Additive Fuzzy Systems: From Generalized Mixtures to Rule Continua

Bart Kosko*
Department of Electrical Engineering, Signal and Image Processing Institute
University of Southern California, Los Angeles, California, 90089, USA

A generalized probability mixture density governs an additive fuzzy system. The fuzzy system's if-then rules correspond to the mixed probability densities. An additive fuzzy system computes an output by adding its fired rules and then averaging the result. The mixture's convex structure yields Bayes theorems that give the probability of which rules fired or which combined fuzzy systems fired for a given input and output. The convex structure also results in new moment theorems and learning laws and new ways to both approximate functions and exactly represent them. The additive fuzzy system itself is just the first conditional moment of the generalized mixture density. The output is a convex combination of the centroids of the fired then-part sets. The mixture's second moment defines the fuzzy system's conditional variance. It describes the inherent uncertainty in the fuzzy system's output due to rule interpolation. The mixture structure gives a natural way to combine fuzzy systems because mixing mixtures yields a new mixture. A separation theorem shows how fuzzy approximators combine with exact Watkins-based two-rule function representations in a higher-level convex sum of the combined systems. Two mixed Gaussian densities with appropriate Watkins coefficients define a generalized mixture density such that the fuzzy system's output equals any given real-valued function if the function is bounded and not constant. Statistical hill-climbing algorithms can learn the generalized mixture from sample data. The mixture structure also extends finite rule bases to continuum-many rules. Finite fuzzy systems suffer from exponential rule explosion because each input fires all their graph-cover rules. The continuum system fires only a special random sample of rules based on Monte Carlo sampling from the system's mixture. Users can program the system by changing its wave-like meta-rules based on the location and shape of the mixed densities in the mixture. Such meta-rules can help mitigate rule explosion. The meta-rules grow only linearly with the number of mixed densities even though the underlying fuzzy if-then rules can have high-dimensional if-part and then-part fuzzy sets. © 2017 Wiley Periodicals, Inc.

1. THE MIXTURE APPROACH TO ADDITIVE FUZZY SYSTEMS

We recast additive fuzzy systems in terms of their governing generalized probability mixture densities. Then a system's if-then rules correspond to the mixed probability densities. This approach lets users exploit the flexibility and expressive

*Author to whom all correspondence should be addressed; e-mail: kosko@usc.edu

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 00, 1–51 (2017)

View this article online at wileyonlinelibrary.com. • DOI 10.1002/int.21925

power of fuzzy-set rules and approximate reasoning. It also lets the theory of additive fuzzy systems exploit the extensive mathematical tools of modern probability theory and machine learning. So the mixture approach combines many of the best aspects of fuzzy inference and probabilistic modeling.

The mixture approach also gives a practical way to extend ordinary finite additive fuzzy systems to uncountably many rules or rule continua. This extension turns on the convex-sum structure of generalized mixture densities. The next section shows how the governing mixture density arises from the additive combination of the fired fuzzy if-then rules. An immediate corollary is that every such fuzzy system defines a conditional expectation. The corresponding conditional variance describes the fuzzy system's second-order uncertainty that results from the inherent uncertainty in the then-part sets and from rule interpolation. Different rules can produce the same conditional expectation but have different conditional variances and thus have different confidence levels for the same output. These conditional terms are just the first and second conditional moments of the generalized mixture. There are infinitely other higher moments and in principle all admit supervised learning laws. Another corollary is a new Bayes theorem that specifies which rules fire to which degree for each input. These results extend to arbitrary combinations of additive fuzzy systems and their governing generalized mixture. Their Bayes theorems specify which combined fuzzy system contributed to the output because they define a posterior distribution over each combined fuzzy system or over the rules that the combined systems use. The last section shows how the mixture structure defines and tunes wave-like meta-rules over the rule continua. These meta-rules tend to grow only linearly with the number of mixed densities in the meta-level mixture.

Fuzzy systems suffer from exponential rule explosion in high dimensions.[1–9] Rule explosion occurs if at least two fuzzy sets (such as `SMALL` and `LARGE`) cover each input and output axis because fuzzy if-then rules combine such sets into Cartesian products in the input-output product space. Figure 1 shows the rule patch that corresponds to a single if-then rule. The Cartesian products define a graph cover that grows exponentially with the number of input or output dimensions. The graph cover of a vector-valued fuzzy system $F : \mathbb{R}^n \to \mathbb{R}^p$ tends to require $O(k^{n+p-1})$ rules. Figure 2 shows this growth in rules when passing from the simple scalar fuzzy system $F : \mathbb{R} \to \mathbb{R}$ to the system $F : \mathbb{R}^2 \to \mathbb{R}$ with two input variables. The scalar fuzzy system $F : \mathbb{R} \to \mathbb{R}$ requires on the order of $O(k^{1+1-1}) = O(k)$ rules. The second system $F : \mathbb{R}^2 \to \mathbb{R}$ requires on the order of $O(k^{2+1-1}) = O(k^2)$ rules.

A linguistic fuzzy rule combines fuzzy-set adjectives into an if-then conditional statement: "If the air is `COOL` then set the air conditioner's motor speed to `SLOW`". The next section gives the formal details in terms of fuzzy-rule membership functions. A paragraph of such statements can define a fuzzy system. A fuzzy set $A \subset \mathbb{R}$ maps input values $x \in \mathbb{R}$ to degrees of membership in the unit interval $[0, 1]$. It thus defines a function $a : \mathbb{R} \to [0, 1]$ where $a(x) = \text{Degree}(x \in A)$.[10] The fuzzy set `COOL` of cool air temperatures maps each real temperature value t to a membership degree in $[0, 1]$. So all air temperatures are cool to some degree even if most are cool only to zero degree. Temperature acts here as a *linguistic variable* that takes on fuzzy-set adjective values such as `COOL` or `COLD` or `WARM`.[11,12] But neither the fuzzy sets nor

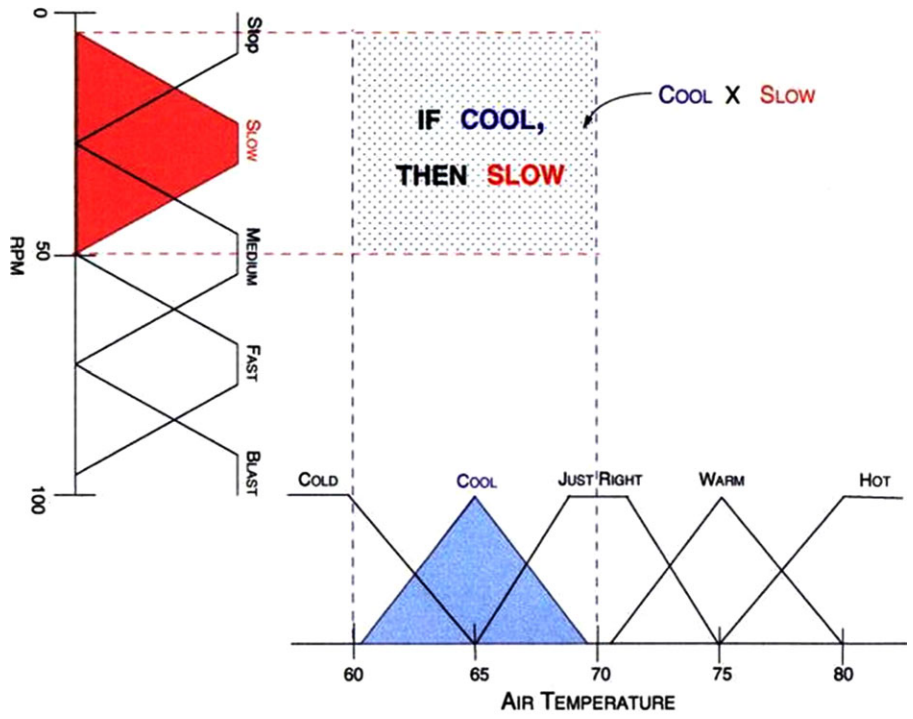

Figure 1. A fuzzy rule as a patch or fuzzy subset of an input-output product space. A Cartesian product combines the if-part fuzzy set COOL with the then-part fuzzy set SLOW to produce the linguistic rule "If the air temperature is COOL then set the air conditioner's motor speed to SLOW." The rule COOL $\times$ SLOW or $R_{\text{cool}\to\text{slow}}$ defines a patch or fuzzy subset of the input-output product space of temperature values and motor speeds. The rule also defines one of the mixed probability densities in the fuzzy system's mixture density $p(y|x)$. The figure shows only the base of the rule. It does not show the barn-like set of membership values above it.

the rules need have any tie to words or natural language. The sets simply quantize an input or output variable or axis.[5,8] Fuzzy function approximation may ignore the linguistic structure altogether for large-scale systems.[8,13,14] Simple unweighted additive fuzzy systems reduce to radial-basis-function networks if the if-part sets are Gaussian bell curves.[15]

Fuzzy rules define fuzzy patches in the input-output product space $X \times Y$. So the rule base gives a patch cover in the product space. The next section shows how to construct and fire these fuzzy subsets of the product space. The firing involves convolution with a delta spike for continuous fuzzy sets. Each vector input x_0 fires all the rules to some degree if we view the input x_0 as the delta spike $\delta(x - x_0)$. Correlations extend this rule firing to the more general case where the input is a fuzzy set A.

Consider the fuzzy rules that might control an air conditioner. The rules map fuzzy sets of temperatures to fuzzy sets of motor speeds. They associate control sets

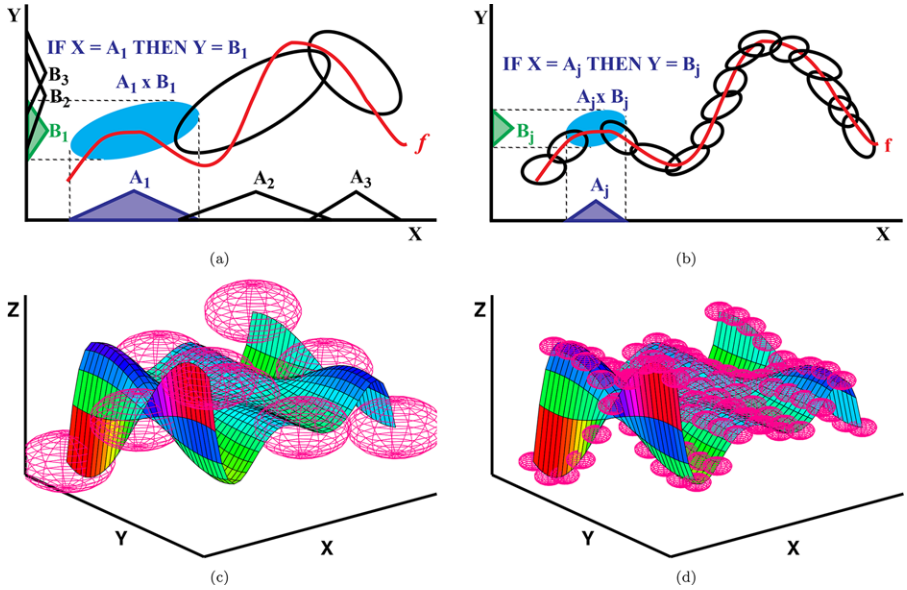

Figure 2. Fuzzy rule explosion in high dimensions. (a) shows how the fuzzy system uses fuzzy rule patches to approximate a function f by covering its graph for one input variable. (b) shows a finer rule-patch approximation in the 1-D case but still linear complexity. (c) shows the quadratic rule complexity when the function and fuzzy system have two inputs. (d) shows a finer rule-patch approximation in the 2-D case.

with temperature sets. Figure 1 shows that the linguistic rule "If the air is COOL then set the air conditioner's motor speed to SLOW" defines a fuzzy subset of the 2-D product space of air temperatures and motor speeds. This rule defines the Cartesian product COOL$\times$SLOW of the triangular if-part fuzzy set COOL and the trapezoidal then-part fuzzy set SLOW. Each pair (t, m) of an input temperature value t and an output motor speed m belongs to the Cartesian product COOL$\times$SLOW to some degree. So the rule defines a fuzzy relation or a fuzzy subset of the product space $X \times Y$.

The pair (t, m) satisfies or belongs to the fuzzy-rule set function $r_{\text{cool}\to\text{slow}} : \mathbb{R} \times \mathbb{R} \to [0,\ 1]$ to degree $r_{\text{cool}\to\text{slow}}(t, m)$. The paper uses the term "set function" to refer to the multivalued indicator functions that define bivalent and fuzzy sets. The term does not refer to a set-valued function. So the rule $R_{\text{cool}\to\text{slow}}$ or COOL$\times$SLOW looks like a barn or hill of membership values that stands above the 2-D planar space.[1,7] The rule looks like a patch or rectangle if one views it from above as in Figure 1. A rule patch *geometrizes* a minimal knowledge unit because it geometrizes an if-then conditional or association between fuzzy sets. The rule patches need not be connected. They almost always are connected in practice because users almost always use connected fuzzy sets for both the if-part and then-part sets.

An additive fuzzy system adds the fired then-part sets $B_j(x)$ of the patch cover and then averages them. This gives rise to the system's governing mixture probability density $p(y|x)$: Each normalized fired then-part set $B_j(x)$ defines a probability density function $p_{B_j}(y|x)$. Theorem 1 states this mixture result for a

finite rule base of fuzzy if-then rules. Theorem 6 extends the result to the case of a rule continuum. The fired if-part set values combine with the fired then-part set values to produce the convex mixing weights $p_j(x)$. The governing mixture is a *generalized* mixture because the convex mixing weights $p_j(x)$ depend on the input x. So each new input x produces a new set of mixing weights $p_j(x)$ in the mixture $p(y|x) = p_1(x)p_{B_1}(y|x) + \cdots + p_m(x)p_{B_m}(y|x)$.

Corollary 1 shows that this generalized mixture structure leads at once to a Bayes theorem that defines the converse or posterior distribution $p(j|y, x)$ over all m rules given the input x and the resulting fuzzy system output $y = F(x)$. The posterior $p(j|y, x)$ states the degree or probability that the jth rule fired. The posterior gives the jth rule's relative magnitude or degree of firing or importance in the overall output. So the posterior density gives direct probabilistic insight into the workings of the otherwise opaque fuzzy system. It makes the fuzzy system an *interpretable* rule-based model.[17] Similar posterior densities show which combined fuzzy system or which of its rules contributed to the joint output $F(x)$ of an arbitrary combination or fusion of multiple additive fuzzy systems.

Theorem 2 characterizes the moment structure of the fuzzy system's generalized mixture. The expectation or first conditional moment of the generalized mixture $p(y|x)$ gives the fuzzy system output $F(x)$ as a realization $E[Y|X = x]$ of the conditional-expectation random variable $E[Y|X]$: $F(x) = E[Y|X = x]$. This expectation is in turn just the convex sum of the centroids c_j of the m then-part sets: $F(x) = p_1(x)c_1(x) + \cdots + p_m(x)c_m(x)$. The second conditional moment of $p(x|y)$ is the realized conditional variance $V[Y|X = x]$. It describes the second-order uncertainty of the fuzzy output $F(x)$ and decomposes into two convex sums. The first sum describes the inherent uncertainty in the then-part sets B_j. The second sum defines a quadratic penalty term for rule interpolation. The conditional variance involves little more computation than computing the output itself and warrants much wider use in actual fuzzy applications. Higher-order conditional moments produce multiple convex sums. Learning changes the if-part sets or the then-part sets and thereby changes the overall rule patch structure.

Fuzzy rule patches endow a fuzzy system with a graph-cover structure. This structure can make it easy to build simple fuzzy systems from words and conditional sentences. The graph cover also implies that additive fuzzy systems are universal function approximators.[2] These properties help explain the vast number of fuzzy-system applications in control and elsewhere.[7,18–21] But the same graph cover creates a systemic rule explosion in the input-output product space. This curse of dimensionality severely limits the number of input variables in fuzzy systems. Figure 2 shows how rule explosion begins when going from just one input variable to two.

A fuzzy system F approximates a function f by covering its graph with rule patches and then averaging the patches that overlap.[2] Panels (a) and (b) of Figure 2 show that the fuzzy approximation gets finer by using more and smaller rule patches. Panels (c) and (d) show the same thing for ellipsoidal rule patches[4] but for a function with two input variables. The averaging corresponds to taking the centroid of the summed fired rule then-parts. Data clusters can estimate the rule patches in unsupervised learning.[1,22,23] Supervised learning can further shape and tune the rules.[4,5,8,13]

Additive fuzzy systems can *uniformly* approximate any continuous function on a compact set.[2] A uniform approximation lets the user pick an error tolerance level $\varepsilon > 0$ in advance. The user can be sure that the error in the approximation is less than ε for *all* input values x: $|F(x) - f(x)| < \varepsilon$ for all vector inputs x. The crucial point is that the x values do not depend on the choice of ε.

A uniform approximation may require a prohibitive number of rules if the input dimension n is large for the vector inputs $x \in \mathbb{R}^n$. Optimal lone rules cover the extrema of the function f.[3] They "patch the bumps." This patch-the-bump result suggests that rule learning can focus on estimating the zeros of the derivative map of f. Supervised learning tends to move rule patches quickly to cover extrema and then moves the extra rules in the graph cover to fill in between extrema.[4,5] But tuning or adapting the fuzzy rules only compounds the computational complexity.[8] Tuning a fuzzy system with just four independent variables often proves intractable in practice. The fuzzy system can also learn by sampling from a trained neural network and thereby convert the neural network to an approximate rule-based system.

The sets that make up the rules need not be fuzzy at all. Rectangular sets define ordinary binary sets. These binary sets still lead to a uniform approximation for enough rules. So the power of fuzzy systems in many cases may lie more in their graph-cover-based ability to approximate functions than in their use of fuzzy sets or their linguistic counterparts.

Additive fuzzy systems F can sometimes exactly *represent* f in the sense that $F(x) = f(x)$ for all x. The Watkins Representation Theorem states the remarkable result that an additive fuzzy system F with just *two* rules can represent *any* bounded real-valued function f of n real variables.[24,25] The Watkins result does require that the user know the functional form of f and build it into the structure of the if-part sets of the two rules. Watkins proved the representation result only for non-constant bounded scalar functions $f : \mathbb{R} \to \mathbb{R}$. But the proof still holds for a vector input $x \in \mathbb{R}^n$. It even holds for infinite-dimensional input spaces. The proof also extends directly to the case of vector outputs in $\mathbb{R}^p$. Then the representation requires exactly $2p$ rules if the vector components f_k are each bounded and not constant. So there is no loss of generality in stating results in the simpler scalar case $f : \mathbb{R} \to \mathbb{R}$. The Watkins Representation Theorem also holds for the so-called TSK or Takagi-Sugeno-Kang fuzzy systems[26,27] since they are special types of additive fuzzy systems.[5] TSK systems are convex sums of nonlinear systems. Most are convex sums of linear operators or matrices.

The Watkins two-rule representation has special force in modern Bayesian statistics because almost all common prior and likelihood probability densities are bounded and have known closed forms.[13] So exact representation allows additive fuzzy systems to absorb many closed-form Bayesian models. Fuzzy approximation also extends the Bayesian framework to rule-based priors and likelihoods that may have no known closed form.[13,14] The uniform approximation of both the prior and the likelihood gives a uniform approximation of the posterior.[13]

We extend the Watkins representation result in two different directions. Theorem 3 shows how to represent sums of bounded functions in combined fuzzy systems where some of the fuzzy systems are ordinary rule-based approximators. This gives a general technique for data fusion or knowledge combination. The fusion process is

hierarchical because mixing mixtures gives a new mixture. Then Theorem 4 shows how to find the generalized mixture $p(y|x)$ whose fuzzy-output average is exactly the bounded real function f. This special mixture density mixes just two densities in direct analogy to the Watkins Representation Theorem that represents f with just two rules. Two mixed normal densities will always suffice if their means lie at the bounded function's infimum and supremum. The result still holds approximately for small perturbations about these values. A related section shows how to use density estimators or the Expectation-Maximization algorithm to find the fuzzy system's generalized mixture from training data. Supervised learning laws can also tune the mixture or its moments or tune other parameters given suitable training data.

Extending a fuzzy system to infinitely many rules does not seem to make sense in the face of exponential rule explosion. Firing or tuning infinitely many rules would appear to define the ultimate form of rule explosion. This need not be if we replace the current default of firing all fuzzy rules for each input with firing only a carefully chosen random subset of rules. But firing too many rules can still lead to some form of rule explosion.

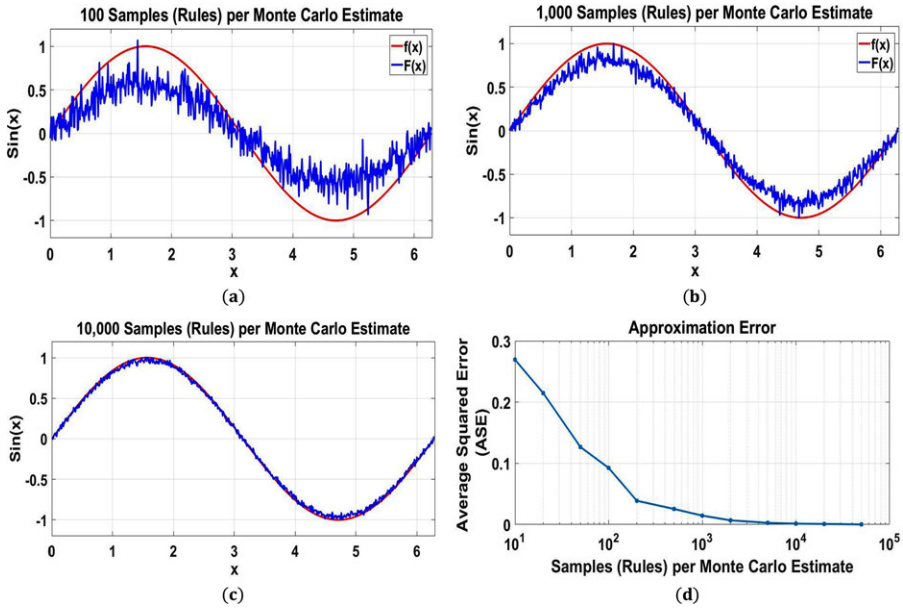

Figure 3. Monte Carlo fuzzy approximation of the target function $f(x) = \sin x$ for the additive rule-continuum fuzzy system F based on the mixture $p(y|x)$ in (167): $p(y|x) = \frac{1-\sin x}{2}\frac{1}{\sqrt{2\pi}}\exp[-\frac{(y+1)^2}{2}] + \frac{\sin x+1}{2}\frac{1}{\sqrt{2\pi}}\exp[-\frac{(y-1)^2}{2}]$. The blue lines in panels (a) - (c) show the additive fuzzy approximator $F(x)$ compared with the sine-wave approximand $f(x) = \sin x$ in red. Panel (a) shows the result of plotting the sample average based on 100 y values drawn at random uniformly from $(-1, 1)$ for each of 629 x values 0.01 apart. Panel (b) shows the better fuzzy approximation that results for 1,000 uniform samples at each input x value. Panel (c) shows the still finer approximation for 10,000 such samples. Panel (d) shows the approximation's slow inverse-square-root decay of the average squared error. Each plotted point averages 10 separate runs for a given number of random draws.

Working with rule continua lets the user define and tune wave-like meta-rules as a higher-level mixture density defined on a virtual rule continuum. The mixed densities define the meta-rules. Then statistical algorithms can tune the mixture meta-rules with training data. The number of such meta-rules tends to grow only linearly with the number of mixed densities. The new cost becomes the difficulty of computing system outputs $F(x)$ from the rule continuum. Some form of Monte Carlo sampling can ameliorate this burden. Figure 3 shows how more Monte Carlo sampling gives a better approximation of a target function f as the number of random samples increases.

The next section develops from first principles the generalized mixture density $p(y|x)$ that governs an additive fuzzy system.

2. THE GENERALIZED MIXTURE STRUCTURE OF ADDITIVE FUZZY SYSTEMS

A probabilistic mixture structure underlies all additive fuzzy systems. This section proves this core mixture result from first principles of fuzzy inference. We start with a review of finite mixture probability density functions (pdfs) and then develop the formal machinery of additive fuzzy systems.

2.1. Mixture Models as Convex Combinations of Probabilities

Mixture models are finite convex combinations of pdfs.[28–30] A convex combination mixture of m pdfs $p_1, \ldots, p_m$ gives a new pdf p:

$$p(y) = \sum_{j=1}^{m} \pi_j p_j(y). \tag{1}$$

The nonnegative *mixing weights* $\pi_1, \ldots, \pi_m$ sum to unity: $\sum_{j=1}^{m} \pi_j = 1$. The mixture density p has m modes if the m pdfs p_j are unimodal and sufficiently spread out.

A mixture model lets the user define and tune a wide range of multimodal pdfs by mixing simple and well-behaved pdfs. This convex sum can model taking random samples from a population made up of m subpopulations. The m subpopulations can consist of m words or images or other patterns. The theorems below show that a mixture describes the m rules in a fuzzy system or the q different fuzzy systems in a combined fuzzy system. Mixtures extend to mixing denumerably and uncountably many pdfs as the final section demonstrates.

A *generalized* mixture $p(y|x)$ depends on the input value x as well as on y. The mixture $p(y|x)$ depends on x because in general both the mixture weights $\pi_1(x), \ldots, \pi_m(x)$ and the mixed densities $p_1(y|x), \ldots, p_m(y|x)$ depend on x:

$$p(y|x) = \sum_{j=1}^{m} \pi_j(x) p_j(y|x). \tag{2}$$

The dependence on x involves much greater complexity and expressive power than does the simpler case of the ordinary mixture $p(y)$ in (1).

Theorem 1 below shows that all additive fuzzy systems define a governing generalized mixture density $p(y|x)$. Theorem 6 and its corollaries show that a generalized mixture $p(y|x)$ still holds for a rule continuum where the additive fuzzy system has as many rules as there are real numbers. Some form of the *standard* additive fuzzy system in (62)–(63) is the most common fuzzy system in practice. It has a simplified generalized mixture $p(y|x)$ where its convex mixing weights $\pi_j(x)$ depend on the input x but the mixed then-part densities $p_{B_j}(y)$ do not depend on x. This simplification arises from the multiplicative structure of its rule firing in (3). It allows users to pre-compute the then-part set centroids and volumes. It also leads to the comparatively tractable supervised learning laws (130)–(137) for tuning the rule parameters. The mixed then-part densities can be Gaussian in most cases.

The most popular mixture model by far is the Gaussian mixture model. Then p_j is a scalar Gaussian $\mathcal{N}(\mu_j, \sigma_j^2)$ or a vector Gaussian $\mathcal{N}(\mu_j, K_j)$ with mean vector μ_j and covariance matrix K_j. The estimation task is to find the m mixture weights and the parameters of the mixed pdfs. The ubiquitous Expectation-Maximization (EM) algorithm usually estimates the mixing weights $\pi_1, \ldots, \pi_m$ and the Gaussian means $\mu_1, \ldots, \mu_m$ and variances (or covariances) $\sigma_1^2, \ldots, \sigma_m^2$.[28–33] The EM algorithm iteratively maximizes the log-likelihood function. The class memberships of the m decision classes correspond to the hidden or latent variables in the EM algorithm. Then carefully injected noise can always speed up convergence of the EM algorithm[34–37] as it climbs the nearest hill of likelihood. The EM updates of the Gaussian mixture have the especially simple form (112)–(115). We discuss below how to use two Gaussian mixture models as one way to estimate an additive fuzzy system's governing mixture $p(y|x)$.

The mixture sum (1) follows from the elementary theorem on total probability. Suppose that m binary hypothesis sets $H_1, \cdots, H_m$ partition a sample space Ω. Suppose further that the set $E \subset \Omega$ represents some observed evidence. Then the theorem on total probability states that the unconditional probability of the evidence $P(E)$ equals the convex combination of the prior probabilities $P(H_j)$ and the likelihoods $P(E|H_j)$: $P(E) = \sum_{j=1}^{m} P(H_j)P(E|H_j)$. This corresponds to the above mixture sum (1) if the evidence E is the input x and if p_j is the prior probability $P(H_j)$ of the j-th class or mixture element. So $p_j(x) = f(x|j)$ holds if the conditional density $f(x|j)$ is the likelihood that we would observe such an x if it came from the jth class or from the jth mixed density.

Total probability also implies at once a Bayes theorem for computing the converse or posterior probability $P(H_j|E)$ or $f(j|x)$. We exploit this fact below for the generalized mixture structure of an additive fuzzy system and the combination of multiple such systems. The Bayes theorem in Corollary 1 is but one such theorem that follows from the governing mixture structure. The mixture-based result in (86) states a more complex Bayes theorem when combining an arbitrary number of rule-based function approximators with an arbitrary number of exact representations.

We turn first to the definition of the fuzzy rules themselves and how inputs fire them.

2.2. Rule Firing as Convolution with a Delta Spike

An additive fuzzy system adds the fired then-parts of its m if-then rules $R_{A_1 \to B_1}, \ldots, R_{A_m \to B_m}$. So the analysis begins with the formal structure of a fuzzy rule $R_{A_j \to B_j}$ and how an input vector $x \in \mathbb{R}^n$ fires the rule. There are many ways to define a fuzzy rule. We use the usual Cartesian product for its simplicity and because it yields the multiplicative structure of the standard additive model below in (62)–(63). This firing result extends to fuzzy subsets $A \subset \mathbb{R}^n$ that act as inputs to the fuzzy system F. Then F maps the power set of all fuzzy input sets to the reals. We use a scalar-valued fuzzy system $F : \mathbb{R}^n \to \mathbb{R}$ for simplicity and with no loss of generality. All results extend directly to vector-valued fuzzy systems $F : \mathbb{R}^n \to \mathbb{R}^p$.

The jth fuzzy rule $R_{A_j \to B_j}$ states that the then-part fuzzy set $B_j \subset \mathbb{R}$ holds or occurs if the if-part fuzzy set $A \subset \mathbb{R}^n$ holds or occurs. The rule has the linguistic form "If $X = A_j$ then $Y = B_j$". We need not view this if-then relation as the simple conditional or material implication of propositional logic where the conditional is false if and only if the if-part or antecedent is true and the then-part or consequent is false. Such a pointwise approach can define a wide range of subset inclusions that depend on the type of fuzzy or multivalued implication operator involved.[38] We instead view the if-then rule as a fuzzy associative memory.[16]

Associative memories map similar input patterns to similar output patterns.[1] The simplest examples are matrices that map input vectors to output vectors using vector-matrix multiplication. The matrix M is a distributed storage medium for the overall input-ouput or if-then relation. The mapping itself is nonlinear as in a neural associative memory. A fuzzy version is a compositional rule of inference based on min-max operations.[1,11,12] Then the memory matrix or finite Cartesian product stores the association (A_j, B_j) where A_j and B_j are discrete fuzzy sets or fuzzy-unit ("fit") vectors. Suppose that the input finite fuzzy set A is similar to the stored if-part set A_j: $A \approx A_j$. Then composing or min-max "multiplying" the memory matrix M with A should give an output fuzzy set B that is similar to B_j: $B \approx B_j$. Then similar inputs map to similar outputs. Our approach extends these finite Cartesian products to operators that act on both input vectors $x \in \mathbb{R}^n$ and on input fuzzy subsets $A \subset \mathbb{R}^n$. We also dispense with the earlier min-max operations and just use ordinary multiplication and addition. The trick is to cast the input vector x_0 as the Dirac delta spike $\delta(x - x_0)$.

The rule $R_{A_j \to B_j}$ is a fuzzy subset of the product space $\mathbb{R}^n \times \mathbb{R}$ because its set function is a mapping $r_{A_j \to B_j} : \mathbb{R}^n \times \mathbb{R} \to [0, 1]$. But there are uncountably many such mappings. The rule mapping has to define a rule "patch" that combines the constituent fuzzy sets A_j and B_j. So we identify it with the fuzzy Cartesian product $A_j \times B_j \subset \mathbb{R}^n \times \mathbb{R}$: $R_{A_j \to B_j} = A_j \times B_j$. The Cartesian product $A_j \times B_j$ uses some pointwise conjunction operator "&" to define the product: $A_j \times B_j = \{(x, y) \in \mathbb{R}^n \times \mathbb{R} : x \in A_j \ \& \ y \in B_j\}$. Older schemes used pairwise minimum.[11,12,39] We use pairwise product to define the rule set function:

$$r_{A_j \to B_j}(x, y) = a_j(x) b_j(y) \tag{3}$$

for all $x \in \mathbb{R}^n$ and all $y \in \mathbb{R}$. Then the input-output pair (x, y) belongs both to the jth rule $R_{A_j \to B_j}$ and to the Cartesian patch $A_j \times B_j$ to degree $a_j(x)b_j(y)$ because $R_{A_j \to B_j} = A_j \times B_j$.

The if-part fuzzy set $A_j \subset \mathbb{R}^n$ has joint set function $a_j : \mathbb{R}^n \to [0, 1]$. The then-part fuzzy set $B_j \subset \mathbb{R}$ has set function $b_j : \mathbb{R} \to [0, 1]$. We allow the range of these then-part sets to extend above 1.[5,8,13] These generalized set functions $b_j : \mathbb{R} \to \mathbb{R}^+$ need only be integrable because of the normalization involved in forming the fuzzy system's mixture density and in the moments of the mixtures. So there is no loss of generality in working with then-part set functions that map to the unit interval.

The symmetry of (3) implies that the rules are bidirectional: $r_{B_j \to A_j}(x, y) = b_j(y)a_j(x)$. Bidirectionality allows the fuzzy system $F : \mathbb{R}^n \to \mathbb{R}$ to run backwards and give the reverse system $G : \mathbb{R} \to \mathbb{R}^n$ using the same rule base but with the rules reversed to $r_{B_j \to A_j}(x, y) = a_j(x)b_j(y)$.[1] Then the reverse output $G(y)$ is just the convex combination of the centroids of the m if-part sets A_j. The centroid bounds trap the bidirectional outputs $F(x)$ and $G(y)$ and keep them from diverging to infinity. We focus here only on unidirectional fuzzy systems.

The product operator (3) generalizes a binary implication operator in a rough sense because its truth value equals zero in the extreme case when $a_j(x) = 1$ and $b_j(y) = 0$.[40] The product operator shares its bidirectional symmetry with Mamdani's minimum implication operator $t(P \to Q) = \min(t(P), t(Q))$ where $t(P) \in [0, 1]$ gives the truth value of statement P.[41] This conjunction-like behavior can comport with informal uses of material implication because it implies that a conditional is false if the if-part is false. Most people view the conditional "If I am dead then I am alive" as false when the speaker is alive. But the statement is true for the usual truth-tabular and asymmetric binary definition of implication where $t(P \to Q) = 1$ if $t(P) = 0$ no matter the binary truth value of the then-part. The statement is false even under the truth-tabular reckoning when the speaker is dead because then the truth-value pair $t(P) = 1$ and $t(Q) = 0$ results.

The if-part set function $a_j : \mathbb{R}^n \to [0, 1]$ usually factors in practice. This occurs because users tend to separately define fuzzy subsets of independent variables such as temperature and humidity and luminosity. Then the if-part coordinates are not in the same units. That does not affect the set's or rule's mathematical structure. Some fuzzy rules are not factorable. This includes the ellipsoidal fuzzy rules in Figure 2. An n-dimensional ellipsoid E does not factor in general into any product $E_1 \times \cdots \times E_n$ of scalar surfaces E_k.[8] Ellipsoidal rules arise naturally from sample covariance matrices.[4]

We do not require that the joint if-part set functions $a_j : \mathbb{R}^n \to [0, 1]$ factor. But we advocate product factorization when one needs factorization because of the mathematical benefits it confers on the standard additive model below in (62)–(63). These benefits become ever more important as the number n of input variables increases. Product factorization gives

$$a_j(x) = \prod_{l=1}^{n} a_j^l(x^l) \tag{4}$$

when each factor fuzzy subset $A_j^l \subset \mathbb{R}$ has scalar set function $a_j^l : \mathbb{R} \to [0, 1]$ for input column vector $x^T = (x^1, \ldots, x^n)$. Earlier fuzzy systems sometimes formed the joint set function a_j by taking componentwise minima $a_j(x) = \min(a_j^1(x^1), \ldots, a_j^n(x^n))$[18,42–44] or by taking some other componentwise triangular-norm operation.[45,46] Triangular norms generalize minimum while their De Morgan dual triangular co-norms generalize maximum. But the minimum function ignores the fit information in the n fit values $a_j^l(x^l)$ except for the smallest fit value when those fit values differ. The product function preserves this information because it only scales the n fit values. The fit values maintain their relative relationships and each contributes to the rule output. Nor need the n components in $a_j(x)$ combine conjunctively. They can combine disjunctively or through any other linguistic or truth function of the n terms $a_j^1(x^1), \ldots, a_j^n(x^n)$. The results below just use the aggregate fit value $a_j(x)$. So the input space can technically have infinite dimension.

Fuzzy sets $A \subset \mathbb{R}^n$ can also act as inputs to the fuzzy system $F : \mathbb{R}^n \to \mathbb{R}$. Then the fuzzy set A fires each of the m rules $R_{A_j \to B_j}$. This general rule firing will show how a vector input $x \in \mathbb{R}^n$ can fire a rule as the special case when the set function reduces to a delta spike. The idea is that A fires the rule $R_{A_j \to B_j}$ to the degree that A matches the jth if-part set $A_j \subset \mathbb{R}^n$. There are many ways to define this matching operation to give the match degree $a_j(A)$. A natural way defines $a_j(A)$ as simply the inner-product correlation[5] between A and A_j:

$$a_j(A) = \int_{R^n} a(x) a_j(x) dx \tag{5}$$

when it exists for fuzzy set A with set function $a : \mathbb{R}^n \to \mathbb{R}$. The correlation $a_j(A)$ is nonnegative and may well extend beyond the unit interval. This poses no problem for additive systems because of normalization. The correlation need only be nonnegative and integrable.

We show next how the vector input x_0 fires the jth rule $R_{A_j \to B_j}$.

Descriptions of fuzzy systems tend to leave this rule firing to the intuition that the input x_0 somehow "picks off" the fit value $a_j(x_0)$ from the entire set-function curve $a_j : \mathbb{R}^n \to [0, 1]$. But how exactly does such picking take place for a continuous set function? Such rule firing would not be an issue if fuzzy systems fired their fuzzy rules based on two-place arguments (x, y) because such arguments would simply evaluate the rule's set function $r_{A_j \to B_j} : \mathbb{R}^n \times \mathbb{R} \to [0, 1]$. The associative-memory structure dictates instead only a partial firing of $R_{A_j \to B_j}$ because x_0 activates only the stored if-part set A_j.

The input x_0 does not directly fire the then-part set B_j. The input x_0 only indirectly fires B_j by firing A_j and thereby invoking the fuzzy rule $R_{A_j \to B_j}$. We could denote this indirect firing of B_j with the prime notation B_j'. More accurate notation is $B_j(x_0)$ with a corresponding conditional set function $b(y|x_0)$ for all y. The conditional mapping $b(.|.) : \mathbb{R} \times \mathbb{R}^n \to [0, 1]$ also makes clear that the conditioning variable x indexes a potentially uncountable family of then-part sets $B_j(x)$.

Suppose first that the if-part set A_j is a finite fuzzy subset of a finite input space $X = \{x_1, \ldots, x_n\}$. Then A_j defines a fit vector or a column vector in the fuzzy hypercube[1]: $A_j^T = (a_j^1, \ldots, a_j^n) \in [0, 1]^n$. Identify the input

x_0 with its singleton set $\{x_0\}$. So we can write it as the unit-binary column vector $\{x_0\}^T = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$. The bit vector $\{x_0\}$ has a 1 in the kth slot that corresponds to x_0 (or x_k). It has a 0 in all the other $n - 1$ slots. Then the finite inner product picks off the desired fit value $a_j(x_0)$ or a_j^k: $\{x_0\}^T A_j = a_j^k$. The same result follows if we use instead the older min-max composition that takes a global maximum of pairwise minima: $\{x_0\}^T \circ A_j = \max(\min(0, a_j^1), \ldots, \min(0, a_j^{k-1}), \min(1, a_j^k), \min(0, a_j^{k+1}), \ldots, \min(0, a_j^n)) = a_j^k$. Extending this firing operation to the continuous case requires the more powerful machinery of Dirac delta functions or generalized functions.

The scalar or 1-D Dirac delta *generalized* function δ acts as if $\delta(x) = 0$ when $x \neq 0$ and $\delta(x) = \infty$ when $x = 0$. Yet it integrates to unity over the whole space $\mathbb{R}$: $\int_{\mathbb{R}} \delta(x)dx = 1$. But technically it has no Riemann integral and its Lebesgue integral equals zero. The delta function naturally models an infinitesimally narrow spike and thus an input point such as x_0. It also obeys the important translation or "sifting" property when it convolves with a continuous function f: $\int_{\mathbb{R}} \delta(x - x_0) f(x)dx = f(x_0)$. It thus naturally "picks off" the value $f(x_0)$ from the entire curve. The delta function further generalizes to n dimensions by factoring into a product of n scalar delta functions: $\delta(x) = \prod_{l=1}^{n} \delta(x^l)$ for vector input $x^T = (x^1, \ldots, x^n)$.

We can view the vector input x_0 as a special case of a continuous fuzzy set A_j by identifying x_0 with the properly translated vector delta spike: $x_0 = \delta(x - x_0)$. Then we arrive at last at the "firing" operation that shows how the input $x_0 \in \mathbb{R}^n$ fires the if-part fuzzy set A_j as the Dirac-delta special case of the inner product in (5):

$$a_j(x_0) = \int_{\mathbb{R}^n} \delta(x - x_0)\, a_j(x)\, dx. \tag{6}$$

So firing an if-part set convolves its set function with a Dirac delta function.

Rule firing follows from the same delta-function convolution. Firing the if-part set A_j leads to the indirect firing of the then-part set B_j. This gives the fired then-part set $B_j(x_0)$ with conditional set function $b_j(.|x_0) : \mathbb{R} \to [0, 1]$. Combining the product rule structure (3) with (6) shows how the vector input x_0 fires the jth fuzzy rule $R_{A_j \to B_j}$ for all $y \in \mathbb{R}$:

$$b_j(y|x_0) = \int_{\mathbb{R}^n} \delta(x - x_0)\, r_{A_j \to B_j}(x, y)\, dx \tag{7}$$

$$= \int_{\mathbb{R}^n} \delta(x - x_0)\, a_j(x)\, b_j(y)\, dx \tag{8}$$

$$= b_j(y) \int_{\mathbb{R}^n} \delta(x - x_0)\, a_j(x)\, dx \tag{9}$$

$$= b_j(y)\, a_j(x_0). \tag{10}$$

So $b_j(y|x_0) = a_j(x_0)\, b_j(y)$. This equality is identical to the result of inserting the argument pair (x_0, y) into the rule set function in (3): $r_{A_j \to B_j}(x_0, y) = a_j(x_0) b_j(y)$.

This gives $r_{A_j \to B_j}(x_0, y) = b_j(y|x_0)$ for all y. We can write this more compactly at the set level as $r_{A_j \to B_j}(x_0) = a_j(x_0)\ B_j$. The product-scaling rule firing $a_j(x_0)\ B_j$ is the "standard" part of the standard additive model below in (62)–(63).

The same derivation holds for a fuzzy-set input $A \subset \mathbb{R}^n$. Replace the delta function $\delta(x - x_0)$ with the input set function $a(x)$. Then the above derivation gives $b(y|A) = a_j(A)b_j(y)$ using (5).

A similar argument shows that using pairwise minimum in (3) or $r_{A_j \to B_j}(x_0, y) = \min(a_j(x_0), b_j(y))$ leads to the fired then-part rule as the min-clip $a_j(x_0) \wedge B_j$. Many earlier fuzzy systems used the min-clip to fire rules and did so on an ad hoc basis. But the min-clip $a_j(x_0) \wedge B_j$ ignores all the information in the then-part set B_j above the level of $a_j(x_0)$. That missing information undermines the system's performance and needlessly complicates learning laws and other operations. The product-scaled firing $a_j(x_0)\ B_j$ gives full weight to all information above the $a_j(x_0)$ level because the fit value $a_j(x_0)$ simply scales or shrinks the entire then-part curve B_j over the same domain. Product-scaled firing also simplifies learning laws as well as function approximation and representation. So we assume the standard product-scaled firing $a_j(x_0)\ B_j$ throughout. The next task is to combine these fuzzy rules into a functioning fuzzy system.

2.3. Generalized Mixtures Govern Additive Fuzzy Systems

A fuzzy system $F : \mathbb{R}^n \to \mathbb{R}$ must somehow combine its rule base of m fuzzy rules $R_{A_1 \to B_1}, \ldots, R_{A_m \to B_m}$ when it processes each fuzzy input $x \in \mathbb{R}^n$ to produce the output $F(x)$.

Early fuzzy systems took the union or componentwise maximum of the m rules.[18,42,47] This reflected an earlier fuzzy ethos of using maximum for disjunction operators and minimum for conjunction operators. It also reflected the so-called extension principle[11,12,42] for composing finite fuzzy vectors with fuzzy relations. This approach derives from Zadeh's original 1965 proposal[10] for extending point mappings to fuzzy-set mappings. But even these systems took the union of only the fired then-part sets $B_j(x)$ and not the rules themselves. So they formed the set function $b(y|x)$ of the combined rule firings $B(x)$ as $b(y|x) = \max(b_1(y|x), \ldots, b_m(y|x))$. This satisfies the min-max philosophical constraint of keeping the set function $b(y|x)$ in the unit interval. But the maximum operation ignores the overlap in all the fired then-part sets $B_j(x)$. It is just this overlap information that changes the relative weight of rule firings and thus that allows the system to distinguish slightly different inputs.

Neural networks offer another way to combine rules. Their default combination technique is to superimpose associative patterns such as (A_j, B_j) onto the same memory medium. This means adding correlation or outer-product matrices for associative memories.[48–52] The additive structure of many learning algorithms either uses or approximates this correlation (Hebbian) structure. But adding memory matrices limits memory capacity and recall accuracy. Each superimposed pattern or association becomes a form of noise or crosstalk for all the others.

An additive fuzzy system[1] $F : \mathbb{R}^n \to \mathbb{R}$ *adds* the m fired then-part sets $B_j(x)$ from the fuzzy rule base $R_{A_1 \to B_1}, \ldots, R_{A_m \to B_m}$ for a given vector $x \in \mathbb{R}^n$. The system

stores the m rules by storing only the rule parameters in its sum structure. This allows an additive fuzzy system to work with a rule continuum because the system uses only a random rule sample or realization from the virtual rule continuum. Taking the sum of fired then-part sets $B_j(x)$ gives the total rule firing $B(x)$ for input x:

$$B(x) = \sum_{j=1}^{m} B_j(x). \tag{11}$$

The combined structure $B(x)$ is a generalized fuzzy set because its set function $b(y|x)$ exceeds unity in general.

We allow m nonnegative rule weights $w_1, \ldots, w_m$ to scale the m rules and thereby alter the combined rule firings $B(x)$. The m weights w_j define the diagonal entries of an $m \times m$ rule-weight matrix W. Then off-diagonal entries w_{kj} define *cross* rules of the form $R_{A_k \to B_j}$. We focus for simplicity just on the usual diagonal rules. So we assume that $w_{kj} = 0$ when k differs from j. Learning can also tune the rule weights and they can depend on the input x. The rule weights slightly generalize (11) to

$$B(x) = \sum_{j=1}^{m} w_j B_j(x) \tag{12}$$

with corresponding set function $b(y|x) = \sum_{j=1}^{m} w_j b_j(y|x)$. A *generalized* additive system results in (12) if we replace the fired then-part set $B_j(x)$ with an arbitrary nonlinear map $B_j : \mathbb{R}^n \to \mathbb{R}^p$.[5] A Tanaka fuzzy dynamical system results if B_j is a stable square matrix. A TSK or Takagi-Sugeno-Kang additive system results if B_j is a linear function $f(x_1, \ldots, x_n)$ of the input x since it is a feedback system.[5]

Normalizing the set function $b(y|x)$ of (12) by its own finite integral yields the fuzzy system's governing mixture density $p(y|x)$:

$$p(y|x) = \frac{b(y|x)}{\int_R b(y|x)\, dy} \tag{13}$$

where we assume that the generalized set function $b(y|x)$ is nonnegative (and never identically zero) and integrable. The conditional $p(y|x)$ is a proper pdf because it is nonnegative and integrates to unity. The mixture $p(y|x)$ results directly from the additive combination of the fired then-part rules (12). Normalization of a maximum or supremum combiner will still produce a probability density but not a mixture density.

We can define a Bayesian or maximum-a-posteriori fuzzy system F_{MAP} as that fuzzy system that finds the mode of the posterior pdf $p(y|x)$:

$$F_{MAP}(x) = \text{argmax}_y \; p(y|x) \tag{14}$$

where the search is over all output values y. Markov Chain Monte Carlo techniques can estimate such MAP values in many cases.[53,54] We will work instead with an additive fuzzy system based on the first moment of the posterior $p(y|x)$. But there may well be cases where an analyst or engineer would want to work with some form of the purely Bayesian model (14).

Theorem 1 states the general mixture result for an additive fuzzy system. It uses the fact that the normalized fired then-part set function $b_j(y|x)$ is a proper pdf $p_{B_j}(y|x)$:

$$p_{B_j}(y|x) = \frac{b_j(y|x)}{\int_R b_j(y|x)\, dy} = \frac{b_j(y|x)}{V_j(x)} \tag{15}$$

where $V_j(x) = \int_R b_j(y|x)\, dy > 0$ is the area or volume under the then-part fuzzy-set curve $B_j(x)$. So we assume that all such curves have positive area and are integrable.

THEOREM 1. *Mixture Theorem for Additive Fuzzy Systems. Additive rule combination* (12) *defines a finite mixture probability density* $p(y|x)$*:*

$$p(y|x) = \sum_{j=1}^{m} p_j(x)\ p_{B_j}(y|x) \tag{16}$$

where the generalized mixture weights $p_j(x)$ *have the ratio form*

$$p_j(x) = \frac{w_j\ V_j(x)}{\sum_{k=1}^{m} w_k\ V_k(x)}. \tag{17}$$

Proof. Insert the additive property (12) of combined fired then-part sets $B_j(x)$ into (13) and expand:

$$p(y|x) = \frac{b(y|x)}{\int_{\mathbb{R}} b(y|x)\, dy} \tag{18}$$

$$= \frac{\sum_{j=1}^{m} w_j\ b_j(y|x)}{\int_{\mathbb{R}} \sum_{k=1}^{m} w_k\ b_k(y|x)\, dy} \tag{19}$$

$$= \frac{\sum_{j=1}^{m} w_j\ b_j(y|x)}{\sum_{k=1}^{m} w_k \int_{\mathbb{R}} b_k(y|x)\, dy} \tag{20}$$

$$= \frac{\sum_{j=1}^{m} w_j\ V_j(x) \frac{b_j(y|x)}{V_j(x)}}{\sum_{k=1}^{m} w_k\ V_k(x)} \tag{21}$$

$$= \frac{\sum_{j=1}^{m} w_j\ V_j(x) p_{B_j(y|x)}}{\sum_{k=1}^{m} w_k\ V_k(x)} \tag{22}$$

$$= \sum_{j=1}^{m} \left(\frac{w_j \; V_j(x)}{\sum_{k=1}^{m} w_k \; V_k(x)} \right) p_{B_j}(y|x) \tag{23}$$

$$= \sum_{j=1}^{m} p_j(x) \; p_{B_j}(y|x) \tag{24}$$

for fired then-part pdf $p_{B_j}(y|x)$ of the jth rule's then-part set B_j. □

The mixture-weight pdfs $p_j(x)$ are generalized because they depend on x. Each input x produces its own set of convex mixture weights. This greatly increases the mixture's nonlinear expressive power over the simple fixed-weight mixture in (1). It also greatly increases the difficulty and computational complexity of adapting or tuning the mixture.

The mixture result (16) further holds more generally for input fuzzy sets A instead of simply input point vectors: $p(y|A) = \sum_{j=1}^{m} p_j(A) \; p_{B_j}(y|A)$ for correlation firing (5). This set generalization holds for all subsequent results and for the same reason even though we will focus only on vector inputs.

The mixture structure implies the existence of a discrete latent or "hidden" random variable Z. The variable Z takes as values the class memberships j of the m subpopulations. The subpopulations here are just the m rules. So $P(Z = j)$ is the *prior* probability that the jth rule fires. It is just this variable Z that the EM algorithm computes conditioned on the data and current parameter estimates.

The pdf $P(y|Z = j, x)$ is a likelihood. We can also write it as $p(y|j, x)$ or in input-indexed form as $P_x(Y = y|Z = j)$ or just $P_x(y|j)$. It is the probability that the fuzzy system F will emit the output value y or $F(x)$ given that the jth rule fired and given the vector input x.

The rule variable Z shows that the basic fuzzy-system mixture density (16) is just the law of total probability. This holds because the rule-firing events $\{Z = 1\}, \ldots, \{Z = m\}$ partition the underlying sample space because they are mutually disjoint and exhaustive for each x. So $\{Y = y\} = \{Y = y\} \cap \{\cup_{j=1}^{m} \{Z = j\}\} = \cup_{j=1}^{m} \{\{Y = y\} \cap \{Z = j\}\}$ for each x. Then the partition gives $P_x(Y = y) = \sum_{j=1}^{m} P_x(\{Y = y\}, \{Z = j\}) = \sum_{j=1}^{m} P_x(Z = j) \; P_x(Y = y|Z = j)$ from the ratio definition of conditional probability $P(B|A) = \frac{P(A \cap B)}{P(A)}$ for $P(A) > 0$. This last sum restates total probability. It is just the mixture result (16) in slightly different notation.

A Bayes theorem follows at once from (16) as it does from all mixtures. The theorem computes the converse probability $p(j|y, x) = P_x(Z = j|Y = y)$ that the jth rule fired given the observed output value y and the input x. This converse or posterior density $p(j|y, x)$ specifies which rule fired to which degree given the input vector x that produced the observed output y. This converse information gives direct insight into how the rule-based system reached its output decision. So it makes the additive fuzzy system an interpretable model.[17] This posterior $p(j|y, x)$ also plays a key part of the EM algorithm's update equations for a Gaussian mixture model.[28–30] It gives a practical way to measure the relative importance of each rule in the rule base for any given input x.

COROLLARY 1. *Bayes Theorem for the mixture density* (16) *of an additive fuzzy system:*

$$p(j|y,x) = \frac{p_j(x)\ p_{B_j}(y|x)}{\sum_{k=1}^{m} p_k(x)\ p_{B_k}(y|x)} \tag{25}$$

where $j = 1, \ldots, m$ *indexes the* m *rules* $R_{A_1 \to B_1}, \ldots, R_{A_m \to B_m}$ *of the additive fuzzy system.*

Proof. This corollary of (16) follows from the definition of conditional probability and then using (16) to eliminate the denominator probability:

$$p(j|y,x) = P_x(Z = j|Y = y) \tag{26}$$

$$= \frac{P_x(Z = j)\ P_x(Y = y|Z = j)}{P_x(Y = y)} \tag{27}$$

$$= \frac{p_j(x)\ p_{B_j}(y|x)}{p(y|x)} \tag{28}$$

$$= \frac{p_j(x)\ p_{B_j}(y|x)}{\sum_{k=1}^{m} p_k(x)\ p_{B_k}(y|x)}. \tag{29}$$

□

The mixture result (16) does not depend on the product rule structure in (3). Both it and the Bayes result (25) hold for any additive fuzzy system.

We show now how the standard or product structure (3) of a SAM or *standard additive model* simplifies both results by eliminating the dependence on the input x in the likelihood pdfs $p_{B_j}(y|x)$:

$$p_{B_j}(y|x) = p_{B_j}(y) \tag{30}$$

so long as $a_j(x) > 0$. The elimination of the conditioning x from $p_{B_j}(y|x)$ follows from (10) and (15):

$$p_{B_j}(y|x) = \frac{b_j(y|x)}{\int_{\mathbb{R}} b_j(y|x)\ dy} = \frac{a_j(x)\ b_j(y)}{a_j(x) \int_{\mathbb{R}}\ b_j(y)\ dy} \tag{31}$$

$$= \frac{b_j(y)}{\int_{\mathbb{R}} b_j(y)\ dy} = p_{B_j}(y). \tag{32}$$

Then (16) gives the SAM mixture density $p_{SAM}(y|x)$ as

$$p_{SAM}(y|x) = \sum_{j=1}^{m} p_j(x)\ p_{B_j}(y). \tag{33}$$

The SAM result (30) likewise simplifies Corollary 1 to the Bayesian ratio

$$p_{SAM}(j|y,x) = \frac{p_j(x)\ p_{B_j}(y)}{\sum_{k=1}^{m} p_k(x)\ p_{B_k}(y)}. \tag{34}$$

Note that a given generalized SAM mixture density $p_{SAM}(y|x)$ defines an m-rule SAM fuzzy system. This requires only that we equate the if-part set function $a_j(x)$ of the set A_j to the mixture weight $p_j(x)$. We can likewise equate the then-part set function $b_j(y)$ of the set B_j to the then-part likelihood density $p_{B_j}(y)$. We can also set the rule weights w_j to unity as a default.

The generalized mixtures (16) and (33) allow arbitrary combination of additive fuzzy systems. This holds because *a mixture of mixtures is itself a mixture*. This self-similar property of mixtures holds for infinite mixtures as well as for finite mixtures. We explore it elsewhere for hierarchical fuzzy systems. We show now how it applies to combining q-many additive fuzzy systems.

A simple way to combine q additive fuzzy systems $F^1, \ldots, F^q$ is just to mix them with q convex coefficients $v^1, \ldots, v^q$. The fuzzy approximation theorem[2] allows each fuzzy system F^k to uniformly approximate an expert or sampled nonlinear process or any other knowledge source.[55] The result is a new higher-level mixture $p(y|x)$:

$$p(y|x) = \sum_{k=1}^{q} v^k\ p_{F^k}(y|x) \tag{35}$$

where $p_{F_k}(y|x)$ is the mixture density of the kth additive fuzzy system F^k. This mixing process can continue indefinitely. It will always produce an overall or master mixture density. The result still combines multiple knowledge sources into a common fuzzy rule base.

A more powerful way to combine fuzzy systems is to combine their rules or throughputs rather than their outputs.[5] This applies to any form of fuzzy system. It has special force for additive systems because of their convex-sum structure. The technique replaces the rule-firing sum in (12) with a weighted sum of such rule firings:

$$B(x) = \sum_{k=1}^{q} v^k B^k(x). \tag{36}$$

A higher-order sum of sums can replace $B(x)$ and so on indefinitely for any method of computing fired rules $B^k(x)$. Suppose in particular that the method is the additive one in (12). Then the same expansion of the generalized set function $b(y|x)$ from the proof of Theorem 1 yields

$$p(y|x) = \sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x)\ p_{B_j}^k(y|x) \tag{37}$$

where now the generalized mixture weight $p_j^k(x)$ has the ratio form

$$p_j^k(x) = \frac{v^k a_j^k(x)\ w_j^k\ V_j^k(x)}{\sum_{k=1}^{q} \sum_{j=1}^{m_k} v^k a_j^k(x)\ w_j^k\ V_j^k(x)}. \tag{38}$$

The SAM mixture $p_{SAM}(y|x)$ just replaces $p_{B_j}^k(y|x)$ with $p_{B_j}^k(y)$.

The Bayes result in Corollary 1 extends to

$$p(k, j|y, x) = \frac{p_j^k(x)\ p_{B_j}^k(y|x)}{\sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x)\ p_{B_j}^k(y|x)} \tag{39}$$

where the new discrete knowledge-source random variable W takes as values the labels $k = 1, \ldots, q$ of the q fuzzy systems or knowledge sources. The posterior pdf $p(k, j|y, x)$ states the probability or relative importance of the jth rule of the kth expert or system given input x and given the observed output y.

A second Bayes Theorem gives the relative importance $p(k|y, x)$ of the kth fuzzy system or expert F_k given the observed value y for the input x. The result follows by marginalizing out the m_k rules of the kth expert or fuzzy system:

$$p(k|y, x) = \sum_{j=1}^{m_k} p(k, j|y, x) \tag{40}$$

$$= \frac{\sum_{j=1}^{m_k} p_j^k(x)\ p_{B_j}^k(y|x)}{\sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x)\ p_{B_j}^k(y|x)}. \tag{41}$$

We turn next to the moment structure of the additive-fuzzy-system mixture density $p(y|x)$ and the related system-level theorems on SAM fuzzy systems.

3. MIXTURE-BASED PROPERTIES OF ADDITIVE FUZZY SYSTEMS

The generalized mixture $p(y|x)$ leads to several key properties of additive fuzzy systems in general and SAM systems in particular. This includes recasting a fuzzy system $F : \mathbb{R}^n \to \mathbb{R}$ as a conditional expectation with its own conditional-variance uncertainty measure. It also includes adaptation rules and both function approximation and exact function representation.

3.1. Moments of Generalized Mixtures

An easy theorem is that *every* fuzzy system F is a conditional expectation: $F(x) = E[Y|X = x]$. This holds just so long as the fuzzy system "defuzzifies" or computes the output $F(x)$ as the centroid of the system's combined rule firings $B(x)$: $F(x) = \text{Centroid}(B(x))$. Any method of combining the rule firings suffices if the corresponding joint set function $b(y|x)$ is nonnegative and integrable. The

proof below in (42) – (46) follows the same lines of that of the next theorem that shows how the integrable set function $b(y|x)$ and thus the mixture $p(y|x)$ leads to all higher-order moments so long as the appropriate integrals exist.

We first establish some moment notation. We will focus on positive-integer central moments about the mean. There are in general uncountably many fractional-order moments. All results also hold for non-central moments. Denote the fuzzy system's kth conditional moment as $\mu^k(x)$. The first non-central moment $\mu(x)$ is just the conditional-expectation realization $E[Y|X = x]$. The second central moment $\mu^{(2)}(x)$ is the conditional variance $V[Y|X = x]$ and so on up to the kth moment $\mu^{(k)}(x) = E[(Y - E[Y|X = x])^k|X = x]$. All the moments result from integrating or averaging against the additive fuzzy system's mixture pdf $p(y|x)$. The mixture structure of $p(y|x)$ simplifies these moments in the additive case.

The first *non*-central moment $\mu(x)$ is the output of the additive fuzzy system if the system uses *centroid* defuzzification of the fired then-part sets $B(x)$: $F(x) = \text{Centroid}(B(x))$. This follows by expanding the definitions:

$$\mu(x) = E[Y|X = x] \tag{42}$$

$$= \int_{\mathbb{R}} y \; p(y|x) \; dy \tag{43}$$

$$= \frac{\int_{\mathbb{R}} y \; b(y|x) \; dy}{\int_{\mathbb{R}} b(y|x) \; dy} \tag{44}$$

$$= \text{Centroid}(B(x)) \tag{45}$$

$$= F(x). \tag{46}$$

So *every centroidal fuzzy system is a conditional expectation*. This result holds even if the system is not additive.

The second *central* moment $\mu^{(2)}(x)$ about the mean is just the system conditional variance $V[Y|X = x]$:

$$\mu^{(2)}(x) = V[Y|X = x] \tag{47}$$

$$= \int_{\mathbb{R}} (y - F(x))^2 \; p(y|x) \; dy. \tag{48}$$

Additive fuzzy systems also use the local moments involved in the then-part of the jth rule $R_{A_j \to B_j}$. These rule-level central and non-central moments arise from the rule likelihood pdf $p_{B_j}(y|x)$ in (15). The first *local* non-central moment is the centroid $c_j(x)$ of fired then-part set $B_j(x)$:

$$c_j(x) = E_{B_j(x)}[Y|X = x] = \int_{\mathbb{R}} y \; p_{B_j}(y|x) \; dy. \tag{49}$$

Then the SAM condition in (30) also leads to dropping the conditioning term x. So the user can pre-compute these unconditional then-part centroids c_j along with the related then-part volumes or areas V_j. The second rule-level moment is the

conditional variance $\sigma^2_{B_j(x)}(x)$ associated with $B_j(x)$:

$$\sigma^2_{B_j(x)}(x) = E_{B_j(x)}[(Y - E_{B_j(x)}[Y|X = x])^2|X = x] \tag{50}$$

$$= \int_{\mathbb{R}} (y - c_j(x))^2 p_{B_j}(y|x)\, dy. \tag{51}$$

The SAM condition (30) also drops the conditioning term $X = x$ here and allows the user to pre-compute the inherent unconditional uncertainty $\sigma^2_{B_j}$ in the then-part of the rule. These pre-computations can substantially speed implementation when computing first-order $F(x)$ and second-order $V[Y|X = x]$ uncertainty terms.

We can now state and prove the moment theorem for additive systems. It states that all higher central moments are convex sums. This holds because they inherit the mixture's convex structure.

THEOREM 2. *Moment Theorem for additive fuzzy systems:*

$$\mu^{(k)}(x) = \sum_{j=1}^{m} p_j(x) \sum_{l=0}^{k} \binom{k}{l} E_{B_j(x)}[(Y - c_j(x))^l]\, [c_j(x) - F(x)]^{k-l} \tag{52}$$

for each positive integer k and combination coefficient $\binom{k}{l} = \frac{k!}{l!(k-l)!}$.

Proof. Use the additive mixture structure (16) for $p(y|x)$ and invoke the binomial theorem $(p + q)^k = \sum_{l=0}^{k} \binom{k}{l} p^l\, q^{k-l}$ in the expansion of $\mu^k(x)$:

$$\mu^{(k)}(x) = E[(Y - E[Y|X = x])^k|X = x] \tag{53}$$

$$= \int_{\mathbb{R}} (y - F(x))^k\, p(y|x)\, dy \tag{54}$$

$$= \int_{\mathbb{R}} (y - F(x))^k \sum_{j=1}^{m} p_j(x)\, p_{B_j}(y|x)\, dy \tag{55}$$

$$= \sum_{j=1}^{m} p_j(x) \int_{\mathbb{R}} (y - F(x))^k\, p_{B_j}(y|x)\, dy \tag{56}$$

$$= \sum_{j=1}^{m} p_j(x) \int_{\mathbb{R}} [(y - c_j(x)) + (c_j(x) - F(x))]^k\, p_{B_j}(y|x)\, dy \tag{57}$$

$$= \sum_{j=1}^{m} p_j(x) \int_{\mathbb{R}} \sum_{l=0}^{k} \binom{k}{l} [y - c_j(x)]^l [c_j(x) - F(x)]^{k-l}\, p_{B_j}(y|x)\, dy \tag{58}$$

$$= \sum_{j=1}^{m} p_j(x) \sum_{l=0}^{k} \binom{k}{l} \left\{ \int_{\mathbb{R}} [y - c_j(x)]^l \, p_{B_j}(y|x) \, dy \right\} [c_j(x) - F(x)]^{k-l} \quad (59)$$

$$= \sum_{j=1}^{m} p_j(x) \sum_{l=0}^{k} \binom{k}{l} E_{B_j(x)}[(Y - c_j(x))^l] \, [c_j(x) - F(x)]^{k-l}. \quad (60)$$

□

The moment theorem has several important corollaries. The first is that any centroidal additive fuzzy system F is just a convex combination of then-part set centroids:

$$F(x) = \sum_{j=1}^{m} p_j(x) \, c_j(x) \quad (61)$$

for convex mixing weights $p_j(x)$ in (17) in Theorem 1 for the non-central case where $\mu^1(x) = \int_{\mathbb{R}} y \; p(y|x) \, dy$. The SAM product-rule condition (3) and (49) eliminate the conditioning variable x on the then-part centroid c_j. This result is what we often call the SAM Theorem[5]:

$$F(x) = \sum_{j=1}^{m} p_j(x) \, c_j \quad (62)$$

where now the convex weights $p_j(x)$ have the more familiar form

$$p_j(x) = \frac{w_j \; a_j(x) \; V_j}{\sum_{k=1}^{m} w_k \; a_k(x) \; V_k}. \quad (63)$$

The result also follows directly by taking the centroid of the combined then-part rule firings $b(y|x)$ as in (42)–(46).

The SAM Theorem (62) states that the output $F(x)$ is just a convex combination of the then-part centroids $c_1, \ldots, c_m$. This gives the important bound $c_{\min} \leq F(x) \leq c_{\max}$ for scalar systems $F : \mathbb{R}^n \to \mathbb{R}$. A similar bound holds for vector-valued systems. Then the SAM output $F(x)$ lies inside a centroid-based hyperbox.[5] This bound greatly simplifies the proof that centroidal additive fuzzy systems can uniformly approximate any continuous function f defined on a compact set $K \subset \mathbb{R}^n$.[2,5]

The SAM Theorem still holds if the if-part sets A_j and then-part sets B_j are rectangles. Then there is no fuzziness in the system. It does remain a rule-based system and still enjoys the crucial property of uniform function approximation. We can further dispense with viewing functions such as $a_j : \mathbb{R}^n \to [0, 1]$ as defining fuzzy sets at all. We can view them instead as indexed families of conditional probabilities. Suppose the set A_j stands for COOL air. Suppose that $X = x$ holds where X is a temperature variable. Then the fuzzy view is that the air is cool to degree $a_j(x)$. But we can also view $a_j(x)$ as the probability that the air is cool *given* that it is x: $P(\text{COOL}|X = x) = a_j(x)$. Then coolness is a binary concept that admits

probabilities of occurrence. All air is either cool or it is not for a given temperature value x. Context and further constraints can detail the degree to which such non-cool air is warm or hot at x. We may well find the fuzzy view of cool air simpler and more intuitive than the probabilistic view. That is a pragmatic judgment and not a mathematical one.

The SAM system (62) also reduces to the center-of-gravity fuzzy systems found in most applications of fuzzy control. The center-of-gravity models assume that all m then-part sets B_j are the same. They often are congruent triangles or simply spikes centered at c_j. So the volumes V_j are all equal and drop out of (63). The same holds impliedly for the rule weights w_j. The implicit assumption is that then-part uncertainty does not matter. That is an odd assumption in a fuzzy framework. But it is accurate in terms of computing just the first-order output $F(x)$. The SAM Theorem (62) shows that $F(x)$ depends only on the centroid c_j and volume V_j of the then-part set B_j. There are infinitely many B_j set shapes compatible with a given centroid c_j and volume V_j. The next result shows that then-part shapes matter for higher-order uncertainty.

Another corollary of the Moment Theorem is a simple closed form for the conditional variance $V[Y|X = x]$ that describes the inherent uncertainty in any system output or answer $F(x)$:

$$V[Y|X = x] = \sum_{j=1}^{m} p_j(x)\, \sigma^2_{B_j(x)} + \sum_{j=1}^{m} p_j(x)\, [c_j(x) - F(x)]^2. \tag{64}$$

This follows from (52) because the combinatorial sum reduces to the three terms $\binom{2}{0} + \binom{2}{1} + \binom{2}{2}$ and because the expectation that corresponds to the middle term equals zero upon using (49) in its expansion. The data-dependent variance $\sigma^2_{B_j(x)}$ is just the expectation with respect to the rule likelihood $p_{B_j}(y|x)$: $\sigma^2_{B_j(x)} = E_{B_j(x)}[(Y - E_{B_j(x)}[Y|X = x])^2|X = x] = \int_{\mathbb{R}} (y - c_j(x))^2 p_{B_j}(y|x)dy$. Then-part set rectangles still result in rule (uniform) uncertainty even though such sets are not fuzzy. The SAM version again drops the parameter dependency on the input variable x:

$$V[Y|X = x] = \sum_{j=1}^{m} p_j(x)\, \sigma^2_{B_j} + \sum_{j=1}^{m} p_j(x)\, [c_j - F(x)]^2 \tag{65}$$

with mixing weights $p_j(x)$ in (63).

The shape of B_j controls the corresponding then-part variance σ_j^2 in (65). So the shape of the then-part sets affects the second-order uncertainty. This still holds in the popular center-of-gravity case when all then-part sets B_j are the same and hence give rise to the same nonzero variance σ^2:

$$V[Y|X = x] = \sigma^2 + \sum_{j=1}^{m} p_j(x)\, [c_j - F(x)]^2. \tag{66}$$

The variance $\sigma^2 > 0$ gives the unresolvable minimum uncertainty in the system. This also shows that two fuzzy systems F^1 and F^2 can satisfy $F^1(x) = F^2(x)$ for all x and yet differ in their second-order uncertainty. One system need have only wider then-part sets than the other.

The two terms on the right side of (65) measure the second-order uncertainty in each output $F(x)$. This acts as a natural confidence measure for the output $F(x)$ without invoking the complexity of Type-2 fuzzy sets or other ad hoc uncertainty structures. The first term gives the convex-weighted uncertainty or variance of the m then-part sets. The mixing weight $p_j(x)$ tends to be large if the input x fires the if-part set A_j to a high degree. This properly gives more weight to the corresponding then-part uncertainty σ_j^2. Larger then-part sets lead to larger variance values σ_j^2. This may lead the user to inverse-scale the rule weight as $w_j = \frac{1}{\sigma_j^2}$ to counteract this effect.

The second term is a weighted penalty term. The quadratic term $[c_j - F(x)]^2$ is an interpolation penalty. Each rule tugs the global output toward its local centroid c_j. An input x that fires A_j to a high degree tends to have a large weight $p_j(x)$. Then the jth centroid c_j might well dominate the convex sum and so $F(x) \approx c_j$ may hold. But other inputs x may not fire any if-part set to high degree. This occurs when the system interpolates for missing rules. The system output $F(x)$ becomes increasingly unreliable as the conditional variance $V[Y|X = x]$ increases. Large values of $V[Y|X = x]$ imply that the system is essentially guessing at the output relative to its stored set of m rules.

An important fact is that any SAM system that can compute a fuzzy output $F(x)$ in (62) can also compute the conditional variance $V[Y|X = x]$ in (65) with the *same* information. Yet apparently no published fuzzy system had ever done so until the 2005 ballistic application paper.[9] So the reader does not know for those thousands of other published systems which outputs $F(x)$ involve more inherent interpolation-based uncertainty than other outputs. A fuzzy engineer could in principle incur negligence liability for not disclosing this inherent system limitation if such failure to disclose causes foreseeable downstream harm.

A matrix covariance version of (65) also holds for vector-valued fuzzy systems $F : \mathbb{R}^n \to \mathbb{R}^p$ and may be of use in problems of state estimation or data fusion. It shows that the $p \times p$ conditional covariance matrix $K_{Y|X=x} = E[(Y - E[Y|X = x])(Y - E[Y|X = x])^T|X = x]$ is a similar convex combination of an inherent then-part covariance term and an interpolation-penalty term:

$$K_{Y|X=x} = \sum_{j=1}^{m} p_j(x)\, K_{B_j(x)} + \sum_{j=1}^{m} p_j(x)\, [c_j(x) - F(x)][c_j(x) - F(x)]^T \qquad (67)$$

where $K_{B_j(x)} = \int_{\mathbb{R}^p} [y - c_j(x)][y - c_j(x)]^T p_{B_j}(y|x)\, dy$ for $y \in \mathbb{R}^p$.

The Moment Theorem also applies to the combination of q additive fuzzy systems $F^1, \ldots, F^q$. Then the combined mixture density $p(y|x)$ in (37) gives the

combined system F in (61) as a double sum:

$$F(x) = \sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x) c_j^k(x) \tag{68}$$

with convex mixing weights $p_j^k(x)$ in (38). This double sum admits recursive formulation[5] for adding new fuzzy systems. The SAM version again drops the dependency on x for the then-part centroid c_j. The corresponding conditional variance is also a double sum:

$$V[Y|X = x] = \sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x) \sigma_{B_j(x)}^{2,k} + \sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x) [c_j^k(x) - F(x)]^2. \tag{69}$$

Similar double-sum expansions hold for all higher-order moments when combining q additive systems.

3.2. Combining Fuzzy Function Approximation and Representation

A SAM fuzzy system can uniformly approximate any continuous function $f : K \subset \mathbb{R}^n \to \mathbb{R}^p$ if K is a compact subset of $\mathbb{R}^n$.[2,5,56] Uniform approximation means that $|F(x) - f(x)| < \varepsilon$ holds for *all* vector inputs x. So the user can pick the desired minimum error level ε in advance and still be sure that the approximation holds for all inputs.

The proof is constructive and here we only sketch it. The proof combines the convex-sum structure of the SAM system in (62) with two properties of compact sets. The convex structure traps the output $F(x)$ in $c_{\min} \leq F(x) \leq c_{\max}$ for a scalar system $F : \mathbb{R}^n \to \mathbb{R}$. A related centroid-hyperrectangle bound holds in the vector-valued case. Then the output vector $F(x) \in \mathbb{R}^p$ always lies in the bounding hypercube. The first compactness property holds for mappings between metric spaces. It states that a continuous function on a compact set is uniformly continuous. Uniform continuity ensures an ε-based distance bound for all points that lie within a fixed distance $\delta > 0$ from one another. The second compactness property is the defining property that every open cover of K has a finite sub-cover.[57] So a finite set of overlapping hypercubes can cover K. The proof arranges these cubes so that each cube corner lies at the midpoint d_j of its neighboring cube. Then it centers the then-part sets B_j so that their centroids lie at $f(d_j)$. These properties lead to the uniform result that $|F(x) - f(x)| < \varepsilon$ for all $x \in K$.

The uniform approximation theorem still holds for the more general additive case so long as the variable fired then-part sets $B_j(x)$ still have the same centroid c_j for all x. So it holds for a min-clip rule firing $a_j(x) \wedge B_j$ if B_j is symmetric because then $a_j(x) \wedge B_j$ and B_j have the same centroid c_j. It need not hold for the supremum defuzzification (14) because then the output $F(x)$ may fall outside the centroid interval $[c_{\min}, c_{\max}]$ or the p-dimensional hyperbox extension.[5]

Fuzzy function representation lets a SAM system F equal f exactly: $F(x) = f(x)$ for all x. This is trivial for a constant function $f(x) = c$ for all x. Then a SAM

with just one rule gives $F(x) = c$ if the sole then-part set B has centroid c. This holds because any nonzero firing $a(x) > 0$ gives $B(x) = a(x)B$. So $B(x)$ has the same centroid c as B has. Then the centroidal output is $F(x) = c$.

The Watkins Representation Theorem proves the remarkable result that a scalar SAM system with just two rules can exactly represent a non-constant function f if f is bounded.[24,25] The catch is that the user must know the closed form of f and then build it into the if-part set structure of the two rules. So the theorem offers little or no guidance in general blind approximation where the user has access only to a few noisy input-output samples of f. But there are cases where the user does know the form of f and can then absorb that structure into the fuzzy system. This occurs in standard Bayesian statistics where the user assumes a closed form prior pdf $h(\theta)$ and a closed form likelihood pdf $g(x|\theta)$. The pdfs are almost always bounded in practice. So a fuzzy system can directly absorb them[13,14] as Theorem 3 shows.

The Watkins Representation Theorem is so important and so little known that we next state and prove it as a separate proposition. Then we will examine some of its consequences and it extend it to representing multiple bounded functions when combining several additive fuzzy systems. The original Watkins Representation Theorem applied to scalar-to-scalar bounded functions $f : \mathbb{R} \to \mathbb{R}$. But the same argument applies to any finite or even infinite input space. So we first state and prove it for the usual scalar-SAM case of $f : \mathbb{R}^n \to \mathbb{R}$. The result extends at once to the vector case $f : \mathbb{R}^n \to \mathbb{R}^p$ where the representation requires $2p$ rules. The two rules in the scalar case have the form "If X is A then Y is B_1" and "If X is not-A then Y is B_2". The set function $a : \mathbb{R}^n \to \mathbb{R}$ for A absorbs the bounded function f into its structure.

PROPOSITION 1. *Watkins Representation*[24,25]*: Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is bounded and not a constant. Then the SAM fuzzy system $F : \mathbb{R}^n \to \mathbb{R}$ in* (62) *can exactly represent f with just two rules: $F(x) = f(x)$ for all $x \in \mathbb{R}^n$.*

Proof. The function f has a supremum $\beta = \sup_{x \in \mathbb{R}^n} f(x)$ and an infimum $\alpha = \inf_{x \in \mathbb{R}^n} f(x)$ because f is bounded. Then $\beta > \alpha$ because f is not constant. Then define the set function of A_1 as $a_1(x) = \frac{\beta - f(x)}{\beta - \alpha}$. Define the set function of A_2 as $a_2(x) = 1 - a_1(x)$. Pick unit rule weights $w_j = 1$ and then-part volumes $V_j = 1$ so that they drop out of the SAM system (62). Center B_1 at centroid $c_1 = \alpha$ and center B_2 at $c_2 = \beta$. Then the two-rule SAM has the form

$$F(x) = \frac{\sum_{j=1}^{2} a_j(x)\ w_j\ V_j\ c_j}{\sum_{k=1}^{2} a_k(x)\ w_k\ V_k} \tag{70}$$

$$= \frac{a_1(x)\ \alpha + (1 - a_1(x))\ \beta}{a_1(x) + 1 - a_1(x)} \tag{71}$$

$$= \left(\frac{\beta - f(x)}{\beta - \alpha} \right) (\alpha - \beta) + \beta \tag{72}$$

$$= f(x). \tag{73}$$

□

A simple example is $f(x) = \sin x$. Sine is bounded with $\alpha = -1 = c_1$ and $\beta = 1 = c_2$. So $a_1(x) = \frac{1-\sin x}{2}$ and $a_2(x) = \frac{1+\sin x}{2}$. Then $F(x) = \sin x$.

A more sophisticated example is the beta pdf $Beta(a, b)$ over the unit interval. Bayesian models often use a beta prior $B(a, b)$ to model a binomial success parameter θ with a conjugate binomial likelihood. Applying Bayes theorem after n Bernoulli trials gives the posterior pdf as the updated beta $B(a + x, b + n - x)$ if x is the number of successes and $n - x$ is the number of failures out of n trials.[13] Then the new beta pdf $B(a + x, b + n - x)$ can serve as the new prior in the next round of binomial trials. The beta pdf $f(\theta)$ with positive scaling parameters a and b has the form $f(\theta) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}$ if Γ is the gamma function: $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}e^{-x}dx$ with $\alpha > 0$. The values θ of the random variable Θ lie in the unit interval $[0, 1]$. Consider the skewed case of $a = 8$ and $b = 5$. Then $\alpha = 0$ and $\beta = \frac{12!}{7!4!}\frac{7^7 4^4}{11^{11}}$ since $\Gamma(n) = (n-1)!$ for any positive integer n. This gives $a_1(\theta) = 1 - \frac{11^{11}}{7^7 4^4}\theta^7(1-\theta)^4$. Then F equals $Beta(8, 5)$: $F(\theta) = \frac{12!}{7!4!}\theta^7(1-\theta)^4$.

The vector extension to bounded functions $f : \mathbb{R}^n \to \mathbb{R}^p$ requires that each scalar function f_k in $f = (f_1, \ldots, f_p)$ be both bounded and non-constant. Define the set function of A_1^k as the first *Watkins coefficient* $a_1^k(x) = \frac{\beta_k - f_k(x)}{\beta_k - \alpha_k}$. So the second Watkins coefficient is $a_2^k(x) = 1 - a_1^k(x) = \frac{f_k(x) - \alpha_k}{\beta_k - \alpha_k}$. Center the kth component's first then-part set B_1^k at the centroid $c_1^k = \alpha_k$. Center B_2^k at the centroid $c_2^k = \beta_k$. Then each SAM vector component F_k is a two-rule scalar-valued SAM that represents f_k as before. So $F(x) = (f_1(x), \ldots, f_p(x)) = f(x)$.

An example of a 2-D Watkins representation is the parametrized unit circle in the plane: $f(x) = (\sin x, \cos x)$ where $0 \leq x < 2\pi$. We have again for sine that $\alpha_1 = -1 = c_1^1$ and $\beta_1 = 1 = c_2^1$. So its Watkins coefficients are $a_1^1(x) = \frac{1-\sin x}{2}$ and $a_2^1(x) = \frac{\sin x+1}{2}$. Cosine is bounded and likewise gives $a_1^2(x) = \frac{1-\cos x}{2}$ and $a_2^2(x) = \frac{\cos x+1}{2}$. Then a 4-rule SAM F represents the unit circle as $F(x) = (\sin x, \cos x)$.

The vector Watkins representation fails for the 3-dimensional helix $f(x) = (\sin x, \cos x, x)$ unless the domain of x is bounded because the identity function $f(x) = x$ need not be bounded. The scalar identity function has as its first Watkins coefficient $a(x) = \frac{v-x}{v-u}$ if $x \in [u, v]$. Note also that the Watkins representation in Proposition 1 still holds for TSK systems[26,27] because TSK systems are additive fuzzy systems[5] and because we can take the output then-part sets to be their constant centroids.

Adding a Watkins representation to a fuzzy system takes care. The two-rule representation of Proposition 1 exhausts the rule base of a simple fuzzy system. The SAM system contains only the two Watkins rules. So we instead absorb the Watkins representation of some function f into the double-sum combined SAM of (68). This means including the two Watkins rules into the combined rule-firing sum $B(x)$ in (36). The next two cases demonstrate the combination technique. The second case generalizes the first. Theorem 3 states the final result.

The first case combines r non-constant bounded real functions $f_1, \ldots, f_r$ into the SAM F in (68). This generalizes Proposition 1 so that F represents a sum of r bounded functions in a *single* rule base of $2r$ rules. The mixture $p(y|x)$ in (103)

below represents this sum as its first moment. We state this new generalization of Proposition 1 as a separate corollary. The vector version is immediate.

COROLLARY 2. *Generalized Watkins Representation for r combined SAM fuzzy systems: Suppose the r real functions $f_1, \ldots, f_r$ are bounded and not constant. Then the combined SAM fuzzy system $F : \mathbb{R}^n \to \mathbb{R}$ in* (68) *is just the sum of the r represented functions:*

$$F(x) = \sum_{k=1}^{r} f_k(x) \tag{74}$$

for all $x \in \mathbb{R}^n$.

Proof. Use the same representation scheme as in the proof of Proposition 1 but with one key change: Scale the kth representation's two then-part centroids as $c_1^k = r\alpha^k$ and $c_2^k = r\beta^k$. Then the result follows from the simple structure of the convex weights $p_j^k(x)$ for a Watkins representation: $p_1^k(x) = \frac{a_1^k(x)}{\sum_{k=1}^r (a_1^k(x)+1-a_1^k(x))} = \frac{a_1^k(x)}{r}$ and $p_2^k(x) = \frac{a_2^k(x)}{r}$. Then $F(x) = \sum_{k=1}^r \sum_{j=1}^2 p_j^k(x) c_j^k = \sum_{k=1}^r (a_1^k(x)\alpha^k + a_2^k(x)\beta^k) = \sum_{k=1}^r f_k(x)$ as in the proof of Proposition 1. □

Corollary 2 has an easy generalization to the case of non-equal system weights $u^1, \ldots, u^r$. The system weights $u^1, \ldots, u^r$ give $F(x)$ as the convex combination $F(x) = \sum_{k=1}^r \lambda^k f_k(x)$ where $\lambda^k = \frac{u^k}{\sum_{k=1}^r u^k}$ if $c_1^k = \alpha^k$ and $c_2^k = \beta^k$.

The representation sum (74) obeys a Bayes theorem that gives the posterior probability $p_{\text{rep}}(k|y, x)$ that the kth representation f_k "fired" or appeared in the combined system given that the input x produced output y or $F(x)$. We can interpret $p_{\text{rep}}(k|y, x)$ as the relative importance of f_k to the combined output $F(x)$. The Watkins representations simplify the mixture density $p(y|x)$ in (37) to give the combined Watkins mixture $p_{\text{rep}}(y|x)$ as

$$p_{\text{rep}}(y|x) = \frac{1}{r} \sum_{k=1}^{r} \sum_{j=1}^{2} a_j^k(x) b_j^k(y). \tag{75}$$

Then $p_{\text{rep}}(y|x)$ and (41) give the posterior pdf $p_{\text{rep}}(k|y, x)$ as

$$p_{\text{rep}}(k|y, x) = \frac{a_1^k(x)(b_1^k(y) - b_2^k(y)) + b_2^k(y)}{\sum_{k=1}^r [a_1^k(x)(b_1^k(y) - b_2^k(y)) + b_2^k(y)]}. \tag{76}$$

The Bayes result (76) shows again that the shapes of the then-part sets B_j^k matter even for exact Watkins representations. The representations assume only that each B_j^k has unit area or volume V_j^k. But the two then-part fit values $b_1^k(y)$ and $b_2^k(y)$ can still differ for the same non-rectangular then-part set B and thus can produce different values of $p_{\text{rep}}(k|y, x)$. The case of congruent rectangles B_j^k gives $b_j^k(y) = c > 0$ for k and j. Then (76) reduces to the discrete uniform density $p_{\text{rep}}(k|y, x) = \frac{1}{r}$ in the

unweighted case. It reduces to $p_{\text{rep}}(k|y, x) = \lambda_k$ for r weighted representations with weights $u^1, \ldots, u^r$. Even the unweighted case of congruent rectangles B_j^k gives a conditional variance $V[Y|X = x]$ in (69) that varies with the input x.

The second combination case generalizes the first. It combines the same r non-constant bounded functions $f_1, \ldots, f_r$ with q rule-based SAM approximators $F^1, \ldots, F^q$. So it generalizes the sum result (74). It shows that the combined system is a convex-weighted sum of two terms. The first term is just the r summed representations (74). The second term is the double sum that combines q approximators as in (68). The result separates the representations from the approximations and then combines the separated systems with higher-level convex weights that depend on x. We next state and prove this general separation theorem for fuzzy knowledge combination.

THEOREM 3. *Separation Theorem for combining fuzzy representations with approximations. Suppose that the r real functions $f_1, \ldots, f_r$ are bounded and non-constant. Suppose that the q functions $F^1, \ldots, F^q$ are SAM fuzzy systems. Combine all $r + q$ functions in a single SAM fuzzy system F as in* (68)*. Then F is a convex combination of the representation sum* (74) *and the double SAM sum* (68) *of the q SAMs:*

$$F(x) = s(x) \sum_{k=1}^{r} f_k(x) + (1 - s(x)) \sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x) c_j^k \tag{77}$$

for convex weights $s(x) = \frac{V_{rep}(x)}{V_{rep}(x)+V_{rule}(x)}$ *and* $1 - s(x) = \frac{V_{rule}(x)}{V_{rep}(x)+V_{rule}(x)}$.

Proof. Represent each function f_k with two rules as in the result (74) above. Define $V_{\text{rep}}(x) = \int_{\mathbb{R}} b_{\text{rep}}(y|x)dy$ and $V_{\text{rule}}(x) = \int_{\mathbb{R}} b_{\text{rule}}(y|x)dy$. Split the total fired then-part set $B(x)$ into representation and rule-based terms:

$$B(x) = B_{\text{rep}}(x) + B_{\text{rule}}(x) \tag{78}$$

with $u_1 = \cdots = u_{r+q} = 1$ without any loss of generality. Then

$$F(x) = \text{Centroid}(B_{\text{rep}}(x) + B_{\text{rule}}(x)) \tag{79}$$

$$= \frac{\int_{\mathbb{R}} y b_{\text{rep}}(y|x)dy + \int_{\mathbb{R}} y b_{\text{rule}}(y|x)dy}{V_{\text{rep}}(x) + V_{\text{rule}}(x)} \tag{80}$$

$$= \left(\frac{V_{\text{rep}}(x)}{V_{\text{rep}}(x) + V_{\text{rule}}(x)}\right) \left[\frac{\int_{\mathbb{R}} y b_{\text{rep}}(y|x)dy}{V_{\text{rep}}(x)}\right] + \left(\frac{V_{\text{rule}}(x)}{V_{\text{rep}}(x) + V_{\text{rule}}(x)}\right) \left[\frac{\int_{\mathbb{R}} y b_{\text{rule}}(y|x)dy}{V_{\text{rule}}(x)}\right] \tag{81}$$

$$= s(x)\text{Centroid}(B_{\text{rep}}(x)) + (1 - s(x))\text{Centroid}(B_{\text{rule}}(x)) \tag{82}$$

$$= s(x)F_{\text{rep}}(x) + (1 - s(x))F_{\text{rule}}(x) \tag{83}$$

$$= s(x) \sum_{k=1}^{r} f_k(x) + (1 - s(x)) \sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x) c_j^k \tag{84}$$

if we replace $F_{\text{rep}}(x)$ with (74) and replace $F_{\text{rule}}(x)$ with the SAM version of (68). □

The same argument shows that the combination's mixture $p(y|x)$ obeys a related decomposition into a mixture of representation and rule-based mixtures:

$$p(y|x) = s(x) p_{\text{rep}}(y|x) + (1 - s(x)) p_{\text{rule}}(y|x). \tag{85}$$

The mixture (85) gives a macro-level Bayes theorem for the probability $p(\text{Representation}|y, x)$ that the observed output y or $F(x)$ given input x is due to $F_{\text{rep}}(x)$ or the representation sum (74) rather than due to $F_{\text{rule}}(x)$ or the q combined SAM systems in (68):

$$p(\text{Representation}|y, x) = \frac{s(x) p_{\text{rep}}(y|x)}{s(x) p_{\text{rep}}(y|x) + (1 - s(x)) p_{\text{rule}}(y|x)}. \tag{86}$$

Then $p(\text{Rule-based}|y, x) = 1 - p(\text{Representation}|y, x)$.

The Separation Theorem and its corollaries can apply to many forms of combining or fusing q knowledge sources $\mathcal{K}^1, \ldots, \mathcal{K}^q$.[58] The knowledge sources can in principle include feedback knowledge nets such as fuzzy cognitive maps.[59–62] Adaptive fuzzy function approximation can use sample data to grow and tune a set of m_k rules so that the kth combined fuzzy system $F^k : \mathbb{R}^n \to \mathbb{R}^p$ approximates a given scalar or vector-valued knowledge source $\mathcal{K}^k : \mathbb{R}^n \to \mathbb{R}^p$. Lack of sample data requires that the fuzzy engineer guess at the initial set of rules or else ask experts for such rules. The engineer can also try to infer the rules from expert documents or watching expert behavior. Then supervised learning may still be able to at least partially tune the F^k rule parameters given some performance samples from $\mathcal{K}^k$. The next two sections show how to tune a fuzzy system's governing mixture $p(y|x)$ and its rule parameters with training data. Unsupervised clustering can also tune parameters. It does so mainly by initializing rule patches to clusters in the sample data.[1,4,5,7–9]

A Watkins representation may apply in the case of scalar-valued data or knowledge sources $\mathcal{K}^k : \mathbb{R}^n \to \mathbb{R}$. A natural application is to pattern classifiers such as linear classifiers or logistic regressors or feedforward neural networks with a single output classifier neuron.[63–65] These systems map an n-dimensional pattern vector $x \in \mathbb{R}^n$ to the output value 1 if x belongs to a given pattern class and map to 0 or -1 otherwise. The pattern inputs may also map to a fuzzy or gray-scale value in the binary unit interval $[0, 1]$ or in the bipolar interval $[-1, 1]$. A simple Watkins combiner can use a modified version of Corollary 2 that weights or mixes the r classifiers f^k to give $F(x) = \sum_{k=1}^{r} \lambda^k f^k(x)$ for convex weights $\lambda^1, \ldots, \lambda^r$ as discussed above. Vector-valued Watkins representation can likewise absorb vector-valued knowledge sources $\mathcal{K}^k : \mathbb{R}^n \to \mathbb{R}^p$. A SAM system can learn from the neural net by sampling from it.

Combining vector-valued classifiers can also use fuzzy rule-based approximators. An important case is the family of mixture-of-experts models[66–68] that combines classifiers or neural networks with generalized mixing weights that depend on the input x. The Separation Theorem can in principle combine these and other committee-based systems along with Watkins-represented classifiers.

We turn next to how to acquire the generalized mixtures $p(y|x)$ that underlie additive fuzzy systems.

3.3. Finding Generalized Mixture Densities for Additive Fuzzy Systems

Using a mixture approach to fuzzy systems requires answering a basic question: Where does the generalized mixture density $p(y|x)$ come from? There are several ways to arrive at a representative mixture $p(x|y)$.

The simplest approach is that of classical knowledge engineering. The user simply defines m desired if-then rules $R_{A_1 \to B_1}, \ldots, R_{A_m \to B_m}$. Then the mixture $p(y|x)$ arises from the additive combination structure and (13). This step converts the fuzzy-set design of the rule base into a purely probabilistic system for further processing.

A more technical approach converts the given associated sets (A_j, B_j) of the rules into a generalized mixture where the normalized rule weights w_j define the convex mixing weights:

$$p(y|x) = \sum_{j=1}^{m} v_j a_j(x) b_j(y) \tag{87}$$

where the convex weight v_j has the form

$$v_j = \frac{w_j}{\sum_{k=1}^{m} w_k} \tag{88}$$

for positive rule weights w_k. This approach assumes that the user can adjust the center and width of a density to represent linguistic fuzzy-set values such as COOL and WARM in Figure 1. We assume without loss of generality that the set functions a_j and b_j have unit volumes and thus behave as ordinary probability density functions. Dividing by non-unity volumes achieves the same result. Then integration and the Moment Theorem give the fuzzy system output $F(x)$ as the convex combination of the fit-valued-scaled centroids $a_j(x)\, c_j$:

$$F(x) = \int y \; p(y|x) \, dy \tag{89}$$

$$= \sum_{j=1}^{m} v_j \; a_j(x) \int y \; b_j(y) \, dy \tag{90}$$

$$= \sum_{j=1}^{m} v_j \; a_j(x) \; c_j . \tag{91}$$

A still more sophisticated approach uses the Watkins representation structure of Proposition 1 to find $p(y|x)$. Consider the problem of finding a generalized mixture $p(y|x)$ such that $F(x) = f(x)$ holds exactly for all x for a given bounded real-valued function f. What $p(y|x)$ produces $F = f$?

We show now that a simple two-term mixture $p(y|x)$ will always achieve the representation $F = f$ so long as the target function f is bounded and not constant. The vector-valued version mixes $2p$ densities.

THEOREM 4. *Mixture-based Representation Theorem. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is bounded and not a constant. Suppose that the generalized mixture density $p(y|x)$ mixes two then-part pdfs with Watkins coefficients:*

$$p(y|x) = \left(\frac{\beta - f(x)}{\beta - \alpha}\right) p_{B_1}(y) + \left(\frac{f(x) - \alpha}{\beta - \alpha}\right) p_{B_2}(y) \tag{92}$$

for equal rule weights $w_1 = w_2 > 0$ and then-part set volumes $V_1 = V_2 > 0$ and where $c_1 = \alpha = \inf_{x \in \mathbb{R}^n} f(x)$ and $c_2 = \beta = \sup_{x \in \mathbb{R}^n} f(x)$. Then the SAM system $F : \mathbb{R}^n \to \mathbb{R}$ based on $p(y|x)$ represents f: $F(x) = f(x)$ for all x.

Proof. The Watkins coefficients $\frac{\beta - f(x)}{\beta - \alpha}$ and $\frac{f(x) - \alpha}{\beta - \alpha}$ are convex weights because they are nonnegative and sum to one. Each convex coefficient $p_j(x)$ equals the corresponding if-part set function $a_j(x)$: $p_j(x) = \frac{w_j V_j a_j(x)}{\sum_{k=1}^m w_k V_k a_k(x)} = \frac{a_j(x)}{a_1(x) + 1 - a_1(x)} = a_j(x)$. But $a_1(x) = \frac{\beta - f(x)}{\beta - \alpha}$ and $a_2(x) = \frac{f(x) - \alpha}{\beta - \alpha}$. So the two-rule SAM structure gives

$$p(y|x) = \sum_{j=1}^{2} p_j(x)\ p_{B_j}(y) \tag{93}$$

$$= \left(\frac{\beta - f(x)}{\beta - \alpha}\right) p_{B_1}(y) + \left(\frac{f(x) - \alpha}{\beta - \alpha}\right) p_{B_2}(y). \tag{94}$$

Then the Moment Theorem (52) and the centroid definitions $c_1 = \alpha = \inf_{x \in \mathbb{R}^n} f(x)$ and $c_2 = \beta = \sup_{x \in \mathbb{R}^n} f(x)$ give the fuzzy system output $F(x)$ as

$$F(x) = E[Y|X = x] \tag{95}$$

$$= \int_{\mathbb{R}} y\ p(y|x)\, dy \tag{96}$$

$$= \left(\frac{\beta - f(x)}{\beta - \alpha}\right) \int_{\mathbb{R}} y\ p_{B_1}(y)\, dy + \left(\frac{f(x) - \alpha}{\beta - \alpha}\right) \int_{\mathbb{R}} y\ p_{B_2}(y)\, dy \tag{97}$$

$$= \left(\frac{\beta - f(x)}{\beta - \alpha}\right) c_1 + \left(\frac{f(x) - \alpha}{\beta - \alpha}\right) c_2 \tag{98}$$

$$= \left(\frac{\beta - f(x)}{\beta - \alpha}\right) \alpha + \left(\frac{f(x) - \alpha}{\beta - \alpha}\right) \beta \tag{99}$$

$$= \frac{\beta\alpha - f(x)\alpha + f(x)\beta - \alpha\beta}{\beta - \alpha} \tag{100}$$

$$= \frac{f(x)[\beta - \alpha]}{\beta - \alpha} \tag{101}$$

$$= f(x). \tag{102}$$

□

This argument shows that in general $E[Y|X = x] = f(x)$ holds for all x for *all* probabilistic system if the first moments exist. Gaussian then-part sets suffice.

The final section shows how this theorem applies in computing with fuzzy rule continua. The result is the Monte Carlo approximation of the bounded non-constant real function $\sin x$ in Figure 3. It mixes two unit-variance normal bell-curve pdfs with appropriate Watkins coefficients based on the target function $\sin x$.

Theorem 4 extends to the sum of r bounded real functions $f_1, \ldots, f_r$ in Corollary 2. The new mixture $p(y|x)$ mixes $2r$ likelihood pdfs $p_{B_{j_k}}(y)$ with scaled Watkins coefficients for generalized mixing weights:

$$p(y|x) = \sum_{j=1}^{r} \left[\frac{1}{r} \left(\frac{\beta - f_j(x)}{\beta - \alpha} \right) p_{B_{j_1}}(y) + \frac{1}{r} \left(\frac{f_j(x) - \alpha}{\beta - \alpha} \right) p_{B_{j_2}}(y) \right] \tag{103}$$

where $\text{Centroid}(B_{j_1}) = \int_{\mathbb{R}} y p_{B_{j_1}}(y) dy = r\alpha_j$ and $\text{Centroid}(B_{j_2}) = \int_{\mathbb{R}} y p_{B_{j_2}}(y) dy = r\beta_j$. Then $F(x) = \int_{\mathbb{R}} y p(y|x) dy = \sum_{j=1}^{m} f_j(x)$. Unit-variance normal likelihoods suffice in (103): $p_{B_{j_1}}(y) = \mathcal{N}(r\alpha_j, 1)$ and $p_{B_{j_2}}(y) = \mathcal{N}(r\beta_j, 1)$.

A natural question is whether the two-pdf mixture representation in (94) is robust against noisy or other perturbations of the two mixed densities. The proof shows that the representation does not depend on the variance or higher-moments of the two mixed densities. Suppose we additively perturb the mean of p_{B_1} by $\epsilon > 0$ and the mean of p_{B_2} by $\delta > 0$. Then the respective means become $\alpha + \epsilon$ and $\beta + \delta$ in the proof at (99). This gives the perturbed representation as

$$F(x) = \frac{\beta - f(x)}{\beta - \alpha}(\alpha + \epsilon) + \frac{f(x) - \alpha}{\beta - \alpha}(\beta + \delta) \tag{104}$$

$$= f(x) + \epsilon \frac{\beta - f(x)}{\beta - \alpha} + \delta \frac{f(x) - \alpha}{\beta - \alpha} \tag{105}$$

$$= f(x) + \epsilon a_1(x) + \delta a_2(x). \tag{106}$$

So the special case of identical perturbations $\epsilon = \delta$ gives a simple additive perturbation:

$$F(x) = f(x) + \epsilon \tag{107}$$

since these set-function values are convex and obey $a_1(x) + a_2(x) = 1$ for all x. The same argument shows that the multiplicative perturbations $\epsilon\alpha$ and $\delta\beta$ lead to

$$F(x) = \epsilon a_1(x)\alpha + \delta a_2(x)\beta \ . \tag{108}$$

Table I. Additive Perturbation Analyis.

	Average Squared Error			
Samples	$\epsilon = 0.0$	$\epsilon = 0.1$	$\epsilon = 0.5$	$\epsilon = 1.0$
500	0.0091	0.0184	0.2611	1.0083
1,000	0.0045	0.0142	0.2524	1.0078
2,000	0.0027	0.0119	0.2509	1.0021
5,000	0.0012	0.0012	0.2507	1.0004
10,000	0.0007	0.0010	0.2499	1.0000

Table II. Multiplicative Perturbation Analysis.

	Average Squared Error			
Samples	$\epsilon = 0.5$	$\epsilon = 1.0$	$\epsilon = 1.5$	$\epsilon = 2.0$
500	0.1336	0.0089	0.1228	0.5027
1,000	0.1320	0.0045	0.1177	0.4571
2,000	0.1313	0 0024	0.1114	0.4550
5,000	0.1320	0.0011	0.1085	0.4532
10,000	0.1312	0.0008	0.1061	0.4524

Then the special case of $\epsilon = \delta$ gives a simple scaled perturbation:

$$F(x) = \epsilon f(x) \; . \tag{109}$$

Both perturbation results hold for the original Watkins Representation Theorem itself because the hypotheses of the theorem also equate the centroid of the first then-part set with the infimum α of f and the second centroid with the supremum β. So the perturbations still apply to the same proof step (99). The somewhat ironical result is that the perturbed representations (106) and (108) show that we can approximate exact representations. The user need not use exactly centered mixed pdfs to get an approximate result.

Table I shows four categories of Monte Carlo simulation results of the additive perturbation (107) for the target function $f(x) = \sin x$ in Figure 3. The first category is the perturbation-free case of $\epsilon = 0$. The three other categories are the perturbations $\epsilon = .1, .5$, and 1. The simulation accuracy improves slowly with sample size n with the inverse of $\sqrt{n}$. Table II similarly shows four categories of multiplicative perturbation (109) for the same target function. The perturbation-free category is $\epsilon = 1$. The other three categories of perturbation are $\epsilon = .5, 1.5$, and 2.

Theorem 4 has two structural limitations. The first is that the function f must be bounded. Truncation of an unbounded function can achieve this in practice. But it comes at the expense of throwing away information about f.

The second limitation can be the most daunting in practice: The user must know the closed form of the bounded function f. But the user may have access only to incomplete samples $(x_1, y_1), \ldots, (x_n, y_n)$ from an unknown source. The samples may also be noisy. Overcoming this second limitation requires using some approximation technique.

Density estimators can separately approximate the joint pdf $p(x, y)$ and the marginal pdf $p(x)$ that define the generalized mixture $p(y|x)$ through the ratio definition $p(y|x) = \frac{p(x,y)}{p(x)}$. Ordinary nonparametric kernel density estimators can serve this task. Kernel estimates generalize histograms.[67] This also leads to kernel-regression estimators for the conditional expectation or regression function $E[Y|X = x]$ and thus to the fuzzy system output $F(x)$.

Most kernel regressors have a ratio structure similar to the convex-combination structure of SAM fuzzy models because additive fuzzy systems add up fired then-part sets in a type of histogram before taking the centroid of the summed sets. But kernel regressors arise from an ad hoc choice of kernel functions. Kernel regressors also have estimation artifacts that depend on the shape of the kernel function and on the adaptation technique used to fit them to data.

The EM algorithm[32,69] offers a practical and reasonably efficient way to estimate the two densities $p(x, y)$ and $p(x)$ based on observed vector samples $(x_1, y_1), \ldots, (x_n, y_n)$. The basic EM trick introduces a new "hidden" variable z to the known variable x and its marginal $p(x)$ by rearranging the definition of conditional probability $p(z|x) = \frac{p(x,z)}{p(x)}$ to express the known marginal $p(x)$ as $p(x) = \frac{p(x,y)}{p(z|x)}$. The probabilities at iteration k depend on a vector of parameters Θ_k. EM works with the log-likelihood $\ln p(x|\Theta_k) = \ln p(x, z|\Theta_k) - \ln p(z|x, \Theta_k)$. So EM estimates the unknown variable z by conditioning on the known data x and the current parameters Θ_k. A basic theorem shows that maximizing the expectation of just the complete log-likelihood term $\ln p(x, z|\Theta_k)$ with respect to the density $p(z|x, \Theta_k)$ can only increase the original log-likelihood at each iteration k. The result is an iterative two-step algorithm that climbs the nearest hill of probability or log-likelihood. The EM algorithm generalizes the backpropagation algorithm of modern neural networks[37] and the k-means clustering algorithm[70] along with many other iterative algorithms. Carefully injected noise always speeds EM convergence on average[36,71] with the largest gains in the early steps up the hill of likelihood.

We assume that the vector samples $(x_1, y_1), \ldots, (x_n, y_n)$ are realizations of some unknown vector random variable Z where $Z^T = [X^T|Y^T]$. The input random variable X is a $d \times 1$ column vector. The output random variable Y is a $p \times 1$ column vector. So the concatenated random vector Z has dimension $(d + p) \times 1$.

Then use separate ordinary Gaussian mixture models (GMMs) $\hat{p}(z)$ and $\hat{p}(x)$ as in (1) to approximate the joint pdf $p(y, x)$ and the marginal pdf $p(x)$:

$$\hat{p}(z) = \hat{p}(x, y) = \sum_{j=1}^{m} \pi_j f_j(z) \tag{110}$$

$$\hat{p}(x) = \sum_{j=1}^{l} \tau_j g_j(x). \tag{111}$$

for convex mixing coefficients $\pi_1, \ldots, \pi_m$ and $\tau_1, \ldots, \tau_l$ that do *not* depend on z or x. The jth normal random vector Z has mean vector $\mu_j = E_{f_j}[Z]$ and positive-definite $(d + p) \times (d + p)$ covariance matrix $K_j = E_{f_j}[(Z - E_{f_j}[Z])(Z - E_{f_j}[Z])^T]$. The pdf f_j is the $(p + d)$-dimensional multivariate normal

density: $f_j(z) = \mathcal{N}(\mu_j, K_j) = \frac{1}{(2\pi)^{(d+p)/2}\sqrt{D(K_j)}} \exp\{-\frac{1}{2}(z-\mu_j)^T K_j^{-1}(z-\mu_j)\}$ if $D(K_j)$ is the determinant of the positive-definite covariance matrix K_j. The pdf g_j is the corresponding d-dimensional multivariate normal pdf.

EM's most popular use finds the maximum-likelihood parameters of a GMM.[31,32] Consider the GMM joint estimator $\hat{p}(x, y)$ in (110). EM estimates the m mixture weights π_j and the m vector means μ_j and covariance matrices K_j of the respective m mixed Gaussian pdfs $f_j(x)$ or $f_j(x|\mu_j, K_j)$. Then the EM algorithm has an especially simple form.[67,72] The parameter vector Θ_k gives the current estimate at iteration k of all $3m$ GMM parameters: $\Theta_k = \{\pi_1(k), \ldots, \pi_m(k), \mu_1(k), \ldots, \mu_m(k), K_1(k), \ldots, K_m(k)\}$. Then the iterations of the GMM–EM algorithm reduce to the following update equations based on the m posterior pdfs $p_W(j|z, \Theta)$ that arise from the Bayes theorem corollary (25) to the mixture density[32]:

$$\pi_j(k+1) = \frac{1}{m}\sum_{i=1}^{m} p_W(j|z_i, \Theta_k) \tag{112}$$

$$\mu_j(k+1) = \frac{\sum_{i=1}^{m} p_W(j|z_i, \Theta_k) z_i}{\sum_{i=1}^{m} p_W(j|z_i, \Theta_k)} \tag{113}$$

$$K_j(k+1) = \frac{\sum_{i=1}^{m} p_W(j|z_i, \Theta_k)(z_i - \mu_j(k))(z_i - \mu_j(k))^T}{\sum_{i=1}^{m} p_W(j|z_i, \Theta_k)} \tag{114}$$

where $W(z) = I_1(z) + \cdots + I_m(z)$. The binary function I_j is the jth indicator-function that codes for the jth class or subpopulation (or rule). The posterior probabilities $p_W(j|z_i, \Theta_k)$ have the ratio Bayesian form

$$p_W(j|z_i, \Theta_k) = \frac{\pi_j(k) f_j(z_i|W = j, \Theta_k)}{\sum_{l=1}^{m} \pi_l(k) f_l(z_i|W = l, \Theta_k)}. \tag{115}$$

3.4. Supervised Learning Laws for Mixtures and Fuzzy Systems

We next derive a new learning law for an additive fuzzy system's *finite* mixture density $p(y|x)$. Then we review and extend the supervised learning laws for an additive fuzzy system F and for higher moments. These learning laws are not practical for the continuum-rule systems of the next section.

A supervised learning law uses gradient descent to minimize some performance measure. The performance measure can be squared error or cross entropy or any other function of the system variables. We will focus on the traditional measure of squared error for simplicity. Unsupervised clustering can also intialize or otherwise tune parameters as well.[1,5]

We first show how supervised learning can tune or adapt a differentiable system mixture $p(y|x)$. Let $d(y|x)$ be a given desired mixture density for the fuzzy system. The desired probability $d(y|x)$ can be any conditional density of the form $\frac{p(x,y)}{p(x)}$.

It need not be a mixture. Define the mixture error $\varepsilon(x)$ as the *desired* outcome $d(y|x)$ minus the *actual* outcome $p(y|x)$: $\varepsilon = d(y|x) - p(y|x)$. This gives the scaled squared error SE for a sampled target mixture $d(y|x)$ as

$$SE = \frac{1}{2}\varepsilon^2 = \frac{1}{2}(d(y|x) - p(y|x))^2 \tag{116}$$

for observed samples $(x_1, y_1), (x_2, y_2), \ldots$ from $d(y|x)$. Let m_j be some parameter of the additive fuzzy system. Then the gradient descent learning law for m_j at iteration $t + 1$ is[1,5,13]

$$m_j(t+1) = m_j(t) - \mu_t \frac{\partial SE}{\partial m_j} \tag{117}$$

for a sequence of (usually decreasing) learning coefficients $\{\mu_t\}$. The chain rule factors the gradient learning term $\frac{\partial SE}{\partial m_j}$ as

$$\frac{\partial SE}{\partial m_j} = \frac{\partial SE}{\partial p(y|x)} \frac{\partial p(y|x)}{\partial m_j} \tag{118}$$

$$= -\varepsilon \frac{\partial p(y|x)}{\partial m_j} \tag{119}$$

from (116) and the definition of ε. So a gradient-based mixture learning law will have the iterative form

$$m_j(t+1) = m_j(t) + \mu_t \varepsilon_t \frac{\partial p(y|x)}{\partial m_j}. \tag{120}$$

The next theorem states the mixture learning law for the jth rule weight w_j of the jth rule in a given SAM system or the jth sub-system F^j in a combined system as in Theorem 3. The theorem shows that the crucial gradient learning term $\frac{\partial p(y|x)}{\partial m_j}$ in (120) depends on the difference $p_{B_j}(y) - p(y|x)$ between the jth rule pdf $p_{B_j}(y)$ in the mixture $p(y|x)$ and the mixture $p(y|x)$ itself. The local density $p_{B_j}(y)$ in effect pulls against the global density $p(y|x)$. The two terms are equal at equilibrium. This local-versus-global structure underlies all SAM-related learning laws.

THEOREM 5. *SAM Mixture Learning:*

$$w_j(t+1) = w_j(t) + \mu_t \varepsilon_t \frac{p_j(x)}{w_j}[p_{B_j}(y) - p(y|x)]. \tag{121}$$

Proof. The quotient rule of differentiation and the SAM version (33) of (22) give

$$\frac{\partial p(y|x)}{\partial w_j} = \frac{\partial}{\partial w_j}\left[\frac{\sum_{j=1}^m w_j a_j(x) V_j p_{B_j}(y)}{\sum_{k=1}^m w_k a_k(x) V_k}\right] \tag{122}$$

$$= \frac{a_j(x)V_j p_{B_j}(y) \sum_{k=1}^m w_k a_k(x)V_k - a_j(x)V_j \sum_{k=1}^m w_k a_k(x)V_k p_{B_k}(y)}{\left(\sum_{k=1}^m w_k a_k(x)V_k\right)^2} \tag{123}$$

$$= \frac{1}{w_j} \frac{w_j a_j(x)V_j}{\sum_{k=1} w_k a_k(x)V_k} p_{B_j}(y)$$

$$- \frac{1}{w_j} \frac{w_j a_j(x)V_j}{\sum_{k=1} w_k a_k(x)V_k} \frac{\sum_{j=1} w_j a_j(x)V_j p_{B_j}(y)}{\sum_{k=1} w_k a_k(x)V_k} \tag{124}$$

$$= \frac{1}{w_j} \frac{w_j a_j(x)V_j}{\sum_{k=1} w_k a_k(x)V_k} p_{B_j}(y) - \frac{p_j(x)}{w_j} p(y|x) \tag{125}$$

$$= \frac{1}{w_j} p_j(x) p_{B_j}(y) - \frac{p_j(x)}{w_j} p(y|x) \tag{126}$$

$$= \frac{p_j(x)}{w_j} [p_{B_j}(y) - p(y|x)] \tag{127}$$

using (63) for $p_j(x)$ and using (15) and (30) for $p_{B_j}(x)$. Put (127) in (120) to give the result (121). □

An equivalent Bayesian version of (121) reveals a learning difference that depends on the gap $p(j|y,x) - p_j(x)$ between the local posterior $p(j|y,x)$ of the jth rule or subsystem and the jth mixture weight $p_j(x)$. Corollary 1 gives the equality $p_j(x)p_{B_j}(y) = p(j|y,x)p(y|x)$. Then the learning term $\frac{\partial p(y|x)}{\partial w_j}$ has the equivalent form

$$\frac{\partial p(y|x)}{\partial w_j} = \frac{p(y|x)}{w_j} [p(j|y,x) - p_j(x)]. \tag{128}$$

Mixture learning slows when $p_j(x)$ or $p(y|x)$ are small. Mixture learning stops when either $p_{B_j}(y) = p(y|x)$ or $p(j|y,x) = p_j(x)$ holds. All conditional moments $\mu^{(k)}(x)$ of $p(y|x)$ admit similar learning laws. Their complexity increases directly with k.

The simplest moment learning law occurs for the conditional expectation $F(x) = E[Y|X = x]$ when $k = 1$. These learning laws are the usual adaptive SAM or ASAM learning laws.[5,8,13] But they are just one of the infinitely many learning laws that show how to adapt the moments of the general mixture density. Even the conditional-variance learning laws are complicated. So we now present the learning laws for only the first-moment case of $k = 1$.

We briefly review the ASAM learning laws as they apply to a single SAM system F with m rules. They also apply to combined fuzzy systems. The error ε is now the difference between the desired value of the first non-central moment and the actual value of the first moment. The Moment Theorem implies that this is just the difference between the sampled functional output $f(x)$ and the actual fuzzy system

output $F(x)$: $\varepsilon = F(x) - f(x)$. Then the squared error SE in (116) becomes

$$SE = \frac{1}{2}(f(x) - F(x))^2. \tag{129}$$

The chain rule and some manipulation give the ASAM law to update the jth rule weight w_j:

$$w_j(t+1) = w_j(t) + \mu_t \varepsilon_t \frac{p_j(x)}{w_j}[c_j - F(x)]. \tag{130}$$

Learning depends on how the local rule output or centroid c_j pulls against the global system output $F(x)$. Updating an independent then-part set volume V_j has the same learning law as updating the rule weight w_j in (130).

The volume learning law changes in the common case when the rule weight depends on the then-part volume and vice versa. Then the partial derivative $\frac{\partial F}{\partial V_j}$ in $\frac{\partial SE}{\partial V_j} = \frac{\partial SE}{\partial F}\frac{\partial F}{\partial V_j}$ expands as[4,5]

$$\frac{\partial F}{\partial V_j} = \mu_t \varepsilon_t p_j(x)\left(\frac{1}{V_j} + \frac{1}{w_j}\frac{\partial w_j}{\partial V_j}\right)[c_j - F(x)]. \tag{131}$$

So $\frac{\partial w_j}{\partial V_j} = 0$ holds when the rule weight w_j does not depend on the jth rule's then-part set volume V_j. This case gives back the volume version of (130) where V_j everywhere replaces w_j. Common practice sets the rule weights inversely proportional to the then-part volumes: $w_j = \frac{1}{V_j}$. Then such wider and thus less-certain then-part sets have less overall weight in the additive combination of fired then-part sets. This inverse weighting makes the rule weights w_j cancel out the then-part volumes V_j in the SAM convex coefficients $p_j(x)$ in (63). Then there is no learning law for the then-part volumes. The learning term (131) confirms this because then $\frac{1}{w_j}\frac{\partial w_j}{\partial V_j} = -\frac{1}{V_j}$ in (131) and so $\frac{\partial F}{\partial V_j} = 0$. A more extreme weighting is the inverse-square weight $w_j = \frac{1}{V_j^2}$. Then $\frac{1}{w_j}\frac{\partial w_j}{\partial V_j} = -2\frac{1}{V_j}$ holds. So (131) gives the sign-changed learning law

$$V_j(t+1) = V_j(t) - \mu_t \varepsilon_t \frac{p_j(x)}{V_j}[c_j - F(x)]. \tag{132}$$

The ASAM learning law for the then-part centroid c_j has the simplest form of all the ASAM learning laws:

$$c_j(t+1) = c_j(t) + \mu_t \varepsilon_t p_j(x). \tag{133}$$

The centroid ASAM law resembles the least-mean-square learning law that underlies the backpropagation algorithm for training multilayer neural networks.[1]

The ASAM learning laws for the if-part set functions $a_j : \mathbb{R}^n \to [0, 1]$ are the most complex. They involve the bulk of the ASAM computational burden because

of their n-dimensional structure. Most if-part set functions factor into the n-fold product of scalar fit values $a_j^1(x^1) \cdots a_j^n(x^n)$ as in (4). Exceptions occur with non-factorable ellipsoidal rules[4] and related covariational rules based on thick-tailed α-stable statistics.[73] Entire families of metric-based rules are not factorable and yet admit ASAM learning laws.[8] We here ignore these exceptions and assume that all if-part set functions factor in accord with (4). Each scalar if-part set function $a_j^i : \mathbb{R} \to [0, 1]$ depends on one or more parameters m_j^k such as a mean or a variance or dispersion term. Then the key learning term $\frac{\partial SE}{\partial m_j}$ in the parameter learning law (117) for the squared-error term SE in (129) involves four partial derivatives:

$$\frac{\partial SE}{\partial m_j} = \frac{\partial SE}{\partial F} \frac{\partial F}{\partial a_j} \frac{\partial a_j}{\partial a_j^k} \frac{\partial a_j^k}{\partial m_j^k} \tag{134}$$

$$= \frac{\partial SE}{\partial F} \frac{\partial F}{\partial a_j} \frac{a_j(x)}{a_j^k(x^k)} \frac{\partial a_j^k}{\partial m_j^k} \tag{135}$$

because $\frac{\partial a_j}{\partial a_j^k} = \prod_{l \neq k}^n a_j^l(x^l) = \frac{a_j(x)}{a_j^k(x^k)}$ from the factorization (4). Then the user need only expand the final partial derivative $\frac{\partial a_j^k}{\partial m_j^k}$ to produce an ASAM learning law for a factorable if-part set function a_j of a given shape.

An important example for adaptive function approximation is the sinc set function a_j^k centered at m_j^k with dispersion or width d_j^k: $a_j^k(x^k) = \sin(\frac{x^k - m_j^k}{d_j^k})/(\frac{x^k - m_j^k}{d_j^k})$. This nonmonotonic generalized set function takes values in $[-.217, 1]$. It also tends to be the best shape set for function approximation given a wide range of test functions when compared with a wide range of if-part set shapes.[8] The ASAM sinc laws[5] for the two parameters m_j^k and d_j^k are

$$m_j^k(t+1) = m_j^k(t) - \mu_t \varepsilon_t \frac{p_j(x)}{a_j^k(x^k)} [c_j - F(x)] \left(a_j^k(x^k) - \cos\left(\frac{x^k - m_j^k}{d_j^k} \right) \right) \frac{1}{x^k - m_j^k} \tag{136}$$

and

$$d_j^k(t+1) = d_j^k(t) - \mu_t \varepsilon_t \frac{p_j(x)}{a_j^k(x^k)} [c_j - F(x)] \left(a_j^k(x^k) - \cos\left(\frac{x^k - m_j^k}{d_j^k} \right) \right) \frac{1}{d_j^k}. \tag{137}$$

There are several other ASAM learning laws.[8,13] But they all involve sufficient computational complexity to make tuning even a 4-D SAM system $F : \mathbb{R}^4 \to \mathbb{R}$ difficult if not computationally infeasible. So the prospect of using such learning laws to tune an uncountable rule *continuum* appears hopeless. The next section

uses the mixture structure of additive fuzzy systems to extend the rule base to rule continua. It then uses Monte Carlo sampling to compute outputs.

4. FUZZY RULE CONTINUA

A simple change of rule index from the discrete index j to the continuous index θ extends a finite or denumerable rule base to a continuum rule base or rule continuum $\mathcal{RB}(\theta)$. This holds for any well-behaved mixture density such as the generalized mixture density $p(y|x)$ in (16). Learning laws can also tune the mixture. The discrete mixture naturally extends to an uncountable "compound"[30]

$$p(y|x) = \int_{\theta=-\infty}^{\theta=\infty} p_\theta(x)\ p_{B_\theta}(y|x)\ d\theta \tag{138}$$

so long as the integral exists. The integral will exist if either integrand pdf p_θ or p_{B_θ} is bounded. We state and prove this continuous mixture result (138) below as Theorem 6 in the special but important case of SAM rule factorization $b_\theta(y|x) = a_\theta(x)b_\theta(y)$ for real scalar rule index θ.

The scalar parameter θ picks out the rule $R_{A_\theta \to B_\theta}$ as it ranges over the rule continuum $\mathcal{RC}(\theta)$ in the additive combination

$$b(y|x) = \int_\Theta w_\theta\ b_\theta(y|x)\ d\theta \tag{139}$$

for $\theta \in \Theta$ and rule weight $w_\theta > 0$. The index set $\Theta \subset \mathbb{R}$ can be any nonempty connected subset of the real line $\mathbb{R}$ or of $\mathbb{R}^n$ in the vector case. The vector case requires $\Theta = (\Theta_1, \ldots, \Theta_n)$ where Θ_k is the scalar index for the kth integral.

The SAM rule firing $b_\theta(y|x)$ still expands as in (10): $b_\theta(y|x) = b_\theta(y)a_\theta(x)$ where $b_\theta : \mathbb{R} \to [0, 1]$ is the set function of the then-part fuzzy set $B_\theta \subset \mathbb{R}$ and $a_\theta : \mathbb{R}^n \to [0, 1]$ is the set function of the if-part fuzzy set $A_\theta \subset \mathbb{R}^n$. So the fuzzy structure remains and only the index changes.

We now show how to define a fuzzy rule continuum.

Consider a normal or Gaussian rule continuum $\mathcal{GRB}(\theta)$ for a scalar parameter θ. The rule $R_{A_\theta \to B_\theta}$ has a vector-Gaussian if-part set function $a_\theta : \mathbb{R}^n \to [0, 1]$ and a scalar-Gaussian then-part set function $b_\theta : \mathbb{R} \to \mathbb{R}$: $a_\theta(\cdot) = \mathcal{N}(\theta \bullet 1, K_\theta)$ and $b_\theta(y) = \mathcal{N}(\theta, \sigma^2)$. The term $\theta \bullet 1$ denotes the n-vector with all elements equal to θ. K_θ is an n-by-n covariance matrix: $K_\theta = E[(x - E[x])(x - E[x])^T]$ for column vector $x \in \mathbb{R}^n$. It equals the identity matrix in the simplest or "white" case.

Another way to define a rule continuum is to define the generalized mixture $p(y|x)$ that defines the continuous additive fuzzy system. We will show below how to use this indirect approach to compute actual outputs $F(x)$.

We first state and prove the basic mixture result for a SAM rule continuum. The proof closely tracks the proof of Theorem 1 because it replaces the discrete rule index j with the continuous rule index θ. The proof assumes that all integrals exist. Then Fubini's Theorem permits commutating the probabilistic integrals because the

integrands are nonnegative. All else is direct expansion and rearrangement as in the finite case.

THEOREM 6. *Generalized Mixture Theorem for a SAM Rule Continuum. Additive rule combination* (139) *and SAM rule-firing factorization* $b_\theta(y|x) = a_\theta(x)b_\theta(y)$ *define a continuum mixture probability density* $p(y|x)$*:*

$$p(y|x) = \int_\Theta p_\theta(x)\ p_{B_\theta}(y)\ d\theta \tag{140}$$

where the generalized mixture weights $p_\theta(x)$ *have the ratio form*

$$p_\theta(x) = \frac{w_\theta\ V_\theta\ a_\theta(x)}{\int_\Lambda w_\lambda\ V_\lambda\ a_\lambda(x)\ d\lambda}. \tag{141}$$

Proof. Normalize the additive rule firing $b(y|x)$ in (139) by its finite integral and expand:

$$p(y|x) = \frac{b(y|x)}{\int_{\mathbb{R}} b(y|x)\ dy} \tag{142}$$

$$= \frac{\int_\Theta w_\theta\ b_\theta(y|x)\ d\theta}{\int_{\mathbb{R}} \int_\Lambda w_\lambda\ b_\lambda(y|x)\ d\lambda\ dy} \tag{143}$$

$$= \frac{\int_\Theta w_\theta\ b_\theta(y|x)\ d\theta}{\int_\Lambda w_\lambda \int_{\mathbb{R}}\ b_\lambda(y|x)\ dy\ d\lambda} \tag{144}$$

$$= \frac{\int_\Theta w_\theta\ a_\theta(x)\ b_\theta(y)\ d\theta}{\int_\Lambda w_\lambda\ a_\lambda(x) \int_{\mathbb{R}}\ b_\lambda(y)\ dy\ d\lambda} \tag{145}$$

$$= \frac{\int_\Theta w_\theta\ V_\theta\ a_\theta(x)\ \left(\frac{b_\theta(y)}{V_\theta}\right)\ d\theta}{\int_\Lambda w_\lambda\ V_\lambda\ a_\lambda(x) \int_{\mathbb{R}}\ \frac{b_\lambda(y)}{V_\lambda}\ dy\ d\lambda} \tag{146}$$

$$= \frac{\int_\Theta w_\theta\ a_\theta(x)\ p_{B_\theta}(y)\ d\theta}{\int_\Lambda w_\lambda\ V_\lambda\ a_\lambda(x) \left(\int_{\mathbb{R}}\ p_{B_\lambda}(y)\ dy\right)\ d\lambda} \tag{147}$$

$$= \int_\Theta \left(\frac{w_\theta\ V_\theta\ a_\theta(x)}{\int_\Lambda w_\lambda\ V_\lambda\ a_\lambda(x)\ d\lambda}\right) p_{B_\theta}(y)\ d\theta \tag{148}$$

$$= \int_\Theta p_\theta(x)\ p_{B_\theta}(y)\ d\theta. \tag{149}$$

□

The same argument extends the Moment Theorem (52) to the case of a rule

continuum for the kth central moment $\mu^{(k)}$:

$$\mu^{(k)}(x) = \int_{\Theta} p_\theta(x) \sum_{l=0}^{k} \binom{k}{l} E_{B_\theta(x)}[(Y - c_\theta(x))^l]\ [c_\theta(x) - F(x)]^{k-l}\ d\theta \quad (150)$$

if we replace the generalized mixture density $p(y|x)$ in the expansion

$$\mu^{(k)} = \int_{\mathbb{R}} (y - F(x))^k\ p(y|x)\ dy \quad (151)$$

with its rule-continuum version in (140). A related result holds for non-central conditional moments.

Replacing the finite sums in the proof of Theorem 5 with finite integrals likewise gives the rule-continuum version of the mixture supervised learning law for the rule weights:

$$w_\theta(t+1) = w_\theta(t) + \mu_t \varepsilon_t \frac{p_\theta(x)}{w_\theta}[p_{B_\theta}(y) - p(y|x)] \quad (152)$$

where once again the continuum-many convex coefficients $p_\theta(x)$ obey the corresponding rule-continuum version of (63). The same argument similarly extends the supervised learning laws for the other parameters of the if-part and then-part fuzzy sets in the rule base.

The mixture learning law (152) raises the basic problem of working with rule continua: How can a practical system use such a learning law to tune the continuum-many rules of the fuzzy system?

The normalizing integral in the convex coefficient p_θ alone runs through the entire rule continuum $\mathcal{RB}(\theta)$ to compute the coefficient's value for just one input x. Such a system would have to approximate the integral and other terms as well as face the inherent curse of dimensionality that affects all such learning laws. The same problem holds for the ASAM laws that tune the parameters of the fuzzy system F itself. The SAM learning laws (130)–(133) and (136)–(137) all require computing the fuzzy-system output $F(x)$ for each x. This computation is trivial in the finite case but not so in the continuum case.

This raises the more immediate problem that we now address: How do we compute the output value $F(x)$ of a rule-continuum fuzzy system?

The fuzzy output $F(x)$ depends on two finite integrals over the rule continuum $\mathcal{RB}(\theta)$:

$$F(x) = \int_{\Theta} p_\theta(x)\ c_\theta\ d\theta \quad (153)$$

and

$$p(x) = \frac{a_\theta(x) w_\theta V_\theta}{\int_\Theta a_\theta(x) w_\theta \; V_\theta \; d\theta} \tag{154}$$

or as just the centroidal weighted average

$$F(x) = \frac{\int_\Theta a_\theta(x) w_\theta V_\theta c_\theta \; d\theta}{\int_\Theta a_\theta(x) w_\theta \; V_\theta \; d\theta}. \tag{155}$$

The ratio (155) shows that computing a single output $F(x)$ requires evaluating and combining as many fuzzy-rule parameters as there are real numbers. This complexity forces the use of approximation techniques in practice.

Monte Carlo simulation offers a practical way to compute the output expectation $E_{p_\theta(x)}[\Theta]$ for a given input x. This stochastic-estimation technique relies on the weak law of large numbers (WLLN). The WLLN states that the sample-mean random variable $\overline{X}_n = \frac{1}{n} \sum_{k=1}^n X_k$ of independent and identically distributed finite-variance random variables $X_1, X_2, \ldots$ converges in probability measure to the population mean $E[X]$: $\lim_{n \to \infty} P(|\overline{X}_n - E[X]| > \epsilon) = 0$ for all $\epsilon > 0$. Monte Carlo simulation interprets an ordinary definite integral $\int_a^b g(x) \, dx$ as the expectation of a random variable X that has a uniform distribution over (a, b)[30]:

$$\int_a^b g(x) \, dx = (b - a) \int_a^b g(x) \frac{dx}{b - a} \tag{156}$$

$$= (b - a) E[X] \tag{157}$$

for $X \sim \mathscr{U}(a, b)$. This famous trick of Von Neumann and Ulam[74] avoids integrating the integrand $g(x)$. It requires computing only values of $(b - a)g(x_k)$ for random uniform draws x_k from (a, b). The random draws can come from any uniform random number generator or from other pdfs in importance sampling. Then the WLLN ensures that

$$\frac{1}{n} \sum_{k=1}^n (b - a) g(x_k) \approx (b - a) E[X] \tag{158}$$

$$= \int_a^b g(x) \, dx \tag{159}$$

for enough random draws x_k. The variance in the WLLN estimate decreases linearly with the number n of draws. So the standard error in the estimate decreases fairly slowly with the inverse of the square root $\sqrt{n}$.[75]

Monte Carlo simulation can estimate the ratio of integrals involved in computing the fuzzy system output $F(x)$ in (153). Importance sampling replaces the expectation of $g(x)$ with respect to the pdf f with the expectation of $g(x) \frac{f(x)}{q(x)}$ with respect to some convenient positive pdf q. A more advanced technique is Markov

chain Monte Carlo or MCMC that works with correlated samples. MCMC estimates $F(x)$ by sampling from a reversible Markov chain whose equilibrium pdf has a first moment that corresponds to the deterministic integral in question.[75,76] Carefully injected noise can speed up MCMC convergence just as it can speed up convergence of the EM algorithm.[54]

We next present a related way to use Monte Carlo estimation to compute $F(x)$ in (153). This approach uses the additive fuzzy system's generalized mixture $p(y|x)$ and inserts it into the non-central version of the moment integral in (151). Then

$$F(x) = \int_{\mathbb{R}} y \; p(y|x) \; dy \tag{160}$$

$$= \int_{\mathbb{R}} y \int_{\Theta} p_\theta(x) \; p_{B_\theta}(y) \; d\theta \; dy \tag{161}$$

$$= \int_{\Theta} p_\theta(x) \left[\int_{\mathbb{R}} y \; p_{B_\theta}(y) \; dy \right] \; d\theta \tag{162}$$

$$= \int_{\Theta} p_\theta(x) \; c_\theta \; d\theta \tag{163}$$

$$= \int_{\Theta} \theta \; p_\theta(x) \; d\theta \tag{164}$$

$$= E_{p_\theta(x)}[\Theta] \tag{165}$$

from (140) and (153) if $\theta = c_\theta$ in the rule continuum. This lets us sidestep the continuum complexity by using Monte Carlo to estimate (160) if we have an appropriate closed form generalized mixture $p(y|x)$.

Suppose we want to find the Gaussian rule continuum $\mathcal{RB}(\theta)$ for a continuous additive fuzzy system F so that F approximates the function $f(x) = \sin x$. This can be a daunting task if we must pick or tune all the rules in the continuum. We already showed that a Watkins representation can exactly represent $\sin x$ with just two rules. But we now want to approximate it by drawing rules from the virtual rule continuum. Theorem 4 and (160)–(165) let us use ordinary Monte Carlo approximation of the integral (160) for a given input x if we have an appropriate generalized mixture density $p(y|x)$.

Consider again the target function $\sin x$. The bounded real function $\sin x$ has infimum -1 and supremum 1. So we can mix two Gaussian pdfs centered at -1 and 1 in (92) and the make simple change of variable $\theta = c_\theta$. This gives the exact mixture representation for $f(x) = \sin x$ as

$$p(y|x) = \frac{1 - \sin x}{2} p_{B_1}(y) + \frac{\sin x + 1}{2} p_{B_2}(y) \tag{166}$$

$$= \frac{1 - \sin x}{2} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{(y+1)^2}{2} \right]$$

$$+ \frac{\sin x + 1}{2} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{(y-1)^2}{2}\right] \tag{167}$$

for the two mixed unit-variance normal then-part pdfs $p_{B_1}(y) = \mathcal{N}(-1, 1)$ and $p_{B_2}(y) = \mathcal{N}(1, 1)$.

Figure 3 shows three Monte Carlo approximations of the continuous additive fuzzy system F that results from uniformly sampling from the $\sin x$-based mixture $p(y|x)$ in (167). The target function $\sin x$ has domain $[0, 2\pi]$. The simulation discretizes $[0, 2\pi]$ in increments of 0.01. Each simulation panel shows 629 values of x in $[0, 2\pi]$. The first panel shows the results of picking 100 values of y at random from the uniform distribution over $(-1, 1)$ for each value of x. Monte Carlo then takes the sample mean of these 100 sample values. The plot itself shows these 629 sample means. The second panel shows the same thing for 1,000 uniform random samples. The third panel shows the finer approximation that results for 10,000 uniform samples at each of the 629 values of x. The final panel shows the slow square-root decay of the average squared error. Each plotted value takes the average of 10 runs for a given number of random draws. So each plotted value is just the biased sample variance because it takes the average of 10 squared terms of the form $(F(x) - \sin x)^2$.

The Monte Carlo approach does not depend on the source of the mixture. A user can also create the controlling mixture $p(y|x)$ from the EM GMM estimation technique in (110)–(114) or by informed guesswork. The key point is that Monte Carlo techniques can estimate the output $F(x)$ because the output equals an expectation.

The EM-GMM approach found only a crude approximation of $\sin x$ compared with the Monte Carlo approximation in Figure 3. Simulations showed that the EM estimates often suffered from overfitting with too many mixed Gaussian pdfs. The best results used only 6 mixed Gaussian pdfs and thus used a number of mixed pdfs that roughly corresponded to the main turning points or extrema of the approximand $f(x) = \sin x$. This also roughly resembled the optimal rule placement strategy that "patches the bumps" or covers the extrema of the approximand $f(x)$ with finite rule patches. But the EM-GMM approach used only blind samples from $\sin x$. The approach did not require prior knowledge of the closed form of the approximand $\sin x$ itself. This cruder mixture approximation scheme may be sufficient for applications when the user has some sample data from the approximand $f(x)$ but does not know the form of the approximand $f(x)$. It may even be sufficient when the user does not know its general contours or the approximate location of its extrema and inflection points.

Informed guesswork guides much of knowledge engineering. It may be the only technique available when there is little or no representative sample data. Fuzzy engineers often center input and output fuzzy sets where they want specific types of control or system performance. The same technique applies to centering the mixed pdfs of the mixture density $p(y|x)$. The engineer might center the mixed pdfs closer together in regions of the input space where he desires more precise control. The fuzzy truck-backer-upper[77] was an early example of such proximity control using thin and wide fuzzy sets. It placed a small number of wide or narrow fuzzy rule patches over the presumed system input-output function. The truck-and-trailer

system backed up to a loading dock in a parking lot. Closer and narrower if-part sets near the loading dock gave finer control in the approach to the loading dock. Only a few wide if-part sets covered the rest of the parking lot. These then-part sets implicitly defined a generalized mixture density $p(y|x)$ upon normalization by the corresponding then-part set areas.

Casting rules in terms of mixed pdfs can mitigate rule explosion. The mixed densities $p_{B_j}(y|x)$ act as meta-rules for the fuzzy system. But building the mixed densities coordinate-wise with fuzzy sets results in the same rule explosion in the input dimension n as results with building an ordinary rule base of if-then rules by quantizing each of the n axes into two or more fuzzy sets or linguistic variables.

The fuzzy engineer can instead program the fuzzy system directly by picking or estimating the generalized mixture $p(y|x)$ to achieve some desired control effect or local approximation as with the earlier truck backer-upper. The number m of mixed densities need not grow exponentially with n. The number of mixed densities need only grow as some polynomial of n. The number of mixed densities may grow only linearly with n in practice in the GMM case of mixing multidimensional Gaussian pdfs and then using EM to tune them.

The main complexity comes from the Monte Carlo approximation involved in computing the outputs $F(x)$. A sufficiently fine approximation may involve using so many sampled rules from the rule continuum that the result is close to an exponential rule explosion. Coarser approximations of $F(x)$ should use fewer such rules. Clever types of importance sampling should use fewer rules still. Whether this tradeoff between the number m of mixed pdfs and the number of sampled rules from the rule continuum is favorable will also depend both on the quality of the governing mixture density $p(y|x)$ and on the efficiency of the virtual-rule sampling scheme.

References

1. Kosko B. Neural networks and fuzzy systems. Prentice-Hall; 1991.
2. Kosko B. Fuzzy systems as universal approximators. IEEE T Comput 1994;43(11):1329–1333.
3. Kosko B. Optimal fuzzy rules cover extrema. Int J Intell Syst 1995;10(2):249–255.
4. Dickerson JA, Kosko B. Fuzzy function approximation with ellipsoidal rules. IEEE T Syst Man Cy B 1996;26(4):542–560.
5. Kosko B. Fuzzy engineering. Prentice-Hall; 1996.
6. Kosko B. Global stability of generalized additive fuzzy systems. IEEE T Syst Man Cy C 1998;28(3):441–452.
7. Kosko B, Isaka S. Fuzzy logic. Sci Am 1993;269(11):62–67.
8. Mitaim S, Kosko B. The shape of fuzzy sets in adaptive function approximation. IEEE T Fuzzy Syst 2001;9(1):637–656.
9. Lee I, Kosko B, Anderson WF. Modeling gunshot bruises in soft body armor with an adaptive fuzzy system. IEEE T Syst Man Cy B 2005;35(4):1374–1390.
10. Zadeh LA. Fuzzy sets. Inform Control 1965;8(3):338–353.
11. Zadeh LA. Outline of a new approach to the analysis of complex systems and decision analysis. IEEE T Syst Man Cy 1973;3(1):28–44.
12. Zadeh LA. The concept of a linguistic variable and its application to approximate reasoning. Inform Sciences 1975;8:199–249.
13. Osoba O, Mitaim S, Kosko B. Bayesian inference with adaptive fuzzy priors and likelihoods. IEEE T Syst Man Cy B 2011;41(5):1183–1197.

14. Osoba O, Mitaim S, Kosko B. Triply fuzzy function approximation for hierarchical bayesian inference. Fuzzy Optimization Decision Making 2012;11(3):241–268.
15. Jang J-S, Sun C-T. Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE T Neural Networ 1993;4(1):156–159.
16. Kosko B. Fuzzy associative memories. In: Fuzzy Expert Syst. CRC Press; 1991.
17. Lakkaraju H, Bach SH, Leskovec J. Interpretable decision sets: A joint framework for description and prediction. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM; 2016. pp 1675–1684.
18. Sugeno M. An introductory survey of fuzzy control. Inform Sciences 1985;36(1):59–83.
19. Kahraman C, Gülbay M, Kabak Ö. Applications of fuzzy sets in industrial engineering: a topical classification. In: Fuzzy applications in industrial engineering. Springer; 2006. pp 1–55.
20. Feng G. A survey on analysis and design of model-based fuzzy control systems. Fuzzy Syst, IEEE T 2006;14(5):676–697.
21. Liao S-H. Expert system methodologies and applications: a decade review from 1995 to 2004. Expert Syst Appl 2005;28(1):93–103.
22. Kong S-G, Kosko B. Adaptive fuzzy systems for backing up a truck-and-trailer. Neural Networ, IEEE T 1992;3(2):211–223.
23. Yager RR, Filev DP. Approximate clustering via the mountain method. Syst, Man and Cy, IEEE T 1994;24(8):1279–1284.
24. Watkins FA. Fuzzy engineering. Ph.D. dissertation, University of California at Irvine; 1994.
25. Watkins F. The representation problem for additive fuzzy systems. In: *Proceedings of the International Conference on Fuzzy Systems (IEEE FUZZ-95)*; 1995. pp 117–122.
26. Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. IEEE T Syst Man Cy 1985;1:116–132.
27. Sugeno M, Kang G. Structure identification of fuzzy model. Fuzzy Set Syst 1988;28(1):15–33.
28. Redner RA, Walker HF. Mixture densities, maximum likelihood and the EM algorithm. SIAM Review 1984;26(2):195–239.
29. McLachlan GJ, Peel D. Finite Mixture Models. ser. Wiley series in probability and statistics: Applied probability and statistics. Wiley; 2004.
30. Hogg RV, McKean J, Craig AT. Introduction to mathematical statistics. Pearson; 2013.
31. Xu L, Jordan MI. On convergence properties of the em algorithm for gaussian mixtures. Neural Computat 1996;8(1):129–151.
32. Moon TK. The expectation-maximization algorithm. IEEE Signal Proc Mag 1996; 13(6):47–60.
33. Russell SJ, Norvig P. Artificial intelligence: A modern approach. Prentice Hall; 2010.
34. Osoba O, Mitaim S, Kosko B. The noisy expectation–maximization algorithm. Fluctuat Noise Lett 2013;12(3):1 350 012-1–1 350 012-30.
35. Audhkhasi K, Osoba O, Kosko B. "Noise benefits in backpropagation and deep bidirectional pre-training. In: *Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN-2013)*, IEEE; 2013. pp 2254–2261.
36. Osoba O, Kosko B. The noisy expectation-maximization algorithm for multiplicative noise injection. Fluctuat Noise Lett 2016;1650007.
37. Audhkhasi K, Osoba O, Kosko B. Noise-enhanced convolutional neural networks. Neural Networ 2016;78:15–23.
38. Bandler W, Kohout L. Fuzzy power sets and fuzzy implication operators. Fuzzy Set Syst 1980;4(1):13–30.
39. Goguen JA. The logic of inexact concepts. Synthese 1969;19(3):325–373.
40. Gaines BR. Foundations of fuzzy reasoning. Int J Man-Mach Stud 1976;8:623–688.
41. Dubois D, Prade H. Fuzzy sets in approximate reasoning, part 1: Inference with possibility distributions. Fuzzy Set Syst 1999;100:73–132.
42. Kandel A. Fuzzy mathematical techniques with applications. Addison-Wesley; 1986.
43. Klir GJ, Folger TA. Fuzzy sets, uncertainty, and information. Prentice Hall; 1988.

44. Zimmermann H-J. Fuzzy set theory and its applications. Springer Science & Business Media; 2011.
45. Yager RR. On ordered weighted averging aggregation operators in multicriteria decision making. IEEE T Syst Man Cy 1988;18(1):183–190.
46. Klement EP, Mesiar R, Pap E. "Triangular norms. position paper i: basic analytical and algebraic properties. Fuzzy Set Syst 2004;143(1):5–26.
47. Terano T, Asai K, Sugeno M. Fuzzy systems theory and its applications. Academic Press Professional, Inc.; 1992.
48. Anderson JA. A memory storage model utilizing spatial correlation functions. Kybernetik 1968;5(3):113–119.
49. Kohonen T. Correlation matrix memories. Comput, IEEE T 1972;100(4):353–359.
50. Anderson JA, Silverstein JW, Ritz SA, Jones RS. Distinctive features, categorical perception, and probability learning: Some applications of a neural model. Psychol Rev 1977;84(5):413.
51. Kosko B. Bidirectional associative memories. Syst Man Cy IEEE T 1988;18(1):49–60.
52. Hopfield JJ. Neural networks and physical systems with emergent collective computational abilities. P Natl Acad Sci 1982;79(8):2554–2558.
53. Brooks S, Gelman A, Jones G, Meng X-L. Handbook of Markov Chain Monte Carlo. CRC Press; 2011.
54. Franzke B, Kosko B. Using noise to speed up markov chain monte carlo estimation. Procedia Comput Sci 2015;53:113–120.
55. Kosko B. Fuzzy knowledge combination. Int J Intell Syst 1986;1(4):293–320.
56. Kreinovich V, Mouzouris GC, Nguyen HT. Fuzzy rule based modeling as a universal approximation tool. In: Fuzzy systems. Springer; 1998. pp 135–195.
57. Munkres J. Topology, 2nd ed. Prentice Hall, Inc; 2000.
58. Kosko B. Fuzzy knowledge combination. Int J Intell Syst 1986;1(4):293–320.
59. Kosko B. Fuzzy cognitive maps. Int J Man-Mach Stud 1986;24(1):65–75.
60. Glykas M. editor. Fuzzy cognitive maps. Springer; 2010.
61. Papageorgiou E. Fuzzy Cognitive Maps for Applied Sciences and Engineering: From Fundamentals to Extensions and Learning Algorithms. ser. Intelligent Systems Reference Library. Springer Berlin Heidelberg; 2013. [Online]. Available: https://books.google.com/books?id=S3LGBAAAQBAJ
62. Osoba O, Kosko B. Fuzzy cognitive maps of public support for insurgency and terrorism. J Defense Model Simulat 2017;14(1):17–32.
63. Rogova G. Combining the results of several neural network classifiers. Neural Networ 1994;7(5):777–781.
64. Kittler J, Hatef M, Duin RP, Matas J. On combining classifiers. IEEE T Pattern Anal 1998;20(3):226–239.
65. Tumer K, Ghosh J. Analysis of decision boundaries in linearly combined neural classifiers. Pattern Recogn 1996;29(2):341–348.
66. Jordan MI, Jacobs RA. Hierarchical mixtures of experts and the em algorithm. Neural Comput 1994;6(2):181–214.
67. Bishop CM. Pattern recognition and machine learning. Springer; 2006.
68. Masoudnia S, Ebrahimpour R. Mixture of experts: a literature survey. Artif Intell Rev 2014;42(2):275–293.
69. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the em algorithm. J Roy Stat Soc B Met 1977;1–38.
70. Osoba O, Kosko B. Noise-enhanced clustering and competitive learning algorithms. Neural Networ 2013;37:132–140.
71. Osoba O, Mitaim S, Kosko B. The noisy expectation–maximization algorithm. Fluctuat Noise Lett 2013;12.
72. Duda RO, Hart PE, Stork DG. Pattern classification. John Wiley & Sons; 2012.
73. Kim HM, Kosko B. Fuzzy prediction and filtering in impulsive noise. Fuzzy Set Syst 1996;77(1):15–33.
74. von Neumann J, Ulam S. Monte Carlo method. National Bureau of Standards Applied Mathematics Series 1951;12:36.

75. Brooks S, Gelman A, Jones G, Meng X-L. Handbook of Markov Chain Monte Carlo. CRC Press; 2011.
76. Robert CP, Casella G. Monte Carlo statistical methods (Springer texts in statistics), 2nd ed. Springer-Verlag; 2005.
77. Kong S-G, Kosko B. Adaptive fuzzy systems for backing up a truck-and-trailer. IEEE T Neural Networ 1992;3(2):211–223.