Gradient Boosted Regression Trees

A brief introduction

This document provides a brief mathematical introduction to the idea of gradient boosting, specifically in the context of gradient boosted regression trees (GBRT). For a more fleshed out and properly sourced version, see the supplemental methods of the GRL publication doi 10.1002/2017GL075661.

A.K. April 2017

Contents

1 Problem Setting

Consider a typical regression problem in which one seeks to predict a continuous variable from a vector of continuous or categorical *features*. The *training data* is a collection of observations of the form $(\boldsymbol{x}, y)$ where $\boldsymbol{x} = (x_1, \ldots, x_p)$ is a feature vector belonging to a p-dimesional feature space and y is the quantity of interest for prediction. Given n such observations:

$$\Big\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\Big\},$$

we wish to find a *predictor* $f : \mathbb{R}^p \to \mathbb{R}$ relating feature vectors to the quantity of interest y. This is achieved by formulating the problem of finding f as an optimization problem over a pre-specified family $\mathbb{F}$ of possible predictors. In other words, we seek an optimal (in a sense to be determined) function f in $\mathbb{F}$.

For a given predictor f we denote its predictions over the training data by:

$$\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_n) := \big(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)\big),$$

which we will refer to as the *prediction vector*. For convenience, we define the evaluation mapping $E : \mathbb{F} \to \mathbb{R}^n$ where n is the number of training samples, which maps each predictor f to its prediction vector:

$$f \mapsto \hat{\boldsymbol{y}} = \big(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)\big)$$

Goodness of fit is formulated in terms of a *loss function* $L : \mathbb{R}^n \to \mathbb{R}^+$ which assigns a measure of error $L(\hat{y})$ to the vector $\hat{\boldsymbol{y}}$ of n predictions with respect to observed values $\boldsymbol{y} = (y_1, \ldots, y_n)$. We will be using the common *sum of squares* loss function, also commonly used in linear regression, which is given by:

$$L(\hat{\boldsymbol{y}}) = \sum_{k=1}^{n} |\hat{y}_k - y_k|^2$$

Finally, our task for fitting a predictor f to the given samples is to solve the optimization problem:

$$\min_{f \in \mathbb{F}} L\left(E(f)\right))$$

The description of a statistical model involves the specification of the family $\mathbb{F}$ of predictors together with an algorithm that finds the optimal predictor f in $\mathbb{F}$ with respecto to L. In the remainder of this section we first describe ordinary regression trees as such a model. We then specify GBRT as an additive (i.e boosted) extension of regression trees or, in a more general sense, as a form of gradient descent over the predictor family $\mathbb{F}$. For this discussion, we will assume a fixed set of n training samples $\{(\boldsymbol{x}_k, y_k)\}$ is given.

2 Regression Trees

A regression tree partitions the feature space into a collection of box-like regions D_i and assigns a constant value y to each region. In other words, each such region is the subset of the feature space constrained by conditions of the form $x_k > T$ or $x_k < T$ where each x_k is an individual feature. Alternatively, such predictors can be viewed as a decision trees in which each node is a comparison of the form $x_k > T$ on some feature x_k. Given a feature vector $\boldsymbol{x}$, a predictor descends down the decision tree by comparing feature values with the conditions at each node. The bottom-most nodes in the tree, namely the *leaves*, are then assigned constant values which are returned by the predictor (1)

The predictor family $\mathbb{F}$ is the collection of all such piecewise constant functions. An individual predictor $f \in \mathbb{F}$ is identified by the specification of its box-like regions as well as the value it assigns to each region. In general, minimizing the loss function over the entire family $\mathbb{F}$ is intractable and instead, an approximating algorithm is typically employed which motivates its interpretation as a decision tree (Algorithm 1).

Since individual features are treated separately in Algorithm 1, it is clear that regression trees pose no difficulty when the feature set contains both continuous and categorical variables.

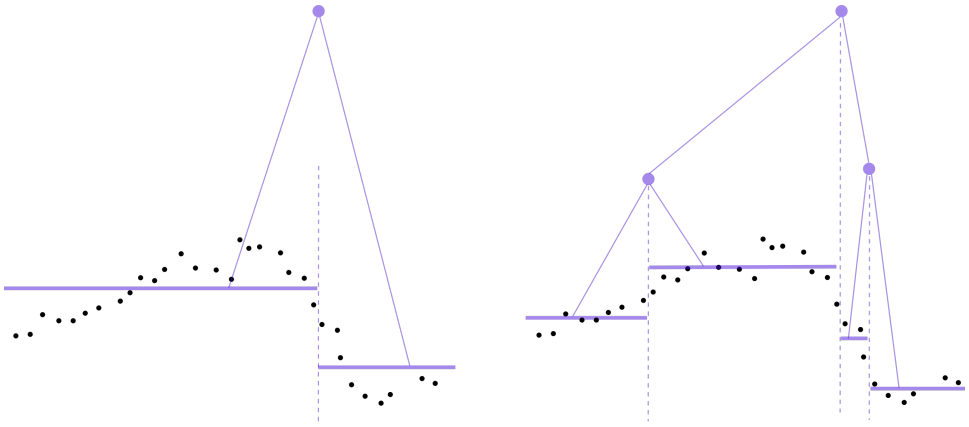

Figure 1: Example of a one dimensional regression tree after one (*left*) and two (*right*) iterations.

Algorithm 1 Ordinary Regression Tree

$D = \mathbb{R}^p$ ▷ The starting region is the entire feature space
$f(D) \leftarrow$ avg (y) ▷ Initialize with constant function
$j \leftarrow 0$ ▷ current tree depth
while $j < J$ **do** ▷ J is the maximum tree depth
  for region R **do** ▷ each region is a leaf in the tree
    $k \leftarrow$ "best" feature $k = 1, \ldots, p$ to split R on
    $T \leftarrow$ "best" threshold for feature x_k in R
    $D_1, D_2 \leftarrow$ partition D along the x_k axis with threshold T
    $f(D_i) \leftarrow$ avg (y) in D_i ▷ assign constants to each new region (leaf)
  end for
end while
return f

3 Boosted Regression

Boosting is a general heuristic in statistical learning which proposes combining multiple predictors, referred to as *weak learners*, into a stronger predictor. Given a family $\mathbb{F}$, a boosted predictor F based on $\mathbb{F}$ is a linear combination of predictors in $\mathbb{F}$, that is:

$$F = \alpha_1 f_1 + \ldots + \alpha_m f_m$$

for some $f_i \in \mathbb{F}$ and $\alpha_i \in \mathbb{R}$. When the family $\mathbb{F}$ is closed under scalar multiplication (as is the case with regression trees), we can simplify the general form of a boosted predictor to:

$$F = f_1 + \ldots + f_m$$

for some $f_i \in \mathbb{F}$. One can describe the problem of training a boosted model in similar terms as before: we seek the optimal predictor F in the boosted predictor family

$$\operatorname{span}(\mathbb{F}) = \Big\{ f_1 + \ldots + f_m \,;\, m \in \mathbb{N},\ f_i \in \mathbb{F} \Big\}$$

such that the training loss

$$\sum_{k=1}^{n} L\big(F(\boldsymbol{x}_k)\big)$$

is minimized. In a boosted regression tree model the underlying predictor family $\mathbb{F}$ consists of ordinary regression trees. However, there are many algorithms for approximating the optimal boosted predictor. Gradient boosting (Algorithm 2) is one such algorithm which is shown to contain many earlier boosting algorithms as a special case. In gradient boosting, at each stage a weak learner f_i is fit to the current *residual* vector which is the difference between known values $\boldsymbol{y}$ and the current prediction vector $\hat{\boldsymbol{y}}$ produced by $f_1 + \ldots + f_{i-1}$.

Our statistical model can thus be summarized as follows. Gradient boosting is used to approximate the optimal boosted regression tree $f_1 + \ldots + f_m$ that minimizes the sum of squares loss.

Algorithm 2 Gradient Boosted Regression Tree

$i \leftarrow 0$ ▷ number of weak learners trained so far
$\boldsymbol{r} \leftarrow \boldsymbol{y}$ ▷ n-dimensional vector of residuals
while $i < M$ **do** ▷ M is the number of weak learners to be trained
  $i \leftarrow i + 1$
  $f_i \leftarrow$ best weak learner fit to $\boldsymbol{r}$ ▷ i.e with training data $\Big\{(\boldsymbol{x}_1, r_1), \ldots, (\boldsymbol{x}_n, r_n)\Big\}$
  $f_i = \nu f$ ▷ regularization via shrinkage (see discussion below)
  $\hat{\boldsymbol{r}} \leftarrow \Big(f_i(\boldsymbol{x}_1), \ldots, f_i(\boldsymbol{x}_n)\Big)$
  $\boldsymbol{r} \leftarrow \boldsymbol{r} - \hat{\boldsymbol{r}}$ ▷ updated residuals
end while
return $f_1 + f_2 + \ldots + f_M$

Boosting has proved to be a fruitful strategy in various settings on multiple fronts. First, by extending the family of predictors from $\mathbb{F}$ to $\operatorname{span}(\mathbb{F})$, it allows for lower bias and variance. For instance, if $\mathbb{F}$ is the very limited family of step functions (i.e regression trees of depth 1), the extended class $\operatorname{span}(\mathbb{F})$ can approximate any function arbitrarily closely. This, however

comes with the risk of overfitting which is mitigated through regularization (discussed below). Furthermore, through the statistical power of averaging, boosting has been shown to increase robustness to noise (see discussion of stability below).

3.1 Boosted Regression as Gradient Descent

In this section we describe an alternative, and more general, description of boosted regression in terms of a form of gradient descent in a function space. This description is the motivation for the name "gradient boosting". However, we will see that under sum of squares loss the two are exactly identical.

First recall the familiar gradiant descent framework: given a scalar field $\mathscr{L}$ over a vector space V we wish to find the point $\boldsymbol{p} \in V$ that minimizes $\mathscr{L}$. We use the fact that at any point p the gradient of $\mathscr{L}$ at $\boldsymbol{p}$, here denoted by $\nabla_{\boldsymbol{p}} \mathscr{L}$, points in the direction of steepest ascent. Therefore, an iterative improvement approach along the direction of steepest descent, is guaranteed to find the global minimizer of $\mathscr{L}$ as long as $\mathscr{L}$ is convex (which is the case here and in most statistical models):

Algorithm 3 Standard Gradient Descent

$\boldsymbol{p} \leftarrow \boldsymbol{v}_1$ ▷ arbitrary starting point in V
$i \leftarrow 0$ ▷ the number of steps so far
while $|\nabla_{\boldsymbol{p}} \mathscr{L}| > \epsilon$ **do** ▷ alternatively, enforce an upper bound on i
  $i \leftarrow i + 1$
  $\boldsymbol{u} = -\nabla_{\boldsymbol{p}} \mathscr{L}$ ▷ direction of steepest descent
  $\boldsymbol{v}_i = \text{argmin}_{\boldsymbol{v}=\alpha \boldsymbol{u}} \mathscr{L}(\boldsymbol{v})$ ▷ line search in one variable α (along $\hat{u}$)
  $\boldsymbol{p} \leftarrow \boldsymbol{p} + \boldsymbol{v}_i$
end while
return $\boldsymbol{p} = \boldsymbol{v}_1 + \boldsymbol{v}_2 + \ldots + \boldsymbol{v}_i$

Now instead of the usual finite dimensional vector space of calculus, let V be the vector space of functions $\text{span}(\mathbb{F})$ which is the domain of optimization in boosted regression. In boosted regression trees $\mathbb{F}$ is the family of piecewise constant functions of a specified maximum tree depth J [1]. The function to be minimized over V is $\mathscr{L} = L \circ E$ where E evaluates a predictor f at the training samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and the loss function L assigns an error to the prediction vector $\hat{\boldsymbol{y}} = E(f)$.

Recall that the gradient of a scalar field $\mathscr{L} : V \to \mathbb{R}$ at a point $\boldsymbol{p}$ is the linear map

$$\nabla_{\boldsymbol{p}} \mathscr{L} : V \to \mathbb{R}$$

that assigns to any vector $\boldsymbol{v}$ the rate of change of $\mathscr{L}$ at $\boldsymbol{p}$ along $\boldsymbol{v}$:

$$\boldsymbol{v} \mapsto \nabla_{\boldsymbol{p}} \mathscr{L}(\boldsymbol{v}) := \lim_{\epsilon \to 0} \frac{1}{\epsilon} [\mathscr{L}(\boldsymbol{p} + \epsilon \boldsymbol{v}) - \mathscr{L}(\boldsymbol{p})]$$

In boosting, we have $V = \text{span}(\mathbb{F})$ and the "gradient" [2] at any "point" $f \in V$ is a linear map $\nabla_f \mathscr{L} : V \to \mathbb{R}$ assigning to each "direction" ϕ (which is now a function in V) the

[1] However, the discussion in this section is independent of the choice of $\mathbb{F}$ as long as it is closed under scalar multiplication. In fact, *any* boosting additive model can be viewed as gradient descent in a function space as described in this section.

[2] technically known as the *Fréchet derivative* over a function space.

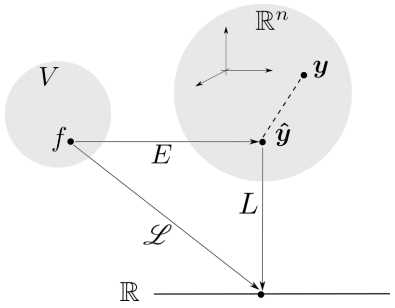

Figure 2: Schematic representation of the functional optimization problem to minimize $\mathscr{L} = L \circ E$ over the vector space $V = \text{span}(\mathbb{F})$.

directional rate of change:

$$\phi \mapsto \nabla_f \mathscr{L}(\phi) := \lim_{\epsilon \to 0} \frac{1}{\epsilon} [\mathscr{L}(f + \epsilon\phi) - \mathscr{L}(f)]$$

In vector calculus, V is a finite dimensional space, say $\mathbb{R}^m$ and $\nabla_{\boldsymbol{p}} \mathscr{L}$, being a linear map, can be represented as multiplication by a $m \times 1$ matrix. This matrix multiplication can be viewed as a dot product such that for any $\boldsymbol{v} \in V$ the rate of change along $\boldsymbol{v}$ is $\nabla_{\boldsymbol{p}} \mathscr{L} \cdot \boldsymbol{v}$. Unfortunately, the dot product representation of the gradient does not generalize in a computationally tractable way to function spaces. However, in our case the structure of our scalar field $\mathscr{L} = L \circ E$ allows a similar formulation due to the simplicity of the evaluation function $E : V \to \mathbb{R}^n$. An application of the familiar chain rule for $\mathscr{L} = L \circ E$ (which is valid in infinite dimensional vector spaces) gives:

$$\nabla_f \mathscr{L} = \nabla_{E(f)} L \circ \nabla_f E$$

But it can easily be verified that E is linear and thus $\nabla_f E \equiv E$ for any f. Therefore, for any predictor $f \in V$ and any direction of change $\phi \in V$ we have:

$$\nabla_f \mathscr{L}(\phi) = \nabla_{E(f)} L \cdot E(\phi)$$

where $\nabla_{E(f)} L$ is the familiar n-dimensional gradient of $L : \mathbb{R}^n \to \mathbb{R}$ and the dot product is the familiar dot product in $\mathbb{R}^n$. Thus, at each step of gradient descent on $V = \text{span}(F)$ to minimize $\mathscr{L}$, the problem of finding the direction of steepest descent at is reduced to:

$$\max_{\phi \in \text{span}(F)} -\nabla_{E(f)} L \cdot E(\phi)$$

where $f = f_1 + f_2 + \ldots + f_i$ is the current position (predictor) in V. However, since this is not in general tractable, the domain of search is typically restricted from $\text{span}(\mathbb{F})$ to $\mathbb{F}$. Furthermore, under regularity conditions, the problem can be equivalently formulated in terms of distance minimization:

$$\min_{\phi \in F} \left\| -\nabla_{E(f)} L - E(\phi) \right\|$$

where $\|\boldsymbol{a} - \boldsymbol{b}\|$ is the usual Euclidean distance in $\mathbb{R}^n$. However, under sum of squares loss, the latter is precisely what weak learners optimize for. More specifically, a weak learner finds $\phi \in \mathbb{F}$ that minimizes the distance between $\hat{y} = E(\phi)$ and some known vector, which here is

$-\nabla_{E(f)} L$ (called *pseudo-residuals*). The "direction" of steepest descent within $\mathbb{F}$, therefore, can be obtained by fitting a weak learner to pseudo-residuals. Note that under the sum of squares L the pseudo-residuals are, up to a factor, identical to ordinary residuals of the vanilla boosting algorithm:

$$-\nabla_{E(f)} L = 2(\boldsymbol{y} - \hat{\boldsymbol{y}})$$

In other words, under sum of squares loss, the gradient descent generalization described here is exactly identical to the simple additive model described in the prevision section.

Algorithm 4 Gradient Boosted Regression

$f \leftarrow \text{avg}\ (y)$ ▷ arbitrary starting point in $\mathbb{F}$
$i \leftarrow 0$ ▷ the number of steps so far
while $i < M$ **do** ▷ M is the max number of steps
  $i \leftarrow i + 1$
  $\phi = \text{argmax}_{\phi \in \mathbb{F}} - \nabla_{E(f)} L \cdot E(\phi)$ ▷ direction of steepest descent
  $f_i = \text{argmin}_{f=\alpha\phi} \mathscr{L}(f)$ ▷ line search in one variable α (along ϕ)
  $f_i = \nu f$ ▷ shrinkage regularization, see discussion below
  $f \leftarrow f + f_i$
end while
return $f = f_1 + \ldots + f_M$

4 Gradient Boosted Regression Tree

The GBRT model is exactly the model described in the previous section with $\mathbb{F}$ being the family of regression trees of maximum depth J. Note that since we are allowing boosted (additive) regression trees we can use small values for J. Furthermore, note that in functional gradient descent, unlike the standard gradient descent, the stopping criteria cannot be, in general, formulated in terms of $|\nabla_f \mathscr{L}| < \epsilon$ since that requires imposing additional mathematical structure on the function space. The typical stoppage scheme is to terminate the search after M steps. This means the output of a GBRT is a sum of M regression trees each of which has depth at most J.

4.1 Regularization

A crucial difference between optimization problems and statistical modelling is the risk of overfitting. In fact, typically one can find predictors that drive the total training loss to zero (Fig. 3).

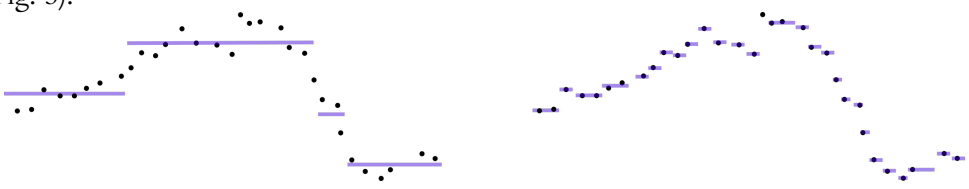

Figure 3: Arbitrarily small training loss with a regression tree (*right*) which is overfit to training data.

The standard scheme for containing the effects of overfitting (also known as *regularization*) for regression trees is to place an upper bound J on the decision tree depth. In gradient boosted trees, two additional regularization strategies are typically simultaneously used: **1)** an upper bound M on the number of descent steps (discussed above) and **2)** *shrinkage* which is used to scale down the contributions calculated in each gradient descent step by a parameter $0 < \nu \leq 1$. A choice of $\nu = 1$ implies no shrinkage, i.e non-regularized gradient descent, and smaller values of ν slow down the march towards arbitrarily small training losses, hence limiting the risk of overfitting.

4.2 Stability

An important desired characteristic of a statistical model is its *stability*, namely its robustness to noise. This is crucial in cases where one either has limited amounts of training data or data is known to be highly noisy. We wish our statistical model to be such that small perturbations in known labels $y_1, \ldots, y_n$ does not lead to large perturbations in the trained predictor. This is another domain where boosted regression trees are superior to ordinary regression trees: single trees are known to be unstable while boosted trees tend to be fairly stable.

In order to evaluate the stability of our model, we can add random noise to known y values of the training set and report the average difference between predictions under noise-free and noisy training data. This procedure is then repeated multiple times over random training and validation sets and with various noise amplitudes.

At various levels of noise we can repeat the standard cross-validation procedure with the addition of noise to training values.

The relative noise *amplitude* $0 \leq A \leq 1$ which is the ratio between the expected value of the added Gaussian noise η with respect to the average y value, namely $A = \mathbb{E}[\eta]/\bar{y}$ where $\bar{y}$ is the average y over the training set. The extreme case of $A = 0$ corresponds to unperturbed training data.

For each partition of data to training and validation sets, multiple predictors should be considered. First, a baseline predictor f_0 is obtained from noise-free training data (i.e $A = 0$). Then, for each relative noise amplitude value A a predictor f_A is obtained from proportionally noisy data:

$$\left\{ \left(\boldsymbol{x}_k^{(t)}, y_k^{(t)} + \eta_k \right) \right\}_{k=1}^{m}$$

where

$$\frac{\eta_k}{\bar{y}} \sim \mathscr{N}\left(0, \frac{\pi A^2}{2}\right)$$

The perturbation in f_A is obtained by comparing its predictions over the validation set with those of the baseline predictor f_0:

$$\left\| f_A - f_0 \right\| = \left[\sum_{k=1}^{n-m} \left| f_A\left(\boldsymbol{x}_k^{(v)}\right) - f_0\left(\boldsymbol{x}_k^{(v)}\right) \right|^2 \right]^{\frac{1}{2}}$$

This perturbation is then averaged over multiple rounds of partitioning the samples into training and validation sets.