Gradient Boosting Regression Trees with Variable Learning Rate Proposal

Austin Barket

Department of Computer Science
The Pennsylvania State University at Harrisburg
Middletown, PA 17057
amb6470@psu.edu

1 Introduction

Boosting is a general machine learning technique that combines learning algorithms that barely beat random guessing, known as a weak or base learners, into a single model with significantly improved accuracy or lower error rates over any of its constituent parts [1] [2].

The gradient boosting machine, originally introduced by Friedman in 1999 is a general boosting framework that leverages the steepest descent numerical optimization method at its core to iteratively train base learners to address the errors made by those before them [3].

Production implementations of gradient boosting machines such as the gbm package in R have found remarkable traction among researchers in a wide variety of fields including robotics and ecology [4] [5] .Interestingly these packages generally implement Friedman's Gradient Boost algorithm as it was originally defined [6], leaving some definite opportunity for research into algorithmic tweaks to improve performance.

A particular component of the algorithm that has not been explored to date is the learning rate, also referred to as shrinkage, which is implemented as a constant parameter to the model. After each iteration the new base learner's prediction is scaled by this parameter as a form of regularization.

This proposal outlines a new way to think about shrinkage for the specialized case where the base learners are regression trees. We hypothesize that by varying the learning rate applied to the prediction of the examples in each individual leaf of the regression tree base learners, we will be able to decrease convergence time without sacrificing resilience to overfitting.

2 Related Work

Boosting finds its roots in a question originally posed by Kearns and Valient in 1988, is weak learnability equivalent to strong learnability[7] [8]? That is if you have a way to learn well enough to beat random guessing, is it inherently true that a strong learner, capable of arbitrarily low error, for that same problem exists? Schapire successfully proved this equivalence in 1990 by proposing and

proving the correctness of a polynomial time boosting model he termed *The hypothesis boosting mechanism* [1].

After Schapire's compelling proof that weak and strong learnability are in fact equivalent, researchers bagan working to improve upon his boosting algorithm. The first practical application of the early boosting algorithms came out of the work of Drucker, Schapire, and Simard at AT&T Bell Labs in 1992. There they applied boosting of neural network base learners to the problem of optical character recognition of handwritten zip codes on USPS letters [9].

In 1995 Freund and Schapire introduced the AdaBoost algorithm which is hailed as solving many of the practical problems suffered by previous boosting algorithms. The unique idea introduced by Adaboost is the notion of applying higher weights at each iteration to the training examples that were misclassified in previous iterations, forcing the new base learners to focus their efforts on these examples. AdaBoost became famous as an excellent out of the bag approach for classification with exceptional resilience to overfitting [10]. However, the details of why exactly AdaBoost worked were unknown until the year 1998 when Friedman, Hastie, and Tibshirani explored the algorithm from an in depth statistical viewpoint. They found that AdaBoost a specialized additive model, and applied knowledge from the long history of statistical additive function approximation to gain a better understanding of AdaBoost and boosting in general [11].

With an increased theoretical statistical understanding of boosting now available, Friedman developed a generalized stagewise additive function approximation boosting model termed the gradient boosting machine in 1999, which he later extended to include a stochastic subsampling approach in 2002 [3] [12]. Gradient Boosting Machines will be explored is great detail in the following sections as extending this model is the focus of the proposed research.

Since their introduction to the machine learning and data mining communities in 1999, gradient boosting machines have found applications in a variety of fields for both classification and regression tasks. Most recently ecology researchers have found great interest in gradient boosting machines, particularly the varient of them that utilizes classification or regression trees as base learners. In 2007, Glenn De'ath extended the R package gbm, creating a new package gbmplus that implements a varient algorithm he terms Aggregated Boosted Trees (ABT). The idea behind ABTs is to perform cross validation to determine an optimal number of iterations for the boosting, then save the models built during cross validation chopping them off at the optimal number of iterations found. To make a prediction, the predictions of all of these boosted trees are computed then averaged. It was found that this approach lead to improved accuracy over gbm alone [13].

Another group of ecological researchers Jane Elith and John Leathwick have also been applying boosted regression trees to their work. One such problem involves predicting whether or not a particular species off eel will be present in unsampled Australian rivers based upon measured environmental factors [5]. Elith and Leathwick implemented their own extensions to the functions in the gbm package in their dismo package in 2015 [14].

3 Gradient Boosting Machine

In machine learning, our goal is often to find an approximation $\hat{F}$ of an unknown function $F : \boldsymbol{x} \rightarrow y$ that maps data instances $\boldsymbol{x}$ to a set of response variables y and best minimizes the expected value of some loss function $\Psi(y, F(\boldsymbol{x}))$. Friedman's gradient boosting machine iteratively constructs a strong learner that approximates F. In each iteration a new weak learner, such as a short regression tree, $h(\boldsymbol{x})$ is trained to fit the errors made by the function approximation so far. This training is based upon an extremely common numerical minimization method known as steepest gradient descent [3] [6]. However, unlike most applications of steepest descent, Friedman's Gradient Boost algorithm computes the negative gradient $\boldsymbol{g}$ in the space of the estimated function itself, not in the space of a finite parameters that define the function. By framing the problem in this way, the function $\hat{F}$ is not limited to a set of functions definable by a finite set of parameters, but rather is defined by a potentially infinite set of parameters, one for each possible value $\boldsymbol{x}$. Obviously, it is impossible to actually compute the gradient and apply steepest-descent in this potentially infinite dimensional function space, but it is possible to perform steepest-descent with respect to the finite space of training examples D [3] [6].

The negative gradient in this restricted subset of function space defines the direction of steepest descent in the loss function for the training examples. Thus by updating the function $\hat{F}$ by this negative gradient, we would move closer to the minimum values of the loss function Ψ for the examples in the training dataset. Of course this is not quite the goal, instead we would like to be able to generalize to all possible data. To accomplish this we train a regression model to predict the negative gradient of the loss function at each step, then update our function with this model's prediction, instead of the value of the negative gradient itself. Friedman's general Gradient Boost algorithm, extended to include his later ideas of subsampling the training data and applying a constant learning rate to improve generalization is provided in Algorithm 1 [3] [12] [6].

input : Training Dataset: $D = (x_i, y_i)$, $i = 1...N$
Bag Fraction: $bf \, \epsilon \, [0, 1]$
Learning Rate: $v \, \epsilon \, [0, 1]$
Number of Base Learners: M
Loss Function: Ψ
Choice of Base Learner: $h(\boldsymbol{x})$ e.g. regression trees

output: A function $\hat{F}(\boldsymbol{x})$ that minimizes the expected value of $\Psi(y, F(\boldsymbol{x}))$

Initialize the approximation of $\hat{F}$

$$\hat{F}_0(\boldsymbol{x}) = argmin_\rho \sum_{i=1}^{N} \Psi(y_i, \rho) \tag{1}$$

for $m \leftarrow 1$ *to* M **do**

Select a random subsample S_m of training data without replacement.

$$S_m \subset D, |S_m| = \tilde{N} = bf \cdot N \tag{2}$$

Approximate the negative gradient $\boldsymbol{g}_m$ of $\Psi(y_i, \hat{F}_{m-1}(\boldsymbol{x}))$ with respect to $\hat{F}_{m-1}(\boldsymbol{x})$.

$$g_{m,i} = -\frac{\partial}{\partial \hat{F}_{m-1}(\boldsymbol{x_i})} \Psi(y_i, \hat{F}_{m-1}(\boldsymbol{x_i})), \; \boldsymbol{x}_i \, \epsilon \, S_m \tag{3}$$

Train a new base learner $h_m(\boldsymbol{x})$ to predict $\boldsymbol{g}_m$ and fit the least squares.

$$\beta_m, h_m(\boldsymbol{x}) = argmin_{\beta, h(\boldsymbol{x})} \sum_{\boldsymbol{x}_i \, \epsilon \, S_m} [g_{m,i} - \beta h(\boldsymbol{x}_i)]^2 \tag{4}$$

Solve for the optimal coefficient ρ that minimizes Ψ. Note: if Ψ is the least squares loss function, $\rho = \beta$ but they may differ for other loss functions.

$$p_m = argmin_\rho \sum_{\boldsymbol{x}_i \, \epsilon \, S_m} \Psi(y_i, \hat{F}_{m-1}(\boldsymbol{x}_i) + \rho h(\boldsymbol{x}_i)) \tag{5}$$

Update your approximation of $\hat{F}$, scaled by the learning rate v

$$\hat{F}_m(\boldsymbol{x}) = \hat{F}_{m-1}(\boldsymbol{x}) + v \cdot \rho_m h_m(\boldsymbol{x}) \tag{6}$$

end

Algorithm 1: Friedman's Gradient Boost Algorithm [3] [4] [12] [13]

4 Proposed Work

4.1 Overview

The goal of the this project is to explore the effect of variable learning rates on gradient boosting machines that utilize regression trees as the base learners. Until now all research and implementations of gradient boosting machines have used only constant learning rates as in Algorithm 1. The conventional wisdom has been to use small learning rates of 0.01 or lower as this always seems to lead to high accuracy models with a low risk of overfitting. However this comes at the cost of increased computation time because more base learners must be trained [6].

We believe that variable learning rates provide an unexploited area of research and hypothesize that an intelligent method of adapting the learning rate can improve the convergence speed without compromising the model's resilience to overfitting. As we will formulate in detail below, the use of regression trees as the base learners presents an interesting possibility of a simple, yet elegant adaptation method. Since the regression trees themselves can be seen as a summation of individual prediction terms, one for each leaf in the tree, a natural adaptation scheme is to apply a different learning rate to the predictions made by each leaf in the tree. Specifically, the lower the number of examples in a given leaf node, the lower its learning rate ought to be to discourage overfitting to the examples. On the contrary, a leaf node containing a large number of training examples poses less danger to the generalization ability of the overall learner, and thus it should be safe to apply relatively high learning rates to these leaves.

4.2 Details

In order to discuss the proposed learning rate adaptation scheme in more detail, we will first specialize Algorithm 1 to use regression trees as base learners.

Regression trees with J leaves will be represented with the following notation.

$$h_m(\boldsymbol{x}) = \sum_{j=1}^{J} b_{m,j} I(\boldsymbol{x} \,\epsilon\, R_{m,j}) \tag{7}$$

Where

J = the number of terminal nodes (leaves) in the tree

$b_{m,j}$ = Prediction made for all instances in $R_{m,j}$.
Dependent on the loss function used. E.g. for squared error,
$b_{m,j} = avg_{x_i \epsilon R_{m,j}}(g_{m,i})$

$R_{m,j}$ = The subset of instances $\boldsymbol{x} \,\epsilon\, S_m$
that are predicted by the j^{th} terminal node.

$$I(\alpha) = \begin{cases} 1 & \alpha \text{ is true} \\ 0 & \alpha \text{ is false} \end{cases}$$

Thus, for the case where the base learners are regression trees with J leaves, the update step (Equation 6) becomes

$$\hat{F}_m(\boldsymbol{x}) = \hat{F}_{m-1}(\boldsymbol{x}) + v \cdot \rho_m \cdot \sum_{j=1}^{J} b_{m,j} I(\boldsymbol{x} \, \epsilon \, R_{m,j}) \tag{8}$$

As mentioned in the overview section, the main goal of this project is to examine the effects of variable learning rates on the gradient boost algorithm. Specifically, we will alter Algorithm 1 to take as input a maximum learning rate v_{max} instead of the constant learning rate v. The following equation will then be used to compute the learning rate for each of the J leaves in Equation 8.

$$v_{m,j} = v_{max} \cdot \frac{|R_{m,j}|}{|S_m|} \tag{9}$$

Note that the regions in regression trees are disjoint by definition. Thus $\sum_{j=1}^{J} |R_{m,j}| = |S_m|$. So we are essentially weighting each leaf's contribution to $\hat{F}$ by the ratio of examples predicted by that leaf. The hypothesis being that as this ratio increases, the generalization quality of that leaf's prediction also increases and the risk of overfitting to those training examples diminishes.

Using equation 9, the update step of Algorithm 1 becomes

$$\hat{F}_m(\boldsymbol{x}) = \hat{F}_{m-1}(\boldsymbol{x}) + \rho_m \sum_{j=1}^{J} v_{m,j} \cdot b_{m,j} I(\boldsymbol{x} \, \epsilon \, R_{m,j}) \tag{10}$$

It's important to note that the usefulness and impact of this adaptation scheme is dependent upon the assumption that the number of examples predicted by the leaves in the regression trees vary sufficiently. If the trees tend to partition the training data evenly, then the proposed adaptation scheme will be approximately the same as just having a constant learning rate. If this proves to be the case one way in which we can increase the impact of the learning rate adaption, would be to allow the number of leaves J to vary from iteration to iteration. Perhaps by randomly selecting J from a small range of values. By varying J we will get shorter and taller trees which will result in different size partitions of the training data and thus different learning rates from tree to tree.

Varying J across iterations could have a positive impact on the model performance of its own. A key factor in the performance of boosting models to have the individual base learners focus on different aspects of the training data [12]. This is why AdaBoost weights its misclassified examples [10], why gradient boosting models fit the errors of previous iterations, and why Friedman's idea of subsampling the data at each iteration, which is essentially a form of bagging or bootstrapping found in other ensemble methods, is so crucial to the performance of the algorithm [12]. We hypothesize that by varying J, the individual

base learners will view the training data at different levels of granularity, and as a result the concepts that they focus on should differ, this increased diversity among base learners may lead to an increase in model performance.

Time permitting, it may also be interesting to consider alternative adaptation schemes or even hybrids between them. For example we may consider starting with a high learning rate, then allowing it to diminish as the iterations go on. Note that such a scheme is not in conflict with the per leaf adaption mentioned above, and the two schemes could be used together, where we begin with a large maximum learning rate, and allow that maximum to vary as the iterations go on.

We plan to modify an existing gradient boosting machine implementation, most likely the gbm or gbmplus R package. There are many benefits to extending one of these packages, chief among them is the built-in plotting and analysis functions that will be readily available to help compare the original implementation's performance against one with variable learning rates. In addition, some of the datasets and R code for the ecological experiments mentioned earlier are available for download [13] [5] [14]. Using gbm will make it very easy to compare our results with those found in this prior research on the standard gbm model.

In the event that gbm proves to be too formidable to extend due to the large number of use cases the package is designed to support, we will extend a simpler implementation that is more directly focused on Friedman's original implementation. Two such options are the JSGBM Java implementation or the Apache Spark implementation written in Scala. If this is the case, additional work will be required to find numeric datasets suitable for use with these implementations.

After finishing the implementation a significant portion of this project will entail parameter tuning and model analysis to determine the impact of the variable learning rates and any other algorithmic tweaks made. For these purposes we will most likely employ analysis similar to that seen in [5].

References

1. R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, Jul. 1990. [Online]. Available: http://dx.doi.org/10.1023/A:1022648800760
2. ——, "The boosting approach to machine learning: An overview," in *Nonlinear Estimation and Classification*, ser. Lecture Notes in Statistics, D. Denison, M. Hansen, C. Holmes, B. Mallick, and B. Yu, Eds. Springer New York, 2003, vol. 171, pp. 149–171.
3. J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. pp. 1189–1232, 2001. [Online]. Available: http://www.jstor.org/stable/2699986
4. A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, p. 21, 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/
5. J. Elith, J. R. Leathwick, and T. Hastie, "A working guide to boosted regression trees," *Journal of Animal Ecology*, vol. 77, no. 4, pp. 802–813, 2008. [Online]. Available: http://dx.doi.org/10.1111/j.1365-2656.2008.01390.x
6. G. Ridgeway, "Generalized boosted models: A guide to the gbm package," *Update*, vol. 1, no. 1, p. 14, 2012.
7. M. Kearns, "Thoughts on hypothesis boosting," *Unpublished manuscript (Machine Learning class project, December 1988)*, 1988.
8. M. Kearns and L. G. Valiant, "Crytographic limitations on learning boolean formulae and finite automata," in *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '89. New York, NY, USA: ACM, 1989, pp. 433–444. [Online]. Available: http://doi.acm.org/10.1145/73007.73049
9. H. Drucker, R. E. Schapire, and P. Simard, "Improving performance in neural networks using a boosting algorithm," in *Advances in Neural Information Processing Systems 5, [NIPS Conference]*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 42–49. [Online]. Available: http://dl.acm.org/citation.cfm?id=645753.668055
10. Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory*, ser. Lecture Notes in Computer Science, P. Vitnyi, Ed. Springer Berlin Heidelberg, 1995, vol. 904, pp. 23–37. [Online]. Available: http://dx.doi.org/10.1007/354059119-2_166
11. J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 04 2000. [Online]. Available: http://dx.doi.org/10.1214/aos/1016218223
12. J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics And Data Analysis*, vol. 38, no. 4, pp. pp 367–378, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167947301000652
13. G. De'Ath, "Boosted trees for ecological modeling and prediction," *Ecology*, vol. 88, no. 1, pp. 243–251, 2007.
14. J. Elith and J. Leathwick, "Boosted regression trees for ecological modeling," 2015.