Learning with Localized Receptive Fields

John Moody and Christian Darken

Research Report YALEU/DCS/RR-649
September 1988

This research report will appear in the Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann, Publishers 1988. The work was supported by ONR grant N00014-86-K-0310, AFOSR grant F49620-88-C0025, and a Purdue Army subcontract.

Learning with Localized Receptive Fields

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

Abstract

We propose a network architecture based upon localized receptive field units and an efficient method for training such a network which combines self-organized and supervised learning. The network architecture and learning rules are appropriate for real-time adaptive signal processing and adaptive control. For a test problem, predicting a chaotic timeseries, the network learns 1000 times faster in digital simulation time than a three layer perceptron trained with back propagation, but requires about ten times more training data to achieve comparable prediction accuracy.

1 Introduction: Networks of Localized Receptive Fields

Over the past forty-five years, most neural network models have utilized the McCulloch and Pitts (MP) neuron [17] or generalizations thereof as the basic computational unit. The MP neuron is a simple threshold device which uses analog summation of inputs to determine a boolean output. (The original formulation also provides for shunting inhibition to veto a net positive input.) Some of the more recently-studied refinements of MP neurons include sigmoidal response functions (soft thresholds), conjunctive or "sigma-pi" input summation, and explicit representation of the temporal response characteristics. The most popular MP-neuron-based network models have used layered architectures, particularly the multi-layer perceptrons, and supervised learning algorithms such as the perceptron, Adaline, Boltzmann machine, and back propagation learning rules.

The most successfully applied network learning model has been the LMS or Adaline learning rule [27]. Formulated in 1959, it has enjoyed widespread application in adaptive signal processing, particularly in telecommunications. (See reference [28].) The most successfully-applied modern network learning model has been back propagation, the multilayer generalization of the Adaline rule. This supervised learning procedure has been used to solve a number of interesting problems, for example sonar target recognition[4], playing backgammon[26], protein secondary structure prediction[22], and predicting chaotic timeseries[11].

In spite of its practical success, back propagation suffers a major handicap in that it is extremely inefficient. In its original formulation based upon gradient descent, it converges very slowly. When implemented using a faster optimization procedure, conjugate gradient, it is still painfully slow. Typical convergence times for both methods on large problems are measured in VAX days or Cray hours. While more refined numerical implementations of back propagation are likely to yield speed improvements, the ultimate learning rate is still limited by the fact that *all* layers of weights in the network are determined by minimization of an error

signal which is specified *only* as a function of the output. A substantial fraction of the learning time required by these networks is spent in the discovery of internal representations, the patterns of incoming connections to internal units. While this fully supervised approach makes sense for doing "off-line" numerical optimization when very precise results are desired, it is probably too inefficient for many real-time problem domains found in such areas as adaptive signal processing or biological information processing.

In a departure from standard network models, we shall consider in this paper networks which are not based upon the MP neuron. Rather, we shall examine networks of units which are imbued *a priori* with simple internal representations. Specifically, we shall describe networks which use a single internal layer of localized receptive field units to perform the essential computation.

Locally-tuned, but overlapping receptive fields are a well-known data structure in real nervous systems. They have been studied in many of regions of cerebral cortex, including the auditory cortex, the visual cortex, and the somatosensory cortex. The orientation selective cell in the visual cortex is perhaps the canonical example [7]. Given that the brain is built out of noisy and imperfect components, the overlapping receptive field representation is believed to be important for improving signal to noise ratios and for providing fault tolerance. It is also invoked to explain the phenomenon of hyperacuity. Similar benefits are likely to be achieved by using this representation in analog computational systems.

In addition to the data representations provided by receptive fields, several authors have studied their development and plasticity. These include physiological [18] and computational [21,24] studies of plasticity in somatosensory cortex and computational models of the development of orientation selective cells in visual cortex [16,12].

We shall consider networks of abstract localized receptive fields which attempt to capture the computational flavor of biological receptive fields. Furthermore, we shall try to capture the plasticity which has been observed and modeled in receptive field maps by formulating self-oganizing dynamics which re-allocate receptive field processing resources to those regions of the input space which are important.

2 Representation of Real-valued Functions

In this section, we show how a network of localized receptive fields can represent real-valued functions $f : \mathcal{R}^n \mapsto \mathcal{R}^p$. The network we shall discuss (see figure 1a.) has n linear inputs and p linear output units (only one of each are shown). The internal units are a single layer of M receptive fields which have localized response functions in the input space. The network thus has two layers of connections. The overall response function of the network (for the specialization $p = 1$) is:

$$f(\vec{x}) = \sum_{\alpha=1}^{M} f^\alpha R^\alpha(\vec{x}) \ , \qquad (1)$$

$$R^\alpha(\vec{x}) \equiv R(\|\vec{x} - \vec{x}^\alpha\|/\sigma^\alpha) \ . \qquad (2)$$

Here, $\vec{x}$ is a real-valued vector in the input space, R^α is the α-th receptive field response function, R is a radially-symmetric function (such as a gaussian) with a single maximum at the origin and which drops off to zero at large radii, $\vec{x}^\alpha$ and σ^α are the center and width of the α-th receptive field, and f^α is the function value associated with each receptive field. One can think of this functional representation as a generalized deconvolution in which the basis functions (receptive fields) are not uniformly distributed over the input space and do not have uniform

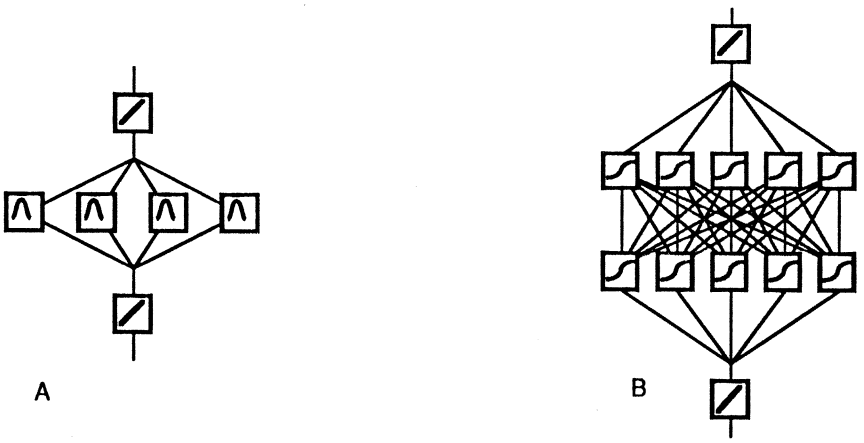

Figure 1: A network of localized receptive field units (A) and a three layer perceptron (B). Unit response functions are depicted graphically. Both networks have linear inputs and outputs. The receptive field network has a single internal layer of non-linear units, while the perceptron has two internal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

width. The use of receptive field representations of form (1) for modeling simple real-valued functions has recently been discussed by Heiligenberg and Baldi [5,1].

To obtain well-localized representations, it is important that the receptive field response functions (2) approach zero rapidly at large radii. We have primarily used gaussians with unit normalization:

$$R^\alpha(\vec{x}) = \exp\left[-\frac{\|\vec{x}-\vec{x}^\alpha\|^2}{(\sigma^\alpha)^2}\right] \quad , \qquad (3)$$

which drop off faster than exponentially. One could also use logistic functions of the form:

$$\frac{1}{1+\exp\left[\|\vec{x}-\vec{x}^\alpha\|/\sigma^\alpha - \theta^\alpha\right]} \quad , \qquad (4)$$

which drop off exponentially. Other functional forms, such as the lorentzian, which drop off as low order power laws are best avoided because of their poor localization properties. The use of such functions dramatically increases the computational expense of simulating receptive field networks.

In network language, one can think of the f^α's as ordinary linear output weights. If the receptive field response function is a gaussian as in equation 3, we can think of the computational unit ("neuron") as having an exponential response function and the inputs as being a sum of post-synaptic responses of quadratic connections scaled by a gain. Thus, the components of the $\vec{x}^\alpha$ are interpreted as input connection values and $(1/\sigma^\alpha)^2$ is interpreted as a neuronal gain. Although this interpretation is not biologically correct, it is useful for comparing the algorithmic aspects of receptive field networks and multi-layer perceptrons. Furthermore, this formulation may be implementable directly in analog hardware.

With the addition of lateral connections between the receptive field units, the network can dynamically produce the normalized response function:

$$f(\vec{x}) = \frac{\sum_\alpha f^\alpha R^\alpha(\vec{x})}{\sum_\alpha R^\alpha(\vec{x})} \; . \qquad (5)$$

This is essentially a weighted average or interpolation between nearby receptive field

function values; it is the functional form we use in our simulations. The dynamical process which results in this output will be discussed elsewhere.

3 Comparison to Multi-Layer Perceptrons

In this section, we shall contrast the receptive field network to a back propagation network with 3 layers of weights, 2 internal layers of graded threshold units, and linear inputs and outputs. Both networks can learn to approximate real-valued mappings. Although the receptive field network is architecturally less complicated, it can easily learn smooth approximations to arbitrary functions on $\mathcal{R}^n$ with only one internal layer, while a perceptron architecture is believed to require three layers (two internal layers) of threshold units.

It is well-known that three layers of threshold units (two hidden layers) are sufficient to solve arbitrary classification problems in $\mathcal{R}^n$ ($n \geq 2$) (see Lippmann [14]). This point is apparent when one considers that at least two layers of threshold units are required to separate a convex region of the input space. Clearly, just one layer of receptive field units can separate convex regions. Arbitrary decision regions (non-convex, non-simply-connected, or disconnected) can be formed as unions of convex regions. This union is accomplished by the output layer.

These results for arbitrary classification problems can be trivially extended to the approximation of arbitrary real-valued functions. The approximation scheme is based on assigning function values to convex regions of the input space. This requires two internal layers of threshold units or just one layer of receptive field units.

Indeed, Lapedes and Farber[10] have speculated that networks of threshold units may tend to "discover" receptive field representations when trained to approximate real-valued functions. They refer to these network configurations as "bumps". Based upon our own experience, we believe that the formation of bumps is unlikely. This point will be discussed in greater detail elsewhere.

The intrinsically-local representation of the input space by receptive field units has several advantages. First, it results in problem solutions which are intuitively transparent. Secondly, it allows the implementation of very fast simulations, since only those regions of the network (only those receptive fields) which respond to a given input need to be updated during either learning or performance modes. And thirdly, it allows the use of self-organized learning techniques analogous to statistical clustering to train the input parameters (receptive field centers) of the network. This dramatically reduces overall learning time. These techniques are discussed in section 4.

4 Self-Organizing Receptive Fields

Learning in a localized receptive field network can be formulated as a two stage hybrid process. The receptive field centers and widths can be determined in a bottom-up or self-organizing manner, while the receptive field amplitudes are found in a top-down manner using the supervised LMS rule. This serves to allocate network resources in a meaningful way by placing the receptive field 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 (receptive field amplitudes) must be calculated using an error signal.

Our formulation uses the venerable k-

means clustering algorithm (or an adaptive generalization thereof) to find the receptive field centers and a contiguity heuristic to determine the receptive field widths. Note that the k-means clustering algorithm is a predecessor to the more recently proposed "feature maps" [8] and "competitive learning" [25] algorithms.

The standard euclidean k-means clustering algorithm finds a set of k cluster centers which represent a local minimum of the total squared euclidean distances E between the k cluster points $\vec{x}_\alpha$ and N exemplars (training vectors) $\vec{x}_i$:

$$E = \sum_{\alpha=1,k} \sum_{i=1,N} M_{\alpha i} (\vec{x}_\alpha - \vec{x}_i)^2 , \qquad (6)$$

where $M_{\alpha i}$ is the cluster membership function, which is a $k \times N$ matrix of 0's and 1's with exactly one 1 per column. The positions of the cluster points $\vec{x}_\alpha$ are varied to minimize E. The cluster membership function is redetermined in an iterative fashion after each minimization of E such that each exemplar point belongs to the nearest cluster point's cluster. The cycle of E minimization and M redetermination is iterated until E and M no longer change. This state is a *local* minimum of E.

An adaptive formulation of k-means requires no storage of past exemplars $\vec{x}_i$ or the cluster membership function $M_{\alpha i}$. This formulation, suitable for real-time applications or analog implementation, uses only the current exemplar $\vec{x}[t]$ as a training signal to modify the nearest cluster vector $\vec{x}_{\text{closest}}$ according to

$$\frac{\partial}{\partial t} \vec{x}_{\text{closest}}[t] = \eta \, (\vec{x}[t] - \vec{x}_{\text{closest}}[t]) , \qquad (7)$$

where η is a learning rate. The determination of which of the $\vec{x}_\alpha$ is $\vec{x}_{\text{closest}}$ can be accomplished in $\log k$ time on a serial computer, constant time on a parallel computer, or approximately constant time in an analog system via a winner-take-all circuit. The learning rate η can be set initially large and then gradually reduced toward zero according to an "annealing schedule".

Once the receptive field centers are found using k-means, their widths can be determined by minimizing the following objective function with respect to the σ^α's :

$$E = \frac{1}{2} \sum_{\alpha=1}^{k} \left[\sum_{\beta=1}^{k} R^\alpha(\vec{x}_\beta) \left(\frac{\vec{x}_\alpha - \vec{x}_\beta}{\sigma^\alpha} \right)^2 - P \right]^2 , \qquad (8)$$

where P is an overlap parameter. Note that this objective function depends only on the locations of the receptive field centers and widths and does *not* depend directly on the input training vectors $\vec{x}_i$. The effect of minimizing this function is to ensure that the receptive field units form a smooth and contiguous interpolation over those regions of the input space which they represent.

Qualitatively similar results to those provided by equation 8 can be obtained more directly by requiring that the width of a given receptive field be set equal to the root mean square value $\langle x^\alpha \rangle_P$ of the Euclidean distances to its P nearest neighboring receptive fields, We call this the "P nearest neighbors" heuristic. It is the solution found by replacing $\sum_{\beta=1}^{k} R^\alpha(\vec{x}_\beta)$ with $\sum_{\text{P nearest}}$ in equation 8. We have found that choosing $P = 1$ and using a uniform global average width σ for all receptive fields α gives near-optimal results for some problems. We call this the "global first nearest neighbor" heuristic and use it in the timeseries prediction problem described in later sections.

After the receptive field centers and widths have been found using these self-organizing techniques, the output weights (receptive field amplitudes) are found using the supervised Adaline or LMS learning rule. The output weights f^α in equations 1 or 5 are varied to minimize the total error:

$$E = \frac{1}{2} \sum_{i=1,N} (f^*(\vec{x}_i) - f(\vec{x}_i))^2 \qquad (9)$$

where $\vec{x}_i$ is the i-th training pattern, f is the network output, and f^* is the desired result. We have found that the supervised learning process converges very quickly using a quasi-Newton method. This is possible because the self-organized learning which preceeds the supervised learning has already done most of the work.

An adaptive formulation of the LMS rule appropriate for real-time or analog implementations is

$$\frac{\partial}{\partial t} f^\alpha[t] = \eta \left(f^*(\vec{x}[t]) - f(\vec{x}[t]) \right) R^\alpha(\vec{x}[t]) \; . \tag{10}$$

5 Timeseries Prediction: A Benchmark Problem

We have chosen timeseries prediction as our test problem for three reasons. First, the signal processing domain is likely to be an important area of application of neural networks, both analog and digital. This is an area where high speed and real-time adaptive algorithms are of great use. Secondly, the particular timeseries which we have chosen has intrinsic interest and is quite difficult to predict. It is the chaotic timeseries generated by the Mackey-Glass differential delay equation. Thirdly, the Mackey-Glass problem serves as a useful benchmark, because it has been well studied in the chaos community and has already been tackled by Lapedes and Farber [11] using a back propagation network.

The Mackey- Glass timeseries [15] is obtained by integrating the following differential-delay equation:

$$\frac{dx[t]}{dt} = -b\, x[t] + a\, \frac{x[t-\tau]}{1 + x[t-\tau]^{10}} \; . \tag{11}$$

Figure 2. shows the resulting timeseries for $\tau = 17$, $a = 0.1$, and $b = 0.2$; 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 $t_{char} \approx 50$.

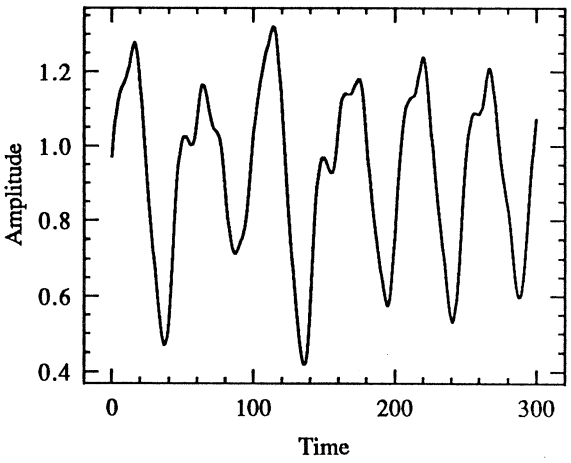

Figure 2: Three hundred successive integer timesteps for the Mackey-Glass chaotic timeseries with delay parameter $\tau = 17$.

The prediction problem is to use the available history of the timeseries up to the present time $x[t]$ to predict its value at a future time $x[t + T]$. The standard method used in timeseries prediction is to imbed D successive points $(x[t - (D - 1)\Delta], \ldots, x[t - \Delta], x[t])$ of the timeseries into a D-dimensional imbedding space and then define a mapping from the imbedding space to a predicted value $x[t + T]$. Classical techniques like global linear autoregression or Gabor-Volterra-Wiener polynomial expansions typically fail [11,3] for $T > t_{char}$, meaning that the normalized prediction error E (defined as [rms prediction error] / [std deviation of series]) is 1.0, so that they are no better at predicting the series than simply using the series' mean value at the predicted value. As will be discussed in the next section, the localized receptive field network is able to achieve $E \approx 0.05$ for $T = 85 \approx 1.7\, t_{char}$.

6 Simulation Results

In this section, we present our simulation results which compare self organized learning with receptive fields to the back-propagation results of Lapedes and Farber for the Mackey-Glass prediction problem. The corresponding networks were shown schematically in figure 1. For our numerical comparison, both networks have four real-valued inputs $(x[t], x[t-\Delta], x[t-2\Delta], x[t-3\Delta])$ and one real-valued output $x[t+T]$. The receptive field network has between 100 and 10,000 internal units arranged in a single layer, while the back propagation network has two internal layers each containing 20 sigmoidal units. The back propagation network has 541 adjustable parameters (weights and thresholds) total. For comparison purposes, we have chosen $\Delta = 6$ and $T = 85$.

Figure 3 contrasts the prediction accuracy E (Normalized Prediction Error) versus number of receptive fields for several different versions (1 to 4) of our algorithm to the back propagation benchmark (A) of Lapedes and Farber [11]. The versions of the receptive field algorithm are arranged in order of increasing computational complexity and prediction accuracy. They 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 a network of form in equation 5 where each input/output training pair $\{\vec{x}_i, f_i\}$ defines a receptive field $\{\vec{x}^\alpha, f^\alpha\}$ of uniform narrow width ($\sigma^\alpha \to 0$).

2. Adaptive receptive fields with one receptive field per data point. Here, the amplitudes are determined by LMS, the widths by the global first nearest neighbors heuristic (see discussion below equation 8), and the centers are chosen to be training data vectors.

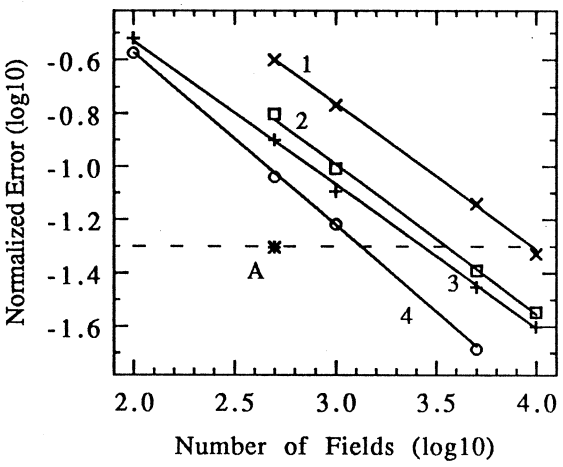

Figure 3: Comparison of Prediction Accuracy *vs.* Number of Receptive Fields for five methods: Nearest-Neighbor (1), Adaptive Receptive Fields (one field per training vector) (2), Adaptive Receptive Fields (ten training vectors per receptive field) (3), Self-Organizing Receptive Fields (4), and Back Propagation (A). The methods are described in the text. For (A), 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.

3. Adaptive receptive fields as in 2, but the training data set is ten times as large as number of receptive fields. The receptive field centers are chosen at random from the training set, while the amplitudes are trained using all data.

4. Self- organizing, adaptive receptive fields. These are similar to 3, but the receptive field centers are found using k-means clustering. Again, the training set has ten times more exemplars than the network has receptive fields.

The back propagation benchmark used a training set with 500 exemplars. The horizontal line in figure 3 is for visual reference and is not intended to imply any scaling law

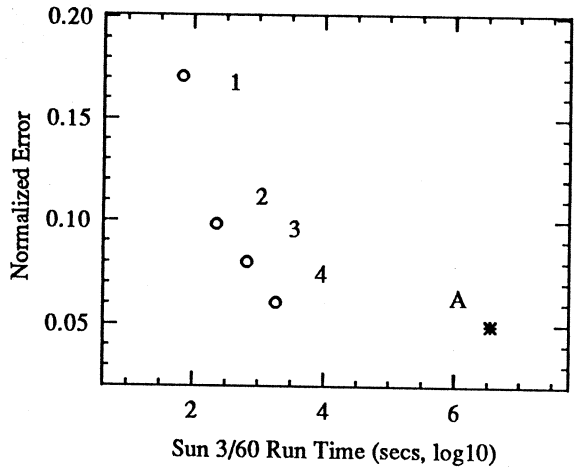

Figure 4: Comparison of Prediction Accuracy *vs.* Simulation Time for five methods: Nearest-Neighbor (1), Adaptive Receptive Fields (one field per training vector) (2), Adaptive Receptive Fields (ten training vectors per receptive field) (3), Self-Organizing Receptive Fields (4), and Back Propagation (A). The methods are described in the text.

for the back propagation result. In fact, the scaling law is not known.

Figure 4 correspondingly shows E versus computational time measured in Sun 3/60 seconds for the 1000 receptive fields case. The labeling is the same as for figure 3. The Lapedes and Farber benchmark required about 1 hour of Cray X/MP time at 90 MFlops [9]. Their implementation used the conjugate gradient minimization technique rather than the more traditional gradient descent. Based upon our experience, conjugate gradient provides convergence speeds which are 10 to 100 times faster than gradient descent. Our simulations on the Sun 3/60 probably achieved about 90 KFlops (the LINPAK benchmark), a factor of 1000 slower. For display purposes, we converted the Cray time into Sun 3/60 time by multiplying by 1000. This means of comparison should not be taken too seriously, because differences in machine architecture, operating system, and detailed software implementation can easily change the results of either the receptive fields methods or back propagation by factors of two, three or perhaps even five. We should note, however, that the Lapedes and Farber implementation was optimized for the Cray, while we made no attempt to fine-tune our implementation on the Sun.

From figure 4, it is clear that the receptive field network achieved comparable precision about **1000 times faster** than the back propagation network implemented with conjugate gradient. The relative speed would probably be a factor of at least 10,000 if compared against back propagation implemented with gradient descent. However, as figure 3 indicates, both the adaptive receptive field and self-organizing receptive field networks required about ten times more data to achieve comparable prediction accuracy.

7 Discussion

Why are the networks of localized receptive fields so much faster? There are basically three reasons. First, since the receptive field representations are well-localized, only a small fraction of the whole network responds to any particular exemplar. This permits the use of very efficient implementation algorithms. We used an adaptive grid [20] to partition the input space and the space of receptive field centers. By contrast, all units must be evaluated and trained for each exemplar in a back propagation network. Secondly, our internal representations permit the use of a two layer network, while the back propagation network required three layers. Thirdly, the combination of self-organized and supervised learning which we have employed for the receptive fields network is much more straightforward than the back propagation procedure. Training all layers of the network by back-propagating an error signal specified only at the output results in slow con-

vergence because of the large number of parameters involved and the non-linearity of the internal units. By contrast, our implementation reduces the learning problem for both input and output layers to simple quadratic minimizations involving small numbers of parameters at the input and only loosely-coupled parameters at the output.

Why do the networks of receptive fields need more data to achieve similar precision to the back propagation network? Basically, the back propagation network is performing a *global*, rather than *local*, fit to the training data. This results in greater generalization from each training example. This strength becomes a weakness when computational efficiency is important.

Which network is better? When data is expensive or hard to obtain, the back propagation approach would be preferred, even though it is computationally inefficient. However, if data is cheap and plentiful, one achieves a big win by using the self-organizing localized receptive fields approach. This latter situation is the one most commonly found in adaptive signal processing or adaptive control, where data is acquired at a high rate and cannot be saved and fast processing is required. It is also the situation faced by nervous systems, in particular sensory and motor systems, where analysis of incoming data must be performed immediately and where there are no anatomical structures for storing raw sensory data for leisurely analysis at a later time.

The prediction of chaotic timeseries is a particularly difficult benchmark problem. Most real-world adaptive signal processing and adaptive control applications are likely to be much simpler. A network of localized receptive fields is likely to be a very useful device for real-time applications where the tried and true Adaline does not have sufficient richness to capture essential non-linear aspects of the processing task.

In addition to real-valued problem domains like signal processing, many problems of interest to the neural networks community are formulated as boolean mapping problems $f : \mathcal{B}^n \mapsto \mathcal{B}^p$. Examples of these are found in references [26] and [22]. In many ways, these problems are typically much easier than the real-valued problems. Generalizations of the receptive field approach to boolean problems are possible. We shall describe these generalizations elsewhere. We expect the dramatically higher efficiency of the self-organizing localized receptive field approach over the conventional back propagation approach to carry over to the boolean domain as well.

A class of problems which are intermediate between the real-valued and boolean problems are classification problems of the kind $f : \mathcal{R}^n \mapsto \mathcal{B}^p$. Several authors have presented algorithms which have qualitative similarities to the localized receptive fields approach and which offer substantial efficiency improvements over back propagation in the classification domain. Huang and Lippmann [6] performed phoneme classification using the Kohonen feature map [8] as a front end and the LMS rule as a back end to attach phoneme labels to feature vectors. They found that the entire process converged about 100 times faster than a two layer back propagation network [13]. Another efficient algorithm for classification problems has been developed by Nestor, Inc. [23].

It should be noted that some efficient digital computational models have recently been proposed which bear some qualitative similarities to the learning with localized receptive fields approach. These algorithms are "local linear prediction", "local quadratic prediction" [3], and "radial basis functions" [2]. The local linear and local quadratic prediction techniques are comparable in simulation speed to learning with receptive fields. While these algorithms are excellent for off-line, non-real-time data anal-

ysis, they are probably not appropriate for real-time information processing of the kind required in adaptive signal processing. This is because they require explicit storage of all incoming data to obtain their high accuracy. This demands large memory. Unlike the neural network approach for which the storage capacity (total number of adjustable network parameters) is fixed in advance, these algorithms operate on potentially unlimited databases. Thus, they are *not* "adaptive" in the neural networks sense in that they do not vary a *fixed set* of internal parameters in response to new data. Furthermore the computations which they require are decidedly non-neural and would be impossible to implement in fixed analog hardware.

Finally, Mackey- Glass prediction results and computation times comparable to those presented here have recently been obtained by Moody [19] using a multi-resolution cerebellar model articulation controller (CMAC) model which incorporates localized receptive field functions to perform improved interpolation.

Acknowledgements

We especially wish to thank one of the editors of this volume, Terry Sejnowski, for a careful proof-reading and many useful comments. We also gratefully acknowledge helpful comments from and discussions with Walter Heiligenberg, Alan Lapedes, Richard Lippmann, Bartlett Mel, John Shewchuk, and John Sidorowich. This research was supported by ONR grant N00014-86-K-0310, AFOSR grant F49620-88-C0025, and a Purdue Army subcontract.

References

[1] P. Baldi and W. Heiligenberg. How sensory maps could enhance resolution through ordered arrangements of broadly tuned receivers. *Biological Cybernetics*, 1988. In press.

[2] M. Casdagli. *Nonlinear Prediction of Chaotic Time Series.* Technical Report, Queen Mary College, London, 1988.

[3] J.D. Farmer and J.J. Sidorowich. *Exploiting Chaos to Predict the Future and Reduce Noise.* Technical Report, Los Alamos National Laboratory, Los Alamos, New Mexico, 1988.

[4] R.P. Gorman and T.J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75, 88.

[5] W. Heiligenberg. Central processing of sensory information in electric fish. *Journal of Comparative Physiology A*, 161:621.

[6] W. Y. Huang and R. P. Lippmann. *Neural Net and Traditional Classifiers.* American Institute of Physics, 1988.

[7] D. Hubel and T.N. Wiesel. Receptive fields, binocular interaction and functional architecture in cat's visual cortex. *Journal of Physiology (London)*, 160:106.

[8] T. Kohonen. *Self-organization and Associative Memory.* Springer-Verlag, Berlin, 1988.

[9] A. Lapedes. 1988. Personal communication.

[10] A. Lapedes and R. Farber. *How Neural Nets Work.* American Institute of Physics, 1987.

[11] A.S. Lapedes and R. Farber. *Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling.* Technical Report, Los Alamos National Laboratory, Los Alamos, New Mexico, 1987.

[12] Ralph Linsker. From basic network principles to neural architecture. *Proc. Nat'l Academy of Sciences USA*, 83, 1986. Series on pages 7508-7512, 8390-8394, 8779-8783.

[13] R.P. Lippmann. 1988. Personal communication.

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

[15] M.C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287.

[16] Ch.v.d. Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14:85, 1973.

[17] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115, 1943.

[18] M.M. Merzenich and J. Kaas. Principles of organization of sensory-perceptual systems in mammals. *Progress in psychobiology and physiological psychology*, 9, 1980.

[19] J. Moody. 1988. In preparation.

[20] S. Omohundro. Efficient algorithms with neural network behavior. *Complex Systems*, 1:273.

[21] J.C. Pearson, L.H. Finkel, and G.M. Edelman. Plasticity in the organization of adult cerebral cortical maps: a computer simulation based on neuronal group selection. *Journal of Neuroscience*, 7(12):4209, 1987.

[22] N. Qian and T. Sejnowski. Predicting the secondary structure of globular protiens using neural network models. *Journal of Molecular Biology*. In press.

[23] D.L. Reilly, C. Scofield, C. Elbaum, and L.N. Cooper. Learning system architectures composed of multiple learning modules. In M. Caudill and C. Butler, editors, *IEEE First International Conference on Neural Networks*, pages II–495, 1987.

[24] H. Ritter and K. Schulten. On the stationary state of kohonen's self-organizing sensory mapping. *Biological Cybernetics*, 54:99, 86.

[25] D.E. Rumelhart and D. Zipser. Feature discovery by competitive learning. *Cognitive Science*, 9:75, 1985.

[26] G. Tesauro and J. Sejnowski. A parallel network that learns to play backgammon. *Artificial Intelligence*, 1988. In press.

[27] B. Widrow and M.E. Hoff. Adaptive switching circuits. In *Wescon Convention Record*, page 96, IRE, 1960.

[28] B. Widrow and R. Winter. Neural nets for adaptive filtering and adaptive pattern recognition. *Computer*, 21:25, 1988.