Training-Time Optimization of a Budgeted Booster

Yi Huang **Brian Powers** **Lev Reyzin**
Mathematics, Statistics, and Computer Science, University of Illinois at Chicago
{yhuang,bpower6,lreyzin}@math.uic.edu

Abstract

We consider the problem of feature efficient prediction – a setting where features have costs, and the learner is limited by a budget constraint on the total cost of the features it can examine in test time. We focus on solving this problem with boosting by optimizing the choice of base learners in the training phase and stopping the boosting process when the learner's budget runs out. We experimentally show that in the case of random costs, our method improves upon a previous approach of Reyzin [8] of drawing as many random samples as the budget allows from a trained `AdaBoost` ensemble.

1 Introduction

The problem of budgeted learning centers on questions around resource constraints imposed on a traditional supervised learning algorithm. Here, we focus on the setting where a learner has ample resources during training time, but is constrained by resources in predicting on new examples. In particular, we assume that accessing the features of new examples is costly (with each feature having its own cost to access it), and predictions must be made without running over a given budget. This budget may or may not be known to the learner. Learners that adhere to such budget constraints are sometimes called **feature-efficient**.

A classic motivation for this problem is the medical testing setting, where features correspond to the results of tests that are often costly or even dangerous to perform. Diagnoses often need to be made on incomplete information, and doctors must order tests thoughtfully in order to stay within whatever budgets the world imposes.

Here, we focus on boosting methods, in particular `AdaBoost`, to make them feature-efficient predictors. This line of work was started by Reyzin [8], who introduced the algorithm `AdaBoostRS`, a feature-efficient version of `AdaBoost`. While `AdaBoostRS` provably converges to the behavior of `AdaBoost` as the feature budget increased, it only considers feature costs and budget at test time. Reyzin left open the problem of whether optimizing during training can improve performance. Here, we answer this question with a resounding yes, giving algorithms that clearly outperform `AdaBoostRS`, especially when costs vary and budget limits are small.

Our approach relies mainly on two observations. The first is that when all features have equal costs stopping the training of `AdaBoost` early, once the budget runs out, will outperform `AdaBoostRS`. Second, when features have different costs, which is the setting that chiefly concerned Reyzin, one can still run `AdaBoost`, but choose weak learners as to better trade-off their cost vs. contribution to the performance of the ensemble. Combining these simple ideas yields our approach.

2 Past Work

Research on this problem goes back at least to Wald [10], who considered the problem of running a clinical trial sequentially, only testing future patients if the validity of the hypothesis in question is still sufficiently uncertain. This question belongs to the broader area of sequential analysis [3].

Ben-David and Dichterman [1] examined the learning theory behind learning using random partial information from examples and discussed conditions for learning in their model. Greiner et al. [5] also 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.

In related settings, Globerson and Roweis [4] looked at building robust predictors that are resilient to corrupted or missing features. Cesa-Bianchi et al. [2] studied how to efficiently learn a linear predictor in the setting where the learner can access only a few features per example. And He et al. [6] trained an MDP for this task, casting it as dynamic feature selection. In the area of boosting, Pelossof et al. [7] analyzed how to speed up margin-based learning algorithms by stopping evaluation when the outcome is close to certain. Sun and Zhou [9] also considered how to order base learner evaluations so as to save prediction time.

However, our main motivation is the recent work of Reyzin [8], who tackled the feature-efficient learning problem using ensemble predictors. He showed that sampling from a weights distribution of an ensemble yields a budgeted learner with similar properties to the original ensemble, and he tested this idea experimentally on `AdaBoost`. The goal of this paper is to improve on Reyzin's approach by incorporating the feature budget into the training phase.

3 AdaBoost and Feature Selection

Our goal in this paper is to produce an accurate classifier given a budget B and a set of m training examples, each with n features and each feature with a cost via cost function $C : [n] \to \mathbb{R}^+$. Reyzin's `AdaBoostRS` [8] takes the approach of ignoring feature cost during training, and then randomly selecting hypotheses from ensemble produced by `AdaBoost` until the budget is reached. Here we look at a different approach—to optimize the cost efficiency of boosting during training, so the ensemble classifier that results is both relatively accurate and affordable.

One straightforward approach is to run `AdaBoost`, paying for the features of the weak learners chosen every round, bookkeeping expenditures and the features used, until we cannot afford to continue. In this case, we are simply stopping `AdaBoost` early. We call this algorithm the "basic" `AdaBoostBT` for Budgeted Training. Surprisingly, this albeit simple methodology produces results that are significantly better than `AdaBoostRS` for both features with a uniform cost and features with random cost across a plethora of datasets.

Algorithm 1 `AdaBoostBT(S,B,C)`, where: $S \subset X \times \{-1, +1\}$, $B > 0$, $C : [i \dots n] \to \mathbb{R}^+$

1: given: $(x_1, y_1), ..., (x_m, y_m) \in S$
2: initialize $D_1(i) = \frac{1}{m}$, $B_1 = B$
3: **for** $t = 1, \dots, T$ **do**
4: train base learner using distribution D_t.
5: get $h_t \in \mathcal{H} : X \to \{-1, +1\}$.
6: **if** the total cost of the unpaid features of h_t exceeds B_t **then**
7: set $T = t - 1$ and **end for**
8: **else** set B_{t+1} as B_t minus the total cost of the unpaid features of h_t, marking them as paid
9: choose $\alpha_t = \frac{1}{2}\ln\frac{1+\gamma_t}{1-\gamma_t}$, where $\gamma_t = \sum_i D_t(i) y_i h_t(x_i)$.
10: update $D_{t+1}(i) = D_t(i)\exp(\alpha_t y_i h_t(x_i))/Z_t$, where Z_t is the normalization factor
11: **end for**
12: output the final classifier $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

We note that, in `AdaBoost`, since training error is upper bounded by $\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} \sqrt{1 - \gamma_t^2}$, at each round of boosting one typically greedily chooses the base learner that minimizes the quantity, which is equivalent to choosing the weak learner that maximizes γ_t. One can simply choose h_t in step 5 of `AdaBoostBT` according to this rule, which amounts to stopping `AdaBoost` early if its budget runs out. As we show in Section 4, this already yields an improvement over `AdaBoostRS`.

However, this approach is clearly suboptimal when costs are not uniform. Namely, it may sometimes be better to choose a worse-performing hypothesis if its cost is lower. Doing so may hurt the

algorithm on that current round, but allow it to afford to boost for longer, more than compensating for the locally suboptimal choice.

The problem is that it is difficult to known exactly how many future rounds of boosting can be afforded under most strategies. Hence, we can make the assumption that for a base learner that costs c, we could afford B_t/c additional rounds of boosting under the simplified assumption that all future rounds will incur the same cost and achieve the same γ_t as in the current round. In this case, minimizing $\prod_{t=1}^{T} Z_t$ is equivalent to minimizing the quantity

$$h_t = \text{argmin}_{h \in \mathcal{H}} \left((1 - \gamma_t(h)^2)^{\frac{1}{c(h)}} \right), \tag{1}$$

where $\gamma_t(h) = \sum_i D_t(i) y_i h(x_i)$. and $c(h)$ is the cost of the features used by h. This is our first proposed criteria for modifying base learner selection in step 5 of `AdaBoostBT`.

There is a potential pitfall with this approach: if we mark every used feature down to cost 0 (since we don't re-pay for features), then the optimization will collapse since every base learner with cost 0 will be favored over all other base learners no matter how uninformative it is. We can obviate this problem by considering the original cost during the selection, but not paying for used features again while updating B_t, as is done in our Algorithm.

As optimizing according to Equation 1 makes a very aggressive assumption of future costs, we consider a smoother optimization for our second approach. If in round t we were to select h_t with cost c, the average cost per round thus far is clearly $\frac{(B-B_t)+c}{t}$. Our second approach uses this average cost t to estimate the number of additional rounds we are going to run. Specifically, in step 5 of `AdaBoostBT`, we select a base learner according to

$$h_t = \text{argmin}_{h \in \mathcal{H}} \left(\left(1 - \gamma_t(h)^2\right)^{\frac{1}{(B-B_t)+c(h)}} \right). \tag{2}$$

4 Experimental Results and Discussion

For our experiments, we used data-sets from the UCI repository, as shown in Table 1. The features and labels were collapsed into binary categories.

data set	ocr17	ocr49	sonar	census	splice	ecoli	breast cancer	heart	ionosphere
num features	403	403	11196	131	240	356	82	371	8114
training size	1000	1000	100	1000	1000	200	500	100	300
test size	5000	5000	108	5000	2175	136	199	170	51

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

Experimental results, given in Figure 1, compare average generalization error rates over 20 trials, each with a random selection of training examples. Features are given costs uniformly at random on the interval $[0, 2]$. As a benchmark, `AdaBoost` was run for 500 rounds irrespective of budget.

The most apparent conclusion from our experiments is that it is not only possible to improve upon `AdaBoostRS` by optimizing base learner selection during training, but the that improvement is dramatic. Further modifications of the basic `AdaBoostBT` tend to yield additional improvements.

Optimizing `AdaBoostBT` according to Equation 1 often tends to perform better than the basic `AdaBoostBT` for small budgets, but it chooses base learners quite aggressively - a low cost base learner is extremely attractive at all rounds of boosting. This makes it possible that the algorithm falls into a trap, as in the sonar and ionosphere data sets, where we have a huge number of features (consequently, many features with cost close to zero). After 500 rounds of boosting, this approach sill had not spent the budget of 5 because the same set of cheap features were re-used round after round leading to a deficient classifier. Similar behavior is seen for the ecoli and heart datasets.

Using Equation 2 in `AdaBoostBT` avoids this trap by considering the average cost instead. The appeal of cheap base learners is dampened as the boosting round increases, with its limiting behavior to choose weak learners that maximize γ. Thus, we can see that using Equation 2, while tending to perform worse than Equation 2 for low budgets, tends to exceed its accuracy for larger budgets.

One unintended drawback of both Equations 1 and 2 is that the selection of base learners by considering cost typically allows for many more rounds of boosting than `AdaBoostBT`. The final classifier

may suffer from over-fitting to the training data, and this can be seen in ecoli, breast cancer and heart data sets where budgeted training algorithms outperform `AdaBoost` itself.

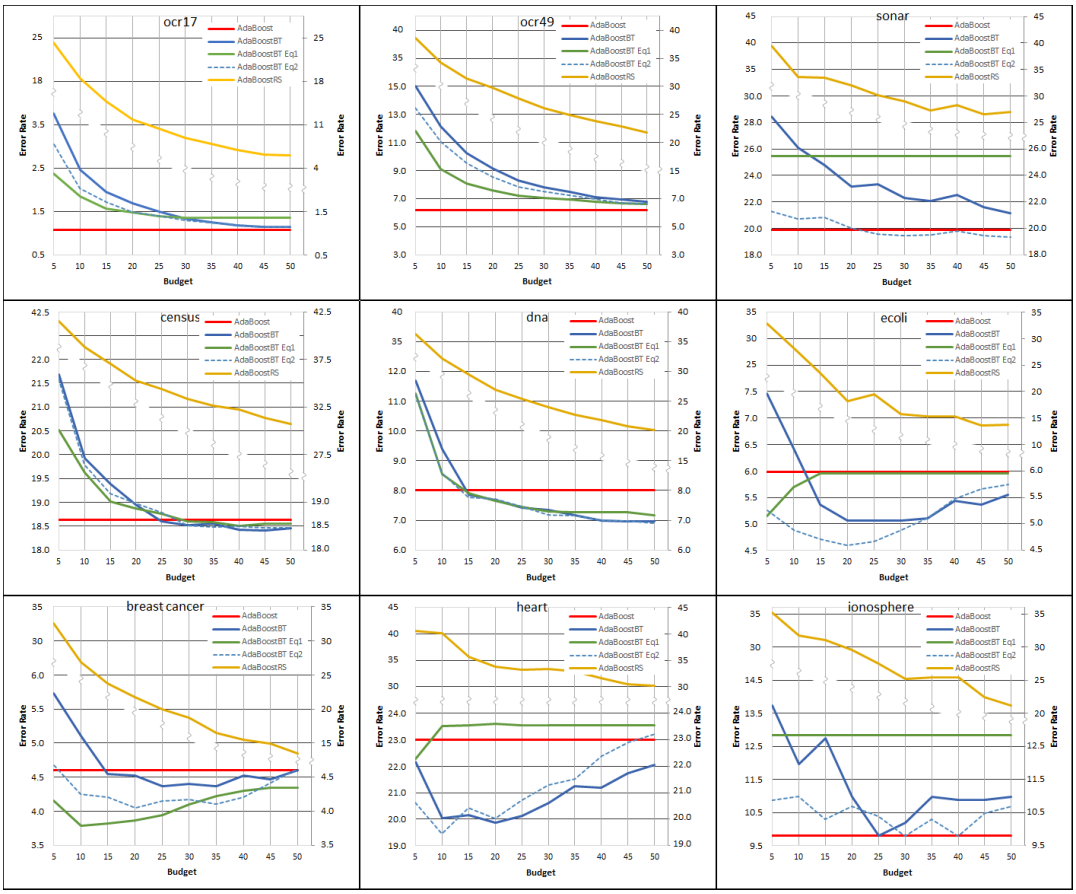

Figure 1: Experimental results comparing our approaches to `AdaBoostRS`using 500 rounds of boosting. Vertical scales are adjusted to make `AdaBoostRS` visible.

References

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

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

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

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

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

[6] He He, Hal Daumé III, and Jason Eisner. Imitation learning by coaching. In *NIPS*, pages 3158–3166, 2012.

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

[8] Lev Reyzin. Boosting on a budget: Sampling for feature-efficient prediction. In *ICML*, pages 529–536, 2011.

[9] Peng Sun and Jie Zhou. Saving evaluation time for the decision function in boosting: Representation and reordering base learner. In *ICML*, 2013.

[10] Abraham Wald. *Sequential Analysis*. Wiley, 1947.