Classification and Regression trees:

- CART
- BOOSTING AND BAGGING
- RANDOM FOREST
- Data Mining Trees: ARF

Tree methods: Dependent variable is categorical

- Classification trees (e.g., CART, C5, Firm, Tree)
- Decision Trees
- Decision Rules

Tree methods: Dependent variable is numeric

- Regression Trees

Classification tree for the cancer groups using 10 principal components of the top 100 cancer genes. The classification rule produces zero mistakes in the training set and five mistakes in the testing set.
Tree methods: Dependent variable is categorical

- Classification trees (e.g., CART, C5, Firm, Tree)
- Decision Trees
- Decision Rules

Tree methods: Dependent variable is numeric
- Regression Trees

• For regression trees two criteria functions are:

  Equal variances (CART): \( h = \frac{N_L \hat{\sigma}_L^2 + N_R \hat{\sigma}_R^2}{N_L + N_R} \)

  Non equal variances: \( h = \frac{N_L \log \hat{\sigma}_L^2 + N_R \log \hat{\sigma}_R^2}{N_L + N_R} \)

• For classification trees: criteria functions

  \( h = P_L \min(p_L^0, p_L^1) + P_R \min(p_R^0, p_R^1) \)
  \( h = P_L \left(-p_L^0 \log p_L^0 - p_L^1 \log p_L^1\right) + P_R \left(-p_R^0 \log p_R^0 - p_R^1 \log p_R^1\right) \) (C5)

  \( h = P_L p_L^0 p_L^1 + P_R p_R^0 p_R^1 \) (CART)

DATA PREPROCESSING RECOMMENDATIONS FOR TREES

a. Make sure that all the factors are declared as factors.
Some times factor variables are read into R as numeric or as character variables. Suppose that a variable RACE on a SAS dataset is coded as 1, 2, 3, 4 representing 4 race groups. We need to be sure that it was not read as a numeric variable, therefore we will first check the types of the variables. We may use the functions "class" and "is.factor" combined with "sapply" in the following way.

  sapply(w, is.factor) or sapply(w, class)

Suppose that the variable “x” is numeric when it is supposed to be a factor. Then we convert it into factor:

  w$x = factor(w$x)

b. Recode factors:
Sometimes the codes assigned to factor levels are very long phrases and when those codes are inserted into the tree the resulting graph can be very messy. We prefer to use short words to represent the codes. To recode the factor levels you may use the function “f.recode”:

  > levels(w$Muscle)
  [1] "" "Mild Weakness"
  [3] "Moderate Weakness" "Normal"

  > musc = f.recode(w$Muscle, c("", "Mild", "Mod", "Norm"))

  > w$Muscle.new = musc
Example Hospital data

```r
hospital = read.table("project2/hospital.txt", sep = "", )
colnames(hospital) <-

hospital = hospital[, -c(1:4, 10)]
hospital$TH = factor(hospital$TH)
hospital$TRAUMA = factor(hospital$TRAUMA)
hospital$REHAB = factor(hospital$REHAB)

u <- rpart(log(1 + SALES12) ~ ., data = hospital, control = rpart.control(cp = .01))
plot(u)
text(u)

u <- rpart(log(1 + SALES12) ~ ., data = hospital, control = rpart.control(cp = .001))
plot(u, uniform = T)
text(u)
```

Regression Tree for log(1+Sales)

```r
HIP95 < 40.5 [Ave: 1.074, Effect: -0.76 ]
HIP96 < 16.5 [Ave: 0.775, Effect: -0.028 ]
RBEDS < 59 [Ave: 0.659, Effect: -0.117 ]
HIP95 < 0.5 [Ave: 1.09, Effect: +0.431 ] -> 1.09
HIP95 >= 0.5 [Ave: 0.551, Effect: -0.108 ]
KNEE96 < 3.5 [Ave: 0.375, Effect: -0.175 ] -> 0.375
KNEE96 >= 3.5 [Ave: 0.99, Effect: +0.439 ] -> 0.99

RBEDS >= 59 [Ave: 1.948, Effect: +1.173 ] -> 1.948
HIP96 >= 16.5 [Ave: 1.569, Effect: +0.495 ]

FEMUR96 < 27.5 [Ave: 1.201, Effect: -0.368 ] -> 1.201
FEMUR96 >= 27.5 [Ave: 1.784, Effect: +0.215 ] -> 1.784

HIP95 >= 40.5 [Ave: 2.969, Effect: +1.136 ]
KNEE95 < 77.5 [Ave: 2.493, Effect: -0.475 ]

Beds < 217.5 [Ave: 2.128, Effect: -0.365 ] -> 2.128
Beds >= 217.5 [Ave: 2.841, Effect: +0.348 ]
OUTV < 53937.5 [Ave: 3.108, Effect: +0.267 ] -> 3.108
OUTV >= 53937.5 [Ave: 2.438, Effect: -0.404 ] -> 2.438

KNEE95 >= 77.5 [Ave: 3.625, Effect: +0.656 ]
SIR < 9451 [Ave: 3.213, Effect: -0.412 ] -> 3.213
SIR >= 9451 [Ave: 3.979, Effect: +0.354 ] -> 3.979
```

Classification tree:

```r
data(tissue)
gr = rep(1:3, c(11, 11, 19))
x <- f.pca(f.toarray(tissue)$scores[, 1:4])
x = data.frame(x, gr = gr)
library(rpart)
t = rpart(factor(gr) ~ ., data = x)
n = 41

node, split, n, loss, yval, (yprob)
* denotes terminal node

1) root 41 22 3 (0.26829268 0.26829268 0.46341463)
2) PC< -0.9359889 23 12 1 (0.47826087 0.47826087 0.04347826)
3) PC1< -1.154355 12 1 1 (0.91666667 0.00000000 0.08333333) *
4) PC2< -1.154355 11 0 2 (0.00000000 1.00000000 0.00000000) *
5) PC3< -0.9359889 19 0 3 (0.00000000 0.00000000 1.00000000) *

plot(t)
text(t)
```
Random forest Algorithm (A variant of bagging)

1. Select ntree, the number of trees to grow, and mtry, a number no larger than number of variables.
2. For i = 1 to ntree:
3. Draw a bootstrap sample from the data. Call those not in the bootstrap sample the "out-of-bag" data.
4. Grow a "random" tree, where at each node, the best split is chosen among mtry randomly selected variables. The tree is grown to maximum size and not pruned back.
5. Use the tree to predict out-of-bag data.
6. In the end, use the predictions on out-of-bag data to form majority votes.
7. Prediction of test data is done by majority votes from predictions from the ensemble of trees.

R-package: randomForest with function called also randomForest

Boosting (Ada boosting)

Input:
Data \((x_i, y_i)\) \(i=1, \ldots, n\) ; \(w_i = 1/n\)

1. Fit tree or any other learning method: \(h_i(x)\)
2. Calculate misclassification error \(E_i\)
3. If \(E_i > 0.5\) stop and abort loop
4. \(b_i = E_i / (1 - E_i)\)
5. for \(i=1, \ldots, n\) if \(h_i(x) = y_i\) \(w_i = w_i b_i\) else \(w_i = w_i\)
6. Normalize the \(w_i's\) to add up to 1.
7. Go back to 1. and repeat until no change in prediction error.

R-package: bagboost with function called also bagboost and also adaboost

Paradigm for data mining: Paradigm for data mining: Paradigm for data mining: Paradigm for data mining:

Selection of interesting subsets

Recursive Partition:
- Find the partition that best approximates the response.
- For moderate/large datasets partition tree maybe too big

Bump Hunting:
Find subsets that optimize some criterion.
Subsets are more "robust"
Not all interesting subsets are found
Data Mining Trees: ARF

SPLIT FOR CONTINUOUS DESCRIPTORS

Naive thought: For the \( j \)th descriptor variable \( x_j \), an “interesting” subset \( \{ a < x_j < b \} \) is one such that

\[
p = \text{Prob}[Z=1 \mid a < x_j < b]
\]

is much larger than

\[
\pi = \text{Prob}[Z=1].
\]

\[T = (p - \pi)/\sigma_p\] measures how interesting a subset is.

Add a penalty term to prevent selection of subsets that are too small or too large.

Data mining tree (ARF)

Method: Select the variable and subset that maximizes

\[T = (p - \pi)/\sigma_p + \lambda \min \{ \log(h), \log(fN)/\log(fN) \}\]

where \( \lambda \) and \( f \) are a prespecified constants and \( h \) is the number of observations within the interval.

Iteration: Iterate the process (like growing a classification tree) a few times until no significant intervals are found.

Minimum bucket size: 15-20 cases (5 is recommended by CART but it is too small)

Continuous response: subsets with high mean or high median

Categorical Predictors: Split into groups

Case Study: Pima Indians Diabetes

- 768 Pima Indian females, 21+ years old
- 268 tested positive to diabetes

Variables:

- PRG: Number of times pregnant
- PLASMA: Plasma glucose concentration in saliva
- BP: Diastolic Blood Pressure
- THICK: Triceps skin fold thickness
- INSULIN: Two hours serum insulin
- BODY: Body mass index(Weight/Height)
- PEDIGREE: Diabetes pedigree function
- AGE: In years
- RESPONSE: 1: Diabetes, 0: Not
**Methodology**

1. **Methodology Objective:**
   
   The Data Space is divided between
   - High response subsets
   - Low response subsets
   - Other

2. **Categorical Responses:**
   Subsets that have high response on one of several categories. The categorical response is converted into several Binary responses.

3. **Continuous Responses:** High mean or low mean response

4. **Categorical Predictors:** Two groups.

5. **Data Visualization:**

6. **PDF report:**

---

**Data mining tree**

**CART Tree**

**CART Tree**

**CART Tree**

**CART Tree**

---

**Subset %Success n**

1. PLASMA in [155,199] & BODY in [29.9,45.7] & PEDIGREE in [0.344,1.394] - 96.364 55
3. PLASMA in [0.127] & AGE in [29.56] - 35.176 199
**Report**

Simple Tree: Only statistically significant nodes.

Full Tree: All nodes.

Table of Numerical Outputs: Detailed statistics of each node

List of Interesting Subsets: List of significant subsets

Conditional Scatter Plot (optional): Data Visualization.

---

**How to Use it**

1. Import the data into R:
   ```r
   library(foreign)
   w <- read.xport("C:/crf155.xpt")
   ```

2. Make sure that all the factors are declared as factors.
   ```r
   sapply(w,is.factor)
   w$x = factor(w$x)
   ```

3. Recode factors: (sometimes),
   ```r
   > levels(w$MusAnkR)
   > musc =f.recode(w$MusAnkR,c(""="Mild","Mod","Norm"))
   > w$musc = musc
   ```

4. Run ARF
   ```r
   mod = f.arf(RSP30 ~ pain0+CSITE+RXGP, data=w,
               highres=c(0,1))
   f.report(mod,file="c:/report.pdf")
   ```

---

**Data Visualization**

Conditional Plot: Condition on one or two variables.

---

**How to Use it**

4. More options
   ```r
   mod1 = f.arf(RSP30 ~ pain0+Wk1chng+RXGP, data=w,
                highres="1",
                varlist=c("RXGP","Wk1chng","RXGP"))
   f.report(mod1,file="c:/repl1rstmeasure.pdf")
   ```

5. More options
   ```r
   mod2 = f.arf(RESPONSE ~ SEX + BBPRS + ANERGIA + SMOKEYN +
                 BCGIS + MNBARNES + MNAIMS + dose, data = all,
                 highres = c("YES","NON","ICR"))
   f.report(mod2,file="c:/rep2Bprs.pdf")
   ```