[7] Y. Izui and A. Pentland, "Analysis of neural networks with redundancy," *Neural Computation*, vol. 2, pp. 226–238, 1990.

[8] C. Neti, M. Schneider, and E. Young, "Maximally fault tolerant neural networks and nonlinear programming," in *Proc. Int. Joint Conf. on Neural Netw.*, San Diego, CA, June 1990, pp. II–483–496.

[9] J. Nijhuis, B. Hofflinger, A. van Schaik, and L. Spaanenburg, "Limits to the fault-tolerance of a feedforward neural network with learning," in the *Proc. 20th IEEE Fault Tolerant Computing Symp.*, 1990.

[10] D. B. Parker, "Learning logic," Tech. Rep. TR-47, Center for Computational Res. in Economics and Management Sci., MIT Cambridge, MA, 1985.

[11] D. S. Phatak and I. Koren, "A study of fault tolerance properties of artificial neural nets," Tech. Rep., Elec. and Comput. Eng. Dep., Univ. of Massachusetts, Amherst, 1991.

[12] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, pp. 1481–1497, Sept. 1990.

[13] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge, MA: MIT Press, 1986.

[14] D. E. Rumelhart, "Learning and generalization," transcript of plenary address appears in *Proc. IEEE ICNN* San Diego, CA, 1988.

[15] B. E. Segee and M. J. Carter, "Fault tolerance of pruned multilayer networks," in *Proc. IJCNN*, Seattle, WA, 1991, pp. II–447–452.

[16] B. E. Segee and M. J. Carter, "Fault sensitivity and nodal relevance relationships in multi-layer perceptrons," Tech. Rep. ECE.IS.90.02, Dep. of Elec. and Comput. Eng., Univ. of New Hampshire, Durham, NH, Mar. 1990.

[17] C. H. Séquin and R. D. Clay, "Fault tolerance in artificial neural networks," in *Proc. IJCNN*, San Diego, CA, June 1990, pp. I–703–708.

[18] R. L. Watrous, "Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, San Diego, CA, 1987, vol. II, pp. 619–627.

[19] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA, 1974.

Fuzzy Systems as Universal Approximators

Bart Kosko

***Abstract*—An additive fuzzy system can uniformly approximate any real continuous function on a compact domain to any degree of accuracy. An additive fuzzy system approximates the function by covering its graph with fuzzy patches in the input-output state space and averaging patches that overlap. The fuzzy system computes a conditional expectation $E[Y \mid X]$ if we view the fuzzy sets as random sets. Each fuzzy rule defines a fuzzy patch and connects commonsense knowledge with state-space geometry. Neural or statistical clustering systems can approximate the unknown fuzzy patches from training data. These adaptive fuzzy systems approximate a function at two levels. At the local level the neural system approximates and tunes the fuzzy rules. At the global level the rules or patches approximate the function.**

I. Fuzzy Approximation as a Fuzzy Covering

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. Fig. 1 shows how fuzzy patches in the input-output product space $X \times Y$ cover the real function $f : X \longrightarrow Y$. In Fig. 1(a) a few large patches approximate f. In Fig. 1(b) several smaller patches better approximate f. The approximation improves as we add more small patches but storage and complexity costs increase. This brief contribution gives the algebraic details of the fuzzy approximation.

A fuzzy system is a set of if-then fuzzy rules that maps inputs to outputs. Each fuzzy rule defines a fuzzy patch in the input–output state space of the function. Fig. 2 shows the fuzzy rule "if X is Negative Small, then Y is Positive Small" as the cartesian product $NS \times PS$ of "fuzzy" [12] or multivalued sets NS and PS. A 3-D plot would show the fuzzy patch $NS \times PS$ as a barn-like structure that rises up from its rectangular base. Each input belongs to some degree to each input fuzzy set. So each input fires all the fuzzy rules to some degree. Experts state the fuzzy rules or a neural or statistical system learns them from sample data. Experts and algorithms can give different sets of fuzzy rules and so give different approximations of the function.

Next, we show that a fuzzy system can approximate any continuous real function defined on a compact (closed and bounded in $\mathcal{R}^n$) domain and show that even a bivalent expert system can uniformly approximate a bounded measurable function. The fuzzy systems have a feedforward architecture that resembles the feedforward multilayer neural systems used to approximate functions [4]. The uniform approximation of continuous functions allows us to replace each continuous fuzzy set with a finite discretization or a point in a unit hypercube [7] or "fuzzy cube" of high dimension.

Hornik and White [4] and others have used the Stone–Weierstrass theorem of functional analysis [9] to show uniform convergence of neural networks. The Stone–Weierstrass theorem states that if $C(X)$ is the sup-norm space of continuous functions on a compact and Hausdorf X and if $\mathcal{A} \subset C(X)$ is a closed algebra and if $\mathcal{A}$ is self-adjoint and separates points and contains the constant functions, then $\mathcal{A} = C(X)$. This gives the result but not much insight into how to build or learn real systems. Radial basis nets sum Gaussian functions and also uniformly approximate continuous functions on compact sets [3]. Additive fuzzy systems [7] with Gaussian fuzzy sets [11] define radial basis nets and so also act as uniform approximators. The theorem below also proves so directly since it holds for all aditive systems. The constructive proof below shows how to use neural systems to learn rules and how to let the rules or patches change with time to track a nonstationary function.

II. Additive Fuzzy Systems

Inputs fire the if-part A_j of all fuzzy rules "if $X = A_j$, then $Y = B_j$" and give scaled sum B'_j as output. Earlier fuzzy systems combined the output fuzzy sets B'_j with pairwise maximum in accord with the so-called "extension principle" [2]. Additive fuzzy systems sum the outputs [7] as in Fig. 3:

$$B = \sum_{j=1}^{m} w_j B'_j. \tag{1}$$

For now we take the adaptation weights w_j as unity: $w_j = 1$.

Different limit theorems show how the different combination schemes behave. Sum combination often tends to give a symmetric unimodal distribution as the output fuzzy set B. It gives the global output centroid in (3) as the simple convex sum of set centroids in (6).

Manuscript received July 6, 1992; revised August 10, 1993. Earlier versions of this work appear in *Proceedings of the 1992 IEEE International Conference on Fuzzy Systems (FUZZ-92)* and *Proceedings of the IJCNN-92.*

The author is with the Department of Electrical Engineering-Systems, Signal and Image Processing Institute, University of Southern California, 3740 McClintock Avenue, EEB 400, Los Angeles, CA 90089-2564 USA; e-mail: kosko@sipi.usc.edu.

IEEE Log Number 9402909.

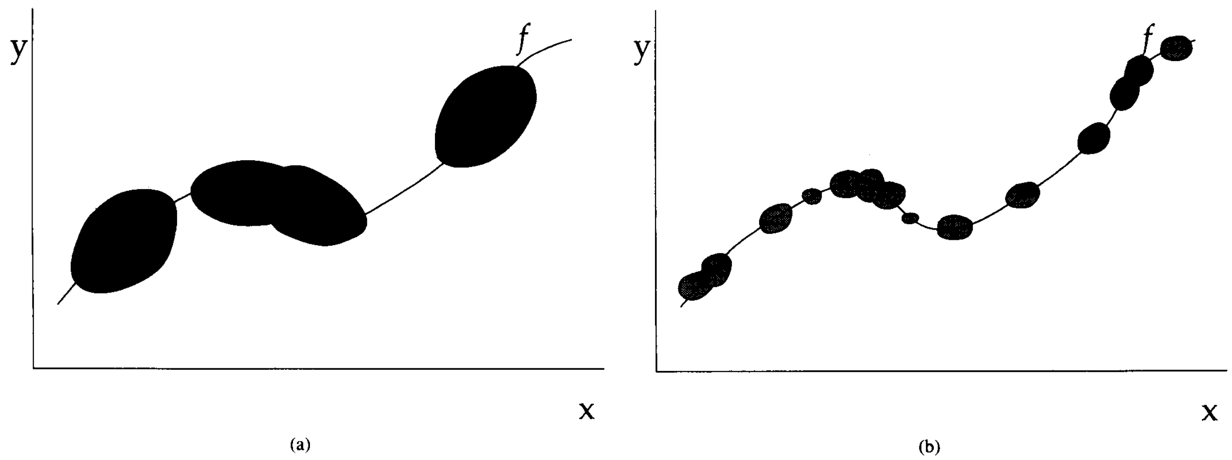

Fig. 1. (a) Four large fuzzy patches cover part of the graph of the unknown function $f : X \longrightarrow Y$. Fewer patches decrease computation and decrease approximation accuracy. (b) More smaller fuzzy patches better cover f, but at greater computational cost. Each fuzzy rule defines a patch in the product space $X \times Y$. In a large but finite number of fuzzy rules or precise rules covers the graph with arbitrary accuracy. Optimal rules cover extrema of f.

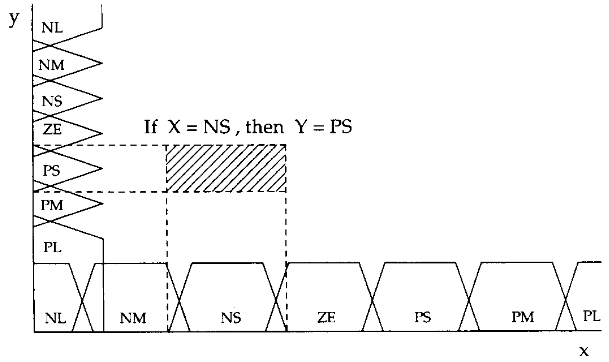

Fig. 2. Fuzzy rule as a state-space patch or cartesian product of fuzzy sets. The product patch $NS \times PS$ stands for the fuzzy rule "if X is Negative Small, then Y is Positive Small." Here trapezoids and triangles define fuzzy or multivalued sets.

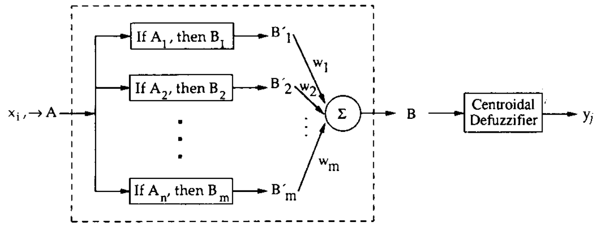

Fig. 3. Additive fuzzy system architecture. Real number x_i defines a unit bit vector or Dirac delta pulse that passes through the FAM system and fires each fuzzy rule to some degree (most to zero degree). The system *adds* the scaled output fuzzy sets to give B. Neural or other adaptive systems can modify the rule weights w_i (or the sets A_j and B_j) with sample data. The centroid of B gives the output number y_j as a convex sum of set centroids.

Combining the scaled or "fired" consequent fuzzy sets $B'_1, \cdots, B'_m$ in Fig. 3 with pairwise maximum gives the envelope of the fuzzy sets and tends towards the uniform distribution [5]. In general the Borel–Cantelli Lemma of probability theory implies that the "extension principle" [2] of fuzzy theory extremizes fuzzy membership values into binary endpoint values:

$$\limsup_{i \to \infty} x_i^1 \wedge x_i^2 \wedge \cdots \wedge x_i^n = b, \tag{2}$$

which holds with probability one [5] for nondegenerate pairwise independent sequences of i.i.d. random variables x_i^j that take values in $[a, b]$. The symbol "$\wedge$" stands for pairwise minimum. So in general the extension principle does not "extend" a fuzzy set at all. It defuzzifies it into a binary distribution. In practice $n = 1 = b$ in fuzzy systems. As the number of nonzero fuzzy-rule outputs grows, the envelope locally grows to the constant value 1, the uniform distribution. Globally the output grows to a rectangular pulse.

Max combination ignores overlap in the fuzzy sets B'_j. Sum combination adds overlap to the peakedness of B. When the input changes slightly, the additive output B changes slightly. The max-combined output may ignore small input changes since for large sets of rules most change occurs in the overlap regions of the fuzzy sets B'_j. Here the overlap problem arises since the centroid tends to stay the same for small changes in input. But the centroid smoothly tracks changes in the fuzzy-set sum (1) as (5) shows.

Centroid defuzzification compounds the max problem. An integrable multivalued set function $m_B : Y \longrightarrow [0, 1]$ gives the centroidal output y_j or $F(x)$as

$$y_j = \frac{\int_{-\infty}^{\infty} y\, m_B(y)\, dy}{\int_{-\infty}^{\infty} m_B(y)\, dy}. \tag{3}$$

We can replace the integral in (3) with small discrete sums indexed only by the number of fuzzy sets that quantize the fuzzy variables [6]. See (5) below. This eliminates both the need to approximate the centroid and its computational burden. Digital VLSI hardware can then implement (3) or an analog VLSI chip can implement it with a simple follower-aggregator circuit [8]. In the limit as the number of combined output sets B'_j grows, centroidal defuzzification may tend to coincide with mode defuzzification since the centroid and mode coincide for a symmetric unimodal distribution.

III. Fuzzy Function Approximation

Sum inference reduces to the weighted sum

$$B = \sum_{j=1}^{m} w_j\, a_i^j\, B_j, \tag{4}$$

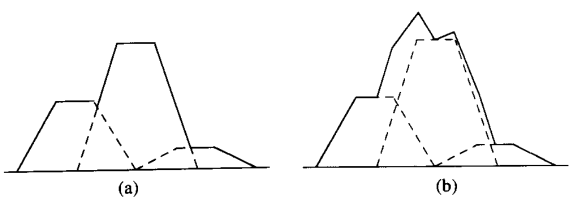

Fig. 4. Sum vs. max combination of rule output sets. The max envelope (a) tends toward a rectangle as the number of trapezoids grows. The sum envelope (b) may tend toward a Gaussian or symmetric unimodal curve. The centroid of the sum envelope in (b) must fall at or between the centroids of the first and third trapezoids. The approximation theorem depends on this fact. Centroids of max envelopes can fall outside these bounds.

where a_i^j is the degree to which input x_i belongs to fuzzy set A_j in the rule or patch $A_j \times B_j$. Additive fuzzy systems can approximate a function $f : X \longrightarrow Y$ by shrinking these patches in size and increasing them in number [7]. For the proof we can assume no learning and equi-credible rules: $w_1 = \cdots = w_m = 1$.

Equation (4) gives a simple approximation theorem. Suppose $f : X \longrightarrow Y$ is measurable and bounded. Then we can view B as a *simple* function if the fuzzy sets B_j are nonfuzzy sets [7]. We replace triangles, trapezoids, and other fuzzy sets with rectangles. Simple functions $s : X \longrightarrow Y$ map X into finitely many values of Y. A simple function equals a finite sum of weighted indicator functions. Choose the fit values a_i^j as the weights and let the nonfuzzy sets or rectangles partition X. Then [10] there is a simple function s_ε epsilon close to f on X. Boundedness of f ensures a uniform approximation. This shows that a large enough AI expert system can approximate any bounded measurable function and reminds us that fuzzy rules reduce in the bivalent case to expert-systems rules. It does not show that an additive fuzzy system with multivalued sets *converges* uniformly to f. So in practice the result gives no error bound. The problem arises because triangles, trapezoids, and other fuzzy sets need not converge uniformly to rectangles.

Uniform convergence holds if we work with continuity instead of measurability. The theorem below requires that $f : X \longrightarrow Y$ is continuous and that X is compact (closed and bounded) in $\mathcal{R}^n$. The theorem shows that in principle an additive fuzzy system with finite fuzzy rules can approximate any continuous function to any degree of accuracy. It includes the Gaussian result in [11] as a special case.

Theorem: An additive fuzzy system F uniformly approximates $f : X \longrightarrow Y$ if X is compact and f is continuous.

Proof: Pick any small constant $\varepsilon > 0$. We must show that $|F(x) - f(x)| < \varepsilon$ for all $x \in X$. X is a compact subset of R^n. $F(x)$ is the centroidal output (3) or (6) of the additive fuzzy system F in (4).

Continuity of f on compact X gives uniform continuity. So there is a fixed distance δ such that, for all x and z in X, $|f(x) - f(z)| < \varepsilon/4$ if $|x - z| < \delta$. We can construct a set of open cubes $M_1, \cdots, M_m$ that cover X and that have ordered overlap in their n coordinates so that each cube corner lies at the midpoint c_j of its neighbors M_j. Pick symmetric output fuzzy sets B_j centered on $f(c_j)$. So the centroid of B_j is $f(c_j)$.

Pick $u \in X$. Then by construction u lies in at most 2^n overlapping open cubes M_j. Pick any w in the same set of cubes. If $u \in M_j$ and $w \in M_k$, then for all $v \in M_j \cap M_k$: $|u - v| < \delta$ and $|v - w| < \delta$. Uniform continuity implies that $|f(u) - f(w)| \le |f(u) - f(v)| + |f(v) - f(w)| < \varepsilon/2$. So for cube centers c_j and $c_k, |f(c_j) - f(c_k)| < \varepsilon/2$.

Pick $x \in X$. Then x too lies in at most 2^n open cubes with centers c_j and $|f(c_j) - f(x)| < \varepsilon/2$. Along the kth coordinate of the range space R^p the kth component of the additive system centroid $F(x)$ lies as in (6) on or between the kth components of the centroids of the B_j sets. So, since $|f(c_j) - f(c_k)| < \varepsilon/2$ for all $f(c_j), |F(x) - f(c_j)| < \varepsilon/2$. Then $|F(x) - f(x)| \le |F(x) - f(c_j)| + |f(c_j) - f(x)| < \varepsilon/2 + \varepsilon/2 = \varepsilon$. $\square$

The proof may require symmetric output fuzzy sets B_j if correlation-minimum encodes [6] the rules $A_j \times B_j$. Correlation-product encoding does not require symmetry since $a_j B_j$ has the same centroid as B_j if $a_j > 0$.

The proof shows that we can replace the fuzzy sets A_j and B_j with finite discretizations or fit vectors $(a_1^j, \cdots, a_n^j)$ and $(b_1^j, \cdots, b_p^j)$. The discrete version of B_j must have a centroid at or close to the centroid of B_j. So we can always work with large-dimensional unit hypercubes and view fuzzy rules or patches as matrix mappings (or *fuzzy associative memories* [7]) between hypercubes or as points in even larger hypercubes.

The proof fails for max-combined sets B. The proof traps the centroidal output $C(B)$ or y_j in (3) between the centroids $C(B_1)$ and $C(B_m)$ if $C(B_1) \le C(B_2) \le \cdots \le C(B_m)$:.

$$C(B) = \frac{\sum_{j=1}^{m} A(B_j) C(B_j)}{\sum_{j=1}^{m} A(B_j)} \tag{5}$$

$$= \sum_{j=1}^{m} c_j C(B_j) \tag{6}$$

for volume or area $A(B_j) = \int_X m_B(x) dx$ and for *convex* area coefficients $c_1, \cdots, c_m$. Wang [11] renames these terms "fuzzy basis functions." The proof works for any combined output set $B = \phi(B_1, \cdots, B_m)$ such that $C(B_1) \le \phi \le C(B_m)$. In general the max combination $\bigvee_{j=1}^{m} B_j$ does not obey $C(B_1) \le C(\bigvee_j B_j) \le C(B_m)$. This inequality holds in the trivial case when sum = max. Since $x + y = \min(x, y) + \max(x, y), x + y = \max(x, y)$ iff $x = 0$ or $y = 0$ iff the combined sets $B_1, \cdots, B_m$ are disjoint. The proof also works for noncentroidal defuzzifiers $D(B)$ that obey $C(B_1) \le D(B) \le C(B_m)$. In general the supremum or max-membership defuzzifier does not obey this inequality.

IV. Fuzzy Systems as Conditional Expectations

How "fuzzy" is the approximation theorem? How "fuzzy" is a fuzzy system? Pure fuzziness stems from the overlap of a thing or set A and its opposite A^c. A is fuzzy iff $A \cap A^c \neq \emptyset$ [7]. Fuzzy systems need not be fuzzy in this pure sense. The finite area $A(B)$ of the output set B means that $A(B)$ normalizes B to give B' as the conditional *probability* density $p(y \mid x)$:

$$B' = \frac{B}{A(B)} \tag{7}$$

$$= p(Y \mid X = x). \tag{8}$$

We can *view* the fuzzy sets A_i and B_j as random sets or as loci of two-point conditional probability densities. The set degree $m_A(x)$ equals $P(A|X = x)$, the probability of event A given that random variable X takes on the domain or index value x. So we can *view* the fit value $m_{PM}(x)$ as the probability that X is Positive Medium if X is x. Then $X = PM$ means the random variable X takes on the entire *random set* Positive Medium as a random-set value. Then for input x each rule fires with some conditional probability. The system emits each output y with some conditional probability. The output equals the local average or conditional mean. If the output is the maximum probability value, the system computes a maximum a posteriori or MAP estimate.

The centroid (3) gives the same result as (8) and implies that the fuzzy system output $F(x)$ equals a realization of the conditional expectation:

$$F(x) = E[Y \mid X = x]. \tag{9}$$

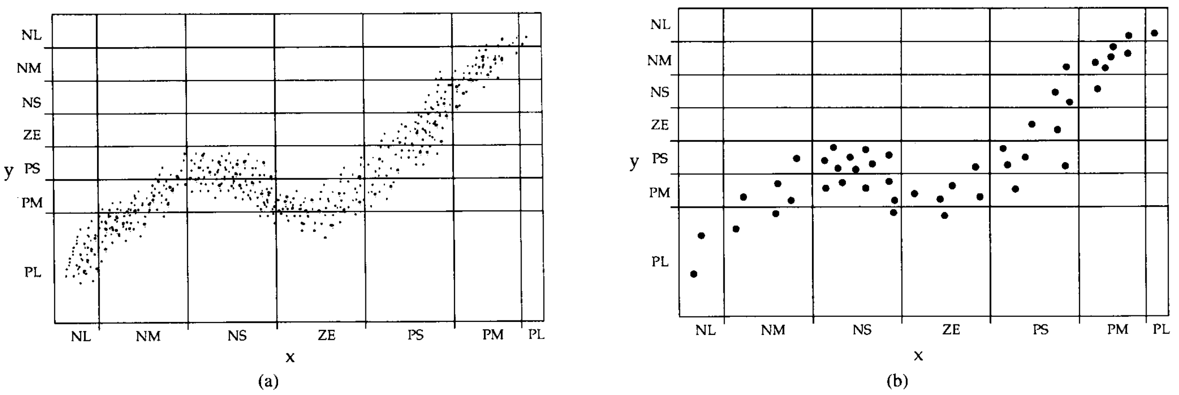

Fig. 5. Product space clustering with vector quantization. (a) Small dots show the observed sample data. (b) Larger dots shows how the quantization vectors distribute after learning. The weight w_{ij} of fuzzy rule "if $X = A_i$, then $Y = B_j$" grows as the $A_i \times B_j$ patch's count of quantization vectors grows.

On this view a fuzzy system is a probability system. It computes the random variable $E[Y \mid X]$ or $E[Y \mid X = NL, NM, \cdots, PL]$.

The summed patches give a *model-free* estimate of $f : X \longrightarrow Y$. This conditional mean is the mean-squared optimum among all nonlinear estimates of f that depend on the input "sets" or densities $NL, NM, \cdots, PL$. The approximation theorem says that additive fuzzy systems form an ε-bundle around f. They form an ε-bundle of conditional means around f.

V. Adaptive Fuzzy Systems

An *adaptive* fuzzy system is a fuzzy system that changes with time. The sets or rules change in shape or number. A learning system changes the fuzzy-rule weights $w_1, \cdots, w_m$ as it samples input-output data $(x_1, y_1), (x_2, y_2), \cdots$. In practice [7], we threshold the weights w_j to 0 or 1. If the weight w_j equals or exceeds the threshold, put $w_j = 1$ and add the fuzzy rule to the fuzzy system or "knowledge base." If w_j falls below the threshold, $w_j = 0$ and ignore the jth fuzzy rule and do not include it in the fuzzy system.

Adaptive fuzzy systems estimate fuzzy rules from sample data. This reduces to patch or cluster estimation in $X \times Y$ called *product space clustering* [7]. Neural or statistical clustering algorithms convert the sample data (x_i, y_i) into cluster estimates. An expert or physical process gives the sample data. Clustering algorithms search for the implicit fuzzy rules that the expert or physical process "used" to generate the data.

Vector quantization estimates clusters. A fixed set of quantization vectors $\mathbf{m}_1, \cdots, \mathbf{m}_v$ tracks the distribution of sample data. In neural systems each quantization vector $\mathbf{m}_j$ defines a fan-in synaptic vector to a neuron that competes in a winner-take-all network. The neural system *learns* or adapts if and only if the quantization/synaptic vector $\mathbf{m}_j$ *moves* in the input-output state space $X \times Y$. In effect each quantization vector $\mathbf{m}_j$ estimates a local cluster in $X \times Y$ and, in optimal mean-squared-error learning, converges to the cluster's centroid exponentially quickly [6]. Globally the quantization vectors estimate the unknown joint probability density $p(x, y)$ that gives rise to the observed data pairs (x_i, y_i). The v quantization vectors estimate the probability of any region C as n_c/v, the number of quantization vectors in C divided by the total number of quantization vectors.

Clusters of quantization vectors estimate fuzzy patches. At any time in the learning process each fuzzy patch $A_i \times B_j$ contains n_c quantization vectors. This gives rise to an adaptive histogram or frequency distribution of quantization vectors in the rs overlapping fuzzy patches or cells. In practice we may count a cell as sufficiently occupied $(w_j = 1)$ if it contains any quantization vectors. For in general sample data greatly outnumbers the fixed quantization vectors. In the extreme case, we can count each sample as a quantization vector [11]. This unbounded cases does not filter noise or compress the sample data. Fig. 5 shows product space clustering with vector quantization after learning has slowed or stopped. The small dots in Fig. 5(a) are the observed samples (x_i, y_i). The large dots in Fig. 5(b) are the quantization vectors $\mathbf{m}_j$. If we set a threshold of two quantization vectors per cell, then product space clustering yields 10 fuzzy rules or patches. Other learning schemes can change the cluster regions by grouping the covariance ellipsoids [1] of the quantization vectors and thus can change the shape of rules and sets.

The above uniform approximation theorem implies that a finite number of quantization vectors $\mathbf{m}_1, \cdots, \mathbf{m}_v$ can learn any sampled continuous function if the learning system samples enough function samples $(x_i, f(x_i))$ and if the quantization vectors converge to local centroids. Several types of competitive learning ensure this convergence [6]. In general, this requires learning with a prohibitively large number of quantization vectors. In the small-sample case it shows that we can shrink the appropriate fuzzy patches and increase their number as we sample more data and increase the number v of adaptive quantization vectors.

VI. Conclusion

A fuzzy system or approximator reduces to a graph cover with local averaging. That is not unique. An additive fuzzy system with Gaussian sets reduces to a radial basis network [3] and that too is but one of many graph coverings. The "fuzziness" or multivalence of sets comes into play when patches or output sets overlap. Nonfuzzy sets can also weight or average the overlap. A fuzzy system is unique in that it ties vague words like "small" and "medium" to the math of curves and fit vectors (points in unit cubes). So it ties natural language and commonsense rules to state-space geometry. But the "fuzzy" sets are equivalent to random sets or loci of two-point conditional probabilities. The fuzzy systems give a model-free estimate of some unknown conditional expectation $E[Y \mid X]$. The approximation power of fuzzy systems lies more in their model freedom than in their fuzzy interpretation. Nonfuzzy sets and rule patches also lead to model-free universal approximators.

Acknowledgment

The author thanks Dr. F. Watkins for insights on the analysis and application of fuzzy systems.

References

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

[2] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*. Orlando, FL: Academic Press, 1980.

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

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

[5] B. Kosko, "Fuzzy knowledge combination," *Int. J. Intell. Syst.*, vol. 1, pp. 293–320, 1986.

[6] ——, "Stochastic competitive learning," *IEEE Trans. Neural Netw.*, vol. 2, no. 5, pp. 522–529, Sept. 1991.

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

[8] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[9] W. Rudin, *Functional Analysis*. New York: McGraw-Hill, 1973.

[10] ——, *Real and Complex Analysis*, second ed. New York: McGraw-Hill, 1974.

[11] L. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 807–814, Sept. 1992.

[12] L. A. Zadeh, "Fuzzy sets," *Inform. Contr.*, vol. 8, pp. 338–353, 1965.

Optimal Centralized Algorithms for Store-and-Forward Deadlock Avoidance

J. Błażewicz, D. P. Bovet, J. Brzeziński, G. Gambosi, and M. Talamo

***Abstract*—In this brief contribution, a problem of deadlock avoidance in store-and-forward networks with at least two buffers per node is considered for fixed as well as dynamic routing. For both cases polynomial time, centralized deadlock avoidance algorithms are proposed and shown to be optimal in a sense of possible buffer utilization. When the number of buffers is equal to one for each node the problem is known to be NP-complete, thus, unlikely to admit a polynomial-time algorithm. The presented results may be also interesting for other applications, some massively parallel computer systems being one of the examples.**

***Index Terms*—Stored-and-forward networks, deadlock avoidance, centralized approach, buffer utilization, complexity analysis.**

I. Introduction

The concept of store-and-forward packet-switching is commonly used in computer and telecommunication networks as well as in some massively parallel systems such as hypercubes or transputer based machines. One of the most important issues arising in these systems is deadlock avoidance. To solve this problem several interesting distributed algorithms have been already proposed [2]–[13], [15], [16], [20], [21], [24], [26], [27], but till now this area offers a great opportunity for improvement ([25]). Distributed algorithms can be rather easily implemented in existing networks. However, it is also clear that none of them is optimal with respect to buffer utilization and to the number of safe states allowed. This is because they are too restrictive, since they cannot use the full knowledge of the network state which would be available in a centralized approach. Thus the latter, if optimal, can be useful as a valuable benchmark for all existing distributed algorithms. Moreover, this optimal approach may also form a base for a construction of new, more efficient, distributed algorithms. Such an adaptation of centralized algorithms to distributed context has been commonly used in practice, and many interesting solutions have been obtained in this way (see e.g., [1]–[19], [22]).

A more theoretical reason for studying centralized deadlock avoidance algorithms follows from the fact that a store-and-forward network may be considered as a special case of a centralized computer system, in which resources are required and then released in a prespecified order for each process (message) in the system [14]. Such a situation has not been investigated yet and the complexity of the corresponding deadlock avoidance problem remained open, except for networks containing nodes with one buffer. This last problem has been proved to be NP-complete [1].

The aim of the present paper follows from the above discussion and may be stated as finding an optimal (from the point of view of buffer utilization) and efficient (polynomial time) centralized algorithm for store-and-forward deadlock avoidance. In Section II, a model of a store-and-forward network is presented. Section III presents a centralized store-and-forward deadlock avoidance algorithm, assuming dynamic routing and more than one buffer per node. In Section IV, fixed routing is considered and corresponding results are presented.

Manuscript received July 8, 1992; revised January 13, 1993 and August 15, 1993.

J. Błażewicz and J. Brzeziński are with the Instytut Informatyki, Politechnika Poznańska—ul. Piotrowo 3a, 60-965 Poznań, Poland.

D. P. Bovet is with the Dipartimento di Scienza dell' Informazione, Università di Roma "La Sapienza", Via Salariz 113, 00198 Roma, Italy.

G. Gambosi is with the Università di Roma "Tor Vergata", Via della Ricerca Scientifica, 00173 Roma, Italy.

M. Talamo is with the Dipartimento di Informatica e Sistemistica, Universitàdi Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy.

IEEE Log Number 9402912.

II. The Model

A *store-and-forward transmission network* can be defined as quadruple $\mathrm{SF} = \{V, E, BN, B\}$, where $V = \{v_1, \cdots, v_n\}$ is a set of nodes, E is a set of (bidirectional) links, BN is a set of buffers and $B : V \mapsto N^+$ is a function such that $\sum_{i=1}^{n} B(v_i) =| BN |$, which associates with each node the number of buffers it contains. It is assumed that all buffers have the same sizes and that each buffer can be allocated to one message at a time.

Denote as $G_{\mathrm{SF}} = (V, E)$ the graph underlying network SF.

Messages have to be transmitted between contiguous nodes along *routes* R_i, i.e., finite paths $\langle v_{i_1}, \cdots, v_{i_k}\rangle$ in G_{SF}. Let us denote as *fixed routing* the case in which the route of a message is known in advance; let us moreover denote as *dynamic routing* the case in which for each message, only the next node along its route is given at any time. Message transmission is performed according to the following assumptions:

1) Message m originates in node $s(m)$ (the *source* of m) if there exists at least one *free* buffer in $v = s(m)$, i.e., a buffer not already assigned to any message. Such buffer can be assigned to m.
2) With each message m contained in node v, there is associated node *next*(m) contiguous to v, representing the next node to which m has to be transmitted.