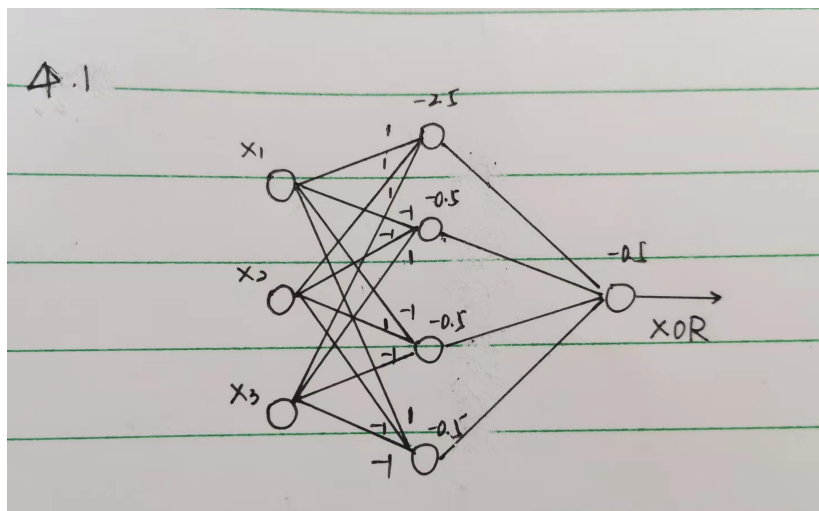


机器学习 MLP 作业

姓名：张泽群 学号：19049100002 班级：1班

4.1

- (1). 不可以，无法处理非线性可分问题
- (2). 可以。
- (3).



4.2

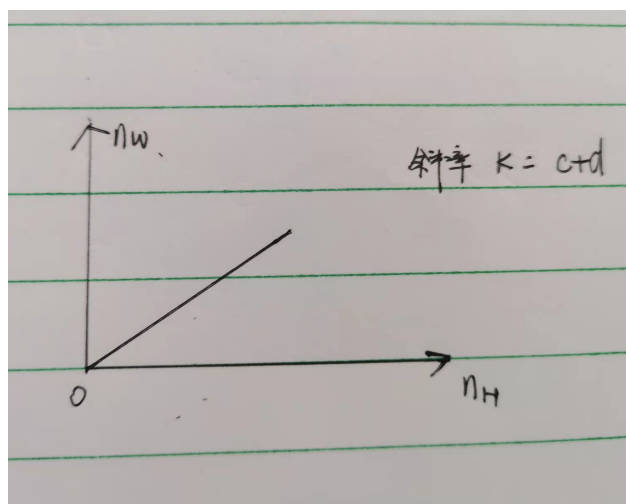
答：如果不使用非线性激活函数，此时激活函数本质上相当于 $f(x)=ax+b$ 。这种情况时，神经网络的每一层输出都是上层输入的线性函数。不难看出，不论神经网络有多少层，输出与输入都是线性关系，与没有隐层的效果是一样的，这个就是相当于是最原始的感知机(Perceptron)。至于感知机，大家知道其连最基本的异或问题都无法解决，更别提更复杂的非线性问题。

因为上述证明隐层为线性激活函数的三层网络等价于两层网络，而二层网络只能处理线性可分问题，具有线性隐藏单元的三层网络无法解决非线性可分离问题。

4.3

(a). 权重参数的个数为 $n_H * (d+c)$

(b). $n_w = n_H * (d+c)$



4.4

$$4.4 \quad \frac{df(x)}{dx} = \frac{ae^{ax}}{(1+e^{ax})^2} = f(x)^2 \left[\frac{1}{f(x)} - 1 \right] a = -af(x)[1-f(x)]$$

4.6

```
import numpy as np

# 激活函数
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# 定义sigmoid函数导数
def dsigmoid(x):
    return x * (1 - x)

# 输入数据集
X = np.array([[0, 0],
               [0, 1],
               [1, 0],
               [1, 1]])

# 输出数据集
Y = np.array([[0],
               [1],
               [1],
               [0]])

# 初始化权重矩阵
syn0 = 2*np.random.random((2, 2)) - 1 # 输入层到隐层的权重矩阵
syn1 = 2*np.random.random((2, 1)) - 1 # 隐层到输出层的权重矩阵

for index in range(60000): # 误差反向传播计算
    L0 = X
    L1 = sigmoid(np.dot(L0, syn0))
    L2 = sigmoid(np.dot(L1, syn1))

    L_rate = 0.2 # 学习率
    L2_error = Y - L2
    L2_delta = L2_error * dsigmoid(L2) * L_rate

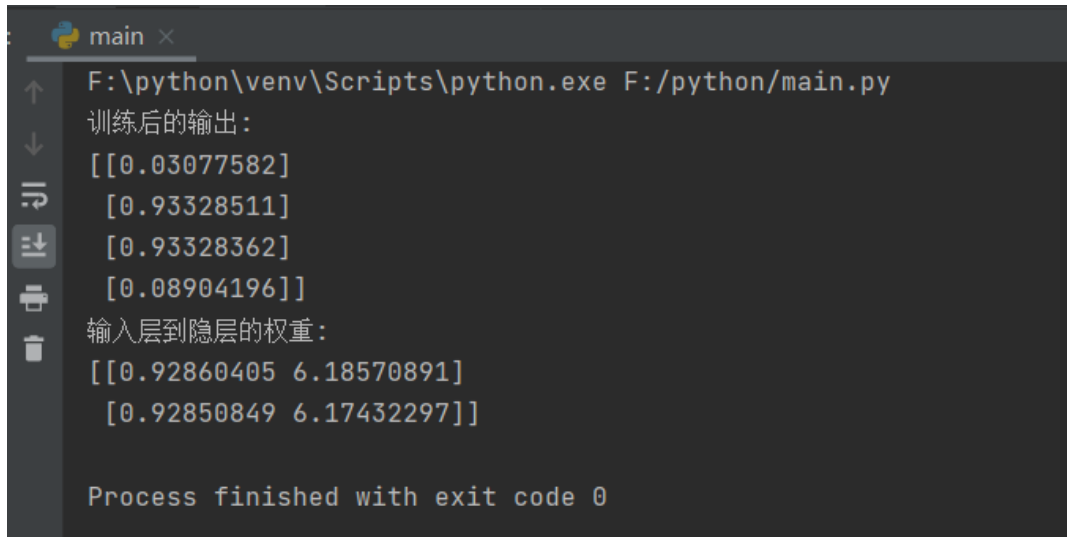
    L1_error = L2_delta.dot(syn1.T)
    L1_delta = L1_error * dsigmoid(L1) * L_rate

    syn1 += L1.T.dot(L2_delta)
    syn0 += L0.T.dot(L1_delta)

print("训练后的输出:")
print(L2)

print("输入层到隐层的权重:")
print(syn0)
```

运行结果：



```
main x
F:\python\venv\Scripts\python.exe F:/python/main.py
训练后的输出：
[[0.03077582]
 [0.93328511]
 [0.93328362]
 [0.08904196]]
输入层到隐层的权重：
[[0.92860405 6.18570891]
 [0.92850849 6.17432297]]

Process finished with exit code 0
```