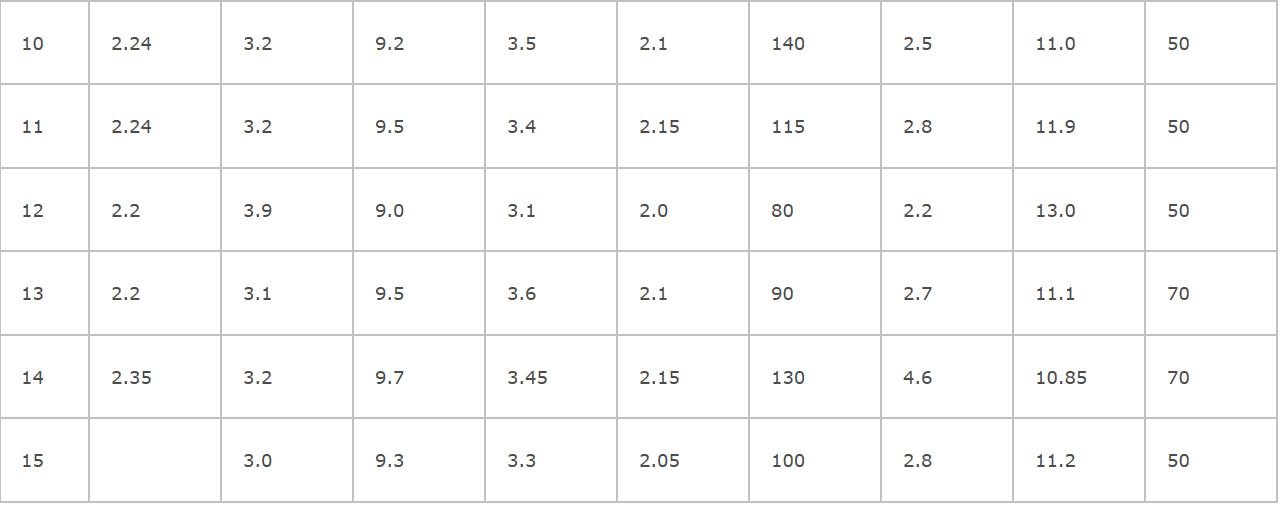
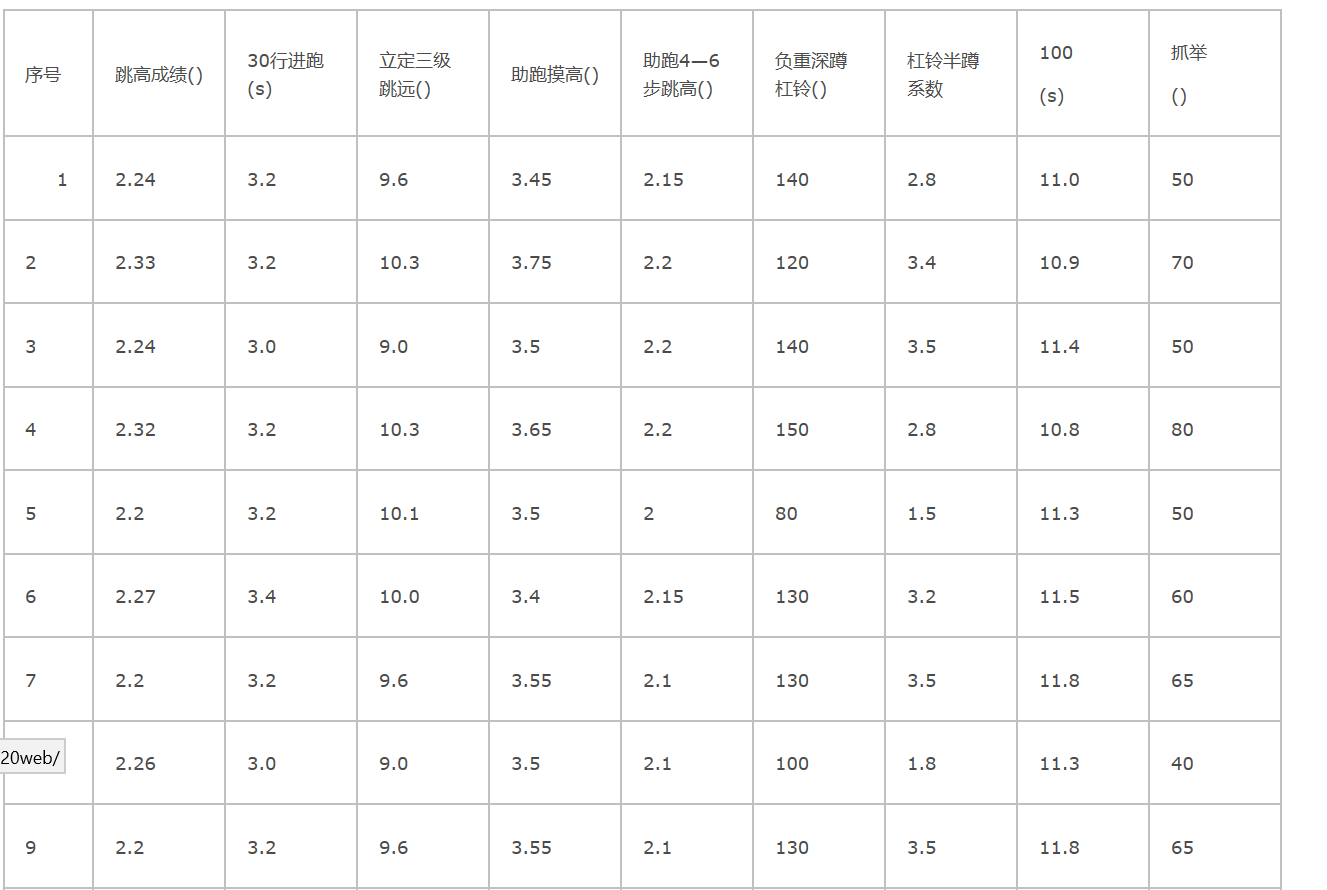
BP神经网络预测算法及实现过程

1.具体应用实例。根据表2，预测序号15的跳高成绩。

表2 国内男子跳高运动员各项素质指标



4.4 (序号15)跳高成绩预测

4.4.1 数据整理

1）我们将前14组国内男子跳高运动员各项素质指标作为输入，即（30m行进跑，立定三级跳远，助跑摸高，助跑4-6步跳高，负重深蹲杠铃，杠铃半蹲系数，100m，抓举），将对应的跳高成绩作为输出。并用matlab自带的premnmx()函数将这些数据归一化处理。

数据集：（注意：每一列是一组输入训练集，行数代表输入层神经元个数，列数输入训练集组数）

P=[3.2 3.2 3 3.2 3.2 3.4 3.2 3 3.2 3.2 3.2 3.9 3.1 3.2;

9.6 10.3 9 10.3 10.1 10 9.6 9 9.6 9.2 9.5 9 9.5 9.7;

3.45 3.75 3.5 3.65 3.5 3.4 3.55 3.5 3.55 3.5 3.4 3.1 3.6 3.45;

2.15 2.2 2.2 2.2 2 2.15 2.14 2.1 2.1 2.1 2.15 2 2.1 2.15;

140 120 140 150 80 130 130 100 130 140 115 80 90 130;

2.8 3.4 3.5 2.8 1.5 3.2 3.5 1.8 3.5 2.5 2.8 2.2 2.7 4.6;

11 10.9 11.4 10.8 11.3 11.5 11.8 11.3 11.8 11 11.9 13 11.1 10.85;

50 70 50 80 50 60 65 40 65 50 50 50 70 70];

T=[2.24 2.33 2.24 2.32 2.2 2.27 2.2 2.26 2.2 2.24 2.24 2.2 2.2 2.35];

4.4.2 模型建立

4.4.2.1 BP网络模型

BP网络（Back-ProPagation Network）又称反向传播神经网络， 通过样本数据的训练，不断修正网络权值和阈值使误差函数沿负梯度方向下降，逼近期望输出。它是一种应用较为广泛的神经网络模型，多用于函数逼近、模型识别分类、数据压缩和时间序列预测等。

BP网络由输入层、隐层和输出层组成，隐层可以有一层或多层，图2是m×k×n的三层BP网络模型，网络选用S型传递函数， 通过反传误差函数 （ （Ti为期望输出、Oi为网络的计算输出），不断调节网络权值和阈值使误差函数E达到极小。

BP网络具有高度非线性和较强的泛化能力，但也存在收敛速度慢、迭代步数多、易于陷入局部极小和全局搜索能力差等缺点。可以先用遗传算法对“BP网络”进行优化在解析空间找出较好的搜索空间，再用BP网络在较小的搜索空间内搜索最优解。

4.4.2.2 模型求解

4.4.2.2.1 网络结构设计

1) 输入输出层的设计

该模型由每组数据的各项素质指标作为输入，以跳高成绩作为输出，所以输入层的节点数为8，输出层的节点数为1。

2) 隐层设计

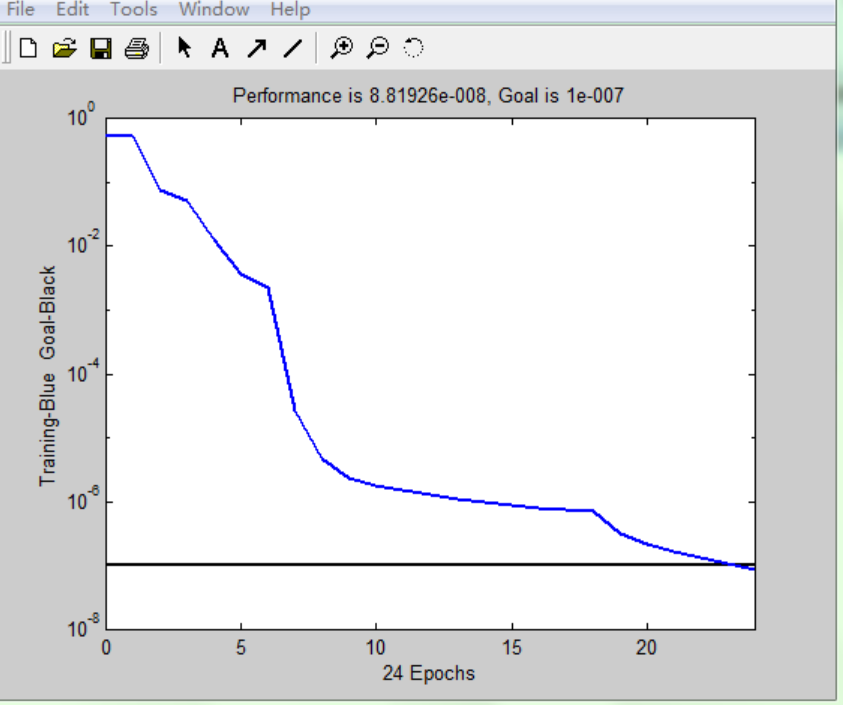
有关研究表明, 有一个隐层的神经网络, 只要隐节点足够多, 就可以以任意精度逼近一个非线性函数。因此, 本文采用含有一个隐层的三层多输入单输出的BP网络建立预测模型。在网络设计过程中, 隐层神经元数的确定十分重要。隐层神经元个数过多, 会加大网络计算量并容易产生过度拟合问题; 神经元个数过少, 则会影响网络性能, 达不到预期效果。网络中隐层神经元的数目与实际问题的复杂程度、输入和输出层的神经元数以及对期望误差的设定有着直接的联系。目前, 对于隐层中神经元数目的确定并没有明确的公式, 只有一些经验公式, 神经元个数的最终确定还是需要根据经验和多次实验来确定。本文在选取隐层神经元个数的问题上参照了以下的经验公式:

其中, n为输入层神经元个数, m 为输出层神经元个数, a 为[ 1, 10]之间的常数。

根据上式可以计算出神经元个数为4-13个之间，在本次实验中选择隐层神经元个数为6.

4.4.2.2.3 模型的实现

此次预测选用MATLAB中的神经网络工具箱进行网络的训练, 预测模型的具体实现步骤如下:

将训练样本数据归一化后输入网络, 设定网络隐层和输出层激励函数分别为tansig和logsig函数, 网络训练函数为traingdx, 网络性能函数为mse,隐层神经元数初设为6。设定网络参数。网络迭代次数epochs为5000次, 期望误差goal为0.00000001, 学习速率lr为0. 01。设定完参数后, 开始训练网络。

matlab代码：

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

?P=[3.2 3.2 3 3.2 3.2 3.4 3.2 3 3.2 3.2 3.2 3.9 3.1 3.2;

9.6 10.3 9 10.3 10.1 10 9.6 9 9.6 9.2 9.5 9 9.5 9.7;

3.45 3.75 3.5 3.65 3.5 3.4 3.55 3.5 3.55 3.5 3.4 3.1 3.6 3.45;

2.15 2.2 2.2 2.2 2 2.15 2.14 2.1 2.1 2.1 2.15 2 2.1 2.15;

140 120 140 150 80 130 130 100 130 140 115 80 90 130;

2.8 3.4 3.5 2.8 1.5 3.2 3.5 1.8 3.5 2.5 2.8 2.2 2.7 4.6;

11 10.9 11.4 10.8 11.3 11.5 11.8 11.3 11.8 11 11.9 13 11.1 10.85;

50 70 50 80 50 60 65 40 65 50 50 50 70 70];

?T=[2.24 2.33 2.24 2.32 2.2 2.27 2.2 2.26 2.2 2.24 2.24 2.2 2.2 2.35];

?[p1,minp,maxp,t1,mint,maxt]=premnmx(P,T);

?%创建网络

?net=newff(minmax(P),[8,6,1],{'tansig','tansig','purelin'},'trainlm');

?%设置训练次数

?net.trainParam.epochs = 5000;

?%设置收敛误差

?net.trainParam.goal=0.0000001;

?%训练网络

?[net,tr]=train(net,p1,t1);

TRAINLM, Epoch 0/5000, MSE 0.533351/1e-007, Gradient 18.9079/1e-010

TRAINLM, Epoch 24/5000, MSE 8.81926e-008/1e-007, Gradient 0.0022922/1e-010

TRAINLM, Performance goal met.

?%输入数据

?a=[3.0;9.3;3.3;2.05;100;2.8;11.2;50];

?%将输入数据归一化

?a=premnmx(a);

?%放入到网络输出数据

?b=sim(net,a);

?%将得到的数据反归一化得到预测数据

?c=postmnmx(b,mint,maxt);

?c

c =

2.2003