Stochastic Computation

John A. Cortese

Department of Electrical Engineering
California Institute of Technology
Pasadena, California

1995

Stochastic Computation

Thesis By
John A. Cortese

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1995

(Defended May 15, 1995)

Acknowledgements

Many people contributed to the research effort which led to this thesis. Fellow Ph.D. students Bhusan Gupta and Jeff Dickson provided considerable moral and technical support. Marvin Simon throughout has been a mentor, as well as a friend. My advisor, Rodney Goodman, made it all possible, by first admitting me, and then supplying a superbly equipped lab in which I could pursue my research.

The support of my family has been critical at every step. My brother David and sister Susan kept up a steady flow of cards and letters filled with words of encouragement and humor. And most important of all, my mother's phone calls and spontaneous boxes of gifts brightened many a day.

To these people, and countless others, I can only say thank you. I would never have gotten this far without you.

Stochastic Computation

John A. Cortese

jcort@micro.caltech.edu

Department Of Electrical Engineering

California Institute Of Technology

Pasadena, CA. 91125

June 4, 1995

Abstract

This thesis approaches computation from a communication theory perspective. Data is given to a computer, which is asked to arrive at a binary hypothesis decision. The computation task is viewed as a signal drawn from an ensemble, corrupted by noise, and passed to a receiver which is asked to make a binary signal detection decision.

To illustrate the approach, learning in a neural network is studied. An algorithm based on statistical communication techniques is developed which allows the determination of the neural network size, architecture, and system parameters. The computation, as interpreted in the communication framework, is assigned an equivalent channel capacity which measures the effectiveness with which the computing system extracts information in the Shannon sense from the input data. Numerical simulations of a neural network recognizing handwritten digits are used to illustrate key points.

CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF TABLES

Stochastic Computing

John A. Cortese

Department Of Electrical Engineering

California Institute Of Technology

Pasadena, CA. 91125

jcort@micro.caltech.edu

June 4, 1995

1. Introduction

This thesis investigates stochastic computation. The theme of the research is that a large number of simple, unreliable computing elements operating in parallel can together achieve very reliable, complex computation. Inspiration for this approach comes from biological cortex, where a huge number of primitive, unreliable cells, together implement incredibly complex data processing functions. The following example illustrates the philosophy of stochastic computing. Imagine you have 15001 Forrest Gump individuals arrayed against a single Albert Einstein. Albert answers single questions correctly with an accuracy of 99.0 percent. Each Gump answers single questions correctly with an accuracy of 51 percent. Imagine each Gump's answer is independent of his fellow Gumps. If you asked all 15001 Gumps the same question, and tallied their answers, who as a group would be more accurate : The single Albert or the Gump

family ? The answer lies in the statistic below. The probability the Gump family is correct is seen below.

$$\sum_{i=7501}^{15001} \left\{ \begin{matrix} 15001 \\ i \end{matrix} \right\} \quad 0.51^{i} \quad 0.49^{15001-i} \quad \equiv \quad 0.993$$

Equation 1. The Gump's Versus Albert Einstein

This is the probability that more than half of the Gump family (7501→15001) chooses the correct answer. Thus you would be correct more often listening to the Gump family than Albert ! The primary point of this example is the ability of a collective system to efficiently compute using simple, individually inaccurate elements through the use of a relatively straightforward interaction algorithm : tally the individual Gump votes, and choose the majority perspective. Note the redundancy of the system. Large sections can fail entirely (Gumps die), yet overall system performance degrades gracefully, not catastrophically.

With a statistical outlook, we shall apply concepts from communication theory to describe stochastic computation. For instance, consider a channel for a communication system. A single bit enters the channel, and a single bit exits the channel. The channel input and output are typically described by a random variable. Assuming for simplicity channel symmetry, the crossover or decode probability of error is ε. However, this model also can represent computing. If the input represents the occurrence of an event α, then the output can represent a guess or opinion of the occurrence of an event β. Specifically, consider a binary symmetric channel model such as below.

Stochastic Computation

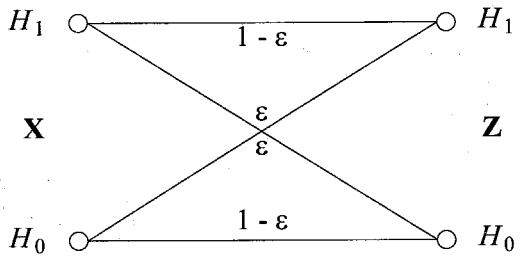


Figure 1. Binary Symmetric Channel Model

This model also represents a binary hypothesis computer. An input event α occurs. The output represents the occurrence of event β. We shall show that many common communication channel models can be used to represent stochastic computing operations, merely by reinterpreting the meaning of the input and output channel random variables. The resulting generalized communication channel models can be systematically cascaded in parallel and serial to achieve a generalized channel model representing an arbitrary hypothesis computation.

2. Neural Computation

As a specific illustration of the principles of stochastic computing, neural computation is considered. Neural systems typically embody the philosophy advocated here. Unsophisticated computing elements, with an interconnection algorithm or architecture, effectively solve complex problems. This thesis introduces a technique for designing and training general neural networks. The original paradigm treated is designing a neural network to solve a binary hypothesis classification problem.

The technique to be introduced allows one to calculate the weights, thresholds, and non-linear sigmoid functions for a multi-layer neural network which performs binary hypothesis testing. Learning of parameters is accomplished using a learning algorithm operating on a labeled (two-class) training set. Numerical results from the various handwritten digit databases are presented to illustrate key points with a real life scenario.

The lessons learned for the binary construction case are then extended to multiple hypothesis classification. Last, the behavior of recurrent networks and associative memories is studied. The tools used throughout are those of classical statistics. Analogies to communication system implementations, such as for signal detection and classification, are made to help visualize the concepts presented. These analogies also serve to provide a link to probability theory via statistical communication theory.

3. General Neural Network Architecture

In the work to follow, neural inputs x_i are considered to be random variables, and not deterministic numbers. The determination of individual neuron parameters is with an aim to optimize the networks performance across the ensemble of all problems which will be presented to the system. The action of any one neuron will be to map an incoming probability density function into a form which can be used to make an efficient statistically optimum decision, of which more will be said later. Here, a neuron is viewed as a deterministic function operating on a random input. This model is opposite to several other neural network learning algorithms. For example, in a Boltzmann machine, a neurons output is a probabilistic function of its

deterministic inputs. Another probabilistic mapping of deterministic inputs is the update rule typically used in simulated annealing approaches, whereby a temperature controls the degree of randomness introduced in the neuron mapping. The approach here can be considered as a dual to the Boltzmann/Simulated Annealing approaches. Learning methods which utilize both probabilistic inputs *and* probabilistic update/decision rules will be discussed briefly at the end of this thesis.

3.1 A Single Neuron

The simplest neural network architecture considered is feedforward, with real valued weights and inputs. The neuron implements a threshold shifted inner product of an input vector with a weight vector. This operation is shown below.

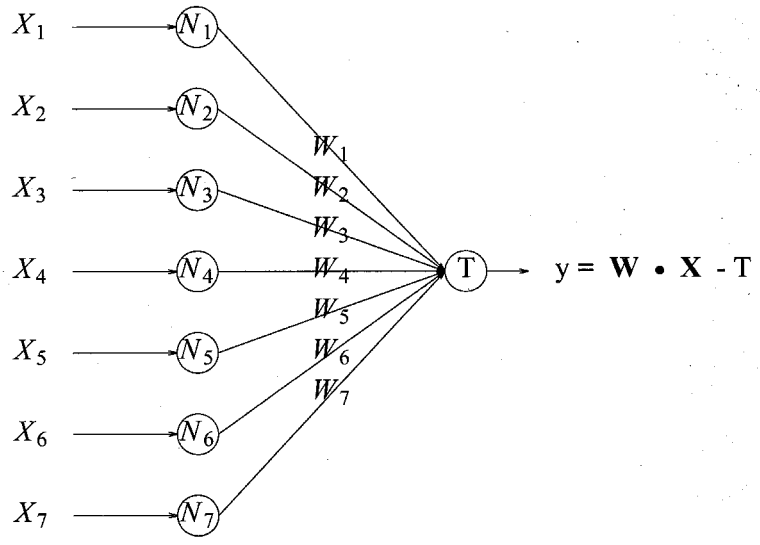


Figure 2. Single Linear Neuron

Stochastic Computation

The determination of the weight vector **W** and threshold T is via a labeled set of training examples.

4. Binary Classification With A Single Neuron

Consider the classification of digitized images of the digits zero and one. Upon presentation of an image, the neuron output must indicate whether a zero (hypothesis H_0) or a one (hypothesis H_1) was seen. One perspective is that the neuron can be modeled as a communication channel. For equal a priori hypothesis H_0 and H_1, one bit of information is always present at the channel input. The channel typically limits the flow of this information to the channel output. The channel capacity quantifies this constriction, indicating the average information content of a channel output, which is a *hypothesis decision*. Physically, the constriction in information flow is due to the projection mapping from an N-dimensional input space into a one-dimensional space. In most practical situations, information will be lost in the mapping. This loss is reflected in a channel capacity of less than one bit. The actual hypothesis decision is made by identifying segments of the one-dimensional real line with a choice of hypothesis : H_0 or H_1. Note that in general, information about a set of hypothesis outcomes is NOT equivalent to the probability of error in choosing among that set of hypotheses. However, for a two state system, there is a one-to-one relationship between information and P_{error}. [1]

Consider the single neuron in the figure above. We shall now determine how the output regions should be labeled with hypothesis decisions. Statistically, the problem is straightforward. For every point on the real axis, we assign the hypothesis which minimizes the probability of error.

That is, given a large number of samples in a labeled training set, and the weight vector **W**, project each training point onto **W**, and obtain a pool of labeled y points. For every point on the one-dimensional y axis, the corresponding hypothesis assigned to that point would be the hypothesis of the nearest projected labeled sample. For large size training sets, a histogram of the projections under the two hypotheses can be constructed. The decision as to whether an unknown data point projection came from hypothesis H_0 or H_1 could then be made based on the relative frequency of the two hypotheses within the histogram bin or cell the unknown data point fell into. For example, consider the two sets of points below. The set of 250 points labeled **0** yield the solid histogram. The set of 250 points labeled **1** yield the dotted histogram.

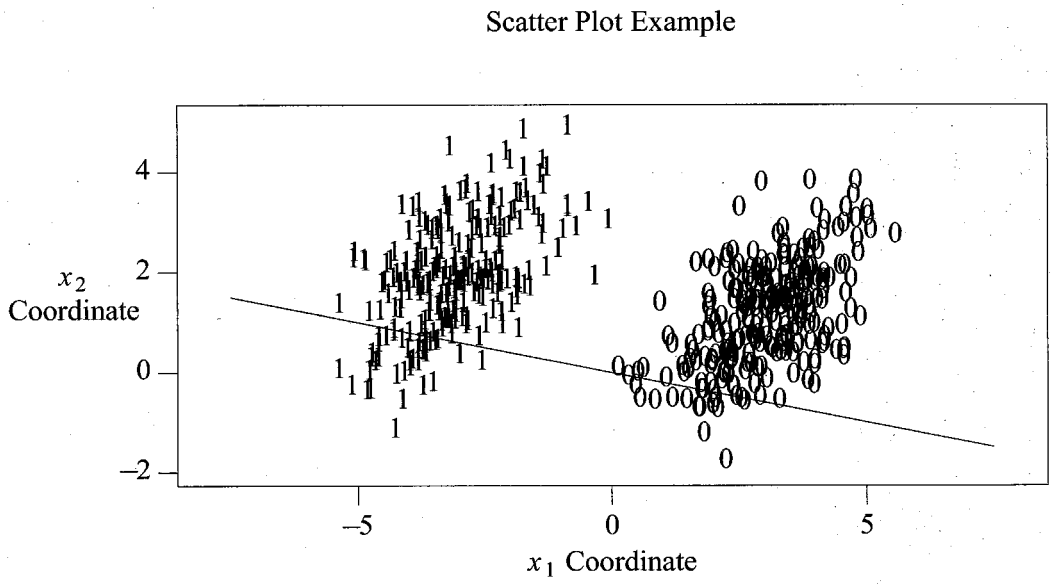


Figure 3. Two Hypothesis Scatter Histogram

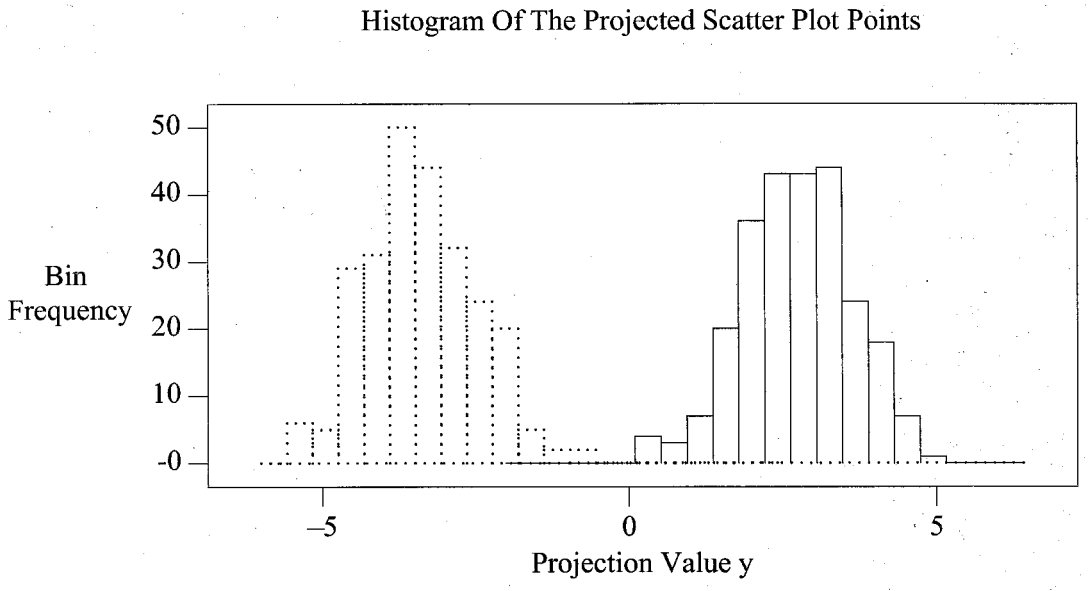


Figure 4. Two Hypothesis Scatter Plot

A good threshold appears to be zero. Thus, our decision rule for deciding H_0 versus H_1 would be that seen below.

$$y \equiv (w_1, w_2) \bullet \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \rightarrow \quad y \underset{H_0}{\overset{H_1}{<>}} 0.$$

Equation 2. Introduction To Determining Thresholds For Hypothesis Projections

The histograms shown are approximations to the conditional probability density functions $P[y|H_1]$ and $P[y|H_0]$. Here y is **W•X**, the projection of the two-dimensional data pairs $\{x_1, x_2\}$ onto the line shown. In the full N - dimensional input space of data points, **X**, the two

conditional probability density functions conditioned on the hypotheses are $P[\mathbf{X}|H_1]$ and $P[\mathbf{X}|H_0]$. In order for any function or network to statistically distinguish between the two hypotheses, these two N - dimensional probability density functions must be different. If this occurs, and the neuron weight and threshold are chosen wisely (which we shall discuss later), a neural network can take advantage of the differences between these two N - dimensional pdf's. The result will be that the two projection pdf's $P[y|H_1]$ and $P[y|H_0]$ will be different. A suitably chosen threshold will enable the system to make a better than random guessing decision as to which hypothesis occurred, given **X** as an input. How much better than random, and other questions related to network performance will be discussed as the concept is developed.

4.1 Random Variable Inputs And How A Neuron Uses Them

The neuron input vector **X** is considered to consist of N random variables x_i. If N is large, and the x_i's are uncorrelated or only weakly correlated, then the scalar quantity $y \equiv \mathbf{W} \bullet \mathbf{X}$ is a Gaussian random variable by the Central Limit Theorem.[2] [3] Shifting a Gaussian random variable by a constant T will only change the mean of the Gaussian random variable. Hence, $\mathbf{W} \bullet \mathbf{X}$ - T is also a Gaussian random variable by the Central Limit Theorem.

The application of the Central Limit Theorem to the projection pdf's $P[y|H_1]$ and $P[y|H_0]$ means both of these pdf's will appear either as Case I or Case II in the figure below. The case of equal means will turn out to be somewhat less common, and Case I will be what typical projections look like.

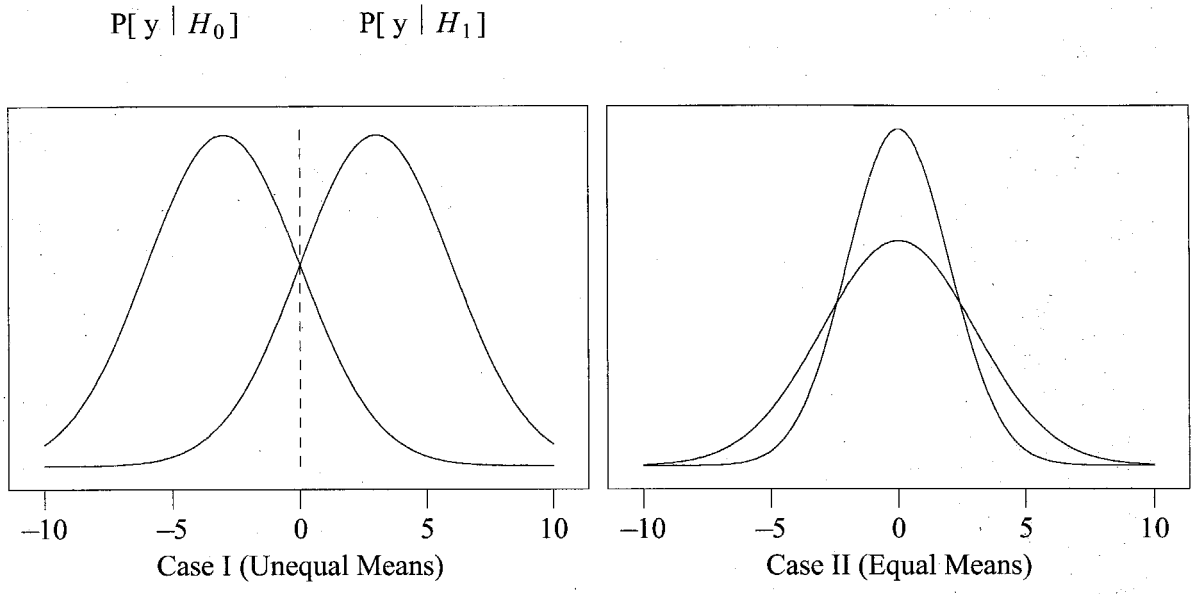


Figure 5. Case I : Unequal Means , Case II : Equal Means

For Case I, a scalar threshold T serves to shift the two probability density functions $P[y|H_i]$ symmetrically about zero. Identification of H_1 if y > 0 and H_0 if y < 0 can then be made. The threshold or shift is calculated to minimize the total probability of error. This error is the a priori weighted sum of the probability tails falling underneath the threshold in the figure for Case I and is shown as a dashed vertical line above.

$$P_{error} = P[H_0] \left\{ \text{Area Of } P[y|H_0] \text{ Above Threshold} \right\}$$
$$+ P[H_1] \left\{ \text{Area Of } P[y|H_1] \text{ Below Threshold} \right\}$$

Equation 3. Probability The Hypothesis Test Will Make An Error

Placing a step function at the output of the neuron creates the desired output binary decision variable z. Thus z indicates a choice between hypothesis H_0 and H_1.

$$z = \text{Unit Step}[y] = H_0 \text{ if } y < 0, \; H_1 \text{ if } y > 0$$

Equation 4. Step Function Neuron Sigmoid

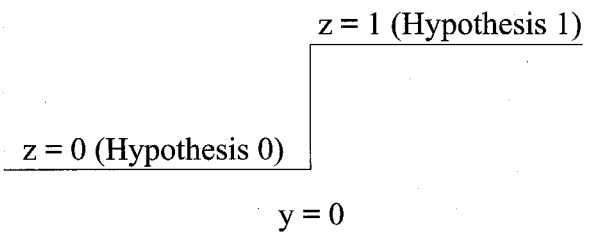


Figure 6. Output Threshold Step Function

4.2 Calculation Of The Weight Vector W

The optimum weight vector, $\mathbf{W}_{optimum}$, for any one neuron is calculated by minimizing the P_{error}. That is, the optimum weight is the vector **W** which yields the lowest overall system P_{error}. Intuitively, $\mathbf{W}_{optimum}$ must be a function of the statistics of both conditional hypotheses, $P[\mathbf{X}|H_0]$ and $P[\mathbf{X}|H_1]$. To see the exact functional relationship between the statistics and $\mathbf{W}_{optimum}$, a review of the Karhunen - Loeve representation is needed.

The K-L representation is a series expansion of an N-dimensional random vector. Each entry in the input data vector **X** below is a random variable.

$$\mathbf{X} = \begin{bmatrix} x_1 \\ . \\ . \\ . \\ . \\ . \\ x_N \end{bmatrix}$$

Equation 5. Random Vector $\mathbf{X}$

We expand the random vector $\mathbf{X}$ in a series, with random variables α_i multiplying N - dimensional deterministic unit magnitude vectors $\mathbf{E}_i$.

$$\mathbf{X} = \sum_{i=1}^{N} \alpha_i \, \mathbf{E}_i$$

Equation 6. N-Dimensional Random Vector Expansion

From a degree of freedom argument, note that all N α_i random variables are needed in the series to fully and accurately represent the N-dimensional random vector $\mathbf{X}$. Now we would like to sort the terms $\alpha_i \mathbf{E}_i$ so that $\alpha_1 \mathbf{E}_1$ is the most *important* term in our series expansion, $\alpha_2 \mathbf{E}_2$ is the next most important term, and so on, all the way up to i = N. In this way, if we truncate the series expansion at some number M < N, we will be getting the *best* possible representation we can, with the limited random degrees of freedom $\{ \alpha_1, \ldots, \alpha_M \}$ available. We consider the expected mean square error to be our measure of importance. That is, we seek M random variables α_i and corresponding deterministic vectors $\mathbf{E}_i$ such that if we define $\hat{\mathbf{X}}$ as $\hat{\mathbf{X}} = \sum_{i=1}^{M} \alpha_i \, \mathbf{E}_i$, then the expected value of $|\mathbf{X} - \hat{\mathbf{X}}|^2 = (\mathbf{X} - \hat{\mathbf{X}})^T \bullet (\mathbf{X} - \hat{\mathbf{X}})$ is smaller than if

we pick any other α_i or $\mathbf{E}_i$. The Karhunen-Loeve representation is exactly this expansion.[4] [5] [6] [7] The $\mathbf{E}_i$ are the eigenvectors of the correlation matrix **C** of the N-dimensional probability density function P[**X**]. **C** is the expected value of the outer product of the random vector **X**.

$$\mathbf{C} \equiv E[\mathbf{X}\mathbf{X}^T] \equiv \begin{bmatrix} E[x_1^2] & E[x_1\ x_2] & \cdots & E[x_1\ x_N] \\ E[x_1\ x_2] & E[x_2^2] & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ E[x_1\ x_N] & E[x_2\ x_N] & \cdots & E[x_N^2] \end{bmatrix}$$

Equation 7. Correlation Matrix

Since a correlation matrix is positive definite, and symmetric, the eigenvalues are positive. The Karhunen - Loeve theory tells us that the relative importance of the {random variable α_i,vector $\mathbf{E}_i$} pair is directly related to the magnitude of the eigenvalue λ_i corresponding to the eigenvector $\mathbf{E}_i$. The largest λ_i eigenvalue indicates the most important $\{\alpha_i,\mathbf{E}_i\}$ pair. Decreasing eigenvalues indicate successively less important expansion terms.[1]

The Karhunen-Loeve representation has several other properties which will be useful in the analysis below. First, the eigenvectors $\mathbf{E}_i$ are orthonormal. Thus, $\mathbf{E}_i \bullet \mathbf{E}_j = \delta_{ij}$. Second, the random variables α_i are uncorrelated. That is, $E[\alpha_i\ \alpha_j] \equiv \delta_{ij}\ E[{\alpha_i}^2]$.

1. Assume the random variables x_i are real. If the x_i are complex, replace the word symmetric with Hermitian. The argument to follow is still valid for complex x_i.

The Karhunen-Loeve theory allows one to expand the two conditional vector pdfs $P[\mathbf{X}_{H_1}]$ and $P[\mathbf{X}_{H_0}]$ as follows.

$$\mathbf{X}_{H_0} = \sum_{i=1}^{N} \alpha_i^{H_0} \ \mathbf{E}_i^{H_0} \qquad\qquad \mathbf{X}_{H_1} = \sum_{i=1}^{N} \beta_i^{H_1} \ \mathbf{F}_i^{H_1}$$

Equation 8. Karhunen-Loeve Expansion For The Conditional Hypothesis Input Vectors

Before we proceed, we will need a lemma about the magnitude of the dot product of two arbitrary vectors in N-dimensional space. Suppose we have two N-dimensional deterministic vectors **A** and **B**. We wish to know the average value of their inner product **A** • **B**. Since these two vectors can point in any direction in N space, and are arbitrary, and hence unrelated to each other, their average dot product is the integral below.

$$\frac{\int_{N-Dim\ Unit\ HemiSphere} x_1 \ dx_1 \ dx_2 \ . \ . \ . \ . \ dx_N}{The\ Surface\ Area\ Of\ An\ N-Dimensional\ Hemisphere} \rightarrow \quad zero \quad as \quad N \rightarrow \infty$$

Equation 9. Magnitude Of The Inner Product Of Two Arbitrary N- Dimensional Vectors

Above, we have set the vector **A** in the x_1 unit vector direction. Integration over the N - dimensional hemisphere centered around this unit vector yields the average **A** • **B** dot product magnitude. Note that we are *not* including negative projections between **A** and **B**. The negative projections all lie in the opposing hemisphere, and yield symmetric magnitude answers with a negative sign. Since we divide by the surface area of a hemisphere, we are accounting for this

symmetry in our answer. The result we shall use is that the dot product $\rightarrow 0$ as $N \rightarrow \infty$. Thus, for large N, we shall conclude $\mathbf{A} \bullet \mathbf{B} \approx 0$.

This asymptotic, large N result is in keeping with our analysis, which will be based on large data input dimensionality. Indeed, the theme of this thesis is that large dimensionality *helps* a stochastic system, and hence a neural network, process data. We saw one aspect of this in the application of the Central Limit Theorem. Below we shall see another large dimensionality advantage. This is all in contrast to Bellman's so - called *Curse Of Dimensionality* rule.[2]

Returning to our Karhunen-Loeve analysis, take only the first term of the expansion above for each of the two conditional pdf vectors.

$$\mathbf{X}_0 \approx \alpha_1 \mathbf{E}_1 \qquad\qquad \mathbf{X}_1 \approx \beta_1 \mathbf{F}_1$$

Equation 10. The Principle Component Of The Two Conditional Hypothesis Data Vectors

Thus, we approximate the N-dimensional random vector $\mathbf{X}_i$ by the one-dimensional random representation $\mathbf{X}_i \approx \alpha_1^{H_i} \mathbf{E}_1^{H_i}$. Consider the vector difference $\mathbf{W} \equiv \mathbf{F}_1 - \mathbf{E}_1$. For an unknown image $\mathbf{X}$, the projection $\mathbf{W} \bullet \mathbf{X}$ is equal to $\mathbf{F}_1 \bullet \mathbf{X} - \mathbf{E}_1 \bullet \mathbf{X}$. If $\mathbf{X} \in H_1$, then $\mathbf{W} \bullet \mathbf{X} \approx \mathbf{F}_1 \bullet \mathbf{X}$. If $\mathbf{X} \in H_0$, then $\mathbf{W} \bullet \mathbf{X} \approx - \mathbf{E}_1 \bullet \mathbf{X}$. The two hypothesis eigenvectors $\mathbf{F}_1$ and $\mathbf{E}_1$ are the Minimum Mean Square Error estimators (MMSE) for $\mathbf{X} \mid H_1$, and $\mathbf{X} \mid H_0$ respectively. Thus, $E[(\mathbf{X} \mid H_0 - \alpha_1 \mathbf{E}_1)^2]$ is at a minimum when $\mathbf{X} \in H_0$.

2. Bellman's adage states that, in general, information processing becomes computationally more difficult exponentially as the dimensionality of the data increases.

Represent **X** as a signal **S** buried in noise **n**. The principle eigenvector projection is akin to making a one - dimensional measurement of $\mathbf{X} \mid H_0$ which injects the least possible noise into the system. That is, we represent the original signal **X** with the approximation $(\mathbf{X} \bullet \mathbf{E}_1)\ \mathbf{E}_1$, and this approximation has the lowest noise of any approximation of $\mathbf{X} \in H_0$.

In everything which follows, a communication systems view is held. There is a signal buried in noise, which is embedded in $\mathbf{R}^N$. The neural network must find and identify the signal, and determine whether the signal was more likely to be drawn from one of two densities : $P[\mathbf{X} \mid H_1]$, or $P[\mathbf{X} \mid H_0]$. In the process of making measurements to determine which hypothesis is more likely true, noise minimization is important so as to increase the quality of the decision. Using principle component eigenvectors minimizes the noise injection, during measurement, into the system.[8]

The specific construction of the weight vector **W** from the two hypotheses principle components is motivated by the following argument(s). For a distribution in $\mathbf{R}^N$ with mean $\neq \mathbf{0}$, the principle eigenvector obtained via the reinforcement algorithm used below will always point in the direction of the mean. Consider the two dimension example below.

$$\mathbf{X}_0 \approx \begin{bmatrix}1\\1\end{bmatrix} \quad \mu_0 \equiv \begin{bmatrix}1\\1\end{bmatrix} \quad \mathbf{C}_0 \equiv \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} \quad \begin{matrix}\lambda_1 \equiv 2\\ \lambda_2 \equiv 0\end{matrix} \quad \mathbf{E}_1 \equiv \pm \begin{bmatrix}1\\1\end{bmatrix} \quad \mathbf{E}_2 \equiv \pm \begin{bmatrix}1\\-1\end{bmatrix}$$

Equation 11. $\mathbf{X}_0$ Mean Projection Onto $\mathbf{E}_1$

If the positive sign is taken for $\mathbf{E}_1$, the projections of $\mathbf{X} \in H_0$ onto $\mathbf{E}_1$ will be positive. Conversely, if the negative sign is taken for $\mathbf{E}_1$, the projections of $\mathbf{X} \in H_0$ onto $\mathbf{E}_1$ will be

negative. However, the reinforcement algorithm always yields the positive signed eigenvector, so that the mean of the projected data is always positive : $E[\ \mathbf{X} \bullet \mathbf{E}_1\ ] \geq 0$. We take advantage of this fact, and create approximately antipodal signals. That is, by taking the difference $\mathbf{F}_1 - \mathbf{E}_1$, we create a separation in the means of the two hypotheses projections, since $E[\ \mathbf{X}(\in H_1) \bullet \mathbf{W}\ ] \geq 0$ and $E[\ \mathbf{X}(\in H_0) \bullet \mathbf{W}\ ] \leq 0$. This separation is roughly centered about zero.

The reason we do not take a linear combination of the form $a\ \mathbf{F}_1 - b\ \mathbf{E}_1$, where $a\ ,\ b \in R$, and $a\ ,\ b \geq 0$, is because pure amplification of a signal does not improve the SNR. That is, under case I above, the area of the tails under the threshold is related to the SNR $\equiv \frac{\mu_0}{\sigma_0}$. Multiplying the random variable obtained via the projection onto $\mathbf{E}_1$ by the real positive constant a changes the mean of the projection random variable to $\mu_0' \equiv a\ \mu_0$ and the variance to $\sigma_0'^2 \equiv a^2\ \sigma_0^2$. The new signal to noise ratio, $\text{SNR}' \equiv \frac{\mu_0'}{\sigma'} \equiv \frac{\mu_0}{\sigma} \equiv$ old SNR. Thus, a pure gain will not increase the neurons ability to make a correct decision. Note that crucial to the line of reasoning is the use of the orthogonal hypothesis assumption. If $\mathbf{X}(\in H_1) \bullet \mathbf{E}_1 \neq 0$, or $\mathbf{X}(\in H_0) \bullet \mathbf{F}_1 \neq 0$, then the neuron indifference to gain argument does not hold.

To summarize our search for $\mathbf{W}_{optimum}$, we calculate the two correlation matrices for $\mathbf{X}_0$ and $\mathbf{X}_1$, and find the eigenvectors corresponding to the largest eigenvalue for each matrix. Taking $\mathbf{W}_{optimum}$ equal to the difference of these two eigenvectors $\mathbf{E}_{H_1} - \mathbf{E}_{H_0}$, serves as a Minimum Mean Square Error (MMSE) approximation to a neural network consisting of a single neuron

implementing a binary hypothesis test. The threshold used for the decision test on the projected data points is approximately zero. Positive projections are concluded as from H_1, and negative projections are decided as coming from H_0.

4.3 Handwritten Digit Recognition Example

As an example to validate the above assumptions, consider the following data calculated from a database of handwritten digits. [9] A total of 1616 images of the handwritten digit **6**, and 1616 images of the digit **7** were used to construct the Principle Component projection histograms. The images consisted of 32 by 32 (1024 total) pixels, with each pixel represented by an 8 bit grayscale number. The continuous curve is the Gaussian density function with the mean and variance tabulated from the projection data.

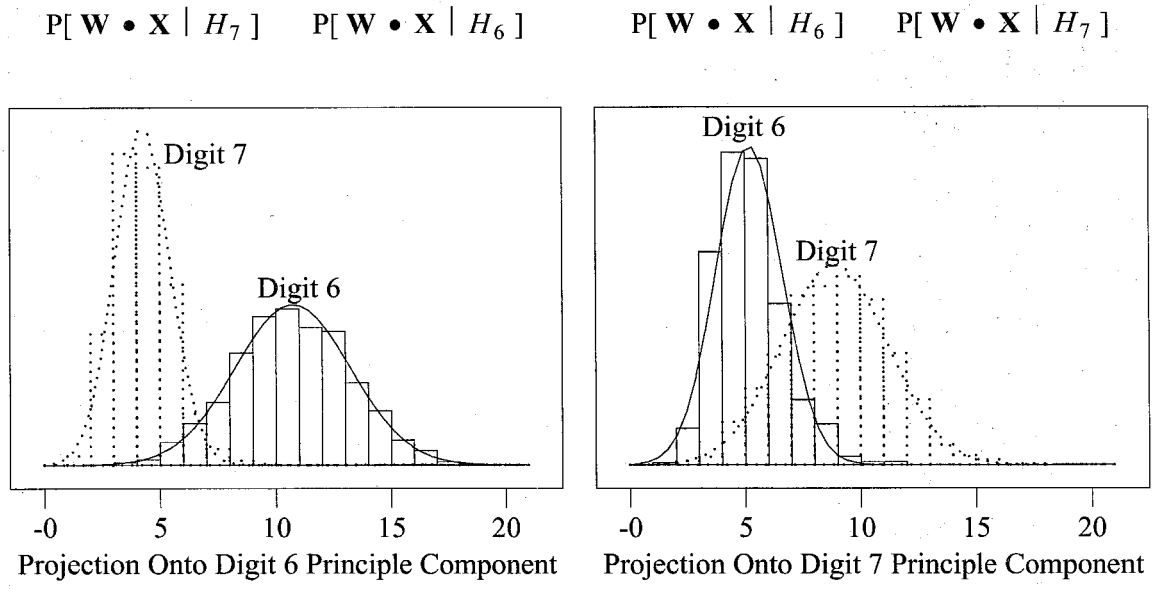


Figure 7. Histogram Of Cedar Database Handwritten Digit 6 And 7 Projections

In both the plot above and the plot below, the continuous curve is the probability density function which the histograms should approximate.

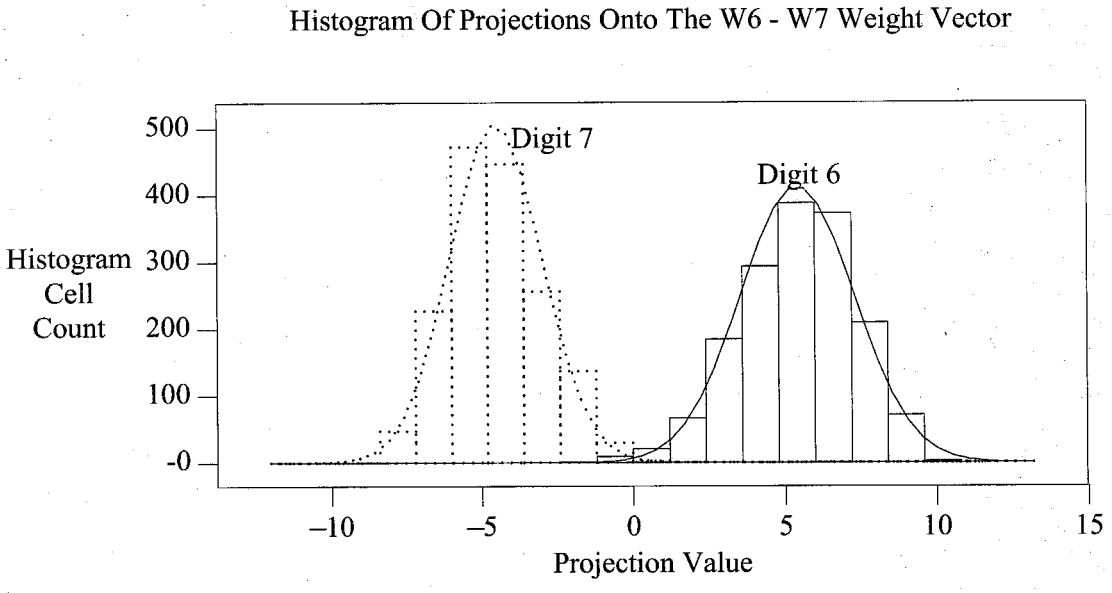


Figure 8. Histogram Of Cedar Database Handwritten Digit 6 And 7 Projections

For later reference, the mean and variance of the projections under the different hypotheses is shown in the table below.

Weight Vector	Digit 6		Digit 7	
	Mean	Variance	Mean	Variance
$\mathbf{W}_6$	10.726	6.224	4.251	1.408
$\mathbf{W}_7$	5.172	2.070	8.886	5.289
$\mathbf{W}_6 - \mathbf{W}_7$	5.418	3.52	-4.521	2.346

TABLE 1. Projection Statistics

In the plot below, the threshold of the neuron with the weight vector $\mathbf{W}_6 - \mathbf{W}_7$ is varied, and 1616 Six images, and 1616 Seven images (3232 images total) are passed through the neuron. The probability of a correct decision is then tallied. Thus, the plot below should give us an idea of the symmetry of the probability of error about the theoretically expected threshold value of zero.

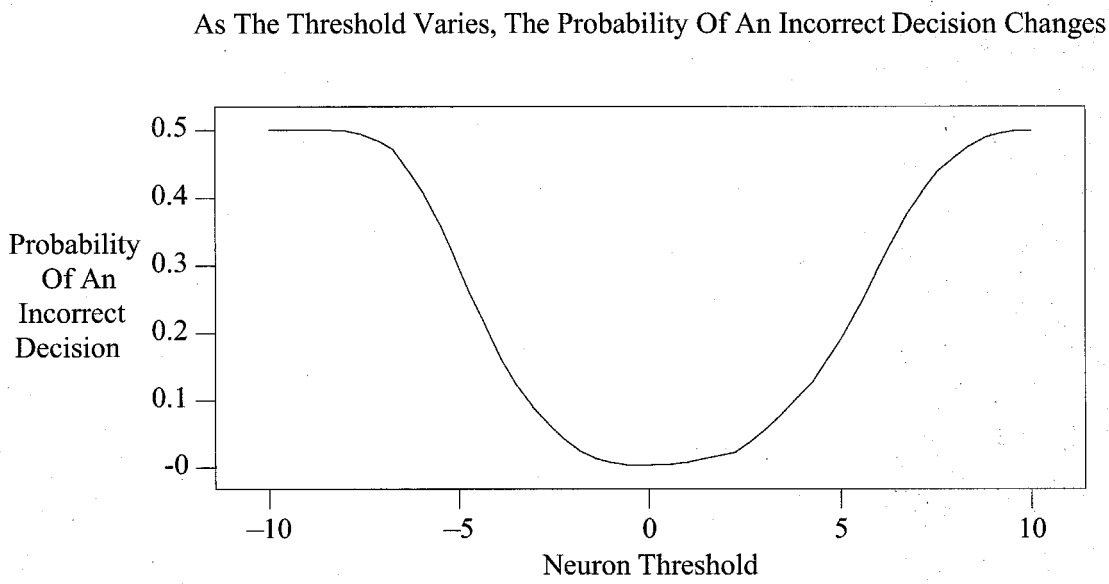


Figure 9. Estimation Decision Error Via Brute Force Variation Of The Neuron Threshold

A closeup of the zero threshold region shows the minimum error is achieved for a threshold of -0.184. The optimum threshold yielded a probability of error of 0.0031 or ten errors out of a total of 3232 images looked at. The jagged nature of the plot is due to the discrete number of points (3232) used in calculating the probability of a misclassification.

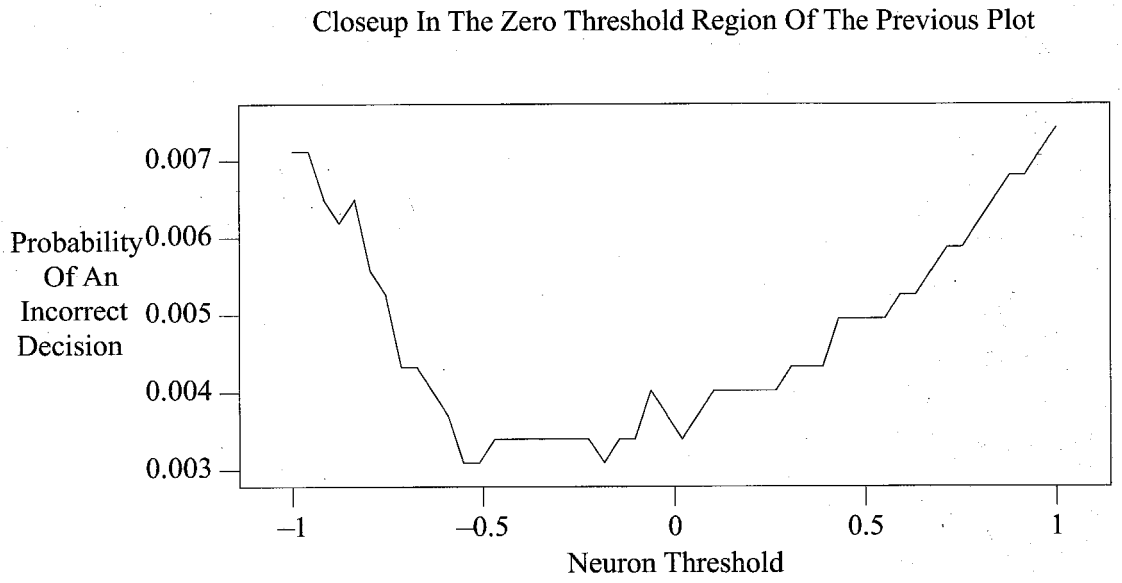


Figure 10. Estimation Decision Error Via Brute Force Variation Of The Neuron Threshold

5. Binary Symmetric Channel Model

A useful picture which aids visualizing single neuron behavior in an information theory context is the Binary Symmetric Channel (BSC) model. Without loss of generality, the two hypotheses are considered equal in a priori probability, meaning we shall consider the areas of the tails falling on both sides of the threshold to be equal. Consider the decision which occur's by thresholding, as shown below.

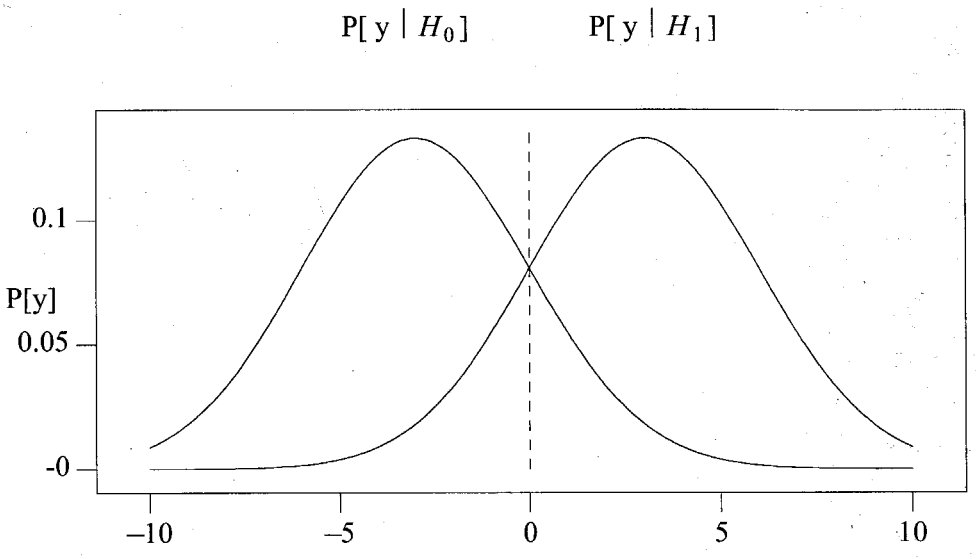


Figure 11. Prob Of Error Is The Area Under The Tails Past The Threshold

The area under the tails past the threshold represents the probability that any one message or bit is misidentified. This area is the crossover probability, ε, for the BSC, as shown below.

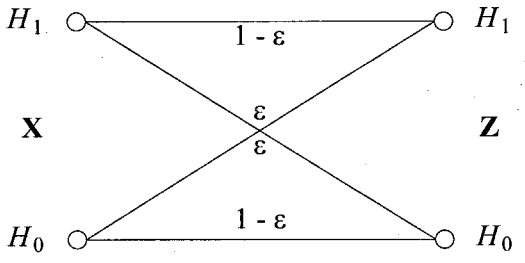


Figure 12. Binary Symmetric Channel Model

The idea is that one bit of information is always being presented at the BSC input. This bit represents which digit, of two possibilities, the *sender* intended to transmit. The digit is

represented as a single point in a high-dimensional space. Hence, the single bit of information is distributed across the large dimension input space. The neuron corresponds to a channel which is projecting the high-dimensional space into a one-dimensional space. By calculating the probability distribution of the projected values, one can calculate the mutual information of the BSC input and output random variables. Since the channel is assumed, for simplicity, to be symmetric, the mutual information is a maximum when the input hypotheses are equal in probability. That is when $P_{H_0} = P_{H_1} = ½$. The resulting channel capacity C, as a function of the crossover probability ε, is seen below.

$$C(\varepsilon) = 1 + \varepsilon \log_2(\varepsilon) + (1 - \varepsilon) \log_2(1 - \varepsilon)$$

Equation 12. Channel Capacity

A plot of C(ε) versus ε is below.

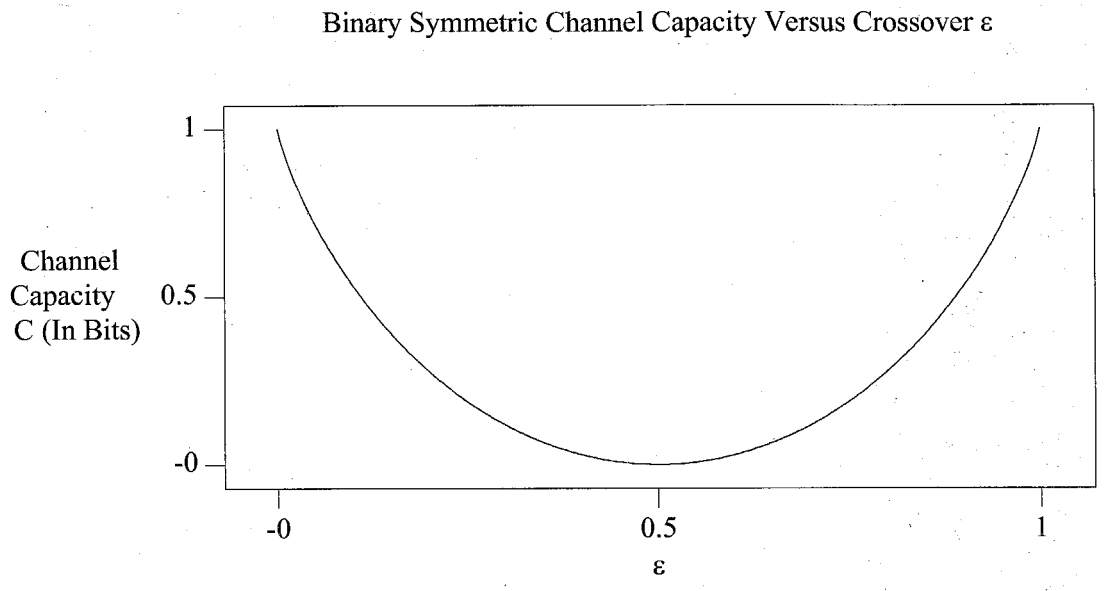


Figure 13. Binary Symmetric Channel Capacity Versus Crossover ε

Channel capacity is an alternative measure of probability of error for binary hypothesis testing. The mental picture of the BSC and channel capacity will be an aid in discussions below on multiple neuron per layer, multiple layer networks.

6. Multiple Neuron Hidden Layer Architecture

The above treatment is for one neuron. However, as discussed in the foreword, the power of parallel processing relies on many simple units operating cooperatively together. To analyze how neurons could work in parallel, consider three neurons which are placed side by side, and their results voted on. That is, each neuron is expected to make a binary Yes/No decision based on the same input data. The majority voting decision of the three neurons is taken to arrive at

the system decision.

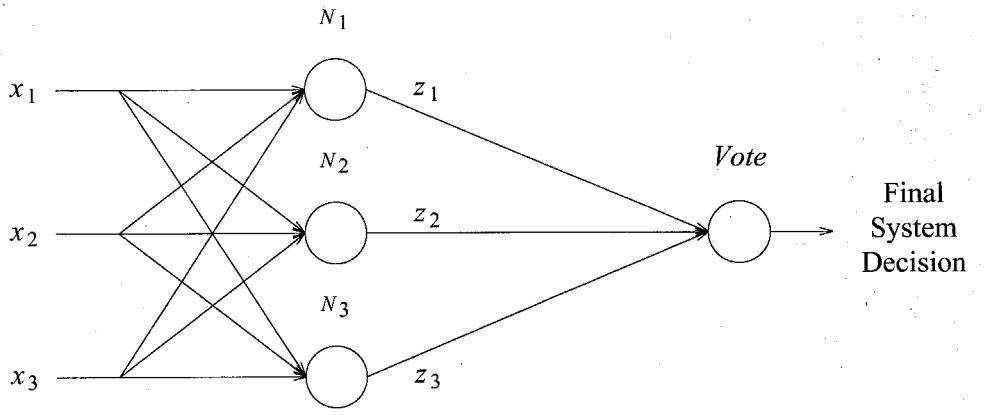


Figure 14. Three Neurons In Parallel

Consider the output set $\mathbf{Z} = \{z_1, z_2, z_3\}^T$. The $\mathbf{Z}$ vector is a binary codeword representing the layers output decision. Before we introduce how to determine the neuron weight set $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\}$, we digress and discuss a feature of real-life pattern recognition situations which we shall exploit.

6.1 Eigenvalue Decay For Handwritten Digits

Consider an ensemble or training set with a total of M images. Each image consists of N x N pixels. Represent these M images as N^2-element vectors $\{\mathbf{X}_{i=1,2,...M}\}$. The correlation matrix for this set of vectors can be constructed as seen below.

$$\mathbf{C} = \frac{1}{M}\sum_{i=1}^{M} E[\mathbf{X}_i \mathbf{X}_i^T] \equiv \begin{bmatrix} E[x_1^2] & E[x_1\ x_2] & \cdots & E[x_1\ x_N] \\ E[x_1\ x_2] & E[x_2^2] & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ E[x_1\ x_N] & E[x_2\ x_N] & \cdots & E[x_N^2] \end{bmatrix}$$

Equation 13. Compiling The Correlation Matrix For The Image Set

C is a positive definite, symmetric matrix, and the eigenvalues are all positive. We order the eigenvalues from largest magnitude to smallest magnitude. The plots below show this eigenvalue ordering for three databases of handwritten digits. The digit to the left of the first data point indicates which set of images that eigenvalue curve represents. The rolloff of the eigenvalues for the correlation matrices for all ten digits is rapid, and the magnitudes of corresponding number (e.g. : largest, second largest, etc.) eigenvalues, across the three databases, are similar.

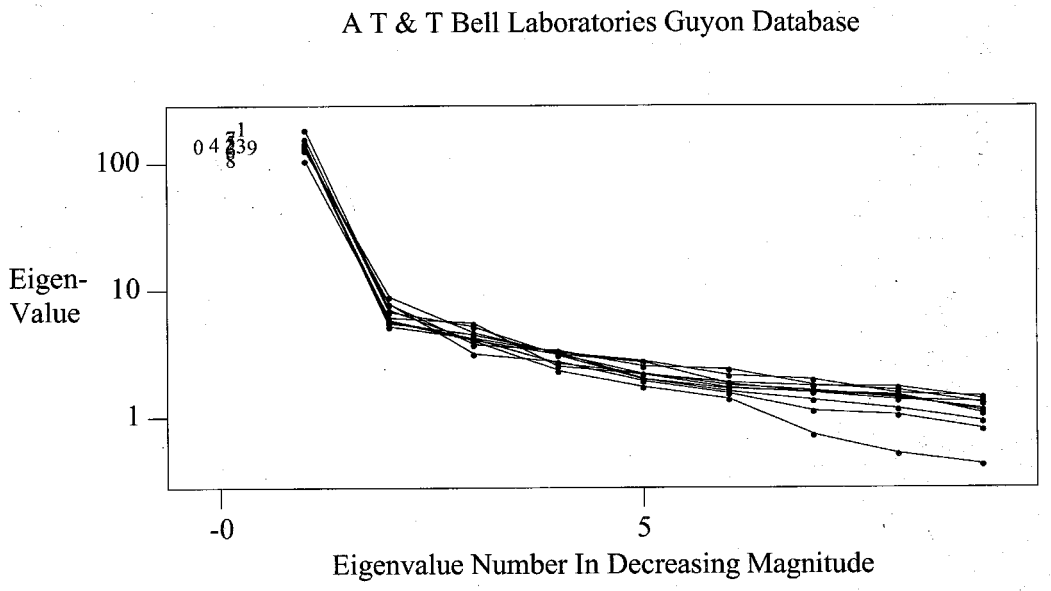


Figure 15. A T & T Bell Laboratories Database Eigenvalue's

The A T & T Bell Laboratories database is composed of 16 by 16 binary pixel images.[10] Below is the same data from the CEDAR handwritten digits database.[11]

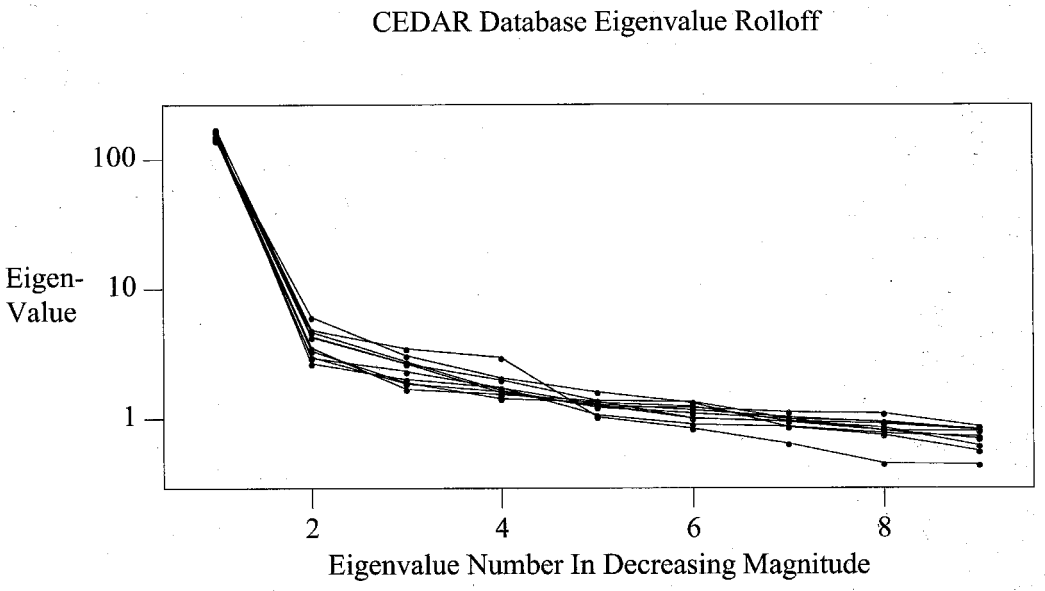


Figure 16. Cedar Database Eigenvalue Rolloff

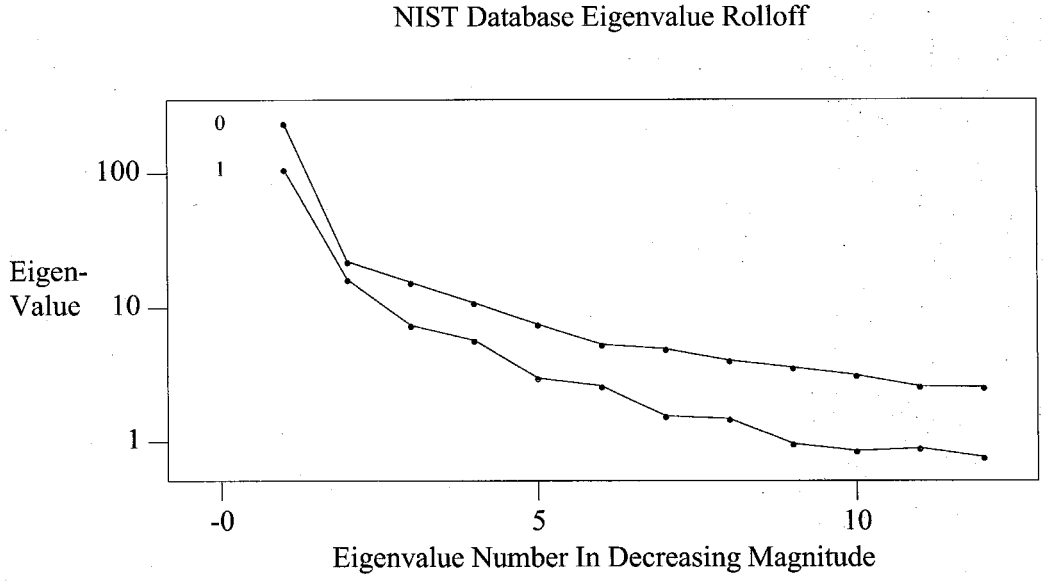


Figure 17. NIST Database Eigenvalue Rolloff

Stochastic Computation

The CEDAR images were of varying size, since they were obtained from digitized images of envelopes passing through the Buffalo, New York Post Office. They were converted to a 32 by 32 8 bit pixel format using the *PbmPlus* set of filters.[12] The NIST images were supplied in 32 by 32 binary pixel format.

The eigenvalues were determined in three fashions. The CEDAR eigenvalues were calculated using C code implementing the OJA algorithm to be discussed below. The A T & T eigenvalues were obtained by first calculating the outer product correlation matrix C as described above. The matrix was passed through MATLAB, and the eigenvalues were extracted using MATLAB routines.[13] The NIST eigenvalues were obtained using C code written by the author which implemented the power method of principle eigenvector and eigenvalue determination. By successively projecting the input space onto the subspace orthogonal to the principle eigenvector, successive principle eigenvalues and eigenvectors can be found. This technique is particularly effective in determining the principle eigenvalues for matrices which are highly singular, which is the situation for handwritten digit correlation matrices. [14] [15] Note there is essentially no difference in the three results, despite having different algorithms operate on distinct databases.

The rapid rolloff of the eigenvalues indicates there is a relatively small subspace of the large dimensional input space which is very important in representing the random input vectors $\mathbf{X}_i$. The exact subspace basis varies from digit to digit. Since the eigenvalue rolloff is rapid, we can assume, by applying our result for random dot products above, that the subspaces of activity for any two digits is orthogonal. Thus, we can represent the entire input space for a two hypothesis

test as $\mathbf{X} \in \mathbf{R}^N \equiv \mathbf{S}_0 \oplus \mathbf{S}_1 \oplus \mathbf{S}_{Noise}$. The space $\mathbf{S}_i$ represents the subspace or basis $\subset \mathbf{R}^N$ where most hypothesis H_i vectors have significant projections. Since the dimension of $\mathbf{S}_0$ and $\mathbf{S}_1$ are assumed to be a small fraction of the total input space dimension N, the subspace $\mathbf{S}_0 \oplus \mathbf{S}_1$ occupies only a small portion of the entire space $\mathbf{R}^N$. The remaining space $\overline{\mathbf{S}_0 \oplus \mathbf{S}_1} \equiv \mathbf{R}^N - \{\mathbf{S}_0 \cup \mathbf{S}_1\}$ is denoted $\mathbf{S}_{Noise}$ and represents the subspace of $\mathbf{R}^N$ where neither H_0 nor H_1 data vectors typically have projections. Applying our large N lemma for random dot products, we can conclude that typically $\mathbf{S}_0 \perp \mathbf{S}_1$.

The subspace orthogonality assumption has wide ranging consequences in this work. First, it provides a method to determine the single layer, multiple neuron weight set $\{\mathbf{W}_i\}$. The set of weights $\{\mathbf{W}_1, \mathbf{W}_2, \ldots\ldots \mathbf{W}_L\}$ for the neurons in a single layer network of width L, should span both of the subspaces $\mathbf{S}_{0,1}$ to provide a reasonably good approximation to the two conditional random vectors $\mathbf{X}_0$ and $\mathbf{X}_1$. Let the dimension of the subspace $\mathbf{S}_i$ be M_i. Let Q equal the maximum of the two subspace dimensions M_i, and let K equal the minimum of the M_i. An orthonormal basis for S_i is the first M_i Karhunen-Loeve eigenvectors, $\mathbf{E}_j^{H_i}$, of the correlation matrix for H_i. One weight vector representation which meets the spanning requirement of both S_i is the following.

$$\text{For j} \in \left\{ 1,\ldots\ldots,K \right\}, \quad \mathbf{W}_j = \mathbf{E}_j^1 - \mathbf{E}_j^0$$

$$\text{For j} \in \left\{ K+1,\ldots\ldots,Q \right\}, \quad \mathbf{W}_j = \pm\mathbf{E}_j^*$$

Equation 14. Multiple Neuron Weight Vector Determination

Here * is the hypothesis which has the larger dimension of the two subspace's $\mathbf{S}_i$, and the ± is taken depending on whether $\mathbf{S}_1$ has the largest M (choose +) or $\mathbf{S}_0$ has the largest M (choose -).

The above decomposition indicates it is of no use to attempt to produce a basis which represents points outside of either $\mathbf{S}_i$, since we are assuming all significant projections occur inside the subspaces $\mathbf{S}_0$ or $\mathbf{S}_1$. Thus, a limit on the number of neurons in any one layer is found to be Q, defined above. That is, L ≡ Q. Adding more neurons to any one layer essentially adds weight vectors which span space outside the $\mathbf{S}_i$, and hence inside $\mathbf{S}_{Noise}$. Projections onto these weight vectors constitutes noise, and hence will only decrease the performance of the overall system by raising the noise floor which a final voting neuron must deal with to arrive at a decision.

The identification of where the subspaces $\mathbf{S}_i$ rolloff or *end* is somewhat subjective. Due to the noise and other variations across the images, the eigenvalues in the spectrum rolloff above do not become perfectly zero, but only *small* outside the subspaces of interest, $\mathbf{S}_i$. In the above graphs, and in most practical cases, it is clear where the eigenvalues have decayed into the noise floor, and the resulting subspace identification can be made easily.

6.1.1 Determining Single Layer, Multiple Neuron Weight Sets

Before we proceed with an example, we note that the construction above of the $\{\mathbf{W}_j\}$ automatically ensures the $\{z_j\}$ are statistically independent under both hypotheses. The orthogonality of the two subspaces $\mathbf{S}_i$ under the large N approximation, coupled with the lemma above, have allowed us to decorrelate the z_i random variables under both hypotheses. This essentially means the weight vectors $\{\mathbf{W}_j\}$ simultaneously diagonalize the correlation matrices

of $\mathbf{X}_1$ and $\mathbf{X}_0$ via a single linear transformation. Thus, regardless of which hypothesis has been presented to the input via the $\mathbf{X}$ input vector, the z_j hidden layer outputs will be uncorrelated random variables. Since we are assuming the input x_i are random variables, and the Central Limit Theorem holds, the z_j are Gaussian random variables. Uncorrelated Gaussian random variables are statistically independent. [16] Thus, from another perspective, the simultaneous diagonalization of the two correlation matrices has determined the hidden layer weight vectors. The columns of the diagonalizing transformation matrix are the weight vectors for the different neurons.[17] This independence property obtained almost as an afterthought from the Karhunen-Loeve eigenvector difference algorithm is an additional bonus in disguise, since coding theory tells us that the maximum information content is obtained when each element of the coding vector is statistically independent from the other elements.[18] Thus, by using the differences of the Karhunen Loeve expansion vectors $\mathbf{E}_i^{H_1}$ and $\mathbf{E}_i^{H_0}$, the ensemble of neurons are working together to extract as much *information* as possible from the high-dimensionality input space, under the mean-square approximation criteria. Since the z_j random variables are statistically independent under both hypotheses, the set $\{z_j\}$ can be thought of as a binary code word being passed from the hidden unit layer to a voting neuron. The voting neuron implements the majority decision rule. This type of majority rule is called hard decision decoding in the statistical communication literature.

6.2 Extending The Binary Symmetric Model For Multiple Neurons in Parallel

As presented above, the idea of a single neuron acting as a channel for one bit of information allowed us to connect the neuron with the Binary Symmetric Channel (BSC) model of

communication. For multiple neurons operating in parallel, this picture can be extended by considering multiple BSC's in parallel. Each neuron in a single layer network corresponds to one BSC. The figure below represents a three neuron layer, with a layer output $\mathbf{Z} \equiv \{z_1, z_2, z_3\}^T$.

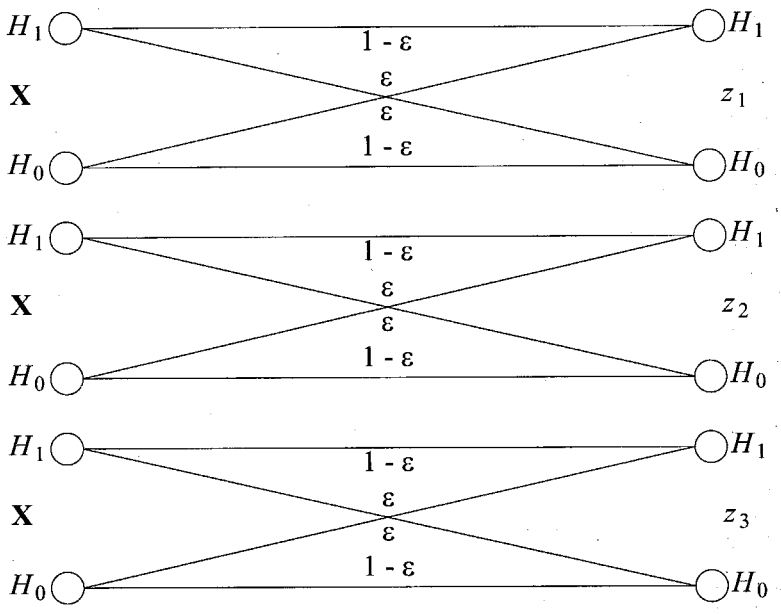


Figure 18. Three Binary Symmetric Channels In Parallel Neuron's

The probability that the ensemble of three BSC's will make a misidentification is summarized by the formula below. Key to this formula is the assumption that the three neurons (channels) are experiencing misidentifications independently, which occurs when the neuron outputs are statistically independent under both hypotheses.

$$\varepsilon' = (1 - \varepsilon_1)\varepsilon_2\varepsilon_3 + \varepsilon_1(1 - \varepsilon_2)\varepsilon_3 + \varepsilon_1\varepsilon_2(1 - \varepsilon_3) + \varepsilon_1\varepsilon_2\varepsilon_3$$

Equation 15. Prob. Of Three Parallel BSC's Making An Error Using Majority Voting

The plot below demonstrates the performance improvement one attains with three parallel BSC's versus just one, as ε is varied. For simplicity, we assume below that each BSC has an identical crossover probability. This is not true in practice, as will be discussed later.

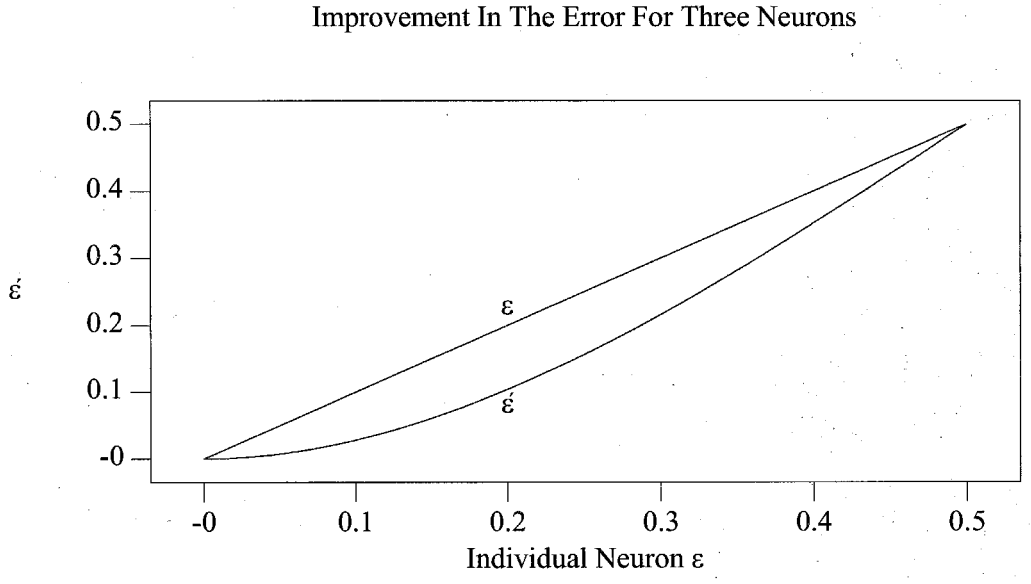


Figure 19. Effective Crossover For Three Neurons In Parallel

The improvement in channel capacity is seen below.

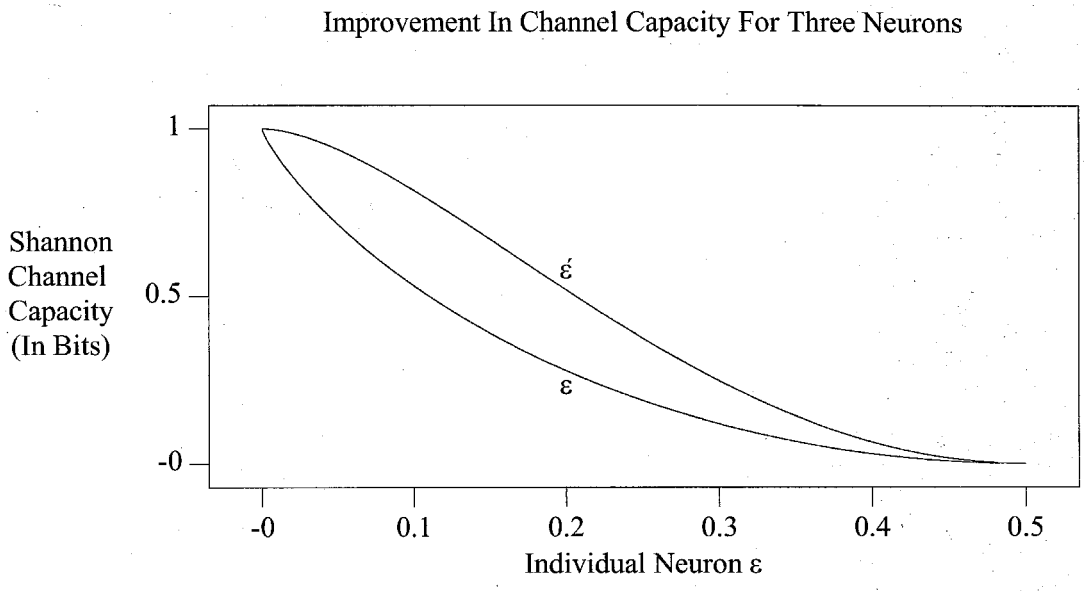


Figure 20. Effective Shannon Channel Capacity (In Bits) For Three Neurons

The ensemble of three BSC's can be pictured as a single effective BSC with crossover probability έ. The effective BSC models the hard voting decision performance of the entire layer. έ is the probability that the three neurons will simultaneously make either two or three decision errors. Since the probability of simultaneous errors is less than the probability that any one neuron will make a single error, overall system performance improves. Adding more neurons in parallel further aids the overall system performance. For N neurons placed in parallel, each producing outputs statistically independent under both hypotheses, the entire voting *system* is in error when ½ N or more of the neurons simultaneously make a misidentification. This becomes increasingly rare, for a fixed single neuron P_{error}, as the number N of neurons is increased. Extending the effective BSC concept outlined above for

three neurons, a parallel bank of BSC's is equivalent to a single effective BSC with crossover probability έ. The plot below shows έ as a function of the number of independent parallel neurons/BSC's, for various individual channel error rates.

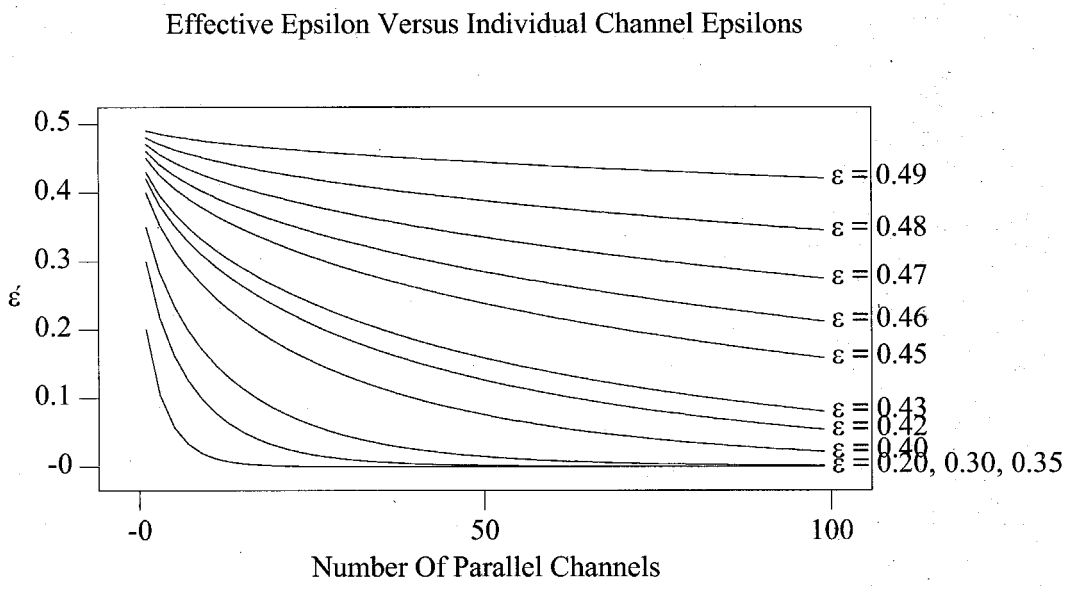


Figure 21. Effective Epsilon Versus Single Channel Epsilon

Below, the improvement in channel capacity is plotted as the number of parallel channels is varied.

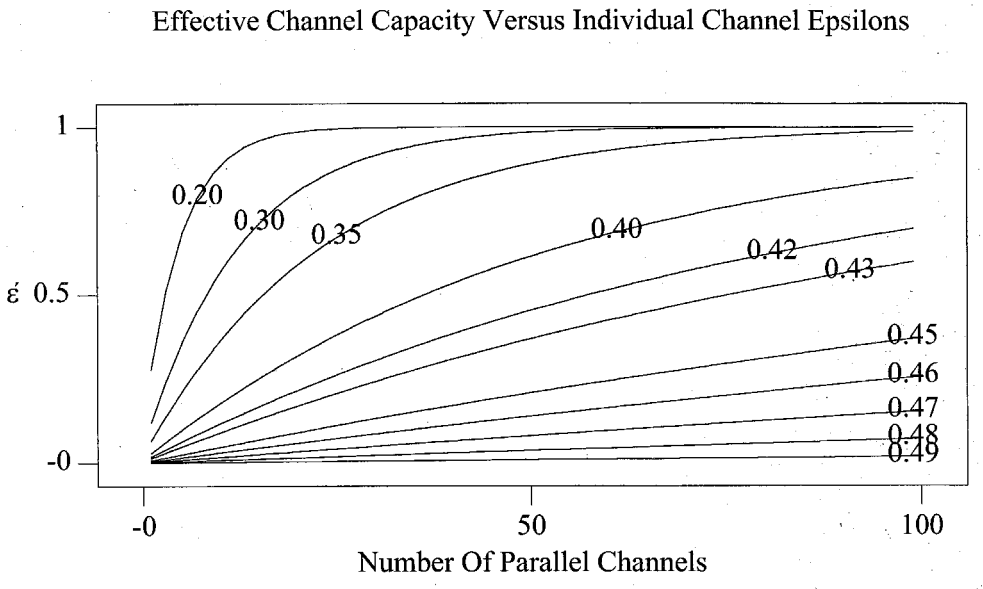


Figure 22. Effective Channel Capacity Versus Single Channel Epsilon

As can be seen from the plots, a large group of neurons with individually poor P_{error} can together achieve as an ensemble a low system P_{error}. This is in keeping with the philosophy of this thesis that large ensembles of statistically independent, marginally performing subsystems, can yield excellent overall system results.

7. Hard Versus Soft Decision Decoding

As discussed above, improvement in overall system performance, denoted by ε, is significant for large numbers of hidden layer neurons. The above improvement is achieved using hard decision decoding. Majority voting implements a hard decision decoding rule. No information about how strongly each neuron "feels" about the decision it is making is passed on to the

voting neuron. This could lead to situations such as that sketched below.

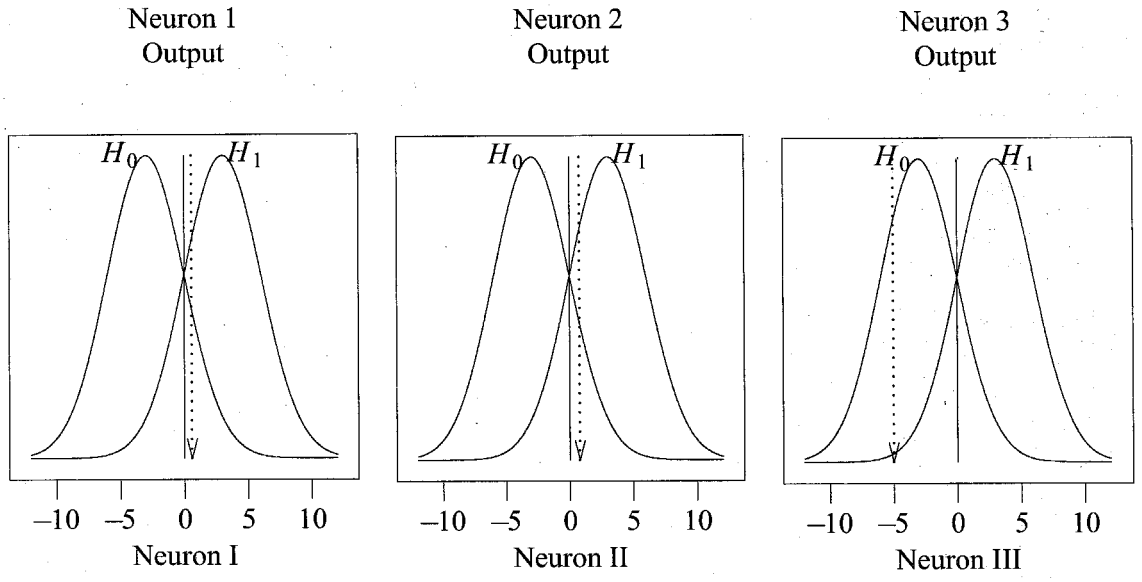


Figure 23. Three Neurons And Hard Decision Misidentification

In the dilemma above, all three neurons have symmetrical conditional probability densities about zero. Hence, the hard decision threshold for each neuron is zero. Two decisions, indicated by dotted arrows, barely fall over the threshold into the H_1 regime. The third neuron decision falls strongly in the H_0 regime. Hard decoding yields a system decision of H_1, whereas our intuition and Bayes analysis indicates H_0 is more likely to have occurred. This hard decision type of error is avoided if each neuron outputs a real number indicating the probability of hypothesis H_0 versus H_1. Ideally, the output of each neuron should be the two element vector below.

$$\left\{ \begin{array}{c} P[H_0|\mathbf{X}] \\ P[H_1|\mathbf{X}] \end{array} \right\}$$

Equation 16. Soft Decision Neuron Outputs

The soft decision is arrived at by summing the individual hypothesis probabilities, and choosing the most probable hypothesis. This is decision decoding in the soft sense.

$$\sum_{Neurons} P_i[H_1|\mathbf{X}] \underset{H_0}{\overset{H_1}{\gtrless}} \sum_{Neurons} P_i[H_0|\mathbf{X}]$$

Equation 17. Soft Decision Rule

A quick glance at the hard decision dilemma above will show soft decoding resolves the issue of two ambivalent outputs which have fallen on the wrong side of the threshold, and one neuron strongly indicating a hypothesis. For the example above, the projection probabilities onto each neuron is tabulated below. These projections were computed from the example conditional densities $P[H_0|\mathbf{X}] \equiv \eta(\nu=-3,\sigma^2=3)$, and $P[H_1|\mathbf{X}] \equiv \eta(\nu=+3,\sigma^2=3)$,

Neuron	Projection	Hypothesis 0	Hypothesis 1	Hard Decision
One	0.6	0.065	0.097	Decide H_1
Two	0.8	0.059	0.102	Decide H_1
Three	-5.0	0.106	0.004	Decide H_0
Hard		1 Vote	2 Votes	Decide H_1
Soft		0.230	0.203	Decide H_0

TABLE 2. Projection Statistics

The soft decision rule is the sum seen above. This is the Maximum-Likelihood receiver (ML). Since in our example, we have assumed equal a priori probabilities for the two hypotheses, this decision structure is also the Maximum-A-Posteriori receiver (MAP). For non-equal a priori probabilities, these two decision methods are *not* equal. The difference between MAP and ML lies in how the a priori probabilities $P[H_0]$ and $P[H_1]$ are used. In MAP, the a priori probabilities are actively used in the decision mechanism. In ML, the a priori probabilities are assumed to be equal. These considerations lead to different thresholds used to determine which hypothesis is chosen. The basic form of a threshold decision rule however is the same for MAP and ML. Consider y_i to be the output of neuron i's projection of input data onto the weight vector $\mathbf{W}_i$. The ML and MAP decision test are shown below

$$\frac{P[H_1|y]}{P[H_0|y]} \underset{H_0}{\overset{H_1}{\gtrless}} 1 \qquad\qquad \frac{P[H_1|y]}{P[H_0|y]} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P[H_0]}{P[H_1]}$$

Equation 18. ML Versus MAP

For MAP, variation of the a priori hypothesis probabilities $P[H_0]$ and $P[H_1]$ results in changes to the optimum threshold. In general, knowledge of the a priori probabilities will allow a more efficient threshold test to be constructed, and result in an overall lower P_{error}. In what follows, we shall assume equal a priori hypotheses to simplify the conceptual presentation. However, it should be kept in mind that unequal a priori probabilities can easily be incorporated into the hypothesis test by altering the threshold.

The analysis above is a departure from the *traditional* treatment of sigmoids in the literature. Typically, all network neurons are embodied with an identical nonlinear function, usually of the form Tanh[γ ($\mathbf{W} \bullet \mathbf{X} - \theta$)]. The γ parameter is called a gain, and the θ parameter is called the threshold or offset. These two parameters are often chosen to optimize criteria such as minimization of the time needed to reach the onset of generalization for a training set. In this work, a typically *unique* sigmoid is determined for each and every neuron in the network. The criteria motivating inclusion of sigmoids is an increase in system error performance. A side result *may* be an increase in the speed with which generalization is achieved. However, this is a secondary consideration here. Thus, each neuron sigmoid is unique. There does not typically exist a global Hyperbolic Tangent Gain γ and offset θ that is suitable for *all* neurons.

The optimum soft decision sigmoid can be determined exactly from the first two moments of the two hypotheses pdf's. This is local information, and is available to the neuron if it has been compiling statistics during the training period. There are statistically efficient unbiased estimators for all of the parameters which each neuron needs to optimize it's sigmoid, which meet the Cramer-Rao bound for optimality. These Cramer-Rao optimum statistical estimators will be further discussed in the implementations section below.

8. Implementing A Soft Decision Rule

Above, symbolically, each neuron outputs a two element vector. Ideally, each neuron should output a single real number. Suppose each neuron output's the quantity $P[H_1|y_j]-P[H_0|y_j]$. The final layer voting neuron would sum over all previous layer outputs to arrive at the system

soft decision.

$$\sum_{j=1}^{\#Neurons\ InTheLayer} P[H_1|y_j] - P[H_0|y_j] \underset{H_0}{\overset{H_1}{\gtrless}} 0$$

Equation 19. Soft Decision Rule

The issue arises as to whether we can obtain and output the probabilities $P[H_i|y_j]$ for each neuron j and each hypothesis i. Looking at the figures above where the two Gaussians overlap, we see that the projection y uniquely determines a point on each conditional pdf $P[H_0|y_j]$ and $P[H_1|y_j]$. If the neuron has compiled a history (perhaps through labeled sample training to be discussed below), so that internal to each neuron is known the mean and standard deviation for each hypothesis, μ_0, μ_1, σ_0, and σ_1, then given the projection value y, we can construct the two probabilities $P[H_1|y_j]$ and $P[H_0|y_j]$ internal to the neuron. Taking these two conditional pdf's difference, we can output a real number. This real number has the two dimensional conditional probabilities alluded to above encoded into it. There is no information loss in this mapping of the two-dimensional data $P[H_1|y]$ and $P[H_0]$ into the one-dimensional number $P[H_1|y] - P[H_0]$. The theoretical reason *why* we can unambiguously map a two-dimensional quantity into a one-dimensional quantity is because we know that one of the hypotheses, either H_0 or H_1 must occur for each trial. This gives us a degree of freedom constraint which makes the 2-D to 1-D map NOT a projection, but a bijective (one to one and onto) function. Let us call the desired neuron output quantity $z_j = P[H_1|y_j] - P[H_0|y_j]$. We use Bayes rule to determine how exactly to obtain $P[H_1|y_j] - P[H_0|y_j]$ from the projection

onto the j'th neuron, $y_j = \mathbf{W}_j \bullet \mathbf{X}$.

$$P[\, H_1 \mid y_j \,] - P[\, H_0 \mid y_j \,] = \frac{P[\, H_1 \,, y_j \,]}{P[\, y_j \,]} - \frac{P[\, H_0 \,, y_j \,]}{P[\, y_j \,]}$$

$$= \frac{P[\, y_j \mid H_1 \,]\ P[\, H_1 \,]}{P[\, y_j \,]} - \frac{P[\, y_j \mid H_0 \,]\ P[\, H_0 \,]}{P[\, y_j \,]}$$

Equation 20. Expand The Probability Of A Hypothesis Given A Projection Using Bayes Law

Rewriting $P[y_j]$ as $P[y_j|H_1]P[H_1] + P[y_j|H_0]P[H_0]$, we obtain the desired function mapping y_j to z_j.

$$\frac{P[y_j|H_1]\ P[H_1] - P[y_j|H_0]\ P[H_0]}{P[y_j|H_1]P[H_1] + P[y_j|H_0]P[H_0]}$$

Equation 21. Applying Bayes Law Again, Obtain The Optimum Sigmoid For A Neuron

Under the equal a priori hypothesis assumption, $P[H_0] \equiv P[H_1]$. Thus, the above ratio reduces to that below.

$$z_j \equiv f(y_j) \equiv \frac{P[y_i|H_1] - P[y_j|H_0]}{P[y_j|H_1] + P[y_j|H_0]}$$

Equation 22. Simplify The Sigmoid By Assuming Equal A Priori Probabilities

Recall the central limit theorem argument we made above. If both the a priori hypothesis probabilities are equal, *and* the variances of both Gaussians, $y|H_0$ and $y|H_1$ are equal, the

hyperbolic Tanh function emerges. Let $\hat{y}$ denote the normalized ratio $\frac{y}{\sigma}$. Expanding the conditional pdf's in the equation above, we obtain the following.

$$\frac{P[y_i|H_1] - P[y_i|H_0]}{P[y_i|H_1] + P[y_i|H_0]} = \frac{e^{-\frac{(y-\mu_1)^2}{2\sigma^2}} - e^{-\frac{(y-\mu_0)^2}{2\sigma^2}}}{e^{-\frac{(y-\mu_1)^2}{2\sigma^2}} + e^{-\frac{(y-\mu_0)^2}{2\sigma^2}}}$$

Equation 23. Expand The Gaussian Conditional Probability Density Functions

$$\hat{y} = \frac{y}{\sigma} \quad \hat{\mu}_0 = \frac{\mu_0}{\sigma} \quad \hat{\mu}_1 = \frac{\mu_1}{\sigma} \quad \alpha = \frac{\hat{y} - \hat{\mu}_1}{\sqrt{2}} \quad \beta = \frac{\hat{y} - \hat{\mu}_0}{\sqrt{2}}$$

Equation 24. Make Substitutions To Normalize Variable With Respect To σ

$$\frac{e^{-\alpha^2} - e^{-\beta^2}}{e^{-\alpha^2} + e^{-\beta^2}} \quad \frac{e^{\frac{\alpha^2+\beta^2}{2}}}{e^{\frac{\alpha^2+\beta^2}{2}}} = \frac{e^{\frac{\beta^2-\alpha^2}{2}} - e^{-\frac{\beta^2-\alpha^2}{2}}}{e^{\frac{\beta^2-\alpha^2}{2}} + e^{-\frac{\beta^2-\alpha^2}{2}}} = Tanh\left[\frac{\beta^2-\alpha^2}{2}\right]$$

Equation 25. Substitute In For α and β

$$\frac{\beta^2-\alpha^2}{2} \equiv \frac{(y^2-2\mu_0 y+\mu_0{}^2)-(y^2-2\mu_1 y+\mu_1{}^2)}{4\,\sigma^2} \equiv \frac{2y(\mu_1-\mu_0)-(\mu_1{}^2-\mu_0{}^2)}{4\,\sigma^2}$$

Equation 26. Expand The Squares, Cancel Common Terms, And Simplify

$$\equiv \frac{\left[2y - (\mu_1 + \mu_0)\right]\left[\mu_1 - \mu_0\right]}{4\,\sigma^2} \equiv \frac{(y - \frac{(\mu_0+\mu_1)}{2}) \quad (\mu_1-\mu_0)}{2\sigma^2} \equiv$$

Equation 27. Rewrite The Tanh Argument As A Slope and Offset

$$\left[y - \frac{(\mu_1 + \mu_0)}{2} \right] \left[\frac{(\mu_1 - \mu_0)}{2\,\sigma^2} \right] \quad , \quad \gamma \equiv \frac{\mu_1 - \mu_0}{2\ \sigma^2} \qquad \theta \equiv \frac{\mu_1 + \mu_0}{2}$$

Equation 28. Define The Slope and Offset Constants

$$z_j \equiv P[H_1|y_j] - P[H_0|y_j] = Tanh[\ \gamma\ (\ y_j - \theta\ )\ ] \equiv f(y_j)$$

Equation 29. Express The Equal Variance Neuron Sigmoid In Final Form

As a brief check on our calculations, note that for σ^2 equal to 1, and $\mu_0 \equiv -1$, and $\mu_1 \equiv 1$, we have $\gamma \equiv \frac{(\mu_1 - \mu_0)}{2\sigma^2} \equiv 1$ and the threshold $\theta \equiv \frac{\mu_1 + \mu_0}{2}$ is zero. By inspection of the equations above, the expression for f(y) is $\frac{e^y - e^{-y}}{e^y + e^{-y}} \equiv Tanh(y)$. This is the same as the $\gamma \equiv 1, \theta \equiv 0$ result.

To recap, for the special case of equal a priori probability of occurrence for both hypotheses, and equal variance for projections under both hypotheses onto a neurons weight vector, the nonlinear sigmoid $Tanh(\ \gamma\ (\ y_j - \theta\ )\ )$ will map individual neuron projections y_j into a probabilistic quantity that a voting neuron can use to implement a soft decoding scheme. As a result, the system performance changes from a hard decision decoding method to an improved soft decision decoding scheme when one uses the nonlinear sigmoid $Tanh(\ \gamma\ (\ (\ \mathbf{W}_j \bullet \mathbf{X}\ ) - \theta\ )$, and the gain γ and threshold θ parameters as defined above. These two parameters are calculated from the first two moments of the two hypotheses statistical distributions. Below is shown a Tanh sigmoid for two conditional pdfs. The

distributions $P[y|H_1]$ and $P[y|H_0]$ are both Gaussian, with $\mu_1 = 1$, $\mu_0 = -1$, and $\sigma_1 = \sigma_0 = 1$. Thus, $\gamma \equiv \frac{(1 + 1)}{2} \equiv 1$ and θ is zero. The fact that $\theta \equiv 0$ should be expected from the symmetry of the conditional probability density functions.

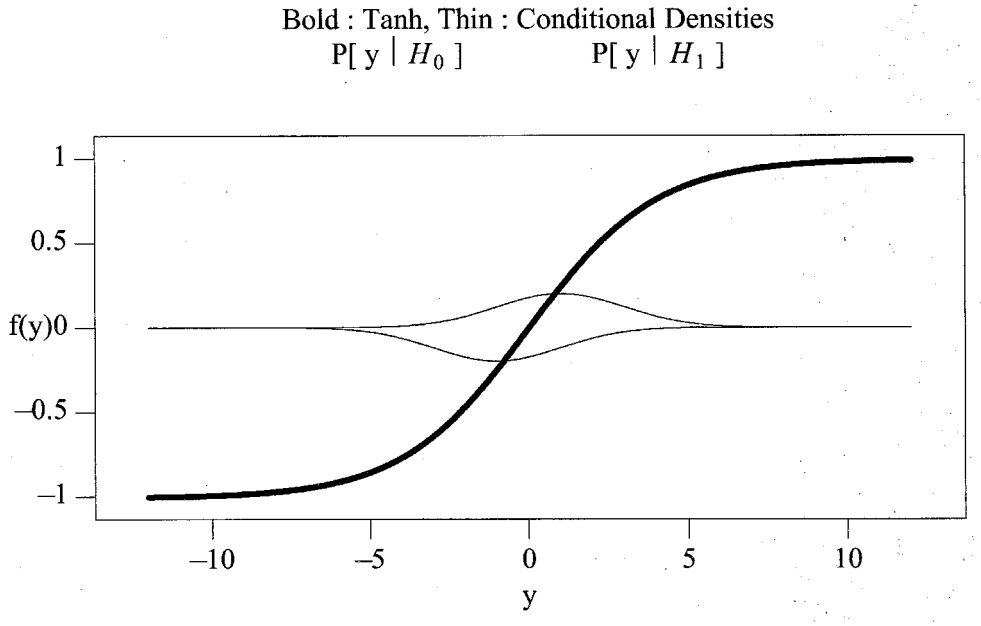


Figure 24. Sigmoid Shape For $P[y|H_1] \equiv \eta(+1,1)$ And $P[y|H_0] \equiv \eta(-1,1)$

9. Multiple Neurons In A Single Layer

9.1 Example With *Six* And *Seven* Images

In this section, we look at the actual performance of five neurons working in parallel on the *Six* and *Seven* image set described above. The five top eigenvectors for the *Six* and *Seven* images was calculated, and theoretical differences used to construct five weight vectors, $\mathbf{W}_1$ through

$\mathbf{W}_5$.

The results are plotted below. The solid histograms and curves are for hypothesis *Six*. The dotted curves are for hypothesis *Seven*. In neurons five through two, the decision threshold was such that hypothesis **7** was less than the threshold and hypothesis **6** was greater than the threshold. For the last neuron shown, neuron 1, the opposite sign was taken. Thus, for neuron one only, the brute force Bit Error Rate (BER) curve is for hypothesis **6** less than the threshold and hypothesis **7** greater than the threshold.

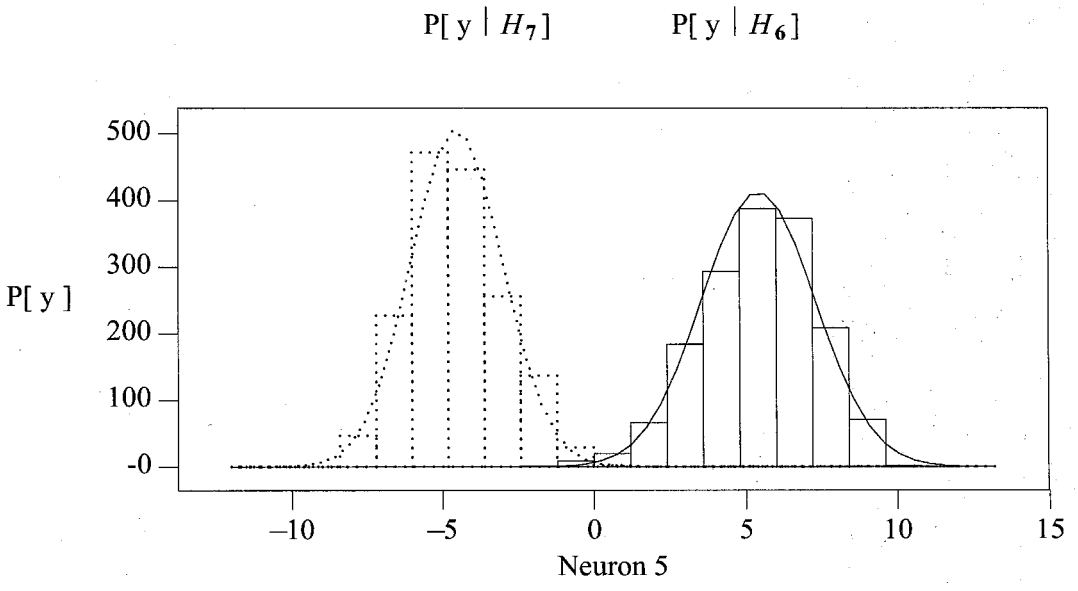


Figure 25. Neuron 5 For The Set Of Five Parallel Neurons

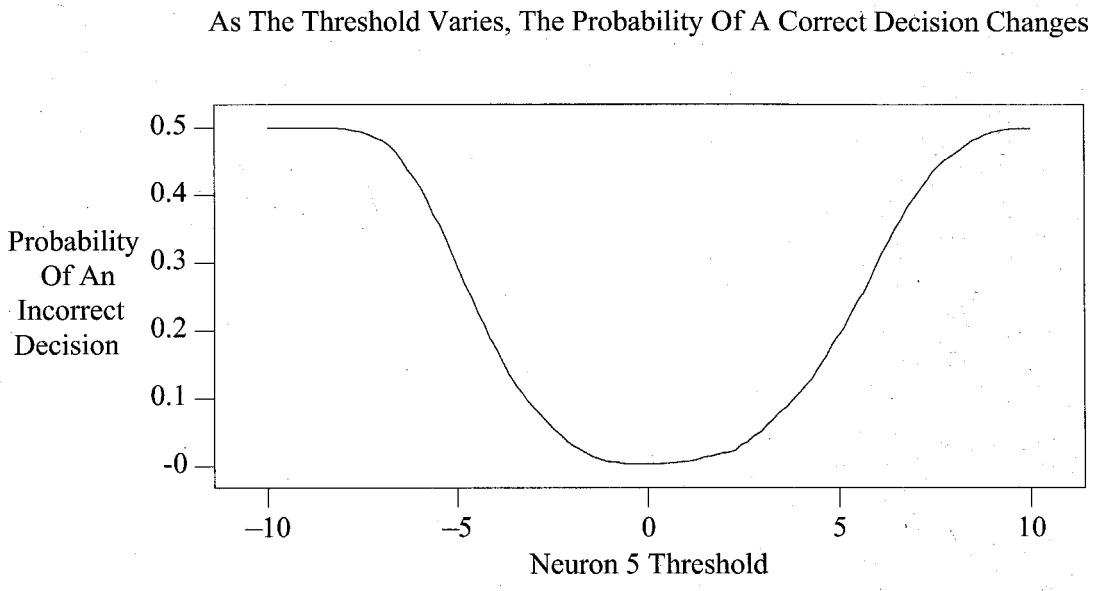


Figure 26. Estimation Decision Error Via Brute Force Variation Of The Neuron Threshold

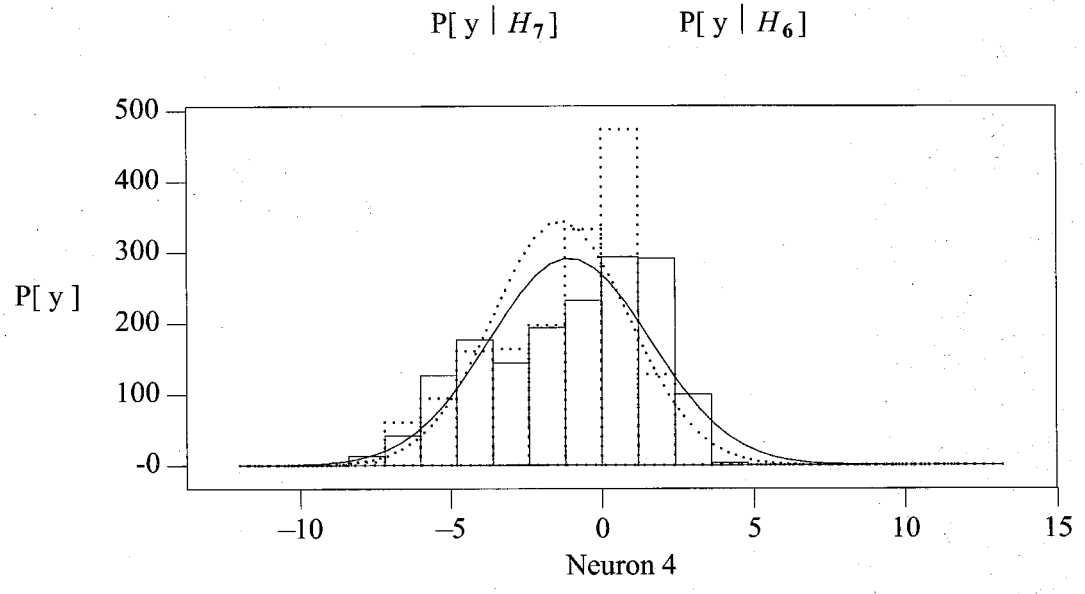


Figure 27. Neuron 4 For The Set Of Five Parallel Neurons

As The Threshold Varies, The Probability Of A Correct Decision Changes

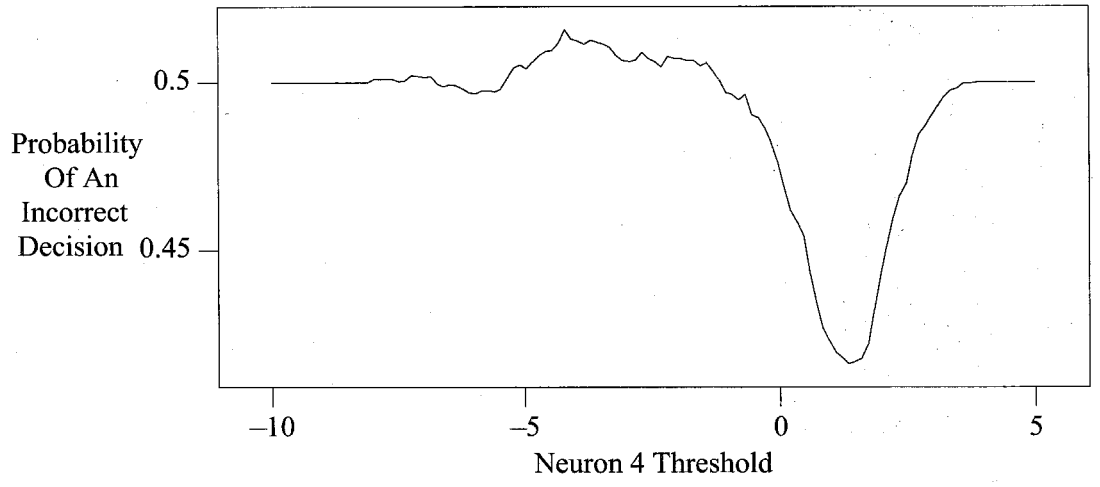


Figure 28. Estimation Decision Error Via Brute Force Variation Of The Neuron Threshold

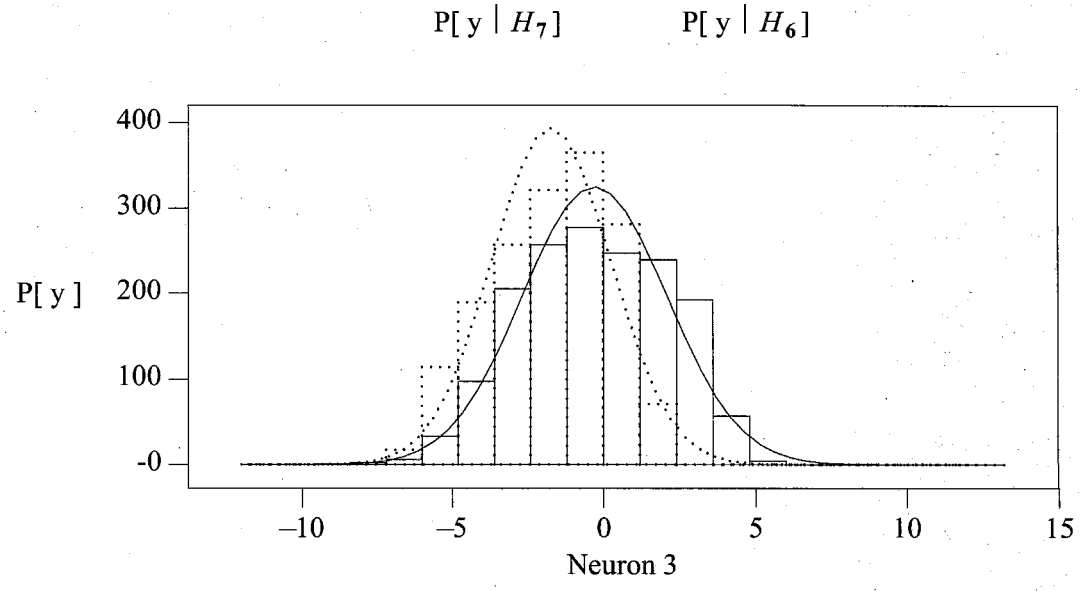


Figure 29. Neuron 3 For The Set Of Five Parallel Neurons

As The Threshold Varies, The Probability Of A Correct Decision Changes

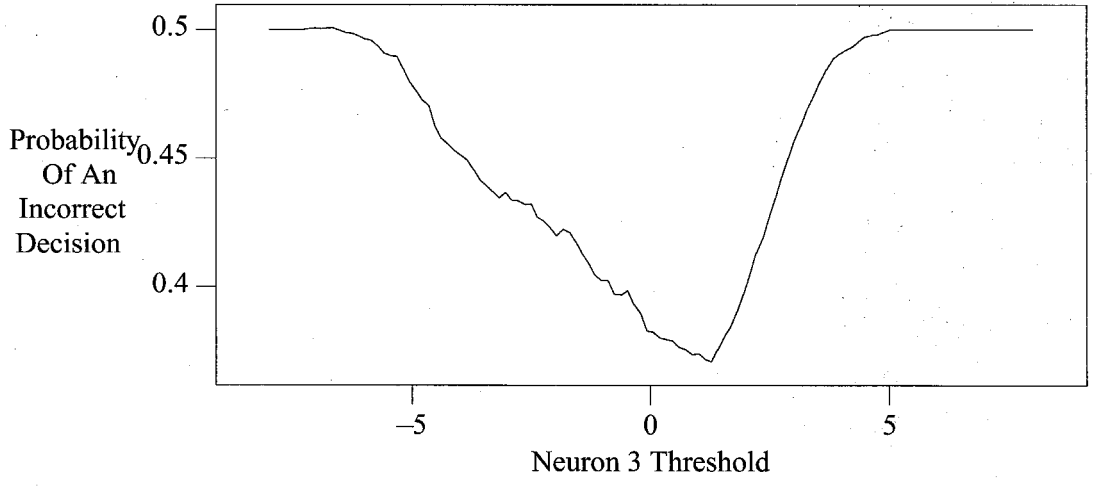


Figure 30. Estimation Decision Error Via Brute Force Variation Of The Neuron Threshold

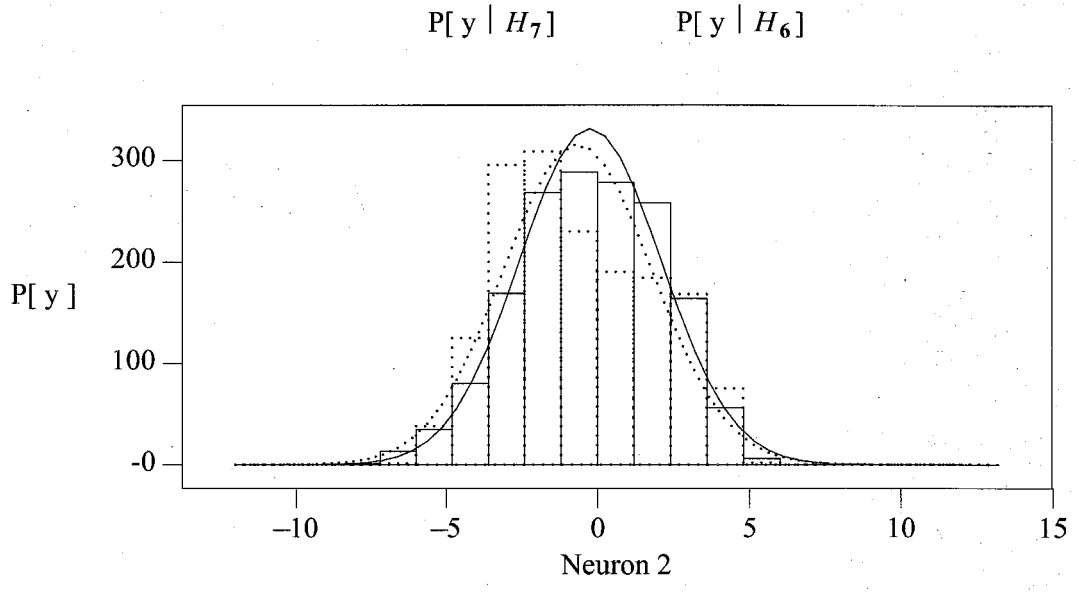


Figure 31. Neuron 2 For The Set Of Five Parallel Neurons

As The Threshold Varies, The Probability Of A Correct Decision Changes

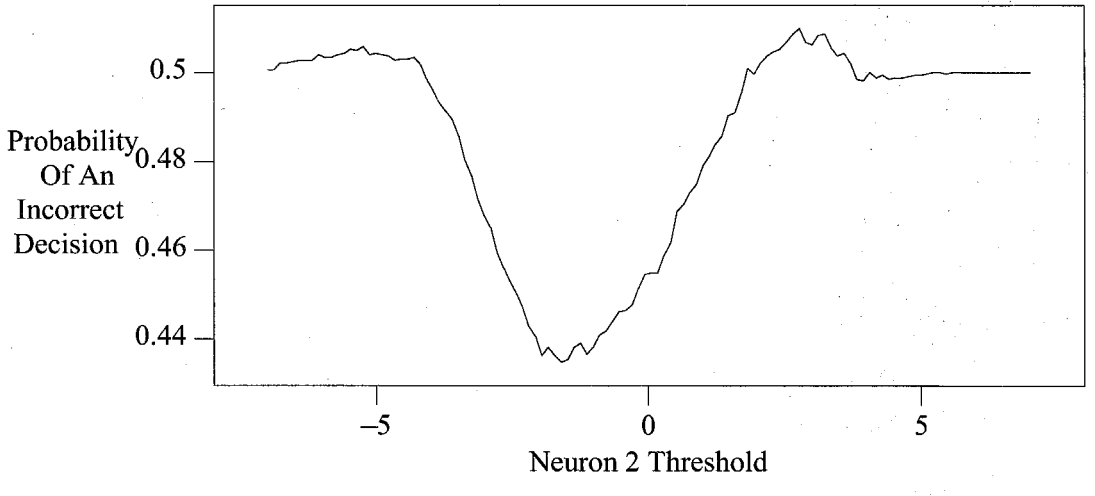


Figure 32. Estimation Decision Error Via Brute Force Variation Of The Neuron Threshold

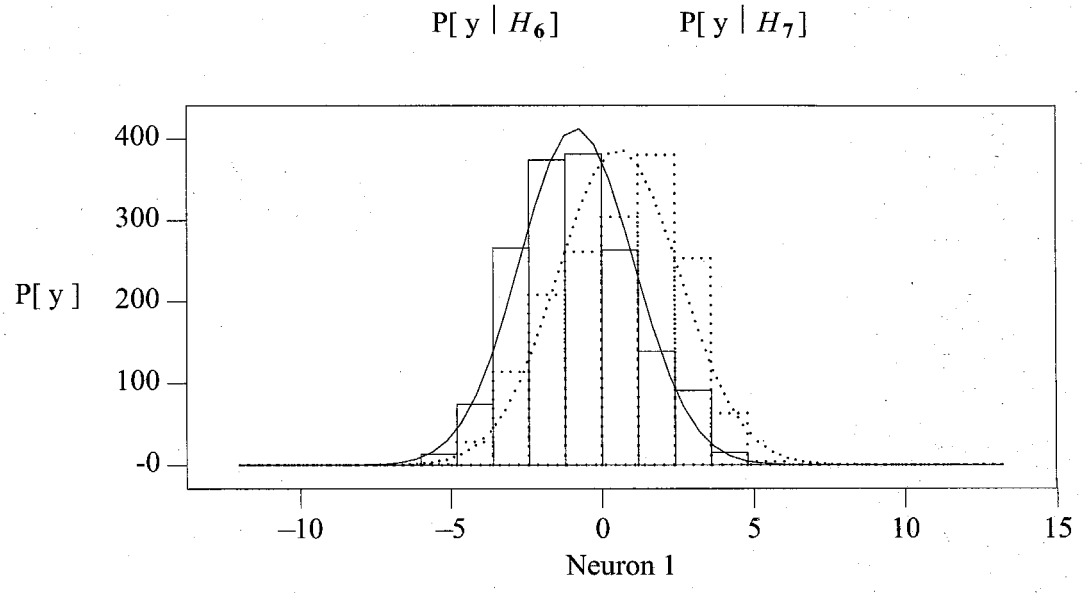


Figure 33. Neuron 1 For The Set Of Five Parallel Neurons

As The Threshold Varies, The Probability Of A Correct Decision Changes

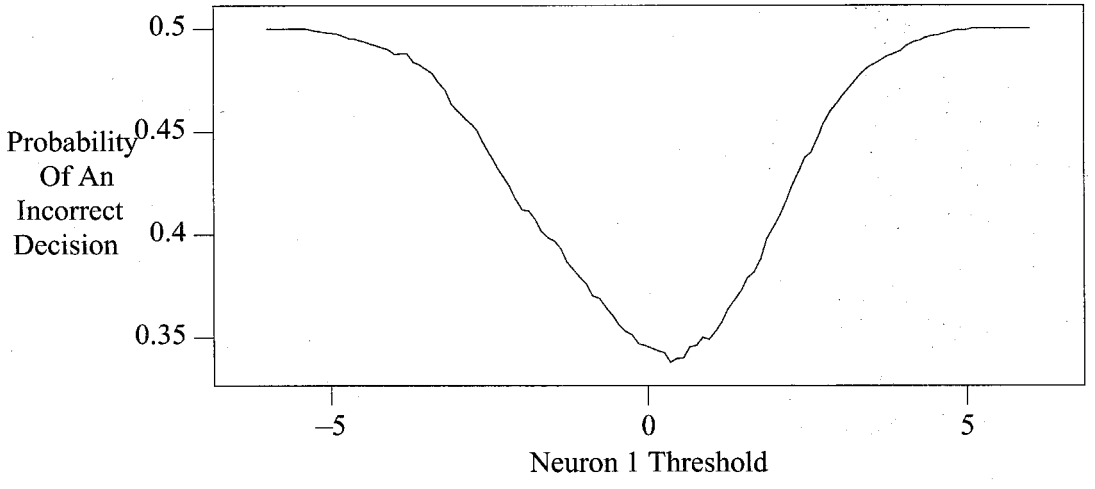


Figure 34. Estimation Decision Error Via Brute Force Variation Of The Neuron Threshold

Neuron	Weight Vector	Digit **6** Mean	Digit **6** Variance	Digit **7** Mean	Digit **7** Variance	BER Parameters Threshold	BER Parameters Min P_e
5	$W_6 - W_7$	5.418	3.52	-4.521	2.346	-0.25	0.0034
4	$W_6 - W_7$	-1.067	7.06	-1.35	5.08	1.34	0.42
3	$W_6 - W_7$	-0.299	5.67	-1.72	3.87	1.28	0.37
2	$W_6 - W_7$	-0.244	5.47	-0.654	6.04	-1.56	0.435
1	$W_6 - W_7$	-0.834	3.52	0.567	4.01	0.35	0.34

TABLE 3. Neuron Projection Statistics

Ideally, it would be nice if perfect decorrelation occurred between all neurons under both hypotheses. Then the Central Limit Theorem could be invoked, and the Gaussian outputs of the five neurons could be assumed statistically independent under both hypotheses. The actual covariance of the neurons under both hypotheses is seen below. The tabulated values are the $\frac{E[s_i s_j]}{\sqrt{E[s_i^2]E[s_j^2]}}$, where s_i is $x_i - \mu_i$. The hypothesis **7** results are above the diagonal. The hypothesis **6** results are below the diagonal.

	Neuron 5	Neuron 4	Neuron 3	Neuron 2	Neuron 1
Neuron 5	1	-0.43513	0.09990	0.03745	-0.099
Neuron 4	-0.005255	1	-0.21411	-0.21810	-0.01400
Neuron 3	0.483652	-0.294616	1	0.4770	-0.09981
Neuron 2	0.473566	-0.069312	0.442220	1	-0.156165
Neuron 1	-0.060548	0.005252	-0.046695	0.040725	1

TABLE 4. Covariance Of The Neuron Projection

A few of the covariances are quite bad. Consider neuron 5 and neuron 3, under the hypothesis an image of a *Six* was seen. Their covariance is 0.48, which is high. However, note the corresponding covariance for the same two neurons under hypothesis **7** is low, only 0.10. This

seems to be the theme for all the high covariances below. Averaging the two hypotheses absolute value covariance's for the neurons yields the tables below.

	Neuron 5	Neuron 4	Neuron 3	Neuron 2	Neuron 1
Neuron 5	1	0.220	0.292	0.256	0.080
Neuron 4	0.220	1	0.254	0.144	0.010
Neuron 3	0.292	0.254	1	0.460	0.073
Neuron 2	0.256	0.144	0.460	1	0.098
Neuron 1	0.080	0.010	0.073	0.098	1

TABLE 5. Covariance Of The Neuron Projection

Most covariances are good. Only the neuron 2 - 3 combination is still high.

9.2 Example With *Three* And *Five* Images

In this section, we look at the actual performance of five neurons working in parallel on a *Three* and *Five* digit image set. The five top eigenvectors for the *Three* and *Five* images were calculated, and eigenvector differences used to construct five weight vectors, $\mathbf{W}_1$ through $\mathbf{W}_5$. The results are shown below. The solid histograms and curves are for hypothesis *Three*. The dotted curves are for hypothesis *Five*. There was a total of 1459 *Three* images and the same number of *Five* images used in the analysis below for each neuron. The vertical line topped by a small bullet denotes the brute force optimum threshold equal to the minimum of the **6** & **7** digit image BER curves.

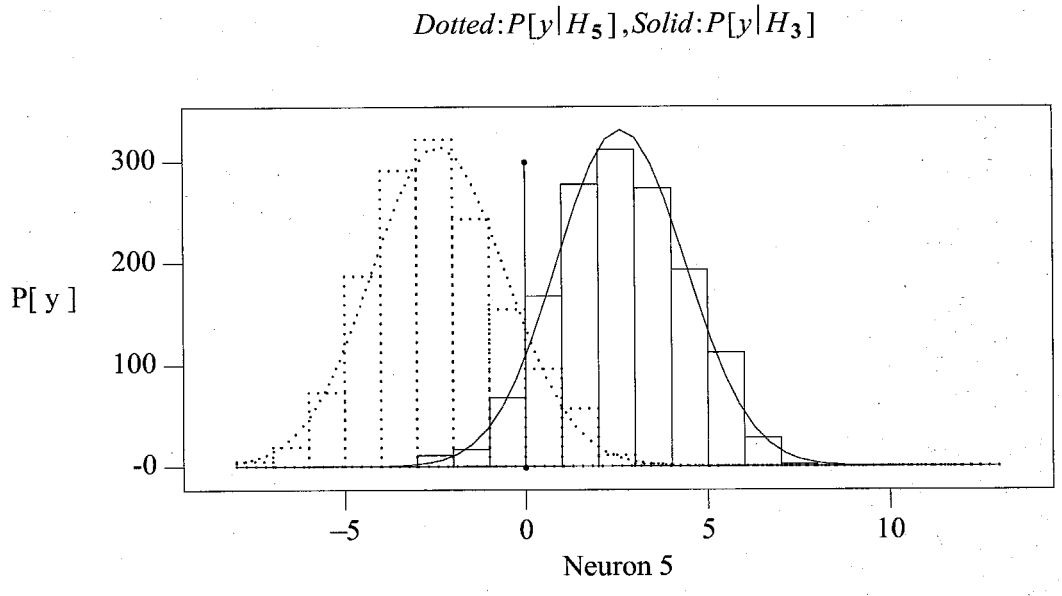


Figure 35. Neuron 5 For The Set Of Five Parallel Neurons For The 3/5 Images

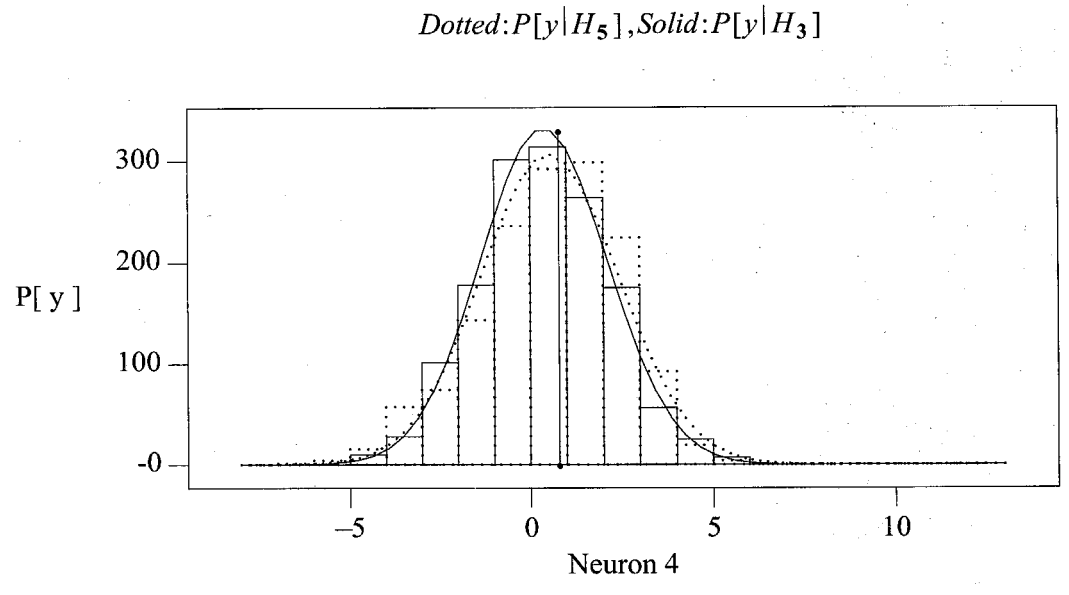


Figure 36. Neuron 4 For The Set Of Five Parallel Neurons For The 3/5 Images

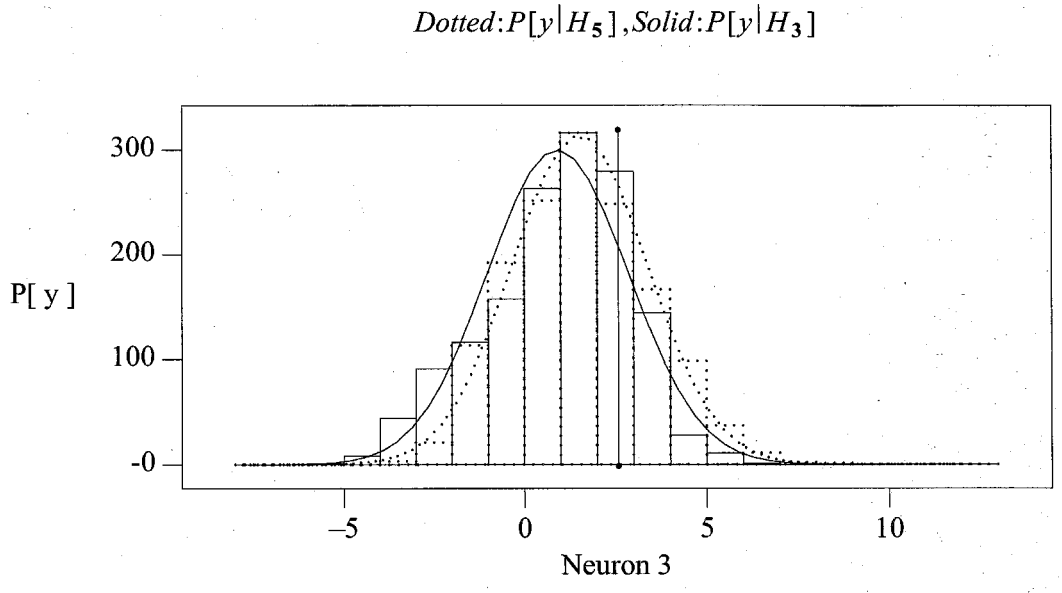


Figure 37. Neuron 3 For The Set Of Five Parallel Neurons For The 3/5 Images

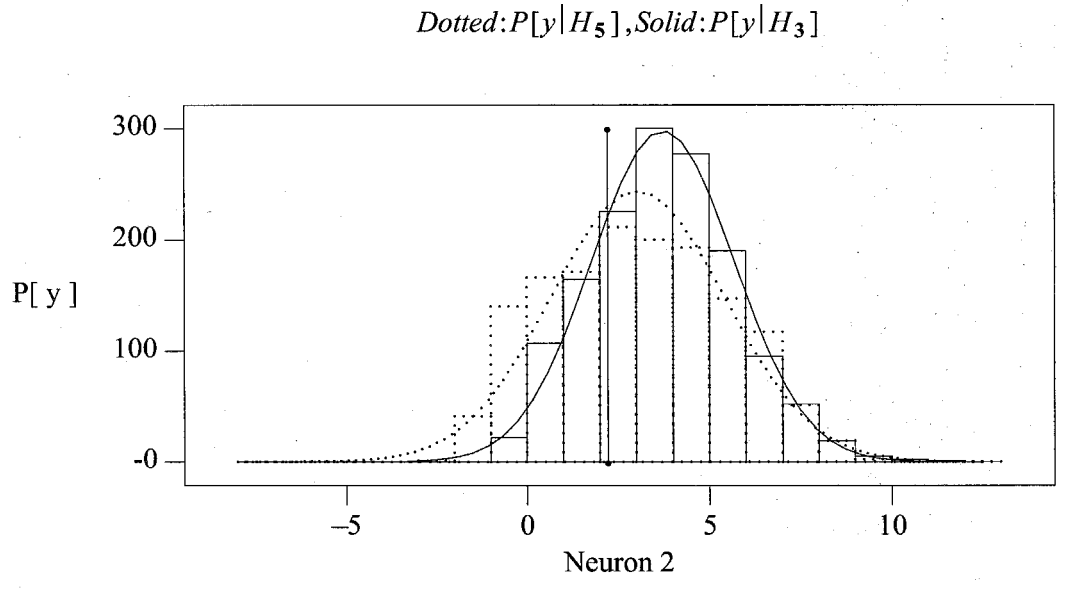


Figure 38. Neuron 2 For The Set Of Five Parallel Neurons For The 3/5 Images

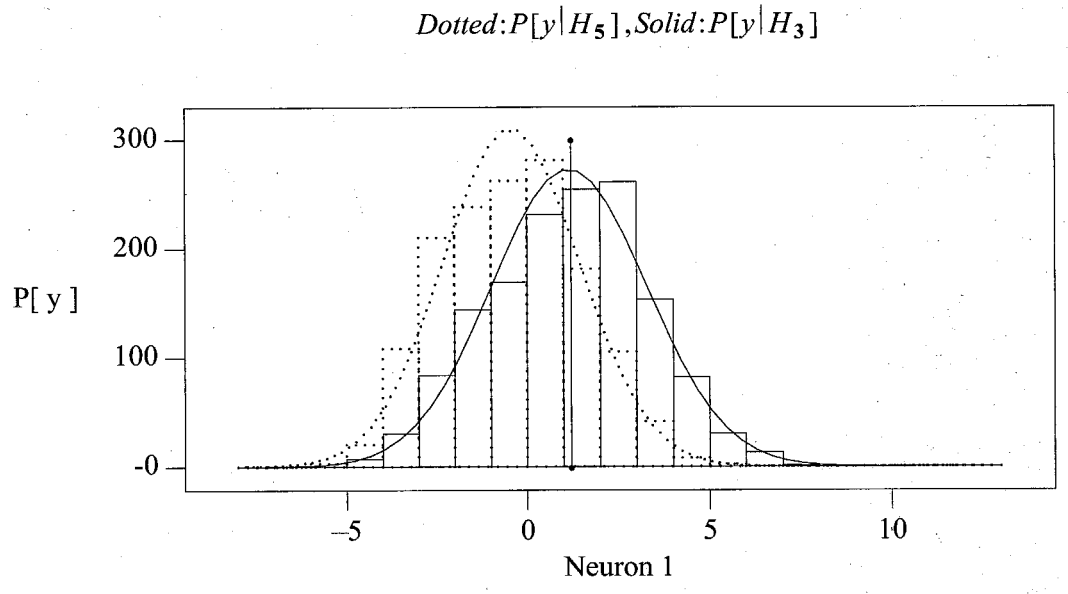


Figure 39. Neuron 1 For The Set Of Five Parallel Neurons For The 3/5 Images

The mean and variances for the two hypotheses are shown below, along with the optimum threshold and the probability of error obtained for that neuron using that threshold.

Neuron	Weight Vector	Digit **5**		Digit **3**		BER Parameters	
		Mean	Variance	Mean	Variance	Threshold	Min P_e
5	$W_3 - W_5$	-2.39	3.44	2.63	3.10	-0.001	0.089
4	$W_3 - W_5$	0.53	3.625	0.37	3.07	0.7895	0.456
3	$W_3 - W_5$	1.53	3.46	0.92	3.78	2.579	0.451
2	$W_3 - W_5$	3.045	5.74	3.73	3.82	2.211	0.417
1	$W_3 - W_5$	-0.44	3.53	1.14	4.57	1.21	0.338

TABLE 6. Neuron Projection Statistics

The covariance of the neurons under both hypotheses is seen below. The tabulated values are the $\frac{E[s_i s_j]}{\sqrt{E[s_i^2]E[s_j^2]}}$, where s_i is $x_i - \mu_i$. The hypothesis **3** results are above the diagonal. The

hypothesis **5** results are below the diagonal.

	Neuron 5	Neuron 4	Neuron 3	Neuron 2	Neuron 1
Neuron 5	1	0.16	0.40	0.68	0.5
Neuron 4	-0.24	1	0.22	0.08	0.40
Neuron 3	-0.55	0.17	1	0.26	0.05
Neuron 2	-0.59	0.01	0.29	1	0.36
Neuron 1	0.30	0.24	-0.55	-0.12	1

TABLE 7. Covariance Of The Neuron Projection

9.3 Soft Decision Numerics With The 3 And 5 Image Set

To illustrate the benefits obtained from soft decision decoding, we shall use neurons 2, 3, and 4 above. We do this for two reasons. First, this subset of three neurons has similar individual P_{Error}, which yields an example closest to the idealized, equal ε discussion above. Second, this subset has minimal correlation between them, under both hypotheses, yielding as close to independent output's as we could get. The theoretical hard decision probability of error for the three equally weighted neurons in a voting scenario is denoted by ε' in the equation below.

$$\varepsilon' \equiv (1 - \varepsilon_2) \varepsilon_3 \varepsilon_4 + \varepsilon_2 (1 - \varepsilon_3) \varepsilon_4 + \varepsilon_2 \varepsilon_3 (1 - \varepsilon_4) + \varepsilon_2 \varepsilon_3 \varepsilon_4$$

Equation 30. Probability Of Neurons 2, 3, And 4 Together Making A Hard Decision Error

The theoretical hard decision result is 0.412, slightly better than the 0.417 performance of neuron 2. The actual hard decision performance obtained by tallying up the three neurons for the two sets of 1459 images under each hypothesis was 0.441, worse than neuron 2 acting alone, yet

better than either neuron 3 or neuron 4's individual performance. The soft decision performance was obtained by equally summing the probability of each hypothesis being true, given the input image **X**.

$$\sum_{i \in (N_2,N_3,N_4)} P_i[H_3|\mathbf{X}] \underset{H_5}{\overset{H_3}{\gtrless}} \sum_{i \in (N_2,N_3,N_4)} P_i[H_5|\mathbf{X}]$$

Equation 31. Soft Decision Rule

Physically, this was obtained by calculating the mean and variance of the projection data from the two sets of images, and constructing the sigmoid function seen below.

$$\frac{P[y_i|H_3] - P[y_i|H_5]}{P[y_i|H_3] + P[y_i|H_5]} \equiv \frac{\frac{e^{-\frac{(y-\mu_3)^2}{2\sigma_3^2}}}{\sigma_3} - \frac{e^{-\frac{(y-\mu_5)^2}{2\sigma_5^2}}}{\sigma_5}}{\frac{e^{-\frac{(y-\mu_3)^2}{2\sigma_3^2}}}{\sigma_3} + \frac{e^{-\frac{(y-\mu_5)^2}{2\sigma_5^2}}}{\sigma_5}}$$

Equation 32. Soft Decision Sigmoid Computed For *Three, Five* Image Example

Each image was passed through the sigmoid, and the three neuron outputs were tallied. If the sum was positive, H_3 was chosen. If the sum was negative, H_5 was chosen. The resulting soft decision error performance was 0.411, which is better than the best neuron (number 2) making a decision alone. A useful approximation to the soft error performance can be constructed as follows. Assuming the Central Limit Theorem holds, the total soft error is upper bounded by

probability tails of the sum of the three neurons, without sigmoids, spilling over a threshold of 0. That is, construct the sum below.

$$Y \equiv \mathbf{W}_2 \bullet \mathbf{X} + \mathbf{W}_3 \bullet \mathbf{X} + \mathbf{W}_4 \bullet \mathbf{X}$$

Equation 33. Soft Error Upper Bound Estimate

The projections are assumed statistically independent, and each projection is Gaussian under the CLT assumption, under both H_3 and H_5. Thus, the random variable Y will also be Gaussian, under both hypotheses. Thus Y given H_3 occurred has mean 2.44, and variance 10.67. Similarly, Y given H_5 occurred has mean 0.985 and variance 12.825. This estimate then becomes the equation below.

$$P_{Soft}^{Estimate} \equiv \frac{Erfc\left[\frac{\mu_3}{\sqrt{2}\ \sigma_3}\right] + Erfc\left[\frac{\mu_5}{\sqrt{2}\ \sigma_5}\right]}{4} \equiv 0.310$$

Equation 34. Soft Error Example Performance Estimate

The soft performance more closely agrees with the ideal hard performance (0.412) than the estimated upper bound for soft performance. Thus, the example indicates the correlation between the neurons does degrade the hard and soft decision performance. Implementation of soft decoding brings the system to approximately where ideal hard performance should be. Many (≈ 25) other runs were made using all three databases, and this example's results were typical of hard versus soft performance behavior. That is, using a soft decision technique usually

yielded performance on par with the theoretical independent neuron, hard decoding P_{error}.

10. Approximating The Tanh Function

In this section, we investigate how well the equal variance Tanh form theoretically works when the variances are unequal. Recall that the sigmoid for equal conditional Gaussian projection pdf's was the following.

$$Tanh\left[\beta \bullet \left[x - T \right] \right] \qquad \beta \equiv \frac{\mu_1 - \mu_0}{2\sigma^2} \qquad T \equiv \frac{\mu_1 + \mu_0}{2}$$

Equation 35. Recapping The Equal Variance Sigmoid

The method of analysis to study how well the sigmoid fits a non equal variance situation is the following. We can always map a pair of random variables with two different means into a pair with means equal to $\pm$ 1 by shifting and scaling the original pair simultaneously via the map $y = m \bullet (x - b)$. No *information* is lost in this map, in that it is a linear map, and hence bijective. However, now the variance of each of the transformed pair is fixed. Thus, by studying the case with means equal to $\pm$ 1, and unequal variances for the two random variables, we should obtain an idea of the robustness of the Tanh approximation. For purposes of comparison, we shall first set the variance ${\sigma_{-1}}^2$ of the random variable with $\mu \equiv -1$ equal to 1, and vary the variance ${\sigma_{+1}}^2$ of the random variable with $\mu \equiv +1$.

First, we track the change in the threshold as the variance is altered. Previously, the threshold was $\frac{\mu_1 - \mu_0}{2}$. Now a second-order approximation to the deviation of the threshold from the

above ideal case is derived using the the plot below, which shows the actual curve which results from using MAPLE to solve for the intersection point of the two random variables.[19] The solid curve is the exact MAPLE result. The dashed and dotted lines are the approximations to follow.

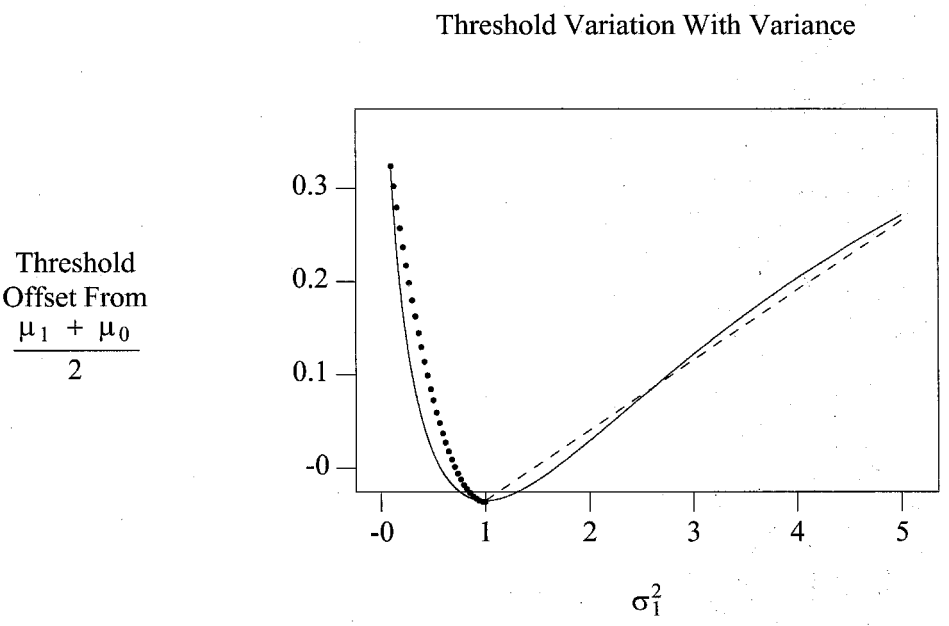


Figure 40. Threshold Variation With Variance

A second-order approximation to the deviation of the threshold from the above is 0.075 ($\sigma_1{}^2$ - 1), where the subscript 1 means the variance at $\mu = 1$, and the variance at $\mu = -1$ is assumed $\equiv 1$.

Using this second-order threshold offset correction, four functions were studied for their goodness of fit. These were Minimum Variance, Maximum Variance, Arithmetic Average, and Geometric Average, of the two variances, when their means have been mapped to ± 1. The best fit was achieved when the geometric average was used. Thus, only this fit curve will be shown

below. The exact equations to be used for determining the Tanh parameters β and θ are therefore the following, for $\sigma_{-1} \equiv 1$.

$$\theta \equiv \frac{\mu_1 + \mu_0}{2} + 0.075 \left[\left(\frac{\sigma_1}{\sigma_{-1}} \right)^2 - 1 \right] \quad For\ \sigma_1 > 1$$

$$\theta \equiv \frac{\mu_1 + \mu_0}{2} + 0.444 \left[\left(\frac{\sigma_1}{\sigma_{-1}} \right)^2 - 1 \right]^2 \quad For\ \sigma_1 < 1$$

$$\forall\, \sigma_1 \rightarrow \gamma \equiv \frac{1}{\sigma_{-1}\ \sigma_1}$$

Equation 36. Second Order Tanh Approximation

Plots for various σ_1 are seen below. The bold curves are the Tanh approximations. The Gaussians are shown, as well as the exact equal probable a priori expression (i.e., the difference of the conditional Gaussians over the sum of the conditional Gaussians).

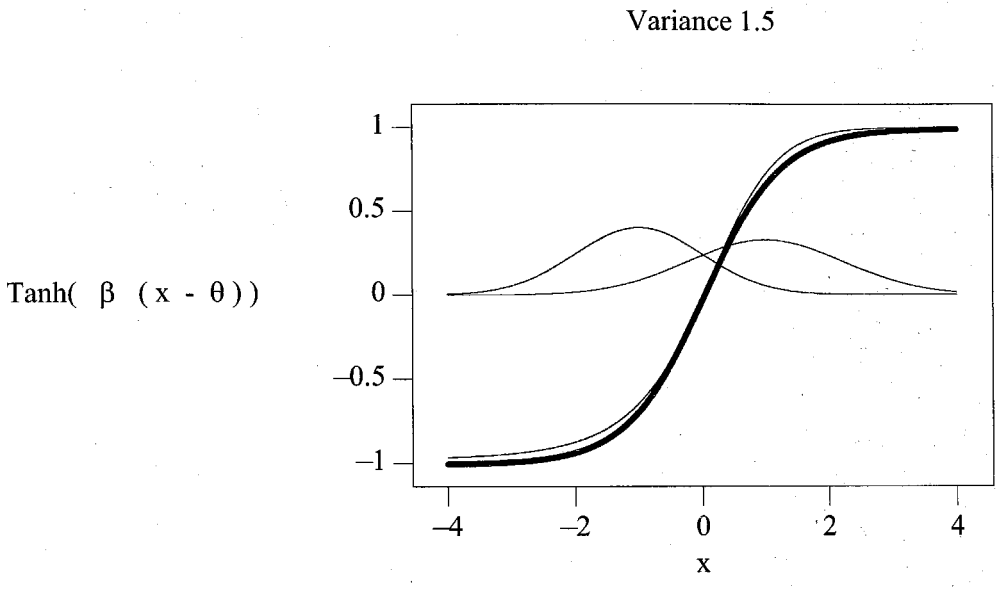


Figure 41. Left-Hand Side Variance 1.0, Right-Hand Side Variance 1.5

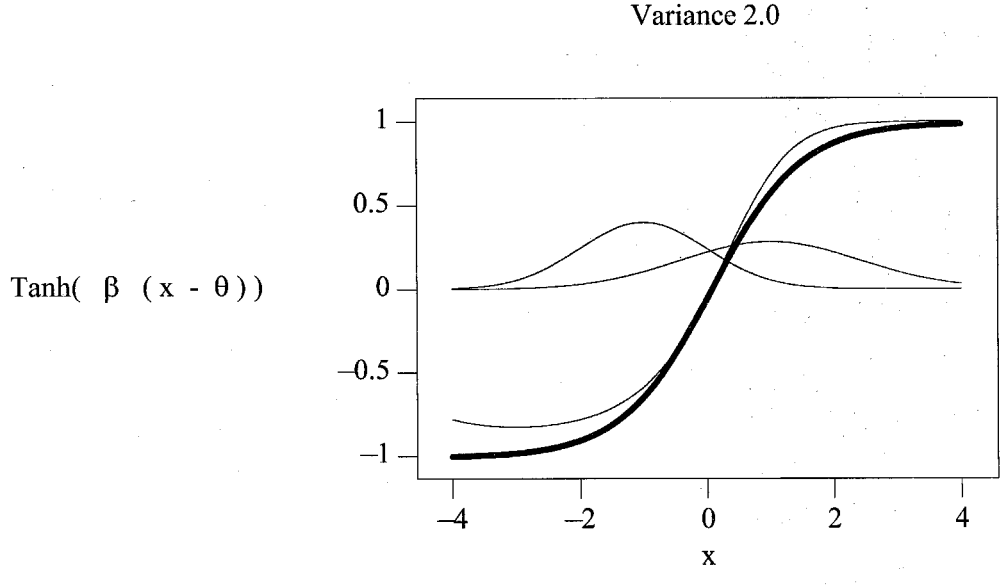


Figure 42. Left-Hand Side Variance 1.0, Right-Hand Side Variance 2.0

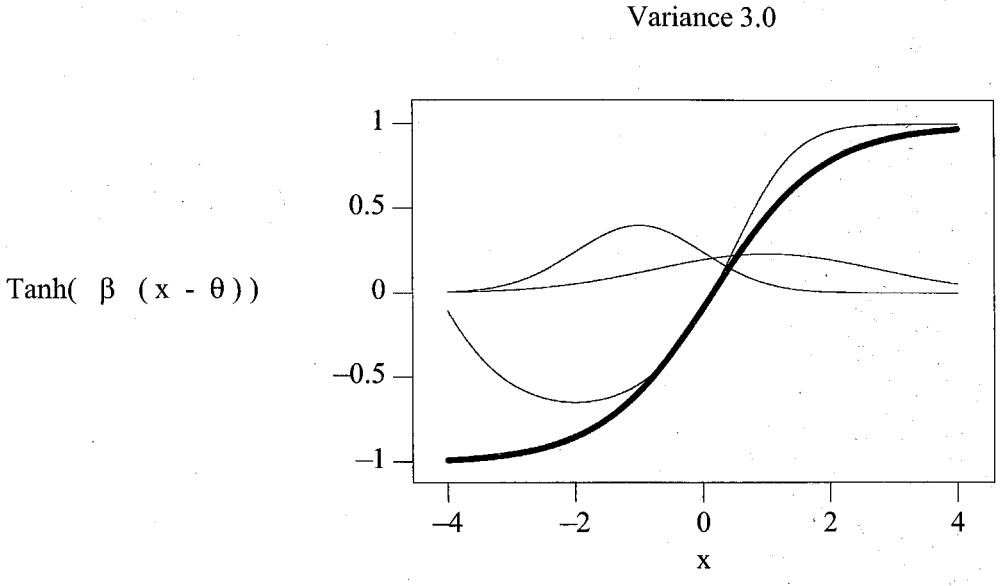


Figure 43. Left-Hand Side Variance 1.0, Right-Hand Side Variance 3.0

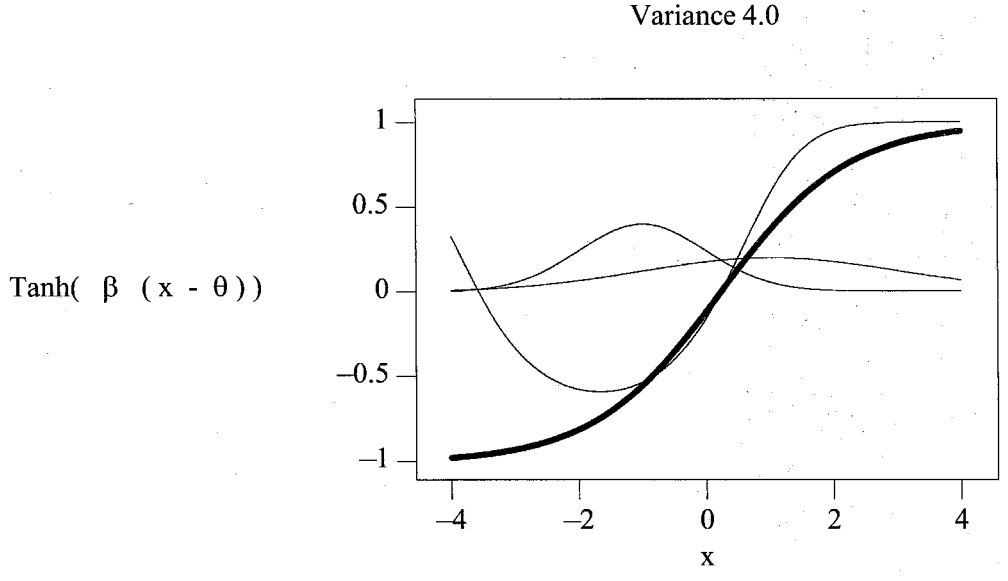


Figure 44. Left-Hand Side Variance 1.0, Right-Hand Side Variance 4.0

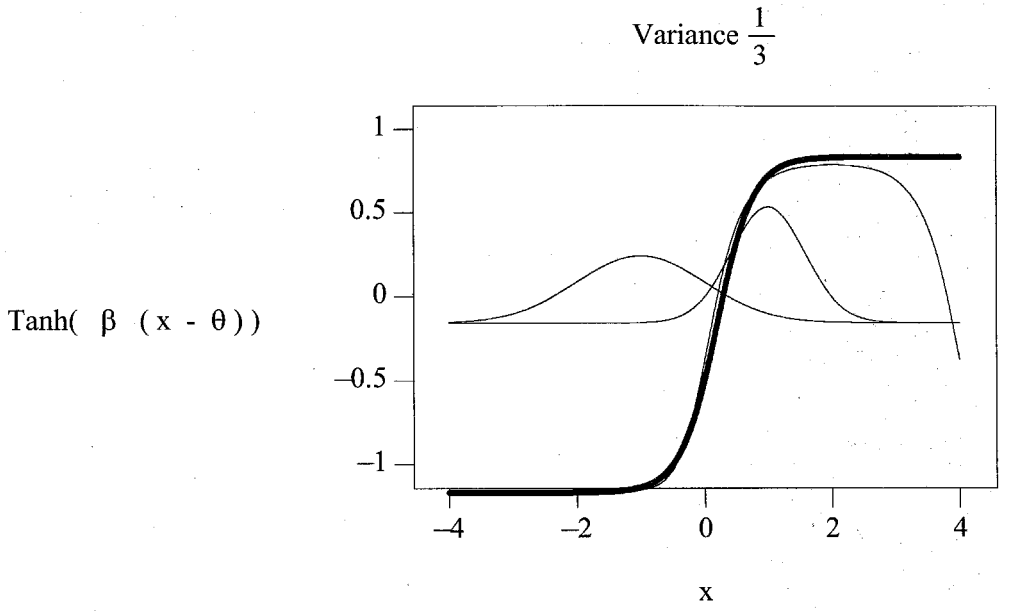


Figure 45. Left-Hand Side Variance 1.0, Right-Hand Side Variance $\frac{1}{3}$

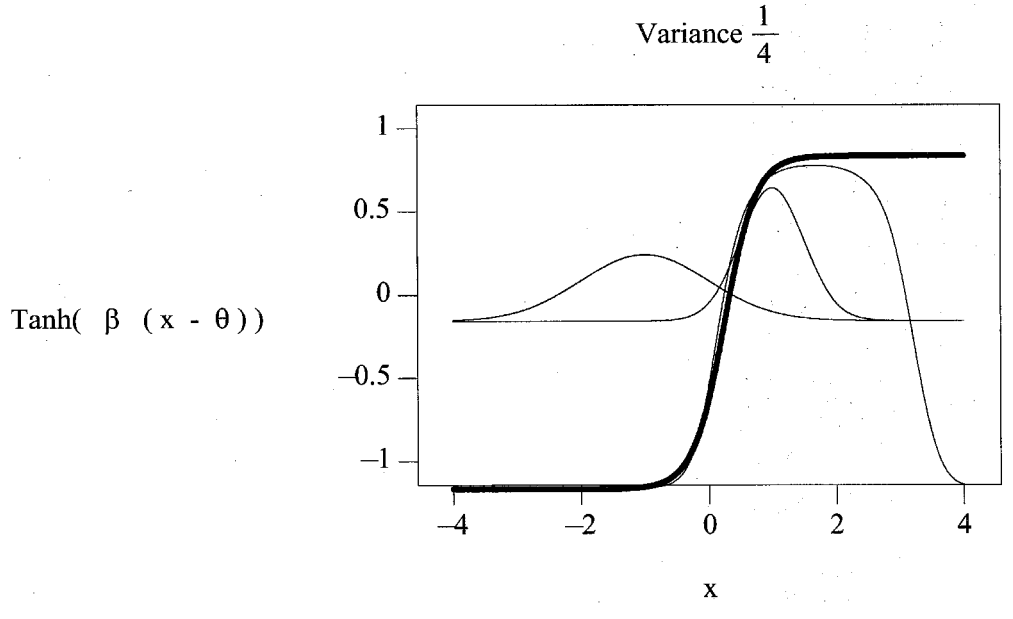


Figure 46. Left-Hand Side Variance 1.0, Right-Hand Side Variance $\frac{1}{4}$

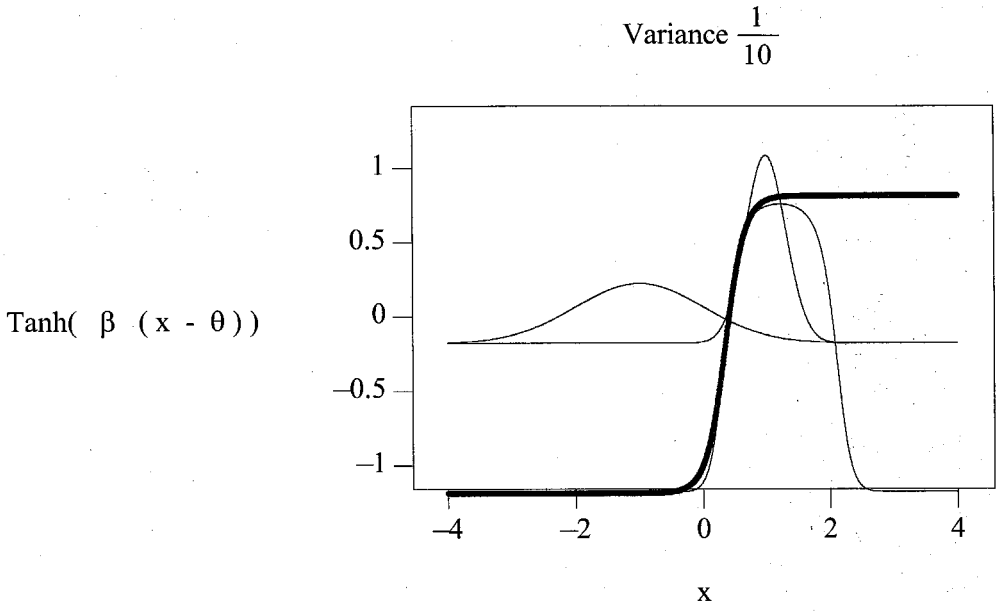


Figure 47. Left-Hand Side Variance 1.0, Right-Hand Side Variance $\frac{1}{10}$

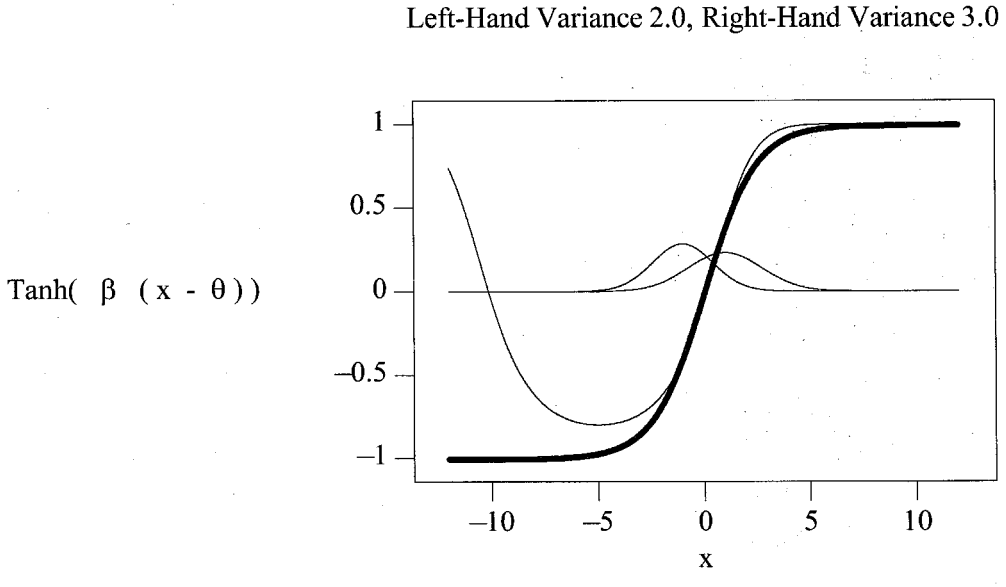


Figure 48. Left-Hand Side Variance 2.0, Right-Hand Side Variance 3.0

Left Variance 2.0, Right Variance 4.0

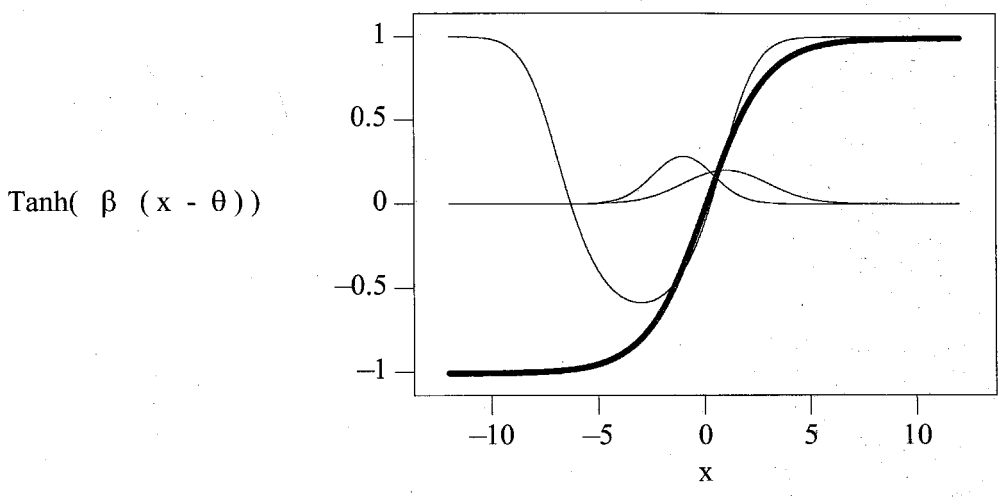


Figure 49. Left-Hand Side Variance 2.0, Right-Hand Side Variance 4.0

Left-Hand Variance 2.0, Right-Hand Variance 6.0

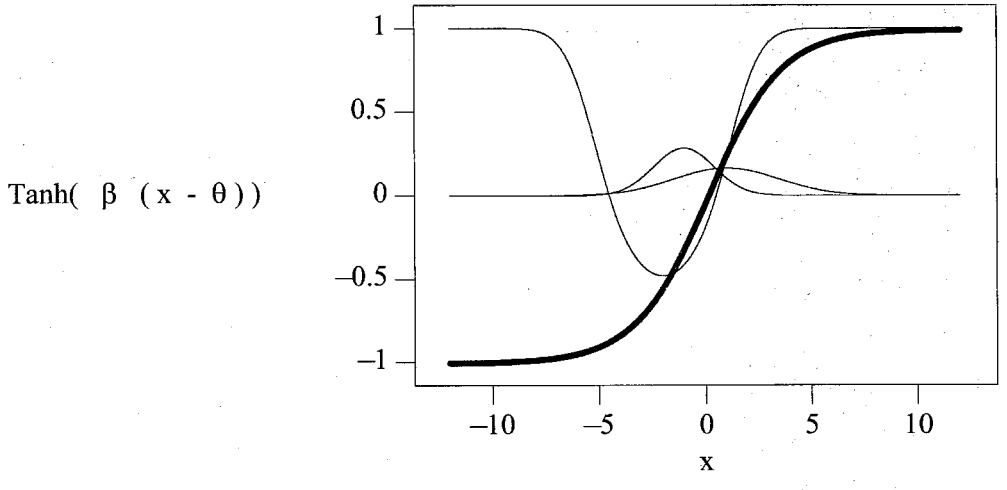


Figure 50. Left-Hand Side Variance 2.0, Right-Hand Side Variance 6.0

The conclusion to be drawn from these plots is that a Tanh function provides an adequate approximation to the soft decision sigmoid, as long as there is a separation of the means and the conditional variances are of the same order of magnitude.

11. Estimating The Sigmoid : A Case Study With Real Data

In real data processing situations, the equal variance assumption does not hold. As an example, glance over the means and variances for the *Six* and *Seven* image projections above. In this section, we get a rough feel for how well the Tanh approximation holds in the non equal variance scenario. To understand what *effective* $\sigma_{effective}$ to use with the Tanh form, we look at setting $\sigma_{effective}$ equal to the minimum, and maximum, of the two hypothesis variances, as well as using the geometric and arithmetic mean of the two variances. The different curves are shown below, as well as the optimum curve obtained from the ideal function ratio of the difference of the Gaussians over their sum. For the purposes of the plots below, the gain γ and threshold θ are the following quantities.

$$\gamma \equiv \frac{\mu_1 - \mu_0}{2\,\sigma_{effective}^{\,2}} \qquad \theta \equiv \frac{\mu_1 + \mu_0}{2}$$

$$Min[\,\sigma_1\,,\,\sigma_0\,] \qquad Max[\,\sigma_1\,,\,\sigma_0\,]$$

$$\frac{\sigma_1 + \sigma_0}{2} \qquad \sqrt{\sigma_1\,\sigma_0}$$

Equation 37. Effective Variance's For Tanh Approximation

We have dropped the second-order threshold correction in the real data calculations because we found the results were not influenced by them.

In the plots below, the solid lines with the small bullets along the curve is the ideal sigmoid, the difference of Gaussians over the sum of Gaussians. The dotted curves are the minimum and maximum variance estimators. The solid curve is the arithmetic mean, and the dashed curve is the geometric mean estimator. Relevant portions of the two hypothesis pdf's are also shown as solid curves.

Digit **6 , 7** Neuron 5 : Estimating γ and θ

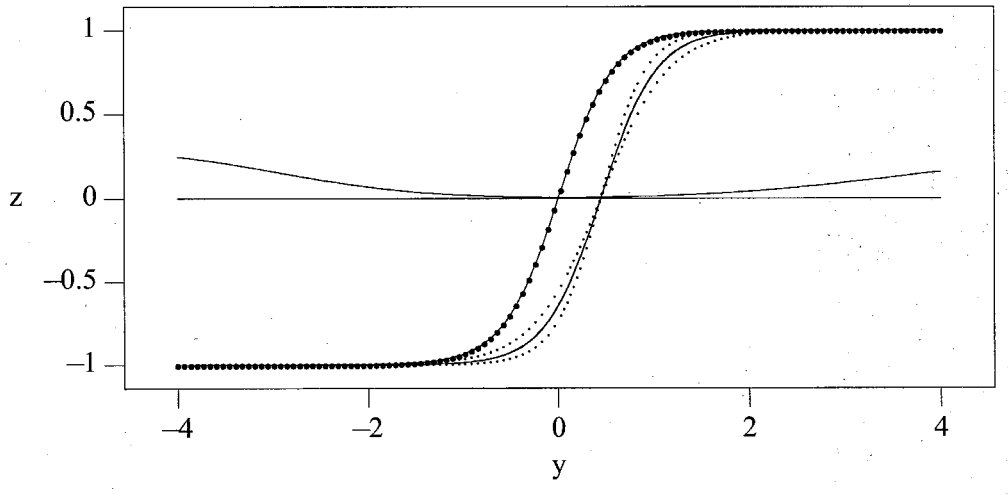


Figure 51. Estimating Sigmoid Gain(γ) and Offset(θ)

Digit **6 , 7** Neuron 4 : Estimating γ and θ

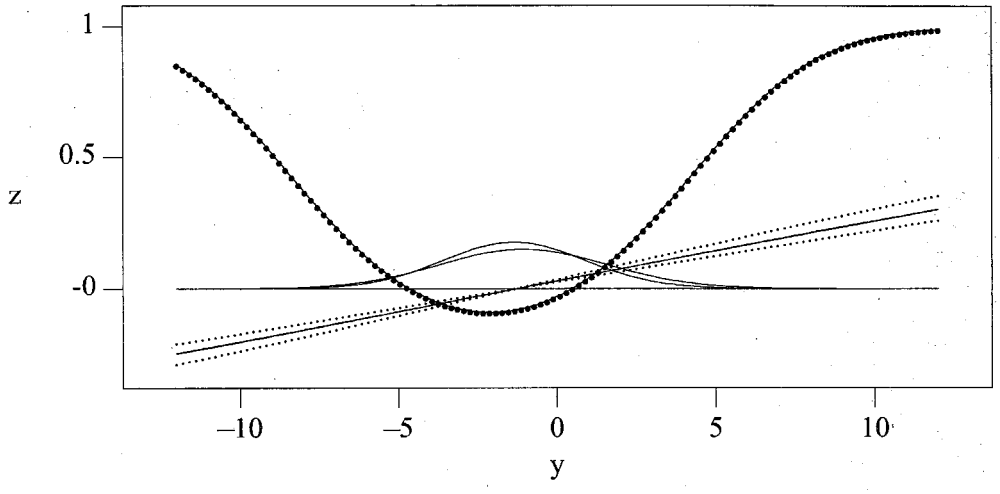


Figure 52. Estimating Sigmoid Gain(γ) and Offset(θ)

Digit **6 , 7** Neuron 3 : Estimating γ and θ

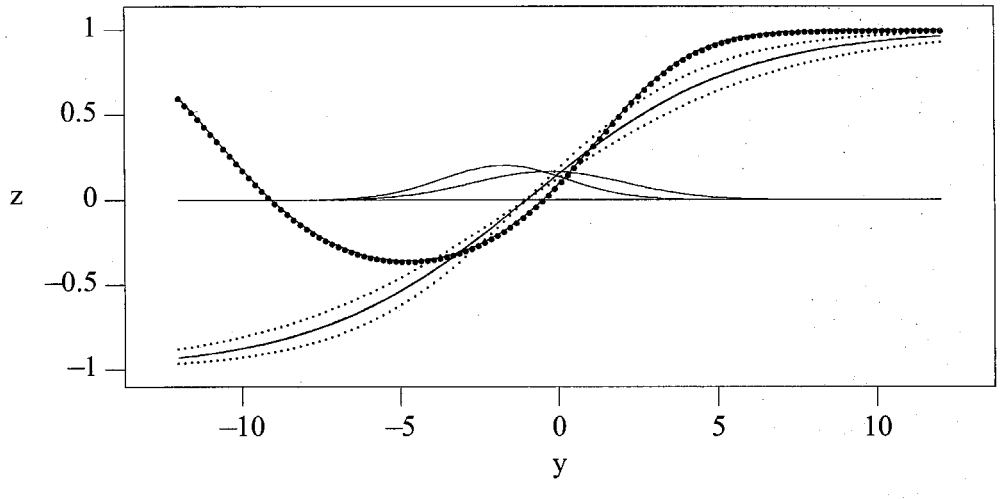


Figure 53. Estimating Sigmoid Gain(γ) and Offset(θ)

Stochastic Computation

Digit **6 , 7** Neuron 2 : Estimating γ and θ

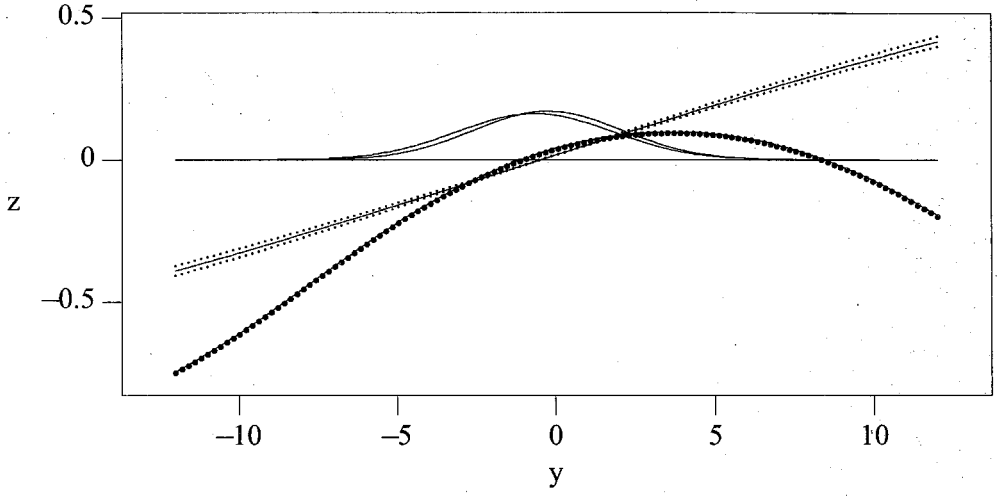


Figure 54. Estimating Sigmoid Gain(γ) and Offset(θ)

Digit **6 , 7** Neuron 1 : Estimating γ and θ

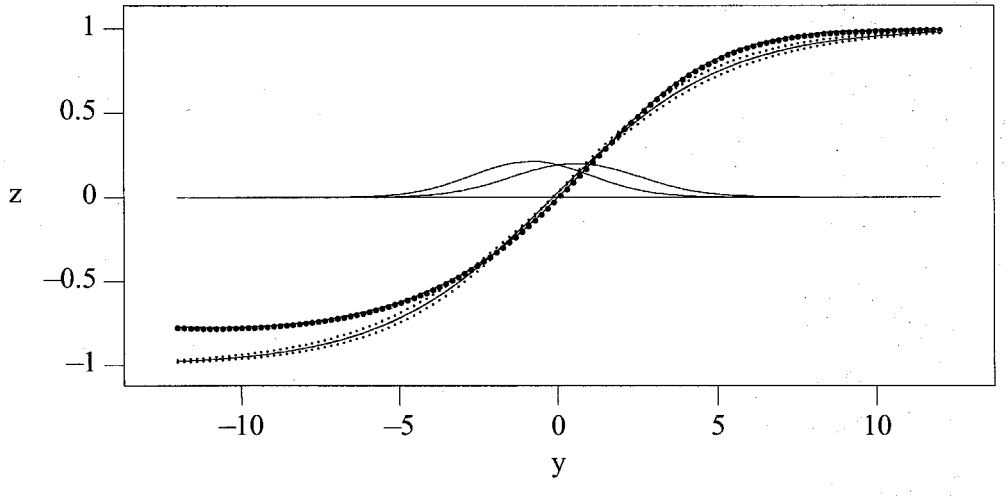


Figure 55. Estimating Sigmoid Gain(γ) and Offset(θ)

Stochastic Computation

The neuron 5, 3, 2, and 1 Tanh approximations are fair. In each of these cases, there is some separation of the means of the two conditional distributions which the Tanh function can take advantage of. For neuron 4, the Tanh approximation is poor. This is to be expected, since the means of the two distributions are both equal to about zero. The information in these cases is in the fact that there are differences in the second moments, the variance. However, the Tanh sigmoid function was developed under the assumption that there would be a separation in the means, and identical conditional variances. For almost equal means, the Tanh gain term, γ, is very small. The Tanh sigmoid is well approximated in this case by a straight line equal to $\frac{(\mu_1 - \mu_0)\ (2\ x - (\mu_1 + \mu_0))}{2\sigma^2}$. This straight line is seen in neuron 4's plot. However, the optimum sigmoid given by the difference in the conditional Gaussians, over their sum, is shaped more like a Mexican hat, so the line is a poor fit.

This example points up the fact that the Tanh function is a good sigmoid only when there is a reasonablely wide separation of the means for the two conditional hypotheses. When the means coincide, the Tanh sigmoid doesn't help the network implement soft decision decoding. Later on, we shall reinvestigate this example in much more detail. First, however, we study the performance tradeoff between soft and hard decoding.

12. Quantifying And Visualizing Soft Versus Hard Decision Decoding

As described above, a nonlinear function acting on the projection $\mathbf{W} \bullet \mathbf{X}$ can move the system from the hard decoding decision regime to the soft decision method, sometime resulting in significant performance improvement. To quantify this improvement, and provide an aid to

visualizing how the improvement is obtained, a three neuron single layer network is studied under both hard and soft decision decoding operation.

Consider the three-dimensional input space $\mathbf{X} \equiv (x_1, x_2, x_3)^T$. A point $\mathbf{X} \in \mathbf{R}^3$ is either a sample from the probability distribution $P[\mathbf{X}|H_1]$ or $P[\mathbf{X}|H_0]$. In the example below, set the mean and correlation matrix equal to the quantities given.

$$\mu_1 \equiv \begin{pmatrix} +1 \\ +1 \\ +1 \end{pmatrix} \qquad \mathbf{C}_1 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mu_0 \equiv \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \qquad \mathbf{C}_0 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Equation 38. Three Neuron Soft Versus Hard Decoding Example

Since both hypothesis correlation matrices are diagonal, the standard basis vectors yield statistically independent outputs when used as weight vectors for the three neurons.

$$\mathbf{W}_1 \equiv \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{W}_2 \equiv \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \mathbf{W}_3 \equiv \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Equation 39. Three Neuron Example Weight Vectors

Due to the symmetry of the two hypotheses density functions, the optimum threshold for each neuron/dimension is zero. Thus, a hard decision decoding rule for this example is the following.

$$\sum_{i=1}^{3} Sign[\mathbf{W}_i \bullet \mathbf{X}] \quad \begin{matrix} H_1 \\ > \\ < \\ H_0 \end{matrix} \quad 0$$

Equation 40. Three Neuron Example Hard Decision Rule

The hard decoding rule divides $\mathbf{R}^3$ into eight regions. Each region is assigned an overall system decision hypothesis, H_1 or H_0, depending on which sign dominates among the three coordinates (x_1, x_2, x_3). These eight regions are shown below. The dark cubes are regions where the hypothesis H_1 is the system decision. (Recall H_1 is the random variable with mean $\mu_1 \equiv (1,1,1)^T$.) The wireframe cubes are areas where H_0 is the system decision.

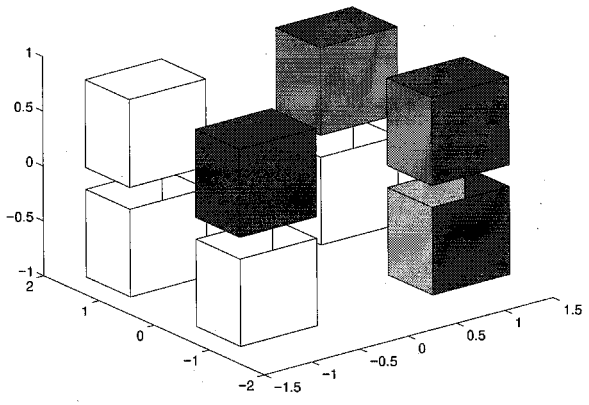


Figure 56. Hard Decision Regions

For soft decision decoding, the probability distributions have symmetries which we take advantage of. For soft decoding, a point $\mathbf{X} \in \mathbf{R}^3$ is assigned that hypothesis whose mean, μ_1 or μ_0, lies closest. The distance used here is the standard Euclidean metric.

$$|| \mathbf{B} - \mathbf{A} || \equiv \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + (b_3 - a_3)^2}$$

$$\left|\left| \mathbf{X} - \begin{bmatrix} +1 \\ +1 \\ +1 \end{bmatrix} \right|\right| \underset{H_0}{\overset{H_1}{\gtrless}} \left|\left| \mathbf{X} - \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \right|\right|$$

Equation 41. Soft Decision Metric For Points $\mathbf{X}$ In $\mathbf{R}^3$

Below, a *picture* of the two hypothesis distributions is seen. The lightly shaded sphere is intended to represent the Gaussian distribution $P[\mathbf{X} \mid H_0]$ with mean vector (-1, -1, -1). The more darkly shaded sphere is intended to represent the Gaussian distribution $P[\mathbf{X} \mid H_1]$ with mean vector (+1, +1, +1).

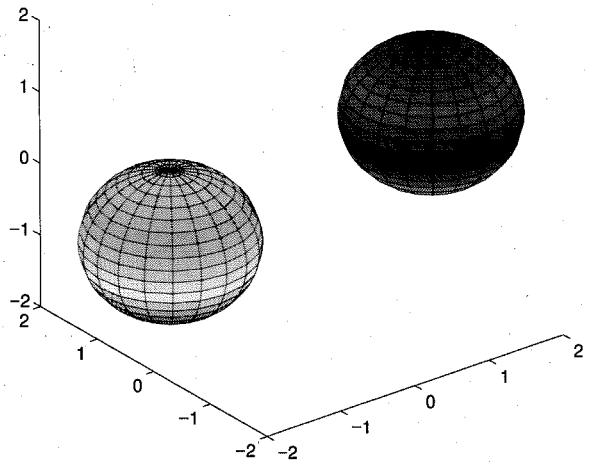


Figure 57. Soft Decision Spherical Distributions

The optimum division of $\mathbf{R}^3$ is given by the hyperplane of points equidistant to both mean vectors, which is not shown since it would blot out the nice spheres !

Contrasting the soft and hard decoding rules, the hard decoding rule is a ragged approximation to the soft decision hyperplane. The P_{error} improvement is the volume in $\mathbf{R}^3$ which is misclassified by the jagged edges of the hard decision boundary. If the misclassified probability distribution has a significant probability of occurrence in these misclassified volumes, the improvement in migrating from a hard decision to a soft decision rule can be significant. That is, if there are significant probability tails of the conditional distributions, then soft decision decoding helps the system performance. For systems which perform well under hard decoding, marginal performance gains are realized. However, as we shall see, and as we advanced above

in our beginning statements, typically each neuron individually does poorly in making a decision. Hence, in real life situations, soft decoding helps system performance. We shall quantify this improvement in an example in a later section.

13. Multiple Neuron Hidden Layer Architecture

In this section, we compare the hard and soft decision decoding output density functions. The two hypothesis projections are Gaussians with equal variance σ^2 and means $\mu_1 = -\mu_0 = 1$. The optimum sigmoid is of the form Tanh(β x). The threshold θ is zero since the means are symmetrically arrayed.

For hard decoding, the sign of the individual neuron projection's $\mathbf{W} \bullet \mathbf{X}$ is tallied for the neurons. The resulting sum's sign yields the decision : $+ \rightarrow H_1, - \rightarrow H_0$. The hard decoding output density functions are Gaussians by our application of the central limit theorem.

For soft decision performance, we use the sum of the Tanh output's to arrive at a decision. If the Tanh sum is positive, we conclude H_1, otherwise we conclude H_0. This is because Tanh is a monotonically increasing function which passes through the origin. A negative input to the Tanh map implies, for that neuron, H_0 is more probable. Similarly, a positive input implies H_1 is more probable. The Tanh function preserves the sign, and allows us to make a system decision based on the sign of the sum of the Tanh neuron outputs. The nonlinear mapping of the Tanh function in the positive and negative regions however allows us to optimize the cooperative nature of the eleven neurons working together to arrive at system decision.

To determine the soft decision performance improvement over the hard decision performance, we need to calculate the probability distribution for the Tanh output. This is done using the change of variable technique.[20] Let z equal the output of the Tanh mapping. The probability density function for z is detailed below, for the case of identical variances for the two conditional projection random variables.

$$\beta \equiv \frac{\mu_1 - \mu_0}{2\,\sigma^2} \qquad T = \frac{\mu_1 + \mu_0}{2} \qquad z = Tanh\left[\,\beta\,(x - T)\,\right]$$

$$p_z(z) = \left|\,\frac{d\ arcTanh(z)}{\beta\ dz}\,\right| \qquad p_x\left[\,\frac{arcTanh(z)}{\beta} + T\,\right]$$

Equation 42. Calculating The Tanh Random Variable Output Probability Density Function

$$\frac{d\ arcTanh(z)}{dz} \equiv \frac{1}{1 - z^2} \quad , \quad arcTanh(z) \equiv Ln\left[\,\sqrt{\frac{1 + z}{1 - z}}\,\right]$$

Equation 43. Interim Expressions Used To Arrive At The Results Below

$$p_z(z) \equiv \frac{e^{-\frac{\left(\frac{arcTanh(z)}{\beta} + T - \mu\right)^2}{2\sigma^2}}}{\beta\,(1-z^2)\,\sqrt{2\,\pi\,\sigma^2}} \qquad For\ \ -1 < z < 1\ ,\ And\ Zero\ Elsewhere$$

Equation 44. Probability Density Function For z With Arbitrary β and Threshold T

Under the assumption of symmetrical means, and identical variances for the two conditional projection random variables, the threshold T is zero, and β becomes $\frac{\mu}{\sigma^2}$. Thus, the density for z

becomes the following.

$$p_z(z) \equiv \frac{\sigma\, e^{\frac{-\left(\frac{\sigma^2}{\mu} Ln\left[\sqrt{\frac{1+z}{1-z}}\right] - \mu\right)^2}{2\sigma^2}}}{\mu\,(1-z^2)\,\sqrt{2\,\pi}} \qquad For\ -1 < z < 1\ ,\ And\ Zero\ Elsewhere$$

Equation 45. Equal Variance, Symmetrical Mean's, β And Threshold Substituted

A plot of the transformed density function is seen below. Below, we have set the means ($\mu_{0,1}$) to ±1, and vary the variance of the projection pdf's, which for both hypotheses is σ^2.

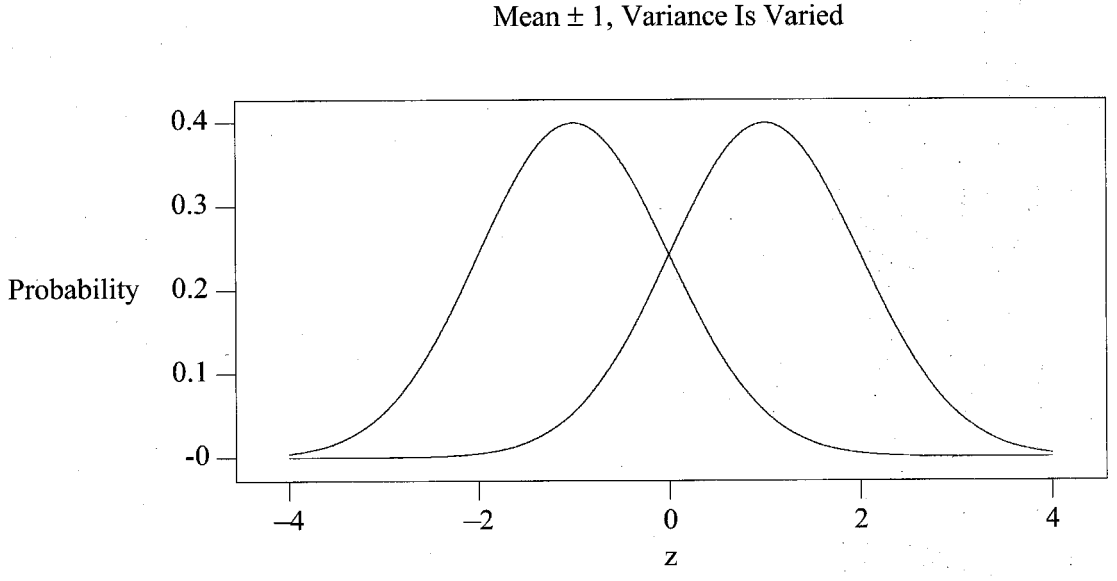


Figure 58. Mean ± 1, Variance 1 Input Gaussian's Pdf's

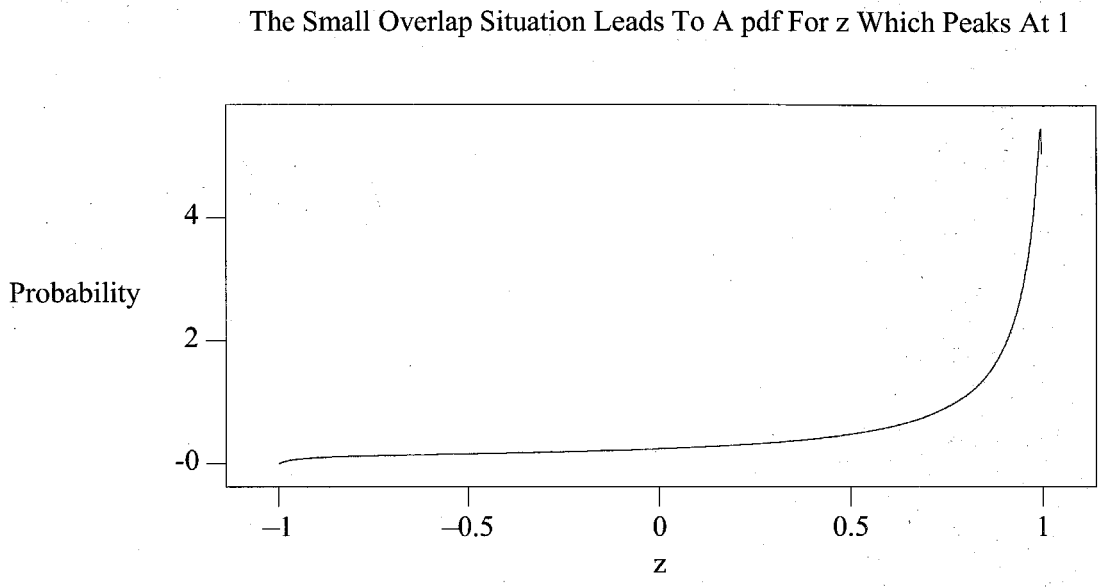


Figure 59. Tanh Output For Mean ± 1, Variance 1 Input Gaussian's

The above plot is the output pdf of a Gaussian random variable with mean 1, and variance 1 passed through a soft decision Tanh sigmoid constructed under the assumption that the two conditional projections pdf's were mean ±1, and variance 1. The pdf for a Gaussian with mean -1, and variance 1, passed through the same sigmoid, would have an identical shape, but reflected over the vertical axis.

The above plot agrees with our intuition of what the soft decision pdf output should look like. For small pdf overlap, and resulting small tails under the threshold, we expect that the probability of a +1, given that our projection $\mathbf{W} \bullet \mathbf{X}$ was above the threshold, would be close to +1 with large probability. Furthermore, the probability should decay or roll off as we approach -1, which is also seen.

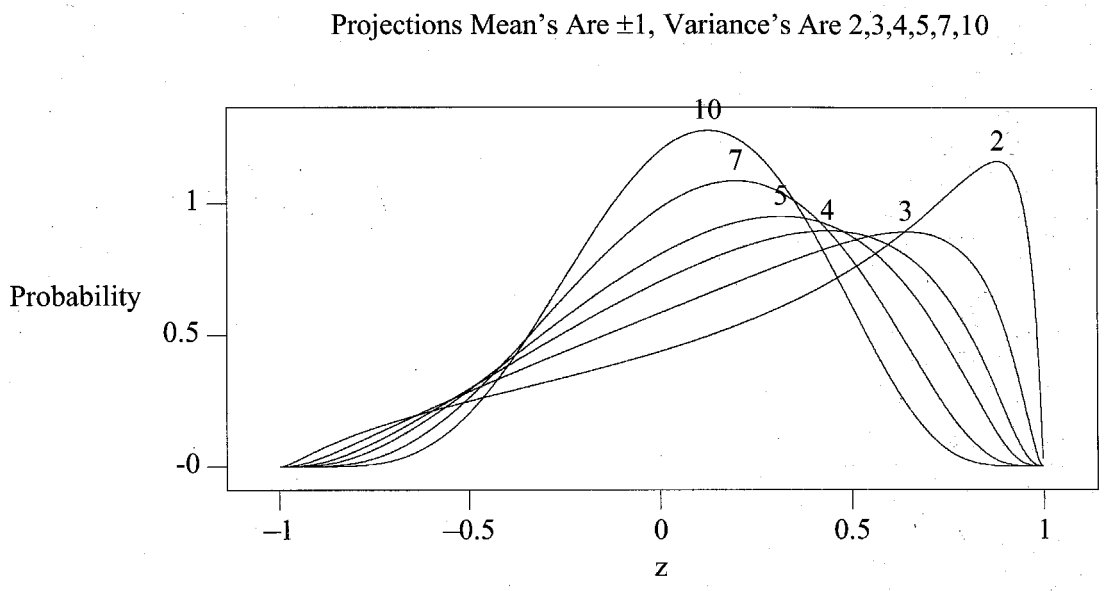


Figure 60. Tanh Pdf For Various Variance's And Mean ± 1

The above plot again agrees with our intuition of what a soft decision output should look like. Consider the limit as $\sigma \rightarrow \infty$. In this case, we would expect the soft output to approach a Gaussian centered at 0, with a broad variance. This is because in this limit, each neuron's output becomes approximately uniform over the interval $(-\infty , \infty)$. Thus, the probability density of the neuron output would be Gaussian, with approximately a zero mean, and a large variance. Looking at the plots above, as the conditional variance is increased from 1 to 10, we indeed see the mean approach zero, and the soft output variance broaden. The behavior of the mean and variance of the Tanh output is quantified in the plot below. For this plot, the mean of the input Gaussian was fixed at 1, and the input variance varied. Note that the mean approaches zero, and the output variance approaches 1, as the input variance $\rightarrow \infty$.

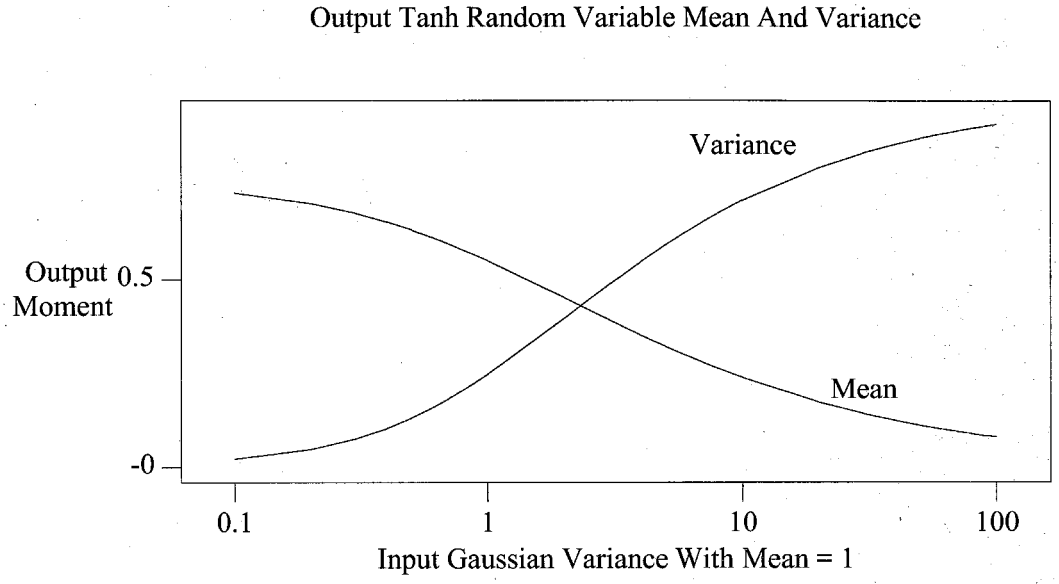


Figure 61. Tanh Density Mean And Variance As A Function Of Input Variance

13.1 Calculating P_{error}^{Hard} Versus P_{error}^{Soft}

The probability of error for a single pair of conditional pdfs centered at ± 1, with different variance's, is seen below in the table and plot.

The table below shows the individual probability of error for a single neuron with conditional pdf's centered on ± 1, and as the variance is varied.

Variance	1	2	4	8	16	32	64	128	256
Pe	0.159	0.240	0.309	0.362	0.401	0.430	0.450	0.465	0.475

TABLE 8. Probability Of Error Versus Variance Of Symmetrical Projections

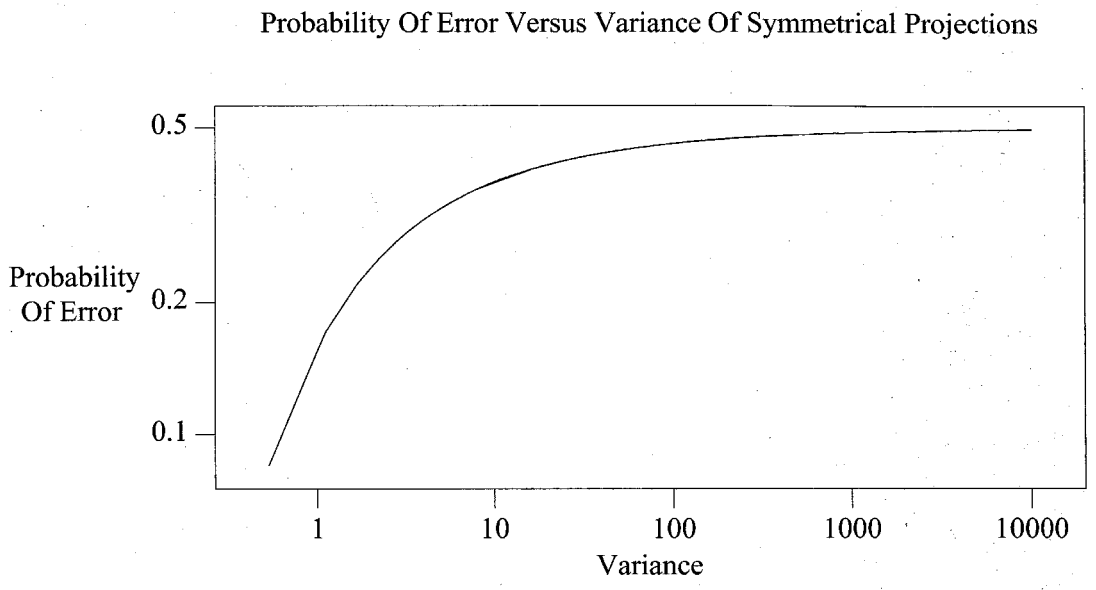


Figure 62. Probability Of Error Versus Variance Of Symmetrical Projections

The above graph is the function of the variance below.

$$P_{Error} \equiv \frac{1}{2} Erfc\left[\frac{1}{\sigma \sqrt{2}} \right]$$

Equation 46. The Probability Of Error For Two Gaussians At ± 1, With Identical Variances

The hard decoding probability of error for M neurons ensemble is given by the sum below. To avoid ties, we shall consider M to always be odd.

$$P(M)_{error}^{Hard} \equiv \sum_{i=\frac{M+1}{2}}^{M} \left[\begin{matrix} M \\ i \end{matrix} \right] p^i \left[1 - p \right]^{M-i}$$

Equation 47. Hard P_{Error} For M Neurons

The probability of a single neuron being incorrect is p. Since the Tanh function is monotonically increasing, and maps zero to zero, the total probability of the Tanh output below zero for the conditional density with mean at +1 must equal the Gaussian tail of the same conditional density below zero. This is the connection between calculating the probability of error for a single neuron before and after the Tanh mapping. This relation is shown below.

$$p \equiv \int_{-\infty}^{0} \frac{\sigma\, e^{\frac{-\left(\frac{\sigma^2}{\mu} Ln\left[\sqrt{\frac{1+z}{1-z}}\right] - \mu\right)^2}{2\sigma^2}}}{\mu\,(1-z^2)\,\sqrt{2\,\pi}}\, dz \equiv \frac{1}{2}\, Erfc\left[\frac{\mu}{\sqrt{2}\,\sigma}\right]$$

Equation 48. Relation Between Probability Of An Error Before And After The Sigmoid

As a check, we set $\mu_1 \equiv +1$, $\mu_0 \equiv -1$, and $\sigma_1 \equiv \sigma_0 \equiv 1$. Numerically integrating using MAPLE, we find the soft integral and the Erfc expression above both yield 0.158655. This soft error integral result implies the Tanh density function was calculated correctly.

For a system of M neurons, if all outputs are statistically independent, then the soft system performance is seen below.

$$P_e^{Soft} \equiv \frac{1}{2} \; Erfc\left[\frac{\mu}{\sigma} \sqrt{\frac{M}{2}} \right]$$

Equation 49. Soft P_{Error} For M Neurons

The plot below compares P_{error}^{Hard} versus P_{error}^{Soft} for the cases of 11, 51, and 101 neurons in parallel, as a function of the variance, for the conditional means at ± 1. The upper curve is the hard decision result, and the lower curve the soft decision result.

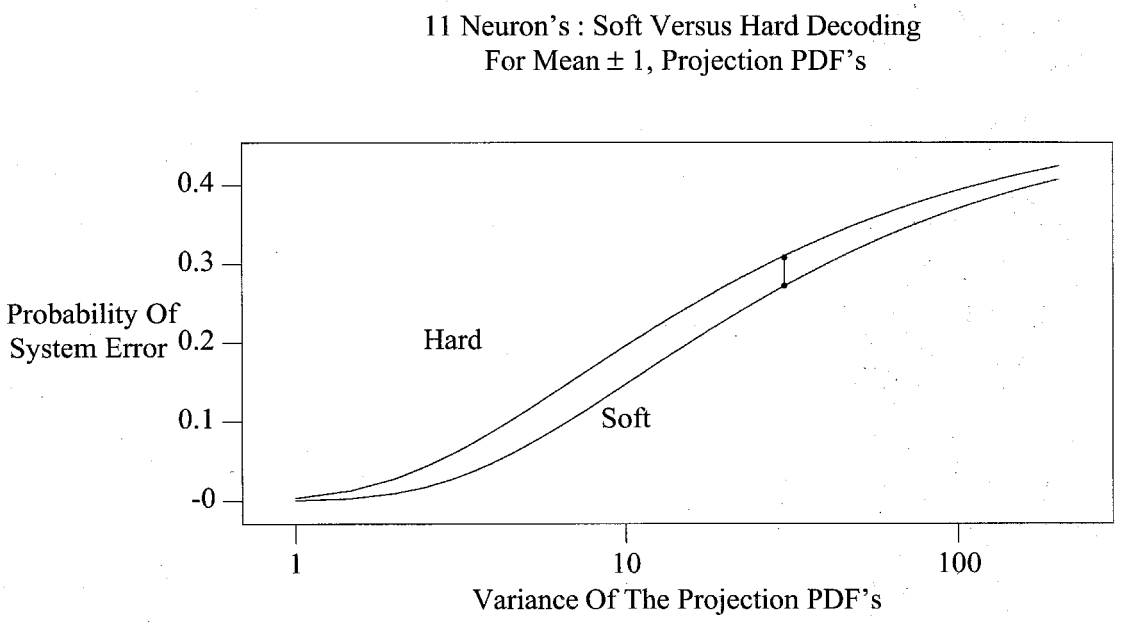


Figure 63. 11 Neuron Variance Versus P_{error}

For the 11 neuron case, to be sure everything is kosher, we conduct the following experiment. Using MATLAB, we generate 200,000 sets of 11 numbers, drawn from a normal distribution

with mean 1, and variance 30. With 100,000 sets, we conduct hard decision decoding. With the other 100,000 sets, we pass the eleven numbers through a Tanh function, with gain β equal to $\frac{1}{30}$. These eleven Tanh outputs are then tallied, and thresholded about zero. From this simulation run, we obtain a P_{error}^{Hard} and a P_{error}^{Soft}. These are shown on the above graph as the solid vertical line segment with bullets on the end. The numerical simulation agreement with the theoretical curves is very good, for both the hard and soft decoding.

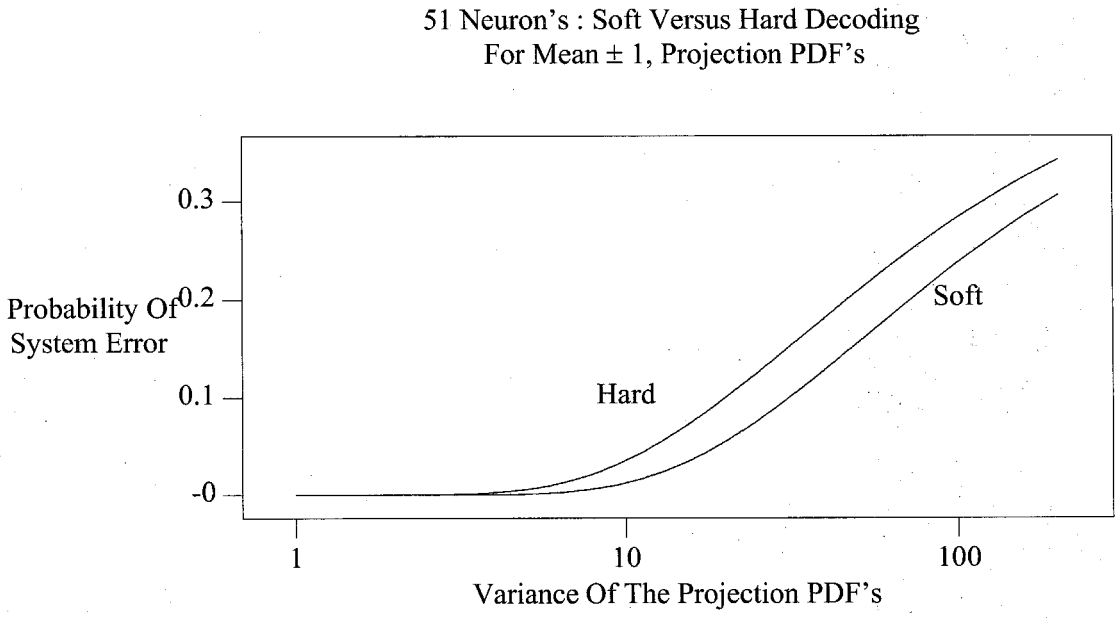


Figure 64. 51 Neuron Variance Versus P_{error}

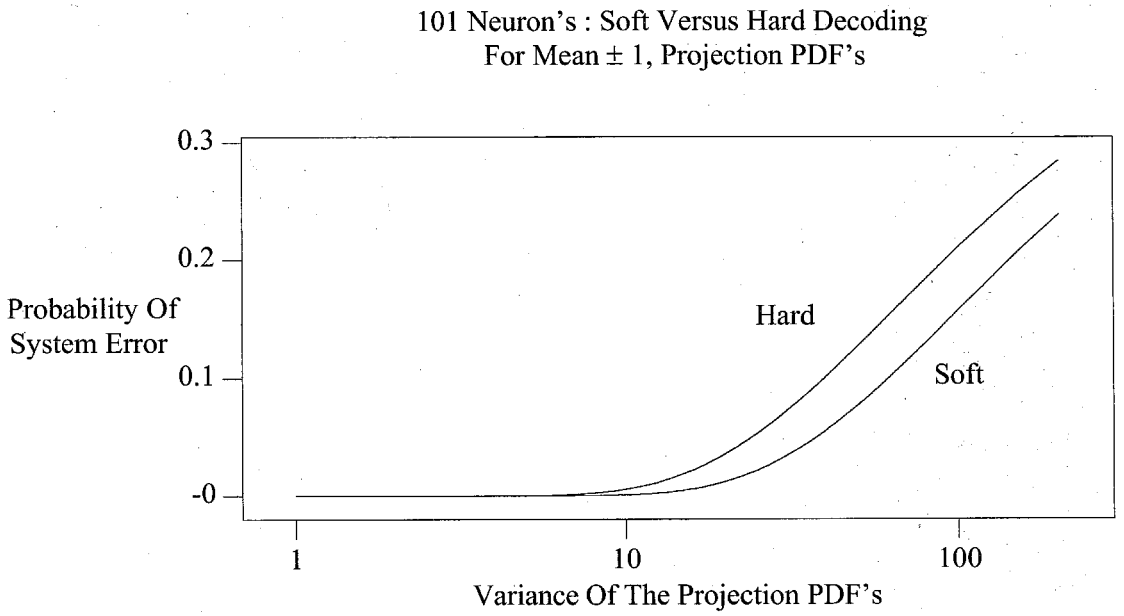


Figure 65. 101 Neuron Variance Versus P_{error}

13.2 Soft And Hard Decoding Versus The Number Of Neurons

We now look at how soft decision decoding does as the number of neurons in any one layer is increased. Here, the mean of the two projection pdf's is set to ± 1 and the variance to 1 for both hypotheses. The p_{error} for the transformed Tanh pdf is calculated, and a soft decision curve obtained, as a function of the number of neurons. We assume for the soft decision that the central limit theorem will yield a Gaussian pdf for the sum of the soft neuron outputs. As the plots above show for the transformed Tanh output densities, individual neuron's with conditional projection densities which overlap a reasonable amount yield a Gaussian-like sigmoid transformed pdf. Thus, typically for numbers of more than about five neurons per layer, applying the central limit theorem means the sum of the Tanh outputs will approach a

Gaussian distribution. We also assume the neuron outputs are statistically independent, so that the soft decision Gaussian pdf will have a mean of $n\ \mu_z$ and a variance of $n\ \sigma_z{}^2$, where z is the output of a single neuron's Tanh mapping. The system probability of soft error is the area of the resulting Gaussian random variable $\sum_{i=1}^{\#Neurons} z_i$ tail under the zero threshold. This was calculated for single neurons above, and shown to be $P_{error} \equiv \frac{1}{2}\ Erfc(\frac{\mu}{\sigma\sqrt{2}})$.

To use the equation above, the output SNR is needed as a function of the input SNR. That is, for $y \equiv \mathbf{W} \bullet \mathbf{X}$, and $z \equiv Tanh(y)$, to determine the system soft error, $\frac{\mu_z}{\sigma_z}$ as a function of $\frac{\mu_y}{\sigma_y}$ is needed. The solid curve below shows this relationship. The dotted curve shows the identity map $\mathrm{SNR}_{out} \equiv \mathrm{SNR}_{input}$.

Tanh SNR Output Versus Projection SNR Input

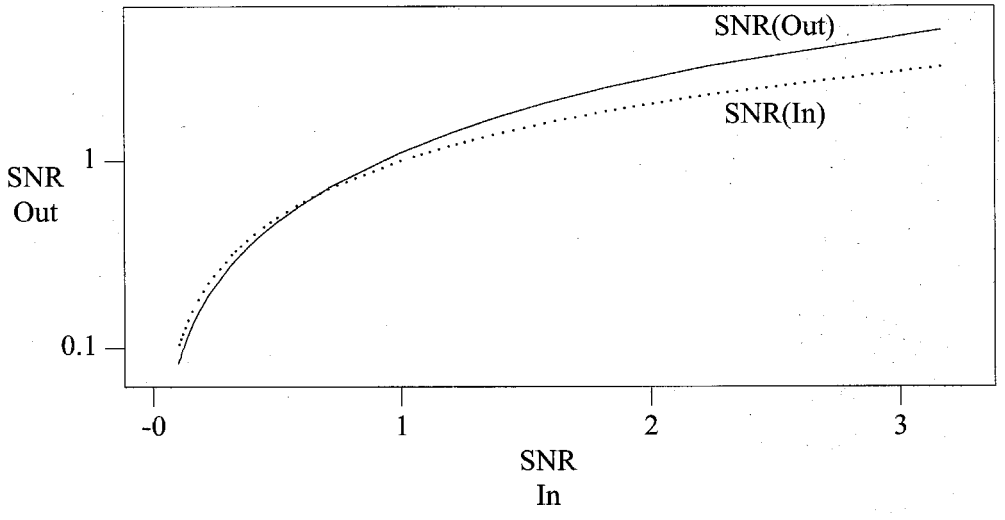


Figure 66. Tanh SNR Output Versus Projection SNR Input

The plot shows the SNR map is the identity for small SNR_{in}, and exceeds SNR_{in} for large SNR_{in}. Thus, a lower bound on system soft P_{error} can be obtained by using the input SNR $\equiv \frac{\mu_y}{\sigma_y}$ in the Erfc equation shown above. This yields the two plots below. The vertical bar is the same MATLAB simulation run described above.

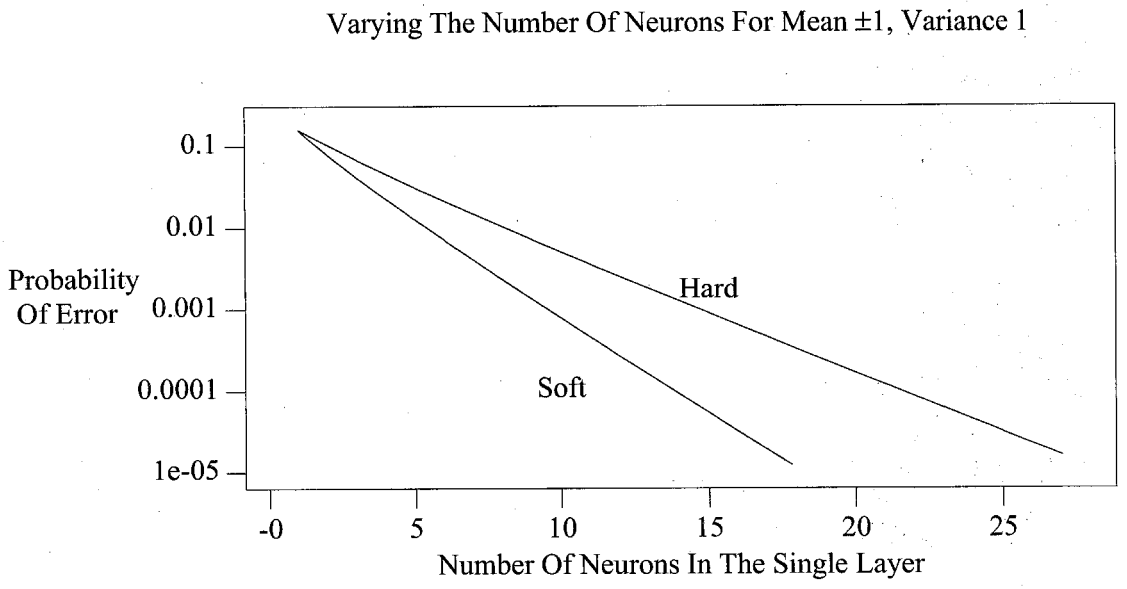


Figure 67. Varying The Number Of Neurons For $\mu \equiv \pm 1, \sigma \equiv 1$

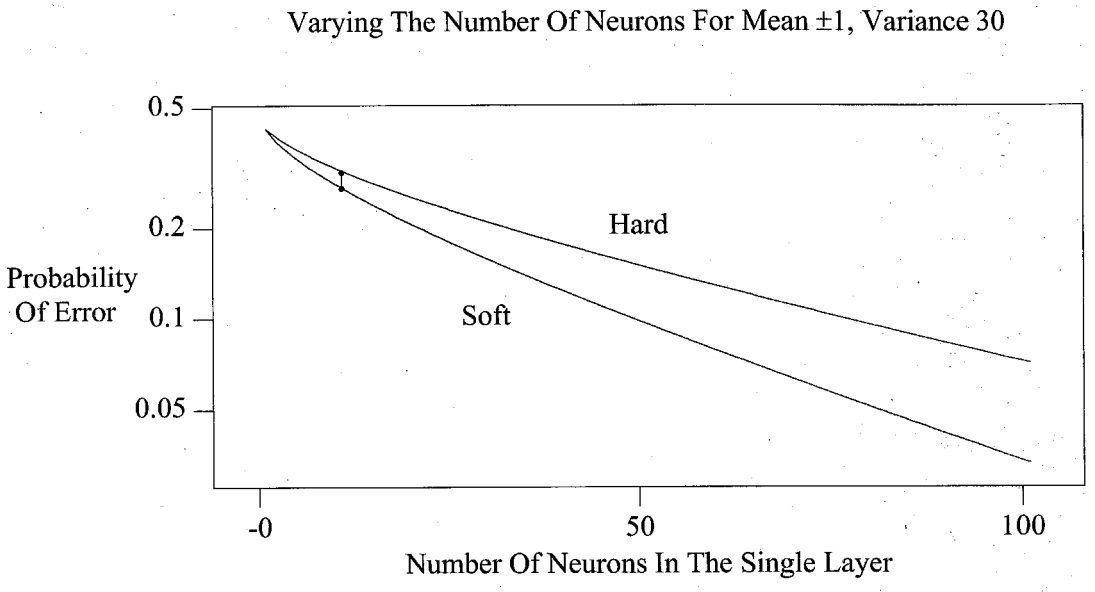


Figure 68. Varying The Number Of Neurons For Mean 1, Variance 30, Projections

As can be seen from the plots, soft decision decoding has an increasing performance advantage over hard decision decoding as the number of neurons increases.

14. Multiple Layer Networks

14.1 Linear Versus Non-Linear Networks

The discussion above centered on how to derive a suitable set of weight vectors for a single layer of neurons. For a linear network (i.e., without nonlinear sigmoids on each neuron output), a multiple (L) layer network has no advantage over a single layer network. This is due to the fact that a linear multiple layer network can be rewritten as a single layer network. That is, suppose layer i takes the set of inputs $\mathbf{X}_i$ and maps these to an output set of neurons $\mathbf{Z}_i$. This

mapping can be achieved by a not necessarily square matrix $\mathbf{M}_i$ such that $\mathbf{Z}_i = \mathbf{M}_i \bullet \mathbf{X}_i$. A cascade of n layers can be written as one matrix transformation $\mathbf{M} = \mathbf{M}_L \bullet \mathbf{M}_{(L-1)} \cdots \mathbf{M}_1$. This yields a single layer network with transfer matrix $\mathbf{M}$, which maps the first input vector $\mathbf{X}_1$ directly to the final output vector $\mathbf{Z}_L$: $\mathbf{Z}_L \equiv \mathbf{M} \bullet \mathbf{X}_1$.

The linear network example implies the power of a multilayer network is due to the nonlinear features of the computation. Above, nonlinearities were introduced via sigmoids which modify the neuron's projections onto a weight vector. However, there is typically an additional source of nonlinearity in biological neural systems. This second source of nonlinear operation is that introduced by lateral inhibition. Before we study *how* lateral inhibition modifies network performance, let us study *why* we would need lateral inhibition.

14.2 Lateral Inhibition And Winner Take All Networks

A Winner Take All (WTA) network is seen in the figure below. The lateral arcs between the cluster of three neurons $\{M_1, M_2, M_3\}$ serve to suppress the output of all neurons except the strongest neuron. The largest magnitude neuron output is unchanged.

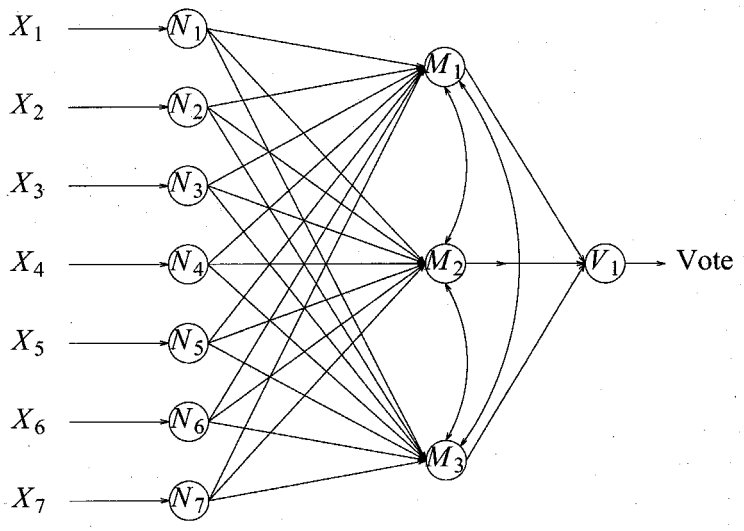


Figure 69. Winner Take All Neural Network

The lateral inhibition which serves to implement the WTA function has been studied by several authors.[21] [22] [23] When the lateral weights are symmetric and equal in magnitude, the WTA function selects from the set of neuron outputs z_i, the largest output and forces the remaining z_i outputs to zero. This operation is statistically equivalent to selecting the largest element from the set $\{z_i\}$. Before proceeding with an explanation of what this sampling behavior implements, recall how a single neuron makes a decision. The hypothesis test's were based on the difference between the means of the two hypotheses $P[H_0 | \mathbf{X}]$ and $P[H_1 | \mathbf{X}]$, projected onto the weight vector for that neuron. A single layer consists of several weight vectors "looking" in different directions of the input space, and expecting the projected means to differ. That is, the single layer decision rule was dependent on case I above occurring. This separation of the projection means does not always occur, as the following texture example will show.

Stochastic Computation

14.3 Textures And Lateral Inhibition

Consider the two 32 by 32 pixel textures seen below.

Figure 70. Discrete Pixel Texture

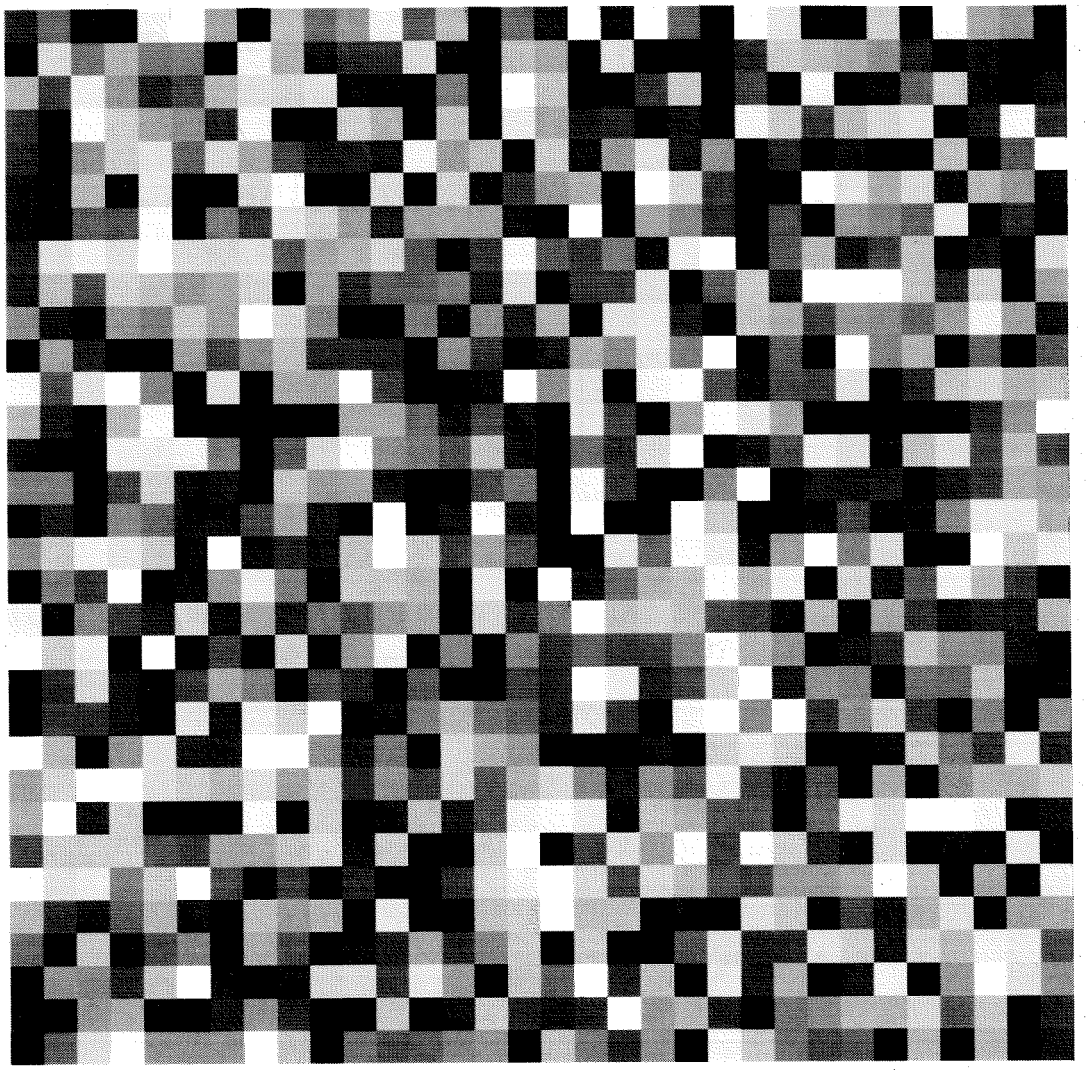


Figure 71. Uniform Pixel Texture

In both images, the pixels vary between -1(black) and +1(white). The first texture consists of random pixels which are ± 1 with equal probability. The second texture has pixels which are drawn from a uniform distribution over the interval [-1,1]. Thus, each pixel in both images is a

random variable with a mean of zero. However, the discrete ± 1 image pixels have a variance of 1, while the uniform pixels have a variance of $\frac{1}{3}$.

$$\mu_0 \equiv \{-1\} \bullet Prob\ Of\ \{-1\} + \{+1\} \bullet Prob\ Of\ \{+1\} \equiv \frac{1}{2} \bullet \{-1\} + \frac{1}{2} \bullet \{+1\} \equiv 0$$

$$\mu_1 \equiv \frac{1}{2}\int_{-1}^{+1} x\ dx \equiv \left[\frac{x^2}{4}\right]_{-1}^{+1} \equiv 0$$

Equation 50. The Hypothesis Pixel Means

$$\sigma^2{}_0 \equiv \{-1\}^2 \bullet Prob\ Of\ \{-1\} + \{+1\}^2 \bullet Prob\ Of\ \{+1\} \equiv \frac{1}{2} + \frac{1}{2} \equiv 1$$

$$\sigma_1{}^2 \equiv \frac{1}{2}\int_{-1}^{+1} x^2\ dx \equiv \left[\frac{x^3}{6}\right]_{-1}^{+1} \equiv \frac{1}{3}$$

Equation 51. The Hypothesis Pixel Variance's

Let hypothesis **D** be the texture consisting of the Discrete ± 1 pixels, while hypothesis **U** is the Uniform random pixel texture. We desire to train a neural network to distinguish between the two hypotheses. Since the pixels are statistically independent of each other in both images, there is, by a symmetry argument, no preferred direction in the 32 by 32 or 1024 dimensional space where the images reside. In each direction, the projected means of both hypotheses is zero. For an arbitrary weight vector, we set the weight vector elements equal to ± 1. This is because any

one pixel is no more important than another, so each weight vector element should have the same magnitude. The sign flip is due to the pixel symmetry about zero under both H_D and H_U. Changing different weight elements sign will allow us to obtain several statistically independent neuron outputs. The independence is shown below. Note that the expectation is taken with respect to both the **X** pixels, and the **W** pixels. The y_i are also Gaussian by the Central Limit Theorem, for sufficiently large dimension N.

$$y_i \equiv \mathbf{W}_i \bullet \mathbf{X} \ , \qquad E[\, y_i \, y_j \,] \equiv \mathbf{W}_i{}^T \bullet \mathbf{C} \bullet \mathbf{W}_j \equiv \mathbf{W}_i{}^T \bullet \mathbf{I} \bullet \mathbf{W}_j$$

$$\equiv \mathbf{W}_i{}^T \bullet \mathbf{W}_j \equiv \sum_{k=1}^{k=1024} E[\, w_i(k) \, w_j(k) \,] \equiv \sum_{k=1}^{k=1024} E[\, w_i(k) \,] \; E[\, w_j(k) \,]$$

$$\equiv 1024 \; \left(\frac{1}{2}(-1) + \frac{1}{2}(+1) \right)^2 \equiv 0$$

Equation 52. Independence Of The Texture Neuron Outputs

Uncorrelated Gaussian random variables are statistically independent. Since in our example the images have a total of 1024 pixels, the one-dimensional random variable y = **W** • **X** will, under hypothesis **D** have mean zero, and variance 1024. Under hypothesis **U**, the random variable y will have mean zero and variance $\frac{1024}{3}$. The two pdf's for $y_{\mathbf{D}}$ and $y_{\mathbf{U}}$ are seen below.

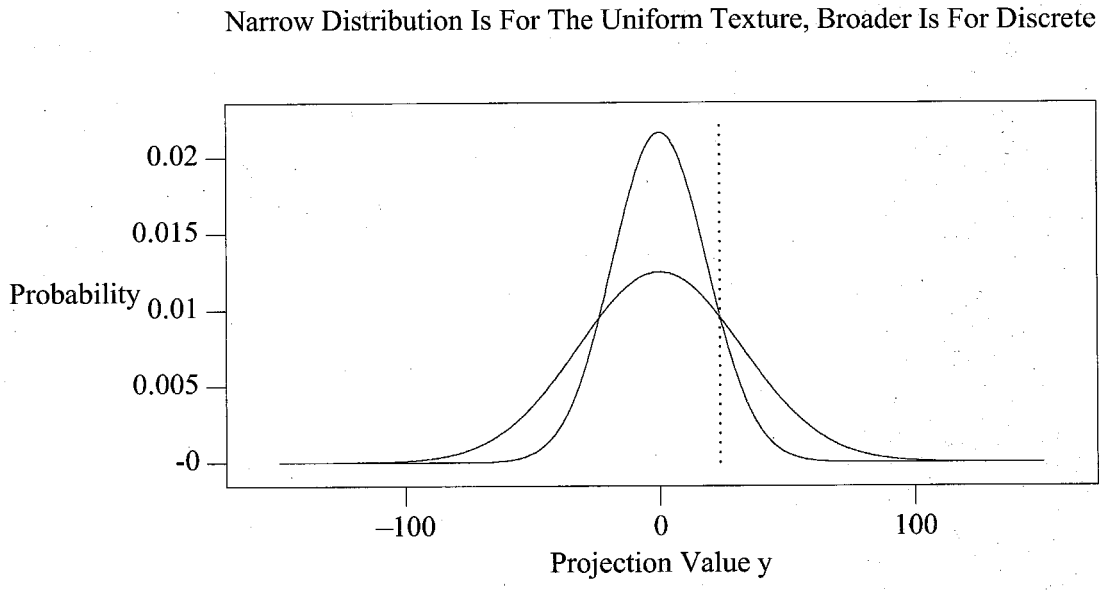


Figure 72. Sigmoid For Equal Mean & Different Variance Projection's

The optimum sigmoid consisting of the difference of the Gaussians over their sum, is seen below.

$$Sigmoid\ \ Function \rightarrow f(y) \equiv \frac{P[y_{\mathbf{D}}] - P[y_{\mathbf{U}}]}{P[y_{\mathbf{D}}] + P[y_{\mathbf{U}}]}$$

Equation 53. Mexican Hat Texture Sigmoid

The shape is markedly different from the Tanh form. The Mexican Hat reflects the fact that for a projection falling near the origin, hypothesis **U** is the more likely pdf to have generated that point. The opposite is true for points far from the origin, where the larger variance distribution for hypothesis **D** is more likely to be responsible for the projected point.

Stochastic Computation

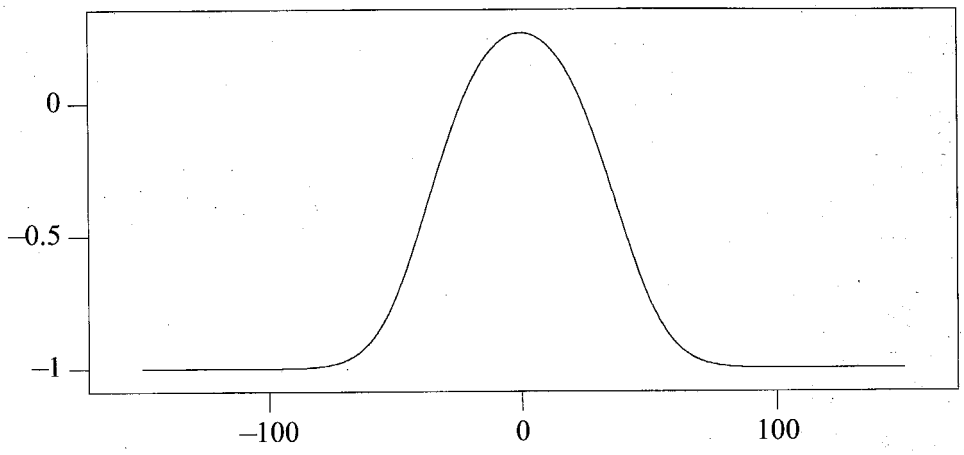


Figure 73. Sigmoid For Equal Mean & Different Variance Projection's

A MATLAB simulation will be used for the purposes of illustration. We generate 50,000 random projections by the method described above, for 32 by 32 pixel images. This yields the histogram plot below.

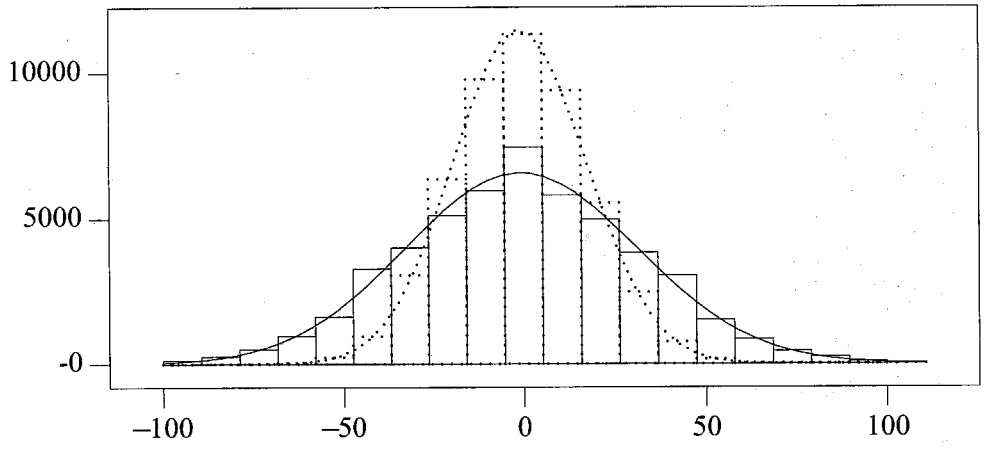


Figure 74. Solid Is For Discrete Texture Images, Dotted For Uniform Texture Images

Stochastic Computation

The envelope curve outline here, and in succeeding plots, are analytically the density function which should be expected. Numeric agreement with theory is good. The tabulated moments illustrate the agreement.

	Discrete Texture		Uniform Texture	
	Mean	Variance	Mean	Variance
Theory	0	1024	0	341.3
MATLAB	0.11	1022.5	0.0	336.2

TABLE 9. MATLAB Texture Moments

Our ultimate goal here is to develop a network which can distinguish these two textures. Our previously developed algorithms do not work well with this type of problem. This is because the information which enables a statistical test to be devised lies in the differing second moments of the two densities. Our previous work took advantage of differing means. One approach would be to arbitrarily put a threshold such as the dotted vertical line in the above plot, to pick off differences in one tail only. However, there is only $P_{\mathbf{D}} \equiv 0.23$, and $P_{\mathbf{U}} \equiv 0.10$, to the right of the threshold, so such a neuron has a $P_{error} \equiv \frac{(\ 0.1\ +\ 0.77\ )}{2} \equiv 0.435$. This is not good performance, although our discussion above indicates we can obtain good system performance with individual neuron error rates such as this, if we use enough neurons in parallel. A soft decision approach using a nonlinear sigmoid helps somewhat. We could implement the full soft decision techniques discussed above. We can also apply a pseudo technique which is partially hard and soft, and which we shall now briefly describe.

The pseudo hard-soft technique is interesting because the Mexican hat structure is one often seen in biological specimens. Consider the Mexican hat shaped curve below, which is the difference of the two hypotheses pdf's. Institute a hard decision rule on the Mexican hat output. This rule remains a zero threshold test, since we are only eliminating a positive constant multiplier, namely the denominator term of the full soft decision sigmoid. Hypothesis **U** (the more peaked Gaussian) is chosen if the sigmoid output is greater than zero, and hypothesis **D** is chosen if the sigmoid output is less than zero. Conduct hard decision voting on the thresholded neuron sigmoid outputs. Our error performance improves by a factor of two, since now we are implementing a dual tail threshold equivalent to the simultaneous application of the two symmetric dotted thresholds in the figure above. Thus, our P_{error} with soft decision decoding becomes 0.37. The performance of this approach lies between true soft and hard performance. However, it is a simple technique which serves to double the individual neuron P_{error} with only a mild increase in complexity.

Regardless of whether hard, soft or the pseudo technique is implemented, there is still considerable error inherent in the decision rule. Lateral inhibition is a nonlinear technique which provides another level of system improvement beyond hard/soft/pseudo performance.

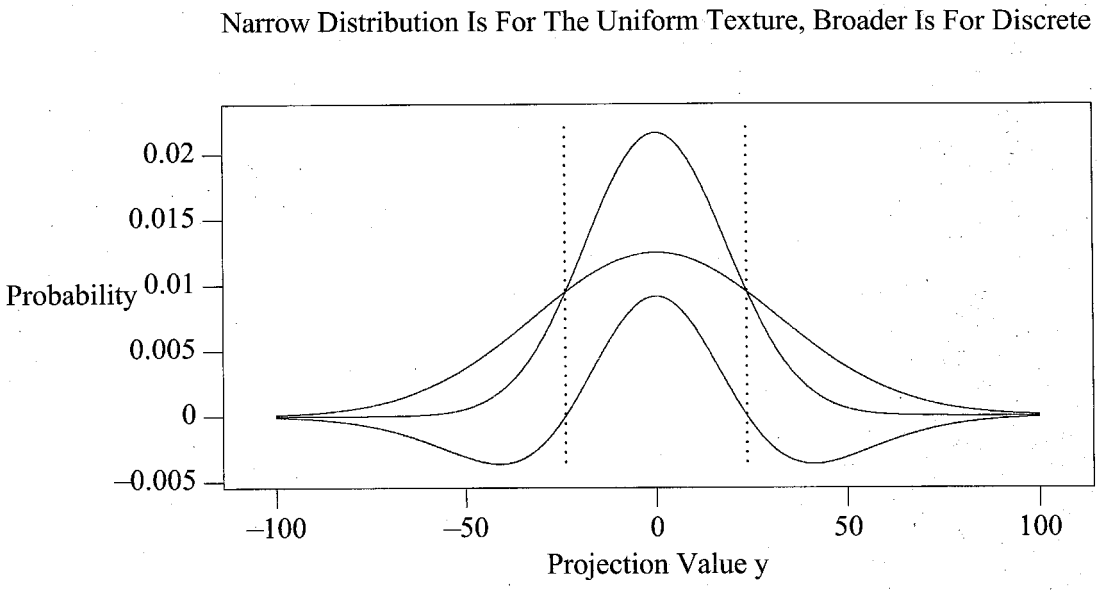


Figure 75. Sigmoid For Equal Mean & Different Variance Projection's

Neurons	11	51	101
Hard	0.329	0.174	0.094
Pseudo	0.185	0.029	0.004

TABLE 10. Hard And Pseudo-Hard Error Rates For The Textures

14.4 Lateral Inhibition And Second Order Moments

Lateral Inhibition implements a nonlinearity which complements other features of the network. Lateral inhibition allows us to map the two hypotheses's decision output pdf's for hard/soft/pseudo decoding, into a form which allows access to the second moment in a statistical test to be performed by a following layer. The specific form of lateral inhibition we shall first investigate is that which chooses the maximum absolute magnitude output of either

the projection **W** • **X** directly in hard decision operation, or the nonlinear sigmoid outputs f(**W** • **X**) in soft decision operation.

Lateral inhibition implements an *Order Statistic*.[24] [25] An order statistic is the probability density function for the following situation. Draw M samples from a zero mean density P[ζ] which is symmetric about zero. Take, in our case, the largest absolute value of this sample of M numbers. Repeat this experiment, and build a histogram of the results. The histogram will, in the limit of large numbers, have the distribution given by the equation below. On the right hand side of the equals sign, p(y) is the probability of a single absolute value sample. Thus, p(y) is twice the Gaussian density for x > 0, and zero for x < 0.

$$P^M(y) \equiv M\ p(y) \left\{ \int_0^y p(z)\ dz \right\}^{M-1}$$

$$where\ y \in (-\infty, \infty)$$

Equation 54. Maximum Order Statistic Formula

The maximum absolute order statistic probability density functions for the discrete and uniform pixel textures are seen below. The sample size order's calculated are 11, 51, and 101. The smaller mean order statistic corresponds to the smaller variance hypothesis, **U**. The larger mean order statistic corresponds to the larger variance hypothesis, **D**. The distributions for **U** and **D** are also shown for comparison purposes. Note that as the order increases, the mean of both **D** and **U** order statistics increases. This agrees with our intuition, since with a larger sample size, we are more likely to pick a larger magnitude point from either distribution.

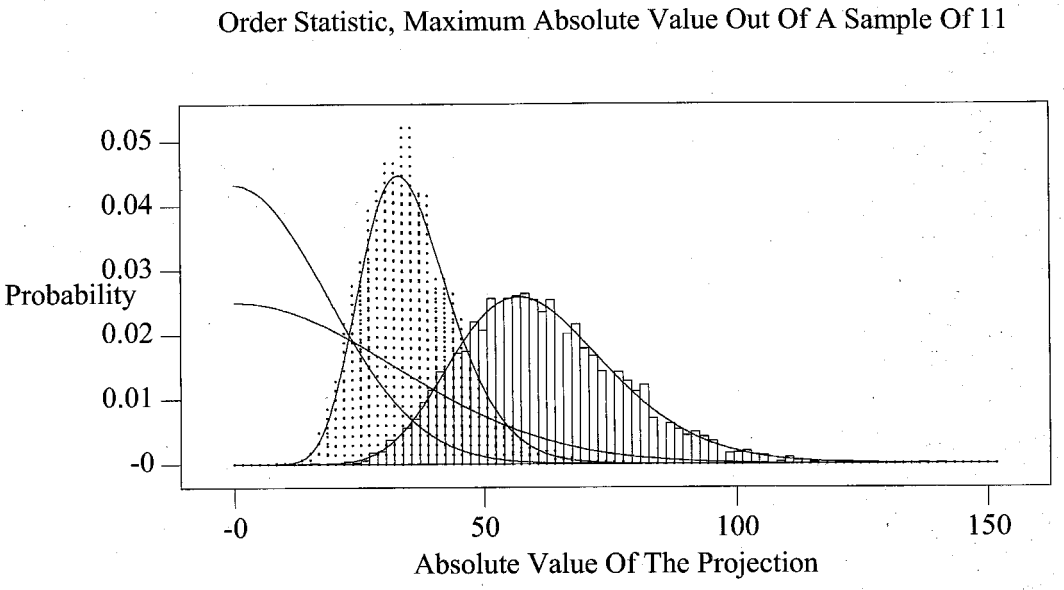


Figure 76. Order 11 Maximum Absolute Value Order Statistic

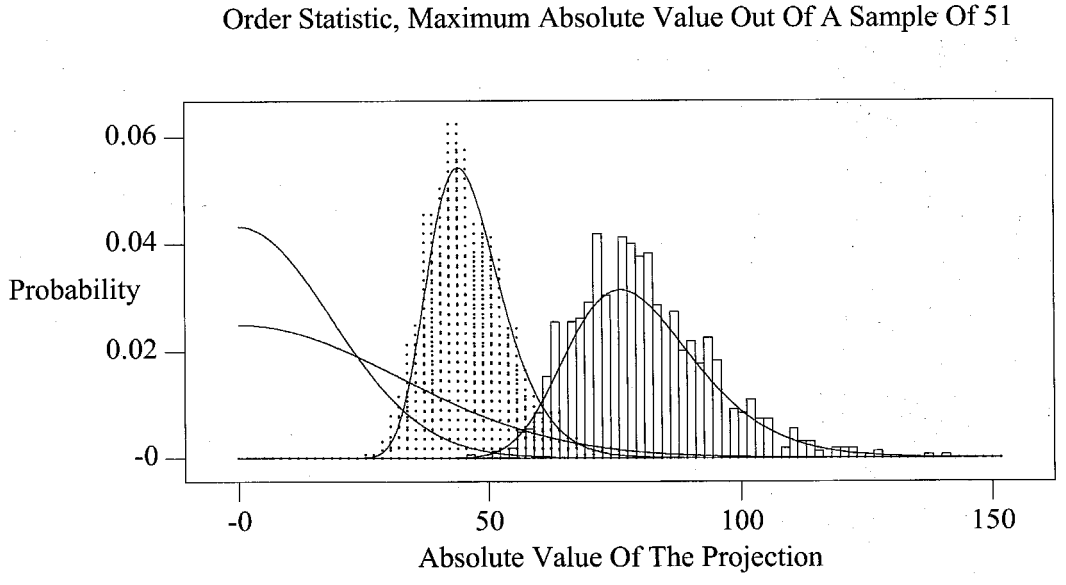


Figure 77. Order 51 Maximum Absolute Value Order Statistic

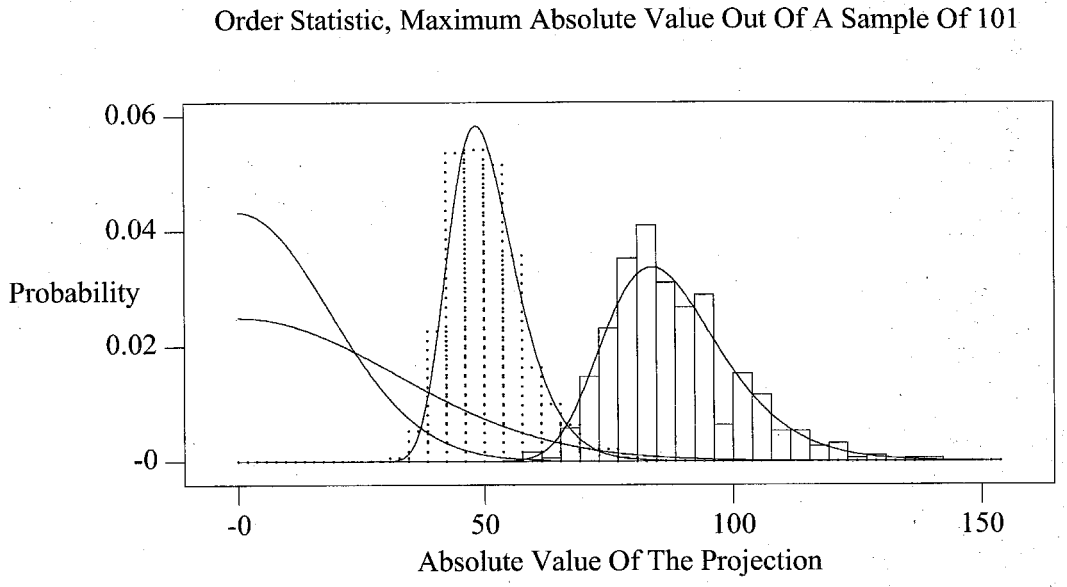


Figure 78. Order 101 Maximum Absolute Value Order Statistic

Discrete	Above	Below
11	0.165	0.835
51	0.044	0.956
101	0.021	0.979
Uniform	Above	Below
11	0.867	0.133
51	0.946	0.054
101	0.969	0.031

Order	Threshold	Perror
11	45.9	0.149
51	60.3	0.049
101	66.6	0.026

TABLE 11. Maximum Absolute Order Statistic Probability Of Error

MATLAB is used to demonstrate lateral inhibition. Above it was mentioned that the 1024 dimensional input space is isotropic, in that no one direction is preferred over another. We use this fact to create M weight vectors, and thus M neurons in a single layer. Generate M weight vectors, each with random $\pm$ 1 elements. The mean and variance of the projections onto each of these weight vectors are identical, yet the projections are statistically independent. Implement lateral inhibition by preserving that neuron output which has the largest magnitude of all M projections, and force the remaining M-1 neuron outputs to zero. This is a Winner Take All Network, where we are using the absolute neuron magnitude as the quantity we are *competing* among the set of neurons outputs. The order statistics above represent the probability density function of the winner of the lateral inhibition. MATLAB is used to generate 50,000 projections for a series of discrete images, and 50,000 projections for uniform textures. Histograms were compiled by taking the maximum absolute value sample from successive samples of sizes of 11, 51, and 101. The agreement of the histograms with the theoretical density functions is excellent.

The maximum absolute order statistic maps two random variables with identical means, and different variances, into a pair of random variables with different means. The order statistic makes the distributions *higher moment* available for a succeeding layer to conduct a linear threshold test on. In this way, a distributions higher moments can be used in a statistical hypothesis test. Order statistics are also effective in separating distributions which have broad variances, and means very closely clustered. Recall that the soft non-linear sigmoids also serve to separate the conditional hypothesis densities. Lateral inhibition coupled with a non linear sigmoid together form a potent mechanism for *pulling apart* distributions so that a succeeding

layer can make a hypothesis test with only small tails spilling across the threshold.

Consider using an order statistic to do hypothesis testing. We input a texture for which a decision is desired. Using our weight vector set, generate a sample of size M, one for each neuron using one of the weight vectors. Take the maximum absolute value of the set of M outputs. Determine a threshold by calculating where the two hypothesis order statistics of size M intersect. If the maximum from the M sample set for the unknown texture exceeds the above threshold, decide the texture came from the broader variance pixel texture, and hence is the discrete texture. Conversely, if the maximum absolute projection lies below the threshold, conclude the texture is from the small variance, uniformly distributed pixel type.

An important fact to note from the order statistic plots is that the overlap of the tails diminishes as the order sample size increases. Thus, the probability of making an error decreases as the number of parallel neurons used to obtain the samples is increased.

14.5 Parity, Order Statistics, And Neural Networks

Consider the problem posed by a determination of the parity of a {0,1} vector. The K-Parity binary hypothesis problem is shown below.

$$Given \quad [a_1, a_2, a_3, \cdots, a_n], \qquad a_i \in 0, 1, \qquad \sum_{i=1}^{k} a_i = \begin{Bmatrix} H_0 : Even \\ H_1 : Odd \end{Bmatrix}$$

Equation 55. K - Parity Problem

The discrete probability distributions and moments about the mean for the even and odd hypotheses are below for various values of K. The solid lines are for the odd distributions, and the dashed lines are for the even distributions.

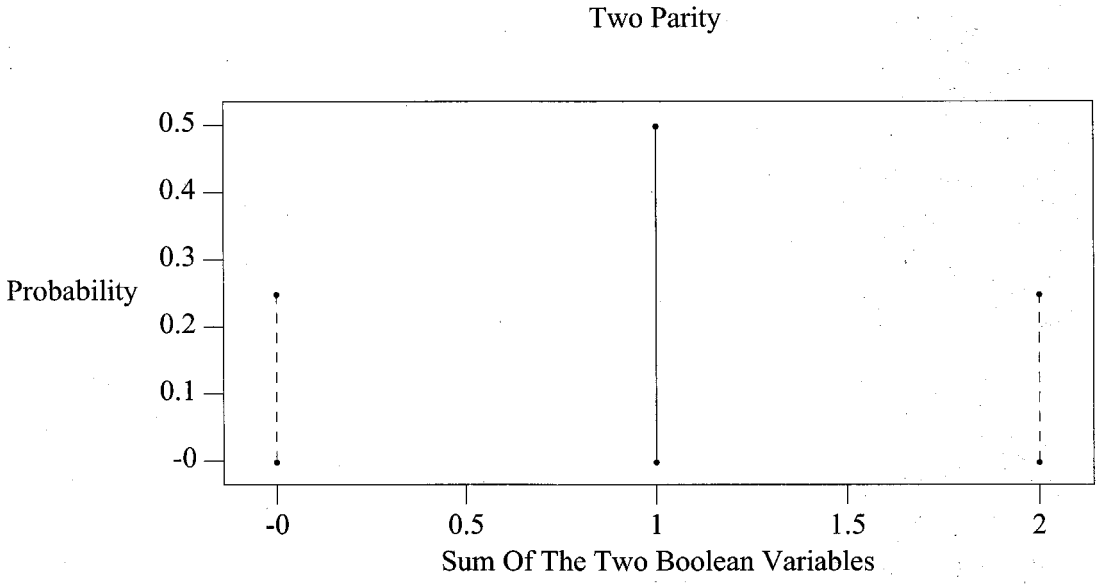


Figure 79. Two Parity

K	Hypothesis	Moment	Value
2	Even	1	1
2	Odd	1	1
2	Even	2	1
2	Odd	2	0

TABLE 12. 2 Parity Problem

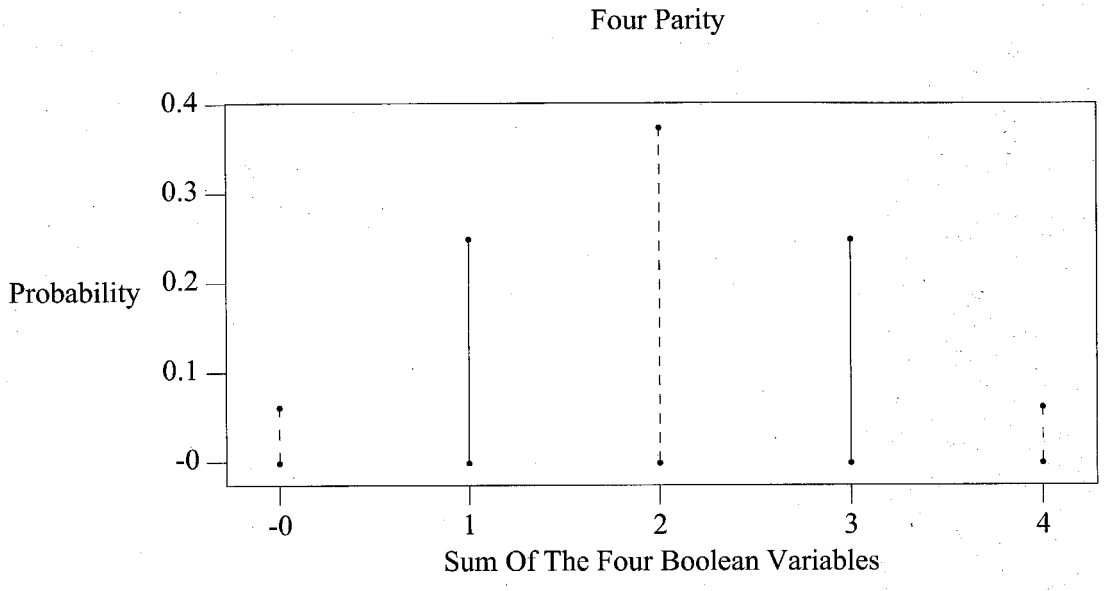


Figure 80. Four Parity

K	Hypothesis	Moment	Value
4	Even	1	2
4	Odd	1	2
4	Even	2	1
4	Odd	2	1
4	Even	3	0
4	Odd	3	0
4	Even	4	4
4	Odd	4	1

TABLE 13. 4 Parity Problem

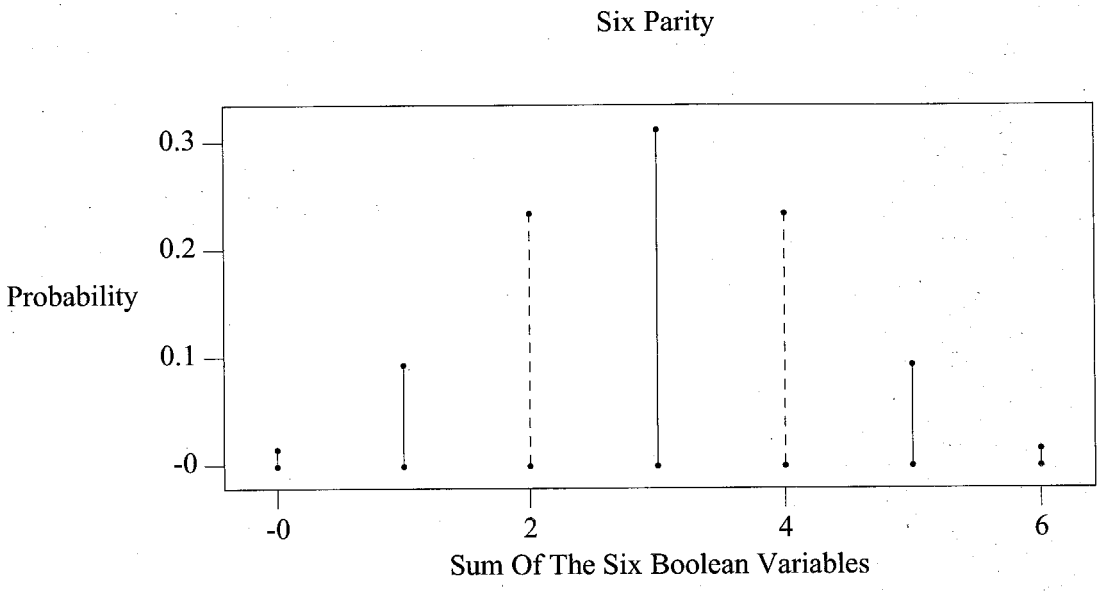


Figure 81. Six Parity

K	Hypothesis	Moment	Value
6	Even	1	3
6	Odd	1	3
6	Even	2	1.5
6	Odd	2	1.5
6	Even	3	0
6	Odd	3	0
6	Even	4	6
6	Odd	4	6
6	Even	5	0
6	Odd	5	0
6	Even	6	46.5
6	Odd	6	24

TABLE 14. 6 Parity Problem

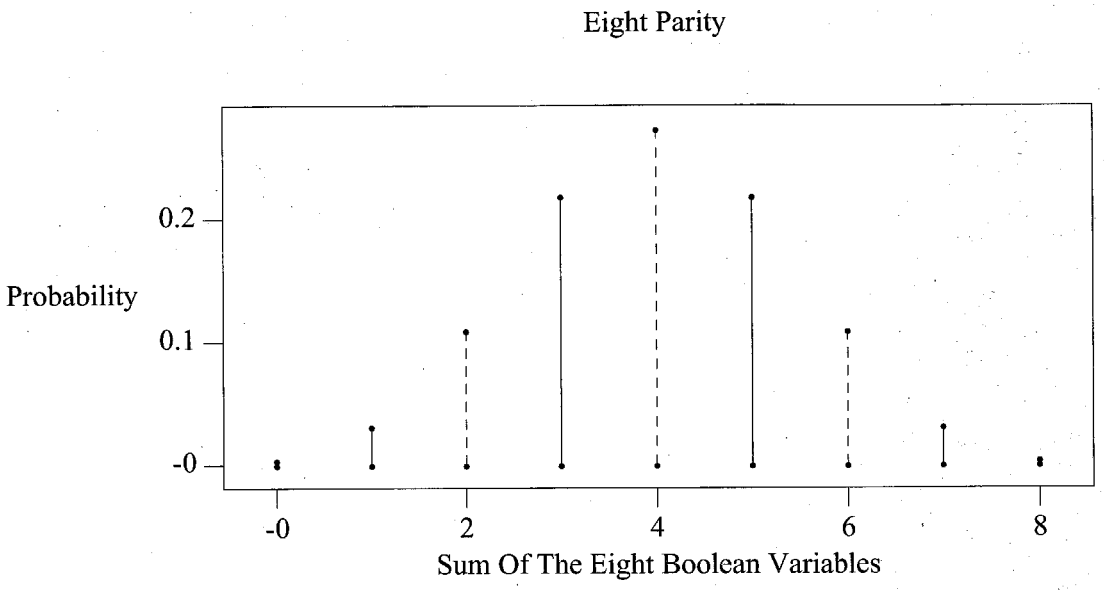


Figure 82. Eight Parity

K	Hypothesis	Moment	Value
8	Even	1	4
8	Odd	1	4
8	Even	2	2
8	Odd	2	2
8	Even	3	0
8	Odd	3	0
8	Even	4	11
8	Odd	4	11
8	Even	5	0
8	Odd	5	0
8	Even	6	92
8	Odd	6	92
8	Even	7	0
8	Odd	7	0
8	Even	8	1136
8	Odd	8	821

TABLE 15. 8 Parity Problem

Figure 83. Moment Tables And Discrete Density Functions For K-Parity

An interesting observation is that the first moment for which the even and odd random variables differ is the K'th. By recursively applying the order statistic operation, each lateral inhibition equipped layer strips away at least one moment from the characteristic function expansion of the two conditional hypotheses random variables $P[y|H_1]$ and $P[y|H_0]$. Thus the depth of a network with lateral inhibition equipped layers, is related to the depth in terms of moments about the mean, of the two conditional input probability density functions, that a network can use in a decision test. By incorporating successive lateral inhibition layers, a network can theoretically recursively strip away moments from the conditional probability density functions until a separation in means occurs. A linear decision test could then be applied to arrive at a

hypothesis decision.

14.6 Parity, Memorization, And Generalization

Another perspective on parity is that the problem is trivially solved using lateral inhibition in the following manner. For every possible *even* permutation $\in \{ 0 , 1 \}^K$, form a weight vector, thereby creating the set $\{\mathbf{W}_{even}\}$. For every unknown input, take Max $|| y_j \equiv \mathbf{W}_j \bullet \mathbf{X} ||$, $\mathbf{W}_j \in \{\mathbf{W}_{even}\}$. If the maximum is equal to K, the number of bits in the parity problem, choose H_{even}. Otherwise choose H_{odd}. What this does is pass the unknown image through a set of filters tuned to all possible even patterns. No attenuation indicates an even pattern. Attenuation indicates an odd pattern. This is brute force behavior. Every input is correctly classified by our lateral inhibition structure, but at the cost of using $2^{\frac{K}{2}}$ neurons in the competition layer ! The network has attained this performance by *Memorization*, not *Generalization*. Now decrease the number of neurons in the hidden, lateral, layer. As the number of neurons L in the competition layer are decreased, the probability that an incoming even image will match one of the competition neurons is $\frac{L}{2^{\frac{K}{2}}}$, yielding a net P_{error} of $\frac{1}{2} - \frac{L}{2^{\frac{K}{2}-1}}$. Thus, the number of neurons needed to be able to have reasonable P_{error} performance is always on the order of 2^K. That is, the network does not generalize, it memorizes.

14.7 Generalization Versus Memorization

To explain our situation above, recall the Bias versus Variance issue in statistical modeling.[26] We have implicitly assumed a statistical model which emphasizes the lower moments. Higher moments beyond the mean and variance, such as the skewness and kurtosis, are not easily extracted and used in a statistical hypothesis test. This is the price we pay and the tradeoff for our central limit theorem and other approximations. The poor performance of neural networks on purely combinatorial problems can perhaps be attributed to the fact that these problems, when viewed from the statistical perspective of densities conditioned on the hypothesis, differ in high moments. Several layers, often wide layers, are needed to access these higher moments. Problems which have significant information in lower moments, such as handwritten character recognition, result in good generalization with only a few neurons. For example, recall how well a single neuron does with the 32 by 32 handwritten digits of *Six* versus *Seven*. Here the probability of error was 0.003 ! More about this moment view will be said in the section below which develops the information theory and channel model of neural computation. However, it should be noted here that this neural stochastic technique does not work well with problems where the information a hypothesis test needs is buried in the higher conditional distribution moments.

15. Order Statistics Versus Hard And Soft Decoding

In this section, we shall look at the performance of the maximum order statistic, and make comparisons to hard and soft decision decoding. We consider two cases. In the first, the

projections pdf's have means at ±1, and a variance of 1. The second case also has means at ± and a variance of 250. As the plots below show, for both cases, the order statistic method of arriving at a hypothesis decision is definitely inferior to the hard and soft decision approach.

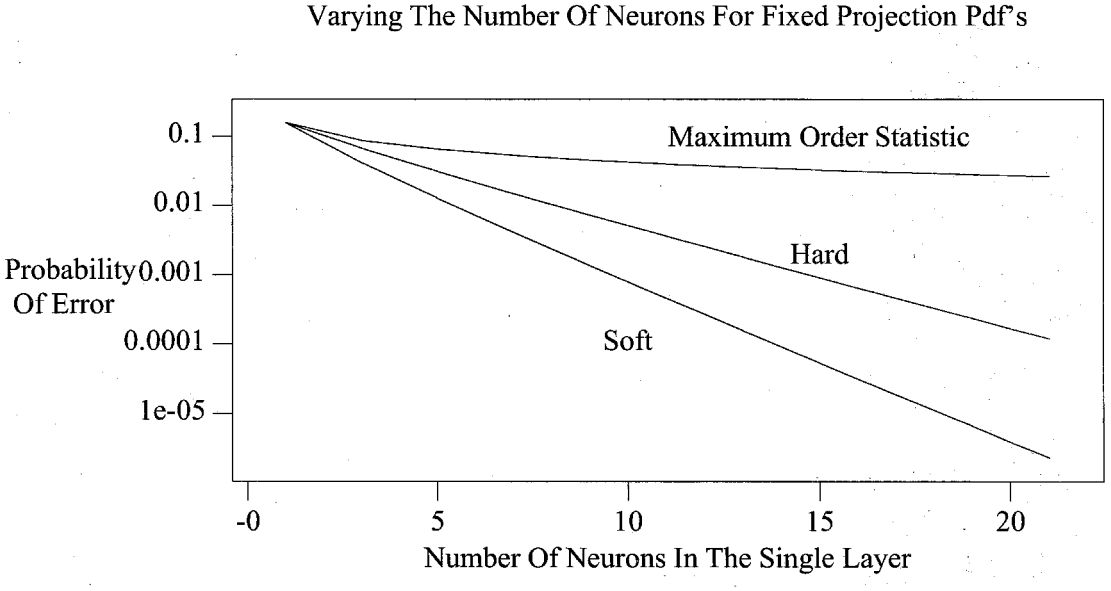


Figure 84. Varying The Number Of Neurons For Fixed Mean 1, Variance 1, Projections

Even when the variance is increased below to 250, but with μ still ±1, order statistic performance more closely approximates hard decision detection, but is still inferior.

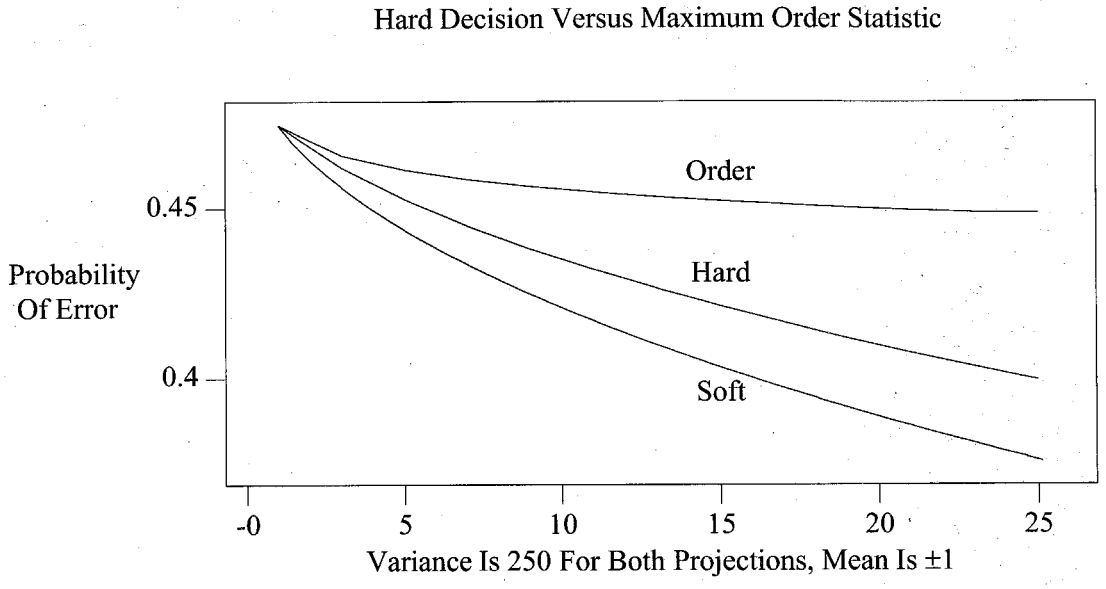


Figure 85. Hard Decision Versus Maximum Order Statistic

The conclusion to be drawn from these two plots is that when there exists a separation of the conditional projection means, there are better ways of implementing a low P_{error} decision test than via the calculation of a maximum order statistic. However, as can be seen from the plots of the digit **6** and **7** image projections above, typically we have neurons with conditional means which are clustered closely together, and have differing variances as well. For neurons such as these, order statistics outperform hard decision decoding, since the order statistic can take advantage of differences in both mean and variance between the projection random variables, whereas the hard decision threshold test uses only differences in the means. To demonstrate this, consider the two pdf's below. They are Gaussians with mean -0.1, variance 1, and mean 0.1, variance 2.

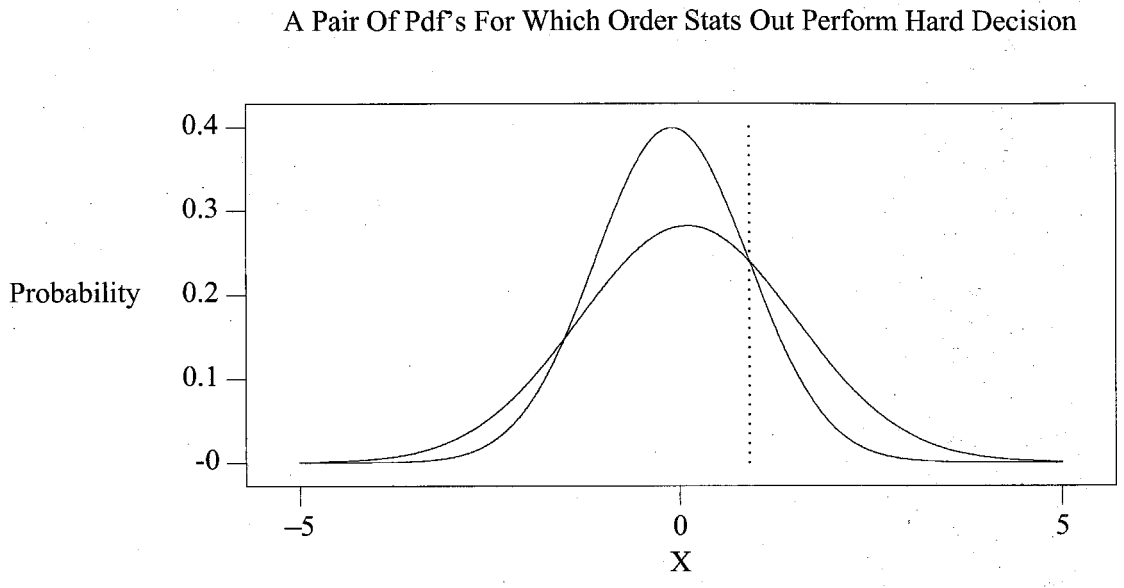


Figure 86. Order Statistic Yields The Better Performance

The eleven neuron order statistic is shown below.

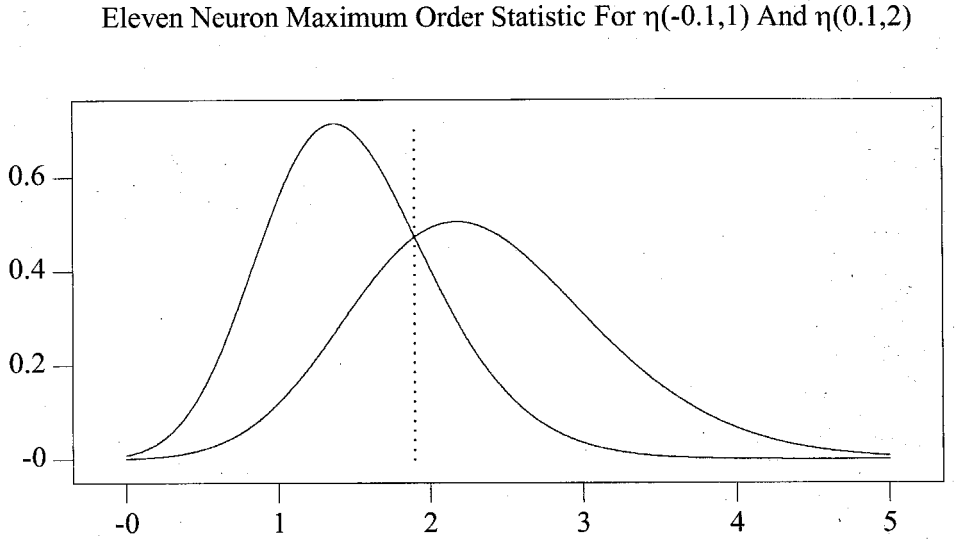


Figure 87. Eleven Neuron Maximum Order Statistic

The probability of error for the thresholded single neuron is 0.436. The resulting eleven neuron hard decision p_{error} is 0.332. The maximum order statistic of size eleven has a threshold at 1.90, and a resulting p_{error} of 0.266. Thus, the order statistic outperforms the hard decision case. This had to happen at some point as the means approached each other, and the variance remained different. The issue is whether it is important. We look at the four neurons besides the Principle component for the Six and Seven images considered above. We in fact have the situation we alluded to in the above analysis. The means and variances of the projections are close enough for the Tanh approximation to marginally hold, and the pdf's are clustered tightly enough to the origin that an order statistic approach is more appropriate than a hard decision rule.

16. Recurrent Networks

A recurrent network is one that feeds back onto itself. An example is shown in the figure below.

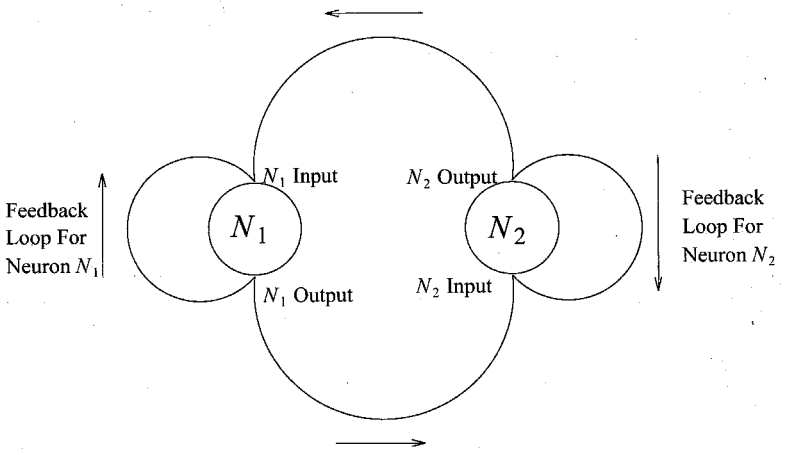


Figure 88. Two Neuron Recurrent Neural Network

16.1 Recurrent Neural Networks

In the network above, each neuron looks at its own value, and that of its neighbor, and produces an output. As discussed previously, this output is $z_i = f(\mathbf{W}_i \bullet \mathbf{X} - T_i)$. For our initial discussion of recurrent networks, we shall assume the value of each neuron is discrete, and either on or off, ± 1. The weight values can be any real number. Thus, z will be thresholded, and a {-1,1} output decision obtained. The feedback structure shown above implements an associative memory. If the state of each neuron is initially set to some pattern, then dynamical updates will occur either indefinitely, or until a fixed point is reached. If a fixed point is reached, then no further changes will occur to any neurons values, and the values of the set of neurons

will remain stabilized. Mathematically, a fixed point is a set of neuron values such that $\mathbf{X} = \{ x_i \}$, and $x_i = f_i(\mathbf{W}_i \bullet \mathbf{X} - T_i)$. A set of desired fixed points is encoded into the network via the values of the weights, thresholds, and sigmoid functions. As an example, consider a three neuron recurrent network. We wish to encode two fixed points into the system, A={1,-1,1} and B={1,1,-1}. First, we shall conduct a brute force analysis. We list all 2^3 or 8 system states. These are seen below.

System State	Hamming From A	Hamming From B	Neuron One	Neuron Two	Neuron Three
{ 1, 1, 1}	1	1	1/dc	dc	dc
{ 1, 1,-1}	2	0	1	1	-1
{ 1,-1, 1}	0	2	1	-1	1
{ 1,-1,-1}	1	1	1/dc	dc	dc
{-1, 1, 1}	2	2	1/dc	dc	dc
{-1, 1,-1}	3	1	1	-1	1
{-1,-1, 1}	1	3	1	1	-1
{-1,-1,-1}	2	2	1/dc	dc	dc

TABLE 16. Three Neuron Recurrent Network With Two Fixed Points

The columns labeled neuron one, two, and three are what the value of these neurons should be if the minimum Hamming distance is to be reduced at every iteration or system update interval. A dc above means "Don't Care", and indicates the current system state is equidistant from both stable states. We could "go either way" with equal probability. Below, we implement a "when in doubt, choose stable state A rule, and set any N_1 dc to 1, any N_2 dc to -1, and any N_3 dc to 1.

System State	Hamming From A	Hamming From B	Transitions To A Stable Point
c = { 1, 1, 1}	1	1	c ☞ A
B = { 1, 1,-1}	2	0	B ☞ B
A = { 1,-1, 1}	0	2	A ☞ A
d = { 1,-1,-1}	1	1	d ☞ A
e = {-1, 1, 1}	2	2	e ☞ A
f = {-1, 1,-1}	3	1	f ☞ B
g = {-1,-1, 1}	1	3	g ☞ A
h = {-1,-1,-1}	2	2	h ☞ A

TABLE 17. Dynamics Of The Three Neuron Recurrent Network

As can be seen, each system state reaches a stable state { A or B } in one step or iteration. This typically doesn't occur in larger size networks. When N is greater than 3, often more than one step is required for the system to reach a stable state. In this case, the system state passes through several intermediate states on the way to a stable state. An important feature of network design is that the cluster of neurons only have specified stable states, and that every possible system state end up in one of these stable points. That is, the issues of stable points other than the desired or specified ones, and whether some initial system state will result in an endlessly cycling through a sequence of non-stable points, does not occur. A *deterministic* approach to whether a network suffers from these two problems is to form an energy function, or Hamiltonian, for the system. The Hamiltonian represents the energy the system has given that the system is in a certain state. The Hamiltonian is constructed so that it is bounded. That is, each state must have a finite energy value. In addition, the Hamiltonian is constrained so that Energy minima only occur at specified stable points. The network update or transition rule is then shown to always result in a reduction of system energy. Since this reduction is bounded,

and always greater than some quantity E_0 due to the finite number of possible system states, a stable point will be reached in a finite number of steps. [27]

In this research, a probabilistic approach to recurrent networks is espoused. The analogy to statistical mechanics is made, where convergence occurs "in the mean". That is, there exist many possible system permutations which can occur. However, probabilistically, given an initial state, one final state or configuration will occur with overwhelming preference. To better sketch out this concept, we appeal to our simple two fixed point system, along with our statistical update rule outlined in the feedforward discussion. Consider a recurrent network where the number of neurons is large, and each neuron is again either on or off. Let the two stable states be labeled A and B as above in the three neuron case. An example of such a system is shown in the figure below. Imagine the image pixels are either black {-1} or white {1}. A single neuron inputs the values of all pixels, and makes an "update" recommendation for a single pixel. Although only a single pixel neuron is shown, it is to be visualized that an identical update process is occurring for all pixels in the image. Imagine our two stable points represent images, say a digitized picture of a handwritten one **1** or zero **0**.

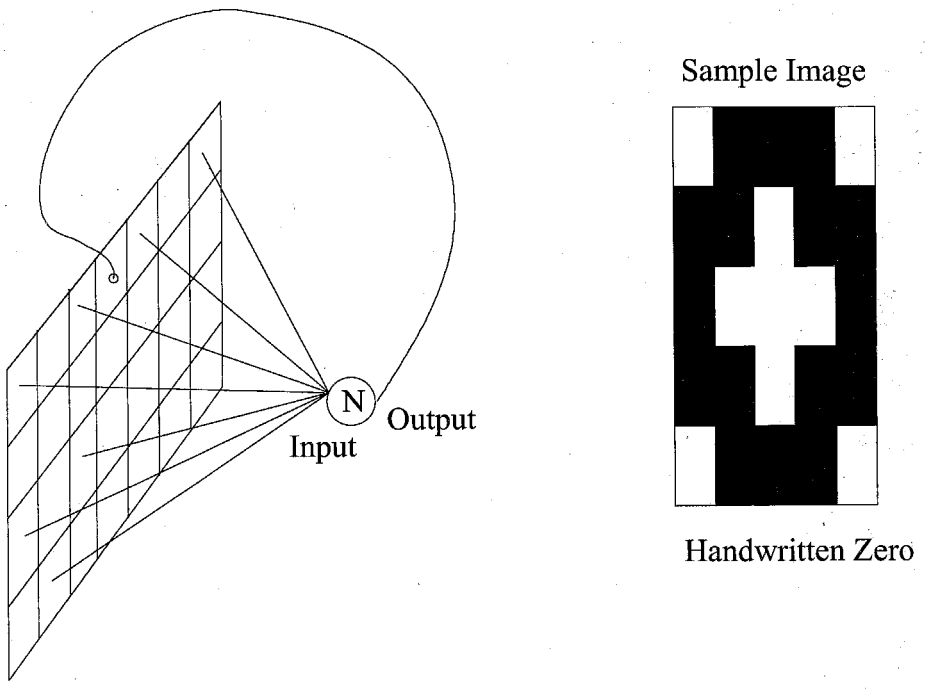


Figure 89. Multiple Pixel, One Neuron Per Pixel, Recurrent Neural Network

Construct a system state vector **X** which consists of the states of each pixel at any one time. Thus, our two stable states are $\mathbf{X}_A$ and $\mathbf{X}_B$. For neuron states which are both +1 or -1 for both patterns A and B, we set the corresponding pixel's neuron to have that value for all input patterns. For neurons which have differing values for the two patterns (eg : +1 for A, -1 for B), we use the feedforward PCA approach to construct a weight vector and threshold for that neuron. Let the weight vector be $\mathbf{W} \equiv \mathbf{X}_A - \mathbf{X}_B$. This is because, for two class data, the Principle Component of the class is that class vector. Thus, the PCA vector for $\mathbf{X}_A$ is $\mathbf{X}_A$ itself, and similarly for $\mathbf{X}_B$. As in the feedforward case, we consider each pixel a random variable, and hence the projection $\mathbf{W} \bullet \mathbf{X}$ is a one-dimensional random variable. Let us now calculate the pdf

of $\mathbf{W} \bullet \mathbf{X}$..

When an image *close* to image A is seen, $\mathbf{W} \bullet \mathbf{X}$ will be positive, and we assign the pixel shade which corresponds to image A at that pixel location. Correspondingly, if an image close to B is seen, than $\mathbf{W} \bullet \mathbf{X}$ is negative, and we assign to the pixel the value corresponding to that pixel location on image B. Implicitly, we have assumed that vectors A and B are statistically independent, so that $\mathbf{A} \bullet \mathbf{B}$ is typically very close to zero. If there is a perfect match, in that $\mathbf{X} = \mathbf{A}$, then $\mathbf{W} \bullet \mathbf{X}$ will be +N, where N is the number of neurons present, and hence the length of $\mathbf{X}$. Similarly, if $\mathbf{X} = \mathbf{B}$, then $\mathbf{W} \bullet \mathbf{X}$ is -N. Let us consider images close to $\mathbf{A}$. Suppose we have a vector $\mathbf{X}'$ which differs from $\mathbf{X}_A$ in m places. Then the Hamming distance between $\mathbf{X}_A$ and $\mathbf{X}'$ is m. The expected value of $\mathbf{W} \bullet \mathbf{X}$ is the expected value of $\mathbf{X}_A \bullet \mathbf{X}' - \mathbf{X}_B \bullet \mathbf{X}'$. By supposition, the expected value of $\mathbf{X}_A \bullet \mathbf{X}'$ is N - 2m. Assuming the pattern vectors $\mathbf{X}_A$ and $\mathbf{X}_B$ are statistically independent, the expected value of $\mathbf{X}_B \bullet \mathbf{X}'$ is equivalent to the $\mathbf{X}_B \bullet E[\mathbf{X}']$. But each pixel in $\mathbf{X}'$ is equiprobable ± 1, so $E[\mathbf{X}']$ is zero, and thus the expected value of $\mathbf{X}_B \bullet \mathbf{X}'$, given that the vector $\mathbf{X}'$ is a Hamming distance m from $\mathbf{X}_A$, and m << N, is zero. Thus, the mean of the projection $\mathbf{W} \bullet \mathbf{X}'$ is N - 2 m. The factor of 2 occurs since, there are N - m perfect bit matches, and m bit mismatches. Thus, the dot product of the two is (N - m) * (1) + m * (-1), or N - 2 m. The variance of the projection is the variance induced by the fluctuating possibilities of the projection of $\mathbf{X}'$ onto $\mathbf{X}_B$. Thus the variance of $\mathbf{W} \bullet \mathbf{X}'$ about the mean N - 2 m is the noise injected by the projection of $\mathbf{X}'$ onto $\mathbf{X}_B$. Since the N bit values of $\mathbf{X}_B$ are assumed independent of each other, the variance is the sum of N independent random variables ϕ, each identically distributed. ϕ is equiprobable of being 1 or -1. Thus, the expected value of ϕ is zero, and the variance of ϕ is

$$\frac{(1-0)^2}{2} + \frac{(-1-0)^2}{2},$$

or 1. Thus the variance of $\mathbf{W} \bullet \mathbf{X}$ is N. This means that the projection of a N bit vector $\mathbf{X}'$ a Hamming distance m from $\mathbf{X}_A$ is normally distributed with mean N - 2 m, and variance N. (Note that implicitly, we have used the normal approximation to the binomial distribution. A detailed discussion of this approximation is in Feller, Chapter VII, *The Normal Approximation To The Binomial Distribution.*)[28] This characterization of the pdf of $\mathbf{W} \bullet \mathbf{X}'$ will allow a stochastic perspective derivation of the two stability properties of recurrent networks discussed above.

The update rule of each neuron modifying a pixel occurs in parallel, and is independent of the change in all other pixels. If the initial seed vector $\mathbf{X}'$ is a Hamming distance m from $\mathbf{X}_A$, convergence to the fixed point $\mathbf{X}_A$ will occur if the Hamming distance of the iterated vector $\mathbf{X}$ is less than m. However, due to the independence assumption of the updates, this is equivalent to requiring that the update occur correctly with probability better than $\frac{m}{N}$. The pdf of the projection of $\mathbf{W} \bullet \mathbf{X}'$ was found above, and due to the symmetry of the situation, a correct bit update will occur with probability equal to the integral below.

$$\frac{1}{\sqrt{2\pi N}} \int_0^{\infty} \exp^{-\frac{(x-N+2m)^2}{2N}} dx \geq 1 - \frac{m}{N}$$

$$\equiv \quad Erf\left[\frac{N-2m}{\sqrt{2N}} \right] \quad \geq 1 - \frac{2\,m}{N}$$

Equation 56. Two Memory Error Function Integral Capacity Inequality

The integral above is the probability that any one bit is chosen correctly on an update. Thus, the integral above must be greater than $\frac{N-m}{N}$ or 1 - $\frac{m}{N}$. This inequality relationship yields the attraction radius, or basin of attraction size, for the two memory associative memory. Below is plotted the probability of a bit being correct on the horizontal axis (1 - $\frac{m}{N}$), and the probability of an updated bit being correct on the vertical axis (1 - $\frac{m}{N}$). Curves are shown for N equal to 25, 100, 200, and 300 below. The steeper the curve, the higher the N. The straight line is the Output equals Input curve. Thus, convergence occurs when the update curve lies above the straight line.

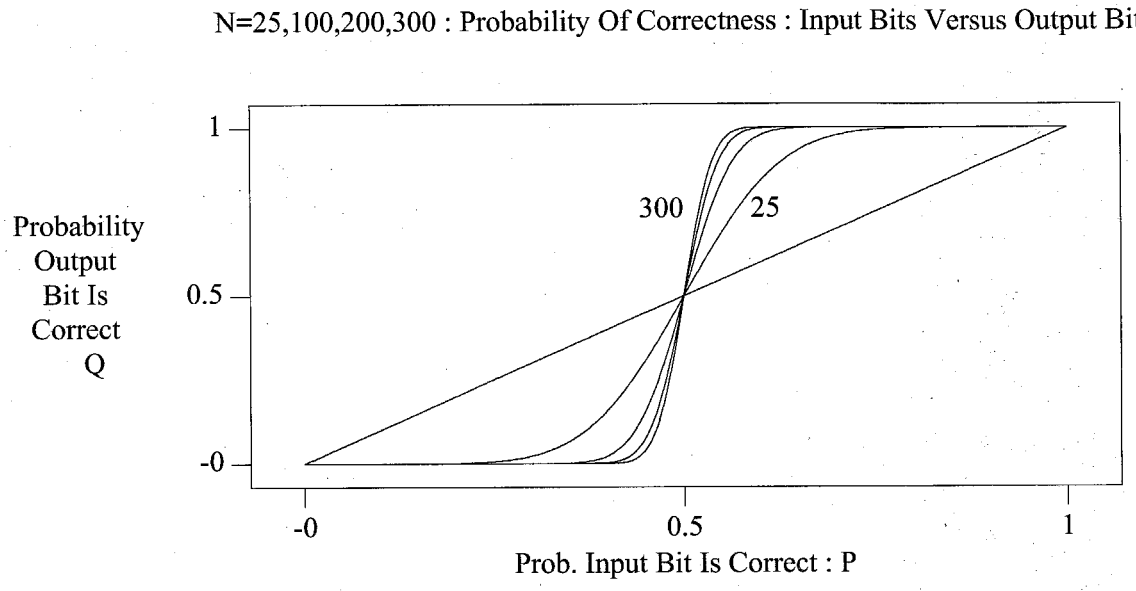


Figure 90. N = 25,100,200,300 : Probability Of Correctness : Input Bits Versus Output Bits

The markers below were obtained using a simulation in MATLAB. First, two fixed point vectors, $\mathbf{X}_A$ and $\mathbf{X}_B$ were randomly generated. Then for each data point, 50000 "nearby" vectors were generated which had a single bit probability P of being equal to the same bit value for that location in the fixed point vector $\mathbf{X}_A$. The update rule described above was implemented with the neuron weight **W** set equal to $\mathbf{X}_A - \mathbf{X}_B$. The output probability Q of a single bit being correctly matched to the fixed point vector $\mathbf{X}_A$ was then calculated upon the update. The 50000 sample average was then plotted. This process was repeated for each bullet marker shown. As can be seen, the simulation data points and the theoretical curves agree fairly well. The number of bits in error initially is the probability a single bit is in error, times the dimension of the system. The corresponding Hamming distance is m, or N (1 - Probability Of A Correct Bit).

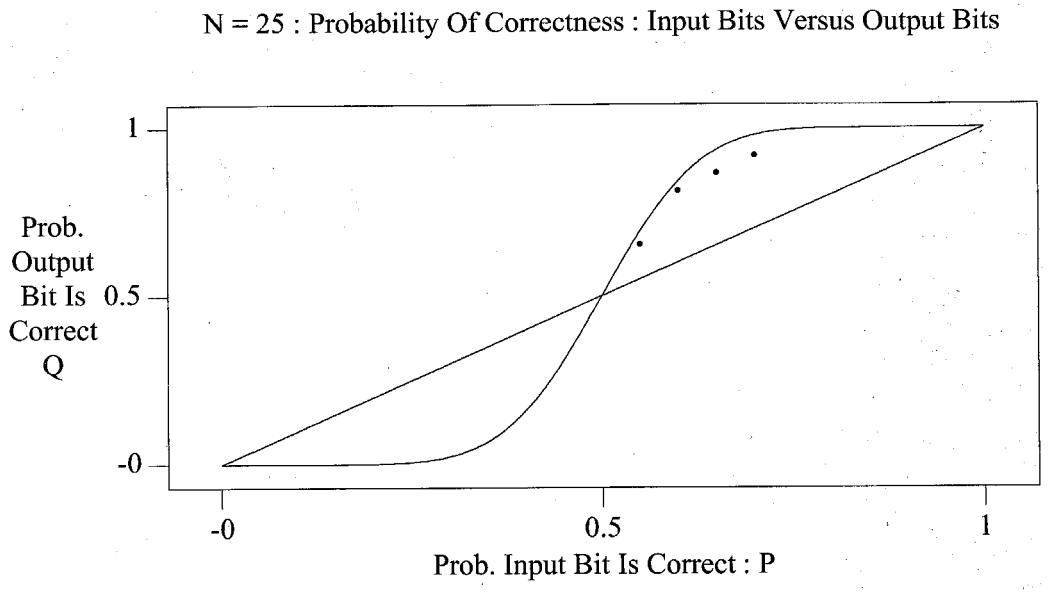


Figure 91. N = 25 : Probability Of Correctness : Input Bits Versus Output Bits

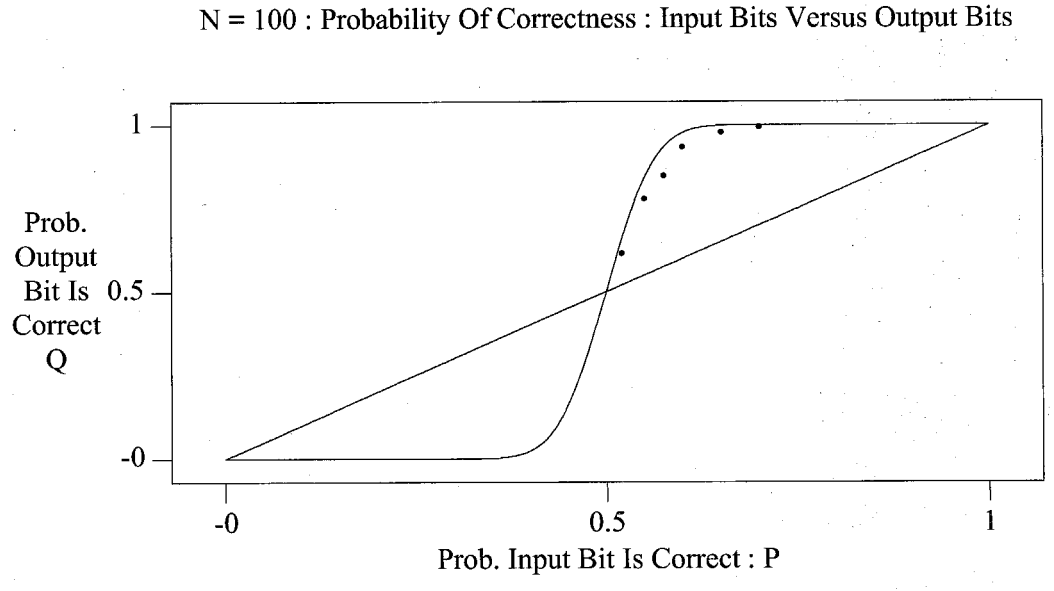


Figure 92. N = 300 : Probability Of Correctness : Input Bits Versus Output Bits

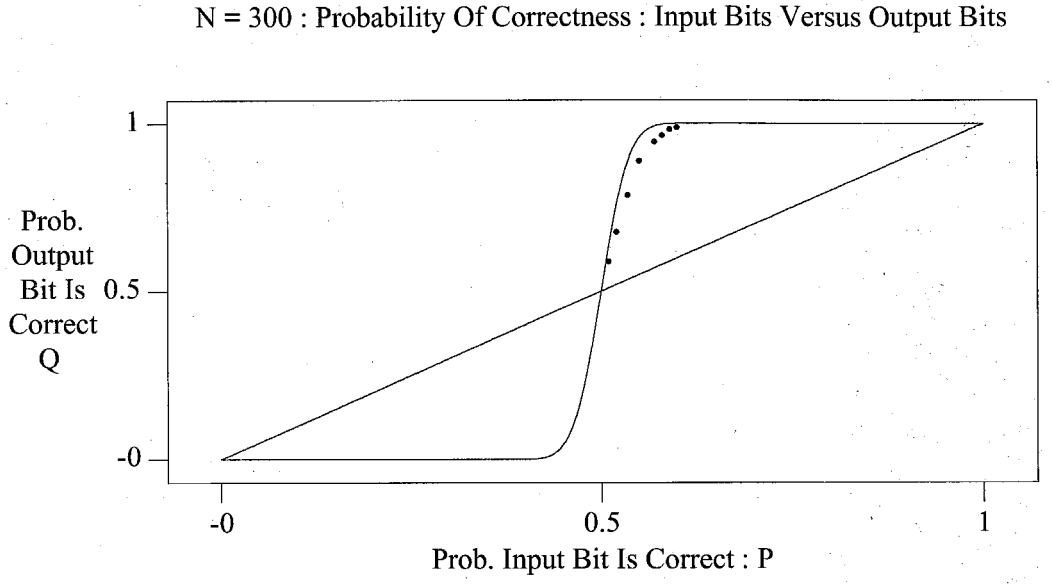


Figure 93. N = 300 : Probability Of Correctness : Input Bits Versus Output Bits

It is important to note that when m is ½ N, then $\frac{1 - 2m}{N}$ is zero, and Erf(0) is 0, so both the left-hand side and right-hand side above are equal to 0. This coincides with our intuition. When there are only two fixed points, and we are less than ½ N Hamming distance from one of them, we should spiral into that fixed point. This is what these two memory curves are telling us. Convergence occurs in the mean when we are less than ½ N bits away from one of the stable points. Since we are always within ½ N bits from one of the two memories, we always converge to one of the two stable points. Indeed, the update rule is in retrospect somewhat silly, since we are simply taking the projection of the input state vector **X** onto the two fixed points, and assigning the output state vector to be that corresponding to the fixed point with the largest projection. Nonlinear sigmoids are not used, as these above were shown to aid the merging of

multineuron information. Each pixel is updated by a single neuron, so no merging occurs, and non-linearities do not help here. Hence, we apparently have shown the obvious : an implementation of a transition matrix for a finite state machine. The utility of the above formalism lies in what happens when there are more than two memories or fixed points.

The extension of the associative memory architecture to handle more than two fixed points is done as follows. Consider a set of fixed points $\mathbf{X}_j, j \in 1,2,3,...M$. For each neuron, we divide the fixed point vectors into two sets, which we call $\mathbf{X}_+$ and $\mathbf{X}_-$, according to whether $\mathbf{X}_j$ has a ± 1 value for the pixel the neuron is assigned to update. Our Principle Components Analysis algorithm is run, yielding two weights, $\mathbf{W}_+$ and $\mathbf{W}_-$, and the corresponding weight vector is constructed as their difference, $\mathbf{W}$ is $\mathbf{W}_+ - \mathbf{W}_-$. This construction is done for every neuron. Note that the partition of the $\mathbf{X}_j$ into the two sets $\mathbf{X}_+$ and $\mathbf{X}_-$ will change from neuron to neuron, and so every neuron will have a different weight vector. Also note that a sigmoid is not used.

For multiple memories, and for each pixel, we have a binary hypothesis : Given the input vector, should this pixel be +1 or -1 ? For each bit, we assemble the training set S_+ and S_-. We place into one set or the other, a desired fixed point vector depending on whether the sign of the fixed point's pixel is +1 or -1 at the pixel location being updated. Do principle components analysis independently on these two training sets, and obtain the largest eigenvalue eigenvector $\mathbf{W}_+$ and $\mathbf{W}_-$ for the two sets S_+ and S_-. The weight for the corresponding neuron will then be $\mathbf{W} = \mathbf{W}_+ - \mathbf{W}_-$. The threshold is set to zero, and a binary test is made at each iteration. Now convergence for m memories does not occur in a single step as seen for two memory systems.

Below are seen a few convergence runs for N = 1200.

	State 1	State 2	State 3	State 4	State 5	State 6
State 1	0	588	590	595	577	588
State 2	588	0	578	595	573	584
State 3	590	578	0	595	597	616
State 4	595	595	595	0	550	587
State 5	577	573	597	550	0	595
State 6	588	584	616	587	595	0

TABLE 18. Hamming Separation For The Six Fixed Points For the Results Below

	State 1	**State 2**	State 3	State 4	State 5	State 6
Initial	585	**581**	611	594	594	599
Step 1	376	**212**	790	509	499	578
Step 2	410	**178**	756	543	465	544
Step 3	476	**112**	690	547	461	548
Step 4	588	**0**	578	595	573	584

TABLE 19. Recurrent Convergence Example

In the above simulation, the weight vector for each neuron was constructed as the sum below.

$$\mathbf{W}(k) \equiv \sum_{i=1}^{M} Sign\left[\mathbf{X}_i(k) \right] \mathbf{X}_i$$

Equation 57. Outer Product Weight Vector Construction

This is a valid approximation for the principle component or primary eigenvector due to the mutual orthogonality of the fixed point vectors and the relative sparseness of the fixed points with respect to the dimension. That is, L, the number of fixed points, is much less than 2^N, the

total of states or configurations possible. Here N is the dimension of **X**, and is the number of neurons present.

The principle component vectors $\mathbf{W}_+$ and $\mathbf{W}_-$, in the large N, small L limit, are approximately the average of the sets S_+ and S_- described above. However, if the vectors in S_+ and S_- are statistically independent, then the averaged weight vector will approach zero as N increases. That is, averaging over larger and larger sets will produce a smoothed vector. Since we are taking the difference of $\mathbf{W}_+$ and $\mathbf{W}_-$, and both of these weight PCA vectors are approaching zero as N increases, our ability to pick useful information off of an initial "seed" vector **X** decreases as N increases. It is useful to keep this in mind during the analytic discussion below.

The analysis for multiple memories is similar to the analysis for the two memory case. Consider what happens when the multiple memory weight vector is constructed as outlined above. Let the number of fixed points be L. We load one of the fixed points into the network. Since the fixed point vector matches ONE of the vectors in the sum from which **W** was derived, the signal component of the projection is still N, as derived above. However, there are now L - 1 noise terms. Where the noise component before was $\sigma^2 = N$, now we have L - 1 independent noise terms, all with variance $\sigma^2 \equiv N$. The total variance is thus (L - 1) N. Our analytical approach for the binary situation still holds. Using it, we obtain the multiple memory relationship seen below. Again, m is the number of incorrect bits in vector of length N. Hence, m is the Hamming distance from one of the system's fixed point vectors. L is the total number of system fixed point vectors.

$$\frac{1}{\sqrt{2\,\pi\,(L-1)\,N}}\int_{0}^{\infty}\exp^{-\frac{(x-N+2\,m)^2}{2\,(L-1)\,N}}\,dx \;\geq\; 1-\frac{m}{N}$$

$$\equiv\; Erf\left[\frac{N-2\,m}{\sqrt{2\,(L-1)\,N}}\right] \;\geq\; 1-\frac{2\,m}{N}$$

Equation 58. Multiple Memory Error Function Integral Relationship

Thus, the addition of more fixed points presents itself as an added noise source for the iterative stochastic update algorithm. Furthermore, the additional noise introduced grows linearly with the number of additional fixed points. We use this equation to study what happens as fixed points are added.

Consider the system with one thousand neurons below.

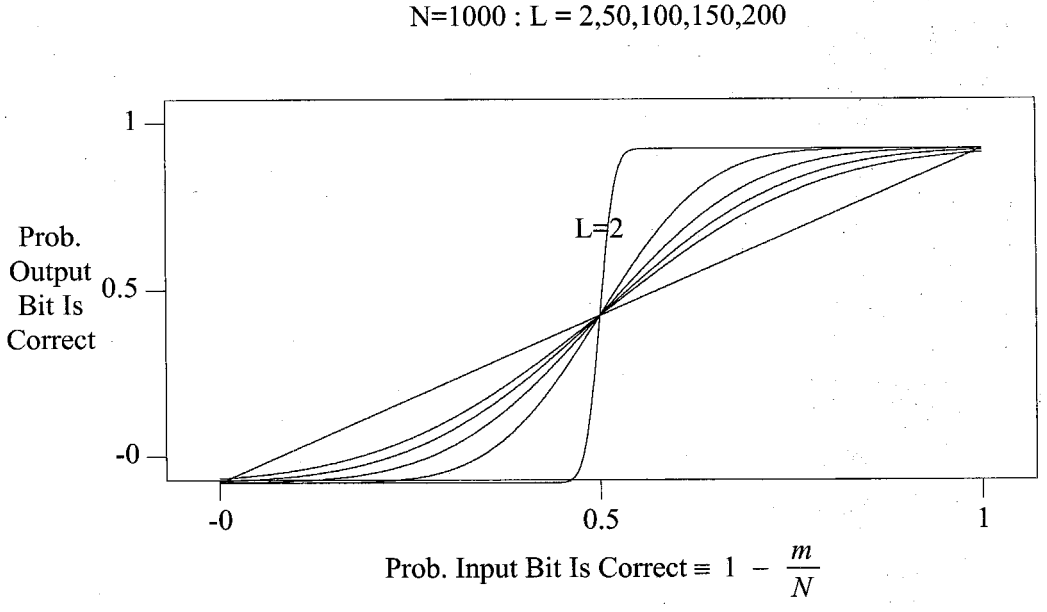


Figure 94. N = 1000 : Probability Of Correctness : Input Bits Versus Output Bits

The behavior for L greater than about 150 is interesting. In this region, the signal component of the projection has reached it's largest value possible, N. However, there is a significant tail under the threshold, even with the Hamming distance equal to zero, and the mean at N. This tail is due to the noise introduced by the large number of fixed points. Thus, even when the signal is "strong", the noise is also high, and the system makes significant errors. As the plot below shows, the noise tends to kick the system back away from a fixed point whenever convergence toward a fixed point seems imminent. Amplifying the region where the system seems to make the transition from stable basins of attraction into a more chaotic type of behavior, we obtain the plot below. The curves are the left-hand side Erf function of the last set of equations. The straight line represents the map $P_{error}(In) \equiv P_{error}(Out) \equiv 1 - \frac{m}{N}$.

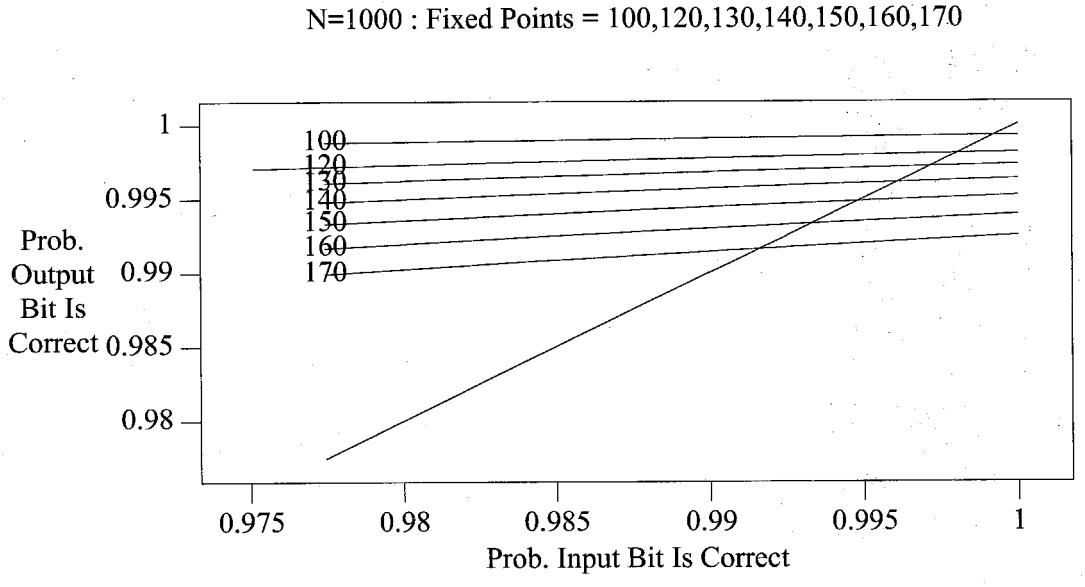


Figure 95. N = 1000 : L = 100, 120, 130, 140, 150, 160, 170

The intersection of the Erf curves with the linear equation marks the points where we encounter a transition in behavior. When the curve lies above the straight line, the system is converging to a fixed point. When the curve lies below the straight line, noise supersedes the fixed points attraction, and the system state moves away from the fixed point. Thus, as the system starts to become overloaded with memories, it responds by small oscillations about a fixed point vector. The system will attempt to stay in the neighborhood of where the Erf curve intersects the straight line. Here, there is a finite, non-zero probability that each updated vector will have bits in error. Which bits differ from the fixed point will randomly change on each iteration. Thus, the behavior will be seen as a chaotic type of motion about the fixed point, instead of convergence to the fixed point, and no dynamic motion afterwards. Exactly this type of chaotic behavior was seen by Hopfield and described in his 1982 paper.[29]

System convergence occurs when the system can reach the (1.0,1.0) point on our plots. However, our system is discretized, and consists of only N neurons, a finite number. Hence, we consider convergence as achieved when the intersection of the Erf curve with the linear line, is close to, but necessarily exactly equal to, 1.0. Define the stability transition point as the point where a vector is equally probable of being thrown away from the nearby fixed point as towards it. The probability the network makes the correct decision on all N bits is $(1 - P_e)^N$. We set this equal to ½. Since N is assumed large, in order for $(1 - P_e)^N$ to be equal to 0.5, the bit error probability P_e must be fairly small. Thus, the approximation $(1-P_e)^N \approx 1 - N P_e$ is valid. Setting the probability of correct recall to ½ yields $1 - N P_e \approx$ ½ or $P_e \approx \frac{1}{2N}$. Putting the error function equal to this cutoff point yields the desired equation.

Stochastic Computation

$$Erf\left[\frac{N-1}{\sqrt{2\,(L-1)\,N}}\right] \equiv \frac{N-1}{N}$$

Equation 59. Stability / Capacity Equation For Multiple Memory Recurrent Networks

As a check, we attempt to duplicate Hopfield's results on stability from his 1982 paper. In that paper, using a network with 100 neurons, he found the system could memorize about 0.15 N, or ≈ 15 memories. Recall however failed if more fixed points were loaded into the network. Solving the above equation for L when N ≡ 100 yields L ≡15.7719, which is in close agreement with Hopfields result's.[30]

The maximum number of memories *does not* scale linearly with the dimensionality of the system. A plot of L_{max} versus N for the cutoff criteria we are using is seen below. This plot was calculated using MAPLE to iteratively solve the capacity equation as the system size N was varied.

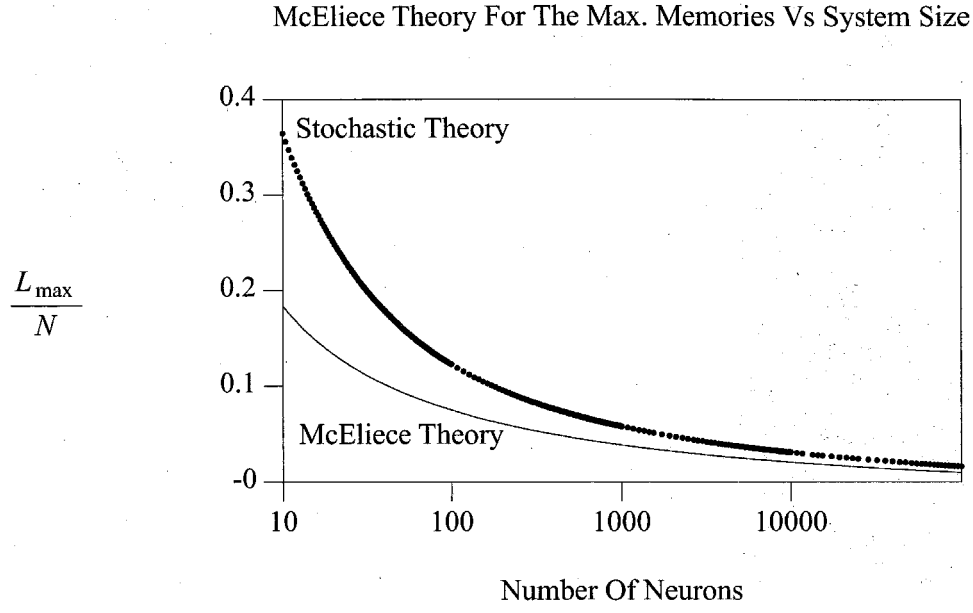
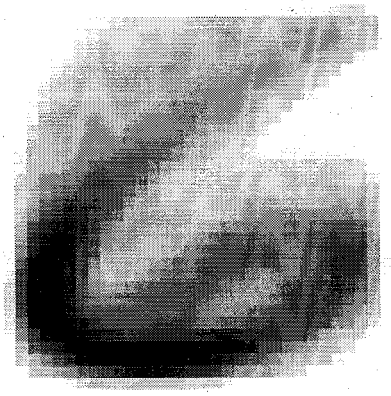


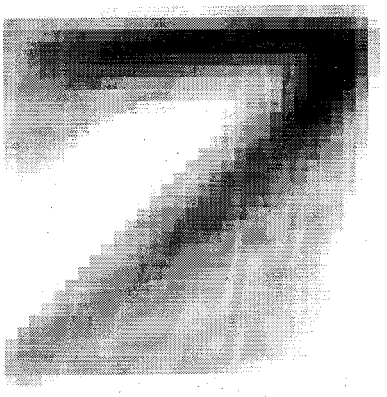
Figure 96. Theoretical Maximum Number Of Stable Points Versus System Size

The bold curve is the stochastic theory advanced here. The thinner solid curve is a theoretical calculation of McEliece et al.[31] These authors showed the capacity $\frac{L_{max}}{N}$ should be proportional to $\frac{1}{2Log(N)}$. We obtain good agreement between the McEliece theory and the stochastic theory over four orders of neuron system size magnitude. Furthermore, the McEliece and stochastic results asymptotically approach each other as $N \rightarrow \infty$.

17. Handwritten Digit Database Descriptions

Three handwritten digits databases are used in this thesis to demonstrate key points. These were obtained from the Center Of Excellence For Document Analysis And Recognition (CEDAR) at SUNY Buffalo University, A T & T Bell Labs, and NIST respectively. The CEDAR data is used for all examples unless otherwise indicated. This is due to the fact that the CEDAR CD-ROM consisted of a large enough number of images for statistically valid conclusions to be drawn. The Bell Labs and NIST databases were substantially smaller in size. In all the numerical simulations, there was no separation of the images into training and test sets. All data was used for training. This is in keeping with the philosophy of this thesis that the statistics characteristic of the underlying problem are being tabulated and are not changing in time. Hence, using the same data for training and testing is not improper. The CEDAR digits were 8 bit grayscale and varied in width and height. This was due to the fact the CEDAR digit segmentation algorithm operated on digitized envelope images. The single digit images were extracted from multiple digit zip code blocks. The varying size CEDAR images were converted by the author to a 32 by 32 pixel image, with 8 bits per pixel, using the *PbmPlus* software suite of image processing utilities. The principle component was found using C code written by the author using an algorithm to be described below. The conversion of the resulting PCA images to postscript for





inclusion in this thesis was done using the *xv* program. The images to the right are the PCA images for the Six (upper) and Seven (lower) digits respectively. As expected, these PCA images correspond approximately to the average across all respective handwritten digits seen during training. This is due to the similarity of principle components with matched filters when the variation across the sample images is small. That is, if all the six digit images were virtually identical, then the training set is not varying, and the optimum receiver structure is a matched filter.[32] In matched filters, the demodulating signal is the image that is expected to be received. Thus, PCA and matched filter structures can be expected to agree when the variation across training samples is small. This indeed occurs, and can be seen from the two PCA images shown above. The PCA images loosely correspond to average, or typical, six and seven digit images.

The A T & T Bell Laboratories digit images were of size 16 by 16 binary pixels. There were a total of 120 images for each digit. These were broken down into 10 samples each from 12 writers.

The NIST data consisted of approximately 3,400 characters of 32 by 32 8 bit pixel images compiled from 49 writers filling out fake forms.[33]

18. Neural Learning Algorithm

There are several steps in the implementation of the algorithm. First, the eigenvectors for the two correlation matrices must be found. Second, the mean and variance of both hypotheses projections onto the neuron's weight vectors must be calculated to determine a threshold point if hard decision decoding is used, and to determine the sigmoid shape if soft decision decoding is

used. We shall discuss the weight vectors, and moment accumulation in the following two sections.

18.1 Single Layer, Multiple Neuron, Weight Vector Calculation

The neural learning rule implemented is based on Erkki Oja's PCA learning rule work.[34] [35] [36] [37]

For a single neuron, learning is accomplished using a labeled training set. Whenever a hypothesis H_0 input vector is presented to the neuron, weight vector $\mathbf{W}_0$ is updated. Similarly, $\mathbf{W}_1$ is updated upon presentation of a hypothesis H_1 input vector. When the neuron is used to classify an unknown input vector, the eigenvector difference weight vector $\mathbf{W} \equiv \mathbf{W}_1 - \mathbf{W}_0$ is formed. The difference weight vector is the neuron weight vector used in the classification mode of operation.

For each input vector $\mathbf{X}$ presented during training, the weight update rule used to update the weight for hypothesis i, $\mathbf{W}^i$, is the following.

$$\mathbf{W}_{n+1} \equiv \mathbf{W}_n + \alpha_n y_n \mathbf{X}_n$$

$$\text{where } y_n \equiv \mathbf{W}_n \bullet \mathbf{X}_n$$

Equation 60. Oja's Single Neuron Weight Vector Update Rule

The subscript n on the data and weight vectors denotes that these are for the n'th iteration in our update process. The algorithm described above is for a single neuron. The neuron will converge to the principal component eigenvector of the correlation matrix for the data presented to it. The

coefficient α_n is a parameter which decreases as $n \rightarrow \infty$. In this work, α_n was set to $\frac{1}{n}$. This setting meets Oja's convergence requirements.[38] The update rule for L neuron's in a layer is seen below.

$$\mathbf{W}_{n+1}{}^{m} \equiv \mathbf{W}_{n}{}^{m} + \alpha_n \left\{ y_n{}^{m}\, \mathbf{X}_n{}^{m} - \phi^m\, y_n{}^{m} \sum_{\substack{i=1 \\ i \neq m}}^{L} y_n{}^{i}\, \mathbf{W}_n{}^{i} \right\}$$

Equation 61. Oja's Weighted Subspace Multiple Neuron Update Rule

Above, n is the iteration number, and m is the neuron whose weight vector $\mathbf{W}^m$ is being updated. The subtraction of the sum serves to provide a symmetry breaking which avoids having all L neurons contending for the principal eigenvector. The ϕ^m are the symmetry breaking coefficients. In this work, they were evenly distributed about 1.0. Thus, for the five neuron case, these were set to 0.8, 0.9, 1.0, 1.1 and 1.2. The smallest ϕ^m yields the principal eigenvector. Successive eigenvectors are obtained as ϕ is increased.

The algorithm above was implemented in C, and run on the three databases discussed above. Less than one hundred epochs were used for training all five neuron layers, although most convergence occurred in the first few epochs. The table below summarizes the numerical calculations and parameters used for both the *Three / Five* and *Six / Seven* digit recognition runs.

Six - Seven Digit Image Oja Learning Run Statistics	
Symmetry Breaking Array	1.1,1.05,1.00,0.95.0.90
Image Size	32 x 32 (1024) Double Float Pixels
Competing Neurons	5
Starting Step Size	0.002
Number Of Epochs Per Hypothesis Training Run	36(.11)
Images Seen Per Epoch	1616
Total Images Seen Per Hypothesis Training Run	58360
Run Time Per Hypothesis	2 Hours, 39 Minutes(Sparc 10)

TABLE 20. 6 : 7 Image Oja Numerical Simulation Parameters

Three - Five Digit Image Oja Learning Run Statistics	
Symmetry Breaking Array	1.1,1.05,1.00,0.95.0.90
Image Size	32 x 32 (1024) Double Float Pixels
Competing Neurons	5
Starting Step Size	0.002
Number Of Epochs Per Hypothesis Training Run	80.00
Images Seen Per Epoch	1459
Total Images Seen Per Hypothesis Training Run	116720
Run Time Per Hypothesis	3 Hours, 57 Minutes(Sparc 10)

TABLE 21. 3 : 5 Image Oja Numerical Simulation Parameters

The run times need some explanation. There are two runs per hypothesis, and two hypotheses. For each hypothesis, one run was made to determine the eigenvector set. This is the run time seen in the tables above. A second, much shorter run consisting of a single epoch was then made to determine the hypotheses projection statistics for the eigenvectors. This is because the eigenvectors needed to be stable before accurate statistics could be determined. The single

epoch time was negligible compared to the training run. Thus, the net computer time needed for both hypotheses was twice that listed above, or approximately 8 hours for determination of all 3-5 eigendata. However, since two Sun Sparc-10's were available in the lab, the two hypothesis computations were done in parallel, so that the true elapsed time was still only 4 hours. The main cpu usage during a series of diagnostic runs seemed due more to i/o access than actual Oja rule updates. Optimization of i/o access in place of the crude C scanf/printf routines used in the code could perhaps have reduced the computer time. For example, in the 3-5 simulation, a grand total of 1024 x 1459 x 82 x 2, or 245 *million* double precision numbers were fetched from disk ! There are four bytes per double precision number, so this is a total i/o throughput of more than 980 MEGABYTES over approximately 4 hours, when the two hypotheses ran in parallel. This averages out to about $\frac{1}{7}$ of a second for every 32 x 32 pixel image processed.

There are several minor variations in the algorithms which Oja and his co-authors present in the papers above. The one specifically implemented for the 6-7 and 3-5 data runs used the version producing unit magnitude eigenvectors.[39]

18.2 Unbiased, Consistent, Sufficient Statistics And Moment Estimation

There is considerable literature on how to optimally extract estimates of moments from a population sample.[40] [41] [42] In this section, we shall review the basic concepts of point estimation using samples drawn from an infinite population. That is, given a set of M data points $\{g_i\}$ drawn from a distribution G, we wish to estimate μ_G and σ_G^2 from $\{g_1, g_2, \ldots\ldots, g_M\}$. Several definitions and concepts are first given. Denote the set $\{g_i\}$ as ξ.

Suppose we wish to estimate some parameter ζ from the set ξ. We devise an estimator $\hat{\zeta}(\xi)$ and study $\hat{\zeta}$. Numerous properties of the estimator $\hat{\zeta}$ are listed below.

$$E[G] \equiv \mu_G \quad , \quad Var(G) \equiv \sigma_G^2$$

$$Unbiased \quad : \quad E[\hat{\zeta}] \equiv \zeta$$

$$Consistent \quad : \quad Unbiased \text{ and } \lim_{M \to \infty} Variance(\hat{\zeta}) \to 0$$

$$Sufficient \quad : \quad Variance(\hat{\zeta}) \leq Variance(\ All\ Other\ Estimators\ \hat{\zeta}'\ )\ Of\ \zeta$$

$$\overline{G} \equiv \frac{\sum_{i=1}^{M} g_i}{M} \rightarrow Sufficient\ Estimator\ Of\ \mu_G \quad , \quad \overline{G} \sim \eta\left(\mu_G\ ,\ \frac{\sigma_G^2}{M}\right)$$

$$Define\ S^2 \equiv \frac{\sum_{i=1}^{M} (\ g_i - \overline{G}\ )^2}{M} \equiv \frac{\sum_{i=1}^{M} g_i{}^2}{M} - \overline{G}^2$$

$$S^2 \equiv \chi^2(M-1) \qquad \{\ S^2\ is\ Chi\ Square\ Distributed\ With\ M-1\ Degrees\ Of\ Freedom\ \}$$

$$E[S^2] \equiv M-1 \quad , \quad Var(S^2) \equiv 2\ (\ M-1\ )$$

$$\frac{M}{\sigma_G^2} S^2 \rightarrow Sufficient\ Estimator\ Of\ \sigma_G{}^2$$

Equation 62. Statistical Definitions And Distributions[3]

A sufficient estimator is one which extracts all possible information from a sample to make a

3. Some authors define S^2 as $\frac{\sum_{i=1}^{M} (\ g_i - \overline{G}\ )^2}{M-1}$.

point estimation. No other estimator can do a better job at estimating ζ. Cramer and Rao derived a general formula for the minimum possible variance an unbiased point estimator can have. This is called the Cramer - Rao bound. Any unbiased point estimator which meets the Cramer - Rao bound is a sufficient statistic for that ζ. There may be several estimators which are fundamentally different, and yet all meet the Cramer - Rao bound. $\overline{G}$ and $\frac{M\,S^2}{\sigma_G^2}$ are sufficient statistics for the mean and variance of G respectively. Note that $\overline{G}$ and S^2 are random variables because different sample sets $\{g_i\}$ may be drawn each time, resulting in different $\overline{G}$ and S^2 values. $\overline{G}$ and $\frac{M\,S^2}{\sigma_G^2}$ are also statistically independent random variables.[43]

In our algorithm development, numerous quantities are estimated from the training set. For a single neuron, these are tabulated below.

Parameter	Name	Why Needed And Where Used
$\mathbf{W}_0$	H_0Eigenvector	Find H_0 Projection Direction
$\mathbf{W}_1$	H_1 Eigenvector	Find H_1 Projection Direction
P[H_0]	A Priori H_0 Probability	For Sigmoid And Threshold
P[H_1]	A Priori H_1 Probability	For Sigmoid And Threshold
E[$\mathbf{W} \bullet \mathbf{X}_0$]	Mean Of H_0 Projections	For Sigmoid
E[$(\mathbf{W} \bullet \mathbf{X}_0)^2$]	For Variance Of H_0 Projections	For Sigmoid
E[$\mathbf{W} \bullet \mathbf{X}_1$]	Mean Of H_1 Projections	For Sigmoid
E[$(\mathbf{W} \bullet \mathbf{X}_1)^2$]	For Variance Of H_1 Projections	For Sigmoid

TABLE 22. Neuron Statistical Parameters Extracted From Training Set

Recall $\mathbf{W} \equiv \mathbf{W}_1 - \mathbf{W}_0$. All parameters in the table, with the exception of the determination of $\mathbf{W}_1$ and $\mathbf{W}_0$, can be calculated by sufficient statistics operating on the data flowing through the

neuron during training. That is, during training, one global wire can connect all neurons, and output a single bit message indicating whether the incoming image is H_1 or H_0. Internal to each neuron, the moment and eigenvector information is compiled and stored. There is no massive global synchronization or communication. Only data, a one bit training on/off signal, and a one bit hypothesis H_0/H_1 indicator.

The Oja et al. update rule used in this thesis has been determined to be unbiased and consistent.[44] However, it has not been proven to be sufficient. That is an open question. Thus, the moment extraction estimators are optimum and minimum variance, while it can only be said the weight update rule asymptotically yields the correct eigenvector(s), as the number of training vectors $\rightarrow \infty$.

19. A Statistical Communications Theory Perspective On Neural Computation

As motivation for a communication system perspective on neural computation, consider the following problem. One is given noisy time series data, {x(0),x(-T),x(-2T),....x(-KT)}, sampled from a continuous function x(t), and asked to make a decision at time t=0 as to which of two possible events occurred. Imagine the events are signals, $S_0(t)$ or $S_1(t)$, corrupted by additive noise. Thus, the observed data is the received signal $x(t) \equiv S_i(t) + n(t)$. We assume (or low pass filter x(t)) that x(t) is bandlimited, and sample at or above the Nyquist rate, to obtain the discrete set {x(nT)}. The solution to this signal detection in noise problem has been well studied in communications theory. The optimum solution, or receiver structure when the noise is white and Gaussian, is the matched filter.[45] [46] [47] For instance, imagine the simple case where $S_0(t)$

is a constant signal of amplitude A, and $S_1(t)$ a constant signal of amplitude -A. If the noise per sample is zero mean, Gaussian, with variance σ^2_{Noise}, and S_0 and S_1 occur equally often, then the optimum weight vector by symmetry considerations has all constant entries, and the final decision of an observed data point is seen below.

$$y \equiv \sum_{n=0}^{K} x(-nT) \quad , \quad P[y|S_0] \equiv \eta\left[(K+1)\ A \ , \ (K+1)\ \sigma^2_{Noise} \right]$$

$$P[y|S_1] \equiv \eta\left[-(K+1)\ A \ , \ (K+1)\ \sigma^2_{Noise} \right] \quad , \quad y \underset{S_1}{\overset{S_0}{\gtrless}} 0$$

Equation 63. Matched Filter Equations

A matched filter will yield the lowest probability of error under the signal and noise assumptions above. For binary, antipodal signals, $S_0(t) \equiv -S_1(t)$, the optimum weight vector or Finite Impulse Response (FIR) taps, are samples of the function W(t) $\equiv S_0(t)$.

In neural type situations, the problem is not as clear cut as deciding whether one of two fixed signals was transmitted. For instance, in handwritten character recognition, everyone has their own way of writing digits. However, any individuals style is repeatable and typically varies only slightly from one instance to another. In this case, it is more appropriate to consider the problem as follows. There are two signal classes, S_0 and S_1, each described by a probability distribution. The signals S_i are considered as random variables. Noise is added, and a decision is sought as to which signal distribution the received data was generated from. This problem again has been solved. It has again a matched filter type structure, only now the sampled

optimum weight function W(t) is the largest eigenvector of the correlation matrix of the sampled signal $S_0(t)$. To quote Haykin : "*The eigenfilter associated with the largest eigenvalue of the correlation matrix of the signal component at the filter input is the optimum filter.*" And "*The optimum filter maximizes the signal to noise ratio for a random signal in additive white noise.*" "*A matched filter on the other hand, maximizes the signal to noise ratio for a known signal in additive white noise.*"[48] This view of neural computation as a random signal drawn from one of two possible signal ensemble's, and then corrupted by uncorrelated (white) Gaussian noise, forms the foundation of the statistical communications model of neural computation.

19.1 Neural Computation, And The Information Theoretic Channel Model

In this section, we discuss the channel model equivalent for a neural network implementing a binary hypothesis test. To simplify the analysis, we shall assume the pdf's for each neuron are symmetric, with $\mu_0 \equiv -\mu_1$ and $\sigma_0^2 \equiv \sigma_1^2$. The arguments to be presented generalize to the more practical situations where these assumptions do not hold.

Earlier, we noted that a single neuron's probability of error performance could be represented by a Binary Symmetric Channel. Several parallel neurons in a single layer could also be collapsed into a single equivalent neuron, and hence represented by a single BSC. Similarly, serially cascaded BSC's representing successive layer neurons can be collapsed down into a single BSC model. The figures below graphically represent these operations. For simplicity, all neurons / BSC's are assumed to have identical ε's.

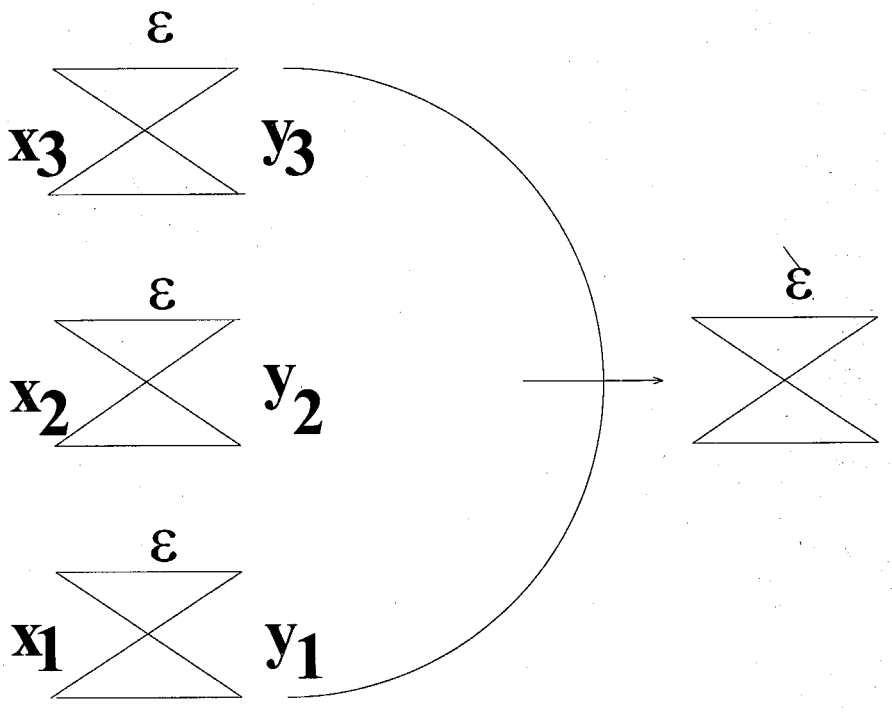


Figure 97. Three Parallel Binary Symmetric Channels Are Equivalent To One

The equivalent parallel ε' is seen below.

$$\varepsilon' \equiv \binom{3}{2} (1 - \varepsilon)\, \varepsilon^2 + \varepsilon^3$$

Equation 64. Equivalent Parallel BSC Crossover

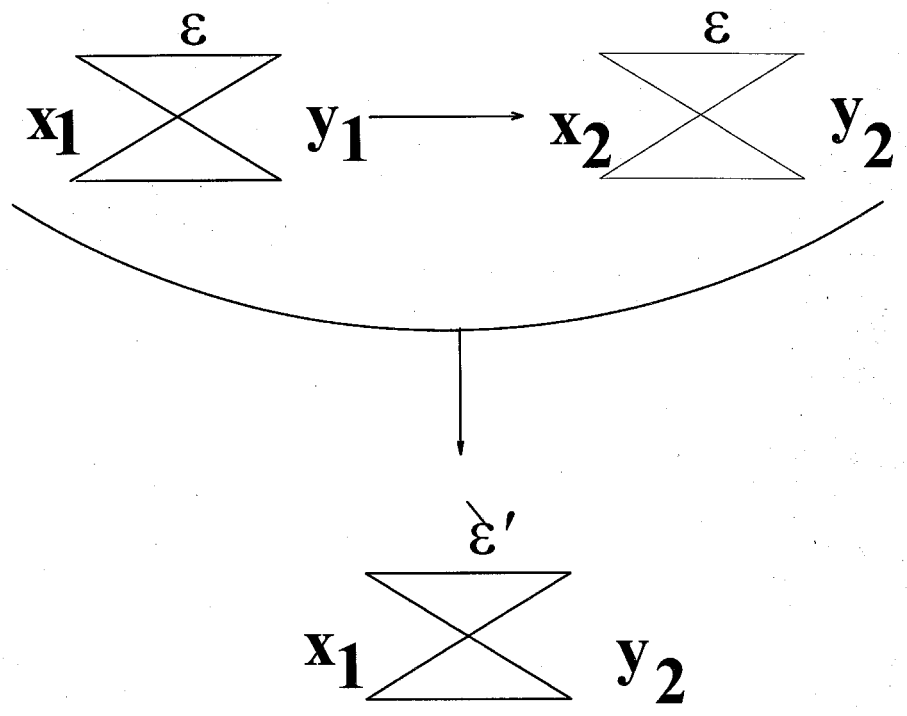


Figure 98. Two Cascaded Binary Symmetric Channels Are Equivalent To One

The equation below gives the equivalent ε' for the cascaded BSC's.

$$\varepsilon' \equiv 2(1-\varepsilon)\varepsilon$$

Equation 65. Equivalent Cascaded BSC Crossover

Below is plotted the equivalent system performance for a three neuron parallel concatenation(**P**) and a two neuron serial (**S**) concatenation. Note that parallel concatenation improves performance, while serial concatenation worsens performance.

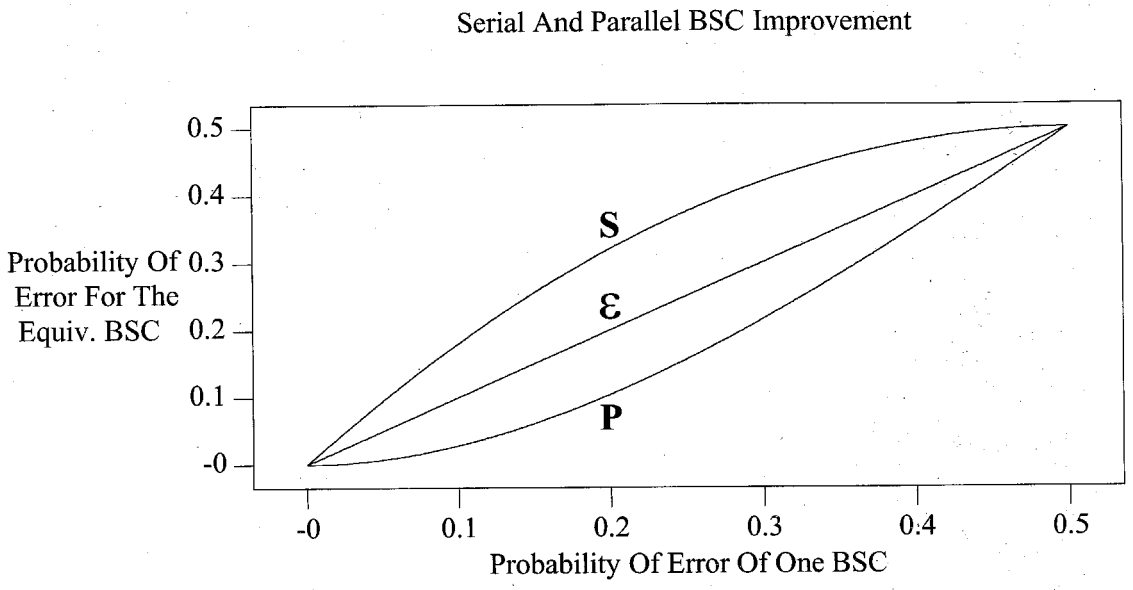


Figure 99. Serial And Parallel BSC Improvement

By recursively applying these parallel and serial reduction operations, an entire network implementing a binary hypothesis test can be collapsed into a single BSC model.

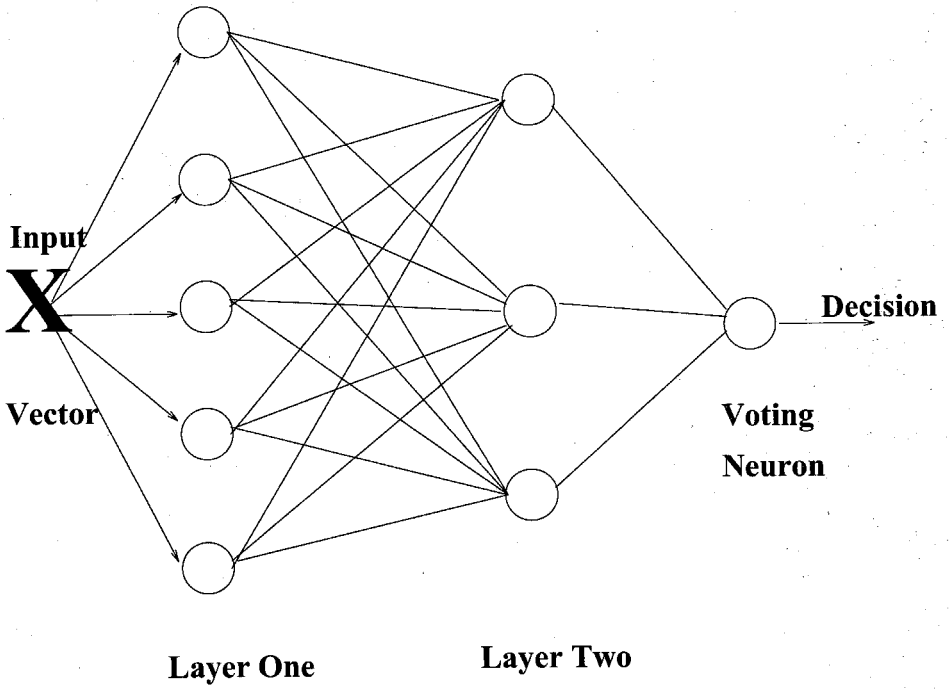


Figure 100. Generic Neural Network

The network above reduces to three serial BSC's, which then reduce to a single BSC. The final, equivalent, BSC has a probability of error ε' which represents the systems error performance.

There is a corresponding channel capacity, $C(\varepsilon')$, for the system above which was developed earlier. But consider the BSC model in our development has a more physical representation than abstract cross over probabilities between states. The cross over probability is given by the tails of a Gaussian distribution spilling over a threshold. This is how a decision of whether a drawn sample came from a Gaussian density function with mean $\pm\mu$ and variance σ^2 is made. This scenario is that of detecting a signal in noise. Under our equal variance, symmetrical means assumption, our neuron is extracting an antipodal ($\pm\mu$) signal buried in Additive White

Gaussian Noise (AWGN). The whiteness is the independence in time across samples. Thus, whiteness comes from the assumption that the noise in a vector element sample is independent from the noise in the same vector element in any other sample. Shannon derived the channel capacity C in bits per second for detection of a signal in AWGN as $C \equiv bandwidth\ Log_2(1 + SNR)$. The Shannon derivation uses bandwidth as the number of samples per second, and the SNR as the ratio of the mean of the signal to the root noise power accumulated in one second. Consider the neural network as reading in one sample vector **X** per second. The dimension of the incoming vector is the number of samples taken per second, and thus proportional to the bandwidth by the application of Nyquists criterion for bandlimited signals. The projection means $\pm\mu$ are the signals. The noise is the standard deviation of the Gaussian distributions about the means. Thus, an analogy is drawn between neural computation and diversity communication via the combination of multiple channels transmitting the same message. That is, each pixel is attempting to convey the same one bit, on/off, message in our hypothesis testing scenario. Therefore, each pixel in the images can be considered as transmitting a binary signal. However, the noise and distortion in the system is so high that only when a large number of parallel signals are sent simultaneously can the binary message be classified with high probability of success. This view of neural hypothesis testing, when coupled with Shannons bandwidth / SNR formula, lead to an interesting picture of network depth versus width.

Suppose you are given the network above. You are then given five more neurons, and asked to put it into the network in the location which will best improve overall system performance. If you used the five to extend the width of either of the two existing layers, you are increasing the

bandwidth of those layers. If you created another layer in an attempt to further sharpen the Gaussians presented to the final voting neuron, you are using the neurons to improve the system SNR. Shannons formula gives us a tradeoff for improving system performance. Improving bandwidth improves channel capacity and thus system performance by a linear factor. Of course, as was discussed above, each layer is bandlimited by the maximum dimension of the two hypotheses subspace given by the number of correlation eigenvalues lying above the noise floor. Similarly, improving SNR increases channel capacity, and thus system performance, but logarithmically. This is another interpretation of the system performance behavior obtained during the parity examples. Performance improvement trails off quickly (logarithmically) as the network depth is increased. Networks which appear to work well are wide and shallow. This is in keeping with the theory advanced above that the statistical model implicit in our central limit theorem and other approximations is attempting to exploit differences in the lower moments of the two conditional densities.

20. Information Flow

Another useful picture is to think about information flow through the network. Linkser has been successful in this area. Linkser proved that using the set of eigenvectors corresponding to the largest eigenvalues of a set of jointly Gaussian random variables as weight vectors, maximizes information flow from input to output.[49] Building on an information flow approach such as Linksers, a useful interpretation of the Shannon conclusion we came to above can be made by appealing to an almost trivial algorithm. Keep adding neurons to a network so as to maximally increase the information flow through the network, from source (input) to sink (output). Recall

each layer has an upper bound saturation limit beyond which adding more neurons does not help. This limit is given by the maximum dimension of the two hypothesis subspaces for the input space to that layer. A general algorithm is to calculate which unsaturated layer has an equivalent BSC representation with the highest probability of error, and supplementing that layer with more neurons. This algorithm can be iteratively implemented until no further improvement occurs, resulting in a network with the maximum channel capacity. When all layers have been saturated, two neurons on a new layer are added. If this layer saturates with these two neuron's, the new layer is not needed. That is, as much SNR enhancement as possible has been achieved.

This general algorithm description has been brief. It's importance is more as a conceptual picture for growing a neural network to an optimum size. A more realistic algorithm is to monitor the eigenvalue corresponding to the weakest eigenvector introduced in a layer. If that eigenvalue is approximately equal to the eigenvalue of his neighbor, then we have reached the flat portion of the eigenvalue spectrum, and more neurons will not help that layers throughput. In practice, given the strength of neural networks seems to be in extracting lower moment information, and keeping in mind how the OJA algorithm learns, it may be more efficient to place excessive numbers of neurons in each layer, and allow neurons to die or turn themselves off from the network when their eigenvalues decay to a level a few orders of magnitude less than the principle eigenvalue.

20.1 K-Winner Take All Networks

In the discussion of lateral inhibition, a single output from a layer was passed on to the next layer when a Winner Take All approach was used. This however, defeats the large dimensionality argument used to motivate several points in the discussions above. A K Winner Take All (K-WTA) implementation is a tradeoff between the two diametrically opposed operations of WTA and large dimensionality inputs. A K-WTA implementation passes on the largest K magnitude neuron outputs to the next layer. This is statistically identical to outputting a vector with the top K order statistics to the next layer. Hence, it is advantageous to look at the variation of the top K order statistics to gain a feel for how K-WTA pdf's differ from the maximum WTA pdf. If, for instance, the spread of the top K-WTA pdf mean's for each hypothesis does not spill over into each other, than a K-WTA implementation can be used to introduce redundancy by averaging the top K-WTA results, and figuring out which hypothesis this average may have come from. However, it is important to realize the top K order statistics are statistically dependent.

Below is graphed the top 5 order statistics for the absolute magnitude of the texture Gaussians discussed above. Thus, the plots below are the top five order statistics for the texture two hypotheses.

$$P[\,y \mid H_0\,] \equiv \left| \eta(\,0\,,\,1024\,) \right| \quad , \quad P[\,y \mid H_1\,] \equiv \left| \eta(\,0\,,\,\frac{1024}{3}\,) \right|$$

Equation 66. Density Functions For The Top 5 Texture Order Statistics

The general order statistic density formula is seen below. Here r is the order, and N is the sample size. Thus r = 99, N = 100 means the density for the 99th out of 100th entry in a sorted list of the absolute values of samples drawn from the corresponding generating density.

$$f_r(x_r) = \frac{N!\ p(x_r)}{(r-1)!\ (N-r)!} \left\{ \int_{-\infty}^{x_r} p(z)\ dz \right\}^{r-1} \left[\int_{x_r}^{\infty} p(z)\ dz \right]^{N-r}$$

$$where\ \ x_r \in (-\infty, \infty)$$

Equation 67. Order Statistic Formula

The plots below are for sample sizes of 11, 51, and 101. These were the sample sizes considered above in the maximum absolute order statistic discussion with the same texture density functions. The bold dotted curves are for the larger variance, $\sigma^2 \equiv 1024$, texture density. The thin, solid curves are for the smaller variance, $\sigma^2 \equiv \frac{1024}{3}$, texture density.

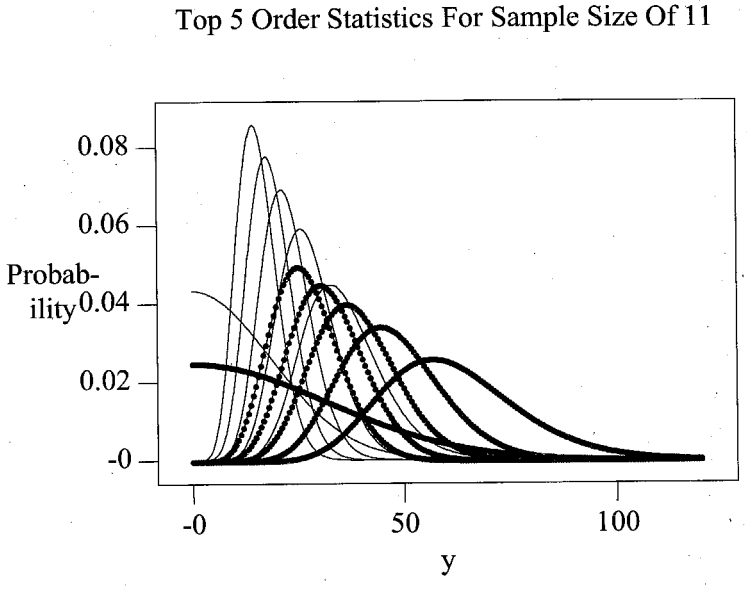


Figure 101. Sample Size 11 - Top 5 Order Statistics

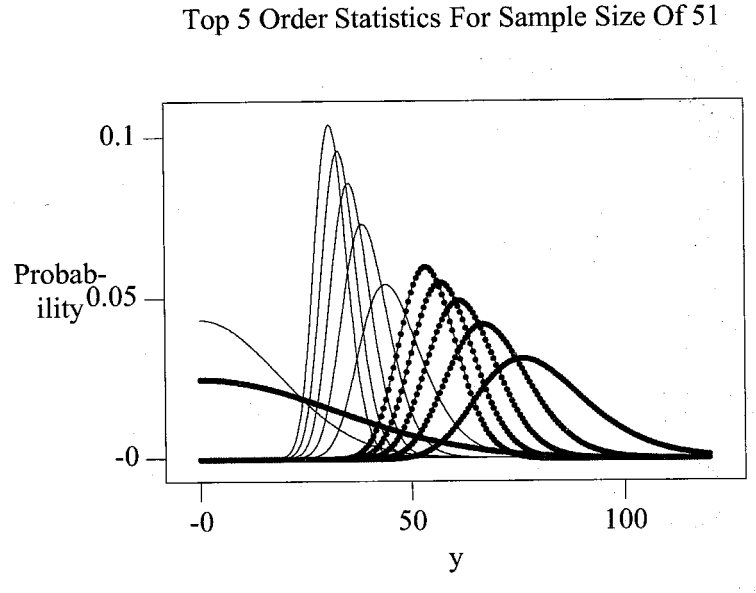


Figure 102. Sample Size 51 - Top 5 Order Statistics

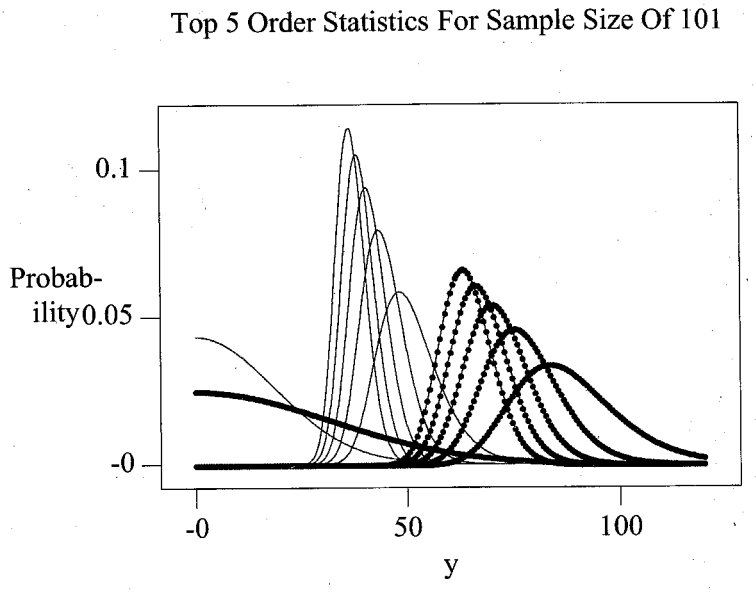


Figure 103. Sample Size 101 - Top 5 Order Statistics

Several effects in the above plots match our intuition and bear comment. First, the means of the order densities decreases as the order decreases. This is expected, because we are now taking a lower magnitude number from the sorted list as the value of r decreases for fixed N, where r and N are defined above. Second, the variance also gets narrower as r decreases. This was expected, since as we approach the maximum of the density function, we expect more points will be generated in that area, and thus there will be less variability in this region. Last, for all r, there is increasing separation among the two hypothesis density functions as the sample size increases. This was also seen in the numerical example of the maximum absolute order statistic.

In general, K-WTA and WTA implementations should be thought of as suppressing the noisy projections, and passing on the high SNR components for further computation. This perspective

of non-linear operations acting on a signal embedded in noise to enhance the SNR of the signal is discussed at great length in Davenport and Root.[50] D&R's discussion focuses on μ law devices, which have a transfer function acting on the received signal, of the form $f(x) \equiv x^v$. Here the received data is the sum of a transmitted signal plus noise : x(t) ≡ s(t) + n(t). D&R conclude that for signals buried in Gaussian noise, the general effect of μ law devices is small signal suppression. That is, for small input SNR's (<< 1), the SNR gain of a μ law device is proportional to the square of input SNR. For large input SNR's (>> 1), the μ law gain is directly proportional to the input SNR. The result of μ law devices is to drive low power signals into the noise floor.[51] WTA and K-WTA implement the same type of non-linear functionality in a radically different fashion. Yet the general concept of small signal suppression is the same.

20.2 Training Set Peculiarities

A few comments are warranted in regards to training sets. In the algorithm described in this thesis, statistics are compiled based on the training data. The training data must faithfully represent the underlying problem probability density functions. This means the user should not attempt to bias the training set to speed agreement on all training set data points. That is, in algorithms such as the perceptron convergence technique, the frequency with which hard to learn points are represented in the training set is often increased. The reason why this is sometimes done is so that the algorithm will spend more time trying to fit these tougher points. By essentially calling attention to points one knows will be difficult, one speeds convergence to a weight vector solution.[52] This should *not* be done in the algorithm described in this thesis, as it biases the probability distributions the network thinks it is learning. The training set for this

thesis's method must accurately reflect the statistics of the problem the network is being asked to solve. This includes both the conditional hypothesis distributions, as well as the relative frequency of the two hypotheses.

20.3 MUSIC And Other Principle Eigenspace Techniques

A basic concept in this algorithm has been the identification of subspaces where there is a substantial amount of activity. This fact, and the idea that the dimension of such a subspace can often be much less than the overall data dimension, has been exploited in many other algorithms.[53]

A few are mentioned here.

20.3.1 MUSIC : **MU**ltiple **SI**gnal **C**lassification

The MUSIC algorithm, and it's close cousin, the minimum norm method, are used for parameter estimation of one or more signals in sampled data.[54] For instance, a classic application of the MUSIC algorithm is in identifying the frequencies of one or more sinusoids in Gaussian noise. The signal waveform type (eg : sinusoids), must be known a priori. The signal also typically has a free parameter (eg : frequency ω). Unknown sampled data is obtained, and the eigenvalue spectrum is calculated. The eigenvalue spectrum is divided up into a signal set of eigenvalues, and a noise set of eigenvalues. The eigenvectors for the noise set are computed. A vector corresponding to the known sampled signal as a function of the free parameter is then projected onto the noise subspace. The free parameter is swept over an allowed range, and the minimum projection determines the parameter. Multiple local minima,

for instance at $\{\omega_1, \omega_2, \cdots \omega_l\}$, indicate multiple signals are present with these frequencies. The minimum norm method is similar. MUSIC is notable because of three facts. First, it identifies signal and noise subspaces, and is oriented towards utilizing projections onto these subspaces to make a signal classification. Second, a detailed analysis of the MUSIC technique for uncorrelated sinusoids buried in Gaussian noise indicates the algorithm asymptotically approaches the Cramer-Rao bound as the number of data samples $\rightarrow \infty$.[55] Third, the MUSIC estimator is a random variable since it depends on a sample from an unknown, but fixed and deterministic, signal population. This random variable is asymptotically Gaussian. This philosophically agrees with the central limit theorem approximation taken early in this thesis work.

Although MUSIC has some qualities in common with this thesis work, it is fundamentally different in several aspects.

20.4 Adding Noise To The Training Data

Several authors have cited improved generalization and faster learning when noise is added to the data before it is handed off to the network. The amount of noise, optimum distribution, and a theoretical basis for noise injection are open questions. This section will briefly discuss noise injection for this model.

A key feature to the operation of this algorithm is the statistical independence of a neural layer's outputs. This independence was achieved by decorrelating the outputs, and assuming the outputs were jointly Gaussian. Hence, the outputs are independent. If the neural output

distributions were *not* Gaussian, decorrelation does not imply independence, and overall system performance degrades. If Gaussian noise is added to each pixel independently, then two effects are seen. First, the resulting projection of the data has a distribution which more closely approximates a Gaussian. Second, the correlation coefficient between the neural outputs decreases. The increased Gaussian behavior is due to the central limit theorem application. If before the projection pdf was the random variable y, now it is y + n, where n is a Gaussian random variable. This sum will be more Gaussian distributed than y alone was.

The second benefit arises because the correlation coefficient between two neuron outputs is changed, as seen below.

$$\hat{y}_k \equiv y_k - \mu_k \quad , \quad \rho^{old} \equiv \frac{E[\,\hat{y}_i \hat{y}_j\,]}{\sqrt{E[\,\hat{y}_i{}^2\,]\;E[\,\hat{y}_j{}^2\,]}}$$

$$\rho^{new} \equiv \frac{E[\,\hat{y}_i \hat{y}_j\,]}{\sqrt{(\,E[\,\hat{y}_i{}^2\,] + N\,\sigma^2\,)\;(\,E[\,\hat{y}_j{}^2\,] + N\,\sigma^2\,)}}$$

Equation 68. Improvement In Cross Correlation Coefficients Due To Injected Noise

Here, σ^2 is the variance of the zero mean noise injected into each pixel, and N is the total dimensionality of the incoming, raw data. For instance, in the 32 by 32 input *Six & Seven* image vectors considered above, the average variance across the five vectors in table 2 was 4.7. Thus, addition of noise with zero mean, and variance 4.7/1024 ≈ 0.0046, to each image pixel will decrease *every* cross correlation coefficient ρ_{ij} by half. Thus, noise injection can theoretically help system performance by improving the statistical independence between neural

outputs.

The tradeoff is the hypothesis projections now have increased variance than in the case without noise. Hence, individual neuron performance is degraded in every case. System performance may however improve due to the decreased cross correlation effects between neuron outputs.

20.5 General Learning Algorithms

	Deterministic Algorithms	**Random Algorithms**
Deterministic Inputs	Perceptron Convergence Backprop	Boltzmann Machine Simulated Annealing
Random Inputs	This Thesis	Future Research

TABLE 23. Algorithm Types

Above we sketch the main types of algorithms. In the authors view, the deterministic input algorithms are subsets of the random input algorithm approaches. This is because taking the variance $\rightarrow 0$ in a random input algorithm yields it's deterministic counterpart. It should be noted various authors have shown deterministic algorithms such as Backprop are also stochastic approximators to random problems.[56] For instance, consider the output $\hat{Y}$ of a network as providing a point estimation for a continuous parameter Y. Halbert White discusses how Backprop produces a set of weights which minimizes the expected mean square error between Y and $\hat{Y}|\mathbf{X}$, $E[\,||Y - \hat{Y}|\mathbf{X}||^2\,]$. The expectation is taken over the joint density function $P(\mathbf{X}, Y)$, and the training set is a labeled sample drawn from the joint distribution of $\mathbf{X}$ and Y.[57]

20.6 Confidence Estimates

Confidence estimates for the hypotheses can be extracted from networks designed as described above. That is, instead of the binary decision, one can obtain the probability of H_0 versus H_1. This is useful for unsymmetrical or complex cost functions associated with the calculation and minimization of a risk for a binary decision. That is, if we wish to minimize a criterion other than probability of error, the estimated probabilities of one hypothesis versus the other are often needed. In all systems, including the ones described in this thesis, a histogram can be constructed during training which estimates the output hypothesis probabilities. However, the algorithm described in this thesis outputs the relative hypothesis probabilities from the voting node when a sigmoid and soft decoding are used. Recall that the soft neuron outputs a real number z of the form P[$H_1 | \mathbf{X}$] - P[$H_0 | \mathbf{X}$]. Since P[$H_1 | \mathbf{X}$] + P[$H_0 | \mathbf{X}$] $\equiv$ 1, trivially one obtains $P[H_1 | \mathbf{X}] \equiv \frac{1+z}{2}$, and $P[H_0 | \mathbf{X}] \equiv \frac{1-z}{2}$.

Typical risk functions R are of the form $\sum C(x,y)$, where x is the hypothesis you guessed, and y is the hypothesis that was actually true. C(x,y) is the cost of the joint set of (x,y) events. Typically, one attempts to arrive at a decision strategy which minimizes the expected value of the risk function $\rightarrow$ E[R]. To illustrate, for our work above, Bit Error Rate (BER) was minimized. This is the same as optimizing the risk function below.

	H_0 **Actually Occurred**	H_1 **Actually Occurred**
Guessed H_0	0 (No Risk)	1 (An Error Costs 1 Unit)
Guessed H_1	1 (An Error Costs 1 Unit)	0 (No Risk)

TABLE 24. Bit Error Rate Risk Function

The general impact of a cost function on a binary hypothesis decision problem is well documented in the literature.[58] For this thesis's approach, the decision which minimizes the risk changes to the general form below.

$$BER \quad Risk \quad Form \ :$$

$$z = P[H_1] - P[H_0] \underset{H_0}{\overset{H_1}{\gtrless}} 0$$

$$General \quad Risk \quad Form \ :$$

$$z = P[H_1] - P[H_0] \underset{H_0}{\overset{H_1}{\gtrless}} \frac{C(0,1) - C(1,0) + C(0,0) - C(1,1)}{C(0,1) + C(1,0) - C(0,0) - C(1,1)}$$

Equation 69. General Risk Formula

The only assumption made in deriving the General Risk Form was that the sum C(0,1) + C(1,0) - C(0,0) - C(1,1) was positive. If not, the hypothesis decision assignment and inequality directions should be exchanged.

20.7 Decision Directed Adaptive Feedback

A few comments regarding unsupervised learning techniques will be made in this section. First, suppose a network can learn enough about a binary hypothesis problem during supervised training to attain a probability of error ≈ 0.4 or better. Then a technique called decision directed feedback can be used to improve this probability of error during unsupervised operation.[59] [60] In this technique, after supervised training is complete, any decision made on an unknown image is fed back to the update algorithm, and considered a label for that input vector $\mathbf{X}$. This best guess as to which hypothesis was seen, together with the corresponding input vector, is used to determine how the weight vectors should be updated, in a similar fashion as to how weight vectors are updated during the labeled training period. This technique is called decision directed feedback by it's inventor, R. W. Lucky.[61] However, the step size used in the decision directed phase of operation typically has to be at least 100-1000 times smaller than that used in the labeled training period. Thus, in Oja's rule, the α_i must be decreased substantially from that used in labeled training. If not, the noisy estimates of the estimated decision may cause divergence and instabilities. A general rule of thumb for multi-dimensional gradient descent algorithms is to keep the step size below the reciprocal of the largest correlation matrix eigenvalue λ_{max}^{-1}.[62] For the handwritten digits used here, that implied α in the Oja rule should be smaller than $\frac{1}{200} \approx 0.005$. In reality, an slightly higher step size was used at the beginning of a simulation run, as the geometric damping factor of $\frac{1}{iterationnumber}$ for the α_i prevented divergence, and 0.005 was frequently too small to allow for significant convergence before the step size died out. For instance, both the *Six & Seven* image runs and the *Three & Five* image

runs were made starting at a step size of 0.002.[4]

21. Conclusion

The view in this work has been that input data and the output decision of a binary hypothesis computation problem can be interpreted as random variables. The act of computing is to manipulate the often high-dimensional input random variables into a one-dimensional form from which a Yes/No decision can be extracted. Hence the thesis title : *Stochastic Computation.*

Such a picture of computation holds much in common with the generalized statistical communication model. In this model, a signal is drawn from one of two possible ensembles. The drawn signal is corrupted by additive noise, and passed to a receiver. The receiver must determine which ensemble the input was most likely to be drawn from. In computation, the decision process is viewed in a similar fashion. Binary hypothesis computation is the determination of whether input data was drawn from a Yes ensemble versus a No ensemble. The corruption of the ensemble *signals* with additive noise acts to ensure robustness in the face of imperfections in the data collection and data processing stages.

A view of communication and computation as the extraction of a generalized signal from a noise corrupted, often high-dimensional input space allows one to apply the tools of communication and information theory to analyze problems in computation. The assignment of

4. Details of the numeric parameters used in the C code which determined the weight vectors and eigenvalue's can be found in tables in the section on the discussion of the Oja algorithm.

an equivalent Shannon channel capacity to computing architectures and algorithms quantifies the ability of the computation to extract a correct decision from input data. For neural networks, the calculation of the channel capacity leads to an interesting interpretation of network depth versus width. The application of a computational channel capacity to other types of computing architectures and algorithms could also possibly lead to a better, more intuitive interpretation of computation system parameters and tradeoffs.

In conclusion, the generalized communication model and a stochastic view of computation lead to the same mathematical formalism. Borrowing tools from statistical communication and information theory, and applying these to computational problems, and vice versa, is a fruitful approach, and opens new areas for analytic investigation.

REFERENCES

1. Caltech **EE 129 Information And Complexity Course Notes**, By Yaser Abu-Mostafa, 1994

2. **An Introduction To Probability Theory And Its Applications,** By William Feller, Volume I, Third Edition, Revised Printing, Wiley Series In Probability And Mathematical Statistics, John Wiley and Sons, 1968, ISBN 0-471-25708-7, Pages 244 and 256

3. **Mathematical Methods Of Statistics**, By Harald Cramer, Princeton University Press, 1945 (Eighteenth Printing 1991), ISBN 0-691-08004-6, Section 17.5.3, Page 219

4. **Fundamentals Of Digital Image Processing**, By Anil K. Jain, Prentice Hall Publishers, 1989, ISBN 0-13-336165-9, Page 163

5. **Digital Communication Techniques : Signal Design And Detection**, By Marvin K. Simon,, Sami M. Hinedi, and William C. Lindsey, Prentice Hall Publishers, 1995, ISBN 0-13-200610-3, Page 237

6. **Statistical Theory Of Signal Detection**, By Carl W. Helstrom, Second Edition, 1968, Pergamon Press Inc., Page 124

7. **An Introduction To The Theory Of Random Signals And Noise**, Wilber B. Davenport Jr., and William L. Root, The Maple Press Publishing Company, York, PA., 1958, Page 96 (Recently Reprinted By The IEEE Press)

8. **Detection Of Signals In Noise**, By Anthony D. Whalen, 1971, Academic Press

9. **CDROM 1 : USPS Office of Advanced Technology Database of Handwritten Cities, States, ZIP Codes, Digits, and Alphabetic Characters.** The Center Of Excellence for Document Analysis and Recognition (CEDAR), 226 Bell Hall, State University of New York at Buffalo, Buffalo, NY 14260-0001

10. **Little 1200 Handwritten Digit Database**, collected by Isabelle Guyon, A T & T Bell Laboratories, 50 Fremont Street, 6th Floor, San Francisco, CA 94105,

11. CEDAR Database, ibid.

12. **PBM-Plus Image Processing Utilities**, By Jef Poskanzer, 1991

13. **MATLAB**, The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA. 01760

14. **Analysis Of Numerical Methods**, By Herbert Bishop Keller and Eugene Isaacson, Dover Publications, Inc., 1994, ISBN 0-486-68029-0, Page 147

15. **Introduction To Numerical Analysis**, By Josef Stoer and Roland Bulirsch, Springer-Verlag Inc., Second Corrected Printing, 1983, ISBN 0-387-90420-4, Chapters IV, VI and VIII.

16. Feller, ibid.

17. **Introduction To Statistical Pattern Recognition**, By Keinosuke Fukunaga, Second Edition, Academic Press, Inc., 1990, ISBN 0-12-269851-7, Page 31

18. **An Introduction To Information Theory**, By John R. Pierce, Second, Revised Edition, Dover Publications, Inc., 1980, ISBN 0-486-24061-4, Pages 143-144, 164-165

19. **MAPLE V : A Symbolic Manipulation Program**, Waterloo Maple Software, Symbolic Computation Group, Department Of Computer Science, University Of Waterloo, Waterloo, Ontario, Canada, N2L 3G1

20. **Mathematical Statistics**, By John E. Freund and Ronald E. Walpole, Third Edition, Prentice-Hall Inc., 1980, ISBN 0-13-562066-X, Page 232

21. **On The K-Winners Take All Network**, By Eric Majani, Ruth Erlanson, and Yaser. S. Abu-Mostafa, The 1988 IEEE Conference On Neural Information Processing Systems, NIPS 1, Morgan Kaufmann Publishers, ISBN 1-558-60015-9, Page 634

22. **Adaptive Network For Optimal Linear Feature Extractions**, By Peter Földiak, 1989, International Joint Conference On Neural Networks (IJCNN), Volume 1, Pages 401-405, Washington, DC.

23. Hertz et al., ibid., Pages 201 - 209

24. **Introduction To Mathematical Statistics**, By Robert V. Hogg, and Allen T. Craig, Fourth Edition, MacMillan Publishing Co., Inc. 1978, ISBN 0-02-355710-9, Pages 154-161

25. **Mathematical Statistics**, By John E. Freund and Ronald E. Walpole, Third Edition, Prentice-Hall Inc., 1980, ISBN 0-13-562066-X, Pages 179-282

26. **Neural Networks and The Bias / Variance Dilemma**, by Stuart Geman, Elie Bienenstock, and Rene′, Neural Computation, The MIT Press, Volume 4, Number 1, January, 1992, Page 1

27. **Introduction To The Theory Of Neural Computation**, By John A. Hertz, Anders S. Krogh, and Richard G. Palmer, Addison-Wesley Publishing Co., 1991, ISBN 0-201-51560-1, Page

21

28. **An Introduction To Probability Theory And Its Applications**, By William Feller, Volume I, Third Edition, Revised Printing, Wiley Series In Probability And Mathematical Statistics, John Wiley and Sons, 1968, ISBN 0-471-25708-7, Page 174

29. **Neural Networks And Physical Systems With Emergent Collective Computational Abilities**, John J. Hopfield, Proceedings Of The National Academy Of Sciences, Volume 79, April, 1982, Pages 2554-2558. Reprinted In **Neural Networks : Theoretical Foundations And Analysis**, Edited By Clifford Lau, IEEE Press, 1992, ISBN : 0-87942-280-7, Hopfield, 1982, ibid., Page 144, Column 2, Paragraph 1.

30. Hopfield, 1982, ibid., Page 144

31. **The Capacity Of The Hopfield Associative Memory**, by Robert J. McEliece, Edward C. Posner, E. R. Rodemich, and Santosh S. Venkatesh, IEEE Transactions On Information Theory, 1987, Volume 33, Pages 461-482.

32. **Adaptive Filter Theory**, By Simon Haykin, Second Edition, 1991, Prentice Hall Inc., ISBN 0-13-013236-5, Page 147

33. **NIST Handwritten Digit Database(fl3)**, NIST Special Database 1, By Michael Garris, National Institute Of Standards

34. **A Simplified Neuron Model As A Principal Component Analyzer** by Erkki Oja, Journal Of Mathematical Biology, 1982, Volume 15, Pages 267-273

35. **Fast Adaptive Formation Of Orthogonalizing Filters And Associative Memory In Recurrent Networks Of Neuron-Like Elements**, By Teuvo Kohonen and Erkka Oja, Biological Cybernetics, Volume 21, Pages 85 - 95, 1976

36. **Principle Component Analysis By Homogeneous Neural Networks, Part I : The Weighted Subspace Criterion**, By Erkki Oja, Hidemitsu Ogawa, and Jaroonsakdi Wangviwattana, IEICE Transactions On Information And Systems, May, 1992, Volume E75-D, Number 3, Pages 366 - 375

37. **Principle Component Analysis By Homogeneous Neural Networks, Part II : Analysis and Extensions Of The Learning Algorithms**, By Erkki Oja, Hidemitsu Ogawa, and Jaroonsakdi Wangviwattana, IEICE Transactions On Information And Systems, May, 1992, Volume E75-D, Number 3, Pages 376 - 382

38. Oja et al., IEICE, Part II, ibid., Page 377

39. Oja et al., ibid., IEICE, Part II, Page 380, Equation 28

40. Hogg and Craig, ibid.

41. Freund and Walpole, ibid.,

42. Feller, Volume I, ibid.,

43. Hogg and Craig, ibid., Page 175

44. Oja, IEICE II, Page 377-379

45. Davenport and Root, ibid., Page 244

46. **Modern Digital Analog Communications Systems**, by Bhagwandas Pannalal Lathi, 1983, CBS College Publishing, ISBN 0-03-058969-X, Page 503

47. **Digital Communications**, by John G. Proakis, 1983, MaGraw-Hill Inc., ISBN 0-07-050927-1, Page 142-143

48. Haykin, **Adaptive Filter Theory**, ibid, Page 147

49. **Self Organization In A Perceptual Network**, By R. Linkser, Computer, March, 1988, Pages 105-117

50. Davenport and Root, ibid, Chapters 12 and 13, Pages 250 - 311.

51. Davenport and Root, ibid, Page 308

52. Perceptron Convergence Theorem, **Perceptrons : An Introduction To Computational Geometry**, Expanded Edition, By Marvin Minsky and Seymour Papert, 1988, The MIT Press, ISBN 0-262-63111-3, Page 167

53. **Introduction To Communication Science And Systems**, By John R. Pierce, and Edward C. Posner, Plenum Press, 1980, ISBN 0-306-40492-3, (BSC Stuff) Pages 312-313, Chapter 13, Sources And Encoding, Section 13.5 Multidimensional Scaling And The Message Behind The Message, Pages 359-367

54. Haykin, **Adaptive Filter Theory**, ibid, Pages 452 - 471

55. **MUSIC, Maximum - Likelihood, and The Cramer - Rao Bound**, by P. Stoica, and B. Nehorai, 1988, IEEE Transactions On Acoustics, Speech, and Signal Processing, Volume 37, Pages 720 - 741.

56. **Artificial Neural Networks : Approximation and Learning Theory**, by Halbert White, 1992, Blackwell Publishers, ISBN 1-55786-329-6, Pages 79-131

57. White, ibid., Page 84

58. **Detection, Estimation, and Modulation Theory - Part I**, By Harry L. Van Trees, John Wiley and Sons, 1968, ISBN 0-471-89955-0, Page 26

59. Caltech EE 164 - Adaptive Signal Processing - 1994 Course Notes

60. EE 164 Class Project In FIR (Finite Impulse Response) & LMS (Least Mean Square) Decision Directed Feedback

61. **Adaptive Signal Processing**, By Bernard Widrow and Samuel D. Stearns, 1985, Prentice Hall Inc., ISBN 0-13-004029-01, Page 247-248

62. Widrow and Stearns, ibid., Page 59