36-724 Spring 2006: Tree-based Methods

Brian Junker

March 24, 2006

- Introduction
- Regression Trees: Basic Ideas
- Classification Trees
- Some Pro’s and Con’s

Introduction

Last time we looked briefly at

- Generalized additive models: gam in R or Splus; package vgam (install from http://www.stat.auckland.ac.nz/PEOPLE/yee) extends gam to multinomial response models (e.g. classification with more than two groups).

- Projection Pursuit regression: ppr in Splus; ppr in library(MASS) (R or Splus); package classPP in R; and neural network models: nnet in R or Splus.

Today we will look at

- Tree-based methods; and Classification and Regression Trees in particular (rpart in R or Splus; package tree in R).
Choosing a completely optimal partition is usually computationally infeasible. Recursive partitioning (rpart in R/Splus; CART in the literature) proceeds in a top-down, one-variable-at-a-time, greedy fashion.

Suppose we have a partition $R_1, R_2, \ldots, R_M$ and we want to decide whether and where to split the $m$th element of the partition $R_m$: 

$$\hat{y}_i = \sum_{m=1}^{M} c_m 1_{\{x_i \in R_m\}}$$

Under squared-error loss,

$$c_m = \frac{\sum_{i=1}^{M} y_i 1_{\{x_i \in R_m\}}}{\sum_{i=1}^{M} 1_{\{x_i \in R_m\}}} = \frac{1}{N_m} \sum_{x_i \in R_m} y_i = \bar{y}_m$$

(and other losses give corresponding point estimates).

---

**Choosing the partition**

Choosing a completely optimal partition is usually computationally infeasible. Recursive partitioning (rpart in R/Splus; CART in the literature) proceeds in a top-down, one-variable-at-a-time, greedy fashion.

Suppose we have a partition $R_1, \ldots, R_{M-1}$ and we want to decide whether and where to split the $m^{th}$ element of the partition $R_m$ into two pieces:

$$R_m^{(1)}(j, s) = \{x : x_j \leq s\} \quad \text{and} \quad R_m^{(2)}(j, s) = \{x : x_j > s\}$$

(where $x_j$ is the $j^{th}$ coordinate of $x = (x_1, \ldots, x_d)$).

Then under squared error loss we really want to find $j, s, c_m^{(1)}$ and $c_m^{(2)}$ solving

$$\min_{j,s} \left[ \min_{c_m^{(1)}} \sum_{x_i \in R_m^{(1)}(j,s)} (y_i - c_m^{(1)})^2 + \min_{c_m^{(2)}} \sum_{x_i \in R_m^{(2)}(j,s)} (y_i - c_m^{(2)})^2 \right]$$

For any $j, s$ the inner minimia are achieved by setting

$$c_m^{(k)} = \text{ave}\{x : x \in R_m^{(k)}\}$$

and the outer minimum is now a matter of searching over $j$ and optimizing over $s$ (which is also essentially a discrete optimization).
Classification Trees

When \( y_i \) indicates class membership (\( y_i = k \iff i \in \text{Class } k \)), a classification tree can be built in the same way, with two minor modifications:

- In partition element \( R_m \) we estimate all \( K \) class membership probabilities:
  \[
  \hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} 1_{\{y_i = k\}}
  \]
  and for \((x, y)\) with \( x \in R_m \) we predict \( \hat{y} = \arg\max_k \hat{p}_{mk} \equiv k(m) \). Note that all observations in \( R_m \) get assigned to the same class, \( k(m) \).
- Usually squared error loss is not used for \( R(T) \). Instead, some common choices are:
  - Misclassification rate: \( R(T) = \frac{1}{N_m} \sum_{x_i \in R_m} 1_{\{y_i \neq k(m)\}} = 1 - \hat{p}_{mk(m)} \)
  - Gini index: \( R(T) = \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk} (1 - \hat{p}_{mk}) \)
  - Information index: \( R(T) = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \)

Growing a tree

The usual approach to growing a tree proceeds in two phases:

- Phase 1: Grow a “full” tree, starting from a single partition element \( R_1 \) consisting of the entire set of rows of \( X \), and continuing to recursively divide elements until there are a minimum number of observations (say, 5) in each partition element. This maximal tree is called \( T_0 \).
- Phase 2: For each \( \alpha \), collapse \( T_0 \) to a subtree \( T = T_\alpha \) that minimizes
  \[
  PL(T) = R(T) + \alpha |T|
  \]
  where \( R(T) \) is some appropriate measure of loss (e.g. squared error) and \( |T| \) is size, in number of leaves, of \( T \). \( T_\alpha \) can be found greedily, that is by successively collapsing the internal node that least increases \( R(T) \) [see Breiman et al., (1984) or Ripley (1996)].

In practice, \( R(T) \) is usually computed by 5- or 10-fold cross validation. The final \( \alpha \), and hence final \( T_\alpha \), can be chosen from a plot of \( \alpha \) vs \( PL(T) \) or of \( \alpha \) vs. \( R(T) \), etc. This is called cost-complexity pruning.
**Some Pro’s and Con’s**

− Their (cross-validation or test-data-set) error rates are usually not as good as other methods.
− Trees have a hard time capturing smooth structure, additive regression structure, interactions, and more generally structure that is not “parallel” to the coordinate axes of the features \(X\) (*this is a bias issue*).
+ Trees are easy to interpret (splits on individual variables) and easy to implement as simple classifiers.
+ Since classification trees essentially give estimates \(\hat{p}_{mk(m)}\) of the likelihood, it is easy to combine them with prior class probabilities to get posterior probabilities of class membership.
+ Loss functions for misclassification can be incorporated into the splitting process, e.g. if we have a loss matrix \(L_{kk'}\) it can be incorporated into the Gini index as \(\sum_{k\neq k'} L_{kk'} \hat{p}_{mk} \hat{p}_{mk'}\).
− Trees are highly dependent on the configuration of the training data and have high variance over repeated training samples (*this is a variance issue*).

We will return to tree based methods later to see how model averaging procedures can improve the bias and variance of tree classifiers (at the expense of simple interpretability!).

**Comments on the different loss functions**

- All (Misclassification, Gini, Info) are similarly-shaped but Misclassification is not continous; the others work better in optimization
- Gini and Information tend to be better at isolating areas of the space where there are no observations, and no more work has to be done.
- Usually *cost-complexity pruning* is guided by misclassification rate, even if some other loss function was used to grow the tree.

For two classes, with \(p = \) prob. of first class,

\[
R_{\text{Misclass}} = 1 - \max(p, 1 - p) \\
R_{\text{Gini}} = 2p(1 - p) \\
R_{\text{Info}} = -[p \log(p) + (1 - p) \log(1 - p)]
\]