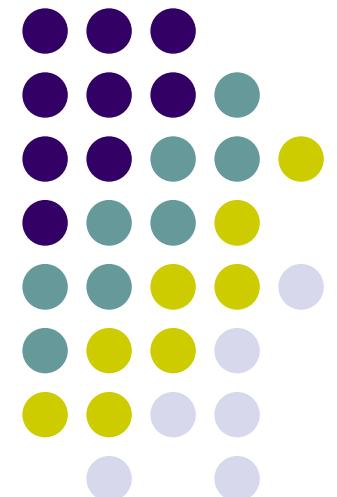
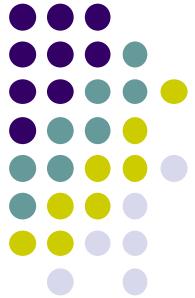


Decision Trees

Jacques Savoy
University of Neuchâtel

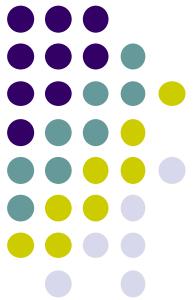


Ian H. Witten, Eibe Frank: Data Mining. Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2005.
J. Ross Quinlan: C4.5 Programs for Machine Learning.
Morgan Kaufmann, San Mateo, 1998.



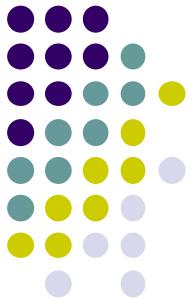
Overview

- **Introduction and Examples**
- Tree Construction Algorithm
- Gain Ratio
- Numerical Attribute
- Pruning
- Complexity
- From Tree to Rules



Introduction

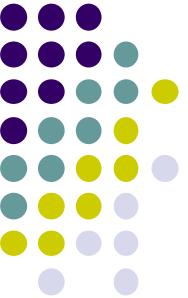
- Simple algorithms and easy to understand (“How-To” manuals)
- Attributes-based examples
- Limited to Boolean classifier (to learn a discrete-valued functions)
- A tree in which
 - node corresponds to a test
 - leaf specifies the value to be returned
- Used in many practical data mining systems



Example: Restaurant Problem

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range (\$, \$\$, \$\$\$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

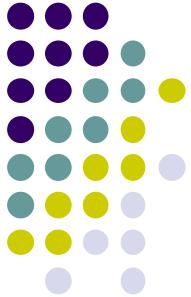


Attribute-based Representations

- Examples described by attribute values (Boolean, discrete, continuous), e.g., situations where I will/won't wait for a table:

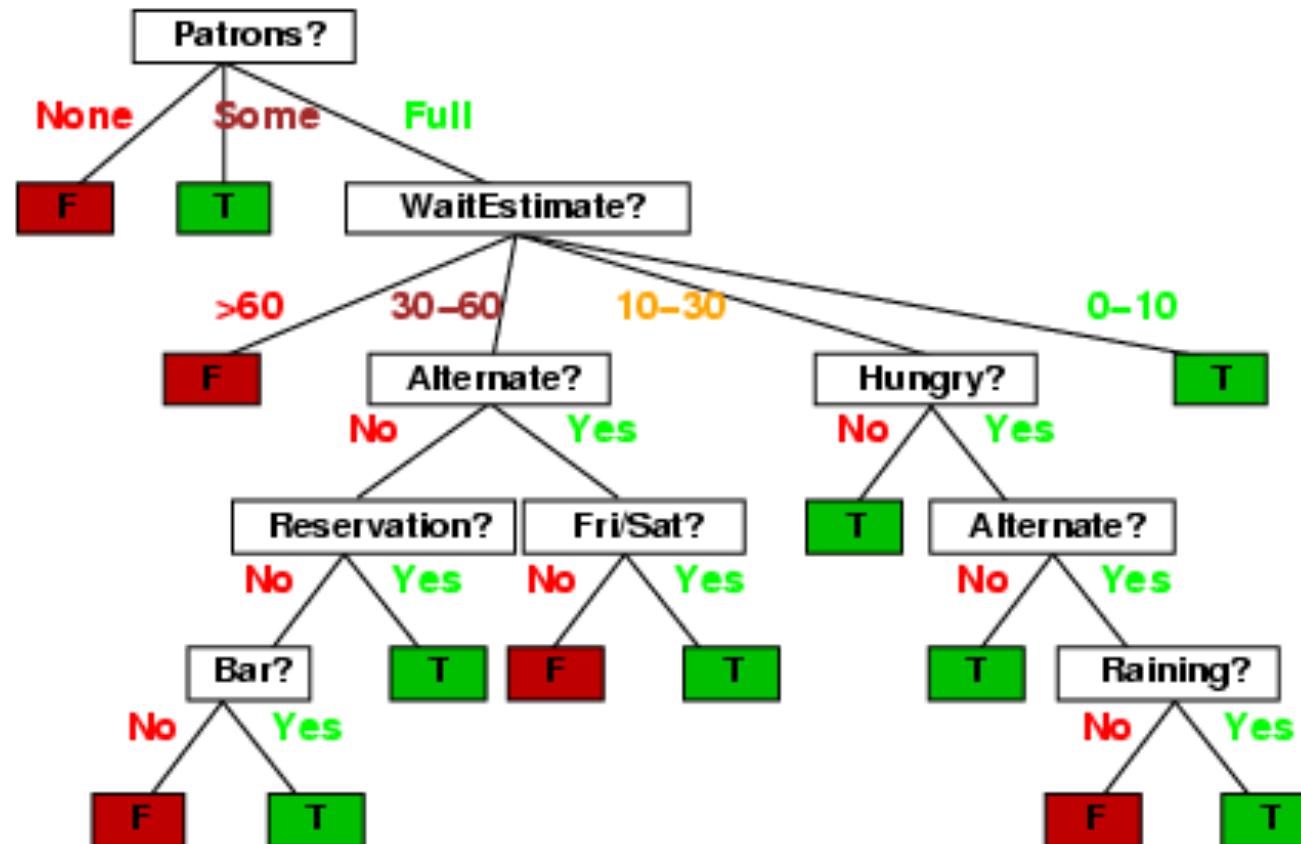
Example	Attributes										Target <i>Wait</i>
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

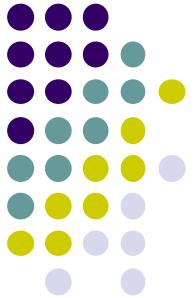
- Classification of examples is positive (T) or negative (F)



Example

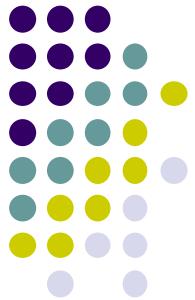
- One possible representation for hypotheses
- Here is my tree (manual construction) for deciding whether to wait:





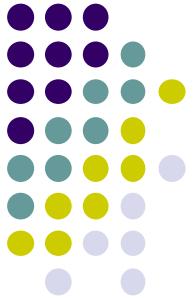
Example

- Problem: decide whether to play given the weather conditions:
 - Outlook: general condition (*sunny*, *overcast*, *rainy*)
 - Temperature: given in degree or (*hot*, *mild*, *cool*)
 - Humidity: numeric (%) or (*high*, *normal*)
 - Windy: Boolean (*true* / *false*)
 - Play: Decision Boolean (*true* / *false*)
- This is just a toy-side (and well-know) example



Example: Weather Problem

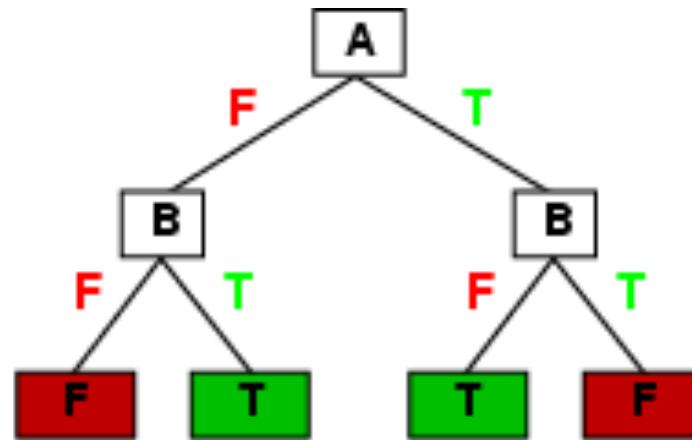
Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	mild	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no



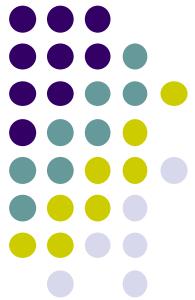
Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions,
truth table row \rightarrow path to leaf:

A	B	$A \text{ xor } B$
F	F	F
F	T	T
T	F	T
T	T	F

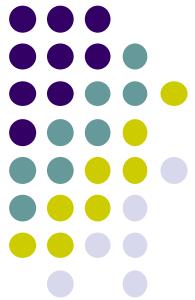


- In our previous example
 $\forall s, \text{WillWait}(s) \leftrightarrow (P_1(s) \vee P_2(s) \vee \dots \vee P_n(s))$
where $P_j(s)$ (conjunction of tests) is a path in the tree



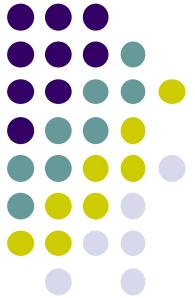
Expressiveness

- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example. Not very useful!
- Could be more difficult for some functions
 - parity function: returns 1 if and only if an even number of inputs are 1
 - majority function: returns 1 if more than half of its inputs are 1
- Prefer to find more *compact* decision trees
- A decision tree may not use all attributes (e.g., the price in the restaurant example)



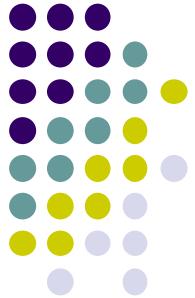
Overview

- Introduction and Examples
- **Tree Construction Algorithm**
- Gain Ratio
- Numerical Attribute
- Pruning
- Complexity
- From Tree to Rules



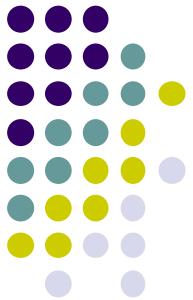
Tree Construction

- Aim: find a (small) tree *consistent* with the training examples
- Seems difficult (due to the large number of hypotheses)
- Table look-up: one path to a leaf for each example
 - but cannot reveal hidden pattern in the data
 - do not generalize well
 - we can usually create a smaller tree



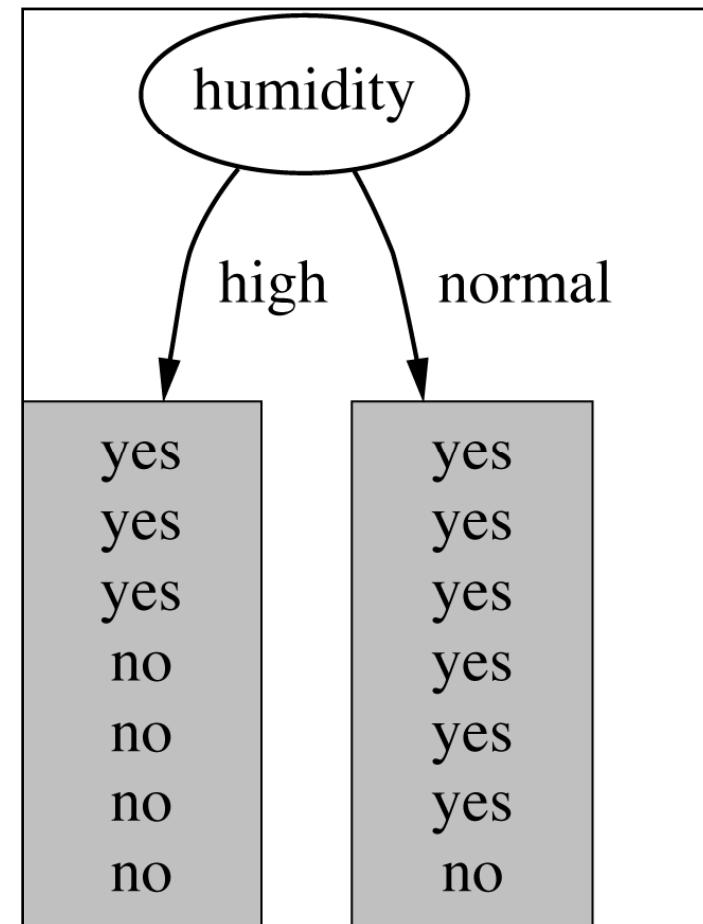
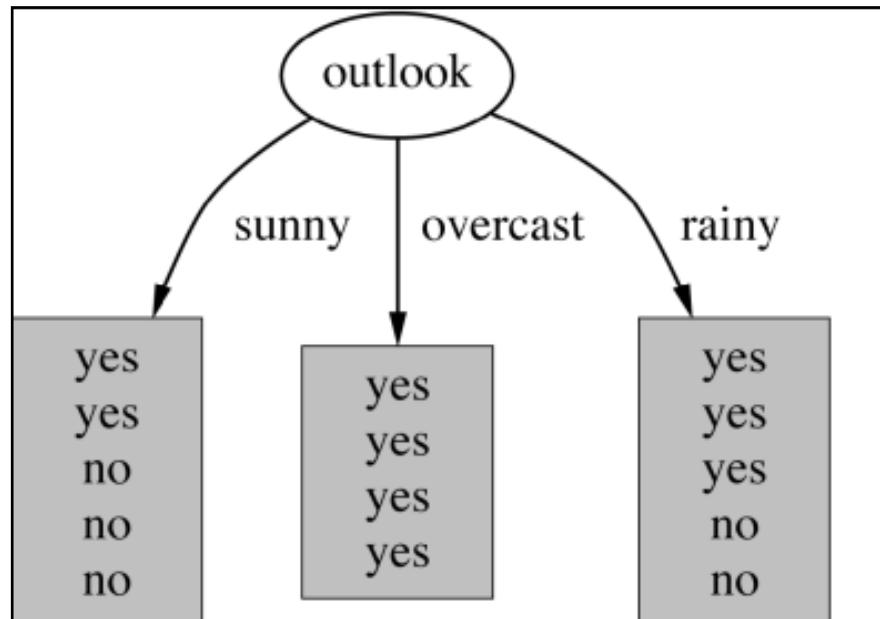
Tree Construction

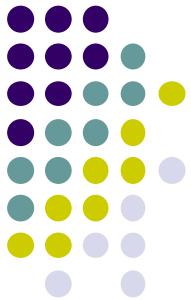
- Strategy: top down
- Recursive *divide-and-conquer* fashion
 - First: select attribute for root node
 - Create branch for each possible attribute value
 - Then: split instances into subsets
One for each branch extending from the node
 - Finally: repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class



Choosing an Attribute

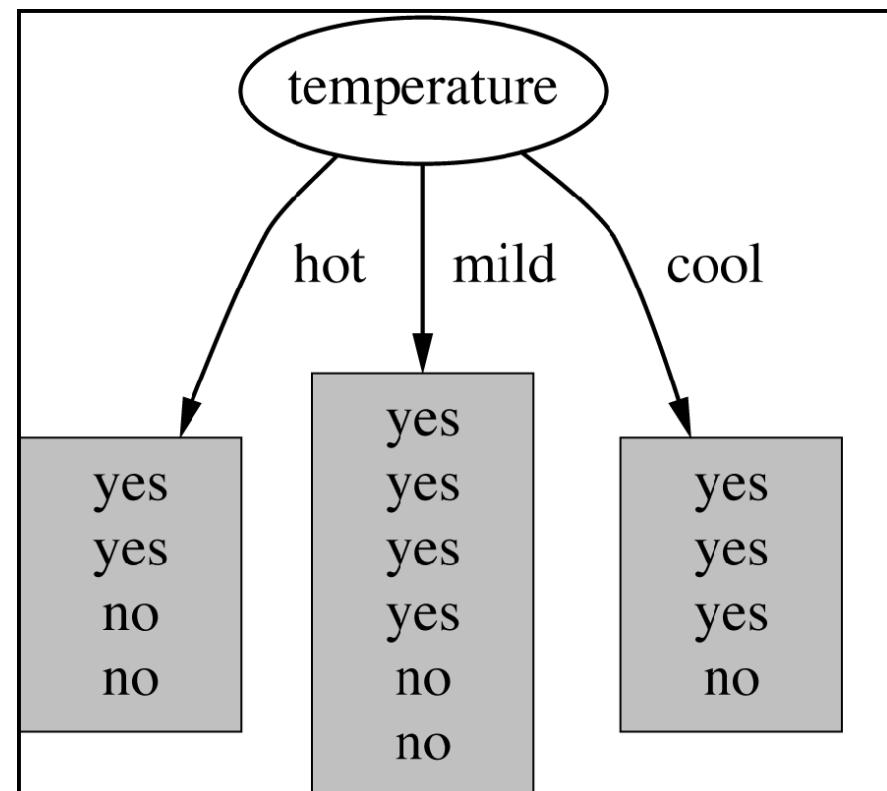
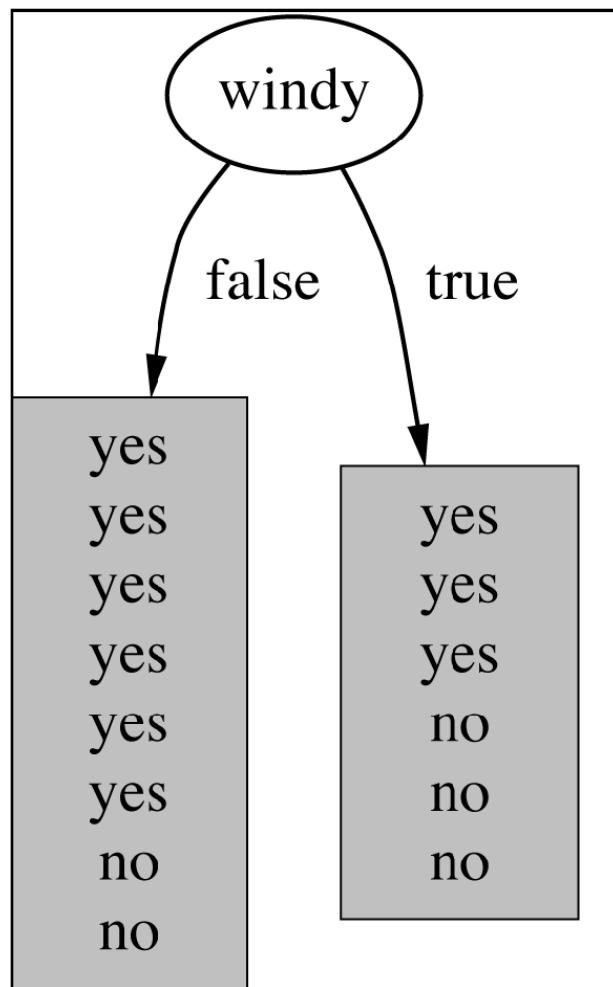
In our weather problem

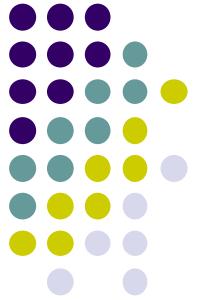




Choosing an Attribute

Other choices





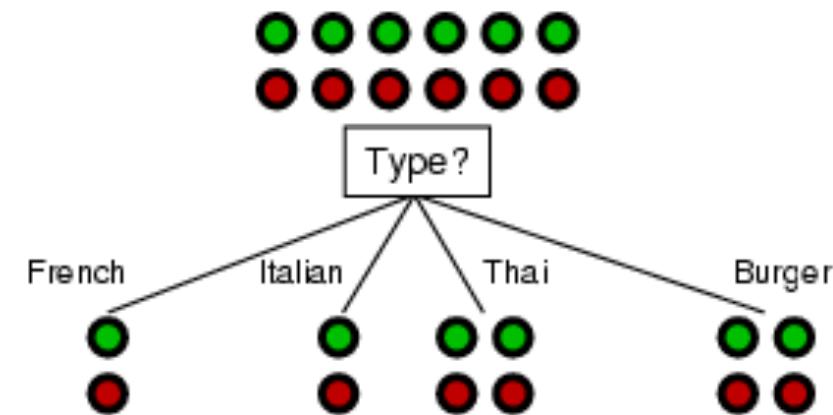
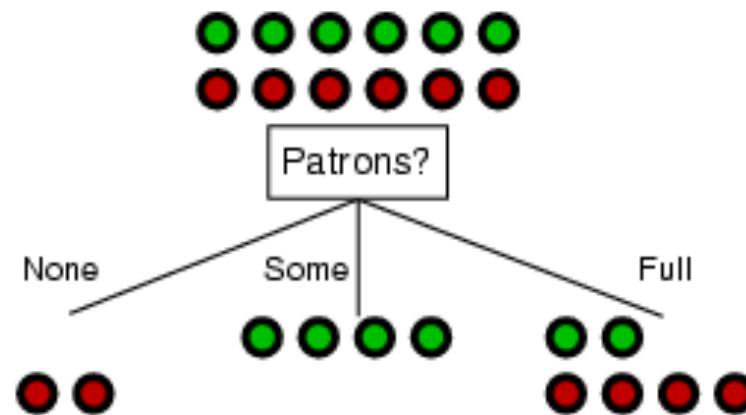
Tree Construction

- Idea: (recursively) choose “most significant” attribute as root of (sub)tree with simple cases:
 - all examples are positive (or negative): create a leaf with 1 (or 0)
 - no examples left (no such example): return the default value (majority classification of the parent’s node)
 - no attributes left: if we have both positive and negative → noise (majority vote)

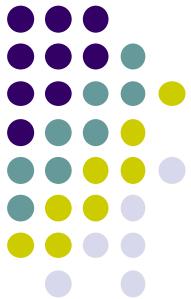


Choosing an Attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”
In our restaurant example

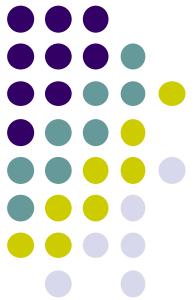


- Patrons?*
Is a better choice (why?)



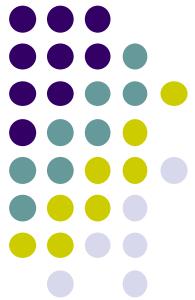
Decision Tree Learning (DTL)

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    best  $\leftarrow$  CHOOSE-ATTRIBUTE(attributes, examples)
    tree  $\leftarrow$  a new decision tree with root test best
    for each value  $v_i$  of best do
      examplesi  $\leftarrow$  {elements of examples with best =  $v_i$ }
      subtree  $\leftarrow$  DTL(examplesi, attributes - best, MODE(examples))
      add a branch to tree with label  $v_i$  and subtree subtree
  return tree
```



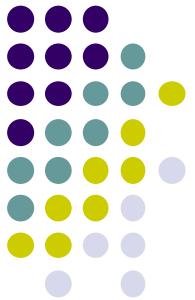
Choosing an Attribute

- Which is the best attribute?
 - Want to get the smallest tree
 - Heuristic: choose the attribute that produces the “purest” nodes
- Popular *impurity criterion*: *information gain*
 - Information gain increases with the average purity of the subsets
- Strategy: choose attribute that gives greatest information gain



Using Information Theory

- Compute the entropy that characterizes the (im)purity of an arbitrary collection of examples
- Measure information in *bit*
 - 1 bit reduces by 50% the uncertainty
 - 0 bit = non uncertainty
- Measured by the \log_2 (reduction)
- Given a probability distribution, the info required to predict an event is the distribution's *entropy*
- Entropy gives the information required in bits
 - (can involve fractions of bits!)
- How to compute the entropy (uncertainty, impurity)?

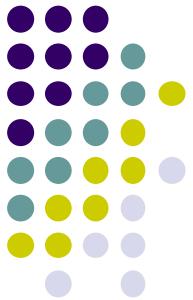


Using Information Theory

- Information Content (Entropy):

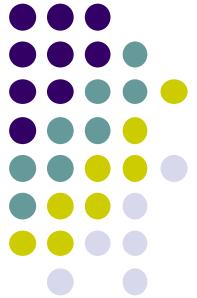
$$\text{Entropy} = I(p_1, p_2, \dots, p_n) = - \sum_{i=1}^n p_i \cdot \log_2(p_i)$$

- Weighted mean
- Example with two messages:
 - $I(0.5, 0.5) = -0.5 \cdot \log_2(0.5) - 0.5 \cdot \log_2(0.5)$
 $= -0.5 \cdot (-1) - 0.5 \cdot (-1) = 1$
 - $I(0.8, 0.2) = -0.8 \cdot \log_2(0.8) - 0.2 \cdot \log_2(0.2) = 0.72$
 - If one of the two issues is more probable, the uncertainty is reduced (the impurity is reduced)



Using Information Theory

- Not limited to only two issues
Can the entropy be greater than 1?
- Example
 - $I(0.4, 0.3, 0.2, 0.1) =$
 $-0.4 \cdot \log_2(0.4) - 0.3 \cdot \log_2(0.3) - 0.2 \cdot \log_2(0.2)$
 $-0.1 \cdot \log_2(0.1) = 1.846$
 - $I(0.25, 0.25, 0.25, 0.25) = 2$
- We admit that $0 \cdot \log_2(0) = 0$ and thus $I(1, 0) = 0$
Explain why $I(1, 0) = 0$?



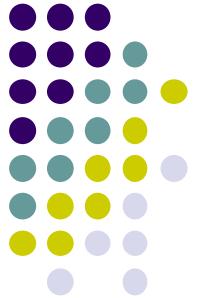
Using Information Theory

- Thus to implement Choose-Attribute in the DTL algorithm, we will use the entropy principle
- Try to separate positive and negative examples
- For a training set containing p positive examples and q negative examples ($n=p+q$):

$$I\left(\frac{p}{n}, \frac{q}{n}\right) = -\frac{p}{n} \cdot \log_2 \left[\frac{p}{n}\right] - \frac{q}{n} \cdot \log_2 \left[\frac{q}{n}\right]$$

- If our training sample (weather) contains 14 cases, 9 positive and 5 negative examples, we have

$$I(9/14, 5/14) = 0.940$$

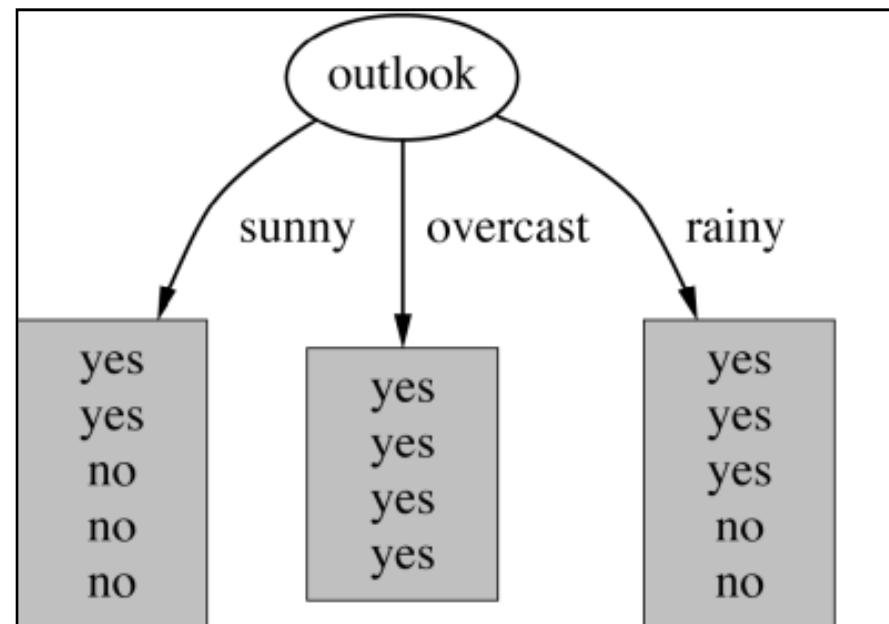


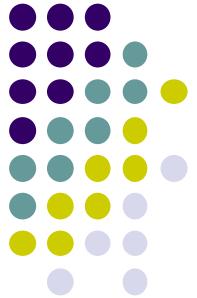
Using Information Theory

With the attribute Outlook,
the 14 cases are
subdivided according to:

- *Outlook = Sunny*
- *Outlook = Overcast*
- *Outlook = Rainy*

Can we compute the
impurity of each
attribute=*value* pair?





Using Information Theory

- *Outlook = Sunny:*

$$I(2/5, 3/5) = -\frac{2}{5} \cdot \log_2 \left[\frac{2}{5} \right] - \frac{3}{5} \cdot \log_2 \left[\frac{3}{5} \right] = 0.971 \text{ bits}$$

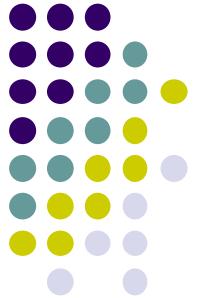
- *Outlook = Overcast:*

$$I(1, 0) = -1 \cdot \log_2(1) - 0 \cdot \log_2(0) = 0 \text{ bits}$$

- *Outlook = Rainy:*

$$I(3/5, 2/5) = -\frac{3}{5} \cdot \log_2 \left[\frac{3}{5} \right] - \frac{2}{5} \cdot \log_2 \left[\frac{2}{5} \right] = 0.971 \text{ bits}$$

- How can we combine these three values?



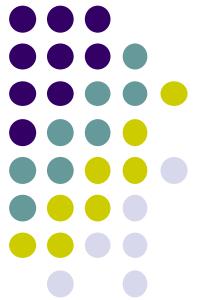
Using Information Theory

- Expected information for the attribute (weighted average)

$$Info(\text{sunny, overcast, rainy}) =$$

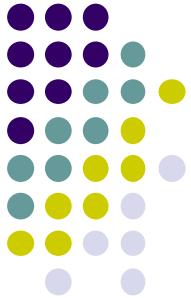
$$+ \frac{5}{14} \cdot 0.971 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0.971 = 0.693 \text{ bits}$$

- Without considering any attribute,
the entropy was 0.940 bits
- If we select “Outlook”,
the entropy is reduced to 0.693 bits
- Before: 0.940, after: 0.693, gain?



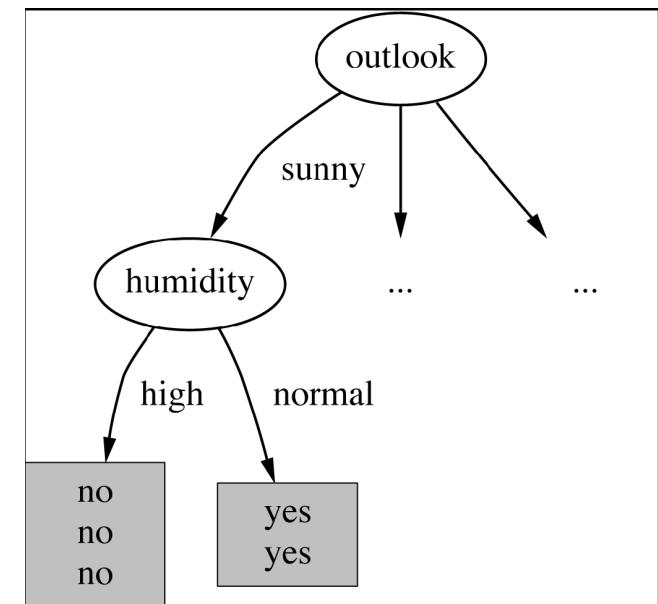
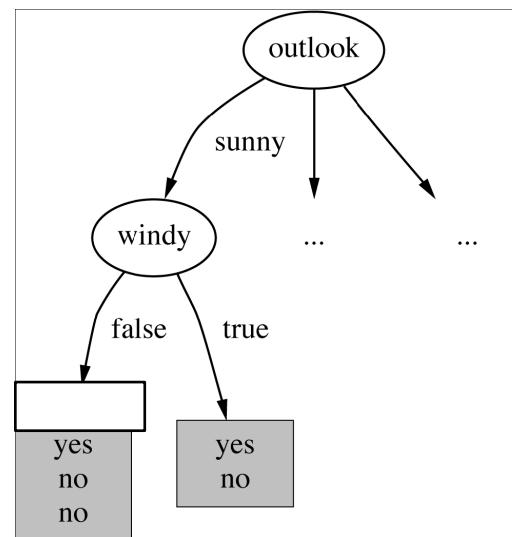
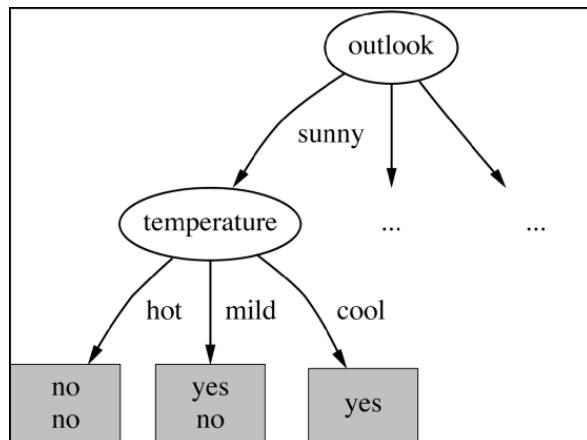
Information Gain

- Information gain:
information before splitting – information after splitting
- With the weather problem
Information gain for attributes from weather data:
 - $\text{gain}(\text{Outlook}) = 0.940 - 0.693 = \mathbf{0.247}$ bits
 - $\text{gain}(\text{Temperature}) = 0.029$ bits
 - $\text{gain}(\text{Humidity}) = 0.152$ bits
 - $\text{gain}(\text{Windy}) = 0.048$ bits
- Among all possible attributes, choose the attribute with the largest IG (*outlook* in our case)



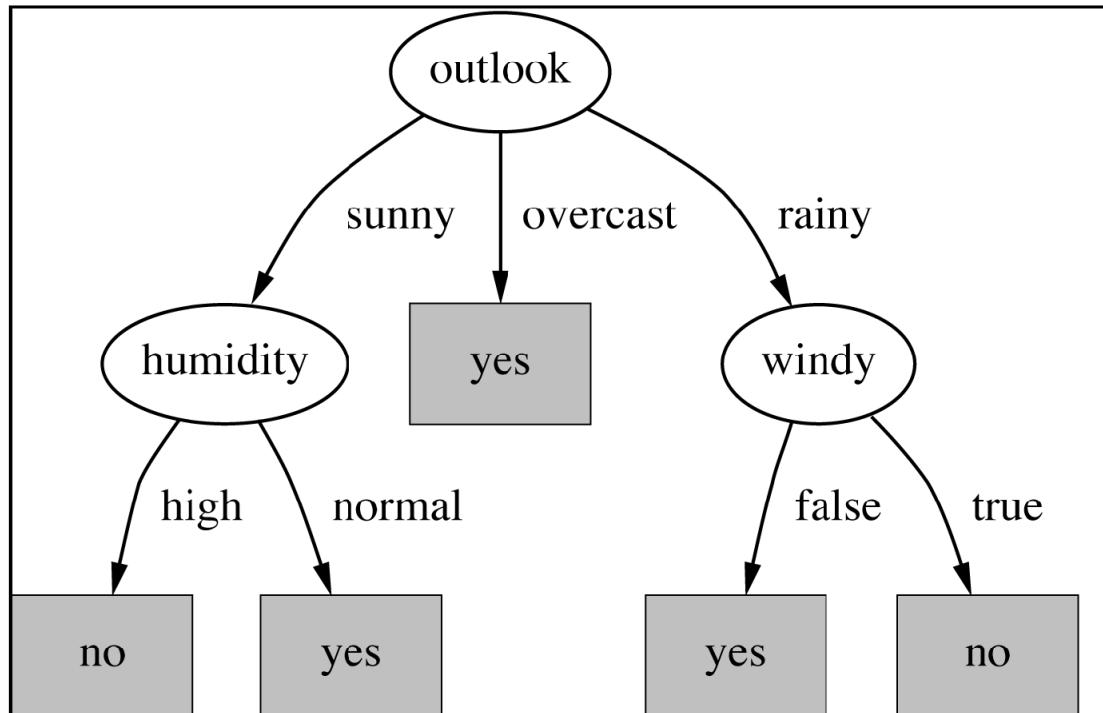
Tree Construction

- Continuing the decision tree
 - $\text{gain}(\text{Temperature}) = 0.571 \text{ bits}$
 - $\text{gain}(\text{Humidity}) = 0.971 \text{ bits}$
 - $\text{gain}(\text{Windy}) = 0.020 \text{ bits}$



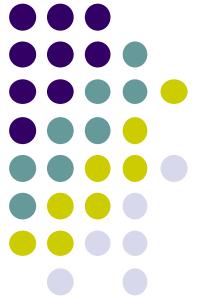


Tree Construction



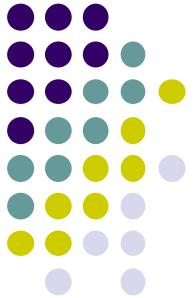
Note: not all leaves need to be pure; sometimes identical instances have different classes

→ Splitting stops when data can't be split any further



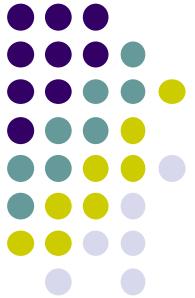
Tree Construction

- Is the information gain the best measure?
- Properties we require from a purity measure:
 - When node is pure, measure should be zero
 - When impurity is maximal (i.e., all classes equally likely), measure should be maximal
 - Measure should obey *multi-stage property* (i.e. decisions can be made in several stages):
$$\text{measure}([2,3,4]) = \text{measure}([2,7]) + (7/9) \times \text{measure}([3,4])$$
- Entropy is the only function that satisfies all three properties!



Example: Tree Construction

- With the restaurant data
Starting point $p = 6$ (positive), $q = 6$ (negative), $n = 12$
 $I(6/12, 6/12) = 1$
- We can select the *Rain* attribute
that can be *true* or *false*
 - with *Rain* = *true*
we can find $p_i = 4$ and $q_i = 4$
 - with *Rain* = *false*
we can find $p_j = 2$ and $q_j = 2$



Information Gain

- The attribute Rain (*true/false*) with 8 *false* → WillWait 4 *false* / 4 *true* and 4 *true* → WillWait 2 *false* / 2 *true*
- The $Info(Rain)$ is equal to 1

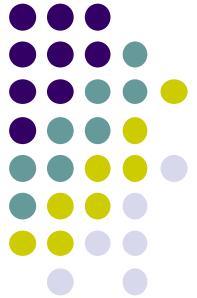
$$Info(Rain) =$$

$$\frac{8}{12} \cdot I(1/2, 1/2) + \frac{4}{12} \cdot I(1/2, 1/2)$$

$$\frac{8}{12} \cdot 2 \cdot [-0.5 \cdot \log(0.5)] + \frac{4}{12} \cdot 2 \cdot [-0.5 \cdot \log(0.5)] = 1$$

a great uncertainty is still present after considering Rain. Thus selecting Rain, not very useful!

- Information Gain
- $$IG(Rain) = 1 - 1 = 0$$



Information Gain

- Consider the attribute Type (*French, Thai, Burger, Italian*)

2 *French* → WillWait 1 *false* / 1 *true*

4 *Thai* → WillWait 2 *false* / 2 *true*

4 *Burger* → WillWait 2 *false* / 2 *true*

2 *Italian* → WillWait 1 *false* / 1 *true*

$$Info(Type) =$$

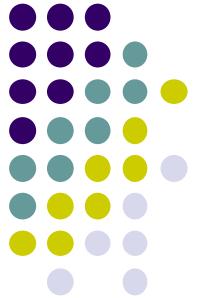
$$\frac{2}{12} \cdot I(1/2, 1/2) + \frac{2}{12} \cdot I(1/2, 1/2)$$

$$+ \frac{4}{12} \cdot I(2/4, 2/4) + \frac{4}{12} \cdot I(2/4, 2/4)$$

$$\frac{2}{12} \cdot 2 \cdot [-0.5 \cdot \log(0.5)] + \frac{2}{12} \cdot 2 \cdot [-0.5 \cdot \log(0.5)]$$

$$+ \frac{4}{12} \cdot 4 \cdot [-0.25 \cdot \log(0.25)] + \frac{4}{12} \cdot 4 \cdot [-0.25 \cdot \log(0.25)] = 1$$

- Type does not reduce the uncertainty



Information Gain

- Consider the attribute Patrons (*Full, Some, None*)

6 *Full* → WillWait 4 *false* / 2 *true*

4 *Some* → WillWait 4 *true*

2 *None* → WillWait 2 *false*

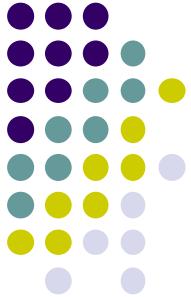
$$Info(Patrons) =$$

$$\frac{2}{12} \cdot I(0, 1) + \frac{4}{12} \cdot I(1, 0) + \frac{6}{12} \cdot I(2/6, 4/6) =$$

$$\frac{2}{12} \cdot 0 + \frac{4}{12} \cdot 0 + \frac{6}{12} \cdot [-0.33 \cdot \log(0.33) - 0.66 \cdot$$

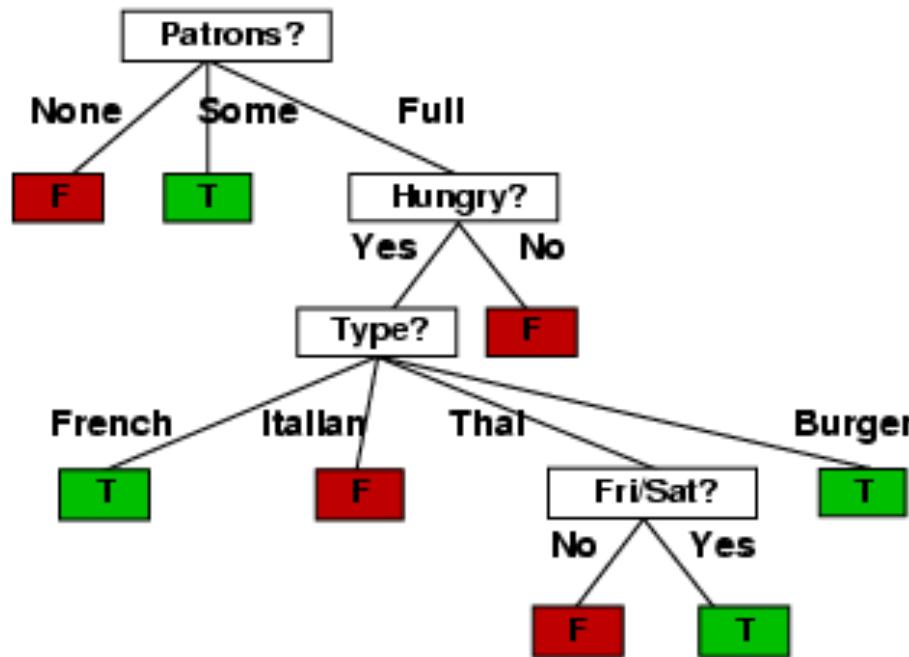
$$\log(0.66)] = 0.459$$

- $IG(Patrons) = 1 - 0.459 = 0.541$ reduces the uncertainty

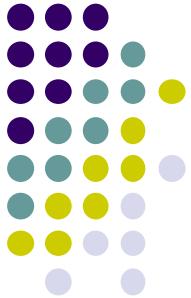


Example

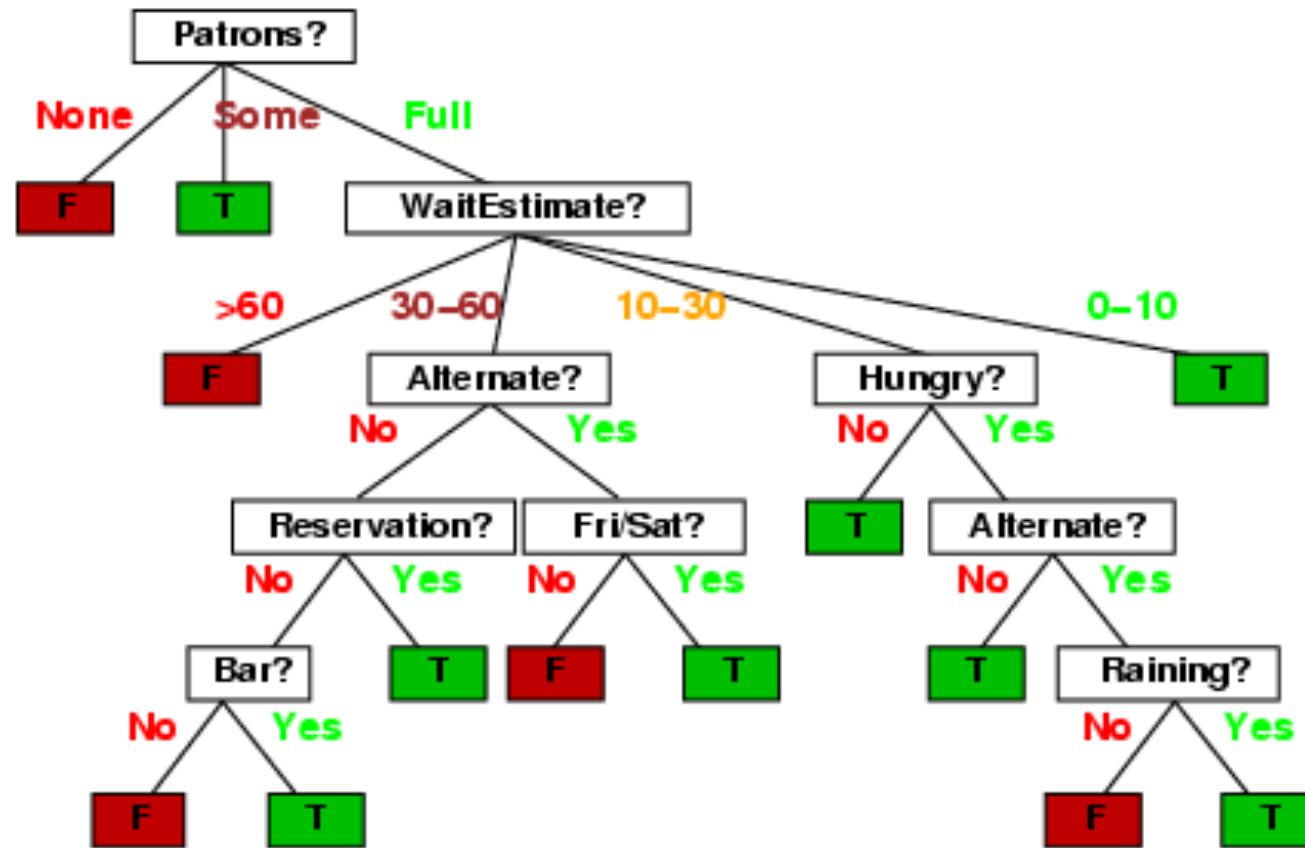
- Decision tree learned from the 12 examples:

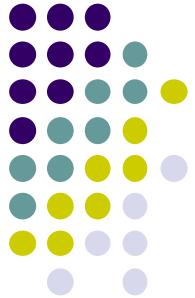


- Substantially simpler than our previous tree - a more complex hypothesis isn't justified by small amount of data



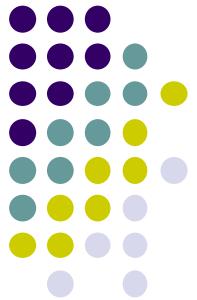
Example: Manual Construction





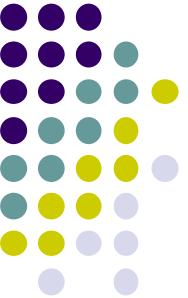
Overview

- Introduction and Examples
- Tree Construction Algorithm
- **Gain Ratio**
- Numerical Attribute
- Pruning
- Complexity
- From Tree to Rules



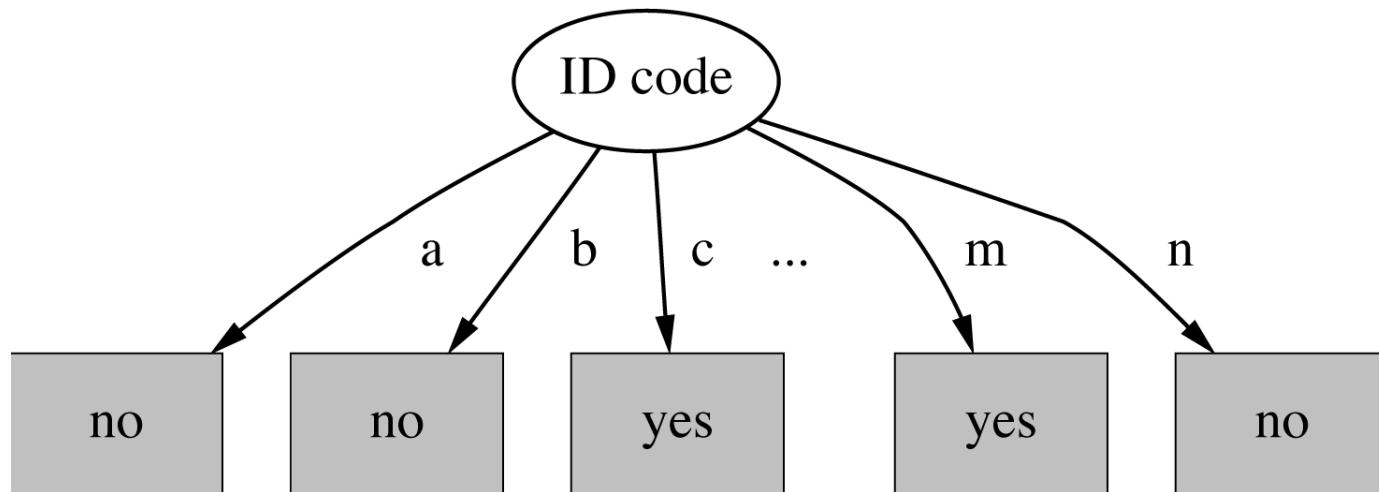
Limit of the Information Gain

- Problematic: attributes with a large number of values (extreme case: ID code)
- Subsets are more likely to be pure if there is a large number of values
- → Information gain is biased towards choosing attributes with a large number of values
- → This may result in *overfitting* (selection of an attribute that is nonoptimal for prediction)
- Another problem: *fragmentation*

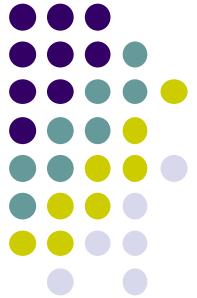


Limit of the Information Gain

- Example with an ID code

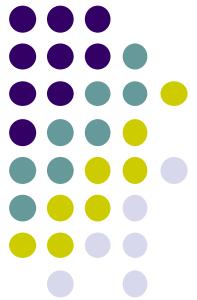


- Entropy of split:
Information gain is maximal for ID code
(namely 0.940 bits)



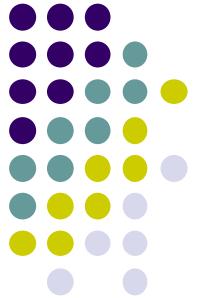
Gain Ratio

- The *Information gain* criterion has a strong bias in favor of tests with many outcomes (e.g., case identification, in which all subsets will necessarily contain cases of a single class).
- In such cases, $\text{Info}(ID) = 0$, and $\text{IG}(ID)$ is max.



Gain Ratio

- *Gain ratio*: a modification of the information gain that reduces its bias
- Gain ratio takes number and size of branches into account when choosing an attribute
 - It corrects the information gain by taking the *intrinsic information* of a split into account
- *Intrinsic information*: entropy of distribution of instances into branches (i.e. how much info do we need to tell which branch an instance belongs to)
Also called *split info*



Gain Ratio

- Example: intrinsic information for ID code
 $I[1/14, 1/14, \dots, 1/14] = 14 \times [-1/14 \times \log(1/14)] = 3.807 \text{ bits}$
- Value of attribute decreases as intrinsic information (or split info) gets larger
- Definition of *gain ratio*:

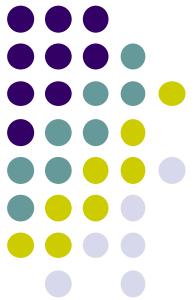
$$\text{Gain ratio(attribute)} = \frac{\text{Gain(attribute)}}{\text{Intrinsic info(attribute)}}$$

$$\text{Gain ratio(ID code)} = \frac{0.940 \text{ bit}}{3.807 \text{ bits}} = 0.246$$



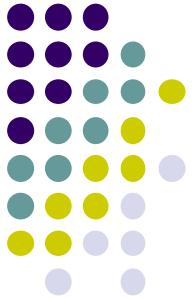
Gain Ratio

Outlook		Temperature	
info	0.693	info	0.911
gain $0.94 - 0.693 =$	0.247	gain $0.94 - 0.911 =$	0.029
split info $I[5/14,4/14,5/14]$	1.577	split info $I[4/14,6/14,4/14]$	1.557
gain ratio $0.247/1.577 =$	0.157	gain ratio $0.029/1.577 =$	0.019
Humidity		Windy	
info	0.788	info	0.892
gain $0.94 - 0.788 =$	0.152	gain $0.94 - 0.892 =$	0.048
split info $I[7/14,7/14]$	1.0	split info $I[8/14,6/14]$	0.985
gain ratio $0.152/1.0 =$	0.152	gain ratio $0.048/0.985 =$	0.049



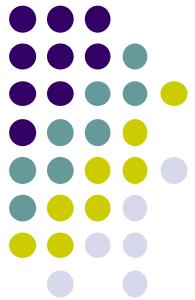
Gain Ratio

- “Outlook” still comes out top
- However: “ID code” has greater gain ratio
 - Standard fix: *ad hoc* test to prevent splitting on that type of attribute
- Problem with gain ratio: it may overcompensate
 - May choose an attribute just because its intrinsic information is very low
 - Standard fix: only consider attributes with greater than average information gain



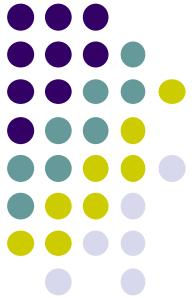
Overview

- Introduction and Examples
- Tree Construction Algorithm
- Gain Ratio
- **Numerical Attribute**
- Pruning
- Complexity
- From Tree to Rules



Numeric Attribute

- Standard method: binary splits
 - e.g., temperature < 45
- Unlike nominal attributes, every attribute has many possible split points
- Solution is straightforward extension:
 - Evaluate info gain (or other measure) for every possible split point of attribute
 - Choose “best” split point
 - Info gain for best split point is info gain for attribute
- Computationally more demanding



Numeric Attribute

- Split on temperature attribute:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Y	No	Y	Y	Y	No	No	Y	Y	Y	No	Y	Y	No

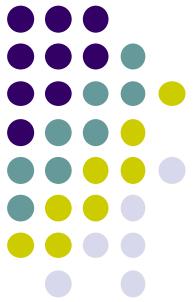
- e.g.

temperature < 71.5: yes/4, no/2

temperature > 71.5: yes/5, no/3

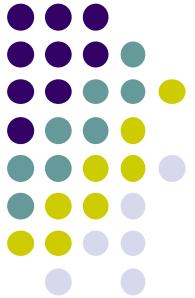
- $\text{Info}([4,2],[5,3])$
 $= 6/14 I([4,2]) + 8/14 I([5,3])$
 $= 0.939 \text{ bits}$

- Place split points halfway between values
- Can evaluate all split points in one pass! (Binary split)



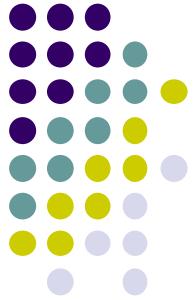
Missing Values

- Consider missing value as one possible value
- Split instances with missing values into pieces
 - A piece going down a branch receives a weight proportional to the popularity of the branch weights sum to 1
- Info gain works with fractional instances
 - use sums of weights instead of counts
- During classification, split the instance into pieces in the same way
 - Merge probability distribution using weights



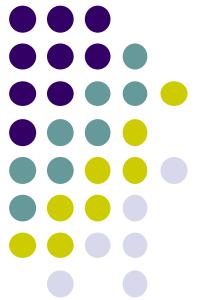
Overview

- Introduction and Examples
- Tree Construction Algorithm
- Gain Ratio
- Numerical Attribute
- **Pruning**
- Complexity
- From Tree to Rules



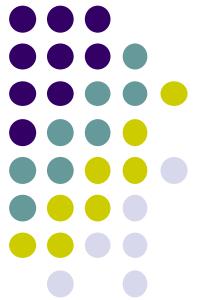
Pruning

- Prevent overfitting to noise in the data
- “Prune” the decision tree
- Two strategies:
 - *Post-pruning*
take a fully-grown decision tree and discard unreliable parts
 - *Pre-pruning*
stop growing a branch when information becomes unreliable
- Post-pruning preferred in practice — pre-pruning can “stop early”



Pre-Pruning

- Based on statistical significance test
 - Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
- Most popular test: *chi-squared test*
- ID3 used chi-squared test in addition to information gain
 - Only statistically significant attributes were allowed to be selected by information gain procedure



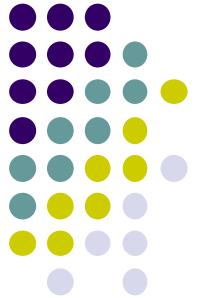
Chi-square test

- Having an attribute with r different values and two decision classes (P and N), we may build a contingency table as follow

	class = P	class = N	
value c_1	n_{1P}	n_{1N}	n_1
value c_2	n_{2P}	n_{2N}	n_2
...
value c_r	n_{rP}	n_{rN}	n_r
	n_P	n_N	n

$$n_j = n_{jP} + n_{jN}$$

$$n = n_P + n_N = \sum_{j=1}^r n_j$$

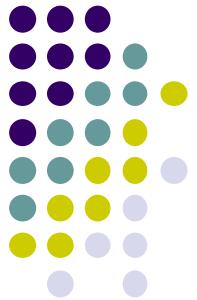


Chi-square test

- Example with three values and two possible decisions

	class = P	class = N	
value c ₁	12	6	18
value c ₂	24	13	37
value c ₃	36	15	51
	72	34	106

- Do we see a relationship between the splitting in the three possible values and the classification P or N?



Chi-square test

- Based on the contingency table, we can define the expected number as:

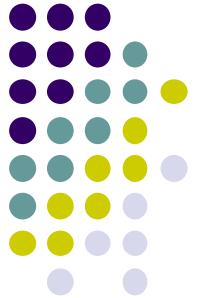
$$\mu_{jP} = \frac{n_P \cdot n_j}{n}$$

$$\mu_{jN} = \frac{n_N \cdot n_j}{n}$$

and we can compute

$$\chi^2 = \sum_{i=1}^r \left(\frac{(n_{iP} - \mu_{iP})^2}{\mu_{iP}} + \frac{(n_{iN} - \mu_{iN})^2}{\mu_{iN}} \right)$$

that follows a chi-distribution with $r-1$ dof
(e.g., with $dof = 1$ and $\alpha = 0.95$, limit value = 3.84, one-tail)

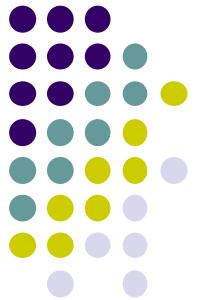


Chi-square test

- Expected values with three values and two possible decisions

	class = P	class = N	
value c ₁	12.2	5.8	18
value c ₂	25.1	11.9	37
value c ₃	34.6	16.3	51
	72	34	106

- Do we see a real differences between the observed values and the expected values (under the assumption that there is no relationship between the attribute and the classification P or N)?



Chi-square test

- Example with three values and two possible decisions

	class = P	class = N	
value c ₁	15	3	18
value c ₂	34	3	37
value c ₃	15	36	51
	64	42	106

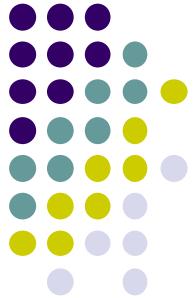
- Chi-square limit (5%, dof=2), 5.991
Chi-square limit (1%, dof=2), 15.086
- Chi-square computed: 46.437



Pre-Pruning

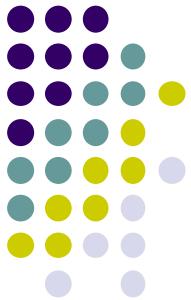
- Pre-pruning may stop the growth process prematurely: *early stopping*
- Classic example: XOR/Parity problem
 - No *individual* attribute exhibits any significant association to the class
 - Structure is only visible in fully expanded tree
 - Pre-pruning won't expand the root node
- But: XOR-type problems rare in practice
- And: pre-pruning faster than postpruning

A	B	class $A \otimes B$
0	0	0
0	1	1
1	0	1
1	1	0



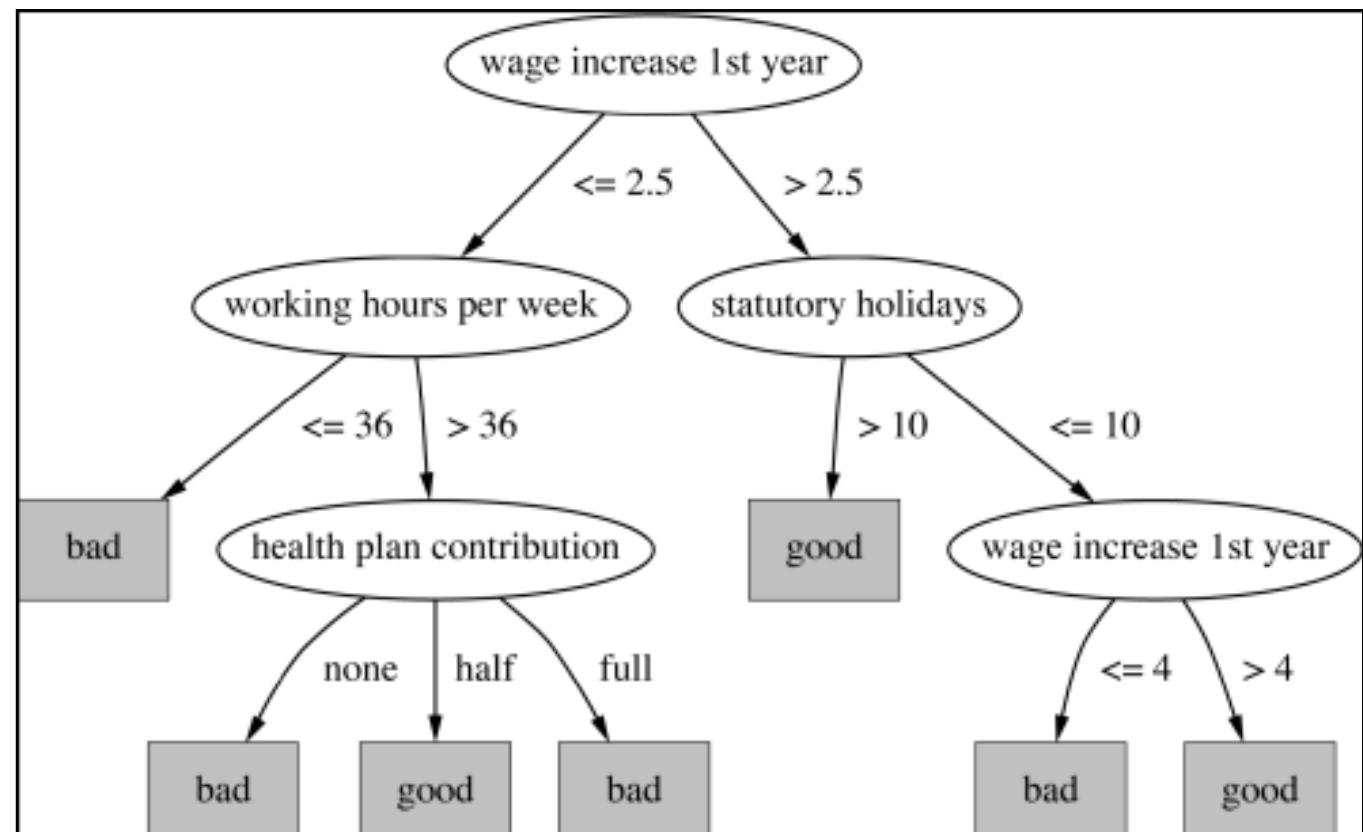
Post-Pruning

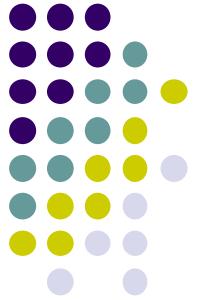
- First, build full tree
- Then, prune it
 - Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects
- Two pruning operations:
 - *Subtree replacement*
 - *Subtree raising*
- Possible strategies:
 - error estimation
 - significance testing
 - MDL principle



Post-Pruning

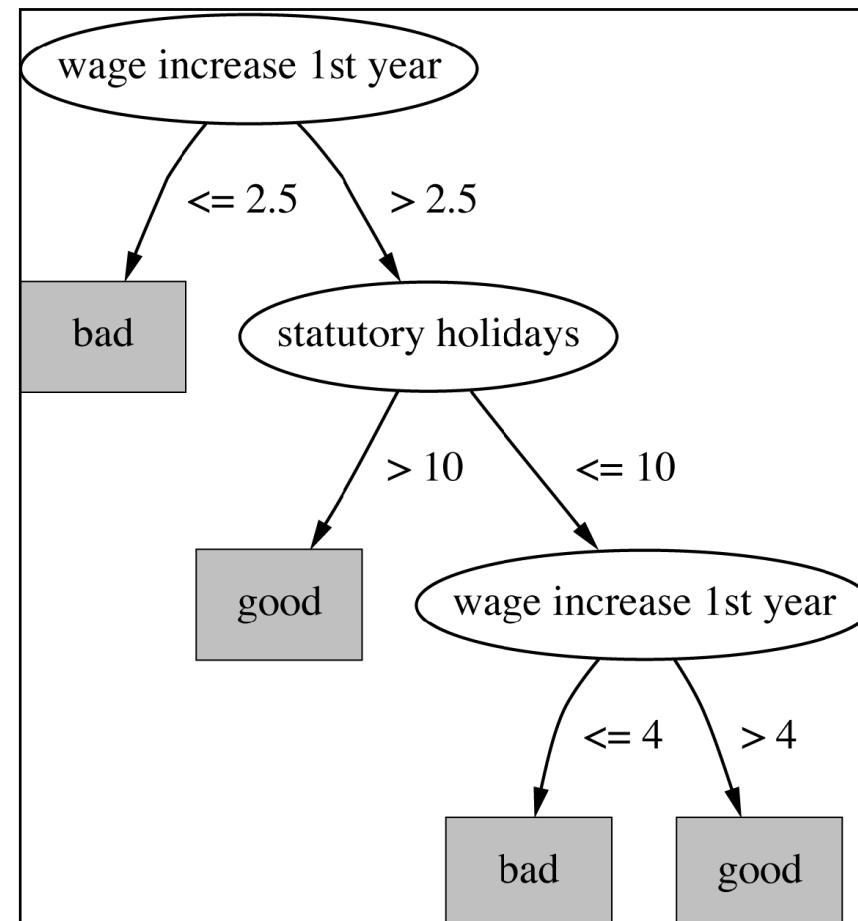
- Subtree Replacement (*Bottom-up*)
- Consider replacing a tree only after considering all its subtrees

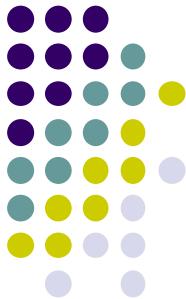




Post-Pruning

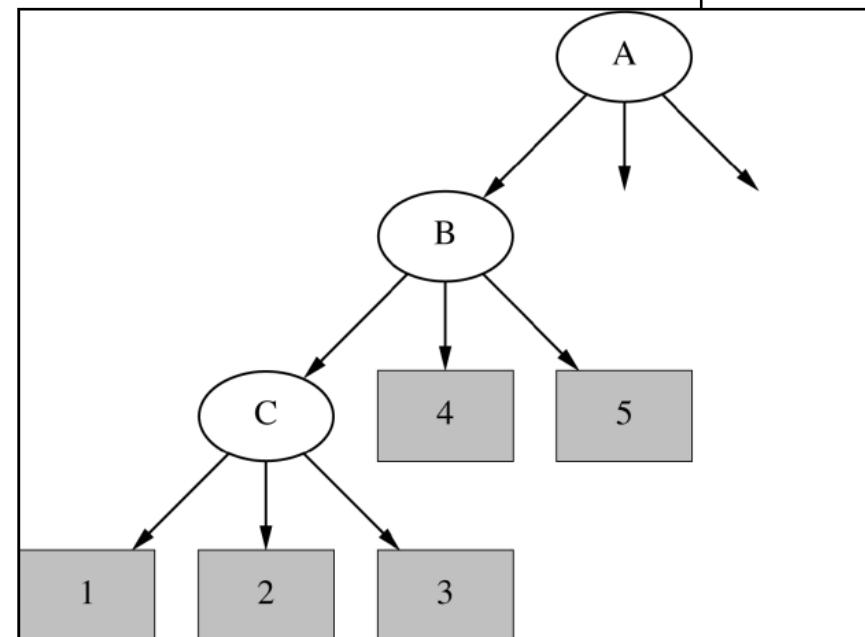
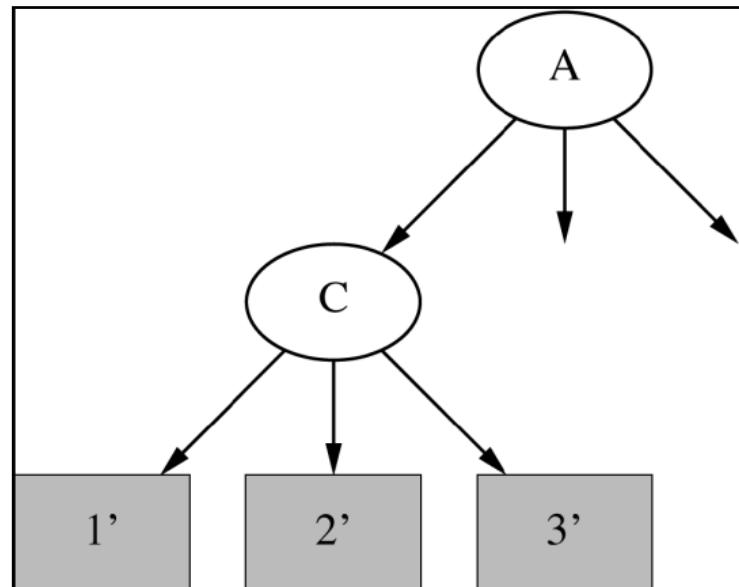
- The final pruned tree

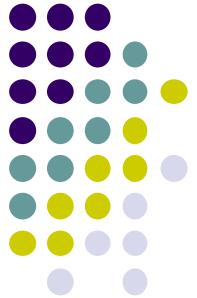




Post-Pruning

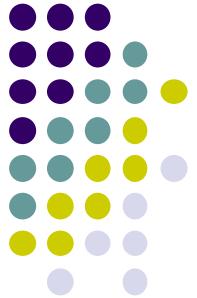
- Subtree raising
- Delete node
 - Redistribute instances
 - Slower than subtree replacement
- *(Worthwhile?)*





Estimating Error Rate

- Prune only if it does not increase the estimated error
- Error on the training data is NOT a useful estimator
(would result in almost no pruning because the tree was built with this data set)
- Use holdout set for pruning (“reduced-error pruning”)
(but reduced the number of examples to build the tree)

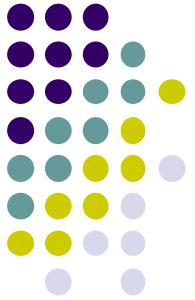


Estimating Error Rate

- C4.5's method
 - Derive confidence interval from training data
 - Use a *heuristic* limit, derived from this, for pruning (seems to work well in practice)
 - Standard Bernoulli process-based method
We have E errors from N instances (but from the training data)
we fix q as the true error probability (with $p + q = 1$)
 - If we fixed c a given confidence (in C4.5, $c = 25\%$)
the confidence limit z is

$$\text{Prob} \left[\frac{f - q}{\sqrt{q \cdot (1 - q) / N}} > z \right] = c$$

with $f = E/N$, the observed error rate and q the true error rate

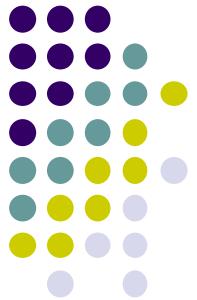


C4.5's Method

- Error estimate for subtree is weighted sum of error estimates for all its leaves
- Upper confidence limit for q , for a node:

$$e = \frac{\left(f + \frac{z^2}{2 \cdot N} + z \cdot \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4 \cdot N^2}} \right)}{1 + \frac{z^2}{N}}$$

- If $c = 25\%$ then $z = 0.69$ (from normal distribution)
- $f = E/N$, is the error on the training data
- N is the number of instances covered by the leaf

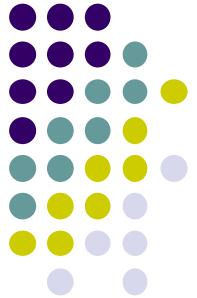


C4.5's Method

- If we have observed an error rate ($f = E/N$) = $2/6 = 0.333$, the upper limit of the true error rate could be (with $c=25\%$ and thus $z = 0.69$)

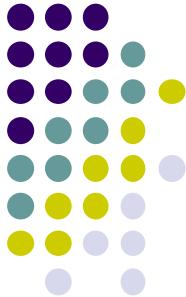
$$e = \frac{\left(0.33 + \frac{0.69^2}{12} + 0.69 \cdot \sqrt{\frac{0.33}{6} - \frac{0.33^2}{6} + \frac{0.69^2}{144}}\right)}{1 + \frac{0.69^2}{6}}$$

$$\frac{\left(0.33 + 0.039 + 0.69 \cdot \sqrt{0.0764 - 0.0181 + 0.0033}\right)}{1 + 0.07935} = 0.47$$



C4.5's Method

- If we have observed an error rate ($f = E/N = 5/14 = 0.357$), the upper limit of the true error rate could be (with $c=25\%$ and thus $z = 0.69$)
 $e = 0.46$
- If we consider a two-class problem, this upper value of an error rate is relatively large.
- In the next example, the e value at the parent node is $e=0.46$ (if we do not split further this node) while the combined error rate for the children is $e=0.51$.



C4.5's Method

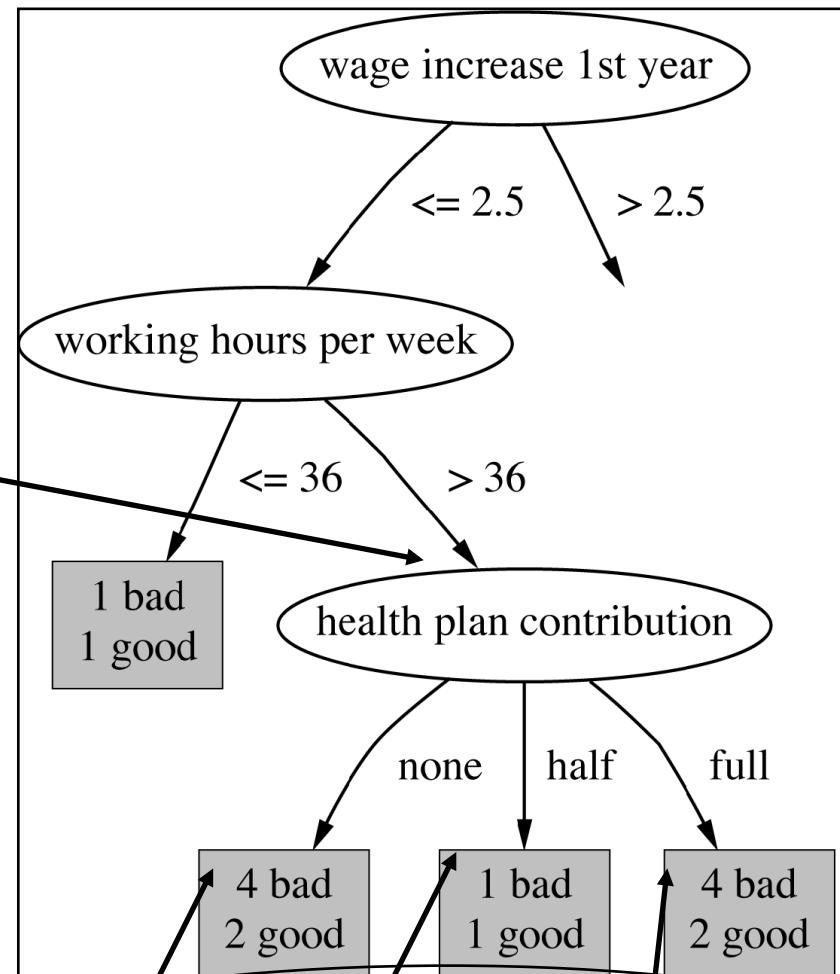
Example

$f=5/14$
 $e=0.46$

$e > 0.46$ so prune!

Combined 6:2:6
gives $e=0.51$

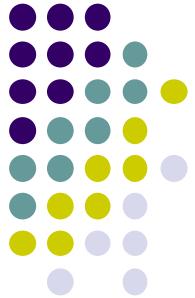
An observed error rate of 0.33 leads to a upper limit of 0.47



$f=0.33$
 $e=0.47$

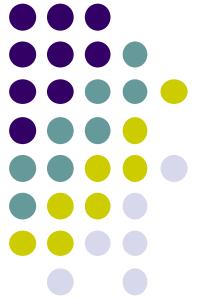
$f=0.5$
 $e=0.72$

$f=0.33$
 $e=0.47$



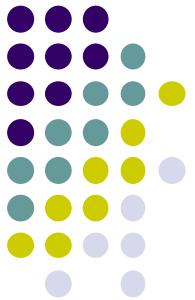
Overview

- Introduction and Examples
- Tree Construction Algorithm
- Gain Ratio
- Numerical Attribute
- Pruning
- **Complexity**
- From Tree to Rules



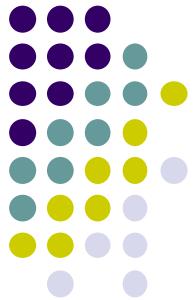
Complexity

- Assume that the training data contains
 - m attributes
 - n training instances
 - we assume that the tree depth $O(\log n)$ (relatively balanced)
- Building a tree $O(m n \log n)$
 - we need to consider the n examples at a given depth
 - and with $\log(n)$ depth, we have $n \log(n)$
 - at each node, all the attributes are considered, thus $O(m n \log n)$
- Subtree replacement $O(n)$
 - During the pruning, we need to estimate the error rate at each tree node (the tree has at most n leaves)



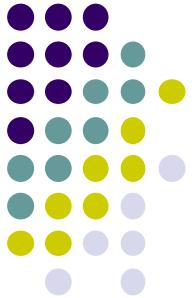
Complexity

- Subtree raising $O(n (\log n)^2)$
 - Every instance may have to be redistributed at every node between its leaf and the root.
 - Cost for redistribution (on average, between the root and its leaf): thus the depth of the tree $O(\log n)$ with n nodes, $O(n \log n)$ plus the reclassification cost $O(\log n)$
- Total cost: $O(m n \log n) + O(n (\log n)^2)$



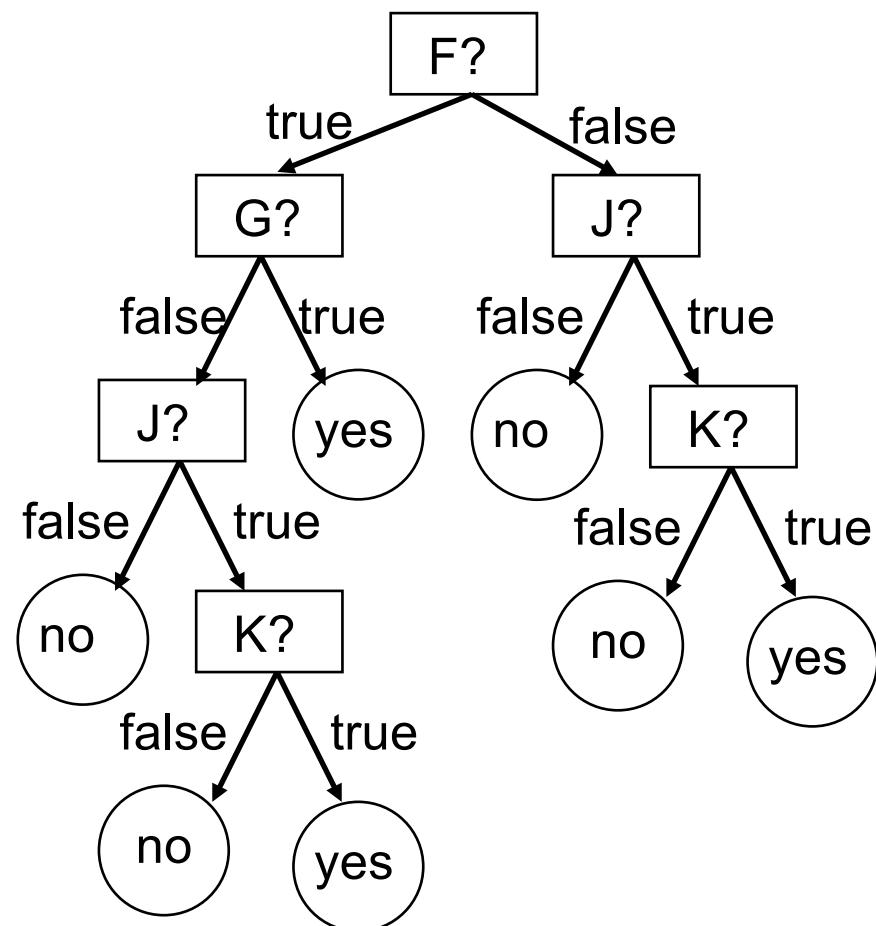
Overview

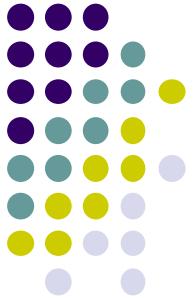
- Introduction and Examples
- Tree Construction Algorithm
- Gain Ratio
- Numerical Attribute
- Pruning
- Complexity
- **From Tree to Rules**



From Tree to Rules

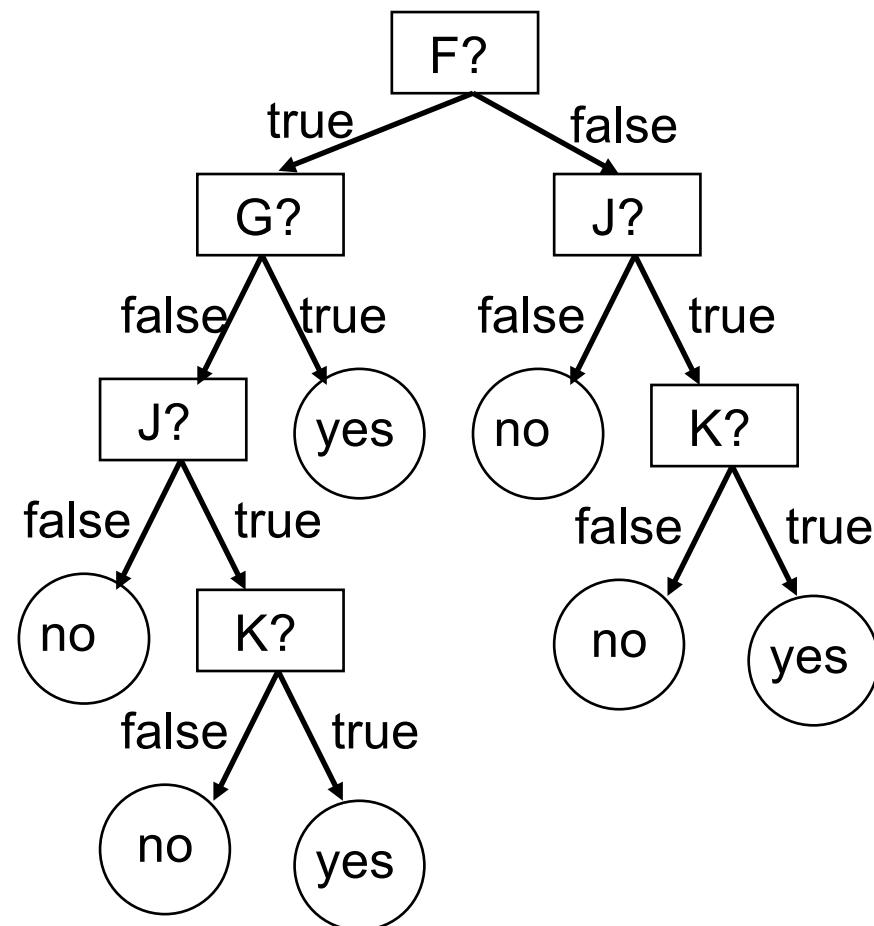
- Simple way: one rule for each leaf





Decision tree learning

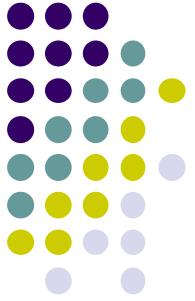
From decision tree to rules



The rules are:

If $(F \cap -G \cap -J)$ then *false*
If $(F \cap -G \cap J \cap -K)$ then *false*
If $(F \cap -G \cap J \cap K)$ then *true*
If $(F \cap G)$ then *true*

...



From Tree to Rules

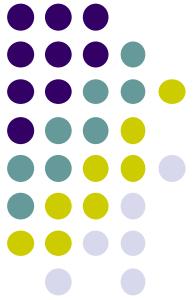
- C4.5 rules: greedily prune conditions from each rule if this reduces its estimated error
 - Can produce duplicate rules
 - Check for this at the end
- Then
 - look at each class in turn
 - consider the rules for that class
 - find a “good” subset (guided by MDL)

If $(F \cap \neg G \cap J \cap K)$ then *true*

If $(J \cap K)$ then *true*

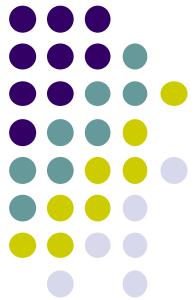
If $(J \cap \neg G \cap F \cap K)$ then *true*

...



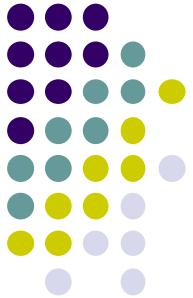
From Tree to Rules

- C4.5 rules: greedily prune conditions from each rule if this reduces its estimated error
 - Can produce duplicate rules
 - Check for this at the end
- Then
 - look at each class in turn
 - consider the rules for that class
 - find a “good” subset (guided by MDL)
- Then rank the subsets to avoid conflicts
- Finally, remove rules (greedily) if this decreases error on the training data



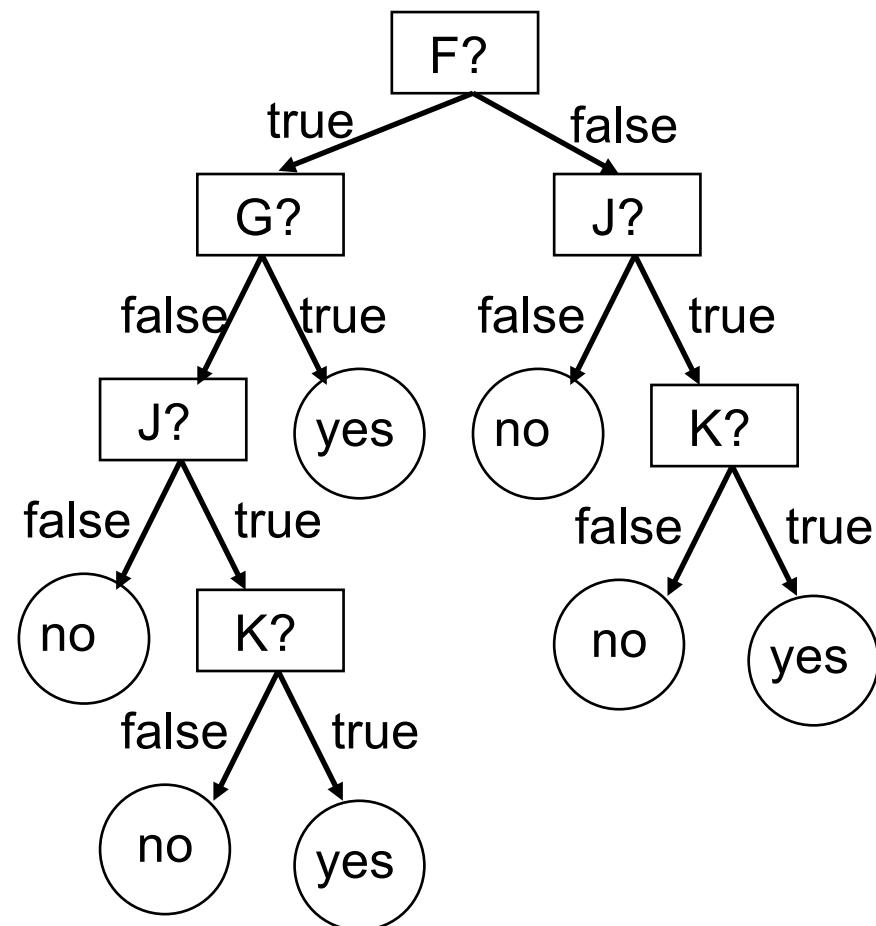
From Tree to Rules

- C4.5 rules slow for large and noisy datasets
- Commercial version C5.0 rules uses a different technique
 - Much faster and a bit more accurate
- C4.5 has two parameters
 - Confidence value (default 25%): lower values incur heavier pruning
 - Minimum number of instances in the two most popular branches (default 2)



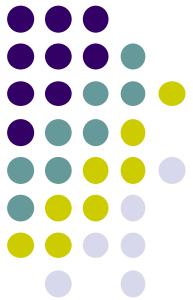
Decision tree learning

From decision tree to rules



The rules are:

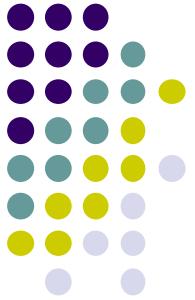
If $(F \cap G)$ then *true*
If $(J \cap K)$ then *true*



Conclusion

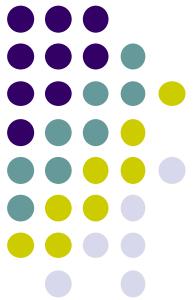
Well adapted when

- Instances are represented by attribute-value pairs
 - Counter-example: Non-rectangular regions
- The target function has discrete output values
 - Counter-example: regression
- The training example may contain errors
- The training data may contain missing values



Conclusion

- The algorithm does not take account for imprecise values
- And if we have continuous values?
Not really a problem, we may split the values into different ranges
- The algorithm does not take account for structured attributes
- Top-down induction of decision trees (TDITDT):
The most extensively studied method of machine learning used in data mining



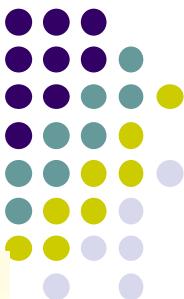
Conclusion

- ID3, algorithm developed by Ross Quinlan
 - Gain ratio just one modification of this basic algorithm
 - → C4.5: deals with numeric attributes, missing values, noisy data
- Different criteria for attribute/test selection rarely make a large difference

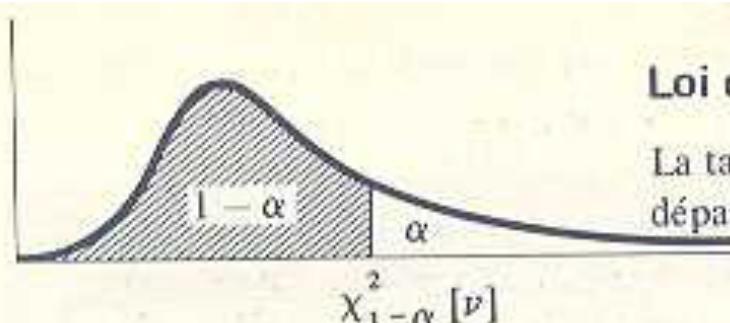


Conclusion

- Different pruning methods mainly change the size of the resulting pruned tree
- C4.5 builds *univariate* decision trees
- Some TDITDT systems can build *multivariate* trees (e.g. CART)
- There are many other attribute selection criteria!
- (But little difference in accuracy of result)



Analysis of Data Assessment



Loi du χ^2 de Pearson

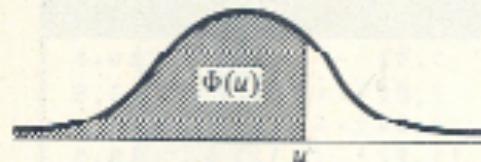
La table donne la limite $\chi_{1-\alpha}^2 [\nu]$ qui a la probabilité α d'être dépassée par la variable aléatoire χ^2 (test unilatéral).

ν : nombre de degrés de liberté

ν	0.999	0.990	0.980	0.950	0.900	0.100	0.050	0.020	0.010	0.001
α	0.0000	0.0002	0.0006	0.0039	0.0158	2.71	3.84	5.41	6.63	10.8
1	0.0020	0.0201	0.0404	0.103	0.211	4.61	5.99	7.82	9.21	13.8
2	0.0243	0.115	0.185	0.352	0.584	6.25	7.81	9.84	11.3	16.3
3	0.0908	0.297	0.429	0.711	1.06	7.78	9.49	11.7	13.3	18.5
4	0.210	0.554	0.752	1.15	1.61	9.24	11.1	13.4	15.1	20.5
5										
6	0.381	0.872	1.13	1.64	2.20	10.6	12.6	15.0	16.8	22.5
7	0.598	1.24	1.56	2.17	2.83	12.0	14.1	16.6	18.5	24.3
8	0.857	1.65	2.03	2.73	3.49	13.4	15.5	18.2	20.1	26.1
9	1.15	2.09	2.53	3.33	4.17	14.7	16.9	19.7	21.7	27.9
10	1.48	2.56	3.06	3.94	4.87	16.0	18.3	21.2	23.2	29.6
11	1.83	3.05	3.61	4.57	5.58	17.3	19.7	22.6	24.7	31.3
12	2.21	3.57	4.18	5.23	6.30	18.5	21.0	24.1	26.2	32.9
13	2.62	4.11	4.77	5.89	7.04	19.8	22.4	25.5	27.7	34.5
14	3.04	4.66	5.37	6.57	7.79	21.1	23.7	26.9	29.1	36.1
15	3.48	5.23	5.98	7.26	8.55	22.3	25.0	28.3	30.6	37.7
16	3.94	5.81	6.61	7.96	9.31	23.5	26.3	29.6	32.0	39.3
17	4.42	6.41	7.26	8.67	10.1	24.8	27.6	31.0	33.4	40.8
18	4.90	7.01	7.91	9.39	10.9	26.0	28.9	32.3	34.8	42.3
19	5.41	7.63	8.57	10.1	11.7	27.2	30.1	33.7	36.2	43.8
20	5.92	8.26	9.24	10.9	12.4	28.4	31.4	35.0	37.6	45.3



Tables of Normal Distribution



2. Fonction de répartition

$$\Phi(u) = P(U < u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^u e^{-\frac{t^2}{2}} dt$$

u	0	1	2	3	4	5	6	7	8	9
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6360	0.6400	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8600	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936