Random Forest (RF)

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mean prediction (regression) or majority vote (classification) of the individual trees. The key steps in RF are:

1. **Bootstrap Sampling:** Generate multiple datasets by sampling with replacement from the original training data.
2. **Training Decision Trees:** Each tree is trained independently on a different bootstrapped dataset.
3. **Prediction Aggregation:** Combine the predictions of all the individual trees.

Prediction Aggregation

$$\hat{y}_i = \frac{1}{B} \sum_{b=1}^{B} f_b(x_i; \theta_b)$$

Optimization Problem for Training Tree b

Each tree b is trained to minimize the following objective function:

$$\min_{\theta_b} \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \Omega(\theta_b)$$

where $\hat{y}_i = f_b(x_i; \theta_b)$ is the prediction of tree b.

Gradient Boosting Decision Trees (GBDT)

Gradient Boosting Decision Trees is an ensemble technique that builds trees sequentially, where each new tree corrects the errors of the previous trees. The key steps in GBDT are:

1. **Initial Prediction:** Start with an initial prediction, usually the mean of the target values.
2. **Additive Model:** Build the model in a stage-wise manner by adding one tree at a time.
3. **Gradient Descent:** Use gradient descent to optimize the loss function.

Ensemble Prediction

The final prediction is the sum of the predictions from all the trees:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i; \theta_k)$$

Optimization Problem for Training the Ensemble

The objective function for training the entire ensemble is:

$$\min_{\{\theta_k\}_{k=1}^{K}} \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(\theta_k)$$

where $\hat{y}_i = \sum_{k=1}^{K} f_k(x_i; \theta_k)$.

Objective for Optimizing Tree k

The objective function for optimizing the k-th tree is:

$$F_k(\theta_k) = \sum_{i=1}^{n} \left[\ell(y_i, \hat{y}_i^{(k-1)} + f_k(x_i)) + \Omega(\theta_k) \right]$$

where $\hat{y}_i^{(k-1)} = \sum_{j=1}^{k-1} f_j(x_i)$.

Taylor Expansion of Loss Function

Using Taylor expansion, the loss function can be approximated as:

$$\ell(y_i, \hat{y}_i^{(k-1)} + f_k(x_i)) \approx \ell(y_i, \hat{y}_i^{(k-1)}) + g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i)$$

where g_i and h_i are the first and second derivatives of $\ell(y_i, \hat{y}_i^{(k-1)})$ with respect to $\hat{y}_i^{(k-1)}$:

$$g_i = \frac{\partial \ell(y_i, \hat{y}_i^{(k-1)})}{\partial \hat{y}_i^{(k-1)}}, \quad h_i = \frac{\partial^2 \ell(y_i, \hat{y}_i^{(k-1)})}{(\partial \hat{y}_i^{(k-1)})^2}$$

Regularization Term

The regularization term penalizes the complexity of the model:

$$\Omega(\theta_k) = \gamma |L_k| + \frac{1}{2} \lambda \|w_k\|_2^2$$

where $|L_k|$ is the number of leaf nodes in tree k, and w_k is the vector of leaf weights.

Leaf Weight Optimization

The optimal weight for each leaf node is given by:

$$w_j^k = -\frac{G_j}{H_j + \lambda}$$

where G_j and H_j are the sums of the gradients and hessians for all data points in leaf j:

$$G_j = \sum_{i \in I_j} g_i, \quad H_j = \sum_{i \in I_j} h_i$$

Gain for Splitting a Node

The gain from splitting a node is calculated as:

$$\text{Gain} = \frac{1}{2} \left(\frac{G_{j(L)}^2}{H_{j(L)} + \lambda} + \frac{G_{j(R)}^2}{H_{j(R)} + \lambda} - \frac{G_j^2}{H_j + \lambda} \right) - \gamma$$

where $j(L)$ and $j(R)$ are the left and right children of node j.