Automatic Acoustic Measurement Optimization for Segmental Speech Recognition

by

Manish D. Muzumdar

Submitted to
the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering

at the

Massachusetts Institute of Technology

May, 1996

Author .....
Department of Electrical Engineering and Computer Science
May 28, 1996

Certified by .....
I. Lee Hetherington
Research Scientist
ering and Computer Science

Accepted by .....
Frederic R. Morgenthaler
Chair, Department Committee on Graduate Students

Automatic Acoustic Measurement Optimization for Segmental Speech Recognition

by

Manish D. Muzumdar

Submitted to
the Department of Electrical Engineering and Computer Science on May 28, 1996
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering

Abstract

Modern speech recognizers often train their acoustic models on fairly simple measurements, such as cepstral coefficients and their time differences. Some researchers, however, favor more complex sets of acoustic measurements that enable them to incorporate speech knowledge and intuition.

The two objectives of this thesis are to develop a framework that allows speech scientists to incorporate complex measurements, and to demonstrate the usefulness of the framework by creating acoustic measurement sets that improve phonetic classification performance.

This thesis reveals the capability of the two-stage framework: the means to optimize complex measurements for maximum discrimination between pairs of confusable phonetic classes, and the means to conduct a performance-based, best-first search through a collection of measurements.

The utility of the framework is evident in its ability to reduce the dimensionality of a baseline set of cepstral measurements. This thesis includes several low-dimensional measurement sets that beat baseline performance.

Thesis Supervisor: I. Lee Hetherington
Title: Research Scientist, Laboratory for Computer Science

Acknowledgements

I would like to thank Lee Hetherington, my thesis supervisor, for his guidance. I am honored to be his very first Master's student. Lee was always available to discuss all matters, by appointment or unannounced, and his insight on speech and on technical implementation were critical to the completion of this thesis. It was a pleasure to work with Lee over the past year.

Victor Zue has my gratitude for the opportunity to research in a group at the forefront of speech technology. I am happy that Victor acted as my co-advisor, reading this thesis and offering sage advice from his vast experience.

In addition, I am grateful to Jim Glass and Ray Chun for imparting their knowledge of phonetic classification, and I would like to thank Lee, Mike McCandless, Jane Chang, and Jim, the developers of SAPPHIRE, the wonderful speech recognition toolkit on which SAILS is built. Finally, thanks to the members of Spoken Language Systems, who made for a friendly and intellectually stimulating environment, not to mention a great softball team!

This thesis is dedicated to my parents, grandparents, and brothers, whose love and encouragement I will always treasure.

This research was supported by a contract from NYNEX Science and Technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today, most speech recognition systems depend on automatic learning techniques. Often, these recognizers train their acoustic models on fairly simple measurements, such as cepstral coefficients and their time differences. Some researchers, however, favor more complex sets of acoustic measurements that enable them to incorporate speech knowledge and intuition [9].

1.1 Complex Sets of Acoustic Measurements

When used in the right way, complex measurements allow classifiers to utilize training data more effectively than simple measurements. Phillips and Zue argue that complex sets of measurements can improve classification by making training algorithms concentrate on aspects of the speech signal important for phonetic contrasts. Focusing on the right features can lead to a reduction in acoustic model complexity, resulting in better parameter estimates for a given amount of training data [9].

Given that complex sets of acoustic measurements are a good idea, how do we best harness their potential? Phillips and Zue state that the modeling process and the choice of incorporated speech knowledge determine the utility of complex measurements. Specifically, they recommend segment-based acoustic models and warn against measurements that rely entirely on human speech knowledge (i.e., do not use any machine training whatsoever).

In a segment-based system, we can use acoustic features in a speech signal to determine the phonetic units that compose the signal. Complex measurements can find these features, including attributes that vary over time, because a series of frames is available from which to extract the measurements. In fact, one can argue that the desire to use complex measurements motivates using a segmental approach in the first place. For example, we can create a measurement to perform a regression on energy over an entire segment. In a frame-based system, however, it is not clear how to extract such a dynamic measurement because there is no notion of a segment over which to perform the regression.

Even in a segment-based system, choosing what speech knowledge to incorporate in measurements is critical. First, we must attack the right attributes, since some speech features are difficult to measure even though we can qualitatively describe them (e.g., tracking formant trajectories [5] and extracting distinctive features [7]). Second, we should not define complex measurements based solely on speech knowledge. Since acoustic-phonetic knowledge is not complete enough to *fully* specify a set of detailed measurements [9], we should use an element of machine learning—in addition to speech theory—to assist us in defining measurements.

1.2 Generalized Measurement Algorithms

Phillips and Zue describe an approach using generalized algorithms that combines speech knowledge and machine training to define complex measurements.

A generalized algorithm can measure acoustic features known to be important for phonetic discrimination, such as the spectral center of gravity or average spectral amplitude. As illustrated in Figure 1-1, each of these generalized algorithms has free parameters which, when set, define a measurement. Example parameters include which portion of the segment in time and in frequency to extract the measurement from.

Therefore, as Phillips and Zue note, a small library of generalized algorithms with free parameters can produce a large space of possible acoustic measurements. By

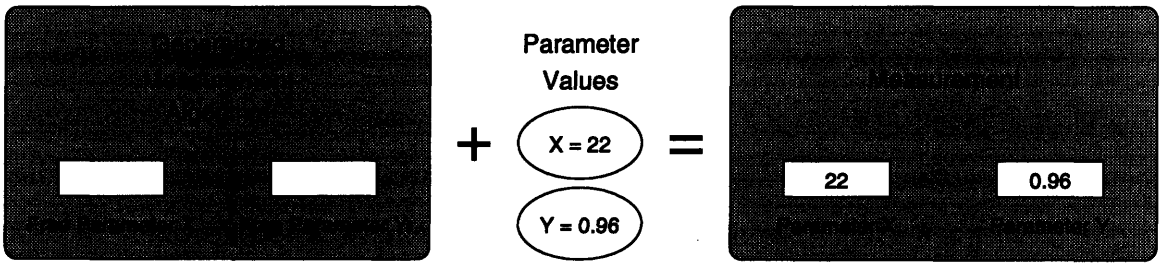


Figure 1-1: Measurement Definition

choosing different sets of generalized algorithms and varying their free parameters, we can determine a subset of measurements that maximizes phonetic discrimination performance on a large body of training data.

1.3 Measurement Optimization

The ultimate objective is to discover the measurement set that results in the highest recognition performance. Given the large space of possible measurements, using a brute-force approach to find the optimal measurements is not practical.

Instead, Phillips and Zue suggest a two-stage search. First, we create a pool of measurements that best discriminate between pairs of confusable classes. Then, we choose measurements from this pool that maximize classification accuracy over all classes.

In the first stage, we create a collection of optimized measurements that discriminate between various pairs of confusable phonetic classes. To create an optimized measurement, we select a generalized algorithm that might prove to be an effective discriminator and vary its parameters. For a given parameter set, we compute a measurement vector for each phone (in either class) and use these vectors to calculate Fisher's linear discriminant [3], a measure of maximum linear separation between the two phonetic classes. The parameter set that generates the highest discrimination score defines the optimized measurement. Fisher's linear discriminant is quickly computed from measurement vectors, thus serving as a good optimization criterion. With the optimization process just described, we can create a few optimized measurements for each of the most confusable classes.

In the second stage, we use the first-stage measurement pool in a best-first search, conducting classification experiments with all phones to generate the N-best measurements that maximize classification accuracy. The optimal measurement set begins empty or with a few baseline measurements not in the first-stage collection. On each iteration, every measurement left in the measurement pool is individually appended to the optimal set, creating one trial set for each measurement. We use each of these trial measurement sets, in turn, to train and test a mixture-diagonal Gaussian classifier, and the trial set that generates the highest classification accuracy becomes the next optimal set. Thus, we add the best measurement to the optimal set on each iteration, yielding the N-best first-stage measurements after N iterations. Figure 1-2 depicts this process with four measurements, represented as solid circles, members of the optimal set in black. The ovals represent trial sets, with their classification accuracies below. The high score for each round appears in bold.

A few years ago, Phillips developed SAILS, a Lisp-machine implementation of the Phillips-Zue approach. This thesis will describe a SAILS implementation for Unix machines, with enhanced capability for user interaction.

1.4 Thesis Overview

The goal of this thesis is twofold:

- to develop a framework that contains generalized measurement algorithms and allows measurement optimization, and
- to demonstrate the usefulness of the framework by creating measurements that improve the performance of a segmental speech recognition system.

In Chapter 2, we describe SAILS, a tool based on the Phillips-Zue approach that semi-automatically discovers acoustic measurement sets for segmental recognition. The SAILS framework includes generalized measurement algorithms, which we examine in detail. We demonstrate the SAILS graphical interface, which allows us to

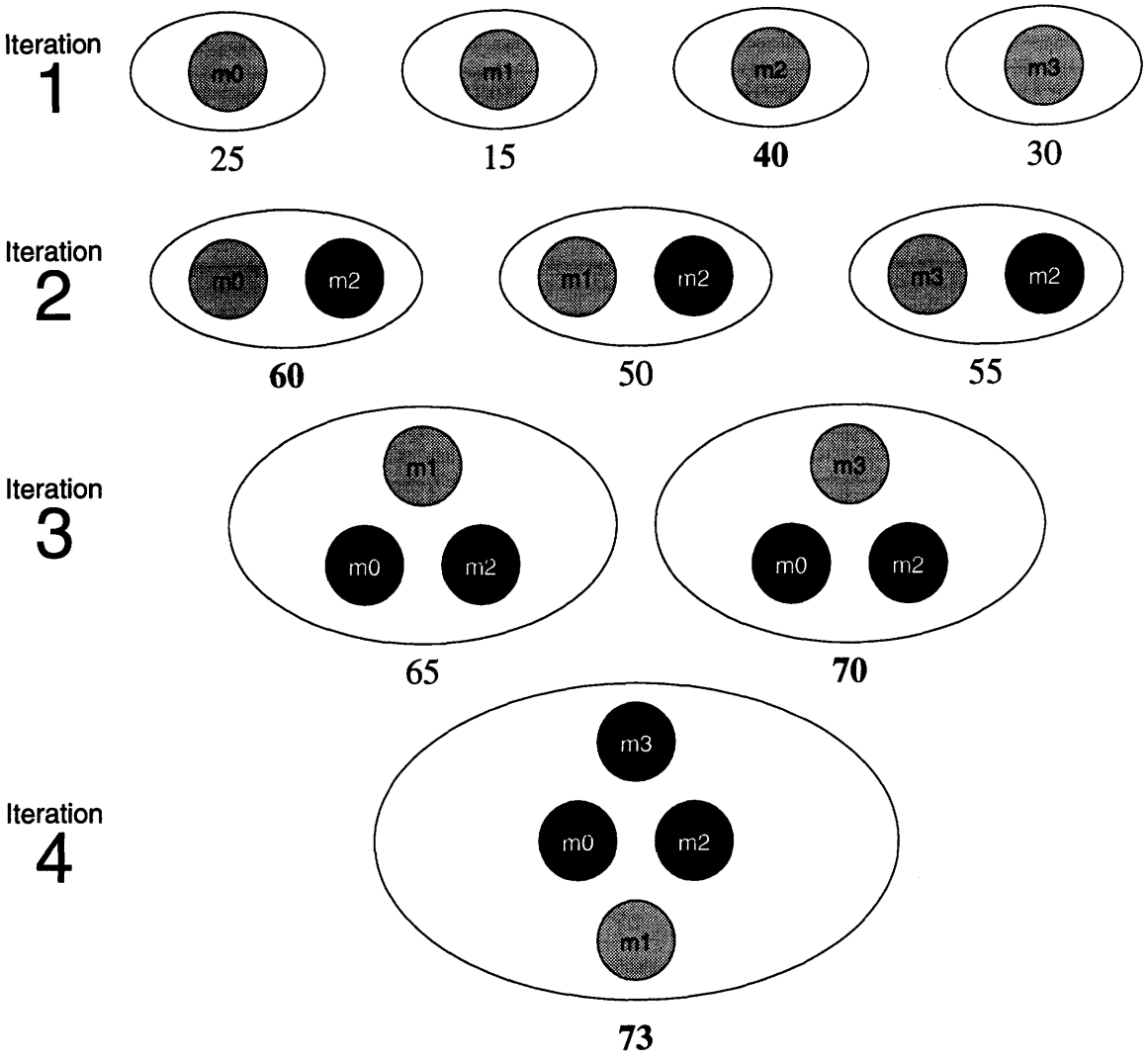


Figure 1-2: Second-stage Search

conduct a first-stage search; we explain how to specify experiments and how to evaluate the discriminative power of measurements through scatter plots and optimization graphs. We also discuss Fisher's linear discriminant, the first-stage optimization criterion. Finally, we reveal the capability of the second-stage search, using it to dissect a generic set of cepstral coefficient measurements.

In Chapter 3, we conduct experiments with SAILS to discover acoustic measurement sets that improve phonetic classification performance because we feel that improvements in classification will translate into improvements in recognition. We discuss our cepstral baseline experiments and our two-stage search through complex measurements, measurements targeted at major phonetic confusions of the baseline system. Next, we examine techniques to reduce the baseline measurement set using the second stage; we perform a computationally efficient, generic measurement search to incrementally build sets of cepstral measurements. Finally, we evaluate these small sets, discovering that we can beat baseline performance with fewer measurements.

In Chapter 4, we draw conclusions from our work and suggest further research that can be conducted with SAILS.

Chapter 2

SAILS

Based on the Phillips-Zue approach, SAILS is a tool that semi-automatically discovers acoustic measurement sets for segmental recognition. In this chapter, we examine the SAILS framework, which contains a library of generalized measurement algorithms and handles measurement optimization in two stages.

2.1 Generalized Measurement Algorithms

SAILS includes generalized algorithms designed to extract measurements from Mel-frequency spectral coefficient (MFSC) and Mel-frequency cepstral coefficient (MFCC) signal representations (see Section 3.1.2). Under the framework, extending the library with additional algorithms is simple. The subsequent sections describe the measurement algorithms currently implemented, algorithms that extract spectral centers of gravity, spectral peaks, and spectral energy.

When discussing the mathematics of each generalized algorithm, we use some common notation. First, $s(n, c)$ is the Mel-based spectral energy for a particular utterance, where n is the frame number and c is the coefficient number. Second, since these algorithms have free parameters for time and frequency intervals, n_i and n_f denote time bounds (in frames), while c_i and c_f denote frequency bounds (in MFSC's). Finally, N is the number of frames in a specified time interval (i.e., $N = n_f - n_i + 1$).

2.1.1 Average Spectral Center of Gravity

The average spectral center of gravity algorithm, **avg_cg**, determines the frequency of densest spectral energy within specified time and frequency intervals. We can use **avg_cg** to extract frequency locations of spectral prominences within a segment, giving us a rough estimate of formant frequencies. This algorithm is most effective in segments with strong formants, namely vowels. Therefore, an **avg_cg** measurement is useful to discriminate between various vowel pairs.

To compute the average center of gravity, we first sum the spectral energy at each MFSC across the specified time interval:

$$S(c) = \sum_{n=n_i}^{n_f} s(n, c) \tag{2.1}$$

Then, the spectral first moment of "superframe" $S(c)$ serves as the result:

$$\overline{cg} = \frac{1}{\sum_{c=c_i}^{c_f} S(c)} \cdot \sum_{c=c_i}^{c_f} c \cdot S(c) \tag{2.2}$$

As an option, **avg_cg** also returns the average amplitude of spectral energy at MFSC $\overline{cg}$. Since $\overline{cg}$ is not discrete, **avg_cg** determines the amplitude (in dB) through linear interpolation:

$$A_{\overline{cg}} = 10 \cdot \log\left(\frac{S(\lfloor \overline{cg} \rfloor) + (\overline{cg} - \lfloor \overline{cg} \rfloor) \cdot (S(\lceil \overline{cg} \rceil) - S(\lfloor \overline{cg} \rfloor))}{N}\right) \tag{2.3}$$

Amplitude can qualify average center of gravity, when such a measurement is used to train across all phones. For example, $\overline{cg}$ might be rather random when dealing with stop closures, but hopefully, training algorithms will give less credence to $\overline{cg}$ when faced with the low amplitude. Conversely, the high spectral amplitude of vowels should mark $\overline{cg}$ as a valuable discriminator.

2.1.2 Slope of Spectral Centers of Gravity

The slope of spectral centers of gravity algorithm, **slope_cg**, finds the slope of the line that best fits the spectral center of gravity of each frame within specified time and frequency intervals. We can use **slope_cg** to extract the dynamics of spectral prominences within a segment, giving us a rough estimate of how sharply formants rise or fall. As with average center of gravity, this algorithm is most effective in segments with strong formants, like vowels. Therefore, a **slope_cg** measurement is useful to discriminate between vowel pairs with differing formant dynamics, such as [iʸ] and [ɪ].

To compute the slope of centers of gravity, we determine the spectral first moment in each frame:

$$cg(n) = \frac{1}{\sum_{c=c_i}^{c_f} s(n,c)} \cdot \sum_{c=c_i}^{c_f} c \cdot s(n,c) \tag{2.4}$$

Then, we use linear regression techniques [10] to find the slope:

$$\Delta cg = \frac{N \cdot S_{xy} - S_x \cdot S_y}{N \cdot S_{xx} - {S_x}^2} \tag{2.5}$$

$$\text{where } S_x = \sum_{n=n_i}^{n_f} n,\ S_y = \sum_{n=n_i}^{n_f} cg(n),\ S_{xx} = \sum_{n=n_i}^{n_f} n^2, \text{ and } S_{xy} = \sum_{n=n_i}^{n_f} n \cdot cg(n)$$

2.1.3 Average Spectral Peak

The average spectral peak algorithm, **avg_peak**, averages the frequencies corresponding to the spectral peak in each frame within specified time and frequency intervals. We can use **avg_peak** to estimate formant frequencies, much like average center of gravity.

Whether one should use **avg_cg** or **avg_peak** depends on the situation. Usually, a center of gravity measurement is more robust; since **avg_cg** is weighted toward frequencies dense with spectral energy, it is more likely to ignore a stray spectral spike created by noise. On the other hand, if segments contain multiple regions of dense energy within the specified frequency bounds, a spectral peak measurement might be best. For example, consider a segment with two energy packets, the stronger

near the upper frequency boundary, the weaker near the lower boundary. **Avg_cg** will return a frequency between the boundaries, while **avg_peak** will return the frequency of the stronger packet, which we hope is more phonetically relevant.

To compute the average spectral peak, we first determine the spectral peak in each frame:

$$p(n) = \max\left(s(n, c_i), s(n, c_{i+1}), \ldots, s(n, c_f)\right) \tag{2.6}$$

Then, the average of $p(n)$ is the result:

$$\overline{p} = \frac{1}{N} \cdot \sum_{n=n_i}^{n_f} p(n) \tag{2.7}$$

Optionally, **avg_peak** also returns the average amplitude of spectral energy at MFSC $\overline{p}$. Since $\overline{p}$ is not discrete, **avg_peak** finds the amplitude (in dB) through linear interpolation of $S(c)$, a "superframe" of spectral energy (see Equation 2.1):

$$A_{\overline{p}} = 10 \cdot \log\left(\frac{S(\lfloor\overline{p}\rfloor) + (\overline{p} - \lfloor\overline{p}\rfloor) \cdot (S(\lceil\overline{p}\rceil) - S(\lfloor\overline{p}\rfloor))}{N}\right) \tag{2.8}$$

Amplitude can qualify average spectral peak much like it can add credibility to an **avg_cg** measurement. The section on average spectral center of gravity contains a relevant discussion.

2.1.4 Slope of Spectral Peaks

The slope of spectral peaks algorithm, **slope_peak**, finds the slope of the line that best fits the spectral peak in each frame within specified time and frequency intervals. We can use **slope_peak** to estimate formant dynamics, much like slope of centers of gravity. As in the previous section, there is a trade-off between **slope_cg** and **slope_peak**. Phillips and Zue used an algorithm similar to slope of spectral peaks to discriminate between [m] and [n] [9].

To compute the slope of spectral peaks, we perform a linear regression [10] on

$p(n)$, the frame-by-frame spectral peak function in Equation 2.6:

$$\Delta p = \frac{N \cdot S_{xy} - S_x \cdot S_y}{N \cdot S_{xx} - {S_x}^2} \tag{2.9}$$

$$\text{where } S_x = \sum_{n=n_i}^{n_f} n,\ S_y = \sum_{n=n_i}^{n_f} p(n),\ S_{xx} = \sum_{n=n_i}^{n_f} n^2, \text{ and } S_{xy} = \sum_{n=n_i}^{n_f} n \cdot p(n)$$

2.1.5 Average Spectral Energy

The average spectral energy algorithm, **avg_energy**, determines the average energy per frame (in dB) within specified time and frequency intervals:

$$\overline{E} = 10 \cdot \log\left(\frac{1}{N} \cdot \sum_{n=n_i}^{n_f} \sum_{c=c_i}^{c_f} s(n,c)\right) \tag{2.10}$$

2.1.6 Ratio of Spectral Energy

The ratio of spectral energy algorithm, **ratio_energy**, finds the energy ratio between two pairs of frequency intervals within specified time bounds. This algorithm is most effective in discriminating between segments which have similar energy concentrations in one frequency band and different energy concentrations in another. Therefore, a **ratio_energy** measurement is useful to discriminate between voiced and unvoiced phones with similar high-frequency energy characteristics. Prime candidates include the strong fricatives [s] and [z], the latter of which sometimes differs in its low-frequency energy due to the presence of voicing.

The result of this algorithm is the ratio (in dB) of the total energy in each frequency interval:

$$E_{ratio} = 10 \cdot \log\left(\frac{\sum_{n=n_i}^{n_f} \sum_{c=c_{i,0}}^{c_{f,0}} s(n,c)}{\sum_{n=n_i}^{n_f} \sum_{c=c_{i,1}}^{c_{f,1}} s(n,c)}\right) \tag{2.11}$$

2.1.7 Other Measurement Algorithms

SAILS includes fairly standard measurement algorithms as well. The avg_vector algorithm allows us to take cepstral coefficient averages over time, while the derivative algorithm lets us incorporate cepstral time derivatives. The equations follow, with $k(n, c)$ as the Mel-frequency cepstral coefficients for a particular utterance:

$$\overline{V}(c) = \frac{1}{N} \cdot \sum_{c=c_i}^{c_f} k(n, c) \tag{2.12}$$

$$\Delta V(c) = \frac{N \cdot S_{xy} - S_x \cdot S_y}{N \cdot S_{xx} - {S_x}^2} \tag{2.13}$$

$$\text{where } S_x = \sum_{n=n_i}^{n_f} n,\ S_y = \sum_{n=n_i}^{n_f} k(n, c),\ S_{xx} = \sum_{n=n_i}^{n_f} n^2, \text{ and } S_{xy} = \sum_{n=n_i}^{n_f} n \cdot k(n, c)$$

Finally, the duration algorithm returns the log of the number of frames in a segment.

2.1.8 Measurement Notation

When referring to a measurement, we list the generalized algorithm and its parameter values, as follows:

- avg_cg start-time end-time start-coeff end-coeff
- avg_vector start-time end-time start-coeff end-coeff
- derivative center-time offset start-coeff end-coeff
- ratio_energy start-time end-time start-coeff-0 end-coeff-0 start-coeff-1 end-coeff-1

In SAILS, we represent time parameters as real numbers, usually between 0 and 1, where "0" maps to the segment beginning and "1" maps to the segment end. Coeff parameters refer to MFSC's or MFCC's depending on the algorithm. As a result, avg_cg 0.0 0.3 0 10 defines a measurement that extracts the average spectral center of gravity between MFSC's 0–10 from the first 30% of each segment. Since we usually take derivative measurements at segment boundaries, we specify their time intervals as an offset (in milliseconds) around both sides of a center time. Consequently, derivative 1.0 20 0 9 defines a measurement that computes time derivatives

for MFCC's 0–9 from a 40-millisecond sequence of frames centered at the end of each segment.

2.2 First Stage

A first-stage search is conducted via a graphical user interface, making measurement pool creation a highly interactive process. The SAILS interface can optimize the free parameters of a generalized algorithm to best discriminate between a pair of confusable phonetic classes. In this section, we discuss the capability of the SAILS interface.

2.2.1 Experiment Specification

Given speech signals and phonetic transcriptions for a set of utterances, we can use the SAILS interface to conduct an experiment in two phases. First, we define a pair of phonetic classes for our experiment; then, we specify the free parameters of the measurement algorithm whose discriminative performance we wish to evaluate.

To define phonetic classes, we enter phone labels, such as those in the phonetic transcriptions. In Figure 2-1, we entered the ARPAbet transcription labels `[ih ix]` as Class 1 and `[ah ax ax-h]` as Class 2. SAILS then collected the MFSC and MFCC tokens that represent each occurrence of [ɪ], [ɨ], [ʌ], [ə], and [ə^{h}] in the set of utterances. Although context-independent tokens were gathered in this example, we have the freedom to specify context before and/or after each phonetic class by entering labels in the appropriate fields.

Once SAILS collects speech tokens for our phonetic classes, we can select a generalized measurement algorithm and specify its free parameters. We show three examples in Figure 2-2, using the average spectral center of gravity algorithm.

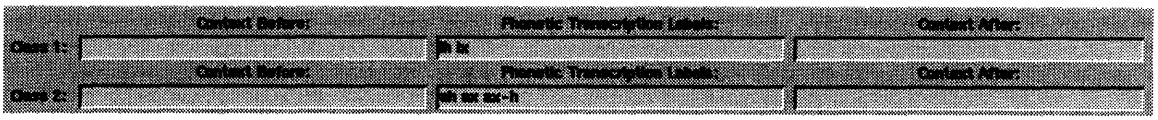
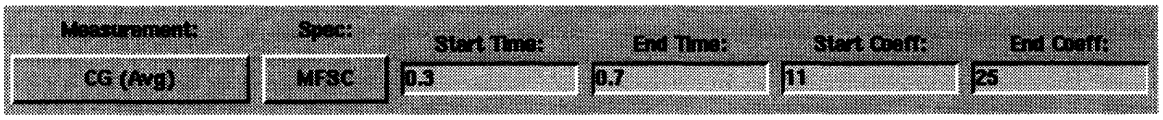
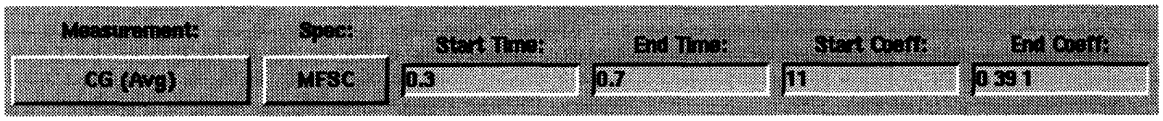


Figure 2-1: Phonetic Class Definition

(a) Complete Measurement



(b) One-dimensional Optimization



(c) Two-dimensional Optimization

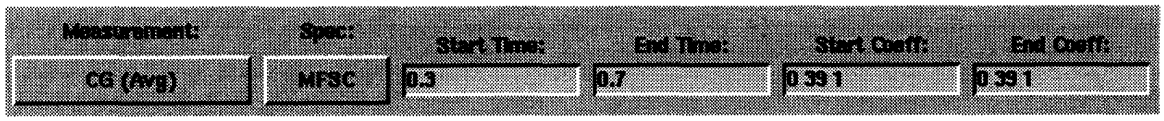


Figure 2-2: Free Parameter Specification

In Figure 2-2(a), we defined a complete measurement by specifying all four parameters. Here, we entered `[0.3]` as **start time**, `[0.7]` as **end time**, `[11]` as **start coeff**, and `[25]` as **end coeff**. Thus, we defined a measurement that extracts the average spectral center of gravity between 933–2444 Hz[1] from the middle 40% of each segment. We can evaluate the discriminative performance of this measurement through a scatter plot, as revealed in Section 2.2.3.

In Figure 2-2(b), we specified a measurement optimization instead of defining a complete measurement. We entered the range list `[0 39 1]` as **end coeff**, telling SAILS to optimize the upper frequency boundary over MFSC's 0–39 in increments of 1 coefficient. SAILS then scored the discriminative performance of the 40 possible measurements, displaying the results in an optimization graph (see Section 2.2.4).

Similarly, we can make SAILS optimize multiple parameters—all of them, if so desired. In Figure 2-2(c), we specified a two-dimensional optimization, telling SAILS to optimize both **coeff** parameters.

In addition to single-algorithm specification, SAILS can extract a series of measurements simultaneously. For example, we can append several complete measurements to any of the specifications in Figure 2-2. In the case of an optimization, SAILS

[1]MFSC 11 is centered at 933 Hz, while MFSC 25 is centered at 2444 Hz.

appends our additional measurements, whose parameters are fixed, to each measurement it creates from the range list(s), before scoring for discriminative performance.

2.2.2 Fisher's Linear Discriminant

In the SAILS interface, we evaluate how effectively measurement algorithms discriminate between two classes of speech tokens through scatter plots and optimization graphs. Both types of output are based on Fisher's linear discriminant [3], which we examine in this section.

Given a series of measurements to extract from two classes of speech tokens, SAILS determines the separability of the classes with Fisher's linear discriminant. SAILS extracts the measurements from each speech token, storing the N-dimensional result in a feature vector for the token. Thus, every token can be represented as a point in the N-dimensional space. If the feature vector is an effective discriminator, points belonging to each class will form two distinct clusters in N-space.

Although techniques exist to measure the separation of clusters in N-space, they can be computationally intensive, especially when considering thousands of points in a large number of dimensions. Since the process of quantifying separation is repeated numerous times during measurement optimization, computation is at a premium. If we project the N-dimensional points onto a line, we can still get an idea of class separability, but at a much smaller computational cost.

As Figure 2-3 reveals, however, the choice of the projection line is critical. Clearly, projecting onto $\mathbf{w}$ in the left graph provides better separation than its counterpart on the right. Fisher's linear discriminant measures the maximum separation between two classes by finding the best projection line, $\mathbf{w}^*$.

Suppose SAILS extracts measurements from n tokens, creating feature vectors, $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Of these, n_1 vectors belong to Class 1, and n_2 vectors belong to Class 2. Let us label these vector sets $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively.

We can project the feature vectors $\mathbf{x}_i$ onto a line $\mathbf{w}$ as follows:

$$\hat{y}_i = \mathbf{w}^t \mathbf{x}_i \tag{2.14}$$

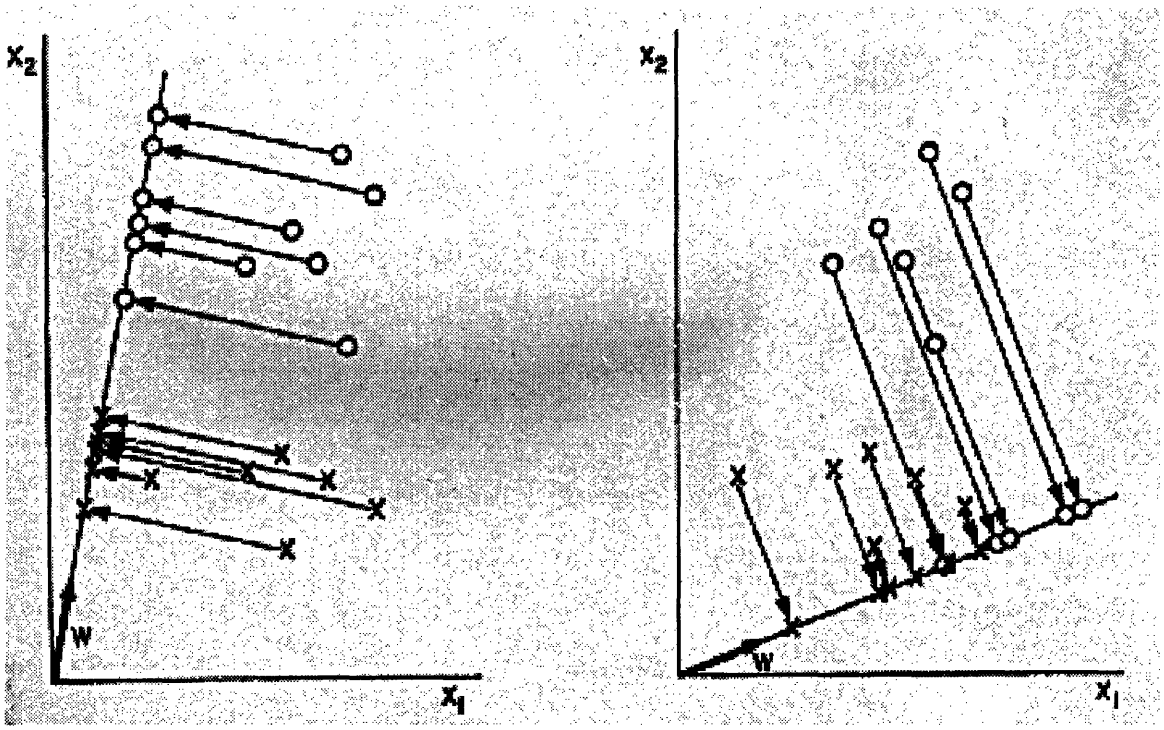


Figure 2-3: Fisher Linear Projection [3]

In this manner, we can map vectors in $\mathcal{X}_1$ and $\mathcal{X}_2$ to two sets of points on a line, $\hat{\mathcal{Y}}_1$ and $\hat{\mathcal{Y}}_2$. Furthermore, we can quantify separation between these projected sets by the difference in their means. The larger the mean difference relative to some measure of variance, the better the separability.

The Fisher criterion function is based on such reasoning:

$$J(\mathbf{w}) = \frac{|\hat{m}_1 - \hat{m}_2|^2}{\hat{s}_1^2 + \hat{s}_2^2} \tag{2.15}$$

where $\hat{m}_i$ is the sample mean of $\hat{\mathcal{Y}}_i$ and $\hat{s}_i^2$ is the *scatter*, $\hat{s}_i^2 = \sum_{\hat{y}\epsilon\hat{\mathcal{Y}}_i}(\hat{y} - \hat{m}_i)^2$.

Using the N-dimensional sample means of the vector sets, $\mathbf{m}_i = (1/n_i)\sum_{\mathbf{x}\epsilon\mathcal{X}_i}\mathbf{x}$, we can rewrite Equation 2.15 as an explicit function of $\mathbf{w}$:

$$J(\mathbf{w}) = \frac{\mathbf{w}^t S_B \mathbf{w}}{\mathbf{w}^t S_W \mathbf{w}} \tag{2.16}$$

where

$$S_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t \tag{2.17}$$

$$S_W = \sum_{\mathbf{x}\epsilon\mathcal{X}_1}(\mathbf{x} - \mathbf{m}_1)(\mathbf{x} - \mathbf{m}_1)^t + \sum_{\mathbf{x}\epsilon\mathcal{X}_2}(\mathbf{x} - \mathbf{m}_2)(\mathbf{x} - \mathbf{m}_2)^t \tag{2.18}$$

Then, the projection line $\mathbf{w}^*$ that maximizes J is:

$$\mathbf{w}^* = {S_W}^{-1}(\mathbf{m}_1 - \mathbf{m}_2) \tag{2.19}$$

Finally, we substitute $\mathbf{w}^*$ into Equation 2.16 to find Fisher's linear discriminant:

$$D = J(\mathbf{w}^*) = (\mathbf{m}_1 - \mathbf{m}_2)^t {S_W}^{-1}(\mathbf{m}_1 - \mathbf{m}_2) \tag{2.20}$$

2.2.3 Scatter Plots

Given speech tokens for a pair of phonetic classes, we can display a scatter plot in the SAILS interface to view the two-class separability. As described in the previous section, SAILS creates a feature vector of extracted measurements for each token and projects these vectors in the linear direction that maximizes separation between points of differing classes.

We can view the projected points in a scatter plot, like the one pictured in Figure 2-4. Here each point represents a speech token; the x-direction is the Fisher direction, while the y-direction is random (to make distinct points more visible).

In this example, we attempted to discriminate between [ɪ,ɨ] (Class 1) and [ʌ,ə,əʰ] (Class 2), using the **avg_cg** measurement from Figure 2-2(a)—that is, average spectral center of gravity extracted from the middle 40% of each segment within frequency bounds of 933–2444 Hz.

As shown in the plot, we can split the projection line in two. Most of the squares, which represent [ʌ,ə,əʰ] tokens, lie to the left of the divide, while most of the circles, which depict [ɪ,ɨ] tokens, fall to the right. As a result, we can classify tokens on the left as Class 2 and tokens on the right as Class 1, resulting in a classification accuracy of

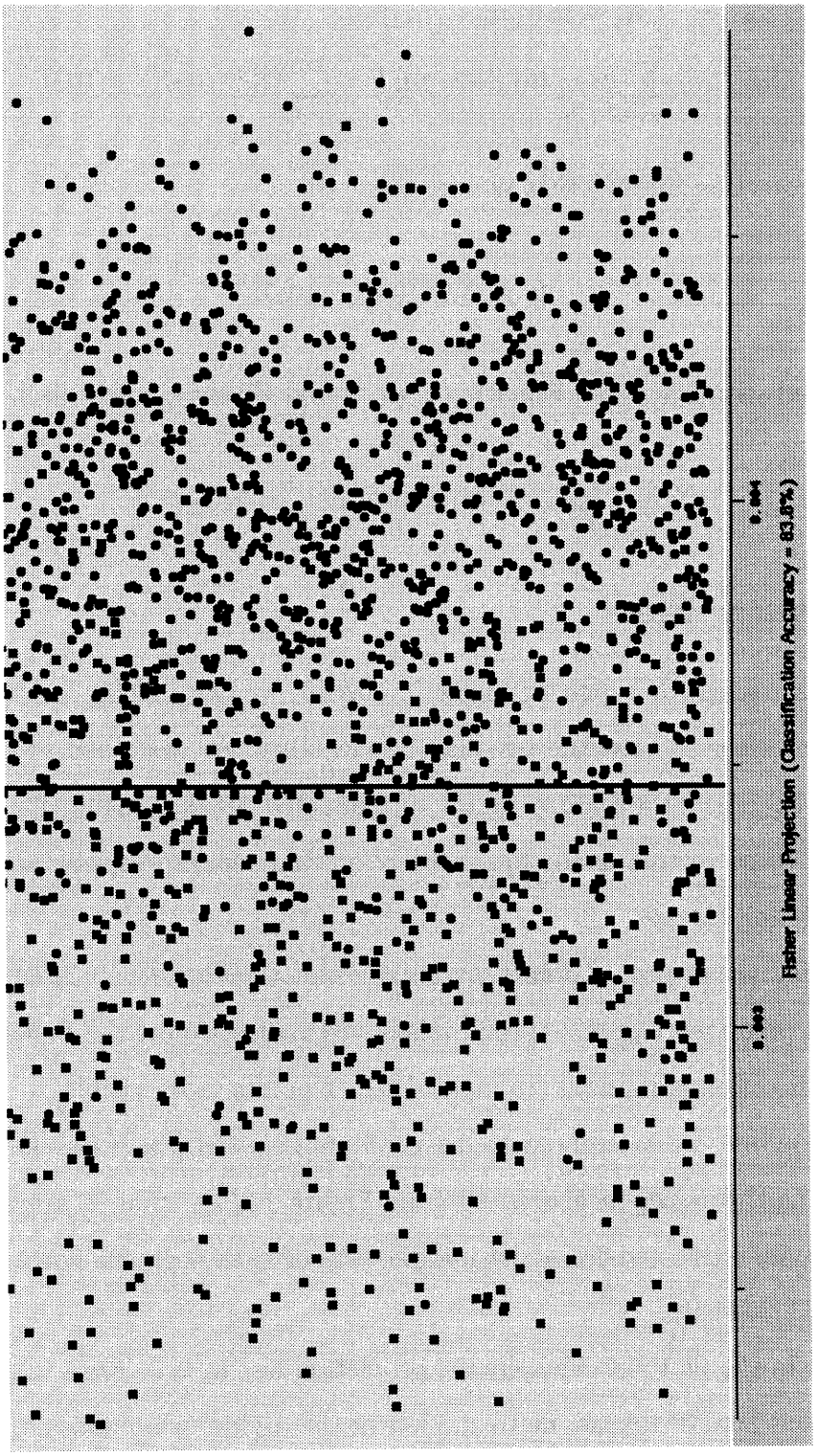


Figure 2-4: Scatter Plot with [ɪ,ɨ] (left) and [ʌ,ə,ə^{h}] (right)

83.8%. In order to maximize accuracy, SAILS places the dividing line at the optimal horizontal location, the location at which the fewest number of tokens are trapped on a side resulting in misclassification.

In addition to reflecting classification accuracy, SAILS scatter plots allow us to evaluate measurements through two main features. First and foremost, scatter plots give us an idea of the separability of the two phonetic classes in consideration. We can view individual points to determine whether tokens in differing classes tend to cluster or not. This enables us not only to experiment with new measurements, but to verify that existing measurements serve as useful discriminators. In Figure 2-4, separability is fairly good; we can see two clouds of points, separated along class lines, rather than one large clump of intermingled points. Of course, the clouds are not entirely distinct, owing to the fact that these phonetic classes are particularly confusable, but as we can witness, even a one-dimensional measurement can provide valuable discrimination. Hopefully, by increasing our feature vector with the right measurements, we can view even further increases in separability.

The second advantage of scatter plots is the ability to investigate individual tokens, especially outliers, points that appear out of place when compared to other points in their phonetic class. We can click on points in the plot to view token information, information like the token's phonetic label, the speaker who produced the token, and the utterance to which the token belongs. In the future, we will be able to see a spectral representation of the selected speech token in the context of its utterance, with graphics representing measurement-extracted features superimposed on the spectrogram. This will assist us not only in verifying that generalized algorithms are indeed extracting features they were designed to, but in viewing why a measurement makes a particular token into an outlier. Such inspection could provide clues to creating better measurements, perhaps by leading to improved parameter choices or by highlighting latent acoustic cues that can best be captured through new algorithms.

2.2.4 Optimization Graphs

Given speech tokens for a pair of phonetic classes, we can optimize the free parameters of a measurement algorithm, viewing the results in an optimization graph.

When we specify optimization ranges in the interface, SAILS varies algorithm parameters accordingly, creating a set of possible measurements. SAILS extracts each measurement, in turn, from the tokens, using the resulting feature vectors to compute Fisher's linear discriminant. Finally, SAILS graphs the discriminant scores, revealing which parameter values yield the best discrimination.

Although the feature vectors for each possible measurement have a unique projection line $\mathbf{w}^*$, comparing discriminant values of different vector sets is fair because the same measure of one-dimensional separation is used throughout (see Equation 2.20). For example, this method is robust with regard to feature vectors whose sizes vary with parameter values, since separability is scored after projection onto a line.

In Figure 2-5, we see a one-dimensional optimization graph, the result of an experiment we specified in Figures 2-1 and 2-2(b), an experiment in which we attempt to optimize **avg_cg**'s **end coeff** parameter to discriminate between [ɪ,ɨ] and [ʌ,ə,ə^{h}]. In the graph, **end coeff** choices are on the x-axis with corresponding discriminant scores on the y-axis. The circles represent local maxima, the largest of which occurs when **end coeff** is 25. Thus, **avg_cg 0.3 0.7 11 25** is the most optimal measurement in the specified range. We can click on any graph point to view a scatter plot of the corresponding measurement. For example, clicking on the big circle displays a scatter plot for the optimal measurement.

It is important to note, however, that several parameter values near the peak produce discriminants that are roughly equal. Therefore, if we choose any value between 23–26 and scatter plot the resulting measurement, we should see roughly the same degree of separation between phonetic classes. In general, the ability to view discriminant values near a maximum can be extremely beneficial, allowing us to collapse several similar measurements into one. For example, after optimizing **avg_cg** to discriminate between various vowel pairs, we might notice that certain

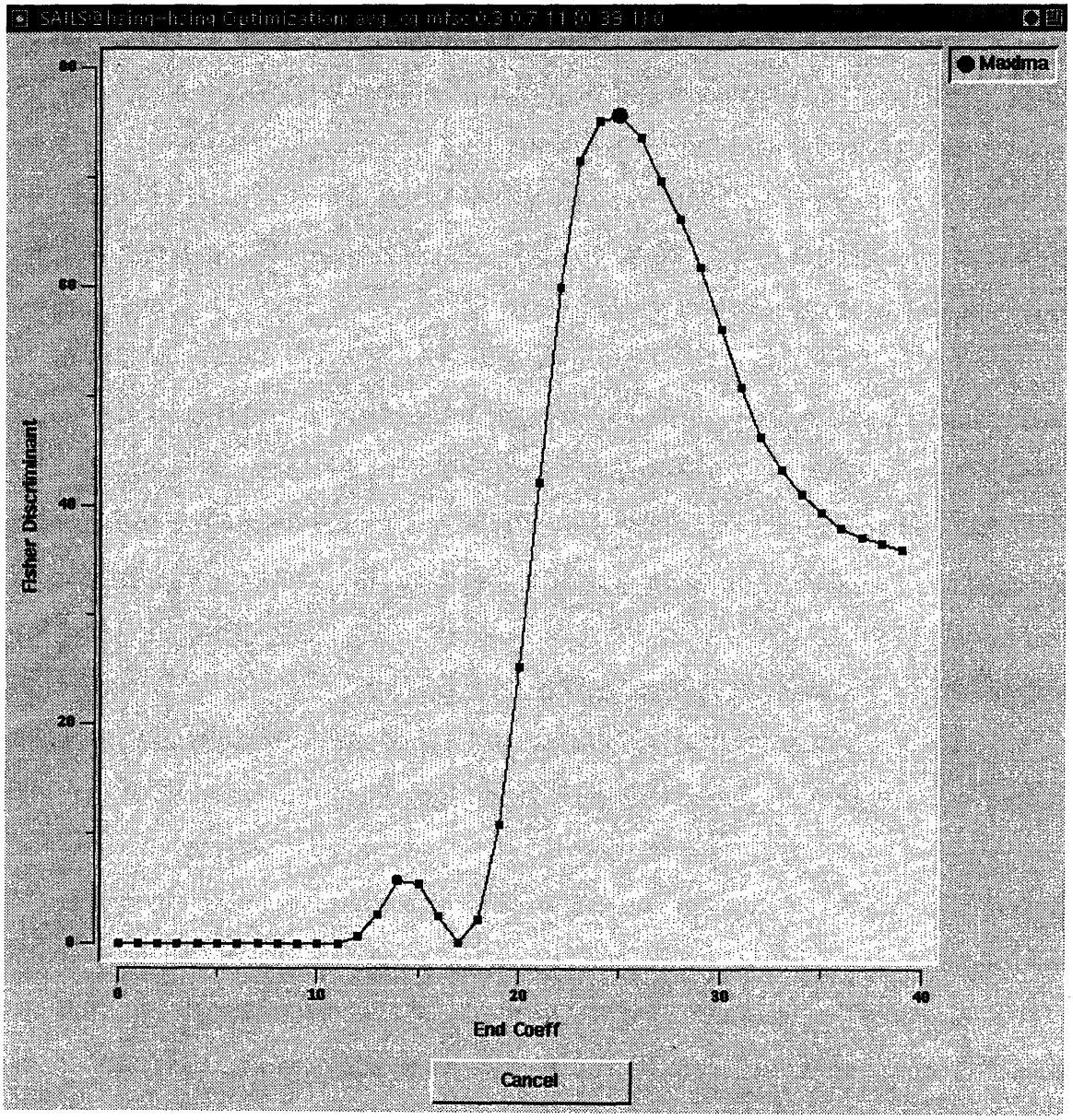


Figure 2-5: One-dimensional Optimization Graph

measurements have similar (but not identical) parameter values. Knowing parameter tolerance for each pairwise-optimal measurement, we might find a single parameter set that comes close to yielding optimal separation for several vowel distinctions.

In Figure 2-6, we see a two-dimensional optimization graph, the result of optimizing **avg_cg** over both **coeff** parameters—see Figure 2-2(c). Here we need to graph three variables: two parameters and their corresponding discriminant scores. SAILS conveys this three-dimensional relation through a contour plot, in which **start coeff** appears on the x-axis, **end coeff** appears on the y-axis, and the black contour lines represent equal discriminant values. Regions where lines are close together indicate steep changes in discriminative performance.

Essentially, a contour plot is the overhead view of a surface and allows us to easily locate maxima, once again represented by circles. In addition to the global maximum at (11,25), or 933–2444 Hz, we can see a smaller peak at (1,16), or 267–1317 Hz. Therefore, this optimization reveals two locally-optimal measurements: **avg_cg 0.3 0.7 1 16** and **avg_cg 0.3 0.7 11 25**. In fact, the frequency ranges roughly correspond to first and second formant locations, verifying that extracting these important features maximizes discrimination.

In general, a two-dimensional optimization is a good idea, especially in the context of interval boundaries. For example, it is natural to optimize **start time** and **end time** together. Moreover, two-dimensional optimizations offer us a better chance of finding the globally-optimal parameter set for a given measurement algorithm. To see this, consider optimizing both **coeff** parameters with a series of one-dimensional optimizations. First, we could guess a **start coeff** and optimize **end coeff** over MFSC's 0–39. Then, we would set **end coeff** to the peak result and optimize **start coeff**. Even if we continued this process until both parameters stabilized, we have no guarantee that the resulting **coeff** values are globally optimal; depending on our initial guess, we might gravitate towards the smaller peak.

Clearly, optimizing multiple parameters offers us better perspective. In this vein, we could attempt to optimize more than two parameters, all four in the case of **avg_cg**. Although such a sweeping experiment precludes us from a fine-grain search,

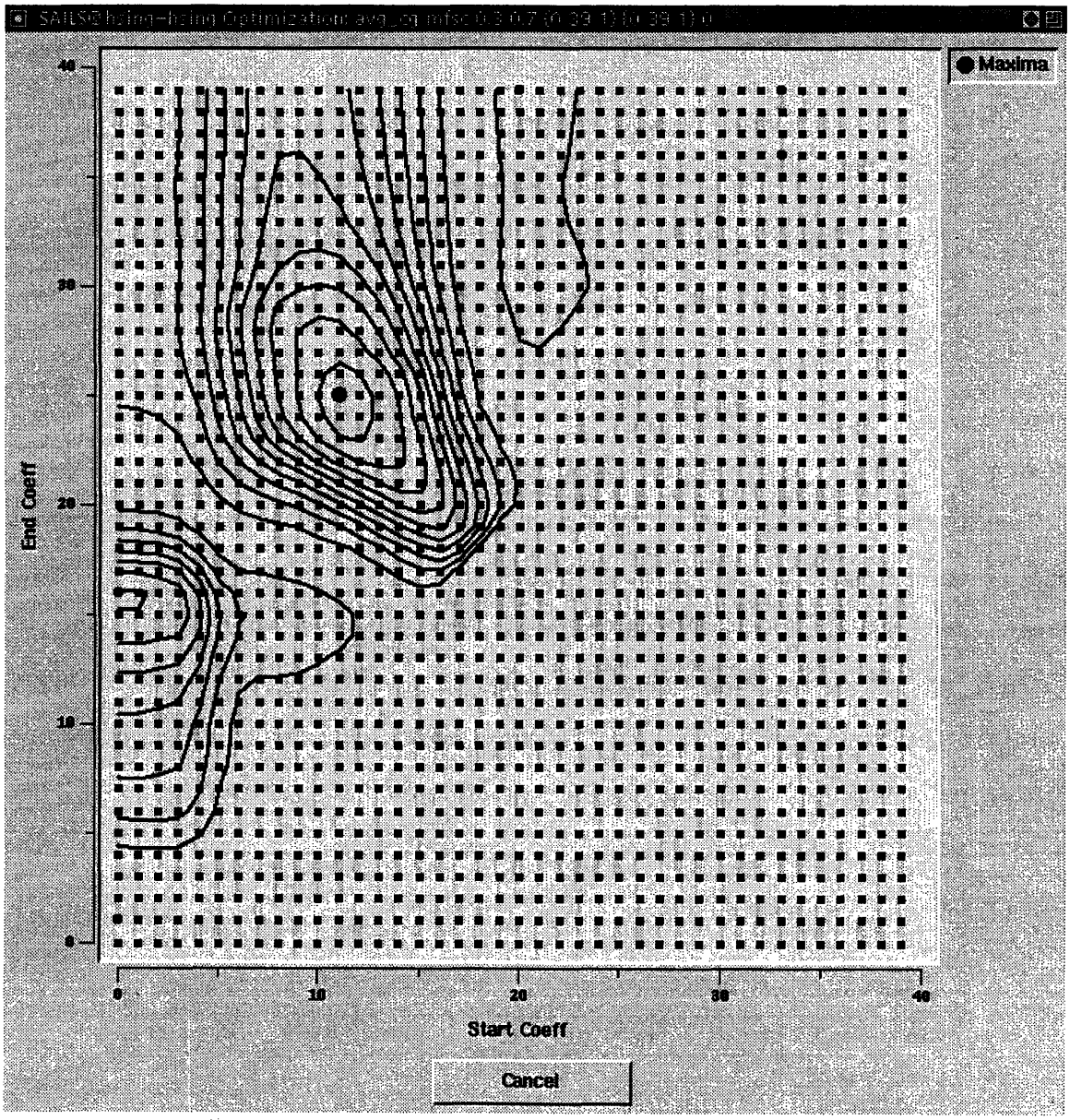


Figure 2-6: Two-dimensional Optimization Graph

coarse parameter optimizations are certainly reasonable. The SAILS interface does not graphically represent higher-dimensional optimizations, but SAILS can write the possible measurements and their discriminant scores to a file. Sorting such output can reveal general peak locations, from which we can conduct high-resolution optimizations in one or two dimensions.

2.3 Second Stage

In SAILS, the second stage is a batch process in which we perform a best-first search to automatically discover the N-best measurements from a measurement pool.

Table 2-1 shows a collection of measurements which often serves as a generic feature vector for our segmental speech recognition system; we include three MFCC averages, two MFCC derivatives, and duration.

As an exercise, we fed these generic measurements to the second-stage search engine, training and testing a mixture-diagonal Gaussian classifier on the TIMIT corpus (see Section 3.1). Table 2-2 reveals the results of the six iterations.

Here an "x" means that the measurement was in the optimal set. On every iteration, SAILS created a trial set for each unused measurement, appending the measurement to the current optimal set. One at a time, SAILS used these trial sets to train acoustic models and test phonetic classification accuracy. The table reflects these accuracies, the highest accuracy for each iteration in bold.

On the first iteration, the optimal measurement set was empty, and the middle_avg

avg_vector	start time	end time	start coeff	end coeff
start_avg	0.0	0.3	0	11
middle_avg	0.3	0.7	0	11
end_avg	0.7	1.0	0	11
derivative	**center time**	**offset (in ms)**	**start coeff**	**end coeff**
start_deriv	0.0	20	0	11
end_deriv	1.0	20	0	11
duration				

Table 2-1: Generic Measurements with 12 MFCC's

#	start_avg	middle_avg	end_avg	start_deriv	end_deriv	duration
1	47.1	**57.5**	48.8	43.5	42.0	15.4
2	**65.1**	x	64.1	64.7	63.4	62.6
3	x	x	69.0	69.4	**70.1**	68.5
4	x	x	71.4	72.2	x	**72.3**
5	x	x	73.4	**74.1**	x	x
6	x	x	**74.9**	x	x	x

Table 2-2: Results of Second-stage Search through Generic Measurements

measurement resulted in the best accuracy at 57.5%. SAILS added **middle_avg** to the optimal set for the second iteration, and the trial set of **start_avg** and **middle_avg** was highest with a 65.1% accuracy. Hence, SAILS added **start_avg** to the optimal set, continuing the search process.

Note that the second-stage search engine chose **end_deriv** next, while the performance of **end_avg** degraded. In fact, **end_avg** became the least significant measurement. In this case, using the five best measurements (i.e., not including **end_avg**) reduces the feature vector by 12 dimensions at a slight performance cost of less than 1%.

This result suggests an alternative use for the second stage, namely one of paring down measurement sets instead of extending them. In the Phillips-Zue approach of Section 1.3, we would use the first and second stages in conjunction to append to a given set of measurements. Used in this manner, the second stage would reveal which pairwise-optimal, first-stage measurements were relevant when trained across all phones.

The experiment above, on the other hand, indicates that feeding chunks of generic measurements to the second stage can prove to be fruitful. Finding lower-dimensional feature vectors that provide comparable performance is extremely desirable, as we can save on computation while improving parameter modeling. On this note, we should divide the generic measurements into even smaller pieces. In the next chapter, we do exactly this, establishing minimal sets of cepstral coefficient measurements.

2.4 Chapter Summary

In this chapter, we described the SAILS framework and revealed its capability. First, we examined the generalized measurement algorithms currently in the framework. Specifically, we stated what features the algorithms were designed to extract and in what phonetic context they might be most useful. We also discussed how we compute measurements with each algorithm.

Second, we demonstrated the SAILS graphical interface, which allows us to conduct a first-stage search. We explained how to specify phonetic classes, define acoustic measurements, and optimize algorithm parameters. In addition, we showed some sample output—a scatter plot and optimization graphs—and discussed how they allow us to evaluate the discriminative performance of measurements. We also described Fisher's linear discriminant, which SAILS uses to produce such output.

Finally, we revealed the power of the second stage, pointing out that we can use this search engine to improve measurement sets in two ways, the first by dissecting feature vectors currently in use, the second, by extending them.

Chapter 3

Measurement Discovery

In this chapter, we use SAILS in a series of experiments in an attempt to improve the performance of SUMMIT, our segmental speech recognizer [11]. Although the ultimate objective is to improve recognition performance, we constrain ourselves to phonetic classification to remove the variability of segmentation present during recognition. We feel, however, that improvements in classification should translate into improvements in recognition.

3.1 Phonetic Classification Overview

To perform context-independent phonetic classification, we first transform a speech waveform into a compact signal representation. From this representation, we extract a set of acoustic measurements from each speech segment, storing the results in a feature vector. Then, a classifier compares each segment's feature vector with trained acoustic models for a complete set of phones, and labels the segment as the phone with the closest-matching model.

In our experiments, we vary the acoustic measurements, but maintain the other classification components. This section covers these common elements: the speech data, the pre-measurement signal processing performed on the data, and the acoustic models of the classifier.

Training Set	# Speakers	# Utterances	# Unique Utts.	# Tokens
train	462	3,696	1716	142,910
Testing Sets	# Speakers	# Utterances	# Unique Utts.	# Tokens
test	118	944	474	35,697
core-test	24	192	192	7,333
dev	50	400	265	15,057

Table 3-1: TIMIT Training and Testing Sets

3.1.1 Corpus

The TIMIT acoustic-phonetic corpus [12], which includes American English utterances from 630 male and female speakers, serves as the speech database for our experiments. We chose TIMIT for its phonetic balance and its widespread use in phonetic classification experiments. To create the corpus, each speaker read ten utterances through a Sennheiser noise-cancelling microphone, two labeled *sa*, three labeled *si*, and five labeled *sx*; TIMIT contains a waveform and its time-aligned phonetic transcription for every utterance read.

To maintain acoustic diversity, we ignore the *sa* utterances, which are common to all speakers. Instead, we draw our training and testing data from the *si* series, which contains a wide and varied selection of phonetic combinations, and the *sx* series, which includes a limited number of phonetic combinations. Each *si* utterance is unique, while each *sx* utterance is repeated by seven speakers.

Table 3-1 shows the NIST data sets[1] used for training and testing [8], where *core-test* is a subset of *test*. To ensure the integrity of our results, speakers are disjoint between *train*, *test*, and *dev*. Furthermore, *sx* utterances in *train* do not appear in the three test sets.

In this thesis, we conduct first-stage measurement optimizations on three sets of 1000 randomly-selected *train* utterances; there are overlaps among these first-stage data sets. For second-stage searches, which consist of a series of classification trials, we train our classifier with *train* and test on *dev*. To show that second-stage feature vectors improve performance regardless of speaker or utterance, we also report

[1]The *sa* utterances are excluded.

IPA	TIMIT	Example	IPA	TIMIT	Example
ɑ	aa	bob	ɨ	ix	debit
æ	ae	bat	iʸ	iy	beet
ʌ	ah	but	ǰ	jh	joke
ɔ	ao	bought	k	k	key
ɑʷ	aw	bout	k□	kcl	k closure
ə	ax	about	l	l	lay
əʰ	ax-h	suspect	m	m	mom
ɚ	axr	butter	n	n	noon
ɑʸ	ay	bite	ŋ	ng	sing
b	b	bee	ɾ̃	nx	winner
b□	bcl	b closure	oʷ	ow	boat
č	ch	choke	oʸ	oy	boy
d	d	day	p	p	pea
d□	dcl	d closure	□	pau	pause
ð	dh	then	p□	pcl	p closure
ɾ	dx	muddy	ʔ	q	bat
ɛ	eh	bet	r	r	ray
l̩	el	bottle	s	s	sea
m̩	em	bottom	š	sh	she
n̩	en	button	t	t	tea
ŋ̩	eng	Washington	t□	tcl	t closure
▯	epi	epenthetic silence	θ	th	thin
ɝ	er	bird	ʊ	uh	book
eʸ	ey	bait	uʷ	uw	boot
f	f	fin	ü	ux	toot
g	g	gay	v	v	van
g□	gcl	g closure	w	w	way
h	hh	hay	y	y	yacht
ɦ	hv	ahead	z	z	zone
ɪ	ih	bit	ž	zh	azure
—	h#	utterance initial and final silence			

Table 3-2: IPA Symbols for TIMIT Phonetic Transcription Labels

Class	Phones	Class	Phones	Class	Phones
ɑ	ɑ,ɔ	e^y	e^y	o^y	o^y
æ	æ	f	f	p	p
ʌ	ʌ,ə,$ə^h$	g	g	r	r
$ɑ^w$	$ɑ^w$	h	h,ɦ	s	s
$ɑ^y$	$ɑ^y$	ɪ	ɪ,ɨ	š	š,ž
b	b	i^y	i^y	t	t
□	$b^□$,$p^□$,$d^□$,$t^□$,$g^□$,$k^□$	ǰ	ǰ	θ	θ
č	č	k	k	ʊ	ʊ
d	d	l	l,l̩	u^w	u^w,ü
ð	ð	m	m,m̩	v	v
ɾ	ɾ	n	n,n̩,ɾ̃	w	w
ɛ	ɛ	ŋ	ŋ,ŋ̩	y	y
ɝ	ɝ,ɚ	o^w	o^w	z	z

Table 3-3: Lee's 39 Phonetic Classes

classification accuracies on *test* and *core-test*.

TIMIT transcriptions label each phonetic token as one of the 61 phones in Table 3-2. Before determining classification accuracy, however, we collapse the actual labels and the classifier's hypotheses to Lee's set of 39 phonetic classes [6]. Table 3-3 reveals these classes, which omit glottal stops and silences.

3.1.2 Signal Representation

SAILS includes generalized algorithms designed to extract measurements from Mel-frequency spectral coefficient (MFSC) and Mel-frequency cepstral coefficient (MFCC) signal representations. Characteristic of such representations is a Mel-frequency warping, which approximates the frequency response of the inner ear. The procedure used to derive Mel-based representations from a speech waveform is as follows:

- Sample the waveform at 16 kHz.
- Multiply the samples by a 25.6 ms Hamming window advanced at a frame rate of 5 ms.
- For each frame, compute a 256-point discrete Fourier transform.
- Square each frame's DFT coefficients to obtain a power spectrum.

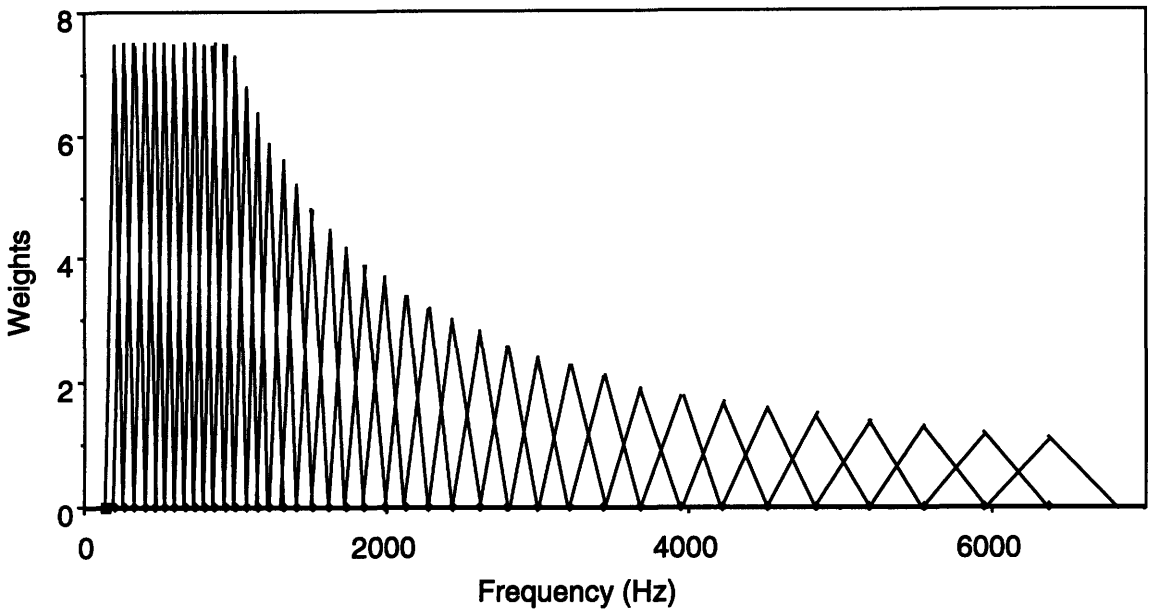


Figure 3-1: Forty Mel-frequency Triangular Filter Banks

- Pass the spectra through a set of 40 Mel-frequency triangular filter banks to generate 40 MFSC's (see Figure 3-1).
- Take the cosine transform of the MFSC's to obtain 40 MFCC's.

Many speech recognizers use a feature vector of low-order MFCC's and their time differences. With SAILS, however, we can extend such vectors with measurements that extract spectral features present in MFSC's.

3.1.3 Principal Components Analysis

Before presenting feature vectors to the classifier, we perform a principal components analysis (PCA) on the extracted measurements [3]. We conduct such analysis, which we base on pooled correlation, for three reasons. First, PCA rotates the feature vector to make the dimensions as orthogonal as possible; this allows us to use simpler acoustic models that assume independence between dimensions. Second, PCA normalizes each extracted measurement, removing the effects of scale. Third, PCA ranks the dimensions by how much of the pooled variance they explain; this allows us to reduce the size of the feature vector, keeping only the highest-ranking dimensions if

so desired. As we learn in Section 3.7, however, dimensionality reduction through PCA does not necessarily translate into improved performance.

3.1.4 Acoustic Models

In our experiments, the acoustic models of the classifier are mixtures of diagonal Gaussian distributions, where each distribution $\Pr_i(\mathbf{x})$ is the product of univariate normal densities [3]:

$$\Pr_i(\mathbf{x}) = \frac{1}{(\sqrt{2\pi})^N \cdot \prod_{j=1}^{N} \sigma_j} \cdot \exp\left(-\frac{1}{2} \cdot \sum_{j=1}^{N} \frac{(x_j - \mu_j)^2}{{\sigma_j}^2}\right) \tag{3.1}$$

where $\mathbf{x}$ is a feature vector of size N and the jth extracted measurement, x_j, has mean μ_j and variance ${\sigma_j}^2$. We take the product of univariate densities on the assumption that the dimensions of the feature vector are independent.

A phone is modeled with mixtures of $\Pr_i(\mathbf{x})$:

$$\Pr(\mathbf{x}) = \sum_{i=1}^{M} m_i \cdot \Pr_i(\mathbf{x}) \tag{3.2}$$

$$\text{such that } \sum_{i=1}^{M} m_i = 1$$

where M is the number of mixtures and m_i is a mixture weight. To keep training and testing computation in check, especially during a lengthy second-stage search, we set the maximum number of mixtures to 16 for all classification experiments.

The Gaussians are trained with k-means clustering, which introduces some randomization. Consequently, we can run the same phonetic classification experiment twice and receive different classification accuracies. From our experiments, however, we conclude that such training noise is within a few tenths of a percent.

3.2 Approach

In the approach described in Section 1.3, we use the second stage to search through a pool of first-stage measurements that are designed to extract discriminating features between pairs of phones.

Although we could begin this search with an empty optimal set, thus creating a feature vector entirely of complex measurements, it appears better to begin with a set of standard, cepstral coefficient measurements for two reasons. First, cepstral measurements are widely used as general discriminators in the context of all phones, a fact we should harness. Second, a feature vector composed only of complex measurements might be extraordinarily large, considering the number of pairwise phonetic distinctions that must be made. Therefore, we should begin with a baseline of cepstral measurements and use SAILS to extend this baseline with feature-extracting measurements, which are targeted at phones that cepstra alone are unable to resolve.

Following such reasoning, we can use the SAILS framework to iteratively improve phonetic classification performance:

1. Train and test a classifier with a set of acoustic measurements.
2. From the results, determine the pairs of phonetic classes that cause the greatest confusion.
3. Use the SAILS interface to create a measurement pool, with a few measurements to discriminate between each of these confusable class pairs.
4. Use the SAILS second stage to generate the N-best measurements from the measurement pool and the initial set (from Step 1).
5. Designate these N measurements as the new measurement set.
6. Repeat as desired.

In subsequent sections, we attempt to follow this procedure.

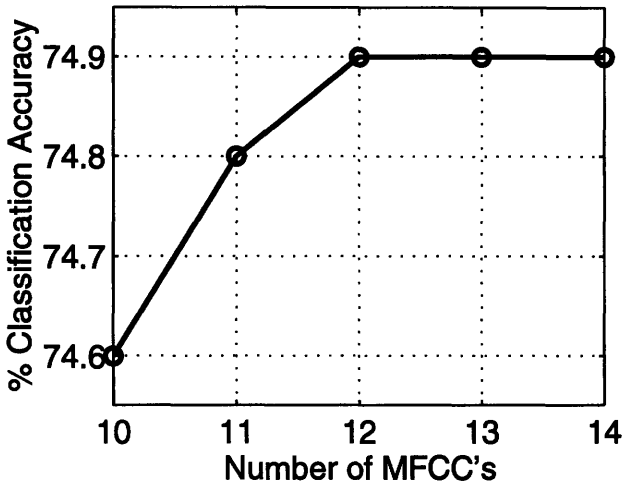


Figure 3-2: Performance of Generic Measurements with Various MFCC's

3.3 Baseline Experiments

Our first task was to establish a baseline of cepstral coefficient measurements to highlight major phonetic confusions and to serve as the initial measurement set for experiments with the second stage.

We chose a generic feature vector of three MFCC averages, two MFCC derivatives, and duration—much like the set listed in Table 2-1. Furthermore, we conducted several classification trials, varying the number of MFCC's in the cepstral measurements. Figure 3-2 shows the results, which are based on the *dev* set. Since performance levels off at 12 MFCC's for our classification system, we selected a 74.9% baseline of 61 dimensions ($= 5 \times 12 + 1$).

As a prelude to our first-stage experiments, we determined which pairs of phonetic classes were most often confused by the baseline system. Figure 3-3 is a matrix that illustrates the degree of confusion between pairs of the 39 phonetic classes. The rows of the matrix represent actual classes from phonetic transcriptions, while the columns represent classifier-hypothesized classes. The shade of each matrix element reflects the frequency of misclassification[2]; therefore, the darker the square, the greater the confusion.

[2]In an ordinary confusion matrix, diagonals are extremely dark, since the classifier is correct for most phonetic tokens. Here we do not shade diagonals to highlight misclassification.

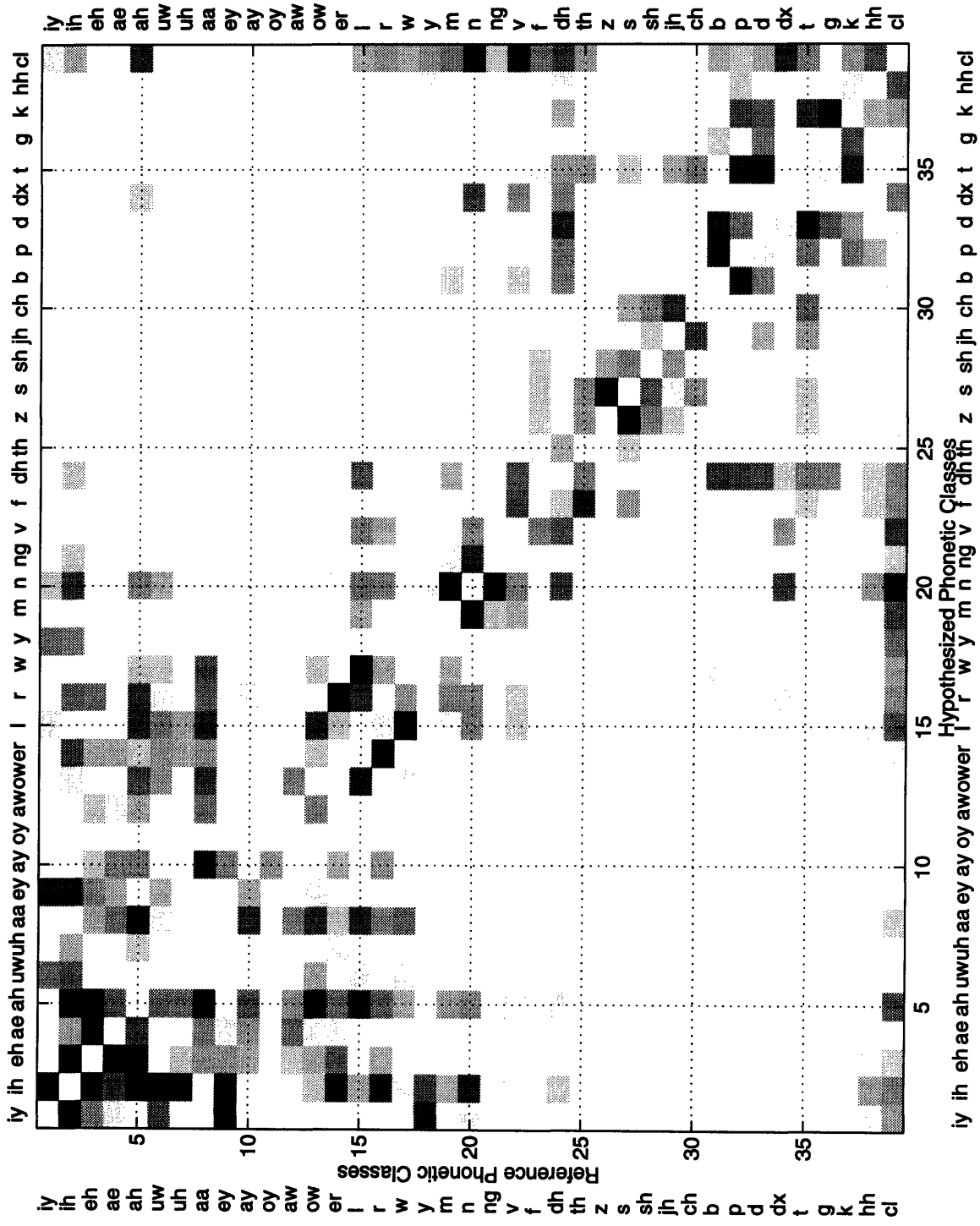


Figure 3-3: Confusion Matrix for Baseline Measurements

The figure reveals major sources of confusion: [ɪ,ɨ] and [ʌ,ə,ə^{h}], [ɪ,ɨ] and [ɛ], [ɪ,ɨ] and [i^{y}], [ɝ] and [r], [m] and [n], and [s] and [z]. Not surprisingly, each of these pairs consists of phones that are similarly produced—vowels, nasals, and strong fricatives. In addition, much of the vowel confusion lies between the short, neutral vowels (i.e., [ɪ], [ɨ], [ʌ], [ə], [ə^{h}], and [ɛ]).

In an effort to alleviate this baseline confusion and improve performance, we targeted the aforementioned phonetic classes in a first- and second-stage search. The next few sections discuss these experiments.

3.4 First-stage Search

Our objective during the first-stage search was to create a pool of complex measurements that best discriminated between the baseline's most confusable pairs of phonetic classes. In each case, we optimized a few of the generalized measurement algorithms at hand, seeking to capture discriminating features.

Since the location and dynamics of formants are important acoustic cues for vowel distinction, we optimized **avg_cg** and **delta_cg** for the confusable pairs of vowel classes. For [s] and [z], spectral images are quite similar, but the presence of low-frequency energy can indicate [z]'s voicing; thus, we optimized **energy_ratio** and **avg_energy** for these strong fricatives. Finally, since formants often fall into [m], while remaining relatively static near [n], we optimized **delta_peak** and **delta_cg**. In all cases, our focus was to find optimal frequency intervals given a reasonable time interval—the segment middle for vowels, the entire segment for fricatives, and the segment boundaries for nasals.

From these searches, a few measurements emerged with true discrimination potential. Table 3-4 summarizes this pool of measurements. We based the merit of a measurement on three criteria. First, a two-dimensional optimization should produce at least one clear and smooth peak. We ignored spikes in an optimization graph, deeming them to be a product of the data set and unlikely to be universally applicable. Second, a measurement should show qualitative and quantitative evidence of

Mean Fisher Score	Class 1	Class 2	Measurement
74.7	ɪ,ɨ	ʌ,ə,əʰ	avg_cg 0.3 0.7 11 25
55.1	ɪ,ɨ	ɛ	avg_cg 0.3 0.7 0 8
52.3	ɪ,ɨ	iʸ	avg_cg 0.3 0.7 8 28
48.9	s	z	ratio_energy 0.0 1.0 0 2 6 12

Table 3-4: First-stage Measurement Pool

discrimination. That is, a scatter plot should reveal separation between tokens of differing classes and the Fisher discriminant value, a measure of two-class separability, should be relatively high. We selected an empirical threshold of about 40 for the Fisher score. Finally, an optimization should be repeatable; optimizing on all three thousand-utterance data sets should produce similar optimization graphs, scatter plot separations, and Fisher discriminant values.

As an example, let us consider the experiment in which we optimized avg_cg's coeff parameters to generate avg_cg 0.3 0.7 11 25 as the optimal discriminator for [ɪ,ɨ] and [ʌ,ə,əʰ]. In Table 3-5, we list the top three optimization results for each of the first-stage data sets, labeled A, B, and C. The repeatability of this search is evident. Specifically, the optimal frequency interval, MFSC's 11–25, is consistent for all three data sets. Moreover, the Fisher scores for the three trials all lie in the low- to mid-seventies. One particular trial, which we use to demonstrate the SAILS interface in Chapter 2, yielded the optimization graph depicted in Figure 3-4. In the graph, we see two clear, smooth peaks that correspond to first and second formant frequencies (see Section 2.2.4). Therefore, discriminative performance is maximized when based on formant location, a result that agrees with our acoustic-phonetic knowledge.

Similarly, avg_cg optimizations between other vowel classes produced dual peaks around F1 and F2. This is as expected because the average spectral center of gravity algorithm was designed to extract formant frequency estimates. The consistent two-peak results verify that this algorithm is indeed functional and effective in pairwise discrimination.

As Table 3-4 shows, we were most successful in optimizing avg_cg for [ɪ,ɨ]. Furthermore, the ratio of spectral energy was useful for [s] versus [z]. In general, these

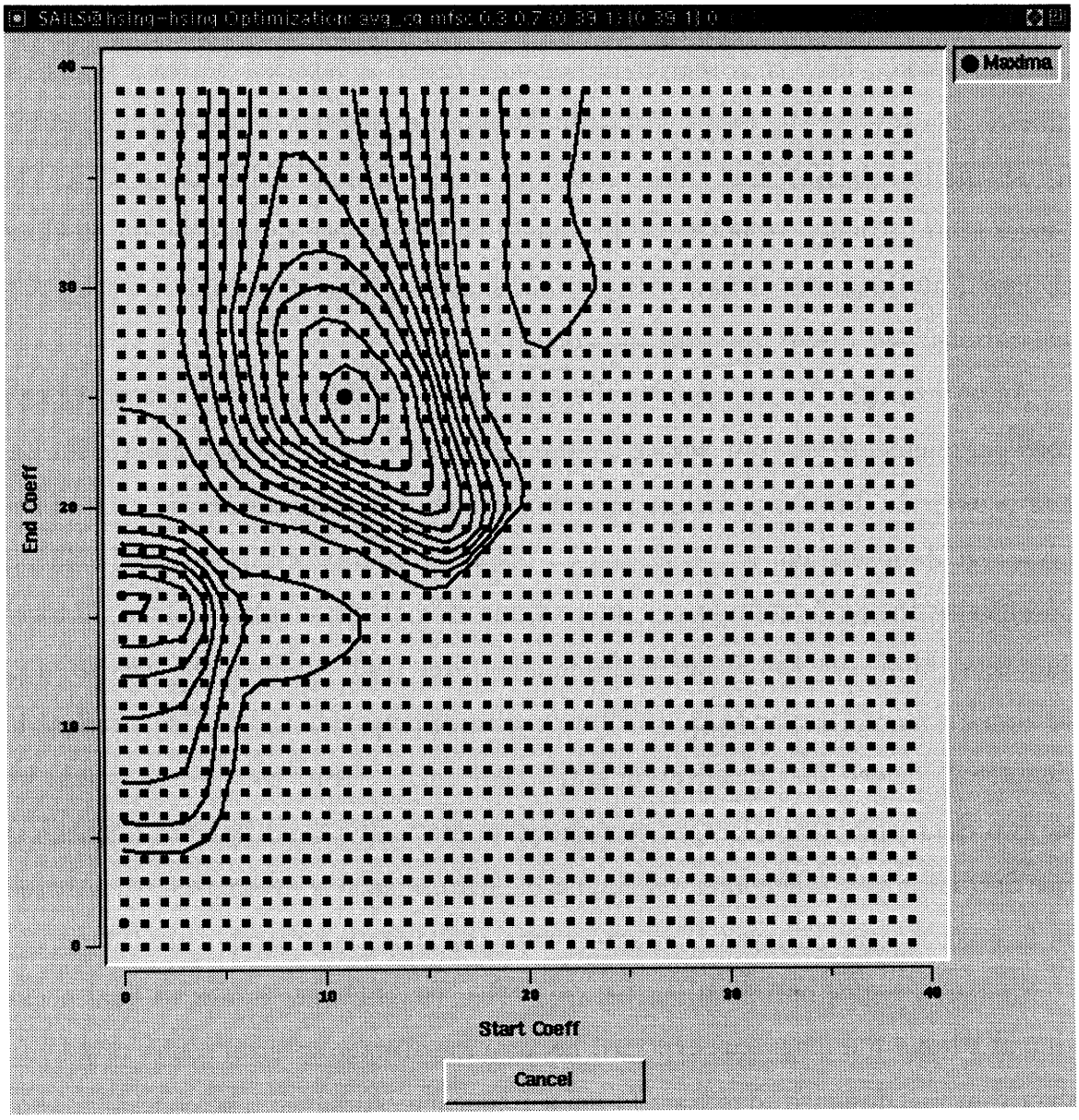


Figure 3-4: Two-dimensional avg_cg Optimization Graph for [ɪ,ɨ] and [ʌ,ə,ə^{h}]

Set	#ɪ,ɨ	#ʌ,ə,əʰ	Fisher	start coeff	end coeff
A	3067	1640	**76.9**	**11**	**25**
			75.6	12	25
			75.5	11	26
B	3136	1640	**71.7**	**11**	**25**
			71.6	12	25
			71.3	12	24
C	3157	1601	**75.6**	**11**	**25**
			75.2	11	24
			74.1	12	24

Table 3-5: Results of **avg_cg** Optimization for [ɪ,ɨ] and [ʌ,ə,əʰ]

two algorithms, which extract static features, were more robust and yielded better results than **delta_cg** and **delta_peak**, which extract dynamic features. Consequently, [m] and [n] separation was virtually non-existent.

3.5 Second-stage Search

At this juncture, our goal was to improve upon baseline classification performance. From the first-stage search, we had a pool of complex measurements optimized to alleviate some of the baseline's phonetic confusion (see Table 3-4), so we conducted a second-stage search through this measurement collection. As described in Section 3.2, the baseline measurements served as the initial set.

Since the complex measurements were tailored to best discriminate between specific phonetic classes, we expected that confusion between these classes would diminish, even if classification accuracy did not exceed the baseline of 74.9%. As the search progressed, however, we discovered that extending the baseline with our complex measurements was often detrimental.

Classification accuracy for extended feature vectors was either at the baseline or a few tenths of a percent below. More importantly, when we compared baseline phonetic confusion to the extensions' phonetic confusion, we often discovered that pairs of classes were slightly more confusable *after* adding a measurement which was optimized to diminish confusion between them.

In fact, we obtained similar results in accuracy and phonetic confusion for a variety of classification experiments with complex measurements. For example, including amplitude with **avg_cg** or using **avg_peak** with optimized center-of-gravity parameters was not fruitful. We even attempted different types of acoustic models, such as the full-covariance Gaussian and mixtures of full-covariance Gaussians. We hoped that such models, which can capture correlation between feature vector dimensions, would benefit from our complex measurements. However, they too showed slight losses like mixtures of diagonal Gaussians.

As an example (for diagonal mixtures), let us compare results from the baseline to those of the baseline plus **avg_cg 0.3 0.7 11 25**, a complex measurement optimized to discriminate between [ɪ,ɨ] and [ʌ,ə,əʰ]. Table 3-6, which is based on three classification trials, shows confusion matrix diagonals: the number of correctly-classified tokens in each of the 39 phonetic classes. The right-most column is the change in each class from the additional **avg_cg** measurement. The sum of these changes is -61, which effects a net loss in performance of about 0.1%.

Although the changes are erratic as a whole, there is a $+20$ change in the vowels and semivowels (rows [iʸ] to [y]). This is promising because **avg_cg** is supposed to increase vowel classification. However, the targets of this experiment, [ɪ,ɨ] and [ʌ,ə,əʰ] suffer a combined loss of -27. This is curious because we expected the greatest positive changes to originate in this pair of classes.

We offer two possible explanations for this behavior. First, the pairwise manner in which we optimize our complex measurements might not generalize to classification with all phones. Given a first-stage measurement that is optimized in the context of only two classes, a 61-phone classifier might not improve in a particular distinction because of all the other phones it can select. The only way to guarantee optimality is to optimize measurement parameters through classification trials. Second, feature vectors composed of complex and baseline measurements might be too large to model well. The number of model parameters increases with feature vector size, eventually resulting in poorer parameter estimates and lower performance. Our baseline experiments suggest that dimensionality could be a problem because performance plateaus

Class	Baseline	+avg_cg	Δ
iʸ	1398	1414	+16
ɪ	**2644**	**2644**	**0**
ɛ	565	600	+35
æ	431	427	−4
ʌ	**1052**	**1025**	**−27**
uʷ	206	210	+4
ʊ	9	8	−1
ɑ	967	990	+23
eʸ	488	485	−3
ɑʸ	498	504	+6
oʸ	64	64	0
ɑʷ	91	87	−4
oʷ	226	219	−7
ɜˠ	1014	995	−19
l	1517	1522	+5
r	1241	1264	+23
w	604	596	−8
y	205	186	−19
m	875	869	−6
n	1930	1902	−28
ŋ	130	128	−2
v	429	435	+6
f	737	729	−8
ð	485	490	+5
θ	83	91	+8
z	727	713	−14
s	1732	1729	−3
š	378	371	−7
ǰ	169	165	−4
č	115	121	+6
b	521	513	−8
p	573	583	+10
d	447	444	−3
ɾ	347	356	+9
t	901	904	+3
g	209	190	−19
k	959	945	−14
h	362	358	−4
□	9193	9185	−8
Sum	34522	34461	−61

Table 3-6: Diagonals of Confusion Matrices

Measurements	Class	ɪ,ɨ	ʌ,ə,əʰ
Baseline only	ɪ,ɨ	90.2	9.8
	ʌ,ə,əʰ	25.7	74.3
Baseline +avg_cg	ɪ,ɨ	92.3	7.7
	ʌ,ə,əʰ	27.8	72.2

Table 3-7: Two-class Confusion Matrices

for generic feature vectors larger than 61 dimensions (12 MFCC's in Figure 3-2).

To verify that a pairwise-optimal measurement yields improved performance when added to the baseline, we repeated the **avg_cg 0.3 0.7 11 25** experiment, only training and testing phones in the relevant classes, namely [ɪ,ɨ] and [ʌ,ə,əʰ]. Table 3-7 is a matrix that reveals the two-class phonetic confusion before and after appending **avg_cg**. The rows represent [ɪ,ɨ] and [ʌ,ə,əʰ] in the phonetic transcriptions, while the columns represent classifier hypotheses. As expected, the additional complex measurement increases the percent of correctly-classified [ɪ,ɨ] tokens, from 90.2% to 92.3%. On the other hand, the percent of correct [ʌ,ə,əʰ] tokens suffers, decreasing from 74.3% to 72.2%.

Two-class experiments with other phonetic class pairs exhibited similar inconsistencies. Since we expected our first-stage optimized measurements to improve—or at the very least, maintain—binary classification performance for the target classes, we concluded that dimensionality was indeed a reason for our inconsistent results. Consequently, we decided to reduce baseline dimensionality before appending complex measurements. In this vein, we attempted PCA reduction techniques to no avail. However, as revealed in the next section, the second stage proved to be quite effective at dimensionality reduction.

3.6 Dimensionality Reduction Experiments

In this section, we discuss our efforts to improve upon the baseline through dimensionality reduction. Smaller feature vectors mean fewer acoustic model parameters, which has several advantages: parameter estimation is enhanced for a given body of training data, less memory is necessary to store acoustic models, and less run-time

computation is required, increasing the speed of recognition.

Therefore, our objective was to use the second stage to reduce the 61-dimensional, generic feature vector of Table 2-1, while retaining—or better yet—increasing performance. Ideally, we wanted to divide the feature vector entirely and feed 61 single-coefficient measurements to the second stage.

3.6.1 Second-stage Search through MFCC Averages

In an effort to restrict the amount of computation necessary for decomposition, we began with a second-stage search through 30 **avg_vector** measurements, covering the bottom ten MFCC's in each of the three time intervals (0.0–0.3, 0.3–0.7, and 0.7–1.0). Figure 3-5 reveals the gains in classification accuracy as the search progressed, while Table 3-8 lists the order in which the top twenty measurements were selected.

From the figure, we can witness the value of each additional cepstral average. In fact, we achieve over 50% classification accuracy with merely five averages. As we increasingly append more measurements, however, the improvements in performance taper, eventually reaching a plateau of about 66%.

The initial selections form a basis for coarse discrimination, a basis common enough to classify a majority of phonetic tokens. In support of such reasoning is the fact that the initial selections are all low-order MFCC's, which model the general spectral shape of a frame. In each of the three time slices, the search chose low-order MFCC averages at first and later appended higher-order coefficients. Although the search did not choose MFCC's in numerical order, it eventually returned to acquire the coefficients that were skipped.

This skipping effect is most likely due to training noise during k-means clustering, which can create a non-deterministic classification accuracy for a given set of measurements. Since classification accuracy is the search criterion during each iteration, training noise can make a slightly lower-performing measurement supersede a better measurement.

With the effects of training noise in mind, we can conjecture a general search trend: in each time interval, the search began with the lowest-order MFCC average

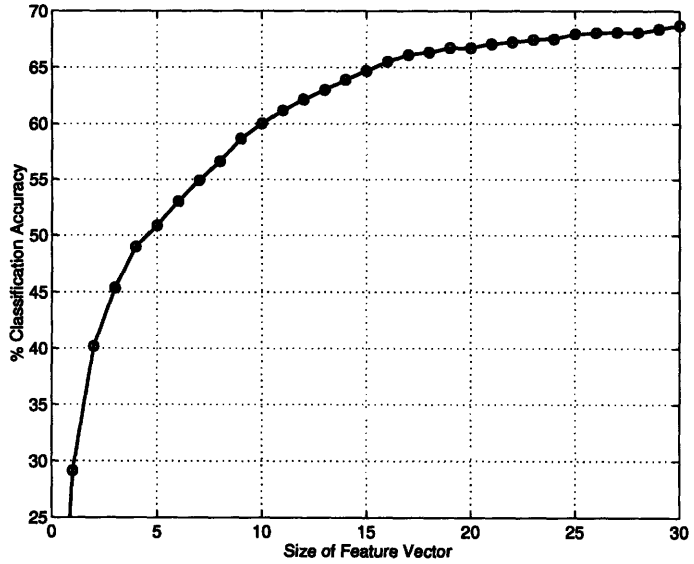


Figure 3-5: Results of Second-stage Search through 30 avg_vector Measurements

Dim	% Acc	0.0–0.3	0.3–0.7	0.7–1.0	MFCC
1	29.14		★		1
2	40.20		★		0
3	45.36		★		3
4	49.00		★		2
5	50.89		★		4
6	53.04	★			3
7	54.89	★			1
8	56.63			★	0
9	58.63	★			0
10	60.02		★		5
11	61.18		★		6
12	62.15			★	1
13	62.98			★	3
14	63.90		★		7
15	64.67	★			2
16	65.50	★			9
17	66.13			★	2
18	66.35		★		8
19	66.73			★	9
20	66.73	★			4

Table 3-8: Selection Order for Top 20 avg_vector Measurements

and monotonically increased the order during subsequent iterations. This hypothesis is intuitively agreeable because increasing MFCC order is akin to increasing spectral resolution. Therefore, in each iteration, the search increased spectral resolution in the time interval that provided the greatest discrimination.

As Table 3-8 reveals, the search exclusively chose middle averages at the outset. This is not surprising, since we expect the center of each phone to contain its spectral essence, the characteristic spectral shape that distinguishes it from other phones. Another reason for the middle-average selection is the fact that the **avg_vector** algorithm is geared to capture the relatively static nature of the segment center, as opposed to the dynamic co-articulation effects more likely to exist at the segment ends. This is because the average of fairly constant MFCC's will reflect these constant values, while the average of changing MFCC's will not capture their dynamics.

3.6.2 Second-stage Search through MFCC Derivatives

To make use of dynamics, we included MFCC derivatives in our next experiment, which is summarized in Table 3-9. In the previous search, we realized only incremental gains in performance after 16 averages. Therefore, we began a new search with averages of sixteen top-performing[3] coefficients in the initial measurement set: averages of 8 MFCC's in the middle time interval, 4 MFCC's at the beginning, and 4 MFCC's at the end. We also included **duration** in the initial set. As a search pool of generic measurements for the second stage, we designated the three lowest-order, single-coefficient derivatives at both segment boundaries (with 20-millisecond offsets), in addition to a few low-order averages not in the initial set.

As the table shows, the search immediately chose four **derivative** measurements, two at the segment start and two at the end. At both boundaries, the MFCC-0 derivatives surfaced before their MFCC-1 counterparts, an order that supports our spectral resolution hypothesis. Furthermore, performance benefits a great deal from low-order derivatives and duration, resulting in a classification accuracy over 72%—nearly 6%

[3]We omitted **avg_vector 0.0 0.3 9 9**, assuming its selection was due to training noise.

Dim	% Acc	Measurement
1	x	duration
2–9	x	avg_vector 0.3 0.7 0 7
10–13	x	avg_vector 0.0 0.3 0 3
14–17	x	avg_vector 0.7 1.0 0 3
18	70.79	derivative 0.0 20 0 0
19	71.76	derivative 1.0 20 0 0
20	72.12	derivative 0.0 20 1 1
21	72.61	derivative 1.0 20 1 1
22	73.31	avg_vector 0.7 1.0 4 4

Table 3-9: Results of Second-stage Search through Derivatives and Averages

greater than twenty-one dimensions of averages only. Finally, the 22-dimensional generic feature vector generates an accuracy within two percent of the baseline, suggesting that high performance with low dimensionality is indeed plausible. This set the stage for our next experiment, in which we performed a modified second-stage search to incrementally build a feature vector of cepstral coefficient measurements.

3.6.3 Generic Measurement Search

In this search, only duration belonged to the initial set. The trial pool consisted of the three lowest-order averages (avg_vector 0.0 0.3 0 0, avg_vector 0.3 0.7 0 0, and avg_vector 0.7 1.0 0 0) and the two lowest-order derivatives (derivative 0.0 20 0 0 and derivative 1.0 20 0 0). After each iteration, the modified second stage replaced the selected measurement with a similar measurement of incremented order. For example, the generic measurement search chose avg_vector 0.3 0.7 0 0 in the first iteration and replaced its selection with avg_vector 0.3 0.7 1 1. In this manner, the search always chose from a complement of five generic measurements.

As is evident from the replacement process, the generic measurement search operates under the hypothesis that a regular second-stage search through the 61 single-coefficient baseline measurements would most likely yield the lowest-order averages and derivatives first and then monotonically increase their orders. In conducting the modified search, roughly the same computation is required for each iteration. Hence, we could build high-performing subsets of the baseline in an efficient and manageable

manner.

Figures 3-6 and 3-7 reveal the results of the generic measurement search through sixty dimensions. In these plots, the x-axis represents the dimensionality of the optimal feature vectors determined by the search, while the y-axis reflects their classification accuracy on the *dev* set.

In Figure 3-6, we can view large performance gains from initial selections followed by a tapering effect after approximately twenty iterations and a plateau beginning near forty dimensions. We presume that initial measurements are used for coarse discrimination and later selections for fine-grain distinctions between tokens the initial measurements could not resolve. The plateau suggests that cepstral measurements eventually cease to provide enough value for such detailed discrimination. At this point, however, complex measurements that extract discriminating features might better serve us. (We experiment with complex measurements in the next section.)

Figure 3-7 zooms in on the incremental performance gains. From the plot, we can observe that several low-dimensional, generic feature vectors beat the 61-dimensional baseline performance of 74.9%. A plateau occurs between 41–57 dimensions and is centered at 75.3–75.4%. Therefore, we consistently exceed the baseline by the margin of 0.4–0.5%. We should caution that we are optimizing performance on the *dev* set. Thus, some performance gain might be attributed to inherently optimizing for this particular set of utterances. However, as we show in Section 3.7, the increased performance of feature vectors from the generic measurement search does indeed translate to other test sets, like *test* and *core-test.*

Returning to Figure 3-7, we see a point on the left that represents *SAILS-22*, the optimal 22-dimensional generic feature vector. We should note that the composition of this feature vector is exactly the same as the 22-dimensional measurement set in the previous experiment (see Table 3-9). This gives further credence to the spectral resolution hypothesis on which we based the generic measurement search. Furthermore, we can confirm that the second stage arrives at consistent, optimal solutions in the long run, even though training noise can vary the order of selected measurements for a particular search.

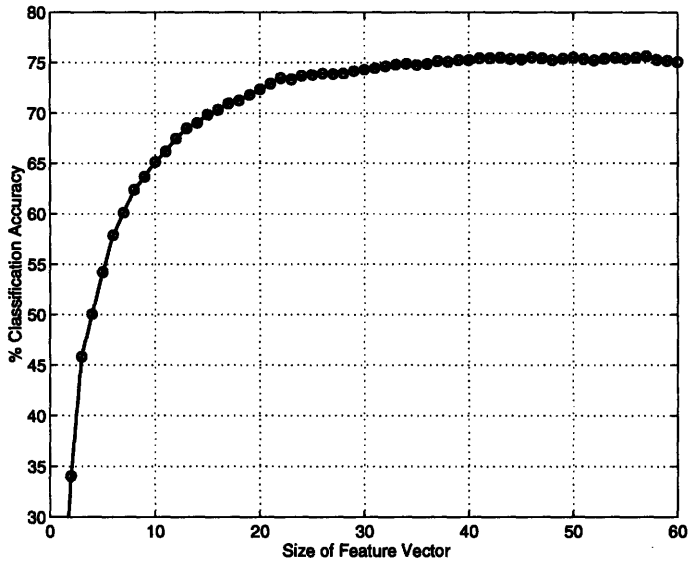


Figure 3-6: Results of Generic Measurement Search

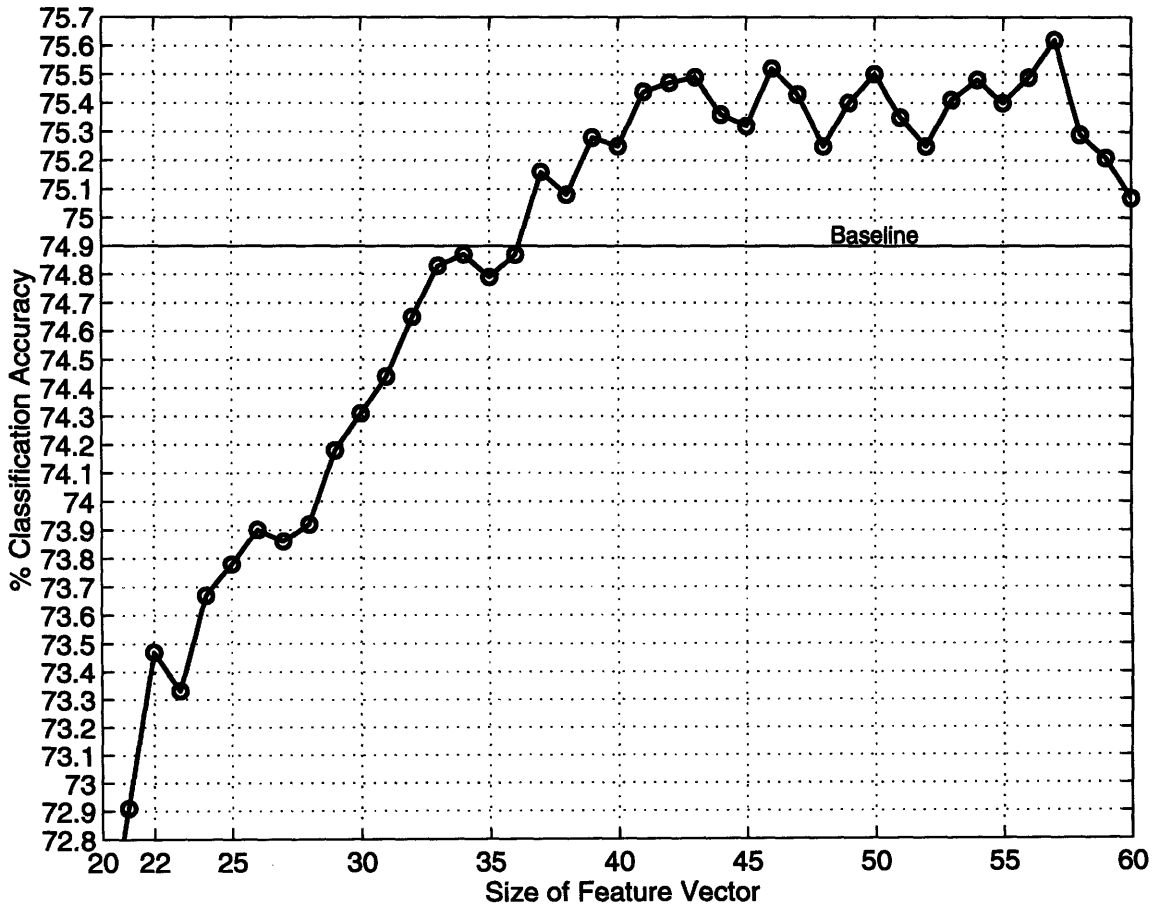


Figure 3-7: Results of Generic Measurement Search (Inset)

We can witness the effects of such training noise through the bumps in the plot. For example, *SAILS-45* and *SAILS-46*, which are probably equivalent performers, result in classification accuracies that are about 0.2% apart. At the plateau, training noise creates deviations that are less than three-tenths of a percent.

Incidentally, the downturn near 60 dimensions is most likely due to either a dimensionality problem, in which the size of the feature vector is simply too large for the models to accommodate, or the result of noisy measurements—that is, the last few dimensions might not provide any discriminative value at all.

In Figure 3-8, we can view the MFCC decomposition of selected feature vectors, namely *SAILS-22*, *SAILS-30*, *SAILS-40*, and *SAILS-50*. The five bars in each graph represent the five generic measurements: the bars at the extremes depict the start and end derivatives, while the bars in the center depict the three averages. The height of each bar represents the number of MFCC's in the corresponding measurement.

The cepstral compositions offer clues as to why *SAILS-40* and *SAILS-50* beat the baseline. Unlike the baseline, these feature vectors use more than 12 MFCC's for certain measurements. In the baseline experiments of Section 3.3, where we increased MFCC order uniformly across the five generic measurements, using more than 12 MFCC's did not appear useful. From *SAILS-40* and *SAILS-50*, we learn that higher-order MFCC's do have merit when applied to the right measurements. Evidently, removing MFCC's from certain measurements is helpful as well. Most likely, *SAILS-40* and *SAILS-50* omit higher-order MFCC's that do not provide a basis for discrimination, resulting only in noise.

In addition, the cepstral compositions reveal two characteristic shapes. A symmetrical shape defines *SAILS-22* and *SAILS-30*, which map to the incline in Figure 3-7. Here the middle average dominates, and the derivatives are scant. However, in *SAILS-40* and *SAILS-50*, which map to the plateau in Figure 3-7, the end derivative assumes greater significance, eclipsing the end average.

As described in Section 3.6.1, the center of each phone plays a large role in defining that phone. Thus, the fact that the middle averages accumulate the most MFCC's is not surprising. Clearly, the acoustic models benefit from enhanced spectral resolution

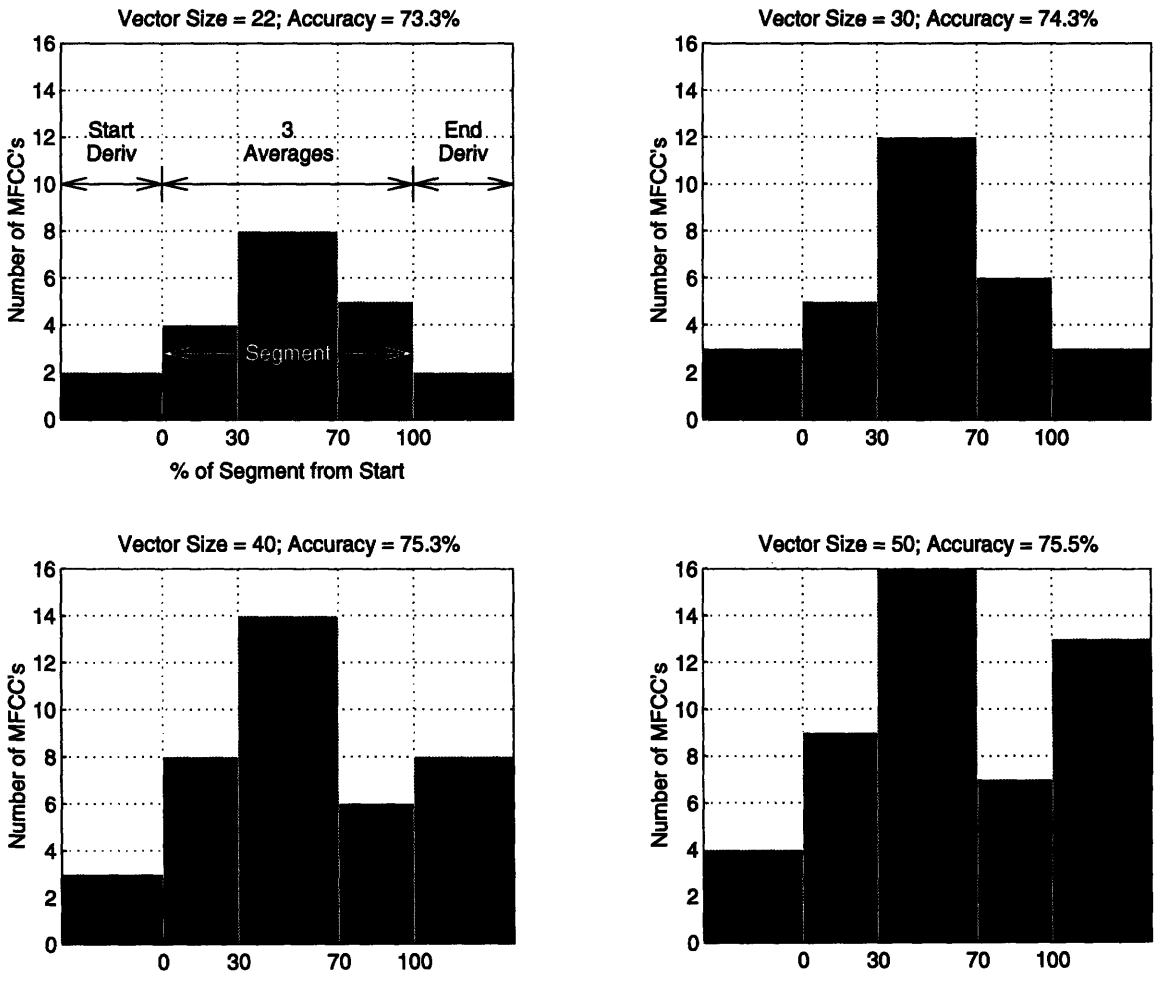


Figure 3-8: Cepstral Composition of *SAILS-xx*

at the segment center. On the other hand, the rise in MFCC order for the end derivative is somewhat curious. At some point, the most discriminating features that cepstra can capture might be the dynamics of co-articulation effects, especially the stronger effects in the latter frames of phones. Such spectral dynamics could create fluctuations in higher-order MFCC's, which the end derivative can extract.

3.6.4 Complex and Generic Measurement Search

To investigate the utility of complex measurements in feature vectors of reduced dimensionality, we conducted a generic measurement search with a trial pool of eight measurements, five generic measurements and three complex measurements (avg_cg 0.3 0.7 11 25, avg_cg 0.3 0.7 0 8, and ratio_energy 0.0 1.0 0 24). As in the previous experiment, the search replaced every generic selection with a higher-order measurement. However, complex measurements were not replaced when chosen. We began the search with an initial set of the *SAILS-22* measurements; thus, the search appended its first selection to *SAILS-22*.

Figure 3-9 is a plot of the search results. Once again, feature vector size is on the x-axis, while *dev*-set classification accuracy is on the y-axis. In addition, we label the points where the search selected a complex measurement. The dotted line traces the results of the purely-cepstral search in Section 3.6.3.

From the labels in the figure, we see that the complex measurements surface fairly early in the search. At this point, feature vectors are of low dimensionality, and their performance roughly mirrors that of the purely-cepstral search. Unfortunately, performance for the complex-generic feature vectors eventually drops, then shadows its cepstral counterpart—that is, subject to training noise, we see a similar rise to a plateau with a somewhat constant loss in performance.

Perhaps we can best explain this result by the dual-edged nature of the complex measurements: although each of these measurements assists in making a particular phonetic distinction, or even provides a basis for broad class discrimination, such as between the vowels (avg_cg) or fricatives (ratio_energy), the complex measurements likely produce degenerative noise when extracted from tokens in other phonetic

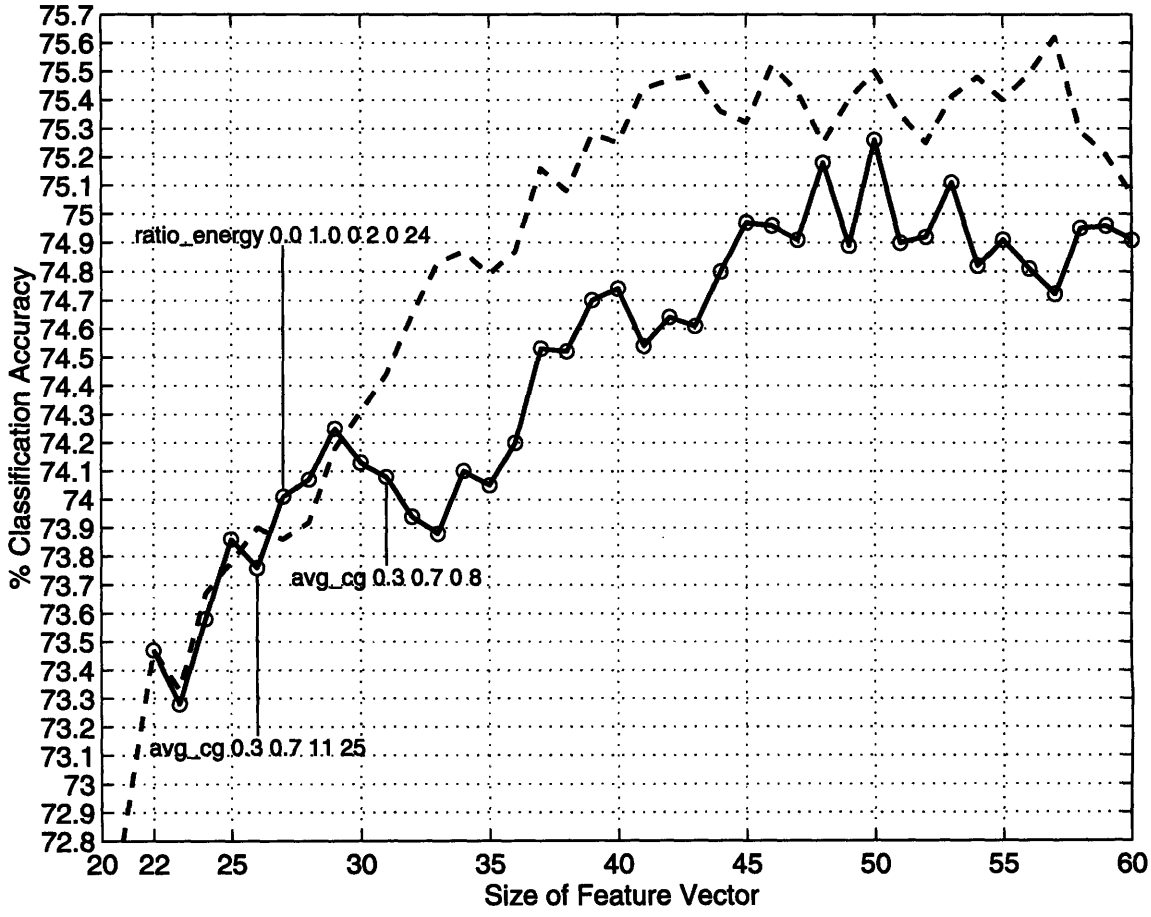


Figure 3-9: Results of Complex and Generic Measurement Search

classes.

Initially, the positives offset the negatives because the complex measurements, drawing from a high-resolution MFSC spectrum, can extract the detailed spectral features necessary to distinguish certain phones. On the other hand, the five, low-order generic measurements see low spectral resolution and cannot capture the same spectral features. However, the noisiness of the extracted complex measurements in other phonetic classes neutralizes the gains. Therefore, as shown in the overlap between solid and dotted lines, the net result is performance approximately equivalent to that of purely-cepstral feature vectors.

As we add increasing numbers of cepstra, the generic measurements improve their ability to capture spectral details. At this point, the complex measurements provide either redundant information or noise. Since our acoustic models cannot benefit from such redundancy, performance degrades due to the noise. For feature vectors of 33 or more dimensions, shifting the solid line three units to the left reveals this loss in performance (see Figure 3-10). If the three complex measurements provide sheer redundancy (and no noise), the shifted line should mirror the dotted line, since the shifted line would represent the effective feature vector size: three dimensions fewer than the actual size. As the figure illustrates, however, this is not true; the net loss due to noise is a few tenths of a percent.

Clearly, the complex measurements we used suffer under our current classification framework, even in feature vectors of fewer dimensions. To effectively use complex measurements in this scheme, we must discover measurements that assist in classifying particular phones, while returning predictable—rather than noisy—values for other phones. In the next chapter, we draw conclusions from our research and offer further insight regarding complex measurements.

3.7 Evaluation of Dimensionality Reduction

In Section 3.6.3, we used the generic measurement search to develop low-dimensional feature vectors, several of which beat the 61-dimensional baseline performance. In

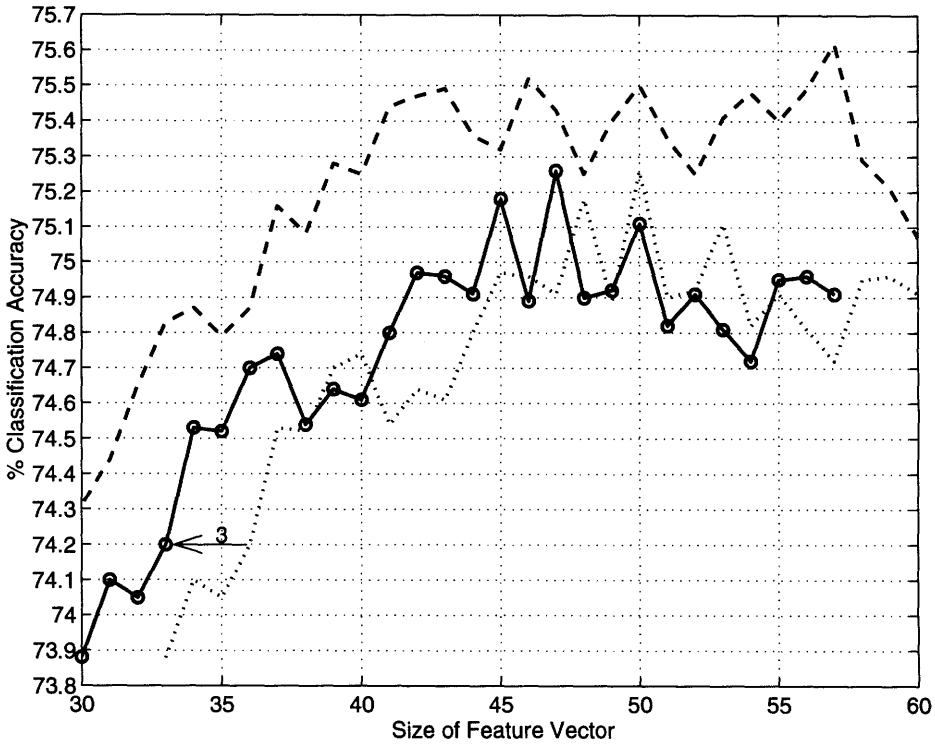


Figure 3-10: Complex-generic Results Shifted to Reveal Noise Loss

each iteration of this search, however, we selected the optimal measurement based on the highest *dev*-set classification accuracy. To some degree, we inherently optimized for *dev*-set utterances. Therefore, we tested selected feature vectors on *test* and *core-test* in an effort to demonstrate their increased performance in other domains. We should reiterate that the speakers and utterances in *test* and *core-test* are disjoint from those in *dev*, which should eliminate the gain due to inherent optimization.

Table 3-10 summarizes results for the baseline and four generic feature vectors, namely *SAILS-22*, *SAILS-30*, *SAILS-40*, and *SAILS-50* from Figure 3-8. For the pair of test sets, we show classification accuracies using two types of Gaussian-mixture models, mixtures of diagonal Gaussians and mixtures of full-covariance Gaussians [3].

In general, we can obtain better performance than the baseline with feature vectors of fewer dimensions. This is evident in the diagonal mixture results on *core-test*, where *SAILS-40* and *SAILS-50* exceed the 74.6% baseline accuracy by 0.5%, and in the full-covariance mixture results on *dev*, where *SAILS-40* and *SAILS-50* surpass the 74.6%

Mixtures	Dim	*test*	*core-test*
Diagonal	61	74.6	74.6
	50	74.8	75.1
	40	74.5	75.1
	30	73.8	73.8
	22	72.2	72.3
Full-covariance	61	76.3	76.3
	50	76.6	76.1
	40	76.7	76.3
	30	76.0	75.7
	22	74.3	74.2

Table 3-10: Performance of *SAILS-xx* with Gaussian-mixture Models

Mixtures	Dim	*test*	*core-test*
Diagonal	61	74.6	74.6
	50	74.3	74.6
	40	73.2	73.1
	30	71.8	71.7
	22	68.1	67.9

Table 3-11: Performance of PCA-reduced Baseline

baseline performance by 0.3–0.4%. These gains are on par with those we observed during the generic measurement search. The two cases where *SAILS-40* and *SAILS-50* perform slightly lower than the baseline might be due to training noise or the peculiarity of using certain acoustic modeling on a particular set of utterances.

We should note that *SAILS-22* and *SAILS-30* perform within a few percent of the baseline with approximately one-third and one-half the number of measurements, respectively. These feature vectors can be attractive alternatives if computation is at a premium.

As stated in Section 3.6, achieving comparable or better performance with fewer dimensions is extremely desirable to improve estimation of acoustic model parameters and to increase run-time recognition speed. Principal components analysis (PCA) is a common approach to dimensionality reduction (see Section 3.1.3). Hence, we reduced the 61-dimensional baseline feature vector through PCA; Table 3-11 lists the results.

Unlike *SAILS-xx*, the PCA-reduced feature vectors do not exceed baseline accuracy on either test set. In addition, SAILS feature vectors outperform PCA-reduced

vectors of comparable size. Specifically, *SAILS-22* and *SAILS-30* beat their 22- and 30-dimensional counterparts by about 2% and 4%, respectively.

This is interesting in light of Figure 3-8, which shows that *SAILS-22* and *SAILS-30* use less than the 12 MFCC's that characterize the baseline system. In effect, the generic measurement search and PCA could both choose from the 61 baseline measurements. The search based its selection purely on performance, while PCA chose dimensions that explained the most pooled variance.

As Tables 3-10 and 3-11 reveal, however, the PCA reduction technique does not necessarily translate into the best performance for a given feature vector size. Moreover, dimensionality reduction through PCA does not decrease run-time measurement computation. For example, to obtain its 22-dimensional feature vector, PCA had to extract all sixty-one baseline measurements before reduction. *SAILS-22*, on the other hand, only extracts twenty-two measurements during run-time.

To conclude this section, we compare our results with those in the literature. Specifically, we would like to highlight that *SAILS-xx* feature vectors provide comparable performance with less computation.

Goldenthal, who uses statistical trajectory models to capture acoustic feature dynamics across a segment, reports a phonetic classification accuracy of 75.2% [4]. His experimental setup is different from ours; he uses a 76-dimensional feature vector—averages of 15 MFCC's across segment thirds, derivatives of 15 MFCC's at segment boundaries with 35-millisecond offsets, and duration. Furthermore, he trains a single full-covariance Gaussian model on a superset of our *train* set and he tests on a subset of our *test* set: *sx* utterances from 50 selected speakers.

Although a direct comparison was not possible, we tested our *SAILS-30* full-covariance mixtures on the 590 *sx* utterances in *test*, receiving a 75.8% accuracy. The sixty-one phonetic models averaged 3.2 mixtures a piece. Since computation is proportional to the number of mixtures times the square of the feature vector size, *SAILS-30* requires approximately half of Goldenthal's computation—$3.2 \times 30^2 \approx \frac{1}{2} \cdot (1 \times 76^2)$. Yet, performance is roughly comparable.

In another study, Chun reports a 76.6% accuracy with a 62-dimensional feature

vector that is very similar to our baseline [2]. In addition to the same derivatives and duration, he averages 12 MFCC's across segment thirds and includes F0, a measure of pitch. Like us, he trains mixtures of full-covariance Gaussians on *train* and tests on *test*.

As a result, we can compare our performance directly with Chun's. In Table 3-10, our full-covariance mixtures produce 76.7% with *SAILS-40* and 76.6% with *SAILS-50*, numbers almost identical to his.

As a final comparison, we consider a set of 36 MFCC measurements optimized with the previous implementation of SAILS. In our phonetic classification framework, these thirty-six dimensions yield 75.8% on *test* using full-covariance mixtures. With the same experimental setup, *SAILS-30* obtains 76.0%, an equivalent performance for slightly less computation.

3.8 Chapter Summary

In this chapter, we conducted experiments with SAILS to discover acoustic measurements sets that improved phonetic classification performance. First, we discussed the classification framework, describing common elements like the speech data, pre-measurement signal processing performed on the data, and the acoustic models of the classifier.

Second, we explained our initial approach to measurement discovery. We hoped to improve classification accuracy by determining pairs of confusable phonetic classes, creating a pool of complex measurements to alleviate this confusion, and generating the N-best measurements to reduce classification error. In this vein, we established a baseline set of cepstral coefficient measurements and determined the major phonetic confusions of the baseline system.

Third, we used the first stage to create a collection of measurements, each of which was optimized to discriminate between a particular pair of confusable classes. We revealed the algorithms applied to each phonetic confusion and the measurement evaluation methods that led to the first-stage collection.

Fourth, we fed this measurement pool to the second stage in an attempt to improve upon baseline performance. Although various second-stage searches through complex measurements were not successful, we learned that reducing the size of the baseline feature vector was in order.

Fifth, we discussed techniques of dimensionality reduction that involved the second stage. Several initial experiments with cepstral coefficient measurements suggested that high performance with low dimensionality was plausible. We modified the second stage and conducted a computationally efficient, generic measurement search to incrementally build cepstral feature vectors, several of which beat baseline performance. We repeated the search, attempting to incorporate complex measurements, but results were not positive. In the next chapter, we offer some possible explanations as to why complex measurements did not improve performance.

Finally, we evaluated selected cepstral feature vectors from the generic measurement search. We noted that smaller feature vectors could beat the baseline on different test sets and with different acoustic models. We also discovered that reducing dimensionality through the generic measurement search resulted in higher performance than conventional reduction methods using principal components. Lastly, we compared the performance of our generic feature vectors to phonetic classification results reported in the literature; we found that our feature vectors offered comparable performance with less computation.

Chapter 4

Conclusion

In this chapter, we first summarize our work. Then, we draw some conclusions from our attempts to improve phonetic classification performance. Specifically, we offer some possible explanations as to why complex measurements did not enhance performance, and we discuss the benefits of our dimensionality reduction research. Finally, we suggest future work that can conducted with SAILS.

4.1 Thesis Summary

We wanted to develop a framework that contains generalized measurement algorithms and allows measurement optimization, and to demonstrate the utility of the framework by creating acoustic measurement sets that improved performance. We achieved both objectives, although in the latter case, through an approach not initially intended.

Our first task was to develop SAILS, a tool that enables us to discover useful sets of acoustic measurements in two stages. In this thesis, we revealed the capability of the SAILS framework: the ability to optimize complex measurement parameters for maximum discrimination between a pair of phonetic classes, and the ability to conduct a performance-based, best-first search through a collection of measurements.

Our second task was to use SAILS to discover measurements that improved baseline performance on phonetic classification. We began with the classical two-stage

approach from the literature, attempting to incorporate complex measurements into the baseline system. When such attempts proved to be less than successful, we shifted the focus of our research to one of dimensionality reduction. In this we succeeded, building several low-dimensional cepstral feature vectors that beat baseline performance.

4.2 Complex Sets of Acoustic Measurements

Our original intent was to use complex sets of acoustic measurements to improve phonetic classification performance. However, as revealed in the previous chapter, we did not succeed in this regard. Appending optimized complex measurements to generic feature vectors is not fruitful when the number cepstral measurements exceeds a fairly low threshold. We offer three possible explanations for this phenomenon.

First and foremost, a feature vector that contains complex measurements might not be well-matched to our Gaussian-based classifier. After all, we optimized complex measurements for pairwise discrimination, while the classifier was trained to make the best 61-phone distinction. Most likely, the classifier prefers a feature vector with universal discriminative power, such as the general data description that MFCC measurements provide. Targeted at a particular phonetic distinction, complex measurements can be quite noisy when applied to other phones. This is certainly evident from our second-stage search through the first-stage measurement pool and from our complex-generic measurement search.

Using a classification and regression tree (CART) [1] for phonetic classification is an alternative that should interact well with our pairwise optimization of complex measurements. CART uses a binary decision at every tree node to classify data; each nodal decision is designed to split data in two parts. In phonetic classification, for example, the decision at the root node might determine whether a phonetic token is sonorant or non-sonorant. Subsequent decisions lead the token to a leaf node that represents a particular phone; the token is then classified as that phone. For each node, we can create a feature vector of complex measurements on which to

base the binary decision. The first stage can optimize each measurement set to best discriminate between the two sets of data that serve as possible choices. Since the manner in which we optimize complex measurements is in line with the manner in which we classify phonetic tokens, complex measurements are likely to succeed in CART-based classification.

A second possible reason why complex measurements did not meet expectations is an overabundance of training data. Since complex measurements target distinctive features, we can train their model parameters with small amounts of training data. Due to their generality, cepstral measurements require larger amounts of such data. With an overabundance of training data, however, cepstral measurements might learn everything that the complex measurements extract, making the feature-extracting measurements redundant.

Finally, the heterogeneity of the feature vectors we used might play against complex measurements. Specifically, we combined spectral, cepstral, and duration measurements in our experiments with complex measurements, and this particular combination might not highlight the discriminative advantages of feature-extracting measurements.

4.3 Dimensionality Reduction

When we shifted the focus of our research to investigate dimensionality reduction with the second stage, we were extremely successful. The performance-based criterion by which the second stage conducts its generic measurement search allows it to surpass the conventional reduction technique of principal components analysis.

The generic measurement search allows us to build feature vectors of standard, cepstral coefficient measurements in an efficient manner. We base this search on the hypothesis that a regular second-stage search will choose single-coefficient cepstral measurements—with the same algorithm and time parameters—in monotonic order, starting with the measurement using the lowest-order cepstral coefficient.

Therefore, if we begin a generic measurement search on N cepstral measurements,

the search will choose the highest performer on each iteration and will replace its selection with a similar measurement of incremented order, returning the search pool to a complement of N measurements. The selection is added to the feature vector, which the search uses in classification trials on the subsequent iteration. As a result, computation is manageable and proportional to the size of the feature vector we wish to produce. In a regular second-stage search, computation is proportional to the size squared. Thus, our intuition about selection order yields significant computational savings.

The generic measurement search is not only efficient but effective. In this thesis, we applied the search to five baseline measurements, discovering that several resulting feature vectors beat baseline performance. We had optimized the baseline by varying the number of cepstral coefficients uniformly across measurements. Our generic feature vectors showed that we could obtain even better performance with fewer dimensions, a satisfying result because the advantages of low dimensionality are clear: run-time recognition speed increases, the estimation of acoustic model parameters improves for a given body of training data, and on a similar note, parameters might require less data to adequately train.

Although we only used five types of measurements in our experiments, the generic measurement search can handle other numbers and types of measurements. For example, we can form a search pool of the baseline five plus averages and derivatives that use different time parameters. A search through a larger combination might produce even higher-performing cepstral feature vectors than those in this thesis. Furthermore, any generalized algorithm that sees more data detail as coefficient order rises can participate in such a generic measurement search. Therefore, non-cepstral algorithms that provide general data description can be used as well.

4.4 Future Work

There are many directions for continued studies in automatic acoustic measurement optimization for segmental speech recognition. We can extend the SAILS framework

to further assist the speech scientist, and we can use the current framework in a variety of experiments.

As far as extending SAILS, four improvements come to mind. First, dreaming up and implementing new generalized algorithms is a must to create more powerful and less noisy complex measurements. Second, viewing the spectral representation of first-stage projected tokens is essential as well. As described in this thesis, we will add the ability to see a speech token in the context of its utterance, superimposing graphics that represent measurement-extracted features. Third, providing probability distributions for each class of projected tokens can be useful. For example, we could divide the projection line into a sequence of intervals and, for each class, count the number of tokens in every interval, plotting the results in a histogram. The resulting distribution can give us an idea of class mean and variance and provide a visual perspective for classification error. Finally, generalizing the second stage to perform other kinds of optimization can be fruitful. With a general framework, we could conduct a best-first search to find optimal window sizes, filter bank frequencies, and the like.

In addition, many opportunities exist for further experimentation. First, we should apply our low-dimensional feature vectors to phonetic recognition. After all, we conducted experiments on the assumption that improvements in classification will translate into improvements in recognition, and we must verify that this assumption holds. Second, we should continue to experiment with the generic measurement search. Specifically, we should create generic feature vectors of greater than sixty dimensions to determine where performance truly begins to decrease. In addition, we should run the generic measurement search through cepstral measurements with various time parameters to determine which parameter values are optimal. Furthermore, we should conduct generic measurement searches in broad phonetic classes, like the vowels, nasals, stops, and fricatives. The cepstral compositions and performances of the resulting broad-class feature vectors might be especially revealing, providing new acoustic-phonetic insight. Third, we should conduct classification trials with varied training set sizes to determine how the quantity of training data affects the generic

measurement search and the two-stage search for complex measurements. Finally, we should consider alternative methods of classification to better incorporate complex measurements. CART [1] or a hierarchical scheme based on phonetic classes [2] should lead to greater success.

Bibliography

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA, 1984.

[2] R. Chun, *A Hierarchical Feature Representation to Phonetic Classification,* M.Eng. thesis, Massachusetts Institute of Technology, Cambridge, MA, March 1996.

[3] R. Duda and P. Hart, *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York, 1973.

[4] W. Goldenthal, *Statistical Trajectory Models for Phonetic Recognition,* Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, September 1994.

[5] M. Hunt, "Delayed decisions in speech recognition—the case of formants", *Pattern Recognition Letters,* 6:121–137, July 1987.

[6] K.-F. Lee, *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System,* Ph.D. thesis, Computer Science Department, Carnegie-Mellon University, Pittsburgh, 1988.

[7] H. Meng, *The Use of Distinctive Features for Automatic Speech Recognition,* S.M. thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1991.

[8] National Institute of Standards and Technology, "TIMIT acoustic-phonetic continuous speech corpus", *NIST Speech Disc 1-1.1,* October 1990.

[9] M. Phillips and V. Zue, "Automatic discovery of acoustic measurements for phonetic classification", in *Proc. Int. Conf. Spoken Language Processing*, pp. 795–798, Banff, October 1992.

[10] G. Strang, *Linear Algebra and Its Applications*. Harcourt Brace Jovanovich, San Diego, third edn., 1988.

[11] V. Zue, J. Glass, M. Phillips, and S. Seneff, "Acoustic segmentation and phonetic classification in the SUMMIT system", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 389–392, Glasgow, May 1989.

[12] V. Zue, S. Seneff, and J. Glass, "Speech database development: TIMIT and beyond", *Speech Communication*, 9(4), 1991.