

Improving Convergence of Parameter Tuning Process of the Additive Fuzzy System by New Learning Strategy

Thi Nguyen, Lee Gordon-Brown, Jim Peterson, and Peter Wheeler

Abstract—An additive fuzzy system comprising m rules with n inputs and p outputs in each rule has at least $m(2n + 2p + 1)$ parameters needing to be tuned. The system consists of a large number of if-then fuzzy rules and takes a long time to tune its parameters especially in the case of a large amount of training data samples. In this paper, a new learning strategy is investigated to cope with this obstacle. Parameters that tend toward constant values at the learning process are initially fixed and they are not tuned till the end of the learning time. Experiments based on applications of the additive fuzzy system in function approximation demonstrate that the proposed approach reduces the learning time and hence improves convergence speed considerably.

Keywords—Additive fuzzy system, improving convergence, parameter learning process, unsupervised learning.

I. THE ADDITIVE FUZZY SYSTEM AND ITS PARAMETERS

THE additive fuzzy system or the so-called Standard Additive Model (SAM) is a particular type of fuzzy systems proposed by Kosko [1, 3, 5, 6].

A fuzzy system $F: R^n \rightarrow R^p$ stores m if-then rules and can uniformly approximate continuous and bounded measurable functions in the compact domain [2]. This approximation theorem allows any choice of if-part fuzzy sets $A_j \subset R^n$. It also allows any choice of the then-part fuzzy sets $B_j \subset R^p$ because the system uses only the centroid c_j and volume V_j of B_j to compute the output $F(x)$ from the vector input $x \in R^n$.

$$F(x) = \text{Centroid} \left(\sum_{j=1}^m w_j a_j(x) B_j \right) \tag{1}$$

$$= \frac{\sum_{j=1}^m w_j a_j(x) V_j c_j}{\sum_{j=1}^m w_j a_j(x) V_j} = \sum_{j=1}^m p_j(x) c_j$$

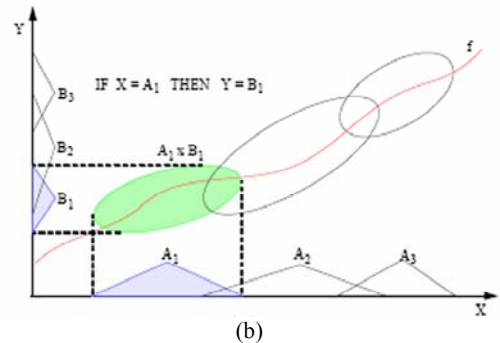
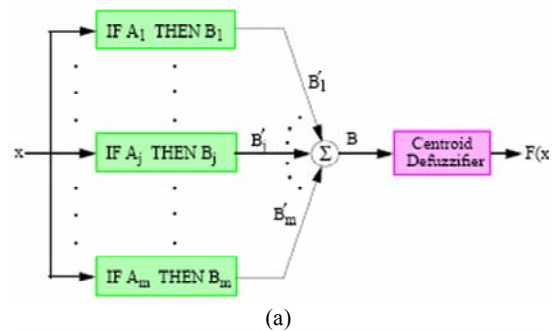


Fig. 1 (a) A parallel structure of SAM. Each input fires each fuzzy rule to some degree to compute $F(x)$ (b) Fuzzy rules define patches in the input-output space

The fuzzy system $F: R^n \rightarrow R^p$ covers the graph of an approximand f with m fuzzy rule patches of the form $A_j \times B_j \subset R^n \times R^p$ or “If $X = A_j$ then $Y = B_j$ ”. If-part set $A_j \subset R^n$ has joint set function $a_j: R^n \rightarrow [0, 1]$ that factors: $a_j(x) = a_j^1(x_1) \dots a_j^n(x_n)$. Then-part fuzzy set

Thi Nguyen was with the Centre for GIS, School of Geography and Environmental Science, Monash University, Victoria, Australia (e-mail: thi.nguyenthanh@gmail.com).

Lee Gordon-Brown is with the Department of Econometrics and Business Statistics, Monash University, Victoria, Australia.

Jim Peterson and Peter Wheeler are with the Centre for GIS, School of Geography and Environmental Science, Monash University, Victoria, Australia.

$B_j \subset R^p$ has set function $b_j: R^p \rightarrow [0, 1]$ and volume (or area in this case $p=1$) V_j and centroid c_j . The convex weights:

$$p_j(x) = \frac{w_j a_j(x) V_j}{\sum_{k=1}^m w_j a_k(x) V_k} \quad (2)$$

give the SAM output $F(x)$ as a convex sum of then-part set centroids.

Fig. 1 shows the parallel structure of the additive systems and its state-space graph cover. The graph cover leads to an exponential rule explosion. [11, 12] proposed using metrical joint unfactorable fuzzy sets based on metric and matrix knowledge to partly overcome this drawback. A fuzzy system needs on the order of k^{n+p+1} rules to approximate a function $f: R^n \rightarrow R^p$ in a compact domain. Optimal rules cover extrema and can help allocate a spare-rule budget in high dimensions [4]. Learning tends to move the rule patches toward the extrema or “bumps” and fill in with rule patches between bumps. Supervised learning tunes the parameters of the if-part set functions and also tunes the then-part volumes and centroids.

The choice of fuzzy set functions [7] affects how well fuzzy systems approximate functions. The most common fuzzy sets are triangles, trapezoids and Gaussian bell curves. The *sinc* set function $\frac{\sin(x)}{x}$ that gave the best and fastest function approximation in most cases [10] is chosen for experiments in this research. The j^{th} *sinc* set function (Fig. 2) centered at m_j and width $d_j > 0$ is defined as

$$a_j(x) = \sin\left(\frac{x - m_j}{d_j}\right) / \left(\frac{x - m_j}{d_j}\right) \quad (3)$$

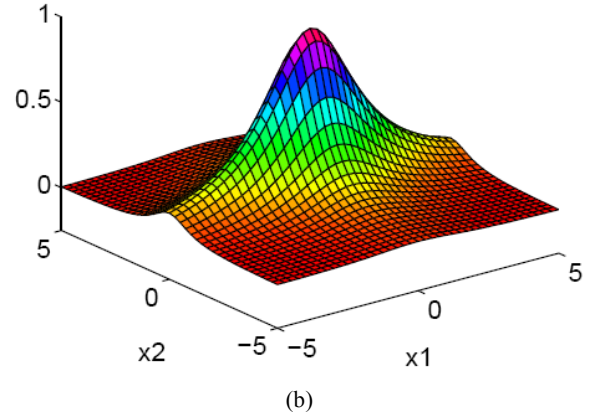
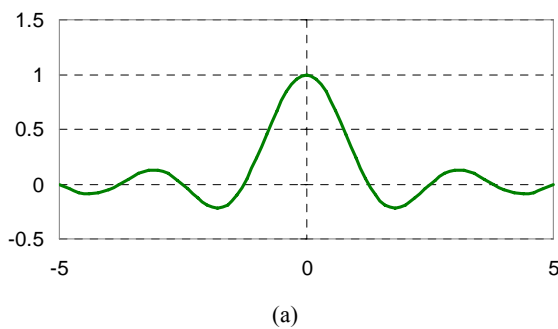


Fig. 2 Sinc set function in 1-D case (a) with centre $m = 0$ and width $d = 0.4$ and in 2-D case (b) with centres $m_1 = m_2 = 0$ and widths $d_1 = 0.8$ and $d_2 = 0.4$

In each if-then fuzzy rule, there are the following parameters: the weight of the rule (w_j), the volume (V_j) and centroid (c_j) of the then-part, and the parameters of if-part set function. Depending on the shape of fuzzy sets chosen, the number of parameters of the if-part set functions is different. For instance, the triangle set function has three parameters (left, centre and right), the trapezoid set function has four parameters (left, left-centre, right-centre, right) or the general bell curves have two parameters (mean and variance). A *sinc* function (Equation 3.) has two parameters: centre m_j and width d_j . So, an additive fuzzy system storing m if-then rules with n inputs and p outputs has at least of the order of $m(2n + 2p + 1)$ parameters needing to be tuned during a supervised learning process.

II. SUPERVISED LEARNING OF FUZZY RULES

Fuzzy rule parameters are tuned by the supervised learning process [3, 8, 9]. The supervised gradient descent can tune all the parameters in the SAM model. We seek to minimize the squared error

$$E(x) = \frac{1}{2} [f(x) - F(x)]^2 \quad (4)$$

of the function approximation. The vector function $f: R^n \rightarrow R^p$ has components $f(x) = (f_1(x), \dots, f_p(x))^T$ and so does the vector function F . Let ξ_j^k denote the k^{th} parameter in the set function a_j . Then the chain rule gives the gradient of the error function with respect to ξ_j^k , with respect to the then-part set centroid $c_j = (c_j^1, \dots, c_j^p)^T$, and with respect to the then-part set volume V_j .

A gradient descent learning law for a SAM parameter ξ has the form:

$$\xi(t+1) = \xi(t) - \mu_t \frac{\partial E}{\partial \xi} \quad (5)$$

where μ_t is the learning rate at iteration t .

Details of learning laws for parameters of an additive fuzzy systems are as the following equations where

$$\varepsilon(x) = -\frac{\partial E}{\partial F} = f(x) - F(x)$$

- Rule weights:

$$w_j(t+1) = w_j(t) + \mu_t \varepsilon(x) \left[c_j - F(x) \right] \frac{p_j(x)}{w_j} \quad (6)$$

- Centroids of then-parts:

$$c_j(t+1) = c_j(t) + \mu_t \varepsilon(x) p_j(x) \quad (7)$$

- Volumes of then-parts:

$$V_j(t+1) = V_j(t) + \mu_t \varepsilon(x) \left[c_j - F(x) \right] \frac{p_j(x)}{V_j} \quad (8)$$

Where the *sinc* set function is used, learning laws for its parameters are:

- Centres of if-parts:

$$m_j(t+1) = m_j(t) + \mu_t \varepsilon(x) \left[c_j - F(x) \right] \frac{p_j(x)}{a_j(x)} \times \left(a_j(x) - \cos\left(\frac{x - m_j}{d_j}\right) \right) \frac{1}{x - m_j} \quad (9)$$

- Widths of if-parts:

$$d_j(t+1) = d_j(t) + \mu_t \varepsilon(x) \left[c_j - F(x) \right] \frac{p_j(x)}{a_j(x)} \times \left(a_j(x) - \cos\left(\frac{x - m_j}{d_j}\right) \right) \frac{1}{d_j} \quad (10)$$

It is easy to realize that a large computational demand needs to be overcome during supervised learning process of fuzzy systems. Therefore, a new scheme rather than conventional approaches for this time-consuming task needs to be investigated.

III. NEW SUPERVISED LEARNING STRATEGY

The new supervised learning strategy is built via the following series of expressions.

$$\Delta_{r,e} = \sum_{i=1}^k \Delta \xi_i \quad (11)$$

where k is the number of parameters of the r^{th} rule and $\Delta \xi_i$ is the change of the i^{th} parameter of the r^{th} rule at the e^{th} epoch. So, $\Delta_{r,e}$ is the sum of changes of parameters of the r^{th} rule at the e^{th} epoch. One epoch of learning means all training data samples are passed through the system once to tune its parameters.

$$\Delta_r = \sum_{e=1}^E \Delta_{r,e} \quad (12)$$

where Δ_r is the sum of parameters' changes of the r^{th} rule after E epochs.

$$\Delta = \sum_{r=1}^m \Delta_r \quad (13)$$

where m is the number of rules in the system and Δ is the sum of parameters' changes of all rules after E epochs.

From Equations (11), (12) and (13), we arrive at:

$$\Delta = \sum_{r=1}^m \sum_{e=1}^E \sum_{i=1}^k \Delta \xi_i \quad (14)$$

To define an appropriate threshold to determine whether the rule is updated much slower than the remainder, we defined the average Δ/m

Establishing the expression Δ/m approximately implies that there will be half of the system rules having changes after E epochs more than this threshold, and the remaining half has the changes below this threshold.

If the r^{th} rule has the sum of changes of its parameters after E epochs satisfying the following inequality, that rule will be initially fixed and not be further trained until late in the learning time.

$$\Delta_r = \sum_{e=1}^E \Delta_{r,e} \leq \eta \frac{\Delta}{m} \quad (10)$$

where η is a constant.

The constant η plays an important role in determining the percentage of rules will be fixed whenever checking is carried out after each E -epoch. For instance, $\eta = 0.2$ means there will be approximately 10% (equal to $0.2 * 50\%$) of rules of the current system will be fixed after each E -epoch.

IV. EXPERIMENTAL EVALUATIONS

In order to evaluate the proposed learning strategy, we apply the additive fuzzy system for function approximation. The experiments are performed on the variety of function types: 1-D (Dimension), 2-D and 3-D with the *sinc* set function and the results are assessed in terms of the mean squared error (MSE) of the function approximation and the convergence time for a fixed learning rate. Below are three sample test functions used as approximands. The variables x, y, z are all investigated in $[-1, 1]$.

$$f(x) = 10 \left(e^{-5|x|} + e^{-3|x-0.8|/10} + e^{-10|x+0.6|} \right)$$

$$g(x, y) = 8 \sin(10x^2 + 5x + 1) \times \left[e^{-\left(\frac{y-0.1}{0.25}\right)^2} - 0.8e^{-\left(\frac{y+0.75}{0.15}\right)^2} - 0.4e^{-\left(\frac{y-0.8}{0.1}\right)^2} \right]$$

$$h(x, y, z) = 0.1 \left(e^{-\frac{|x|}{0.2}} + e^{-\frac{|x-0.8|}{0.3}} + e^{-\frac{|x+0.6|}{0.1}} \right) \times \left(\tan^3(1.5y) + 10 \tan^2(y) - 20 \tan(0.7y) \right) \times \left(\arccos^3(z) - \arccos^2(-z) - \arccos(-z) \right)$$

In the 1-D case, 731 points of the function are sampled to give the training set while this number for the 2-D case is 2715 samples and for the 3-D case is 8120 samples. The samples are collected by spreading points uniformly in the input space with the number much more than wanted and then randomly selecting with the probability equal to wanted number/spread number.

The learning rates were small (Table I, II, III) because each learning law is highly nonlinear. Otherwise learning might not have converged [12]. These rates were set up at the same value for both types of experiments: conventional and new strategies in order to appraise performances. The learning rates used range from $\mu = 10^{-6}$ to 10^{-4} .

TABLE I
SIMULATION WITH $f(x)$ - 1-D CASE SP MEANS SUPERVISED LEARNING

Number of rules	201	
Training samples	731	
MSE before SP	2.867	
MSE expected	0.100	
Learning rate	10^{-4}	
Approaches	Normal	New
Epoch number		300
Constant η		0.25
Epochs performed	1670	2159
Training time	18 min. 51 sec.	13 min. 16 sec.

Different initializations led to convergence to different local minima of the squared error surface. There is no formal way to find the initial conditions that lead to the global minimum [10, 12]. Centers of if-parts m_j are uniformly spread in the determined interval along the x -axis. Centroids of then-part c_j are picked as the values of the sampled approximand f at m_j : $c_j = f(m_j)$. Remaining parameters of fuzzy rules are initialized randomly including: weight of rules, and volume of then-parts.

To evaluate the performances of conventional (normal) and

new approaches, we used the same systems, same training samples, same learning rate and set the same expected MSE value and then measured separately the training time.

Computer configuration for experiments: Pentium(R) 4 1.80 GHz, 760 MB of RAM.

TABLE II
SIMULATION WITH $g(x, y)$ - 2-D CASE

Number of rules	441	
Training samples	2715	
MSE before SP	15.621	
MSE expected	0.500	
Learning rate	10^{-5}	
Approaches	Normal	New
Epoch number		400
Constant η		0.4
Epochs performed	3019	4166
Training time	33 min. 02 sec.	20 min. 15 sec.

TABLE III
SIMULATION WITH $h(x, y, z)$ - 3-D CASE

Number of rules	729	
Training samples	8120	
MSE before SP	22.548	
MSE expected	1.000	
Learning rate	10^{-6}	
Approaches	Normal	New
Epoch number		500
Constant η		0.5
Performed epochs	6482	8377
Training time	56 min. 26 sec.	31 min. 48 sec.

The number of rules as well as the number of training samples in case of the higher-D SAM is much more than those of the lower-D SAM consistent with the curse of high dimensions in fuzzy function approximation - exponential rule explosion. This results directly from the factorability of in-part fuzzy sets in fuzzy if-then rules [12].

By reducing approximately 12.5% (1-D case), 20% (2-D case) and 25% (3-D case) number of fuzzy rules after each 300 (1-D case), 400 (2-D case) and 500 (3-D case) epochs, the number of epochs performed in the new learning scheme is more than that of conventional approach. This leads to the decrease of training time whereas the MSE expected is the same value for both approaches. The learning time is decreased around 30% (1-D case), 39% (2-D case) and 44% (3-D case). The new scheme is more effective in case of high dimension, partly due to the more percentage of fuzzy rules fixed based on the higher value of the constant η ($\eta = 0.5$ in 3-D case compared with $\eta = 0.4$ in 2-D case and $\eta = 0.25$ in 1-D case).

V. CONCLUSION

The new learning scheme has partly overcome the time-consuming and tedious tasks of parameter tuning of fuzzy systems applied in function approximation. Using these arbitrary functions, we found improvements of the order of 35% in convergence speed, and implying increases in the accuracy of fuzzy applications where the optimisation time is limited. The new strategy can be applied not only in function approximation but also in fields such as pattern recognition, signal processing, time series prediction, etc. where SAM as well as other types of fuzzy systems have been explored effectively in recent decades.

REFERENCES

- [1] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall, 1991.
- [2] B. Kosko, "Fuzzy systems as universal approximators", *IEEE Transactions on Computers*, vol. 43, no. 11, 1994, pp. 1329-1333.
- [3] B. Kosko, "Combining fuzzy systems". *Proceedings of the IEEE International Conference on Fuzzy Systems (IEEE FUZZ-95)*, 1995, pp. 1855-1863.
- [4] B. Kosko, "Optimal fuzzy rules cover extrema". *International Journal of Intelligent Systems*, vol. 10, no. 2, 1995, pp. 249-255.
- [5] B. Kosko, *Fuzzy Engineering*. Prentice Hall, 1996.
- [6] B. Kosko, "Global stability of generalized additive fuzzy systems", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 28, no. 3, 1998, pp. 441-452.
- [7] G.J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, Upper Saddle River, NJ, USA, 1995.
- [8] J.A. Dickerson and B. Kosko, "Fuzzy function approximation with supervised ellipsoidal learning", *World Congress on Neural Networks*, vol. 2, 1993, pp. 9-13.
- [9] J.A. Dickerson and B. Kosko, "Fuzzy function approximation with ellipsoidal rules", *IEEE Transactions Systems, Man, and Cybernetics*, vol. 26, no. 4, 1996, pp. 542-560.
- [10] S. Mitaim and B. Kosko, "What is the best shape for a fuzzy set in function approximation?", *Proceeding of the 5th IEEE International Conference on Fuzzy Systems*, vol. 2, 1996, pp. 1237-1243.
- [11] S. Mitaim and B. Kosko, "Adaptive joint fuzzy sets for function approximation", *Proceeding of IEEE International Conference on Neural Networks*, vol. 1, 1997, pp. 537-542.
- [12] S. Mitaim and B. Kosko, "The shape of fuzzy sets in adaptive function approximation", *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, 2001, pp. 637-656.