Tree-based Models for Regression

- Regression trees
- Regression forests
  - `randomForest` based on bagging
  - `gbm` based on boosting

Bagging (Bootstrap Aggregation)

- Training data: $\mathbf{Z} = \{(x_i, y_i)_{i=1}^n\}$
- Bootstrap samples[a]: $\mathbf{Z}^{*b} = \{(x_i^{*b}, y_i^{*b})_{i=1}^n\}$, where $b = 1 : B$

$$
\begin{array}{lcllll}
\mathbf{Z}^{*1} & : & (x_1^{*1}, y_1^{*1}), & (x_2^{*1}, y_2^{*1}), & \cdots, & (x_n^{*1}, y_n^{*1}) \\
\mathbf{Z}^{*2} & : & (x_1^{*2}, y_1^{*2}), & (x_2^{*2}, y_2^{*2}), & \cdots, & (x_n^{*2}, y_n^{*2}) \\
\vdots & & & & & \\
\mathbf{Z}^{*B} & : & (x_1^{*B}, y_1^{*B}), & (x_2^{*B}, y_2^{*B}), & \cdots, & (x_n^{*B}, y_n^{*B})
\end{array}
$$

[a] sample with replacement from $\mathbf{Z}$.

Bagging (Bootstrap Aggregation)

- Training data: $\mathbf{Z} = \{(x_i, y_i)_{i=1}^n\}$
- Bootstrap samples[a]: $\mathbf{Z}^{*b} = \{(x_i^{*b}, y_i^{*b})_{i=1}^n\}$, where $b = 1 : B$
- $\hat{f}^{*b}$: classification/regression function trained by $\mathbf{Z}^{*b}$
- The bagging estimate is defined to be

$$\hat{f}_{\mathsf{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}.$$

- Advantage: reduce variance. So works well for high-variance, low-bias procedures, such as trees.

[a]sample with replacement from $\mathbf{Z}$.

Random Forest

1. For $b = 1 : B$:
   (a) Draw a bs sample $\mathbf{Z}^{*b}$ from the training data.
   (b) Grow a BIG tree T_b (with some restriction).
2. Output the forest $\{T_b\}_{b=1}^{B}$.

To make a prediction at a new point x

Regression: $\frac{1}{B}\sum T_b(x)$.

Restriction when growing a tree in the forest:

- At each split, randomly select m variables from the p variables, and then pick the best split among them.
- The recommended value for m is $\sqrt{p}$ for classification and $p/3$ for regression.
- Purpose: reduce the correlation between trees in the forest.

Out-of-Bag (OOB) Samples

- OOB samples: sample points which are not included in $\mathbf{Z}^{*b}$, i.e., they are not used in building the tree T_b
- The OOB samples can be used to get a test error for T_b.
- The prediction and error rate returned by randomForest are calculated based on OOB. The error is usually close to a CV error.

Variable Importance

- Measure the importance of a variable by the improvement of RSS contributed by this variable.
- At each split, attribute the improvement of RSS to the corresponding splitting variable.
- For each variable, accumulate its improvement of RSS across the tree and then averaged over all the tress in the forest.

- Another measure is computed from permuting OOB samples: For each tree T_b in the forest, calculate the prediction error (MSE for regression) based on OOB samples. Then the same is done after permuting the jth predictor in the OOB samples. The difference between the two (before and after permutation) is then averaged over all trees, and further normalized by the corresponding standard deviation[a].

[a]If the standard deviation of the differences is equal to 0 for a variable, then the division is not applied.

Boosting Trees

- Boost the performance of a set of weak regression trees by cleverly combing them.
- Forward stagewise additive modeling: consider an additive model,

$$F(x) = f_1(x) + f_2(x) + \cdots + f_{T-1}(x) + f_T(x).$$

It is difficult to solve for all f_t's. Instead we solve it using a forward stagewise greedy algorithm.

Forward Stagewise Optimization

1. $F(x) = 0$ and record the current residual $r_i^{(0)} = y_i$
2. For $t = 1$ to T
   - Fit a regression tree f_t to the current residual $r_i^{(t-1)}$
   - Add f_t to F: $F = F + f_t$
   - Update the current residual $r_i^{(t)} = r_i^{(t-1)} - f_t(x_i)$

Tuning parameters for GBM: learning rate η, number of trees T, complexity of f_t's (depth of trees), and subsampling rate.

- Advantages of ensemble methods based on trees
  - Less-processing is needed, e.g., NA can be handled automatically, and no scaling/normalization is required
  - Can handle large number of predictors
- GBM vs randomForest
  - randomForest has less number of tuning parameters, while GBM has more, but with proper tuning, GBM can perform better than randomForest.
- Categorical Predictors
  - Each package treats categorical predictors differently: maximal 32 levels for randomForest and 1024 levels for GBM; XGBoost and some python packages only take numerical input.