AMATH 482 Homework 4

Jiatian Xie

March 11, 2021

Abstract

This paper aims to classify digits. We are able to download the MNIST data set(both training and test sets and labels). In order to classify the digit label of training data set, I firstly perform an SVD analysis of the digit images to see the principal components of it. Then, I am able to have the data projected into PCA spaces and build several classifiers to identify digits in the training set. At last, I will test my identification of digits and also compare the accuracy of different classifiers.

1 Introduction and Overview

The MNIST database is a big database containing handwritten digits that is commonly used for training different image processing systems. It has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed size image. There are two parts of this project:

1.1 SVD analysis of the digit images

I am supposed to do an SVD analysis of the digit images and find out the the interpretation of the **U**, Σ, **V**.

1.2 Build classifiers to identify the digits in training set

In this section, since we are able to project the data into PCA space, we need to set up several different classifiers and verify the accuracy of each of classifiers. The classifiers I build are as follows:

1. Linear Classifier(LDA) for two digits
2. LDA for three digits
3. SVM (support vector machines)
4. Decision tree classifiers

2 Theoretical Background

2.1 Singular Value Decomposition(SVD) Anaylsis

The first concept used in this problem is the Singular Value Decomposition(SVD). The SVD is a matrix factorization of a given matrix A and it comes from the eigenvalue decomposition. If A is an $n \times m$ matrix, then we may write A as a product of three factors:

$$A = U\Sigma V^* \quad , \tag{1}$$

where U is an orthogonal $n \times n$ matrix, V is an orthogonal $m \times m$ matrix, V^* is the transpose of V, and Σ is an $n \times m$ matrix that has all zeros except for its diagonal entries, which are non-negative real numbers. If σ_{ij} is the i, j entry of Σ, then $\sigma_{ij} = 0$ unless $i = j$ and $\sigma_{ii} = \sigma_i \geq 0$. The σ_i are the "singular values" and the columns of u and v are respectively the right and left singular vectors. Based on U, Σ, V matrices, we are able to find out the principal components by seeing the singular values.

2.2 Linear Discriminant Analysis (LDA)

LDA is to determine the difference between two objects. In particular, we want to project two or more data sets onto new bases which maximizes the distance between the inter-class data while minimizing the intra-class data.

For example, when we have two objects to be classified. Consider the right subspace to project on. First, we will calculate the means of each of them and denote them μ_1 and μ_2. We then define the between-class scatter matrix that is the variance within each group:

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$$

Then define the within-class scatter matrix

$$S_w = \sum \sum (x - \mu_j)(x - \mu_j)^T$$

Then, we are able to find a vector **w** when the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem, that is:

$$S_B w = \lambda S_w w$$

Therefore, we are able to use `eig()` in Matlab to get the most reasonable vector w. Then, find out a cut-off(threshold) between two objects.

The logic is similar for linear classifiers of more than two groups. Only change the scatter matrices:

$$S_B = \sum (\mu_j - \mu)(\mu_j - \mu)^T$$
$$S_\omega = \sum \sum (x - \mu_j)(x - \mu_j)^T$$

The process of finding **w** is the same as above.

3 Algorithm Implementation and Development

Firstly, I extracted both training data, test data and their labels in Matlab. Then, I found the dimension of the data set was three. Therefore, I reshaped the training data into 60000*784, where 60000 is the number of the modes(images) and 784(= 28*28) represents the data in each mode. Also, do the same initialization for test data.

Then, since we need to find out the principal components of training dataset, I calculate U, S, V of the training data. Actually, I am hesitating about if we need to combine test data when doing SVD, but I think we only care about the training data set in the first part. I also need the number of modes that are necessary for good image construction. That is, find out the rank of the digit space. I use `nnz(S)` to get the number of nonzero element of all singular value. Now, I am able to look at the singular values to see the weight of each principal component. Also, after projecting three selected V-modes colored by their digit label, I can see if we can separate different digits well.

3.1 Linear Classifier(LDA)

Then, we are going to implement LDA for 2 digits. The process is stated in the previous section. Since we have performed the SVD analysis and now, I can have the training data projected into PCA space by `S*V'`. We can then do to build a classifier for two digits:

1. Find all data labeled by this two digits separately. Since the size of data of different digits are different, I just use the minimum size and adjust both of them into the same size. We need to do the process for both training set and test set.

2. determine the number of features(modes)
3. Then, I do the projection onto principal components. Divide the projection into two parts representing the projections of two digits.
4. calculate the scatter matrices by the theorem I illustrated above.(See Appendix B for details)
5. find **w** using **[V2, D] = eig(Sb,Sw)**
6. project the data set of two digits onto **w**
7. Since we need the consistency for whether digit 1 are above or below the threshold, we can reorder the datasets.

Until now, we get the threshold which can be used to distinguish two digits. Before this, we need a hidden-label vector of 0's and 1's (0 for digit 1 and 1 for digit 2) that can be used to check our performance. We can now perform PCA, then project onto the line we found from above. Then, we can check which images are digit 1 and which images are digit 2 by using the threshold. We can check errors using hiddenlabel and get the success rate.

The classifier for three digits is nearly the same as the classifier for two digits except the change of the scatter matrices.

3.2 Decision Tree Classifiers

The code for decision tree classifiers is direct except we need to consider the number of splits. What I am doing is to set the number as the half the mean number of splits from the default classification tree. I am using 10 here since there are 10 digits.

3.3 SVM support vector machines

It is obvious that we cannot directly use the training data. In other words, I need to rescale the data. That is, with the V-modes, the data should lies between -1 and 1, but SV is too large. Therefore, I need to multiply it by the largest singular value to bring the data into the interval [-1,1]. I would do this with the training set by `SV' ./ max(SV')`. Preform `fitcecoc()` to get Mdl and use `predict()` to get the test labels. At last, compare the test labels with the original labels.

4 Computational Results

4.1 SVD analysis

Based on the SVD analysis of training set, I got the following singular value spectrum 1. It is clear that the first singular value is largest. That is, the first mode is dominant. But we still have a hevay-tail distribution. And I used 15 as the number of modes because I can distinguish about 15 modes by my eyes.

U matrix derived from svd() represents the
See 2, it is a combination of random 3 columns of V-modes which is colored by the digit label. From the graph, we can see that the colors are nearly separate. In other words, we are able to distinguish different digits label in the training set.

4.2 The result of different classifier

Firstly, I chose digits 0 and 3 as my example. Follow the linear classifier, I found the accuracy is 0.9913. See the figure 3. It shows that the accuracy of this classifier is pretty high. Therefore, I perform the classifier for all 45 combinations of two digits. 6 and 9 are most easy to separate and I got 0.9958 accuracy. And I found 5 and 8 are the most difficult to separate although I still got 0.9478 for them.

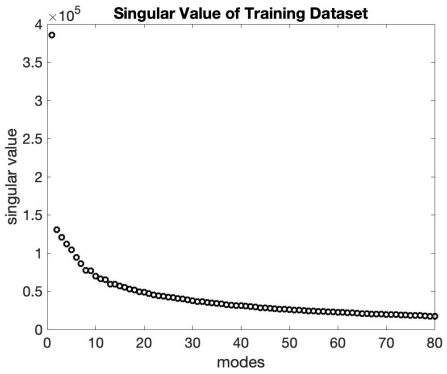

Figure 1: Here is a plot of the singular values that is gotten from svd of training data set

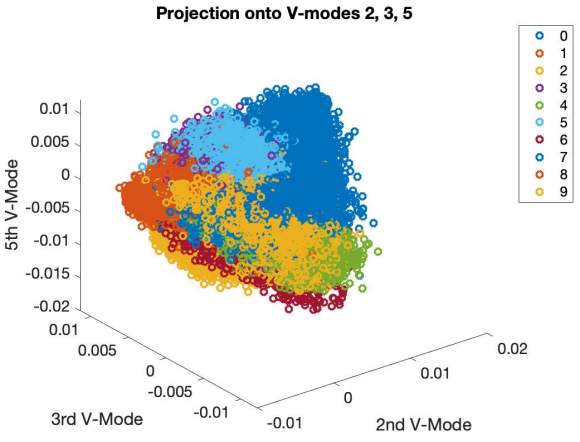

Figure 2: Here is a plot of the projection of 2, 3, 5 V-modes colored by their digit label

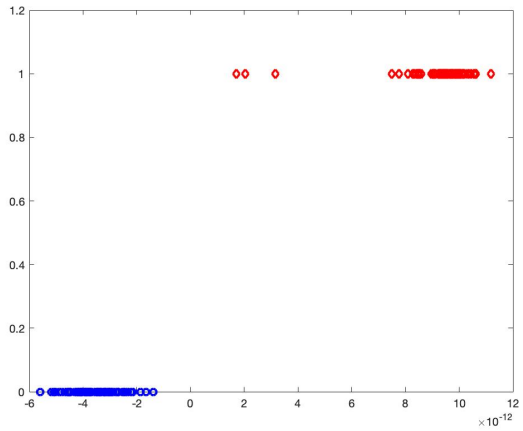

Figure 3: Here is a plot of the projection of 2 digits(0 and 3) by using my classifier

Then, I also create a classifier for three digits. I firstly choose digits 0, 1, and 2 and get an accuracy of 0.5435. It is not so high. Thus, I perform it to all combinations and find the best one is 1,5,7 with the accuracy of 0.9795. The worst case is 2,3,7 with the accuracy of 0.3687.

Next, using the decision tree classifiers, I got the accuracy of 0.6499 between all ten digits. And for SVM (support vector machines), I got 0.9063 which is a good result.

5 Summary and Conclusions

According to the whole project, when meeting the data sets containing different objects, we are able to use different classifiers to distinguish them. In this paper, we have 9 digits mixed in the training data. In order to classify them, I initially perform a PCA analysis to ensure the principal component and the rank of singular value. Then, I am able to build classifiers. Based on my findings, I find that the linear classifier for two digits is best performed because I do lots of adjustment to get a better threshold. So the performance of linear classifier for three digits is sometimes bad. For the decision tree classifiers, it does not play a good job in this data set. SVM, in general, is the best one in my opinion since it separate between all ten digits with an accuracy more than 0.9.

This project is so amazing that we can distinguish different objects in a data set. And we can see various classifiers may perform differently in different problems.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `[U,S,V] = svd(A)` performs a singular value decomposition of matrix A, such that `A = U*S*V'`.
- `k = find(X)` returns a vector containing the linear indices of each nonzero element in array X.
- `[row,col] = ind2sub(sz,ind)` returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices ind for a matrix of size sz. Here sz is a vector with two elements, where sz(1) specifies the number of rows and sz(2) specifies the number of columns.

- `tree = 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.
- `Mdl = 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.
- `ypred = predict(mdl,Xnew)` returns the predicted response values of the linear regression model mdl to the points in Xnew.

Appendix B MATLAB Code for SVD analysis

```
%% Load the data
[trainingdata, traingnd] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
trainingdata = double(reshape(trainingdata, size(trainingdata,1)*size(trainingdata,2), []));
traingnd = double(traingnd);

[testdata, testgnd] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
testdata = double(reshape(testdata, size(testdata,1)*size(testdata,2), []));
testgnd = double(testgnd);

%% SVD analysis
[U,S,V] = svd(trainingdata,'econ');

for label=0:9
    label_indices = find(traingnd == label);
    plot3(V(label_indices, 2), V(label_indices, 3), V(label_indices, 5),...
        'o', 'DisplayName', sprintf('%i',label), 'Linewidth', 2)
    hold on
end
xlabel('2nd V-Mode'), ylabel('3rd V-Mode'), zlabel('5th V-Mode')
title('Projection onto V-modes 2, 3, 5')
legend
set(gca,'Fontsize', 14)

%% plot S
figure(2)
plot(diag(S),'ko','Linewidth',2)
set(gca,'Fontsize',16,'Xlim',[0 80])

xlabel('modes'), ylabel('singular value')
title('Singular Value of Training Dataset')
```

Appendix C MATLAB Code for find the accuracy of two digits classifiers

Appendix D MATLAB Code for digitsThree.m

The method used to find the threshold value for three digits classifier. The process of find the accuracy is the same as two digits.

Appendix E MATLAB Code for SVM and decision tree classifiers

```
% choose 0 and 3 as an example sucrate = 0.9913
sucRate = ones(9,9);
for i = 0:9
    for j = i+1:9
        if(i >= j)
            break
        end
        label1 = find(traingnd == i);
        label2 = find(traingnd == j);
        min_size = min(size(label1),size(label2));
        label1 = label1(1:min_size);
        label2 = label2(1:min_size);
        data1 = trainingdata(:,label1);
        data2 = trainingdata(:,label2);


        testlabel1 = find(testgnd == i);
        testlabel2 = find(testgnd == j);
        min_size = min(size(testlabel1),size(testlabel2));
        testlabel1 = testlabel1(1:min_size);
        testlabel2 = testlabel2(1:min_size);
        testdata1 = testdata(:,testlabel1);
        testdata2 = testdata(:,testlabel2);


        % training
        [U_train,S_train,V_train,threshold_train,w_train,sort_train1,sort_train2] = dc_trainer(data1,da

        % test
        TestSet = [testdata1 testdata2];
        TestNum = size(TestSet,2);
        TestMat = U_train'*TestSet; % PCA projection
        pval = w_train'*TestMat;

        ResVec = (pval > threshold_train);

        % Os are correct and 1s are incorrect
        hiddenlabels = zeros(1,size(TestSet,2));
        hiddenlabels(size(TestSet,2)/2+1:end) = 1;
        err = abs(ResVec - hiddenlabels);
        errNum_svm = sum(err);
        sucRate(i+1,j+1) = 1 - errNum_svm/TestNum;
    end
end
```

```
function [U,S,V,threshold,w,sort1,sort2,sort3] = digitsThree(data1,data2,data3,feature)

    n1 = size(data1,2);
    n2 = size(data2,2);
    n3 = size(data3,3);
    [U,S,V] = svd([data1 data2 data3],'econ');
    datas = S*V';
    U = U(:,1:feature); % Add this in
    datas1 = datas(1:feature,1:n1);
    datas2 = datas(1:feature,n1+1:n1+n2);
    datas3 = datas(1:feature,n1+n2+1:n1+n2+n3);
    m1 = mean(datas1,2);
    m2 = mean(datas2,2);
    m3 = mean(datas3,2);

    Sw = 0;
    for k=1:n1
        Sw = Sw + (datas1(:,k)-m1)*(datas1(:,k)-m1)';
    end
    for k=1:n2
        Sw = Sw + (datas2(:,k)-m2)*(datas2(:,k)-m2)';
    end
    for k=1:n3
        Sw = Sw + (datas3(:,k)-m3)*(datas3(:,k)-m3)';
    end

    m = mean([m1;m2;m3]);
    Sb = (m1-m)*(m1-m).' + (m2-m)*(m2-m).'+(m2-m)*(m2-m).' ;

    [V2,D] = eig(Sb,Sw);
    [lambda,ind] = max(abs(diag(D)));
    w = V2(:,ind);
    w = w/norm(w,2);
    vdata1 = w'*datas1;
    vdata2 = w'*datas2;
    vdata3 = w'*datas3;

    % make sure that the order is data1, data2, data3
    if mean(vdata1)>mean(vdata2) && mean(vdata2)>mean(vdata3)
        w = -w;
        vdata1 = -vdata1;
        vdata2 = -vdata2;
        vdata3 = -vdata3;
    end


    % Don't need plotting here
    sort1 = sort(vdata1);
    sort2 = sort(vdata2);
    sort3 = sort(vdata3);
    t1 = length(sort1);
    t2 = length(sort2);
    t3 = 1;
    threshold = zeros(1,2);
    threshold(1) = (sort1(t1)+sort2(t2))/2;
    threshold(2) = (sort1(t2)+sort2(t3))/2;
```



```
end
```

```
%% decision tree classifiers

tree_train=fitctree(trainingdata.',traingnd,'MaxNumSplits',10,'CrossVal','on');
view(tree_train.Trained{1},'Mode','graph');
tree_Error = kfoldLoss(tree_train);
tree_accurancy = 1-tree_Error;

%% SVM classifier with training data, labels and test set
train_svm = (S*V' ./ max(S*V')).';
X = train_svm;
Y = traingnd;
```