Lab 2

```
import numpy as np
# for AND gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,0,0,1])
epochs = 4
w,b = np.zeros(2),0

for _ in range(epochs):
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
```

Tasks to be performed

• Change the epochs and observe the output. ✓

• Change the random weights and bias and observe the convergence of the network. ✓

• Add logic to stop the training when the error is ‘zero’ for all the inputs. ✓

• Repeat the same process for OR gate

• Try the process for XOR gate and show that the network will not stabilize (Converge).

```
# Change epochs value 4 -> 6
import numpy as np
# for AND gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,0,0,1])
epochs = 6
w,b = np.zeros(2),0
main_epo = 0   # <-- fixing epochs value
```

```python
for _ in range(epochs):

  for x,y in zip(x_list,y_list):
    main_epo += 1
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"<- at epoch",main_epo)
    w += E*x
    b += E
    print(w,b)
    print()
```

```
error :  -1 <- at epoch 1
[0. 0.] -1

error :  0 <- at epoch 2
[0. 0.] -1

error :  0 <- at epoch 3
[0. 0.] -1

error :  1 <- at epoch 4
[1. 1.] 0

error :  -1 <- at epoch 5
[1. 1.] -1

error :  -1 <- at epoch 6
[1. 0.] -2

error :  0 <- at epoch 7
[1. 0.] -2

error :  1 <- at epoch 8
[2. 1.] -1

error :  0 <- at epoch 9
[2. 1.] -1

error :  -1 <- at epoch 10
[2. 0.] -2

error :  -1 <- at epoch 11
[1. 0.] -3

error :  1 <- at epoch 12
[2. 1.] -2

error :  0 <- at epoch 13
```

```
[2. 1.] -2

error :  0 <- at epoch 14
[2. 1.] -2

error :  -1 <- at epoch 15
[1. 1.] -3

error :  1 <- at epoch 16
[2. 2.] -2

error :  0 <- at epoch 17
[2. 2.] -2

error :  -1 <- at epoch 18
[2. 1.] -3

error :  0 <- at epoch 19
[2. 1.] -3

error :  0 <- at epoch 20
[2. 1.] -3

error :  0 <- at epoch 21
[2. 1.] -3

error :  0 <- at epoch 22
[2. 1.] -3

error :  0 <- at epoch 23
[2. 1.] -3

error :  0 <- at epoch 24
[2. 1.] -3
```

Q1. Change the epochs and observe the output

> Before changing the epochs our network don't convergence (at 4 epochs) when we change epochs to 6 then then network convergenced

Q2 Change the random weights and bias

```
import numpy as np

x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,0,0,1])
epochs = 10
w,b = np.random.randint(1, 3, size=2),np.random.randint(1,3, size=1)
main_epo = 0
for _ in range(epochs):
```

```
  for x,y in zip(x_list,y_list):
    main_epo += 1
    z = np.dot(x,w) + b

    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"<- at epoch",main_epo)
    w += E*x
    b += E
    print(w,b)
    print()
```

```
error :  -1 <- at epoch 1
[1 1] [1]

error :  -1 <- at epoch 2
[1 0] [0]

error :  -1 <- at epoch 3
[0 0] [-1]

error :  1 <- at epoch 4
[1 1] [0]

error :  -1 <- at epoch 5
[1 1] [-1]

error :  -1 <- at epoch 6
[1 0] [-2]

error :  0 <- at epoch 7
[1 0] [-2]

error :  1 <- at epoch 8
[2 1] [-1]

error :  0 <- at epoch 9
[2 1] [-1]

error :  -1 <- at epoch 10
[2 0] [-2]

error :  -1 <- at epoch 11
[1 0] [-3]

error :  1 <- at epoch 12
[2 1] [-2]

error :  0 <- at epoch 13
[2 1] [-2]
```

```
error :  0 <- at epoch 14
[2 1] [-2]

error :  -1 <- at epoch 15
[1 1] [-3]

error :  1 <- at epoch 16
[2 2] [-2]

error :  0 <- at epoch 17
[2 2] [-2]

error :  -1 <- at epoch 18
[2 1] [-3]

error :  0 <- at epoch 19
[2 1] [-3]

error :  0 <- at epoch 20
[2 1] [-3]

error :  0 <- at epoch 21
[2 1] [-3]

error :  0 <- at epoch 22
[2 1] [-3]

error :  0 <- at epoch 23
[2 1] [-3]

error :  0 <- at epoch 24
[2 1] [-3]

error :  0 <- at epoch 25
[2 1] [-3]

error :  0 <- at epoch 26
[2 1] [-3]

error :  0 <- at epoch 27
[2 1] [-3]

error :  0 <- at epoch 28
[2 1] [-3]

error :  0 <- at epoch 29
[2 1] [-3]

error :  0 <- at epoch 30
[2 1] [-3]
```

```
error :  0 <- at epoch 31
[2 1] [-3]

error :  0 <- at epoch 32
[2 1] [-3]

error :  0 <- at epoch 33
[2 1] [-3]

error :  0 <- at epoch 34
[2 1] [-3]

error :  0 <- at epoch 35
[2 1] [-3]

error :  0 <- at epoch 36
[2 1] [-3]

error :  0 <- at epoch 37
[2 1] [-3]

error :  0 <- at epoch 38
[2 1] [-3]

error :  0 <- at epoch 39
[2 1] [-3]

error :  0 <- at epoch 40
[2 1] [-3]
```

After changing (W and B) by random values network conconvergenced early

Q3. Add logic to stop the training when the error is ‘zero’ for all the inputs.(Early stoping)

```python
import numpy as np
# for AND gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,0,0,1])
epochs = 10 # means we run 40 epochs
w,b = np.zeros(2),0
quit = 0
mainepo = 0
for _ in range(epochs):
  if quit== 4:
      print("Weight and Bias are stable now.") # stop at 22 epoch
      break
  for x,y in zip(x_list,y_list):
```

```
    z = np.dot(x,w) + b
    # condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    mainepo += 1
    print("error : ",E,"<- at epoch",mainepo)
    w += E*x
    b += E
    quit = (quit + 1) if E==0 else 0
    if quit== 4:
      break
    print(w,b)
```

```
error :  -1 <- at epoch 1
[0. 0.] -1
error :  0 <- at epoch 2
[0. 0.] -1
error :  0 <- at epoch 3
[0. 0.] -1
error :  1 <- at epoch 4
[1. 1.] 0
error :  -1 <- at epoch 5
[1. 1.] -1
error :  -1 <- at epoch 6
[1. 0.] -2
error :  0 <- at epoch 7
[1. 0.] -2
error :  1 <- at epoch 8
[2. 1.] -1
error :  0 <- at epoch 9
[2. 1.] -1
error :  -1 <- at epoch 10
[2. 0.] -2
error :  -1 <- at epoch 11
[1. 0.] -3
error :  1 <- at epoch 12
[2. 1.] -2
error :  0 <- at epoch 13
[2. 1.] -2
error :  0 <- at epoch 14
[2. 1.] -2
error :  -1 <- at epoch 15
[1. 1.] -3
error :  1 <- at epoch 16
[2. 2.] -2
error :  0 <- at epoch 17
[2. 2.] -2
error :  -1 <- at epoch 18
[2. 1.] -3
error :  0 <- at epoch 19
```

```
[2. 1.] -3
error :  0 <- at epoch 20
[2. 1.] -3
error :  0 <- at epoch 21
[2. 1.] -3
error :  0 <- at epoch 22
Weight and Bias are stable now.
```

Q4 apply same network for OR gate

```
import numpy as np
# for OR gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,1,1,1]) # <- just need to change actual(OG)
output
epochs = 4
w,b = np.zeros(2),0

for _ in range(epochs):
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
```

```
error :  -1 <- at epoch 0
[0. 0.] -1
error :  1 <- at epoch 0
[0. 1.] 0
error :  0 <- at epoch 0
[0. 1.] 0
error :  0 <- at epoch 0
[0. 1.] 0
error :  -1 <- at epoch 1
[0. 1.] -1
error :  0 <- at epoch 1
[0. 1.] -1
error :  1 <- at epoch 1
[1. 1.] 0
error :  0 <- at epoch 1
[1. 1.] 0
error :  -1 <- at epoch 2
[1. 1.] -1
error :  0 <- at epoch 2
[1. 1.] -1
```

```
error :  0 <- at epoch 2
[1. 1.] -1
error :  0 <- at epoch 2
[1. 1.] -1
error :  0 <- at epoch 3
[1. 1.] -1
error :  0 <- at epoch 3
[1. 1.] -1
error :  0 <- at epoch 3
[1. 1.] -1
error :  0 <- at epoch 3
[1. 1.] -1
```

Q5. Apply for XOR gate

Xor gate cann't converged by single perceptron because XOR gate is non-linear separable

```
import numpy as np
# for AND gate (Input/Output)
x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,1,1,0])
epochs = 4
w,b = np.zeros(2),0

for _ in range(epochs):
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    ### condition area for y_pred
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"<- at epoch",_)
    w += E*x
    b += E
    print(w,b)
```

```
error :  -1 <- at epoch 0
[0. 0.] -1
error :  1 <- at epoch 0
[0. 1.] 0
error :  0 <- at epoch 0
[0. 1.] 0
error :  -1 <- at epoch 0
[-1.  0.] -1
error :  0 <- at epoch 1
[-1.  0.] -1
error :  1 <- at epoch 1
[-1.  1.] 0
error :  1 <- at epoch 1
```

```
[0. 1.] 1
error :  -1 <- at epoch 1
[-1.  0.] 0
error :  -1 <- at epoch 2
[-1.  0.] -1
error :  1 <- at epoch 2
[-1.  1.] 0
error :  1 <- at epoch 2
[0. 1.] 1
error :  -1 <- at epoch 2
[-1.  0.] 0
error :  -1 <- at epoch 3
[-1.  0.] -1
error :  1 <- at epoch 3
[-1.  1.] 0
error :  1 <- at epoch 3
[0. 1.] 1
error :  -1 <- at epoch 3
[-1.  0.] 0
```