DECISION TREE

Di Ming

Department of Computer Science and Engineering

The University of Texas at Arlington

1. Introduction

- Tree-/Flowchart-like structure

Internal node: a "test" on an attribute

Branch: the outcome of the test

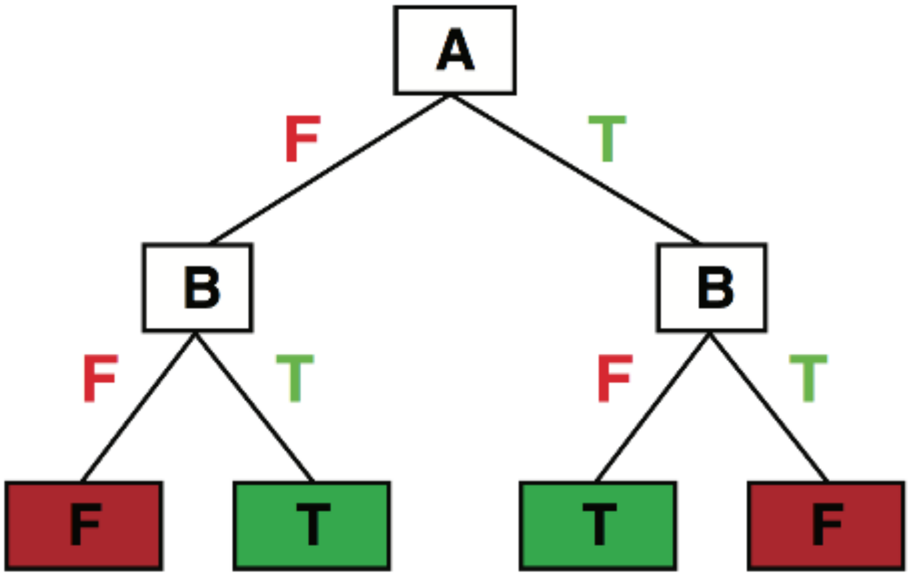
Leaf node: a class label

Path: classification rules from root to leaf

2. Background

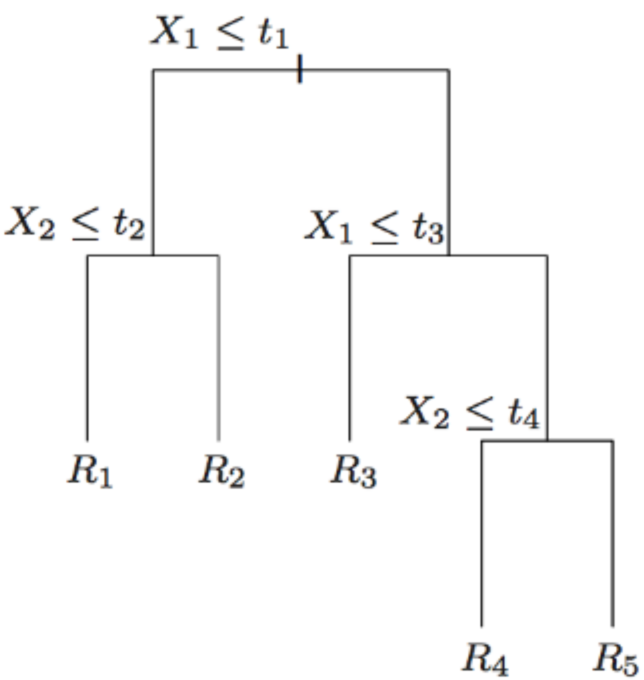
- Toy example: Boolean function "***XOR***"

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



2. Background

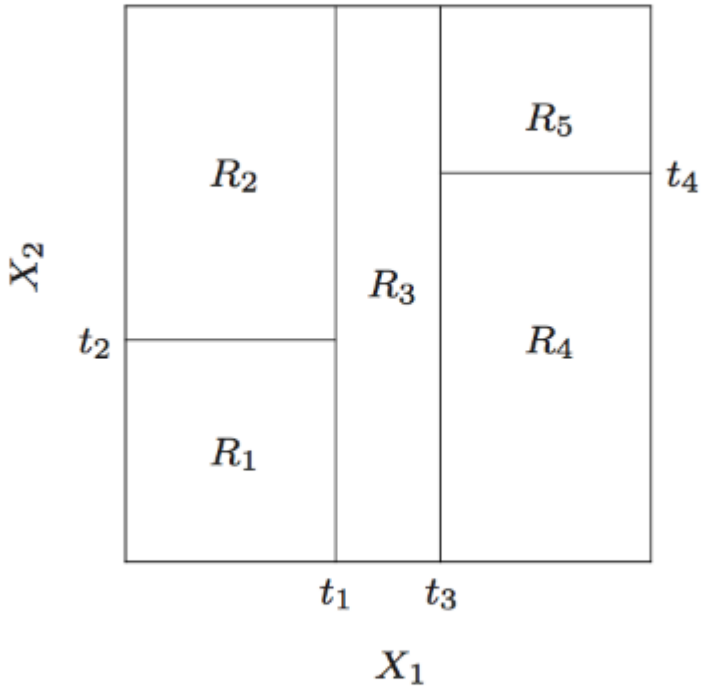
- Tree-based Method



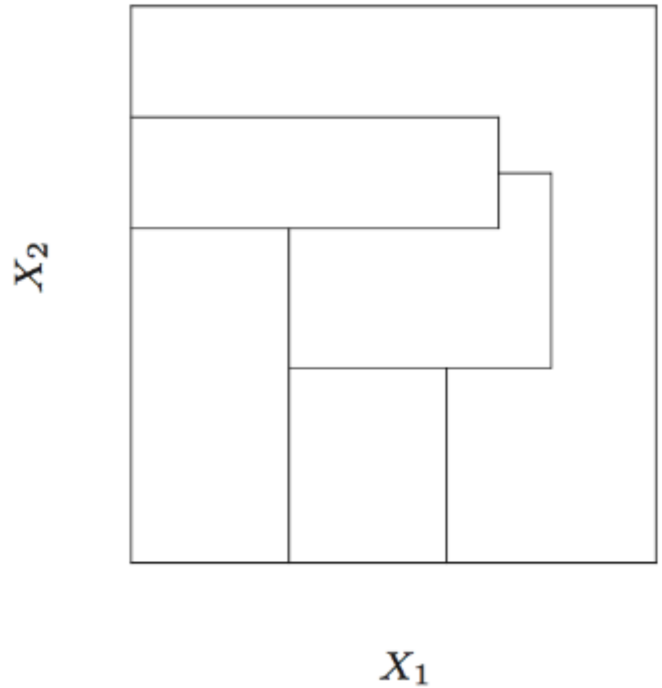
X - features or attributes
t - splitting points
R - label(classification) or response(regression)

2. Background

- Recursive **Binary** Partitions
- Partition **feature space** into **rectangles**



Easy to describe !!!

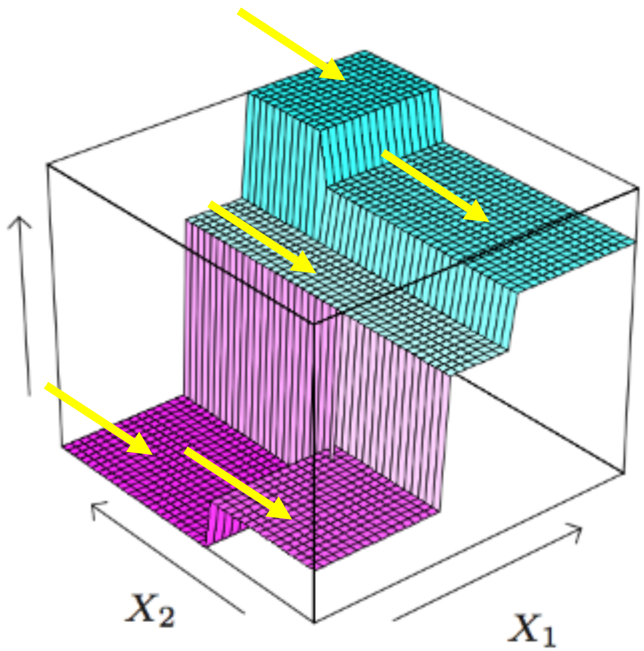


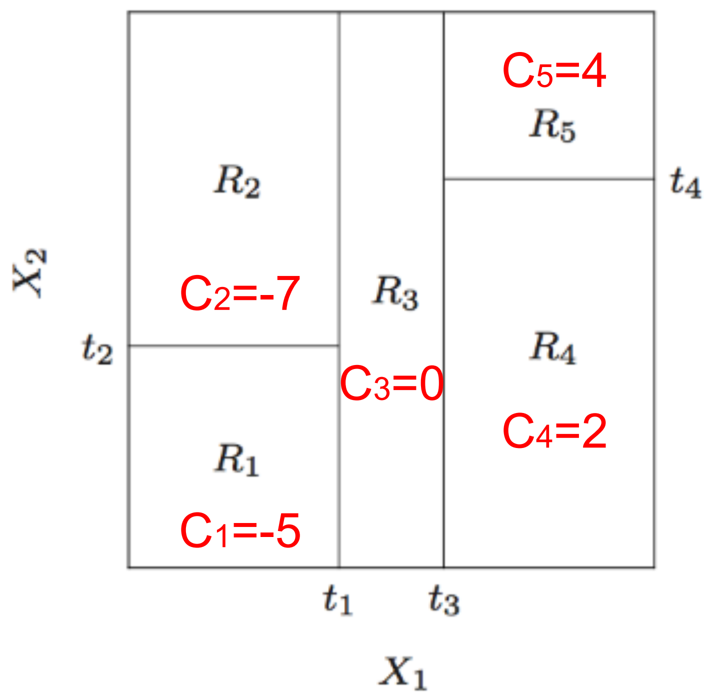
Complicated to describe !!!

2. Background

- 3D visualization

Different **constants** represents **prediction area**.





2. Background

- The corresponding regression model

$$\hat{f}(X) = \sum_{m=1}^{5} c_m \cdot \underbrace{I\{(X_1, X_2) \in R_m\}}_{\textit{indicator function}}$$

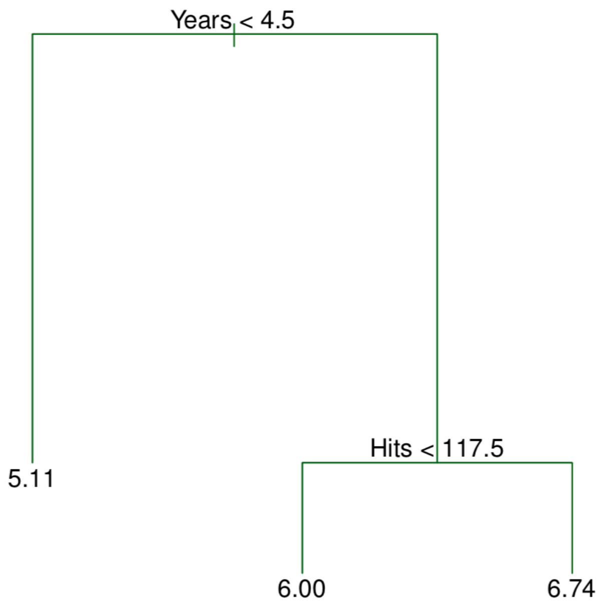
- Where $\underbrace{c_1\text{=-5, } c_2\text{=-7, } c_3\text{=0, } c_4\text{=2, } c_5\text{=4}}_{\textit{responses}}$.

- If c_m is defined as probability, then we have classification model: $Y = \hat{f}(X)$.

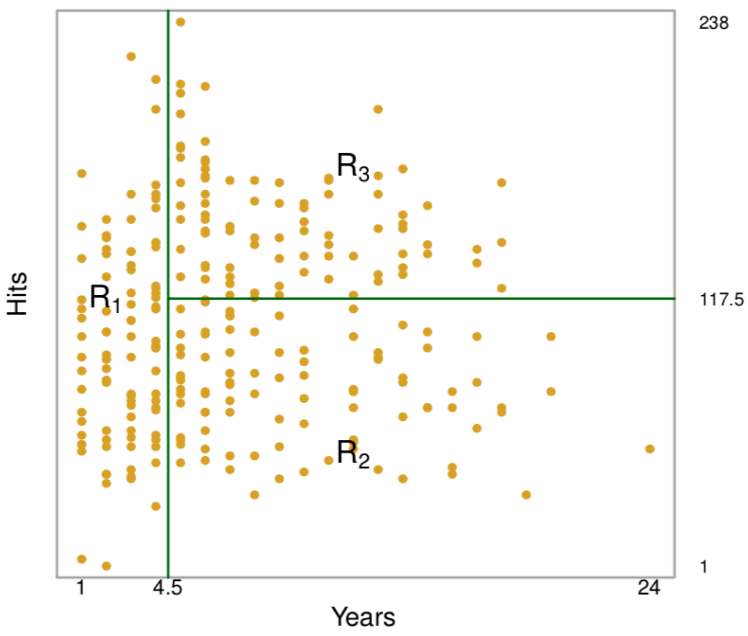
2. Backgroud

- Example: Predicting a baseball player's salary.

Decision Tree



Partition of Feature Space



Model $\hat{f}(X) = \sum_{m=1}^{3} c_m \cdot I\{(X_1, X_2) \in R_m\}$

where $c_1 = 5.11,\ c_2 = 6.0,\ c_3 = 6.74.$

3. Regression Tree

- Definition

Given N observations: (x_i, y_i). $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$

Partition into M regions: $\{R_1, R_2, \ldots, R_M\}$

Response in each region: $\{c_1, c_2, \ldots, c_M\}$

- Regression tree model can be described as:

$$\hat{f}(X) = \sum_{m=1}^{M} c_m \cdot I\{x \in R_m\}$$

3. Regression Tree

- Minimization in a single region R_m

$$\min\ J = \sum_i (y_i - f(x_i))^2 = \sum_i (y_i - c_m)^2$$

- Setting the derivative of J w.r.t. c_m^* to zero:

$$-2\sum_i (y_i - c_m^*) = 0$$

$$c_m^* = {\sum_i y_i}\Big/{\sum_i 1} = avg(y_i | x_i \in R_m)$$

The best c_m is just the average of y_i in region R_m.

3. Regression Tree

- Find best binary partition with all of the data via minimizing sum of squares?

Computationally infeasible !!!

- How to approximately solve the problem?

*Proceed with a **greedy** algorithm.*

3. Regression Tree

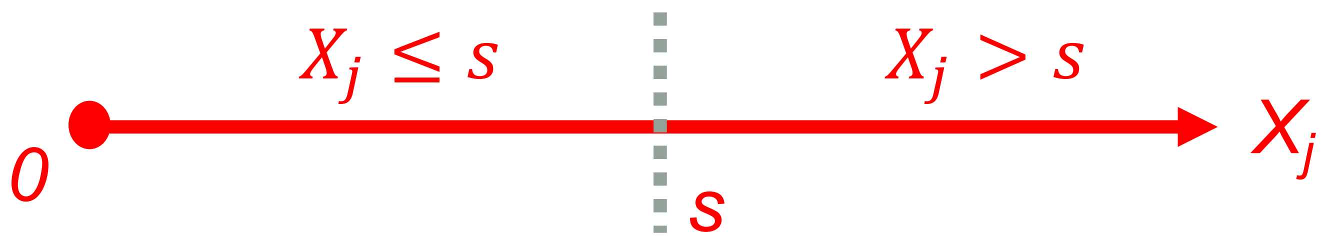
- **Greedy solver**

Consider: splitting variable j and point s

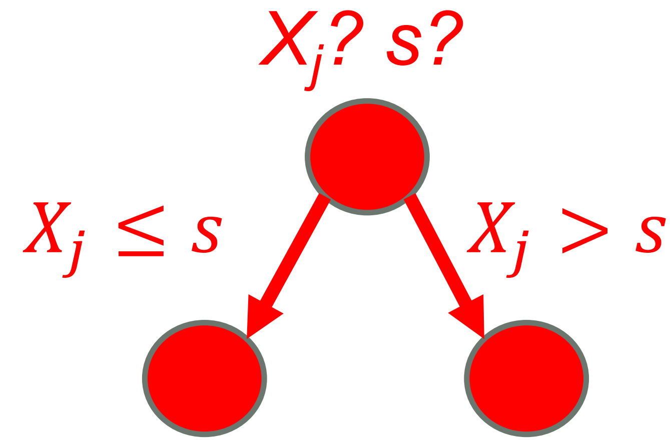
- **Pair of half-planes** is defined as

$$R_1(j,s) = \{X \mid X_j \le s\}$$
$$R_2(j,s) = \{X \mid X_j > s\}$$

e.g. one dimensional feature space:



3. Regression Tree



- Seeking the best *j* and s:

$$min_{j,s}\{min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2\}$$

- For any choice j and s, the inner minimization is solved by:

$$\begin{aligned} c_1 &= avg(y_i | x_i \in R_1(j,s)) \\ c_2 &= avg(y_i | x_i \in R_2(j,s)) \end{aligned}$$

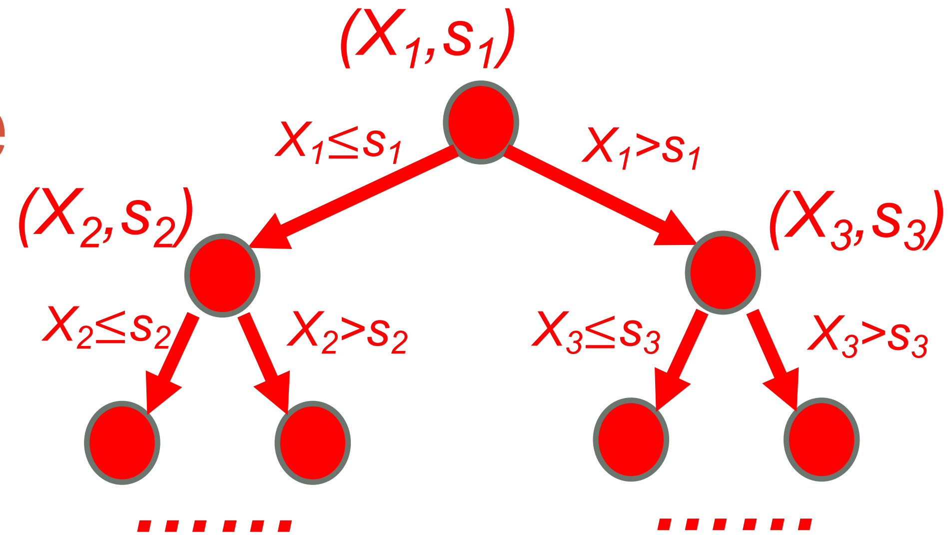
3. Regression Tree

Therefore, best *(j,s)* are discovered as follows

1) *Best* s^*: the optimal splitting point *s* can be determined by *solving inner minimization*.

2) *Best* j^*: the optimal splitting pair (j,s) will be found via scanning all the inputs X_1, X_2, …, X_p, which achieves *the smallest loss*.

3. Regression Tree



- Building decision tree:

1) Partition data in two regions using the best splitting pair ***<j,s>***.
2) Repeat the splitting process on each of the two regions.
3) Then the process (including first two steps) is repeated on all the resulting regions.

3. Regression Tree

- How large should we grow the tree?

Large tree might overfit the data.

Small tree might underfit the data.

- *Tree size* is tuning parameter controlling the model complexity.

- The *optimal tree size* should be *adaptively chosen* from the data.

3. Regression Tree

- *One approach*:

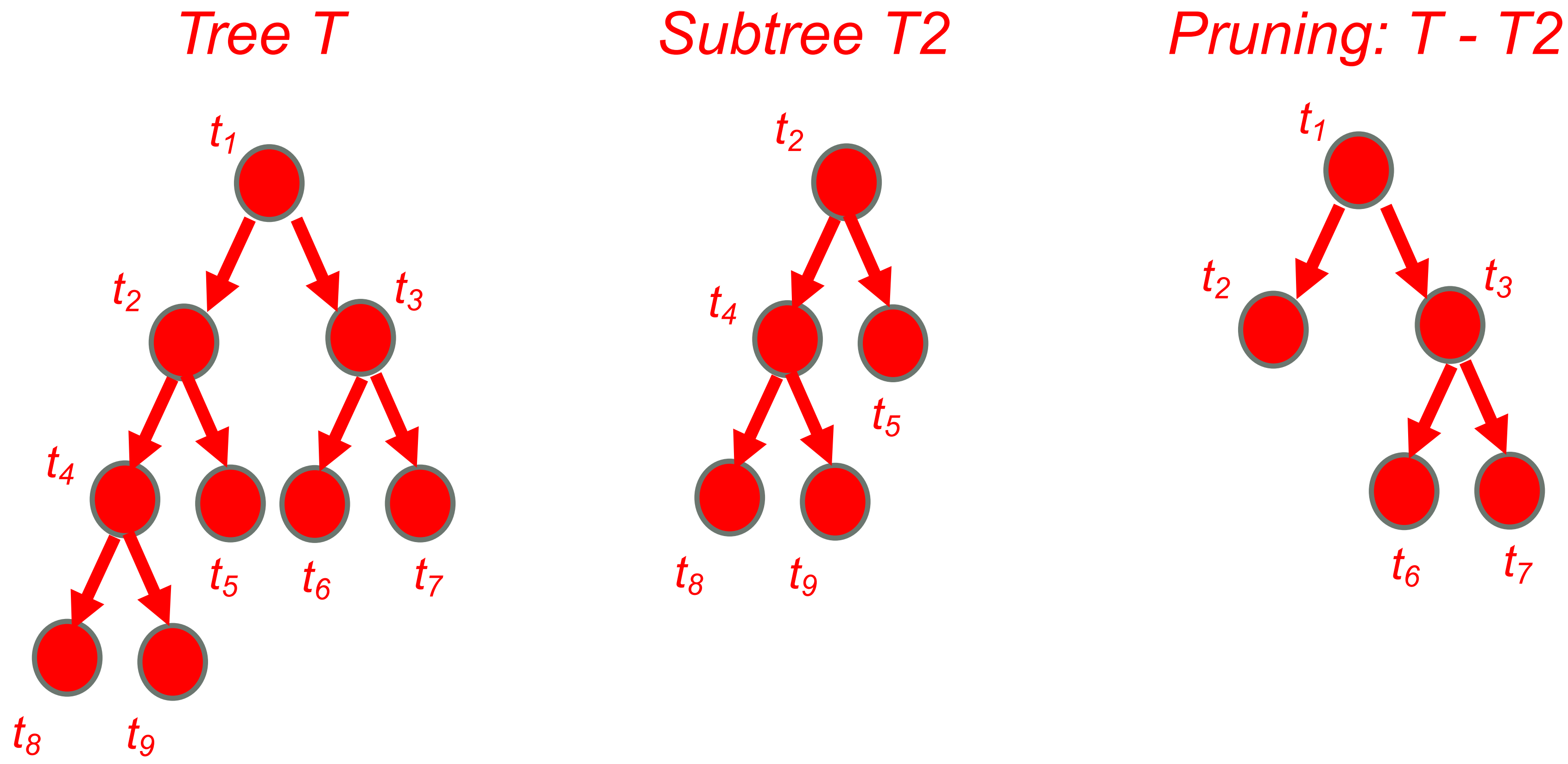
To split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold.

- *Preferred strategy*:

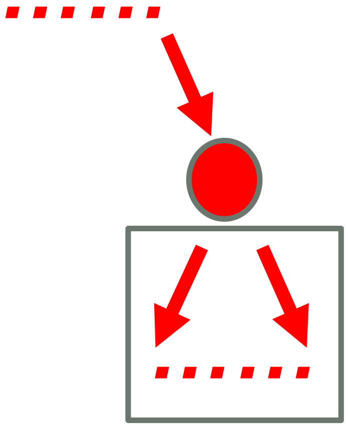
1) Grow a large tree T0.
2) Stopping the splitting process only when some minimum node size (say 5) is reached.
3) Prune tree T0.

3. Regression Tree

- Pruning



3. Regression Tree - Pruning



- ***Reduced error pruning***

1) Consider each node for pruning.
2) Pruning: *removing the subtree* at that node, make it a leaf and assign *the average response* or *the most common class* at that node.
3) A node is removed if *the resulting tree* performs no worse than the original tree on the validation set.
4) Nodes are removed iteratively choosing the node *whose removal most increases the model accuracy*.
5) Pruning continues until *further pruning is harmful*.

3 Regression Tree - Pruning

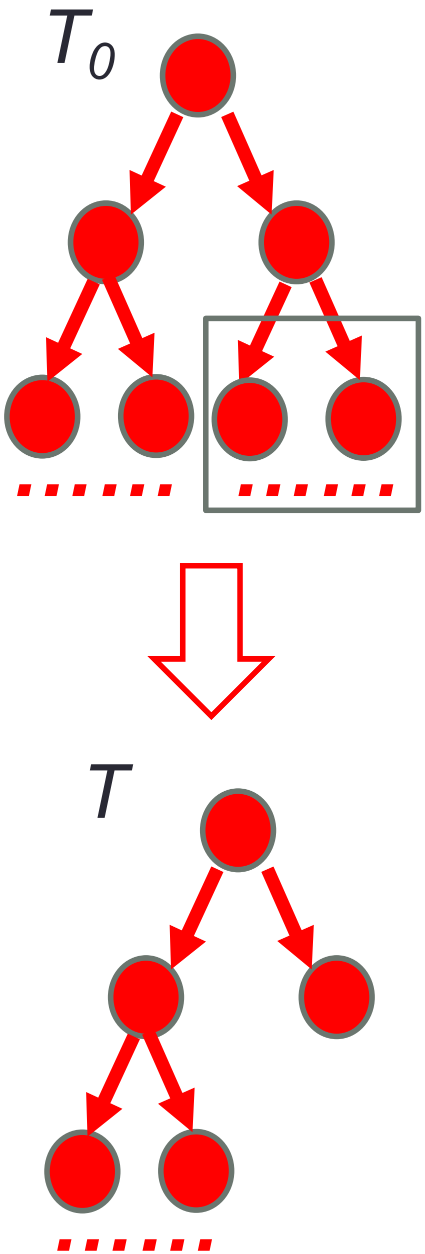
- ***Cost-complexity pruning***

Defining a subtree $T \subset T_0$

$$N_m = \{x_i \in R_m\}$$

$$c_m^* = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - c_m^*)^2$$



3. Regression Tree

- The cost complexity criterion is defined as

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

where $|T|$ is the number of *leaf nodes* in T.

- Tuning parameter α ≥ 0 governs the tradeoff between *tree size* and *sum-of-square error*.

- Using *weakest link pruning* to find best T_a.

4 Classification Tree

- Target (Categorical labels): *1,2,…,K*.

- Node *m* (Region R_m with N_m observation) is represented as:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

- Classify the observation in node m :

$$k(m) = argmax_k \, \hat{p}_{mk}$$

4. Classification Tree

- Different measures for $Q_m(T)$

Misclassification error:

$$\frac{1}{N_m}\sum_{x_i \in R_m} I(y_i \neq k) = 1 - \hat{p}_{mk}$$

Gini index:

$$\sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Cross-entropy (deviance):

$$-\sum_{k=1}^{K} \hat{p}_{mk}\log(\hat{p}_{mk})$$

4 Classification Tree

- Comparison of different measures
- Consider two classes: $<p, 1-p>$

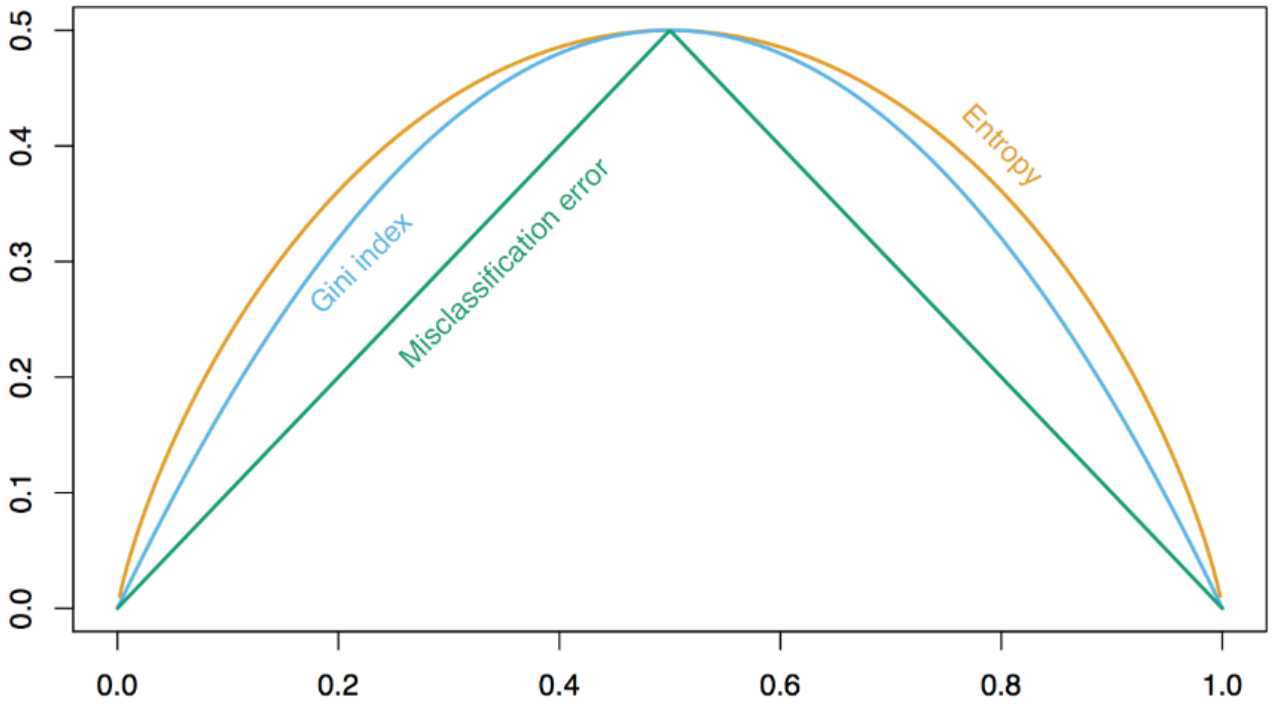
Misclassification error: $1 - max(p, 1 - p)$

Gini index: $2p(1 - p)$

Cross-entropy: $-plogp - (1 - p)log(1 - p)$

4 Classification Tree

- Comparison of different measures



**cross-entropy is scaled to pass through (0.5,0.5).*

5 Random Forest

- Random forest(RF) is a classifier
- RF is built on a forest of decision trees
- Classify a new data object x : every decision tree assigns a label l. The final over-all class label for x is obtained by majority voting, same as in KNN.
  - Why majority voting is good?
- Random forest typically performs very well, similar to SVM, logistic regression.
- RF is an example of ensemble learning (bagging)

5 Random Forest

- Building (training) the RF:
  - Repeat for each decision tree
  - Randomly split training data into (X_train, X_test)
  - From this X_train, build a decision tree
    - But in spliting a node, we only consider a limited number of features (instead of all features in standard decision tree construction)
    - This limited number of features are randomly chosen at each node.
    - The number of this feature set is input parameter. This number could be all features
  - For each decision tree, because the training data is different (and also the feature set could differ), the constructed decision tree is different
  - X_test is used to compute the classification error of this decision tree. This is out-of-bag error (oobERROR). No cross-validation is necessary.

6. References

- “The Elements of Statistical Learning”.

1) *9.2 Tree-Based Methods (Page #305).*

2) *15 Random Forests (Page #587).*