Gradient Boosting and AnyBoost

date: 20261121
status: #finished
tags: #machine_learning #ensemble_learning #gradient_boosting

---

1. Boosting

(i) Strong vs. Weak Models

Boosting is an ensemble technique, just like bagging. It answers the question of how to combine a set of weak models (learners) from a hypothesis class $\mathcal{H}$ into a strong one.

In supervised learning, a model is trained by minimizing an objective, the empirical risk, defined as

$$R(H) = \frac{1}{n} \sum_i L(y_i, H(x_i))$$

where $L(y, F) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ (regression) or $L(y, F) : \{1, ..., K\} \times \mathbb{R}^K \to \mathbb{R}$ (classification) is a loss function, and H denotes the machine learning model, which in our case is the strong model.

A strong model is one with low empirical risk, and a weak model is one with high empirical risk. For example, a full-grown decision tree can be viewed as a strong model, and a decision stump (a decision tree with only one split) is a weak one.

(ii) What Is Gradient Boosting?

Gradient boosting (Friedman, 2001) answers how to combine weak models into a strong one: it uses each weak model as one step of gradient descent, and the strong model is obtained by adding these weak models one at a time. Specifically, we want to obtain a strong model H_T constructed as

$$H_T = H_0 + \sum_{t=1}^{T} \alpha_t h_t$$

where H_0 is the initial model. At each step t, the strong model H_{t-1} from the previous step is given, and a coefficient α_t and a weak model h_t are chosen to update H_{t-1}:

$$H_t = H_{t-1} + \alpha_t h_t$$

such that the empirical risk decreases:

$$R(H_t) \leq R(H_{t-1}).$$

Recall that in parameter optimization, the empirical risk is a function of the parameters θ, and gradient descent updates them as

$$\theta_t = \theta_{t-1} - \eta \nabla R(\theta_{t-1})$$

where η is the learning rate, chosen small enough that the empirical risk decreases. Likewise, to get $R(H_t) \leq R(H_{t-1})$ at each round t, we can apply gradient descent to the model itself. Conceptually, we compute the first-order derivative of the empirical risk at H_{t-1}, and obtain H_t by adding $\alpha_t > 0$ times the negative derivative. This suggests

$$h_t \approx -\frac{\mathrm{d}R}{\mathrm{d}H}(H_{t-1}).$$

This holds only approximately because h_t must belong to $\mathcal{H}$. α_t is either small enough or chosen by a line search, so that the empirical risk actually decreases.

This is gradient descent in function space. However, it is unclear how to compute a gradient with respect to a function. Interestingly, the gradient becomes easy to compute once we change the notation, as shown in Section 2.

The pipeline above is general, and different boosting models differ in how $\{(\alpha_t, h_t)\}_t$ are chosen.

---

2. Gradient Boosting

(i) From a Function of the Hypothesis to a Function of the Predictions

As shown above, it is not clear how to construct $h_t \approx -\frac{\mathrm{d}R}{\mathrm{d}H}(H_{t-1})$. The empirical risk is a function of a hypothesis (a function) H. Instead, we can define another empirical risk, $\tilde{R} : \mathbb{R}^n \to \mathbb{R}$ or $\mathbb{R}^{n \times K} \to \mathbb{R}$, whose input is the

vertically stacked predictions of a hypothesis:

$$\tilde{R}(F_1, F_2, ..., F_n) = \frac{1}{n}\sum_i L(y_i, F_i),$$

where $L(y, F)$ is assumed to be differentiable in F. With this notation, we characterize a hypothesis H_t by its predictions:

$$\begin{pmatrix} H_t(x_1) \\ H_t(x_2) \\ ... \\ H_t(x_n) \end{pmatrix},$$

and we update these predictions by gradient descent to minimize the empirical risk:

$$\begin{pmatrix} H_t(x_1) \\ H_t(x_2) \\ ... \\ H_t(x_n) \end{pmatrix} = \begin{pmatrix} H_{t-1}(x_1) \\ H_{t-1}(x_2) \\ ... \\ H_{t-1}(x_n) \end{pmatrix} + \alpha_t \begin{pmatrix} h_t(x_1) \\ h_t(x_2) \\ ... \\ h_t(x_n) \end{pmatrix}$$

where

$$\begin{pmatrix} h_t(x_1) \\ h_t(x_2) \\ ... \\ h_t(x_n) \end{pmatrix} \approx -\nabla \tilde{R}(H_{t-1}(x_1), H_{t-1}(x_2), ..., H_{t-1}(x_n)).$$

(ii) Simplified Notation

To simplify the notation, we let

$$H_t(X) = \begin{pmatrix} H_t(x_1) \\ H_t(x_2) \\ \vdots \\ H_t(x_n) \end{pmatrix} \in \mathbb{R}^n \text{ or } \mathbb{R}^{n \times K},$$

$$h_t(X) = \begin{pmatrix} h_t(x_1) \\ h_t(x_2) \\ \vdots \\ h_t(x_n) \end{pmatrix} \in \mathbb{R}^n \text{ or } \mathbb{R}^{n \times K},$$

$$L(y, H_t(X)) = \begin{pmatrix} L(y_1, H_t(x_1)) \\ L(y_2, H_t(x_2)) \\ \vdots \\ L(y_n, H_t(x_n)) \end{pmatrix} \in \mathbb{R}^n,$$

and

$$\frac{\partial L}{\partial F}(y, H_t(X)) = \begin{pmatrix} \frac{\partial L}{\partial F}(y_1, H_t(x_1)) \\ \frac{\partial L}{\partial F}(y_2, H_t(x_2)) \\ \vdots \\ \frac{\partial L}{\partial F}(y_n, H_t(x_n)) \end{pmatrix} \in \mathbb{R}^n \text{ or } \mathbb{R}^{n\times K}.$$

We consider the regression task here for simplicity. For multi-class classification, the derivation below still holds as **one** derivation, with $H_t(X)$, $h_t(X)$, and the gradient in $\mathbb{R}^{n\times K}$, and inner products sum over both samples and classes. XGBoost and LightGBM build one tree per class at each step, and then each tree's symbols are in $\mathbb{R}^n$. These K trees are **not** one-vs-all, as the gradient of each class still comes from the softmax over all K scores.

With the simplified notation,

$$\begin{aligned} h_t(X) &\approx -\nabla \tilde{R}(H_{t-1}(X)) \\ &= -\begin{pmatrix} \frac{\partial \tilde{R}}{\partial F_1}(H_{t-1}(X)) \\ \frac{\partial \tilde{R}}{\partial F_2}(H_{t-1}(X)) \\ \vdots \\ \frac{\partial \tilde{R}}{\partial F_n}(H_{t-1}(X)) \end{pmatrix} \\ &= -\begin{pmatrix} \frac{1}{n}\frac{\partial L}{\partial F}(y_1, H_{t-1}(x_1)) \\ \frac{1}{n}\frac{\partial L}{\partial F}(y_2, H_{t-1}(x_2)) \\ \vdots \\ \frac{1}{n}\frac{\partial L}{\partial F}(y_n, H_{t-1}(x_n)) \end{pmatrix} \\ &= -\frac{1}{n}\frac{\partial L}{\partial F}(y, H_{t-1}(X)). \end{aligned}$$

The updating rule becomes

$$H_t(X) \approx H_{t-1}(X) + \frac{\alpha_t}{n}\left[-\frac{\partial L}{\partial F}(y, H_{t-1}(X))\right].$$

(iii) How Does Gradient Boosting Choose α_t and h_t?

This updating rule implies that, at each round t, we train a weak model to predict the negative gradient $-\frac{\partial L}{\partial F}(y, H_{t-1}(X))$ rather than the actual outcome $y \in \mathbb{R}^n$, and the predictions are updated iteratively by adding the predicted negative gradients. The constant $\frac{1}{n}$ can be dropped, because α_t rescales the step anyway. Eventually, we obtain predictions with low empirical risk.

Therefore,

$$h_t = \arg\min_{h \in \mathcal{H}} \left\| h(X) - \left[-\frac{\partial L}{\partial F}(y, H_{t-1}(X)) \right] \right\|^2,$$

and

$$\alpha_t = \arg\min_{\alpha \in \mathbb{R}} \tilde{R}(H_{t-1}(X) + \alpha h_t(X)).$$

Note that the gradient gives only a descent direction. It does not say how much to add to the previous prediction $H_{t-1}(x_i)$, and that step size is what α_t supplies by minimizing the empirical risk **along** $h_t(X)$. In practice, H_t is more often updated as

$$H_t = H_{t-1} + \nu \alpha_t h_t$$

where $\nu \in (0, 1]$ is a hyperparameter called the learning rate, and Friedman (2001) calls this scaling shrinkage. $\nu = 0$ is excluded because the model would never move.

(iv) Initialization

The initialization is

$$H_0 = \arg\min_{c \in \mathbb{R}} \frac{1}{n} \sum_i L(y_i, c).$$

That is, $\{H_0(x_i)\}_i$ are all equal to that optimal c.

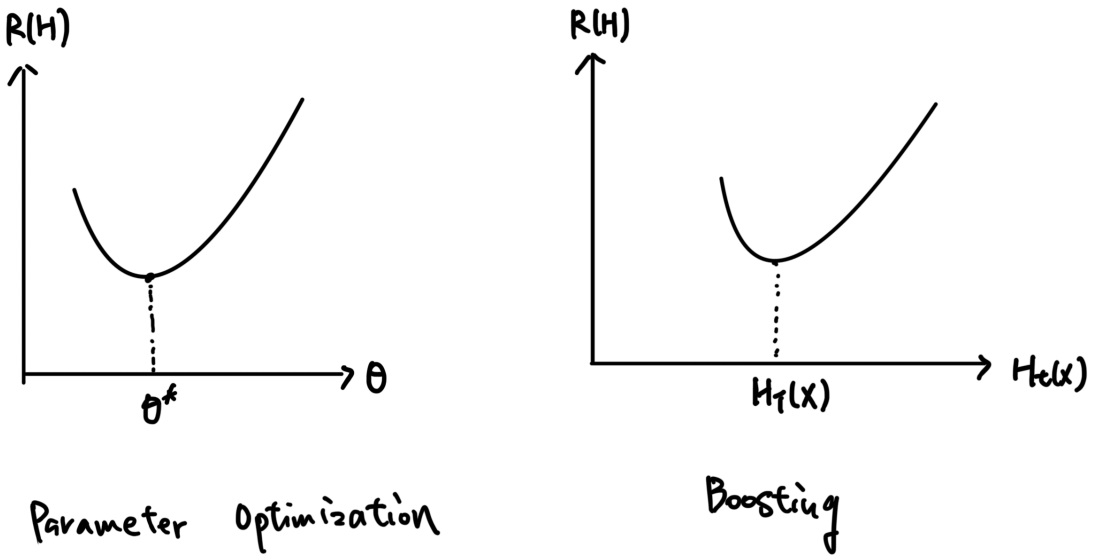
(v) Why Can the Empirical Risk Keep Decreasing?

We need more than one round because h_t is a weak model, so $\left\| h_t(X) - \left[-\frac{\partial L}{\partial F}(y, H_{t-1}(X)) \right] \right\|^2$ can be far from 0. At each t, we get only the best weak model in $\mathcal{H}$, and even with $\nu = 1$, α_t minimizes the empirical risk

only over the line $\{H_{t-1}(X) + \alpha h_t(X) : \alpha \in \mathbb{R}\}$, not over all of $\mathbb{R}^n$. So $\nabla \tilde{R}(H_t(X))$ is generally nonzero.

(vi) Analogy with Gradient Descent on Parameters

Remarkably, boosting can be formulated as optimizing the predictions by gradient descent. In parameter optimization, we iteratively adjust the parameters. Here, we iteratively adjust the predictions $H_{t-1}(X)$ by $\alpha_t h_t(X)$.



---

3. AnyBoost

AnyBoost (Mason et al., 2000) is a generalization of gradient boosting. It gives only a general principle for choosing $\{(\alpha_t, h_t)\}_t$, and gradient boosting follows that principle with a specific way to compute them. This section shows how gradient boosting generalizes to AnyBoost.

The procedure of AnyBoost is as follows.

Fix $H_0 = 0$, and iterate t from 1 to T {

1. Compute $\frac{\partial L}{\partial F}(y, H_{t-1}(X))$ based on H_{t-1}
2. Choose $h_t \in \mathcal{H}$ with $\left[\frac{\partial L}{\partial F}(y, H_{t-1}(X))\right]' h_t(X) < 0$
3. Choose a coefficient $\alpha_t > 0$
4. Update the model: $H_t = H_{t-1} + \alpha_t h_t$

}

4. Reference

- Llew Mason, Jonathan Baxter, Peter L. Bartlett, and Marcus R. Frean. 2000. **Boosting Algorithms as Gradient Descent**. In Advances in Neural Information Processing Systems 12 (NIPS 1999), 512–518. MIT Press. https://papers.nips.cc/paper/1766-boosting-algorithms-as-gradient-descent
- Jerome H. Friedman. 2001. **Greedy Function Approximation: A Gradient Boosting Machine**. The Annals of Statistics 29(5), 1189–1232. https://doi.org/10.1214/aos/1013203451
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. **The Elements of Statistical Learning**, 2nd ed., Chapter 10 (Boosting and Additive Trees). Springer. https://doi.org/10.1007/978-0-387-84858-7