6.5 神经网络实现

6.5.1 数据预处理

- 在训练神经网络前一般需要对数据进行预处理，一种重要的预处理手段是归一化处理。
- 下面简要介绍归一化处理的原理与方法。

- (1) 什么是归一化?
- 数据归一化，就是将数据映射到[0,1]或[-1,1]区间或更小的区间，比如(0.1,0.9) 。

6.5.1 数据预处理

- (2) 为什么要归一化处理?
  - <1>输入数据的单位不一样，有些数据的范围可能特别大，导致的结果是神经网络收敛慢、训练时间长。
  - <2>数据范围大的输入在模式分类中的作用可能会偏大，而数据范围小的输入作用就可能会偏小。
  - <3>由于神经网络输出层的激活函数的值域是有限制的，因此需要将网络训练的目标数据映射到激活函数的值域。
  - 例如神经网络的输出层若采用S形激活函数，由于S形函数的值域限制在(0,1)，也就是说神经网络的输出只能限制在(0,1)，所以训练数据的输出就要归一化到[0,1]区间。
  - <4>S形激活函数在(0,1)区间以外区域很平缓，区分度太小。例如S形函数f(X)在参数a=1时，f(100)与f(5)只相差0.0067。

6.5.2 常见的数据归一化方法

（1）十进制缩放（Decimal scaling）

$$(1) newValue = \frac{originalValue}{oldMax}$$

（2）min-max标准化（Min-Max normalization）

$$(2) newValue = \frac{originalValue - oldMin}{oldMax - oldMin}$$

（3）对数标准化（Logarithmic normalization）

$$(3) newValue = \frac{Log_2(originalValue)}{Log_2(oldMax)}$$

6.5.2 常见的数据归一化方法

一种简单而快速的归一化算法是线性转换算法。

线性转换算法常见有两种形式:

<1> y = (x - min)/(max - min)

其中min为x的最小值，max为x的最大值，输入向量为x，归一化后的输出向量为y 。上式将数据归一化到 [0 , 1]区间，当激活函数采用S形函数时（值域为(0,1)）时这条式子适用。

<2> y = 2 * (x - min) / (max - min) – 1

这条公式将数据归一化到 [-1 , 1] 区间。当激活函数采用双极S形函数（值域为(-1,1)）时这条式子适用。

6.5.3 神经网络的输入和输出数据格式

1、神经网络输入格式

- 落在[0，1]闭区间内的数值类型。
- 实际应用中，需要将分类数据变换为[0,1]区间的数值数据。
- 两种方法——
  - 方法一：将[0,1]区间分为大小相等的间隔，将间隔点上的取值作为分类类型数据的数值表示。
  - 方法二：对输入数据进行二进制编码，增加输入结点，用两个或多个输入节点表示一个输入属性。

【例6.1】

某投资公司的客户数据集中“账户类型”属性为分类类型属性，它有四种取值，分别为“基本账户”、“一般账户”、“临时账户”和“专用账户”。若将“账户类型”属性作为神经网络的输入数据，就必须进行数据变换，使之成为[0,1]区间的数值数据。

变换

- 目标——对“账户类型”属性进行落在[0,1]区间的分类-数值变换。
  方法——使用方法一和方法二进行数据变换。
- 结果——如表6.1所示

序号	分类类型属性值	[0,1]区间数值型属性值（方法一）	[0,1]区间数值型属性值（方法二）
1	基本账户	0	[0,0]
2	一般账户	0.33	[0,1]
3	临时账户	0.67	[1,0]
4	专用账户	1	[1,1]

表6.1 “账户类型”属性的分类-数值变换

6.5.3 神经网络的输入和输出数据格式

2、神经网络输出格式

- 神经网络的输出结点表示为[0，1]区间内的连续值。
- 如果神经网络是分类模型，需要对输出进行变换。

【例6.2】

训练神经网络建立分类模型，能够识别购买BMW5的顾客性别是“男”还是“女”。

神经网络分类模型的输出设置

- 目标——建立输出为性别值的神经网络分类模型识别顾客的性别。
- 方法——

  1）设计有一个输出层节点的体系结构，设置1和0分别为男顾客和女顾客的理想输出。在不能清晰分类的情况下，使用检验集数据来帮助；

  2）设计有两个输出层节点的体系结构：节点1和节点2。男性和女性顾客的正确输出组合分别设置为[1,0]和[0,1]。不能清晰分类的情况下，使用检验集数据来帮助。
- 问题解决——当未知实例x给出一个不确定的输出值v时，使用在v处或附近聚类的大多数检验集实例所属的类别来分类x。

【例6.3】

一个采用min-max标准化用于房屋估价的神经网络已经训练成功，该网络的输出数据为0.18，需要根据该值还原房屋的真正的预估价格（房屋价格范围限定在100到1000（单位：万元）之间）。

计算

- 问题——根据[0,1]区间内的神经网络输出的房屋预估价格和房屋原始价格区间，计算房屋真正的预估价格。
- 解决方法——进行[0,1]区间数据归一化变换的逆变换。

$$newValue = \frac{originalValue - oldMin}{oldMax - oldMin}$$

$$originalValue = newValue(oldMax - oldMin) + oldMin$$

- 结果——0.18*（1000-100）+100=262（万元）

6.5.4 使用Matlab实现神经网络

- 使用Matlab建立前馈神经网络主要会使用到下面3个函数:
- newff： 前馈网络创建函数
- train: 训练一个神经网络
- sim： 使用网络进行仿真

6.5.4 使用Matlab实现神经网络

- (1) newff函数
- <1>newff函数语法
  - newff函数有很多可选参数，这里介绍一种简单的形式。
- 语法： net = newff (A, B, {C} ,‘trainFun’)
- 参数：
  - A：一个n×2的矩阵，第i行元素为输入信号xi的最小值和最大值；
  - B：一个k维行向量，其元素为网络中各层节点数；
  - C：一个k维字符串行向量，每一分量为对应层神经元的激活函数；
  - trainFun： 为学习规则采用的训练算法。

6.5.4 使用Matlab实现神经网络

- <2>常用的激活函数
- 常用的激活函数有:
- a) 线性函数 (Linear transfer function)
- f(x) = x
- 该函数的字符串为’purelin’。

6.5.4 使用Matlab实现神经网络

- b) 对数S形转移函数

$$f(x) = \frac{1}{1 + e^{-x}} \quad (0 < f(x) < 1)$$

- 该函数的字符串为 logsig 。

- c) 双曲正切S形函数

- 也就是上面所
$$f(x) = \frac{2}{1 + e^{-2n}} - 1 \quad (-1 < f(x) < 1)$$
- 该函数的字符串为 tansig 。

6.5.4 使用Matlab实现神经网络

- <3>常见的训练函数
- 常见的训练函数有:
- traingd： 梯度下降BP训练函数(Gradient descent backpropagation)
- traingdx： 梯度下降自适应学习率训练函数

6.5.4 使用Matlab实现神经网络

- <4>网络配置参数
- 一些重要的网络配置参数如下:
- net.trainparam.goal ： 神经网络训练的目标误差
- net.trainparam.show ： 显示中间结果的周期
- net.trainparam.epochs ： 最大迭代次数
- net.trainParam.lr ： 学习率

6.5.4 使用Matlab实现神经网络

- (2) train函数
- 网络训练学习函数。
- 语法：[net, tr, Y1, E] = train(net, X, Y)
- 参数：
  - X：网络实际输入
  - Y：网络应有输出
  - tr：训练跟踪信息
  - Y1：网络实际输出
  - E：误差矩阵

6.5.4 使用Matlab实现神经网络

- (3) sim函数
- 语法：Y=sim(net,X)
- 参数：
  - net：网络
  - X：输入给网络的Ｋ×N矩阵，其中K为网络输入个数，N为数据样本数
  - Y：输出矩阵Q×N，其中Q为网络输出个数

旧版语法-Matlab R2007b前

```
clear all
P = [-1 -1 2 2; 0 5 0 5];      %输入数据
T = [-1 -1 2 2];               %输出数据
```

有几个自变量？共几个样本数据？

X1	-1	-1	2	2
X2	0	5	0	5
Y	-1	-1	2	2

P=P’;
T=T’;

转置

X1	X2	Y
-1	0	-1
-1	5	-1
2	0	2
2	5	2

旧版语法-Matlab R2007b前

```
%构建网络
% 隐含层5个节点，第一层的传递函数是tansig，输出层的传递函数是
   purelin。训练函数是traingd。都取默认值。
net1 = newff(minmax(P),[5 1],{'tansig', 'purelin'}, 'traingd');
net1.trainParam.goal = 1e-5;          %学习目标及训练精度
net1.trainParam.epochs = 300;         %训练次数
net1.trainParam.lr = 0.05;            %学习速率
net1.trainParam.showWindow = 1;   %打开训练窗口
```

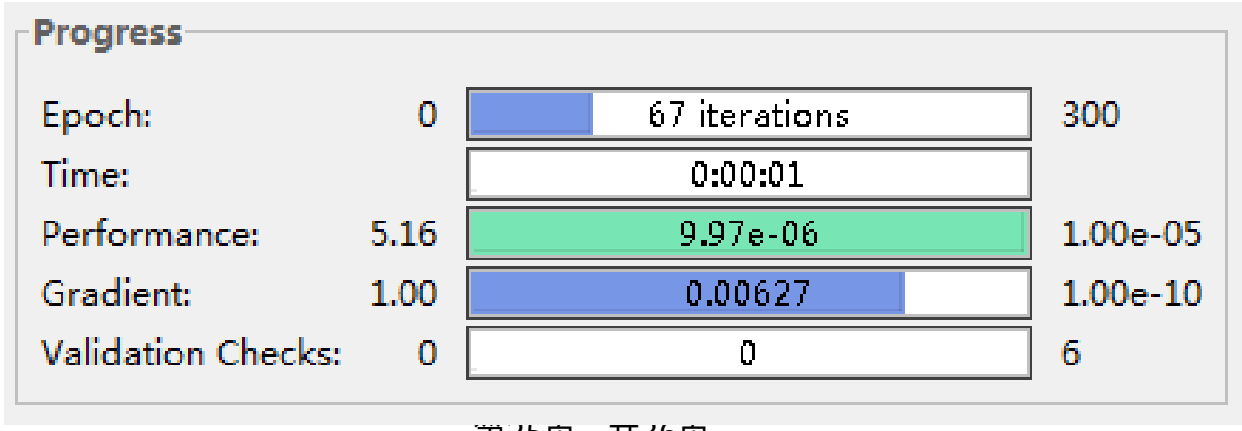
旧版语法-Matlab R2007b前

```
%网络训练
net1= train(net1,P,T);
%网络预测
%训练集作测试集，称为样本内预测
Y1 = sim(net1,P);


%结果分析
disp(['旧式语法 mse: ' num                    ))]);
```

旧版语法-Matlab R2007b前

- 达到了设定的1e-5的目标。不过也收到了警告，建议采用新的参数列表。

```
Warning: NEWFF used in an obsolete way.
> In nntobsu at 18
  In newff at 86
  In BPold at 9
          See help for NEWFF to update calls to the new argument list.

旧式语法 mse: 9.9715e-006
```



新式语法-Matlab R2007b

```
clear all
P = [-1 -1 2 2; 0 5 0 5]; %输入数据
T = [-1 -1 2 2];              %输出数据
%构建网络
%不用求minmax，也不用指定输出层神元数了（newff会根据参数T自行推导）
net2 = newff(P,T,5,{'tansig', 'purelin'}, 'traingd'); % 隐含层5个神经元
net2.trainParam.goal = 1e-5;
net2.trainParam.epochs = 300;
net2.trainParam.lr = 0.05;
net2.trainParam.showWindow = 1;
```

新式语法-Matlab R2007b

```
%网络训练
net2 = train(net2,P,T);
%网络预测
Y2 = sim(net2,P);
%结果分析
disp(['新式语法 mse: ' num2str(mse(T-Y2))]);
```

新式语法-Matlab R2007b

程序输出：

新式语法 mse: 6.3571

可见，远远没有达到1e-5的目标。这是为什么呢？

newff.m分成三大块：主程序、新版实现子函数 new_5p1()、旧版实现子函数 new_5p0()。比较新旧这两个子函数，发现新版设置了 net.divideFcn 属性，其值为'dividerand'。该函数把样本数据三分为训练集、验证集和测试集，默认比例是6:2:2。于是在我的程序中清除该属性再训练：

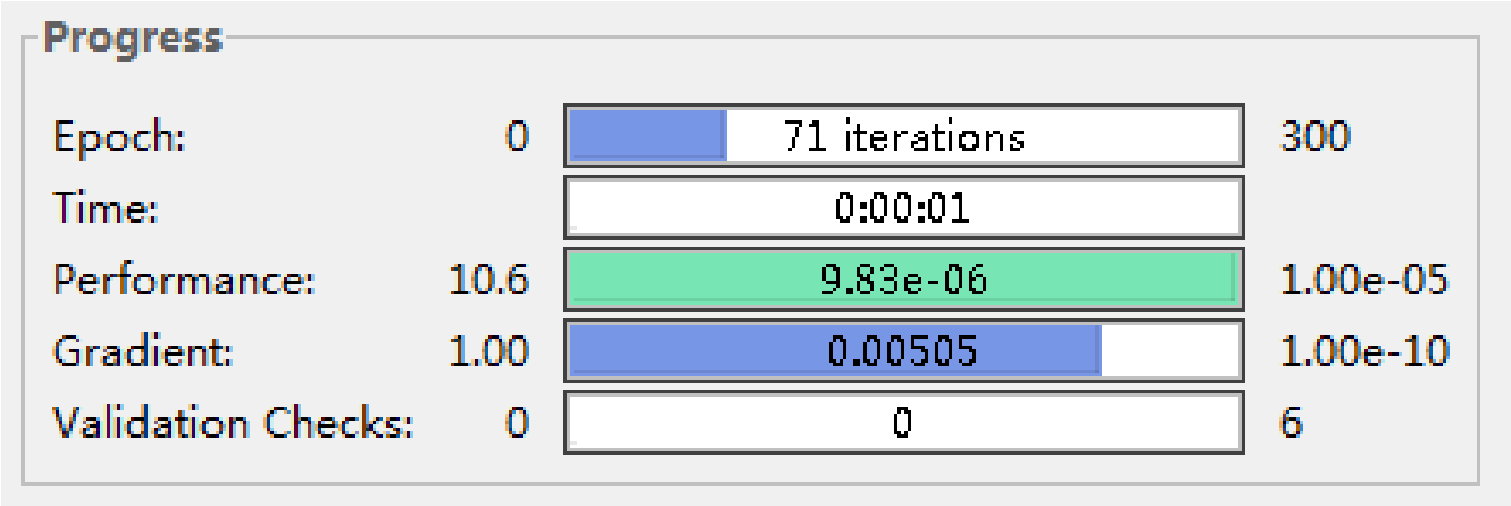
```
% 对于样本极少的情况，不要再三分了
net2.divideFcn = '';
```

新式语法-Matlab R2007b

程序输出：

新式语法，改进 mse: 9.8329e-006

达到了设定的1e-5的目标。



参数设置对神经网络性能的影响

- 在实验中通过调整隐含层节点数，选择不通过的激活函数，设定不同的学习率，可以影响网络性能。
- <1>隐含层节点个数
  - 隐含层节点的个数对于识别率的影响并不大，但是节点个数过多会增加运算量，使得训练较慢。
- <2>激活函数的选择
  - 激活函数无论对于识别率或收敛速度都有显著的影响。在逼近高次曲线时，S形函数精度比线性函数要高得多，但计算量也要大得多。
- <3>学习率的选择
  - 学习率影响着网络收敛的速度，以及网络能否收敛。学习率设置偏小可以保证网络收敛，但是收敛较慢。相反，学习率设置偏大则有可能使网络训练不收敛，影响识别效果。

【例3.6】

将XOR逻辑运算规则表看作由两个运算数为输入属性、运算结果为输出属性的数据集，输出为两个类，一个类的分类值等于1，该类有两个实例；另一个类的分类值等于0，该类也有两个实例。建立BP神经网络。

异或（*exclusive*，XOR）逻辑运算

Operand1	Operand 2	XOR
1	1	0
0	1	1
1	0	1
0	0	0

表3.4 XOR逻辑运算

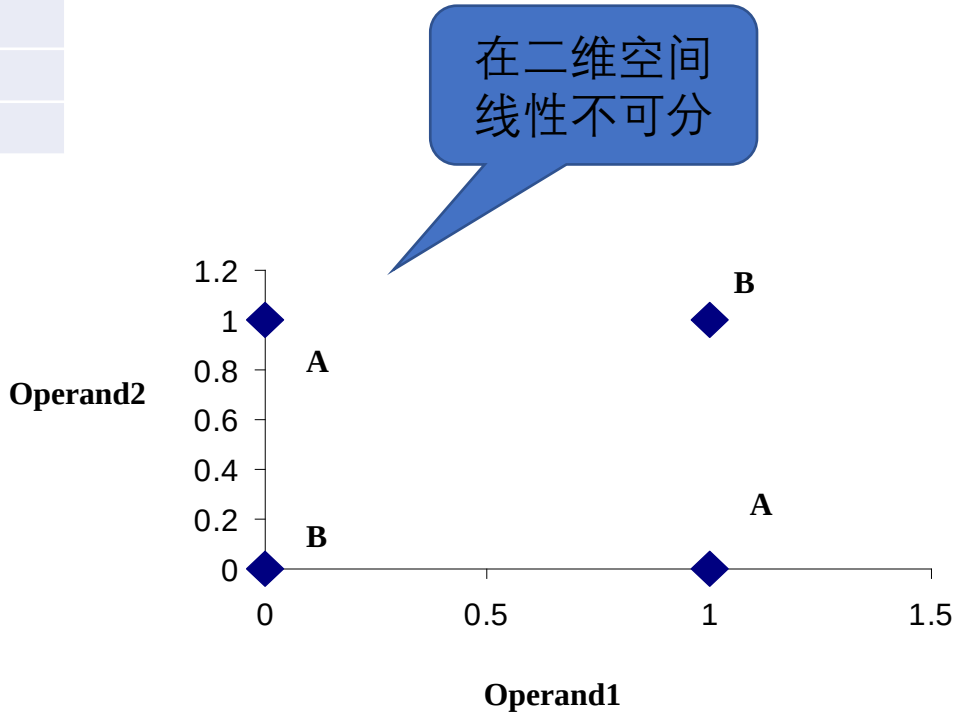


图3.5 XOR函数图

实验步骤

- 步骤1：准备训练数据
- 步骤2：定义网络体系结构，设置相关参数
- 步骤3：训练网络
- 步骤4：解释训练结果
- 步骤5：结果不理想，更改结构，调整参数，重复实验