Fuzzy Function Approximation with Ellipsoidal Rules

Julie A. Dickerson and Bart Kosko

Abstract—**A fuzzy rule can have the shape of an ellipsoid in the input-output state space of a system. Then an additive fuzzy system approximates a function by covering its graph with ellipsoidal rule patches. It averages rule patches that overlap. The best fuzzy rules cover the extrema or bumps in the function. Neural or statistical clustering systems can approximate the unknown fuzzy rules from training data. Neural systems can then both tune these rules and add rules to improve the function approximation. We use a hybrid neural system that combines unsupervised and supervised learning to find and tune the rules in the form of ellipsoids. Unsupervised competitive learning finds the first-order and second-order statistics of clusters in the training data. The covariance matrix of each cluster gives an ellipsoid centered at the vector or centroid of the data cluster. The supervised neural system learns with gradient descent. It locally minimizes the mean-squared error of the fuzzy function approximation. In the hybrid system unsupervised learning initializes the gradient descent. The hybrid system tends to give a more accurate function approximation than does the lone unsupervised or supervised system. We found a closed-form model for the optimal rules when only the centroids of the ellipsoids change. We used numerical techniques to find the optimal rules in the general case.**

I. Learning with Ellipsoidal Rules

A fuzzy system is a set of fuzzy rules that maps inputs to outputs. So it defines a function $f: X \rightarrow Y$. The rules are if-then rules of the form "If X is A, then Y is B." A and B are multivalued or fuzzy sets that contain members to some degree. A is a subset of input space X. B is a subset of output space Y.

Fuzzy rules have a simple geometry in the input-output state space $X \times Y$. They define fuzzy patches or subsets of the state space. Less certain rules are large patches. More precise rules are small patches. The rule "If X is A, then Y is B" defines the fuzzy patch or Cartesian product $A \times B$. In the precise limit the rule patch $A \times B$ is a point if A and B are binary spikes.

An additive fuzzy system adds the output or then-part sets B'_j of the fuzzy rules as in Fig. 3. It covers the graph of the function and averages patches that overlap. These systems can approximate a continuous function to any degree of accuracy with a finite number of fuzzy rules. The rules and their sets can have any shape. In practice the sets have simple shapes like triangles, trapezoids, or bell curves.

Manuscript received November 19, 1993; revised January 15, 1995, and July 31, 1995. This work was supported by the Caltrans PATH Program Agreement 20695MB).

J. A. Dickerson is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA.

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

Publisher Item Identifier S 1083-4419(96)03926-X.

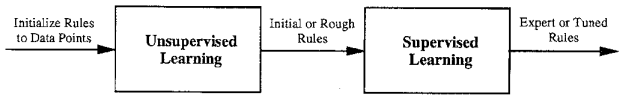

Fig. 1. The hybrid neural system combines unsupervised and supervised neural learning to find and tune the ellipsoidal fuzzy rules.

We propose rules that have the shape of ellipsoids. Ellipsoids arise from the covariance matrices of statistical or neural estimators. And they lead to simple algorithms to tune the rules with training data.

A fuzzy system learns or adapts when its rules change. Then the system function $f: X \rightarrow Y$ varies with time. A rule changes when its fuzzy sets change. Neural networks can change the sets with training data or error reinforcement signals.

We used two types of neural nets to change the ellipsoidal rules in an additive fuzzy system. The first neural system used unsupervised learning to find the covariance matrices of synaptic quantization vectors. A quantization vector "hops" more in regions of sparse or noisy data. Then it has a larger covariance ellipsoid and thus gives a less certain rule. The synaptic vector hops less in regions of dense or less noisy data and gives a more precise rule. The second neural system used supervised learning to tune the rules with stochastic gradient descent. Supervised learning needs an error signal. The neural net must know the function it tries to approximate or at least know some input-output samples from it. Supervised learning changes the centers and eigenvalues of the ellipsoids. This moves, shapes, and orients the ellipsoids in the state space.

The supervised learning takes far more computation than does the unsupervised learning in this algorithm. But it gives rules that better approximate the function. At best the supervised gradient descent finds the local minima of the error surface. How well it learns depends on how well we pick the initial set of fuzzy rules.

We combined the two types of learning in a hybrid neural system that both finds and tunes the ellipsoidal rules as in Fig. 1. Unsupervised learning picks the first set of rules based on the second-order statistics of the training data. Then supervised learning tunes these rule ellipsoids. We tested all three systems on a fifth-order polynomial. The hybrid system learned faster and had less mean-squared error than did the lone unsupervised or supervised system.

Optimal ellipsoidal rules minimize the mean-squared error of the fuzzy function approximation. We found a closed-form model of optimal ellipsoid rules when just the rule centroids

change. We used the numerical Nelder-Meade algorithm [2], [25] to find the optimal rules when more rule parameters change. In simulations supervised and hybrid learning tended to converge to or near the optimal ellipsoidal rules. Optimal rules help deal with the exponential "rule explosion" as the joint dimension $n+p$ of the product space $X \times Y$ grows. Lone optimal rule patches cover the extrema of the approximand [16].

II. Additive Fuzzy Systems

A fuzzy system approximates a function by covering its graph with fuzzy patches and averaging patches that overlap. The approximation improves as the fuzzy patches grow in number and shrink in size. The approximation improves as we add more small patches but storage and complexity costs increase. The additive fuzzy systems have a feedforward architecture that resembles the feedforward multilayer neural systems used to approximate functions [11].

An additive fuzzy system adds the then-par s of fired if- then rules. Other fuzzy systems combine the then-part sets with pairwise maxima. An additive fuzzy system can uniformly approximate continuous [14] or measurable [15] functions. A fuzzy system has rules of the form "If input conditions hold, then output conditions hold" or "If X is A, then Y is B" for fuzzy sets A and B. Each fuzzy rule defines a fuzzy patch or a Cartesian product $A \times B$ as shown in Fig. 2. The fuzzy system covers the graph of a function with fuzzy patches and averages patches that overlap. Uncertain fuzzy sets give a large patch or fuzzy rule. Small or more certain fuzzy sets give small patches.

Additive fuzzy systems fire all rules in parallel and average the scaled output sets B_j' to get the output fuzzy set B as in Fig. 3. Correlation product inference scales each output set B_j by the degree $a_j(x)$ that the rule "IF A_j, THEN B_j" fires. Most rules fire to degree 0. Defuzzification of B gives a number or a control signal output. Centroidal defuzzification with correlation product inference [14], [20] gives the output value y or $F(x)$ given input vector $x \in R^n$:

$$y = F(x) = \text{Centroid}(B) = \frac{\int y b(y)\, dy}{\int b(y)\, dy}$$

$$= \frac{\sum_{j=1}^{m} w_j \text{ Volume}(B_j') \text{ Centroid}(B_j')}{\sum_{j=1}^{m} w_j \text{ Volume}(B_j')}$$

$$= \frac{\sum_{j=1}^{m} w_j V_j a_j(x) c_{y_j}}{\sum_{j=1}^{m} w_j V_j a_j(x)}. \tag{2-1}$$

V_j is the volume of the jth output set B_j. w_j is the weight of the jth rule. To change the rule weight w_j we can change w_j itself or change the volume V_j if the weight depends on the volume Section III of this paper looks at how different

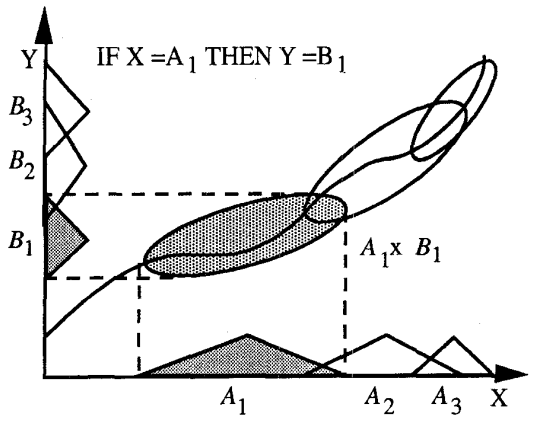

Fig. 2. The fuzzy rule patch "If X is fuzzy set A_1, then Y is fuzzy set B_1" is the fuzzy Cartesian product $A_1 \times B_1$ in the input-output product space $X \times Y$.

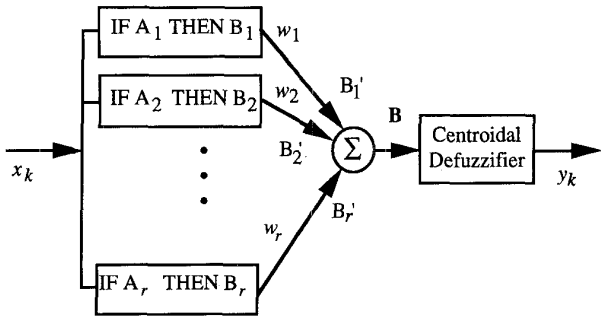

Fig. 3. Additive fuzzy system architecture. The input x_k acts as a delta pulse (or unit bit vector) and fires each rule to some degree. The system adds the scaled output fuzzy sets. The centroid of this combined set gives the output value y_k. The system computes the conditional expectation value $E[Y|X = x_k]$ as a convex sum of the local centroids or centers of the then-part sets B_j'.

rule weights affect the output. We can always normalize the finite volumes V_j to unity to keep some rules from dominating others. c_{y_j} is the centroid of the jth output set. Fit value $a_j(x)$ scales the output set B_j. m is the number of output fuzzy sets. In practice B is connected. It need not be. But then we could view the rule "If X is A, then Y is B" as two or more rules of the form "If X is A, then Y is B_1" and "If X is A, then Y is B_2" where B_1 and B_2 are two of the disjoint components of B. So assume B is connected. Then the rule patch $A \times B$ is connected and a patch proper.

We now formally derive the *standard* additive model in (2-1) that we shall use in this paper. A general additive fuzzy system is a map F: $R^n \rightarrow R^p$. Both in practice and in uniform approximation proofs we restrict the domain to a compact subset $U \subset R^n$ but we need not. Watkins [27], [28] has proved that a scalar additive system F: $R \rightarrow R$ with just two rules can exactly *represent* any bounded function f: $R \rightarrow R$ in the sense that $F(x) = f(x)$ for all x even if f is not continuous. In this case the domain is the entire real line. Here $a_1(x) = (\sup f - f(x))/(\sup f - \inf f)$ and $a_2(x) = 1 - a_1(x)$. B_1 and B_2 are rectangles or other sets with the same area centered at $\inf f$ and $\sup f$.

The additive fuzzy system F: $R^n \rightarrow R^p$ stores m fuzzy patches $A_j \times B_j$ or rules of the form "*If X is A_j, then Y is B_j.*" Here $A_j \subset R^n$ and $B_j \subset R^p$ multivalued or "fuzzy" sets with set functions a_j: $R^n \rightarrow [0,1]$ and b_j: $R^p \rightarrow [0,1]$.

In practice we define the if-part set A_j by its n coordinate-projection sets $A_j^1, \cdots, A_j^n$ and thus $A_j = A_j^1 \times A_j^2 \times \cdots \times A_j^n$. How we define this fuzzy Cartesian product dictates the conjunctive (or t-norm) form of how we factor the joint set function a_j into its coordinate set functions $a_j^1, \cdots, a_j^n$. Minimum combination is the most popular form

$$a_j(x) = a_j^1(x_1) \wedge a_j^2(x_2) \wedge \cdots \wedge a_j^n(x_n) \tag{2-2}$$

for input vector $x = (x_1, \cdots, x_n)$. Product combination

$$a_j(x) = \prod_{i=1}^{n} a_j^i(x_i) \tag{2-3}$$

can simplify the analysis and computation of additive systems with Gaussian [8], [22] or radial-basis [26] set functions of the form

$$a_j^i(x_i) = s_i^j \exp\left[-\frac{1}{2}\left(\frac{x_i - \overline{x}_i^j}{\sigma_i^j}\right)^2\right] \tag{2-4}$$

for scaling constant $0 < s_i^j \leq 1$. The choice of combination operator does not affect the structure of the standard model (2-1). We use min combination below for ellipsoidal rules.

The next step is the additive step. The m fit values $a_j^i(x_i)$ "fire" the then-part sets B_j to give the "inferred" sets B_j'. Again the result combines $a_j^i(x_i)$ and B_j in some conjunctive (t-norm) way and again it depends on how we define the Cartesian patch $A_j \times B_j$. Here min is less popular than product. The min "clip" discards all information in B_j above the fit height $a_j^i(x_i)$ and can thus change the centroid of B_j if B_j is not symmetric. Product combination or *correlation product decoding* [14] keeps all relative information in B_j and does not change its centroid:

$$B_j' = a_j^i(x) B_j. \tag{2-5}$$

We use (2-5) as a default. An additive model [14] then sums these inferred sets to produce the final output set B

$$B = \sum_{j=1}^{m} w_j B_j'. \tag{2-6}$$

Each rule can have a weight w_j that scales B_j' in (2-6). Learning can change these weights or we can use them to model frequency or "usuality" rule weights.

The only constraint on B or b is that it have a finite integral or *volume*

$$0 < V = \int b(y)\, dy < \infty. \tag{2-7}$$

This means that each input x fires at least one rule to non zero degree.

We now assume that the additive fuzzy system maps real vectors into scalars or F; $U \subset R^n \rightarrow R$. Then use the additive assumption (2-6) with a centroidal output to get the standard form of a additive model [14] we use in this chapter

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

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^{m} w_j b_j'(y)\, dy}{\int_{-\infty}^{\infty} \sum_{j=1}^{m} w_j b_j'(y)\, dy} \tag{2-8}$$

$$= \frac{\sum_{j=1}^{m} \int_{-\infty}^{\infty} y\, a_j(x) w_j b_j(y)\, dy}{\sum_{j=1}^{m} \int_{-\infty}^{\infty} a_j(x) w_j b_j(y)\, dy} \tag{2-9}$$

$$= \frac{\sum_{j=1}^{m} a_j(x) w_j V_j \frac{\int_{-\infty}^{\infty} y\, b_j(y)\, dy}{V_j}}{\sum_{j=1}^{m} a_j(x) w_j V_j} \tag{2-10}$$

$$= \frac{\sum_{j=1}^{m} a_j(x) w_j V_j c_j}{\sum_{j=1}^{m} a_j(x) w_j V_j} \tag{2-11}$$

for then-part set volumes

$$V_j = \int_{-\infty}^{\infty} b_j(y)\, dy \tag{2-12}$$

and then-part set centroids

$$c_j = \frac{\int_{-\infty}^{\infty} y\, b_j(y)\, dy}{\int_{-\infty}^{\infty} b_j(y)\, dy}. \tag{2-13}$$

The model in (2-11) is the standard additive fuzzy system and the same as (2-1). Note that (2-8) implies that $F(x) = E]Y|X = x]$ since

$$p(y|x) = \frac{b(x,y)}{\int_{-\infty}^{\infty} b(x,y)\, dy}$$

is a conditional probability density function even if $b_j > 1$ holds. Then (2-11) gives the global conditional mean $F(x)$ as a convex sum of the local conditional means c_j. Note also that the standard additive model (2-1) gives $F(x)$ as a convex sum of local centroids (conditional means)

$$F(x) = \sum_{j} p_j(x) c_j \tag{2-14}$$

for convex coefficients

$$p_j(x) = \frac{a_j(x) w_j V_j}{\sum_{i=1}^{m} a_i(x) w_i V_i}. \tag{2-15}$$

The standard model (2-11) reduces to the Gaussian additive model of Wang and Mendel [26]

$$F(x) = \frac{\sum_{j=1}^{m} \overline{z}^j \left(\prod_{i=1}^{n} \mu_{A_i^j}(x_i) \right)}{\sum_{j=1}^{m} \left(\prod_{i=1}^{n} \mu_{A_i^j}(x_i) \right)} \tag{2-16}$$

for the Gaussian if-part set in (2-4) and Gaussian then-part sets with these identifications

$$y = z \tag{2-17}$$

$$a_j(x) = \prod_{i=1}^{n} a_j^i(x_i) \tag{2-18}$$

$$= \prod_{i=1}^{n} \mu_{A_i^j}(x_i) \tag{2-19}$$

$$V_j = 1 \tag{2-20}$$

$$c_j = \overline{z}^j. \tag{2-21}$$

The choice of product combination (2-2) gives (2-18) and (2-19). The unity volume follows in (2-20) since Wang and Mendel integrate their m then-part Gaussian sets over all of R (and thus use the scaling constant in (2-4) to account for the input truncation to a compact set). (2-21) follows because the mode of a Gaussian set equals its centroid and Wang and Mendel use the mode definition: "the point in R at which $\mu_{B_j}(z)$ achieves its maximum value." They used the Stone-Weierstrass Theorem to prove that additive Gaussian systems with all-product combination in (2-16) are uniform approximators of continuous maps on compact sets This nonconstructive result is a special case of the uniform approximation theorem for all additive systems See [15] for the general theorem and its constructive proof. It holds as well for Gaussian sets with min combination (2-1) of if-part fit values or min clipping of then-part sets B_j. Specht [22] calls his Gaussian model (2-16) a "general regression neural network" and thus shows that a Gaussian Parzen density estimator coincides with a Gaussian standard additive fuzzy system. This still holds of $w_j = 1/(\sigma_i j)^2$

III. Ellipsoidal Fuzzy Rules

A. Ellipsoids as Fuzzy Patches

A fuzzy rule patch can take the form of an ellipsoid [5]. The covariance of the pattern classes in the data can define ellipsoidal patches. The size and shape of the ellipsoid shows how the inputs and outputs relate to each other in some region of the state space. We project the ellipsoid onto the input and output axes to form the fuzzy sets.

The eigenvectors and eigenvalues of a positive-definite matrix $\boldsymbol{A}$ define an ellipsoid in the q-dimensional input-output state space [23]. Here $q = n + p$, n is the number of inputs to the fuzzy system, and p is the number of outputs. The ellipsoid is the locus of all $\boldsymbol{z}$ that satisfy

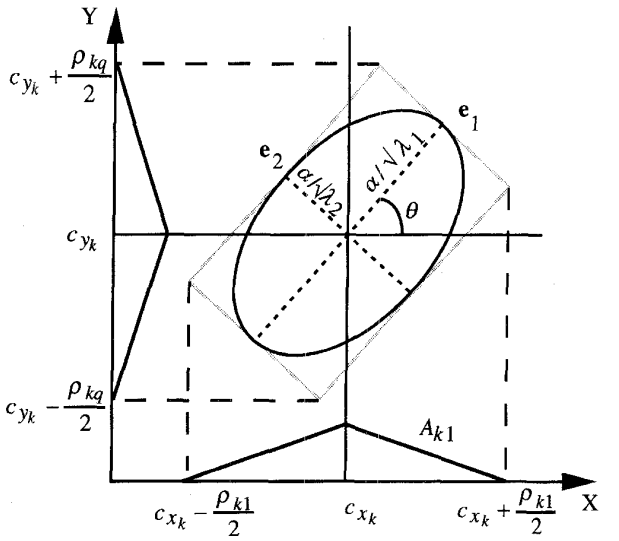

Fig. 4. A positive definite matrix $\boldsymbol{A}$ defines an ellipsoid around the center $\boldsymbol{c}$ of the ellipsoid. The eigenvectors of $\boldsymbol{A}$ define the axes. The eigenvalues define the length of the axes. The ellipsoid projects onto the axes to define the input and output fuzzy sets.

$$\begin{aligned} \alpha^2 &= (\boldsymbol{z} - \boldsymbol{c})^T \boldsymbol{A} (\boldsymbol{z} - \boldsymbol{c}) \\ &= (\boldsymbol{z} - \boldsymbol{c})^T \boldsymbol{P} \Lambda \boldsymbol{P}^T (\boldsymbol{z} - \boldsymbol{c}) \end{aligned} \tag{3-1}$$

where α is a positive real number and $\boldsymbol{c}$ is the center of the ellipsoid. Λ is a diagonal matrix of the eigenvalues $\lambda_1, \cdots, \lambda_q$ of $\boldsymbol{A}$. $\boldsymbol{P}$ is an orthogonal matrix whose columns are the unit eigenvectors $\boldsymbol{e}_1, \cdots, \boldsymbol{e}_q$ of $\boldsymbol{A}$. $\boldsymbol{P}$ rotates the coordinate system to the eigenvectors to orient the ellipsoid. The Euclidean half lengths of the axes equal $\alpha/\sqrt{\lambda_1}, \cdots, \alpha/\sqrt{\lambda_q}$. Fig. 4 shows the geometry of the ellipsoid in two-space. If $\boldsymbol{A}$ is not positive-definite the ellipsoid can concentrate on a lower dimensional hyperplane [23]. We assume that $\boldsymbol{A}$ is positive definite.

To simplify the math we inscribe the ellipsoids in hyperrectangles. Then we project the hyperrectangles onto the axes of the state space to form the fuzzy sets. The kth hyperrectangle has 2^q vertices at $(\pm\alpha_k/\sqrt{\lambda_{k1}}, \cdots, \pm\alpha_k/\sqrt{\lambda_{kq}})$ in the rotated coordinate plane. The unit eigenvectors define direction cosines for each axis of the ellipse. The *direction cosine* [10] $\cos\ \gamma_{kij}$ is the angle between the jth eigenvector and the ith axis for the kth ellipsoid. The projection of the kth hyperrectangle onto the ith axis is centered at c_{ki} on the ith axis and has length ρ_{ki}

$$\rho_{ki} = 2\alpha_k \sum_{j=1}^{q} \frac{|\cos \gamma_{kij}|}{\sqrt{\lambda_{kj}}}. \tag{3-2}$$

Unimodal sets can approximate the ellipsoid projections onto each axis. We use symmetric triangular sets. Fig. 5 shows the ellipsoid patches and their triangular projections for a single-input single-output function. The ellipsoid parameters fix the position, shape, and size of the fuzzy rules. ρ_{ki} defines the base of the triangular fuzzy sets on the ith axis for the kth

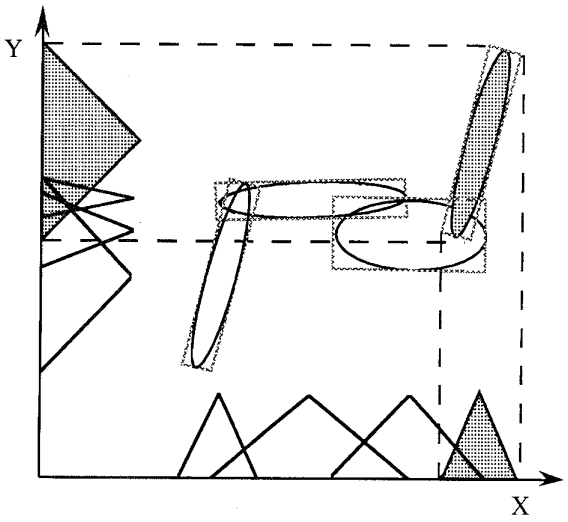

Fig. 5. The projection of each ellipsoid on o the axes of the input- output state space defines a fuzzy set. The ellipsoid defines a fuzzy patch or rule between fuzzy subsets of inputs and outputs.

ellipsoidal rule. The volume of the triangular output set V_k for the kth ellipsoidal rule is

$$V_k = \tfrac{1}{2} \cdot 1 \cdot \rho_{kq}. \tag{3-3}$$

The fit-value degree $a_k(x)$ is

$$a_k(x) = \min_j (a_k^j(x)) \tag{3-4}$$

for multiple inputs in the if-part conjoined with AND. This gives a fuzzy rule of the form "IF x_1 is a_k^1 AND x_2 is a_k^2 AND $\cdots x_n$ is a_k^n, THEN y is B_k." a_k^j is the triangular input set for the kth ellipsoid's hyperrectangle projected on the jth axis

$$a_k^j(x) = \begin{cases} 1 - \dfrac{|x - c_{x_{kj}}| \cdot 2}{\rho_{kj}} & \text{for } |x - c_{x_{kj}}| \le \dfrac{\rho_{kj}}{2} \\ 0 & \text{else.} \end{cases} \tag{3-5}$$

The orientation of the eigenvectors fixes the size of projections. The eigenvectors are *orthonormal* for a symmetric positive-definite matrix [23]. So the direction cosines for the kth ellipsoid obey

$$\begin{aligned} &(\cos\gamma_{ki1})^2 + \cdots + (\cos\gamma_{kiq})^2 = 1 \\ &\cos\gamma_{ki1}\cos\gamma_{kj1} + \cdots + \cos\gamma_{kiq}\cos\gamma_{kjq} \\ &\quad = 0 \quad \text{for all} \quad i, j = 1, \cdots, q, i \ne j. \end{aligned} \tag{3-6}$$

This gives q^2 unknowns and $q + (q-1)!$ equations. $q^2 - q - (q-1)!$ independent variables define the ellipsoid's orientation. For $q = 2$ just one number orients the ellipsoid. For a 2-D ellipsoid the rotation matrix is

$$\boldsymbol{P} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{3-7}$$

Then the hyperrectangle projections are

$$\begin{aligned} \rho_{k1} &= 2\alpha_k \left(\frac{|\cos\theta|}{\sqrt{\lambda_{k1}}} + \frac{|\sin\theta|}{\sqrt{\lambda_{k2}}} \right) \\ \rho_{k2} &= 2\alpha_k \left(\frac{|\sin\theta|}{\sqrt{\lambda_{k1}}} + \frac{|\cos\theta|}{\sqrt{\lambda_{k2}}} \right). \end{aligned} \tag{3-8}$$

The projected sets form a hyperrectangle in the input output state space. Fig. 6(a)–(b) shows how different ellipsoid patches can give the same fuzzy sets and rules. The projections do not directly use all of the information available in the ellipsoidal fuzzy patch such as ellipsoid volume and orientation. Performance improvements may come from using the ellipsoidal patch directly at the expense of more complicated mathematics.

B. Weighting Ellipsoidal Rules

The fuzzy additive system F in (2-1) computes the global conditional mean $F(x) = E[Y|X = x]$ with a convex sum of the conditional means for each rule (2-11). The jth rule tries to make $F(x)$ look like c_{y_j} when it fires. $F(x)$ tends to c_{y_j} when the then-part volume V_j grows with respect to the other then-part volumes

$$\lim_{V_j \to \infty} F(x) = \lim_{V_j \to \infty} \frac{\sum_{i=1}^{m} a_i(x) w_i V_i c_{y_i}}{\sum_{i=1}^{m} a_i(x) w_i V_i} \to c_{y_j} \quad \text{if } a_j(x) > 0. \tag{3-9}$$

The rule weights w_j weight the then-part centroids to approximate the function. To change the rule weights w_j we can change w_j itself or change the volume V_j if the weight depends on the volume: $\partial w_j / \partial V_j \ne 0$. We can learn the rule weights that minimize the mean squared error E with a gradient algorithm

$$V_j(t+1) = V_j(t) - \frac{\partial E}{\partial V_j} \tag{3-10}$$

for the instantaneous mean-squared error E

$$E = \tfrac{1}{2}(f(x) - F(x))^2.$$

The chain rule of differential calculus gives

$$\frac{\partial E}{\partial V_j} = \frac{\partial E}{\partial F}\frac{\partial F}{\partial V_j} \tag{3-11}$$

$$\frac{\partial E}{\partial F} = -(f(x) - F(x)) = -\varepsilon \tag{3-12}$$

and (3-13)–(3-16), shown at the bottom of the next page, where $p_j(x)$ is the convex coefficient defined in (2-15). When we combine (3-10) with the above result we get the volume update equation

$$V_j(t+1) = V_j(t) + \varepsilon_t p_j(x_t)[c_{y_j} - F(x_t)]\left(\frac{1}{V_j} + \frac{1}{w_j}\frac{\partial w_j}{\partial V_j} \right). \tag{3-17}$$

Suppose the rule weights w_j do not depend on the volumes V_j: $\partial w_j / \partial V_j = 0$. Then

$$\left(\frac{1}{V_j} + \frac{1}{w_j}\frac{\partial w_j}{\partial V_j} \right) = \frac{1}{V_j} \tag{3-18}$$

$$V_j(t+1) = V_j(t) + \varepsilon \frac{p_j(x)}{V_j(t)}[c_{y_j} - F(x)]. \tag{3-19}$$

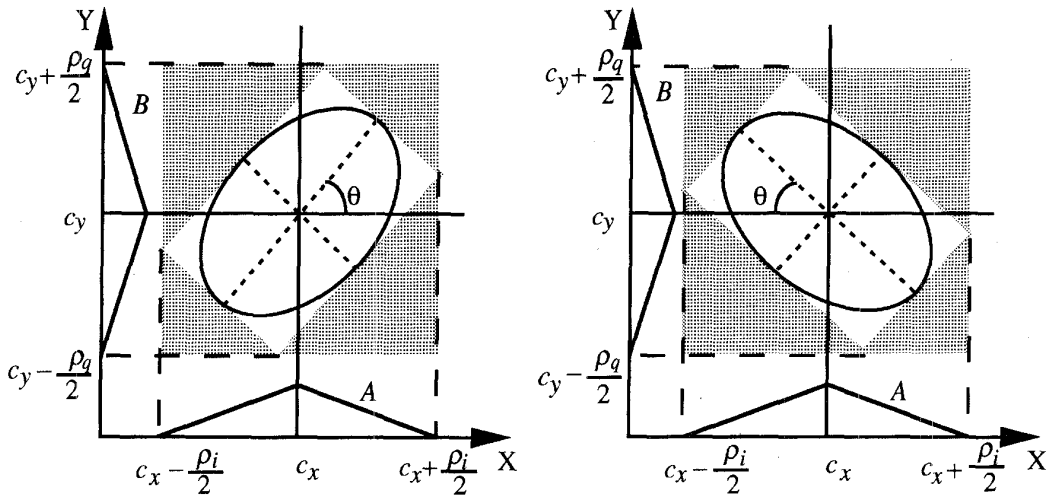

Fig. 6. Symmetric ellipsoidal patches give the same fuzzy rules and sets. The shaded region shows the fuzzy patch in the state space when we define the fuzzy sets as ellipsoidal projections. The projected if-part sets need not be triangular.

If the jth rule fires a little or not at all then V_j does not change. If the jth rule fires strongly ($a_j(x) \approx 1$) then learning depends strongly on how well $F(x)$ matches the then-part set centroid c_{y_j}. When V_j is large it tries to shut off learning since the rule is uncertain. When V_j is small the volume weight is large and it tries to move the output $F(x)$ toward the centroid c_{y_j}.

We can weight the then-part sets as in (2-6) with an inverse volume weight

$$w_j = \frac{1}{V_j} \tag{3-20}$$

to give rules with equal weights. For then

$$\left(\frac{1}{V_j} + \frac{1}{w_j}\frac{\partial w_j}{\partial V_j}\right) = \left(\frac{1}{V_j} - V_j\frac{1}{V_j^2}\right) = 0 \tag{3-21}$$

in (3-16). So the volume weights in (3-17) do not change. The inverse-volume weight (3-20) reduces $F(x)$ to an equal-weight or center-of-gravity [24] additive fuzzy system

$$F(x) = \frac{\sum_{j=1}^{m} a_j(x) c_{y_j}}{\sum_{j=1}^{m} a_j(x)}. \tag{3-22}$$

We can also use an inverse square weighting scheme that gives more weight to smaller more certain fuzzy rules

$$w_j = \frac{1}{V_j^2} \tag{3-23}$$

$$\left(\frac{1}{V_j} + \frac{1}{w_j}\frac{\partial w_j}{\partial V_j}\right) = \left(\frac{1}{V_j} - V_j^2\frac{2}{V_j^3}\right) = -\frac{1}{V_j}. \tag{3-24}$$

$$\frac{\partial F}{\partial V_j} = \frac{\left(\sum_{i=1}^{m} a_i(x) w_i V_i\right) a_j(x)\left(w_j + V_j\frac{\partial w_j}{\partial V_j}\right) c_{y_j} - \left(\sum_{i=1}^{m} a_i(x) w_i V_i c_{y_i}\right) a_j(x)\left(w_j + V_j\frac{\partial w_j}{\partial V_j}\right)}{\left(\sum_{i=1}^{m} a_i(x) w_i V_i\right)^2} \tag{3-13}$$

$$= \frac{a_j(x)\left(w_j + V_j\frac{\partial w_j}{\partial V_j}\right)}{\sum_{i=1}^{m} a_i(x) w_i V_i}\left[\frac{\left(\sum_{i=1}^{m} a_i(x) w_i V_i\right) c_{y_j}}{\sum_{i1=}^{m} a_i(x) w_i V_i} - \frac{\sum_{i=1}^{m} a_i(x) w_i V_i c_{y_i}}{\sum_{i=1}^{m} a_i(x) w_i V_i}\right] \tag{3-14}$$

$$= \frac{a_j(x)}{\sum_{i=1}^{m} a_i(x) w_i V_i}\left(w_j + V_j\frac{\partial w_j}{\partial V_j}\right)[c_{y_j} - F(x)] \tag{3-15}$$

$$= p_j(x)[c_{y_j} - F(x)]\left(\frac{1}{V_j} + \frac{1}{w_j}\frac{\partial w_j}{\partial V_j}\right) \tag{3-16}$$

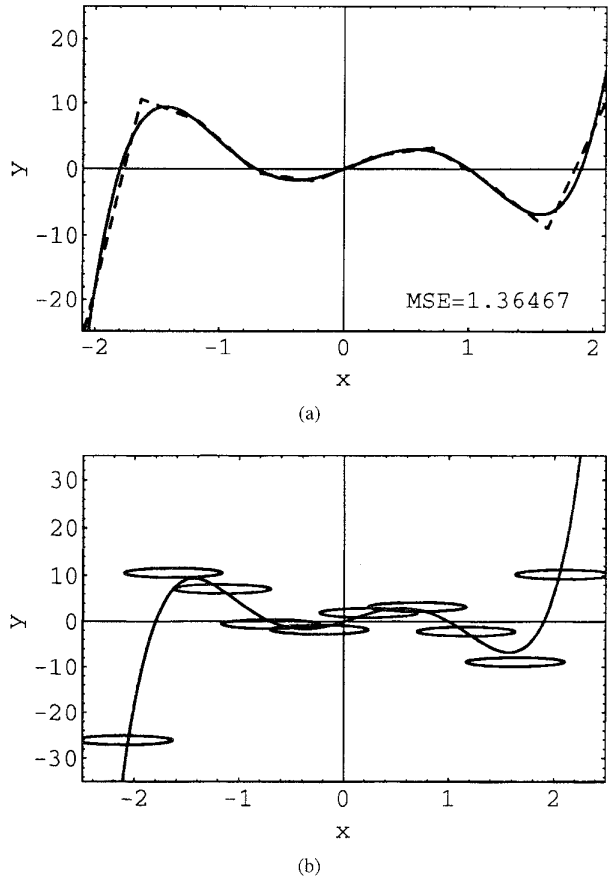

Fig. 7. The effect of different rule weights w_k on the output function approximation when ten rule patches approximate the function. (a) The function approximation for $w_j = 1/V_j$. Rules all have the same effect on the output calculation. (b) The rule patches for $w_j = 1/V_j$.

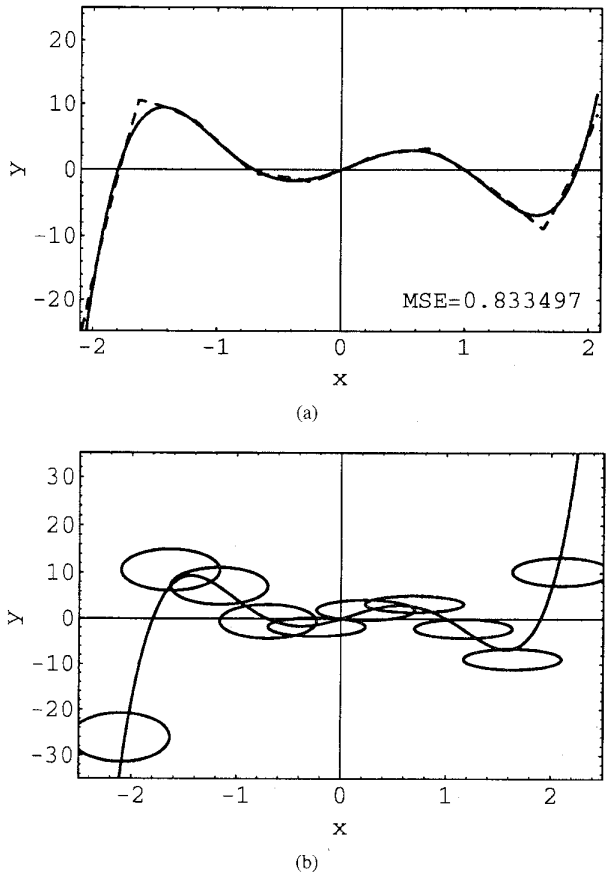

Fig. 8. The effect of different rule weights w_k on the output function approximation when ten rule patches approximate the function. (a) Equal weights on the rules: w_j = constant. (3-19) updates the weights. (b) Rule patches after volume learning. Rules with larger volumes have more effect on the output. Triangular if-part sets give a piecewise-linear fuzzy system.

The new learning increment cancels in sign that of (3-19)

$$V_j(t+1) = V_j(t) - \varepsilon \frac{p_j(x)}{V_j(t)}[c_{y_j} - F(x)]. \tag{3-25}$$

This weights certain rules with a small volume more heavily than rules with a larger volume in the centroid calculation of (2-1)

$$F(x) = \frac{\sum_{j=1}^{m} \frac{a_j(x)}{V_j} c_{y_j}}{\sum_{j=1}^{m} \frac{a_j(x)}{V_j}}. \tag{3-26}$$

The figures below show how learning the weights for ten fuzzy rule patches affects the fuzzy approximation of a fifth-order polynomial $f(x) = 3x(x-1)(x-1.9)(x+0.7)(x+1.8)$. The underlying data distribution is uniform between -2.1 and 2.1. We used (7-3) to find the optimal mean-squared centroids c_{y_j}. The centers of the if-part of the rules are equally spaced across the x-axis. Fig. 7 shows the case where $w_j = 1/V_j$. The weights do not change with time as shown in (3-21). Figs. 3-5 shows the case where the weights w_j are not a function of V_j. (3-19) updates the output set volumes. Figs. 3-6 shows the case when $w_j = 1/v_j^2$. (3-25) updates the output set volumes V_j.

IV. Unsupervised Covariance Ellipsoid Estimation

First-order and second-order statistics of the data can find the ellipsoidal fuzzy patches for an additive fuzzy system [5]. Unsupervised competitive learning laws estimate the covariance matrix $\boldsymbol{K}_k$ and centroid $\boldsymbol{s}_k$ of data clusters [14] as shown in the appendix. The covariance matrix $\boldsymbol{K}_k$ defines an ellipsoid in the input-output space for $\boldsymbol{A} = \boldsymbol{K}_k^{-1}$ in (3-1)

$$\alpha_k^2 = (\boldsymbol{z} - \boldsymbol{s}_k)^T \boldsymbol{K}_k^{-1} (\boldsymbol{z} - \boldsymbol{s}_k) \tag{4-1}$$

where $\boldsymbol{K}_k^{-1}$ is the inverse of the covariance matrix and $\boldsymbol{s}_k$ is the centroid of the kth pattern class.

A. Unsupervised Competitive Learning

Unsupervised competitive learning laws can learn statistics of data clusters [12, 14]. Each data cluster forms a fuzzy rule patch [14]. Adaptive vector quantization (AVQ) systems cluster quantization vectors in the input-output state

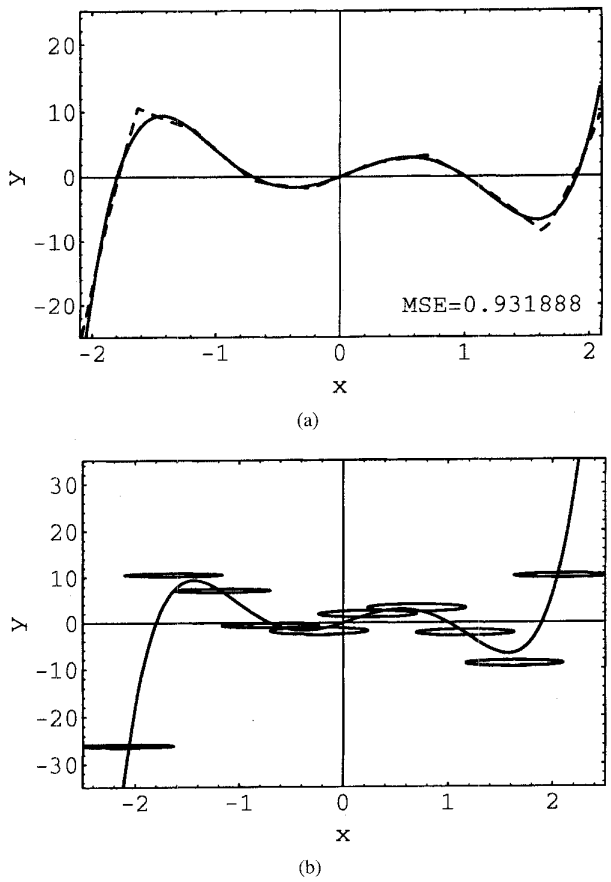

Fig. 9. The effect of different rule weights w_k on the output function approximation when ten rule patches approximate the function. (a) The function approximation for $w_j = 1/V_j^2$. (3-25) updates the weights. (b) Weighted rule patches after volume learning. Rules with larger volumes have more effect on the output.

space. The clustered quantizer vectors track the clusters in the incoming data. An autoassociative AVQ system combines the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$ of the data to form $\boldsymbol{z}^T = [\boldsymbol{x}^T|\boldsymbol{y}^T]$. Each data/quantization vector cluster forms a fuzzy rule patch in this form of *product-space clustering* [14]. The Appendix reviews the unsupervised competitive learning algorithm. Other unsupervised clustering techniques such as MacQueen's adaptive k-means [17] and fuzzy clustering [1], and generalized learning vector quantization [21] could also be adapted to find the fuzzy rules. [24] uses a fuzzy c-means algorithm to cluster data in the output domain only to form the then-part of the fuzzy rules.

The AVQ system clusters each sample with the closest centroid in Euclidean distance. Other distance metrics such as the Mahalanohbis or "city block" distance can be used depending on the data [12]. When data samples are sparse or noisy the covariance of the cluster or rule patch is large. This gives a large ellipsoid. When the data is dense the covariance of the cluster is small. This gives a smaller ellipsoid or more certain fuzzy rule. Fig. 10 shows how the data distribution can change the size of an ellipsoid rule patch. The Appendix shows that if the initial estimate of the covariance matrix $\boldsymbol{K}_k(0)$

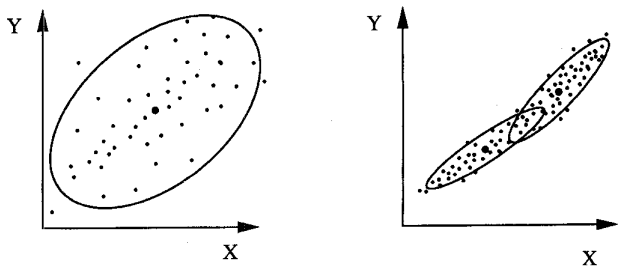

Fig. 10. The size of the ellipsoid rule patches depend on the data. (a) When the data is sparse or noisy the ellipsoid is large. (b) When the data is dense the ellipsoid is small.

is positive definite then $\boldsymbol{K}_k(t)$ is positive definite. So each covariance matrix forms an ellipsoid.

B. Fuzzy Rule Learning

Unsupervised ellipsoidal covariance learning finds the fuzzy rules and sets from the system input-output data [5]. The covariance estimate $\boldsymbol{K}_j^{-1}$ defines the jth ellipsoid in the q-dimensional input-output state space. q is the combined number of inputs n and outputs p to the fuzzy system. The projections of the ellipsoid onto the input and output axes bound the fuzzy sets.

Fig. 11 shows the function approximation for a fifth-order polynomial $f(x) = 3x(x-1)(x-1.9)(x+0.7)(x+1.8)$ when the data distribution is uniform. The weights for each rule are unity $w_k = 1$. More patches better cover the "bumps" in the function. The size of the patches shrinks as the number of patches grows. This gives more certain or less fuzzy rules. Fig. 12 shows the mean-squared error of the function approximation as the number of fuzzy rules grows. More patches give better MSE approximations.

The ellipsoid projections must cover the domain of the function so that the function is defined for all inputs. More overlap in the fuzzy patches smoothes the function approximation. The choice of α fixes how much the ellipsoids overlap. Small α values give a step-like approximation with weights from the output set centroid c_{y_i}. Larger α values smooth the function approximation since the ellipsoids overlap more. We used the same value for α for each of the ellipsoidal fuzzy patches in an approximation: $\alpha = \alpha_k$ for $k = 1, \cdots, r$. Optimal lone rules cover the extrema of f [16] to minimize the mean-squared error of the function approximation. The elliptical rule patches quickly move to cover these extrema in all of the learning schemes we studied.

The random distribution $p(\boldsymbol{z})$ of the data points $\boldsymbol{z}$ affects the accuracy of the approximation. Fewer or noisy data samples in a region result in larger ellipsoids since they have more variance. Smaller patches concentrate in regions of dense samples. Fig. 13 shows the ellipsoids and the function approximation when the data is Gaussian. Most ellipsoids lie near the middle of the function. The edges of the function have fewer and larger ellipsoids since there are fewer data points there.

In practice most control systems spend their time at equilibrium and so most input-output data comes from this region.

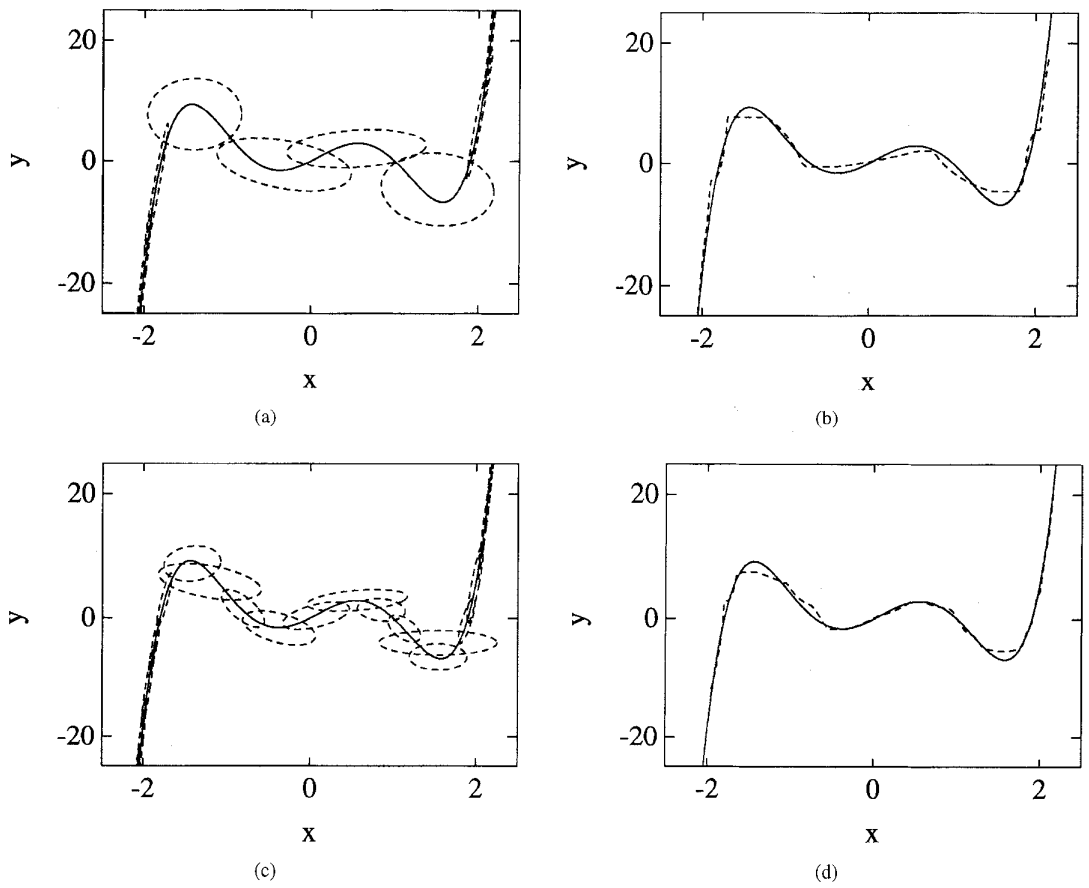

Fig. 11. The additive fuzzy system gave a better approximation as more ellipsoids covered the graph of the fifth order polynomial $f(x) = 3x(x-1)(x-1.9)(x+0.7)(x+1.8)$. (a) 10 ellipsoids cover the function. (b) Function approximation for 10 ellipsoids. (c) 20 ellipsoids cover the function. (d) Function approximation for 20 ellipsoids. Note that the ellipsoidal rules cover the extrema of the function in (a) and fill in between the extrema in (c).

AVQ systems estimate the unknown density $p(z)$ of the data. The quantizing vectors spread out to approximate $p(z)$. The rules far from equilibrium tend to be large since the data is sparse or noisy. For less frequent events the sets are larger to reflect this uncertainty. More certain rules occur where the system spends the most time. Fig. 14(a) shows the ellipsoidal rules found with unsupervised learning for Gaussian distributed data. In 2-1 with unity weights ($w_k = 1$) rules with a large volume V_k have a large effect on the output. Fig. 14(b) shows the results of weighting all of the rules with $w_k = 1$. The large rules on the sides dominate the calculation when they fire. Fig. 14(c) shows the results of weighting all of the rules equally with $w_k = 1/V_k$. Fig. 14(d) uses $w_k = 1/V_k^2$ where smaller rules have a larger effect than larger rules have.

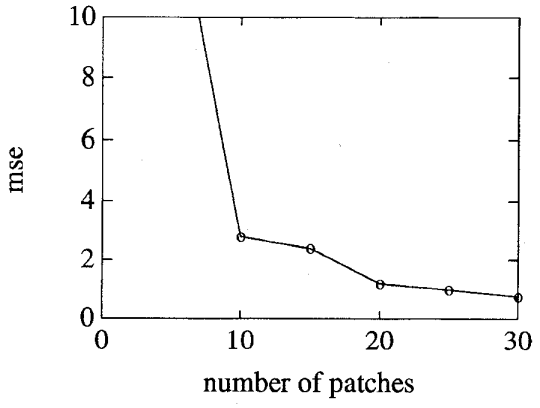

Fig. 12. Mean-squared error of function approximation for a fifth-order polynomial falls as the nurmber of patches increases.

C. Histogram Density Estimation

When the data fall in many pattern classes we can compute the density of the projections along the axes of the state space. Competitive learning estimates the unknown probability density function $p(z)$ that describes the distribution of patterns in the input-output state space. The quantization vectors track the data. Quantizing vectors tend to be dense or sparse where sample vectors are dense or sparse. This gives a nonparametric estimate of the density in the state space.

Low probability regions need enough quantizing vectors to estimate the pattern class statistics. Rare events may be as important as equilibrium events for good control or for stability. This method needs data or expert rules for these rare

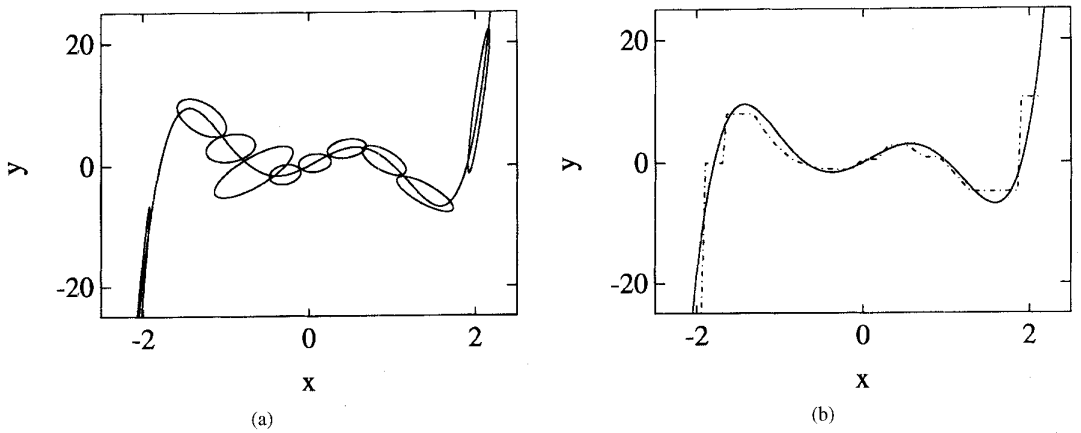

Fig. 13. Unsupervised ellipsoid covariance learning estimates the fuzzy rule patches. (a) 10 ellipsoidal patches when the data is Gaussian. (b) Function approximation for 10 ellipsoids when the data is Gaussian.

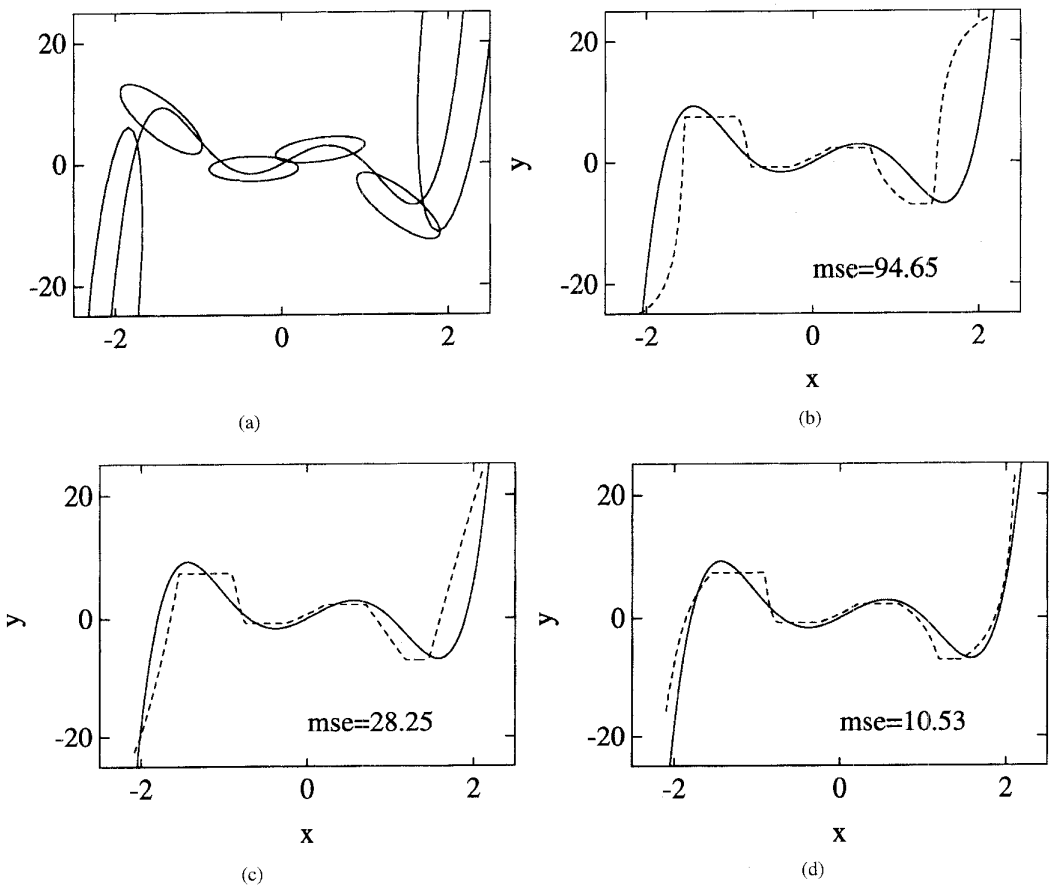

Fig. 14. The effect of different rule weights w_k on the output function approximation when six rule patches approximate the function. (b) Equal weights on the rules: $w_k = 1$. Rules with larger volumes have more effect on the output. (c) $w_k = 1/V_k$. Rules all have the same effect on the output calculation. (d) $w_k = 1/V_k^2$ gives more weight to smaller and thus more certain rules.

events. If data is not available the rules need to be added to the system by a knowledge engineer to ensure proper operation. Large numbers of fuzzy rules result if each quantizing vector counts as a rule. We can combine the quantizing vectors with histogram estimation.

Histogram estimation [7] gives a nonparametric estimate of a probability density. We sum up the weighted triangular projections of the ellipsoids to form the histogram of the density on the ith axis $g_i(zk)$

$$g_i(z_k) = \sum_{j=1}^{m} b_{ij}(z_k) \tag{4-2}$$

m is the number of quantizing vectors. $b_{ij}(z_k)$ is the triangular projection of the jth ellipsoid on the ith axis at z_k. Fig. 15

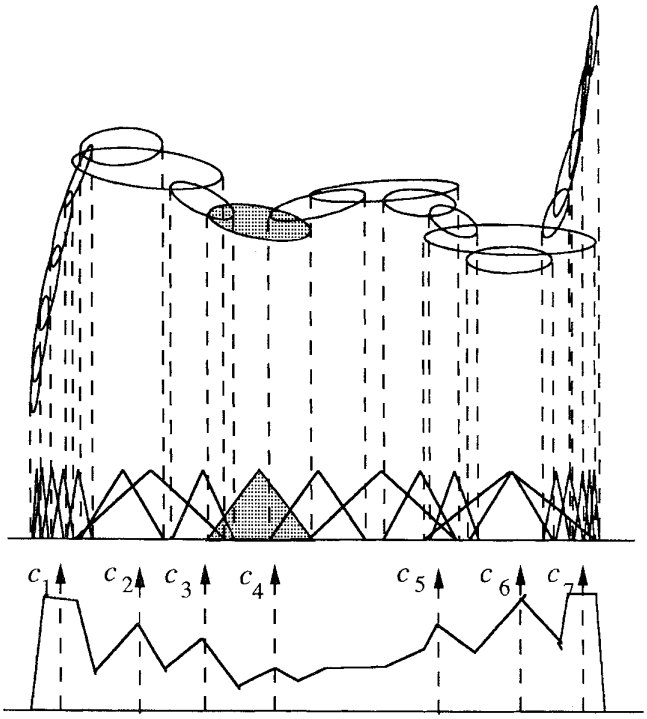

Fig. 15. Large numbers of ellipsoids estimate the unknown density $p(z)$ from the data. The ellipsoids project onto an axis. The sum of these projections forms a histogram. The peaks in this distribution give the centers of the FAM rule cells (shown by the arrow).

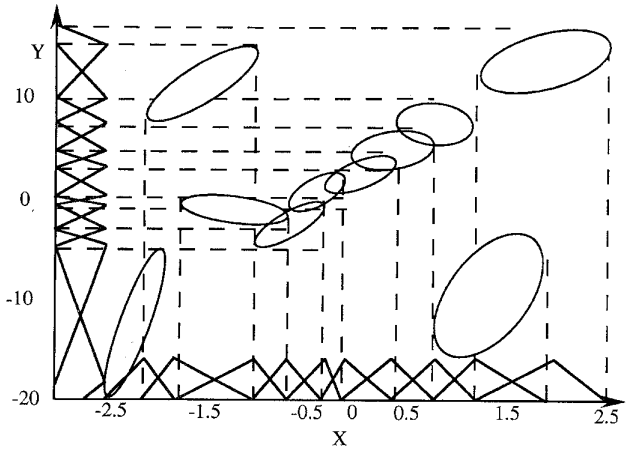

Fig. 16. Ten fuzzy rule patches found with the histogram method. 200 quantizing vectors approximated the unknown density $g(z)$.

shows the estimated density on an axis. The peaks of the distribution are the centers of the fuzzy sets in that dimension. We partition the state space along this grid and count the number of quantizing vectors in each cell. Clusters of quantizing vectors form a rule [5], [13], [14].

The ellipsoid scaling constant α changes the size of the projections on the axis. Larger values of α smooth the data. Small values of α give more resolution in areas where there are many quantizing vectors. Fig. 16 shows the output ellipsoidal patches chosen with the histogram method. 200 quantizing

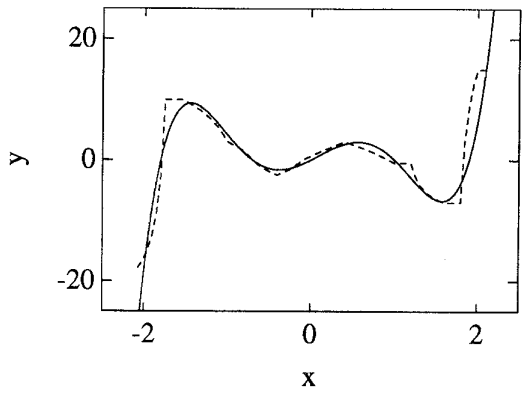

Fig. 17. Function approximation with ten fuzzy rule patches found with the histogram method for 200 quantizing vectors.

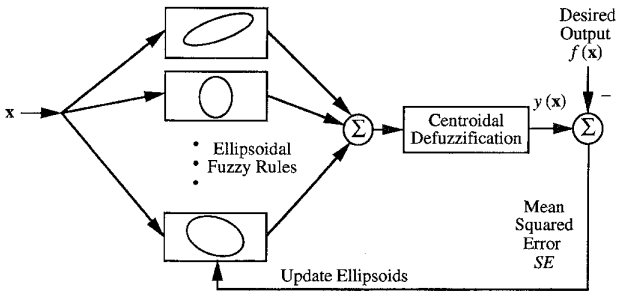

Fig. 18. Supervised ellipsoidal learning architecture.

vectors estimated the histogram. We used the ten highest peaks of the histogram to give the centers of the rules. Fig. 17 shows the function approximation.

V. Supervised Learning of Fuzzy Rule Patches

A supervised gradient descent system learns the ellipsoidal rules as it locally minimizes the mean-squared error of the function approximation as in Fig. 18. The neural system learns the size and shape of the fuzzy rule patches that minimize the function error.

A. Supervised Gradient Descent Algorithm

The supervised gradient descent algorithm [14] takes the gradient of the instantaneous mean-squared error E_k at time k

$$E_k = \tfrac{1}{2} \cdot (f(x_k) - F(x_k))^2 \qquad (5\text{-}1)$$

$f(x_k)$ is the value of the approximated function. $F(x_k)$ is the output of an additive fuzzy system for the input x_k in (2-1). The supervised gradient descent algorithm updates the fuzzy rule patches with the additive fuzzy system model in (2-1). Equations (3-3) and (3-5) define the volume V_i and the membership functions $a_i(x)$ for the ith output fuzzy set.

We use a discrete form of gradient steepest descent

$$\boldsymbol{El}_i(k+1) = \boldsymbol{El}_i(k) + \Omega_{c_k} \Delta_{\boldsymbol{El}_i} E_k \qquad (5\text{-}2)$$

$\boldsymbol{El}_i(k)$ is a concatenated vector of the ith ellipsoids' parameters. Ω_{c_k} is a diagonal matrix of decreasing learning constants.

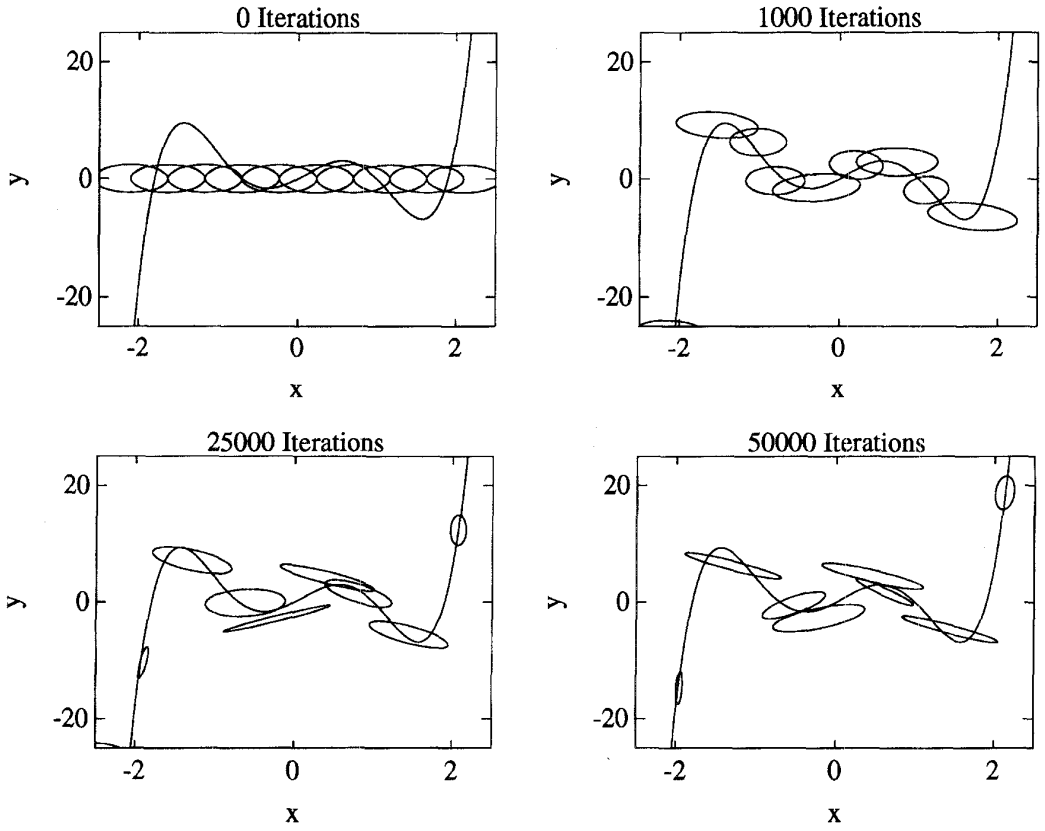

Fig. 19. Uniform initial ellipsoids that cover the domain of the function. The ellipsoids quickly cover the extrema of the function. Over time the ellipsoids spread out to minimize the mean-squared error.

$\Delta_{\boldsymbol{El}_i} E_k$ is the gradient vector of $\boldsymbol{El}_i(k)$. The components of $\boldsymbol{El}_i(k)$ are the output set centroid c_{y_i}, the input set centroid c_{x_i}, the volume of the output set V_i and the base of the input set ρ_{ik}. These variables fix the fuzzy rule patches.

Repeated applications of the chain rule of differential calculus give the ellipsoidal supervised gradient descent algorithm. Four equations update the ellipsoid parameters

$$\Delta c_{y_i}(k) = -\frac{\partial E_k}{\partial c_{y_i}^k} = -\frac{\partial E_k}{\partial F(x_k)} \frac{\partial F(x_k)}{\partial c_{y_i}^k} \tag{5-3}$$

$$\begin{aligned}\Delta c_{x_i}(k) &= -\frac{\partial E_k}{\partial c_{x_i}^k} = -\frac{\partial E_k}{\partial F(x_k)} \frac{\partial F(x_k)}{\partial c_{x_i}^k} \\ &= -\frac{\partial E_k}{\partial F(x_k)} \frac{\partial F(x_k)}{\partial a_i^k} \frac{\partial a_i^k}{\partial c_{x_i}^k}\end{aligned} \tag{5-4}$$

$$\Delta V_i(k) = -\frac{\partial E_k}{\partial V_i^k} = -\frac{\partial E_k}{\partial F(x_k)} \frac{\partial F(x_k)}{\partial V_i^k} \tag{5-5}$$

$$\Delta \rho_{ij}(k) = -\frac{\partial E_k}{\partial \rho_{ij}^k} = -\frac{\partial E_k}{\partial F(x_k)} \frac{\partial F(x_k)}{\partial a_i^k} \frac{\partial a_i^k}{\partial \rho_{ij}^k}. \tag{5-6}$$

The equations below give the partial derivatives used in (5-3) to (5-6)

$$\frac{\partial E_k}{\partial F(x_k)} = -(f(x_k) - F(x_k)) \tag{5-7}$$

$$\frac{\partial F(x_k)}{\partial c_{y_i}^k} = \frac{V_i^k a_i^k}{\sum_{j=1}^m V_j^k a_j^k} \tag{5-8}$$

$$\frac{\partial F(x_k)}{\partial a_i^k} = \frac{V_i^k \sum_{j=1}^m V_j^k a_j^k (c_{y_i}^k - c_{y_i}^k)}{\left(\sum_{j=1}^m V_j^k a_j^k\right)^2} \tag{5-9}$$

$$\frac{\partial a_i^k}{\partial c_{x_{il}}^k} = \begin{cases} \dfrac{2}{\rho_{il}} & \text{if } 0 < x - c_{x_{il}} \leq \dfrac{\rho_{il}}{2} \\ 0 & \text{else} \\ -\dfrac{2}{\rho_{il}} & \text{if } -\dfrac{\rho_{il}}{2} < x - c_{x_{il}} \leq 0 \end{cases} \tag{5-10}$$

$$\frac{\partial F(x_k)}{\partial V_i^k} = \frac{a_i^k \sum_{j=1}^m V_j^k a_j^k (c_{y_i}^k - c_{y_j}^k)}{\left(\sum_{j=1}^m V_j^k a_j^k\right)^2} \tag{5-11}$$

$$\frac{\partial a_i^k}{\partial \rho_{il}^k} = \begin{cases} \dfrac{2|x - c_{x_{il}}|}{\rho_{il}^2} & \text{if } |x - c_{x_{il}}| \leq \dfrac{\rho_{il}}{2} \\ 0 & \text{else.} \end{cases} \tag{5-12}$$

These equations are nonlinear. We used different magnitudes of learning coefficients for different parameters. This tends to

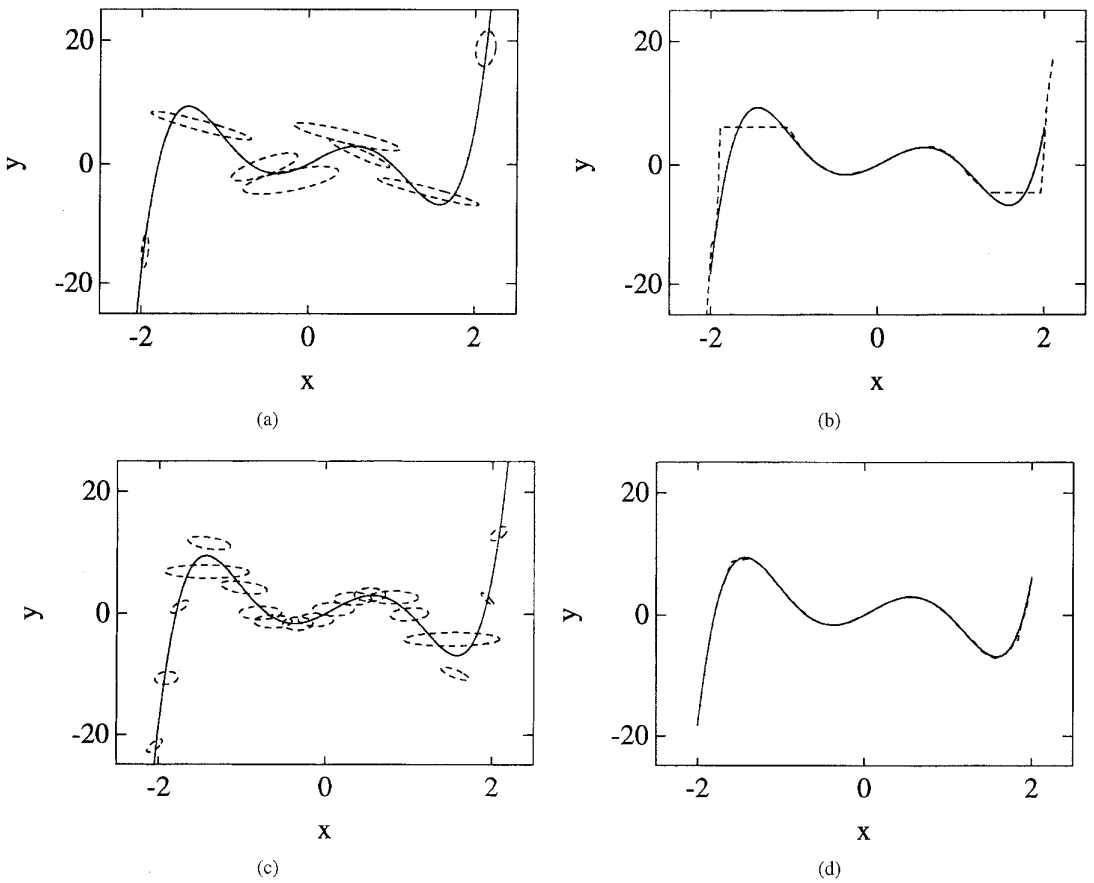

Fig. 20. More fuzzy patches gave less mean-squared error in the function approximation. (a) Ten ellipsoidal rules found with supervised learning. (b) Function approximation for 10 rules after 50 000 iterations. (c) 20 ellipsoidal rules found with supervised learning. (d) Function approximation for 20 rules after 100 000 iterations.

improve the gradient descent algorithm [9]. Some equations contain ρ_{ij}^2 in the denominator. So as the size of the sets shrinks the partial derivative becomes more sensitive to small changes in set size. We used learning coefficients of the form

$$\beta_k = \min(\zeta_k, \beta_{\max}) \tag{5-13}$$

β_k is the learning coefficient at time k [9]. $\beta_{\max}$ is the maximum coefficient size. The learning coefficient is

$$\zeta_k = \zeta\left(1 - \frac{k}{1.1N}\right). \tag{5-14}$$

ζ is the maximum learning coefficient. We kept the output set volumes V_i and the base of the triangular input sets ρ_{il} positive

$$V_i^k = \max(V_i^k, \varepsilon_v) \tag{5-15}$$

$$\rho_{il}^k = \max(\rho_{il}^k, \varepsilon_\rho) \tag{5-16}$$

ε_v and ε_ρ are small positive constants. So each rule forms an ellipsoid.

The supervised gradient descent algorithm can learn the size and shape of the projected fuzzy sets or the ellipsoid parameters [4]. The projected sets form a hyperrectangle in the input output state space. Fig. 6(a)-(b) shows how symmetric ellipsoid patches give the same fuzzy sets and rules. This leaves four parameters for the ith fuzzy rule: the output set centroid c_{y_i}, the input set centroid c_{x_i}, the volume of the output set V_i and the base of the input set ρ_{ik}. The rule weights w_j are equal to one ($w_j = 1$).

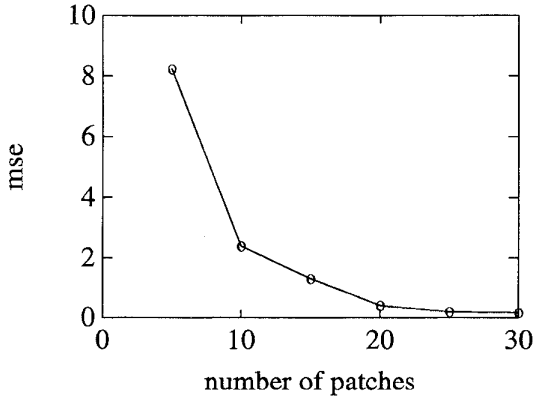

Fig. 21. The mean squared error falls as the number of fuzzy ellipsoidal rule patches grows for 100 000 iterations.

B. Supervised Function Approximation

We initialized the ellipsoids along the x axis. At each iteration the algorithm updated the parameter estimates of two ellipsoid fuzzy patches for the fifth-order polynomial $f(x) = 3x(x-1)(x-1.9)(x+0.7)(x+1.8)$. The supervised algorithm found 8 partial derivatives for each patch to update 4 parame-

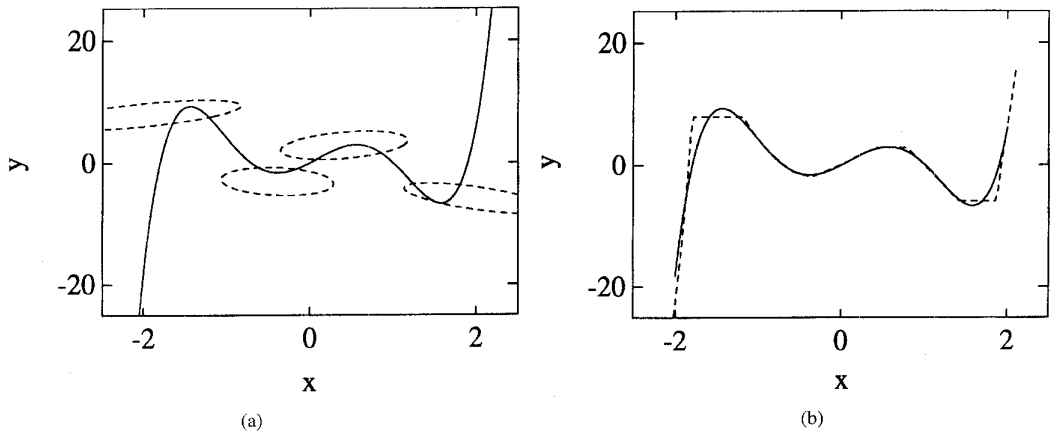

Fig. 22. Backpropagation ellipsoidal learning of fifth-order polynomial with Gaussian data distribution. (a) 10 ellipsoidal patches, most ellipsoids are at extrema but move off the graph after 150 000 iterations. (b) Function approximation for 10 rules.

ters for the single-input single-output case. The nonlinearities in the equations required small learning coefficients for stability. Fig. 19 shows how the ellipsoids learn or move over time.

Fig. 20 shows the ellipsoidal function approximation for this method. The ellipsoids do not cover the graph of the function as in the unsupervised case. The patch projections do cover the x axis. So each input x belongs to at least one fuzzy set to nonzero degree and fires at least one rule. The patch's size and shape do not change if only one fuzzy rule fires for an input. Only the output set centroid can change when the sets do not overlap. Some patches touch and pass through one another as they learn. Fig. 21 shows the mean-squared error as the number of ellipsoids grows. More ellipsoids gave better approximations.

Gaussian data requires more samples for convergence since there are fewer samples at the edge to tune the ellipsoids. Supervised ellipsoidal learning can learn the function. Fig. 22 shows the ellipsoidal fuzzy sets and the function approximation. Patches lie at the peaks in the functions.

VI. Hybrid System: Unsupervised and Then Supervised Ellipsoidal Learning

Our hybrid neural system combined unsupervised and supervised learning in Fig. 1. The hybrid system used unsupervised learning to quickly pick the first set of ellipsoidal fuzzy rules. It picked them in shape, orientation, and number. Then supervised learning tuned the rules.

The hybrid system acts as a human expert who learns to control a system. The unsupervised system acts as the expert as she first controls the system and turns her experience into a skill or a set of rough fuzzy rules. The supervised system acts as the expert as she improves her skill through trial and error.

The hybrid system performed better than did either the lone unsupervised or supervised system for the uniform case. Fig. 23 shows the hybrid approximation for 10 ellipsoidal fuzzy rules. The first stage of unsupervised learning speeded the later stage of supervised learning. Fig. 24 shows the mean-squared error for the fifth-order polynomial $f(x) = 3x(x - 1)(x - 1.9)(x + 0.7)(x + 1.8)$.

The unsupervised case gave a less accurate approximation when the data was Gaussian. The first choice of ellipsoid parameters dictates how well gradient descent performs and how fast it converges. The ellipsoids clustered in the areas of greatest data density. Fig. 25 shows the hybrid learning process for Gaussian data. The supervised system refined this suboptimal choice of ellipsoids. The mean-squared error was higher than it was in the lone supervised case.

The supervised gradient descent algorithm needs more computation than does the unsupervised competitive learning algorithm since at each time it updates two ellipsoids with five parameters. The unsupervised covariance method updates the winning quantizing vector and estimates the covariance and mean. It learns faster and is more robust in convergence. The learning coefficients for the ellipsoidal back propagation algorithm need to be small for stability. So many more iterations are needed to learn the fuzzy rules. The unsupervised method used about one-fifth the iterations as did the supervised method. The hybrid system converged in about one third to one half the iterations of the supervised method. The hybrid method was stabler since the initial approximation errors tended to be smaller. The hybrid system works well when the unsupervised system gives a good estimate of the ellipsoids. The hybrid system converges to a local minimum when the first estimate does not reflect the function.

VII. Optimal Function Approximation

The three learning methods change the position and size of the fuzzy rules. In this section we compare these results with the optimal solution. Optimal ellipsoidal values minimize the mean-squared error E_f of the function approximation

$$E_f = \int_{\boldsymbol{X}} [f(x) - F(x)]^2 \, dx$$

$$= \int_{\boldsymbol{X}} \left[f(x) - \sum_{j=1}^{m} p_j(x) c_{y_j} \right]^2 dx \qquad (7\text{-}1)$$

$$p_j(x) = \frac{V_j a_j(x)}{\sum_{i=1}^{m} V_i a_j(x)}. \qquad (7\text{-}2)$$

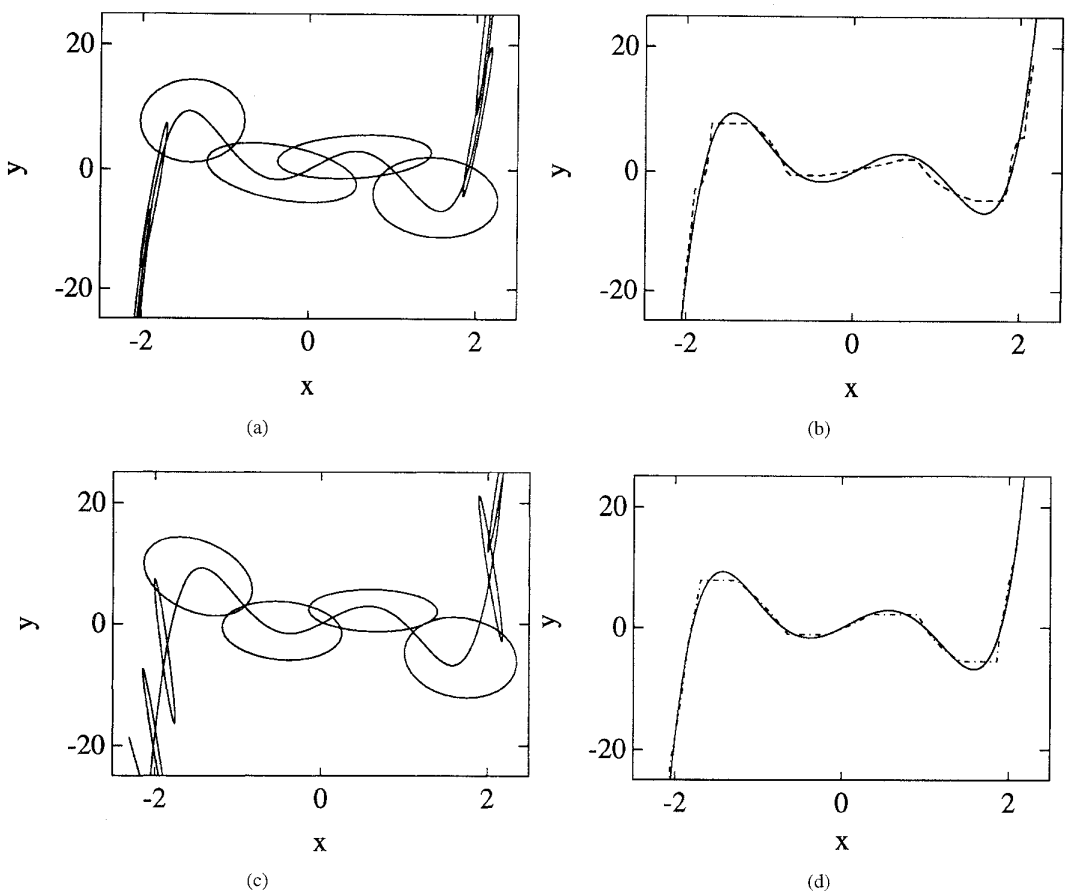

Fig. 23. Combined ellipsoidal learning methods gave a fast function approximation. (a) 10 ellipsoidal rules found with unsupervised learning. (b) Function approximation. (c) The same 10 ellipsoidal rules tuned with supervised learning after 25 000 iterations. (d) Final function approximation.

This equation is linear in c_y if just the output set centroid changes. Batch least squares gives m equations of the form

$$\sum_{j=1}^{m} \left[\int_{\boldsymbol{X}} p_i(x) p_j(x)\, dx \right] \hat{c}_{y_j} = \int_{\boldsymbol{X}} p_i(x) f(x)\, dx = \beta_i. \tag{7-3}$$

Fig. 26 shows the result for ten patches when only the output set centroid changes. The supervised system converges to the optimal mean-squared solution. Note that the optimal ellipsoid rule patches include those that cover the extrema of f. We initialized the rule patches uniformly along the x-axis.

When more parameters vary than the output set centroids the approximation improves. We found the optimal shape and size of the patches that minimize the mean-squared error with the *Matlab* routine *fmins* [25]. This routine finds the solution to a nonlinear equation with the Nelder-Meade method [2]. Fig. 27 shows the results for the optimal patches compared with the results of the hybrid system. The hybrid system places the patches at the extrema of the function as shown in Fig. 27(b). This resembles the results of the optimal system shown in Fig. 27(c).

VIII. Conclusions

Ellipsoids are a simple and natural form for fuzzy rules. Their math is well known and they arise as the covariance or uncertainty "balls" around quantizing vectors. Regions of sparse or noisy data lead to large covariance ellipsoids and thus less certain rules. Tightly clustered data lead to small ellipsoids and more certain rules. Higher-order statistics may improve estimates of fuzzy rule patch size and shape for nonlinear system data [18], [19].

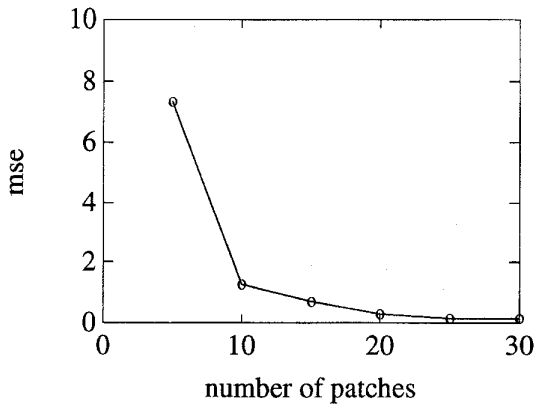

Fig. 24. Mean-squared error falls as the number of ellipsoidal fuzzy rule patches grows for the hybrid neural system.

Supervised learning can tune the ellipsoidal rules when we have error information. Even then the computation is heavy

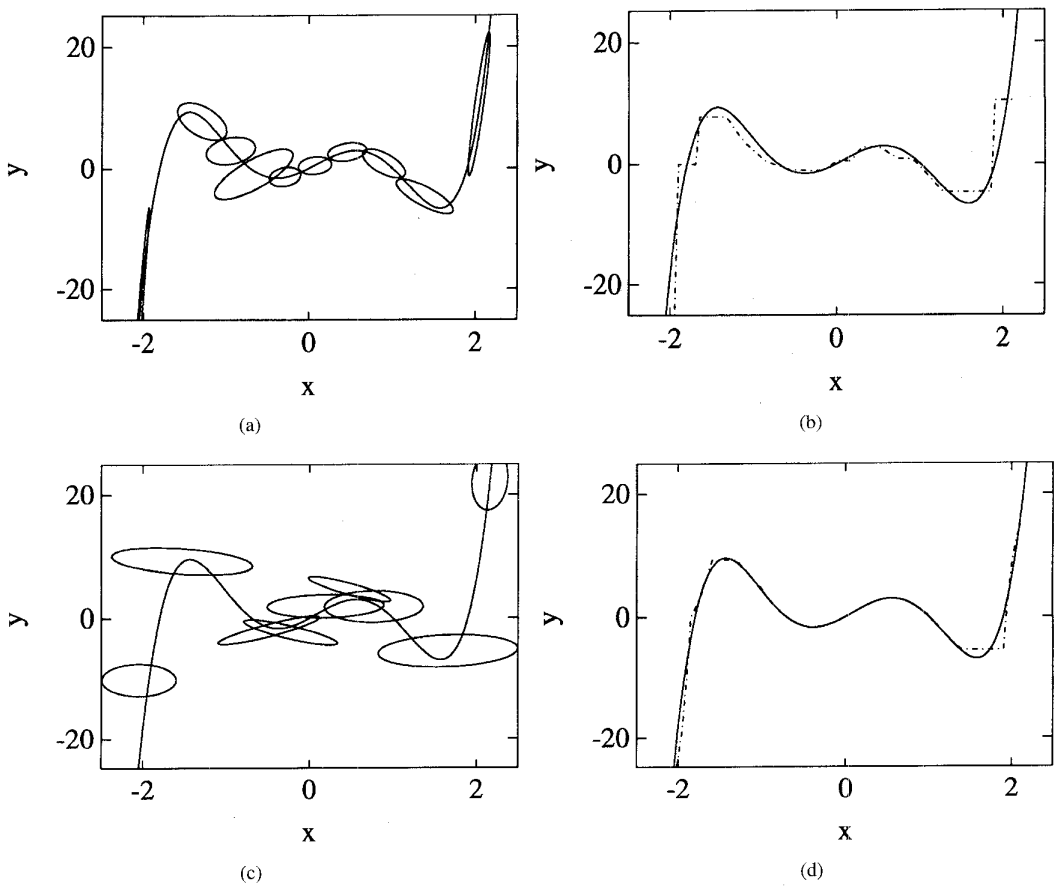

Fig. 25. Hybrid ellipsoidal learning estimates the fuzzy rule patches. (a) 10 ellipsoidal patches when the data is Gaussian. (b) Function approximation (c) 10 ellipsoidal patches from supervised tuning of fuzzy rules after 40 000 iterations. (d) Final function approximation for the Gaussian data using the hybrid algorithm for 10 rules. The hybrid system used one-third the processing that the supervised system did.

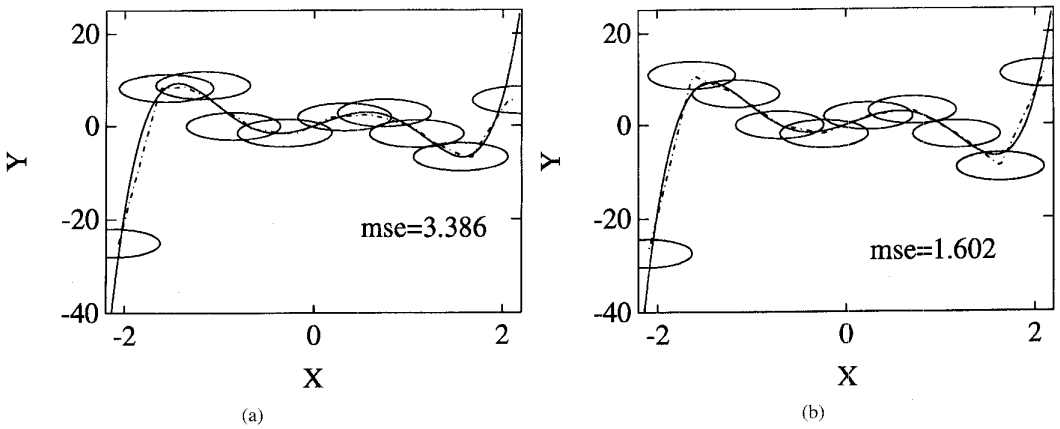

Fig. 26. Comparison between supervised learning (a) and the optimal function approximation (b) when only the centroids change. We spaced the initial sets uniformly along the x-axis between -2.1 and 2.1.

and someone or some system must pick the first set of ellipsoid parameters. Unsupervised learning can pick those parameters and both orient and speed the supervised gradient descent. Hybrid systems use the best of both techniques. They can learn fuzzy rules for problems of car control [3], inverse kinematics, or multimedia control and visualization [6].

Optimal ellipsoidal rules need complete knowledge of the approximand f. Learning can find some of this information as it estimates the optimal parameters. Future ellipsoidal fuzzy approximators may combine learning with closed-form models of both the shape and position of the if-part fuzzy sets.

APPENDIX
UNSUPERVISED COMPETITIVE LEARNING

Unsupervised competitive learning laws can learn the statistics of data clusters [14]. Adaptive vector quantization (AVQ)

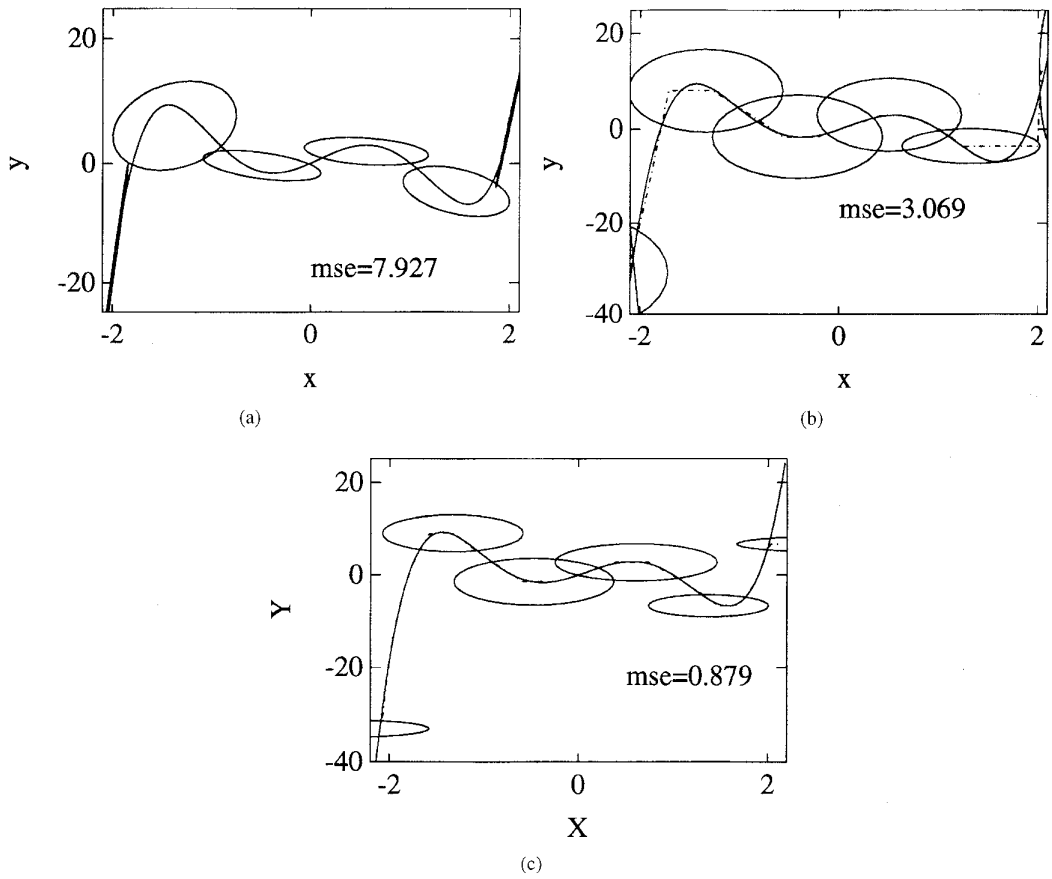

Fig. 27. Function approximation with six fuzzy rules. (a) shows the function approximation and ellipsoid rules after unsupervised learning. (b) shows the function approximation and ellipsoidal rules after 50 000 iterations of supervised learning. (c) shows the optimal function approximation found with least squares and the Nelder-Meade algorithm.

systems cluster quantization vectors in the input-output state space. The clusters in the quantization vectors track the clusters in the data. An autoassociative AVQ system combines the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$ of the data to form $z^T = [\boldsymbol{x}^T|\boldsymbol{y}^T]$. Fig. 28 shows the geometry of an autoassociative AVQ system. Each data cluster forms a fuzzy rule patch in the form of *product space clustering* [14].

Neurons compete for activation from input patterns $\boldsymbol{z}$. The r quantizing vectors $\boldsymbol{s}_j$ define the r columns of the synaptic connection matrix $\boldsymbol{S}$. $\boldsymbol{S}$ connects the q input neurons in the combined input field F_Z to the r competing neurons in the output field F_W

$$\boldsymbol{w} = \boldsymbol{z}^T \boldsymbol{S}. \tag{A-1}$$

The AVQ system compares the current vector random sample $\boldsymbol{z}(t)$ with the r columns of the synaptic connection matrix $\boldsymbol{S}$. The jth neuron wins if the jth quantizing vector $\boldsymbol{s}_j(t)$ is closest in Euclidean distance

$$\|\boldsymbol{s}_j(t) - \boldsymbol{z}(t)\| = \min_i \|\boldsymbol{s}_i(t) - \boldsymbol{z}(t)\|. \tag{A-2}$$

The winning neuron learns the input pattern.

Competitive learning estimates the first-order statistics of the data with the stochastic difference equation

$$\boldsymbol{s}_j(k+1) = \begin{cases} \boldsymbol{s}_j(k) + \psi_k[\boldsymbol{z}_k - \boldsymbol{s}_j(k)] & \\ \qquad\qquad \text{if the } j\text{th neuron wins} & \\ \boldsymbol{s}_j(k) \qquad \text{if the } j\text{t; neuron loses} & \end{cases} \tag{A-3}$$

The learning coefficients ψ_k should decrease to satisfy the conditions for convergence [14]. The winning vector $\boldsymbol{s}_j$ learns or changes. The losing vectors $\boldsymbol{s}_k$ do not change and so do not forget what they have learned. We used $\psi_k = 0.2[1 - k/(1.2N)]$ where N is the number of data points.

The competitive quantizing vectors converge exponentially quickly to pattern-class centroids [13], [14]. At equilibrium the average quantizing vector $E[\boldsymbol{s}_j]$ equals the jth centroid. Centroid estimation requires that a "signal function" $R_j(w_j)$ approximate the indicator function of the cluster $\boldsymbol{D}_j$ with a near binary win-loss function

$$R_j(w_j) = \begin{cases} 1 & \text{if } \boldsymbol{z} \in \boldsymbol{D}_j \\ 0 & \text{if } \boldsymbol{z} \notin \boldsymbol{D}_j \end{cases} \tag{A-4}$$

We scale the components of the data sample $\boldsymbol{z}(t)$ so that all features have equal weight in the distance measure [7].

Most estimators have covariances. Centroids give a first-order estimate of how the unknown probability density function $p(\boldsymbol{z})$ behaves in the regions $\boldsymbol{D}_j$. Local covariances give a

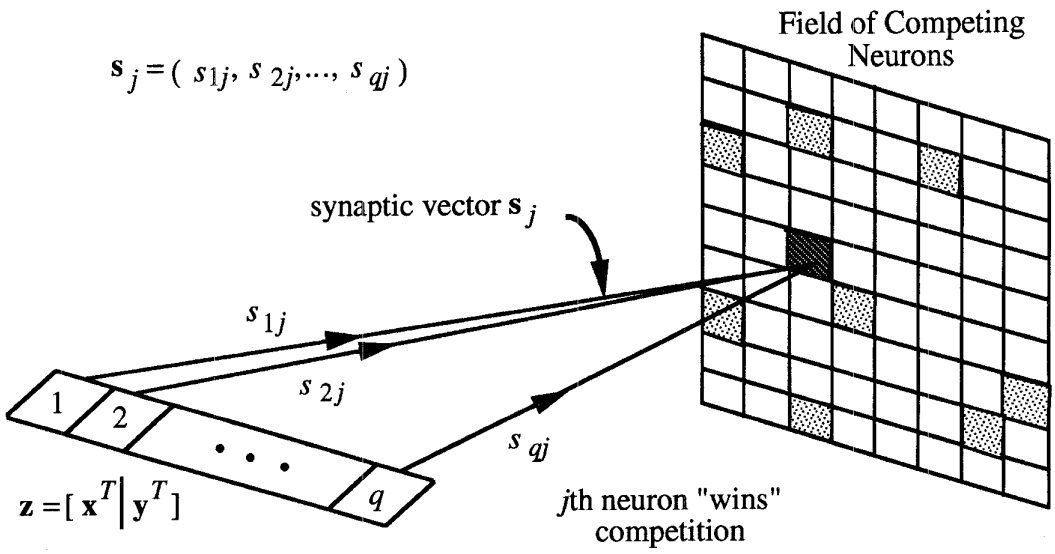

Fig. 28. Autoassociative AVQ architecture combines the input x and the output y. Each quantizing vector competes for activation from input pattern z. The winning neuron learns the input pattern as its synaptic quantizing vector moves toward z.

second order estimate. Competitive learning laws can asymptotically estimate [14] the local conditional covariance matrix K_j in pattern class D_j

$$K_j = E[(z - \bar{z}_j)(z - \bar{z}_j)^T | D_j] \tag{A-5}$$

At each iteration we estimate the unknown centroid $\bar{z}_j$ as the current quantizing vector s_j. In this sense K_j becomes an error conditional covariance matrix. This leads to the discrete stochastic algorithm [14]

$$K_j(k+1) = \begin{cases} K_j(k) + d_k[(z_k - s_j(k))(z_k - s_j(k))^T \\ \quad -K_j(k)] \\ \qquad \text{if the } j\text{th neuron wins} \\ K_j(k) \qquad \text{if the } j\text{th neuron loses} \end{cases} \tag{A-6}$$

for the quantizing vectors. The learning coefficients d_k must also decrease to satisfy the conditions for convergence [14]. We used

$$d_k = 0.2\left[1 - \frac{k}{1.2N}\right]. \tag{A-7}$$

N is the number of data points. $K_j(k+1)$ converges to the local conditional covariance matrix. The covariance estimate defines an ellipsoidal patch in the state space if K_j is positive definite and we set A in (3-1) to K_j^{-1}.

We assume that $K_j(k+1)$ is positive definite by the following logic. Assume that the initial covariance estimate $K_j(0)$ is positive definite ($x^T K_j(0) x > 0$, for all nonzero vectors x). If the jth neuron wins (A-6) updates the covariance matrix

$$K_j(1) = [1 - d_0]K_j(0) + d_0(z_k - s_j(0))(z_k - s_j(0))^T \tag{A-8}$$

$0 < d_0 < 1$ is the learning coefficient in (A-7). The first term $A = (1 - d_0)K_j(0)$ is positive definite if $(1 - d_0)$ is positive [23]. The outer product matrix $B = (z_k - s_j(0))(z_k - s_j(0))^T$ is a positive semi-definite matrix of rank 1 with one positive eigenvalue and the rest zero ($x^T B x \geq 0$ for all nonzero vectors x) [23]. A positive-definite matrix A plus a positive semi-definite matrix B form a positive definite matrix ($x^T(A + B)x > 1$, for all nonzero vectors x). $K_j(1)$ is positive definite. If the jth neuron loses the covariance estimate does not change. The proof proceeds by induction. We initialized the covariance matrix estimate $K_j(0)$ to small positive numbers on the diagonal to ensure that K_j is positive definite.

Steps 1 through 3 give the competitive AVQ algorithm [13]:

A. Competitive AVQ Algorithm

1) Initialize the quantizing vectors: $s_i(0) = z(i), i = 1, \cdots, r$.
2) For a random data sample $z(t)$, find the closest or "winning" quantizing vector $s_j(t)$:

$$(s_j(t) - z(t))^T \Omega (s_j(t) - z(t)) = \min_i (s_i(t) - z(t))^T \Omega (s_i(t) - z(t)) \tag{A-9}$$

where

$$\Omega = \begin{bmatrix} \omega_1^2 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \omega_q^2 \end{bmatrix}$$

is a scaling matrix.
3) Update the winning quantizing vector $s_j(t)$ with (A-3) and its local covariance estimate $K_j(t)$ with (A-6).

We used the scaling matrix Ω

$$\Omega - \begin{bmatrix} 1 & 0 \\ 0 & 0.15^2 \end{bmatrix}. \tag{A-10}$$

This scaling matrix normalized the Euclidean distance calculation in (A-9) so that one variable did not dominate the choice of winners. We scaled the components of the data sample $z(t)$

so that all features have equal weight in the distance measure [7]. Then no feature dominates and the classification is scale invariant.

REFERENCES

[1] J. C. Bezdek, E. C. Tsao, and N. R. Pal, "Fuzzy Kohonen clustering networks," presented at *IEEE Int. Conf. on Fuzzy Systems*, San Diego, pp. 1035–1041, 1992.

[2] J. E. Dennis and D. J. Woods, "New computing environments: microcomputers in large-scale computing," *SIAM*, pp. 116–122, 1987.

[3] J. A. Dickerson and B. Kosko, "Ellipsoidal learning for smart car platoons," presented at *SPIE Int. Symp. on Optical Tools for Manufacturing Technologies—Applications of Fuzzy Logic Technology*, Boston, 1993.

[4] ——, "Fuzzy function approximation with supervised ellipsoidal learning," presented at *World Conf. on Neural Networks (WCNN '93)*, Portland, OR, vol. 2, pp. 9–17, 1993.

[5] ——, "Fuzzy function learning with covariance ellipsoids," presented at *IEEE Int. Conf. on Neural Networks (IEEE ICNN-93)*, San Francisco, pp. 1162–1167, 1993.

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

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Orlando: Harcourt Brace Jovanovich, 1972.

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

[9] S. S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

[10] F. B. Hildebrand, *Advanced Calculus for Applications*, 2d ed. Englewood Cliffs, NJ: Prentice Hall, 1976.

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

[12] T. Kohonen, *Self-Organization and Associative Memory*, 2d ed. Berlin: Springer-Verlag, 1988.

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

[14] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs: Prentice Hall, 1991.

[15] ——, "Fuzzy systems as universal approximators," *IEEE Trans. Computers*, vol. 43, pp. 1329–1333, 1994; an early version appears in *Proc. 1st IEEE Int. Conf. on Fuzzy Systems (IEEE FUZZ-92)*, pp. 1153–1162, Mar. 1992.

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

[17] J. MacQueen, "Some methods for classification and analysis of multivariate observations," presented at *5th Berkeley Symp. of Mathematical Statistics and Probability*, pp. 281–297, 1967.

[18] J. M. Mendel, "Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications," *Proc. IEEE*, vol. 79, pp. 278–305, 1991.

[19] C. Nikias and M. R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, vol. 75, pp. 969–891, 1987.

[20] P. J. Pacini and B. Kosko, "Adaptive Fuzzy System for Target Tracking," *Intell. Syst. Eng.*, vol. 1, pp. 3–21, 1992.

[21] N. R. Pal, J. C. Bezdek, and E. C.-K. Tsao, "Generalized clustering networks and kohonen's self-organizing scheme," *IEEE Trans. Neural Networks*, vol. 4, pp. 549–557, 1993.

[22] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 569–576, 1991.

[23] G. Strang, *Linear Algebra and Its Applications*, 2d ed. New York: Academic, 1980.

[24] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Systems*, vol. 1, pp. 7–31, 1993.

[25] The Mathworks, *The Student Edition of MATLAB*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

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

[27] F. A. Watkins, "Fuzzy engineering," Ph.D Dissertation, Dept. Electrical and Computer Engineering, University of California at Irvine, Order number 9417921, University Microfilms Int., 300 North Zeeb Road, Ann Arbor, MI 48106, 1994.

[28] F. A. Watkins, "The representation problem for additive fuzzy systems," *Proc. 1995 IEEE Int. Conf. on Fuzzy Systems (IEEE FUZZ-95)*, vol. I, 117–122, Mar. 1995.

Julie A. Dickerson received the B.S. degree in electrical engineering from the University of California, San Diego. and the master's and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles. She designed radar systems for Hughes Aircraft company and Martin Marrietta while pursuing the Ph.D. degree.

She is an Assistant Professor, Department of Electrical and Computer Engineering, Iowa State University, Ames. Her current research activities are fuzzy systems, neural networks, smart cars and virtual reality systems.

Bart Kosko received bachelor degrees in philosophy and economics from the University of Southern California, Los Angeles, the masters degree in applied mathematics from the University of California, San Diego, and the Ph.D. degree in electrical engineering from the University of California, Irvine.

He is director of USC's Signal and Image Processing Institute and an associate professor of electrical engineering. He has authored the book *Fuzzy Thinking* and the Prentice Hall texts *Neural Networks* and *Fuzzy Systems, Neural Networks for Signal Processing*, and *Fuzzy Engineering*.

Dr. Kosko is an elected governor of the International Neural Networks Society (INNS). He has chaired or cochaired several neural and fuzzy conferences and cochairs the INNS 1996 World Congress on Neural Networks.