

Hebbian Feature Discovery Improves Classifier Efficiency

Todd Leen †, Mike Rudnick †, and Dan Hammerstrom * †

† Department of Computer Science and Engineering
Oregon Graduate Institute
19600 NW Von Neumann Dr.
Beaverton, OR 97006-1999

* Adaptive Systems, Inc.
1400 NW Compton Dr., Suite 340
Beaverton, OR 97007

Abstract

Neural network implementations of principal component analysis provide a means for unsupervised feature discovery and dimension reduction. This suggests a modular approach to recognition tasks - feature discovery and extraction, followed by classification. We apply this technique to neural network speech recognition. The approach offers a significant reduction in total training time without degrading the classifier performance.

1. Introduction

The application of neural networks to problems in pattern recognition is often carried out with primitive features - e.g. image pixels for problems in machine vision, or a spectrogram for problems in speech recognition. There are fundamental problems with this approach.

Classifier networks trained with raw data do not scale well. The feature space corresponding to the raw data is of such large dimension that memory required is prohibitively large, learning time is prohibitively slow [Hin87], hardware implementation is prohibitively expensive [Ham86], and the required number of training examples is prohibitively large [BaD89]. These problems constitute a scaling catastrophe which hinders the economical solution of large problems.

The scaling problems can be significantly alleviated by encoding the input data in terms of a low dimensional set of features. This compression reduces the dimension of the input space, which allows for classifier networks with fewer weights and shorter training times.

We employ two neural network implementations of principal component analysis (PCA) to reduce the dimension of speech signals. The compressed signals are then used to train a feed forward classification network for vowel recognition. We compare classification performance, network size, and training time for networks trained with both compressed and uncompressed data.

Our results show that a significant reduction in training time, five-fold in our case, can be achieved without a sacrifice in classifier accuracy. This reduction includes the time required to train the compression network. Thus dimension reduction, as performed by unsupervised neural networks, is a viable tool for enhancing the efficiency of neural classifiers.

2. Hebbian Learning and Principal Component Analysis

In a classic paper, Oja [Oja82] shows the relation between Hebbian learning and PCA. Oja treats a single linear neuron with weights that develop according to a Hebbian learning rule. Oja shows that

This work was supported by the Office of Naval Research under contract no. N00014-88-K-0329.

the weight vector converges to the eigenvector of the input correlation correspond to the largest eigenvalue, thus maximizing the output variance.

Sanger [San89] has developed an extension of Oja's algorithm, which uses a projection-like process to extract as many principal components as desired. The network consists of a single layer of weights connecting the N input channels to M ($\leq N$) output nodes. At the presentation of each input pattern, the weight matrix is updated according to

$$\begin{aligned} \delta \omega &= \gamma [\mathbf{y} \mathbf{x}^T - \text{Lower} (\mathbf{y} \mathbf{y}^T) \omega] \\ &= \gamma [\omega \mathbf{x} \mathbf{x}^T - \text{Lower} (\omega \mathbf{x} \mathbf{x}^T \omega^T) \omega] \end{aligned} \quad (1)$$

where \mathbf{x} is the input vector, ω is the $M \times N$ matrix (with rows equal to the weight vectors), \mathbf{y} is the vector of activity on the output layer, and γ is the learning rate. The operator *Lower* sets the upper-triangular part of its argument equal to zero.

The diagonal terms of $\mathbf{y} \mathbf{y}^T$ ensure that the weight vectors converge to unit magnitude. The off-diagonal terms ($\mathbf{y}_i \mathbf{y}_j$), $j < i$, effectively modify the input to the i^{th} neuron by subtracting the components ($\omega_j \cdot \mathbf{x}$) ω_j along the j^{th} weight vector. For a local implementation, this projection-like procedure requires feedback links to the input layer and additional nodes to carry out the subtraction. There is a sequential aspect to the training, since the input to node i depends upon the activity of the previous nodes.

In the original formulation all of the weight vectors are trained on each iteration. However, until the first weight vector is close to convergence, the learning in the second and succeeding nodes is meaningless. In order to minimize wasted computation we modify Sanger's algorithm by adding a learning enable window. Initially, only the first two or three nodes in the string are in the learning window. As the first few nodes begin to converge, the leading edge of the learning window is advanced to the next node in sequence. The trailing edge is advanced when the node in that position has converged.

The second algorithm employed was developed by Oja and Karhunen [OjK85]. The algorithm produces a set of orthonormal weight vectors which span the same subspace as the first M eigenvectors of the input's auto-correlation. In matrix form, the weight update is given by

$$\delta \omega = \gamma [\omega \mathbf{x} \mathbf{x}^T - (\omega \mathbf{x} \mathbf{x}^T \omega^T) \omega] \quad (2)$$

which is, apart from the *Lower* operation, identical to (1). It can be shown that the ensemble-averaged, continuous-time form of (2) has stable equilibria when the rows of ω are orthonormal, and span the space spanned by the first M eigenvectors of the input's correlation; however, there is no global convergence proof available. This algorithm is equivalent to a two-layer linear network $\omega : \mathbf{x} \rightarrow \mathbf{y}$, $\omega^T : \mathbf{y} \rightarrow \mathbf{z}$ trained with $\delta \omega = \gamma [\mathbf{y} (\mathbf{x} - \mathbf{z})^T]$.

3. Vowel Recognition

Our recognition task is the classification of vowels taken from continuous speech. Data are drawn from the phonetically labeled DARPA-TIMIT acoustic phonetic data base [FiD86]. The 12 vowel classes used are: /iy/ (beat), /ih/ (bit), /eh/ (bet), /ae/ (bat), /ix/ (roses), /ax/ (the), /ah/ (butt), /uw/ (boot), /uh/ (book), /ao/ (bought), /aa/ (cot), /er/ (bird).

Each data vector is obtained by extracting a 10 ms. slice from the center of each vowel. From the discrete Fourier transform (DFT) we obtain 64 spectral coefficients covering the range from 0 to 4 kHz. The training data consists of 342 examples of each vowel, selected from utterances of 320 speakers, for a total of 4104 training vectors. The test data consists of 137 examples of each vowel, selected from 100 speakers, for a total of 1644 test vectors.

4. Architecture and Simulations

The experimental system is a heterogeneous network consisting of two subnetworks, an input compression net followed by a classification net. In all cases the classification net consists of a two layer

feed forward network, one hidden layer and one output layer, trained using conjugate gradient descent [BaC89]. We tested various configurations for the compression network with from 5 to 30 output nodes. For comparison the same classification task is performed feeding the full complement of 64 DFT coefficients directly into the classification net.

The heterogeneous network is trained sequentially. First, the compression network is trained using the 64 spectral coefficient vowel data. Second, both the training and testing datasets are passed through the trained compression net yielding a transformed version of each. The transformed training data is then used to train the classifier network. Finally, the transformed test data is used to measure the performance of the classifier net.

All simulations are run on an eight node Sequent Symmetry S81 with revision B cpu upgrade under DYNIX V3.0.12. At each stage of the training process, the training time in Symmetry cpu seconds is recorded.

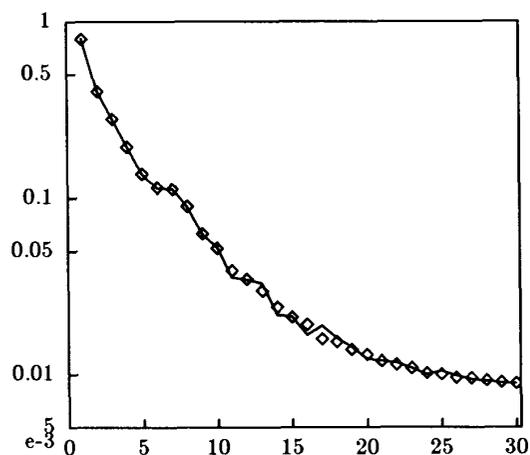


Figure 1: Covariance Eigenvalue Spectrum
diamond = algebraic, line = Sanger

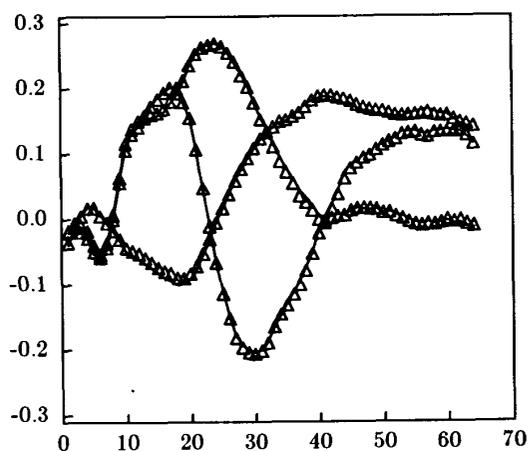


Figure 2: Covariance Eigenvectors
triangle = algebraic, line = Sanger

5. Results

Our results address two questions. Are unsupervised networks able to perform significant dimension reduction on speech data? How much reduction is possible without adversely affecting classifier performance?

5.1. Compression Results

Figure 1 compares the largest thirty eigenvalues obtained from Sanger's algorithm with those obtained by direct algebraic diagonalization of the correlation matrix (Householder reduction, followed by QL decomposition). Both eigenvalue spectra are nearly identical, indicating that the network converged properly. This is the first reported use of a network to extract such a large number of eigenvectors.

The form of this eigenvalue spectrum indicates that one can substantially reduce the dimension of the representation. Since the eigenvalues are all below 0.012 beyond the 20th component, retaining the higher components will not significantly improve the representation accuracy. Our classification results corroborate these expectations.

Figure 2 is a plot of the 64 components of each of the first three eigenvectors of the covariance matrix. Both the network approximation (solid line) and the true eigenvectors (triangles) are plotted. As with the eigenvalues, the eigenvectors are well approximated by the network.

Number of Hidden Nodes	Number of Input Nodes						
	5	10	15	20	25	30	64
8	34.0	45.2	44.8	44.6	45.4	35.3	43.0
16	35.6	46.4	46.8	46.8	48.2	47.7	47.2
32	34.6	47.5	43.3	45.9	40.0	48.1	45.9

Table : Percent Correct Vowel Classification

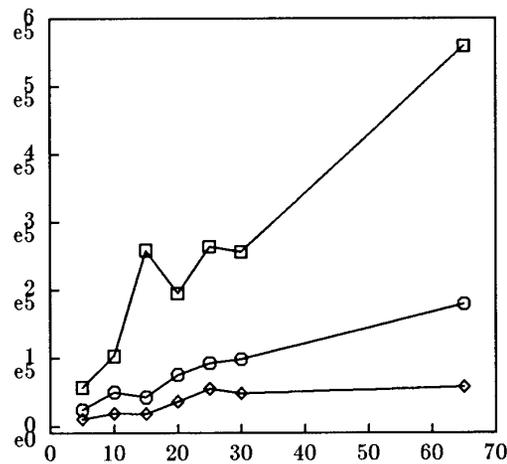


Figure 3: Total Training Time

diamond = 8 hidden nodes
circle = 16 hidden nodes
square = 32 hidden nodes

5.2. Classification Results

The heterogeneous networks generally perform well. Table 1 shows the performance of each of the classifier networks on the test data. Each row gives the percent correct vowel classification for a network with a single hidden layer of 8, 16, or 32 nodes. Columns indicate the number of input nodes feeding the classifier network. For the first six columns, the indicated number of principal components are fed to the classifier net, while for the last column, the full complement of 64 DFT coefficients are fed directly into the classifier. The entries for the 20 and 64 input columns represent averages of 10

training runs each; all other number represent single runs. Although classification results of 34-48 % may seem poor, this is a difficult task - humans perform at only 50% [MCS90].

We note that a mere 10 principal components gives classification performance comparable to that of any other configuration, including using all 64 spectral coefficients. Thus, a six-fold reduction in the dimension of the input to the classifier is achieved without loss in classifier accuracy.

A primary effect of this dramatic dimension reduction is a sharp drop in the time required to train the classifier. Since there are fewer input nodes, there are fewer weights; hence training is faster. As the hidden layer in the classifier becomes larger, the speedup becomes more significant. This behavior is shown in Figure 3. The ordinate gives the number of cpu seconds used to train both the compression and classifier networks. The abscissa gives the number of inputs to the classifier. The three curves (from bottom to top) correspond to 8, 16, and 32 classifier hidden nodes. The points at the far right represent the number of cpu seconds to train the classifier net using all 64 spectral coefficients (no compression network). As before, the data points for 20 and 64 inputs are the averages over 10 runs, while the other points represent single runs. Each curve shows a roughly linear increase in training time with dimension of the input space, mirroring the increase in the number of weights in the classifier. We note that the time required to train the compression net is generally much smaller than the time required to train the classifier net. For example, the training time for the compression net with 20 output nodes is 42%, 17% and 6% of the training time for the corresponding classifiers with 8, 16, and 32 hidden nodes respectively.

Number of Hidden Nodes	Compression Algorithm		
	Covariance	Autocorrelation	Oja & Karhunen
8	44.6	45.3	44.34
16	46.8	47.5	47.87
32	45.9	48.1	47.81

**Table 2: Performance of Compression Algorithms
20 Inputs**

Table 2 compares the performance of classifiers trained on data preprocessed by several different compression nets. The first two columns show the performance of classifiers trained on data compressed by Sanger's algorithm. For the column labeled 'covariance', the mean has been subtracted from the input data prior to training. In this case, the weights in the compression network converge to eigenvectors of the input's covariance matrix. For the column labeled 'auto-correlation', the mean was not subtracted. In the last column, we give the performance of a classifier trained using data compressed according to Oja and Karhunen's algorithm (equation 2). Note that the figures in all columns are similar.

We have achieved similar dimension reduction using data from multiple spectral slices of vowels drawn from seven classes. The data base for those experiments was constructed for the development of a spoken-letter recognition system [CFM90]. The raw data consisted of 192 spectral coefficients plus pitch, duration, and amplitude features. The compressed data consisted of 40 principal components plus pitch, duration, and amplitude features. The classifier trained on the full complement of 195 inputs correctly identified 97.05% of the vowels in the test set, while the classifier trained on 43 inputs correctly identified 97.96% of the vowels in the test set.

6. Discussion

Our results show that unsupervised dimension reduction, as carried out by neural network algorithms, is a viable technique for alleviating the scaling problems inherent in neural network classifiers. The effort devoted to computing the reduced representation is more than repaid by the acceleration in

the classifier training. Smaller dimensional representations also reduce memory and storage requirements, and hardware implementation costs.

Neural implementations of principal component analysis allow the reduced features to be calculated in parallel. In addition, this data preprocessing can be naturally incorporated into the same technology that serves the classification task. This will be particularly important as neural computers become available. In addition, the neural algorithms can presumably respond to changes in the statistics of the input ensemble in real-time. Verifying this conjecture is left to future research.

These techniques can be extended in several ways. The extension from static to spatio-temporal features is under study. Secondly, in the implementations discussed here, each cell's receptive field covers the entire input space. Limited receptive fields will lead to greater modularization and will further reduce implementation costs.

Acknowledgments

The authors are grateful to Prof. Ronald Cole and PhD candidate Yeshwant K. Muthusamy for access to their speech database, and for continued lively discussion and interest in this work.

References

- [BaC89] E. Barnard and R. Cole (1989). "A neural-net training program based on conjugate-gradient optimization," Technical Report No. CSE 89-014 , Department of Computer Science and Engineering, Oregon Graduate Institute.
- [BaD89] E. B. Baum and H. David (1989). "What size net gives valid generalization?," *Neural Computation*, vol. 1, pp. 151-160.
- [CFM90] R. Cole, M. Fanty, Y. Muthusamy and M. Gopalakrishnan (1990). "Speaker-Independent REcognition of Spoken English Letters", Submitted to June 1990 IJCNN, San Diego.
- [FiD86] W. M. Fisher and G. R. Doddington (February 19-20, 1986). "The DARPA Speech Recognition Research Database: Specification and Status," *Proceedings of the DARPA Speech Recognition Workshop* , Palo Alto, CA, pp. 93-99.
- [Ham86] D. Hammerstrom (August 1986). "A Connectivity Analysis of Recursive, Auto-Associative Connection Networks," Tech. Report CS/E-86-009 , Dept. of Computer Science/Engineering, Oregon Graduate Center, Beaverton, Oregon.
- [Hin87] G. Hinton (1987). "Connectionist learning procedures," *CMU Technical Report - CMU-CS-87-115* .
- [MCS90] Y. K. Muthusamy, R. A. Cole and M. Slaney (1990). "Speaker-Independent Vowel Recognition: Spectrograms Versus Cochleagrams," *Proceedings of the IEEE 1990 International Conference on Acoustics, Speech and Signal Processing* , (held at Albuquerque, New Mexico, April 1990).
- [Oja82] E. Oja (1982). "A simplified neuron model as a principal component analyzer," *J. Math. Biology*, vol. 15, pp. 267-273.
- [OjK85] E. Oja and J. Karhunen (1985). "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. of Math. Anal. and Appl.*, vol. 106, pp. 69-84.
- [San89] T. Sanger (1989). "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network," *Neural Networks*, vol. 2, no. 6, pp. 459-473.