In [4]:

```
import random
def flipCoin():
    temp = random.randint(0,1)
    return temp == 0
```

In [40]:

```
def runCoins(numOfCoins):
    randCoin = random.randint(0,999)
    v1 = 0
    vrand = 0
    vmin = 1
    for a in range(int(numOfCoins)):
        minchecker = 0
        for b in range(10):
            if flipCoin():
                if a == 0:
                    v1 += 1
                if a == randCoin:
                    vrand += 1
                minchecker += 1
        if a == 0:
            v1 /= 10
        if a == randCoin:
            vrand /= 10.0
        minchecker /= 10.0
        if minchecker < vmin:
            vmin = minchecker
    return v1, vrand, vmin
```

In [43]:

```
totalTests = 100000
tot1 = 0
totrand = 0
totmin = 0
for a in range(int(totalTests)):
    result = runCoins(1000.0)
    tot1 += result[0]
    totrand += result[1]
    totmin += result[2]
tot1 /= totalTests
totrand /= totalTests
totmin /= totalTests

print("Average first coin: " + str(tot1))
print("Average random coin: " + str(totrand))
print("Average min coin: " + str(totmin))
```

```
Average first coin: 0.5002420000000081
Average random coin: 0.4993450000000041
Average min coin: 0.0376639999999767
```

Justification for #1 just above. Vmin is 0.037664 so clearly vmin is closest to 0.01 (B).

In [302]:

```
import random
from scipy import linalg
import numpy

def doLinRegRun(NUMPOINTS):
    centerpt = [random.uniform(-1,1), random.uniform(-1,1)]
    otherpt = [random.uniform(-1,1), random.uniform(-1,1)]

    points = []

    for a in range(int(NUMPOINTS)):
        points.append([random.uniform(-1,1), random.uniform(-1,1)])

    for point in points:
        if point[1] > (otherpt[1]-centerpt[1])/(otherpt[0]-centerpt[0])*(point[0
]-centerpt[0]):
            point.append(1)
        else:
            point.append(-1)

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

```
    xmat = numpy.asarray(xmat)

    ymat = numpy.asarray(ymat)
    #ymat = numpy.matrix.transpose(ymat)
    #xmat = numpy.matrix.transpose(xmat)
    #print(xmat)
    #print(ymat)

    #Compute pseudo inverse
    pseudoinverse = linalg.pinv(xmat)
    #print(pseudoinverse)
    weight = pseudoinverse.dot(ymat)
    #print(weight)

    w0 = numpy.mean(ymat)

    weight = numpy.asarray([weight[0], weight[1], w0])

    wrongcount = 0
    for point in points:
        if numpy.sign(weight[2] + weight[0]*point[0] + weight[1]*point[1]) != po
int[2]:
            wrongcount += 1
    wrongcount /= NUMPOINTS

    return [weight, [centerpt, otherpt], points, wrongcount]
```

In [303]:

```
NUMTESTS = 1000.0

#Testing for question 5 and 6

eInAvg = 0
eOutAvg = 0

for a in range(int(NUMTESTS)):
    mistake = 0
    result = doLinRegRun(100.0)
    for a in range(1000):
        randomPoint = [random.uniform(-1,1), random.uniform(-1,1)]
        if randomPoint[1] > (result[1][1][1]-result[1][0][1])/(result[1][1][0]-r
esult[1][0][0])*\
        (randomPoint[0]-result[1][0][0]):
            if result[0][0]*randomPoint[0] + result[0][1]*randomPoint[1] + resul
t[0][2] < 0:
                mistake += 1
        else:
            if result[0][0]*randomPoint[0] + result[0][1]*randomPoint[1] + resul
t[0][2] > 0:
                mistake += 1
    eOutAvg += mistake/1000.0
    eInAvg += result[3]

eInAvg /= NUMTESTS
eOutAvg /= NUMTESTS
print("E_in average: " + str(eInAvg))
print("E_out average: " + str(eOutAvg))
```

```
E_in average: 0.040450000000000006
E_out average: 0.04904599999999985
```

The value of E_in is closest to 0.01, so the answer to question 5 is C.

The value of E_out is closest to 0.01, so the answer to question 6 is C.

In [304]:

```
def isPointPositive(point, line):
    result = (point[0] * line[0]) + (point[1] * line[1]) + line[2]
    if result == 0:
        return 3
    elif result > 0:
        return True
    else:
        return False

def doPLARun(points, initWeight, line):
```

```
#centerpt = [random.uniform(-1,1), random.uniform(-1,1)]
#otherpt = [random.uniform(-1,1), random.uniform(-1,1)]

centerpt = line[0]
otherpt = line[1]

points = points

#for a in range(NUMPOINTS):
#    points.append([random.uniform(-1,1), random.uniform(-1,1)])

linevect = initWeight
counter = 0
for a in range(1000):
    misclassified = []
    for point in points:
        score = isPointPositive(point, linevect)
        if score == 3:
            misclassified.append(point)
        elif score:
            if point[2] == -1:
                misclassified.append(point)
        elif not score:
            if point[2] == 1:
                misclassified.append(point)
        else:
            misclassified.append(point)

        #print("Iteration number " + str(counter))
        #print("Number of misclassified points: " + str(len(misclassified)))

        if len(misclassified) == 0:
            break

        counter += 1
        targetpoint = random.choice(misclassified)

        linevect[0] += targetpoint[2]*targetpoint[0]
        linevect[1] += targetpoint[2]*targetpoint[1]
        linevect[2] += targetpoint[2]

    #print(counter)

    wrongcount = 0
    for a in range(10000):
        point1 = [random.uniform(-1,1), random.uniform(-1,1)]
        if point1[1] > (otherpt[1]-centerpt[1])/(otherpt[0]-centerpt[0])*(point1
[0]-centerpt[0]):
            point1.append(1)
        else:
            point1.append(-1)
        score = isPointPositive(point1, linevect)
        if score == True and point1[2] == -1:
            wrongcount+=1
```

```
        elif score == False and point1[2] == 1:

            wrongcount+=1

    #print(wrongcount/10000.0)
    return [counter, wrongcount/10000.0]
```

In [306]:

```
#Testing for problem 7

tot = 0
for a in range(1000):
    result = doRun(10)
    res2 = doPLARun(result[2], result[0], result[1])
    tot += res2[0]
tot /= 1000
print("Average iterations: " + str(tot))
```

```
Average iterations: 5.227
```

The average number of iterations is closest to 1 so the answer to question 7 is A.

In [328]:

```
import random
from scipy import linalg
import numpy

def doLinRegRun(NUMPOINTS, NOISE):
    #GENERATE LINE
    centerpt = [random.uniform(-1,1), random.uniform(-1,1)]
    otherpt = [random.uniform(-1,1), random.uniform(-1,1)]

    #GENERATE POINTS
    points = []
    for a in range(int(NUMPOINTS)):
        points.append([random.uniform(-1,1), random.uniform(-1,1)])

    #LABEL POINTS
    for point in points:
        point.append(numpy.sign(point[0]**2 + point[1]**2 - 0.6))

    #INSERT NOISE
    maxnoises = int(NOISE * NUMPOINTS)
    countnoises = 0
    for point in points:
        if countnoises >= maxnoises:
            break
        if random.uniform(0,1) < 0.1:
            point[2] = -point[2]
            countnoises += 1

    #APPLY ALGORITHM
```

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

    #Compute pseudo inverse
    pseudoinverse = linalg.pinv(xmat)
    #print(pseudoinverse)
    weight = pseudoinverse.dot(ymat)
    #print(weight)

    w0 = numpy.mean(ymat)

    weight = numpy.asarray([weight[0], weight[1], w0])

    wrongcount = 0
    for point in points:
        if numpy.sign(weight[2] + weight[0]*point[0] + weight[1]*point[1]) != po
int[2]:
            wrongcount += 1
    wrongcount /= NUMPOINTS

    return [weight, [centerpt, otherpt], points, wrongcount]
```

In [329]:

```
NUMTESTS = 1000.0

#Testing for question 8

eInAvg = 0

for a in range(int(NUMTESTS)):
    result = doLinRegRun(1000.0, 0.1)
    eInAvg += result[3]

eInAvg /= NUMTESTS
print("E_in average: " + str(eInAvg))
```

```
E_in average: 0.5057119999999996
```

E_in is closest to 0.5 so the answer to question 8 is D.

In [343]:

```
import random
from scipy import linalg
import numpy
```

```
def doNonLinRegRun(NUMPOINTS, NOISE):
    #GENERATE POINTS
    points = []
    for a in range(int(NUMPOINTS)):
        points.append([random.uniform(-1,1), random.uniform(-1,1)])

    #LABEL POINTS
    for point in points:
        point.append(numpy.sign(point[0]**2 + point[1]**2 - 0.6))

    #TRANSFORM POINTS
    newpoints = []
    for point in points:
        newpoints.append([point[0], point[1], point[0]*point[1], point[0]**2, po
int[1]**2, point[2]])
    points = newpoints

    #INSERT NOISE
    maxnoises = int(NOISE * NUMPOINTS)
    countnoises = 0
    for point in points:
        if countnoises >= maxnoises:
            break
        if random.uniform(0,1) < 0.1:
            point[5] = -point[5]
            countnoises += 1

    #APPLY ALGORITHM
    xmat = []
    ymat = []
    for point in points:
        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)

    #Compute pseudo inverse
    pseudoinverse = linalg.pinv(xmat)
    #print(pseudoinverse)
    weight = numpy.matmul(pseudoinverse, ymat)
    #print(weight)

    wrongcount = 0
    for point in points:
        if numpy.sign(weight[0] + weight[1]*point[0] + weight[2]*point[1] + weig
ht[3]*point[2] + \
                      weight[4]*point[3] + weight[5]*point[4]) != point[5]:
            wrongcount += 1
    wrongcount /= NUMPOINTS

    return [weight, [centerpt, otherpt], points, wrongcount]
```

In [355]:

```
#Testing for question 9
totals = [0,0,0,0,0,0]
for a in range(1000):
    result = doNonLinRegRun(1000.0, 0.1)[0]
    totals[0] = totals[0] + result[0]
    totals[1] = totals[1] + result[1]
    totals[2] = totals[2] + result[2]
    totals[3] = totals[3] + result[3]
    totals[4] = totals[4] + result[4]
    totals[5] = totals[5] + result[5]
totals[0] = totals[0]/1000.0
totals[1] = totals[1]/1000.0
totals[2] = totals[2]/1000.0
totals[3] = totals[3]/1000.0
totals[4] = totals[4]/1000.0
totals[5] = totals[5]/1000.0
print("Average weights for linear regression:")
print(totals)
```

```
Average weights for linear regression:
[-1.003068058596127, 0.0003472347653361053, 0.0004186320096207487, -
0.0009046897250025371, 1.5738474636217532, 1.5773609207045522]
```

These weights closely correspond to those of option A (-1, -0.05, 0.08, 0.13, 1.5, 1.5). Therefore the answer to question 9 is A.

In [359]:

```
NUMTESTS = 1000.0

#Testing for question 10

eOutAvg = []

for a in range(int(NUMTESTS)):
    mistake = 0
    res = doNonLinRegRun(1000.0, 0.1)[0]
    for a in range(1000):
        rP = [random.uniform(-1,1), random.uniform(-1,1)]
        if random.uniform(0,1) > 0.1:
            rP.append(numpy.sign(rP[0]**2 + rP[1]**2 - 0.6))
        else:
            rP.append(-numpy.sign(rP[0]**2 + rP[1]**2 - 0.6))
        if rP[2] != numpy.sign(res[0] + res[1]*rP[0] + res[2]*rP[1] + res[3]*rP[
0]*rP[1] + \
                               res[4]*rP[0]**2 + res[5]*rP[1]**2):
            mistake += 1
    eOutAvg.append(mistake/1000.0)

print("E_out average: " + str(numpy.mean(eOutAvg)))
```

```
E_out average: 0.12596700000000002
```

Because E_out is closest to 0.1, the answer to question 10 is B.

In [ ]: