<table>
<thead>
<tr>
<th>Title</th>
<th>An Efficient Smooth Boosting by Filtering (New Trends in Theory of Computation and Algorithm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Hatano, Kohei</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (2006), 1489: 57-63</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2006-05</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/58229">http://hdl.handle.net/2433/58229</a></td>
</tr>
<tr>
<td>Rights</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>

Kyoto University
より効率的なフィルタリング型ブースティング技法
An Efficient Smooth Boosting by Filtering

Kohei Hatano

Abstract

Boosting is a general method to construct a highly accurate classifier by combining "weakly" accurate ones. Smooth boosting algorithms are variants of boosting methods which handle only smooth distributions on the data. They are proved to be noise-tolerant and can be used in the "boosting by filtering" scheme, which is suitable for learning over huge data. However, current smooth boosting algorithms have rooms for improvements: A non-smooth boosting algorithm, InfoBoost can perform more efficiently than typical boosting algorithms by using an information-theoretic criterion for choosing hypotheses. In this paper, we propose a new smooth boosting algorithm with an information-theoretic criterion and we show that it inherits the advantages of two approaches, smooth boosting and InfoBoost.

1 Introduction

In recent years, huge data have become available due to the development of computers and the Internet. In knowledge discovery and machine learning tasks, size of such huge data can reach hundreds of gigabytes or more. So it is important to make knowledge discovery or machine learning algorithms scalable. Sampling is one of effective techniques to deal with large data. That is, instead of using whole the data, we can obtain a summary of the data by sampling randomly from it. There are many results on sampling techniques (see, e.g., [4]) and applications to data mining tasks such as decision tree learning [6], support vector machine [2], and boosting [4, 5].

Especially, boosting is simple and efficient learning method among machine learning algorithms. The basic idea of boosting is combining many slightly accurate hypotheses (which we call "weak" hypotheses) into a highly accurate one. Originally, boosting was invented under the boosting by filtering framework, where the booster can sample examples randomly from the whole data [16, 7]. The main advantage of the filtering framework is that the learner need to store less examples. In general, the learner have to store many example enough in order to evaluate its final hypothesis. On the other hand, in boosting by framework, the booster does not have to store all sampled examples, but have to keep examples only for learning weak hypotheses, which is much smaller than those for the final hypotheses. So the boosting by filtering framework seems to fit learning tasks over huge data. However, early boosting algorithms [16, 7] which work in the filtering framework were not practical because they were not adaptive, i.e., they need the prior knowledge on the accuracy of weak hypotheses.

Madaboost, a modification of AdaBoost [8], is the first adaptive boosting algorithm which works in the filtering framework [5]. Combining with adaptive sampling methods [4], Madaboost is shown to be more efficient than AdaBoost over huge data, while keeping the prediction accuracy. By its nature of updating scheme, Madaboost is categorized as one of "smooth" boosting algorithms [18, 9], where the name, smooth boosting, comes from the fact that these boosting algorithms only deal with smooth distributions over data (In contrast, for example, AdaBoost might construct exponentially skew dis-

\[ \text{Email: hatano@i.kyushu-u.ac.jp} \]
tutions over data). Smoothness of distributions enables boosting algorithms to sample data efficiently. Also, smooth boosting algorithms have theoretical guarantees for noise tolerance in the various noisy learning settings, such as statistical query model [5], malicious noise model [18] and agnostic boosting [9].

However it seems that there is still room for improvements on smooth boosting. A non-smooth boosting algorithm, InfoBoost [1] (which is a special form of real AdaBoost [17]), performs more efficiently than other boosting algorithms in the boosting by subsampling framework, where only a bunch of data is given in advance. More precisely, given hypotheses with error $1/2 - \gamma/2$ ($0 < \gamma < 1$), typical boosting algorithms take $O(1/\gamma^2)$ iterations to learn sufficiently accurate hypothesis. On the other hand, InfoBoost learns in from $O(1/\gamma)$ to $O(1/\gamma^2)$ iterations by taking advantage of the situation when weak hypotheses have low false positive error [10, 11]. So InfoBoost can be more efficient at most by $O(1/\gamma)$ times.

The main difference between InfoBoost and other boosting algorithms such as AdaBoost or MadBoost is the criterion for choosing weak hypotheses. Typical boosting algorithms are designed to choose hypotheses whose errors are minimum with respect to given distributions. In contrast, InfoBoost uses an information-theoretic criterion to choose weak hypotheses. The criterion was previously proposed by Kearns and Mansour [12], and also applied to boosting algorithms using decision trees [12] and branching programs [14]. But, so far, no smooth algorithm is known to have such the nice property of InfoBoost.

In this paper, we modify one of smooth boosting algorithms, AdaFlat [9], as it is quite similar to MadBoost and yet simple to analyze. Our modification is derived in a similar way that InfoBoost is given. As a result, we propose a new smooth boosting algorithm with yet another information-theoretic criterion. Our preliminary experiments show that our modification, which we call MadFlat (Modification of AdaFlat), outperforms MadBoost in the filtering framework.

2 Preliminaries

2.1 Learning Model

We adapt the PAC learning model [19]. Let $\mathcal{X}$ be an instance space and let $\mathcal{Y} = \{-1, +1\}$ be a set of labels. We assume an unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$. Further we assume that $f$ is contained in a known class $\mathcal{F}$ of functions from $\mathcal{X}$ to $\mathcal{Y}$. Let $D$ be an unknown distribution over $\mathcal{X}$. The learner has an access to the example oracle $\text{EX}(f, D)$. When given a call from the learner, $\text{EX}(f, D)$ returns an example $(x, f(x))$ where each $x$ is drawn randomly according to $D$. Let $H$ be a hypothesis space, or a set of functions from $\mathcal{X}$ to $\mathcal{Y}$. We assume that $H \supset \mathcal{F}$. For any distribution $D$ over $\mathcal{X}$, error of hypothesis $h \in H$ is defined as $\text{err}_D(h) \overset{\text{def}}{=} \text{Pr}_D(h(x) \neq f(x))$. Let $S$ be a sample, a set of examples $\{(x_1, f(x_1)), \ldots, (x_m, f(x_m))\}$. For any sample $S$, training error of hypothesis $h \in H$ is defined as $\text{err}_S(h) \overset{\text{def}}{=} |\{(x_i, f(x_i)) \in S \mid h(x_i) \neq f(x_i)\}|/|S|$. Now we define PAC learnability as follows.

Definition 1 (Strong learning). A learning algorithm $A$ is a strong learner for $\mathcal{F}$ if and only if, for any $f \in \mathcal{F}$ and any distribution $D$, given $\varepsilon, \delta$ ($0 < \varepsilon, \delta < 1$), a hypothesis space $H$, and access to the example oracle $\text{EX}(f, D)$ as inputs, $A$ outputs a hypothesis $h \in H$ such that $\text{err}_D(h) = \text{Pr}_D(h(x) \neq f(x)) \leq \varepsilon$ with probability at least $1 - \delta$.

On the other hand, an apparently weaker notion of learning was proposed [16].

Definition 2 (Weak learning). A learning algorithm $A$ is a weak learner for $\mathcal{F}$ if and only if, for any $f \in \mathcal{F}$, given a hypothesis space $H$, and access to the example oracle $\text{EX}(f, D)$ as inputs, $A$ outputs a hypothesis $h \in H$ such that $\text{err}_D(h) \leq 1/2 - \gamma/2$ for a fixed $\gamma$ ($0 < \gamma < 1$).

2.2 Boosting Approach

Schapire proved that strong and weak PAC learnability are equivalent to each other for the first time [16]. Especially the technique to construct
a strong learner by using a weak learner is called "boosting". Basic idea of boosting is the following: First, the booster trains a weak learner with respect to different distributions $D_1, \ldots, D_T$ over the domain $X$, and gets different "weak" hypotheses $h_1, \ldots, h_T$ such that $\text{err}_{D_t}(h_t) \leq 1/2 - \gamma_t/2$ for each $t = 1, \ldots, T$. Then the booster combines weak hypotheses $h_1, \ldots, h_T$ into a final hypotheses $h_{\text{final}}$ satisfying $\text{err}_D(h_{\text{final}}) \leq \varepsilon$.

**Subsampling versus Filtering** We consider two frameworks of boosting, **boosting by subsampling** and **boosting by filtering**. In the subsampling framework, the booster is given a sample $S = ((x_1, f(x_1)), \ldots, (x_m, f(x_m)))$ in advance. The booster constructs the final hypothesis whose training $\overline{\text{err}}_S(h_{\text{final}}) \leq \varepsilon$ by training the weak learner over the given sample $S$. Then the generalization error is estimated by using arguments on VC-dimensions or margin (E.g., see [15]). For example, for typical boosting algorithms, $\text{err}_D(h_{\text{final}}) \leq \overline{\text{err}}_S(h_{\text{final}}) + \tilde{O}(\sqrt{T \log |W|/m})$ with high probability, where $T$ is the size of the final hypotheses, i.e., the number of weak hypotheses combined in $h_{\text{final}}$.

In the filtering framework, on the other hand, the booster deals with the whole instance space $X$ through $\text{EX}(f, D)$. By using statistics obtained from calls of $\text{EX}(f, D)$, the booster tries to minimizes $\text{err}_D(h_{\text{final}})$ directly. There are two advantages of the boosting by filtering over the boosting by subsampling. First of all, the space complexity is reduced. Roughly speaking, the booster needs to store $\tilde{O}(T \log |W|)$ examples in the subsampling framework, whereas, the booster only need to store $\tilde{O}(\log |W|)$ examples in the filtering framework. Second, the booster does not have to determine the size of sample $S$ a priori. There advantages are preferable for learning over huge data.

**Smooth Boosting** Smooth boosting algorithms only deal such distributions $D_1, \ldots, D_t$ that are "smooth" with respect to the original distribution $D$. We define the following measure of smoothness.

**Definition 3.** Let $D$ and $D'$ be any distributions over $X$. We say that $D'$ is $\lambda$-smooth with respect to $D$ if $\sup_{x \in X} D'(x)/D(x) \leq \lambda$.

The smoothness parameter $\lambda$ has crucial roles in robustness of boosting algorithms [5, 18, 9]. Also, it affects the efficiency of sampling methods. For example, by rejection sampling, we use $1/\lambda$ calls of $\text{EX}(f, D)$ on average to simulate a call of $\text{EX}(f, D')$ for a distribution $D'$ that is $\lambda$-smooth w. r. t. $D$.

**Our Assumption and Technical Goal** In the rest of the paper, we assume the **weak hypothesis assumption on $W$** as follows: The learner is given a finite set $W$ of hypotheses such that for any distribution $D'$ over $X$, there exists a hypothesis $h \in W$ satisfying $\text{err}_{D'}(h) \leq 1/2 - \gamma/2$ ($0 < \gamma < 1$).

Now our technical goal is to construct an efficient smooth boosting algorithm which works in both the subsampling and the filtering framework.

## 3 Boosting by Subsampling

In this section, we propose a modification of AdaFlat in the subsampling framework. Let

$$\ell(x) = \begin{cases} 
1, & x \geq 0 \\
-1, & -1 < x < 0 \\
0, & x \leq -1.
\end{cases}$$

The description of our modification is given in Figure 3. Given a sample $S = ((x_1, f(x_1)), \ldots, (x_m, f(x_m)))$, and a combined hypothesis $H_t = \sum_{j=1}^{t} \gamma_{h,j}[\pm 1] h_j(x)$ at iteration $t$, pseudo gain of hypothesis $h$ is given as follows:

$$\Delta_t(h) = \frac{m_{h,\pm}}{m} \ell(-f(x_i)H_i(x_i)),$$

where

$$\mu_{h,t}(\pm 1) = \frac{1}{m_{h,\pm}} \sum_{i : h_i(x_i) = \pm 1} \ell(-f(x_i)H_t(x_i)),$$

$$\gamma_{h,t}(\pm 1) = \frac{1}{m_{h,\pm}} \sum_{i : h_i(x_i) = \pm 1} \ell(-f(x_i)H_t(x_i)),$$

$$D_t(x_i) = \frac{\ell(-f(x_i)H_t(x_i))}{\sum_{i : h_i(x_i) = \pm 1} \ell(-f(x_i)H_t(x_i))},$$

and $m_{h,\pm} = |\{i : h(x_i) = \pm 1\}|$. In the case when $m_{h,\pm}$ is zero, we assume that $\mu_{h,t}(\pm 1) = \gamma_{h,t}(\pm 1) = 0$. At each iteration $t$, our algorithm MadaFlat chooses hypothesis $h$ that maximizes its
MadaFlat

Given: $S = \{(x_1, f(x_1)), ..., (x_m, f(x_m))\}$, and $\epsilon (0 < \epsilon < 1)$
begin
1. $D_1(i) \leftarrow 1/m; \ (i = 1, ..., m)$ $H_1(x) \leftarrow 0$; $t \leftarrow 1$
2. while $\epsilon \overline{err}_S(h_{final}) > \epsilon$ do
   a) $h_t \leftarrow \arg \max_{h \in W} \Delta_t(h)$;
   b) $\gamma_{t+1} = \mu(h_t) \gamma_{t+1}$;
   c) $H_{t+1}(x) = H_t(x) + \alpha_t h_t(x) h_t(x)$;
   d) Define the next distribution $D_{t+1}$ as
   \[ D_{t+1}(i) = \frac{\ell(-f(x_i)H_{t+1}(x_i))}{\sum_{i=1}^m \ell(-f(x_i)H_{t+1}(x_i))}, \]
   e) $t \leftarrow t + 1$;
end-while
3. Output the final hypothesis defined by $h_{final}(x) = \text{sign}(H_{T+1}(x))$.
end.

Figure 1: MadaFlat

pseudo gain $\Delta_t(h)$. For simplicity, we denote $\Delta_t = \Delta_t(h_t)$, $\mu_t[\pm 1] = \mu_t[\pm 1]$ and $\gamma_t = \gamma_t$. Further, let us define $\Delta_t = \sum_{i=1}^m \ell(-f(x_i)H_t(x_i))/m$. For each $h_t$, let $\gamma_t = \sum_{i=1}^m f(x_i)h_t(x_i)D_t(i)$. Note that $\text{err}_S(h_t) = 1/2 - \gamma_t/2$.

First, we show that the smoothness of distributions $D_t$.

Proposition 1. During the execution of MadaFlat, each distribution $D_t$ $(t \geq 1)$ is 1-$\epsilon$-smooth with respect to $D_1$, the uniform distribution over $S$.

Next, we prove the time complexity of MadaFlat.

Theorem 2. Assume the weak hypothesis assumption on $W$. Then, MadaFlat outputs a final hypothesis $h_{final}$ satisfying $\epsilon \overline{err}_S(h_{final}) \leq \epsilon$ within $T = O(1/\epsilon^2 \gamma^2)$ iterations.

4 Boosting by Filtering

In this section, we propose MadaFlat$_{filt}$ in the filtering framework. Let
\[
D_t(x) = \frac{D(x)\ell(-f(x)H_t(x))}{\sum_{x \in x} D(x)\ell(-f(x)H_t(x))}.
\]
We define $\Delta_t(h) = p_h \mu_t[+1]^2 \gamma_t[+1] + (1 - p_h) \mu_t[-1]^2 \gamma_t[-1]$, where $p_h = \text{Pr}_D(h(x) = +1)$,
\[
\mu_t[+1] = \frac{\sum_{x \in x} D(x)\ell(-f(x)H_t(x))}{\sum_{x \in x} D(x)},
\]
\[
\mu_t[-1] = \frac{\sum_{x \in x} D(x)\ell(-f(x)H_t(x))}{\sum_{x \in x} D(x)}.
\]

MadaFlat$_{filt}(\epsilon, \delta, \text{EX}(f, D))$
begin
1. Let $H_1(x) = 0$; $t \leftarrow 1$; $\delta_t \leftarrow \delta / 4$.
2. while $\mu_t \geq \frac{\delta_t}{2}$ do
   a) $(h_t, S_t) \leftarrow \text{HSelect}(1/2, \delta_t)$;
   b) $(\mu_t, \mu_t[+1], \mu_t[-1], \gamma_t[+1], \gamma_t[-1])$ ← empirical estimates over $S_t$;
   c) $\gamma_t[+1] = \mu_t[+1] \gamma_t[+1]; \gamma_t[-1] = \mu_t[-1] \gamma_t[-1]$;
   d) $H_{t+1}(x) = H_t(x) + \alpha_t h_t(x) h_t(x)$;
   e) $t \leftarrow t + 1$; $\delta_t \leftarrow \delta_t / 2(t + 1)$;
end-while
3. Output the final hypothesis defined by $h_{final}(x) = \text{sign}(H_{T+1}(x))$.
end.

HSelect($\epsilon, \delta$)
begin
$m \leftarrow 0$; $S \leftarrow \emptyset$; $i \leftarrow 1$; $\Delta_t \leftarrow \delta$; $\delta' \leftarrow \delta / (2|W|)$;
repeat
$(x, f(x)) \leftarrow \text{EX}(f, D)$;
$S \leftarrow S \cup (x, f(x))$; $m \leftarrow m + 1$;
if $m = \frac{|S|}{\bar{\Delta}_t}$ then
  if $\exists h \in W, \Delta(h, S) \geq \Delta_t$ then return $h$ and $S$;
else $\Delta_t \leftarrow \Delta_t/2$; $i \leftarrow i + 1$; $\delta \leftarrow \delta'/(i + 1)|W|$;
end-if
end-repeat
end.

Figure 2: MadaFlat$_{filt}$

and
\[
\gamma_{t+1} = \frac{\sum_{x \in x} D(x)\ell(-f(x)H_t(x))}{\sum_{x \in x} D(x)\ell(-f(x)H_t(x))},
\]
respectively. Also let
\[
\mu_t = \frac{\sum_{x \in x} D(x)\ell(-f(x)H_t(x))}{\sum_{x \in x} D(x)}.
\]
and $\gamma_{t+1} = p_h \gamma_t[+1] + (1 - p_h) \gamma_t[-1]$. As in the previous section, we use the similar notation: $\Delta_t = \Delta_t(h_t), p_h = p_h, \gamma_t[+1] = \gamma_t[+1]$. We denote $\Delta_t$ as the empirical estimate of the parameter a given a sample $S_t$. The description of MadaFlat$_{filt}$ is given in Figure 2.

First, we prove the following lemma.

Lemma 1. Let $\Delta_t = \hat{\mu}_t[+1]^2 \hat{\gamma}_t[+1] + (1 - \hat{\mu}_t)[+1]^2 \hat{\gamma}_t[-1]^2$ be the empirical estimate of $\Delta_t$ given $S_t$. Then it holds for any $\epsilon (0 < \epsilon < 1)$ that
\[
\Pr_{D_t}(\Delta_t \geq (1 + \epsilon)\Delta_t) \leq b_1 e^{-\frac{2\Delta_t}{c_1}},
\]
where $B_1 = \frac{|S_t|}{\Delta_t}$.
and
\[
\Pr_{D^n}(\hat{\Delta}_t \leq (1 - \epsilon)\Delta_t) \leq b_1 e^{-\frac{c_2 \Delta_m}{2}},
\]
where \(b_1 \leq 8\), \(c_1 \leq 600\), and \(c_2 \leq 64\).

Then we show the property of HSelect.

**Lemma 2.** Fix any step \(t\) in MadaFlat. Let \(\Delta_* = \max_{h \in W} \Delta_t(h)\). Then, the following statements hold. (i) With probability at least \(1 - \delta\), \(\text{HSelect}(\epsilon, \delta)\) outputs a hypothesis \(h \in W\) such that \(\Delta_t(h) > (1 - \epsilon)\Delta_*\). (ii) The number of calls of \(\text{EX}(f, D)\) is
\[
O\left(\frac{\log \frac{1}{\delta} + \log |W| + \log \log \frac{1}{\Delta_*}}{\epsilon^2 \Delta_*}\right).
\]

Finally we obtain the following theorem.

**Theorem 3.** With probability at least \(1 - \delta\),

(i) \(\text{MadaFlat}_{\text{fit}}\) outputs the final hypothesis \(h_{\text{final}}\) such that \(\text{err}_{D}(h_{\text{final}}) \leq \epsilon\),

(ii) \(\text{MadaFlat}_{\text{fit}}\) terminates in \(T = O\left(\frac{1}{\epsilon^2 \gamma^2}\right)\) iterations, and

(iii) the number of calls of \(\text{EX}(f, D)\) is
\[
O\left(\frac{\log \frac{1}{\delta} + \frac{1}{\epsilon^2 \gamma^2} + \log |W| + \log \log \frac{1}{\epsilon^2 \gamma^2}}{\epsilon^2 \gamma^4}\right).
\]

### 5 Experimental Results

In this section, we show some experimental results on both artificial and real data sets. Our experiments consist of two parts.

In the first part, we compare MadaFlat, AdaBoost, InfoBoost, and MadaBoost. in the subsampling framework.

For real data, we use some datasets from UCI machine learning repository [3]. Also, we prepare artificial data in order to examine behavior of boosting algorithms in details. To do so we use \(r\)-of-\(k\) function as the target function. An \(r\)-of-\(k\) function \(f\) over boolean domain \((-1, +1)^N\) consists of \(k\) relevant variables and \(f(x) = +1\) if at least \(r\) of the \(k\) relevant variables takes \(+1\), otherwise \(f(x) = -1\).

#### Table 1: Test errors of boosting algorithms in the subsampling framework.

<table>
<thead>
<tr>
<th>dataset</th>
<th>Ada.</th>
<th>InfoB.</th>
<th>Mada.</th>
<th>MadaF.</th>
</tr>
</thead>
<tbody>
<tr>
<td>kr-vs-kp</td>
<td>0.052</td>
<td>0.056</td>
<td>0.052</td>
<td>0.061</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.025</td>
<td>0.025</td>
<td>0.025</td>
<td>0.024</td>
</tr>
<tr>
<td>sick-euthroid</td>
<td>0.060</td>
<td>0.055</td>
<td>0.060</td>
<td>0.057</td>
</tr>
<tr>
<td>spambase</td>
<td>0.23</td>
<td>0.23</td>
<td>0.23</td>
<td>0.23</td>
</tr>
<tr>
<td>100f70</td>
<td>0.19</td>
<td>0.062</td>
<td>0.19</td>
<td>0.045</td>
</tr>
<tr>
<td>200f70</td>
<td>0.074</td>
<td>0.060</td>
<td>0.074</td>
<td>0.042</td>
</tr>
<tr>
<td>300f70</td>
<td>0.073</td>
<td>0.067</td>
<td>0.073</td>
<td>0.051</td>
</tr>
</tbody>
</table>

Note that \(1\)-of-\(k\) function and \(k/2\)-of-\(k\) function correspond to \(k\)-disjunction and \(k\)-majority, respectively. In [11], it is shown that InfoBoost can learn \(r\)-of-\(k\) functions in \(O(rk)\) steps whereas AdaBoost needs \(O(k^2)\) steps when boolean literals are used as weak hypotheses. For \(r = 1, 3, 5\) and \(k = 10\), we fix a \(r\)-of-\(k\) function over 100 boolean variables as the target function, and we generate 10,000 random examples labeled by each \(r\)-of-\(k\) function, where the random examples are drawn so that positive and negative examples are equally likely. The size of data we use varies from about 3,000 to 10,000.

For each dataset, we prepare decision stumps and the constant hypothesis \(+1\) (i.e. the hypothesis that always answers \(+1\)) as weak hypotheses. In each dataset, each record have numeric attributes or binary attributes. For each numeric attribute, we construct a decision stump with a threshold, which predicts \(+1\) or \(-1\) depending on whether the value of the attribute is below the threshold or not. The threshold is chosen so that the training error of the decision stump is minimized. For each binary attribute, we prepare the decision stump which answers the value of the attribute.

We evaluate the boosting algorithms by cross validation. We split each data randomly 10 times, where each example is put into a training set with probability 0.7 and a test set with with probability 0.3. For each training set, we run the boosting algorithms in 100 steps and evaluate their final hypotheses on the test data. The results are summarized in Table 5.
As shown in Table 5, performance of MadaFlat appears to be comparable to those of others on real datasets. Also, MadaFlat is significantly better on artificial datasets, as well as InfoBoost.

In the second part, we compare MadaBoost and MadaFlat in the filtering framework. Basic settings of our experiments in the filtering framework are the same as those in the subsampling framework, except the following: First of all, in order to obtain large datasets, as is done in [4], we inflate the datasets by preparing 100 copies of each record in the data and changing their order randomly. Consequently, the size of the inflated data varies from 300,000 to 1,000,000. Second, instead of running each algorithm in 100 steps, we run them until they sample 10,000 examples. More precisely, we run MadaFlat with HSelect($\epsilon$, $\delta$), where parameter $\epsilon = 0.5$ and $\delta = 0.1$ are fixed. Also, we run MadaBoost with geometric AdaSelect whose parameters are $\epsilon = 2$, $\epsilon = 0.5$ and $\delta = 0.1$.

Third, note that we use heuristics for Lemma 1. Although Lemma 1 gives a theoretical guarantee to approximate the pseudo gain accurately enough, it is too rough to use in practice. By using the central limit theorem, it is not hard to show that $\Delta$ is asymptotically distributed from $N(\Delta, \sigma^2)$, where $\sigma \leq 5\Delta/m$. This analysis implies that it is safe to replace the condition on $m$ in HSelect with

$$m = \left\lceil \frac{10 (\ln \frac{1}{\sqrt{\delta'}} - \frac{1}{2} \ln \ln \frac{1}{\sqrt{\delta'}})}{\Delta_g} \right\rceil.$$ 

In the following experiments, we use this improved heuristics.

Finally, in addition, we apply MadaBoost and MadaFlat for text categorization tasks on a collection of Reuters news (Reuters-21578). We use the modified Apte ("ModApte") split which contains about 10,000 news documents labeled with topics. We choose two major topics ("earnings" and "acquisitions") and for each of two topics, we let boosting algorithms classify whether a news document belongs to the topic or not. As weak hypotheses, we prepare about 30,000 decision stumps corresponding to words. This experiment is done in the same setting of previous ones, except that we do not inflate this dataset. The results are summarized in Table 5.

As indicated in Table 5 and Figure 3, MadaFlat often outperforms MadaBoost.

6 Summary and Future Work

In this paper, we propose a modification of AdaFlat that uses an information-theoretic criterion for choosing hypotheses. Our preliminary experiments show that our modification appears to outperform MadaBoost in the filtering framework. As future work, advantages and noise tolerance of MadaFlat are yet to be investigated theoretically. Also, we plan to conduct experiments over much larger data.

Acknowledgments

This work is supported in part by the 21st century COE program at Graduate School of Information Science and Electrical Engineering in Kyushu University.

References


