Laboratory # 8

Vowel Lab II: Vowel Features

8.1 Introduction

In Lab #7, we looked at designing low-order digital filters to model human vowel production. Why is this a useful technique? We argued that such models were useful for studying speech production and the like. However, there are more practical, applied reasons for doing this. Suppose that we want to perform *speech recognition* and determine which vowel is being said. A system would be hard pressed to make this determination from the time-domain waveform. Instead, we need some other description of the signal, some set of *features* that describe important aspects of the signal. Digital filter modeling, such as what we did in Lab #7, is one way of extracting features. In this lab, we will investigate a number of types of vowel features and explore what makes a set of "good" features for classification. In the next lab (Lab #9), we will design and test a classifier for vowels.

8.2 Background

8.2.1 An Introduction to Classification

You may recall Lab #6, in which we developed a system for decoding DTMF signals into the sequence of key-presses that produced the original signal. Our DTMF decoder was actually performing *classification* on each slice of the DTMF signal. Classification is a process in which we examine *instances* or some thing (like an object, a number, or a signal) and try to determine which of a number of groups, or *classes*, that instance belongs to. We can think of this as a labeling process. In our DTMF decoder, for example, we looked at a given segment of the signal and labeled it with a number corresponding to an appropriate key press.

Classification is a two-step process. First, we need some information about the instance that we are considering. This information is traditionally referred to as a set of *features.* So that we can deal with our features easily, we generally like to have a set of features to which we can assign numerical *feature values.* When we have more than one number, we typically place all of the feature values into a *feature vector.* We give a distinct feature vector to each instance we wish to classify by measuring the appropriate aspects of the instance. For our DTMF decoder, our features were the strengths of each DTMF frequency in a given segment of the signal. At each time sample, we calculated a seven-element feature vector, one for each DTMF frequencies.

The second step in classification is to use the feature vectors to make our decision. How we make this decision depends on a number of factors, including the number of classes, the number of features, and the amount of prior knowledge we have about each class. There are some standard decision rules that we can use. For instance, if we have a single feature, we can just use a simple threshold. This is actually the same as a more general rule for

N-dimensional[1] feature vectors. This more general rule indicates that we should find the distances[2] between an instance's feature vector and the *mean feature vector*[3] for each class. Then, we classify based on the smallest of these distances. Sometimes, it is better to come up with an application-specific decision rule. For instance, in the DTMF decoder we had a simple decision rule that simply identified the two maximum features and computed the appropriate key press from these. We will discuss the decision-making step of classification in more detail in Lab #9.

8.2.2 A Classification Example

Let's consider an example of classification. Suppose that we have a large number of flowers of two different types, A and B. We know which flowers belong to each type, but they all look very similar. They require different types of care, though, so we would like to be able to give our gardener a simple means of classifying each flower.

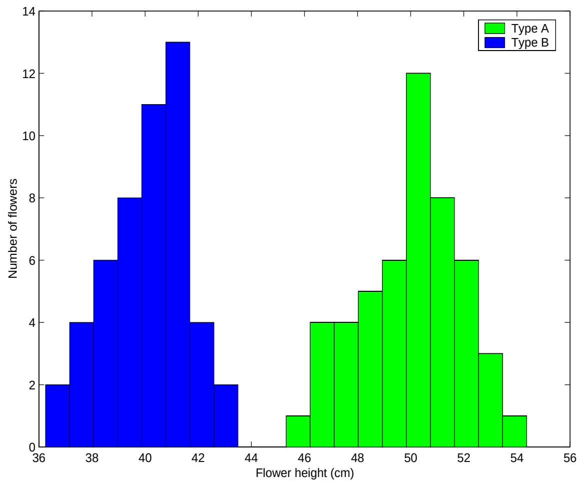

Figure 8.1: A simple example where one feature (flower height) is sufficient to perform classification. This histogram shows how many flowers have a given height.

If we happen to know that type A flowers tend to be taller than type B flowers, we can suggest that our gardener use the flower's height as a feature for classification. We can measure the heights of all of the flowers and then plot a histogram of this data, and we might see something like Figure 8.1. This is an unusually good case. Notice the two *clusters*, one centered around 16 inches and the other around 20 inches. These values are the *mean feature values*. In this case, we determined the mean feature value for one class by simply taking the mean of all of the flower heights that class. Now, our gardener can easily classify the flowers by simply comparing a flower's height to a threshold (18, in this case). Flowers taller than 18 inches belong to type A and flowers shorter than 18 inches belong to type B.

Figure 8.2 (left) shows a histogram of flower heights in a more troublesome scenario. In this case, type A flowers still tend to be taller than type B flowers, but there are a significant number of flowers that our gardener will confuse (that is, misclassify) if we decide exclusively using this one feature. Though we will typically need to deal with some classification error, we can often reduce it by adding more features. Suppose we measure not only the height of

[1]An *N-dimensional vector* is simply a vector that is N elements long.

[3]Though there are a number of useful distance metrics, we will simply use Euclidean distance, which is defined in equation (8.3).

[3]Given a collection of N-dimensional feature vectors from a single class, each feature of the *mean feature vector* is simply equal to the mean value of that feature across the collection. We will examine this in more detail in Section 8.2.5.

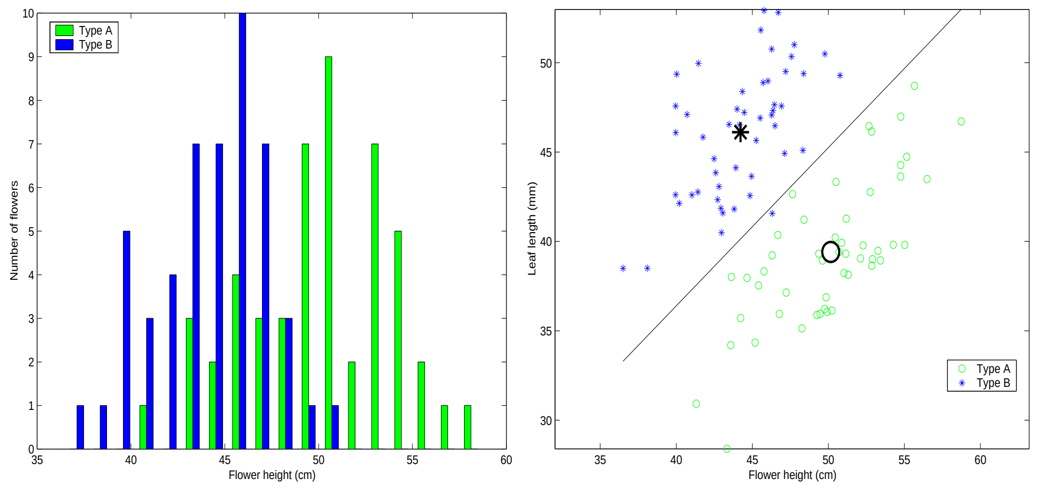

Figure 8.2: An example where a histogram of one feature is *not* sufficient to perform perfect classification (left), but a scatter plot of two features shows a clear separation between the two classes (right).

the flower but also the length of it's leaves. Now we can produce a scatter plot of the two features for each flower, as seen in Figure 8.2 (right). Again there are two distinct clusters, but neither feature is individually sufficient to separate the two clusters[4].

How do we classify in this case? We cannot simply use a threshold on one of the features. Instead, we will use the generalized decision rule discussed above, which is based on mean feature vectors and distances. First, let us consider the two features as a two-dimensional vector. Thus, if a flower is 50 cm tall and has leaves that are 45 mm long, our feature vector is $[50, 45]$. Now, given the feature vectors from each flower of one type, we want to calculate a *mean feature vector* for for flowers of that type. The mean feature vector indicates the central tendency of each feature in a class; that is, it is in some sense the *best representative* of the entire class. To calculate a mean feature vector in this case, we first take the mean, m_1, of all of the flower heights for flowers of one type. Then we take the mean, m_2, of all of the leaf lengths for flowers of the same type. The mean feature vector is then $[m_1, m_2]$. Note that this is the general procedure for calculating the mean of a set of feature vectors, regardless of the number of features. On the scatter plot in Figure 8.2, we've plotted the location of mean feature vector for each class with large symbols.

Before we can classify, we still need to know how to calculate distances between two feature vectors. As we've mentioned we'll just be using the Euclidean distance. The Euclidean distance between two vectors is simply the straight-line distance between their corresponding points on a scatter plot like that in Figure 8.2. To calculate this distance in two dimensions, we simply use the Pythagorean theorem. The straight-line distance is equal to the square root of the sum of the squared differences in feature values. Note that Euclidean distance generalizes to any number of dimensions; the formula can be found later in equation (8.3).

Now we can finally consider the classification of an instance. To perform the classification, we first calculate the distances between the instance's feature vector and the mean feature vectors from each class. Then, we simply classify the instance as a member of the class that has the closest mean feature vector. Consider what this means in terms of the scatter plot. Given a new instance, we can plot it's feature vector on the scatter plot. Then, we classify based on the nearest mean feature vector. For a two-class case such as that shown in Figure 8.2, there exists some set of points that are equally far from both mean feature vectors. These points form a *decision line* that separates the plane into two halves.

[4]That is, if we *project* the scatter plot on to only one of the axes, we can't separate the classes very well. This week's lab section should includes a demonstration of this idea.

We can then classify based on the half of the plane on which a feature vector falls. For example, in Figure 8.2, any flower with a feature vector that falls above the line will be classified as type B. Similarly, any flower with a feature vector that falls below the line will be classified as type A.

With this classification rule, we can correctly classify almost all of the instances. However, note that we're not classifying perfectly. There is one rogue type B close to the rest of the type A's.

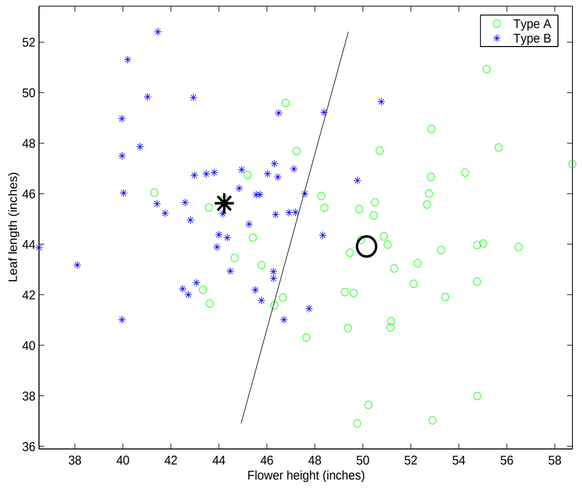

Figure 8.3: An example where two features do not show clear separation.

Of course, two features may not be enough either. If our scatter plot looked like the one in Figure 8.3, then we can still see the two clusters, but we can't perfectly distinguish them based only on these two features. The line we draw for our distance rule will properly classify most of the instances, but many are still classified incorrectly. Once again, we can either accept the errors that will be made or we can try to find another feature to help us better distinguish between the two classes. Unfortunately, visualizing feature spaces with more than two dimensions is rather difficult. However, the intuition we've built for two-dimensional feature spaces extends to higher dimensions. We can calculate mean feature vectors and distances in the roughly the same way regardless of the number of dimensions. In Section 8.2.5, we'll look at some techniques for evaluating higher-dimensional feature spaces.

8.2.3 A few more classification examples

We've looked at a simple classification task with only two classes, but there are some more examples that are instructive. Consider Figure 8.4(A). In this example, the two clusters fall right on top of one another, so we will have very poor classification performance. This is an example where neither of the features assist classification performance very much. In this case, we need to find better features before we can have much luck with classification. Figure 8.4(B) shows a similar example. Here, feature 2 helps us to improve our classification performance but feature 1 does not. Note that it may be worse to have a second feature which is bad than to only have one (good) feature. Unfortunately, determining which features are good and which are bad is difficult when we have more than two (or three) features and can no longer visualize the data.

It is also important to realize that we may have more than just two classes in a classification problem. Figure 8.4(C) shows an example in which we have four classes that have distinct clusters in our feature space. Again, we can use the same distance-based decision metric to classify these clusters. Note the (approximate) decision lines on this plot which

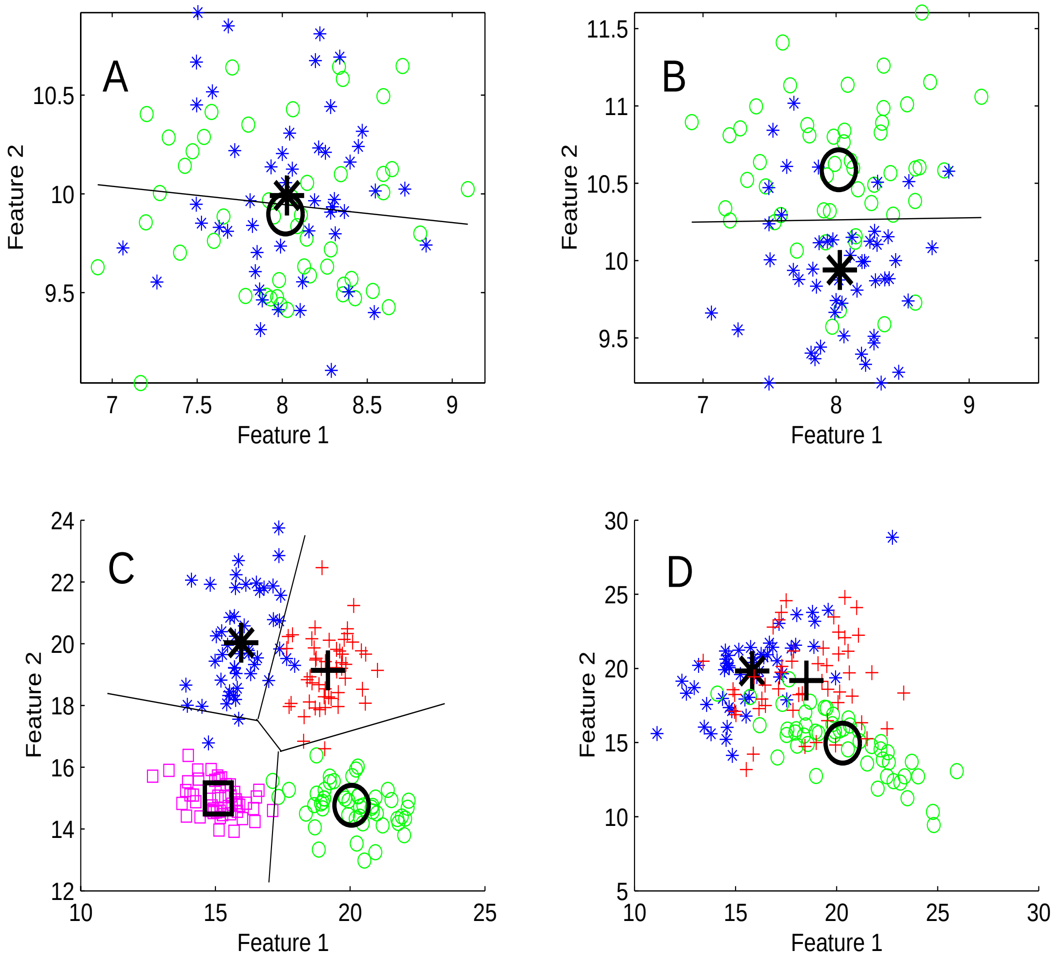

Figure 8.4: (A) Classes overlap, so the features do not allow much discrimination; these are bad features. (B) Feature 2 aids discrimination, but Feature 1 does not. (C) An example with four distinct classes; decision lines are approximate. (D) An example with three indistinct classes.

partition the feature space (i.e., the plane) into four pieces. These indicate which class a given feature vector will be classified as. Of course, multiple classes can be indistinct, too. Figure 8.4(D) shows an example for three indistinct classes. Here, the blue ($*$) and green (o) classes are reasonably distinguishable, but we cannot easily separate the red (+) class from the other two.

8.2.4 Features for Vowels

The classification problem that we are interested in involves the classification of certain types of signals. If you think back to the background section of Lab 7, you may recall that vowels are distinguished by the shape of the vocal tract while the vowel is sounding. We also argued that the vocal tract acted as a filter that shaped the frequency content of some glottal source signal, which was independent of the particular vowel being produced. So, what kinds of features should we use to classify vowels? The obvious answer seems to be "spectral features." It turns out that there are many possible kinds of spectral features, and we've already looked at some of them. A related question asks how we determine which features are good and which ones are not so good. We want features that cause different vowels to be well separated in the feature space and similar vowels to be close to one another. That is, the distance between any two different vowels (say, an "ah" and an "ee") should be large, but the distance between two instances of a single vowel (two different "ah's") should be small.

All-pole modeling and auto-regressive analysis

In the last lab, we explored modeling of the vocal tract filter using simple FIR (all-zero) and IIR (poles and zeros) filters. It turns out that vowel sounds are commonly modeled using all-pole IIR filters that have no zeros. Doing this manually with pole-zero placement is not substantially more difficult than modeling with both poles and zeros, primarily because the acoustics of speech production involve signal reflections that are very well modeled using IIR feedback coefficients.

Another advantage to using all-pole modeling is the fact that we have nice mathematical tools for deriving all-pole models automatically. These tools, which are collectively called *auto-regressive analysis*, fit the spectrum of a time-domain signal with poles in a least-squares sense. Note that these tools work directly with the time-domain waveform rather than its spectrum; typically, they return the resulting IIR feedback coefficients (or *AR coefficients*) for a filter with those poles, rather than the locations of the poles themselves. Figure 8.5 shows a plot of a set of AR coefficients extracted from a speech waveform. We will explore all-pole modeling and auto-regressive analysis in the laboratory assignment.

All of this suggests that we might want to use the poles of our system as our spectral features. While this is a good idea in principle, it has some significant problems in practice. First, we need to establish a meaningful ordering of the poles, which is not trivial on the

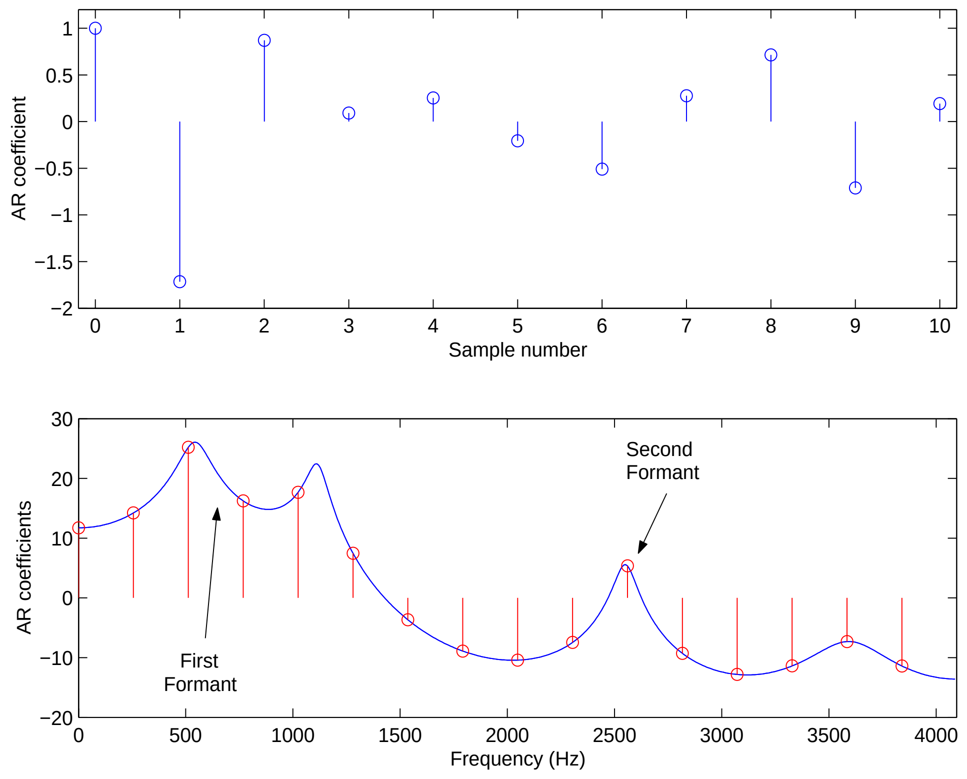

Figure 8.5: Some features for vowel classification. (Top) A set of AR coefficients extracted from a speech waveform. (Bottom) The (sampled) spectrum of the AR model, with formant locations specified.

complex plane. Also, we need to find a useful way of comparing two arrangements of poles so that similar arrangements have a small "distance" and very different arrangements have a large "distance." These problems are difficult to deal with. So what can we use instead? Well, we indicated that auto-regressive analysis produces a set of IIR feedback coefficients. These coefficients are real, already have a meaningful ordering, and we can meaningfully compare arrays of them as feature vectors. How well do these IIR feedback coefficients work as features? You will explore this in the lab assignment.

Other spectral features

There are wide variety of other possible features that we can consider using for classification, but here will consider only two more. One possible alternative is to simply use the frequency response of the vocal tract as a set of features. However, the frequency response is continuous, so we need to sample it. Each sample will be one feature, so the dimensionality of our feature space is related to the number of samples that we take. Finding an appropriate number of samples is not immediately straightforward. If we take too few samples, we may not capture relevant properties of the frequency response. On the other hand, having too many samples may result in "important" frequencies (i.e., good features) being overshadowed by "unimportant" ones (bad features).

One last feature that we will discuss are *formant locations.* Generally, the frequency response of the vocal tract is characterized by a number of peaks at various frequencies. These peaks are called *formants.* It is generally recognized that the frequency of formants is important to the identification of vowels. Unfortunately, there is no clear definition of a formant, and they are surprisingly hard to identify and locate in any sort of consistent, algorithmic way. There are sometimes as many as five formants identified for a speech signal. Here, we will only consider the frequencies of the first two formants.

8.2.5 Evaluating high-dimensional feature spaces

We've already seen how to evaluate a set of features if we have only one, two, or possibly three features. In these cases, we can simply plot a histogram or scatter plot of the features to see how much the classes separate. However, these visualization techniques do not generalize to higher dimensions. What do we do if we want to evaluate a 15-dimensional feature space? There are a number of ways that we can do this. In this lab, we will examine the usefulness of a feature space by looking at the distributions of distances between and within classes. Since we will be using distances to perform classification, this is a useful way to characterize the feature spaces.

In Section 8.2.2, we showed how to compute mean feature vectors and distances for two-dimensional feature vectors. We also argued that the extension to higher dimensions is straightforward. However, here we will provide formal equations for calculating mean vectors and distances between vectors.

Mean vectors and Euclidean distance for multi-dimensional vectors

Suppose that we have a set of N-dimensional feature vectors from M instances of a given class. Let $\mathbf{f_i}$ be the i^{th} feature vector. We calculate the mean feature vector, $\bar{\mathbf{f}}$, for this class as

$$\bar{\mathbf{f}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{f_i} = \frac{1}{M}(\mathbf{f_1}+\mathbf{f_2}+\mathbf{f_3}+\cdots+\mathbf{f_i}). \tag{8.1}$$

Alternatively, if we let $f_{i,j}$ be the j^{th} element of the i^{th} feature vector, we can say that the j^{th} element of the mean feature vector, $\bar{f}_j$, is equal to

$$\bar{f}_j = \frac{1}{M}\sum_{i=1}^{M} f_{i,j} = \frac{1}{M}(f_{1,j}+f_{2,j}+f_{3,j}+\cdots+f_{M,j}). \tag{8.2}$$

We also need to define the distance metric that we will be using. We will be calculating the *Euclidean distance* between two vectors. Let $\mathbf{u}$ and $\mathbf{v}$ be two N-dimensional vectors (i.e., arrays with length N). Also let, v_i be the i^{th} element of $\mathbf{v}$. We calculate the Euclidean distance, d, as

$$d = \sqrt{\sum_{i=1}^{N} (v_i - u_i)^2} = \sqrt{(v_1 - u_1)^2 + (v_2 - u_2)^2 + \cdots + (v_N - u_N)^2}. \tag{8.3}$$

Distance histogram plots

The first method of evaluating a feature space that we will look at involves a plot that we call a *distance histogram plot*. The distance histogram plot is a graph containing several histograms, one for each of our classes. When we generate a distance histogram plot, we have a single *class of interest*. Each of the histograms on the plot shows the distribution of distances from instances in the class of interest to the mean feature vectors for one class. One of these histograms will show the distribution of *intra-class distances*, that is, distances between instances in the class of interest and the mean vector for the class of interest. The remaining histograms show the distribution of *inter-class distances*, that is, distances between instances in the class of interest and a mean vector from some other class. Ideally, we want intra-class distances to be significantly smaller than inter-class distances.

Figure 8.6 shows an example of a distance histogram plot. In this figure, Class 2 is our class of interest. As we would hope, the intra-class distances (labeled as Class 2, the class of interest) are fairly small and cluster near zero. If it does not, we may have a bad set of features. Also as hoped, most of the inter-class distances are larger the intra-class distances. Note that we aren't interested in where the various inter-class distance histograms fall with respect to one another.

The cluster matrix

We can also summarize the distributional information in a form that we call the *cluster matrix*. The cluster matrix measures the mean distance between a class mean and the

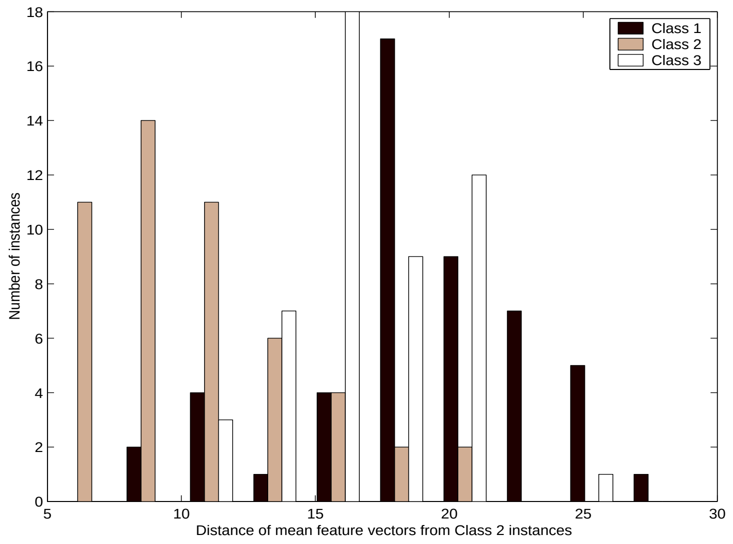

Figure 8.6: An example distance histogram, showing the distribution of distances from the mean feature vector of Class 2.

feature vectors of that class or another class. The size of the cluster matrix is $P \times P$, where P is the number of classes. Let $f_{i,j,k}$ be the k^{th} element of the j^{th} feature vector from the i^{th} class. Let $\bar{f}_{i,k}$ be the mean across all instances of the k^{th} element of the i^{th} class. Finally, let the i^{th} class have M_i feature vectors, and let each feature vector have N dimensions. We calculate the r^{th} row and c^{th} column of the cluster matrix, $C_{r,c}$, as

$$C_{r,c} = \sum_{k=1}^{N} \sum_{j=1}^{M_c} \frac{1}{M_c} \sqrt{(\bar{f}_{r,k} - f_{c,j,k})^2} \tag{8.4}$$

The cluster matrix gives us an indication of both the inter-class separation between the classes (in the off-diagonal elements of the matrix) and the intra-class spread (in the diagonal elements). Thus, we prefer to have a cluster matrix with small diagonal elements and large off-diagonal elements. Note that the cluster matrix is effectively a summary of the information represented in the distance histogram plots for all of classes of interest. Each element of the cluster matrix indicates the center point of one of the histograms on a distance histogram plot. $C_{r,c}$ is the center of the histogram of distances of instances the c^{th} class (the class of interest) from the r^{th} class mean vector.

8.2.6 MATLAB **Command Refresher**

To compute the AR coefficients for a 10^{th} auto-regressive model for a particular waveform, `signal`, use the command

```
AR = aryule(signal,10);
```

To compute the magnitude spectrum, `H`, of a filter defined by feedforward coefficients, `B`, and feedback (AR) coefficients, `A`, use the command

```
H = freqz(B,A);
```

To produce a pole-zero plot for a filter specified by feedforward coefficients `B` and AR coefficients `A`, where `B` and `A` are *row vectors*, use the command

```
zplane(B,A);
```

To produce a scatter plot given a vector of x-values, `X`, and vector of y-values, `Y`, using green o's as the symbol, use the command

```
h = plot(X,Y,'go');
```

Other markers and colors are available; see `help plot` for more information. `h` will contain a *handle* to the data that was just plotted. You can use the `set` command to change the display properties of this data. For instance, you can increase the marker size from 6 points to 12 using the command

```
set(h,'MarkerSize',12);
```

To find the mean vector of a set of vectors, place your vectors in a matrix, `A`, with one vector per row, and use the command

```
mean_vec = mean(A);
```

8.3 Demonstrations in the lab session

1. Classification Introduction
2. Features
3. Vowel features
4. Working with features in MATLAB
5. Evaluating features

8.4 Laboratory assignment

1. In Lab #7, we worked to extract the frequency response of the vocal tract from a signal by dividing the magnitude spectrum of the spectrum of the speech signal by the magnitude spectrum of the glottal source. One common simplification is to assume that the harmonics of the glottal source signal actually have uniform magnitude. Then, the spectral characteristic of both the glottal source and the vocal tract are *lumped* into a single frequency response. For our purposes, this means we can just attempt to match the magnitude spectrum of a signal directly using *Pole Zero Place*.

   Download the file `lab8_ah.mat`. This .mat file contains three variables: `ah_8192` is a recording of someone speaking an "a" vowel, as in the word "father." `gains` contains the magnitude spectrum of the sound at harmonics of the fundamental frequency, and `f0` contains the fundamental frequency itself (105.7 Hz).

   Use *Pole Zero Place*[5] to fit a 10-pole (i.e., 5 conjugate pole pairs) digital filter to the signal `ah_8192`, as you did in Lab #7. As input to `pole_zero_place`, use the vector `gains` and the fundamental frequency in `f0`. Make sure that you use the "decibel" mode when you try to match the vowel's frequency response.

   - [10] Include the resulting GUI window in your report. Make sure that the GUI is in "decibel" mode when you print or copy the GUI window.
   - [2] What are the resulting filter coefficients (i.e., `B` and `A`)?

2. As mentioned section 8.2.4, it is possible to automatically fit poles to a signal's spectrum using a technique called *auto-regressive analysis*. In MATLAB, we use the command `aryule` to perform auto-regressive analysis. The function accepts a signal in the time domain and returns a vector of feedback coefficients for a digital filter (i.e., the `A` vector used in `freqz` or `filter`).

   Use `aryule` to perform 10^{th} order auto-regressive analysis on the signal `ah_8192`. Then use `freqz` to calculate the frequency response of the resulting filter and the filter you calculated in Problem 1.

   - [2] Include the filter coefficients returned by `aryule` in your report.
   - [4] Plot the frequency responses (in decibels) of your hand-fit all-pole model and the AR model produced by `aryule`. Make sure you indicate which curve is which.
   - [3] In two subplots of the same figure, plot the poles of your hand-fit model and the one returned by `aryule` on the complex plane. (Hint: Use the command `zplane` to do this.)

3. Now we would like to begin examining prospective features for vowel classification. The first one we will consider are *formant locations*, which are described in Section 8.2.4. Because we are only considering the locations of the first two formants, this forms a simple two-dimensional feature space that we can easily visualize using scatter plots.

   Download the files `lab8_features.mat` and `extract_formants.m`. `lab8_features.mat` is a file containing five matrices, one for each of five vowels. Each matrix contains AR coefficients calculated from fifty recorded instances of the corresponding vowel. The vowels are "ae" (*a* as in "bait"), "ee" (as in "beet"), "ah" (as in "father"), "oh" (as in "boat"), and "oo" (as in "boot"). The matrices are in a form that we will call a *feature matrix*. That is, a feature matrix has one feature per column and one feature

[5] In order for the "Play Sound" feature to work properly for this lab, you need to comment out the second-to-last line (line 22) in `glottal_source.m`. This is the line containing the command `signal = filter(1,A,signal);`

vector (a set of AR coefficients, in this case) per row. We will consider using the AR coefficients directly as features in another problem.

The function `extract_formants.m` takes a vector of AR coefficients (i.e., one row of a matrix from `lab8_features`) and returns an estimate of the location of the first and second formants in Hz. These two frequencies form the *formant feature vector* for a particular instance. As mentioned in the background section, extracting formants is not easy, so this function doesn't always work. Particularly, this function behaves somewhat unpredictably with the "oh" and "oo" vowels.

(a) Write a function called `calc_formants` that accepts a feature matrix of AR coefficients and returns a feature matrix of formant locations. Use `extract_formants` to compute the formants. Put the first formant in the first column of the output matrix, and the second formant in the second column.

- [6] Include the code for this function in your report.

(b) Now, we want to look at the "ee," "ae," and "ah" classes. Use `calc_formants` to calculate formant feature matrices for these three classes.

- [2] Calculate the mean formant feature vectors for these three classes and include them in your report.
- [8] Create a scatter plot of the formant locations for all three classes, putting the first formant frequency on the x-axis and the second formant frequency on the y-axis. Also, plot the mean feature vector from each class. Make sure that you use unique markers for each of the three classes, that you label the three classes (i.e., using `legend`), and that the mean feature vectors are easy to identify. (Hint: Use black for the mean feature vectors and `set` their marker sizes to 16 or so.)
- [2] From this scatter plot, do you think that these are good features for classification? Why?

(c) Now, we would like to produce distance histograms for our data like those described in section 8.2.5. There are three steps to generating a distance histogram plot. First, you need to calculate the mean feature vectors for each class. Then, you need to calculate the distances between the feature vectors in the class of interest and mean feature vectors. Finally, you need to generate the histograms themselves.

Write a function called `distance_histogram` that accepts three feature matrices and generates a single distance histogram plot. Assume that the first of the three input classes is your class of interest. Use 10 bins when generating the histograms. Also, use `legend` to make sure that it is clear which class is the class of interest. (Hint: If you put your calculated distances into a 50×3 matrix, `A`, you can easily generate the histograms by calling `hist(A,10);`).

- [10] Include the code for your distance histogram-producing function.

(d) Use `distance_histogram` to produce three distance histogram plots with "ee," "ae," and "ah" as your three classes.

- [2] Produce a distance histogram plot with "ee" as your class of interest. Include the plot in your report. (Note: You don't need to print these figures in color. To distinguish the bars on a black and white plot, though use the command `colormap(pink)`.)
- [2] Produce a distance histogram plot with "ae" as your class of interest. Include the plot in your report.
- [2] Produce a distance histogram plot with "ah" as your class of interest. Include the plot in your report.

(e) Note that if had taken the `mean` of the matrices of distances that created to produce the distance histograms, you would have calculated necessary elements of the cluster matrix. To save you some time, though, we've provided you a function (called `cluster_matrix.m`) that computes the complete cluster matrix for you. The function takes one feature matrix for each class that is being considered. Calculate cluster matrix for the formant features. Use "ee" as your first class, "ae" as your second class, and "ah" as your third class.

- [2] Include the cluster matrix in your report.

(f) From the cluster matrix, the scatter plot, and the distance histogram plots that you've produced, you should be able to draw some conclusions.

- [3] For these features, which class is likely to be most misclassified? Which class is likely to be least misclassified? What led you to these conclusions?

4. Now, we'll examine two higher-dimensional feature vectors. The first feature vector is simply the list of 11 AR coefficients from `lab8_features.mat` (the *AR feature vector*).

(a) Calculate the mean AR feature vectors for the "ee," "ae," and "ah" classes.

- [3] Plot the three mean AR feature vectors on the same plot. Make sure you indicate which is which.

(b) Download the file `dist_hist.m`. This function will plot N distance histogram plots when given N feature matrices as inputs. Use it to produce distance histograms for the AR features.

- [2] Include the distance histogram with "ee" as the class of interest.
- [2] Include the distance histogram with "ae" as the class of interest.
- [2] Include the distance histogram with "ah" as the class of interest.

(c) Use `cluster_matrix.m` to calculate the cluster matrix for the AR features. Use "ee" as your first class, "ae" as your second class, and "ah" as your third class.

- [2] Include the cluster matrix in your report.

(d) From the cluster matrix and distance histogram plots that you've produced, you should be able to draw some conclusions.

- [3] For these features, which class is likely to be most misclassified? Which class is likely to be least misclassified? What led you to these conclusions?

5. The second high-dimensional feature vector we will consider contains samples of the spectrum of the AR models in decibels (this is the *spectral feature vector*).

(a) First, we need to calculate the feature matrices for the spectral features. Write a function called `calc_spectral` that takes an AR feature matrix and calculates the spectral feature matrix for the corresponding class. For each instance (i.e., each row) in the the AR feature matrix, you should use `freqz` to calculate a 16-point frequency response vector in decibels. For example, to calculate the spectral feature matrix for the fifth instance in `ae_data`, you would call

```
f = 20*log10(abs(freqz(1,ae_data(5,:),16)));
```

The function should output a new feature matrix of spectral features.

- [6] Include the code for this function in your report.

(b) Use `calc_spectral` to compute spectral feature matrices for the "ee," "ae," and "ah" classes. Compute the mean spectral feature vector for each of these three classes.

- [4] Plot the three mean spectral feature vectors on the same plot. Make sure to indicate which is which.

(c) Use `dist_hist` to produce distance histograms for the AR features.

- [2] Include the distance histogram with “ee” as the class of interest.
- [2] Include the distance histogram with “ae” as the class of interest.
- [2] Include the distance histogram with “ah” as the class of interest.

(d) Use `cluster_matrix.m` to calculate the cluster matrix for the spectral features. Again, use “ee” as your first class, “ae” as your second class, and “ah” as your third class.

- [2] Include the cluster matrix in your report.

(e) From the cluster matrix and distance histogram plots that you’ve produced, you should be able to draw some conclusions.

- [3] For these features, which class is likely to be most misclassified? Which class is likely to be least misclassified? What led you to these conclusions?

6. Now, we can compare the results of the three sets of features that we’ve examined and evaluate their performance.

   - [5] From the cluster matrices and distance histograms that you’ve produced, draw some conclusions about which of the three sets of features is best for distinguishing these three vowel classes. Refer to evidence from both the distance histograms and cluster matrices that support your conclusions.