In [11]:
```
# PROBLEM 5

import numpy

def errorOf(w):
    return (w[0]*numpy.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))**2.0
def gradXOf(w):
    return 2.0*(numpy.exp(w[1]) + 2.0*w[1]*numpy.exp(-w[0]))*(w[0]*num
py.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))
def gradYOf(w):
    return 2.0*(w[0]*numpy.exp(w[1]) - 2.0*numpy.exp(-w[0]))*(w[0]*num
py.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))

w1 = 1.0
w2 = 1.0

count = 0
while errorOf([w1,w2]) > 10e-14:
    temp = gradYOf([w1,w2])
    w1 -= 0.1*gradXOf([w1,w2])
    w2 -= 0.1*temp
    count += 1

print("Number of iterations: " + str(count))
```

```
Number of iterations: 10
```

The number of iterations for E to fall below 10e-14 fpr the first time is 10 iterations. Therefore the answer is D.

In [17]:
```
# PROBLEM 6

import numpy

def errorOf(w):
    return (w[0]*numpy.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))**2.0
def gradXOf(w):
    return 2.0*(numpy.exp(w[1]) + 2.0*w[1]*numpy.exp(-w[0]))*(w[0]*num
py.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))
def gradYOf(w):
    return 2.0*(w[0]*numpy.exp(w[1]) - 2.0*numpy.exp(-w[0]))*(w[0]*num
py.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))

w1 = 1.0
w2 = 1.0

count = 0
while errorOf([w1,w2]) > 10e-14:
    temp = gradYOf([w1,w2])
    w1 -= 0.1*gradXOf([w1,w2])
    w2 -= 0.1*temp
    count += 1

adist = numpy.sqrt((w1-1.0)**2 + (w2-1.0)**2)
print(adist)
bdist = numpy.sqrt((w1-0.713)**2 + (w2-0.045)**2)
print(bdist)
cdist = numpy.sqrt((w1-0.016)**2 + (w2-0.112)**2)
print(cdist)
ddist = numpy.sqrt((w1+0.083)**2 + (w2-0.029)**2)
print(ddist)
edist = numpy.sqrt((w1-0.045)**2 + (w2-0.024)**2)
print(edist)
```

```
1.365717886924672
0.6685948857743971
0.0926123232021653
0.12783573228217807
0.0002669218610597792
```

The shortest distance is from the (u,v) pair in choice E. Therefore the answer is E.

In [5]:
```
# PROBLEM 7

import numpy

def errorOf(w):
    return (w[0]*numpy.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))**2.0
def gradXOf(w):
    return 2.0*(numpy.exp(w[1]) + 2.0*w[1]*numpy.exp(-w[0]))*(w[0]*num
py.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))
def gradYOf(w):
    return 2.0*(w[0]*numpy.exp(w[1]) - 2.0*numpy.exp(-w[0]))*(w[0]*num
py.exp(w[1]) - 2.0*w[1]*numpy.exp(-w[0]))

w1 = 1.0
w2 = 1.0

count = 0
while count<15:
    w1 -= 0.1*gradXOf([w1,w2])
    w2 -= 0.1*gradYOf([w1,w2])
    print("Error: " + str(errorOf([w1,w2])))
    count += 1
```

```
Error: 34.29016311234976
Error: 0.5341425913722001
Error: 0.4326608273241937
Error: 0.3650397350187306
Error: 0.31646807535966437
Error: 0.2797634230640926
Error: 0.25098631167528807
Error: 0.22778329894427699
Error: 0.20865669572438028
Error: 0.19260565861364648
Error: 0.17893474840754628
Error: 0.167145054343084
Error: 0.15686898732952279
Error: 0.14782952252409787
Error: 0.13981379199615315
```

The error after 15 full iterations is closest to 10e-1. Therefore the answer is A.

In [33]:
```
# PROBLEMS 8 AND 9

import numpy
import random

def distBetween(pt1, pt2):
```

```
    return numpy.sqrt( (pt1[0] - pt2[0])**2 + (pt1[1] - pt2[1])**2 +(p
t1[2] - pt2[2])**2 )

def gradError(pt, wt):
    p1 = -pt[2]/(1 + numpy.exp(pt[2]*(wt[0] + wt[1]*pt[0] + wt[2]*pt[1
])))
    p2 = -pt[2]*pt[0]/(1 + numpy.exp(pt[2]*(wt[0] + wt[1]*pt[0] + wt[2
]*pt[1])))
    p3 = -pt[2]*pt[1]/(1 + numpy.exp(pt[2]*(wt[0] + wt[1]*pt[0] + wt[2
]*pt[1])))
    return [p1, p2, p3]

avgEpoch = 0
avgEout = 0
NUMRUNS = 5
for a in range(NUMRUNS):

    # GENERATE POINTS
    centerpt = [random.uniform(-1,1), random.uniform(-1,1)]
    otherpt = [random.uniform(-1,1), random.uniform(-1,1)]
    points = []

    for a in range(100):
        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)


    w = [0.0, 0.0, 0.0]
    oldw = [5.0, 5.0, 5.0]

    # TRAIN
    epochCount = 0
    while distBetween(w, oldw) > 0.01:
        epochCount += 1
        order = list(range(100))
        random.shuffle(order)
        oldw[0] = w[0]
        oldw[1] = w[1]
        oldw[2] = w[2]
        for a in order:
            gerror = gradError(points[a], w)
            w[0] -= 0.01*gerror[0]
            w[1] -= 0.01*gerror[1]
            w[2] -= 0.01*gerror[2]
```

```
    avgEpoch += epochCount

    # COMPUTE E_OUT
    testpoints = []

    for a in range(10000):
        testpoints.append([random.uniform(-1,1), random.uniform(-1,1)]
)
    for point in testpoints:
        if point[1] > (otherpt[1]-centerpt[1])/(otherpt[0]-centerpt[0]
)*(point[0]-centerpt[0]):
            point.append(1)
        else:
            point.append(-1)
    wrongcount = 0
    for point in testpoints:
        wrongcount += numpy.log(1+numpy.exp(-point[2]*(w[0] + w[1]*poi
nt[0] + w[2]*point[1])))
    wrongcount /= 10000
    avgEout += wrongcount

print("Avg epochs: " + str(avgEpoch/NUMRUNS))
print("Avg Eout: " + str(avgEout/NUMRUNS))
```

```
Avg epochs: 350.8
Avg Eout: 0.10409601545857605
```

E_out is closest to 0.100 for N = 100. Therefore the answer to question 8 is D. On average, 350 epochs are taken to converge. Therefore the answer to question 9 is A.

In [ ]: