Lecture 03:
Classification:
Decision Trees

Rawls Profess of MIS
Jaeki Song, Ph.D.

Decision Trees

- Popular classification method in data mining - supervised classification method
  - collection of decision nodes, connected by branches, extending downward from root node to terminating leaf nodes
  - Beginning with root node, attributes tested at decision nodes, and each possible outcome results in branch
  - Each branch leads to decision node or leaf node

Decision Trees

- Pre-classified target variable must be included in training set
- The target variable must be categorical
- Decision trees learn by example, so training set should contain records with varied attribute values
- If training set systematically lacks definable subsets, classification becomes problematic
- Classification and Regression Trees (CART) and C4.5 are two leading algorithms used in data mining

Classification and Regression Trees (CART)

- Classification and Regression Trees (CART) developed by Breiman, 1984
  - Splits at decision nodes are binary, resulting in two branches
  - CART recursively partitions data into subsets with similar values for target variable
  - Algorithm grows tree by evaluating all predictor variables. Then chooses optimal split according to criteria:

Classification and Regression Trees (CART)

Let $\Phi(s \mid t)$ be a measure of the "goodness" of a candidate split s at node t, where

$$\Phi(s \mid t) = 2P_L P_R \sum_{j=1}^{\#classes} \left| P(j \mid t_L) - P(j \mid t_R) \right|$$

$$\begin{cases} t_L = \text{left child node of node } t \\ t_R = \text{right child node of node } t \\ P_L = \dfrac{\text{number of records at } t_L}{\text{number of records in training set}} \\ P_R = \dfrac{\text{number of records at } t_R}{\text{number of records in training set}} \\ P(j \mid t_L) = \dfrac{\text{number of class } j \text{ records at } t_L}{\text{number of records at } t} \\ P(j \mid t_R) = \dfrac{\text{number of class } j \text{ records at } t_R}{\text{number of records at } t} \end{cases}$$

- Optimality measure maximizes split over all possible splits, at node *t*

Classification and Regression Trees (CART)

- Example
  - Predict whether customer is classified “Good” or “Bad” credit risk using three predictor fields

Customer	Savings	Assets	Income ($1000s)	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	High	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	High	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

  - Splits evaluated for *Savings*, *Assets*, and *Income*
  - *Income* is numeric.
    - CART identifies possible splits based on values it contains

Classification and Regression Trees (CART)

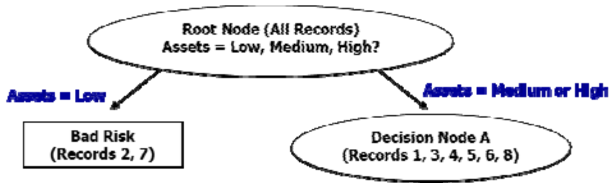
– Nine candidate splits, for t = root node

Candidate Split	Left Child Node, t_L	Right Child Node, t_R
1	*Savings* = *low*	*Savings* ∈ {*medium*, *high*}
2	*Savings* = *medium*	*Savings* ∈ {*low*, *high*}
3	*Savings* = *high*	*Savings* ∈ {*low*, *medium*}
4	*Assets* = *low*	*Assets* ∈ {*medium*, *high*}
5	*Assets* = *medium*	*Assets* ∈ {*low*, *high*}
6	*Assets* = *high*	*Assets* ∈ {*low*, *medium*}
7	*Income* ≤ \$25,000	*Income* > \$25,000
8	*Income* ≤ \$50,000	*Income* > \$50,000
9	*Income* ≤ \$75,000	*Income* > \$75,000

Classification and Regression Trees (CART)

Split	P_L	P_R	$P(j\|t_L)$	$P(j\|t_R)$	$2P_LP_R$	$Q(s\|t)$	$\Phi(s\|t)$
1	0.375	0.625	G: .333 B: .667	G: .8 B: .2	0.46875	0.934	0.4378
2	0.375	0.625	G: 1 B: 0	G: 0.4 B: 0.6	0.46875	1.2	0.5625
3	0.25	0.75	G: 0.5 B: 0.5	G: 0.0667 B: 0.333	0.375	0.334	0.1253
4	0.25	0.75	G: 0 B: 1	G: 0.833 B: 0.167	0.375	1.667	**0.6248**
5	0.5	0.5	G: 0.75 B: 0.25	G: 0.5 B: 0.5	0.5	0.5	0.25
6	0.25	0.75	G: 1 B: 0	G: 0.5 B: 0.5	0.375	1	0.375
7	0.375	0.625	G: 0.333 B: 0.667	G: 0.8 B: 0.2	0.46875	0.934	0.4378
8	0.625	0.375	G: 0.4 B: 0.6	G: 1 B: 0	0.46875	1.2	0.5625
9	0.875	0.125	G: 0.571 B: 0.429	G: 1 B: 0	0.21875	0.858	0.1877

Classification and Regression Trees (CART)



- Optimality measure maximized to 0.6248, when *Assets* = "Low" (Left branch), *Assets* = "Medium or High" (Right branch)
- Left branch terminates to pure leaf node; both records have target value = "Bad Risk"
- Right branch diverse and calls for further partitioning

Classification and Regression Trees (CART)

- ***When is Optimality Measure large?***
  - Measure large when its main components are large

(1) Larger values of $\Phi(s\,|\,t)$ tend to be associated with larger values of its main components : $2P_L P_R$ and $\sum_{j=1}^{\#classes} \left| P(j\,|\,t_L) - P(j\,|\,t_R) \right|$

- *When is component Q(s|t) large?*

(2) Let $Q(s\,|\,t) = \sum_{j=1}^{\#classes} \left| P(j\,|\,t_L) - P(j\,|\,t_R) \right|$

$Q(s\,|\,t)$ is large when the distance between $P(j\,|\,t_L)$ and $P(j\,|\,t_R)$ is maximized across each class value

Classification and Regression Trees (CART)

- ***Classification Error Rate***
  - After tree “fully grown”, not all leaf nodes necessarily homogenous. Some are diverse leaf nodes
  - Leads to level of classification error

Customer	Savings	Assets	Income	Credit Risk
004	High	Low	<=$30K	**Good**
009	High	Low	< $30K	**Good**
027	High	Low	<=$30K	**Bad**
031	High	Low	< $30K	**Bad**
104	High	Low	<=$30K	**Bad**

  - Probability that record in leaf node classified correctly (as “Bad”) is 3/5 = 0.60 = 60%
  - Classification error rate for leaf node is 0.40 = 40%. 2/5 “Good” records classified incorrectly (“Bad”)

C4.5 Algorithm

- C4.5 is extension of ID3 developed by Quinlan
  - Similar to CART, C4.5 builds tree by recursively visiting decision nodes and choosing optimal spit, until no further splits possible
- ***Differences Between CART and C4.5***
  - Unlike CART, C4.5 is not limited to binary splits and produces tree with variable shape
  - C4.5 produces branch for each categorical value. This may result in “bushiness”
  - C4.5 uses different algorithm to measure homogeneity occurring at leaf nodes
  - C4.5 uses information gain or entropy reduction to select optimal split at each decision node
  - In Engineering, information analogous to signal, entropy analogous to noise

C4.5 Algorithm

- Entropy

$$H(X) = -\sum_j p_j \log_2(p_j)$$

  - Event with probability = p, average amount of information, in bits, required to transmit result $-\log_2(p)$
  - Represents smallest number of bits, on average per symbol, needed to transmit stream of symbols corresponding to observed values of X

C4.5 Algorithm

- How to use Entropy?
  - Assume candidate split S, partitions data set T into subsets $T_1, T_2, \ldots, T_k$
  - Mean information requirement calculated as weighted sum of entropies associated with each subset T

$$H_S(T) = \sum_{i=1}^{k} P_i H_S(T_i)$$

  - P_i represents proportion of records in subset T_i

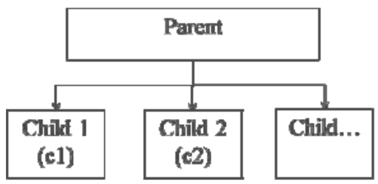
C4.5 Algorithm

- Information gain

gain(S) = H(T) – Hs(T)

  - Represents increase in information by partitioning training data *T* according to candidate split *S*
  - For each candidate split, C4.5 chooses split that has maximum information gain, gain(S)

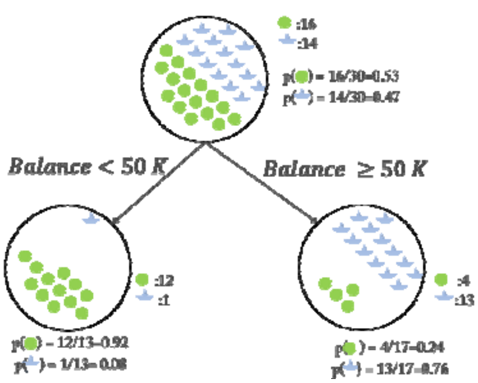
- Information gain = entropy(parent)–[p(c1)×entropy(c1)+p(c2)×entropy(c2) +...]



Note: Higher IG indicates a more informative split by the variable.

C4.5 Algorithm

- Information gain



Entropy(parent)

= −[p(•)×log2 p(•) + p(☆)×log2 p(☆)]

= −[0.53 ×(−0.9) +0.47 ×(−1.1)]

= 0.99 (very impure!)

Left child: entropy(Balance < 50K)

= −[p(•) log2 p(•) + p(☆) log2 p(☆)]

= −[0.92 × (−0.12) + 0.08 × (−3.7)]

= 0.39

Right child: entropy(Balance ≥50K)

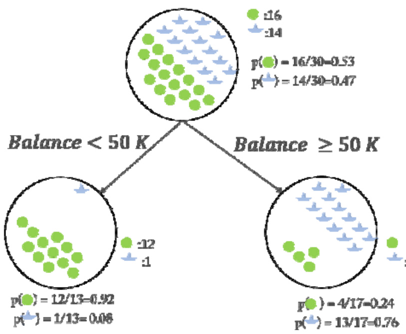
= −[p(•) log2 p(•) + p(☆) log2 p(☆)]

= −[0.24 × (−2.1) + 0.76 × (−0.39)]

= 0.79

C4.5 Algorithm

- Information gain



Entropy(parent)

= 0.99

Left child: entropy(Balance < 50K)

= 0.39

Right child: entropy(Balance ≥50K)

= 0.79

IG for the split based on "balance" variable :

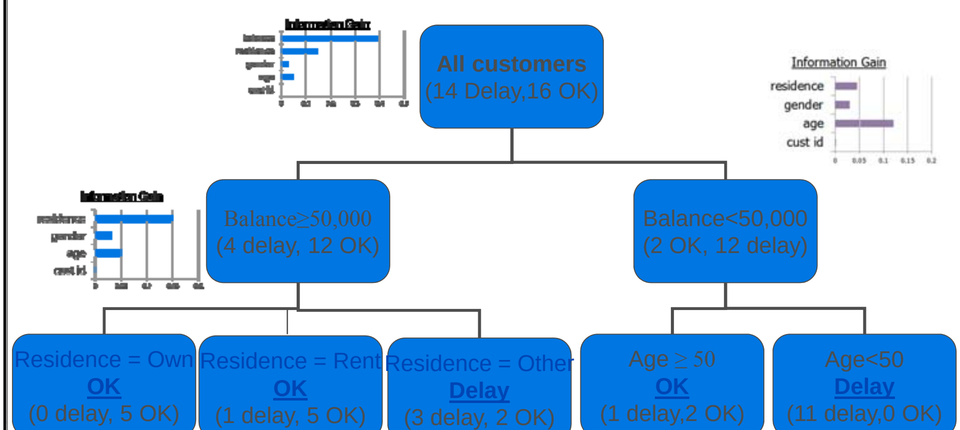
IG = entropy(parent) – [p(Balance < 50K) × entropy(Balance < 50K)

+p(Balance ≥ 50K) × entropy(Balance ≥ 50K)]

= 0.99 – [0.43 × 0.39 + 0.57 × 0.79]

= **0.37**

Tree Structure



Exercise

Customer	Savings	Assets	Income ($1000s)	Credit Risk
1	Medium	High	75	**Good**
2	Low	Low	50	**Bad**
3	High	Medium	25	**Bad**
4	Medium	Medium	50	**Good**
5	Low	Medium	100	**Good**
6	High	High	25	**Good**
7	Low	Low	25	**Bad**
8	Medium	Medium	75	**Good**

Candidate Split	Child Nodes		
1	*Savings = Low*	*Savings = Medium*	*Savings = High*
2	*Assets = Low*	*Assets = Medium*	*Assets = High*
3	*Income <= $25,000*	*Income > $25,000*	
4	*Income <= $50,000*	*Income > $50,000*	
5	*Income <= $75,000*	*Income > $75,000*	

Exercise

Candidate Split	Child Nodes	Information Gain (Entropy Reduction)
1	*Savings = Low* *Savings = Medium* *Savings = High*	0.36 bits
2	*Assets Low* *Assets Medium* *Assets High*	**0.5487 bits**
3	*Income* <= $25,000 *Income* > $25,000	0.1588 bits
4	*Income* < $50,000 *Income* > $50,000	0.3475 bits
5	*Income* <= $75,000 *Income* > $75,000	0.0923 bits

Decision Rules

- Decision Trees produce interpretable output in human-readable form
- Decision Rules constructed directly from Decision Tree output, traversing path from root node to a given leaf node
- Decision Rules have form IF antecedent THEN consequent
- Antecedent consists of attributes values from branches of given path
- Consequent is classification of records contained in particular leaf node, corresponding to path