神经网络方法

1. 多层前向神经网络原理介绍

多层前向神经网络(MLP)是神经网络中的一种，它由一些最基本的神经元即节点组成，下面图 1 就是这样一个网络。这种网络的结构如下：网络由分为不同层次的节点集合组成，每一层的节点输出到下一层节点，这些输出值由于连接不同而被放大、衰减或抑制。除了输入层外，每一节点的输入为前一层所有节点输出值的和。每一节点的激励输出值由节点输入、激励函数及偏置量决定。

如图 1 所示，输入模式的各分量作为第 i 层各节点的输入，这一节点的输出，或者完全等于它们的输入值，或由该层进行归一化处理，使该层的输出值都在+1 或-1 之间。

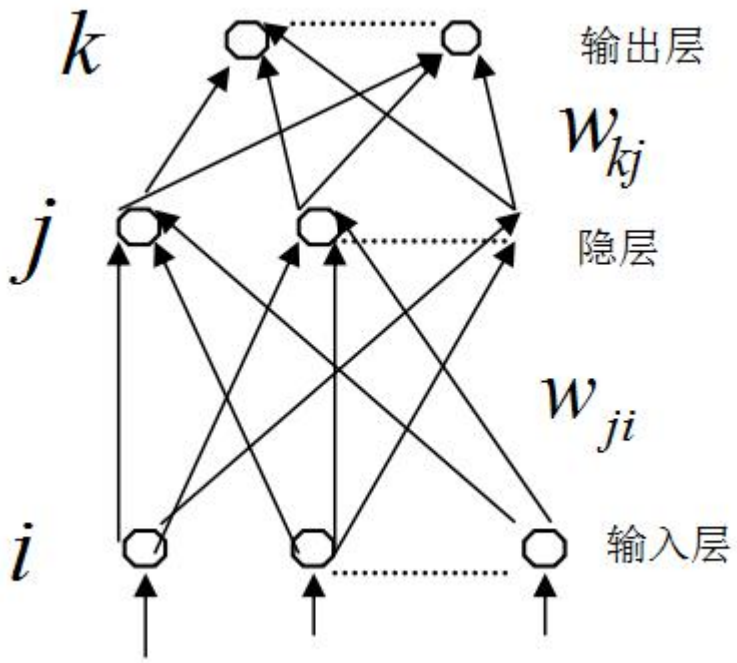

图 1　多层前向神经网络

在第 j 层，节点的输入值为：

$$net_j = \sum w_{ji} o_i + \theta_j \tag{1}$$

式中 θ_j 为阈值，正阀值的作用将激励函数沿 x 轴向左平移.

节点的输出值为：

$$o_j = f(net_j) \tag{2}$$

式中 f 为节点的激励函数，通常选择如下 Sigmoid 函数:

$$f(x) = \frac{1}{1+\exp(-x)} \tag{3}$$

在第k层的网络节点输入为：

$$net_k = \sum w_{kj} o_j + \theta_k \tag{4}$$

而输出为：

$$o_k = f(net_k) \tag{5}$$

在网络学习阶段，网络输入为模式样本$x_p = \{x_{pi}\}$，网络要修正自己的权值及各节点的阀值，使网络输出不断接近期望值t_{pk}，每做一次调整后，换一对输入与期望输出，再做一次调整，直到满足所有样本的输入与输出间的对应。对每一个输入的模式样本，平方误差E_p为：

$$E_p = \frac{1}{2}\sum_k (t_{pk} - o_{pk})^2 \tag{6}$$

而对于全部学习样本，系统的总误差为：

$$E = \frac{1}{2p}\sum_p \sum_k (t_{pk} - o_{pk})^2 \tag{7}$$

在学习过程中，系统将调整连接权和阀值，使E_p尽可能快地下降。

2 Matlab相关函数介绍

2.1)网络初始化函数

net=newff([x_m,x_M],[h_1,h_2,$\cdots$,h_k],{f_1,f_2,$\cdots$,f_k})

其中，x_m和x_M分别为列向量,存储各个样本输入数据的最小值和最大值；第2个输入变量是一个行向量，输入各层节点数；第3个输入变量是字符串，代表该层的传输函数。常用tansig和logsig函数。其中

$$\text{tansig(x)} = \frac{1-e^{-2x}}{1+e^{-2x}}, \text{logsig(x)} = \frac{1}{1+e^{-x}}$$

除了上面方法给网络赋值外，还可以用下面格式设定参数。

Net.trainParam.epochs=1000　设定迭代次数

Net.trainFcn=’traingm’　　设定带动量的梯度下降算法

2.2)网络训练函数

[net,tr,Y1,E]=train(net,X,Y)

其中 X 为$n \times M$ 矩阵，n 为输入变量的个数，M 为样本数,；Y 为$m \times M$ 矩阵，m 为输出变量的个数。X，Y 分别存储样本的输入输出数据。net 为返回后的神经网络对象，tr 为训练跟踪数据，tr.perf 为各步目标函数值。Y1 为网络的最后输出，E1 为训练误差向量。

2.3)网络泛化函数

Y2=sim(net,X1)

其中 X1 为输入数据矩阵，各列为样本数据。Y2 为对应输出值。

3.神经网络实验

3.1 函数仿真实验

产生下列函数在[0,10]区间上间隔为 0.5 的数据，然后用神经网络进行学习。

并推广到[0,10]上间隔为 0.1 上各点的函数值。并分别作出图形。

$$y = 0.2e^{-0.2x} + 0.5 * e^{-0.15x}.\sin(1.25x) \quad 0 \le x \le 10$$

Matlab 程序：

```
x=0:0.5:10;
y=0.2*exp(-0.2*x)+0.5*exp(-0.15*x).*sin(1.25*x);

plot(x,y)  %画原始数据图

net.trainParam.epochs=5000;  %设定迭代步数
net=newff([0,10],[6,1],{'tansig','tansig'}); %初始化网络

net=train(net,x,y);  %进行网络训练

x1=0:0.1:10;
y1=sim(net,x1);  %数据泛化

plot(x,y,'*',x1,y1,'r');  %作对比图
```

从图形上看，神经网络输出的值比原始数据的曲线光滑。说明神经网络对该函数的学习效果很好。

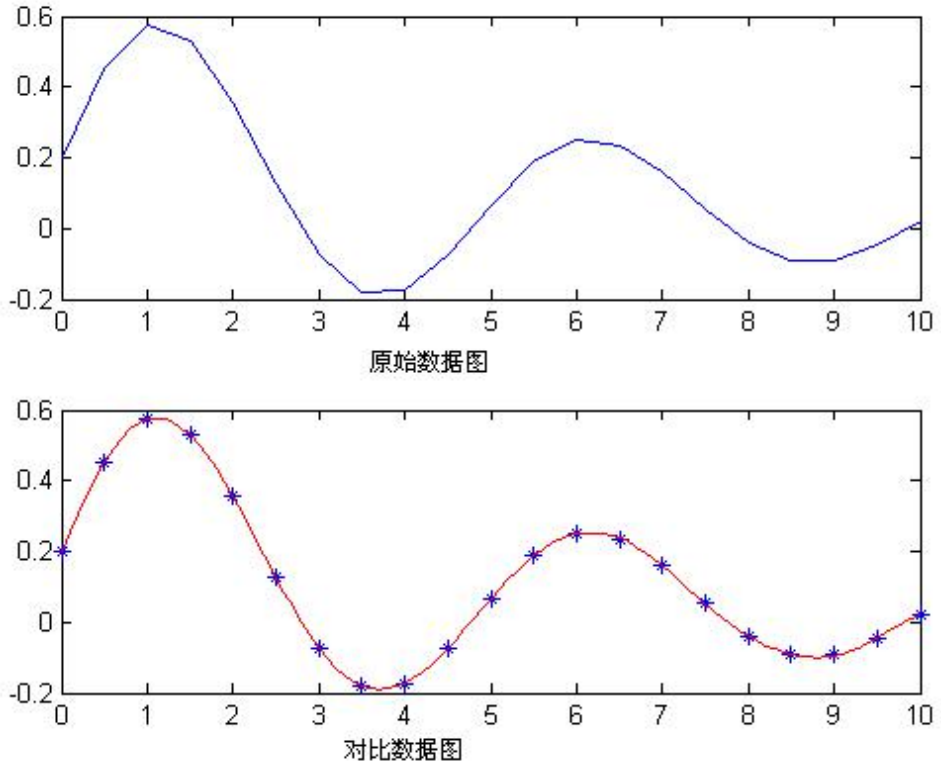

图 2　原始与对比数据图

实验 2　MCM89A 蠓的分类

有两种蠓 Af 和 Apf。根据它们的触角(mm)和翼长(mm)进行区分。现有 9 只 Af 和 6 只 Apf。样本数据如下：

表 2.19　9 只 Af 的触角和翼长

触角	1.24	1.36	1.38	1.38	1.38	1.40	1.48	1.54	1.56
翼长	1.72	1.74	1.64	1.82	1.90	1.70	1.82	1.82	2.08

表 2.20　6 只 Apf 的触角和翼长

触角	1.14	1.18	1.20	1.26	1.28	1.30
翼长	1.78	1.96	1.86	2.0	2.0	1.96

另有 3 只待判的蠓,触角和翼长数据为：(1.24,1.80),(1.28,1.84),(1.40,2.04)。试对它们进行判断。

这里我们可用三层神经网络进行判别。

输入为 15 个二维向量，输出也为 15 个二维向量。其中 Af 对应的目标向量为(1,0)，Apf 对应的目标向量为(0,1)。

Matlab 程序：

```
x=[1.24,1.36,1.38,1.38,1.38,1.40,1.48,1.54,1.56,1.14,1.18,1.20,1.26,1.28,1.30;
   1.72,1.74,1.64,1.82,1.90,1.70,1.82,1.82,2.08,1.78,1.96,1.86,2.0,  2.0,1.96];
y=[1,1,1,1,1,1,1,1,1,0,0,0,0,0,0;
   0,0,0,0,0,0,0,0,0,1,1,1,1,1,1];
   xmin=min(x');  %求各指标最小值
  xmax=max(x');  %求各指标最大值
```

```
net.trainParam.epochs=2500;  %设定迭代步数
net=newff([xmin',xmax'],[5,2],{'logsig','logsig'}); %初始化网络
net=train(net,x,y);  %进行网络训练
x1=[1.24,1.28,1.40;
    1.80,1.84,2.04];%待分样本
y1=sim(net,x1);  %数据泛化

plot(x(1,1:9),x(2,1:9),'*',x(1,10:15),x(2,10:15),'o',x1(1,:),x1(2,:),'p')  %画原始数据图
grid on
```

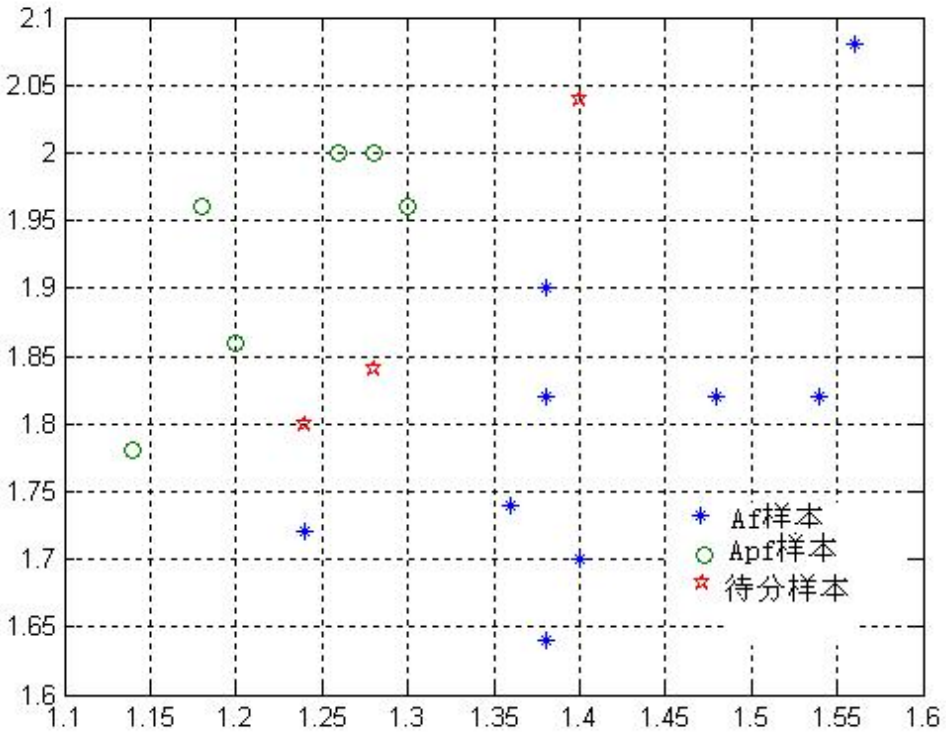

图 3　Af，Apf 及待分样本数据图

三个样本输出值为：

y1=0.1235　0.8995　0.0037

0.8785　0.0951　0.9986

以两个分量越靠近 1 就判断为哪一类。从该结果看，第二个样本分为 Af；而第一和第三个样本分为 Apf。但由于每次训练初始参数的随机性，另外待判 3 个样本在两类的临界区，导致不同的训练结果会有差异。这也正常。