CS156a Problem Set #8

1. When looking at the hard-margin SVM problem, we are addressing the following problem:

Minimize wTw subject to yn(wTxn + b) 1 for n = 1, 2, …, N

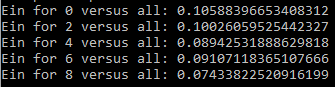
Without going through the Lagrangian dual problem, we are subject to the variables w and b. Since we are trying to minimize the product of the weights to the following constraints, we need quadratic programming, but the total variables only really depend on w on b. We know that:

[w1], [w2], … [wd]] and b = c

Where w is a d x 1 vector and b is equal to some constant c. Since the problem we are facing depend on these two variables, we just add the variables that w and b depend on. Thus, d + 1. The answer is [d].

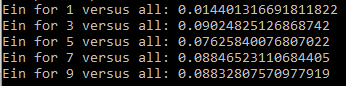
2. Answer is in the code:

1. from sklearn import svm
2. import numpy as np
4. train = open("train.txt", "r")
6. x = []
7. y = []
8. **for** line in train:
9. fields = line.strip().split()
10. **for** l in range(**int**(len(fields)/3)):
11. x.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l])])
12. y.append(**float**(fields[0 + 3 \* l]))
14. train.close()
16. **for** i in range(0, 10, 2):
17. y2 = []
18. **for** z in y:
19. **if** (z == i):
20. y2.append(1)
21. **else**:
22. y2.append(-1)
24. clf = svm.SVC(kernel='poly', C=0.01, degree=2, gamma=1, coef0=1)
25. clf.fit(x, y2)
26. y3 = clf.predict(x)
28. wrong = np.sum(y2 \* y3 < 0)
30. print("Ein for " + str(i) + " versus all: " + str(wrong/len(y2)))



The biggest value is 0 versus all with Ein of 0.106. The answer is [a].

3. Running the same code except changing line 16 to “for i in range(1, 11, 2): “



The smallest value is 1 versus all with Ein of 0.0144. The answer is [a].

4. Picking 0 versus all and 1 versus all, we can calculate the difference:

1. from sklearn import svm
2. import numpy as np
4. sup1 = 0
5. sup2 = 0
6. train = open("train.txt", "r")
8. x = []
9. y = []
10. **for** line in train:
11. fields = line.strip().split()
12. **for** l in range(**int**(len(fields)/3)):
13. x.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l])])
14. y.append(**float**(fields[0 + 3 \* l]))
16. train.close()
18. **for** i in range(2):
19. y2 = []
20. **for** z in y:
21. **if** (z == i):
22. y2.append(1)
23. **else**:
24. y2.append(-1)
26. clf = svm.SVC(kernel='poly', C=0.01, degree=2, gamma=1, coef0=1)
27. clf.fit(x, y2)
29. **if** sup1 == 0:
30. sup1 = len(clf.support\_)
31. **else**:
32. sup2 = len(clf.support\_)
34. print("Support vectors for " + str(i) + " versus all: " + str(len(clf.support\_)))
36. print("Difference: " + str(sup1 - sup2))

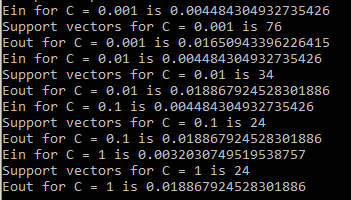


The difference is 1793, which is obviously close to 1800. The answer is [c].

5. Work is in the code:

1. from sklearn import svm
2. import numpy as np
4. sup1 = 0
5. sup2 = 0
6. train = open("train.txt", "r")
8. x = []
9. y = []
10. **for** line in train:
11. fields = line.strip().split()
12. **for** l in range(**int**(len(fields)/3)):
13. x.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l])])
14. y.append(**float**(fields[0 + 3 \* l]))
16. train.close()
18. test = open("test.txt", "r")
20. xtest = []
21. ytest = []
22. **for** line2 in test:
23. fields = line2.strip().split()
24. **for** l in range(**int**(len(fields)/3)):
25. xtest.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l])])
26. ytest.append(**float**(fields[0 + 3 \* l]))
28. xt2 = []
29. yt2 = []
30. **for** w in range(len(ytest)):
31. **if** (ytest[w] == 1):
32. yt2.append(1)
33. xt2.append(xtest[w])
34. elif (ytest[w] == 5):
35. yt2.append(-1)
36. xt2.append(xtest[w])
38. test.close()
40. **for** i in [0.001, 0.01, 0.1, 1]:
41. y2 = []
42. x2 = []
43. **for** z in range(len(y)):
44. **if** (y[z] == 1):
45. y2.append(1)
46. x2.append(x[z])
47. elif (y[z] == 5):
48. y2.append(-1)
49. x2.append(x[z])

52. clf = svm.SVC(kernel='poly', C=i, degree=2, gamma=1, coef0=1)
53. clf.fit(x2, y2)
54. y3 = clf.predict(x2)
55. wrong = np.sum(y2 \* y3 < 0)
57. print("Ein for C = " + str(i) + " is " + str(wrong/len(y2)))
59. print("Support vectors for C = " + str(i) + " is " + str(len(clf.support\_)))
61. y4 = clf.predict(xt2)
62. outwrong = np.sum(y4 \* yt2 < 0)
64. print("Eout for C = " + str(i) + " is " + str(outwrong/len(yt2)))



Now, consider each option:

a. As C goes up, the number of support vectors does not change from 0.1 to 1 so false.

b. As C goes up, the number of support vectors decreases from 76 to 34 from C = 0.001 to 0.01 so false.

c. As C goes up, Eout­ increases from 0.0165 to 0.0189 from C = 0.001 to 0.01 so false.

d. When C = 1, the Ein ­is the lowest at 0.0032 when the others are at 0.0045 so true.

The answer is [d].

6. With code like the last script except add a for loop change the degree aspect of the SVM:

1. from sklearn import svm
2. import numpy as np
4. sup1 = 0
5. sup2 = 0
6. train = open("train.txt", "r")
8. x = []
9. y = []
10. **for** line in train:
11. fields = line.strip().split()
12. **for** l in range(**int**(len(fields)/3)):
13. x.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l])])
14. y.append(**float**(fields[0 + 3 \* l]))
16. train.close()
18. test = open("test.txt", "r")
20. xtest = []
21. ytest = []
22. **for** line2 in test:
23. fields = line2.strip().split()
24. **for** l in range(**int**(len(fields)/3)):
25. xtest.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l])])
26. ytest.append(**float**(fields[0 + 3 \* l]))
28. xt2 = []
29. yt2 = []
30. **for** w in range(len(ytest)):
31. **if** (ytest[w] == 1):
32. yt2.append(1)
33. xt2.append(xtest[w])
34. elif (ytest[w] == 5):
35. yt2.append(-1)
36. xt2.append(xtest[w])
38. test.close()
40. **for** i in [0.0001, 0.001, 0.01, 1]:
41. **for** deg in [2, 5]:
42. y2 = []
43. x2 = []
44. **for** z in range(len(y)):
45. **if** (y[z] == 1):
46. y2.append(1)
47. x2.append(x[z])
48. elif (y[z] == 5):
49. y2.append(-1)
50. x2.append(x[z])

53. clf = svm.SVC(kernel='poly', C=i, degree=deg, gamma=1, coef0=1)
54. clf.fit(x2, y2)
55. y3 = clf.predict(x2)
56. wrong = np.sum(y2 \* y3 < 0)
58. print("Ein for C = " + str(i) + " and Q = " + str(deg) + " is " + str(wrong/len(y2)))
60. print("Support vectors for C = " + str(i) + " and Q = " + str(deg) + " is " + str(len(clf.support\_)))
62. y4 = clf.predict(xt2)
63. outwrong = np.sum(y4 \* yt2 < 0)
65. print("Eout for C = " + str(i) + " and Q = " + str(deg) + " is " + str(outwrong/len(yt2)))

The output is on the top of the next page. Consider each option:

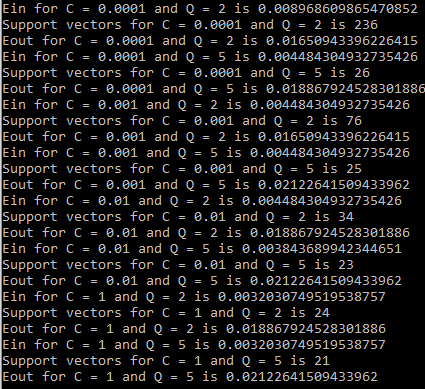
a. Ein at C = 0.001 and Q = 2 is 0.00897 and Q = 5 is 0.00448 so false.

b. Support vectors at C = 0.001 and Q = 2 is 76 and Q = 5 is 25 so true.

c. Ein­ at C = 0.01 and Q = 2 is 0.00448 and Q = 5 is 0.00384 so false.

d. Eout at C = 1 and Q = 2 is 0.0189 and Q = 5 is 0.0212 so false.

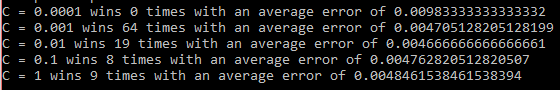
The answer is [b].



7. Work is in the code:

1. from sklearn import svm
2. import numpy as np
3. import random
5. train = open("train.txt", "r")
7. x = []
8. **for** line in train:
9. fields = line.strip().split()
10. **for** l in range(**int**(len(fields)/3)):
11. x.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l]), **float**(fields[0 + 3 \* l])])
13. train.close()
15. first = [0, 0]
16. second = [0, 0]
17. third = [0, 0]
18. fourth = [0, 0]
19. fifth = [0, 0]
20. **for** b in range(100):
21. random.shuffle(x)
22. maxi = []
23. **for** i in [0.0001, 0.001, 0.01, 0.1, 1]:
24. y2 = []
25. x2 = []
26. **for** z in range(len(x)):
27. **if** (x[z][2] == 1):
28. y2.append(1)
29. x2.append([x[z][0], x[z][1]])
30. elif (x[z][2] == 5):
31. y2.append(-1)
32. x2.append([x[z][0], x[z][1]])
34. disagree = 0
36. **for** c in range(10):
37. x3 = x2[c \* **int**(len(x2)/10): (c + 1) \* **int**(len(x2)/10)]
38. y3 = y2[c \* **int**(len(y2)/10): (c + 1) \* **int**(len(y2)/10)]
39. x4 = []
40. y4 = []
41. **for** p in range(len(x2)):
42. **if** x2[p] not in x3:
43. x4.append(x2[p])
44. y4.append(y2[p])
46. clf = svm.SVC(kernel='poly', C=i, degree=2, gamma=1, coef0=1)
47. clf.fit(x4, y4)
48. y5 = clf.predict(x3)
49. disagree += (np.sum(y3 \* y5 < 0)) / len(x3)
51. maxi.append(disagree/10)
53. first[1] += maxi[0]
54. second[1] += maxi[1]
55. third[1] += maxi[2]
56. fourth[1] += maxi[3]
57. fifth[1] += maxi[4]
59. test = 1
60. index = 0
61. **for** j in range(len(maxi)):
62. **if** maxi[j] < test:
63. index = j
64. test = maxi[j]
66. **if** index == 0:
67. first[0] += 1
68. elif index == 1:
69. second[0] += 1
70. elif index == 2:
71. third[0] +=1
72. elif index == 3:
73. fourth[0] +=1
74. elif index == 4:
75. fifth[0] += 1

78. print("C = 0.0001 wins " + str(first[0]) +" times with an average error of " + str(first[1]/100))
79. print("C = 0.001 wins " + str(second[0]) +" times with an average error of " + str(second[1]/100))
80. print("C = 0.01 wins " + str(third[0]) +" times with an average error of " + str(third[1]/100))
81. print("C = 0.1 wins " + str(fourth[0]) +" times with an average error of " + str(fourth[1]/100))
82. print("C = 1 wins " + str(fifth[0]) +" times with an average error of " + str(fifth[1]/100))



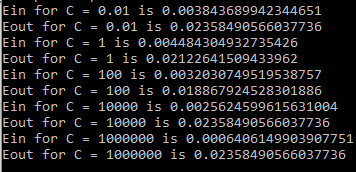
The value with the most wins is C = 0.001. The answer is [b].

8. The average error for C = 0.001 is closest to 0.005. The answer is [c].

9. The work is in the code:

1. from sklearn import svm
2. import numpy as np
4. sup1 = 0
5. sup2 = 0
6. train = open("train.txt", "r")
8. x = []
9. y = []
10. **for** line in train:
11. fields = line.strip().split()
12. **for** l in range(**int**(len(fields)/3)):
13. x.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l])])
14. y.append(**float**(fields[0 + 3 \* l]))
16. train.close()
18. test = open("test.txt", "r")
20. xtest = []
21. ytest = []
22. **for** line2 in test:
23. fields = line2.strip().split()
24. **for** l in range(**int**(len(fields)/3)):
25. xtest.append([**float**(fields[1 + 3 \* l]), **float**(fields[2 + 3 \* l])])
26. ytest.append(**float**(fields[0 + 3 \* l]))
28. xt2 = []
29. yt2 = []
30. **for** w in range(len(ytest)):
31. **if** (ytest[w] == 1):
32. yt2.append(1)
33. xt2.append(xtest[w])
34. elif (ytest[w] == 5):
35. yt2.append(-1)
36. xt2.append(xtest[w])
38. test.close()
40. **for** i in [0.01, 1, 100, 10000, 1000000]:
41. y2 = []
42. x2 = []
43. **for** z in range(len(y)):
44. **if** (y[z] == 1):
45. y2.append(1)
46. x2.append(x[z])
47. elif (y[z] == 5):
48. y2.append(-1)
49. x2.append(x[z])

52. clf = svm.SVC(kernel='rbf', C=i, gamma=1)
53. clf.fit(x2, y2)
54. y3 = clf.predict(x2)
55. wrong = np.sum(y2 \* y3 < 0)
57. print("Ein for C = " + str(i) + " is " + str(wrong/len(y2)))
59. y4 = clf.predict(xt2)
60. outwrong = np.sum(y4 \* yt2 < 0)
62. print("Eout for C = " + str(i) + " is " + str(outwrong/len(yt2)))



The smallest Ein­ happens at C = 1000000 as 0.00064. The answer is [e].

10. The smallest Eout­ happens at C = 100 as 0.0189. The answer is [c].