SINGLE LAYER PERCEPTRON

AND PROBLEM

CODE :-

```
class Perceptron(object):
    """Implements a perceptron network"""
    def _init_(self, input_size, lr=1, epochs=100):
        self.W = np.zeros(input_size+1)
        # add one for bias
        self.epochs = epochs
        self.lr = lr

    def activation_fn(self, x):
        #return (x >= 0).astype(np.float32)
        return 1 if x >= 0 else 0
    def predict(self, x):
        z = self.W.T.dot(x)
        a = self.activation_fn(z)
        return a
    def fit(self, X, d):
        for _ in range(self.epochs):
            for i in range(d.shape[0]):
                x = np.insert(X[i], 0, 1)
                y = self.predict(x)
                e = d[i] - y
                self.W = self.W + self.lr * e * x
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("bias : ",perceptron.W[0])
    print("weights : ",perceptron.W[1:])
```

OUTPUT :-

```
Bias :  -3.0
Weights :  [2. 1.]
```

OR PROBLEM

CODE :-

```
class Perceptron(object):
    """Implements a perceptron network"""
    def _init_(self, input_size, lr=1, epochs=100):
        self.W = np.zeros(input_size+1)
        # add one for bias
        self.epochs = epochs
        self.lr = lr

    def activation_fn(self, x):
        #return (x >= 0).astype(np.float32)
        return 1 if x >= 0 else 0
    def predict(self, x):
        z = self.W.T.dot(x)
        a = self.activation_fn(z)
        return a
    def fit(self, X, d):
        for _ in range(self.epochs):
            for i in range(d.shape[0]):
                x = np.insert(X[i], 0, 1)
                y = self.predict(x)
                e = d[i] - y
                self.W = self.W + self.lr * e * x
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("bias : ",perceptron.W[0])
    print("weights : ",perceptron.W[1:])
```

OUTPUT :-

```
Bias :  -1.0
Weights :  [1. 1.]
```

XOR PROBLEM

Perceptrons can only solve linearly separable problems. In the real world, however, many problems are actually linearly separable. For example, we can use a perceptron to mimic an AND or OR gate. However, since XOR is not linearly separable, we can't use single-layer perceptrons to create an XOR gate.