Stochastic Competitive Learning

Bart Kosko

Abstract—**We examine competitive learning systems as stochastic dynamical systems. This includes continuous and discrete formulations of unsupervised, supervised, and differential competitive learning systems. These systems estimate an unknown probability density function from random pattern samples and behave as adaptive vector quantizers. Synaptic vectors, in feedforward competitive neural networks, quantize the pattern space and converge to pattern class centroids or local probability maxima. A stochastic Lyapunov argument shows that competitive synaptic vectors converge to centroids exponentially quickly and reduces competitive learning to stochastic gradient descent. Convergence does not depend on a specific dynamical model of how neuronal activations change. These results extend to competitive estimation of local covariances and higher-order statistics.**

I. Feedforward Multilayer Competitive Learning Systems

COMPETITIVE learning systems are usually feedforward multilayer neural networks. Neurons compete for the activation induced by randomly sampled pattern vectors $\boldsymbol{x} \in R^n$. An unknown probability density function, $p(\boldsymbol{x})$, characterizes the continuous distribution of random pattern vectors $\boldsymbol{x}$. A random n-vector, $\boldsymbol{m}_j$, of synaptic values fans in to each competing neuron. The synaptic vector $\boldsymbol{m}_j$ defines the jth column (or row) of the synaptic connection matrix M.

Competition selects which synaptic vector $\boldsymbol{m}_j$ the training sample $\boldsymbol{x}$ modifies. In practice competition dynamics reduce to metrical pattern matching. Competitive algorithms seldom use neuronal activation dynamics. The jth neuron "wins" at an iteration if the synaptic vector $\boldsymbol{m}_j$ is the closest, in Euclidean distance, of the m synaptic vectors to the random pattern $\boldsymbol{x}$ sampled at that iteration.

Some scaled form of the difference vector $\boldsymbol{x} - \boldsymbol{m}_j$ additively modifies the closest synaptic vector $\boldsymbol{m}_j$. Different scaling factors determine different competitive learning systems. A positive scaling factor "rewards" the winning jth neuron. The scaled synaptic vector $\boldsymbol{m}_j$ resembles the random sample $\boldsymbol{x}$ at least as much as the unmodified synaptic vector $\boldsymbol{m}_j$ resembled $\boldsymbol{x}$. A negative scaling factor "punishes" the jth neuron. The negatively scaled synaptic vector $\boldsymbol{m}_j$ disresembles $\boldsymbol{x}$ more than the unmodified synaptic vector did. Negative scaling tends to move misclassifying synaptic vectors in R^n out of regions of misclassification.

Autoassociative [12] competitive learning systems have two layers or fields of neurons. The input data field F_X of n neurons passes a randomly sampled pattern vector $\boldsymbol{x}$ forward through an n-by-m matrix M of synaptic values to m "competing" neurons in the competitive field F_Y. A symmetric m-by-m matrix W of within-field synaptic values describes the competition in F_Y. W has a positive main diagonal (or diagonal band) with negative or zero values elsewhere. In practice W has 1's down its main diagonal and -1's elsewhere. Autoassociative competitive learning systems recognize patterns. More generally, they estimate probability density functions $p(\boldsymbol{x})$.

Heteroassociative [8] competitive learning systems have three fields of neurons. The first and third fields, the input and output fields, sample the random-vector association $(\boldsymbol{x}, \boldsymbol{z})$. The second, or "hidden," field contains the competing neurons. If we concatenate the first and third fields into a single field of $n + p$ neurons, competitive learning proceeds as in the autoassociative case.

In practice heteroassociative competitive learning systems estimate only indirectly an unknown joint probability density function $p(\boldsymbol{x}, \boldsymbol{z})$. They directly estimate a sampled continuous function f: $R^n \rightarrow R^p$ from a large number of noisy random vector samples $(\boldsymbol{x}_i, \boldsymbol{z}_i)$. Implicitly the functional pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$ belong to high-probability regions of $R^n \times R^p$.

II. Competitive Learning as Adaptive Vector Quantization

Competitive learning systems adaptively quantize the pattern space R^n. The random synaptic vector $\boldsymbol{m}_j$ represents the local region about $\boldsymbol{m}_j$. Each synaptic vector $\boldsymbol{m}_j$ behaves as a *quantization vector*. The competitive learning system learns as synaptic vectors $\boldsymbol{m}_j$ change in response to randomly sampled training data. Geometrically, the systems learns if and only if some synaptic vector $\boldsymbol{m}_j$ moves in the pattern space R^n.

Competitive learning distributes the m synaptic vectors $\boldsymbol{m}_1, \cdots, \boldsymbol{m}_m$ in R^n to approximate the *unknown* probability density function $p(\boldsymbol{x})$ of the random pattern vector $\boldsymbol{x}$. Where the patterns $\boldsymbol{x}$ are dense or sparse, the synaptic vectors $\boldsymbol{m}_j$ tend to be dense or sparse. Different competitive learning, or *adaptive vector quantization* (AVQ), schemes distribute the synaptic vectors in different ways.

Manuscript received December 10, 1990; revised April 11, 1991. This work was supported by the Air Force Office of Scientific Research (AFOSR-88-0236) and by a grant from the Shell Oil Corporation.

The author is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-0272.

IEEE Log Number 9100945.

We would not need to learn if we knew $p(\boldsymbol{x})$. Numerical techniques [4], [27] could directly determine pattern clusters or classes, centroids, and class boundaries.

All observed patterns are realizations of a single *random vector* $\boldsymbol{x}$ [1], [24]. The random vector $\boldsymbol{x}$ can be interpreted as n ordered scalar random variables: $\boldsymbol{x} = (x_1, \cdots, x_n)$. The function $p(\boldsymbol{x})$ describes the *occurrence probability* of the patterns infinitesimally surrounding $\boldsymbol{x}$.

Pattern clusters or classes are subsets of R^n. Some pattern classes are more probable than others. We suppose k subsets or *decision classes* $D_1, \cdots, D_k$ partition the pattern space:

$$R^n = D_1 \cup \cdots \cup D_k$$

and

$$D_i \cap D_j = \varnothing \qquad \text{if } i \neq j. \tag{1}$$

The distinction between supervised and unsupervised pattern learning depends on the available information. In both cases we do not know the probability density function $p(\boldsymbol{x})$. Thus we use adaptive techniques instead of, say, numerical optimization or calculus-of-variation [4] techniques. Learning provides only a means to a computational end.

Supervised learning requires more information than does unsupervised learning. Unsupervised learning uses minimal information. Pattern learning is *supervised* if the learning algorithm depends on pattern-class information, on the decision classes $D_1, \cdots, D_k$. The system "knows" that $\boldsymbol{x} \in D_i$ and that $\boldsymbol{x} \notin D_j$ for all $j \neq i$. *Unsupervised* learning algorithms do not use class-membership information. They use unlabeled pattern samples. Supervision allows us to compute an error measure or vector. The simplest error measure equals the desired outcome minus the actual outcome. The error measure guides the learning process, often in a feedback error-correction procedure, as the system performs stochastic gradient descent on the unknown mean-squared-error surface.

III. AVQ Class Probability Estimation

The partition property (1) implies that $p(D_1) + \cdots + p(D_k) = 1$ since $p(R^n) = 1$. The *class probability* $p(D_i)$ integrates $p(\boldsymbol{x})$ over D_i:

$$p(D_i) = \int_{D_i} p(\boldsymbol{x})\, d\boldsymbol{x} \tag{2}$$

$$= E[I_{D_i}]. \tag{3}$$

The integral in (2) is an n-dimensional multiple integral. $E[x]$ denotes the mathematical expectation of random variable x. The function $I_S: R^n \to \{0, 1\}$ defines the *indicator* function of set S. $I_S(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in S$, $I_S(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \notin S$. In the probabilistic setting the indicator function I_S is random (Borel measurable [1]), and hence a random variable. Pattern $\boldsymbol{x}$ belongs to exactly one decision class—with probability 1. With probability 0, pattern $\boldsymbol{x}$ can lie on the border of two or more decision classes. Technically $p(\boldsymbol{x}) = 0$ for every $\boldsymbol{x}$ in R^n.

A *uniform partition* gives $p(D_i) = 1/k$ for each decision class D_i in the partition. Uniform partitions are not unique. Some vector quantization schemes attempt to adaptively partition R^n into a uniform partition. Then it should be equally likely that a pattern sample $\boldsymbol{x}$ drawn at random (according to $p(\boldsymbol{x})$) from R^n was drawn from any one of the k decision classes D_i. Then each competing neuron should "win" with the same frequency. Researchers [2], [25] have proposed supervised modifications to competitive learning algorithms that force the competing neurons to win with the same win rate. Economy motivates these modifications: fewer neurons can estimate a sampled continuous function.

Nonuniform partitions are more informative than uniform partitions. They also occur more frequently when we estimate an unknown probability density function $p(\boldsymbol{x})$. When the number m of competing neurons is less than the number k of distinct pattern classes, when $m < k$, some neurons win more frequently than others win. If $p(D_i) > p(D_j)$, the competing neuron that codes for D_i tends to win more frequently than the neuron that codes for D_j. Equivalently, more sample patterns $\boldsymbol{x}$ tend to be closer in Euclidean distance to the corresponding synaptic vector, say $\boldsymbol{m}_i$, that quantizes D_i than to the synaptic vector $\boldsymbol{m}_j$ that quantizes D_j. Below we show that $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ tend to arrive at the respective centroids of D_i and D_j and wander about them in a Brownian motion. Centroids minimize the mean-squared error of vector quantization [3], [20], [23].

In general there are more competing neurons than decision classes, $m > k$. For we can always add neurons to the competitive learning system. Then if $p(D_i) > p(D_j)$, D_i tends to contain more synaptic vectors than D_j contains. In principle all the neurons with synaptic vectors in D_i can have the same win rates. But since metrical classification determines which neuron wins, neurons with synaptic vectors nearer the centroid of D_i tend to win more frequently.

The number of synaptic vectors in decision class D_i gives a nonparametric estimate of the class probability $p(D_i)$: $p(D_i) = n_i/m$, with n_i denoting the number of synaptic vectors in D_i. The quantizing synaptic vectors estimate nonparametrically the probability density function $p(\boldsymbol{x})$. The user need make no probability assumptions about the observed training samples. For any subset or volume $V \subset R^n$, the distributed synaptic vectors estimate the volume probability $p(V)$ as the ratio

$$p(V) = \frac{n_V}{m}. \tag{4}$$

Here n_V denotes the number of synaptic vectors $\boldsymbol{m}_j$ in V, and m equals the total number of synaptic vectors. In the extreme case (4) gives $p(R^n) = 1$, and $p(\varnothing) = 0$.

IV. Deterministic Competitive Learning Laws

Competitive learning means *learn only if win*. Losing neurons, or rather their synaptic fan-in vectors, do not learn. They also do not forget what they have already learned. This produces a nondistributed representation [6]. The synapses in a synaptic vector $\boldsymbol{m}_j$ become "grandmother" synapses. Each synaptic element m_{ij} behaves as a discrete memory unit, as in a random access memory.

In contrast, classical Hebbian [7] or correlation learning distributes learned pattern-vector information across the entire synaptic connection matrix M. But a Hebbian system forgets learned pattern information as it learns new pattern information.

The simplest deterministic *competitive learning* [6], [17], [25] law takes the form

$$\dot{m}_{ij} = S_j(y_j)[S_i(x_i) - m_{ij}]. \tag{5}$$

where $\dot{m}_{ij}$ denotes the time derivative of the synaptic value of the directed axonal connection from the ith neuron in the input field F_X to the jth neuron in the output or competitive field F_Y. The n-by-m matrix M consists of the m_{ij} values. The jth column of M equals the fan-in synaptic vector $\boldsymbol{m}_j = (m_{1j}, \cdots, m_{nj})$. We can add or multiply scaling constants in (5) as desired.

In contrast, the *signal Hebbian learning* [6], [15] takes the form

$$\dot{m}_{ij} = -m_{ij} + S_i(x_i)S_j(y_j). \tag{6}$$

Equations (5) and (6) differ in how they forget. All learning requires some forgetting. The competitive signal S_j in (5) nonlinearly scales the decay or forget term $-m_{ij}$. In practice [8], [12], [13], [25], the competitive signal S_j approximates or equals a zero-one or binary threshold function. Winners forget, losers remember.

Input field F_X contains n neurons. Field F_Y contains m competing neurons. Each neuron in F_X or F_Y defines a function that transduces its real-valued *activation* $x_i(t)$ or $y_j(t)$ into a bounded *signal* $S_i(x_i(t))$ or $S_j(y_j(t))$ at time t. In principle the activation functions, or membrane potential differences, x_i and y_j can be unbounded.

In feedback networks [17], the signal functions S_i and S_j are usually bounded and monotone nondecreasing. Then they have nonnegative activation derivatives S_i' and S_j'. The popular logistic and hyperbolic-tangent signal functions have positive derivatives. Such signal functions S_i and S_j strictly increase:

$$S_i' = \frac{dS_i}{dx_i} > 0 \qquad S_j' = \frac{dS_j}{dy_j} > 0. \tag{7}$$

For instance, the logistic signal function $S(x) = (1 - e^{-cx})^{-1}$ with scale constant $c > 0$ has the increasing activation derivative $S' = cS(1 - S) > 0$. The logistic signal function rapidly approaches a binary threshold function as c increases.

In competitive learning the F_Y signal functions S_j often approximate or equal binary threshold functions. $S_j(t) = 1$ if the jth competing neuron in F_Y wins the competition for activation at time t. $S_j(t) = 0$ if the jth neuron loses.

The F_X signal functions S_i are usually linear in feedforward systems: $S_i(x_i) = x_i$. Then the sample pattern $\boldsymbol{x} = (x_1, \cdots, x_n)$ directly activates the system as the F_X signal state vector $S_X(\boldsymbol{x})$, since $S_X(\boldsymbol{x}) = \boldsymbol{x}$. So in practice the competitive learning law (5) approximates

$$\dot{m}_{ij} = I_{D_j}(\boldsymbol{x})[x_i - m_{ij}] \tag{8}$$

where I_{D_j} denotes the zero-one indicator function of decision class D_j. As discussed below we assume synaptic vector $\boldsymbol{m}_j$ codes for class D_j, perhaps by hovering randomly about the centroid of D_j.

Kohonen's recent [12] *supervised competitive learning* (SCL) law provides a reinforced version of (5):

$$\dot{m}_{ij} = r_j(\boldsymbol{x})S_j[x_i - m_{ij}] \tag{9}$$

where S_j equals a binary threshold function determined metrically. $S_j = 1$ if $\boldsymbol{x}$ is closer in Euclidean distance to the synaptic vector $\boldsymbol{m}_j$ than to all other synaptic vectors $\boldsymbol{m}_i$. The new term r_j in (8) denotes the *reinforcement function* of the jth competing neuron in F_Y. The function r_j rewards when $r_j(\boldsymbol{x}) = 1$, and punishes when $r_j(\boldsymbol{x}) = -1$.

The class membership of the pattern sample $\boldsymbol{x}$ determines the reinforcement signal $r_j(\boldsymbol{x})$. So (9) is a supervised competitive learning law. The signal $r_j(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in D_j$ and if the jth neuron wins or correctly "classifies" $\boldsymbol{x}$—if $I_{D_j}(\boldsymbol{x}) = S_j(\boldsymbol{x}) = 1$; $r_j(\boldsymbol{x}) = -1$ if the winning jth neuron misclassifies the sample pattern $\boldsymbol{x}$. Misclassification means the jth neuron wins but $\boldsymbol{x} \in D_i$, or $\boldsymbol{x} \notin D_j$, for some $i \neq j$. Then $I_{D_j}(\boldsymbol{x}) = 0$, but $I_{D_i}(\boldsymbol{x}) = S_j = 1$. Since, with probability 1, $\boldsymbol{x}$ belongs to exactly one decision class, the reinforcement function reduces to a difference of decision-class indicator functions:

$$r_j = I_{D_j} - \sum_{i \neq j} I_{D_i}. \tag{10}$$

So r_j depends explicitly on the decision-class boundaries.

The unsupervised *differential competitive learning* [17] (DCL) law modulates the vector difference $\boldsymbol{x} - \boldsymbol{m}_j$ with the instantaneous competitive *win rate* $\dot{S}_j$:

$$\dot{m}_{ij} = \dot{S}_j(y_j)[S_i(x_i) - m_{ij}]. \tag{11}$$

The signal velocity $\dot{S}_j$ decomposes as $S_j' \dot{y}_j$ by the chain rule. The idea is *learn only if change*. The signal velocity in (11) behaves in sign much as the reinforcement function behaves in (9). The signal velocity $\dot{S}_j(t)$ is positive or negative according to whether the jth competing neuron's win status increases or decreases at time t. The signal velocity does not depend on the decision-class indicator functions. So the DCL law (11) is unsupervised.

In practice the F_X signal function S_i is linear. Then simulations [13] show that the DCL law and Kohonen's SCL law (9) behave similarly. The DCL synaptic vectors tend to converge to decision class centroids at least as fast as SCL synaptic vectors converge and tend to wander randomly in a Brownian motion about the centroids with less variance than the SCL synaptic vectors wander. The competitive learning laws (5) and (9) ignore the instantaneous win-rate information that the signal velocity provides in

(11). The signal derivative also produces delta-modulation effects [18].

The *pulse-coded* [5], [17] signal function S_j gives an exponentially weighted average of binary pulses:

$$S_j(t) = \int_{-\infty}^{t} y_j(s)e^{s-t}\,ds \tag{12}$$

where $y_j(t) = 1$ if a pulse is present at time t, and $y_j(t) = 0$ if no pulse is present. Then the signal velocity reduces to the locally available difference

$$\dot{S}_j(t) = y_j(t) - S_j(t). \tag{13}$$

The velocity-difference representation (13) directly computes the signal velocity. This allows biological, or silicon, synapses to modify their values in real time with signal velocity information. Biological neurons transmit and receive pulse trains, not real-valued sigmoidal outputs. They can more easily detect, amplify, and emit pulses than multivalued signals. Equation (13) shows that much of the time the arriving pulse $y_j(t)$ indicates the instantaneous sign of the signal velocity.

The pulse-coded differential competitive law approximates [17] the classical competitive law (5), as we can see by substituting (13) into (11) and expanding terms. A related approximation of the signal Hebb law (6) occurs when (13) eliminates a product of signal velocities in a comparable *differential Hebbian learning* [10], [11], [14], [16], [17] law.

V. Stochastic Competitive Learning Laws and Algorithms

Stochastic competitive learning laws are stochastic differential equations. They describe how synaptic random processes change as a function of other random processes. Their solution defines a synaptic random process [26].

The deterministic competitive learning laws (5), (9), and (11) are simple stochastic differential equations if the signal terms $S_i(x_i(t))$ are random variables at each time t. This occurs when the sample vectors $\boldsymbol{x}$ are random samples, realizations of the pattern random-vector process $\boldsymbol{x}$: $R^n \rightarrow R^n$. The randomness in the vector components x_i induces randomness in the signal function S_i and thus in the synaptic vectors $\boldsymbol{m}_j$. In general, at each time t, each term in a stochastic differential equation represents a random variable.

A stochastic differential equation also arises when we "add" random noise to a differential equation. The randomness in the noise process induces randomness in the dependent variables.

The stochastic competitive learning law, in vector notation, takes the form

$$d\boldsymbol{m}_j = S_j(y_j)[\boldsymbol{S}(\boldsymbol{x}) - \boldsymbol{m}_j]\,dt + d\boldsymbol{B}_j. \tag{14}$$

S_j now denotes a steep competitive signal process that takes values in [0, 1], and $\boldsymbol{S}(\boldsymbol{x}) = (S_1(x_1), \cdots, S_n(x_n))$ for random pattern $\boldsymbol{x}$. $\boldsymbol{B}_j$ denotes a Brownian motion diffusion process.

The pseudo derivative [26] of $\boldsymbol{B}_j$ equals, in the mean-squared sense [1], the zero-mean Gaussian *white noise* process $\boldsymbol{n}_j$. The noise process $\boldsymbol{n}_j$ is zero-mean, $E[n_{ij}] = 0$, has finite variance, and is independent of the "signal" term $S_j(y_j)[\boldsymbol{S}(\boldsymbol{x}) - \boldsymbol{m}_j]$. Then we can write competitive learning laws in less rigorous, more intuitive "noise" notation [17]. For example, (14) becomes

$$\dot{\boldsymbol{m}}_j = S_j(y_j)[\boldsymbol{S}(\boldsymbol{x}) - \boldsymbol{m}_j] + \boldsymbol{n}_j. \tag{15}$$

In practice S_j approximates a binary threshold function and behaves as the class indicator function I_{D_j}. Formally we assume that competitive signal functions S_j estimate the indicator function I_{D_j} of the local jth pattern class: $S_j(y_j) \approx I_{D_j}(\boldsymbol{x})$. This property allows us to ignore the complicated nonlinear dynamics of the within-field F_Y competition. The equilibrium and convergence results below depend on this approximation. The property is approximate because the sampled decision classes may vary slightly—or wildly in rare cases—as the synapses converge to their equilibrium values. The F_X signal processes S_i are linear. So (15) reduces to

$$\dot{\boldsymbol{m}}_j = I_{D_j}(\boldsymbol{x})[\boldsymbol{x} - \boldsymbol{m}_j] + \boldsymbol{n}_j. \tag{16}$$

The stochastic supervised competitive learning (SCL) law takes the form

$$\dot{\boldsymbol{m}}_j = r_j(\boldsymbol{x})S_j(y_j)[\boldsymbol{x} - \boldsymbol{m}_j] + \boldsymbol{n}_j. \tag{17}$$

The stochastic version of the differential competitive learning (DCL) law takes the form

$$\dot{\boldsymbol{m}}_j = \dot{S}_j(y_j)[x - m_j] + \boldsymbol{n}_j \tag{18}$$

or, in pulse-coded form,

$$\dot{\boldsymbol{m}}_j = [y_j(t) - S_j(t)][\boldsymbol{x} - \boldsymbol{m}_j] + \boldsymbol{n}_j. \tag{19}$$

The pulse process y_j defines a random *point process*, perhaps Poisson in nature. Thus (19) defines a doubly stochastic synaptic model.

For practical implementation we can write the above three stochastic competitive learning models as the following stochastic difference equations. (Historically, Tsypkin [27] derived the "winning" parts of the UCL algorithm and, with his adaptive Bayes approach, the SCL algorithm in a nonneural context. MacQueen [19] called the UCL difference equation *adaptive k-means clustering* in the mid-1960's.) The stochastic difference equations do not include an independent noise term. The noise processes in the above stochastic differential equations model unmodeled effects, round-off errors, or sample-size defects.

VI. Competitive AVQ Algorithms

1) Initialize synaptic vectors: $\boldsymbol{m}_i(0) = \boldsymbol{x}(i)$, $i = 1, \cdots, m$. This distribution-dependent initialization scheme avoids the problems that occur when all synaptic vectors initially equal, say, the null vector and that require quasi-supervised support algorithms [2].

2) For random sample $\boldsymbol{x}(t)$, find the closest ("winning") synaptic vector $\boldsymbol{m}_j(t)$:

$$\|\boldsymbol{m}_j(t) - x(t)\| = \min_i \|\boldsymbol{m}_i(t) - \boldsymbol{x}(t)\| \tag{20}$$

where $\|\boldsymbol{x}\|^2 = x_1^2 + \cdots + x_n^2$ gives the squared Euclidean norm of $\boldsymbol{x}$.

3) Update the winning synaptic vector(s) $\boldsymbol{m}_j(t)$ by the UCL, SCL, or DCL learning algorithm.

Unsupervised Competitive Learning (UCL):

$$\begin{aligned} \boldsymbol{m}_j(t+1) &= \boldsymbol{m}_j(t) + c_t[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)] \\ \boldsymbol{m}_i(t+1) &= \boldsymbol{m}_i(t) \qquad \text{if } i \neq j \end{aligned} \tag{21}$$

where, as in stochastic approximation [27], $\{c_t\}$ defines a slowly decreasing sequence of learning coefficients. For instance, $c_t = 0.1\ (1 - t/10\,000)$ for 10 000 samples $\boldsymbol{x}(t)$.

Supervised Competitive Learning (SCL):

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + c_t r_j(\boldsymbol{x}(t))[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)] \tag{22}$$

$$= \begin{cases} \boldsymbol{m}_j(t) + c_t[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)] & \text{if } \boldsymbol{x} \in D_j \\ \boldsymbol{m}_j(t) - c_t[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)] & \text{if } \boldsymbol{x} \notin D_j. \end{cases} \tag{23}$$

Equation (23) rewrites the reinforcement function r_j into the update algorithm.

Differential Competitive Learning (DCL):

$$\begin{aligned} \boldsymbol{m}_j(t+1) &= \boldsymbol{m}_j(t) + c_t \Delta S_j(y_j(t))[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)] \\ \boldsymbol{m}_i(t+1) &= \boldsymbol{m}_i(t) \qquad \text{if } i \neq j. \end{aligned} \tag{24}$$

$\Delta S_j(y_j(t))$ denotes the time change of the jth neuron's competitive signal $S_j(y_j)$ in the competition field F_Y:

$$\Delta S_j(y_j(t)) = S_j(y_j(t+1)) - S_j(y_j(t)). \tag{25}$$

In practice [13] we often use only the sign of the signal difference (25) or sgn $[\Delta y_j]$, the sign of the activation difference. We can update the F_Y neuronal activations y_j with an *additive* model:

$$y_j(t+1) = y_j(t) + \sum_{i=1}^{n} S_i(x_i)m_{ij}(t) + \sum_{k=1}^{m} S_k(y_k)w_{kj}. \tag{26}$$

The fixed competition matrix W defines a symmetric *lateral inhibition* topology within F_Y. In the simplest case, $w_{jj} = 1$, and $w_{ij} = -1$ for distinct i and j.

VII. Stochastic Equilibrium and Convergence

Competitive synaptic vectors $\boldsymbol{m}_j$ converge to decision class centroids. The centroids may correspond to local maxima of the sampled but unknown probability density function $p(\boldsymbol{x})$.

In general, when there are more synaptic vectors than probability maxima, the synaptic vectors cluster about local probability maxima. Comparatively few synaptic vectors may actually arrive at pattern-class centroids. We consider only convergence to centroids. We can view any local connected patch of the sample space R^n as a candidate decision class. Each synaptic vector samples such a local patch and converges to its centroid. Approximation $S_j \approx I_{D_j}$ codifies this interpretation.

We first prove the AVQ centroid theorem: If a competitive AVQ system converges, it converges to the centroid of the sampled decision class. We prove this equilibrium theorem only for unsupervised competitive learning, but argue that it holds for supervised and differential competitive learning in many cases of practical interest.

Next we use a Lyapunov argument to reprove and extend the AVQ centroid theorem to the AVQ convergence theorem: Stochastic competitive learning systems are asymptotically stable, and synaptic vectors converge to centroids. So competitive AVQ systems always converge, and converge exponentially fast. Both results hold with probability 1. Historically, MacQueen [19] showed that estimators trained with the discrete "competitive" or k-means clustering algorithm (21) converged to centroids. MacQueen did not show exponential convergence or account for noise disturbances.

AVQ Centroid Theorem:

$$\text{Prob}\ (\boldsymbol{m}_j = \bar{\boldsymbol{x}}_j) = 1 \text{ at equilibrium.} \tag{27}$$

The centroid $\bar{\boldsymbol{x}}_j$ of decision class D_j equals its probabilistic center of mass:

$$\bar{\boldsymbol{x}}_j = \frac{\int_{D_j} \boldsymbol{x}p(\boldsymbol{x})\,d\boldsymbol{x}}{\int_{D_j} p(\boldsymbol{x})\,d\boldsymbol{x}} \tag{28}$$

$$= E[\boldsymbol{x}|\boldsymbol{x} \in D_j]. \tag{29}$$

The random vector $E[\boldsymbol{x}|\cdot]$, the conditional expectation, is a function of Borel measurable [1] subsets D_j of R^n.

Proof: Suppose the jth neuron in F_Y wins the activation competition during the training interval. Suppose the jth synaptic vector $\boldsymbol{m}_j$ codes for decision class D_j: $S_j \approx I_{D_j}$. Suppose the synaptic vector has reached equilibrium:

$$\dot{\boldsymbol{m}}_j = \boldsymbol{0} \tag{30}$$

which holds with probability 1 (or in the mean-square sense, depending on how we define the stochastic differentials). Take expectations of both sides of (30), use the zero-mean property of the noise process, eliminate the synaptic velocity vector $\dot{\boldsymbol{m}}_j$ with the competitive law (16), and expand to give

$$\boldsymbol{0} = E[\dot{\boldsymbol{m}}_j] \tag{31}$$

$$= \int_{R^n} I_{D_j}(\boldsymbol{x})(\boldsymbol{x} - \boldsymbol{m}_j)p(\boldsymbol{x})\,d\boldsymbol{x} + E[\boldsymbol{n}_j] \tag{32}$$

$$= \int_{D_j} (\boldsymbol{x} - \boldsymbol{m}_j) p(\boldsymbol{x})\, d\boldsymbol{x} \tag{33}$$

$$= \int_{D_j} \boldsymbol{x} p(\boldsymbol{x})\, d\boldsymbol{x} - \boldsymbol{m}_j \int_{D_j} p(\boldsymbol{x})\, d\boldsymbol{x} \tag{34}$$

since (30) implies that $\boldsymbol{m}_j$ is constant with probability 1. Solving for the equilibrium synaptic vector $\boldsymbol{m}_j$ gives the centroid $\bar{\boldsymbol{x}}_j$ in (28). Q.E.D.

In general the AVQ centroid theorem concludes that at equilibrium the *average* synaptic vector $E[\boldsymbol{m}_j]$ equals the jth centroid $\bar{\boldsymbol{x}}_j$ at equilibrium:

$$E[\boldsymbol{m}_j] = \bar{\boldsymbol{x}}_j. \tag{35}$$

The equilibrium synaptic vector $\boldsymbol{m}_j$ vibrates in a Brownian motion about the constant centroid $\bar{\boldsymbol{x}}_j$. The vector $\boldsymbol{m}_j$ equals $\bar{\boldsymbol{x}}_j$ on average at each postequilibration instant. Simulated [13] competitive synaptic vectors have exhibited such Brownian wandering about centroids.

Synaptic vectors learn noise as well as signal. So they vibrate at equilibrium. The independent additive noise process $\boldsymbol{n}_j$ in (16) drives the random vibration. The steady-state condition (30) models the rare event that noise cancels signal. In general it models stochastic equilibrium in the absence of additive noise.

The equality

$$\dot{\boldsymbol{m}}_j = \boldsymbol{n}_j \tag{36}$$

models stochastic equilibrium in the presence of additive independent noise.

Taking expectations of both sides of (36) still gives (31), since the noise process $\boldsymbol{n}_j$ is zero-mean, and the argument proceeds as before. Taking a second expectation in (33) and using (31) gives (35).

The AVQ centroid theorem applies to the stochastic SCL law (17) because the algorithm picks winners metrically with the nearest-neighbor criterion (20). The reinforcement function r_j in (10) reduces to $r_j(\boldsymbol{x}) = -I_{D_i}(\boldsymbol{x}) = -1$ when the jth neuron continually wins for random samples $\boldsymbol{x}$ from class D_i. This tends to occur once the synaptic vectors have spread out in R^n, and D_i is close, usually contiguous, to D_j. Then $\boldsymbol{m}_j$ converges to $\bar{\boldsymbol{x}}_j$, the centroid of D_j, since the steady state condition (30) or (36) removes the scaling constant -1 that then appears in (33).

This argument holds only approximately when, in the exceptional case, $\boldsymbol{m}_j$ repeatedly misclassifies patterns $\boldsymbol{x}$ from several classes D_k. Then the difference of indicator functions in (10) replaces the single indicator function I_{D_j} in (32). The resultant equilibrium $\boldsymbol{m}_j$ equals a more general ratio than the centroid. For then we must integrate the density $p(\boldsymbol{x})$ over R^n, not just over D_j.

The AVQ centroid theorem applies similarly to the stochastic DCL law (18). A positive or negative factor scales the difference $\boldsymbol{x} - \boldsymbol{m}_j$. If, as in practice and in (24), a constant approximates the scaling factor, the steady-state condition (30) or (36) removes the constant from (33) and $\boldsymbol{m}_j$, or $E[\boldsymbol{m}_j]$, estimates the centroid $\bar{\boldsymbol{x}}_j$.

The integrals in (31)–(34) are *spatial* integrals over R^n or subsets of R^n. Yet in the discrete UCL, SCL, and DCL algorithms, the recursive equations for $\boldsymbol{m}_j(t+1)$ define *temporal* integrals over the training interval.

The spatial and temporal integrals are approximately equal. The discrete random samples $\boldsymbol{x}(0), \boldsymbol{x}(1), \boldsymbol{x}(2), \cdots$ partially enumerate the continuous distribution of equilibrium realizations of the random vector $\boldsymbol{x}$. The time index in the discrete algorithms approximates the "spatial index" underlying $p(\boldsymbol{x})$. So the recursion $\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + \cdots$ approximates the averaging integral. We sample patterns one at a time. We integrate them all at a time.

The AVQ centroid theorem assumes that stochastic convergence occurs. Synapses converge trivially for continuous deterministic competitive learning, at least in feedforward networks. Convergence is not trivial for stochastic competitive learning in noise.

The AVQ convergence theorem below ensures exponential convergence. The theorem does not depend on how the F_Y neurons change in time provided $S_j \approx I_{D_j}$ holds. The proof uses a stochastic Lyapunov function L. The strictly decreasing deterministic Lyapunov function $E[L]$ replaces [17] the random Lyapunov function L.

A strictly decreasing Lyapunov function yields asymptotic stability [22]. Then the real parts of the eigenvalues of the system Jacobian matrix are strictly negative, and locally the nonlinear system behaves linearly. Synaptic vectors converge [9] exponentially quickly to equilibrium points—to pattern-class centroids—in R^n. Technically, nondegenerate Hessian matrix conditions must also hold. Otherwise some eigenvalues may have zero real parts.

AVQ Convergence Theorem: Competitive synaptic vectors converge exponentially quickly to pattern-class centroids.

Proof: Consider the random quadratic form L:

$$L = \tfrac{1}{2} \sum_i^n \sum_j^m (x_i - m_{ij})^2. \tag{37}$$

Note that if $\boldsymbol{x} = \bar{\boldsymbol{x}}_j$ in (37), then, with probability 1, $L > 0$ if any $\boldsymbol{m}_j \neq \bar{\boldsymbol{x}}_j$, and $L = 0$ iff $\boldsymbol{m}_j = \bar{\boldsymbol{x}}_j$ for every $\boldsymbol{m}_j$.

The pattern vectors $\boldsymbol{x}$ do not change in time. (The following argument still holds if the pattern vectors $\boldsymbol{x}$ change slowly relative to synaptic changes.) This simplifies the stochastic derivative of L:

$$\dot{L} = \sum_i \frac{\partial L}{\partial x_i} \dot{x}_i + \sum_i \sum_j \frac{\partial L}{\partial m_{ij}} \dot{m}_{ij} \tag{38}$$

$$= \sum_i \sum_j \frac{\partial L}{\partial m_{ij}} \dot{m}_{ij} \tag{39}$$

$$= -\sum_i \sum_j (x_i - m_{ij}) \dot{m}_{ij} \tag{40}$$

$$= -\sum_i \sum_j I_{D_j}(x)(x_i - m_{ij})^2 - \sum_i \sum_j (x_i - m_{ij}) n_{ij}. \tag{41}$$

L equals a random variable at every time t. $E[L]$ equals a deterministic number at every t. So we use the average

$E[L]$ as a Lyapunov function for the stochastic competitive dynamical system. For this we must assume sufficient smoothness to interchange the time derivative and the probabilistic integral—to bring the time derivative "inside" the integral. Then the zero-mean noise assumption, and the independence of the noise process $\boldsymbol{n}_j$ with the "signal" process $\boldsymbol{x} - \boldsymbol{m}_j$, give

$$\dot{E}[L] = E[\dot{L}] \tag{42}$$

$$= -\sum_j \int_{D_j} \sum_i (x_i - m_{ij})^2 p(\boldsymbol{x})\, d\boldsymbol{x}. \tag{43}$$

So, on average by the learning law (16), $\dot{E}[L] < 0$ iff any synaptic vector $\boldsymbol{m}_j$ moves along its trajectory. So the competitive AVQ system is *asymptotically* stable [9], [22] and in general converges exponentially quickly to equilibria.

Suppose $\dot{E}[L] = 0$. Then every synaptic vector has reached equilibrium and is constant (with probability 1) if (30) holds. Then [24], since $p(\boldsymbol{x})$ is a nonnegative weight function, the weighted integral of the learning differences $x_i - m_{ij}$ must also equal 0:

$$\int_{D_j} (\boldsymbol{x} - \boldsymbol{m}_j) p(\boldsymbol{x})\, d\boldsymbol{x} = \boldsymbol{0} \tag{44}$$

in vector notation. Equation (44) is identical to (33). So, with probability 1, equilibrium synaptic vectors equal centroids. More generally, as discussed above, (35) holds. Average equilibrium synaptic vectors are centroids: $E[\boldsymbol{m}_j] = \bar{\boldsymbol{x}}_j$. Q.E.D.

The sum of integrals (43) defines the total mean-squared error of vector quantization for the partition $D_1, \cdots, D_k$. The vector integral in (44) equals the gradient of $\dot{E}[L]$ with respect to $\boldsymbol{m}_j$. So the AVQ convergence theorem implies that class centroids—and, asymptotically, competitive synaptic vectors—minimize the mean-squared error of vector quantization.

Then by (16), the synaptic vectors perform stochastic gradient descent on the mean-squared-error surface in the pattern-plus-error space R^{n+1}. The difference $\boldsymbol{x}(t) - \boldsymbol{m}_j(t)$ behaves as an error vector. The competitive system estimates the unknown centroid $\bar{\boldsymbol{x}}_j$ as $\boldsymbol{x}(t)$ at each time t. Learning is unsupervised but proceeds *as if* it were supervised. Competitive learning reduces to stochastic gradient descent.

VIII. Competitive Covariance Estimation

Centroid $\bar{\boldsymbol{x}}_j$ provides only a first-order estimate of how $p(\boldsymbol{x})$ behaves in region D_j. Local covariances provide a second-order description. We can extend the competitive learning laws to asymptotically estimate the local conditional covariance matrix $\boldsymbol{K}_j$:

$$\boldsymbol{K}_j = E[(\boldsymbol{x} - \bar{\boldsymbol{x}}_j)^T(\boldsymbol{x} - \bar{\boldsymbol{x}}_j)|D_j]. \tag{45}$$

At each iteration we estimate the unknown centroid $\bar{\boldsymbol{x}}_j$ as the current synaptic vector $\boldsymbol{m}_j$. Then $\boldsymbol{K}_j$ equals an *error* conditional covariance matrix. We estimate $\boldsymbol{K}_j$ with the stochastic difference-equation algorithm:

$$\boldsymbol{m}_j(k+1) = \boldsymbol{m}_j(k) + c_k[\boldsymbol{x}_k - \boldsymbol{m}_j(k)] \tag{46}$$

$$\boldsymbol{K}_j(k+1) = \boldsymbol{K}_j(k) + d_k[(\boldsymbol{x}_k - \boldsymbol{m}_j(k))^T \cdot (\boldsymbol{x}_k - \boldsymbol{m}_j(k)) - \boldsymbol{K}_j(k)] \tag{47}$$

for winning synaptic row vector $\boldsymbol{m}_j$.

We can initialize $\boldsymbol{K}_j(0)$ as the null matrix $\boldsymbol{0}$. Initializing $\boldsymbol{K}_j(0)$ as the identity matrix implies that the random vector $\boldsymbol{x}$ has uncorrelated components x_i (and so reduces the adaptive vector quantization process to scalar quantization [23] and might skew the learning procedure for small sets of training samples). We can also scale the difference terms in (46) and (47) to produce supervised-competitive and differential-competitive learning versions. If the ith neuron loses the F_Y metrical competition, then

$$\boldsymbol{m}_i(k+1) = \boldsymbol{m}_i(k) \tag{48}$$

$$\boldsymbol{K}_i(k+1) = \boldsymbol{K}_i(k). \tag{49}$$

The quantity $\{d_k\}$ denotes an appropriately decreasing sequence of learning coefficients in (47).

The computational algorithm (47) corresponds to the stochastic differential equation

$$\dot{\boldsymbol{K}}_j = I_{D_j}(\boldsymbol{x})[(\boldsymbol{x} - \boldsymbol{m}_j)^T(\boldsymbol{x} - \boldsymbol{m}_j) - \boldsymbol{K}_j] + \boldsymbol{N}_j. \tag{50}$$

$\{\boldsymbol{N}_j\}$ denotes an independent Gaussian white noise matrix process. Again we assume $S_j \approx I_{D_j}$ holds. Consider first the noiseless equilibrium condition:

$$\boldsymbol{0} = \dot{\boldsymbol{K}}_j \tag{51}$$

$$= I_{D_j}(\boldsymbol{x})[(\boldsymbol{x} - \bar{\boldsymbol{x}}_j)^T(\boldsymbol{x} - \bar{\boldsymbol{x}}_j) - \boldsymbol{K}_j] + \boldsymbol{N}_j \tag{52}$$

where we have replaced $\boldsymbol{m}_j$ with its equilibrium centroid value. Taking expectations on both sides gives

$$\boldsymbol{0} = \int_{D_j} (\boldsymbol{x} - \bar{\boldsymbol{x}}_j)^T(\boldsymbol{x} - \bar{\boldsymbol{x}}_j) p(\boldsymbol{x})\, d\boldsymbol{x} - \boldsymbol{K}_j \int_{D_j} p(\boldsymbol{x})\, d\boldsymbol{x}. \tag{53}$$

Now solve for the equilibrium covariance value $\boldsymbol{K}_j(\infty)$:

$$\boldsymbol{K}_j = \frac{\int_{D_j} (\boldsymbol{x} - \bar{\boldsymbol{x}}_j)^T(\boldsymbol{x} - \bar{\boldsymbol{x}}_j) p(\boldsymbol{x})\, d\boldsymbol{x}}{\int_{D_j} p(\boldsymbol{x})\, d\boldsymbol{x}} \tag{54}$$

$$= \int_{D_j} (\boldsymbol{x} - E[\boldsymbol{x}|D_j])^T(\boldsymbol{x} - E[\boldsymbol{x}|D_j]) p(\boldsymbol{x}|D_j)\, d\boldsymbol{x} \tag{55}$$

$$= E[(\boldsymbol{x} - E[\boldsymbol{x}|D_j])^T(\boldsymbol{x} - E[\boldsymbol{x}|D_j])|D_j] \tag{56}$$

as desired.

The general stochastic-equilibrium condition takes the form

$$\dot{\boldsymbol{K}}_j = \boldsymbol{N}_j. \tag{57}$$

Then taking expectations of both sides of (57) gives (51). Since the asymptotic $\boldsymbol{K}_j$ wanders in a Brownian motion about the $\boldsymbol{K}_j$ in (45), we cannot directly factor $\boldsymbol{K}_j$ out of the second integral as we did in (53). So again we must take expectations a second time and rearrange. This gives

$$E[\boldsymbol{K}_j(\infty)] = E[(\boldsymbol{x} - \bar{\boldsymbol{x}}_j)^T(\boldsymbol{x} - \bar{\boldsymbol{x}}_j)|D_j]. \quad (58)$$

More generally we can use higher-order correlation tensors, and their Fourier transforms or *polyspectra* [21], to estimate nonlinear non-Gaussian system behavior from noisy sample data. In these cases we add the appropriate third-order, fourth-order, and perhaps higher-order difference equations to the stochastic dynamical system (46) and (47).

REFERENCES

[1] K. L. Chung, *A Course in Probability Theory.* New York: Academic Press, 1974.

[2] D. DeSieno, "Adding a conscience to competitive learning," in *Proc. 2nd IEEE Int. Conf. Neural Networks* (ICNN-88), July 1988, vol. I, pp. 117–124.

[3] W. D. Fisher, "In a pooling problem from the statistical decision viewpoint," *Econometrica*, pp. 567–585, 1953.

[4] W. H. Fleming and R. W. Rishel, *Deterministic and Stochastic Optimal Control.* New York: Springer-Verlag, 1975.

[5] M. A. Gluck, D. B. Parker, and E. Reifsnider, "Some biological implications of a differential-Hebbian learning rule," *Psychobiology*, vol. 16, no. 3, pp. 298–302, 1988.

[6] S. Grossberg, "On learning and energy–entropy dependence in recurrent and nonrecurrent signed networks," *J. Statist. Phys.*, vol. 1, pp. 319–350, 1969.

[7] D. O. Hebb, *The Organization of Behavior.* New York: Wiley, 1949.

[8] R. Hect-Nielsen, "Counterpropagation networks," *Appl. Opt.*, vol. 26, no. 3, pp. 4979–4984, 1 Dec. 1987.

[9] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra.* New York: Academic Press, 1974.

[10] A. H. Klopf, "A drive-reinforcement model of single neuron function: An alternative to the Hebbian neuronal network," in *Proc. Amer. Inst. Phys.: Neural Networks for Computing*, Apr. 1986, pp. 265–270.

[11] A. H. Klopf, "A neuronal model of classical conditioning," *Psychobiology*, vol. 16, no. 2, pp. 85–125, 1988.

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

[13] S. G. Kong and B. Kosko, "Differential competitive learning for centroid estimation and phoneme recognition," *IEEE Trans. Neural Networks*, vol. 2, pp. 118–124, Jan. 1991.

[14] B. Kosko, "Differential Hebbian learning," in *Proc. Amer. Inst. Phys. Neural Networks for Computing*, Apr. 1986, pp. 277–282.

[15] B. Kosko, "Adaptive bidirectional associative memories," *Appl. Opt.*, vol. 26, no. 23, pp. 4947–4960, 1 Dec. 1987.

[16] B. Kosko, "Hidden patterns in combined and adaptive knowledge networks," *Int. J. Approximate Reasoning*, vol. 2, no. 4, pp. 377–393, Oct. 1988.

[17] B. Kosko, "Unsupervised learning in noise," *IEEE Trans. Neural Networks*, vol. 1, pp. 44–57, Mar. 1990.

[18] B. Kosko, *Neural Networks and Fuzzy Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1992.

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

[20] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory*, pp. 7–12, 1960.

[21] C. L. Nikias and M. R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proc. IEEE*, vol. 75, pp. 869–891, July 1987.

[22] T. S. Parker and L. O. Chua, "Chaos: A tutorial for engineers," *Proc. IEEE*, vol. 75, pp. 982–1008, Aug. 1987.

[23] W. K. Pratt, *Digital Image Processing.* New York: Wiley, 1978.

[24] W. Rudin, *Real and Complex Analysis*, 2nd ed. New York: McGraw-Hill, 1974.

[25] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cognitive Sci.*, vol. 9, pp. 75–112, 1985.

[26] A. V. Skorokhod, *Studies in the Theory of Random Processes.* Reading, MA: Addison-Wesley, 1965.

[27] Ya. Z. Tsypkin, *Foundations of the Theory of Learning Systems.* New York: Academic Press, 1973.