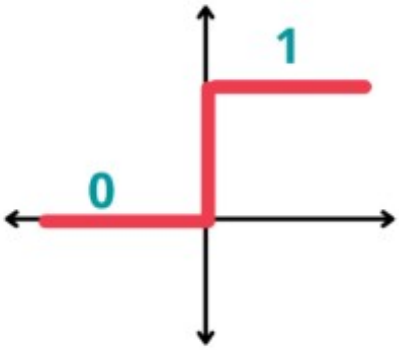
Lab 1

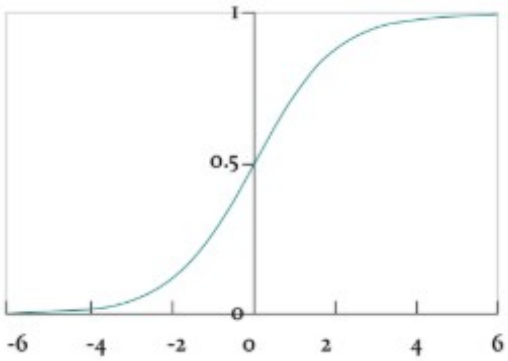
Activation functions

Step function



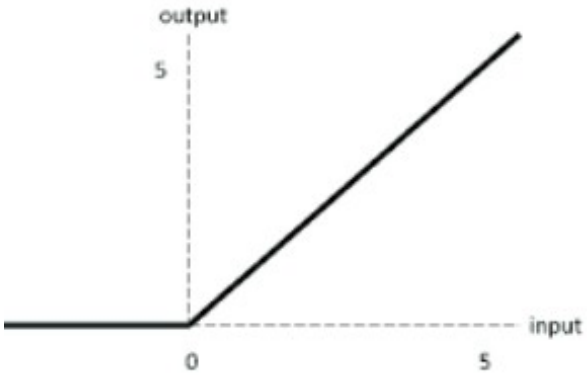

```
def step_function(x): return 1 if (x >= 0) else 0
```

sigmoid funtion




```
def sigmoid(x):
  from math import exp
  return 1/(1+exp(-x))
```

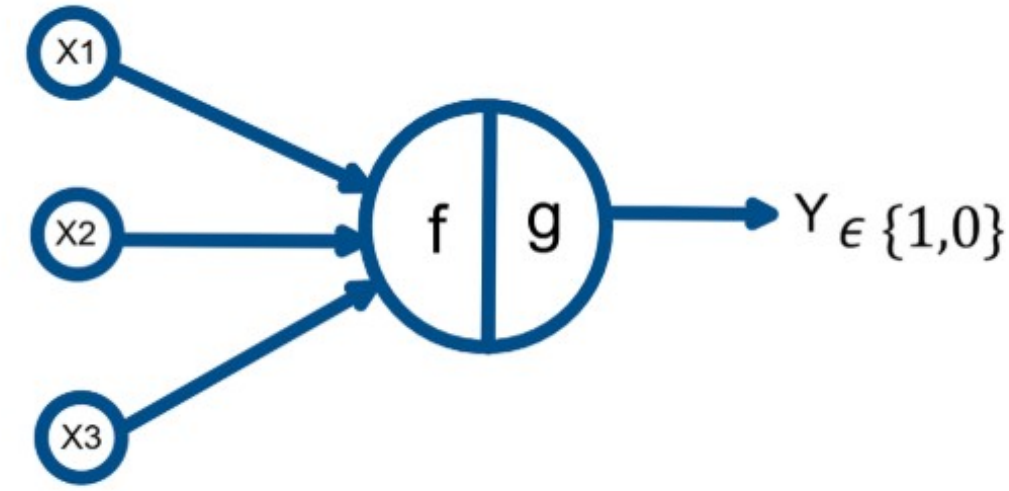
Relu function




```
def relu(x): return max(0,x)
```

Mc Culloch pitts neuron

implementation for AND, OR gate

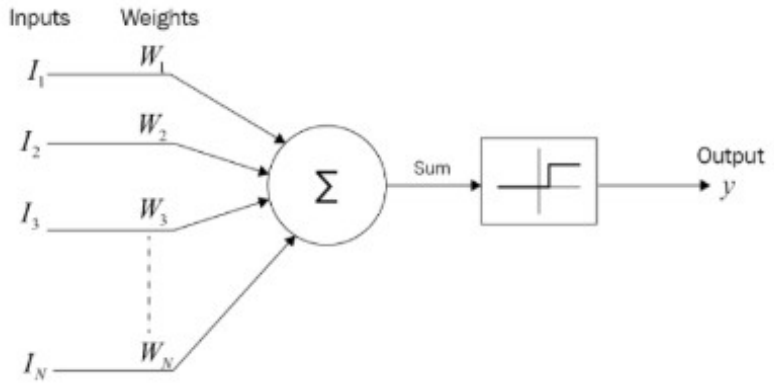



```
import numpy as np

def mc_and(x,w,theta): return 1 if np.dot(x,w) >= theta else 0

w,b = np.ones(3),1
x = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
print("AND gate for 3 input : ",end=' ')
for i in x:
    print(mc_and(i,w,theta=len(x[0])),end=" ") #AND gate for theta = 3
```

```
print("\nOR gate for 3 input : ",end=' ')
for i in x:
    print(mc_and(i,w,theta=1),end=" ") #OR gate for theta = 1
```

```
AND gate for 3 input :  0 0 0 1
OR gate for 3 input :  1 1 1 1
```

Perceptron neuron




```
import numpy as np

class Perceptron:
    def __init__(self, input_dim, epochs=3):
        self.w = np.zeros(input_dim)
        self.b = 0.0
        self.epochs = epochs -1

    def predict(self, x):
        z = np.dot(x, self.w) + self.b
        return 1 if z >= 0 else 0

    def fit(self, X, y,cond=1):
        self.cond = cond
        ep = 0
        while ep <= self.epochs:

            for xi, yi in zip(X, y):
                y_pred = self.predict(xi)
                err = yi - y_pred


                self.w += err * xi
                self.b += err

                if ep % self.cond == 0:
                  print("error rate:", err)
                  print(f"\nEpoch {ep}")
```

```
                    print("w:", self.w, "b:", self.b)

                ep += 1
```

AND Gate

```
X_and = np.array([[0, 0],[0, 1],[1, 0],[1, 1]])
y_and = np.array([0, 0, 0, 1])

and_gate = Perceptron(input_dim=2, epochs=30)
and_gate.fit(X_and, y_and, cond=5)

print("\nAND Gate Results:")
for x in X_and:
    print(x, "->", and_gate.predict(x))
```

```
error rate: -1

Epoch 0
w: [0. 0.] b: -1.0
error rate: -1

Epoch 5
w: [1. 0.] b: -2.0
error rate: -1

Epoch 10
w: [1. 0.] b: -3.0
error rate: 1

Epoch 15
w: [2. 2.] b: -2.0
error rate: 0

Epoch 20
w: [2. 1.] b: -3.0
error rate: 0

Epoch 25
w: [2. 1.] b: -3.0
error rate: 0

Epoch 30
w: [2. 1.] b: -3.0

AND Gate Results:
[0 0] -> 0
[0 1] -> 0
```

```
[1 0] -> 0
[1 1] -> 1
```

OR Gate

```
X_or = np.array([[0, 0],[0, 1],[1, 0],[1, 1]])
y_or = np.array([0, 1, 1, 1])

or_gate = Perceptron(input_dim=2, epochs=10)
or_gate.fit(X_or, y_or)

print("\nOR Gate Results:")
for x in X_or:
    print(x, "->", or_gate.predict(x))
```

```
error rate: -1

Epoch 0
w: [0. 0.] b: -1.0
error rate: 1

Epoch 1
w: [0. 1.] b: 0.0
error rate: 0

Epoch 2
w: [0. 1.] b: 0.0
error rate: 0

Epoch 3
w: [0. 1.] b: 0.0
error rate: -1

Epoch 4
w: [0. 1.] b: -1.0
error rate: 0

Epoch 5
w: [0. 1.] b: -1.0
error rate: 1

Epoch 6
w: [1. 1.] b: 0.0
error rate: 0

Epoch 7
w: [1. 1.] b: 0.0
error rate: -1

Epoch 8
```

```
w: [1. 1.] b: -1.0
error rate: 0

Epoch 9
w: [1. 1.] b: -1.0
error rate: 0

Epoch 10
w: [1. 1.] b: -1.0
error rate: 0

Epoch 11
w: [1. 1.] b: -1.0

OR Gate Results:
[0 0] -> 0
[0 1] -> 1
[1 0] -> 1
[1 1] -> 1
```