2000 IEEE International Conference on Systems, Man & Cybernetics, October 8-11, 2000, Sheraton Music City Hotel, Nashville, Tennessee, USA

On the non-Differentiability of Fuzzy Logic Systems

Paolo Dadone and Hugh F. VanLandingham
The Bradley Department of Electrical and Computer Engineering, Virginia Tech
Blacksburg, Virginia, 24061-0111

Abstract

Tuning the parameters of fuzzy logic systems has become an important issue for their efficient development and utilization. Many techniques, mainly based on the application of gradient descent, have been applied to this task. The class of fuzzy logic systems using piecewise-linear membership functions (e.g., triangular or trapezoidal) and/or minimum or maximum operators possesses an error function that is non-differentiable (i.e., at any point in the search space) with respect to some of its parameters. Therefore the gradient does not always exist, and thus any convergence proof (even if only at a zero gradient point) does not hold any more. This paper discusses the problem and shows some of the issues it raises.

1 Introduction

Quite some attention of the early nineties research on fuzzy logic systems (FLSs) was focused on parametric design. Triangular or trapezoidal membership functions, along with minimum *t-norm*, are common choices in the use of FLSs. Some of the work on tuning FLSs has been geared towards adjusting the parameters in a FLS containing triangular membership functions and/or min—max operators. One of the first approaches to tuning antecedent and consequent parameters of a fuzzy logic system through gradient descent is the one of Nomura et al. [10,11]. In this approach symmetric triangular membership functions and constant consequents are used. The authors tune consequent constants and antecedent membership functions parameters (center and width). The non-differentiability of the error function with respect to the triangle center at a finite number of points is briefly noted in [11] (it is also non-differentiable with respect to the widths). The problem is faced pragmatically by zeroing the corresponding parameter update whenever non-differentiability is encountered. Many other authors used this approach neglecting the non-differentiability problem.

Miyata et al. [9] and Ohki et al. [12] extended the work in [10,11] to the case of some piecewise linear membership functions defined by five parameters each. In this case there are even more points of discontinuity of the derivative that they do not note while applying gradient descent for the minimization of the mean square approximation error. Bersini and Gorrini [2] extend the approach in [10] for adaptive fuzzy control. Guely and Siarry [6] extend the approach in [10,11] to asymmetric triangular membership functions as well as product and minimum *t-norms*. Divergence of the algorithm is experienced for a small number of rules. Shi and Mizumoto [14] develop a gradient descent approach for symmetric triangular membership functions that is very similar to the one in [2]. Wang and Acar [18] use the approach in [10,11] for fuzzy system identification of an inverted pendulum. Godjevac [5] uses the same approach for tuning a controller for mobile robot obstacle avoidance. Jang [7] introduces an adaptive-network-based fuzzy inference system (ANFIS) and shows how back-propagation learning algorithms can be derived from the corresponding network structure. Bell-shaped membership functions are used, even though it is mentioned that any other membership function, such as piecewise linear, could be tuned as well. Katayama et al. [8] propose a gradient-based constrained optimization approach for tuning the centers of triangular membership functions. Glorennec [4] recognizes that triangular membership functions are non-differentiable and thus gradient descent cannot be used. Arabshahi et al. [1,15] present an approach to supervised learning of FLSs with gaussian membership functions, and min-max operators with the comfort that min and max are continuous and differentiable functions. Pedrycz [13] observes the problem existing with non-differentiability of the min and max operators but does not consider it a real problem in the practical operation of the learning algorithm. His concern is that these two-valued derivatives might make the algorithm terminate at a sub-optimal point. Moreover, he observes that using opportune parameterized *t-norms* could eliminate the non-differentiability problem. Teow and Loe [17] propose a Fourier series analysis to estimate the derivative at non-differentiable points when max—min operators are used. This approach fully addresses the problem, even though it proposes a solution to estimate a derivative that in reality does not exist.

This paper discusses the application of gradient descent to supervised learning of fuzzy logic systems using piecewise linear membership functions and/or max—min operators. The next section shows how this generates an optimization problem that is not only nonlinear, but also non-differentiable. In the third section a simple function approximation problem will be introduced, and in the fourth section some of the issues involved with this type of problem will be showed through some numerical results. The final section offers some concluding remarks.

2 Non-differentiability

Supervised learning of FLSs is in general a nonlinear programming problem in which it is desired to find the values of some FLS parameters that minimize an error function, generally quadratic. In the following, systems with singleton fuzzification and center of area (COA) defuzzification will be considered, since those are common and simple choices. The parameters that are generally tuned are the ones defining antecedent and consequent membership functions. The objective function is linear in the consequent parameters, thus originating a relatively simple optimization problem that can be solved by well-known and robust techniques [7,10]. Most of the problems arise in tuning the parameters defining the antecedent membership functions (i.e., typically width and center). Indeed, that is a nonlinear programming problem and under some conditions also a non-differentiable one. As noted in the introduction, a common approach is to use gradient descent for antecedent parameters optimization. This entails the calculation of the gradient of the objective function with respect to the parameters to be adjusted, and thus the derivatives of the output with respect to the adjustable parameters are needed. Whenever piecewise linear membership functions and/or min—max operators are used, the derivative does not exist anymore at all points of the search space. In these cases it is easily seen that the output of the FLS is not differentiable (everywhere, with respect to the adjustable parameters). There are indeed some points where right and left derivative exist but assume different values. In this case a gradient descent approach (as well as any gradient-based method) cannot be applied. For this reason the application of a *false* gradient descent (i.e., a gradient descent where the derivatives are arbitrarily defined at the points where they do not exist) as presented in the literature will not offer any guarantee of convergence (even to a zero gradient point). Indeed, using such a *false* gradient, the direction of search ceases to necessarily be a descent direction. The (negative) gradient direction is necessarily an improving direction, i.e., a direction along which the objective function initially decreases. Thus, if a small enough step-size (or a line search) is used, a decrease in objective function must be observed. In the non-differentiable case, some non-improving steps might be taken during the progress of the algorithm, as a result of not following a (true) gradient direction. Reducing the step-size will not help achieving an improvement. Using a line search would only make things worse since the algorithm would get trapped in a non-differentiable point. If the effects of these non-improving steps compound, the algorithm might end up diverging as was shown in [3].

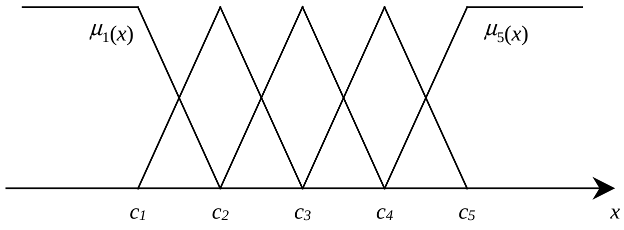

Figure 1. Antecedent membership functions

From a pragmatic point of view, the non-differentiable nature of the problem might not be a concern since the probability of hitting points of non-differentiability is zero. On the other hand, given some training data, these constitute a grid in the input space. The membership functions, for example, move along this grid during the tuning, and might fall onto one of the points or at least arbitrarily close to it. Therefore, the presence of these "glitches" might make the algorithm converge in a slower fashion, when not generating divergent behavior.

The literature reviewed by the authors has little or no discussion about this topic. Indeed, the many authors that do use gradient methods generally do not make any mention of the non-differentiability, and directly derive and apply gradient update equations. The corresponding algorithms are not truly gradient descent algorithms. It seems imperative to recognize and study the non-differentiability problem in order to acknowledge it, eventually establish some convergence proofs, and at least study its effects on algorithmic performance. This is especially important for on-line applications of adaptive FLSs. The next section will introduce a simple example to illustrate the problem.

3 A Function Approximation Example

In this example we consider a single input single output (SISO) Takagi-Sugeno (TS) [16] FLS with triangular antecedent membership functions, constant outputs (i.e., local models) and product inference. The l-th rule for the given FLS will be: $R^{(l)}$: IF x is A_l, THEN $y = p_l$. Let us indicate with R the number of rules, and thus also the number of membership functions – $\mu_i(x)$ – on the (single) input. The membership functions are triangular with a 0.5 level of overlap. Figure 1 shows a graphical explanation of this type of membership functions. Introducing the asymmetrical triangular function:

$$tr(a,b,c,x)=\begin{cases}\dfrac{x-a}{b-a} & if\ a\le x\le b\\ 0 & if\ x<a\vee x>c\\ \dfrac{x-c}{b-c} & if\ b<x\le c\end{cases} \tag{1}$$

the membership functions for the FLS will be:

$$\begin{gathered}\mu_i(x)=tr(c_{i-1},c_i,c_{i+1},x)\quad i=2,3,...,R-1\\ \mu_1(x)=tr(-\infty,c_1,c_2,x)\ \mu_R(x)=tr(c_{R-1},c_R,+\infty,x)\end{gathered} \tag{2}$$

With this particular setting of membership functions:

$$\sum_{i=1}^{R}\mu_i(x)=1\quad \forall x\in X \tag{3}$$

The output of the FLS will thus be:

$$y = \sum_{i=1}^{R} \mu_i(x) p_i \tag{4}$$

Every membership function is defined by its center (and the two adjacent centers), thus this (fuzzy) model will be defined by R parameters for the antecedent and R parameters for the consequent. Using a pattern-by-pattern approach, given S data pairs (x_j, y_{dj}) (j = 1,2, …, S) we wish to minimize the instantaneous errors:

$$E_j(\boldsymbol{c}, \boldsymbol{p}) = \frac{1}{2} e_j^2 \qquad e_j = y(x_j, \boldsymbol{c}, \boldsymbol{p}) - y_{dj} \tag{5}$$

For a generic parameter w the update rule based on gradient descent will be:

$$w^{new} = w^{old} - \eta e_j \left.\frac{\partial y}{\partial w}\right|_{(x_j, w^{old})} \tag{6}$$

where η is the (constant) learning rate (i.e., step size). When w is one of the consequent parameters $\boldsymbol{p}$ we obtain:

$$\frac{\partial y}{\partial p_i} = \mu_i(x) \tag{7}$$

Using Eq. (4) and remembering that every center (c_i) contributes to three membership functions we obtain:

$$\begin{gathered} \frac{\partial y}{\partial c_i} = \sum_{j=-1}^{1} p_{i+j} \frac{\partial \mu_{i+j}(x)}{\partial c_i} \qquad i = 2,3,...,R-1 \\ \frac{\partial y}{\partial c_1} = \sum_{j=0}^{1} p_{j+1} \frac{\partial \mu_{j+1}(x)}{\partial c_1}, \quad \frac{\partial y}{\partial c_R} = \sum_{j=-1}^{0} p_{R+j} \frac{\partial \mu_{R+j}(x)}{\partial c_R} \end{gathered} \tag{8}$$

We finally only need the derivatives of the membership functions with respect to their three parameters. By straightforward derivation of Eq. (1) we obtain:

$$\begin{gathered} \frac{\partial tr(a,b,c,x)}{\partial a} = \begin{cases} 0 & if \;\; x < a \vee x \geq b \\ \dfrac{x-b}{(b-a)^2} & if \;\; a < x < b \end{cases} \\ \frac{\partial tr(a,b,c,x)}{\partial b} = \begin{cases} \dfrac{a-x}{(b-a)^2} & if \;\; a \leq x < b \\ 0 & if \;\; x < a \vee x \geq c \\ \dfrac{c-x}{(b-c)^2} & if \;\; b < x < c \end{cases} \\ \frac{\partial tr(a,b,c,x)}{\partial c} = \begin{cases} 0 & if \;\; x \leq b \vee x > c \\ \dfrac{x-b}{(b-c)^2} & if \;\; b \leq x < c \end{cases} \end{gathered} \tag{9}$$

Equations (9) are still not operational for the application of the gradient descent algorithm. Indeed, in this case we cannot apply the gradient descent algorithm because the membership functions are not differentiable with respect to the adjustable parameters. In each of the Eqs. (9) the derivative is not defined for $x = a$, $x = b$, and $x = c$ respectively. To be able to apply a *false* gradient descent we need to define the value of the derivative at these undefined points (as in the approaches that apply gradient descent to non-differentiable FLSs). In doing this we are not really formally defining a derivative since the derivative does not exist. The choice for the value of the "derivative" at these undefined points is completely arbitrary. It seems reasonable to use either the right or the left derivative with a preference for zero values. With this choice we can rewrite the "derivatives" (loosely meant) as:

$$\begin{gathered} \frac{\partial tr(a,b,c,x)}{\partial a} = \begin{cases} 0 & if \;\; x \leq a \vee x \geq b \\ \dfrac{x-b}{(b-a)^2} & if \;\; a < x < b \end{cases} \\ \frac{\partial tr(a,b,c,x)}{\partial b} = \begin{cases} \dfrac{a-x}{(b-a)^2} & if \;\; a \leq x < b \\ 0 & if \;\; x < a \vee x \geq c \\ \dfrac{c-x}{(b-c)^2} & if \;\; b \leq x < c \end{cases} \\ \frac{\partial tr(a,b,c,x)}{\partial c} = \begin{cases} 0 & if \;\; x \leq b \vee x \geq c \\ \dfrac{x-b}{(b-c)^2} & if \;\; b \leq x < c \end{cases} \end{gathered} \tag{10}$$

4 Results

We choose R = 5, thus having a number of 10 adjustable parameters. It is desired to approximate the function $y = x^2$ in [-1,1]. Let us now suppose that 16 training data (S = 16) are randomly generated (x_i is random in [-1,1] and $y_{di} = x_i^2$) at every epoch, as it could be the case in many applications such as, for example, online ones. The initial values of the parameters $\boldsymbol{p}$ and $\boldsymbol{c}$ are chosen randomly in [0,1] and [-1,1], respectively. A constant learning rate η = 0.01 was used along with a maximum number of 300 epochs. A first run of the algorithm yields the convergence reported in Fig. 2 with the dashed line (b). The mean square error always improves at every epoch, and the final mean square error is 0.01. This case was generated with an *ad-hoc* selection of the training data rather than a really random one. We tried to stimulate the effects of the non-differentiability by using training data that fall exactly at the non-differentiability points. The training set is composed of eleven evenly spaced points in [-1,1]. Those points do not change in each epoch. In each epoch we also introduce five training points equal to the c_i values at the particular iteration. Therefore, we are using some points in the training data were the objective function is non-differentiable with respect to its parameters. If we were using a true gradient descent approach and a small enough step-size, we should always observe a decrease in instantaneous error at every step (not epoch). For this reason η = 0.01 was chosen, in order to exclude (or limit) non-improving steps due to overstepping (i.e., a learning rate that is too big). Moreover, since this is not a true gradient descent, at some steps we might observe an increase in error function only due to the non-improving

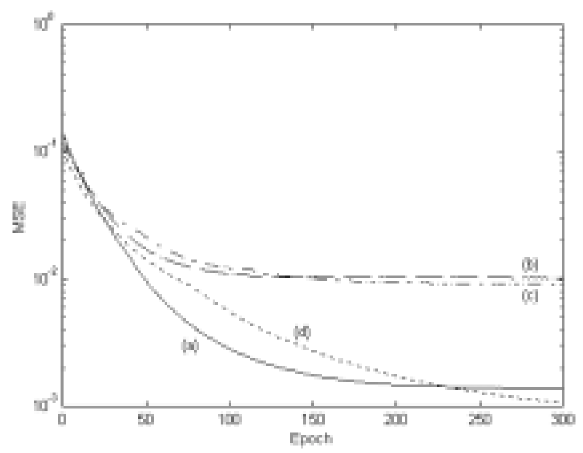

Figure 2. Convergence history for: (a) $\delta = 0.1$, (b) non-differentiable case $\delta = 0$, (c) $\delta = 0.01$, and (d) differentiable case with same training data as (b)

nature of the (non gradient) direction that was taken, and not due to overstepping. Any non-improving step was checked using a very small step-size (10^{-10}) in order to verify the reason for the non-improvement. Five steps per epoch were taken along non-gradient directions, for 300 epochs, thus giving a total of 1500 possible non-improving steps. The nature of these directions (whether improving or not) depends of the particular problem at hand and of all its characteristics. Thus, it is not easy to predict whether a direction will be improving the error or not. On the other hand, we know that the direction of the negative gradient is necessarily improving. In this case, of the total of 1500 possibilities there were 145 non-improving steps of which 4 were due to overstepping (the direction was indeed a descent direction). Thus, roughly 10% of the non-gradient direction turned out to be non-improving. This, added to the fact that hitting by accident one of those non-differentiable points is a very remote event, seems to make this case quite resilient to having big troubles due to the non-differentiability.

The solid line of Fig. 4 shows the situation at one of the non-differentiable points. Here the instantaneous error $E_j(\boldsymbol{c},\boldsymbol{p})$ is shown at one of the non-differentiable points, and the behavior of the error function along the update (non-gradient) direction is depicted as a function of the step size, η. The plot was generated using an initial step size equal to zero, and then 50 step sizes logarithmically spaced between 10^{-10} and 10^{-1}. It can be seen that the direction is indeed a non-improving direction as the error strictly increases for $\eta \in [0, 0.1]$ (at least). The same situation holds for all the other 141 non-improving directions.

In order to understand what are the (practical) damages induced by the non-differentiability, a second experiment was performed. In this experiment the conditions are exactly the same as the previous case, with the only difference that the training data that are dynamically generated are not positioned at $x = c_i$ anymore. They are positioned at $x = c_i + \delta$, where δ is a small parameter that we call a *detuning* parameter. Indeed, its effect is to create a very small detuning from the non-differentiability points, thus making the problem differentiable, and the approach a true gradient descent. It was chosen $\delta = 0.1$ with exactly same initialization (random with same seed). The rationale for this approach is to generate a case very similar to the latter experiment, trying to understand the problems caused by non-differentiability and closeness to it. The final mean square error for this case was 0.0014, that is one order of magnitude smaller than the error obtained in the previous case. The solid line (a) of Fig. 2 shows the convergence history of the mean square error. In the entire evolution of the algorithm there was only one non-improving step and overstepping caused it. It thus seems reasonable to conclude that non-differentiability could theoretically cause some divergence effects if everything goes wrong at the same time (as observed in [3]), but these effects were not observed in this case. Note that the strange approach that was taken to generate training data that fall exactly at the non-differentiable points can be regarded as an on-line approach where the training points are randomly sampled, or in general, are generated by a process that is out of our control. Moreover, in principle we should not be restricted by the training data that we use.

Comparing the errors for these two approaches (with and without detuning) we cannot be entirely sure whether the different convergence history is caused by the non-differentiability or by the different training data sets. Indeed, (some of) the training data are chosen according to the evolution of the antecedent parameters. Since the parameters evolve differently in the two cases, the data that are generated are also different. To investigate this effect, another experiment with same initialization and same data points of the first (non-differentiable) case was conducted. The only difference with the first case is that the data set of the first epoch was exchanged with the last one, since if even the first data set were the same then the algorithm would evolve exactly the same way. Thus, besides one minor ordering modification for the first epoch, all the other conditions are the same. This case converges to a final mean square error of 0.0011, with convergence history showed by the dotted line (d) of Fig. 2. Therefore, training with the same data as the non-differentiable case leads to convergence that is as good as the detuned case (a), and actually even a little better. It has a smaller final error and still seems to have some more space for improvement. Things were not supposed to be exactly the same, but the same qualitative behavior shows that probably the worse performance of the non-differentiable case shown in curve (b) is not due to non rich training data, but effectively it is due to the updates in non-improving directions (combination of training data and parameter values).

Another detuned case with *detuning* parameter $\delta = 0.01$ (and same initialization) was also considered. The final mean square error was 0.0089 and the convergence history

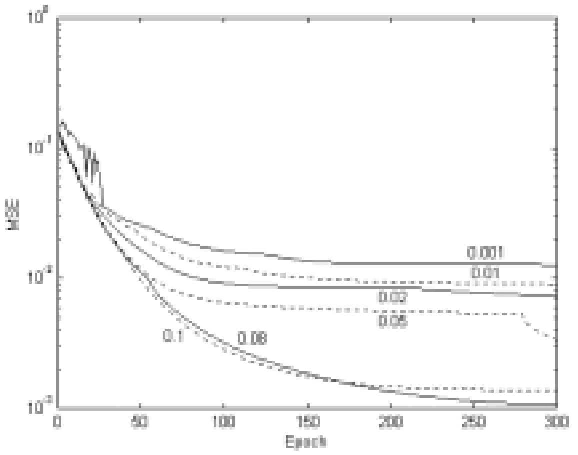

Figure 3. Convergence history for varying δ.

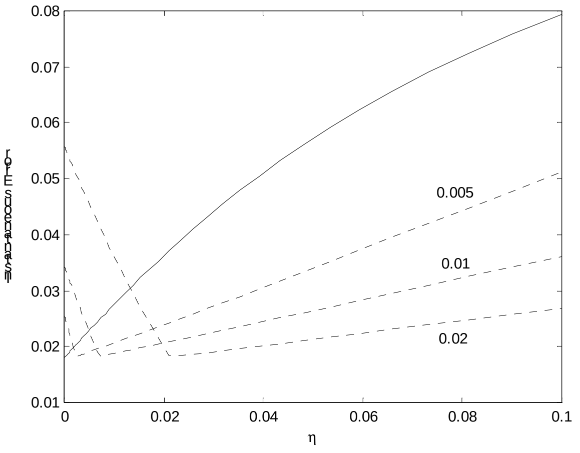

Figure 4. Behavior of the error function along the "gradient direction" for different values of δ.

is shown by the dash-dot line (c) in Fig. 2. Note that, even though all the cases shown in Fig. 2 are initialized exactly the same way, there is a small difference in initial errors due to the slightly different (initial) training data. As we can see from Fig. 2, there is a difference from the differentiable (a) to the non-differentiable (b) case that does not seem due to the different training data as shown by the behavior of line (d), very similar to (a). The convergence curve (c) seems to point to the fact that for $\delta \rightarrow 0$ (a) approaches (b) thus (c) is just an intermediate case. In the evolution of (c) obtained for $\delta = 0.01$ the instantaneous error increases 44 times because of overstepping. Thus a possible explanation is that at the extreme $\delta = 0$ there are non-improving directions, approaching this extreme the problem is differentiable and thus improving directions are generated, but the range of step sizes that offers an improvement becomes smaller and smaller. Moreover, the amount of the decrease becomes smaller and smaller as δ approaches zero, so that for $\delta = 0$ there is a smooth transition to a non-improving direction.

Some experiments in this direction were conducted to ascertain the influence of δ (closeness to non-differentiable points) on convergence. Training of the FLS with several values of δ was attempted and the corresponding learning curves were compared as shown in Fig. 3. Values of $\delta = 0.001$, $\delta = 0.01$, $\delta = 0.02$, $\delta = 0.05$, $\delta = 0.08$, and $\delta = 0.1$ were used. It can be seen that the convergence curves for the MSE vary smoothly (relatively smoothly, still taking into account that the analysis is approximated since the training data are different and we are just looking for main effects) from $\delta = 0.1$ to $\delta = 0.001$. It is easily recognized how the different curves orderly fill the space between the curves obtained for $\delta = 0.001$ and $\delta = 0.1$. Obviously, given the difference in training data, the order is not exactly satisfied at every epoch, but the plot shows a nice and orderly qualitative behavior. This seems to point to the fact that the closer the training data are to these points of non-differentiability, the higher the final mean square error, and thus the slower the algorithm. An insight into the motivation for this behavior, as well as a check of the possible mechanism underlying it, intuitively explained above, can be given by examining the behavior of the instantaneous error as a function of the step-size at one particular point in parameter space and for different values of the detuning parameter δ. Indeed it was already seen that for $\delta = 0$ a non-gradient direction that might not be improving is generated. On the other hand, the directions generated for small values of δ are necessarily going to be improving directions (since they are gradient directions) but the interval for improvement might shrink. This is shown in Fig. 4 through the behavior of the instantaneous error at one point in parameter space as a function of the step-size along the "gradient direction". This behavior is shown for $\delta = 0$ (non-differentiable case, solid line), $\delta = 0.005$, $\delta = 0.01$, and $\delta = 0.02$ (dotted lines). It can be seen that as the detuning parameter, δ, decreases, the interval of step sizes corresponding to improvement in error shrinks, and the amount of decrease in error gets smaller, finally ending in a non-improving direction corresponding to the non-differentiable case (solid line). Therefore, not only the non-differentiability raises theoretical concerns regarding the possible divergence of the algorithm as shown in [3], but it also creates a pragmatical concern since it might slow down the algorithm due to the overtaking of non-improving steps (due to overstepping) caused by excessive closeness to the points of non-differentiability. Moreover, in any type of application, regardless of the type of training data used (random or fixed) the membership functions move along the universe of discourse during the tuning, thus making it possible for them to get arbitrarily close to the points of non-differentiability.

5 Conclusions

This paper highlighted and discussed the threats offered by the non-differentiable nature of the optimization problem in supervised learning of certain FLSs. A simple SISO function approximation example was formulated and discussed. This example was beneficial in illustrating the risks posed by the non-differentiability of triangular membership functions. Indeed, it was shown how non-differentiability could theoretically cause divergence (as it does in a case shown in [3]), and also practically affect the evolution of the algorithm by slowing it down.

Finally, we can conclude that with this example we showed some of the problems involved with the non-differentiable nature of the triangular membership functions. Namely, the fact that they are non-differentiable at a finite number of points. Whenever the corresponding parameters assume any of these values, the gradient ceases to exist, and the resulting direction is not necessarily an improving direction anymore. Pragmatism would suggest that this case is not too important since the probability of hitting a point of non-differentiability is zero. The theoretical problem, however still exists: the approach is not a gradient descent and is not theoretically convergent; and, thus it seems very unsuitable especially for online applications. Moreover, we also showed how even only getting close to the points of non-differentiability deteriorates the performance (speed) of the algorithm, increasing the probability of overstepping. This seems to point to the fact that the non-differentiability problem is not only a mathematical abstraction after all, but indeed it is a pragmatic concern. It is the authors' opinion that research in both the practical and theoretical consequences of non-differentiability of triangular membership functions is necessary if it is desired to tune them. Moreover, the non-differentiability problem needs to be considered in tuning applications of fuzzy logic systems using triangular membership functions and/or min—max operators.

References

[1] P. Arabshahi, R.J. Marks, S. Oh, T.P. Caudell, J.J. Choi, and B.G. Song, "Pointer Adptation and Pruning of Min-Max Fuzzy Inference and Estimation," *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 44(**9**), 696-709, 1997.

[2] H. Bersini, and V. Gorrini, "An Empirical Analysis of one type of Direct Adaptive Fuzzy Control," *Fuzzy Logic and Intelligent Systems*, H. Li, and M. Gupta, Eds., Chapter 11, 289-309, Kluwer.

[3] P. Dadone, and H.F. VanLandingham, "Non-differentiable optimization of fuzzy logic systems," *Submitted for ANNIE 2000*, St. Louis, MI, 2000.

[4] P.Y. Glorennec, "Learning Algorithms for Neuro-Fuzzy Networks," *Fuzzy Control Systems*, A. Kandel, and G. Langholz, Eds., 4-18, CRC Press, Boca Raton, FL, 1994.

[5] J. Godjevac, "Comparative Study of Fuzzy Control, Neural Network Control and Neuro-Fuzzy Control," in *Fuzzy Set Theory and Advanced Mathematical Applications*, D. Ruan Ed., Kluwer Academic, Chapter 12, 291-322.

[6] F. Guely, and P. Siarry, "Gradient Descent Method for Optimizing Various Fuzzy Rule Bases," *Proc. 2nd IEEE Conf. on Fuzzy Systems*, 1241-1246, 1993.

[7] J.S.R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference Systems," *IEEE Trans. on SMC*, 23(**3**), 665-685, 1993

[8] R. Katayama, Y. Kajitani, and Y. Nishida, "A Self Generating and Tuning Method for Fuzzy Modeling Using Interior Penalty Method and Its Application to Knowledge Acquisition of Fuzzy Controller," *Fuzzy Control Systems*, A. Kandel, and G. Langholz, Eds., 198-224, CRC Press, Boca Raton, FL, 1994.

[9] H. Miyata, M. Ohki, and M. Ohkita, "Self-Tuning of Fuzzy Reasoning by the Steepest Descent Method and Its Application to a Parallel Parking," *IEICE Trans. on Information and Systems*, E79-D(**5**), 561-569, 1996.

[10] H. Nomura, I. Hayashi, and N. Wakami, "A Learning Method of Fuzzy Inference Rules by Descent Method," *Proc. 1st IEEE Conf. on Fuzzy Systems*, 203-210, 1992.

[11] H. Nomura, I. Hayashi, and N. Wakami, "Self-Tuning Fuzzy Reasoning By Delta Rule and Its Application to Obstacle Avoidance," *Japanese Journal of Fuzzy Theory and Systems*, 4(**2**), 261-272, 1992.

[12] M. Ohki, H. Miyata, M. Tanaka, and M. Ohkita, "A Self-Tuning of Fuzzy Reasoning by Modifying Learning Coefficients," *Proc. 2nd World Congress of Nonlinear Analysts*, 30(**8**), 5291-5301, 1997.

[13] W. Pedrycz, *Fuzzy Sets Engineering*, CRC Press, Boca Raton, FL, 1995.

[14] Y. Shi, and M. Mizumoto, "A new Approach of Neuro-Fuzzy Learning Algorithm for Tuning Fuzzy Rules," *Fuzzy Sets and Systems*, 112, 99-116, 2000.

[15] B.G. Song, R.J. Marks, S. Oh, P. Arabshahi, T.P. Caudell, and J.J. Choi, "Adaptive Membership Function Fusion and Annihilation in Fuzzy If-Then Rules," *Proc. 2nd IEEE Conf. on Fuzzy Systems*, 961-967, 1993.

[16] T. Takagi, and M. Sugeno, "Fuzzy identification of Systems and its Applications to Modeling and Control," *IEEE Trans. on SMC*, 15(**1**), 1985.

[17] L.N. Teow, and K.F. Loe, "Effective Learning in Recurrent Max-Min Neural Networks," *Neural Networks*, 11, 535-547, 1998.

[18] D. Wang, and L. Acar, "Fuzzy Identification of an Inverted Pendulum," in *Smart Engineering Systems*, C.H. Dagli, M. Akay, A.L. Buczak, O. Ersoy, and B.R. Fernandez, Eds., 8, 231-236.