An effective learning method for max-min neural networks

Loo-Nin Teow
Institute of Systems Science
National University of Singapore
Heng Mui Keng Terrace
Kent Ridge, Singapore 119597
email: loonin@iss.nus.sg

Kia-Fock Loe
Department of Information Systems
and Computer Science
National University of Singapore
Kent Ridge, Singapore 119260
email: loekf@iscs.nus.sg

Abstract

Max and min operations have interesting properties that facilitate the exchange of information between the symbolic and real-valued domains. As such, neural networks that employ max-min activation functions have been a subject of interest in recent years. Since max-min functions are not strictly differentiate, we propose a mathematically sound learning method based on using Fourier convergence analysis of side-derivatives to derive a gradient descent technique for max-min error functions. This method is applied to a "typical" fuzzy-neural network model employing max-rnin activation functions. We show how this network can be trained to perform function approximation; its performance was found to be better than that of a conventional feedforward neural network.

1 Introduction

Max and min operators are widely used in systems which performs fuzzy logic, multi-valued logic or other forms of uncertainty reasoning. Besides generalizing the binary OR and AND operators to the real-valued domain, max and min operators also have attractive properties such as commutativity, monotonicity and associativity. In addition, the max and min operators are the only real-valued logical operators that are continuous and idempotent [Klir and Folger, 1988]. From a theoretical point of view, they form a bridge between the symbolic and real-valued domains. Hence, they facilitate the encoding and extraction of symbolic knowledge in hybrid systems which combine logic and neural networks.

However, the problem with neural networks which employ max and min operators has always been with learning using gradient descent techniques such as backpropagation [Rumelhart *et* al., 198G]. This is because gradient descent requires the activation functions to be fully differentiable. Activation functions with max or min operations, unfortunately, do not satisfy this requirement. One of the goals of this paper is to analyze the calculus of max-min functions, especially with regard to the side-differentiability of such functions. In particular, we show how Fourier convergence analysis can be used to obtain the pseudo-derivatives. This in turn leads to a gradient descent technique for neural networks that employ max-min activation functions. Such an approach is obviously mathematically sound compared with *ad hoc* methods found in the literature.

2 The Calculus of Max-Min Functions

In this paper, a function $f(x)$ is said to be *side-differentiable* if at any point $x = a$, the following limits exist:

$$\frac{\partial f(x=a)}{\partial x^+} = \lim_{\Delta x \to 0^+} \left[\frac{f(a+\Delta x) - f(a)}{\Delta x}\right]$$

$$\frac{\partial f(x=a)}{\partial x^-} = \lim_{\Delta x \to 0^-} \left[\frac{f(a+\Delta x) - f(a)}{\Delta x}\right]$$

If, however, $f(a)$ fails to exist due to discontinuity, we can still define a weaker type of side-differentiability. A function $f(x)$ is said to be *quasi-differentiable* if at any point $x = a$, the following limits exist:

$$\frac{\partial f(x=a^+)}{\partial x^+} = \lim_{\Delta x \to 0^+} \left[\frac{f(a+\Delta x) - f(a+0)}{\Delta x}\right]$$

$$\frac{\partial f(x=a^-)}{\partial x^-} = \lim_{\Delta x \to 0^-} \left[\frac{f(a+\Delta x) - f(a-0)}{\Delta x}\right]$$

where

$$f(a+0) = \lim_{x\to a^+}[f(x)] \quad \text{and} \quad f(a-0) = \lim_{x \to a^-}[f(x)]$$

Side-differentiability implies quasi-differentiability, but not conversely. Higher-order derivatives may also exist. Interested readers can learn more about side-differentiability and quasi-differentiability in reference [Fulks, 1981]. In this paper, we shall only be concerned with functions which are side-differentiable in the first order and quasi-differentiable in the second order.

Let $\max_i^n(f_i(x))$ and $\min_i^n(f_i(x))$ be defined over the real number domain. Although both $\max_i^n(f_i(x))$ and $\min_i^n(f_i(x))$ may not be differentiable at some points, they can nonetheless be side-differentiable at all points, as shown by the following theorems, given here without proof.

Theorem 1 *Given $\overline{G}(x) = \max_i^n(f_i(x))$, such that $f_k(x)$ is continuous and side-differentiable $\forall k = 1 \ldots n$, then $\overline{G}(x)$ is also continuous, and the following side derivatives exist:*

$$\frac{\partial \overline{G}(x)}{\partial x^+} = \max_{i \in \overline{\Theta}(x)} \left(\frac{\partial f_i(x)}{\partial x^+} \right)$$

$$\frac{\partial \overline{G}(x)}{\partial x^-} = \min_{i \in \overline{\Theta}(x)} \left(\frac{\partial f_i(x)}{\partial x^-} \right)$$

$$\textit{where} \quad \overline{\Theta}(x) = \{ i \in \{1 \ldots n\} \mid f_i(x) = \overline{G}(x) \}$$

□

Theorem 2 *Given $\underline{G}(x) = \min_i^n(f_i(x))$, such that $f_k(x)$ is continuous and side-differentiable $\forall k = 1 \ldots n$, then $\underline{G}(x)$ is also continuous, and the following side derivatives exist:*

$$\frac{\partial \underline{G}(x)}{\partial x^+} = \min_{i \in \underline{\Theta}(x)} \left(\frac{\partial f_i(x)}{\partial x^+} \right)$$

$$\frac{\partial \underline{G}(x)}{\partial x^-} = \max_{i \in \underline{\Theta}(x)} \left(\frac{\partial f_i(x)}{\partial x^-} \right)$$

$$\textit{where} \quad \underline{\Theta}(x) = \{ i \in \{1 \ldots n\} \mid f_i(x) = \underline{G}(x) \}$$

□

Let $f'(a) = \frac{\partial f(x=a)}{\partial x}$ where $f(x)$ is a side-differentiable function. $f'(a)$ may fail to exist if $f(x)$ is not fully differentiable at $x = a$. However, the side-limits of $f'(a)$ are equivalent to the side derivatives of $f(a)$, and they exist since $f(x)$ is side-differentiable. Hence, if $\frac{\partial}{\partial x^+}\left(\frac{\partial f(x=a)}{\partial x^+}\right)$ and $\frac{\partial}{\partial x^-}\left(\frac{\partial f(x=a)}{\partial x^-}\right)$ exist, they are necessarily quasi derivatives of $f'(a)$. If these quasi derivatives exist for all x, then $f(x)$ is said to be quasi-differentiable in the second order.

Given a function G(x) which is side-differentiable everywhere, i.e. $\frac{\partial G(x)}{\partial x^+}$ and $\frac{\partial G(x)}{\partial x^-}$ exist for all x; we would like to define a pseudo-derivative $\frac{dG(x)}{dx^\pm}$ in terms of $\frac{\partial G(x)}{\partial x^+}$ and $\frac{\partial G(x)}{\partial x^-}$, as well as determine the conditions under which this is possible. To do this, consider the following theorem:

Theorem 3 *Let a function $F(x)$ be defined over an interval $lb \leq x \leq ub$ (and extended by periodicity), where lb and ub are the lower and upper bounds of the period respectively. If $F(x)$ is side-differentiable in the first order and quasi-differentiable in the second order, then at any point $lb < a < ub$, the Fourier series generated by $\frac{\partial F(x)}{\partial x}$ converges to the value*

$$\frac{1}{2}\left(\frac{\partial F(x=a)}{\partial x^+} + \frac{\partial F(x=a)}{\partial x^-} \right)$$

□

Hence, if we use a Fourier series to approximate the derivative of $F(x)$, then using the above theorem, we can assign the pseudo-derivative of $F(x)$ at each point as the value of convergence at that point, i.e.

$$\frac{dF(x=a)}{dx^\pm} = \frac{1}{2}\left(\frac{\partial F(x=a)}{\partial x^+} + \frac{\partial F(x=a)}{\partial x^-} \right)$$

However, note that this theorem does not apply at the end-points lb and ub. It is also applicable only for periodic functions. To overcome these restrictions, we use the following theorem:

Theorem 4 *Let $F(x)$ and $G(x)$ be functions that are side-differentiable in the first order and quasi-differentiable in the second order. Suppose $F(x)$ and $G(x)$ coincide over the interval $lb \leq x \leq ub$, such that $\forall x \in [lb, ub]$,*

$$F(x) = G(x)$$

$$\frac{\partial F(x)}{\partial x^+} = \frac{\partial G(x)}{\partial x^+}$$

$$\frac{\partial F(x)}{\partial x^-} = \frac{\partial G(x)}{\partial x^-}$$

Suppose further that $F(x)$ is periodic over the interval $lb' \leq x \leq ub'$, where $lb' < lb$ and $ub' > ub$. Then, $\forall x \in [lb, ub]$,

$$\frac{dG(x)}{dx^\pm} = \frac{1}{2}\left(\frac{\partial G(x)}{\partial x^+} + \frac{\partial G(x)}{\partial x^-} \right) \qquad (1)$$

□

If the conditions given in Theorem 4 are satisfied, then we can obtain the pseudo-derivative of $G(x)$ as given in Equation 1. This formula is consistent with the fact that when $G(x)$ is fully differentiable at $x = a$ (i.e. $\frac{\partial G(x=a)}{\partial x}$ exists), then $\frac{\partial G(x=a)}{\partial x} = \frac{\partial G(x=a)}{\partial x^+} = \frac{\partial G(x=a)}{\partial x^-}$, in which case $\frac{dG(x=a)}{dx^\pm} = \frac{\partial G(x=a)}{\partial x}$.

2.1 Other gradient descent techniques

In the fuzzy-neural research literature [Buckley and Hayashi, 1994; Gupta and Rao, 1994; Ishibuchi *et al.*, 1993; Keller *et al.*, 1992; Mitra and Pal, 1994; Pedrycz, 1993; Simpson, 1992; Simpson, 1993], many *ad hoc* gradient descent techniques for max-min functions have been proposed. Some suggested using simple intuition-based heuristics. Yet others side-stepped the differentiability problem by replacing a max-min function with a differentiable one, for example by replacing the max and min operators with sum and product operators respectively.

A more interesting approach would be to use a parameterized function which limits to the max or min operator. One such function is the *softmax* operation commonly used in winner-take-all neural architectures. The softmax operation is defined as follows:

$$\text{softmax}_i^n(f_i(x)) = \sum_i w_i \cdot f_i(x)$$

$$\text{where} \quad w_i = \frac{\exp[\psi \cdot f_i(x)]}{\sum_j \exp[\psi \cdot f_j(x)]}$$

ψ is the scaling parameter. When $\psi = 0$, the result is just the mean of all the constituent functions. When $\psi \to \infty$, softmax approaches the maximum of its constituent functions, while its derivative approaches the mean of the derivatives of its constituent functions. Based on this, we can have the following pseudo-derivative for the max operation:

$$\frac{d\max_i^n(f_i(x))}{dx} = \frac{1}{\mathcal{N}(\overline{\Theta}(x))} \sum_{i \in \overline{\Theta}(x)} \frac{df_i(x)}{dx}$$

$$\text{where} \quad \overline{\Theta}(x) = \left\{ i \in \{1 \ldots n\} \,\middle|\, f_i(x) = \max_i^n(f_i(x)) \right\}$$

and $\mathcal{N}(S)$ denotes the number of elements in set S.

There is, however, a problem with differentiating nested max-min functions using this formulation. For example, given

$$\mathcal{F}(x) = \max[A(x), \max[B(x), C(x)]]$$
$$\mathcal{G}(x) = \max[\max[A(x), B(x)], C(x)]$$

where A, B and C are fully differentiable functions defined over the real number domain. Obviously, $\forall x$, $\mathcal{F}(x) = \mathcal{G}(x)$. However, when $A(x_0) = B(x_0) = C(x_0)$, using the softmax-derived pseudo-derivative would result in

$$\mathcal{F}'(x_0) = \tfrac{A'(x_0)}{2} + \tfrac{B'(x_0)}{4} + \tfrac{C'(x_0)}{4}$$
$$\mathcal{G}'(x_0) = \tfrac{A'(x_0)}{4} + \tfrac{B'(x_0)}{4} + \tfrac{C'(x_0)}{2}$$

which is clearly a contradiction since $\mathcal{F}'(x_0) \neq \mathcal{G}'(x_0)$ when $A'(x_0)$, $B'(x_0)$ and $C'(x_0)$ are not equal to one another. This shows that in the limiting case, the differentiation of softmax functions is not associative.

In fact, this inconsistency arises whenever pseudo-derivatives of max and min operations are used directly when "differentiating" a nested max-rnin function.

The method proposed in this paper, on the other hand, does not have this problem. In our method, each of the side-derivatives of the *entire* function must be evaluated *before* they are combined. Hence, we apply Theorem 1 and Theorem 4 to give

$$\tfrac{d}{dx}[\mathcal{F}(x_0)] = \tfrac{d}{dx}[\mathcal{G}(x_0)]$$
$$= \tfrac{1}{2}(\max[A'(x_0), B'(x_0), C'(x_0)] + \min[A'(x_0), B'(x_0), C'(x_0)])$$

3 A Fuzzy-Neural Network Model

The synthesis of fuzzy logic and neural networks has been a popular theme in research in the past decade [Buckley and Hayashi, 1994; Gupta and Rao, 1994; Ishibuchi *et* a/., 1993; Keller *et* a/., 1992; Mitra and Pal, 1994; Pedrycz, 1993; Simpson, 1992; Simpson, 1993]. This is not surprising considering that neural networks and fuzzy logic complement each other. Neural networks are well known for their learning capabilities, which allow them to model accurately almost any input-output relationship. Fuzzy logic, on the other hand, facilitates the encoding of experts' knowledge in linguistic terms and inferencing from such knowledge using mathematical techniques. Fuzzy-neural networks, by combining these two technologies, can, among other things, allow the fine-tuning of of experts' knowledge as well as a more natural interpretation of the knowledge learnt.

Max and min are the standard logical operations used in fuzzy set theory [Klir and Folger, 1988]. In addition, they have many attractive properties as described in the introduction of this paper. Hence, we employ max-min activation functions in a "typical" fuzzy-neural network architecture commonly found in the fuzzy-neural research literature. Since max-min functions are non-differentiable, we show how such a network can be trained with gradient descent based on using Fourier convergence analysis.

3.1 Description of the model

The fuzzy-neural network model we use has five layers: one input, one output and three hidden (Figure 1). Each unit in the input layer corresponds to an input variable, and is connected to several antecedent fuzzy set units in the first hidden layer. The units in the antecedent layer can be divided into sub-groups, each corresponding to a fuzzy variable. From every sub-group, only one fuzzy set unit may be connected to a rule unit in the second hidden layer, depending on the fuzzy rule associated with that rule unit. The rules comprises of all possible combinations of antecedent fuzzy sets; hence, if there are 3 input variables with each having 2 fuzzy sets, then there will be a total of 2x2x2=8 rules. Each rule unit is connected to the consequent fuzzy set units in the third hidden layer. Like the units in the first hidden layer, consequent fuzzy set units are also grouped in a similar fashion. Each sub-group has connections to the output unit corresponding to its fuzzy output variable.

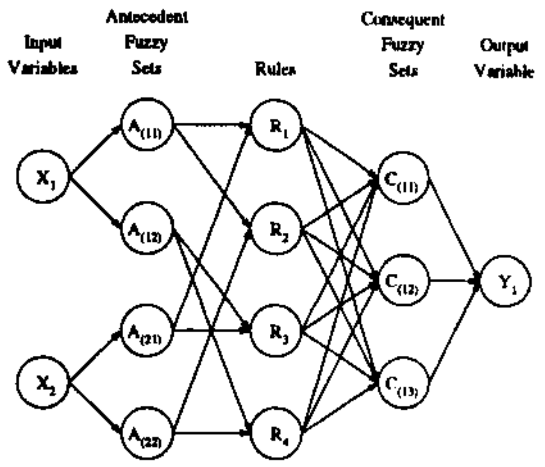

Figure 1: An example of a fuzzy-neural network.

The input units pass the input values to the various antecedent fuzzy set units. The activation value of each antecedent fuzzy set unit is simply the membership degree of the input value in the corresponding fuzzy set. If we define each fuzzy membership function as a Gaussian function with center μ and radius σ, then we have:

$$A_{(mn)} = \exp\left(-\frac{1}{2}\cdot\left(\frac{X_m - \mu_{(mn)}}{\sigma_{(mn)}}\right)^2\right) \tag{2}$$

Each rule unit computes the fuzzy conjunction of its corresponding antecedents by taking the minimum of their fuzzy membership degrees:

$$R_k = \min_{(mn)\in\Upsilon(k)}(A_{(mn)}) \tag{3}$$

where $\Upsilon(k)$ is the set of indices to antecedents in a rule.

Each consequent fuzzy set unit computes the *weighted* fuzzy disjunction of the rule activations as follows:

$$C_{(ij)} = \max_k(W_{(ij)k}\cdot R_k) \tag{4}$$

where $W_{(ij)k}$ is a weight value specifying the degree to which a rule contributes to a consequent. It is constrained to the range [0.0,1.0] as required by fuzzy set theory.

Finally, each output unit performs defuzzification as follows:

$$Y_i = \frac{\sum_j (C_{(ij)}\cdot V_{ij})}{\sum_j C_{(ij)}} \tag{5}$$

where $V_{(ij)}$ is the output value associated with a single-element consequent fuzzy set.

3.2 Training by Gradient Descent

Given an input-target pattern, the error E is defined as

$$E = \sum_i (T_i - Y_i)^2 \tag{6}$$

where Y_i is the i^th output value, and T_i is the corresponding target value.

The consequent values V, the weights W, and the input fuzzy membership functions (parameterized by μ and σ) can all be modified using gradient descent. Each parameter ϕ (which can be either V_{pq}, $W_{(pq)r}$, $\mu_{(st)}$ or $\sigma_{(st)}$) may be changed by the following increment:

$$\Delta^\tau\phi = -\alpha_\phi\cdot\frac{dE}{d\phi} + \beta_\phi\cdot\Delta^{\tau-1}\phi \tag{7}$$

where τ is the training iteration; α_ϕ is the learning rate; β_ϕ is the momentum term; and $\Delta^0\phi = 0$.

Since the equation containing V (Equation 5) does not contain any max or min operations and is fully differentiable, $\Delta^\tau V_{pq}$ can be derived using normal differentiation:

$$\frac{dE}{dV_{pq}} = -2(T_p - Y_p)\cdot\frac{C_{(pq)}}{\sum_j C_{(pj)}} \tag{8}$$

For the rest of the parameters W, μ and σ, we need to apply Theorem 4. In the case of W, the derivation is as follows:

$$\begin{aligned}\frac{dE}{dW^\pm_{(pq)r}} &= \frac{1}{2}\left(\frac{\partial E}{\partial W^+_{(pq)r}} + \frac{\partial E}{\partial W^-_{(pq)r}}\right)\\ &= \mathcal{E}_{pq}\cdot\left(\frac{\partial C_{(pq)}}{\partial W^+_{(pq)r}} + \frac{\partial C_{(pq)}}{\partial W^-_{(pq)r}}\right)\end{aligned}$$

$$\text{where}\quad \mathcal{E}_{pq} = -(T_p - Y_p)\cdot\left(\frac{V_{pq} - Y_p}{\sum_n C_{(pn)}}\right)$$

All weight changes are adjusted such that each weight is clipped to the range [0.0,1.0], for reasons explained in the earlier subsection.

Applying Theorem 1 to Equation 4, we have

$$\frac{\partial C_{(pq)}}{\partial W^+_{(pq)r}} = \max_{k\in\Psi(pq)}(\delta_{rk}\cdot R_k) \tag{9}$$

$$\frac{\partial C_{(pq)}}{\partial W^-_{(pq)r}} = \min_{k\in\Psi(pq)}(\delta_{rk}\cdot R_k) \tag{10}$$

where

$$\Psi(pq) = \{k \mid W_{(pq)k}\cdot R_k = C_{(pq)}\} \tag{11}$$

and

$$\delta_{rk} = \begin{cases}1 & \text{if } r = k\\ 0 & \text{otherwise}\end{cases} \tag{12}$$

For μ, we have

$$\frac{dE}{d\mu^\pm_{(st)}} = \sum_i\sum_j\left[\mathcal{E}_{ij}\cdot\left(\frac{\partial C_{(ij)}}{\partial\mu^+_{(st)}} + \frac{\partial C_{(ij)}}{\partial\mu^-_{(st)}}\right)\right] \tag{13}$$

Applying Theorem 1 to Equation 4,

$$\frac{\partial C_{(ij)}}{\partial \mu^{+}_{(st)}} = \max_{k \in \Psi(ij)} \left(W_{(ij)k} \cdot \frac{\partial R_k}{\partial \mu^{+}_{(st)}} \right) \tag{14}$$

$$\frac{\partial C_{(ij)}}{\partial \mu^{-}_{(st)}} = \min_{k \in \Psi(ij)} \left(W_{(ij)k} \cdot \frac{\partial R_k}{\partial \mu^{-}_{(st)}} \right) \tag{15}$$

Applying Theorem 2 to Equation 3,

$$\frac{\partial R_k}{\partial \mu^{+}_{(st)}} = \min_{(mn) \in \Upsilon(k) \cap \Phi(k)} \left(\frac{\partial A_{(mn)}}{\partial \mu_{(st)}} \right) \tag{16}$$

$$\frac{\partial R_k}{\partial \mu^{-}_{(st)}} = \max_{(mn) \in \Upsilon(k) \cap \Phi(k)} \left(\frac{\partial A_{(mn)}}{\partial \mu_{(st)}} \right) \tag{17}$$

where Υ is as defined in Equation 3 and

$$\Phi(k) = \{(mn) \mid A_{(mn)} = R_k\} \tag{18}$$

Obviously, if $(mn) \neq (st)$, then $\frac{\partial A_{(mn)}}{\partial \mu_{(st)}} = 0$. Using normal differentiation on Equation 2, we get

$$\frac{\partial A_{(st)}}{\partial \mu_{(st)}} = A_{(st)} \cdot \frac{X_s - \mu_{(st)}}{\sigma^2_{(st)}} \tag{19}$$

Likewise for σ.

3.3 Experiments and Results

The fuzzy-neural network is trained and tested on the task of function approximation. Specifically, the function to be approximated is a highly non-linear 3-input *sinc* function:

$$T = sinc(x, y, z) = \frac{\sin(x)}{x} \times \frac{\sin(y)}{y} \times \frac{\sin(z)}{z} \tag{20}$$

Description of Experiments

Input values for both training and testing data are sampled uniformly from the ranges [-10,10] × [-10,10] × [-10,10] for input variables x, y and z respectively. Their target values are computed using Equation 20 and then scaled to the range [0.05,0.95]. The training and testing data sets each contains 100 input-target patterns.

The fuzzy-neural network in this experiment has 3 input units, each of which has 5 fuzzy sets, making a total of 15 antecedent fuzzy set units. There are altogether 5x5x5=125 rules to cover all the antecedent combinations. There is one output unit with 5 consequent fuzzy sets. The total number of modifiable network parameters, which includes the input fuzzy membership functions, the rule-to-consequent weights and the consequent values, is 660. The fuzzy membership functions for each input variable is initialized to $\{\mu\} = \{-10, -5, 0, 5, 10\}$ and $\{\sigma\} = \{1, 1, 1, 1, 1\}$. The consequent values are initialized to $\{V\} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The rule-to-consequent weights are initialized with random values from a uniform distribution over the range [0.45,0.55]. During learning, each σ value is constrained to lie below 1.0, because it was found that when the Gaussian radius of a fuzzy set is too wide, it practically becomes useless.

For comparison, we also train a conventional feedforward sigmoidal neural network using standard backpropagation. The network has 3 inputs units and 1 output unit. In order that we may make a fairer comparison, the number of hidden units was chosen such that the number of modifiable network parameters, which includes the biases and connection weights, is as close to that of the fuzzy-neural network as possible. As such, there are 132 hidden units, making a total of G61 modifiable network parameters. All weights and biases are initialized with random values from a uniform distribution over the range [-0.01,0.01].

In a training session, presentation of an input-target pattern constitutes an iteration, while presentation of the entire training set constitutes an epoch. Weights update is performed via pattern mode, i.e. after the presentation of each pattern, the weights are changed according the equations given in the previous sub-section.

All learning rates and momentum terms for both the fuzzy-neural network and the sigmoidal network are 0.3 and 0.1 respectively. Both networks are trained for 200 epochs over 10 trials.

Finally, in each trial, each trained network is tested for generalization on the testing data.

Network model	M.S.E.$_{train}$	M.S.E.$_{test}$	size
Fuzzy-neural	0.10×10^{-3}	1.22×10^{-3}	660
Sigmoidal	3.52×10^{-3}	3.23×10^{-3}	661

Table 1: Comparison between the fuzzy-neural network and the sigmoidal neural network in terms of the Mean Squared Error (M.S.E.), averaged over 10 trials, on both the training data and the testing data after being trained. The size *of* each network, in terms of the number of modifiable parameters, is also given.

Results and Discussion

We evaluate each network's performance using the Mean Squared Error (M.S.E.), i.e. the total squared error over all the patterns in a data set divided by the number of patterns.

Figure 2 show the M.S.E. curves on the training data for the fuzzy-neural network and the sigmoidal neural network, averaged over 10 trials, with 200 epochs in each trial. The fuzzy-neural network is able to reach a much lower mean squared error in a shorter time than the sigmoidal neural network.

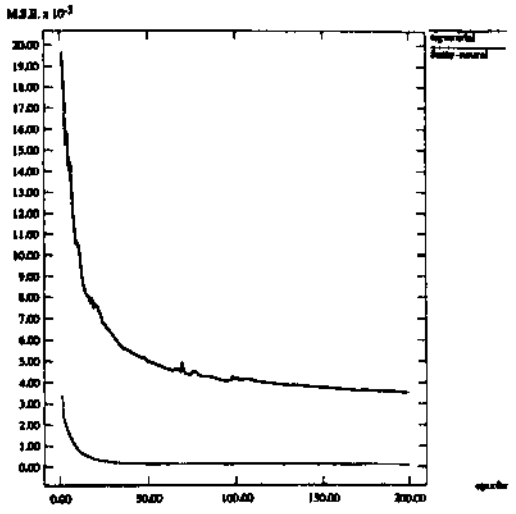

Figure 2: Mean Squared Error (M.S.E.) curves on the training data for the fuzzy-neural network and the sigmoidal neural network, averaged over 10 trials, with 200 epochs in each trial.

Table 3.3 gives the comparison between the fuzzy-neural network and the sigmoidal neural network in terms of the Mean Squared Error (M.S.E.), averaged over 10 trials, on both the training data and the testing data after being trained. As expected, the fuzzy-neural network has a much lower mean squared error on the training data than the sigmoidal neural network. Its generalization power, as seen by the performances on the testing data, is also better than that of the sigmoidal neural network.

Both Figure 2 and Table 3.3 demonstrate how the fuzzy-neural network can effectively model a highly non-linear function as compared to a sigmoidal neural network with nearly the same number of network parameters. The authors did not make an exhaustive search of all possible network architectures and learning parameters in either model, so this comparison cannot be a universal one.

4 Conclusions

A learning method that utilizes gradient descent based on using Fourier convergence analysis for max-min functions was presented. It has been applied to effectively train a feedforward fuzzy-neural network model. This model employs max and min as its logical operations and Gaussian functions as its input fuzzy sets. We have shown how the network can be trained to approximate a highly non-linear function. It learns and generalizes better than a conventional feedforward neural network.

In conclusion, we have shown that our proposed gradient descent technique does allow max-min neural networks to learn effectively. Our approach should be extensible to other neural networks that have non-differentiable activations functions. This remains a topic of further research.

References

[Buckley and Hayashi, 1994] Buckley J.J. and Hayashi Y. (1994). "Fuzzy neural networks: A survey". *Fuzzy Sets and Systems* 66, pp. 1-13.

[Fulks, 1981] Fulks W. (1981). *Advanced Calculus - An Introduction to Analysis,* Third Edition, Wiley Trans-Edition.

[Gupta and Rao, 1994] Gupta M.M. and Rao D.H. (1994). "On the principles of fuzzy neural networks". *Fuzzy Sets and Systems* 61, pp. 1-18.

[Ishibuchi *et al,* 1993] Ishibuchi H., Fujioka R., and Tanaka H. (1993). "Neural Networks That Learn from Fuzzy If-Then Rules". *IEEE Transactions on Fuzzy Systems,* Vol.1, No.2, pp.85-97.

[Keller *et ai,* 1992] Keller J.M., Yager R.R., and Tahani H. (1992). "Neural network implementation of fuzzy logic". *Fuzzy Sets and Systems* 45, pp.1-12.

[Klir and Folger, 1988] Klir G.J. and Folger T.A. (1988). *Fuzzy Sets, Uncertainty, and Information,* Prentice-Hall International Editions.

[Mitra and Pal, 1994] Mitra S. and Pal S.K. (1994). "Logical Operation Based Fuzzy MLP for Classification and Rule Generation". *Neural Networks,* Vol.7, No.2, pp.353-373.

[Pedrycz, 1993] Pedrycz W. (1993). "Fuzzy neural networks and neurocomputations". *Fuzzy Sets and Systems* 56, pp. 1-28.

[Rumelhart *et ai,* 1986] Rumelhart D.E., Hinton G.E., and Williams R.J. (1986). "Learning internal representations by error propagation". *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume I: Foundations,* editors Rumelhart D.E., McClelland J.L., et al., Chap.8, pp.318-362. MIT Press, Cambridge.

[Simpson, 1992] Simpson P.K. (1992) "Fuzzy Min_Max Neural Networks - Part 1: Classifications". *IEEE Transactions on Neural Networks,* Vol.3, No.5, pp.776-786.

[Simpson, 1993] Simpson P.K. (1993) "Fuzzy Min_Max Neural Networks - Part 2: Clustering". *IEEE Transactions on Neural Networks,* Vol.1, No.l, pp.32-45.