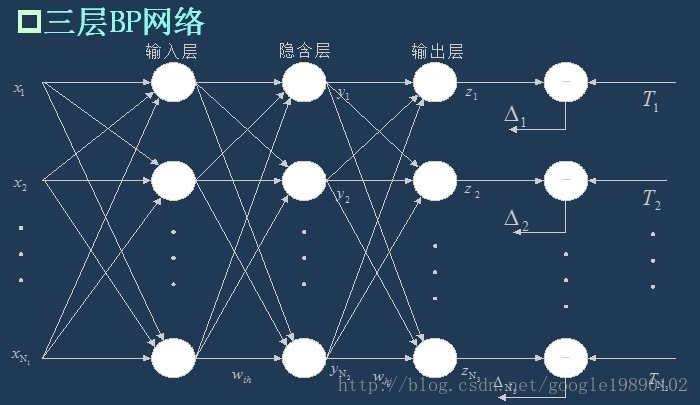
**[简单易学的机器学习算法——神经网络之BP神经网络](https://blog.csdn.net/google19890102/article/details/32723459)**

# 一、BP神经网络的概念

    BP神经网络是一种多层的前馈神经网络，其主要的特点是：信号是前向传播的，而误差是反向传播的。具体来说，对于如下的只含一个隐层的神经网络模型：



(三层BP神经网络模型)

BP神经网络的过程主要分为两个阶段，第一阶段是信号的前向传播，从输入层经过隐含层，最后到达输出层；第二阶段是误差的反向传播，从输出层到隐含层，最后到输入层，依次调节隐含层到输出层的权重和偏置，输入层到隐含层的权重和偏置。

# 二、BP神经网络的流程

    在知道了BP神经网络的特点后，我们需要依据信号的前向传播和误差的反向传播来构建整个网络。

## 1、网络的初始化

    假设输入层的节点个数为http://latex.codecogs.com/gif.latex?n，隐含层的节点个数为http://latex.codecogs.com/gif.latex?l，输出层的节点个数为http://latex.codecogs.com/gif.latex?m。输入层到隐含层的权重http://latex.codecogs.com/gif.latex?%5Comega_%7Bij%7D，隐含层到输出层的权重为http://latex.codecogs.com/gif.latex?%5Comega_%7Bjk%7D，输入层到隐含层的偏置为http://latex.codecogs.com/gif.latex?a_j，隐含层到输出层的偏置为http://latex.codecogs.com/gif.latex?b_k。学习速率为http://latex.codecogs.com/gif.latex?%5Ceta，激励函数为http://latex.codecogs.com/gif.latex?g%5Cleft&space;(&space;x&space;%5Cright&space;)。其中激励函数为http://latex.codecogs.com/gif.latex?g%5Cleft&space;(&space;x&space;%5Cright&space;)取Sigmoid函数。形式为：

http://latex.codecogs.com/gif.latex?g%5Cleft&space;(&space;x&space;%5Cright&space;)=%5Cfrac%7B1%7D%7B1+e%5E%7B-x%7D%7D

## 2、隐含层的输出

    如上面的三层BP网络所示，隐含层的输出http://latex.codecogs.com/gif.latex?H_j为

http://latex.codecogs.com/gif.latex?H_j=g%5Cleft&space;(&space;%5Csum_%7Bi=1%7D%5E%7Bn%7D%5Comega&space;_%7Bij%7Dx_i+a_j&space;%5Cright&space;)

## 3、输出层的输出

http://latex.codecogs.com/gif.latex?O_k=%5Csum_%7Bj=1%7D%5E%7Bl%7DH_j%5Comega&space;_%7Bjk%7D+b_k

## 4、误差的计算

    我们取误差公式为：

http://latex.codecogs.com/gif.latex?E=%5Cfrac%7B1%7D%7B2%7D%5Csum_%7Bk=1%7D%5E%7Bm%7D%5Cleft&space;(&space;Y_k-O_k&space;%5Cright&space;)%5E2

其中http://latex.codecogs.com/gif.latex?Y_k为期望输出。我们记http://latex.codecogs.com/gif.latex?Y_k-O_k=e_k，则http://latex.codecogs.com/gif.latex?E可以表示为

http://latex.codecogs.com/gif.latex?E=%5Cfrac%7B1%7D%7B2%7D%5Csum_%7Bk=1%7D%5E%7Bm%7De_k%5E2

以上公式中，http://latex.codecogs.com/gif.latex?i=1%5Ccdots&space;n，http://latex.codecogs.com/gif.latex?j=1%5Ccdots&space;l，http://latex.codecogs.com/gif.latex?k=1%5Ccdots&space;m。

## 5、权值的更新

    权值的更新公式为：

**这里需要解释一下公式的由来：**

这是误差反向传播的过程，我们的目标是使得误差函数达到最小值，即http://latex.codecogs.com/gif.latex?min&space;E，我们使用梯度下降法：

* 隐含层到输出层的权重更新

则权重的更新公式为：

http://latex.codecogs.com/gif.latex?w_%7Bjk%7D=w_%7Bjk%7D+%5Ceta&space;H_je_k

* 输入层到隐含层的权重更新

http://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cpartial&space;E%7D%7B%5Cpartial&space;w_%7Bij%7D%7D=%5Cfrac%7B%5Cpartial&space;E%7D%7B%5Cpartial&space;H_j%7D%5Ccdot&space;%5Cfrac%7B%5Cpartial&space;H_j%7D%7B%5Cpartial&space;%5Comega&space;_%7Bij%7D%7D

其中

则权重的更新公式为：

http://latex.codecogs.com/gif.latex?%5Comega&space;_%7Bij%7D=%5Comega&space;_%7Bij%7D+%5Ceta&space;H_j%5Cleft&space;(&space;1-H_j&space;%5Cright&space;)x_i%5Csum_%7Bk=1%7D%5E%7Bm%7D%5Comega&space;_%7Bjk%7De_k

## 6、偏置的更新

    偏置的更新公式为：

* 隐含层到输出层的偏置更新

则偏置的更新公式为：

http://latex.codecogs.com/gif.latex?b_k=b_k+%5Ceta&space;e_k

* 输入层到隐含层的偏置更新

http://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cpartial&space;E%7D%7B%5Cpartial&space;a_j%7D=%5Cfrac%7B%5Cpartial&space;E%7D%7B%5Cpartial&space;H_j%7D%5Ccdot&space;%5Cfrac%7B%5Cpartial&space;H_j%7D%7B%5Cpartial&space;a_j%7D

其中

则偏置的更新公式为：

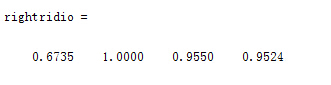
http://latex.codecogs.com/gif.latex?a_k=a_k+%5Ceta&space;H_j%5Cleft&space;(&space;1-H_j&space;%5Cright&space;)%5Csum_%7Bk=1%7D%5E%7Bm%7D%5Comega&space;_%7Bjk%7De_k

## 7、判断算法迭代是否结束

    有很多的方法可以判断算法是否已经收敛，常见的有指定迭代的代数，判断相邻的两次误差之间的差别是否小于指定的值等等。

# 三、实验的仿真

    在本试验中，我们利用BP神经网络处理一个四分类问题，最终的分类结果为：



## MATLAB代码

**主程序**

**[plain]** [view plain](https://blog.csdn.net/google19890102/article/details/32723459) [copy](https://blog.csdn.net/google19890102/article/details/32723459)

1. %% BP的主函数
3. % 清空
4. clear all;
5. clc;
7. % 导入数据
8. load data;
10. %从1到2000间随机排序
11. k=rand(1,2000);
12. [m,n]=sort(k);
14. %输入输出数据
15. input=data(:,2:25);
16. output1 =data(:,1);
18. %把输出从1维变成4维
19. for i=1:2000
20. switch output1(i)
21. case 1
22. output(i,:)=[1 0 0 0];
23. case 2
24. output(i,:)=[0 1 0 0];
25. case 3
26. output(i,:)=[0 0 1 0];
27. case 4
28. output(i,:)=[0 0 0 1];
29. end
30. end
32. %随机提取1500个样本为训练样本，500个样本为预测样本
33. trainCharacter=input(n(1:1600),:);
34. trainOutput=output(n(1:1600),:);
35. testCharacter=input(n(1601:2000),:);
36. testOutput=output(n(1601:2000),:);
38. % 对训练的特征进行归一化
39. [trainInput,inputps]=mapminmax(trainCharacter');
41. %% 参数的初始化
43. % 参数的初始化
44. inputNum = 24;%输入层的节点数
45. hiddenNum = 50;%隐含层的节点数
46. outputNum = 4;%输出层的节点数
48. % 权重和偏置的初始化
49. w1 = rands(inputNum,hiddenNum);
50. b1 = rands(hiddenNum,1);
51. w2 = rands(hiddenNum,outputNum);
52. b2 = rands(outputNum,1);
54. % 学习率
55. yita = 0.1;
57. %% 网络的训练
58. for r = 1:30
59. E(r) = 0;% 统计误差
60. for m = 1:1600
61. % 信息的正向流动
62. x = trainInput(:,m);
63. % 隐含层的输出
64. for j = 1:hiddenNum
65. hidden(j,:) = w1(:,j)'\*x+b1(j,:);
66. hiddenOutput(j,:) = g(hidden(j,:));
67. end
68. % 输出层的输出
69. outputOutput = w2'\*hiddenOutput+b2;
71. % 计算误差
72. e = trainOutput(m,:)'-outputOutput;
73. E(r) = E(r) + sum(abs(e));
75. % 修改权重和偏置
76. % 隐含层到输出层的权重和偏置调整
77. dw2 = hiddenOutput\*e';
78. db2 = e;
80. % 输入层到隐含层的权重和偏置调整
81. for j = 1:hiddenNum
82. partOne(j) = hiddenOutput(j)\*(1-hiddenOutput(j));
83. partTwo(j) = w2(j,:)\*e;
84. end
86. for i = 1:inputNum
87. for j = 1:hiddenNum
88. dw1(i,j) = partOne(j)\*x(i,:)\*partTwo(j);
89. db1(j,:) = partOne(j)\*partTwo(j);
90. end
91. end
93. w1 = w1 + yita\*dw1;
94. w2 = w2 + yita\*dw2;
95. b1 = b1 + yita\*db1;
96. b2 = b2 + yita\*db2;
97. end
98. end
100. %% 语音特征信号分类
101. testInput=mapminmax('apply',testCharacter',inputps);
103. for m = 1:400
104. for j = 1:hiddenNum
105. hiddenTest(j,:) = w1(:,j)'\*testInput(:,m)+b1(j,:);
106. hiddenTestOutput(j,:) = g(hiddenTest(j,:));
107. end
108. outputOfTest(:,m) = w2'\*hiddenTestOutput+b2;
109. end
111. %% 结果分析
112. %根据网络输出找出数据属于哪类
113. for m=1:400
114. output\_fore(m)=find(outputOfTest(:,m)==max(outputOfTest(:,m)));
115. end
117. %BP网络预测误差
118. error=output\_fore-output1(n(1601:2000))';
120. k=zeros(1,4);
121. %找出判断错误的分类属于哪一类
122. for i=1:400
123. if error(i)~=0
124. [b,c]=max(testOutput(i,:));
125. switch c
126. case 1
127. k(1)=k(1)+1;
128. case 2
129. k(2)=k(2)+1;
130. case 3
131. k(3)=k(3)+1;
132. case 4
133. k(4)=k(4)+1;
134. end
135. end
136. end
138. %找出每类的个体和
139. kk=zeros(1,4);
140. for i=1:400
141. [b,c]=max(testOutput(i,:));
142. switch c
143. case 1
144. kk(1)=kk(1)+1;
145. case 2
146. kk(2)=kk(2)+1;
147. case 3
148. kk(3)=kk(3)+1;
149. case 4
150. kk(4)=kk(4)+1;
151. end
152. end
154. %正确率
155. rightridio=(kk-k)./kk

**激活函数**

**[plain]** [view plain](https://blog.csdn.net/google19890102/article/details/32723459) [copy](https://blog.csdn.net/google19890102/article/details/32723459)

1. %% 激活函数
2. function [ y ] = g( x )
3. y = 1./(1+exp(-x));
4. end