Knowledge Discovery and Data Mining

Unit # 4

Acknowledgement

- Most of the slides in this presentation are taken from course slides provided by
  - Han and Kimber (Data Mining Concepts and Techniques) and
  - Tan, Steinbach and Kumar (Introduction to Data Mining)

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity

Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  - Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = $1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$

C1	1
C2	5

P(C1) = 1/6 P(C2) = 5/6

Gini = $1 - (1/6)^2 - (5/6)^2 = 0.278$

C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6

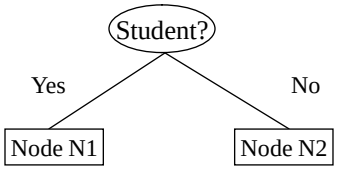
Gini = $1 - (2/6)^2 - (4/6)^2 = 0.444$

Classification: Motivation

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



Gini(N1)
= 1 – (6/7)² – (1/7)²
= 0.24

Gini(N2)
= 1 – (3/7)² – (4/7)²
= 0.49

Gini(Student)
= 7/14 * 0.24 +
7/14 * 0.49
= ??

GINI Index for Buy Computer Example

- Gini (Income):
- Gini (Credit_Rating):
- Gini (Age):

Alternative Splitting Criteria based on Entropy

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

– Measures homogeneity of a node.
  - Maximum (log n_c) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information

– Entropy based computations are similar to the GINI index computations

– Hint: $\log_2 p = \ln p / \ln(2)$

Entropy in a nut-shell

Low Entropy

High Entropy

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

- Measures misclassification error made by a node.
  - Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

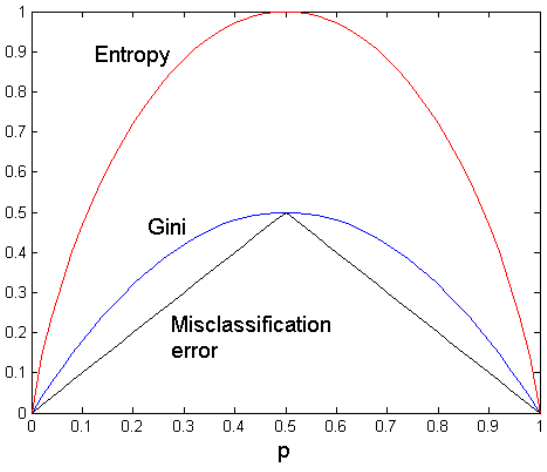
C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

For a 2-class problem:



Example

Attribute 1	Attribute 2	Attribute 3	Class
A	70	T	C1
A	90	T	C2
A	85	F	C2
A	95	F	C2
A	70	F	C1
B	90	T	C1
B	78	F	C1
B	65	T	C1
B	75	F	C1
C	80	T	C2
C	70	T	C2
C	80	F	C1
C	80	F	C1
C	96	F	C1

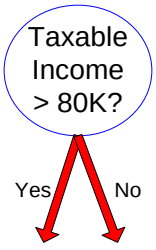
Example II

Height	Hair	Eyes	Class
Short	Blond	Blue	+
Tall	Blond	Brown	-
Tall	Red	Blue	+
Short	Dark	Blue	-
Tall	Dark	Blue	-
Tall	Blond	Blue	+
Tall	Dark	Brown	-
Short	Blond	Brown	-

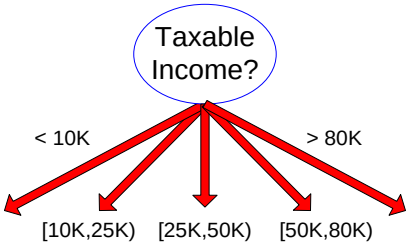
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be computationally intensive

Splitting Based on Continuous Attributes (Cont'd)



(i) Binary split



(ii) Multi-way split

Continuous Attributes: Computing GINI Index

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	*0.300*	0.343	0.375	0.400	0.420

Categorical Attributes: Computing GINI Index

- From a historical perspective, Gini Index always created a binary tree.
- As a result, in case of multiple values, it merged them together to find the best binary split
- For each distinct value, gather counts for each class in the dataset

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Handling of Multi-state Variable

- The way both Gini Index and Entropy are presented, they become biased to variables having multiple states.
- To over this, the following approach was recommended (in C4.5 using Entropy but can be generalized to Gini Index as well).
  - Gain = SR (D) – SR_A (D)
  - Where SR = splitting rule metric
  - D = class variable
  - A = an attribute on which the splitting rule is conditioned

Buy Computer Example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

SplitInfo

- Gini (buy) = 0.46
  - Gini $_{Age}$ (buy) = 0.34 : Gain = 0.12
  - Gini $_{inc}$ (buy) = 0.44 : Gain = 0.02
  - Gini $_{std}$ (buy) = 0.37 : Gain = 0.09
  - Gini $_{rat}$ (buy) = 0.43 : Gain = 0.03
- SplitInfo = unconditional splitting rules on the variables. If one is using Gini then it becomes
  - Splitinfo (age) = Gini (age) = 0.66
  - Splitinfo (inc) = Gini (inc) = 0.65
  - Splitinfo (std) = Gini (std) = 0.5
  - Splitinfo (rat) = Gini (rat) = 0.49

Gain_ratio

- To obtain gain ratio, we divide gain by splitinfo
  - Gain_ratio (age) = 0.12 / 0.66 = 0.18 (0.175)
  - Gain_ratio (inc) = 0.02 / 0.65 = 0.03
  - Gain_ratio (std) = 0.09 / 0.5 = 0.18 (0.184)
  - Gain_ratio (rat) = 0.03 / 0.49 = 0.06
- A similar computation would have been done if we were using Entropy or even Miclassification Error

Inducing a decision tree

- There are many possible trees
- How to find the most compact one
  - that is consistent with the data?
- The *key* to building a decision tree - which attribute to choose in order to branch.
- The *heuristic* is to choose the attribute with the minimum GINI/Entropy.

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive manner
  - At start, all the training examples are at the root
  - Attributes are categorical
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., GINI/Entropy)
- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no examples left

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction. The leaf node holds the class prediction
- Rules are easier for humans to understand
- Example

IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"
IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"
IF *age* = "31…40" THEN *buys_computer* = "*yes*"
IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"
IF *age* = "<=30" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

Background

- ID3 (Iterative Dichotomiser 3)
  - published in 1986 but proposed in 1983
  - Only works on non-continuous (discrete) attributes
  - Uses Information Gain/Entropy as the splitting rule
- CART
  - Published in 1984
  - Uses Gini Index as the splitting rule
  - Binary trees
- C4.5
  - Extension of ID3 and published in 1993
  - Works on continuous attributes
  - Uses modified Gain/Entropy metric as the splitting rule to defy advantage to variables having multiple states

Characteristics of Decision Tree Induction

- Decision tree induction is a non-parametric approach for building classification models. In other words, it doesn't require any prior assumptions regarding the type of probability distributions satisfied by the class and other attributes.
- Finding an optimal decision tree is an NP-complete problem. Many decision tree algorithms employ a heuristic-based approach to guide their search in the vast hypothesis space. For example, the algorithm discussed in this unit uses a greedy, top-down, recursive partitioning strategy for growing a decision tree.

Characteristics of Decision Tree Induction (Cont'd)

- Techniques developed for constructing decision trees are computationally inexpensive, making it possible to quickly construct models even when the training set size is very large. Furthermore, once a decision tree has been built, classifying a test record is extremely fast, with a worst-case complexity of O(w), where w is the maximum depth of the tree.
- Decision tree, specially smaller-sized trees, are relatively easy to interpret.
- Decision tree algorithms are quite robust to the presence of noise.

Characteristics of Decision Tree Induction (Cont'd)

- The presence of redundant attributes does not adversely affect the accuracy of decision trees. An attribute is redundant if it is strongly correlated with another attribute in the data. One of the two redundant attributes will not be used for splitting once the other attribute has been chosen.
- Studies have shown that the choice of impurity measures has little effect on the performance of decision tree induction algorithms.

Advantages of Decision Tree Based Classification

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

Feature Discetization

- Unsupervised Discretization
  - Used in Clustering
- Supervised Discretization
  - Used in Classification

Unsupervised Feature Discretization Techniques

- The task of feature discretization techniques is to discretize the values of continuous features into a small number of intervals, where each interval is mapped to a discrete symbol.
- Suppose the set of values for a given feature are {3, 2, 1, 5, 4, 3, 1, 7, 5, 3}. After sorting, these values can be placed into three bins
  - {1, 1, 2, 3, 3, 3, 4, 5, 5, 7}

Value Reduction

- One of the main problems of the previous method is to find the best cutoffs for bins.
- The value-reduction problem can be stated as an optimization problem in the selection of k bins: given the number of bins k, distribute the values in the bins to minimize the average distance of a value from its bin mean or median.
- The distance is usually measured as the squared distance for a bin mean and as the absolute distance for a bin median.

Value Reduction – A Heuristic Algorithm

- Sort all values for a given feature.
- Assign approximately equal number of sorted adjacent values (vi) to each bin, where the number of bins is given in advance.
- Move a border element vi from one bin to the next (or previous) when that reduces the global distance error (ER) (the sum of all distances from each vi to the mean or mode of its assigned bin).

Working of the Algorithm

- The set of values for a feature f is {5, 1, 8, 2, 2, 9, 2, 1, 8, 6}.
- Split them into three bins (k = 3), where the bins will be represented by their modes.
- Initial bins are {1, 1, 2 2, 2, 5 6, 8, 8, 9}
- Modes for the three bins are {1, 2, 8}. The error, ER, is 0+0+1+0+0+3+2+0+0+1=7
- After moving two elements from BIN2 into BIN1 and one element from BIN3 to BIN2 in the next three iterations, the final distribution of elements are {1, 1, 2, 2, 2, 5, 6, 8, 8, 9}
- The total minimized error, ER, is 4.

Value Reduction Exercise

- Perform Bin-based values reduction with the best cutoffs for the following:
  - The feature Attribute 2 (in slide # 18, Unit # 4) using mean values as representatives for two bins.
  - Repeat the same exercise for three bins