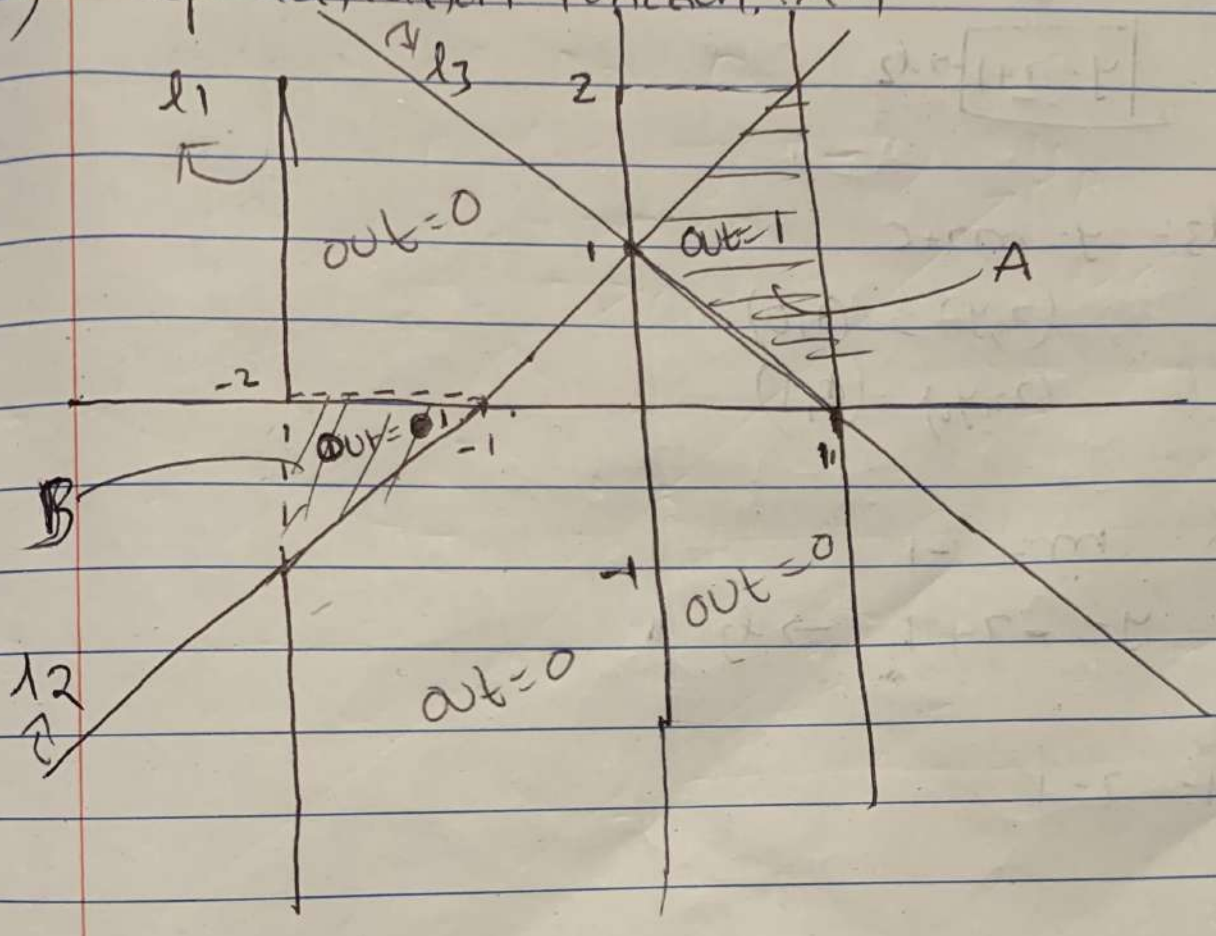
09th Feb, 2020

Homework-2

1) Step activation function, 7l4



Let's find equations of lines.

l_1: $x = -2$

l_2:

$y = mx + c$

$(x_1, y_1) = (-1, 0)$

$(x_2, y_2) = (-2, -1)$

$$m = \frac{-1-0}{-2+1} = 1$$

$y = x + c \Rightarrow y_1 = x_1 + c \Rightarrow 0 = -1 + c$

$c = 1$

$\boxed{y = x + 1} \rightarrow l_2$

l_3:- $y = mx + c$

$(x_1, y_1) = (1, 0)$

$(x_2, y_2) = (0, 1)$

$m = -1$

$y = -x + 1 \rightarrow l_3$

l_4:- $x = 1$

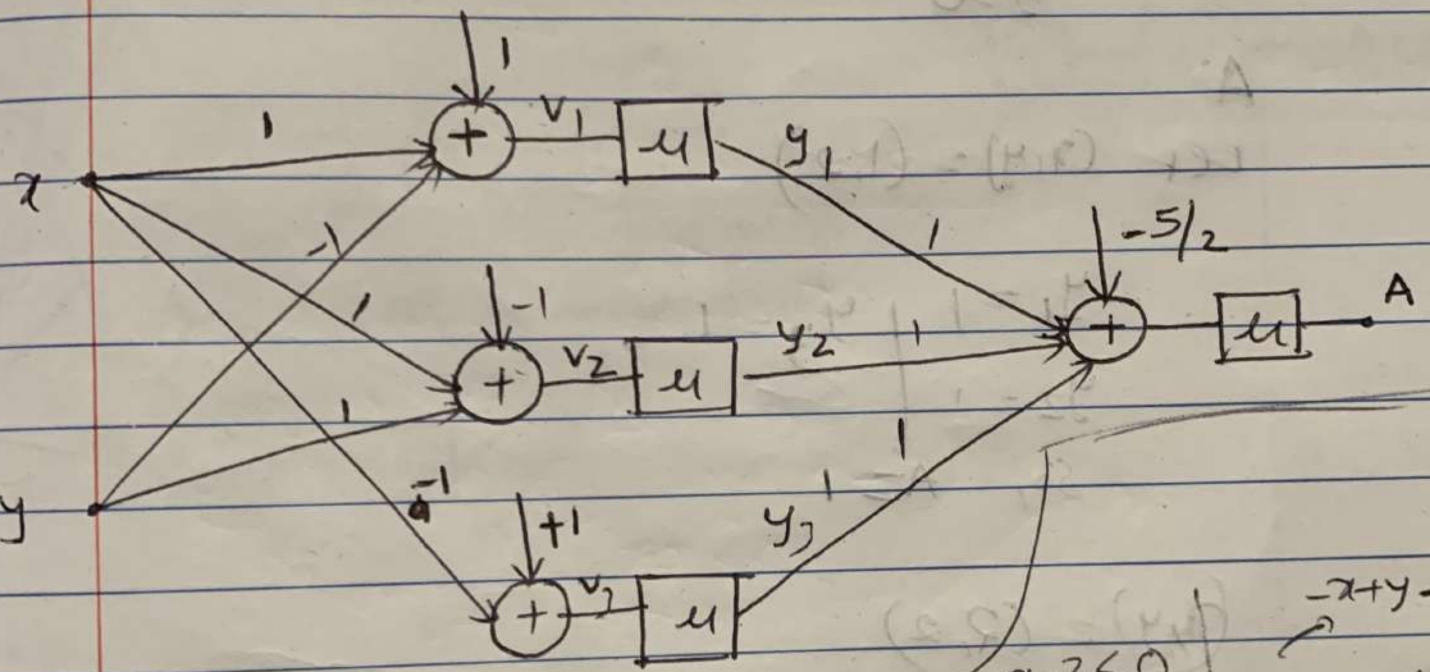
Region A:-

formed by l_2, l_3 and l_4

l_2: $y = x+1$: $x-y+1 \geq 0$

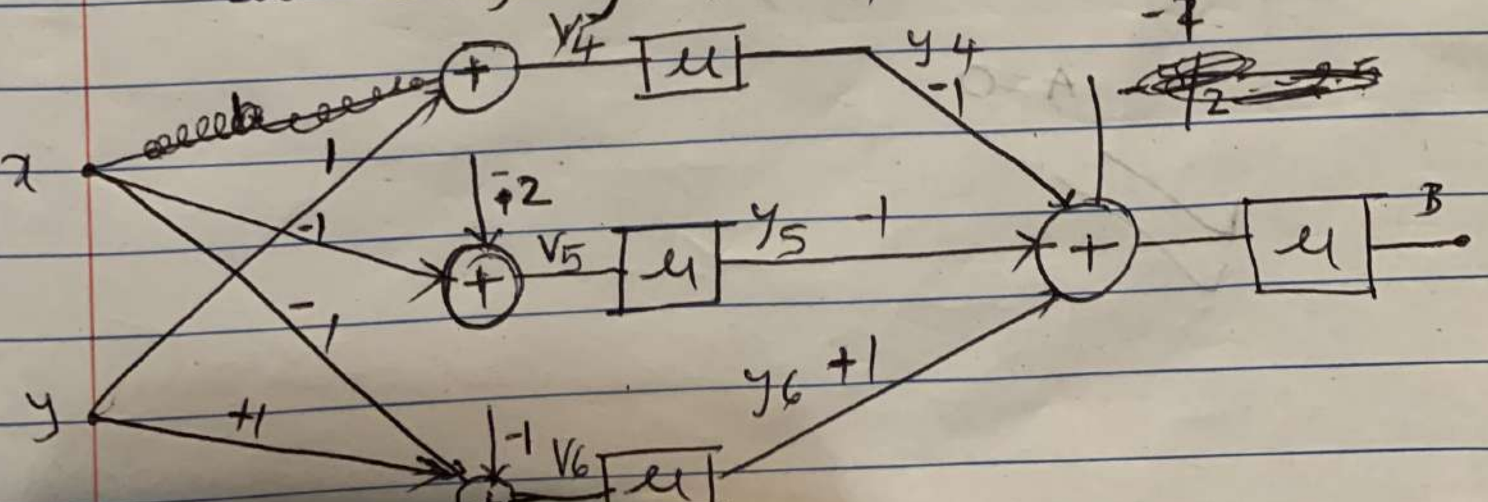
l_3: $y = -x+1$: $x+y-1 \geq 0$

l_4: $x = 1$: ~~$x \leq x+1 \leq 0$~~ $-x+1 \geq 0$



Region B:-

bounded by $y<0$, $x+2>0$, $x-2<0$, $x-y+1 \leq 0$, $-x+y-1 \geq 0$



Verification

B: Let $(x, y) = (-1, 0)$

$y_4 = 1$

$y_5 = 1$

$y_6 = 0$

$B = 0$

A:

Let $(x, y) = (1, 2)$

$y_1 = 1$ | $y_3 = 1$

$y_2 = 1$

So, $A = 1$

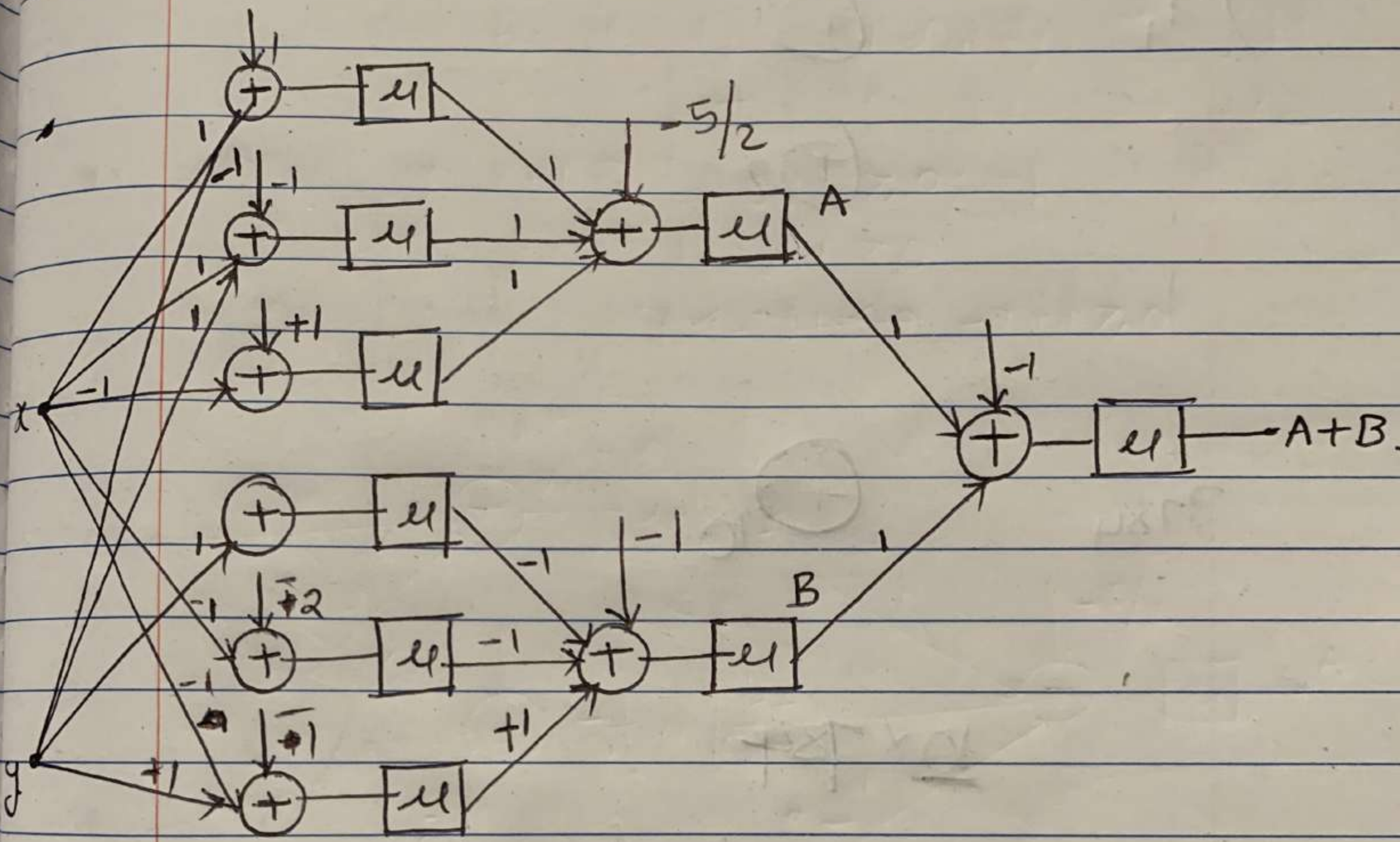
$(x, y) = (2, 2)$

$y_1 = 1$; $y_2 = 1$, $y_3 = 0$

$A = 0$.

✓

Now the combined Network is.



In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from numpy import random
```

In [2]:
```
from mnist import MNIST

mndata = MNIST(r"C:\Users\kalya\OneDrive - University of Illinois at Chicago\!UIC\!Semesters\2nd Sem\Courses
\CS 559 NN\Homeworks\HW2\Q2\data\t")

xtrain, ytrain = mndata.load_training()
xtest, ytest = mndata.load_testing()
xtrain = np.reshape(xtrain,(60000,784,1))
xtest = np.reshape(xtest,(10000,784,1))
xtrain = (xtrain)/255
xtest = (xtest)/255
xtrain.shape
```

Out[2]:
```
(60000, 784, 1)
```

In [3]:
```
w = random.normal(size = (10,784))

d = np.zeros(shape = (len(xtrain),10))
for i in range(len(ytrain)):
    d[i][ytrain[i]] = 1

dt = np.zeros(shape = (len(xtest),10))
for i in range(len(ytest)):
    dt[i][ytest[i]] = 1
```

In [4]:
```
d = d.reshape(60000, 10,1)
dt = dt.reshape(10000, 10,1)
d[0].shape
```

Out[4]:
```
(10, 1)
```

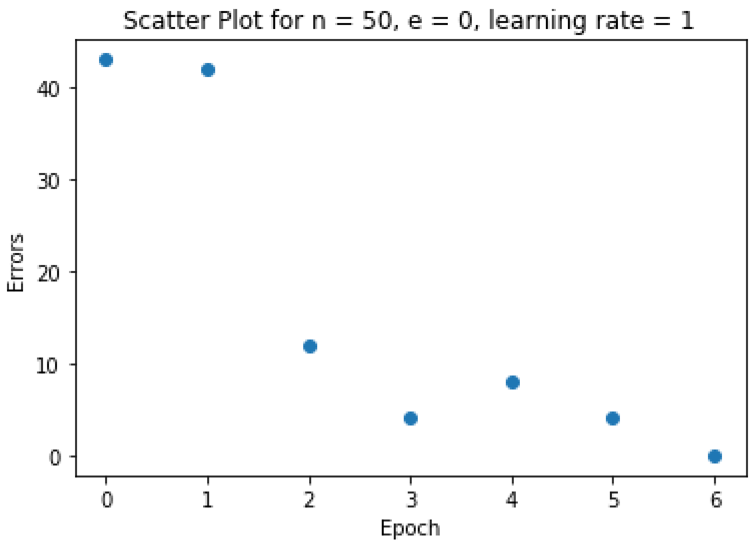
In [5]:
```
def af(x):
    max = np.argmax(x)
    y = []
    for i in range(10) :
        if(i == max):
            y.append(1)
        else :
            y.append(0)
    return y
```

In [6]:
```
def error(x, d, w, n):
    count = 0
    for i in range(n):
        p = np.matmul(w, x[i])
        y = np.array(af(p)).reshape(10,1)
        if np.any(d[i] - y):
            count += 1
    return count
```

In [7]:
```

def func(x,d,w, lr, e, n):
    epoch = 0
    errors = []
    errors.append(error(x, d, w, n))
    while errors[epoch]/n > e:
        count = 0
        for i in range(n):
            p = np.matmul(w, x[i])
            y = np.array(af(p)).reshape(10,1)
            if np.any(d[i] - y):
                count += 1
            w = w +  np.matmul(np.subtract(d[i],y),np.transpose(x[i])) * lr
        errors.append(count)
        epoch += 1
    return w, errors
```

In [8]:
```
# 2) f)
wopt, errors = func(xtrain, d, w, 1, 0, 50)
epo =[]
for i in range(len(errors)):
    epo.append(i)
plt.scatter(epo,errors)
#print(np.array_equal(wopt,w))
plt.title("Scatter Plot for n = 50, e = 0, learning rate = 1")
plt.ylabel("Errors")
plt.xlabel("Epoch")
```

Out[8]:
```
Text(0.5, 0, 'Epoch')
```

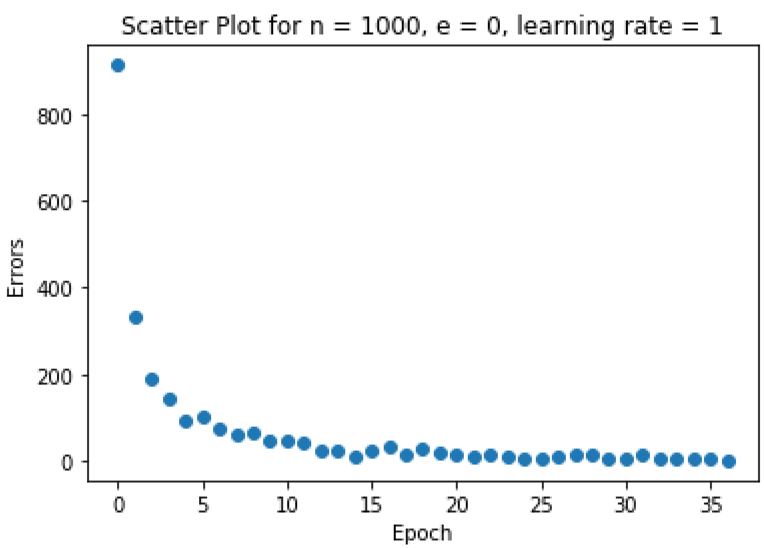


In [9]:
```
count = error(xtest, dt, wopt, 10000)
print(count)
print("Percentage of testing error for training with n = 50, e = 0, learning rate = 1 is " + str((count/10000
)*100))
```

```
4306
Percentage of testing error for training with n = 50, e = 0, learning rate = 1 is 43.059999999999995
```

In [10]:
```python
# 2) g)
wopt, errors = func(xtrain, d, w, 1, 0, 1000)
epo =[]
for i in range(len(errors)):
    epo.append(i)
plt.scatter(epo,errors)
plt.title("Scatter Plot for n = 1000, e = 0, learning rate = 1")
plt.ylabel("Errors")
plt.xlabel("Epoch")
```

Out[10]: Text(0.5, 0, 'Epoch')



In [11]:
```python
count = error(xtest, dt, wopt, 10000)
print(count)
print("Percentage of testing error for training with n = 1000, e = 0, learning rate = 1 is " + str((count/10000)*100))
```

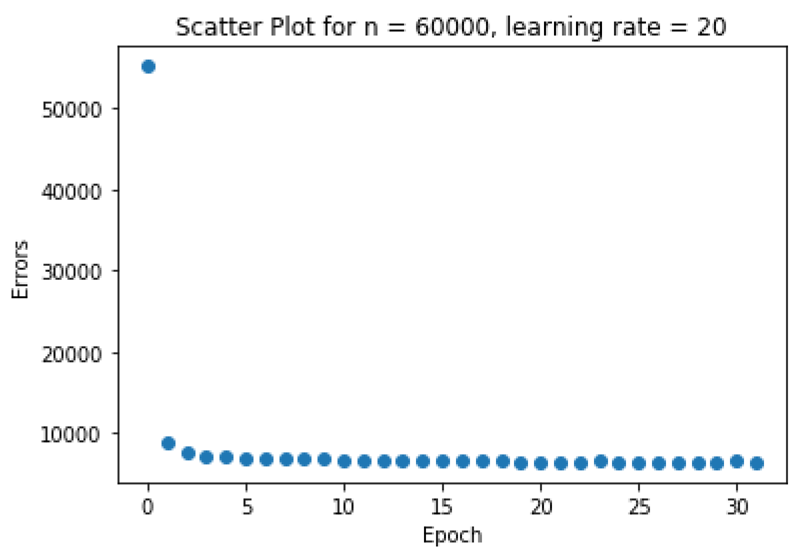
```
1722
Percentage of testing error for training with n = 1000, e = 0, learning rate = 1 is 17.22
```

In [12]:
```
# 2) h)

def func60000(x,d,w, lr, n):
    epoch = 0
    errors = []
    errors.append(error(x, d, w, n))
    while epoch < 31:
        count = 0
        for i in range(n):
            p = np.matmul(w, x[i])
            y = np.array(af(p)).reshape(10,1)
            if np.any(d[i] - y):
                count += 1
            w = w +  np.matmul(np.subtract(d[i],y),np.transpose(x[i])) * lr

        errors.append(count)
        epoch += 1
    print(count)
    return w, errors
```

In [13]:
```
wopt, errors = func60000(xtrain, d, w, 20, 60000)
epo =[]
for i in range(len(errors)):
    epo.append(i)
plt.scatter(epo,errors)
plt.title("Scatter Plot for n = 60000, learning rate = 20")
plt.ylabel("Errors")
plt.xlabel("Epoch")
```

```
6409
```

Out[13]:
```
Text(0.5, 0, 'Epoch')
```

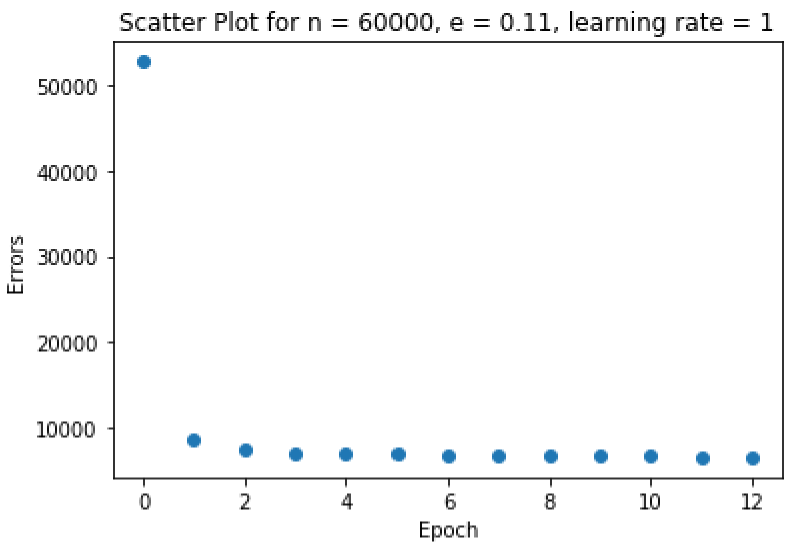


In [14]:
```
count = error(xtest, dt, wopt, 10000)
print(count)
print("Percentage of testing error for training with n = 60000, learning rate = 20 is " + str((count/10000)*1
00))
```

```
1178
Percentage of testing error for training with n = 60000, learning rate = 20 is 11.78
```

In [21]:
```
# 2) i)
w = random.normal(size = (10,784))

wopt, errors = func(xtrain, d, w, 1, 0.11, 60000)
epo =[]
for i in range(len(errors)):
    epo.append(i)
plt.scatter(epo,errors)
plt.title("Scatter Plot for n = 60000, e = 0.11, learning rate = 1")
plt.ylabel("Errors")
plt.xlabel("Epoch")
```

Out[21]:
```
Text(0.5, 0, 'Epoch')
```

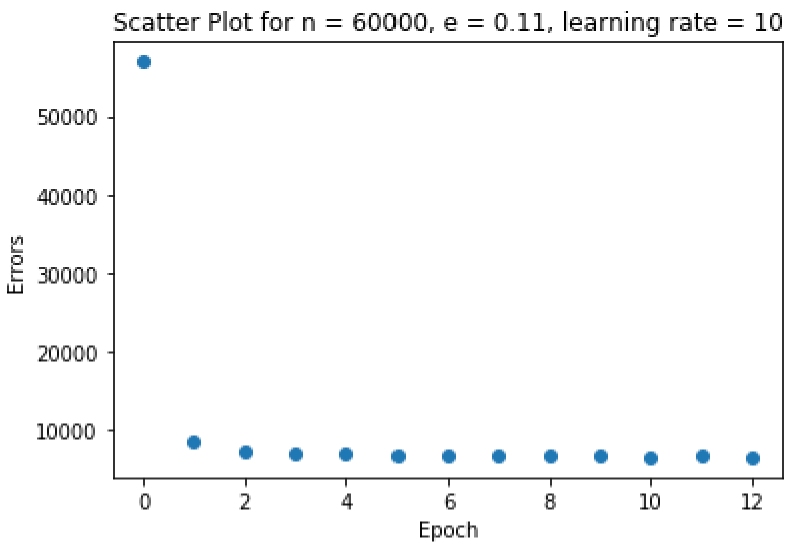


In [22]:
```
count = error(xtest, dt, wopt, 10000)
print(count)
print("Percentage of testing error for training with n = 60000, e = 0.11, learning rate = 1 is " + str((count
/10000)*100))
```

```
1313
Percentage of testing error for training with n = 60000, e = 0.11, learning rate = 1 is 13.13
```

In [23]:
```
w = random.normal(size = (10,784))

wopt, errors = func(xtrain, d, w, 10, 0.11, 60000)
epo =[]
for i in range(len(errors)):
    epo.append(i)
plt.scatter(epo,errors)
plt.title("Scatter Plot for n = 60000, e = 0.11, learning rate = 10")
plt.ylabel("Errors")
plt.xlabel("Epoch")
```

Out[23]: Text(0.5, 0, 'Epoch')

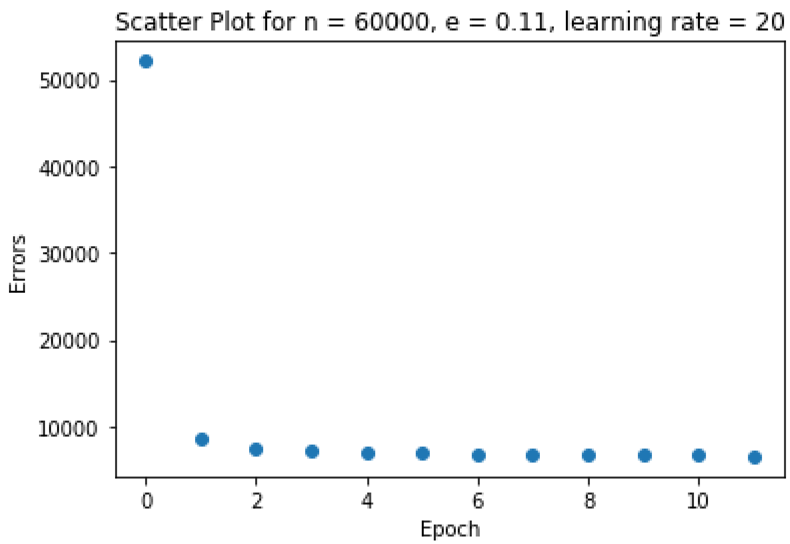


In [24]:
```
count = error(xtest, dt, wopt, 10000)
print(count)
print("Percentage of testing error for training with n = 60000, e = 0.11, learning rate = 10 is " + str((coun
t/10000)*100))
```

```
1385
Percentage of testing error for training with n = 60000, e = 0.11, learning rate = 10 is 13.850000000000001
```

In [25]:
```
w = random.normal(size = (10,784))

wopt, errors = func(xtrain, d, w, 20, 0.11, 60000)
epo =[]
for i in range(len(errors)):
    epo.append(i)
plt.scatter(epo,errors)
plt.title("Scatter Plot for n = 60000, e = 0.11, learning rate = 20")
plt.ylabel("Errors")
plt.xlabel("Epoch")
```

Out[25]: Text(0.5, 0, 'Epoch')

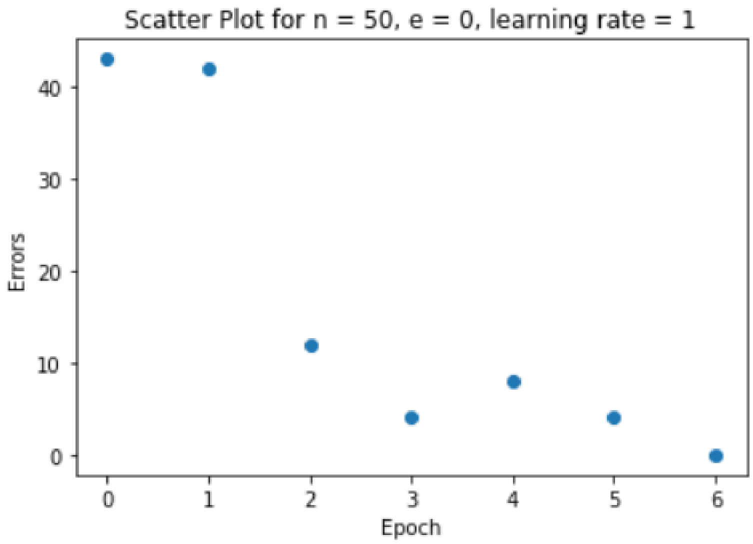


In [26]:
```
count = error(xtest, dt, wopt, 10000)
print(count)
print("Percentage of testing error for training with n = 60000, e = 0.11, learning rate = 20 is " + str((coun
t/10000)*100))
```

```
1320
Percentage of testing error for training with n = 60000, e = 0.11, learning rate = 20 is 13.200000000000001
```

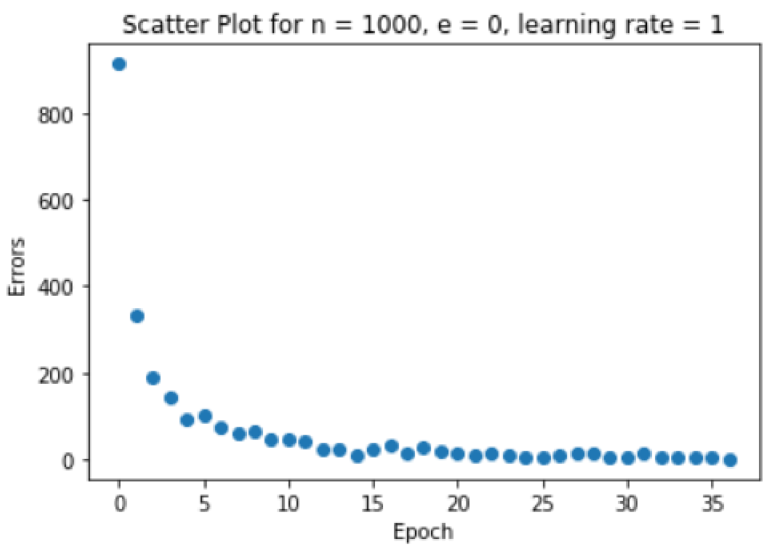
In [ ]:

2)f)



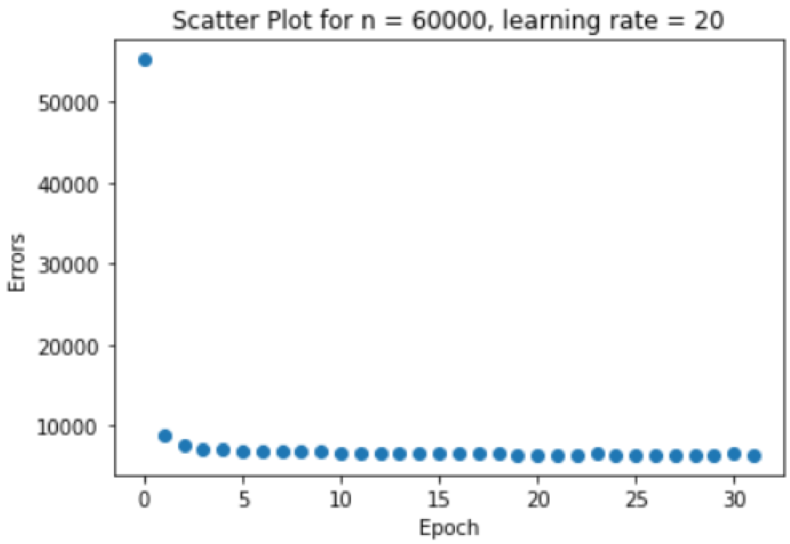
The percentage of testing error for training with n = 50, e = 0, learning rate = 1 is 43.06. The significant discrepancy between the percentage error is due to a smaller number of samples (50) in training.

g)



Since we used a greater number of samples for training this time, we got less percentage testing error of 17.22. It makes sense as we get more optimal weights with a greater number of training samples.

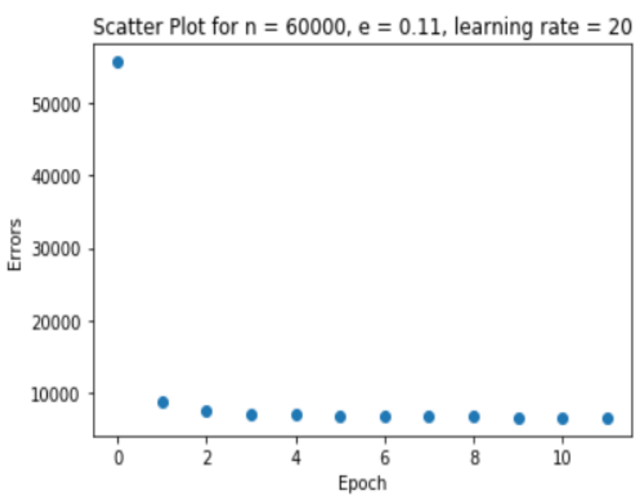
h)

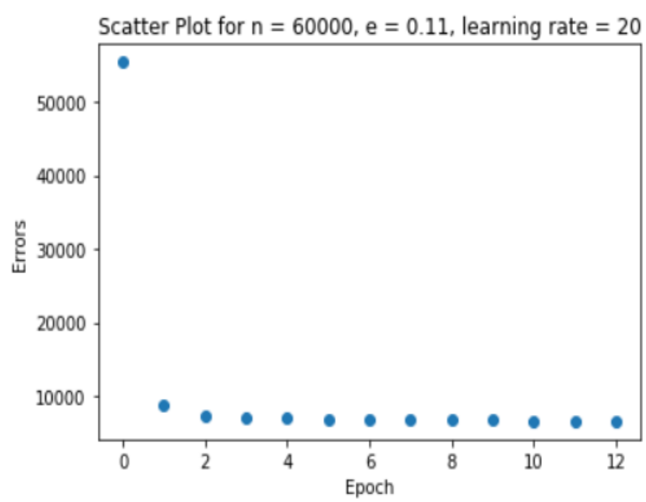


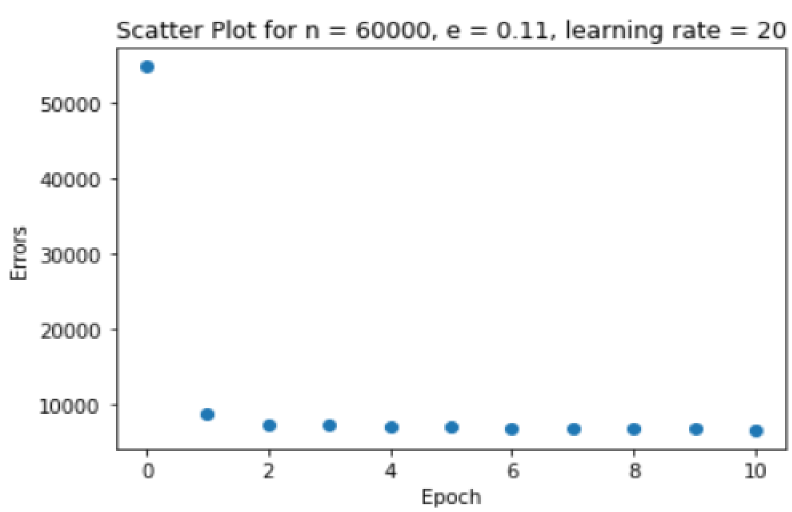
The PTA didn't converge with n = 60000, e = 0, learning rate = 20. So, I modified the algorithm to stop after 30 epochs. The number of misclassifications reduced drastically in second epoch, but they didn't reduce much from second epoch after each epoch.

In this case, I got lesser percentage testing error of 11.78 because we used 60000 samples for training.

I)







Based on my results, for error threshold of 0.1, the PTA didn't converge (or might have converged after a lot of time) with learning rates: {1,10,20}. And with a greater learning rate, the PTA converged quickly. But the percentage of testing error has no correlation with the learning rate.