Decision Tree

PROF. JACQUES SAVOY
UNIVERSITY OF NEUCHATEL

Content

Introduction

Tree construction

Gain ratio

Numerical attribute & missing value

Pruning

Credit scoring

Generate rules

Introduction

Simple algorithms and easy to understand (“How-To” manuals).

Attributes-based examples (e.g., IF attribute=this value, THEN …).

Limited to Boolean classifier (to learn a discrete-valued functions).

Need to transform numerical attributes into discrete values.

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 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. Wait estimate: estimated waiting time (0-10, 10-30, 30-60, >60).

From: Russell, S., & Norvig, P. (2016). Artificial Intelligence, A Modern Approach. Prentice Hall, London (UK).

Attribute-based Representations

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

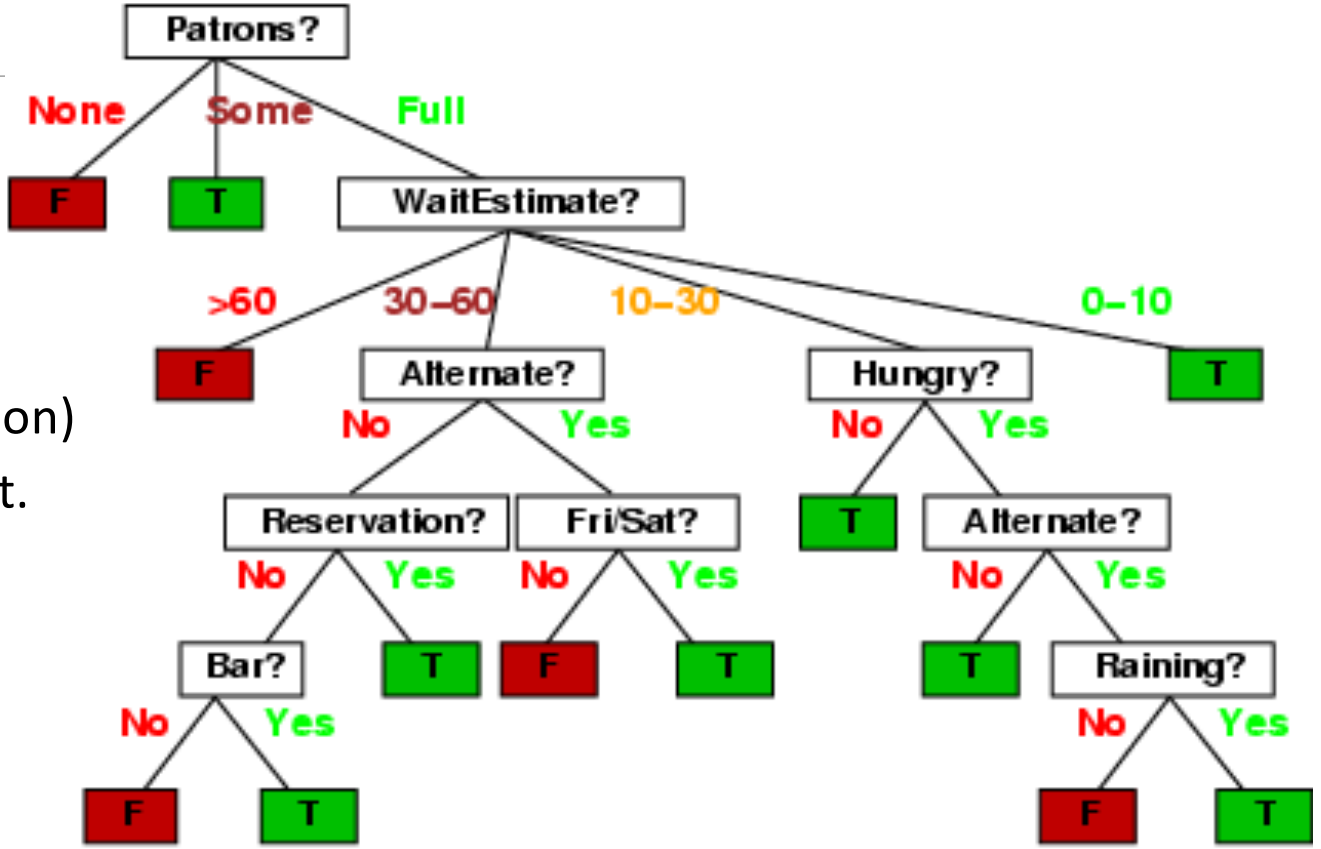
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	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

Example

One possible representation for hypotheses.

Here is my tree (manual construction) for deciding whether to wait or not.

This solution is *consistent*.



Example: Weather Problem

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

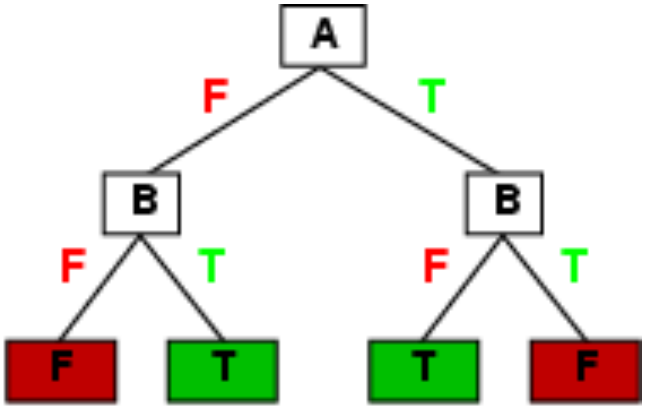
Expressiveness

Decision trees can express any function of the input attributes.

For a Boolean function,

truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



In the restaurant example

$\forall$ s, WillWait(s) $\leftrightarrow$ (P_1(s) v P_2(s) v … v P_n(s))
where P_j(s) (conjunction of test) 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).

Content

Introduction

Tree construction

Gain ratio

Numerical attribute & missing value

Pruning

Credit scoring

Generate 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.
- But the solution is *consistent.*

Is it possible to find a better solution (tree construction)?

Tree Construction

Strategy: Top-down recursive.

Recursive *divide-and-conquer* fashion.

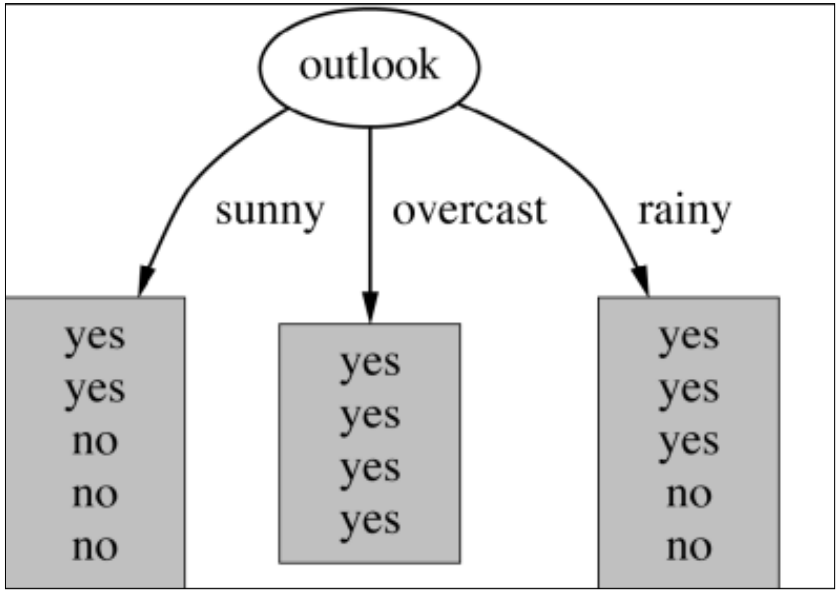
1. First: select attribute for root node.
2. Create branch for each possible attribute value.
3. Split instances into subsets.
   One for each branch extending from the node.
4. Repeat recursively for each branch, using only instances that reach the branch.

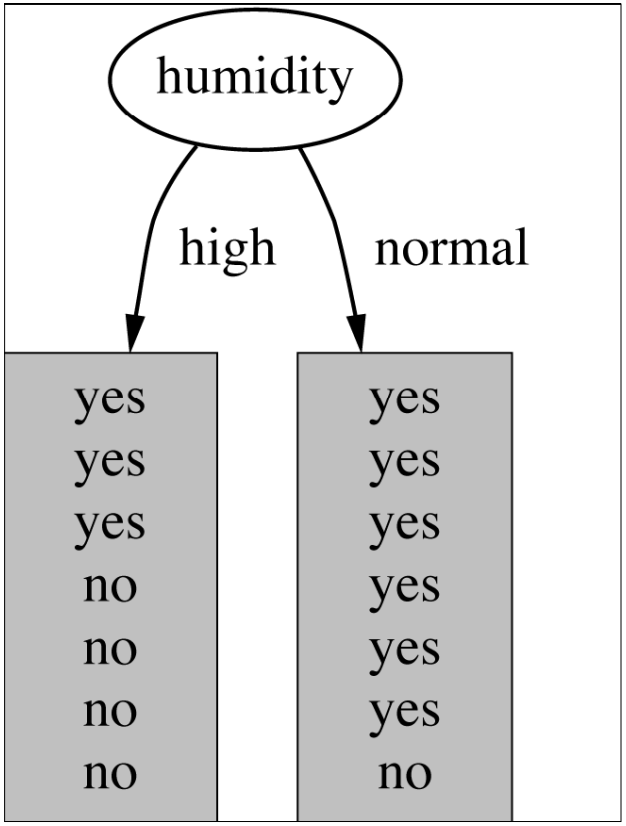
Reclusively: Yes but each time to a smaller set of instances.

Stop when a simple solution is reached: all instances have the same class.

Choosing an Attribute

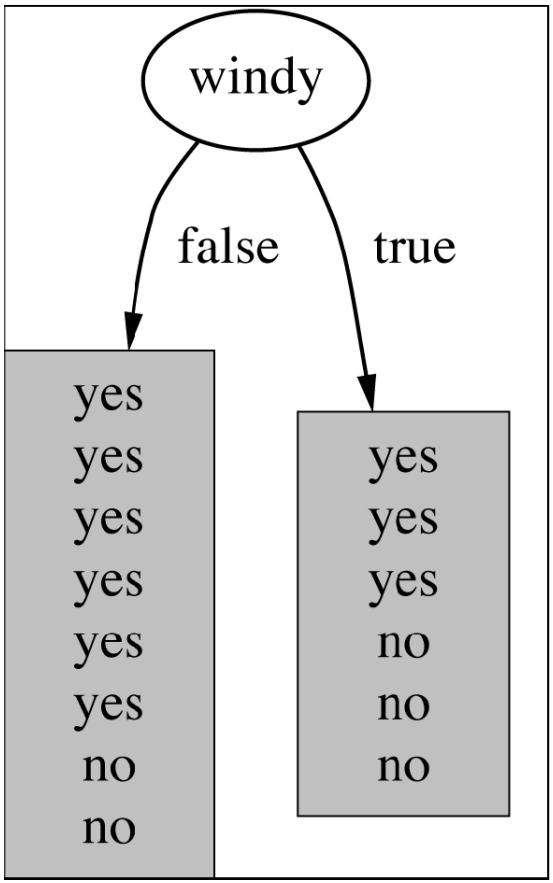
With our weather problem

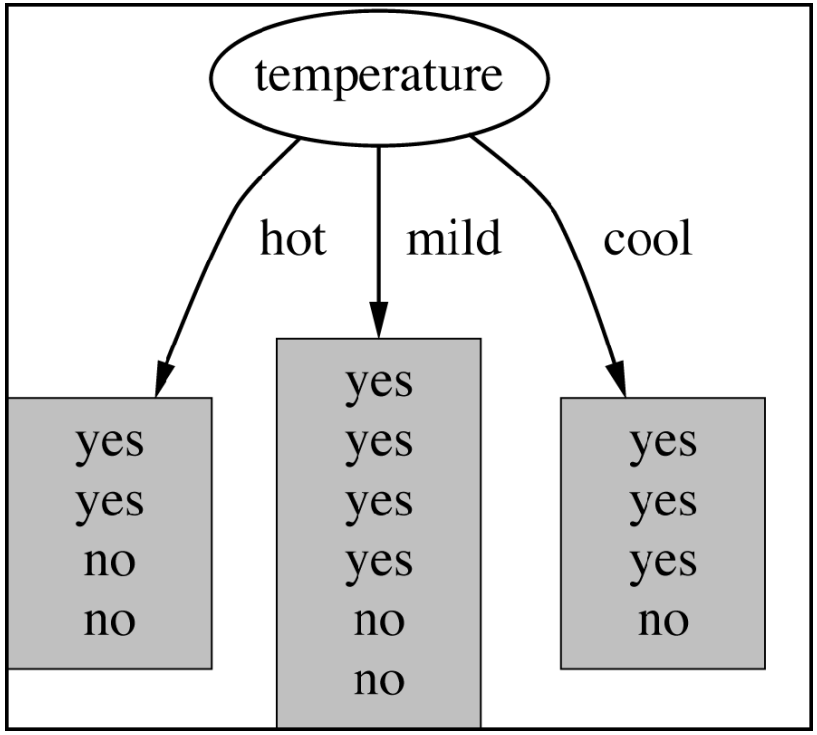




Choosing an Attribute

Other choices





Tree Construction

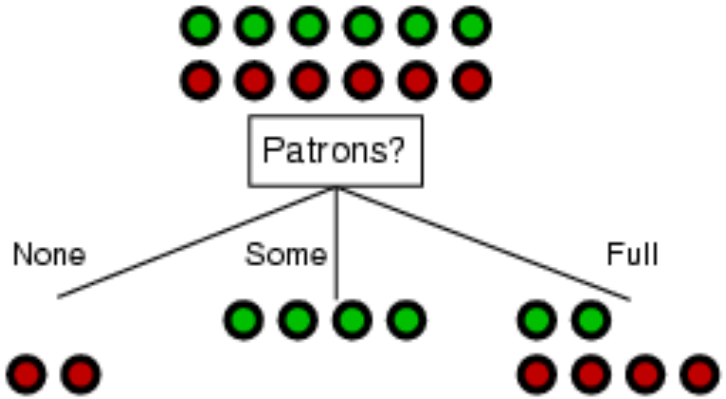
Idea: (recursively) choose “most significant” attribute as root of (sub)tree with simple cases:

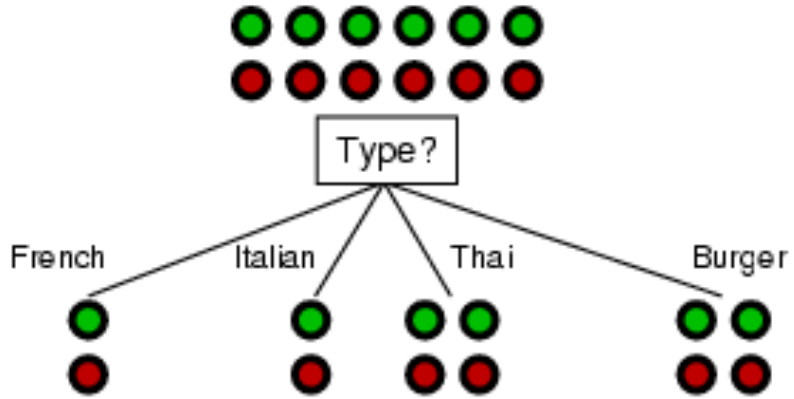
- all examples are positive (or negative): create a leaf with the label 1 (or 0).
- no examples left (no such example):
  return the default value (majority classification of the parent's node).
- no attributes left: if we have both positive and negative → noise (majority vote).

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 ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, 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 a 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):

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

Weighted mean

Example with two messages:

- I(0.5, 0.5) = -0.5·log$_2$(0.5) -0.5·log$_2$(0.5) = -0.5·(-1) -0.5·(-1) = 1
- I(0.8, 0.2) = -0.8·log$_2$(0.8) -0.2·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: over four possible values:

- 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*):

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

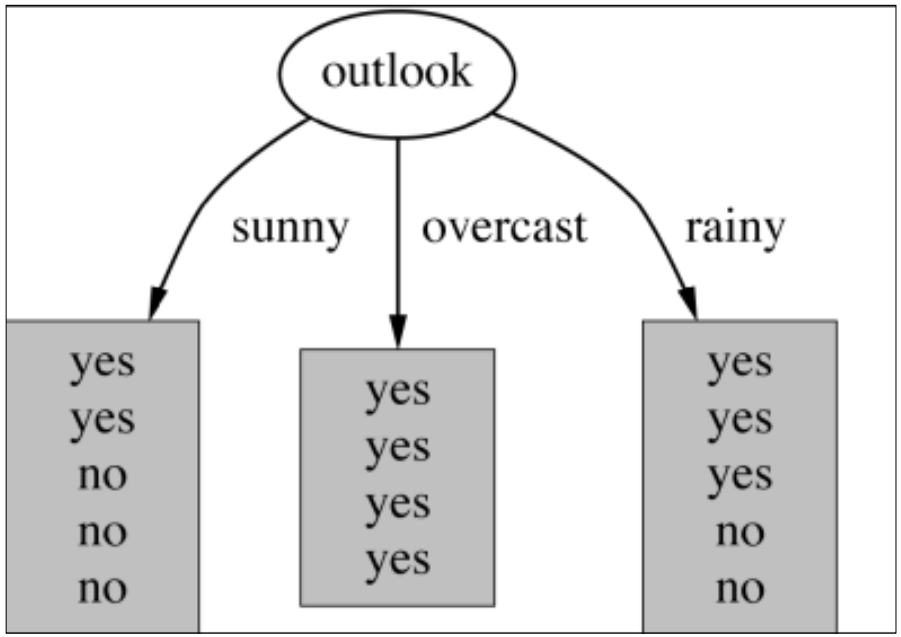
If our training sample (weather) contains 14 cases,
9 positive and 5 negative examples, we have I(9/14, 5/14) = **0.940**

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)

$$Info(\text{sunny, overcast, rainy}) =$$
$$+ \tfrac{5}{14} \cdot 0.971 + \tfrac{4}{14} \cdot 0 + \tfrac{5}{14} \cdot 0.971 = 0.693 \text{ bits}$$

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:

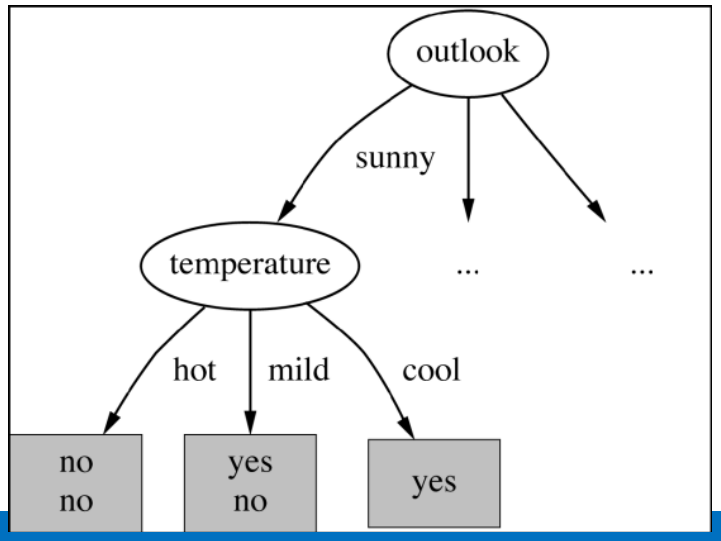
- gain(*Outlook*) = 0.940 – 0.693 = **0.247** bits
- gain(*Temperature*) = 0.029 bits
- gain(*Humidity*) = 0.152 bits
- gain(*Windy*) = 0.048 bits

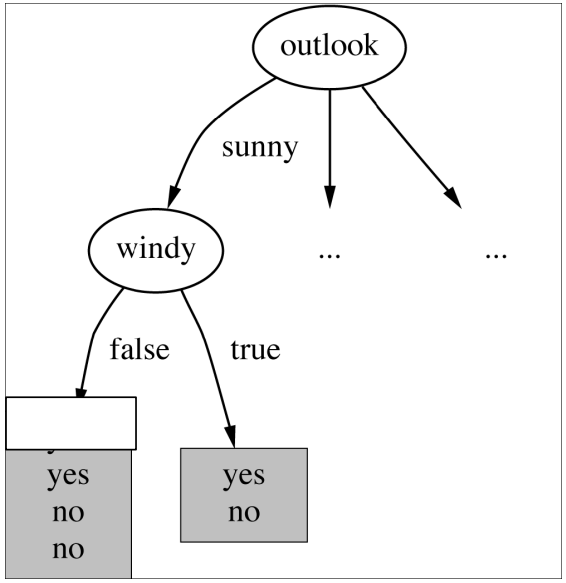
Among all possible attributes, choose the attribute with the largest IG
(*outlook* in our case).

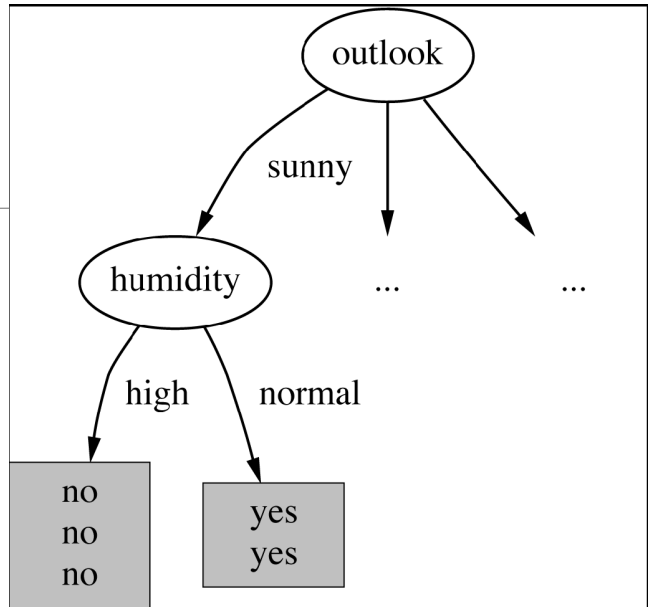
Tree Construction

Continuing the decision tree

- gain(*Temperature*) = 0.571 bits
- gain(*Humidity*) = 0.971 bits
- gain(*Windy*) = 0.020 bits

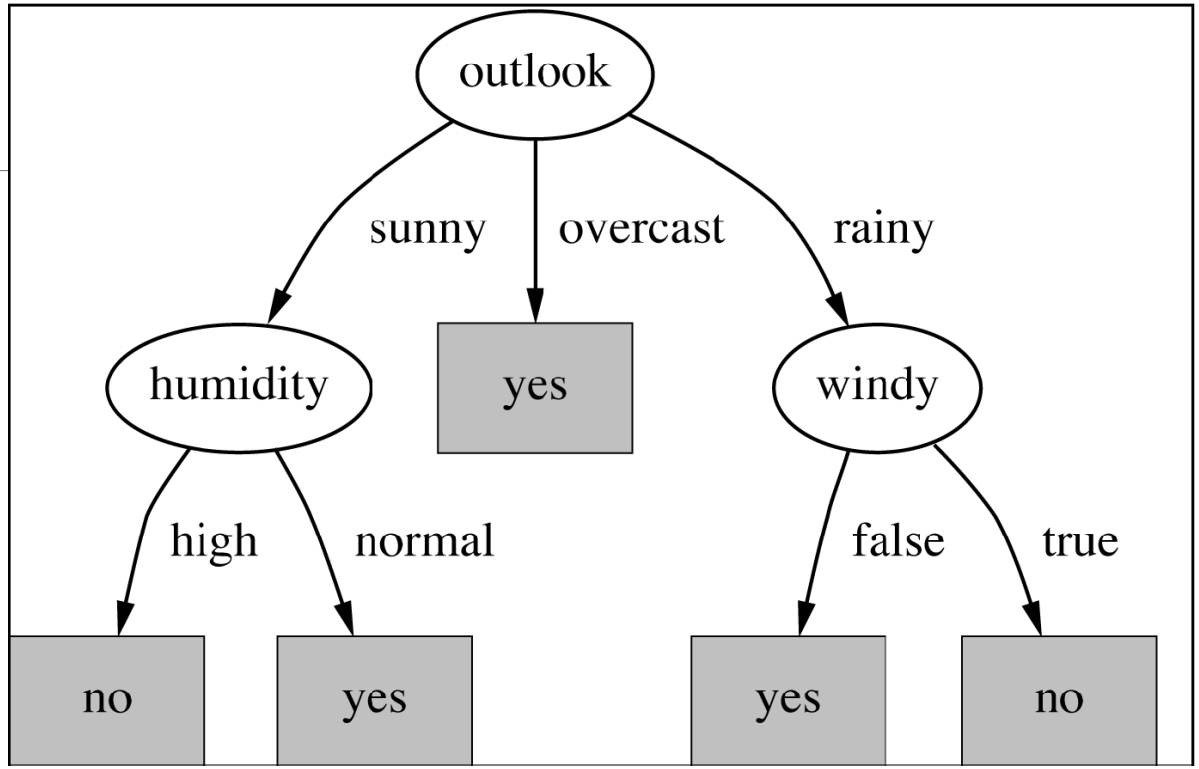






Tree Construction

The final decision tree



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

→ Splitting stops when data can't be splitted any further.

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 → Will wait 1 *false* / 1 *true*

4 Thai → Will wait 2 *false* / 2 *true*

4 Burger → Will wait 2 *false* / 2 *true*

2 Italian → Will wait 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 → Will wait 4 *false* / 2 *true*

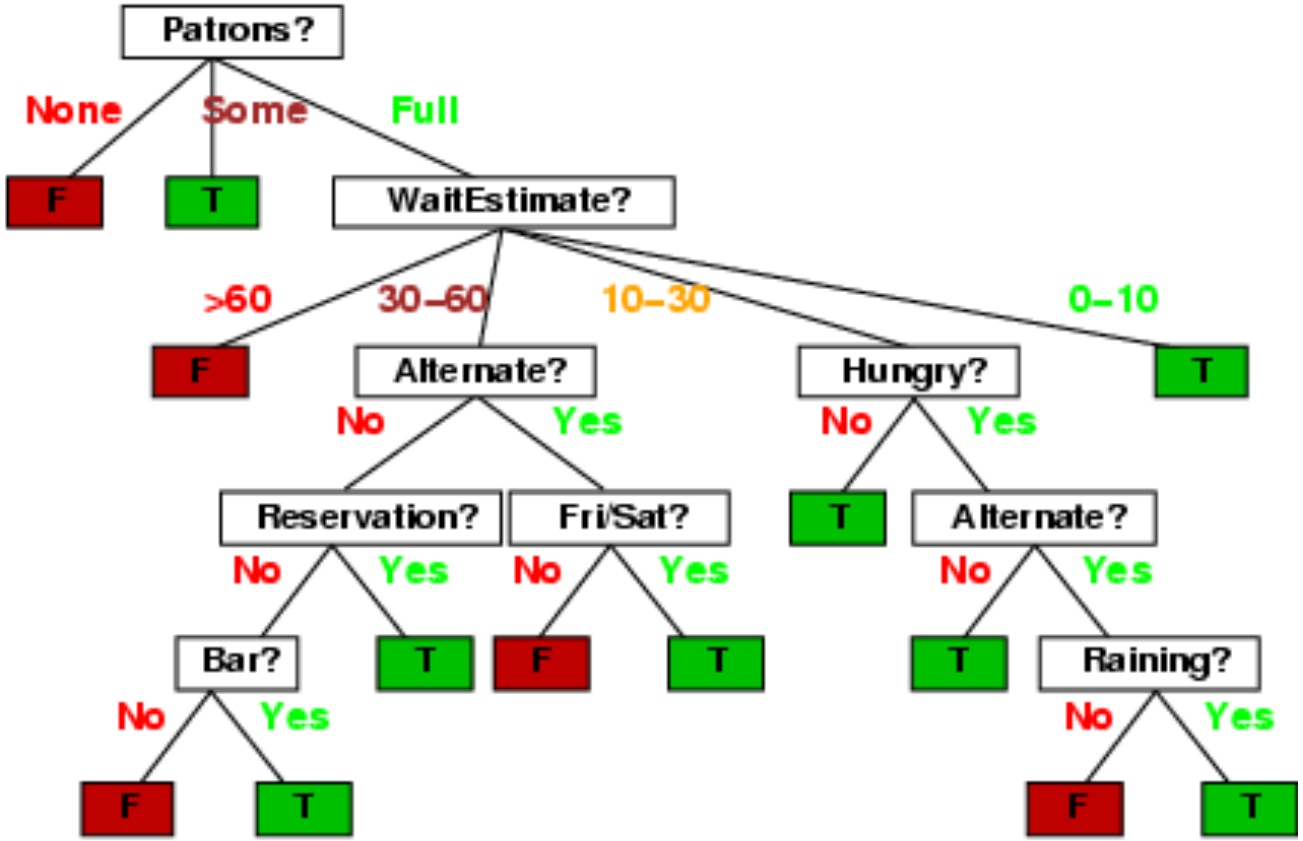
4 Some → Will wait 4 *true*

2 None → Will wait 2 *false*

$$Info(Patrons) = \frac{2}{12} \cdot I(0,1) + \frac{4}{12} \cdot I(1,0) + \frac{6}{12} \cdot I(2/6, 4/6) = \frac{2}{12} \cdot 0 + \frac{4}{12} \cdot 0 + \frac{6}{12} \cdot [-0.33 \cdot \log(0.33) - 0.66 \cdot \log(0.66)] = 0.459$$

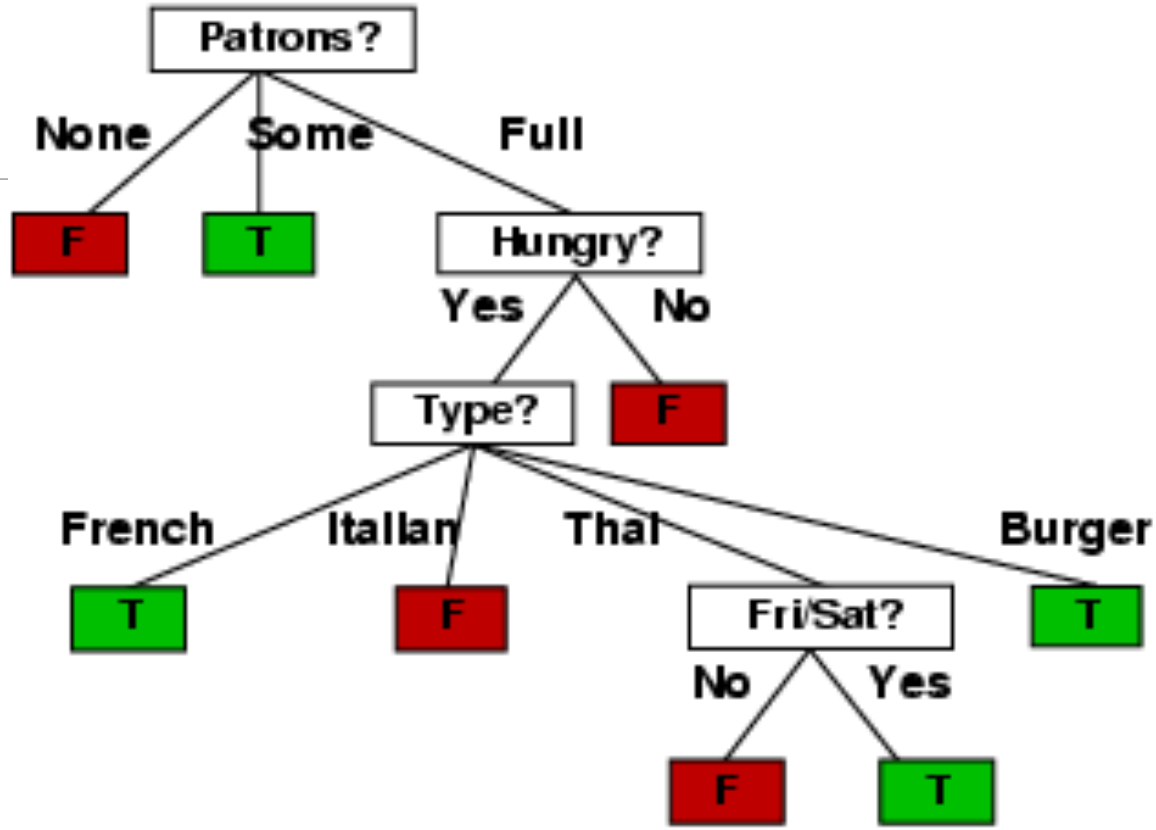
IG(Patrons) = 1 - 0.459 = 0.541 reduces the uncertainty.

Example: Manual Construction



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.

Content

Introduction

Tree construction

Gain ratio

Numerical attribute & missing value

Pruning

Credit scoring

Generate rules

Limit of the Information Gain

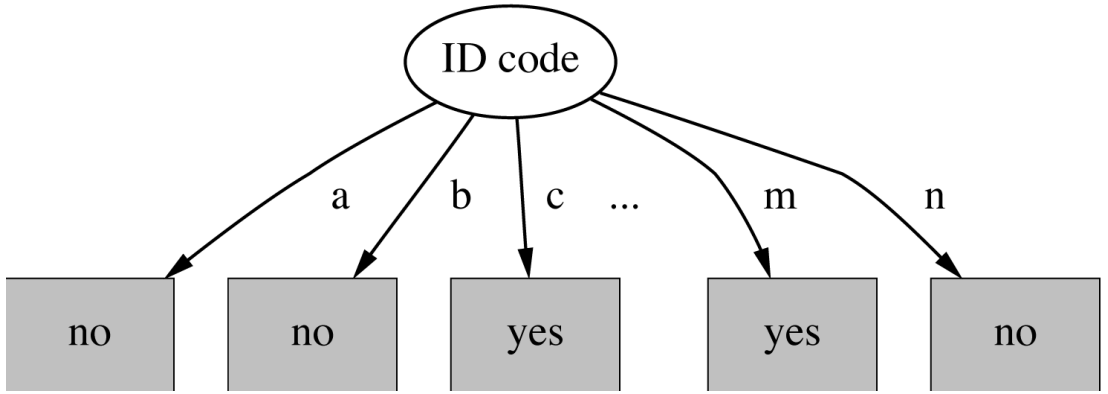
Problematic: attributes with a large number of values (extreme case: ID code).

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 non-optimal for prediction)

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



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 (weather dataset)
I[1/14,1/14,..., 1/14] =14 × [−1/14 × log(1/14)] = 3.807 bits.

Value of attribute decreases as intrinsic information (or split info) gets larger.

Definition of *gain ratio*:

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

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

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.

Content

Introduction

Tree construction

Gain ratio

Numerical attribute & missing value

Pruning

Credit scoring

Generate rules

Numeric Attribute

Standard method: binary splits.

- e.g., temperature < 24

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:

18	18	20	21	21	22	22	22	24	24	27	27	28	29
Y	Y	Y	Y	Y	Y	No	No	No	Y	No	No	Y	Y

- e.g. temperature < 22.5: *yes*/6, *no*/2
  temperature > 22.5: *yes*/3, *no*/3
- Info([6,2],[3,3]) = 8/14 I([6,2]) + 6/14 I([3,3]) = 0.811 bits

Place split points halfway between values.

Can evaluate all split points in one pass! (Binary split).

Missing Values

When generating the tree:
Consider missing value as one possible value.

When testing:
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.
- Merge decision using weights (e.g., majority).

Content

Introduction

Tree construction

Gain ratio

Numerical attribute & missing value

Pruning

Credit scoring

Generate rules

Pruning

Prevent overfitting to noise in the data.

Thus: “Prune” the decision tree.

Two strategies (with various variants):

- *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	15	3	18
value c_2	34	3	37
value c_3	15	36	51
	64	42	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, to compare with limit values in the table (e.g., with *dof* = 1 and a = 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	10.9	7.1	18
value c_2	22.3	14.7	37
value c_3	30.8	20.2	51
	64	42	106

Chi-square computed: 39.77

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

Decision: 39.77 > 15.086, reject Ho, there is a relationship between the feature C and the target attribute P or N.

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 ⊗ 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*

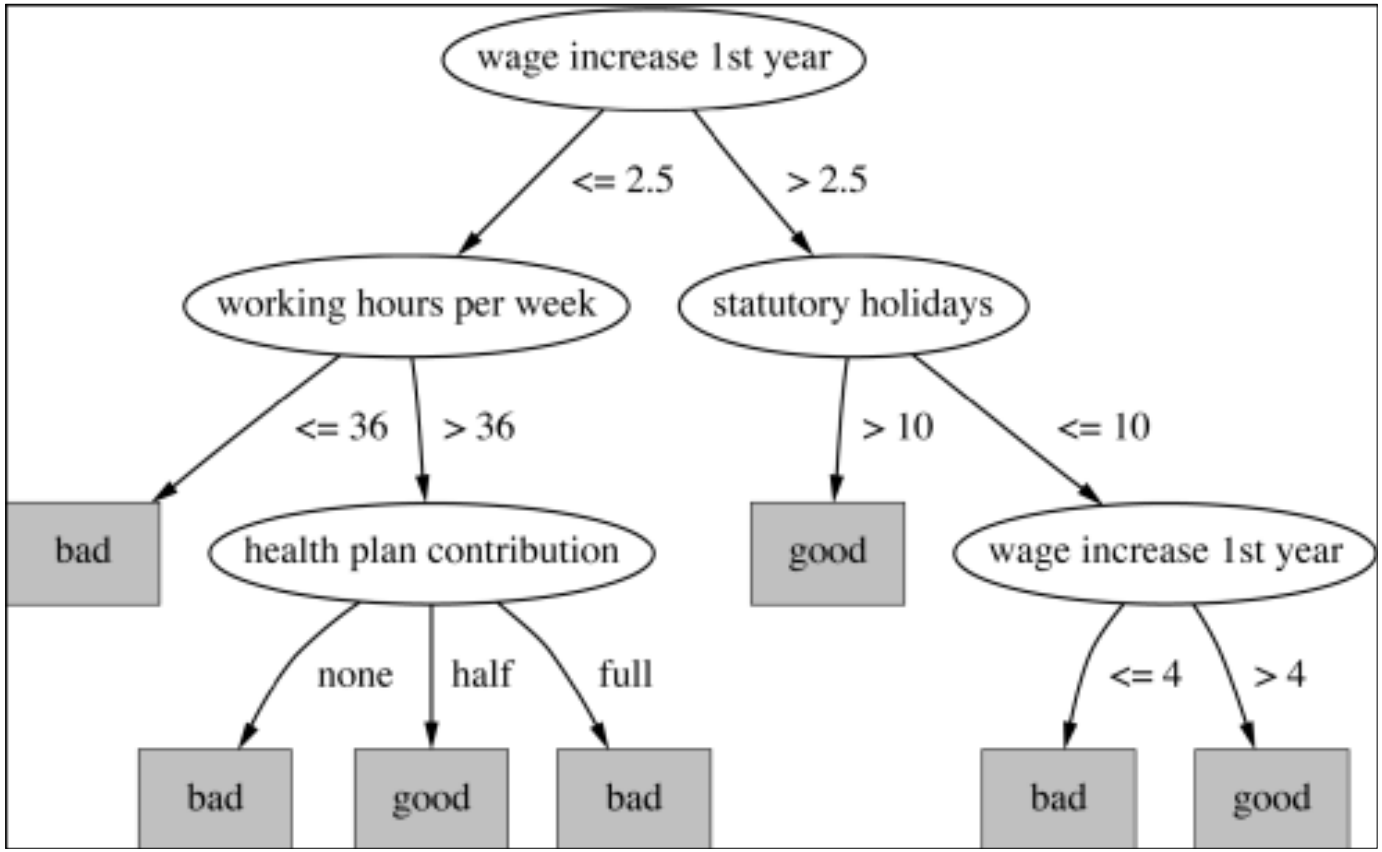
Possible strategies:

- error estimation
- significance testing
- MDL principle

Post-Pruning

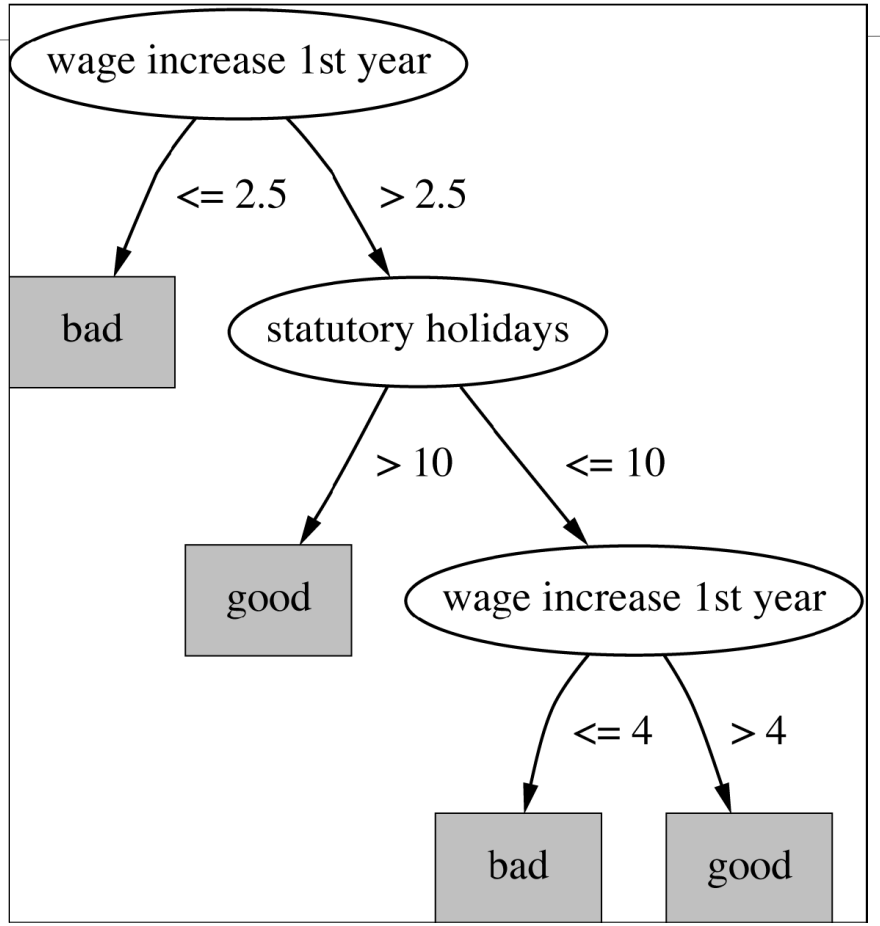
Subtree replacement (*bottom-up)*

Consider replacing a tree only after considering all its subtrees.



Post-Pruning

The final pruned tree



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

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

  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* = 0.675 (from the Gaussian 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 denoted *e* of the true error rate could be (with c=25% and thus *z* = 0.675):

$$e = \frac{\left(0.33 + \frac{0.675^2}{12} + 0.675 \cdot \sqrt{\frac{0.33}{6} - \frac{0.33^2}{6} + \frac{0.675^2}{144}}\right)}{1 + \frac{0.675^2}{6}}$$

$$= \frac{\left(0.33 + 0.0379 + 0.675 \cdot \sqrt{0.0555 - 0.01852 + 0.0047}\right)}{1 + 0.07935} = 0.4708$$

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 = 0.447.

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=0.447 (if we do not split further this node)
while the combined error rate for the children is e=0.506.

C4.5's Method

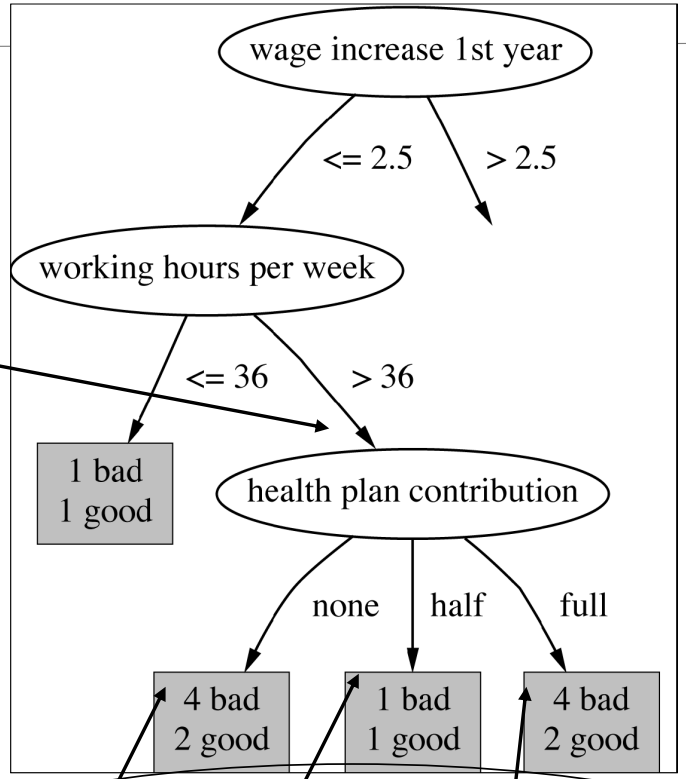
Example

f=5/14
e=0.447

e > 0.447 so prune!

Combined 6:2:6
gives e=0.506

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

Content

Introduction

Tree construction

Gain ratio

Numerical attribute & missing value

Pruning

Credit scoring

Generate rules

Credit Scoring

Classical example in machine learning.

A dataset extracted form the UCI website.

- 1,000 observations with 17 attributes.
- Features to obtain a credit from a bank.
- decision: default (yes/no).
- Bank balance given in DM (Deutsche mark)
- For the required loan, we have some information (duration, purpose, amount, …)

Some preprocessing has been done.

Statlog (German Credit Data) Data Set

Download: Data Folder, Data Set Description

Abstract: This dataset classifies people described by a set of attributes as good or bad credit risks. Comes in two formats (one all numeric). Also comes with a cost matrix

Data Set Characteristics:	Multivariate	**Number of Instances:**	1000	**Area:**	Financial
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	20	**Date Donated**	1994-11-17
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	328188

Source:

Professor Dr. Hans Hofmann
Institut f"ur Statistik und "Okonometrie
Universit"at Hamburg
FB Wirtschaftswissenschaften
Von-Melle-Park 5
2000 Hamburg 13

Credit Scoring

Import the dataset Risky Bank Loans

```
> credit <- read.csv("./Data/credit.csv")
> str(credit)
'data.frame':    1000 obs. of  17 variables:
$ checking_balance    : Factor w/ 4 levels "< 0 DM","> 200 DM",..: 1 3 4 1...
$ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30...
$ credit_history      : Factor w/ 5 levels "critical","good",..: 1 2 1 2 4...
$ purpose             : Factor w/ 6 levels "business","car",..: 5 5 4 5 2...
$ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948...
$ savings_balance     : Factor w/ 5 levels "< 100 DM","> 1000 DM",..: 5 1 1...
$ employment_duration : Factor w/ 5 levels "< 1 year","> 7 years",..: 2 3 4...
$ percent_of_income   : int  4 2 2 2 3 2 3 2 2 4 ...
$ years_at_residence  : int  4 2 3 4 4 4 4 2 4 2 ...
```

Credit Scoring

```
# Look at two characteristics of the customer
> table(credit$checking_balance)
  < 0 DM   > 200 DM   1 - 200 DM    unknown
     274         63         269         394
> table(credit$savings_balance)
   < 100 DM     > 1000 DM    100 - 500 DM   500 - 1000 DM   unknown
      603            48           103            63            183
# View two characteristics of the requested loan
> summary(credit$months_loan_duration)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4.0    12.0    18.0    20.9    24.0    72.0
> summary(credit$amount)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    250    1366    2320    3271    3972   18420
```

Credit Scoring

Distribution of the decision

```
 > table(credit$default)
   no  yes
  700  300
```

Generate the training (900 observations) and test set (100 instances)

```
 > set.seed(57891)
> runif(4)
[1] 0.1948440 0.3772854 0.7194458 0.6547758
> set.seed(57891); order(runif(4))
[1] 1 2 4 3
```

```
 > set.seed(57891); credit.rand <- credit[order(runif(1000)), ]
```

Credit Scoring

Split the data into the training and test sets

```
> credit.train <- credit.rand[1:900,]
> credit.test  <- credit.rand[901:1000,]
```

Check the proportion of decision variable into the two sets

```
> prop.table(table(credit.train$default))
       no        yes
0.7055556  0.2944444

 > prop.table(table(credit.test$default))
  no   yes
 0.65  0.35
```

Similar proportions in both sets

Credit Scoring

Build the simplest decision tree

```
> library(C50)
> credit.model <- C5.0(credit.train[-17], credit.train$default)
```

Display simple facts about the tree

```
> credit.model

  Call:
  C5.0.default(x = credit.train[-17], y = credit.train$default)

 Classification Tree

 Number of samples: 900 Number of predictors: 16
 Tree size: 60
```

Credit Scoring

Display simple facts about the tree

```
> summary(credit.model)
  Decision tree:
checking_balance in {> 200 DM,unknown}: no (409/52)
checking_balance in {< 0 DM,1 - 200 DM}:
:...months_loan_duration > 22:
    :...savings_balance in {> 1000 DM,unknown}:
    :   :...checking_balance = < 0 DM:
    :   :   :...existing_loans_count <= 1: yes (12/3)
    :   :   :   existing_loans_count > 1: no (4)
    :   :   checking_balance = 1 - 200 DM:
    :   :   :...percent_of_income > 1: no (17)
    :   :       percent_of_income <= 1:
    :   :       :...amount <= 9629: no (2)
    :   :           amount > 9629: yes (2)
```

Credit Scoring

```
Evaluation on training data (900 cases):
        Decision Tree
      ----------------
      Size      Errors
        58  118 (13.1%)   <<

       (a)   (b)    <- classified as
      ----  ----
       616    19    (a): class no
        99   166    (b): class yes
```

The accuracy rate = (616 + 166) / 900 = 782 / 900 = 0.8689

Is this good?

Compared to an human expert?

Credit Scoring

Evaluation on the remaining instances

```
> credit.pred <- predict(credit.model, credit.test)
> credit.pred
[1] yes no  no  no  yes no  no  no  yes yes yes no  no  no
[15] no  no  no  no  no  no  no  no  no  no  yes no  no  no …
 Levels: no yes
> credit.test$default
[1] yes no  no  yes yes no  no  no  yes no  yes no  yes no
[15] no  no  no  no  yes no  no  no  no  no  yes yes no  yes…
Levels: no yes
```

Simple estimation of the accuracy rate

```
> sum (credit.pred == credit.test$default) / 100.0
[1] 0.73
```

Credit Scoring

Summary of the performance (inspect the two sources of errors)

```
> table(credit.pred, credit.test$default)

credit.pred  no  yes
       no    62   24
       yes   3    11
```

Correct decision for 62 + 11 = 73 cases.

Incorrect decisions:

a) 24 times: Prediction="no" (will not default), but the applicant will default!

b) 3 times: Prediction="yes" (will default), but the applicant will not

Content

Introduction

Tree construction

Gain ratio

With R

Numerical attribute & missing value

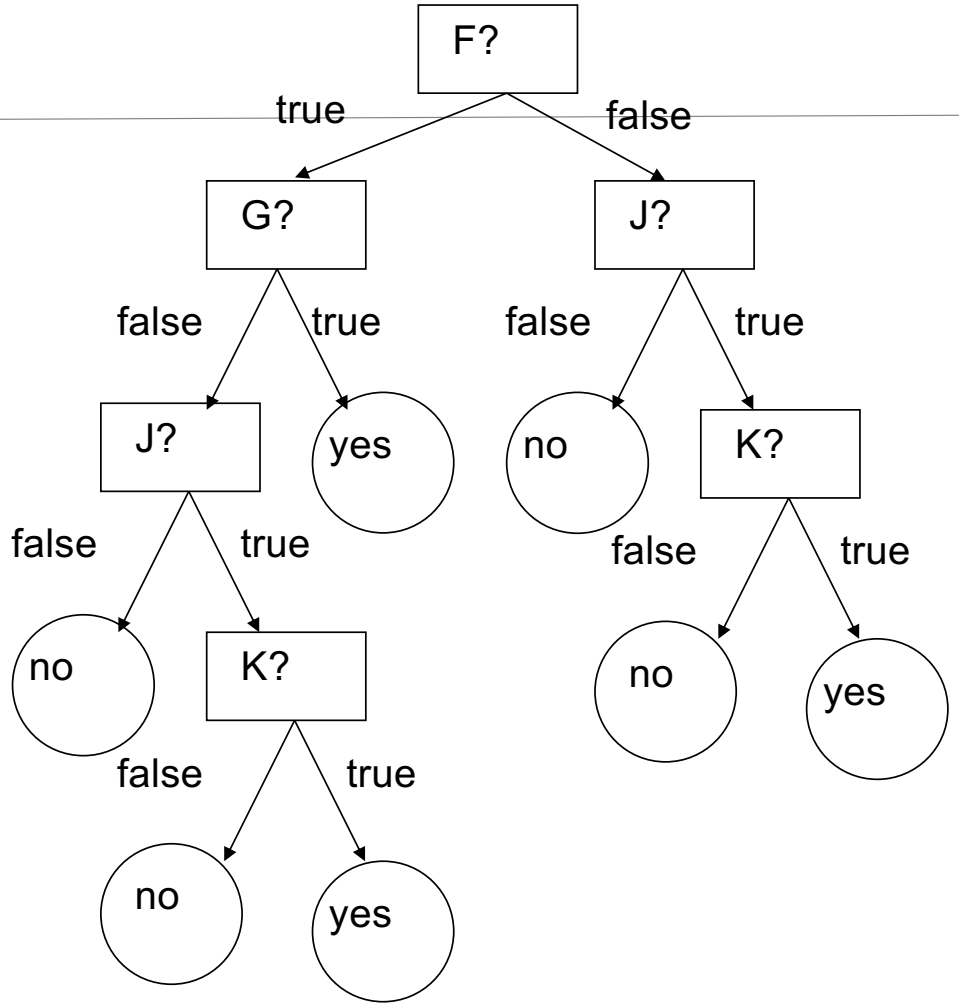
Pruning

Credit scoring

Generate rules

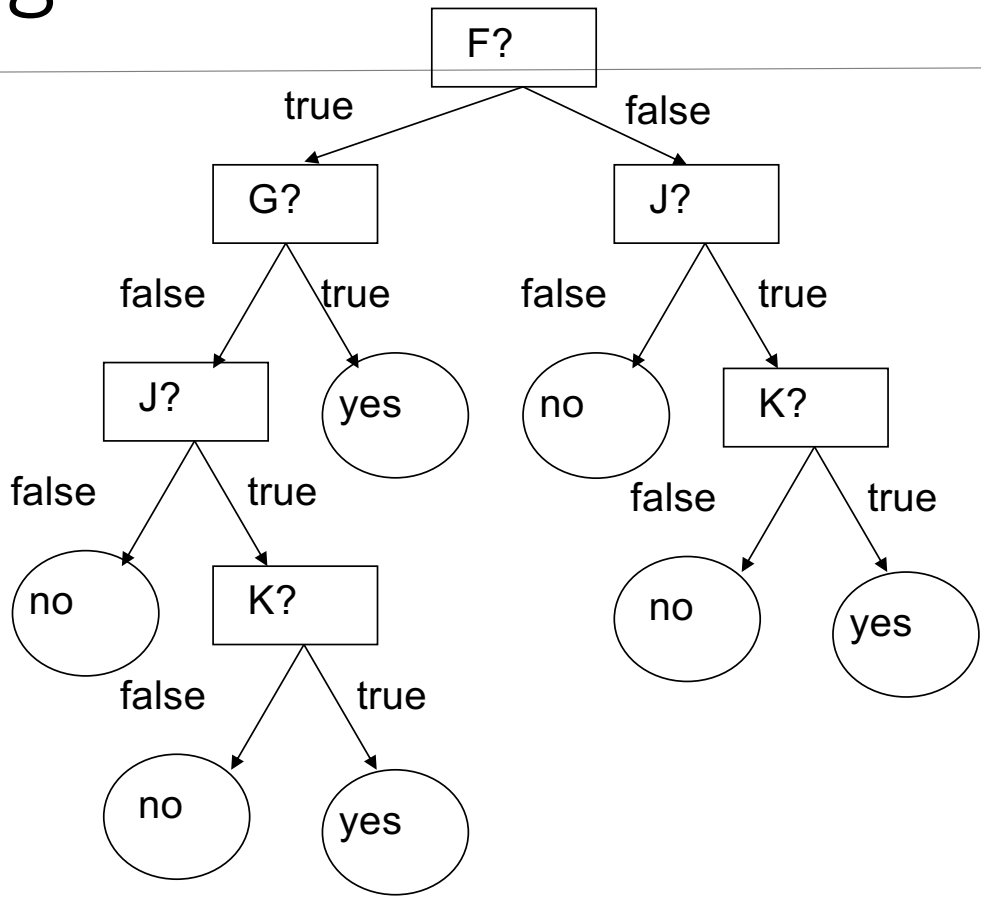
From Tree to Rules

Simple way: one rule for each leaf.



Decision tree learning

From decision tree to rules



The rules are:

If (F ∩ -G ∩ -J) then *false*
If (F ∩ -G ∩ J ∩ -K) then *false*
If (F ∩ -G ∩ J ∩ K) then *true*
If (F ∩ 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 ∩ -G ∩ J ∩ K) then *true*
If (J ∩ K) then *true*
If (J ∩ -G ∩ F ∩ K) then *true*
…

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:

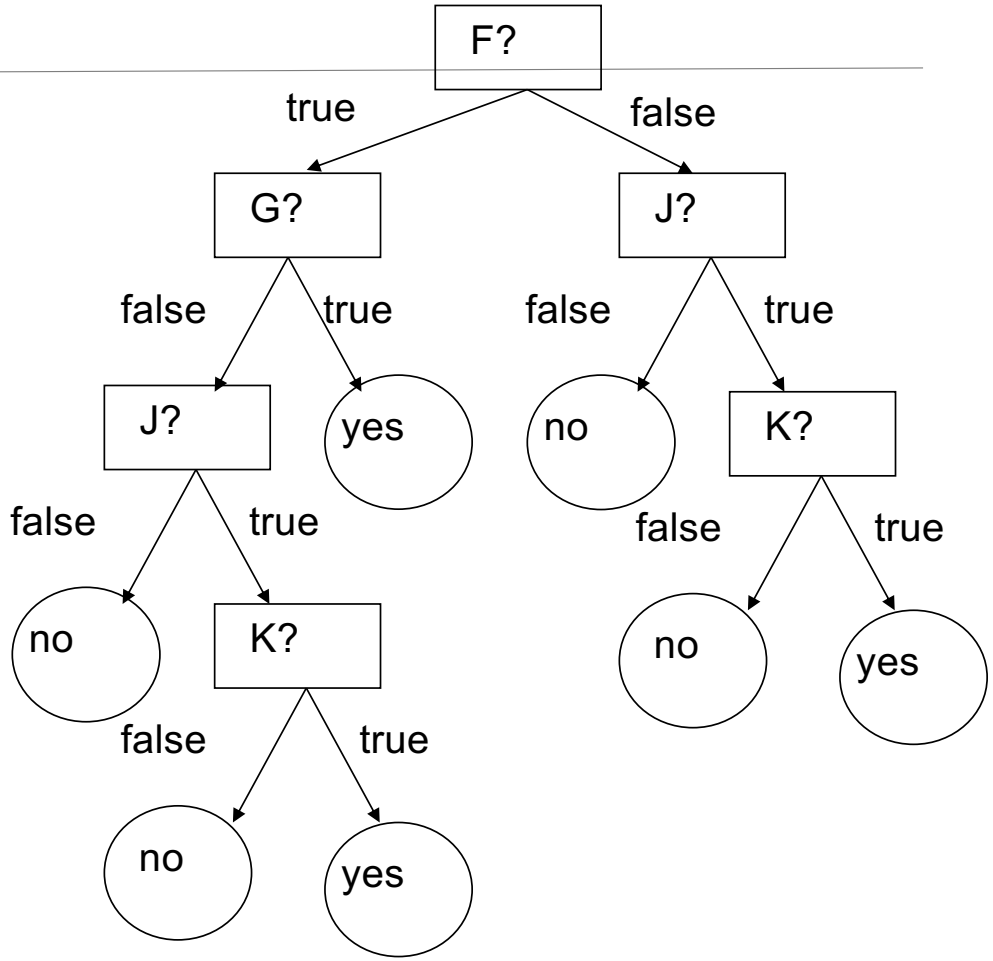
- Confidence value (default 25%): lower values incur heavier pruning.
- Minimum number of instances in the two most popular branches (default: 2).

Decision tree learning

From decision tree to rules

The rules are:

If (F ∩ G) then *true*
If (J ∩ K) then *true*



Conclusion

Instances are represented by attribute-value pairs.

- Counter-example: Non-rectangular regions.

The target function has discrete output values.

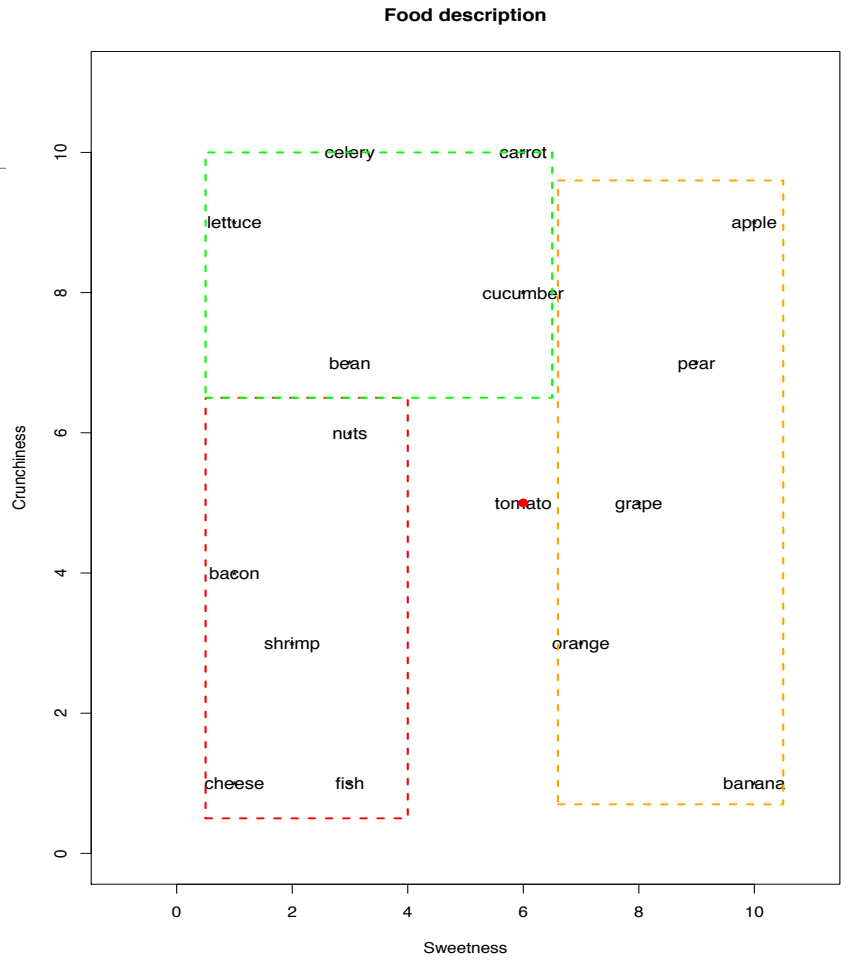
- Counter-example: regression.
  Can be done using CART.

Attributes could be nominal or numerical.

The training data may contain missing values.

Examples may contain errors.

Use as feature selection (see the idea with bagging).



Conclusion

The result is clearly interpretable (humans can understand decisions).

Generate compact tree (number of nodes) after pruning.

Can be use to select the most important features (may not use all attributes).

Fast at testing time (O(tree depth)).

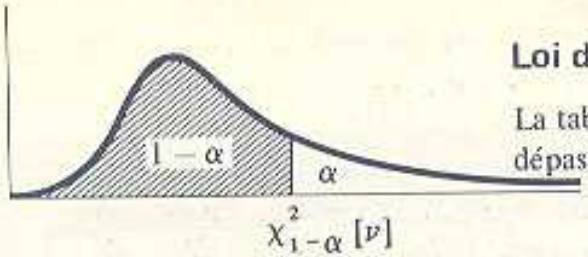
The algorithm does not take account for imprecise values.

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.

But the solution is greedy (may not find the optimal tree).

Chi-square distribution



Loi du χ^2 de Pearson

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

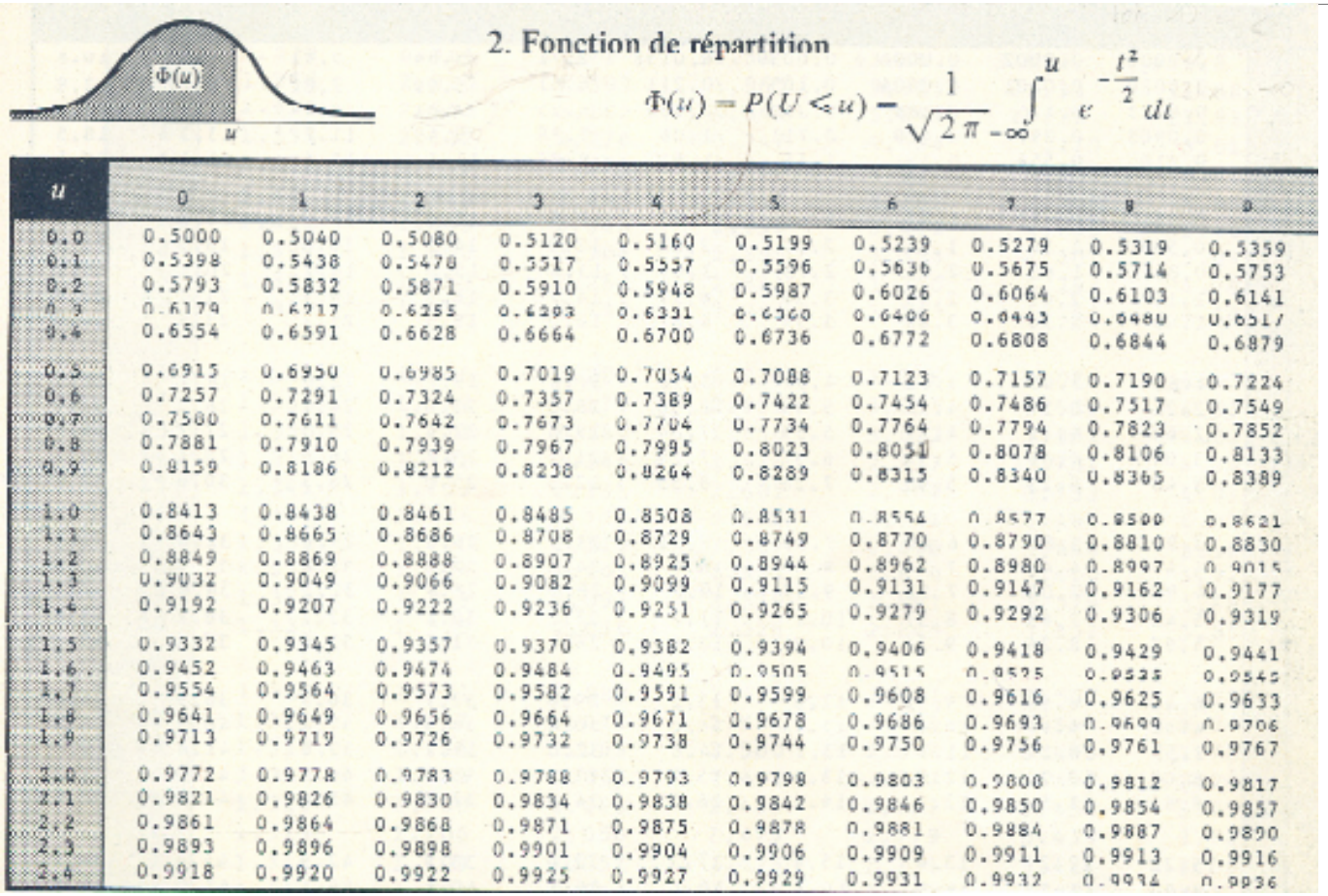
ν : nombre de degrés de liberté

ν \ α	0.999	0.990	0.980	0.950	0.900	0.100	0.050	0.020	0.010	0.001
1	0.0000	0.0002	0.0006	0.0039	0.0158	2.71	3.84	5.41	6.63	10.8
2	0.0020	0.0201	0.0404	0.103	0.211	4.61	5.99	7.82	9.21	13.8
3	0.0243	0.115	0.185	0.352	0.584	6.25	7.81	9.84	11.3	16.3
4	0.0908	0.297	0.429	0.711	1.06	7.78	9.49	11.7	13.3	18.5
5	0.210	0.554	0.752	1.15	1.61	9.24	11.1	13.4	15.1	20.5
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

Chi-square distribution

ν \ α	0.999	0.990	0.980	0.950	0.900	0.100	0.050	0.020	0.010	0.001
21	6.45	8.90	9.91	11.6	13.2	29.6	32.7	36.3	38.9	46.8
22	6.98	9.54	10.6	12.3	14.0	30.8	33.9	37.7	40.3	48.3
23	7.53	10.2	11.3	13.1	14.8	32.0	35.2	39.0	41.6	49.7
24	8.08	10.9	12.0	13.8	15.7	33.2	36.4	40.3	43.0	51.2
25	8.65	11.5	12.7	14.6	16.5	34.4	37.7	41.6	44.3	52.6
26	9.22	12.2	13.4	15.4	17.3	35.6	38.9	42.9	45.6	54.1
27	9.80	12.9	14.1	16.2	18.1	36.7	40.1	44.1	47.0	55.5
28	10.4	13.6	14.8	16.9	18.9	37.9	41.3	45.4	48.3	56.9
29	11.0	14.3	15.6	17.7	19.8	39.1	42.6	46.7	49.6	58.3
30	11.6	15.0	16.3	18.5	20.6	40.3	43.8	48.0	50.9	59.7
35	14.7	18.5	20.0	22.5	24.8	46.1	49.8	54.2	57.3	66.6
40	17.9	22.2	23.8	26.5	29.1	51.8	55.8	60.4	63.7	73.4
45	21.3	25.9	27.7	30.6	33.4	57.5	61.7	66.6	70.0	80.1
50	24.7	29.7	31.7	34.8	37.7	63.2	67.5	72.6	76.2	86.7
55	28.2	33.6	35.7	39.0	42.1	68.8	73.3	78.6	82.3	93.2
60	31.7	37.5	39.7	43.2	46.5	74.4	79.1	84.6	88.4	99.6
70	39.0	45.4	47.9	51.7	55.3	85.5	90.5	96.4	100.4	112.3
80	46.5	53.5	56.2	60.4	64.3	96.6	101.9	108.1	112.3	124.8
90	54.2	61.8	64.6	69.1	73.3	107.6	113.1	119.6	124.1	137.2
100	61.9	70.1	73.1	77.9	82.4	118.5	124.3	131.1	135.8	149.4

Tables of Normal Distribution



2. Fonction de répartition

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

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	0,6368	0,6406	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	0,8599	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

Tables of Normal Distribution

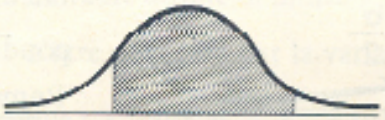
We use standard normal table to find the area under normal curve (random variable Z).

u	$\Phi(u)$	u	$\Phi(u)$	u	$\Phi(u)$	u	$\Phi(u)$	u	$\Phi(u)$
2.50	0.993790	2.80	0.997445	3.10	0.999032	3.40	0.999663	3.70	0.999892
2.52	0.994132	2.82	0.997599	3.12	0.999096	3.42	0.999687	3.72	0.999900
2.54	0.994457	2.84	0.997744	3.14	0.999155	3.44	0.999709	3.74	0.999908
2.56	0.994766	2.86	0.997882	3.16	0.999211	3.46	0.999730	3.76	0.999915
2.58	0.995060	2.88	0.998012	3.18	0.999264	3.48	0.999749	3.78	0.999922
2.60	0.995339	2.90	0.998134	3.20	0.999313	3.50	0.999767	3.80	0.999928
2.62	0.995604	2.92	0.998250	3.22	0.999359	3.52	0.999784	3.82	0.999933
2.64	0.995855	2.94	0.998359	3.24	0.999402	3.54	0.999800	3.84	0.999938
2.66	0.996093	2.96	0.998462	3.26	0.999443	3.56	0.999815	3.86	0.999943
2.68	0.996319	2.98	0.998559	3.28	0.999481	3.58	0.999828	3.88	0.999948
2.70	0.996533	3.00	0.998650	3.30	0.999517	3.60	0.999841	3.90	0.999952
2.72	0.996736	3.02	0.998736	3.32	0.999550	3.62	0.999853	3.92	0.999956
2.74	0.996928	3.04	0.998817	3.34	0.999581	3.64	0.999864	3.94	0.999959
2.76	0.997110	3.06	0.998893	3.36	0.999610	3.66	0.999874	3.96	0.999963
2.78	0.997282	3.08	0.998965	3.38	0.999638	3.68	0.999883	3.98	0.999966

N.B. Si u est négatif, prendre le complément à l'unité de la valeur lue dans la table.

Exemple : $\Phi(-0{,}64) = 1 - \Phi(0{,}64) = 1 - 0{,}7389 = 0{,}2611$

$P(-u < U \leq u) = 2\,\Phi(u) - 1$



$P(u_1 < U \leq u_2) = \Phi(u_2) - \Phi(u_1)$