Decision Trees

Robert M. Haralick

Computer Science, Graduate Center
City University of New York

Making a Distinction

- Probability
  - Bayes: Discrete P(c,d)
  - Subspace Methods
  - Logistic Regression: P(c | d)
    - Arbitrary Function
    - Linear
  - Class Conditional Gaussian: P(d | c)
    - Quadratic
    - Equal Class Covariance Matrices: Linear
- Boundary Modeling
  - Decision Tree
  - Fisher Linear Rule
  - Support Vector Machine

Decision Trees: Binary Recursive Partitioning

Definition

A Decision Tree is a classifier whose structural form is a tree.

- Each node of the tree at the same tree level corresponds to a mutually exclusive subset of measurement space
- The nodes of the tree are either decision nodes or leaf nodes
- At each decision node of the tree a distinction is made that partitions its subset of measurement space
- Each leaf node is associated with an assigned class

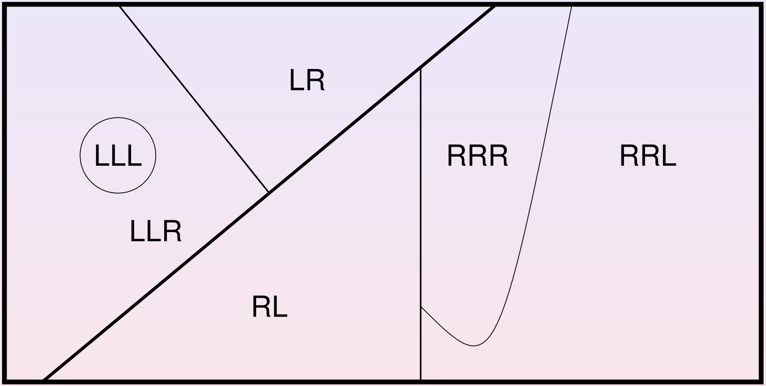
Advantages

- Understandable rules
- Quick On-line computation
- Continuous or categorical variables.
- Provide a clear indication of which dimensions are most relevant for accurate classification

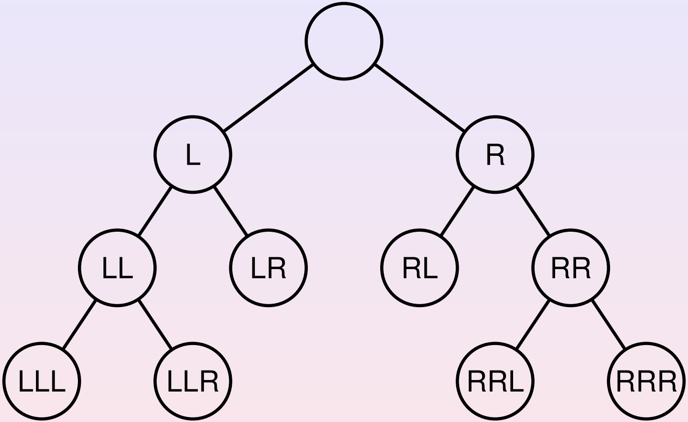
Disjunction of Conjunctions

- On any branch down the tree, the decision region is specified by the conjunction of the constraints of the nodes in the branch
- There are many branches, each of which represents a disjunction of these conjunctions

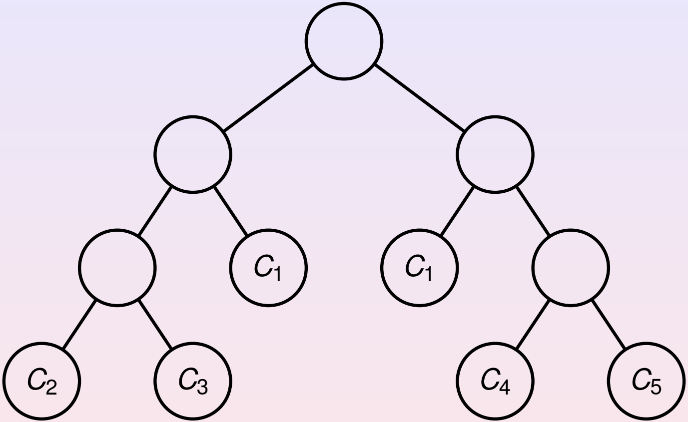
Measurement Space Partitioning



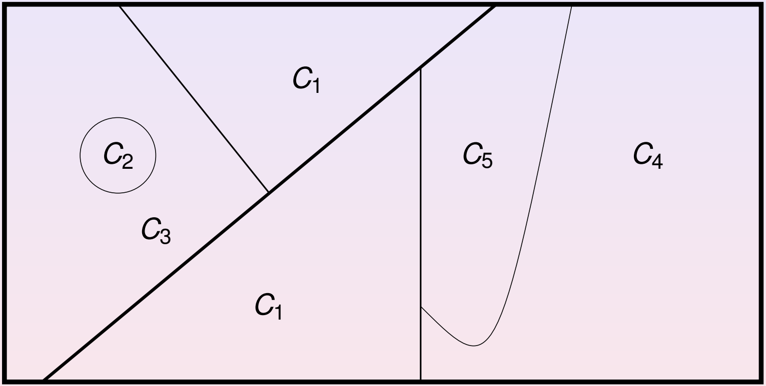
Decision Tree

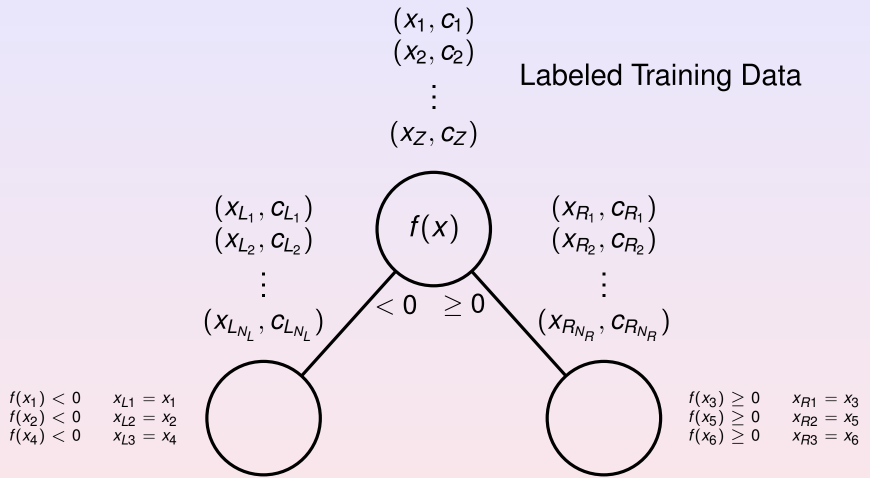


Decision Tree Leaf Nodes



Measurement Space Partitioning





Each x is an N-tuple

Decision Node Issues

- What to Distinguish
- How to Distinguish
- How to evaluate the goodness of a Distinguishment

What to Distinguish

- One subset of classes from another
  - One class from the others
    - c_2 from $c_1, c_3, \ldots, c_K$
  - Two or more classes from the others
    - c_2, c_4 from $c_1, c_3, c_5, \ldots, c_K$

How to Distinguish

- The tuple $x = (x_1, \ldots, x_N)$
- A threshold t
- By component of the tuple
  - Decide using x_n
  - If $x_n < t$, go left; else go right
- By Linear Decision Rule Distinguish one group of classes from its complement
  - Decide using $r = \sum_{n=1}^{N} w_n x_n$
  - if $r < t$, go left; else go right
- By Quantizing and Using Table-Lookup
  - $M \leq N$
  - $\{i_1, \ldots, i_M\} \subseteq \{1, \ldots, N\}$
  - $a = address(q_1(x_{i_1}), \ldots, q_M(x_{i_M}))$
  - $T(a) < 0$, go left; else go right
- Distance to a point q
  - If $||x - q|| < t$ go left; else go right

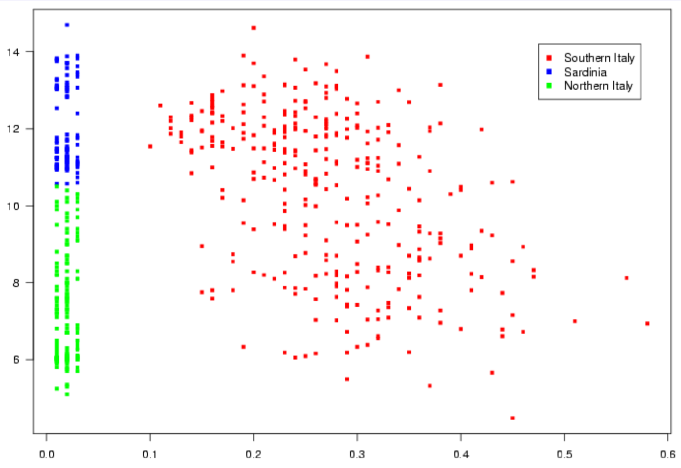
,

Where Did the Olives Come From?

- Classes
  - Northern Italy
  - Southern Italy
  - Sardinia
- Fatty Acid Measurements
  - Eicosenoic: x_1
  - Linoleic: x_2

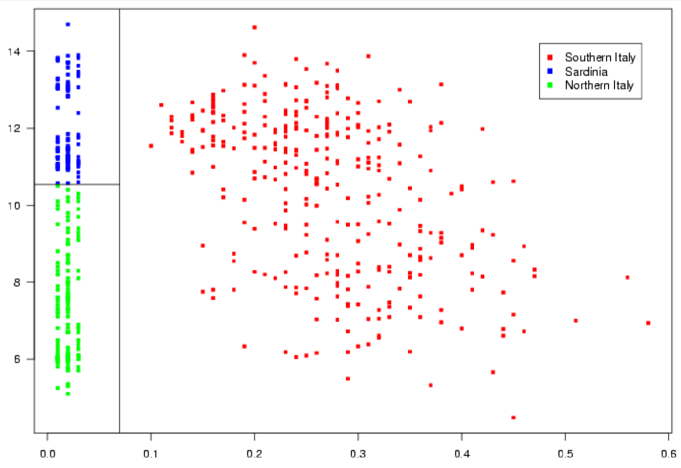
Olives

Linoleic



Eicosenoic

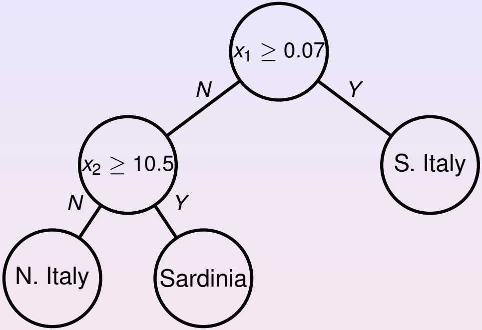
Olives



Linoleic

Eicosenoic

Decision Tree



Constructing The Tree

- Suppose we are constructing the tree
- We are at one node
- Coming into the node is the training set for the node
  - The sequence of tuples and their classes
- The node will make a distinction solely based on the tuples
- The classes of the tuples will be used for evaluation
- After making a distinction, the training set is partitioned into two cells
  - One cell of tuples and their classes on the left
  - The other cell of tuples and their classes on the right

Constructing The Tree

- For each class c on the left, there is a probability that it occurs there: $P_L(c)$
- For each class c on the right, there is a probability that it occurs there: $P_R(c)$
- The purpose of making the distinction is to separate one class from another
- If there are two classes the best that can happen is to
  - Have all of one class on the left
  - Have all of the other class on the right
- The worst thing to happen is to have an equal mixture of the classes
  - Each class having probability $1/2$ on the left and on the right
- There needs to be an evaluation of a distinction
- So that a distinction could be chosen that does the best job of separating the classes

Impurity Function

Definition

Let C be an index set for K classes. The probability of class k occurring is p_k. A function ϕ is an Impurity Function for K classes if and only if

- It is defined on the K-dimensional simplex $\{(p_1, \ldots, p_K) \mid p_k \geq 0, k = 1, \ldots, K;\ \sum_{k=1}^{K} p_k = 1\}$
- Maximum only at $(\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K})$
- Minimum only at the points $(1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 0, 1)$
- Symmetric function: for any permutation $\pi_1, \ldots, \pi_K$ of $1, \ldots, K$, $\phi(p_1, \ldots, p_K) = \phi(p_{\pi_1}, \ldots, p_{\pi_K})$

The smaller the value the Impurity Function has the better.

Purity Function

Definition

Let C be an index set for K classes. The probability of class k occurring is p_k. A function ϕ is an Purity Function for K classes if and only if

- It is defined on the K-dimensional simplex $\{(p_1, \ldots, p_K) \mid p_k \geq 0, k = 1, \ldots, K;\ \sum_{k=1}^{K} p_k = 1\}$
- Minimum only at $(\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K})$
- Maximum only at the points $(1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 0, 1)$
- Symmetric function: for any permutation $\pi_1, \ldots, \pi_K$ of $1, \ldots, K$, $\phi(p_1, \ldots, p_K) = \phi(p_{\pi_1}, \ldots, p_{\pi_K})$

The larger the value a Purity Function has the better.

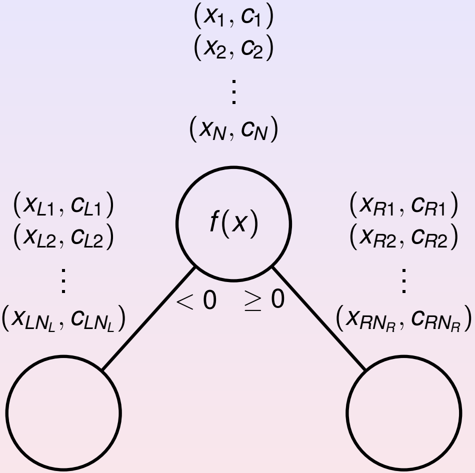
Evaluation of Distinction Choice

- Impurity Function of node class probabilities
  - Entropy of the class probabilities
    - $E_L = -\sum_c P_L(c) \log P_L(c)$
    - $E_R = -\sum_c P_R(c) \log P_R(c)$
    - $E = E_L P_L + E_R P_R$
  - Gini Index of Diversity
    - $G = \sum_{c \in C} \sum_{\{c' \in C | c \neq c'\}} P_L(c) P_R(c') = 1 - \sum_{c \in C} P_L(c) P_R(c)$
  - Misclassification
    - $M_L = 1 - \max_c P_L(c)$
    - $M_R = 1 - \max_c P_L(c)$
    - $M = P_L M_L + P_R M_R$
- Purity Function of node class probabilities
  - Purity Index = 1-Gini Index = $\sum_{c \in C} P_L(c) P_R(c)$
- Twoing Criterion
  - $\left| \frac{P_L P_R}{4} \left(\sum_c |P_L(c) - P_R(c)| \right)^2 \right|$

Algorithm

- Go through all possible distinctions that have been chosen to be used
- For each distinction, evaluate the result
- Select the best distinction
- Repeat Until Node Training Set is too small
  - Make node leaf node
  - Assign majority class

Decision Tree Node



Entropy After Distinction

- C^L Subset of Left Classes
- C^R Subset of Right Classes
- $C^* = C^L \cup C^R$
- $\langle (x_{L_1}, c_{L_1}), (x_{L_2}, c_{L_2}), \ldots (x_{L_{N_L}}, c_{L_{N_L}}) \rangle$ Left Child Data
- $\langle (x_{R_1}, c_{R_1}), (x_{R_2}, c_{R_2}), \ldots (x_{R_{N_R}}, c_{R_{N_R}}) \rangle$ Right Child Data
- $P_L(c) = \frac{\#\{n \mid c_{L_n} = c\}}{N_L}, c \in C^*$
- $P_R(c) = \frac{\#\{n \mid c_{R_n} = c\}}{N_R}, c \in C^*$
- $E_L = -\sum_{c \in C*} P_L(c) \log(P_L(c))$
- $E_R = -\sum_{c \in C*} P_R(c) \log(P_R(c))$
- $P_L = \frac{N_L}{N_L + N_R}, P_R = \frac{N_R}{N_L + N_R}$
- $E = P_L E_L + P_R E_R$
- The Smaller the Entropy the Better

Misclassification Rate After Distinction

- C^L Subset of Left Classes
- C^R Subset of Right Classes
- $C^* = C^L \cup C^R$
- $\langle (x_{L_1}, c_{L_1}), (x_{L_2}, c_{L_2}), \ldots, (x_{L_{N_L}}, c_{L_{N_L}}) \rangle$ Left Child Data
- $\langle (x_{R_1}, c_{R_1}), (x_{R_2}, c_{R_2}), \ldots, (x_{R_{N_R}}, c_{R_{N_R}}) \rangle$ Right Child Data
- $P_L(c) = \frac{\#\{n \mid c_{Ln} = c\}}{N_L}, c \in C^*$
- $P_R(c) = \frac{\#\{n \mid c_{Rn} = c\}}{N_R}, c \in C^*$
- $M_L = 1 - \max_{c \in C*} P_L(c)$
- $M_R = 1 - \max_{c \in C*} P_R(c)$
- $P_L = \frac{N_L}{N_L + N_R}$, $P_R = \frac{N_R}{N_L + N_R}$
- $M = P_L M_L + P_R M_R$
- The Lower the Misclassification The Better

Purity After Distinction

- C^L Subset of Left Classes
- C^R Subset of Right Classes
- $C^* = C^L \cup C^R$
- $\langle (x_{L_1}, c_{L_1}), (x_{L_2}, c_{L_2}), \ldots (x_{L_{N_L}}, c_{L_{N_L}}) \rangle$ Left Child Data
- $\langle (x_{R_1}, c_{R_1}), (x_{R_2}, c_{R_2}), \ldots (x_{R_{N_R}}, c_{R_{N_R}}) \rangle$ Right Child Data
- $P_L(c) = \frac{\#\{n \in [1, L_{N_L}] \mid c_{L_n} = c\}}{N_L}, c \in C^*$
- $P_R(c) = \frac{\#\{n \in [1, N_R] \mid c_{R_n} = c\}}{N_R}, c \in C^*$
- $I_L = \sum_{c \in C*} P_L(c)^2$
- $I_R = \sum_{c \in C*} P_R(c)^2$
- $P_L = \frac{N_L}{N_L + N_R}, P_R = \frac{N_R}{N_L + N_R}$
- $I = P_L I_L + P_R I_R$
- The Larger the Purity the Better

Twoing Criterion After Distinction

- C^L Subset of Left Classes
- C^R Subset of Right Classes
- $C^* = C^L \cup C^R$
- $\langle (x_{L_1}, c_{L_1}), (x_{L_2}, c_{L_2}), \ldots, (x_{L_{N_L}}, c_{L_{N_L}}) \rangle$ Left Child Data
- $\langle (x_{R_1}, c_{R_1}), (x_{R_2}, c_{R_2}), \ldots, (x_{R_{N_R}}, c_{R_{N_R}}) \rangle$ Right Child Data
- $P_L(c) = \frac{\#\{n \in [1, N_L] \mid c_{L_n} = c\}}{N_L}, c \in C^*$
- $P_R(c) = \frac{\#\{n \in [1, N_R] \mid c_{R_n} = c\}}{N_R}, c \in C^*$
- $P_L = \frac{N_L}{N_L + N_R}, P_R = \frac{N_R}{N_L + N_R}$
- $\left| \frac{P_L P_R}{4} \left(\sum_{c \in C^*} |P_L(c) - P_R(c)| \right)^2 \right|$
- The Larger the Twoing Criterion the Better

Distinction By Feature

Consider the training tuple sequence. $\langle x_1, x_2, \ldots, x_Z \rangle$. Arrange a matrix with x_n as the n^{th} row.

$$\begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1N} \\ x_{21} & x_{22} & \ldots & x_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ x_{z1} & x_{z2} & \vdots & x_{zN} \\ \vdots & \vdots & \vdots & \vdots \\ x_{Z1} & x_{Z2} & \ldots & x_{ZN} \end{pmatrix}$$

Distinction By Feature

- Node Data: $(x_1, c_1), (x_2, c_2), \ldots (x_N, c_N)$
- $x_n = (x_{n1}, x_{n2}, \ldots, x_{nK})$
- For each component k sort in ascending order: $x_{(1)k} \leq x_{(2)k} \leq \ldots \leq x_{(N)k}$
- For each $(n, k) \in \{1, \ldots, N\} \times \{1, \ldots, K\}$
  - n defines the threshold
  - k defines the component
  - Define $f_{nk}(z_1, \ldots, z_K) = z_k - (x_{(n)k} + x_{(n+1)k})/2$
  - If $f_{nk}(z_1, \ldots, z_K) < 0$ go left; else go right
- Let (n^*, k^*) maximize the criterion
- Use $f_{n^*k^*}$ to make the distinction

Algorithm

- Go through all possible partitions of the classes present at the node
- For each class partition go through all possible distinctions
- For each class partition and each way of distinction, evaluate the result
- Select the best partition and the best way of distinction

All Possible Class Partitions

- $\{\{c_1\}, \{c_2, c_3, c_4\}\}$
- $\{\{c_2\}, \{c_1, c_3, c_4\}\}$
- $\{\{c_3\}, \{c_1, c_2, c_4\}\}$
- $\{\{c_4\}, \{c_1, c_2, c_3\}\}$
- $\{\{c_1, c_2\}, \{c_3, c_4\}\}$
- $\{\{c_1, c_3\}, \{c_2, c_4\}\}$
- $\{\{c_1, c_4\}, \{c_2, c_3\}\}$
- $\{C_L, C_R\}$

Best Distinction: Fisher Linear Discriminant

- $\{C_L, C_R\}$ Desired partition
- Node Data $(x_1, c_1), (x_2, c_2), \ldots (x_N, c_N)$
- Desired Left Child
  - $X_L = \{x_n \mid c_n \in C_L\}$
  - Mean $\mu_L = \frac{1}{N_L} \sum_{x \in X_L} x$
  - Scatter $S_L = \sum_{x \in X_L} (x - \mu_L)(x - \mu_L)'$
- Desired Right Child
  - $X_R = \{x_n \mid c_n \in C_R\}$
  - Mean $\mu_R = \frac{1}{N_R} \sum_{x \in X_R} x$
  - Scatter $S_R = \sum_{x \in X_R} (x - \mu_R)(x - \mu_R)'$
- Within Group Scatter $S_W = S_L + S_R$
- Between Group Scatter $S_B = (\mu_L - \mu_R)(\mu_L - \mu_r)'$
- Find w to maximize $J(w) = \frac{w' S_B w}{w' S_W w}$

All Possible Distinctions

- Find w to maximize $J(w) = \frac{w' S_B w}{w' S_W w}$
- $w = S_W^{-1}(\mu_L - \mu_R)$
- Node Data $(x_1, c_1), (x_2, c_2), \ldots (x_N, c_N)$
- $y_n = w' x_n$
- Sort $y_{(1)}, \ldots, y_{(N)}$
- $\theta_n = (y_{(n)} + y_{(n+1)})/2$
- Distinction Functions $f(x) = w'x - \theta_k$, $k = 1, \ldots, N-1$

Algorithm

- Go through all possible partitions of the classes present at the node
- For each class partition go through all possible distinctions
- For each class partition and each way of distinction, evaluate the result
- Select the best partition and the best way of distinction

Stopping Criterion

- Data in node is all of same class
- Node is at maximum tree depth
- Number of instances in node is too small
- Best splitting criteria is smaller than a threshold
- Cross Validation

Cross Validation

- Divide Training set into Q parts, $L_1, \ldots, L_Q$
- Use $L_1, \ldots, L_{Q-2}$ to develop tree
- Use L_{Q-1} to determine if a node lives
  - The incoming data to a node has an error rate
  - The children nodes have an error rate
  - If the children nodes have an error rate signicantly smaller than the parent node keep the children nodes
  - Else make the parent node a leaf node
- Use L_Q to estimate the error rate of the tree
- Then go round robin using $L_2, \ldots L_{Q-1}$ to develop the tree
- Use L_Q to determine if a node lives
- Use L_1 to estimate the error rate of the tree

Quality of Training Data

- Poor Training Data gives Poor Results
- Insufficient Data Sample
  - Does not Capture Distribution
  - Does not Reflect the Real World Distribution
- Number of Observations in Each Class
  - Does not reflect the class prior probabilities

Decision Forests

- Construct multiple decision trees
- Classify new tuple x by maximum a posterior probability

Multiple Decision Trees

- Setup
  - Training Sample of size N
  - Dimensionality M
  - Select $n < N$
  - Select $m < M$
- Repeat many times
  - From the training sample, randomly sample of size n
  - Randomly select m features
  - Construct Decision Tree using sample

A Posteriori Probability

- T Trees
- For new tuple x and tree t,
  - The leaf node for x has $N(t; x)$ tuples from the training set landing there
  - The number of tuples landing there whose true class is c is $n(c, t; x)$
  - Posterior probability for class c is $P(c \mid t) = \frac{n(c,t;x)}{N(t;x)}$
  - Prior probability for tree t is $P(t \mid x) = \frac{N(t;x)}{\sum_{s=1}^{T} N(s;x)}$
- A Posteriori Probability for class c

$$P(c \mid x) = \sum_{t=1}^{T} P(c \mid t, x) P(t \mid x)$$

Formal Statement

Let $T =< (y_n, x_n) >_{n=1}^{N}$ be the training data

- y_n is the response values
- x_n is the vector of predictor values
- $L(y, y')$ is the loss between y and its prediction y'

Find a function f to minimize

$$E[\sum_{n=1}^{N} L(y_n, f(x_n))]$$

- If y is real valued, the problem is a regression.
- If y is unordered labels, the problem is a classification problem

Strengths of Decision Trees

- Decision Rules are Understandable
- Online computation is quick
- Can handle continuous and categorical variables
- The variables that are important are the ones it uses

Weaknesses of Decision Trees

- The tree is not natural for estimating continuous values
  - Can use a regression for leaf nodes
- Does not Work Well with Many Classes
- Computationally Expensive to Train
- Decision boundaries are aligned with axes
  - Rectangular Regions