Journal of Machine Learning Research 17 (2016) 1-40 Submitted 8/14; Revised 7/16; Published 9/16

Bounding the Search Space for Global Optimization of Neural Networks Learning Error: An Interval Analysis Approach

Stavros P. Adam ADAMSP@UPATRAS.GR
Computational Intelligence Laboratory
Department of Mathematics
University of Patras
GR-26110 Patras, Greece

George D. Magoulas GMAGOULAS@DCS.BBK.AC.UK
Department of Computer Science and Information Systems
Birkbeck College, University of London
Malet Street, London WC1E 7HX, UK

Dimitrios A. Karras DAKARRAS@TEISTE.GR
Department of Automation
Technological Educational Institute of Sterea Hellas
34400 Psahna, Evia, Greece

Michael N. Vrahatis VRAHATIS@MATH.UPATRAS.GR
Computational Intelligence Laboratory
Department of Mathematics
University of Patras
GR-26110 Patras, Greece

Editor: Kevin Murphy

Abstract

Training a multilayer perceptron (MLP) with algorithms employing global search strategies has been an important research direction in the field of neural networks. Despite a number of significant results, an important matter concerning the bounds of the search region—typically defined as a box—where a global optimization method has to search for a potential global minimizer seems to be unresolved. The approach presented in this paper builds on interval analysis and attempts to define guaranteed bounds in the search space prior to applying a global search algorithm for training an MLP. These bounds depend on the machine precision and the term "guaranteed" denotes that the region defined surely encloses weight sets that are global minimizers of the neural network's error function. Although the solution set to the bounding problem for an MLP is in general non-convex, the paper presents the theoretical results that help deriving a box which is a convex set. This box is an outer approximation of the algebraic solutions to the interval equations resulting from the function implemented by the network nodes. An experimental study using well known benchmarks is presented in accordance with the theoretical results.

Keywords: neural network training, bound constrained global optimization, interval analysis, interval linear equations, algebraic solution

1. Introduction

Multi-layer perceptrons are feed forward neural networks featuring universal approximation properties used both in regression problems and in complex pattern classification tasks. Actually, an MLP is a means used to encode empirical knowledge about a physical phenomenon, i.e., a real world process. This encoding is done in terms of realizing a function F that is close enough to a target function $d = f(x)$ representing the underlying process. The target function is rarely formulated in analytical terms but it is defined as a set of input-output values $\{\mathrm{x}_l, \mathrm{d}_l\}$, $1 \leqslant l \leqslant p$, which is the training set resulting from observations of the underlying process.

More formally, an MLP is used to implement a model $F(\mathrm{x}, \mathrm{w})$ about a physical process which is a mapping $F : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^k$ whose values $F(\mathrm{x}_l, \mathrm{w})$ have minimum distance from their corresponding d_l, where $\mathrm{x}_l = (x_{1,l}, x_{2,l}, \ldots, x_{n,l})^\top$ and $\mathrm{w} = (w_1, w_2, \ldots, w_m)^\top$. Given that x_l are known, this minimum distance depends on the values of the weights w. This formulates a problem of parametric model identification on the basis of the training set composed of the inputs and their corresponding outputs. Determining the values of the weights is done by an iterative process commonly known as network training, which consists in minimizing some cost function $E : \mathbb{R}^k \rightarrow \mathbb{R}$ of the network output.

The dominant approach when training feedforward neural networks has been to adopt local search in the weight space using some gradient descent technique in order to minimize the output error function. Adaptation of the weights is done with the well known error back-propagation (BP) algorithm. A detailed review of BP and other training algorithms based on gradient descent can be found in Haykin (1999). However, local search techniques are, in the general case, unable to detect with certainty a global or a "good" local minimum, and consequently the corresponding minimizer, as they may stuck in "bad" local minima. Researchers in the area of neural networks have tried to overcome this defect by proposing efficient weight initialization techniques (Adam et al., 2014; Nguyen and Widrow, 1990). Moreover, a number of effective search strategies have been proposed (Magoulas et al., 1997, 1999) which concern definition or adaptation of relevant parameters of the search procedure such as the step-size, the momentum term etc.

The counter approach to local search is to use global search of the free parameter space. The techniques based on global optimization can be either exact or heuristic (Horst and Pardalos, 1995). Exact global optimization methods can be roughly defined as deterministic or stochastic (Pàl, 2010). Deterministic global search techniques perform exhaustive search of the weight space and they guarantee to locate a global minimizer (Tang and Koehler, 1994). However, often the time needed for this task is unacceptable, especially for real life applications, and usually it is achieved with excessive use of computational resources. This is mainly due to the size of the parameter space which grows with the number of free parameters (weights) of the neural networks. An alternative approach to deterministic search has been the use of stochastic global search methods which are based on random sampling of the search space adopting random or adaptive random search strategies. Such methods include Monte Carlo techniques (Brooks, 1958; Caflisch, 1998) and multistart methods (Boender et al., 1982). On the other hand, heuristic methods include simulated annealing (Engel, 1988; Kirkpatrick et al., 1983) and techniques based on evolutionary computation such as the genetic algorithm (GA) (Castillo et al., 2000; Montana and Davis, 1989), Particle Swarm Optimization (PSO) (Gudise and Venayagamoorthy, 2003; van den Bergh and Engelbrecht, 2000) and differential evolution algorithms (Ilonen et al., 2003).

Although these methods possess several desirable characteristics and have proven to be effective in several applications, they cannot guarantee—in the general case—that they are able to detect a global minimizer in the first run. Moreover, their performance, in terms of the overall search time, depends on the scale of the problem. To a large extent, this is due to inappropriate initial conditions because the search space is only heuristically known (Parsopoulos and Vrahatis, 2010). A common technique to reduce the uncertainty about the search space and enhance the efficiency of these methods is to define the region of the search space to be "as large as possible". A survey of global optimization techniques used for neural network training can be found in Duch and Korczak (1998) as well as in Plagianakos et al. (2001, 2006). Finally, it is worth noting that a number of research work has focused on combining local and global search in order to aleviate the disadvantages of local seach methods or in order to decrease the overall search time in the case of global search of the weight space (Bagirov et al., 2009; Caprani et al., 1993; Ng et al., 2010; Shang and Wah, 1996; Voglis and Lagaris, 2009; Xu, 2002).

A common characteristic to all global search procedures used for training MLPs is that the bounds of the search region are intuitively defined. This approach, often, proves to be inadequate to support training MLPs with global optimization methods. Hence, defining effective bounds of the search space is a cornerstone for using global search strategies in real life applications of feedforward neural networks. The approach presented in this paper is based on concepts of interval analysis and attempts to define guaranteed bounds of the search space prior to applying a global search procedure for neural network training. The term "guaranteed" denotes that the bounds of the region defined surely enclose weights that are global minimizers of the error function. Once the box is defined, it is up to the global optimization method to explore the region identified by the proposed approach in order to find the set of weights that minimize the error function of the multilayer perceptron.

The rest of the paper is organized as follows. Section 2 is devoted to a detailed description of the problem background and the motivation for the proposed approach. Section 3 presents the solution proposed to tackle this problem. In Section 4 we derive the theoretical results for bounding the feasible solution set of the problem and discuss these results. In Section 5 we validate the proposed method on real world problems and discuss the experimental results. Finally, Section 6 summarizes the paper with some concluding remarks.

2. Problem Formulation in a Global Optimization Context

Hereafter, we consider feed forward neural networks having 3 layers with n, h, and o nodes in the input, the hidden and the output layers, respectively. It has been proved that standard feedforward networks with only a single hidden layer can approximate any continuous function uniformly on any compact set and any measurable function to any desired degree of accuracy (Cybenko, 1989; Hornik et al., 1989; Kreinovich and Sirisaengtaksin, 1993; White, 1990). Hence, it has general importance to study 3-layer neural networks. On the other hand, as it will be shown, the assumption regarding the number of layers is not restrictive, given that the weights of each layer are treated with respect to the output range of the previous layer and the input domain of the next one. Hence, it is straightforward to extend this analysis to networks with multiple hidden layers. Moreover, despite the variety of activation functions found in the literature (Duch and Jankowski, 1999), in this paper we have retained the most common assumptions adopted for MLPs. This means that the nodes in the hidden layer are supposed to have a sigmoid activation function which may be one of the following:

$$\text{logistic sigmoid } : \sigma_1(net) = \frac{1}{1+e^{-\alpha\, net}}, \tag{1a}$$

$$\text{hyperbolic tangent } : \sigma_2(net) = \tanh(\beta\, net) = \frac{e^{\beta\, net} - e^{-\beta\, net}}{e^{\beta\, net} + e^{-\beta\, net}}, \text{ or} \tag{1b}$$

$$\sigma_2(net) = \frac{2}{1+e^{-2\,\beta\, net}} - 1,$$

where *net* denotes the input to a node and α, β are the slope parameters of the sigmoids. Hereafter, let us assume that $\alpha = \beta = 1$ and note that this assumption has no effect on the theoretical analysis presented in this paper. Nevertheless, for application purposes using different values for the slope parameters has an affect on the results obtained, as it will be explained later in Subsection 3.2. In addition to the previous assumptions, let us suppose that the nodes in the output layer may have either one of the above sigmoid activation functions or the linear one. Finally, we assume that all nodes in the hidden layer use the same activation function and the same stands for the output layer nodes.

With respect to the above assumptions the function computed by a feed forward neural network can be defined as $F : \mathbb{R}^n \times \mathbb{R}^{(n+1)h} \times \mathbb{R}^{(h+1)o} \to \mathbb{R}^o$, where $F(\mathrm{X}, \mathrm{W}_1, \mathrm{W}_2)$ denotes the actual network output, $\mathrm{X} \in \mathbb{R}^n$ refers to the input patterns, $\mathrm{W}_1 \in \mathbb{R}^{(n+1)h}$ is the matrix of the input-to-hidden layer connection weights and biases, and $\mathrm{W}_2 \in \mathbb{R}^{(h+1)o}$ denotes the matrix of the hidden-to-output layer connection weights and biases. Moreover, let us denote $\mathrm{T} \in \mathbb{R}^o$ the target output of the function approximated by the network. Obviously, if any of the parameters, $\mathrm{X}, \mathrm{W}_1, \mathrm{W}_2$, and T is interval valued then the network is an interval neural network (Hu et al., 1998).

The process of neural networks training aims to solve the following optimization problem:

$$\arg\,\min\nolimits_{\mathrm{W}_1 \in \mathbb{R}^{(n+1)h}, \mathrm{W}_2 \in \mathbb{R}^{(h+1)o}} \|F(\mathrm{X}, \mathrm{W}_1, \mathrm{W}_2) - \mathrm{T}\|_q \tag{2}$$

for some appropriate norm $\|\cdot\|_q$, considering that the ideal minimum of $\| F(\mathrm{X}, \mathrm{W}_1, \mathrm{W}_2) - \mathrm{T} \|_q$ is 0, which means that $F(\mathrm{X}, \mathrm{W}_1, \mathrm{W}_2) = \mathrm{T}$.

As said above, training the network is defining the values of the elements of the matrices W_1 and W_2 that solve the previous unconstrained minimization problem. The domain of these weights is not known in advance and so, in practice, the global optimization procedures used to solve this problem search for the global minimum in a box $\mathbb{D}_{\mathrm{w}} \subset \mathbb{R}^{(n+1)h} \times \mathbb{R}^{(h+1)o}$. This means that the initial unconstrained global optimization problem is arbitrarily converted to an ad-hoc bound constrained optimization one. The bounds of this box and therefore its size are defined intuitively. Such a choice may lead to unsuccessful training or give a computationally expensive solution that is impractical to use in real life problems. The argument that the box $\mathbb{D}_{\mathrm{w}}$ is intuitively defined is supported by a number of examples found in the literature.

For instance, Saraev (2012) gives an example of interval global optimization for training a network that approximates the function $z = 0.5 \sin\left(\pi x^2\right) \sin\left(2\pi y\right)$ and defines the initial box to be $[w]_0 = [-10, 10]^s$, where s is the number of unknown weights. The results reported in the paper indicate that the global optimization technique succeeded in training the network using this initial box. However, the author does not provide any justification on how the initial search box was defined.

Another example is given by the approach formulated by Jamett and Acuña (2006) to solve the problem of weight initialization for an MLP based on interval analysis. These researchers argue that their approach "solves the network weight initialization problem performing an exhaustive global search for minimums in the weight space by means of interval arithmetic. The global minimum is obtained once the search has been limited to the estimated region of convergence." For the experimental evaluation proposed in Jamett and Acuña (2006) the interval weights are initially defined as wide as necessary, with widths up to 10^6.

In Hu et al. (1998) the authors consider a 3-layer interval neural network and they propose to adjust the interval weights with the use of the Newton interval method solving a system of nonlinear equations of the $h(n+o)$ unknown weights. This is possible under the hypothesis that the network is, what they call, a Type 1 interval neural network, i.e., a network for which the number lo of nonlinear equations, formed using all available patterns $\mathrm{x}^k, 1 \leqslant k \leqslant l$, satisfy the inequality $lo \leqslant h(n+o)$. When this inequality is not true they propose to use an interval branch-and-bound algorithm to solve the minimization problem related to the network training. However, the authors do not provide any kind of concrete experiment where interval global optimization is actually used for training the network.

In addition to the above, one should mention the problems related to the initialization and the bounds of the search region reported with heuristic and population based approaches when solving global optimization problems and more specifically when training MLPs. For instance, Helwig and Wanka (2008) showed that when PSO is used to solve problems with boundary constraints in high-dimensional search spaces, many particles leave the search space at the beginning of the optimization process. This commonly known problem of "particle explosion" (Clerc and Kennedy, 2002) for PSO in high-dimensional bounded spaces highlights the importance of defining bound handling procedures.

Moreover, when dealing with neural network training, one should take into account the remarks reported by Gudise and Venayagamoorthy (2003) regarding the selection of parameters for PSO. The authors report that during their experiments in order to determine the optimal PSO parameters they found that "A small search space range of $[-1, 1]$ did not provide enough freedom for the particles to explore the space and hence they failed to find the best position". So, they gradually increased the size of the search space from $[-1, 1]$ to $[-200, 200]$ and they observed that a search space range of $[-100, 100]$ was the best range for having optimal performance of the PSO algorithm. Finally, when no limits were set on the search space range, the convergence rate of PSO decreased and there appeared to be even cases of no convergence.

According to the above examples, it is obvious that the outcome of training an MLP with global optimization is unpredictable as there is no information concerning the region where a global minimum of the error function is located. To the best of our knowledge this statement is true no matter the type of the global optimization procedure used. So, researchers and practitioners in the field proceed using random (intuitively defined) bounds which result in trial and error training strategies. This paper argues that it is possible to derive the bounds of the search region for this type of global optimization task. It will be shown that the box defined by these bounds is guaranteed to enclose some global minimizers of the network output error function. This permits to obtain a value for the global minimum which is at least equal to the "machine ideal global minimum", that is the best value for the global minimum that can be computed within the limits set by the machine precision. The results obtained in this paper apply to any type of global optimization procedure used to train an MLP.

3. Bounding the Weight Space Using Interval Analysis

The idea underlying our analysis is to consider the bounds of the range of each node in the output, the hidden and the input layer respectively. These bounds are implicit constraints imposed by the type of the activation function for the nodes in the output and the hidden layers. Moreover, explicit constraints are imposed by the values of the input patterns for the input layer. Performing local inversion of the activation function at the level of each node and using the constraints on its inputs imposed by the output of the previous layer results in formulating suitable linear interval equations. The subsequent analysis of the solution sets of these equations permits to derive the bounds of a box enclosing the region where some global minimizers of the network output error function are located. The approach proposed leads to defining a suitable approximation $\mathbb{D}_{\mathrm{w}}$ of the feasible set S for the error function of the network.

3.1 The Interval Analysis Formalism

Interval arithmetic has been defined as a means of setting lower and upper bounds on the effect produced on a computed quantity by different types of mathematical errors. The efficiency of methods built on interval arithmetic is related to these bounds, which need to be as narrow as possible. Thus, the major focus of interval analysis is to develop interval algorithms producing sharp solutions when computing numerical problems. A real interval, or interval number, is a closed interval $[a, b] \subset \mathbb{R}$ of all real numbers between (and including) the endpoints a and b, with $a \leqslant b$. The terms interval and interval number are used interchangeably. Whenever $a = b$ the interval is said to be degenerate, thin or even point interval. An interval $[a, b]$ where $a = -b$ is called a symmetric interval. Regarding notation, an interval $\boldsymbol{x}$ may be also denoted $[x]$, or $[\underline{x}, \overline{x}]$, or even $[x_L, x_U]$ where subscripts $_L$ and $_U$ stand for lower and upper bounds respectively. Notation for interval variables may be uppercase or lowercase (Alefeld and Mayer, 2000). Throughout this paper we will use the notation $[x]$ whenever we refer to an interval number, while notation $[\underline{x}, \overline{x}]$ will be used when explicit reference to the interval bounds is required. Moreover, an n-dimensional interval vector, that is a vector having n real interval components $([v_1], [v_2], \ldots, [v_n])^\top$, will be denoted $[\mathrm{v}]$ or even $[\underline{\mathrm{v}}, \overline{\mathrm{v}}]$. Finally, uppercase letters will be used for matrices with interval elements as for an $n \times m$ interval matrix $[\mathrm{A}] = ([a_{ij}])_{\substack{i=1,2,\ldots,n \\ j=1,2,\ldots,m}}$.

Generally, if $[x] = [\underline{x}, \overline{x}]$ is a real interval then the following notation is used:

$$\begin{aligned}
\mathrm{rad}([x]) &= (\overline{x} - \underline{x})\,/2, \ \text{is the radius of the interval } [x],\\
\mathrm{mid}([x]) &= (\overline{x} + \underline{x})/2, \ \text{is the midpoint (mean value) of the interval } [x],\\
|[x]| &= \max\{|\underline{x}|, |\overline{x}|\}, \ \text{is the absolute value (magnitude) of the interval } [x],\\
\mathrm{d}([\mathrm{x}]) &= \overline{x} - \underline{x}, \ \text{is the diameter (width) of the interval } [x],\\
\mathbb{IR}&, \ \text{denotes the set of closed real intervals},\\
\mathbb{IR}^n&, \ \text{denotes the set of } n\text{−dimensional vectors of closed real intervals.}
\end{aligned}$$

Let $\diamond$ denote one of the elementary arithmetic operators $\{+, -, \times, \div\}$ for the simple arithmetic of real numbers x, y. If $[x] = [\underline{x}, \overline{x}]$ and $[y] = [\underline{y}, \overline{y}]$ denote real intervals then the four elementary arithmetic operations are defined by the rule

$$[x] \diamond [y] = \{\, x \diamond y \,|\, x \in [x],\, y \in [y]\}.$$

This definition guarantees that $x \diamond y \in [x] \diamond [y]$ for any arithmetic operator and any values x and y. In practical calculations each interval arithmetic operation is reduced to operations between real machine numbers. For the intervals $[x]$ and $[y]$ it can be shown that the above definition produces the following intervals for each arithmetic operation:

$$\begin{aligned}
[x] + [y] &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}],\\
[x] - [y] &= [\underline{x} - \overline{y}, \overline{x} - \underline{y}],\\
[x] \times [y] &= \left\{\min\left(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\right), \max\left(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\right)\right\},\\
[x] \div [y] &= [x] \times \frac{1}{[y]}, \text{ with}\\
\frac{1}{[y]} &= \left[\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right], \text{ provided that } 0 \notin \left[\underline{y}, \overline{y}\right].
\end{aligned}$$

The usual algebraic laws of arithmetic operations applied to real numbers need to be reconsidered regarding finite arithmetic on intervals. For instance, a non-degenerate (thick) interval has no inverse with respect to addition and multiplication. So, if $[x], [y]$, and $[z]$ are non-degenerate intervals then,

$$\begin{aligned}
[x] + [y] = [z] &\nRightarrow [x] = [z] - [y],\\
[x] \times [y] = [z] &\nRightarrow [x] = [z] \times \frac{1}{[y]}.
\end{aligned}$$

The following sub-distributive law holds for non-degenerate intervals $[x], [y]$, and $[z]$,

$$[x] \times ([y] + [z]) \subseteq [x] \times [y] + [x] \times [z].$$

Note that the usual distributive law holds in some particular cases; if $[x]$ is a point interval, if both $[y]$ and $[z]$ are point intervals, or if $[y]$ and $[z]$ lie on the same side of 0. Hereafter, the multiplication operator sign × will be omitted as in usual algebraic expressions with real numbers. Interval arithmetic operations are said to be *inclusion isotonic* or *inclusion monotonic* or even *inclusion monotone* given that the following relations hold,

$$\begin{array}{ll}\text{if} \quad [a],[b],[c],[d] \in \mathbb{IR} \quad \text{and} \quad [a] \subseteq [b],\ [c] \subseteq [d] \\ \text{then} \quad [a] \diamond [c] \subseteq [b] \diamond [d], \quad \text{for} \quad \diamond \in \{+,-,\times,\div\}.\end{array}$$

The property of *inclusion isotony* for interval arithmetic operations is considered to be the fundamental principle of interval analysis. More details on interval arithmetic and its extensions can be found in Alefeld and Mayer (2000); Hansen and Walster (2004), and Neumaier (1990).

Standard interval functions φ are extensions of the corresponding real functions $F = \{\text{sin(),cos(),tan(),arctan(),exp(),ln(),abs(),sqr(),sqrt()}\}$ and they are defined via their range, i.e., $\varphi([x]) = \{\varphi(x) | x \in [x]\}$, for $\varphi \in F$, (Alefeld and Mayer, 2000). Given that, these real functions are continuous and piecewise monotone on any compact interval in their domain of definition, the values $\varphi([x])$ can be computed directly from the values at the bounds of $[x]$. For non-monotonic functions such as the trigonometric ones the computation is more complicated. For instance, $\sin([0,\pi]) = [0,1]$ which differs from the interval $[\sin(0), \sin(\pi)]$. In these cases, the computation is carried out using an algorithm (Jaulin et al., 2001). Finally, it must be noted that the standard interval functions are inclusion monotonic.

Let $f : D \subset \mathbb{R} \to \mathbb{R}$ be a real function and $[x] \subseteq D$ an interval in its domain of definition. The range of values of f over $[x]$ may be denoted by $R(f;[x])$ (Alefeld and Mayer, 2000) or simply $f([x])$. Computing the range $f([x])$ of a real function by interval analysis tools practically comes to enclosing the range $f([x])$ by an interval which is as narrow as possible. This is an important task in interval analysis which can be used for various reasons, such as localizing and enclosing global minimizers and global minima of f on $[x]$, verifying that $f([x]) \subseteq [y]$ for some given interval $[y]$, verifying the nonexistence of a zero of f in $[x]$ etc.

Enclosing the range of f over an interval $[x]$ is achieved by defining a suitable interval function $[f] : \mathbb{IR} \to \mathbb{IR}$ such that $\forall [x] \in \mathbb{IR},\ f([x]) \subset [f]([x])$. This interval function is called an inclusion function of f. What is important when considering an inclusion function $[f]$ is that it permits to compute a box $[f]([x])$ which is guaranteed to contain $f([x])$, whatever the shape of $f([x])$ (Jaulin et al., 2001). Note that the so-called natural inclusion function is defined if $f(x), x \in D$ is computed as a finite composition of elementary arithmetic operators $\{+,-,\times,\div\}$ and standard functions $\varphi \in F$ as above. The natural inclusion function of f is obtained by replacing the real variable x by an interval variable $[x] \subseteq D$, each operator or function by its interval counterpart and evaluating the resulting interval expression using the rules in the previous paragraphs. The natural inclusion function has important properties such as being inclusion monotonic and if f involves only continuous operators and continuous standard functions it is convergent (see Jaulin et al., 2001).

3.2 Weights of Hidden-to-output Layer Connections

Let us compute the bounds of the intervals of the weights for any output node. As we will see the procedure described herein does not depend on the number of inputs to the node but merely on the type of its activation function.

The output z_k of the k-th node in the output layer is given by

$$z_k = \sigma\left(\sum_{j=1}^{h} w_{kj} y_j + w_{kb}\right), \tag{3}$$

where y_j is the output of the j-th node in the hidden layer. During training the output z_k may be any value in the range of the sigmoid activation function σ used by the output layer nodes. So, $z_k \in [0,1]$ for the logistic sigmoid and $z_k \in [-1,1]$ for the hyperbolic tangent.

The activation functions defined in (1) are bijective and so they are invertible. Using the inverse σ^{-1} of the activation function σ on both sides of (3) gives:

$$\sigma^{-1}(z_k) = \sigma^{-1}\left(\sigma\left(\sum_{j=1}^{h} w_{kj} y_j + w_{kb}\right)\right)$$

and finally the equation,

$$\sum_{j=1}^{h} w_{kj} y_j + w_{kb} = \sigma^{-1}(z_k). \tag{4}$$

The inverse of the activation functions are

$$\begin{aligned} \text{for the logistic sigmoid} : \sigma_1^{-1} &= -\ln\frac{1-x}{x}, \text{ and} \\ \text{for the hyperbolic tangent} : \sigma_2^{-1} &= \operatorname{atanh}(x) = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right). \end{aligned}$$

The output of each activation function is an interval. So, the values of the inverse of the activation functions are,

$$\begin{aligned} \text{for the logistic sigmoid} : \sigma_1^{-1}([0,1]) &= [-\infty,+\infty], \\ \text{for the hyperbolic tangent} : \sigma_2^{-1}([-1,1]) &= [-\infty,+\infty]. \end{aligned}$$

For nodes with sigmoid activation functions when the output value is one of the exact bounds of the intervals $[0,1]$ or $[-1,1]$, then the corresponding node is saturated. This implies that connection weights take very large values. However, in practice this is rarely achieved and as Rumelhart et al. (1986) note: "The system cannot actually reach its extreme values of 1 or 0 without infinitely large weights. Therefore, in a practical learning situation in which the desired outputs are binary $\{0,1\}$, the system can never actually achieve these values. Therefore, we typically use the values of 0.1 and 0.9 as the targets, even though we will talk as if values of $\{0,1\}$ are sought." This consideration is adopted for the values of the output layer nodes in various practical pattern recognition applications of MLPs as well as in research papers such as Sprinkhuizen-Kuyper and Boers (1999); Yam and Chow (2001).

This means that a node with a sigmoid activation function becomes saturated with input values much greater than $-\infty$ and drastically smaller than $+\infty$, and its output is still considered to be 0 (or -1), and 1, respectively. Hence, as far as saturation of the node is concerned, for practical use instead of the interval $[-\infty,+\infty]$ one may consider a substitute $[-\mathfrak{b},\mathfrak{b}]$, where $\mathfrak{b} > 0$. In consequence, the node output is not any of the "ideal" intervals

$[0,1]$ or $[-1,1]$ but merely some approximation such as $[0+\varepsilon, 1-\varepsilon]$ or $[-1+\varepsilon, 1-\varepsilon]$ where ε depends on the value of $\mathfrak{b}$ and, obviously, it should satisfy the precision required for the problem.

Henceforth, the interval $[-\mathfrak{b}, \mathfrak{b}]$, for some $\mathfrak{b} > 0$, will be used instead of the $[-\infty, +\infty]$ for the input values of the sigmoid activation function of any node in the output, or in the hidden, layer. With this assumption the intervals resulting for the inverse of the activation functions of the output layer nodes are:

$$\text{for the logistic sigmoid : } \sigma_1^{-1}([0+\varepsilon, 1-\varepsilon]) = [-\mathfrak{b}_1, \mathfrak{b}_1], \tag{5a}$$

$$\text{for the hyperbolic tangent : } \sigma_2^{-1}([-1+\varepsilon, 1-\varepsilon]) = [-\mathfrak{b}_2, \mathfrak{b}_2]. \tag{5b}$$

So, $\mathfrak{b}_1$ and $\mathfrak{b}_2$ are such that $\sigma_1([-\mathfrak{b}_1, \mathfrak{b}_1]) = [0+\varepsilon, 1-\varepsilon] \triangleq [0,1]_\varepsilon$ and $\sigma_2([-\mathfrak{b}_2, \mathfrak{b}_2]) = [-1+\varepsilon, 1-\varepsilon] \triangleq [-1,1]_\varepsilon$.

With these results (4) implies that

$$\sum_{j=1}^{h} w_{kj} y_j + w_{kb} \in [-\mathfrak{b}, \mathfrak{b}], \tag{6}$$

where $\mathfrak{b} = \mathfrak{b}_1$ or $\mathfrak{b} = \mathfrak{b}_2$ depending on the type of the activation function used. Recall that in Section 2 we assumed that the slope parameters α and β of the sigmoids are considered to be equal to 1. If this assumption does not hold then it is easy to see that the bounds $\mathfrak{b}_1$ and $\mathfrak{b}_2$, defined here, need to be divided by α and β, respectively.

Taking into account the type of the activation function of the hidden layer nodes we have that $y_j \in [0,1]$ for the logistic sigmoid and $y_j \in [-1,1]$ for the hyperbolic tangent. However, we note that the above assumptions, regarding the approximations of the interval $[-\infty, +\infty]$, also hold for the sigmoid activation functions of the hidden layer nodes. So, in practical situations, we consider that $y_j \in [0,1]_\varepsilon$ and $y_j \in [-1,1]_\varepsilon$, where $[0,1]_\varepsilon \subset [0,1]$ and $[-1,1]_\varepsilon \subset [-1,1]$. Then, because of the inclusion monotonicity property of the interval arithmetic operators and the activation functions, in the analysis presented in this paper, for the interval equations we use the intervals $[0,1]$ and $[-1,1]$ instead of $[0,1]_\varepsilon$ and $[-1,1]_\varepsilon$, respectively. This means that any solution set or enclosure of a solution set determined, herein, for an interval equation using the wider intervals $[0,1]$ and $[-1,1]$ also constitute enclosures of the solution set of the same equation using the narrower ones, $[0,1]_\varepsilon$ and $[-1,1]_\varepsilon$.

Using all these results we may formulate the following two linear interval inequalities for the unknown weights of the hidden-to-output layer connections:

$$\sum_{j=1}^{h} [w_{kj}][0,1] + [w_{kb}] \leqslant [-\mathfrak{b}, \mathfrak{b}],$$

$$\sum_{j=1}^{h} [w_{kj}][-1,1] + [w_{kb}] \leqslant [-\mathfrak{b}, \mathfrak{b}].$$

Hence, the interval $[-\mathfrak{b}, \mathfrak{b}]$ determines an enclosure of the intervals $[w_{kj}]$, $1 \leqslant j \leqslant h$. However, as the aim of this paper is to bound the box containing the values of w_{kj} which solve

Equation (4), instead of solving the above inequalities we may, equivalently, consider the following equations and define approximations of the corresponding solution sets:

$$\sum_{j=1}^{h} [w_{kj}][0,1] + [w_{kb}] = [-\mathfrak{b}, \mathfrak{b}], \tag{7a}$$

$$\sum_{j=1}^{h} [w_{kj}][-1,1] + [w_{kb}] = [-\mathfrak{b}, \mathfrak{b}]. \tag{7b}$$

Before, examining solvability issues and the relation of the solutions to these equations with the problem at hand, let us discuss, hereafter, some issues concerning the parameter $\mathfrak{b}$ and its relation to ε.

3.2.1 On the definition of the parameters $\mathfrak{b}$ and ε

The previous analysis introduced the interval $[-\mathfrak{b}, \mathfrak{b}]$ for the input values of a sigmoid activation function. This interval may also be defined for other types of activation functions and so its bounds depend on the type of the activation function. In the following sections we will show that these bounds, actually, define the volume of the area in the network weight space where global minimizers of the network's error function are located. The analysis presented in this paper does not rely on some specific values of the bounds of the interval $[-\mathfrak{b}, \mathfrak{b}]$. Instead, the results obtained depend on the activation functions of the network nodes and the calculation precision set for the problem. So, they have $\mathfrak{b}$'s as parameters.

As noted above, for sigmoid activation functions, the interval $[-\mathfrak{b}, \mathfrak{b}]$ implicitly defines the intervals $[0,1]_\varepsilon$ or $[-1,1]_\varepsilon$ which approximate the intervals $[0,1]$ or $[-1,1]$, respectively. Conversely, by setting some specific value to ε and using the inverse of the activation functions, one is able to define a value for $\mathfrak{b}$, and in consequence the width of the interval $[-\mathfrak{b}, \mathfrak{b}]$. As this interval determines enclosures for the intervals $[w_{kj}]$, $1 \leqslant j \leqslant h$, it is obvious that its width determines the magnitudes of the corresponding dimensions of the minimizers' search area. Here, this search area is considered to be a hyper-box in the weight space, determined by the intervals $[-\mathfrak{b}, \mathfrak{b}]$, for all network nodes. On the other hand, selecting some specific value for ε determines the numerical precision used by the network output for solving the problem at hand. Therefore, an interval $[-\mathfrak{b}, \mathfrak{b}]$ defines some ε inducing some numerical precision for the problem, and vice-versa. The smaller the area defined by small values of $\mathfrak{b}$ the bigger the values of ε and so the lower the precision. Conversely, the highest the precision for computing the network outputs the smaller the values of ε and so the larger the search area defined by larger $\mathfrak{b}$'s.

This problem can be formulated in the following terms: 'How large can ε be in order to minimize $\mathfrak{b}$ and in consequence the search area of global minimizers without compromising the numerical precision for solving the problem'. Hence, some tradeoff between $\mathfrak{b}$ (volume of the search area) and ε (numerical precision) seems to be required here. The impact of selecting a value for $\mathfrak{b}$, or equivalently for ε, on the numerical precision of the problem mainly concerns the accuracy required for the network output in order to match the problem requirements. In a classification context and for practical applications, this issue is probably insignificant given that, typically, for such problems an approximate response of the order of

$\varepsilon = 0.1$ is sufficient for the classification task. On the other hand, for function approximation problems where accuracy of the network output is really a prerequisite, the output nodes use linear activation functions as for these kind of problems networks perform regression tasks (Hornik et al., 1989; White, 1990). As we will see, hereafter, such approximation assumptions do not apply to linear activation functions. Moreover, for hidden nodes using sigmoid activation functions, as we will see, approximating the intervals $[0, 1]$ and $[-1, 1]$ by $[0, 1]_\varepsilon$ and $[-1, 1]_\varepsilon$, respectively, where ε is of the order of the machine precision, is sufficient for carrying out the neural computation. Actually, such an approximation is within the numerical precision adopted for neural computations in various contexts.

The problem of numerical precision pertaining neural computations has been addressed by several researchers. For instance, some of the earlier research include the work of Hoehfeld and Fahlman (1992) who studied the possibility of training a network using the cascade-correlation algorithm in a limited numerical precision learning context. Moreover, Holi and Hwang (1993) addressed the question of the precision required in order to implement neural network algorithms on low precision hardware. The authors performed a theoretical analysis of the error produced by finite precision computation and investigated the precision that is necessary in order to perform successful neural computations both for training a multilayer perceptron with back-propagation as well as for forward activation and retrieving. A more recent work (Draghici, 2002) deals with the capabilities of neural networks using weights with limited precision and the work of Vassiliadis et al. (2000) states that a representation error of 2^{-10} provides a limit for a safe approximation error in neural functions computations. Such an error is within the limits of machine precision and double precision arithmetic. The most recent research focuses on the numerical precision required for deep neural networks and their hardware implementation (Courbariaux et al., 2014; Gupta et al., 2015; Judd et al., 2016).

The precision level used for a neural network and the problem at hand is a matter that needs to be addressed prior to any implementation, either software or hardware. Following the above literature, in this paper, we may state that double precision floating point arithmetic proposed by the IEEE (754–2008) standard (Higham, 2002; IEEE, 2008) provides a very good precision level as required by modern neural computation applications and hardware implementations. Hence, for the experimental evaluation of the proposed approach the value of the machine epsilon, commonly known as double precision epsilon ($\varepsilon = 2.2204 \times 10^{-16}$) is retained in this paper. This value of ε is the precision (eps) used for the computing environment MATLAB (MATLAB-Documentation, 2014) as well as for the majority of the programming languages according to IEEE (754–2008) standard. As a consequence we have the following values for $\mathfrak{b}$:

$$\text{for the logistic sigmoid} : \mathfrak{b}_1 = 36.0437,$$
$$\text{for the hyperbolic tangent} : \mathfrak{b}_2 = 18.3684.$$

Note that, under the specific assumptions of Rumelhart et al. (1986) the intervals resulting for the inverse of the activation functions of the output layer nodes are,

$$\text{for the logistic sigmoid} : \sigma_1^{-1}([0.1, 0.9]) = [-2.1972, 2.1972],$$
$$\text{for the hyperbolic tangent} : \sigma_2^{-1}([-0.9, 0.9]) = [-1.4722, 1.4722].$$

The analysis presented in this subsection relies on the assumption that the activation function of the output nodes is the logistic sigmoid or the hyperbolic tangent. However, when an MLP is used for function approximation or regression then usually linear activations are used in the output nodes. In order to establish a unique formalism for our analysis we need to define an interval of the type $[-\mathfrak{b}, \mathfrak{b}]$ for a pure linear activation function of the network's output nodes.

The sample of the input data, by default, defines an interval $[\underline{z}_k, \overline{z}_k]$ for the range of values of the k-th output node. A pure linear activation function f_{lin} is invertible in its domain. So, using inversion we obtain an interval of the type $[a_k, b_k] = f_{lin}^{-1}([\underline{z}_k, \overline{z}_k])$ for the input values of the linear activation function. Seemingly, such an interval is not symmetric and so, in order to keep pace with the previous assumptions, we extend it to obtain a symmetric interval of the type $[-\mathfrak{b}, \mathfrak{b}]$. This is achieved by setting $\mathfrak{b}_{3,\mathrm{k}} = \max\{|a_k|, |b_k|\}$ and considering the interval $[-\mathfrak{b}_{3,\mathrm{k}}, \mathfrak{b}_{3,\mathrm{k}}]$ instead of $[a_k, b_k]$ which actually permits to adopt the above assumptions and the ensuing conclusions. It is obvious that, during training it depends on the optimization procedure to narrow the search space back to the initial interval $[a_k, b_k]$.

Note that, if sampling of the input space is performed with a distribution which is close to the underlying distribution of the input space then the sample data are representative of the real data and hence the output of any node is reliably set by the values derived from the patterns. If sampling of the input space is not correct then the input data will provide erroneous dimensions of the search area. However, this unfortunate outcome will happen whatever the network or the training algorithm used.

The above considerations are necessary in order to tackle the hidden-to-output connection weights using Equations (7a) and (7b) as the unique formalism for the problem. Hereafter, whatever the activation function of the output nodes its range of values will be considered a symmetric interval of the form $[-\mathfrak{b}, \mathfrak{b}]$.

3.2.2 On the Solution of the Interval Equations

In interval analysis, the formal notation $[\mathrm{A}]\,[\mathrm{x}] = [\mathrm{b}]$ (or $[\mathrm{A}]\,\mathrm{x} = [\mathrm{b}]$) of an interval linear system, where $[\mathrm{A}] \in \mathbb{IR}^{m\times n}$ is an m-by-n matrix of real intervals and $[\mathrm{b}] \in \mathbb{IR}^{m}$ is an m-dimensional vector of real intervals, denotes different interval problems (Shary, 1995). Respectively, the solution or solution set to this interval linear system has been defined in a number of possible ways (Shary, 1997). The paper uses the so-called *algebraic solution*, which is an interval vector $[\mathrm{x}^a]$ such that, when substituting into $[\mathrm{A}]\,[\mathrm{x}] = [\mathrm{b}]$ and executing all interval arithmetic operations, results in the valid equality $[\mathrm{A}]\,[\mathrm{x}^a] = [\mathrm{b}]$. Later, in Section 4, we will show the implication of an *algebraic solution* in deriving an outer approximation of the solutions of Equations (7a) and (7b). For a theoretical justification of this choice see Shary (1996, 2002) and references therein. An algebraic solution for the bounding of the weight space is an interval vector of the form $[\mathrm{w}_k^a] = ([w_{k1}^a], [w_{k2}^a], \ldots, [w_{kh}^a], [w_{kb}^a])^\top$, such that each of the relations,

$$[w_{k1}^a][0,1] + [w_{k2}^a][0,1] + \cdots + [w_{kh}^a][0,1] + [w_{kb}^a][1,1] = [-\mathfrak{b}, \mathfrak{b}],$$
$$[w_{k1}^a][-1,1] + [w_{k2}^a][-1,1] + \cdots + [w_{kh}^a][-1,1] + [w_{kb}^a][1,1] = [-\mathfrak{b}, \mathfrak{b}],$$

corresponding to (7a) and (7b), respectively, is a valid interval equality.

A suitable approximation of the feasible solution set to the training problem (2) can be derived considering the bounds of the interval components of the algebraic solution to each of the Equations (7a) and (7b). In the following section we are defining the bounds of an algebraic solution to each of the aforementioned equations. This is accomplished by taking into account the specific type of these equations and considering the algebraic solutions (Shary, 1997) to these equations. Before proceeding to defining these algebraic solutions let us consider two issues; one concerning solvability of Equations (7a) and (7b) and the other dealing with effectively solving these equations.

Solvability of the above Equations (7a) and (7b) can be studied either considering each equation as an $1 \times (h+1)$ system and using the results in Rohn (2003), or applying the conclusions of Ratschek and Sauer (1982). Using the latter in our case is more appropriate given that Ratschek and Sauer (1982) have considered the concept of the algebraic solution to an interval linear equation (see also Shary, 1996). Hence, the conclusions concerning the solvabilty of a single interval equation of the general form $\boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 + \cdots + \boldsymbol{A}_n\boldsymbol{x}_n = \boldsymbol{B}$, as denoted and studied in Ratschek and Sauer (1982), apply also in our case. The hypotheses of Ratschek and Sauer (1982, Theorem 1) are satisfied and so the above Equations (7a) and (7b) are solvable having eventually multiple solutions.

Solving a linear interval equation or defining an optimal enclosure of the solution set is a matter addressed by several researchers; see for example Beaumont (1998); Hansen (1969); Hansen and Sengupta (1981); Jansson (1997); Kreinovich et al. (1993); Neumaier (1986); Rohn (1989, 1995); Shary (1995) and extensive references therein. However, there is quite a little work done on solving an underdetermined linear interval equation. To the best of our knowledge, for this matter Neumaier (1986) refers to the work of Rump (1983) while the work of Popova (2006) proposes an algorithm which improves the approach used in Hölbig and Krämer (2003) and Krämer et al. (2006). In addition, INTLAB (Rump, 1999) provides an algorithm for solving underdetermined linear systems of interval equations which, however, does not seem to converge in all cases.

Finally, note that Ratschek and Sauer (1982) examined in detail the convexity of the solution set of the equation $\boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 + \cdots + \boldsymbol{A}_n\boldsymbol{x}_n = \boldsymbol{B}$ if this equation is solvable. According to Ratschek and Sauer (1982, Theorem 6) the solution set to each of the Equations (7a) and (7b) is not convex. However, the approach proposed in the next section results in defining an outer approximation which is a convex set.

3.3 Weights of Input-to-hidden Layer Connections

The output of the j-th node in the hidden layer used in the previous subsection is defined by

$$y_j = \sigma\left(\sum_{i=1}^{n} w_{ji}x_i + w_{jb}\right), \tag{8}$$

where x_i is the i-th input, i.e., the output of the i-th node in the input layer. Here, let us assume that scaling is applied (Bishop, 1995) to the neural network inputs x_i in order to facilitate training (LeCun, 1993), and so $x_i \in [0,1]$ or $x_i \in [-1,1]$. Moreover, the output y_j may be any value in the range of the sigmoid activation function σ used by the hidden layer nodes. So, $y_j \in [0,1]$ for the logistic sigmoid and $y_j \in [-1,1]$ for the hyperbolic tangent.

All these hypotheses suggest that defining the intervals of the weights of the input-to-hidden layer connections is a problem formulated exactly as the problem of defining the weights of the hidden-to-output layer connections of the previous subsection. In consequence, we formulate the following equations:

$$\sum_{i=1}^{n} [w_{ji}]\,[0,1] + [w_{jb}] = [-\mathfrak{b}, \mathfrak{b}]\,, \tag{9a}$$

$$\sum_{i=1}^{h} [w_{ji}]\,[-1,1] + [w_{jb}] = [-\mathfrak{b}, \mathfrak{b}]\,. \tag{9b}$$

The same arguments as those in the previous subsection apply concerning the solvability of Equations (9a) and (9b). In addition, solving these equations and defining outer approximations of the corresponding solution sets is covered by the same analysis as in the previous subsection.

4. An Algebraic Solution to Bounding the Weight Space

For the neural network training problem defined in (2), the solution set of each of the resulting Equations (7a), (7b) and (9a), (9b) is extremely difficult to be described analytically. Hence, in this section, we derive an outer approximation (an enclosure) of an algebraic solution, that is more practical to define and use. This enclosure is a convex polytope, depending on the MLP architecture, and more specifically on the type of the activation function of the network nodes, and is derived under the assumption that the neural network inputs x_i are scaled in the interval $[0,1]$ or $[-1,1]$. First, the theoretical results are derived and then the conditions for their applicability are discussed.

4.1 Theoretical Results

As seen in Subsection 3.2, above, an algebraic solution exists for each of the Equations (7a), (7b) and (9a), (9b). Here, given an algebraic solution $[\mathrm{w}^a]$ we proceed with computing limit values for the bounds of the interval components of $[\mathrm{w}^a]$. In the sequel, given these limit values we derive outer approximations of the solutions of Equations (7a), (7b) and (9a), (9b).

In order to simplify their representation, we define the outer approximations of the solutions as convex sets or convex polytopes. However, what really matters with these outer approximations for each of the equations is that, they enclose the solutions containing the connection weights which are the global minimizers of the problem defined by (2).

Theorem 1 *Consider the equation,*

$$[w_1]\,[0,1] + [w_2]\,[0,1] + \cdots + [w_n]\,[0,1] + [w_b]\,[1,1] = [-\mathfrak{b}, \mathfrak{b}]\,. \tag{10}$$

Then, whenever an algebraic solution is regarded for this equation, the following statements are valid:

(i) For any variable $[w_i] = [\underline{w}_i, \overline{w}_i]\,, 1 \leqslant i \leqslant n$ the maximum possible value of $\overline{w}_i$ is $2\mathfrak{b}$ or the minimum possible value of $\underline{w}_i$ is $-2\mathfrak{b}$.

(ii) For the variable $[w_b] = [\underline{w}_b, \overline{w}_b]$ the maximum possible value of $\overline{w}_b$ is $\mathfrak{b}$ and the minimum possible value of $\underline{w}_b$ is $-\mathfrak{b}$.

Proof First, let us note that for the product $[x, y]\,[0, 1]$ the following equalities are valid depending on the values of x and y:

$$[x, y]\,[0, 1] = [0, y]\,, \text{ if } 0 \leqslant x \leqslant y, \tag{11a}$$

$$[x, y]\,[0, 1] = [x, 0]\,, \text{ if } x \leqslant y \leqslant 0, \tag{11b}$$

$$[x, y]\,[0, 1] = [x, y]\,, \text{ if } x \leqslant 0 \leqslant y. \tag{11c}$$

In addition let $[t_j]$ denote the product $[w_j]\,[0, 1]$, for $1 \leqslant j \leqslant n$. Then, with respect to the above Equalities (11) the sum of all terms $[t_j]$ of the left hand side of Equation (10), that is, $[t_1] + [t_2] + \cdots + [t_n]$ may be written as the sum of three intervals of the form $[0, s_k] + [-s_l, 0] + \left[-s^L_m, s^H_m\right]$ defined as follows:

$$[t_1] + [t_2] + \cdots + [t_k] = [0, s_k]\,, \qquad 0 \leqslant k \leqslant n, \tag{12a}$$

$$[t_1] + [t_2] + \cdots + [t_l] = [-s_l, 0]\,, \qquad 0 \leqslant l \leqslant n, \tag{12b}$$

$$[t_1] + [t_2] + \cdots + [t_m] = \left[-s^L_m, s^H_m\right], \qquad 0 \leqslant m \leqslant n, \tag{12c}$$

with $k + l + m = n$, $s_k \geqslant 0$, $s_l \geqslant 0$ and $s^L_m, s^H_m \geqslant 0$.

Statement (i)

Let i be an index such that for the interval $[\underline{w}_i, \overline{w}_i]$ we have $\overline{w}_i > 0$. We will show that $\overline{w}_i \not> 2\mathfrak{b}$. Excluding the term $[t_i]$ let us consider that the above integers k, l, and m are defined for the sum $[t_1]+[t_2]+\cdots+[t_{i-1}]+[t_{i+1}]+\cdots+[t_n]$. This means that, $0 \leqslant k \leqslant n-1$, $0 \leqslant l \leqslant n-1$, $0 \leqslant m \leqslant n-1$, and $k+l+m = n-1$. In addition, $[t_i] = \left[\underline{t}_i, \overline{t}_i\right]$ with $\overline{t}_i = \overline{w}_i$ and

$$\underline{t}_i = \begin{cases} 0 & \text{if } 0 \leqslant \underline{w}_i \\ \underline{w}_i & \text{if } \underline{w}_i < 0. \end{cases}$$

Then Equation (10) becomes,

$$[0, s_k] + [-s_l, 0] + \left[-s^L_m, s^H_m\right] + [\underline{t}_i, \overline{w}_i] + [\underline{w}_b, \overline{w}_b] = [-\mathfrak{b}, \mathfrak{b}]\,,$$

or even,

$$\left[-s_l - s^L_m - |\underline{t}_i| + \underline{w}_b, s_k + s^H_m + \overline{w}_i + \overline{w}_b\right] = [-\mathfrak{b}, \mathfrak{b}]\,.$$

This interval equation translates to:

$$s_k + s^H_m + \overline{w}_i + \overline{w}_b = \mathfrak{b}, \tag{13a}$$

$$-s_l - s^L_m - |\underline{t}_i| + \underline{w}_b = -\mathfrak{b}. \tag{13b}$$

Furthermore, let us suppose that $\overline{w}_i > \mathfrak{b}$. Given that $s_k + s^H_m \geqslant 0$ then (13a) holds true if $\overline{w}_b < 0$ and $\overline{w}_b \leqslant \mathfrak{b} - (s_k + s^H_m + \overline{w}_i)$. Setting $\overline{w}_i = \mathfrak{b} + x$, where $x > 0$, then the previous inequality gives that $\overline{w}_b \leqslant \mathfrak{b} - (s_k + s^H_m + \mathfrak{b} + x)$, or else, $\overline{w}_b \leqslant -(s_k + s^H_m) - x < 0$.

On the other hand $[\underline{w}_b, \overline{w}_b]$ is a valid interval if $\underline{w}_b \leqslant \overline{w}_b$. It follows that $\underline{w}_b < 0$. Using (13b) we have that $\underline{w}_b = -\mathfrak{b} + s_l + s^L_m + |\underline{t}_i|$. From this equation we deduce that the minimum possible value for $\underline{w}_b$ is $-\mathfrak{b}$ attained when $s_l + s^L_m + |\underline{t}_i| = 0$. In consequence we have $-\mathfrak{b} \leqslant \underline{w}_b \leqslant \overline{w}_b \leqslant -(s_k + s^H_m) - x$. So, $-\mathfrak{b} \leqslant -(s_k + s^H_m) - x$. The last inequality gives that $-\mathfrak{b} + (s_k + s^H_m) \leqslant -x$, and finally, $x \leqslant \mathfrak{b} - (s_k + s^H_m)$. This suggests that $x \leqslant \mathfrak{b}$ and the maximum value is attained when $s_k + s_m^H = 0$. Hence, $\overline{w}_i \leqslant 2\mathfrak{b}$. Note that $\overline{w}_i = 2\mathfrak{b}$ when $[w_1] = [w_2] = \cdots = [w_{i-1}] = [w_{i+1}] = \cdots = [w_n] = [0]$, $\underline{w}_i = 0$, and $[w_b] = [-\mathfrak{b}, -\mathfrak{b}]$.

Now, suppose that $\underline{w}_i < -\mathfrak{b}$ and $[w_i] = [\underline{w}_i, \overline{w}_i]$. Then following the same reasoning as above we may infere that $-2\mathfrak{b} \leqslant \underline{w}_i$. In this case $\underline{w}_b$ is a positive number that diminishes the excess on the lower bound of the left hand side interval of the Equation (10) produced by $\underline{w}_i$. Note that $\underline{w}_i = -2\mathfrak{b}$ when $[w_1] = [w_2] = \cdots = [w_{i-1}] = [w_{i+1}] = \cdots = [w_n] = [0]$, $\overline{w}_i = 0$, and $[w_b] = [\mathfrak{b}, \mathfrak{b}]$.

Statement (ii)
In order to prove this statement let us suppose that $[\underline{w}_b, \overline{w}_b]$ is such that $\overline{w}_b > \mathfrak{b}$. In this case $\overline{w}_b = \mathfrak{b} + x$ with $x > 0$. Considering Equalities (12) we rewrite Equation (10) as,

$$[0, s_k] + [-s_l, 0] + \left[-s_m^L, s_m^H\right] + [\underline{w}_b, \overline{w}_b] = [-\mathfrak{b}, \mathfrak{b}],$$

which gives,

$$\begin{aligned} s_k + s_m^H + \overline{w}_b &= \mathfrak{b}, \\ -s_l - s_m^L + \underline{w}_b &= -\mathfrak{b}. \end{aligned}$$

The hypothesis $\overline{w}_b = \mathfrak{b} + x$ with $x > 0$ implies that $s_k + s_m^H = \mathfrak{b} - \overline{w}_b = -x$. This is impossible given that, by definition, $s_k + s_m^H \geqslant 0$. Hence, the only possibility for x is to have $x = 0$ which means that $\overline{w}_b \leqslant \mathfrak{b}$. Using the same reasoning we may infer that $-\mathfrak{b} \leqslant \underline{w}_b$. ■

The above theorem defines maximum and minimum values for the bounds of any interval parameter in Equations (7a) and (9a). Moreover, it suggests that in Equation (10) the interval $[w_b]$ acts as a term that cuts off the excess in the upper or the lower bound of the sum of the weight intervals. However, this is true only for either the upper or the lower bound but not for both bounds at the same time. The following Theorem 2 constitutes a generalization of *statement (i)* in the proof of Theorem 1.

Theorem 2 *Suppose that the interval vector* $\left([w_1^*], [w_2^*], \cdots, [w_n^*], [w_b^*]\right)^\top$ *is an algebraic solution of Equation* (10) *of Theorem 1. If* $S \subseteq I_n = \{1, 2, \cdots, n\}$ *denotes a set of indices* $\{i_1, i_2, \cdots, i_q\}$ *such that all* $[w_{i_j}^*] \neq 0,\ 1 \leqslant j \leqslant q$, *then only one of the following two relations can be true:*

$$(i) \quad -2\mathfrak{b} \leqslant \sum_{j=1}^{q} \underline{w}_{i_j}^* < -\mathfrak{b}, \ \text{or}$$

$$(ii) \quad \mathfrak{b} < \sum_{j=1}^{q} \overline{w}_{i_j}^* \leqslant 2\mathfrak{b}.$$

Proof The proof is obtained using the same reasoning as for Theorem 1. ■

Corollary 3 *Suppose that the interval vector* $\left([w_1^*], [w_2^*], \cdots, [w_n^*], [w_b^*]\right)^\top$ *is an algebraic solution of the equation* $[w_1][0,1] + [w_2][0,1] + \cdots + [w_n][0,1] + [w_b][1,1] = [-\mathfrak{b}, \mathfrak{b}]$. *Then for any interval* $[w_i^*]$ *both the upper and the lower bounds of this interval cannot exceed the bounds of the interval* $[-\mathfrak{b}, \mathfrak{b}]$.

Proof Suppose that there exists an index i such that for the interval $[\underline{w}_i^*, \overline{w}_i^*]$ both inequalities $\underline{w}_i^* < -\mathfrak{b}$ and $\mathfrak{b} < \overline{w}_i^*$ hold true. That is $[-\mathfrak{b}, \mathfrak{b}] < [\underline{w}_i^*, \overline{w}_i^*]$. Using the reasoning of Theorem 1 in order to reduce the excessive effect produced by $[\underline{w}_i^*, \overline{w}_i^*]$ on the left hand side of the interval of the equation we need to define $[\underline{w}_b, \overline{w}_b]$ with $\underline{w}_b > 0$ and $\overline{w}_b < 0$ which obviously is not a valid interval. Hence, it is impossible for some variable $[\underline{w}_i, \overline{w}_i]$ to have a solution such that $[-\mathfrak{b}, \mathfrak{b}] < [\underline{w}_i^*, \overline{w}_i^*]$. ■

Theorem 2 and Corollary 3, derived above, show the complexity of the set containing the algebraic solutions of Equations (7a) and (9a). However, the following Corollary defines an outer approximation of this solution set which is a convex set (polytope).

Corollary 4 *The algebraic solutions of Equation* (10) *of Theorem 1 are enclosed in the polytope defined by* $[-2\mathfrak{b}, 2\mathfrak{b}]^n \times [-\mathfrak{b}, \mathfrak{b}]$.

Proof Let the interval vector $\left([w_1^*], [w_2^*], \cdots, [w_n^*], [w_b^*]\right)^\top$ be a solution of Equation (10). Then for each $[w_j^*]$, $1 \leqslant j \leqslant n$ we have that $[w_j^*] < [-2\mathfrak{b}, 2\mathfrak{b}]$. Moreover $[w_b^*] < [-\mathfrak{b}, \mathfrak{b}]$. This proves the Corollary. ■

Theorem 5 *Consider the equation,*

$$[w_1]\,[-1,1] + [w_2]\,[-1,1] + \cdots + [w_n]\,[-1,1] + [w_b]\,[1,1] = [-\mathfrak{b}, \mathfrak{b}]\,. \tag{14}$$

Then, whenever an algebraic solution is regarded for this equation, for any variable $[w_i] = [\underline{w}_i, \overline{w}_i]\,, 1 \leqslant i \leqslant n$ *the maximum possible value of* $\overline{w}_i$ *is* $\mathfrak{b}$ *and the minimum possible value of* $\underline{w}_i$ *is* $-\mathfrak{b}$. *The same holds for the maximum possible value of* $\overline{w}_b$ *and the minimum possible value of* $\underline{w}_b$.

Proof First note that any interval multiplied by $[-1,1]$ gives a symmetric interval. Actually, $[x,y]\,[-1,1] = [-m,m]$, where $m = \max\{|x|,|y|\}$, for any interval $[x,y]$. If $[t_j]$ denotes $[w_j]\,[-1,1]$, for $1 \leqslant j \leqslant n$, then Equation (14) becomes $[t_1] + [t_2] + \cdots + [t_n] + [\underline{w}_b, \overline{w}_b] = [-\mathfrak{b}, \mathfrak{b}]$, with all $[t_j]$ being symmetric intervals. So, the following two equalities must hold:

$$\underline{t}_1 + \underline{t}_2 + \cdots + \underline{t}_n + \underline{w}_b = -\mathfrak{b}, \tag{15a}$$
$$\bar{t}_1 + \bar{t}_2 + \cdots + \bar{t}_n + \overline{w}_b = \mathfrak{b}, \tag{15b}$$

with $\underline{t}_j \leqslant 0$ and $0 \leqslant \bar{t}_j$ for $j = 1, 2, \ldots, n$.

Let i be an index such that for the interval $[\underline{w}_i, \overline{w}_i]$ we have $\underline{w}_i < -\mathfrak{b}$ and $\mathfrak{b} < \overline{w}_i$. Thus, the term $\left[\underline{t}_i, \bar{t}_i\right] > [-\mathfrak{b}, \mathfrak{b}]$.

Then (15a) is true if $\underline{t}_1 + \underline{t}_2 + \cdots + \underline{t}_{i-1} + \underline{t}_{i+1} + \cdots + \underline{t}_n = 0$ and $\underline{w}_b > 0$. In addition (15b) is true if $\bar{t}_1 + \bar{t}_2 + \cdots + \bar{t}_{i-1} + \bar{t}_{i+1} + \cdots + \bar{t}_n = 0$ and $\overline{w}_b < 0$. The above suggest that $[\underline{w}_b, \overline{w}_b]$ defined so it is not a valid interval. In consequence the interval $\left[\underline{t}_i, \bar{t}_i\right] \leqslant [-\mathfrak{b}, \mathfrak{b}]$ and so neither $b < \overline{w}_i$ nor $\underline{w}_i < -b$. Hence, $[\underline{w}_i, \overline{w}_i] \leqslant [-\mathfrak{b}, \mathfrak{b}]$.

For the interval $[\underline{w}_b, \overline{w}_b]$ suppose that $\underline{w}_b < -\mathfrak{b}$. This means that $\underline{w}_b = -\mathfrak{b} - x$ for some $x > 0$. Under these assumptions (15a) holds true if $\underline{t}_1 + \underline{t}_2 + \cdots + \underline{t}_n = x$, which is impossible given that $\underline{t}_1 + \underline{t}_2 + \cdots + \underline{t}_n < 0$. The reasoning is the same if we suppose that $\overline{w}_b > \mathfrak{b}$. Hence the minimum possible value of $\underline{w}_b$ is $-\mathfrak{b}$ and the maximum possible value of $\overline{w}_b$ is $\mathfrak{b}$. ■

The following corollary is a direct conclusion of Theorem 5.

Corollary 6 *The algebraic solutions of Equation* (14) *of Theorem 5 are enclosed in the hypercube* $[-\mathfrak{b}, \mathfrak{b}]^{n+1}$.

Proof Let the interval vector $([w_1^*], [w_2^*], \cdots, [w_n^*], [w_b^*])^\top$ be a solution of Equation (14). Then for each $[w_j^*]$, $1 \leqslant j \leqslant n$ we have that $[w_j^*] \leqslant [-\mathfrak{b}, \mathfrak{b}]$. The same stands for $[w_b^*]$. This proves the Corollary. ■

Theorem 5 and Corollary 6 help defining a convex outer approximation of the set of algebraic solutions to Equations (7b) and (9b).

4.2 Evaluation of the Theoretical Results

In the context of our minimization problem, solving the linear interval Equations (7a), (7b), (9a) and (9b) is equivalent to defining a solution in the tolerance solution set of each of these equations. This is achieved considering algebraic solutions for the following reasons. First, such an interval solution is proven to exist (Ratschek and Sauer, 1982) for each of these equations. Moreover, the algebraic solution of a linear interval system is a solution to the corresponding linear interval tolerance problem (Shary, 1996, Proposition 1). Finally, our aim is not to define exact bounds of the algebraic solution interval vector, but to identify the bounds of an enclosure of the algebraic solution.

Recall that Shary (2002) defined the tolerance (or tolerable) solution set of an interval equation $F([\mathrm{a}], \mathrm{x}) = [\mathrm{b}]$ as the set formed by all point vectors $\mathrm{x} \in \mathbb{R}^n$ such that for any $\tilde{\mathrm{a}} \in [\mathrm{a}]$ the image $F(\tilde{\mathrm{a}}, \mathrm{x}) \in [\mathrm{b}]$. This can be written in one of the following two forms:

$$\sum\nolimits_{tol} (F, [\mathrm{a}], [\mathrm{b}]) = \{\mathrm{x} \in \mathbb{R}^n \mid (\forall \tilde{\mathrm{a}} \in [\mathrm{a}])(\exists \tilde{\mathrm{b}} \in [\mathrm{b}]) (F(\tilde{\mathrm{a}}, \mathrm{x}) = \tilde{\mathrm{b}})\},$$
$$\sum\nolimits_{\subseteq} (F, [\mathrm{a}], [\mathrm{b}]) = \{\mathrm{x} \in \mathbb{R}^n \mid F([\mathrm{a}], \mathrm{x}) \subseteq [\mathrm{b}]\}.$$

The theoretical results of the previous subsection permit to define suitable convex polytopes containing the algebraic solutions of Equations (7a), (9a) and hypercubes for the algebraic solutions of Equations (7b), (9b). The algebraic solutions of the linear interval Equations (7a), (7b) and (9a), (9b), enclosed in their respective convex sets, are also solutions in the corresponding tolerance solution sets of these linear equations which may well be the activation functions of any output node. We are now interested in examining the relation of these convex sets with Equations (3) and (8). Let us consider the following non-linear interval equations:

$$\sigma_1 \left(\sum_{j=1}^{h} [w_{kj}][y_j] + [w_{kb}] \right) = [0, 1], \tag{16a}$$

$$\sigma_2 \left(\sum_{j=1}^{h} [w_{kj}][y_j] + [w_{kb}] \right) = [-1, 1], \tag{16b}$$

for any node k in the output layer using a sigmoid activation function, as well as the equations:

$$\sigma_1\left(\sum_{i=1}^{n}[w_{ji}][x_i]+[w_{jb}]\right)=[0,1], \tag{17a}$$

$$\sigma_2\left(\sum_{i=1}^{n}[w_{ji}][x_i]+[w_{jb}]\right)=[-1,1], \tag{17b}$$

defined for any node j in the hidden layer using a sigmoid activation function. Note that, $[y_j]$ denotes the interval of the output values of the j-th node in the hidden layer and $[x_i]$ denotes the interval of the values of the i-th component of the input patterns. The above equations are the interval counterparts of Equations (3) and (8).

We will show that the convex polytopes, derived in the previous subsection, constitute inner approximations of the tolerance solution sets for the interval Equations (16) and (17), namely the convex polytope for the Equations (16) and (17) having $[y_j]=[0,1]$ and $[x_i]=[0,1]$ while the hypercube does it, for the same equations, when $[y_j]=[-1,1]$ and $[x_i]=[-1,1]$.

First, let us formulate the following Proposition regarding Equations (16a), (16b) and (17a), (17b).

Proposition 7 *The tolerance solution set corresponding to each of the Equations* (16a)*,* (16b) *and* (17a)*,* (17b) *is unbounded.*

Proof Let us consider Equation (16a) and a real vector $\mathrm{w}^1\in\mathbb{R}^{h+1}$ such that

$$\sigma_1\left(\sum_{j=1}^{h}w_j^1[y_j]+w_{h+1}^1\right)\subseteq[0,1].$$

Then, there exists a real vector $\mathrm{w}^2\in\mathbb{R}^{h+1}$ for which $\mathrm{w}^1\dot{\leqslant}\mathrm{w}^2$, where the operator "$\dot{\leqslant}$" denotes the component-wise operator "$\leqslant$". If instead of the real vectors we consider their interval counterparts, that is, the degenerate interval vectors $[\mathrm{w}^1]$ and $[\mathrm{w}^2]$, it is obvious that $[\mathrm{w}^1]\subseteq[\mathrm{w}^2]$. Then, given that the sigmoid σ_1 is inclusion monotonic we have that

$$\sigma_1\left(\sum_{j=1}^{h}w_j^1[y_j]+w_{h+1}^1\right)\subseteq\sigma_1\left(\sum_{j=1}^{h}w_j^2[y_j]+w_{h+1}^2\right)\subseteq[0,1].$$

Hence, for any real vector $[\mathrm{w}^1]$ in the tolerance solution set there will always exist a "larger" vector $[\mathrm{w}^2]$ for which the output interval produced will enclose the output interval computed for $[\mathrm{w}^1]$. So, the tolerance solution set of Equation (16a) is unbounded. The proof is similar for the other equations. ■

It is obvious that the outcome of this Proposition is due to the asymptotic convergence of each of the sigmoids to its extreme values, i.e., to $0,1$ or to $-1,1$. Hence, if one aims in finding a suitable inner approximation of each tolerance solution set, then one should define an upper and a lower limit for the output of the corresponding sigmoid. Actually, this is how we proceeded in Subsection 3.2. Subsequently, the direct consequence of this approach is to consider the solution set of some linear interval equation suitably defined to take into account the input values of a node. So, for each of the non-linear Equations (16a) and (16b) of some node in the output layer two linear interval equations, namely (7a) and (7b), are defined depending on the type of the sigmoid used. In addition, for each of the non-linear interval Equations (17a) and (17b) of any node in the hidden layer the two linear interval equations derived are (9a) and (9b).

For any node with a sigmoid activation function there are two possible types of boxes, as defined by Corollaries 4 and 6, depending on the intervals of the input values. Here, let us examine the range of application of these boxes in terms of their usage in (16a), (16b) and (17a), (17b). The convex polytope $[-2\mathfrak{b}, 2\mathfrak{b}]^h \times [-\mathfrak{b}, \mathfrak{b}]$ is used whenever the input signals are in the interval $[0, 1]$. Then, from (16a) and (16b) we derive the following relations:

$$\sigma_1\left(\sum_{j=1}^{h} [-2\mathfrak{b}_1, 2\mathfrak{b}_1]\,[0,1] + [-\mathfrak{b}_1, \mathfrak{b}_1]\right) \subseteq [0,1],$$

$$\sigma_2\left(\sum_{j=1}^{h} [-2\mathfrak{b}_2, 2\mathfrak{b}_2]\,[0,1] + [-\mathfrak{b}_2, \mathfrak{b}_2]\right) \subseteq [-1,1].$$

A direct conclusion of these relations is that the convex polytope $[-2\mathfrak{b}_1, 2\mathfrak{b}_1]^h \times [-\mathfrak{b}_1, \mathfrak{b}_1]$ constitutes an inner approximation of the tolerance solution set of the non-linear interval equation (16a). The same stands for the box $[-2\mathfrak{b}_2, 2\mathfrak{b}_2]^h \times [-\mathfrak{b}_2, \mathfrak{b}_2]$ and the non-linear interval equation (16b). Obviously, the hypercubes $[-\mathfrak{b}_1, \mathfrak{b}_1]^{(h+1)}$ and $[-\mathfrak{b}_2, \mathfrak{b}_2]^{(h+1)}$ are used instead of the corresponding convex polytopes when the input signals to a node are in the interval $[-1, 1]$. The same reasoning applies to Equations (17a) and (17b).

Besides these conclusions other interesting results are derived, hereafter, for the above convex polytopes. First, it is easy to verify that the following two equations hold true:

$$\sum_{j=1}^{h} [-2\mathfrak{b}_1, 2\mathfrak{b}_1]\,[0,1] + [-\mathfrak{b}_1, \mathfrak{b}_1] = (2h+1)\,[-\mathfrak{b}_1, \mathfrak{b}_1]\,,$$

$$\sum_{j=1}^{h} [-2\mathfrak{b}_2, 2\mathfrak{b}_2]\,[-1,1] + [-\mathfrak{b}_2, \mathfrak{b}_2] = (2h+1)\,[-\mathfrak{b}_2, \mathfrak{b}_2]\,.$$

So, we have the relations:

$$\sigma_1\left((2h+1)\,[-\mathfrak{b}_1, \mathfrak{b}_1]\right) \subseteq [0,1],$$
$$\sigma_2\left((2h+1)\,[-\mathfrak{b}_2, \mathfrak{b}_2]\right) \subseteq [-1,1].$$

If we set $\lambda = \lambda(h)$, a small number, $0 < \lambda \ll 1$, depending on the number of inputs h of the node then we may write that

$$\sigma_1((2h+1)[-\mathfrak{b}_1, \mathfrak{b}_1]) = [0+\lambda, 1-\lambda] = [0,1]_\lambda \subseteq [0,1], \text{ and}$$
$$\sigma_2((2h+1)[-\mathfrak{b}_2, \mathfrak{b}_2]) = [-1+\lambda, 1-\lambda] = [-1,1]_\lambda \subseteq [-1,1].$$

In consequenece, recalling the definition of the intervals $[-\mathfrak{b}_1, \mathfrak{b}_1]$ and $[-\mathfrak{b}_2, \mathfrak{b}_2]$, in Subsection 3.2, it is obvious that the following two relations hold true:

$$[0,1]_\varepsilon = [0+\varepsilon, 1-\varepsilon] \subset [0+\lambda, 1-\lambda] = [0,1]_\lambda, \tag{18a}$$

$$[-1,1]_\varepsilon = [-1+\varepsilon, 1-\varepsilon] \subset [-1+\lambda, 1-\lambda] = [-1,1]_\lambda. \tag{18b}$$

In consequence we may state that, for any node in the output layer using a sigmoid activation function, having input signals in the interval $[0,1]$ and connection weights in the convex polytope $[-2\mathfrak{b}, 2\mathfrak{b}]^h \times [-\mathfrak{b}, \mathfrak{b}]$ then the range of output values are in the interval $[0,1]_\lambda$ or $[-1,1]_\lambda$ for some λ smaller than the precision ε used for the problem.

When the inputs to a node are in the interval $[-1,1]$ then using the box $[-\mathfrak{b}, \mathfrak{b}]^{(h+1)}$ we arrive to similar results and formulate a similar statement. Finally, note that, using the same reasoning we get exactly the same results for any node in the hidden layer bringing out the validity of the theoretical results when used in (17a) and (17b) corresponding to Equations (9a) and (9b).

To further advance this reasoning suppose that the real matrices $\mathrm{W}_1^s \in \mathbb{R}^{(n+1)h}$ and $\mathrm{W}_2^s \in \mathbb{R}^{(h+1)o}$ constitute a global minimizer of the minimization problem (2), i.e., the distance $\|F\left(\mathrm{X}, \mathrm{W}_1^s, \mathrm{W}_2^s\right) - \mathrm{T}\|_q \leqslant \varepsilon$, where ε is the machine precision or the best precision set for the problem. Also, suppose that p patterns are available for the problem, which means that $\mathrm{X} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_p\}$ and the corresponding target outputs are $\mathrm{T} = \{\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_p\}$. The input patterns are n-dimensional vectors while the target outputs are o-dimensional. For any input pattern x_l, where $1 \leqslant l \leqslant p$, let z_l be the o-dimensional network output for this pattern, that is, $\mathrm{z}_l = F(\mathrm{x}_l, \mathrm{W}_1^s, \mathrm{W}_2^s)$.

Now, if the norm q is such that $1 \leqslant q$ and considering that the total network output error is averaged over all patterns then we can set the network output error for the l-th pattern to be $(\sum_{k=1}^{o} |z_{k,l} - t_{k,l}|^q)^{1/q} \leqslant \varepsilon$. From this, for the k-th output node we may consider that $|z_{k,l} - t_{k,l}|^q \leqslant \sum_{k=1}^{o} |z_{k,l} - t_{k,l}|^q \leqslant \varepsilon^q$. Hence, $|z_{k,l} - t_{k,l}|^q \leqslant \varepsilon^q$, which gives that $|z_{k,l} - t_{k,l}| \leqslant \varepsilon$. In consequence, $t_{k,l} - \varepsilon \leqslant z_{k,l} \leqslant t_{k,l} + \varepsilon$. So, if $t_{k,l} = 0$ or $t_{k,l} = -1$ then $z_{k,l} \in [0, 0+\varepsilon]$ or $z_{k,l} \in [-1, -1+\varepsilon]$. Similarly, if $t_{k,l} = 1$ then $z_{k,l} \in [1-\varepsilon, 1]$. Hence, we may suppose that there exists some small positive constant, say λ, such that $0+\lambda \leqslant z_{k,l} \leqslant 0+\varepsilon$, or, $-1+\lambda \leqslant z_{k,l} \leqslant -1+\varepsilon$, and $1-\varepsilon \leqslant z_{k,l} \leqslant 1-\lambda$. Then, we may deduce that $[0+\varepsilon, 1-\varepsilon] \subset [0+\lambda, 1-\lambda]$ and $[-1+\varepsilon, 1-\varepsilon] \subset [-1+\lambda, 1-\lambda]$. This is exactly what has been derived above with relations (18). So, we deduce that the real matrices $\mathrm{W}_1^s \in \mathbb{R}^{(n+1)h}$ and $\mathrm{W}_2^s \in \mathbb{R}^{(h+1)o}$ supposed to constitute a global minimizer of the minimization problem (2) are, indeed, located in the corresponding convex polytope, as defined above. The reasoning is similar if q is the infinity norm.

Following the above discussion we consider that it is legitimate to argue that the convex polytopes identified by the proposed method are guaranteed, within the machine precision accuracy, to enclose some global minimizers of the network output error function.

4.3 Discussion on the Theoretical Results

The proposed approach operates under the assumption that the input data to a destination node are either in the interval $[0,1]$ or in $[-1,1]$. Typically, this happens in pattern recognition or classification applications; for example, this is the case for the interval used for coding the corresponding components of the input patterns if the weights concern input-to-hidden layer connections. For a weight of a hidden-to-output layer connection this interval is the range of values of the sigmoid activation function of the corresponding hidden node. So, for an input data point in $[0, 1]$, the initial interval for the value of the corresponding weight is $[-2\mathfrak{b}, 2\mathfrak{b}]$; otherwise, if the value of the input data point is in $[-1, 1]$ then the initial interval for the corresponding weight is considered to be $[-\mathfrak{b}, \mathfrak{b}]$. In all cases, the interval for the value of any bias weight is $[-\mathfrak{b}, \mathfrak{b}]$.

Depending on the type of the activation function parameter $\mathfrak{b}$ is denoted by $\mathfrak{b}_1$ for the logistic sigmoid, $\mathfrak{b}_2$ for the hyperbolic tangent, and $\mathfrak{b}_3$ for the pure linear activation function. Moreover, recall that the specific value in each case is implementation dependent being a function of the precision ε set for the problem. In this paper, we consider that ε is the machine epsilon corresponding to double precision and so the specific values for $\mathfrak{b}$ should be $\mathfrak{b} = \mathfrak{b}_1 \geqslant 36.05$ and $\mathfrak{b} = \mathfrak{b}_2 \geqslant 18.4$, while the specific value for $\mathfrak{b} = \mathfrak{b}_3$ is defined by the problem data. Table 1 summarizes these results for a 3-layer perceptron having n, h and o nodes in the input, the hidden and the output layers respectively, and two possible ranges, $[0, 1]$ and $[-1, 1]$, for the input data coding.

The convex sets in Table 1 are defined according to the above theoretical results and more specifically the conclusions of Corollaries 4 and 6. However, if one adopts a "broad interpretation" of Corollary 3 it would be possible to consider narrower intervals for nodes with inputs in the interval $[0, 1]$. So, instead of the interval $[-2\mathfrak{b}, 2\mathfrak{b}]$ one would rather consider the interval $[-\mathfrak{b}, \mathfrak{b}]$. Obviously with this argument the convex set defined, in this case, is no longer a polytope, in general terms, but rather a hypercube. However, we should note that this result is based on heuristic considerations, as the complexity of the structure of the solution sets of Equations (7a) and (9a) leaves no space for such a theoretical conclusion. The validity of this heuristic conjecture is tested in the following Section 5.

The values defined, previously, for $\mathfrak{b}$ and the subsequent volumes of the convex polytopes, though formally proven, are relatively large. So, they may be not convincing regarding the advantage offered to the global optimization based training by the proposed approach. Here, let us recall the assumption that, often, in practical applications, the output values of a node are expected to be in the interval $[0.1, 0.9]$ for the sigmoid activation function or in the interval $[-0.9, 0.9]$ for the hyperbolic tangent. Then according to Paragraph 3.2.1 the corresponding values for $\mathfrak{b}$, denoted here by $\mathfrak{b}^*$, should be $\mathfrak{b}_1^* \geqslant 2.2 \approx 2.1972$ and $\mathfrak{b}_2^* \geqslant 1.5 \approx 1.4722$. Despite the apparent disagreement of the intervals $[0.1, 0.9]$ and $[-0.9, 0.9]$ with the linear interval equations (10) and (14), respectively, the previous theoretical results are still valid. As an example, let us compute the bounds of the weights for the k-th output node having a sigmoid activation function with output in the interval $[0.1, 0.9]$ and inputs y_j in the interval $[0.1, 0.9]$ supposing that any node in the hidden layer has the logistic sigmoid activation function, as well. With these hypotheses Equation (7a) becomes

$$\sum_{j=1}^{h} [w_{kj}][0.1, 0.9] + [w_{kb}] = [-\mathfrak{b}^*, \mathfrak{b}^*], \tag{19}$$

where $\mathfrak{b}^*$ is the specific $\mathfrak{b}$ as noted previously. In addition, due to inclusion monotonicity $\sum_{j=1}^{h}[w_{kj}][0.1, 0.9] + [w_{kb}] \subseteq \sum_{j=1}^{h}[w_{kj}][0, 1] + [w_{kb}]$. On the other hand, at the expense of lower precision, we may have the equation $\sum_{j=1}^{h}[w_{kj}][0, 1] + [w_{kb}] = [-\mathfrak{b}^*, \mathfrak{b}^*]$. Then, the solution set defined for this specific form of Equation (7a) is a tolerance solution set of the linear interval equation (19). In consequence, the convex polytope $[-2\mathfrak{b}, 2\mathfrak{b}]^h \times [-\mathfrak{b}, \mathfrak{b}]$, for any $\mathfrak{b} = \mathfrak{b}_1^* \geqslant 2.2$, is a tolerance solution set of the linear interval equation (19), and constitutes an inner approximation of the tolerance solution set of the non-linear interval equation

$$\sigma_1 \left(\sum_{j=1}^{h} [w_{kj}][y_j] + [w_{kb}] \right) = [0.1, 0.9], \tag{20}$$

where $[y_j] = [0.1, 0.9]$ for $1 \leqslant j \leqslant h$. It is obvious that this reasoning applies to any linear interval equation, such as (7a), (7b), (9a) and (9b).

If the scaling assumption of the neural network inputs x_i does not hold (this may happen with function approximation or regression problems), it is still possible to solve the problem; however, the solution does not comply with the algebraic formalism adopted in the previous section. In this case, let us suppose that $x_i \in \left[\underline{x_i}, \overline{x_i}\right]$, for $i = 1, 2, \ldots, n$ where the value of the functional $\chi\left(\left[\underline{x_i}, \overline{x_i}\right]\right) \in [-1, 1]$ (see Ratschek and Sauer, 1982), which means that the bounds of the interval $\left[\underline{x_i}, \overline{x_i}\right]$ may have any value provided that $\underline{x_i} \leqslant \overline{x_i}$. For the bias we have that the input is $\chi([1, 1]) = 1$, so the hypotheses of Ratschek and Sauer (1982, Theorem 1) are satisfied and in consequence even in this case the Equations (9a) and (9b) have a solution. However, it is clear that the resulting equation does not comply with the formalism adopted in the previous subsection for the Equations (7a), (7b) and (9a), (9b). Although a detailed coverage of this issue is considered out-of-the-scope of this paper, it can briefly be stated that this situation can be dealt with by considering the work of Popova (2006) and the algorithm proposed therein.

5. Testing

In this section we give some concrete examples of the approach presented above. Our objective is to perform some tests, using well known benchmarks, in order to show that the convex sets, derived following the analysis of this paper, indeed enclose global minimizers of the cost function associated with the output of the MLP used for each problem. To this end, the interval global optimization software (GOP), provided by Pàl and Csendes (2009), is used.

The reason for adopting an interval global optimization procedure instead of some version of PSO or a GA-based approach is easily understood if one takes into account the fact that interval global optimization is a deterministic approach which is guaranteed to locate the best available minimizers and the interval for the corresponding global minimum. On the other hand, convergence capabilities of population based and heuristic approaches depend on a number of heuristically defined random parameters. Hence, a proof of convergence for these methods is either probabilistic (Bertsimas and Tsitsiklis, 1993; Eiben et al., 1991), or it relies on empirical results (Pedersen, 2010). This means that, by using a global search method of this class, it is practically impossible to verify that the boxes derived by the proposed approach indeed contain the best available minimizers.

Table 1: Approximation of the initial weight set

Activation function	Input Data Interval	
	[0,1]	[-1,1]
HL: logistic sigmoid	$[-2\mathfrak{b}_1, 2\mathfrak{b}_1]^{n*h} \times [-\mathfrak{b}_1, \mathfrak{b}_1]^h$	$[-\mathfrak{b}_1, \mathfrak{b}_1]^{(n+1)*h}$
OL: logistic sigmoid	$[-2\mathfrak{b}_1, 2\mathfrak{b}_1]^{h*o} \times [-\mathfrak{b}_1, \mathfrak{b}_1]^o$	$[-\mathfrak{b}_1, \mathfrak{b}_1]^{(h+1)*o}$
HL: logistic sigmoid	$[-2\mathfrak{b}_1, 2\mathfrak{b}_1]^{n*h} \times [-\mathfrak{b}_1, \mathfrak{b}_1]^h$	$[-\mathfrak{b}_1, \mathfrak{b}_1]^{(n+1)*h}$
OL: hyperbolic tangent	$[-2\mathfrak{b}_2, 2\mathfrak{b}_2]^{h*o} \times [-\mathfrak{b}_2, \mathfrak{b}_2]^o$	$[-\mathfrak{b}_2, \mathfrak{b}_2]^{(h+1)*o}$
HL: logistic sigmoid	$[-2\mathfrak{b}_1, 2\mathfrak{b}_1]^{n*h} \times [-\mathfrak{b}_1, \mathfrak{b}_1]^h$	$[-\mathfrak{b}_1, \mathfrak{b}_1]^{(n+1)*h}$
OL: linear	$[-2\mathfrak{b}_3, 2\mathfrak{b}_3]^{h*o} \times [-\mathfrak{b}_3, \mathfrak{b}_3]^o$	$[-\mathfrak{b}_3, \mathfrak{b}_3]^{(h+1)*o}$
HL: hyperbolic tangent	$[-2\mathfrak{b}_2, 2\mathfrak{b}_2]^{n*h} \times [-\mathfrak{b}_2, \mathfrak{b}_2]^h$	$[-\mathfrak{b}_2, \mathfrak{b}_2]^{(n+1)*h}$
OL: logistic sigmoid	$[-2\mathfrak{b}_1, 2\mathfrak{b}_1]^{h*o} \times [-\mathfrak{b}_1, \mathfrak{b}_1]^o$	$[-\mathfrak{b}_1, \mathfrak{b}_1]^{(h+1)*o}$
HL: hyperbolic tangent	$[-2\mathfrak{b}_2, 2\mathfrak{b}_2]^{n*h} \times [-\mathfrak{b}_2, \mathfrak{b}_2]^h$	$[-\mathfrak{b}_2, \mathfrak{b}_2]^{(n+1)*h}$
OL: hyperbolic tangent	$[-2\mathfrak{b}_2, 2\mathfrak{b}_2]^{h*o} \times [-\mathfrak{b}_2, \mathfrak{b}_2]^o$	$[-\mathfrak{b}_2, \mathfrak{b}_2]^{(h+1)*o}$
HL: hyperbolic tangent	$[-2\mathfrak{b}_2, 2\mathfrak{b}_2]^{n*h} \times [-\mathfrak{b}_2, \mathfrak{b}_2]^h$	$[-\mathfrak{b}_2, \mathfrak{b}_2]^{(n+1)*h}$
OL: linear	$[-2\mathfrak{b}_3, 2\mathfrak{b}_3]^{h*o} \times [-\mathfrak{b}_3, \mathfrak{b}_3]^o$	$[-\mathfrak{b}_3, \mathfrak{b}_3]^{(h+1)*o}$

- HL : Hidden Layer
- OL : Output Layer
- n, h and o are the number of nodes in the input, hidden and output layer respectively
- $\mathfrak{b}_1$ denotes the value of $\mathfrak{b}$ in the case of the logistic sigmoid activation function
- $\mathfrak{b}_2$ denotes the value of $\mathfrak{b}$ in the case of the hyperbolic tangent activation function
- $\mathfrak{b}_3$ denotes the value of $\mathfrak{b}$ in the case of the linear activation function

5.1 Experimental Setup

In order to perform this experimental verification, we follow the next three steps:

1. Consider an MLP which is typically used for solving the corresponding problem.

2. Write the MATLAB code implementing the natural inclusion function of the MLP mapping, (an example is given in Appendix A).

3. Use the GOP procedure to solve the minimization problem defined for this natural inclusion function, where the bounds for the weights are set by the convex polytope computed by the proposed approach.

Typically, GOP is used for solving a global optimization problem with bound constraints and performs an exhaustive search of the initially defined set of boxes. The criterion used to stop the search process is a tolerance level of the width of the interval computed as the value of the inclusion function. So, given that in this paper we are interested only in verifying the existence of global minimizers and not in effectively locating a global minimum, as this happens when training the MLP, we specify a large value for the tolerance level of the width of the inclusion function interval. Hence, we use GOP to obtain rough estimates of the interval enclosing the global minimum of the network's output error function.

Moreover, GOP uses gradient information of the objective function to perform branch-and-bound computations. In order to comply with this requirement we use the Mean-

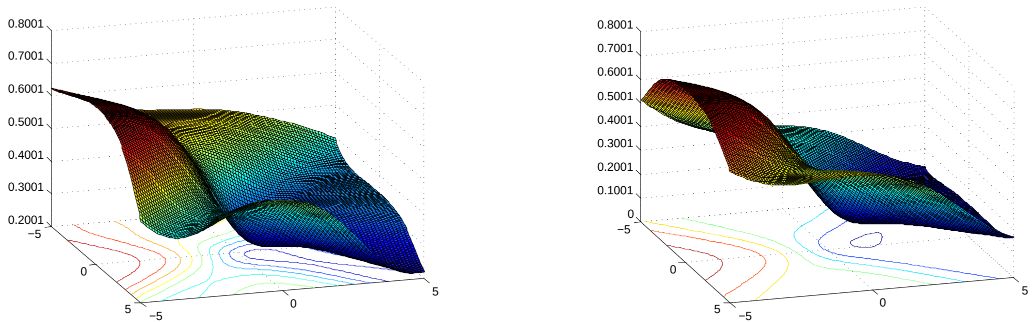
Squared-Error (MSE) for computing the distance between the actual network output and the target output. However, using a differentiable distance function such as MSE is not restrictive. Actually, the analysis provided throughout Section 4 makes no assumption regarding the type of the cost function of the network output. What really matters is that the set of boxes defined by the proposed method surely encloses weight vectors that are global minimizers, which means that the range of the error function computed on this set of boxes is an interval enclosing its global minimum.

Note that for all examples the boxes are computed using for $\mathfrak{b}_1$ and $\mathfrak{b}_2$ the values proposed in Paragraph 3.2.1. Obviously, one may set these values according to the precision required for the problem at hand. However, in order to comply with the theoretical considerations of the paper the values chosen for $\mathfrak{b}_1$ and $\mathfrak{b}_2$ should be such that the corresponding sigmoid is driven to the saturation region.

5.2 Experiments and Results

5.2.1 A simple problem

The Single Neuron problem: Here, we consider the simplest possible case consisting of a single neuron having one input, a bias and using the logistic sigmoid activation. The advantage of proposing such a rudimentary example is the facility it offers in visualizing its error function. In the following figures one may observe the form of the error functions for two simple artificial mappings with weights in the intervals $[-5.0, 5.0]$ and $[-5.0, 5.0]$. Using GOP it is easy to locate the global minima for each of these functions for various boxes of the weights. The results obtained for various trials are given hereafter.



(a) Mapping 1

(b) Mapping 2

Figure 1: A simple neuron and the error functions for two artificial mappings

Initial box for the weights: $[-72.1, 72.1][-36.05, 36.05]$, tolerance $= 0.001$. Intervals computed by GOP:

```
Function name:  SINGLE NEURON MAPPING 1
The set of global minimizers is located in the union of the following boxes:
c1:  [21.75680336914063, 37.45827910156252][20.27820312500000, 36.05010000000001]
c2:  [21.54557275390626, 21.68639316406251][19.99656210937500, 20.27820312500000]
c3:  [21.12311152343751, 21.40475234375001][19.71492109375000, 19.99656210937500]
```

```
The global minimum is enclosed in:
[0.176989707623682269, 0.177214552453163560]
```

```
Function name:  SINGLE NEURON MAPPING 2
The set of global minimizers is located in the union of the following boxes:
c1:  [-1.44335920410156, -1.05610307617187][1.09140893554687, 1.40825507812500]
The global minimum is enclosed in:
[0.093025775813289955, 0.093289721540457949]
```

In these two cases, it is obvious that the location of the minimizers greatly differs from one error function to the other. Moreover, in the case of the first error function the global minimum is defined exclusively for positive values of the weights. This shows that the input data greatly affect the location of the global minimum.

Let us now consider a narrower interval. Initial box for the weights: $[-5.0, 5.0][-5.0, 5.0]$, tolerance = 0.01. Intervals computed by GOP:

```
Function name:  SINGLE NEURON MAPPING 1
The set of global minimizers is located in the union of the following boxes:
c1:  [4.68750000000000, 5.00000000000000][4.21875000000000, 4.68750000000000]
The global minimum is enclosed in:
[0.215127653743816594, 0.219438417879982417]
```

```
Function name:  SINGLE NEURON MAPPING 2
The set of global minimizers is located in the union of the following boxes:
c1:  [-2.03125000000000, -0.62500000000000][0.85937500000000, 1.87500000000000]
c2:  [-2.34375000000000, -2.18750000000000][2.03125000000000, 2.18750000000000]
The global minimum is enclosed in:
[0.089012518962891946, 0.093326904683555062]
```

In this trial, narrower intervals were used for the bounds of the weights. As far as the first mapping is concerned, it's easy to notice that, despite the larger value for the tolerance, the intervals used do not enclose the previously detected global minimum. For the intervals specified the so-called global minimum is a local one, despite being the best that GOP was able to locate. For the second mapping, the intervals estimated for both the location of the global minimizers as well as for the global minimum are larger, due to the higher tolerance value, and enclose the intervals computed by GOP in the previous trial.

The main outcome of these examples is that they are consistent with the analysis provided in the paper. These trials provide "good" examples of boxes with global minimizers, as well as "counter-examples" of boxes without global minimizers. The results confirm that heuristic arguments, concerning the weight intervals, are inefficient, whenever global optimization based training is considered for a neural network. Moreover, the convex set defined by the proposed approach provides reliable estimation of these weight intervals.

For bound constrained problems the global optimizer GOP detects the best available minimizers within the box defined for the error function and hence the interval estimate of the corresponding global minimum. Given the size of the search space, in the following experiments it has not been possible to perform exhaustive search of the global minimum and so we tried to obtain rough estimations of the interval enclosing it.

5.2.2 Problems – Networks with Logistic Sigmoid Activation Functions

The XOR problem: Let us consider a 2-2-1 network used for the XOR problem. The purpose of this experiment is to allow us to verify the results of the proposed approach against previous literature (see Hamey, 1995, 1998; Sprinkhuizen-Kuyper and Boers, 1999), which has investigated analytically the local and global minima in the XOR problem. All network nodes use the logistic sigmoid activation function, inputs to all nodes are in the interval $[0, 1]$ and the desired outputs are the bounds of the interval $[0, 1]$. The network is fully connected and so there are $2 * 2 + 2 * 1 = 6$ unknown weights as well as $2 + 1 = 3$ bias weights. The interval equation of the weights to any node in the network is one of (7a) or (9a). Thus, if we set $\mathfrak{b} = \mathfrak{b}_1 = 36.05$ we obtain,

$$[w_1][0,1] + [w_2][0,1] + \cdots + [w_n][0,1] + [w_b][1,1] = [-36.05, 36.05]$$

and so, the initial box where global minimizers are expected to be located is

$$\underbrace{[-72.1, 72.1]^{2*2} \times [-36.05, 36.05]^2}_{\text{Hidden layer}} \times \underbrace{[-72.1, 72.1]^{2*1} \times [-36.05, 36.05]^1}_{\text{Output layer}}.$$

The output provided by GOP for the above network is:

```
Function name:  XOR
The set of global minimizers is located in the union of the following boxes:
c1:  [-72.09999999999999, 72.09999999999999] ···
[-36.05000000000000, 36.05000000000000]
The global minimum is enclosed in:
[0.000000000000000000, 0.250000000000000000].
```

The results of this experiment confirm the theoretical conclusions regarding the location of the global minimizers.

In addition to the previous experiment, it is tentative to examine, here, if the narrower box, defined using a loose interpretation of Corollary 3, is likely to enclose the global minimizers of the network output error. So, if the box $\underbrace{[-36.05, 36.05]^{2*2} \times [-36.05, 36.05]^2}_{\text{Hidden layer}}$ $\times \underbrace{[-36.05, 36.05]^{2*1} \times [-36.05, 36.05]^1}_{\text{Output layer}} = [-36.05, 36.05]^9$ is used as the initial box then the output provided by GOP is:

```
Function name:  XOR
The set of global minimizers is located in the union of the following boxes:
c1:  [-36.05000000000000, 36.05000000000000] ···
[-36.05000000000000, 36.05000000000000]
The global minimum is enclosed in:
[0.000000000000000000, 0.250000000000000000].
```

The result of this last experiment shows that a narrower box can be effectively used for searching the global minimizers. Nevertheless, we should re-iterate that, as mentioned in

Subsection 4.3, using such a narrower box lacks the necessary theoretical justification.

Finally, if we make the assumption that the outputs of the sigmoids are in the interval $[0.1, 0.9]$ the initial box is considered to be $[-4.4, 4.4]^6 \times [-2.2, 2.2]^3$ which, according to Table 1, is $\underbrace{[-4.4, 4.4]^{2*2} \times [-2.2, 2.2]^2}_{\text{Hidden layer}} \times \underbrace{[-4.4, 4.4]^{2*1} \times [-2.2, 2.2]^1}_{\text{Output layer}}$. Then the output provided by GOP is:

```
Function name:  XOR
The set of global minimizers is located in the union of the following boxes:
c1:  [-4.40000000000000, 4.40000000000000] ···
[-2.20000000000000, 2.20000000000000]
The global minimum is enclosed in:
[0.000000000000000000, 0.160000000000000031].
```

Overall, the outcomes of the three experiments are in line with previous research (Hamey, 1995, 1998; Sprinkhuizen-Kuyper and Boers, 1999). Actually, Hamey (1998) applied to the XOR problem a new methodology for the analysis of the error surface and showed that starting from any point with finite weight values, there exists a finite non-ascending trajectory to a point with error equal to zero. Moreover, Sprinkhuizen-Kuyper and Boers (1999) proved that "the error surface of the two-layer XOR network with two hidden units has a number of regions with local minima" and concluded that "from each finite point in weight space, a strictly decreasing path exists to a point with error zero". The conclusions of these papers resulted following an exhaustive analysis of the error surface of the network output. Finally, both papers conclude that for a 2-2-1 network using finite weights a global minimum is reachable in the error surface of the XOR problem. This conclusion constitutes a qualitative validation of the results obtained in this paper.

Finally, let us consider the case where a set of boxes does not contain a global minimizer. An example of such a set of boxes is reported by GOP hereafter.

```
Function name:  XOR
The set of global minimizers is located in the union of the following boxes:
c1:
[-72.09999999999999, 102.09999999999999][-72.09999999999999, 102.09999999999999]
[-72.09999999999999, 102.09999999999999][-72.09999999999999, 102.09999999999999]
[   36.05000000000000, 136.05000000000001][   36.05000000000000, 136.05000000000001]
[   72.09999999999999, 102.09999999999999][   72.09999999999999, 102.09999999999999]
[   36.05000000000000, 136.05000000000001]
The global minimum is enclosed in:
[0.498825225561647045, 0.500000000000000111].
```

The IRIS classification problem: Let us consider a 4-5-3 network used for the IRIS classification problem. This benchmark is known as Fisher's IRIS problem. Based on the values of sepal length and width, petal length and width, the class of IRIS plant needs to be predicted. Inputs are scaled in the interval $[0, 1]$, all network nodes use the logistic sigmoid activation function and the desired outputs are binary $\{0, 1\}$. The network is fully connected and so there are $4 * 5 + 5 * 3 = 35$ unknown weights along with $5 + 3 = 8$ bias weights. The interval equation of the weights to any node in the network is one of (7a) or (9a) and so if we set $\mathfrak{b} = \mathfrak{b}_1 = 36.05$ we have again,

$$[w_1][0,1] + [w_2][0,1] + \cdots + [w_n][0,1] + [w_b][1,1] = [-36.05, 36.05].$$

So, one may consider the following initial box for searching the global minimizers,

$$\underbrace{[-72.1, 72.1]^{4*5} \times [-36.05, 36.05]^5}_{\text{Hidden layer}} \times \underbrace{[-72.1, 72.1]^{5*3} \times [-36.05, 36.05]^3}_{\text{Output layer}}.$$

The output provided by GOP for the above network is:

```
Function name:  IRIS
The set of global minimizers is located in the union of the following boxes:
c1:  [−72.09999999999999, 72.09999999999999] ···
[−36.05000000000000, 36.05000000000000]
The global minimum is enclosed in:
[0.000000000000000000, 0.750000000000000000].
```

Moreover, under the assumption that the output of the sigmoids are in the interval $[0.1, 0.9]$ the initial box is considered to be $[-4.4, 4.4]^{35} \times [-2.2, 2.2]^8$ which, according to Table 1, is $\underbrace{[-4.4, 4.4]^{4*5} \times [-2.2, 2.2]^5}_{\text{Hidden layer}} \times \underbrace{[-4.4, 4.4]^{5*3} \times [-2.2, 2.2]^3}_{\text{Output layer}}$. Then the output provided by GOP is:

```
Function name:  IRIS
The set of global minimizers is located in the union of the following boxes:
c1:  [−4.40000000000000, 4.40000000000000] ···
[−2.20000000000000, 2.20000000000000]
The global minimum is enclosed in:
[0.000000000000000000, 0.480000000000002480].
```

The results of these IRIS experiments are encouraging. Again, the results of the first experiment roughly confirm the theoretical conclusions regarding the location of the global minimizers, while the other experiment indicates that narrower boxes can be effectively used for searching the global minimizers, eventually providing a narrower interval estimate for the global minimum.

5.2.3 A Problem – Network with Hyperbolic Tangent Activation Functions

The British vowels recognition problem: Consider the 3-layer 10-20-11 network used for the British vowels data recognition benchmark. This benchmark, known as Deterding Data (Deterding, 1989), is a speaker independent recognition problem of the eleven steady state vowels of British English using a specified training set of 10 lpc (linear prediction coefficients) derived log area ratios. Inputs for this problem are scaled in the interval $[-1, 1]$ and all network nodes use the hyperbolic tangent sigmoid activation function. The network is fully connected and so there are $10 * 20 + 20 * 11 = 420$ unknown weights as well as $20 + 11 = 31$ bias weights. In this case the interval equation of the weights to any node in the network is one of (7b) or (9b). Setting $\mathfrak{b} = \mathfrak{b}_2 = 18.04$ one obtains the following equation,

$$[w_1][-1,1] + [w_2][-1,1] + \cdots + [w_n][-1,1] + [w_b][1,1] = [-18.4, 18.4].$$

So, the initial box where global minimizers are potentially located is the hypercube

$$\underbrace{[-18.4, 18.4]^{(10+1)*20}}_{\text{Hidden layer}} \times \underbrace{[-18.4, 18.4]^{(20+1)*11}}_{\text{Output layer}} = [-18.4, 18.4]^{451}.$$

The output provided by GOP for the above network is:

```
Function name:  BRITISH VOWELS

The set of global minimizers is located in the union of the following boxes:
c1:  [−18.40000000000000, 18.40000000000000] ···
[−18.40000000000000, 18.40000000000000]
The global minimum is enclosed in:
[0.000000000000000000, 11.000000000000000000].
```

Under the assumption that the output of the sigmoids are in the interval $[-0.9, 0.9]$ the initial box is expected to be $\underbrace{[-1.5, 1.5]^{(10+1)*20}}_{\text{Hidden layer}} \times \underbrace{[-1.5, 1.5]^{(20+1)*11}}_{\text{Output layer}} = [-1.5, 1.5]^{451}$. The output provided by GOP for the above network is:

```
Function name:  BRITISH VOWELS

The set of global minimizers is located in the union of the following boxes:
c1:  [−1.50000000000000, 1.50000000000000] ···
[−1.50000000000000, 1.50000000000000]
The global minimum is enclosed in:
[0.000000000000000000, 8.910000000000142251].
```

The results of the first experiment confirm the theoretical conclusions regarding the location of the global minimizers. In addition, the second experiment confirms the hypothesis that the narrower box, derived using the interval $[-0.9, 0.9]$ for the range of the hyperbolic activation functions, can be effectively used for searching the global minimizers of the network learning error.

5.2.4 A Problem – Network with Hyperbolic Tangent and A Pure Linear Activation Function

A sinusoidal function approximation problem: Consider the 3-layer 2-21-1 network used for approximating the function $y = 0.5\sin\left(\pi x_1^2\right)\sin\left(2\pi x_2\right)$ defined in the original paper of Nguyen and Widrow (Nguyen and Widrow, 1990) where they introduce their weight initialization method. Inputs for this problem are in the interval $[-1, 1]$, all nodes in the hidden layer use the hyperbolic tangent sigmoid activation function and nodes in the output layer are linear. The network is fully connected and so there are $2*21 + 21*1 = 63$ unknown weights as well as $21 + 1 = 22$ bias weights. The interval equation of the weights to any node in the hidden layer is (7b). Retaining for $\mathfrak{b} = \mathfrak{b}_2 = 18.04$, as in the previous benchmark, we have

$$[w_1][-1,1] + [w_2][-1,1] + \cdots + [w_n][-1,1] + [w_b][1,1] = [-18.4, 18.4].$$

Moreover, the interval equation of the weights to any node in the output layer is,

$$[w_1][-1,1] + [w_2][-1,1] + \cdots + [w_n][-1,1] + [w_b][1,1] = [-0.5, 0.5].$$

So, the initial box used for searching the global minimizers is

$$\underbrace{[-18.4, 18.4]^{(2+1)*21}}_{\text{Hidden layer}} \times \underbrace{[-0.5, 0.5]^{(21+1)*1}}_{\text{Output layer}}.$$

The output provided by GOP for the above network is:

```
Function name:  FUNCTION APPROXIMATION
The set of global minimizers is located in the union of the following boxes:
c1:  [−18.40000000000000, 18.40000000000000] ···
[−0.50000000000000, 0.50000000000000]
The global minimum is enclosed in:
[0.000000000000000000, 0.048430679703413922].
```

5.3 Discussion and Open Problems

The above experiments provide substantial evidence regarding the validity of the theoretical results, showing that the convex set computed by the proposed approach for each problem contains some global minimizers of the network's error function. In addition, it is easy to see that there exist areas in the weight space which do not contain global minimizers.

In all cases the approach for evaluating the theoretical results was based on an interval global optimization procedure. However, the theoretical analysis presented in Section 4 made no assumption regarding the global optimization procedure used to minimize the error function of the MLP. Therefore, the set of boxes enclosing the global minimizers is determined independently of the global optimization method used for training, and so it is also valid for population-based or stochastic global search methods. It is worth noting here that the results reported by Gudise and Venayagamoorthy (2003) constitute an experimental verification of this statement when PSO is used to train an MLP. Moreover, the experimental analysis provided in that paper, concerning the computational cost induced by the training procedure in relation with the width of the search space, may be seen as an evidence of the advantage offered by our approach.

An interesting point of the above experiments concerns the results obtained when the boxes are defined under the assumption that the outputs of the sigmoid and the hyperbolic tangent activation functions are $[0.1, 0.9]$ and $[-0.9, 0.9]$ instead of $[0, 1]$ and $[-1, 1]$. These results indicate that the weight boxes defined so, besides being thinner, also result in a narrower interval enclosing the global minimum. We note here, without further details, that the derivation of these boxes and the entailing results, may be justified using the following theorem by Moore (1966) as formulated in Alefeld and Mayer (2000, Theorem 1).

Theorem 8 *Let $f : D \subset \mathbb{R}^n \to \mathbb{R}$ be continuous and let $[x] \subseteq [x]^0 \subseteq D$. Then (under mild additional assumptions) $q(R(f;[x]), [f]([x])) \leqslant \gamma \|\mathrm{d}([x])\|_\infty$, $\gamma \geqslant 0$, and $\mathrm{d}([f]([x])) \leqslant \delta \|\mathrm{d}([x])\|_\infty$, $\delta \geqslant 0$.*

Here, $q(\cdot,\cdot)$ is the Hausdorff distance between intervals which measures the quality of the interval inclusion function $[f]$ as an enclosure of the range of f over an interval $[x]$. This Theorem states that if the inclusion function exists then the Hausdorff distance between $R(f;[x])$ and $[f]([x])$ goes linearly to zero with the diameter $\mathrm{d}([x])$ (Alefeld and Mayer, 2000). In addition the diameter of the inclusion function goes linearly to zero if $\mathrm{d}([x])$ is approaching zero. The values of parameters γ and δ are very important for defining thinner boxes.

Regarding the previous comment it is important to note that considering narrower intervals for the outputs of the nonlinear activation functions of the nodes cannot be done arbitrarily. Actually, narrowing the intervals of the activation functions' output values causes a decrease in their level of saturation which translates to less sharp values for the network outputs. Obviously, this introduces a degree of uncertainty to the network outputs and thereby to the network function itself. The importance of this outcome seems to be problem dependent. If the underlying problem deals with pattern classification this outcome may have no effect on the classification task and in the best case even increase the generalization ability of the network. On the other hand when dealing with a function approximation or regression problem defining less sharp values for the network outputs is very likely to cause large deviations from the target outputs. To the best of our knowledge there is no research report attempting to define any proper saturation level for the activation functions in order to fit any unknown function with infinite accuracy.

Finally, an issue that needs to be commented concerns the impact of the proposed method on the global optimization procedure, in terms of computational complexity and cost. As the results of the experiments underline, this issue has not been addressed in this paper mainly for two reasons; first the objective of our paper is not to detect the global minimizers of the network error function but merely to delimit the area where a global optimization procedure should search for them. Secondly, evaluating the computational cost of our approach requires either a suitable mathematical formulation or significant computational effort in order to effectively measure the impact of our approach on specific global optimization procedures. Definitely, this issue constitutes our main concern for continuing this research.

6. Conclusion

The approach presented in this paper deals with computing guaranteed bounds for the region where global optimization procedures should search for global minimizers when training an MLP. In contrast to current practice, which defines these bounds heuristically, the proposed approach relies on interval analysis and exploits both the network architecture and information of the input patterns for shaping the search region. The resulting feasible domain of the network's output error function is rather complicated and so a suitable approximation is derived in the form of a convex polytope or a hypercube depending on the network architecture and the problem at hand.

The analysis presented deals with 3-layer feed forward neural networks but the results can easily be extended to networks with more than one hidden layer. Moreover, this analysis covers the widely used type of feed forward neural networks using some kind of sigmoid activation function for the nodes in the hidden layer. Nodes in the output layer may use either a sigmoid or a linear activation function. The conclusions derived are applicable to both standard MLPs and interval feed forward neural networks.

The examples and the experiments given on well known benchmarks highlight the application of the theoretical results formally derived in the paper. The results of the experiments provide significant evidence that the global minimizers of the considered network error functions fall within the bounds of the polytope defined by our method. Moreover, the proposed approach is not restrictive in terms of the hidden layers considered for the MLP or the bounds of the interval for node's output values.

A thorough performance analysis of global optimization procedures with and without using the proposed approach was not part of the objectives of this paper — this will be considered in future work. Also, we are planning to investigate the possibility of dealing with MLPs that use other types of activation functions as well as explore the applicability of the results to other types of networks such as the radial basis function and the recurrent networks.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable suggestions and comments on earlier version of the manuscript, that helped to significantly improve the paper at hand.

Appendix A. Sample Code for an MLP Natural Inclusion Function

Here, we give an example of the MATLAB code used for the natural inclusion function of the function implemented by a 3-layer MLP for the XOR benchmark. It is straightforward to obtain the natural inclusion function by simply replacing real variables by their interval counterparts. So, all variables corresponding to the weights of the MLP, as well as the quantities computed with these variables, are intervals. Interval computations are automatically carried out using INTLAB (Rump, 1999).

```
% the following define the activation functions
avfHL = @(x)[1./(1+exp(-x))]; % for the hidden layer
avfOL = @(x)[1./(1+exp(-x))]; % for the output layer

% the code for the inclusion function
function y = evxor(W0)
% W0 is the vector of the weight intervals
% global network parameters and training data
global p t np n h o avfHL avfOL

W1 = reshape(W0(1:h*n,1),h,n);
```

```
b1 = repmat(reshape(W0((h*n)+1:(h*n)+h*1,1),h,1),1,np);
W2 = reshape(W0((h*(n+1))+1:(h*(n+1))+o*h,1),o,h);
b2 = repmat(reshape(W0(h*(n+1)+o*h+1:end,1),o,1),1,np);

y = sum((t-avfOL(W2*avfHL(W1*p+b1)+b2)).^2)./np;
% end function
```

References

S.P. Adam, D.A. Karras, G.D. Magoulas, and M.N. Vrahatis. Solving the linear interval tolerance problem for weight initialization of neural networks. *Neural Networks*, 54:17–37, 2014.

G. Alefeld and G. Mayer. Interval analysis: theory and applications. *Journal of Computational and Applied Mathematics*, 121:421–464, 2000.

A.M. Bagirov, A.M. Rubinov, and J. Zhang. A multidimensional descent method for global optimization. *Optimization*, 58(5):611–625, 2009.

O. Beaumont. Solving interval linear systems with linear programming techniques. *Linear Algebra and its Applications*, 281:293–309, 1998.

D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statistical Science*, 8(1):10–15, 1993.

C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

C.G.E. Boender, A.H.G. Rinnooy Kan, G.T. Timmer, and L. Stougie. A stochastic method for global optimization. *Mathematical Programming*, 22:125–140, 1982.

S.H. Brooks. A discussion of random methods for seeking maxima. *Operations Research*, 6 (2):244–251, 1958.

R.E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998.

O. Caprani, B. Godthaab, and K. Madsen. Use of a real-valued local minimum in parallel interval global optimization. *Interval Computations*, 2:71–82, 1993.

P.A. Castillo, J.J. Merelo, A. Prieto, V. Rivas, and G. Romero. G-prop: Global optimization of multilayer perceptrons using GAs. *Neurocomputing*, 35:149–163, 2000.

M. Clerc and J. Kennedy. The particle swarm – explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6 (1):58–73, 2002.

M. Courbariaux, Y. Bengio, and J-P. David. Training deep neural networks with low precision multiplications. *ArXiv e-prints*, abs/1412.7024, 2014. Available electronically via http://arxiv.org/abs/1412.7024.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals and Systems*, 2:303–314, 1989.

D.H. Deterding. *Speaker Normalisation for Automatic Speech Recognition.* PhD thesis, University of Cambridge, 1989.

S. Draghici. On the capabilities of neural networks using limited precision weights. *Neural Networks*, 15(3):395–414, 2002.

W. Duch and N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2:163–212, 1999. Available electronically at ftp://ftp.icsi.berkeley.edu/pub/ai/jagota/vol2_6.pdf.

W. Duch and J. Korczak. Optimization and global minimization methods suitable for neural networks. *Neural Computing Surveys*, 2:163–212, 1998. Available electronically via http://www.fizyka.umk.pl/publications/kmk/99globmin.html.

A.E. Eiben, E.H.L. Aarts, and K.M. Van Hee. Global convergence of genetic algorithms: A Markov chain analysis. In Hans-Paul Schwefel and Reinhard Manner, editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 3–12. Springer, Berlin, 1991.

J. Engel. Teaching feed-forward neural networks by simulated annealing. *Complex Systems*, 2(6):641–648, 1988.

V.G. Gudise and G.K. Venayagamoorthy. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the IEEE Swarm Intelligence Symposium, SIS '03*, pages 110–117, 2003.

S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1737–1746. JMLR Workshop and Conference Proceedings, 2015.

L.G.C. Hamey. Analysis of the error surface of the XOR network with two hidden nodes. Computing Report 95/167C, Department of Computing, Macquarie University, NSW 2109, Australia, 1995.

L.G.C. Hamey. XOR has no local minima: A case study in neural network error surface analysis. *Neural Networks*, 1(4):669–681, 1998.

E.R. Hansen. On the solution of linear algebraic equations with interval coefficients. *Linear Algebra and its Applications*, 2(2):153–165, 1969.

E.R. Hansen and S. Sengupta. Bounding solutions of systems of equations using interval analysis. *BIT Numerical Mathematics*, 21:203–211, 1981.

E.R. Hansen and G.W. Walster. *Global Optimization Using Interval Analysis.* Marcel Dekker, New York, 2004.

S. Haykin. *Neural Networks A Comprehensive Foundation.* Prentice-Hall, Upper Saddle River, New Jersey, 1999.

S. Helwig and R. Wanka. Theoretical analysis of initial particle swarm behavior. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, editors, *Parallel Problem Solving from Nature – PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 889–898. Springer, Berlin, 2008.

N. Higham. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, 2002.

M. Hoehfeld and S. E. Fahlman. Learning with limited numerical precision using the cascade-correlation algorithm. *IEEE Transactions on Neural Networks*, 3(4):602–611, 1992.

C. Hölbig and W. Krämer. Self-verifying solvers for dense systems of linear equations realized in C-XSC. Universität Wuppertal, Preprint BUGHW-WRSWT 2003/1, 2003. Available electronically via http://www.math.uni-wuppertal.de/wrswt/literatur.html.

J. L. Holi and J. N. Hwang. Finite precision error analysis of neural network hardware implementations. *IEEE Transactions on Computers*, 42(3):281–290, 1993.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

R. Horst and P.M. Pardalos. *Handbook of Global Optimization.* Kluwer Academic Publishers, Dordrecht, 1995.

C. Hu, M. Beheshti, A. Berrached, A. de Korvin, and O. Sirisaengtaksin. On interval weighted three-layer neural networks. In *Proceedings of the 31st Annual Simulation Symposium*, pages 188–194. IEEE Computer Society Press, 1998.

IEEE. Standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008.

J. Ilonen, J.K. Kamarainen, and J. Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1):93–105, 2003.

M. Jamett and G. Acuña. An interval approach for weight's initialization of feedforward neural networks. In *Proceedings of the 5th Mexican International Conference on Artificial Intelligence, MICAI 2006*, volume 4293 of *LNCS*, pages 305–315. Springer–Verlag, Berlin, 2006.

C. Jansson. Calculation of exact bounds for the solution set of linear interval systems. *Linear Algebra and its Applications*, 251:321–340, 1997.

L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis. With Examples in Parameter and State Estimation, Robust Control and Robotics.* Springer–Verlag, London, 2001.

P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, N. E. Jerger, and A. Moshovos. Proteus: Exploiting numerical precision variability in deep neural networks. In *Proceedings of the 2016 International Conference on Supercomputing*, ICS '16, pages 23:1–23:12, New York, 2016. ACM.

S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

W. Krämer, U. Kulisch, and R. Lohner. *Numerical Toolbox for Verified Computing II: Advanced Numerical Problems.* Springer–Verlag, Berlin, 2006.

V. Kreinovich and O. Sirisaengtaksin. 3-layer neural networks are universal approximators for functionals and for control strategies. *Neural Parallel & Scientific Computations*, 1: 325–346, 1993.

V. Kreinovich, A.V. Lakeyev, and S.I. Noskov. Optimal solution of interval linear systems is intractable (NP-hard). *Interval Computations*, 1:6–14, 1993.

Y. LeCun. Efficient learning and second-order methods. Tutorial at Neural Information Processing Systems Conference, NIPS, 1993.

G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis. Effective back-propagation training with variable stepsize. *Neural Networks*, 10(1):69–82, 1997.

G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11(7):1769–1796, 1999.

MATLAB-Documentation. Floating-point relative accuracy – MATLAB eps. Online documentation, 2014. Retrieved 25 July 2014 from http://www.mathworks.com/help/matlab/ref/eps.html.

D.J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In N.S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 762–767. Morgan Kaufmann, 1989.

R.E. Moore. *Interval Analysis.* Prentice–Hall, Englewood Cliffs, New Jersey, 1966.

A. Neumaier. Linear interval equations. In K. Nickel, editor, *Interval Mathematics 1985*, volume 212 of *Lecture Notes in Computer Science*, pages 109–120. Springer, Berlin, 1986.

A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge University Press, New York, 1990.

C. Ng, D. Li, and L. Zhang. Global descent method for global optimization. *SIAM Journal on Optimization*, 20(6):3161–3184, 2010.

D. Nguyen and B. Widrow. Improving the learning speed of two-layer neural networks by choosing initial values of the adaptive weights. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN'90*, volume 3, pages 21–26, Ann Arbor, Michigan, 1990.

L. Pàl. *Global Optimization Algorithms for Bound Constrained Problems.* PhD Dissertation, PhD School in Computer Science, University of Szeged, Szeged, Hungary, 2010.

L. Pàl and T. Csendes. INTLAB implementation of an interval global optimization algorithm. *Optimization Methods and Software*, 24(4-5):749–759, 2009. Available electronically via http://www.inf.u-szeged.hu/~csendes/Reg/regform.php.

K.E. Parsopoulos and M.N. Vrahatis. *Particle Swarm Optimization and Intelligence: Advances and Applications.* Information Science Publishing (IGI Global), Hershey, Pennsylvania, 2010.

M.E.H. Pedersen. *Tuning & Simplifying Heuristical Optimization.* PhD thesis, University of Southampton, School of Engineering Sciences, Computational Engineering and Design Group, 2010.

V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis. Learning in multilayer perceptrons using global optimization strategies. *Nonlinear Analysis: Theory, Methods & Applications*, 47(5):3431–3436, 2001.

V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis. Improved learning of neural nets through global search. In Janos D. Pinter, editor, *Global Optimization*, volume 85 of *Nonconvex Optimization and Its Applications*, pages 361–388. Springer, 2006.

E.D. Popova. Improved solution enclosures for over- and underdetermined interval linear systems. In Ivan Lirkov, Svetozar Margenov, and Jerzy Wasniewski, editors, *Large-Scale Scientific Computing*, volume 3743 of *Lecture Notes in Computer Science*, pages 305–312. Springer, Berlin, 2006.

H. Ratschek and W. Sauer. Linear interval equations. *Computing*, 28:105–115, 1982.

J. Rohn. Systems of linear interval equations. *Linear Algebra and its Applications*, 126: 39–78, 1989.

J. Rohn. Checking bounds on solutions of linear interval equations is NP-hard. *Linear Algebra and its Applications*, 223–224:589–596, 1995.

J. Rohn. Solvability of systems of linear equations. *SIAM Journal on Matrix Analysis and Applications*, 25:237–245, 2003.

D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing.* MIT Press, Cambridge, Massachusetts, 1986.

S.M. Rump. Solving algebraic problems with high accuracy. In U.W. Kulisch and W.L. Miranker, editors, *A New Approach to Scientific Computation*, pages 51–120. Academic Press, New York, 1983.

S.M. Rump. INTLAB – INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic, Dordrecht, Netherlands, 1999.

P.V. Saraev. Numerical methods of interval analysis in learning neural network. *Automation and Remote Control*, 73(11):1865–1876, 2012.

Y. Shang and B.W. Wah. Global optimization for neural network training. *IEEE Computer*, 29(3):45–54, 1996.

S.P. Shary. On optimal solution of interval linear equations. *SIAM Journal on Numerical Analysis*, 32(2):610–630, 1995.

S.P. Shary. Algebraic approach to the interval linear static identification, tolerance, and control problems, or one more application of Kaucher arithmetic. *Reliable Computing*, 2 (1):3–33, 1996.

S.P. Shary. Algebraic approach in the "outer problem" for interval linear equations. *Reliable Computing*, 3(2):103–135, 1997.

S.P. Shary. A new technique in systems analysis under interval uncertainty and ambiguity. *Reliable Computing*, 8:321–418, 2002.

I.G. Sprinkhuizen-Kuyper and E.J.W. Boers. The local minima of the error surface of the 2-2-1 XOR network. *Annals of Mathematics and Artificial Intelligence*, 25(1–2):107–136, 1999.

Z. Tang and G.J. Koehler. Deterministic global optimal FNN training algorithms. *Neural Networks*, 7(2):301–311, 1994.

F. van den Bergh and A.P. Engelbrecht. Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 26:84–90, 2000.

S. Vassiliadis, M. Zhang, and J. G. Delgado-Frias. Elementary function generators for neural-network emulators. *IEEE Transactions on Neural Networks*, 11(6):1438–1449, 2000.

C. Voglis and I.E. Lagaris. Towards "ideal multistart". A stochastic approach for locating the minima of a continuous function inside a bounded domain. *Applied Mathematics and Computation*, 213(1):216–229, 2009.

H. White. Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings. *Neural Networks*, 3:535–549, 1990.

P. Xu. A hybrid global optimization method: the one-dimensional case. *Journal of Computational and Applied Mathematics*, 147(12):301–314, 2002.

J.Y.F. Yam and T.W.S. Chow. Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients. *IEEE Transactions on Neural Networks*, 12:430–434, 2001.