Lab - 1 | 22 January 2026

Meta Pitts Neuron (AND & OR Gates)

```
import numpy as np
def mp_neuron(x,w,theta):
  return 1 if np.dot(x, w) >= theta else 0

print("--------AND GATE--------")

# AND Gate
w_and = np.array ([1,1])
theta_and = 2
for x in [(0,0), (0,1), (1,0), (1,1)]:
  print(x, mp_neuron(np.array(x), w_and, theta_and))

print("--------OR GATE--------")

# OR Gate
w_or = np.array ([1,1])
theta_or = 1
for x in [(0,0), (0,1), (1,0), (1,1)]:
  print(x, mp_neuron(np.array(x), w_or, theta_or))


print("+------------------------------+")
print("Chirag Miglani, 1/23/SET/BCS/414")
print("22 January 2026")
```

```
--------AND GATE--------
(0, 0) 0
(0, 1) 0
(1, 0) 0
(1, 1) 1
--------OR GATE--------
(0, 0) 0
(0, 1) 1
(1, 0) 1
(1, 1) 1
+------------------------------+
Chirag Miglani, 1/23/SET/BCS/414
22 January 2026
```

AND & OR GATE(3-inputs)

```
import numpy as np
def mp_neuron(x,w,theta):
  return 1 if np.dot(x, w) >= theta else 0

print("--------AND GATE(3-inputs)--------")

w_and = np.array ([1,1,1])
theta_and = 3
for x in [(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)]:
  print(x, mp_neuron(np.array(x), w_and, theta_and))


print("--------OR GATE(3-inputs)--------")

w_or = np.array ([1,1,1])
theta_or = 1
for x in [(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)]:
  print(x, mp_neuron(np.array(x), w_or, theta_or))

print("+------------------------------+")
print("Chirag Miglani, 1/23/SET/BCS/414")
print("22 January 2026")
```

```
--------AND GATE(3-inputs)--------
(0, 0, 0) 0
(0, 0, 1) 0
(0, 1, 0) 0
```

```
(0, 1, 1) 0
(1, 0, 0) 0
(1, 0, 1) 0
(1, 1, 0) 0
(1, 1, 1) 1
--------OR GATE(3-inputs)--------
(0, 0, 0) 0
(0, 0, 1) 1
(0, 1, 0) 1
(0, 1, 1) 1
(1, 0, 0) 1
(1, 0, 1) 1
(1, 1, 0) 1
(1, 1, 1) 1
+------------------------------+
Chirag Miglani, 1/23/SET/BCS/414
22 January 2026
```

Artificial Neuron

```
import numpy as np

def art_neuron(x, w, b, activation='step'):
  z = np.dot(x, w) + b

  if activation == 'step':
    return 1 if z >= 0 else 0
  elif activation == 'sigmoid':
    return 1 / (1 + np.exp(-z))
  elif activation == 'relu':
    return max(0, z)
  else:
    return "Invalid Activation Function"

x = np.array([1,2])
w = np.array([0.5,1.5])
b = -1

print("--------------------")
print("Step Output     :", art_neuron(x, w, b, activation='step'))
print("--------------------")
print("Sigmoid Output     :", art_neuron(x, w, b, activation='sigmoid'))
print("--------------------")
print("ReLu Output     :", art_neuron(x, w, b, activation='relu'))


print("+------------------------------+")
print("Chirag Miglani, 1/23/SET/BCS/414")
print("22 January 2026")
```

```
--------------------
Step Output     : 1
--------------------
Sigmoid Output     : 0.9241418199787566
--------------------
ReLu Output     : 2.5
+------------------------------+
Chirag Miglani, 1/23/SET/BCS/414
22 January 2026
```

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("Chirag Miglani, 1/23/SET/BCS/414")
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
+------------------------------+
Chirag Miglani, 1/23/SET/BCS/414
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("Chirag Miglani, 1/23/SET/BCS/414")
print("22 January 2026")
```

```
[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. 1.] -3
error :  0 <- at epoch 9
[2. 1.] -3
+------------------------------+
Chirag Miglani, 1/23/SET/BCS/414
22 January 2026
```

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("Chirag Miglani, 1/23/SET/BCS/414")
print("22 January 2026")
```

```
[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
error :  0 <- at epoch 7
[1 3] -4
error :  0 <- at epoch 7
[1 3] -4
error :  0 <- at epoch 7
[1 3] -4
error :  0 <- at epoch 8
[1 3] -4
error :  0 <- at epoch 8
[1 3] -4
error :  0 <- at epoch 8
[1 3] -4
error :  0 <- at epoch 8
[1 3] -4
error :  0 <- at epoch 9
[1 3] -4
error :  0 <- at epoch 9
[1 3] -4
error :  0 <- at epoch 9
[1 3] -4
error :  0 <- at epoch 9
[1 3] -4
+------------------------------+
Chirag Miglani, 1/23/SET/BCS/414
22 January 2026
```

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 iterations
w,b = np.zeros(2),0

for _ in range(epochs):
  errors_in_epoch = False # Flag to track errors in the current epoch
  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 for all the following inputs.")
    print("+------------------------------+")
    print("Chirag Miglani, 1/23/SET/BCS/414")
    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 following inputs.
+------------------------------+
Chirag Miglani, 1/23/SET/BCS/414
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 for all inputs.")
    break

print("+------------------------------+")
print("Chirag Miglani, 1/23/SET/BCS/414")
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.
+------------------------------+
Chirag Miglani, 1/23/SET/BCS/414
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 for all inputs.")
    break

print("+------------------------------+")
print("Chirag Miglani, 1/23/SET/BCS/414")
print("22 January 2026")
```

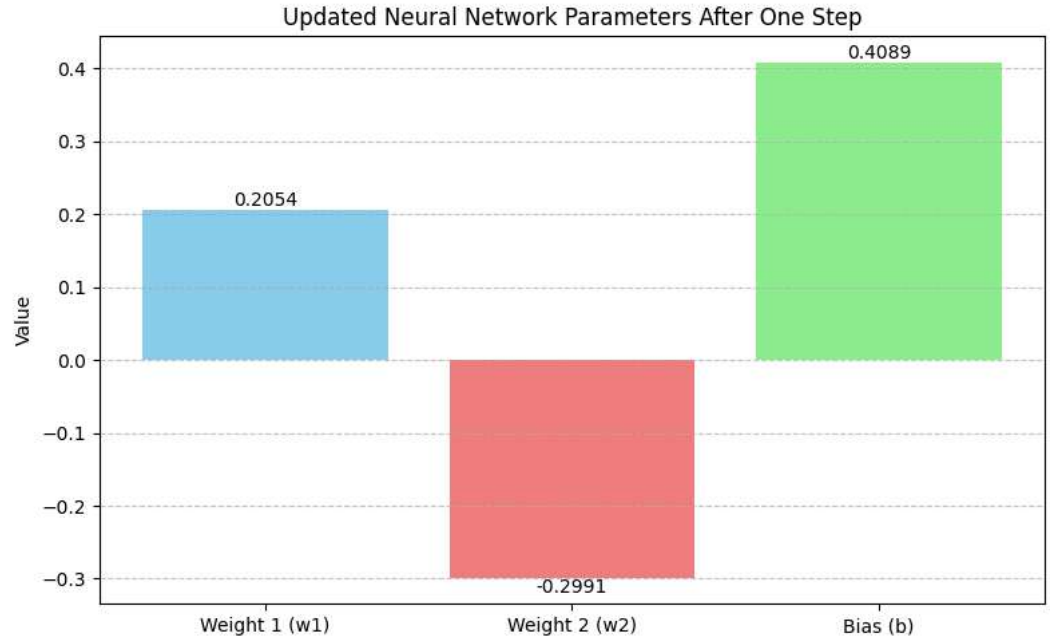
```
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 3
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 3
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 3
Updated weights: [-1.  0.], Updated bias: 0
Error: -1, Epoch: 4
Updated weights: [-1.  0.], Updated bias: -1
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
Error: 1, Epoch: 7
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 7
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 7
Updated weights: [-1.  0.], Updated bias: 0
Error: -1, Epoch: 8
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 8
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 8
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 8
Updated weights: [-1.  0.], Updated bias: 0
Error: -1, Epoch: 9
Updated weights: [-1.  0.], Updated bias: -1
Error: 1, Epoch: 9
Updated weights: [-1.  1.], Updated bias: 0
Error: 1, Epoch: 9
Updated weights: [0. 1.], Updated bias: 1
Error: -1, Epoch: 9
Updated weights: [-1.  0.], Updated bias: 0
+-----------------------------+
Chirag Miglani, 1/23/SET/BCS/414
22 January 2026
```

Lab - 3 | 05 February 2026

```
import numpy as np
import matplotlib.pyplot as plt

x1, x2 = 0.6, 0.1
y = 1
w1, w2, b = 0.2, -0.3, 0.4
lr = 0.1

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

z = w1 * x1 + w2 * x2 + b
y_pred = sigmoid(z)

dL_dy_pred = y_pred - y

Dy_pred_dz = y_pred * (1 - y_pred)

dL_dz = dL_dy_pred * Dy_pred_dz
```

```
dL_dw1 = dL_dz * x1
dL_dw2 = dL_dz * x2
dL_db = dL_dz

w1 -= lr * dL_dw1
w2 -= lr * dL_dw2
b -= lr * dL_db

print(f"------------------------Update W1 is stated below:------------------------")
print(f"{w1:.4f}, w2: {w2:.4f}, b: {b:.4f}\n")
print(f"--------------------------------------------------------------------------")

labels = ['Weight 1 (w1)', 'Weight 2 (w2)', 'Bias (b)']

values = [w1, w2, b]

plt.figure(figsize=(8, 5))
plt.bar(labels, values, color=['skyblue', 'lightcoral', 'lightgreen'])

plt.title('Updated Neural Network Parameters After One Step')
plt.ylabel('Value')
plt.grid(axis='y', linestyle='--', alpha=0.7)

for i, value in enumerate(values):
    plt.text(i, value, f'{value:.4f}', ha='center', va='bottom' if value >= 0 else 'top')

plt.tight_layout()
plt.show()
```

```
------------------------Update W1 is stated below:------------------------
0.2054, w2: -0.2991, b: 0.4089

--------------------------------------------------------------------------
```




```
import numpy as np
import matplotlib.pyplot as plt

X = np.array([
    [0,0],
    [0,1],
    [1,0],
    [1,1]
])
y = np.array([0,0,0,1])

w = np.array([0.1, 0.2])
b = 0.3
lr = 0.1

def sigmoid(z):
    return 1/(1+np.exp(-z))
```

```
    return 1/(1+np.exp(-z))

errors_per_epoch = []

for epoch in range(1000):
    total_absolute_error = 0
    for i in range(len(X)):
        z = np.dot(X[i], w) + b
        y_pred = sigmoid(z)

        total_absolute_error += abs(y[i] - y_pred)

        dz = y_pred - y[i]

        dw = dz * X[i]
        db = dz

        w -= lr * dw
        b -= lr * db

    errors_per_epoch.append(total_absolute_error)

print("---------- Trained Perceptron Parameters ----------")
print(f"Final Weights: {np.round(w, 4)}")
print(f"Final Bias: {np.round(b, 4)}")
print("--------------------------------------------------")

epoch_numbers = range(1, len(errors_per_epoch) + 1)

plt.figure(figsize=(10, 6))
plt.plot(epoch_numbers, errors_per_epoch, marker='o', linestyle='-', color='skyblue', markersize=4)
plt.title('Total Absolute Error per Epoch (AND Gate Perceptron)')
plt.xlabel('Epoch')
plt.ylabel('Total Absolute Error')
plt.grid(True)
plt.show()
```

```
---------- Trained Perceptron Parameters ----------
Final Weights: [5.6015 5.5954]
Final Bias: -8.566
--------------------------------------------------
```