Lecture 19

Ensemble Methods: Random Forests

Murat Unal

Johnson Graduate School of Management
Cornell University

11/04/2021

Agenda

Recap: Bagging

Random Forests

Application in R

Ensemble methods

- We can overcome the weaknesses of a single tree by combining many individual trees.
- The following methods use trees as building blocks to construct more powerful prediction models.

1. Bagging
2. Random forests
3. Boosting

Bagging

- Individual decision trees are non-robust, i.e. have high variability.
- Averaging across observations reduces variance.
- Bagging(Bootstrap aggregation) uses this idea by taking repeated samples from the (single) training data set and averaging the results.
- This reduces the variance of the individual trees.

Bagging

1. Create B different bootstrapped training data sets.
2. Train a decision tree $\hat{f}^b(x)$ on each of the samples.
3. Average all the predictions to obtain:

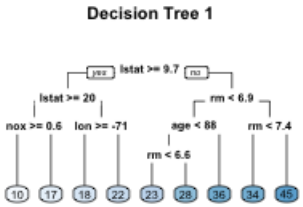
$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

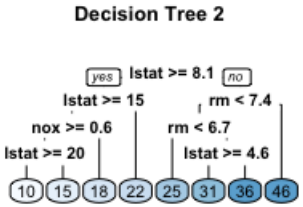
4. The final model is given by $\hat{f}_{bag}(x)$

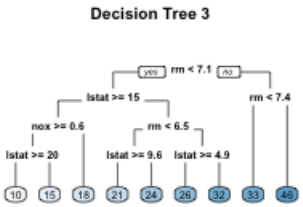
Random forests

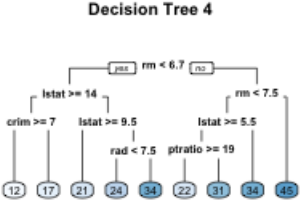
- If there's a single strong predictor in our model, then all the bagged trees will have the same predictor as an important input and all the trees will be highly correlated.
- This will prevent bagging from reducing the variance among the trees.

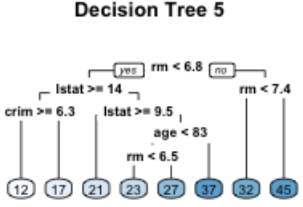
Random forests

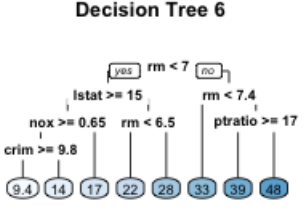












Source: HOML

Random forests

- 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.

Random forests

- As in bagging, we build a number of decision trees on bootstrapped training samples.
- In order to **decorrelate** its trees, a random forest only considers a random subset of predictors when making each split (for each tree).

Random forests

- Random forests help to reduce tree correlation by injecting more randomness into the tree-growing process
- While growing a decision tree during the bagging process, random forests perform split-variable randomization where each time a split is to be performed, the search for the split variable is limited to a random subset of m_{try} of the original p features.
- Typical default values are $m_{try} = \frac{p}{3}$ for regression and $m_{try} = \sqrt{p}$ for classification, but this should be considered a tuning parameter.

Random forests

- By restricting the variables our tree sees at a given split we:
  - nudge trees away from always using the same variables,
  - increasing the variation across trees in our forest,
  - which potentially reduces the variance of our estimates.

Random forests

- The following hyperparameters should be tuned for optimal ranfom forest performance
  1. $mtry$: The number of variables to randomly sample as candidates at each split. When $mtry = p$ the model equates to bagging. When $mtry = 1$ the split variable is completely random, so all variables get a chance but can lead to overly biased results.

  2. $ntree$: Number of trees. We want enough trees to stabalize the error but using too many trees is unncessarily inefficient, especially when using large data sets.

Random forests

3. $sampsize$: The number of samples to train on. The default sampling scheme for random forests is bootstrapping where 100% of the observations are sampled with replacement. The sample size parameter determines how many observations are drawn for the training of each tree. Assess 3–4 values of sample sizes ranging from 50–100%.

4. $nodesize$: minimum number of samples within the terminal nodes. Controls the complexity of the trees. Smaller node size allows for deeper, more complex trees and smaller node results in shallower trees.

5. $maxnodes$: maximum number of terminal nodes. Another way to control the complexity of the trees. More nodes equates to deeper, more complex trees and less nodes result in shallower trees

References

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2017)

An Introduction to Statistical Learning

Springer.

https://www.statlearning.com/

Ed Rubin (2020)

Economics 524 (424): Prediction and Machine-Learning in Econometrics

Univ, of Oregon.