Worst Case Analysis of Weight Inaccuracy Effects in Multilayer Perceptrons

Davide Anguita, *Member, IEEE,* Sandro Ridella, *Member, IEEE,* and Stefano Rovetta

***Abstract*— We derive here a new method for the analysis of weight quantization effects in multilayer perceptrons based on the application of interval arithmetic. Differently from previous results, we find worst case bounds on the errors due to weight quantization, that are valid for every distribution of the input or weight values. Given a trained network, our method allows to easily compute the minimum number of bits needed to encode its weights.**

***Index Terms*— Interval arithmetic, multilayer perceptron, quantization, robustness.**

I. Introduction

ELECTRONIC implementation of multilayer perceptrons is mainly focused on the issue of weight inaccuracies. In digital implementations the inaccuracies are due to the limited size of the registers used to store the values of weights. In analog implementations, noise and other artifacts (biases, component tolerance, etc.) have a similar effect, decreasing the accuracy of weights.

These inaccuracies result in undesirable effects in the behavior of neural devices both in training and in feedforward phases. For these reasons, several methods have been proposed in the literature that deal with this problem. One possible solution aims at finding modified learning algorithms [1] that reduce the accuracy requirements of the network; a second, but equally important idea is to analyze the effect of weight errors in order to predict the final performance of the network. This has always been done using statistical or heuristic techniques [2]–[5] that can give an answer only in an average sense; furthermore, several limiting assumptions must be done (e.g., weight probability distributions, linearity conditions, etc.) in order to carry on the derivation of the formulas in closed form.

In this paper we propose a new method, based on interval arithmetic [6], to perform a worst case analysis of the effects produced by weight inaccuracies. The use of interval arithmetic in artificial neural networks was originally proposed in [7] as an extension of the multilayer perceptron. Some applications include, for example, methods for fuzzy regression [8] and new training algorithms for obtaining robust networks [9]. Building on these results, we show how to obtain inaccuracy bounds that are valid regardless any input or weight distribution, as far as they are limited quantities. Due to space constraints, we cannot review in detail the application of interval arithmetic to neural networks and refer the reader to the work described in [8] and [9].

Manuscript received February 26, 1998; revised July 7, 1998 and November 13, 1998.

D. Anguita, S. Ridella, and S. Rovetta are with the Department of Biophysical and Electronic Engineering, University of Genova, Genova, Italy.

Publisher Item Identifier S 1045-9227(99)01912-8.

In Section II we briefly review the interval arithmetic multilayer perceptron (IAMLP), as proposed by Ishibuchi [8]. The main results of this work are described in Section III and experimental results are presented in Section IV.

II. Interval Arithmetic Multilayer Perceptron

An interval X is a compact representation for all the values of $\Re$ between two extreme points

$$X = [x^L, x^U] \tag{1}$$

where $x^L \leq x^U$ (obviously, if $x^L = x^U$ then the interval reduces to a real value). In this work, we will indicate intervals with capital letters ($X, Y, \cdots$) and the corresponding lower and upper bounds with lower case letters with appropriate superscripts ($[x^L, x^U], [y^L, y^U], \cdots$). We use the notation proposed in [8] throughout this paper. An example are the relation operators, defined as follows: an interval X is greater than an interval Y ($X > Y$) iff $x^L > y^U$; an interval is said to contain another interval ($X \supseteq Y$) if $x^L \leq y^L$ and $x^U \geq y^U$.

It easy to show that interval arithmetic is not a straightforward extension of arithmetic on reals; there are many subtle differences that arise dealing with intervals that are far from obvious. Deeper insight on interval arithmetic and examples of its application can be found in [6] and [10].

A multilayer perceptron (MLP) with interval weights is called interval arithmetic multilayer perceptron (IAMLP) [8], [9]. The equations describing it are the following, assuming, for simplicity, one hidden layer and one output neuron

$$H_i = f(Y_i^0) = f\left(\sum_j W_{ij}^0 x_j + W_{i0}^0\right) \tag{2}$$

$$O = f(Y^1) = f\left(\sum_i W_i^1 H_i + W_0^1\right). \tag{3}$$

The input of the IAMLP consists of a vector of real values x_j. The weights and biases of the network W_{ij}^0, W_i^1 are intervals and, consequently, so are the quantities Y_i^0, H_i, Y_i^1, and O. In the following, we will assume that both the inputs and the weights of the network are bounded quantities, in particular: $x \in [0, 1]$ and $W \subseteq [-w_{\max}, w_{\max}]$. As usual, the activation function of the neurons is the logistic function $f(x) = 1/(1+\exp(-x))$ except in regression problems where the output coincides with the stimulus Y^1.

Analogously to the MLP, it has been showed [8] that training an IAMLP can be performed defining an error function and rewriting the back propagation algorithm for the case where the weights of the network are intervals (IABP [8], [11]).

III. Worst Case Analysis of Inaccuracies

We develop here an inaccuracy analysis assuming that the weights can be implemented with a prescribed tolerance: this implies that a weight can be described through an interval

$$[w-\Delta w,\, w+\Delta w]=[w^L,\, w^U] \tag{4}$$

where the interval represents all the possible values that a weight w can assume due to noise, quantization errors or any other undesired effect. For example, in digital implementations $\Delta w = Q/2$ where Q is the quantization step.

A. General Inaccuracy Bounds

We compute the bounds on the inaccuracy of the variables of the network in a backward way, starting from the output and going toward the input.

Let us consider the output of the network $O=[o-\Delta o,\, o+\Delta o]$ and the corresponding stimulus $Y^1=[y^1-\Delta y^1,\, y^1+\Delta y^1]$. Using the fact that the logistic function is strictly increasing, we have

$$\Delta y^1 \le 4\Delta o \tag{5}$$

by noting that the maximum derivative of the activation function (1/4) is obtained at the origin.

Proceeding backward we can bound the inaccuracy at the output of the hidden layer (Δh). Let N_{w^1} be the number of weights, including the bias, of the output layer, then

$$Y^1 \supseteq \sum_{i=1}^{N_{w^1}} [h_i-\Delta h,\, h_i+\Delta h][w_i^1-\Delta w^1,\, w_i^1+\Delta w^1]. \tag{6}$$

Using the rules of interval arithmetic we can expand the right size of (6)

$$\begin{aligned} Y^1 \supseteq \sum_{i=1}^{N_{w^1}} h_i w_i^1 &+ \sum_{W_i^1>0} [-h_i\Delta w^1 - w_i^1\Delta h + \Delta h\Delta w^1, \\ &\qquad h_i\Delta w^1 + w_i^1\Delta h + \Delta h\Delta w^1] \\ &+ \sum_{0\in W_i^1} [-h_i\Delta w^1 + w_i^1\Delta h - \Delta h\Delta w^1, \\ &\qquad h_i\Delta w^1 + w_i^1\Delta h + \Delta h\Delta w^1] \\ &+ \sum_{W_i^1<0} [-h_i\Delta w^1 + w_i^1\Delta h - \Delta h\Delta w^1, \\ &\qquad h_i\Delta w^1 - w_i^1\Delta h - \Delta h\Delta w^1]. \end{aligned} \tag{7}$$

Let $N_{w^1}=N_{w^1}^+ + N_{w^1}^- + N_{w^1}^0$ where $N_{w^1}^+$, $N_{w^1}^-$ are, respectively, the number of positive and negative weights and $N_{w^1}^0$ is the number of weights containing zero. Recalling that $W_i^1 \in [-w_{\max}, w_{\max}]$, we can write

$$w_i^1 \in \begin{cases} [\Delta w^1,\, w_{\max}^1-\Delta w^1] & W_i^1>0 \\ [-\Delta w^1,\, \Delta w^1] & 0\in W_i^1 \\ [-w_{\max}^1+\Delta w^1,\, -\Delta w^1] & W_i^1<0 \end{cases} \tag{8}$$

and find the following relation:

$$\begin{aligned} [-\Delta y^1,\, \Delta y^1] \supseteq [&-N_{w^1}\Delta w^1 - (N_{w^1}-N_{w^1}^0)w_{\max}^1\Delta h \\ &- 2(N_{w^1}^0-N_{w^1}^+)\Delta h\Delta w^1 \\ &N_{w^1}\Delta w^1 + (N_{w^1}-N_{w^1}^0)w_{\max}^1\Delta h \\ &+ 2(N_{w^1}^0-N_{w^1}^-)\Delta h\Delta w^1] \end{aligned} \tag{9}$$

from which the main result follows:

$$\Delta h \le \frac{\Delta y^1 - N_{w^1}\Delta w^1}{(N_{w^1}-N_{w^1}^0)w_{\max}^1 + 2(N_{w^1}^0 - \min\{N_{w^1}^+,\, N_{w^1}^-\})\Delta w^1}. \tag{10}$$

For simplicity, we have considered the bias in the same way of the other weights: in this case (10) bounds the inaccuracy of its fixed input.

Equation (10) can be easily simplified when the following reasonable assumptions on the network hold: 1) positive and negative weights are approximately equally numerous ($N_w^+ \simeq N_w^-$); 2) weights containing zero are very rare ($N_w^0 \simeq 0$); and 3) the inaccuracies are negligible with respect to the weights range ($w_{\max}^1 \gg \Delta w^1$). Assumption 2) is supported by the fact that $0\in W_i$ implies $-\Delta w^1 \le w_i^1 \le \Delta w^1$.

The simplified bound is

$$\Delta h \le \frac{\Delta y^1 - N_{w^1}\Delta w^1}{N_{w^1} w_{\max}^1}. \tag{11}$$

Equation (11) gives us an intuitive relation between the inaccuracy of the weights and the inaccuracy of the stimulus, considering that $\Delta h > 0$

$$\frac{\Delta y^1}{\Delta w^1} > N_{w^1}. \tag{12}$$

Similar bounds can be easily derived for the weights of the first layer (W_{ij}^0), for each hidden neuron i, relating Δx, Δy_i^0, and Δw^0.

If desired, less stringent bounds can be found, starting from (7). In fact, if we know both w and Δw, we can compute the error of the network as a function of these quantities and its inputs, using (2) and (3). This issue will be developed in the following section, but it is important to note that the above bounds are valid for any input pattern, any number of hidden neurons or inputs and any weight distribution. They are useful as general guidelines and in the synthesis of circuits by inverting the formulas.

B. Inaccuracy Bounds in Practice

In this example we target a digital implementation, therefore the weights inaccuracies are due to a quantization effect. For simplicity we will choose $\Delta w^0 = \Delta w^1 = \Delta w = Q/2$, where Q is the quantization step and $w_{\max} = \max\{w_{i,\max}^0,\, w_{\max}^1\}$.

The inaccuracy of the output is

$$\Delta o = \frac{o_{\max}-o_{\min}}{2(2^{B_o}-1)} \tag{13}$$

where $O\subseteq[o_{\min},\, o_{\max}]$ and B_o is the number of bits used to describe it. If $B_o=1$ then the output of the network is purely binary, while $B_o>1$ allows a finer evaluation of the output of the network. The output range depends on the activation function of the neuron: if we choose the logistic function, then $[o_{\min},\, o_{\max}]=[0,\, 1]$; if the output is linear (e.g., in regression problems) then $[o_{\min},\, o_{\max}] \subseteq [-N_{w^1}w_{\max}^1,\, N_{w^1}w_{\max}^1]$.

From (10), or (11), the accuracy needed at the output of the hidden layer can be derived as a function of the accuracy of the weights. In practice, we would like to find a value of

Δw as large as possible in order to minimize the number of bits needed to implement the weights. In fact, the following obvious relation holds:

$$B_w \geq \log_2 \left(\frac{2w_{\max}}{\Delta w} + 1 \right) \tag{14}$$

where B_w is the word size in bits of the weights.

Let us suppose that a trained IAMLP with weights $W_i \subseteq [-w_{\max}, w_{\max}]$ has been obtained, for which the output error is less or equal to the desired error. We know that for any particular choice of weights $w_i^* \subseteq W_i$, the network will behave as expected; then we can choose the following quantization step:

$$Q = 2\Delta w = \min_i \left(w_i^U - w_i^L \right). \tag{15}$$

With this choice we guarantee that at least one of the quantized values lies inside an interval weight W_i.

Let us consider a trained MLP whose weights w_i have been found by a conventional learning algorithm. We build an IAMLP by transforming each real weight in an interval using a slack variable $\epsilon > 0$: $[w_i - \epsilon, w_i + \epsilon]$. Then, we are faced to solve the following constrained optimization problem: maximize ϵ with the constraint $E(\epsilon) \leq E_{\max}$ where $E(\epsilon)$ is the error function of the network and $E_{\max}$ is its maximum admissible value, as defined by the user. This is a simple monodimensional optimization problem, albeit a constrained one, noting that $E(\epsilon)$ is a nondecreasing function of ϵ.

Sometimes, in practice, it is worthwhile to find different quantization steps for each of the two layers of the MLP, as in the previous sections; in this case it is sufficient to consider two slack variables ϵ_1, ϵ_2 and perform a two-dimensional optimization.

IV. Experimental Results

We present here some experimental results on both artificial and real classification problems.

Some comments are necessary in order to correctly interpret the output of the IAMLP when used for classification problems. It is worthwhile to recall that the output is an estimate of the conditional probability of a class C_i, where the index i spans all the possible classes, with respect to the input vector x [12]; therefore, the interval output indicates the range of admissible values of such probability. Let us consider a simple two-class case (i.e., $i = \{0, 1\}$): we are usually interested in the decision deriving from the observation of $p(C_{0,1}/x)$, therefore, observing the interval output O, we have

$$x \in \begin{cases} C_1 & O > \frac{1}{2} \\ \{C_0, C_1\} & \frac{1}{2} \in O \\ C_0 & O < \frac{1}{2}. \end{cases} \tag{16}$$

The case $1/2 \in O$ corresponds to an output interval for which both classes are admissible. More subtle analysis of the interval output of the network can be performed, especially in the multiclass case, but the discussion of these issues is out of the scope of this paper and we refer the reader to [13] for further insights. Condition (16) corresponds to a correct classification of a pattern p with target $t_p = \{0, 1\}$ if and only if

$$\left(o_p^U(1 - t_p) + o_p^L t_p - \tfrac{1}{2}\right)\left(t_p - \tfrac{1}{2}\right) > 0. \tag{17}$$

In the following experiments, the error function E will be the percentage of patterns not satisfying (17).

TABLE I
Effect of Weight Inaccuracies on Network Output

B_w	$r^U - r^L$
15	5.3×10^{-3}
13	2.1×10^{-2}
11	8×10^{-2}

The artificial example is a simple two class problem proposed by Lippmann [14]: a square unit area is divided in two regions by a centered circle of area $1/2$, corresponding to a radius $r = 1/\sqrt{2\pi} \approx 0.399$. We generated 1000 random points inside the square, assigned them to two different classes according to their position relative to the circle and, finally, trained a network with four hidden neurons. The discriminating surface of this network, after learning, is able to reproduce almost perfectly the circle that separates the two classes. Obviously, the output of the IAMLP is an interval that corresponds, approximately, to two circles of radii r^L and r^U: they are obtained by fitting the curves corresponding to $o^L = 1/2$ and $o^U = 1/2$.

In Table I the approximate difference of the radii of the two circles is reported, when varying the number of bits used to represent the weights of the network. This gives a clear indication of the influence of inaccuracies on the size of the uncertainty region ($1/2 \in O$).

The results on real-world problems are presented using plots of number of bits B_w versus the percentage of misclassification (Fig. 1). The curves are obtained deriving from the actual number of bits the size of the intervals $[w - \Delta w, w + \Delta w]$ from (14) and applying this value to the IAMLP, as described by (2) and (3). Finally, using (17), the percentage of errors can be computed. Worst case bounds are computed starting from (14) and using (10); this allows to quantify the number of additional bits required, in general, to correctly implement the network without any additional information.

The "Breast Cancer Wisconsin" dataset has been obtained from the UCI repository[1] and is a two-class classification problem. We trained a network with nine inputs and two hidden neurons on the 683 patterns of this dataset, obtained discarding the ones with missing values. Equation (10) suggests that the trained network can be implemented with only 12 bits per weight. Using the less general bounds, as depicted in (Fig. 1), it is possible to implement the network with eight bits or even five bits if the user can accept a classification error of approximately 10%.

[1]UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

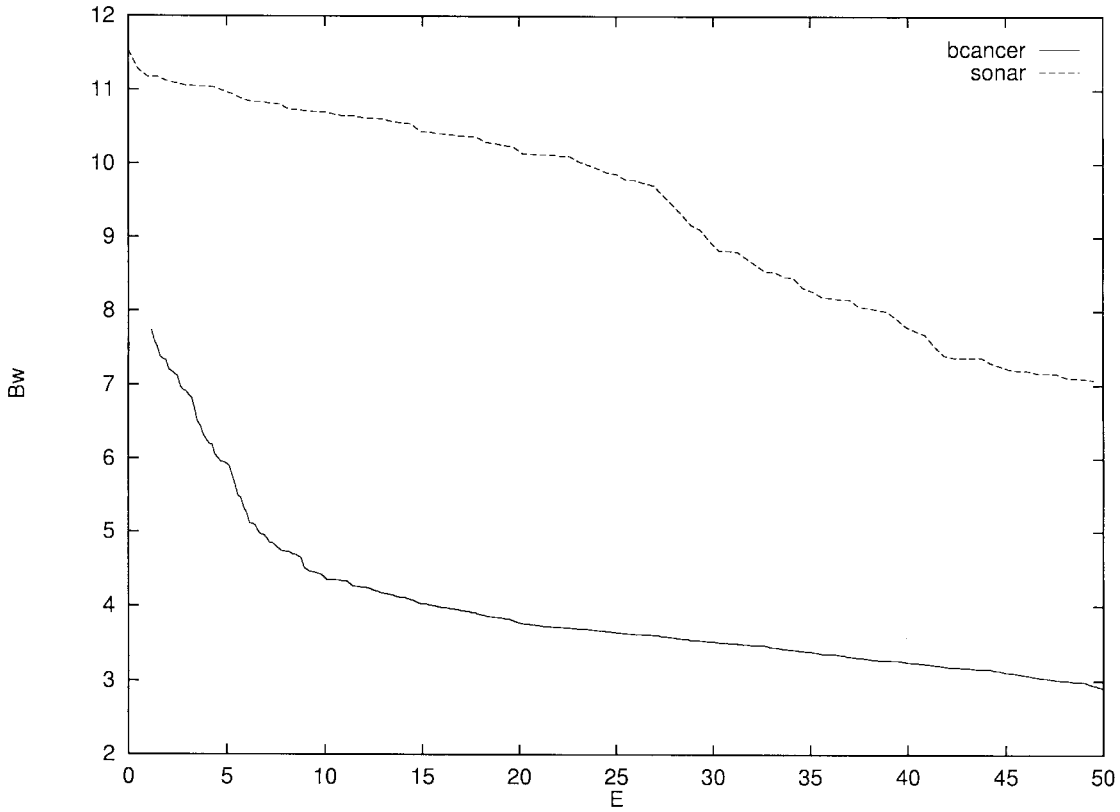

Fig. 1. The real-world problems: precision required B_w versus percentage of misclassifications E.

The "Sonar" dataset is another well-known example of a difficult, although linearly separable, classification problem; it consists of 208 patterns, each one composed of 60 features. The results are obtained training a single perceptron, without hidden neurons. We can observe that the problem requires a greater accuracy, and the dependency is less regular than in the previous case. As a comparison, note that an accuracy of 11 bits is required if we want to keep the error below 10%. The general bound gives a minimum accuracy of 15 bits. Note that these values are comparable with similar results obtained empirically on the same classification problem [4].

V. Conclusion

In this work we presented a method for estimating the admissible level of accuracy for multilayer perceptron weights, given the desired output error. Interval arithmetic is used to express the relevant quantities. This results in a compact representation of the problem and in simple calculations. The proposed approach differs from many other techniques in that it is a worst case analysis; therefore, its estimates are independent of data distributions and do not require any assumption on probability densities.

References

[1] S. Sakaue, T. Kohda, H. Yamamoto, S. Maruno, and Y. Shimeki, "Reduction of required precision bits for back-propagation applied to pattern recognition," *IEEE Trans. Neural Networks*, vol. 4, pp. 270–275, Mar. 1993.

[2] J. Y. Choi and C. H. Choi, "Sensitivity analysis of multilayer perceptrons with differentiable activation functions," *IEEE Trans. Neural Networks*, vol. 3, pp. 101–107, Jan. 1992.

[3] Y. Xie and M. A. Jabri, "Analysis of the effects of quantization in multilayer neural networks using a statistical model," *IEEE Trans. Neural Networks*, vol. 3, pp. 334–338, Mar. 1992.

[4] M. Hoehfeld and S. E. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 602–611, July 1992.

[5] G. Dündar and K. Rose, "The effects of quantization on multilayer neural networks," *IEEE Trans. Neural Networks*, vol. 6, pp. 1446–1451, Nov. 1995.

[6] G. Alefeld and J. Herzberger, *Introduction to Interval Computation.* Reading, MA: Addison-Wesley, 1986.

[7] H. Ishibuchi and H. Tanaka, "An extension of the BP-algorithm to interval input vectors—Learning from numerical data and expert's knowledge," in *Proc. 1991 IEEE Int. Joint Conf. Neural Networks*, Singapore, Nov. 18–21, 1991, vol. 2, pp. 1588–1593.

[8] H. Ishibuchi, H. Tanaka, and H. Okada, "An architecture of neural networks with interval weights and its application to fuzzy regression analysis," *Fuzzy Sets Syst*, vol. 57, pp. 27–39, 1993.

[9] D. Anguita, S. Ridella, S. Rovetta, and R. Zunino, "Limiting the effects of weight errors in feed forward networks using interval arithmetic," in *Proc. IEEE Int. Conf. Neural Networks*, Washinghton DC, June 3–6, 1996, pp. 414–417.

[10] R. B. Kearfott, M. Dawande, K. Du, and C. Hu, "INTLIB: A portable fortran 77 interval standard-function library," *ACM Trans. Math. Software*, vol. 20, no. 4, pp. 447–459, Dec. 1994.

[11] D. Anguita, S. Ridella, S. Rovetta, and R. Zunino, "Incorporating *a priori* knowledge into neural networks," *Electron. Lett.*, vol. 31, no. 22, pp. 1930–1931, Oct. 1995.

[12] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Trans. Neural Networks*, vol. 1, Dec. 1990.

[13] H. Ishibuchi and A. Miyazaki, "Determination of inspection order for classifying new samples by neural networks," in *Proc. IEEE Int. Conf. Neural Networks*, Orlando, FL, June 27–29, 1994, vol. 5, pp. 2907–2910.

[14] L. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust., Speech, Signal Processing Mag.*, vol. 4, pp. 4–22, 1987.