COURSE NAME:
DATA WAREHOUSING & DATA MINING

LECTURE 17
TOPICS TO BE COVERED:

- Decision tree

CLASSIFICATION BY DECISION TREE INDUCTION

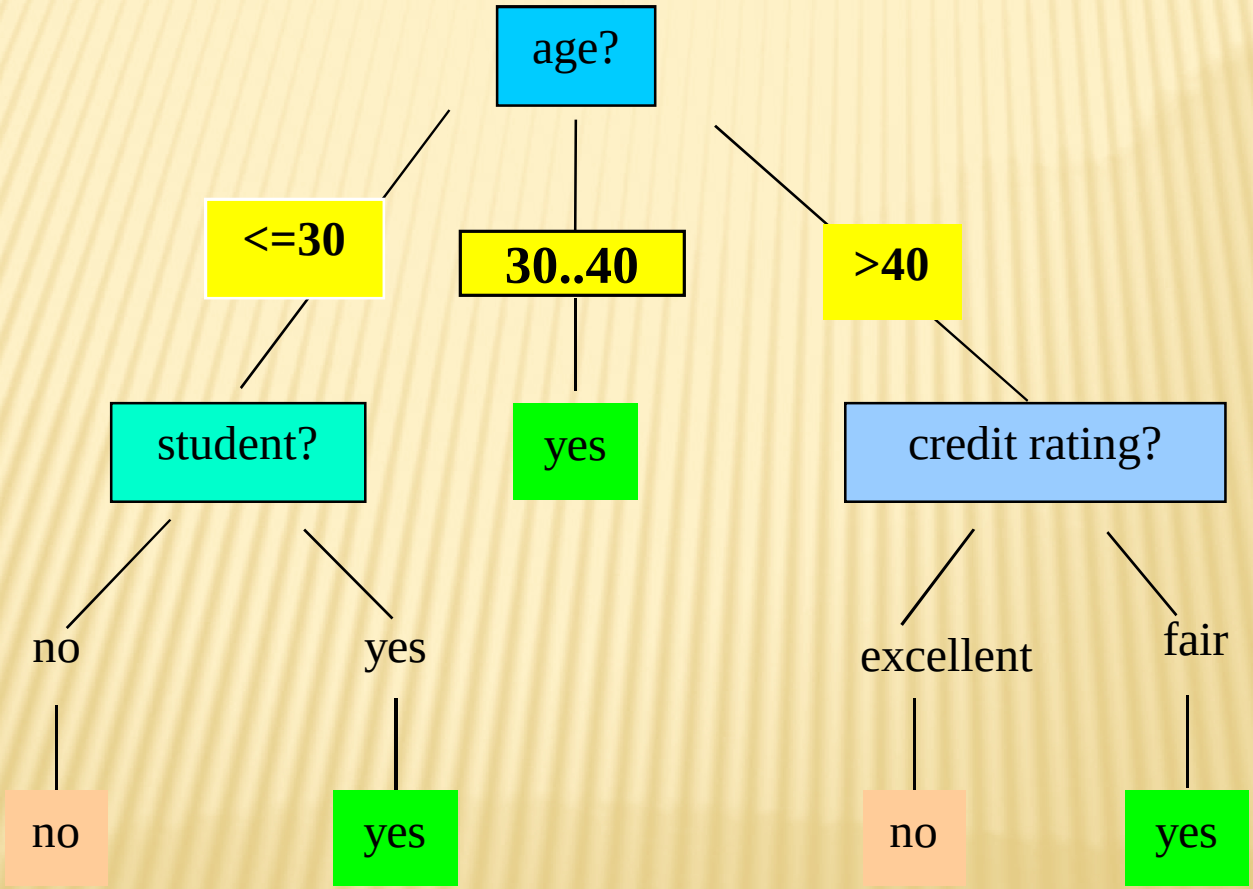
- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

TRAINING DATASET

This follows an example from Quinlan's ID3

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

OUTPUT: A DECISION TREE FOR *"BUYS_COMPUTER"*



DECISION TREE INDUCTION

During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser).

This work expanded on earlier work on concept learning systems, described by E. B. Hunt, J. Marin, and P. T. Stone. Quinlan later presented C4.5 (a successor of ID3), which became a benchmark to which newer supervised learning algorithms are often compared.

In 1984, a group of statisticians (L. Breiman, J. Friedman, R. Olshen, and C. Stone) published the book Classification and Regression Trees (CART), which described the generation of binary decision trees.

These two cornerstone algorithms spawned a flurry of work on decision tree induction.

DECISION TREE INDUCTION

Algorithms for decision tree induction also follow such a top-down approach, which starts with a training set of tuples and their associated class labels. The training set is recursively partitioned into smaller subsets as the tree is being built.

ALGORITHM FOR DECISION TREE INDUCTION

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples 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 samples left

ATTRIBUTE SELECTION MEASURE

- Three popular attribute selection measures—information gain, gain ratio, and gini index.

- The notation used herein is as follows.

  Let D, the data partition, be a training set of class-labeled tuples. Suppose the class label attribute has m distinct values defining m distinct classes, C_i (for i = 1, …, m).

  Let $C_{i,D}$ be the set of tuples of class C_i in D. Let |D| and $|C_{i,D}|$ denote the number of tuples in D and $C_{i,D}$, respectively.

INFORMATION GAIN

- ID3 uses information gain as its attribute selection measure.
- Let node *N represent or hold the tuples of partition D. The attribute with the highest information gain is chosen as the splitting attribute for* node *N. This attribute minimizes the information needed to classify the tuples in the* resulting partitions and reflects the least randomness or "impurity" in these partitions. Such an approach minimizes the expected number of tests needed to classify a given tuple and guarantees that a simple (but not necessarily the simplest) tree is found. The expected information needed to classify a tuple in *D is given by*

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

INFORMATION GAIN

- where p_i *is the probability that an arbitrary tuple in D belongs to class* C_i *and is estimated* by $|C_{i,D}| / |D|$.
- *A log function to the base 2 is used, because the information is encoded in* bits.
- *Info(D) is just the average amount of information needed to identify the class label* of a tuple in *D.*
- *Note that, at this point, the information we have is based solely on the* proportions of tuples of each class. *Info(D) is also known as the entropy of D.*

INFORMATION GAIN

- Now, suppose we were to partition the tuples in *D on some attribute A having v distinct* values, $\{a_1, a_2, \ldots, a_v\}$, *as observed from the training data.*
- *If A is discrete-valued,* these values correspond directly to the *v outcomes of a test on A. Attribute A can be used* to split *D into v partitions or subsets,* $\{D_1, D_2, \ldots, D_v\}$, *where* D_j *contains those tuples in D that have outcome* a_j *of A.*
- *These partitions would correspond to the branches grown* from node *N. Ideally, we would like this partitioning to produce an exact classification* of the tuples. That is, we would like for each partition to be pure. However, it is quite likely that the partitions will be impure (e.g., where a partition may contain a collection of tuples from different classes rather than from a single class). How much more information would we still need (after the partitioning) in order to arrive at an exact classification?

INFORMATION GAIN

- This amount is measured by

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

- The term $|D_j|/|D|$ *acts as the weight of the jth partition.*
- *$Info_A(D)$ is the expected information* required to classify a tuple from *D based on the partitioning by A.*
- *The smaller the* expected information (still) required, the greater the purity of the partitions.

INFORMATION GAIN

- Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on *A). That is,*

$$Gain(A) = Info(D) - Info_A(D).$$

INFORMATION GAIN

- In other words, *Gain(A) tells us how much would be gained by branching on A. It is the* expected reduction in the information requirement caused by knowing the value of *A.*
- The attribute *A with the highest information gain, (Gain(A)), is chosen as the splitting* attribute at node *N. This is equivalent to saying that we want to partition on the attribute A that would do the "best classification," so that the amount of information still required* to finish classifying the tuples is minimal (i.e., minimum $Info_A(D)$).

EXAMPLE

Class-labeled training tuples from the *AllElectronics* customer database.

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

EXAMPLE

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

the expected information needed to classify a tuple in D if the tuples are partitioned according to *age* is

$$\begin{aligned} Info_{age}(D) = &\frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ &+\frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) \\ &+\frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ = &\ 0.694 \text{ bits.} \end{aligned}$$

Hence, the gain in information from such a partitioning would be

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

GAIN RATIO

- The gain ratio is defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}.$$

- Where SplittingInfo(A) is describe as
- It applies a kind of normalization to information gain using a “split information” value defined analogously with *Info(D) as*

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right).$$

- This value represents the potential information generated by splitting the training data set, *D, into v partitions, corresponding to the v outcomes of a test on attribute A.*

EXAMPLE

- Computation of gain ratio for the attribute *income. A test on income splits the data of previous example* into three partitions, namely *low, medium, and high, containing four, six, and*
- four tuples, respectively.

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right).$$

$$= 0.926.$$

we have *Gain(income) = 0.029. Therefore, GainRatio(income)* = 0.029/0.926 = 0.031.

GINI INDEX

- the Gini index measures the impurity of *D, a data partition or set of training tuples, as*

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

- where p_i *is the probability that a tuple in D belongs to class* C_i *and is estimated by* $|C_{i,D}| / |D|$. *The sum is computed over m classes.*

- The Gini index considers a binary split for each attribute. Let's first consider the case where *A is a discrete-valued attribute having v distinct values,* $\{a_1, a_2, \ldots, a_v\}$, *occurring* in *D*

- if a binary split on *A partitions D into D_1 and D_2, the* gini index of *D given that partitioning is*

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2).$$

- The reduction in impurity that would be incurred by a binary split on a discrete- or continuous-valued attribute *A is*

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

- The attribute t equivalently, has the minimum Gini index) is selected as the splitting attribute.

EXAMPLE

- Induction of a decision tree using gini index. Let *D be the training data of previous example where* there are nine tuples belonging to the class *buys_computer = yes and the remaining five* tuples belong to the class *buys_computer = no. A (root) node N is created for the tuples* in D.
- *Gini index to compute the Impurity of D:*

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

- To find the splitting criterion for the tuples in *D, we need to compute the gini index* for each attribute. Let's start with the attribute *income and consider each of the possible* splitting subsets. Consider the subset {*low, medium}.This would result in 10 tuples in*

  partition D_1 *satisfying the condition "income* $\in$*{low, medium}." The remaining four* tuples of *D would be assigned to partition* D_2*. The Gini index value computed based on* this partitioning is

$$
\begin{aligned}
&Gini_{income \in \{low,medium\}}(D) \\
&\quad = \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\
&\quad = \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\
&\quad = 0.450 \\
&\quad = Gini_{income \in \{high\}}(D).
\end{aligned}
$$