Solving the linear interval tolerance problem for weight initialization of neural networks

S.P. Adam[a,b,*], D.A. Karras[c], G.D. Magoulas[d], M.N. Vrahatis[a]

[a] *Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR-26110 Patras, Greece*
[b] *Department of Telecommunications and Informatics, Technological Educational Institute of Epirus, 47100 Arta, Greece*
[c] *Department of Automation, Technological Educational Institute of Sterea Hellas, 34400 Psahna, Evia, Greece*
[d] *Department of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, UK*

Abstract

Determining good initial conditions for an algorithm used to train a neural network is considered a parameter estimation problem dealing with uncertainty about the initial weights. Interval Analysis approaches model uncertainty in parameter estimation problems using intervals and formulating tolerance problems. Solving a tolerance problem is defining lower and upper bounds of the intervals so that the system functionality is guaranteed within predefined limits. The aim of this paper is to show how the problem of determining the initial weight intervals of a neural network can be defined in terms of solving a linear interval tolerance problem. The proposed Linear Interval Tolerance Approach copes with uncertainty about the initial weights without any previous knowledge or specific assumptions on the input data as required by approaches such fuzzy sets or rough sets. The proposed method is tested on a number of well known benchmarks for neural networks trained with the back-propagation family of algorithms. Its efficiency is evaluated with regards to standard performance measures and the results obtained are compared against results of a number of well known and established initialization methods. These results provide credible evidence that the proposed method outperforms classical weight initialization methods.

*Corresponding author
Email address: adamsp@upatras.gr (S.P. Adam)

Keywords:
Neural networks, Weight initialization, Interval analysis, Linear interval tolerance problem

---

1. Introduction

The purpose of Interval Analysis (IA) is to set upper and lower bounds on the effect produced on some computed quantity by different types of mathematical computing errors (rounding, approximation, uncertainty etc) (Moore, 1966; Hansen & Walster, 2004). Intervals are used to model uncertainty in parameter estimation problems such as the noise associated with measured data. Such problems arise in engineering design or mathematical modeling where tolerances in the relevant parameters need to be defined in terms of upper and lower bounds so that the desired functionality is guaranteed within these bounds. The interval-based algorithms are used to reliably approximate the set of consistent values of parameters by inner and outer intervals and thus take into account all possible options in numerical constraint satisfaction problems.

The promising features of IA motivated researchers from different disciplines to invest in the study and implementation of IA methods whenever reliable numerical computations are required. Currently, this research field is rapidly growing due to the increasing computation power of modern hardware. Examples of applications range from finite element analysis (Degrauwe et al., 2010) and data analysis (Garloff et al., 2007), to stock market forecasting (Hu & He, 2007), reliability of mechanical design (Penmetsa & Grandhi, 2002), and many more. Research in the area of neural networks has also benefitted from IA and a number of efforts utilizing concepts and methods from IA are reported in the literature. Examples are those by de Weerdt et al. (2009) on the use of IA for optimizing the neural network output, Ishibuchi & Nii (1998) on the generalization ability of neural networks, Xu et al. (2005) on robust stability criteria for interval neural networks, Li et al. (2007) regarding training of neural networks, and others.

An important problem encountered when training a neural network is to determine appropriate initial values for the connection weights. Effective weight initialization

is associated to performance characteristics such as the time needed to successfully train the network and the generalization ability of the trained network. Inappropriate weight initialization is very likely to increase the training time or even to cause non convergence of the training algorithm, while another unfortunate result may be to decrease the network's ability to generalize well, especially when training with back-propagation (BP), a procedure suffering from local minima, (Haykin, 1999; Hassoun, 1995; Lee et al., 1991). These are defaults and limitations for having successful practical application of neural networks in real life processes.

The importance manifested by the research community for this subject has been demonstrated by the number of research work published in this area. The proposed approaches can be, roughly, divided into two categories. Methods in the first category perform input data clustering in order to extract significant information (feature vectors or reference patterns) pertaining the pattern space and initial connection weights are chosen to be near the centers of these clusters. The main drawback of these methods is the computational cost needed to preprocess the input data. Often this cost may be prohibitive for these methods to be used in real world applications. The second category includes those methods that are based on random selection of initial weights from a subset of $\mathbb{R}^n$, which is an interval defined considering important properties of the pattern space and/or the parameters of the training process.

The notion of the interval, underlying random weight selection methods, suggests the idea to use IA in order to deal with uncertainty about the initial weights. Hence, the unknown initial weights are considered to be intervals with unknown bounds. Under generally adopted assumptions about the input to any node, the resulting unknown interval quantity is then limited within specific upper and lower bounds. Ensuring that scientific computations provide results within guaranteed limits is an issue mentioned by researchers in IA as a tolerance problem. In consequence, the approach proposed herein gives rise to formulating a linear interval tolerance problem which is solved to determine significant intervals for the initial weights. Shary (1995); Pivkina & Kreinovich (2006); Beaumont & Philippe (2001) and other researchers propose different methods for solving a tolerance problem. Besides formulating the problem of determining initial weights as a linear interval tolerance problem, we also present here

a new algorithm for defining the required solution to the specific tolerance problem.

The proposed linear interval tolerance approach (LIT-Approach) deals with uncertainty about the initial weights based exclusively on numerical information of the patterns without any assumption on the distribution of the input data. IA provides the means of handling uncertainty in parameters in much the same way this happens with other approaches such as the possibilistic approach with Fuzzy sets (Zadeh, 1978), Evidence theory (Shafer, 1976), Rough sets (Pawlak, 1991) or methods combining properties of these approaches. However, methods using fuzzy sets require parameters of the membership functions to be tuned and eventually some preprocessing of the input data to be done if pertinent input variables need to be identified. Moreover, when using rough sets one needs to process the input data in order to deal with the indiscernibility relation and establish upper and lower approximations of the concepts pertaining the problem, see (Bello & Verdegay, 2012). Finally, application of the Dempster-Shafer (evidence) theory is a matter of subjective estimation of uncertainty as it assumes that values of belief (or plausibility) are given by an expert. Unlike all these approaches, the interval computation used for LIT-Approach needs only elementary statistics of the input data to be computed such the sample mean, the sample standard deviation or the median and the quartiles of the sample.

It is worth noting here the approach formulated by Jamett & Acuña (2006) as an interval approach for weight initialization. The solution proposed "solves the network weight initialization problem, performing an exhaustive search for minima by means of interval arithmetic. Then, the global minimum is obtained once the search has been limited to the region of convergence". For the experimental evaluation proposed, interval weights are initially defined as wide as necessary (with amplitudes up to 10^6). In addition, the IA solution adopted by these researchers extends to defining an interval version of the gradient descent procedure. On the contrary, the method presented in this paper uses IA concepts only for computing effective intervals for the initial weights and therefore it is not computationally expensive.

The sections of this paper are organized as follows. Section 2 is devoted to a presentation of the IA concepts underpinning the LIT-Approach. Section 3 presents the analysis of LIT-Approach including both theoretical results and the weight initializa-

tion algorithm. Section 4 is dedicated to the experimental evaluation of our approach and its comparison with well known initialization procedures. Finally, Section 5 summarizes the paper with some concluding remarks.

2. Interval Analysis and the Tolerance Problem

2.1. Interval Arithmetic

The arithmetic defined on sets of intervals, rather than sets of real numbers is called interval arithmetic. An interval or interval number I 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 number and interval are used interchangeably. Whenever $a = b$ the interval is said to be degenerate, thin or even point interval. An interval X may be also denoted as $\left[\underline{X}, \overline{X}\right]$, $[X]$ or even $[X_L, X_U]$ where subscripts $_L$ and $_U$ stand for lower and upper bounds respectively. Interval variables may be uppercase or lowercase, (Alefeld & Mayer, 2000). In this paper, identifiers for intervals and interval objects (variables or vectors) will be denoted with boldface lowercase such as $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ and boldface uppercase notation will be used for matrices, e.g. $\boldsymbol{X}$. Lowercase letters will be used for the square bracketed notation of intervals $[\underline{x}, \overline{x}]$, or the elements of an interval as a set. An interval $[\underline{x}, \overline{x}]$ where $\underline{x} = -\overline{x}$ is called a symmetric interval. Finally, if $\boldsymbol{x} = [\underline{x}, \overline{x}]$ then the following notation will be used in this paper.

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

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

$$\boldsymbol{x} \diamond \boldsymbol{y} = \{\, x \diamond y \,|\, x \in \boldsymbol{x},\ y \in \boldsymbol{y}\} \tag{1}$$

This definition guarantees that $x \diamond y \in \boldsymbol{x} \diamond \boldsymbol{y}$ for any arithmetic operator and any values of x and y. In practical calculations each interval arithmetic operation is reduced to operations between real numbers. If $\boldsymbol{x} = [\underline{x}, \overline{x}]$ and $\boldsymbol{y} = [\underline{y}, \overline{y}]$ then it can be shown that the above definition produces the following intervals for each arithmetic operation:

$$\boldsymbol{x} + \boldsymbol{y} = [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \tag{2a}$$

$$\boldsymbol{x} - \boldsymbol{y} = [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \tag{2b}$$

$$\boldsymbol{x} \times \boldsymbol{y} = \left\{\min\left(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy}\right), \max\left(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy}\right)\right\} \tag{2c}$$

$$\boldsymbol{x} \div \boldsymbol{y} = \boldsymbol{x} \times \frac{1}{\boldsymbol{y}}, \text{ with} \tag{2d}$$

$$\frac{1}{\boldsymbol{y}} = \left[\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right], \text{ provided that } 0 \notin \left[\underline{y}, \overline{y}\right] \tag{2e}$$

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 $\boldsymbol{x}, \boldsymbol{y}$ are non-degenerate intervals then,

$$\boldsymbol{x} + \boldsymbol{y} = z \not\Rightarrow \boldsymbol{x} = z - \boldsymbol{y}, \tag{3a}$$

$$\boldsymbol{x} \times \boldsymbol{y} = z \not\Rightarrow \boldsymbol{x} = z \times \frac{1}{\boldsymbol{y}}. \tag{3b}$$

The following sub-distributive law holds for non-degenerate intervals $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{z}$,

$$\boldsymbol{x} \times (\boldsymbol{y} + \boldsymbol{z}) \subseteq \boldsymbol{x} \times \boldsymbol{y} + \boldsymbol{x} \times \boldsymbol{z}. \tag{4}$$

One may easily verify that the usual distributive law holds if $\boldsymbol{x}$ is a point interval or if both $\boldsymbol{y}$ and $\boldsymbol{z}$ are point intervals. Hereafter, the multiplication operator $\times$ will be omitted as in usual algebraic expressions with real numbers. A very important property of interval arithmetic operations is that,

$$\begin{aligned} &\text{if} \quad \boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d} \in \mathbb{IR} \quad \text{and} \quad \boldsymbol{a} \subseteq \boldsymbol{b},\ \boldsymbol{c} \subseteq \boldsymbol{d} \\ &\text{then} \quad \boldsymbol{a} \diamond \boldsymbol{c} \subseteq \boldsymbol{b} \diamond \boldsymbol{d}, \qquad \diamond \in \{+, -, \times, \div\}. \end{aligned} \tag{5}$$

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

2.2. Interval Linear Systems

An interval linear system is a system of the form,

$$\mathbf{A}\boldsymbol{x} = \mathbf{b} \tag{6}$$

where $\mathbf{A} \in \mathbb{IR}^{m\times n}$, also noted $\left[\underline{A}, \overline{A}\right]$, is an m-by-n matrix of real intervals, $\mathbf{b} \in \mathbb{IR}^{m}$, also noted $\left[\underline{b}, \overline{b}\right]$, is an m-dimensional vector of real intervals and $\boldsymbol{x}$ is the n-dimensional vector of unknown interval variables. Solving a system of linear interval equations has attracted the interest of several researchers in the field of IA for more than forty years. Initially, research focused on systems with square interval matrices ($\mathbf{A} \in \mathbb{IR}^{n\times n}$) and a number of different methods for studying and solving such systems have been proposed.

To solve the above system of interval linear equations $\mathbf{A}\boldsymbol{x} = \mathbf{b}$, generally, means to compute the solution set defined as

$$\sum(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n \,|\, \tilde{A}x = \tilde{b} \text{ for real } \tilde{A} \in \mathbf{A},\ \tilde{b} \in \mathbf{b}\}. \tag{7}$$

That is, $\sum(\mathbf{A}, \mathbf{b})$ is the set of all solutions for all matrices $\tilde{A} \in \mathbf{A}$ with real elements and all vectors $\tilde{b} \in \mathbf{b}$ having real number components. This set is generally not an interval vector but a rather complicated set that is usually impractical to define and use (Hansen & Walster, 2004). In practice, defining this solution set resulted in proposing methods such as the interval versions of Gaussian elimination or the Gauss-Seidel method which compute vectors that bound $\sum(\mathbf{A}, \mathbf{b})$. Note that these interval algorithms differ significantly from corresponding point algorithms as they use *preconditioning* with a point matrix for the algorithms to be effective (Hansen & Walster, 2004; Neumaier, 1990). Other frequently used methods are those based on the Rump/Krawczyk iteration (Krawczyk, 1969; Rump, 2001).

An important issue was to define the narrowest interval vector containing the solution set $\sum(\mathbf{A}, \mathbf{b})$. This interval vector is called the *hull* of the solution set. Determining the hull is a problem that is NP-hard as shown by Heindl et al. (1998), and so, in general, methods try to compute only outer bounds for the hull. Other important research results include: the work by Rohn (2003) on the solvability of systems of linear interval equations with rectangular matrices, the algorithm proposed by Hansen (2006), to

solve over-determined systems and the work presented by Kubica (2010) on interval methods for under-determined nonlinear systems.

For such methods one may refer to Alefeld & Herzberger (1983); Neumaier (1990); Kreinovich et al. (1997); Hansen & Walster (2004); Hansen (1992); Kearfott (1996). The number of different methods proposed to solve systems of linear interval equations underlines the importance of the subject, especially regarding the difficulty to generally indentify the hull of the solution set of such a system. Research effort has been dedicated on the evaluation of different methods solving systems of linear interval equations. Important works on this matter include Neumaier (1984); Goldsztejn (2007); Ning & Kearfott (1997); Rohn (1993).

2.3. Tolerance Problem and the Tolerance Solution Set

The tolerance problem arises in engineering design and system modeling and refers to the estimation of the tolerance of certain parameters of a system or a device so that its behaviour i.e. its output is guaranteed within specified bounds. In mathematical terms, if $F : \mathbb{R}^n \to \mathbb{R}^m$ is the mapping relating variables $\mathrm{x} = (x_1, x_2, \ldots, x_n)^\top$ with output parameters $\mathrm{y} = (y_1, y_2, \ldots, y_m)^\top$, then the tolerance problem is associated with the computation of a domain for the variables of F such that the corresponding $\mathrm{y} = F(\mathrm{x})$ lie within some predefined range, (Neumaier, 1986, 1990).

In Shary (2002) the tolerance problem is described as a particular problem related with the analysis of a system. Using intervals and quantifier formalism to model uncertainty, about a system's parameters, Shary defines three types of solutions to the general input-state-output equation describing a system. These solutions are sets of values providing answers to different issues of systems analysis. Hence, according to Shary (2002), for the interval equation $F(\mathbf{a}, \mathrm{x}) = \mathbf{b}$ of a system with n unknown parameters $\mathrm{x} \in \mathbb{R}^n$, there are three particular cases of the general *AE-solution set*:

- the United solution set consisting of the solutions of all point equation systems of the form $F(\tilde{a}, \mathrm{x}) = \tilde{b}$ with $\tilde{a} \in \mathbf{a}$ and $\tilde{b} \in \mathbf{b}$,
- the Controllable solution set containing all point vectors x such that for any $\tilde{b} \in \mathbf{b}$ one can find the right $\tilde{a} \in \mathbf{a}$ such that $F(\tilde{a}, \mathrm{x}) = \tilde{b}$, and finally,

- the Tolerable (or Tolerance) solution set formed by all point vectors x such that for any $\tilde{a} \in \mathbf{a}$ the image $F(\tilde{a}, \mathrm{x}) \in \mathbf{b}$.

In the case of a static linear system F has the form of the interval linear system $\mathbf{A}\mathrm{x} = \mathbf{b}$ and the solution set defined by (7) is the United solution set. Using the notation introduced in Shary (1995) the solution sets defined previousely are:
United solution set:

$$\sum\nolimits_{\exists\exists}(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n \mid (\exists \tilde{A} \in \mathbf{A})(\exists \tilde{b} \in \mathbf{b})(\tilde{A}x = \tilde{b})\} \tag{8}$$

Controllable solution set:

$$\sum\nolimits_{\exists\forall}(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n \mid (\forall \tilde{b} \in \mathbf{b})(\exists \tilde{A} \in \mathbf{A})(\tilde{A}x = \tilde{b})\} \tag{9}$$

Tolerance solution set:

$$\sum\nolimits_{\forall\exists}(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n \mid (\forall \tilde{A} \in \mathbf{A})(\exists \tilde{b} \in \mathbf{b})(\tilde{A}x = \tilde{b})\} \tag{10}$$

$$\sum\nolimits_{\subseteq}(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n \mid \mathbf{A}x \subseteq \mathbf{b}\} \tag{11}$$

Both the Controllable and the Tolerance solution sets are subsets of the more general United solution set. The specific uncertainty problem defines which of the above solution sets contains the solution of the problem. With respect to the assumption that F describes the input-output relation of a static linear system, the tolerance solution set provides answers to the question whether there are input signals $\tilde{\mathrm{x}}$ to the system such that the output $\mathbf{A}\mathrm{x}$ remains within specified limits $\mathbf{b}$. Moreover, it is worth noting here that the elements of the solution sets, as defined previously, are not just points in $\mathbb{R}^n$ but they may be intervals in $\mathbb{IR}^n$ as well Shary (1995); Pivkina & Kreinovich (2006).

3. Weight Initialization with the LIT-Approach

3.1. Random Selection of Initial Weights

Random initialization of connection weights seems to be the most widely used approach for real world applications. A number of approaches such as those presented in this section claim the reputation to provide improvement in BP convergence speed and

avoidance of bad local minima, (Nguyen & Widrow, 1990; Wessels & Barnard, 1992). Unless differently defined, hereafter d_{in} denotes the number of inputs to a node.

Fahlman (1988) studies on random weight initialization techniques resulted in the use of a uniform distribution over the interval $[-1.0, 1.0]$. This seems to constitute a simplified approach for use in any problem without further hypotheses.

Boers & Kuiper (1992) initialize the weights using a uniform distribution over the interval $\left[-3/\sqrt{d_{in}}, 3/\sqrt{d_{in}}\right]$. This interval is defined so that the stimulus of any node is located around the origin of the axes where the sigmoid activation function has its steepest slope. This interval is the same as the one defined by the conventional method of Wessels & Barnard (1992). However, in order to avoid false local minima detected when applying this conventional method, Wessels & Barnard (1992) also propose a more refined method adopting a different strategy for the input-to-hidden layer connections and for the hidden-to-output layer connections.

Bottou (1988) defines the interval $\left[-a/\sqrt{d_{in}}, a/\sqrt{d_{in}}\right]$, where a is chosen so that the weight variance corresponds to the points of the maximal curvature of the activation function. For the logistic sigmoid activation function a is set to be approximately equal to 2.38 and 0.66 for the hyperbolic tangent. Criticism on this approach concerns the fact that it was not compared against other methods.

Kim & Ra (1991) calculated a lower bound for the initial length of the weight vector of a neuron to be $\sqrt{\eta/d_{in}}$ where η is the learning rate used by the training procedure.

Smieja (1991) based on the study of the hyperplanes dynamics, proposes uniformly distributed weights normalized to the magnitude $2/\sqrt{d_{in}}$ for each node. The thresholds for the hidden units are initialized to a random value in the interval $\left[-\sqrt{d_{in}}/2,\ \sqrt{d_{in}}/2\right]$ and the thresholds of the output nodes are set to zero.

Drago & Ridella (1992) proposed a method aiming to avoid flat regions in the error surface in an early stage of training. Their method is called statistically controlled activation weight initialization (SCAWI). They determine the maximum magnitude of the weights through statistical analysis. They show that the maximum magnitude of the weights is a function of the paralyzed neuron percentage (PNP), which is in turn related to the convergence rate. By determining the optimal range of PNP through computer simulations, the maximum magnitude of the weights can be obtained. The

weights are uniformly distributed over the interval $[-r, r]$ with $r = 1.3/\sqrt{1 + n_i v^2}$ for the hidden layer nodes and $r = 1.3/\sqrt{1 + 0.3n_h}$ for the output layer nodes. Here, n_i denotes the number of inputs to the network and n_h is the number of nodes in the hidden layer. In addition v^2 is the mean of the expectation of the quadratic values of the inputs, $v^2 = 1/n_i \sum_{i=1}^{n_i} E[I_i^2]$.

Nguyen & Widrow (1990) proposed a simple modification of the widely used random initialization process of Fahlman (1988). The weights connecting the output units to the hidden units are initialized with small random values over the interval $[-0.5, 0.5]$. The initial weights at the first layer are designed to improve the learning capabilities of the hidden units. Using the magnification factor defined by the relation, $\beta = 0.7H^{1/N}$ where H is the number of hidden units and N is the number of inputs, the weights are randomly selected in the interval $[-1, 1]$ and then scaled by $v = \beta v/\| v \|$ where v is the first layer weight vector. Results obtained by Pavelka & Procházka (2004), provide significant experimental evidence on the superiority of Nguyen-Widrow's method against typical random initialization techniques.

In addition to the above, a number of interesting methods related to this context have been formulated by Osowski (1993); Chen & Nutter (1991); Yam & Chow (1995, 1997); LeCun (1993); Schmidhuber & Hochreiter (1996), as well as by others researchers.

Despite the availability of such an armory of weight initialization methods, it seems that, there does not exist any, widely accepted, assessment, regarding the effectiveness of these methods with some specific problem or a class of problems. Research efforts concerning the comparison of different weight initialization techniques include those reported in Thimm & Fiesler (1994); Fernández-Redondo & Hernández-Espinosa (2001). Thimm and Fiesler compared several random weight initialization schemes using a very large number of computer experiments. They concluded that the best initial weight variance is determined by the dataset, but differences for small deviations are not significant and weights in the range $[-0.77, 0.77]$ seem to give the best mean performance. Fernández-Redondo & Hernández-Espinosa (2001) presented an extensive experimental comparison of seven weight initialization methods; those reported

by Kim & Ra (1991); Li et al. (1993); Palubinskas (1994); Shimodaira (1994); Yoon et al. (1995); Drago & Ridella (1992). Researchers claim that methods described in Palubinskas (1994); Shimodaira (1994) above proved to give the better results from all methods tested. However, they argue that the method presented in Shimodaira (1994) suffers from the need of pre-processing.

3.2. Analysis of the LIT-Approach

Let us consider a multi-layer perceptron (MLP) with 3 layers, input, hidden and output. Let N, H and O denote the number of nodes of the three layers, respectively. The analysis presented hereafter refers to any node, say j ($1 \leqslant j \leqslant H$), in the hidden layer and so the results apply without any further assumption to every node in the hidden layer. Nodes in the hidden and the output layers are considered to have a sigmoid activation function which is either the logistic function or the hyperbolic tangent. In consequence, the output of any node, say the jth, is given by

$$y_j = \mathrm{sig}(\sum_{i=1}^{N} w_{ji} x_i + w_{jb}), \;\; 1 \leqslant j \leqslant H, \tag{12}$$

while output of a node in the output layer is given by

$$z_k = \mathrm{sig}(\sum_{j=1}^{H} w_{kj} y_j + w_{kb}), \;\; 1 \leqslant k \leqslant O. \tag{13}$$

Note that w_{ji} is the weight of the connection from the ith input node to the jth hidden one. Moreover, w_{jb} and w_{kb} denote the weights of the bias connections to the jth hidden and the kth output nodes respectively.

Sigmoid functions (sig) are able to effectively discriminate between inputs when these inputs lie in the so-called active region of their domain, that is the input range where the derivative of the activation function has a large value. When training the network, in order to avoid problems such as premature saturation, a realistic hypothesis is to start training with such weight values that the node input would be in the active region of the sigmoid function, (Boers & Kuiper, 1992; Yam & Chow, 1997). Then, the training algorithm is responsible to explore the domain of definition of the sigmoid function, in order to determine those values of the weights that minimize the error of

the network output. For any node, say the jth, in the hidden layer having its input in the active region of the sigmoid means that:

$$-a \leqslant \sum_i w_{ji} x_i + w_{jb} \leqslant a, \tag{14}$$

where $-a$ and a are the lower and the upper bounds of the active region of the sigmoid activation function.

Suppose that p patterns are available for training and each pattern is represented by an N-dimensional vector $\mathrm{x} = (x_1, x_2, \ldots, x_N)^\top$. Then expression (14) yields the following linear system of p inequalities with $N + 1$ unknown variables $w_{j1}, w_{j2}, \ldots, w_{jN}, w_{jb}$.

$$\begin{cases} -a \leqslant \sum\limits_i w_{ji} x_i^1 + w_{jb} \leqslant +a \\ -a \leqslant \sum\limits_i w_{ji} x_i^2 + w_{jb} \leqslant +a \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ -a \leqslant \sum\limits_i w_{ji} x_i^p + w_{jb} \leqslant +a \end{cases} \tag{S1}$$

Note that in general, $p > N+1$ and so this system is over-determined and has a solution only if $p - (N + 1)$ pattern vectors are linearly dependent. Problems where the number of features is higher than the number of patterns are known as High Dimension Low Sample Size (HDLSS) problems and constitute a special research topic, (Ahn et al., 2007; Yata & Aoshima, 2010).

Weight initialization methods define symmetric intervals for selecting values of the initial weights. Hence, it is legitimate to assume that each unknown weight w_{ji} is a real number taken from a symmetric interval $[w_{ji}] = [-w_{ji}, w_{ji}]$, $1 \leqslant i \leqslant N$ and $[w_{jb}] = [-w_{jb}, w_{jb}]$ is the symmetric interval for the unknown thresholds. If $[a] = [-a, a]$ denotes the interval for the active range of the activation function of the jth node, then expression (14) may be written in interval form as,

$$\sum_i [w_{ji}] x_i + [w_{jb}] \subseteq [a]. \tag{15}$$

In accordance to subsection 2.3 this relation defines w_{ji} as a solution to the tolerance problem associated with the equation

$$\sum_i [w_{ji}] x_i + [w_{jb}] = [a]. \tag{16}$$

From another point of view, if one considers the p input patterns available for training then this equation expands to the following interval system of linear equations,

$$\begin{cases} [w_{j1}]x_1^1 + [w_{j2}]x_2^1 + \cdots + [w_{jN}]x_N^1 + [w_{jb}] = [a] \\ [w_{j1}]x_1^2 + [w_{j2}]x_2^2 + \cdots + [w_{jN}]x_N^2 + [w_{jb}] = [a] \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ [w_{j1}]x_1^p + [w_{j2}]x_2^p + \cdots + [w_{jN}]x_N^p + [w_{jb}] = [a] \end{cases} \tag{S2}$$

Let us denote this system $\mathbf{X}\mathbf{w}_{\boldsymbol{j}} = \mathbf{a}$, with $\mathbf{X}$ being the $p \times (N+1)$ matrix formed by the $p \times N$ matrix of the pattern values augmented with the p-dimensional vector $(1, 1, \ldots, 1)^\top$ for the threshold, $\mathbf{w}_\mathrm{j} = ([w_{j1}], [w_{j2}], \ldots, [w_{jN}], [w_{jb}])^\top$ is the $(N+1)$-dimensional vector of the unknown weight intervals and $\mathbf{a} = ([a], [a], \ldots, [a])^\top$ the p-dimensional vector of the right hand side. Note that the elements of $\mathbf{X}$ are real numbers which are here considered as point intervals in order to comply with notation of subsection 2.3.

From an algebraic point of view a solution to this interval linear system is an interval vector $\mathbf{w}_{\boldsymbol{a}}$ such that substituting it to (S2) and executing all interval arithmetic operations yields the valid equality $\mathrm{X}\mathbf{w}_{\boldsymbol{a}} = \mathbf{a}$. All the interval vectors that are algebraic solutions to some interval linear equation or a system of equations form a non-empty set. So, speaking about the solution of an interval equation (system of equations, inequalities, etc.) on its own is irrelevant with respect to the specific uncertainty problem modelled by the interval equation (system of equations etc.) at hand (Shary, 2002). The right thing would be to refer to the solution of some specific problem statement relating to the interval equation (system of equations, inequalities, etc.). Hence, what really matters with the solution of the latter interval system is to obtain an interval vector, say $\mathbf{w}_{\boldsymbol{j}}^* = ([w_{j1}^*], [w_{j2}^*], \ldots, [w_{jN}^*], [w_{jb}^*])^\top$, such that for all given patterns $\mathrm{x}^k \in \mathbb{R}^N,\ 1 \leqslant k \leqslant p$ the following relation is valid,

$$\sum_{i=1}^{N} [w_{ji}^*]x_i^k + [w_{jb}^*] \subseteq [a]\,. \tag{17}$$

This statement clearly identifies the system (S2) as the formulation of an interval linear tolerance problem for the initialization of the connection weights to any node in the hidden layer of an MLP.

Different algorithms have been proposed to construct interval solutions to the linear tolerance problem in terms of its *inner interval approximations*, (Shary, 1995; Beaumont & Philippe, 2001). Prior to discussing the existence of an algorithm for deriving a solution for this linear interval tolerance problem we need to discuss the non emptiness of the tolerance solution set of the system (S2).

Lemma 1. *Let the interval linear system* $\mathbf{Ax} = \mathbf{b}$*, where* $\mathbf{A} \in \mathbb{IR}^{\mathrm{m}\times\mathrm{n}}$ *is an m-by-n matrix of real intervals,* $\mathbf{b} \in \mathbb{IR}^{\mathrm{m}}$ *is an m-dimensional vector of real intervals* $\mathbf{b} = \{\mathrm{b}_1, \mathrm{b}_2, \cdots, \mathrm{b}_\mathrm{m}\}$ *and* $\mathbf{x}$ *is the n-dimensional vector of unknown interval variables. If* $0 \in \mathrm{b}_\mathrm{k}$ *for all* $k \in \{1, 2, \cdots, m\}$ *then the tolerance solution of this system is not empty.*

Proof. It is straightforward to see that the trivial m-dimensional vector $\mathbf{t_0} = (0, 0, \cdots, 0)$ is such that $\mathbf{At_0} \subseteq \mathbf{b}$. Thus the tolerance solution set of this system is not empty. □

However, the trivial solution may not be adequate for the problem at hand. To further advance with this issue one may prove the algebraic solvability of the given system (S2) then solve the system and finally select the solutions that are in the tolerance solution set (Shary, 1995). Another way to proceed is a constructive approach which consists in proposing an algorithm for constructing tolerance solutions. A number of approaches are presented in Shary (1995). The proposed LIT-Approach is also a constructive one.

Here let us present the algorithm of Shaidurov using the same notation as given in Shary (1995). Let an interval $m \times n$ matrix $\mathbf{A} = (\mathbf{a_{ij}})$ and an interval right-hand side m dimensional vector $\mathbf{b} = (\mathbf{b_i})$ and let $\sum_{\forall\exists}(\mathbf{X}, \mathbf{b})$ denote the solution set to the associated linear interval tolerance problem.

Algorithm

For some given $t \in \sum_{\forall\exists}(\mathbf{X}, \mathbf{b})$, $t = (t_1, t_2, \cdots, t_n)^\top$ calculate the intervals

$$\mathbf{r}_i = \frac{\mathrm{rad}(\mathbf{b}_i) - |\mathrm{mid}(\mathbf{b}_i) - \sum_{j=1}^{n} \mathbf{a}_{ij} t_j|}{\sum_{j=1}^{n} |\mathbf{a}_{ij}|}, \tag{18}$$

$i = 1, 2, \cdots, m$, and then put $\rho = \min\limits_{1 \leqslant i \leqslant m} \mathbf{r}_i$. The vector $(t + \rho\mathbf{e})$ is a solution to the linear tolerance problem. Note that $\mathbf{e}$ is the interval vector $([-1, 1], [-1, 1], \ldots, [-1, 1])^\top$.

Regarding the tolerance problem for weight initialization the hypothesis of having an initial solution to start with this algorithm can be satisfied by taking the initial vector

t to be the trivial vector $(0, 0, \cdots, 0)$. A similar method proposed by Neumaier (1986) as well as other approaches can be found in Shary (1995). Moreover, a discussion regarding various aspects and optimality criteria of the different algorithms can be found in Shary (1995) and Pivkina & Kreinovich (2006). The question concerning the best approach when solving the weight initialization tolerance problem depends on the performance parameters set for the weight initialization problem itself. We consider that this question has both theoretical and practical importance and needs to be separately addressed outside this paper.

3.2.1. Theoretical results

Hereafter, we present our approach to constructing a solution to the tolerance problem for the initialization of weights. We take advantage of the fact that the intervals are symmetric and build the proposed method based on the following mathematical results omitting the hypothesis of disposing an initial solution vector t. Without loss of generality and for the sake of readability the notation used is the same as above for equations (15)–(17).

Lemma 2. *For any symmetric intervals* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *such that* $\mathbf{w}_1 \subseteq \mathbf{w}_2$ *and any real numbers* x_1 *and* x_2 *such that* $x_1 \leqslant x_2$ *then the relation* $x_1\mathbf{w}_1 \subseteq x_2\mathbf{w}_2$ *is satisfied.*

Proof. The relation $x_1 \leqslant x_2$ implies that $[x_1, x_1] \subseteq [x_2, x_2]$ holds true for the point intervals corresponding to x_1 and x_2. Hence, given that the interval multiplication is inclusion isotonic the relation $x_1\mathbf{w}_1 \subseteq x_2\mathbf{w}_2$ is satisfied. □

Lemma 3. *Suppose that the interval equation* $[x][w] = [a]$*, where* $[a]$ *is a symmetric interval,* $[a] = [-a, a]$*, and* $[x] = [x_L, x_U]$ *with* $0 < x_L \leqslant x_U$*. Then the solution of the equation is* $[-w, w] = [-a, a]/x_U$*.*

Proof. Let us assume that $[w]$ is an interval of the form $[w_L, w_U]$. Then the multiplication operation of intervals implies for $[-a, a]$ that $-a = \min\{x_L w_L, x_L w_U, x_U w_L, x_U w_U\}$ and $a = \max\{x_L w_L, x_L w_U, x_U w_L, x_U w_U\}$. Moreover, the inequality $0 < x_L \leqslant x_U$ implies that $w_L < 0 < w_U$ and so $-a = x_U w_L$ and $a = x_U w_U$. Thus, the solution of the interval equation is $[-w, w] = [-a, a]/x_U$. □

When the coefficient of $[w]$ is not an interval $[x]$ but a finite set of p real numbers $x^1, x^2, \ldots, x^p$ then one may consider this as an interval linear system of p equations of the variable $[w]$. Then, the following Lemma 4 gives a solution to this interval linear system.

Lemma 4. *Suppose that the interval system of p linear equations with one variable* $[w]$ *of the form,* $x[w] = [a]$*, where* $[a]$ *is a symmetric interval,* $[a] = [-a, a]$*, and x is a real number from a set with finite number of elements,* $x \in \mathrm{X} = \{x^1, x^2, \ldots, x^p\}$*. Suppose that* $x^m = \max_{x^k \in \mathrm{X}} |x^k|$*. Then the interval* $[w^m]$ *which is a solution of the interval equation* $x^m[w] = [a]$*, is such that,* $\forall x^k \in \mathrm{X}$*,* $x^k[w^m] \subseteq [a]$*, and hence,* $[w^m]$ *is a member of the tolerance solution set for this interval system.*

Proof. One may observe that, $[w^m] = [-a, a]/x^m$, according to Lemma 3 and considering $x^m = [x^m, x^m]$ to be a point inteval. Given that, $|x^k| \leqslant x^m$, for any $x^k \in \mathrm{X}$, it follows that, $x^k \leqslant x^m$, and $x^k/x^m \leqslant 1$. In consequence, $x^k[w^m] = [-a, a]x^k/x^m \subseteq [-a, a]$. Hence $[w^m]$ is a solution in the tolerance solution set. □

The following proposition is a generalization of the previous Lemma 4 for an interval system of p linear equations with n unknown variables and symmetric right-hand side intervals.

Proposition 1. *Consider the interval system of linear equations of the form,* $x_1[w_1] + x_2[w_2] + \cdots + x_n[w_n] = [a]$*, with* $[a]$ *being a symmetric interval,* $[a] = [-a, a]$*, and each* x_i *a real number from a set with finite number of elements, that is,* $x_i \in \mathrm{X}_i = \{x_i^1, x_i^2, \ldots, x_i^p\} \subset \mathbb{R}$*,* $1 \leqslant \mathrm{i} \leqslant \mathrm{n}$*. In addition, for* $1 \leqslant i \leqslant n$ *let* $x_i^m = \max_{x_l \in \mathrm{X}_i} |x_l|$*, and* $[w^*]$ *be the interval defined by the relation* $[w^*] = [-a, a]/\sum_i x_i^m$*. Then the vector* $\mathbf{w}^* = ([w_1^*], [w_2^*], \ldots, [w_n^*])$ *with* $[w_i^*] = [w^*]$*,* $1 \leqslant i \leqslant n$ *constitutes a solution in the tolerance solution set for this interval system.*

Proof. For every $x_i^k \in \mathrm{X}_i$, $1 \leqslant k \leqslant p$, it stands that $x_i^k \leqslant x_i^m$. Then, according to Lemma 2, the relation $x_i^k[w_i^*] \subseteq x_i^m[w_i^*]$ is valid. So, for any combination of elements of the sets $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n$, we have:

$$x_1^{k_1}[w_1^*] \subseteq x_1^m[w_1^*]$$

$$x_2^{k_2}[w_2^*] \subseteq x_2^m[w_2^*]$$

$$\vdots$$

$$x_n^{k_n}[w_n^*] \subseteq x_n^m[w_n^*]$$

Adding the above relations and given that interval addition is inclusion isotonic we have that,

$$\begin{aligned}
&x_1^{k_1}[w_1^*] + x_2^{k_2}[w_2^*] + \cdots + x_n^{k_n}[w_n^*] \subseteq \\
&x_1^m[w_1^*] + x_2^m[w_2^*] + \cdots + x_n^m[w_n^*] \subseteq \\
&x_1^m[-a,a]\frac{1}{\sum_i x_i^m} + x_2^m[-a,a]\frac{1}{\sum_i x_i^m} + \cdots \\
&\qquad\qquad + x_n^m[-a,a]\frac{1}{\sum_i x_i^m} = \\
&(x_1^m + x_2^m + \cdots + x_n^m)[-a,a]\frac{1}{\sum_i x_i^m} = [-a,a].
\end{aligned}$$

This proves the proposition. □

This proposition applies directly to the interval linear system (S2) above or to $\mathbf{X}\mathbf{w}_j = \mathbf{a}$. Notice that each of the sets X_i corresponds to a column vector of $\mathbf{X}$ and the interval vector $([w_1], [w_2], \ldots, [w_n])^\top$ stands for the interval vector $\mathbf{w}_j$ of the weights to any node j. So the following relation defines a solution to the system (S2).

$$[w_{ji}^*] = [-a, a]/(U + 1), \tag{19}$$

with, $U = \sum_{i=1}^{N} u(i)$, and $u(i) = \max_{1 \leqslant k \leqslant p} (|x_i^k|)$, where $|x_i^k|$ denotes the absolute value of x_i^k. These intervals stand for any weight interval $[w_{ji}]$ as well as for the bias $[w_{jb}]$ and verify relation (17). So, this solution is a member of the tolerance solution set.

3.2.2. Refining the method

The above approach effectively tackles the problem of neural saturation by decoupling the weights from the patterns. This problem has already been addressed by other researchers using mathematically questionable hypotheses (Yam & Chow, 2000). The solution provided by this approach takes into account the outliers for each component of the input sample. However, in practice the random selection of weights reduces the

impact of the input to the hidden node y_j induced by outliers with large values. Recalling the arguments of Wessels & Barnard (1992), the standard deviation of the input y_j to a hidden node is given by $\sigma_{y_j} = (w\sqrt{d_{in}})/3$ where d_{in} is the number of inputs to the node and w defines the interval $[-w, w]$ where the weights are randomly selected from. It is easy to verify that if w is computed using our approach then even for small values of d_{in} (e.g. 5) the value of σ_{y_j} is very small (0.53) and tends to become smaller ($\rightarrow$ 0.13) as d_{in} increases. This means that the intervals computed by the proposed method can be widened while still satisfying the tolerance conditions. Hence, the idea is to "modulate" each interval with respect to the effective range of the input sample and thus differentiate the weight intervals corresponding to different features of the input data. This is achieved by taking into account some statistics of the input data (e.g. the variance).

Let us denote s_{x_i} a statistic providing summary information about the ith input data component x_i such as the third *quartile* ($Q3$) or any *q-quantile* marking the boundary of approximately $4/5$ of the input data. These statistics provide important information about the location of the majority of the input data regardless the distribution of the sample. If the input data display normal distribution then some multiple of the sample standard deviation can be used instead. Given this hypothesis and following definitions of Proposition 1 above we may conclude that $[w^*_{ji}]\mathrm{s}_{x_i} \subseteq [w^*_{ji}]x^m_i$. Equating the two sides of this relation and solving permits to derive the interval

$$[W^*_{ji}] = [w^*_{ji}]x^m_i/\mathrm{s}_{x_i}, \tag{20}$$

which effectively satifies the previous assumptions. Moreover, this relation widens the weight intervals with respect to the majority of the input data and as argued previously it complies "statistically" with the tolerance problem solution.

In the above heuristic using some suitably chosen s_{x_i}, such as $Q3$, to divide the right-hand side of (20) is done in order ensure enlargement of the weight intervals with respect to the majority of the input data. In descriptive statistics, outliers are expected to lie outside the interval $[Q1 - k(Q3 - Q1), Q3 + k(Q3 - Q1)]$ for some nonnegative constant k and $Q3 - Q1$ being the Inter Quartile Range (IQR) (Agresti & Franklin, 2009). Note that typically, for statistical packages such as Minitab and SPSS, $k = 1.5$

(Meyers et al., 2013). So, if the value of an outlier, say x_i^l, is used instead of s_{x_i}, then this oulier should be carrefully chosen otherwise depending on this value the fraction x_i^m/x_i^l in equation (20) tends to one. In consequence, depending on the input data distribution and the outlier used this heuristic will probably result in unnoticeable (i.e. insignificant from practical point of view) enlargement of the weight intervals.

Furthermore, the use of the above heuristic results in defining interval weights whose ranges are inversely proportional to the variance of the corresponding input data components. So, for an input data component, say x_i, with a high variance value, defining a shorter weight interval implies that it is likely to select smaller weight values for this input. In consequence, for some given w_{jb} the intercept $-w_{jb}/w_{ji}$ of the hyperplane defined by a hidden node with the x_i axis (see Figure 1) is more likely to cover the range of values of x_i being positioned inside the majority of the values of the input data distribution, rather than an intercept that passes through the axes origin, or one that lies far away from the values of x_i. On the contrary when the values of x_i have a small variance then the initial weight interval should be larger. This implies that the initial weights are likely to have large values so that the intercept $-w_{jb}/w_{ji}$ is more likely to be in the range of values of x_i, see Figure 1. Moreover, the other benefit expected by defining intervals with variable ranges is to diversify as much as possible the sets of initial weights selected for the hidden nodes. Hence, different nodes tend to define initial hyperplanes whose distance from the origin of the axes given by $|w_{jb}|/\sum_{i=1}^{N} w_{ji}^2$ is as diversified as possible.

Concerning the inital distance of any hyperplane from the origin of the axes we need to note that $0 \leqslant |w_{jb}|/\sum_{i=1}^{N} w_{ji}^2$. The effect of widening produced by (20) on the weight intervals tends to move the hyperplanes towards the beginning of the pattern axes as it tends to increase the denominator in the distance formula. On the other hand, theoretically there is no upper bound for this distance. This is a common issue to all weight initialization techniques that randomly select initial weights from some interval defined around 0 with very small real values. In our approach this may occur if all weights are selected from extremeley narrow symmetric intervals which in their turn are computed if the interval $[-a, a]$ is divided by a big number corresponding to the quantity $U + 1$, when the problem at hand has a huge number of features. However, as

we will show later in section 4 even in the case of a real life problem such as the MNIST dataset (LeCun et al., 2004) with 784 features the algorithm demonstrates a very interesting behaviour outperforming other weight initialization techniques. A thorough study of the UCI repository of machine learning database (Frank & Asuncion, 2010) shows that problems with a very big number of features are treated as dimensionality reduction or feature extraction ones before being considered as classification or regression problems.

The above considerations and the results obtained are valid for continuous valued input patterns. For some input x_i which is binary or a constant value then $\mathrm{s}_{x_i} = 0$. This constitutes a major incovenience as it results in a division by 0 for the fraction x_i^m/s_{x_i} in equation (20). To avoid this problem we choose to leave the interval $[w_{ji}^*]$ unchanged by imposing $\mathrm{s}_{x_i} = 1$. For this we require $\beta \leqslant \mathrm{s}_{x_i}$ where this lower bound is defined as $\beta = 0.1$. Whenever $\mathrm{s}_{x_i} < \beta$ we impose $\mathrm{s}_{x_i} = 1$. The following formula summarizes the rule for computing s_{x_i} and applies whenever this quantity is used in this paper.

$$\frac{1}{2}\{\mathrm{sgn}\,\{\mathrm{s}_{x_i} - \beta\} + 1\}\mathrm{s}_{x_i} - \frac{1}{2}\{\mathrm{sgn}\,\{\mathrm{s}_{x_i} - \beta\} - 1\} \tag{21}$$

where sgn denotes the sign function. This choice introduces a kind of "discontinuity" which can be avoided if one chooses $\mathrm{s}_{x_i} = \beta$. However, even this option is still a heuristic one. In a future correspondence we could investigate the possibility to adaptively define β as an interval derived by the data and discuss the impact of such a formulae on specific experiments. In the present research the benchmarks and real world problems tackled provide no hints as to which is the optimum formula for s_{x_i} definition in this specific case.

Typically, normalization or scaling is applied (Bishop, 1995) so that the input samples are in the interval $[-1, 1]$, mainly in order to facilitate training (LeCun, 1993). These operations normally do not alter the status of the input data. So, the previous considerations remain valid and the use of the term s_{x_i} for properly modulating the original weight intervals $[w_{ji}^*]$ still applies after normalization or scaling of the input data. For the rest of this paper, we assume that the values of the input patterns are normalized to be in the interval $[-1, 1]$ or the interval $[0, 1]$. Under these hypotheses we may state that the relation $[w_{ji}^*]\mathrm{s}_{x_i} \subseteq [w_{ji}^*]1$ is valid and suggests that solving the

following equations:

$$\left[W_{ji}\right] \mathrm{s}_{x_i} = [w^*_{ji}], \quad 1 \leqslant i \leqslant N \tag{22}$$

permits to define the intervals,

$$\left[W^*_{ji}\right] = [w^*_{ji}]\frac{1}{\mathrm{s}_{x_i}}, \quad 1 \leqslant i \leqslant N \tag{23}$$

that obviously satisfy the relation,

$$\begin{aligned} [W^*_{j1}]\mathrm{s}_{x_1} + [W^*_{j2}]\mathrm{s}_{x_2} + \cdots + [W^*_{jN}]\mathrm{s}_{x_N} + [w^*_{jb}]1 = \\ [w^*_{j1}] + [w^*_{j2}] + \cdots + [w^*_{jN}] + [w^*_{jb}] = \\ = [a] \end{aligned} \tag{24}$$

Hence, the interval vector $\mathbf{W}^*_j = ([W^*_{j1}], [W^*_{j2}], \ldots, [W^*_{jN}], [w^*_{jb}])^\top$ is a solution in the tolerance solution set of the interval system (S2). Recall that s_{x_i} is computed using formula (21).

3.2.3. Initializing hidden-to-hidden and hidden-to-output layer connection weights

The analysis presented above focuses on effective initialization of weights of the input-to-hidden layer connections. Earlier implementations of a complete algorithm were based on minimal assumptions regarding the initial values of weights for hidden-to-hidden and hidden-to-output layer connections, that is, random selection of values in the interval $[-1, 1]$. This choice gave rather satisfactory results in the case of small sized networks and datasets, see section 4, suites 1 and 2 of experiments. In order to define a full scale algorithm for initializing weights of any MLP two issues are considered here. The first deals with saturation of the nodes in any hidden layer, while the second defines an order of magnitude for the weights of connections leading to ouput layer nodes.

In order to avoid saturation of any node in the kth hidden layer we adopt the hypotheses of the previous analysis. This means that weights of connections linking a node in the hidden layer k with the ouputs of nodes in the layer $k-1$ are randomly selected in the interval $[-a_k/(H_{k-1}+1), a_k/(H_{k-1}+1)]$, where H_{k-1} is the number of nodes of the layer $k-1$ and a_k is the active range of the activation function of the node

in the hidden layer k. The previous formula for nodes in the hidden layer k is derived considering that the outputs of the layer $k-1$ have a maximum value equal to 1. In practice, instead of $(H_{k-1}+1)$ the value of H_{k-1} can be used without any difference regarding the training performance.

For the weights of the hidden-to-output connections different approaches are proposed by different researchers (subsection 3.1). In order to optimize the choice of these weights we used the formula $[-3A/\sqrt{d_{in}}, 3A/\sqrt{d_{in}}]$ introduced in Wessels & Barnard (1992) where instead of d_{in} we set H for the number of hidden layer nodes. The authors in that paper determined the value of the scale factor $A = 1$ through experiments with small sized networks. We adopted the same approach but we also experimented with networks with a higher number of nodes in the hidden layer. For these networks when $A = 1$ the fraction $3A/\sqrt{H}$ becomes too small yielding extremely narrow weight intervals for the hidden-to-output layer connections which slow the training process. By gradually increasing the value of A we observed that the network performance improved and so we came up with the following rule of thump.

The value of $A = 1$ is valid for networks with a relatively small number of nodes in the hidden layer i.e. $H \lessapprox 30$. For medium to larger sized networks i.e. $H > 30$ the best network performance was observed when $A > 1$. Experimented with $H = 36$ we found that $A \approx 1.2$ and $A \approx 3$ for $H = 300$. Finally, for $H = 650$ we noticed that A should be set to 4 for nodes with the logistic sigmoid activation function while for nodes with the hyperbolic tangent this value should be $A \approx 2$. We cannot guarantee that these results are optimal for every considered dataset. However, the resulting intervals roughly confirm the findings for the weight intervals reported in Nguyen & Widrow (1990) and Thimm & Fiesler (1994). To the best of our knowledge there is no specific study on this matter in the literature and in light of these results this should constitute an interesting point for deeper investigation.

3.3. Algorithm and Discussion

3.3.1. Algorithm Description

The algorithm implementing the above approach computes one specific interval $[W^*_{ji}]$ for each component i of the input data as well as the interval $[w^*_{jb}]$ for the thresh-

old. Thus, $n+1$ intervals are computed once and they are used for selecting the weights of any node in the hidden layer.

Input Data Coding

1. Continuous input data are scaled to be in the interval $[-1, 1]$ (or $[0, 1]$). Binary variables are set to $\{-1, 1\}$ (or $\{0, 1\}$).
2. For each continuous valued input data variable x_i compute the third quartile Q_3 and set $s_{x_i} = Q_3$. If x_i displays normal distribution compute the sample standard deviation σ_{x_i} and set $s_{x_i} = 2\sigma_{x_i}$. If x_i can be approximated by the normal distribution then $s_{x_i} = k\sigma_{x_i}$ for some suitably chosen k. If x_i is not continuous then Apply rule (21) above.
3. Define the value of the parameter a for the bounds of the active region interval $[-a, a]$ depending on the type of the activation function of the jth node, see subsection 3.3.2 hereafter.

Computing Weights of Input to Hidden Layer Connections

4. For each node j in the hidden layer and any input connection i the weight w_{ji} is randomly selected with uniform distribution from the interval $[W^*_{ji}]$ defined using relation (23) above.
5. For each node j in the hidden layer the weight w_{jb} of the bias is randomly selected with uniform distribution from the interval $[w^*_{jb}]$ defined using relation (19) above.

Computing Weights of Connections from Hidden Layer $(k-1)$ *to Hidden Layer* (k)

6. These weights are random numbers selected to be uniformly distributed in the interval $[-a_k/H_{k-1}, a_k/H_{k-1}]$ as defined in the previous subsection.

Computing Weights of Hidden to Output Layer Connections

7. Weights of the hidden to the output layer connections are random numbers selected to be uniformly distributed in the interval $\left[-3A/\sqrt{N}, 3A/\sqrt{N}\right]$ where the scale factor A is defined in the previous subsection.

3.3.2. Discussion

Step **3** of the algorithm requires setting the bounds of $[-a, a]$ for the active region of the sigmoid activation function. This interval is assumed to be the region where the derivative of the sigmoid activation function is greater than or equal to $0.04D_{max}$ or $0.05D_{max}$ where D_{max} denotes the maximum magnitude of the derivative of the sigmoid, (Yam & Chow, 2000, 2001). For example in case a logistic sigmoid activation function is used then $a = 4.59$ or $a = 4.34$. For the experiments shown in this paper the values adopted are those defined by the Neural Network Toolbox of MATLAB, that is, $a = 4$ for the logistic sigmoid and $a = 2$ for the hyperbolic tangent. These values are computed for $\lambda = 1$ where λ is the slope parameter of the sigmoid activation function.

Most of the issues pertaining the fomulation of the LIT-Approach were analyzed and resolved in earlier subsections. Here we will briefly refer to the ability of the proposed method to cope with *prematurely saturated units* and *symmetry breaking*. These matters are reported in the literature (Hassoun, 1995; Haykin, 1999) as troubles of neural network training that need to be addressed by weight initialization. Regarding *premature saturation of the units* the proposed method by default defines initial weights which prevent saturation of the hidden nodes at an early stage of training. In addition, *symmetry breaking* that is preventing nodes from adopting similar functions is addressed using random weight selection from intervals with different bounds.

Besides these matters, Wessels & Barnard (1992) note that another problem is what they call *false local minima* for which they name three possible causes. These are the following: *Stray hidden nodes*, that is nodes defining initial decision boundaries which have been moved out of the region of the sample patterns. *Hidden nodes* having *duplicating function* are the nodes that define separating hyperplanes having the same initial position and orientation. Finally, *dead regions* in the pattern space are created when in these regions the hidden nodes are arranged so that they all happen to be inactive, that is, there are no hyperplanes defined by the hidden nodes inside these regions. The LIT-Approach tackles these issues based on the way it defines the weight intervals. The issues regarding *stray hidden nodes* and *dead regions* are sufficiently addressed based on the way the LIT-Approach defines the initial weight intervals and then on the way

the hidden nodes define the initial hyperplanes to be in the heart of the pattern data, see subsection 3.2.2. Moreover, hidden nodes are not likely to have *duplicating function* due to the random weight selection. The LIT-Approach, while not specifically designed to tackle these specific problems, it, however, addresses them efficiently as shown by the results of the experiments hereafter. Based on the advantages of distributions such as those proposed in Sonoda & Murata (2013) there might be improvements concerning how the LIT-approach tackles random weight selection, now defined by uniform distribution.

Finally, we need to note that the proposed approach does not intend to deal with the problem of structural local minima in the weight space. This issue concerns the training phase of an MLP and it has effectively been tackled in Magoulas et al. (1997).

4. Experimental Evaluation

In order to assess the effectiveness of the proposed method we designed and conducted three different suites of experiments. The first suite deals with the comparison of the performance of the proposed method against six different weight initialization methods which are based on random selection of initial weights from predefined intervals. The benchmarks used for this first suite mainly concern classification problems, while one of them deals with regression and a second with prediction of a highly nonlinear phenomenon. Moreover, a number of experiments were executed on function approximation and they are presented in a separate subsection. The second suite constitutes a thorough comparison of the proposed LIT-Approach with the well known initialization procedure proposed by Nguyen & Widrow (1990).

The performance measures considered for all experiments are: the convergence success of the training algorithm, the convergence rate and the generalization performance achieved for the test patterns. The convergence success of the training algorithm is the number of initial weight sets for which the training algorithm reached the predefined convergence criteria. The convergence rate is the number of epochs needed for the training to converge. For benchmarks with continuous valued output, generalization performance is computed using the mean absolute error of the output of the network

and the target output, and for classification benchmarks, generalization is defined as the percentage of successfully classified previously unknown test patterns. The analysis of the experimental results was carried out using the statistical analysis package SPSS v17.0 (Green & Salkind, 2003), STATService 2.0 (Parejo et al., 2012) and the R statistical computing environment.

Hereafter, training a network for some specific benchmark with initial weights selected using some weight initialization method is called a trial. A training experiment is a set of trials corresponding to training the network for some specific benchmark using a set of initial weights selected by the same weight initialization method.

4.1. Suite 1 of Experiments

4.1.1. Experimental Setup

This suite of experiments was set up in order to investigate the efficiency of the proposed approach on a relatively broad spectrum of real world problems. Comparison is done against the following (in alphabetical order of the abbreviations used) well known weight initialization methods; BoersK, (Boers & Kuiper, 1992), Bottou, (Bottou, 1988), KimRa, (Kim & Ra, 1991), NW, (Nguyen & Widrow, 1990), SCAWI, (Drago & Ridella, 1992), and Smieja, (Smieja, 1991).

The real world problems adopted for the experiments are benchmarks reported in various research papers used to compare performance of different weight initialization methods, as for example Fernández-Redondo & Hernández-Espinosa (2001); Thimm & Fiesler (1997); Yam & Chow (2000, 2001). These real world problems are briefly described in the following paragraph. Detailed description and more information can de found in the UCI repository of machine learning database (Frank & Asuncion, 2010) and references cited therein.

1. Auto MPG prediction (inputs:7, outputs:1). This dataset concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multi-valued discrete and 4 continuous attributes. The number of instances is 398. Six patterns with missing values have been removed.

2. British language vowels recognition (inputs:10, outputs:11). As stated in the benchmark summary, this is a speaker independent recognition problem of the

eleven steady state vowels of British English using a specified training set of 10 linear prediction coefficients derived log area ratios. The original dataset comprises 991 instances pronounced by different speakers. A subset containing the first 330 instances were retained for training and testing.

3. Glass identification (inputs:9, outputs:1). Based on 9 attributes, this classification of types of glass was motivated by criminological investigation. The dataset used is the *glass2* downloadable from PROBEN1 (1994) ftp site. It consists of 214 instances already pre-processed and so there are no missing values.

4. Servo prediction (inputs:12, outputs:1). Originally this benchmark was created by Karl Ulrich (MIT) in 1986 and refers to a highly non-linear phenomenon that is predicting the rise time of a servomechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages. The dataset consists of 167 patterns and has no missing values.

5. Solar sunspot prediction (inputs:12, outputs:1). The dataset contains the sunspot activity for the years 1700 to 1990. The task is to predict the sun spot activity for one of those years given the activity of the preceding twelve years. A total of 279 different patterns are derived from the raw data.

6. Wine classification (inputs:13, outputs:3). These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The dataset contains 178 instances and has no missing values.

The original datasets were preprocessed to eliminate duplicate patterns and values were scaled to match requirements set by the weight selection procedures. These operations were performed according to PROBEN1 guidelines, (Prechelt, 1994). Unless otherwise stated, the datasets were partitioned to training sets using approximately 75% of the patterns and to test sets using the remaining 25%. For the Servo prediction benchmark the training set was made using 84 patterns and the test set using 83 patterns. The training and the test sets are defined once and used for all experiments. During network training the patterns of the training set are presented in the same order using the *trains*

i.e. the online sequential training procedure of MATLAB.

A total number of 42 experiments were set up for these 6 problems and the 7 weight initialization methods. Each experiment was carried out using a set of 100 initial weight vectors, selected by the corresponding method. The same network architecture was initialized with these vectors and trained using online BP. The network architecture and the training parameters, used in this arrangement, are reported in Table 1. These parameters are similar to those found by Thimm & Fiesler (1994).

a) Learning rate is the rate used by the vanilla BP online algorithm.

b) Convergence criterion is either the goal set for the minimization of the error function or the minimum percentage of the training patterns correctly classified by the network.

c) Max cycles denote the maximum number of BP cycles. During a cycle all training patterns are presented to the network in random order and weights are updated after every training pattern. Training stops when Max cycles number is reached.

d) Input data scale indicates the interval used by all weight initialization algorithms except the Nguyen-Widrow algorithm, which scales input data values in the interval $[-1, 1]$.

Table 1: Architectures of networks and training parameters used for the Suite 1 of the experiments

Benchmark	Network architecture	Activation function†	Learning rate	Convergence criterion	Max cycles	Input data scale
Auto-MPG	7-3-1	logsig	0.3	0.01	500	[-1,1]
British Vowels	10-20-11	tansig	0.05	90%	800	[-1,1]
Glass	9-10-8-6	logsig	0.6	0.04	800	[0,1]
Servo	12-3-1	logsig	0.1	0.008	500	[-1,1]
Solar	12-5-1	logsig	0.3	0.005	500	[0,1]
Wine	13-6-3	tansig	0.2	95%	500	[-1,1]

† logsig denotes the logistic sigmoid function and tansig is the hyperbolic tangent

4.1.2. Analysis of the Results

Tables 2, 3 and 4 report the experimental results on the benchmarks considered for the aforementioned performance measures. A quick look at these results shows that the proposed approach improves network performance for all parameters.

The comparison of the efficiency of the different initialization methods is based

Table 2: Convergence success results in 100 trials for the Suite 1 of the experiments

Benchmark	Initialization Algorithms						
	BoersK	Bottou	KimRa	LIT-A	NW	SCAWI	Smieja
Auto-MPG	100	100	100	100	100	100	100
British Vowels	91	100	100	100	91	100	100
Glass	22	21	0	72	12	0	0
Servo	100	100	100	100	99	100	100
Solar	100	100	100	100	92	100	100
Wine	100	100	100	100	100	100	100

on the statistical analysis of the results obtained. In order to evaluate the statistical significance of the observed performance one-way ANOVA (Green & Salkind, 2003) was used to test equality of means. ANOVA relies on three assumptions: independence, normality and homogeneity of variances of the samples. This procedure is robust with respect to violations of these assumptions except in the case of unequal variances with unequal sample sizes, which is true for the Glass benchmark as the larger group size is more than 1.5 times the size of the smaller group.

The validity of the normality assumption is omitted and Levene's test for testing equality of variances is conducted, (Ramachandran & Tsokos, 2009). Homogeneity of variances is rejected for all cases by Levene's test and so Tamhane's posthoc procedure, (Ramachandran & Tsokos, 2009), is applied to perform multiple comparisons analysis of the samples. The significance level set for these tests is $\alpha = 0.05$. The p-value (*Sig.*) indicated for each initialization method, concerns comparison with the proposed method and the mean value is marked with an $*$ when equality of means is rejected p-value < 0.05. The analysis of the Glass benchmark results was performed pairwise between successful initialization methods using the Mann-Whitney test.

Table 5 reports for each initialization method how many times a method delivers

Table 3: Convergence rate results for the Suite 1 of the experiments

Benchmarks		Initialization Algorithms						
		LIT-A	BoersK	Bottou	KimRa	NW	SCAWI	Smieja
Auto-MPG	*Mean*	2.19	2.51	2.38	2.97*	4.73*	2.34	2.26
	St.D.	0.46	1.03	0.86	0.39	2.88	0.59	0.52
	Sig.		0.106	0.690	0.000	0.000	0.638	1.000
British Vowels	*Mean*	471.29	521.88*	476.99	487.96	517.77*	468.83	476.58
	St.D.	41.59	94.16	46.15	36.50	79.28	54.82	53.82
	Sig.		0.000	1.000	0.060	0.000	1.000	1.000
Servo	*Mean*	88.07	87.31	86.37	124.07*	77.17	96.02*	86.88
	St.D.	8.66	14.75	12.87	3.14	35.29	9.02	8.96
	Sig.		1.000	0.999	0.000	0.071	0.000	1.000
Solar	*Mean*	142.67	150.26	159.74*	169.88*	233.05*	160.15*	158.08*
	St.D.	30.73	28.66	30.90	18.33	91.24	29.87	29.74
	Sig.		0.794	0.003	0.000	0.000	0.001	0.008
Wine	*Mean*	11.42	10.57*	10.73*	11.65	10.50	11.08	11.26
	St.D.	1.10	1.68	1.42	0.98	5.49	1.32	1.52
	Sig.		0.001	0.004	0.932	0.899	0.658	1.000
Glass	*Min*	600	616	660	–	563	–	–
	Max	799	785	795	–	789	–	–
	Median	721.00	730.50	759.00*	–	718.00	–	–
	Sig.		0.214	0.001	-	0.975	–	–

* denotes that the mean value of the initialization method is significantly different from the mean value of LIT-A using the indicated *p*-values (*Sig.*) computed by the posthoc analysis of the ANOVA results.

– denotes that the initialization method failed to meet the convergence criteria exceeding the maximum number of cycles in all trials.

superior, equal or inferior performance when compared (pairwise comparisons) with all other methods regarding convergence rate and generalization. The advantage offered by the proposed method to achieve better convergence rate is manifested by these results. So, performance of a neural network when weights are initialized with the proposed method is superior in 42% of the cases. In 50% of the cases performance is the same with all other methods and only in 6% the proposed method delivers inferior performance to the training algorithm.

In terms of generalization the proposed method though having a marginally better

score when compared to the method of Kim and Ra proves to be better than all the other methods in all benchmarks, except in the case of the Glass benchmark, see Table 4. For the Glass benchmark the generalization performance seems to be better for the methods of Boers-Kuiper, Bottou and Nguyen-Widrow compared to our LIT-A method. However, one should also take into account the number of successful experiments for each method. Generalization "achieved" by the proposed method is superior in 47% of the cases, while in 42% of the cases performance is the same with other methods and only in 11% of the cases the proposed method delivered inferior performance to the training algorithm. Only the method of Kim and Ra seems to give similar performance with the proposed LIT-Approach.

4.1.3. Non-parametric Statistical Analysis and Posthoc Procedures

In order to comply with reported best practice in the evaluation of the performance of neural networks, (Luengo et al., 2009; García et al., 2010; Derrac et al., 2011), we evaluated the statistical significance of the observed performance results applying the Friedman test. This test ranks the performance of a set of k algorithms and can detect a significant difference in the performance of at least two algorithms. More specifically, the Friedman test is a non-parametric statistical procedure similar to the parametric two-way ANOVA used to test if at least two of the k samples represent populations with different medians. The null hypothesis H_0 for Friedman's test states equality of medians between the populations while the alternative hypothesis H_1 is defined as the negation of the null hypothesis.

Table 6 uses two subtables to depict the average rankings computed through the above statistical test for the convergence rate and the generalization performance. At the bottom of each subtable we give the statistic of each test along with the corresponding p-value. The p-values computed strongly suggest rejection of the null hypothesis at the $\alpha = 0.05$ level of significance. This means that the initialization algorithms have some pattern of larger and smaller scores (medians) among them i.e. there exist significant differences among the considered algorithms.

The significant differences detected by the above test procedure concern the overall comparison of the algorithms as a set entailing that the performance of at least one

Table 4: Generalization performance results for the Suite 1 of the experiments

Benchmarks		Initialization Algorithms						
		LIT-A	BoersK	Bottou	KimRa	NW	SCAWI	Smieja
Auto-MPG	*Mean*	0.0743	0.0750	0.0748	0.0738	0.0778*	0.0739	0.0741
	St.D.	0.0041	0.0052	0.0047	0.0018	0.0073	0.0043	0.0041
	Sig.		0.999	1.000	1.000	0.001	1.000	1.000
British Vowels	*Mean*	96.23%	93.72%*	96.29%	96.27%	94.52%*	95.86%	96.18%
	St.D.	1.60%	2.85%	1.37%	1.18%	2.70%	1.81%	1.45%
	Sig.		0.000	1.000	1.000	0.000	0.933	1.000
Servo	*Mean*	0.0792	0.0854*	0.0839*	0.0797*	0.0969*	0.0842*	0.0801*
	St.D.	0.0010	0.0055	0.0045	0.0002	0.0145	0.0049	0.0020
	Sig.		0.000	0.000	0.000	0.000	0.000	0.002
Solar	*Mean*	0.0924	0.0924	0.0919	0.0876*	0.1025*	0.0911	0.0917
	St.D.	0.0040	0.0030	0.0030	0.0019	0.0107	0.0029	0.0034
	Sig.		1.000	0.999	0.000	0.000	0.160	0.980
Wine	*Mean*	99.84%	98.98%*	99.38%*	99.80%	98.69%*	99.58%	99.40%*
	St.D.	0.57%	1.20%	1.05%	0.64%	1.48%	0.93%	1.22%
	Sig.		0.000	0.003	1.000	0.000	0.282	0.025
Glass	*Min*	59.62%	67.31%	65.38%	-	65.38%	-	-
	Max	71.15%	75.00%	75.00%	-	75.00%	-	-
	Median	65.38%	71.15%*	69.32%*	-	69.23%*	-	-
	Sig.		0.000	0.000	-	0.000	-	-

* denotes that the mean value of the initialization method is significantly different from the mean value of LIT-A using the indicated p-values (*Sig.*) computed by the posthoc analysis of the ANOVA results.

– denotes that the initialization method failed to meet the convergence criteria exceeding the maximum number of cycles in all trials.

initialization algorithm differs from the others. However, the Friedman test cannot provide information on which algorithms are different from the others and so a multiple comparison analysis needs to be conducted. For the sake of our evaluation we need to carry out a multiple comparisons analysis between performance of the LIT-Approach and performance of each one of the other initialization methods. This is a multiple comparisons (pairwise) analysis (Derrac et al., 2011) with a control algorithm which results in formulating $k - 1$ hypotheses one for each of the $k - 1$ comparisons, where in our case $k = 7$. A better performance for the convergence rate of an algorithm trans-

Table 5: Summary of pairwise comparisons score for each method for the Suite 1 of the experiments

Initialization method	Convergence rate			Generalization		
	Superior	Equal	Inferior	Superior	Equal	Inferior
BoersK	12	19	5	6	17	13
Bottou	12	20	4	10	20	6
KimRa	4	12	20	16	15	5
LIT-A	15	19	2	17	15	4
NW	6	13	17	4	5	27
SCAWI	7	20	9	8	20	8
Smieja	8	24	4	9	20	7

lates here to a smaller number of epochs and a better performance for generalization is taken to be a smaller classification or approximation error. So, the objective of the tests is minimization and in consequence the control procedure is automatically selected to be the algorithm with the lowest ranking score. This algorithm is LIT-Approach for both performance measures, see Table 6.

For the non-parametric test (Friedman) used we consider the ranking scores com-

Table 6: Average ranking achieved by the Friedman test (Suite 1 of the experiments)

Initialization method	Convergence Rate	Generalization
BoersK	3.75	4.69
Bottou	3.71	4.02
KimRa	5.15	3.33
LIT-A	3.15	2.81
NW	4.40	5.26
SCAWI	4.03	4.08
Smieja	3.82	3.82
Statistic	305.76	511.50
p-value	0.12e-09	0.23e-09

puted for each algorithm. Then the posthoc analysis aims in determining if the difference between the ranking score of the proposed LIT-Approach and the ranking score of each of the other algorithms are significantly different. The test statistic z and the corresponding p-value for comparing LIT-Approach and each of the other algorithms are computed using the online STATService (Parejo et al., 2012) environment. The p-value

(2-tailed) corresponding to the z-statistic of each comparison is determined using normal approximation and can be compared with some appropriate level of significance α.

However, these p-values are not suitable for multiple comparisons as they do not account for the Family-Wise Error Rate (FWER) produced by accumulation of Type I error in the case of a family of hypotheses associated with the multiple comparisons tests (Derrac et al., 2011). To cope with this matter, instead of using posthoc procedures to adjust the level of significance α, we choose to compute the adjusted p-values (APVs) corresponding to the Holm (Bonferroni-Holm) and the Benjamini-Hochberg adjustment methods. Information on these adjustment methods can be found in R-Documentation (2013) and references cited therein. These APVs can be used to test the corresponding hypotheses i.e. to compare corresponding algorithms directly with any significance level α and give a "metric" of how different these algorithms are (Luengo et al., 2009).

The unadjusted and the adjusted p-values for the pairwise comparison of the proposed algorithm with each one of the other methods are presented in Table 7 for both convergence rate and generalization. Note that the precision retained for the p-values given in this Table and all similar Tables hereafter is up to the fourth decimal digit. The adjusted p-values for the Friedman test in this Table show significant difference between the ranking of the LIT-Approach and the other methods for both convergence rate and generalization. This translates to an improvement of the LIT-Approch over all the other weight initialization algorithms.

The computations necessary for Table 6 as well as for all similar Tables hereafter in this paper were carried out using STATService (Parejo et al., 2012). Computations for Table 7 as well as for all similar Tables hereafter were executed using the R environment for Statistical Data Analysis. Finally, it is worth noting that the results obtained using the non-parametric statistical analysis confirm those provided by ANOVA.

4.1.4. Comments and Remarks

Despite the stochastic nature of the training scheme adopted for this suite we may argue that the results obtained are suggestive of the potential offered by the proposed

Table 7: p-values of multiple comparisons (Suite 1 of the experiments)

Initialization algorithm	Unadjusted	Adjusted	
		Bonferroni-Holm	Benjamini-Hochberg
	Convergence rate (Control algorithm is LIT-A)		
BoersK	0.0000	0.0000	0.0000
Bottou	0.0000	0.0000	0.0000
KimRa	0.0000	0.0000	0.0000
NW	0.0000	0.0000	0.0000
SCAWI	0.0000	0.0000	0.0000
Smieja	0.0000	0.0000	0.0000
	Generalization (Control algorithm is LIT-A)		
BoersK	0.0000	0.0000	0.0000
Bottou	0.0000	0.0000	0.0000
KimRa	0.0000	0.0000	0.0000
NW	0.0000	0.0000	0.0000
SCAWI	0.0000	0.0000	0.0000
Smieja	0.0000	0.0000	0.0000

method. In terms of convergence success (Table 2) the proposed method seems to contribute to the best score for the training algorithm. Moreover, the advantage offered by the proposed method to achieve better convergence rate is manifested by the results given in Table 3 and Table 5. Lastly, one may easily notice that in terms of generalization performance the proposed method though marginally superior when compared, using ANOVA, to the Kim-Ra it proves to be better than all the other methods in all benchmarks, except the Glass benchmark, see Tables 4 and 5. These conclusions are strongly supported by the non-parametric statistical analysis with the Friedman test. Though these results are indicative and for comparison purposes, they provide significant evidence regarding the efficiency of the proposed method.

4.2. Function approximation

4.2.1. Setup of the Experiments

In order to cover the whole range of problems for which MLPs are used, one needs to consider the problem of approximating analytically defined non-linear functions. This constitutes a necessary prerequisite for a "fair" comparison first and foremost with the method of Nguyen and Widrow, as these researchers initially demonstrated

their method on a function approximation problem. The functions used as benchmarks are defined in the following paragraphs.

Function 1. The function for this benchmark is the one reported in the original paper of Nguyen & Widrow (1990),

$$y = 0.5 \sin\left(\pi x_1^2\right) \sin\left(2\pi x_2\right).$$

The network used here, is a 3-layer (2-21-1) architecture with the hyperbolic tangent activation function for the hidden layer nodes as well as for the output node. A total of 625 (= 25 × 25) points are randomly selected, using uniform ditribution, in the interval $[-1, 1] \times [-1, 1]$. Among these points 450 are used for training and 175 for testing the network.

Function 2. The function considered here is a variant of the function considered in Yam & Chow (2000, 2001). This function is a mapping of eight input variables, taken in the interval [0, 1], into three output ones defined by the following three equations:

$$\begin{aligned} y_1 &= \left(x_1 x_2 + x_3 x_4 + x_5 x_6 + x_7 x_8\right)/4 \\ y_2 &= \sqrt{\left(x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8\right)}/8 \\ y_3 &= \left(1 - y_1\right)^{1/3} \end{aligned}$$

For this benchmark, a 8-12-3 network architecture was used with logistic activation functions for nodes of the hidden and the output layer. A set of 75 patterns is formed by randomly sampling, with uniform distribution, values for the input variables and calculating output values. Among these input-output patterns, 50 are used for training the network and 25 for testing, as in Yam & Chow (2001).

Function 3. The function used here is a real-valued non-linear function of two variables, taken in the interval $[-1, 1]$, defined by the formula:

$$y = \sin\left(2\pi x_1 x_2\right)/\left(2\pi x_1 x_2\right).$$

A 3-layer network architecture with 30 nodes in the hidden layer was adopted for this benchmark. All nodes in the hidden layer as well as the output node have a hyperbolic

tangent activation function. The training set is formed by taking 320 patterns of the total 400 (= 20×20) that are randomly selected using uniform distribution. The rest 80 patterns constitute the test set.

A total of 21 training experiments were executed for the above 3 functions and the 7 (LIT-A plus other six) weight initialization methods considered in this section. Each training experiment is made up of a hundred (100) initial weight vectors derived using one of the weight initialization methods. Networks in all experiments are trained using the Levenberg-Marquardt method (LM), (Hagan et al., 1996; Marquardt, 1963; Hagan & Menhaj, 1994). The performance goal for the network output error is set to $1.0e-04$ for *Functions 1* and *2*, and $1.0e-03$ for *Function 3*. If the performance goal is not met when a maximum number of 1000 epochs is reached then the training stops. The learning rate for all experiments of *Function 1* is set to 0.1 and 0.5 for the other two benchmarks. Results of the training experiments are reported in Tables 8 – 10 hereafter.

For the benchmarks $Function1$ and $Function3$ LIT-A was applied using $1.5\sigma_{x_i}$ for the term s_{x_i}. This choice is based on the assumption that the input data are approximately normally distributed, and therefore s_{x_i} in the LIT-Approach was "roughly" approximated using $1.5\sigma_{x_i}$ instead of the third quartile $Q3$.

4.2.2. Analysis of the Results

The results obtained regarding the performance measures set are shown in Tables 8–10. A rough observation of these results shows that the proposed LIT-Approach remains on top of the other methods as in the previous suite 1 concerning convergence rate (Table 9) while being among the best methods regarding convergence succes (Table 8) and generalization (Table 10).

Comparison between the performance of the different initialization methods, for

Table 8: Convergence success results for the Function Approximation benchmarks

Benchmarks	Initialization Methods						
	BoersK	Bottou	KimRa	LIT-A	NW	SCAWI	Smieja
Function 1	73	91	82	92	74	85	82
Function 2	100	100	100	100	76	100	100
Function 3	81	100	100	84	81	91	47

Table 9: Convergence rate results for the Function Approximation benchmarks

Benchmarks		Initialization Algorithms						
		BoersK	Bottou	KimRa	LIT-A	NW	SCAWI	Smieja
Function 1	*Mean*	379.79	504.70	538.73	337.05	275.70	333.72	401.71
	St.D.	217.57	177.66	164.74	198.95	212.12	184.73	225.23
Function 2	*Mean*	6.89	6.87	8.93	6.49	69	7.23	7.36
	St.D.	1.61	1.54	2.25	1.40	191.28	1.95	2.34
Function 3	*Mean*	60.89	16.74	23.48	18.57	46.68	30.16	208.40
	St.D.	159.11	58.83	11.01	46.94	105.11	91.91	273.50

Table 10: Generalization performance results for the Function Approximation benchmarks

Benchmarks		Initialization Algorithms						
		BoersK	Bottou	KimRa	LIT-A	NW	SCAWI	Smieja
Function 1	*Mean*	0.0090	0.0090	0.0091	0.0089	0.1595	0.0090	0.0090
	St.D.	0.0007	0.0006	0.0006	0.0007	0.0000	0.0007	0.0007
Function 2	*Mean*	0.0130	0.0134	0.0144	0.0131	0.4686	0.0133	0.0141
	St.D.	0.0020	0.0024	0.0019	0.0025	0.0000	0.0019	0.0024
Function 3	*Mean*	0.0210	0.0221	0.0222	0.0211	0.6704	0.0218	0.0211
	St.D.	0.0043	0.0009	0.0009	0.0032	0.0000	0.0023	0.0048

the function mapping experiments, is carried out using the non-parametric Friedman test. The average rankings computed are reported in Table 11. The hypotheses of the Friedman test and the posthoc procedures performed are the same as those for the suite 1 of the experiments. The unadjusted and the adjusted p-values of the posthoc procedures are given in Table 12. The fact that the control procedure for the convergence rate is LIT-Approach together with the p-values in Table 12 underline the superiority of the proposed method. On the other hand the algorithm of Bottou is considered to be the control procedure for the generalization. This does not prove that Bottou's method performs better than LIT-Approach as the corresponding unadjusted and adjusted p-values denote that the performance of these algorithms is the same.

Table 11: Average ranking achieved by the Friedman test (Function Approximation benchmarks)

Initialization method	Convergence Rate	Generalization
BoersK	3.87	3.60
Bottou	3.59	3.24
KimRa	4.91	3.80
LIT-A	2.99	3.25
NW	4.57	6.43
SCAWI	3.45	3.48
Smieja	4.61	4.20
Statistic	195.05	486.06
p-value	0.10e-09	0.21e-09

Table 12: *p*-values of multiple comparisons (Function Approximation benchmarks)

Initialization algorithm	Unadjusted	Adjusted	
		Bonferroni-Holm	Benjamini-Hochberg
	Convergence rate (Control algorithm is LIT-A)		
BoersK	0.0000	0.0000	0.0000
Bottou	0.0007	0.0020	0.0009
KimRa	0.0000	0.0000	0.0000
NW	0.0000	0.0000	0.0000
SCAWI	0.0091	0.0182	0.0106
Smieja	0.0000	0.0000	0.0000
	Generalization (Control algorithm is Bottou)		
BoersK	0.0394	0.1586	0.0690
KimRa	0.0015	0.0073	0.0034
LIT-A	0.9397	1.0000	1.0000
NW	0.0000	0.0000	0.0000
SCAWI	0.1706	0.5119	0.2389
Smieja	0.0000	0.0000	0.0000

4.2.3. Comments and Remarks

In terms of convergence success (Table 8) all initialization methods seem to have similar performance while the method of Bottou gives the best results as the networks initialized with this method are trapped in local minima for only 9 trials (2% of the total number of trials). The results reported in Table 9 support the improvement in convergence rate offered by the proposed method, when compared with other weight initialization methods in the context of these function approximation problems. Con-

cerning generalization, results in Table 10 indicate that the performance observed for the LIT-Approach seems to be among the best of all methods. These remarks are confirmed by the results of the Friedman test in Table 11 and the posthoc procedures in 12, especially regarding the convergence rate.

On the other hand generalization performance of the LIT-Approach is found to be marginally weaker than the performance of Bottou's method. Compared with the results of suite 1 performance of the LIT-Approach in terms of generalization is considered here suboptimal. Various reasons may account for this. One reason is the fact that the distribution of the input data for benchmarks $Function1$ and $Function3$ was considered to be the normal and therefore it was "roughly" approximated using the term $1.5\sigma_{x_i}$. Actually, it seems that defining weight intervals with different ranges for the input variables seems meaningless for function approximation problems as these are different from classification ones. Moreover, speaking about ouliers and extreme input data in the case of a function approximation benchmark is useless. However, in the context of the LIT-Approach we have not considered that function approximation should be treated as a special case. So, it remains as an open issue for further investigation, in the case of function approximation, the estimation of the optimal value k for the term $1.5\sigma_{x_i}$, given a specific function approximation problem. However, these claims, as well as others, need to be further investigated with more benchmarks together with taking into account the whole input set of training patterns.

What is noteworthy here is the poor generalization performance achieved by the network when initialized with the Nguyen-Widrow method; in this case, it is very likely that the network gets trapped in local minima. One should note that, seemingly, the Nguyen-Widrow method showed notable performance in convergence rate when experimenting with the non-linear *Function 1*, which is reported in the original paper by Nguyen and Widrow (Nguyen & Widrow, 1990).

4.3. Suite 2 of Experiments

4.3.1. Experimental Setup

Setting up this suite is motivated by the importance the research community has devoted to the weight initialization method of Nguyen and Widrow. In addition, pop-

ular neural network packages, such as the Neural Network Toolbox of MATLAB and the Encog Neural Network Framework (2013), use this technique as the default initialization algorithm for neural networks.

The datasets used for these experiments and basic features of the problems are briefly outlined here. More details on these benchmarks can be found in the UCI repository of machine learning database (Frank & Asuncion, 2010) and references cited therein.

1. Iris classification benchmark (inputs:4, outputs:3). 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. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The training set used consists of 120 examples and the test set of 30 examples.

2. Pima Indians Diabetes problem (inputs:8, outputs:2). The aim of this real world-classification task is to decide when a Pima Indian individual is diabetes positive or not. The values of the input attributes represent personal data and result from a medical examination. The dataset consists of a total of 768 patterns. The training set used consists of 576 patterns and the test set of the 192 remaining patterns.

3. Thyroid classification problem (inputs:21, outputs:3). Based on patient query data and patient examination data the task is to decide whether the patient's thyroid has over function, normal function, or under function. Using the original *thyroid1* dataset the training set is made from 5400 patterns while the test set is made from 1800 patterns.

4. Yeast classification problem (inputs:8, outputs:10). Yeast is a relatively complicated organism possessing different types of proteins, related to the cytoskeletal structure of the cell, the nucleus organization, membrane transporters and metabolic related proteins (as mitochondrial proteins). After the necessary preprocessing, Yeast data is found to include 1453 patterns, that is, there are 1453 proteins labeled according to 10 sites. As the data set is radically imbalanced, the training set was generated by randomly selecting approximately the 70% of patterns from each of the 10 sites, giving a total of 1019 training patterns. The

rest of the patterns, i.e. 434, were included in the test set.

5. Gene2 classification (inputs:120, outputs:3). This is a binary problem with 120 input attributes and 3 output classes. The goal of this classification task is to decide, from a window of 60 DNA sequence elements (nucleotides), whether the middle is either an intron/exon boundary (a donor), or an exon/intron boundary (an acceptor), or none of them. The dataset for this problem was created based on the Splice-junction Gene Sequences dataset from the UCI repository, (Frank & Asuncion, 2010). It consists of 2990 patterns (duplicates are excluded) and it is partitioned to form the training set (2243 patterns) and the test set (747 patterns).

The original datasets were preprocessed to eliminate duplicate patterns and values were scaled to match requirements set by the weight selection procedures. These operations were performed according to PROBEN1 guidelines, (Prechelt, 1994). Unless otherwise stated, the training sets used are made with 75% of the patterns of the initial dataset and the test sets with the rest 25% of the patterns.

For each benchmark a neural network architecture was defined. Batch processing was used for training with five well known training algorithms, namely, the Adaptive gradient descent with momentum (AGDM) (Vogl et al., 1988; Hagan et al., 1996), Resilient back-propagation (RBP) (Riedmiller & Braun, 1993), the Levenberg-Marquardt method (LM) (Hagan et al., 1996; Marquardt, 1963; Hagan & Menhaj, 1994), Scaled conjugate gradient (SCG) (Moller, 1993) and the Broyden-Fletcher-Goldfarb-Shanno method (BFGS) (Gill et al., 1981). For each benchmark a set of a thousand (1000) initial weight vectors was created by each one of the two initialization methods and used in all experiments. A total of 50 (= 5 benchmarks $\times$ 5 training algorithms $\times$ 2 initialization methods) experiments were carried out, giving a total of 25000 weights for each initialisation method. Architectures of the networks and training parameters used are those reported in the literature (Anastasiadis, 2005) to be the most appropriate for each problem; see Table 13. Note that the value of the momentum coefficient was set to 0.5 and all networks are fully connected without intra-layer or supra-layer connections.

Table 13: Architectures of networks and training parameters used for the Suite 2 of experiments

Benchmark	Network architecture	Activation function*	Total weights	Learning rate†	Max epochs‡	Goal for MSE‡	Min gradient‡
Iris	4-2-3	logsig	19	0.90	5000	0.01	1.0e-09
Diabetes	8-2-2-2	logsig	30	0.50	2000	0.14	1.0e-09
Thyroid	21-4-3	logsig	103	0.50	2000	0.035	1.0e-09
Yeast	8-16-10	logsig	314	0.50	5000	0.05	1.0e-09
Gene2	120-4-2-3	tansig	503	0.50	5000	0.0075	1.0e-09

* logsig denotes the logistic sigmoid function and tansig is the hyperbolic tangent

† Learning rate is the initial learning rate used for training

‡ Training stops when at least one of the following conditions is satisfied, Max epochs is reached, Error of the network output becomes lower than or equal to Goal for MSE, Min gradient is reached during training.

Table 14: Convergence success results for the Suite 2 of experiments

Benchmark	Training Algorithms									
	AGDM		BFGS		LM		RBP		SCG	
	NW	LIT-A	NW	LIT-A	NW	LIT-A	NW	LIT-A	NW	LIT-A
Iris	18.2	100.0	11.5	46.3	42.8	96.0	76.1	99.9	46.9	98.5
Diabetes	12.2	77.2	21.5	73.9	40.3	63.4	39.1	77.3	40.5	70.2
Thyroid	19.0	99.1	58.1	73.9	85.2	99.9	100.0	100.0	93.6	91.3
Yeast	94.5	100.0	35.3	98.3	68.1	94.4	100.0	100.0	84.7	100.0
Gene2	20.1	86.5	1.8	6.7	32.5	43.9	6.9	35.2	26	75.7

All numbers indicated are in the range $0 \ldots 100$ to denote the percentage of successful training trials

4.3.2. Analysis of the Results

The results obtained regarding the performance measures set are shown in Tables 14, 15 and 16. A rough observation of these results shows that the proposed LIT-Approach delivers successful network performance for all parameters.

The statistical analysis, applied on results in Tables 15 and 16, concerns the comparison of the two initialization techniques using t-test for independent unpaired samples data. However, while the samples are independent by default, application of the t-test assumes that these samples are drawn from normally distributed populations with equal variances. The Shapiro-Wilk test (Shapiro & Wilk, December 1965) was used to test the normality assumption, with $\alpha = 0.05$, while SPSS automatically uses Levene's test for equality of variances, (Green & Salkind, 2003). In all experiments, except one (training Gene2 benchmark with the BFGS algorithm for generalization performance)

the hypothesis of normality is rejected (p-value < 0.05) for the results of both weight initialization methods and so the non-parametric Mann-Whitney test for independent samples is used to compare equality of medians. The statistical significance of the comparison, that is the p-value (*Sig.*) indicated underneath the results for every pair of experiments, is the one calculated by the Mann-Whitney test (or the t-test, when the normality assumption is validated), (Green & Salkind, 2003).

Results in Table 15, are indicative that in all cases, except in two of them, the pro-

Table 15: Convergence rate results for the Suite 2 of experiments

	Training Algorithms									
	AGDM		BFGS		LM		RBP		SCG	
	NW	LIT-A	NW	LIT-A	NW	LIT-A	NW	LIT-A	NW	LIT-A
	Iris									
Min	174	111	9	21	4	3	91	33	15	14
Max	4927	143	1076	3272	3532	621	4027	1381	4508	205
Median	981	117	59	51	13	7	512	67	167	26
Sig.	0.000		0.863		0.000		0.000		0.000	
	Diabetes									
Min	254	366	63	56	6	5	67	74	59	48
Max	1996	1999	1935	1835	1925	1903	1977	1996	1993	1971
Median	1021	1436	250	123	27	18	465	681	390	166
Sig.	0.000		0.000		0.000		0.000		0.000	
	Thyroid									
Min	282	211	186	176	10	10	92	84	570	329
Max	1992	528	1976	1971	1999	1659	1455	687	1991	1980
Median	402	319	788.5	575	45	25	268	103	1229	732.5
Sig.	0.000		0.000		0.000		0.000		0.000	
	Yeast									
Min	729	906	168	187	7	7	74	73	102	103
Max	4753	1594	4636	4348	4560	4432	280	197	4701	800
Median	1553	1148.5	403	423	20	12	130	112.5	196	145
Sig.	0.000		0.534		0.000		0.000		0.000	
	Gene2									
Min	489	283	110	57	12	11	436	222	129	61
Max	4971	4956	4757	4990	4976	4675	4946	4654	4974	4962
Median	2426	720	1936	794	31	24	2281	697.5	1118	155
Sig.	0.000		0.016		0.000		0.000		0.000	

posed method produced initial weight vectors which permitted faster convergence of the training algorithm in use. In the other two cases, the Iris and the Yeast benchmarks, the two weight initialization methods seem to allow for the same performance of the BFGS training algorithm.

With regards to generalization, results are presented in Table 16 and denote the percentage of successfully classified unknown patterns. In the case of Gene2 benchmark for networks trained with the BFGS algorithm, mean values (marked with *) are used instead of medians, as the Shapiro-Wilk test approved the normality of populations which allowed us to use the t-test.

Results, in Table 16, show that the proposed initialization method led to networks that generalize better in about 50% of the cases (12 out of the 25 comparisons) while the Nguyen-Widrow initialization technique was superior in only 3 cases. One may notice inconsistency between the p-values indicating significant difference between the medians while these medians appear, for the Iris problem and some training algorithms, to be the same. Indeed, the values of the medians suggest that the two methods display the same performance in terms of generalization. However, for the NW algorithm the score of 95.56% is taken over 18.2% of successful trials while for the LIT-Approach 95.56% corresponds to 100% of successful trials.

4.3.3. Comments and Remarks

The proposed method demonstrates better performance than the one proposed by Nguyen and Widrow in all cases. However, despite the fact that these benchmarks concern classification problems while the method of Nguyen and Widrow was originally demonstrated for a function approximation problem, we believe that, what really affects performance of this method is the neural networks architecture itself and not the type of the problem at hand. The magnification factor multiplying the randomly selected input-to-hidden layer weights for a (2-21-1) network is given by the formula $0.7H^{1/N} = 21^{1/2} \simeq 4.5$ where H denotes the number of hidden layer nodes. Weights of this magnitude seem to define in the weight space a starting point which accelerates convergence of the training algorithm.

On the other hand, it is easy to notice that when the network has a differently

Table 16: Generalization performance results (%) for the Suite 2 of experiments

	Training Algorithms									
	AGDM		BFGS		LM		RBP		SCG	
	NW	LIT-A	NW	LIT-A	NW	LIT-A	NW	LIT-A	NW	LIT-A
					Iris					
Min	95.56	95.56	91.11	91.11	91.11	91.11	93.33	93.33	91.11	93.33
Max	97.78	97.78	97.78	97.78	97.78	97.78	97.78	97.78	97.78	97.78
Median	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	97.78
Sig.	0.000		0.000		0.000		0.351		0.000	
					Diabetes					
Min	70.83	69.27	68.23	65.63	68.75	67.71	68.23	68.23	68.75	68.23
Max	78.65	78.65	80.73	79.17	79.69	79.17	79.17	79.17	79.69	80.21
Median	75.52	76.04	75.52	75.52	75.52	75.00	75.00	75.52	75.52	75.52
Sig.	0.000		0.029		0.011		0.000		0.234	
					Thyroid					
Min	92.28	93.78	92.22	92.50	92.10	92.18	92.25	93.05	97.11	96.67
Max	94.22	94.17	98.28	98.50	99.00	99.00	98.30	98.75	98.33	98.44
Median	93.67	94.06	97.44	97.78	98.28	98.33	97.07	97.90	97.83	97.86
Sig.	0.000		0.000		0.000		0.000		0.788	
					Yeast					
Min	55.07	58.53	44.47	43.55	38.82	45.39	58.29	58.53	43.55	51.15
Max	64.06	64.06	62.44	61.29	62.90	63.82	64.52	64.29	63.59	63.36
Median	60.83	61.52	55.99	52.77	57.14	60.60	61.52	61.52	59.45	60.37
Sig.	0.000		0.000		0.000		0.081		0.000	
					Gene2					
Min	78.18	79.38	82.33	82.06	79.38	78.31	79.79	79.79	79.12	76.04
Max	90.76	91.43	89.83	89.29	90.36	90.50	89.70	90.76	90.50	90.63
Median	85.68	87.15	85.54†	85.82†	86.61	85.54	86.75	86.08	85.81	86.75
Sig.	0.000		0.541†		0.000		0.098		0.000	

(%) all numbers in this table denote percentage.

† denotes that with the BFGS algorithm, mean values are used instead of medians, as the Shapiro-Wilk test approved the normality of populations which allowed us to use the *t*-test.

shaped architecture, that is, the number of input nodes is higher than the number of hidden nodes, the factor $H^{1/N}$ tends to reduce to 1. In consequence the initialization method tends to degenerate to the commonly used random weight selection in the in-

terval $[-1, 1]$. Hence, performance of the Nguyen-Widrow method strongly depends on the number of nodes of the input and the hidden layers. This argument may be experimentally confirmed by progressively training a network while gradually increasing the number of hidden layer nodes. Some training trials we performed on the Iris problem showed that convergence rate of the Nguyen-Widrow method increases when increasing the hidden layer units. Nevertheless, the price to pay for this is the decrease in generalization performance.

4.4. Suite 3 of Experiments

4.4.1. Experimental Setup

The objective of this suite is twofold. Firstly, it was set up in order to test the ability of the LIT-Approach to deal with problems having a big number of features and thus see if the assumptions underlying the method remain valid when the method addresses large real life problems. The second objective is to test the performance of the LIT-Approach against more recent competitors that do not belong to the "family" of methods which randomly select initial weights from some predefined interval. Among such methods we retained the following:

Linear-Least-Squares initialization of MLPs through backpropagation of the desired response (LLSQ). The method uses a technique for backpropagating the desired response of an MLP through its nonlinear layers. This permits to train each layer by solving a linear system of equations per layer which is the solution to a linear least squares problem. The authors claim that besides initialization the method can be used for training the network (Erdogmus et al., 2005).

Computing Linear-Least-Squares layer by layer in forward direction (FLLS). The outputs of the first hidden layer are assigned with random numbers in the active region of the activation function of the nodes. The inverses of these output values are computed and a linear least squares problem is solved to define the weights of the input to the hidden layer. Using these weights and the input patterns, actual outputs are computed and used to repeat the process towards the next layer until the output layer. The method

determines the initial weights by successively solving one linear least squares problem per layer in a feed forward way (Yam & Chow, 1997).

Particle Swarm Optimization based weight initialization (PSOI). The method was initially proposed in van den Bergh (1999). Particle Swarm Optimization is used to define the most pertinent initial weights which are then used for subsequent training by BP. Actually, PSO performs pretraining of the MLP for some iterations before activating BP. The method is an evolutionary approach to weight initialization which, however, suffers itself from the initialization problem. In our experiments PSO is activated for 20 iterations and the weights computed are used, in the sequel, by the online BP.

Hereafter we will refer to these methods using their acronyms. In addition to the above in this suite we used the method of Bottou. Hence we form a complete test of comparisons between five methods; the previous three and the two methods that had the best performance in the other test suites. It is important to note that in this suite the algorithm implementing LIT-Approach uses the third quartile of the input data Q_3 instead of some multiple of the standard deviation. Moreover, the weights of the hidden-to-output nodes are computed using assumptions introduced in subsection 3.3. The benchmarks used for this test suite are defined hereafter in alphabetical order.

1. Far-infrared Laser (FIL) (inputs:50, outputs:1). This is an extension of the Data Set A from the Santa Fe Competition Data (Weigend & Gershenfeld, 2001) consisting of sampled values from the emission intensity of far-infrared laser (NH_3-FIR) (Hüebner et al., 1989). It is a time series forecasting problem and the aim is to predict the intensity of a far infared laser at a particular moment from past samples. In our tests we choose to predict the value of the quantity x at time $k+1$ given the past 50 samples $x_k, x_{k-1}, \cdots, x_{k-49}$ as in Yam & Chow (2001). The total number of patterns is 10043. The training set was made with 8000 patterns and the rest 2043 were used for the test set.

2. Landsat Satellite Data (LSAT) (inputs:36, outputs:6). This dataset was generated taking a small section (82 rows and 100 columns) from the original Landsat data.

The dataset consists of the multi-spectral values of pixels in 3x3 neighborhoods in a satellite image. The aim is to predict the classification associated with the central pixel in each neighborhood. Each line in the data contains 36 values, that is, the pixel values in the four spectral bands times the 9 pixels in the 3x3 neighbourhood. The classification label of the central pixel is a number corresponding to one of the seven classes. Note that class 6 has no examples in this dataset. The total number of 6435 patterns available was partitioned in the training set composed of 4435 patterns and the test set having 2000 patterns.

3. Multiple Features Data Set (MFEAT) (inputs:649, outputs:10). This dataset was created by Robert P.W. Duin, Dept. of Applied Physics, Univ. of Delft. It consists of features of handwritten digits ('0'–'9') extracted from a collection of Dutch utility maps. A number of 200 patterns per class, that is a total of 2000 patterns have been digitized in binary images. These digits are represented in terms of the following six feature sets given in separate files:

- mfeat-fou: 76 Fourier coefficients of the character shapes
- mfeat-fac: 216 profile correlations
- mfeat-kar: 64 Karhunen-Love coefficients
- mfeat-pix: 240 pixel averages in 2 x 3 windows
- mfeat-zer: 47 Zernike moments
- mfeat-mor: 6 morphological features

The 2000 patterns, contained in each file, are stored in ASCII on 2000 lines. The first 200 patterns correspond to class '0', the next 200 to class '1', that is, sets of 200 patterns for each of the classes '0'–'9'. The training set for our experiments consists of 1500 patterns and the test set has 500 patterns.

4. The MNIST database of handwritten digits (inputs:784, outputs:10). The MNIST database was constructed from NIST's Special Database 1 (SD-1) and Special Database 3 (SD-3), which contain binary images of handwritten digits (LeCun et al., 2004). The MNIST training set has a total of 60000 patterns, that is, 30000 patterns from SD-1 and 30000 patterns from SD-3. The test set is composed of

5000 patterns from SD-1 and 5000 patterns from SD-3 that is a total of 10000 patterns. The sets of writers of the training set and test set were disjoint. For performance reasons of our experiments we formed a training set consisting of 10% of the patterns of the original training set and a test set with 10% of the patterns of the original test set. For every class in the database we selected the first 10% of the patterns belonging to this class thus forming a balanced sample of the original dataset. So the training set for our experiments consists of 6000 patterns and the test set has 1000 patterns.

A total of 18 (16+2) training experiments were executed for the above 4 benchmarks and the 4 (LIT-A, Bottou, FLLS, PSOI) weight initialization methods considered in this subsection. The other 2 experiments concern the LLSQ method which was tested only against the FIL and LSAT benchmarks. Each training experiment is made up of a hundred (100) initial weight vectors derived using one of the weight initialization methods. The same network architecture was initialized with these vectors and trained using online BP. The network architecture and the training parameters, used in this arrangement, are reported in Table 17. Benchmarks are listed in increasing order of the number of features using their acronyms.

Table 17: Architectures of networks and training parameters used for the Suite 3 of the experiments

Benchmark	Network architecture	Activation function†	Learning rate	Convergence criterion	Max cycles	Input data scale
LSAT	36-36-6	tansig‡	0.9	90%	250	[-1,1]
FIL	50-20-1	logsig	0.9	0.001	100	[0,1]
MFEAT	649-649-10	logsig	0.15	97.5%	500	[-1,1]
MNIST	784-300-10	logsig	0.15	95%	500	[-1,1]

† logsig denotes the logistic sigmoid function and tansig is the hyperbolic tangent

‡ Landsat benchmark uses the hyperbolic tangent for the hidden layer nodes and the logistic sigmoid for nodes in the output layer. All other networks use the same activation function for all nodes.

4.4.2. Analysis of the Results

Tables 18, 19 and 20 report the experimental results on the benchmarks for the performance measures considered. The symbol – is used in these Tables to denote that the

corresponding initialization method failed to meet the convergence criteria exceeding the maximum number of cycles in all trials. A quick look at these results shows that the three newly introduced initialization methods have very poor performane especially in the case of the benchmarks with a big number of features. We need to note that regarding the method LLSQ these tables report the results only for the first two benchmarks that is LSAT and FIL.

The comparison between the performance of the initialization methods was carried out using ANOVA for the LSAT and FIL benchmarks. The difference between the proposed LIT-Aproach and the other methods is indicated with a * and supported by the corresponding p-value (*Sig.*). In addition comparison of the initialization methods in these benchmarks is carried out using the non-parametric test of Friedman and the posthoc procedures of Bonferroni-Holm and Benjamini-Hochberg in the same context as for the suite 1 of the experiments. The average rankings computed are reported in Table 21. These rankings roughly confirm the results observed regarding the mean values of the performance parameters. Pairwise comparison results reported in Table 22 reward the performance of LIT-Approach in terms of convergence rate while they reveal that the proposed method has the same performace with Bottou's method regarding generalization.

Table 18: Convergence success results in 100 trials for the Suite 3 of the experiments

Benchmark	Initialization Algorithms				
	Bottou	FLLS	LIT-A	LLSQ	PSOI
LSAT	100	0	100	72	68
FIL	100	0	100	0	100
MFEAT	100	0	100	**	0
MNIST	100	0	100	**	0

4.4.3. Comments and Remarks

The results of these experiments coincide with those already obtained in the previous suites. As seen above the LIT-Approach is dominant in terms of convergence speed and seems to be equal, or at most slightly weaker, in terms of generalization compared with Bottou's method. The method of Bottou is generally powerful while it seems to

Table 19: Convergence rate results for the Suite 3 of the experiments

Benchmarks		Initialization Algorithms				
		LIT-A	Bottou	FLLS	LLSQ	PSOI
LSAT	*Mean*	110	120.46	251*	143.32*	150.95*
	St.D.	40.96	31.36	0.00	78.08	75.92
	Sig.		0.363	0.000	0.000	0.000
FIL	*Mean*	7.51	7.15	0.00*	0.00*	9.6*
	St.D.	0.67	0.69	0.00	0.00	3.25
	Sig.		0.589	0.000	0.000	0.000
MFEAT	*Mean*	5.39	12.97*	–	**	–
	St.D.	4.36	1.02	–	**	–
	Sig.		–	–	**	–
MNIST	*Mean*	7.66	11.66*	–	**	–
	St.D.	0.61	0.48	–	**	–
	Sig.		–	–	**	–

* denotes that the mean value of the initialization method is significantly different from the mean value of LIT-A using the indicated p-values (*Sig.*) computed by the posthoc analysis of the ANOVA results.

– denotes that the initialization method failed to meet the convergence criteria exceeding the maximum number of cycles in all trials.

** denotes that the initialization method was not tested for this benchmark.

be weaker when addressing problems with big number of features such as MFEAT and MNIST. These remarks are also supported by the non-parametric statistical analysis tests, Tables 21 and 22, for both performance characteristics. What is disarming is the seemingly bad performance of the other methods. We need to note here that in these experiments we do take into account the resources needed in terms of time and memory for an initialization procedure to run. The reason is that the weight initialization methods based on random weight selection need very little memory to run and they preprocess the input patterns only in order to extract simple statistics of the sample. On the other hand the memory and time requirements set by methods not based on random weight selection constitute a serious barrier for their application in real life problems. Finally, besides the limitations analyzed in subsection 3.2, LIT-Approach seems to perform very well even in the case of the selected real life problems with big

Table 20: Generalization performance results for the Suite 3 of the experiments (FIL in mean absolute error)

Benchmarks		Initialization Algorithms				
		LIT-A	Bottou	FLLS	LLSQ	PSOI
LSAT	*Mean*	88.27%	88.33%	45.19%*	85.42%*	83.56%*
	St.D.	0.42%	0.44%	13.73%	6.39%	9.05%
	Sig.		0.972	0.000	0.000	0.000
FIL	*Mean*	0.0155	0.0158	0.2765*	0.0668*	0.0176*
	St.D.	0.0007	00010	0.1451	0.0282	0.0017
	Sig.		0.157	0.000	0.000	0.000
MFEAT	*Mean*	98.52%	98.69%	–	**	–
	St.D.	0.23%	0.24%	–	**	–
	Sig.		–	–	**	–
MNIST	*Mean*	89.69%	89.57%	–	**	–
	St.D.	0.41%	0.26%	–	**	–
	Sig.		–	–	**	–

* denotes that the mean value of the initialization method is significantly different from the mean value of LIT-A using the indicated p-values (*Sig.*) computed by the posthoc analysis of the ANOVA results.

– denotes that the initialization method failed to meet the convergence criteria exceeding the maximum number of cycles in all trials.

** denotes that the initialization method was not tested for this benchmark.

Table 21: Average ranking achieved by the Friedman test (Suite 3 of the experiments)

Initialization method	Convergence Rate	Generalization
Bottou	2.01	1.63
LIT-A	1.54	1.64
FLLS	4.30	4.50
LLSQ	3.80	3.87
PSOI	3.35	3.36
Statistic	890.44	1094.19
p-value	0.25e-09	0.0

number of patterns. It is worth noting that we have not considered benchmarks with even higher number of features as these are normally treated with feature selection and/or dimensionality reduction methods before applying a classification method that

Table 22: p-values of multiple comparisons (Suite 3 of the experiments)

Initialization algorithm	Unadjusted	Adjusted	
		Bonferroni-Holm	Benjamini-Hochberg
	Convergence rate (Control algorithm is LIT-A)		
Bottou	0.0000	0.0001	0.0000
FLLS	0.0000	0.0000	0.0000
LLSQ	0.0000	0.0000	0.0000
PSOI	0.0000	0.0000	0.0000
	Generalization (Control algorithm is Bottou)		
LIT-A	0.9554	1.0000	1.0000
FLLS	0.0000	0.0000	0.0000
LLSQ	0.0000	0.0000	0.0000
PSOI	0.0000	0.0000	0.0000

requires parameter initialization.

5. Conclusion

In this paper we studied an interval analysis approach for neural network weight initialization with the aim to deal with uncertainty about the initial weights. Instead of algebraically solving a linear interval system we formulated and solved a linear interval tolerance problem. Hence, a self contained standalone algorithm is proposed that inherently includes major concepts such as: the number of inputs to a node in the first hidden layer, the statistical information of the input data, effective positioning of the hyperplanes in the pattern space and full utilization of the dynamic range of the activation function. Both the theoretical analysis and the experimental results suggest that the proposed LIT-Approach successfully tackles the problem of neural saturation while avoiding false local minima.

The proposed LIT-Approach has been compared against other well known random weight initialization techniques on a number of well known real world benchmarks. The experiments carried out cover a broad range of problems using networks with architectures of increasing complexity. The results obtained are suggestive of the efficiency of the proposed method while providing an overall classification framework for some of the most well known weight initialization methods. For all performance

characteristics set, the proposed method is either on top of other initialization methods or at least exhibits similar performance with the other methods. Moreover, it is easy to notice that the proposed method demonstrates stable inter-problem performance behaviour. We believe that all these features make the proposed method a reliable algorithm for use in real life problems.

Solving the linear interval tolerance problem seems to successfully address the problem of weight initialization. However, some important questions need to be investigated in future research. These questions concern the possibility to define among all solutions in the Tolerance solution set, the optimal one, if any, for the weight initialization problem; the evaluation of different algorithms solving the linear interval tolerance problem for weight initialization, and others, such as the potential offered by such an approach to other types of neural networks.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable suggestions and comments on earlier draft of the manuscript, that helped to significantly improve the paper at hand. Special thanks are due to Prof. Erdogmus for kindly offering the software implementing the method LLSQ, as well as to Mr. George Dimakopoulos, statistical analyst at the Technological Educational Institute of Epirus, for his helpful comments regarding ANOVA.

References

Agresti, A., & Franklin, C. (2009). *Statistics: The Art and Science of Learning from Data*. (3rd ed.). Boston, MA: Pearson Education.

Ahn, J., Marron, J., Muller, K. M., & Chi, Y.-Y. (2007). The high-dimension, low-sample-size geometric representation holds under mild conditions. *Biometrika*, *94*, 760–766.

Alefeld, G., & Herzberger, J. (1983). *Introduction to Interval Computations*. New York, NY: Academic Press.

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

Anastasiadis, A. D. (2005). *Neural Networks Training and Applications using Biological Data*. Ph.D. thesis Birkbeck College, University of London.

Beaumont, O., & Philippe, B. (2001). Linear interval tolerance problem and linear programming techniques. *Reliable Computing*, *7*, 433–447.

Bello, R., & Verdegay, J. L. (2012). Rough sets in the soft computing environment. *Information Sciences*, *212*, 1 – 14.

van den Bergh, F. (1999). Particle swarm weight initialization in multi-layer perceptron artificial neural networks. In *Proceedings of the ICAI, Development and Practice of Artificial Intelligence Techniques* (pp. 41–45). Durban, South Africa.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc.

Boers, E. G. W., & Kuiper, H. (1992). *Biological Metaphors and the Design of Modular Artificial Neural Networks*. Master's thesis Leiden University Netherlands.

Bottou, L. Y. (1988). Reconnaissance de la parole par reseaux multi-couches. In *Proceedings of the International Workshop Neural Networks Applications, Neuro-Nimes'88* (pp. 197–217). Nimes, France.

Chen, C. L., & Nutter, R. S. (1991). Improving the training speed of three-layer feed-forward neural nets by optimal estimation of the initial weights. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN'91* (pp. 2063–2068). Seattle, WA volume 3.

Degrauwe, D., Lombaert, G., & Roeck, G. D. (2010). Improving interval analysis in finite element calculations by means of affine arithmetic. *Computers and Structures*, *88*, 247–254.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*, 3–18.

Drago, G. P., & Ridella, S. (1992). Statistically controlled activation weight initialization (SCAWI). *IEEE Transactions on Neural Networks*, *3*, 627–631.

Erdogmus, D., Fontenla-Romero, O., Principe, J., Alonso-Betanzos, A., & Castillo, E. (2005). Linear-least-squares initialization of multilayer perceptrons through back-propagation of the desired response. *IEEE Transactions on Neural Networks*, *16*, 325–337.

Fahlman, S. E. (1988). *An empirical study of learning speed in back-propagation networks*. Technical Report CMU–CS–88–162 School of Computer Science, Carnegie Mellon University Pittsburg.

Fernández-Redondo, M., & Hernández-Espinosa, C. (2001). Weight initialization methods for multilayer feedforward. In *Proceedings of the European Symposium on Artificial Neural Networks, ESANN'2001* (pp. 119–124). Bruges, Belgium: D-Facto.

Encog Neural Network Framework (2013). URL: http://www.heatonresearch.com/encog/articles/nguyen-widrow-neural-network-weight.html.

Frank, A., & Asuncion, A. (2010). UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, *180*, 2044–2064.

Garloff, J., Idriss, I., & Smith, A. (2007). Guaranteed parameter set estimation for exponential sums: The three-terms case. *Reliable Computing*, *13*, 351–359.

Gill, P. E., Murray, W., & Wright, M. H. (1981). *Practical Optimization*. New York, NY: Academic Press.

Goldsztejn, A. (2007). Comparison of the Hansen–Sengupta and the Frommer–Lang–Schnurr. *Computing*, *79*, 53–60.

Green, S. B., & Salkind, N. J. (2003). *Using SPSS for Windows and Macintosh: Analyzing and Understanding Data*. (3*rd* ed.). Upper Saddle River, NJ: Prentice Hall.

Hagan, M. T., Demuth, H. B., & Beale, M. H. (1996). *Neural Network Design*. Boston, MA: PWS Publishing.

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, *5*, 989–993.

Hansen, E. (2006). Solving over-determined systems of interval linear equations. *Reliable Computing*, *12*, 239–243.

Hansen, E., & Walster, G. W. (2004). *Global Optimization Using Interval Analysis*. (2nd ed.). New York, NY: Marcel Dekker.

Hansen, E. R. (1992). Bounding the solution of interval linear equations. *SIAM Journal on Numerical Analysis*, *29*, 1493–1503.

Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press.

Haykin, S. (1999). *Neural Networks A Comprehensive Foundation*. (2nd ed.). Upper Saddle River, NJ: Prentice-Hall.

Heindl, G., Kreinovich, V., & Lakeyev, A. (1998). Solving linear interval systems is NP-hard even if we exclude overflow and underflow. *Reliable Computing*, *4*, 383–388.

Hu, C., & He, L. (2007). An application of interval methods to stock market forecasting. *Reliable Computing*, *13*, 423–434.

Hüebner, U., Abraham, N. B., & Weiss, C. O. (1989). Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared NH3 laser. *Phys. Rev. A*, *40*, 6354.

Ishibuchi, H., & Nii, M. (1998). Improving the generalization ability of neural networks by interval arithmetic. In L. C. Jain, & R. K. Jain (Eds.), *Proc. 2nd Int. Conf. Knowledge-Based Intelligent Electronic Systems, KES 1998*. IEEE.

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

Kearfott, R. B. (1996). Interval computations: introduction, uses, and resources. *Euromath Bulletin*, *2*, 95–112.

Kim, Y. K., & Ra, J. B. (1991). Weight value initialization for improving training speed in the back-propagation network. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN'91* (pp. 2396–2401). Seattle, WA volume 3.

Krawczyk, R. (1969). Newton-algorithmen zur bestimmung von nullstellen mit fehlerschranken. *Computing*, *4*, 187–201.

Kreinovich, V., Lakeyev, A., Rohn, J., & Kahl, P. (1997). *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Dordrecht, Netherland: Kluwer Academic.

Kubica, B. J. (2010). Interval methods for solving underdetermined nonlinear systems. *Reliable Computing*, *15*, 207–217.

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

LeCun, Y., Cortes, C., & Burges, C. J. (2004). The MNIST database of of handwritten digits. URL: http://yann.lecun.com/exdb/mnist/.

Lee, Y., Oh, S. H., & Kim, M. W. (1991). The effect of initial weights on premature saturation in back-propagation learning. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN'91* (pp. 765–770). Seattle, WA volume I.

Li, G., Alnuweiri, H., & Wu, Y. (1993). Acceleration of back-propagation through initial weight pre–training with delta rule. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN'93* (pp. 580–5858). San Francisco, CA volume 1.

Li, H., Li, H., & Du, Y. (2007). A global optimization algorithm based on novel interval analysis for training neural networks. In *Proc. 2nd Int. Conf. Advances in Computation and Intelligence* ISICA'07 (pp. 286–295). Berlin, Heidelberg: Springer-Verlag.

Luengo, J., García, S., & Herrera, F. (2009). A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests. *Expert Systems with Applications*, *36*, 7798–7808.

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

Marquardt, D. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, *11*, 431–441.

Meyers, L. S., Gamst, G. C., & Guarino, A. J. (2013). *Performing Data Analysis Using IBM SPSS*. (1st ed.). Hoboken, NJ: John Wiley.

Moller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, *6*, 525–533.

Moore, R. E. (1966). *Interval Analysis*. Englewood Cliffs, NJ: Prentice–Hall.

Neumaier, A. (1984). New techniques for the analysis of linear interval equations. *Linear Algebra and its Applications*, *58*, 273–325.

Neumaier, A. (1986). Tolerance analysis with interval arithmetic. Freiburger Intervall-Berichte 86(9). Albert-Ludwigs-Universität, Freiburg.

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

Nguyen, D., & Widrow, B. (1990). 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* (pp. 21–26). Ann Arbor, MI volume 3.

Ning, S., & Kearfott, R. B. (1997). A comparison of some methods for solving linear interval equations. *SIAM Journal on Numerical Analysis*, *34*, 1289–1305.

Osowski, S. (1993). New approach to selection of initial values of weights in neural function approximation. *Electronics Letters*, *29*, 313–315.

Palubinskas, G. (1994). Data-driven weight initialization of back-propagation for pattern recognition. In *Proceedings of the International Conference on Artificial Neural Networks, ICANN'94* (pp. 851–854). London volume 2.

Parejo, J. A., García, J., Ruiz-Cortés, A., & Riquelme, J. C. (2012). Statservice: Herramienta de análisis estadístico como soporte para la investigación con metaheurísticas. In *Actas del VIII Congreso Expañol sobre Metaheurísticas, Algoritmos Evolutivos y Bio-inspirados*.

Pavelka, A., & Procházka, A. (2004). Algorithms for initialization of neural network weights. In *Sborník príspevku 12 rocníku konference MATLAB 2004* (pp. 453–459). Prague volume 2.

Pawlak, Z. (1991). *Rough Sets - Theoretical Aspects of Reasoning About Data*. Dordrecht, Netherlands: Kluwer Academic Publishers.

Penmetsa, R. C., & Grandhi, R. V. (2002). Efficient estimation of structural reliability for problems with uncertain intervals. *Computers and Structures*, *80*, 1103–1112.

Pivkina, I., & Kreinovich, V. (2006). *Finding Least Expensive Tolerance Solutions and Least Expensive Tolerance Revisions: Algorithms and Computational Complexity*.

Technical Report UTEP–CS–06–37 Department of Computer Science, University of Texas at El Paso El Paso.

Prechelt, L. (1994). *PROBEN1, A set of benchmarks and benchmarking rules for neural network training algorithms*. Technical Report 21/94 Fakultät für Informatik, Universität Karlsruhe Germany.

PROBEN1 (1994). Anonymous ftp site. URL: ftp://ftp.ira.uka.de/pub/neuron/.

R-Documentation (2013). Adjust P-values for Multiple Comparisons. URL: http://stat.ethz.ch/R-manual/R-patched/library/stats/html/p.adjust.html Package stats version 2.15.2.

Ramachandran, K. M., & Tsokos, C. P. (2009). *Mathematical Statistics with Applications*. London, UK: Elsvier.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the International Conference on Neural Networks, ICNN 1993* (pp. 586–591). San Francisco, CA.

Rohn, J. (1993). Cheap and Tight Bounds: The recent result by E. Hansen can be made more efficient. *Interval Computations*, *4*, 13–21.

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

Rump, S. M. (2001). Self-validating methods. *Linear Algebra and its Applications*, *324*, 3–13.

Schmidhuber, J., & Hochreiter, S. (1996). *Guessing can outperform many long time lag algorithms*. Technical Note IDSIA–19–96 Dalle Molle Institute for Artificial Intelligence Manno–Lugano, Switzerland.

Shafer, G. A. (1976). *Mathematical Theory of Evidence*. Princeton, NJ: Princeton University Press.

Shapiro, S. S., & Wilk, M. B. (December 1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*, 591–611.

Shary, S. P. (1995). Solving the linear interval tolerance problem. *Mathematics and Computers in Simulation*, *839*, 53–85.

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

Shimodaira, H. (1994). A weight value initialization method for improved learning performance of the back-propagation algorithm in neural networks. In *Proceedings of the 6th International Conference on Tools with Artificial Intelligence, ICTAI'94* (pp. 672–675). New Orleans.

Smieja, F. J. (1991). *Hyperplane spin dynamics, network plasticity and back-propagation learning*. GMD Report National Resesearch Centre for Information Science (GMD) Bonn (St. Augustine), Germany.

Sonoda, S., & Murata, N. (2013). Nonparametric Weight Initialization of Neural Networks via Integral Representation. URL: http://arxiv.org/abs/1312.6461.

Thimm, G., & Fiesler, E. (1994). *High-Order and Multilayer Perceptron Initialization*. Technical Report 94–07 IDIAP Research Institute Martigny, Switzerland.

Thimm, G., & Fiesler, E. (1997). High-order and multilayer perceptron initialization. *IEEE Transactions on Neural Networks*, *8*, 349–359.

Vogl, T. P., Mangis, J. K., Rigler, J. K., Zink, W. T., & Alkon, D. L. (1988). Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, *59*, 257–263.

de Weerdt, E., Chu, Q. P., & Mulder, J. A. (2009). Neural network output optimization using interval analysis. *IEEE Transactions on Neural Networks*, *20*, 638–653.

Weigend, A. S., & Gershenfeld, N. (2001). The Santa Fe Time Series Competition Data. http://www-psych.stanford.edu/~andreas/Time-Series/.

Wessels, L. F. A., & Barnard, E. (1992). Avoiding false local minima by proper initialization of connections. *IEEE Transactions on Neural Networks*, *5*, 899–905.

Xu, S., Lam, J., & Ho, D. W. (2005). Novel global robust stability criteria for interval neural networks with multiple time-varying delays. *Physics Letters A*, *342*, 322–330.

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

Yam, Y. F., & Chow, T. W. S. (1995). Determining initial weights of feedforward neural networks based on least squares method. *Neural Processing Letters*, *2*, 13–17.

Yam, Y. F., & Chow, T. W. S. (1997). A new method in determining the initial weights of feedforward neural networks for training enhancement. *Neurocomputing*, *16*, 23–32.

Yam, Y. F., & Chow, T. W. S. (2000). A weight initialization method for improving training speed in feedforward neural networks. *Neurocomputing*, *30*, 219–232.

Yata, K., & Aoshima, M. (2010). Intrinsic Dimensionality Estimation of High-Dimension, Low Sample Size Data with D-Asymptotics. *Communications in Statistics - Theory and Methods*, *39*, 1511–1521.

Yoon, H.-S., Bae, C.-S., & Min, B.-W. (1995). Neural networks using modified initial connection strengths by the importance of feature elements. In *Proceedings of the 1995 IEEE International Conference Systems, Man and Cybernetics* (pp. 458–461). Vancouver, BC volume 1.

Zadeh, L. A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, *1*, 3–28.

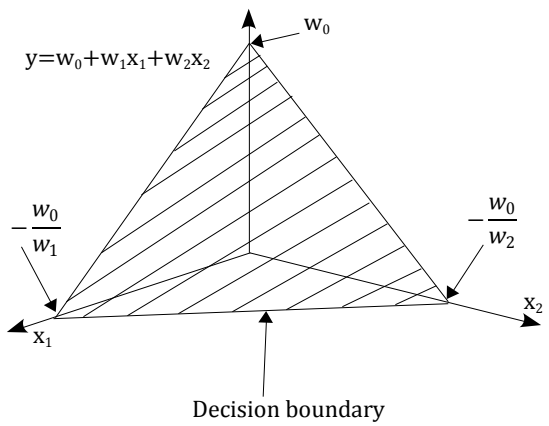

Figure 1: Hyperplane position in the augmented pattern space. The intercepts with the axes and the decision boundary are shown too.