BAGGING, RANDOM FOREST, BOOSTING

Mr. S. B. Adnaik (Ph.D., Statistics)

Bagging, Random Forest, Boosting

- Bagging
- Random Forest
- Boosting

Bagging, Random Forest, Boosting

- Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method; let's introduce it here because it is particularly useful and frequently used in the context of decision trees.

- Note that given a set of n independent observations Z_1, Z_2, ...Z_n, each with variance σ^2, the variance of the mean 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 training sets.

Bagging, Random Forest, Boosting

- 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 b^{th} 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, Random Forest, Boosting

- The above prescription i.e. averaging predictions is applied to regression trees.
- For classification trees: for each test observation, 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.

Bagging, Random Forest, Boosting

Out-of-Bag Error Estimation:

- It turns out that 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.

Bagging, Random Forest, Boosting

Out-of-Bag Error Estimation:...

- We can predict the response for the i^{th} observation using each of the trees in which that observation was OOB. This will yield around B/3 predictions for the i^{th} observation, which we average.
- This estimate is essentially the LOO (Leave One Out) cross-validation error for bagging, if B is large.

Bagging, Random Forest, Boosting

- Bagging
- Random Forest
- Boosting

Bagging, Random Forest, Boosting

- Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. This reduces 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 ~ sqrt(p), that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors

Bagging, Random Forest, Boosting

- Bagging
- Random Forest
- Boosting

Bagging, Random Forest, 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.
- Note that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Notably, each tree is built on a bootstrap data set, independent of the other trees.
- Boosting works in a similar way, except that the trees are grown sequentially: each tree is grown using information from previously grown trees.

Bagging, Random Forest, Boosting

- Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
- For b = 1, 2, ... ,B, repeat:
  - Fit a tree $\hat{f}^b$ with d splits (d + 1 terminal nodes) to the training data (X, r).
  - Update $\hat{f}$ by adding in a shrunken version of the new tree:

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

  - Update the residuals,

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

- Output the boosted model,

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

Bagging, Random Forest, Boosting

Idea behind Boosting:

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially over-fitting, 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.

Bagging, Random Forest, Boosting

Idea behind Boosting:...

- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter slows the process down even further, allowing more and different shaped trees to attack the residuals.

Boosting for classification is similar in spirit to boosting for regression, but is a bit more complex.

Bagging, Random Forest, Boosting

Tuning parameters for boosting

- The number of trees B. Unlike bagging and random forests, boosting can over fit if B is too large, although this over fitting tends to occur slowly if at all. We use cross-validation to select B.
- 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.

Bagging, Random Forest, Boosting

Tuning parameters for boosting

- 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.

Bagging, Random Forest, Boosting

R/Python Session

Thank You