Training-Time Optimization of a Budgeted Booster

Yi Huang *Brian Powers Lev Reyzin

University of Illinois at Chicago

{*yhuang,bpower6,lreyzin*}*@math.uic.edu*

July 30, 2015

Motivation: Making Predictions with a Budget

We must classify a test example but can't afford to know all the facts.

Features may be **costly** to observe

- Time,
- Money,
- Energy,
- Health risk

Motivating scenarios:

- Medical diagnosis,
- Internet applications,
- Mobile devices

Feature-Efficient Learners

Goal: Supervised Learning Algorithm with:

- Budget $B > 0$
- Feature costs $C : [i, \ldots, n] \to \mathbb{R}^+$
- Limited by budget at test time

We call such a learner **feature-efficient**.

A Sampling of Related Work

- **Sequential analysis**: When to stop sequential clinical trials
  [Wald 47] and [Chernoff '72]
- **PAC learning** with incomplete features
  [Ben-David-Dichterman '93] and [Greiner et al. '02]
- Robust prediction with **missing features**
  [Globerson-Roweis '06]
- Learning **linear functions** by few features
  [Cesa-Bianchi et al. '10]
- Incorporating **feature costs** in CART impurity
  [Xu et al. '12]
- **MDPs** for feature selection [He et al. '13]

Idea: A Feature-Efficient Boosting Algorithm

An approach using Random Sampling [Reyzin '11]:

1. Run AdaBoost to produce an ensemble predictor.
2. Sample from ensemble randomly until budget is reached.
3. Take importance-weighted average vote of samples.

Performance converges to that of AdaBoost as $B \to \infty$...

But **is there room for improvement?**

Budgeted Training

Yes!

"Budgeted Training" uses the following principles:

- Use the budget to optimize training.
- Stop training early when budget runs out.
  - The resulting predictor will be feature-efficient.
- Modify base learner selection when costs are non-uniform.

Algorithm: AdaBoost

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

```
given: (x_1, y_1), ..., (x_m, y_m) ∈ S
initialize D_1(i) = 1/m
for t = 1, ..., T do
    train base learner using distribution D_t.
    get h_t ∈ H : X → {-1, +1}.
    choose α_t = (1/2) ln((1+γ_t)/(1-γ_t)), where γ_t = Σ_i D_t(i) y_i h_t(x_i).
    update D_{t+1}(i) = D_t(i) exp(α_t y_i h_t(x_i)) / Z_t,
end for
output the final classifier H(x) = sign(Σ_{t=1}^T α_t h_t(x))
```

given: $(x_1, y_1), ..., (x_m, y_m) \in S$

initialize $D_1(i) = \frac{1}{m}$

for $t = 1, \ldots, T$ **do**

  train base learner using distribution D_t.

  get $h_t \in \mathcal{H} : X \rightarrow \{-1,+1\}$.

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

Algorithm: AdaBoost with Budgeted Training

`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 $\boxed{h_t \in \mathcal{H} : X \to \{-1, +1\}}$.
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 $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Selection of Weak Learners

In AdaBoost, weak learners are selected to drive down the training error bound [Freund & Schapire '97]

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

- If costs are uniform (T is known), choose the weak learner that maximizes $|\gamma_t|$.
- If costs are non-uniform:
  - High edges give smaller terms, but
  - Low costs allow for more terms in the product.
  - **How should we trade-off edge vs cost?**

A Greedy Optimization

To estimate T we assume **future rounds will be like the current**.

So $T = \frac{B}{c(h)}$.

Then the selection becomes

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

A Smoother Optimization

Alternate estimate of T based on milder assumption: **The cost of future rounds will be the average cost so far.**

The resulting selection rule is

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

Idea: Using average cost should produce a smoother optimization.

A Look at SpeedBoost

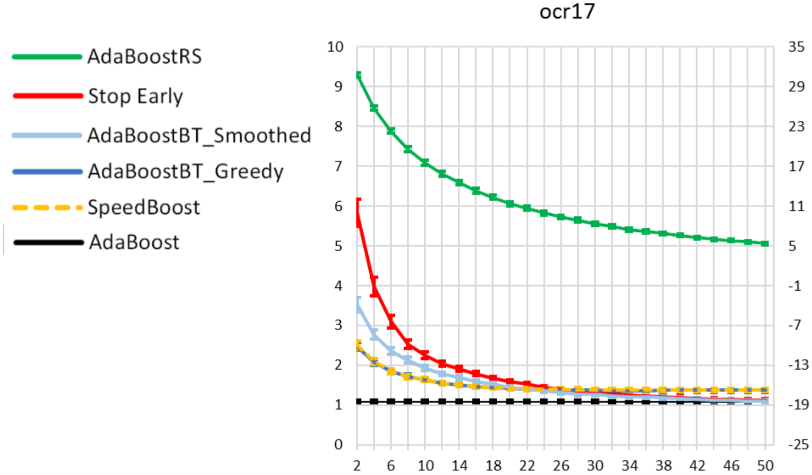
SpeedBoost [Grubb-Bagnell '12] produces a feature-efficient ensemble in another way.

An objective R is chosen (e.g. a loss function).

While the budget allows:
A Weak learner h and weight α are chosen to maximize

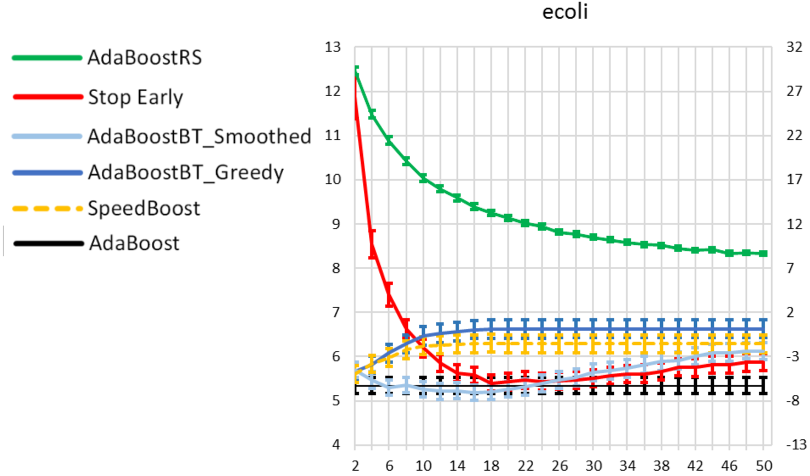
$$\frac{R(f_{i-1}) - R(f_{i-1} + \alpha h)}{c(h)}.$$

Experimental Results: $C \sim Unif(0,2)$



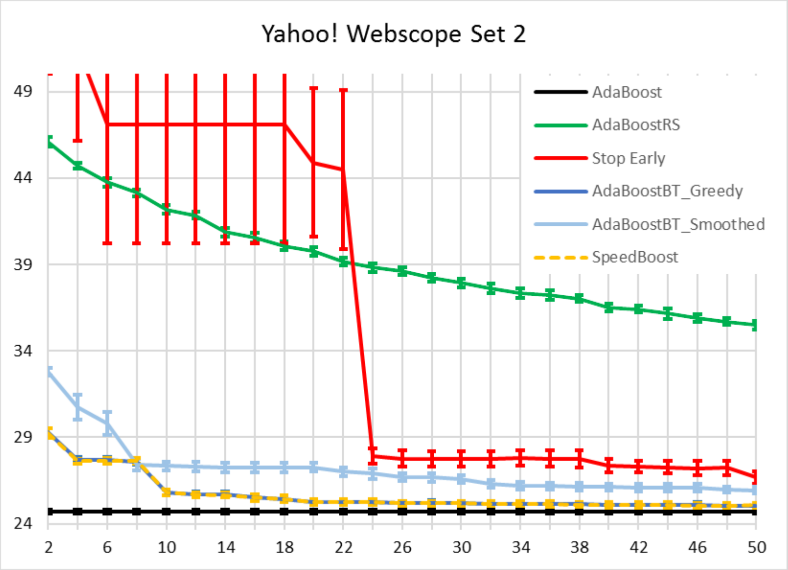
Budget on horizontal axis, test error rate on vertical (AdaBoostRS error on right). AdaBoost at T=400 as a benchmark.

Experimental Results: $C \sim Unif(0,2)$



Budget on horizontal axis, test error rate on vertical (AdaBoostRS error on right). AdaBoost at T=400 as a benchmark.

Experimental Results: Real World Data



Observations

- Budgeted training improves significantly on `AdaBoostRS`.
- Modifying with Greedy and Smoothed optimizations tend to yield additional improvements:
  - Greedy tends to win for small budgets.
  - Smoothed tends to win for larger budgets.
- SpeedBoost and our Greedy Budgeted Training perform almost identically.
  - There is an explanation using a Taylor series expansion.

Observations

- Too many cheap features can kill Greedy Optimization.
- Smoothed optimization avoids this trap, since cost becomes less important as $t \to \infty$.
- Both Greedy and Smoothed optimizations run a higher risk of over-fitting than simply stopping early.

Future Work

- Improve optimization for cost distributions with few cheap features.
- Consider adversarial cost models.
- Refine optimizations by considering the complexity term in AdaBoost's generalization error bound.
- Study making other machine learning algorithms feature-efficient through budgeted training.

Thank you

Thank you!

Visit my poster at Panel 4