# Context-Sensitive Vowel Recognition Using the FSCL-LVQ Classifier

A Thesis

Presented in Partial Fulfillment of the Requirements for

the Degree Master of Science in the

Graduate School of The Ohio State University

by

Edward L. Riegelsberger, B. S.

* * * * *

The Ohio State University

1992

Master's Examination Committee:

Ashok K. Krishnamurthy

Stanley C. Ahalt

Approved by

Ashok K. Krishnamurthy

Adviser

Department of Electrical Engineering

In memory

of my grandfather

# Acknowledgments

I must first give a special thanks to Dr. Ashok Krishnamurthy, my advisor, for his endless supply of helpful guidance, patience, and encouragement. I would like to thank Dr. Stanley Ahalt for his interest and helpful suggestions. I also wish to thank Dr. Timothy Anderson for his guidance at the early stages of this work and his continued interest in its outcome. Finally, I give my thanks to The Ohio State University for its support of this research in the form of a Graduate Fellowship.

# VITA

| | |
|---|---|
| September 4, 1967 | Born—Lima, Ohio, U.S.A. |
| June 1990 | B.S. Electrical Engineering,<br>Summa Cum Laude<br>The Ohio State University<br>Columbus, Ohio, U.S.A. |
| 1990 – 1991 | University Fellow<br>The Ohio State University<br>Columbus, Ohio, U.S.A. |
| 1991 – present | Graduate Research Associate<br>Department of Electrical Engineering<br>The Ohio State University<br>Columbus, Ohio, U.S.A. |

# FIELDS OF STUDY

| | |
|---|---|
| Major Field: | Electrical Engineering |
| Studies in: | Signal Processing, Digital Design,<br>Computer Architecture,<br>Artificial Intelligence |

# Table of Contents

# List of Figures

# List of Tables

# CHAPTER I

# INTRODUCTION

This thesis studies the addition of dynamic information to phoneme classification using self-organizing artificial neural networks. Instead of simply concatenating adjacent spectral slices and then learning to distinguish and classify them, we provide additional contextual clues about the preceding and following phonemes. Such a setup significantly reduces the dimensionality of the problem thus allowing for easier training with little loss in classification accuracy.

A set of vowel classification experiments is performed using the FSCL-LVQ classifier. The task is the speaker independent classification of 10 vowels excised from continuous speech taken from the DARPA TIMIT acoustic phonetic database [1]. These experiments have dual purposes in that they address two topics. The first topic concerns the addition of contextual information at the phoneme level. The second topic is a performance comparison of two auditory-model based feature sets with a conventional speech feature representation. An array of experiments is performed and the same collection of results is used to draw conclusions about both topics.

One way of adding dynamic information at the phoneme level to a phoneme classifier is merely to inform the recognizer of the class of the phonemes preceding and following the phoneme in question. This has been done by Leung and Zue [2] using a

multilayer perceptron (MLP) architecture trained using the back–propagation algorithm. Their network is informed of the context phonemes using a string of indicator functions, each indicating one of the possible phoneme classes.

The number of phonemes comprising human language is large. The English language contains roughly 40 phonemes. It is not surprising that in the experiments of Leung and Zue, the dimension of data vectors grow to well over 100 elements. The amount of training data necessary to cover a problem of this high dimensionality is correspondingly large.

A new form for representing phonetic context, place-of-articulation, is presented that helps to reduce the dimensionality problem inherent in the use of phoneme indicators for classifying contextual data. Place of articulation influences vowel features because of the phenomenon of coarticulation. This broad phonetic category offers significant linguistic and acoustic information in a compact size. It is used to reduce the dimensionality of the problem and decrease training set size while retaining a competitive level of performance improvement.

The experiments described herein investigate the application of contextual information in a pattern classification problem using competitive learning networks, specifically those trained with learning vector quantization (LVQ) methods. In a more general sense, some empirical evidence is presented concerning how context works. The results are used to explore questions about how context works and how to effectively integrate different data sets into artificial neural networks.

There are a number of questions regarding the integration of different data in a

single classification scheme. We know that different data sets will have their own intrinsic error metrics, so what is the best way to incorporate these different data sets in a single classification scheme? In non-trivial classification tasks, some problems may arise as more information (context) is added. With the increase in information and dimensionality, a greater number of training samples is necessary to define the joint distribution of the context and data. Also, different data sets may have their own intrinsic error metrics. The merging of two data sets, although statistically straightforward, may not be trivial for a classification algorithm.

In the vowel classification domain, it seems reasonable that the importance of contextual information should be secondary to that of vowel feature information. In the data space, feature information is sufficient to represent the basic organization of vowel clusters. Context merely adds dimensions to refine these clusters. In an attempt to take advantage of this, three different methods for adding context information into a two stage competitive learning algorithm are proposed and evaluated. These methods use training algorithms in which the significance of the feature data is recognized.

The next two chapters introduce speech feature representations and the FSCL-LVQ algorithm respectively. Chapter III also introduces the incorporation of context into the FSCL-LVQ algorithm through the analysis of two simple context problems. Chapter IV presents an array of context-sensitive vowel classification experiments which address questions in phoneme recognition using ANNs, phonemic context, and auditory feature sets. Chapter V provides a discussion of the results followed by the brief summary and conclusions of Chapter VI.

# CHAPTER II

# FEATURE EXTRACTION METHODS

The initial stage of any speech recognition system is a preprocessor which performs feature extraction. Feature extraction transforms raw data into a more compact, robust form. A good feature extractor reduces the raw data leaving only the information pertinent to the task at hand. The quality of the speech feature representation is fundamental to the performance of the entire recognition system. An effective speech feature representation draws out those characteristics which best distinguish and identify speech sounds, and performs robustly within a variety of noise environments. It has been suggested that the utility of a speech feature representation is also dependent upon the speech classification scheme (i.e. one feature representation may work best for HMM recognizers while another is best for certain ANN recognizers.) Therefore, the quality of these feature representations is best evaluated in actual speech classification contexts.

Many feature extractors exist that use conventional signal processing techniques such as LPC coefficients, cepstral coefficients, and DFT coefficients. In the search for alternative methods of speech feature extraction, the human auditory system has been studied and modeled as a natural speech feature extractor, with the expectation that it may lead to a more effective speech feature representation. Two existing

auditory based models are the Synchrony/Mean-rate Model of Stephanie Seneff [3] and Karen Payton's Model of the Auditory Periphery [4]. Both models are multi-stage computational models of the auditory system that emulate the transduction of acoustic vibrations into nerve impulses by mechanisms within the ear. They are fundamentally different in the aspects of the human auditory system that they model. The Payton model simulates the physiological mechanisms of the ear, while the Seneff model reproduces significant properties of the human auditory system based on physiological and psychoacoustic observations. Detailed descriptions of these two models follow.

## 2.1 The Payton Auditory Model

The Payton model of the auditory periphery models the physiological mechanisms of the ear in four stages: the middle ear, the basilar membrane, the secondary sharpening filter, and the hair-cell synapse. It produces twenty frequency dispersed channel outputs and each channel output predicts the firing probability of a population of auditory neurons. A block diagram of the auditory periphery model can be seen in Figure 1[1].

Stage one is a model of the middle ear. Essentially, this stage functions as a linear bandpass filter in which displacement at the tympanic membrane, due to sound pressure at the eardrum, is transduced to stapes velocity at the base of the cochlea. The outer ear and effects of the pinna are not considered.

Stage two is responsible for frequency analysis as performed by the basilar mem-

[1]Figure 1 is taken directly from Figure 1 of Payton's paper.[4]

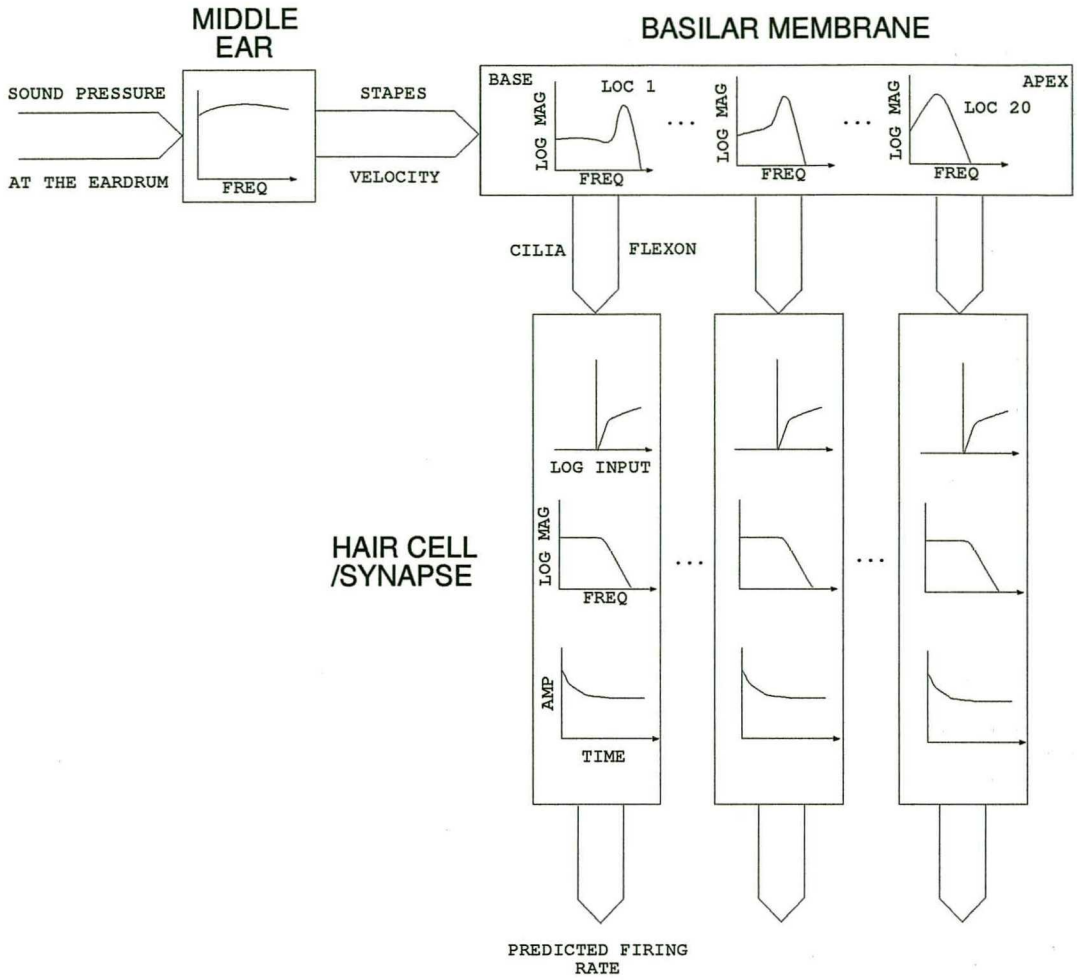


Figure 1: Block Diagram of Payton Auditory Periphery Model

brane. Mechanical motion of the basilar membrane due to excitation by the stapes at the base of the cochlea is described by a two-dimensional model of cochlear mechanics realized through continuous differential equations. These are discretized into difference equations for computation. The motion of the basilar membrane has the property of dispersing spectral energy spatially along the length of the membrane. Utilizing this property, twenty 'taps', equally spaced along the length of the basilar

membrane, form a bank of twenty channels. The rest of the model uses the outputs of these channels.

The linear sharpening filter of stage three fine tunes observed membrane transfer characteristics and also helps to match phase shifts observed in nerve-fiber responses. In an anatomical sense, stages two and three combine to represent the overall performance of the basilar membrane.

The final stage is a hair-cell/synapse model which provides for observed post membrane neural transduction effects. These effects include rectification, logarithmic scaling, low-pass filtering, and adaptation found in hair-cell/synapse responses. Stage four is the only non-linear stage in the model.

## 2.2 The Seneff Auditory Model

The Seneff Synchrony/Mean-rate model is a three stage system which produces two sets of outputs. The model does not attempt to model the exact mechanics of the auditory periphery but instead reproduces relevant properties observed in the auditory system. Figure 2(a)[2] shows how the Seneff model is organized.

The first stage is a linear bank of filters which performs frequency analysis. The distribution of filters and their bandwidths are based on a Bark scale and have been iteratively tuned for frequency response characteristics observed in the basilar membrane. The filters collectively cover a frequency range from about 130Hz to 6400Hz.

The output of each channel of the filterbank is fed to the second stage. Stage II is the hair-cell synapse model which simulates the nonlinear transduction in the cochlea

[2]Figure 2 is taken directly from Figure 1 of Seneff's paper.[3]

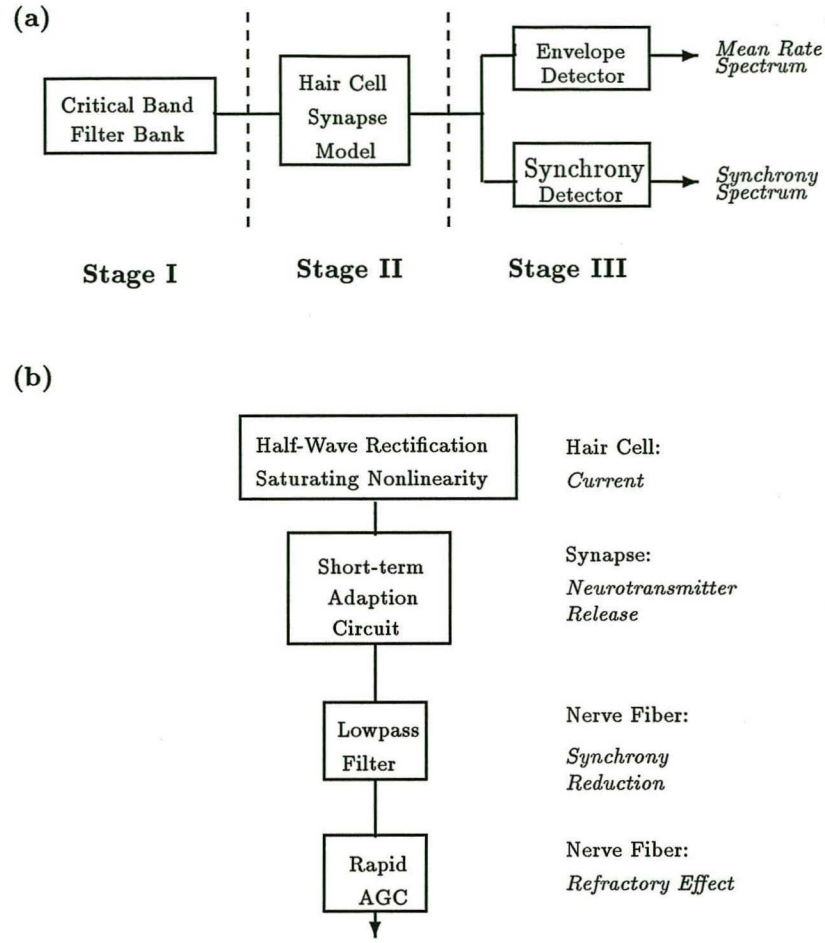


Figure 2: Block Diagram of Seneff Model

and produces an output somewhat similar to auditory-nerve fiber responses. Stage II consists of four serially connected substages: half-wave rectification, short-term adaptation, lowpass filtering, and rapid AGC (automatic gain control). Figure 2(b) shows the organization of the substages of Stage II of the Seneff model. The Stage II combination has been shown by Seneff to imitate auditory phenomena such as short-term adaptation, rapid adaptation, and forward masking. The output of this stage

is not simulated nerve firings but merely a representation of the probability of nerve firing. Since there are a finite number of channels, each channel can be thought to represent a group of spatially located auditory neurons.

Stage III operates on the output of Stage II to produce two sets of outputs: the Mean Rate Spectrogram, and the Generalized Synchrony Spectrogram. It has been suggested by Seneff that the unique qualities of both these outputs can be used in tandem for speech recognition. We believe the synchrony spectrogram alone is sufficient for the classification vowels. Therefore, in this study only the synchrony spectrogram is used.

The Generalized Synchrony Spectrogram is a spectral representation of the input waveform in which features, such as formants, are more easily seen. Each channel of the Generalized Synchrony Spectrogram is produced by the Generalized Synchrony Detector (GSD). The GSD extracts its spectral representation from the Stage II output by measuring the periodicity of each channel (synchrony) with respect to its center frequency. Periodicity is measured as presented in Figure 3[3], which depicts the construction of a single generalized synchrony detector (GSD). Input data in each channel is compared (subtracted) from a delayed version of itself. The delay is set to one-half of the period of the channel's center frequency. If the input is perfectly periodic at the channel center frequency, like a sine wave, the difference should be zero. The output of the GSD is the ratio of the sum to the difference of the input waveform and its delayed equivalent. The greater the periodicity, the smaller the difference, and the larger the output value. Both the sum and the difference measures

[3]Figure 3 is taken directly form Figure 11(a) of Seneff's paper.[3]

are passed through integrators (summers) that act to smooth the GSD over time. A saturating half-wave non-linearity is located at the output to limit the magnitude of the GSD in cases of high periodicity. A threshold value is included to reduce some weak periodicities that can be characterized as the spontaneous discharge rate.

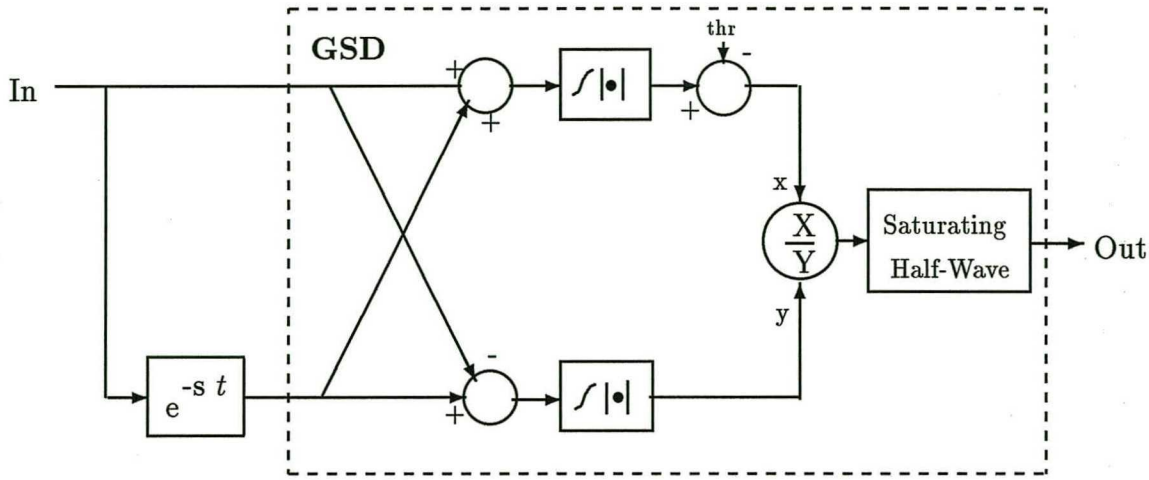


Figure 3: Schematic of Generalized Synchrony Detector

## 2.3 LPC Weighted Cepstral Coefficients

Due to the formant/resonance properties of the vocal tract, speech signals have been found to be well modeled by all pole systems. Linear prediction analysis, an all-pole modeling technique, is a commonly used method for speech feature extraction. A related but alternative technique for modeling the short time speech spectrum is based on cepstrum analysis. Cepstrum analysis, a nonlinear signal processing method, can be used to transform linear prediction coefficients (LPC) derived using

linear prediction analysis into cepstral coefficients. Cepstral coefficients have been found to be an equally viable alternative for speech feature representations.

The LPC spectrum of a signal can be estimated using a number of different methods. Most are based on some type of estimation of the signal's autocorrelation function (ACF). With a reasonable estimate of the ACF, the linear prediction coefficients can be found easily using the Yule-Walker equations. An efficient algorithm, named the Levinson algorithm, exists for recursively solving this set of equations. We derived linear prediction coefficients using what is commonly know as the autocorrelation method. In this method, a biased estimator is used to calculate the ACF. The linear prediction coefficients are then found using the Levinson algorithm.

Cepstral coefficients can be calculated from the LPC spectrum using the recursive formula of Equation 2.1. This equation converts the p-th order LPC coefficients $(a_k, 1 \leq k \leq p, a_0 = 1)$ to the q-th order LPC cepstrum coefficients $(c_k, 0 \leq k \leq q)$. The variable $\varepsilon$ represents the residual error from the linear prediction analysis.

$$c_k = \begin{cases} \frac{1}{2}\log\varepsilon & k = 0 \\ -a_k - \frac{1}{k}\sum_{n=1}^{k-1}(k-n)c_{k-n}a_n & k > 0 \end{cases} \tag{2.1}$$

To improve on the utility of cepstral coefficients, Juang, Rabiner, and Wilpon suggest the signal processing technique of "liftering". This technique has been found "to reduce the variability of the statistical components of LPC-based spectral measurements" [5]. The liftering procedure performs windowing in the cepstral domain to reduce variability. They report improved results when using liftering in several speech recognition tasks using Euclidean distance as a distortion measure. The resulting windowed coefficients are referred to as weighted cepstral coefficients. A number of

liftering windows are proposed by Juang, Rabiner, and Wilpon, and the raised sine lifter of Equation 2.2 seems most promising. This window is a function of window length $L$, which is also cepstral order.

$$w(k) = 1 + 0.5L \sin \frac{k\pi}{L}, \qquad k = 1 : L \tag{2.2}$$

For these three feature representations to be examined in a comparative fashion, the number of channels/coefficients in each must be chosen fairly. Both the Payton and Seneff models are designed with 20 channels. This channel size is within the bounds of those seen in other works. Typically, LPC and WCC orders are kept low, around order 12. Overly high model orders in all pole models can result in spurious peaks that are detrimental to classification tasks. Sixteen LPC weighted cepstral coefficients is a reasonable model order that is not too large yet is near to the channel size of the auditory models. The model orders selected for all three feature representations are believed to be sufficient and competitive without giving advantage to any one representation.

# CHAPTER III

# FSCL-LVQ

In recent years artificial neural networks (ANNs) have been applied to many signal processing tasks including speech and image recognition. Inspired by biological neural systems, these networks consist of numerous highly interconnected small computational elements called neurons. Typically, specialized algorithms are used to "train" these networks to perform tasks, often in the pattern classification domain. Much has been written covering the broad topic of neural networks. The reader is referred to "An Introduction to Computing with Neural Nets" by Richard Lippmann[6] for a general introduction to artificial neural networks.

The neural network architectures used in this document are based on competitive learning schemes. In these networks, neurons are defined as vectors within an input space. Neurons compete in a winner-take-all contest in which the neuron $\mathbf{w_c}$ nearest to an input vector $\mathbf{x}$, with respect to some error metric $d(\mathbf{x}, \mathbf{w_i})$, wins.

$$d(\mathbf{x}, \mathbf{w_c}) = \min_i \{d(\mathbf{x}, \mathbf{w_i})\} \tag{3.1}$$

In the performance phase of a competitive learning network, the index of the winning neuron is typically the output of the network. In the training phases, a learning rule is typically used that iteratively modifies the position of a neuron based

on the distance between it and a training vector, according to some error metric. Equation 3.2 is an example of a generic competitive learning rule where a neuron $\mathbf{w}$ is modified in response to an input vector $\mathbf{x}$.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{w}(t)] \tag{3.2}$$

The time varying scalar variable $\alpha$ usually decreases over time so that the network eventually converges.

Keeping with terminology used in vector quantization (VQ), neurons are referred to as reference vectors or codewords. Each codeword is a vector of the same dimension as the input vector. In context discussions, groups of elements within these vectors are referred to as fields. Entire collections of codewords are called codebooks.

The ANN used in this investigation of context-sensitive vowel recognition is a two stage algorithm called FSCL-LVQ. The first stage, Frequency-Sensitive Competitive Learning (FSCL), is an unsupervised competitive learning algorithm that incorporates a "fairness" function to insure even utilization of all codewords. The second stage, Learning Vector Quantization (LVQ), is a supervised algorithm which modifies the FSCL codebook to fine tune decision boundaries and improve classification performance. A more detailed discussion of both algorithms is given in the next two sections. The final section introduces the use of these algorithms in context-sensitive problems.

## 3.1 FSCL

Frequency-Sensitive-Competitive-Learning (FSCL) [7] is a type of self-organizing competitive learning network much like Kohonen's self-organizing feature map[8]. FSCL incorporates into its training algorithm a "fairness" function which adds sensitivity to the frequency of codeword use. As a result, codebooks are generated with more evenly utilized codewords.

FSCL associates with each codeword $\mathbf{w_i}$ a variable $u_i$ that represents the of the number of times that codeword has been winner. This win count variable is used within a fairness function $\mathcal{F}(u_i)$ to augment a standard error metric $d(\mathbf{x}, \mathbf{w_i})$ and create a frequency sensitive error metric $d^*(\mathbf{x}, \mathbf{w_i})$. $\mathcal{F}(u_i)$ is an increasing function of $u_i$ which serves to de-emphasize overly used codewords.

$$d^*(\mathbf{x}, \mathbf{w_i}) = \mathcal{F}(u_i) d(\mathbf{x}, \mathbf{w_i}) \tag{3.3}$$

The winning codeword is modified according the learning rule of Equation 3.4. The scalar variable $\epsilon$ is the learning rate, which monotonically decreases to zero over time. All other codewords remain unchanged.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \epsilon(t)[\mathbf{w}(t) - \mathbf{x}(t)] \tag{3.4}$$

Even though FSCL's codebook selection may not be optimal with respect to any conventional error metric, it consistently results in a codebook with codewords that are more evenly distributed throughout the data space.

## 3.2 LVQ

Learning Vector Quantization (LVQ) [8] is a technique which improves the performance of self-organizing networks in classification tasks. Typically, it starts with a codebook trained by some unsupervised learning technique and iteratively fine tunes the decision boundaries defined by the classes and locations of codewords. The resulting network performs classification simply by using a nearest-neighbor selection criterion; the class of the closest (winning) codeword is chosen to be the class of the test vector. Several variations on this basic LVQ algorithm have been proposed. The most common are LVQ1, LVQ2, and LVQ3, all proposed by Kohonen. All create decision regions that are near-optimal, although piecewise linear.

### 3.2.1 LVQ1

The LVQ1 algorithm is very similar to the basic competitive learning update scheme given in Equation 3.2 with a slight modification that incorporates classification to the scheme.

For a given training token $\mathbf{x}$ with class $c_x$, nearest neighbor selection of a codebook occurs as described in Equation 3.1. The winning codeword $\mathbf{w_c}$ with class $c_c$ is then updated according to the rules of Equations 3.5 and 3.6.

If $c_c = c_x$ ($x$ is classified correctly),

$$\mathbf{w_c}(t+1) = \mathbf{w_c}(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{w_c}(t)] \tag{3.5}$$

If $c_c \neq c_x$ ($x$ is classified incorrectly),

$$\mathbf{w_c}(t+1) = \mathbf{w_c}(t) - \alpha(t)[\mathbf{x}(t) - \mathbf{w_c}(t)] \tag{3.6}$$

The scalar coefficient $\alpha(t)$ is a monotonically decreasing function of time. Typically, $\alpha(t)$ is initialized around 0.01 and decreases linearly to zero over 10000 or more iterations.

In simple examples, the decision boundaries created by LVQ1 has been demonstrated to coincide closely with those of a Bayes classifier. One effect of the LVQ1 algorithm is the reduction of the point density of the codewords around the Bayesian decision boundaries. This "depletion region" of codewords does not appear to be detrimental.

### 3.2.2 LVQ2

The LVQ2 algorithm attempts to better approximate Bayes decision boundaries by making adjustments to pairs of codewords that incorrectly define a descrimination surface. This algorithm does not exhibit "depletion regions" as produced in LVQ1.

For each training token $\mathbf{x}$ with class $c_x$, LVQ2 uses a nearest neighbor selection scheme to choose the closest (winning) codeword $\mathbf{w_w}$ with class $c_w$, and the second closest (runner up) codeword $\mathbf{w_r}$ with class $c_r$. If the class of $\mathbf{x}$, $c_x$, is different from the winning class $c_w$ but the same as the runner up class $c_r$ then the codewords are modified according to Equation 3.7. In all other cases, no adjustment is made.

If $(c_x \neq c_w)$, $(c_x = c_r)$ and $x$ falls within the "window",

$$\begin{aligned} \mathbf{w_w}(t+1) &= \mathbf{w_w}(t) - \alpha(t)[\mathbf{x}(t) - \mathbf{w_w}(t)] \\ \mathbf{w_r}(t+1) &= \mathbf{w_r}(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{w_r}(t)] \end{aligned} \tag{3.7}$$

For the rule of Equation 3.7 to apply, the training vector $\mathbf{x}$ must fall within a

"window" defined in terms of the relative distances $d_w$ and $d_r$ from $\mathbf{w_w}$ and $\mathbf{w_r}$ respectively. This "window" criterion is defined in Equation 3.8.

Token $\mathbf{x}$ is within the "window" if

$$min(d_w/d_r, d_r/d_w) > s, \qquad s = (1-w)/(1+w) \tag{3.8}$$

Optimal window width $w$ depends on the number of available training tokens. A window width of 0.2 (20%) is a reasonable value for training with a small number of training tokens.

The LVQ2 algorithm has be shown to improve classification accuracy by shifting decision surfaces toward the Bayes decision surface. If LVQ2 is performed for too many iterations, the initial improvements can be lost due to codewords "drifting away". This occurs because during every update, the correction of the correct class codeword is always of greater magnitude than that of the incorrect class codeword causing the distance between the two codewords $\| w_r - w_w \|$ to decrease monotonically.

### 3.2.3 LVQ3

The LVQ3 algorithm combines elements from both LVQ1 and LVQ2 to produce a very stable and efficient algorithm. Like LVQ2, for each training token $\mathbf{x}$ with class $c_x$, LVQ2 uses a nearest neighbor selection scheme to choose the closest two codewords $\mathbf{w_i}$ with class $c_i$, and $\mathbf{w_j}$ with class $c_j$. If the class of $\mathbf{x}$ is the same as one of the winning codeword classes and different from the other, an update scheme similar to LVQ2 is employed with the same window criterion of Equation 3.8. If the classes of

both codewords are the same as the class of $\mathbf{x}$, then an update rule similar to that used by LVQ1 is applied.

If $(c_x \neq c_i)$, $(c_x = cj)$ and $\mathbf{x}$ falls within the "window",

$$\begin{aligned} \mathbf{w_i}(t+1) &= \mathbf{w_i}(t) - \alpha(t)[\mathbf{x}(t) - \mathbf{w_i}(t)] \\ \mathbf{w_j}(t+1) &= \mathbf{w_j}(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{w_j}(t)] \end{aligned} \tag{3.9}$$

For $k \in \{i, j\}$, and $c_x = c_i = c_j$

$$\mathbf{w_k}(t+1) = \mathbf{w_k}(t) + \epsilon\alpha(t)[\mathbf{x}(t) - \mathbf{w_k}(t)] \tag{3.10}$$

The scalar learning constant $\epsilon$ in Equation 3.10 is dependent upon the size of the window, being smaller for narrower windows. Reasonable values for $\epsilon$ range between 0.1 and 0.5. Optimal value for $\epsilon$ depend on the window width and the size of the training set.

LVQ3 is self-stabilizing in that its optimal codeword placement does not change with additional learning. The incorporation of corrections (Equation 3.10) that ensure the continued rough approximation of the class distributions are responsible for this stabilizing property. They are also responsible for some reduction in classification performance. This is because the corrections that maintain class distributions interfere with corrections that improve classification performance.

In the FSCL-LVQ algorithm, the assignment of classes to the codewords occurs after the FSCL phase but before the LVQ phase. This is done in such a way that all of the possible classes are represented by at least one codeword. In cases where the number of codewords is much greater than the number of classes, a typically sufficient

rule is a majority voting scheme in which the class with most training vectors nearest to a given codeword is chosen to be the class of that codeword.

## 3.3 Context-Sensitive FSCL-LVQ

In Kohonen's paper, "The Self-Organizing Map" [8], the idea of context-sensitive learning is introduced as a method for topographically mapping symbolic items in a way that represents their logical similarities. Kohonen suggests that these logical similarities can be drawn from the context of the symbol. To incorporate contextual information into the learning process, the symbolic expression of an item $x_s$ and the representation of its context $x_c$ can be combined to form a vector sum of two orthogonal components as shown in Equation 3.11.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x_s} \\ \mathbf{x_c} \end{bmatrix} = \begin{bmatrix} \mathbf{x_s} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{x_c} \end{bmatrix} \tag{3.11}$$

In the symbol mapping problem, Kohonen signified the importance of weighting the symbol and context parts properly such that "the norm of the context part predominates over that of the symbol part during the self-organizing process". Due to the nature of some other task, in may be equally advantagous to assign weighting such that the symbol/data part predominates over the context part. In both cases, the need to emphasize some aspect of the data over another during the self-organization process is recognized.

The nature of the context information as well as the form in which it is presented will influence its degree of emphasis with respect to the symbol information. Contextual information can come in both continuously valued and discretely valued forms.

Examples of continuously valued contexts include temperature or pressure in sensitive physical measurements. Possible discrete valued contexts include word or symbol indices, or vectors of binary indicators that signify arbitrary class memberships.

In the next two sections, two simple experiments are presented that explore the addition of context in a continuous domain and in a mixed continuous/discrete domain using FSCL-LVQ. In these experiments, we wish to establish some empirical evidence of how contextual information manifests itself in competitive learning networks. A better understanding of the relation between context and symbol information may lead to more effective ways of emphasising certain information during the self-organization process. It may also suggest better ways of incorporating contexual information into the learning process.

### 3.3.1 Context in Continuous Domain

A binary classification experiment is presented that uses two dimensional data. Both distributions are Gaussian in $x$ and uniformly distributed in $y$ over the range $[0, 1)$. Equations 3.12 and 3.13 define the probability density functions $f_{C1}$ and $f_{C2}$ of Class 1 and Class 2 respectively.

$$C1: \qquad f_{C1}(x,y) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left\{\frac{-1}{2\sigma_1^2}(x-\mu_1)^2\right\} \tag{3.12}$$

$$C2: \qquad f_{C2}(x,y) = \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left\{\frac{-1}{2\sigma_2^2}(x-\mu_2)^2\right\} \tag{3.13}$$

$$p_{C1} = Prob\{x \in C1\}$$

$$p_{C2} = Prob\{x \in C2\}$$

$$p_{C1} = p_{C2} = 0.5 \tag{3.14}$$

The symbols $\sigma_1$ and $\sigma_2$ represent the standard deviations of $p_{C1}(x,y)$ and $p_{C2(x,y)}$ respectively. The symbols $\mu_1$ and $\mu_2$ represent the means of $p_{C1}(x,y)$ and $p_{C2}(x,y)$ respectively. Mean $\mu_2$ is defined to be a linear function of $y$ as given in Equation 3.15. The parameters $m$ and $c$ can be adjusted to control the amount of overlap between $p_{C1}(x,y)$ and $p_{C2}(x,y)$. Mean $\mu_1$ is constant with respect to $y$.

$$\mu_2 = my + c \tag{3.15}$$

For the purposes of investigating context, the $x$ components of the distributions are defined to be the "data" portion and the $y$ components the context. Therefore context-independent studies make use of only the $x$ component. Context-dependent studies use both the $x$ and $y$ components.

Optimal classification criterion is defined using Bayes classifiers. The Bayes risk $R_B$ for simple binary hypotheses is given in Equation 3.16.

$$R_B = p_{C1}L_{C1}\alpha + p_{C2}L_{C2}(1-\beta) \tag{3.16}$$

The symbols $\alpha$ and $(1-\beta)$ represent the false alarm and miss probabilities respectively. $L_{C1}$ and $L_{C2}$ are the loss coefficients associated with incorrectly choosing Class 1 or Class 2 respectively. Bayes risk $R_B$ for a simple binary hyphothesis is minimized by the likelihood ratio test of Equation 3.17 where $dF_{C1}(x,y) = f_{C1}(x,y)dx$ and $dF_{C2}(x,y) = f_{C2}(x,y)dx$.

$$\phi(x) = \begin{cases} 1, & dF_{C1}(x,y)/dF_{C2}(x,y) > p_{C1}L_{C1}/p_{C2}L_{C2} \\ 0, & otherwise \end{cases} \tag{3.17}$$

By making equal the loss coefficients $L_{C1}$ and $L_{C2}$ and noting that $p_0 = p_1$ as stated in the problem definition of Equation 3.14, the likelihood ratio test can be simplified

into the form of Equation 3.18.

$$\phi(x) = \begin{cases} 1, & dF_{C1}(x,y)/dF_{C2}(x,y) > 1 \\ 0, & otherwise \end{cases} \tag{3.18}$$

The corresponding decision boundary is given in Equation 3.19.

$$f_{C1}(x,y) = f_{C2}(x,y) \tag{3.19}$$

Accuracy for a given class is determined by integrating the probability density function of that class over the classification region defined for that class. Optimal percentage accuracy for given set of classification boundaries is found by summing the individual accuracy for all the classes, weighted by their probability of occurance. In the context-sensitive example, optimal percentage accuracy for a single boundary decision rule is given in Equation 3.20

$$p_{C1}\int\int_{\phi(x)=1} f_{C1}(x,y)dxdy + p_{C2}\int\int_{\phi(x)=0} f_{C2}(x,y)dxdy \tag{3.20}$$

In the context independent case, the resulting probability density functions are the marginal distributions $g_{C1}(x)$ and $g_{C2}(x)$ of $f_{C1}(x,y)$ and $f_{C2}(x,y)$ with respect to $x$ of the original two dimensional density functions.

$$C1: \qquad g_{C1}(x) = \int_0^1 \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left\{\frac{-1}{2\sigma_1^2}(x-\mu_1)^2\right\} dy \tag{3.21}$$

$$C2: \qquad g_{C2}(x) = \int_0^1 \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left\{\frac{-1}{2\sigma_2^2}(x-\mu_2)^2\right\} dy \tag{3.22}$$

The optimal decision boundary for the context independent case is given in Equation 3.23.

$$g_{C1}(x) = g_{C2}(x) \tag{3.23}$$

Optimal percentage accuracy is calculated using Equation 3.24 for the context independent case, where $b$ is the optimal Bayes decision boundary.

$$0.5 \int_{-\infty}^{b} g_{C1}(x)dx + 0.5 \int_{b}^{\infty} \int g_{C2}(x)dx \tag{3.24}$$

MATLAB [10] is used extensively to calculate the marginal distributions, optimal classification boundaries and optimal classification accuracies defined in Equations 3.19 through 3.24. Integration is performed numerically using a rectangular rule to discretize the density functions.

Some experimentation is necessary to determine values for $\mu_1$, $m$, $c$, $\sigma_1$, and $\sigma_2$ that yield interesting results. Interesting results have a statistically significant difference in classification accuracies between the context independent and context dependent cases and fall short of perfect classification by at least 5%. The resulting parameter values are as follows: $\mu_1 = 0.0$, $m = 8.0$, $c = 1.68$, $\sigma_1 = 4.0$, and $\sigma_2 = 1$.

With the given parameter values, the optimal context-sensitive boundary is calculated and shown in Figure 4. The curve appears to be a straight line but actually is curved slightly. The optimal classification accuracy for this case is 87.33%. Figure 5 plots the calculated marginal distributions for class 1 and class 2. Their intersection at $x = 2.174$ is the optimal Bayes boundary and yields an optimal classification accuracy of 81.01% for the context independent case.

Experiments are performed using the FSCL-LVQ paradigm on both the context dependent and context independent cases. The four codewords used in the network are sufficient to create near optimal decision boundaries. Experimental results can be found in Table 1 and Table 2. These tables show classification accuracy for different

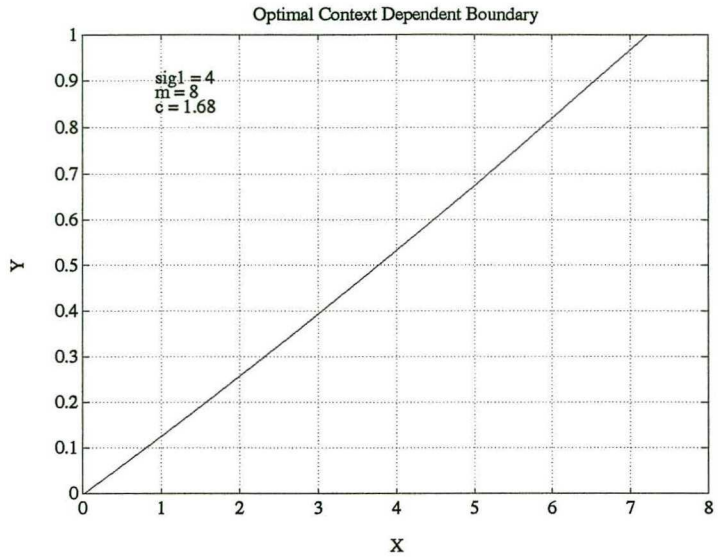


Figure 4: Optimal Context-Sensitive Decision Boundary

numbers of iterations and different learning rates. Both LVQ2 and LVQ3 are used in the LVQ stage of training. It results indicate that LVQ2 performs better than LVQ3 in the context-sensitive classification task.

A closer look at the location of the codewords in the LVQ2 trained network found them ouside of the of the data space ($y > 1, y < 0$). Regardless, the decision boundaries defined by these codewords were close to the optimal Bayes decision boundaries and as a result performed well. The LVQ3 trained network had all its codewords placed withing the data space, although the decision boundaries defined by those codewords were poor. For either method to get good decision boundaries, the codwords have to be placed well outside the areas of high probability density. This is no problem for LVQ2, but it conflicts with the strategies of LVQ3, which attempt to maintain a rough estimate of the probability density function.

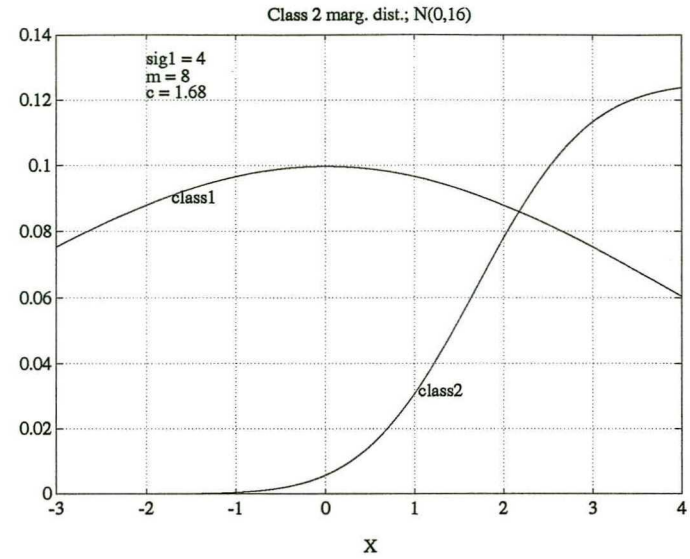


Figure 5: Marginal Distributions for Context-Independent Classification. The intersection of the two curves defines the optimal Bayes boundary.

In addition to illustrating the use of LVQ2 and LVQ3 in the FSCL-LVQ paradigm, this experiment demonstrates the fallibility of LVQ methods if not used with care. When the number of codebooks is small, the nonlinearity of codebook selection task becomes more apparent through convergence to local minima. A significant question is whether these LVQ methods converge to the optimal Bayes decision boundaries as the number of codewords increase to infinity. We do not know of such a proof in the literature and intend to investigate this question in the future.

From a context-sensitive learning point of view, the addition of context information produces results near to the maximum theoretically possible. The networks, when trained properly, are learning the joint distribution of the context and the data information. This observation raises a question. Is there any difference between

Table 1: Continuous Domain Example: Context Dependent Results

| lvq alg. | lrn. rates | # iter | % accuracy |
|---|---|---|---|
| LVQ2 | 0.005/- | 70 | 86.17 |
| LVQ2 | 0.005/- | 50 | 87.17 |
| LVQ2 | 0.003/- | 50 | 87.00 |
| LVQ2 | 0.003/- | 80 | 86.83 |
| LVQ2 | 0.003/- | 35 | 81.17 |
| LVQ2 | 0.007/- | 45 | 87.50 |
| LVQ2 | 0.007/- | 60 | 84.83 |
| LVQ2 | 0.007/- | 35 | 88.00 |
| LVQ3 | 0.01/0.2 | 200 | 80.50 |
| LVQ3 | 0.01/0.1 | 100 | 80.33 |
| LVQ3 | 0.005/0.3 | 300 | 80.50 |
| LVQ3 | 0.005/0.2 | 50 | 80.17 |
| LVQ3 | 0.005/0.02 | 50 | 82.33 |
| LVQ3 | 0.01/0.05 | 80 | 80.33 |

context-sensitive learning and the learning of joint distributions? The remainder of this thesis, especially Chaper V, attempts to answer this question.

Table 2: Continuous Domain Example: Context Independent Results

| LVQ alg. | lrn. rates | # iter | % accuracy |
|---|---|---|---|
| LVQ2 | 0.007/- | 35 | 80.50 |
| LVQ2 | 0.007/- | 45 | 80.33 |
| LVQ2 | 0.003/- | 50 | 80.17 |
| LVQ2 | 0.005/- | 40 | 80.50 |
| LVQ3 | 0.01/0.05 | 150 | 80.33 |
| LVQ3 | 0.005/0.02 | 150 | 80.50 |
| LVQ3 | 0.005/0.02 | 50 | 80.83 |

### 3.3.2 Context in Mixed Domain

A second binary classification is presented in which context information is binary in nature. Both classes of distributions are uniform in two dimensions.

$$C1: \quad f_{C1}(x,y) = 0.5; \quad \{0.5 < x < 1.5, 0.5 < y < 1.5\}, \{2.0 < x < 3.0, 1.0 < y < 2.0\} \tag{3.25}$$

$$C2: \quad f_{C2}(x,y) = 0.5; \quad \{0.5 < x < 1.5, 1.0 < y < 2.0\}, \{2.0 < x < 3.0, 0.5 < y < 1.5\} \tag{3.26}$$

$$Prob\{x \in C1\} = Prob\{x \in C2\} = 0.5 \tag{3.27}$$

Context information:

$$Contextclass \;\; 1: \;\; \gamma = 0, \;\; \{1.0 < y < 2.0\} \tag{3.28}$$

$$Contextclass \;\; 2: \;\; \gamma = 1, \;\; \{0.5 < y < 1.5\} \tag{3.29}$$

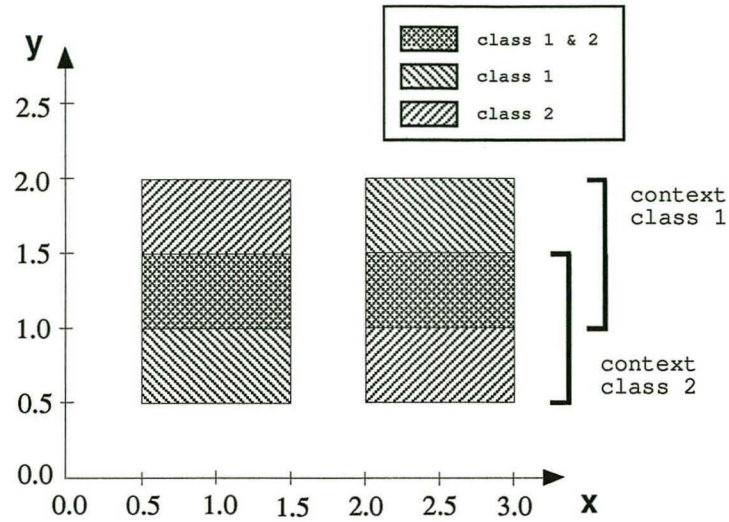


Figure 6: Class Distributions for Mixed Domain Context Example

Data is generated using MATLAB according to the expressions of the probability density functions $f_{C1}(x, y)$ and $f_{C2}(x, y)$ given in Equations 3.25 and 3.26. Data points are produced in each of four subregions as shown in Figure 6, defined by the sample class and a binary context variable $\gamma \in \{0, 1\}$.

$$\begin{aligned}
&regionA: \quad \{0.5 < x < 1.5, 1.0 < y < 2.0\} \quad class: \quad C2, \quad \gamma = 0 \\
&regionB: \quad \{0.5 < x < 1.5, 0.5 < y < 1.5\} \quad class: \quad C1, \quad \gamma = 1 \\
&regionC: \quad \{2.0 < x < 3.0, 1.0 < y < 2.0\} \quad class: \quad C1, \quad \gamma = 0 \\
&regionD: \quad \{2.0 < x < 3.0, 0.5 < y < 1.5\} \quad class: \quad C2, \quad \gamma = 1
\end{aligned} \tag{3.30}$$

Without context information, an optimal Bayes decision rule for detecting class 1

is given in Equation 3.31 and yields an optimal classification accuracy of 75%.

$$\begin{aligned} x < 1.25, \quad & \{0.5 < y < 1.5\} \\ x > 1.25, \quad & \{2.0 < y < 3.0\} \end{aligned} \tag{3.31}$$

With context information, the problem is completely specified. The optimal decision rule for detecting class 1 of Equation 3.32 yields 100% classification accuracy.

$$\begin{aligned} \gamma = 0, \quad x < 1.0 \quad & \{0.5 < y < 1.5\} \\ \gamma = 0, \quad x > 1.0 \quad & \{2.0 < y < 3.0\} \\ \gamma = 1, \quad x > 1.5 \quad & \{0.5 < y < 1.5\} \\ \gamma = 1, \quad x > 1.5 \quad & \{2.0 < y < 3.0\} \end{aligned} \tag{3.32}$$

Training using FSCL-LVQ using a codebook of eight codewords produces classifiers with 100% accuracy in the context dependent case and a near optimal 74.1% accuracy in the context dependent case. FSCL training places two codewords into both disjoint regions. LVQ training properly adjusts the decision boundaries to optimize classification accuracy although as a result, the codewords sometimes move outside of the regions of uniform distribution.

This example demonstrate learning with discretely valued context. We use this type of context in the vowel classification experiments of the next chapter. LVQ2 and LVQ3 handle the discrete values with no problems. Again, training simply learns the joint distribution of the context and data information. Although it is less clear in

this example, the joint distribution is defined in three dimensions: the XY data plane and the context dimension.

# CHAPTER IV

# EXPERIMENTS IN CONTEXT-SENSITIVE VOWEL CLASSIFICATION

These experiments have dual purposes in that they address two topics. The first topic is a performance comparison of two auditory-model based feature sets with a conventional speech feature representation. The second topic concerns the addition of contextual information at the phoneme level. A single set of experiments is performed and the same collection of results is used to draw conclusions about both topics.

This section presents details of the experiments by first explaining the auditory/conventional feature set comparison aspects of the experiment. Details about the context part of the experiments follow. Note that even though the feature set discussion is separate from the context discussion, the experimental results are generated from one set of procedures.

Speech data for all of the experiments is taken from a subset of the DARPA TIMIT Acoustic Phonetic Database. This database is organized with phonetic transcriptions which allow for ease in vowel extraction and in the determination of adjacent phoneme types. Occurrences of seven vowels ($a$, æ, $\Lambda$, $\varepsilon$, ɚ, I, $i$) and three diphthongs ($a^y$, $e^y$, $o^w$) extracted from the continuous speech of ten talkers (7 male/3 female) form the data set used in this study. The data set, containing approximately 703 tokens, is rather

small. We believe that a data set of this size is sufficient to provide a good qualitative feel for the results. To make use of this data set, and provide talker independence, the following testing methods are used. For talker independent experiments, data from one talker is used for testing and data from the remaining nine talkers is used for training. A composite performance measure is computed as the mean of the performance measures with each of the ten talkers being the test talker.

## 4.1 Comparison of Speech Representations

This subsection explains the implementation details of three speech feature set representations. Speech feature sets for training and testing are derived from three different sources: the synchrony response of Seneff's auditory model, the auditory model proposed by Payton, and LPC derived weighted cepstral coefficients. Comparisons of classification results for each of the speech feature representations are discussed.

The Seneff speech feature sets are obtained from the output of the generalized synchrony detector. The Seneff model program we use produces 40 channels of output. By averaging pairs of adjacent channels, the output is reduced to 20 channels that span the same frequency range. A feature vector is generated for each vowel by averaging spectral slices from the middle third of the duration of the vowel, and then normalizing with respect to energy.

A Payton model based feature set is generated for each vowel by averaging the spectral output over the middle third of the duration of the vowel and then normalizing with respect to energy. For more distinguishable Payton model feature sets, it is necessary to subtract the spontaneous firing rate from each of the twenty output

Table 3: Composite Percentage Classification Error: Feature Representations

| Context Method | Feature Set | Training Set | Testing Set |
|---|---|---|---|
| No Context | Seneff | 23.11% | 51.74% |
| | Payton | 28.14% | 53.69% |
| | WCC | 25.51% | 58.71% |
| Method A | Seneff | 18.73% | 48.21% |
| | Payton | 20.73% | 46.48% |
| | WCC | 20.53% | 48.20% |
| Method B | Seneff | 18.32% | 47.64% |
| | Payton | 21.79% | 49.49% |
| | WCC | 21.11% | 48.40% |
| Method C | Seneff | 22.54% | 49.15% |
| | Payton | 24.62% | 47.66% |
| | WCC | 22.74% | 47.67% |

channels prior to averaging and normalization.

The third group of speech feature sets is derived from weighted cepstral coefficients. For each vowel, the first 16 LPC coefficients are calculated using the autocorrelation method over the middle third of the vowel's duration. These LPC coefficients are then used to generate the 16 cepstral coefficients which comprise the feature set.

Recognition tests using the FSCL/LVQ classifier are performed using all three speech representations. The networks are set up with 64 codewords with each codeword size being the size of the feature vector. FSCL training is performed for 1000 iterations, with an initial learning rate of 0.01, linearly decreasing to zero. LVQ training is performed for 500 iterations, with a window width of 0.25 and an initial learning rate of 0.05, which linearly decreases to zero.

Classification results can be seen in Table 3. Methods A, B, and C are three different procedures for introducing context into the network. The details of these methods are given in the following subsection. Without regard to the details of the context-dependent methods, we can still use their results in a comparative fashion to evaluate the performance of the speech feature representations.

## 4.2 Addition of Context at Phoneme Level

In this set of experiments, context information is added to the feature sets described above and vowel classification tests are performed to study the effect of context on performance. Three methods for adding context to a competitive learning network are proposed and evaluated.

Table 4: Composite Percentage Classification Error: Context Methods

| Feature Set | Context Method | w/o context | method A | method B | method C |
|---|---|---|---|---|---|
| Seneff | training set | 23.11% | 18.73% | 18.32% | 22.54% |
| | testing set | 51.74% | 48.21% | 47.64% | 49.15% |
| Payton | training set | 28.14% | 20.73% | 21.79% | 24.62% |
| | testing set | 53.69% | 46.48% | 49.49% | 47.66% |
| WCC | training set | 25.51% | 20.53% | 21.11% | 22.74% |
| | testing set | 58.71% | 48.20% | 48.40% | 47.67% |

There is a variety of context information that can be used to enhance the performance of a speech recognition system. In phoneme recognition applications, it seems reasonable that information about the phonemes preceding and following the phoneme in question would be useful. It is in this way that we define context.

One way for adding context at the phoneme level to a phoneme classifier is merely to inform the recognizer of the class of the phoneme preceding and following the phoneme in question. This has been done by Leung and Zue [2] using a multilayer perceptron (MLP) architecture trained using the back–propagation algorithm. Their network is informed of the context phonemes using a string of indicator functions, each indicating one of the possible phoneme classes. The number of phonemes comprising a human language is not small, and it is not surprising that the dimension of data vectors grows to be quite large.

The most straightforward reason for adding phonemic context to a phoneme classifier is to attempt to compensate for coarticulation effects. Coarticulation is the effect that adjacent phonemes have on each other due to overlapping in their articulatory positions. For example, in English the /k/ of "keel" is articulated farther forward in the mouth than the /k/ of "call". Coarticulation causes the place of articulation of the consonant to be modified by the following vowel.

To reduce the domain of the context representation from the forty possible phoneme types, yet model coarticulation effects, phonemes are reclassified by place-of-articulation. Place-of-articulation classifications, at least for consonants, are indicative of their primary articulator. Six place-of-articulation categories are used: labial, dental, palatal–alveolar, velar, 'r', and no-consonant. Silences and vowels are grouped in a separate no-consonant category since place-of-articulation for these sounds are not defined in the same way as consonants. The phoneme 'r', being somewhat unique in its effect on surrounding phonemes (by markedly lowering third formant frequency), is given a

category of its own. This category should also be applied to other rhotacizing sounds. The assignment of phonemes in the TIMIT database to place-of-articulation class can be found in Appendix B.

Six indicator functions, one for each of the place categories, are used to define a six element context field. Two context fields, one for the preceding phoneme and one for the following phoneme, are concatenated with the feature field created by a speech preprocessor for a given vowel. The resulting vectors comprise the context dependent training and testing set.

For the proper evaluation of any context method, context independent results must first be obtained for comparison. Feature vectors, as is, are used to train the FSCL/LVQ classifier. Training parameters and training detail are as described in the previous section.

The first and most straightforward method of introducing context is to train FSCL/LVQ with the context dependent data set in the same manner as with the context independent data set. We will refer to this method of adding context as method A.

For the vowel recognition task, it seems reasonable that the importance of context information should be secondary to that of feature information. In the data space, feature information is sufficient to represent the basic organization of vowel clusters. Context merely adds dimensions which refine these clusters. Perhaps some advantage can be gained through the use of training algorithms in which the significance of the feature data is recognized. One way of doing this is to use a training algorithm to learn

the underlying vowel feature space independent of context. With this framework in place, context information can then be added. In this manner, the feature information is used to form clusters before the context information is introduced.

Two implementations of the previously described procedure are proposed for FSCL/LVQ. Both variations, method B and method C, have two training stages: training on data without context information, and retraining on data with context information. The two variations differ in the way that FSCL and LVQ are used in each stage. Method B performs FSCL training on data without context, and uses the resulting codebook as a starting point for LVQ training on data with both feature and context information. Method C performs FSCL and LVQ training on data without context, and uses the resulting codebook as a starting point for more LVQ training on data with both feature and context information present. Vectors without context are created from the vectors with context simply by zeroing out the context fields. Training parameters for the context-dependent methods are identical to those in the context independent case described earlier.

Composite percentage classification results for all methods are given in Table 4. The data in this table is identical to that in Table 3. The table has been rearranged to better facilitate context method comparisons. An improvement of 3 to 5 percent can readily be seen from the addition of context using any method.

To better understand the results, n-best classification statistics are presented for all three feature sets in Figures 7, 8, and 9. The improvements due to context are consistent across context methods and feature representations. The classification

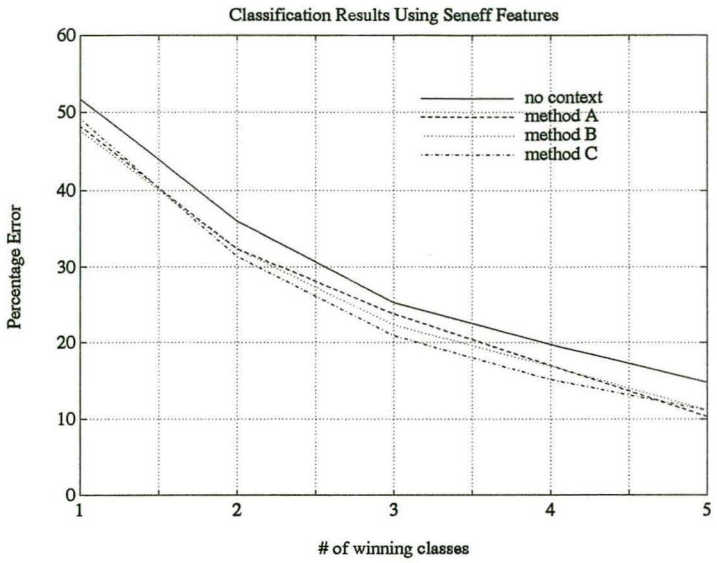


Figure 7: Classification results verses number of winning classes for the Seneff speech representations.

improvements due to context are steady through the best-five results and decline only as classification performance approaches 100%.

WCC performs far worse than the auditory feature sets in the context-independent case and as a result, sees the greatest gains in the use of context. The context-dependent results are fairly consistent among context methods and feature representations. Each feature representation performed best on a different context method. Using the same training parameters and number of iterations, the Seneff model learned the training data sets consistently better than the other representations and performed best on the context-independent tests. These results suggest that the use of the Seneff model representation results in faster convergence than the other representations.

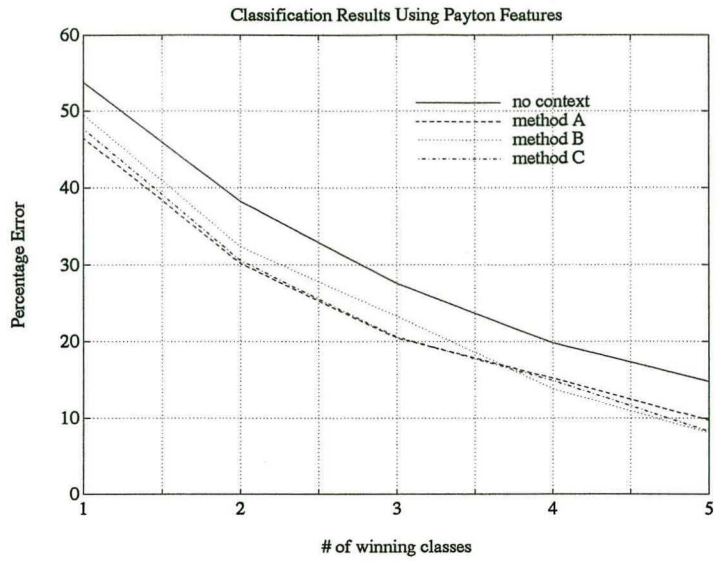


Figure 8: Classification results verses number of winning classes for the Payton speech representations.

## 4.3 Analysis

There are many factors in the above experiments that could have helped or hurt performance. As a result, many questions can be asked about the role of phonemic context, context incorporation methods, and LVQ techniques in the effectiveness of the experiments.

The representation of the phonemic context, binary indicator vectors, may have caused problems in the learning process through its discrete nature. Since the context information is inherently discrete, some other manner of representation, such as a binary index for each place category, or a more ordered representation of place ranging from frontal articulations (bilabials) to back articulations (velars), may better impart the context information. Alternatively, a context representation with a

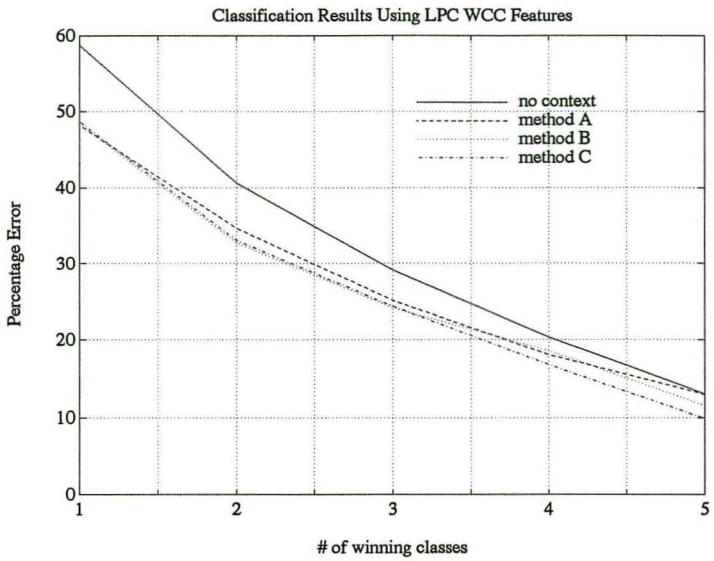


Figure 9: Classification results verses number of winning classes for the LPC WCC speech representations.

more continuous nature may allow more graceful learning with better results.

The simple context-sensitive FSCL-LVQ examples of Chapter III demonstrate that learning with context can be reduced to the learning of the joint distribution of the context and vowel information. In the course of the above experiments, we proposed that the inherent relation between the phonemic context and vowel features could be exploited in such a way that vowel features were emphasized over context. Two of the three methods of introducing context added such emphasis by training on vowel information before context information was added. No improvements or degradations in performance over direct training was observed. Why were there no differences in performance between the three context methods. Is it valid to expect such differences?

Finally, how do the respective LVQ algorithms impact the effectiveness of the learning process. Do the LVQ algorithms adversely react to the addition of context, especially in tests using methods B and C. How do the LVQ algorithms handle the discrete nature of the context representation? LVQ2 was used in the above vowel classification experiments with reasonable results. And although unreported, informal vowel classification tests using other LVQ versions produced comparable yet slightly worse results, we can not be sure that LVQ2 is the best method for the job. Perhaps LVQ3 properties including the retention of rough class distribution properties would serve to better emphasise the vowel clusters and de-emphasis the more random nature of the context data.

The remainder of this section studies in detail the context-dependent vowel recognition experiments described above. The analysis will help to better understand the effects of phonemic context on the vowel classification experiments, and address some questions raised, Due to the large scale experimental procedures used, multiple repetitions of the experiments is prohibitively costly in terms of time and memory. Therefore, the focus of analysis is on the the codebooks generated during this single set of experiments.

### 4.3.1 The Data Set

Important details in the understanding of the experimental results can be found in the data itself. Observations can be made from the usage tables in Appendix A that classify and enumerate the data set in terms of vowel class and context situation. Some primary observations follow:

1. There are more "iy" vowel samples than any other vowel in the set. There are twice as many "iy" vowels as any other vowel.

2. The dental place of articulation is about twice as common as any other place of articulation, in both the preceding and following context case. Labial is second common, with no-consonant in third place.

3. There are many vowel/context combinations that are not represented. The context space is sparse. For a given vowel, less than half of the pre/post context combinations are represented. Among the entire data set, there are no palatal-alveolar/palatal-alveolar or 'r'/'r' combinations and only single tokens for the velar/velar, velar/palatal-alveolar, and no-consonant/palatal-alveolar context combinations.

The fact that all of the possible context and vowel combinations are not represented does not deter from the results obtained. Obviously, the data set is too small for all of the interrelations between vowels and context to be learned. It is reasonable to expect that some of the more prominent vowel and context interrelations will be learned.

Over represented and under represented classes do have the potential to cause improvements unrelated to the coarticulatory context effects we desire to learn. The statistics of the vowel/context combinations in natural speech may outweigh the statistics of the coarticulatory context. In this case, the gains achieved by the incorporation of additional information in the form of context are dependent only on the text spoken. Classification of different utterances may result in poorer results.

### 4.3.2 Codebook study

In this subsection, the codebooks produced for three of the ten speakers are examined with respect to context learning using Method A. In an attempt to better understand how the codewords were distributed throughout the context space, each codeword was classified by hand into a preceding and following context category. Observations on these categorizations are discussed.

The organization of codewords by FSCL is orderly and basically as expected. All of the codewords' context elements were between 0 and 1, usually with no more than two or three non-zero elements. In many cases, the context fields had a single one and the remainder zeros. It was also common to have one of the two non-zero elements predominate over the other. For hand classification of these context fields, the class whose corresponding element was substantially larger than any of the other elements, became the class of the context field. There were a few cases in which the context field seemed to be shared by two or more classes. This is to be expected and even hoped for since it is reasonable to assume that in some context/vowel situations, a speech feature vector could be relatively invariant to contextual place of articulations. The statistics of the classifications matched those of the data set such that the majority of context fields were hand classified dental.

The fine tuning of the codewords by LVQ2 produced context fields that were much harder to hand classify. Elements within the context fields were no longer ranged between 0 and 1 and rarely were there any zero valued elements. Many of the elements were negatively valued. In the majority of the cases, a context field did have

an element whose magnitude dominated over the others, and these elements were used to classify the context field in a way similar to the hand metric for classifying the FSCL context fields. Those element values which dominated over other element values were in the range of 5.0 to 10.0. Some elements got as large as 12.0. This is an alarming observation since the Euclidean error metric is used as the distance measure, and feature vector elements ranged between 0 and 1 in the data set. This observation can be attributed to the modified version of LVQ which was used in this set of experiments. It has been shown that this version has a tendency to cause codewords to diverge and can be unstable. Even with these alarming magnitudes, there was a reasonably strong correlation between the hand classifications of the FSCL codewords and the hand classifications of the LVQ codewords.

### 4.3.3 The Effects of Modifying Codeword Fields

To gain more insight on how context-dependence works and how the data and context fields interact, a series of tests were performed in which different codeword fields were modified (nullified) and the amount of classification degradation observed. The results are somewhat inconclusive since there is a major amount of degradation in all tests. Therefore, only a brief discussion of the procedures and some observations follow.

Four tests are performed on codebooks from two speakers using all three context methods. In each test, classification is performed using codebooks and data in which one or more fields is removed. Classification is performed for previous context and data (pd), data and following context (df), data only (do), and context only (co). The first three n-best results can be seen in Table 5 compared against the unmodified

Table 5: Codebook Study Results: Percentage Classification Error for Speakers JLS and CRH

| Speaker JLS, Method A | | | |
|---|---|---|---|
| Test | n=1 | n=2 | n=3 |
| pd | 98.67 | 68.00 | 54.67 |
| df | 78.67 | 65.33 | 54.67 |
| do | 76.00 | 64.00 | 60.00 |
| co | 78.67 | 72.00 | 62.67 |
| or | 37.33 | 17.33 | 13.33 |

| Speaker CRH, Method A | | | |
|---|---|---|---|
| Test | n=1 | n=2 | n=3 |
| pd | 62.32 | 53.62 | 49.28 |
| df | 63.77 | 47.83 | 39.13 |
| do | 62.32 | 53.62 | 44.93 |
| co | 62.32 | 55.07 | 49.28 |
| or | 43.48 | 36.23 | 30.43 |

| Speaker JLS, Method B | | | |
|---|---|---|---|
| Test | n=1 | n=2 | n=3 |
| pd | 96.00 | 85.33 | 58.67 |
| df | 74.67 | 64.00 | 52.00 |
| do | 89.33 | 70.67 | 57.33 |
| co | 82.67 | 72.00 | 64.00 |
| or | 38.67 | 24.00 | 18.67 |

| Speaker CRH, Method B | | | |
|---|---|---|---|
| Test | n=1 | n=2 | n=3 |
| pd | 78.26 | 50.72 | 46.38 |
| df | 73.91 | 53.62 | 42.03 |
| do | 63.77 | 57.97 | 46.38 |
| co | 76.81 | 53.62 | 52.17 |
| or | 43.48 | 31.88 | 26.09 |

| Speaker JLS, Method C | | | |
|---|---|---|---|
| Test | n=1 | n=2 | n=3 |
| pd | 98.67 | 66.67 | 41.33 |
| df | 68.00 | 42.67 | 25.33 |
| do | 98.67 | 86.67 | 64.00 |
| co | 78.67 | 69.33 | 61.33 |
| or | 52.00 | 30.67 | 10.67 |

| Speaker CRH, Method C | | | |
|---|---|---|---|
| Test | n=1 | n=2 | n=3 |
| pd | 94.20 | 88.41 | 47.83 |
| df | 91.30 | 86.96 | 46.38 |
| do | 91.30 | 91.30 | 78.26 |
| co | 78.26 | 49.28 | 46.38 |
| or | 42.03 | 33.33 | 26.09 |

codebook performance (or). In all cases performance dropped by at least 20%. No one of the four test situations performed significantly better than any others although the pd tests in all but one case got the worst results. Closer study revealed that after removing one or more fields from the codebooks, only a small subset of the entire codebook was ever selected during the classification process. The tests with the worst results typically used only 6 or 8 codewords representing only two or three classes.

It is a little surprising that there was no graceful degradation with respect to the context fields. Expecially so in the cases where only one of the two context fields were zeroed out. The zeroing out of one context field results in changing a single element from 1 to 0. At first glance it is remarkable that the change of a single element by 1 can change performance in such a drastic way. Realize though that the vowel feature field and the context fields are normalized with respect to energy. Therefore, zeroing out any one of the three fields results in reducing the norm of the data vector to 1/3 its original length. Clearly, too much weighting was given to the context data. The optimal weighting of context with respect to the vowel features is an interesting problem that warrants further investigation.

# CHAPTER V

# DISCUSSION

The experiments described in this document serve many purposes. They study vowel classification using the FSCL-LVQ classifier. They introduce the use of place-of-articulation information as a valid form of incorporating contextual information at the phonemic level. They also provide a qualitative measure of the effectiveness of auditory based feature extraction methods and make comparisons to a conventional feature extraction method.

Equally important although less conclusive, this thesis addresses the issue of context-sensitive classification. There seems to be a difference between some types of data which we call context in certain classification problems. In the symbol mapping problem of Kohonen described in section 3.3, the significance of weighting the norm of the context part over the norm of the symbol part during the self-organization process was noted. In the vowel classification experiments of chapter IV, we emphasize different parts of the data and context in an attempt to recognize the intuitive differences in importance of the vowel features and their phonemic context. Two methods were proposed (method B and method C) that gave emphasis to the vowel features early on in the training process. Experimental results using these two methods were compared to straightforward training on the vowel features and context data together

(method A). No differences in performance over straightforward training could be seen from the use of the two alternative context-sensitive methods. This result could be used to claim that context-sensitive methods cannot achieve any better performance than straightforward training, and that there is no difference between context and vowel feature data. We believe that this would be a premature conclusion.

Scaling of different fields of data in the classification process seems to be a logical way to emphasis one field of data over another. Within the domain of competitive learning algorithms and FSCL-LVQ, we realize that scaling should only have an effect on the self-organization process, not on the fine tuning of classification boundaries. Typically self-organizing algorithms seek to minimize some overall error function $E$, typically mean-square. Equation 5.1 shows a typical error function in which $d\mathbf{x}$ is the volume differential in the $\mathbf{x}$ space and $\mathbf{w_c}$ is the "winning" codebook vector according to some distance metric.

$$E = \int \| \mathbf{x} - \mathbf{w_c} \|^2 \, p(\mathbf{x}) d\mathbf{x} \tag{5.1}$$

When a dimension in the $\mathbf{x}$ space is scaled by a scalar $\alpha$, the contribution of that dimension to the overall error measure is scaled by $\alpha^2$. The resulting optimal codebook arrangement that minimizes Equation 5.1 is thereby altered.

Optimal classification boundaries, which every classification algorithm strives to achieve, minimize an error function. Often this error function is a Bayes risk, defined in Equation 5.2 for a simple binary problem.

$$R_B = p_{C1} L_{C1} \alpha + p_{C2} L_{C2} (1 - \beta) \tag{5.2}$$

The symbols $\alpha$ and $(1-\beta)$ represent the false alarm and miss probabilities respectively. $L_{C1}$ and $L_{C2}$ are the loss coefficients associated with incorrectly choosing Class 1 or Class 2 respectively. Bayes risk $R_B$ is typically minimized by a form of likelihood ratio test $\phi(\mathbf{x})$ as given in Equation 5.3 for a binary detection problem.

$$\phi(x) = \begin{cases} 1, & dF_{C1}(x,y)/dF_{C2}(x,y) > p_{C1}L_{C1}/p_{C2}L_{C2} \\ 0, & otherwise \end{cases} \tag{5.3}$$

If the decision boundaries define by this likelihood ratio test $\phi(\mathbf{x})$ are weighted by the same factor as the data is weighted, there will be no net difference in classification.

In the FSCL-LVQ classifier, the placement of codewords by FSCL as determined by the scaling of the data, can have a profound effect on the codebook to which LVQ coverges. In addition, even though scaling should not have any effect upon optimal classification, LVQ with a finite number of codewords is not an optimal classifier and may be affected by scaling when the number of codewords is small.

An effective context-sensitive scaling algorithm for competitive learning networks should take into account the issues mentioned above. A general form for the addition of weighting to typical competitive learning schemes is the weighted Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ as shown in Equation 5.4.

$$d_{\mathbf{W}}(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^{\mathrm{H}}\mathbf{W}^{-1}(\mathbf{x}-\mathbf{y}) \tag{5.4}$$

The familiar squared Euclidean distance measure is a type of squared weighted Euclidean distance in which the weight matrix $\mathbf{W}$ is the identity matrix $\mathbf{I}$.

One major area of future research is in the determination of the optimal weighting matrix $W$ for a given context-sensitive problem. One weight matrix worth investigating is based upon a metric called the Mahalanobis distance. The Mahalanobis

distance uses the inverse covariance matrix $\mathbf{K_x}^{-1}$ of $\mathbf{x}$ as its weight matrix and can be shown to normalize and decorrelate $\mathbf{x}$. A more detailed discussion of the Mahalanobis distance can be found in Appendix C.

Certainly, the simple classification examples of section 3.3 demonstrate that context dependent classification can be viewed simply as the joint distrubution of data and contex. Limited to this view, the issue of context-sensitive classification becomes unremarkable. Effective learning of the joint distribution of two or more fields of data typically requires significantly more training samples than learning of any of the fields of data alone. The amount of data necessary to fully describe the joint distribution of the vowel features and the phonemic context information in the vowel classification problem presented earlier is extremely large. Upfront, we acknowledge the fact that the training data set is insufficient to make this description. That is one of the significant advantages of place-of-articulation as a context feature representation over simple phonemic indicator functions. The dimensionality of the joint data space is significantly reduced. Still, the usage tables of Appendix A and the discussion of section 4.3.1 indicate that more data is necessary. Is there any way to make use of this additional information even though it is not adequate?

An alternative way of defining context in a classification scheme is as any extra information that can be included in the classification task, even if insufficient to define a joint distribution. There should be some measurable correlations between the context data and the definition of classification boundaries. These correlations can be used to improve classification results. We do not know the best way of utilizing

these correlations yet.

One possible technique for better incorporating context within FSCL-LVQ classification ensures that the context-free information is not forgotten. This is done using a learning algorithm that randomly scales the context fields from 0 to 1 during training. The amount of scaling of the context fields can be refered to as the degree of certainty in the context information. Such techniques might prevent the network from over-relying on the context information.

Context can also be introduced gradually into the training process by using a context gain factor that monotonically increaases with iteration count. Such a technique can monitor the performance of the network as the gain factor increases to determine the best amount of scaling for the context fields. Although this technique may be computationally slow, it may reveal insightful information on the effects of scaling in FSCL-LVQ classifiers.

# CHAPTER VI

# SUMMARY AND CONCLUSIONS

## 6.1 Summary

Through empirical investigations, we have demonstrated the application of contextual information into a pattern classification problem. Study has been limited to competitive learning networks, specifically those trained using learning vector quantization (LVQ) methods.

The context-sensitive problem that inspired this work is in speech recognition. The characteristics of a given phoneme is effected by the phonemes surrounding it. Thus, information about the surrounding phonemes should improve phoneme classification. This surrounding information is a prime example of the "context" we refer to throughout this document. Our definition of context encompasses any extra information that can be included in a classification problem.

In any classification or estimation scheme, performance should improve with the application of additional information to the task. It follows that the addition of information in the form of "context" as defined above, should also improve performance results. This fact was demonstrated in two contrived classification problems that use FSCL-LVQ to generate classifiers for data with and without context. These examples

illustrate how context learning is equivalent to learning the joint distribution of the two data sets. In non-trivial classification tasks, some problems may arise as more information (context) is added. With the increase in information and dimensionality, a greater number of training samples is necessary to define the joint distribution of the context and data. Also, different data sets will have their own intrinsic error metrics. The merging of two data sets, although statistically straightforward, may not be trivial for a classification algorithm.

In this paper the addition of time/context information at the initial phoneme classification stage is studied for self-organizing artificial neural networks. Instead of simply concatenating adjacent spectral slices and then learning to distinguish and classify them, we instead provide contextual clues about the preceding and following phonemes. Such a setup significantly reduces the dimensionality of the problem allowing for easier training with little loss in classification accuracy.

A large scale experiment using the FSCL-LVQ classifier was performed to investigate context as applied to phoneme recognition. The task was limited to the recognition of 10 vowels in a talker independent context. Context was incorporated from information about the phonemes surrounding each vowel in the form of broad phonetic categories. The mean accuracy of the vowel classification was low, approximately 50%. The addition of context information gave improvements in performance of 3% to 5% over classification without context. Calculation of n-best statistics found the gains from the use of context information to be consistent.

In the same vowel classification study, a new form for representing phonetic con-

text, place-of-articulation, was offered. Place of articulation data is known to be correlated to the vowel features. This relation is commonly referred to as coarticulation. As a results, this broad phonetic category offers significant linguistic and acoustic information in a compact size.

From the same experiment in which we tested the use of context in vowel classification, results were applied to the evaluation of auditory model based speech feature extractors. The same set of classification experiments was performed using three different speech feature representations: the Seneff Synchrony Model (a model inspired by psycho-acoustic observations), the Payton Model of the Auditory Periphery(a model based on physiological data), and LPC based weighted cepstral coefficients(WCC). Doing so enabled the comparison of the effectiveness of the two auditory based speech feature extractors against a conventional signal processing method.

Classification results using each of the different speech feature extraction methods were comparable. In this particular task, the weighted cepstral coefficients performed slightly worse than the auditory based models. Performance of the Seneff and Payton models were near identical. This perhaps is due to the fact that both auditory models are filterbank based models in contrast to WCCs, which is a parametric method. These results in no way prove that one method is better than another, but instead suggests that auditory based models can be competitive with other signal processing methods in speech feature extraction tasks.

## 6.2 Future Work

The study of adding context into classification tasks has uncovered a number of questions about the integration of different data in a single classification scheme. We know that different data sets can have their own intrinsic error metrics, so what is the best way to incorporate these different data sets in a single classification scheme. There may be a great deal of correlation between the data sets, plus there may be some statistically insignificant information inherently embedded in the data. Although these issues do not affect the classification task in a statistical sense, the may have a profound effect in the way that different learning algorithms adapt.

The determination of the optimal amount of weighting of context information will be studied extensively in the future. Empirically, experiments can be performed in which the weighting is modified and perhaps some heuristic rules can be derived. Two additional methods for incorporating context into the FSCL-LVQ classifier were presented in Chapter V. These methods may yield insights into the more optimal weighting of context. Both techniques use scaling in a dynamic manner to incorporate context without degredating feature information. These two methods will be applied to the same context-sensitive vowel classification experiments described above, although limited to the Seneff feature representation only.

The Mahalanobis distance also presents new opportunities for study within the domain of input scaling. In addition to a straightforward examination of the Mahalanobis distance as an error metric, further study of the Mahalanobis distance may answer some questions about the effects of decorrelation and normalization of input

data in competitive learning algorithms. If there are any gains or significant effects from the normalization and decorrelation of input data, a closer look at the optimal unsupervised learning algorithms of Sanger[9] is warranted. This algorithm is based on a maximum information preservation principle that is closely related to Principle Component Analysis in statistics. As a front end in pattern classification schemes, such an algorithm could be used to transform input data into a form more easily absorbed by pattern classification paradigms.

# Appendix A

# USAGE TABLES FOR VOWEL AND CONTEXT DATA

The following tables count the number of combinations of preceding and following context combinations for a given vowel exist in the data set. In each table, the columns are arranged by following (post) phoneme place-of-articulation and the rows are arranged by preceding (pre) phoneme place-of-articulation. Place-of-articulation labels are abreviated: labial (lb), dental (dt), palatal-alveolar (pt), velar (vl), 'r' (r), and no-consonant (nc). Tables 6 through 8 contain the usage tables for all ten vowels. Table 9 is a usage table for the entire data set.

Table 6: Vowel Usage Tables: "aa", "ae", "ah", and "ay"

| aa | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 2 | 2 | 0 | 5 | 2 | 0 |
| | dt | 1 | 3 | 0 | 0 | 10 | 0 |
| | pa | 1 | 1 | 0 | 0 | 3 | 0 |
| | vl | 0 | 1 | 0 | 0 | 2 | 0 |
| | r | 1 | 3 | 1 | 1 | 0 | 0 |
| | nc | 0 | 2 | 0 | 0 | 4 | 0 |

| ae | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 0 | 2 | 5 | 0 | 0 | 0 |
| | dt | 5 | 21 | 2 | 0 | 0 | 1 |
| | pa | 0 | 4 | 0 | 0 | 0 | 0 |
| | vl | 1 | 0 | 0 | 0 | 0 | 0 |
| | r | 4 | 1 | 11 | 0 | 0 | 0 |
| | nc | 3 | 10 | 4 | 0 | 0 | 0 |

| ah | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 0 | 13 | 0 | 0 | 0 | 0 |
| | dt | 12 | 5 | 2 | 0 | 0 | 0 |
| | pa | 7 | 2 | 0 | 0 | 0 | 0 |
| | vl | 1 | 1 | 1 | 0 | 0 | 0 |
| | r | 0 | 1 | 0 | 0 | 0 | 0 |
| | nc | 3 | 3 | 1 | 0 | 1 | 0 |

| ay | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 3 | 8 | 1 | 1 | 0 | 6 |
| | dt | 6 | 7 | 11 | 0 | 1 | 3 |
| | pa | 0 | 0 | 0 | 0 | 0 | 0 |
| | vl | 0 | 0 | 0 | 0 | 0 | 0 |
| | r | 1 | 1 | 0 | 0 | 0 | 0 |
| | nc | 0 | 1 | 0 | 0 | 1 | 5 |

Table 7: Vowel Usage Tables: "eh", "er", "ey", and "ih"

| eh | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 0 | 16 | 4 | 0 | 3 | 1 |
| | dt | 4 | 10 | 1 | 0 | 0 | 0 |
| | pa | 1 | 3 | 0 | 0 | 7 | 0 |
| | vl | 1 | 5 | 2 | 0 | 0 | 0 |
| | r | 0 | 2 | 0 | 0 | 0 | 0 |
| | nc | 4 | 12 | 0 | 0 | 0 | 0 |

| er | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 3 | 9 | 3 | 1 | 0 | 4 |
| | dt | 1 | 5 | 0 | 0 | 0 | 4 |
| | pa | 0 | 0 | 0 | 0 | 2 | 1 |
| | vl | 0 | 8 | 1 | 0 | 1 | 2 |
| | r | 0 | 0 | 0 | 0 | 0 | 0 |
| | nc | 0 | 7 | 0 | 1 | 0 | 10 |

| ey | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 6 | 8 | 1 | 3 | 0 | 4 |
| | dt | 2 | 15 | 2 | 2 | 0 | 2 |
| | pa | 1 | 0 | 1 | 1 | 0 | 0 |
| | vl | 3 | 0 | 0 | 0 | 0 | 0 |
| | r | 0 | 1 | 1 | 1 | 0 | 0 |
| | nc | 1 | 1 | 0 | 0 | 0 | 0 |

| ih | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 1 | 7 | 2 | 1 | 1 | 1 |
| | dt | 7 | 20 | 1 | 2 | 0 | 2 |
| | pa | 0 | 0 | 0 | 0 | 0 | 0 |
| | vl | 1 | 2 | 0 | 0 | 0 | 7 |
| | r | 1 | 6 | 0 | 0 | 0 | 1 |
| | nc | 1 | 5 | 0 | 0 | 0 | 1 |

Table 8: Vowel Usage Tables: "iy", and "ow"

| iy | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 2 | 17 | 3 | 1 | 0 | 3 |
| | dt | 21 | 20 | 7 | 1 | 7 | 21 |
| | pa | 0 | 1 | 0 | 0 | 0 | 0 |
| | vl | 0 | 0 | 0 | 0 | 0 | 14 |
| | r | 2 | 14 | 0 | 2 | 0 | 9 |
| | nc | 4 | 8 | 1 | 0 | 1 | 4 |

| ow | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 0 | 3 | 1 | 0 | 0 | 1 |
| | dt | 6 | 16 | 0 | 2 | 0 | 2 |
| | pa | 0 | 1 | 0 | 0 | 0 | 3 |
| | vl | 1 | 0 | 1 | 0 | 0 | 0 |
| | r | 1 | 3 | 1 | 0 | 0 | 1 |
| | nc | 4 | 0 | 0 | 0 | 0 | 0 |

Table 9: Vowel Usage Tables: Totals

| Totals | | | | | | | |
|---|---|---|---|---|---|---|---|
| vowel context | | post | | | | | |
| | | lb | dt | pa | vl | r | nc |
| pre | lb | 17 | 85 | 20 | 12 | 6 | 20 |
| | dt | 65 | 122 | 26 | 7 | 18 | 35 |
| | pa | 10 | 12 | 1 | 1 | 12 | 4 |
| | vl | 8 | 17 | 5 | 0 | 3 | 23 |
| | r | 10 | 32 | 14 | 4 | 0 | 11 |
| | nc | 20 | 49 | 6 | 1 | 7 | 20 |

# Appendix B

# PHONEME TO PLACE-OF-ARTICULATION MAP

Tables 10 and 11 list all of the phonemes in the TIMIT database in IPA symbol form and in TIMIT symbol form respectively. Phonemes are arranged by the place-of-articulation context class in which they have been assigned. Some types of phonemes, such as glides, are difficult to classify in terms of place. As a results, there are many valid ways to segment the phoneme corpus, and Table 10 is just one possible selection.

Table 10: Mapping of Phoneme to Place-of-articulation using IPA Symbols

| Labial | Dental | | Velar | Palatal-Alveolar | 'r' | No-consonant | |
|---|---|---|---|---|---|---|---|
| p | t | $\theta$ | k | $\check{c}$ | r | h | ɦ |
| $p^{o}$ | $t^{o}$ | l | $k^{o}$ | y | | $\varepsilon$ | ɪ |
| m | ɾ | d | g | <sil> | | ɔ | æ |
| f | n | $d^{o}$ | ŋ | ʃ | | $\alpha$ | ʌ |
| b | $\tilde{r}$ | n̥ | $g^{o}$ | ǯ | | u | ʊ |
| $b^{o}$ | s | ð | ŋ̊ | ǰ | | ɛ̞ | ü |
| m̥ | z | l̥ | | <sil> | | $a^{y}$ | $o^{y}$ |
| v | | | | | | $e^{y}$ | $i^{y}$ |
| w | | | | | | $a^{w}$ | $o^{w}$ |
| | | | | | | ə | ɚ |
| | | | | | | ɨ | ɚ̥ |
| | | | | | | ʔ | <sil> |

Table 11: Mapping of Phoneme to Place-of-articulation using TIMIT symbols

| Labial | Dental | | Velar | Palatal-Alveolar | 'r' | No-consonant | |
|---|---|---|---|---|---|---|---|
| p | t | th | k | ch | r | hh | hv |
| pcl | tcl | l | kcl | y | | eh | ih |
| m | dx | d | g | pau | | ao | ae |
| f | n | dcl | ng | sh | | aa | ah |
| b | nx | en | gcl | zh | | uw | uh |
| bcl | s | dh | eng | jh | | er | ux |
| em | z | el | | epi | | ay | oy |
| v | | | | | | ey | iy |
| w | | | | | | aw | ow |
| | | | | | | ax | axr |
| | | | | | | ix | ax-h |
| | | | | | | q | h# |

Table 11: Mapping of Phoneme to Place-of-articulation using TIMIT symbols

| Labial | Dental | | Velar | Palatal-Alveolar | 'r' | No-consonant | |
|---|---|---|---|---|---|---|---|
| p | t | th | k | ch | r | hh | hv |
| pcl | tcl | l | kcl | y | | eh | ih |
| m | dx | d | g | pau | | ao | ae |
| f | n | dcl | ng | sh | | aa | ah |
| b | nx | en | gcl | zh | | uw | uh |
| bcl | s | dh | eng | jh | | er | ux |
| em | z | el | | epi | | ay | oy |
| v | | | | | | ey | iy |
| w | | | | | | aw | ow |
| | | | | | | ax | axr |
| | | | | | | ix | ax-h |
| | | | | | | q | h# |

# Appendix C

# MAHALANOBIS DISTANCE

The squared weighted Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ can be written in the quadratic form of equation C.1.

$$d_{\mathbf{W}}(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^{\mathrm{H}}\mathbf{W}^{-1}(\mathbf{x}-\mathbf{y}) \tag{C.1}$$

The familiar squared Euclidean distance measure is a type of squared weighted Euclidean distance in which the weight matrix $\mathbf{W}$ is the identity matrix $\mathbf{I}$.

By limiting the weight matrix $\mathbf{W}$ to the class of positive definite matrices, some interesting properties can be discovered. If $\mathbf{W}$ is positive definite, its inverse is also positive definite and can be factored as shown in equation C.2 where $\mathbf{T}$ is full rank. If $\mathbf{W}$ is positive semidefinite, it also can be factors as in equation C.2 althought $T$ will not be full rank or its principle minors will be only nonnegative.

$$\mathbf{W}^{-1} = \mathbf{T}^{H}\mathbf{T} \tag{C.2}$$

This decomposition can be used to show that the squared weighted Euclidean distance $d_W(\mathbf{x},\mathbf{y})$ is equivalent to the squared Euclidean distance $d_E(\mathbf{x},\mathbf{y})$ of the input vectors transformed by the matrix $\mathbf{T}$.

$$d_W(\mathbf{x},\mathbf{y}) \;=\; (\mathbf{x}-\mathbf{y})^{\mathrm{H}}\mathbf{W}^{-1}(\mathbf{x}-\mathbf{y}) \tag{C.3}$$

$$d_W(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^{\mathrm{H}}\mathbf{T}^{\mathrm{H}}\mathbf{T}(\mathbf{x}-\mathbf{y}) \quad (C.4)$$

$$d_W(\mathbf{x},\mathbf{y}) = (\mathbf{T}(\mathbf{x}-\mathbf{y}))^{\mathrm{H}}(\mathbf{T}(\mathbf{x}-\mathbf{y})) \quad (C.5)$$

$$d_W(\mathbf{x},\mathbf{y}) = (\mathbf{T}\mathbf{x}-\mathbf{T}\mathbf{y})^{\mathrm{H}}(\mathbf{T}\mathbf{x}-\mathbf{T}\mathbf{y}) \quad (C.6)$$

$$d_W(\mathbf{x},\mathbf{y}) = d_E(\mathbf{T}\mathbf{x},\mathbf{T}\mathbf{y}) \quad (C.7)$$

If the weight matrix is chosen to be the correlation matrix $\mathbf{R_x}$ of a random vector $x$, then the matrix transform $\mathbf{T}$ can be shown to decorrelate and normalize the input vectors.

$$\mathbf{R_x} = E[\mathbf{d}\mathbf{d}^{\mathrm{H}}] \quad (C.8)$$

$$(\mathbf{T}^{\mathrm{H}}\mathbf{T})^{-1} = E[\mathbf{d}\mathbf{d}^{\mathrm{H}}] \quad (C.9)$$

$$\mathbf{T}^{-1}\mathbf{T}^{-\mathrm{T}} = E[\mathbf{d}\mathbf{d}^{\mathrm{H}}] \quad (C.10)$$

$$\mathbf{I} = \mathbf{T}E[\mathbf{d}\mathbf{d}^{\mathrm{H}}]\mathbf{T}^{\mathrm{H}} \quad (C.11)$$

$$\mathbf{I} = E[\mathbf{T}\mathbf{d}\mathbf{d}^{\mathrm{H}}\mathbf{T}^{\mathrm{H}}] \quad (C.12)$$

$$\mathbf{I} = E[(\mathbf{T}\mathbf{d})(\mathbf{T}\mathbf{d})^{\mathrm{H}}] \quad (C.13)$$

$$\mathbf{I} = E[\mathbf{b}\mathbf{b}^{\mathrm{H}}], \qquad \mathbf{b} = \mathbf{T}\mathbf{d} \quad (C.14)$$

Similarly, if the weight matrix is chosen to be the covariance matrix $\mathbf{K_x}$ of a random vector $x$, then the matrix transform can be shown to decorrelate and normalize input vectors minus their statistical mean $\mathbf{m_x}$. A squared weighted Euclidean distance that uses the covariance matrix of the data as the weight matrix is refered to as a Mahalanobis distance.

$$\mathbf{K_x} = E[(\mathbf{d}-\mathbf{m})(\mathbf{d}-\mathbf{m})^{\mathrm{H}}] \quad (C.15)$$

$$(\mathbf{T}^{\mathrm{H}}\mathbf{T})^{-1} = E[(\mathbf{d}-\mathbf{m})(\mathbf{d}-\mathbf{m})^{\mathrm{H}}] \quad \text{(C.16)}$$

$$\mathbf{T}^{-1}\mathbf{T}^{-\mathrm{T}} = E[(\mathbf{d}-\mathbf{m})(\mathbf{d}-\mathbf{m})^{\mathrm{H}}] \quad \text{(C.17)}$$

$$\mathbf{I} = \mathbf{T}E[(\mathbf{d}-\mathbf{m})(\mathbf{d}-\mathbf{m})^{\mathrm{H}}]\mathbf{T}^{\mathrm{H}} \quad \text{(C.18)}$$

$$\mathbf{I} = E[\mathbf{T}(\mathbf{d}-\mathbf{m})(\mathbf{d}-\mathbf{m})^{\mathrm{H}}\mathbf{T}^{\mathrm{H}}] \quad \text{(C.19)}$$

$$\mathbf{I} = E[\mathbf{T}(\mathbf{d}-\mathbf{m})(\mathbf{T}(\mathbf{d}-\mathbf{m}))^{\mathrm{H}}] \quad \text{(C.20)}$$

$$\mathbf{I} = E[\mathbf{b}\mathbf{b}^{\mathrm{H}}], \qquad \mathbf{b} = \mathbf{T}(\mathbf{d}-\mathbf{m}) \quad \text{(C.21)}$$

This works for any $\mathbf{R_x}$ and any $\mathbf{K_x}$ since they are always positive semidefinite, and are positive definite if they are full rank. Frequently, the covariance matrix is not known and must be estimated from the data using estimators such as the ones given in equations C.22 and C.23.

$$\mathbf{K_x} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i}-\mathbf{m_i})(\mathbf{x_i}-\mathbf{m_i})^{\mathrm{H}} \quad \text{(C.22)}$$

$$\mathbf{R_x} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i}\mathbf{x_i}^{\mathrm{H}} \quad \text{(C.23)}$$

These estimates cannot be guaranteed to be positive definite or positive semidefinite. In these cases, the properties of the weighted distance cannot be defined.

There are many ways of decomposing a matrix in this manner. The resulting weighted distance measure is unaffected, although it adds a twist to the possible interpretations of the effects of the squared weighted Euclidean distance.

The above analysis yields a better understanding of squared weighted Euclidean distances and more specifically the Mahalanobis distance. The use of this distance metric in competative learning schemes such as FSCL or LVQ has not yet been studied.

# REFERENCES


[1] S. Seneff and V. W. Zue, "Transcription and alignment of the timit database," in John S. Garofolo, editor, *Getting Started with the DARPA TIMIT CD-ROM: An Acoustic Phonetic Continuous Speech Database*, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 1988.

[2] H. Leung and V. W. Zue, "Some phonetic recognition experiments using artificial neural nets," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, (New York, NY), pp. 422–425, 1988.

[3] S. Seneff, "A Joint Synchrony/Mean–Rate Model of Auditory Speech Processing," *Journal of Phonetics*, vol. 16, pp. 55–76, 1988.

[4] K. Payton, "Vowel processing by a model of the auditory periphery: A comparison to eighth-nerve responses," *Journal of the Acoustical Society of America*, vol. 83, pp. 145–162, January 1988.

[5] B. Juang, L. R. Rabiner, and J. G. Wilpon, "On the use of bandpass filtering in speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 947–954, July 1987.

[6] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, vol. 4, pp. 4–22, April 1987.

[7] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton, "Competitive Learning Algorithms for Vector Quantization," *Neural Networks*, vol. 3, pp. 277–290, 1990.

[8] T. Kohonen, "The Self-Organizing Map," *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, September 1990.

[9] T. D. Sanger, "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network," tech. rep., MIT AI lab.

[10] C. Moler, J. Little, S. Bangert, and S. Kleiman, "PRO-MATLAB User's Guide," The MathWorks Inc., Sherborn, MA, 1987.