Nonparametric Classification Methods

- We now examine some modern, computationally intensive methods for regression and classification.
- Recall that the LDA approach constructs a line (or plane or hyperplane) that separates our numerical data into groups, and uses this linear boundary as a classifier.
- A *nonparametric* approach, K-nearest neighbors, allows the boundary between groups to be very flexible and data-determined.

K-Nearest Neighbors (KNN) Classification

- For some positive integer K, and some data vector $\mathbf{x}_0$ corresponding the a “test” observation, the KNN classifier picks the K observations in the training data that are closest to $\mathbf{x}_0$.
- This set of K observations is denoted by $\mathcal{N}_0$.
- Then for test observation $\mathbf{x}_0$, the probability that it belongs to group j is estimated as
$$\frac{1}{K}\sum_{i\in\mathcal{N}_0} I(y_i = j),$$
i.e., the proportion of training observations within the $\mathcal{N}_0$ “window” that belong to group j.

Role of K in the KNN Classifier

- The choice of K controls the size of the window.
- When K is a small whole number, the KNN classifier can predict the known observations (training data) very well, but may predict poorly for new data (test data).
- As K gets larger, the KNN classifier may get better at classifying new observations, up to a point.
- But if K gets overly large, KNN loses its flexibility and the classification boundary becomes nearly linear.
- Some medium value of K often produces a moderately flexible classifier with the best prediction results on test data.

K-Nearest Neighbors Regression

- Note that KNN can be used in the regression setting as well, to predict a numerical response variable based on one or more numerical explanatory variables.
- At any given value x_0 (or set of values $\mathbf{x}_0$) of the explanatory variable(s), we find the K nearest observations in the observed (training) data set (call this set of points $\mathcal{N}_0$).
- “Closeness” is measured by the distance of an observation’s vector of explanatory variables to $\mathbf{x}_0$.
- The predicted response (Y value) at $\mathbf{x}_0$ is the average Y value of these observations in $\mathcal{N}_0$.

Trees and Random Forests

- *Regression trees* are used when we have one response variable which we want to predict/explain using possibly several explanatory variables.
- The goals of the regression tree approach are the same as the goals of multiple regression:
  1. Determine which explanatory variables have a significant effect on the response.
  2. Predict a value of the response variable corresponding to specified values of the explanatory variables.

More about Regression Trees

- The regression tree is a method that is more algorithm-based than model-based.
- We form a regression tree by considering possible divisions of the data into partitions based on the value of one of the predictors:
- Example: For data **D**, let
  D_1 = data for which $X_1 < 12$
  D_2 = data for which $X_1 \geq 12$

$$\begin{tikzcd} & \boxed{\mathbf{D}} \arrow[dl, dash] \arrow[dr, dash] & \\ \boxed{D_1} & & \boxed{D_2} \end{tikzcd}$$

More about Regression Trees

- Calculate the mean of the responses in each partition set.
- Compute the residual sum of squares for this partitioning:

$$RSS_{\text{partitioning}} = RSS_{part1} + RSS_{part2}$$

$$= \sum_{i \in part1} (y_i - \bar{y}_{part1})^2 + \sum_{i \in part2} (y_i - \bar{y}_{part2})^2$$

- Of *all possible* ways to split the data, pick the partitioning that produces the smallest RSS.

Creating the Regression Tree

- Continue the algorithm by making subpartitions from the most recent splitting.
- The result is a treelike structure subdividing the data.
- This also works well when a predictor is categorical – we can subdivide the data based on the categories of the predictor.
- Splitting on one variable separately within partitions of another variable is essentially finding an interaction between the two variables.

Determining the Final Tree

- The usual regression diagnostics can be used – if problems appear, we can try transforming the response (*not* the predictors).
- Eventually we will want to stop splitting and obtain our final tree.
- A criterion to select to “best” tree is the cost-complexity:

$$CC(Tree) = \sum_{i \in \{\text{terminal.nodes}\}} RSS_i + \lambda \times (\text{number.of.terminal.nodes})$$

- The first piece measures fit and the second piece penalizes an overly complex tree.

Determining the Final Tree (continued)

- Another approach to tree selection is *cross-validation*.
- We select a random subset of the data, build a tree with that subset, and use the tree to predict the responses of the remaining data.
- Then a cross-validation prediction error can be calculated: A tree with low CV error is preferred.

Using Software to Find the Tree

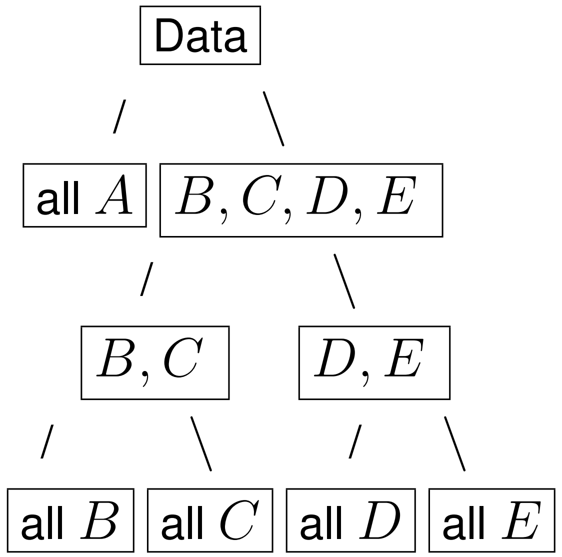
- The `rpart` function in the `rpart` package of `R` produces regression tree analyses.
- See example with Boston housing data:
  A plot of the graph of the tree reveals the important variables.
- More (or less) complex trees may be obtained by adjusting the `cp` argument in the `prune.rpart` function.
- The `cp` value is directly proportional to λ, so a larger value of `cp` encourages a simpler tree.
- The `plotcp` function can guide tree selection by plotting CV error against `cp`: We look for the elbow in the plot.

Classification Trees

- *Classification Trees* work similarly when the response is *categorical* (or discrete).
- Instead of using RSS as a criterion to guide the splits, a “node purity” criterion is used.
- With classification trees, the goal is that the terminal nodes should classify observations into the correct groups.
- Ideally, we set up the tree so the class types within a particular split are all of one kind:

Classification Trees

- Example: Data having 5 known groups, A, B, C, D, E:



- This ideal situation does not usually occur in reality.

Node Purity

- The *node purity* measures how close a node is to having observations of only one category.
- There are several possible measures of node purity. The `rpart` function by default uses the Gini Index:
- If p_{ik} = the proportion of sample observations assigned to node i having category k, then the Gini Index is $\sum_i D_i$, where for node i,

$$D_i = 1 - \sum_k p_{ik}^2$$

- In the "ideal" case when all nodes are perfectly pure, the Gini Index is minimized (at zero).
- See example with `hsb` data.

Random Forests

- The random forest approach is an `ensemble` method – it generates many individual classifiers/predictions and aggregates then to produce a better overall method.
- As the name suggests, a random forest consists of *many trees*.
- It relies on the principle of *bagging* (bootstrap aggregating) proposed by Leo Breiman.
- Different trees are constructed using n_{tree} bootstrap resamples of the data, and the nodes are split based on random subsets of predictors, each of size m_{try}.
- Classification of new observations is done by majority vote among the constructed trees.

Random Forests for Regression

- In regression, prediction is done by averaging predicted response values across the predicted trees.
- The error rate is typically assessed by predicting out-of-bag (OOB) data – the data not chosen for the bootstrap sample – using each constructed tree.
- The `randomForest` function in the `randomForest` package will obtain a random forest, for either regression (continuous response) or classification (categorical response).
- It also provides a measure of which explanatory variables are most important.
- See examples on the course web page.