FilterBoost: Regression and Classification on Large Datasets

Joseph K. Bradley JKBRADLE@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University

Data Analysis Project
Joint work with Robert E. Schapire SCHAPIRE@CS.PRINCETON.EDU
Advisor: Carlos Guestrin GUESTRIN@CS.CMU.EDU

Abstract

We study boosting in the filtering setting, where the booster draws examples from an oracle instead of using a fixed training set and so may train efficiently on very large datasets. Our algorithm FilterBoost, which is based on a logistic regression technique proposed by Collins et al. (2002), requires fewer assumptions to achieve bounds equivalent to or better than previous work. Our proofs demonstrate the algorithm's strong theoretical properties for both classification and conditional probability estimation, and we validate these results through extensive experiments. Empirically, our algorithm proves more robust to noise and overfitting than batch boosters in conditional probability estimation and proves competitive in classification.

We give several extensions of FilterBoost to the multiclass case, proving PAC bounds on each. In particular, we make use of the ideas of pseudoloss and Error-Correcting Output Codes used by Freund and Schapire (1997) and Schapire (1997) to create easily implementable boosters, and we show that the generalization of Output Codes by Allwein et al. (2000) extends to FilterBoost. This work represents one of the first studies of boosting-by-filtering for multiclass problems.

Contents

1. Introduction

Boosting provides a ready method for improving existing learning algorithms for classification. Taking a weaker learner as input, boosters use the weak learner to generate weak hypotheses which are combined into a classification rule more accurate than the weak hypotheses themselves. Boosters such as AdaBoost (Freund and Schapire, 1997) have shown considerable success in practice.

Most boosters are designed for the *batch* setting where the learner trains on a fixed example set. This setting is reasonable for many applications, yet it requires collecting all examples before training. Moreover, most batch boosters maintain distributions over the entire training set, making their computation and storage requirements costly for very large datasets. To make boosting feasible on larger datasets, learners can be designed for the *filtering* setting. The batch setting provides the learner with a fixed training set, but the filtering setting provides an oracle which can produce an unlimited number of labeled examples, one at a time. This idealized model may describe learning problems with online example sources, including very large datasets which must be loaded piecemeal into memory. By using new training examples each round, filtering boosters avoid maintaining a distribution over a training set and so may use large datasets much more efficiently than batch boosters.

This paper presents FilterBoost, an adaptive boosting-by-filtering algorithm. We show it is applicable to both conditional probability estimation, where the learner predicts the *probability* of each label given an example, and classification. In Section 2, we describe the algorithm, after which we interpret it as a stepwise method for fitting an additive logistic regression model for conditional probabilities. We then bound the number of rounds and examples required to achieve any target error in $(0, 1)$. Previous filtering boosters have either worse bounds or require impractical assumptions in their analyses. We also show that FilterBoost can use the confidence-rated predictions from weak hypotheses described by Schapire and Singer (1999).

We extend the basic FilterBoost algorithm to several multiclass versions in Section 3. We follow the work of Freund and Schapire (1997) and present FilterBoost.M2, an algorithm analogous to their AdaBoost.M2. FilterBoost.OC is analogous to AdaBoost.OC (Schapire, 1997), using output codes to reduce a multiclass problem to multiple binary problems. We also show that the generalization of this output coding idea presented by Allwein et al. (2000) for batch learning extends readily to the filtering setting.

In Section 4, we give results from extensive experiments. For conditional probability estimation, we show that FilterBoost often outperforms batch boosters, which prove less robust to overfitting. For classification, we show that filtering boosters' efficiency on large datasets allows them to achieve higher accuracies faster than batch boosters in many cases.

1.1 Boosting

We begin with an explanation of traditional boosting in the batch setting.

Let X be the set of all examples and Y a discrete set of labels. For simplicity, assume X is countable, and consider only binary labels $Y = \{-1, +1\}$. We assume there exists an unknown target distribution D over labeled examples $(x, y) \in X \times Y$ from which both training and test examples are generated. The goal in classification is to choose a hypothesis

$h : X \to Y$ which minimizes the classification error $\Pr_D[h(x) \neq y]$, where the subscript indicates that the probability is with respect to (x, y) sampled randomly from D.

In the batch setting, a booster is given a fixed training set S and a weak learner which, given any distribution D_t over training examples S, is guaranteed to return a weak hypothesis $h_t : X \to \mathbb{R}$ such that the error $\epsilon_t \equiv \Pr_{D_t}[\text{sign}(h_t(x)) \neq y] < 1/2$. For T rounds t, the booster builds a distribution D_t over S, runs the weak learner on S and D_t, and receives h_t. The booster usually then estimates ϵ_t using S and weights h_t with $\alpha_t = \alpha_t(\epsilon_t)$. After T rounds, the booster outputs a final hypothesis H which is a linear combination of the weak hypotheses (e.g. $H(x) = \sum_t \alpha_t h_t(x)$). The sign of $H(x)$ indicates the predicted label $\hat{y}$ for x.

Two key elements of boosting are constructing D_t over S and weighting weak hypotheses. D_t is built such that misclassified examples receive higher weights than in D_{t-1}, eventually forcing the weak learner to classify previously poorly classified examples correctly. Weak hypotheses h_t are generally weighted such that hypotheses with lower errors receive higher weights.

1.2 Boosting-by-Filtering

We describe a general framework for boosting-by-filtering which includes most existing algorithms as well as our algorithm Filterboost. The filtering setting assumes the learner has access to an example oracle, allowing it to use entirely new examples sampled i.i.d. from D on each round. However, while maintaining the distribution D_t is straightforward in the batch setting, there is no fixed set S on which to define D_t in filtering. Instead, the booster simulates examples drawn from D_t by drawing examples from D via the oracle and reweighting them according to D_t. Filtering boosters generally accept each example (x, y) from the oracle for training on round t with probability proportional to the example's weight $D_t(x, y)$. The mechanism which accepts examples from the oracle with some probability is called the filter.

Thus, on each round, a boosting-by-filtering algorithm draws a set of examples from D_t via the filter, trains the weak learner on this set, and receives a weak hypothesis h_t. Though a batch booster would estimate ϵ_t using the fixed set S, filtering boosters may use new examples from the filter. Like batch boosters, filtering boosters may weight h_t using $\alpha_t = \alpha_t(\epsilon_t)$, and they output a linear combination of $h_1, \ldots, h_T$ as a final hypothesis. The filtering setting allows the learner to estimate the error of H_t to arbitrary precision by sampling from D via the oracle, so filtering boosters do this to decide when they have reached the target accuracy.

A key benefit of the filtering setting is that it can simplify the analysis of algorithms. The direct access to the target distribution via the oracle, rather than indirect access via a fixed dataset, allows analyses to directly consider the test error of the algorithm. Batch algorithms are generally analyzed by considering the training error and the generalization error separately.

Since boosting-by-filtering algorithms' analyses directly bound the test error, they indicate that the algorithms can achieve arbitrarily low test error as they run for more rounds (if the weak learning assumption continues to hold). This may seem to be a great benefit when compared with batch algorithms, which have generalization error bounds which grow with

the number of boosting rounds. However, both types of algorithms can achieve arbitrarily low test error if they use enough training examples. The difference is that the number of training examples must be decided a priori for batch boosters, while the number may be chosen adaptively for filtering boosters.

Both filtering and batch boosters require time proportional to $N \cdot T$, where N is the number of training examples used, to compute the distributions over examples. For filtering, each of T rounds requires taking $\mathrm{O}(T)$ time to compute the weight of each of the approximately N/T examples used on that round. The main benefits from filtering, therefore, are cheaper storage requirements —even one example at a time if the weak learner is an online algorithm— and the ability to adaptively choose the number of boosting rounds (and therefore the number of training examples).

1.3 Related Work

There have been a number of boosting-by-filtering algorithms previously proposed. The first polynomial-time booster, by Schapire (1990), was designed for filtering. Later filtering boosters included two more efficient ones proposed by Freund (1992, 1995), but both are non-adaptive, requiring a priori bounds on weak hypothesis error rates and combining weak hypotheses via unweighted majority votes. MadaBoost (Domingo and Watanabe, 2000) is competitive with AdaBoost empirically but theoretically requires weak hypotheses' error rates to be monotonically increasing, an assumption we found to be violated often in practice. Bshouty and Gavinsky (2002) proposed another boosting-by-filtering algorithm, but, like Freund's, their algorithm requires an a priori bound on weak hypothesis error rates. AdaFlat$_{filt}$ (Gavinsky, 2003) and GiniBoost (Hatano, 2006) do not have these limitations, but the former has worse bounds than other adaptive algorithms while the latter explicitly requires finite weak hypothesis spaces. We give a reference chart comparing these algorithms with ours in Figure 2.

Much work on boosting-by-filtering is based upon the idea of smooth boosting (Bshouty and Gavinsky, 2002; Gavinsky, 2003; Servedio, 2003; Hatano, 2006), in which boosting algorithms ensure that weights over training examples do not increase at an exponential rate. Intuitively, ignoring issues with normalizing weights, boosting algorithms which increase example weights exponentially quickly can create distributions with a great deal of weight on very few examples. The filter may then take a very long time to return an example, even when the test error is still high. In fact, Domingo and Watanabe (2000) proved that AdaBoost's weighting scheme increases example weights too quickly to be used in the filtering setting.

FilterBoost is based on a modification of AdaBoost by Collins et al. (2002) designed to minimize logistic loss. Their batch algorithm has yet to be shown to achieve arbitrarily low test error, but we use techniques inspired by MadaBoost to adapt the algorithm to the filtering setting and prove generalization bounds.

We postpone our discussion of related work on multiclass boosting until Section 3.

2. The FilterBoost Algorithm

The FilterBoost algorithm, given in Figure 1, is modeled after the aforementioned algorithm by Collins et al. (2002) and MadaBoost (Domingo and Watanabe, 2000). It is identical to

Define $F_t(x) \equiv \sum_{t'=1}^{t-1} \alpha_{t'} h_{t'}(x)$
Function $Oracle()$ returns labeled example (x, y) from target distribution $D(x, y)$

Algorithm 1: *FilterBoost*

Input: ε: target error rate, δ: confidence parameter, τ: edge estimate parameter, WL: weak learner which computes $h : X \longrightarrow \mathbb{R}$
Output: $H : X \longrightarrow Y$: final combined hypothesis

for $t = 1, 2, 3, \ldots$ **do**
- $\delta_t \longleftarrow \frac{\delta}{3t(t+1)}$
- Call `Filter(`t`, `ε`, `δ_t`)` to get m_t examples for training WL; get h_t
- $\hat{\gamma}'_t \longleftarrow$ `GetEdge(`t`, `ε`, `δ_t`, `τ`)`
- $\alpha_t \longleftarrow \frac{1}{2} \ln\left(\frac{1/2+\hat{\gamma}'_t}{1/2-\hat{\gamma}'_t}\right)$
- Define $H_t(x) = \text{sign}\Big(F_{t+1}(x)\Big)$

(The algorithm exits from the `Filter()` function.)

Algorithm 2: *GetEdge*

Input: t: boosting round, ε: target error rate,, δ_t: confidence parameter, τ: relative error allowed in edge estimates
Output: $\hat{\gamma}'_t$: corrected edge estimate

Use EGBStop to estimate $\gamma_t \equiv \frac{1}{2} - \text{E}\left[[\![\text{sign}(h_t(x)) \neq y]\!]\right]$ w.r.t. examples (x, y) drawn from `Filter(`t`, `ε`, `δ_t`)`, within relative error τ with probability at least $1 - \delta_t$; get estimate $\hat{\gamma}_t$.
return $\hat{\gamma}_t/(1 + \tau)$

Algorithm 3: *Filter*

Input: t: boosting round, ε: target error rate, δ_t: confidence parameter
Output: labeled example (x, y)

Define $r = \#$ calls to `Filter()` so far on round t
$\delta'_t \longleftarrow \frac{\delta_t}{r(r+1)}$
for $(i = 0;\ i < \frac{2}{\varepsilon} \ln(\frac{1}{\delta'_t});\ i = i + 1)$ **do**
- $(x, y) \longleftarrow$ `Oracle()`
- $q_t(x, y) \longleftarrow \frac{1}{1+e^{yF_t(x)}}$
- **return** (x, y) with probability $q_t(x, y)$

Exit FilterBoost and **return** H_{t-1}

Figure 1: The FilterBoost algorithm

the version of the algorithm presented in our initial paper (Bradley and Schapire, 2007), with the exception that the *GetEdge*() function uses the EGBStop algorithm (Mnih et al., 2008) instead of the Nonmonotonic Adaptive Sampling (NAS) algorithm (Watanabe, 2000; Domingo et al., 2002); this change is discussed more later.

Given an example oracle, weak learner, and target error rate $\varepsilon \in (0, 1)$, FilterBoost iteratively constructs the combined hypothesis $H_t(x) \equiv \sum_{t'=1}^{t-1} \alpha_{t'} h_{t'}(x)$ until it has error at most ε. The algorithm also takes a confidence parameter $\delta \in (0, 1)$ which upper-bounds the allowed probability of failure.

On round t, FilterBoost draws m_t examples from the filter to train the weak learner, which produces weak hypothesis h_t. The size of m_t must be large enough to ensure h_t has error $\epsilon_t < 1/2$ with high probability (at least $1 - \delta_t$), where $\epsilon_t \equiv \mathrm{E}[[\![\mathrm{sign}(h_t(x)) \neq y]\!]]$. (We write $[\![\pi]\!]$ to denote an indicator function with value 0 if the statement π is false and 1 if π is true.) This value m_t may depend upon the weak learner or problem setting. The edge of h_t is $\gamma_t = 1/2 - \epsilon_t$, and this edge is estimated by the function $GetEdge()$ and is used to set h_t's weight α_t.

Function $GetEdge()$ uses the EGBStop algorithm (Mnih et al., 2008). EGBStop draws an adaptively chosen number of examples from the filter and returns an estimate $\hat{\gamma}_t$ of the edge of h_t within relative error τ of the true edge γ_t with high probability. The $GetEdge()$ function revises this estimate as $\hat{\gamma}'_t = \hat{\gamma}_t/(1+\tau)$.

The $Filter()$ function generates examples (x, y) from D_t by repeatedly drawing (x, y) from the oracle, calculating the weight $q_t(x, y) \propto D_t(x, y)$, and accepting (x, y) with probability $q_t(x, y)$. The algorithm exits from the $Filter()$ function.

2.1 Analysis: Conditional Probability Estimation

We begin our analysis of FilterBoost by interpreting it as an additive model for logistic regression, for this interpretation will later aid in the analysis for classification. Such models take the form

$$\log \frac{\Pr[y=1|x]}{\Pr[y=-1|x]} = \sum_t f_t(x) = F(x), \qquad \text{which implies} \qquad \Pr[y=1|x] = \frac{1}{1+e^{-F(x)}}$$

where, for FilterBoost, $f_t(x) = \alpha_t h_t(x)$. Dropping subscripts, we can write the expected negative log likelihood of example (x, y) after round t as

$$\pi(F_t + \alpha_t h_t) = \pi(F + \alpha h) = \mathrm{E}\left[-\ln \frac{1}{1+e^{-y(F(x)+\alpha h(x))}}\right] = \mathrm{E}\left[\ln\left(1+e^{-y(F(x)+\alpha h(x))}\right)\right].$$

Taking a similar approach to the analysis of Friedman et al. (2000), we show in the following theorem that FilterBoost performs an approximate stepwise minimization of this negative log likelihood. The proof is in the Appendix.

Theorem 1 *Define the expected negative log likelihood $\pi(F + \alpha h)$ as above. Given F, FilterBoost chooses h to minimize a second-order Taylor expansion of π around $h = 0$. Given this h, it then chooses α to minimize an upper bound of π.*

The batch booster given by Collins et al. (2002) which FilterBoost is based upon is guaranteed to converge to the minimum of this objective when working over a finite sample. Note that FilterBoost uses weak learners which are simple classifiers to perform regression. AdaBoost too may be interpreted as an additive logistic regression model of the form $\Pr[y = 1|x] = \frac{1}{1+e^{-2F(x)}}$ with $\mathrm{E}[\exp(-yF(x))]$ as the optimization objective (Friedman et al., 2000).

2.2 Analysis: Classification

In this section, we interpret FilterBoost as a traditional boosting algorithm for classification and prove bounds on its generalization error. We first give a theorem relating err_t, the error

rate of H_t over the target distribution D, to p_t, the probability with which the filter accepts a random example generated by the oracle on round t.

Theorem 2 *Let $err_t = \Pr_D[H_t(x) \neq y]$, and let $p_t = \mathrm{E}_D[q_t(x,y)]$. Then $err_t \leq 2p_t$.*

Proof:

$$\begin{aligned} err_t &= \Pr_D[H_t(x) \neq y] = \Pr_D[yF_{t-1}(x) \leq 0] \\ &= \Pr_D[q_t(x,y) \geq 1/2] \leq 2 \cdot \mathrm{E}_D[q_t(x,y)] \quad \text{(using Markov's inequality)} \\ &= 2p_t \qquad \blacksquare \end{aligned}$$

We next use the expected negative log likelihood π from Section 2.1 as an auxiliary function to aid in bounding the required number of boosting rounds. Viewing π as a function of the boosting round t, we can write $\pi_t = -\sum_{(x,y)} D(x,y)\ln(1-q_t(x,y))$. Our goal is then to minimize π_t, and the following lemma captures the learner's progress in terms of the decrease in π_t on each round. This lemma assumes edge estimates returned by *GetEdge*() are exact, i.e. $\hat{\gamma}'_t = \gamma_t$, which leads to a simpler bound on T in Theorem 4. We then consider the error in edge estimates and give a revised bound in Lemma 6 and Theorem 7. The proofs of Lemmas 3 and 6 are in the Appendix.

Lemma 3 *Assume for all rounds t that the edge $\gamma_t \neq 0$ and γ_t is estimated exactly. Let $\pi_t = -\sum_{(x,y)} D(x,y)\ln(1-q_t(x,y))$. Then*

$$\pi_t - \pi_{t+1} \geq p_t\left(1 - 2\sqrt{1/4 - \gamma_t^2}\right).$$

Combining Theorem 2, which bounds the error of the current combined hypothesis in terms of p_t, with Lemma 3 gives the following upper bound on the required rounds T.

Theorem 4 *Let $\gamma = \min_t |\gamma_t|$, and let ε be the target error. Given Lemma 3's assumptions, if FilterBoost runs for*

$$T > \frac{2\ln(2)}{\varepsilon\left(1 - 2\sqrt{1/4 - \gamma^2}\right)}$$

rounds, then $err_t < \varepsilon$ for some t, $1 \leq t \leq T$. In particular, this is true for $T > \frac{\ln(2)}{2\varepsilon\gamma^2}$.

Proof. For all (x,y), since $F_1(x,y) = 0$, then $q_1(x,y) = 1/2$ and $\pi_1 = \ln(2)$. Now, suppose $err_t \geq \varepsilon, \forall t \in \{1,...,T\}$. Then, from Theorem 2, $p_t \geq \varepsilon/2$, so Lemma 3 gives

$$\pi_t - \pi_{t+1} \geq \frac{1}{2}\varepsilon\left(1 - 2\sqrt{1/4 - \gamma^2}\right)$$

Unraveling this recursion as $\sum_{t=1}^{T}(\pi_t - \pi_{t+1}) = \pi_1 - \pi_{T+1} \leq \pi_1$ gives

$$T \leq \frac{2\ln(2)}{\varepsilon\left(1 - 2\sqrt{1/4 - \gamma^2}\right)}.$$

So, $err_t \geq \varepsilon, \forall t \in \{1, ..., T\}$ is contradicted if T exceeds the theorem's lower bound. The simplified bound follows from the first bound via the inequality $1-\sqrt{1-x} \leq x$ for $x \in [0,1]$. ■

Theorem 4 shows FilterBoost can reduce generalization error to any $\varepsilon \in (0,1)$, but we have thus far overlooked the probabilities of failure introduced by three steps: training the weak learner, deciding when to stop boosting, and estimating edges. We bound the probability of each of these steps failing on round t with a confidence parameter $\delta_t = \frac{\delta}{3t(t+1)}$ so that a simple union bound ensures the probability of some step failing to be at most FilterBoost's confidence parameter δ. Finally, we revise Lemma 3 and Theorem 4 to account for error in estimating edges.

Training the weak learner: The number m_t of examples the weak learner trains on must be large enough to ensure weak hypothesis h_t has a non-zero edge. It should be set according to the choice of weak learner and will depend upon the confidence parameter δ_t.

Deciding when to stop: To decide when to stop boosting (i.e. when $err_t \leq \varepsilon$), we can use Theorem 2, which upper-bounds the error of the current combined hypothesis H_t in terms of the probability p_t that $Filter()$ accepts a random example from the oracle. If the filter rejects enough examples in a single call, we know p_t is small, so H_t is accurate enough. Theorem 5 formalizes this intuition.

Theorem 5 *In a single call to $Filter(t)$, if n examples have been rejected, where $n \geq \frac{2}{\varepsilon}\ln(1/\delta'_t)$, then $err_t \leq \varepsilon$ with probability at least $1-\delta'_t$.*

Proof. Suppose $p_t > \varepsilon/2$. Then the probability that the filter rejects n sequential examples is $(1-p_t)^n < (1-\varepsilon/2)^n$. So, if $(1-\varepsilon/2)^n \leq \delta'_t$, then $p_t \leq \varepsilon/2$ with probability at least $1-\delta'_t$. From Theorem 2, we know $p_t \leq \varepsilon/2$ implies $err_t \leq \varepsilon$. The condition $(1-\varepsilon/2)^n \leq \delta'_t$ gives our bound on n to ensure $err_t \leq \varepsilon$ with high probability. ■

Theorem 5 provides a stopping condition which is checked on each call to $Filter()$. Each check may fail with probability at most $\delta'_t = \frac{\delta_t}{r(r+1)}$ on the r^{th} call to $Filter()$ so that a union bound ensures FilterBoost stops prematurely on round t with probability at most δ_t. Theorem 5 uses a similar argument to that used for MadaBoost, giving similar stopping criteria for both algorithms.

Estimating weak hypothesis edges: We estimate weak hypotheses' edges γ_t using the EGBStop algorithm by Mnih et al. (2008). The EGBStop algorithm is guaranteed, with probability at least $1-\delta_t$, to compute an estimate of the mean of a bounded random variable X within relative error τ of the truth using at most

$$C \cdot \max\left\{\frac{\sigma^2}{\tau^2\mu^2}, \frac{R}{\tau|\mu|}\right\}\left(\log\frac{1}{\delta_t} + \log\log\frac{R}{\tau|\mu|}\right)$$

samples, where C is a constant, σ^2 is the variance of X, μ is the mean of X, and R is the range of X. In our case, this means that with probability at least $1-\delta_t$, we can compute an estimate $\hat{\gamma}_t$ of the true edge γ_t within relative error $\tau \in (0,1)$ using at most

$$C \cdot \max\left\{\frac{\sigma^2}{\tau^2\gamma_t^2}, \frac{1}{\tau|\gamma_t|}\right\}\left(\log\frac{1}{\delta_t} + \log\log\frac{1}{\tau|\gamma_t|}\right)$$

filtered samples, where C is a constant and σ^2 is the variance of γ_t. As Mnih et al. state, their bound is very close to optimal, for it is close to a lower bound on the number of

samples needed by an adaptive sampling algorithm of this form proven by Dagum et al. (2000). Dagum et al. proved that, to estimate the mean of nonnegative, bounded X within relative error τ, at least

$$C' \cdot \max\left\{\frac{\sigma^2}{\tau^2\mu^2}, \frac{1}{\tau\mu}\right\} \cdot \log\frac{2}{\delta_t}$$

samples will be needed with probability at least $1-\delta_t$, where C' is a constant. The EGBStop algorithm's bound is worse by the presence of the range R and the log log term; in our case, $R = 1$ (for edges), and the log log term is generally very small.

The original FilterBoost algorithm (Bradley and Schapire, 2007) used the Nonmonotonic Adaptive Sampling (NAS) algorithm (Watanabe, 2000; Domingo et al., 2002) instead of EGBStop. To compute $\hat{\gamma}_t$, the NAS algorithm uses at most $\frac{2(1+2\tau)^2}{(\tau\gamma_t)^2}\ln(\frac{1}{\tau\gamma_t\delta_t})$ filtered examples. This bound is, however, generally worse than the bound for EGBStop. (Note that if the booster has an a priori lower bound on all γ_t, the number examples may be chosen using a Hoeffding-style bound, as is done by Bshouty and Gavinsky (2002).)

FilterBoost conservatively revises the edge estimate by a factor of $1/(1+\tau)$ to help in the following analysis.

A complete bound: With this guarantee on edge estimates from EGBStop, we can rewrite Lemma 3 as follows:

Lemma 6 *Assume for all t that $\gamma_t \neq 0$ and γ_t is estimated to within $\tau \in (0,1)$ relative error. Let $\pi_t = -\sum_{(x,y)} D(x,y)\ln(1-q_t(x,y))$. Then*

$$\pi_t - \pi_{t+1} \geq p_t\left(1 - 2\sqrt{1/4 - \gamma_t^2\left(\frac{1-\tau}{1+\tau}\right)^2}\right).$$

Using Lemma 6, the proof of which is in the Appendix, the following theorem modifies Theorem 4 to account for error in edge estimates.

Theorem 7 *Let $\gamma = \min_t|\gamma_t|$. Let ε be the target error. Given Lemma 6's assumptions, if FilterBoost runs for*

$$T > \frac{2\ln(2)}{\varepsilon\left(1 - 2\sqrt{1/4 - \gamma^2(\frac{1-\tau}{1+\tau})^2}\right)}$$

rounds, then $err_t < \varepsilon$ for some t, $1 \leq t \leq T$.

When comparing FilterBoost with batch boosters, note that the FilterBoost bounds' dependence on the weak hypothesis space complexity is hidden in m_t, while for it is explicit for batch algorithms.

An alternate bound for FilterBoost may be derived using techniques from Shalev-Shwartz and Singer (2006). They use the framework of convex repeated games to define a general method for bounding the performance of online and boosting algorithms. For FilterBoost, their techniques, combined with Theorem 2, give a bound similar to that in Theorem 4 but proportional to ε^{-2} instead of ε^{-1}.

	Setting	Rounds required (Big-Oh)	Need a priori bound γ	Assume edges decreasing	Can use infinite weak hypothesis spaces
FilterBoost	filtering	$1/\varepsilon$	No	No	Yes
MadaBoost	filtering	$1/\varepsilon$	No	Yes	Yes
Bshouty and Gavinsky (2002)	filtering	$\log(1/\varepsilon)$	Yes	No	Yes
AdaFlat$_{filt}$	filtering	$1/\varepsilon^2$	No	No	Yes
GiniBoost	filtering	$1/\varepsilon$	No	No	No
AdaBoost	batch	$\log(1/\varepsilon)$	No	No	Yes

Figure 2: A comparison of FilterBoost with previous boosting algorithms. Note that all of the filtering boosters may use boosting tandems to achieve $O(\log(1/\varepsilon))$ rounds, but this results in more complicated algorithms and final hypotheses.

2.3 Comparison with Previous Algorithms

The bounds from Theorems 4 and 7 show FilterBoost requires at most $\mathrm{O}(\varepsilon^{-1}\gamma^{-2})$ boosting rounds. MadaBoost (Domingo and Watanabe, 2000), which we test in our experiments, resembles FilterBoost but uses truncated exponential weights $q_t(x,y) = \min\{1, \exp(yF_{t-1}(x))\}$ instead of the logistic weights $q_t(x,y) = (1+\exp(yF_t(x)))^{-1}$ used by FilterBoost. The algorithms' analyses differ, with MadaBoost requiring the edges γ_t to be monotonically decreasing, but both lead to similar bounds on the number of rounds T proportional to ε^{-1}.

The non-adaptive filtering boosters of Freund (1992, 1995) and of Bshouty and Gavinsky (2002) and the batch booster AdaBoost (Freund and Schapire, 1997) have smaller bounds on T, proportional to $\log(\varepsilon^{-1})$. However, we can use boosting tandems, a technique used by Freund (1992) and Gavinsky (2003), to create a filtering booster with T bounded by $\mathrm{O}(\log(\varepsilon^{-1})\gamma^{-2})$. Specifically, we can use FilterBoost to boost the accuracy of the weak learner to some constant and, in turn, treat FilterBoost as a weak learner and use a boosting algorithm from Freund (1992) to achieve any target error. This technique does not require FilterBoost to make additional assumptions, and it turns FilterBoost into an adaptive booster with a bound on T proportional to $\log(\varepsilon^{-1})$. Note, however, that boosting tandems result in more complicated final hypotheses. To our knowledge, no adaptive boosting-by-filtering algorithm (which does not require a priori bounds on weak hypothesis edges) achieves a bound on T proportional to $\log(\varepsilon^{-1})$ without boosting tandems.

We compare FilterBoost with AdaBoost and recent boosting-by-filtering algorithms in Figure 2. These include AdaFlat$_{filt}$ (Gavinsky, 2003) and GiniBoost (Hatano, 2006). As discussed in Section 1.3, FilterBoost avoids unrealistic assumptions made by previous boosting-by-filtering algorithms, save for AdaFlat$_{filt}$, which has a worse bound on the number of boosting rounds required. Note that all of these algorithms require a number of rounds proportional to $1/\gamma^2$.

2.4 Confidence-Rated Predictions

Schapire and Singer (1999) show AdaBoost benefits from confidence-rated predictions, where weak hypotheses return predictions whose magnitudes indicate confidence. These values are chosen to greedily minimize AdaBoost's exponential loss function over training data, and this aggressive weighting can result in faster learning. FilterBoost may use confidence-rated predictions in an identical manner. In the proof of Lemma 3, the decrease in the negative log likelihood π_t of the data (relative to H_t and the target distribution D) is lower-bounded by $p_t - p_t \sum_{(x,y)} D_t(x,y) e^{-\alpha_t y h_t(x)}$. Since p_t is fixed, maximizing this bound is equivalent to minimizing the exponential loss over D_t.

Schapire and Singer (1999) show that, for a fixed training set $\{(x_i, y_i)\}$, this bound is optimized when

$$\begin{aligned} \alpha_t h_t(x) &= \tfrac{1}{2} \ln \left(W_+^{\text{sign}(h_t(x))} / W_-^{\text{sign}(h_t(x))} \right) \\ \text{where } W_y^{\hat{y}} &= \sum_{i:\text{sign}(h_t(x_i))=\hat{y} \wedge y_i = y} D_t(x_i, y_i) \end{aligned}$$

Schapire & Singer recommend smoothing these predictions by adding a small constant to each $W_y^{\hat{y}}$. We add 10^{-6} in our experiments with confidence-rated predictions.

2.5 Implementation Details

Vanilla FilterBoost accepts examples (x, y) from the oracle with probability $q_t(x, y)$, but it may instead accept all examples and weight each with $q_t(x, y)$. This is analogous to using importance weighting instead of importance sampling. Weighting instead of filtering examples increases accuracy but also increases the size of the training set passed to the weak learner. For estimating edges γ_t, however, weighting takes almost the same amount of time as filtering. For efficiency, we recommend filtering when training the weak learner but weighting when estimating edges.

In practice, it is best to modify FilterBoost's *GetEdge*() function for efficiency. The EGBStop algorithm used to estimate edges γ_t uses many examples, and using several orders of magnitude fewer sacrifices little accuracy. The same is true for MadaBoost, which uses the NAS algorithm but may use EGBStop instead. We recommend using $C_n \log(t+1)$ examples to estimate γ_t, where C_n is a constant and the log factor scales the number approximately as EGBStop would. In our experiments, we train weak learners with $C_m \log(t+1)$ examples as well, for PAC sample complexity bounds generally scale proportional to $\log(1/\delta_t) \propto \log(t)$. In our experiments, we use $C_n = C_m = 300$. A more rigorous but less practical approach would use the sample complexity bounds for the weak learners.

These modifications mean τ (error in edge estimates) and δ (confidence) have no effect on our tests. If FilterBoost were run without the modifications, $\tau \in (0, 1)$ could be chosen by hand or model selection. Intuitively, large τ makes the algorithm tend to choose α_t poorly, increasing the number of required boosting rounds. With very low τ, the *GetEdge*() function requires many examples. We could choose τ to optimize the example bounds from Section 2.2. We performed a numerical analysis comparing, for varying minimum weak learner edge $\gamma = \min_t \gamma_t$, a simplified, convex bound on the example complexity of FilterBoost for constant τ versus the numerically computed optimal choice of $\tau = \tau(\gamma)$. We found that, for $\gamma \in (.01, .49)$, setting $\tau = 1/5$ leads to an example complexity bound at most 12% higher than for the optimal τ, and we suggest using this τ in practice.

	Problem	Weak hypotheses predict	Requirements for h
AdaBoost.M1	multiclass	1 of k labels	error $< 1/2$ for k-class problem
AdaBoost.M2	multiclass	confidence in each label	pseudoloss $< 1/2$
AdaBoost.OC	multiclass	binary label	error $< 1/2$ on binary relabeling
AdaBoost.MH	multiclass, multilabel	confidence in each label	error $< 1/2$, for each $y \in Y$
AdaBoost.MO	multiclass	confidence in each label	error $< 1/2$, for each y' in relabeling Y'
AdaBoost.MR	multiclass, multilabel	confidence in each label	h scores correct labels above incorrect, on avg.

Figure 3: A comparison of various methods for extending AdaBoost to multiclass and multilabel problems.

FilterBoost assumes that the oracle generates examples from a fixed distribution. To simulate an oracle from a fixed dataset, one may randomly permute the data and use examples in the new order. In practice, filtering boosters can achieve higher accuracy by cycling through training sets again instead of stopping once examples are depleted; we call this method "recycling" examples.

3. Multiclass and Multilabel Classification

While most boosting algorithms are proposed in terms of binary classification, almost all may be extended to multiclass and multilabel classification in similar ways. We briefly review some of the methods used for extending AdaBoost to multiclass and multilabel problems before describing analogous extensions for FilterBoost. In the following, we let $k = |Y|$, the number of classes or labels.

Figure 3 compares some multiclass and multilabel extensions to AdaBoost. AdaBoost.M1 and AdaBoost.M2 were proposed in the original AdaBoost paper (Freund and Schapire, 1997). AdaBoost.M1 makes the rather strong assumption that weak hypotheses can choose the correct label out of k labels with probability $> 1/2$, i.e. significantly better than random guessing. AdaBoost.M2 is more realistic, allowing the booster to make use of weak hypotheses which do only slightly better than random guessing. Dietterich and Bakiri (1995) proposed the use of using Error-Correcting Output Codes (ECOC) to reduce a multiclass problem to a set of binary problems, and AdaBoost.OC (Schapire, 1997) combines this idea with AdaBoost to create a simple algorithm for multiclass problems. AdaBoost.MH, AdaBoost.MO, and AdaBoost.MR were proposed by Schapire and Singer (1999). AdaBoost.MH solves multilabel problems by reducing the problem of mapping X to subsets of Y to the problem of mapping $X \times Y$ to binary labels, thereby minimizing Hamming loss. It may also be applied to multiclass problems if the weak learner's predictions are confidence-rated; in this case, it outputs the label predicted with the highest confidence. AdaBoost.MO uses ECOC to create a different reduction from multiclass to binary; it relabels examples with binary labels and then calls AdaBoost.MH on the relabeled examples. AdaBoost.MR solves multiclass and multilabel problems by trying to minimize a ranking loss which penalizes

examples for pairs of labels in which the higher-ranked label is not a true label while the lower-ranked label is a true label. Allwein et al. (2000) gives a general method for reducing multiclass problems to binary for margin-based classifiers such as boosters and Support Vector Machines (SVMs) (Vapnik, 1995; Cortes and Vapnik, 1995). Their method generalizes many other approaches by using a generalization of the coding matrix method used by Dietterich and Bakiri (1995).

3.1 FilterBoost.M2

We first modify FilterBoost for multiclass problems following the example of AdaBoost.M2. The resulting algorithm, FilterBoost.M2, is given in Figure 4. This algorithm uses a natural extension to the multiclass case which allows the use of real-valued weak hypotheses; these weak hypotheses may either model all classes at once or consider each separately as long as they do better than random guessing with respect to a pseudoloss. Moreover, the algorithm permits a simple proof via a reduction to FilterBoost, and it allows a simple proof for FilterBoost.OC, described in the following section.

For FilterBoost.M2, we use weak hypotheses which output values in $[-1, 1]$ for any example-label pair (x, y), where higher values predicted by a weak hypothesis indicate higher confidence that example x should have label y. (Outputing values in $[-1, 1]$ rather than $\mathbb{R}$ does not affect the generality of our results.) The final hypothesis combines the outputs of weak hypotheses in a weighted sum, outputting the label which maximizes this sum. We use notation parallel to that used by FilterBoost, but note that the variables are defined in analogous but distinct ways; for example, p_t is still the probability an example from the oracle is accepted by the filter on round t, but it is defined differently here than in FilterBoost.

While FilterBoost implicitly maintains a single weight $q_t(x, y)$ for each example, FilterBoost.M2 has a weight for each pair $((x, y), y')$ where (x, y) is a labeled example and y' is any label. This weight $q'_t((x, y), y')$, defined in Figure 4, is close to 1 when the current combined hypothesis H_{t-1} strongly prefers an incorrect label y' to the true label y, is close to 0 when H_{t-1} strongly prefers y to y', and is $1/2$ when H_{t-1} has no preference.

The weak hypotheses' error rates (and edges) are no longer defined in terms of binary error but, rather, are defined using a pseudoloss:

$$\text{ploss}_{q_t}(h_t, (x, y)) \equiv \frac{1}{2}\left(1 - h_t(x, y) + \sum_{y' \neq y} q_t((x, y), y') h_t(x, y')\right)$$

where $q_t((x, y), y')$ is defined as in Figure 4. When h_t predicts the correct label y with high confidence and the incorrect labels y' with low confidence, this pseudoloss is near 0; when h_t predicts y with much lower confidence than y', it is near 1. The error is the expectation of this pseudoloss with respect to samples drawn from the filter.

We first adapt Theorem 2 to FilterBoost.M2. The resulting Theorem 8 is similar to that for FilterBoost but is worse by a factor of $1/(k-1)$. We may slightly modify Theorem 5 (using the analogous FilterBoost.M2 variables) to get a similar stopping condition to that for FilterBoost in Theorem 9.

Theorem 8 *Let* $err_t = \Pr_D[H_t(x) \neq y]$, *and let* $p_t = \mathrm{E}_D[p'_t(x, y)]$. *Then* $err_t \leq \frac{2p_t}{k-1}$.

Define $q'_t((x,y),y') \equiv 1/(1+\exp(\sum_{t'=1}^{t-1} \frac{1}{2}\alpha_{t'}(h_{t'}(x,y) - h_{t'}(x,y'))))$
Define $p'_t(x,y) \equiv \sum_{y' \neq y} q'_t((x,y),y')$
Define $q_t((x,y),y') \equiv q'_t((x,y),y')/p'_t(x,y)$
Function $Oracle()$ returns labeled example (x,y) from target distribution $D(x,y)$

Algorithm 4: *FilterBoost.M2*

Input: ε: target error rate, δ: confidence parameter, τ: edge estimate parameter, WL: weak learner which computes $h : X \times Y \longrightarrow [-1,1]$ where $|Y| = k$
Output: $H : X \longrightarrow Y$: final combined hypothesis

for $t = 1, 2, 3, \ldots$ **do**
  $\delta_t \longleftarrow \frac{\delta}{3t(t+1)}$
  Call `Filter(`t, ε, δ_t`)` to get m_t examples for training WL; get h_t
  $\hat{\gamma}'_t \longleftarrow$ `GetEdge(`t, ε, δ_t, τ`)`
  $\alpha_t \longleftarrow \frac{1}{2}\ln\left(\frac{1/2+\hat{\gamma}'_t}{1/2-\hat{\gamma}'_t}\right)$
  Define $H_t(x) = \arg\max_{y \in Y} \sum_{t'=1}^{t} \alpha_t h_t(x,y)$
(The algorithm exits from the `Filter()` function.)

Algorithm 5: *GetEdge*

Input: t: boosting round, ε: target error rate,, δ_t: confidence parameter, τ: relative error allowed in edge estimates
Output: $\hat{\gamma}'_t$: corrected pseudoedge estimate

Use EGBStop to estimate $\gamma_t \equiv \mathrm{E}\left[\frac{1}{2}\left(h_t(x,y) - \sum_{y' \neq y} q_t((x,y),y')h_t(x,y')\right)\right]$ w.r.t. examples (x,y) drawn from `Filter(`t, ε, δ_t`)`, within relative error τ with probability at least $1-\delta_t$; get estimate $\hat{\gamma}_t$.
return $\hat{\gamma}/(1+\tau)$

Algorithm 6: *Filter*

Input: t: boosting round, ε: target error rate, δ_t: confidence parameter
Output: labeled example (x,y)

Define $r = \#$ calls to `Filter()` so far on round t
$\delta'_t \longleftarrow \frac{\delta_t}{r(r+1)}$
for $(i = 0;\ i < \frac{2}{(k-1)\varepsilon}\ln(\frac{1}{\delta'_t});\ i = i+1)$ **do**
  $(x,y) \longleftarrow$ `Oracle()`
  return (x,y) with probability $p'_t(x,y)/(k-1)$
Exit FilterBoost.M2 and **return** H_{t-1}

Figure 4: FilterBoost.M2: multiclass FilterBoost based on AdaBoost.M2

Proof:

$$\begin{aligned}
err_t &= \Pr_D\left[H_t(x) \neq y\right] \\
&= \Pr_D\left[\exists y' \neq y : \sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x,y) \leq \sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x,y')\right] \\
&= \Pr_D\left[\exists y' \neq y : q'_t((x,y),y') \geq 1/2\right] \\
&\leq \Pr_D\left[\sum_{y' \neq y} q'_t((x,y),y') \geq 1/2\right] \\
&\leq 2\mathrm{E}_D\left[\sum_{y' \neq y} q'_t((x,y),y')\right] \\
&= \frac{2p_t}{k-1} \qquad \blacksquare
\end{aligned}$$

Theorem 9 *In a single call to Filter(t), if n examples have been rejected, where* $n \geq \frac{2}{(k-1)\varepsilon}\ln(1/\delta_t')$*, then* $err_t \leq \varepsilon$ *with probability at least* $1-\delta_t'$*.*

Following the analysis of AdaBoost.M2 by Freund and Schapire (1997), we can derive the following theorem which bounds the number of rounds of boosting required to achieve a given target error. Note that this error is not a pseudoloss but classification error $\Pr_D[H_t(x) \neq y]$. The proof of Theorem 10, which uses a reduction of FilterBoost.M2 to FilterBoost, is in the Appendix.

Theorem 10 *Let* $\gamma = \min_t |\gamma_t|$*. Let* ε *be the target error. Given Lemma 6's assumptions, if FilterBoost.M2 runs*

$$T > \frac{2(k-1)\ln(2)}{\varepsilon\left(1-2\sqrt{1/4-\gamma^2(\frac{1-\tau}{1+\tau})^2}\right)}$$

rounds, then $err_t < \varepsilon$ *for some* t*,* $1 \leq t \leq T$*.*

We have now proven that FilterBoost.M2 can boost a weak learner, achieving arbitrarily high accuracy as long as the weak learner can do better than random guessing on any distribution it is given. This algorithm allows the use of weak learners which consider all possible labels at once, rather than dividing them into k separate problems. While this flexibility can permit more powerful weak learners, it is often simpler to implement binary weak learners like those used by FilterBoost. We now introduce another extension of FilterBoost which solves multiclass problems using binary weak learners.

3.2 FilterBoost.OC

We propose FilterBoost.OC (Output Codes), a multiclass extension to FilterBoost, which is analogous to AdaBoost.OC, a multiclass extension to AdaBoost proposed by Schapire (1997). It is based upon the idea of using Error-Correcting Output Codes to reduce multiclass problems to binary, as proposed by Dietterich and Bakiri (1995), allowing the booster to use binary weak learners. As Schapire (1997) shows empirically, though this reduction to binary problems generally leads to less gain in accuracy per boosting round, the efficiency of the method leads to more gain in accuracy per unit of time. Our algorithm and arguments reducing FilterBoost.OC to a special case of FilterBoost.M2 are essentially identical to those used by Schapire (1997), simply translated into the filtering setting.

The FilterBoost.OC algorithm is given in Figure 5. Each boosting round, it chooses a coloring μ_t which maps labels in Y to binary labels in $\{-1, 1\}$. To train the weak learner, it draws examples from the oracle, relabels them using this coloring, and passes these binary-labeled examples to the weak learner. Intuitively, this means that the weak hypotheses are chosen to divide examples into two sets, where each set corresponds to some subset of Y. Therefore, a weak hypothesis $h_t : X \longrightarrow \{-1, 1\}$ defines a hypothesis $\tilde{h}_t : X \longrightarrow 2^Y$, where 2^Y denotes all subsets of Y; as defined in Figure 5, $\tilde{h}_t$ predicts y as a label of x if and only if h_t predicts the correct binary label $\mu_t(y)$ of x. Thus, we are effectively choosing a weak hypothesis $\tilde{h}_t : X \times Y \longrightarrow \{-1, 1\}$ (where $\tilde{h}_t(x, y) = -1$ indicates that x does not have label y and 1 indicates that x does have label y). This mapping to $\{-1, 1\}$ instead of $[-1, 1]$ is a special case of the form for weak hypotheses used by FilterBoost.M2. Viewing

Define $\tilde{h}_t(x) \equiv \{y \in Y : h_t(x) = \mu_t(y)\}$

Define $q'_t((x,y),y') \equiv 1/\left(1+\exp\left(\sum_{t'=1}^{t-1}\alpha_{t'}\left([\![y \in \tilde{h}_t(x)]\!]-[\![y' \in \tilde{h}_t(x)]\!]\right)\right)\right)$

Function $Oracle()$ returns labeled example (x,y) from target distribution $D(x,y)$

Algorithm 7: *FilterBoost.OC*

Input: ε: target error rate, δ: confidence parameter, τ: edge estimate parameter, WL: weak learner which computes $h : X \longrightarrow \{-1,1\}$

Output: $H : X \longrightarrow Y$: final combined hypothesis

for $t = 1,2,3,\ldots$ **do**
- $\delta_t \longleftarrow \frac{\delta}{3t(t+1)}$
- Choose $\mu_t : Y \longrightarrow \{-1,1\}$
- Call `FilterWL(`t, ε, δ_t, μ_t`)` to get m_t examples for WL; relabel using μ_t; train WL to get h_t
- $\hat{\tilde{\gamma}}'_t \longleftarrow$ `GetEdge(`t, ε, δ_t, τ, μ_t`)`
- $\alpha_t \longleftarrow \frac{1}{2}\ln\left(\frac{1/2+\hat{\tilde{\gamma}}'_t}{1/2-\hat{\tilde{\gamma}}'_t}\right)$
- Define $H_t(x) = \arg\max_{y \in Y}\sum_{t'=1}^{t}\alpha_t[\![y \in \tilde{h}_t(x)]\!]$

(The algorithm exits from the `Filter()` function.)

Algorithm 8: *GetEdge*

(Same as for FilterBoost.M2, but with $h_t(x,y)$ replaced with $[\![y \in \tilde{h}_t(x)]\!]$.)

Algorithm 9: *FilterWL*

(Same as for FilterBoost.M2, but with
(a) r = # calls to `Filter()` and `FilterWL()` so far on round t and
(b) $p'_t(x,y)$ replaced by $\sum_{y' \neq y} q'_t((x,y),y')[\![\mu_t(y) \neq \mu_t(y')]\!]$, with q'_t defined as above.)

Algorithm 10: *Filter*

(Same as for FilterBoost.M2, but with
(a) r = # calls to `Filter()` and `FilterWL()` so far on round t and
(b) q'_t defined as above.)

Figure 5: FilterBoost.OC: multiclass FilterBoost based on AdaBoost.OC

weak hypotheses h_t (for .OC) as $\tilde{h}_t$ (for .M2), we can verify the weights q'_t and edges γ_t are equivalent to those used for FilterBoost.M2; in fact, FilterBoost.OC is a special case of FilterBoost.M2 with a particular form for weak hypotheses. We may therefore apply the same analysis.

However, FilterBoost.OC chooses examples for training the weak learner from a different distribution than that from which it chooses examples to estimate the edges of the weak hypotheses. In order to give an analog to Theorem 10 for FilterBoost.OC, we must first relate the pseudoloss $\tilde{\epsilon}_t = 1/2 - \tilde{\gamma}_t$ of $\tilde{h}_t$ to the error rate $\epsilon_t = 1/2 - \gamma_t$ of h_t. We do so in the following lemma. This proof is omitted since it is identical to that for AdaBoost.OC (Schapire, 1997) except that sums are over all possible examples instead of a fixed set of training examples.

Lemma 11 *Let $\tilde{\gamma}_t$ be the pseudoloss of $\tilde{h}_t$ with respect to the distribution D_t induced by $Filter()$ on round t (as defined in Figure 5). Let γ_t be the error of h_t with respect to the distribution induced by $FilterWL()$ and the coloring μ_t. Then*

$$\tilde{\gamma}_t = \gamma_t U_t \qquad \text{where} \quad U_t = \mathrm{E}_{D_t}\left[[\![\mu_t(y) \neq \mu_t(y')]\!]\right].$$

Combining this bound with Theorem 10 gives the following bound for FilterBoost.OC.

Theorem 12 *Let $\gamma U = \min_t |\gamma_t U_t|$. Let ε be the target error. Given Lemma 6's assumptions, if FilterBoost.OC runs*

$$T > \frac{2(k-1)\ln(2)}{\varepsilon\left(1 - 2\sqrt{1/4 - (\gamma U)^2(\frac{1-\tau}{1+\tau})^2}\right)}$$

rounds, then $err_t < \varepsilon$ for some t, $1 \leq t \leq T$.

As Schapire (1997) dicusses, colorings μ_t may easily be chosen so that $U_t > 1/2$ in expectation, but choosing colorings optimally is a hard problem.

3.3 Allwein et al. (2000) in the Filtering Setting

Allwein et al. (2000) propose a general method for reducing multiclass problems to a set of binary problems using a coding matrix $\mathbf{M} \in \{-1, 0, +1\}^{k \times l}$. Each column in this matrix represents a separate binary classification problem: for each column $s \in \{1, \ldots, l\}$, we relabel each training example (x, y) as $(x, M(y, s))$, ignoring examples with labels such that $M(y, s) = 0$. We then train a binary learner on these examples to get a hypothesis $f_s : X \longrightarrow \mathbb{R}$. They present two possible methods for combining the predictions of these hypotheses on a new example x into a predicted label $\hat{y}$:

$$\hat{y} = \arg\max_{y'} \sum_{s=1}^{l} \mathrm{sign}(M(y', s) f_x(x)) \quad \text{Hamming decoding}$$

$$\hat{y} = \arg\min_{y'} \sum_{s=1}^{l} L(M(y', s) f_x(x)) \quad \text{loss-based decoding}$$

where L is the loss the binary learner tries to minimize. (For FilterBoost, this is the logistic loss.) We refer the reader to Allwein et al. (2000) for a more detailed discussion of this approach, including how to choose the coding matrix $\mathbf{M}$.

It is thus straightforward to use this approach with FilterBoost; we simply choose a coding matrix and run FilterBoost on each of l subproblems. Moreover, the analysis of training error from Allwein et al. (2000) in the batch setting works for the analysis of the test error in the filtering setting. In particular, we can derive the following theorem by replacing the average binary loss ε (in their notation) over a finite training set (averaged over binary problems $s \in \{1, \ldots, l\}$) with the expected binary loss ϵ over the target distribution (again averaged over binary problems). Note that the loss for binary problem s ignores examples (x, y) such that $M(y, s) = 0$, so $\epsilon \equiv \mathrm{E}_D[\frac{1}{l}\sum_{s=1}^{l} L(M(y, s) f_s(x))[\![M(y, s) \neq 0]\!]]$.

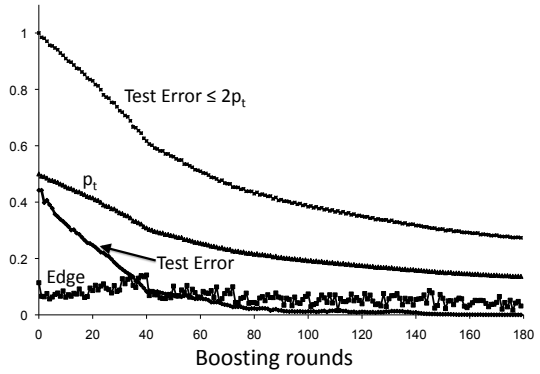

Figure 6: A sample run of FilterBoost: noisy majority vote data, WL = decision stumps, 50,000 training exs.

Theorem 13 *(Analogous to Theorem 1 and Corollary 2 from Allwein et al. (2000).) Let l, ϵ, L, M be defined as above. Let $\rho = \min_{i \neq j}\{l - M(i,\cdot) \cdot M(j,\cdot)\}$. Then the test error using Hamming-based decoding is at most*

$$\frac{2l\epsilon}{\rho L(0)},$$

and the test error using loss-based decoding is at most

$$\frac{l\epsilon}{\rho L(0)}.$$

4. Experiments

We conducted an extensive set of experiments exploring the performance of FilterBoost and comparing it against other filtering and batch boosters. Before giving these results, we present a single run of FilterBoost in Figure 6 to illustrate the algorithm's important properties. With each round, FilterBoost's test error decreases; note that the bound on the test error using p_t is quite loose. The weak hypotheses' edges tend to decrease slowly on average but do not decrease monotonically. The probability p_t with which the filter accepts examples decreases with each round, so each round uses more examples and takes longer. Batch boosters, on the other hand, take the same amount of time during each round of boosting, but they take significantly longer for larger datasets. As we will demonstrate, batch boosters are more efficient on small datasets, but as the dataset size increases, filtering boosters become much more efficient and, for many problems, are able to achieve better test performance. FilterBoost particularly excels at conditional probability estimation, and it is competitive with classification.

4.1 Experimental Setup

We run FilterBoost using the techniques detailed in Section 2.5. We test FilterBoost with and without confidence-rated predictions (labeled "(C-R)" in our results).

We compare FilterBoost against the filtering booster MadaBoost (Domingo and Watanabe, 2000), which does not require an a priori bound on weak hypotheses' edges and has similar bounds without the complication of boosting tandems. We implement MadaBoost with the same modifications as FilterBoost but did not test it with confidence-rated predictions. (It is not clear how to do so since the analysis of MadaBoost assumes binary weak hypotheses.)

We test FilterBoost against two batch boosters: the well-studied and historically successful AdaBoost (Freund and Schapire, 1997) and the previously mentioned algorithm from Collins et al. (2002) which is essentially a batch version of FilterBoost (labeled "AdaBoost-LOG"). We test both with and without confidence-rated predictions as well as with and without resampling (labeled "(resamp)"). In resampling, the booster trains weak learners on small sets of examples sampled from the distribution D_t over the training set S rather than on the entire set S, and this technique often increases efficiency with little effect on accuracy. Our batch boosters use sets of size $C_m \log(t+1)$ for training, like the filtering boosters, but use all of S to estimate edges γ_t since this can be done efficiently. We test the batch boosters using confidence-rated predictions and resampling in order to compare FilterBoost with batch algorithms optimized for the efficiency which boosting-by-filtering claims as its goal.

We test each booster using decision stumps and decision trees as weak learners to discern the effects of simple and complicated weak hypotheses. The decision stumps minimize training error, and the decision trees greedily maximize information gain and are pruned using 1/3 of the data. Both weak learners aim to minimize exponential loss when outputing confidence-rated predictions.

We use four datasets, described in the Appendix. Briefly,we use two synthetic sets: Majority (majority vote) and Twonorm (Breiman, 1998), and two real sets from the UCI Machine Learning Repository (Newman et al., 1998): Adult (census data; from Ron Kohavi) and Covertype (forestry data with 7 classes merged to 2; Copyr. Jock A. Blackard & Colorado State U.). We average over 10 runs, using new examples for synthetic data (with 50,000 test examples except where stated) and 10-fold cross validation for real data.

4.2 Running Time

Figure 7 compares the boosters' runtimes. As expected, filtering boosters run slower per round than batch boosters on small datasets but much faster on large ones. On small datasets, filtering boosters end up recomputing the weight of each example many times (when recycling examples). In addition, the probability the filter accepts an example quickly shrinks when the booster has seen that example many times. On larger datasets, filtering boosters become much more efficient; this holds true even when the datasets are small enough to fit in main memory. MadaBoost and FilterBoost have asymptotically identical running times, with MadaBoost being slightly faster by a constant due to how weights are computed.

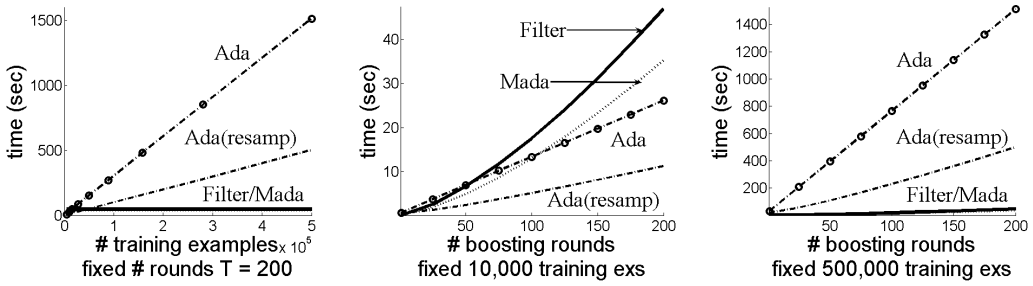

Figure 7: Running times: Ada/Filter/MadaBoost. Majority; WL = stumps.

4.3 Conditional Probability Estimation

In Section 2.1, we discussed the interpretation of FilterBoost and AdaBoost as stepwise algorithms for conditional probability estimation. We test both algorithms and the variants discussed above on all four datasets. We do not test MadaBoost, as it is not clear how to use it to estimate conditional probabilities. We use log loss and root mean squared error (RMSE) to measure learning, where the log loss on examples $\{(x_i, y_i)\}$ is $\sum_i -\ln(\widehat{\Pr}[y_i|x_i])$ and the RMSE is $(\sum_i([\![y_i = +1]\!] - \widehat{\Pr}[+1|x_i])^2)^{1/2}$. In each plot, we compare FilterBoost with the best of AdaBoost and AdaBoost-LOG: AdaBoost was best with decision stumps and AdaBoost-LOG with decisions trees.

Figures 8, 9, and 10 give the results on Adult, Majority, and Twonorm, respectively. These results indicate that both FilterBoost variants are competitive with batch algorithms when boosting decision stumps: FilterBoost does a bit better on Adult, similarly on Majority, and a bit worse on Twonorm. With decision trees, vanilla FilterBoost seems to do strictly better on all of the datasets. Though FilterBoost(C-R) does better than AdaBoost(C-R), all algorithms except for FilterBoost seem to overfit. For comparison, batch logistic regression via gradient descent achieves RMSE 0.3489 and log (base e) loss .4259 on Majority; FilterBoost with decision stumps, interpretable as a stepwise method for logistic regression, seems to be approaching these asymptotically.

The Covertype dataset is an exception to our results and highlights a danger in filtering and in resampling for batch learning: the complicated structure of some datasets seems to require a complicated weak learner and to require that the weak learner train on the entire dataset. With decision stumps, the filtering boosters are competitive, but decision trees give better results than decision stumps. Only the non-resampling batch boosters achieve high accuracies with decision trees. The first decision tree trained on the entire training set achieves losses which seem to be unreachable with decision stumps and with boosters which train on small subsets of the dataset. unachievable by any of the filtering or resampling batch boosters when using To compete with non-resampling batch boosters, both filtering boosters and resampling batch boosters must use C_m on the order of 10^5, by which point they become very inefficient.

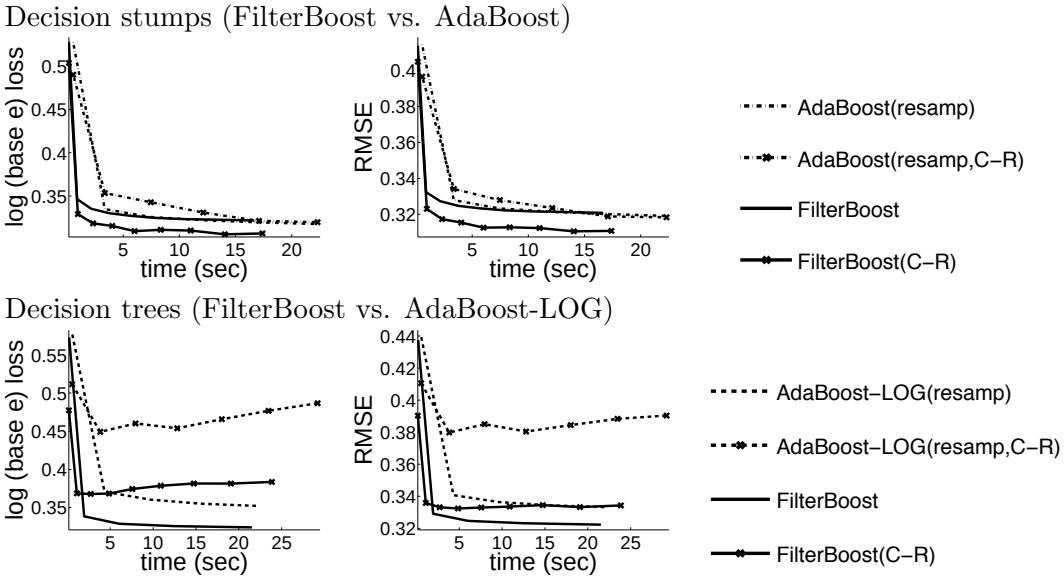

Figure 8: Conditional Probability Estimation on **Adult. 40,698 training exs.**

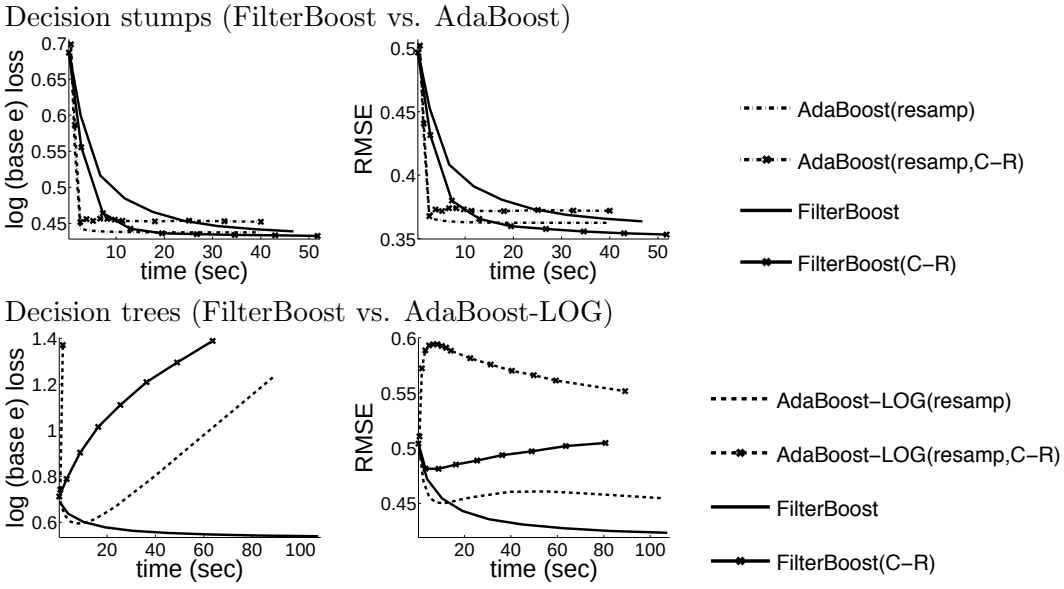

Figure 9: Conditional Probability Estimation on **Majority. 10,000 training exs.**

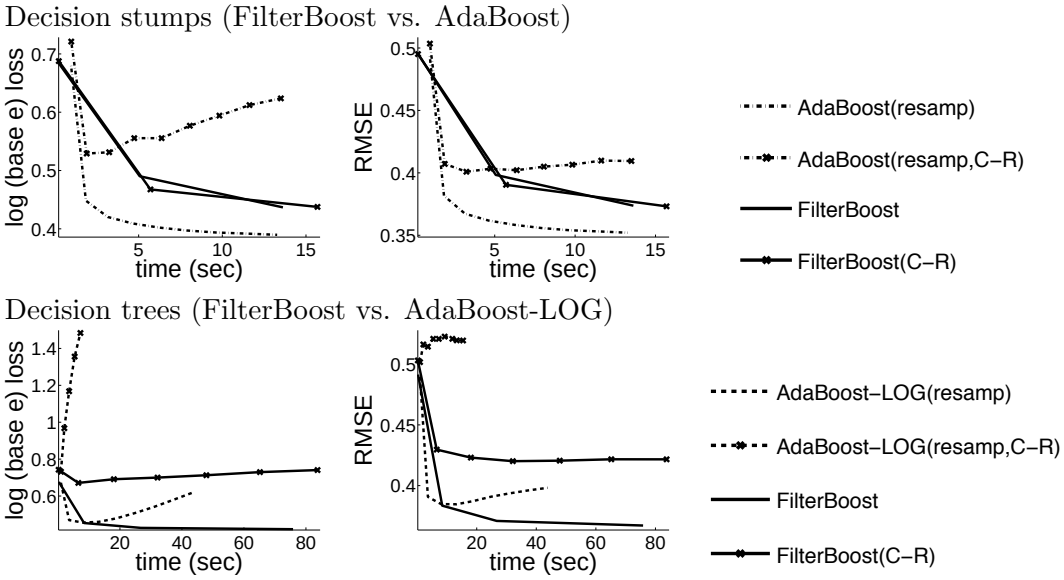

Figure 10: Conditional Probability Estimation on **Twonorm. 10,000 training exs.**

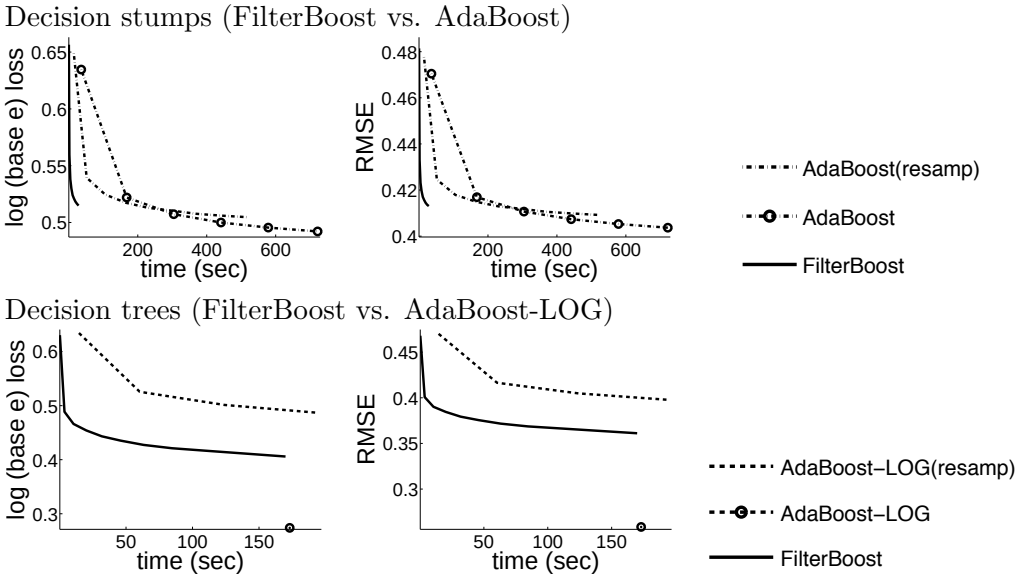

Figure 11: Conditional Probability Estimation on **Covertype. 522,909 training exs.**

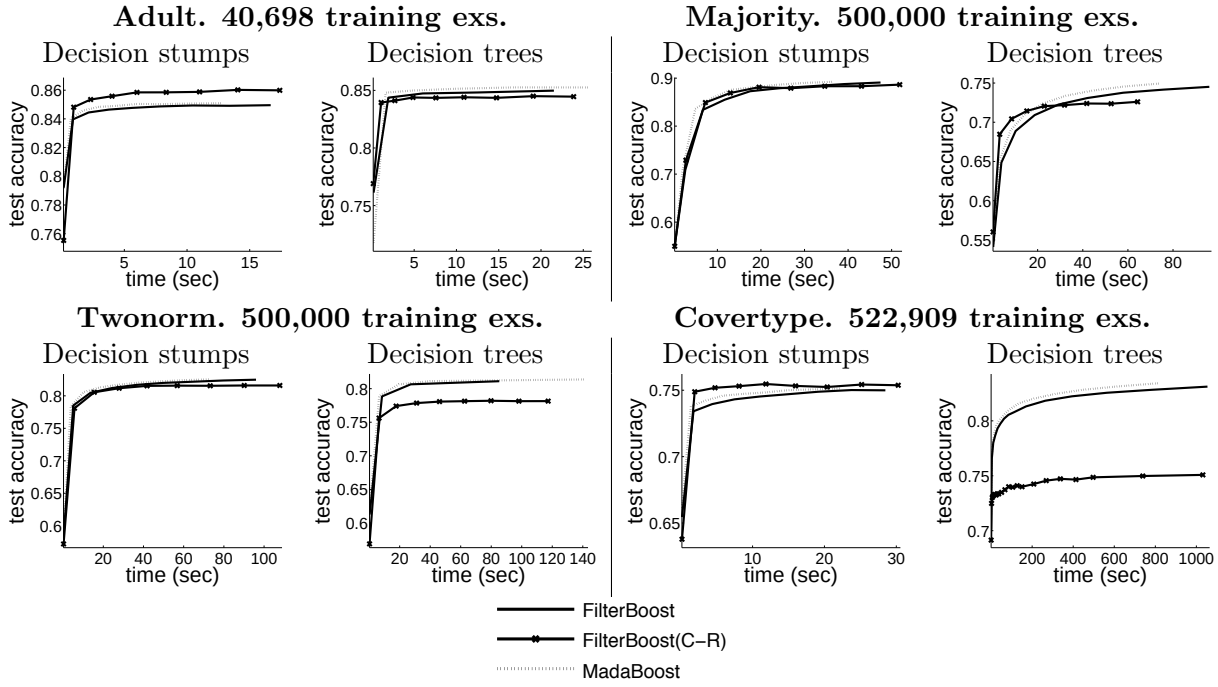

Figure 12: Classification: filtering boosters only.

4.4 Classification

Figure 12 shows that vanilla FilterBoost and MadaBoost perform similarly in classification. Confidence-rated predictions allow FilterBoost to outperform MadaBoost when using decision stumps (on Adult and Covertype) but sometimes cause FilterBoost to perform poorly with decision trees (on Twonorm and Covertype).

Figure 13 compares FilterBoost with AdaBoost for each weak learner. (AdaBoost outperformed AdaBoost-LOG for classification.) FilterBoost is competitive with AdaBoost, doing better on Adult, about the same on the small Majority dataset, and a bit worse on Twonorm. The large Majority dataset shows where FilterBoost performs best: with decision stumps, all boosters achieve higher accuracies with the larger dataset, on which filtering algorithms are much more efficient. Majority is represented well as a linear combination of decision stumps, so the boosters all learn more slowly when using the overly complicated decision trees. However, this problem generally affects filtering boosters less than most batch variants, especially on larger datasets.

Figure 14 compares FilterBoost with AdaBoost on Covertype. As for conditional probability estimation, FilterBoost performs better when using decision stumps, but with decision trees, AdaBoost without resampling performs much better than both FilterBoost and AdaBoost with resampling.

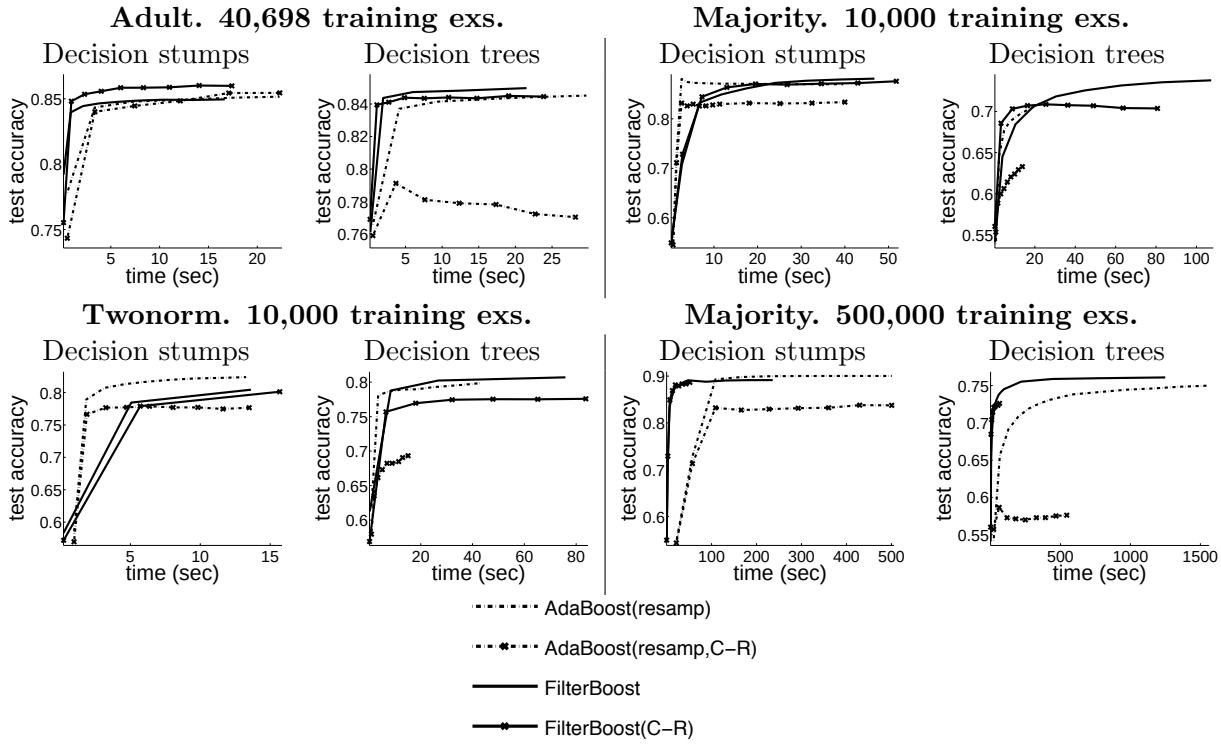

Figure 13: Classification: FilterBoost vs. AdaBoost and AdaBoost-LOG on Adult, Majority, and Twonorm.

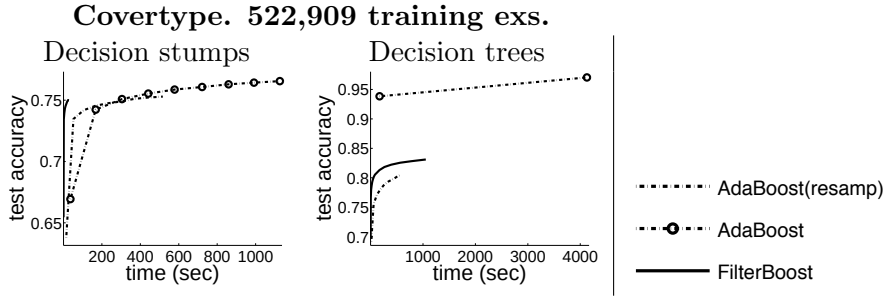

Figure 14: Classification: FilterBoost vs. AdaBoost and AdaBoost-LOG on Covertype.

5. Concluding Remarks

We have proposed a novel boosting-by-filtering algorithm and shown its applicability to classification and conditional probability estimation. Our experiments prove FilterBoost is competitive with existing batch and filtering boosters in both applications. FilterBoost can often learn faster than batch boosters by making efficient use of large datasets, and it tends to be more resilient to overfitting, especially in conditional probability estimation.

We suggest several open questions. Bshouty and Gavinsky (2002) develop their filtering booster in terms of smooth boosting, which implies robustness to certain types of noise, and it would be interesting to interpret FilterBoost as a smooth booster to justify its empirically shown robustness. Also, FilterBoost and MadaBoost's bounds on the required number of boosting rounds are exponentially larger in $1/\varepsilon$ than those for batch AdaBoost and the filtering boosters of Freund (1995) and Bshouty and Gavinsky (2002). FilterBoost is empirically competitive with AdaBoost, though, so it would be interesting to prove either a tighter upper bound or a lower bound on T for FilterBoost. Finally, FilterBoost is based upon its batch equivalent, the algorithm proposed by Collins et al. (2002). While FilterBoost is provably a boosting algorithm in the filtering setting, it is not yet known if Collins et al.'s algorithm is a boosting algorithm in the batch setting.

Appendix A: Proof of Theorem 1

Let $\pi(F+\alpha h) = \mathrm{E}[\ln(1+e^{-y(F(x)+\alpha h(x))})]$. Given the previous estimate $F(x)$, we first fix α and choose $h(x)$ to minimize a second-order expansion of $\pi(F+\alpha h)$ around $h(x)=0$.

$$\begin{aligned}\pi(F+\alpha h) &= \mathrm{E}\left[\ln(1+e^{-yF(x)}) - \frac{y\alpha h(x)}{1+e^{yF(x)}} + \frac{1}{2}\frac{y^2\alpha^2 h(x)^2 e^{yF(x)}}{(1+e^{yF(x)})^2}\right] \\ &= \mathrm{E}\left[\ln(1+e^{-yF(x)}) - \frac{y\alpha h(x)}{1+e^{yF(x)}} + \frac{1}{2}\frac{\alpha^2 e^{yF(x)}}{(1+e^{yF(x)})^2}\right]\end{aligned}$$

For $\alpha > 0$, minimizing this approximation of $\pi(F+\alpha h)$ with respect to $h(x)$ is equivalent to maximizing the weighted expectation $\mathrm{E}_q[yh(x)] \equiv \mathrm{E}[q(x,y)yh(x)]$ where $q(x,y) = \frac{1}{1+e^{yF(x)}}$. This criterion is optimized for $f(x) = \mathrm{sign}(\mathrm{E}_q[y|x])$.

Now, given $h(x)$, FilterBoost chooses α to minimize the upper bound

$$\pi(F+\alpha h) \le \mathrm{E}[e^{-y(F(x)+\alpha h(x))}].$$

This is the same optimization objective used by AdaBoost and is minimized when $\alpha = \frac{1}{2}\log(\frac{1/2+\gamma}{1/2-\gamma})$ where γ is the edge of $h(x)$; this is exactly the α used by FilterBoost. ■

Appendix B: Proof of Lemma 3

$$\pi_t - \pi_{t+1} = \sum_{(x,y)} D(x,y)\ln\left(\frac{1-q_{t+1}(x,y)}{1-q_t(x,y)}\right) \quad (1)$$

Since $q_t(x,y) = \frac{1}{1+e^{yF_t(x)}}$, $F_t(x) = \sum_{t'=1}^{t-1} \alpha_{t'} h_{t'}(x)$,

$$e^{yF_t(x)} = \frac{1}{q_t(x,y)} - 1 \text{ and} \tag{2}$$

$$q_{t+1}(x,y) = \frac{1}{1 + e^{yF_t(x)+\alpha_t y h_t(x)}} \tag{3}$$

Defining $v_t(x,y) = \alpha_t y h_t(x)$, combining (2) and (3) gives

$$q_{t+1}(x,y) = \frac{1}{1 + (\frac{1}{q_t(x,y)} - 1)e^{v_t(x,y)}} = \frac{q_t(x,y)}{q_t(x,y) + (1 - q_t(x,y))e^{v_t(x,y)}} \tag{4}$$

Substituting (4) into (1), and using $\ln(1+z) \leq z$, gives

$$\begin{aligned} \pi_t - \pi_{t+1} &= -\textstyle\sum_{(x,y)} D(x,y) \ln(q_t(x,y)e^{-v_t(x,y)} + 1 - q_t(x,y)) \\ &\geq -\textstyle\sum_{(x,y)} D(x,y)(-q_t(x,y) + q_t(x,y)e^{-v_t(x,y)}) \\ &= \textstyle\sum_{(x,y)} D(x,y) q_t(x,y) - \sum_{(x,y)} D(x,y) q_t(x,y) e^{-v_t(x,y)} \end{aligned}$$

Let $D_t(x,y) = \frac{D(x,y)q_t(x,y)}{p_t}$. Then we can write

$$\pi_t - \pi_{t+1} \geq p_t - p_t \sum_{(x,y)} D_t(x,y) e^{-\alpha_t y h_t(x)} \tag{5}$$

Using $\alpha_t = \frac{1}{2}\ln(\frac{1/2+\gamma_t}{1/2-\gamma_t})$ and $\epsilon_t \equiv \Pr_{D_t}[\text{sign}(h_t(x)) \neq y]$ lets us write

$$\sum_{(x,y)} D_t(x,y) e^{-\alpha_t y h_t(x)} = e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t = 2\sqrt{\frac{1}{4} - \gamma_t^2}$$

Substituting this factor into (5) completes the proof. ■

Appendix C: Proof of Lemma 6

The proof is identical to Lemma 1 up to (5). Now, though, $\alpha_t = \frac{1}{2}\ln(\frac{1/2+\hat{\gamma}'_t}{1/2-\hat{\gamma}'_t})$. Using $\Pr[|\hat{\gamma}_t - \gamma_t| \leq \tau\gamma_t] > 1 - \delta_t$ and $\hat{\gamma}'_t = \frac{\hat{\gamma}_t}{1+\tau}$, we know $\gamma_t \geq \frac{\hat{\gamma}_t}{1+\tau}$ with probability at least $1-\delta_t$, which in turn implies $\hat{\gamma}'_t \leq \gamma_t$. So we may rewrite and bound the sum in (5) as:

$$\begin{aligned} &\sum_{(x,y)} D_t(x,y) e^{-\alpha_t y h_t(x)} = e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t \\ = &\left(\frac{\frac{1}{2} - \hat{\gamma}'_t}{\frac{1}{2} + \hat{\gamma}'_t}\right)^{1/2} \left(\frac{1}{2} + \gamma_t\right) + \left(\frac{\frac{1}{2} + \hat{\gamma}'_t}{\frac{1}{2} - \hat{\gamma}'_t}\right)^{1/2} \left(\frac{1}{2} - \gamma_t\right) \\ \leq &\left(\frac{\frac{1}{2} - \hat{\gamma}'_t}{\frac{1}{2} + \hat{\gamma}'_t}\right)^{1/2} \left(\frac{1}{2} + \hat{\gamma}'_t\right) + \left(\frac{\frac{1}{2} + \hat{\gamma}'_t}{\frac{1}{2} - \hat{\gamma}'_t}\right)^{1/2} \left(\frac{1}{2} - \hat{\gamma}'_t\right) \\ = &\ 2\sqrt{1/4 - \hat{\gamma}'^2_t} \end{aligned}$$

Substituting in $\hat{\gamma}'_t = \frac{\hat{\gamma}_t}{1+\tau}$ and using $\hat{\gamma}_t \geq \gamma_t(1-\tau)$ gives

$$\begin{aligned} \textstyle\sum_{(x,y)} D_t(x,y)e^{-\alpha_t y h_t(x)} &\leq 2\sqrt{1/4 - (\tfrac{\hat{\gamma}_t}{1+\tau})^2} \\ &\leq 2\sqrt{1/4 - \gamma_t^2(\tfrac{1-\tau}{1+\tau})^2} \end{aligned}$$

Substituting into (5) gives the required bound. ■

Appendix D: Proof of Theorem 10

In this proof, we closely follow the methodology and notation of Freund and Schapire (1997). Note, however, that we use labels $\{-1, 1\}$ instead of $\{0, 1\}$ and weak hypothesis predictions in $[-1, 1]$ instead of $[0, 1]$, respectively. We reduce FilterBoost.M2 to FilterBoost and invoke Theorem 7. FilterBoost variables are denoted with tildes. For each FilterBoost.M2 training example (x, y), we define $k-1$ training examples for FilterBoost: $\tilde{x}_{(x,y),y'} \equiv ((x, y), y')$ for each incorrect label $y' \neq y$; note that $((x, y), y')$ is an unlabeled example, and its label is $\tilde{y} = -1$. (In our notation, we omit the label -1 since all labels given to FilterBoost are -1.) FilterBoost's target distribution is $\tilde{D}((x, y), y') = D(x, y)/(k-1)$. We give FilterBoost a weak hypothesis $\tilde{h}_t$ defined using the weak hypothesis h_t chosen by FilterBoost.M2:

$$\tilde{h}_t((x, y), y') = \frac{1}{2}\left(-h_t(x, y) + h_t(x, y')\right).$$

We estimate edges in FilterBoost not by the normal *GetEdge*() function but, rather, by using all of the examples $\tilde{x}_{(x,y),y'}$ corresponding to examples (x, y) used by FilterBoost.M2 to estimate edges. We review existing notation and define new notation for FilterBoost.M2 which helps us relate it to FilterBoost:

$$\begin{aligned} q'_t((x, y), y') &= \text{weight on example } (x, y), \text{ label } y' && (6) \\ p'_t(x, y) &= \sum_{y' \neq y} q'_t((x, y), y') = \text{probability filter accepts example } (x, y) && (7) \\ p_t &= \sum_{(x,y)} D(x, y)p'_t(x, y) = 1/(\text{E}[\#\text{ examples filter uses in 1 call}]) && (8) \\ D_t(x, y) &= \frac{D(x, y)p'_t(x, y)}{p_t} = \text{distribution induced by filter on round } t && (9) \end{aligned}$$

Plugging in the definition of $\tilde{h}_t$ and $\tilde{y} = -1$ into $\tilde{q}_t$, we can see

$$q'_t((x, y), y') = \tilde{q}_t((x, y), y'). \quad (10)$$

Note this implies $p_t = (k-1)\tilde{p}_t$. We can now verify that $\gamma_t = \tilde{\gamma}_t$, which will imply that $\alpha_t = \tilde{\alpha}_t$. To see $\gamma_t = \tilde{\gamma}_t$, consider the errors $\epsilon_t = 1/2 - \gamma_t$ and $\tilde{\epsilon}_t = 1/2 - \tilde{\gamma}_t$:

$$\begin{aligned} \tilde{\epsilon}_t &= \text{E}_{\tilde{D}_t}\left[\frac{1 - \tilde{y}\tilde{h}_t((x, y), y')}{2}\right] && \text{(by definition of error for FilterBoost)} \\ &= \frac{1}{2}\text{E}_{\tilde{D}_t}\left[1 + \tilde{h}_t((x, y), y')\right] && \text{(using } \tilde{y} = -1) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}\mathrm{E}_{\tilde{D}_t}\left[1 - h_t(x,y) + h_t(x,y')\right] \qquad \text{(by definition of } \tilde{h}_t\text{)} \\
&= \frac{1}{2}\sum_{(x,y)} D(x,y) \sum_{y' \neq y} \frac{1}{k-1} \frac{\tilde{q}_t((x,y),y')}{\tilde{p}_t}\left[1 - h_t(x,y) + h_t(x,y')\right] \qquad \text{(definition of } \tilde{D}_t\text{)} \\
&= \frac{1}{2}\sum_{(x,y)} D(x,y) \sum_{y' \neq y} \frac{q'_t((x,y),y')}{p_t}\left[1 - h_t(x,y) + h_t(x,y')\right] \qquad \text{(by Eq. (10))} \\
&= \frac{1}{2}\sum_{(x,y)} \frac{D(x,y)p'_t(x,y)}{p_t}\left[1 - h_t(x,y) + \sum_{y' \neq y} q_t((x,y),y')h_t(x,y')\right] \qquad \text{(by Eq. (7))} \\
&= \mathrm{E}_{D_t}\left[\frac{1}{2}\left(1 - h_t(x,y) + \sum_{y' \neq y} q_t((x,y),y')h_t(x,y')\right)\right] \qquad \text{(by Eq. (9))} \\
&= \epsilon_t \qquad \text{(by definition of error for FilterBoost.M2)}
\end{aligned}
$$

This above equality shows that the estimates of weak hypothesis error rates are the same in expectation for the two algorithms; an analogous argument shows that the empirical estimates are identical since we use the same examples (by how we constructed this reduction). We have thus shown that our instantiation of FilterBoost produces the same results as FilterBoost.M2 when run via this reduction. The rest of this proof is essentially identical to that in Freund and Schapire (1997), but we reproduce it here in our notation. Note that their proof is with respect to training error on a fixed training set, while ours is with respect to the test error.

Suppose, for FilterBoost.M2, that $H_t(x) \neq y$ for some (x, y). Then

$$\sum_{t'=1}^{t} \alpha_t h_t(x,y) \leq \sum_{t'=1}^{t} \alpha_t h_t(x, H_t(x)),$$

which implies

$$\sum_{t'=1}^{t} \alpha_t \tilde{h}_t((x,y), H_t(x)) = \frac{1}{2}\sum_{t'=1}^{t} \alpha_t(-h_t(x,y) + h_t(x, H_t(x))) \geq \frac{1}{2}\sum_{t'=1}^{t} \alpha_t,$$

so $\tilde{H}_t((x,y), H_t(x)) = 1$ by the definition of $\tilde{H}_t$. This implies

$$\Pr_D\left[H_t(x) \neq y\right] \leq \Pr_D\left[\exists y' \neq y : \tilde{H}_t((x,y),y') = 1\right].$$

Using $\tilde{y} = -1$ for all FilterBoost examples and the definition of $\tilde{D}$, we get

$$\Pr_D\left[\exists y' \neq y : \tilde{H}_t((x,y),y') = 1\right] \leq (k-1)\Pr_{\tilde{D}}\left[\tilde{H}_t((x,y),y') = 1\right],$$

where the term on the right is $(k-1)$ times the error rate of FilterBoost. We can combine this bound with Theorem 7 to complete the proof. ■

Appendix E: Datasets

Majority is generated by a majority vote rule among 40 of 100 binary attributes, with labels corrupted with 10% probability. Twonorm is a noisy synthetic dataset with 20 real-valued attributes from Breiman (1998). Adult is from the UCI Machine Learning Repository (Newman et al., 1998), donated by Ron Kohavi. Adult consists of 14-attribute census data, with labels indicating income level, and eliminating examples with missing attribute values left 45222 examples. Covertype (copyrighted by Jock A. Blackard and Colorado State U.) is also from the UCI Machine Learning Repository. It contains 54-attribute forestry data, where examples are locations and labels indicate the type of tree cover. The original dataset has 7 classes, but we combined the 6 smallest to make the dataset binary, leaving the largest (49% of the examples) alone.

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

J. K. Bradley and R. E. Schapire. Filterboost: Regression and classification on large datasets. In *Advances in Neural Information Processing Systems 20*, 2007.

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26:801–849, 1998.

N. H. Bshouty and D. Gavinsky. On boosting with polynomially bounded distributions. *Journal of Machine Learning Research*, 3:483–506, 2002.

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48:253–285, 2002.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for monte carlo estimation. *SIAM Journal on Computing*, 29:1484–1496, 2000.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

C. Domingo and O. Watanabe. Madaboost: a modification of adaboost. In *13th Annual Conference on Computational Learning Theory*, pages 180–189, 2000.

C. Domingo, R. Galvadà, and O. Watanabe. Adaptive sampling methods for scaling up knowledge discovery algorithms. *Data Mining and Knowledge Discovery*, 6:131–152, 2002.

Y. Freund. An improved boosting algorithm and its implications on learning complexity. In *5th Annual Conference on Computational Learning Theory*, pages 391–398, 1992.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121:256–285, 1995.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28:337–407, 2000.

D. Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *Journal of Machine Learning Research*, 4:101–117, 2003.

K. Hatano. Smooth boosting using an information-based criterion. In *17th International Conference on Algorithmic Learning Theory*, pages 304–319, 2006.

V. Mnih, C. Szepesvari, and J.-Y. Audibert. Empirical bernstein stopping. In *25th International Conference on Machine Learning*, 2008.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. *UCI Repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html].* U. of California, Dept. of Information and Computer Science, Irvine, CA, 1998.

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

R. E. Schapire. Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 313–321, 1997.

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

R. A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4, 2003.

S. Shalev-Shwartz and Y. Singer. Convex repeated games and fenchel duality. In *Advances in Neural Information Processing Systems 19*, 2006.

V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

O. Watanabe. Simple sampling techniques for discovery science. *IEICE Trans. Information and Systems*, E83-D(1):19–26, 2000.