Perceptron for AND

In [1]:
```
import numpy as np
#First Code
class Perceptron(object):
    """Implements a perceptron network"""
    def __init__(self, input_size):
        self.W = np.zeros(input_size+1)

#Second Code
    def activation_fn(self, x):
        return 1 if x >= 0 else 0

# Third Code
    def predict(self, x):
        x = np.insert(x, 0, 1)
        z = self.W.T.dot(x)
        a = self.activation_fn(z)
        return a


#fourth code
    def __init__(self, input_size, lr=1, epochs=10):
        self.W = np.zeros(input_size+1)
    # add one for bias
        self.epochs = epochs
        self.lr = lr

# fifth code
    def fit(self, X, d):
        for _ in range(self.epochs):
            for i in range(d.shape[0]):
                y = self.predict(X[i])
                e = d[i] - y
                self.W = self.W + self.lr * e * np.insert(X[i], 0, 1)
if __name__ == '__main__':
    X = np.array([
        [0, 0],
        [0, 1],
        [1, 0],
        [1, 1]
    ])
    d = np.array([0, 0, 0, 1])
```

```
    perceptron = Perceptron(input_size=2)
    perceptron.fit(X, d)
    print(perceptron.W)
```

```
[-3.  2.  1.]
```

Perceptron for OR

In [2]:
```
import numpy as np
#First Code
class Perceptron(object):
    """Implements a perceptron network"""
    def __init__(self, input_size):
        self.W = np.zeros(input_size+1)

#Second Code
    def activation_fn(self, x):
        return 1 if x >= 0 else 0

# Third Code
    def predict(self, x):
        x = np.insert(x, 0, 1)
        z = self.W.T.dot(x)
        a = self.activation_fn(z)
        return a


#fourth code
    def __init__(self, input_size, lr=1, epochs=10):
        self.W = np.zeros(input_size+1)
    # add one for bias
        self.epochs = epochs
        self.lr = lr

# fifth code
    def fit(self, X, d):
        for _ in range(self.epochs):
            for i in range(d.shape[0]):
                y = self.predict(X[i])
                e = d[i] - y
                self.W = self.W + self.lr * e * np.insert(X[i], 0, 1)
if __name__ == '__main__':
    X = np.array([
        [0, 0],
        [0, 1],
        [1, 0],
        [1, 1]
    ])
    d = np.array([0, 1, 1, 1])
```

```
    perceptron = Perceptron(input_size=2)
    perceptron.fit(X, d)
    print(perceptron.W)
```

```
[-1.  1.  1.]
```

In [ ]: