AARMS Statistical Learning

Today
1. Intro Tree-Based Methods
2. Decision Tree
   - Regression Tree
   - Classification Tree

1. Introduction to Tree-based Methods

- for Regression and classification
- These involve stratifying or segmenting the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods

* Cons & pros

- Tree-based methods are simple and useful for interpretation
- However, typically they are not compe~~titate~~ with the best supervised learning approaches in terms of prediction accuracy.
- Hence we discuss "bagging", "random forest" and "boosting". These methods grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expenses of some loss interpretation.

2. Decision Tree

- Regression — Response variable is continuous
- Classification — Response variable is categorical

* Terminology for Trees

- Root node, terminal nodes (leaves)
- Decision Trees are typically drawn upside down, in the sense that the leaves are the bottom of the tree.
- The points along the tree where the predictor space is split are refered to as internal nodes.

2.1. Regression Trees

* Details of Tree-building process

- We divide the predictor space — i.e. the set of possible values for $X_1, X_2, \cdots X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$
- For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observation R_j.
- In theory, the regions could be any shape. However, we choose to divide the predictor space into high-dimensional rectangle, or boxes, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes, $R_1, \cdots R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a top-down, greedy approach that is known recursive binary splitting
- The approach is top-down because it begins at the top of

the tree, and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.

- It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

* Process
  - First select the predictor X_j, and the cutpoint s such that splitting the predictor space into the regions $\{X \mid X_j < s\}$ and $\{X \mid X_j \geq s\}$ leads to the greatest possible reduction in RSS.
  - Next, repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.
  - However, this time, instead of splitting the entire regions, we now have three regions.
  - Again, look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.
  - We predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.

* Tree Size.

(The process described above may produce good predictions on the ~~tree~~ training set, but is likely to overfit the data, leading to poor test set performance. Why?)

⇒ A smaller tree with fewer splits (i.e. fewer regions $R_1, \cdots R_J$) might lead to lower variance and better interpretation at the cost of a little bias.

⇒ Solutions :

① grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold due to each split exceeds some (high) threshold.

⇒ "short-sighted : a seemingly worthless split early on in the tree might be followed by a very good split — i.e. a split that leads to a large reduction in RSS later on.

② A better strategy is to grow a very large tree T_0 and then prune it back in order to obtain a subtree.

⇒ cost-complexity pruning — aka weakest link pruning

* Cost-complexity pruning

— Consider a sequence of trees indexed by a non-negative tuning parameter α. For each value of α, there ~~corresponding~~ correspond a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. $|T|$ indicates the number of terminal nodes of the tree T, R_m is the rectangle (i.e. the subset of predictor space) corresponding to the mth terminal nodes. $\hat{y}_m$ is the mean of the training observations in R_m

— The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the ~~training~~ training data.

- We select an optimal value $\hat{\alpha}$ using cross-validation
- Then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

* Tree algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function ~~of~~ of α.
3. Use k-fold cross-validation to choose α. For each $k=1, \cdots K$.
   3.1. Repeat Steps 1. and 2 on the $\frac{K-1}{K}$th fraction of the training data, excluding the kth fold.
   3.2. Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.
   Average the results, and pick α to minimize the average error.
4. Return the subtree from step 2 that corresponds to the chosen value of α.

2.2 Classification Tree

- Similar to a regression tree, except need to predict a qualitative responce rather than a quantitative one.
- Predict that each observation belongs to the most commonly occurring class of training observations in the regions to which it belongs.

* Splitting criterion
- Cannot use RSS.
- Classification error rate : the fraction of the training observations in that region that do not belong to the most common class.

$$E = 1 - \max_k(\hat{p}_{mk})$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class.

⇒ However classification error is not sufficiently sensitive for tree-growing, and in practice two other ~~measurements~~ measures are preferable.

- Gini index

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}) \qquad \left(\text{Total } G = \sum_{m=1}^{M} \frac{n_m}{n} \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})\right)$$

a measure of total variance across the K classes. The Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one.

⇒ referred as a measure of node purity, i.e, a small value indicates that a node contains predominantly observations from a single class.

- Cross-entropy

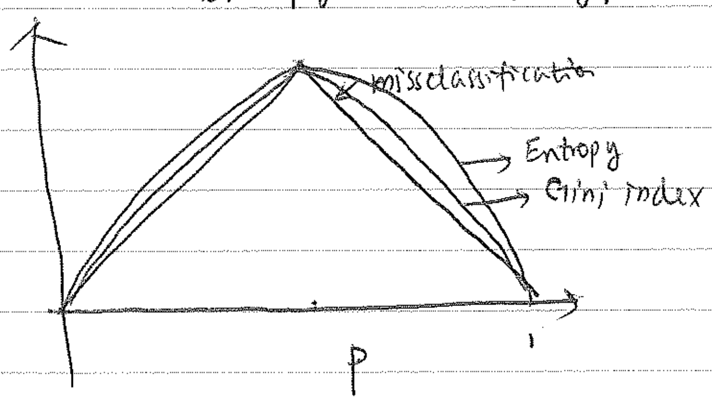
$$\left(\text{Total } D = -\sum_{m=1}^{M} \frac{n_m}{n} \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}\right)$$

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

⇒ Gini & Cross-entropy are very similar numerically.

Ex1. = For a two-class classification problem

Misclassification rate	$1 - \max(p, 1-p)$
Gini index	$2p(1-p)$
Cross-Entropy	$-p\log p - (1-p)\log(1-p)$



Ex2:

(400, 400) → (300, 100), (100, 300)

(400, 400) → (200, 400), (200, 0)

Misclassification rate

Gini index

Entropy