EE 520 HW3 Question 2

Kevin Mack, Clarkson University 10/04/2017

Problem Statement

In this problem we are attempting to represent samples of different types of music onto a basis set in order to perform classification. It is our task to choose an appropriate basis that will separate the different types of music in space, and to then use linear discriminant analysis (LDA) in order to perform the classification step.

The Data

It is first necessary to read in the music data and get it into a form where we can apply our mathematical techniques. Each data set will consist of short samples of different songs from the same music type. In this work, we select music made from the didgeridoo and classic country music. These choices are made based on the fact that to our own ears, the classification between the two types is easily made; this makes it a good place to start for implementing a classification algorithm in code. The following MATLAB code will read in samples from each data set, using a sample length of $N = 500000$ (which corresponds to roughly 10 seconds). Each song will form a data vector $\mathbf{d}_i$, where i is the number of different songs in the sample. From each data vector a data matrix $\mathbf{D}$ will be formed, where each column corresponds to a different song of that category. The data matrix for country music will henceforth be referred to as $\mathbf{D}_c$, and the didgeridoo music $\mathbf{D}_d$.

Listing 1: Matlab code – Gather data and define matrices.

```
1
2  didgeridoo_files = dir('*.wav');
3  dsounds = length(didgeridoo_files);
4  test_sounds = 3;
5  fs_d = zeros(1, dsounds);
6  for i = 1:dsounds
7      currentfilename = didgeridoo_files(i).name;
8      [currentsound, fs_d(i)] = audioread(currentfilename);
9      didgeridoo_cells{i} = currentsound;
10 end
11
12 country_files = dir('*.WAV');
13 csounds = length(country_files);
14 fs_c = zeros(1, csounds);
15 for i = 1:csounds
```

```
    currentfilename = country_files(i).name;
    [currentsound, fs_c(i)] = audioread(currentfilename);
    country_cells{i} = currentsound;
end

% get all sounds into column matrix
column_length = 500000;%number of samples per son

% form matrix of didgeridoo music
d_matrix = zeros(column_length, dsounds);
for i = 1:dsounds
    s = cell2mat(didgeridoo_cells(i));
    s = s(1:column_length,1);
    d_matrix(:, i) = s';
end

% form matrix of country music
c_matrix = zeros(column_length, csounds);
for i = 1:csounds
    s = cell2mat(country_cells(i));
    s = s(1:column_length,1);
    c_matrix(:, i) = s';
end
```

Picking an Appropriate Basis

It is important to choose a basis that will separate the different types of music in space. In image recognition wavelets are a popular choice for a basis- this is because they are an excellent edge detection method, which is important for recognizing shapes in images. However, in this example wavelets are unlikely to provide a proper basis, as edge detection is not necessarily important in detecting different types of music. It makes intuitive sense that a basis which can detect frequency patterns would be of great use, since music is simply a combination of sinusoids of different frequencies and amplitudes. This leaves us with a few different options, namely the Fourier, Discrete Cosine, and Discrete Sine transforms. In order to simplify the problem (hopefully without losing any accuracy) the Discrete Cosine Transform (DCT) is used to extract the frequency content of the signal. The Fourier Transform, though appropriate in this scenario, would require processing of real and imaginary values which we avoid for the sake of simplicity. There are several forms for the discrete cosine transform (DCT). To check to see if the frequency content between the two classes can be exploited, we can create spectrograms of samples from each class. In Figure 1, it can be observed that the didgeridoo music and country music have different frequency content. This is a good indication that using the DCT is appropriate for this case. In MATLAB, the implementation of the DCT command is known as the DCT-II, which is the most commonly used form. The equation for the transform is given by

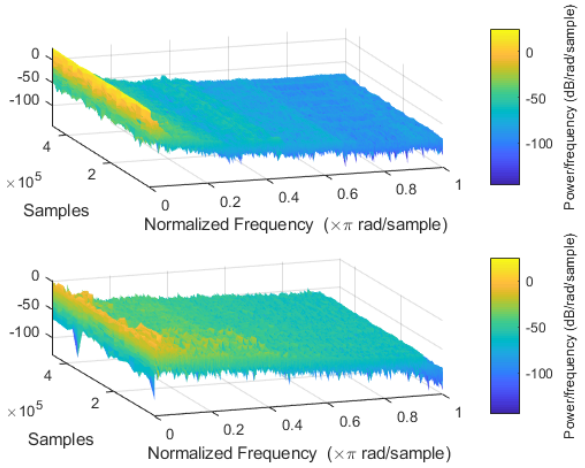

Figure 1: Spectrograms of one sample of music from each class. The figure on top is a sample of didgeridoo music, and the figure on the bottom is a sample of country music. It can be observed that the country music sample has a much wider range of frequency content, while the didgeridoo music is mostly concentrated in the low frequencies. This is an indication that using a Discrete Cosine basis will aid in classification between these two types of music.

Eq.(1):

$$\mathbf{X}_k = \sum_{n=0}^{N-1} x_k cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \quad k = 0, ..., N-1 \tag{1}$$

The DCT is essentially the same as the real component as the Fourier Transform, which is why it was selected for this scenario. The following MATLAB code shows how the DCT is performed on the data matrices $\mathbf{D}_c$ and $\mathbf{D}_d$.

Listing 2: Matlab code – Extract frequency content of music samples

```
% take dct of each signal (use Fourier Basis)
d_matrix_dct = zeros(column_length, dsounds);
for i = 1:dsounds
    d_matrix_dct(:,i) = dct(d_matrix(:,i));
end

c_matrix_dct = zeros(column_length, csounds);
for i = 1:csounds
    c_matrix_dct(:,i) = dct(c_matrix(:,i));
end
```

Training the Sound Data

After choosing the basis set, it is then time to train the data for classification. In this work the method for training will consist of Linear Discriminant Analysis (LDA). The basis set, if chosen

correctly, will cause the two classes of data to separate by their mean. Once the data is separated, the LDA step will be an attempt to find a suitable projection to maximize the distance between the inter-class data while minimizing the distance between the intra-class data. Formally, this two class projection can be written as maximization problem given by Eq.(2).

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{2}$$

where $\mathbf{w}$ is the projection, $\mathbf{S}_B$ and $\mathbf{S}_W$ are the scatter matrices for between-class and within class, respectively. They are given by

$$\mathbf{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{3}$$

$$\mathbf{S}_W = \sum_{j=1}^{2} \sum_{x} (\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^T \tag{4}$$

The criterion given by Eq.(2) is commonly known as the generalized Rayleigh quotient, and whose solution is known to be solved by a generalized eigenvalue problem,

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_w \mathbf{w} \tag{5}$$

where λ and its associated eigenvector give the quantity of interest and the projection basis. This analysis is implemented following the SVD decomposition. The following code is the **sound_trainer.m** file which performed these steps.

Listing 3: Matlab code – Train audio samples

```
function [result, w, U,S,V,th ] = sounds_trainer( sounds1, sounds2, feature )
%Function that receives audio data matrices and uses the SVD to project the
% data into a space where it can be easily classified by a linear
% discriminant
n1 = length(sounds1(1,:)); n2 = length(sounds2(1,:));

[U,S,V] = svd([sounds1, sounds2], 0); %reduced SVD
sounds = S*V';
U = U(:,1:feature);

sounds1 = sounds(1:feature, 1:n1);
sounds2 = sounds(1:feature, n1+1:n1+n2);

figure
subplot(2,1,1)
plot(diag(S), 'ko','Linewidth',[2])
set(gca, 'Fontsize',[14],'Xlim',[0 80])
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',[2])
```

```
set(gca,'Fontsize',[14],'Xlim',[0 80])

[vrow, vcol] = size(V);
figure
basis = 5;
for j=1:basis
    subplot(basis,2,2*j-1)
    plot(1:(vcol/2), abs(V(1:(vcol/2),j)),'ko-')
    subplot(basis,2,2*j)
    plot((vcol/2+1):vcol,abs(V((vcol/2+1):vcol,j)),'ko-')
end

m1 = mean(sounds1, 2);
m2 = mean(sounds2, 2);

Sw =0;
for i = 1:n1
    Sw = Sw+(sounds1(:,i)-m1)*(sounds1(:,i)-m1)';
end
for i = 1:n2
    Sw = Sw+(sounds2(:,i)-m1)*(sounds2(:,i)-m2)';
end

Sb = (m1-m2)*(m1-m2)';

[V2, D] = eig(Sb, Sw);
[lambda,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);

vsounds1 = w'*sounds1; vsounds2 = w'*sounds2;

result = [vsounds1, vsounds2];

if mean(vsounds1) > mean(vsounds2)
    w = -w;
    vsounds1 = -vsounds1;
    vsounds2 = -vsounds2;
end

sortsounds1 = sort(vsounds1);
sortsounds2 = sort(vsounds2);

t1 = length(sortsounds1);
t2 = 1;

while sortsounds1(t1)>sortsounds2(t2)
    t1 = t1-1;
```

```
67      t2 = t2+1;
68  end
69
70  th = sortsounds1(t1)+sortsounds2(t2)/2;
71  bins = 20;
72  figure
73  subplot(1,2,1)
74  histogram(sortsounds1, bins); hold on, plot([th th], [0,10],'r')
75  title('Sounds 1')
76  subplot(1,2,2)
77  histogram(sortsounds2, bins); hold on, plot([th th], [0,10],'r')
78  title('Sounds 2')
```

The function receives as input two data matrices and an integer value for the number of features to be included in the classification scheme. The data matrices are the frequency content (DCT transformed) from the audio data.

Results

The results of training the data are represented by a graph that is output in the **sound_trainer.m** function. Once the LDA solution is found, a histogram of the results can be plotted which details how well the classification scheme worked. There are two histograms which represent the data from each individual class. Also, the discriminant line is plotted on each graph to give an indication of how well the two classes were separated. In this case it appears the basis set was a valid choice. The intra-class samples are clustered very tightly together, but the inter-class samples are clustered sufficiently far from each other. This means that a simple linear discriminant will be effective in classifying these two types of music. The results pictured in Figure 2 are the results from using 10 features for classification and 500000 samples per song.

The results are dependent on two main factors. The first, being the number of samples taken from each song. Since audio samples are taken at high frequency (44kHz) it is important to balance having too many samples for computational efficiency, and too little samples resulting in a insufficient temporal duration. It is important to allow the song a sufficient amount of time to produce its frequency characteristics in order to be able to classify between different types of music.

The second important factor is the number of features used for classification. As can be seen in Figure 2 the principal orthogonal modes extracted from the SVD have varying degrees of energy associated with their principal vectors. The number of principal vectors, which we take as our basis vectors, play a key role in helping to classify the audio samples.

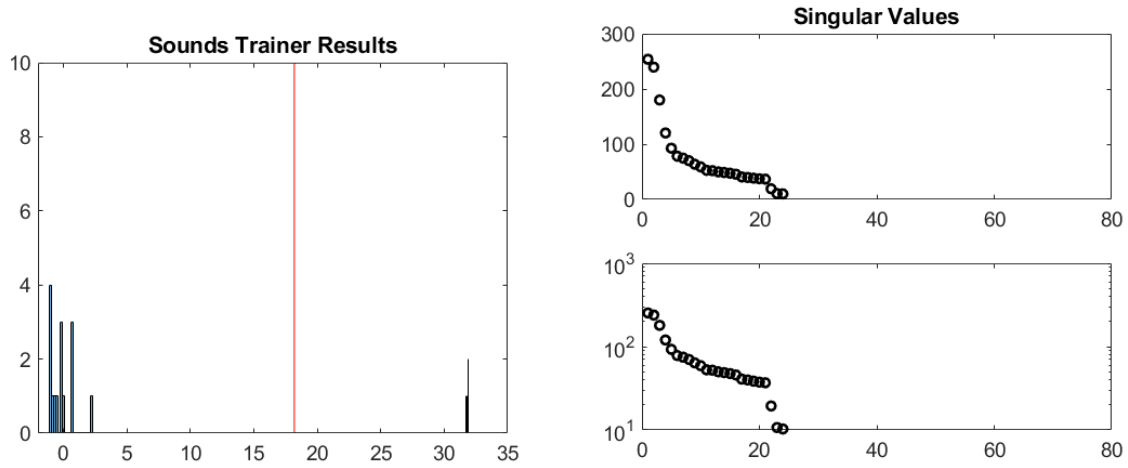

Figure 2: In the histogram on the left, the results from training are displayed for $N = 500000$ and $F = 10$. The DCT basis, along with LDA, has done a sufficient job of clustering the intra-class samples closely together, while separating the inter-class samples. The discriminant line is also displayed, with no samples crossing over for misclassification. The singular values from the SVD are displayed in the right graph, with the log-scale being on the bottom. Due to the complexity of the data, it can be seen that up until roughly the twentieth value, there isn't a large drop off in energy between the different modes.

Finally, It is important to take into account the available computing power and the target computation time in order to balance the accuracy of the algorithm vs. its computational expense. Alternatively, it may be possible to implement a sparse resampling scheme into the algorithm. This would allow the song to progress in time, while keeping the number of samples to a desired (lower) value.