Analyses of Multi-Way Branching Decision Tree Boosting Algorithms

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

May 29, 2001

Abstract

We improve the analyses for decision tree boosting algorithms. Our result consists of two parts. In the first part, we analyze a decision tree learning algorithm proposed by Takimoto and Maruoka [11] that produces a K-way branching decision tree (where $K \geq 2$ is a given constant) for multiclass classification. Under the assumption that the algorithm can always obtain a weak hypothesis with advantage $\geq \gamma$, Takimoto and Maruoka showed that $(1/\varepsilon)^{(K/\gamma)}$ boosting steps are sufficient for obtaining a tree with the desired training error ε. We improve this bound by proving that $(1/\log K)(1/\varepsilon)^{(\log K/\gamma)}$ steps are indeed sufficient. In the second part, we study a multi-way branching decision tree learning algorithm proposed by Mansour and McAllester [7] that produces a tree with nodes having different number of branches. By using a simpler analysis than the one given in [7], we simplify and improve their algorithm.

1 Introduction

Boosting is a technique to construct a "strong" hypothesis combining many "weak" hypotheses. This technique is first proposed by Schapire [8] originally to prove the equivalence between strong and weak learnability in PAC-learning. Many researchers have improved boosting techniques such as AdaBoost [5] and so on. (See for example, [4, 3, 9, 10].) Among them, Kearns and Mansour [6] showed that the learning process of well-known decision tree learning algorithms such as CART [2] and C4.5 [2] 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 hypotheses 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 node that corresponds to some hypothesis $h \in H_2$ (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 nodes of T, i.e., T's tree size, provided that for any distribution, there always exists some hypothesis 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 multiclass learning. Their algorithm uses, for any fixed $K \geq 2$, K-class hypotheses, i.e., hypotheses whose range is R such that $|R| = K$. On the other hand, Mansour and McAllester gave a generalized algorithm that constructs a decision tree (for binary classification) by using k-class hypotheses for any $k \leq K$. That is, their algorithm may construct a decision tree having nodes with different number of branches.

In this paper, we improve the analyses of these two algorithms. First, we examine the algorithm of Takimoto and Maruoka. For learning a K-way branching decision tree by using K-class hypotheses with advantage $\geq \gamma$, Takimoto and Maruoka proved that $(1/\varepsilon)^{(K/\gamma)}$ steps are sufficient (in other words, the obtained tree size is bounded by $(1/\varepsilon)^{(K/\gamma)}$) to achieve training error ε. We improve this bound by showing that $(1/\log K)(1/\varepsilon)^{(\log K/\gamma)}$ steps are indeed sufficient.

Secondly, we study the algorithm by Mansour and McAllester. Consider the situation of constructing a decision tree of size s for a given s by using k-class hypotheses, where $k \leq K$ for some constant $K \geq 2$. 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 hypothesis h satisfying the following:

(1) h is either binary (in which case $k = 2$) or k-class with some k, $2 < k \leq K$, 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 hypothesis that is better than a binary hypothesis 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 hypotheses with advantage γ. (Note that $g_\gamma(2) = 1$ by definition.)

We simplify their analysis, thereby obtaining the following much simpler and slightly improved condition, which also makes the algorithm simpler. (Here we consider the same situation and the goal as above.)

<u>Our condition</u>

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

(1) h is either binary (in which case $k = 2$) or k-class with some k, $2 < k \leq K$, such that $k \leq s/s'$, and

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

This condition is much simpler, and in fact, the above explained intuition becomes more clear under this condition. The item (2) of this new condition means that k-class hypothesis h is better than an "equivalent" depth $\lceil \log k \rceil$ decision tree consisting of binary hypotheses with advantage γ. That is, if we can always find such a hypothesis 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 hypotheses with advantage γ.

Note that 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.

Moreover, our modified algorithm does not need to know γ, which is unknown in the realistic setting; Mansour and McAllester's algorithm has to know γ.

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

2 Preliminaries

We introduce our learning model briefly. Our model is based on PAC learning model proposed by Valiant[12]. Let X denote an instance space. We assume the unknown target function $f : X \to Y = \{1, \ldots, N\}$ ($N \geq 2$). 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 any constant ε and δ ($0 < \varepsilon, \delta < 1$), to output an hypothesis $h_f : X \to Y$ such that its *generalization error*

$$\epsilon(h_f) \stackrel{\text{def}}{=} \Pr_P[f(x) \neq h_f(x)]$$

is below ε, with probability $\geq 1 - \delta$.

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:

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.
2. $size(h_f) = o(m)$, where $size(\cdot)$ is 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 when a hypothesis is represented as as a decision tree, the second condition of Occam learning criterion can be interpreted that the number of leaves of the tree is 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 which meets the criteria.

Here we recall some basic notations about decision trees. In advance, we assume a set H of hypothesis $h : X \to R_h$, where R_h is a finite set such that $2 \leq |R_h| \leq K$, and K is any fixed integer. We allow each $h \in H$ to have different ranges. We denote the set of decision trees determined by H as $\mathcal{T}(H)$. A decision tree consists of internal nodes and leaves. Let T be any decision tree in $\mathcal{T}(H)$. Each internal node is labeled a hypothesis $h \in H$ and it has $|R_h|$ child nodes corresponding to the values of h, here child nodes are either internal nodes or leaves. Each leaf is labeled with an element of Y, the range of target function f.

Now we explain how to interpret a decision tree T. Suppose an instance $x \in X$ is given to T. First x goes to the root node of T. Then x goes to the child node corresponding to the values of $h(x)$. Finally, x reaches to some leaf. T answer the label of the leaf. Given a sample S, the way to label leaves is specified as follows: Let S_ℓ be the subset of S that reach to a leaf ℓ. The label of ℓ is specified with the major label among labeled examples in S_ℓ.

When the number of branches of each node in T is some $k \geq 2$, we call T *k-way branching decision tree.* In general cases, including cases when the number of branches of each node in T is different, we call T *multi-way branching decision tree.*

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 satisfying $\Pr_D[f(x) \neq h_f(x)] \leq 1/2 - \gamma$. However, in this paper, we measure the goodness of hypotheses from an information-theoretic viewpoint. Here we use an extended entropy proposed by Takimoto and Maruoka [11].

Definition 1. A function $G : [0,1]^N \to [0,1]$ is *pseudo-entropy function* if, for any $q_1, \ldots, q_N \in [0,1]$ such that $\sum_{i=1}^{N} q_i = 1$,

1. $\min_{i=1,\ldots,N}(1 - q_i) \leq G(q_1, \ldots, q_N)$,
2. $G(q_1, \ldots, q_N) = 0 \iff q_i = 1$ for some i, and
3. G is concave.

For example, Shannon entropy function $H_N(q_1, \ldots, q_N) = \sum_{i=1}^{N} q_i \log(1/q_i)$ is a pseudo-entropy function. Then we define the entropy of function f using a pseudo-entropy function G.

Definition 2. *G-entropy* of f with respect to D, denoted $H_G^D(f)$, is defined as

$$H_D^G(f) \stackrel{\text{def}}{=} G(q_1, \ldots, q_N),$$

where $q_i = \Pr_D[f(x) = i]$ $(i \in Y)$.

We can interpret G-entropy as "impurity" of value of f under the distribution D. For example, if f takes only one value, G-entropy becomes the minimum. If the value of f is random, G-entropy becomes the maximum.

Then we define the conditional G-entropy given a hypothesis $h : X \to R_h$, where R_h is a finite set but possibly different from Y.

Definition 3. *Conditional G-entropy* of f given h with respect to D, denoted as $H_D^G(f|h)$, is defined as

$$H_D^G(f|h) \stackrel{\text{def}}{=} \sum_{j \in R_h} \Pr_D[h(x) = j] G(q_{1|j}, \ldots, q_{N|j}),$$

where $q_{i|j} = \Pr_D[f(x) = i | h(x) = j]$ $(i \in Y,\ j \in R_h)$.

Now we show the relationship between the error and G-entropy. Since the range of h may be different from that of f, we need to give a way to interpret h's values. More precisely, We define a mapping $M : R_h \to Y$:

$$M(j) \stackrel{\text{def}}{=} \arg_{i \in Y} \max q_{i|j}$$

Then, the following fact holds.

Proposition 1.

$$\Pr_D[f(x) \neq M(h(x))] \leq H_D^G(f|h)$$

Proof.

$$\begin{aligned}\Pr_D[f(x) \neq M(h(x))] &= \sum_{j \in R_h} \Pr_D[h(x) = j] \Pr_D[f(x) \neq M(h(x)) | h(x) = j] \\ &= \sum_{j \in R_h} \Pr_D[h(x) = j]\{1 - \max_{i \in Y} q_{i|j}\} \\ &= \sum_{j \in R_h} \Pr_D[h(x) = j] \min_{i \in Y} q_{i|j} \\ &\leq \sum_{j \in R_h} \Pr_D[h(x) = j] G(q_{1|j}, \ldots, q_{N|j}) = H_D^G(f|h)\end{aligned}$$

□

We note that if $H_D^G(f|h) = 0$, the error probability $\Pr_D[f(x) \neq M(h(x))]$ also becomes 0. Following relationship between "error-based" and "information-based" hypotheses was first proved by Kearns and Mansour [6].

Lemma 2 (Kearns & Mansour [6]). Suppose $N = 2$ and $G(q_1, q_2) = 2\sqrt{q_1 q_2}$. For any distribution D over X, if there exists a hypothesis $h : X \to Y = \{1, 2\}$ such that $\Pr_D[f(x) \neq h(x)] \leq 1/2 - \delta$, then there exists a hypothesis $h' : X \to Y$ such that

$$H_D^G(f) - H_D^G(f|h') \geq \frac{\delta^2}{16} H_D^G(f).$$

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

Definition 4. Let f be any function from X to $Y = \{1, \ldots, N\}$ ($N \geq 2$). Let $G : [0,1]^N \to [0,1]$ be any pseudo-entropy function. Let H be any set of hypotheses. H and G satisfy the γ*-weak hypothesis assumption* for f if for any distribution D over X, there exists a hypothesis $h \in H$ satisfying

$$H_D^G(f) - H_D^G(f|h) \geq \gamma H_D^G(f),$$

where $0 < \gamma \leq 1$. We call this constant γ *advantage* and we refer to the reduction $H_D^G(f) - H_D^G(f|h)$ as *gain*. .

3 Learning K-Way Branching Decision Trees

We study the decision tree learning algorithm proposed by Takimoto and Maruoka [11]. For the target function $f : X \to Y = \{1, \ldots, N\}$ ($N \geq 2$), this algorithm constructs K-way branching decision trees where each node has exact K branches and K is any fixed integer ($K \geq 2$). We assume that the algorithm is given some pseudo-entropy function G and a set H_K of K-class hypotheses $h : X \to R$ such that $|R| = K$. in advance. We call the algorithm $\mathbf{TOPDOWN_{G,H_K}}$.

The description of $\mathbf{TOPDOWN_{G,H_K}}$ is given in Figure 1. In what follows, we explain the idea and the outline of this algorithm. $\mathbf{TOPDOWN_{G,H_K}}$, given a sample S and an integer $t \geq 1$ as input, outputs a decision tree with t internal nodes, where internal nodes are labeled with functions in H_K; that is, the obtained tree is a K-way decision tree. The algorithm's goal is to obtain, for this decision tree, one that has small training error on S under the uniform

$\underline{\mathbf{TOPDOWN}_{\mathbf{G},\mathbf{H_K}}(S,t)}$

begin

  $T \leftarrow$ the single leaf tree;

  for t **times do**

    $\ell \leftarrow \arg\max_{\ell \in L(T)} w_\ell \cdot G(q_{1|\ell}, q_{2|\ell}, \ldots, q_{N|\ell})$;

    $S_\ell \leftarrow \{\langle x, f(x)\rangle \in S | T(x) = \ell\}$;

    $D_\ell \leftarrow$ the uniform distribution over S_ℓ;

    $h \leftarrow \arg\max_{h \in H_K}(H^G_{D_\ell}(f) - H^G_{D_\ell}(f|h))$;

    $T \leftarrow T_{\ell,h}$;

  end-for

  Output T ;

end.

Figure 1: Algorithm $\mathbf{TOPDOWN}_{\mathbf{G},\mathbf{H_K}}$

distribution D over S. For this, the algorithm tries to reduce the conditional G-entropy of f given a constructed decision tree.

Let T be a decision tree that the algorithm has constructed so far. (Initially, T is a tree with a single leaf.) Let $L(T)$ denote the set of leaves of T. Then T can be regarded as a mapping any instance $x \in X$ to a leaf in $L(T)$. For each $\ell \in L(T)$, we define $w_\ell \stackrel{\text{def}}{=} \Pr_D[T(x) = \ell]$ and $q_{i|\ell} \stackrel{\text{def}}{=} \Pr_D[f(x) = i | T(x) = \ell]$ (where $i \in Y$).

Then the training error of T, which we denote as $\hat{\epsilon}(T)$, is computed as follows:

$$\hat{\epsilon}(T) \ = \Pr_D[f(x) \neq M(T(x))],$$

where $M(\ell) \stackrel{\text{def}}{=} \arg_{i \in Y} \max q_{i|\ell}$ for any $\ell \in L(T)$. We denote the conditional G-entropy of f given T as

$$H^G_D(f|T) \stackrel{\text{def}}{=} \sum_{\ell \in L(T)} w_\ell G(q_{0|\ell}, \ldots, q_{N|\ell}).$$

From Proposition 1, we have $\hat{\epsilon}(T) \leq H^G_D(f|T)$. In order for reducing this entropy, the algorithm makes a local change to the tree T. At each local change, the algorithm chooses a leaf $\ell \in L(T)$ and $h \in H_K$ and replace ℓ with a new internal node labeled h (and its K new child leaves). The tree obtained in this way is denoted $T_{\ell,h}$. We note that the total number of leaves increases by $K-1$.

We explain the way to choose ℓ and h at each local change. The algorithm chooses ℓ that maximizes $w_\ell G(q_{1|\ell}, \ldots, q_{N|\ell})$; it calculates a sample S_ℓ that reaches ℓ. Finally the algorithm choose $h \in H_K$ that maximizes the gain $H^G_{D_\ell}(f) - H^G_{D_\ell}(f|h)$, where D_ℓ is the uniform distribution over S_ℓ.

Note that

$$H^G_D(f|T) - H^G_D(f|T_{\ell,h}) = w_\ell(H^G_{D_\ell}(f) - H^G_{D_\ell}(f|h)).$$

Thus, if the gain is positive, then we reduce the conditional G-entropy.

For the efficiency of this boosting algorithm, Takimoto and Maruoka showed the following result.

Theorem 3 (Takimoto and Maruoka[11]). Assume that H_K and G satisfy the γ-weak hypothesis assumption. If

$$t \geq \left(\frac{1}{\varepsilon}\right)^{\frac{K}{\gamma}},$$

then $\mathbf{TOPDOWN_{G,H_K}}(S,t)$ outputs T with $\hat{\epsilon}(T) \leq \varepsilon$.

3.1 Our Analysis

We improve the analysis of $\mathbf{TOPDOWN_{G,H_K}}$. First we define some notation. For any leaf ℓ, we define *weight* W_ℓ as

$$W_\ell \stackrel{\text{def}}{=} w_\ell G(q_{1|\ell}, \ldots, q_{N|\ell}).$$

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_K and G satisfy γ-weak hypothesis assumption, then for any leaf ℓ and weights of ℓ's child leaves $W_1, \ldots, W_K$, we have

$$W_1 + \cdots + W_K \leq (1-\gamma)W_\ell.$$

From Fact 2 (1), in order to bound the training error of a tree, it suffices for us to consider the weight of the tree. On the other hand, it follows from Fact 2 (2) that at each boosting step, the weight of a tree gets decreased by γW_ℓ, if the leaf ℓ is "expanded" by this boosting step. 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 γW_ℓ. That is, we consider the worst-case under this situation, and we analyze how the tree's weight gets decreased in the worst-case. Notice that only the freedom left here is the distribution of the weights $W_1, \ldots, W_K$ of child leaves of ℓ under the constraint $W_1 + \cdots + W_K = (1-\gamma)W_\ell$. Therefore, for analyzing the worst-case, we would like to know the way to distribute the weight W_ℓ to its child nodes in order to minimize the decrease of the total tree weight. This motivates to define the following combinatorial game.

Weight Distribution Game

- Initially, the player is given a single-leaf tree where weight of the leaf is W, $(0 \leq W \leq 1)$, and any integer t $(t \geq 1)$.
- For t times, the player repeat the following procedures:
  1. Choose a leaf ℓ that has the maximum weight. Let W_ℓ is the weight of the leaf ℓ.
  2. Expand the leaf ℓ; that is, replace the leaf with an internal node with K child leaves and assign weights $W_1, \ldots, W_K$ of child nodes so that the following equation holds:

$$W_1 + \cdots + W_K = (1-\gamma)W_\ell.$$

Player's goal: Maximize the sum of weights of the tree after t step.

For the best strategy for the player in this game, we have the following result. (The proof will be given in the next subsection.)

Theorem 4. In the Weight Distribution Game, the tree weight is maximized if the player distribute the weight of expanded leaf *equally* to all its child nodes.

Now the worst-case situation for our boosting algorithm is clear from this result and our discussion above. That is, the tree weight gets decreased *slowest* when every chosen hypothesis 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 5. Assume that H_K and G satisfy the γ-weak hypothesis assumption. If

$$t \geq \frac{1}{\log K}\left(\frac{1}{\varepsilon}\right)^{\frac{\log K}{\gamma}},$$

then $\hat{\epsilon}(T) \leq \varepsilon$.

Proof. In Weight Distribution Game, if the player distribute the weight of each leaf equally among its new child leaves, the player always choose a leaf that is not expanded yet in the *oldest generation*, i.e., a set of leaves nearest from the root node.

Suppose that after the t th expansion, all leaves in the i th generation are expanded (We assume the initial leaf is in the first generation) and there are t' leaves that are not expanded yet in the $i+1$ th generation ($0 \leq t' \leq K^i$). Note that the number of nodes in the $i+1$ th generation is K^i.

Then the number of all expanded leaves t is given by $t = \sum_{j=1}^{i} K^{j-1} + t' = (K^i - 1)/(K-1) + t'$.

One can observe that just after all leaves in each generation are expanded, the weight of the tree is multiplied by $(1-\gamma)$. Thus after all expansions in the i th generation, the weight of the tree is $W(1-\gamma)^i$. A weight of each leaf in the $i+1$ generation is $W(1-\gamma)^i/K^i$, thus the weight of the tree after the t th expansion is

$$W(1-\gamma)^i - t'\gamma\frac{W(1-\gamma)^i}{K^i} = W(1-\gamma)^i\left(1 - \frac{t'\gamma}{K^i}\right).$$

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

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

From Proposition 15(see Appendix), we have

$$i + \frac{t'}{K^i} \geq \frac{1}{\log K}\ln\{(\log K)(K^i + (K-1)t')/(K-1)\}.$$

Thus, we derive that

$$\begin{aligned}\exp[-\gamma(i + t'/K^i)] &\leq \exp\left[-\gamma\frac{\ln\{\frac{\log K}{K-1}(K^i + (K-1)t')\}}{\log K}\right]\\ &= \left(\frac{\log K}{K-1}|T|\right)^{\frac{-\gamma}{\log K}},\end{aligned}$$

where $|T|$ is the number of leaves of T, i.e., $|T| = t(K-1)+1 = K^i + (K-1)t'$. From the assumption that $t \geq (1/\log K)(1/\varepsilon)^{(\log K/\gamma)}$, we have

$$\begin{aligned}
\left(\frac{\log K}{K-1}|T|\right)^{\frac{-\gamma}{\log K}} &= \left(\frac{\log K}{K-1}\{t(K-1)+1\}\right)^{\frac{-\gamma}{\log K}} \\
&< \left\{\frac{\log K}{K-1}t(K-1)\right\}^{\frac{-\gamma}{\log K}} \\
&= (\frac{1}{\varepsilon})^{\frac{\log K}{\gamma}\cdot\frac{-\gamma}{\log K}} \\
&= \varepsilon
\end{aligned}$$

□

By checking the proof of Theorem 5, we can easily see that the following simple bound can be obtained for the case $K = 2$, which is identical to Kearns and Mansour's result [6].

Theorem 6. Suppose $K = 2$. Assume that H_2 and G satisfy the γ-weak hypothesis assumption. Then we have

$$\hat{\epsilon}(T) \leq |T|^{-\gamma},$$

where $|T|$ is the number of leafs of T.

3.2 Weight Distribution Game

We prove Theorem 4. First we prepare some notations. Let $\mathcal{D}_K$ denote the set of all possible distributions over $\{1, \ldots, K\}$. In particular, let d_K^* be the uniform one, i.e., $d_K^* = (1/K, \ldots, 1/K)$. For any sequences of distributions $d_K^{(1)}, \ldots, d_K^{(t)} \in \mathcal{D}_K$, we denote

$$\mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(t)})$$

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

1. the initial weight is W.
2. the way to distribute at the i th step is specified with $d_K^{(i)} \quad (1 \leq i \leq t)$.

By using this notation, Theorem 4 can be written as follows.

Theorem 7. For any weight $W \in [0,1]$, any $t \geq 1$ and any sequence of distributions $d_K^{(1)}, \ldots, d_K^{(t)} \in \mathcal{D}_K$,

$$\mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(t)}) \leq \mathrm{sum}_\gamma(W, \underbrace{d_K^*, \ldots, d_K^*}_{t})$$

To prove the theorem, we show the following relation holds for any integer u, $1 \leq u \leq t$, and any sequence of distributions $d_K^{(1)}, \ldots, d_K^{(u)} \in \mathcal{D}_K$,

$$\mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(u)}\, \underbrace{d_K^*, \ldots, d_K^*}_{t-u}) \leq \mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(u-1)}\, \underbrace{d_K^*, \ldots, d_K^*}_{t-u+1}).$$

We prove this relation by induction on u.

First we prove the base step of the induction.

Lemma 8. For any weight $W \in [0,1]$, any $t \geq 1$ and any distribution $d_K \in \mathcal{D}_K$, we have

$$\mathrm{sum}_\gamma(W, d_K, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}) \leq \mathrm{sum}_\gamma(W, \underbrace{d_K^*, \ldots, d_K^*}_{t}).$$

Proof. Suppose that the player does not choose a leaf whose weight is the maximum; instead the player choose a leaf that is not expanded in the oldest generation whose weight is the maximum among weights of all leaves in the same generation. We denote the sum under the situation above as $\mathrm{sum'}_\gamma$. Clearly, we have

$$sum_\gamma(W, d_K, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}) \leq sum'_\gamma(W, d_K, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}). \tag{1}$$

Now we prove the following inequality.

$$sum'_\gamma(W, d_K, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}) \leq sum_\gamma(W, \underbrace{d_K^*, \ldots, d_K^*}_{t}). \tag{2}$$

Let T' and T^* be the trees corresponding to the left and right side of the inequality above respectively.

First, consider the case when T' and T^* are completely balanced, i.e., the case just after all leaves of the trees in the i th generation are expanded for some i. Then the weights of both T' and T^* are $W(1-\gamma)^i$ thus inequality (2) holds.

Secondly, we consider the other case, that is, T' and T^* are not completely balanced. Suppose that T' and T^* are trees such that all leaves in the i th generation and t' leaves in $i+1$ th generation are expanded($1 \leq t' \leq K^i$). We denote the numbers of nodes in $i+1$ th generation of both trees as $J = K^i$. We also denote the weights of nodes in the $i+1$ th generation of T' as $W_1, \ldots, W_J$. Without loss of generality, we assume that $W_1 \geq \cdots \geq W_J$. Then it holds that $\sum_{j=1}^{J} W_j = W(1-\gamma)^i$. On the other hand, a weight of each node in the $i+1$ th generation of T^* are the same. We denote the weights as $\widehat{W} = W(1-\gamma)^i/K^i$. Now we claim that for any t', $1 \leq t' \leq J$,

$$\sum_{j=1}^{t'} W_j \geq t'\widehat{W}.$$

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 that contradicts that $\sum_{j=1}^{J} W_j = J\widehat{W} = W(1-\gamma)^i$. This complete the proof of the claim.

Then, we have

$$\begin{aligned}
\mathrm{sum'}_\gamma(W, d_K, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}) &= W(1-\gamma)^i - \gamma \sum_{j=1}^{t'} W_j \\
&\leq W(1-\gamma)^i - \gamma t'\widehat{W} \\
&= \mathrm{sum}_\gamma(W, \underbrace{d_K^*, \ldots, d_K^*}_{t}).
\end{aligned}$$

Now we have proved the inequality (2). Combining inequality (1) and (2), we complete the proof. $\square$

Next we prove the induction step.

Lemma 9. For any weight $W \in [0,1]$, any $t \geq 1$, any $u \geq 1$ and any sequence of distributions $d_K^{(1)}, \ldots, d_K^{(u)} \in \mathcal{D}_K$,

$$\mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(u)}, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}) \leq \mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(u-1)}, \underbrace{d_K^*, \ldots, d_K^*}_{t}).$$

Proof. We prove this inequality from the right side. Let T be the tree that corresponds to

$$\mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(u-1)}, \underbrace{d_K^*, \ldots, d_K^*}_{t}).$$

Let L_{u-1} be the set of leaves of T after $u-1$ th expansion. Then, we have

$$sum_\gamma(W, d_K^{(1)}, \ldots, d_K^{(u-1)}, \underbrace{d_K^*, \ldots, d_K^*}_{t}) = \sum_{j \in L_{u-1}} sum_\gamma(W_j, \underbrace{d_K^*, \ldots, d_K^*}_{t_j}),$$

where t_j is the number of expansions for a leaf $j \in L_{u-1}$ and j's descendants. Note that $t = \sum_{j \in L_{u-1}} t_j$. Let $j^* = \arg\max_{j \in L_{u-1}} W_j$ (j^* is the u th leaf to be expanded.) Let T_{j^*} be the subtree of T rooted at node j^*. From Lemma 8 we have,

$$sum_\gamma(W_{j^*}, \underbrace{d_K^*, \ldots, d_K^*}_{t_{j^*}}) \geq sum_\gamma(W_{j^*}, d_K^{(u)}, \underbrace{d_K^*, \ldots, d_K^*}_{t_{j^*}-1}).$$

Let $\widehat{T}_{j^*}$ be the tree corresponding to the right side of the inequality above. We denote $\widehat{T}$ as the tree produced by removing T_{j^*} from T and adding $\widehat{T}_{j^*}$ at the node j^* in T. $\widehat{T}_{j^*}$ and denote $\widehat{sum}_\gamma(\ldots)$ be the sum for $\widehat{T}$. Then, we have

$$\sum_{j \in L_{u-1}} sum_\gamma(W_j, \underbrace{d_K^*, \ldots, d_K^*}_{t_j}) \geq \widehat{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(u)}, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}).$$

The tree $\widehat{T}$ may not be produced according to the rule that the leaf that has the maximum weight is to be expanded first. Thus, the weights of $\widehat{T}$ may be larger than the tree produced according to the rule. Now we have

$$\widehat{sum}_\gamma(W, \underbrace{d_K^{(1)}, \ldots, d_K^{(u)}}_{u}, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}) \geq sum_\gamma(W, \underbrace{d_K^{(1)}, \ldots, d_K^{(u)}}_{u}, \underbrace{d_K^*, \ldots, d_K^*}_{t-1}).$$

This complete the proof. □

Now the proof of our theorem is easy.

Proof of theorem 4. From Lemma 9, for any sequence of distributions $d_K^{(1)}, \ldots, d_K^{(t)}$,

$$\begin{aligned}
\mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(t)}) &\leq \mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(t-1)}, d_K^*) \\
&\leq \mathrm{sum}_\gamma(W, d_K^{(1)}, \ldots, d_K^{(t-2)}, d_K^*, d_K^*) \\
&\leq \ldots \\
&\leq \mathrm{sum}_\gamma(W, \underbrace{d_K^*, \ldots, d_K^*}_{t})
\end{aligned}$$

□

TOPDOWN-M$_{\mathbf{G,H}}$ (S, s)

begin

$T \leftarrow$ the single-leaf tree;

While $(|T| < s)$ **do**

$\ell \leftarrow \arg\max_{\ell \in L(T)} w_\ell G(q_{0|\ell}, q_{1|\ell})$;

$S_\ell \leftarrow \{\langle x, f(x)\rangle \in S | T(x) = \ell\}$;

$D_\ell \leftarrow$ the uniform distribution over S_ℓ ;

$h \leftarrow \arg\max_{h \in H, \text{acceptable}} \frac{H^G_{D_\ell}(f) - H^G_{D_\ell}(f|h)}{\lceil \log |R_h| \rceil}$

$T \leftarrow T_{\ell,h}$;

end-while

Output T ;

end.

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

4 Learning Multi-Way Branching Decision Trees

We propose an 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.

Throughout the rest of this paper, we only consider the case of learning boolean functions. The generalization to the multiclass case is easy. In previous sections, we deal with K-way branching decision trees, where K is some fixed integer and $K \geq 2$. But here we allow each node has a different number of branches.

Let H be a set of hypotheses where each $h \in H$ is a function from X to R_h $(2 \leq |R_h| \leq K)$. We assume that H contains a set of binary hypotheses H_2. The algorithm is given H and some pseudo-entropy function $G : [0,1]^2 \to [0,1]$ beforehand. We call the algorithm **TOPDOWN-M$_{\mathbf{G,H}}$**. **TOPDOWN-M$_{\mathbf{G,H}}$**, given a sample S and an integer $s \geq 2$ as input outputs a multi-way branching decision tree T with $|T| = s$. We say that a hypothesis h is *acceptable* for 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 hypotheses are acceptable. However H contains H_2 thus the algorithm can always select a hypothesis that is acceptable for any T and s. **TOPDOWN-M$_{\mathbf{G,H}}$** is a generalization of **TOPDOWN$_{\mathbf{G,H_2}}$**. The main modification is the criterion to choose hypotheses. **TOPDOWN-M$_{\mathbf{G,H}}$** chooses the hypothesis $h : X \to R_h$ such that maximizes the gain over $\lceil \log |R_h| \rceil$, not merely comparing the gain. This is because the algorithm must compare hypotheses that have different size of ranges. We show the detail of the algorithm is in Figure 2.

Note that if $H_2 \subset H$ satisfies the γ-weak hypothesis assumption and hypothesis $h : X \to R_h$ is selected for ℓ, then,

$$H^G_{D_\ell}(f) - H^G_{D_\ell}(f|h) \geq \gamma \lceil \log |R_h| \rceil H^G_{D_\ell}(f).$$

4.1 Our analysis

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

First let us recall some notation. For any leaf ℓ, we define *weight* W_ℓ as $W_\ell \stackrel{\text{def}}{=} w_\ell G(q_{0|\ell}, q_{1|\ell})$. The weight of a tree is just the total weight of all its leaves. Then by definition, we have the following relations.

Fact 2. 1. $\hat{\epsilon}(T) \leq \sum_{\ell \in L(T)} W_\ell$.

2. If H_2 and G satisfy γ-weak hypothesis assumption, then for any leaf ℓ and weights of ℓ's child leaves $W_1, \ldots, W_k$ $(2 \leq k \leq K)$. we have $W_1 + \cdots + W_k \leq (1 - \gamma\lceil\log k\rceil)W_\ell$.

From Fact 2 (1), it suffice for us to consider a weight of a tree in order to bound the training error of the tree. On the other hand, it follows from Fact 2 (2) that at each boosting step, the weight of a tree get decreased by $\gamma\lceil\log k\rceil W_\ell$, if a leaf ℓ is expanded by this boosting step and a k-class hypothesis is chosen for ℓ. Thus, we need to analyze how the weight of the tree gets reduced under "the worst-case", that is, 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-class hypothesis is chosen for ℓ.

Notice that only the freedom left here are (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, in order 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 so as to minimize the decrease of the total tree weight. To do this, we modify Weight Distribution Game as follows.

Weight Distribution Game-M

- Initially, the player is given a single-leaf tree T where weight of the leaf is W, $(0 \leq W \leq 1)$, and any integer s $(s \geq 2)$.
- While $|T| < s$, the player repeat the following procedures:
  1. Choose the leaf ℓ that has the maximum weight.
  2. Choose any integer k (≥ 2) satisfying either $k = 2$ or $2 < k \leq s/|T|$.
  3. Expand the leaf ℓ; replace the leaf with an internal node with k child leaves and assign weights $W_1, \ldots, W_k$ of child nodes so that the following equation holds: $W_1 + \cdots + W_k = (1 - \gamma\lceil\log k\rceil)W_\ell$.

Player's goal: Maximize the weight of the tree with the number of leaves is s.

For the best strategy for the player, we have the following result. (The proof will be given in the next subsection.)

Theorem 10. In the Weight Distribution Game-M, the tree weight is maximized if the player always chooses the number of child leaves $k = 2$ and distribute the weight of expanded leaf *equally* to all its child nodes.

Now the worst-case situation for our boosting algorithm is clear from this result and our discussion above. That is, the tree weight gets decreased *slowest* when every chosen hypothesis is binary and divides the leaf's entropy equally to its all child nodes. Thus, by analyzing this case, we would be able to derive an upperbound of the training error.

Theorem 11. Assume that $H_2 \subset H$ satisfies the γ-weak hypothesis assumption for f. Then, **TOPDOWN-M**$_{\mathbf{G,H}}(S, s)$ outputs T with $\hat{\epsilon}(T) \leq H_G^D(f|T) \leq s^{-\gamma}$.

Proof. In Weight Distribution Game-M, if the player choose the number of child nodes $k = 2$ and distribute the weight of each leaf equally among its new child leaves, the player always choose a leaf in the oldest generation that is not expanded yet. Note that after $s-1$ expansions, the number of the leaves of the tree becomes s. Let $t = s - 1$. Suppose that after the t th expansion, all leaves in the i th generation are expanded (We assume the initial leaf is in the first generation) and there are t' leaves not expanded in the $i+1$ th generation ($0 \leq t' \leq 2^i$). Then the number of all expansion t is given by $t = \sum_{j=1}^{i} 2^{j-1} + t' = 2^i - 1 + t'$.

One can observe that just after all leaves in each generation are expanded, the weight of the tree is multiplied by $(1-\gamma)$. Thus after all expansions in the i th generation, the weight of the tree is at most $W(1-\gamma)^i$. Because weight of each leaf in the $i+1$ generation is $W(1-\gamma)^i/2^i$, the weight of the tree after the t th expansion is $W(1-\gamma)^i(1-t'\gamma/2^i)$. Note that $1 - x \leq e^{-x}$ for any $0 \leq x \leq 1$ and $W \leq 1$. Then we have

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

From the fact that $i + t'/2^i \geq \ln(2^i + t')$, we have

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

□

4.2 Weight Distribution Game-M

We prove Theorem 10. First we prepare some notations. Recall that $\mathcal{D}_k$ is the set of all possible distributions over $\{1, \ldots, k\}$ ($2 \leq k \leq K$), and $d_k^* \in \mathcal{D}_k$ is the uniform one. We define $\mathcal{D} = \bigcup_{k \geq 2} \mathcal{D}_k$. Note that for any distribution $d_k \in \mathcal{D}$, the subscript k is the size of the domain of d_k. For any sequences of distributions $d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)} \in \mathcal{D}$ ($t \geq 1$), we denote $\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)})$ as the weight of the tree obtained by the following way:

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

Then we consider the sequence of distributions correspond to the sequence of hypotheses that are acceptable for s. We say that such sequences are acceptable. More precisely, a sequence of distributions $d_{k_1}^{(1)}, \ldots, d_{k_t}^{(}t)$ with length t is *acceptable* for s if $s = (k_1 - 1) + \cdots + (k_t - 1) + 1$ and for any integer i ($1 \leq i \leq t$), $k_i = 2$, or $2 < k_i \leq s/\{(k_1 - 1) + \cdots + (k_{i-1} - 1) + 1\}$. By using this notation, Theorem 10 can be written as follows.

Theorem 12. For any weight W, any integer $s \geq 2$ and any sequence of distributions $d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)}$ $\in \mathcal{D}$ that is acceptable for s ,

$$\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)}) \leq \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_2^*, \ldots, d_2^*}_{s-1}).$$

To prove the theorem, we show the following relation holds for any integer u ($1 \leq u \leq s-1$) and any sequence of distributions $d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)} \in \mathcal{D}$,

$$\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)} \underbrace{d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)} \underbrace{d_2^*, \ldots, d_2^*}_{t+k_{u-1}-1}),$$

where t is the integer such that the sequence $d_{k_1}^{(1)}, \ldots, d_{k_U}^{(u)}, d_2^*, \ldots, d_2^*$ is acceptable for s. We prove this relation by induction on u.

Lemma 13. For any weight $W \in [0,1]$, any integer $s \geq 2$, any distribution $d_k \in \mathcal{D}$, and the sequence of distributions $d_k, d_2^*, \ldots, d_2^*$, that is acceptable for s, with length t,

$$\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_k, d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_2^*, \ldots, d_2^*}_{s-1}).$$

Proof. Suppose that the player does not choose a leaf whose weight is the maximum but select a leaf in the oldest generation, that is not expanded yet, whose weight is the maximum among weights of all leaves in the same generation. We denote the sum under the situation above as $\mathrm{sum}'_{\gamma\lceil\log k\rceil}$. Then, we have

$$\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_k, d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}'_{\gamma\lceil\log k\rceil}(W, \underbrace{d_k, d_2^*, \ldots, d_2^*}_{t}).$$

Now we prove that the following inequality.

$$\mathrm{sum'}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_k, d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_k^*, d_2^*, \ldots, d_2^*}_{t}). \tag{3}$$

Let T' and T^* be the trees corresponding to left and right side of the above inequality respectively.

First, consider the case when T' and T^* are completely balanced, i.e., the case just after all leaves of the trees in the $i+1$ th generation are expanded for some i. Then the weights of both T' and T^* are $W(1-\gamma\lceil\log k\rceil)(1-\gamma)^i$ thus inequality (3) holds.

Secondly, we consider the other case, that is, T' and T^* are not completely balanced. Suppose that T' and T^* are trees such that all leaves in the $i+1$ th generation and t' leaves in the $i+2$ th generation are expanded ($1 \leq t' \leq k2^i$). We denote the numbers of children in the $i+2$ th generation of both trees as $J = k2^i$. We also denote the weights of nodes in the $i+2$ th generation of T' as $W_1, \ldots, W_J$. Without loss of generality, we assume that $W_1 \geq \cdots \geq W_J$. Then it holds that $\sum_{j=1}^{J} W_j = W(1-\gamma\lceil\log k\rceil)(1-\gamma)^i$. On the other hand, a weight of each node in the $i+2$ th generation of T^* are the same. We denote the weights as $\widehat{W} = W(1-\gamma)^i/k2^i$. Now we claim that for any t', $1 \leq t' \leq J$,

$$\sum_{j=1}^{t'} W_j \geq t'\widehat{W}.$$

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 that contradicts that $\sum_{j=1}^{J} W_j = J\widehat{W} = W(1-\gamma\lceil\log k\rceil)(1-\gamma)^i$. This complete the proof of the claim. Then, we have

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

Now we have proved the inequality (3). Next we prove the following inequality.

$$\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_k^*, d_2^*, \ldots, d_2^*}_{t}) \leq \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_2^*, \ldots, d_2^*}_{s-1}).$$

Suppose s is given as follows:$s = 2^i + s'$ ($2^i \le s \le 2^{i+1}$, $s' \ge 0$). Then, we have

$$\begin{aligned}
\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_2^*, \ldots, d_2^*}_{s-1}) =& (1-\gamma)^i W - s'\gamma\frac{(1-\gamma)^i}{2^i}W \\
=& (1-\gamma)^i \left(1 - \frac{s'\gamma}{2^i}\right) W \\
\stackrel{\text{def}}{=}& \phi_\gamma(s)W.
\end{aligned}$$

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

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

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

$$\frac{\phi_\gamma(s)}{(1-\gamma)^{\lfloor\log k\rfloor}} = \phi_\gamma\left(\frac{s}{2^{\lfloor\log k\rfloor}}\right) \le \phi_\gamma\left(\frac{s}{k}\right) \le \phi_\gamma\left(\frac{s}{2^{\lceil\log k\rceil}}\right) = \frac{\phi_\gamma(s)}{(1-\gamma)^{\lceil\log k\rceil}}.$$

Now the following inequality holds:

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

□

Next we prove the induction step.

Lemma 14. For any weight $W \in [0,1]$, any integer $s \ge 2$, any sequence of distributions $d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, d_2^*, \ldots, d_2^*$,that is acceptable for s, with length $u+t$, and the sequence of distributions $d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)}, d_2^*, \ldots, d_2^*$ with length $t + u + k_u - 2$,

$$\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \le \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)}, \underbrace{d_2^*, \ldots, d_2^*}_{t+k_u-1})$$

Proof. We prove this inequality from the right side. Note that if the sequence $d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}$, d_2^*, $\ldots$, d_2^* with length $u+t$ is acceptable for s, then, the sequence $d_{k_1}, \ldots, d_{k_{u-1}}^{(u-1)}, d_2^*, \ldots, d_2^*$ with length $u + t + k_u - 2$ is also acceptable for s.

Let T be the tree corresponds to the sum $\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)}, \underbrace{d_2^*, \ldots, d_2^*}_{t+k_u-1})$.

Let L_{u-1} be the set of leaves of T after the $u-1$ th expansion. Then, we have

$$\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_{u-1}}^{(u-1)}, \underbrace{d_2^*, \ldots, d_2^*}_{t+k_u-1}) = \sum_{\ell\in L_{u-1}} \mathrm{sum}_{\gamma\lceil\log k\rceil}(W_\ell, \underbrace{d_2^*, \ldots, d_2^*}_{t_\ell}),$$

where t_ℓ is the number of expansions for leaf ℓ and its descendants leaves. Note that $t = \sum_{\ell\in L_{u-1}} t_\ell$. 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^*. Let $\widehat{T}_{\ell^*}$ be a tree that has the following properties: (i) $|T_{\ell^*}| = |\widehat{T}_{\ell^*}|$, and (ii) $\widehat{T}_{\ell^*}$ is generated according to $d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t_{\ell^*}-k_u+1}$, whereas T_{ℓ^*} is generated according to $\underbrace{d_2^*, \ldots, d_2^*}_{t_{\ell^*}}$. Now we consider replacing T_{ℓ^*} with $\widehat{T}_{\ell^*}$. To do this, we need to guarantee that $k_u \le |T_{\ell^*}|$. That is clear when $k_u = 2$. Now we consider the other case. Because the sequence $d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, d_2^*, \ldots, d_2^*$ is acceptable for s , thus by definition, $k_u \le s/|L_{u-1}|$. On the other hand, T_{ℓ^*} is the biggest subtree among all subtrees rooted at leaves in L_{u-1}. That implies $s/|L_{u-1}| \le |T_{\ell^*}|$. Now we guarantee that $k_u \le |T_{\ell^*}|$. From Lemma 13, we have

$$\mathrm{sum}_{\gamma\lceil\log k\rceil}(W_{\ell^*}, \underbrace{d_2^*, \ldots, d_2^*}_{t_\ell}) \ge \mathrm{sum}_{\gamma\lceil\log k\rceil}(W_{\ell^*}, d_{k_u}^{(u)}\ \underbrace{d_2^*, \ldots, d_2^*}_{t_\ell-k_u+1}).$$

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

$$\sum_{\ell\in L_{u-1}} \mathrm{sum}_{\gamma\lceil\log k\rceil}(W_\ell, \underbrace{d_2^*, \ldots, d_2^*}_{t_\ell}) \ge \widehat{\mathrm{sum}}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}).$$

The tree $\widehat{T}$ may not be produced according to the rule that the leaf that has the maximum weight is to be expanded first. Thus, the sum of weights of $\widehat{T}$ may be larger than the tree produced according to the rule. Now we have

$$\widehat{\mathrm{sum}}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}) \ge \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_u}^{(u)}, \underbrace{d_2^*, \ldots, d_2^*}_{t}).$$

This complete the proof. □

Now the proof of our theorem is easy.

Proof for Theorem 12. From lemma 13 and lemma 14, for any sequence of distribution $d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)}$ that is acceptable for s, we have

$$\begin{aligned}
\mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_t}^{(t)}) &\le \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_{t-1}}^{(t-1)}, \underbrace{d_2^*, \ldots, d_2^*}_{k_{t-1}}) \\
&\le \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, d_{k_1}^{(1)}, \ldots, d_{k_{t-2}}^{(t-2)}, \underbrace{d_2^*, \ldots, d_2^*}_{k_{t-1}+k_t-2}) \\
&\le \ldots \\
&\le \mathrm{sum}_{\gamma\lceil\log k\rceil}(W, \underbrace{d_2^*, \ldots, d_2^*}_{s-1}).
\end{aligned}$$

□

5 Acknowledgement

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.

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] 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.

[7] Y. Mansour and D. McAllester. Boosting with multi-way branching in decision trees. In *Advances in Neural Information Processing Systems*, volume 12, pages 300–306. The MIT Press, 2000.

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

[9] 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.

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

[11] 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.

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

6 Appendix

Proposition 15. *For $0 \le t' < K^i$ ($i \ge 1$, $K \ge 2$),*

$$i + \frac{t'}{K^i} \ge \frac{1}{\log K} \ln \left\{K^i + (K-1)t'\right\} + \frac{1}{\log K} \ln \left(\frac{K-1}{\log K}\right).$$

Proof. Let

$$f(t') = i + \frac{t'}{K^i} - \frac{1}{\log K} \ln \left\{K^i + (K-1)t'\right\} - \frac{1}{\log K} \ln \left(\frac{K-1}{\log K}\right).$$

$$\begin{aligned} f'(t') &= \frac{1}{K^i} - \frac{1}{\log K} \frac{K-1}{K^i + (K-1)t'} \\ &= \frac{(\log K)\{K^i + (K-1)t'\} - (K-1)K^i}{K^i(\log K)\{K^i + (K-1)t'\}}. \end{aligned}$$

$$\begin{aligned} f'(t') = 0 &\iff (\log K)\{K^i + (K-1)t'\} - (K-1)K^i = 0 \\ &\iff (\log K)(K-1)t' = (K - \log K - 1)K^i \\ &\iff t' = \frac{(K - \log K - 1)K^i}{(\log K)(K-1)} \stackrel{\text{def}}{=} t^*. \end{aligned}$$

$$f''(t') = \frac{(K-1)^2}{\log K} \cdot \frac{1}{\{K^i + (K-1)t'\}^2} \ge 0.$$

Thus f is convex and $f(t')$ is the minimum when $t' = t^*$. Finally, we have

$$\begin{aligned} &f(t') \ge f(t^*) \\ &\quad = i + \frac{K - \log K - 1}{(\log K)(K-1)} \\ &\quad\quad - \frac{1}{\log K} \ln \left\{K^i + \frac{(K - \log K - 1)K^i}{\log K}\right\} + \frac{1}{\log K} \ln \left(\frac{K-1}{\log K}\right) \\ &\quad = \frac{K - \log K - 1}{(\log K)(K-1)} - \frac{1}{\log K} \ln \left\{1 + \frac{(K - \log K - 1)}{\log K}\right\} \\ &\quad\quad + \frac{1}{\log K} \ln \left(\frac{K-1}{\log K}\right) \\ &\quad = \frac{K - \log K - 1}{(\log K)(K-1)} - \frac{1}{\log K} \ln \left(\frac{K-1}{\log K}\right) + \frac{1}{\log K} \ln \left(\frac{K-1}{\log K}\right) \ge 0. \end{aligned}$$

□