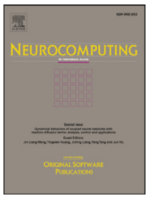
Neurocomputing 360 (2019) 294–303

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



An adaptive nonmonotone active set – weight constrained – neural network training algorithm

Ioannis E. Livieris*, Panagiotis Pintelas

Department of Mathematics, University of Patras, GR 265-00, Greece

ARTICLE INFO

Article history:
Received 17 January 2019
Revised 19 April 2019
Accepted 9 June 2019
Available online 13 June 2019

Communicated by Dr. Ding Wang

Keywords:
Artificial neural networks
Constrained optimization
Weight bounds
ASA method
Conjugate gradient algorithm
Accuracy

ABSTRACT

In this work, a new direction for improving the classification accuracy of artificial neural networks is proposed by bounding the weights of the network, during the training process. Furthermore, a new adaptive nonmonotone active set – weight constrained – neural network training algorithm is proposed in order to demonstrate the efficacy and efficiency of our approach. The proposed training algorithm consists of two phases: a gradient projection phase which utilizes an adaptive nonmonotone line search and an unconstrained optimization phase which exploits the box structure of the bounds. Also, a set of switching criteria is defined for efficiently switching between the two phases. Our preliminary numerical experiments illustrate that the classification efficiency of the proposed algorithm outperforms classical neural network training algorithms, providing empirical evidence that it provides more stable, efficient and reliable learning.

1. Introduction

Artificial Neural Networks (ANNs) are mathematical models which have been universally established as an intelligent mechanism for processing information in order to deal with function approximation, pattern recognition, process estimation and prediction. In recent years, various types and structures of neural networks have been developed while the feedforward neural networks probably constitute the most popular and widely used family of ANNs. Due to their excellent capability of self-learning and self-adapting they have showed enhanced generalization ability, adaptation competency and potent nonlinear input–output mapping [1–3]. Furthermore, they have often been found to be more accurate than other classification techniques thus they have been successfully applied in numerous applications of artificial intelligence [4–10].

Mathematically, the problem of *training* an ANN is highly consistent with the unconstrained optimization theory. More specifically, it can be formulated as the minimization of an error function $E(w)$ which depends on the connection weights $w \in \mathbb{R}^n$ of the network, namely

$$\min\{E(w)\ :\ w \in \mathbb{R}^n\}. \tag{1}$$

A traditional way to solve this problem is by an iterative gradient-based training algorithm which generates a sequence of weights $\{w_k\}$ using the update formula

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

where k is the current iteration usually called *epoch*, $w_0 \in \mathbb{R}^n$ is a given starting point, $\eta_k > 0$ is a stepsize (or learning rate) and d_k is a descent search direction. Moreover, the gradient can be easily obtained by means of back propagation of errors through the network layers [11].

The process of training an ANN has two significant performance considerations to be taken into account: *convergence speed* and *error minimization*. Increasing the convergence speed allows building of models on larger datasets with large numbers of parameters while reducing errors usually improves model prediction accuracy, thereby making the neural network more efficient. In the literature, since the development of the back-propagation [11], several modified and new algorithms have been proposed for improving the efficiency of the minimization error process and increase the generalization ability of the trained network. Most of these algorithms are based on the unconstrained optimization theory and utilize second order derivative related information, such as conjugate gradient algorithms [1,2,12,13] and quasi-Newton algorithms [14–16]. Nevertheless, most of these traditional training algorithms are monotone which implies that the convergence rate may be considerably reduced, especially when the iterations are trapped near a narrow curved valley [17].

* Corresponding author.
E-mail address: livieris@gmail.com (I.E. Livieris).

https://doi.org/10.1016/j.neucom.2019.06.033
0925-2312/© 2019 Elsevier B.V. All rights reserved.

To address this problem, an interesting approach has been suggested by Grippo et al. [18] who proposed to allow the iterative sequence to occasionally generate points with nonmonotone objective values, exploiting the accumulated information with regard to the most recent values of the function. The motivation behind this strategy is that, usually the first choice of a trial point by a minimization algorithm conceals significant information about the problem structure [18]. On the basis of this idea, many researchers have proposed new nonmonotone neural network training algorithms [19–24] providing some encouraging and promising results.

It is commonly known that the classification efficiency of a trained neural network depends on its architecture but mostly on the values of its weights. Nevertheless, some weights may have significantly larger values than others, therefore dominating the output of the network. In other words, the generalization ability of the neural network is degrading since some inputs and neurons are not efficiently exploited. To address this problem, Livieris [25] considered a novel approach by bounding the weights, during the training process, in order to be defined in a more uniform way. More analytically, the problem of training an ANN is considered as a constrained optimization problem, namely

$$\min\{E(w) \,:\, w \in \mathcal{B}\}, \tag{2}$$

with

$$\mathcal{B} = \{w \in \mathbb{R}^n \,:\, l \le w \le u\}, \tag{3}$$

where the vectors $l, u \in \mathbb{R}^n$ denote the lower and upper bounds on the weights w of the optimization problem, respectively. In order to evaluate the efficacy and the efficiency of this approach, a weight constrained training algorithm was proposed, named WCNN, presenting some interesting and promising results.

Motivated by the previous works, we propose a new weight constrained neural network training algorithm which is based on the well established constrained optimization theory and consists of two phases. In the first phase, the proposed algorithm exploits a property of an active-set estimate which ensures a significant reduction in the error function (2) when setting the bounds of all those weights estimated as active; while in the second phase a superlinear convergent conjugate gradient method is used in the subspace of the weights estimated as non-active ones.

The remainder of this paper is organized as follows: Section 2 presents a detailed description of the proposed neural network training algorithm. Section 3 presents our numerical experiments utilizing the performance profiles Dolan and Morè [26]. Finally, Section 4 presents the discussion and our concluding remarks.

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

For any $w \in \mathcal{B}$, let us define the set $\mathcal{A}(w)$ of active indices, as follows:

$$\mathcal{A}(w) = \{i \in [1, n] \,:\, w_i = l_i \text{ or } w_i = u_i\}.$$

Moreover, let $g_I(w)$ denote the vector whose components associated with the set $\mathcal{A}(w)$ are zero, while the rest of the components are identical to those of $g(w)$, namely

$$g_{I_i}(w) = \begin{cases} 0, & \text{if } i \in \mathcal{A}(w); \\ g_i(w), & \text{if } i \notin \mathcal{A}(w). \end{cases} \tag{4}$$

Finally, the active indices are considered to satisfy the strict complementarity condition in case $w_i = l_i$ and $g_i(w) > 0$ or $w_i = u_i$ and $g_i(w) < 0$.

2. Proposed neural network training algorithm

In this section, we present the proposed weight-constrained neural network training algorithm which constitutes the main contribution of this research. We recall that our training methodology is considered as a constrained optimization problem since bounds in the form of box constraints are applied to the neural network's weights, during the training process. The motivation for placing bounds on the values of weights, aims at efficiently training an ANN by defining the weights in a more uniform way in order to reduce the likelihood that some weights will "blow up" to unrealistic values. The proposed training algorithm demonstrates the effectiveness of our methodology and is based on the ASA method [27], one of the most successful and efficient large-scale bound-constrained optimization methods.

The proposed training algorithm consists of two phases: a gradient projection phase which utilizes an adaptive nonmonotone line search and an unconstrained optimization phase which exploits the box structure of the constraints in (2). Additionally, a set of switching criteria is defined for efficiently switching between the two phases.

2.1. Phase I: adaptive nonmonotone gradient projection

We present the adaptive Nonmonotone Gradient Projection Algorithm (aNGPA). Let P denote the projection onto the feasible set $\mathcal{B}$, namely

$$P(w) = \arg\min_{\overline{w} \in \mathcal{B}} \|w - \overline{w}\|.$$

Let $w_k \in \mathcal{B}$ be the current iterate. We compute an initial iterate $\overline{w}_k = w_k - \overline{\eta}_k^{\text{CBB}} g_k$, where $\overline{\eta}_k^{\text{CBB}}$ is the sage guarded Cyclic Barzilai and Borwein (CBB) stepsize, that is

$$\overline{\eta}_k^{\text{CBB}} = \max\{\eta_{\min}, \min\{\eta_{\max}, \eta_k^{\text{CBB}}\}\} \tag{5}$$

where $\eta_{\min} < \eta_{\max}$ are scalars for bounding the stepsize and

$$\eta_k^{\text{CBB}} = \begin{cases} \dfrac{s_{k-1}^T s_{k-1}}{y_{k-1}^T s_{k-1}} & \text{if } k \pmod{m} = 0; \\ \eta_{k-1}^{\text{CBB}} & \text{otherwise,} \end{cases} \tag{6}$$

where $m \ge 1$ is the cyclic length. The theoretical and numerical advantages of CBB stepsize as well as its adaptive calculation have been discussed in detail by Dai et al. [28].

The point $\overline{w}_k$ can be infeasible, so its projection $P(\overline{w}_k)$ on the feasible set is computed. In other words, if $\overline{w}_k$ is outsize $\mathcal{B}$, we apply the projection P to obtain a point $P(\overline{w}_k)$ on the boundary of $\mathcal{B}$. The search direction $d_k = P(\overline{w}_k) - w_k$ is along the line segment $[w_k, P(\overline{w}_k)]$. Finally, the next iteration w_{k+1} is obtained, utilizing a nonmonotone line search along the line segment connecting $P(\overline{w}_k)$ with w_k.

The performed line search can be considered as a modified version of the nonmonotone line search proposed by Hager and Zhang [27], equipped with an adaptive strategy for finding estimates of the memory element (also named nonmonotone learning horizon) M at each iteration based on the concept of the Lipschitz constant. More specifically, we adapt the approach presented in [23] which dynamically determines the size of M exploiting the morphology of the error surface through a local estimation of the Lipschitz constant. The significant advantage of this adaptive strategy relates to the fact of avoiding utilizing a poorly user-defined nonmonotone learning horizon. A high level description of the steps of this phase are presented in Table 1.

In Step 6, E_k^r denotes the "reference" error function value which is adaptively calculated based on a CBB scheme as in [27]. It is worth noticing that the condition $E_k \le E_k^r$ guarantees that the Armijo-type line search (Step 11) can be satisfied. Moreover, the

Table 1
Phase I: adaptive nonmonotone gradient projection.

```
Procedure: aNGPA
Input: $w_k$ – Current iterate.
Output: $w_{k+1}$ - Next iterate.

Step 1.  Calculate $\overline{\eta}_k^{\text{CBB}}$ using (5) and (6).
Step 2.  Set $d_k = P(w_k - \overline{\eta}_k^{\text{CBB}} g_k) - w_k$.
Step 3.  Calculate an approximation of the Lipschitz constant
         $L_k = \max\left\{L_{\min}, \min\left\{L_{\max}, \frac{\|g_k - g_{k-1}\|}{w_k - w_{k-1}}\right\}\right\}$.
Step 4.  Calculate $M_k$ by                                  /* Nonmonotone learning horizon */
         $M_k = \begin{cases} \max\{M_{\min}, \min\{M_{k-1}+1, M_{\max}\}\}, & \text{if } L_{k-2} > L_{k-1} > L_k; \\ \max\{M_{\min}, \min\{M_{k-1}-1, M_{\max}\}\}, & \text{if } L_{k-2} < L_{k-1} < L_k; \\ M_{k-1}, & \text{otherwise.} \end{cases}$
Step 5.  Set $E_k^{\max} = \max\{E(w_{k-i}) \ : \ 0 \le i \le \min(k, M_k - 1)\}$.   /* Local maximum of error */
                                                             /* function values near $w_k$ */
Step 6.  Choose $E_k^r \in [E_k, max\{E_{k-1}^r, E_k^{\max}\}]$ and $E_k^r \le E_k^{\max}$ infinitely often.
Step 7.  Set $E_R = \min\{E_k^r, E_k^{\max}\}$.                /* Check if the error function */
                                                             /* values change "too slowly" */
Step 8. If $(E(w_k + d_k) \le E_R + \sigma g_k^T d_k)$ then      /* Line search */
Step 9.      Set $\eta_k = 1$.
Step 10. else
Step 11.     Determine $\eta_k$ satisfying the following Armijo-type condition
             $E(w_k + \eta_k d_k) \le E_R + \sigma \eta_k g_k^T d_k$.
Step 12. end if
Step 13. Update the weights $w_{k+1} = w_k + \eta_k d_k$.
```

Table 2
Phase II: Unconstrained optimization.

```
Procedure: CG-DESCENT
Input: $w_k$ – Current iterate.
Output: $w_{k+1}$ –  Next iterate.

Step 1.  Calculate the active gradient components $g_I(w_k)$ using (4).
Step 2.  Compute the descent direction $d_k$ using (7) where the update parameter $\beta_k$ is defined by (8), (9) and (10).
Step 3.  Compute the learning rate $\eta_k$ satisfying the Wolfe line seach conditions
         $\phi(w_k) \le \phi(0) + c_1 \eta_k \phi'(0)$,
         $\phi'(w_k) \ge c_2 \phi(0)$,
       where $\phi(\eta) = E(P(w_k - \eta d_k))$ and $0 < c_1 < c_2 < 1$.
Step 4.  Update the weights $w_{k+1} = P(w_k + \eta_k d_k)$.
```

requirement "$E_k^r \le E_k^{\max}$ infinitely often" is needed for the global convergence result [27] and essentially checks if the error function values decrease "*too slowly*" in which case $E_k^r = E_k^{\max}$.

2.2. Phase II: unconstrained optimization

Although aNGPA in theory can deal with box constrained optimization quite efficiently and its global convergence for general functions has established [27], in practice its convergence speed can be slow especially near a local minimizer. To address this problem, similar to [27], CG-DESCENT algorithm is applied which operates in a lower-dimensional space since some weights of w are fixed. A significant advantage of this approach is that the search direction is computed only in the subspace composed by non-active variables. This allows savings in computational time, especially when dealing with large neural networks. In other words, after a suitable working face is detected by aNGPA, the proposed algorithm switches to the CG-DESCENT algorithm, in order to optimize over that face.

In general, CG-DESCENT [29] constitutes an efficient state-of-the-art unconstrained optimization method. This algorithm generates a sequence of points $\{w_k\}$, starting from an initial point $w_0 \in \mathbb{R}^n$, using the iterative formula

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

where $\eta_k > 0$ is the step length chosen by some line search and the search direction d_k is defined by

$$d_k = \begin{cases} -g_0, & \text{if } k = 0; \\ -g_k + \overline{\beta}_k^{HZ} d_{k-1}, & \text{otherwise.} \end{cases} \tag{7}$$

The update parameter $\overline{\beta}_k^{HZ}$ is calculated by

$$\overline{\beta}_k^{HZ} = \max\{\beta_k^{HZ}, \theta_k\}, \tag{8}$$

where

$$\beta_k^{HZ} = \frac{1}{d_{k-1}^T y_{k-1}} \left(y_{k-1} - d_{k-1} \frac{\|y_{k-1}\|^2}{d_{k-1}^T y_{k-1}} \right)^T g_k, \tag{9}$$

$$\theta_k = \frac{-1}{\|d_{k-1}\| \min\{\delta, \|g_{k-1}\|\}}. \tag{10}$$

where δ is a scalar.

2.3. Adaptive nonmonotone active set – weight constrained – neural network training algorithm

In the rest of this section, we present the proposed algorithm which utilizes the aNGPA to identify the active constraints (face of the feasible set $\mathcal{B}$) and the unconstrained optimization algorithm CG DESCENT to optimize $E(w)$ over a face identified by the aNGPA, exploiting its computational efficiency. Before, presenting the necessary set of rules for switching between the two phases, we give some useful notations.

Table 3
Algorithm 1: adaptive nonmonotone active set – weight constrained – neural network training algorithm.

```
Algorithm: Adaptive nonmonotone Active set – weight constrained – Neural Network (AANN)
Input:  $w_0$ – Initial weights.
        $\mu$ – Hyper-parameter.
        $\rho$ – Hyper-parameter.
Output: $w_k$ – Weights of the trained ANN.

Step 1.  Initiate $k = 0$ and $E^r_{-1} = E(w_0)$.
Step 2.  Set $Status$ = "aNGPA" and $Restart = False$.
Step 3.  repeat
Step 4.    if ($Status$ = "aNGPA") then                                /* Phase I: aNPGA */
Step 5.      Execute aNGPA.
Step 6.      Set $Restart = False$.
Step 7.      if ($\mathcal{U}(w_k) = \emptyset$) then
Step 8.        if $\|g_I(w_k)\| < \mu\|P(w - g(w)) - w\|$ then
Step 9.          $\mu = \rho\mu$.
Step 10.       else
Step 11.         $Status$ = "CG-DESCENT".                              /* Switch to CG-DESCENT */
Step 12.       end if
Step 13.     else if ($\mathcal{A}(w_k) = \mathcal{A}(w_{k-1}) = \cdots = \mathcal{A}(w_{k-n_1})$) then
Step 14.       if $\|g_I(w_k)\| \geq \mu\|P(w - g(w)) - w\|$ then
Step 15.         $Status$ = "CG-DESCENT".                              /* Switch to CG-DESCENT */
Step 16.       end if
Step 17.     end if
Step 18.   else                                                        /* Phase II: Unconstrained optimization */
Step 19.     Execute CG-DESCENT.
Step 20.     if ($\|g_I(w_k)\| < \mu\|P(w - g(w)) - w\|$) then
Step 21.       $Status$ = "aNGPA".                                     /* Restart aNGPA */
Step 22.       $Restart = True$.
Step 23.     else if ($|\mathcal{A}(w_k)| > |\mathcal{A}(w_{k-1})|$) then
Step 24.       if ($\mathcal{U}(w_k) \neq \emptyset$ and $|\mathcal{A}(w_k)| \leq |\mathcal{A}(w_{k-1})| + n_2$) then
Step 25.         $Status$ = "aNGPA".                                   /* Restart aNGPA */
Step 26.       end if
Step 27.       $Restart = True$.
Step 28.     else
Step 29.       $Restart = False$.
Step 30.     end if
Step 31.   end if
Step 32.   Set $k = k + 1$.
Step 33. until (stopping criterion).
```

For any $w \in \mathcal{B}$, and for fixed parameters $\alpha \in (0, 1)$ and $\beta \in (1, 2)$, we define the (undecided index) set

$$\mathcal{U}(w) = \left\{ i \in [1, n] : \begin{array}{l} |g_i(w)| \geq \|P(w - g(w)) - w\|^{\alpha} \text{ and} \\ w_i \geq \|P(w - g(w)) - w\|^{\beta} \end{array} \right\}.$$

which constitutes a fundamental set, embedded in the switching criteria. In fact, each index i contained in $\mathcal{U}$ corresponds to the component w_i which is not close to zero and the associated gradient component $g_i(w)$ is relatively large. The switching criteria are based on whether the set of undecided indices is empty or the active set subproblem is solved with sufficient accuracy.

In case, the set $\mathcal{U}(w)$ is empty, we consider that the indices with large associated gradient components are almost identified. Therefore, in this case, we switch from aNGPA to the unconstrained optimization algorithm CG-DESCENT to exploit its superior convergence.

Accordingly, when the ratio between the norm active gradient components $\|g_I(w)\|$ is sufficiently small relative to the error estimator $\|P(w - g(w)) - w\|$, we switch from CG-DESCENT to aNGPA. In other words, in case the condition

$$\|g_I(w_k)\| < \mu\|P(w_k - g(w_k)) - w_k\|, \tag{11}$$

where $\mu \in (0, 1)$ is a parameter is satisfied then we switch from CG-DESCENT to aNGPA. Notice that, aNGPA allows bound components of w_k to move into the interior of the feasible set in contrast to CG-DESCENT where bound components are fixed. Therefore, by switching from CG-DESCENT to aNGPA, the iterates are able to explore a new face of the feasible set.

At this point, we present a high level description of the proposed Adaptive nonmonotone Active set – weight constrained – Neural Network (AANN) training algorithm (Table 3). Initially, the aNGPA procedure is applied until the active constraints satisfying strict complementarity are identified. Then, we switch from aNGPA to the unconstrained optimization algorithm CG-DESCENT (Step 19) which is applied until a subproblem has been solved (Step 21). When new constraints are identified as active then the algorithm decides to restart either CG-DESCENT or aNGPA. Notice that, by restarting the CG-DESCENT, we denote that the iterates are generated by the CG-DESCENT using as the initial point the current iterate w_k. By restarting the aNGPA, we denote that w_0 in the aNGPA is initiated with the current iterate w_k.

Furthermore, in case $\mathcal{U}(w_k) = \emptyset$ and $\|g_I(w_k)\| < \mu\|P(w - g(w)) - w\|$ then the algorithm decreases parameter μ by a factor $\rho \in (0, 1)$ in which case the accuracy with which the subproblems are solved by CG-DESCENT is increased. As a result, parameter μ may become sufficiently small which implies that the condition (11) is never satisfied; therefore the algorithm may never switch to the aNGPA phase. In such a case, the constrained optimization problem (2) is solved by the CG-DESCENT algorithm; nevertheless, this case rarely observed in our numerical experiments.

Finally, it is worth noticing that since the proposed algorithm is based on the ASA method, it has $O(n)$ complexity and its global convergence has been established in [27].

3. Numerical experiments

In this section, we present experimental results in order to evaluate the performance of the proposed neural network training algorithm in four famous classification problems acquired from the UCI Repository of Machine Learning Databases [30]: the Wisconsin

Table 4
Parameter specification of procedure aNGPA.

Hyper-parameter	Description
$\eta_{\min} = 10^{-20}$ and $\eta_{\max} = 10^{20}$.	Bounds on the CBB stepsize.
$L_{\min} = 10^{-3}$ and $L_{\max} = 10^{8}$.	Bounds on the estimation of Lipschitz constant.
$M_{\min} = 3$ and $M_{\max} = 15$.	Bounds on the nonmonotone learning horizon.
$\sigma = 10^{-4}$.	Parameter entering in the nonmonotone line search.
$m = 4$.	Cyclic length.

Table 5
Parameter specification of procedure CG-DESCENT.

Hyper-parameter	Description
$\sigma_1 = 0.1$ and $\sigma_2 = 0.9$.	Parameter entering in the Wolfe line search.
$\delta = 0.4$.	Parameter entering in the update parameter.

Table 6
Parameter specification of algorithm AANN.

Hyper-parameter	Description
$\mu = 10^{-1}$.	$\|g_I(w_k)\| < \mu \|P(w - g(w)) - w\|$ implies the unconstrained optimization subproblem was efficiently solved.
$\rho = 0.5$.	Decay factor used to decrease μ in aNGPA.
$n_1 = 2$ and $n_2 = 1$.	Integer connected with active set repetitions or change.

diagnosis breast cancer problem, the Escherichia coli problem, the Pima Indians diabetes problem and the Yeast problem as in [25].

Our experimental analysis was obtained by conducting a two phase procedure: In the first phase, the classification performance of the proposed algorithm AANN was evaluated against the state-of-the-art neural network training algorithms Resilient backpropagation [31] and Levenberg–Marquardt training algorithm [32], and the efficient conjugate gradient algorithm CG-DESCENT. In the second phase, we evaluate the performance of AANN against the recently proposed Weight Constrained Neural Network (WCNN) training algorithm [25].

All neural networks had logistic activation functions, received the same sequence of input patterns and the initial weights were initiated using the Nguyen–Widrow method [33]. For evaluating classification accuracy we have used the standard procedure called *stratified k-fold cross-validation*. The implementation code was written in Matlab 7.6 and CGHZ, Rprop, LM and WCNN were utilized with their default optimized parameter settings [25,29,31,32]. Tables 4–6 present the values of parameters of aNPGA, CG-DESCENT and AANN, respectively. The simulations have been carried out on a PC (2.66 GHz Quad-Core processor, 4GB RAM) running Linux operating system while the results have been averaged over 100 simulations.

The cumulative total for a performance metric over all simulations does not seem to be too informative, since a small number of simulations tend to dominate these results. For this reason, we utilize the performance profiles of Dolan and Morè [26] relative to the performance metrics: *accuracy* and F_1-*score*, to present perhaps the most complete information in terms of robustness, efficiency and solution quality. The use of performance profiles eliminates the influence of a small number of simulations on the benchmarking process and the sensitivity of results associated with the ranking of solvers [26]. The performance profile plots, for every $\tau \geq 1$, the proportion $P(\tau)$ of simulations for which any training algorithm has a performance within a factor τ of the best algorithm.

3.1. First phase of experiments

In the sequel, we briefly describe each classification problem and present the performance comparison between the proposed algorithm AANN and the state-of-the-art algorithms Resilient backpropagation, Levenberg–Marquardt and CG-DESCENT. Notice that the classical training algorithm were utilized with their default optimized parameter settings. It is worth noticing that the Levenberg–Marquardt algorithm utilizes the gradient vector and the Jacobian matrix of (1); thus, it has been established as a popular choice for training small ANNs. In contrast, CG-DESCENT, Resilient backpropagation as well as AANN due to their low memory requirements and their simplicity of computations, can be also applied to train large neural networks. Furthermore, we recall that the main difference between the proposed AANN and the state-of-the-art training algorithm is that AANN trains a neural network with bounds on its weights, in order to reduce the likelihood that some weights will “blow up” to unrealistic values.

In order to explore the sensitivity of the proposed algorithm to the selection of bounds of the weights, similar to Livieris [25], we have selected three different bounds for the weights. Summarizing, the curves in the following figures have the following meaning

- “AANN$_1$” stands for algorithm AANN with bounds on the weights $-1 \leq w_i \leq 1$.
- “AANN$_2$” stands for algorithm AANN with bounds on the weights $-2 \leq w_i \leq 2$.
- “AANN$_3$” stands for algorithm AANN with bounds on the weights $-5 \leq w_i \leq 5$.
- “CGHZ” stands for the classical CG-DESCENT [29].
- “Rprop” stands for Resilient backpropagation [31].
- “LM” stands for Levenberg-Marquardt training algorithm [32].

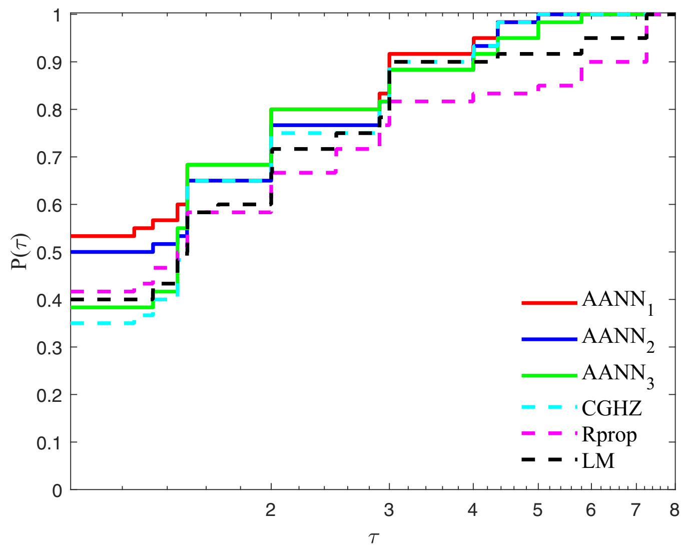
3.1.1. Wisconsin diagnosis breast cancer classification problem

The first benchmark concerns the diagnosis of breast cancer malignancy. The data were collected at the University of Wisconsin Hospital for the diagnosis and prognosis of breast tumors solely based on FNA test. This test involves fluid extraction from a breast mass using a small gauge needle and then visual inspection of the fluid under a microscope. The dataset contains 569 instances (357 benign – 212 malignant), where each instance has 32 attributes regarding FNA test measurements. We utilized a neural network with 1 hidden layer of 568 neurons and an output layer of 2 neurons [34]. The error minimization is set to 0.02, the maximum number of epochs is set to 2000 and all networks were tested using 10-fold cross validation.

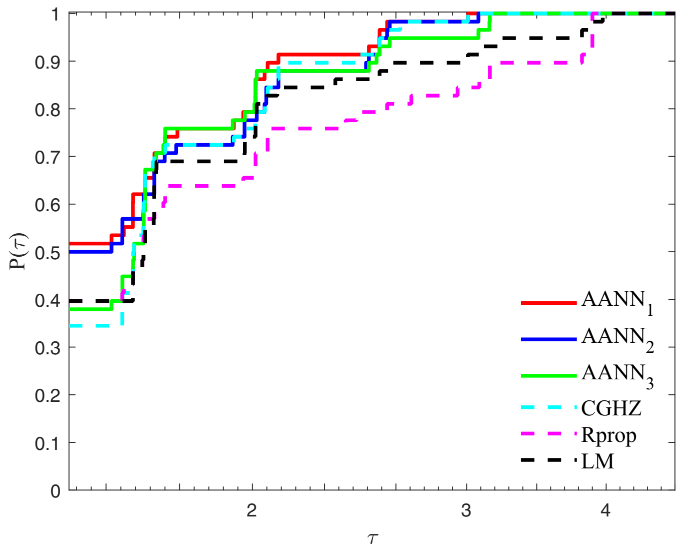
Fig. 1(a) and (b) presents the performance profiles for the breast cancer classification problem, based on accuracy and F_1-score, respectively. Firstly, it is worth noticing that AANN$_1$ and AANN$_2$ outperformed the classical training algorithms in terms of classification accuracy, presenting the highest probabilities of being the optimal solvers. AANN$_1$ and AANN$_2$ report 53.3% and 50% of simulations with the highest classification accuracy, respectively; while CGHZ, Rprop and LM report 35%, 41.6% and 40%, respectively. With regards to the F_1-score performance metric, AANN$_1$ and AANN$_2$ exhibited almost identical performance, outperforming the rest training algorithms. Therefore, the interpretation of Fig. 1 demonstrates that application of the bounds on the weights of the neural network, increased the overall classification accuracy.

3.1.2. Escherichia coli classification problem

This problem is based on an 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

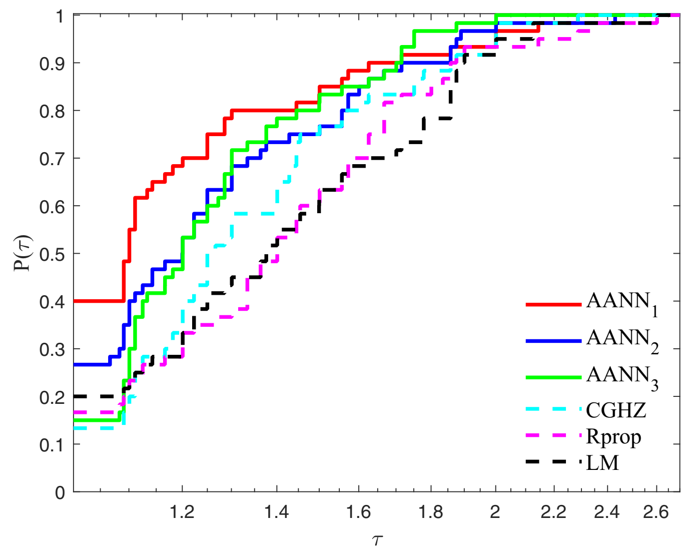


(a) Performance based on accuracy

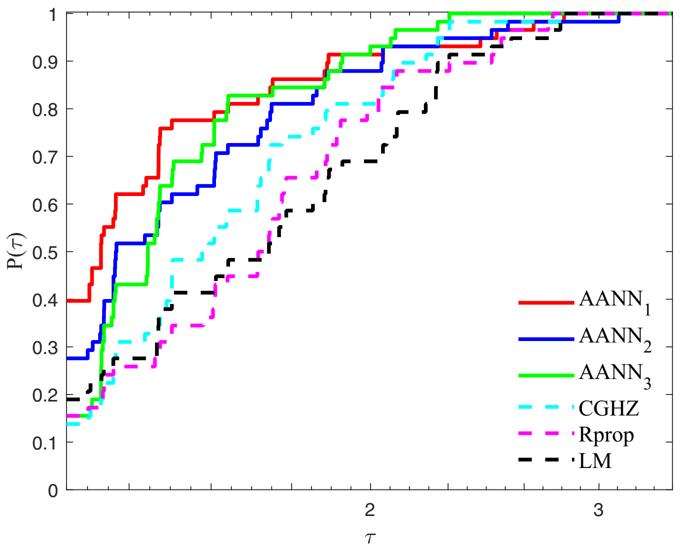


(b) Performance based on F_1-score

Fig. 1. Log_{10} scaled performance profiles for the Wisconsin diagnosis breast cancer classification problem.



(a) Performance based on accuracy



(b) Performance based on F_1-score

Fig. 2. Log_{10} scaled performance profiles for the Escherichia coli classification problem.

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%) [35]. We utilized a neural network with 1 hidden layer of 16 neurons and an output layer of 8 neurons [36]. The error goal is set to 0.02 within the limit of 2000 epochs and all networks were tested using 4-fold cross-validation [37].

Fig. 2 illustrates the performance profiles for $AANN_1$, $AANN_2$, $AANN_3$, CGHZ, Rprop and LM, regarding the Escherichia coli classification problem. Clearly, $AANN_1$ exhibit the highest probability of being the optimal solver, significantly outperforming all other training algorithms, followed by $AANN_2$. It is worth mentioning that $AANN_1$ and $AANN_2$ report 40% and 26.7% of simulations with the highest classification accuracy, respectively; while the state-of-the-art training algorithms CGHZ, Rprop and LM present 13.3%, 15% and 20%, respectively. Therefore, we can easily conclude that the tighter the bounds on the weights, the more efficient the resulting classification performance will be, in most cases. With regards to the performance of the proposed algorithm, $AANN_1$ illustrates the best performance; while $AANN_3$ exhibited the worst performance, relative to both performance metrics.

3.2. Pima Indian diabetes classification problem

The aim of this real-world classification task is to decide when a Pima Indian female is diabetes positive or not. The data of this benchmark consists of 768 different patterns each of them having 8 features of real continuous values and a class label (diabetic positive or not). We used a neural network with 2 hidden layers of 4 neurons each and an output layer of 2 neurons [38]. The error minimization is set to 0.14, the maximum number of epochs is set to 2000 and all networks were tested using 10-fold cross validation [39].

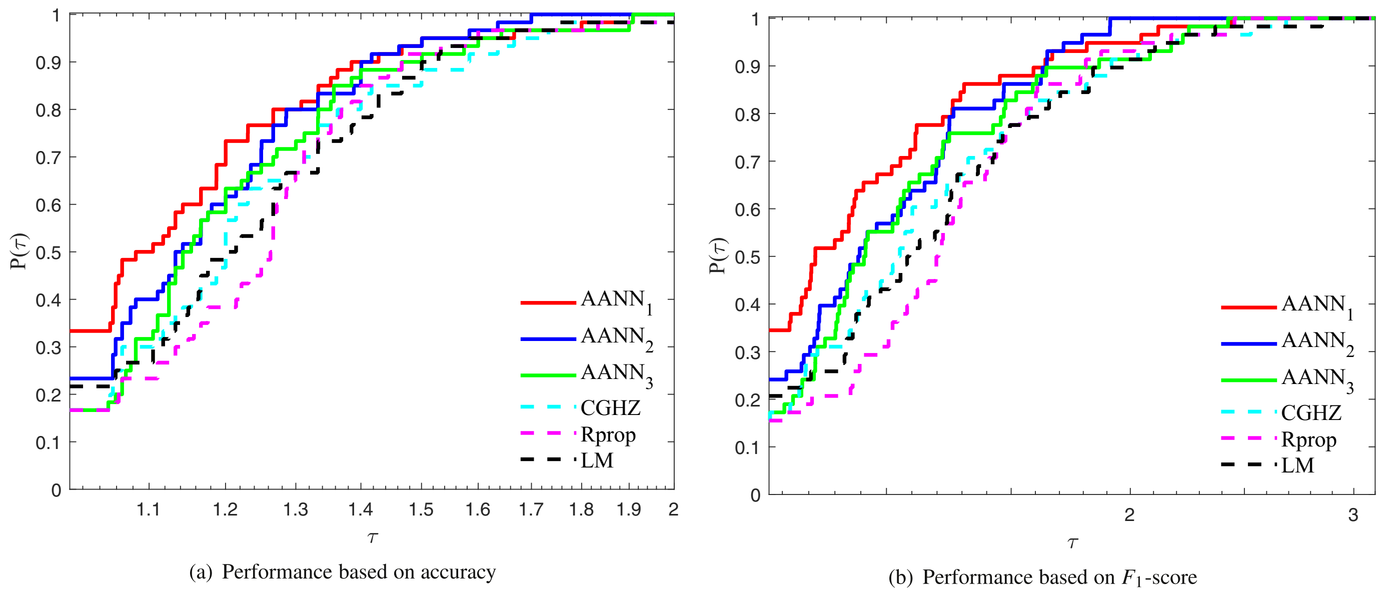


Fig. 3. Log_{10} scaled performance profiles for the Pima Indians diabetes classification problem.

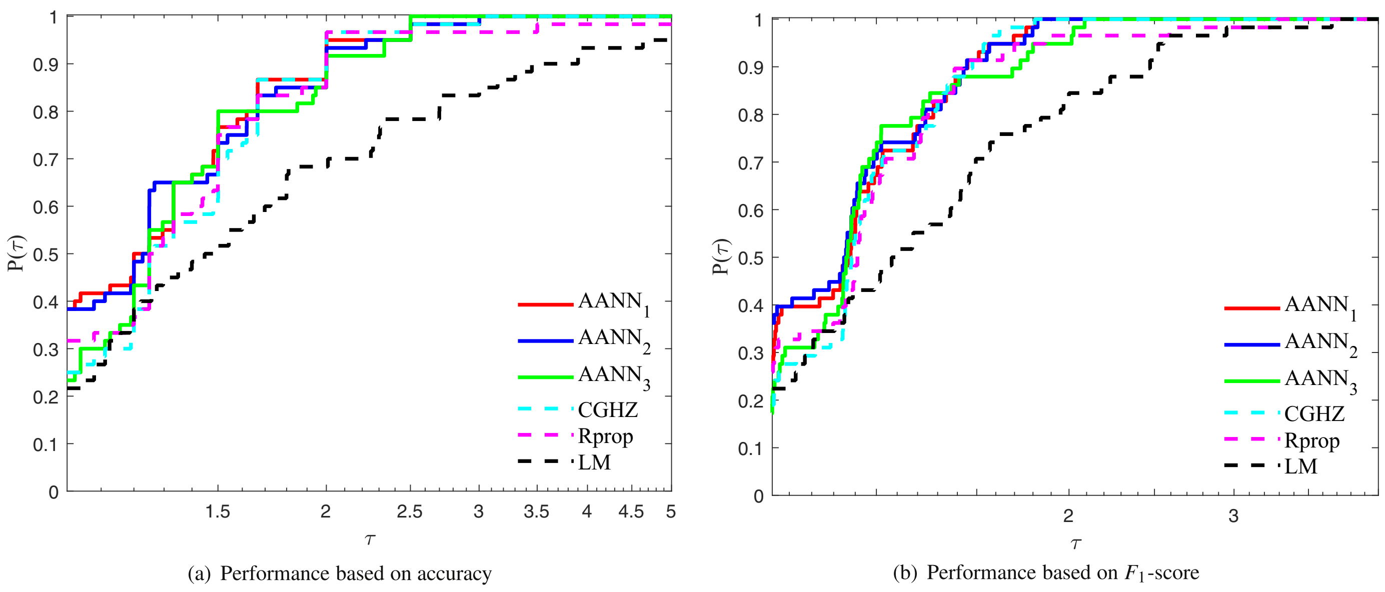


Fig. 4. Log_{10} scaled performance profiles for the Yeast classification problem.

Fig. 3(a) and (b) presents the performance profiles for the Pima Indian diabetes classification problem, based on the performance metrics accuracy and F_1-score, respectively. Firstly, it is worth mentioning that AANN_1 exhibits the highest probability of being the optimal algorithm since its curve lie on the top, regarding both performance profiles. Furthermore, AANN_2 and LM exhibit similar performance outperforming CGHZ and RPROP.

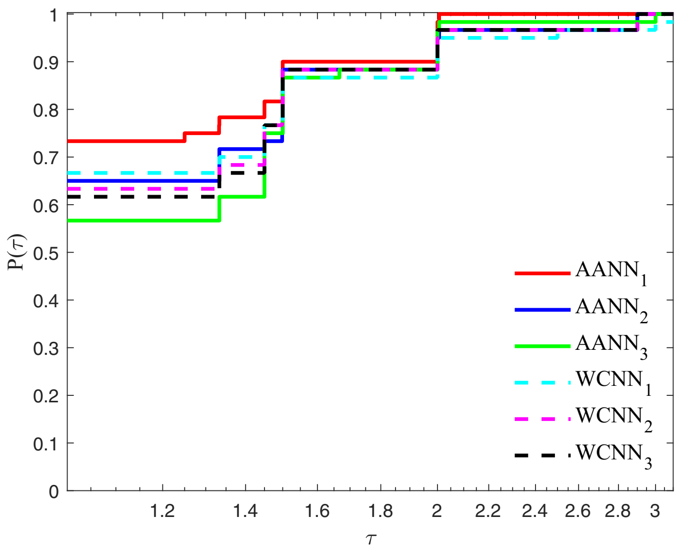
3.3. Yeast classification problem

This problem is based on an imbalanced dataset and concerns the determination of the cellular localization of the yeast proteins into ten localization sites. Saccharomyces cerevisiae (yeast) is the simplest Eukaryotic organism. For this classification problem, we utilized a neural network with 1 hidden layer of 16 neurons and an output layer of 10 neurons [36]. The error minimization is set to 0.05, the maximum number of epochs is set to 2000 and all networks were tested using 10-fold cross validation [37].

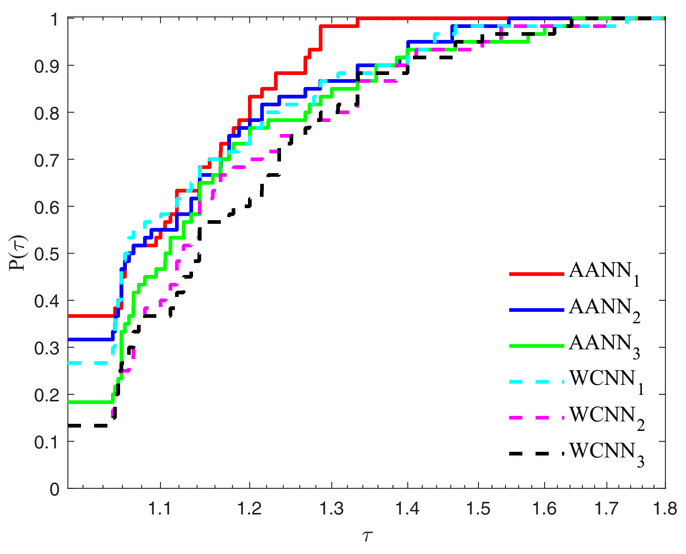
Fig. 4 presents the performance profile for the Yeast classification problem, investigating the performance of each training algorithm. AANN_1 and AANN_2 illustrate the highest probability of being the optimal training algorithm in terms of classification accuracy, since their curves lie on the top. Both AANN_1 and AANN_2 report 38.3% of simulations with the highest classification accuracy, respectively; while CGHZ, RPROP and LM report 25%, 31.6% and 21.6%, respectively. Moreover, the interpretation of Fig. 4(b) demonstrates that AANN_1 exhibits the best performance based on the F_1-score metric, followed by AANN_2. Finally, we conclude that the tighter the bounds get, the higher the chance for good generalization performance (i.e., the classification ability of the neural network will be higher).

3.4. Second phase of experiments

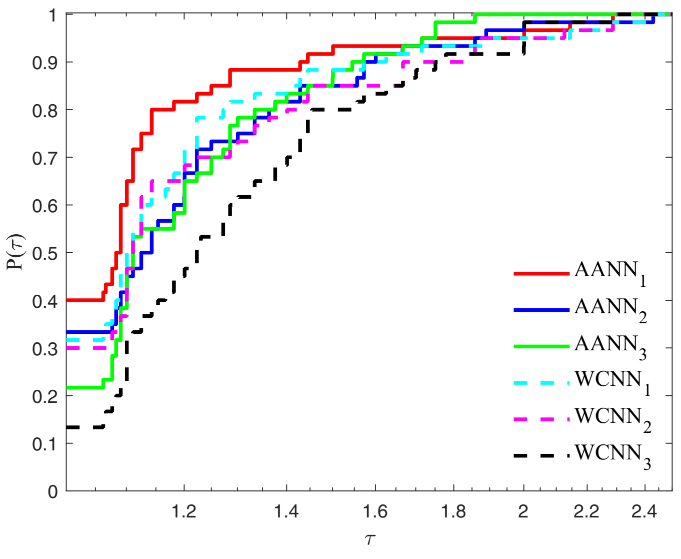
In the sequel, we compare the performance of the proposed neural network training algorithm AANN with that of the recently



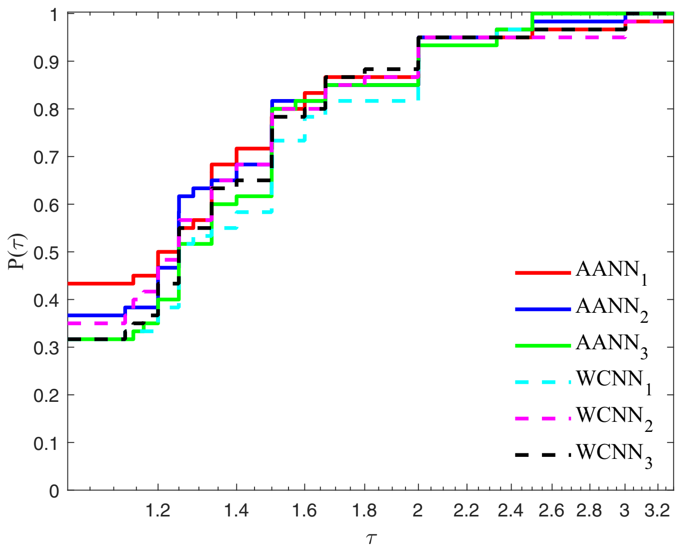
(a) Wisconsin diagnosis breast cancer problem



(c) Pima Indians diabetes problem



(b) Escherichia coli problem



(d) Yeast problem

Fig. 5. Log_{10} scaled performance profiles based on accuracy of AANN against WCNN.

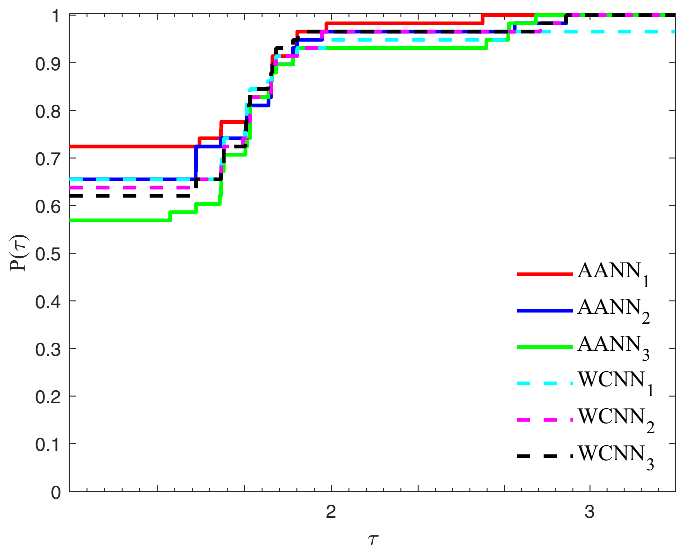
proposed Weight Constrained Neural Network (WCNN) training algorithm [25]. The curves in the following figures have the following meaning

- "AANN_1" stands for algorithm AANN with bounds on the weights $-1 \le w_i \le 1$.
- "AANN_2" stands for algorithm AANN with bounds on the weights $-2 \le w_i \le 2$.
- "AANN_3" stands for algorithm AANN with bounds on the weights $-5 \le w_i \le 5$.
- "WCNN_1" stands for algorithm WCNN [25] with bounds on the weights $-1 \le w_i \le 1$.
- "WCNN_2" stands for algorithm WCNN [25] with bounds on the weights $-2 \le w_i \le 2$.
- "WCNN_3" stands for algorithm WCNN [25] with bounds on the weights $-5 \le w_i \le 5$.

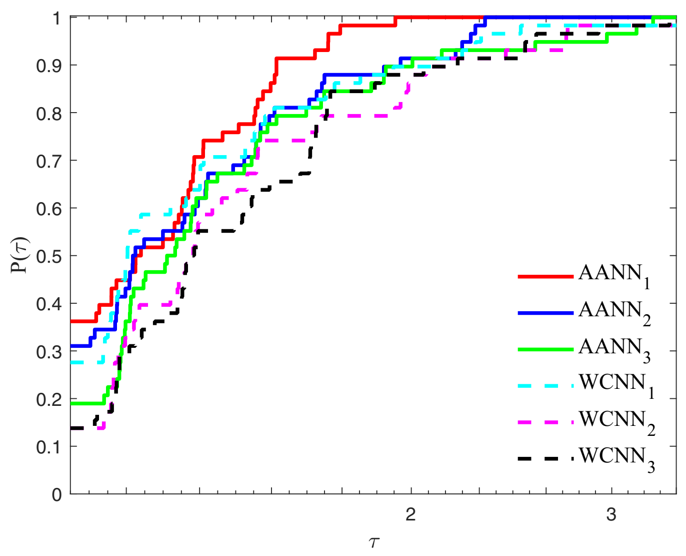
Fig. 5 presents the performance profiles of all versions of AANN and WCNN based on accuracy, regarding all classification problems. Firstly, it is worth noticing that all versions of AANN outperform the corresponding versions of WCNN, in terms of accuracy. AANN_1 outperforms all constrained training algorithms, followed by AANN_2 and WCNN_1 which report similar performance. Conclusively, it is worth noticing that similarly with the other benchmarks examined, the classification efficiency increases as the bounds of the weights get tighter.

Fig. 6 reports the performance profiles based on F_1-score of each constrained training algorithm, regarding all classification problems. Similar observations can be made with Fig. 5. Clearly, all versions of AANN outperform the corresponding versions of WCNN, relative to all benchmarks. AANN_1 demonstrates the best performance since its curves lie on the top in all classification problems, followed by AANN_2. Moreover, AANN_3 and WCNN_3 present the worst performance among all constrained training algorithms, with AANN_3 reporting slightly better performance.

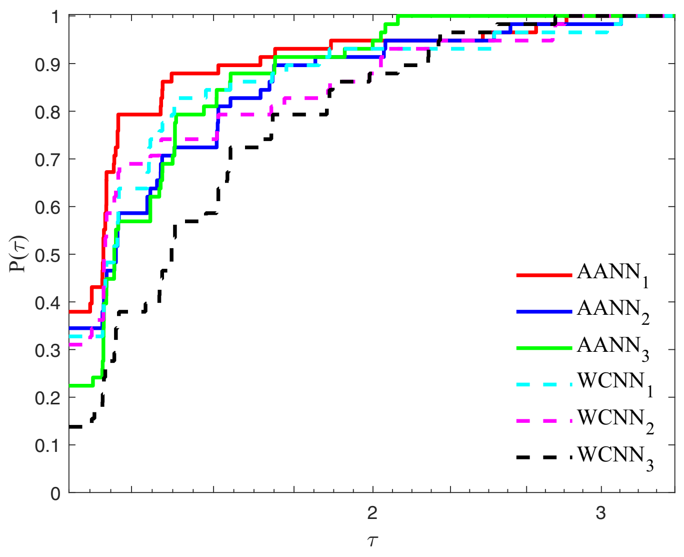
Conclusively, we point out that the interpretation of Figs. 5 and 6 show that regarding the constrained training algorithms, AANN develops neural networks with increased prediction accuracy in most cases. Furthermore, we can easily conclude that in general the tighter the bounds on the weights, the more efficient the resulting classification performance will be.



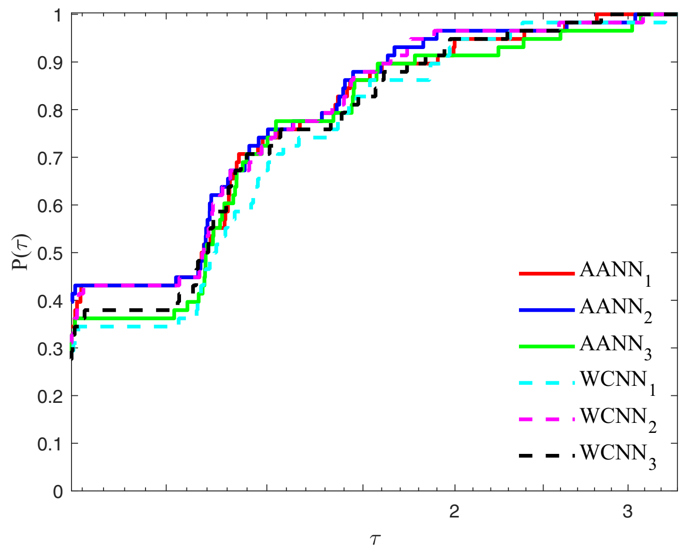
(a) Wisconsin diagnosis breast cancer problem



(c) Pima Indians diabetes problem



(b) Escherichia coli problem



(d) Yeast problem

Fig. 6. Log$_{10}$ scaled performance profiles based on F_1-score of AANN against WCNN.

4. Conclusions

In this work, we proposed a new – weight constrained – neural network training algorithm, based on the ASA method. The proposed algorithm exploits a property of an active-set estimate which ensures a significant reduction in the error function when setting to the bounds all those weights estimated as active; and utilizes an advanced conjugate gradient method in the subspace of the weights estimated as non-active to increase convergence. Moreover, the utilized nonmonotone line search of the proposed algorithm is equipped with an adaptive strategy for defining the nonmonotone learning horizon, exploiting the morphology of the error surface through a local estimation of the Lipschitz constant. Our preliminary numerical experiments demonstrate the classification efficiency of the proposed algorithm, providing empirical evidence that it provides more stable, efficient and reliable learning.

Additionally, it is worth mentioning that the bounds on the weights of a neural network increased the overall classification accuracy in most cases. Placing these constraints on the values of weights, reduces the likelihood that some weights will “blow up” to unrealistic values. Therefore, we conclude that the proposed methodology efficiently trains neural networks with improved classification ability.

It is worth noticing that the determination of optimal bounds on the weights is a rather difficult and challenging task; thus more research and experiments are needed. In our preliminary numerical experiments we observed that the classification efficiency increases as the bounds of the weights get tighter. Nevertheless, since this is not a general case, we cannot draw a safe conclusion and more experiments are needed. Our opinion is that since benchmarks and weights initialization are interrelated. we cannot draw a safe conclusion and more experiments are needed. To this end, the question of what should be the values of the bounds for each benchmark or which additional constraints should be applied is still under investigation. An interesting idea could be to auto-adjust the bounds during the training process using a strategy based on the utilization of a validation set. Probably, the required research to answer these questions, may reveal additional and crucial information and questions.

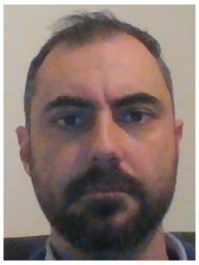
In our future work, we intent to incorporate and pursue an extensive empirical experimental evaluation of our proposed algorithmic framework to more advanced and complex architectures such as deep neural networks and Long Short Term Memory neural networks. Additionally, another interesting direction for future research is to evaluate the classification performance of AANN algorithm, together with regularization techniques such as dropout [40–42]. Moreover, since our experimental results are quite encouraging, our next step could be to evaluate the proposed algorithm in other real world datasets, such as educational, health care, etc.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] I.E. Livieris, P. Pintelas, An improved spectral conjugate gradient neural network training algorithm, Int. J. Artif. Intell. Tools 21 (01) (2012) 1250009.
[2] I.E. Livieris, P. Pintelas, A new conjugate gradient algorithm for training neural networks based on a modified secant equation, Appl. Math. Comput. 221 (2013) 491–502.
[3] R. Ahmed, M.E. Sayed, S.A. Gadsden, J. Tjong, S. Habibi, Artificial neural network training utilizing the smooth variable structure filter estimation strategy, Neural Comput. Appl. 27 (3) (2016) 537–548.
[4] M. ławryńczuk, Training of neural models for predictive control, Neurocomputing 73 (7–9) (2010) 1332–1343.
[5] H.B. Demuth, M.H. Beale, O. De Jess, M.T. Hagan, Neural Network Design, Martin Hagan, 2014.
[6] J. Misra, I. Saha, Artificial neural networks in hardware: a survey of two decades of progress, Neurocomputing 74 (1–3) (2010) 239–255.
[7] S. Asadi, J. Shahrabi, P. Abbaszadeh, S. Tabanmehr, A new hybrid artificial neural networks for rainfall–runoff process modeling, Neurocomputing 121 (2013) 470–480.
[8] S.P. Xiao, H.H. Lian, K.L. Teo, H.B. Zeng, X.H. Zhang, A new Lyapunov functional approach to sampled-data synchronization control for delayed neural networks, J. Frankl. Inst. 355 (17) (2018) 8857–8873.
[9] X.M. Zhang, Q.L. Han, X. Ge, D. Ding, An overview of recent developments in Lyapunov–Krasovskii functionals and stability criteria for recurrent neural networks with time-varying delays, Neurocomputing 313 (2018a) 392–401.
[10] X.M. Zhang, Q.L. Han, J. Wang, Admissible delay upper bounds for global asymptotic stability of neural networks with time-varying delays, IEEE Trans. Neural Netw. Learn. Syst. 99 (2018b) 1–11.
[11] D. Rumelhart, G. Hinton, R. Williams, Learning internal representations by error propagation, in: D. Rumelhart, J. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, Massachusetts, 1986. 318–362
[12] J. Wang, W. Wu, J.M. Zurada, Deterministic convergence of conjugate gradient method for feedforward neural networks, Neurocomputing 74 (14–15) (2011) 2368–2376.
[13] J. Wang, B. Zhang, Z. Sun, W. Hao, Q. Sun, A novel conjugate gradient method with generalized Armijo search for efficient training of feedforward neural networks, Neurocomputing 275 (2018) 308–316.
[14] Q. Liu, J. Liu, R. Sang, J. Li, T. Zhang, Q. Zhang, Fast neural network training on FPGA using quasi-Newton optimization method, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. (2018).
[15] S.F. McLoone, V.S. Asirvadam, G.W. Irwin, A memory optimal BFGS neural network training algorithm, in: Proceedings of the 2002 International Joint Conference on Neural Networks, Vol. 1, IEEE, 2002. 513–518
[16] H. Badem, A. Basturk, A. Caliskan, 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 (2017) 506–526.
[17] W. Sun, J. Han, J. Sun, Global convergence of nonmonotone descent methods for unconstrained optimization problems, J. Comput. Appl. Math. 146 (2) (2002) 89–98.
[18] L. Grippo, F. Lampariello, S. Lucidi, A nonmonotone line search technique for Newton's method, SIAM J. Numer. Anal. 23 (4) (1986) 707–716.
[19] C.C. Peng, G.D. Magoulas, Nonmonotone Levenberg–Marquardt training of recurrent neural architectures for processing symbolic sequences, Neural Comput. Appl. 20 (6) (2011a) 897–908.
[20] C.C. Peng, G.D. Magoulas, Nonmonotone BFGS-trained recurrent neural networks for temporal sequence processing, Appl. Math. Comput. 217 (12) (2011b) 5421–5441.
[21] C.C. Peng, G.D. Magoulas, Adaptive Nonmonotone Conjugate Gradient Training Algorithm for Recurrent Neural Networks, ICTAI, IEEE, 2007. 374–381
[22] C.C. Peng, G.D. Magoulas, Advanced adaptive nonmonotone conjugate gradient training algorithm for recurrent neural networks, Int. J. Artif. Intell. Tools 17 (05) (2008) 963–984.
[23] V. Plagianakos, G. Magoulas, M. Vrahatis, Determing nonmonotone strategies for effective training of multi-layer perceptrons, IEEE Trans. Neural Netw. 13 (6) (2002) 1268–1284.
[24] I.E. Livieris, P. Pintelas, A new class of nonmonotone conjugate gradient training algorithms, Appl. Math. Comput. 266 (2015) 404–413.
[25] I.E. Livieris, Improving the classification efficiency of an ANN utilizing a new training methodology, Informatics 6 (1) (2018).
[26] E. Dolan, J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.
[27] W.W. Hager, H. Zhang, A new active set algorithm for box constrained optimization, SIAM J. Optim. 17 (2) (2006) 526–557.
[28] Y.H. Dai, W.W. Hager, K. Schittkowski, H. Zhang, The cyclic Barzilai–Borwein method for unconstrained optimization, IMA J. Numer. Anal. 26 (3) (2006) 604–627.
[29] W.W. Hager, H. Zhang, A new conjugate gradient method with guaranteed descent and an efficient line search, SIAM J. Optim. 16 (1) (2005) 170–192.
[30] D. Dua, E.K. Taniskidou, UCI machine learning repository, 2017, http://archive.ics.uci.edu/ml.
[31] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, in: Proceedings of the IEEE International Conference on Neural Networks, IEEE, 1993. 586–591
[32] M.T. Hagan, M.B. Menhaj, Training feed-forward networks with the Marquardt algorithm, IEEE Trans. Neural Netw. 5 (6) (1994) 989–993.
[33] D. Nguyen, B. Widrow, Improving the learning speed of 2-layer neural network by choosing initial values of adaptive weights, Biol. Cybern. 59 (1990) 71–113.
[34] I. Anagnostopoulos, I. Maglogiannis, Neural network-based diagnostic and prognostic estimations in breast cancer microscopic instances, Med. Biol. Eng. Comput. 44 (9) (2006) 773–784.
[35] P. Liang, B. Labedan, M. Riley, Physiological genomics of Escherichia coli protein families, Physiol. Genom. 9 (2002) 15–26.
[36] A. Anastasiadis, G. Magoulas, M. Vrahatis, New globally convergent training scheme based on the resilient propagation algorithm, Neurocomputing 64 (2005) 253–270.
[37] P. Horton, K. Nakai, Better prediction of protein cellular localization sites with the *k*-nearest neighbors classifier, Intell. Syst. Mol. Biol. (1997) 368–383.
[38] L. Prechelt, PROBEN1-A set of benchmarks and benchmarking rules for neural network training algorithms, in: Technical Report 21/94, Fakultt fr Informatik, University of Karlsruhe, 1994.
[39] J. Yu, S. Wang, L. Xi, Evolving artificial neural networks using an improved PSO and DPSO, Neurocomputing 71 (2008) 1054–1060.
[40] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, 2016, pp. 1050–1059.
[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[42] Y. Gal, Z. Ghahramani, A theoretically grounded application of dropout in recurrent neural networks, Adv. Neural Inf. Process. Syst. (2016) 1019–1027.



Ioannis E. Livieris received his B.Sc., M.Sc., and Ph.D. degrees in Mathematics from the University of Patras, Greece in 2006, 2008, and 2012, respectively. He is currently an adjunct professor in Technological Educational Institute of Western Greece. His research interests include numerical optimization, neural networks, data mining and machine learning.

Panagiotis Pintelas is a professor of Computer Science with the Informatics Division of Department of Mathematics at Patras University, Greece. His research interests include software engineering, AI and ICT in education, machine learning and data mining. He was involved in or directed several dozens of National and European research and development projects.