In [11]:
```
from sklearn import svm
import random
import numpy

def formatOneToOne(points, digit1, digit2):
    formattedInputs = []
    formattedOutputs = []

    for point in points:
        if point[0] == digit1:
            formattedInputs.append([point[1], point[2]])
            formattedOutputs.append(1)
        elif point[0] == digit2:
            formattedInputs.append([point[1], point[2]])
            formattedOutputs.append(-1)

    return [formattedInputs, formattedOutputs]



def formatOneToMany(points, digit):
    formattedInputs = []
    formattedOutputs = []

    for point in points:
        if point[0] == digit:
            formattedInputs.append([point[1], point[2]])
            formattedOutputs.append(1)
        else:
            formattedInputs.append([point[1], point[2]])
            formattedOutputs.append(-1)

    return [formattedInputs, formattedOutputs]
```

In [101]:
```
def doSoftMarginSVM(trainInputs, trainOutputs, testInputs, testOutputs
, c, Q):
    tool = svm.SVC(kernel='poly', C=c, degree=Q, gamma=1, coef0=1)
    tool.fit(trainInputs, trainOutputs)
    trainErrorCount = 0
    testErrorCount = 0
    for i in range(len(trainInputs)):
        if tool.predict([trainInputs[i]]) != trainOutputs[i]:
            trainErrorCount += 1
    for i in range(len(testInputs)):
        if tool.predict([testInputs[i]]) != testOutputs[i]:
            testErrorCount += 1
    return [trainErrorCount/len(trainInputs), testErrorCount/len(testI
nputs)]
```

In [56]:
```
#PROBLEM 2

#IMPORT DATA
trainstrs = []
teststrs = []
trainlines = open('features.train.txt', 'r')
testlines = open('features.test.txt', 'r')
for line in trainlines:
    trainstrs.append(line.split())
for line in testlines:
    teststrs.append(line.split())

trainpts = []
testpts = []
for stri in trainstrs:
    trainpts.append([float(stri[0]), float(stri[1]), float(stri[2])])
for stri in teststrs:
    testpts.append([float(stri[0]), float(stri[1]), float(stri[2])])

for digit in [0, 2, 4, 6, 8]:
    #FORMAT DATA
    temp = formatOneToMany(trainpts, digit)
    trainInputs = temp[0]
    trainOutputs = temp[1]
    temp = formatOneToMany(testpts, digit)
    testInputs = temp[0]
    testOutputs = temp[1]

    #print(trainInputs)
    #print(trainOutputs)
    #print(testInputs)
    #print(testOutputs)

    result = doSoftMarginSVM(trainInputs, trainOutputs, testInputs, te
stOutputs, 0.01, 2)
    print('E_in for ' + str(digit) + ' versus all: ' + str(result[0]))
```

```
E_in for 0 versus all: 0.10588396653408312
E_in for 2 versus all: 0.10026059525442327
E_in for 4 versus all: 0.08942531888629818
E_in for 6 versus all: 0.09107118365107666
E_in for 8 versus all: 0.07433822520916199
```

We can see that 0 versus all yields the highest E_in. Therefore the answer to question 2 is A.

In [57]:
```
#PROBLEM 2

#IMPORT DATA
trainstrs = []
teststrs = []
trainlines = open('features.train.txt', 'r')
testlines = open('features.test.txt', 'r')
for line in trainlines:
    trainstrs.append(line.split())
for line in testlines:
    teststrs.append(line.split())

trainpts = []
testpts = []
for stri in trainstrs:
    trainpts.append([float(stri[0]), float(stri[1]), float(stri[2])])
for stri in teststrs:
    testpts.append([float(stri[0]), float(stri[1]), float(stri[2])])

for digit in [1, 3, 5, 7, 9]:
    #FORMAT DATA
    temp = formatOneToMany(trainpts, digit)
    trainInputs = temp[0]
    trainOutputs = temp[1]
    temp = formatOneToMany(testpts, digit)
    testInputs = temp[0]
    testOutputs = temp[1]

    result = doSoftMarginSVM(trainInputs, trainOutputs, testInputs, te
stOutputs, 0.01, 2)
    print('E_in for ' + str(digit) + ' versus all: ' + str(result[0]))
```

```
E_in for 1 versus all: 0.014401316691811822
E_in for 3 versus all: 0.09024825126868742
E_in for 5 versus all: 0.07625840076807022
E_in for 7 versus all: 0.08846523110684405
E_in for 9 versus all: 0.08832807570977919
```

We can see that E_in is lowest for 1 versus all. Therefore the answer to question 3 is A.

In [59]:
```
def getSoftMarginSVMVectorNumber(trainInputs, trainOutputs, c, Q):
    tool = svm.SVC(kernel='poly', C=c, degree=Q, gamma=1, coef0=1)
    tool.fit(trainInputs, trainOutputs)
    return len(tool.support_)
```

In [60]:
```
#PROBLEM 4

#IMPORT DATA
trainstrs = []
teststrs = []
trainlines = open('features.train.txt', 'r')
testlines = open('features.test.txt', 'r')
for line in trainlines:
    trainstrs.append(line.split())
for line in testlines:
    teststrs.append(line.split())

trainpts = []
testpts = []
for stri in trainstrs:
    trainpts.append([float(stri[0]), float(stri[1]), float(stri[2])])
for stri in teststrs:
    testpts.append([float(stri[0]), float(stri[1]), float(stri[2])])

for digit in [0, 1]:
    #FORMAT DATA
    temp = formatOneToMany(trainpts, digit)
    trainInputs = temp[0]
    trainOutputs = temp[1]

    result = getSoftMarginSVMVectorNumber(trainInputs, trainOutputs, 0
.01, 2)
    print('Number of support vectors for ' + str(digit) + ' versus all
: ' + str(result))
```

```
Number of support vectors for 0 versus all: 2179
Number of support vectors for 1 versus all: 386
```

We can see that the difference is 2179 - 386 = 1793 which is closest to 1800, so the answer to question 4 is C.

In [62]:
```
#PROBLEM 5

#IMPORT DATA
trainstrs = []
teststrs = []
trainlines = open('features.train.txt', 'r')
testlines = open('features.test.txt', 'r')
for line in trainlines:
    trainstrs.append(line.split())
for line in testlines:
    teststrs.append(line.split())

trainpts = []
testpts = []
for stri in trainstrs:
    trainpts.append([float(stri[0]), float(stri[1]), float(stri[2])])
for stri in teststrs:
    testpts.append([float(stri[0]), float(stri[1]), float(stri[2])])

#FORMAT DATA
temp = formatOneToOne(trainpts, 1, 5)
trainInputs = temp[0]
trainOutputs = temp[1]
temp = formatOneToOne(testpts, 1, 5)
testInputs = temp[0]
testOutputs = temp[1]

for C in [0.001, 0.01, 0.1, 1]:
    print('C = ' + str(C) + ':')
    result = doSoftMarginSVM(trainInputs, trainOutputs, testInputs, te
stOutputs, C, 2)
    print('E_in: ' + str(result[0]))
    print('E_out: ' + str(result[1]))
    print('Number of support vectors: ' + str(getSoftMarginSVMVectorNu
mber(trainInputs, trainOutputs, C, 2)))
    print(' ')
```

```
C = 0.001:
E_in: 0.004484304932735426
E_out: 0.01650943396226415
Number of support vectors: 76

C = 0.01:
E_in: 0.004484304932735426
E_out: 0.018867924528301886
Number of support vectors: 34

C = 0.1:
E_in: 0.004484304932735426
E_out: 0.018867924528301886
Number of support vectors: 24

C = 1:
E_in: 0.0032030749519538757
E_out: 0.018867924528301886
Number of support vectors: 24
```

We can see that maximum C produces minimum E_in, so the answer to question 5 is D.

In [64]:
```
#PROBLEM 6

#IMPORT DATA
trainstrs = []
teststrs = []
trainlines = open('features.train.txt', 'r')
testlines = open('features.test.txt', 'r')
for line in trainlines:
    trainstrs.append(line.split())
for line in testlines:
    teststrs.append(line.split())

trainpts = []
testpts = []
for stri in trainstrs:
    trainpts.append([float(stri[0]), float(stri[1]), float(stri[2])])
for stri in teststrs:
    testpts.append([float(stri[0]), float(stri[1]), float(stri[2])])

#FORMAT DATA
temp = formatOneToOne(trainpts, 1, 5)
trainInputs = temp[0]
trainOutputs = temp[1]
temp = formatOneToOne(testpts, 1, 5)
testInputs = temp[0]
testOutputs = temp[1]

for Q in [2, 5]:
    print('Q = ' + str(Q))
    for C in [0.0001, 0.001, 0.01, 1]:
        print('C = ' + str(C) + ':')
        result = doSoftMarginSVM(trainInputs, trainOutputs, testInputs
, testOutputs, C, Q)
        print('E_in: ' + str(result[0]))
        print('E_out: ' + str(result[1]))
        print('Number of support vectors: ' + str(getSoftMarginSVMVect
orNumber(trainInputs, trainOutputs, C, Q)))
        print(' ')
    print(' ')
```

```
Q = 2
C = 0.0001:
E_in: 0.008968609865470852
E_out: 0.01650943396226415
Number of support vectors: 236

C = 0.001:
E_in: 0.004484304932735426
E_out: 0.01650943396226415
Number of support vectors: 76

C = 0.01:
E_in: 0.004484304932735426
E_out: 0.018867924528301886
Number of support vectors: 34

C = 1:
E_in: 0.0032030749519538757
E_out: 0.018867924528301886
Number of support vectors: 24


Q = 5
C = 0.0001:
E_in: 0.004484304932735426
E_out: 0.018867924528301886
Number of support vectors: 26

C = 0.001:
E_in: 0.004484304932735426
E_out: 0.02122641509433962
Number of support vectors: 25

C = 0.01:
E_in: 0.003843689942344651
E_out: 0.02122641509433962
Number of support vectors: 23

C = 1:
E_in: 0.0032030749519538757
E_out: 0.02122641509433962
Number of support vectors: 21
```

We can see that the number of support vectors is lower at C = 0.001 when Q = 5. Therefore the answer to problem 6 is B.

In [132]:
```
#PROBLEM 7 & 8

#IMPORT DATA
trainstrs = []
teststrs = []
trainlines = open('features.train.txt', 'r')
testlines = open('features.test.txt', 'r')
for line in trainlines:
    trainstrs.append(line.split())
for line in testlines:
    teststrs.append(line.split())

trainpts = []
testpts = []
for stri in trainstrs:
    trainpts.append([float(stri[0]), float(stri[1]), float(stri[2])])
for stri in teststrs:
    testpts.append([float(stri[0]), float(stri[1]), float(stri[2])])

#FORMAT DATA
temp = formatOneToOne(trainpts, 1, 5)
trainInputs = temp[0]
trainOutputs = temp[1]
temp = formatOneToOne(testpts, 1, 5)
testInputs = temp[0]
testOutputs = temp[1]

counts = [0, 0, 0, 0, 0]
avg_errors = [0,0,0,0,0]
iteration = 1
for repeat in range(100):
    iteration += 1
    errors = []
    for C in [0.0001, 0.001, 0.01, 0.1, 1]:
        totalError = 0
        a = list(range(len(trainInputs)))
        random.shuffle(a)
        for i in range(10):
            tempInputs = []
            tempOutputs = []
            validateInputs = []
            validateOutputs = []
            for k in range(len(a)):
                if k < (len(a))/10:
                    validateInputs.append(trainInputs[a[k]])
                    validateOutputs.append(trainOutputs[a[k]])
                else:
                    tempInputs.append(trainInputs[a[k]])
                    tempOutputs.append(trainOutputs[a[k]])
            result = doSoftMarginSVM(tempInputs, tempOutputs, validate
```

```
Inputs, validateOutputs, C, 2)
            totalError += result[1]/10.0
        errors.append(totalError)

    for i in range(5):
        avg_errors[i] += errors[i]/100.0

    if errors[0] <= errors[1] and errors[0] <= errors[2] and errors[0]
<= errors[3] and errors[0] <= errors[4]:
        counts[0] += 1
    elif errors[1] <= errors[2] and errors[1] <= errors[3] and errors[
1] <= errors[4]:
        counts[1] += 1
    elif errors[2] <= errors[3] and errors[2] <= errors[4]:
        counts[2] += 1
    elif errors[3] <= errors[4]:
        counts[3] += 1
    else:
        counts[4] += 1

print(counts)
print(avg_errors)
```

```
[27, 38, 15, 11, 9]
[0.009299363057324843, 0.004649681528662418, 0.005605095541401275, 0
.005159235668789808, 0.004777070063694267]
```

We can see that the most chosen C is the second one, which corresponds to C=0.001. Therefore the answer to question 7 is B. We can see that the average error corresponding to C=0.001 is closest to 0.005, so the answer to question 8 is C.

In [134]:
```
#PROBLEM 9 & 10

def doSoftMarginSVM2(trainInputs, trainOutputs, testInputs, testOutput
s, c):
    tool = svm.SVC(kernel='rbf', gamma=1, C=c)
    tool.fit(trainInputs, trainOutputs)
    trainErrorCount = 0
    testErrorCount = 0
    for i in range(len(trainInputs)):
        if tool.predict([trainInputs[i]]) != trainOutputs[i]:
            trainErrorCount += 1
    for i in range(len(testInputs)):
        if tool.predict([testInputs[i]]) != testOutputs[i]:
            testErrorCount += 1
    return [trainErrorCount/len(trainInputs), testErrorCount/len(testI
nputs)]
```

```
#IMPORT DATA
trainstrs = []
teststrs = []
trainlines = open('features.train.txt', 'r')
testlines = open('features.test.txt', 'r')
for line in trainlines:
    trainstrs.append(line.split())
for line in testlines:
    teststrs.append(line.split())

trainpts = []
testpts = []
for stri in trainstrs:
    trainpts.append([float(stri[0]), float(stri[1]), float(stri[2])])
for stri in teststrs:
    testpts.append([float(stri[0]), float(stri[1]), float(stri[2])])


temp = formatOneToOne(trainpts, 1, 5)
trainInputs = temp[0]
trainOutputs = temp[1]
temp = formatOneToOne(testpts, 1, 5)
testInputs = temp[0]
testOutputs = temp[1]

#print(trainInputs)
#print(trainOutputs)
#print(testInputs)
#print(testOutputs)

for C in [0.01, 1, 100, 10000, 1000000]:
    result = doSoftMarginSVM2(trainInputs, trainOutputs, testInputs, t
estOutputs, C)
    print('C = ' + str(C))
    print('E_in for 1 versus 5: ' + str(result[0]))
    print('E_out for 1 versus 5: ' + str(result[1]))
    print(' ')
```

```
C = 0.01
E_in for 1 versus 5: 0.003843689942344651
E_out for 1 versus 5: 0.02358490566037736

C = 1
E_in for 1 versus 5: 0.004484304932735426
E_out for 1 versus 5: 0.02122641509433962

C = 100
E_in for 1 versus 5: 0.0032030749519538757
E_out for 1 versus 5: 0.018867924528301886

C = 10000
E_in for 1 versus 5: 0.0025624599615631004
E_out for 1 versus 5: 0.02358490566037736

C = 1000000
E_in for 1 versus 5: 0.0006406149903907751
E_out for 1 versus 5: 0.02358490566037736
```

We can see that the lowest E_in occurs at C=1000000, so the answer to question 9 is E. We can see that the lowest E_out occurs at C=100, so the answer to question 10 is C.

In [ ]: