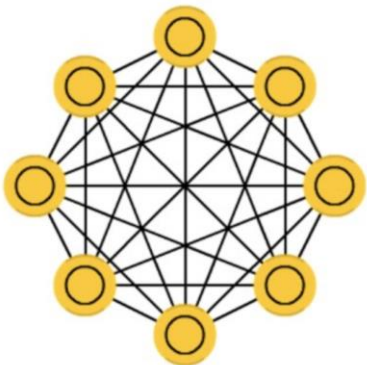


计算机科学与技术学院 认知科学与类脑计算实验 课程实验报告

实验题目：实验一 Hopfield 模型的实现		学号：201600150109
日期：2019.5.17	班级：2016 级人工智能	姓名：沈棋韬
Email: qitaoshen@gmail.com		
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		
实验软件和硬件环境： Windows10 Anaconda3.4 Python3.7		
实验原理和方法： 离散型Hopfield神经网络： <p>离散随机Hopfield神经网络：每个神经元只取二元的离散值0、1或-1、1。神经元i和神经元j之间的权重由 W_{ij} 决定。神经元有当前状态 u_i 和输出 v_i。虽然 u_i 可以使连续值，但 v_i 在离散模型中是二值的。神经元状态和输出的关系如下，也就是离散型Hopfield神经网络演化方程：</p> $u_i(t+1) = \sum_{j=1}^n W_{ij} v_j(t) + I_i$ $v_i(t+1) = f(u_i) = \begin{cases} 1 & \text{if } u_i > 0 \\ 0 & \text{if } u_i \leq 0 \end{cases}$ <p>http://blog.csdn.net/lixiaohu1992</p>  <p>上图为Hopfield神经网络结构图。</p>		

实验步骤：（不要求罗列完整源代码）

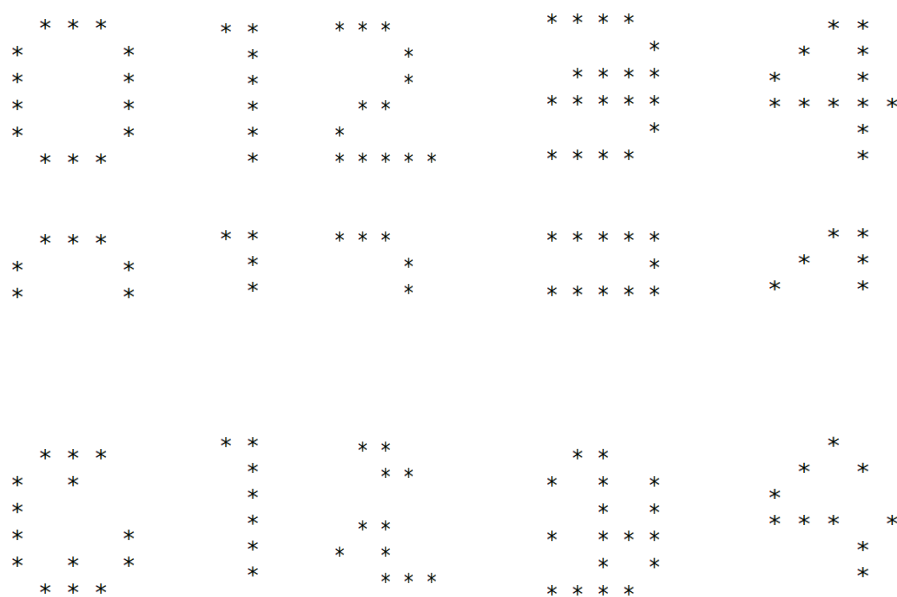
1. 调用 neupy 中的 hopfield 网络

输入用矩阵表示的数字，然后进行训练。

在测试时，先对图片加上噪声，再进行预测。

噪声：仅保留图片的下半部分

如图所示：从上向下分别为原始图片，加噪声后的图片，预测结果。



2. 自己实现基础的 hopfield 网络并训练

参考文献：

Hopfield 联想记忆网络运行步骤为：

第一步：设定记忆模式。将欲存储的模式进行编码，得到取值为1和-1的记忆模式($m < n$):

$$U_k = [u_1^k, u_2^k, \dots, u_i^k, \dots, u_n^k]^T \quad k=1, 2, \dots, m。$$

第二步：设计网络的权值。

$$w_{ij} = \begin{cases} \frac{1}{N} \sum_{k=1}^m u_i^k u_j^k, & i \neq j \\ 0, & i = j \end{cases}$$

其中 w_{ij} 一旦计算完毕，将保持不变。

第三步：初始化网络状态。将欲识别模式 $U = [u_1, u_2, \dots, u_i, \dots, u_n]^T$ 设为网络状态的初始状态， $v_i(0) = u_i$ 为网络中任意神经元 i 在 $t=0$ 时刻的状态。

第四步：迭代收敛。随机地更新某一神经元的状态

$v_i(t+1) = \text{sgn}[\sum_{j=1}^n w_{ij} x_j(n)]$ ，反复迭代直至网络中所有神经元的状态不变为止。

计算 weight:

```

def train(X):
    N = len(X[0])
    print("N: ", N)
    P = len(X)
    print("P: ", P)
    feature = [0]*N
    returnfeature = []
    for i in range(N):
        m = feature[:]
        returnfeature.append(m)
    for i in range(N):
        for j in range(N):
            if i==j:
                continue
            totalweight = 0
            for u in range(P):
                totalweight += X[u][i] * X[u][j]
            returnfeature[i][j] = totalweight/float(N)
    return returnfeature

```

预测结果:

```

def predict(infeature , weighfeature):
    returnfeature = infeature
    choose = []
    for i in range(len(infeature)):
        choose.append(random.randint(0,len(infeature)-1))
    for i in choose:
        totalweight = 0
        for j in range(len(infeature)):
            totalweight += weighfeature[i][j] * infeature[j]
        if totalweight >= 0:
            returnfeature[i] = 1
        else: returnfeature[i] = -1
    return returnfeature

```

噪声: 随机添加噪声:

测试样本时迭代 1000 次。

结果展示:

```

**      ***      ***      ****      *****      **      **
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *

**      ***      ***      ****      *****      **      **
****      *      *      *      *      *      *      *
****      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *

**      ***      ***      ****      *****      **      **
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *

```

结论分析与体会：

可见第二次自己实现的网络更加稳定，在噪声很强的情况下仍然能识别出原来的数字。

Hopfield 具有联想记忆功能。

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1—3 道问答题：

问题 1：结果一直不收敛。

解决方法：原本使用 0 和 1 来画出数字，现在改成-1 和 1， 并且把阈值设置成 0. 就解决了这个问题。