AMATH 482 Homework 1

Zach Zlepper

Mar 7, 2021

Abstract

Using LDA, SVM and Decision Tree algorithms, we explore the ability of a computer to classify images correctly. In this paper, we will explore some of the fundamental mathematics behind these algorithms as well as implement our own version on the MNIST data set including using the SVD.

1 Introduction and Overview

Machine learning allows computers to recognize images. We are given the MNIST data set and are asked to classify the digits using the LDA algorithm we have learned in class as well as SVM and decision trees. These algorithms are the foundation of new AI systems which have shown promise to solve problems like self driving cars. Although these algorithms are not state of the art, they provide an easy lens to see the power of Machine learning through.

2 Theoretical Background

2.1 SVD and PCA

In order to get to Principle component analysis, we must first discuss the Singular Value Decomposition. The SVD is a Decomposition of a matrix A such that

$$\begin{gathered}\mathbf{A} = \mathbf{U\Sigma V}^* \text{ where} \\ \mathbf{U} \in \mathbb{C}^{m\times m} \text{ is unitary} \\ \mathbf{V} \in \mathbb{C}^{n\times n} \text{ is unitary} \\ \mathbf{\Sigma} \in \mathbb{R}^{m\times n} \text{ is diagonal}\end{gathered}$$

The SVD has a geometric interpretation as well which is as follows:

- Multiplying by V^* rotates our vectors
- Multiplying by Σ stretches our vectors
- Multiplying by U rotates our vectors again

For our problem the V matrix represents how the numbers are represented in the PCA basis, the U matrix are the principle components, and the S Matrix hold the energy of the Principle components.

So how do we compute such a decomposition? The answers is we find the eigenvalues and eigenvectors of the matrix A^TA and AA^T After this eigen value calculation shown in HW3, we can see the U and V matrices contain the eigenvectors of AA^T and A^TA respectively with Σ^2 as the eigenvalues of both AA^T and A^TA. The square root of the eigenvalues of these equations produces the singular values which are an output in the SVD.

We will use the SVD as the backbone algorithm for the PCA which allows high dimensional data to be deconstructed to lower dimensional representations. In addition, the larger the singular values the more energy. This fact when used in PCA allows us to see which principle components are more present in the system. We will add some statistical theory such as the the covariance matrix to display the property that the SVD have an uncorrelated components. In hw 3 we showed that the components that the SVD yields are uncorrelated. $\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T$, where C_X is a square $m \times m$ is zero.This calculation shows that the covariance matrix of SVD components yields an uncorrelated decomposition of the data becasue the off diagnal values are 0. Not only are they uncorrelated but they also can be broken down to find the most important component to the signal by using the singular values.

2.2 LDA

The basic idea is we are taking a high dimensional data set and we are projecting onto a linear subspace and then seeing the overlap between the separate components. In particular, the μ_i of the data should be spread out for a good LDA.We want to find a projection that maximizes the distance between inter-class data while minimizing the intra-class data. To find the between spread Matix $\mathbb{S}_B$, we use the equations for n classes

$$\mathbb{S}_B = \sum_{j=1}^{j=n} (\mu_j - \mu)(\mu_j - \mu)^T \tag{1}$$

where μ is the overall mean and μ_j is the mean of each subgroup for n $\geq$ 3 groups. For two classes, we would use

$$\mathbb{S}_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tag{2}$$

To find the within class scatter we use the equations

$$\mathbb{S}_w \sum_{j=1}^{j=n} \sum_{x} (x - \mu)(x - \mu)^T \tag{3}$$

. We then need to find a w such that

$$w = argmax \frac{w^T S_B w}{w^T S_w w} \tag{4}$$

Then we can solve the eigenvalue problem and find the w that corresponds to the largest λ. Then we have to multiply our data by w'. This projects our data onto w but then we have to calculate our error. We do this by finding a cutoff that approximately has the same number of errors on either side of the threshold line.

$$S_B w = \lambda S_w w \tag{5}$$

2.3 SVM and Decision Trees

We will also use SVM and decision tree algorithms in this paper. These algorithms were state of the art unit 2014.

3 Algorithm Implementation and Development

We will walk through the algorithm for calculating the LDA. In order for use to understand what our data looks we can plot the V modes Figure 1 I plotted the 3, 5, and 7 values in our PCA space. For LDA, we are going to project these digits onto a line in algorithm 1.

For our 2 case LDA, we can use our home made function dcTrainer. However, with 3 classes we must either change the code to accommodate 3 or more class as described above or use fitcdiscr and a kfoldloss approach to see the accuracy. Under the hood, these algorithms are using the same mathematics that we review previously.

We can also see by plotting the singular values and measuring the energy that we only need about 154 modes to capture 95% of the energy. This allows use significant reduction in modes from 784 to have a similarly accurate classification. See Figure 2.

For the SVM and Decision Tree Algorithms, we will use prepacked functions that implement them. With these prebuilt functions, I used kfoldloss to calculate the error.

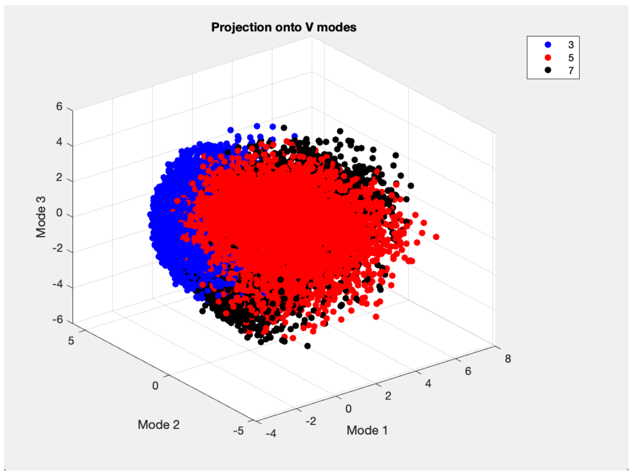


Figure 1: Spacial representation of Data 3 5 7

Algorithm 1: LDA Algorithm

```
Import MNIST data set
Reshape the data into vectors from Matrices
Standardize the Data around the mean
Figure out subset of Data such as types of digits to classify
Calculate $S_w$
Calculate $S_B$
Find w with the eig Command
Project onto w by multiplying data by w'
Calculate the Threshold for the Data
Plot
```

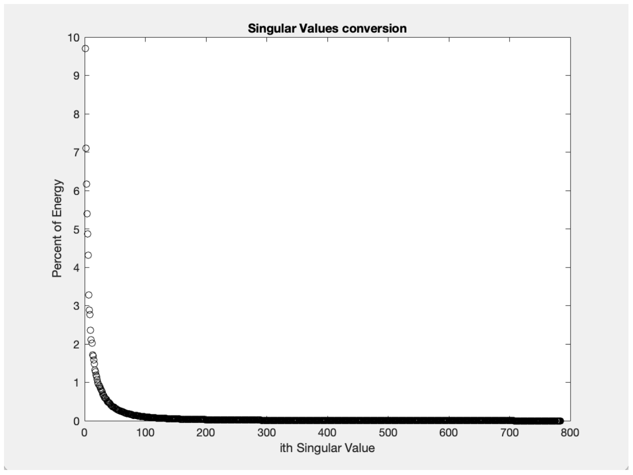


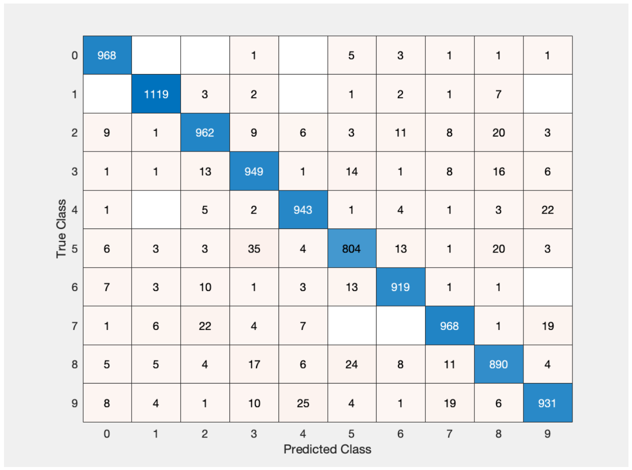
Figure 2: Singular Value and Energy

4 Computational Results

Below is our error based on our algorithms. We can see from our 10 Digit SVM chart in Figure 3 that our easiest to classify were 1 and 0 and most difficult were 3 and 5. We can especially see the descrepency with the table 1 and the 2 Digit LDA difference for 0 and 1 as well as 3 and 5.

	Algorithm	Success Rate
1	2 Digit LDA (1,2)	.9875
2	3 Digit LDA (1,2,3)	.9325
3	2 Digit Tree Classification (1,2)	.9839
4	2 Digit SVM (1,2)	.9985
5	10 Digit SVM	.967
6	10 Digit Tree	.75 (200 branches)
7	2 Digit LDA (0,1)	.99
8	2 Tree LDA (0,1)	.99
9	2 Digit SVM (0,1)	.99
10	2 Digit LDA (3,5)	.96
11	2 Tree LDA (3,5)	.99
12	2 Digit SVM (3,5)	.99

Table 1: Errors based on algorithm.



True Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	968			1		5	3	1	1	1
1		1119	3	2		1	2	1	7	
2	9	1	962	9	6	3	11	8	20	3
3	1	1	13	949	1	14	1	8	16	6
4	1		5	2	943	1	4	1	3	22
5	6	3	3	35	4	804	13	1	20	3
6	7	3	10	1	3	13	919	1	1	
7	1	6	22	4	7			968	1	19
8	5	5	4	17	6	24	8	11	890	4
9	8	4	1	10	25	4	1	19	6	931

Figure 3: Errors for the 10 digit SVM

5 Summary and Conclusions

We have seen the power of LDA, SVM, and Decision Tree algorithms on two digit, three digit, and ten digit classification. We get really good accuracy with a large data set around 60000. Not only did we implement these algorithms but we discussed some of the mathematics behind the LDA as well.

Appendix A MATLAB Functions

"diag(X)" : Returns a column vector of the diagonal values of a matrix.
"Mdl = fitcecoc(Tbl,ResponseVarName)" : returned a fitted multiclass models for support vector machines or other classifiers. "Mdl = fitcdiscr(Tbl,ResponseVarName)" : returns a fitted discriminent classifier.
"tree = fitctree(Tbl,ResponseVarName)": Fit binary decision tree for multiclass classification
"max(A)" : Returns maximum values in vector A
"mean(A)" : Returns the mean of the vector A.
"[M,I] = max(A)" : Returns the maximum value M and the index I of the array.
"label = predict(Mdl,X)" Predicts labels using discriminant analysis classification model Mdl
"size(X)" : returns dimensions of the matrix X
"svd(X, 'econ') : computes the SVD of the matrix X.
"zeros(A,B,C)" : Creates a matrix filled with zeros of size "A x B x C".

Appendix B MATLAB Code

```
%%
clear all; clc; close all;
%%
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
%%
%Two Digit Test Set
[imagesTest, labelsTest] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
conversionTest = zeros(784, 10000);
for c = 1:10000
    conversionTest(:, c) = im2double(reshape(imagesTest(:,:,c),784,1));
end

LDAtest = conversionTest;
conversion = zeros(784, 60000);
for c = 1:60000
    conversion(:, c) = im2double(reshape(images(:,:,c),784,1));
end

LDAtrain = conversion;
%standardizing conversion
[m,n]=size(conversion); % compute data size
mn=mean(conversion,2); % compute mean for each row
conversion = conversion-repmat(mn,1,n);% subtract mean

%standardizing converiontest
%standardizing conversion
[m,n]=size(conversionTest); % compute data size
mn=mean(conversionTest,2); % compute mean for each row
conversionTest = conversionTest-repmat(mn,1,n);% subtract mean

%%
[U, S, V] = svd(conversion, 'econ');
lambda = diag(S).^2;
figure();
plot(1:784, lambda/sum(lambda)*100,'ko');
title('Singular Values conversion');
xlabel('ith Singular Value')
ylabel('Percent of Energy')
%%
total = 0;
all = sum(lambda);
for c = 1:154
    total = lambda(c)/all + total;
end
%154 features gives 95% of energy


projection_training = U(:, 1:154)'*conversion;
projection_test = U(:, 1:154)'* conversionTest;
```

Listing 1: Example code from external file.

```
%%

Proj = S*V';

figure
scatter3(Proj(1, labels ==7), Proj(2, labels ==7),Proj(3, labels ==7), 'b', 'filled')
hold on
scatter3(Proj(1, labels ==3), Proj(2, labels ==3),Proj(3, labels ==3), 'r', 'filled')
scatter3(Proj(1, labels ==5), Proj(2, labels ==5),Proj(3, labels ==5), 'k', 'filled')
legend('3', '5', '7')
title('Projection onto V modes')
xlabel('Mode 1')
ylabel('Mode 2')
zlabel('Mode 3')



%%

feature = 154;
dog = LDAtrain(:, labels == 0);
cat = LDAtrain(:, labels == 1);

[UD,SD,VD,threshold,w,sortdog,sortcat] = dc_trainer(dog,cat,feature);


%Two Digit Success Rates
TestNum = size(LDAtest(:, labelsTest == 0 | labelsTest == 1),2); % wavelet transform
TestMat = UD'*LDAtest(:, labelsTest == 0 | labelsTest == 1); % PCA projection
TwoDigitTestLables  = labelsTest(labelsTest == 0 | labelsTest == 1);
TwoDigitTestLables(TwoDigitTestLables == 0) = 0;
TwoDigitTestLables(TwoDigitTestLables == 1) = 1;

pval = w'*TestMat;
ResVec = (pval > threshold);
err = abs(ResVec - TwoDigitTestLables');
errNum = sum(err);
LDA2rrror = 1 - errNum/TestNum;
%%
%Three Digit Classification

Sample = projection_test(:, labelsTest == 1 | labelsTest == 2| labelsTest == 3);
X = projection_training(:, labels == 1 | labels == 2 | labels == 3);
Y = labels(labels == 1 | labels == 2 | labels == 3);
Mdl = fitcdiscr(X',Y','discrimType', 'diagLinear');
cMdl = crossval(Mdl);
classErrorLDA3 = kfoldLoss(cMdl);
LDA3Error = 1-classErrorLDA3;



%% TREE two
X = projection_training(:, labels == 1 | labels == 0 );
Y = labels(labels == 1 | labels == 0 );
tree = fitctree(X', Y', 'MaxNumSPlits', 200, 'CrossVal', 'on');
classErrorTree = kfoldLoss(tree);
TreeError = 1- classErrorTree;
```

```
%% SVM Two
TestNum = size(projection_training(:, labels == 1 | labels == 0 ),2);
X = projection_training(1:154, labels == 1 | labels == 0 );
```

```
%% SVM TEN

TestNum = size(projection_training(:, :),2);
X = projection_training(:, :)./ max(projection_training(:, :));
Y = labels(:);
Mdl = fitcecoc(X',Y');
scaled_test = projection_test(:,:)./max(projection_test(:, :));
label_approx = predict(Mdl, scaled_test');
hidden_labels = labelsTest(:);

err = abs(label_approx - hidden_labels);
errNum = sum(err);
sucRate10SVM = 1 - errNum/TestNum;
%%
figure();
confusionchart(hidden_labels,label_approx)

%% Tree 10
X = projection_training(:, : );
Y = labels(:);
tree = fitctree(X', Y', 'MaxNumSPlits', 200, 'CrossVal', 'on');
classErrorTree = kfoldLoss(tree);
TreeError10 = 1- classErrorTree;
```

Listing 3: Matlab code for this assignment.