Classifying Digits

Jiasui Qin

March 5, 2020

Abstract

When it comes to classification problems, a data-driven approach(machine learning) has been widely used to help learn the difference between different categories. This process involves extracting representative features of data and learning how these features relate to each category. This paper focuses on both image recognition and image classification. Techniques include wavelet transformation, singular value decomposition, and linear discriminant analysis.

1 Introduction and Overview

I will train on images of digits ranging from 0 to 9 and build a classifier to identify them. Both of the training data and the test data are extracted from the MNIST data set: http://yann.lecun.com/exdb/mnist/. There are 60000 training images and 10000 test images. Each image is represented as a 28×28 matrix with pixel values. My goal is to build classifiers on digits 0, 1, and 2 using linear discriminant analysis. I will also compare the performance of our LDA model with the SVM(support vector machines) model and the decision tree model at the end. Before building the classifier, I first reduce the size of the data by edge detection and then analyze the data using SVD.

My strategy contains five steps:

1. Apply the wavelet transformation to detect the edges for every image.
2. Find the principal components of the transformed data to see how three digits differ in the principal component basis.
3. Use linear discriminant analysis to determine the threshold that separates different categories.
4. Test the algorithm on new data (test data) to see its accuracy.
5. Compare the accuracy of different models.

2 Theoretical Background

2.1 Wavelet Transformation

Edges are sharp transitions in an image. If the image is a function, then edges must have large derivatives. We can represent the derivative of a function $f(x)$ as

$$\begin{aligned} f'(x) &\approx \frac{f(x+h) - f(x-h)}{2h} \\ &= \frac{1}{2h}[(1)(f(x+h)) + (-1)f(x-h)] \end{aligned}$$

We can notice that the second part $[(1)(f(x+h)) + (-1)f(x-h)]$ is simply the Haar wavelet transformation of $f(x)$. In other words, multiplying $f(x)$ by the Haar wavelet approximates the gradient of the function. Therefore, the Haar wavelet is centered at the edges when the derivative is big, applying Haar wavelet transformation can help us detect the edges.

2.2 Singular Value Decomposition(SVD)

Every matrix $A \in C^{m \times n}$ has an SVD. We can decompose A into three matrices:

$$A = USV^T$$

where $U \in R^{m \times m}$ and $V \in R^{n \times n}$ are unitary matrices, and $S \in R^{m \times n}$ is diagonal. The values σ_n on the diagonal of S are called the singular values of the matrix A. The vectors u_n which make up the columns of U are called the left singular vectors of A. The vectors v_n which make up the columns of V are called the right singular vectors of A.[3]

Properties of SVD:

1. The singular values are uniquely determined and are always non-negative real numbers.
2. The singular values are always ordered from largest to smallest along the diagonal of S.
3. The number of nonzero singular values is the rank of A.

A larger singular value means that it contains more information. Therefore, we can sum the first N terms that captures most of the information to get a rank N approximation. The low-rank approximations are a tool for dimensionality reduction for big data. SVD tells us how to keep the fewest number of dimensions of the original data that still represent the most amount of information.

2.3 Linear Discriminant Analysis(LDA)

LDA can be derived as a supervised classification method. Suppose a random variable x comes from one of K classes, LDA tries to divide the data space into K disjoint regions that represent all the classes and then allocates x to class j if x is in region j. The goal of LDA is to find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data.

Suppose we want to classify group 1 and group 2 with means denoted as u_1, u_2, then the between-class scatter matrix, which is a measure of the variance between the groups, is defined as:

$$S_B = (u_2 - u_1)(u_2 - u_1)^T.$$

While the within-class scatter matrix, which is a measure of the variance within each group, is defined as:

$$S_W = \sum_{j=1}^{2} \sum_{x} (x - u_j)(x - u_j)^T$$

If we have more than two groups to classify, the between-class scatter matrix and the within-class scatter matrix are defined as below with u being the overall mean and u_j being the mean of each group:

$$S_B = \sum_{j=1}^{N} (u_j - u)(u_j - u)^T.$$

$$S_W = \sum_{j=1}^{N} \sum_{x} (x - u_j)(x - u_j)^T$$

The goal is to find a vector w such that:

$$w = argmax \frac{w^T S_B w}{w^T S_w w}.$$

The vector w that maximizes the above quotient is the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem:

$$S_B w = \lambda S_w w.$$

3 Algorithm Implementation and Development

3.1 Wavelet Transformation

Matlab has a built-in function for discrete wavelet transformation: [cA, cH, cV, cD] = dwt2(X, 'name')[1]. X is our 2D image, 'name' specifies the wavelet type, which would be Haar in this case. The output contains four parts, cA is an approximation coefficient matrix that represents the low-frequency content. cH, cV, and cD represent the horizontal details(left-right gradients), vertical details(up-down gradients), and diagonal details(diagonal gradients) respectively. In our case, I will add horizontal details and vertical details together to represent our data.

3.2 Singular Value Decomposition(SVD)

Matlab also has a built-in function for SVD: [U,S,V] = svd(X)[4] where X is a matrix representing all our images after the Haar wavelet transformation. To project the data onto the principal component basis, we can simply multiple S by the transpose of V. Then, to get a low rank approximation, I choose to keep only the first 50 PCA modes(first 50 rows of the projected matrix) out of 196 modes.

3.3 Linear Discriminant Analysis(LDA)

After projecting onto the selected PCA modes, we need to find data that only corresponds to digit 0, 1, and 2 by selecting columns with labels 0, 1 and 2. Then, we can build the classifier with the following steps:

1. Compute the mean of each category, and use the mean to compute the between-class scatter matrix as well as the within-class scatter matrix.
2. Use the MATLAB eig()[2] function with these two matrices as arguments to find w.
3. Multiply original data by w to project data onto w.
4. Set the threshold between those values, which is the midpoint of the overlapping points in this case.

Algorithm 1: LDA

m_0 = mean(Zeros,2); m_1 = mean(Ones,2); m_2 = mean(Twos,2);
$S_W = 0; S_B = 0;$
for $m = m_0, m_1, m_2$, Data = Zeros, Ones, Twos **do**
  for k = each column of Data **do**
    $S_W = S_W$ + (Data(:,k) - m)×(Data(:,k) - m)'
  end for
end for
m_{all} = overall mean of Zeros, Ones, and Twos
for $m = m_0, m_1, m_2$ **do**
  $S_B = S_B + (m - m_{all}) \times (m - m_{all})'$
end for
$[V, D] = eig(S_B, S_W)$
w = column vector in V that is associated with the largest value in D
Zeros = $w'\times$ Zeros; Ones = $w'\times$ Ones; Twos = $w'\times$ Twos;
threshold = midpoint of overlapping points between each pair of categories

4 Computational Results

4.1 SVD

Let's look at what the first nine principal components look like and how data points belonging to each digit class spread for the first 3 V-modes:

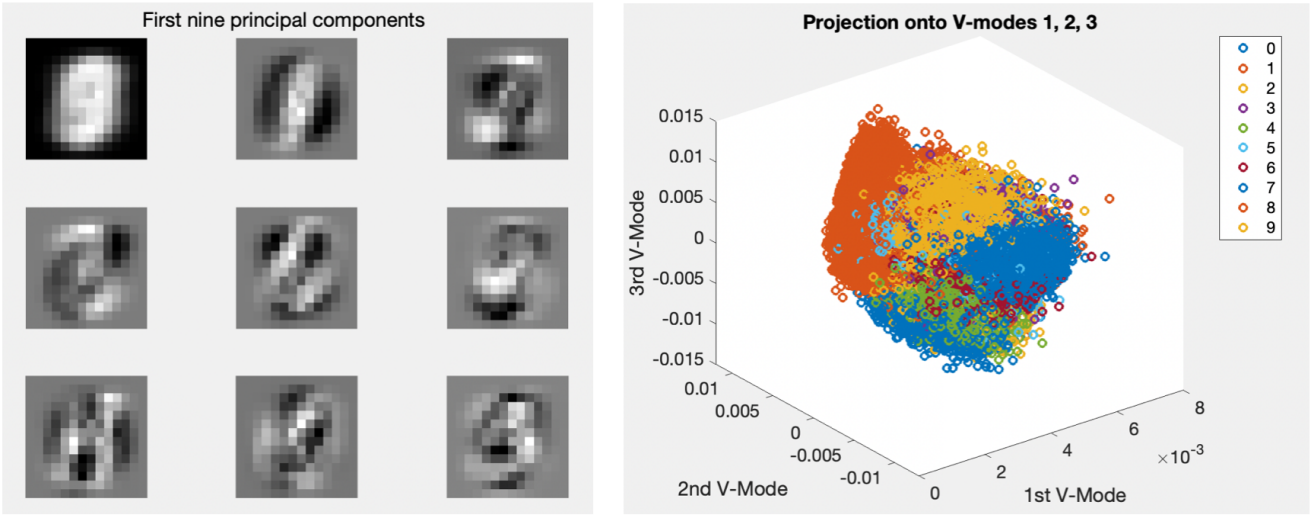

Figure 1: First nine principal components and Projection onto V modes 1,2,3

As we can see, the first principal component captures most of the information, while the other principal components captures information in different areas of the image to determine whether the center or the edges of the digits are covered. According to the projection onto the first 3 V-modes, we notice that data points of the same class tend to cluster together. Let's look at the singular values to see how much weight each principal component carries.

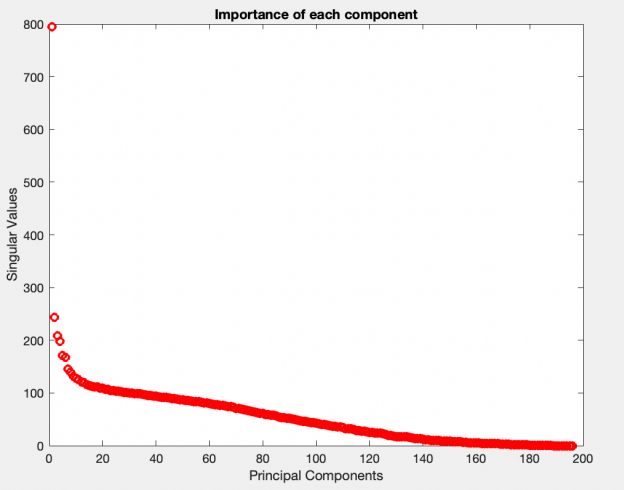

Figure 2: Singular values for each principal component

Obviously, the first mode is dominant. But we can also notice that there isn't a sharp drop-off in the singular values and all of the first 60 components have values above 100. This heavy-tail distribution shows that there is still information in the later modes.

4.2 LDA

4.2.1 Classify three digits

The left side of the following picture are histograms of the projected values with a vertical line for the decision-making threshold. The right side of the pictures shows the projected result of each category.

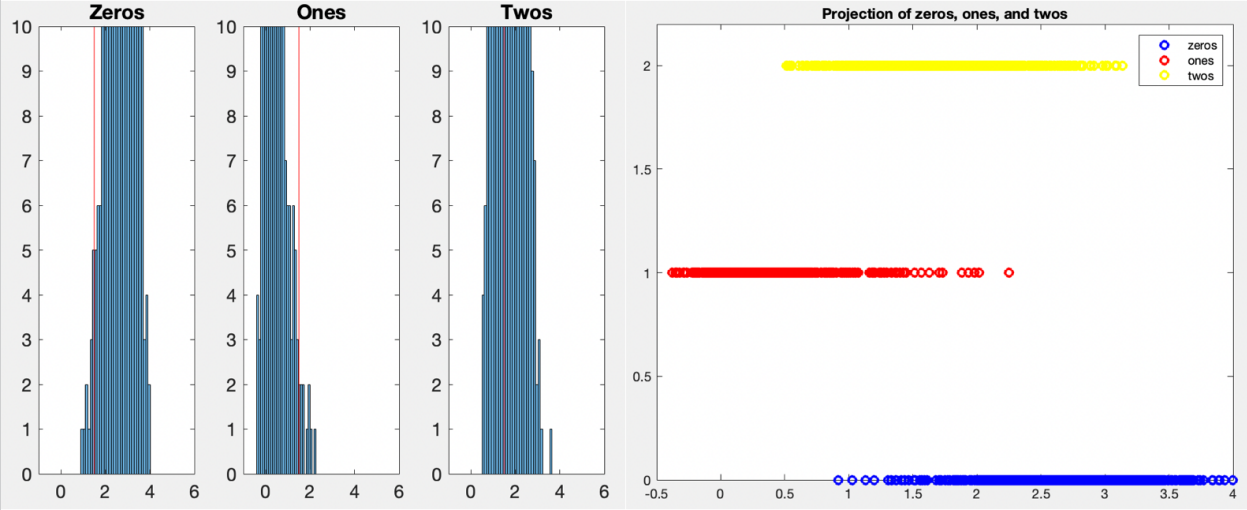

Figure 3: Performance of LDA on all three digits

Based on the result, we can conclude that classifying between zeros and ones is the easiest, while classifying between zeros and twos is the hardest. The following two sections show the performance of LDA on both of these cases and the comparison with other machine learning models.

4.2.2 Classify zero and one

The model makes 3 errors out of 211 testing images, which achieves a success rate of 0.9858. After training with the SVM(support vector machines) model and the decision tree classifier, the loss rate of both models are 0.0156 and 0.0091 respectively. Let's have a look at histograms of the projected values with a vertical line for the threshold and the projected result of each category.

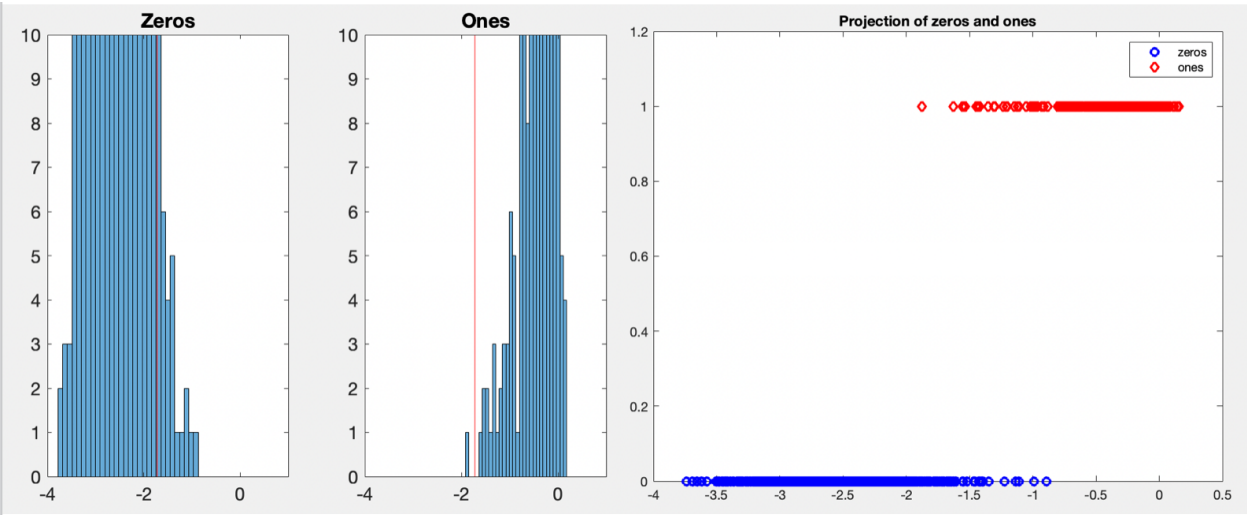

Figure 4: Performance of LDA on zero and one

Let's look at the test examples that are misclassified by the LDA model.

Figure 5: Misclassified testing images

4.2.3 Classify zero and two

The model makes 12 errors out of 201 testing images with a success rate of 0.9403. After training with the SVM(support vector machines) model and the decision tree classifier, the loss rate of both models are 0.0585 and 0.0227 respectively. Here are the histograms of the projected values and the projected result of each category similar as above.

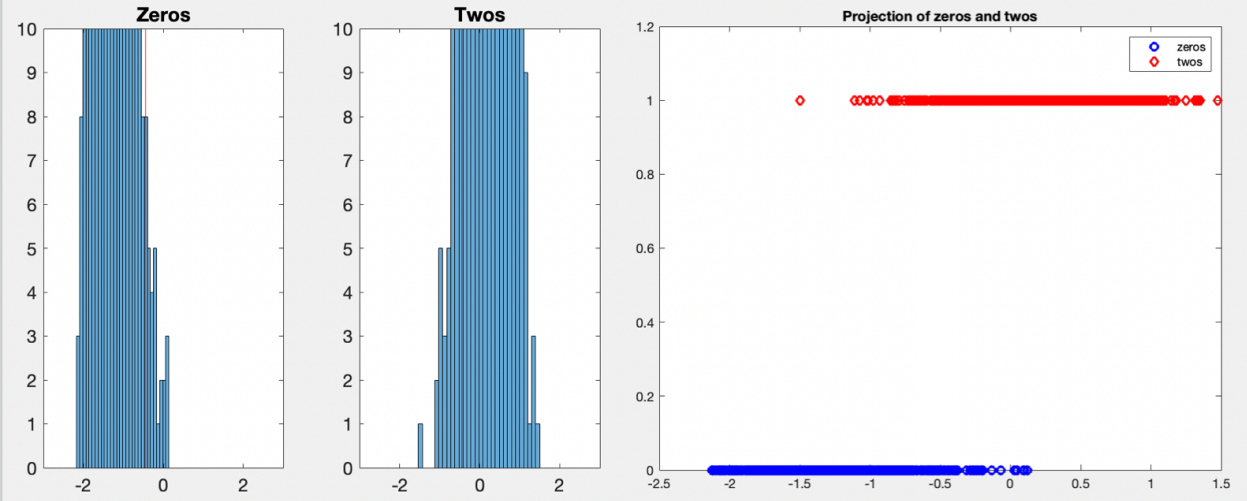

Figure 6: Performance of LDA on zero and two

Let's also look at three test examples that are misclassified by the LDA model.

Figure 7: Misclassified testing images

5 Summary and Conclusions

Applying the Haar wavelet transformation and singular value decomposition helps us reduce the dimension of the data from 784 to 50 while still keeping the most important information. Compare the accuracy of the

LDA model with the loss rate of the SVM model and the decision tree model, the performance of LDA is not as good as the other two models since we only trained a portion of the data for a faster training speed. But in general, all models have a really good performance and are able to classify different digits correctly for most cases. The scores of different models also prove that classifying between zeros and twos is actually harder than classifying between zeros and ones.

References

[1] *dwt2()*. URL: `https://www.mathworks.com/help/wavelet/ref/dwt2.html`.

[2] *eig()*. URL: `https://au.mathworks.com/help/matlab/ref/eig.html`.

[3] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[4] *svd()*. URL: `https://www.mathworks.com/help/matlab/ref/double.svd.html`.

Appendix A MATLAB Functions

- `zeros(m,n)` returns an m-by-n matrix of zeros.
- `ones(m,n)` returns an m-by-n matrix of ones.
- `im2double(I)` converts the image I to double precision.
- `B = reshape(A,sz)` reshapes A using the size vector, sz, to define size(B).
- `mean(A)` computes the mean value. If A is a vector, then mean(A) returns the mean of the elements. If A is a matrix, then mean(A) returns a row vector containing the mean of each column.
- `sort(A)` sorts the elements of A in ascending order.
- `norm(v,p)` returns the generalized vector p-norm.
- `abs(A)` returns the absolute value of each element in array A.
- `[cA,cH,cV,cD] = dwt2(X,wname)` computes the single-level 2-D discrete wavelet transform (DWT) of the input data X using the wname wavelet. dwt2 returns the approximation coefficients matrix cA and detail coefficients matrices cH, cV, and cD (horizontal, vertical, and diagonal, respectively).
- `[U,S,V] = svd(A,'econ')` produces an economy-size decomposition of m-by-n matrix A. The economy-size decomposition removes extra rows or columns of zeros from the diagonal matrix of singular values, S, along with the columns in either U or V that multiply those zeros in the expression A = U*S*V'. Removing these zeros and columns can improve execution time and reduce storage requirements without compromising the accuracy of the decomposition.
- `[V,D] = eig(A)` returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors, so that A*V = V*D.
- `diag(V)` returns a square diagonal matrix with the elements of vector V on the main diagonal.
- `fitctree(X,Y)` returns a fitted binary classification decision tree based on the input variables contained in matrix X and output Y. The returned binary tree splits branching nodes based on the values of a column of X.
- `fitcsvm(X,Y)` returns an SVM classifier trained using the predictors in the matrix X and the class labels in vector Y for one-class or two-class classification.
- `crossval(fun,X)` performs 10-fold cross-validation for the function fun, applied to the data in X. The rows of X correspond to observations, and the columns of X correspond to variables.
- `kfoldLoss(cvmodel)` returns the cross-validation loss of cvmodel.

Appendix B MATLAB Code

```
function dcData = dcwavelet(dcfile)
    [m,n] = size(dcfile);
    pxl = sqrt(m);
    nw = m/4;
    dcData = zeros(nw,n);
    for k = 1:n
        X = im2double(reshape(dcfile(:,k),pxl,pxl));
        [~,cH,cV,~]=dwt2(X,'haar');
        codcH1 = rescale(abs(cH));
        codcV1 = rescale(abs(cV));
        codedge = codcH1+codcV1;
        dcData(:,k) = reshape(codedge,nw,1);
    end
end
```

Listing 1: Function of the Haar wavelet transformation

```
clear; clc; close all
[train_images_ori, train_labels] = mnist_parse('train-images.idx3-ubyte', 'train-labels.idx1-ubyte');
[test_images_ori, test_labels] = mnist_parse('t10k-images.idx3-ubyte', 't10k-labels.idx1-ubyte');
train_images = reshape(train_images_ori,[784 60000]);
test_images = reshape(test_images_ori,[784 10000]);
train_wave = dcwavelet(train_images);
test_wave = dcwavelet(test_images);

[U,S,V] = svd(train_wave,'econ');
PCA = S*V';

% pick 0, 1, 2
feature = 50;
Zeros = [];
Ones = [];
Twos = [];
for i = 1:10000
    if train_labels(i) == 2
        Twos(:,end+1) = PCA(1:feature,i);
    elseif train_labels(i) == 1
        Ones(:,end+1) = PCA(1:feature,i);
    elseif train_labels(i) == 0
        Zeros(:,end+1) = PCA(1:feature,i);
    end
end
size = min(size(Zeros,2),min(size(Ones,2),size(Twos,2)));

%classify 0 and 1
Ones = Ones(:,1:size);
Zeros = Zeros(:,1:size);
m_zero = mean(Zeros,2);
m_one = mean(Ones,2);

Sw = 0;
for k = 1:size
    Sw = Sw + (Zeros(:,k) - m_zero)*(Zeros(:,k) - m_zero)';
end
for k = 1:size
    Sw = Sw + (Ones(:,k) - m_one)*(Ones(:,k) - m_one)';
end
Sb = (m_zero-m_one)*(m_zero-m_one)';

[V2, D] = eig(Sb,Sw);
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);
vZero = w'*Zeros;
vOne = w'*Ones;
```

Listing 2: LDA

```
if mean(vZero)>mean(vOne)
    w = -w;
    vZero = -vZero;
    vOne = -vOne;
end

sortOnes = sort(vOne);
sortZeros = sort(vZero);
t1 = length(sortZeros);
t2 = 1;
while sortZeros(t1)>sortOnes(t2)
    t1 = t1-1;
    t2 = t2+1;
end
threshold = (sortOnes(t1) + sortZeros(t2))/2;

U = U(:,1:feature);
test_proj = U'*test_wave; % PCA projection

test_1_0_ori = [];
test_1_0 = [];
label_1_0 = [];
for i = 1:1000
    if test_labels(i) == 1 || test_labels(i) == 0
        test_1_0_ori(:,end+1) = test_images(:,i);
        test_1_0(:,end+1) = test_proj(:,i);
        label_1_0(:,end+1) = test_labels(i);
    end
end

val_1_0 = w'*test_1_0;
ResVec = (val_1_0>threshold); %zeros = 0, ones = 1
err = abs(ResVec - label_1_0);
errNum = sum(err);
sucRate = 1 - errNum/211;

%classify 0 and 2
Twos = Twos(:,1:size);
m_two = mean(Twos,2);

Sw2 = 0;
for k = 1:size
    Sw2 = Sw2 + (Zeros(:,k) - m_zero)*(Zeros(:,k) - m_zero)';
end
for k = 1:size
    Sw2 = Sw2 + (Twos(:,k) - m_two)*(Twos(:,k) - m_two)';
end
Sb2 = (m_zero-m_two)*(m_zero-m_two)';
```

Listing 3: LDA

```
[V3, D2] = eig(Sb2,Sw2);
[lambda, ind] = max(abs(diag(D2)));
w2 = V3(:,ind);
w2 = w2/norm(w2,2);

vZero2 = w2'*Zeros;
vTwo2 = w2'*Twos;

if mean(vZero2)>mean(vTwo2)
    w2 = -w2;
    vZero2 = -vZero2;
    vTwo2 = -vTwo2;
end

sortTwos2 = sort(vTwo2);
sortZeros2 = sort(vZero2);
t1 = length(sortZeros2);
t2 = 1;
while sortZeros2(t1)>sortTwos2(t2)
    t1 = t1-1;
    t2 = t2+1;
end
threshold2 = (sortTwos2(t1) + sortZeros2(t2))/2;

test_2_0_ori = [];
test_2_0 = [];
label_2_0 = [];
for i = 1:1000
    if test_labels(i) == 2 || test_labels(i) == 0
        test_2_0_ori(:,end+1) = test_images(:,i);
        test_2_0(:,end+1) = test_proj(:,i);
        label_2_0(:,end+1) = test_labels(i);
    end
end
val_2_0 = w2'*test_2_0;

%zeros = 0, twos = 2
ResVec2 = []
for j = 1:201
    if val_2_0(j)>threshold2
        ResVec2(j) = 2
    else
        ResVec2(j) = 0
    end
end

err2 = abs(ResVec2 - label_2_0);
errNum2 = 0;
for j = 1:size(err2)
    if err2(j) > 0
        errNum2 = errNum2+1;
    end
end
```

Listing 4: LDA

```
sucRate2 = 1 - errNum2/201;

%classify three numbers
Sw_3 = Sw;
for k = 1:size
    Sw_3 = Sw_3 + (Twos(:,k) - m_two)*(Twos(:,k) - m_two)';
end

m_all =  mean([Zeros Ones Twos],2);
Sb_3 = (m_zero-m_all)*(m_zero-m_all)'+(m_one-m_all)*(m_one-m_all)'+(m_two-m_all)*(m_two-m_all)';

[V4, D3] = eig(Sb_3,Sw_3);
[lambda, ind] = max(abs(diag(D3)));
w3 = V4(:,ind);
w3 = w3/norm(w3,2);

vZero3 = w3'*Zeros;
vOne3 = w3'*Ones;
vTwo3 = w3'*Twos;

%% decision tree
tree=fitctree(train_wave',train_labels,'CrossVal','on');
treeError = kfoldLoss(tree)

zero_one  = [Zeros Ones]
zero_two  = [Twos Ones]
zero_one_label  = [zeros([1 991]) ones([1 991])]
zero_two_label  = [zeros([1 991]) 2*ones([1 991])]

tree_1_0=fitctree(zero_one', zero_one_label,'CrossVal','on');
tree_2_0=fitctree(zero_two', zero_two_label,'CrossVal','on');
treeError_1_0 = kfoldLoss(tree_1_0)
treeError_2_0 = kfoldLoss(tree_2_0)

%% SVM
SVM_1_0 = fitcsvm(zero_one',zero_one_label);
CV_1_0 = crossval(SVM_1_0);
SVMLoss_1_0 = kfoldLoss(CV_1_0)

SVM_2_0 = fitcsvm(zero_two',zero_two_label);
CV_2_0 = crossval(SVM_2_0);
SVMLoss_2_0 = kfoldLoss(CV_2_0)
```

Listing 5: LDA