Variable Learning Rate Gradient Boosting Regression Trees

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

Jeremy Blum
Department of Computer Science
The Pennsylvania State University at Harrisburg
Middletown, PA 17057
Email: jjb24@psu.edu

***Abstract*—In this research we 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 [1].**

I. Introduction

Boosting is a 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 [2] [3].

The gradient boosting machine, originally introduced by Friedman in 1999 is a general boosting framework that leverages the steepest descent numerical optimization method to iteratively train base learners to address the errors made by those before them [4].

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 [5] [6]. Interestingly these packages generally implement Friedman's Gradient Boost algorithm as it was originally defined [1], 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 predictions are scaled by this parameter as a form of regularization.

This proposal outlines a new way to think about shrinkage for the common 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.

Revise

II. 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* [2].

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 [4] [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 [6]. Elith and Leathwick implemented their own extensions to the functions in the gbm package in their dismo package in 2015 [14].

III. The Gradient Boosting Machine

In supervised learning, our goal is to find an approximation $\hat{F}$ of an unknown function $F : \vec{x} \rightarrow y$ that maps data instances $\vec{x}$ to a set of response variables y and best minimizes the expected value of some loss function $\Psi(y, F(\vec{x}))$. Friedman's gradient boosting machine achieves this by iteratively constructing a strong learner that approximates F from many weak learners. In each iteration a new weak learner, such as a short regression tree, $h(\vec{x})$ 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 [1] [4] . However, unlike most applications of steepest descent, Friedman's Gradient Boost algorithm computes the negative gradient $\vec{g}$ in the space of the estimated function itself, not in the space of a finite set of 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 $\vec{x}$. 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 [1] [4].

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 $\hat{F}$ directly 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 instead 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. Friedman's general Gradient Boosting Machine, 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 [1] [4] [12].

---

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 e.g. RMSE
Choice of Base Learner: $h(\vec{x})$ e.g. regression trees

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

Initialize the approximation of $\hat{F}$

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

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

Select a random subsample S_m of training data without replacement.

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

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

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

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

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

Solve for the optimal coefficient ρ that minimizes Ψ.

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

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

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

end

Algorithm 1: Friedman's Gradient Boost Algorithm [4] [5] [12] [13]

IV. Simplifying Assumptions

For the purposes of the current research, we will consider only the case where Ψ is the squared error function and the goal is to predict a real valued response variable. Without

loss of generality we will add a coefficient of $\frac{1}{2}$ to Ψ so that the negative of the partial derivative of Ψ with respect to the predicted value of an instance $\vec{x}_i$ is simply the residual of that prediction. In this case the component wise calculation of the negative gradient (Equation 3) becomes Equation 7.

$$g_{m,i} = -\frac{\partial}{\partial \hat{F}_{m-1}(\vec{x_i})} \frac{(y_i - \hat{F}_{m-1}(\vec{x_i}))^2}{2} = (y_i - \hat{F}_{m-1}(\vec{x_i})) \tag{7}$$

As alluded to in the introduction, the proposed variable learning rate scheme is specially designed for the common case where the base learners are regression trees. When building the regression trees, we define the next best split as the split that leads to the largest reduction in squared error on the training examples. Thus the least squares coefficient β in Equation 4 will always be 1, as will the optimal coefficient ρ in Equations 5 and 6.

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

$$h_m(\vec{x}) = \sum_{j=1}^{J} b_{m,j} I(\vec{x} \, \epsilon \, R_{m,j}) \tag{8}$$

Where

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

$b_{m,j} =$ Prediction made for all instances in $R_{m,j}$.
For squared error, $b_{m,j} = avg_{x_i \epsilon R_{m,j}}(g_{m,i})$

$R_{m,j} =$ The subset of instances $\vec{x} \, \epsilon \, S_m$
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 and the loss function is the squared error, the update step (Equation 6) becomes

$$\hat{F}_m(\vec{x}) = \hat{F}_{m-1}(\vec{x}) + v \cdot \sum_{j=1}^{J} b_{m,j} I(\vec{x} \, \epsilon \, R_{m,j}) \tag{9}$$

The choice of regression tree base learners introduces two additional parameters to the gradient boosting algorithm, namely the maximum number of splits in each tree (aka interaction depth) and the minimum number of observations in each leaf node. These parameters work together to regularize the complexity of the regression trees, and in general these parameters are chosen to ensure short trees and stout leaves to weaken the predictive power of each individual base learner and strengthen the boosted model's ability to generalize [6].

Note that although we restricted the current study to regression tasks using the squared error loss function, the variable learning rate scheme defined and tested in the following sections could easily be applied to any of the loss functions originally defined by Friedman in [4] [12], and most famously implemented in the gbm R package originally written by Ridgeway [1]. Please see these references for information on the many specialized derivations of Algorithm 1 for various learning tasks and to better understand the reasoning behind the equations defined in this section.

V. Variable Learning Rates for Regression Tree Base Learners

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 compute a different learning rate for each leaf node in each base learner. Specifically a simple linear mapping will be used to ensure that the lower the number of examples in a given leaf node, the lower the computed learning rate. We hypothesize that this will encourage the training to learn rapidly from the most typical training examples, which will fall into large leaf nodes along with their similar peers, while diminishing the impact of outlying and noisy examples, which will be isolated by the regression tree's splitting algorithm. Overall we expect this scheme to lead to faster convergence time while strongly discouraging overfitting.

Specifically, we will alter Algorithm 1 to take as input both a minimum and maximum learning rate v_{min} and v_{max} respectively, 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 9.

$$v_{m,j} = \frac{|R_{m,j}|}{|S_m|}(v_{max} - v_{min}) + v_{min} \tag{10}$$

This equation maps the range of possible leaf node sizes $[0, |S_m|]$ to the range of possible learning rates $[v_{min}, v_{max}]$. Note that even when the regression trees are built with a minimum number of observations in each leaf node, a common regularization parameter, the missing value branches of the trees can still have any number of examples in them, including zero if no missing values existed in the training data. Thus zero is used as the minimum leaf size regardless of the minimum number of observations parameter.

Using equation 10, the update step (Equation 9) becomes...

$$\hat{F}_m(\vec{x}) = \hat{F}_{m-1}(\vec{x}) + \rho_m \sum_{j=1}^{J} v_{m,j} \cdot b_{m,j} I(\vec{x} \, \epsilon \, R_{m,j}) \tag{11}$$

VI. Experiment

First, we implemented the Friedman's gradient boosting machine algorithm as defined and simplified in sections III and IV in java.

VII. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

REFERENCES

[1] G. Ridgeway, "Generalized boosted models: A guide to the gbm package," *Update*, vol. 1, no. 1, p. 14, 2012.

[2] 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

[3] ——, "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.

[4] 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

[5] 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/

[6] 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

[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/10.1007/3\-540\-59119-2_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.