

Data Mining

– Classification Algorithms (II)



In last lecture

- General procedure for constructing a classification model (classifier).
 - Partition the data into Training and Testing.
 - Train the classifier.
 - Test the classifier before using it.
 - Accuracy is not appropriate
 - Precision, Recall, F-Measure
- Combine the decisions of different classifiers
 - Majority vote
 - Linear Weight Combination



Major Classification Algorithms

- ~~Nearest Neighbor Algorithm~~
- ~~Bayesian Classification~~
- Classification by decision tree induction

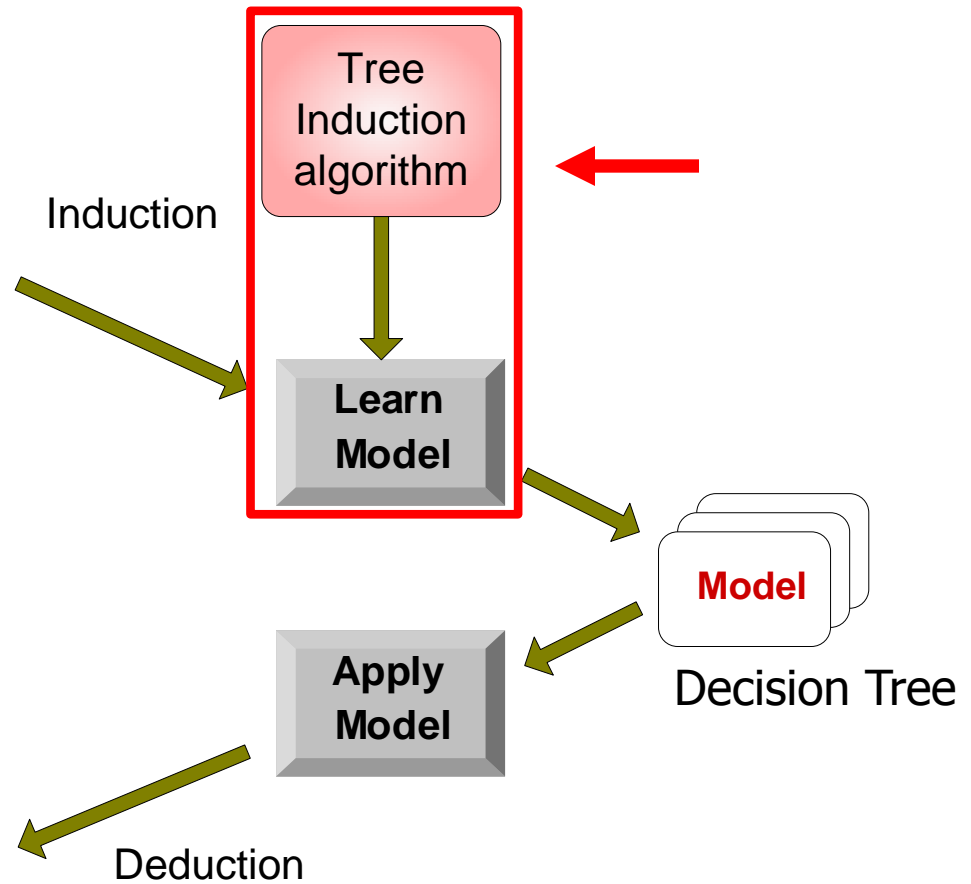
Decision Tree Classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set





What is Decision Tree?

- Flow-chart-like tree structure
 - Internal node: a test on an attribute
 - Age < 30?
 - Leaf node: classes
 - Buys computer: yes or no?

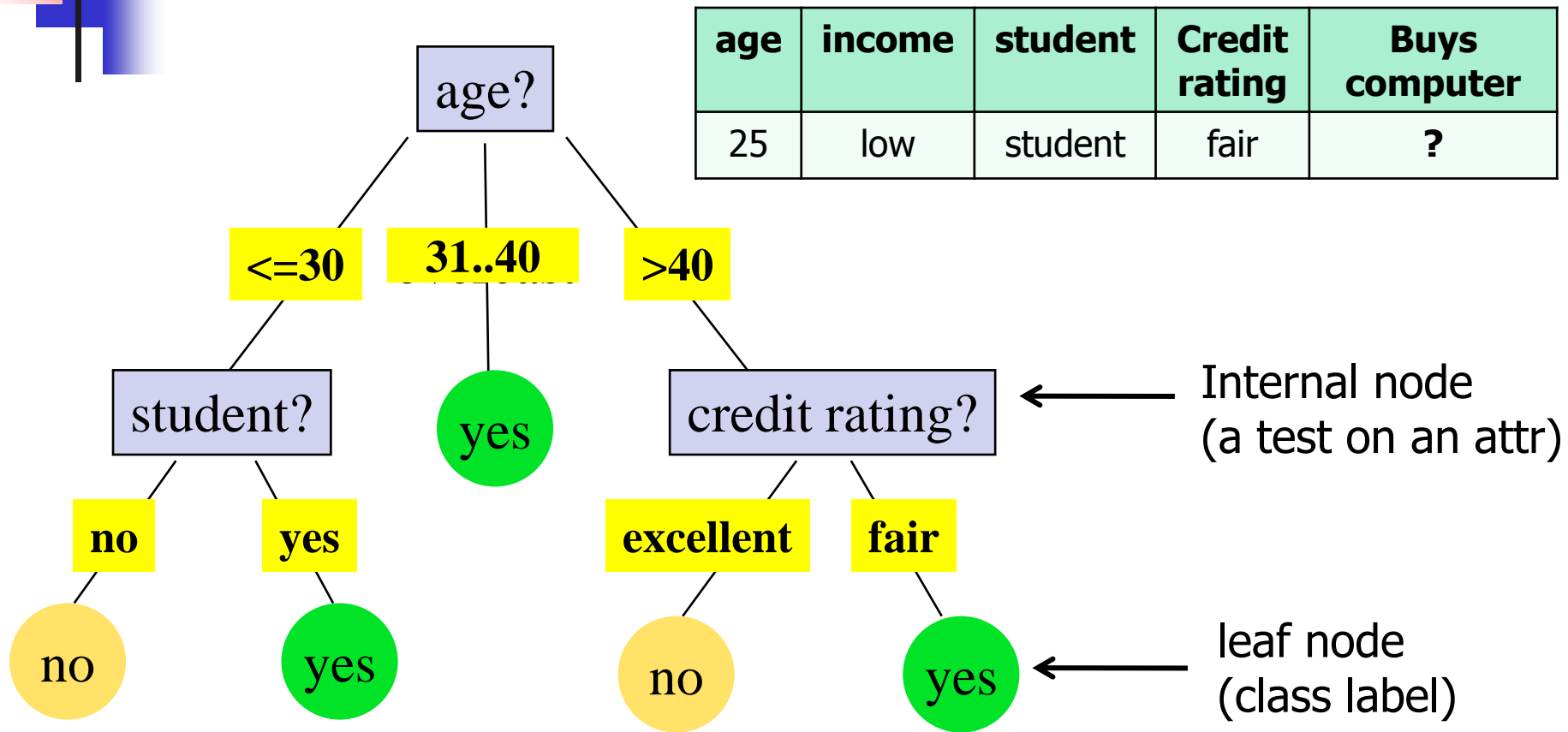


An Example

Whether a customer is likely to purchase a computer

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

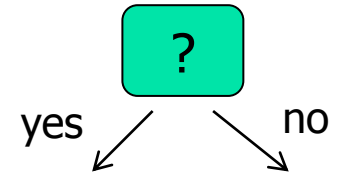
A Decision Tree for "Buys Computer"



Decision Tree Induction

- Many algorithms

- Hunt's Algorithm (one of the earliest)
- CART
- **ID3(Iterative Dichotomiser 3), C4.5**
- SLIQ,SPRINT



- General structure

- Attribute selection by **Information Entropy**
- Decision tree induction
- Rule generation



A Simple Thought on Decision Tree

- A simple minded algorithm:

If **Age = a** and **Income = b** and **Student = c**
and **Credit Rating = d**
then **buys_comuter = ?**

- This structure will yield a branch for each row, not eliminating superfluous attributes, and is unnecessarily complex.
- The complete classification space for m attributes is of size: $\prod_{j=1}^m N_j$
(N_j is the number of values of attribute j)



An Observation on Rule Generation

- Rules are based solely upon the input examples.
- Rules should be able to predict results for **other cases**.
- If prediction is not achieved, then the rules must be alterable either automatically or editorially.



The Problem Statement

- Given a set of data described by **a set of attributes** and **an outcome class**, the problem is to find a **minimum** decision tree that will classify the values of the class based on the values of given attributes.



Building the Decision Tree

- Collect examples to build **a training set**.
- Decide the **class**.
- Select one of the attributes to be the **starting point** or root node of the tree.
- Split up the training set into a number of smaller tables, each containing examples with the **same value** of the selected attribute.
- If the values in the class is partitioned, then the process is complete. Otherwise, select a **new attribute** and split the set again.



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 entropy**)
- 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

An Example:

find a means of predicting which **company profiles** will lead to a increase or decrease in profits based on the following data

attributes

class

Profit	Age	Competition	Type
Down	Old	No	Software
Down	Midlife	Yes	Software
Up	Midlife	No	Hardware
Down	Old	No	Hardware
Up	New	No	Hardware
Up	New	No	Software
Up	Midlife	No	Software
Up	New	Yes	Software
Down	Midlife	Yes	Hardware
Down	Old	Yes	Software

Let's have a go ...

For the Company Profile example we perform a split on attribute **Age**:

		Profit	Age	Competition	Type
Age	old	down	old	no	software
		down	old	no	hardware
		down	old	yes	software
	new	up	new	no	hardware
		up	new	no	software
		up	new	yes	software
	midlife	down	midlife	yes	software
		up	midlife	no	hardware
		up	midlife	no	software
		down	midlife	yes	hardware

The Training Set Becomes smaller ...

After the Second Split on Attribute Competition

	Profit		Profit	Competition	Type
Age	old → down	→		no	software
				no	hardware
				yes	software
	new → up			no	hardware
				no	software
				yes	software
middle life →	competition →	no	up	no	hardware
			up	no	software
		yes	down	yes	hardware
			down	yes	software

The Rules Derived from the Tree

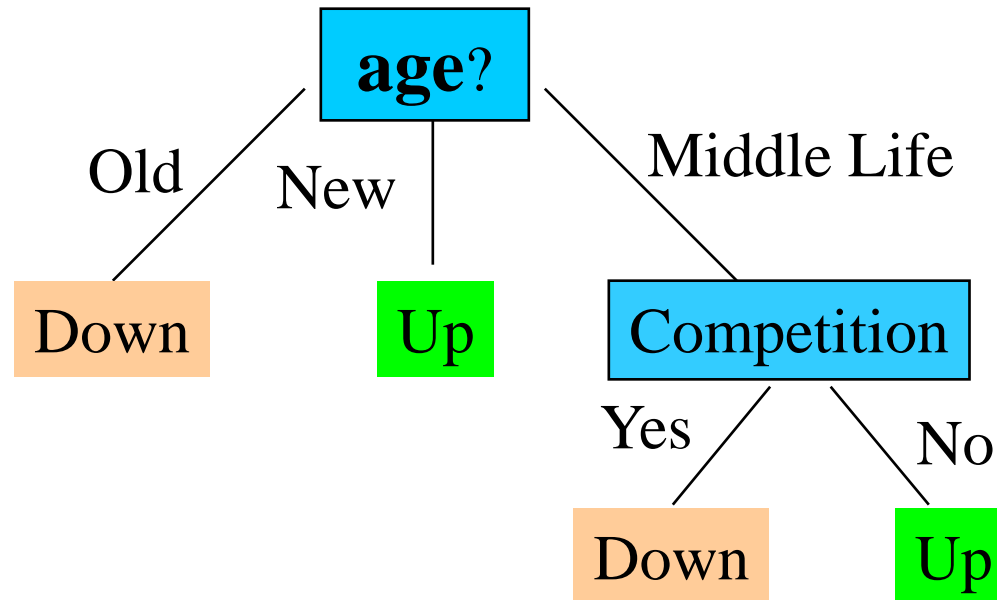
	IF	THEN
Rule1	age is old	profit is down
Rule2	age is new	profit is up
Rule3	age is midlife AND competition is no	profit is up
Rule4	age is midlife AND competition is yes	profit is down

Can the original training set be re-generated base of this set of rules?

Can this set of rules be used to predict all cases in the problem domain (i.e., the PROFIT prediction)?

Output: A Decision Tree for *“Company profiles”*

Profit=up ? down





We know how to create a tree now, but we still need to know where to start...

- Which attribute AGE, TYPE, or COMPETITION is **most strongly** associated with the class PROFIT?
- Note that:
 - PROFIT has 2 states: UP or DOWN
 - COMPETITION has 2 states: NO or YES
 - TYPE has 2 states: SOFTWARE or HARDWARE
 - AGE has 3 states: OLD, NEW or MIDLIFE

So, we need to differentiate the “predictability” of the attributes towards the class.

Competition	Profit
no	down
no	up
no	down
no	up
no	up
no	up
<hr/>	
yes	down
yes	up
yes	down
yes	down

Is there any indication between the values of Competition and Values of Profit? No, not at all.

$$P(\text{Profit} = \text{up} \mid \text{Comp} = \text{No}) = 4/6 = .67$$

$$P(\text{Profit} = \text{down} \mid \text{Comp} = \text{No}) = 2/6 = .33$$

$$P(\text{Profit} = \text{up} \mid \text{Comp} = \text{Yes}) = 1/4 = .25$$

$$P(\text{Profit} = \text{down} \mid \text{Comp} = \text{Yes}) = 3/4 = .75$$

So what?

It has to be compared with the other attributes.



The Conditional Probability seems to be a right tool to show the relations between the attribute and the class.

TYPE:

$$P(\text{Profit} = \text{up} \mid \text{Type} = \text{Software}) = 0.5$$

$$P(\text{Profit} = \text{down} \mid \text{Type} = \text{Software}) = 0.5$$

$$P(\text{Profit} = \text{up} \mid \text{Type} = \text{Hardware}) = 0.5$$

$$P(\text{Profit} = \text{down} \mid \text{Type} = \text{Hardware}) = 0.5$$

AGE:

$$P(\text{Profit} = \text{up} \mid \text{Age} = \text{Old}) = 0.0$$

$$P(\text{Profit} = \text{down} \mid \text{Age} = \text{Old}) = 1.0$$

$$P(\text{Profit} = \text{up} \mid \text{Age} = \text{New}) = 1.0$$

$$P(\text{Profit} = \text{down} \mid \text{Age} = \text{New}) = 0.0$$

$$P(\text{Profit} = \text{up} \mid \text{Age} = \text{Midlife}) = 0.5$$

$$P(\text{Profit} = \text{Down} \mid \text{Age} = \text{Midlife}) = 0.5$$

When Age = Old, the probability of Profit =up is zero.



The problem:

How do we select an attribute?

- We need to find a formula to determine the most significant attribute in a calculation of the probabilities of the attributes
- How can this be formalized? How can this be extended to deeper levels of the decision tree?
- Need some predictive statistic based on conditional probability that can be applied to sort out the 'best' attribute at each level.

Amongst three attributes: AGE, COMPETITION, and TYPE, which attribute carries “more” information than others?



Can Information be Measured?

- Why should information be measurable?
 - Information Theory
 - Information retrieval, coding, and processing techniques
- If it is measurable what can we do about it?
 - Applicability in data mining
 - Quantification of the factors that affect our decisions (to order the importance of different messages that we receive).



Information Theory

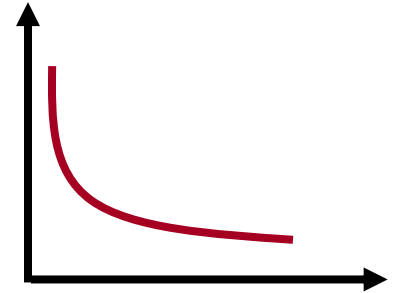
- How much information a message contains depends on the extent to which it resolves uncertainty.
- The more certain a message is, the less information it conveys (i.e., the amount of information increases as the probability of the message decreases, they are inversely related).
- The greater the number of possible messages, the greater the amount of information conveyed (additivity of accumulated information)

Information Measuring

■ Self-information

$$\log\left(\frac{x}{y}\right) = \log(x) - \log(y)$$
$$\log 1 = 0$$

$$I(M) = \log\left(\frac{1}{p(M)}\right) = -\log(p(M))$$



- If we use \log_2 , the unit of measuring is “bit”.
- Inversity between Information and Uncertainty
 - infrequently occurring messages contain more information than more frequently occurring messages
 - a certain message, i.e. of probability 1, has an information measure of zero



Information Measuring

■ Information additivity

$$\begin{aligned} I(m_1) + I(m_2) &= \log\left(\frac{1}{p(m_1)}\right) + \log\left(\frac{1}{p(m_2)}\right) = -\log(p(m_1)) - \log(p(m_2)) \\ &= -\log(p(m_1) \times p(m_2)) = \log\left(\frac{1}{p(m_1) \times p(m_2)}\right) \\ &= I(m_1 + m_2) \end{aligned}$$

- a compound message of two (or more) unrelated (or mutually independent) messages would have a quantity of information that is the sum of the measures of information of each message individually
- In probability theory, the total probability of two events are the multiplication of two probabilities.



An Example of Information Bit

- In the simplest case where one of two equally probable messages is selected, each with a probability of 50%, the quantity of information is $\log(1/0.5)$, or $\log_2 2$.
- The log of 2 to the base 2 is 1 (i.e., $\log_2 2 = 1$). Thus by choosing base 2, we are able to deal with the simplest cases and use $\log_2 2$ as a unit of information measuring of 1.

What is the information quantity of a series of messages each carrying a different probability?

The **average** information content is calculated by the sum of the probabilities multiplied by the information bits of each message.

For example: in a two letter message **A** or **B** with $2/3$ and $1/3$ probabilities, the average information content for it is:

$$\frac{2}{3} \times \log_2(1/(2/3)) + \frac{1}{3} \times \log_2(1/(1/3)) = 0.918$$

information bits.

Probability of each message

Self-information

Since the probabilities of messages may not be equal, we should weight them according to their probabilities.



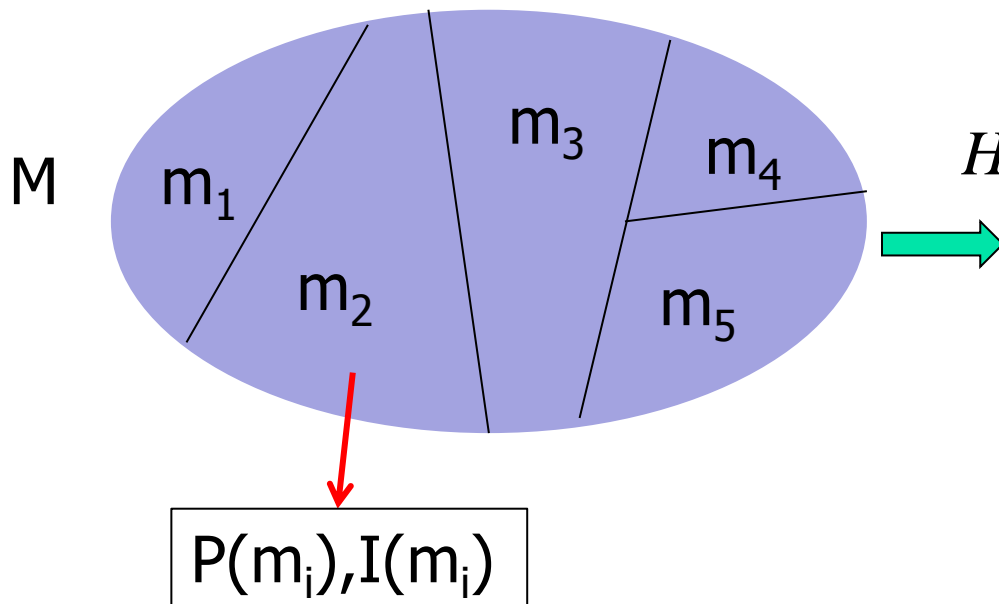
Shannon's Formula

- **Information entropy**
 - a measure of the amount of **uncertainty**. It is defined as the average self-information of a message m from that message space
 - a measure of the uncertainty of classification of that object with regards to all objects being classified.
- Given a set of objects C and a partitioning $c_1 \dots c_n$, the entropy of this classification scheme is given by an addition of the information carried by individual partitions:

$$H(C) = - \sum_1^n p(c_i) \times \log_2(p(c_i))$$

Where $P(C_i)$ is the probability of partition c_i .

An Example




$$\begin{aligned} H(M) &= \sum_{i=1}^n p(m_i) \times I(m_i) \\ &= -\sum_{i=1}^n p(m_i) \times \log_2(p(m_i)) \end{aligned}$$



Example: the information content for the class PROFIT

- $H(C) = - (p(\text{Profit} = \text{up}) \times \log_2(p(\text{Profit} = \text{up})) + p(\text{Profit} = \text{down}) \times \log_2(p(\text{Profit} = \text{down})))$
 $= - (0.5 \times (-1) + 0.5 \times (-1))$
 $= 1$
- This shows that one information bit is needed to represent two different states of the PROFIT: up and down. No information about how the value of up or down is classified based on **other attribute values**.



So, if we use Shannon's formula to determine the most significant attribute in the derivation of the decision tree, we need to use the conditional probability.

- For a probability of a given attribute value (say, AGE is old), what is the information entropy of the class C?

$$H(C | a_j) = - \sum_{i=1}^n p(c_i | a_j) \times \log_2(p(c_i | a_j)) \quad (\text{Formula 1})$$

Where $i = 1 \dots n$. (n different class values). The function $p(c_i | a_j)$ is the probability that the class value is c_i when the attribute has its j^{th} value.

$$H(C) = - \sum_{i=1}^n p(c_i) \times \log_2(p(c_i))$$



AGE is old

$$H(C | a_j) = -\sum_{i=1}^n p(c_i | a_j) \times \log_2(p(c_i | a_j))$$

$$H(\text{PROFIT} | \text{Age} = \text{old})$$

$$= -p(\text{PROFIT} = \text{up} | \text{Age} = \text{old}) \times \log_2(p(\text{PROFIT} = \text{up} | \text{Age} = \text{old}))$$

$$- p(\text{PROFIT} = \text{down} | \text{Age} = \text{old}) \times \log_2(p(\text{PROFIT} = \text{down} | \text{Age} = \text{old}))$$

$$= -0 \times \log_2 0 - 1 \times \log_2 1$$

$$= 0$$

How does each attribute contribute in classifying the class?

$$H(C | A) = \sum_{j=1}^m [p(a_j) \times H(C | a_j)] \quad (\text{Formula 2})$$

Where $j = 1 \dots m$. m is the total number of values for the attribute A .

$$H(\text{PROFIT} | \text{Age})$$

$$= p(\text{Age} = \text{new}) \times H(\text{PROFIT} | \text{Age} = \text{new}) + p(\text{Age} = \text{old}) \times H(\text{PROFIT} | \text{Age} = \text{old})$$

$$+ p(\text{Age} = \text{midlife}) \times H(\text{PROFIT} | \text{Age} = \text{midlife})$$

$$= \frac{3}{10} \times 0 + \frac{3}{10} \times 0 + \frac{4}{10} \times 1$$

$$= 0.4$$

How is a Significant Attribute Determined?

- The final process is then to select the attribute with **smallest entropy** and the partition the data on the basis of this attribute's values. Thus the final choice is made from

$$\min_{t=1}^n \{H(C | A_t)\}$$

(Formula 3)

Age is the best attribute in our case.

Where $t = 1 \dots n$. n is the total number of attributes for the class C .

- One observation is that all three formulas can be merged as one expression, since they are nested (i.e., Formula 1 is nested in Formula 2, and Formula 2 is nested in formula 3).



Induction Process Using ID3

- Determine main attributes and outcomes of tasks;
- Generate or find a training set;
- Convert numeric data into discrete values by range;
- Apply ID3 (Recursively);
 - Use three formulas to select an attribute to split the training set (sort on the attribute and split the rows that the class values are matched with the attribute values).
 - Repeat the above process to use three formulas to select an attribute again for further splitting of the rest of the test set, until all rows of the test set are considered.
- Prune the tree.



Problems in ID3

- Noisy data: poor measurement, data entry errors, or inappropriate outcome recorded.
- Unknown attribute values: can be treated as
 - special values denoted *;
 - replaced with some average for the attribute;
 - or by pre-processing the data and not including in the training set
- Excessive Branching: when a large number of attributes are included. Not all attributes figure as prominently in predicting the outcome. The relatively unimportant attributes should be discarded.



How big should a Training Set be?

- A small training set may result in many **empty leaves** in the decision tree. The training set needs to be increased in size, and the process repeated.
- Some training sets have not identified all the relevant attributes. If this occurs, **then the resultant decision tree will have identical sub-branches for different class attributes. this is called a clash,** and is readily checked by machine and person. The only resolution for this is to call in the expert in order to identify **additional attributes.**



Reading List: Classification Algorithm

- Decision Tree Learning on Very Large Data Sets, By Lawrence O. Hall, Nitesh Chawla and Kevin W. Bowyer.
- A Tutorial of Induction of Decision Trees, By Xue Li.
- Quinlan, J.R. (1986): Induction of Decision Trees. Machine Learning 1: 81-106. 1986.