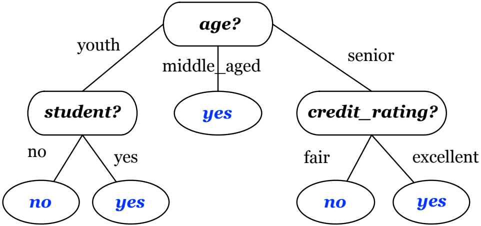
Decision Tree

Decision Tree



- each internal node denotes a test on an attribute
- each branch represents an outcome of the test
- each leaf node holds a class label

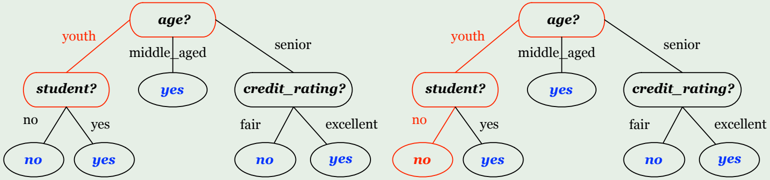
Prediction

- given a new (unseen) tuple: the associated class is unknown

Example

<*Name*, *age*, *student*, *credit_rating*, *buys_computer*> = <Stavros, youth, no, excellent, ?>

<*Name*, *age*, *student*, *credit_rating*, *buys_computer*> = <Stavros, youth, no, excellent, no>



- test the attribute values of the tuple against the decision tree
  - start from the root and trace a path to a leaf node (top-down), based on the attribute values of the tuple
- the class value included in this leaf is assigned to the tuple

Building a Decision Tree

- there are several popular decision tree algorithms, including ID3, C4.5 and CART
- we will describe a general greedy framework followed by the majority of the algorithms
- next, we will discuss some components of this framework (which essentially differentiate the various algorithms) in more detail, as well as some other additional issues on decision tree induction

General Algorithm *Decision_Tree_Induction*

- top-down and recursive

Input

- node N
  - the first time the algorithm is called, N is the root of the tree
- dataset D of training tuples
  - initially the entire training set
- *attribute_list*: holds the set of attributes
  - initially the attributes that remained after data preprocessing
- *attribute_selection_method*: a heuristic process for selecting the attribute that "best" discriminates the given tuples according to class

output

- decision tree

Example

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**

Steps

Step 1

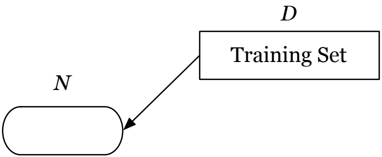
- associate node N with dataset D
- trivial; meant to emphasize that each decision tree node represents a subset of the original (entire) training set

Step 2

- end this process if one of the terminating conditions is satisfied (will be discussed soon)

Step 1

The algorithm is called with the initial single root node *N*, the entire training set as *D*, {age, *income*, *student*, *credit_rating*} as the *attribute_list*, and a certain *attribute_selection_method*



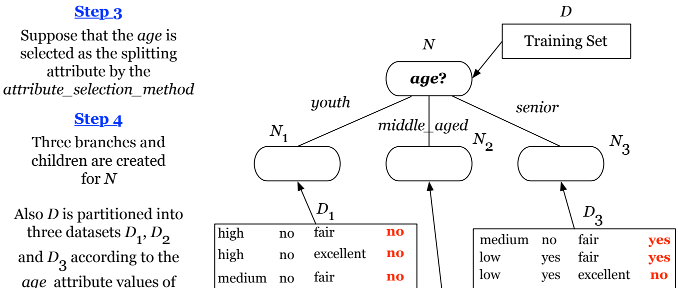
Step 2

No terminating condition is satisfied yet

Step 3

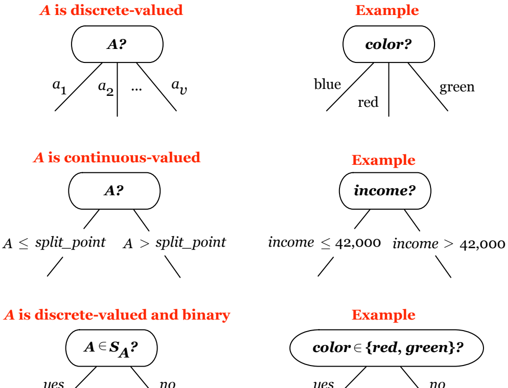
Call *attribute_selection_method*

- use a splitting criterion to select an attribute to test at node N
  - try to "best" partition D into subsets, such that each subset is as "pure" as possible
  - a subset of D is pure, if it contains tuples belonging to the same class
- this test will result in creating new nodes (as children of N), each of which representing a subset of D



Step 4

- a branch for each of the outcomes of the splitting criterion
- a new node N_i is created for each branch i
- if the number of new nodes is m, D is partitioned accordingly into m subsets $D_1, D_2, \ldots, D_m$
  - each D_i contains the tuples that satisfy the splitting criterion outcome of branch i



Example...

Step 3

Suppose that the *age* is selected as the splitting attribute by the *attribute_selection_method*

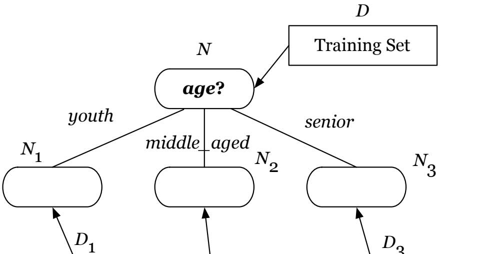
Step 4

Three branches and children are created for N

Also D is partitioned into three datasets D_1, D_2 and D_3 according to the *age* attribute values of the tuples

Step 5

The algorithm is recursively called for (N_1, D_1), (N_2, D_2) and (N_3, D_3), after removing *age* from *attribute_list*



D_1

high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

D_2

high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

D_3

medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

The tuples in D_1, D_2 and D_3 are of the form
<***income, student, credit_rating, class***>

Step 5

- remove the splitting attribute A from *attribute_list*
- call *Decision_Tree_Induction*(N_i, D_i, *attribute_list*, *attribute_selection_method*) recursively for every newly created (N_i, D_i) pair

Terminating Conditions

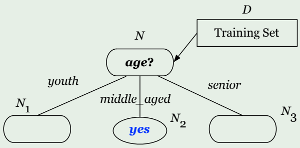
1. all of the tuples in partition D belong to the same class
   - in this case, N becomes a leaf and is labeled with that class

Example

In the procedure call for (N_2, D_2)

Step 2

Since all the tuples in D_2 have the same class, the procedure terminates after making N_2 a leaf node storing the class label ("*yes*")



D_2

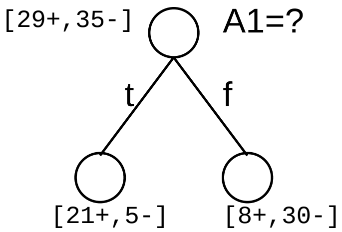
high	no	fair	**yes**
low	yes	excellent	**yes**
medium	no	excellent	**yes**
high	yes	fair	**yes**

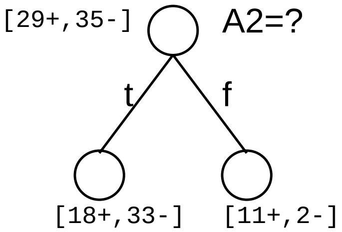
2. there are no remaining attributes in *attribute_list* to help partitioning the tuples of D further
   - N becomes a leaf and is labeled with the majority class in D
3. D is empty
   - N becomes a leaf and is labeled with the majority class of its parent's dataset

Attribute Selection Measures

- ideally, the best splitting criterion is the one that decomposes D into subsets having only tuples of a single class (these subsets are called pure)
- since it may not be always possible to select a splitting criterion that derives only pure subsets, the attribute selection measure provides a ranking for each attribute
- the attribute with the best score is selected as the splitting attribute
- we will study three attribute selection measures
  - Information Gain (used in ID3)
  - Gain Ratio (used in C4.5)
  - Gini Index (used in CART)

Which Attribute is Best?





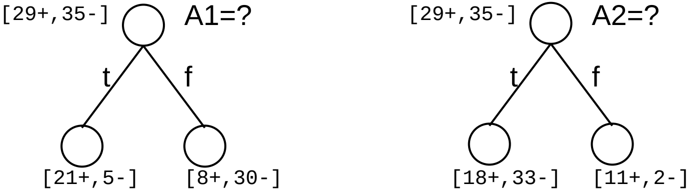
Example

I am thinking of an integer between 1 and 1,000 – what is it?
What is the first question you would ask?

- "Is it 752?", or
- "Is it a prime number between 123 and 239?", or
- "Is it between 1 and 500?"

Which answer provides the most information?

Idea

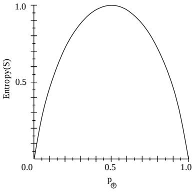


Want attributes that split examples into sets that are relatively pure in one label

How close is a set of instances to having just one label?

Entropy: Intuitive Notion

- measures the impurity, uncertainty, irregularity, surprise
- suppose we have two discrete classes (in general, can have > 2)
  - D: a sample of training examples
  - $p_\oplus$: proportion of positive examples in D
  - $p_\ominus$: proportion of negative examples in D
- optimal purity (impurity/uncertainty$=$ 0): either
  - $p_\oplus = 1, p_\ominus = 0$
  - $p_\oplus = 0, p_\ominus = 1$
- least pure (maximum impurity/uncertainty):
  - $p_\oplus = 0.5, p_\ominus = 0.5$



Entropy: Definition

$$Entropy(D) \equiv -p_{\oplus} \log_b p_{\oplus} - p_{\ominus} \log_b p_{\ominus}$$

- if $p_{\oplus} = 0$, take $p_{\oplus} \log_b p_{\oplus} = 0$

What units is entropy measured in?

Depends on the base b of the log

- $b = e$: nats
- $b = 2$: bits (adopted here)

Example

Entropy of a fair coin = 1 bit

Example

D is a collection of 14 examples, 9 positive and 5 negative

$$\text{Entropy}([9+, 5-]) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

Properties

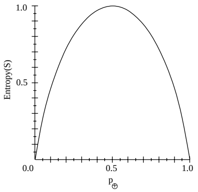
All members of D belong to the same class

- $\rightarrow$ entropy$= 0$

D has an equal number of $+$ve and -ve examples

- $\rightarrow$ entropy$= 1$

Otherwise, entropy is between 0 and 1

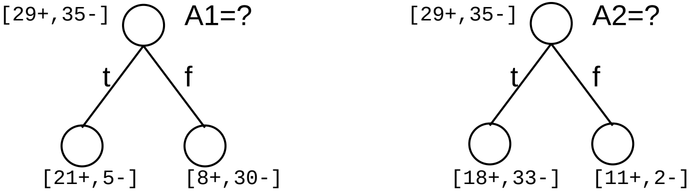


In general, if the target attribute can take on m different values

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

- entropy is between 0 and $\log_2 m$

Information Gain



What is the uncertainty removed by splitting on the value of A?

Definition

The information gain of D relative to attribute A is the expected reduction in entropy caused by knowing the value of A

- D_v: the set of examples in D where attribute A has value v

$$Gain(D, A) \equiv Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Entropy(D_v)$$

Example

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**

- entropy(D) $= 0.940$
- we have to compute $Gain(A)$ for every attribute A

Example

- let us first focus on $A = age$

$$
\begin{aligned}
entropy(D_{\text{youth}}) &= -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \\
entropy(D_{\text{middle\_aged}}) &= -\frac{4}{4}\log_2\frac{4}{4} \\
entropy(D_{\text{senior}}) &= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}
\end{aligned}
$$

$$
\begin{aligned}
\text{weighted average} &= \frac{5}{14} \times entropy(D_{\text{youth}}) + \frac{4}{14} \times entropy(D_{\text{middle\_aged}}) \\
&\quad + \frac{5}{14} \times entropy(D_{\text{senior}}) \\
&= 0.694 \text{ bits}
\end{aligned}
$$

- information gain for age: $Gain(D, age) = 0.940 - 0.694 = 0.246$ bits
- Similarly, $Gain(D, income) = 0.029$, $Gain(D, student) = 0.151$, and $Gain(D, credit\_rating) = 0.048$

Continuous-Valued Attributes

Example

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

Discretize the continuous-valued attributes

- Create a boolean attribute A_c that is true if $A < c$ and false otherwise

How to pick the threshold c?

Continuous-Valued Attributes

Example

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

1. sort the attribute values in the training set ($\{v_1, v_2, \ldots, v_m\}$)
2. generate a set of candidate thresholds midway between these values
   - cut at $44, 54, 66, 76, 85$
   - there are thus $m - 1$ candidate thresholds
3. evaluate these candidate thresholds by the information gain

Example

Example

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

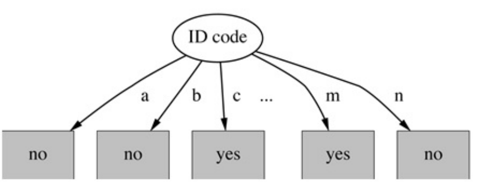
```
6 items, total weight 6.0
Att A1
   Cut at 44.000  (gain 0.191):
   Cut at 54.000  (gain 0.459):
   Cut at 66.000  (gain 0.082):
   Cut at 76.000  (gain 0.000):
   Cut at 85.000  (gain 0.191):
```

Best cut: 54.000

Attribute Selection Measures: Gain Ratio

Example

- if we try to split according to a unique identifier (e.g., *product_id*)



- there will be as many partitions as the number of tuples, and each partition will have a single tuple
- entropy after splitting = 0, which means that the gain is maximized, and $product\_id$ is chosen as the splitting criterion
- obviously, such a partitioning is useless for classification

Information gain is biased towards tests with many outcomes

Gain Ratio

- gain ratio is an extension of information gain, where a large number of partitions is penalized
- penalty should be
  - large when data is evenly spread
  - small when all data belong to one branch
- first compute $SplitInfo_A(D)$, which measures the entropy of the partitioning according to the v distinct values of A:

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

- gain ratio is:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

- we select as the splitting criterion the A that leads to the highest $GainRatio(A)$ value

Example

RID	age	income	student	credit-rating	class:buy_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

Using attribute income

1st partition (low) **D1** has **4 tuples**
2nd partition (medium) **D2** has **6 tuples**
3rd partition (high) **D3** has **4 tuples**

$$Gain(income) = 0.029$$

$$GainRatio(income) = \frac{0.029}{0.926} = 0.031$$

$$SplitInfo_{income}(D) = -\frac{4}{14}\log_2\left(\frac{4}{14}\right) - \frac{6}{14}\log_2\left(\frac{6}{14}\right) - \frac{4}{14}\log_2\left(\frac{4}{14}\right) = 0.926$$

Attribute Selection Measures: Gini Index

- suppose that the class attribute of D has m distinct class values $C_1, C_2, \ldots, C_m$
- $|D|$: cardinality of D, $|C_i|$: number of tuples in D having class label C_i
- probability p_i that a tuple of D belongs to class C_i:$|C_i|/|D|$
- Gini index

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

- $Gini(D)$ is small if most of the tuples belong to a few classes

Gini Index...

- if data set D is split on A into two subsets D_1 and D_2, the gini index is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- reduction in impurity

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- select as the splitting criterion the A that leads to the highest $\Delta gini(A)$ value

Example: PlayTennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Over-fitting in Decision Trees

Tree output

```
A1 = overcast: + (4.0)
A1 = sunny:
|    A3 = high: - (3.0)
|    A3 = normal: + (2.0)
A1 = rain:
|    A4 = weak: + (3.0)
|    A4 = strong: - (2.0)
```

Over-fitting in Decision Trees

Example

Consider adding noisy training example #15:
$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$

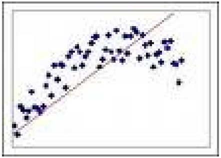
```
A1 = overcast: + (4.0)
A1 = sunny:
|    A3 = high: - (3.0)
|    A3 = normal: + (2.0)
A1 = rain:
|    A4 = weak: + (3.0)
|    A4 = strong: - (2.0)
```

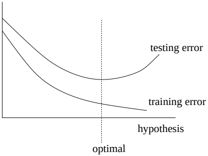
```
A1 = overcast: + (4.0)
A1 = sunny:
|    A2 = hot: - (3.0)
|    A2 = cold: + (1.0)
|    A2 = mild:
|    |    A3 = high: - (1.0)
|    |    A3 = normal: + (1.0)
A1 = rain:
|    A4 = weak: + (3.0)
|    A4 = strong: - (2.0)
```

- more complex than the original tree
- fits the noisy data better than the old tree

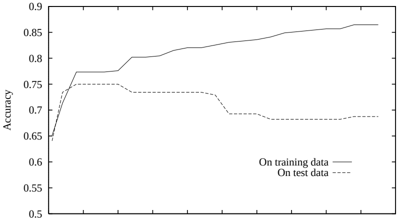
Overfitting

- special anomalies of the training set may have been incorporated in the tree





- may further affect the accuracy of the tree on the test set (i.e., during prediction)
- sometimes smaller trees are preferable because of their interpretability



Tackling Overfitting

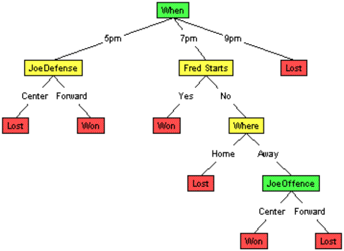
- early-stopping
- pruning

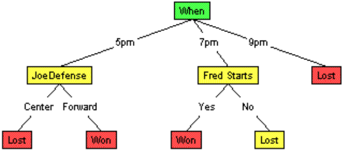
Early-Stopping

- add extra terminating conditions at Step 2 of the decision tree induction algorithm
  - stop if the number of tuples is fewer than some user-specified threshold
- may be difficult to choose an appropriate threshold

Pruning

- grow the whole tree, then prune





How to prune a decision node?

- remove the subtree rooted at that node
- make it a leaf node

The leaves that are left may not necessarily be pure

- assign it the most common classification of the training examples affiliated with that node

Pruning...

The pruned tree usually misclassifies more training examples than the unpruned tree

When to stop pruning?

- uses a validation set
- stop if the pruned tree performs worse than the original on the validation set

Which node to prune?

- remove the one that most improves validation set accuracy
- greedy approach

Pruning Algorithm

Reduced-error pruning

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node
2. Greedily remove the one that most improves validation set accuracy