Journal of Machine Learning Research 4 (2003) 101-117 Submitted 1/03; Published 5/03

Optimally-Smooth Adaptive Boosting and Application to Agnostic Learning

Dmitry Gavinsky GAVINSKY@CPSC.UCALGARY.CA
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada, T2N 1N4

Editor: Dana Ron

Abstract

We describe a new boosting algorithm that is the first such algorithm to be both smooth and adaptive. These two features make possible performance improvements for many learning tasks whose solutions use a boosting technique.

The boosting approach was originally suggested for the standard PAC model; we analyze possible applications of boosting in the context of agnostic learning, which is more realistic than the PAC model. We derive a lower bound for the final error achievable by boosting in the agnostic model and show that our algorithm actually achieves that accuracy (within a constant factor).

We note that the idea of applying boosting in the agnostic model was first suggested by Ben-David, Long and Mansour (2001) and the solution they give is improved in the present paper. The accuracy we achieve is exponentially better with respect to the standard agnostic accuracy parameter β.

We also describe the construction of a boosting "tandem" whose asymptotic number of iterations is the lowest possible (in both γ and ε) and whose smoothness is optimal in terms of $\tilde{O}(\cdot)$. This allows adaptively solving problems whose solution is based on smooth boosting (like noise tolerant boosting and DNF membership learning), while preserving the original (non-adaptive) solution's complexity.

1. Introduction

Boosting is a learning method discovered by Schapire (1990). It proves computational equivalence between two learning models: the model of *distribution-free (strong) PAC-learning* and that of *distribution-free weak PAC-learning* (the PAC-model was first introduced by Valiant, 1984; the strong and the weak cases were distinguished by Kearns and Valiant, 1994). This (theoretical) equivalence between the two models may be used to solve various problems in the domain of Learning Theory—a number of such problems are currently known whose "strong" solution was achieved by initially constructing a weak learner and then applying a boosting algorithm.

On the other hand, for many concept classes no weak (PAC) learner has been found so far (e.g., for the class of DNF formulas, see Jackson 1997). In such cases one possible approach is to modify the learning model to make the task of learning simpler. For example, in the case of DNF learning a solution is known for the "simplified" PAC model, where the learner is allowed to ask membership queries and the target distribution is always uniform (Jackson, 1997); at the same time, in its unrestricted form the task is considered rather difficult.

It turns out that the boosting approach, originally developed for the standard PAC model, may be adapted to other more or less similar learning models. For example, the solution suggested in Jackson (1997) for the DNF learning problem is partially based on a previously known boosting algorithm, adapted for the uniform model with allowed membership queries.

Recently Ben-David, Long and Mansour (2001) have shown that the boosting approach may also be applied in the model of agnostic learning, which is in a sense "more practical" than PAC (see Subsection 1.1.3 for model description).

In this paper we construct a boosting algorithm whose requirements for some resources are "optimally modest"; as a result, this algorithm may be used in a number of "near-PAC" learning models, achieving the optimal performance.

1.1 Learning Models

All the models considered in this paper have one feature in common: a learner always communicates with a D-oracle which produces examples $(x, f(x))$, where f is the target function and x is chosen from the domain X according to the distribution D.

In this paper we will always assume that f is binary-valued to $\{-1,1\}$. We will sometimes use as a complexity parameter a measure of the representation size of f, which we denote by $I(f)$ (its definition depends on the target class and is usually clear from the context). Therefore the difference between the following models lies in the requirements for the learner's complexity and accuracy.

1.1.1 PAC LEARNING (STRONG)

The target function f is assumed to come from some hypotheses class C. The learner receives parameters ε and δ; its goal is to produce with probability at least $1-\delta$ a binary hypothesis h_{fin}, satisfying:

$$\varepsilon_{h_{\text{fin}}} \triangleq \Pr_D[h_{\text{fin}}(x) \neq f(x)] \leq \varepsilon.$$

The time complexity of the learner should be polynomial in $1/\varepsilon$, $log(1/\delta)$ and $I(f)$. (Sometimes δ is referred to as a *confidence parameter*.)

1.1.2 PAC LEARNING (WEAK)

The target function f is assumed to come from some hypotheses class C. The learner receives a parameter δ; its goal is to produce with probability at least $1-\delta$ a hypothesis h_{fin}, satisfying:

$$\gamma_{h_{\text{fin}}} \triangleq \mathbf{E}_D[h(x)\cdot f(x)] \geq \gamma_{min},$$

where γ_{min} decreases inversely polynomially in $I(f)$ (we will use the notation γ_h to denote the value of $\mathbf{E}_D[h(x)\cdot f(x)]$ for any real-valued h defined over X). In the context of weak PAC learning we will not require the produced hypothesis to be binary (it may be real-valued to $[-1,1]$).

The time complexity of the learner should be polynomial in $I(f)$ and $log(1/\delta)$.

1.1.3 AGNOSTIC LEARNING

In the framework of agnostic learning no a priori assumptions regarding the target function f are made; instead we require that the learner's response h_{fin} should be "close to the best possible" up to some parameter. Formally, agnostic learning means that we define some learning hypotheses class

F to be used for learning. For any target function f, the best possible accuracy a hypothesis from F can achieve is either equal or infinitesimally close to

$$\operatorname*{err}_D(F) \triangleq \inf_{h' \in F} \left\{ \Pr_D \left[h'(x) \neq f(x) \right] \right\}. ^{1}$$

The learner receives a parameter δ and has to produce with probability at least $1-\delta$ a binary hypothesis h_{fin}, satisfying:

$$\varepsilon_{h_{\text{fin}}} \leq \operatorname*{err}_D(F) + \beta,$$

for some β announced a priori. A learner satisfying this is called a β-*optimal* learner.

We do not define a general notion of agnostic learning efficiency here; for a detailed overview of the agnostic learning model, see Kearns, Schapire and Sellie (1994).

1.2 Boosting

A boosting algorithm B is supplied with an auxiliary algorithm WL (the *weak learner*). B communicates with WL, in a sequence of sessions: During session i B chooses a distribution D_i over X and emulates for WL a D_i-oracle, answering according to the target function f. In response WL has to learn f w.r.t. D_i, each session ends when WL constructs its hypothesis.

We will denote the total number of performed boosting sessions by T and the weak learner's response to D_i by h_i. We will sometimes refer to individual sessions as "iterations".

The boosting algorithm itself "faces" some target distribution D, according to which it receives instances from the oracle. After the T'th session is finished, B produces its final hypothesis h_{fin} whose accuracy w.r.t. D is higher than the accuracies of the received weak hypotheses (w.r.t. corresponding D_i-s).

When the boosting is performed in the PAC model, the weak learner is usually a weak PAC-learner and the booster itself is required to satisfy the strong PAC requirements (see Subsection 1.1.1). In this case the booster is considered efficient when its complexity is polynomial in the γ_{min} parameter of WL, as well as in standard PAC complexity parameters. Note that when WL is an efficient weak PAC learner (i.e., γ_{min} is inverse-polynomial low), the whole tandem B+WL constitutes an efficient strong PAC learner.

For the case of agnostic model (see Subsection 1.1.3) we will derive efficiency requirements for both B and WL later in the paper. (Since the PAC model is "more natural" for boosting, we will describe and analyze our algorithm as a PAC booster, and then we will separately consider its usage under agnostic settings.)

1.2.1 Boosting by Sampling versus Boosting by Filtering

In this paper we consider two boosting modes: boosting by *sampling* and boosting by *filtering*.

Boosting by sampling means that the learning is performed in two stages. In the first stage the algorithm collects a sufficient number of learning examples (i.e., a subset $S \subset X$ together corresponding values of f) by repeatedly calling the oracle. In the second stage the booster performs learning over this collection of examples only. The training sample S is of polynomial size.

[1] Note that sometimes the agnostic model is defined so that the class used for accuracy evaluation is allowed to differ from that used to produce approximating hypotheses.

The booster's goal is to achieve some required accuracy over the training set (which is often the absolute accuracy, so that the hypothesis coincides with f for each $x \in S$). Then information-theoretic techniques like VC theory (Vapnik, 1982) and Occam's Razor (Blumer, Ehrenfeucht, Haussler and Warmuth, 1989) may be applied to measure the overall accuracy w.r.t. D of the same hypothesis.

Of course, the hypothesis may be not accurate over whole X even if it is correct over S. Such "switching" from sub-domain S to whole X is sometimes called *generalization*, and the additional error introduced by generalization is called a *generalization error*.

In contrast, when boosting by filtering the booster takes the whole set of instances as its learning domain. The examples received by the booster from the PAC-oracle are not stored, but are "filtered": each example is either forwarded to WL or rejected (i.e., is not used at all).

This approach has two obvious advantages over boosting by sampling: the space complexity is reduced (the examples are not stored) and no generalization error is introduced. At the same time, the analysis and the algorithm itself become slightly more complicated, because now the booster cannot get exact statistics by running through all the instances of the domain and needs to use some estimation schemes (based on various statistical laws of large numbers, like the Chernoff bound).

Among the possible reasons for using boosting by sampling is that the booster is not smooth: In general, a distribution which is not polynomially near-D cannot be efficiently simulated using repeated calls to a D-oracle, when X is superpolynomially large.

1.2.2 SMOOTH BOOSTING AND ADAPTIVE BOOSTING

In this paper we consider two special features of boosting algorithm: *smoothness* and *adaptiveness*.

The term "adaptiveness" means that the algorithm doesn't require a priori lower bound for γ_i; instead the algorithm "takes advantage" of each single weak hypothesis, as good as it is. More formally, while a non-adaptive booster has its complexity bounds polynomial in $1/\min\{\gamma_{h_i} | 1 \le i \le T\}$, adaptive algorithm's running time is polynomial in

$$1/\mathop{\mathbf{E}}_{1\le i\le T}\left[poly(\gamma_{h_i})\right].$$

(For example, we will derive complexity bounds for our algorithm in terms of $1/\mathbf{E}_{1\le i\le T}\left[\gamma_i^2\right]$.)

The term "smoothness" means that the distributions D_i emulated by the booster do not diverge dramatically from the target distribution D of the booster itself. To measure the "smoothness" of a distribution D_i, we define a *smoothness parameter*:

$$\alpha_i \triangleq \sup_{x\in X}\frac{D_i(x)}{D(x)}.$$

We define a smoothness parameter for a boosting algorithm as sup of possible smoothness parameters of distributions emulated by it. A boosting algorithm with smoothness parameter α will be called α-smooth.

For example, if the target distribution is known to be always uniform and the booster is smooth, WL will has to deal only with distributions which are near to the uniform (which simplifies its quest sometimes). This idea is basic for the only known solution of DNF membership learning w.r.t. uniform (Jackson, 1997, Klivans and Servedio, 1999). Among other known applications for smooth boosting algorithms are noise-tolerant learning (Freund, 1999, Domingo and Watanabe, 2000, Servedio, 2001), learning via extended statistical queries (Bshouty and Feldman, 2001) and agnostic boosting (Ben-David, Long and Mansour, 2001).

1.3 Our Results

In this paper we construct a boosting algorithm called *AdaFlat*, which is optimal from several points of view. We show that *AdaFlat* is adaptive and near-optimally smooth (within a constant multiplicative factor of 2).

Let us consider a variation of the above adaptiveness notation: Call a booster *output-adaptive* if the size of the final hypothesis it produces is adaptive (i.e., is bounded above by a polynomial in $1/\mathbf{E}_{1\le i\le T}\left[poly(\gamma_{h_i})\right]$). For notation clarity we will sometimes refer to the “usual” adaptiveness (as defined above) as *time-adaptiveness*.

If the size of the final hypothesis depends polynomially on γ_{h_i}-s, then time-adaptiveness naturally implies output-adaptiveness. To the best of our knowledge, this is the case for all known boosting algorithms.

Claim 1 *If the size of the final hypothesis depends polynomially on γ_{h_i}-s, then smoothness is essential for performing adaptive boosting over a superpolynomially large domain.*

Proof of Claim 1 The booster is output-adaptive, assume that it is not smooth.

On the one hand, non-smoothness obliges to use boosting by sampling. On the other hand, for an output-adaptive booster it is not in general possible to bound from above “adaptively” the size of the final hypothesis before the boosting process ends.

In order to fix the size of the learning sample S so that the final error (after generalization) would be at most ε, it is essential to have an upper bound for the size of the final hypothesis (e.g., see Vapnik 1982 and Blumer, Ehrenfeucht, Haussler and Warmuth 1989 for overview of generalization techniques).

Because the size of the learning domain must be fixed before boosting by sampling starts and at that time we do not have an adaptive upper bound for the size of the final hypothesis, we have to use a learning sample of non-adaptive size, which turns the boosting algorithms itself into non-time-adaptive.

$\blacksquare_{\textit{Claim 1}}$

To the best of our knowledge, *AdaFlat* is the first smooth adaptive booster.

An algorithm *MadaBoost* constructed by Domingo and Watanabe (2000) is smooth, but it is adaptive only for a non-increasing sequence of weak hypotheses accuracies (i.e., when $\gamma_{h_{i+1}} \le \gamma_{h_i}$ for $1 \le i \le T-1$). Besides, their result applies only to the case of binary-valued weak hypotheses, which seems to produce some difficulties when Fourier spectrum approach is used for weak learning (Mansour, 1994, Blum, Furst, Jackson, Kearns, Mansour and Rudich, 1994, Bshouty, Jackson and Tamon, 1999).

Another similar result was achieved by Bshouty and Gavinsky (2001): their algorithm is smooth and output-adaptive, but not time-adaptive: While their algorithm makes adaptive number of boosting iterations and constructs final hypothesis of adaptive size, the time complexity of each iteration depends upon the value of γ_{min} (which should be provided before the boosting starts).

We show that *AdaFlat* may be used in the framework of agnostic learning. We derive a lower bound for the final error achievable by agnostic boosting; we show that our algorithm achieves that accuracy (within a constant factor of 2). Our upper bound on the final error is

$$\frac{1}{1/2-\beta}\operatorname*{err}_D(F)+\zeta,$$

where ζ is any real so that the time complexity of the solution is polynomial in $1/\zeta$.

The idea of applying boosting in the agnostic model is due to Ben-David, Long and Mansour (2001). Our result is an exponential improvement w.r.t. β over the solution suggested in Ben-David, Long and Mansour (2001), whose error is upper bounded by

$$\frac{1}{1/2-\beta}\operatorname*{err}_D(F)^{2(1/2-\beta)^2/\ln(1/\beta-1)}+\zeta.$$

(In particular, this answers an open question posed in Ben-David, Long and Mansour 2001.)

Algorithm *AdaFlat* performs the lowest possible asymptotic number of boosting iterations in terms of γ.

Next we construct a "boosting tandem", consisting of *AdaFlat* "joined" with another boosting algorithm (this approach was first used by Freund (1992) and since then has become very popular for constructing boosting algorithms). The underlying idea is to use one booster (*AdaFlat*, in our case) to boost hypotheses from "weak" accuracy $\frac{1}{2}+\gamma$ to some fixed accuracy $\frac{1}{2}+\mathbf{const}$, and then to amplify $(\frac{1}{2}+\mathbf{const})$-accurate hypotheses to $1-\varepsilon$ using another boosting algorithm. This approach is used when the first booster is more efficient in terms of γ and the second one is more efficient in terms of ε.

Moreover, since adaptiveness means certain behavior of a booster w.r.t. the parameter γ and *AdaFlat* is used as the 'bottom level" for the tandem, the whole construction remains adaptive.

Naturally, the final hypothesis structure of the tandem is more complicated. On the other hand, using this approach we achieve the lowest possible asymptotic number of boosting iterations both in γ and in ε, and also the lowest possible smoothness factor in terms of $\tilde{O}$.[2]

Since our tandem algorithm is also adaptive, it may be used in order to solve adaptively and with optimal number of iterations various boosting tasks, including those requiring that the boosting algorithm be smooth.

2. Preliminaries

For simplicity, in our analysis of boosting schemes we will not allow WL to fail.

2.1 Agnostic Boosting Approach

The main idea standing behind agnostic boosting is as follows: suppose we are given a β-optimal learning algorithm; it will be used as a weak learner and therefore is referred to by WL.

Consider some target distribution D. By definition, the weak learner being executed "in a straightforward manner" must provide a $(1-\mathrm{err}_D(F)-\beta)$-accurate hypothesis in the worst case. Let us modify slightly the target distribution thus achieving a new distribution D_i of smoothness α_i (measured w.r.t. D). Obviously, it holds that

$$\operatorname*{err}_{D_i}(F)\leq\alpha_i\cdot\operatorname*{err}_D(F),$$

and WL must provide a hypothesis h_i whose error is $\alpha_i\cdot\mathrm{err}_D(F)+\beta$, in the worst case.

If distribution D_i was produced by a boosting algorithm on stage i, then using boosting notation we may write:

$$\alpha_i\cdot\operatorname*{err}_D(F)+\beta\geq\varepsilon_i=\frac{1}{2}-\gamma_i,$$

[2] We use the symbol $\tilde{O}(t(n))$ for $O(t(n))\cdot\mathrm{poly}(\log t(n))$, where $t(n)$ is any function of n.

which leads to

$$\gamma_i \geq \frac{1}{2} - \beta - \alpha_i \cdot \underset{D}{\mathrm{err}}(F). \tag{1}$$

While Ben-David, Long and Mansour (2001) mainly considered agnostic learning in the context of boosting by sampling, our result applies equally to both boosting by sampling and boosting by filtering. In fact, it seems that the main difficulty lies in constructing a weak learner; possible solutions for this determine the applicability and the efficiency of the boosting methods. Some examples of "agnostic weak learners" may be found in Ben-David, Long and Mansour (2001).

3. Optimally-Smooth Adaptive Boosting

In this section we construct a booster working by sampling and in Section 5 we build a modification of the algorithm which works by filtering. For the construction of *AdaFlat* we modify another algorithm, first introduced by Impagliazzo (1995) in a non-boosting context and later recognized as a boosting algorithm by Klivans and Servedio (1999).

Algorithm *AdaFlat* is represented in Figure 1, the following notation is used:

$$N_i(s) \triangleq f(s) \cdot \textstyle\sum_{j=0}^{i-1} l_j \cdot h_j(s),\ N_0(s) \triangleq 0,$$

$$m(N) \triangleq \begin{cases} 1 & N \leq 0 \\ 1-N & 0 < N < 1 \\ 0 & 1 \leq N \end{cases},$$

$$\mu_i \triangleq \frac{\sum_{s \in S} m(N_i(s))}{|S|},\ \gamma_i(s) \triangleq \frac{h_i(s) \cdot f(s)}{2},\ \mathrm{sign}(y) \triangleq \begin{cases} 1 & y \geq 0 \\ -1 & y < 0 \end{cases}.$$

When the domain S is of polynomial size, simulation of D_i-s is straightforward. Note that the number of iterations performed by the algorithm defines the number of weak hypotheses combined in a final hypothesis.

```
AdaFlat(WL, S, ε)
    set: $i = 0$
    while $\Pr_{s \in S}[h_{\mathrm{fin}}(s) \neq f(s)] \geq \varepsilon$
        define: $D_i(s) \triangleq \frac{m(N_i(s))}{\sum_{s \in S} m(N_i(s))}$
        call $WL$, providing it with distribution $D_i$;
          denote the returned weak hypothesis by $h_i$
        set: $\gamma_i = \sum_{s \in S} D_i(s) \cdot \gamma_i(s)$
        set: $l_i = 2\mu_i \gamma_i$
        define: $h_{\mathrm{fin}}(s) \triangleq \mathrm{sign}\left(\sum_{j=0}^{i} l_j \cdot h_j(s)\right)$
        set: $i = i + 1$
    end-while
    Output the final hypothesis $h_{fin}$
```

Figure 1: The *AdaFlat*(WL, S, ε) hypothesis boosting algorithm.

3.1 *AdaFlat*'s Analysis

Claim 2 *Algorithm AdaFlat executed with parameters (WL, S, ε) performs*

$$T \leq \frac{\varepsilon^{-2}}{4\mathbf{E}_{0\leq i<T}\left[\gamma_i^2\right]} \tag{2}$$

boosting iterations and produces a final hypothesis which is accurate over S and possesses the structure of a weighted majority vote of T weak hypotheses. The smoothness parameter of AdaFlat satisfies:

$$\alpha \leq \varepsilon^{-1}. \tag{3}$$

Proof of Claim 2 Define the following reward function:

$$B(N) \triangleq \begin{cases} N & N \leq 0 \\ N - \frac{N^2}{2} & 0 < N < 1 \\ \frac{1}{2} & 1 \leq N \end{cases}.$$

As follows from a second-order Taylor expansion, for any $c \in \mathbf{R}$ it holds:

$$B(N+c) \geq B(N) + \frac{dB}{dN} \cdot c + \inf B''(N) \cdot \frac{c^2}{2},$$

where $B''(N) \triangleq \min\{\frac{dB^2}{dN \cdot dN^+}, \frac{dB^2}{dN \cdot dN^-}\}$. By noting that $m(N) = \frac{dB}{dN}$, we get:

$$B(N+c) \geq B(N) + m(N) \cdot c - \frac{c^2}{2}.$$

Further, denote:

$$\begin{array}{l} c_i(s) \triangleq N_i(s) - N_{i-1}(s) = l_i \cdot h_i(s) \cdot f(s) = 2l_i\gamma_i(s), \\ B_i \triangleq \frac{\sum_{s\in S} B(N_i(s))}{|S|}, \\ \Delta_B^i \triangleq B_{i+1} - B_i. \end{array}$$

Consequently,

$$\begin{array}{rl} \Delta_B^i & \geq \frac{\sum_{s\in S} m(N_i(s))\cdot c_i(s)}{|S|} - \frac{\sum_{s\in S} c_i^2(s)}{2|S|} \\ & \geq 2l_i \frac{\sum_{s\in S} m(N_i(s))\cdot\gamma_i(s)}{\sum_{s\in S} m(N_i(s))} \cdot \mu_i - \frac{l_i^2}{2} \\ & = 2l_i\gamma_i\mu_i - \frac{l_i^2}{2}\ , \end{array}$$

and using the expression set for l_i by *AdaFlat*, we get:

$$\Delta_B^i \geq 2\gamma_i^2 \cdot \mu_i^2. \tag{4}$$

Since $B(N) \leq \frac{1}{2}$, the last inequality leads to

$$T \leq \frac{1}{4\mathbf{E}_{0\leq i<T}\left[\gamma_i^2 \cdot \mu_i^2\right]}. \tag{5}$$

We can see that the number of iterations T depends on $\mathbf{E}_{0\le i<T}\left[\gamma_i^2\right]$ (rather than on $\min^2_{0\le i<T}[\gamma_i]$), and therefore the algorithm is adaptive.

Next, it holds that

$$\frac{1}{\alpha_i} = \frac{1}{|S|}\cdot\frac{1}{\max_{s\in S} D_i(s)} = \frac{\mu_i}{\max_{s\in S} m(N_i(s))}. \tag{6}$$

But the algorithm would stop as soon as the error of the last constructed h_{fin} becomes less or equal to ε, so for each i it holds that

$$\operatorname*{argmax}_{s\in S} m(N_i(s)) = \min_{s\in S} N_i(s) \le 0,$$

and therefore $\max_{s\in S} m(N_i(s)) = 1$. At the same time, this means that $\mu_i \ge \varepsilon$ as long as *AdaFlat* continues to run (which follows from the definition of μ_i). Applying this to (6), we get

$$\alpha_i^{-1} = \mu_i \ge \varepsilon, \tag{7}$$

which leads to (3).

Combining (7) with (5), we receive statement (2), and the result follows.

$\blacksquare$*Claim 2*

4. Agnostic Boosting

In this section we apply *AdaFlat* to agnostic boosting. (We refer to *AdaFlat* which works by sampling and not to *AdaFlat*$_{Filt}$ introduced in Section 5; however, algorithm *AdaFlat*$_{Filt}$ may be used for agnostic boosting as well.) As a result, we achieve upper bound on final error of

$$\frac{1}{1/2-\beta}\operatorname*{err}_D(F)+\zeta,$$

where ζ is any real, so that the time complexity of the solution is polynomial in $1/\zeta$, as well as in other standard complexity parameters.

The analysis of our new application is straightforward: Combining (1), (4) and (7) gives us that:

$$\begin{aligned}
&\Delta_B^i \ge 2\left(\tfrac{\gamma_i}{\alpha_i}\right)^2 \ge \left(\varepsilon\left(\tfrac{1}{2}-\beta\right)-\text{err}_D(F)\right)^2,\\
&T \le \tfrac{1}{4}\left(\varepsilon\left(\tfrac{1}{2}-\beta\right)-\text{err}_D(F)\right)^{-2},\\
&\varepsilon \le \frac{\frac{1}{2\sqrt{T}}+\text{err}_D(F)}{\frac{1}{2}-\beta}.
\end{aligned}$$

That is, applying *AdaFlat* to the task of agnostic boosting allows to get a hypothesis which $\left(\frac{\text{err}_D(F)}{\frac{1}{2}-\beta}+\zeta\right)$-approximates the target concept. For that, *AdaFlat* needs to perform

$$T \le \frac{1}{4}\left(\zeta\cdot\left(\frac{1}{2}-\beta\right)\right)^{-2}$$

iterations.

Recall that if we straightly apply WL, we get a $(\mathrm{err}_D(F)+\beta)$-approximation. Therefore, our approach actually amplifies WL if and only if

$$\underset{D}{\mathrm{err}}(F) < \beta \cdot \frac{1-2\beta}{1+2\beta}.$$

Note that asymptotic evaluation would work well for the number of iterations needed, but is not sufficient for the smoothness estimation (3). That is because the smoothness property determines the "base" for the final error expression $\left(\frac{\mathrm{err}_D(F)}{\frac{1}{2}-\beta}\right)$, which seems to be rather "hard" (in particular, cannot be decreased simply by performing a larger number of boosting iterations). The same thing holds regarding the parameter β.

In Section 6 we show that $AdaFlat$ (and $AdaFlat_{Filt}$) are near-optimally smooth up to the constant multiplicative factor of 2, therefore the final hypothesis has half the best possible relative correspondence with the target for the agnostic boosting approach.

5. Boosting Using Filtering

Algorithm $AdaFlat_{Filt}$ is shown in Figure 2. Note that the algorithm receives confidence parameter δ. Recall that now the target distribution is D itself and the instance space is $\{(x, f(x)) \mid x \in X\}$.

The algorithm has two subroutines: $D_i gen$ which is used to produce examples for WL and $Evaluate(V, b-a, \delta)$ which, with probability δ at least, returns a value μ' s.t. $|\mu' - \mathbf{E}[V]| \le \frac{\mathbf{E}[V]}{5}$ when V receives values from $[a,b]$ and $\mathbf{E}[V] \neq 0$. The latter subroutine is based on the Chernoff bound, its time complexity is

$$O\left(\frac{(b-a)^2 \cdot \ln(\delta^{-1}) \cdot \ln((\mathbf{E}[V])^{-1})}{(\mathbf{E}[V])^2}\right).$$

5.1 $AdaFlat_{Filt}$'s Analysis

Denote by $T[WL]$ the time complexity of WL running over the instance space X, and by $Q[WL]$ the corresponding query complexity (i.e., the number of requested examples).

The time and query complexity bounds introduced by the following claim are not tight, they are reconsidered and improved in Subsection 5.2.

Claim 3 *Suppose that algorithm $AdaFlat_{Filt}$ is executed with parameters $(WL, \varepsilon, \delta)$. Then with probability at least δ the following statements hold:*

- *The algorithm performs*

$$T \le \frac{3}{4\varepsilon^2 \cdot \mathbf{E}_{0\le i<T}\left[\gamma_i^2\right]} \tag{8}$$

  boosting iterations.

- *The algorithm produces a final hypothesis possessing the structure of a weighted majority vote of T weak hypotheses whose prediction error over the learning domain (X) is ε at most.*

- *The smoothness parameter of $AdaFlat_{Filt}$ satisfies*

$$\alpha \le \varepsilon^{-1}.$$

```
$AdaFlat_{Filt}(WL, \varepsilon, \delta)$
    set: $i = 0,\ \delta_0 = \frac{\delta}{2}$
    while $\mu'_i(m) \geq \frac{4\varepsilon}{5}$
        call $WL$, providing it with distribution generated by $D_i gen$;
          denote the returned weak hypothesis by $h_i$
        set: $\mu'_i(m) = Evaluate\left(m(N_i(s))|_{s\sim D}, 1, \frac{\delta_i}{2}\right)$
        set: $\gamma'_i = Evaluate\left( \left(\gamma_i(s)\cdot m(N_i(s))\right)\Big|_{s\sim D}, 1, \frac{\delta_i}{2}\right)$
        set: $l'_i = 2\mu'_i(m)\gamma'_i,\ i = i+1,\ \delta_i = \frac{2}{3}l'^2_{i-1}\cdot\delta$
    end-while
    define: $h_{\mathrm{fin}}(s) \triangleq \mathrm{sign}\left(\sum_{j=0}^{i} l'_j \cdot h_j(s)\right)$
    Output the final hypothesis $h_{fin}$

$D_i gen$
    do
        get $(x_j, f(f_j))$ from the oracle; choose $r \sim \mathbf{U}_{0,1}$
        if $(r < m(N_i(s)))$ then return $(x_j, f(x_j))$
    end-do

$Evaluate(V, b-a, \delta)$
    set: $\mu_g = \frac{1}{2},\ i = 0,\ \sigma = 0,\ \delta = \frac{\delta}{2}$
    do
        set: $i = i+1,\ \sigma = \sigma + < sample\ from\ V >$
        if $i = \left\lceil \frac{18(b-a)^2 \ln(\frac{2}{\delta})}{\mu_g^2} \right\rceil$ then
            set: $\mu' = \frac{\sigma}{i}$
            if $|\mu'| \geq \mu_g$ then return $\mu'$
            set: $\mu_g = \frac{\mu_g}{2},\ \delta = \frac{\delta}{2}$
        end-if
    end-do
```

Figure 2: The $AdaFlat_{Filt}(WL, \varepsilon, \delta)$ hypothesis boosting algorithm.

- *The query complexity of the algorithm is*

$$\tilde{O}\left(\frac{\varepsilon^{-1}\cdot Q[WL] + \mathbf{E}_{0\leq i<T}\left[\gamma_i^{-2}\right] + \varepsilon^{-2}}{\varepsilon^2 \cdot \mathbf{E}_{0\leq i<T}\left[\gamma_i^2\right]}\right). \tag{9}$$

- *The time complexity of the algorithm is*

$$\tilde{O}\left(\frac{\varepsilon^{-1}\cdot Q[WL] + T[WL] + \mathbf{E}_{0\leq i<T}\left[\gamma_i^{-2}\right] + \varepsilon^{-2}}{\varepsilon^2 \cdot \mathbf{E}_{0\leq i<T}\left[\gamma_i^2\right]}\right). \tag{10}$$

Proof of Claim 3 Notice that $AdaFlat_{Filt}$ is smooth as well, Equation (3) still holds.

Now we calculate the overall number T of iterations (by adjusting slightly the analysis given for $AdaFlat$ in Subsection 3.1). Start by assuming that all the estimations performed by $Evaluate$ (lines 4 and 5 of $AdaFlat_{Filt}$) are accurate within a relative factor of $\frac{1}{5}$ (i.e., within 20% accuracy), later we show that the probability of this event is $1-\delta$ at least.

That is, suppose that during the i'th iteration the assumption holds; therefore, the resulting value received for l'_i estimates within a relative factor of $\frac{1}{2}$ the "real" value of l_i, as defined before. In this case, Equation (4) can be rewritten as follows:

$$\Delta_B^i \geq \frac{3}{2} \cdot \gamma_i^2 \mu_i^2 .$$

The halting condition $\left(\mu'_i(m) < \frac{4\varepsilon}{5}\right)$ guarantees, on the one hand, that the final error is smaller than ε (i.e., $AdaFlat_{Filt}$ is accurate), and, on the other hand, that throughout the iterations μ_i is at least $\frac{2\varepsilon}{3}$. This leads to (8).

Next, we prove the following statement: the probability that our estimations accuracy assumption fails during stage 0 and k following stages is $(\frac{1}{2}+B_k)\cdot\delta$ at most. Note that, under the assumption of statement correctness for stages $0-k$, the value of δ_{k+1} set in line 6 is not greater than $\Delta_B^k \cdot \delta$, and the result follows. As it is always true that $B_k \leq \frac{1}{2}$, the failure probability is bounded from above by δ and $AdaFlat_{Filt}$ satisfies the confidence requirement.

It remains to evaluate the time and number of queries consumed by each boosting iteration (under the estimations accuracy assumption). It holds for $i \geq 1$ that $\delta_i > \frac{1}{15}\varepsilon^2\gamma_{i-1}^2\delta$, therefore two calls to $Evaluate$ take

$$O\left(\ln(\delta^{-1}\varepsilon^{-1}\gamma_{i-1}^{-1}) \cdot \left(\frac{\ln(\gamma_i^{-1})}{\gamma_i^2} + \frac{\ln(\varepsilon^{-1})}{\varepsilon^2}\right)\right).$$

Getting a single example from $D_i gen$ takes in average

$$\frac{1}{\mu_i} \leq \varepsilon^{-1}$$

time, therefore the (average) query complexity of a single call to WL is

$$O\left(\varepsilon^{-1} \cdot Q[WL]\right).$$

The resulting query complexity of a single boosting iteration is

$$O\left(\varepsilon^{-1} \cdot Q[WL] + \ln(\delta^{-1}\varepsilon^{-1}\gamma_{i-1}^{-1}) \cdot \left(\frac{\ln(\gamma_i^{-1})}{\gamma_i^2} + \frac{\ln(\varepsilon^{-1})}{\varepsilon^2}\right)\right), \tag{11}$$

and the time complexity is

$$O\left(\varepsilon^{-1} \cdot Q[WL] + T[WL] + \ln(\delta^{-1}\varepsilon^{-1}\gamma_{i-1}^{-1}) \cdot \left(\frac{\ln(\gamma_i^{-1})}{\gamma_i^2} + \frac{\ln(\varepsilon^{-1})}{\varepsilon^2}\right)\right). \tag{12}$$

In terms of $\tilde{O}$, these expressions respectively correspond to

$$\tilde{O}\left(\varepsilon^{-1} \cdot Q[WL] + \gamma_i^{-2} + \varepsilon^{-2}\right)$$

queries and

$$\tilde{O}\left(\varepsilon^{-1} \cdot Q[WL] + T[WL] + \gamma_i^{-2} + \varepsilon^{-2}\right)$$

time needed for the i'th boosting iteration.

Obviously, the case of $i = 0$ will not rise the average iteration complexity resulting from these observations. Combining them with Equation (8) gives the required bounds (9) and (10), and the result follows.

$\blacksquare$*Claim 3*

5.2 Implementation Considerations

The complexity bounds (11) and (12) (see the proof of Claim 3) depend on both γ_i and γ_{i-1}, which may be viewed as a certain kind of "memory", or a loss of adaptiveness. This effect may be removed by halving δ after each iteration (in line 6), in this case the logarithmic terms are replaced by polynomials "without memory". To our point of view, the complexity resulting from the "attaching" δ_i to Δ_B^{i-1} is better, despite the mentioned weakness (which, in fact, completely disappears in terms of $\tilde{O}$).

Next, note that if γ_i approaches the value of 0 this considerably increases the time needed for $Evaluate$ in order to estimate its value; to avoid this, we may introduce a lower bound for "acceptable" γ_i-s. B.t.w., the case of negative γ_i rises no difficulties (it is "turned into positive" by corresponding negative l_i).

Another consideration may useful as well in this connection. It seems like a very natural assumption that WL, while producing a weak hypothesis, is capable to report the received accuracy. In this case, the term of $\mathbf{E}_{0 \le i < T}\left[\gamma_i^{-2}\right]$ disappears from the complexity bounds: for instance, the time complexity would be bounded by

$$\tilde{O}\left(\frac{\varepsilon^{-1} \cdot Q[WL] + T[WL] + \varepsilon^{-2}}{\varepsilon^2 \cdot \mathbf{E}_{0 \le i < T}\left[\gamma_i^2\right]}\right).$$

Further, examples may be "reused" throughout the computation, both for WL teaching and for values estimation.[3] Of course, in this case the algorithm requires additional memory of size equal to the query complexity. Assuming that WL reports the accuracies, both storage and query complexity is bounded by

$$\tilde{O}\left(\varepsilon^{-1} \cdot Q[WL] + \varepsilon^{-2}\right).$$

Notice that our adaptive technique is aimed to minimize the number of performed iterations, from this point of view it is worth to make use of each received weak hypothesis, including those with small $|\gamma_i|$-s. On the other hand, slightly different approach should be used to minimize the overall time and/or query complexity: for example, consider, as described above, setting a lower bound for acceptable $|\gamma_i|$-s in order to avoid "too expensive" mean value estimations.

6. Optimality of $AdaFlat$ and $AdaFlat_{Filt}$

In this section we consider the smoothness parameter and the number of iterations performed by our algorithms.

[3] Note that this is still learning by filtering and no generalization error is introduced by such modification.

Claim 4 *The following holds for the algorithms $AdaFlat$ and $AdaFlat_{Filt}$:*

- *Their smoothness parameters are not higher than two times the minimum required for successful boosting.*
- *The asymptotic number of iterations they perform is optimal in terms of γ.*

Proof of Claim 4 The second part of the claim follows from a result by Freund (1995): The number T of iterations of any boosting scheme must satisfy

$$T = \Theta\left(\gamma^{-2} \cdot \ln \varepsilon^{-1}\right).$$

The fact that $\alpha \leq \varepsilon^{-1}$ is near-optimal up to the multiplicative factor of 2 becomes clear from the following argument: Consider a general boosting algorithm, suppose that it is not allowed to diverge from the original distribution D by more than

$$\varepsilon^{-1} \cdot (\frac{1}{2} - \frac{1}{2}\gamma) = \frac{1-\gamma}{2\varepsilon}. \tag{13}$$

Now suppose that the weak learner chooses some region $Y \subset X$ such that

$$D(Y) = \varepsilon,$$

and then produces the following hypothesis:

$$h_Y = \begin{cases} f(x) & if\ x \notin Y \\ -f(x) & otherwise \end{cases}.$$

Note that in this case the region Y should not be chosen randomly. We only require that the booster (which is supposed to be general, i.e., not restricted to a specific weak learner) has no information about Y and therefore must treat this region as a randomly chosen one.

As long as the target distribution for the weak learner has smoothness parameter not greater than (13), the weak learner may repeatedly return the hypothesis h_Y and still satisfy its specifications.

On the other hand, since form the booster's point of view the region Y is randomly chosen, the best thing the booster can do is to attach a list of polynomially many points from Y as a "correction" to h_Y. Since such a correction has, in general, only exponentially small impact, it can be neglected, and the final error may be assumed to be equal to $D(Y) = \varepsilon$.

The maximum smoothness parameter we have allowed is $(1-\gamma)/2\varepsilon$, and since γ decreases polynomially as the input size grows, the parameter can be made arbitrary close to $\varepsilon^{-1}/2$, as required.
$\blacksquare_{\textit{Claim 4}}$

7. Boosting Tandems

In this section we construct a boosting tandem, or combine two boosting algorithm in a kind of hierarchy. The upper level algorithm views the lower level algorithm as its weak learner, while the latter communicates with the "real" weak learner. While in the case of usual boosting the weak learner provides a polynomially-accurate weak hypothesis which is afterwards "amplified" by the booster into a polynomially-accurate strong hypothesis, in the boosting tandem model the corresponding evolution is **polynomial weak → constant correspondence → polynomial strong**.

This technique was first used by Freund (1992). Its advantage is that if one algorithm is more efficient in terms of γ and the other in terms of ε, this approach makes use of the "strong sizes" of the both, setting their "weak" parameters to constants.

Naturally, we are interested in preserving the adaptiveness and efficiency in terms of γ of $AdaFlat_{Filt}$, and it will be used as a low level.[4] In this case the smoothness factor of $AdaFlat_{Filt}$ will be bounded by a constant.

As a high level, we use an algorithm introduced in Freund (1992) (and used there for the same purpose). It performs $O(\ln(\varepsilon^{-1}) \cdot f(\gamma))$ iterations and its smoothness is $\tilde{O}(\varepsilon^{-1})$.

Putting everything together, we achieve the number of iterations (i.e., that of calls to the weak learner) bounded by

$$O\left(\frac{\ln(\varepsilon^{-1})}{\mathbf{E}_{0\leq i<T}\left[\gamma_i^2\right]}\right)$$

and smoothness of

$$\alpha = \tilde{O}(\varepsilon^{-1}).$$

Note that the resulting algorithm is adaptive. (The whole construction is similar to that made by Klivans and Servedio 1999; their result, however, is not adaptive.)

As mentioned in Section 6, this number of iterations corresponds to the lower bound. The price that we pay for the improvement is further complication of the final hypothesis structure,[5] and also a logarithmic in ε^{-1} growth of the smoothness parameter.[6] Notice that for the case of agnostic boosting considered in Section 4, bounding the smoothness strictly was shown to be critical.

An interesting application for the tandem is for near-uniform DNF learning with membership queries. The problem was solved for the first time by Jackson (1997); The most efficient solution known so far is that by Klivans and Servedio (1999), where they use a similar tandem for their construction. Our algorithm possesses the same complexity, and therefore the complexity of the solution equals that achieved by Klivans and Servedio (1999); moreover, our solution it is adaptive. The latter fact directly addresses an open question posed by Jackson (1997). Another attempt to use an adaptive algorithm in the context of near-uniform DNF learning was made by Bshouty and Gavinsky (2001); the result received there has weaker complexity bounds and it is adaptive only w.r.t. the size of the final hypothesis, but not w.r.t. the time complexity of the solution.

8. Other Applications and Further Work Directions

As mentioned before, in addition to the contexts of agnostic boosting and near-uniform DNF learning with membership queries, smoothness is critical for noise-tolerant learning (Freund, 1999, Domingo and Watanabe, 2000, Servedio, 2001), for learning via extended statistical queries (Bshouty and Feldman, 2001) and for agnostic learning (Ben-David, Long and Mansour, 2001).

Our algorithm can be used to solve all these tasks adaptively; the boosting tandem introduced in Section 7 achieves performance as efficient as that of other boosting algorithms known so far.

[4] The same approach works for $AdaFlat$ as well.

[5] The final hypothesis generated by the tandem is represented as a majority vote of weighted majority votes, instead of a single weighted majority vote, as generated by $AdaFlat$.

[6] Recall that for $AdaFlat$ and $AdaFlat_{Filt}$ it was strictly bounded by ε^{-1}.

We based our analysis of the application of *AdaFlat* to agnostic boosting upon the adaptiveness feature of the booster (if we would use a lower bound on γ_i-s instead, the achieved result would be noticeably weaker). An interesting open question is whether this adaptiveness feature can be similarly taken into consideration in the analysis of other smoothness dependent learning tasks, in particular, it would be interesting to gain some performance improvement for the widely studied task of DNF membership learning.

9. Acknowledgments

I would like to thank Nader Bshouty for his guidance and advice.

References

A. Blumer, A. Ehrenfeucht, D. Haussler and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the ACM 36(4), pp. 929-965*, 1989.

N. Bshouty and V. Feldman. On Using Extended Statistical Queries to Avoid Membership Queries. *Proceedings of the 14th Annual Conference on Computational Learning Theory, pp. 529-545*, 2001.

A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. *Proceedings of the 26th Symposium on Theory of Computing, pp. 253-262*, 1994.

N. Bshouty and D. Gavinsky. On Boosting with Optimal Poly-Bounded Distributions. *Proceedings of the 14th Annual Conference on Computational Learning Theory, pp. 490-506*, 2001.

N. Bshouty, J. Jackson and C. Tamon. More efficient PAC-learning of DNF with membership queries under the uniform distribution. *Proceedings of the 12th Annual Conference on Computational Learning Theory, pp. 286-295*, 1999.

S. Ben-David, P. M. Long and Y. Mansour. Agnostic Boosting. *Proceedings of the 14th Annual Conference on Computational Learning Theory, pp. 507-516*, 2001.

C. Domingo and O. Watanabe. MadaBoost: A modification of AdaBoost. *Proceedings of the 13th Annual Conference on Computational Learning Theory, pp. 180-189*, 2000.

Y. Freund. An improved boosting algorithm and its implications on learning complexity. *Proceedings of the 5th Annual Conference on Computational Learning Theory, pp. 391-398*, 1992.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation 121(2), pp. 256-285*, 1995.

Y. Freund. An adaptive version of the boost by majority algorithm. *Proceedings of the 12th Annual Conference on Computational Learning Theory, pp. 102-113*, 1999.

R. Impagliazzo. Hardcore Distributions for Somewhat Hard Problems. *Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 538-545*, 1995.

J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences 55(3), pp. 414-440*, 1997.

A. R. Klivans and R. A. Servedio. Boosting and Hard-Core Sets. *Proceedings of the 40th Annual Symposium on Foundations of Computer Science, pp. 624-633*, 1999.

M. J. Kearns, R. E. Schapire and L. M. Sellie. Towards Efficient Agnostic Learning. *Machine Learning 17, pp. 115-141*, 1994.

M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM 41(1), pp. 67-95*, 1994.

Y. Mansour. Learning Boolean Functions via the Fourier Transform. *Theoretical Advances in Neural Computing and Learning, Kluwe Academic Publishers,* , 1994.

R. E. Schapire. The strength of weak learnability. *Machine Learning 5(2), pp. 197-227*, 1990.

R. Servedio. Smooth Boosting and Learning with Malicious Noise. *Proceedings of the 14th Annual Conference on Computational Learning Theory, pp. 473-489*, 2001.

V. N. Vapnik. Estimation of Dependences Based on Empirical Data. *Springer,* , 1982.

L. Valiant. A theory of learnable. *Communications of the ACM 27(11), pp. 1134-1142*, 1984.