集成式学习

* 1. 什么是集成式学习 (boosting and bagging)

集成式学习是基于这样一种思想:对于一个复杂任务来说,将多个专家的判断进行适当的综合所得出的判断,要比其中任何一个专家单独的判断好。通俗点说,就是“三个臭皮匠顶个诸葛亮”。其中所说的专家相当于一个分类器，但面临某一个复杂的问题时，单一的分类器未必能将问题有效的解决。我们把这种分类器称之为若分类器，一般说来弱分类器只给出比随机猜测好一点的一个分类结果。面对这种复杂的问题，我们又无法轻松的找到一个强分类器，因此一个自然的想法就是能不能利用多个这种弱分类器构成出某种强分类器。这便是集成学习的思想。我们用e表示一个弱分类器的错误概率 ，因为弱分类只比随机猜测强一点，因此e<0.5，但e又和我们期望的错误率相差较远。假定随机猜测的概率为0.5，定义r=0.5-e，因为e<0.5，所以r>0，这个r表示我们当前的弱分类比随机猜测强的

其中主要由Booststrap方法，其中又包括bagging和 boosting 等具体是构造方法。

**Bootstraping:** 名字来自成语“pull up by your own bootstraps”，意思是依靠你自己的资源，称为自助法，它是一种有放回的抽样方法，它是非参数统计中一种重要的估计统计量方差进而进行区间估计的统计方法。其核心思想和基本步骤如下：

　　（1） 采用重抽样技术从原始样本中抽取一定数量（自己给定）的样本，此过程允许重复抽样。

　　（2） 根据抽出的样本计算给定的统计量T。

　　（3） 重复上述N次（一般大于1000），得到N个统计量T。

　　（4） 计算上述N个统计量T的样本方差，得到统计量的方差。

　　应该说Bootstrap是现代统计学较为流行的一种统计方法，在小样本时效果很好。通过方差的估计可以构造置信区间等，其运用范围得到进一步延伸。

**bagging：**bootstrap aggregating的缩写。让该学习算法训练多轮，每轮的训练集由从初始的训练集中随机取出的n个训练样本组成，某个初始训练样本在某轮训练集中可以出现多次或根本不出现，训练之后可得到一个预测函数序列h\_1，⋯ ⋯h\_n ，最终的预测函数H对分类问题采用投票方式，对回归问题采用简单平均方法对新示例进行判别。训练R个分类器f\_i，分类器之间其他相同就是参数不同。其中f\_i是通过从训练集合中(N篇文档)随机取(取后放回)N次文档构成的训练集合训练得到的。对于新文档d，用这R个分类器去分类，得到的最多的那个类别作为d的最终类别。

**Bagging与Boosting**的区别：我们用下面的表格总结两者的区别。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 训练集 | 预测函数 | 准确性 | 使用要求 |
| Bagging | 随机选择,各轮训练集相互独立 | 没有权重;可以并行生成 | 在有些数据集中，boosting会引起退化 | 要求“不稳定”的分类方法 |
| Boosting | 各轮训练集并不独立,它的选择与前轮的学习结果有关 | 有权重;只能顺序生成 | 在大多数数据集中，boosting的准确性比bagging高 | 要求“不稳定”的分类方法  训练集的小变动能够使得分类模型显著变动 |

可以看出，二者的主要区别是取样方式不同。Bagging采用均匀取样，而Boosting根据错误率来取样，因此Boosting的分类精度要优于Bagging。Bagging的训练集的选择是随机的，各轮训练集之间相互独立，是一个纯粹的降低相关度的方法，而Boosting的各轮训练集的选择与前面各轮的学习结果有关；Bagging的各个预测函数没有权重，而Boosting是有权重的；Bagging的各个预测函数可以并行生成，而Boosting的各个预测函数只能顺序生成。对于象神经网络这样极为耗时的学习方法。Bagging可通过并行训练节省大量时间开销。bagging和boosting都可以有效地提高分类的准确性。在大多数数据集中，boosting的准确性比bagging高。但在有些数据集中，boosting会引起过拟合 （overfitting）导致泛化能力下降。

因为Boosting方法在应用中更加的广泛，本文将着重介绍Boosting的方法

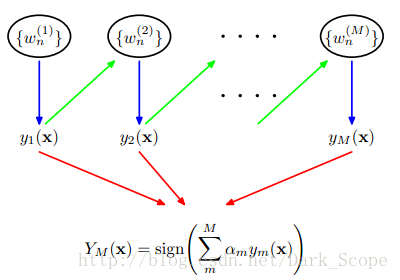
**Boosting方法**

Boosting是一种提高任意给定学习算法准确度的方法。它的思想起源于 Valiant提出的 PAC ( Probably Approximately Correct)学习模型。Valiant和 Kearns提出了弱学习和强学习的概念 ,识别错误率小于1/2,也即准确率仅比随机猜测略高的学习算法称为弱学习算法;识别准确率很高并能在多项式时间内完成的学习算法称为强学习算法。同时 ,Valiant和 Kearns首次提出了 PAC学习模型中弱学习算法和强学习算法的等价性问题,即任意给定仅比随机猜测略好的弱学习算法 ,是否可以将其提升为强学习算法 ? 如果二者等价 ,那么只需找到一个比随机猜测略好的弱学习算法就可以将其提升为强学习算法 ,而不必寻找很难获得的强学习算法。1990年, Schapire最先构造出一种多项式级的算法 ,对该问题做了肯定的证明 ,这就是最初的 Boosting算法。一年后 ,Freund提出了一种效率更高的Boosting算法。但是,这两种算法存在共同的实践上的缺陷 ,那就是都要求事先知道弱学习算法学习正确的下限。1995年 , Freund和 schap ire改进了Boosting算法 ,提出了 AdaBoost (Adap tive Boosting)算法[ 5 ],该算法效率和 Freund于 1991年提出的 Boosting算法几乎相同 ,但不需要任何关于弱学习器的先验知识 ,因而更容易应用到实际问题当中。之后 , Freund和 schapire进一步提出了改变 Boosting投票权重的 AdaBoost . M1,AdaBoost . M2等算法 ,在机器学习领域受到了极大的关注。

* 1. adaboost算法

由于Boosting算法在解决实际问题时有一个重大的缺陷,即他们都要求事先知道弱分类算法分类正确率的下限,这在实际问题中很难做到。后来 Freund 和 Schapire提出了 AdaBoost 算法,该算法的效率与 Freund 方法的效率几乎一样,却可以非常容易地应用到实际问题中。AdaBoost 是Boosting 算法家族中代表算法,AdaBoost 主要是在整个训练集上维护一个分布权值向量 D( x) t ,用赋予权重的训练集通过弱分类算法产生分类假设 Ht ( x) ,即基[分类器](http://baike.baidu.com/view/895803.htm),然后计算他的错误率,用得到的错误率去更新分布权值向量 D( x) t ,对错误分类的样本分配更大的权值,正确分类的样本赋予更小的权值。每次更新后用相同的弱分类算法产生新的分类假设,这些分类假设的序列构成多分类器。对这些多分类器用[加权](http://baike.baidu.com/view/384.htm)的方法进行联合,最后得到决策结果。这种方法不要求产生的单个分类器有高的识别率,即不要求寻找识别率很高的基分类算法,只要产生的基分类器的识别率大于 0.5 ,就可作为该多分类器序列中的一员。

在实际应用中，寻找多个识别率不是很高的弱分类算法比寻找一个识别率很高的强分类算法要容易得多,AdaBoost 算法的任务就是完成将容易找到的识别率不高的弱分类算法提升为识别率很高的强分类算法,这也是 AdaBoost 算法的核心指导思想所在,如果算法完成了这个任务,那么在分类时,只要找到一个比随机猜测略好的弱分类算法,就可以将其提升为强分类算法,而不必直接去找通常情况下很难获得的强分类算法。通过产生多[分类器](http://baike.baidu.com/view/895803.htm)最后联合的方法提升弱分类算法,让他变为强的分类算法,也就是给定一个弱的学习算法和训练集,在训练集的不同子集上,多次调用弱学习算法,最终按加权方式联合多次弱学习算法的预测结果得到最终学习结果。其算法示意图如下：



整个Adaboost 迭代算法可以分成3步：

1. 初始化训练数据的权值分布。如果有N个样本，则每一个训练样本最开始时都被赋予 x同的权重：1/N。
2. 训练弱分类器。具体训练过程中，如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它的权重就被降低；相反，如果某个样本点没有被准确地分类，那么它的权重就得到提高。然后，权重更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。
3. 将各个训练得到的弱分类器组合成强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。换言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

其数学形式为：

输入:训练数据集 T={(x1,y1),(x2,y2),...,(xN,yN)},其中 xi∈X⊆Rn,表示输入数据,yi∈Y={-1,+1}, 表示类别标签;弱学习算法h(x)。

输出:最终分类器 H(x)。

流程:

(1) 初始化训练数据的概率分布,刚开始为均匀分布 。, 其中表示第i次训练过程中每个样本的权重。其初始值对于所有样本等概率分布，即。

(2) 对 m=1,2,...,M, 分别进行弱分类器的选择，并计算每个弱分类器的权重和更新样本权重。

(a) 使用具有权值分布Dm的训练数据集进行学习(任意选一种模型都可以,例如朴素贝叶斯,

决策树,SVM 等,并且每一轮迭代都可以用不同的模型),得到一个弱分类器

 （1）

其中表示一个弱分类器，这个分类器将样本从特征空间X分布映射到一个二值分布空间，其中-1表示负样本，+1表示正样本。

(b) 计算 fm(x)在训练数据集上的分类误差率

 (2)

其中表示的是概率值，表示只是函数，即当括号内的表达式成立时，=1，否则=0。

(c) 计算弱分类器hm(x)的系数

 (3)

(d) 更新训练数据的权值分布为，其中为

 （4）

其中，为归一化因子，通过归一化因子，使得成为一个分布，即

(3)通过上述（2）过程，得到M个弱分类器，将M个基本分类器进行线性组合得到

 （5）

则最终的分类器为：

 (6)

其中，步骤(1) 初始时假设训练数据集具有均匀分布,即每个训练样本在弱分类器的学习中作用相同。

步骤(2) (c)中 αm 表示 hm(x)在最终分类器中的重要性。由式(公式 2)可知,当 em ≤1/2 时,αm≥0, 并且 αm 随着 em 的减小而增大,即意味着误差率越小的基本分类器在最终分类器中的作用越大。 步骤（2）(d)中公式（4）可以写成:

 （7）

由此可知,被弱分类器 hm(x)误分类的样本的权值增加,而被正确分类的样本的权值减小。因此误分类样本在下一轮学习中起到更大的作用。不改变所给的训练数据,而不断改变训练数据权值的分布,使得训练数据在基本分类器的学习中起不同的作用,这是 AdaBoost 的一个特点。

步骤(3) 这里,αm 之和并不等于1。h(x)的符号决定实例 x 的类别,h(x)的绝对值表示分类的确信度。利用基本分类器进行线性组合得到最终分类器是 AdaBoost 的另一个特点。

* 1. Adaboost 的损失度函数

1.2小节给出了adaboost算法的计算过程，那么这个算法为什么能将若干个弱分类器集合成一个强分类器呢？我们从最小损失度函数的角度分析，首先定义训练到m时的每个样本的损失函数，其定义如下：

 （8）

其中，表示由前m个弱分类器组合得到的函数。通过公式（7）可以看出，若能够将样本正确的分出来，则和具有相同的符号，即>0，此时的算式函数的取值很变小，否则损失函数会变大。那么根据公式（7）得到分类器在数据集上的损失函数为

 （8）

现在我们将拆解成前m-1个分类器的和的形式和第m个弱分类器，得到

 （9）

将公式带回到公式（7）中，得到

 （10）

如果我们定义为样本的权重，则有，又因为这个弱分类器只能取{-1,+1}的值，则根据能否将样本正确分开，公式（10）可以改写成

 （11）

那么此时样本集合上的损失函数为：

 （12）

根据公式（2）可以，将公式(2)带入公式（12）得到

 （13）

通过上面公式的变化，可以看出经过m次弱分类器训练adaboost算法的损失函数为公式（13），通过添加第m个分类器， 我们希望将损失函数变得最小，即对的倒数为0.，得到

 （14）

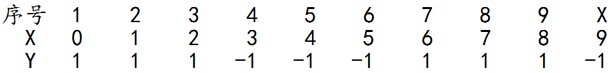
根据公式（14）可以计算出的最优值，得到：

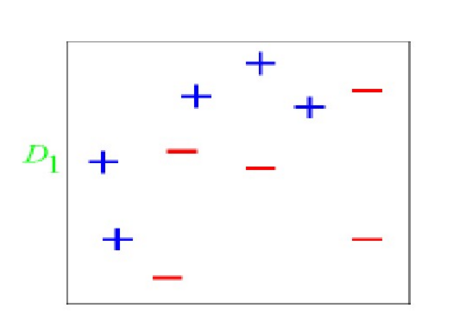
 （15）

可以发现公式（15）和1.2小节中adaboost给出的权重一样，这就是adaboost算法在损失函数上的解释。在分析的过程中，我们定义，尽管和在数值上并不相等，不过它们只是相差一个乘法因子，因为要进行归一化，即要求满足。

Adaboost实例解析

   例1. 下面，给定下列训练样本,其样本分布如下图所示，请用AdaBoost算法学习一个强分类器。





图中“+”和“-”分别表示正负样本

    求解过程：初始化训练数据的权值分布，令每个权值W1i = 1/N = 0.1，其中，N = 10，i = 1,2, ..., 10，然后分别对于m = 1,2,3, ...等值进行迭代。

    拿到这10个数据的训练样本后，根据 X 和 Y 的对应关系，要把这10个数据分为两类，一类是“1”，一类是“-1”，根据数据的特点发现：“0 1 2”这3个数据对应的类是“1”，“3 4 5”这3个数据对应的类是“-1”，“6 7 8”这3个数据对应的类是“1”，9是比较孤独的，对应类“-1”。抛开孤独的9不讲，“0 1 2”、“3 4 5”、“6 7 8”这是3类不同的数据，分别对应的类是1、-1、1，直观上推测可知，可以找到对应的数据分界点，比如2.5、5.5、8.5 将那几类数据分成两类。当然，这只是主观臆测，下面实际计算下这个过程。

**迭代过程1**

对于m=1，在权值分布为**D1**（10个数据，每个数据的权值皆初始化为0.1）的训练数据上，经过计算可得：

1. 阈值v取2.5时误差率为0.3（x < 2.5时取1，x > 2.5时取-1，**则6 7 8分错**，误差率为0.3），
2. 阈值v取5.5时误差率最低为0.4（x < 5.5时取1，x > 5.5时取-1，则3 4 5 6 7 8皆分错，误差率0.6大于0.5，不可取。故令x > 5.5时取1，x < 5.5时取-1，则0 1 2 9分错，误差率为0.4），
3. 阈值v取8.5时误差率为0.3（x < 8.5时取1，x > 8.5时取-1，则3 4 5分错，误差率为0.3）。

所以无论阈值v取2.5，还是8.5，总得分错3个样本，故可任取其中任意一个如2.5，弄成第一个基本分类器为：

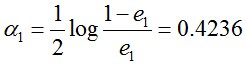


上面说阈值v取2.5时则6 7 8分错，所以误差率为0.3，更加详细的解释是：因为样本集中

1. 0 1 2对应的类（Y）是1，因它们本身都小于2.5，所以被h1(x)分在了相应的类“1”中，分对了。
2. 3 4 5本身对应的类（Y）是-1，因它们本身都大于2.5，所以被h1(x)分在了相应的类“-1”中，分对了。
3. 但6 7 8本身对应类（Y）是1，却因它们本身大于2.5而被h1(x)分在了类"-1"中，所以这3个样本被分错了。
4. 9本身对应的类（Y）是-1，因它本身大于2.5，所以被h1(x)分在了相应的类“-1”中，分对了。

从而得到h1(x)在训练数据集上的误差率（被h1(x)误分类样本“6 7 8”的权值之和）**e1=P(h1(xi)≠yi) = 3\*0.1 = 0.3**。

然后根据误差率e1计算h1的系数：



这个a1代表h1(x)在最终的分类函数中所占的权重，为0.4236。  
接着更新训练数据的权值分布，用于下一轮迭代：



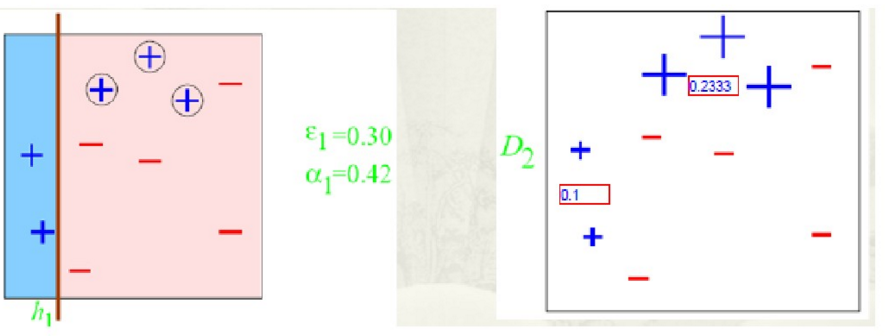
值得一提的是，由权值更新的公式可知，每个样本的新权值是变大还是变小，取决于它是被分错还是被分正确。

即如果某个样本被分错了，则yi \* hm(xi)为负，负负等正，结果使得整个式子变大（样本权值变大），否则变小。

第一轮迭代后，最后得到各个数据**新**的权值分布**D2**= (0.0715, 0.0715, 0.0715, 0.0715, 0.0715,  0.0715, 0.1666, 0.1666, 0.1666, 0.0715)。由此可以看出，因为样本中是数据“6 7 8”被h1(x)分错了，所以它们的权值由之前的0.1增大到0.1666，反之，其它数据皆被分正确，所以它们的权值皆由之前的0.1减小到0.0715。

分类函数f1(x)= a1\*h1(x) = 0.4236h1(x)。

此时，得到的第一个基本分类器sign(f1(x))在训练数据集上有3个误分类点（即6 7 8）。第一步的计算过程如下图：



    从上述第一轮的整个迭代过程可以看出：被误分类样本的权值之和影响误差率，误差率影响基本分类器在最终分类器中所占的权重。

**迭代过程2**

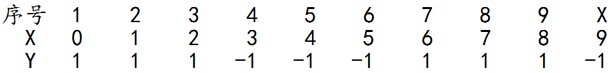
对于m=2，在权值分布为**D2**= (0.0715, 0.0715, 0.0715, 0.0715, 0.0715,  0.0715, 0.1666, 0.1666, 0.1666, 0.0715)的训练数据上，经过计算可得：

1. 阈值v取2.5时误差率为0.1666\*3（x < 2.5时取1，x > 2.5时取-1，则6 7 8分错，误差率为0.1666\*3），
2. 阈值v取5.5时误差率最低为0.0715\*4（x > 5.5时取1，x < 5.5时取-1，则0 1 2 9分错，误差率为0.0715\*3 + 0.0715），
3. **阈值v取8.5**时误差率为0.0715\*3（x < 8.5时取1，x > 8.5时取-1，**则3 4 5分错**，误差率为0.0715\*3）。

所以，阈值v取8.5时误差率最低，故第二个基本分类器为：

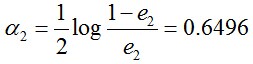


面对的还是下述样本：



很明显，h2(x)把样本“3 4 5”分错了，根据D2可知它们的权值为0.0715, 0.0715,  0.0715，所以h2(x)在训练数据集上的误差率e2=P(h2(xi)≠yi) = 0.0715 \* 3 = 0.2143。

计算h2的系数：

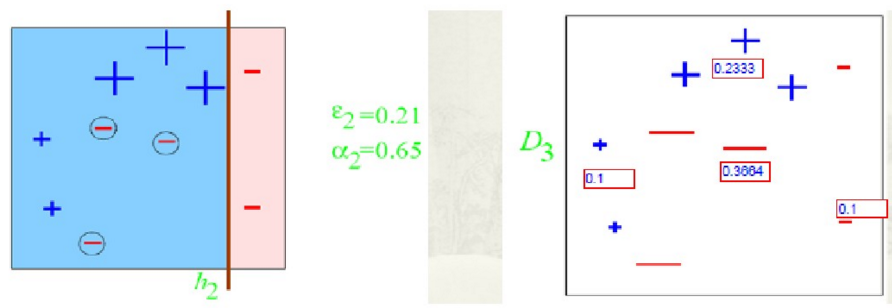


更新训练数据的权值分布：



D3 = (0.0455, 0.0455, 0.0455, 0.1667, 0.1667,  0.01667, 0.1060, 0.1060, 0.1060, 0.0455)。被分错的样本“3 4 5”的权值变大，其它被分对的样本的权值变小。  
f2(x)=0.4236Gh(x) + 0.6496Gh(x)

此时，得到的第二个基本分类器sign(f2(x))在训练数据集上有3个误分类点（即3 4 5）。第二步的过程如下图



**迭代过程3**

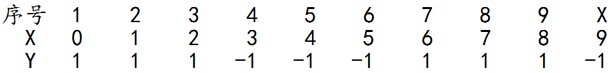
对于m=3，在权值分布为**D3**= (0.0455, 0.0455, 0.0455, 0.1667, 0.1667,  0.01667, 0.1060, 0.1060, 0.1060, 0.0455)的训练数据上，经过计算可得：

1. 阈值v取2.5时误差率为0.1060\*3（x < 2.5时取1，x > 2.5时取-1，则6 7 8分错，误差率为0.1060\*3），
2. **阈值v取5.5**时误差率最低为0.0455\*4（x > 5.5时取1，x < 5.5时取-1，**则0 1 2 9分错**，误差率为0.0455\*3 + 0.0715），
3. 阈值v取8.5时误差率为0.1667\*3（x < 8.5时取1，x > 8.5时取-1，则3 4 5分错，误差率为0.1667\*3）。

所以阈值v取5.5时误差率最低，故第三个基本分类器为：



依然还是原样本：



此时，被误分类的样本是：0 1 2 9，这4个样本所对应的权值皆为0.0455，

所以h3(x)在训练数据集上的**误差率e3**= P(h3(xi)≠yi) = **0.0455\*4** = 0.1820。

计算h3的系数：

http://img.voidcn.com/vcimg/000/000/917/259_026_84e.jpg

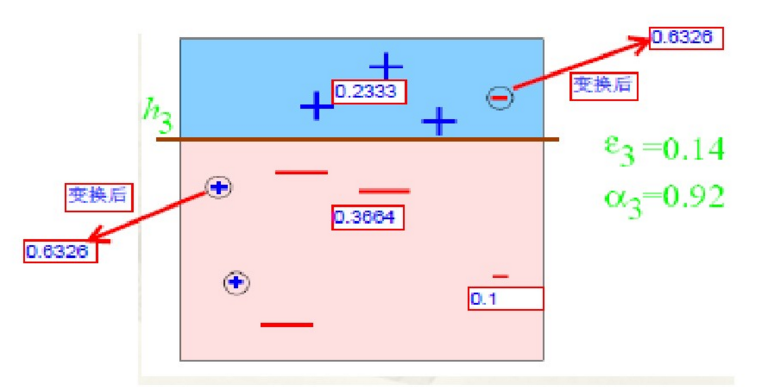
更新训练数据的权值分布：



**D4**= (0.125, 0.125, 0.125, 0.102, 0.102,  0.102, 0.065, 0.065, 0.065, 0.125)。被分错的样本“0 1 2 9”的权值变大，其它被分对的样本的权值变小。

f3(x)=0.4236h1(x) + 0.6496h2(x)+0.7514h3(x)

第三步计算过程如下



此时，得到的第三个基本分类器sign(f3(x))在训练数据集上有0个误分类点。至此，整个训练过程结束。

    G(x) = sign[f3(x)] = sign[ a1 \* h1(x) + a2 \* h2(x) + a3 \* h3(x) ]，将上面计算得到的a1、a2、a3各值代入G(x)中，得到**最终的分类器**为：H(x) = sign[f3(x)] = sign[ 0.4236h1(x) + 0.6496h2(x)+0.7514h3(x) ]。

* 1. AdaBoost 算法的训练误差分析

1.4.1AdaBoost算法的最终分类器的训练误差界为

 （15）

其中，和分别由公式（6），（5）和（4）给出。

我们首先分析等号左边的式子，表示的是一个指示函数，即若分类器能够将样本区分开，则=0，否则=1；而对于来说，当分类器能区分开样本时是一个大于0的数（和同符号），则有>1，即>;当分类器不能区分开样本时，是一个小于0的数，但仍然有>0，即>。通过上面的分析，公式（15）左边的部分得到了证明。下面我们来证明右边的部分。

因为有所样本的初始化值都为，也就是说，将其带入公式（15）的中间位置的公式得到

 （16）

将公式（5）带入公式（16）进一步得到

 （17）

根据公式（4），我们可以得到，当m=1时，有

，观察求和公式右边前两项可以得到：

（18）

同理，我们可以将连乘符号中m=2拿出来和组合，得到Z2，一次类推，我们可以得到成立，而是使得公式（15）中，等号部分得证。

通过公式（15），我们得到了adaboost算法的误差的上届，而且这公式说明，可以在每一轮选取最适当的hm使得Zm最小，从而使训练误差下降最快。

下面我们来讨论在二分类问题中，的表达形式和上界。根据Zm的定义有

 （19）

而的取值只有-1和+1，因此根据能够将样本正确区分开，有

 （20）

而将公式（3）带入公式（20）得到

 （21）

从而得到：

 （22）

将带入公式（22）得到

 （23）

又因为对两边开方得到带入公式（23）得到

 （24）

若存在一个，使得对于任意满足，则有

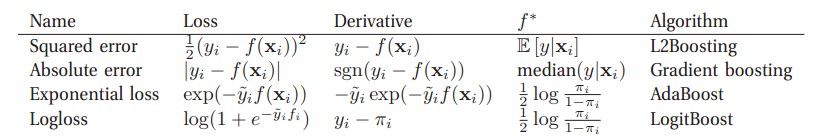
 （25）

将公式（25）带回到公式（15）中得到：

 （26）

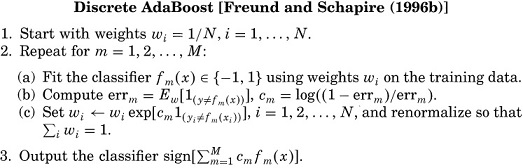
通过公式（25）说明，随着组合弱分类器数量M的增加，组合出的请分类器的性能随着指数下降，这使得该算法具有非常大的吸引力。

1.5 Boosting家族

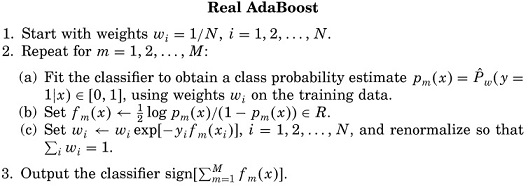
在1.2小节中，我们介绍了adaboost算法的基本流程，而在1.3小节中我们通过损失函数的概念推导出了adaboost算法的数学原理，从本质上看，adaboost将可以用指数损失函数定义，而在每次添加的弱分类器的权重是通过最小化指数损失函数得到。一般将上述adaboost称之为Discrete boost算法，它表示一种基于指数损失函数的二分类问题。当然我们可以通过不通过的损失函数得到不同的boosting算法，将这种基于梯度最小化方法得到boosting方法称之为GradientBoosting算法， 

不同Gradient Boosting算法的计算流程：

Discrete Adaboost算法的步骤如下：

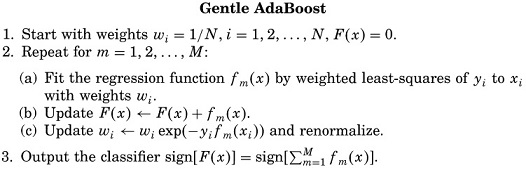


　　可以看出，Discrete AdaBoost的每一个弱分类的输出结果是1或-1，并没有属于某个类的概率，略显粗糙。如果让每个弱分类器输出样本属于某个类的概率，则可以得到Real AdaBoost算法，其步骤如下：



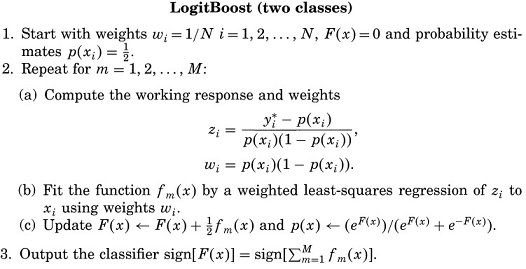
　　Real Adaboost每个弱分类器输出样本属于某类的概率后，通过一个对数函数将0-1的概率值映射到实数域，最后的分类器是所有映射函数的和。

将Real Adaboost算法每次迭代的两部合并，直接产生一个映射到实数域的函数，则就成了Gentle AdaBoost， 其算法步骤如下：



　　Gentle AdaBoost则在每次迭代时，基于最小二乘去做一个加权回归，最后所有回归函数的和作为最终的分类器。

　　LogitBoost算法则和Gentle AdaBoost算法有点相像，不过其每次进行回归拟合的变量z是在不断更新的，Gentle AdaBoost使用的是y。LogitBoost算法步骤如下：



最后总结一下GradientBoosting算法的基本流程：

