

HW8

1) [a] We are trying to minimize W which we had split into $(w_0 + w_1 x_1 + \dots + w_d x_d)$ where w_0 is w_0 hence w has d parameters and b has 1 parameter hence we have quadratic programming with $d+1$ variables with $d+1$ variables.

(2) [a]

```
hw8.py > ...
1 import numpy
2 from sklearn import svm
3 import random
4
5 train = numpy.loadtxt('../features.train')
6 test = numpy.loadtxt('../features.test')
7 digits_train = train[:, 0]
8 X_train = train[:, 1:]
9 digits_test = test[:, 0]
10 X_test = test[:, 1:]
11
12
13 def make_binary(digit_class, digits):
14     scores = []
15     for score in digits:
16         if (score == digit_class):
17             scores.append(1)
18         else:
19             scores.append(-1)
20     return scores
21
22 def Ein_d_classifier, c, q):
23     binary_train = make_binary(d_classifier, digits_train)
24     classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
25     classifier.fit(X_train, binary_train)
26     return 1 - classifier.score(X_train, binary_train)
27 > def Rout_d_classifier, c, q):
28 > def support_vectors_d_classifier, c, q):
29 > def Ecv(digits, X, folds, c, q):
30 > def find_best_c_d_classifier, runs:
31 > def q1():
32     print("for 0, Ein is: ", Ein0, 0.01, 2)
33     print("for 2, Ein is: ", Ein2, 0.01, 2)
34     print("for 4, Ein is: ", Ein4, 0.01, 2)
35     print("for 6, Ein is: ", Ein6, 0.01, 2)
36     print("for 8, Ein is: ", Ein8, 0.01, 2)
37
38 > def q2():
39 > def q3():
40 > def q4():
41 > def q5():
42 > def q6():
43 > def q7():
44 > def q8():
45 > def q9():
46 > def q10():
47 > def q11():
48 > def q12():
49 > def q13():
50 > def q14():
51 > def q15():
52 > def q16():
53 > def q17():
54 > def q18():
55 > def q19():
56 > def q20():
57 > def q21():
58 > def q22():
59 > def q23():
60 > def q24():
61 > def q25():
62 > def q26():
63 > def q27():
64 > def q28():
65 > def q29():
66 > def q30():
67 > def q31():
68 > def q32():
69 > def q33():
70 > def q34():
71 > def q35():
72 > def q36():
73 > def q37():
74 > def q38():
75 > def q39():
76 > def q40():
77 > def q41():
78 > def q42():
79 > def q43():
80 > def q44():
81 > def q45():
82 > def q46():
83 > def q47():
84 > def q48():
85 > def q49():
86 > def q50():
87 > def q51():
88 > def q52():
89 > def q53():
90 > def q54():
91 > def q55():
92 > def q56():
93 > def q57():
94 > def q58():
95 > def q59():
96 > def q60():
97 > def q61():
98 > def q62():
99 > def q63():
100 > def q64():
101 > def q65():
102 > def q66():
103 > def q67():
104 > def q68():
105 > def q69():
106 > def q70():
107 > def q71():
108 > def q72():
109 > def q73():
110 > def q74():
111 > def q75():
112 > def q76():
113 > def q77():
114 > def q78():
115 > def q79():
116 > def q80():
117 > def q81():
118 > def q82():
119 > def q83():
120 > def q84():
121 > def q85():
122 > def q86():
123 > def q87():
124 > def q88():
125 > def q89():
126 > def q90():
127 > def q91():
128 > def q92():
129 > q2()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER VARIABLES

```
# thiernothierino-MacBook-Pro CS 156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS 156a/hw8.py"
for 0, Ein is: 0.1058236653480316
for 2, Ein is: 0.10826859525442321
for 4, Ein is: 0.110718365107669
for 6, Ein is: 0.11322823507669
for 8, Ein is: 0.1157382851867669
for 0, Ein is: 0.09107118365107669
for 2, Ein is: 0.094338225820162
for 4, Ein is: 0.097848225820162
for 6, Ein is: 0.101358225820162
for 8, Ein is: 0.104868225820162
```

Hence according to the above code
the highest Ein is for 0 versus all hence
the correct answer is [a]

Ln 123, Col 5 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit R D

(3) [a]

```
hw8.py > @ make_binary
1 import numpy
2 from sklearn import svm
3 import random
4
5 train = numpy.loadtxt('../features.train')
6 test = numpy.loadtxt('../features.test')
7 digits_train = train[:, 0]
8 X_train = train[:, 1:]
9 digits_test = test[:, 0]
10 X_test = test[:, 1:]
11
12
13 def make_binary(digit_class, digits):
14     scores = []
15     for score in digits:
16         if (score == digit_class):
17             scores.append(1)
18         else:
19             scores.append(-1)
20     return scores
21
22 def Ein_d_classifier, c, q:
23     binary_train = make_binary(d_classifier, digits_train)
24     classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
25     classifier.fit(X_train, binary_train)
26     return 1 - classifier.score(X_train, binary_train)
27 > def Rout_d_classifier, c, q:
28 > def support_vectors_d_classifier, c, q:
29 > def Ecv(digits, X, folds, c, q):
30 > def find_best_c_d_classifier, runs:
31 > def q21():
32     print("for 1, Ein is: ", Ein1, 0.01, 2)
33     print("for 2, Ein is: ", Ein2, 0.01, 2)
34     print("for 4, Ein is: ", Ein4, 0.01, 2)
35     print("for 6, Ein is: ", Ein6, 0.01, 2)
36     print("for 8, Ein is: ", Ein8, 0.01, 2)
37
38 > def q22():
39 > def q23():
40 > def q24():
41 > def q25():
42 > def q26():
43 > def q27():
44 > def q28():
45 > def q29():
46 > def q30():
47 > def q31():
48 > def q32():
49 > def q33():
50 > def q34():
51 > def q35():
52 > def q36():
53 > def q37():
54 > def q38():
55 > def q39():
56 > def q40():
57 > def q41():
58 > def q42():
59 > def q43():
60 > def q44():
61 > def q45():
62 > def q46():
63 > def q47():
64 > def q48():
65 > def q49():
66 > def q50():
67 > def q51():
68 > def q52():
69 > def q53():
70 > def q54():
71 > def q55():
72 > def q56():
73 > def q57():
74 > def q58():
75 > def q59():
76 > def q60():
77 > def q61():
78 > def q62():
79 > def q63():
80 > def q64():
81 > def q65():
82 > def q66():
83 > def q67():
84 > def q68():
85 > def q69():
86 > def q70():
87 > def q71():
88 > def q72():
89 > def q73():
90 > def q74():
91 > def q75():
92 > def q76():
93 > def q77():
94 > def q78():
95 > def q79():
96 > def q80():
97 > def q81():
98 > def q82():
99 > def q83():
100 > def q84():
101 > def q85():
102 > def q86():
103 > def q87():
104 > def q88():
105 > def q89():
106 > def q90():
107 > def q91():
108 > def q92():
109 > def q93():
110 > def q94():
111 > def q95():
112 > def q96():
113 > def q97():
114 > def q98():
115 > def q99():
116 > def q100():
117 > def q101():
118 > def q102():
119 > def q103():
120 > def q104():
121 > def q105():
122 > def q106():
123 > def q107():
124 > def q108():
125 > def q109():
126 > def q110():
127 > def q111():
128 > def q112():
129 > q21()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER VARIABLES

```
# thiernothierino-MacBook-Pro CS 156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS 156a/hw8.py"
for 1, Ein is: 0.01440131669181172
for 2, Ein is: 0.01692504876807822
for 4, Ein is: 0.01846523118684801
for 6, Ein is: 0.0198328075997919
for 8, Ein is: 0.021200000000000002
```

Hence according to the above code the
lowest Ein is for 4 vs all hence the
correct answer is [a]

Ln 18, Col 30 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit R D

(4) [C]

```
hw8.py > make_binary
1 import numpy
2 from sklearn import svm
3 import random
4
5 train = numpy.loadtxt('../features.train')
6 test = numpy.loadtxt('../features.test')
7 digits_train = train[:, 0]
8 X_train = train[:, 1:]
9 digits_test = test[:, 0]
10 X_test = test[:, 1:]
11
12
13 def make_binary(digit_class, digits):
14     scores = []
15     for score in digits:
16         if (score == digit_class):
17             scores.append(1)
18         else:
19             scores.append(-1)
20     return scores
21
22 > def c1(d_classifier, c, q):-
23 >     def E1(c1,d_classifier, c, q):-
24 >         def support_vectors(d_classifier, c, q):-
25 >             binary_train = make_binary(d_classifier, digits_train)
26 >             classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
27 >             classifier.fit(X_train, binary_train)
28 >             return sum(classifier.n_support_)
29 >     def E2(digits, X, folds, c, q):-
30 >         def find_best_c(d_classifier, runs):-
31 >             def q1():-
32 >                 def q2():-
33 >                     print("for 0, number of support vectors are : ", support_vectors(0, 0.01, 2))
34 >                     print("for 1, number of support vectors are : ", support_vectors(1, 0.01, 2))
35 >                     print("Their difference in number of support vectors is: ", support_vectors(0, 0.01, 2) - support_vectors(1, 0.01, 2))
36 >                 def q3():-
37 >                     def q4():-
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER VARIABLES

```
for 0, Ein is: 0.00846523110664001
for 8, Ein is: 0.00832887570977919
• thierno@Thiernos-MacBook-Pro CS 156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS 156a/hw8.py"
for 0, number of support vectors are: 2179
for 1, number of support vectors are: 3886
Their difference in number of support vectors is: 1703
o thierno@Thiernos-MacBook-Pro CS 156a %
```

hence according to the above code, the correct answer is [C] as it is closest.

Ln 18, Col 14 Spaces: 4 UTF-8 LF ↻ Python 3.10.6 64-bit ⌂ ⌂

~~8~~ Instead of Screenshotting the same code,
the below Code is the same for all functions
and Variables for questions 5 through 10.

```
hw8.py > ...
1 import numpy
2 from sklearn import svm
3 import random
4
5 train = numpy.loadtxt('../features.train')
6 test = numpy.loadtxt('../features.test')
7 #One_v_One
8 def one_v_one(data, num1, num2):
9     new_data = []
10    for thing in data:
11        if thing[0] == num1 and thing[0] == num2:
12            continue
13        else:
14            new_data.append(thing)
15    return numpy.array(new_data)
16
17 etrain = one_v_one(train, 1, 5)
18 etest = one_v_one(test, 1, 5)
19 digits_train = etrain[:, 0]
20 Xtrain = etrain[:, 1:]
21 digits_test = etest[:, 0]
22 Xtest = etest[:, 1:]
23
24 def make_binary(digit_class, digits):
25     scores = []
26     for score in digits:
27         if (score == digit_class):
28             scores.append(1)
29         else:
30             scores.append(-1)
31     return scores
32
33 def Ein(d_classifier, c, q):
34     binary_train = make_binary(d_classifier, digits_train)
35     classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
36     classifier.fit(X_train, binary_train)
37     return 1 - classifier.score(X_train, binary_train)
38
39 def Eout(d_classifier, c, q):
40     binary_train = make_binary(d_classifier, digits_train)
41     binary_test = make_binary(d_classifier, digits_test)
42     classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
43     classifier.fit(X_train, binary_train)
44     return 1 - classifier.score(X_test, binary_test)
45
46
47 hw8.py > ...
48 def Eout_d_classifier(c, q):
49     binary_train = make_binary(d_classifier, digits_train)
50     binary_test = make_binary(d_classifier, digits_test)
51     classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
52     classifier.fit(X_train, binary_train)
53     classifier.fit(X_train, binary_train)
54     return 1 - classifier.score(X_test, binary_test)
55
56 def support_vectors(d_classifier, c, q):
57     binary_train = make_binary(d_classifier, digits_train)
58     classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
59     classifier.fit(X_train, binary_train)
60     return sum(classifier.n_support_)
61
62 def Ecv(digits, X, folds, c, q):
63     k = int(len(digits)/folds)
64     tot_error = 0
65     for n in range(0, k * folds, k):
66         D_val_X = X[n:n+k]
67         D_val_Y = digits[n:n+k]
68         D_train_X = numpy.concatenate((X[:n:k], X[:n]), axis=0)
69         D_train_Y = numpy.concatenate((digits[:n:k], digits[:n]), axis=0)
70         classifier = svm.SVC(C=c, kernel='poly', degree=q, gamma=1.0, coef0=1.0)
71         classifier.fit(D_train_X, D_train_Y)
72         tot_error += (1 - classifier.score(D_val_X, D_val_Y))
73     return tot_error/len(digits)
74
75 def find_best_(d_classifier, runs):
76     X_train = X_train
77     binary_train = make_binary(d_classifier, digits_train)
78     wins = {0.0001 : [0, 0], 0.001 : [0, 0], 0.01 : [0, 0], 0.1 : [0, 0], 1 : [0, 0]}
79     for i in range(runs):
80         curr_min = numpy.Inf
81         curr_c = 0
82         for c in [0.0001, 0.001, 0.01, 0.1, 1]:
83             evc = Ecv(numpy.array(binary_train), numpy.array(X_train), 10, c, 2)
84             if evc < curr_min:
85                 curr_min = evc
86                 curr_c = c
87         data = list(zip(binary_train, X_train))
88         random.shuffle(data)
89         binary_train, X_train = list(zip(*data))
90         wins[curr_c[0]] += 1
91         wins[curr_c[1]] += evc
92
93     return wins
```

(5) [d]

```

hw8.py > ...
    curr_c, curr_q, curr_evc = last_layer(modular)
78     wins[curr_c][0] += 1
79     wins[curr_c][1] += evc
80     return wins
81 > def q2():
82     > def q3():
83         > def q4():
84             > def q5(q):
85                 print("for c = 0.001, Ein, support vectors, and Eout are: ", [Ein[1], 0.001, q], support_vectors[1, 0.001, q], Eout[1, 0.001, q])
86                 print("for c = 0.01, Ein, support vectors, and Eout are: ", [Ein[1, 0.01, q], support_vectors[1, 0.01, q], Eout[1, 0.01, q]])
87                 print("for c = 0.1, Ein, support vectors, and Eout are: ", [Ein[1, 0.1, q], support_vectors[1, 0.1, q], Eout[1, 0.1, q]])
88                 print("for c = 1, Ein, support vectors, and Eout are: ", [Ein[1, 1, q], support_vectors[1, 1, q], Eout[1, 1, q]])
89             > def q6():
90                 > def q7():
91                     > def q8():
92                         > def q9():
93                             > def q10():
94                                 q5()
95

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

Traceback (most recent call last):
File "/Users/thierno/Desktop/Caltech/5more/Fall 22/CS 156a/hw8.py", line 142, in <module>
 q5()
TypeError: q5() missing 1 required positional argument: 'q'
> thierno@Thiernos-MacBook-Pro:~/Desktop/Caltech/5more/Fall 22/CS 156a/hw8.py"
 for c = 0.001, Ein, support vectors, and Eout are: [0.00448438493735439, 76, 0.0165094396226412]
 for c = 0.01, Ein, support vectors, and Eout are: [0.00448438493735439, 34, 0.018867924528301883]
 for c = 0.1, Ein, support vectors, and Eout are: [0.00448438493735439, 24, 0.038867924528301883]
 for c = 1, Ein, support vectors, and Eout are: [0.003283749519538215, 24, 0.018867924528301883]
 thierno@Thiernos-MacBook-Pro:~/Desktop/Caltech/5more/Fall 22/CS 156a/hw8.py]

← According to the above code
the correct answer is d as.

It is the only one that matches
the outputted behaviour of our code.

```

hw8.py > @ q5
    curr_c, curr_q, curr_evc = last_layer(modular)
78     wins[curr_c][0] += 1
79     wins[curr_c][1] += evc
80     return wins
81 > def q2():
82     > def q3():
83         > def q4():
84             > def q5(q):
85                 print("for c = 0.0001, Ein, support vectors, and Eout are: ", [Ein[1], 0.0001, q], support_vectors[1, 0.0001, q], Eout[1, 0.0001, q])
86                 print("for c = 0.001, Ein, support vectors, and Eout are: ", [Ein[1, 0.001, q], support_vectors[1, 0.001, q], Eout[1, 0.001, q]])
87                 print("for c = 0.01, Ein, support vectors, and Eout are: ", [Ein[1, 0.01, q], support_vectors[1, 0.01, q], Eout[1, 0.01, q]])
88                 print("for c = 0.1, Ein, support vectors, and Eout are: ", [Ein[1, 0.1, q], support_vectors[1, 0.1, q], Eout[1, 0.1, q]])
89             > def q6():
90                 > def q7():
91                     > def q8():
92                         > def q9():
93                             > def q10():
94                                 q5()
95

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

for c = 1, Ein, support vectors, and Eout are: [0.003283749519538215, 21, 0.021266415094339646]
 thierno@Thiernos-MacBook-Pro:~/Desktop/Caltech/5more/Fall 22/CS 156a/hw8.py"
 for c = 0.0001, Ein, support vectors, and Eout are: [0.00895680985470878, 236, 0.0165094396226412]
 for c = 0.001, Ein, support vectors, and Eout are: [0.00448438493735439, 76, 0.0165094396226412]
 for c = 0.01, Ein, support vectors, and Eout are: [0.00448438493735439, 34, 0.018867924528301883]
 for c = 0.1, Ein, support vectors, and Eout are: [0.00448438493735439, 24, 0.038867924528301883]
 for q = 3 we have: [0.00448438493735439, 26, 0.018867924528301883]
 for c = 0.0001, Ein, support vectors, and Eout are: [0.00448438493735439, 25, 0.021266415094339646]
 for c = 0.001, Ein, support vectors, and Eout are: [0.00448438493735439, 23, 0.021266415094339646]
 for c = 0.01, Ein, support vectors, and Eout are: [0.003283749519538215, 23, 0.021266415094339646]
 thierno@Thiernos-MacBook-Pro:~/Desktop/Caltech/5more/Fall 22/CS 156a/hw8.py]

← According to the
above code, the correct
answer is [b] as.

It is the only one
that matches the behaviour
of our output.

(7) [b]

```
hw8.py > ...
78     for c in [0.0001, 0.001, 0.01, 0.1, 1]:
79         evc = Evc(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
80         if evc < curr_min:
81             curr_min = evc
82             curr_c = c
83     data = list(zip(binary_train, x_train))
84     random.shuffle(data)
85     binary_train, x_train = list(zip(*data))
86     wins[curr_c][0] += 1
87     wins[curr_c][1] += evc
88
89     return wins
90
91 > def q2():
92     <-- We have:
93     > def q3():
94     > def q4():
95     > def q5(q):
96     > def q6():
97     > def q7():
98
99     score_board = find_best_cif(100)
100    print("c= 0.0001 won this many times: ", score_board[0.0001][0])
101    print("c= 0.001 won this many times: ", score_board[0.001][0])
102    print("c= 0.01 won this many times: ", score_board[0.01][0])
103    print("c= 0.1 won this many times: ", score_board[0.1][0])
104    print("c= 1 won this many times: ", score_board[1][0])
105
106 > def q8():
107 > def q9():
108 > def q10():
109 > def q11():
110 q7()
```

```
for c in [0.0001, 0.001, 0.01, 0.1, 1]:
    evc = Evc(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
    if evc < curr_min:
        curr_min = evc
        curr_c = c
data = list(zip(binary_train, x_train))
random.shuffle(data)
binary_train, x_train = list(zip(*data))
wins[curr_c][0] += 1
wins[curr_c][1] += evc
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
for c = 0.1, Ein, support vectors, and Out are: [0.004464304932735439, 24, 0.818867924526381883]
for c = 1, Ein, support vectors, and Out are: [0.0032830749519538215, 24, 0.818867924526381883]
for q = 5 we have:
for c = 0.0001, Ein, support vectors, and Out are: [0.004464304932735439, 25, 0.822264598439646]
for c = 0.001, Ein, support vectors, and Out are: [0.0032830749519538215, 23, 0.822264598439646]
for c = 0.01, Ein, support vectors, and Out are: [0.0032830749519538215, 25, 0.818867924526381883]
for c = 0.1, Ein, support vectors, and Out are: [0.0032830749519538215, 21, 0.822264598439646]
@thermo@thermo-MacBook-Pro CS 156a % cd "/Users/thermo/Desktop/Caltech/Smore/Fall 22/CS 156a/hw8.py"
c= 0.0001 won this many times: 0
c= 0.001 won this many times: 27
c= 0.01 won this many times: 9
c= 0.1 won this many times: 1
@thermo@thermo-MacBook-Pro CS 156a %
```

According to the above code
the correct answer is [b]

Ln 142, Col 5 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit R Q

(8) [c]

```
hw8.py > ...
63     def find_best_cif(classifier, runs):
64         train = X_train
65         binary_train = make_binary_id_classifier(digits_train)
66         wins = {0.0001: [0, 0], 0.001: [0, 0], 0.01: [0, 0], 0.1: [0, 0]}
67         for i in range(runs):
68             curr_min = numpy.inf
69             curr_c = 0
70             for c in [0.0001, 0.001, 0.01, 0.1, 1]:
71                 evc = Evc(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
72                 if evc < curr_min:
73                     curr_min = evc
74                     curr_c = c
75             data = list(zip(binary_train, x_train))
76             random.shuffle(data)
77             binary_train, x_train = list(zip(*data))
78             wins[curr_c][0] += 1
79             wins[curr_c][1] += evc
80
81     return wins
82
83 > def q2():
84 > def q3():
85 > def q4():
86 > def q5():
87 > def q6():
88 > def q7():
89 > def q8():
90     score_board = find_best_cif(100)
91     print("c= 0.001 had an average Evc of: ", (score_board[0.001][1]) / (100 + (score_board[0.001][0])))
92
93 > def q9():
94 > def q10():
95 > def q11():
96 > def q12():
97 q7()
```

```
for c in [0.0001, 0.001, 0.01, 0.1, 1]:
    evc = Evc(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
    if evc < curr_min:
        curr_min = evc
        curr_c = c
data = list(zip(binary_train, x_train))
random.shuffle(data)
binary_train, x_train = list(zip(*data))
wins[curr_c][0] += 1
wins[curr_c][1] += evc
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
c= 1 won this many times: 8
@thermo@thermo-MacBook-Pro CS 156a % cd "/Users/thermo/Desktop/Caltech/Smore/Fall 22/CS 156a/hw8.py"
c= 0.0001 had an average Evc of: 0.046257512641324794
@thermo@thermo-MacBook-Pro CS 156a %
```

Hence according to the above code, the
Correct answer is [c]

(a) [e]

```
for i in range(runs):
    curr_min = numpy.inf
    curr_c = 0
    for c in [0.0001, 0.001, 0.01, 0.1, 1]:
        evc = Evc(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
        if evc < curr_min:
            curr_min = evc
            curr_c = c
    data = list(zip(binary_train, x_train))
    random.shuffle(data)
    binary_train, x_train = list(zip(*data))
    wins[curr_c][0] += 1
    wins[curr_c][1] += evc
return wins
```

```
def q2():
def q3():
def q4():
def q5(q):
def q6():
def q7():
def q8():
def q9():
    binary_train = make_binary(1, digits_train)
    curr_min = numpy.inf
    curr_c = 0
    for c in [0.01, 1, 100, 10 ** 4, 10 ** 6]:
        classifier = svm.SVC(C=c, kernel='rbf', degree=2, gamma=1.0, coef0=1.0)
        classifier.fit(X_train, binary_train)
        Ein = 1 - classifier.score(X_train, binary_train)
        if (Ein < curr_min):
            curr_min = Ein
            curr_c = c
    return curr_c
print(q9())
```

Hence according to the above code the

Correct answer is [e]

(b) [c]

```
for i in range(runs):
    curr_min = numpy.inf
    curr_c = 0
    for c in [0.0001, 0.001, 0.01, 0.1, 1]:
        evc = Evc(numpy.array(binary_train), numpy.array(x_train), 10, c, 2)
        if evc < curr_min:
            curr_min = evc
            curr_c = c
    data = list(zip(binary_train, x_train))
    random.shuffle(data)
    binary_train, x_train = list(zip(*data))
    wins[curr_c][0] += 1
    wins[curr_c][1] += evc
return wins
```

```
def q2():
def q3():
def q4():
def q5(q):
def q6():
def q7():
def q8():
def q9():
def q10():
    binary_train = make_binary(1, digits_train)
    binary_test = make_binary(1, digits_test)
    curr_min = numpy.inf
    curr_c = 0
    for c in [0.01, 1, 100, 10 ** 4, 10 ** 6]:
        classifier = svm.SVC(C=c, kernel='rbf', degree=2, gamma=1.0, coef0=1.0)
        classifier.fit(X_train, binary_train)
        Eout = 1 - classifier.score(X_test, binary_test)
        if (Eout < curr_min):
            curr_min = Eout
            curr_c = c
    return curr_c
print(q10())
```

Hence according to the above code,

the correct answer is [c]