Communicated by Demetri Psaltis

# Fast Learning in Networks of Locally-Tuned Processing Units

John Moody
Christian J. Darken
*Yale Computer Science, P.O. Box 2158, New Haven, CT 06520, USA*

We propose a network architecture which uses a single internal layer of locally-tuned processing units to learn both classification tasks and real-valued function approximations (Moody and Darken 1988). We consider training such networks in a completely supervised manner, but abandon this approach in favor of a more computationally efficient hybrid learning method which combines self-organized and supervised learning. Our networks learn faster than backpropagation for two reasons: the local representations ensure that only a few units respond to any given input, thus reducing computational overhead, and the hybrid learning rules are linear rather than nonlinear, thus leading to faster convergence. Unlike many existing methods for data analysis, our network architecture and learning rules are truly adaptive and are thus appropriate for real-time use.

Neurons with response characteristics which are "locally-tuned" or "selective" for some range of the input variables are found in many parts of nervous systems. For example, the cochlear stereocilia cells have locally-tuned response to frequency, while cells in somatosensory cortex respond selectively to stimulation from localized regions of the body surface. The orientation-selective cells in visual cortex respond selectively to stimulation which is both local in retinal position and local in angle of object orientation, while cells in nucleus laminaris of barn owl are tuned to specific interaural time delays. Populations of locally-tuned cells are typically arranged in cortical maps in which the values of the variables to which the cells respond vary with position in the map.

The locally-tuned response of the stereocilia cells is a consequence of their biophysical properties. In the three other examples, however, the locality of response is a consequence of the network architecture of each of the various systems. In these cases, the response is local not in specific pre-synaptic activities, but rather in variables which have meaning only at a systems level. This locality is a computational property of the *system* and should *not* be confused with the biophysical response properties of *cells*, which are usually treated in the abstract as a thresholding of a weighted sum of inputs.

In this letter, we present a simple, idealized network model based upon abstract processing units with locally-tuned response functions. We consider training such a network using purely supervised learning, but conclude that this offers no significant advantages over backpropagation. We then show that such a network can be efficiently trained to perform computationally interesting tasks via a combination of linear supervised and linear self-organizing techniques. The combination of locality of representation and linearity of learning offers tremendous speed advantages relative to backpropagation.

As will become apparent, our model is best suited for learning to approximate continuous or piecewise continuous real-valued mappings $f : \mathcal{R}^n \mapsto \mathcal{R}^m$ where $n$ is sufficiently small. This class of functions includes classification problems as a special case. The model is not well suited to learning logical mappings such as parity, because such mappings are not usually piecewise continuous.

It should be noted at the outset that local methods for density estimation (for example, Parzen Windows), classification, interpolation, and approximation have been widely used for many years and are known to have attractive computational properties including in some cases straightforward parallelizability and rapid speed of solution. However, most of the local methods which have been studied are intrinsically "non-neural," meaning that they can not be easily implemented in an adaptive network of fixed architecture and size.

A network of $M$ locally-tuned units (Moody and Darken 1988) (see figure 1a) has an overall response function:

$$\boxed{f(\vec{x}) = \sum_{\alpha=1}^{M} A^\alpha R^\alpha(\vec{x})} \tag{1.1}$$

$$\boxed{R^\alpha(\vec{x}) \equiv R(\|\vec{x} - \vec{x}^\alpha\|/\sigma^\alpha).} \tag{1.2}$$

Here, $\vec{x}$ is a real-valued vector in the input space, $R^\alpha$ is the response function of the $\alpha$-th locally-tuned unit, $R$ is a radially-symmetric function with a single maximum at the origin and which drops off rapidly to zero at large radii, $\vec{x}^\alpha$ and $\sigma^\alpha$ are the center and width in the input space of the $\alpha$-th unit, and $A^\alpha$ is the weight or amplitude associated with each unit. For this work, we have chosen gaussian response functions with unit normalization:

$$R^\alpha(\vec{x}) = \exp\left[-\frac{\|\vec{x} - \vec{x}^\alpha\|^2}{(\sigma^\alpha)^2}\right]. \tag{1.3}$$

One can think of the functional representation in equation (1.1) as a decomposition into basis functions which are not orthonormal, not uniformly distributed over the input space, and do not have uniform width

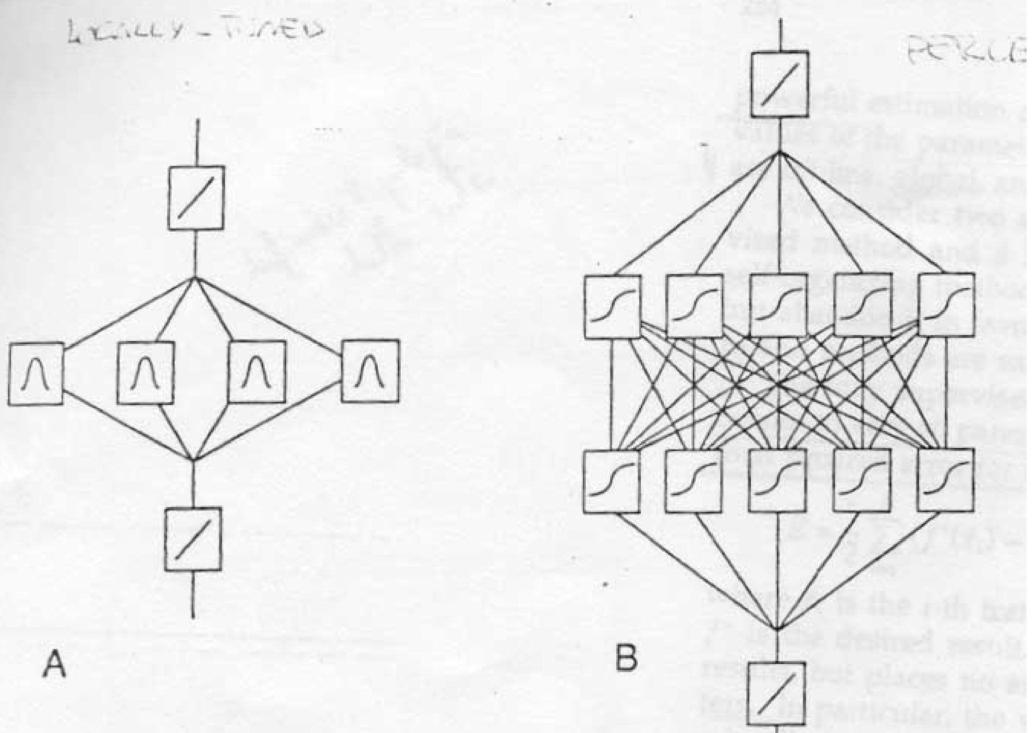

Figure 1: A network of locally-tuned processing units (a) and a three layer perceptron (b). Unit response functions are depicted graphically. Both networks have linear inputs and outputs. The network of locally-tuned units has a single internal layer, while the perceptron has two internal layers of sigmoidal units.

With the addition of lateral connections between the processing units, the network can dynamically produce the normalized response function (Moody 1989b):

$$f(\vec{x}) = \frac{\sum_\alpha A^\alpha R^\alpha(\vec{x})}{\sum_\alpha R^\alpha(\vec{x})}. \tag{1.4}$$

This is essentially a weighted average or interpolation between weights or learned function values $A^\alpha$ of nearby processing units.

The locality of the unit response functions is important for attaining fast simulation speeds on serial machines. For any given input, only the small fraction of processing units with centers very close (in the input space) to the input will respond with activations which differ significantly from zero. Thus, only those with centers close enough to the input need to be evaluated and trained. We efficiently identify these units by partitioning the input space with an adaptive grid (Omohundro 1987) and doing a short search.

The functional forms given by equations (1.1) to (1.4) are special cases of the kernel representations developed by statisticians. A number of powerful estimation and regression techniques exist for finding the best values of the parameters $\{\vec{x}^\alpha, \sigma^\alpha, A^\alpha\}$. However, many of these methods are off-line, global, and are not easily adapted to real-time use.

We consider two styles of learning in our networks, a fully supervised method and a hybrid method which combines supervised with self-organizing methods. We consider the fully supervised method first, but abandon it in favor of the hybrid method. Both the supervised and hybrid methods are easily implemented adaptively.

The fully supervised method uses an error measure $E$ defined at the output to vary *all* parameters $\{\vec{x}^\alpha, \sigma^\alpha, A^\alpha\}$ in a network of form (1.1). The total squared error for an entire training set is:

$$E = \frac{1}{2}\sum_{i=1}^{N} (f^*(\vec{x}_i) - f(\vec{x}_i))^2 , \tag{1.5}$$

where $\vec{x}_i$ is the $i$-th training pattern, $f$ is the total network output, and $f^*$ is the desired result. The supervised method yields high precision results, but places no architectural restrictions on the network parameters. In particular, the widths $\sigma^\alpha$ are not restricted to remain small, so a locally-tuned representation is not guaranteed. Thus, the supervised learning method does not obtain the computational advantages of locality. Furthermore, the supervised method casts learning as a non-linear optimization problem; this results in both slow convergence and unpredictable solutions (locally-tuned units are sometimes "squeezed out" of the region of the input space which contains data).

It is interesting, however, to compare the learning performance of a supervised network of gaussian units with a backpropagation network of comparable size. Following Lapedes and Farber (Lapedes and Farber 1987), we consider a simple quadratic approximation problem, predicting the evolution of the logistic map: $x[t+1] = 4x[t](1 - x[t])$. The networks of type (1.1) have 4, 5, or 6 internal gaussian units. The backpropagation network has a single internal layer with 5 sigmoidal units, a linear output unit, and an additional linear connection between the input and the output. All networks were trained using the conjugate gradient optimization procedure for 200 iterations (line minimizations) on a data set of 1000 points and were tested on a data set of 500 points. The number of modifiable network parameters, training set error, test set error, and total computation time in Sun 3/50 CPU seconds is shown in the following table:

| Network | #Par | Time | Train Error | Test Error |
|---|---|---|---|---|
| 5 Sigmoidal Units w/ Linear Term | 17 | 3802 sec | 0.58% | 0.59% |
| 4 Gaussian Units | 12 | 3137 sec | 0.62% | 0.64% |
| 5 Gaussian Units | 15 | 4278 sec | 0.38% | 0.41% |
| 6 Gaussian Units | 18 | 4945 sec | 0.26% | 0.27% |

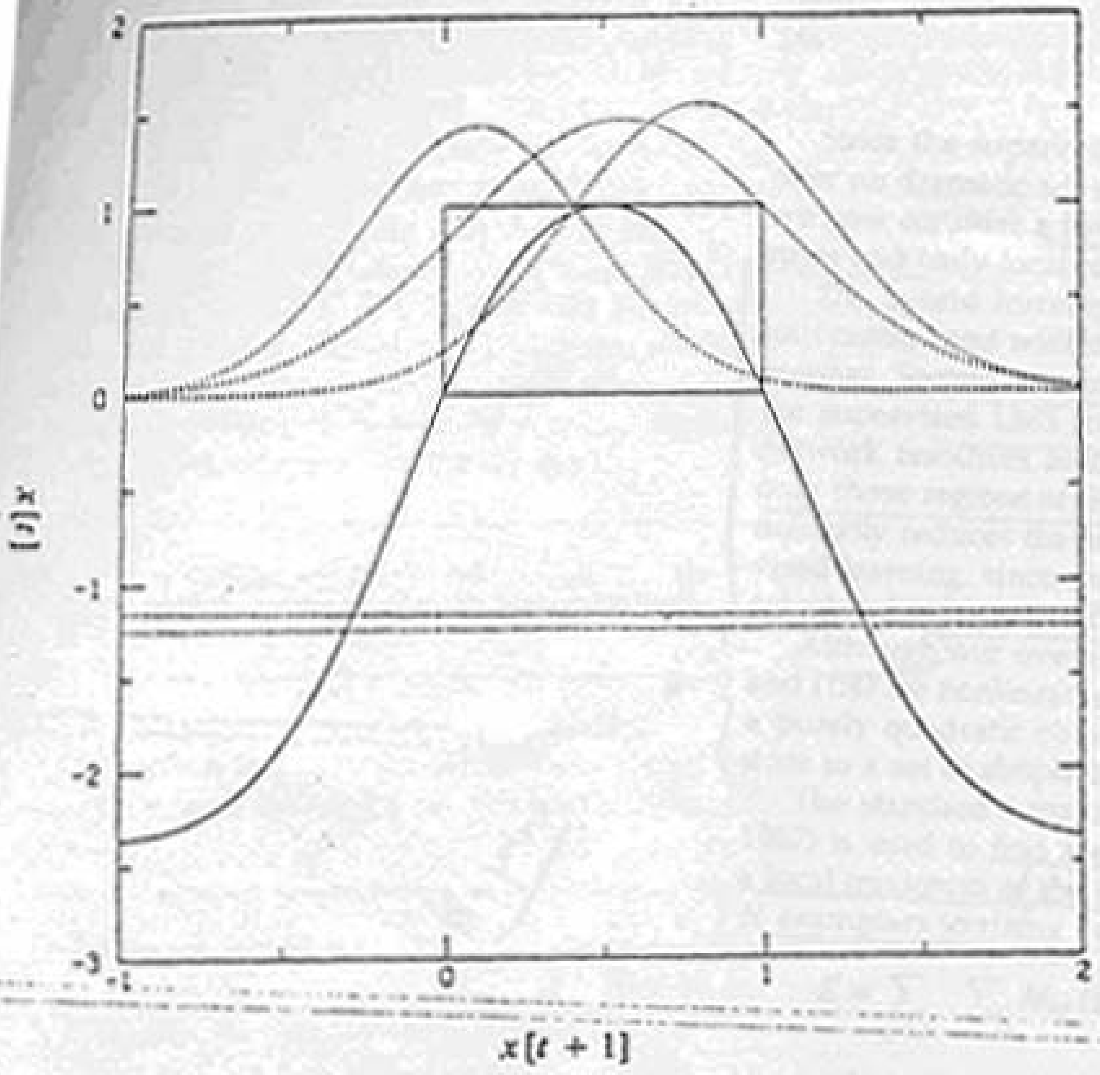

Figure 2: Supervised learning solution found by a network of 5 gaussian units (dashed lines) for the quadratic logistic mapping. The envelope of the gaussians (solid line) for the region in which training data is present (the interval [0, 1]) is bracketed by the rectangle. Notice that two of the gaussians have inverted and flattened out to provide a near-constant offset.

Note that the networks with 5 and 6 gaussian units achieve better performance than the more traditional sigmoidal network and do so with a comparable number of network parameters. This is probably because the gaussian shape is better suited to approximating the logistic map. Unfortunately, the gaussian units do not learn appreciably faster than the backpropagation network, and there is no reason why they should have. Furthermore, the gaussians assumed large widths, thus losing the locality intended in equation (1.1). A sample solution is shown in figure 2; see (Lapedes and Farber 1987) for a sample solution found by a sigmoidal net.

Since the supervised learning method with gaussian units seems to offer no dramatic advantages over a standard backpropagation network, we now consider a hybrid learning method which yields linear learning rules and truly local representations.

The hybrid learning process works as follows. First, the processing unit centers and widths are determined in a bottom-up or self-organizing manner. Second, the amplitudes are found in a top-down manner using the supervised LMS rule. The bottom-up component serves to allocate network resources in a meaningful way by placing the unit centers in only those regions of the input space where data is present. It also dramatically reduces the amount of computational time required by supervised learning, since only the output weights (unit amplitudes) must be calculated using an error signal.

Although our overall network response functions in equations (1.1) and (1.4) are nonlinear, each of our learning rules is based on minimizing a purely quadratic objective function. This reduces the learning procedure to a set of simple linear update rules.

The standard $k$-means clustering algorithm (Lloyd 1957; MacQueen 1967) is used to find a set of $k$ processing unit centers which represent a local minimum of the total squared euclidean distances $E$ between the $N$ exemplars (training vectors) $\vec{x}_i$ and the nearest of the $k$ centers $\vec{x}_\alpha$:

$$E = \sum_{\alpha=1,k} \sum_{i=1,N} M_{\alpha i} (\vec{x}_\alpha - \vec{x}_i)^2 . \tag{1.6}$$

Here, $M_{\alpha i}$ is the cluster membership function, which is a $k \times N$ matrix of 0's and 1's with exactly one 1 per column which identifies the processing unit to which a given exemplar belongs. The (local) minimization of $E$ can be formulated as an iterative process on a complete training set (Lloyd 1957) or as a real-time, adaptive process (MacQueen 1967). A variant of adaptive $k$-means is presented below in equation (1.7).

The processing unit widths are determined using various "$P$ nearest-neighbor" heuristics. These heuristics vary the widths in order to achieve a certain amount of response overlap between each unit and its neighbors so that they form a smooth and contiguous interpolation over those regions of the input space which they represent. The heuristics can be formulated in an adaptive fashion and can be shown to minimize various objective functions with respect to the $\sigma^\alpha$'s or $(\sigma^\alpha)^2$'s and are thereby stable. Some of the possible objective functions are quadratic, leading to simple linear update rules. A particularly simple example is the "global first nearest-neighbor" heuristic. It uses a uniform average width $\sigma = \langle \Delta x_{\alpha\beta} \rangle$ for all units where $\Delta x_{\alpha\beta}$ is the euclidean distance in the input space between each unit $\alpha$ and its nearest-neighbor $\beta$ and $\langle\rangle$ indicates a global average over all such pairs. Other heuristics based on purely local computations yield individually-tuned widths $\sigma^\alpha$.

The response function amplitudes (output weights $A^\alpha$ in equations (1.1) or (1.4)) are varied to minimize either the total error in equation

(1.5) or in the case of real time implementations, an instantaneous estimate of the error. We have found that convergence occurs very quickly. This is possible because the self-organized learning which precedes the supervised learning has already done most of the work.

To demonstrate the practical use of the hybrid learning method, we consider two representative test problems: the classification of phonemes and the prediction of a chaotic time series.

The phoneme classification problem is to classify 10 distinct vowel sounds on the basis of their first and second formant frequencies. The data (provided by Huang and Lippmann (1988)) consists of a training set with 338 phoneme exemplars and a test set with 333 exemplars. Using standard (off-line) k-means, an overlap parameter $P = 2$, and off-line LMS, the classification results for the hybrid learning system are:

| # Gaussian Units | 20 | 40 | 60 | 80 | 100 |
|---|---|---|---|---|---|
| % Error on Test Set | 26.7% | 24.9% | 21.3% | 19.5% | 18.0% |

These results are comparable to those found by Huang and Lippmann using a variety of techniques. For convenience, we reproduce their results here:

| Classification Method | % Error on Test Set | Number of Training Tokens |
|---|---|---|
| K Nearest Neighbors (Non-Adaptive) | 18.0% | 338 |
| Gaussian Classifier (Non-Adaptive) | 20.4% | 338 |
| 2-Layer Back Prop (Adaptive) | 19.8% | 50,000 |
| Feature Map, 100 Nodes (Adaptive) | 22.8% | 10,000 (Feature Map Nodes)<br>50 (Output Nodes) |

The most similar model to ours is the feature map classifier (Kohonen 1988). Our model achieves better classification results given the same number of nodes (in this case 100), because the gaussian response functions yield smooth interpolations of the classification regions, rather than sharp discontinuities from one cluster region to the next. (Furthermore, the feature map classifier makes an intrinsic assumption about the underlying dimensionality of the problem, typically producing either a one or two-dimensional map. A two-dimensional map is appropriate for this phoneme classification problem, but is unlikely to be the optimal choice for arbitrary domains. In contrast, the $k$-means algorithm discovers the intrinsic dimensionality of the input data.) Both the feature map and the locally-tuned networks learn substantially faster than backpropagation nets (typically factors of hundreds in CPU time).

Although our results above are for off-line learning, the adaptive $k$-means clustering algorithm is actually simpler than the off-line version and yields solutions of similar quality. Furthermore, adaptive clustering methods are being implemented in special purpose hardware, and are likely to be very important for real-time learning systems. Figure 3 shows an example of real-time clustering using the phoneme training exemplars. The initial cluster centers are randomly chosen exemplars. At each time step, a random exemplar $\vec{x}^i$ is chosen and the nearest cluster center $\vec{x}^\alpha$ is moved by an amount:

$$\Delta \vec{x}^\alpha = \eta(\vec{x}^i - \vec{x}^\alpha), \tag{1.7}$$

where $\eta = 0.03$ is the learning rate. The resulting trajectories for 20 cluster centers are shown for a run of 3000 exemplar samples. The final positions are indicated by triangles, and the circles represent the widths of the gaussians as determined by the $P = 2$ nearest neighbors heuristic.

As a second representative test problem, we follow Farmer and Sidorowich (1987) and consider the prediction of a chaotic time series. As it is usually formulated, this problem requires finding a real-valued mapping $f : \mathcal{R}^n \mapsto \mathcal{R}$ which takes a sequence of $n$ recent samples of a time series and predicts the value of the time series at a future moment. It is assumed that the underlying process which generates the time series is unknown. We shall compare our network's learning and generalizing capabilities to a three-layer perceptron studied by Lapedes and Farber (1987) (see figure 1b). The particular time series we use for comparison results from integrating the Mackey-Glass differential delay equation:

$$\frac{dx[t]}{dt} = -b\,x[t] + a\,\frac{x[t-\tau]}{1 + x[t-\tau]^{10}}. \tag{1.8}$$

Figure 4 shows the resulting time series for $\tau = 17$, $a = 0.2$, and $b = 0.1$; note that it is quasi-periodic since no two cycles are the same. The characteristic time of the series, given by the inverse of the mean of the power spectrum, is $t_{char} \approx 50$. Note that classical techniques like global linear autoregression or Gabor-Volterra-Wiener polynomial expansions typically do no better than chance at predicting such a time series beyond $t_{char}$.

For our numerical comparison, both networks (Fig. 1) have four real-valued inputs ($x[t], x[t-\Delta], x[t-2\Delta], x[t-3\Delta]$) and one real-valued output $x[t+T]$ with $\Delta = 6$ and $T = 85 > t_{char}$. The network of locally-tuned units (Fig. 1a) has between 100 and 10,000 internal units arranged in a single layer, while the backpropagation network (Fig. 1b) has two internal layers each containing 20 sigmoidal units. The backpropagation network thus has 541 adjustable parameters (weights and thresholds) total.

Figure 5 contrasts the prediction accuracy $E$ (Normalized Prediction Error) versus number of internal units for three versions of our algorithm to the backpropagation benchmark (A) of Lapedes and Farber (1987). The three versions of the learning algorithm are:

1. Nearest neighbor prediction. Here, the nearest data point in the training set is used as a predictor. This behavior is actually a special case for the network of equation 1.4 where each input/output training pair $\{\vec{x}_i, f_i\}$ defines a processing unit $\{\vec{x}^\alpha, f^\alpha\}$ of infinitely narrow width ($\sigma^\alpha \rightarrow 0$).

2. Adaptive processing units with one unit per data point. Here, the amplitudes are determined by LMS, the widths by the global first

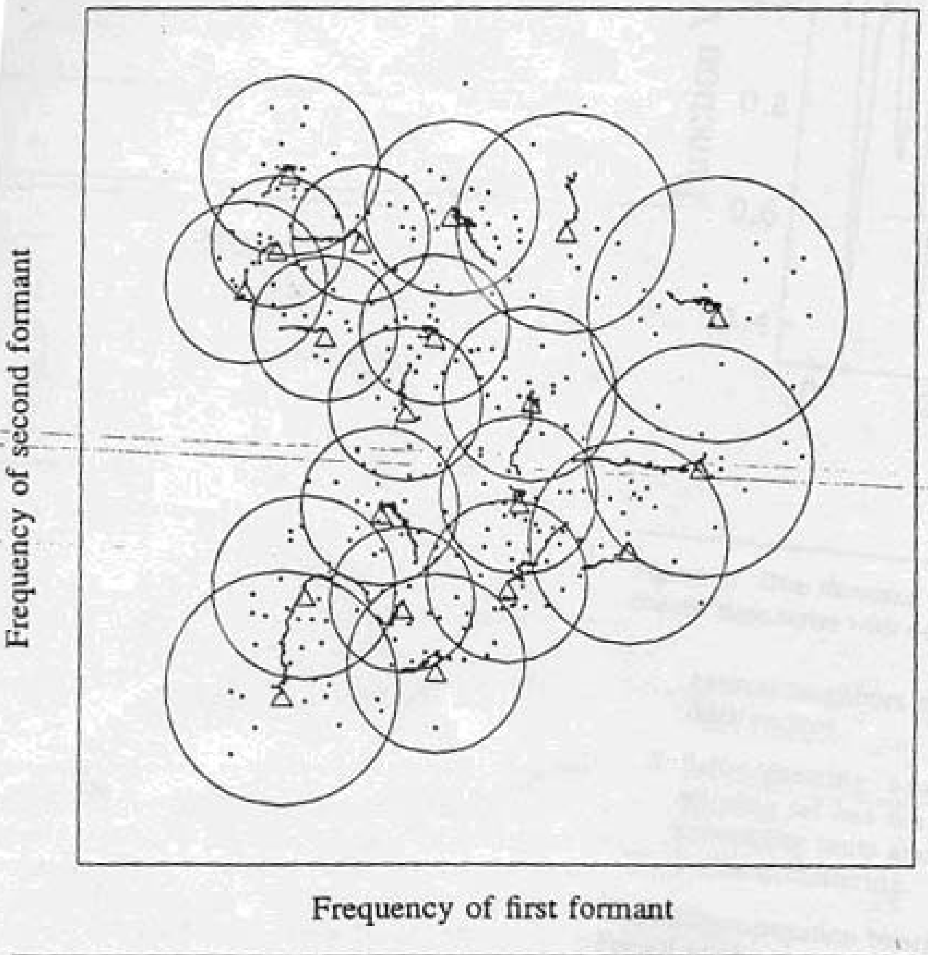

Figure 3: Adaptive $k$-means clustering applied to phoneme data.

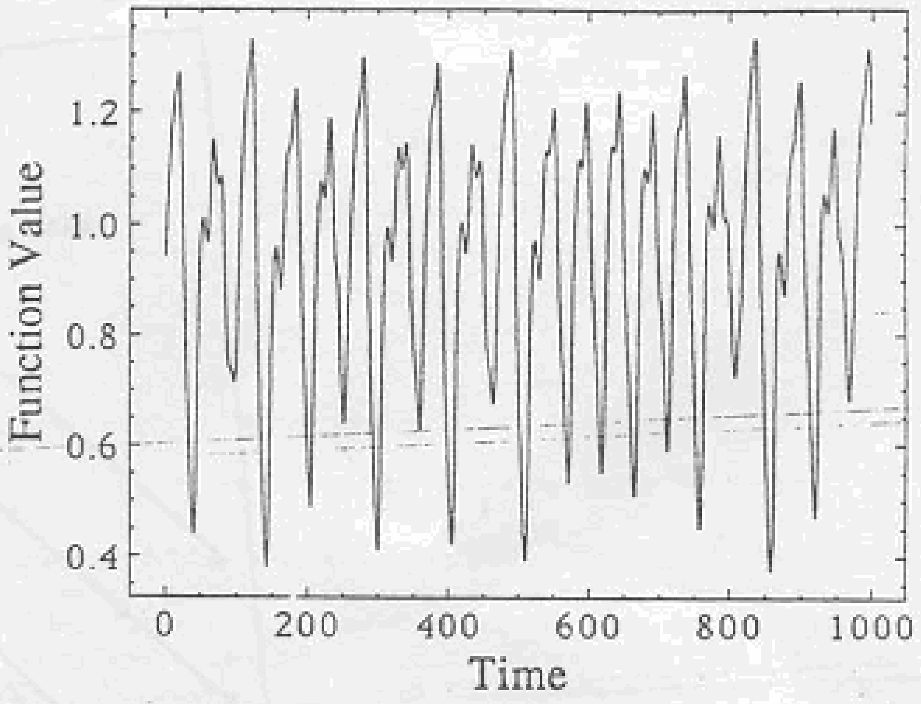

Figure 4: One thousand successive integer timesteps for the Mackey-Glass chaotic time series with delay parameter $\tau = 17$.

nearest neighbors heuristic, and the centers are fixed to be training data vectors.

3. Self-organizing, adaptive processing units. Similar to 2, but the training set has ten times more exemplars than the network has processing units and the processing unit centers are found using $k$-means clustering.

The backpropagation benchmark used a training set with 500 exemplars. For all methods, prediction accuracies were measured on a 500-member test set.

Note that versions 1 and 2 require storage of past time series data; version 2 assigns a processing unit to each data point. Hence, neither of these methods is appropriate for real-time signal processing with fixed memory. However, version 3 is fully adaptive in that a fixed set of network parameters can be varied in response to new data in real time and does not, in principle, require storage of previous data. In order to make a *fair* comparison to the backpropagation benchmark, however, we optimized our networks on a fixed training set, rather than measure time-averaged, real-time performance.

As is apparent from figure 5 (note horizontal reference line), method 2 achieves a prediction accuracy equivalent to backpropagation with about 7 times as much training data, while method 3 requires about 27 times

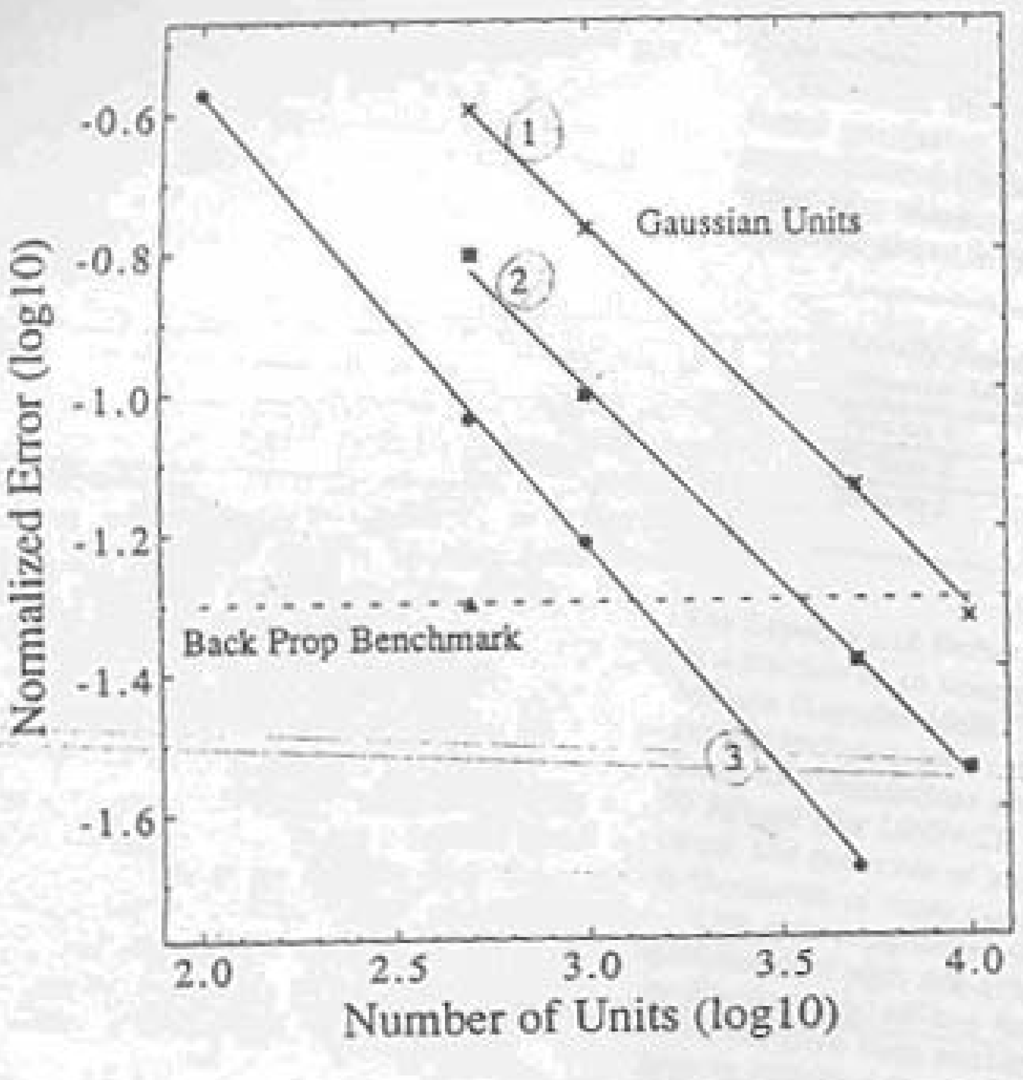

Figure 5: Comparison of prediction accuracy vs. number of internal units for four methods: (1) first nearest neighbor, (2) adaptive units (one unit per training vector), (3) self-organizing units (ten training vectors per processing unit), and (A) backpropagation. The methods are described in the text. For backpropagation, the abscissa indicates the number of training vectors. The horizontal line associated with (A) is provided for visual reference and is not intended to suggest a scaling law. In fact, the scaling law is not known.

as much data to reach equivalent accuracy. These differences in training data requirements stem from the fact that a backpropagation network is fully supervised, learns a global fit to the function, and is thus a more powerful generalizer, while the network of locally-tuned units learns only local representations.

However, the larger data requirements of the networks of locally-tuned processing units are outweighed by dramatic improvements in computational efficiency. The following table shows $E$ versus computational time measured in Sun 3/60 seconds for each of the three methods described above in the 1000 internal units case.

| Locally-Tuned Network Model | Normalized Prediction Error | Computation Time (Sun 3/60 CPU secs) |
|---|---|---|
| Version 1 | 17.1% | 67 secs (0.019 hours) |
| Version 2 | 9.9% | 229 secs (0.064 hours) |
| Version 3 | 6.1% | 1858 secs (0.51 hours) |

The Lapedes and Farber backpropagation results required a few minutes to a fraction of an hour of Cray X/MP time running at an impressive 90 MFlops (Lapedes 1988). Their implementation used the conjugate gradient minimization technique and achieved approximately 5% Normalized Error. Our simulations on the Sun 3/60 probably achieved no more than 90 KFlops (the LINPACK benchmark). Taking these differences into account, the networks of locally-tuned processing units learned hundreds to thousands of times faster than the backpropagation network.

Our own experiences with backpropagation applied to this problem are consistent with this difference. Using a variety of methods including on-line learning, off-line learning, gradient descent, and conjugate gradient, we have been unable to get a backpropagation network to come close to matching the prediction accuracy of locally-tuned networks in a few hours of Sun 3 time.

Although our discussion has focused on algorithms which can be implemented as real-time adaptive systems, such as backpropagation and the networks of locally-tuned units we have presented, a number of off-line algorithms for multidimensional function modeling achieve excellent performance both in terms of efficiency and precision.

These include methods of exact interpolation based on rational radially-symmetric basis functions (Powell 1985). In spite of the apparent locality of their functional representations, these methods are actually global since the basis functions do not drop off exponentially fast. Furthermore, they require a separate basis function for each data point and require computing time which scales as $N^3$ in the size of the data set. Finally, these algorithms are not adaptive.

Approximation methods based on local linear and local quadratic fitting have been championed recently by Farmer and Sidorowich (1987). These algorithms utilize local representations in the input space, but are not appropriate for real-time use since they require multi-dimensional tree data structures which are cumbersome to modify on the fly and would be extremely difficult to implement in special purpose hardware.

The off-line methods require explicit storage of past data and assume that all such data is retained. In contrast, the neural net approach requires storage of only a fixed set of tunable network parameters; this number is independent of the total amount of data observed over time.

An alternative adaptive approach based upon hashing and a hierarchy of locally-tuned representations has been explored by Moody (1989a) with very favorable results.

## Note Added in Proof

After this manuscript was accepted for publication, we learned that Hanson and Burr (1987) had suggested using a single layer of locally-tuned units in place of two layers of sigmoidal or threshold units. This was also suggested independently by Lapedes and Farber (1987b). These authors, along with Lippmann (1987), describe constructions whereby localized "bumps" or convex regions can be built from two layers of sigmoidal or threshold units respectively.

## Acknowledgments

We gratefully acknowledge helpful comments from and discussions with Andrew Barron, Doyne Farmer, Walter Heiligenberg, Alan Lapedes, Y.C. Lee, Richard Lippmann, Bartlett Mel, Demetri Psaltis, Terry Sejnowski, John Shewchuk, John Sidorowich, and Tony Zador. We furthermore wish to thank Bill Huang and Richard Lippmann for kindly providing us with the phoneme data. This research was supported by ONR grant N00014-86-K-0310, AFOSR grant F49620-88-C0025, and a Purdue Army subcontract.

## References

Farmer, J.D. and J.J. Sidorowich. 1987. Predicting chaotic time series. *Physical Review Letters*, 59, 845.

Hanson, S.J. and D.J. Burr. 1987. Knowledge representation in connectionist networks. Bellcore Technical Report.

Huang, W.Y. and R.P. Lippmann. 1988. Neural net and traditional classifiers. *In:* Neural Information Processing Systems, ed. D.Z. Anderson, 387–396. American Institute of Physics.

Kohonen, T. 1988. *Self-organization and Associative Memory*. Berlin: Springer-Verlag.

Lapedes, A.S. 1988. Personal communication.

Lapedes, A.S. and R. Farber. 1987. *Nonlinear signal processing using neural networks: Prediction and system modeling*. Technical Report, Los Alamos National Laboratory, Los Alamos, New Mexico.

———. 1988. How neural nets work. *In:* Neural Information Processing Systems, ed. D.Z. Anderson, 442–456. American Institute of Physics.

Lippmann, R.P. 1987. An introduction to computing with neural nets. *IEEE ASSP Magazine*, **4:4**.

Lloyd, S.P. 1957. Least squares quantization in PCM. Bell Laboratories Internal Technical Report. *IEEE Transactions on Information Theory*, **IT-28:2**, 1982.

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. *In:* Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics, and Probability, eds. L.M. LeCam and J. Neyman. Berkeley: U. California Press, 281.

Moody, J. 1989a. Fast learning in multi-resolution hierarchies. *In:* Advances in Neural Information Processing Systems, ed. D. Touretzky. Morgan-Kaufmann, Publishers.

———. 1989b. In preparation.

Moody, J. and C. Darken. 1988. Learning with localized receptive fields. *In:* Proceedings of the 1988 Connectionist Models Summer School, eds. Touretzky, Hinton, and Sejnowski. Morgan-Kaufmann, Publishers.

Omohundro, S. 1987. Efficient algorithms with neural network behavior. *Complex Systems*, **1**, 273.

Powell, M.J.D. 1985. *Radial basis functions for multivariate interpolation: a review.* Technical Report DAMPT 1985/NA12, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Silver Street, Cambridge CB3 9EW, England.

Received 10 October 1988; accepted 27 October 1988.