Blinkered Gradient Boosted Trees

John Hawkins

December 2017

Abstract

Gradient boosting is a machine learning technique for iteratively building a model by fitting the residuals of the existing model. In spite of its generality as a technique, there are multiple scenarios in which it may fail to adequately model a given task. One of these situations is the presence of excessive noise in the training data. In this paper we propose a technique that will mitigate the impact of noise under certain circumstances. By making several non-standard assumptions about the way noise is distributed in data, we are able to derive a simple variation of the gradient boosting algorithm. We conduct experiments with several public data sets and show that our approach can result in improved accuracy on some problems when compared with several common gradient boosting implementations. While the approach is not a universal improvement, due to its unique treatment of specific kinds of noise it represents a novel tool for machine learning practitioners.

1 Introduction

Gradient Boosting has proven itself to be one of the most significant advances in the field of supervised machine learning. Its general utility in providing accurate models is perhaps no better demonstrated than by the fact that it has come dominate the field of competitive machine learning (as explified by the competitions run by Kaggle) [4]. In spite of its success there remain a range of specific circumstances which limit the ability of gradient boosting (and other algorithms) to provide adequate reaults. One of these circumstances is the presence of noise in the data. Early boosting algorithms, like AdaBoost, were known to suffer in the presence of noise **??**. Although modern gradient boosting is less susceptible the issue remains because there is always potential for the residuals to become dominated by the presence of noise.

Many variations of the gradient boosting algorithm have appeeared, for example XGboost and LiteGBM, both of which impove the comptational efficiency, as well as the effectiveness of the base learners and the parameterisation of their combination. However, in all cases the theoretical underpinnings of the gradient boosting algorithm remain unchanged. The algorithm

consists of learning an ensemble of base learners as additive combination. In each iteration the new model is trained to predict the residuals of the preceding ensemble. Each weak learner is added to the ensemble with a small weighting, usually called a learning rate, which prevents over-fitting.

In this paper we develop a variation of the gradient boosting algorithm by making several non-standard assumptions about the noise present in the data. This involves breaking the assumption that the noise is identically distributed across the problem space. In particular we assume that the distribution of the noise varies in a way that can at least partially predicted by the known regressors for the machine learning problem.

We explore the consequences of this changed assumption and derive a variation of the gradient boosting algorithm which can potentially exploit this to build a model that fits the problem with varying levels of complexity. Noisier regions of the problem space are fit with simpler models that the low noise regions, and selectively applied by predicting whether a given sample belongs to a region that surpasses the noise threshold for each level of the boosted model.

2 Method

We start the derivation of the method by assuming that the noise is not randomly distributed with respect to the features or predictors. This assumption is contrary to the standard expression for the learning problem, an example of which is shown in Equation 1 taken from Friedman 1999 [2].

$$y_i = F^*(x_i) + \epsilon_i \tag{1}$$

This expression defines the target variable y_i of instance i as a function of the feature vector x plus some Gaussian noise ϵ_i. It is generally assumed that the noise term is independent of the feature vector X. It may correspond to inherent noise, or may be due to unobserved causal factores that are not present in the features, and hence limit our ability to perfectly represent the target using the given features.

We modifying this assumption in the following way: the noise term ϵ_i could be dependent upon the feature vector x_i. In real world processes there are multiple reasons why noise might be correlated with predictors. Noise can come from sampling problems, which are not uniformly distributed. Noise might be due to limitations of the devices or methodology used to collect data, which again tend not to be uniformly distributed. Most noise is due to some physical process, which itself is regulated by variables, if any of the variables that regulate the noise are present as features then we should expect the statistical structure of the noise to vary over the features space.

This means that we should expect the learning problem in many real world problems to be

$$y_i = F^*(x_i) + \epsilon(N(x_i)) \tag{2}$$

In which $\epsilon(N(x_i))$ refers to a sample drawn from a noise distribution N that is parameterised by the specific variables of the sample x_i.

When we are training a model to predict y_i, then we typically depict the fit of the model using a similar equation.

$$\hat{y}_i = \hat{F}(x_i) \tag{3}$$

Such that some function $Fhat$ has been fit such that given a feature vector x_i it provides an estimate of y. The error of such a model can be calculated for each record as the difference between the actual and predicted value:

$$err_i = y_i - \hat{y}_i \tag{4}$$

Note that error is typically calculated using some loss function $Ly, \hat{y}$, however we will ignore that complication for the moment. We can substitue Equations 2 and 3 into Equation 4 and with some rearrangement end up with the following expression for the error:

$$err_i = (F^*(x_i) - \hat{F}(x_i)) + \epsilon(N(x_i)) \tag{5}$$

In other words the expected error in our model is the difference between the true underlying function and the estimated one, plus the expected noise in that region of the problem space.

For many algorithms the change we have made to the distribution of noise is not something that can easily be exploited. However, the gradient boosting algorithm has the potential to exploit this new feature of the problem. Usually noise undermines a gradient boosting step because it is attempting to use the residual of the existing model to build a new iteration that mproves the previous model. One might imagine that at some point the residuals come to be dominated by the noise term, and hence a gradient boosted algorithm cannot see the signal for the noise.

If however, the noise is not identically distributed acorss the feature space then we could potentially direct each round of gradient boosting to ignore noisy examples, and focus on learning from a subset of the data.

In order to do so we need to provide some kind of selection criteria for deciding when each iteration of the gradient boosting should ignore an example. We introduce an additional threshold meta-parameter which determines when an example should be ignored. In addition we introduce a parameter that adjsts this downward with each iteration of learning. The idea is that we start the learning process identically to the gradient boosting algorithm. After each round of training we identify those sample whose error lies beyond the acceptable threhsold and exclude them. We expect

that as the algorithm proceeds the number of samples used shrinks, and we spend more time ignoring outliers.

Fianlly, in order to apply such a model we cannot simply make predictions using all stages of the gradient boosting because deeper level trees have been trained only on examples from a subset of feature space. In other words our gradient boosted model covers the problem space at different levels of granularity, depending on our expectations of the noise distribution.

What we need to do is only apply each subsequent gradient boosting in a way that it is attenuated according to our estimate of whether or not it is appropriate. We do this by fitting a classification model at each stage of gradient boosting which predicts whether or not a given example belongs to the signal set (of relatively low noise examples). We essentially build a model to estimate the probability that sample i is considered to be low noise at depth j in our gradient boosted model: $p_i^j(lownoise|x_i)$.

The final prediction can be calculated according to the following formula:

$$\hat{y}_i = base(x_i) + \sum_{j=1}^{M} p_i^j(lownoise|x_i) * r^j(x_i) \tag{6}$$

Our prediction yhat is a combination of the base model and the sum of all residual estimates over the M levels of boosting each of which attenuated by the estimate of the probability that this is a low noise example at that depth of the boosted model.

We can now ask how such an assumption changes our expectations about the errors made by a learning algorithm, and how we should adapt the algorithm to accomodate noise of this kind.

Learning algorithms generally involve a tension between fitting a function to the structure of the training data, and controlling its complexity so that it will generalise beyond the training data. The problem with fitting the training data perfectly is twofold. firstly most training sets will be an incomplete representation of the task to be learned. This is essense of the bias/variance dilemna, complex models with low bias can fit a give data set in a multitude of ways, resulting in high variance. The second problem with fitting the training data perfectly stems from the presence of noise.

In both cases the solution is to control the complexity of the model through some form of regularization.

If we presume that the learning algorithm is unbiased in its ability to learn the underlying (non-noise) part of the relationship between the features and response, then we would expect under this new defintion that the error made by the learning algorithm will not be evenly distributed across the feature space. Instead it will be weighted toward regions of the feature space that result in more noise due to the sensitivity of N to the values in x_i.

This leads us to assume that there will be some correlation between the magnitude of the error made by the model and the likelihood that this error is due to the presence of noise. This assumption requires only that our feature vector and model be rich enough that we expect the model to be able to learn the underlying relationship between dependent and independent variables well enough that noisier samples stand out. In the absence of noise we expect error to be low, when noise is present we expect it to be high. Because our change to the learning problem leads us to expect us non-uniform noise, we therefore expect the presence of a noise gradient that is correlated with the error.

Combining these two assumptions and the general idea of a gradient boosting leads us to an alternative learning algorithm we call Blinkered Gradient Boosting. We draw on the analogy of a racehorse wearing leather blinkers which prevent if from being distracted from activity on the edge of the racecourse. The metaphorical algorithmic blinkers prevent the model from being distracted by the large residual errors that are likely due to noise and hence unsolvable. It is worth noting the idea of minimising the influence of outliers in the error function is not new, it is the reason for the use of the Huber loss function []. However, gradient boosted methods generally assume that any error from previous iterations of the learning algorithm is due to a poor fit and need to be corrected. This is reason gradient boosting can be sensitive to noise.

3 Method

The core idea behind blinkered gradient boosting is that at each iteration of the training algorithm we exclude some portion of the samples due to the fact that their error is beyond the limits set by the blinkers. These blinkers are set at an outer bound with a new meta-paramter. They are gradually reduced in size by factor set by an additional meta-parametrs, or set to reduce by an amount set by the number of training epochs.

In principle we could use values for these blinkers that are assymetric around zero, however because all standard loss functions are symetric around zero we use a single blinker value that applies to positive and negative eror.

As well as excluding certain examples from the training data, we also train a binary classifier to predict whether a given example belongs to the set of data points whose error will be within those bounds. Ideally, as we apply the sequence of models to new data we do not want to attempt to correct predictions whose error is likley to be due to noise. We include these classifiers as an optional aspect of the algorithm. The user can choose whether to include them in the learning and scoring of data.

The most general version of the algorithm is shown below, as an adaptation of the algorithm described by Friedman 1999 [2].

Algorithm 1 Blinkered Gradient Boost

1: $F_0(x) = argmin_\rho \sum_{i=1}^{N} \mathcal{L}(y_i, \rho)$
2: **for** $m = 1$ to M **do**
3: $\quad B_m = \mathcal{B} - (m-1)\delta$
4: $\quad \Omega = \{i \in [1, N] \mid \mathcal{L}(y_i, F_{m-1}(x_i)) < B_m\}$
5: $\quad \tilde{y}_i = -\left[\frac{\partial \mathcal{L}(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}, i \in \Omega$
6: $\quad a_m = argmin_a \sum_{i \in \Omega} \mathcal{L}(\tilde{y}_i, h(x_i, a))$

Following from Friedman we define the algorithm in terms of an arbitrary loss function, illustrating that the technique is flexible and can be applied to any problem in which the notion of feature dependent noise is deemed appropriate. Our implementation is specific to our purposes, which is the forecasting of timeseries data with a large number of external regressors.

We implement our method as an S4 class in R, that wraps around a collection of H2O regression trees. H2O was chosen to provide the base learners because we have been dealing with large distributed data sets and required a method to implement and test algorithms on these data sets.

4 Data Sets

We extract 5 data sets from the UCI machine learning repository. We chose sets for which there were greater than ten thousand examples and more than 20 features, because we are looking for data sets that might fullfill the conditions for which the algorithm is designed.

Those data sets with their respective statistics are shown in Table XX.

5 Evaluation

We take three large regression data sets that are publically available and apply our technique alongside the gradient boosting implementation provided with H2O. We use default parameters and several common alternatives for the sake of a rounded comparison.

We evaluate the performance using Mean Absolute Error on a hold out data set. The models are trained using a early stopping on an validation data set. Both the validation and hold out test data were created as out-of-time data to reflect the kinds of real world processes we are interested in.

The code for the evaluation is provided in our GitHub repository.

Note, that due to liscensing restrictions we are unable to dsitribute the data sets themselves. In order to run it you need to download the public data sets from their sources (shown in Table **??**).

Lending CLub https://www.lendingclub.com/info/download-data.action

Walmart Sales Data https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/data

6 Results

7 Conclusion

By questioning a single standard assumption about the nature of the noise present in a data set we have been able to derive an variation of the gradient boosting algorithm. We have then showed using a selection of data sets that this model has utility for real world data.

References

[1] Friedman, J.H., Hastie, T., Tibshirani, R., (2000) *Additive logistic regression: a statistical view of boosting.*, Annals of Statistics, 28, 337407.

[2] Friedman, J.H., (2001) *Greedy Function Approximation: A Gradient Boosting Machine.*, Annals of Statistics, 29, 11891232.

[3] Friedman, J.H., (2002) *Stochastic gradient boosting.*, Computational Statistics and Data Analysis, 38, 367378.

[4] Andrew Fogg *Anthony Goldbloom gives you the secret to winning Kaggle competitions* https://www.import.io/post/how-to-win-a-kaggle-competition/ 2016.

[5] Robert Schapire *The boosting approach to machine learning an overview* MSRI Workshop on Nonlinear Estimation and Classification, 2002 (eds D.D. Denison, M. H. Hansen, C. Holmes, B. Mallick, B. Yu). Springer, New York.

[6] Yoav Freund *An adaptive version of the boost by majority algorithm.* Machine Learning, 43(3):293318, June 2001.