Identify the image data

Zhiyang Cheng

March 10, 2021

1 Introduction and Overview

This assignment we are attempted to train the image data and use different methods to identify the image data. The image data is ranging from digits 0 to 9. We can download the MNIST data set (both training and test sets and labels): http://yann.lecun.com/exdb/mnist/. To load the MNIST data into MATLAB some starter codes are given in the minist parse.m.
After the data are projected into PCA space, I will build a classifier to identify individual digits in the training set. By building a linear classifier (LDA), I will try to identify 2 random numbers and 2 random numbers and figure out which two digits in the data set are most easy/difficult to separate. At the end, compare the performance between LDA, SVM and decision trees on the hardest and easiest pair of digits to separate.

2 Theoretical Background

2.1 The Singular Value Decomposition - SVD

The Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices.

$$M = U\Sigma V^* \tag{1}$$

The singular value decomposition of an m× n complex matrix M is a factorization of the form $U\Sigma V^*$.
U is an $m \times m$ complex unitary matrix, Σ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and V is an $n \times n$ complex unitary matrix.

2.2 Linear Discriminant Analysis - LDA

LDA is a method to find a linear combination of features that characterizes or separates two or more classes of objects or events. In our case, we are separating different digits from the given image data. Suppose we are classifying group 1 and group 2 with their means as u1,u2, then the between-class scatter matrix is defined as:

$$S_B = (u2 - u1)(u2 - u1)^T \tag{2}$$

If we have more than two groups, the between-class scatter matrix which is S_B and the within-class scatter matrix which is Sw are defined as below:

$$S_B = \sum_{j=1}^{N}(u_j - u)(u_j - u)^T S_W = (u2 - u1)(u2 - u1)^T \tag{3}$$

$$S_W = \sum_{j=1}^{N}\sum_{x}(x - u_j)(x - u_j)^T \tag{4}$$

Our goal is to find w:

$$w = argmax \frac{w^T S_B w}{w^T S_W w} \tag{5}$$

We can achieve our goal by using the equations above since:

$$S_B w = \lambda S_W w \tag{6}$$

LDA works when the measurements made on independent variables for each observation are continuous quantities. When dealing with categorical independent variables, the equivalent technique is discriminant correspondence analysis

2.3 Support Vector Machines

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM maps training examples to points in space so as to maximise the width of the gap between the two categories.

2.4 Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a ”test” on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules

3 Algorithm Implementation and Development

3.1 Setting up

We can first Load the data from the given website using the given code minist parse.m. Then We got the training data set using the first 8 lines of code. We then subtract the row-wise mean of the training data set using the mean function. We then use the matlab command svd to get the the U, S, V matrix. After getting the result, we know that each column of U is a eigen vector, S is the singular value and V' is the linear coefficient. We then do the low rank approximation by setting up threshold.

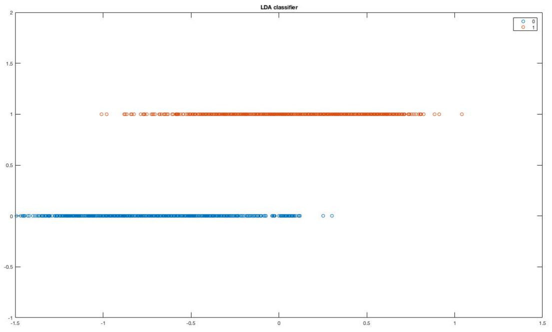
3.2 Analyze

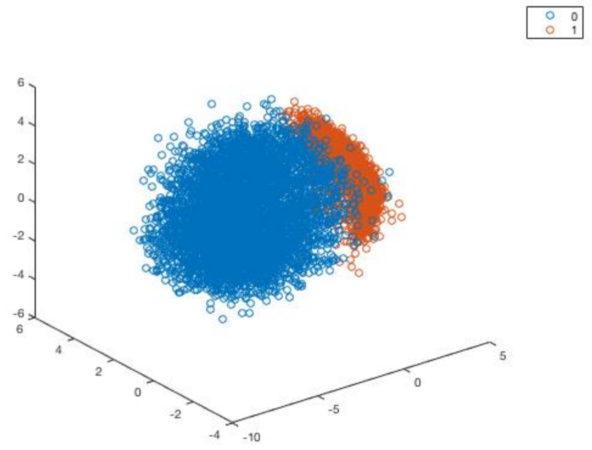
We then train LDA identifier using training data set.For LDA model, we can construct matrix of each digits. We then can get Sw within class covariance and construct Sb between class covariance. We then perform Singular Value Decomposition of Sw. After building up the identifier, we then write a function to check how well the model does by comparing the data from the prediction and the testing data. We then train SVM models using training dataset. For the SVM modle, we use fitcsvm command and for the decision tree model, we use fitctree command. And lastly calculate how well does they predict the result for the groups [0,1] and [5,8]. [0,1] is the easiest group to identify using LDA and [5,8] is the hardest group to identify using LDA.

4 Computational Results

4.1 Separating 0 and 1

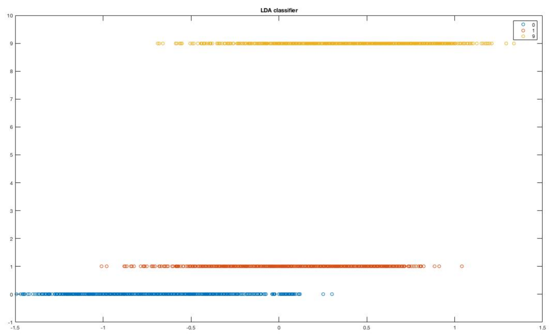
Based on the graph below, we are using the LDA method to separate 0 and 1. It shows the projected result of each digit. The accuracy to identify 0 and 1 is around 0.9986.

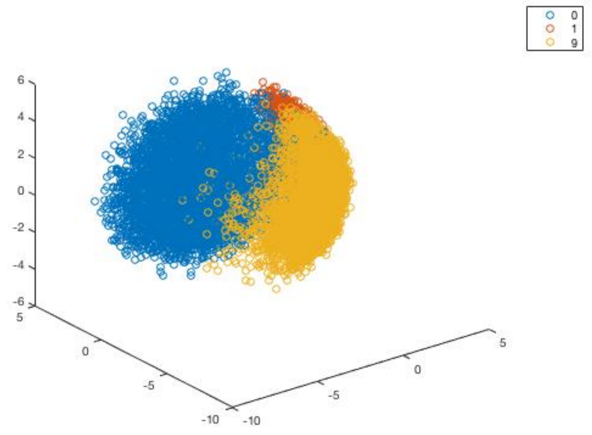




4.2 Separating 0, 1 and 9

Based on the graph below, we are using the LDA method to separate 0, 1, and 9. It shows the projected result of each digit. The accuracy to identify 0 and 9 is around 0.9864.





Based on table below, there are three principal component. We compared all of the resulting accuracy and getting the result that [0,1] is the easiest to identify and [5,8] is the hardest to identify.

	digit1	digit2	accuracy
1	0	1	99.86
2	0	9	98.64
3	0	7	99.45
4	1	4	99.48
5	5	8	93.36

Table 1: Accuracy to separate two digits in the data set

5 Summary and Conclusions

For [0,1] what we got as the easiest to separate using LDA, SVM's accuracy is around 99.98 and the decision tree got around 99.99.For [5,8] what we got as the hardest to separate using LDA, SVM's accuracy is around 75.36 and the decision tree method got around has 93.31. For this assignment, we learned how to build up a identifier and compare LDA, SVM and decision trees. While the accuracy being about the same for the easiest compared pairs, the SVM and decision tress definitely take longer time. I think LDA works the best for our case here.

6 References

https://en.wikipedia.org/wiki/Linear$_discriminant_analysis$
$https://en.wikipedia.org/wiki/Singular_value_decomposition$
$https://en.wikipedia.org/wiki/Support-vector_machine$
$https://en.wikipedia.org/wiki/Decision_tree: :text=A$

Appendix A MATLAB Functions

- `[U,S,V] = svd(x)` performs a singular value decomposition of matrix x
- `fitctree(X,Y)` returns a fitted binary classification decision tree based on the input variables contained in matrix X and output Y
- `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.

Appendix B MATLAB Code

```
clear all; close all; clc;
load fisheriris

[train_image, train_label] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
train_image = im2double(reshape(train_image, size(train_image,1)*size(train_image,2), []).');
train_label = im2double(train_label);
train_image = train_image';

[test_image,  test_label] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
test_image = im2double(reshape(test_image, size(test_image,1)*size(test_image,2), []).');
test_label = im2double(test_label);
test_image = test_image';


meanFort = mean(train_image,2);
train_image = double(train_image)-repmat(mn,1,length(train_image));

[U, S, V ] = svd(train_image, 'econ');

en = 0;
result = sum(diag(S));
thres = 0.8;
r = 0;

while en < thres
    r = r + 1;
    en = en + S(r,r)/result;
end

rank = r;

train_image = (U(:, 1:rank))'*train_image;
test_image = (U(:, 1:rank))'*test_image;

lamd = diag(S).^2;

for i = [5,8]
    Projection_idx = train_image(:, find(train_label == i));
    plot3(Projection_idx(1,:), Projection_idx(2,:), Projection_idx(3,:),...
          'o', 'DisplayName', num2str(i));
    hold on
end
legend show


sc = get(groot, 'ScreenSize');


X = train_image;
T = test_image;
```

```
dimen = size(train_image,1);
s1 = zeros(dimen);
s2 = zeros(dimen);
trainsize = size(train_image, 2);
testsize = size(test_image, 2);
Mu = mean(train_image, 2);

for i = [0,1]

    mask = (train_label ==  i);
    x = X(:, mask);
    ni = size(x, 2);
    pi = ni / trainsize;
    mu_i = mean(x, 2);

    Si = (x - repmat(mu_i, [1,ni]))*(x - repmat(mu_i, [1,ni]))';

    s1 = s1 + Si ;
    s2 = s2 + (mu_i - Mu) * (mu_i - Mu)';
end

M = pinv(s1) * s2;
[U, D, V] = svd(M);

G2 = U(:,1:rank);
Y2 = G2' * X;

2dfig = figure('Name', '2-D Plot');
set(2dfig,[60 60 sc(3)-120 sc(4) - 140]);

for number = [0,1]

    mask = (train_label ==  number);
    a = Y2(1,mask);
    b = Y2(2,mask);

    d = [a'; b'];
    plot(d, 1*number*ones(size(d)),'o',...
        'DisplayName', num2str(number)); hold on

end
legend show

%% accuracy

function [accu] = classifyNN(test_data, train_data, test_label, train_label)

train_size = size(train_data, 2);
test_size = size(test_data, 2);
c = zeros(test_size, 1);

parfor test_digit = 1:1:test_size

    test_mat = repmat(test_data(:, test_digit), [1,train_size]);
    distance = sum(abs(test_mat - train_data).^2);
    [M,I] = min(distance);
    if train_label(I) == test_label(test_digit)
        c(test_digit) = c(test_digit) + 1;
    end
end

accu = double(sum(c)) / test_size;
end

n1 = 0;
n2 = 1;

Y = G2' * X;
Y_t = G2'* T;

train_n= Y(:, find(train_label == n1|train_label ==n2));
test_n = Y_t(:, find(test_label == n1 |test_label ==n2));

accuracy = classifyNN(test_n, train_n,...
    test_label(find(test_label == n1 |test_label ==n2)), ...
    train_label(find(train_label == n1 |train_label ==n2)));

%% SVM
feat = 50;
z0 = [];
o1 = [];
t2 = [];
for i = 1:10000
    if train_labels(i) == 2
        t2(:,end+1) = PCA(1:feature,i);
    elseif train_labels(i) == 1
        o1(:,end+1) = PCA(1:feature,i);
    elseif train_labels(i) == 0
        z0(:,end+1) = PCA(1:feature,i);
end end
size = min(size(z0,2),min(size(o1,2),size(t2,2)));
zeroone  = [z0 o1]
zerotwo  = [t2 o1]
svm1 = fitcsvm(zeroone',zero1one);
cv1 = crossval(svm1);
svma1 = kfoldLoss(CV_1_0)
svm2 = fitcsvm(zerotwo',zero2two);
cv2 = crossval(svm2);
svma2 = kfoldLoss(cv2)

%% decision tree
twave = dcwavelet(train_images);
tree=fitctree(twave',train_labels,'CrossVal','on');
terror = kfoldLoss(tree)
label1  = [zeros([1 991]) ones([1 991])]
label2  = [zeros([1 991]) 2*ones([1 991])]
tree1=fitctree(zero_one', label1,'CrossVal','on');
tree2=fitctree(zero_two', label2,'CrossVal','on');
```

```
terror1 = kfoldLoss(tree1)
terror2 = kfoldLoss(tree2)
```