

A Generalization of backpropagation to interval arithmetic

メタデータ	言語: English 出版者: 公開日: 2017-10-03 キーワード (Ja): キーワード (En): 作成者: Hernandez, C.A., Espi, J., Nakayama, Kenji メールアドレス: 所属:
URL	http://hdl.handle.net/2297/11892

A GENERALIZATION OF BACKPROPAGATION TO INTERVAL ARITHMETIC

C.A. Hernández^{1,2}, J. Espí^{1,2}, K. Nakayama³.

¹Department of Computers and Electronics, Valencia University, Doctor Moliner, 50. 46100 Burjassot (Valencia). SPAIN
E-mail: HERNANDC@EVALUN11.BITNET

² Robotics Institute, Valencia University, Hugo de Moncada, 4. Valencia. SPAIN

³Department of Electrical and Computers Engineering, Kanazawa University, Kodatsuno 2-20-40, Kanazawa-shi, Ishikawa-ken. 920 JAPAN
E-mail: Nakayama@haspnn1.cc.kanazawa-u.ac.jp

Abstract

In this paper we present a new generalization of the Backpropagation learning algorithm by using interval arithmetic.

The proposed algorithm contains Backpropagation as a particular case and permits the use of training samples and targets which can be indistinctly points and intervals.

Among the possible applications of this algorithm we report its usefulness to integrate expert's knowledge and experimental samples and also its ability to handle "don't care attributes" in a simple and natural way in comparison with Backpropagation.

1.- Introduction

One of the first successful Neural Networks training paradigms was Backpropagation with Multilayers Neural Networks [1], this paradigm is also one of the most used nowadays.

There has been many modifications of this training algorithm, one of them is the extension to interval arithmetic proposed in [2]. Interval arithmetic [3] allows Backpropagation to use real vectors and interval vectors as training samples and targets for a Neural Network. In this way an interval at the input of the Neural Network is converted to another interval at the output.

A severe limitation of the algorithm proposed in [2] is that it has only an output unit and can only be applied to classification problems with two classification classes.

In this paper we propose a new and direct extension of Backpropagation to interval arithmetic which will be called Interval Arithmetic Backpropagation (IABP). The proposed algorithm can be used with any number of output units and every equation reduces to the equations of Backpropagation for the case of a real vector input, under this point of view IABP can be considered a generalization of BP.

We show that this algorithm can integrate experts knowledge and training samples in the training set in the same way of [2] and we also show that it can handle "don't care attributes" in a very simple and advantageous way in comparison with Backpropagation [4].

In the next sections we introduce the basic equations of interval arithmetic and we present the new algorithm.

2.- Interval Arithmetic.

The basic equations of interval arithmetic [2,3] which are useful in the following development are:

Sum of intervals:

$$A + B = [a^L, a^U] + [b^L, b^U] = [a^L + b^L, a^U + b^U]$$

Product by a real number:

$$m \cdot A = m \cdot [a^L, a^U] = \begin{cases} [m \cdot a^L, m \cdot a^U] & , \text{ if } m \geq 0 \\ [m \cdot a^U, m \cdot a^L] & , \text{ if } m < 0 \end{cases}$$

Exponential function:

$$\exp A = \exp [a^L, a^U] = [\exp a^L, \exp a^U]$$

3.- Interval Arithmetic Backpropagation.

First of all, we should define a generalization to interval arithmetic of the transfer function of the neuron:

$$f(\text{Net}) = f([net^L, net^U]) = [f(net^L), f(net^U)] \quad \text{where:} \quad f(x) = \frac{1}{1 + \exp(-x)}$$

The definition is consistent because $f(x)$ increases monotonically. Next we will define the relationships in the neural network.

The input patterns in general will be interval vectors:

$$I_{p,i} = [I_{p,i}^L, I_{p,i}^U]$$

The output of the hidden units:

$$H_{p,j} = f(\text{Net}_{p,j}) \quad \text{where:} \quad \text{Net}_{p,j} = \sum_{i=1}^{N_{inputs}} w_{j,i} \cdot I_{p,i} + \theta_j$$

and:

$$net_{p,j}^L = \sum_{i=1, w_{j,i} \geq 0}^{N_{inputs}} w_{j,i} I_{p,i}^L + \sum_{i=1, w_{j,i} < 0}^{N_{inputs}} w_{j,i} I_{p,i}^U + \theta_j, \quad net_{p,j}^U = \sum_{i=1, w_{j,i} \geq 0}^{N_{inputs}} w_{j,i} I_{p,i}^U + \sum_{i=1, w_{j,i} < 0}^{N_{inputs}} w_{j,i} I_{p,i}^L + \theta_j$$

Analogously, the output of the Neural Network:

$$O_{p,k} = f(\text{Net}_{p,k}) = f\left(\sum_{i=1}^{N_{hidden}} w_{k,i} \cdot H_{p,i} + \theta_k\right)$$

The targets will also be, in general, a interval vector:

$$T_{p,k} = [t_{p,k}^L, t_{p,k}^U]$$

And the Mean Square Error function can be defined as a generalization of BP Error function:

$$E_p = \frac{1}{4} \cdot \sum_{k=1}^{N_{outputs}} \{ (t_{p,k}^L - O_{p,k}^L)^2 + (t_{p,k}^U - O_{p,k}^U)^2 \}$$

Like in BP, the learning in IABP is the process to minimize the above cost function, the weights are changed according to the following function:

$$\Delta w_{j,i}(t+1) = \eta \cdot \left(-\frac{\partial E_p}{\partial w_{j,i}}\right) + \beta \cdot \Delta w_{j,i}(t)$$

where η is the step size and β the momentum. The partial derivatives are calculated in the appendix, the only precaution in the calculus is to consider the sign of $w_{j,i}$.

The percentage correct error function can also be defined for classification problems. A possible definition with threshold 0.5 is: suppose the target for a pattern P is $T_j = [1, 1]$ and $T_i = [0, 0]$ for $i \neq j$ and $i=1 \dots N_{output}$. Then, this pattern should count positively in the percentage if $O_{p,j}^U \geq 0.5$ and $O_{p,i}^L < 0.5$.

4.- Experimental results.

In this section three bidimensional examples are presented, Fig.1,2,3. The example in Fig.1 corresponds to a mixture of real and interval vectors in the training set, Table 1. In Fig.1 it is represented the training samples and a line which represents the threshold 0.5 of the output units, i.e., the achieved classification. This result correspond to a Neural Network with 2 output units, 5 hidden units and 2 inputs. The example in Fig.2 corresponds to the results of a Neural

Network with 2 inputs, 7 hidden units and 3 outputs, the data set is in Table 2. Finally, the example 3 correspond to a Neural Network with 2 inputs, 8 hidden units and 5 outputs. The data set is in Table 3. In all the examples it is achieved a perfect classification of the training set.

Integration of expert's knowledge and sample data. The kind of expert's knowledge considered is a set of "if ...then" rules like the following one: if $X_{p,1} \in [A_1, B_1] \dots$ and $X_{p,n} \in [A_n, B_n]$ then $X_p \in G_K$ where X_p is a pattern vector and G_K its classification class.

This type of rules can be easily codified by using interval arithmetic ($[A_i, B_i]$ will be the inputs of the Neural Network), and can be included in the training set together with sample data, e.g. Example 1.

Handle of "don't care attributes".

The codification of "don't care attributes" for normal BP, in the case of discrete, inputs was already studied in [4]. Suppose we have the sample vector: [1,2,D,2] where D means "don't care". For normal BP we should include in the training set all the vectors which result from codifying D with all its possible values or with two values, the maximum possible value d_{max} and minimum d_{min} . This yields an exponential increase in the training set if we have more than one D in a vector. With interval arithmetic we can codify D like an interval $[d_{min}, d_{max}]$, the result is only a training vector and this approach is valid for the discrete and the continuous case. We have reproduced the best result in [4] for the three examples used there IS1, IS2 and IS3 by using an interval arithmetic codification of the "don't care attributes". The training set in our case was smaller.

Table 1. Training set of example 1.

Classes.	Training Samples.						Target
C.1 Inp.X	[4,4]	[8,8]	[11,11]	[13,13]	[13,13]	[0,10]	[1,0]
C.1 Inp.Y	[11,11]	[11,11]	[3,3]	[6,6]	[10,10]	[0,10]	[1,0]
C.2 Inp.Y	[14,14]	[15,15]	[2,2]	[4,4]	[15,15]	[0,20]	[0,1]
C.2 Inp.X	[2,2]	[6,6]	[14,14]	[15,15]	[15,15]	[16,20]	[0,1]

Table 2. Training set of example 2.

Classes.	Training Samp.	Target	
C.1 Inp.X	[8,14]	[14,18]	[1,0,0]
C.1 Inp.Y	[16,20]	[5,15]	[1,0,0]
C.2 Inp.X	[1,5]	[6,12]	[0,1,0]
C.2 Inp.Y	[8,14]	[9,11]	[0,1,0]
C.3 Inp.X	[4,10]	[13,19]	[0,0,1]
C.3 Inp.Y	[1,5]	[1,3]	[0,0,1]

Table 3. Training set of example 3.

Classes.	Training Samp.	Target	
C.1 Inp.X	[14,18]	[13,19]	[1,0,0,0,0]
C.1 Inp.Y	[5,15]	[1,3]	[1,0,0,0,0]
C.2 Inp.X	[6,12]		[0,1,0,0,0]
C.2 Inp.Y	[9,11]		[0,1,0,0,0]
C.3 Inp.X	[1,5]		[0,0,1,0,0]
C.3 Inp.Y	[8,14]		[0,0,1,0,0]
C.4 Inp.X	[4,10]		[0,0,0,1,0]
C.4 Inp.Y	[1,5]		[0,0,0,1,0]
C.5 Inp.X	[8,14]		[0,0,0,0,1]
C.5 Inp.Y	[16,20]		[0,0,0,0,1]

Fig.1. Example 1, classification.

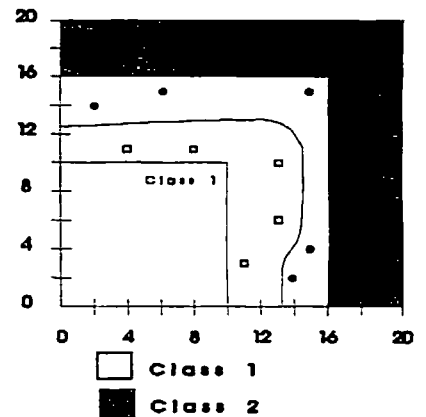


Fig.2. Example 2, classification.

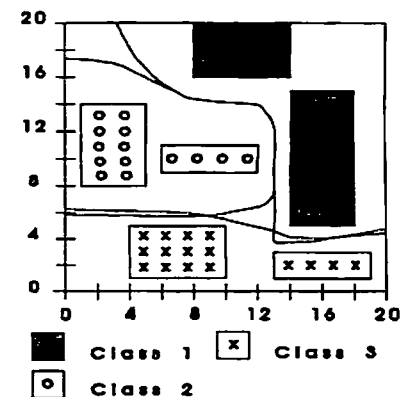
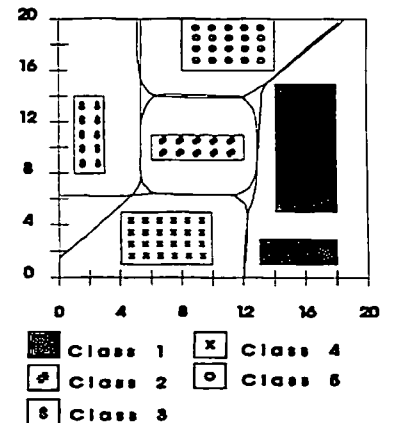


Fig.3. Example 3, classification.



5.- Conclusion and Discussion.

We have generalized the BP training algorithm to interval arithmetic and we have shown two possible applications of this new algorithm: integration of expert's knowledge and sample data and the handle of "don't care attributes".

In general, this algorithm will add flexibility to the codification of inputs and targets. For example, in the case we have a strong subjectivity and imprecision (e.g., the codification of symptoms in a medical diagnosis classification problem) the use of intervals in the codification may reduce this subjectivity and imprecision, perhaps it would be a more appropriate way to codify these cases.

6.- Appendix.

For the weights between the hidden units and the outputs:

$$\frac{\partial E_p}{\partial w_{k,i}} = \frac{\partial E_p}{\partial O_{p,k}^L} \frac{\partial O_{p,k}^L}{\partial net_{p,k}^L} \frac{\partial net_{p,k}^L}{\partial w_{k,i}} + \frac{\partial E_p}{\partial O_{p,k}^U} \frac{\partial O_{p,k}^U}{\partial net_{p,k}^U} \frac{\partial net_{p,k}^U}{\partial w_{k,i}}$$

which gives the result:

$$\frac{\partial E_p}{\partial w_{k,i}} = \begin{cases} \delta_{p,k}^L H_{p,i}^L + \delta_{p,k}^U H_{p,i}^U, & \text{if } w_{k,i} \geq 0 \\ \delta_{p,k}^L H_{p,i}^U + \delta_{p,k}^U H_{p,i}^L, & \text{if } w_{k,i} < 0 \end{cases} \quad \text{where:} \quad \begin{cases} \delta_{p,k}^L = -\frac{1}{2} (t_{p,k}^L - O_{p,k}^L) O_{p,k}^L (1 - O_{p,k}^L) \\ \delta_{p,k}^U = -\frac{1}{2} (t_{p,k}^U - O_{p,k}^U) O_{p,k}^U (1 - O_{p,k}^U) \end{cases}$$

For the weights between hidden units and input:

$$\begin{aligned} \frac{\partial E_p}{\partial w_{j,i}} = & \sum_{k, w_{k,i} \geq 0} \left\{ \frac{\partial E_p}{\partial O_{p,k}^L} \frac{\partial O_{p,k}^L}{\partial net_{p,k}^L} \frac{\partial net_{p,k}^L}{\partial H_{p,i}^L} \frac{\partial H_{p,i}^L}{\partial net_{p,j}^L} \frac{\partial net_{p,j}^L}{\partial w_{j,i}} \right\} + \sum_{k, w_{k,i} < 0} \left\{ \frac{\partial E_p}{\partial O_{p,k}^L} \right. \\ & \left. \frac{\partial O_{p,k}^L}{\partial net_{p,k}^L} \frac{\partial net_{p,k}^L}{\partial H_{p,i}^U} \frac{\partial H_{p,i}^U}{\partial net_{p,j}^U} \frac{\partial net_{p,j}^U}{\partial w_{j,i}} \right\} + \sum_{k, w_{k,i} \geq 0} \left\{ \frac{\partial E_p}{\partial O_{p,k}^U} \frac{\partial O_{p,k}^U}{\partial net_{p,k}^U} \frac{\partial net_{p,k}^U}{\partial H_{p,i}^L} \frac{\partial H_{p,i}^L}{\partial net_{p,j}^L} \frac{\partial net_{p,j}^L}{\partial w_{j,i}} \right\} \\ & + \sum_{k, w_{k,i} < 0} \left\{ \frac{\partial E_p}{\partial O_{p,k}^U} \frac{\partial O_{p,k}^U}{\partial net_{p,k}^U} \frac{\partial net_{p,k}^U}{\partial H_{p,i}^U} \frac{\partial H_{p,i}^U}{\partial net_{p,j}^U} \frac{\partial net_{p,j}^U}{\partial w_{j,i}} \right\} \end{aligned}$$

which gives the result:

$$\frac{\partial E_p}{\partial w_{j,i}} = \begin{cases} \delta h_{p,j}^L I_{p,i}^L + \delta h_{p,j}^U I_{p,i}^U, & \text{if } w_{j,i} \geq 0 \\ \delta h_{p,j}^L I_{p,i}^U + \delta h_{p,j}^U I_{p,i}^L, & \text{if } w_{j,i} < 0 \end{cases} \quad \text{where:} \quad \begin{cases} \delta h_{p,j}^L = \left(\sum_{k, w_{k,i} \geq 0} \delta_{p,k}^L w_{k,j} + \sum_{k, w_{k,i} < 0} \delta_{p,k}^U w_{k,j} \right) \cdot H_{p,j}^L \cdot (1 - H_{p,j}^L) \\ \delta h_{p,j}^U = \left(\sum_{k, w_{k,i} \geq 0} \delta_{p,k}^U w_{k,j} + \sum_{k, w_{k,i} < 0} \delta_{p,k}^L w_{k,j} \right) \cdot H_{p,j}^U \cdot (1 - H_{p,j}^U) \end{cases}$$

7.- References.

- [1] D.E. Rumelhart, J.L. McClelland and the PDP Research Group. *Parallel Distributed Processing* Vol.1, MIT Press, Cambridge, 1988.
- [2] H. Ishibuchi and H. Tanaka. "An extension of the BP-Algorithm to Interval Input Vectors -Learning from Numerical Data and Expert's Knowledge-". IJCNN-91. Singapoure. pp. 1588-1593.
- [3] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press. New York. 1983.
- [4] H.M. Lee and C. Hsu. "The Handling of Don't Care Attributes". IJCNN-91 . Singapoure. pp. 1085-1091.