The Shape of Fuzzy Sets in Adaptive Function Approximation

Sanya Mitaim and Bart Kosko

***Abstract*—The shape of if-part fuzzy sets affects how well feedforward fuzzy systems approximate continuous functions. We explore a wide range of candidate if-part sets and derive supervised learning laws that tune them. Then we test how well the resulting adaptive fuzzy systems approximate a battery of test functions. No one set shape emerges as the best shape. The sinc function often does well and has a tractable learning law. But its undulating sidelobes may have no linguistic meaning. This suggests that the engineering goal of function-approximation accuracy may sometimes have to outweigh the linguistic or philosophical interpretations of fuzzy sets that have accompanied their use in expert systems. We divide the if-part sets into two large classes. The first class consists of n-dimensional joint sets that factor into n scalar sets as found in almost all published fuzzy systems. These sets ignore the correlations among vector components of input vectors. Fuzzy systems that use factorable if-part sets suffer in general from exponential rule explosion in high dimensions when they blindly approximate functions without knowledge of the functions. The factorable fuzzy sets themselves also suffer from what we call the second curse of dimensionality: The fuzzy sets tend to become binary spikes in high dimension. The second class of if-part sets consists of the more general but less common n-dimensional joint sets that do not factor into n scalar fuzzy sets. We present a method for constructing such unfactorable joint sets from scalar distance measures. Fuzzy systems that use unfactorable if-part sets need not suffer from exponential rule explosion but their increased complexity may lead to intractable learning laws and inscrutable if-then rules. We prove that some of these unfactorable joint sets still suffer the second curse of dimensionality of spikiness. The search for the best if-part sets in fuzzy function approximation has just begun.**

***Index Terms*—Adaptive fuzzy system, curse of dimensionality, fuzzy function approximation, fuzzy sets.**

I. THE SHAPE OF FUZZY SETS: FROM TRIANGLES TO WHAT?

WHAT is the best shape for fuzzy sets in function approximation? Fuzzy sets can have any shape. Each shape affects how well a fuzzy system of if-then rules approximate a function. Triangles have been the most popular if-part set shape but they surely are not the best choice [24], [32] for approximating nonlinear systems. Overlapped symmetric triangles or trapezoids reduce fuzzy systems to piecewise linear systems. Gaussian bell-curve sets give richer fuzzy systems with simple learning laws that tune the bell-curve means and variances. But this popular choice comes with a special cost: It converts fuzzy systems to radial-basis-function neural networks or to other well-known systems that predate fuzzy systems [3], [17], [20], [27], [28], [30]. These Gaussian systems make important benchmarks but there is no scientific advance involved in their rediscovery.

Triangles and Gaussian bell curves also do not represent the vast function space of if-part fuzzy sets. But then which shapes do? This question has no easy answer. A key part of the problem is that we do not know what should count as a meaningful taxonomy of fuzzy sets. We can distinguish continuous fuzzy sets from discontinuous sets, differentiable from nondifferentiable sets, monotone from nonmonotone sets, unimodal from multimodal sets, and so on. But these binary classes of fuzzy sets may still be too general to permit a fruitful analysis in terms of function approximation or in terms of other performance criteria. Yet a taxonomy requires that we draw lines somewhere through the function space of all fuzzy sets.

We draw two lines. The first line answers whether a joint fuzzy set is factorable or unfactorable. Consider any fuzzy set $A \subset R^n$ with arbitrary set function $a : R^n \rightarrow [0,1]$ (or the slightly more general case where a maps R^n or some other space X into some connected real interval $[u,v] \subset R$). The multidimensional nature of fuzzy set A presents a structural question that does not arise in the far more popular scalar or one-dimensional case: Is A *factorable*? Does $A \subset R^n$ factor into a Cartesian product of n scalar fuzzy sets $A_j \subset R : A = A_i \times \cdots \times A_n$?

The general answer is no. Factorability is rare in the space of all n-dimensional mappings of R^n into numbers. It corresponds to uncorrelatedness or independence in probability theory. Yet much analysis focuses on the factorable exceptions of hyperrectangles and multivariate Gaussian probability densities. And almost all published fuzzy systems use rules that deliberately factor the if-part sets into scalar sets. This often yields factorable joint set functions of the form $a_j(x) = a_j^1(x_1) \times \cdots \times a_j^n(x_n)$ or $a_j(x) = \min(a_j^1(x_1), \ldots, a_j^n(x_n))$. Consider this rule for a simple air-conditioner controller: "If the air is warm and the humidity is high then set the blower to fast." A triangle or trapezoid or bell curve might describe the fuzzy subset of warm air temperatures or of high humidity values. A product of these two scalar sets forms a factorable fuzzy subset $A_1 \times A_2 \subset R^2$. But users tend not to work with even simple unfactorable two-dimensional (2-D) sets such as ellipsoids: "If the temperature-humidity values lie in the warm-high planar ellipsoid then set the motor speed to fast." Few unfactorable fuzzy subsets of the plane or of R^n are as simple geometrically or as tractable mathematically as ellipsoids [1], [2].

Below we study how well feedforward additive fuzzy systems can approximate test functions for both adaptive factorable and unfactorable if-part fuzzy sets. We first derive supervised learning laws for a wide range of fuzzy sets of different shape and then test them against one another in terms of how accurately they approximate the test functions in a squared-error

Manuscript received November 19, 1999; revised April 13, 2001. This work was supported in part by the National Science Foundation under Grant ECS-0 070 284 and in part by the Thailand Research Fund under Grant PDF/29/2543.

S. Mitaim is with the Department of Electrical Engineering, Thammasat University, Pathumthani 12121, Thailand.

B. Kosko is with the Department of Electrical Engineering—Systems, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-2564 USA.

Publisher Item Identifier S 1063-6706(01)06658-9.

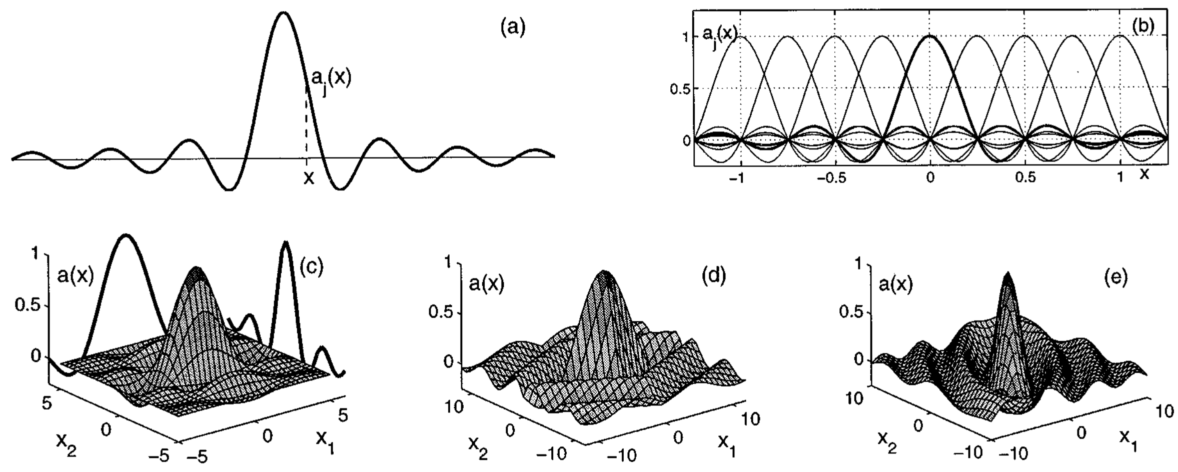


Fig. 1. Samples of sinc set functions for one-input and two-input cases. (a) Scalar sinc set function for one-input case. (b) Nine scalar sinc set functions for input x. All sinc set functions have the same width but their centers differ. (c) Product sinc set function for two-input case. The set function has the form $a_j(x) = a_j(x_1, x_2) = a_j^1(x_1) \times a_j^2(x_2)$. The shadows show the scalar sinc set functions $a_j^i : R \to R$ for $i = 1, 2$ that generate $a_j : R^2 \to R$. (d) Joint l^1 metrical sinc set function: $a_j(x) = \text{sinc}(d_j^1(x, m_j))$. (e) Joint quadratic metrical sinc set function: $a_j(x) = \text{sinc}(d_j^2(x, m_j))$.

sense. Then we form factorable n-dimensional fuzzy sets from the scalar factors and compare them both against one another and against some new unfactorable joint fuzzy sets. Exponential rule explosion severely constrains the extent of the simulations. We also uncover a second curse of dimensionality: Factorable sets tend toward binary spikes in high dimension. Unfactorable sets need not suffer exponential rule explosion. But we prove that some of them also suffer from spikiness in high dimensions.

We draw a second line between parametrized and nonparametrized fuzzy sets. We study only parametrized fuzzy sets because only for them could we define learning laws (that tune the parameters). We did not study recursive fuzzy sets such as those that can arise with B-splines [33] or other recursive algorithms. It also is not clear how to fairly compare parametrized if-part set functions with nonparametrized set functions for the task of *adaptive* function approximation.

The simulation results do not pick a clear-cut winner. Nor would we expect them to do so given the ad hoc nature of our choices of both candidate set functions and test functions. But the results do suggest that some nonobvious set functions should be among those that a fuzzy engineer considers when building or tuning a fuzzy system. Along the way we also developed an extensive library of new set functions and derived their often quite complex learning laws.

Perhaps the most surprising and durable finding is that the sinc function $(\sin x/x)$ of signal processing often converges fastest and with greatest accuracy among candidates that include triangles, Gaussian and Cauchy bell curves, and other familiar set shapes. This appears to be the first use of the sinc function as a fuzzy set. We could find no theoretical reason for its performance as a nonlinear interpolator in a fuzzy system despite its well-known status as the linear interpolator in the Nyquist sampling theorem and its signal-energy optimality properties [21]. We also combined two hyperbolic tangents to give a new bell curve that often competes favorably with other if-part set candidates. We call this new bell curve the difference hyperbolic tangent [18].

Fig. 1 shows scalar and joint sinc set functions. Fig. 1(a) shows the decaying sidelobes that can take on negative values. This requires that we view the sinc as a generalized fuzzy set [14] whose set function maps into a totally ordered interval that includes negative values: $a : R \to [-0.217, 1]$. An exercise shows that such a bipolar set-function range does not affect the set-theoretic structure of A in terms intersection, union, or complementation because the corresponding operations of minimum, maximum, and order reversal depend on only the total ordering (with a like result for triangular or t-norms [8]). Fig.1(c) shows the 2-D factorable sinc that results when we multiply two scalar sinc functions as we might do to compute the degree to which a two-vector input $x = (x_1, x_2)$ fires the two if-part factors of a rule of the form "If X_1 is A_1 and X_2 is A_2 then Y is B_1." Fig. 1(d) and (e) show two new unfactorable 2-D set functions built from the scalar sinc function and a distance metric.

Below we derive the supervised learning law that tunes these sinc set functions given input–output samples from a test function. The factorable joint set functions are far easier to tune than are the unfactorable sets because we need only add one more term to a partial-derivative expansion and then multiply the results for tuning the individual factors. Fig. 2 shows how a 2-D factorable or product sinc set evolves as the process of supervised learning unfolds when a sinc-based fuzzy system approximates a test function.

The sinc finding raises a broader issue: Does an if-part fuzzy set need to have a linguistic meaning? The very definition of the sinc set function $a : R \to [-0.217, 1]$ already requires that we broaden our usual notion of "degrees" that range from 0% to 100% to a more general totally ordered scale. But the sinc's undulating and decaying sidelobes admit no easy linguistic interpretation. We could simply think of the smooth bell-shaped *envelope* of the sinc and treat it as we would any other unimodal curve that stands for warm air or high humidity or fast blower speeds. That would solve the problem in practice and would allow engineers to safely interpret a domain expert's fuzzy concepts as appropriately centered and scaled sinc sets. But that would not address the conceptual problem of how to make sense

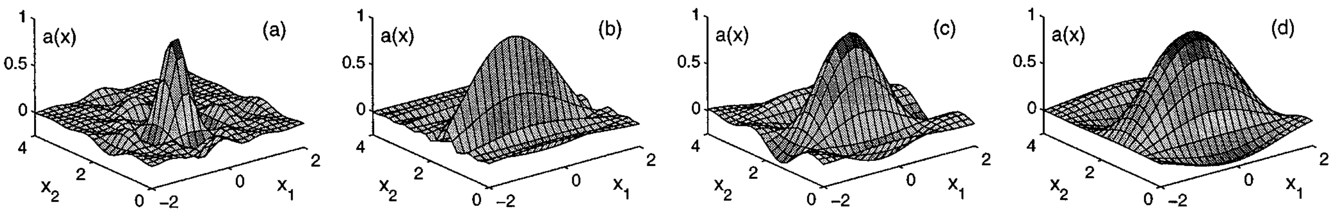


Fig. 2. Samples of evolution of a product sinc if-part set function in an adaptive function approximator. Supervised learning tunes the parameters of the product sinc set function such as its center and width on each parameter axis x_1 and x_2: (a) a sinc set function at initial state, (b) the same sinc set after 10 epochs of learning, (c) after 500 epochs, and (d) the sinc set converges after 3000 epochs.

of all those local minima and maxima in such a multimodal set function.

A pragmatic answer is that a given if-part fuzzy set need not have a precise linguistic meaning or have any tie to natural language at all. Function approximation is a global property of a fuzzy system. If-part fuzzy sets are local parts of local if-then rules. The central goal is accurate function approximation. This can outweigh the linguistic and philosophical concerns that may have attended earlier fuzzy expert systems. Engineers designed many of those earlier systems not to accurately approximate some arbitrary nonlinear function but to accurately model an expert's knowledge as the expert stated it in if-then rules.

So the real issue is the gradual shift in performance criteria from accuracy of linguistic modeling to accuracy of function approximation. Progress in fuzzy systems calls into question the earlier goal of simply modeling what a human says. That goal remains important for many applications and no doubt always will. But it should not itself constrain the broader considerations of fuzzy function approximation. The function space of all if-part fuzzy sets is simply too vast and too rich for natural language to restrict searches through it.

II. Fuzzy Function Approximation and Two Curses of Dimensionality

We work with scalar-valued additive fuzzy systems $F : R^n \rightarrow R$. These systems approximate a function $f : R^n \rightarrow R$ by covering the graph of f with fuzzy rule patches and averaging patches that overlap [14]. An if-then rule of the form "If X is A then Y is B" defines a fuzzy Cartesian patch $A \times B$ in the input–output space $X \times Y$. The rules can use fuzzy sets of any shape for either their if-part sets A or then-part sets B. This holds for the feedforward standard additive model (SAM) fuzzy systems discussed below. Their generality further permits any scheme for combining if-part vector components because all theorems assume only that the set function maps to numbers as in $a : R^n \rightarrow [u, v]$. The general fuzzy approximation theorem [11] also allows any choice of if-part set or then-part sets for a general additive model and still allows any choice of if-part set for the SAM case that in turn includes most fuzzy systems in use [15].

The fuzzy approximation theorem does not say which shape is the best shape for an if-part fuzzy set or how many rules m a fuzzy system should use when it approximates a function. The shape of if-part sets A_j affects how well the feedforward SAM F approximates a function f and how quickly an adaptive SAM F approximates it when learning based on input–output samples from f tunes the parameters of A_j and the centroids c_j and volumes V_j of the then-part set B_j. The shape of then-part sets B_j does not affect the *first-order* behavior of a feedforward SAM F beyond the effect of the volume V_j and centroid c_j. This holds because the SAM output computes only a convex-weighted sum of the then-part centroids c_j for each vector input x

$$F(x) = \sum_{j=1}^{m} p_j(x) c_j \tag{1}$$

where $p_j(x) \geq 0$ and $\sum_{j=1}^{m} p_j(x) = 1$ for each $x \in R^n$ as defined in (6). p_j depends on B_j only through its volume or area V_j (and perhaps through its rule weight). We also note that (1) and (2) imply that $F(x) = E[Y \,|\, X = x]$ [14]. But the shape of B_j does affect the *second-order* uncertainty or conditional variance $V[Y \,|\, X = x]$ of the SAM output $F(x)$ [14]

$$V[Y \,|\, X = x] = \sum_{j=1}^{m} p_j(x) \sigma_{B_j}^2 + \sum_{j=1}^{m} p_j(x)[c_j - F(x)]^2 \tag{2}$$

where $\sigma_{B_j}^2$ in an SAM is the then-part set variance

$$\sigma_{B_j}^2 = \int_{-\infty}^{\infty} (y - c_j)^2 p_{B_j}(y)\, dy \tag{3}$$

and where $p_{B_j}(y) = b_j(y)/V_j$ is an integrable probability density function and $b_j : R \rightarrow [0, 1]$ is the integrable set function of then-part set B_j. The first term on the right side of (2) gives an input-weighted sum of the then-part set uncertainties. The second term measures the interpolation penalty that results from computing the SAM output $F(x)$ in (1) as simply the weighted sum of centroids. The output conditional variance (2) further simplifies if all then-part sets B_j have the same shape and thus all have the same inherent uncertainty σ^2

$$V[Y \,|\, X = x] = \sigma^2 + \sum_{j=1}^{m} p_j(x)[c_j - F(x)]^2. \tag{4}$$

So a given input x minimizes the system uncertainty or gives an output $F(x)$ with maximal confidence if it fires the jth rule dead-on (so $F(x) = c_j$) and does not fire the other $m-1$ rules at all ($p_k(x) = 0$ for $k \neq j$). This justifies the common practice of centering a symmetric unimodal if-part fuzzy set A_j at a point where the other $m-1$ if-part sets have zero membership degree. It does not justify the equally common practice of ignoring the thickness or thinness of the then-part sets B_j and even replacing them with the maximally confident choice of binary "singleton" spikes centered at the centroid c_j. The second-order structure of a fuzzy system's output depends crucially on the size and shape of the then-part sets B_j.

We allow learning to tune the volumes V_j and centroids c_j of the then-part sets B_j in our adaptive function-approximation simulations. A then-part set B_j with volume V_j and centroid c_j can have an infinitude of shapes. And again many of these shapes will change the output uncertainty in (2) or (4). But we too shall ignore the second-order behavior that (2) and (4) describe.

High dimensions present further problems for fuzzy function approximation. Feedforward fuzzy systems suffer at least two curses of dimensionality. The first is the familiar exponential rule explosion. This results directly from the factorability of if-part fuzzy sets in fuzzy if-then rules. The second curse is one that we call the second curse of dimensionality: factorable if-part sets tend to binary spikes as the dimension n increases.

Consider first rule explosion for blind function approximation. Suppose we can factor the if-part fuzzy set $A : A = A_1 \times \cdots A_n$. Nontrivial if-then rules require that we use at least two scalar factors for each of the n orthogonal axes in R^n as in the minimal fuzzy partition of air temperatures into warm and not-warm temperatures or into low and high temperatures. A fuzzy system must cover the graph of the function f with rule patches. That entails that the if-part sets cover the system's domain—else the fuzzy system F would not be defined on those regions of the input space. So such a rule-patch cover of the domain of a fuzzy system $F : C \subset R^n \to R$ entails a rule explosion on the order of k^n where C is some compact subset of R^n. We will for convenience often denote functions as $F : R^n \to R$ or as $a : R^n \to [0, 1]$ where we understand that the domain is only some compact subset of R^n.

There is a related exception that deserves comment. Watkins [31], [32] has shown that if we not only know the functional form of f but build it into the very structure of the if-part sets A_j then we can exactly *represent* many functions in the sense of $F(x) = f(x)$ for all x and can do so with a number of rules that grows linearly with the dimension n. This does not apply in *blind* approximation where we pick the tunable if-part sets A_j in advance and then train them and other parameters based on exact or noisy input–output samples from the approximand function f. But it suggests that there may be many types of middle ground where partial knowledge of f may reduce the rule complexity from exponential to polynomial or perhaps to some other tractable function of dimension.

All factorable if-part sets suffer the second curse of dimensionality. They ignore input structure and collapse to binary-like spikes in high dimensions. The separate factors a_j^i ignore correlations and other nonlinearities among the input variables [5]. This structure can be quite complex in high dimensions. The product form $a_j^1(x_1) \times \cdots \times a_j^n(x_n)$ tends toward a spike in R^n for large n when $a_j^i < 1$. The Borel–Cantelli lemma of probability theory shows that $\min(a_j^1(x_1), \ldots, a_j^n(x_n))$ tends to zero with probability one [9] as $n \to \infty$ if the random sequence $x_1, x_2, \ldots$ is independent and identically distributed. This also holds for *any* t-norm combination of factors because of the generalized t-norm bound $T(a_j^1(x_1), \ldots, a_j^n(x_n)) \leq \min(a_j^1(x_1), \ldots, a_j^n(x_n))$. Factorable joint set functions degenerate in high dimensions. This curse of dimensionality can combine with the better known curse of exponential rule explosion. The result can be a function approximator with a vast set of spiky rules.

Joint unfactorable sets tend to preserve input correlations [5]. They need not collapse to spikes in high dimensions or suffer from the like rotten-apple effect of falling to zero when just one term equals zero. This also suggests that some unfactorable joint fuzzy sets may lessen or even defeat the curse of dimensionality.

The second part of this paper shows how to create and tune metrical joint set functions. These joint set functions preserve at least the metrical structure of inputs and do not try to factor a nonlinear function into a product or other combination of n terms. The idea is to use one well-behaved scalar set function like $\operatorname{sinc}(x)$ [18] and apply it to an n-dimensional distance function $d_j(x)$ rather than multiply n of the scalar set functions: $a_j(x) = \operatorname{sinc}(d_j(x))$ rather than $a_j(x) = \prod_{i=1}^n \operatorname{sinc}(x_i)$. Then supervised learning tunes the metrical joint set function as it tunes the metric. The next section reviews the standard additive fuzzy systems that we use to derive parameter learning laws and to test candidate if-part sets in terms of their accuracy of function approximation.

III. Additive Fuzzy Systems and Function Approximation

This section reviews the basic structure of additive fuzzy systems. The Appendix reviews and extends the more formal math structure that underlies these adaptive function approximators.

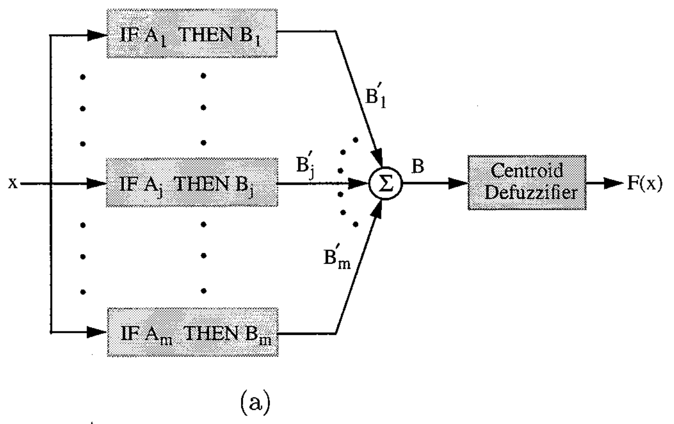
A fuzzy system $F : R^n \to R^p$ stores m rules of the word form "If $X = A_j$ Then $Y = B_j$" or the patch form $A_j \times B_j \subset X \times Y = R^n \times R^p$. The if-part fuzzy sets $A_j \subset R^n$ and then-part fuzzy sets $B_j \subset R^p$ have set functions $a_j : R^n \to [0, 1]$ and $b_j : R^p \to [0, 1]$. Generalized fuzzy sets [14] map to intervals other than $[0, 1]$. The system can use the joint set function a_j or some factored form such as $a_j(x) = a_j^1(x_1) \ldots a_j^n(x_n)$ or $a_j(x) = \min(a_j^1(x_1), \ldots, a_j^n(x_n))$ or any other conjunctive form for input vector $x = (x_1, \ldots, x_n) \in R^n$ [10].

An additive fuzzy system [10], [11] sums the "fired" then-part sets B'_j

$$B(x) = \sum_{j=1}^m w_j B'_j = \sum_{j=1}^m w_j a_j(x) B_j. \tag{5}$$

Fig. 3(a) shows the parallel fire-and-sum structure of the SAM. These nonlinear systems can uniformly approximate any continuous (or bounded measurable) function f on a compact domain [19]. Engineers often apply fuzzy systems to problems of control [4] but fuzzy systems can also apply to problems of communication [22] and signal processing [5], [6] and other fields.

Fig. 3(b) shows how three rule patches can cover part of the graph of a scalar function $f : R \to R$. The patch-cover structure implies that fuzzy systems $F : R^n \to R^p$ suffer from *rule explosion* in high dimensions. A fuzzy system F needs on the order of k^{n+p-1} rules to cover the graph and thus to approximate a vector function $f : R^n \to R^p$. Optimal rules can help deal with the exponential rule explosion. Lone or local mean-



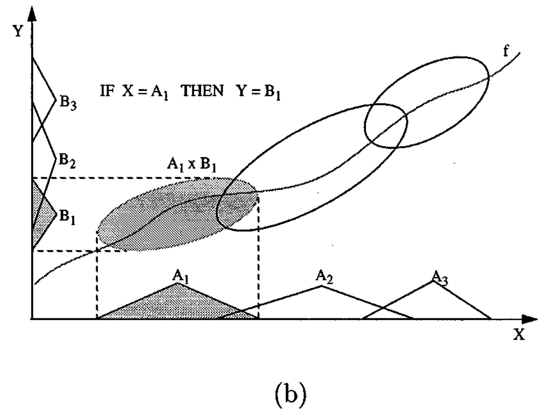


Fig. 3. Feedforward fuzzy function approximator. (a) The parallel associative structure of the additive fuzzy system $F : R^n \rightarrow R^p$ with m rules. Each input $x_0 \in R^n$ enters the system F as a numerical vector. At the set level x_0 acts as a delta pulse $\delta(x - x_0)$ that combs the if-part fuzzy sets A_j and gives the m set values $a_j(x_0) = \int_{R^n} \delta(x - x_0)a_j(x)\,dx$. The set values "fire" or scale the then-part fuzzy sets B_j to give B'_j. An SAM scales each B_j with $a_j(x)$. Then the system sums the B'_j sets to give the output "set" B. The system output $F(x_0)$ is the centroid of B. (b) Fuzzy rules define Cartesian rule patches $A_j \times B_j$ in the input–output space and cover the graph of the approximand f. This leads to exponential rule explosion in high dimensions. Optimal lone rules cover the extrema of the approximand as in Fig. 4.

squared optimal rule patches cover the extrema of the approximand f [13], [14]. They "patch the bumps" as in Fig. 4. The Appendix presents a simple proof of this fact. Better learning schemes move rule patches to or near extrema and then fill in between extrema with extra rule patches if the rule budget allows.

The scaling choice $B'_j = a_j(x)B_j$ gives an SAM. The Appendix further shows that taking the centroid of $B(x)$ in (5) gives the following SAM ratio [10], [11], [13], [14]:

$$F(x) = \frac{\sum_{j=1}^{m} w_j a_j(x) V_j c_j}{\sum_{j=1}^{m} w_j a_j(x) V_j} = \sum_{j=1}^{m} p_j(x) c_j. \tag{6}$$

Here V_j is the finite positive volume or area of then-part set B_j and c_j is the centroid of B_j or its center of mass. The convex weights $p_1(x), \ldots, p_m(x)$ have the form $p_j(x) = (w_j a_j(x) V_j / \sum_{i=1}^{m} w_i a_i(x) V_i)$. The convex coefficients $p_j(x)$ change with each input vector x. Sections V and VIII derive the gradient learning laws of all parameters of the SAM for different shapes of if-part sets.

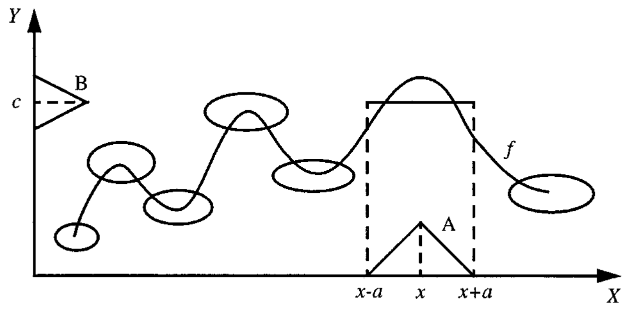


Fig. 4. Lone optimal fuzzy rule patches cover the extrema of approximand f. A lone rule defines a flat line segment that cuts the graph of the local extremum in at least two places. The mean value theorem implies that the extremum lies between these points. This can reduce much of fuzzy function approximation to the search for zeroes $\hat{x}$ of the derivative map $f' : f'(\hat{x}) = 0$.

IV. Scalar and Joint Factorable Fuzzy Set Functions

A scalar set function $a_j : R \rightarrow [0, 1]$ measures the degree to which input $x \in R$ belongs to the fuzzy or multivalued set $A_j : a_j(x) = \text{Degree}(x \in A_j)$. A joint factorable set $A_j \subset R^n$ derives from n scalar sets $A_j^i \subset R$. Any conjunctive operator such as a t-norm can combine n scalar sets to obtain a joint factorable set.

A. Scalar Fuzzy Sets

We tested a wide range of if-part set functions. Below we list the scalar form of most of these set functions. The sinc function was multimodal and could take on negative values in [−0.217, 1]. We viewed these negative values as low degrees of set membership.

1) *Triangle set function.* We define the triangle set function as a three-tuple (l_j, m_j, r_j) where $l_j > 0$ and $r_j > 0$. $m_j \in R$ denotes the location of a peak of the triangle

$$a_j(x) = \begin{cases} 1 - \dfrac{m_j - x}{l_j}, & \text{if } m_j - l_j \le x \le m_j \\ 1 - \dfrac{x - m_j}{r_j}, & \text{if } m_j < x \le m_j + r_j \\ 0, & \text{else} \end{cases} \tag{7}$$

We can also define the symmetric triangle set function with two parameters that are its *center* m_j and *width* d_j as

$$a_j(x) = \begin{cases} 1 - \left| \dfrac{x - m_j}{d_j} \right|, & \text{if } |x_j - m_j| < d_j \\ 0, & \text{else.} \end{cases} \tag{8}$$

2) *Trapezoid set function.* We define the trapezoid set function as a four-tuple (l_j, ml_j, mr_j, r_j) where $ml_j \le mr_j \in R$. $l_j > 0$ and $r_j > 0$ denote the distance of the support of a function to the left and right of ml_j and mr_j. We can view the *center* as $m_j = (1/2)(ml_j + mr_j)$

$$a_j(x) = \begin{cases} 1 - \frac{ml_j - x}{l_j}, & \text{if } ml_j - l_j \le x \le ml_j \\ 1, & \text{if } ml_j \le x \le mr_j \\ 1 - \frac{x - mr_j}{r_j}, & \text{if } mr_j < x \le mr_j + r_j \\ 0, & \text{else} \end{cases} \tag{9}$$

3) *Clipped-parabola (Quadratic) set function.* A clipped-parabola set function (or quadratic set function) centered at m_j and with "width" d_j has the form

$$a_j(x) = \begin{cases} 1 - \left(\frac{x - m_j}{d_j}\right)^2, & \text{if } \left(\frac{x - m_j}{d_j}\right)^2 < 1 \\ 0, & \text{else} \end{cases} \quad (10)$$

This quadratic set function differs from the quadratic set function in [26].

4) *Gaussian set function.* The Gaussian set function depends on the mean m_j and standard deviation $d_j/\sqrt{2}$

$$a_j(x) = \exp\left\{-\left(\frac{x - m_j}{d_j}\right)^2\right\}. \quad (11)$$

5) *Cauchy set function.* The Cauchy set function is a bell curve with thicker tails than the Gaussian bell curve and with infinite variance and higher order moments [5]

$$a_j(x) = \frac{1}{1 + \left(\frac{x-m_j}{d_j}\right)^2}. \quad (12)$$

6) *Laplace set function.* The Laplace set function is an exponential curve

$$a_j(x) = \exp\left\{-\frac{|x - m_j|}{d_j}\right\} \quad (13)$$

where m_j is the center and $d_j > 0$ picks the decay rate of the curve.

7) *Sinc set function.* We define the sinc set function centered at m_j and *width* $d_j > 0$ as

$$a_j(x) = \sin\left(\frac{x - m_j}{d_j}\right) \Big/ \left(\frac{x - m_j}{d_j}\right). \quad (14)$$

The sinc set function is a map $a_j : R \to [-0.217, 1]$. So the denominator of a sinc SAM can in theory become zero or negative. The system design must take care when these negative set values enter the SAM ratio in (6). We set a logic flag to check if the denominator is zero or negative.

8) *Logistic set function.* The logistic or sigmoid function has the form of $S_j(x) = 1/(1 + \exp\{-x\})$. We define a symmetric logistic set function centered at m_j with *width* $d_j > 0$ as

$$a_j(x) = 2S\left(-\left(\frac{x - m_j}{d_j}\right)^2\right) = \frac{2}{1 + e^{(\frac{x-m_j}{d_j})^2}}. \quad (15)$$

The factor 2 gives $\max_{x\in R} a_j(x) = 1$.

9) *Hyperbolic tangent set function.* This set function has the form

$$a_j(x) = 1 + \tanh\left(-\left(\frac{x - m_j}{d_j}\right)^2\right) \quad (16)$$

where m_j and d_j define the center and the width of the bell curve.

10) *Hyperbolic secant set function.* Again m_j and $d_j > 0$ define the center and width of this scalar set function

$$a_j(x) = \operatorname{sech}\left(\frac{x - m_j}{d_j}\right). \quad (17)$$

11) *Differential logistic set function.* The derivative of the logistic function is a bell curve form of probability density function. $S'(x) = S(x)(1 - S(x))$ holds for a logistic function $S(x) = 1/(1 + \exp\{-x\})$. So we define this new set function as

$$a_j(x) = 4S\left(\frac{x - m_j}{d_j}\right)\left[1 - S\left(\frac{x - m_j}{d_j}\right)\right]. \quad (18)$$

The factor 4 gives $\max_{x\in R} a_j(x) = 1$.

12) *Difference logistic set function.* The logistic or sigmoid function with *steepness* $\alpha_j > 0$ has the form of $S_j(x) = 1/(1+\exp\{-\alpha_j x\})$. We define a symmetric logistic set function centered at m_j with *width* $l_j > 0$ as

$$a_j(x) = \frac{1}{D_j}[S_j(x - m_j + l_j) - S_j(x - m_j - l_j)]. \quad (19)$$

The normalizer $D_j = S_j(l_j) - S_j(-l_j)$ ensures that $\max_{x\in R} a_j(x) = 1$.

13) *Difference hyperbolic tangent set function.* This new set function has the difference form

$$a_j(x) = \frac{1}{D_j}\left[\tanh\left(\frac{x - m_j + l_j}{d_j}\right) - \tanh\left(\frac{x - m_j - l_j}{d_j}\right)\right]. \quad (20)$$

This results in a bell curve. The term $l_j > 0$ defines the "width" of the function and $D_j = 2\tanh(l_j/d_j)$ gives the normalization factor.

Fig. 5 plots the scalar set functions for sample choices of parameters. Simulations in Section VI compare how these scalar set functions perform in adaptive fuzzy function approximation in terms of squared error.

B. Joint Factorable Sets: Product Set Functions

This class includes joint set functions $a_j : R^n \to [0, 1]$ that factor $a_j(x) = g(a_j^1(x_1), \ldots, a_j^n(x_n))$ for some function $g : [0, 1]^n \to [0, 1]$. The popular factorable joint set functions combine the scalar set functions with product

$$a_j(x) = a_j^1(x_1) \times \cdots \times a_j^n(x_n) \quad (21)$$

or other t-norms such as min

$$a_j(x) = \min\left(a_j^1(x_1), \ldots, a_j^n(x_n)\right) \quad (22)$$

for scalar set functions $a_j^i : R \to [0, 1]$. We form the product set functions from scalar set functions in Section IV-A as in Fig. 6. Section VI compares the results of adaptive function approximation of these set functions for two- and three-input cases.

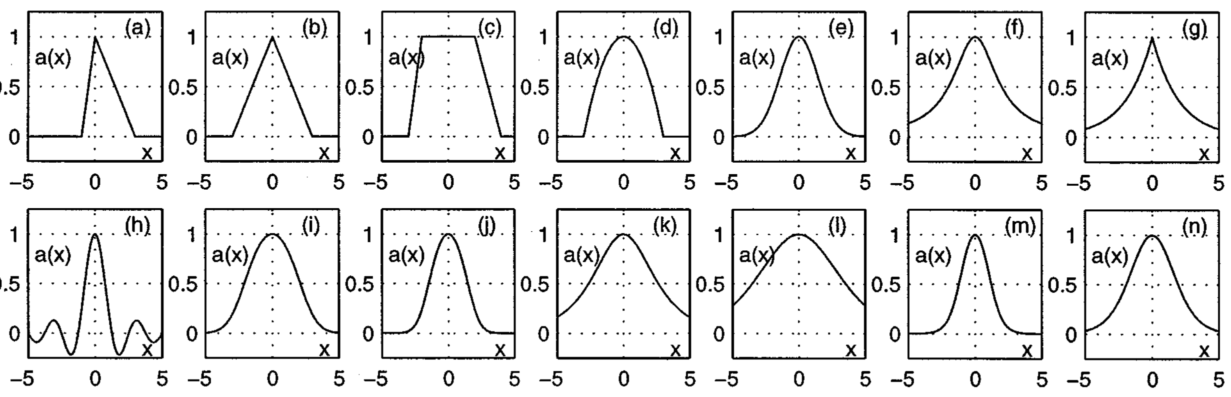


Fig. 5. Set functions centered at $m = 0$. (a) Triangle: $l = 1$ and $r = 3$. (b) Symmetric triangle: $d = 3$. (c) Trapezoid: $l = 1, ml = -2, mr = 2$, and $r = 2$.(d) Parabola: $d = 2$. (e) Gaussian: $d = 2$. (f) Cauchy: $d = 2$. (g) Laplace: $d = 2$. (h) Sinc: $d = 0.4$. (i) Logistic: $d = 2$. (j) Hyperbolic Tangent: $d = 2$. (k) Hyperbolic secant: $d = 2$. (l) Differential logistic: $d = 2$. (m) Difference logistic: $\alpha = 2$ and $l = 1$. (n) Difference hyperbolic tangent: $d = 2$ and $l = 1$.

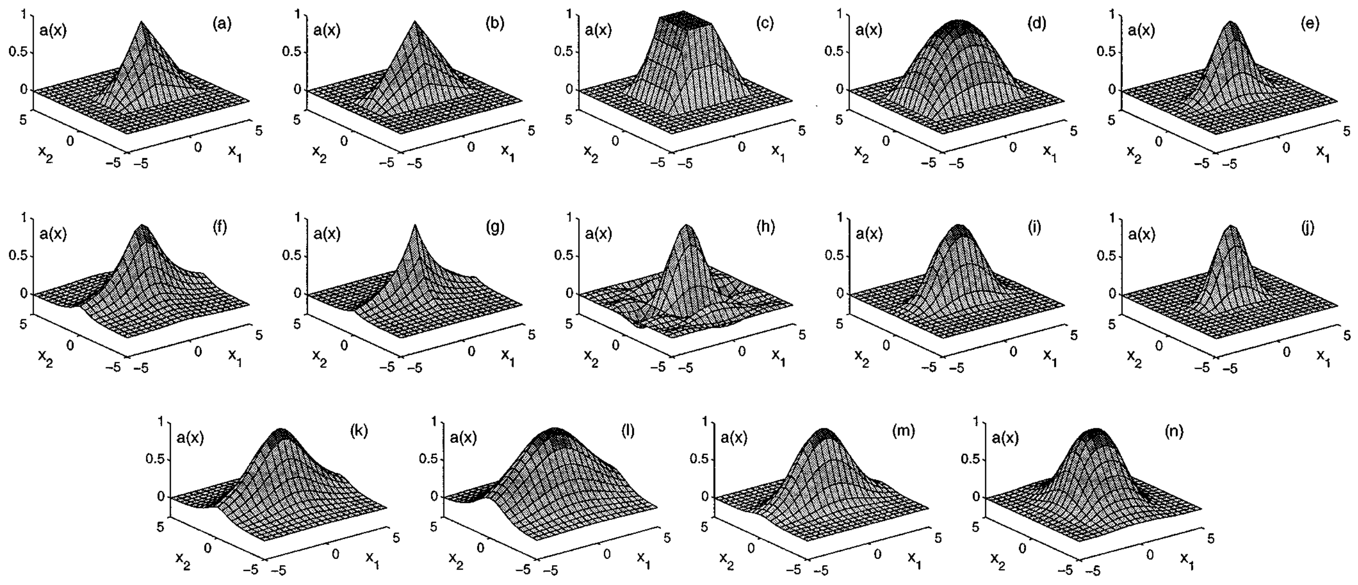


Fig. 6. Product joint set functions centered at $m = 0$. (a) Triangle: $a^1 \sim (0, 3, 4)$ and $a^2 \sim (0, 2, 2)$. (b) Symmetric triangle: $d^1 = 4$ and $d^2 = 2$. (c) Trapezoid: $a^1 \sim (2, -1, 1, 3)$ and $a^2 \sim (1, -1.5, 1.5, 2)$. (d) Parabola: $d^1 = 4$ and $d^2 = 3$. (e) Gaussian: $d^1 = 2$ and $d^2 = 1$. (f) Cauchy: $d^1 = 2$ and $d^2 = 1$. (g) Laplace: $d^1 = 2$ and $d^2 = 1$. (h) Sinc: $d^1 = 0.8$ and $d^2 = 0.4$. (i) Logistic: $d^1 = 2$ and $d^2 = 1$. (j) Hyperbolic tangent: $d^1 = 2$ and $d^2 = 1$. (k) Hyperbolic secant: $d^1 = 2$ and $d^2 = 1$. (l) Differential logistic: $d^1 = 2$ and $d^2 = 1$. (m) Difference logistic: $\alpha^1 = 1, l^1 = 2, \alpha^2 = 2$, and $l^2 = 1$. (n) Difference hyperbolic tangent: $d^1 = 1, l^1 = 2, d^2 = 2$, and $l^2 = 1$.

V. Supervised Learning in SAMs: Scalar and Product Sets

Supervised gradient descent can tune all the parameters in the SAM model (6) [12], [14]. A gradient descent learning law for a SAM parameter ξ has the form

$$\xi(t+1) = \xi(t) - \mu_t \frac{\partial E}{\partial \xi} \tag{23}$$

where μ_t is a learning rate at iteration t. We seek to minimize the squared error

$$E(x) = \frac{1}{2}(f(x) - F(x))^2 \tag{24}$$

of the function approximation. The vector function $f : R^n \rightarrow R^p$ has components $f(x) = (f_1(x), \ldots, f_p(x))^T$ and so does the vector function F. We consider the case when $p = 1$. A general form for multiple output when $p > 1$ expands the error function $E(x) = \|f(x) - F(x)\|$ for some norm $\|\cdot\|$. Let ξ_j^k denote the kth parameter in the set function a_j. Then the chain rule gives the gradient of the error function with respect to the if-part set parameter ξ_j^k with respect to the then-part set centroid $c_j = (c_j^i, \ldots, c_j^p)^T$ and with respect to the then-part set volume V_j

$$\frac{\partial E}{\partial \xi_k^j} = \frac{\partial E}{\partial F}\frac{\partial F}{\partial a_j}\frac{\partial a_j}{\partial \xi_k^j}, \quad \frac{\partial E}{\partial c_j} = \frac{\partial E}{\partial F}\frac{\partial F}{\partial c_j}, \quad \text{and}$$
$$\frac{\partial E}{\partial V_j} = \frac{\partial E}{\partial F}\frac{\partial F}{\partial V_j} \tag{25}$$

where

$$\frac{\partial E}{\partial F} = -(f(x) - F(x)) = -\varepsilon(x) \tag{26}$$

$$\frac{\partial F}{\partial a_j} = \frac{\left(\sum_{i=1}^m a_i(x)V_i\right)(V_j c_j) - V_j\left(\sum_{i=1}^m a_i(x)V_i c_i\right)}{\left(\sum_{i=1}^m a_i(x)V_i\right)^2}$$
$$= \frac{[c_j - F(x)]V_j}{\sum_{i=1}^m a_i(x)V_i} = [c_j - F(x)]\frac{p_j(x)}{a_j(x)}. \tag{27}$$

The SAM ratios (6) with equal rule weights $w_1 = w_2 = \cdots = w_m = w$ give [12], [14]

$$\frac{\partial F}{\partial c_j} = \frac{a_j(x)V_j}{\sum_{i=1}^m a_i(x)V_i} = p_j(x) \tag{28}$$

$$\frac{\partial F}{\partial V_j} = \frac{a_j(x)[c_j - F(x)]}{\sum_{i=1}^m a_i(x)V_i} = [c_j - F(x)]\frac{p_j(x)}{V_j}. \tag{29}$$

Then the learning laws for the then-part set centroids c_j and volumes V_j have the final form

$$c_j(t+1) = c_j(t) + \mu_t \varepsilon(x) p_j(x) \tag{30}$$

$$V_j(t+1) = V_j(t) + \mu_t \varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{V_j}. \tag{31}$$

The learning laws for the if-part set parameters follow in like manner for both scalar and joint sets as we show below.

We first derive learning laws for parameters of the scalar if-part set functions. Each set function a_j gives different partial derivatives of a_j with respect to its kth parameter ξ_k^j in (25). The learning laws for the parameters of each scalar set functions are as follows.

1) Triangle set function

$$m_j(t+1) = \begin{cases} m_j(t) - \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{1}{l_j}, \\ \quad \text{if } m_j - l_j < x < m_j \\ m_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{1}{r_j}, \\ \quad \text{if } m_j < x < m_j + r_j \\ m_j(t), \quad \text{else} \end{cases} \tag{32}$$

$$l_j(t+1) = \begin{cases} l_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{m_j - x}{l_j^2}, \\ \quad \text{if } m_j - l_j < x < m_j \\ l_j(t), \quad \text{else} \end{cases} \tag{33}$$

$$r_j(t+1) = \begin{cases} r_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{x - m_j}{r_j^2}, \\ \quad \text{if } m_j < x < m_j + r_j \\ r_j(t), \quad \text{else.} \end{cases} \tag{34}$$

2) Trapezoid set function

$$ml_j(t+1) = \begin{cases} ml_j(t) - \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{1}{l_j}, \\ \quad \text{if } ml_j - l_j < x < ml_j \\ ml_j(t), \quad \text{else} \end{cases} \tag{35}$$

$$mr_j(t+1) = \begin{cases} mr_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{1}{r_j}, \\ \quad \text{if } mr_j < x < mr_j + r_j \\ mr_j(t), \quad \text{else} \end{cases} \tag{36}$$

$$l_j(t+1) = \begin{cases} l_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{ml_j - x}{l_j^2}, \\ \quad \text{if } ml_j - l_j < x < ml_j \\ l_j(t), \quad \text{else} \end{cases} \tag{37}$$

$$r_j(t+1) = \begin{cases} r_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{x - mr_j}{r_j^2}, \\ \quad \text{if } mr_j < x < mr_j + r_j \\ r_j(t), \quad \text{else.} \end{cases} \tag{38}$$

3) Clipped-parabola set function

$$m_j(t+1) = \begin{cases} m_j(t) + 2\mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{x - m_j}{d_j^2}, \\ \quad \text{if } \left(\frac{x - m_j}{d_j}\right)^2 < 1 \\ m_j(t), \quad \text{else} \end{cases} \tag{39}$$

$$d_j(t+1) = \begin{cases} d_j(t) + 2\mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)}\frac{(x - m_j)^2}{d_j^3}, \\ \quad \text{if } \left(\frac{x - m_j}{d_j}\right)^2 < 1 \\ d_j(t), \quad \text{else.} \end{cases} \tag{40}$$

4) Gaussian set function

$$m_j(t+1) = m_j(t) + 2\mu_t\varepsilon(x)p_j(x)[c_j - F(x)]\frac{x - m_j}{d_j^2} \tag{41}$$

$$d_j(t+1) = d_j(t)2\mu_t\varepsilon(x)p_j(x)[c_j - F(x)]\frac{(x - m_j)^2}{d_j^3}. \tag{42}$$

5) Cauchy set function

$$m_j(t+1) = m_j(t) + 2\mu_t\varepsilon(x)p_j(x)[c_j - F(x)]\frac{x - m_j}{d_j^2}a_j(x) \tag{43}$$

$$d_j(t+1) = d_j(t) + 2\mu_t\varepsilon(x)p_j(x)[c_j - F(x)]\frac{(x - m_j)^2}{d_j^3} \times a_j(x). \tag{44}$$

6) Laplace set function

$$m_j(t+1) = m_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]p_j(x) \times \text{sign}(x - m_j)\frac{1}{|d_j|} \tag{45}$$

$$d_j(t+1) = d_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]p_j(x) \times \text{sign}(d_j)\frac{|x - m_j|}{d_j^2}. \tag{46}$$

7) Sinc set function

$$m_j(t+1) = m_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)} \times \left(a_j(x) - \cos\left(\frac{x - m_j}{d_j}\right)\right)\frac{1}{x - m_j} \tag{47}$$

$$d_j(t+1) = d_j(t) + \mu_t\varepsilon(x)[c_j - F(x)]\frac{p_j(x)}{a_j(x)} \times \left(a_j(x) - \cos\left(\frac{x - m_j}{d_j}\right)\right)\frac{1}{d_j}. \tag{48}$$

8) Logistic set function

$$m_j(t+1) = m_j(t) + \mu_t\varepsilon(x)p_j(x)[c_j - F(x)] \times (2 - a_j(x))\frac{x - m_j}{d_j^2} \tag{49}$$

$$d_j(t+1) = d_j(t) + \mu_t\varepsilon(x)p_j(x)[c_j - F(x)] \times (2 - a_j(x))\frac{(x - m_j)^2}{d_j^3}. \tag{50}$$

9) Hyperbolic tangent set function

$$m_j(t+1) = m_j(t) + 2\mu_t\varepsilon(x)p_j(x)[c_j - F(x)] \times (2 - a_j(x))\frac{x - m_j}{d_j^2} \tag{51}$$

$$d_j(t+1) = d_j(t) + 2\mu_t\varepsilon(x)p_j(x)[c_j - F(x)] \times (2 - a_j(x))\frac{(x - m_j)^2}{d_j^3}. \tag{52}$$

10) Hyperbolic secant set function

$$m_j(t+1) = m_j(t) + \mu_t \varepsilon(x) p_j(x)[c_j - F(x)] \times \frac{1}{d_j} \tanh\left(\frac{x-m_j}{d_j}\right) \quad (53)$$

$$d_j(t+1) = d_j(t) + \mu_t \varepsilon(x) p_j(x)[c_j - F(x)] \times \frac{x-m_j}{(d_j)^2} \tanh\left(\frac{x-m_j}{d_j}\right). \quad (54)$$

11) Differential logistic set function

$$m_j(t+1) = m_j(t) + \mu_t \varepsilon(x) p_j(x)[c_j - F(x)] \times \frac{1}{d_j}\left[1 - 2S\left(\frac{x-m_j}{d_j}\right)\right] \quad (55)$$

$$d_j(t+1) = m_j(t) + \mu_t \varepsilon(x) p_j(x)[c_j - F(x)] \times \frac{x-m_j}{d_j^2}\left[1 - 2S\left(\frac{x-m_j}{d_j}\right)\right]. \quad (56)$$

12) Difference logistic set function

$$\begin{aligned} m_j(t+1) &= m_j(t) + \mu_t \varepsilon(x) p_j(x)[c_j - F(x)] \\ &\quad \times \alpha_j[S_j(x - m_j + l_j) + S_j(x - m_j - l_j) - 1] \end{aligned} \quad (57)$$

$$\begin{aligned} \alpha_j(t+1) &= \alpha_j(t) + \mu_t \varepsilon(x) \frac{p_j(x)}{a_j(x)}[c_j - F(x)] \frac{1}{D_j}[[x - m_j + l_j] \\ &\quad \times S_j(x - m_j + l_j)[1 - S_j(x - m_j + l_j)] \\ &\quad - [x - m_j - l_j] S_j(x - m_j - l_j)[1 - S_j(x - m_j - l_j)] \\ &\quad - l_j a_j(x)(S_j(l_j)[1 - S_j(l_j)] + S_j(-l_j)[1 - S_j(-l_j)])] \end{aligned} \quad (58)$$

$$\begin{aligned} l_j(t+1) &= l_j(t) + \mu_t \varepsilon(x)[c_j - F(x)] \frac{p_j(x)}{a_j(x)} \frac{\alpha_j}{D_j} \\ &\quad \times [S_j(x - m_j + l_j)[1 - S_j(x - m_j + l_j)] \\ &\quad + S_j(x - m_j - l_j)[1 - S_j(x - m_j - l_j)] \\ &\quad - a_j(x)(S_j(l_j)[1 - S_j(l_j)] + S_j(-l_j)[1 - S_j(-l_j)])]. \end{aligned} \quad (59)$$

13) Difference hyperbolic tangent set function

$$\begin{aligned} m_j(t+1) &= m_j(t) + \mu_t \varepsilon(x) p_j(x) \frac{c_j - F(x)}{d_j} \\ &\quad \times \left[\tanh\left(\frac{x - m_j + l_j}{d_j}\right) + \tanh\left(\frac{x - m_j - l_j}{d_j}\right)\right] \end{aligned} \quad (60)$$

$$\begin{aligned} d_j(t+1) &= d_j(t) + \mu_t \varepsilon(x) \frac{p_j(x)}{a_j(x)}[c_j - F(x)] \frac{1}{D_j d_j} \\ &\quad \times \left[\frac{x - m_j + l_j}{d_j} \tanh^2\left(\frac{x - m_j + l_j}{d_j}\right)\right. \\ &\quad - \frac{x - m_j - l_j}{d_j} \tanh^2\left(\frac{x - m_j - l_j}{d_j}\right) \\ &\quad \left. - \frac{2l_j}{d_j} + 2\frac{l_j}{d_j} a_j(x)\left[1 - \tanh^2\left(\frac{l_j}{d_j}\right)\right]\right] \end{aligned} \quad (61)$$

$$\begin{aligned} l_j(t+1) &= l_j(t) + \mu_t \varepsilon(x) \frac{p_j(x)}{a_j(x)}[c_j - F(x)] \frac{1}{D_j d_j} \\ &\quad \times \left[2 - \tanh^2\left(\frac{x - m_j + l_j}{d_j}\right) - \tanh^2\left(\frac{x - m_j - l_j}{d_j}\right)\right. \\ &\quad \left. - 2a_j(x)\left[1 - \tanh^2\left(\frac{l_j}{d_j}\right)\right]\right]. \end{aligned} \quad (62)$$

We also can approximate the learning laws for the symmetric triangle and trapezoid set functions with Gaussian learning laws for their centers and the widths. Like results hold for the learning laws of factorable n-D set functions. A factored set function $a_j(x) = a_j^1(x_1) \ldots a_j^n(x_n)$ leads to a new form for the error gradient. The gradient with respect to the parameter m_j^k of the jth set function a_j has the form

$$\frac{\partial E}{\partial m_j^k} = \frac{\partial E}{\partial F} \frac{\partial F}{\partial a_j} \frac{\partial a_j}{\partial a_j^k} \frac{\partial a_j^k}{\partial m_j^k} \quad (63)$$

where

$$\frac{\partial a_j}{\partial a_j^k} = \prod_{i \neq k}^{n} a_j^i(x_i) = \frac{a_j(x)}{a_j^k(x_k)}. \quad (64)$$

VI. Simulation Results I: Scalar and Joint Product Sets

We trained the SAMs with different set functions to approximate different functions. We scored each test in terms of the squared error (SE) of the function approximation for a constant learning rate μ.

We uniformly sampled 201 points of the function in the one-dimensional (1-D) case to give a training set. The 2-D case used $31 \times 31 = 961$ samples. The 3-D case used $20 \times 20 \times 20 = 8000$ samples. One epoch passed all 201, 961, or 8000 samples through the SAM to train it.

We then finely sampled the function to obtain the test data for each function. So the training data set and the test data set are different but do overlap due to the sampling pattern. The one-input cases used 241 samples, the two-input cases used $51 \times 51 = 2601$ samples and the three-input cases used $31 \times 31 \times 31 = 29\,791$ samples to test how well fuzzy systems approximate the approximands.

The 1-D SAMs used 12 rules, while the 2-D SAMs used 64 rules. The 3-D SAMs used 125 rules. Different initializations led to convergence to different local minima of the SE surface. There is no formal way to find the initial conditions that lead to the global minimum, so we had to guess at them. We spread rule patches uniformly along the input space. So we spread the if-part set centers m_j uniformly along the x-axis. We picked the then-part set centroids c_j as the values of the sampled approximand f at m_j: $c_j = f(m_j)$. We set the then-part volumes (areas) to unity at first: $V_1 = \cdots = V_m = 1$. Then supervised learning tuned each SAM parameter.

We used a constant learning rate μ throughout each training session. We also tried different learning rates to see whether the system converged to different solutions and picked the best results as a representative for that case. But at each try the learning rates for each parameter were the same. The learning rates were

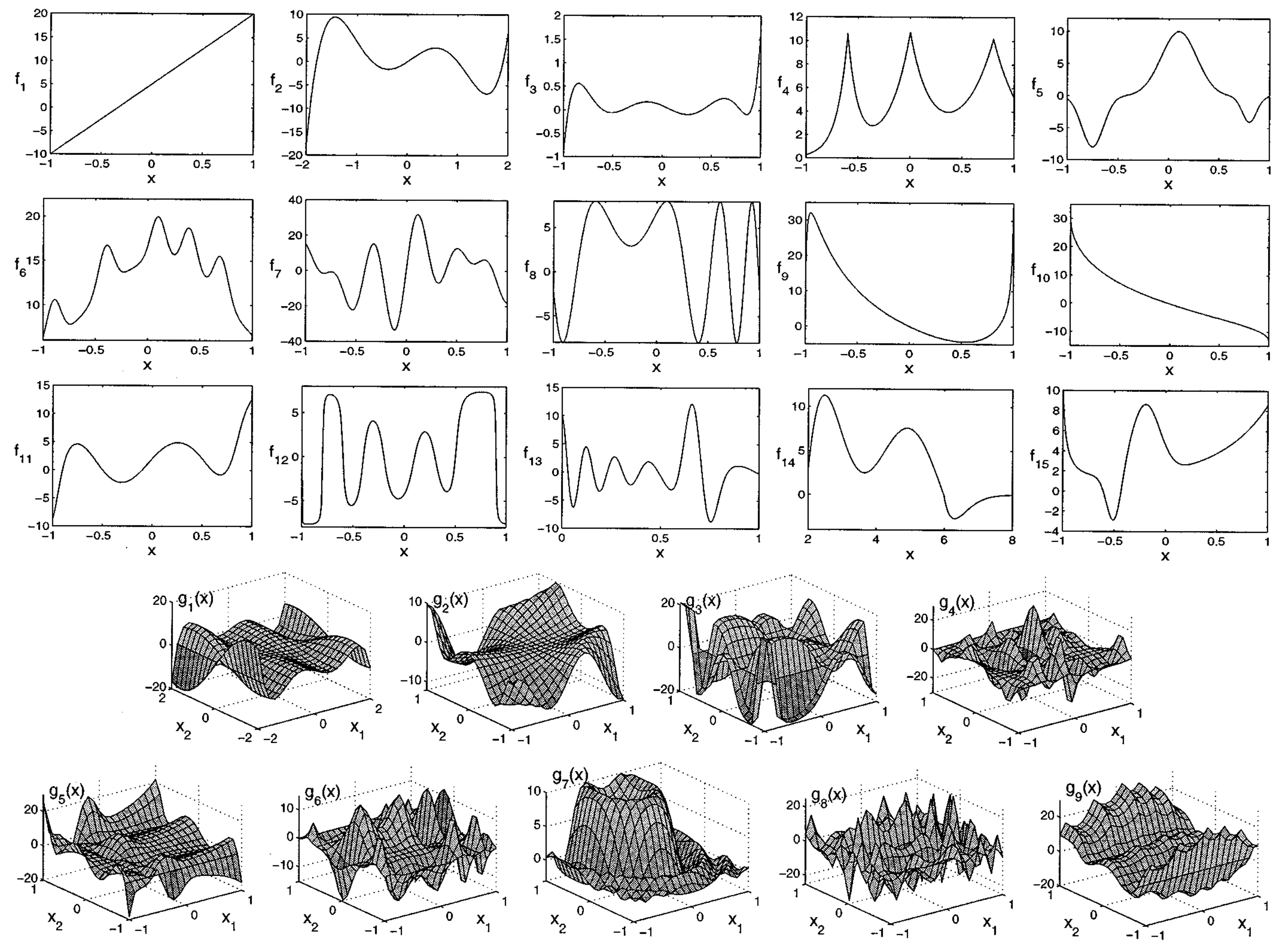


Fig. 7. Samples of 1-D and 2-D test approximands.

small because each learning law is highly nonlinear–else the learning might not have converged. The learning rates that we used ranged from $\mu = 10^{-8}$ to 10^{-4}. We compared the results for each learning rates and picked the best ones. Below we list test functions we used as approximands.

A. 1-D Test Functions

We defined functions of one variable $f : X \subset R \to R$ to test the scalar fuzzy sets in the SAM models. We also used functions from the literature [1], [7]. We roughly classify the test functions that we used and list some of them as follow.

1) Polynomial and Rational Functions: This class of approximands consisted of polynomial functions and rational functions of different degrees. The two simplest functions in this class are a constant function and a straight line function. We do not list constant functions here because we can represent any constant function with any kind of fuzzy system with only one rule. We did include a straight line function in our test case (see Fig. 7). The test functions were as follows:

$$f_1(x) = 15x + 5 \quad \text{for } x \in [-1, 1] \tag{65}$$

$$f_2(x) = 3x(x-1)(x-1.9)(x-0.7)(x+1.8) \quad \text{for } x \in [-2, 2] \tag{66}$$

$$f_3(x) = \frac{\begin{pmatrix} 100(x+0.95)(x+0.6)(x+0.4) \\ \times(x-0.1)(x-0.4)(x-0.8)(x-0.9) \end{pmatrix}}{(x+1.7)(x-2)^2} \quad \text{for } x \in [-1, 1]. \tag{67}$$

2) Exponential Functions: This class of set function includes Gaussian bell-curve and Laplace functions. The hyperbolic tangent is one form of ratio of exponential functions. We tested the approximands below on the interval $x \in [-1, 1]$

$$f_4(x) = 10\left(e^{-\frac{|x|}{0.2}} + e^{-\frac{|x-0.8|}{0.3}} + e^{-\frac{|x+0.6|}{0.1}}\right) \tag{68}$$

$$f_5(x) = 10e^{-(\frac{x-0.1}{0.25})^2} - 8e^{-(\frac{x+0.75}{0.15})^2} - 4e^{-(\frac{x-0.8}{0.1})^2} \tag{69}$$

$$f_6(x) = 15e^{-(x-0.1)^2} + 5\left[e^{-(\frac{x-0.1}{0.1})^2} + e^{-(\frac{x-0.4}{0.1})^2} + e^{-(\frac{x-0.7}{0.1})^2} + e^{-(\frac{x+0.4}{0.1})^2} + e^{-(\frac{x+0.9}{0.1})^2}\right]. \tag{70}$$

3) Polynomials Based on Trigonometric Functions: This class of functions includes many functions. A truncated Fourier expansion of any function belongs to this class. We also include the inverse of these trigonometric functions within this class of test cases. All of the functions have as their domain the set $X = [-1, 1]$

$$f_7(x) = 10[\sin(4x + 0.1) + \sin(14x) + \sin(11x - 0.2) + \sin(17x + 0.3)] \tag{71}$$

$$f_8(x) = 8\sin(10x^2 + 5x + 1) \tag{72}$$

$$f_9(x) = 0.01\tan^3(1.5x) + 10\tan^2(x) - 20\tan(0.7x) \tag{73}$$

$$f_{10}(x) = \arccos^3(x) - \arccos^2(-x) - \arccos(-x) \tag{74}$$

$$f_{11}(x) = 10\tan^{-1}[10(x + 0.9)(x + 0.5) \times (x + 0.1)(x - 0.6)(x - 0.75)] \tag{75}$$

$$f_{12}(x) = 5\tan^{-1}\left(\frac{\begin{array}{c}2000(x - .1)(x - .3)(x - .5)\\ \times(x - .9)(x - 1.1)(x + .2)\\ \times(x + .4)(x + .6)(x + .8)(x + 1)\end{array}}{x^2 + 1.5x + 1}\right). \tag{76}$$

4) Combination of Exponential, Rational, and Trigonometric Functions: We formed a mixed class of functions from the above classes. A sinc function $\sin x/x$ also belongs to this class because it is a rational function of trigonometric and polynomial functions

$$f_{13}(x) = 1 + 10e^{-100(x-0.7)^2}\frac{\sin\left(\frac{125}{x+1.5}\right)}{x + 0.1} \quad \text{for } x \in [0, 1] \tag{77}$$

$$f_{14}(x) = \begin{cases} \frac{10-x}{8}\left(\frac{(x-2.5)^2(x-5)^2(x-9)^3x^3}{400+200(x-0.8)^2} + 12\right) \\ \quad \text{for } 2 \le x < 6 \\ e^{-3(x-6)}\left(\frac{0.005(x-2.5)^2(x-5)^2(x-9)^3x^3}{400+200(x-0.8)^2} + 12\right) \\ \quad \text{for } 6 \le x \le 8 \end{cases} \tag{78}$$

$$f_{15}(x) = \frac{1}{x^3 + 1.1} - 5e^{-\left(\frac{x+0.5}{0.1}\right)^2} + 7e^{-\left(\frac{x+0.2}{0.2}\right)^2} + 2e^{-2(x-0.3)} \quad \text{for } x \in [-1, 1]. \tag{79}$$

Fig. 7 plots some of the 1-D approximands.

B. 2-D Test Functions

We created 2-D test functions $f : X \subset R^2 \to R$ from the 1-D test functions. A product of two 1-D functions created 2-D test functions. We also defined new 2-D set functions that were unfactorable. Below we list some samples of the approximands that we tested. All test functions have as their domain the set $X = [-1, 1]\times = [-1, 1] \times [-1, 1]$ except for the test function g_1

$$g_1(x_1, x_2) = 3x_1(x_1 - 1)(x_1 - 1.9)(x_1 + 0.7) \times (x_1 + 1.8)\sin(x_2) \quad \text{for } -2 \le x_1, x_2 \le 2 \tag{80}$$

$$g_2(x_1, x_2) = 5x_1^2x_2 + 2x_1^2 - 3x_2^2 + 6\sin(5x_1x_2^2) \tag{81}$$

$$g_3(x_1, x_2) = 10\tan^{-1}\left(\frac{10(x_1 - 0.2)(x_1 - 0.7)(x_1 + 0.8)}{x_1 + 1.4}\right) \times \tan^{-1}\left(\frac{\begin{array}{c}10(x_2 - 0.2)(x_2 + 0.8)(x_2 - 0.7)\\ \times(x_2 + 0.2)(x_2 - 1.5)\end{array}}{(x_2 + 1.4)(x_2 - 1.1)x_2(x_2 + 0.3) + 0.7}\right) \tag{82}$$

$$g_4(x_1, x_2) = \frac{1}{10}f_7(x_1)f_5(x_2) \tag{83}$$

$$g_5(x_1, x_2) = \frac{1}{5}f_{15}(x_1)f_{11}(x_2) \tag{84}$$

$$g_6(x_1, x_2) = \frac{1}{5}f_8(x_1)f_5(x_2) \tag{85}$$

$$g_7(x_1, x_2) = 10\frac{\sin(10x_1^2 + 5x_2^2 - 6x_2)}{10x_1^2 + 5x_2^2 - 6x_2} \tag{86}$$

$$g_8(x_1, x_2) = \frac{1}{10}f_7(x_1)f_8(x_2) \tag{87}$$

$$g_9(x_1, x_2) = f_6(x_1)\tan^{-1}(10(x_2 + 0.8) \times (x_2 + 0.3)(x_2 - 0.4)(2x_2 - 0.7)). \tag{88}$$

Fig. 7 plots the surface of some of these samples of 2-D approximands.

C. 3-D Test Functions

We created 3-D test functions $f : X \subset R^3 \to R$ as products of 1-D test functions. We also define new 3-D set functions that were unfactorable. All test functions have as their domain the set $X = [-1, 1] \times [-1, 1] \times [-1, 1]$. Below we list some samples of the approximands that we tested

$$h_1(x_1, x_2, x_3) = 60x_1(x_1 - 0.5)(x_1 - 0.95)(x_1 + 0.35) \times (x_1 + 0.9)\left(3\sin(6x_2x_3) + 6\tan^{-1}(4x_2^2) \times \tan^{-1}(3x_3)\tan^{-1}(2x_2x_3^2) - 5x_2^2x_3\right) \tag{89}$$

$$h_2(x_1, x_2, x_3) = \frac{1}{200}f_15(x_1)f_9(x_2)f_{10}(x_3) \tag{90}$$

$$h_3(x_1, x_2, x_3) = \frac{1}{10}f_3(x_1)f_5(x_2)f_{12}(x_3) \tag{91}$$

$$h_4(x_1, x_2, x_3) = \left(1 + 10e^{-100(0.5x_1+0.3)^2}\frac{\sin\left(\frac{125}{0.5x_1+2}\right)}{5x_1 + 6}\right) \times f_6(x_2)f_3(x_3) \tag{92}$$

$$h_5(x_1, x_2, x_3) = e^{-(x_1x_2-0.7)(x_1x_3-0.5)}\frac{\sin(125/(x_1 + 1.5))}{x_2 + 1.1} + (5x_1x_2^2 - 6x_3^3)\tan^{-1}(10x_1x_2 + x_3^2). \tag{93}$$

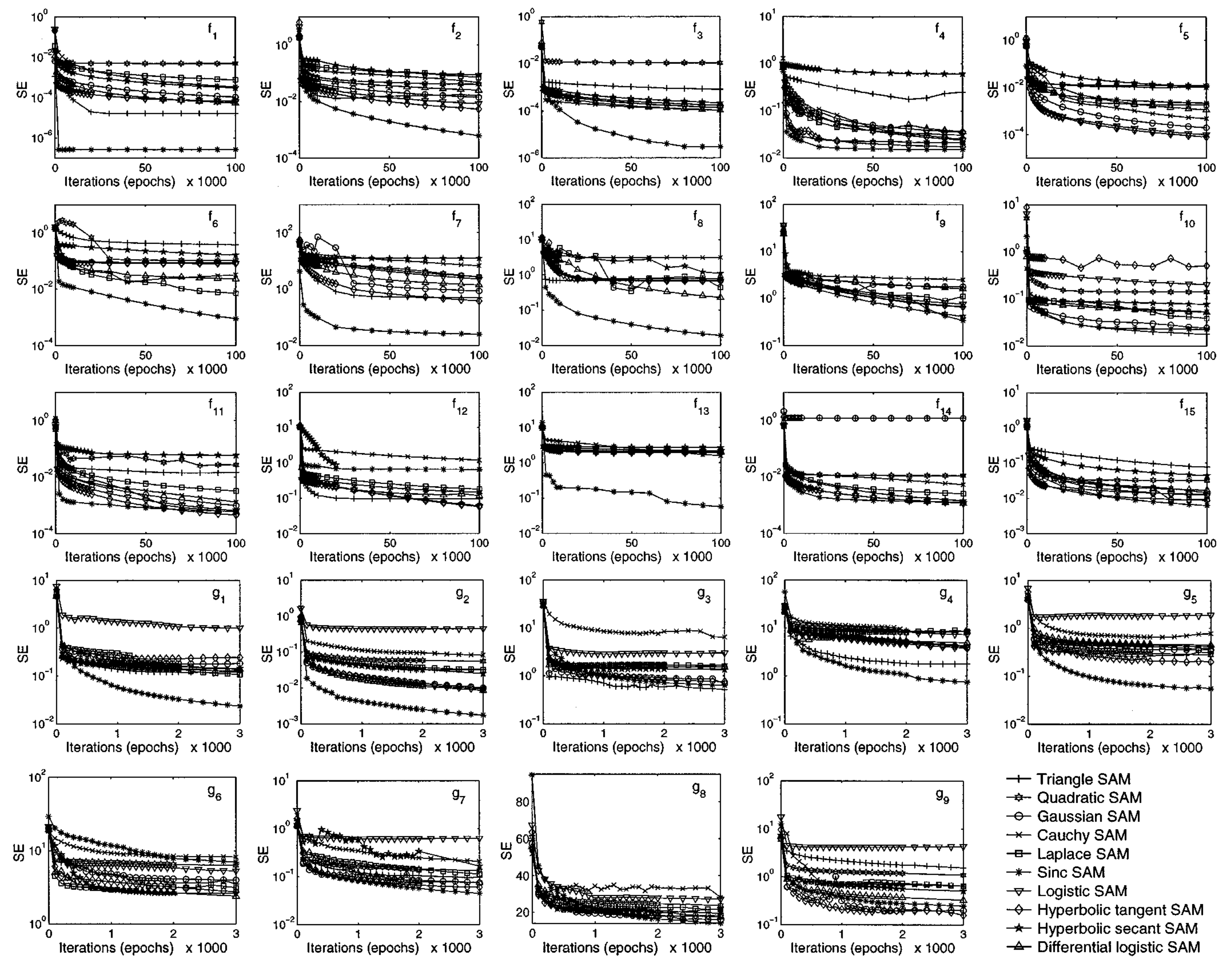


Fig. 8. Convergence plots of squared error versus iteration steps. We picked the best results from different learning rates from each set function. The approximands are 1-D and 2-D approximands in Fig. 7.

D. Results: Comparison of Squared Errors

We gave one point to the set function whose squared error (SE) was the lowest for each test approximand. In case of a tie (when their SEs are well within 20%) we gave a fraction of a point for each tying competitors. We also count as winners the set functions whose SEs lie within 20% of the lowest SE. We tested the learning laws with various learning rates (from $\mu = 10^{-8}$ to $\mu = 10^{-4}$) and also with different initial widths for set functions of bell-curve shape.

Fig. 8 plots the SEs against the number of learning cycles. The simulation results show that the sinc set function often converged faster and more accurately than did the other set functions. The 2-D and 3-D cases with factored set functions showed like patterns. The pie charts in Fig. 9 show the frequency with which each set function performed best in the test cases for the scalar sets and factorable (product) sets. Note that the sinc shape wins in one and two dimensions while it loses to Gaussian and hyperbolic tangent shapes in three dimensions.

A joint set function $a_j : R^n \rightarrow [0,1]$ measures the degree to which input $x \in R^n$ belong to the fuzzy or multivalued set $A_j \subset R^n$: $a_j(x) = \text{Degree}(x \in A_j)$. Most fuzzy systems factor the joint set function though some use distance to maintain the joint structure and thus to maintain the correlation among input components [5]. We further examine how factorable and unfactorable joint set functions affect function approximation.

VII. Joint Unfactorable Fuzzy Sets: Transformed Metrics

This section considers a class of joint set functions $a_j : R^n \rightarrow [0,1]$ that do not factor. We focus on a small class of metrical joint set functions: $a_j[x] = g(d(x; m_j, K_j)) = g(d_j(x))$ for some metric d_j and some scalar function g such as a Gaussian, triangle, or sinc set function.

We first define the metric $d_j(x) = d_j(x; m_j, K_j)$ as a quadratic form with positive definite matrix K_j

$$d_j(x)^2 = (x - m_j)^T K_j (x - m_j). \tag{94}$$

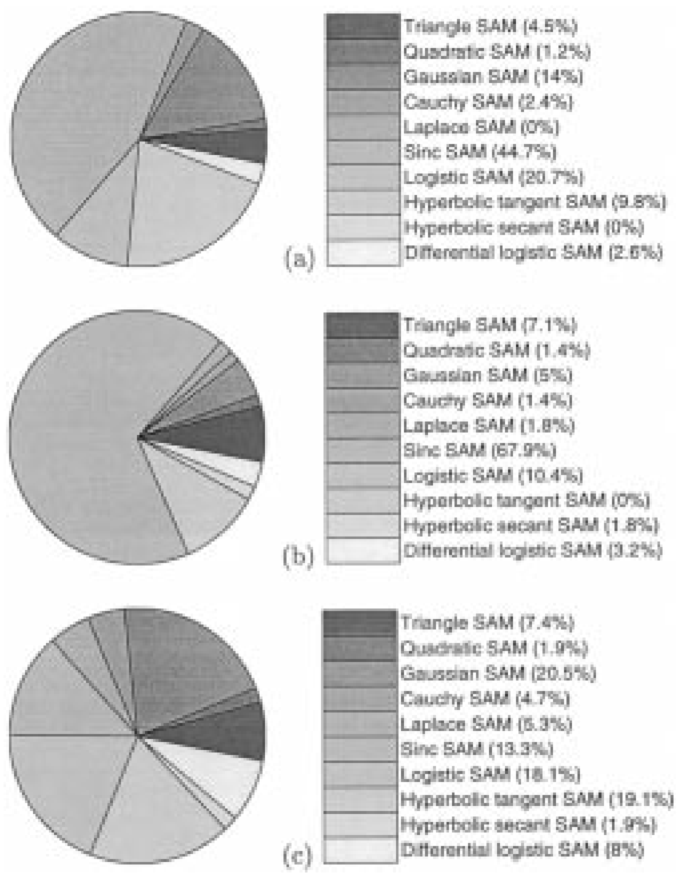


Fig. 9. Proportions of test cases where each function performed best. Multidimensional sets are factorable (product) sets of the scalar ones. The winners in each case are from the best learning rates from $\mu = 10^{-8}$ to $\mu = 10^{-4}$. (a) 1-D, (b) 2-D, and (c) 3-D test cases.

Then we can create metrical joint set functions a_j from this metric d_j and the scalar set functions g: $a_j[x] = g(d_j(x))$. Below we show the cases when g takes the form of a piecewise linear function $g(x) = ax + b$ (this gives a metrical triangle), parabolic function $g(x) = ax^2 + bx + c$, Cauchy function $g(x) = 1/(1 + x^2)$, Gaussian function $g(x) = e^{-x^2}$, Laplace function $g(x) = e^{-|x|}$, sinc function $g(x) = \sin x/x$, hyperbolic tangent $g(x) = 1 + \tanh(-x^2)$, logistic function $g(x) = 2S(-x^2)$ where $S(x) = 1/(1 + e^{-x})$, hyperbolic secant $g(x) = \operatorname{sech} x$, or the derivative of logistic function $g(x) = S'(x)$.

1) *Symmetric metrical triangle set function.* This set function defines the degree to which an input vector $x \in R^n$ belongs to set A_j with linear function

$$a_j[x] \equiv a_j(d_j(x)) = \begin{cases} 1 - d_j(x), & \text{if } d_j(x) < 1 \\ 0, & \text{else} \end{cases} \tag{95}$$

2) *Joint Gaussian set function.* This set function derives from the probability density function of a jointly normal random vector [23]

$$a_j[x] = e^{-d_j(x)^2} = e^{-(x-m_j)^T K_j (x-m_j)}. \tag{96}$$

So K_j is analogous to the inverse covariance matrix $(1/2)K^{-1}$ and m_j is analogous to the mean vector in the normalized joint Gaussian probability density [23]. The joint Gaussian set factors when the positive definite matrix K_j is diagonal.

The joint Gaussian set function has the Mahalanobis distance as its exponent if K_j^{-1} is a covariance matrix. We apply this method to scalar set functions to create metrical joint set functions below.

3) *Metrical parabolic set function.* The set value linearly falls as the square of the distance d_j grows

$$a_j[x] = \begin{cases} 1 - d_j(x)^2, & \text{if } d_j(x) < 1 \\ 0, & \text{else.} \end{cases} \tag{97}$$

4) *Joint Cauchy.* The joint Cauchy set function derives from the probability density function of joint Cauchy random variables [25]. We discard the constant that normalizes the density function to a unit integral and obtain the joint Cauchy set function

$$a_j[x] = \frac{1}{[1 + (x - m_j)^T K_j (x - m_j)]^{(n+1)/2}}. \tag{98}$$

5) *Metrical Cauchy set function.* This set function differs from the actual joint Cauchy density in (98). It has a simpler form

$$a_j[x] = \frac{1}{1 + d_j(x)^2} = \frac{1}{1 + (x - m_j)^T K_j (x - m_j)}. \tag{99}$$

6) *Metrical Laplace set function.* The scalar Laplace function forms the metrical set function as

$$a_j[x] = \exp\{-d_j(x)\} = e^{-\sqrt{(x-m_j)^T K_j (x-m_j)}}. \tag{100}$$

This metrical set function reduces to the factorable product set if the positive definite matrix K_j is diagonal.

7) *Metrical sinc set function.* The scalar sinc function forms a joint metrical set from a metric d_j as

$$a_j[x] = \frac{\sin(d_j(x))}{d_j(x)}. \tag{101}$$

8) *Metrical logistic set function.* The logistic function defines this metrical joint set as

$$a_j[x] = \frac{2}{1 + \exp\{d_j(x)^2\}}. \tag{102}$$

9) *Metrical hyperbolic tangent set function.* This metrical joint set has the form

$$a_j[x] = 1 + \tanh(-d_j(x)^2). \tag{103}$$

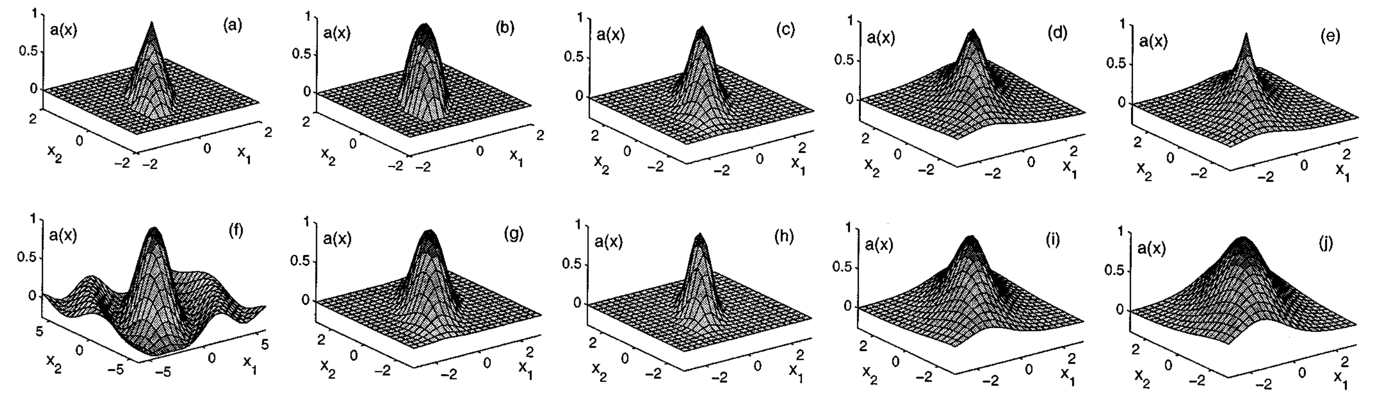


Fig. 10. Metrical joint set functions with $m = 0$ and $K = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$ and l^2 distance. (a) Symmetric metrical triangle set function. (b) Metrical parabola set function. (c) Joint (metrical) Gaussian set function. (d) Metrical Cauchy set function. (e) Metrical Laplace set function. (f) Metrical sinc set function. (g) Metrical logistic set function. (h) Metrical hyperbolic tangent set function. (i) Metrical hyperbolic secant set function. (j) Metrical differential logistic set function.

10) *Metrical hyperbolic secant set function.* We form the metrical joint set from the hyperbolic secant function as

$$a_j[x] = \operatorname{sech}(d_j(x)). \tag{104}$$

11) *Metrical differential logistic set function.* The derivative of the logistic function also defines a metrical joint set

$$a_j[x] = 4S(d_j(x))(1 - S(d_j(x))). \tag{105}$$

Fig. 10 shows some of the above joint set functions with centers at $m_j = 0$ and with $K_j = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$.

The metric d_j reduces to the weighted l^2 metric for the diagonal matrix $K_j = \operatorname{diag}((\kappa_j^1)^2, \ldots, (\kappa_j^n)^2)$ $d_j(x) = \sqrt{\sum_{i=1}^n |\kappa_j^i(x_i - m_j^i)|^2}$. So we can generalize this metrical measure to the weighted l^p metric

$$d_j^p(x) = \left[\sum_{i=1}^n \left|\kappa_j^i\left(x_i - m_j^i\right)\right|^p\right]^{\frac{1}{p}} \tag{106}$$

for $p > 0$ and use it to create joint metrical set functions. We replaced the weights κ_j^i from the diagonal matrix K with scales $1/\sigma_j^i$. So we replaced $|\kappa_j^i(x_i - m_j^i)|^p$ with $|(x_i - m_j^i)/\sigma_j^i|^p$ to conform with the form of factorable sets in Section IV-B. The l^p metrical distance has the form

$$d_j^p(x) = \left[\sum_{i=1}^n \left|\frac{x_i - m_j^i}{\sigma_j^i}\right|^p\right]^{\frac{1}{p}}. \tag{107}$$

So the l^p metrical set function a_j^p follows as

$$a_j^p(x) = g\left(d_j^p(x)\right) \tag{108}$$

for some scalar function $g : R \to R$ and for d_j^p as in (107). This gives a general form for l^p metrical sets. The real function $g : R \to R$ can be any generalized scalar set function. Popular examples of g are triangle and Gaussian functions.

We also tested the metrical sets with the l^1 or "city block" metric

$$d_j^1(x) = \sum_{i=1}^n \left|\frac{x_i - m_j^i}{\sigma_j^i}\right| \tag{109}$$

where $\sigma_j^i > 0$ in (107). The l^1 set function a_j^1 has the form

$$a_j^1[x] = g\left(d_j^1(x)\right) = g\left(\sum_{i=1}^n \left|\frac{x_i - m_j^i}{\sigma_j^i}\right|\right). \tag{110}$$

Fig. 11 shows some of the l^1 metrical sets with $m = 0$ and $\sigma = [2\ 1]$ for the 2-D input case. The function g takes the form of a symmetrical triangle, parabola, Gaussian, Cauchy, Laplace, sinc, logistic, hyperbolic tangent, or differential logistic function.

We now consider the extreme case of the l^p metrical set functions when $p = \infty$. This gives the "max" metric. The l^∞ set function has the form

$$a_j^\infty[x] = g\left(d_j^\infty(x)\right) \tag{111}$$

$$= g\left(\lim_{p\to\infty}\left(\sum_{i=1}^n \left|\frac{x_i - m_j^i}{\sigma_j^i}\right|^p\right)^{\frac{1}{p}}\right) \tag{112}$$

$$= g\left(\max_{1\le i\le n}\left|\frac{x_i - m_j^i}{\sigma_j^i}\right|\right). \tag{113}$$

Note that $|(x_i - m_j^i)/\sigma_j^i|$ is never negative. So if $g(x)$ is monotone decreasing for $x \geq 0$ (such as for a triangle or Gaussian function or any unimodal function where a_j peaks at $x = m_j$) then

$$a_j^\infty[x] = g\left(\max_{1\le i\le n}\left|\frac{x_i - m_j^i}{\sigma_j^i}\right|\right) \tag{114}$$

$$= \min_{1\le i\le n} g\left(\left|\frac{x_i - m_j^i}{\sigma_j^i}\right|\right) \tag{115}$$

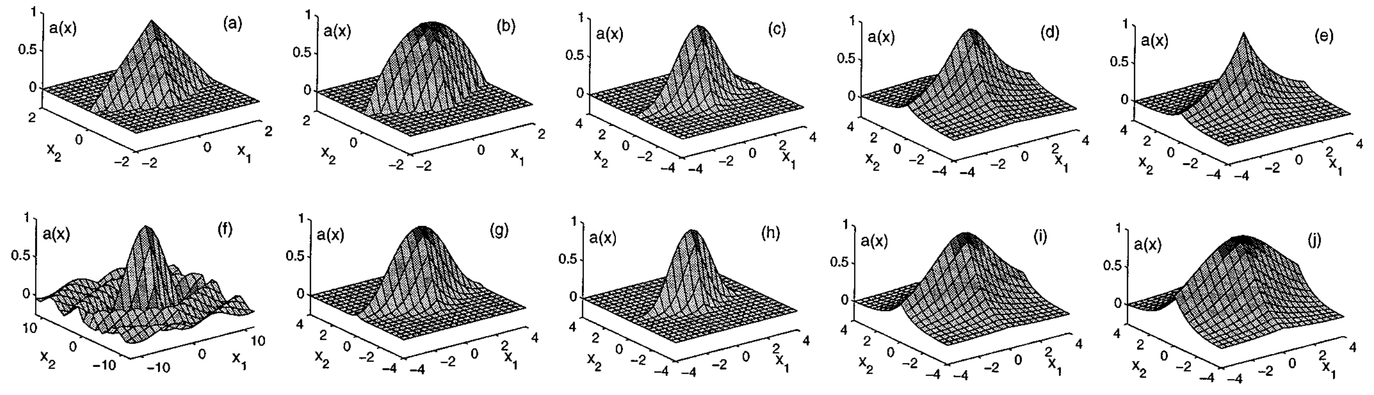


Fig. 11. Metrical joint set functions with $m = 0, \sigma = [2\ \ 1]$, and l^1 distance. (a) Symmetric metrical triangle set function. (b) Metrical parabola set function. (c) Metrical Gaussian set function. (d) Metrical Cauchy set function. (e) Metrical Laplace set function. (f) Metrical sinc set function. (g) Metrical logistic set function. (h) Metrical hyperbolic tangent set function. (i) Metrical hyperbolic secant set function. (j) Metrical differential logistic set function.

$$= \min_{1\le i\le n} a_{ji}\left(\left|\frac{x_i - m_j^i}{\sigma_j^i}\right|\right) \tag{116}$$

holds for a scalar set function $a_{ji}(x_i) = g(|(x_i - m_j^i)/\sigma_j^i|)$. So the l^∞ metrical joint sets a_j^∞ with g monotone decreasing are equivalent to the factorable sets with the min conjunctive operator. Fig. 12 shows the sets of points that give the same distance from the origin with l^p metric for $p = 1, 2,$ and ∞. So factorable set functions with min bound the metrical set functions in (108) through the l^p metric in (107).

The shape and orientation of the "hills" of if-part fuzzy sets may help fuzzy systems better approximate certain functions in that region. So we transform the translated input vector $(x - m_j) \in R^n$ to $A_j(x - m_j)$ where $A_j : R^n \to R^n$ is any linear or nonlinear operator [16]. We transform the translated vector $x - m_j$ instead of the input vector x because it is easier to keep track of the "center" vector m_j (if we use a unimodal set function such as the Gaussian and some mapping A_j such that $A_j(x) = 0$ if and only if $x = 0$).

Here we show the simple case of a linear transformation. Say A_j is an $n \times n$ matrix $A_j \in R^{n\times n}$. Then define the norm (or distance with the vector m_j) as

$$d_j^p(x) = \|A_j(x - m_j)\|_p \tag{117}$$

for the jth metrical set function $a_j[x] = g(d_j^p(x))$ as above. The p-norm $\|x\|_p$ of a vector $x = [x_1, \ldots, x_n]^T \in R^n$ has the form

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{\frac{1}{p}}. \tag{118}$$

So we can rewrite the quadratic distance $d_j^2(x) = \sqrt{(x - m_j)^T K_j (x - m_j)}$ in (94) in the form of (117). The matrix K_j is symmetrical nonnegative definite: $K_j = K_j^T \ge 0$. So $K_j = \sqrt{K_j}\sqrt{K_j}$ and $\sqrt{K_j} = (\sqrt{K_j})^T$ [29]. This implies that

$$d_j^2(x) = \sqrt{(x - m_j)^T K_j (x - m_j)} \tag{119}$$

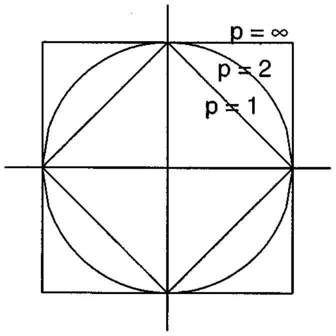


Fig. 12. Spheres in different metric spaces.

$$= \sqrt{(x - m_j)^T \sqrt{K_j^T}\sqrt{K_j}(x - m_j)} \tag{120}$$

$$= \|\sqrt{K_j}(x - m_j)\|_2. \tag{121}$$

This has the form $\|A_j(x - m_j)\|_p$ where $A_j = \sqrt{K_j}$ and $p = 2$.

Users may encode more useful information in the nonlinear operator A_j to reduce the number of fuzzy rules and perhaps lessen the rule explosion. Finding good combinations of nonlinear maps A_j and metrics d_j^p and functional form g remains an open research problem.

VIII. Supervised Learning in SAMs: Metrical Sets

The learning laws for the then-part set centroids c_j and volumes V_j remain the same for any if-part fuzzy sets. Only the learning laws for if-part set parameters have new forms. The joint metrical set functions depend on the metric d_j. So we tune the parameters that define the metric d_j. For the quadratic metric $d_j(x)^2 = (x - m_j)^T K_j (x - m_j)$ we tune the vector m_j and the matrix K_j

$$m_j(t+1) = m_j(t) - \mu_t \nabla_{m_j} E \tag{122}$$

$$K_j(t+1) = K_j(t) - \mu_t \nabla_{K_j} E. \tag{123}$$

The partial derivatives (or gradients in the vector-matrix cases) follow from (24) in like manner

$$\nabla_{m_j} E = \frac{\partial E}{\partial F}\frac{\partial F}{\partial a_j}\frac{\partial a_j}{\partial d_j}\nabla_{m_j} d_j \tag{124}$$

$$\nabla_{K_j} E = \frac{\partial E}{\partial F}\frac{\partial F}{\partial a_j}\frac{\partial a_j}{\partial d_j}\nabla_{K_j} d_j. \tag{125}$$

We have derived the first two partial derivatives in (26) and (27). The partial derivative $(\partial a_j)/(\partial d_j)$ depends on which scalar set function we use to create the joint set function.

1) Symmetric metrical triangle set function

$$\frac{\partial a_j}{\partial d_j} = \begin{cases} -1, & \text{if } d_j(x) < 1 \\ 0, & \text{else,} \end{cases} \tag{126}$$

2) Metrical parabola set function

$$\frac{\partial a_j}{\partial d_j} = \begin{cases} -2d_j(x), & \text{if } d_j(x) < 1 \\ 0, & \text{else.} \end{cases} \tag{127}$$

3) Joint Gaussian set function

$$\frac{\partial a_j}{\partial d_j} = -2d_j(x)a_j[x]. \tag{128}$$

4) Joint Cauchy set function

$$\frac{\partial a_j}{\partial d_j} = -(n+1)\frac{d_j(x)}{1+d_j(x)^2}a_j[x]. \tag{129}$$

5) Metrical Cauchy set function

$$\frac{\partial a_j}{\partial d_j} = -2d_j(x)a_j[x]^2. \tag{130}$$

6) Metrical Laplace set function

$$\frac{\partial a_j}{\partial d_j} = -a_j[x]. \tag{131}$$

7) Metrical sinc set function

$$\frac{\partial a_j}{\partial d_j} = \frac{1}{d_j(x)}(\cos(d_j(x)) - a_j[x]). \tag{132}$$

8) Metrical logistic set function

$$\frac{\partial a_j}{\partial d_j} = -d_j(x)(2 - a_j[x])a_j[x]. \tag{133}$$

9) Metrical hyperbolic tangent set function

$$\frac{\partial a_j}{\partial d_j} = -2d_j(x)(2 - a_j[x])a_j[x]. \tag{134}$$

10) Metrical hyperbolic secant set function

$$\frac{\partial a_j}{\partial d_j} = -\tanh(d_j(x))a_j[x]. \tag{135}$$

11) Metrical differential logistic set function

$$\frac{\partial a_j}{\partial d_j} = -S(d_j(x))a_j[x]. \tag{136}$$

These partial derivatives $(\partial a_j)/(\partial d_j)$ hold for any metric d_j that users might choose. They are independent of the function d_j that we use to transform the input vector x into the scalar $d_j(x)$.

We now derive the gradients of the metric d_j with respect to the vector m_j and matrix K_j for the quadratic case $d_j(x)^2 = (x-m_j)^T K_j (x-m_j)$. The gradients have the form

$$\nabla_{m_j} d_j = -\frac{1}{d_j(x)}K_j(x-m_j) \tag{137}$$

$$\nabla_{K_j} d_j = \frac{1}{2d_j(x)}(x-m_j)(x-m_j)^T \tag{138}$$

since $K_j = K_j^T$.

We might use diagonal matrices K_j to reduce the computation. This reduces the quadratic form of d_j to a weighted l^2 norm. We can also use any l^p norm to compute $d_j^p(x)$ as mentioned earlier. We also examine set functions from the l^1 norm as in (109). The partial derivatives have the form

$$\frac{\partial d_j^1}{\partial m_j^k} = -\text{sgn}\left(x_k - m_j^k\right)\frac{1}{\left|\sigma_j^k\right|} \tag{139}$$

$$\frac{\partial d_j^1}{\partial \sigma_j^k} = -\frac{\left|x_k - m_j^k\right|}{\left(\sigma_j^k\right)^2} \tag{140}$$

for $\sigma_j^k > 0$. The learning laws for the set functions that use the l^p metric in (106) follow in like manner. We now derive the learning laws for the metrical set function $a_j[x] = g(d_j^p(x))$ where d_j^p takes the form in (117) and A_j is a matrix $A_j \in R^{n\times n}$. Let $[A_j]_i$ denote the ith row of an $n\times n$ matrix A_j and put $m_j = [m_j^1,\ldots,m_j^n]^T$. We can rewrite the norm $d_j^p(x)$ as

$$d_j^p(x) = \|A_j(x-m_j)\|_p \tag{141}$$

$$= \left(\sum_{i=1}^{n} |[A_j]_i(x-m_j)|^p\right)^{\frac{1}{p}}. \tag{142}$$

So the gradient (in row vector notation) for the kth row of A_j is

$$\nabla_{[A_j]_k} d_j^p = \frac{1}{p}\left(\sum_{i=1}^{n} |[A_j]_i(x-m_j)|^p\right)^{\frac{1}{p}-1} \times \nabla_{[A_j]_k}\sum_{i=1}^{n} |[A_j]_i(x-m_j)|^p \tag{143}$$

$$= \frac{1}{p}\left(d_j^p(x)\right)^{-1}\nabla_{[A_j]_k}|[A_j]_k(x-m_j)|^p \tag{144}$$

$$= \frac{1}{d_j^p(x)}|[A_j]_k(x-m_j)|^{p-1}\operatorname{sgn}([A_j]_k(x-m_j)) \times \nabla_{[A_j]_k}([A_j]_k(x-m_j)) \tag{145}$$

$$= \frac{1}{d_j^p(x)}|[A_j]_k(x-m_j)|^{p-1} \times \operatorname{sgn}([A_j]_k(x-m_j))(x-m_j)^T. \tag{146}$$

The gradient of the metric d_j^p with respect to m_j (in column vector notation) follows in like manner

$$\nabla_{m_j} d_j^p = \frac{1}{p}\left(\sum_{i=1}^{n}|[A_j]_i(x-m_j)|^p\right)^{\frac{1}{p}-1} \times \nabla_{m_j}\sum_{i=1}^{n}|[A_j]_i(x-m_j)|^p \tag{147}$$

$$= \frac{1}{p}\left(d_j^p(x)\right)^{-1}\sum_{i=1}^{n}\nabla_{m_j}|[A_j]_i(x-m_j)|^p \tag{148}$$

$$= \frac{p}{d_j^p(x)}\sum_{i=1}^{n}p|[A_j]_i(x-m_j)|^{p-1} \times \nabla_{m_j}|[A_j]_i(x-m_j)| \tag{149}$$

$$= \frac{1}{d_j^p(x)}\sum_{i=1}^{n}|[A_j]_i(x-m_j)|^{p-1} \times \operatorname{sgn}([A_j]_i(x-m_j))\left(-[A_j]_i^T\right) \tag{150}$$

$$= -\frac{1}{d_j^p(x)}\sum_{i=1}^{n}|[A_j]_i(x-m_j)|^{p-1} \times \operatorname{sgn}([A_j]_i(x-m_j))[A_j]_i^T. \tag{151}$$

We can further tune the parameter p in the l^p metric in (106)

$$\frac{\partial d_j^p}{\partial p} = -\frac{1}{p}d_j^p(x)\ln d_j^p(x) + \frac{1}{p}\left(\sum_{i=1}^{n}\left|\kappa_j^i\left(x_i-m_j^i\right)\right|^p\right)^{\frac{1}{p}-1} \times \left(\sum_{i=1}^{n}\left|\kappa_j^i\left(x_i-m_j^i\right)\right|^p\ln\left|\kappa_j^i\left(x_i-m_j^i\right)\right|\right) \tag{152}$$

$$= -\frac{1}{p}d_j^p(x)\ln d_j^p(x) + \frac{1}{p}\frac{1}{\left(d_j^p(x)\right)^{p-1}} \times \left(\sum_{i=1}^{n}\left|\kappa_j^i\left(x_i-m_j^i\right)\right|^p\ln\left|\kappa_j^i\left(x_i-m_j^i\right)\right|\right). \tag{153}$$

The partial derivative when the metric d_j^p has the form (117) has the similar form

$$\frac{\partial d_j^p}{\partial p} = -\frac{1}{p}d_j^p(x)\ln d_j^p(x) + \frac{1}{p}\frac{1}{\left(d_j^p(x)\right)^{p-1}} \times \left(\sum_{i=1}^{n}|[A_j]_i(x-m_j)|^p\ln|[A_j]_i(x-m_j)|\right). \tag{154}$$

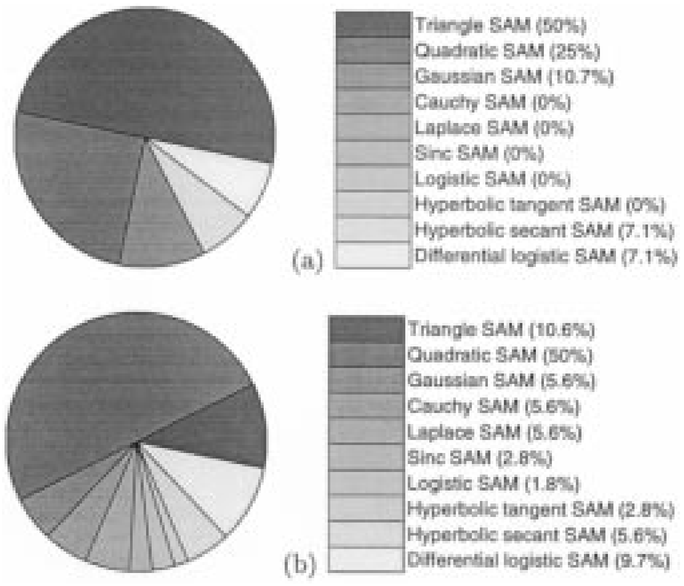


Fig. 13. l^2 metrical sets. Proportions of test cases where each metrical set function performed best. (a) 2-D test cases. (b) 3-D test cases. Note that the metrical triangle and the metrical quadratic switch from first and second place for the 2-D test cases to second and first place for the 3-D test cases.

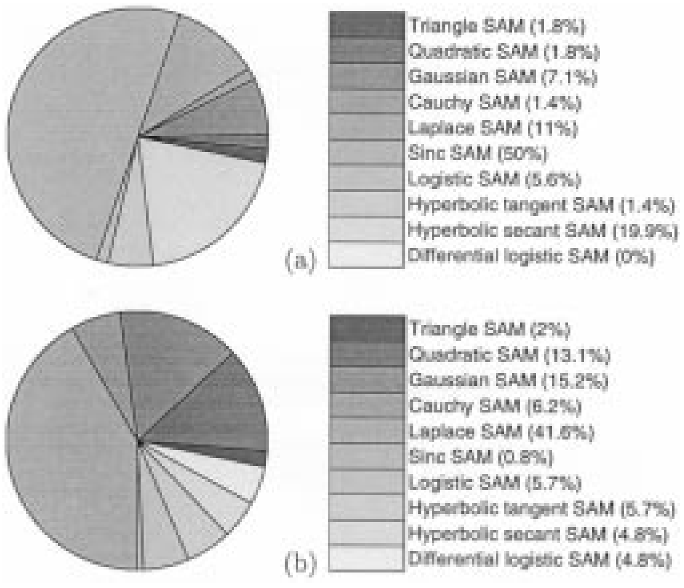


Fig. 14. l^1 metrical sets. Proportions of test cases where each metrical set function performed best. (a) 2-D test cases. (b) 3-D test cases. The l^1-metrical sinc goes from winner for the 2-D test cases to loser for the 3-D test cases. The l^1-metrical Laplacian emerges as the winner for the first time in the 3-D case.

IX. Simulation Results II: Joint Metrical Sets

Figs. 13 and 14 show the second results of quadratic l^2 and l^1 metrical sets in 2-D and 3-D test cases. Fig. 13 shows that the metrical triangle performs best in the 2-D experiments while the l^2-metrical quadratic performs second best. This outcome reverses in the 3-D experiments. There the l^2-metrical quadratic if-part set performs best while the metrical triangle performs second best. Fig. 14 shows that the l^1-metrical sinc wins for the 2-D test cases but loses for the 3-D test cases (when the l^1-metrical Laplace wins).

A. The Second Curse of Dimensionality and Unfactorable Metrical Sets

Our final result is negative: even unfactorable joint set functions can suffer the second curse of dimensionality of spikiness in high dimensions. The following theorem illustrates this claim for metrical set functions that depend on diagonal matrices. The result may also hold for many nondiagonal matrices.

Theorem: Suppose that a metrical set function a_j^p has the form

$$a_j^p(x) = g\left(d_j^p(x)\right) \tag{155}$$

for the l^p metric $d_j^p(x) = \|A_j(x - m_j)\|_p$. Here A_j is an $n \times n$ positive-definite diagonal matrix and $g : R_+ \to [0,1]$ is a monotone decreasing function such that $g(x) \to 0$ as $x \to \infty$. Then a_j^p suffers the second curse of dimensionality: it collapses to a spike in high dimension as n grows to ∞.

Proof: Recall that factorable set functions with min conjunction $a_j(x) = \min_i g(|(x_i - m_j^i)/d_j^i|)$ collapse to spikes in high dimensions for monotone decreasing g such that $g(x) \to 0$ as $x \to \infty$ (see Section II). So we need show only that for a given metrical set function a_j^p in (155) there exists a *factorable* set function $\tilde{a}_j$ (generated from the same function g) that bounds a_j: $a_j(x) \le \tilde{a}_j(x)$. Then the metrical set a_j^p collapses to a spiky surface in high dimensions.

For a matrix A_j it follows that

$$\begin{aligned}
a_j^p(x) &= g(\|A_j(x - m_j)\|_p) && (156)\\
&= g(\|A_j x - A_j m_j\|_p) && (157)\\
&= g(\|\tilde{x} - \tilde{m}_j\|_p) && (158)\\
&\le g(\|\tilde{x} - \tilde{m}_j\|_\infty) = a_j^\infty(x)\\
&\quad \text{since } \|x\|_p \ge \|x\|_\infty \text{ (see Lemma below)}\\
&\quad \text{and } g : R_+ \to [0,1] \text{ is monotone decreasing} && (159)\\
&= g\left(\max_i \left|\tilde{x}_i - \tilde{m}_j^i\right|\right) = \min_i g\left(\left|\tilde{x}_i - \tilde{m}_j^i\right|\right)\\
&\quad \text{since } g \text{ is monotone decreasing} && (160)\\
&= \min_i \tilde{a}_j^i(x) = \tilde{a}_j(x) && (161)
\end{aligned}$$

where $\tilde{a}_j^i(x) = g(|[A_j]_i x - [A_j]_i m_j|)$ and $[A_j]_i$ is the ith row of A_j. So $\min_i \tilde{a}_j^i(x)$ bounds $a_j^p(x)$. Q.E.D.

Lemma: $\|x\|_p \ge \|x\|_\infty$ if $x = [x_1, \ldots, x_n] \in R^n$.

Proof: Consider $x \in R^n$. Then

$$\sum_{i=1}^n |x_i|^p \ge |x_j|^p \quad \text{for all } j = 1, \ldots, n \tag{162}$$

$$\Leftrightarrow \left(\sum_{i=1}^n |x_i|^p\right)^{1/p} \ge |x_j| \quad \text{for all } j = 1, \ldots, n \tag{163}$$

$$\Leftrightarrow \left(\sum_{i=1}^n |x_i|^p\right)^{1/p} \ge \max_j |x_j| = \lim_{r\to\infty} \left(\sum_{i=1}^n |x_i|^r\right)^{1/r}. \tag{164}$$

So $\|x\|_p \ge \|x\|_\infty$. Q.E.D.

Note that the set function $\tilde{a}_j$ may not count as a factorable set function since each component $\tilde{a}_j^i$ takes as input the whole vector $x \in R^n$. Then the ith row of A_j transforms the input vector x into a scalar $\tilde{x}_i$. Therefore $\{\tilde{x}_i\}_{i=1}^n$ may not be independent and so the theorem (Borel–Cantelli lemma) [9] need not apply. The theorem does apply if A_j is diagonal.

X. Conclusion: The Search Goes On

At least three main conclusions follow from the above if-part fuzzy-set definitions, learning laws, and simulations of how these if-part sets affect adaptive fuzzy function approximation. The first conclusion is that curses of dimensionality alone will impose tight limits on empirical searches for the best shape of parametrized if-part fuzzy sets. The complexity of the learning laws further compounds this computational burden. It limited our simulation experiments to no more than three dimensions. The sets that performed well in these smaller dimensions may not do so in higher dimensions. The winner histograms even changed dramatically when going from one to two to three dimensions. The second dimensionality curse of set spikiness will also have greater force for searches through the spaces of four- and higher dimensional set functions.

The second conclusion is that common sense or even expert intuition may offer little guidance for picking good if-part sets in higher dimensions. Indeed, they may mislead even in the scalar case. The frequent winning status of the sinc set in the simulations shows that. This seems to be the first time anyone has used the sinc function as a fuzzy set and yet such sets may well have improved the performance of many real fuzzy systems. Surely there are many more scalar if-part sets that would perform even better for these and other test functions and that would appear even less intuitive or have less linguistic meaning than does the sinc function. Again, the engineering goal of accurate function approximation will tend to lead the search for the best if-part set far beyond where the earlier goal of accurate linguistic modeling would take it. And the success of the sinc set and the hyperbolic-tangent bell curve further suggest that the familiar Gaussian or Cauchy or other familiar unimodal curves will not emerge as optimal set functions in other searches.

The third conclusion follows from the other two: The search for the best shape of if-part (and then-part) sets will continue. There are as many continuous if-part fuzzy subsets of the real line as there are real numbers. The set of all if-part fuzzy subsets of the real line has the higher cardinality of the set of all subsets of the real line. Fuzzy theorists will never exhaust this search space. Each theorist can draw different lines through the space to form set taxonomies or to focus the search or to pose narrow or broad optimality problems. We suspect that many such searches will take care to distinguish factorable from unfactorable sets though they may well ignore our distinction of parametrized versus nonparametrized sets. The unfactorable sets hold the promise that they may lessen if not defeat exponential rule explosion even if they may still suffer from set spikiness. These searches may be endless in principle but that itself does not mean that they are not worthwhile. They can on occasion produce new tools.

Appendix
The Standard Additive Model (SAM) Theorem

This Appendix derives the basic ratio structure (6) of a standard additive model fuzzy system and review the local structure of optimal fuzzy rules.

SAM Theorem: Suppose the fuzzy system $F : R^n \rightarrow R^p$ is a standard additive model: $F(x) = \text{Centroid}(B(x)) = \text{Centroid}(\sum_{j=1}^m w_j a_j(x) B_j)$ for if-part joint set function $a_j : R^n \rightarrow [0, 1]$, rule weights $w_j \geq 0$, and then-part fuzzy set $B_j \subset R^p$. Then $F(x)$ is a convex sum of the m then-part set centroids

$$F(x) = \frac{\sum_{j=1}^m w_j a_j(x) V_j c_j}{\sum_{j=1}^m w_j a_j(x) V_j} = \sum_{j=1}^m p_j(x) c_j. \tag{165}$$

The convex coefficients or discrete probability weights $p_1(x), \ldots, p_m(x)$ depend on the input x through

$$p_j(x) = \frac{w_j a_j(x) V_j}{\sum_{i=1}^m w_i a_i(x) V_i}. \tag{166}$$

V_j is the finite positive volume (or area if $p = 1$) and c_j is the centroid of then-part set B_j

$$V_j = \int_{R^p} b_j(y_1, \ldots, y_p)\, dy_1 \ldots dy_p > 0 \tag{167}$$

$$c_j = \frac{\int_{R^p} y b_j(y_1, \ldots, y_p)\, dy_1 \ldots dy_p}{\int_{R^p} b_j(y_1, \ldots, y_p)\, dy_1 \ldots dy_p}. \tag{168}$$

Proof: There is no loss of generality to prove the theorem for the scalar-output case $p = 1$ when $F : R^n \rightarrow R^p$. This simplifies the notation. We need but replace the scalar integrals over R with the p-multiple or volume integrals over R^p in the proof to prove the general case. The scalar case $p = 1$ gives (167) and (168) as

$$V_j = \int_{-\infty}^{\infty} b_j(y)\, dy \tag{169}$$

$$c_j = \frac{\int_{-\infty}^{\infty} y b_j(y)\, dy}{\int_{-\infty}^{\infty} b_j(y)\, dy}. \tag{170}$$

Then the theorem follows if we expand the centroid of B and invoke the SAM assumption $F(x) = \text{Centroid}(B(x)) = \text{Centroid}(\sum_{j=1}^m w_j a_j(x) B_j)$ to rearrange terms

$$F(x) = \text{Centroid}(B(x)) = \frac{\int_{-\infty}^{\infty} y b(y)\, dy}{\int_{-\infty}^{\infty} b(y)\, dy} \tag{171}$$

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^m w_j b_j'(y)\, dy}{\int_{-\infty}^{\infty} \sum_{j=1}^m w_j b_j'(y)\, dy} \tag{172}$$

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^m w_j a_j(x) b_j(y)\, dy}{\int_{-\infty}^{\infty} \sum_{j=1}^m w_j a_j(x) b_j(y)\, dy} \tag{173}$$

$$= \frac{\sum_{j=1}^m w_j a_j(x) \int_{-\infty}^{\infty} y b_j(y)\, dy}{\sum_{j=1}^m w_j a_j(x) \int_{-\infty}^{\infty} b_j(y)\, dy} \tag{174}$$

$$= \frac{\sum_{j=1}^m w_j a_j(x) V_j \frac{\int_{-\infty}^{\infty} y b_j(y)\, dy}{V_j}}{\sum_{j=1}^m w_j a_j(x) V_j} \tag{175}$$

$$= \frac{\sum_{j=1}^m w_j a_j(x) V_j c_j}{\sum_{j=1}^m w_j a_j(x) V_j}. \tag{176}$$

Now we give a simple *local* description of optimal lone fuzzy rules [13], [14]. We move a fuzzy rule patch so that it most reduces an error. We look (locally) at a minimal fuzzy system $F : R \rightarrow R$ of just one rule. So the fuzzy system is constant in that region: $F = c$. Suppose that $f(x) \neq c$ for $x \in [a, b]$ and define the error

$$e(x) = (f(x) - F(x))^2 = (f(x) - c)^2. \tag{177}$$

We want to find the best place $\hat{x}$. So the first-order condition gives $\nabla e = \mathbf{0}$ or

$$0 = \frac{\partial e(x)}{\partial x} = 2(f(x) - c)\frac{\partial f(x)}{\partial x}. \tag{178}$$

Then $f(x) \neq c$ implies that

$$\frac{\partial e(x)}{\partial x} = 0 \Leftrightarrow \frac{\partial f(x)}{\partial x} = 0 \tag{179}$$

at $x = \hat{x}$. So the extrema of e and f coincide in this case. Fig. 4 shows how fuzzy rule patches can "patch the bumps" and so help minimize the error of approximation. □

References

[1] J. A. Dickerson and B. Kosko, "Fuzzy function approximation with supervised ellipsoidal learning," in *Proc. World Congr. Neural Networks (WCNN-93)*, vol. 2, July 1993, pp. 9–17.

[2] ——, "Fuzzy function approximation with ellipsoidal rules," *IEEE Trans. Syst., Man, Cybern.*, pt. B, vol. 26, pp. 542–560, Aug. 1996.

[3] J. J.-S. Jang and C.-T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. Neural Networks*, vol. 4, no. 1, pp. 156–159, Jan. 1993.

[4] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neurofuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Englewood Cliffs: Prentice-Hall, 1996.

[5] H. M. Kim and B. Kosko, "Fuzzy prediction and filtering in impulsive noise," *Fuzzy Sets Syst.*, vol. 77, no. 1, pp. 15–33, Jan. 15, 1996.

[6] ——, "Neural fuzzy motion estimation and compensation," *IEEE Trans. Signal Processing*, vol. 45, pp. 2515–2532, Oct. 1997.

[7] H. M. Kim and J. M. Mendel, "Fuzzy basis functions: Comparison with other basis functions," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 158–168, May 1995.

[8] G. J. Klir, U. H. St. Clair, and B. Yuan, *Fuzzy Set Theory: Foundations and Applications*. Englewood Cliffs: Prentice-Hall, 1997.

[9] B. Kosko, "Fuzzy knowledge combination," *Int. J. Intell. Syst.*, vol. I, pp. 293–320, 1986.

[10] ——, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Englewood Cliffs: Prentice-Hall, 1991.

[11] ——, "Fuzzy systems as universal approximators," *IEEE Trans. Computers*, vol. 43, no. 11, pp. 1329–1333, Nov. 1994.

[12] ——, "Combining fuzzy systems," *Proc. IEEE Int. Conf. Fuzzy Systems (IEEE FUZZ-95)*, pp. 1855–1863, Mar. 1995.

[13] ——, "Optimal fuzzy rules cover extrema," *Int. J. Intell. Syst.*, vol. 10, no. 2, pp. 249–255, Feb. 1995.

[14] ——, *Fuzzy Engineering*: Prentice Hall, 1996.

[15] ——, "Global stability of generalized additive fuzzy systems," *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, pp. 441–452, Aug. 1998.

[16] E. Kreyszig, *Introductory Functional Analysis with Applications*. New York: Wiley, 1978.

[17] S. Lee and R. M. Kil, "Multilayer feedforward potential function network," *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, pp. 141–152, 1988.

[18] S. Mitaim and B. Kosko, "What is the best shape for a fuzzy set in function approximation?," *Proc. 5th IEEE Int. Conf. Fuzzy Systems (FUZZ-96)*, vol. 2, pp. 1237–1243, Sept. 1996.

[19] ——, “Adaptive joint fuzzy sets for function approximation,” *Proc. 1997 IEEE Int. Conf. Neural Networks (ICNN-97)*, vol. 1, pp. 537–542, June 1997.

[20] J. Moody and C. Darken, “Fast learning in networks of locally-tuned processing unit,” *Neural Computat.*, vol. 1, no. 2, pp. 281–294, 1989.

[21] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[22] P. J. Pacini and B. Kosko, “Adaptive fuzzy frequency hopper,” *IEEE Trans. Commun.*, vol. 43, pp. 2111–2117, June 1995.

[23] A. Papoulis, *Probability and Statistics*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[24] W. Pedrycz, “Why triangular membership functions?,” *Fuzzy Sets Syst.*, vol. 64, pp. 21–30, 1994.

[25] S. J. Press, “Multivariate stable distributions,” *J. Multivariate Anal.*, vol. 2, pp. 444–462, 1972.

[26] T. A. Runkler and J. C. Bezdek, “Function approximation with polynomial membership functions and alternating cluster estimation,” *Fuzzy Sets Syst.*, vol. 101, pp. 207–218, 1999.

[27] H. W. Sorenson and D. L. Alspach, “Recursive Bayesian estimation using Gaussian sums,” *Automatica*, vol. 7, no. 4, pp. 465–479, July 1971.

[28] D. F. Specht, “A general regression neural network,” *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 549–557, 1991.

[29] G. Strang, *Linear Algebra and Its Applications*, 3rd ed. New York: Harcourt Brace Jovanovich, 1988.

[30] L. X. Wang and J. M. Mendel, “Fuzzy basis functions, universal approximation, and orthogonal least-squares learning,” *IEEE Trans. Neural Networks*, vol. 3, pp. 807–814, Sept. 1992.

[31] F. A. Watkins, “The representation problem for additive fuzzy systems,” *Proc. IEEE Int. Conf. Fuzzy Systems (IEEE FUZZ-95)*, Mar. 1995.

[32] ——, “The trouble with triangles,” in *Proc. INNS World Congr. Neural Networks (WCNN-96)*, San Diego, CA, Sept. 1996, pp. 1123–1126.

[33] J. Zhang and A. Knoll, “Designing fuzzy controllers by rapid learning,” *Fuzzy Sets Syst.*, vol. 101, no. 2, pp. 287–301, Jan. 1999.