A quick look at gradient boosting

Derek Huang

March 28, 2020

1 Introduction

We briefly describe gradient boosting from a variational calculus perspective by considering steepest descent in function space, giving an explicit formula for the optimal function step ζ_m. Then, we discuss approximating ζ_m in practice, write a generic algorithm for gradient boosting, and discuss two special cases of gradient boosting, namely two-class AdaBoost and gradient tree boosting. We show how two-class AdaBoost falls under the gradient boosting framework and write an algorithm for gradient tree boosting.

2 General gradient boosting

2.1 Fitting additive models

A general additive model $F : X \to \mathbb{R}$ can be written concisely as

$$F(\mathbf{x}) = \sum_{m=1}^{M} \beta_m \phi_m(\mathbf{x}) \tag{2.1}$$

Here each $\phi_m : X \to \mathbb{R}$, usually referred to as a *base learner* or *basis function* is a simpler function, where in practice one sees that $\forall m \in \{1, \dots M\}, \phi_m \in \mathcal{H}_\theta$, where $\mathcal{H}_\theta$ is a hypothesis set of function indexed with hyperparameter vector θ. Each $\beta_m \in \mathbb{R}$ is an associated weight, or *expansion coefficient*, for ϕ_m. In general, additive models are built in a greedy forward stagewise fashion, i.e, with each $\beta_m \phi_m$ added sequentially. Defining a per-example loss function $L : Y \times \mathbb{R} \to \mathbb{R}_+$, given an incomplete model with $m-1$ basis functions F_{m-1}, the optimal β_m and ϕ_m added at the mth iteration are given by

$$\beta_m, \phi_m = \arg\min_{\beta, \phi} \mathbb{E}_{\mathbf{x}, y}[L(y, F_{m-1}(\mathbf{x}) + \beta\phi(\mathbf{x}))] \tag{2.2}$$

We can view the minimization problem from a variational calculus perspective, treating the expected loss function as a *functional*, where we choose F_m by taking a step in the direction of steepest descent in function space. This approach is termed *gradient boosting*, and is one particular way to greedily choose β_m and ϕ_m.

2.2 Functional gradient descent

We can view the expression for expected loss, given a convex, differentiable loss function $L : Y \times \mathbb{R} \to \mathbb{R}_+$ and a function $f : X \to \mathbb{R}$, where we assume $f \in \mathcal{H}$, where $\mathcal{H}$ is a reproducing kernel Hilbert space, as a linear functional $J : \mathcal{H} \to \mathbb{R}_+$. We may then define the linear functional J, for a function $f \in \mathcal{H}$, as[1]

$$J[f] \triangleq \mathbb{E}_{\mathbf{x}, y}[L(y, f(\mathbf{x}))] = \int_{X \times Y} L(y, f(\mathbf{x})) \psi_{\widetilde{X}, \widetilde{Y}}(\mathbf{x}, y)\, dy\, d\mathbf{x} \tag{2.3}$$

Here we assumed the existence of a properly defined joint probability density function $\psi_{\widetilde{X}, \widetilde{Y}} : X \times Y \to \mathbb{R}_+$ and also make the implicit assumption that $J[f] < \infty$. The optimization problem we wish to solve can thus be concisely written as $\min_f J[f]$, but it is difficult to say a priori how one should choose f. Iterative steepest

[1] This section is loosely based on material from [1] and [4].

descent would prescribe that we start from an initial guess f_0, and then for each step m, given a previous guess f_{m-1}, define f_m by taking a step in the direction opposite to that of the functional gradient, which we denote as $\delta J/\delta f$, evaluated at f_{m-1}. It is clear that we can use iterative steepest descent in function space as a very greedy way to perform forward stagewise additive modeling.

Noting that we can rewrite the minimization problem in (2.2) in functional form as $\min_\zeta J[F_{m-1}+\zeta]$, where our current guess is F_{m-1}, to find $\zeta \in \mathcal{H}$ using steepest descent, we first consider the first variation of our functional, which we denote as $\delta J[F_{m-1}]$. For some arbitrarily small $\varepsilon > 0$, $\forall \lambda \in \mathcal{H}$, we define a perturbation in F_{m-1} as $\delta F_{m-1} \triangleq \varepsilon\lambda$. Since J is linear, we can write its first variation, or differential, $\delta J[F_{m-1}]$ as

$$\begin{aligned}\delta J[F_{m-1}] &= J[F_{m-1}+\delta F_{m-1}] - J[F_{m-1}] = \int_{X\times Y} L(y, F_{m-1}(\mathbf{x}) + \delta F_{m-1}(\mathbf{x}))\psi_{\widetilde{X},\widetilde{Y}}(\mathbf{x}, y)\, dy\, d\mathbf{x} - J[F_{m-1}] \\ &\approx \int_{X\times Y}\left[L(y, F_{m-1}(\mathbf{x})) + \frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\delta F_{m-1}(\mathbf{x})\right]\psi_{\widetilde{X},\widetilde{Y}}(\mathbf{x}, y)\, dy\, d\mathbf{x} - J[F_{m-1}] \\ &= \int_{X\times Y}\frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\delta F_{m-1}(\mathbf{x})\psi_{\widetilde{X},\widetilde{Y}}(\mathbf{x}, y)\, dy\, d\mathbf{x}\end{aligned}$$

Since we assumed that $J[f] < \infty$[2], we may apply Fubini's theorem. Then, in terms of iterated integrals, and also rewriting $\psi_{\widetilde{X},\widetilde{Y}}$ as the product of the conditional density $\psi_{\widetilde{Y}|\mathbf{x}}$ and the marginal density $\psi_{\widetilde{X}}$, we have

$$\begin{aligned}\delta J[F_{m-1}] &= \int_X\left[\int_Y \frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\psi_{\widetilde{Y}|\mathbf{x}}(y)\, dy\right]\psi_{\widetilde{X}}(\mathbf{x})\delta F_{m-1}(\mathbf{x})\, d\mathbf{x} \\ &= \int_X \psi_{\widetilde{X}}(\mathbf{x})\mathbb{E}_{y|\mathbf{x}}\left[\frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\right]\delta F_{m-1}(\mathbf{x})\, d\mathbf{x} \\ &= \varepsilon\int_X \psi_{\widetilde{X}}(\mathbf{x})\mathbb{E}_{y|\mathbf{x}}\left[\frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\right]\lambda(\mathbf{x})\, d\mathbf{x}\end{aligned}$$

If the conditional expectation and λ are L^2 integrable[3], then we may define the functional gradient as in [1], as the functional variation $\delta J[F_{m-1}]$ is the L^2 inner product $\varepsilon\langle\psi_{\widetilde{X}}\mathcal{E}_y, \lambda\rangle$, where $\mathcal{E}_y$ is defined such that

$$\mathcal{E}_y(\mathbf{x}) = \mathbb{E}_{y|\mathbf{x}}\left[\frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\right] = \int_Y \frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\psi_{\widetilde{Y}|\mathbf{x}}(y)\, dy$$

Therefore, the functional gradient of J evaluated at F_{m-1} is given by

$$\frac{\delta J}{\delta f}[F_{m-1}] = \psi_{\widetilde{X}}\mathcal{E}_y \tag{2.4}$$

We can also express the functional derivative of J evaluated at $F_{m-1}(\mathbf{x})$ as

$$\frac{\delta J}{\delta f(\mathbf{x})}[F_{m-1}] = \psi_{\widetilde{X}}(\mathbf{x})\mathbb{E}_{y|\mathbf{x}}\left[\frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\right] \tag{2.5}$$

Then, again considering the functional form of the minimization problem in (2.2), $\min_\zeta J[F_{m-1}+\zeta]$, it is thus clear that the optimal $\zeta_m \in \mathcal{H}$ prescribed by steepest descent is defined such that

$$\zeta_m(\mathbf{x}) = -\gamma_m\frac{\delta J}{\delta f(\mathbf{x})}[F_{m-1}] = \gamma_m\psi_{\widetilde{X}}(\mathbf{x})\mathbb{E}_{y|\mathbf{x}}\left[-\frac{\partial L}{\partial F_{m-1}(\mathbf{x})}(y, F_{m-1}(\mathbf{x}))\right] \tag{2.6}$$

Here the constant $\gamma_m > 0$ is chosen to solve $\min_\zeta J[F_{m-1}+\zeta]$ and controls the step size. We can see that for some $\mathbf{x} \in X$, $\zeta_m(\mathbf{x})$ is the conditional expectation of the negative partial derivative of the loss function L with respect to $F_{m-1}(\mathbf{x})$ given $\mathbf{x}$, scaled by a constant $\gamma_m > 0$ and the density $\psi_{\widetilde{X}}(\mathbf{x})$.

[2]We need not take an absolute value since both the loss function L and the joint density function $\psi_{\widetilde{X},\widetilde{Y}}$ are nonnegative.

[3]This is not hard if we take $\mathcal{H}$ to be a suitable subset of L^2. Also, $\psi_{\widetilde{X}} \in L^2$ as it is a density function.

2.3 Approximating ζ_m

In practice, one is only given a finite set $\mathcal{D}$ of training examples $(\mathbf{x}_k, y_k)$ to use for estimating the negative conditional expectation function $-\mathcal{E}_y$ and the marginal density function $\psi_{\widetilde{X}}$ before choosing the step size γ_m. At any iteration m, to estimate $-\mathcal{E}_y$, we first need to compute $\forall(\mathbf{x}_k, y_k) \in \mathcal{D}$ the partial derivatives[4]

$$\partial_{F_{m-1}(\mathbf{x}_k)} L_k \triangleq \frac{\partial L}{\partial F_{m-1}(\mathbf{x}_k)}(y_k, F_{m-1}(\mathbf{x}_k))$$

We can then directly estimate $-\mathcal{E}_y$ by fitting a base learner $\phi_m \in \mathcal{H}_\theta$ that solves

$$\phi_m = \arg\min_{\phi} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \widetilde{L}\left(-\partial_{F_{m-1}(\mathbf{x}_k)} L_k, \phi(\mathbf{x}_k)\right) \tag{2.7}$$

Here $\widetilde{L} : \mathbb{R}^2 \to \mathbb{R}_+$ is the loss function used for fitting ϕ_m to the partial derivatives $\partial_{F_{m-1}(\mathbf{x}_k)} L_k$, typically squared error loss. However, the marginal density function $\psi_{\widetilde{X}}$ and the step length γ_m still need to be determined. We therefore need to find a function $\hat{\psi}_{\widetilde{X}}$ that approximates $\psi_{\widetilde{X}}$, but since in practice it is assumed each input point is equally likely to be selected, then $\forall(\mathbf{x}_k, y_k) \in \mathcal{D}$, we can choose $\hat{\psi}_{\widetilde{X}}(\mathbf{x}_k) = 1/|\mathcal{D}|$. Then, the quantity $\gamma_m \hat{\psi}_{\widetilde{X}}(\mathbf{x}_k)$ approximating $\gamma_m \psi_{\widetilde{X}}(\mathbf{x})$ is constant $\forall(\mathbf{x}_k, y_k) \in \mathcal{D}$ for a fixed choice of γ_m and equals $\gamma_m/|\mathcal{D}|$. Denoting $\beta_m \triangleq \gamma_m/|\mathcal{D}|$, $\beta_m > 0$ can be found directly as the quantity that solves

$$\beta_m = \arg\min_{\beta} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} L(y_k, F_{m-1}(\mathbf{x}_k) + \beta \phi_m(\mathbf{x}_k)) \tag{2.8}$$

Note that we may use a first order Taylor expansion to rewrite the minimization problem in (2.8) as

$$\min_{\beta} \left\{ \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} L(y_k, F_{m-1}(\mathbf{x}_k)) + \frac{\beta}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \partial_{F_{m-1}(\mathbf{x}_k)} L_k \phi_m(\mathbf{x}_k) \right\}, \beta > 0$$

Since for any iteration m, the total loss must be nonnegative and our chosen step size $\beta_m > 0$, we note that if $\sum_{k=1}^{|\mathcal{D}|} \partial_{F_{m-1}(\mathbf{x}_k)} L_k \phi_m(\mathbf{x}_k) < 0$, then, if we are being very greedy[5], we can find β_m explicitly through

$$\beta_m = -\frac{\sum_{k=1}^{|\mathcal{D}|} L(y_k, F_{m-1}(\mathbf{x}_k))}{\sum_{k=1}^{|\mathcal{D}|} \partial_{F_{m-1}(\mathbf{x}_k)} L_k \phi_m(\mathbf{x}_k)} \tag{2.9}$$

However, if $\sum_{k=1}^{|\mathcal{D}|} \partial_{F_{m-1}(\mathbf{x}_k)} L_k \phi_m(\mathbf{x}_k) \geq 0$, then the optimal greedy choice for β_m must be zero, as any $\beta_m > 0$ increases the total loss. Then, our best approximation for ζ_m is thus given by $\beta_m \phi_m$.

In practice, it is often common to choose, for each iteration m, a small constant $\eta_m > 0$ called the *learning rate* to scale down the magnitude of β_m for the purpose of reducing overfitting. Typically, for any iteration m, the learning rate is chosen to be constant, i.e. $\forall m \in \{1, \dots M\}$, $\eta_m \triangleq \eta > 0$, but this is not required.

2.4 A gradient boosting algorithm

Now that we know how to choose our function ζ_m by selecting the appropriate constant $\beta_m > 0$ and base learner ϕ_m, we may write down a generic algorithm to perform gradient boosting given a training data set $\mathcal{D}$ of examples $(\mathbf{x}_k, y_k)$, where the goal is to learn an additive $F : X \to \mathbb{R}$ of the form defined in (2.1).

[4] We have consistently abused this notation. If L has arguments y^* and $\hat{y}$, which represent the target and predicted values respectively, then the partial derivative with respect to the second argument, evaluated at $(y, F_{m-1}(\mathbf{x}))$, should be written as

$$\frac{\partial L}{\partial \hat{y}}(y, F_{m-1}(\mathbf{x}))$$

However, it does make sense to emphasize the partial derivative being taken with respect to a change $F_{m-1}(\mathbf{x})$.

[5] If at any iteration m we try to set the total loss to zero, our choice of β_m is given by (2.9).

Algorithm 1: Gradient boosting

1. Define an initial constant model $F_0 \triangleq \kappa \in \mathbb{R}$ and the loss functions $L : Y \times \mathbb{R} \to \mathbb{R}_+$, $\widetilde{L} : \mathbb{R}^2 \to \mathbb{R}_+$.
2. For each boosting stage $m = 1, \dots M$,
   (a) For $k = 1, \dots |\mathcal{D}|$, compute the loss function partial derivatives

$$\partial_{F_{m-1}(\mathbf{x}_k)} L_k \triangleq \frac{\partial L}{\partial F_{m-1}(\mathbf{x}_k)}(y_k, F_{m-1}(\mathbf{x}_k))$$

   Then, fit a function $\phi_m : X \to \mathbb{R}$ to the negative loss function partials, where $\phi_m \in \mathcal{H}_\theta$ and

$$\phi_m = \arg\min_{\phi} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \widetilde{L}\left(-\partial_{F_{m-1}(\mathbf{x}_k)} L_k, \phi(\mathbf{x}_k)\right)$$

   Typically, the loss function $\widetilde{L}$ is chosen to be squared error loss.
   (b) Choose the step size $\widetilde{\beta}_m > 0$ that is given by

$$\widetilde{\beta}_m = \arg\min_{\beta} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} L(y_k, F_{m-1}(\mathbf{x}_k) + \beta\phi_m(\mathbf{x}_k))$$

   The final step size β_m is given by $\beta_m = \eta_m \widetilde{\beta}_m$, where $\eta_m > 0$ is the learning rate.
   (c) Define the updated model F_m, where $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m \phi_m(\mathbf{x})$.
3. Output the final model[6] $F : X \to \mathbb{R}$, where $F \triangleq F_M$ and

$$F(\mathbf{x}) = \kappa + \sum_{m=1}^{M} \beta_m \phi_m(\mathbf{x})$$

The initial constant model $F_0 \triangleq \kappa$ is typically determined in an problem-specific manner. For regression problems, the typical choice is to set F_0 to be the sample mean of the y-values in the training set, that is $\kappa = \frac{1}{|\mathcal{D}|}\sum_{k=1}^{|\mathcal{D}|} y_k$. For a binary classification problem, where the response labels are $\{-1, 1\}$ and F is a discriminant function we are interested in learning, then $\kappa = 0$ is the natural choice.

3 Two special cases

It is worthwhile to discuss two well-known, special cases of gradient boosting, namely two-class AdaBoost and gradient boosting where each ϕ_m is a fixed-size tree, termed *gradient tree boosting*.

3.1 Two-class AdaBoost

Recalling that AdaBoost uses the exponential loss function $L : \{-1, 1\} \times \mathbb{R} \to \mathbb{R}_+$, where $L(y, \hat{y}) = e^{-y\hat{y}}$, we see that we may rewrite (2.6) to show that under AdaBoost, the optimal ζ_m is given by

$$\zeta_m(\mathbf{x}) = \gamma_m \psi_{\widetilde{X}}(\mathbf{x}) \mathbb{E}_{y|\mathbf{x}}\left[y e^{-yF_{m-1}(\mathbf{x})}\right] = \gamma_m \psi_{\widetilde{X}}(\mathbf{x}) \mathbb{E}_{y|\mathbf{x}}\left[w_m(\mathbf{x}, y) y\right] \tag{3.1}$$

Here we write $w_m(\mathbf{x}, y) \triangleq e^{-yF_{m-1}(\mathbf{x})}$ as a weight for the pair $(\mathbf{x}, y)$, where $y \in \{-1, 1\}$. For any example $(\mathbf{x}_k, y_k) \in \mathcal{D}$, as in AdaBoost, we denote its weight as $w_m^{(k)} = w_m(\mathbf{x}_k, y_k)$. Using more familiar probabilistic notation for the conditional expectation, denoting the random variables for the input and response with $\widetilde{X}$

[6] We could redefine ϕ_1 to include the constant κ, or explicitly index from 0 and define $\beta_0\phi_0 \triangleq \kappa$.

and $\widetilde{Y}$ respectively, we can write $\mathbb{E}_{y|\mathbf{x}}[w_m(\mathbf{x}, y)y]$ as $\mathbb{E}\big[w_m(\widetilde{X}, \widetilde{Y})\widetilde{Y} \mid \widetilde{X} = \mathbf{x}\big]$, which represents the conditional expectation of the *weighted* response. Then, under the framework of gradient boosting, ϕ_m is given by

$$\phi_m = \arg\min_{\phi} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \widetilde{L}\left(-\partial_{F_{m-1}(\mathbf{x}_k)} L_k, \phi(\mathbf{x}_k)\right) = \arg\min_{\phi} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \widetilde{L}\big(w_m^{(k)} y_k, \phi(\mathbf{x}_k)\big) \tag{3.2}$$

Here $\widetilde{L} : \mathbb{R}^2 \to \mathbb{R}_+$ is the loss function used to fit $\phi_m : X \to \{-1, 1\}$. Note that since ϕ_m only outputs values in $\{-1, 1\}$, AdaBoost is in fact approximating the negative loss function partials very roughly, constraining the magnitude at any point to 1. Total loss is thus minimized pointwise at any training point $(\mathbf{x}_k, y_k) \in \mathcal{D}$ if and only if $w_m^{(k)} y_k f(\mathbf{x}_k) > 0$, where the weight $w_m^{(k)}$ increases the impact of the point on the total loss.

We note that in two-class AdaBoost, it is typically written that ϕ_m is chosen such that

$$\phi_m = \arg\min_{\phi} \sum_{k \in I^{\mathsf{c}}(\phi)} w_m^{(k)} \tag{3.3}$$

Here $I^{\mathsf{c}}(\phi) \triangleq \{k \in \mathbb{N} : y_k \neq \phi(\mathbf{x}_k), (\mathbf{x}_k, y_k) \in \mathcal{D}\}$. Since in two-class AdaBoost the loss function L is the exponential loss function, we were able to reinterpret the negative loss function partials in (3.2) as weighted response values. Then, it is clear that (3.2) expresses (3.3) under the gradient boosting framework.

Now the remaining task is to approximate $\gamma_m \psi_{\widetilde{X}}$, where in practice $\psi_{\widetilde{X}}$ is approximated with $\hat{\psi}_{\widetilde{X}} \triangleq 1/|\mathcal{D}|$, so $\forall (\mathbf{x}_k, y_k) \in \mathcal{D}$, $\gamma_m \hat{\psi}_{\widetilde{X}}(\mathbf{x}_k) = \gamma_m / |\mathcal{D}|$ is constant for any choice of γ_m. We then define $\beta_m > 0$ in the usual way, where $\beta_m \triangleq \gamma_m / |\mathcal{D}|$, which under the framework of gradient boosting satisfies

$$\beta_m = \arg\min_{\beta} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} L(y_k, F_{m-1}(\mathbf{x}_k) + \beta \phi_m(\mathbf{x}_k)) \tag{3.4}$$

However, since the L in AdaBoost is the exponential loss function, the minimization in (3.4) becomes

$$\min_{\beta} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} e^{-y_k (F_{m-1}(\mathbf{x}_k) + \beta \phi_m(\mathbf{x}_k))} = \min_{\beta} \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} w_m^{(k)} e^{-y_k \beta \phi_m(\mathbf{x}_k)}$$

We recall our definition of $I^{\mathsf{c}}(\phi)$, and define $I(\phi) \triangleq \{k \in \mathbb{N} : y_k = \phi(\mathbf{x}_k), (\mathbf{x}_k, y_k) \in \mathcal{D}\}$, where necessarily we have $|I(\phi)| + |I^{\mathsf{c}}(\phi)| = |\mathcal{D}|$. Since $y_k, \phi_m(\mathbf{x}_k) \in \{-1, 1\}$, $\forall (\mathbf{x}_k, y_k) \in \mathcal{D}$, we see that

$$\min_{\beta} \sum_{k=1}^{|\mathcal{D}|} w_m^{(k)} e^{-y_k \beta \phi_m(\mathbf{x}_k)} = \min_{\beta} \left\{ \sum_{k \in I(\phi_m)} w_m^{(k)} e^{-\beta} + \sum_{k \in I^{\mathsf{c}}(\phi_m)} w_m^{(k)} e^{\beta} \right\} = \min_{\beta} J_m(\phi_m, \beta, \mathcal{D})$$

Since J_m is convex in β, then $\partial J_m / \partial \beta$ evaluated at β_m is zero and the global minimum of J_m. We compute

$$\frac{\partial J_m}{\partial \beta}(\phi_m, \beta_m, \mathcal{D}) = -\sum_{k \in I(\phi_m)} w_m^{(k)} e^{-\beta_m} + \sum_{k \in I^{\mathsf{c}}(\phi_m)} w_m^{(k)} e^{\beta_m} = 0$$

After factoring out the $e^{-\beta_m}$ and e^{β_m} terms, rearranging, and taking the natural logarithm, we have

$$\beta_m + \log\left(\sum_{k \in I^{\mathsf{c}}(\phi_m)} w_m^{(k)}\right) = -\beta_m + \log\left(\sum_{k \in I(\phi_m)} w_m^{(k)}\right)$$

Then, rearranging and using properties of logarithms, we arrive at the expression

$$\beta_m = \frac{1}{2} \log\left(\frac{\sum_{k \in I(\phi_m)} w_m^{(k)}}{\sum_{k \in I^{\mathsf{c}}(\phi_m)} w_m^{(k)}}\right) \tag{3.5}$$

We see that we have an analytical expression for each basis expansion coefficient β_m in terms of the training example weights $w_m^{(k)}$, which is exactly how β_m is defined in two-class AdaBoost. It is now clear that two-class AdaBoost is a special case of gradient boosting with an exponential loss function where $\forall (\mathbf{x}_k, y_k) \in \mathcal{D}$, the negative loss function partials $-\partial_{F_{m-1}(\mathbf{x}_k)} L_k$ can be interpreted as weighted response values $w_m^{(k)} y_k$, and where each β_m can be determined analytically using the weights $w_m^{(k)}$.

3.2 Gradient tree boosting

The case where $\forall m \in \{1, \dots M\}$, ϕ_m is a fixed-size decision tree is an interesting special case of gradient boosting. In general, we will abstractly define a regression tree $\tau : X \to \mathbb{R}$ with T mutually disjoint terminal regions $G_j \subset X$, $j \in \{1, \dots T\}$, or leaves, as a set of weights $\{w_{G_1}, \dots w_{G_T}\}$, where the mapping from an input $\mathbf{x} \in X$ to a region is a given by $\xi : X \to \mathcal{G}$, where $\mathcal{G} \triangleq \{G_1, \dots G_T\} \subset 2^X$. Therefore, for some $\mathbf{x} \in X$, we may write $\tau(\mathbf{x}) = w_{\xi(\mathbf{x})}$. Alternatively, we may also write $\tau(\mathbf{x}) = \sum_{j=1}^{T} w_{G_j} \mathbb{I}\{\mathbf{x} \in G_j\}$. It is important to note that we assume the tree structure represented by ξ is constructed through recursive binary splits.

Given a set of training examples $\mathcal{D}$, it is clear by definition that gradient tree boosting approximates the conditional expectation term in (2.6) roughly by fitting a set of T constants $\{\omega_{G_1}, \dots \omega_{G_T}\}$. However, instead of finding a single $\beta_m \triangleq \gamma_m / |\mathcal{D}| > 0$ as in (2.8) to approximate $\gamma_m \psi_{\widetilde{X}}$, a tree can naturally find T nonnegative step sizes $\beta_m^{(j)}$, $\forall j \in \{1, \dots T\}$ when fitting the final tree weights $w_{G_1}, \dots w_{G_T}$, where $w_{G_j} \triangleq \beta_m^{(j)} \omega_{G_j}$. We may now write an algorithm for gradient tree boosting that is a straightforward modification of the generic algorithm given in section 2.4 to include the fitting of the $\beta_m^{(j)}$ constants for each region $G_1, \dots G_T$.

Algorithm 2: Gradient tree boosting

1. Define an initial constant model $F_0 \triangleq \kappa \in \mathbb{R}$, the loss function $L : Y \times \mathbb{R} \to \mathbb{R}_+$, and the impurity measure $Q : 2^X \times 2^{X \times Y} \to \mathbb{R}_+$. Choose a fixed maximum tree size $T \in \mathbb{N}$, where typically we have $T \triangleq 2^d$, where $d \in \mathbb{N}$ is the maximum depth of the regression tree.

2. For each boosting stage $m = 1, \dots M$,

   (a) For $k = 1, \dots |\mathcal{D}|$, compute the loss function partial derivatives

   $$\partial_{F_{m-1}(\mathbf{x}_k)} L_k \triangleq \frac{\partial L}{\partial F_{m-1}(\mathbf{x}_k)}(y_k, F_{m-1}(\mathbf{x}_k))$$

   Then, fit a regression tree $\phi_m : X \to \mathbb{R}$ to the negative loss function partials to yield a set of terminal regions $\mathcal{G}_m \triangleq \{G_m^{(1)}, \dots G_m^{(T_m)}\}$, $T_m \leq T$, with an input to leaf map $\xi_m : X \to \mathcal{G}_m$, where

   $$\mathcal{G}_m = \arg \min_{\{G_m^{(1)}, \dots G_m^{(T_m)}\}} \sum_{j=1}^{T_m} Q(G_j, \mathcal{D})$$

   (b) For $j = 1, \dots T_m$, fit the terminal region weights $w_{G_m^{(j)}} \in \mathbb{R}$ for each terminal region $G_m^{(j)}$, where

   $$w_{G_m^{(j)}} = \arg\min_{w} \sum_{k \in I_{G_j}} L(y_k, F_{m-1}(\mathbf{x}_k) + w)$$

   Here we defined the index set $I_{G_j} \triangleq \{k \in \mathbb{N} : \mathbf{x}_k \in G_j, (\mathbf{x}_k, y_k) \in \mathcal{D}\}$.

   (c) Define the updated model F_m, where $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \eta_m \phi_m(\mathbf{x})$, with $\eta_m > 0$ the learning rate. We may also equivalently write the tree $\phi_m(\mathbf{x})$ as $w_{\xi_m(\mathbf{x})}$ or $\sum_{j=1}^{T_m} w_{G_m^{(j)}} \mathbb{I}\{\mathbf{x} \in G_m^{(j)}\}$.

3. Output the final model $F : X \to \mathbb{R}$, where $F \triangleq F_M$ and

   $$F(\mathbf{x}) = \kappa + \sum_{m=1}^{M} \eta_m \phi_m(\mathbf{x}) = \kappa + \sum_{m=1}^{M} \eta_m w_{\xi_m(\mathbf{x})} = \kappa + \sum_{m=1}^{M} \eta_m \sum_{j=1}^{T_m} w_{G_m^{(j)}} \mathbb{I}\{\mathbf{x} \in G_m^{(j)}\}$$

As usual, the initial constant model $F_0 \triangleq \kappa \in \mathbb{R}$ is problem specific, where for regression one usually chooses $\kappa = \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} y_k$ and $\kappa = 0$ for binary classification. We note that one point of interest is how to define the impurity measure Q. Typically, mean squared error is often used, so for some nonempty $H \subset X$, we have

$$Q(H, \mathcal{D}) = \frac{1}{|H|} \sum_{k \in I_H} \left[-\partial_{F_{m-1}(\mathbf{x}_k)} L_k + \frac{1}{|H|} \sum_{k \in I_H} \partial_{F_{m-1}(\mathbf{x}_k)} L_k \right]^2 \tag{3.6}$$

Here $I_H \triangleq \{k \in \mathbb{N} : \mathbf{x}_k \in H, (\mathbf{x}_k, y_k) \in \mathcal{D}\}$, and (3.6) incorporates the fact the quantity that minimizes the total squared error for a region H is simply $\frac{1}{|H|}\sum_{k\in I_H} -\partial_{F_{m-1}(\mathbf{x}_k)} L_k$ as expected.

It then follows that given this definition of Q, the optimal split point $\mathcal{S} \triangleq (u^*, v^*)$ of a nonempty region $H \subset X$ at the u^*th training example in $\mathcal{D}$ by the v^*th feature column into the disjoint regions $H_L(u^*, v^*)$ and $H_R(u^*, v^*)$, assuming that we have $X \subseteq \mathbb{R}^n$, is such that

$$\mathcal{S} = \min_{u\in\{1,\dots|\mathcal{D}|\},v\in\{1,\dots n\}} \left\{ \frac{|H_L|}{|H|} Q(H_L(u,v),\mathcal{D}) + \frac{|H_R|}{|H|} Q(H_R(u,v),\mathcal{D}) \right\}$$

Here $H_L(u,v) \triangleq \{\mathbf{x} \in X : x_v < (\mathbf{x}_u)_v, (\mathbf{x}_u, y_u) \in \mathcal{D}\}$ and $H_R \triangleq \{\mathbf{x} \in X : x_v \geq (\mathbf{x}_u)_v, (\mathbf{x}_u, y_u) \in \mathcal{D}\}$. Note that the normalization constants $|H_L|/|H|$ and $|H_R|/|H|$ are necessary since we defined Q as mean, not total, squared error. Subsequent splits are determined by recursing into $H_L(u^*, v^*)$ and $H_R(u^*, v^*)$.

References

[1] Bagnell, D. (2012, Nov 14). *Functional Gradient Descent* [Lecture notes]. Carnegie Mellon University. http://www.cs.cmu.edu/~16831-f12/notes/F12/16831_lecture21_danielsm.pdf.

[2] Bartlett, P. (2008). *Reproducing Kernel Hilbert Spaces* [Lecture notes]. UC Berkeley. https://people.eecs.berkeley.edu/~bartlett/courses/281b-sp08/7.pdf

[3] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.

[4] Kang, K., Weinberger, C., & Cai, W. (2006, May 3). A Short Essay on Variational Calculus. http://micro.stanford.edu/~caiwei/Forum/2006-05-03-VarCalc/vari_calculus_v04.pdf.