Lab - 1 | 22 January 2026

↳ 6 cells hidden

Lab - 2 | 29 January 2026

Perceptron learning Algorithm for Implementing AND Logic Gate

```
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)
print("+-----------------------------+")
print("Shivam Sharma, 1/23/SET/BCS/424")
print("22 January 2026")
```

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

```
[1. 0.] -2
error :  1 <- at epoch 1
[2. 1.] -1
error :  0 <- at epoch 2
[2. 1.] -1
error :  -1 <- at epoch 2
[2. 0.] -2
error :  -1 <- at epoch 2
[1. 0.] -3
error :  1 <- at epoch 2
[2. 1.] -2
error :  0 <- at epoch 3
[2. 1.] -2
error :  0 <- at epoch 3
[2. 1.] -2
error :  -1 <- at epoch 3
[1. 1.] -3
error :  1 <- at epoch 3
[2. 2.] -2
+------------------------------+
Shivam Sharma, 1/23/SET/BCS/424
22 January 2026
```

Tasks to be performed

1. Change the epochs and observe the output
2. Change the random weights and bias and observe the convergence of the network.
3. Add logic to stop the training when the error is ‘zero’ for all the inputs.
4. Repeat the same process for OR gate
5. Try the process for XOR gate and show that the network will not stabilize (Converge).

Tasks in Order :

1) Change the epochs and observe the output

```
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
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)
print("+------------------------------+")
print("Shivam Sharma, 1/23/SET/BCS/424")
print("22 January 2026")
```

```
error :  -1 <- at epoch 2
[2. 0.] -2
error :  -1 <- at epoch 2
[1. 0.] -3
error :  1 <- at epoch 2
[2. 1.] -2
error :  0 <- at epoch 3
[2. 1.] -2
error :  0 <- at epoch 3
[2. 1.] -2
error :  -1 <- at epoch 3
[1. 1.] -3
error :  1 <- at epoch 3
[2. 2.] -2
error :  0 <- at epoch 4
[2. 2.] -2
error :  -1 <- at epoch 4
[2. 1.] -3
error :  0 <- at epoch 4
[2. 1.] -3
error :  0 <- at epoch 4
[2. 1.] -3
error :  0 <- at epoch 5
[2. 1.] -3
error :  0 <- at epoch 5
[2. 1.] -3
error :  0 <- at epoch 5
[2. 1.] -3
error :  0 <- at epoch 5
[2. 1.] -3
error :  0 <- at epoch 6
[2. 1.] -3
error :  0 <- at epoch 6
[2. 1.] -3
error :  0 <- at epoch 6
[2. 1.] -3
error :  0 <- at epoch 6
[2. 1.] -3
error :  0 <- at epoch 7
```

```
[2. 1.] -3
error :  0 <- at epoch 7
[2. 1.] -3
error :  0 <- at epoch 7
[2. 1.] -3
error :  0 <- at epoch 7
[2. 1.] -3
error :  0 <- at epoch 8
[2. 1.] -3
error :  0 <- at epoch 8
[2. 1.] -3
error :  0 <- at epoch 8
[2. 1.] -3
error :  0 <- at epoch 8
[2. 1.] -3
error :  0 <- at epoch 9
[2. 1.] -3
error :  0 <- at epoch 9
[2. 1.] -3
error :  0 <- at epoch 9
```

2) Change the random weights and bias and observe the convergence of the network.

```
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
w,b = np.array([2,5]),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)
print("+------------------------------+")
print("Shivam Sharma, 1/23/SET/BCS/424")
print("22 January 2026")
```

```
error :  -1 <- at epoch 0
[2 5] -1
error :  -1 <- at epoch 0
[2 4] -2
```

```
error :  -1 <- at epoch 0
[1 4] -3
error :  0 <- at epoch 0
[1 4] -3
error :  0 <- at epoch 1
[1 4] -3
error :  -1 <- at epoch 1
[1 3] -4
error :  0 <- at epoch 1
[1 3] -4
error :  0 <- at epoch 1
[1 3] -4
error :  0 <- at epoch 2
[1 3] -4
error :  0 <- at epoch 2
[1 3] -4
error :  0 <- at epoch 2
[1 3] -4
error :  0 <- at epoch 2
[1 3] -4
error :  0 <- at epoch 3
[1 3] -4
error :  0 <- at epoch 3
[1 3] -4
error :  0 <- at epoch 3
[1 3] -4
error :  0 <- at epoch 3
[1 3] -4
error :  0 <- at epoch 4
[1 3] -4
error :  0 <- at epoch 4
[1 3] -4
error :  0 <- at epoch 4
[1 3] -4
error :  0 <- at epoch 4
[1 3] -4
error :  0 <- at epoch 5
[1 3] -4
error :  0 <- at epoch 5
[1 3] -4
error :  0 <- at epoch 5
[1 3] -4
error :  0 <- at epoch 5
[1 3] -4
error :  0 <- at epoch 6
[1 3] -4
error :  0 <- at epoch 6
[1 3] -4
error :  0 <- at epoch 6
[1 3] -4
error :  0 <- at epoch 6
[1 3] -4
error :  0 <- at epoch 7
```

```
[1 3]  4
```

3) Add logic to stop the training when the error is 'zero' for all the inputs.

```
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 # Increased epochs to ensure convergence without fixed
w,b = np.zeros(2),0

for _ in range(epochs):
  errors_in_epoch = False # Flag to track errors in the current epc
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print("error : ",E,"is at epoch",_)
    if E != 0:
      w += E*x
      b += E
      errors_in_epoch = True
    print(w,b)
  if not errors_in_epoch:
    print(f"Training stopped at epoch {_} as error has become zero
    print("+-----------------------------+")
    print("Shivam Sharma, 1/23/SET/BCS/424")
    print("22 January 2026")
    break
```

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

```
error :  -1 is at epoch 2
[1. 0.] -3
error :  1 is at epoch 2
[2. 1.] -2
error :  0 is at epoch 3
[2. 1.] -2
error :  0 is at epoch 3
[2. 1.] -2
error :  -1 is at epoch 3
[1. 1.] -3
error :  1 is at epoch 3
[2. 2.] -2
error :  0 is at epoch 4
[2. 2.] -2
error :  -1 is at epoch 4
[2. 1.] -3
error :  0 is at epoch 4
[2. 1.] -3
error :  0 is at epoch 4
[2. 1.] -3
error :  0 is at epoch 5
[2. 1.] -3
error :  0 is at epoch 5
[2. 1.] -3
error :  0 is at epoch 5
[2. 1.] -3
error :  0 is at epoch 5
[2. 1.] -3
Training stopped at epoch 5 as error has become zero for all the fol
+------------------------------+
Shivam Sharma, 1/23/SET/BCS/424
22 January 2026
```

4) Repeat the same process for OR gate

```
import numpy as np

x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,1,1,1])
epochs = 10
w,b = np.zeros(2),0

print("--------OR GATE Perceptron Learning--------")
for _ in range(epochs):
  errors_in_epoch = False
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print(f"Error: {E}, Epoch: {_}")
    if E != 0:
      w += E*x
      b += E
      errors_in_epoch = True
    print(f"Updated weights: {w}, Updated bias: {b}")
  if not errors_in_epoch:
    print(f"Training stopped at epoch {_} as error has become zero
    break

print("+------------------------------+")
print("Shivam Sharma, 1/23/SET/BCS/424")
print("22 January 2026")
```

```
--------OR GATE Perceptron Learning--------
Error: -1, Epoch: 0
Updated weights: [0. 0.], Updated bias: -1
Error: 1, Epoch: 0
Updated weights: [0. 1.], Updated bias: 0
Error: 0, Epoch: 0
Updated weights: [0. 1.], Updated bias: 0
Error: 0, Epoch: 0
Updated weights: [0. 1.], Updated bias: 0
Error: -1, Epoch: 1
Updated weights: [0. 1.], Updated bias: -1
Error: 0, Epoch: 1
Updated weights: [0. 1.], Updated bias: -1
Error: 1, Epoch: 1
Updated weights: [1. 1.], Updated bias: 0
Error: 0, Epoch: 1
Updated weights: [1. 1.], Updated bias: 0
Error: -1, Epoch: 2
Updated weights: [1. 1.], Updated bias: -1
Error: 0, Epoch: 2
Updated weights: [1. 1.], Updated bias: -1
Error: 0, Epoch: 2
Updated weights: [1. 1.], Updated bias: -1
Error: 0, Epoch: 2
Updated weights: [1. 1.], Updated bias: -1
Error: 0, Epoch: 3
Updated weights: [1. 1.], Updated bias: -1
Error: 0, Epoch: 3
Updated weights: [1. 1.], Updated bias: -1
Error: 0, Epoch: 3
Updated weights: [1. 1.], Updated bias: -1
Error: 0, Epoch: 3
Updated weights: [1. 1.], Updated bias: -1
Training stopped at epoch 3 as error has become zero for all inputs.
+------------------------------+
Shivam Sharma, 1/23/SET/BCS/424
22 January 2026
```

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

```
import numpy as np

x_list = np.array([[0,0],[0,1],[1,0],[1,1]])
y_list = np.array([0,1,1,0])
epochs = 10
w,b = np.zeros(2),0

print("--------OR GATE Perceptron Learning--------")
for _ in range(epochs):
  errors_in_epoch = False
  for x,y in zip(x_list,y_list):
    z = np.dot(x,w) + b
    y_pred = 1 if z >= 0 else 0
    E = y - y_pred
    print(f"Error: {E}, Epoch: {_}")
    if E != 0:
      w += E*x
      b += E
      errors_in_epoch = True
    print(f"Updated weights: {w}, Updated bias: {b}")
  if not errors_in_epoch:
    print(f"Training stopped at epoch {_} as error has become zero f
    break

print("+------------------------------+")
print("Shivam Sharma, 1/23/SET/BCS/424")
print("22 January 2026")
```

```
--------OR GATE Perceptron Learning--------
Error: -1, Epoch: 0
Updated weights: [0. 0.], Updated bias: -1
Error: 1, Epoch: 0
Updated weights: [0. 1.], Updated bias: 0
Error: 0, Epoch: 0
Updated weights: [0. 1.], Updated bias: 0
Error: -1, Epoch: 0
Updated weights: [-1.  0.], Updated bias: -1
Error: 0, Epoch: 1
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 1
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 1
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 1
Updated weights: [-1.  0.], Updated bias: 0
```

```
Error: -1, Epoch: 2
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 2
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 2
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 2
Updated weights: [-1.  0.], Updated bias: 0
Error: -1, Epoch: 3
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 3
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 3
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 3
Updated weights: [-1.  0.], Updated bias: 0
Error: -1, Epoch: 4
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 4
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 4
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 4
Updated weights: [-1.  0.], Updated bias: 0
Error: -1, Epoch: 5
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 5
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 5
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 5
Updated weights: [-1.  0.], Updated bias: 0
Error: -1, Epoch: 6
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 6
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 6
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 6
Updated weights: [-1.  0.], Updated bias: 0
Error: -1, Epoch: 7
Updated weights: [-1.  0.], Updated bias: -1
```