//图片导入成为向量

def loadimages( i )

//输入i , 图片转为i \* i的矩阵

//N张图片转成N个 [i^2 + 1 , 1]的向量

return DataSet , Labels

//权值初始化

def WeightInit(m , i , j):

//第m层，相连的两层的神经元数量为i与j

return W(大小为m\*j\*i)

//激励函数

def f(x):

return sigmoid(x);

//f(x)的导数

def ft(x):

return f(x)\*(1-f(x))

//前向推导

def forward(M , W , data , label) :

/\*M是层数向量，代表每一层的神经元数量，例如[267 , 100 , 10]，W是权值矩阵，data是单个数据，label是该数据对应的标签\*/

net = []; O = [];y=[];E=[];

O[0] = data;

net[0]=0;

for m=1 to len(M - 1):

net[m]=W[m-1]\*O[m-1]; //第m层神经元净输入

O[m] = f(net[m]);//第m层神经元净输出

y = O[len(M-1)];

E = 1/2\*(y-label)^2;

return net , O , y , E

//后向推导

def backward(M , W , net , O , y , E，label):

grad= [];

for m = len(M)-1 to 1: //从输出层回退

for j = 0 to M(m)-1: //第m层j个神经元

for i = 0 to M(m-1)-1:

if(m = len(M)-1): //如果是输出层

grad[m][j] = -(label - y)ft(net[m][j])

W[m-1][j][i] -= learningrate (常量)\*

gard[m][j]\*O[m-1][i]

else:

k = 0 to M(m+1)-1 //k是上一层的神经元数量 grad[m][j] = ft(net[m][j])\*

W[m-1][i][j] -= learningrate \* grad[m][j]\*O[m-1][i]

return W

//利用测试集计算出正确率

def test(testSet , testLabel , W):

return 正确率

main:

//定义i = 16

//定义M=[16^2+1 , 100 , 10]

trainDataSet , trainLabel = loadimages(i)

testDataSet , testLabel = loadiamges(i)

W = Weightinit();

for n = 0 to len(trainDataSet)-1:

net[n], O[n] , y[n] , E[n] = forward(M , W, trainDataSet[n] , trainLabel[n]);

W = backward(M , W , net[n] , O[n] , y[n] , E[n]);

//...训练完成后测试

test(testDataSet , testLabels , W)