OPT2019: 11th Annual Workshop on Optimization for Machine Learning

Accelerating boosting via accelerated greedy coordinate descent

Xiaomeng Ju * XIAOMENG.JU@STAT.UBC.CA
University of British Columbia

Yifan Sun * YIFAN.0.SUN@GMAIL.COM
INRIA-Paris

Sharan Vaswani * VASWANIS@MILA.QUEBEC
Mila, Université de Montréal

Mark Schmidt SCHMIDTM@CS.UBC.CA
University of British Columbia

Abstract

We exploit the connection between boosting and greedy coordinate optimization to produce new accelerated boosting methods. Specifically, we look at increasing block sizes, better selection rules, and momentum-type acceleration. Numerical results show training convergence gains over several data sets. The code is made publicly available.

1. Introduction

Boosting [26] is a technique for sequentially building complex machine learning models as a linear combination of weak learners. Specifically, at each iteration, a weak learner is trained on a proxy loss function over data that is constantly reweighted to emphasize badly fitted training samples. Despite its simplicity, the success of boosting methods on tabular data [6] makes it a ubiquitous and highly relevant tool in data science.

When run for a long time, boosting methods suffer computational costs from fitting and retaining a large number of weak learners. This motivates *accelerating* the convergence of simple boosting methods. Traditionally, such techniques included subsampling data or features randomly [9, 13] or greedily [5], or improving the weak learners [6, 14].

Another avenue is to exploit the close connection between boosting and greedy coordinate descent (GCD) methods. In particular, it has previously been observed that Adaboost [11] is equivalent to coordinate minimization under the Gauss-Southwell rule [17, 19, 25], and LogitBoost [10] is inspired by a greedy selection rule of a Taylor approximation of the loss function. Similar observations have also been made about gradient boosted machines [12, 20]. This has inspired the development of several "accelerated" boosting schemes, such as MultiBoost [16], GradBoost [8], LPBoost [7], and Accelerated Gradient Boosting (AGB) [2].

This work explores the practical impact of GCD-inspired acceleration techniques on a generalized boosting framework. The exact connection is reviewed in Section 2, and acceleration is presented in 3 and demonstrated numerically in 4. Although more work is required to show theoretical speedup, in practice we often observe faster training error convergence.

2. Boosting and Gauss Southwell coordinate minimization

Consider the empirical risk minimization problem for binary classification

* Equal Contribution
Code is made available at https://github.com/xmengju/Accelerated_Boosting

Adaboost	$t = \underset{t'=1,\dots,T}{\operatorname{argmax}} \left\lvert \nabla f(w^{(t-1)}) \right\rvert_{t'}$
LogitBoost	$t = \underset{t'=1,\dots,T}{\operatorname{argmin}} \; (\nabla f(w^{(t-1)}))_{t'} + \frac{1}{2}(\nabla^2 f(w^{(t-1)}))_{t',t'}$
Gradient Boosting	$t = \underset{t'=1,\dots,T}{\operatorname{argmin}} \; (\nabla f_{\text{reg}}(w^{(t-1)}))_{t'}$ where $f_{\text{reg}}(w) = f(w) + \frac{1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T} w_t (h^{(t)}(x_i))^2$

Table 1: Explicit update rules of boosting schemes, which inspire our coordinate selection rules when increasing block sizes. Proofs are given in Appendix A.

$$\underset{w\in\mathbb{R}^T,\, h^{(t)}\in\mathcal{H}}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i \bar{h}(x_i)) \qquad \text{subject to} \quad \bar{h}(x) = \sum_{t=1}^{T} w_t h^{(t)}(x)$$

where $\{x_i \in \mathbb{R}^d, y_i \in \{\pm 1\}\}_{i=1}^N$ are the training data and labels, and $\mathcal{L} : \mathbb{R} \to \mathbb{R}$ is some convex smooth loss function acting over the sample *margin* (binary label $\times$ soft prediction). Here, $\mathcal{H}$ is a class of weak learners (e.g. linear combinations of decision stumps [12], neural networks [28], or other types of base functions [4, 27].

Boosting algorithms for classification form the classifier $\bar{h} \in \mathbf{span}(\mathcal{H})$ sequentially, where at iteration t, we first train $h^{(t)}$ to minimize the weighted misclassification error, and then update w_t optimally. In Adaboost, the data weight is exactly the negative gradient of the margin loss for that sample, e.g.

$$f_i(w^{(t)}) = \mathcal{L}\left(y_i \sum_{t'=1}^{t} w_{t'} h^{(t')}(x_i)\right), \quad \mathcal{L}(\theta) = \exp(-\theta), \quad f(w) = \frac{1}{N}\sum_{i=1}^{N} f_i(w)$$

$$\Rightarrow \frac{\partial}{\partial w_\tau} f_i(w^{(t)}) = \underbrace{-\frac{\partial \mathcal{L}(\theta)}{\partial \theta}}_{\text{sample weight } d_i^{(t)}} \cdot \underbrace{-\frac{\partial \theta}{\partial w_\tau}}_{\propto \text{ misclass. error}}, \qquad \theta = y_i \sum_{t'=1}^{t} w_{t'} h^{(t')}(x_i).$$

(Adaboost, which usually minimizes the exponential loss, has $\mathcal{L}(\theta) = \exp(-\theta) = -\mathcal{L}'(\theta)$, so this weighting corresponds to multiplicitively punishing each data sample for its current loss.) Therefore,

$$h^{(t)} = \underset{h\in\mathcal{H}}{\operatorname{argmin}} \, d_i^{(t)} \mathbf{1}_{h(x_i)\neq y_i} \quad \Rightarrow \quad t = \underset{t=1,\dots,T}{\operatorname{argmin}} \, \tfrac{\partial}{\partial w_\tau} f(w^{(t)}) \qquad h^{(t)} \in \mathcal{H} \text{ for } \textit{all } t, \text{ past or future}$$

$$\Rightarrow \quad \underbrace{t = \underset{t=1,\dots,T}{\operatorname{argmax}} \left| \frac{\partial}{\partial w_\tau} f(w^{(t)}) \right|}_{\text{Gauss Southwell selection rule}} \qquad \text{if } \mathcal{H} \text{ is closed under sign.}$$

Similar connections have been observed for LogitBoost and Gradient boosting machines. (See also [3, 25].) We list a full summary in Table 1 and refer to Appendix A for derivations.

3. Improvements

Block size Block coordinate updates can often offer speedup over their coordinate counterparts. However, in boosting, increasing to $K > 1$ introduces several complications. First, the update of the weak learner weight may not be closed form, as it is in LogitBoost and Adaboost. Second, without retraining weak learners, there is no guarantee that the new coordinates selected represent the largest gradient magnitude value in the past *and* future, as we only have access to the weak learners of the past.

However, in the regime where training weak learners accounts for the bulk of the computational burden, taking several gradient steps in order to fine tune past weights may be worth it, especially since aggregating fewer weak learners provides both storage and generalization advantages. MultiBoost [16] employs fast coordinate descent with a mix of sequential (cyclic) and stochastic (random) coordinate selection rules. In the extreme case of LPBoost [7] and TotalBoost [29], *all* past weights are perfectly fitted in each step in a fully corrective way.

3.1. Coordinate selection rule

It has been previously shown [23] that the coordinate selection *rule* can also affect convergence rates. Inspired by the connection of LogitBoost and Adaboost to coordinate-wise Taylor approximations of the loss function, we explore these modified selection rules as well.

1. **Largest gradient magnitude (GMAX).** Equivalent to the Gauss-Southwell update rule.
2. **Smallest gradient value (GMIN).** In practice, if $\mathcal{H}$ is not closed under sign, then Adaboost is not exactly equivalent to the Gauss-Southwell update. We explicitly force this rule to evaluate any performance differences with the previous scheme.
3. **Smallest second-order Taylor approximation (GHMIN).** This scheme is inspired by LogitBoost and picks the coordinate that minimizes the coordinate-wise Taylor expansion; we extend this to general loss functions.
4. **Random selection.** Uniform selection over weights w_τ for $\tau = 1, ..., t-1$.

3.2. Accelerating coordinate descent

Several works present fast boosting methods developed based on accelerated optimization schemes. Conjugate gradient boosting (CGBoost) [15] accelerates GBM by performing conjugate gradient descent in the function space. The works of Mukherjee et al. [21], Biau et al. [1], and Lu et al. [18] incorporate Nesterov's acceleration techniques into the design of boosting algorithms. A convergence rate has been proved for accelerated gradient boosting by [18]. We explore both the impact of Nesterov [22] and Polyak acceleration [24], as they are most straightforward to implement on general boosting schemes.

4. Experiments

In all experiments, we use decision stumps as weak learners, and use 10 gradient steps at each iteration, with a Armijo-Wolfe line search to determine the steplength. We use the abbreviations for each update rule given in the previous section, appended by the block size, e.g. 5-ghmin means that, aside from updating the latest weight, we update 5 additional weights according to the GHMIN selection rule.

Figs. 1 and 2 compare the different acceleration schemes on random data, and Fig. 3 on several real datasets. For classification, overall, we found these techniques most effective when using the exponential loss function; when logistic loss is used, both the training and testing error simply converge better for all methods, almost equally, suggesting that there is something sub-optimal about the exponential loss function

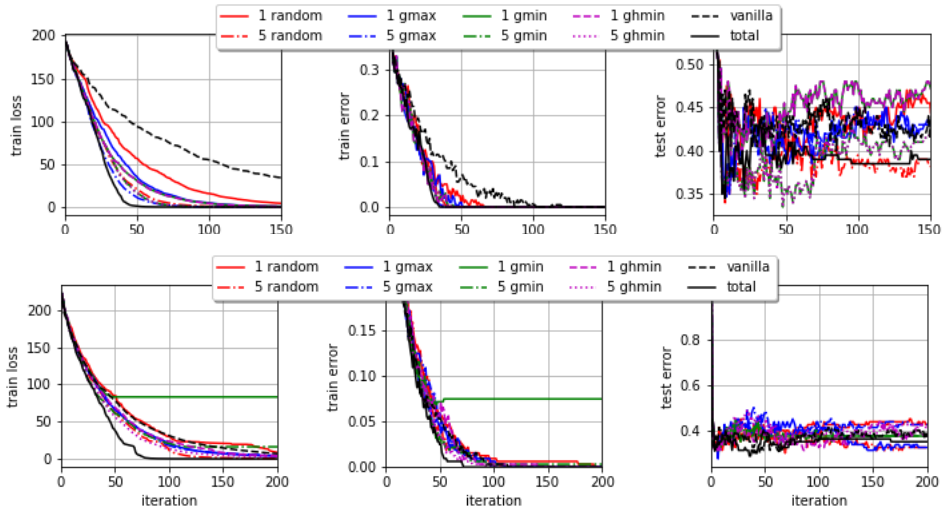

Figure 1: **Classification on random data.** Boosting via exponential (top) and logistic (bottom) loss.

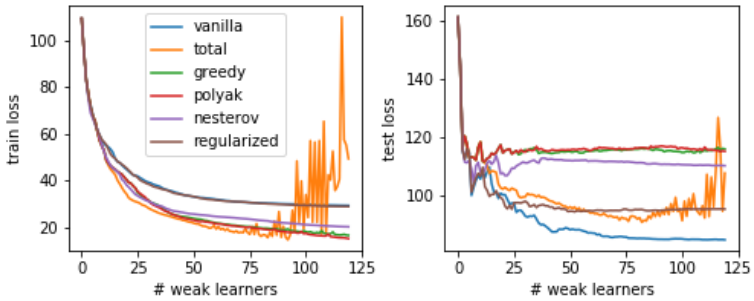

Figure 2: **Acceleration and regularization.** Boosting on random data via LAD loss (regression). Except for vanilla and total, all run with 5-GMAX selection rule. For l2 loss under Nesterov acceleration, numerical errors caused the method to terminate early.

itself. More consistent improvements are observed in regression problems; however, the accelerated convergence in training error is not always carried out to the test error and improvements in optimization do not always translate to improvements in generalization.

Fig. 4 compares the average runtime of boosting over 100 iterations across 10 random data trials. It is hard to compete against the vanilla algorithm, suggesting that this method is more advantageous when more computationally burdensome weak learners are used. Additionally, while we don't see much variance across different update rules, different loss functions have variable runtimes as bad conditioning causes the line search to slow down.

5. Future work

We believe that finding a scheme that improves performance in test error as much as train error is essential for practical usage. Additionally, we would like to extend our results to weak learners similar to those used in XGBoost [6] and LightGBM [14] to improve the performance of state-of-the-art boosting methods. Finally, we would like to give improved convergence rates under these acceleration schemes.

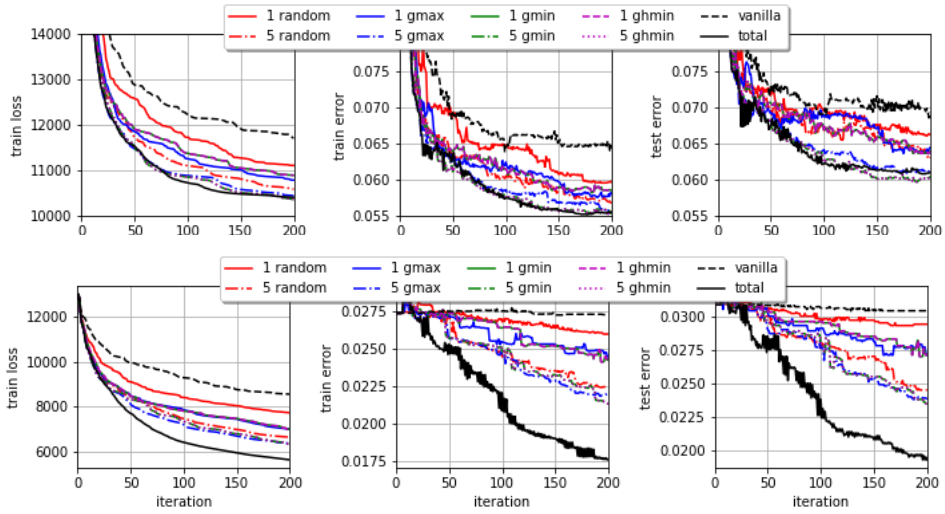

Figure 3: **Classification on real data.** Boosting via exponential (top) and logistic (bottom) loss.

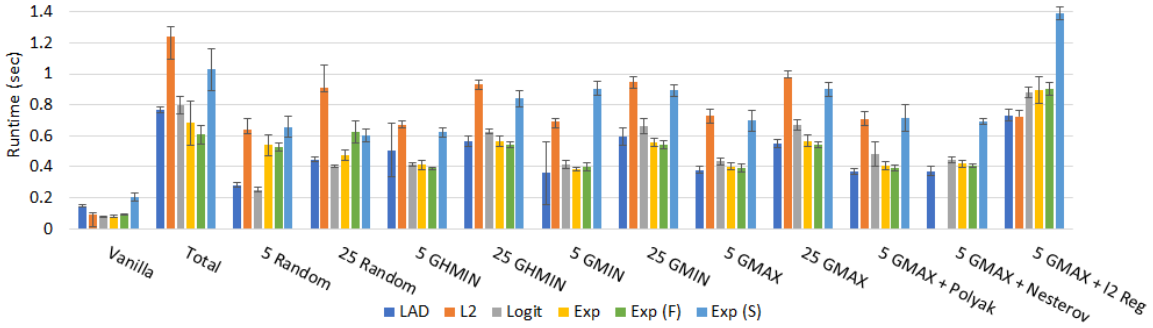

Figure 4: **Runtime comparison.** Total runtime after 100 iterations, comparing different methods. In the first four experiments, there are 100 samples and 50 features. In Exp (F), the number of features is doubled, and in Exp (S), the number of data samples is doubled.

References

[1] Gérard Biau, Benoît Cadre, and Laurent Rouvière. Accelerated gradient boosting. *Machine Learning*, pages 1–22, 2018.

[2] Gérard Biau, Benoît Cadre, and Laurent Rouvière. Accelerated gradient boosting. *Machine Learning*, 108(6):971–992, 2019.

[3] Peter J Bickel, Ya'acov Ritov, and Alon Zakai. Some theory for generalized boosting algorithms. *Journal of Machine Learning Research*, 7(May):705–732, 2006.

[4] Peter Bühlmann and Bin Yu. Boosting with the l 2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

[5] Róbert Busa-Fekete and Balázs Kégl. Bandit-aided boosting. In *OPT 2009: 2nd NIPS Workshop on Optimization for Machine Learning*, 2009.

[6] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[7] Ayhan Demiriz, Kristin P Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

[8] John Duchi and Yoram Singer. Boosting with structural sparsity. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 297–304. ACM, 2009.

[9] Gerard Escudero, Lluís Màrquez, and German Rigau. Boosting applied to word sense disambiguation. In *European Conference on Machine Learning*, pages 129–141. Springer, 2000.

[10] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Citeseer, 1996.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 28(2):337–407, 2000.

[12] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[13] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4): 367–378, 2002.

[14] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[15] Ling Li, Yaser S Abu-Mostafa, and Amrit Pratap. Cgboost: Conjugate gradient in function space. 2003.

[16] Guosheng Lin, Chunhua Shen, Anton van den Hengel, and David Suter. Fast training of effective multi-class boosting using coordinate descent optimization. In *Asian Conference on Computer Vision*, pages 782–795. Springer, 2012.

[17] Francesco Locatello, Anant Raj, Sai Praneeth Karimireddy, Gunnar Rätsch, Bernhard Schölkopf, Sebastian Stich, and Martin Jaggi. On matching pursuit and coordinate descent. In *International Conference on Machine Learning*, pages 3204–3213, 2018.

[18] Haihao Lu, Sai Praneeth Karimireddy, Natalia Ponomareva, and Vahab Mirrokni. Accelerating gradient boosting machine. *arXiv preprint arXiv:1903.08708*, 2019.

[19] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000.

[20] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In *Advanced lectures on machine learning*, pages 118–183. Springer, 2003.

[21] Indraneel Mukherjee, Kevin Canini, Rafael Frongillo, and Yoram Singer. Parallel boosting with momentum. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer, 2013.

[22] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[23] Julie Nutini, Issam Laradji, and Mark Schmidt. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv preprint arXiv:1712.08859*, 2017.

[24] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[25] Gunnar Rätsch, Sebastian Mika, and Manfred K Warmuth. On the convergence of leveraging. In *Advances in Neural Information Processing Systems*, pages 487–494, 2002.

[26] Robert E Schapire, Yoav Freund, Peter Bartlett, Wee Sun Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

[27] Matthias Schmid and Torsten Hothorn. Boosting additive models using component-wise p-splines. *Computational Statistics & Data Analysis*, 53(2):298–311, 2008.

[28] Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural computation*, 12(8):1869–1887, 2000.

[29] Manfred K Warmuth, Jun Liao, and Gunnar Rätsch. Totally corrective boosting algorithms that maximize the margin. In *Proceedings of the 23rd international conference on Machine learning*, pages 1001–1008. ACM, 2006.