Classification
And
Regression
Trees

CART provides a non-parametric approach to classification.

References:
Breiman, L., Friedman, J., Olshen and Stone, Classification and

Salford Systems documentation:
CART: Classification and regression
trees (Steinberg and Colla).
http://Salford-Systems.com

Tree functions in S-plus and R

Chapman & Hall, New York.
(Chapter 9, Tree-based models
Clark, L.A. and Pregibon, A.)

Springer-Verlag, New York (Chapter 9)

rpart functions
tree functions
Logistic regression:
(cused 31 successes
54 failures)

Classify as a "success" if

\[-2.231 + 0.9945 E_3 + 0.7835 A_{treat}
+ 0.9134 A_{treat} - 1.149 \text{ work}
> \log \left( \frac{1}{0.06} \right)\]
Basic idea:

Goal: Classify each individual into one of $g$ pre-defined groups using information contained in the variables $X_1, X_2, \ldots, X_p$.

Method: Construct a binary tree.

Binary tree:

- A decision is made at each non-terminal node
  - one variable
  - one boundary

- Cases that reach the node are divided into two groups

- Reduce impurity
Features of CART:

1. Classification rule is simple
   - Series of binary decisions
   - Is a value for some variable smaller or larger than some boundary?

2. Automatically does variable selection as it builds the tree.

3. Uses both interval and categorical variables
   - Ordinal categorical variables
   - Nominal categorical variables

4. Invariant under monotone transformations of interval variables and any 1-1 transformation of nominal variables

5. Automatically deals with missing values
   - CART uses surrogate variables
   - tree() function in S-plus does not handle missing values well.

6. Can be unstable for small samples
   - Deleting a few cases can alter the variable selected at some decision nodes.
   - Alter estimates of misclassification probabilities

7. Produce an ordering of the explanatory variables from "most" important to "least" important.
(a) Crudely identifies non-linear and non-smooth boundaries between groups

(b) Crudely identifies non-linear and non-smooth boundaries between groups
(4) When a linear discriminant rule is appropriate (or a smooth boundary exists between two groups), it will take a tree with many nodes to approximate it.
10. CART is insensitive to "outliers".

11. CART might reveal interactions or non-linear dependencies among variables
   - One variable may only appear to produce an important split for certain ranges of values of some other variables
   - Identify thresholds
   - Use this information to refine logistic regression models

12. CART can use the same variable to make splits in different parts of the tree.

13. CART will show you the "best" alternative splits.

14. CART uses cross-validation to
   - Prune the original tree
   - Estimate misclassification probabilities

     CART is just an algorithm
Growing a classification tree:

**Basic strategy:**

1. Construct a large tree with more nodes and branches than are really needed.

2. Prune off some nodes and branches to obtain a "right sized" tree.
   - minimize an estimate of the expected misclassification cost
   - complexity parameter
   - cross validation

**Constructing the initial "large" tree**

- For each decision node
  - select the variable that gives the greatest reduction in node impurity
  - select the boundary

- Stop when
  - all nodes are sufficiently "pure"
  - or, terminal nodes have fewer than 5 cases.
Measure of node purity:

A node is "pure" if there are only cases from one group at that node.

For node "r"

\[ \pi_{1r} = \text{proportion of cases from group 1} \]
\[ \pi_{2r} = \text{proportion of cases from group 2} \]
\[ \vdots \]
\[ \pi_{pr} = \text{proportion of cases from group p} \]

Node "r" is pure if \( \pi_{ir} = 1 \) for the i-th group

Measure of node impurity:

- Achieve a maximum at
  \[ (\pi_{1r}, \ldots, \pi_{pr}) = (\frac{1}{p}, \frac{1}{p}, \ldots, \frac{1}{p}) \]
- Achieve a minimum (usually zero) for a "pure" node
  \[ \pi_{ir} = 1 \text{ for some } i \]
- Symmetric function of \( (\pi_{1r}, \ldots, \pi_{pr}) \)

**CART**: Gini index (default for rpart)

\[ I(r) = \sum_{i=1}^{p} \pi_{ir} \pi_{jr} = 1 - (\pi_{1r}^2 + \pi_{2r}^2 + \cdots + \pi_{pr}^2) \]

**S-plus**: Deviance

\[ I(r) = -2 \left[ \frac{p}{\sum_{i=1}^{p} \pi_{ir} \log(\pi_{ir})} \right] \]
Choosing the split:

- Consider each available variable

\[ X_i: \min_{b_i} \max_{a_i} \ldots \]

use \( X_i \leq a_i \) or \( X_i > a_i \) or \ldots \?

\[ X_i: \min_{b_i} \max_{a_i} \ldots \]

use \( X_i \leq b_i \) or \( X_i > b_i \) or \ldots \?

- For each possible split compute

\[ I(r) = 1 - \frac{\hat{\pi}_{i,r}^2}{\pi_{i,r}^2} \]

\[ I(r+1) = 1 - \frac{\hat{\pi}_{i,r+1}^2}{\pi_{i,r+1}^2} \]

\[ I(r+2) = 1 - \frac{\hat{\pi}_{i,r+2}^2}{\pi_{i,r+2}^2} \]

- Use the variable and boundary that gives the greatest reduction in impurity; i.e., maximizes

\[ I(r) - \left[ I(r+1) + I(r+2) \right] \]

If misclassification costs are unequal, \( I(r) \) becomes

\[ I(r) = \sum_{i,j} c(i,j) m_i \pi_{i,r} \]

\[ = \sum_{i,j} \left[ c(i,j) + c(j,i) \right] m_i \pi_{i,r} \]

Grow a large tree and stop when

- each node is "pure"
- or, "non-pure" terminal nodes have fewer than 5 cases
Pruning:

- A smaller tree may better classify cases not in the training samples
- Prune the "weakest" branch first

Cross validation:

- Split the N units in the training samples into V subsets of "equal" size (V=10)
- Set aside the data for one of the subsets
  - build a new tree using the data in the remaining V-1 subsets
  - run the cases in the subset that was set aside down the tree.

Complexity parameter:

\[ M(r) + \alpha N(r) \]

- an "optimal" value for \( \alpha \) is derived from the cross validation results
- this "optimal" \( \alpha \) is used to prune the original "large" tree
Summary:

- CART is just an algorithm
- CART provides a "non-linear" look at the data
- Usually does not classify as well as logistic regression or neural nets
- Cart may give you ideas for improving a logistic regression model:
  - interactions
  - stratification
- Easy to understand decision rules
  \[ \log \left( \frac{P}{1 - P} \right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \]
# This R code is posted as tree crim.R
# It creates a classification tree for the alcoholic treatment data from the lecture. Variables with missing values are not included in the analysis.

# Attach the rpart library of functions
library(rpart)

# First enter the data into a data frame. The data are posted on the file crimeR.dat where missing data are indicated by NA.
crm <- read.table("c:/stat501/crimeR.dat")

# Assign labels to variables (columns of the data frame)
names(crm) <- crm.var

# Create a factor object (categorical variable) for the response variable

## Create factors for some predictor variables

# Compute a classification tree and put the results into an object called crim.rp
set.seed(123)
crm.rp <- rpart(crm ~ AGE + SEX + EDUC + EMOTION + ETREAT + FLIVE + ATREAT + ALCADD + HEALTH + FINANCE + PDRINK + SIBS + WORK + WAGES + JOBS + DAGE + DFREQ + STOP + DRY + DRUGS, crm, cp = 0.001)
# Write out a summary description of the tree

summary(crm.rp)

Call: rpart(formula = y ~ X9 + AGE + SEX + EDUC + EMOTION + ETREAT + FLIVE + ATREAT + ALCADD + HEALTH + FINANCE + FMARR + PDRAIN + SIBS + WORK + WAGES + JOBS + DAGE + DFREQ + STOP + DRY + DRUGS,     
data = crm, cp = 0.001)

n= 88

CP nsplit rel error xerror xstd
1 0.16129032 0 1.0000000 1.0000000 0.1445491
2 0.09677419 1 0.9031971 1.0645161 0.1464992
3 0.03262866 2 0.7419345 1.2535826 0.1498954
4 0.00100000 4 0.6774194 1.193548 0.1493767

Node number 1: 88 observations,                 
complexity param=0.1612903                 
predicted class=Dropout                      
expected loss=0.3522727                      
class counts:  57  31                         
probabilities: 0.648 0.352                   
left son=2 (69 obs)  right son=3 (19 obs)    
Primary splits:                               
  DRY  = 10.5 to the left, Improve=3.780754, (0 missing) 
  WAGES = 5.5  to the left, Improve=3.591022, (0 missing) 
  ATREAT = 1.5  to the left, Improve=2.286194, (3 missing) 
  HEALTH = 1.5  to the left, Improve=1.948344, (1 missing) 
  WORK  = 1.5  to the right, Improve=1.828290, (0 missing) 

Node number 2: 69 observations,                 
complexity param=0.09677419                   
predicted class=Dropout  expected loss=0.2753623 
class counts:  50 19                          
probabilities: 0.725 0.275                    
left son=4 (56 obs)  right son=5 (13 obs)    
Primary splits:                               
  SIBS  = 1.5  to the right, Improve=3.531755, (8 missing) 
  ETREAT = 1.5  to the left, Improve=2.982353, (1 missing) 
  WAGES = 5.5  to the left, Improve=2.939375, (0 missing) 
  ATREAT = 1.5  to the left, Improve=1.952568, (3 missing) 
  WORK  = 1.5  to the right, Improve=1.892596, (0 missing) 

Node number 3: 19 observations                 
predicted class=Success  expected loss=0.3684211  
class counts:  7 12                          
probabilities: 0.368 0.632
Node number 4: 56 observations, complexity param=0.03225806
predicted class=D ropout expected loss=0.1964286
class counts: 45 11
probabilities: 0.804 0.196
left son=8 (35 obs) right son=9 (21 obs)
Primary splits:
FMA R R splits as R R L L,
FLIVE splits as L L R L,
DR Y < 0.1 to the right,
ETREAT < 1.5 to the left,
WAGES < 9.5 to the left,
Surrogate splits:
FLIVE splits as L L R L,
AGE < 36 to the left,
WAGES < 14 to the left,
DAGE < 21.5 to the left,
DR Y < 0.1 to the right
Node number 5: 13 observations
predicted class=Success expected loss=0.3846154
class counts: 5 8
probabilities: 0.385 0.615
Node number 6: 35 observations
predicted class=D ropout expected loss=0.08571429
class counts: 32 3
probabilities: 0.914 0.086

Node number 9: 21 observations, complexity param=0.03225806
predicted class=D ropout expected loss=0.3809524
class counts: 13 8
probabilities: 0.619 0.381
left son=18 (9 obs) right son=19 (12 obs)
Primary splits:
JOBS < 1.5 to the left,
AGE < 44.5 to the right,
WAGES < 11 to the right,
PDRINK < 1.5 to the right,
EDUC < 2.5 to the left,
Surrogate splits:
WAGES < 11 to the right,
AGE < 36 to the right,
FINANCE < 1.5 to the right,
PDRINK < 1.5 to the left,
SIBS < 2.5 to the left

Node number 18: 9 observations
predicted class=D ropout expected loss=0.1111111
class counts: 8 1
probabilities: 0.889 0.111
Node number 19: 12 observations
predicted class=Success expected loss=0.4166667
class counts: 5 7
probabilities: 0.417 0.583
# Print out a description of what happened
# at each node of the tree

print(crm rp, digits=3)

n = 28
node), split, n, loss, yval, (yprob)
  * denotes terminal node
1) root 88 31 Dropout (0.6477 0.3523)
   2) DRY < 10.5 69 19 Dropout (0.7246 0.2754)
   4) SIQS = 1.5 56 11 Dropout (0.8036 0.1964)
   8) FMARR = 3.4 35 3 Dropout (0.9143 0.0857)*
  9) FMARR = 1.2 21 5 Dropout (0.6190 0.3810)
   18) JOBS < 1.5 9 1 Dropout (0.8889 0.1111)*
   19) JOBS = 1.5 12 5 Success (0.4167 0.5833)*
  5) SIQS < 1.5 13 5 Success (0.3846 0.6154)*
   3) DRY = 10.5 19 7 Success (0.3684 0.6316)*

# Display information on complexity parameter

plotcp(crm rp)
printcp(crm.rp)

Variables actually used in tree construction:
[1] DRY  FMARR JOBS SIBS

Root node error: 31/88 = 0.35227

n= 88

   CP nsplit rel error xerror xstd
1 0.161290    0  1.00000 1.0000 0.14455
2 0.096774    1  0.83871 1.0645 0.14650
3 0.032258    2  0.74194 1.2258 0.14889
4 0.001000    4  0.67742 1.1935 0.14938

# Display the tree

plot(crm.rp, uniform=T)
text(crm.rp, use.n=T)
title("Classification Tree")
Neural Networks:

- "Non-linear" black box approach
- Classification procedure "learns" from the data.
- Layers of logistic regression models
- nnnet library in R or S-PLUS


Support Vector Machines

- Find hyperplanes that best separate groups subject to some specified level of "misclassification"

- Maximize distances from hyperplanes to "nearest" correctly classified data points

- Allow non-linear functions of original variables

\[ f(x) = \beta_0 + \beta_1 h_1(x) + \beta_2 h_2(x) + \cdots \]

- Minimize

\[
\sum_{i=1}^{n} \left(1 - y_i f(x_i)\right)^2 + \frac{1}{2c} \|w\|^2
\]

\[ x_i \in \{\text{group 1, group 2}\} \]

\[ \|w\| \text{ is the distance from the hyperplane to the closest points of each group} \]
Code for support vector machines

R:
library(e1071)

S:
library(libsvm)

use the svm() function
use cross-validation to assess performance

Ref:

Single layer learning revisited: a stepwise
procedure for building and training a
neural network. In Neuro-Computing:
Algorithms, Architectures, and Applications,
ed. F. Fogelman Soulie and J. Haelmst,
Berlin, Springer-Verlag

Vembles & Ripley (2002) Sections 12.5 and 12.6