实验六：神经网络

代码：

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import random  
import math  
  
dataset = pd.read\_csv('D:\FileRecv\人工智能课程实验\课程实验\第六节 神经网络watermelon\_3.csv', delimiter=" ")  
  
dataset=np.array(dataset)  
m,n=np.shape(dataset)  
for i in range(m):  
 for j in range(n):  
 dataset[i,j]=round(dataset[i,j],3) #定义数据格式，dataset[i,j]保留小数点后3位  
  
trueY=dataset[:,n-1] #取类别标签  
X=dataset[:,:n-1] #取属性部分  
m,n=np.shape(X) #行列值  
  
#according to P101, init the parameters  
# v = d\*q .输入层到输出层权重  
v=[[random.random() for i in range(n+1)] for j in range(n)]  
# w = q\*l .隐藏层到输出层权重  
w=[[random.random() for i in range(1)] for j in range(n+1)]  
  
#激活函数  
def sigmoid(iX,dimension):#iX一维矩阵  
 if dimension==1:  
 for i in range(len(iX)):  
 iX[i] = 1 / (1 + math.exp(-iX[i]))  
 else:  
 for i in range(len(iX)):  
 iX[i] = sigmoid(iX[i],dimension-1)  
 return iX  
  
'''  
累积误差逆传播算法：  
累积BP算法直接针对累积误差最小化，他在读取整个训练集D一遍后才对  
参数进行更新，其参数更新的频率低得多  
'''  
eta=0.2 #训练速率  
d=n #输入向量的维度  
l=1 #输出向量的维度  
q=d+1 #隐层神经元的数目  
theta=[random.random() for i in range(l)] #输出层神经元的阈值  
gamma=[random.random() for i in range(q)] #隐层神经元阈值  
trueY=trueY.reshape((m,l))  
maxIter=5000 #最大训练时间  
while(maxIter>0):  
 maxIter-=1  
 sumE=0  
 alpha = np.dot(X, v)#p101 line 2 from bottom, shape=m\*q  
 b = sigmoid(alpha - gamma,2) # b=f(alpha-gamma), shape=m\*q  
 beta = np.dot(b, w) # shape=(m\*q)\*(q\*l)=m\*l  
 predictY = sigmoid(beta - theta,2) # shape=m\*l ,p102--5.3  
  
 E = sum(sum((predictY - trueY) \* (predictY - trueY))) / 2 # 5.4 均方误差  
 g = predictY \* (1 - predictY) \* (trueY - predictY) # shape=m\*l p103--5.10  
 e = b \* (1 - b) \* ((np.dot(w, g.T)).T) # shape=m\*q , p104--5.15  
 w += eta \* np.dot(b.T, g) #shape (q\*l)=(q\*m) \* (m\*l) 隐层到输出层连接权  
 theta -= eta \* g # 5.12 阈值  
 v += eta \* np.dot(X.T, e) # 5.13 (d,q)=(d,m)\*(m,q) 输入层到隐层连接权  
 gamma -= eta \* e # 5.14 阈值  
  
  
def predict(iX):  
 *'''  
 beta： 输出层神经元接收到的输入  
 theta：输出层神经元的阈值  
 '''* alpha = np.dot(iX, v) #从输入层到隐层  
 b=sigmoid(alpha-gamma,2) # 隐层输出  
 beta = np.dot(b, w) #从隐层到输出层  
 predictY=sigmoid(beta - theta,2) #输出层输出  
 return predictY  
  
def plotBestFit(dataArr,labelMat1,labelMat2):  
 *'''  
 分类效果展示  
 @****:param*** *weights 回归系数  
 @****:param*** *path 数据文件路径  
 @****:return*** *null  
 '''* n = len(dataArr) #取行数  
 xcord1 = []; ycord1 = []  
 xcord2 = []; ycord2 = []  
 xcord3 = []; ycord3 = []  
 xcord4 = []; ycord4 = []  
  
 for i in range(n): #将训练前的数据分类存储  
 if int(labelMat1[i])== 1:#  
 xcord1.append(dataArr[i][0]); ycord1.append(dataArr[i][1])  
 else:  
 xcord2.append(dataArr[i][0]); ycord2.append(dataArr[i][1])  
 for i in range(n): #将训练后的数据分类存储  
 if int(labelMat2[i])== 1:  
 xcord3.append(dataArr[i][0]); ycord3.append(dataArr[i][1])  
 else:  
 xcord4.append(dataArr[i][0]); ycord4.append(dataArr[i][1])  
  
 '''  
 神经网络预测结果  
 '''  
 plt.figure("BPML1")  
 plt.title('Original')  
 plt.scatter(xcord1, ycord1, s=30, c='red', marker='s')  
 plt.scatter(xcord2, ycord2, s=30, c='green')  
 plt.xlabel('X1');plt.ylabel('X2')  
  
 fig = plt.figure("BPML2") #新建一个画图窗口  
 ax = fig.add\_subplot(111) #添加一个子窗口  
 ax.set\_title('Forecast')  
 ax.scatter(xcord3, ycord3, s=30, c='red', marker='s')  
 ax.scatter(xcord4, ycord4, s=30, c='green')  
 plt.xlabel('X1'); plt.ylabel('X2')  
 plt.show()  
  
def main():  
 result = predict(X)  
 h = []  
 for i in range(len(result)):  
 if result[i] > 0.5:  
 h.append(1)  
 else:  
 h.append(0)  
 plotBestFit(dataset,trueY,h)  
  
  
main()