ACTL3142 Week 9 - Tree-based Methods

Tadhg Xu-Glassop

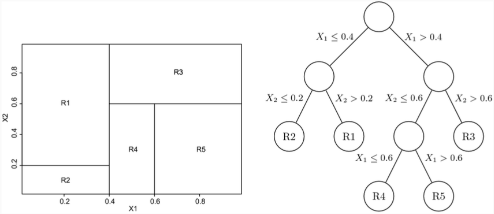
2025T2

Decision Trees

A decision tree partitions the predictor space into *terminal nodes* or *leaves* to make predictions.

Given a new observation lies in a particular node:

- In a regression, the average of all training observations in that node is the prediction;
- In a classification, the majority class of all training observations in that node is the prediction.



Training a Decision Tree

Recursive Binary Splitting

1. Considering all p predictors, find the additional split that leads to the largest reduction in some score;
   - In regression, this could be MSE;
   - In classification, this could be Gini index or entropy.
2. Repeat the above, finding more splits in the data until some stopping criterion is reached (nodes have ≤ 5 observations, maximum depth reached, etc.).

Controlling Flexibility

Intuitively, more nodes $\implies$ more flexibility...

Cost-Complexity Pruning

The cost-complexity criterion has the form,

$$\text{Total cost} = \text{Measure of fit} + \text{Measure of complexity}.$$

For example,

$$C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_m)^2 + \alpha |T|,$$

where $|T|$ is the number of terminal nodes.

With the above, we can train a 'big' tree, 'prune' it by finding the a solution to $C_\alpha(T)$ by working backwards, and then cross-validate all the solutions to pick the best one.

Ensemble Methods

Trees are typically rubbish by themselves! But, there is strength in numbers...

Ensemble Methods

An **ensemble method** is a prediction method that combines several models together to make predictions. Say we train d models;

- In a regression problem, we would make predictions with all d models and then take the average;
- In a classification problem, we would make predictions with all d models and then take the majority class.

Bagging

A shortfall of trees is they tend to fit to noise very easily and mistake random patterns for trends we want to capture.

Bagging

1. Re-sample the original dataset repeatedly with replacement (bootstrap), obtaining B different training data sets.
2. Train a deep decision tree on all B training sets (no need to prune - why?).
3. Make a prediction using all B models as usual with an ensemble method.

Bagging

A shortfall of trees is they tend to fit to noise very easily and mistake random patterns for trends we want to capture.

Bagging

1. Re-sample the original dataset repeatedly with replacement (bootstrap), obtaining B different training data sets.
2. Train a deep decision tree on all B training sets (no need to prune - why?).
3. Make a prediction using all B models as usual with an ensemble method.

Does B control the bias-variance tradeoff?

Bagging

A shortfall of trees is they tend to fit to noise very easily and mistake random patterns for trends we want to capture.

Bagging

1. Re-sample the original dataset repeatedly with replacement (bootstrap), obtaining B different training data sets.
2. Train a deep decision tree on all B training sets (no need to prune - why?).
3. Make a prediction using all B models as usual with an ensemble method.

Does B control the bias-variance tradeoff?
No !! It's simply a variance reduction method.

Random Forests

An issue with bagging is that across all B samples, the models tend to be very similar, where the top splits are the same strong predictors. This leads to the predictions being highly correlated and leading to a lesser variance reduction, An issue with bagging is that across all B samples, the models tend to be very similar, where the top splits are the same strong predictors. This leads to the predictions being highly correlated and leading to a lesser variance reduction,

Random Forest

A random forest is very similar to boosting, but when training each tree at each split only a random subset of size $m < p$ of the predictors is considered.

This adds variability to the trees, which decorrelates them and leads to a better variance reduction !

Boosted Trees

Boosted models **sequentially** train on an updated training set with changing residuals based on previous models.

Boosted Trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i.
2. For $b = 1, 2, \ldots, B$:
   a. Fit a tree $\hat{f}^b$ with d splits $(d+1)$ terminal nodes to the training data (X, r).
   b. Update $\hat{f}$ by adding a shrunken version of the new tree,
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda f^b(x).$$
   c. Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$
3. The final model is the last iteration of $\hat{f}(x)$.

Notes on Boosted Trees

The idea of training on updated residuals and in a sequential manner is to learn off the mistakes of the past models - if there are big residuals in some predictions from past models, future models will aim to rectify this.

Notes on Boosted Trees

The idea of training on updated residuals and in a sequential manner is to learn off the mistakes of the past models - if there are big residuals in some predictions from past models, future models will aim to rectify this.

Number of splits d is very small - usually 1 or 2, making the models 'slow learners.' Hence, it makes sense to call λ the 'learning rate' (higher lambda puts more emphasis on updated residuals and so the models will change quicker).

Notes on Boosted Trees

The idea of training on updated residuals and in a sequential manner is to learn off the mistakes of the past models - if there are big residuals in some predictions from past models, future models will aim to rectify this.

Number of splits d is very small - usually 1 or 2, making the models 'slow learners.' Hence, it makes sense to call λ the 'learning rate' (higher lambda puts more emphasis on updated residuals and so the models will change quicker).

The number of trees B controls the flexibility of the model and thus the bias-variance tradeoff (note the difference between this and random forests!)

Notes on Boosted Trees

The idea of training on updated residuals and in a sequential manner is to learn off the mistakes of the past models - if there are big residuals in some predictions from past models, future models will aim to rectify this.

Number of splits d is very small - usually 1 or 2, making the models 'slow learners.' Hence, it makes sense to call λ the 'learning rate' (higher lambda puts more emphasis on updated residuals and so the models will change quicker).

The number of trees B controls the flexibility of the model and thus the bias-variance tradeoff (note the difference between this and random forests!)

There are lots of hyperparameters here (and even more in other versions of boosted trees!) These are tuned in so many ways, but most commonly a grid-search.