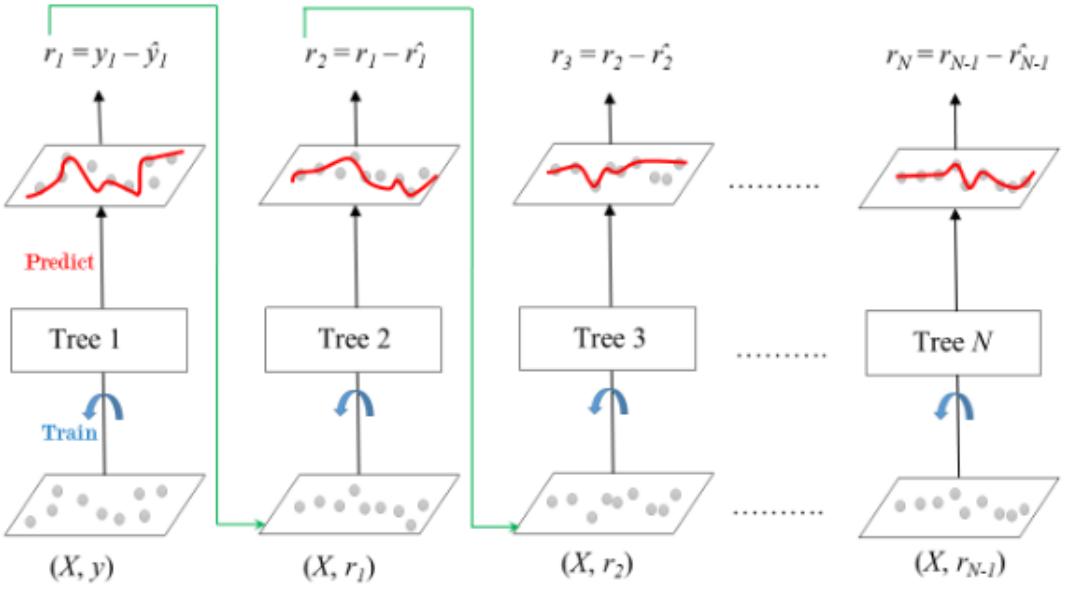
Gradient Boosting

Gradient Boosting is an ensemble learning technique that builds a series of weak learners (typically decision trees) sequentially to create a strong predictive model.

Here's an explanation of how it works

1. Initial prediction: Start with a simple model that makes an initial prediction.
2. Calculate residuals: Compute the difference between the actual values and the predicted values (residuals).
3. Train weak learner: Fit a weak learner (e.g., a shallow decision tree) to predict these residuals.
4. Update predictions: Add the predictions of the weak learner to the existing model's predictions, typically scaled by a learning rate.
5. Iterate: Repeat steps 2-4 for a specified number of iterations, each time fitting a new weak learner to the residuals of the current ensemble.
6. Final model: The final model is the sum of the initial prediction and all the weak learners' contributions.



Tree1 is trained using the feature matrix *X* and the labels *y*. The predictions labeled *y1(hat)* are used to determine the training set residual errors *r1*. Tree2 is then trained using the feature matrix *X* and the residual errors *r1* of Tree1 as labels. The predicted results *r1(hat)* are then used to determine the residual *r2*. The process is repeated until all the M trees forming the ensemble are trained.

Since all trees are trained now, predictions can be made. Each tree predicts a label and the final prediction is given by the formula,

$$y(pred) = y1 + (eta * r1) + (eta * r2) + ... + (eta * rN)$$

Advantages

1. High performance: Gradient boosting often achieves state-of-the-art results on many machine learning tasks.
2. Handles non-linear relationships well: It can capture complex interactions between features.
3. Feature importance: It provides a natural way to measure feature importance.
4. Flexibility: Can be used for both regression and classification problems.
5. Robust to outliers: The sequential nature of the algorithm makes it less sensitive to outliers.

Disadvantages

1. Computationally intensive: Training can be slow, especially for large datasets.
2. Risk of overfitting: Without proper tuning, it can easily overfit the training data.
3. Sensitivity to hyperparameters: Performance can vary significantly based on hyperparameter choices.
4. Less interpretable: While feature importance is available, the overall model can be complex and hard to interpret.
5. Memory intensive: It requires the entire dataset to be in memory for efficient training.