HW1

1) Signum function:- $sgn(x) = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases}$

-1 - False

1 - true

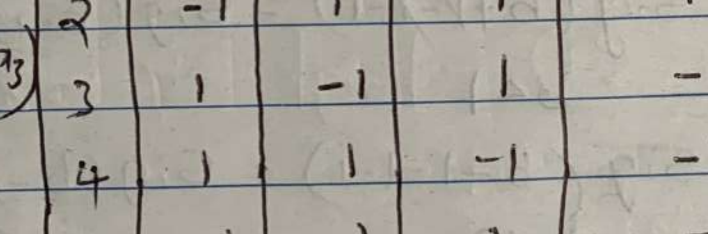
$f(x) = \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2$

$\bar{x}_1 x_2 x_3$:- AND of $\bar{x}_1$, x_2 and x_3

~~y=sig~~ (b ~~x_1~~ x_2+x_3)

	x_1	x_2	x_3	Desired o/p
1	-1	-1	-1	-1
2	-1	1	1	1
3	1	-1	1	-1
4	1	1	-1	-1
5	-1	-1	1	-1
6	1	-1	-1	-1
7	-1	1	-1	-1
8	1	1	1	-1

① ⇒ ~~y = -1 = sig (b~~



$y = Sig(b - x_1 + x_2 + x_3)$

$$y = \mathrm{sig}(b - x_1 + x_2 + x_3)$$

① $\Rightarrow$ $y = -1 = \mathrm{sig}(b + 1 - 1 - 1) = \mathrm{sig}(b - 1)$

$$b - 1 < 0$$

$$b < 1$$

② $\Rightarrow$ $y = 1 = \mathrm{sig}(b + 1 + 1 + 1) = \mathrm{sig}(b + 3)$

$$b + 3 > 0$$

$$b > -3 \quad — \text{Ⓐ}$$

③ $\Rightarrow$ $y = -1 = \mathrm{sig}(b - 1 - 1 + 1) = \mathrm{sig}(b - 1)$

$$b - 1 < 0$$

$$b < 1$$

④ $\Rightarrow$ $y = -1 = \mathrm{sig}(b - \cancel{1} + \cancel{1} - 1) = \mathrm{sig}(b - 1)$

$$b < 1$$

⑤ $\Rightarrow$ $y = -1 = \mathrm{sig}(b + \cancel{1} - \cancel{1} - 1) = \mathrm{sig}(b - 1)$

$$b < 1$$

⑥ $\Rightarrow$ $y = -1 = \mathrm{sig}(b - 1 - 1 - 1) = \mathrm{sig}(b - 3)$

$$b < 3$$

⑦ $\Rightarrow$ $y = -1 = \mathrm{sig}(b + 1 + \cancel{1} - \cancel{1}) = \mathrm{sig}(b + 1)$

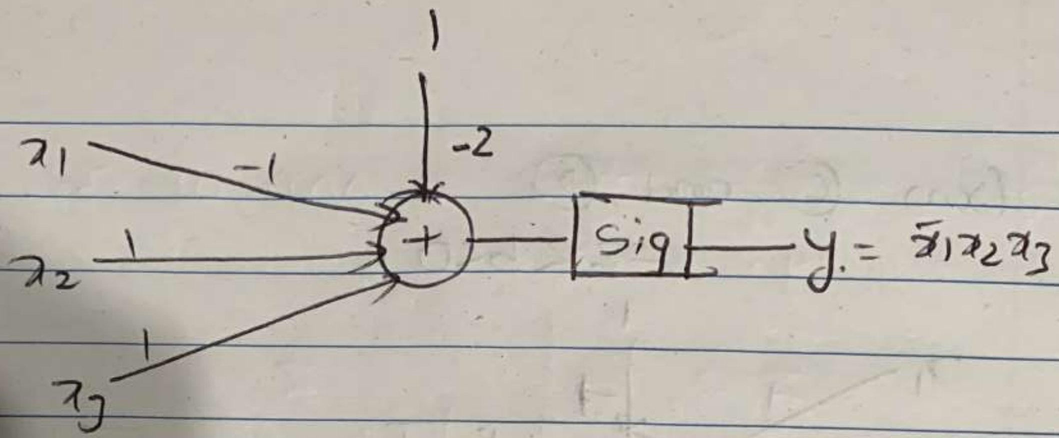
$$b + 1 < 0$$

$$b < -1 \quad — \text{Ⓑ}$$

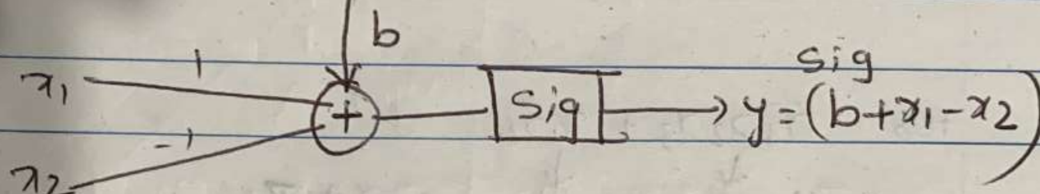
⑧ $\Rightarrow$ $y = -1 = \mathrm{sig}(b - \cancel{1} + \cancel{1} + 1) = \mathrm{sig}(b + 1)$

$$b < -1$$

from Ⓐ and Ⓑ :- $-3 < b < -1$



Now $x_1 \bar{x}_2$



S	x_1	x_2	Desired O/p ($x_1\bar{x}_2$)
1	-1	-1	-1
2	1	-1	1
3	-1	1	-1
4	1	1	-1

① $\Rightarrow y = -1 = \text{sig}(b - 1 + 1) = \text{sig}(b)$ — (C)

$b < 0$

② $\Rightarrow y = 1 = \text{sig}(b + 1 + 1) = \text{sig}(b + 2)$ — (D)

$b > -2$

③ $\Rightarrow y = -1 = \text{sig}(b - 1 - 1) = \text{sig}(b - 2)$

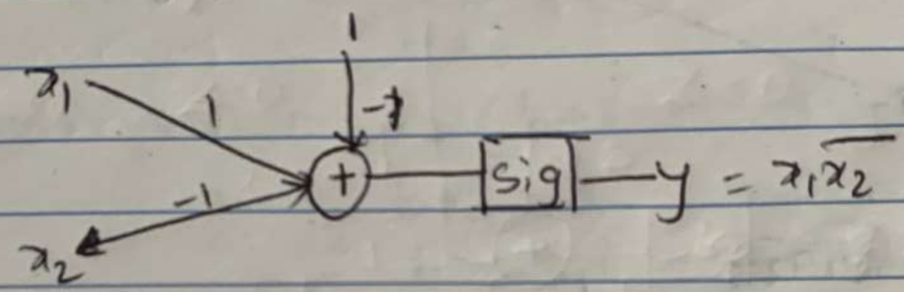
$b < 2$

④ $\Rightarrow y = -1 = \text{sig}(b + 1 - 1) = \text{sig}(b)$

$b < 0$

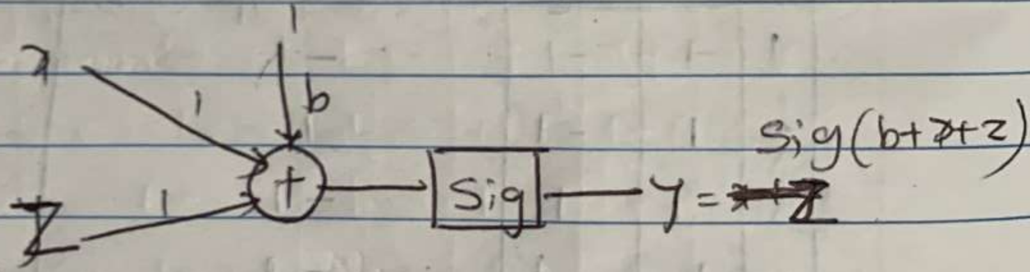
from (C) and (D)

$-2 < b < 0$



For $\bar{x}_1 x_2 x_3 + x_1 \bar{x}_2$

$$\underbrace{\bar{x}_1 x_2 x_3}_{x} + \underbrace{x_1 \bar{x}_2}_{z} = x + z$$



s	x	z	o/p
1	-1	-1	-1
2	-1	1	1
3	1	-1	1
4	1	1	1

(1) $\Rightarrow y = -1 = sig(b - 1 - 1) = sig(b - 2)$

$b < 2$ — (i)

(2) $\Rightarrow y = 1 = sig(b - 1 + 1) = sig(b)$

$b > 0$

③ ⇒ $y = 1 = sig(b+1-1) = sig(b)$

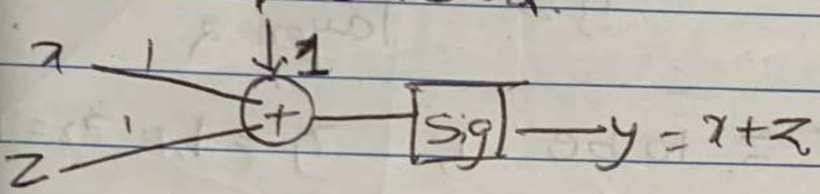
$b > 0$ — (ii)

④ ⇒ $y = 1 = sig(b+1+1) = sig(b-2)$

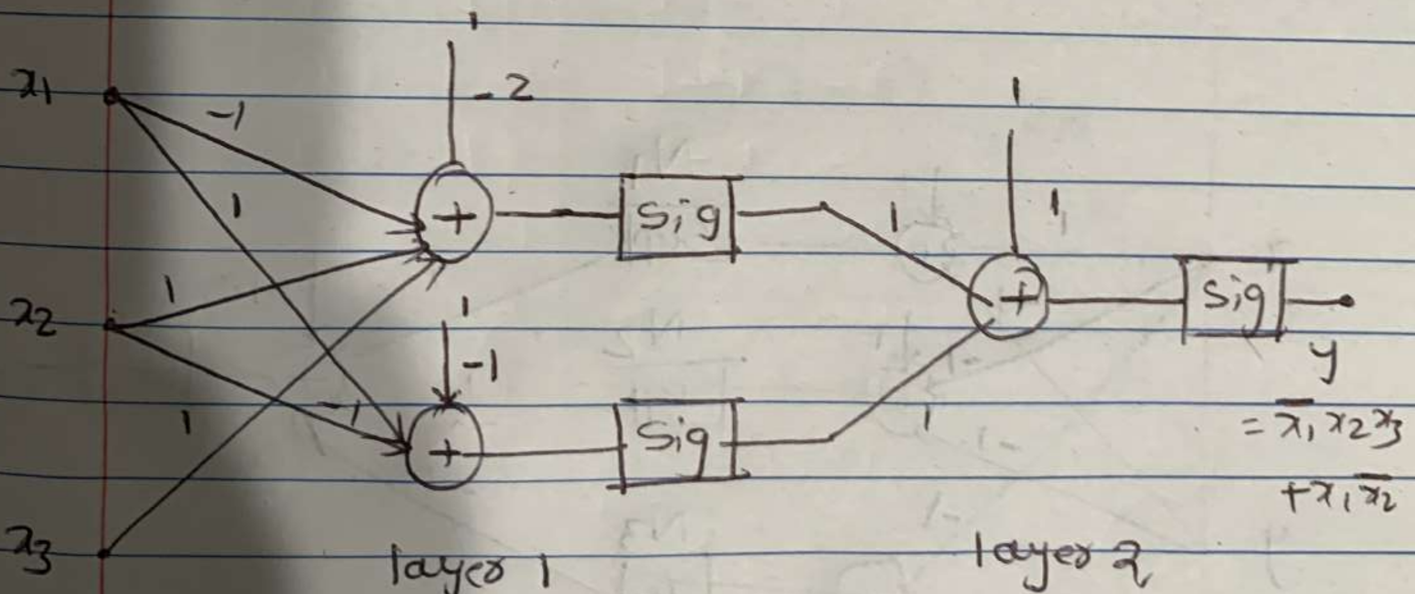
$b > 2$

from (i) and (ii)

$0 < b < 2$



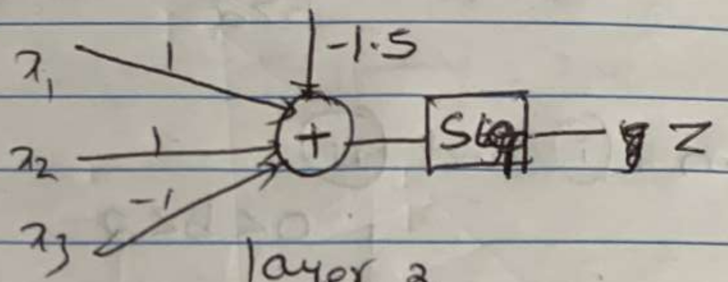
Now, final network is.



2) The second layer is AND of x_1, x_2 and $\bar{x}_3$

i.e $y = x_1 x_2 \bar{x}_3$

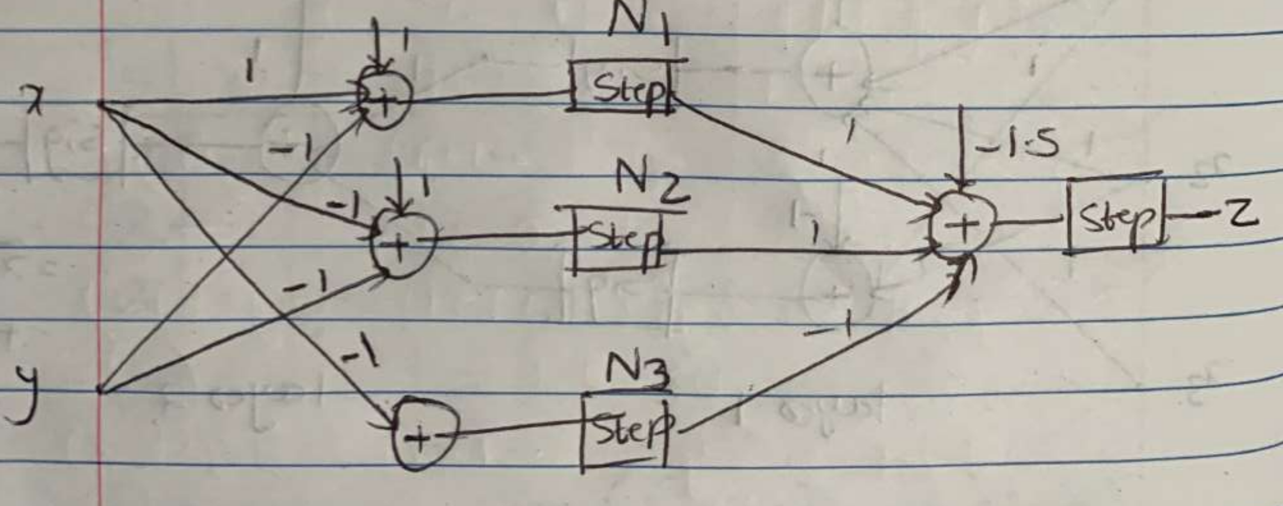
So,



layer 2

For Z to be 1, $x_1 = 1$, $x_2 = 1$ and $x_3 = 0$

<u>Given network</u>



layer 1

layer 2

In layer 1, a) output of $N_1 = 1$

i.e $y_1 = sig(1+x-y) = 1$

$1+x-y \geq 0 \Rightarrow \underline{x-y \geq -1}$ — l_A

$\cancel{y \leq x+1}$

b) output of $N_2 = 1$

i.e $y_2 = sig(1-x-y) = 1$

$1-x-y \geq 0$

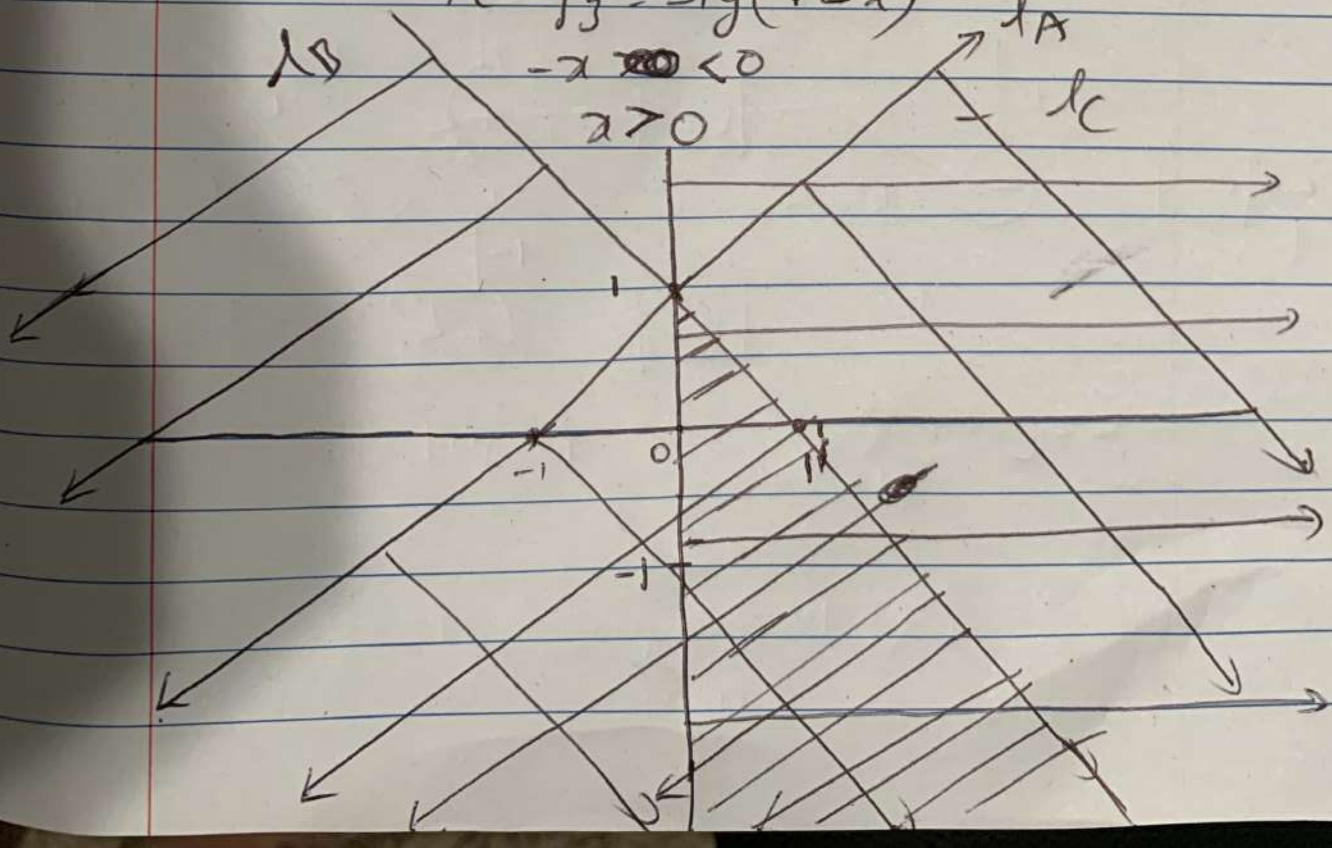
$x+y \leq 1$ — l_B

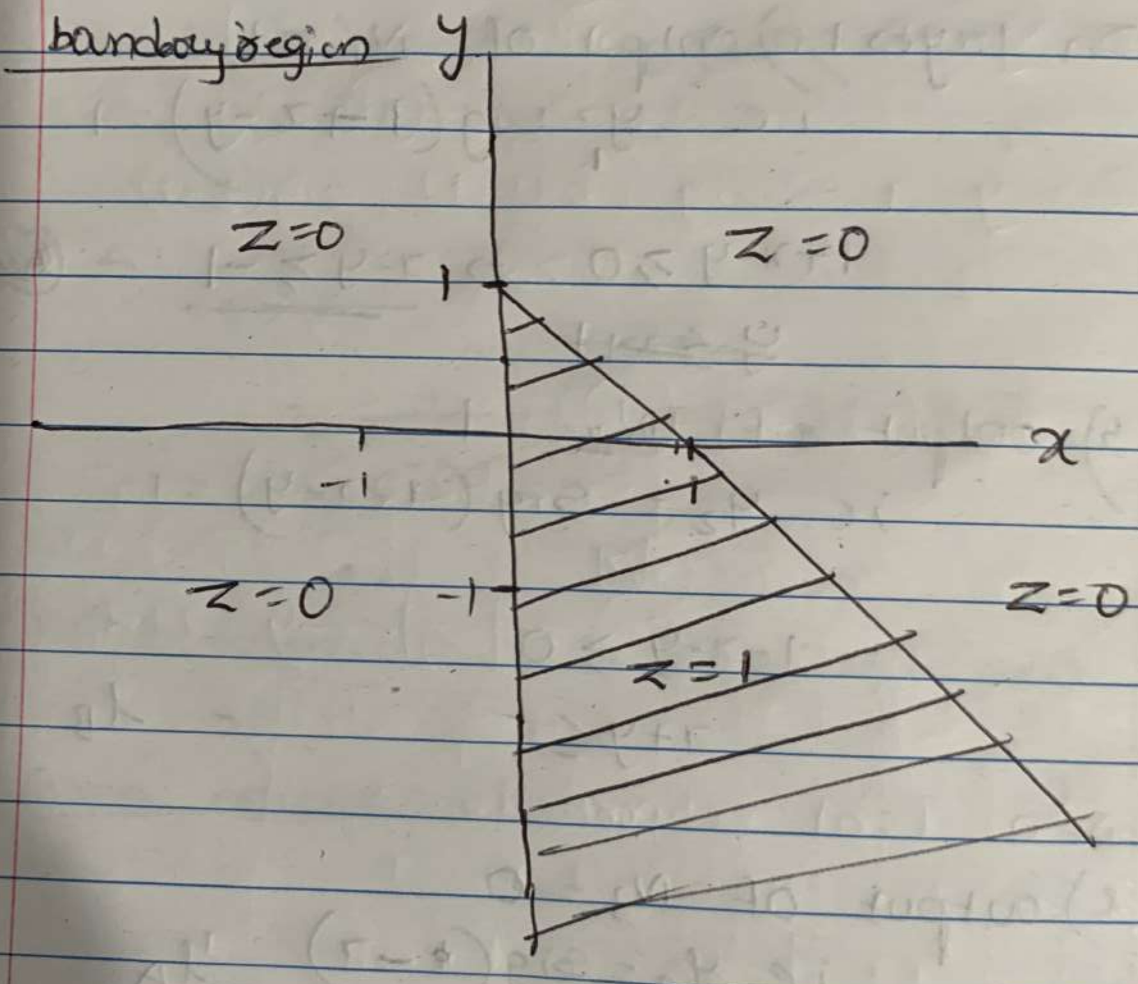
c) output of $N_3 = 0$

i.e $y_3 = sig(-x)$

$-x < 0$

$x > 0$





bandary region
y
x
Z=0
Z=0
Z=0
Z=0
Z=1
1
-1
1
-1

In [1]:
```
from random import *
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
b = []
w0 = uniform(-0.25, 0.25)
w1 = uniform(-1, 1)
w2 = uniform(-1, 1)
b.append([w0,w1,w2])
wopt = np.asarray(b)
print(wopt)
```

```
[[ 0.07398882 -0.15041019  0.28511816]]
```

In [3]:
```
# n = 100
```

In [4]:
```
a = []
for i in range(0,100):
    x1 = uniform(-1, 1)
    x2 = uniform(-1, 1)
    a.append([1, x1, x2])
    i += 1
S = np.asarray(a)
#print(S)
```

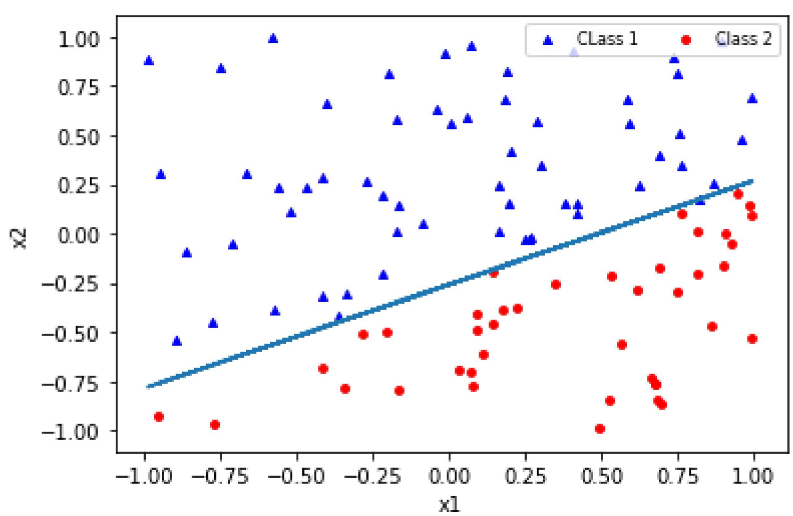
In [5]:
```
wTopt = np.transpose(wopt)
#print(wopt[0][1])
```

In [6]:
```
S1 = []
S2 = []
for i in range(0,100):
    if np.matmul(S[i], wTopt) >= 0:
        S1.append(S[i])
    else:
        S2.append(S[i])
```

In [7]:
```
print(len(S1))
print(len(S2))
```

```
59
41
```

In [8]:
```
fig, ax = plt.subplots()
xs = [x[1] for x in S1]
ys = [y[2] for y in S1]

# produce a legend with the unique colors from the scatter
scatter1 = plt.scatter(xs, ys, color ='blue', marker  = '^', s = 16)




xs = [x[1] for x in S2]
ys = [y[2] for y in S2]
scatter2 = plt.scatter(xs, ys, color ='red', s = 15)

plt.legend((scatter1, scatter2),
           ('CLass 1', 'Class 2'),
           scatterpoints=1,
           loc='upper right',
           ncol=3,
           fontsize=8)

xs = [x[1] for x in S]
ys = [y[2] for y in S]
#print(len(xs))
y = []
for i in range(len(S)):
    y.append(-((wopt[0][0]+(wopt[0][1]*xs[i]))/wopt[0][2]))




plt.plot(xs, y)

plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



In [9]:
```
# PTA
```

In [10]:
```
b = []
wa = uniform(-1, 1)
wb = uniform(-1, 1)
wc = uniform(-1, 1)
b. append([wa,wb,wc])
w = np.asarray(b)
t = w
print(w)
```

```
[[ 0.07611122  0.26930747 -0.57670416]]
```

In [11]:
```
# Epoch 0
m = []
e = []
wT = np.transpose(w)
count = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
    i += 1
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
    i += 1
```

In [12]:
```
# Learing Rate 1
n = 1

# Epoch 1
count = 0
epoch = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
        w =  w + n*S1[i]

    i += 1
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
        w =  w - n*S2[i]
    i += 1
wT= np.transpose(w)
e.append(epoch)
m.append(count)
m
```

Out[12]:
```
[77]
```

In [13]:
```python
while count != 0:
    epoch += 1
    count = 0
    for i in range(len(S1)):
        if np.matmul(S1[i], wT) < 0:
            w =  w + n*S1[i]
            count += 1
        i += 1
    for i in range(len(S2)):
        if np.matmul(S2[i], wT) >= 0:
            w =  w - n*S2[i]
            count += 1
        i += 1
    m.append(count)
    e.append(epoch)
    wT = np.transpose(w)
print("Final weights for learing rate 1 " + str(w))
```

```
Final weights for learing rate 1 [[ 11.07611122 -22.19536636  42.95259398]]
```

In [14]:
```python
# Testing
count = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
    i += 1
print(count)
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
    i += 1

count
```

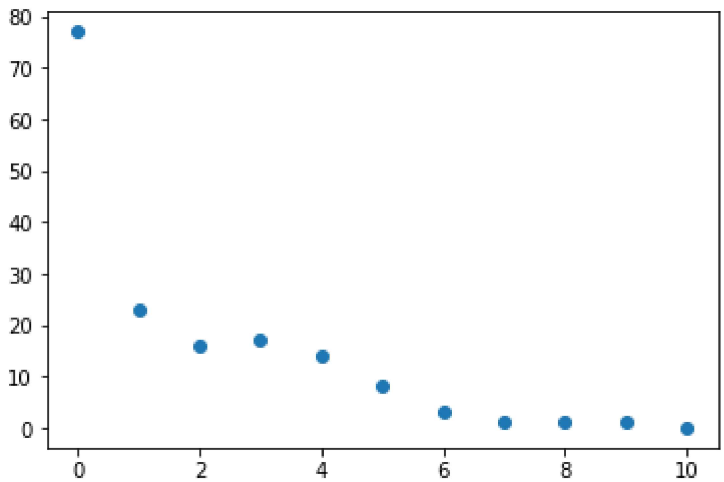
```
0
```

Out[14]: 0

In [15]:
```
print(e)
print(m)
plt.scatter(e,m)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[77, 23, 16, 17, 14, 8, 3, 1, 1, 1, 0]
```

Out[15]: <matplotlib.collections.PathCollection at 0x196a7c3d0f0>



In [16]:
```
# Learning Rate 10
n = 10
w = t
wT = np.transpose(w)
e= []
m =[]
# Epoch 1
count = 0
epoch = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
        w =  w + n*S1[i]

    i += 1
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
        w =  w - n*S2[i]
    i += 1
wT= np.transpose(w)
e.append(epoch)
m.append(count)
```

In [17]:
```
while count != 0:
    epoch += 1
    count = 0
    for i in range(len(S1)):
        if np.matmul(S1[i], wT) < 0:
            w =  w + n*S1[i]
            count += 1
        i += 1
    for i in range(len(S2)):
        if np.matmul(S2[i], wT) >= 0:
            w =  w - n*S2[i]
            count += 1
        i += 1
    m.append(count)
    e.append(epoch)
    wT = np.transpose(w)
print("Final weights for learing rate 10 " + str(w))
```

```
Final weights for learing rate 10 [[ 110.07611122 -221.60765386  432.9090093
3]]
```

In [18]:
```
# Testing
count = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
    i += 1
print(count)
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
    i += 1

count
```

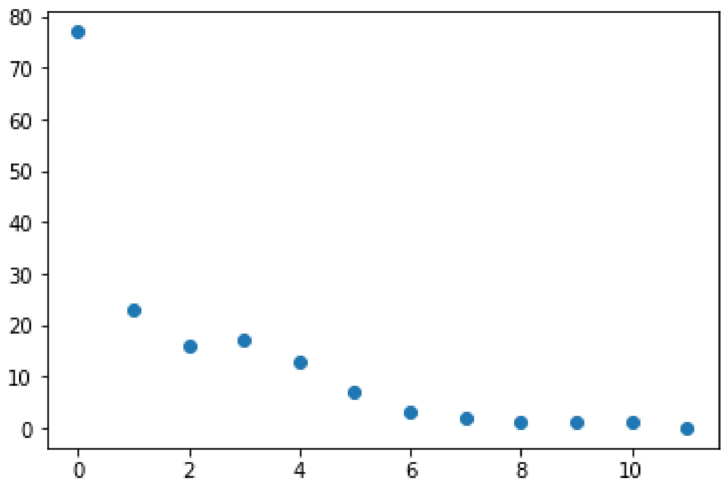
```
0
```

Out[18]:
```
0
```

In [19]:
```
print(e)
print(m)
plt.scatter(e,m)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
[77, 23, 16, 17, 13, 7, 3, 2, 1, 1, 1, 0]
```

Out[19]: <matplotlib.collections.PathCollection at 0x196a7ca9080>



In [20]:
```
# Learning Rate 0.1
n = 0.1
w = t
wT = np.transpose(w)
e= []
m =[]
# Epoch 1
count = 0
epoch = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
        w =  w + n*S1[i]

    i += 1
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
        w =  w - n*S2[i]
    i += 1
wT= np.transpose(w)
e.append(epoch)
m.append(count)
```

In [21]:
```
while count != 0:
    epoch += 1
    count = 0
    for i in range(len(S1)):
        if np.matmul(S1[i], wT) < 0:
            w =  w + n*S1[i]
            count += 1
        i += 1
    for i in range(len(S2)):
        if np.matmul(S2[i], wT) >= 0:
            w =  w - n*S2[i]
            count += 1
        i += 1
    m.append(count)
    e.append(epoch)
    wT = np.transpose(w)
print("Final weights for learing rate 0.1 " + str(w))
```

```
Final weights for learing rate 0.1 [[ 1.17611122 -2.33942677  4.49016557]]
```

In [22]:
```
# Testing
count = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
    i += 1
print(count)
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
    i += 1

count
```

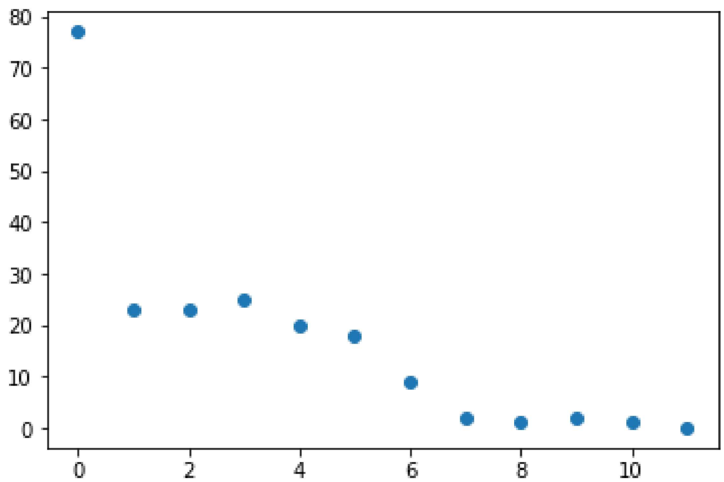
```
0
```

Out[22]:
```
0
```

In [23]:
```
print(e)
print(m)
plt.scatter(e,m)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
[77, 23, 23, 25, 20, 18, 9, 2, 1, 2, 1, 0]
```

Out[23]: <matplotlib.collections.PathCollection at 0x196a7d11e80>



In [ ]:

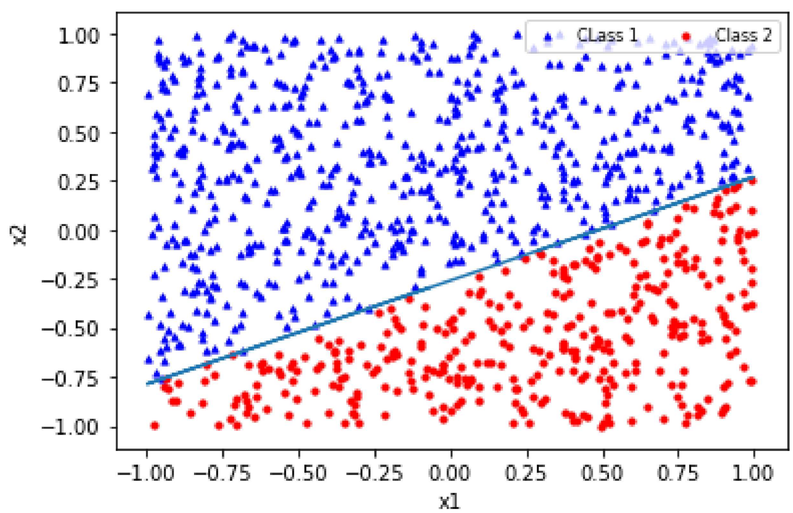
In [24]:
```
# n = 1000

a = []
for i in range(0,1000):
    x1 = uniform(-1, 1)
    x2 = uniform(-1, 1)
    a.append([1, x1, x2])
    i += 1
S = np.asarray(a)
#print(S)
```

In [25]:
```
wTopt = np.transpose(wopt)
```

In [26]:
```
S1 = []
S2 = []
for i in range(0,1000):
    if np.matmul(S[i], wTopt) >= 0:
        S1.append(S[i])
    else:
        S2.append(S[i])
print(len(S1))
print(len(S2))
```

```
638
362
```

In [27]:
```
fig, ax = plt.subplots()
xs = [x[1] for x in S1]
ys = [y[2] for y in S1]

# produce a legend with the unique colors from the scatter
scatter1 = plt.scatter(xs, ys, color ='blue', marker  = '^', s = 10)




xs = [x[1] for x in S2]
ys = [y[2] for y in S2]
scatter2 = plt.scatter(xs, ys, color ='red', s = 10)

plt.legend((scatter1, scatter2),
           ('CLass 1', 'Class 2'),
           scatterpoints=1,
           loc='upper right',
           ncol=3,
           fontsize=8)

xs = [x[1] for x in S]
ys = [y[2] for y in S]
#print(len(xs))
y = []
for i in range(len(S)):
    y.append(-((wopt[0][0]+(wopt[0][1]*xs[i]))/wopt[0][2]))




plt.plot(xs, y)

plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



In [28]:

```
# Epoch 0
m = []
e = []
w = t
wT = np.transpose(w)
count = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
    i += 1
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
    i += 1
```

In [29]:

```
# Learing Rate 1
n = 1

# Epoch 1
count = 0
epoch = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
        w =  w + n*S1[i]

    i += 1
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
        w =  w - n*S2[i]
    i += 1
wT= np.transpose(w)
e.append(epoch)
m.append(count)
```

In [30]:
```
while count != 0:
    epoch += 1
    count = 0
    for i in range(len(S1)):
        if np.matmul(S1[i], wT) < 0:
            w =  w + n*S1[i]
            count += 1
        i += 1
    for i in range(len(S2)):
        if np.matmul(S2[i], wT) >= 0:
            w =  w - n*S2[i]
            count += 1
        i += 1
    m.append(count)
    e.append(epoch)
    wT = np.transpose(w)
print("Final weights for learing rate 1 " + str(w))
```

```
Final weights for learing rate 1 [[ 105.07611122 -215.03790194  409.4783406
]]
```

In [31]:
```
# Testing
count = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
    i += 1
print(count)
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
    i += 1

count
```

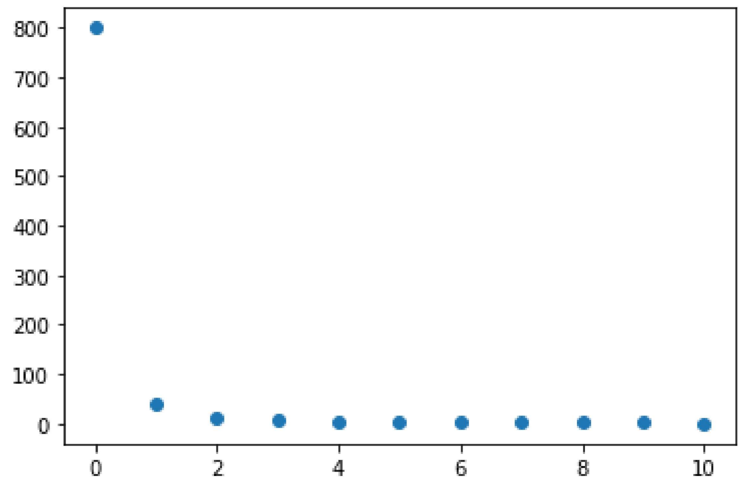
```
0
```

Out[31]:
```
0
```

In [32]:
```
print(e)
print(m)
plt.scatter(e,m)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[800, 41, 13, 7, 4, 2, 2, 2, 2, 2, 0]
```

Out[32]: <matplotlib.collections.PathCollection at 0x196a7e3f0f0>



In [33]:
```
# Learning Rate 10
n = 10
w = t
wT = np.transpose(w)
e= []
m =[]
# Epoch 1
count = 0
epoch = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
        w =  w + n*S1[i]

    i += 1
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
        w =  w - n*S2[i]
    i += 1
wT= np.transpose(w)
e.append(epoch)
m.append(count)
```

In [34]:
```
while count != 0:
    epoch += 1
    count = 0
    for i in range(len(S1)):
        if np.matmul(S1[i], wT) < 0:
            w =  w + n*S1[i]
            count += 1
        i += 1
    for i in range(len(S2)):
        if np.matmul(S2[i], wT) >= 0:
            w =  w - n*S2[i]
            count += 1
        i += 1
    m.append(count)
    e.append(epoch)
    wT = np.transpose(w)
print("Final weights for learing rate 10 " + str(w))
```

```
Final weights for learing rate 10 [[ 1060.07611122 -2160.25697083  4093.18590
864]]
```

In [35]:
```
# Testing
count = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
    i += 1
print(count)
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
    i += 1

count
```

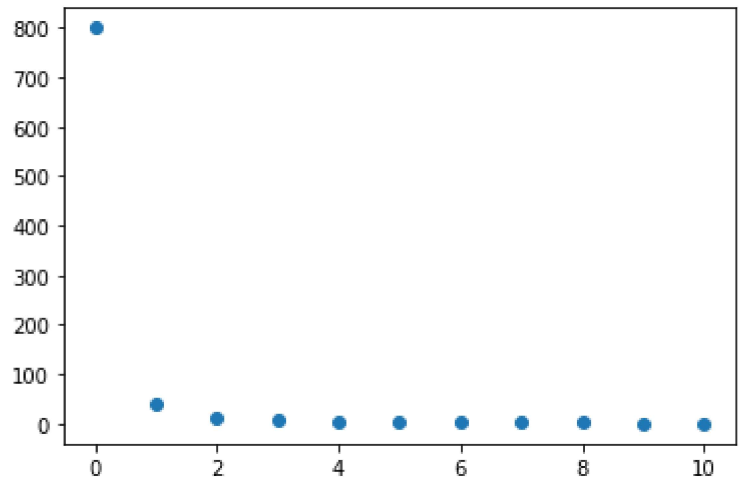
```
0
```

Out[35]:
```
0
```

In [36]:
```
print(e)
print(m)
plt.scatter(e,m)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[800, 41, 12, 8, 4, 2, 2, 2, 2, 1, 0]
```

Out[36]: <matplotlib.collections.PathCollection at 0x196a7e9eb70>



In [37]:
```
# Learning Rate 0.1
n = 0.1
w = t
wT = np.transpose(w)
e= []
m =[]
# Epoch 1
count = 0
epoch = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
        w =  w + n*S1[i]

    i += 1
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
        w =  w - n*S2[i]
    i += 1
wT= np.transpose(w)
e.append(epoch)
m.append(count)
```

In [38]:
```
while count != 0:
    epoch += 1
    count = 0
    for i in range(len(S1)):
        if np.matmul(S1[i], wT) < 0:
            w =  w + n*S1[i]
            count += 1
        i += 1
    for i in range(len(S2)):
        if np.matmul(S2[i], wT) >= 0:
            w =  w - n*S2[i]
            count += 1
        i += 1
    m.append(count)
    e.append(epoch)
    wT = np.transpose(w)
print("Final weights for learing rate 0.1 " + str(w))
```

```
Final weights for learing rate 0.1 [[ 10.47611122 -21.29689005  40.43167596]]
```

In [39]:
```
# Testing
count = 0
for i in range(len(S1)):
    if np.matmul(S1[i], wT) < 0:
        count += 1
    i += 1
print(count)
for i in range(len(S2)):
    if np.matmul(S2[i], wT) >= 0:
        count += 1
    i += 1

count
```

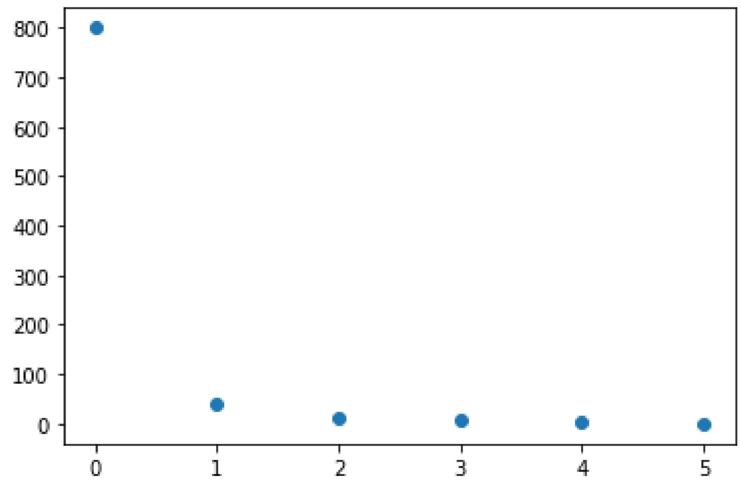
```
0
```

Out[39]:
```
0
```

In [40]:
```
print(e)
print(m)
plt.scatter(e,m)
```

```
[0, 1, 2, 3, 4, 5]
[800, 40, 13, 7, 2, 0]
```

Out[40]: <matplotlib.collections.PathCollection at 0x196a7f0a7b8>



In [ ]:

3) e) optimal weights = wopt = [w0, w1, w2] = [0.07398882 -0.15041019 0.28511816]

f) Randomly picked weights for PTA: [w0', w1', w2'] = [0.07611122 0.26930747 -0.57670416]

j) vii) Final weights for learning rate 1 = [11.07611122 -22.19536636 42.95259398]. These weights are a lot different compared to th e optimal weights above.

n) Based on my results, I found no relationship between learning rate and no of epochs needed for PTA to converge. It makes sense as the number of epochs mainly depend on the observations i.e, data

Number of samples	Learning Rate	Number of epochs for convergence
100	0.1	12
100	1	11
100	10	12
1000	0.1	6
1000	1	11
1000	10	11

o) Yes. We would get same results i.e. there wouldn't be any correlation between the learning rate and number of epochs needed for convergence. (I tried running with different weights)

p) I got higher weights in case of n = 1000. It makes sense as there are 1000 samples. It seems the ratio of the weights has correlation with learning rate for both n = 100 and n = 1000. But the number of epochs for convergence has no correlation with the number of samples.

Number of samples	Learning Rate	Final Weights	Number of epochs for convergence
100	0.1	[1.17611122 -2.33942677 4.49016557]	12
100	1	[11.07611122 -22.19536636 42.95259398]	11
100	10	[110.07611122 -221.60765386 432.90900933]	12
1000	0.1	[10.47611122 -21.29689005 40.43167596]	6
1000	1	[105.07611122 -215.03790194 409.4783406]	11
1000	10	[1060.07611122 -2160.25697083 4093.18590864]	11