A Simple Boosting Algorithm Using Multi-Way Branching Decision Trees

Hatano, Kohei
Department of Mathematical and Computing Sciences, Tokyo Institute of Technology

出版情報：Theory of Computing Systems. 37 (4), pp.503-518, 2004-07. Springer-Verlag
バージョン：
権利関係：

A Simple Boosting Algorithm Using Multi-Way Branching Decision Trees

Kohei Hatano
Department of Mathematical and Computing Sciences,
Tokyo Institute of Technology,
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552, Japan
E-mail: hatano@is.titech.ac.jp

April 3, 2003

Abstract

We improve the analysis of the decision tree boosting algorithm proposed by Mansour and McAllester. For binary classification problems, the algorithm of Mansour and McAllester constructs a multi-way branching decision tree using a set of multi-class hypotheses. Mansour and McAllester proved that it works under certain conditions. We give a rigorous analysis of the algorithm and simplify the conditions. From this simplification, we can provide a simpler algorithm, for which no prior knowledge on the quality of weak hypotheses is necessary.

1 Introduction

Boosting is a technique to construct a "strong" hypothesis combining many "weak" hypotheses. This technique was first proposed by Schapire [11] to prove the equivalence between strong and weak learnability in PAC-learning. Many researchers have proposed various improved boosting techniques such as AdaBoost [5] and so on. (See for example, [4, 3, 12, 13, 8, 10].) Among them, Kearns and Mansour [7] showed that the learning process of well-known decision tree learning algorithms such as CART [2] and C4.5 [9] can

be regarded as boosting, thereby giving some theoretical justification to those popular decision tree learning tools.

More precisely, Kearns and Mansour formalized the process of constructing a decision tree as the following boosting algorithm. For any binary classification problem, let H_2 be a set of binary "branching functions" for this classification problem. Starting from the trivial single-leaf decision tree, the learning algorithm improves the tree by replacing some leaf of the tree (chosen according to a certain rule) with an internal labeled a branching function $h \in H_2$ (again chosen according to a certain rule). It is shown that the algorithm outputs a tree T with its training error below $s^{-\gamma}$, where s is the number of leaves of T, provided that for any distribution, there always exists some branching function in H_2 whose "advantage" is larger than γ $(0 < \gamma \leq 1)$ for the classification problem. This implies that $(1/\varepsilon)^{(1/\gamma)}$ steps are sufficient for the desired training error ε. (See the next section for the detail; in particular, the definition of "advantage".)

There are two extensions of the result of Kearns and Mansour. Takimoto and Maruoka generalized the algorithm for multi-class learning [14]. Their algorithm uses, for any fixed $K \geq 2$, K-way branching functions. (For a detailed analysis, see also [6].) On the other hand, Mansour and McAllester gave a generalized algorithm that constructs a decision tree (for binary classification) by using multi-way branching functions. That is, their algorithm may construct a decision tree having nodes with different number of branches.

In this paper, we improve the analysis of Mansour and McAllester's algorithm.

Consider the situation of constructing a decision tree of size s for a given s by using multi-way branching functions. Mansour and McAllester showed that their algorithm produces a size s decision tree with training error bound $s^{-\gamma}$ under the following condition.

<u>The condition of Mansour and McAllester</u>

At each boosting step, there always exists a branching function h satisfying the following:

(1) h is either binary (in which case $k = 2$) or k-way branching function with some k, $k > 2$, such that $k \leq (s/s')(\gamma e_\gamma(k)/2)$, where s' is the current decision tree size, and

(2) h has advantage larger than $\gamma g_\gamma(k)$.

Here g_γ and e_γ are defined by

$$g_\gamma(k) \;=\; \frac{1-e_\gamma(k)}{\gamma} \;\approx\; \ln k, \quad \text{and} \quad e_\gamma(k) \;=\; \prod_{i=1}^{k-1}\left(1-\frac{\gamma}{i}\right).$$

This result intuitively means that if we can assume, at each boosting step, some multi-way branching function that is better than a binary branching function with advantage γ, then the algorithm produces a tree that is as good as the one produced by the original boosting algorithm using only binary branching functions with advantage γ. (Note that $g_\gamma(2) = 1$ by definition.)

We give a detailed analysis, thereby obtaining the following condition, which also makes the algorithm simpler. (Here we consider the same situation and the goal as above.)

Our condition

At each boosting step, there always exists a branching function h satisfying the following:

(1) h is either binary (in which case $k = 2$) or k-way branching function with some k, $k > 2$, such that $k \leq s/s'$, and

(2) h has advantage larger than $\gamma\lceil \log k \rceil$.

This condition is simpler, and the above explained intuition becomes clearer under this condition. The item (2) of this new condition means that k-way branching function h is better than an "equivalent" depth $\lceil \log k \rceil$ decision tree consisting of binary branching functions with advantage γ. That is, if we can always find such a multi-way branching function at each boosting step, then the algorithm produces a tree that is as good as the one produced by the original boosting algorithm using only binary branching functions with advantage γ.

In fact, based on this new interpretation, we propose to compare the quality of weak k-way branching functions for different k based on the quantity computed as the information gain over $\lceil \log k \rceil$. This simplifies the original algorithm of Mansour and McAllester, and moreover, by this modification, we no longer need to know a lower bound γ for the advantage of binary weak hypotheses (i.e., branching functions).

Technically, the item (2) of our condition is stronger (i.e., worse) than the original one; this is because $\lceil \log k \rceil \geq g_\gamma(k)$. But the item (1) of our condition is weaker (i.e., better) than the original one.

In our argument, we introduce *Weight Distribution Game* for analyzing the worst-case error bound.

2 Preliminaries

2.1 Learning Model

We introduce our learning model briefly. Our model is based on PAC learning model proposed by Valiant[15]. Let X denote an instance space. We assume the unknown target function $f : X \to \{0, 1\}$. The learner is given a *sample* S of m labeled examples, $S = (\langle x_1, f(x_1)\rangle, \ldots, \langle x_1, f(x_m)\rangle)$, where each x_i is drawn independently randomly with respect to an unknown distribution P over X. The goal of the learner is, for any given constants ε and δ ($0 < \varepsilon, \delta < 1$), to output a hypothesis $h_f : X \to \{0, 1\}$ such that its *generalization error* $\epsilon(h_f) \stackrel{\text{def}}{=} \Pr_P[f(x) \neq h_f(x)]$ is below ε, with probability at least $1 - \delta$.

In order to accomplish the goal, it is sufficient to design learning algorithms based on "Occam Razor."[1]. Namely, it is sufficient to construct a learning algorithm that outputs a hypothesis h_f satisfying the following conditions: For sufficiently large sample, (1) h_f's *training error* $\hat{\epsilon}(h_f) \stackrel{\text{def}}{=} \Pr_D[f(x) \neq h_f(x)]$ is small, where D is the uniform distribution over S, and (2) $size(h_f) = o(m)$, where $size(\cdot)$ represents the length of the bit string for h_f under some fixed encoding scheme.

In this paper we consider decision tree learning, i.e., the problem of constructing a decision tree satisfying the above PAC learning criteria. More precisely, for a given target f, we would like to obtain some decision tree T representing a hypothesis h_T whose generalization error is bounded by a given ε (with high probability). Note that if a hypothesis is represented as a decision tree, the second condition of Occam learning criterion can be interpreted as the number of leaves of the tree being sufficiently small with respect to the size of the sample. By the Occam Razor approach mentioned above, we can construct a decision tree learning algorithm that meets the criteria.

2.2 Decision Trees

First we define some formal notations on decision trees.

We assume a set H of branching functions where each $h \in H$ is a function from X to a finite set R_h, and we also assume $|R_h| \geq 2$. We allow each $h \in H$ to have different range.

A H-tree T is a rooted tree in which each internal node is labeled with a branching function $h \in H$, and each leaf is labeled with 0 or 1. Each internal node in T has $|R_h|$ child nodes corresponding to the values of h, where child nodes are either internal nodes or leaves. We denote the set of H-trees as $\mathcal{T}(H)$.

A H-tree can be interpreted as a binary classifier in the following way: When an instance $x \in X$ is given, it visits the root node and tested by the labeled branching function at first. Next it goes to the child node corresponding the value of the function. Finally, by repeating this procedure, x reaches some leaf. The answer of the tree for x is the label of the leaf.

For any H-tree T, let $In(T)$ and $L(T)$ be the set of internal nodes of T, and the set of leaves of T respectively. We define $N(T) = In(T) \cup L(T)$, i.e., the set of all nodes in T. For any node $n \in N(T)$, depth of n is the length of path between the root node and n. Especially, the root node has depth 0. Depth of a H-tree is the depth of its node that has the maximum depth among all nodes.

Let S be a given *sample*, a set of examples of f, of size m. For any S and any node $n \in N(T)$, we denote S_n be the set of examples $\langle x, f(x) \rangle$ in S such that x reaches n. We define $\widehat{p_n} \stackrel{\text{def}}{=} |S_n|/|S|$ and $\widehat{q_n} \stackrel{\text{def}}{=} |\{\langle x, f(x) \rangle \in S_n | f(x) = 1\}|/|S_n|$ for any node $n \in N(T)$.

We assume how to label leaves of T when given a sample S: The label of each leaf $\ell \in L(T)$ is 0 if $q_\ell \leq 1/2$, otherwise, 1. Then the training error of T, which we denote $\hat{\epsilon}(T)$, is calculated as follows:

$$\hat{\epsilon}(T) = \sum_{\ell in L(T)} \widehat{p}_\ell \min\{\widehat{q}_\ell, 1 - \widehat{q}_\ell\}.$$

2.3 Weak Hypothesis and Boosting

Now we introduce the notion of boosting. In the classical definition, boosting is to construct a hypothesis h_f such that $\Pr_D[f(x) \neq h_f(x)] \leq \varepsilon$ for any given ε combining hypotheses $h_1, \ldots, h_t$, where each h_i satisfies that $\Pr_{D_i}[f(x) \neq h_i(x)] \leq 1/2 - \gamma$ for some distribution D_i over X ($i = 1, \ldots, t$, for some $t \geq 1$). However, in this paper, we measure the goodness of hypotheses (i.e.,

branching function) from an information-theoretic point of view. For this we use the *index function* defined by Kearns and Mansour[7].

Definition 1. A function $I : [0,1] \to [0,1]$ is a *index function* if, for any $q \in [0,1]$,

1. $\min\{q, 1-q\} \leq I(q)$,
2. $I(0) = I(1) = 0$, and $I(1/2) = 1$.

For example, Shannon's entropy function $q\log(1/q)+(1-q)\log(1/(1-q))$, which is used in C4.5[9], is an example of an index function. Now we define $I(T)$ as follows:

$$I(T) \stackrel{\text{def}}{=} \sum_{\ell \in L(T)} \widehat{p}_\ell I(\widehat{q}_\ell).$$

We note that, by the definition of the index function,

$$\hat{\varepsilon}(T) = \sum_{\ell in L(T)} \widehat{p}_\ell \min\{\widehat{q}_\ell, 1-\widehat{q}_\ell\} \leq \sum_{\ell in L(T)} \widehat{p}_\ell I(\widehat{q}_\ell) = I(T).$$

For any sample W, let $q_W = |\{\langle x, f(x)\rangle \in W | f(x) = 1\}|/|W|$. For any function $h \in H$, T_h be the H-tree that consists of a single internal node labeled with h and $|R_h|$ leaves. Let $I_W(T_h)$ be the value of $I(T_h)$ measured with respect to the sample W.

Following relationship between "error-based" and "information-based" hypotheses was first proved by Kearns and Mansour [7].

Lemma 1 (Kearns and Mansour [7]). Suppose $I(q) = \sqrt{q(1-q)}$. For any sample W, if there exists a branching function $h : X \to \{0,1\}$ such that $\Pr_D[f(x) \neq h(x)] \leq 1/2 - \delta$, then there exists a branching function $h' : X \to \{0,1\}$ such that

$$I(q_W) - I_W(T_h) \geq \frac{\delta^2}{16} I(q_W).$$

Motivated from the above lemma, we state our assumption. We assume a set of "information-based weak hypotheses" of the target function f.

Definition 2. Let f be any boolean function over X . Let $I : [0,1] \to [0,1]$ be any index function. Let H be any set of branching functions. H and I satisfy the *γ-weak hypothesis assumption* for f if for any sample W, there exists a branching function $h \in H$ satisfying

$$I(q_W) - I_W(T_h) \geq \gamma I(q_W),$$

where $0 < \gamma \leq 1$. We call the fraction $(I(q_W) - I_W(T_h))/I(q_W)$ *advantage* of the branching function h with respect to W and I. We refer to the reduction $I(q_W) - I_W(T_h)$ as *gain* .

3 Learning Binary Decision Trees

Before studying the multi-way branching decision tree learning algorithm, we explain a binary decision tree learning algorithm proposed by Kearns and Mansour [7].

For the binary target function f, this algorithm constructs binary decision trees. We assume that the algorithm is given some index function I and a set H_2 of binary branching functions $h : X \to R_h$, $|R_h| = 2$ in advance. We call the algorithm $\mathbf{TOPDOWN_{I,H_2}}$. The description of $\mathbf{TOPDOWN_{I,H_2}}$ is given in Figure 1.

The algorithm makes a local change to the tree T in order to reduce $I(T)$. At each local change, the algorithm chooses a leaf $\ell \in L(T)$ and a branching function $h \in H_2$ and replaces ℓ with a new internal node labeled h (and its two new child leaves). The tree obtained in this way is denoted $T_{\ell,h}$. This $T_{\ell,h}$ has one more leaf than T.

We explain the way to choose ℓ and h at each local change. The algorithm chooses ℓ that maximizes $\widehat{p}_\ell I(\widehat{q}_\ell)$; it calculates a sample S_ℓ that is a subset of S reaching ℓ. Finally the algorithm chooses $h \in H_2$ that maximizes the gain $I(\widehat{q}_\ell) - I_{S_\ell}(T_h)$. Note that $I(T) - I(T_{\ell,h}) = \widehat{p}_\ell(I(\widehat{q}_\ell) - I_{S_\ell}(T_h))$. Thus, if the gain is positive, then we can reduce the value of the index function.

For the efficiency of this boosting algorithm, Kearns and Mansour showed the following result.

Theorem 2 (Kearns and Mansour [7]). Assume that H_2 and I satisfy the γ-weak hypothesis assumption. Then, $\mathbf{TOPDOWN_{G,H_2}}(S, s)$ outputs T with

$$\hat{\epsilon}(T) \leq I(T) \leq s^{-\gamma}.$$

TOPDOWN$_{\mathbf{I},\mathbf{H_2}}(S, s)$
begin
  $T \leftarrow$ the single leaf tree;
  While $|T| < s$ **times do**
    $\ell \leftarrow \arg\max_{\ell \in L(T)} \widehat{p}_\ell I(\widehat{q}_\ell)$;
    $h \leftarrow \arg\max_{h \in H_2} I(\widehat{q}_\ell) - I_{S_\ell}(T_h)$;
    $T \leftarrow T_{\ell,h}$;
  end-while
  Output T ;
end.

Figure 1: Algorithm **TOPDOWN$_{\mathbf{I},\mathbf{H_2}}$**

4 Learning Multi-Way Branching Decision Trees

We propose a simpler version of Mansour and McAllester's algorithm, which constructs multi-way branching decision trees. We weaken and simplify some technical conditions of their algorithm.

Let H be a set of branching functions where each $h \in H$ is a function from X to R_h $(2 \leq |R_h|)$. We assume that H contains a set of binary branching function H_2. The algorithm is given H and some index function $I : [0,1] \to [0,1]$ beforehand. We call the algorithm **TOPDOWN-M$_{\mathbf{I},\mathbf{H}}$**. **TOPDOWN-M$_{\mathbf{I},\mathbf{H}}$**, given a sample S and an integer $s \geq 2$ as input, outputs a multi-way branching decision tree T with $|T| = s$.

The algorithm is a generalization of **TOPDOWN$_{\mathbf{I},\mathbf{H_2}}$**. One of the main modification is the criterion to choose branching functions. The algorithm **TOPDOWN-M$_{\mathbf{I},\mathbf{H}}$** chooses the branching $h : X \to R_h$ that maximizes the gain over $\lceil \log |R_h| \rceil$, not merely comparing the gain. Because the given size of the tree is limited, in order to reduce $I(T)$ as much as possible, it is natural to choose a branching function with smaller range among branching functions that have the same amount of gain. On the other hand, the criterion that Mansour and McAllester's algorithm uses is the gain over $g_\gamma(|R_h|)$. Note that it is necessary to know γ to compute g_γ.

The other modification is a constraint of branching functions with respect to the size of the tree. We say that a branching function h is *acceptable* for

TOPDOWN-$\mathbf{M}_{\mathbf{I},\mathbf{H}}(S, s)$
begin
  $T \leftarrow$ the single-leaf tree;
  While $(|T| < s)$ **do**
    $\ell \leftarrow \arg\max_{\ell \in L(T)} \widehat{p}_\ell I(\widehat{q}_\ell)$;
    $h \leftarrow \arg\max_{\substack{h \in H, \\ \text{acceptable for } T \text{ and } s}} \frac{I(\widehat{q}_\ell) - I_{S_\ell}(T_h)}{\lceil \log |R_h| \rceil}$
    $T \leftarrow T_{\ell,h}$;
    end-while
  Output T ;
end.

Figure 2: Algorithm **TOPDOWN-$\mathbf{M}_{\mathbf{I},\mathbf{H}}$**

tree T and target size s if either $|R_h| = 2$ or $2 < |R_h| \leq s/|T|$.

Note that if $|T| \geq s/2$, then only binary branching functions are acceptable. As H contains H_2, the algorithm can always select a branching function that is acceptable for any T and s. We show the details of the algorithm in Figure 2. One can observe that if H_2 satisfies the γ-weak hypothesis assumption, then there always exists a binary branching function $h_2 \in H_2$ satisfying $I(\widehat{q}_\ell) - I_{S_\ell}(T_h) \geq \gamma I(\widehat{q}_\ell)$. Therefore, if a branching function $h : X \to R_h$ is selected for ℓ, then we have

$$\begin{aligned} \frac{I(\widehat{q}_\ell) - I_{S_\ell}(T_h)}{\lceil \log |R_h| \rceil} &= \max_{\substack{h' \in H, \\ \text{acceptable for } T \text{ and } s}} \frac{I(\widehat{q}_\ell) - I_{S_\ell}(T_{h'})}{\lceil \log |R_h| \rceil} \\ &\geq I(\widehat{q}_\ell) - I_{S_\ell}(T_{h_2}) \\ &\geq \gamma I(\widehat{q}_\ell). \end{aligned}$$

We give an analysis for the algorithm **TOPDOWN-$\mathbf{M}_{\mathbf{I},\mathbf{H}}$**.

First we define some notations. For any node $n \in N(T)$, we define *weight* w_n as

$$w_n \stackrel{\text{def}}{=} \widehat{p}_n I(\widehat{q}_n).$$

The weight of a tree is just the total weight of all its leaves. Then by definition, we immediately have the following relations.

Fact 1. 1. $\hat{\epsilon}(T) \leq \sum_{\ell \in L(T)} w_\ell$.

2. If H_2 and I satisfy γ-weak hypothesis assumption, then for any node n and weights of n's child nodes $w_1, \ldots, w_k$ ($k \geq 2$), we have $w_1 + \cdots + w_k \leq (1 - \gamma\lceil \log k \rceil) w_n$.

From Fact 1 (1), in order to bound the training error of the tree, it suffices for us to consider the weight of the tree. On the other hand, it follows from Fact 1 (2) that at each boosting step, the weight of the tree gets decreased by at least $\gamma\lceil \log k \rceil w_\ell$, provided that a leaf ℓ is "expanded" by this boosting step and a k-way branching function is chosen for ℓ. Thus, we need to analyze how the weight of the tree gets decreased under the situation that, at each boosting step, (i) the leaf ℓ of the largest weight is selected and expanded, and (ii) the weight gets decreased *exactly* by $\gamma\lceil \log k \rceil w_\ell$, when a k-way branching function is chosen for ℓ. That is, we consider the worst-case under the situation and discuss how the tree's weight gets decreased in the worst-case. Notice that the only freedom left here is (i) the number of child nodes k under the constraint $k = 2$ or $2 < k \leq s/|T|$, and (ii) the distribution of the weights $w_1, \ldots, w_k$ of child leaves of ℓ under the constraint $w_1 + \cdots + w_k = (1 - \gamma\lceil \log k \rceil) w_\ell$. Therefore, for analyzing the worst-case, we would like to know the way to determine the number of child nodes and the way to distribute the weight w_ℓ to its child nodes that minimize the decrease of the total tree weight. This observation motivates us to define the following combinatorial game between a player and an adversary, where a player corresponds to a strategy of a decision tree learning algorithm.

Weight Distribution Game

1. Both players are informed of a given initial weight w, $0 \leq w \leq 1$, and a given target size s. A game starts from a single-leaf tree T consisting of one leaf node of weight w.

2. While $|T| < s$, both players repeat the following procedures:

   (a) The player chooses a leaf $\ell \in L(T)$.

   (b) The adversary chooses any integer k (≥ 2) satisfying either $k = 2$ or $2 < k \leq s/|T|$, and expand the leaf ℓ by replacing the leaf with an internal node with k child leaves. Then the adversary assigns weights $w_1, \ldots, w_k$ to these new child nodes so that the following equation holds:

$$w_1 + \cdots + w_k = (1 - \gamma\lceil \log k \rceil) w_\ell.$$

Player's (resp., Adversary's) goal: Minimize (resp., maximize) the weight of the final tree T with s leaves.

Our decision tree learning algorithm corresponds to the player who always chooses a leaf with the maximum weight. Then the worst-case situation for the algorithm corresponds to the case when the adversary chooses the best strategy to this player. As our main technical result, we analyze the final weight of the tree of this game as follows. (The proof is given in the next section.)

Theorem 3. In the Weight Distribution Game, consider the player who always chooses a leaf with the maximum weight. Then the tree weight is maximized if the adversary always chooses the number of child leaves k equals to 2 and distributes the weight of the expanded leaf *equally* to all its child nodes.

Now the worst-case situation for our boosting algorithm is clear from this result. That is, the value of index function $I(T)$ gets decreased *slowest* when (1) every chosen branching function is binary and (2) divides the leaf's entropy equally to its all child nodes. Thus, by analyzing this case, we would be able to derive an upper bound of the training error.

Theorem 4. Assume that $H_2 \subset H$ satisfies the γ-weak hypothesis assumption for f. Then, $\mathbf{TOPDOWN\text{-}M_{I,H}}(S, s)$ outputs T with

$$\hat{\epsilon}(T) \leq I(T) \leq s^{-\gamma}.$$

Proof. Assume that in the Weight Distribution Game, the player always chooses the leaf whose weight is the maximum. On the other hand, suppose that the adversary chooses the number of child nodes $k = 2$ and distributes the weight of each leaf equally among its new child leaves in the game. Then each node of the same depth has the same weight and any node with smaller depth has larger weight. Thus the player is forced to choose a leaf of the minimum depth.

As the output tree is a binary tree, the number of leaves of the tree becomes s after $s - 1$ expansion. Let $t = s - 1$.

Suppose that after t expansions, all nodes of depth $j \leq i$ are expanded, and there are t' leaves of depth $i+1$ expanded$(0 \leq t' \leq 2^i)$. Then the number of all expansions t is given by $t = \sum_{j=0}^{i} 2^j + t' = 2^{i+1} - 1 + t'$.

Note that just after all nodes of each depth are expanded, the weight of the tree is multiplied by $(1-\gamma)$. Thus after all expansions of leaves of depth i, the weight of the tree is $w(1-\gamma)^{i+1}$. Since there are 2^{i+1} nodes whose depth is $i+1$, the weight of each node of depth $i+1$ is $w(1-\gamma)^{i+1}/2^{i+1}$ after $2^{i+1}-1$ expansion. Now the weight of the tree after the t th expansion is

$$(2^{i+1}-t')\frac{w(1-\gamma)^{i+1}}{2^{i+1}} + 2t'\frac{w(1-\gamma)^{i+2}}{2^{i+2}} = w(1-\gamma)^{i+1}\left(1-\frac{t'\gamma}{2^{i+1}}\right).$$

Note that $1-x \leq e^{-x}$ for any $0 \leq x \leq 1$ and $w \leq 1$. Then we have

$$w(1-\gamma)^{i}\left(1-\frac{t'\gamma}{2^{i+1}}\right) \leq \exp[-\gamma(i+t'/2^{i+1})].$$

Using $\ln(1+x) \leq x$, we obtain $\ln(2^{i+1}+t') \leq i+1+t'/2^{i+1}$. Finally,

$$\begin{aligned}\exp[-\gamma(i+t'/2^{i+1})] &\leq \exp[-\gamma\ln(2^{i+1}+t')] \\ &= s^{-\gamma}.\end{aligned}$$

□

4.1 Weight Distribution Game

In this section we prove Theorem 3.

First we prepare some notations. Let $\mathcal{D}_k$ denote the set of all possible distributions over $\{1,\ldots,k\}$ $(k \geq 2)$. In particular, let d_k^* be the uniform one, i.e., $d_k^* = (1/k,\ldots,1/k)$. We also define $\mathcal{D} = \bigcup_{k\geq 2}\mathcal{D}_k$. Here, for any distribution $d_k \in \mathcal{D}$, the subscript k is the size of the domain of d_k. Note that a sequence of distributions defines a sequence of the adversary's choices in the Weight Distribution Game. Therefore we refer to a sequence of distributions as an *adversary's strategy*.

For any adversary's strategy $d_{k_1}^{(1)},\ldots,d_{k_t}^{(t)} \in \mathcal{D}^t$ $(t \geq 1)$, we denote

$$\mathrm{sum}_{\max}(w, d_{k_1}^{(1)},\ldots,d_{k_t}^{(t)})$$

as the weight of the tree obtained by the following way:

1. the initial weight is w;
2. at the i th expansion, the number of child nodes is k_i and the way to distribute weights is specified by $d_{k_i}^{(i)}$ $(1 \leq i \leq t)$; and

3. the player chooses the leaf whose weight is the maximum among all leaves.

For convenience, in this definition, we neglect some rules of the Weight Distribution Game, i.e., given size and the constraint of distributions.

Then we consider a sequence of distributions corresponding to a sequence of branching functions that are acceptable for s. We say that such a sequence is acceptable. More precisely, an adversary's strategy $d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)}$ with length t is *acceptable* for s if

1. $s = (k_1 - 1) + \cdots + (k_t - 1) + 1$, and
2. for any integer i $(1 \leq i \leq t)$,

$$2 \leq k_i \leq \frac{s}{(k_1 - 1) + \cdots + (k_{i-1} - 1) + 1}.$$

By using this notation, Theorem 3 can be re-written as follows.

Theorem 5. For any weight w, any integer $s \geq 2$ and any adversary's strategy $d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)} \in \mathcal{D}^t$ that is acceptable for s,

$$\mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)}) \leq \mathrm{sum}_{\max}(w, \underbrace{d_2^*, \ldots, d_2^*}_{s-1}).$$

The main idea of the proof is the following: Given a multi-way branching tree produced in the game, we choose a multi-way branching node (e.g., having k-branches) from the bottom and replace it with an 'almost equivalent' binary tree (having $k - 1$ internal nodes). Then we show that the tree obtained by the operation has larger weight. But this naive idea is not sufficient; The changed tree possibly cannot be constructed when the player always chooses the leaf with the maximum weight. In other words, to produce the modified tree, the player might be forced to choose some leaf whose weight is *not* the maximum at some expansion. From later on, we consider other strategies for the player to solve this technical problem.

Let us consider a player different from that in the Weight Distribution Game. We say that the player is a *u-max* player if it chooses the leaf with the maximum weight at each expansion *until the u th expansion*. Note that a u-max player may not choose the leaf with the maximum weight after $u+1$ expansions. Then we have the following proposition.

Proposition 6. For any weight $w \in [0,1]$, any integer t, u, , any u-max player p_u, and any adversary's strategy $d_{k_1}^{(1)}, \ldots d_{k_u}^{(u)}, d_2^*, \ldots, d_2^* \in \mathcal{D}^{u+t}$,

$$\mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\mathrm{u-max}}(w, d_{k_1}^{(1)}, \ldots d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}),$$

where $\mathrm{sum}_{\mathrm{u-max}}(\ldots)$ is the weight of the tree obtained by the player p_u.

Proof. Let p_{max} be the player that always choose a leaf with the maximum weight among leaves at each expansion. We denote $n_1, \ldots, n_J$ as the nodes that are leaves just after u expansions (Here $J = (k_1 - 1) + \cdots + (k_u - 1) + 1$). W.l.o.g., we assume that $w_{n_1} \geq w_{n_2} \geq \ldots w_{n_J}$. Note that p_{max} chooses n_1 at the $u+1$ th expansion.

We prove this proposition by induction on t. Suppose that p_u chooses the leaf n_j at the $u+1$ th expansion. First of all, for $t = 1$ and any u, it is easily verified that

$$\begin{aligned}
\mathrm{sum}_{\max}(w, d_{k_1}, \ldots d_{k_u}, d_2^*) &= \sum_{i=1}^{J} w_{n_i} - \gamma w_{n_1} \\
&\leq \sum_{i=1}^{J} w_{n_i} - \gamma w_{n_j} \\
&= \mathrm{sum}_{\mathrm{u-max}}(w, d_{k_1}, \ldots d_{k_u}, d_2^*).
\end{aligned}$$

Next, assume that for $t = t'$ ($t' \geq 2$) and for any u the proposition holds. Let $t = t' + 1$. Suppose that p_u chooses n_1 during the remaining t expansions. Note that each output tree can be made by sequences of distributions having different orders. Thus, w.l.o.g., we assume that p_u chooses n_1 at the $u+1$ th expansion. Since p_u behaves as in the same way as p_{max} until $u+1$ expansions, the proposition is true by the assumption of the induction.

Otherwise, assume that p_u never chooses n_1 during the rest of t expansions. Note that n_1 is still leaf at the $u+t$ th expansion. Let n' be the leaf chosen by p_u at the $u+t$ th expansion. Then it holds that $w_{n_1} \geq w_{n'}$ since n' is a descendant of some node n_i ($2 \leq i \leq J$). Therefore, if p_u chooses n_1 instead of n' at the $u+t$ th expansion, the weight of the output tree becomes smaller. We denote the sum obtained by choosing n' as $\mathrm{sum}_{(\mathrm{u+1})-\max}(\ldots)$.

Now we have, by the assumption of the induction,

$$\begin{aligned}
&\mathrm{sum}_{\mathrm{u-max}}(w, d_{k_1}, \ldots d_{k_u}, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \\
&\geq \mathrm{sum}_{(\mathrm{u}+1)-\mathrm{max}}(w, d_{k_1}, \ldots d_{k_u}, d_2^*, \underbrace{d_2^*, \ldots, d_2^*}_{t'}) \\
&\geq \mathrm{sum}_{\mathrm{max}}(w, d_{k_1}, \ldots d_{k_u}, \underbrace{d_2^*, \ldots, d_2^*}_{t}).
\end{aligned}$$

□

Then we prove the following two propositions.

Proposition 7. For any weight $w \in [0, 1]$, any integer t, k ($k \geq 2$) and any distribution $d_k \in \mathcal{D}_k$,

$$\mathrm{sum}_{\mathrm{max}}(w, d_k, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\mathrm{max}}(w, d_k^*, \underbrace{d_2^*, \ldots, d_2^*}_{t}).$$

Proof. First of all we consider a Weight Distribution Game in which (1) the player chooses a leaf ℓ of the minimum depth among all leaves of the current tree at each expansion, and (2) ℓ has the maximum weight among all leaves of the same depth. Note that in this game, the player behaves like the "breadth-fast search" regardless of the adversary's choices. We denote the weight of the tree in the game as $\mathrm{sum}_{\mathrm{bf}}(\ldots)$. (Here "bf" stands for "breadth-fast".) Applying Proposition 6 for $u = 1$, we have

$$\mathrm{sum}_{\mathrm{max}}(w, d_k, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\mathrm{bf}}(w, d_k, \underbrace{d_2^*, \ldots, d_2^*}_{t}). \tag{1}$$

Next we prove the following inequality:

$$\mathrm{sum}_{\mathrm{bf}}(w, d_k, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\mathrm{max}}(w, d_k^*, \underbrace{d_2^*, \ldots, d_2^*}_{t}). \tag{2}$$

Let T and T^* be the trees corresponding to left and right side of the above inequality respectively. Similarly, let p_{bf} and p_{max} be the corresponding players respectively. Since the adversary for p_{max} always assign weights uniformly, w.l.o.g., we assume that p_{max} behaves as in the same way as p_{bf}. Thus T and T' have the same shape.

If T and T^* are completely balanced, i.e., in both trees each leaf has the same depth i for some $i \geq 1$. Then the weights of both T and T^* are $w(1-\gamma\lceil\log k\rceil)(1-\gamma)^{(i-1)}$. Thus the inequality (2) holds.

Otherwise, T and T^* are not completely balanced. Suppose that T and T^* are trees of depth $i+1$ ($i \geq 1$) in which all nodes of depth $i-1$ and t' nodes of depth i are expanded ($1 \leq t' < k2^i$). Here the number of leaves in T (resp. T^*) is $k+t = k2^{i-1}+t'$. We denote the number of nodes of depth i in both trees as J ($J = k2^{i-1}$). Let $w_1, \ldots, w_J$ be the weights of node of depth i in T. W.l.o.g., we assume that $w_1 \geq \cdots \geq w_J$. Then it holds that $\sum_{j=1}^{J} w_j = w(1-\gamma\lceil\log k\rceil)(1-\gamma)^{i-1}$. On the other hand, the weight of each node of depth $i+1$ in T^* are the same, which we denote as $\widehat{w}$. Note that $\sum_{j=1}^{J} w_j = \sum_{j=1} \widehat{w} = w(1-\gamma\lceil\log k\rceil)(1-\gamma)^{i-1}$ Thus we have $\widehat{w} = w(1-\gamma\lceil\log k\rceil)(1-\gamma)^{i-1}/J$. Now we claim that for any t' ($1 \leq t' \leq J$),

$$\sum_{j=1}^{t'} w_j \geq t'\widehat{w}. \tag{3}$$

Suppose not, namely, $\sum_{j=1}^{t'} w_j < t'\widehat{w}$ for some t'. Then we have $w_{t'} < \widehat{w}$. Because the sequence $\{w_j\}$ is monotone decreasing, it holds that $\sum_{j=1}^{J} w_j < J\widehat{w}$ But this contradicts the fact that $\sum_{j=1}^{J} w_j = J\widehat{w} = w(1-\gamma\lceil\log k\rceil)(1-\gamma)^{i-1}$. This completes the proof of the claim (3). Then, we have

$$\begin{aligned}
\mathrm{sum}_{\mathrm{bf}}(w, d_k, \underbrace{d_2^*, \ldots, d_2^*}_{t}) =& w(1-\gamma\lceil\log k\rceil)(1-\gamma)^{i-1} - \gamma\sum_{j=1}^{t'} w_j \\
\leq& w(1-\gamma\lceil\log k\rceil)(1-\gamma)^{i-1} - \gamma t'\widehat{w} \\
=& \mathrm{sum}_{\max}(w, d_k^*, \underbrace{d_2^*, \ldots, d_2^*}_{t}).
\end{aligned}$$

Finally, conbining inequalities (1) and (2), we complete the proof.

□

Proposition 8. For any weight $w \in [0,1]$, any integer t, $k(k \geq 2)$ and any distribution $d_k \in \mathcal{D}_k$, it holds that

$$\mathrm{sum}_{\max}(w, d_k^*, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\max}(w, \underbrace{d_2^*, \ldots, d_2^*}_{t+k-1}).$$

Proof. Let $s = k + t$, i.e., the number of leaves in the trees corresponding to the both side of the inequalities above. Suppose s is given as follows: $s = 2^i + s'$ ($2^i \leq s < 2^{i+1}$, $s' \geq 0$). Then, we have

$$\begin{aligned}\mathrm{sum}_{\max}(w, \underbrace{d_2^*, \ldots, d_2^*}_{t+k-1}) =& (1-\gamma)^i w - s'\gamma \frac{(1-\gamma)^i}{2^i} w \\ =& (1-\gamma)^i \left(1 - \frac{s'\gamma}{2^i}\right) w \\ \stackrel{\text{def}}{=}& \phi_\gamma(s) w.\end{aligned}$$

On the other hand, suppose s is given as follows: $s = k2^{i'} + s''$ ($k2^{i'} \leq s < k2^{i'+1}$, $s'' \geq 0$). Then, we have

$$\begin{aligned}\mathrm{sum}_{\max}(w, d_k^*, \underbrace{d_2^*, \ldots, d_2^*}_{t}) =& (1-\gamma\lceil\log k\rceil)(1-\gamma)^{i'} w \\ & - s''\gamma \frac{(1-\gamma\lceil\log k\rceil)(1-\gamma)^{i'}}{k2^{i'}} w \\ =& (1-\gamma\lceil\log k\rceil)(1-\gamma)^{i'} \left(1 - \frac{s''\gamma}{k2^{i'}}\right) w \\ =& (1-\gamma\lceil\log k\rceil)\phi_\gamma(s/k) w\end{aligned}$$

We note that $\phi_\gamma(2s) = (1-\gamma)\phi_\gamma(s)$ and ϕ_γ is monotone non-increasing function. That implies,

$$\frac{\phi_\gamma(s)}{(1-\gamma)^{\lfloor\log k\rfloor}} \leq \phi_\gamma\left(\frac{s}{k}\right) \leq \frac{\phi_\gamma(s)}{(1-\gamma)^{\lceil\log k\rceil}}.$$

Now the following inequality holds:

$$\begin{aligned}\mathrm{sum}_{\max}(w, d_k^*, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq& (1-\gamma\lceil\log k\rceil)\frac{\phi_\gamma(s)}{(1-\gamma)^{\lceil\log k\rceil}} \\ \leq& \phi_\gamma(s) \\ =& \mathrm{sum}_{\max}(w, \underbrace{d_2^*, \ldots, d_2^*}_{t+k-1}).\end{aligned}$$

This completes the proof. □

Combinining Proposition 7 and 8, we obtain the following lemma.

Lemma 9. For any weight $w \in [0,1]$, any integer $t \geq 1$, and any distribution $d_k \in \mathcal{D}_k$,

$$\mathrm{sum}_{\max}(w, d_k, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\max}(w, \underbrace{d_2^*, \ldots, d_2^*}_{t+k-1}).$$

Intuitively, this lemma states that if we replace the root node having k branches with the binary decision tree having k branches in the bottom (without leaves), we can construct the tree that has larger weight than the original one.

Then we generalize Lemma 9 as follows:

Lemma 10. For any weight $w \in [0,1]$, any integer $s \geq 2$, and any adversary's stragegy $d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, d_2^*, \ldots, d_2^* \in \mathcal{D}^{u+t}$, that is acceptable for s,

$$\mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)}, \underbrace{d_2^*, \ldots, d_2^*}_{t+k_u-1})$$

Proof. We prove this inequality starting from the right side. Note that if the adversary's strategy $d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, d_2^*, \ldots, d_2^*$ of length $u+t$ is acceptable for s, then the adversary's strategy $d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)}, d_2^*, \ldots, d_2^*$ with length $u+t+k_u-2$ is also acceptable for s.

Let T be the tree corresponding to the sum

$$\mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)}, \underbrace{d_2^*, \ldots, d_2^*}_{t+k_u-1}),$$

and let L_{u-1} be the set of nodes in T that are leaves just after the $u-1$ th expansion. Then, we have

$$\mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)}, \underbrace{d_2^*, \ldots, d_2^*}_{t+k_u-1}) = \sum_{\ell \in L_{u-1}} \mathrm{sum}_{\max}(w_\ell, \underbrace{d_2^*, \ldots, d_2^*}_{t_\ell}), \quad (4)$$

where t_ℓ is the number of expansions for leaf ℓ and its descendant leaves. Note that $t+k_u-1 = \sum_{\ell \in L_{u-1}} t_\ell$. Let ℓ^* be the leaf in L_{u-1} that has the maximum weight. (So ℓ^* is the u th leaf to be expanded.) Let T_{ℓ^*} be the subtree of T rooted at ℓ^*. Here $|T_{\ell^*}| = t_{\ell^*} + 1$. Let $\widehat{T}_{\ell^*}$ be a tree that has the following properties: (1) $|T_{\ell^*}| = |\widehat{T}_{\ell^*}| = t_{\ell^*} + 1$, and (2) $\widehat{T}_{\ell^*}$ is generated according to the adversary's strategy $d_{k_u}, \underbrace{d_2^*, \ldots, d_2^*}_{t_{\ell^*}-k_u+1}$, whereas T_{ℓ^*} is generated according

to the adversary's strategy $\underbrace{d_2^*, \ldots, d_2^*}_{t_{\ell^*}}$. Now we consider replacing T_{ℓ^*} with $\widehat{T}_{\ell^*}$. Before doing this, we need to guarantee that $k_u \leq |T_{\ell^*}|$, otherwise $t_{\ell^*} - k_u + 1 < 0$. This is clear when $k_u = 2$. Then we consider the other case. Because the adversary's strategy $d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, d_2^*, \ldots, d_2^*$ is acceptable for s, by definition, we have $k_u \leq s/|L_{u-1}|$. On the other hand, T_{ℓ^*} is the largest subtree among all subtrees rooted at leaves in L_{u-1}. This implies $s/|L_{u-1}| \leq |T_{\ell^*}|$. Now we guarantee that $k_u \leq |T_{\ell^*}|$. From Lemma 9, we have

$$\mathrm{sum}_{\max}(w_{\ell^*}, \underbrace{d_2^*, \ldots, d_2^*}_{t_\ell}) \geq \mathrm{sum}_{\max}(w_{\ell^*}, d_{k_u}^{(u)} \underbrace{d_2^*, \ldots, d_2^*}_{t_\ell - k_u + 1}).$$

Thus we conclude that by replacing subtree T_{ℓ^*} with $\widehat{T}_{\ell^*}$, the weight of T becomes small. We denote the replaced tree as $\widehat{T}$, and denote its weight as $\widehat{\mathrm{sum}}(\ldots)$. Then, we have

$$\sum_{\ell \in L_{u-1}} \mathrm{sum}_{\max}(w_\ell, \underbrace{d_2^*, \ldots, d_2^*}_{t_\ell}) \geq \widehat{\mathrm{sum}}(w, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}). \quad (5)$$

Note that $\widehat{T}$ can be produced by a player who chooses leaves of the maximum weight at the initial u expansions. By Proposition 6, we have

$$\widehat{\mathrm{sum}}(w, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \geq \mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}). \quad (6)$$

Combining inequalities (4), (5), and (6), we complete the proof. □

Now the proof of our theorem is easy.

Proof for Theorem 5 From Lemma 10, for any adversary's strategy $d_{k_1}, \ldots, d_{k_t} \in \mathcal{D}^t$ that is acceptable for s, we have

$$\begin{aligned}
\mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)}) \leq & \mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_t-1}^{(t-1)}, \underbrace{d_2^*, \ldots, d_2^*}_{k_t - 1}) \\
\leq & \mathrm{sum}_{\max}(w, d_{k_1}^{(1)}, \ldots, d_{k_{t-2}}^{(t-2)}, \underbrace{d_2^*, \ldots, d_2^*}_{k_{t-1} + k_t - 2}) \\
\leq & \ldots \\
\leq & \mathrm{sum}_{\max}(w, \underbrace{d_2^*, \ldots, d_2^*}_{s-1}).
\end{aligned}$$

□

5 Acknowledgements

I am grateful to Prof. Osamu Watanabe for his constant support during this research, and to Tadashi Yamazaki for important hints and comments. I also thank members of Watanabe research group for helpful discussions. Finally I thank anonymous referees for many suggestions to improve my draft.

References

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, April 1987.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[3] Carlos Domingo and Osamu Watanabe. MadaBoost: A modification of AdaBoost. In *Proceedings of 13th Annual Conference on Computational Learning Theory*, pages 180–189. Morgan Kaufmann, San Francisco, 2000.

[4] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, September 1995.

[5] Yoav Freund and Robert E. Schapire:. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[6] Kohei Hatano. Analyses of multi-way branching decision tree boosting algorithms. Technical Report C-152, Dept. of Math. and Computing Sciences, Tokyo Institute of Technology, 2001.

[7] M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999.

[8] Yishay Mansour and David McAllester. Boosting using branching programs. In *Proc. 13th Annual Conference on Computational Learning Theory*, pages 220–224. Morgan Kaufmann, San Francisco, 2000.

[9] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[10] Yoshifumi Sakai. A Naive Boosting Algorithm Using DNF Formulas. *Trans. IEICE-D*, J84-D-I(1), 2000. in Japanese.

[11] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[12] Robert E. Schapire. Using output codes to boost multiclass learning problems. In *Proc. 14th International Conference on Machine Learning*, pages 313–321. Morgan Kaufmann, 1997.

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

[14] Eiji Takimoto and Akira Maruoka. On the boosting algorithm for multiclass functions based on information-theoretic criterion for approximation. In *Proceedings of the 1st International Conference on Discovery Science*, volume 1532 of *Lecture Notes in Artificial Intelligence*, pages 256–267. Springer-Verlag, 1998.

[15] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.