实验报告：霍普菲尔德网络结构设计

王瑞哲　PB19071509

信息科学技术学院

摘　要：根据递归神经网络（Recursive Neural Network）设计方法，考虑最典型的霍普菲尔德网络（Hopfield Network）模型，结合Matlab程序实现了给定平衡点下的网络收敛过程，验证了离散型霍普菲尔德网络的联想记忆功能；同时借助Matlab工具箱绘图功能，直观展现了网络收敛过程，并对比了不同循环次数对网络收敛效果的影响。

关键词：递归神经网络；霍普菲尔德网络；Matlab；平衡点；联想记忆；收敛效果

递归神经网络（Recursive Neural Network，或称反馈神经网络）是在静态神经网络建模的基础上，考虑到实际应用中的被控对象通常是时变的这一因素，加入信息延时和信息反馈的神经网络系统。由于网络中存在递归信息，网络的状态是随时间变化而变化的，其运动轨迹必然存在着稳定性的问题，这就是递归网络与前向网络在网络性能分析上的最大问题之一。

霍普菲尔德网络（Hopfield Network）是递归神经网络的典型代表，也是最简单的全反馈网络。它只有一层网络，激活函数为阈值函数，将 k 时刻的网络输出反馈到对应的网络输入端，并直接作为下一时刻网络的输入，组成动态系统。按照处理输入样本的不同，Hopfield 网络可以分成两种不同的类型：离散型(DHNN)和连续型(CHNN)。前者适合于处理输入为二值逻辑的样本，主要用于联想记忆；后者适合于处理输入为模拟量的样本，主要用于分布存储。

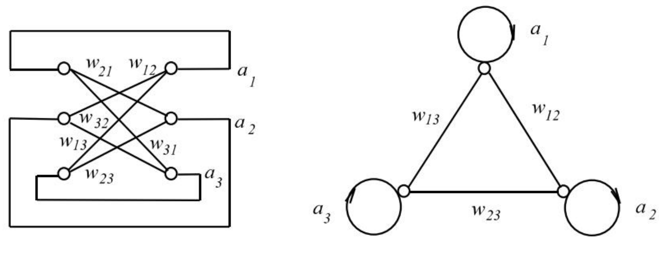
1 实验原理

1.1 全局反馈递归神经网络的平衡点作用

反馈网络（Recurrent Network），又称全局反馈递归网络，其目的是为了设计一个网络存储一组平衡点，使得当给定网络一组初始值时，网络通过自行运行而收敛到所存储的某个平衡点上。如果将反馈网络稳定的平衡状态看作一种记忆，那么网络由任一初始状态向稳态的转化过程，实质上是一种寻找记忆的过程，网络所具有的稳定平衡点是实现联想记忆的基础。

离散型霍普菲尔德网络（DHNN）的激活函数为二值型，其输入、输出为{0，1}的反馈网络，可以用于实现联想记忆功能。

1.2 DHNN 模型结构与联想记忆功能



(a)霍普菲尔德网络结构图　　(b)等价的霍普菲尔德网络图

图 1　霍普菲尔德网络图

在 DHNN 模型中，每个神经元节点的输出可以有两值状态，-1 或 1（0 或 1），其输出类似于 MP 神经元，可表示为：

$$n_j(t)=\sum_{\substack{i=1\\i\neq j}}^{r} w_{ij}a_j$$

$$a_j(t+1)=\mathrm{sgn}[n_j(t)]$$

$$=\begin{cases}1,n_j(t)\geq 0\\-1,n_j(t)<0\end{cases}$$

联想记忆功能是 DHNN 的一个重要应用范围。要想实现联想记忆，反馈网络必须具有两个基本条件：

①网络能收敛到稳定的平衡状态，并以其作为样本的记忆信息；

②具有回忆能力，能够从某一残缺的信息回忆

起所属的完整的记忆信息。

DHNN 实现联想记忆的过程分为两个阶段：学习记忆阶段和联想回忆阶段。

反馈网络有两种基本的工作方式：串行异步和并行同步方式;对于 s 个神经元的反馈网，络 DHNN 有 2^s 个状态的可能性。

1.3 正交化的权值设计方法

这一方法的基本思想和出发点是为了满足下面四个要求：

a) 保证系统在异步工作时的稳定性，即它的权值是对称的，满足, i, j = 1, 2, ..., s；

b) 保证所有要求记忆的稳定平衡点都能收敛到自己；

c) 使伪稳定点的数目尽可能的少；

d) 使稳定点的吸引力尽可能的大。

正交化设计方法的数学设计较为复杂，但与外积和法相比较，所设计出的平衡稳定点能够保证收敛到自己并且有较大的稳定域。MATLAB 工具箱中已将此设计方法写进了函数 newhop.m 中，可按如下方式调用：

```
net=newhop(T);
w=net.lw{1,1},b=net.b{1};
```

用目标矢量给出一组平衡点，由函数 newhop.m 可以设计出对应反馈网络，保证网络对给定的目标矢量的输入能收敛到稳定的平衡点。

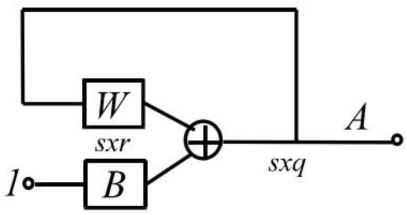


图 2　正交化方法设计的霍普菲尔德网络结构图

2 实验内容

2.1 实验要求

试设计一个反馈网络存储下列目标平衡点：

$$T=\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}$$

并用 6 组任意随机初始列矢量，包括一组在目标平衡点连线的垂直平分线上的一点作为输入矢量对所设计的网络的平衡点进行测试，观察 3 次循环的每一次的输出结果。给出最后收敛到各自平衡点（或不稳定的平衡点）结果的次数。

2.2 实验代码实现部分

首先按 1.3 部分，建立正交化设计霍普菲尔德网络模型：

```
clear;
T = [1 -1;-1 1];                %初始化目标平衡点
net = newhop(T);          %建立霍普菲尔德网络
W = net.lw{1,1};
b = net.b{1};   %W,b 分别为网络的权值和误差
N = 50;                                   %最大迭代次数
```

然后，利用 unifrnd (-1, 1, 2, 6)语句，随机生成一组 2×6（r×q）输入矢量 P。其中，r=2 代表输入网络变元维数，与所给定的目标矢量 T 相匹配；q=6 则依题意选取 6 组矢量，并依题意将其中一组改为在目标平衡点连线的垂直平分线上。最终取这样的一组 P 如下：

P = [-0.6294　0.2316　0.1460　0.9150　-0.8268　0.5000;
-0.8116　0.8480　-0.8268　0.9298　0.2460　-0.5000];

其中手动调整了最后一组在目标平衡点连线的垂直平分线上。利用语句：

```
[Y,Pf,Af] = sim(net,{6,N},[],P);
```

将 P 输入至网络中，并观察输出结果。其中 N 代表最大循环迭代次数。利用 Y{i}观察第 i 次循环后的结果，如下：

第一次循环：

A_1=[-0.0549　-0.2376　0.4892　0.1972　-0.6880　0.5809
-0.2666　0.4785　-0.6411　0.2144　0.5584　-0.5809];

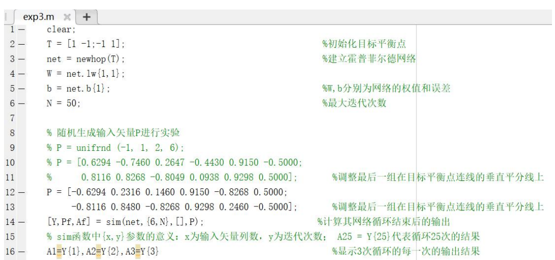
第二次循环：

A_2=[-0.0871　-0.3892　0.6396　0.0359　-0.7385　0.6749
-0.1588　0.4429　-0.6735　0.0559　0.7096　-0.6749];

第三次循环：

A_3=[-0.1349　-0.4774　0.7590 -0.0014　-0.8445　0.7842
-0.1509　0.4894　-0.7666　0.0214　0.8380　-0.7842];

最后收敛到各自的平衡点所需的次数为 17，9，6，34，6，6。此部分的实现代码如下图所示：



```
exp3.m
clear;
T = [1 -1;-1 1];                                  %初始化目标平衡点
net = newhop(T);                                  %建立霍普菲尔德网络
W = net.lw{1,1};
b = net.b{1};                                     %W,b分别为网络的权值和误差
N = 50;                                           %最大迭代次数

% 随机生成输入矢量P进行实验
% P = unifrnd (-1, 1, 2, 6);
% P = [0.6294 -0.7460 0.2647 -0.4430 0.9150 -0.5000;
%      0.8116 0.8268 -0.8049 0.0938 0.9298 0.5000];      %调整最后一组在目标平衡点连线的垂直平分线上
P = [-0.6294 0.2316 0.1460 0.9150 -0.8268 0.5000;
     -0.8116 0.8480 -0.8268 0.9298 0.2460 -0.5000];     %调整最后一组在目标平衡点连线的垂直平分线上
[Y,Pf,Af] = sim(net,{6,N},[],P);                  %计算其网络循环结束后的输出
% sim函数中{x,y}参数的意义：x为输入矢量列数，y为迭代次数； A25 = Y{25}代表循环25次的结果
A1=Y{1},A2=Y{2},A3=Y{3}                           %显示3次循环的每一次的输出结果
```

图 3　霍普菲尔德网络设计代码部分

2.3 实验结果的绘图展示

MATLAB 提供了强大的绘图指令。若能将上述实验的结果绘图展示，便可以更直观地看出霍普菲尔德网络对于输入矢量训练至网络平衡点的趋势。

在上述实验中，每次循环的结果都会被存储在 Y{i}中，每个 Y{i}都是一个 2×6 的矩阵。如果我们要考察单独某组二维坐标（如输入 P 矢量的第一列（-0.6294，-0.8116），我们可以将每一个 Y{i}中的第一列元素都提取出来并合成一个新的矩阵 Sum1，以用来表示第一组坐标在整个循环过程中的变化状况。对应的代码如下：

```
% 绘图展示        P(1,:)表示提取矩阵P的第一行，P(:,1)表示提取矩阵P的第一列
figure;
hold on
title('Hopfield Network Training State'); ylabel('Y'); xlabel('X');
Sum1=[];Sum2=[];Sum3=[];Sum4=[];Sum5=[];Sum6=[];
for i=0:N
    if i == 0
        A = P;          % 初始输入矢量也画出
    else
        A = Y{i};       % 计算第i次迭代输出
    end
    Sum1 = cat(2,Sum1,A(:,1));
    % 每次迭代输出的第一列按列合并入Sum1以完整表示第一组矢量变化情况，下同
    Sum2 = cat(2,Sum2,A(:,2));
    Sum3 = cat(2,Sum3,A(:,3));
    Sum4 = cat(2,Sum4,A(:,4));
    Sum5 = cat(2,Sum5,A(:,5));
    Sum6 = cat(2,Sum6,A(:,6));
end
```

图 4　绘图展示代码部分一

然后将得到的六组输入矢量以散点图方式绘制在图像上。采用 plot 函数，并详细设置了其参数以美化图像使显示更为直观清晰。代码部分为：

```
plot(Sum1(1,:),Sum1(2,:),'r-o','LineWidth',1,'MarkerEdgeColor','k','MarkerSize',6,'MarkerFaceColor','red');
plot(Sum2(1,:),Sum2(2,:),'y-o','LineWidth',1,'MarkerEdgeColor','k','MarkerSize',6,'MarkerFaceColor','yellow');
plot(Sum3(1,:),Sum3(2,:),'b-o','LineWidth',1,'MarkerEdgeColor','k','MarkerSize',6,'MarkerFaceColor','blue');
plot(Sum4(1,:),Sum4(2,:),'g-o','LineWidth',1,'MarkerEdgeColor','k','MarkerSize',6,'MarkerFaceColor','green');
plot(Sum5(1,:),Sum5(2,:),'m-o','LineWidth',1,'MarkerEdgeColor','k','MarkerSize',6,'MarkerFaceColor','magenta');
plot(Sum6(1,:),Sum6(2,:),'c-o','LineWidth',1,'MarkerEdgeColor','k','MarkerSize',6,'MarkerFaceColor','cyan');
legend('P1','P2','P3','P4','P5','P6','Location','Best');
```

图 5　绘图展示代码部分二

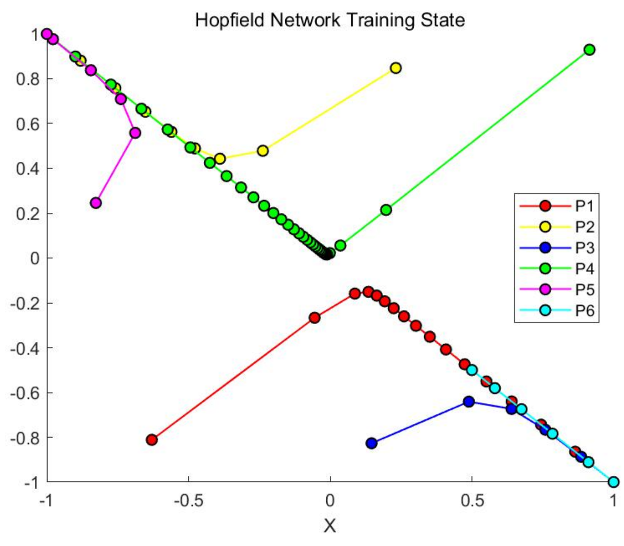


图 6　50 次训练迭代下训练效果图

根据实验结果，P1~P6 这六组坐标收敛到各自的平衡点所需的次数分别为 17，9，6，34，6，6。在图像上也可以看出，代表 P4 的绿色线条在原点附近有多点密集分布，而这恰恰是该网络的一个不稳定平衡点。在实验中所设置的 N=50 的情况下，网络可以通过足够多次循环使得这一点收敛至稳定平衡点；但若将 N 设置得更小，如若设置 N=20，则结果变为：

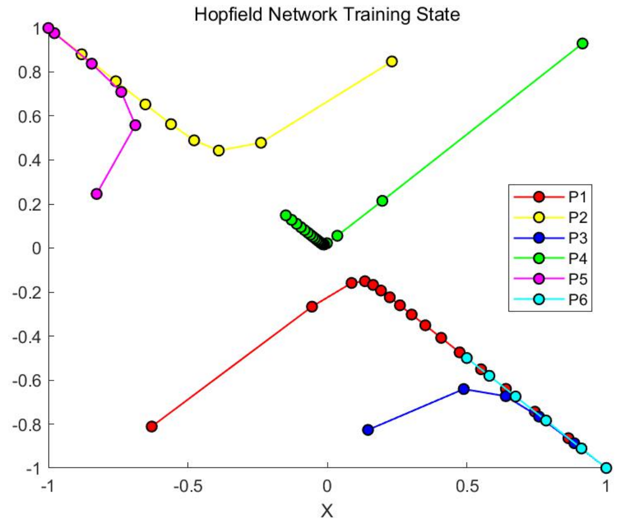


图 7　20 次训练迭代下训练效果图

显然代表 P4 这组坐标的绿色线条并未收敛至我们所希望的稳定平衡点，而是收敛在不稳定平衡点（0，0）附近。根据两次实验效果对比可以看出，循环次数对最终收敛效果有一定影响。通过查阅课本，一般情况下，此网络对任意初始随机值，运行 60 次左右均能够收敛到网络所储存的某个平衡点上。

3 实验分析及心得体会

本次实验根据递归神经网络设计方法，考虑最典型的 Hopfield 网络模型，结合 Matlab 程序实现了给定平衡点下的网络收敛过程，验证了离散型霍普菲尔德网络的联想记忆功能。同时借助 Matlab 工具箱绘图功能，直观展现了网络收敛过程，并对比了不同循环次数对网络收敛效果的影响。

通过本次实验，我对 Hopfield 网络原理有了更深层次的认识，进一步加深了对于 Hopfield 网络设计和训练方法的理解，体会了全局反馈神经网络的平衡点作用，进而感受到了离散霍普菲尔德网络的联想记忆功能，并掌握离散霍普菲尔德网络设计的一般方法；同时进一步掌握了 matlab 软件的绘图功

能，通过需求将网络训练结果进行合理的数学变形，最终能在散点图上得到清晰的训练进程的展示，清晰而直观；进而在这一过程中体会到了训练次数对于网络训练效果的影响，并通过绘制多张图像，展示并分析出这些区别所在。

参考文献：

[1] 丛爽. 面向 MATLAB 工具箱的神经网络理论与应用（第三版）[M]. 合肥：中国科学技术大学出版社，2009.4: 293-305.