EE 520 Homework 3 Questions 1,3,4

Kevin Mack, Clarkson University 10/6/2017

Question 1: Classifying Images of Cats and Dogs (LDA)

First we must read in images of cats and dogs and arrange them into a data matrix. Cats and Dogs will each have their own data matrices, where each column is one image. The code below will implement this and display a few images of cats and dogs. The images will be displayed in gray-scale, which is also how they will be processed for classification.

Listing 1: Matlab Code – Read in image data and display set of images

```
% EE520 HW3 Q1
load('catData.mat')
load('dogData.mat')

% look at a set of 9 cats
cat_images = zeros(9,64,64);
for i = 1:9
    image = reshape(cat(:,i), [64,64]);
    cat_images(i,:,:) = image;
end

figure
for i = 1:9
    subplot(3,3,i)
    imshow(mat2gray(reshape(cat_images(i,:,:), [64,64])))
end

% look at a set of 9 dogs
dog_images = zeros(9,64,64);
for i = 1:9
    image = reshape(dog(:,i), [64,64]);
    dog_images(i,:,:) = image;
end

figure
for i = 1:9
    subplot(3,3,i)
    imshow(mat2gray(reshape(dog_images(i,:,:), [64,64])))
end
```

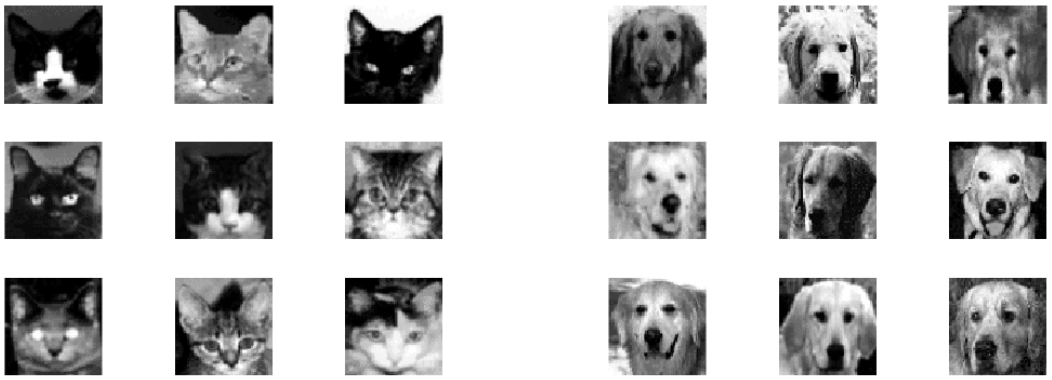

Figure 1: A set of the original images used for training. The images are of low resolution (64x64) but will still be sufficient for classification. This is import because if large images are used, computational complexity vastly increases.

Wavelet Decomposition

In order to classify the images it will be import to first project them onto a basis that can extract key features from the samples. In this case a wavelet basis is appropriate, due to fact that wavelets are excellent for edge detection. Each data matrix will be sent to a function called **dc_wavelet.m**, which will decompose the images into their wavelet basis. The output of this function is a matrix that is 1024xN where N is the number of images in the original data set. The wavelet function reduces each image to a 32x32 representation (from 64x64 original image size). The following code will implement the wavelet decomposition and display key features of the output. The output images are represented in Figure 2.

Listing 2: Matlab Code – Wavelet decomposition code

```
function [ dcData ] = dc_wavelet( dcfile )
%This function receives image data and returns a haar wavelet ↩
    decomposition
%matrix
[m,n] = size(dcfile);
nw = 32*32;
nbcol = size(colormap(gray),1);
%close
dcData = zeros(nw, n);
for i = 1:n
    X = double(reshape(dcfile(:,i), [64,64]));
    [cA,cH,cV,cD] = dwt2(X, 'haar');
    cod_cH1 = wcodemat(cH,nbcol);
    cod_cV1 = wcodemat(cV,nbcol);
    cod_edge = cod_cH1+cod_cV1;
    dcData(:,i) = reshape(cod_edge, [nw, 1]);
end
```

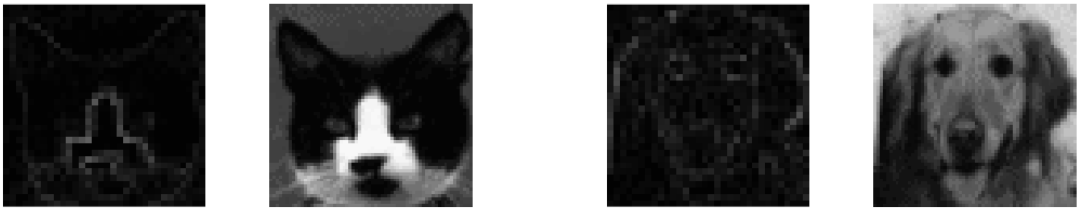

Figure 2: The wavelet decomposition for the cat (on the left) and the dog (on the right) is displayed. The Haar wavelet was effective in detecting the edges in each picture. This indicates that even these low resolution images will be useful in a classification algorithm.

Linear Discriminant Analysis

Following the wavelet decomposition, the data is now ready for Linear Discriminant Analysis. The LDA algorithm is described more in depth in the report on Question 2, but in summary the algorithm will attempt to find a basis set that maximizes the distance between the two different classes in space (while minimizing the distance between intra-class samples). In this analysis, we will make use of the SVD in order to extract the dominant modes of each decomposed image. The following MATLAB code comprises the **catdog_trainer.m** function, which will perform the LDA and output results.

Listing 3: Matlab Code – Training algorithm for cat and dog images

```
function [ sortcats,sortdogs,result,w,U,S,V,th ] = catdog_trainer( cats, ↩
    dogs, feature, plots )
%Linear Discriminant Analysis for training images of cats and dogs
n1 = length(cats(1,:)); n2 = length(dogs(1,:));

[U,S,V] = svd([cats, dogs], 0); %reduced SVD
sounds = S*V';
U = U(:,1:feature);
cats = sounds(1:feature, 1:n1);
dogs = sounds(1:feature, n1+1:n1+n2);

if(plots)
    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])
```

```
18      set(gca,'Fontsize',[14],'Xlim',[0 80])
19  end
20
21  [vrow, vcol] = size(V);
22  if(plots)
23      figure
24      basis = 3;
25      for j=1:basis
26          subplot(basis,2,2*j-1), plot(1:(vcol/2), abs(V(1:(vcol/2),j)),'ko-'
                )
27          subplot(basis,2,2*j), plot((vcol/2+1):vcol,abs(V((vcol/2+1):vcol,j
                )),'ko-')
28      end
29      subplot(3,2,1),title('Cats')
30      subplot(3,2,2),title('Dogs')
31  end
32
33  m1 = mean(cats, 2);
34  m2 = mean(dogs, 2);
35
36  Sw =0;
37  for i = 1:n1
38      Sw = Sw+(cats(:,i)-m1)*(cats(:,i)-m1)';
39  end
40  for i = 1:n2
41      Sw = Sw+(dogs(:,i)-m1)*(dogs(:,i)-m2)';
42  end
43
44  Sb = (m1-m2)*(m1-m2)';
45
46  [V2, D] = eig(Sb, Sw);
47  [lambda,ind] = max(abs(diag(D)));
48  w = V2(:,ind); w = w/norm(w,2);
49
50  vcats = w'*cats; vdogs = w'*dogs;
51
52  result = [vcats, vdogs];
53
54  if mean(vcats) > mean(vdogs)
55      w = -w;
56      vcats = -vcats;
57      vdogs = -vdogs;
58  end
59  % cats < threshold < dogs
60  sortcats = sort(vcats);
61  sortdogs = sort(vdogs);
62
```

```
t1 = length(sortcats);
t2 = 1;

while sortcats(t1)>sortdogs(t2)
    t1 = t1-1;
    t2 = t2+1;
end
th = sortcats(t1)+sortdogs(t2)/2;
if(plots)
    bins = 30;
    figure
    subplot(1,2,1)
    histogram(sortcats, bins); hold on, plot([th th], [0,10],'r')
    %set(gca,'Xlim',[-1.55 -1.35],'Ylim',[0 10],'Fontsize',[14])
    title('Cats')
    subplot(1,2,2)
    histogram(sortdogs, bins); hold on, plot([th th], [0,10],'r')
    %set(gca,'Xlim',[31.595 31.5955],'Ylim',[0 10],'Fontsize',[14])
    title('Dogs')
end
```

The results from the LDA are represented in Figures 3 and 4 which show the energy in each principal mode, the first three principal modes, and the overall results respectively. Also, figure 5 shows the classification error when the number of principal modes (features) for classification is increased from 3 to 80. As can be seen from Figures 4 and 5 the algorithm performs fairly well, and increases in accuracy as more features are used in the classification. Even though up to 80 features have been used, the LDA step saves this analysis from what is commonly referred to as ,"the curse of dimensionality"; this means that LDA reduces the number of dimensions in the projection of the data in such a way that maximizes the separation of the classes.

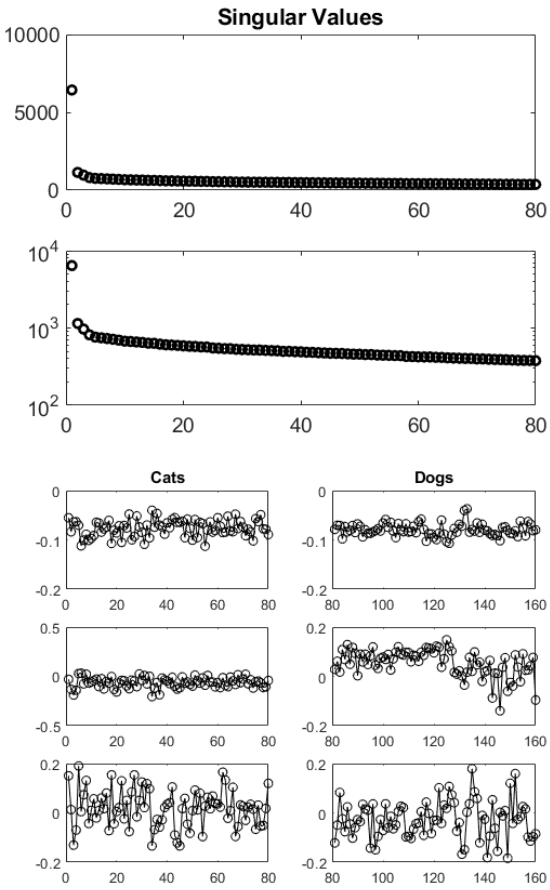

Figure 3: The graphs depict the energy in the modes (left) as well as the first three modes themselves (right). In the first three modes, it can be seen that there are differences in the values between the cat and dog images. It can also be seen that the first mode has significantly more energy than all of the other modes.

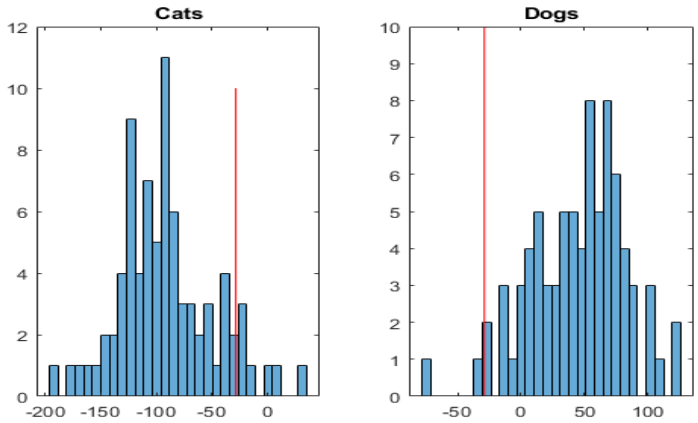

Figure 4: The histograms display the clustering of the data as well as the threshold for classification. It can be seen that some samples from each class fall outside the threshold. In order to mitigate this more advanced techniques may be performed, or more sample images may be used to train the algorithm.

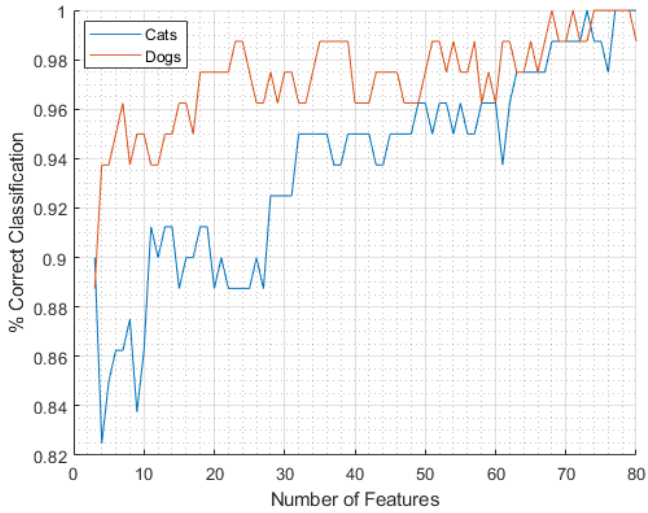

Figure 5: This graph depicts the accuracy of the classification algorithm as the number of features used is increased. In this case there are only a small set of images tested, and the accuracy values fluctuate slightly. It can be seen that after roughly forty features there isn't much benefit to the added computation. In practice, the trade-off between computational complexity and accuracy would be used to determine the number of features used.

Question 2: Classifying Types of Music

See attached report.

Question 3: Classifying Images of Cats and Dogs (k-Means)

In this question we are tasked with attempting a different classification algorithm than the LDA from Question's 1 and 2. In this case the k-means classification algorithm will be used on the cats and dogs data from Question 1. The details of reading in and preparing the data for classification are provided in the first section, and will be skipped here. In this section the SVD is used in order to reduce the number of dimensions for the dataset. Three pricinpal orthogonal modes were used to represent the data in space in order to perform the k-means clustering algorithm. A sub-sample of the clusters and their respective means is depicted in Figure 6. In this case the correct percentage of classifications for the images were 0.87 for dogs, 0.95 for cats.

Question 4:

Part A:

Question: What if you needed to make a partition of $k > 2$ elements?

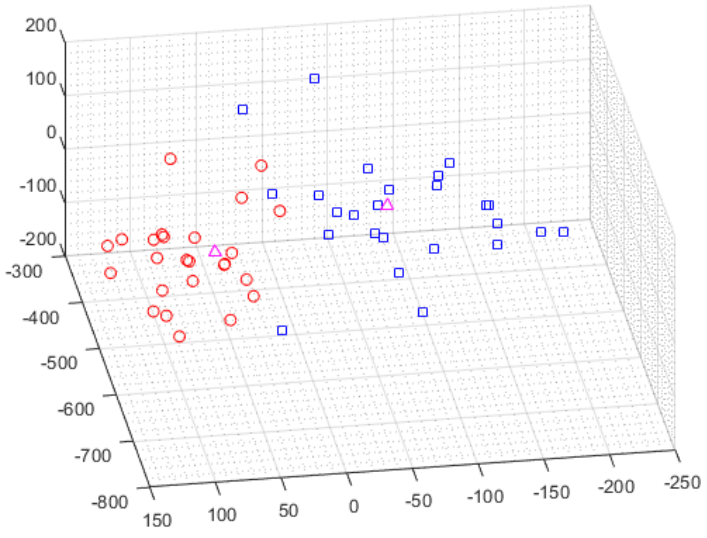

Figure 6: Three dimensional graph displaying the clusters of data. The circles represent the cat image data, while the squares represent the dog image data. The means are represented by triangles. It can be seen that using three dimensions works well in this case, and there is no need to include higher dimensions (which may actually reduce the accuracy of classification).

If more classes of images are used for a classification then the algorithms used (LDA and k-means) will need to be utilized in a more general form. In the case of LDA, it may be of interest to partition the data in order to classify between one class and all others. In addition to this "one against the rest" classification, it is possible to do pairwise classification of the samples. For example, if there are classes A, B, and C then you would have classify between each possible pair of classes. Additionally, it is possible to classify between m number of classes in LDA. This would require computing more inter and intra-class scatter matrices, as well as the statistics for each class, in order to perform dimensionality reduction and classification.
In the context of this homework, if we added images of giraffe faces to the mix of data we could attempt to pick out the images that are of a certain class (cats for example). If we do this for each class, then we can apply each individual linear discriminant to be able to determine the class of a given image.

Part B:

Question: What if you want the method itself to suggest a good number of partition elements?

In the case of LDA, the number of classes must be known before the algorithm is applied. This is because LDA is part of the supervised learning class of algorithms. There are other types, unsupervised learning methods, which will achieve their results without the apriori knowledge of the number of classes. A few unsupervised methods include k-means, mixture models, neural networks, and hierarchical clustering.