In [29]:

```
from sklearn import svm
import random
import numpy

def formatOneToOne2(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 formatOneToOne(points, digit1, digit2):
    formatted = []

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

    return formatted



def formatOneToMany2(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]

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

    return formatted;
```

In [15]:

```
# PROBLEM 7

import random
from scipy import linalg
import numpy

def doWeightDecayRun(lambdy, digit):
    #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
    trainpts = formatOneToMany(trainpts, digit)
    testpts = formatOneToMany(testpts, digit)

    #TRANSFORM POINTS
    #newtrainpoints = []
    #newtestpoints = []
    #for point in trainpts:
    #    newtrainpoints.append([point[0], point[1], point[0]**2, point
[1]**2, point[0]*point[1], \
    #                           numpy.abs(point[0] - point[1]), numpy.
abs(point[1] + point[0]), point[2]])
    #for point in testpts:
    #    newtestpoints.append([point[0], point[1], point[0]**2, point[
1]**2, point[0]*point[1], \
    #                          numpy.abs(point[0] - point[1]), numpy.abs(point[1] + point[0]), point[2]])

    #APPLY ALGORITHM
    xmat = []
    ymat = []
    for point in trainpts:
        xmat.append([1, point[0], point[1]])
        ymat.append(point[2])
    xmat = numpy.asarray(xmat)
    ymat = numpy.asarray(ymat)


    lambd = lambdy
    ztz = numpy.matmul(numpy.transpose(xmat), xmat)
    pluslami = numpy.add(ztz, numpy.identity(len(xmat[0])) * lambd)
    invzt = numpy.matmul(numpy.linalg.inv(pluslami), numpy.transpose(xmat))
    weight = numpy.matmul(invzt, ymat)

    #COMPUTE IN-SAMPLE ERROR
    wrongtraincount = 0
    for point in trainpts:
        if numpy.sign(weight[0] + weight[1]*point[0] + weight[2]*point[1]) != point[2]:
            wrongtraincount += 1
    wrongtraincount /= len(trainpts)

    #COMPUTE OUT-OF-SAMPLE ERROR
    wrongtestcount = 0
    for point in testpts:
        if numpy.sign(weight[0] + weight[1]*point[0] + weight[2]*point[1]) != point[2]:
            wrongtestcount += 1
    wrongtestcount /= len(testpts)

    return [wrongtraincount, wrongtestcount]
```

In [20]:

```
# PROBLEM 7
for digit in [5,6,7,8,9]:
    print('For ' + str(digit) + ' versus all, [E_in, E_out]: ')
    print(doWeightDecayRun(1, digit))
    print()
```

```
For 5 versus all, [E_in, E_out]:
[0.07625840076807022, 0.07972097658196313]

For 6 versus all, [E_in, E_out]:
[0.09107118365107666, 0.08470353761833582]

For 7 versus all, [E_in, E_out]:
[0.08846523110684405, 0.07324364723467862]

For 8 versus all, [E_in, E_out]:
[0.07433822520916199, 0.08271051320378675]

For 9 versus all, [E_in, E_out]:
[0.08832807570977919, 0.08819133034379671]
```

We can see that the lowest E_in occurs with 8 versus all. Therefore the answer to question 7 is D.

In [26]:

```
# PROBLEM 8

import random
from scipy import linalg
import numpy

def doWeightDecayRunTransformed(lambdy, digit):
    #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
    trainpts = formatOneToMany(trainpts, digit)
    testpts = formatOneToMany(testpts, digit)

    #TRANSFORM POINTS
    newtrainpoints = []
    newtestpoints = []
    for point in trainpts:
        newtrainpoints.append([point[0], point[1], point[0]*point[1],
point[0]**2, point[1]**2, point[2]])
    for point in testpts:
        newtestpoints.append([point[0], point[1], point[0]*point[1], p
oint[0]**2, point[1]**2, point[2]])

    #APPLY ALGORITHM
    xmat = []
    ymat = []
    for point in newtrainpoints:
        xmat.append([1, point[0], point[1], point[2], point[3], point[
4]])
        ymat.append(point[5])
    xmat = numpy.asarray(xmat)
    ymat = numpy.asarray(ymat)


    lambd = lambdy
    ztz = numpy.matmul(numpy.transpose(xmat), xmat)
    pluslami = numpy.add(ztz, numpy.identity(len(xmat[0])) * lambd)
    invzt = numpy.matmul(numpy.linalg.inv(pluslami), numpy.transpose(x
mat))
    weight = numpy.matmul(invzt, ymat)

    #COMPUTE IN-SAMPLE ERROR
    wrongtraincount = 0
    for point in newtrainpoints:
        if numpy.sign(weight[0] + weight[1]*point[0] + weight[2]*point
[1] + weight[3]*point[2]\
                      + weight[4]*point[3] + weight[5]*point[4]) != po
int[5]:
            wrongtraincount += 1
    wrongtraincount /= len(newtrainpoints)

    #COMPUTE OUT-OF-SAMPLE ERROR
    wrongtestcount = 0
    for point in newtestpoints:
        if numpy.sign(weight[0] + weight[1]*point[0] + weight[2]*point
[1] + weight[3]*point[2]\
                              + weight[4]*point[3] + weight[5]*point[4]) != po
int[5]:
                wrongtestcount += 1
    wrongtestcount /= len(newtestpoints)

    return [wrongtraincount, wrongtestcount]
```

In [27]:

```
# PROBLEM 8
for digit in [0,1,2,3,4]:
    print('For ' + str(digit) + ' versus all, [E_in, E_out]: ')
    print(doWeightDecayRunTransformed(1, digit))
    print()
```

```
For 0 versus all, [E_in, E_out]:
[0.10231792621039638, 0.10662680617837568]

For 1 versus all, [E_in, E_out]:
[0.012343985735838706, 0.021923268560039861]

For 2 versus all, [E_in, E_out]:
[0.10026059525442327, 0.09865470852017937]

For 3 versus all, [E_in, E_out]:
[0.09024825126868742, 0.08271051320378675]

For 4 versus all, [E_in, E_out]:
[0.08942531888629818, 0.09965122072745392]
```

We can see that the lowest E_out is for 1 versus all, so the answer to question 8 is B.

In [28]:

```
# PROBLEM 9
for digit in [0,1,2,3,4,5,6,7,8,9]:
    print('For ' + str(digit) + ' versus all, [E_in, E_out]: ')
    print('No transformation: ')
    print(doWeightDecayRun(1, digit))
    print('Transformation: ')
    print(doWeightDecayRunTransformed(1, digit))
    print()
```

```
For 0 versus all, [E_in, E_out]:
No transformation:
[0.10931285146070498, 0.11509715994020926]
Transformation:
[0.10231792621039638, 0.10662680617837568]

For 1 versus all, [E_in, E_out]:
No transformation:
```

```
[0.01522424907420107, 0.02242152466367713]
Transformation:
[0.012343985735838706, 0.02192326856003986]

For 2 versus all, [E_in, E_out]:
No transformation:
[0.10026059525442327, 0.09865470852017937]
Transformation:
[0.10026059525442327, 0.09865470852017937]

For 3 versus all, [E_in, E_out]:
No transformation:
[0.09024825126868742, 0.08271051320378675]
Transformation:
[0.09024825126868742, 0.08271051320378675]

For 4 versus all, [E_in, E_out]:
No transformation:
[0.08942531888629818, 0.09965122072745392]
Transformation:
[0.08942531888629818, 0.09965122072745392]

For 5 versus all, [E_in, E_out]:
No transformation:
[0.07625840076807022, 0.07972097658196313]
Transformation:
[0.07625840076807022, 0.07922272047832586]

For 6 versus all, [E_in, E_out]:
No transformation:
[0.09107118365107666, 0.08470353761833582]
Transformation:
[0.09107118365107666, 0.08470353761833582]

For 7 versus all, [E_in, E_out]:
No transformation:
[0.08846523110684405, 0.07324364723467862]
Transformation:
[0.08846523110684405, 0.07324364723467862]

For 8 versus all, [E_in, E_out]:
No transformation:
[0.07433822520916199, 0.08271051320378675]
Transformation:
[0.07433822520916199, 0.08271051320378675]

For 9 versus all, [E_in, E_out]:
No transformation:
[0.08832807570977919, 0.08819133034379671]
Transformation:
```

```
[0.08832807570977919, 0.08819133034379671]
```

We can see that E_out of 5 versus all is improved a small amount but less than 5%, so the answer to question 9 is E.

In [31]:
```
# PROBLEM 10

import random
from scipy import linalg
import numpy

def doWeightDecayRunTransformedVersus(lambdy, digit1, digit2):
    #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
    trainpts = formatOneToOne(trainpts, digit1, digit2)
    testpts = formatOneToOne(testpts, digit1, digit2)

    #TRANSFORM POINTS
    newtrainpoints = []
    newtestpoints = []
    for point in trainpts:
        newtrainpoints.append([point[0], point[1], point[0]*point[1],
point[0]**2, point[1]**2, point[2]])
    for point in testpts:
        newtestpoints.append([point[0], point[1], point[0]*point[1], p
oint[0]**2, point[1]**2, point[2]])

    #APPLY ALGORITHM
    xmat = []
    ymat = []
    for point in newtrainpoints:
        xmat.append([1, point[0], point[1], point[2], point[3], point[
4]])
        ymat.append(point[5])
    xmat = numpy.asarray(xmat)
    ymat = numpy.asarray(ymat)


    lambd = lambdy
    ztz = numpy.matmul(numpy.transpose(xmat), xmat)
    pluslami = numpy.add(ztz, numpy.identity(len(xmat[0])) * lambd)
    invzt = numpy.matmul(numpy.linalg.inv(pluslami), numpy.transpose(x
mat))
    weight = numpy.matmul(invzt, ymat)

    #COMPUTE IN-SAMPLE ERROR
    wrongtraincount = 0
    for point in newtrainpoints:
        if numpy.sign(weight[0] + weight[1]*point[0] + weight[2]*point
[1] + weight[3]*point[2]\
                      + weight[4]*point[3] + weight[5]*point[4]) != po
int[5]:
            wrongtraincount += 1
    wrongtraincount /= len(newtrainpoints)

    #COMPUTE OUT-OF-SAMPLE ERROR
    wrongtestcount = 0
    for point in newtestpoints:
        if numpy.sign(weight[0] + weight[1]*point[0] + weight[2]*point
[1] + weight[3]*point[2]\
                      + weight[4]*point[3] + weight[5]*point[4]) != po
int[5]:
            wrongtestcount += 1
    wrongtestcount /= len(newtestpoints)

    return [wrongtraincount, wrongtestcount]
```

In [34]:

```
# PROBLEM 10
for lambdy in [1, 0.01]:
    print('For 1 versus 5, lambda = ' + str(lambdy) + ', [E_in, E_out]
: ')
    print(doWeightDecayRunTransformedVersus(lambdy, 1, 5))
    print()
```

```
For 1 versus 5, lambda = 1, [E_in, E_out]:
[0.005124919923126201, 0.025943396226415096]

For 1 versus 5, lambda = 0.01, [E_in, E_out]:
[0.004484304932735426, 0.02830188679245283]
```

We can see that from lambda=1 to lambda=0.01, E_in decreases but E_out decreases. Therefore overfitting occurs from lambda=1 to lambda=0.01 so the answer to question 10 is A.

In [ ]:

```
```