STATISTICAL LEARNING

Today
1. Cons & Pros of trees
2. Bagging
3. Random Forest

1. Cons & Pros of trees

- Trees are very easy to explain to people. In fact, they are even easier to explain than ~~~~-linear regression!
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small)
- Tree can easily handle qualitative predictors without the need to create dummy variables.
- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

⇒ However, by aggregating many decision trees, the predictive performance of trees can be substantially improved.

2. Bagging

- Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
- Recall that given a set of n independent observations $Z_1, \cdots Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations

is given by σ^2/n

- In other words, averaging a set of observations reduces variance.
  ⇒ of course, this is not practical because we generally do not have access to multiple ~~traing~~ training data sets.
- instead we can bootstrap, by taking repeated samples from the (single) training data set.
  - In this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x. We then average all the predictions to obtain

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

⇒ This is called bagging.

* Bagging classification trees
- For classification trees: for each test observations, we record the Class predicted by each of the B trees, and take a majority vote:
  ⇒ the overall prediction is the most commonly occurring class among the B predictions

* Out-of-bag Error Estimation

There is a very straightforward way to estimate the test error of a bagged model

- Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. One can show that on average, each bagged tree makes use ~~of~~ around two-thirds of the observations
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations.
- predict the response for the ith observation using each of the trees in which that observation was OOB.
  ⇒ This will yield around B/3 predictions for the ith

observation, which we average.

- This estimate is essentially the LOO cross-validation error for bagging if B is large.

3. Random Forests

- Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees.

  ⇒ This reduce the variance when we average the trees.

- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, "a random selection of m predictors" is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.
- A fresh selection of m predictors is taken at each split, and typically, we choose $m \approx \sqrt{p}$. i.e. the number of predictors considered at each split is approximately equal to the square root of the total # of predictors.

4. Boosting

- like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. We only discuss boosting for decision trees
- Boosting works in a similar way, except that the trees are grown sequentially; each tree is grown using information from previous grown trees.

* Boosting Algorithm for regression trees

Step1. Set $\hat{f}(x)=0$ and $r_i = y_i$ for all i in the training set.

Step 2. For $b=1, 2, \cdots, B$ repeat:

2.1. Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r)

2.2 Update $\hat{f}$ by adding ~~th~~ a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

2.3. Update the residuals

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

Step 3. Output the boosted model

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

* Basic idea behind this approach?
  - Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.
  - Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.
  - Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.
  - By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well.
  - The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

⇒ R: gba

* Tuning parameters for boosting

① The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. we use cross-validation to select B.

② The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. very small λ can require using a very large value of B in order to achieve good performance.

③ The number of splits d in each tree, which controls the complexity of the boosted ensemble. Often $d=1$ works well, in which case each tree is a stump, consisting of a single split and resulting in an additive model. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

* Variable importance measure

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.
- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.