### 实验一 Hopfield模型的实现

1982年，J．Hopfield提出了可用作联想存储器的互连网络，这个网络称为Hopfield网络模型，也称Hopfield模型。Hopfield神经网络模型是一种循环神经网络，从输出到输入有反馈连接。Hopfield网络有离散型和连续型两种。反馈神经网络由于其输出端有反馈到其输入端；所以，Hopfield网络在输入的激励下，会产生不断的状态变化。当有输入之后，可以求取出Hopfield的输出，这个输出反馈到输入从而产生新的输出，这个反馈过程一直进行下去。如果Hopfield网络是一个能收敛的稳定网络，则这个反馈与迭代的计算过程所产生的变化越来越小，一旦到达了稳定平衡状态；那么Hopfield网络就会输出一个稳定的恒值。对于一个Hopfield网络来说，关键是在于确定它在稳定条件下的权系数。

**实验目的：**加深对Hopfield模型的理解，能够使用Hopfield模型解决实际问题

**实验原理：**Hopfield网络从输出到输入有反馈连接，在输入的激励下，会产生不断的状态变化，是一种单层反馈神经网络，也可以被视为一种循环神经网络。Hopfield神经网络是反馈网络中最简单且应用广泛的模型，它具有联想记忆的功能，是神经网络发展历史上的一个重要的里程碑。离散Hopfield网络可以视为一个类脑模型，主要是因为其可用于联想记忆，即联想存储器，这是类人智能的特点之一。人类的所谓“触景生情”就是见到一些类同于过去接触的景物，容易产生对过去情景的回味和思忆。对于Hopfield 网络，用它作联想记忆时，首先通过一个学习训练的过程确定网络中的权系数，使所记忆的信息在网络的n维超立方体的某一个顶角达到能量最小，当网络的权重矩阵确定之后，只要向网络给出输入向量，这个向量可能是局部数据．即不完全或部分不正确的数据，但是网络仍然能够产生所记忆信息的完整输出。

离散Hopfield网络是一个单层网络，有n个神经元节点，每个神经元的输出均接到其它神经元的输入。各节点没有自反馈。每个节点都可处于一种可能的状态（1或－1），即当该神经元所受的刺激超过其阀值时，神经元就处于一种状态（比如1），否则神经元就始终处于另一状态（比如 －1）。

**实验内容：**根据Hopfield神经网络的相关知识，设计一个具有联想记忆功能的离散型Hopfiled神经网络。要求该网络可以正确识别0-9这10个数字，当数字被一定的噪声干扰后，仍具有较好的识别效果。

简单例子如下所示：

import random

def calcWeight(savedsample):

N = len(savedsample[0])

P = len(savedsample)

mat = [0]\*N

returnMat = []

for i in range(N):

m = mat[:]

returnMat.append(m)

for i in range(N):

for j in range(N):

if i==j:

continue

sum = 0

for u in range(P):

sum += savedsample[u][i] \* savedsample[u][j]

returnMat[i][j] = sum/float(N)

return returnMat

def calcXi(inMat , weighMat):

returnMat = inMat

choose = []

for i in range(len(inMat)//5):

#随机改变N/5个神经元的值，该参数可调，也可同时改变所有神经元的值

choose.append(random.randint(0,len(inMat)-1))

for i in choose:

sum = 0

for j in range(len(inMat)):

sum += weighMat[i][j] \* inMat[j]

if sum>=0:

returnMat[i] = 1

else: returnMat[i] = -1

return returnMat

sample = [[1,-1,-1,-1,1,

1,1,-1,-1,1,

1,-1,1,-1,1,

1,-1,-1,1,1,

1,-1,-1,-1,1],

[1,1,1,1,1,

1,-1,-1,-1,-1,

1,1,1,1,1,

1,-1,-1,-1,-1,

1,1,1,1,1],

[1,1,1,1,-1,

1,-1,-1,-1,1,

1,1,1,1,-1,

1,-1,-1,1,-1,

1,-1,-1,-1,1],

[-1,1,1,1,-1,

1,-1,-1,-1,1,

1,-1,-1,-1,1,

1,-1,-1,-1,1,

-1,1,1,1,-1]]

def addnoise(mytest\_data,n):

for x in range(n):

for y in range(n):

if random.randint(0, 10) > 7:

mytest\_data[x \* n + y] = -mytest\_data[x \* n + y]

return mytest\_data

def regularout(data,N):

for j in range(N):

ch = ""

for i in range(N):

ch += " " if data[j\*N+i] == -1 else "X"

print(ch)

weightMat = calcWeight(sample)

regularout(sample[1],5)

test = addnoise(sample[1],5)

regularout(test,5)

for i in range(2000):

test = calcXi(test,weightMat)

regularout(test,5)

**实验要求：**

1. 设计6\*5数字点阵。有数字部分用1表示，空白部分用-1表示，将数字0-9的矩阵设计好存储到1\*30矩阵中。
2. 创建网络。
3. 产生带噪声的数字点阵。带噪声的数字点阵，即点阵的某些位置的值发生了变化。模拟产生带噪声的数字矩阵方法有很多种，如固定噪声产生法和随机噪声产生法。
4. 数字识别测试。将带噪声的数字点阵输入到创建好Hopfiled网络，网络的输出是与该数字点阵最为接近的目标向量，从而实现联想记忆功能。