### 实验三 Hebb学习

由赫布提出的Hebb学习规则是一个无监督学习规则，这种学习的结果是使网络能够提取训练集的统计特性，从而把输入信息按照它们的相似性程度划分为若干类。这一点与人类观察和认识世界的过程非常吻合，人类观察和认识世界在相当程度上就是在根据事物的统计特征进行分类。Hebb学习规则只根据神经元连接间的激活水平改变权值，因此这种方法又称为相关学习或并联学习。

巴普洛夫实验是一个典型的Hebb学习模型，其具体内容是每次给狗送食物以前响起铃声，这样经过一段时间以后，铃声一响，狗就开始分泌唾液。

**实验目的：**加深对Hebb学习模型的理解，能够使用Hebb学习模型解决简单问题

**实验原理：**Hebb算法最简单可以描述为:如果一个处理单元从另一处理单元接收输入激励信号，而且如果两者都处于高激励电平，那么处理单元之间的加权就应当增强。用数学来表示，就是两节点的连接权将根据两节点的激励电平的乘积来改变，即

Δwij=wij(n+1)-wij(n)=ηyixj

其中wij(n)表示第(n+1)次调解前，从节点j到节点i的连接权值；wij(n+1)是第(n+1)次调解后，从节点j到节点i的连接权值；η为学习速率参考；xj为节点j的输出，并输入到节点i；yi为节点i的输出。

对于Hebb学习规则，学习信号简单地等于神经元的输出

r = f(WiTX)

权向量的增量变成

ΔWi= ηf(WiTX)X

这个学习规则在学习之前要求在 Wi = 0附近的小随机值上对权重进行初始化。这个规则说明了如果输出和输入的叉积是正的，则权增加，否则减小。

**实验内容：**根据Hebb学习模型的相关知识，使用Python语言实现一个简单Hebb学习模型。简单程序示例如下：

import numpy as np

class Hebb(object):

def \_\_init\_\_(self):

self.weights = np.zeros((30,30))

def train(self, input\_vector, iter\_, rate):

for i in range(iter\_):

for vec in input\_vector:

vec = np.matrix(vec)

#这里可以使用广义hebb算法中的权重更新算法

self.weights = self.weights + rate \* vec.getT().dot(vec)

return self.weights

def predict(self, input\_vector):

return self.weights.dot(np.matrix(input\_vector).getT())

zero = [

-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

]

one = [

-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

]

two = [

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,

]

half\_zero = [

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,

]

hebb = Hebb()

w=hebb.train([zero, one, two], 600, 0.2)

pre = hebb.predict(half\_zero)

print(pre.reshape((6,5)))

def draw\_bin\_image(image\_matrix):

for row in image\_matrix.tolist():

print('| ' + ' '.join(' \*'[int(val)] for val in row))

def acti\_fun(x):

for i in range(x.size):

if x[i] > 0:

x[i] =1

else:

x[i] =0

return x

draw\_bin\_image(acti\_fun(np.array(half\_zero)).reshape((6,5)))

| \* \* \* \* \*

| \* \*

| \* \* \*

| \* \*

|

|

draw\_bin\_image(acti\_fun(pre).reshape((6,5)))

| \* \* \*

| \* \*

| \* \*

| \* \*

| \* \*

| \* \* \*

**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0-2的矩阵设计好存储到列表中。
2. 创建网络。
3. 数字识别测试。将训练数据加入噪声作为测试数据，输入到创建并训练好的网络中，网络的输出是与该数字点阵最为接近的目标向量。