Gradient Boosted Regression Trees

CART trees have many strong advantages:

- they are extremely fast to build and to execute,
- they are insensitive to feature scaling, and
- they require very little storage.

However, they are typically not competitive by themselves because it is hard to find the right tradeoff between variance and bias. If the tree is built without restricting its depth or pruning, it will almost always achieve zero training error (after accounting for noise in labels). Therefore, a full depth tree will overfit to the training set. In other words, this model will have *low bias* but *high variance*.

If, on the other hand, a tree is built with limited depth (e.g. max. depth 4), the classifier will be unable to model the training set effectively, Therefore, this model will have *high bias* but *low variance*.

Gradient Boosted Regression Trees (GBRT) is an algorithm that results from boosting applied to regression trees — similar to Random Forests, which is bagging applied to regression trees.

Gradient boosting reduces the bias of a classifier, analogous to the way bagging reduces its variance.

GBRT learns an ensemble of trees, and each tree is trained to correct the *residual* (the remaining error) of the existing ensemble.

Unfortunately, tuning the max. depth hyperparameter is a crude way of trading off bias with variance. Because the depth is inherently integral in nature, we cannot perform the fine-grained adjustment that is typically necessary to balance bias and variance.

Random Forests, as introduced in the previous module, allow us to circumvent this problem. Instead of worrying about the bias-variance tradeoff while training trees, we train many full trees on bootstraps (with a slightly modified splitting criterion). The average prediction of the ensemble of trees shows lower variance than a single tree would exhibit.

Gradient Boosted Regression Trees (GBRT) is a similar algorithm, but it does not use **bagging to reduce variance**. Instead, it uses **boosting to reduce bias.**

Boosting Weak Learners

Boosting is a general method to reduce the bias of "weak learners". Weak learners are essentially classifiers that cannot achieve zero training error on a given data set — a typical sign for high bias. Similar to bagging, boosting combines many such "weak" classifiers together into an ensemble. However, this time we assign weights to each weak learner. More specifically, let our individual classifiers be $h_1, \ldots, h_m$ with weights $\alpha_1, \ldots, \alpha_m$, where each $\alpha_j > 0$. The ensemble classifier is a weighted average of the individual ones.

$$H(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j h_j(\mathbf{x})$$

The key to boosting is to understand how the classifiers h_j and weights α_j are obtained.

Gradient Boosted Regression Trees (GBRT)

Boosting is a general concept similar to bagging, and there are many different specific boosting algorithms. Here we will focus on one of the most popular algorithms, Gradient Boosted Regression Trees (GBRT). Similar to Random Forests, which is essentially bagging for CART trees, GBRT is gradient boosting for CART trees.

In GBRT, **each tree h_j at iteration j is trained not to predict the label y_i directly, but to predict the *residual* of the current ensemble classifier**. In other words, for each data point $(\mathbf{x}_i, y_i)$, the tree h_j trains on the pair $(\mathbf{x}_i, r_i)$, where r_i is the residual $r_i = y_i - H(\mathbf{x}_i)$ between the label and the prediction of the current ensemble classifier.

We initialize the ensemble classifier to predict $H(\mathbf{x}_i) = 0$ for all inputs. This means that the very first residual is $r_i = y_i$ for all inputs. However, because the trees with limited depth have high bias, we know they have high training error. In other words, they won't be able to predict the training labels correctly, and there will be a residual. The trees at the second iteration are trained to predict exactly this remainder. This second iteration of trees are added to the ensemble with a weight, and so it continues with each iteration. Note that the weights α_j are set to a constant $\alpha_j = \alpha$ identical for all trees.

Here is the pseudocode for GBRT and a visualization of a GBRT classifier trained on a binary "spiral" data set.

procedure GBRT($D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, maxdepth, m, α)
  $H \leftarrow 0$
  for $j = 1 : m$ **do**
    for $i = 1 : n$ **do**
      $t_i \leftarrow y_i - H(\mathbf{x}_i)$
    end for
    $h_j \leftarrow \text{CART}(\{(\mathbf{x}_i, t_i)\}_{i=1}^{n}, \text{maxdepth})$
    $H \leftarrow H + \alpha h_j$
  end for
  return H
end procedure