Training-Time Optimization of a Budgeted Booster

Lev Reyzin
UIC, Math Dept

work joint with Yi Huang and Brian Powers

Budgeted Learning

- Looking at **features** may be expensive.
  - medical diagnosis
  - Internet applications
- In many applications, this is especially true during **prediction**, more so than in training.
- Still want to predict as accurately as possible.

Model

- Goal is to do supervised learning, using a limited number of features in test-time.
  - Given a **budget on total cost**: on each example, the learner must look at no more features than allowed by the budget.
  - Each feature has an associated cost.
  - Budget only **limited in test** data, not training.

- I will refer to predictors that do this as **feature-efficient**.

Lots of work on this problem

- **Sequential analysis**: when to stop sequential clinical trials.
  - Wald ('47) and Chernoff ('72)
  - Pelossof et al. ('10) speed up margin-based algorithms
- **PAC learning** analysis with incomplete features.
  - Ben-David and Dichterman ('93) and also Greiner et al. ('02)
- Robust prediction with **corrupted/missing features**.
  - Globerson and Roweis ('06)
- Learning **linear hypotheses** without using many features
  - Cesa-Bianchi et al. ('10)
- Incorporating feature costs into CART **impurity** function
  - Xu et al ('12)
- **MDPs** for feature selection
  - He et al ('13)

One Previous Idea [R ’11]

- An ensemble is a weighted vote of many simple rules.

- The simple rules a usually feature-efficient.

- take a vote of only a few rules.

AdaBoostRS

Train AdaBoost (or your favorite ensemble).

Then, to predict:

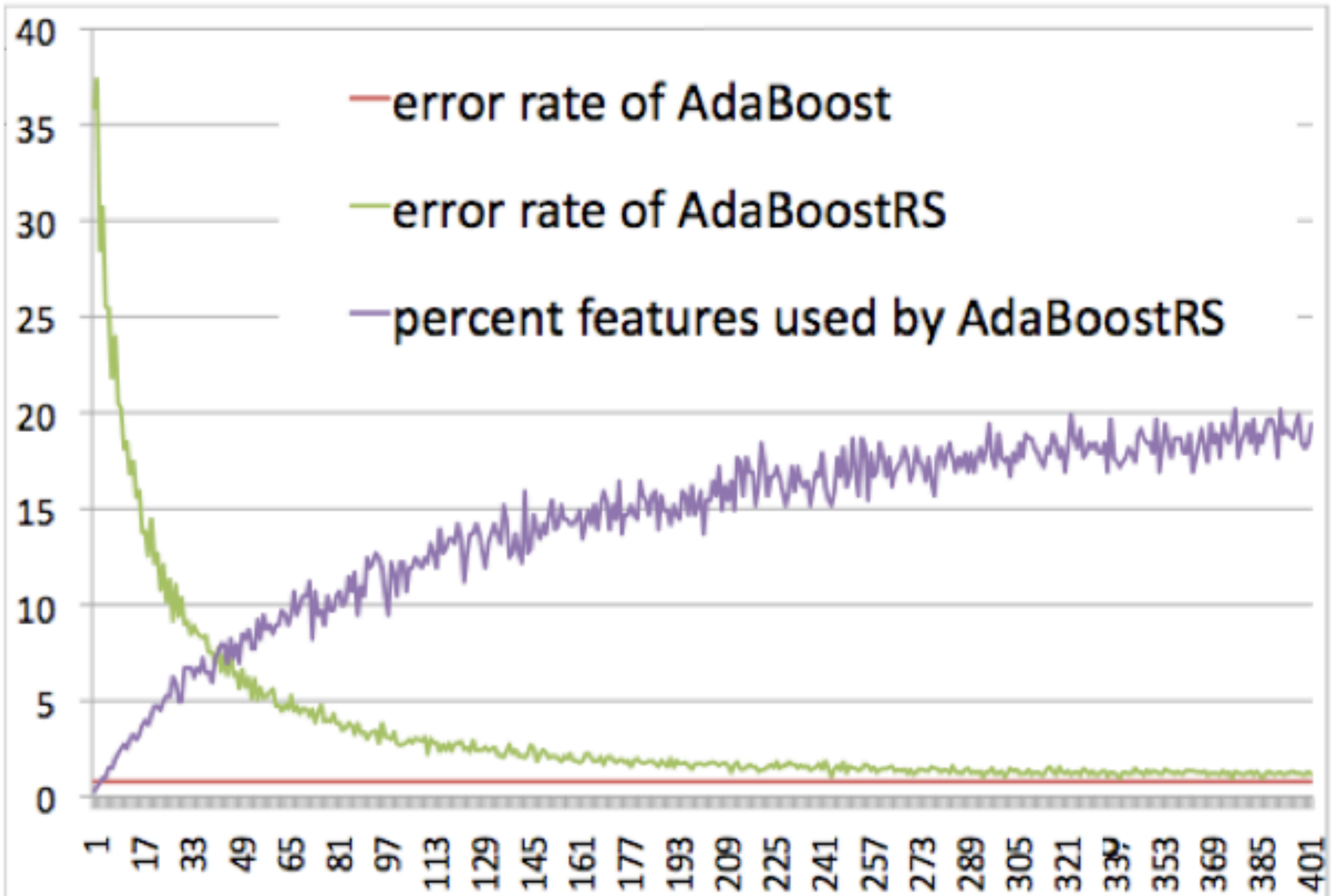
1. Sample the weak learners according to their ensemble weights and feature costs.
2. Take a vote of the sampled weak learners (weighing so that the final vote is an unbiased estimate of the full vote)

Intuition on why this works:

If ensemble has strong preference, then sampling will converge fast. If ensemble is split, who cares?

(margin bound [Schapire et al ’98])

Experiments with AdaBoostRS [R '11]



On ocr17 dataset. x-axis is number of samples taken.

Room for Improvement

These ideas only optimize after the ensemble is built. Can we improve on AdaBoostRS by moving the optimization into the training phase?

Turns out: yes, by a lot! [Huang-Powers-R '14]

- Naïve idea: stop training AdaBoost when Budget is out
- Improvement: choose weak learners more wisely

`AdaBoost (S )` where: $S \subset X \times \{-1, +1\}$

given: $(x_1, y_1), \ldots, (x_m, y_m) \in S$
initialize $D_1(i) = \frac{1}{m}$
for $t = 1, \ldots, T$ **do**
    train base learner using distribution D_t.
    get $h_t \in \mathcal{H} : X \to \{-1, +1\}$.

    choose $\alpha_t = \frac{1}{2} \ln \frac{1+\gamma_t}{1-\gamma_t}$, where $\gamma_t = \sum_i D_t(i) y_i h_t(x_i)$.
    update $D_{t+1}(i) = D_t(i) \exp(\alpha_t y_i h_t(x_i)) / Z_t$,
end for
output the final classifier $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

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

1: given: $(x_1, y_1), \ldots, (x_m, y_m) \in S$
2: initialize $D_1(i) = \frac{1}{m}$, $B_1 = B$
3: **for** $t = 1, \ldots, T$ **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 $D_{t+1}(i) = D_t(i)\exp(\alpha_t y_i h_t(x_i))/Z_t$,
11: **end for**
12: output the final classifier $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

How to choose h_t?

Training error of AdaBoost is bounded by [Freund & Schapire '97]

$$\hat{\Pr}[H(x) \neq y] \leq \prod_{t=1}^{T} \sqrt{1 - \gamma_t^2}$$

When budgets are an issue, we need to optimize two effects:

a) high edges make individual terms smaller
b) low costs allow for more terms in the product

Two Minimizations

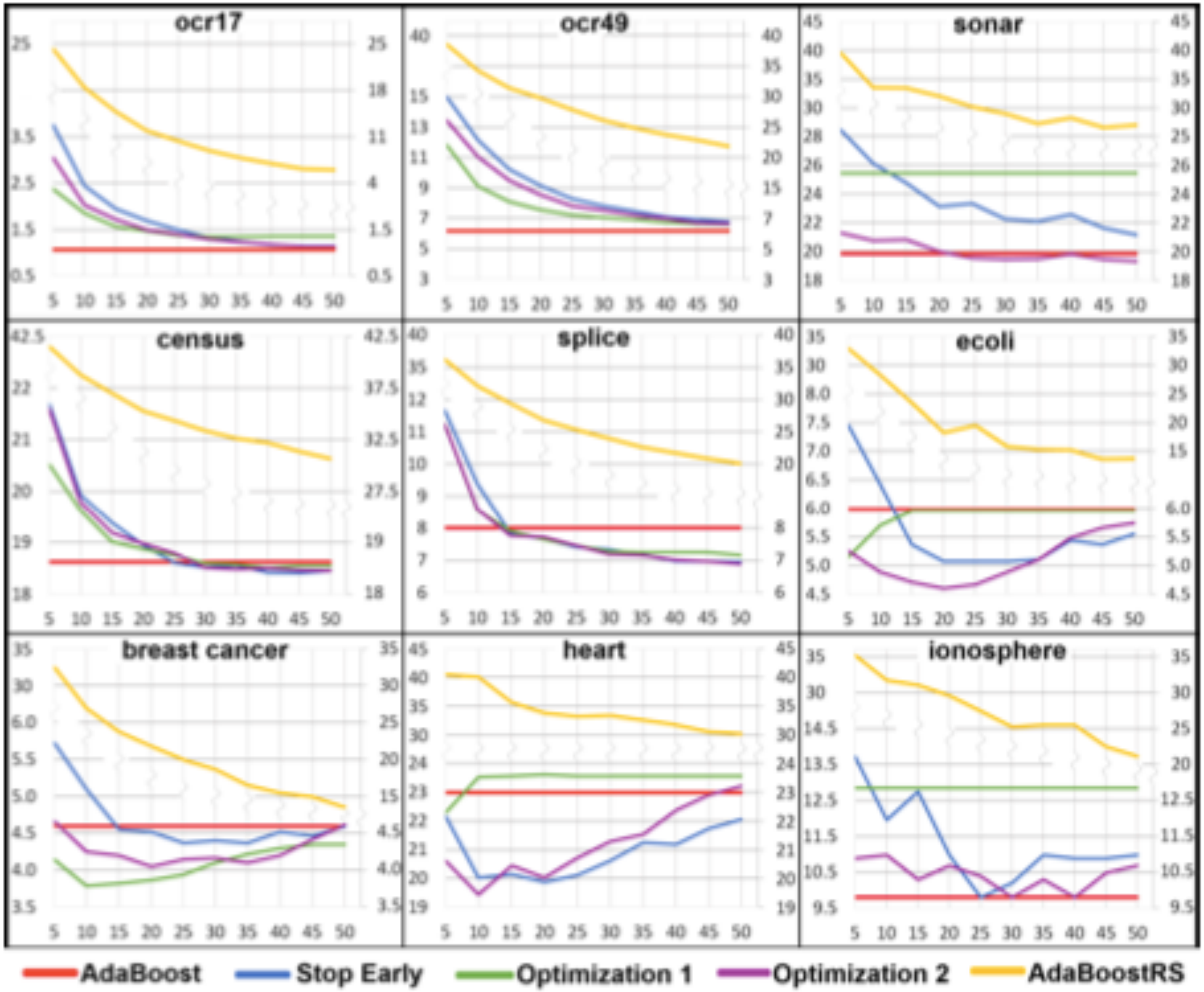
First idea: assume all future rounds will behave like current. Leads to optimization

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

Second idea: smoothed version of first.

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

Experiments with costs ~ U(0,2)



Discussion and Future Work

- Moving budget optimization into the training phase really improves performance!
- Still need to consider adversarial cost models and test on real data (that has feature costs).
- Using confidence-rated predictions may help.