Random Forests & Gradient Boosting

Random Forests

- Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.
- The generalization error for forests converges to a limit as the number of trees in the forest becomes large; depends on the strength of the individual trees in the forest and the correlation between them.

A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(x, \theta_k), k = 1, ...\}$ where the $\{\theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x.

Use of the Strong Law of Large Numbers shows that they always converge so that overfitting is not a problem.

The accuracy of a random forest depends on the strength of the individual tree classifiers and a measure of the dependence between them.

The results are insensitive to the number of features selected to split each node. Usually, selecting one or two features gives near optimum results.

Adaboost: Determine for classifier G_m a weight $\alpha_m = \log(\frac{1-err_m}{err_m})$, then $G(x) = \Sigma\alpha_m G_m$

Adaboost has no random elements and grows an ensemble of trees by successive reweightings of the training set where the current weights depend on the past history of the ensemble formation. But just as a deterministic random number generator can give a good imitation of randomness, Adaboost is emulating a random forest.

Random forests for regression: $\{h(x, \theta_k)\}$ takes on numerical values instead of class labels

Gradient Boosting

- algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction

Consider a gradient boosting algorithm with M stages. At each stage $m(1 \leq m \leq M)$ of gradient boosting, suppose some imperfect model F_m. In order to improve F_m, our algorithm should add some new estimator $h_m(x)$. Thus

$$F_{m+1}(x_i) = F_m(x_i) + h_m(x_i) = y_i$$

or equivalently,

$$h_m(x_i) = y_i - F_m(x_i).$$

Therefore, gradient boosting will fit h_m to the residual $y_i - F_m(x_i)$. Observe that residuals $h_m(x_i)$ are proportional to the negative gradients of the mean squared error (MSE) loss:

$$L_{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - F(x_i))^2$$

$$-\frac{\partial L_{MSE}}{\partial F(x_i)} = \frac{2}{n}(y_i - F(x_i)) = \frac{2}{n} h_m(x_i).$$