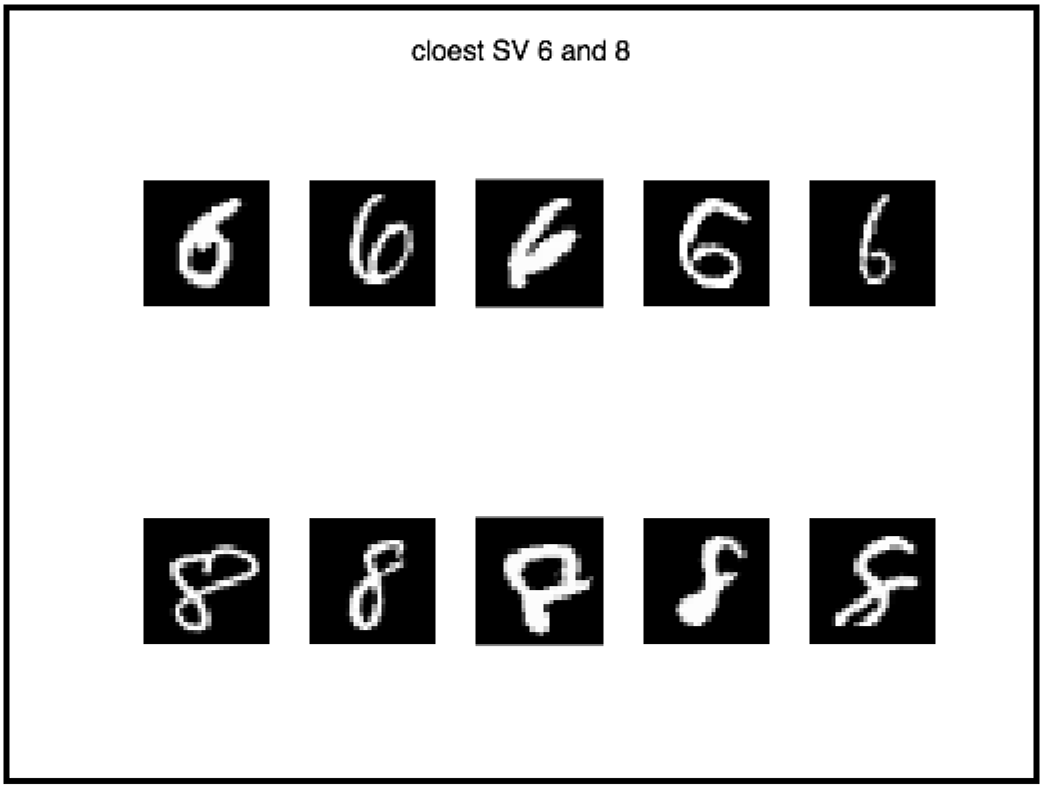
GaotongWu
A13809639
ECE175a
HW7

1. Two class SVM using linear kernels K(x, y) = xTy.

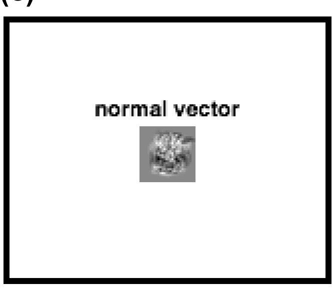
(a)

```
Accuracy = 100% (83/83) (classification)
```

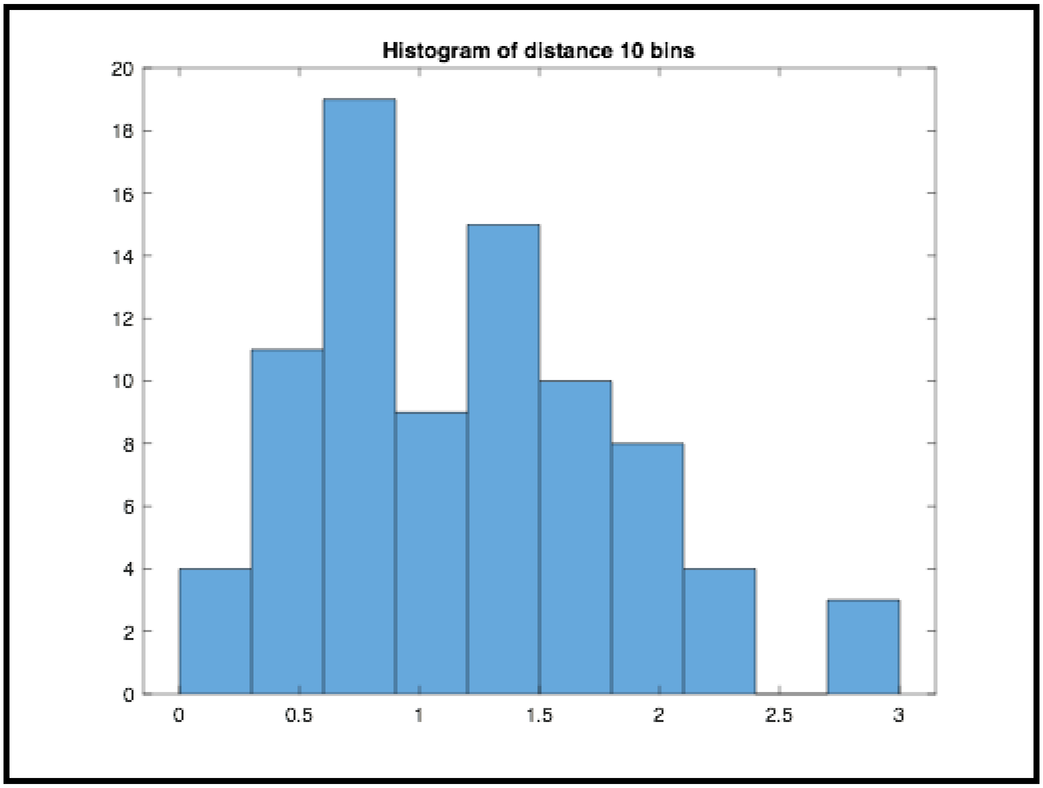
(b)



(c)

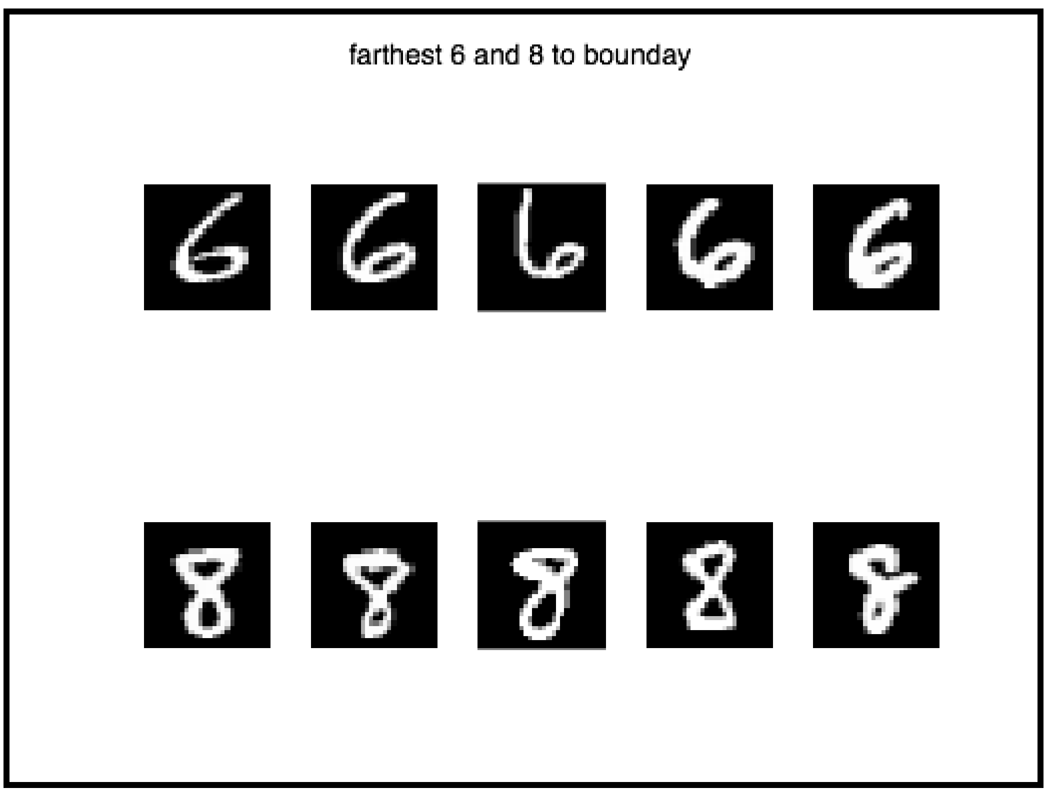


(d)



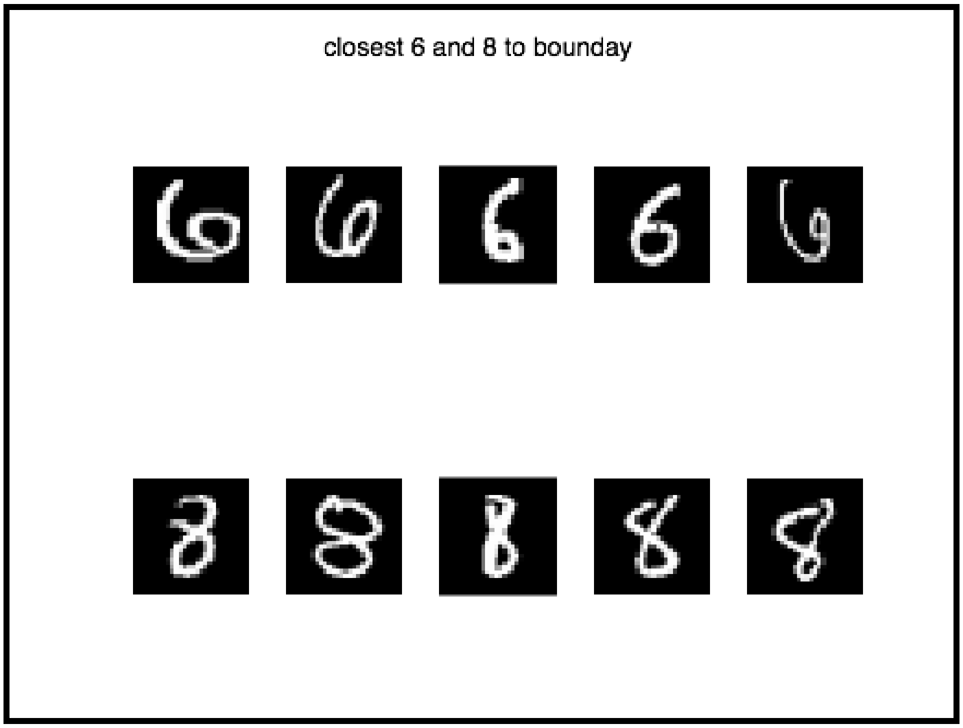
Histogram of distance 10 bins
20
18
16
14
12
10
8
6
4
2
0
0
0.5
1
1.5
2
2.5
3

(e)



farthest 6 and 8 to bounday

(f)



These digits look less like the digit 6 and 8 compared to the ones from (e); as images getting closer to the decision boundary, they are less confident to be that class.

2. Ten-digit class SVM using Gaussian kernel K(x, y) = exp(−γ||x − y||2), γ > 0.
(a)

```
highest_acc =

   95.6000


idx =

    16


best_c =

     2


best_gamma =

    0.0312
```

The best pair is c=2, gamma=0.0312, with the highest accuracy=95.6%

(b)

```
Accuracy = 95.6% (478/500) (classification)
```

Which means the classification error is 4.4%

3. Compare all the algorithms

(a)

- NN has an error rate of 0.094;
- Gaussian Classification has an error rate of 0.22 because we assume the covariance matrix to be an identity matrix which is too strict.
- Gaussian classification on PCA space has higher error rates than 0.22 on dimensions lower than 40; as dimensions increase, error rate drops to 0.22; but as we keep increasing, error rate goes up because of noise at high dimensions.
- SVM classification has an error rate of 0.044, which is the lowest.

(b)

1. **Nearest Neighbor:**

- Advantage: easy to realize, no need to train;
- Disadvantage: large computation since we need to compute the distances of each test sample to all other training samples.

2. **Gaussian Classification:**

- Advantage: can be used to estimate
- Disadvantage: there will be noise and redundant components in high dimensions which will increase the error rate.

3. **Gaussian classification on PCA space:**

- Advantage: can be used to eliminate noise in high dimensions thus reducing error rates;
- Disadvantage: there is a possibility that components with low eigenvalues might contain some useful information but we eliminate them.

4. **SVM classification:**

- Advantage: can use kernel functions to perform non-linear classification by mapping the data into a higher dimension;
- Disadvantage: for multi-classes classification, SVM can only do it indirectly by doing multiple two-classes classifications.

Problem 1

```
%Two classes
[index6,~]=find(labelTrain==6);
[index8,~]=find(labelTrain==8);
index=[index6;index8];
label68train=zeros(size(index6,1)+size(index8,1),1);
label68train=[labelTrain(index6);labelTrain(index8)];
train68=zeros(784,size(label68train,1));
for i=1:size(label68train,1)
      train68(:,i)=reshape(imageTrain(:,:,index(i)),[784,1]);
end
train68=train68'/255;
model1=svmtrain(label68train,train68,['-c 2^-4 -t 0']);
label68test=zeros(500,1);
for i=1:500
    if labelTest(i)==6
        label68test(i)=6;
    end
    if labelTest(i)==8
        label68test(i)=8;
    end
end
test68=zeros(784,500);
for i=1:500
    if label68test(i)~=0
      test68(:,i)=reshape(imageTest(:,:,i),[784,1]);
    end
end
x=find(label68test==0);
test68(:,x)=[];
label68test(x)=[];
test68=test68'/255;
[predicted_label,accuracy1,decision_values]=svmpredict(label68test,test68,
model1);
w=model1.SVs' * model1.sv_coef;
b=-model1.rho;
if (model1.Label(1) == 6)
    w = -w; b = -b;
end
%plot the normal vector
figure;
norm_vector=reshape(w,[28,28]);
imshow(norm_vector,[]);
title('normal vector')
d=zeros(model1.totalSV,1);
```

```
for i=1:model1.totalSV
    d(i)=abs(w'*model1.SVs(i,:)'+b)/norm(w);
end
[~,index]=sort(d);
closest5SV=zeros(28,28,5);
%plot the 5 cloest SVs
SV6=zeros(784,size(model1.sv_indices,1));
SV8=zeros(784,size(model1.sv_indices,1));
for i=1:size(model1.sv_indices,1)
    if    label68train(model1.sv_indices(i))==6
          SV6(:,i)=train68(model1.sv_indices(i),:)';
    end
    if    label68train(model1.sv_indices(i))==8
          SV8(:,i)=train68(model1.sv_indices(i),:)';
    end
end
SV6(:,all(~any(SV6),1))=[];
SV8(:,all(~any(SV8),1))=[];
dSV6=zeros(size(SV6,2),1);
dSV8=zeros(size(SV8,2),1);
for i=1:size(SV6,2)
    dSV6(i)=abs(w'*SV6(:,i)+b)/norm(w);
end
for i=1:size(SV8,2)
    dSV8(i)=abs(w'*SV8(:,i)+b)/norm(w);
end
[minSV6,indexminSV6]=sort(dSV6);
[minSV8,indexminSV8]=sort(dSV8);
figure;
for i=1:5
   subplot(2,5,i);
  imshow(reshape(SV6(:,indexminSV6(i))',[28,28]));
end
for i=1:5
  subplot(2,5,i+5);
  imshow(reshape(SV8(:,indexminSV8(i))',[28,28]));
end
suptitle('cloest SV 6 and 8');
[a6,~]=find(label68test==6);
[a8,~]=find(label68test==8);
d68test=zeros(size(test68,1),1);
d6test=zeros(size(a6,1),1);
d8test=zeros(size(a8,1),1);
test6=zeros(28,28,size(a6,1));
test8=zeros(28,28,size(a8,1));
for i=1:size(d68test,1)
    d68test(i)=abs(w'*test68(i,:)'+b)/norm(w);
end
%plot the histogram
```

```
figure;
histogram(d68test,10);
title('Histogram of distance 10 bins');
for i=1:size(a6,1)
  d6test(i)=abs(w'*test68(a6(i),:)'+b)/norm(w);
  test6(:,:,i)=reshape(test68(a6(i),:),[28,28]);
end
for i=1:size(a8,1)
  d8test(i)=abs(w'*test68(a8(i),:)'+b)/norm(w);
  test8(:,:,i)=reshape(test68(a8(i),:),[28,28]);
end
[max6,d6max]=sort(d6test,'descend');
[max8,d8max]=sort(d8test,'descend');
[min6,d6min]=sort(d6test);
[min8,d8min]=sort(d8test);
%plot 5 farthest and 5 cloest for each class
figure;
for i=1:5
  subplot(2,5,i);
  imshow(test6(:,:,d6max(i)),[]);
end
for i=1:5
  subplot(2,5,i+5);
  imshow(test8(:,:,d8max(i)),[]);
end
suptitle('farthest 6 and 8 to bounday');
figure;
for i=1:5
  subplot(2,5,i);
  imshow(test6(:,:,d6min(i)),[]);
end
for i=1:5
  subplot(2,5,i+5);
  imshow(test8(:,:,d8min(i)),[]);
end
suptitle('closest 6 and 8 to bounday');
```

Problem 2

```
traintotal=zeros(784,5000);
testtotal=zeros(784,500);
for i=1:5000
    traintotal(:,i)=reshape(imageTrain(:,:,i),[784,1]);
end
for i=1:500
    testtotal(:,i)=reshape(imageTest(:,:,i),[784,1]);
end
```

```
folds=2;
c=[2^-3;2^-1;2^1;2^3;2^5;2^7;2^9;2^11];
gamma=[2^-11;2^-9;2^-7;2^-5;2^-3;2^-1];
d=2;
cv_acc=zeros(48,3);
k=0;
for i=1:8
    for m=1:6
     k=k+1;
    cv_acc(k,3)=svmtrain(labelTrain,traintotal'/255,sprintf('-c %d -g %d -
v %d -t %d',c(i),gamma(m),folds,d));
    cv_acc(k,1)=i;
    cv_acc(k,2)=m;
    end
end
[highest_acc,idx] = max(cv_acc(:,3));
best_c=c(cv_acc(idx,1));
best_gamma=gamma(cv_acc(idx,2));
model2=svmtrain(labelTrain,traintotal'/255,['-c 2 -g 0.0313 -t 2']);
[predicted_label,accuracy2,decision_values]=svmpredict(labelTrain,traintot
al'/255, model2);
```