Lecture Notes on Generalized Residuals and MSE-Fitting in Gradient Boosting

Oliver Dürr

19th November 2024

Status of this document: This document is a GTP-Auto generated draft (with supervision). Limitations include: consitency in notation, completeness, and links to modern libraries like CatBoost. See also Elements of Statistical Learning, for a more detailed treatment of Gradient Boosting.

Overview

Gradient Boosting is a powerful method for building predictive models, especially when using decision trees as weak learners. **It is fundamentally a two-step procedure:**

1. **Generalized Residuals:** Compute the gradient of the loss function with respect to the current predictions. This step determines the **generalized residuals** which indicate how the model should be corrected.

2. **MSE-Fitting:** Fit a new weak learner to these residuals using the Mean Squared Error (MSE) criterion. This step updates the model by learning to correct its errors.

This clear division into two steps is central to how Gradient Boosting iteratively refines predictions, combining the strengths of decision trees used as weak learner and gradient-based optimization.

Step 1: Computing Generalized Residuals r_i

Loss Function

The loss function $L(y_i, h(x_i))$ measures the discrepancy between the observed value y_i and the prediction $h(x_i)$. In the simple case $h(x_i)$ is a single point estimate $\hat{y}_i(x_i)$.[1] Examples include:

- Mean Squared Error (MSE) for regression.

- Log-Loss for classification.

[1] The same framework can be extended to probabilistic models that output entire distributions rather than single values. For example, in probabilistic regression, he loss function could measure the negative log-likelihood of the observed value under the predicted distribution, such as a Gaussian or Student-t distribution. Examples are given below.

Gradient of the Loss Function

The gradient of the loss function w.r.t. the predicted values meassures in which direction and how much the predictions should be adjusted. These are the gerenalized residual r_i. In a physics analogy, the gradient is the force towards the correct predictions.

For MSE, we use the loss function:

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

The derivative of the loss function with respect to the predicted value $\hat{y}$ is:

$$\frac{\partial L}{\partial \hat{y}} = \hat{y}_i - y_i, \quad r_i = y_i - \hat{y}_i.$$

Hence, the generalized residuals are the usual residuals $r_i = y_i - \hat{y}_i$ for regression problems. Generalized residuals for other loss functions are provided below.

Step 2: Fitting Using MSE

Goal

Fit a new weak learner $h(x)$ to the generalized residuals r_i by minimizing the Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(r_i - h(x_i))^2.$$

Here:

- n: Number of data points.
- $h(x_i)$: Prediction of the weak learner for data point i.

Process

1. Train a decision tree (or other weak learner) to approximate the generalized residuals r_i.
2. Minimize the MSE between the predictions of the weak learner and the residuals.

Combining the Steps

Updating Predictions

The model predictions are updated iteratively by adding the scaled output of the new weak learner:

$$\hat{y}_i^{\text{new}} = \hat{y}_i^{\text{old}} + \eta h(x_i),$$

where η is the learning rate that controls the step size.

Iterative Optimization

This process is repeated for a specified number of iterations or until convergence:

- Compute generalized residuals.
- Fit a weak learner to the residuals using MSE.
- Update predictions.

Notes on Gradient Boosting

- **Non-Differentiable Models:** Decision trees are not differentiable, so standard gradient descent cannot be applied. Instead, the optimization focuses on the predictions.
- **Gradient in Function Space:** Gradient Boosting optimizes in the function space by finding a function $h(x)$ that reduces the loss.
- **Flexibility:** The method accommodates various loss functions, provided their gradients can be computed.

Conclusion

The Gradient Boosting process involves:

- Calculating generalized residuals (gradients of the loss function).
- Fitting a weak learner to these residuals using MSE.
- Iteratively refining the model predictions.

This two-step approach allows Gradient Boosting to effectively optimize models that are non-differentiable, such as decision trees, while maintaining flexibility in handling different loss functions.

Gradient Boosting for 2-Class Classification

Gradient Boosting can be adapted for classification tasks, particularly binary classification, by using a procedure similar to logistic regression. In this context:

Predicted Output

The model's output $h(x)$ takes values between $-\infty$ and $+\infty$ and is often rerred to as logits. These logits are then passed through the logistic (sigmoid) function to map them to probabilities:

$$p(x) = \frac{1}{1 + e^{-h(x)}},$$

where $p(x)$ is the probability of the positive class.

Loss Function for Classification

For binary classification, as for all probabilistic models, the standard loss function is the **negative log-likelihood**, which is defined as the negative logarithm of the probability assigned to the observed class by the model. For binary classification, this can be compactly written as:

$$L(y, h(x)) = -\left[y \log(p(x)) + (1-y)\log(1-p(x))\right],$$

where:

- $y \in \{0, 1\}$: True class label.
- $p(x)$: Predicted probability of the positive class.

Generalized Residuals in Classification

The generalized residuals in classification are derived as the negative gradient of the Log-Loss with respect to the logits $h(x)$. This gives (see appendix A for the derivation):

$$r_i = y_i - p(x_i),$$

where r_i measures the difference between the true label y_i and the predicted probability $p(x_i)$.

Fitting the Residuals

As in regression, a weak learner (e.g., a decision tree) at stage m is trained to fit the residuals r_i by minimizing the Mean Squared Error:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(r_i - h_m(x_i))^2,$$

where $h_m(x)$ is the output of the m-th weak learner. The model's predictions (logits) are updated as before.

Connecting to Logistic Regression

This procedure is analogous to logistic regression but enhances its flexibility by iteratively learning corrections with weak learners. Unlike logistic regression, which assumes a linear decision boundary, Gradient Boosting learns complex, non-linear boundaries through the ensemble of weak learners.

Multiclass Classification

Gradient Boosting can also be extended to handle multiclass classification problems. Instead of predicting a single logit for binary classification, the model predicts a vector of logits $\mathbf{h}(x)$, one for each class.

Predicted Output

For multiclass classification with K classes, the model outputs a vector of logits $\mathbf{h}(x) = [h_1(x), h_2(x), \ldots, h_K(x)]$, where $h_k(x)$ represents the logit for class k. These logits are converted into class probabilities using the softmax function:

$$p_k(x) = \frac{\exp(h_k(x))}{\sum_{j=1}^{K} \exp(h_j(x))},$$

where $p_k(x)$ is the predicted probability of class k.

Loss Function for Multiclass Classification

The loss function for multiclass classification is the **negative log-likelihood**, defined as:

$$L(y, \mathbf{h}(x)) = -\log(p_y(x)),$$

where:

- y: True class label, an integer in $\{1, 2, \ldots, K\}$.
- $p_y(x)$: Predicted probability for the true class y, computed from the softmax output.

Generalized Residuals for Multiclass Classification

The generalized residuals are computed as the negative gradient of the loss function with respect to each logit $h_k(x)$:

$$r_{ik} = \begin{cases} 1 - p_k(x_i) & \text{if } y_i = k, \\ -p_k(x_i) & \text{if } y_i \neq k. \end{cases}$$

Here:

- r_{ik}: Residual for the k-th class and the i-th data point.
- $p_k(x_i)$: Predicted probability of class k for data point i.

Fitting the Residuals

A separate weak learner is trained for each class k to fit the residuals r_{ik}. For each class, the weak learner minimizes the Mean Squared Error:

$$\text{MSE}_k = \frac{1}{n} \sum_{i=1}^{n} (r_{ik} - h_{m,k}(x_i))^2,$$

where $h_{m,k}(x_i)$ is the prediction of the m-th weak learner for class k and data point i. Updating the model is as before.

Conclusion

Gradient Boosting for multiclass classification follows the same two-step approach as binary classification:

1. Compute generalized residuals for each class using the gradient of the negative log-likelihood.

2. Fit a weak learner for each class to the residuals.

This method enables Gradient Boosting to handle multiclass problems effectively while maintaining the flexibility and interpretability of the binary case.

Other Loss Functions

One of the key strengths of Gradient Boosting is its flexibility. Many loss functions can be used to tailor the algorithm to specific applications, beyond the standard Mean Squared Error (MSE) or Log-Loss. For instance, in fields where domain-specific performance metrics are crucial, custom loss functions can be designed to directly optimize those metrics.

Modern Gradient Boosting libraries, such as CatBoost, provide APIs for implementing custom loss functions. This allows users to go beyond pre-defined objectives and adapt the model to their unique needs. For example, CatBoost enables you to define a custom loss function and seamlessly integrate it into the training process.

For more details on how to implement custom loss functions in CatBoost, refer to the official documentation:

`https://catboost.ai/docs/concepts/loss-functions.html`

This feature demonstrates the true flexibility of Gradient Boosting, making it suitable for a wide range of tasks, from regression and classification to specialized optimization problems in real-world domains like document retrieval.

A Appendix A: Derivation of Generalized Residuals for Binary Classification

The loss function for binary classification is the negative log-likelihood:

$$L(y, h(x)) = -\left[y \log(p(x)) + (1-y)\log(1-p(x))\right],$$

where $p(x) = \frac{1}{1+e^{-h(x)}}$ is the probability of the positive class.

The generalized residuals are defined as the negative gradient of the loss function with respect to the logits $h(x)$:

$$r_i = -\frac{\partial L}{\partial h(x)}.$$

Step-by-Step Derivation

AUTO-GENERATED

1. **Compute the partial derivative of L with respect to $p(x)$:**

$$\frac{\partial L}{\partial p(x)} = -\frac{y}{p(x)} + \frac{1-y}{1-p(x)}.$$

2. **Compute the derivative of $p(x)$ with respect to $h(x)$:**

$$\frac{\partial p(x)}{\partial h(x)} = p(x)(1-p(x)).$$

3. **Apply the chain rule:**

$$\frac{\partial L}{\partial h(x)} = \frac{\partial L}{\partial p(x)} \cdot \frac{\partial p(x)}{\partial h(x)}.$$

Substitute the results from steps 1 and 2:

$$\frac{\partial L}{\partial h(x)} = \left(-\frac{y}{p(x)} + \frac{1-y}{1-p(x)}\right) \cdot p(x)(1-p(x)).$$

4. **Simplify the expression:** Using algebra, the above simplifies to:

$$\frac{\partial L}{\partial h(x)} = p(x) - y.$$

Thus, the generalized residuals are:

$$r_i = -\frac{\partial L}{\partial h(x)} = y_i - p(x_i).$$

Conclusion

This derivation shows that the generalized residuals for binary classification are simply the difference between the observed class label y_i and the predicted probability $p(x_i)$. This forms the basis for updating weak learners in Gradient Boosting.