Boosting 理论基础*)

涂承胜[1,2] 陆玉昌[2]

(重庆三峡学院计算机科学系 重庆 404000)[1]

(清华大学计算机科学技术系 智能技术与系统国家重点实验室 北京 100084)[2]

摘 要 Boosting 是提高学习算法准确度的有效方法。本文主要介绍了 Boosting 的问题框架 PAC 模型、与 Boosting 相似并有助于 AdaBoost 研究的在线分配模型和 AdaBoost 算法，并对 AdaBoost 算法的参数和弱假设选择等进行了分析。

关键词 PAC 学习模型，在线分配模型，算法分析

The Theory Base of Boosting

TU Cheng-Sheng[1,2] LU Yu-Chang[2]

(Dept. of Computer Science, Chong Qing Three Gorges College, ChongQing 404000)[1]

(Computer Science and Technology Dept., Tsing Hua University The State Key Laboratory of Intelligent Technology and System, Beijing 100084)[2]

Abstract Boosting is an effective method for improving thd accuracy of any given learning algorithm, which generate multiple versions of a hypothesis and combine them to create an aggregate hypothesis, This paper first introduces the problem framework of Boostinig: PAC learning model, and Online prediction model, a similar algorithm with boosting but with great help for boosting' s research. Besides, if also introduces and analysis the algorithm of adaboost itself, as well as its parameters and week hypotheses' selection.

Keywords PAC learning model, Online prediction model, Algorithm analysis

1 概述

Boosting 由 Freund 和 Schapire 于 1990 年提出[1]，是提高预测学习系统预测能力的有效工具之一，也是组合学习中最具代表性的方法，它试图提供一种提升任意学习算法精度的普遍方法。Kearns 和 Valiant[2,3]指出：在 Valiant 的 PAC 模型[4]中，若存在一个多项式级的学习算法以识别一组概念，且识别的正确率很高，那么这组概念是强学习的；如果学习算法识别一组概念的正确率仅比随机猜测略好，那么这组概念是弱学习的。Kearns 和 Valiant 提出了弱学习算法与强学习算法的等价问题：在 PAC 模型中一个"弱"学习器是否能被"提升"为一个具有任意精度的"强"学习算法[5,6]？如果两者等价，那么在学习概念时，只要找到一个比随机猜测略好的学习算法，就可以将其提升为强学习算法，而不必直接去找通常情况下很难获得的强学习算法。为此，1989 年 Schapire 提出了第一个可证明的多项式时间 Boosting 算法[1]，对此问题作了肯定回答。之后，Freund 设计了一个更有效的通过重取样或过滤运作的 Boost-by-majority 算法[7]。在某种意义上该算法是优化的，但实践上却有一些缺陷：它们要求事先知道弱学习算法正确率的下限，这在实际问题中难以实现。

1995 年 Freund 和 Schapire[7]介绍了通过调整权重而运作的算法：AdaBoost (Adaptive Boost)，AdaBoost. M1，AdaBoost. M2，AdaBoost. R，解决了早期 Boosting 算法很多实践上的困难[8]。为解决类别数很大时的多类问题，1997 年 Schapire 和 Singer 提出了 AdaBoost. M2 与 ECOC 算法[9](由 Dietterich 和 Bakiri 提出)结合的 AdaBoost. OC[10]算法。1998 年 Schapire 和 Singer 提出了 AdaBoost 的泛化形式，并引入自信率预测以改善 Boosting 的性能。他们还提出了基于汉明距离和损失排序的多类多标签 Boosting 算法 AdaBoost. MH，AdaBoost. MR，并介绍了 AdaBoost. MH 与 ECOC 结合的另一种 Boost 方法 AdaBoost. MO[11]。1998 年 Friedmand 等提出了被称之为"Gentle AdaBoost"的 AdaBoost 之变种算法[12]，它较少地强调野点(Outliers，训练样本中被标错或本身就难以分类的样本)。1999 年 Freund 介绍了 Boost-by-majority 算法的一个自适应扩展版本 BrownBoost，它采用了不强调野点[13]的方法。

AdaBoost 是 Boosting 家族中的基础算法。Boosting 家族中的大部分扩展(算法)都由它得来，对 AdaBoost 的分析结论也适用于其它的 Boosting 方法。限于篇幅，本文主要介绍 Boosting 的基础理论：Boosting 的问题框架 PAC 模型、与 Boosting 相似并有助于 AdaBoost 研究的在线分配模型和 AdaBoost 算法，对 AdaBoost 算法的参数和弱假设选择等进行了分析。对于 AdaBoost 的泛化错误及其与结构风险最小化、VC 维、支持向量机及 margin 理论的关系等理论将另文介绍。

2 PAC 学习模型

PAC (Probably Approximately Correct) 学习模型是 Boosting 学习方法的基础框架。PAC 模型：设 X 是一个称为"域"的集合。一个概念是一个布尔函数 $c: X \to \{0,1\}$。一个概念类 C 是概念的集合。学习者接受带类别标签的例子 $(x, c(x))$，其中 x 是按照域 X 中一些确定但未知的任意分布 D 随

*)资助项目：重庆市教委科技项目(编号：031104)资助。中国国家重点基础研究发展项目"973 项目"(编号：G1998030414)资助。涂承胜 副教授，清华大学访问学者，研究方向为数据采掘与知识发现，机器学习。陆玉昌 教授，研究方向为数据采掘与知识发现，机器学习，知识工程。

机选择的，且 $c\in C$ 是目标概念。一段时间后，学习者必须输出一个假设 $h: X\to[0,1]$。$h(x)$的值可被理解成对 χ 的标签的随机预测：以概率 $h(x)$ 为 1，以概率 $1-h(x)$ 为 0(这也可以扩展到 h 是介于{0,1}的随机映射的情况)。假设 h 的错误是期望值 $E_{x\sim D}(|h(x)-c(x)|)$，其中 x 是按 D 选择的。若 $h(x)$被理解为一个随机预测，则该错误就是非正确预测的概率。

强 PAC 学习算法：给定 $\varepsilon,\delta>0$ 和随机的例子，以概率 $1-\delta$ 输出一个最大错误为 ε 的假设。并且，运行时间是 $1/\varepsilon$，$1/\varepsilon$ 和其它参数(如接受例子的大小，目标概念的大小和复杂度)的多项式。

弱 PAC 学习算法：满足与强 PAC 学习算法一样的条件，但 $\varepsilon\geqslant 0.5-\gamma$，其中 $\gamma>0$ 要么是一个常数，要么以 $1/p$ 减小，其中 p 是相关参数中的一个多项式。其中 γ 称为"边"。1990 年 Schapire 指出，高效的 PAC 学习算法可以被转换为高效的 PAC 强学习算法。他提出的 PAC 转换算法是：从 3 个不同的分布中产生 3 个假设，预测时让这 3 个假设进行多数投票。具体地说，分布 $D_1=D$ 为原始分布，得出假设 h_1；分布 $D_2=$选择使 $P_{x\sim D_2}[h_1(x)=c(x)]=P_{x\sim D_2}[h_1(x)\neq c(x)]=1/2$ 的例子，得出假设 h_2；$D_3=$从 D 中选出，去除那些 $h_1(x)=h_2(x)$的例子，得出假设 h_3。其中 $c(x)$为从实例 x 到标签集的映射。最终假设 $h_{final}(x)=MAJ[h_1(x),h_2(x),h_3(x)]$。此处使用递归以实现之。可以证明，如果 $error_{D_1}(h_1),error_{D_2}(h_2),error_{D_3}(h_3)\leqslant\varepsilon$，则 $error_D(h_{final}(x))\leqslant 3\varepsilon^2-2\varepsilon^3$。1990 年 Freund 提出 Boost-by-majority，可以不用递归而组合任意数目的弱假设。开始假设错误为 $1/2-\gamma$，要想使错误达到 ε，必须组合的假设数目为：$T=O(\frac{1}{\gamma^2}\ln\frac{1}{\varepsilon})$。这些早期 Boosting 算法的问题是：进一步的改善需要 Boosting 的更多迭代；算法和边界依赖于最坏情况分布下弱学习器的错误，对于此我们事先并不知道；甚至在该错误预先知道的情况下，如果初始错误小，则迭代回数$=O(\frac{1}{\gamma^2})$并无好处；实践中最多迭代 7 回是可行的。AdaBoost 则在一定程度上避免了这些问题。

3 在线预测模型

AdaBoost 算法是由在线分配算法导出的。在线分配算法 $Hedge(\beta)$来自 Littlestone 和 Warmuth 的 WMA(Weighted majority algorithm)的简化。分配代理(Distribution agent)A 有 N 个选择或策略，标号为 $1,\cdots,N$。迭代序号 $t=1,2,\cdots,T$。A 决定这些策略上的分布 P^t。$P_i^t\geqslant 0$，且 $\sum_{i=1}^{N}P_i^t=1$。每个策略的损失为 l_i^t。则 A 总损失为 $\sum_{i=1}^{N}p_i^t l_i^t=P^t\cdot l^t$，称之为综合损失。不失一般性，假定任何策略所受的损失均限于 $l_i^t\in[0,1]$。除此以外，没有作任何假设。算法 A 的目的是最小化相对于最佳策略损失的累计和，即 A 尽量最小化它的净损失 $L_A-\min_i L_i$，其中 $L_A=\sum_{t=1}^{T}P^t\cdot l^t$，$L_i=\sum_{t=1}^{T}l_i^t$。$Hedge(\beta)$的伪码如下所示。

输入：参数：$\beta\in[0,1]$；初始化权重矢量 $W^1\in[0,1]^N$，且 $\sum_{i=1}^{N}w_i^1=1$；迭代次数 T；对 t=1，2，…，T，执行下述步骤：

1. 选择分配 $P^t=W^t/\sum_{i=1}^{N}w_i^t$；
2. 从环境中得到每次"实验"的损失矢量 $l^t\in[0,1]^N$；
3. 所受损失 $P^t\cdot l^t$；
4. 设置新的权重矢量 $w_i^{t+1}=w_i^t\beta^{l_i^t}$。

如果无偏好，w_i^1。权重更新规则可变为：$w_i^{t+1}=w_i^t\cdot U_\beta(l_i^t)$，其中 $U_\beta:[0,1]\to[0,1]$是任意函数，参数 $\beta\in[0,1]$，对所有的 $\gamma\in[0,1]$满足 $\beta^\gamma\leqslant U_\beta(\gamma)\leqslant 1-(1-\beta)\gamma$。

定理：对损失矢量的任意序列 $l^1,\cdots,l^T$，对任意 $i\in\{1,\cdots,N\}$有 $L_{Hedge(\beta)}\leqslant\frac{-\ln(w_i^1)-L_i\ln\beta}{1-\beta}$。更一般地，对任意非空集合 $S\subseteq\{1,\cdots,N\}$，有 $L_{Hedge(\beta)}\leqslant\frac{-\ln(\sum_{i\in S}w_i^1)-(\ln\beta)\max_{i\in S}L_i}{1-\beta}$。

算法 $Hedge(\beta)$和 AdaBoost 有明显的相似性。这种相似性反映了在线分配模型和 Boosting 问题之间惊人的对偶关系。也可以说，Boosting 问题可以直接映射或约简为在线分配问题。实际上，AdaBoost 修改了在线分配模型以提高弱学习算法的性能。"策略"对应于 AdaBoost 的"例子"，而"实验"对应于 AdaBoost 的弱假设。它们的区别是：在 $Hedge(\beta)$中若策略成功则权重增加，而 AdaBoost 中同例子相关的权重是假若例子"难"则增加。两个算法的主要技术差别是：AdaBoost 中的参数 β 不再是提前确定的，而是根据 ε_t 每回都在变化。对于一些 $\gamma>0$ 和所有的 $t=1,\cdots,T$，若提前给定 $\varepsilon_t\leqslant 1/2-\gamma$，则我们直接应用 $Hedge(\beta)$算法及其分析：固定 β 为 $1-\gamma$，设 $l_i^t=1-|h_t(x_i)-y_i|$，h_f 是 AdaBoost 的，但为所有的 T 个假设指定相等的权重。则 $P^t\cdot l^T$ 恰好是分布为 P^t 的 h_t 的精度，根据假设，它最小是 $1/2+\gamma$。同样，设 $S=\{i: h_f(x_i)\neq y_i\}$，若 $t\in S$，则根据 h_f 的定义及 $y\in\{0,1\}$，有 $\frac{L_i}{T}=\frac{1}{T}\sum_{t=1}^{T}l_i^t=1-\frac{1}{T}\sum_{t=1}^{T}|y_i-h_t(x_i)|=1-|y_i-\frac{1}{T}\sum_{t=1}^{T}h_t(x_i)|\leqslant 1/2$。根据上述定理，有

$$T\cdot(1/2+\gamma)\leqslant\sum_{t=1}^{T}P^t\cdot l^t\leqslant\frac{-\ln(\sum_{i\in S}D(i))+(\gamma+\gamma^2)(T/2)}{r}。$$

这表明，h_f 的错误 $\varepsilon=\sum_{i\in S}D(i)$，最大是 $e^{-T\gamma^2/2}$。AdaBoost 比直接应用 $Hedge(\beta)$有两个优点：①通过更精练的分析和选择 β，得到关于错误的更优的边界；②算法不需要弱假设精度的先验知识。取而代之的是它在每次迭代中衡量 h_t 的精度，并设置相应的参数。更新因子 β_t 随着 ε_t 而减小，ε_t 是使分布 P^t 和 P^{t+1}之间的差别增加。减小 β_t 也就是增加权重 $\ln(1/\beta_t)$，它是同最终假设中的 h_t 相关的。这就导致：越精确的假设会使产生的分布的变化越大，并且更能影响最终假设的输出。

4 AdaBoost 算法及其参数选择

AdaBoost 算法是 Boosting 家族的代表算法，其目的是根据给定的例子 x 预测标签 y，使用 Boosting 是为了找出假设 $H(x)$，其伪代码如下 4.1 算法所示。算法的输入为训练集 $S=((x_1,y_1),\cdots,(x_m,y_m))$。每个成员都是带标签的训练例。$x_i\in X$，$X$ 代表领域或实例空间。$y_i\in Y$(Y 为标签集)。学习器接受的例子(x_i,y_i)是从分布为 P 的 $X\times Y$ 上随机的选择。假定是两类问题，$Y=\{-1,+1\}$。它是多类问题扩展的基础。Adaboost 反复调用给定的弱或基学习算法，其主要思想之一是在训练集中维护一套权重分布。在第 t($t=1,\cdots,T$，T 为迭代次数，可以由某一算法给定)回迭代时样本(x_i,y_i)上的分布权值记为 $D_t(i)$。初始时，所有例子的权重都设为相等(即 1/m)。但每一回错分的实例其权重将增加，以使弱学习器被迫集中在训练集中的难点(实例)上。弱学习器的任务就是根据分布 D_t 找到合适的弱假设 $h_t: X\to R$。最简单的情况下每个 h_t 的范围是二值的：$\{-1,+1\}$。于是该学习器的任务就是最

小化错误 $\varepsilon_t=\Pr_{i\sim D_t}[h_t(x_i)\neq y_i]$。一旦得到 h_t，AdaBoost 选择一个参数 $\alpha_t\in R$，该参数直观测量 h_t 的重要程度。最终假设 H 是 T 次循环后，用加权多数投票把 T 个弱假设之输出联合起来得到的。对二值 h_t，典型地，设：$\alpha_t=\frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$。

4.1 算法

输入：m 个带标记实例的序列$\langle(x_1,y_1),\cdots,(x_m,y_m)\rangle$，其中 $x_i\in X,y_i\in Y=\{-1,+1\}$。$m$ 个实例上的分布 D，弱学习算法 WeakLearn，迭代次数 T。

初始化：$D_1(i)=1/m$，对 $t=1,\cdots,T$ 循环执行：

①用分布 D_t 训练弱学习器；

②得到弱假设 $h_t: X\to R$；

③选择 $\alpha_t\in R$；

④修改：$D_{t+1}(i)=\frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$。其中 Z_t 是归一化因子(使 D_{t+1}为分布)；

输出：$H(x)=Sign(\sum_{t=1}^{T}\alpha_t h_t(x))$。

这个算法是加入了自信率预测的 AdaBoost 算法，即泛化的 AdaBoost 算法。把 $h(x)$的符号理解成赋给实例 x 的预测标签(−1 或+1)，而其大小 $|h(x)|$作为该预测的自信度。因此，若 $h(x)$离 0 越远则自信度越高。该算法和原始 AdaBoost 算法的主要区别是：弱假设可以定义在整个 R 上而不是被限制在$[-1,+1]$范围内(尽管有时我们也确实需要限制这个范围)；不象原始 AdaBoost 算法那样指明 α_t 的选择，以留下可讨论的余地。例如，可以使 α_t 随分类自信度而变化。

设 $f(x)=\sum_{t=1}^{T}\alpha_t h_t(x)$。于是 $H(x)=sign(f(x))$。可以证明，H 的训练错误的上界为：$\frac{1}{m}|\{i: H(x_i)\neq y_i\}|\leqslant\prod_{t=1}^{T}Z_t$。为了最小化训练错误，一个合理的方法可能是通过在 Boosting 的每一回迭代上最小化 Z_t 以贪婪地最小化上述边界。这个思想可以用于选择 α_t 和作为弱假设 h_t 选择的一个普遍标准。

从另一个角度看 AdaBoost，设 $\mathfrak{h}=\{g_1,\cdots,g_N\}$为所有可能弱假设的空间。为简化起见暂时假设它是有限的。AdaBoost 试图找到这些弱假设的一个线性阈值以给出好的预测，即找到以下形式的函数：$H(x)=sign(\sum_{j=1}^{N}a_j g_j(x))$。可以看出 H 的训练错误数目最多是：$\sum_{i=1}^{m}\exp(-y_i\sum_{j=1}^{N}a_j g_j(x_i))$。AdaBoost 可被看作是在系数 a_j 上通过沿坐标轴贪婪地搜索以最小化这个表达式的一种方法。在每一回迭代 t 上，相应于 h_t 的坐标 j 被选出，即 $h_t=g_j$。下一步，通过 α_t 修改系数 a_j；所有其它系数不变。可以证明 Z_t 确实度量了 $\sum_{i=1}^{m}\exp(-y_i\sum_{j=1}^{N}a_j g_j(x_i))$中指数和的新旧值之比，使得 $\prod_t Z_t$ 为该表达式的最终值(假定所有 a_j 开始都是 0)。

4.2 选择 α_j

为简化表达，固定 t，并让 $u_i=y_i h_t(x_i)$，$Z=Z_t$，$D=D_t$，$h=h_t$ 和 $\alpha=\alpha_t$。为不失一般性，假设对所有 i，$D(i)\neq 0$。目标是找到 α，它可以最小化或近似最小化为 α 函数的 Z。

4.2.1 导出原始 AdaBoost 中对 α_t 的选择

原始 AdaBoost 中 $\alpha_t=\frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$。它是此处将要推导出的新版本的一种特殊情况。对有范围为$[-1,+1]$的弱假设 h，其 α 的选择可以通过如下形式近似 Z 而获得：$Z=\sum_i D(i)e^{-\alpha u_i}\leqslant\sum_i D(i)(\frac{1+u_i}{2}e^{-\alpha}+\frac{1-u_i}{2}e^{\alpha})$。该不等号成立，因为 $u_i\in[-1,+1]$，如果 h 有范围$\{-1,+1\}$(使得 $u_i\in\{-1,+1\}$)时等号成立。该上边界由 $e^{-\alpha u}$对任意常数 $\alpha\in R$ 的凸性导出。如果选择 $\alpha=\frac{1}{2}(\frac{1+r}{1-r})$来最小化上述不等式的右边，其中 $r=\sum_i D(i)u_i$。于是有 $Z\leqslant\sqrt{1-r^2}$。可以证明，对 $h_t\in[-1,+1]$，选择 $\alpha_t=\frac{1}{2}(\frac{1+r_t}{1-r_t})$，其中 $r_t=\sum_i D(i)y_i h_t(x_i)=E_{i\sim D_t}[y_i h_t(x_i)]$。于是 H 的训练错误最多为 $\prod_{t=1}^{T}\sqrt{1-r_t^2}$。这就应该在 Boosting 的每回迭代中寻找最大化$|r_t|$的 h_t。如果 h_t 有范围$\{-1,+1\}$，则 $\Pr_{i\sim D_t}[h_t(x_i)\neq y_i]=\frac{1-r_t}{2}$；更一般地，如果 h_t 有范围$[-1,+1]$，则$\frac{1-r_t}{2}$与原始 AdaBoost 中定义的错误 ε_t 等价。

4.2.2 一个一般情况下的数字方法

此处给出一个关于 α 的一般数字方法以最小化 Z。Z 的一阶导是 $Z'(\alpha)=-Z\sum_i D_{t+1}(i)u_i$。因此，如果 D_{t+1}用最小化 Z_t(使 $Z'(\alpha)=0$)的 α_t 值形成，则有 $\sum_i D_{t+1}(i)u_i=E_{i\sim D_{t+1}}[y_i h_t(x_i)]=0$。也就是说，对分布 D_{t+1}，h_t 与标签 y_i 完全无关。可以很容易地验证 $Z''(\alpha)=\frac{d^2Z}{d\alpha^2}$对所有 $\alpha\in R$ 严格为正(去除对所有 i，$u_i=0$ 的平凡情况)。因此，$z'(\alpha)$最多有一个 0。进而，如果存在 i 使 $u_i<0$，则随 $\alpha\to\infty$有 $Z'(\alpha)\to\infty$；如果存在 i 使 $u_i>0$，则随 $\alpha\to-\infty$有 $Z'(\alpha)\to-\infty$。这就意味着 $Z'(\alpha)$至少有一个根，除所有非 0 的 u_i 都有相同的符号的退化情况外。总之，因为 $Z'(\alpha)$严格递增，我们可以数值地找到 $Z(\alpha)$惟一的最小值。

4.2.3 可拒绝的弱假设的分析方法

现在假定每个弱假设 h_t 的范围被限制在$\{-1,0,+1\}$。预测"0"表示拒绝决策。定义：$w_b=\sum_{i:u_i=b}D(i)$对 $b\in\{-1,0,+1\}$，其中 $u_i=y_i h_t(x_i)$。继续省略 t，并用 W_+和 W_-分别代替 $W_{+1}W_{-1}$。Z 因此可以被写作 $Z=W_0+W_-e^{\alpha}+W_+e^{-\alpha}$。可以很容易地证明当 $\alpha=\frac{1}{2}\ln(\frac{W_+}{W_-})$时，$Z=W_0+2\sqrt{W_-W_+}$被最小化。原始 AdaBoost 算法其选择更保守：$\alpha=\frac{1}{2}\ln(\frac{W_++\frac{1}{2}W_0}{W_-+\frac{1}{2}W_0})$。若 $W_0=0$，则二者相同。

4.2.4 寻找弱假设的标准

在 4.2.1 中导出的错误边界还可以指导弱学习算法的设计。过去，弱学习算法的目标是对给定分布 D_t 找到仅有少量错误的弱假设 h_t。然而，从上面的分析中可以看出，实际上可以有不同的标准。特别地，我们可以在每一回迭代中最小化 Z_t 以贪婪地最小化错误的上边界。不失一般性，对任意常数 $\alpha\in R$，任意弱学习器可以用它随意伸缩任意弱假设 h。所以 Z 可以写作 $Z=\sum_i D(i)\exp(-y_i h(x_i))$。其目的就是要最小化它。某些算法可以很容易地被修改以直接最小化这类的"损失"函数，例如基于梯度的方法，如 BP，可以很容易地最小化 Z，这比最小化均方误差容易得多。下面将展示决策树算法在基于上述准则被修改的情况下如何找到好的弱假设。现集中于其预测基于划分领域 X 的弱假设。具体地，每个这样的弱假设和把 X 划分到不相交块 X_1，$\cdots$，X_N 的划分相联系。这些块覆盖所有 X 且对所有 $x,x'\in X_j$ 有 $h(x)=h(x')$。简言之，h 的预测仅依赖于某给定实例落入的块 X_j。决策树就是这种假设的一个经典例子，其叶子定

义了领域的一个划分。假定 $D=D_t$ 且已经找到了空间上的一个划分 $X_1,\cdots,X_N$。对每个划分块应该作什么样的预测？即对于给定划分，如何找到最小化 Z 的函数 $h:X\to R$？设 $c_j=h(x)$ 对 $x\in X_j$。目的是找到 c_j。对每个 j 和 $b\in\{-1,0,+1\}$，设 $W_b^j=\sum_{i:x_i\in X_j\wedge y_i=b}D(i)=Pr_{i\sim D}[x_i\in X_j\wedge y_i=b]$ 是标签为 b 落入块 j 的例子的部分权之和。Z 可以被重写为：$Z=\sum_j\sum_{i:x_i\in X_j}D(i)\exp(-y_ic_j)=\sum_j(W_+^je^{-c_j}+W_-^je^{c_j})$。通过标准计算得出，当 $c_j=\frac{1}{2}\ln(\frac{W_+^j}{W_-^j})$ 时它被最小化，此时

$$Z=2\sum_j\sqrt{W_+^jW_-^j}\qquad(1)$$

c_j 的符号等于块 j 中的加权多数类。更进一步，如果块 j 中有正负样本基本相等的划分，则 c_j 将接近于 0(低的预测自信度)。(1)式中的准则也可在生成决策树时作为分裂准则，代替 Gini Index 或熵函数准则。即决策树可以通过贪婪选择导致(1)式中 Z 值最大下降的分裂建立起来。进而，如果需要 Boost 多个决策树，则每棵树都可以通过上述分裂准则，叶子上的预测则由上面的 c_j 给出。上面的方案有可能出现这样的情况：W_-^j 或 W_+^j 可能会很小或甚至是 0，这种情况 c_j 下的值将会很大甚至无限。实践中，这种大的预测可能导致数值问题。另外，在理论上也有理由怀疑大的过于自信的预测将增加过份学习(Over fitting)的趋势。因此，对于某些合适的小正值 $\varepsilon(\varepsilon>0)$，建议使用"平滑"的 c_j 值：$c_j=\frac{1}{2}\ln(\frac{W_+^j+\varepsilon}{W_-^j+\varepsilon})$。因为 W_-^j 和 W_+^j 都在 0 和 1 之间，通过 $\frac{1}{2}\ln(\frac{1+\varepsilon}{\varepsilon})\approx\frac{1}{2}\ln(\frac{1}{\varepsilon})$ 来约束 $|c_j|$ 是有效的。可以推出，此时 $Z\leqslant 2\sum_j\sqrt{W_-^jW_+^j}+\sqrt{2N\varepsilon}$。可以看出，如果选择 $\varepsilon\ll 1/(2N)$，平滑并未降低准则的质量。一般地，可取 ε 为 $1/m$ 阶。

结论 AdaBoost 具有快，简单，易于编程的优点。除了迭代次数 T 外不需调整参数；不需要弱学习器的先验知识，因此可以灵活地和任意方法结合寻找弱假设；给定足够数据和一个能够可靠地仅仅提供中等精度的弱学习器，它可以提供学习的一套理论保证。这是学习系统设计思想的一个转变：不是试图设计一个在整个空间都精确的学习算法，而是集中于寻找仅比随机预测好的弱学习算法。其缺点：AdaBoost 在特定问题上的实际性能很明显依赖于数据和弱学习器；与理论相一致，AdaBoost 在不足的数据集，过于复杂的或太弱的弱假设上性能不好；它看起来对噪声还很敏感。

参考文献

1 Schapire R E. The strength of weak learnability. Machine Learning, 1990,5(2):197～227
2 Kearns M, Valiant L G. Learning Boolean Formulae or Factoring:[Technical Report TR－1488]. Cambridge, MA:Havard University Aiken Computation Laboratory, 1988
3 Kearns M, Valiant L G. Crytographic Limitation on Learning Boolean Formulae and Finite Automata. In: Proc. of the 21st annual ACM Symposium on Theory of Computing, New York. NY: ACM press, 1989. 433～444
4 Valiant L G. A theory of the learnable. Communications of the ACM, 1984, 27(11):1134～1142
5 Kearns M J, Vazirani L G. Learning Boolean formulae or finite automata is as hard as factoring:[Technical Report TR－14－88]. Harvard University Aiken Computation Laboratory, Aug. 1988
6 Kearns M J, Vazirani L G. Cryptographic limitations on learning Boolean formulae and finite automata. Journal of the Association for Computing Machinery, 1994, 41(1):67～95
7 Freund Y. Boosting a weak learning algorithm by majority. Information and computation, 1995, 141(2):256～285
8 Freund Y, Schapire R E. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Science, 1997, 55(1):119～139
9 Dietterich T G, Bakiri G. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 1995, 2:263～286
10 Schapire R E, Singer Y. Using output codes to boost multiclass learning problems. In: Machine Learning: Proc. of the Fourteenth Intl. Conf. 1997. 313～321
11 Schapire R E, Singer Y. Improved boosting algorithms using confidence-related predictions. In: Proc. of the eleventh Annual Conf. on Computational Learning Theory, 1998. 80～91
12 Friedman J, Hastie T, Tibshirani R. Additivelogistic regression: a statistical view of boosting:[Technical Report]. 1998
13 Freund Y. An adaptive version of the boost by majority algorithm. In: Proc. of the Twelfth Annual Conf. on Computational Learning Theory, 1999

(上接第 10 页)

37 Ontolingua. http://www-ksl. stanford. edu/knowledge-sharing/ontolingua/
38 CycL. http://www. cyc. com/tech. html #cycl
39 LOOM. http://www. isi. edu/isd/LOOM/LOOM-HOME. html
40 F-Logic. http://www. informatik. uni-freiburg. de/~dbis/Publications/95/flogic-jacm. html
41 XOL. http://www. ai. sri. com/~pkarp/xol/
42 Concept Graph. http://www. jfsowa. com/cg/
43 Gomez-Perez A, Fernandez M, De Vincente A J. Towards a Method to Conceptualize Domain Ontologies, ECAI-96 Workshop on Ontological Engineering, Budapest, 1996
44 Fernandez M, Gomez-perez A, Juristo N. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. AAAI-97 Spring Symposium on Ontological Engineering, Stanford University, March 1997
45 CommonKADS. http://www. commonkads. uva. nl/
46 KACTUS. http://www. swi. psy. uva. nl/projects/Kactus/toolkit/intro. html
47 SENSUS. http://www. isi. edu/natural-language/resources/sensus. html
48 ONIONS. http://saussure. irmkant. rm. cnr. it/onto/onions. html
49 (KA)². http://ka2portal. aifb. uni-karlsruhe. de/
50 OntoText. http://www. ontotext. com/index. html
51 Text-To-Onto. http://ontoserver. aifb. unikarlsruhe. de/text-toonto/
52 GATE. http://www. ontotext. com/gate
53 Ontosaurus. http://www. isi. edu/isd/ontosaurus. html
54 Blaxquez M, Fernandez M, Garcia-Pinar J M, Gomez-Perez. Building Ontologies at the Knowledge Level Using the Ontology Design Environment. Downloadable from http://ksi. cpsc. ucalgary. ca/KAW/KAW98/blazquez/index. html.
55 EXPECT. http://www. isi. edu/expect/
56 WebOnto. http://kmi. open. ac. uk/projects/webonto
57 On-To-Knowledge. http://www. ontoknowledge. org/index. shtml
58 OntoWeb. http://www. ontoweb. org/
59 OntoBroker. http://ontobroker. aifb. uni-karlsruhe. de/
60 Jin Z, Bell D, Wilkie F, Leahy D. Automatically Acquiring Requirements of Business Information Systems by Reusing Business Ontology. In: Proc. of ECAI'98 Workshop on Applications of Ontology and Problem Solving Methods, Brighton, UK, 1998
61 金芝. 基于本体的自动需求获取. 计算机学报, 2000(5)
62 Berners-Lee T, Hendler J, Lassila O. The Semantic Web, Scientific American, 2001
63 Hahn U, Romacker M. Content Management in the SYNDIKATE System - How Technical Documents are Automatically Transformed to Text Knowledge Bases. Data & Knowledge Engineering, 2000, 35(2): 137～159
64 Dahlgren K. A linguistic Ontology. International Journal of Human-Computer Studies, 1995, 43(5)
65 Sure Y, Maedche A, Staab S. Leveraging Corporate Skill Knowledge - From Proper to OntoProper. In Mahling D, Reimer U, eds. Proc. of the Third Intl. Conf. on Practical Aspects of Knowledge Management. Basel, Switzerland, Oct. 2000
66 Angele J, Schnurr H-P, Staab S, Studer R. The Times they are a-changin'-- the Corporate History Analyser. In: Mahling D, Reimer U, eds. Proc. of the Third Intl. Conf. on Practical Aspects of Knowledge Management. Basel, Switzerland, Oct. 2000
67 Staab S, Schnrr H-P, Studer R, Sure Y. Knowledge Processes and Ontologies. IEEE Intellignece Systems, 2001, 16(1)