Amath 482 Homework 4: Classifying Digits

Yilin Cao

March 10, 2021

Abstract

The objective of this project is to identify a given set of handwritten digits. We will start by performing SVD analysis of the MINIST dataset and project the data into PCA space. In addition, we will apply LDA, SVM, and decision tree classifier to identify the digits in the dataset as well as the hardest and easiest pair of digits to separate.

I. Introduction and Overview

In this project, we will extract data from the MINIST database of handwritten digits, which contains 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.

II. Theoretical Background

1. Singular Value Decomposition

Singular Value decomposition, defined as $A=U\Sigma V^T$, splits a matrix into a number of constitutive components, which will perform tranformation that stretches/compresses and rotates as given set of vectors. The columns of U and V are orthonormal and the matrix Σ is a diagonal matrix with positive real entries sorted from largest to smallest. By performing an SVD on matrix A, it first applied a rotation with V^T then stretches by the diagonal matrix Σ. Lastly, it rotates back using the orthonormal matrix U. The resulted matrix from SVD would reduce redundancy and find the maximal variance signal. By the properties of SVD, every matrix A has an SVD if the proper basis of the range and domain, U and V, are chosen. The resulted matrix will then be useful for simplified calculations.

2. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a generalization of Fisher's linear discriminant. It is an algorithm that finds a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. In order to simplify the procedure by LDA, we have made the following assumptions. The algorithm assumes a constant varaince for each variable. In addition, the data is Gaussian or normal, which means that when plotting the data will result in a bell curve.

3. Multiclass LDA

In the case where there are more than two classes, the analysis used in the derivation of the Fisher discriminant can be extended to find a subspace which appears to contain all of the class variability.

III. Algorithm Implementation and Development

SVD.m:

- load images labels
- select a rand graph to show to check the data
- reshape the data
- implement SVD reshaped images to get U S V matrices
- show 10 components in U matrices
- plot singular value spectrum
- plot 3D by V in column 2,3,5

minist_2digits.m：

- load images labels for training
- reshape the data
- select 2 digits
- implement SVD reshaped images to get U S V matrices
- plot singular value spectrum
- plot 3D by V in column 1,2,3
- plot 2D by V in column 1,2
- LDA classifies 2 digits
- load images labels for testing
- reshape the data
- select 2 digits
- classify 2 digits in test data
- calculate accuracy

minist_3digits.m:

- load images labels for training
- reshape the data
- select 3 digits
- use SVD_LDA to get model for classifying 3 digits
- load images labels for testing
- reshape the data
- select 3 digits
- classify 3 digits in test data
- calculate accuracy

digits2by2.m：

- load images labels for training
- reshape the data
- load images labels for testing
- reshape the data
- select number 0,1,2,3,4,5,6,7,8,9
- use SVD_LDA to get model for classifying 2 digits, 45 models together
    - classify 2 digits in test data by every model
    - calculate accuracy
- find the hardest(easiest) classified 2 digits

SVM_digits.m:

- load images labels for training
- reshape the data
- load images labels for testing
- reshape the data
- train SVM model to classify 10 digits
- classify 10 digits in test data
- calculate accuracy

decision_tree_digits.m:

- load images labels for training
- reshape the data
- load images labels for testing
- reshape the data
- train decision_tree model to classify 10 digits
- classify 10 digits in test data
- calculate accuracy

Compare_3_methods_hardest(easiest).m:

- load images labels for training
- reshape the data
- select 2 digits
- load images labels for testing
- reshape the data
- select 2 digits

- use SVD_LDA to get model for classifying 2 digits
    - classify 2 digits in test data

    - calculate accuracy

- train SVM model to classify 2 digits
    - classify 2 digits in test data
    - calculate accuracy

- train decision_tree model to classify 2 digits
    - classify 2 digits in test data
    - calculate accuracy

SVD_LDA.m:

- select 2 input digits in input images
- use SVD,LDA to get model for classifying
- return model

SVD_LDA_predict.m:

- if number of model is 1, return the result of classifying 2 digits
- if number of model is not 1, return the result of classifying k digits

IV. Computational Results

Analysis of MINIST dataset:

1. SVD analysis
   [U,S,V] = svd(reshape_images,'econ');
2. Singular Value Spectrum

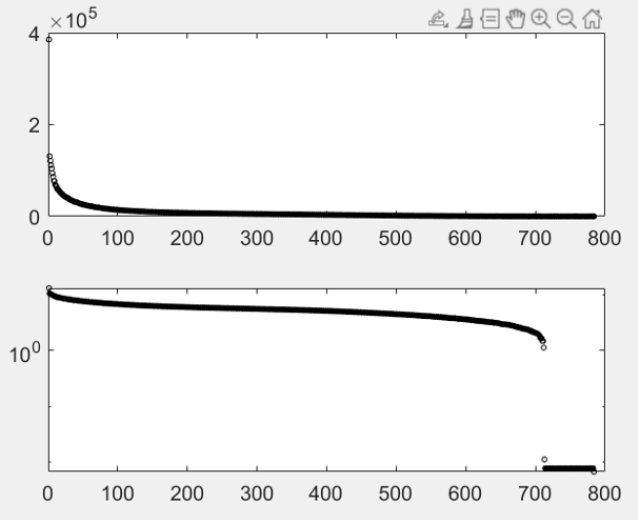


Figure 1. Singular Value Spectrum

The first mode is dominant. However, we can also see that there isn't a sharp drop-off in the singular values. That is, we have a heavy-tail distribution, meaning there is still information in the later modes.

3. Interpretation of the U, Σ, and V matrices
   If [U,S,V] = svd(A,'econ') where U $\in C^{mxm}$ is unitary, U $\in C^{nxn}$ is unitary, $\Sigma \in R^{mxn}$ is diagonal, then
4. Project onto three selected V-modes (columns) colored by their digit label.

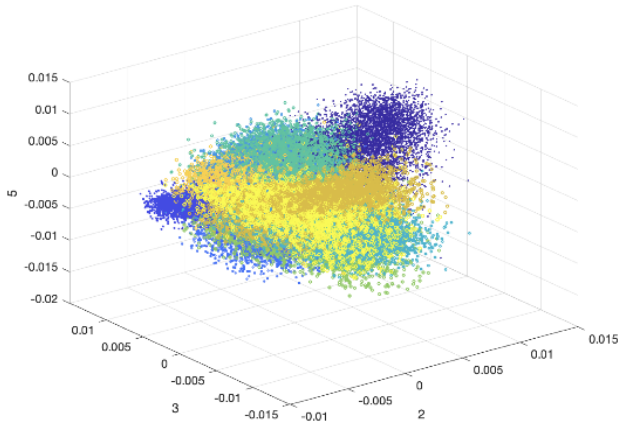


Figure 2. 3D plot of projection onto Mode 2,3,5

Build a classifier to identify individual digits in the training set:

- Pick two digits with LDA classifier
  Pick digits 0 and 1, and the accuracy is 0.99953.

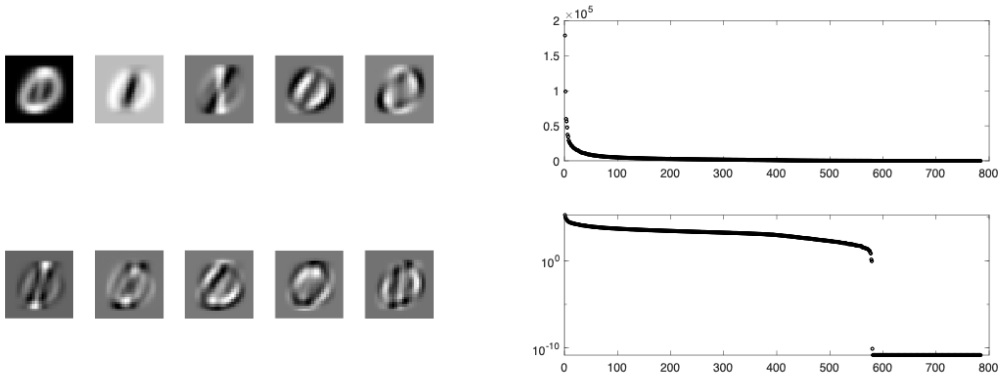
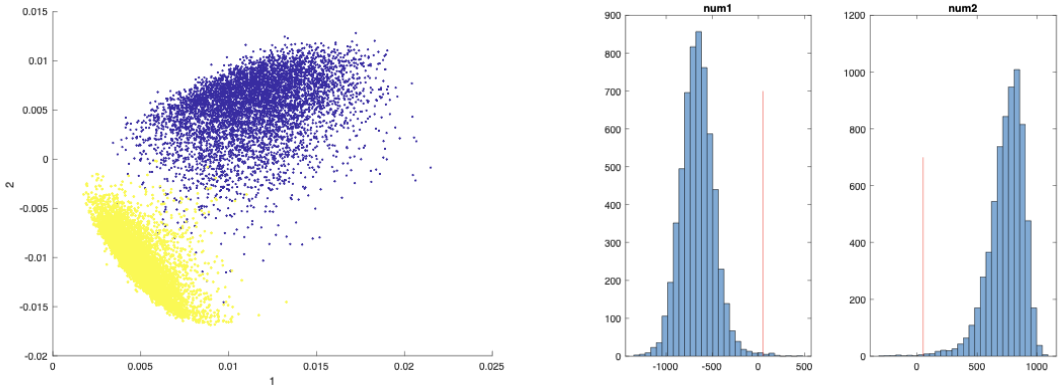


Figure 3.



- Pick three digits with LDA classifier
  Pick digits 0, 1, 6, and the accuracy is 0.98471.
- Which two digits in the data set appear to be the most difficult to separate?
  Digits 0 and 1 appear to be the most difficult to separate.
  Maximum accuracy is 0.99953.
- Which two digits in the data set are most easy to separate?
  Digits 4 and 9 appear to be the most difficult to separate.
  Maximum accuracy is 0.93973.
- SVM (support vector machines) and decision tree classifier
  The accuracy of SVM classifier is 0.9069.
  The accuracy of decision tree classifier is 0.8777.
- Compare the performance between LDA, SVM and decision trees on the hardest and easiest pair of digits to separate.
  The hardest pair of digits is digit 4 and 9.
  Performance:
    - The accuracy of LDA classifier is 0.93973
    - The accuracy of SVM classifier is 0.96585
    - The accuracy of decision tree classifier is 0.95229

The easiest pair of digits is digit 0 and 1.
Performance:

- The accuracy of LDA classifier is 0.99953
- The accuracy of SVM classifier is 0.99858
- The accuracy of decision tree classifier is 0.99622

V. Summary and Conclusions

Appendix A. MATLAB functions

abs(X) : calculates the absolute value of the input element. If input in complex, then returns the complex magnitude. This was used to normalize our dataset.

svd(A) : performs a singular value decomposition to the $x\ y$ matrix.

Appendix B. MATLAB codes

```
close all;
```

```
clc;
clear;
%%  load images labels
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
% [test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[row,col,num]=size(images);
%% select a rand graph to show
figure;
rand_image_inx = round(rand*num);
rand_image = images(:,:,rand_image_inx);
imshow(rand_image)
%% reshape
reshape_images = zeros(row*col,num);
for i=1:num
   reshape_images(:,i) = reshape(images(:,:,i),row*col,1);
end
%%  SVD
[U,S,V] = svd(reshape_images,'econ');
figure;
for k =1:10
   subplot(2,5,k);
   ut1 = reshape(U(:,k),28,28);
   ut2 = rescale(ut1);
   imshow(ut2);
end
%% Sngular value spectrum
figure;
subplot(2,1,1);
plot(diag(S),'ko','Linewidth',0.5,'MarkerSize',3);
set(gca,'Fontsize',12,'Xlim',[0 800]);
subplot(2,1,2);
semilogy(diag(S),'ko','Linewidth',0.5,'MarkerSize',3);
set(gca,'Fontsize',12,'Xlim',[0 800]);
%% plot 3D
figure;
XX = V(:,2);
YY = V(:,3);
ZZ = V(:,5);
clabels = labels+1;
scatter3(XX,YY,ZZ,clabels,clabels)
xlabel('2');
ylabel('3');
zlabel('5');
```

```
close all;
clc;
clear;
%% load images labels
```

```
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
% [test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[row,col,num]=size(images);
%% reshape
reshape_images = zeros(row*col,num);
for i=1:num
   reshape_images(:,i) = reshape(images(:,:,i),row*col,1);
end
%% select 2 digits
select_num = [0,1];
ii = 1;
jj = 1;
for i=1:num
   if labels(i)== select_num(1);
      select_num1_images(:,ii) = reshape_images(:,i);
      ii = ii+1;
   end
   if labels(i)== select_num(2);
      select_num2_images(:,jj) = reshape_images(:,i);
      jj = jj+1;
   end
end
%% SVD
select_num_images = [select_num1_images select_num2_images];
[U,S,V] = svd(select_num_images,'econ');
figure;
for k =1:10
   subplot(2,5,k);
   ut1 = reshape(U(:,k),28,28);
   ut2 = rescale(ut1);
   imshow(ut2);
end
%% singular value spectrum
figure;
subplot(2,1,1);
plot(diag(S),'ko','Linewidth',0.5,'MarkerSize',3);
set(gca,'Fontsize',12,'Xlim',[0 800]);
subplot(2,1,2);
semilogy(diag(S),'ko','Linewidth',0.5,'MarkerSize',3);
set(gca,'Fontsize',12,'Xlim',[0 800]);
%% plot 3D
figure;
XX = V(:,1);
YY = V(:,2);
ZZ = V(:,3);
[~,num1_len] = size(select_num1_images);
[~,num2_len] = size(select_num2_images);
labels_num1 = ones(num1_len,1)*( select_num(1)+1);
labels_num2 = ones(num2_len,1)*( select_num(2)+1);
clabels = [labels_num1;labels_num2];
scatter3(XX,YY,ZZ,clabels,clabels)
xlabel('1');
ylabel('2');
zlabel('3');
%% plot 2D
figure;
scatter(XX,YY,clabels,clabels);
xlabel('1');
ylabel('2');
%%
feature = 20;
digits = S*V';
n1 = size(select_num1_images,2);
n2 = size(select_num2_images,2);
num1 = digits(1:feature,1:n1);
```

```
num2 = digits(1:feature,n1+1:n1+n2);
m1 = mean(num1,2);
m2 = mean(num2,2);
Sw = 0; % within class variances
for k = 1:n1
   Sw = Sw + (num1(:,k) - m1)*(num1(:,k) - m1)';
end
for k = 1:n2
   Sw = Sw + (num2(:,k) - m2)*(num2(:,k) - m2)';
end
Sb = (m1-m2)*(m1-m2)'; % between class
[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);
vnum1 = w'*num1;
vnum2 = w'*num2;
if mean(vnum1) > mean(vnum2)
   w = -w;
   vnum1 = -vnum1;
   vnum2 = -vnum2;
end
%%
sortnum1 = sort(vnum1);
sortnum2 = sort(vnum2);
t1 = length(sortnum1);
t2 = 1;
while sortnum1(t1) > sortnum2(t2)
   t1 = t1 - 1;
   t2 = t2 + 1;
end
threshold = (sortnum1(t1) + sortnum2(t2))/2;

figure;
subplot(1,2,1)
histogram(sortnum1,30);hold on;plot([threshold threshold],[0,700],'r')
title('num1');
subplot(1,2,2)
histogram(sortnum2,30);hold on;plot([threshold threshold],[0,700],'r')
title('num2');

[test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[trow,tcol,tnum]=size(test_images);
%% reshape
treshape_images = zeros(trow*tcol,tnum);
for i=1:tnum
   treshape_images(:,i) = reshape(test_images(:,:,i),trow*tcol,1);
end
%% select 2 digits
ii = 1;
jj = 1;
for i=1:tnum
   if test_labels(i)== select_num(1)
      tselect_num1_images(:,ii) = treshape_images(:,i);
      ii = ii+1;
   end
   if test_labels(i)== select_num(2)
      tselect_num2_images(:,jj) = treshape_images(:,i);
      jj = jj+1;
   end
end
%%
tselect_num_images = [tselect_num1_images tselect_num2_images];
[~,tnum1_len] = size(tselect_num1_images);
[~,tnum2_len] = size(tselect_num2_images);
```

```
tlabels_num1 = ones(tnum1_len,1)*( select_num(1));
tlabels_num2 = ones(tnum2_len,1)*( select_num(2));
tlabels = [tlabels_num1;tlabels_num2];
%%
TestNum = size(tselect_num_images,2);
for t = 1:TestNum
   TestMat = U'*tselect_num_images(:,t);
   pval = w'*TestMat(1:feature,:);
   if pval<threshold
      ResVec = select_num(1);
   else
      ResVec = select_num(2);
   end
   results(t) = ResVec;
end
results = results';
err = 0;
TestNum = numel(tlabels);
for num_i=1:TestNum
   if results(num_i)==tlabels(num_i)
      ;
   else
      err = err +1;
   end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
```

```
close all;
clc;
clear;
%% load images labels
```

```
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
% [test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[row,col,num]=size(images);
%% reshape
reshape_images = zeros(row*col,num);
for i=1:num
   reshape_images(:,i) = reshape(images(:,:,i),row*col,1);
end
%% select 2 digits
select_num = [0,1,6];
k = size(select_num,2);
model = {};
for i=1:k-1
   for j =i+1:k
      model{i,j}= SVD_LDA(reshape_images,labels,[select_num(i) select_num(j)]);
   end
end

[test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[trow,tcol,tnum]=size(test_images);
%% reshape
treshape_images = zeros(trow*tcol,tnum);
for i=1:tnum
   treshape_images(:,i) = reshape(test_images(:,:,i),trow*tcol,1);
end
%% select 2 digits
ii = 1;
jj = 1;
kk = 1;
for i=1:tnum
   if test_labels(i)==select_num(1)
      tselect_num1_images(:,ii) = treshape_images(:,i);
      ii = ii+1;
   end
   if test_labels(i)==select_num(2)
      tselect_num2_images(:,jj) = treshape_images(:,i);
      jj = jj+1;
   end
   if test_labels(i)==select_num(3)
      tselect_num3_images(:,kk) = treshape_images(:,i);
      kk = kk+1;
   end
end
%%
tselect_num_images = [tselect_num1_images tselect_num2_images tselect_num3_images];
[~,tnum1_len] = size(tselect_num1_images);
[~,tnum2_len] = size(tselect_num2_images);
[~,tnum3_len] = size(tselect_num3_images);
tlabels_num1 = ones(tnum1_len,1)*(select_num(1));
tlabels_num2 = ones(tnum2_len,1)*(select_num(2));
tlabels_num3 = ones(tnum3_len,1)*(select_num(3));
tlabels = [tlabels_num1;tlabels_num2;tlabels_num3];
results = SVD_LDA_predict(tselect_num_images,model);
err = 0;
TestNum = numel(tlabels);
for i=1:TestNum
   if results(i)==tlabels(i)
      ;
   else
      err = err +1;
   end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
```

```
close all;
clc;
clear;
%% load images labels
```

```
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
% [test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[row,col,num]=size(images);
%% reshape
reshape_images = zeros(row*col,num);
for i=1:num
   reshape_images(:,i) = reshape(images(:,:,i),row*col,1);
end
%%
[test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[trow,tcol,tnum]=size(test_images);
%% reshape
treshape_images = zeros(trow*tcol,tnum);
for i=1:tnum
   treshape_images(:,i) = reshape(test_images(:,:,i),trow*tcol,1);
end
%%
select_num = [0,1,2,3,4,5,6,7,8,9];
k = size(select_num,2);
for i=1:k-1
   for j =i+1:k
      model= SVD_LDA(reshape_images,labels,[select_num(i),select_num(j)]);
      % select 2 digits,clear selected number
      clear tselect_num1_images;
      clear tselect_num2_images;
      clear tlabels;
      ii = 1;
      jj = 1;
      for indx=1:tnum
         if test_labels(indx)==select_num(i)
            tselect_num1_images(:,ii) = treshape_images(:,indx);
            ii = ii+1;
         end
         if test_labels(indx)==select_num(j)
            tselect_num2_images(:,jj) = treshape_images(:,indx);
            jj = jj+1;
         end
      end
      %%
      tselect_num_images = [tselect_num1_images tselect_num2_images];
      [~,tnum1_len] = size(tselect_num1_images);
      [~,tnum2_len] = size(tselect_num2_images);
      tlabels_num1 = ones(tnum1_len,1)*select_num(i);
      tlabels_num2 = ones(tnum2_len,1)*select_num(j);
      tlabels = [tlabels_num1;tlabels_num2];
      %%
      results = SVD_LDA_predict(tselect_num_images,{model});
      err = 0;
      TestNum = numel(tlabels);
      for num_i=1:TestNum
         if results(num_i)==tlabels(num_i)
            ;
         else
            err = err +1;
         end
      end
      sucRate = 1 - err/TestNum;
      disp(['Accuracy is :',num2str(sucRate)]);
      sucRateall(i,j)=sucRate;
   end
end
max_acc = 0;
max_row = 1;
max_col = 1;
min_acc = 1;
```

```
min_row = 1;
min_col = 1;
for i=1:k-1
   for j =i+1:k
      if max_acc<sucRateall(i,j)
         max_acc = sucRateall(i,j);
         max_row =select_num(i);
         max_col = select_num(j);
      end
      if min_acc>sucRateall(i,j)
         min_acc = sucRateall(i,j);
         min_row =select_num(i);
         min_col = select_num(j);
      end
   end
end
disp(['Maximum Accuracy is :',num2str(max_acc)]);
disp(['Number ',num2str(max_row),' and ',num2str(max_col)]);
disp(['Minimum Accuracy is :',num2str(min_acc)]);
disp(['Number ',num2str(min_row),' and ',num2str(min_col)]);
```

```
close all;
clear;
clc;
%% load images labels
```

```
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
% [test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[row,col,num]=size(images);
%% reshape
reshape_images = zeros(row*col,num);
for i=1:num
   reshape_images(:,i) = reshape(images(:,:,i),row*col,1);
end
%%
[test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[trow,tcol,tnum]=size(test_images);
%% reshape
treshape_images = zeros(trow*tcol,tnum);
for i=1:tnum
   treshape_images(:,i) = reshape(test_images(:,:,i),trow*tcol,1);
end
%% SVM classifier with training data, labels and test set
%
tic;
tTree = templateTree('surrogate','on');
tEnsemble = templateEnsemble('GentleBoost',100,tTree);
options = statset('UseParallel',true);
Mdl = fitcecoc(reshape_images',labels,'Coding','onevsall','Learners',tEnsemble,...
            'Prior','uniform','Options',options);%,'NumBins',50
toc;
%
testLength = length(test_labels);
testResults = -1*ones(testLength,1);
parfor i=1:testLength
   testResults(i) = predict(Mdl,treshape_images(:,i)');
end
% Mdl = fitcecoc(reshape_images',labels);
% test_results = predict(Mdl,treshape_images);
err = 0;
TestNum = numel(test_labels);
for i=1:TestNum
   if testResults(i)==test_labels(i)
      ;
   else
      err = err +1;
   end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
```

```
close all;
clear;
clc;
%% load images labels
```

```
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
% [test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[row,col,num]=size(images);
%% reshape
reshape_images = zeros(row*col,num);
for i=1:num
   reshape_images(:,i) = reshape(images(:,:,i),row*col,1);
end
%%
[test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[trow,tcol,tnum]=size(test_images);
%% reshape
treshape_images = zeros(trow*tcol,tnum);
for i=1:tnum
   treshape_images(:,i) = reshape(test_images(:,:,i),trow*tcol,1);
end
%%
ctree=fitctree(reshape_images',labels);
% view(ctree);
results = predict(ctree,treshape_images');
err = 0;
TestNum = numel(test_labels);
for i=1:TestNum
   if results(i)==test_labels(i)
      ;
   else
      err = err +1;
   end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
```

```
close all;
clc;
clear;
%% LDA
```

```
% load images labels
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
% [test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[row,col,num]=size(images);
% reshape
reshape_images = zeros(row*col,num);
for i=1:num
   reshape_images(:,i) = reshape(images(:,:,i),row*col,1);
end
% select 2 digits
select_num = [0,1];
select_num1 = select_num(1);
select_num2 = select_num(2);
ii = 1;
jj = 1;
for i=1:num
   if labels(i)==select_num1
      select_num1_images(:,ii) = reshape_images(:,i);
      ii = ii+1;
   end
   if labels(i)==select_num2
      select_num2_images(:,jj) = reshape_images(:,i);
      jj = jj+1;
   end
end
%%
[~,num1_len] = size(select_num1_images);
[~,num2_len] = size(select_num2_images);
labels_num1 = ones(num1_len,1)*(select_num1);
labels_num2 = ones(num2_len,1)*(select_num2);
select_num_labels = [labels_num1;labels_num2];
select_num_images = [select_num1_images select_num2_images];
%%
select_num1 = select_num(1);
select_num2 = select_num(2);
ii = 1;
jj = 1;
for i=1:num
   if labels(i)==select_num1
      select_num1_images(:,ii) = reshape_images(:,i);
      ii = ii+1;
   end
   if labels(i)==select_num2
      select_num2_images(:,jj) = reshape_images(:,i);
      jj = jj+1;
   end
end
%%
[~,num1_len] = size(select_num1_images);
[~,num2_len] = size(select_num2_images);
labels_num1 = ones(num1_len,1)*(select_num1);
labels_num2 = ones(num2_len,1)*(select_num2);
select_num_labels = [labels_num1;labels_num2];
select_num_images = [select_num1_images select_num2_images];
%%
[test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[trow,tcol,tnum]=size(test_images);
%% reshape
treshape_images = zeros(trow*tcol,tnum);
for i=1:tnum
   treshape_images(:,i) = reshape(test_images(:,:,i),trow*tcol,1);
end
%% select 2 digits
ii = 1;
jj = 1;
```

```
for i=1:tnum
   if test_labels(i)==select_num1
      tselect_num1_images(:,ii) = treshape_images(:,i);
      ii = ii+1;
   end
   if test_labels(i)==select_num2
      tselect_num2_images(:,jj) = treshape_images(:,i);
      jj = jj+1;
   end
end
%%
tselect_num_images = [tselect_num1_images tselect_num2_images];
[~,tnum1_len] = size(tselect_num1_images);
[~,tnum2_len] = size(tselect_num2_images);
tlabels_num1 = ones(tnum1_len,1)*(select_num1);
tlabels_num2 = ones(tnum2_len,1)*(select_num2);
tlabels = [tlabels_num1;tlabels_num2];
%% SVD
[U,S,V] = svd(select_num_images,'econ');
%
feature = 20;
digits = S*V';
n1 = size(select_num1_images,2);
n2 = size(select_num2_images,2);
num1 = digits(1:feature,1:n1);
num2 = digits(1:feature,n1+1:n1+n2);
m1 = mean(num1,2);
m2 = mean(num2,2);
Sw = 0; % within class variances
for k = 1:n1
   Sw = Sw + (num1(:,k) - m1)*(num1(:,k) - m1)';
end
for k = 1:n2
   Sw = Sw + (num2(:,k) - m2)*(num2(:,k) - m2)';
end
Sb = (m1-m2)*(m1-m2)'; % between class
[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);
vnum1 = w'*num1;
vnum2 = w'*num2;
if mean(vnum1) > mean(vnum2)
   w = -w;
   vnum1 = -vnum1;
   vnum2 = -vnum2;
end
%
sortnum1 = sort(vnum1);
sortnum2 = sort(vnum2);
t1 = length(sortnum1);
t2 = 1;
while sortnum1(t1) > sortnum2(t2)
   t1 = t1 - 1;
   t2 = t2 + 1;
end
threshold = (sortnum1(t1) + sortnum2(t2))/2;
%
TestNum = size(tselect_num_images,2);
for t = 1:TestNum
   TestMat = U'*tselect_num_images(:,t);
   pval = w'*TestMat(1:feature,:);
   if pval<threshold
      ResVec = select_num(1);
   else
```

```
        ResVec = select_num(2);
    end
    results(t) = ResVec;
end

err = 0;
TestNum = numel(tlabels);
for num_i=1:TestNum
    if results(num_i)==tlabels(num_i)
        ;
    else
        err = err +1;
    end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
%% SVM classifier with training data, labels and test set
%
tic;
tTree = templateTree('surrogate','on');
tEnsemble = templateEnsemble('GentleBoost',100,tTree);
options = statset('UseParallel',true);
Mdl = fitcecoc(select_num_images',select_num_labels,'Coding','onevsall','Learners',tEnsemble,...
            'Prior','uniform','Options',options);%,'NumBins',50
toc;
%
testLength = length(tlabels);
testResults = -1*ones(testLength,1);
parfor i=1:testLength
    testResults(i) = predict(Mdl,tselect_num_images(:,i)');
end
% Mdl = fitcecoc(reshape_images',labels);
% test_results = predict(Mdl,treshape_images);
err = 0;
TestNum = numel(tlabels);
for i=1:TestNum
    if testResults(i)==tlabels(i)
        ;
    else
        err = err +1;
    end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
%
ctree=fitctree(select_num_images',select_num_labels);
% view(ctree);
results = predict(ctree,tselect_num_images');
err = 0;
TestNum = numel(tlabels);
for i=1:TestNum
    if results(i)==tlabels(i)
        ;
    else
        err = err +1;
    end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);


close all;
clc;
clear;
%% LDA
```

```
% load images labels
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
% [test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[row,col,num]=size(images);
% reshape
reshape_images = zeros(row*col,num);
for i=1:num
   reshape_images(:,i) = reshape(images(:,:,i),row*col,1);
end
% select 2 digits
select_num = [4,9|;
select_num1 = select_num(1);
select_num2 = select_num(2);
ii = 1;
jj = 1;
for i=1:num
   if labels(i)==select_num1
      select_num1_images(:,ii) = reshape_images(:,i);
      ii = ii+1;
   end
   if labels(i)==select_num2
      select_num2_images(:,jj) = reshape_images(:,i);
      jj = jj+1;
   end
end
%%
[~,num1_len] = size(select_num1_images);
[~,num2_len] = size(select_num2_images);
labels_num1 = ones(num1_len,1)*(select_num1);
labels_num2 = ones(num2_len,1)*(select_num2);
select_num_labels = [labels_num1;labels_num2];
select_num_images = [select_num1_images select_num2_images];
%%
[test_images, test_labels] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
[trow,tcol,tnum]=size(test_images);
% reshape
treshape_images = zeros(trow*tcol,tnum);
for i=1:tnum
   treshape_images(:,i) = reshape(test_images(:,:,i),trow*tcol,1);
end
% select 2 digits
ii = 1;
jj = 1;
for i=1:tnum
   if test_labels(i)==select_num1
      tselect_num1_images(:,ii) = treshape_images(:,i);
      ii = ii+1;
   end
   if test_labels(i)==select_num2
      tselect_num2_images(:,jj) = treshape_images(:,i);
      jj = jj+1;
   end
end
%%
tselect_num_images = [tselect_num1_images tselect_num2_images];
[~,tnum1_len] = size(tselect_num1_images);
[~,tnum2_len] = size(tselect_num2_images);
tlabels_num1 = ones(tnum1_len,1)*(select_num1);
tlabels_num2 = ones(tnum2_len,1)*(select_num2);
tlabels = [tlabels_num1;tlabels_num2];
%% SVD
[U,S,V] = svd(select_num_images,'econ');
%
feature = 20;
digits = S*V';
```

```
n1 = size(select_num1_images,2);
n2 = size(select_num2_images,2);
num1 = digits(1:feature,1:n1);
num2 = digits(1:feature,n1+1:n1+n2);
m1 = mean(num1,2);
m2 = mean(num2,2);
Sw = 0; % within class variances
for k = 1:n1
   Sw = Sw + (num1(:,k) - m1)*(num1(:,k) - m1)';
end
for k = 1:n2
   Sw = Sw + (num2(:,k) - m2)*(num2(:,k) - m2)';
end
Sb = (m1-m2)*(m1-m2)'; % between class
[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);
vnum1 = w'*num1;
vnum2 = w'*num2;
if mean(vnum1) > mean(vnum2)
   w = -w;
   vnum1 = -vnum1;
   vnum2 = -vnum2;
end
%
sortnum1 = sort(vnum1);
sortnum2 = sort(vnum2);
t1 = length(sortnum1);
t2 = 1;
while sortnum1(t1) > sortnum2(t2)
   t1 = t1 - 1;
   t2 = t2 + 1;
end
threshold = (sortnum1(t1) + sortnum2(t2))/2;
%
TestNum = size(tselect_num_images,2);
for t = 1:TestNum
   TestMat = U'*tselect_num_images(:,t);
   pval = w'*TestMat(1:feature,:);
   if pval<threshold
      ResVec = select_num(1);
   else
      ResVec = select_num(2);
   end
   results(t) = ResVec;
end

err = 0;
TestNum = numel(tlabels);
for num_i=1:TestNum
   if results(num_i)==tlabels(num_i)
      ;
   else
      err = err +1;
   end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
%% SVM classifier with training data, labels and test set
%
tic
tTree = templateTree('surrogate','on');
tEnsemble = templateEnsemble('GentleBoost',100,tTree);
options = statset('UseParallel',true);
```

```
Mdl = fitcecoc(select_num_images',select_num_labels,'Coding','onevsall','Learners',tEnsemble,...
            'Prior','uniform','Options',options);%,'NumBins',50
toc;
%
testLength = length(tlabels);
testResults = -1*ones(testLength,1);
parfor i=1:testLength
   testResults(i) = predict(Mdl,tselect_num_images(:,i)');
end
% Mdl = fitcecoc(reshape_images',labels);
% test_results = predict(Mdl,treshape_images);
err = 0;
TestNum = numel(tlabels);
for i=1:TestNum
   if testResults(i)==tlabels(i)
      ;
   else
      err = err +1;
   end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
%%
ctree=fitctree(select_num_images',select_num_labels);
% view(ctree);
results = predict(ctree,tselect_num_images');
err = 0;
TestNum = numel(tlabels);
for i=1:TestNum
   if results(i)==tlabels(i)
      ;
   else
      err = err +1;
   end
end
sucRate = 1 - err/TestNum;
disp(['Accuracy is :',num2str(sucRate)]);
```

```
function [model]=SVD_LDA(images,labels,input_num)
[~,num]=size(images);
%% select 2 digits
ii = 1;
```

```
jj = 1;
for i=1:num
   if labels(i)==input_num(1)
      select_num1_images(:,ii) = images(:,i);
      ii = ii+1;
   end
   if labels(i)==input_num(2)
      select_num2_images(:,jj) = images(:,i);
      jj = jj+1;
   end
end
%% SVD
select_num_images = [select_num1_images select_num2_images];
[U,S,V] = svd(select_num_images,'econ');
%%
feature = 20;
digits = S*V';
n1 = size(select_num1_images,2);
n2 = size(select_num2_images,2);
num1 = digits(1:feature,1:n1);
num2 = digits(1:feature,n1+1:n1+n2);
m1 = mean(num1,2);
m2 = mean(num2,2);
Sw = 0; % within class variances
for k = 1:n1
   Sw = Sw + (num1(:,k) - m1)*(num1(:,k) - m1)';
end
for k = 1:n2
   Sw = Sw + (num2(:,k) - m2)*(num2(:,k) - m2)';
end
Sb = (m1-m2)*(m1-m2)'; % between class
[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);
vnum1 = w'*num1;
vnum2 = w'*num2;
if mean(vnum1) > mean(vnum2)
   w = -w;
   vnum1 = -vnum1;
   vnum2 = -vnum2;
end
%%
sortnum1 = sort(vnum1);
sortnum2 = sort(vnum2);
t1 = length(sortnum1);
t2 = 1;
while sortnum1(t1) > sortnum2(t2)
   t1 = t1 - 1;
   t2 = t2 + 1;
end
threshold = (sortnum1(t1) + sortnum2(t2))/2;

model.U = U;
model.w = w;
model.threshold=threshold;
model.number = input_num;




function results=SVD_LDA_predict(images,model)
   feature = 20;
   [~,img_num] =size(images);
   [m,n] = size(model);
```

```
    if n==1
        for t = 1:img_num
            U = model{1,1}.U;
            w = model{1,1}.w;
            threshold = model{1,1}.threshold;
            select_num = model{1,1}.number;
            TestMat = U'*images(:,t);
            pval = w'*TestMat(1:feature,:);
            if pval<threshold
                ResVec = select_num(1);
            else
                ResVec = select_num(2);
            end
            results(t) = ResVec;
        end
    else
        for t = 1:img_num
            k = 1;
            for i=1:n-1
                for j = i+1:n
                    U = model{i,j}.U;
                    w = model{i,j}.w;
                    threshold = model{i,j}.threshold;
                    select_num = model{i,j}.number;
                    TestMat = U'*images(:,t);
                    pval = w'*TestMat(1:feature,:);
                    if pval<threshold
                        ResVec(k) = select_num(1);
                        k=k+1;
                    else
                        ResVec(k) = select_num(2);
                        k=k+1;
                    end
        %           ResVec(i) = (pval > threshold);
                end
            end
            results(t) = mode(ResVec);
        end
    end
results = results';
```