	0	0	0	0	0	0	0
exc	4	0	22	3	2	0	2	2	0	0
me1	0	0	7	32	4	0	1	0	0	0
me2	8	0	4	13	15	5	4	2	0	0
me3	17	0	0	0	2	123	4	17	0	0
mit	54	0	2	2	6	11	146	20	3	0
nuc	138	0	0	0	44	13	26	208	0	0
pox	7	0	0	0	0	0	1	1	11	0
vac	11	0	2	0	1	6	2	7	0	1

AA-L-BFGS

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	318	0	0	1	0	8	37	99	0	0
erl	0	5	0	0	0	0	0	0	0	0
exc	4	0	22	3	2	0	2	2	0	0
me1	0	0	7	32	4	0	1	0	0	0
me2	6	0	4	13	17	5	4	2	0	0
me3	17	0	0	0	2	123	3	18	0	0
mit	60	0	2	2	6	10	140	20	4	0
nuc	139	0	0	0	45	13	28	204	0	0
pox	6	0	0	0	0	0	2	1	11	0
vac	11	0	2	0	1	6	2	7	0	1

AANN

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	320	0	0	1	0	7	33	102	0	0
erl	0	5	0	0	0	0	0	0	0	0
exc	4	0	22	3	2	0	2	2	0	0
me1	0	0	7	32	4	0	1	0	0	0
me2	8	0	4	13	15	5	4	2	0	0
me3	17	0	0	0	2	123	4	17	0	0
mit	56	0	2	2	6	11	143	20	3	0
nuc	140	0	0	0	44	13	26	207	0	0
pox	7	0	0	0	0	0	1	1	11	0
vac	11	0	2	0	1	6	2	7	0	1

iWCNN

Table 10 Confusion matrices of all training algorithms with bounds $[-1, 1]$ for Yeast problem

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	324	0	0	1	0	7	35	96	0	0
erl	0	5	0	0	0	0	0	0	0	0
exc	4	0	24	1	2	0	2	2	0	0
me1	0	0	6	34	3	0	1	0	0	0
me2	8	0	4	13	15	5	4	2	0	0
me3	17	0	0	0	2	125	3	16	0	0
mit	53	0	2	2	6	9	150	19	3	0
nuc	131	0	0	0	43	13	24	218	0	0
pox	6	0	0	0	0	0	2	1	11	0
vac	11	0	2	0	1	6	2	7	0	1

AA-L-BFGS

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	324	0	0	1	0	7	34	97	0	0
erl	0	5	0	0	0	0	0	0	0	0
exc	5	0	22	3	2	0	2	2	0	0
me1	0	0	6	33	3	0	1	0	0	0
me2	8	0	4	13	15	5	4	2	0	0
me3	22	0	0	0	2	120	3	16	0	0
mit	57	0	3	2	6	9	145	19	3	0
nuc	137	0	0	0	44	13	25	210	0	0
pox	6	0	0	0	0	0	2	1	11	0
vac	11	0	2	0	1	6	2	7	0	1

AANN

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	324	0	0	1	0	7	34	97	0	0
erl	0	5	0	0	0	0	0	0	0	0
exc	4	0	24	1	2	0	2	2	0	0
me1	0	0	6	34	3	0	1	0	0	0
me2	8	0	4	13	15	5	4	2	0	0
me3	22	0	0	0	2	120	3	16	0	0
mit	57	0	3	2	6	9	145	19	3	0
nuc	134	0	0	0	43	13	24	215	0	0
pox	6	0	0	0	0	0	2	1	11	0
vac	11	0	2	0	1	6	2	7	0	1

iWCNN

Table 11 Confusion matrices of all training algorithms with bounds $[-2, 2]$ for Yeast problem

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	319	1	0	0	1	3	38	99	1	0
erl	0	5	0	0	0	0	0	0	0	0
exc	4	0	21	4	2	0	3	1	0	0
me1	0	0	7	34	3	0	1	0	0	0
me2	8	0	4	13	15	5	4	2	0	0
me3	17	0	0	0	2	124	4	16	0	0
mit	58	0	2	2	6	9	145	19	3	0
nuc	182	1	2	0	1	12	25	206	0	0
pox	4	0	1	1	0	0	2	2	10	0
vac	13	0	3	0	2	7	2	3	0	0

AA-L-BFGS

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	318	1	0	0	1	3	39	99	1	0
erl	1	4	0	0	0	0	0	0	0	0
exc	4	0	21	4	2	0	3	1	0	0
me1	0	0	5	34	3	0	1	0	0	0
me2	8	0	4	14	14	5	4	2	0	0
me3	17	0	0	0	2	125	4	17	0	0
mit	60	0	2	2	6	10	138	20	4	0
nuc	184	1	2	0	2	13	27	200	0	0
pox	4	0	1	1	0	0	2	2	10	0
vac	14	0	3	0	2	6	2	3	0	0

AANN

	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
cyt	320	0	0	0	0	3	32	106	1	0
erl	0	5	0	0	0	0	0	0	0	0
exc	5	0	20	4	2	0	3	1	0	0
me1	0	0	6	35	3	0	1	0	0	0
me2	8	0	4	14	14	5	4	2	0	0
me3	17	0	0	0	2	120	4	20	0	0
mit	60	0	2	2	6	10	140	20	4	0
nuc	184	1	2	0	2	11	26	203	0	0
pox	4	0	1	1	0	0	2	2	10	0
vac	13	0	3	0	2	7	2	3	0	0

iWCNN

Table 12 Confusion matrices of all training algorithms with bounds $[-5, 5]$ for Yeast problem

Summarizing, the interpretation of Figures 4-6 and Tables 4-12 demonstrate the generalization ability of the proposed algorithm AA-L-BFGS. Moreover, we are able to conclude that the classification efficiency increases as the bounds of the weights get tighter; nevertheless, this is not a general case.

5 Conclusions & future research

In this work, we proposed a new advanced active set limited memory BFGS algorithm for training weight-constrained neural networks. A significant property of the proposed algorithm AA-L-BFGS is that it approximates the curvature of the error function with high-order accuracy exploiting the advanced secant condition proposed in [23]. Moreover, AA-L-BFGS exploits the efficient active set identification technique of Facchinei et al. [10] for handling the box constraints on the weights. Under mild conditions, we established the global convergence of the proposed algorithm provided that the line search satisfies the Armijo-type condition. It is worth noticing that in contrast to most weight-constrained training algorithms which uses Wolfe line search, the utilization of the modified Armijo line search, substantially decreases the number of gradient evaluations, and as a result the CPU

time of the training process is considerably decreased. The advantages of AA-L-BFGS are empirically proved by the reported experimental results which illustrate the classification efficiency of the proposed algorithm, presenting empirical evidence that it provides more efficient and reliable learning.

Since the value of the weight bounds is still under consideration an interesting next step could be the development of an auto-adjustable strategy based on a validation set to dynamically determine the bounds, during the training process. Another aspect of future research could be to incorporate in our proposed framework, more elegant and sophisticated preprocessing methods such as sensitivity methods and oversampling SMOTE techniques [6, 19] as well as to adopt other theoretical techniques which aim on reducing the size and complexity of network. Finally, we intend to pursue extensive empirical experiments in order to evaluate the performance of AA-L-BFGS on larger and more complex architectures such as recurrent neural networks and deep neural networks.

Compliance with ethical standards.

Conflict of interest: The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

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. S.M. Awan, M. Aslam, Z.A. Khan, and H. Saeed. An efficient model based on artificial bee colony optimization algorithm with neural networks for electric load forecasting. *Neural Computing and Applications*, 25(7-8):1967–1978, 2014.
3. A.T. Azar. Fast neural network learning algorithms for medical applications. *Neural Computing and Applications*, 23(3-4):1019–1034, 2013.
4. H. Badem, A. Basturk, A. Caliskan, and M.E. Yuksel. A new efficient training strategy for deep neural networks by hybridization of artificial bee colony and limited–memory BFGS optimization algorithms. *Neurocomputing*, 266:506–526, 2017.
5. K. Bilski, J. Smolag, and A.I. Galushkin. The parallel approach to the conjugate gradient learning algorithm for the feedforward neural networks. In *International Conference on Artificial Intelligence and Soft Computing*, pages 12–21. Springer, 2014.
6. K. Demertzis and L. Iliadis. Intelligent bio-inspired detection of food borne pathogen by DNA barcodes: the case of invasive fish species Lagocephalus Sceleratus. In *International Conference on Engineering Applications of Neural Networks*, pages 89–99. Springer, 2015.
7. E. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.
8. D. Dua and E. Karra Taniskidou. UCI machine learning repository, 2017.
9. Y. Erzin and T.O. Gul. The use of neural networks for the prediction of the settlement of one-way footings on cohesionless soils based on standard penetration test. *Neural Computing and Applications*, 24(3-4):891–900, 2014.
10. F. Facchinei, J. Júdice, and J. Soares. An active set Newton algorithm for large-scale nonlinear programs with box constraints. *SIAM Journal on Optimization*, 8(1):158–186, 1998.
11. L. Gatys, A.S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems*, pages 262–270, 2015.
12. J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, MA, 1991.
13. P. Horton and K. Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. In *Ismb*, volume 4, pages 109–115, 1996.
14. L. Iliadis, S.D. Mansfield, S. Avramidis, and Y.A. El-Kassaby. Predicting Douglas-fir wood density by artificial neural networks (ANN) based on progeny testing information. *Holzforschung*, 67(7):771–777, 2013.
15. D.A. Karras and S.J. Perantonis. An efficient constrained training algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 6(6):1420–1434, 1995.
16. K. Kayaer and T. Yıldırım. Medical diagnosis on Pima Indian diabetes using general regression neural networks. In *Proceedings of the international conference on artificial neural networks and neural information processing (ICANN/ICONIP)*, pages 181–184, 2003.
17. C.B. Khadse, M.A. Chaudhari, and V.B. Borghate. Conjugate gradient back-propagation based artificial neural network for real time power quality assessment. *International Journal of Electrical Power & Energy Systems*, 82:197–206, 2016.
18. S. Kostić and D. Vasović. Prediction model for compressive strength of basic concrete mixture using artificial neural networks. *Neural Computing and Applications*, 26(5):1005–1024, 2015.
19. F. Li, X. Zhang, X. Zhang, C. Du, Y. Xu, and Y.C. Tian. Cost-sensitive and hybrid-attribute measure multi-decision tree over imbalanced data sets. *Information Sciences*, 422:242–256, 2018.
20. D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.
21. I.E. Livieris. Improving the classification efficiency of an ANN utilizing a new training methodology. *Informatics*, 6(1), 2018.
22. I.E. Livieris. Forecasting economy-related data utilizing constrained recurrent neural networks. *Algorithms*, 12(85), 2019.
23. I.E. Livieris and P. Pintelas. A new conjugate gradient algorithm for training neural networks based on a modified secant equation. *Applied Mathematics and Computation*, 221:491–502, 2013.
24. I.E. Livieris and P. Pintelas. A new class of nonmonotone conjugate gradient training algorithms. *Applied Mathematics and Computation*, 266:404–413, 2015.
25. I.E. Livieris and P. Pintelas. An adaptive nonmonotone active set -weight constrained- neural network training algorithm. *Neurocomputing*, 2019.
26. I.E. Livieris and P. Pintelas. An improved weight-constrained neural network training algorithm. *Neural Computing and Applications*, 2019.
27. A.J. Maren, C.T. Harston, and R.M. Pap. *Handbook of neural computing applications*. Academic Press, 2014.
28. 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.
29. M.O. Noordewier, G.G. Towell, and J.W. Shavlik. Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in neural information processing systems*, pages 530–536, 1991.
30. S.J. Perantonis and D.A. Karras. An efficient constrained learning algorithm with momentum acceleration. *Neural Networks*, 8(2):237–249, 1995.

31. 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.
32. Z.J. Shi and S. Wang. Modified nonmonotone Armijo line search for descent method. *Numerical Algorithms*, 57(1):1–25, 2011.
33. Zhong Wan, Shuai Huang, and Xiao Dong Zheng. New cautious BFGS algorithm based on modified Armijo-type line search. *Journal of Inequalities and Applications*, 2012(1):241, 2012.
34. Z. Wei, G. Li, and L. Qi. New quasi-Newton methods for unconstrained optimization problems. *Applied Mathematics and Computation*, 175(2):1156–1188, 2006.
35. G. Yuan and X. Lu. An active set limited memory BFGS algorithm for bound constrained optimization. *Applied Mathematical Modelling*, 35:3561–3573, 2011.