Brief Papers

Differential Competitive Learning for Centroid Estimation and Phoneme Recognition

Seong-Gon Kong and Bart Kosko

Abstract—**We compared a differential-competitive-learning (DCL) system with two supervised competitive-learning (SCL) systems for centroid estimation and for phoneme recognition. DCL provides a new form of unsupervised adaptive vector quantization. Standard stochastic competitive-learning systems learn only if neurons win a competition for activation induced by randomly sampled patterns. DCL systems learn only if the competing neurons *change* their competitive signal. Signal-velocity information provides unsupervised *local reinforcement* during learning. The sign of the neuronal signal derivative rewards winners and punishes losers. Standard competitive learning ignores instantaneous *win-rate* information. Synaptic fan-in vectors adaptively quantize the randomly sampled pattern space into nearest-neighbor decision classes. More generally, the synaptic-vector distribution estimates the unknown sampled probability density function $p(x)$. Simulations showed that unsupervised DCL-trained synaptic vectors converged to class centroids at least as fast as, and wandered less about these centroids than, SCL-trained synaptic vectors did. Simulations on a small set of English phonemes favored DCL over SCL for classification accuracy.**

I. Adaptive Vector Quantization for Phoneme Recognition

PHONEME recognition is a simple form of speech recognition. We can recognize a speech sample phoneme by phoneme. The phoneme recognition system learns only a comparatively small set of minimal syllables or phonemes. More advanced systems learn and recognize words, phrases, or sentences. There are orders of magnitude more such speech units than phonemes. Words and phrases can also undergo more complex forms of distortion and time warping.

In principle we can recognize phonemes and speech with *vector quantization* methods. These methods search for a small but representative set of prototypes, which we can then use to match sample patterns with nearest-neighbor techniques.

In neural-network phoneme recognition, a sequence of discrete phonemes from a continuous speech sample produces a series of neuronal responses. Kohonen's [4] supervised neural phoneme-recognition system successfully classifies 21 Finnish phonemes. This stochastic competitive-learning system behaves as an adaptive vector quantization system.

Traditional vector-quantization systems may attempt to minimize a mean-squared-error or entropic performance measure. Formal minimization assumes knowledge of the sampled probability density function $p(\boldsymbol{x})$ and perhaps additional knowledge of how some parameters functionally depend on other parameters. $p(\boldsymbol{x})$ describes the continuous distribution of patterns in R^n. In general, we do not know this probabilistic information. Instead we use learning algorithms to adaptively estimate $p(\boldsymbol{x})$ from sample realizations. This procedure often reduces to *stochastic approximation* [11], [12].

Adaptive-vector-quantization (AVQ) systems adaptively quantize pattern clusters in R^n. Stochastic competitive-learning systems are neural AVQ systems. Neurons compete for the activation induced by randomly sampled patterns. The corresponding synaptic fan-in vectors adaptively quantize the pattern space R^n. The p synaptic vectors $\boldsymbol{m}_j$ define the p columns of the synaptic connection matrix M. M interconnects the n input or linear neurons in the input neuronal field F_X to the p competing nonlinear neurons in the output field F_Y.

In the simplest case the p synaptic vectors estimate centroids or modes of the sampled probability density function $p(\boldsymbol{x})$. The estimates are nonparametric. The user need not know or assume which probability density function $p(\boldsymbol{x})$ generates the training samples, the observed realizations of the underlying stochastic pattern process.

Pattern learning is *supervised* if the system uses pattern-class information. Suppose the k decision classes $\{D_j\}$ partition the pattern space R^n:

$$R^n = \bigcup_{j=1}^{k} D_j \quad \text{and} \quad D_i \cap D_j = \varnothing \quad \text{if } i \neq j. \tag{1}$$

The system knows and uses the class membership of each pattern $\boldsymbol{x}$. The system knows that $\boldsymbol{x} \in D_i$ and that $\boldsymbol{x} \notin D_j$ for all $j \neq i$. Pattern learning is *unsupervised* if the system does not know or use class-membership information. Unsupervised learning algorithms use *unlabeled* pattern samples.

Formally supervised learning depends on class *indicator functions* $\{I_{D_j}\}$:

$$I_{D_j}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in D_j \\ 0 & \text{if } \boldsymbol{x} \notin D_j. \end{cases} \tag{2}$$

I_{D_j} indicates whether pattern $\boldsymbol{x}$ belongs to decision class D_j. Unsupervised learning algorithms blindly cluster samples. They do not depend on class indicator functions. The random indicator functions define the *class probabilities* $P(D_1), \cdots, P(D_k)$, since

$$P(D_j) = \int_{D_j} p(\boldsymbol{x})\, d\boldsymbol{x} \tag{3}$$

$$= \int_{R^n} I_{D_j}(\boldsymbol{x})\, p(\boldsymbol{x})\, d\boldsymbol{x} \tag{4}$$

$$= E[I_{D_j}]. \tag{5}$$

Manuscript received February 22, 1990; revised July 24, 1990. This paper was supported by the Air Force Office of Scientific Research under Grant AFOSR-88-0236 and by a grant from Nippon Telephone and Telegraph.

The authors are with the Department of Electrical Engineering, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-0272.

IEEE Log Number 9038994.

1045-9227/91/0100-0118$01.00 © 1991 IEEE

$E[x]$ denotes the mathematical expectation of scalar random variable x. The partition property and $P(R^n) = 1$ imply $P(D_1) + \cdots + P(D_k) = 1$.

Learning algorithms estimate the unknown probability density function $p(\boldsymbol{x})$. We need not learn if we know $p(\boldsymbol{x})$. Instead we could compute the desired quantities with optimization, numerical-analytical, or calculus-of-variation techniques. For instance, we could directly compute the centroids $\bar{\boldsymbol{x}}_j$ of the pattern classes D_j. The centroids minimize the total mean-squared-error of vector quantization $\mathcal{E}$,

$$\mathcal{E} = \tfrac{1}{2} \sum_{j}^{k} \int_{D_j} \sum_{i}^{n} (x_i - m_{ij})^2 \, p(\boldsymbol{x}) \, d\boldsymbol{x}. \tag{6}$$

For if we set the gradient vector $\nabla_{m_j} \mathcal{E}$ equal to the null vector and solve for the optimal $\hat{\boldsymbol{m}}_j$, we get

$$\mathbf{0} = \nabla_{m_j} \mathcal{E} \tag{7}$$

$$= \int_{D_j} (\boldsymbol{x} - \hat{\boldsymbol{m}}_j) \, p(\boldsymbol{x}) \, d\boldsymbol{x} \tag{8}$$

$$= \int_{D_j} \boldsymbol{x} p(\boldsymbol{x}) \, d\boldsymbol{x} - \hat{\boldsymbol{m}}_j \int_{D_j} p(\boldsymbol{x}) \, d\boldsymbol{x}. \tag{9}$$

Then

$$\hat{\boldsymbol{m}}_j = \frac{\int_{D_j} \boldsymbol{x} p(\boldsymbol{x}) \, d\boldsymbol{x}}{\int_{D_j} p(\boldsymbol{x}) \, d\boldsymbol{x}} \tag{10}$$

$$= \bar{\boldsymbol{x}}_j, \tag{11}$$

as claimed (when positive-definite Hessian conditions hold).

Mean-squared-error optimal learning drives synaptic vectors to the unknown centroids $\bar{\boldsymbol{x}}_j$ of the locally sampled pattern classes. More generally [8], $E[\boldsymbol{m}_j] = \bar{\boldsymbol{x}}_j$ holds asymptotically as the random synaptic vector $\boldsymbol{m}_j$ wanders in a Brownian motion about the centroid $\bar{\boldsymbol{x}}_j$. We observed this Brownian wandering in the simulations discussed below (Fig. 6).

If there are exactly p distinct pattern classes or clusters, the p synaptic row vectors $\boldsymbol{m}_1(t), \cdots, \boldsymbol{m}_p(t)$ should asymptotically approach the centroid of a distinct pattern class. In general we do not know the number k of pattern classes. If there are fewer synaptic vectors than the number k of pattern classes, if $p < k$, the synaptic vectors should approach the centroids of the p most massive, most probable pattern clusters.

If $p > k$, the synaptic vectors should approximate the entire density function $p(\boldsymbol{x})$. More synaptic vectors should arrive at more probable regions. Where patterns $\boldsymbol{x}$ are dense or sparse, synaptic vectors $\boldsymbol{m}_j$ should be dense or sparse. The local count of synaptic vectors then gives an accurate nonparametric estimate of the volume probability $P(V)$ for volume $V \subset R^n$:

$$P(V) = \int_V p(\boldsymbol{x}) \, d\boldsymbol{x} \tag{12}$$

$$\approx \frac{\text{number of } \boldsymbol{m}_j \in V}{p}. \tag{13}$$

In the extreme case that $V = R^n$, this approximation gives $P(V) = p/p = 1$. For small or improbable subsets V, $P(V) = 0/p = 0$.

Differential-competitive-learning (DCL) provides a new [7] unsupervised form of AVQ. DCL modifies stochastic synaptic vectors with a competing neuron's *change* in output signal. The neuronal signal velocity locally reinforces the synaptic vector. The time derivative's sign changes resemble the supervised sign changes in supervised-competitive-learning (SCL) algorithms. SCL systems use more information than DCL systems, since signal velocities do not depend on class-membership information. In particular, the DCL algorithm in (40) below does not use the class membership of the training sample $\boldsymbol{x}$.

Both DCL-trained and SCL-trained synaptic vectors tend to rapidly converge to pattern-class centroids [8]. Our simulated DCL synaptic vectors converged faster to bipolar centroids—points in $\{-1, 1\}^n$—than did SCL synaptic vectors when, as in biological neural networks, a sigmoidal signal function nonlinearly transduced neuronal activations to bounded signals. DCL systems exploit a *win-rate* dependent sequence of learning coefficients. The faster the neuron wins or loses, the more the synaptic vector resembles or disresembles the sampled pattern. SCL systems ignore this instantaneous rate information.

In practice input neurons have linear signal functions: $S_i(x_i) = x_i$. The user presents the random sample $\boldsymbol{x}$ to the system as the *output* of the F_X neurons. In this case, our simulated DCL and SCL synaptic vectors converged equally quickly. But the DCL-trained synaptic vectors wandered less about class centroids than did SCL-trained synaptic vectors.

II. Stochastic Competitive Learning Algorithms

Autoassociative AVQ neural networks are two-layer feedforward networks trained with competitive learning. The input neuronal field F_X receives the sample data and passes it forward through synaptic connection matrix $\boldsymbol{M}$ to the p competing neurons in field F_Y. (Heteroassociative AVQ networks correspond to three-layer feedforward networks.) Synchronous feedforward flow obviates the neural interpretation. AVQ neural systems are simply signal processing algorithms.

The metaphor of competing neurons reduces to nearest-neighbor classification. The system compares the current vector random sample $\boldsymbol{x}(t)$ in Euclidean distance to the p columns of the synaptic connection matrix $\boldsymbol{M}$, to the p synaptic vectors $\boldsymbol{m}_1(t), \cdots, \boldsymbol{m}_p(t)$. If the jth synaptic vector $\boldsymbol{m}_j(t)$ is closest to $\boldsymbol{x}(t)$, then the jth neuron "wins" the competition for activation at time t.

Many within-field feedback dynamical systems approximate this nearest-neighbor, winner-take-all behavior. Mathematically, the jth competing neuron should behave as a class indicator function: $S_j = I_{D_j}$. More generally the jth F_Y neuron only estimates I_{D_j}. Then misclassification can still occur: $S_j(\boldsymbol{x}\boldsymbol{m}_j^T + f_j) = 1$ but $I_{D_j}(\boldsymbol{x}) = 0$ for row vectors $\boldsymbol{x}$ and $\boldsymbol{m}_j$, where f_j denotes the inhibitive within-field feedback the jth neuron receives.

We modify the nearest or "winning" synaptic vector $\boldsymbol{m}_j$ with a simple difference learning law. We add some scaled form of $\boldsymbol{x}(t) - \boldsymbol{m}_j(t)$ to $\boldsymbol{m}_j(t)$ to form $\boldsymbol{m}_j(t+1)$. We can also update near-neighbors of the winning neuron. In practice, and in the simulations below, we modify only one synaptic vector at a time. We do not modify "losers": $\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t)$.

The stochastic *unsupervised-competitive-learning* (UCL) algorithm represents the simplest competitive-learning algorithm. Pattern recognition theorists first studied the UCL algorithm but called it *adaptive K-means clustering* [10]. Kohonen extended the UCL algorithm to two supervised versions, SCL1 [3], [4] and SCL2 [5]. The supervisor must know the class membership of each sample pattern $\boldsymbol{x}$. The SCL1 and SCL2 algorithms linearly "reward" correct classifications as in the UCL algorithm. They "punish" incorrect classifications with a sign change. We obtain all three algorithms from the following three-step algo-

rithm if we replace the third step with the appropriate stochastic difference equation.

Competitive AVQ Algorithms

1) Initialize synaptic vectors: $\boldsymbol{m}_i(0) = \boldsymbol{x}(i)$, $i = 1, \cdots, p$. Sample-dependent initialization avoids many pathologies that can distort nearest-neighbor learning.

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

$$\left\| \boldsymbol{m}_j(t) - \boldsymbol{x}(t) \right\| = \min_i \left\| \boldsymbol{m}_i(t) - \boldsymbol{x}(t) \right\|, \tag{14}$$

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

3) Update the winning synaptic vector $\boldsymbol{m}_j(t)$ with the UCL, SCL1, or SCL2 learning algorithm.

A. Unsupervised Competitive Learning (UCL)

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + c_t[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)], \tag{15}$$

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) \quad \text{if } i \neq j, \tag{16}$$

where $\{c_t\}$ denotes a slowly decreasing sequence of learning coefficients. In our simulations, $c_t = 0.1(1 - (t/2000))$ for 2000 training samples. The UCL algorithm (15) restates the classical adaptive K-means clustering algorithm.

Stochastic approximation [11] requires a decreasing gain sequence $\{c_t\}$ to suppress random disturbances and to guarantee convergence to a local minima of mean-squared performance measures. The learning coefficients should decrease slowly,

$$\sum_{t=1}^{\infty} c_t = \infty, \tag{17}$$

but not too slowly,

$$\sum_{t=1}^{\infty} c_t^2 < \infty. \tag{18}$$

Harmonic-series coefficients, $c_t = 1/t$, satisfy these constraints. For fast *robust* [2] stochastic approximation, only the harmonic-series coefficients satisfy these constraints.

B. Supervised Competitive Learning 1 (SCL1)

$$\boldsymbol{m}_j(t+1) = \begin{cases} \boldsymbol{m}_j(t) + c_t[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)] \\ \qquad \text{if } \boldsymbol{x}(t) \in D_j \\ \boldsymbol{m}_j(t) - c_t[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)] \\ \qquad \text{if } \boldsymbol{x}(t) \notin D_j, \end{cases} \tag{19}$$

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) \quad \text{if } i \neq j. \tag{20}$$

SCL1 supervises or reinforces synaptic modification. $\boldsymbol{m}_j$ learns positively if the system correctly classifies the random sample $\boldsymbol{x}$. $\boldsymbol{m}_j$ learns negatively, or forgets selectively, if the system misclassifies the random sample. Then $\boldsymbol{m}_j$ tends to move out of regions of misclassification in R^n. Tsypkin [12] first derived the SCL1 algorithm as a special case of his adaptive Bayes classifier.

We can rewrite the SCL1 update equation (19) as

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + c_t r_j(\boldsymbol{x}(t))\left[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)\right] \tag{21}$$

if we define the supervised *reinforcement function* r_j as

$$r_j = I_{D_j} - \sum_{i \neq j} I_{D_i}. \tag{22}$$

r_j depends explicitly on class indicator functions. r_j rewards correct pattern classifications with $+1$ and punishes misclassifications with -1. We implicitly assume the jth neuron accurately estimates the jth indicator function: $S_j(\boldsymbol{x}\boldsymbol{m}_j^T + f_j) \approx I_{D_j}(\boldsymbol{x})$.

The SCL2 algorithm modifies slightly the SCL1 algorithm. The SCL2 algorithm better estimates the optimal Bayes decision-theoretic boundary in some cases. The Bayes decision boundary minimizes the misclassification error. It represents the crossing point of the unknown conditional densities $p(\boldsymbol{x} \mid D_i)$ and $p(\boldsymbol{x} \mid D_j)$.

The nearest-neighbor decision boundary corresponds to the hyperplane that bisects the line that connects the two class centroids. If the pattern distribution is asymmetric—if, for instance, local density functions with different variances generate different decision classes—then the SCL1 decision boundary may not resemble the Bayes decision boundary. Nearest-neighbor classification tends to perform better in the equal variance case than in the unequal variance case.

C. Supervised Competitive Learning 2 (SCL2)

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) - c_t[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)], \tag{23}$$

$$\boldsymbol{m}_l(t+1) = \boldsymbol{m}_l(t) + c_t[\boldsymbol{x}(t) - \boldsymbol{m}_l(t)], \tag{24}$$

if $\boldsymbol{x} \in D_l$ instead of $\boldsymbol{x} \in D_j$, and if $\boldsymbol{m}_j(t)$ is the nearest synaptic vector and $\boldsymbol{m}_l(t)$ is the next-to-nearest synaptic vector:

$$\left\| \boldsymbol{m}_j(t) - \boldsymbol{x}(t) \right\| < \left\| \boldsymbol{m}_l(t) - \boldsymbol{x}(t) \right\| = \min_{i \neq j} \left\| \boldsymbol{m}_i(t) - \boldsymbol{x}(t) \right\|, \tag{25}$$

and if $\boldsymbol{x}(t)$ falls in a class-dependent "window." In all other cases,

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t). \tag{26}$$

The window defines a hyperrectangle in R^n centered at the midpoint of the hyperline that connects the centroids of D_j and D_l. If $\boldsymbol{x}(t)$ does not fall in the hyperwindow, we modify no synaptic vector. We defined the R^n window between D_j and D_l as the n-dimensional hyperrectangle $[\overline{m}_1 - d, \overline{m}_1 + d] \times \cdots \times [\overline{m}_n - d, \overline{m}_n + d]$, where $\overline{m}_{jl}$ denotes the midpoint $\overline{m}_{jl} = (\overline{m}_1, \cdots, \overline{m}_n) = (\boldsymbol{m}_j + \boldsymbol{m}_l)/2$, and d denotes the window half-width. We put $d = 2.5$.

III. Differential Competitive Learning

The *differential competitive learning* (DCL) law [7] combines competitive and differential Hebbian learning:

$$\dot{m}_{ij} = \dot{S}_j(y_j)\left[S_i(x_i) - m_{ij}\right], \tag{27}$$

or in vector notation

$$\dot{\boldsymbol{m}}_j = \dot{S}_j(y_j)\left[\boldsymbol{S}(\boldsymbol{x}) - \boldsymbol{m}_j\right], \tag{28}$$

where $\boldsymbol{S}(\boldsymbol{x}) = (S_1(x_1), \cdots, S_n(x_n))$ and $\boldsymbol{m}_j = (m_{1j}, \cdots, m_{nj})$. m_{ij} denotes the synaptic weight between the ith neuron in input neuronal field F_X and the jth neuron in competitive field F_Y. Nonnegative *signal functions* S_i and S_j transduce the real-valued *activations* x_i and y_j into the bounded monotone nondecreasing *signals* $S_i(x_i)$ and $S_j(y_j)$. $\dot{m}_{ij}$ and $\dot{S}_j(y_j)$ denote the

time derivatives of m_{ij} and $S_j(y_j)$, synaptic and signal velocities.

The stochastic calculus version of the DCL law relates random processes:

$$dm_{ij} = dS_j[S_i - m_{ij}] + dB_{ij}. \tag{29}$$

B_{ij} denotes a Brownian-motion diffusion process centered at the origin. We can rewrite (29) in "noise" notation as

$$\dot{m}_{ij} = \dot{S}_j[S_i - m_{ij}] + n_{ij}. \tag{30}$$

The "noise" process n_{ij} has zero mean, $E[n_{ij}] = 0$, and has finite variance, $V[n_{ij}] = \sigma_{ij}^2 < \infty$. The random-sampling AVQ framework implicitly assumes that all competitive learning laws are stochastic differential or difference equations. Such stochastic synaptic vectors $\boldsymbol{m}_j$ tend to converge to pattern-class centroids, and converge exponentially quickly [8].

$S_j(y_j)$ measures the competitive status of the jth competing neuron in F_Y. Usually, S_j approximates a binary threshold function. S_j may equal a steep binary logistic sigmoid,

$$S_j(y_j) = \frac{1}{1 + e^{-cy_j}}, \tag{31}$$

for some constant $c > 0$. The jth neuron wins the laterally inhibitive competition if $S_j = 1$, loses if $S_j = 0$.

In (27) $\boldsymbol{m}_j$ learns only if $S_j(y_j)$ changes. This contrasts with the classical competitive learning law

$$\dot{m}_{ij} = S_j(y_j)[S_i(x_i) - m_{ij}], \tag{32}$$

which modulates the difference $\boldsymbol{S}(\boldsymbol{x}) - \boldsymbol{m}_j$ with the win–loss signal S_j, not its velocity $\dot{S}_j$. In (32) $\boldsymbol{m}_j$ learns only if the competitive signal S_j exceeds zero—only if the jth neuron "wins" the activation competition.

Real neurons transmit and receive pulse trains. Pulse-coded signal functions S_j reveal the connection between competitive and differential competitive learning. A pulse-coded signal function uses an exponentially fading window [1] of sampled binary pulses:

$$S_j(t) = \int_{-\infty}^{t} y_j(s)\, e^{s-t}\, ds, \tag{33}$$

where $y_j(t) = 1$ if a pulse occurs at t, and $y_j(t) = 0$ if no pulse occurs at t. Then [9]

$$\dot{S}_j(t) = y_j(t) - S_j(t). \tag{34}$$

So the DCL law (27) reduces to

$$\dot{m}_{ij} = y_j[S_i - m_{ij}] - S_j[S_i - m_{ij}]. \tag{35}$$

When the second term in (35) is sufficiently small, DCL reduces to competitive learning. This occurs when a losing neuron suddenly wins, for then $y_j = 1$ and $S_j \approx 0$. In the stochastic case, the *random* pulse function y_j represents an arbitrary random point process, and converts (35) to a doubly stochastic model.

Similarly, the classical differential Hebbian law [6]

$$\dot{m}_{ij} = -m_{ij} + \dot{S}_i\dot{S}_j \tag{36}$$

reduces to signal Hebbian learning on average (in the absence of pulses):

$$\dot{m}_{ij} = -m_{ij} + \dot{S}_i\dot{S}_j \tag{37}$$

$$= -m_{ij} + S_iS_j + [x_iy_j - x_iS_j - y_jS_i] \tag{38}$$

$$\approx -m_{ij} + S_iS_j, \tag{39}$$

on average. The approximation holds exactly if and only if no x_i or y_j pulses are present, a frequent event. Differential-Hebbian-learning synapses "fill in" with Hebbian learning when pulses are absent.

For discrete implementation, we use the DCL algorithm as a stochastic difference equation.

A. Differential Competitive Learning (DCL)

1) Initialize: $\boldsymbol{m}_i(0) = \boldsymbol{x}(i)$.

2) Find winning $\boldsymbol{m}_j(t)$: $\|\boldsymbol{m}_j(t) - \boldsymbol{x}(t)\| = \min_i \|\boldsymbol{m}_i(t) - \boldsymbol{x}(t)\|$.

3) Update winning $\boldsymbol{m}_j(t)$:

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + c_t\Delta S_j(y_j(t))\left[S(\boldsymbol{x}(t)) - \boldsymbol{m}_j(t)\right] \quad \text{if the } j\text{th neuron wins,} \tag{40}$$

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) \quad \text{if the } i\text{th neuron loses.} \tag{41}$$

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

$$\Delta S_j(y_j(t)) = \operatorname{sgn}\left[S_j(y_j(t+1)) - S_j(y_j(t))\right]. \tag{42}$$

We define the *signum operator* sgn (x) as

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0. \end{cases} \tag{43}$$

We update the F_Y neuronal activations y_j with the additive model

$$y_j(t+1) = y_j(t) + \sum_i^n S_i(x_i(t))\, m_{ij}(t) + \sum_k^p S_k(y_k(t))\, w_{kj}. \tag{44}$$

In our simulations, the first sum in (44) reduced to

$$\sum_i^n x_i(t)\, m_{ij}(t) \tag{45}$$

when we did not transform the input patterns $\boldsymbol{x}$ with a nonlinear signal function S_i. Input or F_X neurons in feedforward networks usually behave linearly: $S_i(x_i) = x_i$.

For linear inputs, we computed the second sum in (44) for linear signal functions S_k. Since we allowed only one winner per iteration, this sum reduced to a single term $y_k w_{kj}$, where k denotes the winning neuron.

The $p \times p$ matrix W defined the F_Y within-field synaptic connection strengths:

$$W = \begin{bmatrix} +2 & -1 & -1 & \cdots & -1 \\ -1 & +2 & -1 & \cdots & -1 \\ & & \vdots & & \\ -1 & -1 & -1 & \cdots & +2 \end{bmatrix}. \tag{46}$$

Diagonal elements w_{ii} equaled 2, off-diagonal elements equaled -1. Fig. 1 shows the connection topology of the laterally inhibitive DCL network.

Each neuron in F_Y codes for a specific pattern class. By (44) and the "cosine law,"

$$S(\boldsymbol{x}) \cdot \boldsymbol{m}_j = \|S(\boldsymbol{x})\|\, \|\boldsymbol{m}_j\| \cos(S(\boldsymbol{x}), \boldsymbol{m}_j) \tag{47}$$

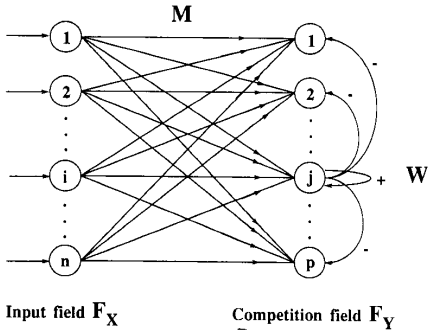

Fig. 1. Topology of the laterally inhibitive DCL network.

positive learning ($\dot{m}_{ij} > 0$) tends to occur when the system classifies $\boldsymbol{x}$ to the nearest pattern class D_j.

If we represent the F_X signal function S_i with the bipolar logistic function,

$$S_i(x_i) = \frac{2}{1 + e^{-cx_i}} - 1, \tag{48}$$

$c > 0$, then the DCL algorithm (40) abstracts the corresponding bipolar pattern from the real-valued input. The unsupervised sign change ΔS_j in the DCL law (40) resembles the reward–punish sign change in the SCL1 and SCL2 algorithms. This suggests that we can meaningfully compare the algorithms' performance on the same training and test data.

If we choose $S_i(x_i)$ as a linear function of the input, if $S_i(x_i) = x_i$, then the discrete version of DCL resembles the UCL, SCL1, and SCL2 algorithms. We used both linear and nonlinear formulations to compare DCL with SCL1 and SCL2. The supervised SCL1 and SCL2 algorithms always outperformed the UCL algorithm. So we limited our DCL comparisons to SCL1 and SCL2 systems.

For most simulations we used linearly transformed data, $S_i(x_i) = x_i$. In these cases we approximated the signal difference ΔS_j as the activation difference Δy_j:

$$\Delta S_j(y_j(t)) \approx \Delta y_j(t) \tag{49}$$

$$= \operatorname{sgn}[y_j(t+1) - y_j(t)]. \tag{50}$$

This approximation holds exactly over the linear part of a signal function's range. For then $S_j' = dS_j/dy_j = c$ for some constant $c > 0$. Then

$$\dot{S}_j = S_j' \dot{y}_j \tag{51}$$

$$= c\dot{y}_j. \tag{52}$$

The constant c does not affect the signum operator used in Δy_j.

Linear data often produce large activation sums $\Sigma_i^n x_i m_{ij}$ that saturate nonlinear signals S_j to extreme values. Then the signal difference ΔS_j equals zero and may not discriminate changes in the competitive status. The activation difference Δy_j remains sensitive to these changes.

IV. Comparison of Competitive and Differential Competitive Learning for Centroid Estimation

We compared the DCL algorithm with the SCL1 and SCL2 algorithms for estimating centroids. All algorithms adaptively moved the synaptic vectors $\boldsymbol{m}_j$ to pattern-class centroids. They

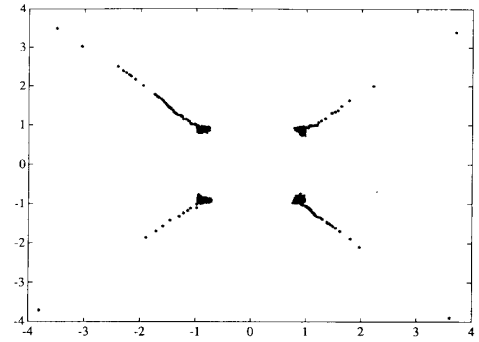

Fig. 2. Convergence of DCL-trained synaptic vectors to bipolar centroids of four Gaussian clusters. Bipolar logistic signal functions $S_i(x_i)$ nonlinearly transduce the real-valued input vector $\boldsymbol{x}$ into a bipolar vector in $\{-1, 1\}^n$.

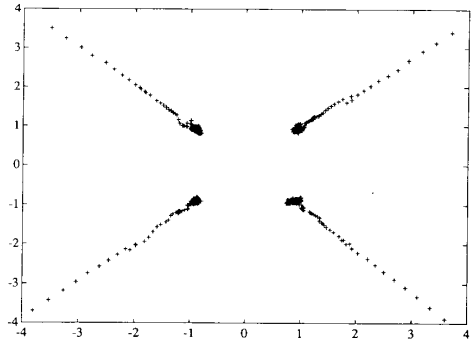

Fig. 3. Centroid convergence of SCL1 synaptic vectors trained with the same patterns as in Fig. 2. Bipolar logistic signal functions nonlinearly transduce real input patterns to bipolar patterns.

differed in how quickly the trained synaptic vectors reached the centroids and how much the synaptic vectors wandered about the centroids. The DCL algorithm moved the synaptic vectors to centroids at least as fast as did the SCL1 and SCL2 algorithms. Once the synaptic vectors reached the pattern-class centroids, the DCL-trained synaptic vectors wandered less about the centroids than the SCL-trained synaptic vectors wandered.

The DCL algorithm converged to centroids faster than the SCL1 and SCL2 algorithms converged. Convergence rates were the same for linear signal functions $S_i(x_i) = x_i$. The pattern space consisted of 2000 two-dimensional Gaussian-distributed pattern vectors with variance 121 and with centroids or modes at (20, 20), (20, −20), (−20, 20), and (−20, −20). Fig. 2 shows centroid convergence of DCL synaptic vectors with inputs transformed with bipolar signal functions. Fig. 3 shows the slower convergence of the SCL1 algorithm with the same transformed Gaussian Data. "*" denotes DCL synaptic vectors. "+" denotes SCL1 synaptic vectors. Figs. 4 and 5 show centroid convergence for the same Gaussian data when the systems used linear signal functions.

DCL-trained synaptic vectors wandered with less mean-squared error about centroids than did SCL-trained synaptic vectors. Fig. 6 shows mean-squared wandering about the Gaussian pattern-class centroid (−20, 20). Fig. 6 represents several such experiments with different Gaussian and non-

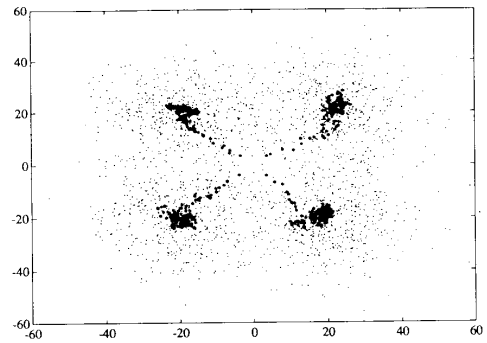

Fig. 4. Convergence of DCL-trained synaptic vectors to Gaussian pattern-class centroids. Same pattern distribution as in Figs. 2 and 3. Input data not transformed: $S_i(x_i) = x_i$.

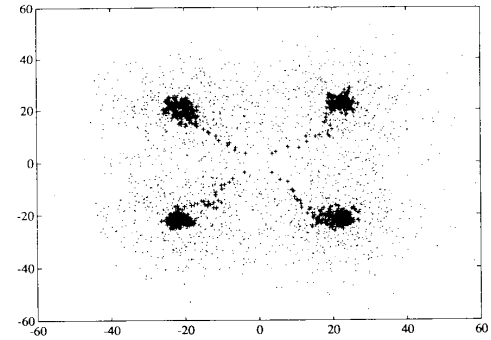

Fig. 5. Convergence of SCL1-trained synaptic vectors to Gaussian pattern-class centroids for the same pattern distribution as in Fig. 4.

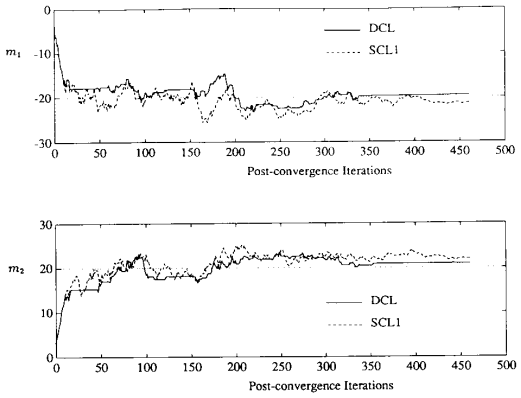

Fig. 6. Trajectories of the synaptic vectors after reaching the Gaussian pattern-class centroid at $(-20, 20)$. Solid lines represent DCL-trained synaptic vectors. Dashed lines represent SCL1-trained synaptic vectors. The two graphs plot separately the m_1 and m_2 components of the synaptic vector $\boldsymbol{m} = (m_1, m_2)$.

Gaussian pattern distributions. Solid lines denote the convergence of the DCL synaptic vector. Dashed lines denote convergence of the SCL1 synaptic vector. We calculated the mean-squared error (MSE) of centroid wandering for the class centered at $(-20, 20)$ after 200 iterations. Other centroids produced comparable MSE of centroid wandering. In the first case, we used 540 Gaussian samples with variance 25. Then, for the DCL algorithm, the MSE of centroid wandering equaled 0.48. For the SCL1 algorithm, it equaled 1.48. In the second case, we used 554 Gaussian samples with variance 121. Then, for the DCL algorithm, the MSE of centroid wandering equaled 4. For the SCL1 algorithm, it equaled 7.11.

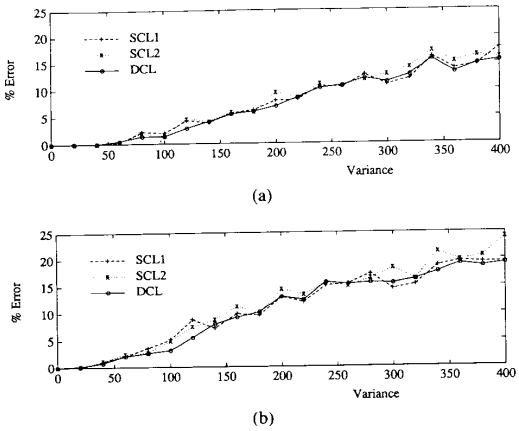

Fig. 7. AVQ misclassification rates for two Gaussian clusters: (a) with equal variance centered about the centroids $(-20, 0)$ and $(20, 0)$ and (b) with unequal variance. In (b), the pattern class centered about $(20, 0)$ has twice the variance of the pattern class centered about $(-20, 0)$.

Next we compared the DCL system to the SCL1 and SCL2 systems for pattern classification accuracy. We trained each AVQ system with 500 Gaussian-distributed samples for each pattern class, and for each variance level centered about the same centroids $(-20, 0)$ and $(20, 0)$. We set variance levels at 20 units. For each variance level, we tested each AVQ system with 1000 new Gaussian-distributed samples for each pattern class. Fig. 7(a) shows the misclassification rates of the DCL, SCL1, and SCL2 systems for two representative Gaussian classes with equal variances. Fig. 7(b) shows misclassification performance for each AVQ system when we repeated the simulation in Fig. 7(a) for unequal variances. The pattern class with centroid $(20, 0)$ had twice the variance of the pattern class with centroid $(-20, 0)$. The three clustering algorithms behaved similarly for increasing variance values.

V. Phoneme Recognition Simulations

We obtained speech training samples from samples of continuous male speech with different English pronunciations. We used a time-dependent Fourier spectrum to extract features from the speech waveforms. An anti-alias low-pass filter prefiltered the speech signals. We then digitized the signals to 8 bits with a 10 kHz sampling frequency. A Hamming window divided the digitized speech signal into 256 sample segments. The fast Fourier transform algorithm gave 256 complex Fourier coefficients for each of the 256 windowed sample segments. We divided the 200 Hz–5 kHz frequency range into 16 regions. We divided the 200 Hz–3 kHz frequency range into 12 equal regions and the 3–5 kHz frequency range into four equal regions. Six Fourier coefficients represented each region in the 200 Hz–3 kHz range. 13 Fourier coefficients represented each region in the 3–5 kHz range. We calculated average power spectra over each region

TABLE I
PERCENTAGE MISCLASSIFICATION RATES OF THE DCL, SCL1, AND SCL2 SYSTEMS FOR THE NINE ENGLISH PHONEMES /a, e, i, o, u, f, s, n, t/

	DCL	SCL1	SCL2
/a/	0	0	0
/e/	3	9	4
/i/	0	0	4
/o/	5	3	16
/u/	0	1	2
/f/	28	43	53
/s/	1	2	6
/n/	3	4	7
/t/	52	48	26

to form a 16-dimensional pattern vector. We produced 16-dimensional phoneme pattern vectors by repeatedly sliding the Hamming window by 100 samples.

The sample space consisted of real and artificial phonemes. The artificial phonemes were Gaussian random vectors with variation 9 centered at the real phoneme vectors. We generated these noisy phoneme samples to provide the AVQ systems with a statistically representative set of training samples.

The simulation compared the DCL, SCL1, and SCL2 learning systems for classification of nine representative English phonemes: five vowels /*a*, *e*, *i*, *o*, *u*/, two fricatives /*f*, *s*/, one nasal /*n*/, and one plosive sound /*t*/. Table I lists the misclassification rates. The AVQ systems tended to more accurately classify vowel and nasal sounds than they classified fricative and plosive sounds.

We trained each competitive AVQ system with 1000 Gaussian-distributed random phoneme vectors clustered into nine pattern classes. Each pattern class was centered about the original spoken phoneme and radially distributed with variance $\sigma^2 = 9$. We randomly selected training data according to a uniform probability distribution to simulate nine equiprobable pattern classes. We tested each AVQ system with 100 new Gaussian-distributed phoneme samples for each phoneme type. Except for the two phonemes /*o*/ and /*t*/, the DCL system misclassified no more frequently than the SCL systems misclassified.

VI. CONCLUSIONS

The DCL system performed well in centroid estimation and phoneme recognition. DCL synaptic vectors converged faster to centroids than did SCL1 synaptic vectors when logistic bipolar signal functions transformed the input sample. DCL synaptic vectors wandered less about pattern-class centroids than SCL synaptic vectors wandered.

Our phoneme-recognition simulations were preliminary, but agreed with our centroid-estimation simulations. The phoneme-recognition simulations suggest that unsupervised DCL systems will perform as well as supervised SCL1 and SCL2 systems in many pattern environments, even though DCL systems use less pattern-class information.

In general we do not know in advance whether $\boldsymbol{x} \in D_i$ for every training sample $\boldsymbol{x}$, and for every pattern class D_i, for an arbitrary classification, filtering, or estimation problem. We may not even know approximately the number or characteristics of the underlying decision classes. We can still apply DCL techniques in these cases and expect SCL-level performance. But we may never know how SCL systems would perform on the same data.

REFERENCES

[1] M. Gluck, D. Parker, and E. Reifsnider, "Some biological implications of a differential-Hebbian learning rule," *Psychobiol.*, vol. 16, no. 3, pp. 298–302, 1988.
[2] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.
[3] T. Kohonen, *Self-Organization and Associative Memory, 2nd edition*. New York: Springer-Verlag, 1988.
[4] ——, "The neural phonetic typewriter," *IEEE Comput. Mag.*, pp. 11–22, Mar. 1988.
[5] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with neural networks: Benchmarking studies," in *Proc. Int. Conf. Neural Networks (ICNN-88)*, 1988, vol. I, pp. 61–68.
[6] B. Kosko, "Differential Hebbian learning," in *Proc. Amer. Inst. Phys. Conf.: Neural Networks for Comput.*, Apr. 1986, pp. 277–282.
[7] ——, "Unsupervised learning in noise," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 44–57, Mar. 1990.
[8] ——, "Stochastic competitive learning," in *Proc. Summer 1990 Int. Joint Conf. Neural Networks (IJCNN-90)*, June 1990, vol. II, pp. 215–226.
[9] ——, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1990.
[10] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Prob.*, 1967, pp. 281–297.
[11] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, pp. 400–407, 1951.
[12] Y. Z. Tsypkin, *Foundations of the Theory of Learning Systems*. New York: Academic, 1973.