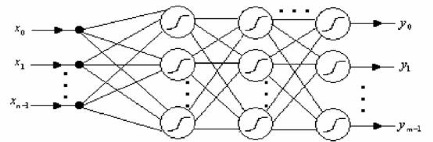
BP（Back Propagation）网络是1986年由Rumelhart和McCelland为首的科学家小组提出，是一种按误差逆传播算法训练的多层前馈网络， 是目前应用最广泛的神经网络模型之一。BP网络能学习和存贮大量的输入-输出模式映射关系，而无需事前揭示描述这种映射关系的数学方程。

一个神经网络的结构示意图如下所示。

[](http://hi.csdn.net/attachment/201106/11/0_13077848769Azw.gif)

BP神经网络模型拓扑结构包括输入层（input）、隐层(hide layer)和输出层(output layer)。输入层神经元的个数由样本属性的维度决定，输出层神经元的个数由样本分类个数决定。隐藏层的层数和每层的神经元个数由用户指定。每一层包含 若干个神经元，每个神经元包含一个而阈值[clip_image004](http://hi.csdn.net/attachment/201106/11/0_13077848762ih9.gif)，用来改变神经元的活性。网络中的弧线[clip_image006](http://hi.csdn.net/attachment/201106/11/0_1307784876B5V5.gif)表示前一层神经元和后一层神经元之间的权值。每个神经元都有输入和输出。输入层的输入和输出都是训练样本的属性值。

对于隐藏层和输出层的输入[clip_image008](http://hi.csdn.net/attachment/201106/11/0_1307784876xBB5.gif) 其中，[clip_image006[1]](http://hi.csdn.net/attachment/201106/11/0_13077848771DbB.gif)是由上一层的单元i到单元j的连接的权；[clip_image010](http://hi.csdn.net/attachment/201106/11/0_1307784877uBi9.gif)是上一层的单元i的输出；而[clip_image004[1]](http://hi.csdn.net/attachment/201106/11/0_1307784877lfqU.gif)是单元j的阈值。

神经网络中神经元的输出是经由赋活函数计算得到的。该函数用符号表现单元代表的神经元活性。赋活函数一般使用simoid函数（或者logistic函数）。神经元的输出为：

[clip_image012](http://hi.csdn.net/attachment/201106/11/0_1307784877D511.gif)

除此之外，神经网络中有一个学习率（l）的概念，通常取0和1之间的值，并有助于找到全局最小。如果学习率太小，学习将进行得很慢。如果学习率太大，可能出现在不适当的解之间摆动。

交代清楚了神经网络中基本要素，我们来看一下BP算法的学习过程：

BPTrain(){

      初始化network的权和阈值。

      while 终止条件不满足 {

            for samples中的每个训练样本X {

            // 向前传播输入

            for 隐藏或输出层每个单元j {

[clip_image014](http://hi.csdn.net/attachment/201106/11/0_1307784877W1Wz.gif)；// 相对于前一层i，计算单元j的净输入 [clip_image016](http://hi.csdn.net/attachment/201106/11/0_13077848780025.gif)；// 计算单元j的输出

            }

            // 后向传播误差

            for 输出层每个单元j {

[clip_image018](http://hi.csdn.net/attachment/201106/11/0_1307784878BqIX.gif)；// 计算误差

            }

            for 由最后一个到第一个隐藏层，对于隐藏层每个单元j {

[clip_image020](http://hi.csdn.net/attachment/201106/11/0_13077848783V2C.gif)；// k是j的下一层中的神经元

            }

            for network中每个权[clip_image006[2]](http://hi.csdn.net/attachment/201106/11/0_130778487808HT.gif) {

[clip_image022](http://hi.csdn.net/attachment/201106/11/0_1307784878ZO9o.gif)； // 权增值

[clip_image024](http://hi.csdn.net/attachment/201106/11/0_1307784879kFDD.gif)； // 权更新

             }

             for network中每个偏差[clip_image004[2]](http://hi.csdn.net/attachment/201106/11/0_1307784879I522.gif) {

[clip_image027](http://hi.csdn.net/attachment/201106/11/0_130778487983OM.gif)； // 偏差增值

[clip_image029](http://hi.csdn.net/attachment/201106/11/0_1307784882l8g3.gif)；// 偏差更新

             }

      }

}

算法基本流程就是：

1、初始化网络权值和神经元的阈值（最简单的办法就是随机初始化）

2、前向传播：按照公式一层一层的计算隐层神经元和输出层神经元的输入和输出。

3、后向传播：根据公式修正权值和阈值

直到满足终止条件。

算法中还有几点是需要说明的：

1、关于[clip_image031](http://hi.csdn.net/attachment/201106/11/0_13077848831pjP.gif)，[clip_image031[1]](http://hi.csdn.net/attachment/201106/11/0_1307784883TJg7.gif)是神经元的误差。

对于输出层神经元[clip_image018[1]](http://hi.csdn.net/attachment/201106/11/0_1307784883lAfa.gif)，其中，[clip_image010[1]](http://hi.csdn.net/attachment/201106/11/0_1307784883iOfc.gif)是单元j的实际输 出，而[clip_image035](http://hi.csdn.net/attachment/201106/11/0_13077848839Hbt.gif)是j基于给定训练样本的已知类标号的真正输出。

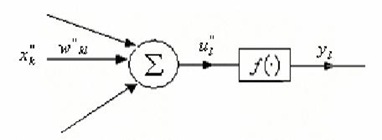
对于隐藏层神经元[clip_image037](http://hi.csdn.net/attachment/201106/11/0_13077848849Z9s.gif)，其中，[clip_image039](http://hi.csdn.net/attachment/201106/11/0_1307784887VV61.gif)是由下一较高层中单元k到单元j的连接权，而[clip_image041](http://hi.csdn.net/attachment/201106/11/0_13077848870ki0.gif)是单元k的误差。

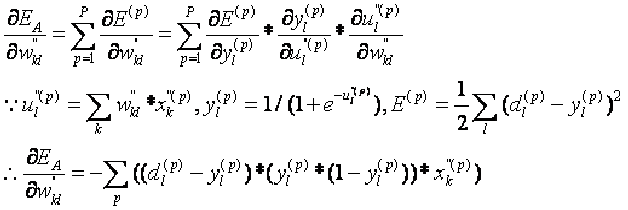
权值增量是[clip_image022[1]](http://hi.csdn.net/attachment/201106/11/0_1307784887drR1.gif)，阈值增量是[clip_image027[1]](http://hi.csdn.net/attachment/201106/11/0_1307784888pT24.gif)，其中[clip_image043](http://hi.csdn.net/attachment/201106/11/0_1307784888MAKM.gif)是学习率。

对于[clip_image031[2]](http://hi.csdn.net/attachment/201106/11/0_1307784888zpE9.gif)的推导采用了梯度下降的算法。推导的前提是保证输出单元的均方差最小。[clip_image046](http://hi.csdn.net/attachment/201106/11/0_1307784888Ii6I.gif)，其中P是样本总数，m是输出层神经元个数[clip_image048](http://hi.csdn.net/attachment/201106/11/0_13077848894SrV.gif)是样本实际输出，[clip_image050](http://hi.csdn.net/attachment/201106/11/0_1307784889K1PK.gif)是神经网络输出。

梯度下降思路就是对[clip_image052](http://hi.csdn.net/attachment/201106/11/0_1307784889Mqz9.gif)求[clip_image039[1]](http://hi.csdn.net/attachment/201106/11/0_1307784892wDrr.gif)的导数。

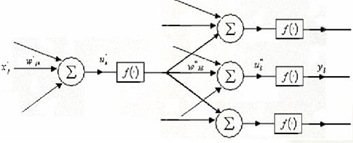
对于输出层：

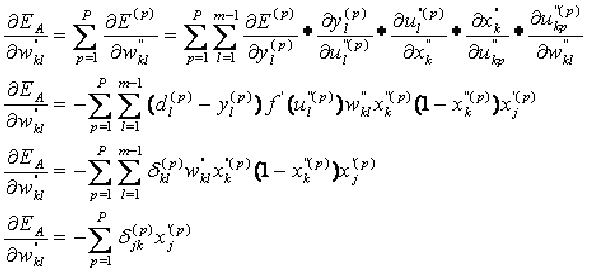
[](http://hi.csdn.net/attachment/201106/11/0_1307784892Rr5i.gif)

[](http://hi.csdn.net/attachment/201106/11/0_1307784896YzmS.gif)

其中的[clip_image058](http://hi.csdn.net/attachment/201106/11/0_1307784896181s.gif)就是[clip_image018[2]](http://hi.csdn.net/attachment/201106/11/0_1307784896h4Oe.gif)。

对于隐藏层：

[](http://hi.csdn.net/attachment/201106/11/0_1307784896dAMD.gif)

[](http://hi.csdn.net/attachment/201106/11/0_1307784896m2iA.gif)

其中[clip_image064](http://hi.csdn.net/attachment/201106/11/0_1307784896rzKu.gif)=[clip_image066](http://hi.csdn.net/attachment/201106/11/0_13077848973w8v.gif)就是隐藏层的误差计算公式。

2、关于终止条件，可以有多种形式：

§ 前一周期所有的[clip_image068](http://hi.csdn.net/attachment/201106/11/0_1307784897504O.gif)都太小，小于某个指定的阈值。

§ 前一周期未正确分类的样本百分比小于某个阈值。

§ 超过预先指定的周期数。

§ 神经网络的输出值和实际输出值的均方误差小于某一阈值。

一般地，最后一种终止条件的准确率更高一些。

在实际使用BP神经网络的过程中，还会有一些实际的问题：

1、 样本处理。对于输出，如果只有两类那么输出为0和1，只有当[clip_image070](http://hi.csdn.net/attachment/201106/11/0_1307784897cQO3.gif)趋于正负无穷大的时候才会输出0，1。因此条件可适当放宽，输出>0.9时就认为是1，输出<0.1时认为是0。对于输入，样本也需要做归一化处理。

2、 网络结构的选择。主要是指隐藏层层数和神经元数决定了网络规模，网络规模和性能学习效果密切相关。规模大，计算量大，而且可能导致过度拟合；但是规模小，也可能导致欠拟合。

3、 初始权值、阈值的选择，初始值对学习结果是有影响的，选择一个合适初始值也非常重要。

4、 增量学习和批量学习。上面的算法和数学推导都是基于批量学习的，批量学习适用于离线学习，学习效果稳定性好；增量学习使用于在线学习，它对输入样本的噪声是比较敏感的，不适合剧烈变化的输入模式。

5、 对于激励函数和误差函数也有其他的选择。

总的来说BP算法的可选项比较多，针对特定的训练数据往往有比较大的优化空间。