ON OVERFITTING AND THE EFFECTIVE NUMBER OF HIDDEN UNITS

ANDREAS S. WEIGEND

CU-CS-674-93 September 1993

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

An Overfitting and the Effective Number of Hidden Units

Andreas S. Weigend

CU-CS-674-93 October 1993

Department of Computer Science
University of Colorado at Boulder
Campus Box 430
Boulder, Colorado 80309-0430 USA

CU-CS-674-93

In *Proceedings of the 1993 Connectionist Models Summer School*, edited by
M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend,
pp. 335–342. Hillsdale, NJ: Erlbaum Associates (1994).

ON OVERFITTING AND THE EFFECTIVE NUMBER OF HIDDEN UNITS

Andreas S. Weigend
Department of Computer Science and Institute of Cognitive Science
University of Colorado at Boulder
Boulder, CO 80309-0430
weigend@cs.colorado.edu

The problem of overfitting has drawn significant attention with sometimes apparently contradictory results, even about the mere presence or absence of the effect. In such experiments, the available data is split into two sets: a training set (used for the estimation of the parameters of the model; i.e., the weights and biases of the network), and a test set (used for the estimation of the generalization performance; i.e., the performance on out-of-sample data not used for fitting). An operational definition of overfitting is that the out-of-sample error starts to increase with training time after having gone through a minimum. A standard interpretation is that the network starts, at this point, to pick up idiosyncrasies of the training set that do not generalize to the test set.

This paper consists of two parts. In Part 1, we resolve the apparent contradiction by showing that the very presence (or absence) of overfitting can be related to the use of different test error measures. We also show that the relationship between network size and both shape and value of the overfitting curve is not as straightforward as it might seem: small networks start overfitting before they have learned all they could, and the best solutions are usually obtained with networks with a very large number of potential parameters. In Part 2, we characterize the activity of the hidden units, both individually and as an ensemble. This allows us to introduce an *effective* number of hidden units—in contrast to the *potential* number of hidden units considered in Part 1. All quantities are plotted as functions of the number of epochs; all learning is done with simple error backpropagation.

PART 1: HOW OVERFITTING DEPENDS ON THE ERROR MEASURE AND THE (POTENTIAL) NUMBER OF HIDDEN UNITS

We first briefly describe the architecture and the problem (phoneme classification). We then present three results:

1. *Overfitting can depend on the error measure used for testing. Specifically, we find that the cross-entropy error shows clear overfitting whereas the sum squared error does not.*
2. *Analyzing overfitting as a function of network size (the number of hidden units is varied), we find that fairly small networks (that never reach good performance) also, already, show overfitting.*
3. *Analyzing the performance as a function of network size, we find that large networks generalize better than small ones (the training is stopped in each case at the minimum of the test set).*

Data and Architecture

We illustrate our discussion with quantities obtained from a network trained to classify phonemes. The data is described in detail by Elman and Zipser (1988). The networks we use all have 160 inputs. The input units are fully connected to a varying number of sigmoids with a (0,1) range. The networks all have six outputs, corresponding to the six phonemes given by the task. We use normalized exponentials for the output activations (also called softmax). In order to estimate class probabilities, we choose cross-entropy error: the error for each pattern is chosen to be the logarithm of the activation value of the output unit that corresponds to the target class (Rumelhart et al., still in press). Training starts with small initial weights (uniformly distributed in [-0.01,0.01]). We choose a very small learning rate (a value of 0.01) and no momentum in order to avoid artifacts for the learning curves.

Result 1: **Overfitting depends on how the test error is measured.**

In recent years, conflicting claims have arisen on how prone to overfitting neural networks are when trained with error backpropagation. To clarify this issue, we describe results on a classification experiment (phoneme recognition). In the first setting, we train a network (with 15 hidden units) to minimize cross-entropy, and then test its out-of-sample performance by using several error measures.[1] The results are shown in Figure 1.

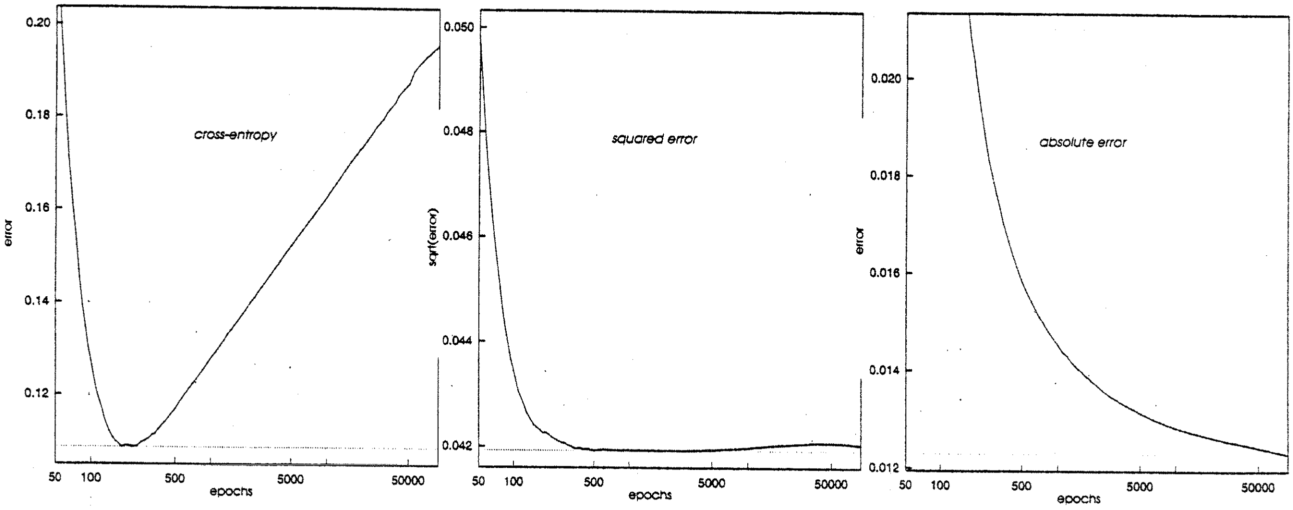

Figure 1. Out-of-sample errors as a function of training time for three error measures: cross-entropy, squared error, and absolute error. These three curves are from one and the same network that was trained with cross-entropy and tested (on the same data in each case) with different error measures.

Figure 1 shows that

- the cross-entropy error *in*creases after passing though a minimum
- the sum squared error remains essentially flat as training continues for a long time
- the sum absolute error still *de*creases after 100,000 epochs.

It turns out that the number of false classifications (defined as the number of cases where the output with the largest activation does not correspond to the target class) remains constant after 500 epochs (at 0.4%). The network becomes overconfident: it moves the output of the target unit farther and farther away from the desired value of 1.0. This has the strongest effect on cross-entropy, since its logarithmic divergence at zero is the least forgiving. The squared error measure is less extreme, and absolute error, as the most robust measure of the three, penalizes large errors in comparison to small errors the least.

So far, we have considered the effect of different functional forms for the error evaluation and found these variations to be sufficient for producing the presence or absence of overfitting. From now on, we evaluate the performance only with cross-entropy. The quantity varied next is the number of hidden units.[2]

[1] In this first experiment, we keep the number of hidden units constant. Another variable, not discussed further in this paper, is the dependence of overfitting on the *size of the training set.* On the one extreme, if only very few training patterns are used (say, 20), no increase in the test error can be observed since the network quickly memorizes these few training patterns without any error. Because there is no error left, the weights do not change any more, and thus the test error does not change either. On the other extreme, if an infinite number of noise-free training patterns explores the test space completely, there would not be any overfitting either. The choice of a training set of 540 patterns throughout this paper corresponds to an intermediate regime where overfitting can occur.

[2] Since inputs and outputs are given by the problem, their numbers cannot be varied for a given problem. A degree of freedom that we did not explore is the number of hidden *layers*; we use throughout a single layer of hidden units.

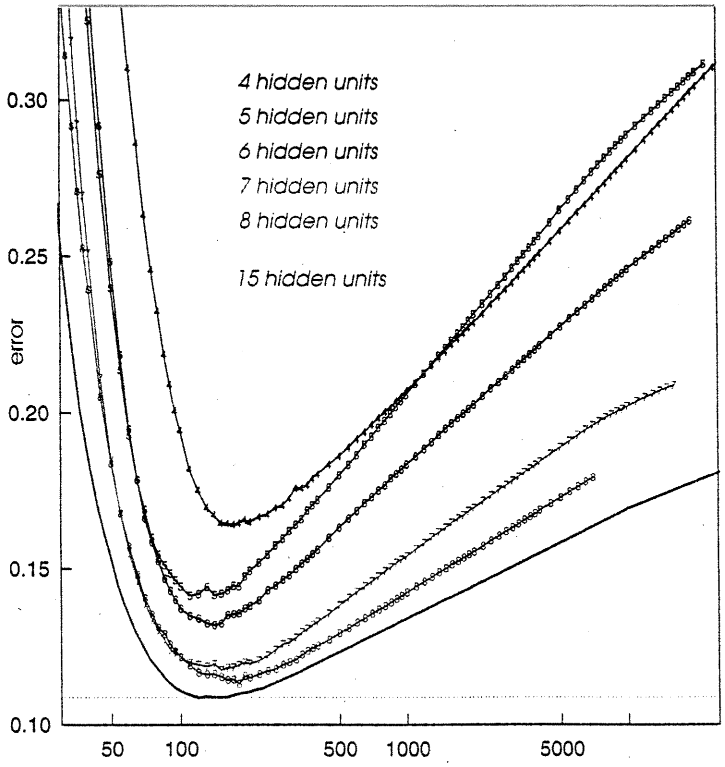

Figure 2. Out-of-sample errors for networks with different numbers of hidden units, plotted as a function of training time in epochs.

Figure 2 shows the test errors for networks with 4 to 15 hidden units. We also tried much larger networks (with up to 100 hidden units); their learning curves did not differ significantly from those of the network with 15 hidden units. Note that we did not average over different runs for each network size because averaging can wash out individually fairly sharp minima, and thus suggest erroneously the absence of overfitting.

What is not important in the figure is the number of epochs where the individual minima occur; the location of the minimum can always be moved by scaling the learning rate as a function of the number of hidden units. Two features, however, are important:

Result 2: **Small networks also overfit.**

We here consider the *shape* of the individual curves.

Perhaps surprisingly, even very small networks overfit. Rather than capturing (with their limited resources) as many properties as they can which do generalize, even very small networks begin to extract properties of the training set that do not generalize to the test set. This fact implies that heuristics such as "pick a network 'just large enough' and train it 'to convergence'" can be inappropriate. Another point that follows from the figure is that the existence of overfitting does not imply that the network size is sufficient for a given task, let alone too large.

We now turn to the value of the minimum of each curve in Figure 2.

Result 3: **There is no such thing as too large a network.**

The best performance on the test set is obtained by large networks, provided they are stopped early (at the minimum of the test set).[3] The out-of-sample error for the smallest network is significantly worse (by about 50%) than the error for the largest network.

This fact (that the largest networks are best) has a number of possible explanations. The first explanation focuses on the backpropagation search process (gradient descent): the higher dimensional the space, the easier it is to find a good solution. The

[3] To be methodologically clean, the final error should be reported on a *new* test set, and the present test set should be considered as a cross-validation set (see Weigend, Huberman, and Rumelhart, 1990). In the experiment reported here, no more data were available. As a simple test we divided the 540 test data points into two sets at random. The performance on both of them was comparable.

second explanation interprets a large network as a superposition of smaller networks. Let us draw the analogy to measuring a quantity (any quantity) T times: if the measurements are uncorrelated, the error on the true value of that quantity measured drops as $1/\sqrt{T}$. If all measurements are identical, there is neither gain nor harm. Similarly, the large network can be viewed as averaging several smaller networks, thus leading to possibly better (and in the worst case, equal) performance than a single, small network.[4] The third explanation views early stopping as a kind of regularizer; for linear networks (no hidden units), the regularizer corresponds to standard weight-decay (or ridge regression); see Guyon et al. (1992), and Moody (1992).

Summarizing Result 3, the heuristic of using large networks and stopping early can be useful. In our experience, this result is not limited to networks for classification but also holds for networks for regression.[5]

PART 2: **THE EFFECTIVE NUMBER OF HIDDEN UNITS**

In connectionist modeling, it is important to distinguish between potential network size and effective size. Whereas the potential size is fixed by the architecture, the effective size changes (usually increases) with training time. In Part 2 we present a measure that describes the effective number of hidden units of a network. Whereas Part 1 focused on the test errors, Part 2 focuses on the activation values of the hidden units and answers questions such as: How do the responses of hidden units change as a function of training time? Do all hidden units "get busy" at the same time in training, or one after the other? How correlated are their responses? And when they are all busy, do they do different things or not?

We first consider the hidden units individually (i) by plotting histograms of the activation values for each hidden unit (at certain points in the training process), and (ii) by computing the variances of the individual responses. We then analyze the ensemble of the hidden units by computing the eigenvalues of the covariance matrix of the activations, and by plotting them (again) against training time.

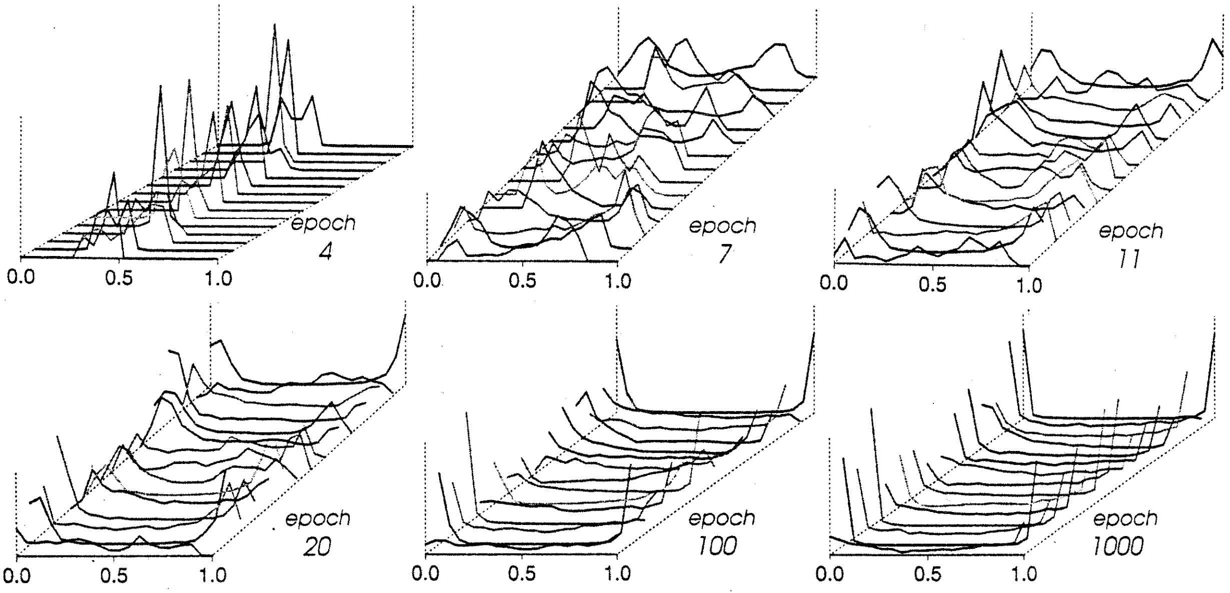

Figure 3. Histograms of hidden unit activations after 4, 7, 11, 20, 100, and 1000 epochs. In each of the six plots, we show the histograms of each of the 15 hidden units in different levels of gray.

[4] A more theoretical exposition can be found in the paper by Perrone (this volume). An alternative (Bayesian) perspective is offered by Buntine and Weigend (1991): rather than picking the maximum likelihood value, the various sub-nets can be interpreted as sampling the solution space, thus shifting the prediction towards a *maximum a posteriori* solution.

[5] The *Santa Fe Time Series Prediction and Analysis Competition* (Weigend and Gershenfeld, 1993) can be viewed as a comparison of different methods for regression. The best predictions for Data Set A (1,000 points) were obtained by a feed-forward network with 1,105 parameters (Wan, 1993)—more parameters than available data points! The training of the network was stopped early, and the stopping point was guided by the performance on a cross-validation set (a part of the available competition data that Wan had set aside). One example where we found the early stopping heuristic to be insufficient is the prediction of financial data: an explicit regularizer (such as *weight-elimination)* seems necessary to obtain forecasts better than chance (Weigend, Huberman and Rumelhart, 1992).

Focus 1: ***Individual*** **hidden units: histograms and variances.**

As a first attempt to describe the "busy-ness" of each hidden unit, we chop the (0,1) range into (evenly sized) bins and count how many input patterns produce a hidden unit activation that falls into that bin. (Such a plot is called a histogram.) For each epoch, we combine the 15 histograms of the 15 hidden units into one three-dimensional plot. (The order of the hidden units in the depth dimension is arbitrary.)

Early in training (after 4 epochs), all hidden units have activations around the central value of 0.5: the small weights and biases only allow for a small net-input, yielding activations around 0.5. As training continues, the range covered by the hidden unit activations broadens, and eventually the units show almost binary behavior; i.e., the activations tend to cluster around the extreme values of zero and one.

Such histograms only portray a snapshot in time (the state at a certain epoch). In order to visualize the changes in learning, it is desirable to characterize these 15 curves (in each plot) by one number each, such as the *variance* of hidden unit *i*,

$$VAR_i = 1/(N\text{-}1) \sum_p (x_i(p) - \langle x_i \rangle)^2 \,,$$

where the sum extends over the *N* patterns (indexed by*(p)*). The angular brackets <> denote the mean over patterns,

$$\langle x_i \rangle = 1/N \sum_p x_i(p) \,.$$

The variance measures the amount of busy-ness: on the one hand, if the activations just hover around the mean, the variance is small. On the other hand, if the activation is zero for half of the patterns and one for the other half, the maximum of 0.25 for the variance is reached. In order to be able to express the busy-ness in the same mathematical units as the activations, it is convenient to use the square root of the variance, or *standard deviation.* The maximum value the standard deviation can take for signals limited to [0,1] is 0.5 (= √0.25). In Figure 4, we plot the standard deviation of each hidden unit against the epoch number.

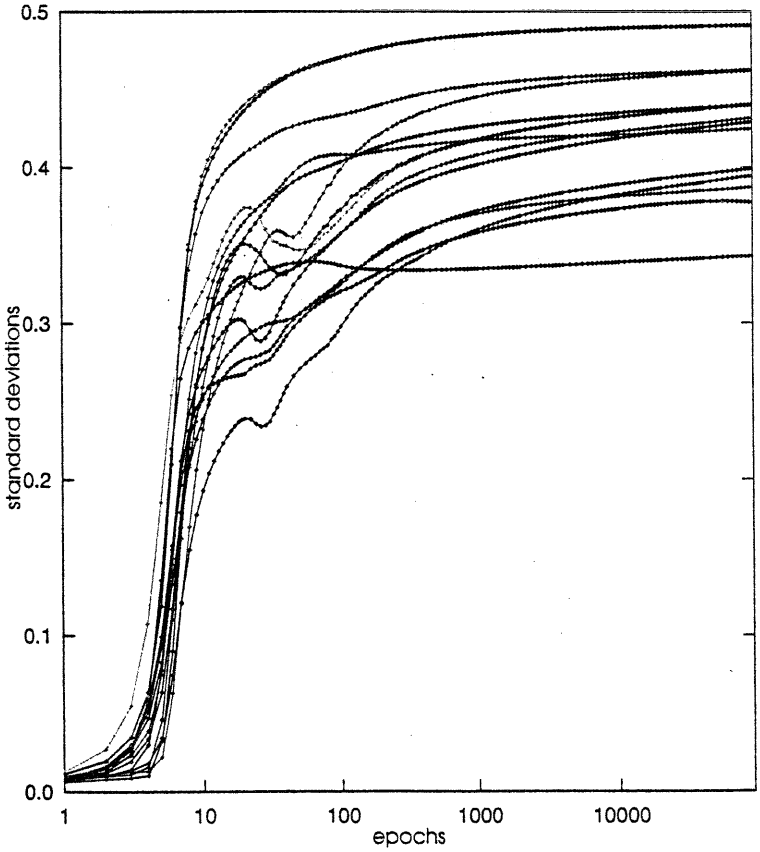

Figure 4. Standard deviations (square roots of the variances) of each individual hidden unit plotted against training time (in epochs on a logarithmic scale). Each curve corresponds to a hidden unit.

Figure 3 and Figure 4 both allow us to answer one of the questions: all hidden units "wake up" at the same time. To approach the next question (whether the units just duplicate each other or whether they do different things), we need to analyze them as an ensemble, rather than individually.

Focus 2: ***Ensemble*** **of hidden units: eigenvalues of covariance matrix (principal components).**

We begin this section with a geometrical interpretation of the hidden unit responses: each response can be viewed as a point in h-dimensional space. The 540 input patterns of the test set (presented as input, one after another) thus generate a set of 540 points in that space.[6] Potentially, these points could fill the entire h-dimensional space. However, let us consider the case where all the points lie on a one-dimensional straight line (in the h-dimensional space). This would allow us to re-express the network as a network with only one hidden unit. Similarly, if the points were on a two-dimensional (hyper-) plane, the effective number of hidden units would be two.[7]

We are interested in the dimensionality of the point cloud, but we do not care about its orientation in space. Ideally, we would like to rotate the original coordinates (corresponding to the hidden units in the network) to a new set of coordinates such that the first direction captures as much spread of the point cloud as possible, the second, while orthogonal to the first, as much of the remaining spread as possible, etc.

The mathematical approach consists of two steps. First, the correlational structure of the point cloud has to be captured. Considering only two-point interactions,[8] we use the *covariance* between hidden units i and j, given by

$$COV_{ij} = 1/(N-1) \sum_p (x_i^{(p)} - \langle x_i \rangle)\,(x_j^{(p)} - \langle x_j \rangle)\ .$$

The sum extends over the patterns. Angular brackets <> denote the average over patterns. COV is by construction a symmetric matrix. Its main diagonal contains the h variances VAR_i introduced above. In addition, there are (at most) $h(h-1)/2$ different off-diagonal elements.[9]

Second, we have to find the effective dimension of the point cloud—it is given by the number of significantly non-zero *eigenvalues* of the covariance matrix. They describe the spreads along the new axes of the point cloud.[10] The square roots of the eigenvalues are called *principal components.*

In summary, the mathematical recipe to find the effective dimension consists of two steps:

1. Compute how responses of pairs of hidden units co-vary with each other (as patterns are presented at the input to the network). These numbers are combined in the covariance matrix.
2. Compute the eigenvalues of the covariance matrix. The number of significantly non-zero eigenvalues serves as *effective number of hidden units.*

The eigenvalues are computed after each epoch of learning. Numerically stable ways (such as singular value decomposition, implemented in most numerical and statistical packages) are available. In Figure 5, we plot the square roots of all the eigenvalues of the hidden unit ensemble as a function of training time.

[6] The distributions of the points of the test and the training set are very similar–differences between test and training set are on the 1% level of the hidden unit activations. Note that the effect of overfitting discussed in Part 1 lives off this small difference.

[7] We here consider only linear subspaces. The points might lie on a curved manifold of smaller dimension than the embedding space. If we were interested to find the number of degrees of freedom for a manifold of any shape, we could use the information theoretic measure of *redundancy*; see Gershenfeld and Weigend (1993). The generality of this measure, however, has a price: in order to estimate it reliably, several more orders of magnitude of data are needed than we have available in the present example. We thus restrict ourselves to the *linear effective* dimension of the hidden unit space. If the processing that follows the hidden units is strictly linear, the linear effective dimension is sufficiently general.

[8] By using the covariance matrix, we only explore two-point interactions (or second-order correlations). This is equivalent to assuming that the points are Gaussian distributed, since higher than second-order moments vanish for a Gaussian.

[9] When the covariance is normalized to lie between -1 and 1 (by dividing it by both of the standard deviations of the two individual hidden units), the (linear) correlation coefficient between hidden unit i and j is obtained.

[10] The eigen*vectors* of a matrix (or a linear operator) point in the directions that do not get rotated, but only contracted (if the absolute value of the corresponding eigenvalue is less than unity), or expanded (if it is larger than unity). The directions (also called principal directions) are not important to us, but the stretch of the point cloud along each axis is: its extension is given by the eigen*value* corresponding to the eigendirection.

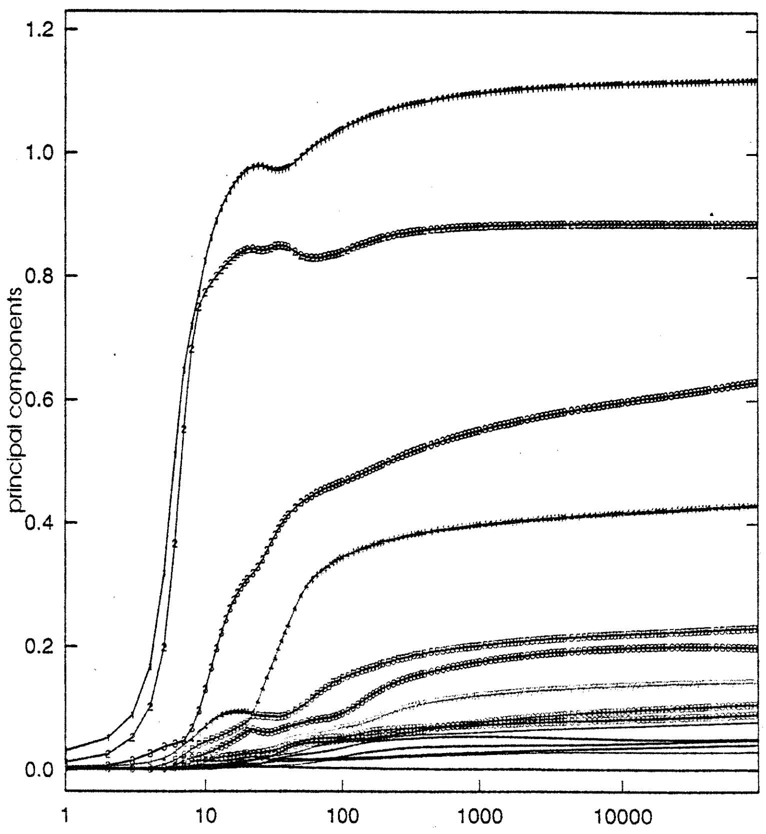

Figure 5. Evolution of principal components (square roots of the eigenvalues of the covariance matrix).

The network extracts the principal components sequentially. After 10 epochs, for example, only two eigenvalues have emerged; the remaining 13 are still dormant. Having two significant eigenvalues means that the hidden unit activation vectors (the responses to the input patterns) lie on a two-dimensional hyperplane in the 15-dimensional space. At this point in training, a third dimension is just slowly emerging. Compare this to Figure 4 where, also at epoch 10, all hidden units have already reached at least half of their final standard deviation.

Is this behavior surprising? Gradient descent with a simple cost function (such as cross-entropy or squared error) does not provide any incentive in for the hidden units to form an orthogonal representation. It is a "greedy algorithm:" the sole goal of the weight updates is to reduce the overall error. Thus, all hidden units go after the same features and only differentiate as much (or as little) as necessary to reduce the overall error. In other words, the hidden units form a distributed representation.

The information contained in Figures 4 and 5 is complementary. For example, an effective hidden unit dimension of one could be obtained either by a single active units and the rest sleeping, or by all of them busily doing the same. We know from Figure 4 that the latter is the case in backpropagation learning.

Summarizing Part 2, we saw that, on the one hand, all hidden units "wake up" at essentially the same time, as exhibited both by the histograms of their activation values (Figure 3) and by their individual variances (Figure 4). On the other hand, the units start off by doing the same thing and only later begin to differentiate: the effective number of hidden units (the number of significantly non-zero eigenvalues of the covariance matrix) grows slowly with training time (Figure 5). The effective number of hidden units is often very small compared to the potential number of hidden units given by the architecture of the network.

Acknowledgments

We thank Dave Rumelhart and the Stanford PDP group for all the discussions. We thank Jeff Elman for the data. We acknowledge the feedback of friends and colleagues when the figures of this paper were presented: at FAW (Ulm, Summer 1991; in particular Steve Hanson's comments), CLNL (Berkeley, Summer 1991), IJCNN (Singapore, Fall 1991), Neural Networks for Computing (Snowbird, Spring 1992), and at CMSS (Boulder, Summer 1993). We also thank Brian Bonnlander for his comments on this first written version. The analysis method of Part 2 was first presented in the context of time series prediction at INTERFACE (Seattle, Spring 1991; see Weigend and Rumelhart, 1991).

References

Buntine, W. L., and A. S. Weigend (1991) "Bayesian Backpropagation." *Complex Systems* **5**: 603–643.

Elman, J. L., and D. Zipser (1988) "Discovering the hidden structure of speech." *J. Acoust. Soc. Am.* **83**: 1615-1626.

Gershenfeld, N. A., and A. S. Weigend (1993) "The Future of Time Series." In *Time Series Prediction: Forecasting the Future and Understanding the Past,* edited by A. S. Weigend, and N. A. Gershenfeld, pp. 1-70. Addison-Wesley.

Guyon, I., V. Vapnik, B. Boser, L. Bottou, and S. A. Solla (1992) "Structural Risk Minimization for Character Recognition." In *Advances in Neural Information Processing Systems 4*, edited by J. E. Moody, S. J. Hanson, and R. P. Lippmann, pp. 471–479. Morgan Kaufmann.

Moody, J. (1992) "The *Effective* Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Systems." In *Advances in Neural Information Processing Systems 4*, edited by J. E. Moody, S. J. Hanson, and R. P. Lippmann, pp. 847–854. Morgan Kaufmann.

Perrone, M. P. (1994) "General Averaging Results for Convex Optimization." In *Proceedings of the 1993 Connectionist Models Summer School* , edited by M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend, pp. 364–371. Lawrence Erlbaum.

Rumelhart, D. E., R. Durbin, R. Golden, and Y. Chauvin (still in press) "Backpropagation: The Basic Theory." In *Backpropagation: Theory, Architectures and Applications*, edited by Y. Chauvin and D. E. Rumelhart. Lawrence Erlbaum.

Wan, E. A. (1993) "Time Series Prediction Using a Connectionist Network with Internal Delay Lines." In *Time Series Prediction: Forecasting the Future and Understanding the Past,* edited by A. S. Weigend, and N. A. Gershenfeld, pp. 195-217. Addison-Wesley.

Weigend, A. S., B. A. Huberman, and D. E. Rumelhart (1990) "Predicting the Future: A Connectionist Approach." *International Journal of Neural Systems* **1**: 193–209.

Weigend, A. S., and D. E. Rumelhart (1991) "Generalization through Minimal Networks with Application to Forecasting." In *INTERFACE '91–23rd Symposium on the Interface: Computing Science and Statistics*, edited by E. M. Keramidas, pp. 362–370. Interface Foundation of North America.

Weigend, A. S., B. A. Huberman, and D. E. Rumelhart (1992) "Predicting Sunspots and Exchange Rates with Connectionist Networks." In *Nonlinear Modeling and Forecasting*, edited by M. Casdagli, and S. Eubank, pp. 395–432. Addison-Wesley.

Weigend, A. S., and N. A. Gershenfeld, editors (1993) *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley.