《神经网络与深度学习》第二次作业

# 1 作业要求

请用python语言设计一个三层的BP全连接神经网络，两个输入，两个输出，具体结构不限，提供6个训练样本，4个测试样本，实现网络的训练和测试，并提供软件使用说明。

# 2 实现

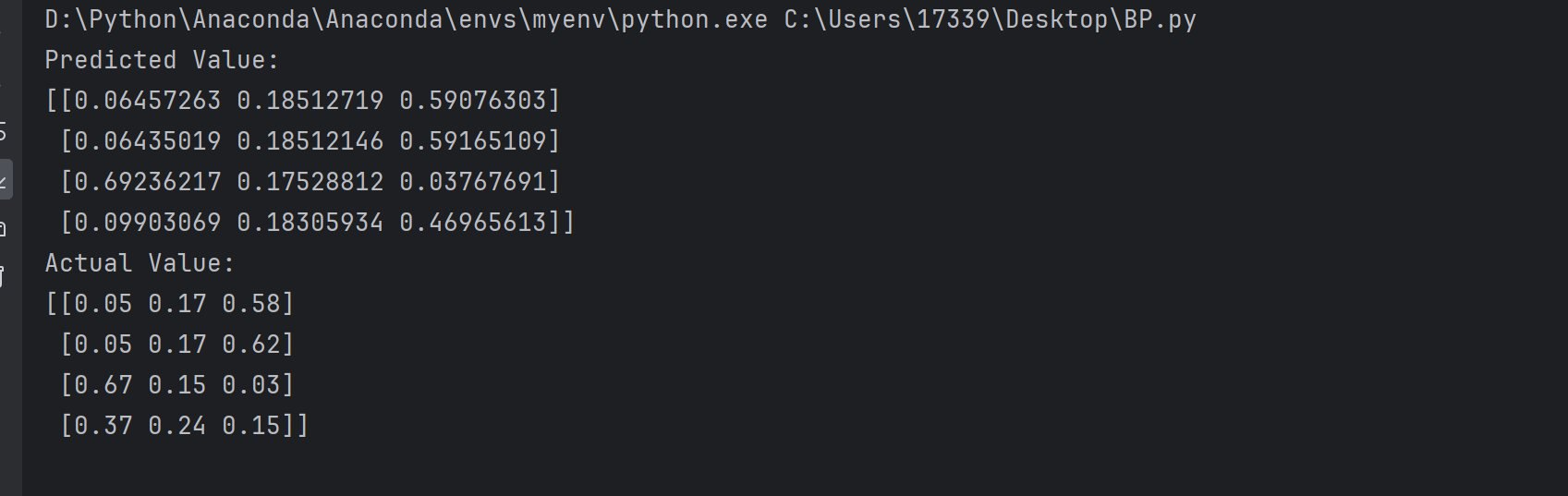
## 2.1 BP神经网络的构建

import numpy as np  
  
*# 定义激活函数*def sigmoid(x):  
 return 1 / (1 + np.exp(-x))  
  
*# 定义神经网络类*class BPNetwork:  
 def \_\_init\_\_(self):  
 *# 初始化权重* self.weights1 = np.random.rand(2, 3)  
 self.weights2 = np.random.rand(3, 3)  
 self.weights3 = np.random.rand(3, 3)  
 *# 初始化偏置* self.bias1 = np.random.rand(1, 3)  
 self.bias2 = np.random.rand(1, 3)  
 self.bias3 = np.random.rand(1, 3)  
  
 def forward(self, x):  
 *# 前向传播，每一层的激活函数均为sigmoid* self.hidden1 = sigmoid(np.dot(x, self.weights1) + self.bias1)  
 self.hidden2 = sigmoid(np.dot(self.hidden1, self.weights2) + self.bias2)  
 self.output = sigmoid(np.dot(self.hidden2, self.weights3) + self.bias3)  
 return self.output  
  
 def train(self, x, y, learning\_rate=0.01):  
 *# 反向传播，1000epoch* for i in range(100000):  
 *# 前向传播* self.forward(x)  
  
 error = y - self.output *# 误差  
 #cost = error\*error  
 #print(f"cost=",error)  
  
 # sigmoid梯度公式 f'(x)=f(x)(1-f(x))* output\_delta = error \* self.output \* (1 - self.output) *#连接层3的梯度* hidden2\_error = np.dot(output\_delta, self.weights3.T) *# 连接层2的误差* hidden2\_delta = hidden2\_error \* self.hidden2 \* (1 - self.hidden2) *# 连接层2的梯度* hidden1\_error = np.dot(hidden2\_delta, self.weights2.T) *# 连接层1的误差* hidden1\_delta = hidden1\_error \* self.hidden1 \* (1 - self.hidden1) *# 连接层1的梯度  
  
 # 更新权重* self.weights3 += learning\_rate \* np.dot(self.hidden2.T, output\_delta)  
 self.weights2 += learning\_rate \* np.dot(self.hidden1.T, hidden2\_delta)  
 self.weights1 += learning\_rate \* np.dot(x.T, hidden1\_delta)  
  
 *# 更新偏执* self.bias3 += learning\_rate \* np.sum(output\_delta, axis=0)  
 self.bias2 += learning\_rate \* np.sum(hidden2\_delta, axis=0)  
 self.bias1 += learning\_rate \* np.sum(hidden1\_delta, axis=0)  
  
 def predict(self, x):  
 *# 预测* return self.forward(x)  
  
*# 创建神经网络实例*nn = BPNetwork()  
  
*# 训练样本*X\_train = np.array([[20,10], [1,5], [6, 14], [6,2], [10,4], [12,2]])  
y\_train = np.array([[0.44,0.22, 0.11], [0.03,0.14,0.69], [0.09,0.21, 0.49], [0.5625,0.1875,0.0625], [0.51,0.20, 0.08], [0.73,0.12,0.02]])  
  
*# 测试样本*X\_test = np.array([[2,7], [3,11], [9,2], [11,7]])  
y\_test = np.array([[0.05,0.17,0.58], [0.05,0.17,0.62], [0.67,0.15,0.03], [0.37,0.24,0.15]])  
  
*# 训练神经网络*nn.train(X\_train, y\_train)

## 2.2 数据样本构建和模型训练

*# 训练样本*X\_train = np.array([[20,10], [1,5], [6, 14], [6,2], [10,4], [12,2]])  
y\_train = np.array([[0.44,0.22, 0.11], [0.03,0.14,0.69], [0.09,0.21, 0.49], [0.5625,0.1875,0.0625], [0.51,0.20, 0.08], [0.73,0.12,0.02]])  
  
*# 测试样本*X\_test = np.array([[2,7], [3,11], [9,2], [11,7]])  
y\_test = np.array([[0.05,0.17,0.58], [0.05,0.17,0.62], [0.67,0.15,0.03], [0.37,0.24,0.15]])  
  
*# 训练神经网络*nn.train(X\_train, y\_train)  
  
*# 测试神经网络*predictions = nn.predict(X\_test)  
print("Predicted Value:")  
print(predictions)  
print("Actual Value:")  
print(y\_test)

## 2.3 运行结果



## 2.4 功能解释

构建了一个三个连接层的BP神经网络，每个连接层的激活函数均为sigmoid函数。整个网络的输入格式为，输出格式为。训练样本按照以下公式给出：

根据运行结果来看，训练后的BP神经网络很好的拟合了公式，但同样存在预测错误的情况。