CATS AND DOGS : WHAT'S THE DIFFERENCE?

AUSTIN ADAMS AND AMY MULGREW

1. Introduction

How can a computer distinguish between two different objects? For people it is easy to say a picture is either a cat or a dog, but the computer is not quite as visually capable. In order to classify a test picture as a dog or a cat, the computer needs to be trained to do so. Four methods are discussed: Principal Angles, PCA and Principal Angles, Fisher Discriminant Analysis, and Wavelet Edge Finding. These are organized into a description, results of testing against itself, and result of testing on unknown 38 images.

2. Principal Angles Method

Principal Angles is a method by which to find the smallest angle between two subspaces. The smaller the angle between subspaces the closer the subspaces are considered to be.

2.1. Method and Classifier.

2.1. Method and Classifier. The Principal Angles Method is as follows:

(1) Find orthonormal bases for input matrices X and Y labeled Q_X and Q_Y.
(2) Find svd of cosine: perform svd on $Q_X^T Q_Y$, singular values list as $(\sigma_1, \sigma_2, \cdots, \sigma_n)$
(3) $Y = \begin{cases} Q_Y - Q_X(Q_X^T Q_Y) & \text{if } rank(Q_X) < rank(Q_Y) \\ Q_X - Q_Y(Q_Y^T Q_X) & else \end{cases}$
(4) Find svd of sine: perform svd on Y, singular values list as $(\mu_1, \mu_2, \cdots, \mu_m)$
(5) k^{th} angle is given by $\theta_k = \begin{cases} \arccos(\sigma_k) & \text{if } \sigma_k^2 < 0.5 \\ \arcsin(\mu_k) & \text{if } \mu_k^2 \leq 0.5 \end{cases}$ for k from 1 to $min(rank(Q_X, Q_Y))$

This was taken from [1].

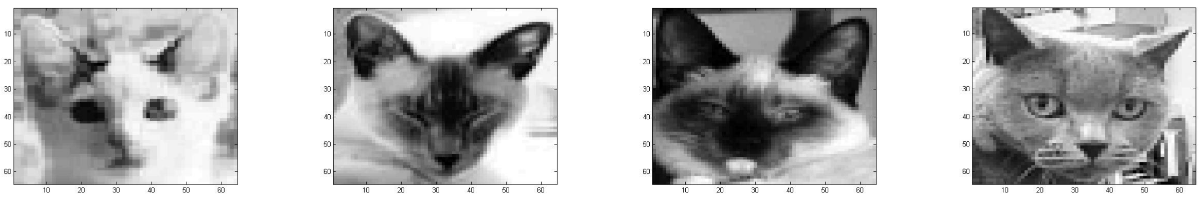
To use principal angles as a classifier, take a test picture and run it against both the training dog set and the training cat set. Whichever set has the smaller angle is what to classify the test picture as. See Appendix B and C for the specific codes used.

2.2. Classification Errors of Principal Angles Method. Table 1 shows the results of the Principal Angles Method on its own training data. Figures 1 and 2 show the cats and dogs classified wrongly by this method. See Appendix D for how the confusion matrix was constructed.

Date: May 20, 2010.

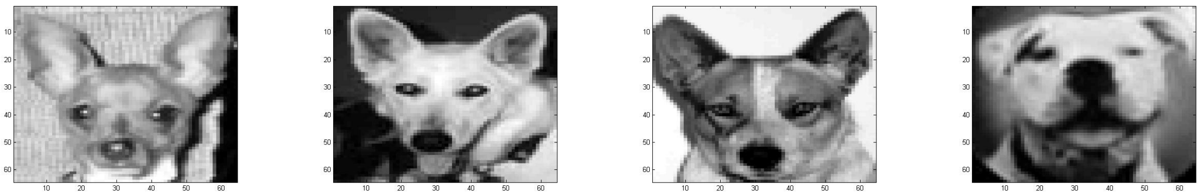
	Classified as cat	Classified as dog
Actually a cat	76	4
Actually a dog	5	75

TABLE 1. Confusion Matrix for Principal Angles Method

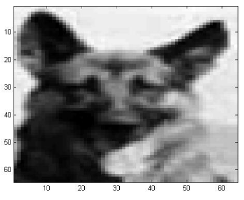


(a) (b) (c) (d)

FIGURE 1. These four pictures are the cats that were mislabeled by the Principal Angles Method



(a) (b) (c) (d)



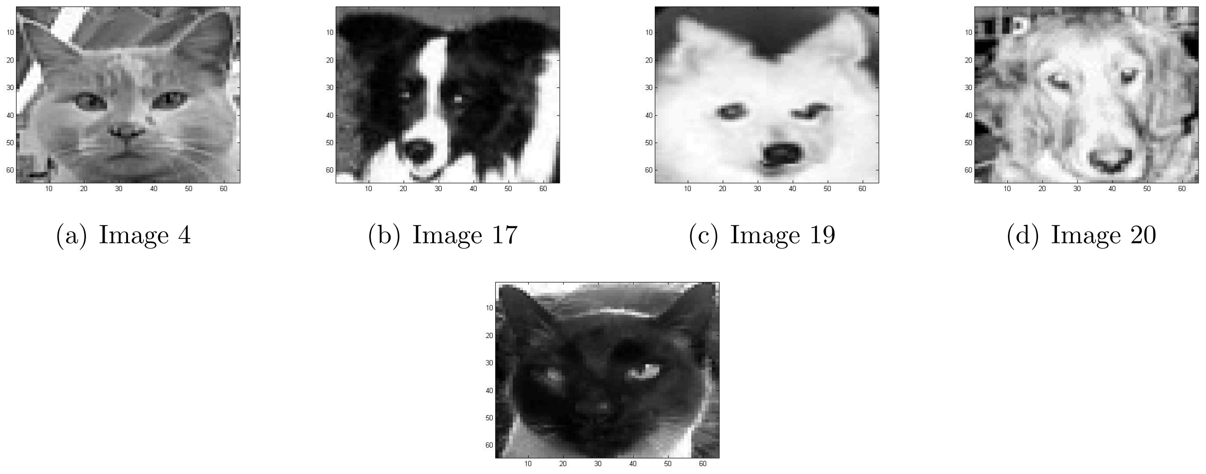
(e)

FIGURE 2. These five pictures are the dogs that were mislabeled by the Principal Angles Method

2.3. **Predicted Class Membership for Unknowns.** Dogs are listed as a 0 and cats are listed as a 1. See Table 2 for all of the labels and their corresponding actual values. There are five mistakes in labeling, for a total of $\frac{33}{38}$ or around 87% accuracy. The specific animals labeled incorrectly were [4 17 19 20 34] as seen in Figure 3.

Image number	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
Labeled	0	1	0	0	1	1	1	1	0	1	0	0	1	1	0	0	1	0	1
Actual	0	1	0	1	1	1	1	1	0	1	0	0	1	1	0	0	0	0	0
Image number	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Labeled	1	1	0	1	0	0	0	1	1	0	1	0	0	1	0	1	1	0	1
Actual	0	1	0	1	0	0	0	1	1	0	1	0	0	1	1	1	1	0	1

Table 2. Class Membership as Predicted by Principal Angles



(a) Image 4 (b) Image 17 (c) Image 19 (d) Image 20

(e) Image 34

Figure 3. These are the test images incorrectly labeled by the Principal Angles Method.

3. PCA/Principal Angles

3.1. **Method and Classifier.** PCA/PA Matlab pseudo code:

(1) Load the gallery files which consist of 80 cats and 80 dogs.
(2) Perform Principal Component Analysis (PCA) dimensionality reduction on the entire gallery and retain 99% of the energy.
   (a) Calculate the Ensemble Average of the gallery.
   (b) Mean Subtract the gallery.

   (c) Find the economic Singular Value Decomposition (SVD) of the mean subtracted gallery.
   (d) Calculate the value of D that retains 99% of the energy. (turns out to be $D = 231$)
   (e) Reduce the dimension of the KL basis and the coefficients of the gallery using D.
   (f) Return the reduced KL basis, the Ensemble Average and the reduced coefficients of the gallery.
   (g) Save these results to use in the classification, so that the classification avoids the computationally expensive task of SVD.
(3) Mean subtract the gallery's Ensemble Average from the probe data (a 4096×38 matrix of the missing cats and dogs).
(4) Apply the reduced KL basis to the mean subtracted probe data to obtain the reduced coefficients of the probe in the KL basis.
(5) Beginning with the first animal in the probe, create θ_1, the angle between the probe animal and the subspace of the cats from the gallery, and θ_2, the angle between the probe animal and the subspace of the dogs from the gallery.
(6) If $\theta_1 < \theta_2$, then classify the probe animal as a cat, since it most closely resembles the cats. Otherwise, classify the probe animal as a dog.

In the previous section the images of cats and dogs are classified using the Principal Angle Method without prior manipulation of the pictures. Since the images have 64×64 pixels, the matrices we use are large. Not all of the information that the matrices hold is essential to the classification. Thus, to expediate computation and reduce extraneous data transfer, we use the method of Principal Component Analysis (PCA) to reduce the dimensions of the matrices.

A key element to PCA is the extraction of eigenvalues and eigenvectors of a covariance matrix. We begin with a covariance matrix of size 4096×160, since we have 160 images in the gallery of cats and dogs, each with 64×64 (=4096) pixels. While still maintaining 99% of the energy (important information) of the images, we can reduce the matrix to 231×160. See reference [1]. PCA produces the set of eigenvectors sorted according to the magnitude of their corresponding eigenvalues. The first 12 eigenvectors, or eigenanimals in this case, are displayed in Figure 4. The first two eigenanimals emphasize the location and the roundness of the face, which is the most prominent quality shared by all the images. The location of the facial components seem emphasized in eigenanimals 3, 5, 6, 9 and 10.

The results improve slightly by reducing the dimension with PCA and then applying the Principal Angle method from the previous section as a classifier. See Appendix H for the specific codes used.

3.2. **Classification Errors of the Modified Principal Angles Method.**

Table 3 shows the results of the Modified Principal Angles Method on its own training data. Note that according to the file none of the training set was mislabeled.

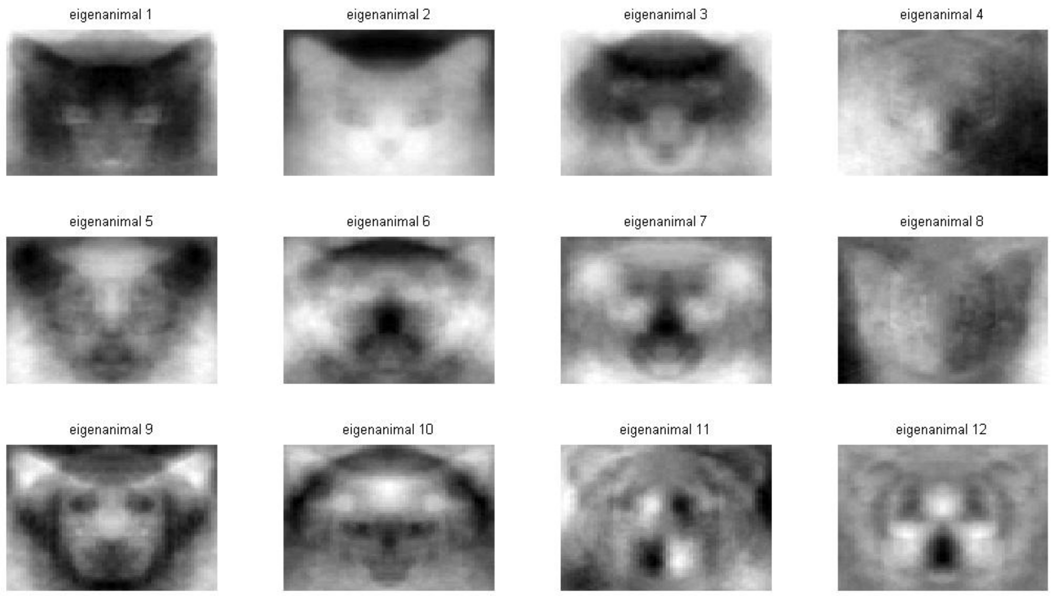


FIGURE 4. The first 12 eigenanimals displaying the top 12 characteristics found in the images.

	Classified as Cat	Classified as Dog
Actually a Cat	80	0
Actually a Dog	0	80

TABLE 3. Confusion matrix for Modified Principal Angles Method

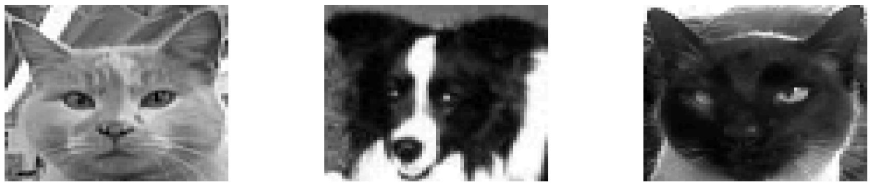
3.3. **Predicted Class Membership for Unknowns.** Dogs are listed as a 0 and cats are listed as a 1. See table 4 for all of the labels and their corresponding actual values. There are three mistakes in labeling, for a total of $\frac{35}{38}$ or around 92% accuracy. The specific animals labeled incorrectly were [4 17 34] as seen in Figure 5.

4. PCA/FDA

Fisher's linear discriminant analysis (FDA) is a classification method that finds an optimal projection to one dimensional space and projects all the data to a real line. The goal is to separate the training data completely on the projection so that we can pick a threshold value

Image Number	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
Labeled	0	1	0	**0**	1	1	1	1	0	1	0	0	1	1	0	0	**1**	0	0
Actual	0	1	0	**1**	1	1	1	1	0	1	0	0	1	1	0	0	**0**	0	0
Image Number	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Labeled	0	1	0	1	0	0	0	1	1	0	1	0	0	1	**0**	1	1	0	1
Actual	0	1	0	1	0	0	0	1	1	0	1	0	0	1	**1**	1	1	0	1

TABLE 4. Class Membership as predicted by Modified Principal Angles



(a) Image 4 (b) Image 17 (c) Image 34

FIGURE 5. These three images were mislabeled by the Modified Principal Angles Method

which separates the cats and the dogs. Once we project the unknown data to the real line, if it is on the dog side of the threshold, it is classified as a dog, otherwise it is said to be a cat.

4.1. **Method and Classifier.** PCA/FDA Matlab pseudo code:

(1) Load the gallery files which consist of 80 cats and 80 dogs.
(2) Perform Principal Component Analysis (PCA) dimensionality reduction on the entire gallery and retain 99% of the energy as decribed previously.
(3) Mean subtract the gallery's Ensemble Average from the probe data (a 4096×38 matrix of the missing cats and dogs).
(4) Apply the reduced KL basis to the mean subtracted probe data to obtain the reduced coefficients of the probe in the KL basis.
(5) Perform Fisher's linear discriminant analysis (FDA) for two classes to find the optimal projection ω for the linear separation of the two classes: cats and dogs.
(6) Project the coefficients of the gallery of cats, gallery of dogs and probe onto ω. Now all the data lies in one dimension.
(7) Calculate the mean of the cat gallery and the mean of the dog gallery. If the mean of the cat gallery is less than the mean of the dog gallery, flip every element projected onto ω about 0 so that the order of the gallery data will be dogs $<$ threshold $<$ cats. Note that the threshold value is the mean of the largest dog value and the smallest cat value.

(8) Classify the elements of the probe by comparing each value with the threshold. If a probe element is less than the threshold, it is classified as a dog. If a probe element is greater than the threshold, it is classified as a cat.

4.2. **Classification Errors on FDA method.** Table 5 shows the results of the FDA method on its own training data. Note how poorly it does against its own training data. Figures 6 and 7 show some of the cats and dogs wrongly classified by this method.

	Classified as Cat	Classified as Dog
Actually a Cat	72	8
Actually a Dog	18	62

TABLE 5. Confusion matrix for two class FDA.

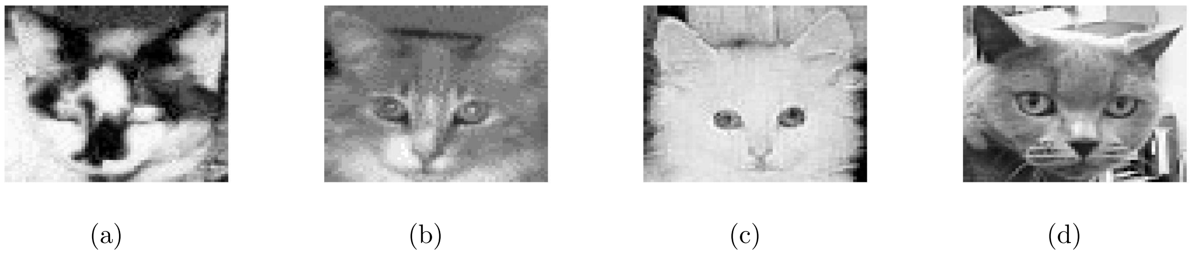


FIGURE 6. These four pictures are some of the cats that were mislabeled by the FDA Method

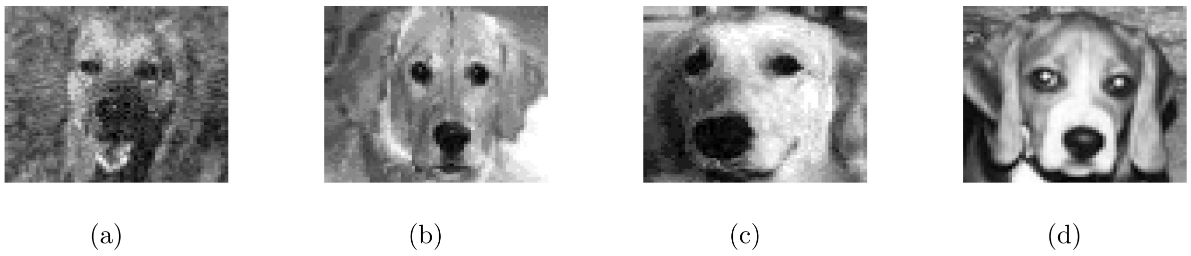


FIGURE 7. These four pictures are some of the dogs that were mislabeled by the FDA Method

4.3. **Predicted Class Membership for Unknowns.** Dogs are again listed as a 0 and cats are listed as a 1. See table 6 for a list of all the labels and what the actual label is. Note that despite the errors in the confusion matrix above there are only a few errors against the unknown images. There are three mistakes in labeling, for a total of $\frac{35}{38}$ or around 92% accuracy. The specific animals labeled incorrectly were [17 18 23] seen in Figure 9. Figure 8 shows the result of projecting the gallery and the unknown images to the real line.

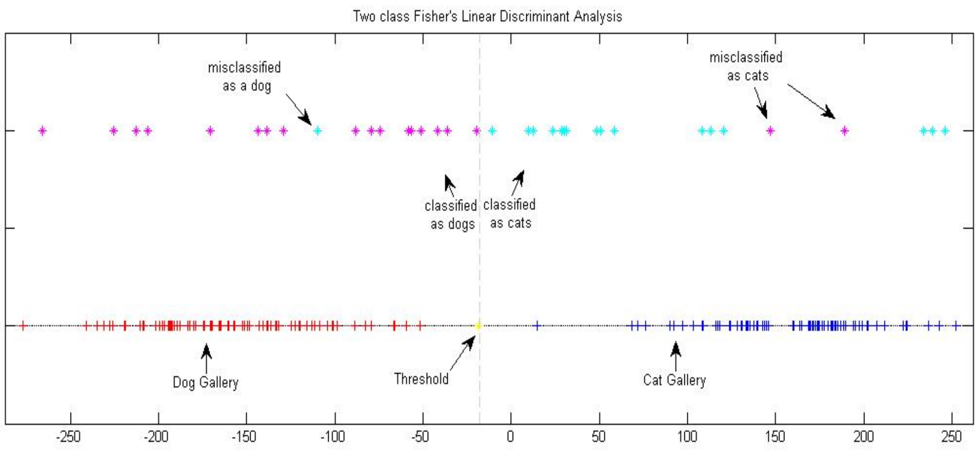


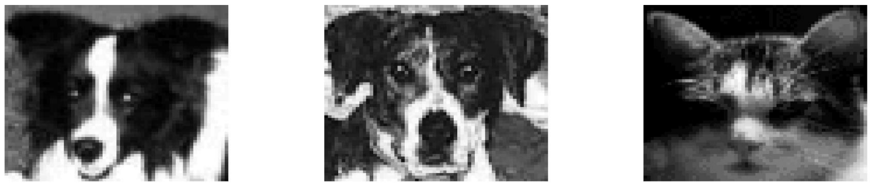
FIGURE 8. Two class FDA. Dogs from the probe are displayed in magenta, while cats from the probe are displayed in cyan.

Image Number	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
Labeled	0	1	0	1	1	1	1	1	0	1	0	0	1	1	0	0	**1**	**1**	0
Actual	0	1	0	1	1	1	1	1	0	1	0	0	1	1	0	0	**0**	**0**	0
Image Number	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Labeled	0	1	0	**0**	0	0	0	1	1	0	1	0	0	1	1	1	1	0	1
Actual	0	1	0	**1**	0	0	0	1	1	0	1	0	0	1	1	1	1	0	1

TABLE 6. Class Membership as predicted by FDA.

5. Wavelet Edge Detection

Wavelets were used as a way to extract the edges from each image. The result was then put through the Principal Angles Method as detailed above.



(a) Image 17 (b) Image 18 (c) Image 23

Figure 9. These three pictures are the images that were mislabeled by the FDA Method

5.1. **Method and Classifier.** The method used is as follows:

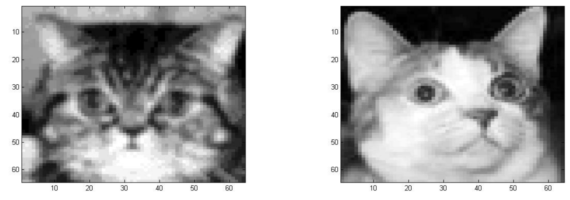
(1) Perform one level discrete wavelet transform on each image.
(2) Add the horizontal and vertical detail components, and call this your new image.
(3) Take all the new images through the Principal Angles Method.

This method is just a modification of the Principal Angles Method, but it works much better.

5.2. **Classification Errors of Wavelet/Principal Angles Method.** Table 7 shows the results of the Wavelet Edge method on its own training data. Figures 10 and 11 show the cats and dogs classified wrongly by this method. See Appendix G for how the confusion matrix was constructed.

	Classified as cat	Classified as dog
Actually a cat	78	2
Actually a dog	6	74

Table 7. Confusion Matrix for Wavelet Edge Detection Method



(a) (b)

Figure 10. These two pictures are the cats that were mislabeled by the Wavelet Edge Detection Method.

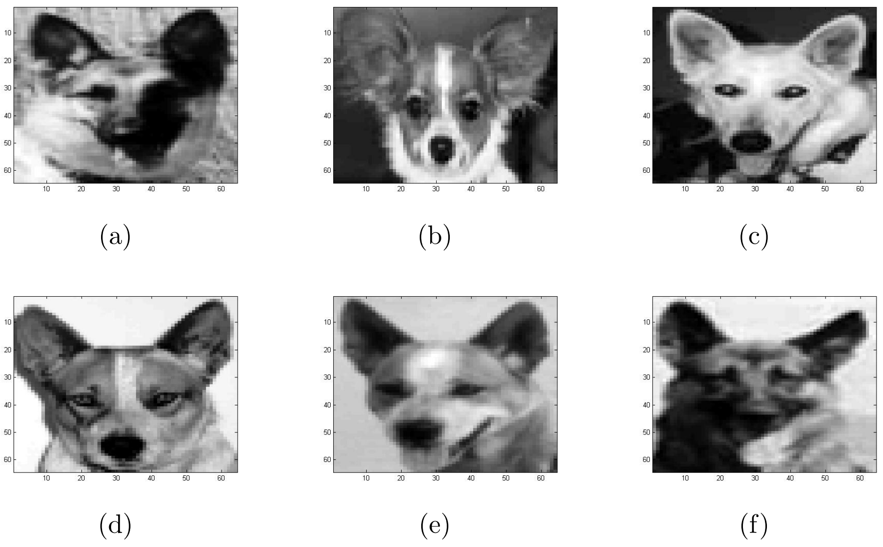


Figure 11. These six pictures are the dogs that were mislabeled by the Wavelet Edge Method

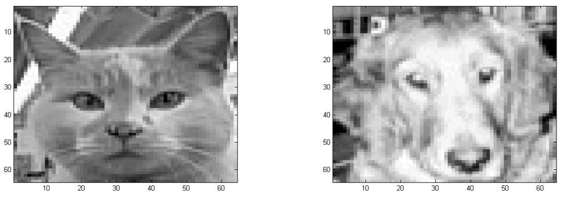
5.3. **Predicted Class Membership for Unknowns.** Again dogs are listed as a 0 and cats are listed as a 1. See Table 8 for a list of all the labels and what the actual label is. There are only two mistakes in labeling, for a total of $\frac{36}{38}$ or around 95% accuracy. The specific animals labeled incorrectly were [4 19] seen in Figure 12.

Image number	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
Labeled	0	1	0	0	1	1	1	1	0	1	0	0	1	1	0	0	0	0	1
Actual	0	1	0	1	1	1	1	1	0	1	0	0	1	1	0	0	0	0	0
Image number	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Labeled	0	1	0	1	0	0	0	1	1	0	1	0	0	1	1	1	1	0	1
Actual	0	1	0	1	0	0	0	1	1	0	1	0	0	1	1	1	1	0	1

Table 8. Class Membership as Predicted by Principal Angles

References

[1] Jen-Mei Chang. *MATRIX METHODS FOR GEOMETRIC DATA ANALYSIS AND PATTERN RECOGNITION*, 2009.



(a) Image 4 (b) Image 19

Figure 12. These two pictures are the test probes that were mislabeled by the Wavelets

Appendix A. Main script 1

```
%% MATH 695 Final Project
% Written by Austin Adams

%Two main methods of classifying images will be explored
%Cats == 1    %Dogs == 0

load animals
load PatternRecAns
Cats=animals(:,1:80);
Dogs=animals(:,81:160);
%% Principal Angles Method

%[CM_PAM,PAMcatswrong,PAMdogswrong]=PAM_confmat(Cats,Dogs);

%labels_PAM=class_principal(Cats,Dogs,TestSet);

%ResultsPAM=abs(labels_PAM-hiddenlabels);

%% Wavelet Edge Detection Method

[CM_WAV,WAVcatswrong,WAVdogswrong]=WAV_confmat(Cats,Dogs);

labels_WAV=class_wave(Cats,Dogs,TestSet);

ResultsWAV=abs(labels_WAV-hiddenlabels)
```

Appendix B. Principal Angles Method

```
%This code computes small principal angles between two subspaces
%Written by Austin Adams
%Input:X,Y
%   X=n-by-p matrix
%   Y=n-by-q matrix

%Output:Anglesk
%   Anglesk=principal angles between subspaces R(X)=X and R(Y)=Y

function [Anglesk]=prinangles(X,Y)
Qx=orth(X); %Orthogonal bases
Qy=orth(Y);

Co=svd((Qx')*Qy,0);  %Cosine values

rX=rank(X);
rY=rank(Y);
if rX >= rY
    YNew = Qy - Qx*(Qx'*Qy);
else
    YNew = Qx - Qy*(Qy'*Qx);
end;
Si=svd(YNew,0);  %Sine values
Si=sort(Si);  %This sort makes the whole thing work
            %Since Co is in descending order, Si needs to be in ascending
            %order

for i=1:min(rX,rY)
    if (Co(i))^2 < 0.5
        Anglesk(i) = acos(Co(i));
    elseif (Si(i))^2 <= 0.5
        Anglesk(i) = asin(Si(i));
    end
end
```

Appendix C. Principal Angles Classifier

```
%%This code classifies cats and dogs using principal angles
%Written by Austin Adams
```

```
%Input:Cats,Dogs,Probes
%   Cats=gallery of cats(each image is 4096 by 1)
%   Dogs=gallery of dogs(ditto)
%   Probes=test images(ditto)

%Output:Class
%   Class=labels(1==cat 0==dog) in vector form

function [labels]=class_principal(Cats,Dogs,Probes)
[mp,np]=size(Probes);
labels=1:np;
for i=1:np
    Pvscat=prinangles(Probes(:,i),Cats);
    Pvsdog=prinangles(Probes(:,i),Dogs);
    if Pvscat < Pvsdog
        labels(i)=1;
    elseif Pvsdog < Pvscat
        labels(i)=0;
    end
end
```

Appendix D. Confusion Matrix for Principal Angles

```
%%This code evaluates the confusion matrix for principle angles method
%Written by Austin Adams

%Input:Cats,Dogs
%   Cats=first class
%   Dogs=second class
%Note Cats and Dogs must be same size

%Output:CM
%   CM=confusion matrix

function [CM,wrongCats,wrongDogs]=PAM_confmat(Cats,Dogs)
[m,n]=size(Cats);
CM=zeros(2);
wrongCats=[];
```

```
wrongDogs=[];
for i=1:n
    test_cat=Cats(:,i);
    test_dog=Dogs(:,i);
    probes=[test_cat test_dog];
    if i==1
        cats=Cats(:,2:n);
        dogs=Dogs(:,2:n);
    elseif i==n
        cats=Cats(:,1:n-1);
        dogs=Dogs(:,1:n-1);
    else
        cats=[Cats(:,1:i-1) Cats(:,i+1:n)];
        dogs=[Dogs(:,1:i-1) Dogs(:,i+1:n)];
    end

    labels=class_principal(cats,dogs,probes);

    if labels(1)==1
        CM(1,1)=CM(1,1)+1;
    else
        CM(1,2)=CM(1,2)+1;
        wrongCats=[wrongCats i];
    end

    if labels(2)==0
        CM(2,2)=CM(2,2)+1;
    else
        CM(2,1)=CM(2,1)+1;
        wrongDogs=[wrongDogs i];
    end
end
```

Appendix E. Wavelet Edges Method

```
%%This code finds the edge of an image
%Written by Austin Adams

%Input: Data
```

```
%   Data=original images, each in column form (Should be 4096 by 1 for each
%   image)

function [Edges]=wave_edge(Data)
[m,n]=size(Data);
for i=1:n
    [a,h,v,d]=dwt2(reshape(Data(:,i),64,64),'haar');
    edges=h+v;
    Edges(:,i)=reshape(edges,32*32,1);
end
```

Appendix F. Wavelet Edges Classification with Principal Angles

```
%%This code classifies cats and dogs using wavelets and principal angles
%Written by Austin Adams

%Input:Cats,Dogs,Probes
%   Cats=gallery of cats(each image is 4096 by 1)
%   Dogs=gallery of dogs(ditto)
%   Probes=test images(ditto)

%Output:Class
%   Class=labels(1==cat 0==dog) in vector form

function [labels]=class_wave(Cats,Dogs,Probes)
[mp,np]=size(Probes);
Cat_edge=wave_edge(Cats);
Dog_edge=wave_edge(Dogs);
Probe_edge=wave_edge(Probes);
labels=1:np;
for i=1:np
    Pvscat=prinangles(Probe_edge(:,i),Cat_edge);
    Pvsdog=prinangles(Probe_edge(:,i),Dog_edge);
    if Pvscat < Pvsdog
        labels(i)=1;
    elseif Pvsdog < Pvscat
        labels(i)=0;
    end
end
```

Appendix G. Wavelet Confusion Matrix

```
%%This code evaluates the confusion matrix for wave/principle angles method
%Written by Austin Adams

%Input:Cats,Dogs
%   Cats=first class
%   Dogs=second class
%Note Cats and Dogs must be same size

%Output:CM
%   CM=confusion matrix

function [CM,wrongcats,wrongdogs]=WAV_confmat(Cats,Dogs)
[m,n]=size(Cats);
CM=zeros(2);
wrongcats=[];
wrongdogs=[];
for i=1:n
    test_cat=Cats(:,i);
    test_dog=Dogs(:,i);
    probes=[test_cat test_dog];
    if i==1
        cats=Cats(:,2:n);
        dogs=Dogs(:,2:n);
    elseif i==n
        cats=Cats(:,1:n-1);
        dogs=Dogs(:,1:n-1);
    else
        cats=[Cats(:,1:i-1) Cats(:,i+1:n)];
        dogs=[Dogs(:,1:i-1) Dogs(:,i+1:n)];
    end

    labels=class_wave(cats,dogs,probes);

    if labels(1)==1
        CM(1,1)=CM(1,1)+1;
    else
        CM(1,2)=CM(1,2)+1;
        wrongcats=[wrongcats i];
    end

    if labels(2)==0
        CM(2,2)=CM(2,2)+1;
    else
        CM(2,1)=CM(2,1)+1;
        wrongdogs=[wrongdogs i];
    end
end
```

Appendix H. Main Script 2

```
 %% Math 695 Final Project
% Written by Amy Mulgrew

% load galleryFiles, do PCA on full gallery 99% (keep ensemble average), mean
% subtract the probe, Classify with FDA and with Principal Angles.
% Cats == 1  %Dogs == 0

% Input: Test set (Probe) & actual classification of the Test set (Actual)
% Output:
    % view: 3-by-38 matrix, top row actual classification, second row FDA
        % classification, third row PA classification
    % confFDA: confusion matrix from FDA method
    % confPA: confusion matrix from PA method
    % time: time it takes the program to run

function [ view confFDA confPA time] = MulgrewFinalProject2(Probe, Actual)
tic;
load galleryFiles
catsGallery = [D1 D1flip];
dogsGallery = [D2 D2flip];
Gallery = [catsGallery dogsGallery];
[AGal,U,D,GalEnsAvg] = PCA(Gallery,99);
Ud = U(:,1:D);  %dimension reduced KL basis
[dim N] = size(AGal);
catsKL = AGal(:,1:(N/2));        %split back into the 2 sets of data points
dogsKL = AGal(:,(N/2+1):N);
nProbe = size(Probe,2);
```

```
N = 2* size(catsKL,2);
ProbeMS = Probe - repmat(GalEnsAvg,1,nProbe);  %Mean Subtract the Probe
ProbeKL = Ud'*ProbeMS; %coefs of Probe
classFDA = NaN(1,nProbe);
classPA = NaN(1,nProbe);
%% FDA
w = finalFDA(catsKL, dogsKL);
vcats = w'*catsKL;
vdogs = w'*dogsKL;
vProbe = w'*ProbeKL;

if mean(vcats)< mean(vdogs)
    w = -w;
    vcats = -vcats; vdogs = -vdogs; vProbe = -vProbe;
end
%dogs < threshold < cats
sortcats = sort(vcats);
sortdogs = sort(vdogs);
threshold = .5*(sortcats(1) + sortdogs(end));

% plot FDA
m = sortdogs(1);
M = sortcats(end);
x = m-10:M+10;
figure(1);
clf
plot(x,0,'k-');
axis([x(1) x(end) -.5 1.5])
hold on
plot(threshold,0,'y*');

for i = 1:length(sortdogs)
    plot(sortdogs(i),0,'r+');    %dogs red
    plot(sortcats(i),0,'b+');   % cats blue
end

for i = 1:nProbe
    if Actual(i) == 0
        plot(vProbe(i),1,'m*'); %is a dog, so make pink
    else
```

```
        plot(vProbe(i),1,'c*'); %is a cat, so make cyan
    end

    if vProbe(i) <= threshold   %thinks it's a dog
        classFDA(i) = 0;
    else classFDA(i) = 1;    %thinks it's a cat
    end
end

%% PA
mm = N;  %%number of elements in the Gallery
AGal = [catsKL dogsKL];
for probe = 1:size(ProbeKL,2)
    for class = 1:2
        Digit = AGal(:,(class-1)*(mm/2)+1:class*(mm/2));
        theta(class) = PrincipalAngles(Digit,ProbeKL(:,probe));
    end

    if theta(1) < theta(2)   % thinks is a cat
        classPA(probe) = 1;
    else classPA(probe) = 0;  %thinks is dog
    end
end

%% Results
view = [Actual; classFDA ; classPA];

% confusion matrix for FDA & PA
confFDA = zeros(2,2);
confPA = zeros(2,2);
for i = 1:length(Actual)
    if Actual(i) == 1  %is a cat
        if classFDA(i) == 1 %thinks cat correctly
            confFDA(1,1) = confFDA(1,1) + 1;
        else confFDA(1,2) = confFDA(1,2) + 1;  %thinks dog incorrectly
        end
        if classPA(i) == 1 %thinks cat correctly
            confPA(1,1) = confPA(1,1) + 1;
        else confPA(1,2) = confPA(1,2) + 1;  %thinks dog incorrectly
        end
    else
        if classFDA(i) == 0  %thinks dog correctly
            confFDA(2,2) = confFDA(2,2) + 1;
        else confFDA(2,1) = confFDA(2,1) + 1; %thinks cat incorrectly
        end
        if classPA(i) == 0  %thinks dog correctly
            confPA(2,2) = confPA(2,2) + 1;
        else confPA(2,1) = confPA(2,1) + 1; %thinks cat incorrectly
        end
    end
end
time = toc;
```

Appendix I. Principal Component Analysis

```
%% Principal Component Analysis
% Written by Amy Mulgrew
function [AX,U,D,XEnsAvg] = PCA(X,EnergyRetained)
% function PCA.m
% input: X matrix, desired energy retained
% output: AX (reduced set of coefficients for X in KL basis)(DxP),
%       and   U (reduced set of eigenvectors/KL Basis)(NxD),
%       and D (D value that retains the desired energy)

[N P]= size(X);
X = double(X);
XEnsAvg = (1/P)*sum(X,2);                          %calculate Ensemble Average

X = X - repmat(XEnsAvg,1,P);                 %mean subtract X

[U,sigma,V] = svd(X,0);                              % expensive calculation
AX = sigma*V';          % these are the expansion coefficients for X in KL Basis

evals = (diag(sigma)).^2;
% picks a D value based off retaining 99% Energy like in HW2
percent = .01*EnergyRetained;
D=1;
for i = 1:P
    Si = evals(1:i,:);
    E(i) = sum(Si)/sum(evals);
    if E(i) < percent
        D = D+1;
    end
end
D = D - 1;
AX = AX(1:D,:);
```

Appendix J. Principal Angle Finder

```
%% This code finds the principal angle, theta, between the subspaces input
% Written by Amy Mulgrew

%Input: X, Y (two matrices with same number of rows)
%Ouput: theta

function [ theta ] = PrincipalAngles( X,Y )

p = size(X,2);
q = size(Y,2);

% SVD for cosine
[Qx,R]=qr(X,0);
[Qy,R]=qr(Y,0);

M = (Qx)'*Qy;
C = svd(M,0);

 rkX = rank(Qx);
 rkY = rank(Qy);
 if rkX >= rkY;
  B = Qy - Qx*((Qx)'*Qy);
 else
  B = Qx - Qy*((Qy)'*Qx);
 end

%SVD for sine
S = svd(B,0);
S = sort(S);

p = min(p,q);
```

```
theta = zeros(p,1);  % initialize theta

for i = 1:p
 if (C(i))^2 < .5
  theta(i) = acos(C(i));
 elseif (S(i))^2 <= .5
  theta(i) = asin(S(i));
 end
end
```

Appendix K. Optimal projection for FDA

```
%% This code finds the optimal projection w for FDA
% Written by Amy Mulgrew

%% Input: catsKL, dogsKL
%   catsKL: the reduced dimension coefficients of the cats in the KL basis
%   dogsKL: the reduced dimension coefficients of the dogs in the KL basis

%% Output: w

function [w] = finalFDA(catsKL, dogsKL)

n = zeros(1,2);
n(1) = size(catsKL,2);
n(2) = size(dogsKL,2);
m(:,1) = sum(catsKL,2)/(n(1));        % calculate the class-wise means
m(:,2) = (1/n(2))*sum(dogsKL,2);


%% create the between class scatter matrix: Sb
Sb = (m(:,2) - m(:,1))*(m(:,2) - m(:,1))';

%% create the within class scatter matrix: Sw
dim = size(catsKL,1);
Sw = zeros(dim);
for i = 1:n(1)
    new = (catsKL(:,i)-m(:,1))*(catsKL(:,i)-m(:,1))';
    Sw = Sw + new;
end
```

```
for i = 1:n(2)
    new = (dogsKL(:,i)-m(:,2))*(dogsKL(:,i)-m(:,2))';
    Sw = Sw + new;
end

%% solve generalized eigenvalue problem Sb*w = lambda*Sw*w
SwI = pinv(Sw);
S = SwI*Sb;
[Unew,Sigma,V] = svd(S,0);
w = Unew(:,1);                  %w is the eigenvector corresponding to the
                                %largest eigenvalue, so with svd, it is
                                % the first eigenvector
```

Department of Mathematics and Statistics, California State University, Long Beach, 1250 Bellflower Blvd., Long Beach, CA 90840-1001