CLASSIFICATION OF CATS AND DOGS

BY TUYEN LY, JOHN HAMMER
MATH 521 SPRING 2012
CSULB DEPT. OF MATHEMATICS

Abstract. This paper is a report on the methods and findings of two image classification algorithms. Specifically, given a training set of 160 grayscale images of cat and dog faces the authors explore two standard classification methods to correctly identify 38 unclassified cat and dog images. The first method discussed uses Linear Discriminant Analysis (LDA) to separate the two types of images from each other. The second method extracts and measures the novel details of each test image relative to the training set. Furthermore, to increase the rate of accuracy the images are preprocessed using a filter mask that highlights the basic shape of the image subject. Results demonstrate that the second method performs better, resulting in a 94% classification rate.

1. Introduction

As children, we learn how to identify different kinds of animals. Our brain stores and sorts out information we receive from various sources (i.e. parents, television, books, etc.) about the distinctive features belonging to such creatures. So when we look at different animals, we are able to tell the difference between say, a cat and a dog. That means there must be some uniquely inherent features possessed in each animal.

Our goal is to create a classification algorithm that trains the computer to extract these distinctive features and correctly classify animals, namely cats and dogs. There are various features to create our classifier around, such as shape of the faces, fur details, the angles created from the faces, etc. After serious consideration, we chose to ignore the minute details and focused our attention to design a classifier that uses the shape of the animal faces to distinguish one from another. In general, cats tend to have rigid pointy ears and round faces while dogs tend to have droopy ears and elongated facial structures.

Given a set of 160 grayscale images (80 cats and 80 dogs) we use the methods of Linear Discriminant Analysis (LDA) and Kohonen's Novelty Filter to train the classifier. Finally, we test the accuracy of our algorithms against a set of 38 images (19 cats and 19 dogs) not found in the training set.

2. Theoretical Background

2.1. **Principle Component Analysis.** Prior to performing the classification algorithms, we transform each 64×64 grayscale image into a 4096×1 column vector by concatenating the columns in the image array. As a result, the dimension of the data matrices become quite large. To handle data sets of such size we employ the well-researched theory of Singular Value Decomposition.

Recall that for any real-valued matrix $A_{m\times n}$ there exists two orthogonal matrices $U_{m\times m}, V_{n\times n}$ and a diagonal matrix $\Sigma_{m\times n}$ with diagonal entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$ such that

$$A = U\Sigma V^T$$

The column vectors in U form a basis for the space in which A resides. The diagonal entries of Σ are called the *singular values*, and their values can be used to rank the vectors in U by level of importance when representing the information in A.

Principle Component Analysis (PCA) takes advantage of this observation by simply excluding "non-important" basis vectors, leaving behind what we call the *principle components*. Now, the typical way of selecting the principle components is by calculating the *cumulative energy* of the singular values using the formula

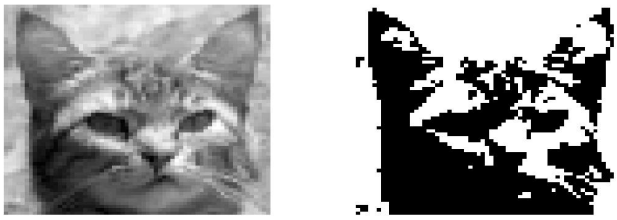
$$\frac{\sum_{i=1}^{d} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}$$

where $r = rank(A)$ and d is the first number of singular values that yields a cumulative energy higher than 97%. Therefore, we take the first d column vectors of U as the approximated basis for the matrix A. We then have the approximated matrix $A^{(d)}$ given by

$$A^{(d)}_{m\times n} = U_{m\times d}\Sigma_{d\times d}V^T_{d\times n}$$

Choosing the principle components formally reduces the dimension of the data matrix while still retaining the most important information. Therefore, PCA makes for a great process to simplify data sets prior to performing intense calculations.

2.2. **Filter Mask.** Since we are mainly interested in the overall shape of the dog or cat, we need a way to erase minute details and separate the subject from the background. The filter mask is a preprocessing step we use to simplify the image data. It works by replacing each pixel's intensity with a zero or a one. We choose how to reassign each pixel intensity by comparing it to the median intensity of the entire image. The procedure is explained in section 3.2. Below is an example image that has been transformed using the filter mask.



(A) Original Image (B) Filter Mask Image

FIGURE 1

We found that this process works best on pictures that have a homogeneous (smooth) background and whose subjects occupy most of the image.

2.3. **Linear Discriminant Analysis.** The general theory behind Linear Discriminant Analysis (LDA) is quite involved, and a well-written text on the construction of the method is found in [**1**]. In any case, we will examine the process here.

Consider two data sets, D_1 and D_2, with number of data points n_1 and n_2 respectively. The objective of LDA is to find an optimal direction $\mathbf{w}$ such that the projections of the data points are grouped with their peers and are separated from strangers. Figure 2 below illustrates this idea with three different data sets.

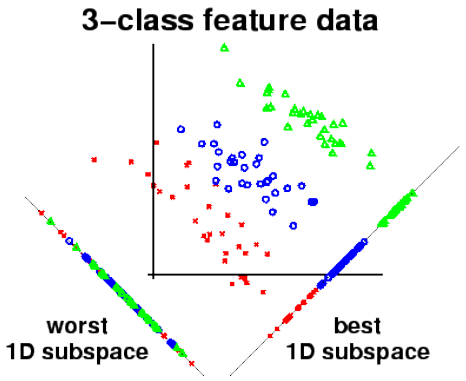


FIGURE 2. An example of poorly-chosen projection line and an optimal projection line

As you can see, we want data points from the same class to be close together and data points from different classes to be far apart. We now put these ideas into mathematical terms.

Define the *class-wise means*

$$\mathbf{m}_1 = \frac{1}{n_1} \sum_{\mathbf{x} \in D_1} \mathbf{x} \qquad \text{and} \qquad \mathbf{m}_2 = \frac{1}{n_2} \sum_{\mathbf{y} \in D_2} \mathbf{y}$$

Then the means of the projected data are given by $\hat{m}_1 = \mathbf{w}^T m_1$ and $\hat{m}_2 = \mathbf{w}^T m_2$. Since we want to maximize the distance between the different data sets we write

$$N(\mathbf{w}) = \max_{\mathbf{w}} (\hat{m}_2 - \hat{m}_1)^2$$

This expression can also be written as

$$N(\mathbf{w}) = \mathbf{w}^T S_B \mathbf{w}$$

where $S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$ is called the *between-class scatter matrix*. Furthermore, we want to minimize the variance of the projected data for each class. We describe this by the scalar value

$$D(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1,2} \sum_{\mathbf{x} \in D_i} (\mathbf{w}^T \mathbf{x} - \hat{m}_i)^2$$

Again, this expression can be written as

$$D(\mathbf{w}) = \mathbf{w}^T S_W \mathbf{w}$$

where $S_W = \sum_{i=1,2} \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$ is called the *within-class scatter matrix*. Putting these together as a ratio we get what is called the generalized Rayleigh quotient:

$$J(\mathbf{w}) = \frac{N(\mathbf{w})}{D(\mathbf{w})} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

Now, it can be shown that the maximum $\mathbf{w}$ that satisfies $J(\mathbf{w})$ is found by solving the generalized eigenvalue problem

$$S_B \mathbf{w} = \lambda S_W \mathbf{w}$$

Therefore, we have a way to find the optimal direction that separates the two data sets!

2.4. **Novelty Filter.** The novelty filter algorithm relies primarily on the theory of orthogonal projection matrices. By definition, a matrix P is called an *orthogonal projection matrix* if the subspaces $\mathscr{R}(P)$ and $\mathscr{R}(I-P)$ are orthogonal. The matrix $I-P$ here is called the *complementary projection matrix*. Furthermore, it can be shown that any matrix of the form $P = MM^T$ is a projection matrix.

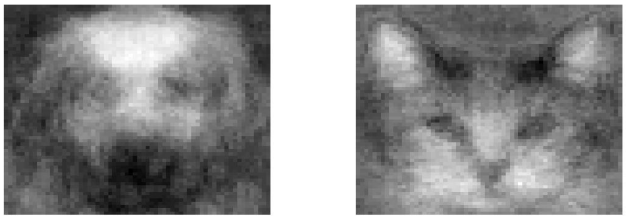
Orthogonal projection matrices are important to us because they can decompose a space W into orthogonal subspaces. Therefore, any vector in that space can be uniquely written as a sum of vectors from each orthogonal subspace. Formally, if $\mathbf{x} \in W$ and $W = \mathscr{R}(P) \dot{\oplus} \mathscr{R}(I-P)$, then $\mathbf{x}$ can be written as

$$\mathbf{x} = \mathbf{u} + \mathbf{v}$$

where $\mathbf{u} \in \mathscr{R}(P)$ and $\mathbf{v} \in \mathscr{R}(I-P)$. The novelty filter uses this idea to separate familiar patterns in test data from new, or *novel*, patterns. In particular, the familiar patterns will live in $\mathscr{R}(P)$ while the novel patterns will reside in $\mathscr{R}(I-P)$.

3. Algorithm Implementation and Development

3.1. **Principle Component Analysis.** The method for PCA we used is similar to the version discussed in class. We made, however, one change in the process: **We do not mean subtract the data.** The reason is largely due to the high variability of the dog data. That is, we found that the dogs were so different in appearance that the ensemble average would take away critical information from each image. Figure 3 shows the ensemble average images of the training data. Observe that the cat ensemble average almost looks like it could be a training image!



(A) Average Ensemble of Dogs (B) Average Ensemble of Cats

Figure 3

The procedure for PCA is given below, and the MATLAB code can be found in section 6.4:

1. Construct the optimal basis using Singular Value Decomposition
2. Find the best d approximation of the basis using the cumulative energy procedure.

3.2. **Filter Mask.** The filter mask algorithm is very simple. Again, its purpose is to transform the intensity of each pixel in the image to either a zero or a one. The general procedure is described below, and the MATLAB code can be found in section 6.5:

Let $D = \{d^{(1)}|\dots|d^{(n)}\}$ be a data matrix of dimension $m \times n$.
For each $d^{(k)}$, $k = 1, \dots, n$
    Calculate the median of $d^{(k)}$
    for $j = 1, \dots, m$
        if $d_j^{(k)} < median$, set $d_j^{(k)} = 0$
        if $d_j^{(k)} \geq median$, set $d_j^{(k)} = 1$
    end
end

3.3. **Linear Discriminant Analysis.** LDA attempts to find a projection vector **w** that groups like data and separates dissimilar data in an optimal way. The general algorithm is given below, and the MATLAB code can be found in section 6.6

Given two sets of data, D_1 and D_2, with number of data points n_1 and n_2 respectively, do the following:

1. Define the class-wise means
$$m_1 = \frac{1}{n_1}\sum_{x\in D_1} x \qquad \text{and} \qquad m_2 = \frac{1}{n_2}\sum_{y\in D_2} y$$
2. Construct the between-class scatter matrix
$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$
3. Construct the within-class scatter matrix
$$S_W = \sum_{i=1}^{2}\sum_{x\in D_i}(x - m_i)(x - m_i)^T$$
4. Solve the generalized eigenvalue problem given by
$$S_B\mathbf{w} = \lambda S_W\mathbf{w}$$
where $\mathbf{w}$ is the generalized eigenvector corresponding to the largest generalized eigenvalue λ. We take $\mathbf{w}$ to be the optimal projection vector.

Next, we describe the classification process that uses the optimal projection vector $\mathbf{w}$. First, we project the data sets, D_1 and D_2, onto a number line spanned by $\mathbf{w}$. Second, we use the average of the *projected* class-wise means, $\hat{m_1}$ and $\hat{m_2}$, to establish a boundary value on the number line that separates the domains of each set. This average is given by
$$\frac{\hat{m_1} + \hat{m_2}}{2}$$

Third, and finally, we project each test image onto the number line and determine which domain it resides in relative to the boundary value. The MATLAB code can be found in section 6.7.

3.4. **Novelty Filter.** The novelty filter classification method requires only a few steps. We have outlined the procedure below, and the MATLAB code can be found in section 6.8.
Given two sets of training data, D_1 and D_2, and a testing set X, do the following:

1. Construct an orthonormal basis for each training set using PCA. Denote them U_1 and U_2.
2. Calculate the complementary orthogonal projection matrix given by
$$P_i = I - U_iU_i^T \qquad i = 1, 2$$
3. For each element $x^{(k)} \in X$
    Calculate the projection $v_i = P_ix^{(k)}$, for i=1,2
    if $norm(v_1) < norm(v_2)$
        Classify $x^{(k)}$ as a member of D_1.
    else
        Classify $x^{(k)}$ as a member of D_2.
    end
end

4. Computational Results

After running the classification methods, our results were fairly good. The optimal vector and the boundary value (indicated by □) that LDA obtained clearly separates the training set (indicated by ∗), shown in Figure 4. The red marks represent the "dog " data and the blue marks represent the "cat " data. Figure 4 also shows how the test set (indicated by ◯) were classified.

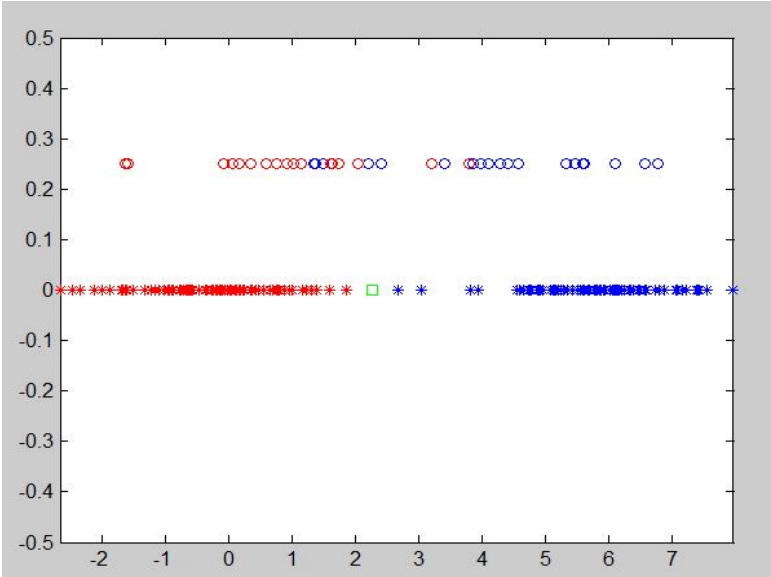


Figure 4. Classification Results Using LDA

The confusion matrix in Figure 5 shows the numerical results of the LDA classification. Dogs were correctly identified more than the cats. As you can see, 16 dogs were correctly classified as dogs, but 3 dogs were considered cats. On the other hand, 15 cats were correctly classified, and 4 cats were mistaken for dogs. In summary, 31 out of 38 images were correctly classified, yielding a 81.58% accuracy for the LDA method.

	as Dogs	as Cats
Dogs	16	3
Cats	4	15

Figure 5. Confusion Matrix Using LDA

Figure 6 shows the images of the cats and dogs that were misclassified using the LDA algorithm. Image 19 has a lot of noise in the background, so that might be the reason why the LDA classifier

wasn't able to correctly classify that particular image.

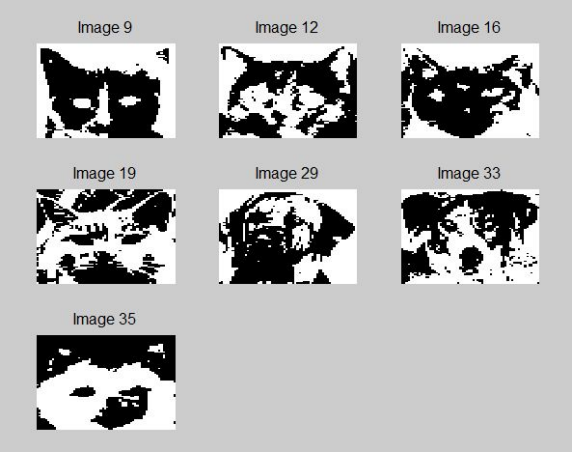


FIGURE 6. Misclassified Images Using LDA

The novelty filter algorithm performed much better than LDA. In Figure 7, the confusion matrix shows that all the cats were correctly classified, and only 2 dogs were mistaken for cats. A total of 36 out of 38 images were correctly classified, yielding a 94.74% accuracy for the Novelty Filter method.

	as Dogs	as Cats
Dogs	17	2
Cats	0	19

FIGURE 7. Confusion Matrix Using Novelty Filter

Figure 8 shows the two images of the dogs that were misclassified. Both images of the dogs had pointy ears, which could have caused the misclassification. Upon closer examination, we found that cat eyes were found to be in the center of the image, while dog eyes tended to be in the upper half of the picture. Notice that the two misclassified dogs have their eyes in the center of the image.

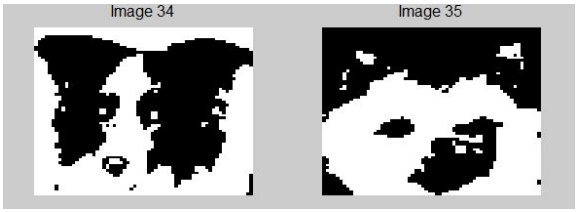


FIGURE 8. Misclassified Images Using Novelty Filter

5. SUMMARY

In summary, creating a classifier to identify images of cats and dogs can be done with good accuracy. It seems that the simpler algorithm (i.e. Novelty Filter) performs better than the more complex algorithm (LDA). Using less information seems to give better results, whereas obtaining more information seems to "confuse" the algorithm. Applying the filter mask to set the image pixels to "zeros" or "ones" made the images simple and the shape more discernible. This was a very simple transformation of the image, yet resulted in high classification rates.

Even though the Novelty Filter gave our highest accuracy, there could be some modifications done to improve the classification even further. One idea is to use the filter mask after applying the median filter or the average filter to "smooth out" the images. This could potentially reduce the background noise and make the facial features more distinctive, possibly giving higher rates.

6. MATLAB Code

6.1. **Main Script.**

```
% Final Project Main Script
% by John Hammer, Tuyen Ly
% Math 521, Spring 2012
clear all;

%--- Get Raw Training Data and Raw Testing Data ---%
M = 160;
N = 38;
Train_Raw = mkData('TIFFTraining', M);
Test_Raw = mkData('TIFFTesting', N);
True = [ones(N/2,1); zeros(N/2,1)]; % Correct Classification Sequence


%--- Run Filter Mask on Training and Test Data ---%
Train_Bin = binaryData(Train_Raw, M);
Test_Bin = binaryData(Test_Raw, N);


%--- Run Classification Algorithms on Filtered Data ---%
Result = Hammer_Ly_Final_LDA(Train_Bin, Test_Bin, M, N);
mkResults(Test_Bin, True, Result, N);

Result = Hammer_Ly_Final_PCA(Train_Bin, Test_Bin, M, N);
mkResults(Test_Bin, True, Result, N);
```

6.2. **Function mkData.**

```
% mkData.m
% by John Hammer, Tuyen Ly
% Math 521, Spring 2012
function Data = mkData(directory,N)
% directory is the path where the images are stored
% N is the number of images to convert to column vectors
% Data is the data matrix containing the column vectors
Data = zeros(64*64,N);

cd(directory);
Names = dir('*.tif');
for k = 1:N
    Img = double(imread(Names(k).name));
    Data(:,k) = reshape(Img,64*64,1);
end
cd('..');
end
```

6.3. **Function mkResults.**

```
% mkResults.m
% by John Hammer, Tuyen Ly
% Math 521, Spring 2012
function mkResults(Test, True, Result, N)
% Test is the test image Set
% True holds the CORRECT classifications of test images
% Result is the METHOD classifications of test images
% Dog is zero, Cat is one
% N is the number of Test images
%
% Conf is a 2x2 confusion matrix given by
% [ Dogs as Dogs , Dogs as Cats ]
% [ Cats as Cats , Cats as Dogs ]


%--- Construct 2x2 Confusion Matrix ---%
Conf = zeros(2);
X = True - Result;

for k = 1:N
    if X(k) == -1
        Conf(1,2) = Conf(1,2) + 1;      % Dog as Cat
    elseif X(k) == 1
        Conf(2,2) = Conf(2,2) + 1;      % Cat as Dog
    else
        if True(k) == 0
            Conf(1,1) = Conf(1,1) + 1;  % Dog as Dog
        else
            Conf(2,1) = Conf(2,1) + 1;  % Cat as Cat
        end
    end
end
disp('Results are');
disp(Conf);

%--- Display incorrectly classified images ---%
i = 0;
figure()
nWrong = sum(Conf(:,2));

for k = 1:N
    if X(k) ~= 0
        i = i+1;
        subplot(ceil(sqrt(nWrong)),ceil(sqrt(nWrong)),i);
        imagesc(reshape(Test(:,k),64,64)); colormap(gray); axis off;
```

```
        str = sprintf('Image %d',k); title(str);
    end
end
end % end function
```

6.4. **Function PCA.**

```
% PCA.m
% by John Hammer, Tuyen Ly
% Math 521, Spring 2012
function [D,KL,A] = PCA(X, TOL)
% X is the Data Ensemble
% TOL is the tolerance
% D is the reduced dimension
% KL is the KL basis
% A contains the coordinates of each data point in X relative to KL basis


%--- Perform SVD ---%
[KL,S,V] = svd(X,'econ');

%--- Cumulative Energy Procedure ---%
ds = diag(S).^2;
dsum = sum(ds);
ksum = 0;
for k = 1:length(ds)
    ksum = ksum + ds(k);
    if (ksum/dsum) > TOL
        D = k;              % Best D Approx for KL basis
        break;
    end
end

KL = KL(:,1:D);             % KL basis
A = S(1:D,1:D)*V(:,1:D)';   % Expansion coefs
end
```

6.5. **Function Filter Mask.**

```
% binaryData.m
% by John Hammer, Tuyen Ly
% Math 521, Spring 2012
function [Data] = binaryData(Data, N)
% Apply Filter Mask to the Data set

for k = 1:N
    mv = median(Data(:,k)); % The medain value in Data(:,k)

    for j = 1:64*64         % Change each value to either zero or one
        if Data(j,k) < mv
            Data(j,k) = 0;
        else
            Data(j,k) = 1;
        end
    end

end
end % end function
```

6.6. **Function LDA.**

```
function [W] = LDA(D1,D2,M)
% D1 is the first Data Set
% D2 is the second Data Set
% M is the row dimension for both classes
% W is the Optimal Projection Vector

%--- Find number of data points in each set ---%
n1 = size(D1,2);
n2 = size(D2,2);

%--- Define class-wise means ---%
m1 = sum(D1,2)/n1;
m2 = sum(D2,2)/n2;

%--- Define between-class scatter matrix ---%
v = m2-m1;
Sb = v*v';

%--- Define within-class scatter matrix ---%
Sw = zeros(M,M);
for i = 1:n1
    v = D1(:,i) - m1;
    Sw = Sw + v*v';
end

for i = 1:n2
    v = D2(:,i) - m2;
    Sw = Sw + v*v';
end

%--- Solve generalized eigenvalue problem Sb*W=L*Sw*W ---%
[V,D] = eig(Sb,Sw);

%--- Find Eigenvector with Largest Eigenvalue ---%
evals = diag(D);
evmax = evals(1);
ind = 1;                          % Index of largest EV
for i=2:length(evals)
    if evmax < evals(i)
        evmax = evals(i);
        ind = i;
    end
end
W = V(:,ind)/norm(V(:,ind),2);  % Optimal Projection vector
end
```

6.7. **LDA Classification Algorithm.**

```
function [Result] = Hammer_Ly_Final_LDA(Train,Test,M,N)
% LINEAR DISCRIMINANT ANALYSIS
% Train is the set of Training data
% Test is the set of Testing data
% M is the number of images in Train
% N is the number of images in Test
% Result holds the classifications of test data
% dog is zero, cat is one
Result = zeros(N,1);

%--- Reduce Dimension of Training Data using PCA ---%
[d,KL,A] = PCA(Train, 0.97);
Cats = A(:,1:M/2);
Dogs = A(:,M/2+1:M);

%--- Reduce Dimension of Test Data ---%
Test = KL'*Test;

%--- Perform LDA and Project Data onto Real Line ---%
[W] = LDA(Cats,Dogs,d);
C = W'*Cats;
D = W'*Dogs;

%--- Find threshold value alpha ---%
Cmin = min(C);
Cmax = max(C);
Dmin = min(D);
Dmax = max(D);
Cpos = 0;
if Cmax < Dmax                        % Cats are left of alpha
    alpha = (Cmax + Dmin)/2;
    Cpos = 1;
    span = [Cmin Dmax -0.5 0.5];
else
    alpha = (Cmin + Dmax)/2;
    span = [Dmin Cmax -0.5 0.5];
end


%--- Plot Projected Data ---%
figure();
plot(C,zeros(M/2,1),'b*'); hold on;  % Plot Cat Data Set in Blue
plot(D,zeros(M/2,1),'r*');           % Plot Dog Data Set in Red
plot(alpha,0,'gs');                  % Plot threshold value
```

```
%--- Classify and Plot Test Animals ---%
for k = 1:N
    P = W'*Test(:,k);
    if Cpos
        if P < alpha
            Result(k) = 1;
        end
    else
        if P > alpha
            Result(k) = 1;
        end
    end

    if k <= N/2
        plot(P,0.25,'bo');
    else
        plot(P,0.25,'ro');
    end
end
hold off; axis(span);
end % end function
```

6.8. **Novelty Filter Classification Algorithm.**

```
% Novelty Filter Classification Algorithm
function [Result] = Hammer_Ly_Final_PCA(Train,Test,M,N)
% Train is the set of Training Data
% Test is the set of Testing Data
% M is the number of entries in Train
% N is the number of entries in Test
% Result is a vector of the Test classification
% dog is zero, cat is one
Result = zeros(N,1);

Set0 = Train(:,M/2+1:M); % Dogs
Set1 = Train(:,1:M/2);   % Cats

%--- Construct Optimal (Orthogonal) Bases ---%
[d0,KL0,A0] = PCA(Set0, 0.97); % Dogs
[d1,KL1,A1] = PCA(Set1, 0.97); % Cats

%--- Calculate Novelty of Test Data Relative to each Basis ---%
for k = 1:N
    x0 = KL0'*Test(:,k);
    x1 = KL1'*Test(:,k);

    % Calculate Norms
    n0 = norm(KL0*x0 - Test(:,k),2);
    n1 = norm(KL1*x1 - Test(:,k),2);

    if n1 < n0
        Result(k) = 1;
    end
end
end % end function
```

Bibliography

1. J.-M. Chang. *Matrix Methods for Geometric Data Analysis and Pattern Recognition.* Dec. 2011