计算机科学 2003Vol. 30№. 3

Boosting 家族 AdaBoost 系列代表算法

涂承胜[1] 刁力力[2] 鲁明羽[2,3] 陆玉昌[3]

(重庆三峡学院计算机科学系 重庆万州 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]

(烟台大学计算机学院 烟台 264005)[3]

The Typical Algorithm of AdaBoost Series in Boosting Family

TU Cheng-Sheng[1] DIAO Li-Li[2] LU Ming-Yu[2,3] LU Yu-Chang[3]

(Dept. of Computer Science, Chongqing Three Gorges College, Chongqing Wanzhou 404000)[1]

(Computer Science and Technology Dept., TsingHua University, The State Key Laboratory of Intelligent Technology and System, Beijing China, 100084)[2,3]

(Computer School, Yantai University, Yantai 264005)[3]

Abstract Boosting is one of the most representational ensemble prediction methods. It can be divided into two series: Boost-by-majority and Adaboost. This paper briefly introduces the research status of Boosting and one of its serials—AdaBoost, analyzes the typical algorithms of AdaBoost.

Keywords Data mining, Machine learning, Combining prediction, Algorithms

1 引言

Boosting 由 Freund 和 Schapire 于 1990 年提出[4]，是提高预测学习系统预测能力的有效工具，也是组合学习中最具代表性的方法。其代表算法可分为 Boost-by-majority 和 AdaBoost 两个系列。Boosting 操纵训练例子以产生多个假设，从而建立通过投票结合的预测器集合。AdaBoost 在训练例子上维护一套概率分布，在每一回迭代中 AdaBoost 在每个例子上调整这种分布，成员分类器在训练例子上的错误率被计算出来并以此在训练例子上调整概率分布。权重改变的作用是在被误分的例子上放置更多的权重，在分类正确的例子上减少其权重。通过单个分类器的加权投票建立最终分类器，每个分类器按其在训练集上的精度而加权[6]。

Kearns 和 Valiant 首先提出：在 Valiant 的 PAC 模型[1]中，一个“弱”学习算法是否能被“提升”为一个具有任意精度的“强”学习[2,3]？1989 年 Schapire 提出了第一个可证明的多项式时间 Boosting 算法[4]，对上述问题作了肯定回答。1995 年 Freund 和 Schapire 介绍了通过调整权重而运作的 AdaBoost、AdaBoost. M1、AdaBoost. M2、AdaBoost. R 算法，解决了早期 Boosting 算法很多实践上的问题[6]。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[7]相结合的 AdaBoost. OC 算法[8]。1998 年 Schapire 和 Singer 提出了 AdaBoost 更一般的形式，并引入自信率预测以改善 Boosting 的性能。他们还提出了解决多类多标签问题的 AdaBoost. MH、AdaBoost. MR 算法，介绍了 ECOC 与 AdaBoost. MH 相结合的 AdaBoost. MO 算法[9]。AdaBoost 对野点(Outliers)的识别能力强。但野点数目过多时，过分强调“困难”实例将有损 AdaBoost 的性能。为此，1998 年 Friedmand 等提出了被称之为“Gentle AdaBoost”的 AdaBoost 变种算法[10]，它较少地强调野点。

本文集中介绍了 AdaBoost 系列的典型算法。限于篇幅，Boost-by-majority 系列典型算法另文介绍[13]。就其应用 AdaBoost 算法主要解决了：两类问题、多类单标签问题、多类多标签问题、大类单标签问题、回归问题。它用全部的训练样本进行学习。

2 AdaBoost 系列算法

AdaBoost 算法是 Boosting 家族的代表算法，使用 Boosting 是为了根据给定的例子 x 预测标签 y，找出假设 $H(x)$。算法的输入为训练集 $S=\langle(x_1,y_1),\cdots,(x_m,y_m)\rangle$。每个成员都是带标签的训练例。$x_i\in X$，$X$ 表示领域或实例空间。$y_i\in Y$（Y 为标签集）。学习器接受的例子是从分布为 P 的 $x\times y$ 上随机选择出来的。假定要学习的是两类问题，$Y=\{-1,+1\}$。它是多类问题扩展的基础。AdaBoost 反复调用给定的弱学习或基学习算法，其主要思想是在训练集中维护一套权重分布。在第 t(t=1,…,T，T 为迭代次数，可由某一算法给定)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(1/m)。但是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 $\alpha_t\in R$，该参数直观地测量 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 个弱假设的输出联合起来得到的。对二值 h_t，典型地，设：$\alpha_t=\frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$。

2.1 AdaBoost 算法及其泛化形式

2.1.1 AdaBoost 算法

输入 N 个带标记实例的序列$\langle(x_1,y_1),\cdots,(x_N,y_N)\rangle$，$N$ 个

涂承胜 讲师，清华大学访问学者。刁力力 博士研究生。鲁明羽 博士研究生。陆玉昌 教授。

实例上的分布 D，弱学习算法 WeakLearn，迭代次数 T。

初始化 对 $i=1,2,\cdots,T$，置权重向量 $W_i^1=D(i)$（一般情况下初始分布一致，即 $D(i)=1/N$）。

步骤（对 $t=1,2,\cdots,T$）：

①调整分布 $P^t=W^t/\sum_{i=1}^{n}W_i^t$。

②调用 WeakLearn，传递分布 P^t 给它；返回假设 $h_t:X\to[0,1]$。

③计算 h_t 的错误：$\varepsilon_t=\sum_{i=1}^{N}P_i^t|h_t(x_i)-y_i|$。

④置 $\beta_t=\varepsilon_t/(1-\varepsilon_t)$。

⑤计算新的权重向量：$W_i^{t+1}=W_i^t\beta_t^{1-|h_t(x_i)-y_i|}$。

输出 假设：$h_f(x)=$

$$\begin{cases}1 & \text{若}\sum_{t=1}^{T}(\log\frac{1}{\beta_t})h_t(x)\geqslant\frac{1}{2}\sum_{t=1}^{T}\log\frac{1}{\beta_t}\\ 0 & \text{其它}\end{cases}$$

可以证明，AdaBoost 调用给定的弱学习算法 WeakLearn 时，将产生错误率为 $\varepsilon_1,\cdots,\varepsilon_T$ 的假设。假设每个 $\varepsilon_t\leqslant1/2$，于是最终假设 h_f 的错误 $\varepsilon=\Pr_{i\sim D}[h_t(x_i)\neq y_i]$ 的上边界为：$\varepsilon\leqslant 2^T\prod_{t=1}^{T}\sqrt{\varepsilon_t(1-\varepsilon_t)}$。

2.1.2 *泛化的 AdaBoost 算法*

输入 $((x_1,y_1),\cdots,(x_N,y_N))$，其中 $x_i\in X,y_i\in\{-1,+1\}$。

初始化 $D_1(i)=1/N$。

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

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

泛化 AdaBoost 的算法并引入自信率预测以改善 Boosting 的性能。把 $h(x)$ 的符号理解成赋给实例 x 的预测标签（-1 或+1），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 *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（尽管有时我们确实需要限制这个范围）。可以证明，H 的训练错误的上界为：$\frac{1}{N}|\{i:H(x_i)\neq y_i\}|\leqslant\prod_{t=1}^{T}Z_t$。

2.2 AdaBoost.M1 和 AdaBoost.M2 算法

AdaBoost.M1，AdaBoost.M2 算法用于解决多类单标签问题。每个待分类样本只能属于多个类别中的单个类。定义 $[\![\pi]\!]$，若 π 为真则 $[\![\pi]\!]$ 为 1，否则 $[\![\pi]\!]$ 为 0。

2.2.1 AdaBoost.M1 *算法*

输入 N 个例子的序列 $((x_1,y_1),\cdots,(x_N,y_N))$，标签 $y_i\in Y=\{1,\cdots,k\}$，实例上的分布 D，弱学习算法 WeakLearn，迭代次数 T。

初始化 对 $i=1,2,\cdots,T$，置权重向量 $W_i^1=D(i)$。

步骤（对 $t=1,2,\cdots,T$）：

①整分布 $P^t=W^t/\sum_{i=1}^{N}W_i^t$。

②调用 WeakLearn，传递分布 P^t 给它；返回假设 $h_i:X\to Y$。

③计算 h_t 的错误：$\varepsilon_t=\sum_{i=1}^{N}P_i^t[\![h_t(x_i)\neq y_i]\!]$。若 $\varepsilon_t>1/2$ 则设 $T=t-1$ 并退出循环。

④置 $\beta_t=\varepsilon_t/(1-\varepsilon_t)$。

⑤计算新的权重向量：$W_i^{t+1}=W_i^t\beta_t^{1-[\![h_t(x_i)\neq y_i]\!]}$。

输出 假设：$h_f(x)=\arg\max_{y\in Y}\Sigma(\log\frac{1}{\beta_t})[\![h_t(x)=y]\!]$。

AdaBoost.M1 调用给定弱学习算法 WeakLearn 时，将产生错误率为 $\varepsilon_1,\cdots,\varepsilon_T$ 的假设。假设每个 $\varepsilon_t\leqslant1/2$，于是最终假设 h_f 的错误 $\varepsilon=\Pr_{i\sim D}[h_t(x_i)\neq y_i]$ 的上边界为：$\varepsilon\leqslant 2^T\prod_{t=1}^{T}\sqrt{\varepsilon_t(1-\varepsilon_t)}$。

这是 AdaBoost 最直接的多类扩展。当弱学习器强到在 AdaBoost 产生的困难分布上也能获得合适的高精度时它足以解决多类问题。如果弱学习器不能在这些困难分布上获得至少 50%的精度则该方法将失败。为此，通常采用简化多类问题为多个二值问题加以解决。AdaBoost.M2，AdaBoost.MH，AdaBoost.MR 等算法多采取这种思想解决多类问题。

2.2.2 AdaBoost.M2 *算法*

输入 N 个例子序列 $((x_1,y_1),\cdots,(x_N,y_N))$，标签 $y_i\in Y=\{1,\cdots,k\}$，实例上的分布 D，弱学习算法 WeakLearn，迭代次数 T。

初始化 对 $i=1,2,\cdots,T$，置权重向量 $W_{i,y}^1=D(i)/(k-1),y\in Y-\{y_i\}$。

步骤（$t=1,2,\cdots,T$）：

①置 $W_i^t=\sum_{y\neq y_i}W_{i,y}^t,q_t(i,y)=W_{i,y}^t/W_i^t$　对 $y\neq y_i$，设 $D_t(i)=W_i^t/\sum_{i=1}^{N}W_i^t$。

②调用 WeakLearn，传递分布 D_t 和标签权重函数 q_t 给它；返回假设 $h_t:X\times Y\to[0,1]$。

③计算 h_t 的伪损失：$\varepsilon_t=\frac{1}{2}\sum_{i=1}^{N}D_t(i)(1-h_t(x_i,y_i)+\sum_{y\neq y_i}q_t(i,y)h_t(x_i,y))$。

④置 $\beta_t=\varepsilon_t/(1-\varepsilon_t)$。

⑤计算新的权重向量：$W_{i,y}^{t+1}=W_{i,y}^t\beta_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y))}$。

输出 假设：$h_f(x)=\arg\max_{y\in Y}\Sigma(\log\frac{1}{\beta_t})h_t(x,y)$。

AdaBoost.M2 是 AdaBoost.MR 的一种特殊情况，对每个有正确标签 y_i 的样本 x_i 和每个非正确标签 y（y 表示除 y_i 外的其它 k-1 个类）提出这样的二值问题：对 x_i，其正确标签是 y 还是 y_i？我们用给定的假设 h 来回答这 k-1 个二值问题，即把正确标签 y_i 和非正确标签 y 区分开来。假设 h 是从 $\{0,1\}$ 中取值，如果 $h(x_i,y)=0$ 和 $h(x_i,y_i)=1$，对上述问题的回答是 y_i；若 $h(x_i,y)=1$ 和 $h(x_i,y_i)=0$，对上述问题的回答为 y。若 $h(x_i,y)=h(x_i,y_i)$，则从两者中随机选择一个。在 h 从 $[0,1]$ 中取值时，可以把 $h(x,y)$ 理解为一随机决策（random decision）：选择一随机位 $b(x,y)\in\{0,1\}$，其值等于 1 的概率为 $h(x,y)$，其值等于 0 的概率为 $1-h(x,y)$。于是，选择不正确回答 y 的概率是：

$$\Pr[b(x_i,y_i)=0\wedge b(x_i,y)=1]+\frac{1}{2}\Pr[b(x_i,y_i)=b(x_i,y)]=\frac{1}{2}(1-h(x_i,y_i)+h(x_i,y))$$

如果所有 k-1 个问题的回答是同等重要，则把假设的损失定义为平均值：

$$\frac{1}{k-1}\sum_{y\neq y_i}\frac{1}{2}(1-h(x_i,y_i)+h(x_i,y))$$
$$=\frac{1}{2}(1-h(x_i,y_i)+\frac{1}{k-1}\sum_{y\neq y_i}h(x_i,y))$$

然而，不同的区分问题很可能在不同情况下有不同的重要性。在不同问题上附加不同重要性的方法是给每个问题赋一个权重。因此，对每个实例 x_i 和不正确的标签 $y\neq y_i$ 指定权重 $q(i,y)$，它与区分非正确标签 y 和正确标签 y_i 相联系。代之入上式，其结果称之为 h 在训练例 i 上关于 q 的伪损失：

$$ploss_q(h,i)\approx\frac{1}{2}(1-h(x_i,y_i)+\sum_{y\neq y_i}q(i,y)h(x_i,y))$$

函数 $q:\{1,\cdots,N\}\times Y\to[0,1]$称为标签权重函数。对所有 i，有：$\sum_{y\neq y_i}q(i,y)=1$。弱学习器的目标是对给定分布 D 和权重函数 q 最小化预期的伪损失：$ploss_{D,q}(h):=E_{i\sim D}[ploss_q(h,i)]$。通过控制 D 和 q，本算法有效地使弱学习器不仅集中于困难的实例，也集中于最难于排除的非正确类标签上。相反地，这种伪损失衡量可能使弱学习器易于得到弱优势。AdaBoost.M2 调用给定的弱学习算法 WeakLearn 时，将产生伪损失为 $\varepsilon_1,\cdots,\varepsilon_T$ 的假设。于是最终假设 h_f 的错误 $\varepsilon=\Pr_{i\sim D}[h_i(i)\neq y_i]$的上边界为：$\varepsilon\leqslant(k-1)2^T\prod_{i=1}^{T}\sqrt{\varepsilon_i(1-\varepsilon_i)}$。

2.3 AdaBoost.MR 和 AdaBoost.MH 算法

AdaBoost.MR 和 AdaBoost.MH 算法的扩展可以用于高效地处理标签集合[11]，解决多类多标签问题。每个待分类对象可以属于多个类别中的一个或多个，即所谓“兼类”。单标签分类问题是多标签问题的特例。设 $\mathcal{Y}$ 为标签或类的有限集，令 $K=|\mathcal{Y}|$。在多标签情况下，每个实例 $x_i\in X$ 可能属于 $\mathcal{Y}$ 中的多个标签。因此，带标签的例子是(x,Y)对，其中 $Y\subseteq\mathcal{Y}$ 是赋给 x 的标签集。学习的目的是需要寻找试图预测赋给例子的标签集的一个标签的假设。即目的是在某新观察 (x,Y) 上寻找最小化 $H(x)\notin Y$ 的概率的 $H:X\to\mathcal{Y}$。这种测量被称之为假设 H 的 *one-error*，因为它度量的是连一个标签都不正确的概率。用 $one\text{-}error_D(H)$ 表示假设 H 在观察 (x,Y) 上关于分布 D 的 *one-error*。即：$one\text{-}error_D(H)=\Pr_{(x,Y)\sim D}[H(x)\notin Y]$。

2.3.1 AdaBoost.MH 算法

输入 $((x_1,Y_1),\cdots,(x_N,Y_N))$，其中 $x_i\in X,Y_i\subseteq\mathcal{Y}$。

初始化 $D_1(i,l)=1/(Nk)$。

对 t=1,…,T 循环执行：

①分布 D_t 训练弱学习器；

②得到弱假设 $h_t:X\times\mathcal{Y}\to R$；

③选择 $\alpha_t\in R$；

④修改：$D_{t+1}(i,l)=(D_t(i,l)\exp(-\alpha_tY_i[l]h_t(x_i,l)))/Z_t$。其中 Z_t 是归一化因子(使 D_{t+1}为分布)；

输出 最终假设：$H(x,l)=Sign(\sum_{t=1}^{T}\alpha_th_t(x,l))$。

AdaBoost.MH 的思想是基于汉明损失，其目的是找到与每个实例相联系的标签集的假设。通过对每个样本 x 和每个标签 y 产生一套如下的二值问题：“对样本 x，其正确标签是 y 还是其他?”来运作。假定现在的目标是仅预测所有正确标签。学习算法产生预测标签集的假设，其损失取决于预测集和观察结果有多大的差异。因此，损失为：$\frac{1}{k}E_{(x,Y)\sim D}[h(x)\Delta Y]$，其中 Δ 表示对称差别。该损失称之为 H 的汉明损失，以 $hloss_D(H)$ 表示。为最小化它，把该问题分解成 k 个正交的二值分类问题。即可将 Y 看作是 k 个二值标签的特定值(依赖于某标签 y 是否包含在 Y 中)。相似地，$h(x)$可被看作 k 个二值预测。于是该损失可被认为是 k 个二值问题上 h 错误率的平均。对 $Y_i\subseteq\mathcal{Y},l\in\mathcal{Y}$，定义 $Y[l]$为：$Y[l]=\begin{cases}+1 & \text{若 } l\in Y\\ -1 & \text{若 } l\notin Y\end{cases}$。为简化表示，用 $H(x,l)=H(x)[l]$定义的二元函数 $H:X\times\mathcal{Y}\to\{+1,-1\}$表示相应的函数 $H:X\to2^{\mathcal{Y}}$。由此可见 Boosting 对汉明损失的最小化。约简的主要思想是用 k 个例子$((x_i,l),Y_i[l])$(对 $l\in\mathcal{Y}$)代替每个训练例(x_i,Y_i)。这种约简也导致了最终假设的选取。可以证明，H 的汉明损失的上界为：$hloss(H)\leqslant\prod_{i=1}^{T}Z_i$。

本算法可用于最小化 *one-error*。设 $H^1(x)=\arg\max_y\sum_i\alpha_ih_i(x,y)$，对观察 $(x,Y)(Y\neq\phi)$上的任意分布 D，有：$one\text{-}error_D(H^1)\leqslant k\cdot hloss_D(H)$。

2.3.2 AdaBoost.MR 算法

输入 $((x_1,Y_1),\cdots,(x_N,Y_N))$，其中 $x_i\in X,Y_i\subseteq\mathcal{Y}$。

初始化 $D_1(i,l_0,l_1)=$

$$\begin{cases}1/(N\cdot|Y_i|\cdot|\mathcal{Y}-Y_i|) & \text{若 } l_0\notin Y_i \text{ 且 } l_1\in Y_i\\ 0 & \text{其它}\end{cases}$$

对 t=1,…,T 循环执行：

①用分布 D_t 训练弱学习器；

②得到弱假设 $h_t:X\times\mathcal{Y}\to R$；

③选择 $\alpha_t\in R$；

④修改：$D_{t+1}(i,l_0,l_1)=(D_t(i,l_0,l_1)\exp(\frac{1}{2}\alpha_t(h_t(x_i,l_0)-h_t(x_i,l_1))))/Z_t$。其中 Z_t 是归一化因子(使 D_{t+1}为分布)；

输出 最终假设：$f(x,l)=\sum_{t=1}^{T}\alpha_th_t(x,l)$。

AdaBoost.MR 的主要思想是基于排序损失，其目的是寻找一个对标签进行排序的假设，希望正确的标签能得到最高的级别。形式化地表示，找出形如 $f:X\times\mathcal{Y}\to R$ 的假设，将其理解为：对一给定实例 x，$\mathcal{Y}$ 中标签应按照 $f(x)$排序。即，如果 $f(x,l_1)>f(x,l_2)$，某标签 l_1 被认为级别高于 l_2。对于某观察 (x,Y)，仅关心关键对 $l_0,l_1(l_0\notin Y,l_1\in Y)$的相对排序关系。如果 $f(x,l_1)\leqslant f(x,l_0)$，则称 f 错排了关键对 l_0,l_1。此处的目标是寻找仅有少量错排的函数 f，使 Y 中的标签排在 Y 外的标签之前。因此，需要最小化预期的错排的关键对。这种度量被称之为排序损失。对观察上的某分布 D，被定义为：$E_{(x,Y)\sim D}[|\{(l_0,l_1)\in(\mathcal{Y}-Y)\times Y:f(x,l_1)\leqslant f(x,l_0)\}|/|Y||\mathcal{Y}-Y|]$，用 $rloss_D(f)$表示。对任意观察 Y 都不会为空，也不会等于 $\mathcal{Y}$，否则就无需排序了。本算法在$\{1,\cdots,N\}\times Y\times Y$ 上维护分布 D_t。只有 l_0,l_1 是相关于(x_i,Y_i)的关键对时，该分布在三元组(i,l_0,l_1)上是非 0 的。可以证明，f 的排序损失的上界为：$rloss(f)\leqslant\prod_{t=1}^{T}Z_t$。

2.3.3 改进的 AdaBoost.MR 算法

输入 $((x_1,Y_1),\cdots,(x_n,Y_n))$，其中 $x_i\in X,Y_i\subseteq\mathcal{Y}$。

初始化 $v_1(i,l)=(n\cdot|Y_i|\cdot|\mathcal{Y}-Y_i|)^{-\frac{1}{2}}$。

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 $h_t: X\times\mathcal{Y}\to R$；

③选择 $\alpha_t\in R$；

④修改：$v_{t+1}(i,l)=(v_t(i,l)\exp(-\frac{1}{2}\alpha_t Y_i[l]h_t(x_i,l)))/\sqrt{Z_t}$，其中：

$Z_t=\sum_i[(\sum_{l\in Y_i}v_t(i,l)\exp(\frac{1}{2}\alpha_t h_t(x_i,l)))(\sum_{l\notin Y_i}v_t(i,l)\exp(-\frac{1}{2}\alpha_t h_t(x_i,l)))]$。

输出 最终假设：$f(x,l)=\sum_{t=1}^{T}\alpha_t h_t(x,l)$。

原 AdaBoost.MR 方法当有很多标签时，在时间和空间方面效率不是很高，并要对每个训练例(x_i,Y_i)维护$|Y_i|\cdot|\mathcal{Y}-Y_i|$个权重，且每回都需修改每个权重。因此，每次迭代所需的空间和时间复杂度可相差 $O(nk^2)$。实际上，相同的算法每次迭代所需的空间和时间复杂度可以为 $O(nk)$，只需在$\{1,\cdots,n\}\times\mathcal{Y}$上维护权重 v_t。如 l_0,l_1 是相关于(x_i,Y_i)的关键对，则有 $D_t(i,l_0,l_1)=v_t(i,l_0)\cdot v_t(i,l_1)$。可以证明，本算法和原 AdaBoost.AR 算法是等价的，而开销只有 $O(nk)$。本算法可用于最小化 *one-error*。设 $H^1(x)=\arg\max_y\sum_t f(x,y)$，对观察$(x,Y)(Y\neq\phi)$上的任意分布 D，有：$one\text{-}error_D(H^1)\leqslant(k-1)\cdot rloss_D(f)$。

2.4 AdaBoost.MO 和 AdaBoost.OC 算法

该算法用于解决大类单标签问题。ECOC(Error-correcting output codes)由 Dieterich 和 Bakiri 于 1995 年提出[7]，它是鲁棒的、通过把多类学习问题简化为一系列的两类问题以解决大类问题的方法。假定类别数 K 很大，新的学习问题可以通过随机地把这 K 个类划分为两个子集 A_l 和 B_l 建立起来。输入数据可以被重新赋予标签，使得在 A_l 中的每个原始类得到标签"0"，B_l 中的每个原始类得到标签"1"。这些重新被标记的数据被传递给学习算法，学习算法以此建立分类器 h_l。重复该过程 L 次(产生不同的子集 A_l 和 B_l)，可以得到 L 个分类器的组合：$h_1,\cdots,h_L$。现给定一新的数据点，ECOC 让每个 h_l 来分类它。如果 $h_l(x)=0$，A_l 中的每个类得到一次投票；若 $h_l(x)=1$，B_l 中的每个类得到一次投票。在 L 个分类器都投票后，有最高票数的类被选作综合预测的结果。Dieterich 和 Bakiri 宣称该技术在很多困难的分类问题上可提高 C4.5 和 BP 算法的性能。Boosting 是一个提高给定基学习或"弱"学习算法精度的一般方法。Boosting 与 ECOC 两者优势的结合将提高学习效率和增强解决多类问题的能力。

2.4.1 AdaBoost.OC 算法

是 ECOC 与 AdaBoost.M2 的综合，其主要优势是实现简单，它可以使用任何可解决两类问题的学习算法[8]。尽管有多种解决多类问题的 Boosting 方法，AdaBoost.OC 比其它方法快得多，其性能优于 ECOC 和 Bagging，且在产生基学习算法上所需的编程工作更少。

AdaBoost.M2 不是为每个例子预测单个类，而是对每个例子选择"合理"的标签集。这样弱假设的分类性能通过伪损失得到测量。对某个给定例子，该伪损失在以下情况下惩罚弱假设：(1)在预测出的合理标签集中没有包含正确标签；(2)每个被包括在合理标签集中的非正确标签。对某个给定例，最终的综合假设选择在弱假设选出的标签集中最频繁出现的标签(可能会给不同的弱假设以不同的权值)。伪损失的确切形式在 Boosting 算法的控制之下，弱学习算法因此必须可以处理损失度量形式的变化。这样 Boosting 算法使弱学习器不仅集中于难于预测的例子，也集中于那些难于与正确标签分开的标签的例子。该方法(AdaBoost.M2)也有一些弱点，首先，它需要设计一个与 Boosting 算法定义的伪损失相适应的弱学习器，且以合理标签集的形式产生预测。因为大多数成型的学习算法是基于错误的，这就可能需要编程人员额外的修改或创造(如果不能得到弱学习器的源码则 AdaBoost.OC 根本得不到应用)。其次，伪损失方法可能会相当慢。典型的，该弱学习器的运行时间会比基于错误的解决 K 类问题的算法慢 $O(k)$倍。

该方法结合了二者的长处：弱学习算法仅需处理基于通常错误而不是费时费事的伪损失的二值问题；该算法有坚实的理论保证，即如果弱学习器能持续地产生稍好的弱假设(相应于训练集上的分布和二值例子标签)，最终综合假设的错误将可以任意小。

输入 例子序列$\langle(x_1,y_1),\cdots,(x_n,y_n)\rangle$，其中 $x_i\in X$，$y_i\in Y$。迭代次数 T。

初始化 对 $i=1,2,\cdots,T$，置 $\tilde{D}_i(i,l)=[\![l\neq y_i]\!]/(n(k-1))$ $y\in Y-\{y_i\}$(在所有非正确标签上一致)。

步骤：(对 t=1,2,…,T)：

①计算着色函数 $\mu_t: Y\to\{0,1\}$(把标签集 Y 划分为两个部分)。

②设 $U_t=\sum_{i=1}^{n}\sum_{l\in Y}\tilde{D}_t(i,l)[\![\mu_t(y_i)\neq\mu_t(l)]\!]$。

③设 $D_t(i)=(\sum_{l\in Y}\tilde{D}_t(i,l)[\![\mu_t(y_i)\neq\mu_t(l)]\!])/U_t$。

④在例子$(x_1,\mu_t(y_1)),\cdots,(x_n,\mu_t(y_n))$上依分布 D_t 训练弱学习器；返回假设 $h_t: X\to[0,1]$。

⑤设 $\tilde{h}_t(x)=\{l\in Y: h_t(x)=\mu_t(l)\}$。

⑥计算伪损失 $\tilde{\varepsilon}_t=\frac{1}{2}\sum_{i=1}^{n}\sum_{l\in Y}\tilde{D}_t(i,l)\cdot([\![y_i\notin\tilde{h}_t(x_i)]\!]+[\![l\in\tilde{h}_t(x_i)]\!])$。

⑦置 $\alpha_t=\frac{1}{2}\ln(\frac{1-\tilde{\varepsilon}_t}{\tilde{\varepsilon}_t})$。

⑧修改：$\tilde{D}_{t+1}(i,l)=(\tilde{D}_t(i,l)\cdot\exp(\alpha_t([\![y_i\notin\tilde{h}_t(x_i)]\!]+[\![l\in\tilde{h}_t(x_i)]\!])))/Z_t$。其中 Z_t 为归一化因子(使 $\tilde{D}_{t+1}$ 和为 1)。

输出 最终假设：$H_{final}(x)=\arg\max_{l\in Y}\sum_{t=1}^{T}\alpha_t[\![h_t(x)=\mu_t(l)]\!]$。

每回迭代，数据都依照着色函数 μ_t 重新加标签，弱学习器是按 D_t 加权的重新加标签的数据训练而得，得到的弱假设用 h_t 表示。弱学习算法的目的是最小化重新加标签和设权重的数据的训练错误，即最小化 $\varepsilon_t=\sum_{i=1}^{n}D_t(i)[\![h_t(x_i)\neq\mu_t(y_i)]\!]=\Pr_{i\sim D_t}[h_t(x)\neq\mu_t(y_i)]$。假定某弱假设 $h_t: X\to[0,1]$ 已经按照某 $\mu_t: Y\to\{0,1\}$计算出。在例 x 上 h_t 的二值分类可被看成是对满足 $h_t(x)=\mu_t(l)$的标签 l 的投票。h_t 定义这些标签是"合理"的。为简化伪损失设定，自然地用软假设 $\tilde{h}_t$ 代替 h_t：$\tilde{h}_t(x)=\mu_t^{-1}(h_t(x))=\{l\in Y: h_t(x)=\mu_t(l)\}$。可以证明，如果 h_t 有错误率 $\varepsilon_t=1/2-\gamma_t$，则 $\tilde{h}_t$ 的伪损失也会比 1/2 稍好，若选择 $U_t>0$ 的话。实际上 AdaBoost.OC 是 AdaBoost.M2 中弱的软假设的一种特殊情况。由此可得 H_{final} 训练错误的上界：$(k-1)\prod_{t=1}^{T}\sqrt{1-4(\gamma_t U_t)^2}\leqslant(k-1)\exp(-2\sum_{t=1}^{T}(\gamma_t U_t)^2)$。

关于选择 μ_t，希望它可最大化 $U_t=\sum_{i=1}^{n}\sum_{l\in Y}\tilde{D}_t(i,l)[\![\mu_t$

$(y_i)\neq\mu_i(l)\ ]\!]$。U_i 的值仅依赖于 $\tilde{D}_i$ 和 μ_i，不依赖于弱假设。这意味着可在调用弱假设前找到可最大化 U_i 的 u_i。选择 u_i 的最简方法是对每个标签 $l\in Y$ 随机地从{0,1}中一致、独立地选择每个值 $u_i(l)$。对任意的 $l\neq l'$，$u_i(l)\neq u_i(l')$ 的概率为 1/2。因此，U_i 的期望值也是 1/2。比较精确的方法是随机地选择 u_i，但要确保对标签集的近乎平等的划分。即，在所有的确实有 $[k/2]$ 的标签被映射为 0 的着色函数中随机一致地选取 u_i。如果 $l\neq l'$，则 $u_i(l)\neq u_i(l')$ 的概率为 $(1+1/(k(1))/2$，于是 U_i 的期望值比 1/2 稍好。另一方法是使用组合优化来最大化 U_i。U_i 可被重写为：$U_i=\sum_{l,l'\in Y}[\![\ \mu_i(l)\neq\mu_i(l')\ ]\!]\ w_i(l,l')$，其中 $w_i(l,l')=\sum_{i=1}^{n}\tilde{D}_i(i,l)\ [\![\ l'=y_i\ ]\!]$。可看出，最大化 U_i 是"MAX-CUT"问题的一种特殊形式，它是 NP-完全的。有很多解决它的复杂近似方法，此处不再详述。实验结果表明，AdaBoost. OC 的性能比 ECOC 和 Bagging 都好，与 AdaBoost-.M2 一样。

2.4.2 AdaBoost. MO 算法　是 ECOC 与 AdaBoost. MH 的结合。

输入　$((x_1,y_1),\cdots,(x_n,y_n))$，其中 $x_i\in X$，$y_i\in\mathcal{G}$，和映射 λ：$\mathcal{G}\to 2^{\mathcal{G}'}$。

①在重新加标签的数据 $((x_1,\lambda(y_1)),\cdots,(x_n,\lambda(y_n)))$ 上运行 AdaBoost. MH。

②返回形如 $H(x,y')=sign(f(x,y'))$ 的最终假设 H，其中 $f(x,y')=\sum_t\alpha_t h_t(x,y')$。

③输出：修改后的最终假设(包括选择 1 和选择 2)：

选择 1：$H_1(x)=\arg\max_{y\in\mathcal{G}}|\lambda(y)\Delta H(x)|$

选择 2：$H_2(x)=\arg\max_{y\in\mathcal{G}}\sum_{y'\in\mathcal{G}'}\exp(-\lambda(y)[y']f(x,y'))$

λ 映射标签集 $\mathcal{G}$ 到一个未指定的标签集 $\mathcal{G}'$ 的子集中。$\mathcal{G}'$ 不需要和 $\mathcal{G}$ 一样。令 $k'=|\mathcal{G}'|$。此处用与 ECOC 稍有差异的方法来选择 λ 使不同的标签映射到相互远离(根据其对称差)的集合中。在 k' 不是太小时，通过完全随机地选择 λ 来得到相似的效果(即，对 $y\in\mathcal{G}$ 和 $l\in\mathcal{G}'$，在 $\lambda(y)$ 中以相同的概率包括或不包括 l)。一旦某函数 λ 被选择，就可直接传送数据 $(x_i,\lambda(y_i))$ 给 AdaBoost. MH。训练结束后，如何给新实例 x 分类？ECOC 方法是在 x 上估计 H 以获得集合 $H(x)\subseteq\mathcal{G}'$。于是可选择标签 $y\in\mathcal{G}$，其映射的输出码 $\lambda(y)$ 有与 $H(x)$ 最短的汉明距离，即选择 $\arg\max_{y\in\mathcal{G}}|\lambda(y)\Delta H(x)|$。该方法的一个弱点是它忽略了每个标签是否包含在 $H(x)$ 中的自信度。一个替代的方法是预测标签 $\arg\max_{y\in\mathcal{G}}\sum_{y'\in\mathcal{G}'}\exp(-\lambda(y)[y']f(x,y'))$。

设 $\rho=\min_{l_1,l_2\in\mathcal{G};\ l_1\neq l_2}|\lambda(l_1)\Delta\lambda(l_2)|$。可以证明，AdaBoost. MO 训练错误的上边界，对于"选择 1"，为 $\frac{2k'}{\rho}\prod_{t=1}^{T}Z_t$；对于"选择 2"，为 $\frac{k'}{\rho}\prod_{t=1}^{T}Z_t$。

2.5 AdaBoost. R 算法

AdaBoost. R 算法用于解决回归问题。回归问题实际上是实值预测问题。该算法的目的是估计出与待分类对象相联系的实数值。

输入　N 个例子序列 $((x_1,y_1),\cdots,(x_N,y_N))$，标签 $y_i\in Y=[0,1]$，实例集上的分布 D，弱学习算法 WeakLearn，迭代次数 T。

初始化　对 $i=1,2,\cdots,T$，置权重向量 $w^1_{i,y}=D(i)|y-y_i|/Z$，$y\in Y$，其中 $Z=\sum_{i=1}^{N}D(i)\int_0^1|y-y_i|dy$。

步骤(对 t=1,2,…,T)：

①置 $P^t=W^t/(\sum_{i=1}^{N}\int_0^1 w^t_{i,y}dy)$。

②调用 WeakLearn，传递密度 P^t 给它；返回假设 $h_t:X\to Y$。

③计算 h_t 的伪损失：$\varepsilon_t=\sum_{i=1}^{N}|\int_{y_i}^{h_t(x_i)}p^t_{i,y}dy|$。若 $\varepsilon_t>1/2$，则设 $T=t-1$ 并退出循环。

④置 $\beta_t=\varepsilon_t/(1-\varepsilon_t)$。

⑤对 $i=1,2\cdots,T$，$y\in Y$，计算新的权重向量：

$$w^{t+1}_{i,y}=\begin{cases}w^t_{i,y} & \text{若 } y_i\leqslant y\leqslant h_t(x_i)\text{ 或 }h_t(x_i)\leqslant y\leqslant y_i\\ w^t_{i,y}\beta_t & \text{其他}\end{cases}$$

输出　假设：$h_f(x)=\inf\{y\in Y:\sum_{t:\,h_t(x)\leqslant y}\log(1/\beta_t)\geqslant\frac{1}{2}\sum_t\log(1/\beta_t)\}$。

此处 Y=[0,1]。把回归问题简化为"对每个实例 (x_i,y_i) 和无限集 $\{y|y\in Y\}$ 中每个 y，正确标签 y_i 比 y 大还是小？"的二值问题的无限集合。用波浪线标记简化空间中的所有变量。对每个例子 (x_i,y_i)，定义一个简化空间的连续例子集，其中的例子由 (x,y) 对所有 $y\in[0,1]$ 索引；简化空间的相关例子是 $\tilde{x}_{i,y}=(x_i,y)$，标签是 $\tilde{y}_{i,y}=[\![\ y\geqslant y_i\ ]\!]$。具体地说，每个假设 $h:X\to Y$ 被简化为一个二值假设 $\tilde{h}:X\times Y\to\{0,1\}$，其定义为：$\tilde{h}(x,y)=[\![\ y\geqslant h(x)\ ]\!]$。因此，$\tilde{h}$ 试图用估计值 $h(x)$ 回答这些二值问题。在简化中，训练集上的分布 D 被映射为 (i,y) 对上密度 $\tilde{D}$ 的分布。映射方式为：在简化空间中分类错误的最小化就等价于在原问题上 MSE 的最小化。为实现这一点，定义：$\tilde{D}(i,y)=D(i)|y-y_i|/Z$，$y\in Y$，其中 $Z=\sum_{i=1}^{N}D(i)\int_0^1|y-y_i|dy$ 为归一化常量。显然，$1/4\leqslant Z\leqslant 1/2$。如果按 $\tilde{D}$ 计算 $\tilde{h}$ 的二值错误，它直接与均方误差成比例：$\sum_{i=1}^{N}\int_0^1|\tilde{y}_{i,y}-\tilde{h}(\tilde{x}_{i,y})|\tilde{D}(i,y)dy=\frac{1}{2Z}\sum_{i=1}^{N}D(i)(h(x_i)-y_i)^2$。比例常数是 $1/(2Z)\in[1,2]$。AdaBoost. R 对每一实例 i 和标签 $y\in Y$ 维护权重 $w^t_{i,y}$。初始权重函数 W^1 是定义的密度 $\tilde{D}$。通过归一化它得到密度 P^t 并将它传递给弱学习器。弱学习器的目的是找到一假设 $h_t:X\to Y$，它能最小化 ε_t。最终假设 h_f 与简化过程一致。每个简化的弱假设 $\tilde{h}_t(x,y)$ 是 y 的非减函数。因此，在简化空间中通过 AdaBoost 产生的最终假设 $\tilde{h}_f$(为这些假设加权和的阈值)也是 y 的非减函数。$\tilde{h}_f$ 的输出若为二值则意味着：对每个 x 有一个 y 值，对所有 $y'<y$，有 $\tilde{h}_f(x,y')=0$，而对所有 $y'>y$，有 $\tilde{h}_f(x,y')=1$。这就是算法中 h_f 给出的值。实际上 h_f 是计算弱假设的加权中值。

在一不可数点集上维护权重 $w^t_{i,y}$ 是乎是不可能的。然而，当把 $w^t_{i,y}$ 看作 y 的函数时，它是一分段线性函数。对 $t=1$，$w^t_{i,y}$ 有 2 个线性段，且权值的每次修改都隐含着在点 $h_t(x_i)$ 上把其中一段分为两半。所有的操作是初始化、保存和修改这样的分段函数。同时，算法中出现的积分也可被显式地估计，因为它们仅包含分段线性函数的求和。AdaBoost. R 调用给定的弱学习算法 WeakLearn，将产生错误率为 $\varepsilon_1,\cdots,\varepsilon_T$ 的假设。

(下转第 145 页)

本吻合。FT的指令条数虽然没有减少得那么多，但其取数操作却减少为原来的1/8，由此带来了指令执行时延(EXCUTION LATENCY_CYCLE)(1/26)、取指周期(INST_FETCH_CYCLE)(1/28)和存储器访问周期(MEMORY_CYCLE)(1/6)的显著减少，可见LOAD操作的优化对程序的加速比贡献很大。

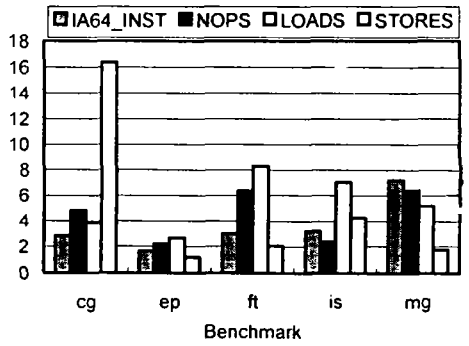

图13 orcc编译的指令条数比较(O0/O3)

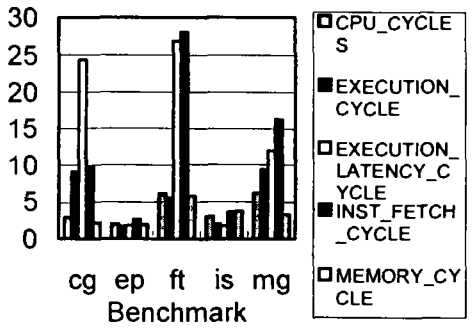

图14 周期分布(O0/O3)

从统计结果(表3)中还可以看出，尽管经过强劲的优化措施，程序执行过程中的空操作仍占有相当的比例(36%)，这说明处理器资源还有30%以上的空闲，如果能有效地利用这些处理资源，将获得更高的加速比或吞吐量。

表3 各类操作所占总操作的百分比

操作类型 \ 编译选项	-O0	-O3
空操作(NOP)	45%	36%
取数操作(LOAD)	22%	15%
存数操作(STORE)	5%	6%

结论 本文通过在安腾处理器上编译运行NAS Benchmarks，从程序的运行时间、编译时间和产生的可执行文件的大小3个方面，评价了orcc、sgicc和gcc的性能。通过上文的比较，大致可得出如下结论：

·orcc和sgicc性能相当，约为gcc的2倍。

·orcc比sgicc有效地控制了编译时间，但仍比gcc慢10倍。

·orcc和sgicc产生的可执行文件大小相当，约为gcc产生文件的7倍。

通过对程序运行时的监测，发现空操作占有相当的比例，处理器还有30%的空闲。减少存储器访问操作或隐藏存储器访问时延可以显著地提高程序的性能。

参考文献

1 Intel IA-64 Architecture Software Developer's Manual. Volume 1: IA-64 Application Architecture. 2000

2 Roy J, Sun C, Chengyong W. Open Research Compiler(ORC)for Itanium Processor Family(IPF). MICRO-34 Tutorial. 2001

3 http://science.nas.nasa.gov/Software/NPB

4 Agarwal R C, Alpern B, Carter L, Gustavson F G, Klepacki D, Lawrence R, Zubair M. High PerformanceParallel Implementations of the NAS Kernel Benchmarks on the IBM SP2. IBM Systems Journal, 1995, 34:263～272

5 Goedecker S. Fast Radix 2,3,4, and 5 Kernels for Fast Fourier Transformations on Computers with overlapping multiply-add instructions. SIAM Journal on Scientific Computing, 1997, 16:1605～1611

6 Boisseau J, Carter L, Gatlin K, Majumdar A, Snavely A. NAS Benchmarks on the Tera MTA. In: Workshop on Multi-Threaded Execution, Architecture, and Compilers. 1998

7 ftp://ftp.hpl.hp.com/pub/linux-ia64/pfmon-0.06.tar.gz

(上接第34页)

最终假设 h_f 的均方误差 $\varepsilon=E_{i\sim D}[(h_f(x_i)-y_i)^2]$ 的上边界为：$\varepsilon\leqslant 2^T\prod_{t=1}^{T}\sqrt{\varepsilon_t(1-\varepsilon)}$。

参考文献

1 Valiant L G. A theory of the learnable. Communications of the ACM 1984,27(22):1134～1142

2 Kearns M K, 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

3 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

4 Schapire R E. The strength of weak learnability. Machine Learning, 1990,5(2):197～227

5 Freund Y. Iyer R, Schapire R E, Singer Y. An efficient bossting algorithm for combining preferences. In: Machine Learning: Proc. of the Fifteenth Int Conf. 1998

6 Freund Y, Schapire R E. Adecision-theoretic generalization of online learning and an application to boosting. Journal of Computer and System Science, 1997,55(1):119～139

7 Dietterich T G, Bakiri C. Solving multiclass learning problems viaerror-correcting output codes. Journal of Artificial Intelligence Research, 1995,2:263～286

8 Schapire R E, Singer Y. Using output codes to boost multiclass learning problems. In: Machine Learning: Proc. of the Fourteenth International Conference, 1997. 313～321

9 Schapire P E, Singer Y. Improved boosting algorithms using confidence-related predictions. In: Proc. of the Eleventh Annual Conf. on Computational Learning Theory, 1998. 80～91

10 Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting: [Technical Report]. 1998

11 Schapire R E, Singer Y. BoostTexer: A system for multiclass multi-label text categorization. Machine Learning, 1998

12 刁力力，等. 数据挖掘与组合学习. 计算机科学，2001，28(7)

13 涂承胜，刁力力，鲁明明，陆玉昌. Boosting家族Boost-by-majority系列代表. 计算机科学，2003，30(4)