Global Stability of Generalized Additive Fuzzy Systems

Bart Kosko, *Member, IEEE*

***Abstract*—This paper explores the stability of a class of feedback fuzzy systems. The class consists of generalized additive fuzzy systems that compute a system output as a convex sum of linear operators. Continuous versions of these systems are globally asymptotically stable if all rule matrices are stable (negative definite). So local rule stability leads to global system stability. This relationship between local and global system stability does not hold for the better known discrete versions of feedback fuzzy systems. A corollary shows that it does hold for the discrete versions in the special but practical case of diagonal rule matrices. The paper first reviews additive fuzzy systems and then extends them to the class of generalized additive fuzzy systems. The Appendix derives the basic ratio structure of additive fuzzy systems and shows how supervised learning can tune their parameters.**

***Index Terms*— Feedback stability, function approximation, fuzzy systems, learning, neural networks, radial basis functions, rule explosion.**

I. Feedback Fuzzy Systems and Rule Explosion

FEEDBACK fuzzy systems take their own output as input. They use a set of if–then rules to define continuous or discrete autonomous dynamical systems on a real vector space

$$\dot{x} = F(x) \tag{1}$$

$$x(k+1) = F(x(k)). \tag{2}$$

The feedback fuzzy system F is a vector field $F\colon R^n \to R^n$ that has the origin as a fixed point: $F(0) = 0$. Feedback fuzzy systems can arise in control [3], [11], [38]–[41], [44], [45], in signal processing [4], [40], or in models of complex social or medical processes [2], [5], [14]–[16], [29]–[31], [35], [36], [53], where subsystems affect one another in closed causal loops.

The feedback structure often arises because a feedforward fuzzy system $F\colon R^n \to R^n$ suffers from rule explosion in high dimensions [18]–[19]. Fuzzy systems are universal function approximators [17] as are feedforward neural networks [7]–[8]. But fuzzy systems need on the order of k^{n+p-1} rules to uniformly (and "blindly") approximate a continuous or bounded measurable function $f\colon R^n \to R^p$ on a compact domain. Fig. 1 shows how a few fuzzy rule patches can cover the graph of a simple scalar function.

Most fuzzy systems in practice have been simple feedforward fuzzy systems of low dimension. These scalar maps $F\colon R^n \to R$ convert a vector input x of sensor measurements into a scalar control output $F(x)$. The Appendix derives the formal structure of this mapping. This shows how to convert a set of linguistic if–then rules into a closed-form equation. Few electronics applications have required dedicated hardware. The math of such fuzzy systems is simple enough that engineers need only reprogram the microprocessor chip that already controls a microwave oven, washing machine, subway braking system, camcorder lens, or a car transmission [18].

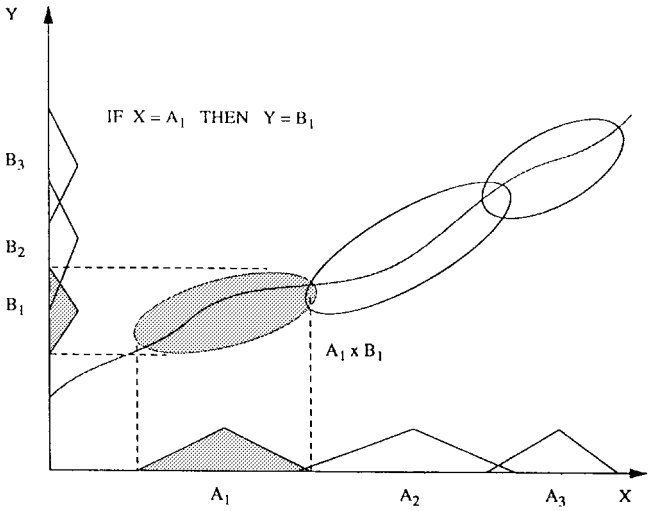

Fig. 1. Fuzzy system F approximates a function $f\colon R^n \to R^p$ by covering its graph with rule patches. Each rule patch is a fuzzy Cartesian product $A_j \times B_j \subset R^n \times R^p$ of an if-part fuzzy set $A_j \subset R^n$ and a then-part fuzzy set $B_j \subset R^p$. The number of rule patches in the cover grows exponentially on the order of k^{n+p-1} as the dimensions n and p grow linearly. Learning schemes move and shape the patches. Lone optimal rule patches cover the extrema of the function.

Engineers design these rule-based systems in four steps. They first pick the system's input and output variables. They define fuzzy subsets of these variables. They relate these fuzzy sets into I/O rules. Then they tune the fuzzy system with test data. Engineers tuned the first fuzzy systems by trial and error. Modern systems use neural networks, genetic algorithms, or other statistical learning schemes to tune the fuzzy sets and fuzzy rules. Such automated techniques become more important for fuzzy systems that use more than three input variables both because of the exponential growth in the number of rules and because humans do not guess well either at rules or equations that describe multivariable systems.

Consider a feedforward fuzzy system that controls a car's air conditioner (AC). Suppose the fuzzy AC system measures

Manuscript received October 29, 1997; revised January 30, 1998.

The author is with the Electrical Engineering Department, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-2564 USA (e-mail: kosko@sipi.usc.edu).

Publisher Item Identifier S 1094-6977(98)03890-5.

the air temperature and humidity and then adjusts the blower or fan. A simple fuzzy system might use these three rules.

IF the air is *cool* and *dry* THEN set the blower to *slow*.

IF the air is *warm* and *medium dry* THEN set the blower to *fast*.

IF the air is *hot* and *low dry* THEN set the blower to *blast*.

The input variable air temperature might have the fuzzy subsets *cold, cool, medium, warm*, and *hot*. The input variable air humidity might have the fuzzy subsets *low dry, medium dry*, and *dry*. The output variable blower speed might have the fuzzy subsets *stop, slow, medium, fast*, and *blast*. Then the fuzzy system F converts paired numerical measurements (x_1, x_2) of air temperature (x_1) and air humidity (x_2) into an output numerical blast speed $F(x_1, x_2)$.

A curve or triangle-like fuzzy set A_i might define what the user or engineer means by cool air. Each measured air temperature x_1 is both *cool* to degree $a_1(x_1) \in [0, 1]$ and *not cool* to degree $1 - a_1(x_1)$. This is the sense in which fuzzy systems are "fuzzy" or vague—concepts are deterministic but multivalued. Human judgment or statistical learning must pick the shape of these sets. The set shapes may change or the sets may shift along their variable's axis when new users ride in the car or when the old user's tastes change. Engineers seek efficient learning schemes to tune these set shapes and thus to approximate optimal rules.

Optimal rules offer the best way to deal with a fixed-rule budget. Optimal rules define fuzzy rule patches in the I/O state space that cover the extrema of f and the extrema of the approximation error function $e = (f - F)^2$ [19]. This still holds after the m rule patches of F have covered all of the extrema of f. Then adding one more rule to F gives the new fuzzy system F' of $m+1$ rules or rule patches. The best place to put this new rule patch is where it covers the largest bump of the new residual error curve $e' = (f - F')^2$. The supervised learning schemes in the Appendix tend to move rule patches toward these optimal positions in the I/O state space.

Optimal rules do not prevent rule explosion. They ease its computational burden as well as any rules can for a given shape of if-part fuzzy sets A_j. The popular symmetric triangular if-part sets give only a piecewise-linear fuzzy system F. Gaussian and Cauchy and other bell curve if-part sets give richer and smoother fuzzy systems F. The Appendix shows how to derive learning laws that tune fuzzy sets with these and other shapes. We assume throughout that the user picks the shape of the if-part sets A_j with no knowledge or use of the functional form of the approximand f.

More complex if-part sets can prevent rule explosion if they depend on the approximand f. The exponential rule complexity of blind approximation reduces to linear complexity in the rare case where we know the exact form of the approximand f and where the rules reflect the known f in the structure of their if-part fuzzy sets [49]. But we do not know f in practice. If we did there would be no need to approximate it. The search for optimal rules allows us in practice to guess only at the turning points of f and thus not have to guess at all of f itself. This converts the search for optimal rule patches to a search for the zeroes of the derivative map f': $f'(\hat{x}) = 0$.

Feedback fuzzy systems offer a way to approximate dynamical systems with a fixed set of rules. But feedback leads to far more complex dynamics and can end in chaos or instability. Most feedback systems are unstable. System trajectories need not converge to fixed points or to any region of the state space near them. Gradient systems are an exception. Gradient systems have the form $\dot{x} = -\nabla P(x)$ and are stable because the scalar potential P acts as a Lyapunov function for the system. Most stable neural systems are gradient systems with quadratic potentials [16], [25] but such systems are rare in the space of nonlinear dynamical systems. Feedback fuzzy systems of the form $\dot{x} = F(x)$ are not gradient systems.

Most research on stable fuzzy systems has used Lyapunov functions to give a sufficient condition for system stability. DeGlas [3] fuzzified the LaSalle invariance sets of stable equilibrium points to extend the usual results on Lyapunov stability [9]. Kiska and Gupta [11] put forth a nonquadratic energy function for feedback fuzzy systems based on min–max operators. They did not show if it acts as a Lyapunov function for a fuzzy system. Tong [45] observed that no scheme to control or stabilize a fuzzy system has yet shown how to measure fuzzy degrees of stability. That remains true today.

Recent results still apply the standard bivalent definitions of stability to well-defined feedback fuzzy systems F. Tanaka [38]–[41] simplified an additive fuzzy system to give the discrete feedback system in (2) as a convex sum of standard linear control systems. He then found a quadratic function that acts as a Lyapunov function for the system. But the odds of finding such a Lyapunov function fall as the number of rules grows. We review this result below in Theorem 2. Wang [48] used simple Gaussian additive systems (which have the same form as neural radial-basis function networks [24], [32]) to approximate stable control systems along the lines that Polycarpou and Ioannou [28] followed when they used neural approximators for the same task.

This paper gives sufficient conditions for the stability of (1) and (2) when the feedback fuzzy system F is a simple type of additive fuzzy system [16]. These nonlinear systems compute the global output $F(x)$ as a convex sum of the rule outputs: $F(x) = \sum_{j=1}^{m} p_j(x) B_j x$. The square matrix or linear operator B_j defines the then-part of the jth if-part rule "If $X = A_j$ then $Y = B_j$." Each rule acts as a linear subsystem as in the feedback scheme of Tanaka and Sugeno [38]. The system itself is nonlinear because the convex coefficents $p_1(x), \cdots, p_m(x)$ change with each input or state vector x. The system becomes a standard linear system [46] in the special case when the system has just one rule (when $m = 1$). The stability results below hold for all if-part fuzzy sets $A_j \subset R^n$ of all shapes.

Theorem 1 below proves that the continuous system (1) is globally asymptotically stable if each rule matrix is negative definite and thus if each local subsystem is stable. Tanaka [38]–[41] has shown that this result does not hold in the discrete case (2). Stable subsystems can still lead to global instability. Tanaka has applied his feedback model to the neural-fuzzy test problem of backing up a truck and one or more trailers [6], [13], [26], [40], [43]. We present Tanaka's result in the framework of a generalized additive system and

prove a corollary that gives a practical criterion for global stability of the discrete feedback fuzzy system (2).

The next section reviews standard additive fuzzy models and extends them to generalized additive models. The third section proves sufficient conditions for global asymptotic stability of these generalized additive models. The last section looks at the limits of these results and suggests other areas and other feedback fuzzy systems that future research might explore. The Appendix shows how supervised learning can tune the parameters of a standard additive model and tune the if-part and rule-weight parameters of a generalized additive fuzzy system. It derives the general update laws for if-part set parameters and exact learning laws to tune exponential and other bell-curve if-part sets.

II. Generalized Additive Fuzzy Systems

An *additive* fuzzy system [16], [21] F: $R^n \to R^p$ stores m rules of the form "If $X = A_j$ then $Y = B_j$" and adds the "fired" then-parts $B'_j(x)$ to give the output set $B(x)$

$$B(x) = \sum_{j=1}^{m} w_j B'_j(x) = \sum_{j=1}^{m} w_j a_j(x) B_j(x) \tag{3}$$

for scalar rule weights $w_j > 0$. The system is "fuzzy" [51], [52] or "vague" [1] because the m rules relate multivalued subsets of the input and output spaces.

A fuzzy or multivalued set $A \subset R^n$ has a multivalued indicator function a: $R^n \to [0,1]$. We call this map a set function or a "membership" [51] function because $a(x)$ measures the degree to which the object $x \in R^n$ belongs to the set A: $a(x) = Degree(x \in A)$. The then-part set $B_j \subset R^p$ has a like set function b_j: $R^p \to [0,1]$. A finite discrete space $X = \{x_1, \cdots, x_n\}$ has continuum many fuzzy subsets $A \in X$ in the form of fuzzy unit or *fit* [16] vectors $A = (a_1, \cdots, a_n)$. Then we can write $a_i = a(x_i)$ for the discrete set function a: $X \to [0,1]$. So each fuzzy set or fit vector defines a unique point in the unit hypercube $[0,1]^n$. This cube or sets-as-points framework [16] extends to countable spaces X. A rule patch $A_j \times B_j \subset R^n \times R^p$ has set function $R_{A_j \times B_j}$: $R^n \times R^p \to [0,1]$. This defines a fuzzy matrix in the finite case. The value of a rule patch most often equals a simple product of fit values at each point in the I/O state space: $R_{A_j \times B_j}(x,y) = a_j(x) b_j(y)$.

The second sum in (3) states that the additive model is *standard* because it expands the fired then-part set $B'_j(x)$ with product scaling or correlation-product encoding: $B'_j(x) = a_j(x) B_j(x)$ for then-part set $B'_j(x)$. This implies the product patch structure $R_{A_j \times B_j}(x,y) = a_j(x) B_j(y)$.

Simple fuzzy systems use a fixed then-part fuzzy set B_j. A *generalized* fuzzy system views B_j as an arbitrary linear or nonlinear mapping B_j: $R^n \to R^p$. We will restrict B_j to the linear or matrix map on the input space B_j: $R^n \to R^n$. Then $x' = B_j(x) = B_j x$ is a state vector in the input state space.

The jth if-part fuzzy set $A_j \subset R^n$ has joint set function a_j: $R^n \to [0,1]$. The joint set function may factor into n scalar or marginal set functions a_j^1: $R \to [0,1], \cdots, a_j^n$: $R \to [0,1]$ to give the factored set function $a_j(x) = \prod_{i=1}^{n} a_j^i(x_i)$ or $a_j(x) = \min(a_j^1(x_1), \cdots, a_j^n(x_n))$ or any other combination of the scalar set functions. Most if-part combiners act as *and* or conjunctive combiners [12]. The product combiner remains sensitive to changes in the scalar values that the min combiner tends to ignore.

Joint set functions preserve correlations among input components. This reflects the topological fact that we cannot factor most fuzzy subsets $A_j \subset R^n$ into the Cartesian product $A_j^1 \times \cdots \times A_j^n$ for scalar fuzzy sets $A_j^i \subset R$ with set functions a_j^i: $R \to [0,1]$. Rectangular and joint Gaussian set functions are rare exceptions that not only factor but factor into scalar rectangles and Gaussian bell curves. Normalized distance measures can define joint set functions that preserve input correlations and that have a simple closed form [10]. Neural "competitive" learning schemes can form these joint set functions as ellipsoidal rule patches in the I/O space [4].

The stability theorems below hold for all joint set functions a_j: $R^n \to [0,1]$. We assume only that each vector input $x \in R^n$ fires at least one rule and thus belongs to at least one if-part set A_k to nonzero degree: $a_k(x) > 0$. This just means that the fuzzy system F is a well-defined function over some domain space.

The simplest then-part sets or operators B_j are fixed fuzzy subsets of a then-part vector space. Then the jth then-part fuzzy set $B_j \subset R^p$ has integrable set function b_j: $R^p \to [0,1]$ with finite positive volume or area V_j and centroid c_j

$$V_j = \int_{R^p} b_j(y)\, dy > 0 \tag{4}$$

$$c_j = \frac{\displaystyle\int_{R^p} y b_j(y)\, dy}{\displaystyle\int_{R^p} b_j(y)\, dy}. \tag{5}$$

We can extend the fixed then-part sets to point-to-set maps $B_j(x) \subset R^p$ or fuzzy sets that may change with each vector input x. Then the then-part set function b_j defines a map from a product space to real numbers or b_j: $R^n \times R^p \to [0,1]$. Each input x picks out a new then-part fuzzy set $B_j(x)$ through the restricted set function $b_j(x, \cdot)$: $R^p \to [0,1]$. These variable then-part set functions must replace the fixed then-part functions in (4) and (5) to give the variable then-part volumes $V_j(x)$ and centroids $c_j(x)$.

The Appendix shows that the sum (3) leads to a SAM or *standard additive model* if the system output $F(x)$ computes the centroid of the output set $B(x)$ when it "defuzzifies" $B(x)$ or maps the fuzzy set to a scalar or vector [50]

$$F(x) = \text{Centroid}\left(\sum_{j=1}^{m} w_j a_j(x) B_j(x)\right) \tag{6}$$

$$= \frac{\displaystyle\sum_{j=1}^{m} w_j a_j(x) V_j(x) c_j(x)}{\displaystyle\sum_{j=1}^{m} w_j a_j(x) V_j(x)} \tag{7}$$

$$= \sum_{j=1}^{m} p_j(x) c_j(x) \tag{8}$$

for the m convex coefficients

$$p_j(x) = \frac{w_j a_j(x) V_j(x)}{\sum_{i=1}^{m} w_i a_i(x) V_i(x)}. \tag{9}$$

Note that for each input x there must be at least one positive convex coefficient: $p_k(x) > 0$. This holds because x belongs to at least one of the m if-part sets to nonzero degree: $a_k(x) > 0$. The proof of Theorem 1 below depends on this fact.

The SAM system gives the global output $F(x)$ as the convex sum of local outputs or centroids. This convex structure holds for all additive fuzzy systems [16]–[17]. The Appendix also shows how supervised learning or gradient descent can tune the four SAM parameters in (7).

The SAM system can also take a fuzzy set $A \subset R^n$ as input and still compute a vector or point output $F(A) \in R^p$. A correlation achieves this and extends the if-part set functions in a natural way [21]: $a_j(A) = \int_{R^n} a(x) a_j(x)\, dx$. This replaces the convolved delta pulse $\delta(x - x_0)$ or binary "singleton" set with the set function a: $R^n \to [0,1]$ in the fit-valued integral $a_j(x_0) = \int_{R^n} \delta(x - x_0) a_j(x)\, dx$. Then the proof of the SAM theorem still goes through and gives the set-SAM output $F(A) = \sum_{j=1}^{m} p_j(A) c_j$.

A problem occurs if we try to extend the SAM fuzzy dynamical system $\dot{x} = F(x)$ to a set-SAM dynamical system of the form $\dot{A} = F(A)$. The set SAM still gives a convex sum of then-part centroids and thus a vector. Its centroidal structure does not give a set as output. We could view the vector output as the center of a set or use it in some other *ad hoc* scheme to produce a final output set $F(A)$. But there is no direct way to convert a set SAM into a dynamical system. We will work instead with dynamical systems on point spaces. Yet even here we have to extend the SAM framework to a generalized SAM to ensure practical conditions for global stability.

We now derive a like SAM theorem for a fully generalized SAM. We will use a special form of this model to define the fuzzy dynamical systems (1) and (2). The key idea is to replace the generalized "set function" $b_j(x, \cdot)$ with a Kroeneker delta pulse in the discrete case and with a Dirac delta function in the continuous case.

Suppose B_j is an arbitrary map from the input vector space R^n to the output vector space R^p. The map B_j: $R^n \to R^p$ is again just an n-by-p matrix B_j in the linear case we will arrive at. The map B_j is a nonlinear operator in general. It maps each x to a new output vector $y_j = B_j(x)$. The output vector y_j depends on x as $y_j(x)$. We omit this notation for simplicity. We can view this nonlinear-operator case as a special case of the above point-to-set case $b_j(x, \cdot)$: $R^p \to [0,1]$ if we view the discrete then-part fuzzy set $B_j(x)$ as the singleton set $\{y_j(x)\}$. This gives a unit pulse or binary set function

$$b_j(x, \cdot) = \begin{cases} 1, & \text{if } B_j(x) = y_j \\ 0, & \text{if } B_j(x) \neq y_j. \end{cases} \tag{10}$$

An exercise shows that (10) gives a discrete version of the SAM Theorem. Here the summable count $c(B_j) = b_j(y_1) + b_j(y_2) + \cdot$ replaces the volume V_j in (4).

The continuous case $B_j \subset R^p$ requires that we replace the singleton set $\{y_j\}$ with a delta pulse

$$b_j(x, \cdot) = \delta(y - y_j) = \delta(y - y_j(x)). \tag{11}$$

Then each x gives a generalized "set" $B_j(x)$ with unit volume and with a new spike or range point y_j for its centroid $c_j(x)$

$$V_j(x) = \int_{R^p} b_j(x,y)\, dy = \int_{R^p} \delta(y - y_j)\, dy = 1 \tag{12}$$

$$c_j(x) = \frac{\int_{R^p} y b_j(x,y)\, dy}{\int_{R^p} b_j(x,y)\, dy} = \int_{R^p} y \delta(y - y_j)\, dy = y_j(x). \tag{13}$$

Then the additive combiner in (3) further reduces to

$$B(x) = \sum_{j=1}^{m} w_j B_j'(x) = \sum_{j=1}^{m} w_j a_j(x) B_j(x) = \sum_{j=1}^{m} w_j a_j(x) \delta(y - y_j) \tag{14}$$

and leads to the generalized SAM Theorem in (19)

$$F(x) = \text{Centroid}(B(x)) \tag{15}$$

$$= \frac{\int_{R^p} y b(x,y)\, dy}{\int_{R^p} b(x,y)\, dy} \tag{16}$$

$$= \frac{\int_{R^p} y \sum_{j=1}^{m} w_j a_j(x) \delta(y - y_j)\, dy}{\int_{R^p} \sum_{j=1}^{m} w_j a_j(x) \delta(y - y_j)\, dy} \tag{17}$$

$$= \frac{\sum_{j=1}^{m} w_j a_j(x) \int_{R^p} y \delta(y - y_j)\, dy}{\sum_{j=1}^{m} w_j a_j(x) \int_{R^p} \delta(y - y_j)\, dy} \tag{18}$$

$$= \frac{\sum_{j=1}^{m} w_j a_j(x) y_j}{\sum_{j=1}^{m} w_j a_j(x)} \tag{19}$$

$$= \sum_{j=1}^{m} p_j(x) y_j(x) \tag{20}$$

$$= \sum_{j=1}^{m} p_j(x) B_j(x) \tag{21}$$

for the m variable convex coefficients

$$p_j(x) = \frac{w_j a_j(x)}{\sum_{j=1}^{m} w_i a_i(x)}. \tag{22}$$

The convex sum (21) defines a nonlinear map F: $R^n \to R^p$ if $m > 1$ and thus if the system has two or more rules. This holds even if each then-part set B_j is a linear map or matrix. No one has yet found a stability result for these general nonlinear-operator systems.

We next reverse the arrow of generality and briefly review special cases of the generalized SAM model in (21). These range from feedback models to the popular but often *ad hoc* "center of gravity" (COG) model of feedforward fuzzy systems.

Tanaka [38]–[41] has explored the special case of (21) when the operator B_j is not only a matrix but an n-by-n square matrix. Tanaka ignores the rule weights w_j in his *ad hoc* SAM model. So he implicity assumes that they all equal the same positive value: $w_1 = \cdots = w_m > 0$. Then the rule weights cancel from the SAM ratio in (19). Tanaka uses this simple SAM system to define the discrete autonomous dynamical system in (2) as a convex sum of vector-matrix products

$$x(k+1) = F(x(k)) \tag{23}$$

$$= \sum_{j=1}^{m} p_j(x(k)) B_j x(k). \tag{24}$$

The square matrix B_j can house the coefficients of a piecewise-polynomial or more complex set function b_j [38]. Triangles and trapezoids are examples of such continuous piecewise-polynomial set functions. The square matrix B_j can also house a separate control or forcing function $u_j(k)$. The continuous case gives the feedback fuzzy system in (1) as a like convex sum

$$\dot{x} = \sum_{j=1}^{m} p_j(x) B_j x. \tag{25}$$

The next section gives sufficient conditions for global asymptotic stability of the dynamical systems in (24) and (25).

A still less-general SAM case is the so-called TSK or TS case [34], [37]. Sugeno [33] and Terano [42] call this the "third inference method" for how a fuzzy system F maps an input x to an output $F(x)$. This method replaces B_j with a piecewise linear map or appropriate n-by-p matrix operator B_j in the unweighted SAM equation. This gives the feedforward fuzzy system F as a convex sum of linear functions

$$F(x) = \sum_{j=1}^{m} p_j(x) B_j(x) \tag{26}$$

$$= \frac{\sum_{j=1}^{m} a_j(x) f_j(x_1, \cdots, x_n)}{\sum_{j=1}^{m} a_j(x)} \tag{27}$$

$$= \frac{\sum_{j=1}^{m} a_j(x) \left[b_0^j + b_1^j x_1 + b_2^j x_2 + \cdots + b_n^j x_n \right]}{\sum_{j=1}^{m} a_j(x)} \tag{28}$$

in Sugeno's notation [33]. Then the jth rule has the form

$$\text{IF } X = A_j \quad \text{THEN } Y = b_0^j + b_1^j x_1 + b_2^j x_2 + \cdots + b_n^j x_n \tag{29}$$

where the then-part term describes a piecewise linear set function such as a triangle or trapezoid. Least squares or a Kalman filter can use sample data to pick and tune the coefficients [33]. The first SAM model (7) should technically apply here because the piecewise-linear sets have different areas or volumes $V_j(x)$ and centroids $c_j(x)$ and because these volumes and centroids change with each input x. The varying volumes $V_j(x)$ affect both the linguistic "meaning" of the then-part sets and how they weight the SAM output $F(x)$.

The simpler and more popular SAM's replace the then-part operator B_j: $R^n \to R^p$ with a simple and fixed fuzzy set $B_j \subset R^p$ or b_j: $R^p \to [0, 1]$. Then (21) and (7) reduce to the simple but popular fixed-parameter SAM system

$$F(x) = \frac{\sum_{j=1}^{m} w_j a_j(x) V_j c_j}{\sum_{j=1}^{m} w_j a_j(x) V_j}. \tag{30}$$

This model reduces to the still more popular COG fuzzy model

$$F(x) = \frac{\sum_{j=1}^{m} a_j(x) P_j}{\sum_{j=1}^{m} a_j(x)} \tag{31}$$

if the modes or "peaks" P_j of the then-part sets $B_j \subset R^p$ equal the then-part set centroids c_j and if the then-part sets B_j all have the same areas or volumes V_j and the same rule weights w_j: $P_j = c_j, V_1 = \cdots = V_m > 0$, and $w_1 = \cdots = w_m > 0$.

Mamdani [22]–[23] first put forth the COG model as an *ad hoc* way to convert m discrete fuzzy if-then rules into a simple control system F. The model is *ad hoc* because Mamdani and other engineers use a simple additive ratio like (31) but claim that they combine rules not with a sum but with pairwise maximum or union in accord with the so-called "extension principle" [33], [42], [52]. The centroid of the output set $B(x) = \cup_{j=1}^{m} B'_j = \cup_{j=1}^{m} a_j(x) B_j$ does not give the SAM ratio (30) or (31) or any other tractable form. Indeed such pairwise maxima drive B toward a fixed rectangle or cube as the number m of overlapping then-part sets B_j grows [17]. So the centroid of the output set $B(x)$ tends toward a constant value as m grows.

The trouble with the COG model is that it ignores the area or volume of the then-part set functions B_j. Then we can always replace the fuzzy sets B_j with nonfuzzy rectangles or cubes that have the same centroids or even with Kroeneker delta pulses or Dirac delta functions centered at the centroids. The latter switch amounts to putting $b_j(y) = \delta(y - c_j)$ for continuous fuzzy sets B_j. Then the SAM Theorem goes through as in the more general case of (12)–(21) above.

Learning or hand tuning can change the centroids and volumes V_j of the then-part sets B_j. The COG model does not permit this. Some engineers have extended the *ad hoc*

COG model to include a then-part variance parameter to act like a volume or rule weight. Suppose the m then-part sets B_j are scalar Gaussian bell curves. Then their set functions have unit area or $V_j = 1$. But we may still want to punish rules or give them less weight if their then-part bell-curve B_j has a larger variance σ_j^2. The SAM model (30) reduces to this weighted Gaussian COG model [48] if $w_j = 1/\sigma_j^2$.

The COG SAM reduces in turn to the popular radial basis function (RBF) systems of neural network theory. A simple Gaussian SAM gives both a COG model and the popular RBF model of Moody [24] and Specht [32]. Wang and Mendel [47] have recently restated this RBF model in fuzzy notation as a simple scalar Gaussian SAM F: $R^n \to R$

$$F(x) = \frac{\sum_{j=1}^{m} \overline{z}^j \left(\prod_{i=1}^{n} \mu_{A_i^j}(x_i) \right)}{\sum_{j=1}^{m} \left(\prod_{j=1}^{n} \mu_{A_i^j}(x_i) \right)}. \tag{32}$$

The SAM (30) reduces to (32) for independent Gaussian sets with product combination of if-part set functions of the form with scaling constant in the unit interval and for the following identifications:

$$y = z \tag{33}$$

$$a_j(x) = \prod_{i=1}^{n} a_j^i(x_i) \tag{34}$$

$$= \prod_{i=1}^{n} \mu_{A_i^j}(x_i)$$

$$V_j = 1 \tag{35}$$

$$c_j = \overline{z}^j. \tag{36}$$

The unity volume follows in (35) since the m then-part Gaussian sets extend over all of R. Equation (36) follows because the mode of a Gaussian set equals its centroid and Wang and Mendel use the mode definition "$\overline{z}^j$ is the point in R at which $\mu_{B^j}(z)$ achieves its maximum value." They further call the SAM convex coefficients $p_j(x)$ "fuzzy basis functions" in this Gaussian case even though they are not orthogonal. These new names add no new content to the RBF SAM in (32) or to the weighted RBF SAM with $w_j = 1/\sigma_j^2$.

Specht [32] arrived at the RBF model (32) as a special case of a Parzen probability density estimator. He observed that (32) had the form of a Gaussian conditional expectation. A like result holds for all of the above SAM models

$$F(x) = \text{Centroid}(B(x)) \tag{37}$$

$$= \frac{\int_{R^p} yb(x,y)\, dy}{\int_{R^p} b(x,y)\, dy} \tag{38}$$

$$= \int_{R^p} yp(y|x)\, dy \tag{39}$$

$$= E[Y|X = x] \tag{40}$$

for each $x \in R^n$. This holds because the ratio in (38) of the joint distribution to the marginal defines a proper conditional probability density

$$p(y|x) = \frac{b(x,y)}{\int_{R^p} b(x,y)\, dy} \tag{41}$$

even though $b(x,y) > 1$ may hold for integrable b. This result proves that *all* centroidal fuzzy systems are probabilistic systems [17], [21], [27].

The result also shows that the structure of the then-part sets B_j matters. A conditional variance $V[Y|X = x]$ describes the uncertainty in each fuzzy system output $F(x)$. Define the then-part probability density as the normalized set function: $p_{B_j}(y) = b_j(y)/V_j$. This gives the then-part variance as $\sigma_{B_j}^2 = \int_{-\infty}^{\infty} (y - c_j)^2 p_{B_j}(y)\, dy$. This leads in turn to a measure of the SAM uncertainty [20]–[21]

$$V[Y|X = x] = \sum_{j=1}^{m} p_j(x)\sigma_{B_j}^2 + \sum_{j=1}^{m} p_j(x)(c_j - F(x))^2. \tag{42}$$

The second term in (42) acts as a penalty term for rule interpolation in a SAM. The system output $F(x)$ has the most confidence if only one of the rules fires dead on. The COG case reduces the first term in (42) to the lone value σ^2. Two COG's can have the same first-order or $F(x)$ values if they have the same if-part sets and if they have then-part sets with the same centroids. But their outputs will differ in their system variance if the then-part set volumes or variances differ.

All of these additive fuzzy systems follow from the additive assumption $B(x) = \Sigma_{j=1}^{m} w_j B_j'(x)$. This additive scheme can also combine any number of feedforward or feedback fuzzy systems [20]. We just view the output set $B^k(k)$ of the kth fuzzy system F_k: $R^n \to R^p$ as a fired then-part set and weight it with a new system weight v_k. The new weights $v_1, \cdots, v_q$ need not sum to unity but they often do in practice [21]. This gives the total system output as the weighted sum of all rule firings: $\boldsymbol{B}(x) = \Sigma_{k=1}^{q} v_k B^k(x)$. Then the centroid of $\boldsymbol{B}$ gives the total system output $F(x)$ as a higher order SAM ratio. The fuzzy systems $F_1, \cdots, F_q$ need not each be additive in structure.

III. Stability of Generalized Additive Dynamical Systems

We first prove the asymptotic global stability of the unforced continuous generalized additive fuzzy system

$$\dot{x} = F(x) = \sum_{j=1}^{m} p_j(x) B_j x \tag{43}$$

from (25). A control input column vector u can steer the generalized SAM feedback system (43) through m control matrices C_j

$$\dot{x} = \sum_{j=1}^{m} p_j(x)[B_j x + C_j u]. \tag{44}$$

A like convex sum holds for the forced discrete generalized SAM in (24)

$$x(k+1) = \sum_{j=1}^{m} p_j(x(k))[B_j x(k) + C_j u(k)]. \tag{45}$$

The unforced case assumes $u = 0$ and lets us look at the stability of the equilibrium vector $x_e = \lim_{k\to\infty} x(k)$ if it exists. We take the equilibrium vector to be the origin: $x_e = 0$.

The equilibrium point x_e is *stable* in the sense of Lyapunov if small changes in the initial conditions lead to only small changes in the state trajectory [9], [46]: for all k_0 and all $\varepsilon > 0$ there is a $\delta > 0$ such that $||x(k_0) - x_e|| < \delta$ implies $||x(k) - x_e|| < \varepsilon$ for all $k \geq k_0$. The norm is the Euclidean norm: $||x||^2 = x_1^2 + \cdots + x_n^2$. The point x_e is *asymptotically stable* if it is stable and if it attracts the state trajectory: for all k_0 there is a $\delta' > 0$ such that $||x(k_0) - x_e|| < \delta'$ implies $\lim_{k\to\infty} ||x(k) - x_e|| = 0$. The equilibrium is *globally* asymptotically stable if we can pick δ' to be arbitrarily large. When these results hold they hold uniformly for the autonomous system (43) and its discrete version in (51).

We seek a smooth *Lyapunov function* L: $R^n \to R$ for the continuous feedback SAM model (43) that is positive definite $L(x) > 0$ when $x \neq 0$ and that obeys $L(x) = 0$ when $x = 0$ and that grows to infinity as the vector squared norm $||x||^2$ grows to infinity: $L(x) \to \infty$ as $x^T x \to \infty$. This holds if we take L as the quadratic form $x^T x$ or as the more general quadratic form $x^T P x$ for some n-by-n positive definite matrix P. Then standard results in Lyapunov stability theory [9] imply that the dynamical system (43) has a stable equilibrium $x_e = 0$ if $\dot{L} \leq 0$ and has a globally asymptotically stable equilibrium $x_e = 0$ if $\dot{L} < 0$ along system trajectories for all $x \neq 0$. A discrete Lyapunov function $L(x(k))$ leads to stability for the unforced version of the discrete unforced dynamical system in (51) if $\Delta L \leq 0$ and to global asymptotic stability if $\Delta L < 0$ along system trajectories.

We can now prove the main result of this paper. The generalized SAM in (43) is globally asymptotically stable if all the local rule matrices B_j are stable. A stable matrix B_j in the continuous case means that B_j is negative definite: $x^T B_j x < 0$ for all nonnull state vectors $x \neq 0$. This result extends the usual stability result for linear systems and reduces to it in the one-rule case when $m = 1$.

Theorem 1 (Continuous SAM Stability): The generalized feedback SAM system

$$\dot{x} = \sum_{j=1}^{m} p_j(x) B_j x \tag{46}$$

with convex coefficients

$$p_j(x) = \frac{w_j(x) a_j(x)}{\sum_{i=1}^{m} w_i(x) a_i(x)} \tag{47}$$

is globally asymptotically stable if each then-part rule matrix B_j is negative definite.

Proof: Choose the Lyapunov function L as the quadratic form $L(t) = x^T(t)x(t)$. Then

$$\dot{L} = 2x^T \dot{x} \tag{48}$$

$$= 2x^T \sum_{j=1}^{m} p_j(x) B_j x \tag{49}$$

$$= 2 \sum_{j=1}^{m} p_j(x) x^T B_j x. \tag{50}$$

At all times t there is at least one convex coefficient p_k that obeys $p_k(x(t)) > 0$. So $\dot{L} < 0$ along trajectories if each then-part matrix B_j is negative definite. Q.E.D.

The proof of Theorem 1 extends the stability proof of the continuous-time linear system or one-rule case. The feedback SAM F in (43) is nonlinear since the convex coefficients $p_j(x)$ change with each input x. The fact that at each time t at least one term obeys $p_k(x(t)) > 0$ lets us treat the convex sum of matrices as if it were a simple sum with constant coefficients. This does not hold in the discrete case.

We next look at the stability of the unforced discrete feedback fuzzy systems in (44)

$$x(k+1) = F(x(k)) = \sum_{j=1}^{m} p_j(x(k)) B_j x(k). \tag{51}$$

A stable matrix B_j in this discrete case means that the linear subsystem $x(k+1) = B_j x(k)$ is asymptotically stable and thus that all n eigenvalues $\lambda_j^1, \cdots, \lambda_j^n$ of B_j lie in the unit circle in the complex z-plane: $\lim_{k\to\infty} B_j^k = \varnothing$ if $|\lambda_j^i| < 1$ for all i.

A key question is whether the discrete system (51) is stable if each rule matrix B_j is stable. Tanaka [38]–[39] first showed that the answer is no for a slightly simpler unweighted SAM model. But we show below that the answer is yes in the special case where each rule matrix B_j is not only stable but diagonal. Tanaka showed that stability did not hold for a simple two-rule system with the two then-part "set" matrices

$$B_1 = \begin{pmatrix} 1 & -\frac{1}{2} \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad B_2 = \begin{pmatrix} -1 & -\frac{1}{2} \\ 1 & 0 \end{pmatrix}.$$

Matrix B_1 has the two eigenvalues $\lambda_1 = \frac{1}{2} \pm i\,\frac{1}{2}$, and B_2 has the two eigenvalues $\lambda_2 = -\frac{1}{2} \pm i\,\frac{1}{2}$. All four eigenvalues lie in the unit circle. So both matrices or linear subsystems are stable. Tanaka showed that the unweighted convex sum (51) is unstable for trapezoidal then-part sets A_1 and A_2 and constant or unity rule weights $w_1 = w_2 > 0$. He did find a sufficient condition for stability of (51). We restate it here in the more general SAM case without proof.

Theorem 2 [Discrete SAM Stability (Tanaka)]: The generalized discrete-time feedback SAM system

$$x(k+1) = \sum_{j=1}^{m} p_j(x(k)) B_j x(k) \tag{52}$$

with convex coefficients

$$p_j(x(k)) = \frac{w_j(x(k)) a_j(x(k))}{\sum_{i=1}^{m} w_i(x(k)) a_i(x(k))} \tag{53}$$

is globally asymptotically stable if there exists a common positive definite matrix P such that all m of the then-part matrices $B_j^T P B_j - P$ are negative definite.

There is no known way to find such a common positive definite matrix P. The odds of finding such a P fall with each new rule we add to the system. Exponential rule explosion offers little hope of finding such a P for large-scale systems.

We next present a corollary to Theorem 2 that gives a sufficient condition for the identity choice $P = I$. Then the discrete system (52) will be globally asymptotically stable if each rule matrix B_j is stable and if each B_j is *diagonal* with real coefficients

Corollary (*Discrete Diagonal SAM Stability*): The generalized discrete-time feedback SAM system

$$x(k+1) = \sum_{j=1}^{m} p_j(x(k)) B_j x(k) \tag{54}$$

is globally asymptotically stable if all m then-part matrices B_j are diagonal and stable.

Proof: The diagonal matrix $B_j = \mathrm{Diag}[b_j(1,1), \cdots, b_j(n,n)]$ is symmetric and lists its eigenvalues along its main diagonal. So the choice $P = I$ in Theorem 2 gives

$$B_j^T P B_j - P = B_j^2 - I. \tag{55}$$

The diagonal matrix $B_j^2 - I$ is negative definite iff all of its eigenvalues are negative. This holds iff each diagonal entry obeys $b_j^2(i,i) - 1 < 0$ iff $b_j^2(i,i) < 1$ iff $|b_j(i,i)| < 1$. The last condition $|b_j(i,i)| < 1$ holds for all i and thus for all eigenvalues of the matrix B_j. This is just the definition of stability for matrix B_j. Q.E.D.

This practical result holds for any positive definite matrix P of the form $P = cI$ with constant $c > 0$. For then $B_j^T P B_j - P = cB_j^2 - cI$. So the diagonal condition $cb_j^2(i,i) - c < 0$ still leads to the stability condition $|b_j(i,i)| < 1$.

IV. Conclusions

We have shown that the global asymptotic stability of generalized additive fuzzy systems can depend on the matrix structure of the rules. The same proofs do not go through if we replace the matrices with fuzzy set functions as in the SAM system in (7) or with more general nonlinear operators. The stability of these dynamical systems remains an open and active area of research.

Fuzzy research has also yet to produce a practical definition for the partial stability of fuzzy or nonfuzzy dynamical systems. Other research [3], [11], [45] has raised this question but not answered it. A frequency approach might take this measure of stability as some limiting proportion of initial conditions in a region that lead to standard stability. A qualitative approach might cast the measure in terms of statistical robustness or relax the binary constructs that lie beneath the standard definition of system stability.

Still broader research questions involve the stability and approximation power of fuzzy (autonomous) dynamical systems $\dot{x} = F(x)$ that consist of a knowledge net of interlocking rules or rule cycles. Fuzzy cognitive maps [2], [5], [14]–[16], [29]–[31], [35]–[36], [53] offer one such class of knowledge or causal networks. These dynamical networks have node dynamics that resemble the neuronal dynamics of some asymmetric feedback neural networks. But cognitive maps combine in nonneural ways and learn with causal laws that differ from synaptic learning laws. They involve few stability results when causal learning laws change their rule structure [15]–[16]. More complex cognitive maps result when feedforward fuzzy systems like SAM's or other cognitive maps define the I/O structure of a cognitive map's nodes or fuzzy sets. No one has found a stability result for such hierarchical fuzzy dynamical systems.

These knowledge networks offer one way to combat fuzzy rule explosion in dynamical approximation. But they do so at the risk of instability, computational intractability, and system inscrutability. These knowledge networks $\dot{x} = F(x)$ use a fixed number of rules or edges to approximate a dynamical system $\dot{x} = f(x)$. The n fuzzy nodes or sets of fuzzy cognitive maps and other knowledge networks define their global system trajectory as a path in the n-dimensional unit hypercube $[0,1]^n$. Each parameter choice carves this fuzzy cube [21] into k attractors that can differ in shape as well as in type. A repeller boundary may separate an attractor basin that contains a fixed point or limit cycle or limit torus from a basin that contains an aperiodic equilibrium or chaotic attractor. We may want the fuzzy knowledge network $\dot{x} = F(x)$ and its fixed number n of nodes or rules to approximate both qualitative and quantitative aspects of some known or unknown dynamical system $\dot{x} = f(x)$.

Simple mean-squared approximation $F \approx f$ is not likely to ensure that the fuzzy dynamical approximator F has the same qualitative structure as the approximand dynamical system f has. The attractors in the fuzzy system $\dot{x} = F(x)$ should approximate the number, shape, and type of the attractors in the approximand dynamical system $\dot{x} = f(x)$. This can involve the search for system embeddings and mutual information or the search for Lyapunov exponents and fractal dimensions and other invariants of the system's orbits and attractors. Research in temporal neural networks [25], [28] has not solved or even often addressed the like problems of neural dynamical approximation. Fuzzy dynamical approximation promises to remain an open research area well into the next century if not well into the next millennium.

Appendix
SAM Theorem and Supervised Learning

This appendix presents and derives the basic SAM Theorem used in Section II. It also shows how supervised gradient descent can tune or learn the parameters in the SAM Theorem. The last part of the Appendix derives scalar learning laws for simple but popular fuzzy set functions. Simulations have shown that these adaptive set functions can quickly approximate a wide range of sampled functions.

The SAM theorem assumes that the fuzzy system $F: R^n \to R^p$ stores m if–then rules. The m then-part sets $B_j(x)$ can change with each input vector x and thus so can the then-part volumes $V_j(x)$ and centroids $c_j(x)$.

SAM Theorem: Suppose the fuzzy system F: $R^n \to R^p$ is a SAM: $F(x) = \text{Centroid}(B(x)) = \text{Centroid}(\Sigma_{j=1}^m w_j a_j(x) B_j(x))$. Then $F(x)$ is a convex sum of the m then-part set centroids

$$F(x) = \frac{\sum_{j=1}^m w_j a_j(x) V_j(x) c_j(x)}{\sum_{j=1}^m w a_j(x) V_j(x)} \tag{A1}$$

$$= \sum_{j=1}^m p_j(x) c_j(x). \tag{A2}$$

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

$$p_j(x) = \frac{w_j a_j(x) V_j(x)}{\sum_{k=1}^m w_k a_k(x) V_k(x)}. \tag{A3}$$

$V_j(x)$ is the finite positive volume (or area if $p = 1$ in the range space R^p), and $c_j(x)$ is the centroid of then-part set $B_j(x)$

$$V_j(x) = \int_{R^p} b_j(x, y_1, \cdots, y_p)\, dy_1 \cdots dy_p > 0 \tag{A4}$$

$$c_j(x) = \frac{\int_{R^p} y b_j(x, y_1, \cdots, y_p)\, dy_1 \cdots dy_p}{\int_{R^p} b_j(x, y_1, \cdots, y_p)\, dy_1 \cdots dy_p}. \tag{A5}$$

The popular scalar case of $p = 1$ reduces (A4) and (A5) to

$$V_j(x) = \int_{-\infty}^{\infty} b_j(x, y)\, dy \tag{A6}$$

$$c_j(x) = \frac{\int_{-\infty}^{\infty} y b_j(x, y)\, dy}{\int_{-\infty}^{\infty} b_j(x, y)\, dy}. \tag{A7}$$

Proof: The theorem follows by expanding the centroid of the combined output set $B(x)$ and invoking the SAM assumption in the hypothesis of the theorem to rearrange terms

$$F(x) = \text{Centroid}(B(x)) = \frac{\int_{R^p} y b(x, y)\, dy}{\int_{R^p} b(x, y)\, dy} \tag{A8}$$

$$= \frac{\int_{R^p} y \sum_{j=1}^m w_j b'_j(x, y)\, dy}{\int_{R^p} \sum_{j=1}^m w_j b'_j(x, y)\, dy} \tag{A9}$$

$$= \frac{\int_{R^p} y \sum_{j=1}^m w_j a_j(x) b_j(x, y)\, dy}{\int_{R^p} \sum_{j=1}^m w_j a_j(x) b_j(x, y)\, dy} \tag{A10}$$

$$= \frac{\sum_{j=1}^m w_j a_j(x) \int_{R^p} y b_j(x, y)\, dy}{\sum_{j=1}^m w_j a_j(x) \int_{R^p} b_j(x, y)\, dy} \tag{A11}$$

$$= \frac{\sum_{j=1}^m w_j a_j(x) V_j(x) \frac{\int_{R^p} y b_j(x, y)\, dy}{V_j(x)}}{\sum_{j=1}^m w_j a_j(x) V_j(x)} \tag{A12}$$

$$= \frac{\sum_{j=1}^m w_j a_j(x) V_j(x) c_j(x)}{\sum_{j=1}^m w_j a_j(x) V_j(x)}. \tag{A13}$$

Q.E.D.

Supervised learning changes SAM parameters with error data. The error at each time t is the desired system output minus the actual SAM output: $\varepsilon_t = d_t - F(x_t)$. Then supervised gradient descent can learn or tune SAM systems [21] by changing the rule weights w_j in (A14) below. Or it can change the then-part volumes V_j, the then-part set centroids c_j, or the parameters of the if-part set functions a_j. The rule weight w_j enters the ratio form of the weighted SAM system

$$F(x) = \frac{\sum_{j=1}^m w_j a_j(x) V_j c_j}{\sum_{j=1}^m w_j a_j(x) V_j} \tag{A14}$$

in the same way as does the then-part volume V_j in (A1). So both have the same learning law if we replace the nonzero weight w_j with the nonzero volume V_j

$$w_j(t+1) = w_j(t) - \mu_t \frac{\partial E}{\partial w_j} \tag{A15}$$

$$= w_j(t) - \mu_t \frac{\partial E}{\partial F} \frac{\partial F}{\partial w_j} \tag{A16}$$

$$= w_j(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{w_j(t)} [c_j - F(x_t)] \tag{A17}$$

for instantaneous squared error $E_t = \frac{1}{2}(d_t - F(x_t))^2$ with desired-minus-actual error $\varepsilon_t = d_t - F(x_t)$. Then the volumes change in the same way if they do not depend on the weights (which they do in most ellipsoidal learning schemes [4], [21])

$$V_j(t+1) = V_j(t) - \mu_t \frac{\partial E}{\partial V_j} \tag{A18}$$

$$= V_j(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{V_j(t)} [c_j - F(x_t)] \tag{A19}$$

for some decreasing sequence of learning weights $\{\mu_t\}$. The learning law (A17) follows since $(\partial E / \partial w_j) = -\varepsilon$ and since (A20)–(A22) (shown at the bottom of the next page) from the SAM Theorem.

The centroid c_j in the SAM Theorem has the simplest learning law

$$c_j(t+1) = c_j(t) - \mu_t \frac{\partial E}{\partial F} \frac{\partial F}{\partial c_j} \tag{A23}$$

$$= c_j(t) + \mu_t \varepsilon_t p_j(x_t). \tag{A24}$$

So the terms w_j, V_j, and c_j do not change when $p_j \approx 0$ and thus when the jth if-part set barely fires: $a_j(x_t) \approx 0$. The centroid learning law (A24) is a convex-weighted version of the classical Widrow–Hoff LMS learning law [16].

Tuning the if-part sets involves more computation since the update law contains at least one extra partial derivative. Suppose if-part set function a_j is a function of l parameters $a_j = a_j(m_j^1, \cdots, m_j^l)$. Then we can update each parameter with

$$m_j^k(t+1) = m_j^k(t) - \mu_t \frac{\partial E}{\partial F} \frac{\partial F}{\partial a_j} \frac{\partial a_j}{\partial m_j^k} \tag{A25}$$

$$= m_j^k(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{a_j(x_t)} [c_j - F(x_t)] \frac{\partial a_j}{\partial m_j^k}. \tag{A26}$$

Factored joint if-part sets $a_j = \prod_{i=1}^n a_j^i(m_j^i, d_j^i)$ further complicate the learning law. Here each joint set factors into n scalar fuzzy sets. Each scalar fuzzy set function depends for simplicity only on its two mean-like and dispersion-like parameters m_j^i and d_j^i. Then the chain of partial derivatives in (A25) gains a fourth term

$$m_j^k(t+1) = m_j^k(t) - \mu_t \frac{\partial E}{\partial F} \frac{\partial F}{\partial a_j} \frac{\partial a_j}{\partial a_j^k} \frac{\partial a_j^k}{\partial m_j^k}. \tag{A27}$$

Product factorization gives this new term as $\partial a_j(x)/\partial a_j^k = \prod_{i \neq k}^n a_j^i(x_i)$. This new term multiplies the second term on the right-hand side of (A26). It also reflects how factorization assumes that the if-part components do not correlate but combine independently of one another.

Exponential if-part set functions can reduce the learning complexity. They have the form $a_j = e^{f_j(m_j^1, \cdots, m_j^t)}$ and obey $\partial a_j/\partial m_j^k = a_j(\partial f_j(m_j^1, \cdots, m_j^l)/\partial m_j^k)$. Then the parameter update law (A26) simplifies to

$$m_j^k(t+1) = m_j^k(t) + \mu_j \varepsilon_t p_j(x_t)[c_j - F(x_t)] \frac{\partial f_j}{\partial m_j^k}. \tag{A28}$$

This can arise for independent exponential or Gaussian sets $a_j(x) = \prod_{i=1}^n e^{f_j^i(x_i)} = e^{\sum_{i=1}^n f_j^i(x_i)} = e^{f_j(x)}$. The exponential $a_j(x) = e^{\sum_{i=1}^n u_j^i(v_j^i - x_i)}$ has parameter partial derivatives $\partial f_j/\partial u_j^k = v_j^k - x_k(t)$ and $\partial f_j/\partial v_j^k = u_j^k$. This gives the exponential learning laws

$$u_j^k(t+1) = u_j^k(t) + \mu_t \varepsilon_t p_j(x)[c_j - F(x)](v_j^k - x_k) \tag{A29}$$

$$v_j^k(t+1) = v_j^k(t) + \mu_t \varepsilon_t p_j(x)[c_j - F(x)]u_j^k \tag{A30}$$

for vector inputs $x = (x_1, \cdots, x_n) \in R^n$.

The Gaussian $a_j(x) = e^{-(1/2)\sum_{i=1}^n (x_i - m_j^i/\sigma_j^i)^2}$ has mean partial derivative $\partial f_j/\partial m_j^k = x_k - m_j^k/(\sigma_j^k)^2$ and variance partial derivative $\partial f_j/\partial \sigma_j^k = (x_k - m_j^k)^2/(\sigma_j^k)^3$. This gives the Gaussian learning laws

$$m_j^k(t+1) = m_j^k(t) + \mu_t \varepsilon_t p_j(x)[c_j - F(x)] \frac{x_k - m_j^k}{(\sigma_j^k)^2} \tag{A31}$$

$$\sigma_j^k(t+1) = \sigma_j^k(t) + \mu_t \varepsilon_t p_j(x)[c_j - F(x)] \frac{(x_k - m_j^k)^2}{(\sigma_j^k)^3}. \tag{A32}$$

These Gaussian laws are the familiar learning laws of RBF's. Gaussian set functions reduce the SAM model to Specht's [32] RBF network or "generalized regression neural network." The Gaussian learning laws offer a good way to cheat when tuning the much simpler (but nondifferentiable) triangle if-part sets found in many applications. We can use the smooth update laws (A31) and (A32) to update triangles or trapezoids or other sets by viewing their centers and widths as the Gaussian means and variances. We can also derive direct piecewise learning laws for these simple if-part set functions.

We can derive other supervised SAM learning laws from other set functions. These include two strong competitors to

---

$$\frac{\partial F}{\partial w_j} = \frac{a_j(x)V_j c_j \sum_{i=1}^m w_i a_i(x)V_i - a_j(x)V_j \sum_{i=1}^m w_i a_i(x)V_i c_i}{\left(\sum_{i=1}^m w_i a_i(x)V_i\right)^2} \tag{A20}$$

$$= \frac{w_j a_j(x)V_j}{w_j \sum_{i=1}^m w_i a_i(x)V_i} \left[\frac{c_j \sum_{i=1}^m w_i a_i(x)V_i}{\sum_{i=1}^m w_i a_i(x)V_i} - \frac{\sum_{i=1}^m w_i a_i(x)V_i c_i}{\sum_{i=1}^m w_i a_i(x)V_i} \right] \tag{A21}$$

$$= \frac{p_j(x)}{w_j} [c_j - F(x)] \tag{A22}$$

the Gaussian SAM laws. The first is the set of Cauchy SAM learning laws

$$m_j(t+1) = m_j(t) + \mu_t \varepsilon_t \frac{p_j(x)}{a_j(x)} [c_j - F(x)] \cdot \left(\frac{x - m_j}{d_j^2} \right) a_j(x) \quad \text{(A33)}$$

$$d_j(t+1) = d_j(t) + \mu_t \varepsilon_t \frac{p_j(x)}{a_j(x)} [c_j - F(x)] \cdot \left(\frac{x - m_j}{d_j} \right)^2 \frac{a_j(x)}{d_j}. \quad \text{(A34)}$$

These laws tune the "mean" and dispersion terms of the generalized Cauchy set functions a_j: $R \to R^+$ of the form

$$a_j(x) = \frac{1}{1 + \left(\frac{x - m_j}{d_j} \right)^2}. \quad \text{(A35)}$$

Cauchy and Gaussian probability densities belong to the same family of alpha-stable densities [10], [21]. But Cauchy variables do not have finite variances or higher moments. They do have nearly the same bell-curve shape as the Gaussian bell curves. Their ratio form gives an easier set of if-part sets to compute with than do the exponentials of Gaussians.

The other set of learning laws are the sinc SAM learning laws

$$\frac{\partial a_j}{\partial m_j} = \left(a_j(x) - \cos\left(\frac{x - m_j}{d_j} \right) \right) \frac{1}{x - m_j}, \quad \text{if } x \neq m_j \quad \text{(A36)}$$

$$\frac{\partial a_j}{\partial m_j} = 0, \quad \text{if } x = m_j \quad \text{(A37)}$$

$$\frac{\partial a_j}{\partial d_j} = \left(a_j(x) - \cos\left(\frac{x - m_j}{d_j} \right) \right) \frac{1}{d_j}. \quad \text{(A38)}$$

These laws tune the popular sinc function of signal processing:

$$a_j(x) = \frac{\sin\left(\frac{x - m_j}{d_j} \right)}{\frac{x - m_j}{d_j}}. \quad \text{(A39)}$$

Simulations show that sinc SAM's tend to converge faster and more accurately than do Gaussian or Cauchy SAM's. SAM systems that use factored joint sinc or Cauchy set functions must include the fourth partial derivative in (A27) in their learning laws. Sinc set functions have an extended range that lets them take on negative values. These values and the infinite lobes in the sinc function may have no linguistic "meaning."

REFERENCES

[1] M. Black, "Vagueness: An exercise in logical analysis," *Phil. Sci.*, vol. 4, pp. 427–455, 1937.

[2] P. Craiger, "Causal structure, model inferences, and fuzzy cognitive maps: Help for the behavioral scientist," in *Proc. 1994 World Congr. Neural Networks (INNS WCNN-94)*, vol. I, pp. 836–841.

[3] M. DeGlas, "Invariance and stability of fuzzy systems," *J. Math. Anal. Applicat.*, vol. 99, no. 2, pp. 299–319, 1981.

[4] J. A. Dickerson and B. Kosko, "Fuzzy function learning with covariance ellipsoids," in *Proc. IEEE Int. Conf. Neural Networks (IEEE ICNN-93)*, pp. 1162–1167.

[5] ———, "Virtual worlds as fuzzy cognitive maps," *Presence*, vol. 3, pp. 173–189, Spring 1994.

[6] J. A. Dickerson, H. M. Kim, and B. Kosko, "Fuzzy control of platoons of smart cars," in *Proc. IEEE FUZZ-94*, pp. 1632–1637.

[7] E. Hartman, J. D. Keeler, and J. Kowalski, "Layered neural networks with Gaussian hidden units as universal approximators," *Neural Computat.*, vol. 2, pp. 210–215, 1990.

[8] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[9] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[10] H. M. Kim and B. Kosko, "Fuzzy prediction and filtering in impulsive noise," *Fuzzy Sets Syst.*, vol. 77, pp. 15–33, Jan. 15, 1996.

[11] J. B. Kiszka, M. M. Gupta, and P. N. Nikiforuk, "Energetistic stability of fuzzy dynamic systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 783–792, Nov. 1985.

[12] G. J. Klir and T. A. Folger, *Fuzzy Sets, Uncertainty, and Information*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[13] S.-G. Kong and B. Kosko, "Adaptive fuzzy systems for backing up a truck-and-trailer," *IEEE Trans. Neural Networks*, vol. 3, pp. 211–223, Mar. 1992.

[14] B. Kosko, "Fuzzy cognitive maps," *Int. J. Man-Mach. Stud.*, vol. 24, pp. 65–75, Jan. 1986.

[15] ———, "Hidden patterns in combined and adaptive knowledge networks," *Int. J. Approx. Reasoning*, vol. 2, no. 4, pp. 377–393, 1988.

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

[17] ———, "Fuzzy systems as universal approximators," *IEEE Trans. Comput.*, vol. 43, pp. 1329–1333, Nov. 1994; an earlier version appears in the *Proc. 1st IEEE Int. Conf. Fuzzy Syst. (IEEE FUZZ-92)*, pp. 1153–1162.

[18] B. Kosko and S. Isaka, "Fuzzy logic," *Sci. Amer.*, vol. 269, pp. 76–81, July 1993.

[19] B. Kosko, "Optimal fuzzy rules cover extrema," *Int. J. Intell. Syst.*, vol. 10, pp. 249–255, Feb. 1995; an earlier version appears in the *Proc. 1994 World Congr. Neural Networks (INNS WCNN-94)*, vol. I, pp. 697–698.

[20] ———, "Combining fuzzy systems," in *Proc. IEEE FUZZ-95*, vol. IV, pp. 1855–1863.

[21] ———, *Fuzzy Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[22] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Stud.*, vol. 7, pp. 1–13, 1977.

[23] E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Trans. Comput.*, vol. C-26, pp. 1182–1191, Dec. 1977.

[24] J. Moody and C. Darken, "Fast learning in networks of locally tuned processing units," *Neural Computat.*, vol. 1, pp. 281–294, 1989.

[25] K. S. Narendra and A. M. Annaswamy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, Mar. 1990.

[26] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN-89)*, vol. II, pp. 357–363.

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

[28] M. M. Polycarpou and P. A. Ioannou, "Identification and control of nonlinear systems using neural network models: Design and stability analysis," Dept. Elect. Eng. Syst., Univ. Southern California, Los Angeles, EE Rep. 91-09-01, 1991. A shorter version of this paper appeared as M. M. Polycarpou and P. A. Ioannou, "Modeling, identification, and stable adaptive control of continuous-time nonlinear dynamical systems using neural networks," in *Proc. 1992 Amer. Contr. Conf.*, pp. 36–40.

[29] R. Satur, Z. Liu, and M. Gahegan, "Multi-layered FCM's applied to context dependent learning," in *Proc. IEEE FUZZ-95*, vol. II, pp. 561–568.

[30] M. Schneider, E. Shnaider, A. Kandel, and G. Chew, "Constructing fuzzy cognitive maps," in *Proc. IEEE FUZZ-95*, vol. IV, pp. 2281–2288.

[31] P. C. Silva, "Fuzzy cognitive maps over possible worlds," in *Proc. IEEE FUZZ-95*, vol. II, pp. 555–560.

[32] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural Networks*, vol. 4, pp. 549–557, Dec. 1991.

[33] M. Sugeno, "An introductory survey of fuzzy control," *Inf. Sci.*, vol. 36, pp. 59–83, 1985.

[34] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets Syst.*, vol. 28, pp. 15–33, 1988.

[35] W. R. Taber, "Knowledge processing with fuzzy cognitive maps," *Expert Syst. Applicat.*, vol. 2, pp. 82–87, Feb. 1991.

[36] W. R. Taber and M. Siegel, "Estimation of expert weights using fuzzy

cognitive maps," in *Proc. IEEE 1987 Int. Conf. Neural Networks (IEEE ICNN-87)*, vol. II, pp. 319–326.
[37] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, Jan. 1985.
[38] K. Tanaka and M. Sugeno, "Stability analysis and design of fuzzy control systems," *Fuzzy Sets Syst.*, vol. 45, pp. 135–156, Jan. 24, 1992.
[39] K. Tanaka and M. Sano, "A robust stabilization problem of fuzzy control systems and its application to backing up control of a truck-trailer," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 119–134, May 1994.
[40] K. Tanaka and K. Yoshioka, "Design of fuzzy controller for backer-upper of a five-trailers and truck," in *Proc. 1995 IEEE Int. Conf. Fuzzy Syst. (FUZZ-95)*, pp. 1543–1548.
[41] K. Tanaka, "Stability and stabilizability of fuzzy-neural-linear control systems," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 438–447, Nov. 1995.
[42] T. Terano, K. Asai, and M. Sugeno, *Fuzzy Systems Theory and Its Applications*. New York: Academic, 1992.
[43] D. Tilbury, M. M. Murray, and S. S. Sastry, "Trajectory generation for the N-trailer problem using Goursat normal form," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 802–819, May 1995.
[44] R. M. Tong, "A control engineering review of fuzzy systems," *Automatica*, vol. 13, pp. 559–569, 1977.
[45] ———, "Some properties of fuzzy feedback systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, pp. 327–331, June 1980.
[46] K. S. Tsakalis and P. A. Ioannou, *Linear Time-Varying Systems: Control and Adaptation*. Englewood Cliffs: Prentice-Hall, 1993.
[47] L. 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.
[48] L. Wang, "Design and analysis of fuzzy identifiers of nonlinear dynamic systems," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 11–23, Jan. 1995.
[49] F. A. Watkins, "The representation problem for additive fuzzy systems," in *Proc. IEEE FUZZ-95*, vol. I, pp. 117–122.
[50] R. R. Yager and D. P. Filev, "On the issue of defuzzification and selection based on a fuzzy set," *Fuzzy Sets Syst.*, vol. 55, pp. 255–273, 1993.
[51] L. A. Zadeh, "Fuzzy sets," *Inf. Contr.*, vol. 8, pp. 338–353, 1965.
[52] ———, *Fuzzy Sets and Applications: Selected Papers*, R. R. Yager, S. Ovchinnikov, R. M. Tong, and H. T. Nguyen, Eds. New York: Wiley, 1987.
[53] W. R. Zhang, S. S. Chen, and J. C. Bezdek, 'Pool 2: A generic system for cognitive map development and decision analysis," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 31–39, Jan. 1989.

Bart Kosko (M'85) received B.S. degrees in philosophy and economics from the University of Southern California (USC), Los Angeles, the M.S. degree in applied mathematics from the University of California at San Diego, and the Ph.D. degree in electrical engineering from the University of California at Irvine.

He is an Associate Professor of Electrical Engineering at USC and a past Director of USC's Signal and Image Processing Institute. He has authored the textbooks *Fuzzy Engineering* (Englewood Cliffs, NJ: Prentice-Hall, 1996), *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, (Englewood Cliffs, NJ: Prentice-Hall, 1991), and *Neural Networks for Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall, 1991), the trade book *Fuzzy Thinking* (New York: Hyperion, 1993), and the novel *Nanotime* (New York: Avon, 1997). He is the coeditor of the November 1998 *Proceedings of the IEEE* Special Issue on Intelligent Signal Processing.

Dr. Kosko is an elected Governor of the International Neural Network Society and has chaired and cochaired many neural and fuzzy systems conferences.