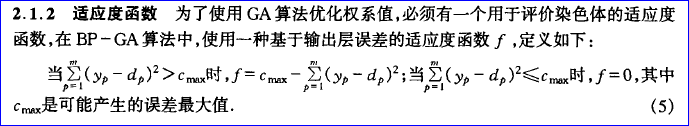
**神经网络**

# 概述

BP(Back Propagation)神经网络是1986年由Rumelhart和McCelland为首的科学家小组提出,是一种按误差逆向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.它的学习规则是使用梯度下降法,通过反向传播来不断调整网络的权值和阈值,使网络的误差平方和最小.BP神经网络模型拓扑结构包括输入层(input)、隐藏层 (hidden layer)和输出层(output layer).

Gauss-Newton Approximation To Bayesian Learning

Hecht-nielsen证明具有1个隐藏层的3层前馈型神经网络可以逼近任何多变量函数，故采用3层BP神经网络实现刀具磨损的预测。



# 激活函数

## Sigmoid 型函数（S型函数）

Sigmoid 型函数是指一类S 型曲线函数，为两端饱和函数。

### Logistic函数（逻辑斯蒂函数）



### Tanh函数（双曲正切函数）



两者关系



## 修正线性单元（Rectified Linear Unit，ReLU）



## 带泄露的ReLU（Leaky ReLU）



# 反向传播算法

——第层神经元的个数；

——神经网络的层数，输入层是第0层；

——第层第个神经元到第层第个神经元的权重；

——第层到第层的权重矩阵，；

——第层第个神经元的偏置；

——第层的偏置向量，

——第层第个神经元的输入值；

——第层所有神经元的输入值向量，；

——第层第个神经元的激活值；

——第层所有神经元的激活值向量，。

不难看出，的值取决于上一层神经元的激活：



写成向量形式：

!神经网络结构，它并不共享从文本的不同位置上学到的特征

假设训练数据





损失函数为：





反向传播过程中我们只能计算单个训练样本的和,扯淡吧！

## 输出层的误差方程

为了简单起见，在下面的推导中，对于单个样本，去掉的上标，简记为



仅与和相关，来衡量第层第个神经元的误差



等式右边第一项衡量了代价函数随网络最终输出的变化快慢，而第二项则衡量了激活函数输出随的变化快慢。当激活函数饱和，即时，无论多大，最终，输出神经元进入饱和区，停止学习。





写成向量形式





为Hadamard积，即矩阵的点积，对应元素相乘就行。

## 误差传递方程



证明



因为



所以





写成向量的形式为



## 代价函数对权重的改变率



## 代价函数对偏置的改变率



总结：BP算法四个核心公式

**输出层的误差方程** 

**误差传递方程** 

**代价函数对权重的改变率** 

**代价函数对偏置的改变率** 

向量形式









当层的激活输出接近0的时候，无论返回的误差有多大，或者都很小，所以说神经元饱和不利于训练。当输入神经元没有被激活，或者输出神经元处于饱和状态，权重和偏置会学习的非常慢，这不是我们想要的效果。这也说明了为什么我们平时总是说激活函数的选择非常重要。

## BP 算法计算某个训练数据的代价函数对参数的偏导数

1. 准备历史数据
2. 确定神经网络结果层数，每层的神经元个数；
3. 初始化参数；
4. 利用前向传播公式计算每层的状态和激活值





1. 计算输出层的误差：



1. 计算第层误差，：



1. 代价函数对计算每层的参数的偏导数：





1. 更新每层参数：





1. 返回第4步，直到收敛。

# BP神经网络设计

在进行BP网络的设计时，一般应从网络的层数、每层中的神经元个数和激活函数的选取、初始值的设定以及学习速率等几个方面来进行考虑，下面是一些选取的原则。

## 网络的层数

理论已经证明，具有偏差和至少一个S型隐层加上一个线性输出层的网络，能够逼近任何有理函数，增加层数可以进一步降低误差，提高精度，但同时也是网络 复杂化。另外不能用仅具有非线性激活函数的单层网络来解决问题，因为能用单层网络解决的问题，用自适应线性网络也一定能解决，而且自适应线性网络的运算速度更快，而对于只能用非线性函数解决的问题，单层精度又不够高，也只有增加层数才能达到期望的结果。

## 隐藏层神经元的个数

网络训练精度的提高，可以通过采用一个隐含层，而增加其神经元个数的方法来获得，这在结构实现上要比增加网络层数简单得多。一般而言，我们用精度和训练网络的时间来恒量一个神经网络设计的好坏：

（1）神经元数太少时，网络不能很好的学习，训练迭代的次数也比较多，训练精度也不高。

（2）神经元数太多时，网络的功能越强大，精确度也更高，训练迭代的次数也大，可能会出现过拟合(over fitting)现象。

由此，我们得到神经网络隐藏层神经元个数的选取原则是：在能够解决问题的前提下，再加上一两个神经元，以加快误差下降速度即可。

在网络设计过程中, 隐藏层神经元数的确定十分重要。隐藏层神经元个数过多, 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。网络中隐藏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, 对于隐藏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。本文在选取隐藏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。

# 梯度消失问题及其解决办法

误差反向传播方程为：



当时，





当时，





可以看到，激活函数的导数都小于1，所以误差每经过一层传递都会不断衰减。当神经网络层数较多时，误差不断减小，甚至接近于0，导致梯度也不断减小，甚至消失，是的整个网络难以训练，这就是梯度消失问题（Vanishing Gradient Problem）





可选用线性激活函数，导数为1。