A SMOOTHING ALGORITHM FOR TRAINING MAX-MIN NEURAL NETWORKS

[1]LONG LI, [1]TIAN XU, [2]YAN LIU, [3]JIE YANG

[1]Department of Mathematics and Computational Science, Hengyang Normal University, Hengyang, 421008, China

[2]Department of Applied Mathematics, Dalian Polytechnic University, Dalian, 116034, China

[3]School of Mathematical Sciences, Dalian University of Technology, Dalian, 116024, China

ABSTRACT

In this paper, a smoothing algorithm for training max-min neural networks is proposed. Specifically, we apply a smooth function to approximate max-min functions and use this smoothing technique twice, once to eliminate the inner min operator and once to eliminate the max operator. In place of actual network output by its approximation function, we use all partial derivatives of the approximation function with respect to weight to substitute those of the actual network output. Then, the smoothing algorithm is constructed by the gradient descent method. This algorithm can also be used to solve fuzzy relational equations. Finally, two numerical examples are provided to show the effectiveness of our smoothing algorithm for training max-min neural networks.

Keywords: *Smoothing Algorithm, Max-min Neural Networks, Max-min Functions, Approximation*

1. INTRODUCTION

In recent years, fuzzy neural networks have attracted considerable attention for their useful applications in such fields as control, pattern recognition, image processing, forecasting, etc., as described in [1, 2, 3, 4, 5, 6]. In all these applications, there are different fuzzy neural-network architectures proposed for different purposes and fields. However, no matter how different architectures these neural networks have, two important operations min and max are often involved. Among various architectures, the so-called max-min neural networks have been extensively studied and applied [6, 7, 8, 9].

Before we proceed, let us firstly introduce the max-min neural network considered in this paper and the problem of the max-min neural-network learning. Suppose that this network has n input nodes and one output node, and that we are supplied with a set of training samples $\{X^s, T^s\}_{s=1}^S \in [0,1]^n \times [0,1]$, where X^s is an n dimension input vector, T^s is the corresponding desired output, and S is the number of training samples. The topological structure of this network is illustrated in Fig. 1. Using w_i to denote the weight between node i in input layer and output

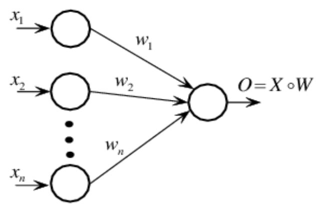

Figure 1: Max-Min Neural Networks

node and O^s to denote the actual network output corresponding to the training sample X^s, the I/O relationship of the max-min neural network is described by

$$O^s = X^s \circ W = \bigvee_{i=1}^{n} (x_i^s \wedge w_i)$$

where $\vee$ and $\wedge$ are max and min operations respectively, $\circ$ is the composition operation of $\vee$ and $\wedge$, and $X^s = (x_1^s, x_2^s, \cdots, x_n^s)$ and $W = (w_1, w_2, \cdots, w_n)$ are the s-th input vector and the weight vector, respectively. Define the cost function $J(W)$ for this max-min neural network as follows:

$$J(W) = \frac{1}{2}\sum_{s=1}^{S}(T^s - O^s)^2 = \frac{1}{2}\sum_{s=1}^{S}(T^s - X^s \circ W)^2$$

Our task is to train this max-min neural network such that it can fit, up to a given precision, the given set of desired network input and output pairs, i.e., to find W^* such that

$$J(W^*) = \min J(W)$$

For this purpose, we use the conventional gradient descent method. First, we choose an arbitrary initial value W^0 and a constant learning rate $\eta > 0$. Then, the weight vector W is refined by the following iterative learning process

$$W^{k+1} = W^k + \Delta W^k = W^k - \eta \frac{\partial J(W^k)}{\partial W^k}, \quad k = 0, 1, 2, \cdots$$

where $\frac{\partial J(W^k)}{\partial W^k} = \frac{\partial J(W^k)}{\partial O^s} \frac{\partial O^s}{\partial W^k}$.

Because of the difficulty in the analysis of min and max operations, the training of this max-min neural network appears not to be approachable rigorously and systematically. The lack of an appropriate analytical tool for the min and max operations greatly limits their applicability. An initial attempt has been made in [10, 11] in trying to differentiate max-min operations and to apply them to the training of max-min neural networks. The key idea of their approach is using the unit step function. The derivatives of the functions max and min have a "crisp" behaviour (We name the algorithm derived by these derivatives as Algorithm 1.):

$$\frac{\partial(x \wedge w)}{\partial w} = \begin{cases} 1, & \text{if } x \geq w \\ 0, & \text{if } x < w \end{cases}$$
$$\frac{\partial(x \vee w)}{\partial w} = \begin{cases} 1, & \text{if } x \leq w \\ 0, & \text{if } x > w. \end{cases}$$

By applying this learning process, it is not guaranteed that the network will learn, obviously because the value of $\frac{\partial O^s}{\partial W^k}$ is null in the majority of cases. To improve this behaviour, a new procedure is developed in [12]. The authors use Gödel's implication and give a kind of so-called smoothed derivatives of the functions max and min as follows (We name the algorithm derived by these derivatives as Algorithm 2.):

$$\frac{\partial(x \wedge w)}{\partial w} = \begin{cases} 1, & \text{if } x \geq w \\ x, & \text{if } x < w \end{cases}$$

$$\frac{\partial(x \vee w)}{\partial w} = \begin{cases} 1, & \text{if } x \leq w \\ w, & \text{if } x > w. \end{cases}$$

However, the derivatives of functions $x \wedge w$ and $x \vee w$ in [10, 11, 12] are formal and can not hold for their corresponding operations in mathematics.

To overcome this shortcoming, the authors have made another attempt in [13] in developing a rigorous theory for the differentiation of max-min functions by means of functional analysis, and derived an algorithm for training max-min neural networks. The derivatives of the functions max and min are defined as (We name the algorithm derived by these derivatives as Algorithm 3.):

$$\frac{\partial(f(x) \wedge g(x))}{\partial x} = lor(g(x) - f(x))\frac{\partial f(x)}{\partial x} + lor(f(x) - g(x))\frac{\partial g(x)}{\partial x}$$
$$\frac{\partial(f(x) \vee g(x))}{\partial x} = lor(f(x) - g(x))\frac{\partial f(x)}{\partial x} + lor(g(x) - f(x))\frac{\partial g(x)}{\partial x}$$

where the function $lor(x) = \begin{cases} 1, & \text{if } x > 0 \\ \frac{1}{2}, & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases}$. Then the partial derivatives $\frac{\partial O^s}{\partial w_i}$ have the following representation:

$$\frac{\partial O^s}{\partial w_i} = lor\left((w_i \wedge x_i^s) - \bigvee_{j \neq i} (w_j \wedge x_j^s) \right) \cdot lor(x_i^s - w_i)$$

Although a rigorous mathematical analysis for the differentiation of max-min functions is given, the learning performance of Algorithm 3 for training the max-min neural network is similar to the learning performance of Algorithm 1.

In this paper, a smoothing algorithm for training max-min neural networks is proposed. Specifically, we apply a smooth function to approximate max-min functions and use this smoothing technique twice, once to eliminate the inner minoperator and once to eliminate the max operator. In place of actual network output by its approximation function, we use all partial derivatives of the approximation function with respect to weight to substitute those of the actual network output. Then, the smoothing algorithm is constructed by the gradient descent method. This algorithm can also be used to solve fuzzy relational equations. Finally, two numerical examples are provided to show the effectiveness of our smoothing algorithm for training max-min neural networks.

The rest of this paper is organized as follows. In Section 2, we introduce a smoothing method to approximate max-min functions. Section 3 gives our smoothing algorithm for training max-min neural networks. Two numerical examples are provided in Section 4 to show the effectiveness of our smoothing algorithm. Some brief conclusions are drawn in Section 5.

2. SMOOTHING METHOD TO APPROXIMATE MAX-MIN FUNCTIONS

Suppose $g(x) = \bigvee_{i=1}^{m} g_i(x)$, we first introduce the smoothing technique described in [14, 15] to approximate the max function $g(x)$ with the following exponential function

$$g(x,t) = t \ln \sum_{i=1}^{m} \exp\left(\frac{g_i(x)}{t}\right) \qquad (1)$$

with a parameter $t > 0$.

Since

$$\bigwedge_{i=1}^{m} f_i(x) = -\bigvee_{i=1}^{m} \left(-f_i(x)\right).$$

Let $g_i(x) = -f_i(x)$. By applying the smooth function $g(x,t)$ to approximate the function $\bigwedge_{i=1}^{m} f_i(x)$, we get

$$-g(x,t) = -t \ln \sum_{i=1}^{m} \exp\left(\frac{-f_i(x)}{t}\right) \qquad (2)$$

The following lemma summarizes some interesting properties of the function $g(x,t)$ defined by (1).

Lemma 1 Suppose $g_i(x)$ are all continuously differentiable functions,

$$g(x) = \bigvee_{i=1}^{m} g_i(x)$$

and $g(x,t)$ is defined by (1), then we have:

(i) $g(x,t)$ is increasing with respect to t, and $g(x) \le g(x,t) \le g(x) + t\ln m$;

(ii) $g(x,t)$ is continuously differentiable for all $t > 0$, and

$$\bigtriangledown_x g(x,t) = \sum_{i=1}^{m} \lambda_i(x,t) \bigtriangledown g_i(x),$$

where

$$\lambda_i(x,t) = \frac{\exp\left(g_i(x)/t\right)}{\sum_{j=1}^{m} \exp\left(g_j(x)/t\right)} \in (0,1), \quad \sum_{i=1}^{m} \lambda_i(x,t) = 1$$

Particularly, if $g_i(x)$ are all linear functions, then $g(x,t)$ is an infinite order differentiable function for all $t > 0$.

3. SMOOTHING ALGORITHM FOR TRAINING MAX-MIN NEURAL NETWORKS

To begin with, let a max-min neural network with n input nodes and one output node be given. With the same notations we have introduced in Section 1, the I/O relationship of the max-min neural network is described by

$$O^s = X^s \circ W = \bigvee_{i=1}^{n} (x_i^s \wedge w_i) \qquad (3)$$

and the cost function J for this max-min neural network is defined as

$$J(W) = \frac{1}{2}\sum_{s=1}^{S}(T^s - O^s)^2 = \frac{1}{2}\sum_{s=1}^{S}(T^s - X^s \circ W)^2 \quad (4)$$

Our task is to train this max-min neural network such that it can fit the given set of desired network input and output pairs to a given precision. For this purpose, we use the conventional idea of gradient descent to design an algorithm to minimize J. Since the max-min function $O^s(w_i)$ is not differentiable, it is difficult to use classical methods to derive the differentiation formulas for $O^s(w_i)$ with respect to w_i. As a remedy for this point, we apply the smoothing technique introduced in Section 2 twice to approximate max-min functions $O^s = \bigvee_{i=1}^{n} (x_i^s \wedge w_i)$ as follows:

Firstly, we use the smoothing technique to approximate the min function

$$x_i^s \wedge w_i$$

with the smooth approximation

$$g^s(w_i,t) = t\ln\left(\exp\left(x_i^s/t\right) + \exp\left(w_i/t\right)\right) \quad (5)$$

with a parameter $t < 0$.

Then, we use it again to approximate the max function

$$\bigvee_{i=1}^{n} g^s(w_i,t)$$

with the smooth approximation

$$G^s(w_i,t,h) = h\ln\sum_{i=1}^{n}\exp\left(\frac{g^s(w_i,t)}{h}\right) \qquad (6)$$

with two parameters $t < 0$ and $h > 0$.

Using the smooth function $G^s(w_i,t,h)$ to approximate the actual network output O^s, we can get the following result about the approximating precision.

Theorem 1 Let $O^s = \bigvee_{i=1}^{n} (x_i^s \wedge w_i)$ and $G^s(w_i,t,h) = h\ln \sum_{i=1}^{n} \exp\left(\frac{g^s(w_i,t)}{h}\right)$, where $g^s(w_i,t) = t\ln\left(\exp(x_i^s/t) + \exp(w_i/t)\right)$, $t<0$ and $h>0$. Then, we have

$$O^s + t\ln 2 \le G^s(w_i,t,h) \le O^s + h\ln n$$

Proof. Since $x_i^s \wedge w_i \le x_i$, $x_i^s \wedge w_i \le w_i$ and $t<0$, it is easy to get that

$$\exp\left(\frac{x_i^s \wedge w_i}{t}\right) \le \exp\left(\frac{x_i^s}{t}\right) + \exp\left(\frac{w_i}{t}\right) \le 2\exp\left(\frac{x_i^s \wedge w_i}{t}\right)$$

Then, we have

$$(x_i^s \wedge w_i) + t\ln 2 \le t\ln\left(\exp\left(x_i^s/t\right) + \exp\left(w_i/t\right)\right) \le (x_i^s \wedge w_i) \quad (7)$$

Similarly, we can get that

$$\bigvee_{i=1}^{n} g^s(w_i,t) \le G^s(w_i,t,h) \le \bigvee_{i=1}^{n} g^s(w_i,t) + h\ln n \quad (8)$$

According to (7), we have

$$t\ln 2 + \bigvee_{i=1}^{n}(x_i^s \wedge w_i) \le \bigvee_{i=1}^{n} g^s(w_i,t) \le \bigvee_{i=1}^{n}(x_i^s \wedge w_i) \quad (9)$$

The combination of (8) and (9) leads to

$$\bigvee_{i=1}^{n}(x_i^s \wedge w_i) + t\ln 2 \le G^s(w_i,t,h) \le \bigvee_{i=1}^{n}(x_i^s \wedge w_i) + h\ln n.$$

Notice that $O^s = \bigvee_{i=1}^{n}(x_i^s \wedge w_i)$. So this completes the proof of Theorem.

In place of the actual network output $O^s(w_i)$ by its smooth approximation (6) and using $\frac{\partial G^s(w_i,t,h)}{\partial w_i}$ to substitute $\frac{\partial O^s(w_i)}{\partial w_i}$, we can get the following all partial differentials of J with respect to w_i, and they have the following representations:

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial O^s}\frac{\partial O^s}{\partial w_i} = -\sum_{s=1}^{S}(T^s - O^s)\lambda^s(w_i,t,h)\mu^s(w_i,t) \quad (10)$$

where $\lambda^s(w_i,t,h) = \frac{\exp(g^s(w_i,t)/h)}{\sum_{i=1}^{n}\exp(g^s(w_i,t)/h)}$, $\mu^s(w_i,t) = \frac{\exp(w_i/t)}{\exp\left(x_i^s/t\right)+\exp(w_i/t)}$, $t<0$ and $h>0$.

Based on the above illustration, we can derive our smoothing algorithm for training *max-min* neural networks as follows:

Choose an arbitrary initial value W^0 and a constant learning rate $\eta > 0$. Then, the weight vector W is refined by the following learning iteration process

$$W^{k+1} = W^k + \Delta W^k = W^k - \eta\frac{\partial J(W^k)}{\partial W^k}, \quad k = 0,1,2,\cdots \quad (11)$$

where

$$W^k = (w_1^k, w_2^k, \cdots, w_n^k)$$

and

$$\frac{\partial J(W^k)}{\partial W(k)} = \left(\frac{\partial J(W^k)}{\partial w_1(k)}, \frac{\partial J(W^k)}{\partial w_2(k)}, \cdots, \frac{\partial J(W^k)}{\partial w_n(k)}\right).$$

Remark: The *max-min* neural network can be viewed as a fuzzy relational system $X \circ R = Y$. The training of the neural network is to identify the fuzzy relation R of a fuzzy relational equation based on the pairs (X,Y). Hence, our smoothing algorithm can also be used to solve fuzzy relational equations and be extended to multiple input and multiple output systems.

4. NUMERICAL EXAMPLES

In this section, to demonstrate the validity of our smoothing algorithm derived in Section 3 for training *max-min* neural networks, we will compare it with the other three algorithms introduced in Section 1 by the following two examples.

Example 1. The training sample pairs for this example are taken from the literature [16] and are shown in Tab. 1. In this example, the initial weight vector W^0 is chosen stochastically in [0,1] and the learning rate η is 0.05. The maximum number of iteration epoches and the error bound are set 1000

Table 1 : Training Sample Pairs for Example 1

s	X^s	T^s
1	(1.0 0.9 0.9 0.9 1.0)	(0.9)
2	(0.6 0.6 0.4 0.3 0.6)	(0.6)
3	(0.7 0.6 0.5 0.6 0.5)	(0.6)
4	(0.0 0.9 0.6 0.5 0.4)	(0.6)
5	(0.2 0.2 0.1 0.4 0.2)	(0.2)
6	(0.1 0.1 0.2 0.5 0.1)	(0.2)

and 1.0e-5, respectively. We set the parameters t and h values in our smoothing algorithm as $t = -0.3$ and $h = 0.3$. In this case, 10 trials are carried out for our smoothing algorithm and other three algorithms introduced in Section 1. The average errors and numbers of iteration epoches across the 10 trials are shown in Tab. 2. We see

Table 2 : Comparison of the results for Example 1

Learning algorithm	Average errors	Average numbers of iteration epoches
Our smoothing algorithm	9.8907e-06	547
Algorithm 1	0.0367	1000
Algorithm 2	9.9935e-06	819
Algorithm 3	0.0367	1000

from Tab. 2 that the performance of our smoothing algorithm is better than that of other three algorithms. We also illustrate as an example in Fig. 2 the convergence behavior of our smoothing algorithm in one of the 10 trials. We note that cost

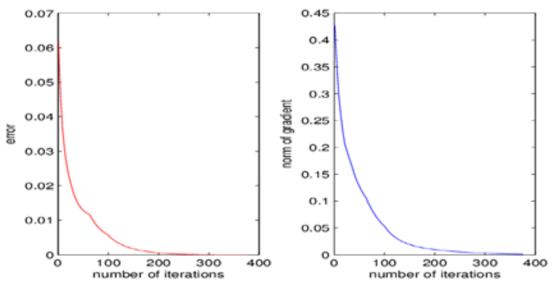

Figure 2: Error And Norm Of Gradient Of The Cost Function For Example 1

function $J(W)$ decreases monotonically and the norm of $\frac{\partial J(W)}{\partial W}$ tends to zero.

Example 2. The training sample pairs for this example are taken from [13] and are shown in Tab. 3. In this example, the initial weight vector W^0 is chosen stochastically in [0,1] and the learning rate η is 0.1. The maximum number of iteration epoches

Table 3 : Training Sample Pairs For Example 2

s	X^s	T^s
1	(0.20 0.40 0.43)	(0.27)
2	(0.10 0.40 0.85)	(0.30)
3	(0.20 0.95 0.30)	(0.59)
4	(0.20 1.00 0.80)	(0.61)
5	(1.00 0.70 0.20)	(0.75)
6	(1.00 0.70 0.65)	(0.80)
7	(1.00 0.40 0.43)	(0.88)
8	(0.80 0.30 0.70)	(0.77)

and the error bound are set 1000 and 0.005, respectively. We set the parameters t and h values in our smoothing algorithm as $t=-0.1$ and $h=0.1$. In this case, 10 trials are also carried out for our smoothing algorithm and other three algorithms. The average errors and numbers of iteration epoches and the times of reaching the error bound within 1000 epoches across the 10 trials are shown in Tab. 4. We also see from Tab. 4 that the performance of our smoothing algorithm is better

Table 4: Comparison Of The Results For Example 2

Learning algorithm	Average errors	Average numbers of iteration epoches	Times of reaching of iteration epochs the error bound
Our smoothing algorithm	0.0048	32	10
Algorithm 1	0.0317	724	3
Algorithm 2	0.0050	418	7
Algorithm 3	0.0317	724	3

than that of other three algorithms. Furthermore, we illustrate as an example in Fig. 3 the convergence behavior of our smoothing algorithm in one of the 10 trials. We also note that cost function $J(W)$ decreases monotonically and the norm of $\frac{\partial J(W)}{\partial W}$ tends to zero.

5. CONCLUSION

This paper has introduced a smoothing technique to approximate max-min functions, and subsequently applied it to construct a smoothing algorithm for training max-min neural networks. The algorithm can also be used to solve fuzzy relational equations. Specifically, we apply a smooth function to approximate max-min functions and use this smoothing technique twice, once to eliminate the inner min operator and once to eliminate the max

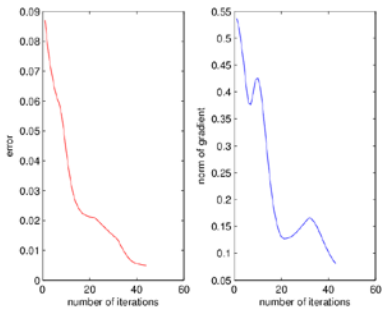

Figure 3: Error And Norm Of Gradient Of The Cost Function For Example 2

operator. In place of actual network output by its approximation function, we use all partial derivatives of the approximation function with respect to weight to substitute those of the actual network output. Then, the smoothing algorithm is constructed by the gradient descent method. Finally, two numerical examples are provided to show the effectiveness of our smoothing algorithm for training max-min neural networks.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (11171095) and College Students' Research Learning and Innovative Experiment Foundation of Hunan Province (cx1108)

REFERENCES:

[1] I. S. Baruch, R. Lopez, J. O. Guzman, J. M. Flores, "A fuzzy-neural multi-model for nonlinear systems identification and control", *Fuzzy Sets and Systems*, Vol. 159, 2008, pp. 2650-2667.

[2] H. Song, C. Miao, Z. Shen, Y. Miao, B. Lee, "A fuzzy neural network with fuzzy impact grades", *Neurocomputing*, Vol. 72, 2009, pp. 3098-3122.

[3] J. R. Castro, O. Castillo, P. Melin, A. Rodríguez-Díaz, "A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks", *Information Sciences*, Vol. 179, 2009, pp. 2175-2193.

[4] C. Juang, Y. Lin, C. Tu, "A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing", *Fuzzy Sets and Systems*, Vol. 161, 2010, pp. 2552-2568.

[5] A. Khajeh, H.Modarress, "Prediction of solubility of gases in polystyrene by Adaptive Neuro-Fuzzy Inference System and Radial Basis Function Neural Network", *Expert Systems with Applications*, Vol. 37, 2010, pp. 3070-3074.

[6] Y. Li, Z. Wu, "Fuzzy feature selection based on min-max learning rule and extension matrix", *Pattern Recognition*, Vol. 41, 2008, pp. 217-226.

[7] A. Quteishat, C. Lim, "A modified fuzzy min-max neural network with rule extraction and its application to fault detection and classification", *Applied Soft Computing*, Vol. 8, 2008, pp. 985-995.

[8] J. Park, T. Kim, T, Sugie, "Output feedback model predictive control for LPV systems based on quasi-min-max algorithm", *Automatica*, Vol.47, 2011, pp. 2052-2058.

[9] H. Dastkhan, N. Gharneh, H. Golmakani, "A linguistic-based portfolio selection model using weighted max–min operator and hybrid genetic algorithm", *Expert Systems with Applications*, Vol. 38, 2011, pp. 11735-11743.

[10] R. J. Marks II, S. Oh, P. Arabshahi, T. P. Caudell, J. J. Choi, B. G. Song, "Steepest descent adaptation of min-max fuzzy if-then rules", *In Proc. IJCNN*, Beijing, China, Vol. III, 1992, pp. 471-477.

[11] A. Nikov, S. Stoeva, "Quick fuzzy backpropagation algorithm", *Neural Networks*, Vol. 14, 2001, pp. 231-244.

[12] A. Blanco, M. Delgado and I. Requena, "Identification of fuzzy relational equations by fuzzy neural networks", *Fuzzy Sets and Systems*, Vol. 71, 1995, pp. 215-226.

[13] X. Zhang, C. Hang, "The min-max function differentiation and training of fuzzy neural networks", *IEEE Trans. on Neural Networks*, Vol.7, No.5, 1996, pp. 1139-1149.

[14] J. Peng, Z. Lin, "A non-interior continuation method for generalized linear complementarity problems", *Math. Program. Ser. A*, Vol. 86, 1999, pp. 533-563.

[15] X. Tong, L. Qi, F. Wu, H. Zhou, "A smoothing method for solving portfolio optimization with CVaR and applications in allocation of generation asset", *Applied Mathematics and Computation*, Vol. 216,2010, pp. 1723-1740.

[16] C. T. Yeh, "On the minimal solutions of max-min fuzzy relational equations", *Fuzzy Sets and Systems*, vol. 159, 2008, pp. 23-39.