社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－AL－97　(5)
2004／10／14

分割と併合に基づくブースティング

瀧本　英二†　　小屋　州平†　　丸岡　　章†

† 東北大学大学院情報科学研究科
〒 980-8579 仙台市青葉区荒巻字青葉 6-6-05
E-mail: †{t2,s_koya,maruoka}@maruoka.ecei.tohoku.ac.jp

あらまし　InfoBoost は，学習対象に関する相互情報量が 0 ではない複数の仮説を統合して精度の高い仮説を構築するブースティングアルゴリズムである．本稿は，InfoBoost が，ドメインの分割と併合を繰り返すブランチングプログラムを生成するアルゴリズムとみなせることを示す．ただし，ブランチングプログラムを定義する決定グラフの各頂点には重みが割り当てられ，各事例に対する統合仮説の値は，その事例がたどる道上の重みの和に基づいて定められる．さらに，併合の手続きを一般化し，新しいブースティング手法のクラス BP.InfoBoost を提案する．InfoBoost は，分割された領域をすべて併合してしまう極端なアルゴリズムとみなせる．一方，併合をあまり行わないと，仮説の性能向上の効率は良くなるが過学習のリスクを伴う．本稿では，仮説の性能向上の効率化と過学習のリスクの回避を両立する併合手法を提案する．

キーワード　ブースティング，決定木，ブランチングプログラム，相互情報量

Boosting Based on Divide and Merge

Eiji TAKIMOTO†, Syuhei KOYA†, and Akira MARUOKA†

† GSIS, Tohoku University,
Aoba 6-6-05, Aramaki, Aoba-ku, Sendai 980-8579, Japan
E-mail: †{t2,s_koya,maruoka}@maruoka.ecei.tohoku.ac.jp

Abstract　InfoBoost is a boosting algorithm that improves the performance of the master hypothesis by combining weak hypotheses with non-zero mutual information about the target. We give a somewhat surprising observation that InfoBoost can be viewed as an algorithm for growing a branching program that divides and merges the domain repeatedly. Here, weights are assigned to nodes of the decision graph that the branching program defines, and for a given instance, the value that the master hypothesis takes is determined by the sum of the weights over the path induced by the instance. We generalize the merging process and propose a new class of boosting algorithms called BP.InfoBoost with various merging schema. InfoBoost is a BP.InfoBoost with an extreme scheme that merges all nodes in each round. The other extreme that merges few nodes yields an algorithm that improves the master hypothesis very efficiently but has a risk of overfitting. We propose a merging scheme between these extremes that improves the master hypothesis efficiently while avoiding the risk of overfitting.

Key words　boosting, decision tree, branching program, mutual information

1. Introduction

Since the stage-wise process of AdaBoost was explained in terms of a Newton-like method for minimizing an exponential cost function [2], [5], designing and analyzing boosting algorithms by using results from the optimization theory has been a mainstream of this research area [3], [4], [9], [11]. In the most common scheme of boosting, the booster iteratively makes probability weightings over the sample and receives weak hypotheses that slightly correlate with the sample with respect to the current weightings. The booster then produces a linear combination of the weak hypotheses as a master hypothesis with a good performance guarantee expressed in terms of the margin [6], [12]. Here, the correlation between a weak hypothesis and the sample is defined as the expected margin, which intuitively mean how well the hy-

pothesis classifies the sample.

As oppose to the margin based criterion, there is an information-theoretic approach where the correlation is defined by the mutual information, i.e., how much amount of information the hypotheses bring on the sample. In this scheme, the requirement for weak hypotheses can be relaxed to have non-zero mutual information, rather than non-zero margin, to obtain a good master hypothesis. The first theoretical result on information-based boosting is due to Kearns and Mansour [8] who give theoretical justification to empirically successful heuristics in top-down decision tree learning. The result is naturally generalized for learning multiclass classification problems [14] and is improved to avoid overfitting by merging some sets of nodes [10]. A modification of the latter algorithm is applied to the problem of boosting in the presence of noise [7]. These algorithms are quite different from AdaBoost-like algorithms in that they grow a decision tree or a branching program as a master hypothesis that is far from the form of linear combinations. Moreover, the probability weightings they make are not based on the principle of minimizing any cost function. Later we give an evidence that these algorithms are inefficient in the sense that they make the complexity of the master hypothesis unnecessarily large. Aslam proposed a promising method of information-based boosting called InfoBoost [1]. InfoBoost is interesting because the weight update is obtained by minimizing the same cost function as AdaBoost and the master hypothesis has an "quasi-linear" form (the coefficients are not constants but depend on the values of weak hypotheses).

In this paper, we generalize InfoBoost and propose a new class of information-based boosting algorithms. We first give a somewhat surprising observation that InfoBoost can be viewed as a process of growing a branching program that divides and merges the domain repeatedly. We show that the merging process used in InfoBoost can be replaced by *any* merging scheme with the boosting property preserved. So we have a class of boosting algorithms with various merging schema. We call any of them a BP.InfoBoost. BP.InfoBoost assigns to each node a weight as well as a weak hypothesis, and the master hypothesis is a threshold function of the sum of the weights over the path induced by a given instance. Note that this is different from the previous BP based boosting algorithms [7], [10] where the master hypothesis is just a branching program. InfoBoost is a BP.InfoBoost using an extreme scheme that merges all nodes in each round. The other extreme that merges no nodes yields an algorithm for growing a decision tree. We particularly call this version DT.InfoBoost. DT.InfoBoost has a *totally corrective* update property. (The notion of totally corrective updates was originally proposed for a better booster in the margin based criterion [9].) Specifically, a totally corrective update makes the weight D_{t+1} for the next round so that *all* weak hypotheses $h_1, \ldots, h_t$ obtained so far are uncorrelated with the sample with respect to D_{t+1}. This implies that any weak hypothesis h_{t+1} that correlates with the sample must contain novel information that $h_1, \ldots, h_t$ do not have. Note that AdaBoost and InfoBoost make D_{t+1} so that only h_t is uncorrelated. So we can expect that DT.InfoBoost improves the master hypothesis much faster than InfoBoost. On the other hand, since the size of the decision tree may grow exponentially in t, DT.InfoBoost has more risk of overfitting than InfoBoost[1].

There must be an appropriate merging scheme between the two extremes that takes advantages of the two extremes. In this paper, we propose a merging scheme that reduces the training error of the master hypothesis nearly as fast as the one we would have without merge while keeping the master hypothesis (branching program) in a moderate size.

2. Preliminaries

Let X denote an instance space and $Y = \{-1, +1\}$ the set of labels. We fix a sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \subseteq X \times Y$ on which we discuss the performance of boosting algorithms. The procedure of boosting is described as in the following general protocol with two parties, the booster and the weak learner. On each round t, the booster makes a probability distribution D_t on the index set $\{1, \ldots, m\}$ of the sample S and gives S with D_t to the weak learner; The weak learner returns a weak hypothesis $h_t : X \to Y$ to the booster (denoted $h_t = \mathrm{WL}(S, D_t)$); The booster updates the distribution to D_{t+1}. Repeating the above procedure for T rounds for some T, the booster combines the weak hypotheses $h_1, \ldots, h_T$ obtained so far and makes a master hypothesis.

In this paper we measure the performance of a weak hypothesis h_t in terms of the information that h_t brings. For the entropy function, we use $G(p) = 2\sqrt{p(1-p)}$ defined for $p \in [0, 1]$ so that we can interpret the progress of boosting as the product of conditional entropies. This function is used to describe the performance of various boosting algorithms [1], [8], [10], [14]. Note that since the function G upper bounds Shannon entropy, small entropy in terms of G implies small Shannon entropy. For a probability distribution D over $\{1, \ldots, m\}$, we sometimes consider X and Y as random variables that take values x_i and y_i (resp.) with probability $D(i)$. Especially we call Y the target random variable. Let the probability that the target Y takes value 1 under D be

1): The decision tree produced by DT.InfoBoost uses the same weak hypothesis h_t as the decision rule at all nodes of depth $t-1$. So we can expect that DT.InfoBoost has less risk of overfitting than the classical top-down algorithms that produce trees with exponentially many different decision rules.

denoted by

$$p = \Pr_D(y_i = 1) = \sum_{i=1}^{m} D(i)[\![y_i = 1]\!],$$

where $[\![C]\!]$ is the indicator function that is 1 if the condition C holds and 0 otherwise. Then *the entropy of the target Y with respect to D* is defined as

$$H_D(Y) = G(p) = 2\sqrt{p(1-p)}.$$

Furthermore, for any function $g : X \to Z$ for some countable set Z, *the conditional entropy of Y given g with respect to D* is defined as

$$\begin{aligned} H_D(Y|g) &= \sum_{z \in Z} \Pr_D(g(x_i) = z) H_D(Y|g(x_i) = z) \\ &= \sum_{z \in Z} \Pr_D(g(x_i) = z) G(p_z) \end{aligned}$$

where $p_z = \Pr_D(y_i = 1 \mid g(x_i) = z)$. The entropies $H_D(Y)$ and $H_D(Y|g)$ are interpreted as uncertainty of the target before and after seeing the value of g, respectively. So the mutual information $H_D(Y) - H_D(Y|g)$ represents the amount of information that g brings. In particular, since the distribution D_t we consider in this paper always satisfies $H_{D_t}(Y) = 1$, we measure the performance of the weak hypothesis h_t by the conditional entropy $H_{D_t}(Y|h_t)$. The inequality $H_{D_t}(Y|h_t) < 1$ implies that the weak hypothesis h_t brings non-zero information about the target.

Here we give some basic properties of the entropies. Let D be a distribution over $\{1, \ldots, m\}$ and g' be another function defined on X. Then, we have

$$\begin{aligned} H_D(Y|g) &= 2 \sum_{z \in Z} \sqrt{\Pr_D(g(x_i) = z, y_i = 1)} \\ &\quad \cdot \sqrt{\Pr_D(g(x_i) = z, y_i = -1)}, \quad (1) \end{aligned}$$

$$H_D(Y|g, g') \leqq H_D(Y|g). \quad (2)$$

3. InfoBoost growing a Branching Program

In this section, we show that InfoBoost can be viewed as a top-down algorithm for growing a branching program.

First we briefly review how InfoBoost works. In each round t, when given a weak hypothesis h_t, InfoBoost updates the distribution to

$$D_{t+1}(i) = \frac{D_t(i) \exp\bigl(-\alpha_t[h_t(x_i)] h_t(x_i) y_i\bigr)}{Z_t}$$

for appropriately chosen real numbers $\alpha_t[-1]$ and $\alpha[1]$, where Z_t is for normalization. The two parameters $\alpha_t[-1]$ and $\alpha_t[1]$ are chosen so that the normalization factor Z_t is minimized. Interestingly, for these choices of parameters we have $Z_t = H_{D_t}(Y|h_t)$. The master hypothesis that InfoBoost produces is given by $\mathrm{sign}(F_T(x))$, where

```
Input: S = {(x_1, y_1), ..., (x_m, y_m)}
Initialize D_1(i) = 1/m;
for t = 1 to T do
    h_t = WL(S, D_t);
    Choose real numbers α_t[-1] and α_t[1] so that
        α_t[a] = (a/2) ln( Pr_D(h_t(x_i) = a, y_i = 1) / Pr_D(h_t(x_i) = a, y_i = -1) )
    for any a ∈ {-1, 1};
    Update D_t to D_{t+1} so that
        D_{t+1}(i) = D_t(i) exp(-α_t[h_t(x_i)] h_t(x_i) y_i) / Z_t
    holds for 1 ≦ i ≦ m, where Z_t is for normalization;
Let F_T : x ↦ Σ_{t=1}^{T} α_t[h_t(x)] h_t(x);
Output sign(F_T(x))
```

Fig. 1 The algorithm of InfoBoost.

$$F_T(x) = \sum_{t=1}^{T} \alpha_t[h_t(x)] h_t(x). \quad (3)$$

The detail of the algorithm is given in Fig. 3.. The performance of InfoBoost is summarized as in the following inequality [1]:

$$\Pr_U\bigl(\mathrm{sign}(F_T(x_i)) \neq y_i\bigr) \leqq \prod_{t=1}^{T} H_{D_t}(Y|h_t), \quad (4)$$

where U is the uniform distribution over $\{1, \ldots, m\}$. This implies that the training error of the master hypothesis decreases rapidly as T becomes large, as long as the weak hypotheses h_t bring non-zero information with respect to D_t, i.e., $H_{D_t}(Y|h_t) < 1$. In particular, if all hypotheses h_t satisfy $H_{D_t}(Y|h_t) \leqq 1 - \gamma$ for some $\gamma > 0$, then $\Pr_U\bigl(\mathrm{sign}(F_T(x_i)) \neq y_i\bigr) \leqq \epsilon$ with $T = (1/\gamma)\ln(1/\epsilon)$.

Now we interpret InfoBoost as a process of growing a branching program. At the beginning of round t (except $t = 1$), we have the set L_{t-1} of two leaves. When given h_t, each leaf $l \in L_{t-1}$ becomes an internal node with two child leaves l_{-1} and l_1, just as in a decision tree growing algorithm. Note that the same hypothesis h_t is used for the decision done at any $l \in L_{t-1}$ and the outcomes -1 and 1 correspond to the edges (l, l_{-1}) and (l, l_1), respectively. Here we have four leaves. Then for each $a \in \{-1, 1\}$, the set $\{l_a \mid l \in L_{t-1}\}$ of two leaves are merged into a single leaf and it is labeled with weight $a \cdot \alpha_t[a]$. The new leaves corresponding to $a = -1$ and $a = 1$ are called a (-1)-node and a 1-node, respectively. Thus, in the end of round t, we have the set L_t of two leaves again. So each round consists of two phases, dividing and merging the subset of the instance space. See Fig. 2.

By virtue of the branching program representation, we give natural interpretations of how the distribution D_{t+1} and the combined hypothesis F_T are determined. An instance $x \in X$ induces a path of the branching program based on the outcome sequence $h_1(x), \ldots, h_T(x)$ in the obvious way. Let $\ell_t : X \to L_t$ denote the function that maps an instance

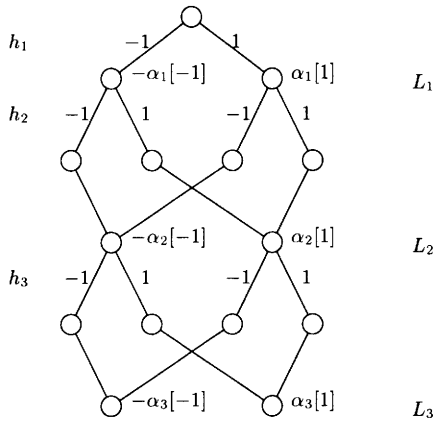

Fig. 2 A branching program that InfoBoost grows.

x to the node in L_t on the path that x induces. Note that for the branching program generated by InfoBoost, $\boldsymbol{\ell}_t(x)$ depends only on $h_t(x)$ and so $H_D(Y|\boldsymbol{\ell}_t) = H_D(Y|h_t)$ for any D. Intuitively, the distribution D_{t+1} is determined so that $\boldsymbol{\ell}_t$ (and thus h_t) is uncorrelated with Y and the total weight of leaf $l \in L_t$ is proportional to the uncertainty of Y at the leaf l. More precisely, (as we will see later for a general branching program), the distribution D_{t+1} satisfies that

$$H_{D_{t+1}}(Y|\boldsymbol{\ell}_t) = H_{D_{t+1}}(Y|h_t) = 1 \tag{5}$$

and for any $l \in L_t$,

$$\Pr_{D_{t+1}}(\boldsymbol{\ell}_t(x_i) = l) = \frac{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l) H_{D_t}(Y|\boldsymbol{\ell}_t(x_i) = l)}{H_{D_t}(Y|\boldsymbol{\ell}_t)}. \tag{6}$$

The property of (5) is essential because otherwise the next hypothesis h_{t+1} would be the same as h_t for which $H_{D_{t+1}}(Y|h_{t+1}) < 1$ but no additional information is obtained. The property of (6) is reasonable since this means that the leaf with larger entropy will be more focused with the hope that the next hypothesis h_{t+1} would reduce the whole entropy efficiently.

The combined hypothesis F_T given by (3) is also represented in terms of the branching program. For each node $l \in L_t$, let $w[l]$ denote the weight labeled at l. That is, $w[l] = -\alpha_t[-1]$ if l is a (-1)-node and $w[l] = \alpha_t[1]$ if l is a 1-node. It is easy to see that for a given instance x, $F_T(x)$ is represented as the sum of the weights of the nodes on the path induced by x. That is,

$$F_T(x) = \sum_{t=1}^{T} w[\boldsymbol{\ell}_t(x)]. \tag{7}$$

4. BP.InfoBoost

In this section, we generalize the merging scheme used by InfoBoost and propose a class of boosting algorithms called

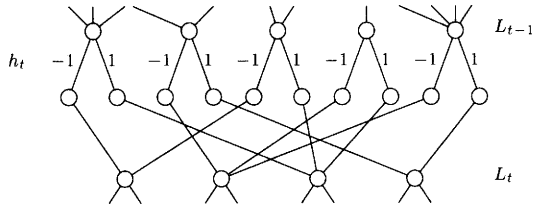

Fig. 3 A branching program that a BP.InfoBoost grows. Note that any leaf in L_t is determined by a leaf in L_{t-1} and the outcome of h_t.

BP.InfoBoost with various merging schema. Curiously, we show that the bound (4) on the training error remains to hold for BP.InfoBoost with *any* merging scheme.

First we describe how BP.InfoBoost works. For convenience, we represent the set of leaves L_t as a partition of $\{-1,1\}^t$. That is, a leaf $l \in L_t$ is a subset of $\{-1,1\}^t$. For a given partition L_t, the function $\boldsymbol{\ell}_t : X \to L_t$ is defined as

$$\boldsymbol{\ell}_t(x) = l \Leftrightarrow (h_1(x), \ldots, h_t(x)) \in l.$$

At the beginning of round t, we have the set L_{t-1} of leaves. Now L_{t-1} may contain more than two leaves. When given h_t, each leaf $l \in L_{t-1}$ becomes an internal node with two child leaves l_{-1} and l_1. Formally, the leaf l_a with $a \in \{-1,1\}$ is a subset of $\{-1,1\}^t$ and given by $l_a = \{(v,a) \mid v \in l\}$. So, an instance x that reaches a leaf $l \in L_{t-1}$ (i.e., $\boldsymbol{\ell}_{t-1}(x) = l$) goes to l_a if $h_t(x) = a$. Here we have twice as many leaves as in L_{t-1}. Then for any $a \in \{-1,1\}$ the set $\{l_a \mid l \in L_{t-1}\}$ is partitioned *somehow* into $L'_{a,1}, \ldots, L'_{a,k_a}$ for some k_a and the leaves in each $L'_{a,i}$ are merged into a single leaf, which is formally given by $\bigcup_{l_a \in L'_{a,i}} l_a$. Thus we have the set L_t of leaves of size $k_{-1} + k_1$. Fig. 3 illustrates how the branching program grows. A merging scheme is a procedure of deciding how the sets $\{l_a \mid l \in L_{t-1}\}$ are partitioned. For example, InfoBoost is a BP.InfoBoost with a particular merging scheme that lets each set $\{l_a \mid l \in L_{t-1}\}$ be a partition of itself.

Each leaf $l \in L_t$ is labeled with the weight given by

$$w[l] = \frac{1}{2} \ln \frac{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i = 1)}{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i = -1)}. \tag{8}$$

With the notations w and $\boldsymbol{\ell}_t$, we can write the update rule and the master hypothesis as in the same way as for the case of InfoBoost. That is, the distribution is updated to

$$D_{t+1}(i) = \frac{D_t(i) \exp(-w[\boldsymbol{\ell}_t(x_i)] y_i)}{Z_t} \tag{9}$$

where Z_t is for normalization, and the master hypothesis is $\mathrm{sign}(F_T(x))$, where F_T is given by (7). The details of the algorithm is given in Fig. 4.. Note that $\boldsymbol{\ell}_t$ can be viewed as a domain-partitioning weak hypothesis and the weights $w[l]$ derived in (8) are identical to the confidences assigned by Schapire and Singer [13].

Input: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
Initialize $D_1(i) = 1/m$, $L_0 = \{\epsilon\}$, $\boldsymbol{\ell}_0 : x \mapsto \epsilon$;
for $t = 1$ to T do
  $h_t = \mathrm{WL}(S, D_t)$;
  Let $l_a = \{(v, a) \mid v \in l\}$ for $l \in L_{t-1}$ and $a \in \{-1, 1\}$;
  for $a \in \{-1, 1\}$ do
    Partition $\{l_a \mid l \in L_{t-1}\}$ into $L'_{a,1}, \ldots, L'_{a,k_a}$;
  $L_t = \left\{ \bigcup_{l_a \in L'_{a,i}} l_a \;\middle|\; 1 \leq i \leq k_a, a \in \{-1, 1\} \right\}$;
  Let $\boldsymbol{\ell}_t : x \mapsto l \in L_t$ where $(h_1(x), \ldots, h_t(x)) \in l$;
  for $l \in L_t$ do
    Let $w[l] = \frac{1}{2} \ln \frac{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i = 1)}{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i = -1)}$;
  Update D_t to
    $D_{t+1}(i) = D_t(i) \exp\bigl(-w[\boldsymbol{\ell}_t(x_i)] y_i\bigr) / Z_t$
  for $1 \leq i \leq m$, where Z_t is for normalization;
Let $F_T : x \mapsto \sum_{t=1}^{T} w[\boldsymbol{\ell}_t(x)]$;
Output $\mathrm{sign}(F_T(x))$

Fig. 4 The algorithm of a BP.InfoBoost.

Now we show that a BP.InfoBoost with any merging scheme has the same form of upper bound on the training error. We need some technical lemmas. The first one is immediately obtained by recursively applying the update rule (9).

Lemma 1: *Let Z_t be the normalization factor in (9) and D_{T+1} be the distribution obtained in the final round. Then,*

$$\begin{aligned} D_{T+1}(i) &= \frac{D_1(i)}{Z_1 \cdots Z_T} \exp\left(-\sum_{t=1}^{T} w[\boldsymbol{\ell}_t(x_i)] y_i \right) \\ &= \frac{D_1(i)}{Z_1 \cdots Z_T} \exp\bigl(-F_T(x_i) y_i\bigr). \end{aligned}$$

The next lemma shows that the normalization factor Z_t can be rewritten as the conditional entropy.

Lemma 2: $Z_t = H_{D_t}(Y|\boldsymbol{\ell}_t) \leqq H_{D_t}(Y|h_t)$.

Proof. First we show the equality part. By (8) we have

$$\begin{aligned} Z_t &= \sum_{i=1}^{m} D_t(i) \exp\left(-w[\boldsymbol{\ell}_t(x_i)] y_i\right) \\ &= \sum_{l \in L_t} \sum_{a \in \{-1,1\}} \sum_{i=1}^{m} D_t(i) [\![\boldsymbol{\ell}_t(x_i) = l, y_i = a]\!] \exp(-w[l] a) \\ &= \sum_{l \in L_t} \sum_{a \in \{-1,1\}} \Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i = a) \\ &\qquad \cdot \left(\frac{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i \neq a)}{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i = a)} \right)^{1/2} \\ &= 2 \sum_{l \in L_t} \sqrt{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i = 1)} \\ &\qquad \cdot \sqrt{\Pr_{D_t}(\boldsymbol{\ell}_t(x_i) = l, y_i = -1)} \\ &= H_{D_t}(Y|\boldsymbol{\ell}_t). \end{aligned}$$

The last line is derived from (1).

For the inequality part, since the last component of any vector in the set $\boldsymbol{\ell}_t(x)$ is $h_t(x)$, we have $H_{D_t}(Y|\boldsymbol{\ell}_t) = H_{D_t}(Y|\boldsymbol{\ell}_t, h_t)$, which is at most $H_{D_t}(Y|h_t)$ by (2). □

Now we are ready to give an upper bound on the training error.

Theorem 1: *Let F_T be the combined hypothesis of a BP.InfoBoost. Then*

$$\Pr_U\bigl(\mathrm{sign}(F_T(x_i)) \neq y_i\bigr) \leqq \prod_{t=1}^{T} H_{D_t}(Y|\boldsymbol{\ell}_t) \leqq \prod_{t=1}^{T} H_{D_t}(Y|h_t).$$

Proof. By Lemma 2, it suffices to show that $\Pr_U\bigl(\mathrm{sign}(F_T(x_i)) \neq y_i\bigr) \leqq \prod_{t=1}^{T} Z_t$. Since the initial distribution D_1 is uniform, we have

$$\Pr_U\bigl(\mathrm{sign}(F_T(x_i)) \neq y_i\bigr) = \sum_{i=1}^{m} D_1(i) [\![\mathrm{sign}(F_T(x_i)) \neq y_i]\!].$$

Using $[\![\mathrm{sign}(F_T(x_i)) \neq y_i]\!] \leqq \exp\bigl(-F_T(x_i) y_i\bigr)$ and Lemma 1, we have

$$\begin{aligned} \Pr_U\bigl(\mathrm{sign}(F_T(x_i)) \neq y_i\bigr) &\leqq \sum_{i=1}^{m} D_1(i) \exp\bigl(-F_T(x_i) y_i\bigr) \\ &= Z_1 \cdots Z_T \sum_{i=1}^{m} D_{T+1}(i) \\ &= Z_1 \cdots Z_T. \end{aligned}$$

□

This theorem means that the upper bound on the training error for a BP.InfoBoost is not larger than that for InfoBoost provided that the same distributions D_t and weak hypotheses h_t are used. Moreover, when there is a large gap between $H_{D_t}(Y|\boldsymbol{\ell}_t)$ and $H_{D_t}(Y|h_t)$, then the BP.InfoBoost would improve the master hypothesis faster than InfoBoost.

Next we consider any possible construction of the master hypothesis. Since any master hypothesis $f : X \to \{-1, 1\}$ should be constructed from the weak hypotheses $h_1, \ldots, h_T$, it must hold that $H_U(Y|h_1, \ldots, h_T) \leqq H_U(Y|f)$ and so the l.h.s. gives an information-theoretic limitation of the performance of f. The next theorem shows how much the entropy $H_U(Y|h_1, \ldots, h_T)$ is reduced as T grows. For an instance x, $\boldsymbol{\ell}_{1..T}(x)$ is short for the path $(\boldsymbol{\ell}_1(x), \ldots, \boldsymbol{\ell}_T(x)) \in L_{1..T} \equiv L_1 \times \cdots \times L_T$.

Theorem 2:

$$\begin{aligned} H_U(Y|h_1, \ldots, h_T) &= H_U(Y|\boldsymbol{\ell}_{1..T}) \\ &= \left(\prod_{t=1}^{T} H_{D_t}(Y|\boldsymbol{\ell}_t) \right) H_{D_{T+1}}(Y|\boldsymbol{\ell}_{1..T}). \end{aligned}$$

Proof. The first equality holds since there is one-to-one correspondence between the set of paths $\boldsymbol{\ell}_{1..T}(x)$ of the branching program and the set of outcome sequences $\bigl(h_1(x), \ldots, h_T(x)\bigr)$.

By (1) we have

$$\begin{aligned} &H_{D_{T+1}}(Y|\boldsymbol{\ell}_{1..T}) \\ &= 2 \sum_{\vec{l} \in L_{1..T}} \sqrt{\Pr_{D_{T+1}}(\boldsymbol{\ell}_{1..T}(x_i) = \vec{l}, y_i = 1)} \end{aligned}$$

$$\cdot\sqrt{\Pr_{D_{T+1}}\left(\boldsymbol{\ell}_{1..T}(x_i)=\vec{l},y_i=-1\right)}.$$

Lemma 1 says that the distribution $D_{T+1}(i)$ depends on the path $\boldsymbol{\ell}_{1..T}(x_i)$ on which an instance x_i goes through. So for a particular path $\vec{l}=(l_1,\ldots,l_T)\in L_{1..T}$ and for any $a\in\{-1,1\}$, we have

$$\begin{aligned}
&\Pr_{D_{T+1}}\left(\boldsymbol{\ell}_{1..T}(x_i)=\vec{l},y_i=a\right)\\
&=\sum_i D_{T+1}(i)[\![\boldsymbol{\ell}_{1..T}(x_i)=\vec{l},y_i=a]\!]\\
&=\frac{\exp\left(-a\sum_{t=1}^T w[l_t]\right)}{Z_1\cdots Z_T}\sum_i D_1(i)[\![\boldsymbol{\ell}_{1..T}(x_i)=\vec{l},y_i=a]\!]\\
&=\frac{\exp\left(-a\sum_{t=1}^T w[l_t]\right)}{Z_1\cdots Z_T}\Pr_U\left(\boldsymbol{\ell}_{1..T}(x_i)=\vec{l},y_i=a\right).
\end{aligned}$$

By multiplying the above probabilities with $a=1$ and $a=-1$ the exponential terms are canceled and we get

$$H_{D_{T+1}}(Y|\boldsymbol{\ell}_{1..T})=\frac{H_U(Y|\boldsymbol{\ell}_{1..t})}{Z_1\cdots Z_T},$$

which completes the theorem. □

Theorem 1 and Theorem 2 imply that our master hypothesis $\mathrm{sign}(F(x))$ would be nearly optimal if $H_{D_{T+1}}(Y|\boldsymbol{\ell}_{1..T})$ is not too small. Although we do not know a lower bound on $H_{D_{T+1}}(Y|\boldsymbol{\ell}_{1..T})$ in general, the distribution satisfies $H_{D_{T+1}}(Y|\boldsymbol{\ell}_T)=1$ as shown in (5) for the case of InfoBoost. That is, BP.InfoBoost updates the distribution so that $\boldsymbol{\ell}_T$ is uncorrelated with Y. The next lemma shows that a BP.InfoBoost makes D_{t+1} so that $\boldsymbol{\ell}_t$ is uncorrelated with the sample. This implies that the weak hypothesis h_{t+1} (and $\boldsymbol{\ell}_{t+1}$ as well) with non-zero correlation must have some information that the function $\boldsymbol{\ell}_t$ does not have. This gives an intuitive justification why the improvement of the master hypothesis for a BP.InfoBoost is upper bounded by the product of $H_{D_t}(Y|\boldsymbol{\ell}_t)$.

Lemma 3: *For any $1\leqq t\leqq T$, a BP.InfoBoost updates the distribution to D_{t+1} so that $H_{D_{t+1}}(Y|\boldsymbol{\ell}_t)=1$ holds.*

Proof. It suffices to show that for any leaf $l\in L_t$, $\Pr_{D_{t+1}}(y_i=1\mid\boldsymbol{\ell}_t(x_i)=l)=1/2$, or equivalently

$$\Pr_{D_{t+1}}\left(\boldsymbol{\ell}_t(x_i)=l,y_i=1\right)=\Pr_{D_{t+1}}\left(\boldsymbol{\ell}_t(x_i)=l,y_i=-1\right). \tag{10}$$

By (9) and (8), it is straightforward to see that the both sides of (10) are equal to

$$\frac{\sqrt{\Pr_{D_t}\left(\boldsymbol{\ell}_t(x_i)=l,y_i=1\right)\Pr_{D_t}\left(\boldsymbol{\ell}_t(x_i)=l,y_i=-1\right)}}{Z_t}$$

□

Note that [7] also suggests to make D_{t+1} be uncorrelated with $\boldsymbol{\ell}_t$ but in a trivial way: D_{t+1} is restricted to a single leaf l and set so that $\Pr_{D_{t+1}}(y_i=1\mid l)=1/2$. This means that the weak leaner must be invoked for all leaves in L_t with different distributions. On the other hand, our distribution (9) assigns positive weights to all nodes l unless $H_{D_t}(Y\mid\boldsymbol{\ell}_t(x_i)=l)=0$, and the weak leaner is invoked only once for each depth t.

5. DT.InfoBoost

As we stated, InfoBoost is a BP.InfoBoost with a particular merging scheme that merges all nodes l_1 into one leaf and all nodes l_{-1} into the other leaf. Since $H_{D_t}(Y|\boldsymbol{\ell}_t)=H_{D_t}(Y|h_t)$, InfoBoost takes no advantage of the branching program representation.

Next we consider the BP.InfoBoost with no merging phase, namely, DT.InfoBoost. In this case, since a path of the decision tree is uniquely determined by the leaf of the path, we have $H_{D_t}(Y|\boldsymbol{\ell}_t)=H_{D_t}(Y|h_1,\ldots,h_t)$. Since $H_{D_t}(Y|h_1,\ldots,h_t)$ is likely to be much smaller than $H_{D_t}(Y|h_t)$, DT.InfoBoost would improve the master hypothesis more efficiently than InfoBoost. Lemma 3 implies

$$H_{D_t1}(Y|h_1,\ldots,h_t)=H_{D_{t+1}}(Y|\boldsymbol{\ell}_t)=1.$$

So, DT.InfoBoost has a totally corrective update. This means that all the hypotheses $h_1,\ldots,h_t$ obtained so far are uncorrelated with Y. So the next hypothesis h_{t+1} will bring truly new information that $h_1,\ldots,h_t$ do not have, provided $H_{D_{t+1}}(Y|h_{t+1})<1$. This contrasts with the case of InfoBoost, where h_{t+1} is only guaranteed to bring information that h_t does not have. Moreover, with the fact that $H_{D_{T+1}}(Y|\boldsymbol{\ell}_T)=1$, Theorem 1 and Theorem 2 gives the performance of DT.InfoBoost nicely as

$$\begin{aligned}
\Pr_U\left(\mathrm{sign}(F_T(x_i))\neq y_i\right)&\leqq H_U(Y|h_1,\ldots,h_T)\\
&=\prod_{t=1}^T H_{D_t}(Y|h_t).
\end{aligned}$$

This implies that DT.InfoBoost extracts information very effectively from the weak learner. Actually, we observe in experiments that DT.InfoBoost reduces the generalization error as well as the training error very rapidly in early rounds. Unfortunately, however, the generalization error sometimes turns to be increasing in later rounds. This is because the complexity of the master hypothesis grows exponentially in the number of rounds.

Generally, there is a tradeoff between the size of L_t and $H_{D_t}(Y|\boldsymbol{\ell}_t)$. More precisely, the larger the size of L_t is (i.e, the less nodes we merge), the smaller $H_{D_t}(Y|\boldsymbol{\ell}_t)$ becomes (i.e., the more rapidly the training error decreases). In the next section, we propose a merging scheme that makes $H_{D_t}(Y|\boldsymbol{\ell}_t)$ nearly as small as the one we would have without merge, while keeping L_t in a moderate size.

6. A merging scheme

Assume we are given a weak hypothesis h_t in the tth round

and each node $l \in L_{t-1}$ grows two child leaves l_1 and l_{-1}. Now we have twice as many leaves as L_{t-1}. Let the set of leaves be denoted by

$$\tilde{L}_t = \{l_1 \mid l \in L_{t-1}\} \cup \{l_{-1} \mid l \in L_{t-1}\}$$

and let $\tilde{\boldsymbol{\ell}}_t : X \to \tilde{L}_t$ be defined analogously, that is, $\tilde{\boldsymbol{\ell}}_t(x)$ is the leaf in $\tilde{L}_t$ that instance x reaches. If we merge no nodes in $\tilde{L}_t$, then we would have $L_t = \tilde{L}_t$ and $\tilde{\boldsymbol{\ell}}_t = \boldsymbol{\ell}_t$. So, $H_{D_t}(Y|\tilde{\boldsymbol{\ell}}_t)$ gives a lower bound on $H_{D_t}(Y|\boldsymbol{\ell}_t)$ for $\boldsymbol{\ell}_t$ induced by any merging for $\tilde{L}_t$. Let $H_{D_t}(Y|\tilde{\boldsymbol{\ell}}_t) = 1 - \gamma_t$. The merging scheme we give guarantees that $H_{D_t}(Y|\boldsymbol{\ell}_t) \leqq 1 - c\gamma_t$ for some constant $0 < c < 1$. For this purpose, we could use the merging method developed by Mansour and McAllester [10].

[Lemma 4] ([10]) For any function $f : X \to Z$, any distribution D over $X \times Y$, and for any $0 < \lambda, \delta < 1$, there exists a function $g : Z \to M$ such that

$$H_D(Y|g \circ f) \leqq (1 + \lambda)H_D(Y|f) + \delta$$

and

$$|M| = O((1/\lambda)\log(1/\delta)).$$

Clearly, letting $\tilde{\boldsymbol{\ell}}_t$, $\tilde{L}_t$ and L_t correspond to f, Z and M, respectively, we have a merging scheme induced by g so that $g \circ f = \boldsymbol{\ell}_t$. So by choosing $\lambda = (1 - c)\gamma_t/2(1 - \gamma_t)$ and $\delta = (1 - c)\gamma_t/2$, we have $H_{D_t}(Y|\boldsymbol{\ell}_t) \leqq 1 - c\gamma_t$. Unfortunately, however, the size of L_t is too big, i.e., $|L_t| = O((1/\gamma_t)\ln(1/\gamma_t))$. In the following, we give a new merging method that guarantees the same bound on $H_{D_t}(Y|\boldsymbol{\ell}_t)$ with significantly small L_t, i.e., $|L_t| = O(\ln(1/\gamma_t))$.

Theorem 3: *Let $f : X \to Z$ with $H_D(Y|f) = 1 - \gamma$ for some $0 < \gamma < 1$. Then, for any $0 < c < 1$, there exists a function $g : Z \to M$ such that*

$$H_D(Y|g \circ f) \leqq 1 - c\gamma$$

and

$$|M| = O\left(\frac{\log(1/((1-c)\gamma))}{\log(1 + 1/c)}\right).$$

Proof. For each $z \in Z$, let

$$p_z = \Pr_D(f(x_i) = z) \quad \text{and} \quad q_z = \Pr_D(y_i = 1 \mid f(x_i) = z).$$

Note that

$$H_D(Y|f) = \sum_{z \in Z} p_z G(q_z) = 1 - \gamma, \tag{11}$$

where $G(q) = 2\sqrt{q(1-q)}$ is our entropy function. Let $a = 2c/(1 - c)$, $\epsilon_0 = 0$ and for each $j \geqq 1$, let

$$\epsilon_j = \left(\frac{1 + a}{a}\right)^{j-1} \frac{c\gamma}{a}.$$

Let k be the smallest integer such that $\epsilon_k \geqq 1$. Now we define the function g. For each $1 \leqq j \leqq k$, let

$$S_{-j} = \{z \in Z \mid p_z < 1/2, \epsilon_{j-1} \leqq 1 - G(q_z) < \epsilon_j\}$$

and

$$S_j = \{z \in Z \mid p_z \geqq 1/2, \epsilon_{j-1} \leqq 1 - G(q_z) < \epsilon_j\}.$$

That is, Z is partitioned into $S_{-k} \cup \cdots \cup S_k$. Let $M = \{j, -j \mid 1 \leqq j \leqq k\}$ and for any $z \in Z$, let $g(z) = j$ such that $z \in S_j$. It is easy to see that $|M| = O\left(\frac{\log(1/((1-c)\gamma))}{\log(1+1/c)}\right)$. So it suffices to show that $H_D(Y|g \circ f) \leqq 1 - c\gamma$.

For each j, let

$$p(S_j) = \sum_{z \in S_j} p_z \quad \text{and} \quad \mu_j = \sum_{z \in S_j} \tilde{p}_z q_z,$$

where $\tilde{p}_z = p_z/p(S_j)$ with $z \in S_j$. Then,

$$H_D(Y|g \circ f) = \sum_{j \in M} p(S_j) G(\mu_j).$$

Since $G(\mu_j) \leqq 1 - \epsilon_{j-1}$, we have

$$\sum_{j \in M} p(S_j) G(\mu_j) \leqq 1 - \sum_{j \in M} p(S_j)\epsilon_{j-1}.$$

If $\sum_{j \in M} p(S_j)\epsilon_{j-1} \geqq c\gamma$, then we are done. So in what follows, we assume

$$\sum_{j \in M} p(S_j)\epsilon_{j-1} < c\gamma. \tag{12}$$

Since both $G(\mu_j)$ and $\sum_{z \in S_j} \tilde{p}_z G(q_z)$ are in the range $(1 - \epsilon_j, 1 - \epsilon_{j-1}]$, we have

$$G(\mu_j) \leqq \sum_{z \in S_j} \tilde{p}_z G(q_z) + \epsilon_j - \epsilon_{j-1}$$

and this with (11) gives

$$H_D(Y|g \circ f) \leqq 1 - \gamma + \sum_{j \in M} p(S_j)(\epsilon_j - \epsilon_{j-1}). \tag{13}$$

By our choices of ϵ_j,

$$\epsilon_{j-1} = a(\epsilon_j - \epsilon_{j-1})$$

holds for $j \geqq 2$. Plugging this into (13) and using (12), we get

$$H_D(Y|g \circ f) \leqq 1 - \gamma + \frac{c\gamma}{a} + \epsilon_1(p(S_{-1}) + p(S_1)) \leqq 1 - c\gamma.$$

□

Again, letting $\tilde{\boldsymbol{\ell}}_t$ and $\boldsymbol{\ell}_t$ correspond to f and $g \circ f$, respectively, we have a merging scheme as desired. Intuitively, the parameter c controls the tradeoff between $H_{D_t}(Y|\boldsymbol{\ell}_t)$ and $|L_t|$. That is, if we let c close to 1, then $H_{D_t}(Y|\boldsymbol{\ell}_t)$ is as small as $H_{D_t}(Y|\tilde{\boldsymbol{\ell}}_t)$ while $|L_t|$ is unbounded. This implies that the BP.InfoBoost behaves like DT.InfoBoost. On the other hand, if we let c close to 0, then $H_{D_t}(Y|\boldsymbol{\ell}_t)$ is unbounded (naturally bounded by $H_{D_t}(Y|h_t)$) while $|L_t|$ is as small as a constant. Thus, the BP.InfoBoost behaves like InfoBoost. So, we can expect that BP.InfoBoost with appropriate choice of parameter c outperforms both InfoBoost and DT.InfoBoost.

Finally, we remark that we develop the merging method

in a different philosophy from the one of Mansour and McAllester [10] in the following two points.

- Our merging scheme is based on the current distribution D_t while that of [10] is based on the uniform distribution over the sample.
- Our merging scheme aims to make the function $\boldsymbol{\ell}_t$ perform not much worse than $\tilde{\boldsymbol{\ell}}_t$ as a *weak* hypothesis. On the other hand, [10] aims to make $\boldsymbol{\ell}_t$ perform getting better as a *master* hypothesis, so that $H_U(Y|\boldsymbol{\ell}_t)$ converges to 0. So their method needs much more leaves and the complexity of the branching program becomes unnecessarily large.

7. Remarks and future works

(1) Our analysis suggests that the weak learner should produce h_t so that $H_{D_t}(Y|\boldsymbol{\ell}_t)$, rather than $H_{D_t}(Y|h_t)$, is as small as possible.

(2) Our algorithm uses the same cost function $\sum_i e^{-F_T(x_i)y_i}$ as AdaBoost and InfoBoost. This seems to nicely mesh with our entropy function $G(q) = 2\sqrt{q(1-q)}$. What is the relation between the choices of cost functions and corresponding entropy functions?

(3) The combined hypothesis $F_T(x) = \sum_{t=1}^{T} w[\boldsymbol{\ell}_t(x)]$ can be rewritten as the dot product $F_T(x) = W \cdot H(x)$ where W and $H(x)$ are vectors indexed by paths $\sigma \in L_1 \times \cdots \times L_T$ and its $\sigma = (\sigma_1, \ldots, \sigma_T)$th components are $W_\sigma = \sum_{t=1}^{T} w[\sigma_t]$ and $H_\sigma(x) = \prod_{t=1}^{T} [\![\boldsymbol{\ell}_t(x) = \sigma_t]\!]$, respectively. This defines feature maps from the node space to the path space. Can we analyze BP.InfoBoost in terms of the path space?

(4) We need to give a criterion of choosing the tradeoff parameter c (that may depend on round t).

(5) We need to analyze the complexity of the master hypothesis, say, in terms of VC dimension.

(6) We are evaluating the performance of BP.InfoBoost on data sets from the UCI Machine Learning Repository. Preliminary experiments show that the BP.InfoBoost with the merging scheme developed in Section 6. performs well as compared to InfoBoost and DT.InfoBoost.

Acknowledgments

The authors are grateful to Tatsuya Watanabe for showing a preliminary version of merging method for Theorem 3.

References

[1] J. A. Aslam. Improving algorithms for boosting. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 200–207. Morgan Kaufmann, San Francisco, 2000.

[2] L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493-1518, 1999.

[3] C. Domingo and O. Watanabe. MadaBoost: A modification of AdaBoost. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 180–189. Morgan Kaufmann, San Francisco, 2000.

[4] N. Duffy and D. Helmbold. A geometric approach to leveraging weak learners. *Theoret. Comput. Sci.*, 284(1):67–108, 2002.

[5] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 2:337–374, 2000.

[6] A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *15th AAAI*, pages 692–699, 1998.

[7] A. Kalai and R. A. Servedio. Boosting in the presence of noise. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 9-11, 2003*, pages 196–205. ACM Press, 2003.

[8] M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. *J. of Comput. Syst. Sci.*, 58(1):109–128, 1999.

[9] J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 134–144. ACM Press, New York, NY, 1999.

[10] Y. Mansour and D. A. McAllester. Boosting using branching programs. *J. of Comput. Syst. Sci.*, 64(1):103–112, 2002. Special Issue for COLT 2000.

[11] G. Rätsch and M. K. Warmuth. Maximizing the margin with boosting. In *15th Annual Conference on Computational Learning Theory, COLT 2002, Sydney, Australia, July 2002, Proceedings*, volume 2375 of *Lecture Notes in Artificial Intelligence*, pages 334–350. Springer, 2002.

[12] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

[13] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[14] E. Takimoto and A. Maruoka. Top-down decision tree learning as information based boosting. *Theoret. Comput. Sci.*, 292(2):447–464, 2003.