```
import numpy as np


def sigmoid(x):
  return 1.0/(1.0 + np.exp(-x))

def sigmoid_der(x):
  return x*(1.0 - x)

class Perceptron:
    def __init__(self, inputs):
        self.inputs = inputs
        self.l=len(self.inputs)
        self.li=len(self.inputs[0])

        self.wi=np.random.random((self.li, self.l))
        self.wh=np.random.random((self.l, 1))

    def gen(self, inp):
        s1=sigmoid(np.dot(inp, self.wi))
        s2=sigmoid(np.dot(s1, self.wh))
        return s2

    def train(self, inputs,outputs, it):
        for i in range(it):
            l0=inputs
            l1=sigmoid(np.dot(l0, self.wi))
            l2=sigmoid(np.dot(l1, self.wh))

            l2_err=outputs - l2
            l2_delta = np.multiply(l2_err, sigmoid_der(l2))

            l1_err=np.dot(l2_delta, self.wh.T)
            l1_delta=np.multiply(l1_err, sigmoid_der(l1))

            self.wh+=np.dot(l1.T, l2_delta)
            self.wi+=np.dot(l0.T, l1_delta)
```

```
inp = np.array([[0,0], [0,1], [1,0], [1,1] ])
out = np.array([ [0], [1],[1],[0] ])


n = Perceptron(inp)
print(n.gen(inp))
n.train(inp, out, 10000)
print(n.gen(inp))
```

```
[[0.78934885]
 [0.8367789 ]
 [0.83393277]
 [0.86973592]]
[[0.02372013]
 [0.98131248]
 [0.98167362]
 [0.01462259]]
```