In [5]:
```
#PROBLEM 2

import random
from scipy import linalg
import numpy

def doNonLinRegRun():
    #IMPORT POINTS
    trainstrs = []
    teststrs = []
    trainlines = open('in.dta.txt', 'r')
    testlines = open('out.dta.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])
])

    #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.a
bs(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.ab
s(point[1] + point[0]), point[2]])

    #APPLY ALGORITHM
    xmat = []
    ymat = []
    for point in newtrainpoints:
        xmat.append([1, point[0], point[1], point[2], point[3], point[
4], point[5], point[6]])
        ymat.append(point[7])
    xmat = numpy.asarray(xmat)
```

```
    ymat = numpy.asarray(ymat)

    pseudoinverse = linalg.pinv(xmat)
    weight = numpy.matmul(pseudoinverse, 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] + weight
[6]*point[5] + weight[7]*point[6]) != point[7]:
            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] + weight
[6]*point[5] + weight[7]*point[6]) != point[7]:
            wrongtestcount += 1
    wrongtestcount /= len(newtestpoints)

    return [wrongtraincount, wrongtestcount]
```

In [8]:
```
print('In-sample error, out-of-sample error respectively: ' + str(doNo
nLinRegRun()))
```

```
In sample error, out of sample error respectively: [0.02857142857142
857, 0.084]
```

Because the in sample error and out of sample error are closest to 0.03 and 0.08, the answer to question 2 is A.

In [14]:
```
#PROBLEM 3

import random
from scipy import linalg
import numpy

def doWeightDecayRun(k):
    #IMPORT POINTS
    trainstrs = []
    teststrs = []
    trainlines = open('in.dta.txt', 'r')
    testlines = open('out.dta.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])
])

    #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.a
bs(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.ab
s(point[1] + point[0]), point[2]])

    #APPLY ALGORITHM
    xmat = []
    ymat = []
    for point in newtrainpoints:
        xmat.append([1, point[0], point[1], point[2], point[3], point[
4], point[5], point[6]])
        ymat.append(point[7])
    xmat = numpy.asarray(xmat)
    ymat = numpy.asarray(ymat)


    lambd = 10 ** k
    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] + weight
[6]*point[5] + weight[7]*point[6]) != point[7]:
            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] + weight
[6]*point[5] + weight[7]*point[6]) != point[7]:
            wrongtestcount += 1
    wrongtestcount /= len(newtestpoints)

    return [wrongtraincount, wrongtestcount]
```

In [16]:
```
print('In-sample error, out-of-sample error respectively: ' + str(doWe
ightDecayRun(-3)))
```

```
In sample error, out of sample error respectively: [0.02857142857142
857, 0.08]
```

Because the in sample error and out of sample error are closest to 0.03 and 0.08, the answer to question 3 is D.

In [17]:
```
print('In-sample error, out-of-sample error respectively: ' + str(doWe
ightDecayRun(3)))
```

```
In sample error, out of sample error respectively: [0.37142857142857
144, 0.436]
```

Because the in sample error and out of sample error are closest to 0.04 and 0.04, the answer to question 4 is E.

In [19]:
```
print('In-sample error, out-of-sample error respectively (k = 2): ' +
str(doWeightDecayRun(2)))
print('In-sample error, out-of-sample error respectively (k = 1): ' +
str(doWeightDecayRun(1)))
print('In-sample error, out-of-sample error respectively (k = 0): ' +
str(doWeightDecayRun(0)))
print('In-sample error, out-of-sample error respectively (k = -1): ' +
str(doWeightDecayRun(-1)))
print('In-sample error, out-of-sample error respectively (k = -2): ' +
str(doWeightDecayRun(-2)))
```

```
In sample error, out of sample error respectively (k = 2): [0.2, 0.2
28]
In sample error, out of sample error respectively (k = 1): [0.057142
85714285714, 0.124]
In sample error, out of sample error respectively (k = 0): [0.0, 0.0
92]
In sample error, out of sample error respectively (k = -1): [0.02857
142857142857, 0.056]
In sample error, out of sample error respectively (k = -2): [0.02857
142857142857, 0.084]
```

We can see that the smallest out-of-sample classification error occurs when k = -1. So the answer to question 5 is D.

In [25]:
```
print('In-sample error, out-of-sample error respectively (k = 8): ' +
str(doWeightDecayRun(8)))
print('In-sample error, out-of-sample error respectively (k = 4): ' +
str(doWeightDecayRun(4)))
print('In-sample error, out-of-sample error respectively (k = 3): ' +
str(doWeightDecayRun(3)))
print('In-sample error, out-of-sample error respectively (k = 2): ' +
str(doWeightDecayRun(2)))
print('In-sample error, out-of-sample error respectively (k = 1): ' +
str(doWeightDecayRun(1)))
print('In-sample error, out-of-sample error respectively (k = 0): ' +
str(doWeightDecayRun(0)))
print('In-sample error, out-of-sample error respectively (k = -1): ' +
str(doWeightDecayRun(-1)))
print('In-sample error, out-of-sample error respectively (k = -2): ' +
str(doWeightDecayRun(-2)))
print('In-sample error, out-of-sample error respectively (k = -3): ' +
str(doWeightDecayRun(-3)))
print('In-sample error, out-of-sample error respectively (k = -4): ' +
str(doWeightDecayRun(-4)))
print('In-sample error, out-of-sample error respectively (k = -8): ' +
str(doWeightDecayRun(-8)))
```

```
In-sample error, out-of-sample error respectively (k = 8): [0.428571
42857142855, 0.456]
In-sample error, out-of-sample error respectively (k = 4): [0.428571
42857142855, 0.452]
In-sample error, out-of-sample error respectively (k = 3): [0.371428
57142857144, 0.436]
In-sample error, out-of-sample error respectively (k = 2): [0.2, 0.2
28]
In-sample error, out-of-sample error respectively (k = 1): [0.057142
85714285714, 0.124]
In-sample error, out-of-sample error respectively (k = 0): [0.0, 0.0
92]
In-sample error, out-of-sample error respectively (k = -1): [0.02857
142857142857, 0.056]
In-sample error, out-of-sample error respectively (k = -2): [0.02857
142857142857, 0.084]
In-sample error, out-of-sample error respectively (k = -3): [0.02857
142857142857, 0.08]
In-sample error, out-of-sample error respectively (k = -4): [0.02857
142857142857, 0.084]
In-sample error, out-of-sample error respectively (k = -8): [0.02857
142857142857, 0.084]
```

We can see that the lowest the out-of-sample error gets to is 0.056 (for k = -1). Therefore the answer to question 6 is B.

In [ ]: