Boosting on a Budget: Sampling for Feature-Efficient Prediction

Lev Reyzin LREYZIN@CC.GATECH.EDU
Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA

Abstract

In this paper, we tackle the problem of feature-efficient prediction: classification using a limited number of features per test example. We show that modifying an ensemble classifier such as `AdaBoost`, by sampling hypotheses from its final weighted predictor, is well-suited for this task. We further consider an extension of this problem, where the costs of examining the various features can differ from one another, and we give an algorithm for this more general setting. We prove the correctness of our algorithms and derive bounds for the number of samples needed for given error rates. We also experimentally verify the effectiveness of our methods.

1. Introduction

Oftentimes, while a learning algorithm has ample time and resources to build a good predictor in the training phase, accessing the features of a new example at test time can be costly. This problem often arises in medical diagnosis, where we can think of looking at a new feature as performing a new test on the patient. Tests can be expensive, take valuable time, and be inconvenient or possibly dangerous to the patients; thus, it is important to be able to diagnose as well as possible without performing unnecessary tests. This is the problem we address herein.

More precisely, we consider the setting where a learning algorithm is given a limit on the number of features it may observe on a given example in the test phase, and subject to this constraint, must try to predict as accurately as possible. We call this problem **feature-efficient prediction**. In this model, the learner may look at arbitrarily many features during training and is only limited in test time.

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

In this paper, we propose a technique for sampling an ensemble predictor for tackling this problem. Ensemble predictors classify new examples by taking a weighted vote of many **base learners**, each of which can be a simple predictor that looks at only a few features. Specifically, we modify the boosting algorithm `AdaBoost` to obtain a feature-efficient algorithm. Our method, however, is sufficiently general as to be applicable to any ensemble method, e.g. boosting, bagging, and even linear predictors, whose classification is simply a weighted vote of feature values.

The idea behind our modification is simple but powerful: to train an entire ensemble classifier, and then, for each new example, to predict by sampling the base learners' votes using a distribution induced by their weights. In this paper, we show that using base learners that use only a few features and sampling a limited number of hypotheses can still provide nearly the accuracy of the full ensemble predictor, and yet use significantly fewer features in the process.

We consider two versions of the problem: the **uniform cost** version, where the cost of examining each feature is the same and the **arbitrary cost** version, where the cost of examining each feature is arbitrary. We give algorithms for both these versions above and demonstrate the effectiveness of our approach in theory and experiment.

2. Background

2.1. Past Work

Work on various forms of feature efficiency has taken many forms – here, we give a brief overview of some relevant literature.

In the area of sequential analysis, Wald (1947) began a line of research considering the problem of running a clinical trial sequentially, only testing future patients if the validity of the hypothesis in question is still sufficiently uncertain. A thorough treatment of sequential analysis is given in (Chernoff, 1972).

In learning theory, a variant of our problem was stud-

ied by Ben-David and Dichterman (1993), who examined the theory behind learning using random partial information from examples and discussed conditions for learning in their model.

Greiner et al. (2002) considered the problem of feature-efficient prediction, where a classifier must choose which features to examine before predicting. They showed that a variant of PAC-learnability is still possible even without access to the full feature set.

The problem of predicting quickly and efficiently has also received interest due to its relevance in internet-scale applications. In that vein, Globerson and Roweis (2006) looked at building robust predictors that are resilient to corrupted or missing features.

Recently, Cesa-Bianchi et al. (2010) studied how to efficiently learn a linear predictor, in the setting where the learner can access only a few features per example. Their work tackles a problem similar to our own, but is confined to learning linear predictors.

In another related recent work, Pelossof et al. (2010) analyzed how to speed up margin-based learning algorithms by stopping evaluation when the outcome is close to certain. In our setting, we know in advance when we need to stop evaluation and try to optimize accordingly. In their setting, they try to achieve a certain accuracy, and subject to that, determine when they can stop the evaluation.

2.2. Boosting

Our idea is to exploit the power of ensemble predictors, focusing on `AdaBoost` – here we give a short introduction to boosting and the related concepts we rely on.

Boosting algorithms combine moderately inaccurate prediction rules and make a weighted majority vote to form a single classifier. On each round, a boosting algorithm generates a new prediction rule, or base learner, to use and then places more weight on the examples classified incorrectly. Hence, boosting constantly focuses on correctly classifying the hardest examples. A nice overview of boosting appears in Schapire (2003).

In this paper, we will make use of `AdaBoost` (Freund & Schapire, 1997), the most ubiquitous of the boosting algorithms, yet our results hold for any ensemble predictor. The `AdaBoost` algorithm is given in Algorithm 1.

Soon after its appearance, it was observed that `AdaBoost` tends not to **overfit** training data with more rounds of boosting, despite that the combined classifier's complexity increases with every round. For more detail, we refer the reader to the rich amount of work in this area (Grove & Schuurmans, 1998; Schapire et al., 1998; Mason et al., 1998; Reyzin & Schapire, 2006).

Algorithm 1 `AdaBoost` (Freund & Schapire, 1997)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$
where $x_i \in X$, $y_i \in Y = \{-1, +1\}$.

Initialize $D_1(i) = 1/m$.

for $t = 1, \ldots, T$ **do**
  Train base learner using distribution D_t.
  Get base classifier $h_t : X \to \{-1, +1\}$.
  Let $\gamma_t = \sum_i D_t(i) y_i h_t(x_i)$.
  Choose $\alpha_t = \frac{1}{2} \ln \frac{1+\gamma_t}{1-\gamma_t}$.
  Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

  where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).
end for

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

An important part of the history of trying to account for `AdaBoost`'s performance is Schapire et al.'s (1998) explanation of `AdaBoost`'s tendency not to overfit in terms of the margins of the training examples, where the **margin** is a quantity that can be interpreted as measuring the confidence and correctness in the prediction of the combined classifier on the training examples.

The power of the margin bound is that it depends only on the margins distribution and not on the number of rounds of boosting, and it turns out that the margins distribution `AdaBoost` generates generally improves with more rounds (Schapire et al., 1998) – this motivates our specific choice of `AdaBoost` for the boosting algorithm. As we shall see in Section 3.2, the work of Schapire et al. (1998) drives our analysis of Algorithm 2.

3. Algorithm and Analysis for Uniform Costs

In this section we present and analyze our algorithm for feature-efficient prediction in the uniform cost set-

ting. The goal is to predict as accurately as possible, while examining at most $B > 0$ features on any given test example.

3.1. Algorithm

We propose Algorithm 2 which we call `AdaBoostRS` (for `AdaBoost`, Randomly Sampled), for feature-efficient prediction. `AdaBoostRS` uses `AdaBoost` in training and then samples from `AdaBoost`'s hypothesis distribution. `AdaBoostRS` samples anew for each test example to avoid correlating their errors. We assume the budget limit B is greater than the number of (relevant) features; otherwise, the full `AdaBoost` vote can simply be used.

For each hypothesis h we denote by $n(h)$ the number of features required for evaluating the hypothesis. For example, a decision stump h has $n(h) = 1$.

Algorithm 2 `AdaBoostRS`

Given: $(x_1, y_1), \ldots, (x_m, y_m)$
where $x_i \in X$, $y_i \in Y = \{-1, +1\}$.
Generate $(\alpha_1, h_1), \ldots, (\alpha_T, h_T)$ using Algorithm 1.

Given: test example x, a bound B.
Initialize:

$$p(i) = \frac{\alpha_i}{\sum_{i=1}^{T} \alpha_t}$$

and $k = 0$, $F = \emptyset$, $\tau = 0$, and $v_0(x) = 0$.

while

$$k + \max_{1 \le i \le T} n(h_i) < B$$

do
  choose h_i according to $p(i)$
  let F_{h_i} be the set of features used by h_i
  let $l = |F_{h_i} \setminus F|$
  $k = k + l$
  $F = F \cup F_{h_i}$
  if $h_i(x) = 1$ **then**
    $v_{\tau+1}(x) = v_\tau(x) + 1$,
  else
    $v_{\tau+1}(x) = v_\tau(x) - 1$.
  end if
  increment τ by 1.
end while

Predict: $f(x) = \text{sign}(v_\tau)$.

3.2. Margin Bound

Schapire et al. (1998) derived a bound on the generalization error of a voting classifier based on its training margins. The margin of example (x, y) depends on the votes $h_t(x)$ with weights α_t of all the hypotheses:

$$\text{margin}(x, y) = \frac{y \sum_t \alpha_t h_t(x)}{\sum_t \alpha_t}.$$

The magnitude of the margin represents the strength of agreement of the base classifiers, and its sign indicates whether the combined vote produces a correct prediction. Then, using the distribution of the margins on training examples, one can bound the generalization error of an ensemble predictor as follows.

Theorem 1 (**margin bound** (Schapire et al., 1998))**.** *Let D be a distribution over $X \times \{-1, +1\}$ and let S be a sample of m examples chosen independently at random according to D. Suppose the base classier space H has VC-dimension d, and let $\delta \in (0, 1]$. Then with probability at least $1 - \delta$ over the random sample[1], every function f taking a weighted average over hypotheses in H satisfies the following bound for all $\theta > 0$,*

$$\boldsymbol{P}_D[yf(x) \le 0] \le \boldsymbol{P}_S[yf(x) \le \theta] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right).$$

We notice that this margin bound depends most heavily on the margins near the bottom of the distribution, since having generally high smallest margins allows θ to be large, and therefore the quantity $\tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$ to be small, without $\mathbf{P}_S[yf(x) \le \theta]$ getting too large.

Before getting to a consequence of Theorem 1, we will need the following bound.

Theorem 2 (**Hoeffding's inequality** (1963))**.** *Let $X_1, \ldots, X_n$ be independent real-valued random variables such that each $X_i \in [a_i, b_i]$ and let $S = \sum_{i=1}^{n} X_i$. Then for every $t > 0$,*

$$\boldsymbol{P}\left[|S - \boldsymbol{E}[S]| > t\right] \le \exp\left(-\frac{2t^2}{\sum_{i=1}^{n} (b_i - a_i)^2}\right).$$

Next we present a corollary of Theorem 1, a version of which also appears within the margin bound proof in (Schapire et al., 1998).

Corollary 3. *Under conditions of Theorem 1, if Algorithm 2 is run, taking B samples of hypotheses to produce a vote $f(x)$, with probability at least $1 - \delta$, for all θ,*

$$\boldsymbol{P}_D[yf(x) \le 0] \quad \le \quad \boldsymbol{P}_S[yf(x) \le \theta] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right) + e^{-B\theta^2/8}.$$

[1] The dependence on δ in the bound is hidden in the $\tilde{O}$.

Proof. We examine the probability the sampling procedure in Algorithm 2 will yield a margin of $\theta/2$ on a training instance whose margin in the full vote is θ.

Each hypothesis gives a binary vote and each weighted sampling is a random variable in $X \in \{-1, +1\}$, of mean θ. Let X_i be the ith vote, and let $U = \sum_{i=1}^{B} X_i$. Hence, $\mathbf{E}[U] = B\theta$.

The probability their unweighted average is $\leq \theta/2$ can be bounded by Hoeffding's inequality.

$$\begin{aligned}\mathbf{P}\left[|U - \mathbf{E}[U]| \geq \frac{B\theta}{2}\right] &\leq e^{-\frac{2(B\theta/2)^2}{4B}} \\ &= e^{-B\theta^2/8}.\end{aligned}$$

Combining the above with Theorem 1 and replacing θ for $\theta/2$ in the bound (this difference is hidden inside the $\tilde{O}$) gives the additional error and completes the proof. □

In general, given the margins, this technique can be used to bound the number of disagreements in prediction between `AdaBoost` and `AdaBoostRS`. We see that $B = \Theta\left(\frac{1}{\theta^2}\right)$ is sufficient for the guarantee in Theorem 1.

This gives the following proposition.

Proposition 4. *We observe that in the limit of the number of samples taken, the resulting classification of `AdaBoostRS` is equal to the vote of its underlying boosting algorithm, as in: let $v_\tau(x)$ be the vote generated by `AdaBoostRS` as defined in Algorithm 2, then with probability* 1*,*

$$\lim_{\tau \to \infty} \text{sign}(v_\tau(x)) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

When Algorithm 2 runs with decision stumps as the base hypotheses, we can be more concrete about the analysis. A decision stump is a one-node decision tree, which is a base learner that uses only one feature to make its prediction. If an example x has n binary features, then there are $2n$ possible decision stumps, one for the presence and absence of each feature. There, we can replace d with $\ln(n)$ in the margin bound – and also know that the number of samples we can take bounds the number of features that can be examined.

We also note the existence of similar (and sometimes stronger) margin bounds (see (Breiman, 1999; Langford et al., 2001; Wang et al., 2008)). These carry similar forms and use many of the same techniques as (Schapire et al., 1998) – Corollary 3 could be modified to give bounds for `AdaBoostRS` in terms of different properties of the margins.

Finally, this algorithm is easy to run online by using an online boosting algorithm instead of `AdaBoost`.

3.3. Balls and Bins

Another advantage of this method is that often a hypothesis will be selected that can be evaluated by looking at features that have already been examined. If we take k samples uniformly at random from n features, then the probability a given feature is not sampled is

$$\left(1 - \frac{1}{n}\right)^k = \left(1 - \frac{1}{n}\right)^{nk/n} \leq \left(\frac{1}{e}\right)^{k/n}.$$

Therefore, the expected number of unsampled features is bounded by $n\left(\frac{1}{e}\right)^{k/n}$. Thus, even if we draw $k = n$ samples, we still are expected to have left a constant fraction of the features unsampled.

Substituting $k = \Theta\left(\frac{1}{\theta^2}\right)$, we get a bound of $n\left(\frac{1}{e}\right)^{1/n\theta^2}$, which means that if θ is fixed, even as we have more features, we need to sample a vanishing fraction to meet the full margin bound results.

Of course, the α distribution is normally not uniform on the decision stumps, and therefore on the features as well. This only improves the situation, as it is not hard to see that the uniform distribution on features is the most pessimistic case for our analysis. We shall see in Section 5 that empirical results suggest that the number of features left unexamined is significantly better than this bound.

3.4. Observations

We observe that Algorithm 2 has some other nice properties. For one, it does not need to know the limit on number of features in advance. In the medical setting, it gives a procedure for running tests until the patient refuses any more or until time runs out. Then, the final vote will consist of the classifiers sampled until that point.

One advantage of `AdaBoostRS` over the work of Cesa-Bianchi et al. (2010) is that it, like `AdaBoost`, can be a non-linear predictor if, say, decision trees are used as the base classifier. Assuming the trees are of small depth, the algorithm should remain feature-efficient and still be able to exploit the power of non-linearity.

4. Non-Uniform Costs

We now consider extending the model to the case where different features impose different costs to the budget. To define our notation, we denote the ith example $x_i = \{x_i^1, \ldots, x_i^n\}$, where each x_i^j is the jth feature of example x_i.

We now assume that each feature j incurs cost of $c_j \geq 1$. If $F_{h(x)}$ is the set of features hypothesis h examines in predicting example x, we say that the cost of using these features is

$$c(h(x)) = \sum_{x^j \in F_{h(x)}} c_j,$$

and we want to impose the constraint that the incurred cost for predicting any example is

$$c(h(x)) \leq B.$$

If the costs are equal, then $c(h(x)) = n(h(x))$. We assume that $\max_j c_j \leq B$, as in no feature alone will bring us over budget. We can again assume the budget us low enough such that we cannot simply take the full vote of `AdaBoost`.

We notice that for this task, we could simply use `AdaBoostRS` in the following manner: we can disregard the costs in our sampling procedure, and only consider (the new, non-uniform) feature costs when deciding when we've gone over-budget. This only changes the number of samples allowed to be taken, but not the correctness of the algorithm.

However, for this more general task, we present Algorithm 3, which we call $\texttt{AdaBoostRS}_{\texttt{AC}}$ (for `AdaBoostRS`, Arbitrary Cost version). The main idea behind $\texttt{AdaBoostRS}_{\texttt{AC}}$ is to sample from the the αs distribution according to each hypotheses weight divided by its cost. In this way, we increase the probability of sampling from low-cost hypotheses (and decrease the probability of sampling high-cost ones), so that the algorithm could have the opportunity to take more samples. To correct for the bias this creates, we use the technique of **importance sampling** to properly take feature costs into account when taking the weighted vote. Sutton and Barto (1998) describe this method in detail.

Herein we will prove some of its properties, and then in Section 5.4, we experimentally show its effectiveness. We note that if all costs are equal, then $\texttt{AdaBoostRS}_{\texttt{AC}}$ becomes `AdaBoostRS`.

The following theorem shows that $\texttt{AdaBoostRS}_{\texttt{AC}}$ produces an unbiased estimate of the full `AdaBoost` vote, regardless of now many samples it uses.

Theorem 5. *Let $v_\tau(x)$ be the vote generated by $\texttt{AdaBoostRS}_{\texttt{AC}}$ as defined in Algorithm 3, then for all number of samples $\tau > 0$,*

$$\text{sign}\left(\mathbf{E}[(v_\tau(x))]\right) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Algorithm 3 $\texttt{AdaBoostRS}_{\texttt{AC}}$ (Arbitrary Cost)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$
where $x_i \in X$, $y_i \in Y = \{-1, +1\}$.
Given feature costs: $c_i, \ldots, c_n \geq 1$.
Generate $(\alpha_1, h_1), \ldots, (\alpha_T, h_T)$ using Algorithm 1.

Given: test example x, a bound B.
Initialize:

$$p(i) = \frac{\alpha_i}{c(h_i)\sum_{i=1}^{T}(\alpha_t / c(h_t))}$$

and $k = 0$, $F = \emptyset$, $\tau = 0$ and $v_0(x) = 0$.

while

$$k + \max_{1 \leq i \leq T} c(h_i) < B$$

do
  choose h_i according to $p(i)$
  let F_{h_i} be the set of features used by h_i
  let $l = F_{h_i} \setminus F$
  $k = k + \sum_{x^j \in l} c_j$
  $F = F \cup F_{h_i}$
  if $h_i(x) = 1$ **then**
    $v_{\tau+1}(x) = v_\tau(x) + c(h_i)$
  else
    $v_{\tau+1}(x) = v_\tau(x) - c(h_i)$.
  end if
  increment τ by 1
end while

Predict: $\text{sign}(v_\tau)$.

Proof. By definition

$$\begin{aligned}
\mathbf{E}[(v_\tau(x))] &= \sum_\tau \sum_i p(i) c(h_i) h_i(x) \\
&= \sum_\tau \left(\sum_i \frac{\alpha_i c(h_i) h_i(x)}{c(h_i)\sum_{i=1}^{T}(\alpha_t / c(h_t))} \right) \\
&= \frac{\tau \sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{i=1}^{T}(\alpha_t / c(h_t))}.
\end{aligned}$$

So we have

$$\text{sign}\left(\mathbf{E}[(v_\tau(x))]\right) = \text{sign}\left(\frac{\tau \sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{i=1}^{T}(\alpha_t / c(h_t))}\right).$$

Observing that

$$\frac{\tau}{\sum_{i=1}^{T}(\alpha_t / c(h_t))} > 0$$

completes the proof. □

We conclude this section with a proposition similar to Proposition 4.

Proposition 6. *In the limit of the number of samples taken, the resulting classification of* `AdaBoostRS`$_{AC}$ *is equal to the vote of its underlying boosting algorithm, as in: let* $v_\tau(x)$ *be the vote generated by* `AdaBoostRS`$_{AC}$ *as defined in Algorithm 3, then with probability* 1,

$$\lim_{\tau\to\infty} \text{sign}(v_\tau(x)) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

5. Experimental Results

In this section we experimentally test `AdaBoostRS` and `AdaBoostRS`$_{AC}$.

5.1. Data

We considered the following datasets: census, splice, ocr17, and ocr49, all available from the UCI repository. The splice dataset was modified to collapse the two splice categories into one to create binary-labeled data.

Table 1. Dataset sizes, and numbers of features, for training and test.

	census	splice	ocr 17	ocr 49
training	1000	1000	1000	1000
test	5000	2175	5000	5000
num features	130	239	402	402

Also, ocr17 and ocr49 contain randomly chosen subsets of the NIST database of handwritten digits consisting only of the digits 1 and 7, and 4 and 9 (respectively); in addition, the images have been scaled down to 14×14 pixels, each with only four intensity levels. Table 1 shows the number of training and test examples used in each.

5.2. Error Rates of `AdaBoost`

Because `AdaBoostRS` and `AdaBoostRS`$_{AC}$ both use `AdaBoost` as the underlying algorithm, in Table 2 we give the error rate of `AdaBoost`, with decision stumps as a base learner, using its full vote, when run for 500 rounds, which the number of rounds used in all our experiments. This table is meant to place the results in the following sections in context.

5.3. Results for `AdaBoostRS`

In this section, we experimentally test `AdaBoostRS`.

Corollary 3 predicts that the error rate of `AdaBoostRS` should fall exponentially in the number of samples it uses. This prediction is confirmed in Figures 1 and 2 (log plot).

Table 2. Error rates (in percent) of `AdaBoost`, run for 500 rounds with decision stumps as the base learner.

	census	splice	ocr 17	ocr 49
`AdaBoost`	18.3	8.1	0.8	6.5

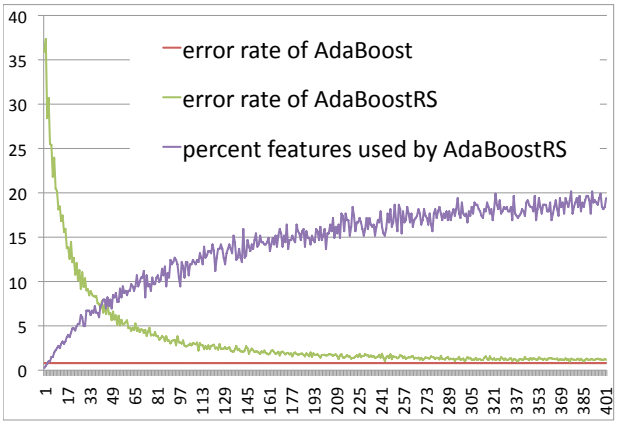

Figure 1. A graph of the error rate of `AdaBoostRS` on the ocr17 dataset and the percent of features it is using. The horizontal axis is the number of samples drawn by `AdaBoostRS`.

Figure 1 also makes clear that `AdaBoostRS`'s error rate quickly asymptotes to the error rate of `AdaBoost`. Furthermore, we can also see that the number of features used does not rise as quickly as the number of samples taken. This is behavior that we would expect from our analysis in Section 3.3, as features get resampled and only contribute to the budget upon their first examination.

Table 3. Number of samples needed to reach given relative error rates to `AdaBoost`. The numbers inside the parentheses are the error rates of `AdaBoost` (using the full vote) on the respective datasets.

	100%	50%	25%	10%
census (18.3)	15	77	211	637
splice (8.1)	97	205	421	823
ocr17 (0.8)	178	244	339	505
ocr49 (6.5)	167	296	694	1020

In Table 3 we examine the number of samples needed to achieve the given relative error rates compared to `AdaBoost`. Table 6 shows the percent of the features that were used to achieve the relative error rates of

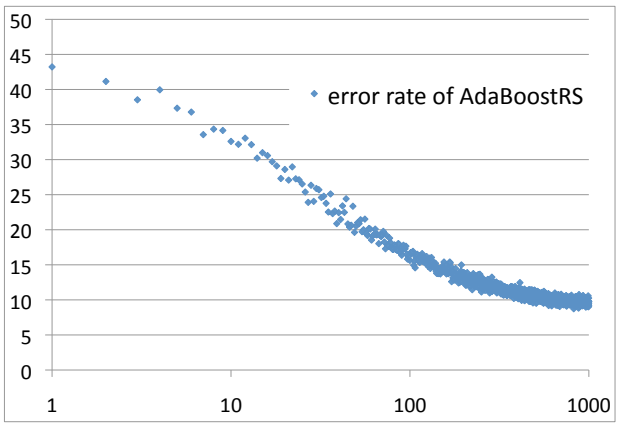

Figure 2. A graph of the error rate of `AdaBoostRS` on the splice dataset, as a function of the number of samples. The horizontal axis is on a log scale.

Table 4. Percent of features used to reach given relative error rates to `AdaBoost`. The numbers inside the parentheses are the error rates of `AdaBoost` (using the full vote) on the respective datasets.

	100%	50%	25%	10%
census (18.3)	10.0	29.2	40.8	48.5
splice (8.1)	26.8	38.9	52.7	61.5
ocr17 (0.8)	16.4	18.4	19.9	20.6
ocr49 (6.5)	21.6	26.4	31.3	32.3

`AdaBoost`. For these datasets, the α distribution is sufficiently lumpy (far for uniform) as to give `AdaBoostRS` good relative error rates to `AdaBoost` without using too many features, even when taking many samples.

There results confirm the effectiveness of the sampling procedure in `AdaBoostRS`.

5.4. Results for $\texttt{AdaBoostRS}_{\texttt{AC}}$

We also experimentally test the algorithm $\texttt{AdaBoostRS}_{\texttt{AC}}$ in the non-uniform cost setting. In our experiments, we assign each feature a cost chosen uniformly at random in the interval $[0, 1]$. We compare the performance of $\texttt{AdaBoostRS}_{\texttt{AC}}$ and `AdaBoostRS` in the non-uniform costs setting and show that our careful modification of the `AdaBoostRS` (to make $\texttt{AdaBoostRS}_{\texttt{AC}}$) yields practical improvement in performance. The results appear in Table 5, and we can clearly see that $\texttt{AdaBoostRS}_{\texttt{AC}}$ outperformes `AdaBoostRS` for the two different budget values.

One explanation of $\texttt{AdaBoostRS}_{\texttt{AC}}$'s superior performance can be found in Table 6. If we examine the number of samples taken, we note that the number of samples `AdaBoostRS` (without taking costs into account) is expected to take is slightly less than twice the budget. By not paying attention to feature costs, in expectation, it adds 0.5 to the total cost used on each round. $\texttt{AdaBoostRS}_{\texttt{AC}}$, meanwhile, is designed to pay attention to feature costs, and is therefore able to use significantly more samples without going over budget.

Table 5. Error rates (in percent), averaged over 50 trials, of $\texttt{AdaBoostRS}_{\texttt{AC}}$ and `AdaBoostRS` using budgets of 10 and 20 when features have random costs drawn i.i.d. from $[0, 1]$. The underlying algorithm, `AdaBoost`, is run for 500 rounds.

	$\texttt{AdaBoostRS}_{\texttt{AC}}$	`AdaBoostRS`
census ($B = 11$)	32.2	32.8
census ($B = 21$)	25.5	26.4
splice ($B = 11$)	25.7	27.0
splice ($B = 21$)	19.2	20.4
ocr17 ($B = 11$)	9.2	10.5
ocr17 ($B = 21$)	3.5	4.3
ocr49 ($B = 11$)	27.4	28.3
ocr49 ($B = 21$)	20.2	21.4

Table 6. Number of samples taken (τ), averaged over 50 trials, of $\texttt{AdaBoostRS}_{\texttt{AC}}$ and `AdaBoostRS` using budgets of 11 and 21 when features have random costs drawn i.i.d. from $[0, 1]$. The underlying algorithm, `AdaBoost`, is run for 500 rounds.

	$\texttt{AdaBoostRS}_{\texttt{AC}}$	`AdaBoostRS`
census ($B = 11$)	26.2	20.7
census ($B = 21$)	45.7	41.3
splice ($B = 11$)	33.8	20.6
splice ($B = 21$)	56.2	40.0
ocr17 ($B = 11$)	29.4	20.6
ocr17 ($B = 21$)	49.3	40.5
ocr49 ($B = 11$)	33.6	21.1
ocr49 ($B = 21$)	55.7	40.3

6. Discussion

In this work, we examined a boosting approach to tackling the feature-efficient learning problem. We leave room for many interesting problems including:

- What is the best boosting algorithm to use given this framework?
- We noticed, in experiments, that deterministically using the top-weighted (by α_i/c_i) hypotheses sometimes outperforms our approach, yet the former method is not known to be mathematically justified. It would be interesting to develop a better theoretical understanding that explains

precisely when our approach yields the best advantage over naive methods (and when it doesn't).

- Can we explicitly take feature costs into account in training?
- How can we modify other popular learning algorithms to make them more feature-efficient?

We conclude by noting that ever since the appearance of the margin bounds in (Schapire et al., 1998), it has been a natural question how well a hypothesis-sampling algorithm would perform. Our work addresses this question.

Acknowledgments

A preliminary version of this paper appeared in the ICML 2010 Budgeted Learning Workshop (Reyzin, 2010).

The author thanks Dana Angluin and Rob Schapire for helpful discussions and the anonymous reviewers of both this version and the workshop paper for useful suggestions. The author especially thanks a reviewer for catching a bug in one of the proofs in this paper.

This work is funded in part by a Simons Foundation Postdoctoral Fellowship. Parts of this work were done while the author was at Yahoo! Research, New York; the author also acknowledges that this material is based upon work supported by the National Science Foundation under Grant #0937060 to the Computing Research Association for the Computing Innovation Fellowship program.

References

Ben-David, Shai and Dichterman, Eli. Learning with restricted focus of attention. In *COLT*, pp. 287–296, New York, NY, USA, 1993. ACM.

Breiman, Leo. Prediction games and arcing classifiers. *Neural Computation*, 11(7):1493–1517, 1999.

Cesa-Bianchi, Nicolò, Shalev-Shwartz, Shai, and Shamir, Ohad. Efficient learning with partially observed attributes. *CoRR*, abs/1004.4421, 2010.

Chernoff, Herman. *Sequential Analysis and Optimal Design*. SIAM, 1972.

Freund, Yoav and Schapire, Robert E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55 (1):119–139, 1997.

Globerson, Amir and Roweis, Sam T. Nightmare at test time: robust learning by feature deletion. In *ICML*, pp. 353–360, 2006.

Greiner, Russell, Grove, Adam J., and Roth, Dan. Learning cost-sensitive active classifiers. *Artif. Intell.*, 139(2):137–174, 2002.

Grove, Adam J. and Schuurmans, Dale. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI/IAAI*, pp. 692–699, 1998.

Hoeffding, Wassily. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

Langford, John, Seeger, Matthias, and Megiddo, Nimrod. An improved predictive accuracy bound for averaging classifiers. In *ICML*, pp. 290–297, 2001.

Mason, Llew, Bartlett, Peter L., and Baxter, Jonathan. Direct optimization of margins improves generalization in combined classifiers. In *NIPS*, pp. 288–294, 1998.

Pelossof, Raphael, Jones, Michael, and Ying, Zhiliyang. Speeding-up margin based learning via stochastic curtailment. In *ICML/COLT Budgeted Learning Workshop*, Haifa, Israel, June 25 2010.

Reyzin, Lev. Boosting on a feature budget. In *ICML/COLT Budgeted Learning Workshop*, Haifa, Israel, June 25 2010.

Reyzin, Lev and Schapire, Robert E. How boosting the margin can also boost classifier complexity. In *ICML*, pp. 753–760, 2006.

Schapire, Robert E. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*. Springer, 2003.

Schapire, Robert E., Freund, Yoav, Bartlett, Peter, and Lee, Wee Sun. Boosting the margin: A new explanation for the effectiveness of voting methods. *the Annals of Statistics*, 26(5):1651–1686, 1998.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Wald, Abraham. *Sequential Analysis*. Wiley, 1947.

Wang, Liwei, Sugiyama, Masashi, Yang, Cheng, Zhou, Zhi-Hua, and Feng, Jufu. On the margin explanation of boosting algorithms. In *COLT*, pp. 479–490, 2008.