

HW 7 Report

1

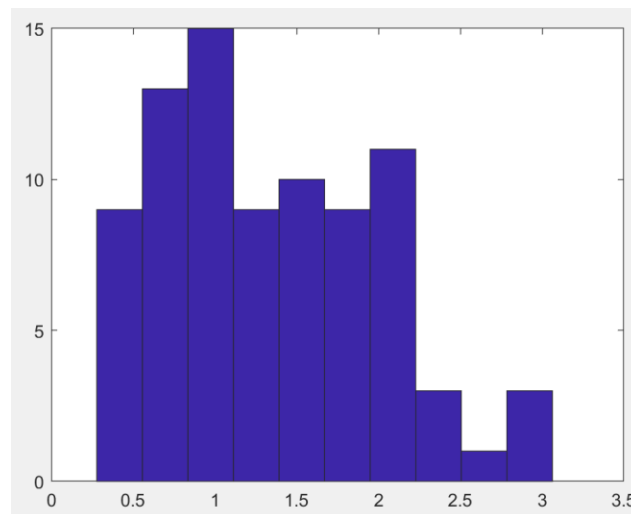
- (a) The classification accuracy was 100% (I used the order of -t -c for svmopts).
- (b) The five support vectors closest to the classification boundary for each class:



- (c) The normal vector to the separating hyperplane:



- (d) The histogram for all testing data distances to the calcification boundary using 20 bins:



(e) Five images in testing set that farthest from the boundary for each class:



(f) Five images in testing set that closest to the boundary for each class:



Compare above plots with the two in part e, we can conclude that the digits closer to the boundary look more scandalized.

2

(a) According to my results in the table below, there are 6 best pairs: gamma is 2^{-5} , and C is either 2, 2^3 , 2^5 , 2^7 , 2^9 , or 2^{11} . All these pairs give us 95.6% accuracy.

cv_accuracy						
8x6 double						
	1	2	3	4	5	6
1	33.7200	79.0600	88.8800	88.5600	19.3600	11.2600
2	80.3800	89.2600	92.3800	94.4000	31.0600	11.2600
3	89.2000	91.8400	94.5800	95.6000	74.5600	18.3600
4	91.4800	93.1400	94.9000	95.6000	74.5600	18.3600
5	92.5400	93	94.8000	95.6000	74.5600	18.3600
6	91.9000	92.7200	94.8000	95.6000	74.5600	18.3600
7	91.3800	92.6600	94.8000	95.6000	74.5600	18.3600
8	91.3200	92.6600	94.8000	95.6000	74.5600	18.3600

- (b) Using the pairs above to do SVM classification on the test set, I got an error rate of 4.4% which is the lowest error rate I got in this quarter.

```
optimization finished, #iter = 638
nu = 0.059567
obj = -60.553123, rho = 0.191351
nSV = 348, nBSV = 2
Total nSV = 3029
Accuracy = 95.6% (478/500) (classification)
fx>> |
```

3

- (a) Compare between KNN classifier, Gaussian with BDR, BDR on PCA, and SVM, the accuracy on the test set are listed below:

KNN: 90.6%
BDR: 77.8%
PCA: 95.2%
SVM: 95.6%

Clearly, we can yield the conclusion that among 4 classification techniques, Support Vector Machine has the best accuracy on classifying handwritten digits.

- (b) Advantages and disadvantages of each method:

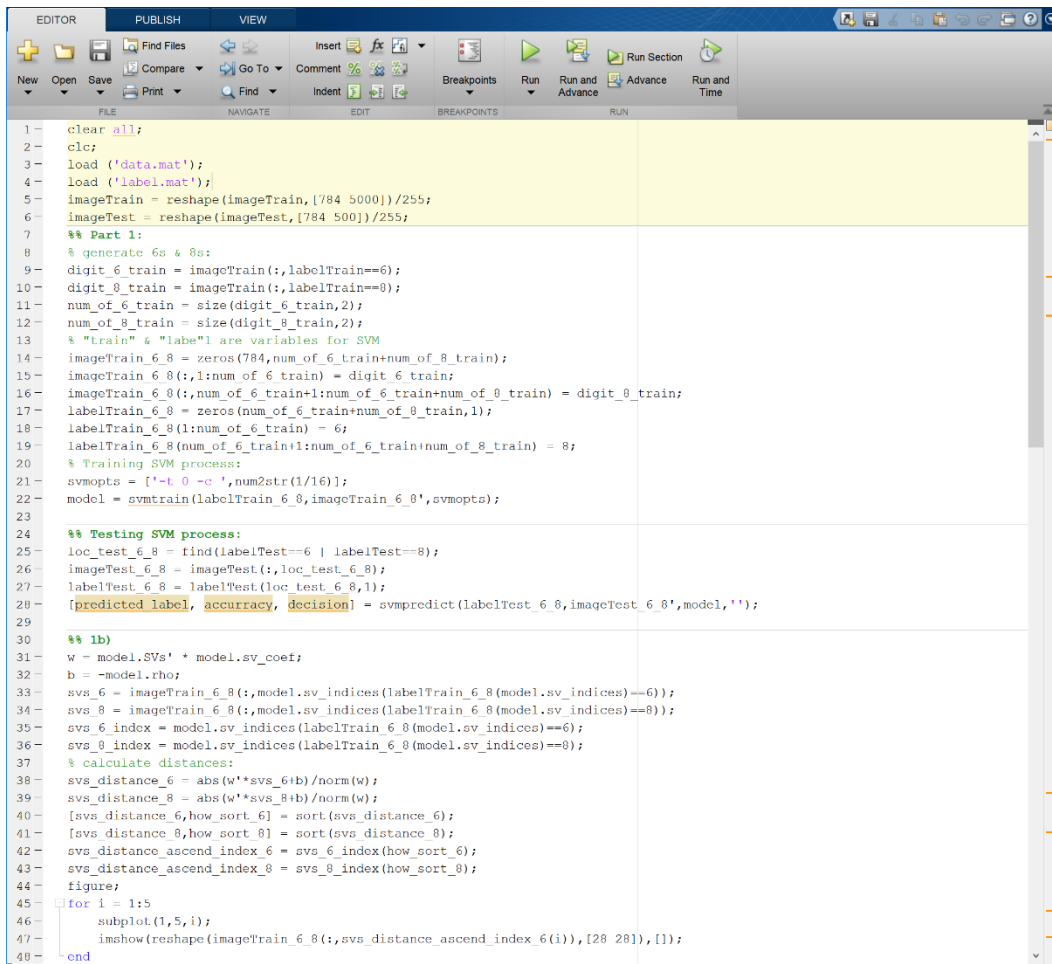
KNN: It is very robust to training set, and also, its concept is very easy to understand. However, it can be confused in real-world application when deciding which metric to use. Different metric will have huge impacts in accuracy.

Gaussian BDR: It is simple and easy to use and fast. BDR can make probabilistic predictions. However, among four techniques, BDR got the lowest accuracy. Also, as the dimension grows, it requires a huge amount of training data to maintain certain accuracy.

PCA: It hugely reduced the size of data sets and therefore makes the algorithm run faster. Also, it improves the accuracy by dropping unnecessary dimensions that produce noises (and almost no features). However, one of the disadvantages of PCA is when we over use it, and this may eliminate some important features without we knowing it.

SVM: SVM does almost perfectly for binary classification, and it did really well in our 10-classes classification. However, finding the best parameters for SVM may take some time. Also, according to some articles, SVM may be too sensitive and over-fitting the model sometimes.

Appendix



```
1 clear all;
2 clc;
3 load('data.mat');
4 load('label.mat');
5 imageTrain = reshape(imageTrain,[784 5000])/255;
6 imageTest = reshape(imageTest,[784 500])/255;
7 %% Part 1:
8 % generate 6s & 8s:
9 digit_6_train = imageTrain(:,labelTrain==6);
10 digit_8_train = imageTrain(:,labelTrain==8);
11 num_of_6_train = size(digit_6_train,2);
12 num_of_8_train = size(digit_8_train,2);
13 % "train" & "label" are variables for SVM
14 imageTrain_6_8 = zeros(784,num_of_6_train+num_of_8_train);
15 imageTrain_6_8(:,1:num_of_6_train) = digit_6_train;
16 imageTrain_6_8(:,num_of_6_train+1:num_of_6_train+num_of_8_train) = digit_8_train;
17 labelTrain_6_8 = zeros(num_of_6_train+num_of_8_train,1);
18 labelTrain_6_8(1:num_of_6_train) = 6;
19 labelTrain_6_8(num_of_6_train+1:num_of_6_train+num_of_8_train) = 8;
20 % Training SVM process:
21 svmopts = ['-t 0 -c ',num2str(1/16)];
22 model = svmtrain(labelTrain_6_8,imageTrain_6_8,svmopts);
23
24 %% Testing SVM process:
25 loc_test_6_8 = find(labelTest==6 | labelTest==8);
26 imageTest_6_8 = imageTest(:,loc_test_6_8);
27 labelTest_6_8 = labelTest(loc_test_6_8,1);
28 [predicted_label, accuracy, decision] = svmpredict(labelTest_6_8,imageTest_6_8,model,'');
29
30 %% 1b)
31 w = model.SVs' * model.sv_coef;
32 b = -model.rho;
33 sv_6 = imageTrain_6_8(:,model.sv_indices(labelTrain_6_8(model.sv_indices)==6));
34 sv_8 = imageTrain_6_8(:,model.sv_indices(labelTrain_6_8(model.sv_indices)==8));
35 sv_6_index = model.sv_indices(labelTrain_6_8(model.sv_indices)==6);
36 sv_8_index = model.sv_indices(labelTrain_6_8(model.sv_indices)==8);
37 % calculate distances:
38 sv_6_distance_6 = abs(w'*sv_6+b)/norm(w);
39 sv_8_distance_8 = abs(w'*sv_8+b)/norm(w);
40 [sv_6_distance_6,how_sort_6] = sort(sv_6_distance_6);
41 [sv_8_distance_8,how_sort_8] = sort(sv_8_distance_8);
42 sv_6_distance_ascend_index_6 = sv_6_index(how_sort_6);
43 sv_8_distance_ascend_index_8 = sv_8_index(how_sort_8);
44 figure;
45 for i = 1:5
46     subplot(1,5,i);
47     imshow(reshape(imageTrain_6_8(:,sv_6_distance_ascend_index_6(i)),[28 28]),[]);
48 end
```

```

49- figure;
50- for i = 1:5
51-     subplot(1,5,i);
52-     imshow(reshape(imageTrain_6_8(:,svs_distance_ascend_index_8(i)),[28 28]),[]);
53- end
54-
55- % figure;
56- % imshow(reshape(imageTrain_6_8(:,svs_distance_ascend_6(22)),[28 28]),[]);
57- % figure;
58- % imshow(reshape(imageTrain_6_8(:,svs_distance_ascend_6(23)),[28 28]),[]);
59- % figure;
60- % imshow(reshape(imageTrain_6_8(:,svs_distance_ascend_6(24)),[28 28]),[]);
61- %
62- %% 1c)
63- figure;
64- imshow(reshape(w,[28 28]),[]);
65- %% 1d)
66- test_distance = abs(w'*imageTest_6_8+b)/norm(w);
67- figure;
68- hist(test_distance,10);
69- %% 1e)
70- % data values:
71- imagetest_6 = imageTest_6_8(:,labelTest_6_8==6);
72- imagetest_8 = imageTest_6_8(:,labelTest_6_8==8);
73- % data index:
74- imagetest_6_index = find(labelTest_6_8==6);
75- imagetest_8_index = find(labelTest_6_8==8);
76- % test datas distance:
77- test_data_distance_6 = abs(w'*imagetest_6+b)/norm(w);
78- test_data_distance_8 = abs(w'*imagetest_8+b)/norm(w);
79- % sort:
80- [test_data_distance_6,how_sort_test_6] = sort(test_data_distance_6,'descend');
81- [test_data_distance_8,how_sort_test_8] = sort(test_data_distance_8,'descend');
82- % sorted index:
83- test_distance_descend_6_index = imagetest_6_index(how_sort_test_6);
84- test_distance_descend_8_index = imagetest_8_index(how_sort_test_8);
85- % plot 5 forest:
86- figure;
87- for i = 1:5
88-     subplot(1,5,i);
89-     imshow(reshape(imageTest_6_8(:,test_distance_descend_6_index(i)),[28 28]),[]);
90- end
91- figure;
92- for i = 1:5
93-     subplot(1,5,i);
94-     imshow(reshape(imageTest_6_8(:,test_distance_descend_8_index(i)),[28 28]),[]);
95- end
96-
97- %% 1f) Plot 5 closest:
98- figure;
99- for i = 1:5
100-     subplot(1,5,i);
101-     i = 44-i;
102-     imshow(reshape(imageTest_6_8(:,test_distance_descend_6_index(i)),[28 28]),[]);
103- end
104- figure;
105- for i = 1:5
106-     subplot(1,5,i);
107-     i = 41-i;
108-     imshow(reshape(imageTest_6_8(:,test_distance_descend_8_index(i)),[28 28]),[]);
109- end
110-
111-
112-
113- %% Part 2a): (with -t 2)
114- c_value = [-3;-1;1;3;5;7;9;11];
115- g_value = [-11,-9,-7,-5,-3,-1];
116- cv_accuracy = zeros(8,6);
117- for i = 1:8
118-     for j = 1:6
119-         cv_accuracy(i,j) = svmtrain(labelTrain,imageTrain',sprintf('-t 2 -c %f -g %f -v %d', 2^c_value(i), 2^g_value(j), 2));
120-     end
121- end
122- %% 2B): train use highest c and g:
123- model = svmtrain(labelTrain,imageTrain',sprintf('-t 2 -c %f -g %f', 2, 2^(-5)));
124- [predicted_label, accuracy, decision] = svmpredict(labelTest,imageTest',model,'');
125-
126-

```