STAT 770 Dec. 7 Lecture 27

Decision Tree Methods vs Logistic Regression

Reading and Topics for this lecture: **rpart** and **randomForest** software descriptions (posted to special Decision Tree module in ELMS), plus the R Scripts for this class: and `IntXPred.RLog` and `RandomForests.RLog`.

(1) General discussion of Logistic Regression as Classification

(2) Motivation for Decision Trees as search for Interactions

(3) High-level discussion of `CART` and `rpart`

(4) Script case-studies, of `rpart` and `randomForest`

Logistic Regression as a Classification Method

• With binary responses Y_i and predictors $\underline{X}_i$: logistic regression provides predictors $I_{[\underline{X}_i^{tr} \widehat{\beta} \geq c]}$ for $Y_i = 1$.

• Effective classification rules may be complicated, depending on (higher-order) interactions or nonlinear recodes of the coordinates of $\underline{X}_i$.

• Stepwise model-selection strategies offer screening approach for model terms: but how could one find important higher-order interactions? Search among many predictors may fail for combinatorial reasons.

• Decision trees look directly for successive branchings, may arrive at combinations of variables without searching among all such combinations.

CART and Recursive Partitioning

Sources:

Classification and Regression Trees, L. Breiman et al. (1980)

H. Zhang & B. Singer (2010) *Recursive Partitioning and Applications*, Springer.

Similar R packages `rpart` and `tree`, "long introduction" to `rpart` by Therneau and Atkinson.

All these tree-based methods consist of two parts: successive (greedy) search for 'splitting' of nodes to decrease an index as much as possible. Tree is "grown" until a stopping criterion on # levels or size of nodes is reached, then "pruned".

Recursive (Binary) Partitioning, cont'd

stage K of tree: set U of units partitioned into nodes $\{A_j\}_{j=1}^K$

Split node A_j into $A_{j,1}, A_{j,2}$, where $A_{j,1} = \{i \in A_j : X_{i,k_j} \le a_{k_j}\}$ or $\{i \in A_j : X_{i,k_j} \in C_{kj}\}$ (for *factor-column* X_{i,k_j})

Splitting index – choose node, split to maximize change

$$\boxed{\Delta I = p(A_j) I(r(A_j)) - p(A_{j,1}) I(r(A_{j,1})) - p(A_{j,2}) I(r(A_{j,2}))}$$

where $r(B) = |B \cap [Y = 1]|/|B|$, and $p(B) = |B|/|U|$

$I(p) =$ concave fcn, e.g. $p(1-p)$ or $-p \log p - (1-p) \log(1-p)$

Pruning – minimize misclassification rate penalized by $\alpha \cdot$#nodes

Random Forest Idea

- Grow *many* trees, on randomly sampled subsets of data, with splits at each stage based on a small random sample of $\underline{X}$ coordinates

- aggregate over many trees by averaging predictions from mini-tree prediction rules.

- look in Scripts for examples