Decision Trees

Outline

- Top-Down Decision Tree Construction
- Choosing the Splitting Attribute
- Information Gain and Gain Ratio

DECISION TREE

- An internal node is a test on an attribute.
- A branch represents an outcome of the test, e.g., Color=red.
- A leaf node represents a class label or class label distribution.
- At each node, one attribute is chosen to split training examples into distinct classes as much as possible
- A new case is classified by following a matching path to a leaf node.

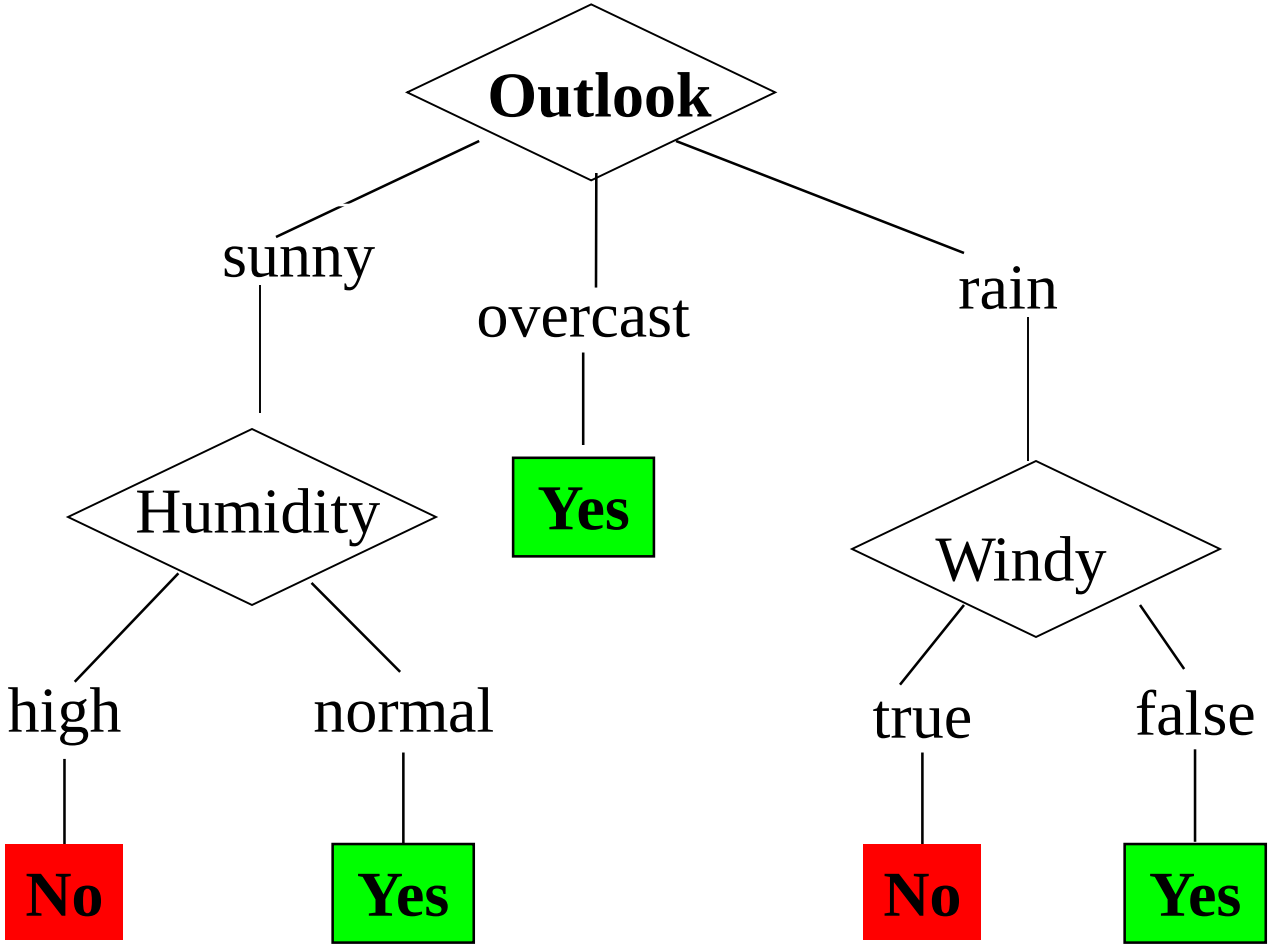
Weather Data: Play or not Play?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Note:
Outlook is the Forecast,
no relation to Microsoft email program

Input

Example Tree for "Play?"



Output

Building Decision Tree

- Top-down tree construction
  - At start, all training examples are at the root.
  - Partition the examples recursively by choosing one attribute each time.
- Bottom-up tree pruning
  - Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases.

Choosing the Splitting Attribute

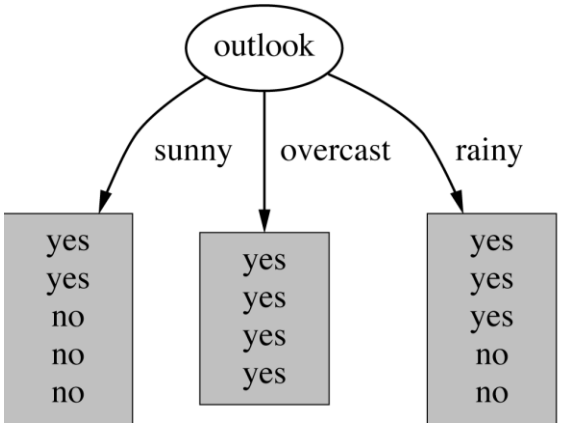
- At each node, available attributes are evaluated on the basis of separating the classes of the training examples. A Goodness function is used for this purpose.
- Typical goodness functions:
  - information gain (ID3/C4.5)
  - information gain ratio
  - gini index

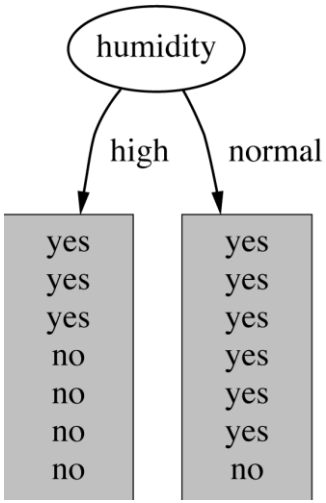
Which attribute to select?

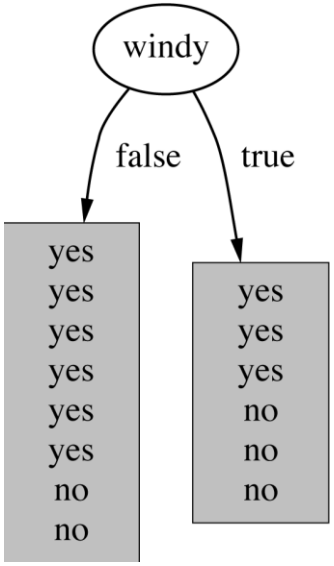
A	B	C	D		Class
0		0	0		Y
0		1	0		Y
0		0	1		Y
0		1	0		Y
1		0	0		N
1		1	1		N
1		0	1		N
1		1	0		N

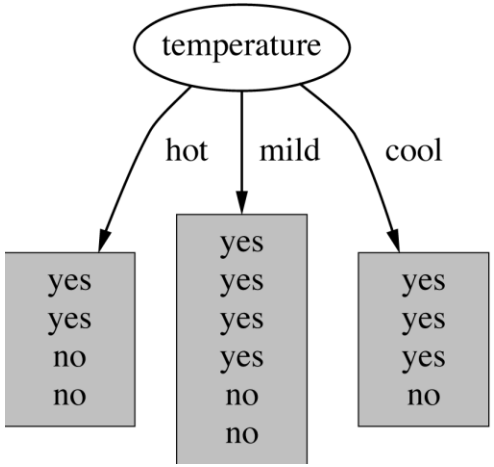
Attribute A vs attribute C ?

Which attribute to select?









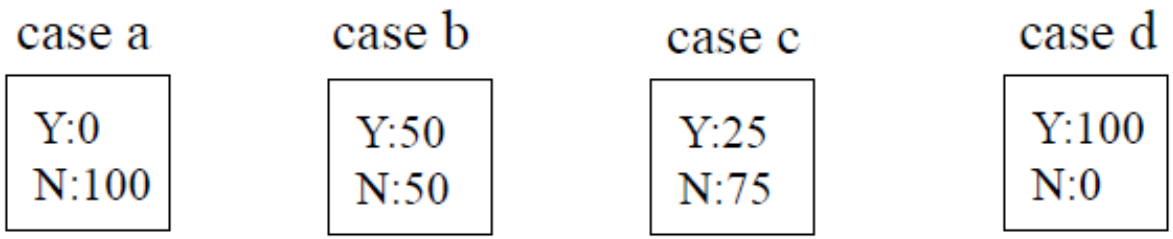
A criterion for attribute selection

- Which is the best attribute?
  - The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the "purest" nodes

- How do we measure the degree of purity/certainty/clearness ?
  - or degree of impurity/uncertainty/noisiness

- Popular *impurity criterion*: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces

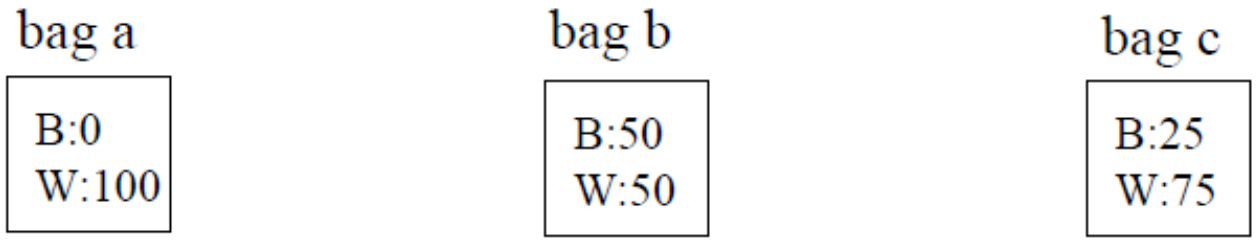
A criterion for attribute selection



Which case is more pure/certain ?

- Question now is how do we represent the degree of impurity in Number (or Formula)
- Any idea ?
  - How about this: degree of impurity = #(Y)/(#(Y)+#(N))

Computing information



- Suppose you pick one ball randomly out of these bags.
- You goal is to know the color of the ball (Black or White)

- You pick one ball out of bag a:
  - How much information do you get ? 0

- You pick one ball out of bag b:
  - How much information do you get 1

- You pick one ball out of bag c:
  - How much information do you get $0 < < 1$

Computing information

- Entropy function: represents the degree of impurity in prob. dist.
  - Also represents the amount of information prob distribution contains
- Information is measured in *bits*
  - Given a probability distribution, the info required to predict an event is the distribution's *entropy*
  - Entropy gives the information required in bits (this can involve fractions of bits!)
- Formula for computing the entropy:

$$\text{entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

*Claude Shannon

"Father of information theory"

Born: 30 April 1916

Died: 23 February 2001

Claude Shannon, who has died aged 84, perhaps more than anyone laid the groundwork for today's digital revolution. His exposition of information theory, stating that all information could be represented mathematically as a succession of noughts and ones, facilitated the digital manipulation of data without which today's information society would be unthinkable.

Shannon's master's thesis, obtained in 1940 at MIT, demonstrated that problem solving could be achieved by manipulating the symbols 0 and 1 in a process that could be carried out automatically with electrical circuitry. That dissertation has been hailed as one of the most significant master's theses of the 20th century. Eight years later, Shannon published another landmark paper, *A Mathematical Theory of Communication*, generally taken as his most important scientific contribution.

Shannon applied the same radical approach to cryptography research, in which he later became a consultant to the US government.

Many of Shannon's pioneering insights were developed before they could be applied in practical form. He was truly a remarkable man, yet unknown to most of the world.

Example: attribute "Outlook"

- "Outlook" = "Sunny":

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971\,\text{bits}$$

- "Outlook" = "Overcast":

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0\,\text{bits}$$

Note: log(0) is not defined, but we evaluate 0*log(0) as zero

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971\,\text{bits}$$

- Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$

$$= 0.693\,\text{bits}$$

Computing the information gain

- Decision Tree uses information gain
- Popular *impurity criterion*: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces
- Strategy: choose attribute that results in greatest information gain

Computing the information gain

- Information gain:

(information before split) – (information after split)

$$\text{gain}(\text{"Outlook"}) = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693 = 0.247 \text{ bits}$$

- Information gain for attributes from weather data:

$$\text{gain}(\text{"Outlook"}) = 0.247 \text{ bits}$$

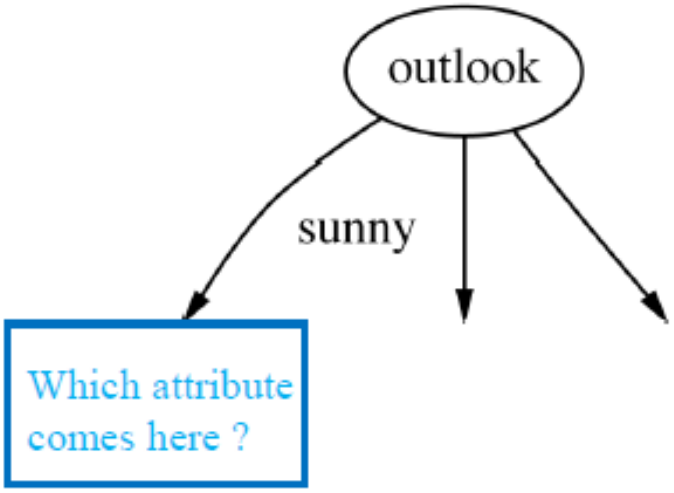
$$\text{gain}(\text{"Temperature"}) = 0.029 \text{ bits}$$

$$\text{gain}(\text{"Humidity"}) = 0.152 \text{ bits}$$

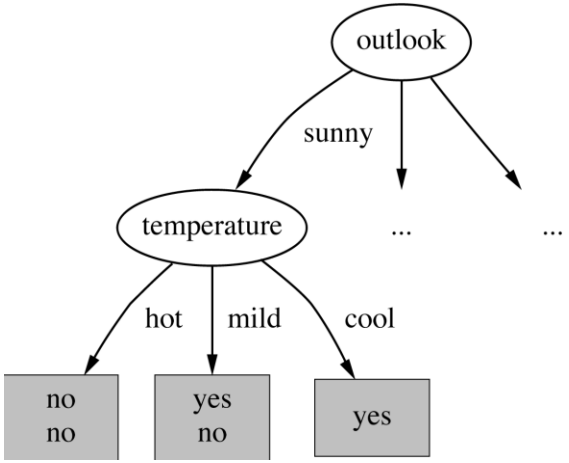
$$\text{gain}(\text{"Windy"}) = 0.048 \text{ bits}$$

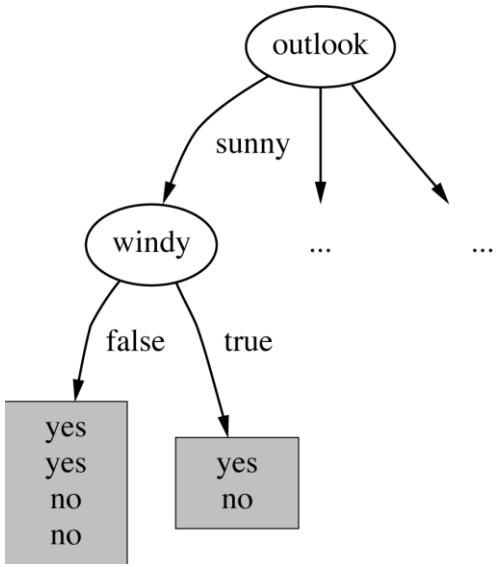
Continuing to split

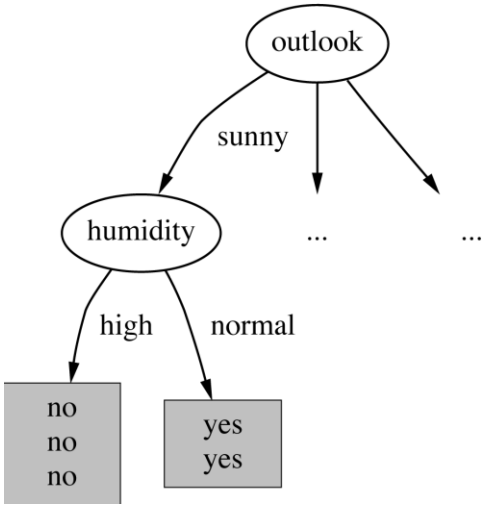
So Outlook becomes the root of the decision tree



Continuing to split





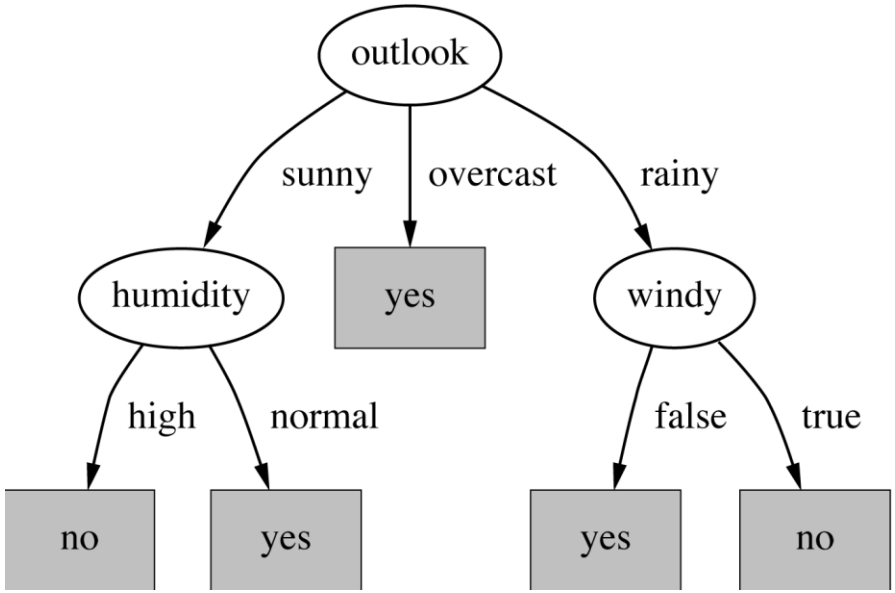


$$gain(\text{" Humidity"}) = 0.971 \text{ bits}$$

$$gain(\text{" Temperature"}) = 0.571 \text{ bits}$$

$$gain(\text{" Windy"}) = 0.020 \text{ bits}$$

The final decision tree



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

  $\Rightarrow$ Splitting stops when data can't be split any further

*CART Splitting Criteria: Gini Index

- If a data set T contains examples from n classes, gini index, gini(T) is defined as

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in T.

gini(T) is minimized if the classes in T are skewed.

*Gini Index

After splitting T into two subsets T1 and T2 with sizes N1 and N2, the gini index of the split data is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute providing smallest $gini_{split}(T)$ is chosen to split the node.

Discussion

- Algorithm for top-down induction of decision trees ("ID3") was developed by Ross Quinlan
  - Gain ratio just one modification of this basic algorithm
  - Led to development of C4.5, which can deal with numeric attributes, missing values, and noisy data
- There are many other attribute selection criteria! (But almost no difference in accuracy of result.)

Summary

- Top-Down Decision Tree Construction
- Choosing the Splitting Attribute
- Information Gain biased towards attributes with a large number of values
- Gain Ratio takes number and size of branches into account when choosing an attribute

Highly-branching attributes

- 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

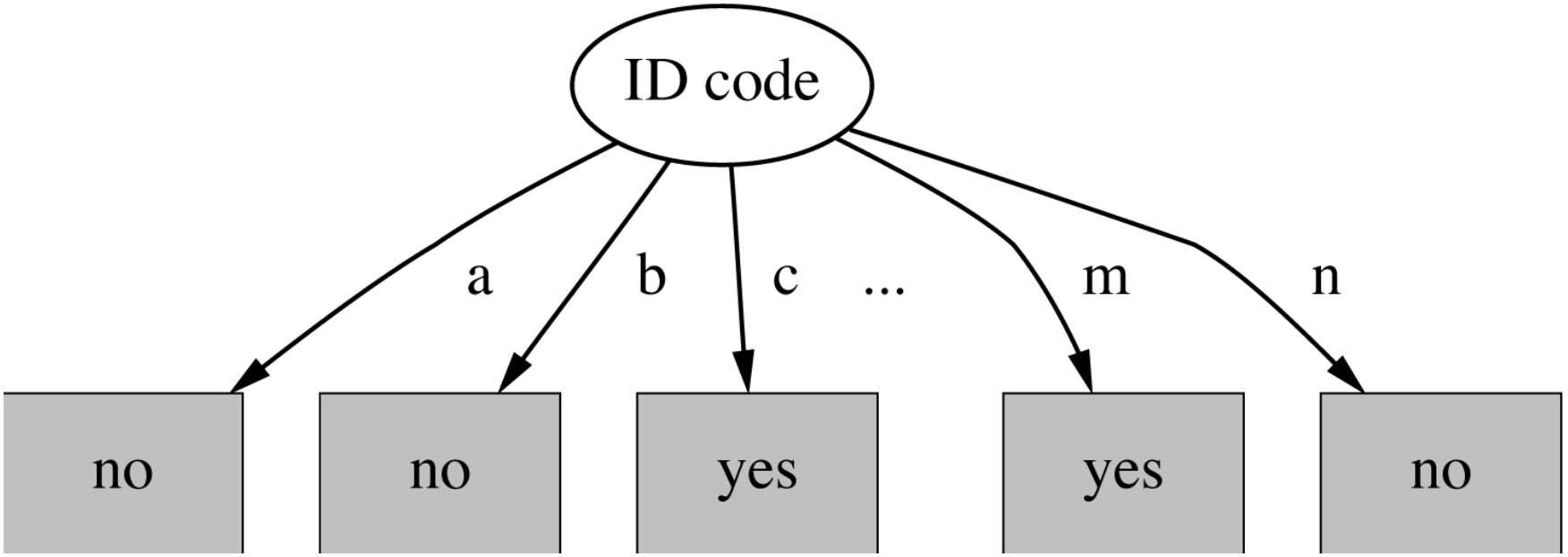
  ⇒Information gain is biased towards choosing attributes with a large number of values

  ⇒This may result in *overfitting* (selection of an attribute that is non-optimal for prediction)

Weather Data with ID code

ID	Outlook	Temperature	Humidity	Windy	Play?
A	sunny	hot	high	false	No
B	sunny	hot	high	true	No
C	overcast	hot	high	false	Yes
D	rain	mild	high	false	Yes
E	rain	cool	normal	false	Yes
F	rain	cool	normal	true	No
G	overcast	cool	normal	true	Yes
H	sunny	mild	high	false	No
I	sunny	cool	normal	false	Yes
J	rain	mild	normal	false	Yes
K	sunny	mild	normal	true	Yes
L	overcast	mild	high	true	Yes
M	overcast	hot	normal	false	Yes
N	rain	mild	high	true	No

Split for ID Code Attribute



Entropy of split = 0 (since each leaf node is “pure”, having only one case.

Information gain is maximal for ID code

Gain ratio

- *Gain ratio*: a modification of the information gain that reduces its bias on high-branch attributes
- Gain ratio should be
  - Large when data is evenly spread
  - Small when all data belong to one branch
- 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 (i.e. how much info do we need to tell which branch an instance belongs to)

Gain Ratio and Intrinsic Info.

- Intrinsic information: entropy of distribution of instances into branches

$$IntrinsicInfo(S,A) \equiv -\sum \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}.$$

- *Gain ratio* (Quinlan'86) normalizes info gain by:

$$GainRatio(S,A) = \frac{Gain(S,A)}{IntrinsicInfo(S,A)}.$$

Computing the gain ratio

- Example: intrinsic information for ID code

$$\text{info}([1,1,\ldots,1) = 14 \times (-1/14 \times \log 1/14) = 3.807 \text{ bits}$$

- **Importance of attribute decreases as intrinsic information gets larger**
- Example of gain ratio:

$$\text{gain\_ratio}(\text{"Attribute"}) = \frac{\text{gain}(\text{"Attribute"})}{\text{intrinsic\_info}(\text{"Attribute"})}$$

- Example:

$$\text{gain\_ratio}(\text{"ID\_code"}) = \frac{0.940 \text{ bits}}{3.807 \text{ bits}} = 0.246$$

Gain ratios for weather data

Outlook		**Temperature**	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.362
Gain ratio: 0.247/1.577	0.156	Gain ratio: 0.029/1.362	0.021

Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

More on the 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:
    - First, only consider attributes with greater than average information gain
    - Then, compare them on gain ratio

Decision Tree Pruning

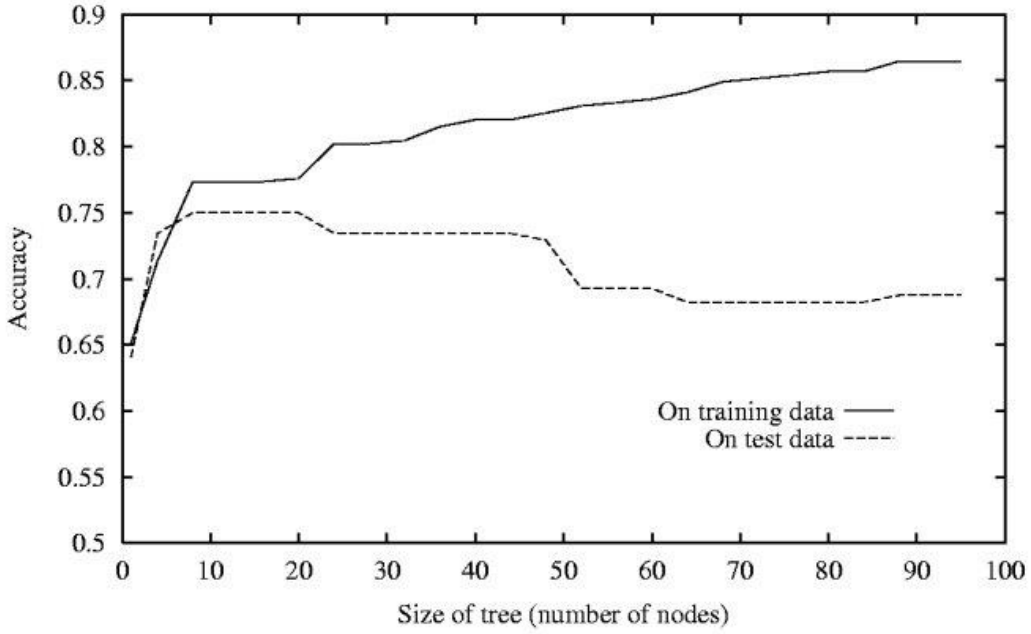
Overfitting

- Hypothesis h overfits iff ∃ h′ with
  - $error_{train}(h) < error_{train}(h')$
  - $error_{true}(h) > error_{true}(h')$

- Hypothesis h **overfits** the data if there exists h' with greater error than h over training examples but less error than h over entire distribution of instances

Overfitting

- Serious problem for all inductive learning methods
- Trees may grow to include irrelevant attributes (e.g., Date, Color, etc.)
- Noisy examples may add spurious nodes to tree

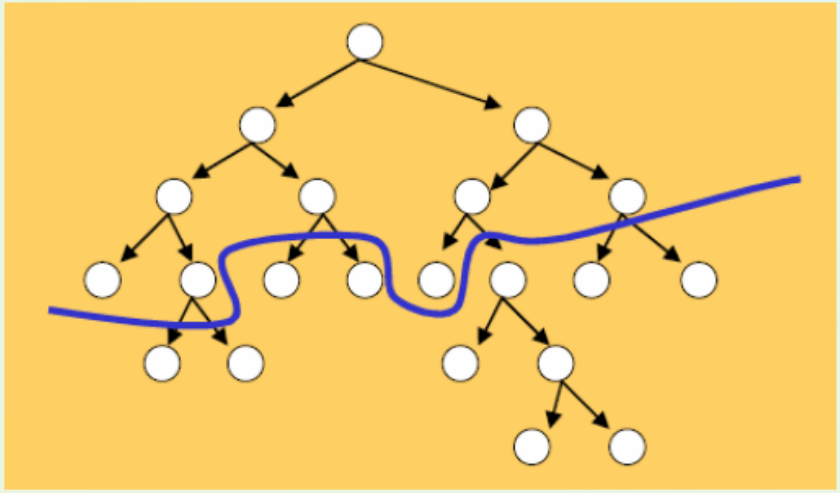


Occam's razor

- "Non sunt multiplicanda entia praeter necessitatem."
  - William of Occam (1285-1347)
- In English: Entities are not to be multiplied beyond necessity (law of parsimony)
- In machine learning (and science): Prefer simpler hypotheses over more complex ones (Occam's razor)
- "Everything should be made as simple as possible, *but not simpler*"
  - Albert Einstein (1879-1955)

Tree pruning

- Avoid overfitting the data by tree pruning.
- After pruning the classification accuracy on unseen data may increase!



Pruning

- Goal: Prevent overfitting to noise in the data
- Two strategies for "pruning" the decision tree:
  - *Prepruning* - stop growing a branch when information becomes unreliable
  - *Postpruning* - take a fully-grown decision tree and discard unreliable parts
- Postpruning preferred in practice—prepruning can "stop too early"

Prepruning

- 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*
  - *Chi-square test: test whether two variables are independent to each other*
- 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

1) Richness and Happiness are independent

	Happy	Unhappy	Total
Rich	8	12	20
Poor	32	48	80
Total	40	60	100

2) Richness and Happiness are significantly dependent

	Happy	Unhappy	Total
Rich	20	0	20
Poor	20	60	80
Total	40	60	100

- Expected value of (Rich, Happy) = #(Rich)*#(Happy)/#(Total)=20*40/100=8
- Expected value of (Poor, Unhappy) = #(Poor)*#(Unhappy)/#(Total)=80*60/100=48
- …

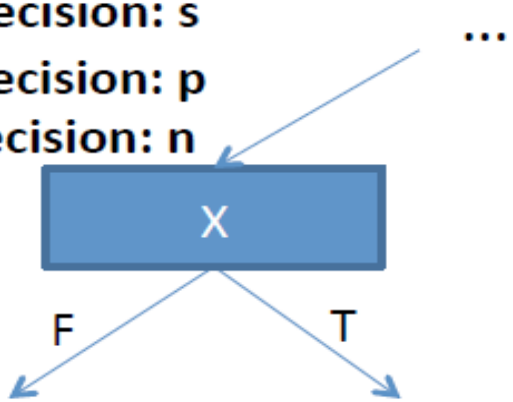
- The more different between expected counts($e_{i,j}$)and real counts($a_{i,j}$), the more dependent two variables are

$$\chi^2 = \sum_{i,j} \frac{(e_{i,j} - a_{i,j})^2}{e_{i,j}}$$

- For case 2), $\chi^2 = \frac{(8-20)^2}{8} + \frac{(12-0)^2}{12} + \frac{(32-20)^2}{32} + \frac{(48-60)^2}{48}$

Chi-Square Test

of instances entering this decision: s
of + instances entering this decision: p
of - instances entering this decision: n



instances here: s_f
of + instances here: p_f
of - instances here: n_f

instances here: s_t
of - instances here: p_t
of - instances here: n_t

	p	n	Total
T			
F			
Total			

Calculate the Chi-square (X^2) value between X (T/F) and label (p/n)

Prepruning

	a	b	class
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

- 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 post-pruning

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:
  1. *Reduced Error Pruning*
  2. *Subtree replacement*
- Possible strategies:
  - error estimation
  - significance testing
  - MDL principle

Reduced Error pruning

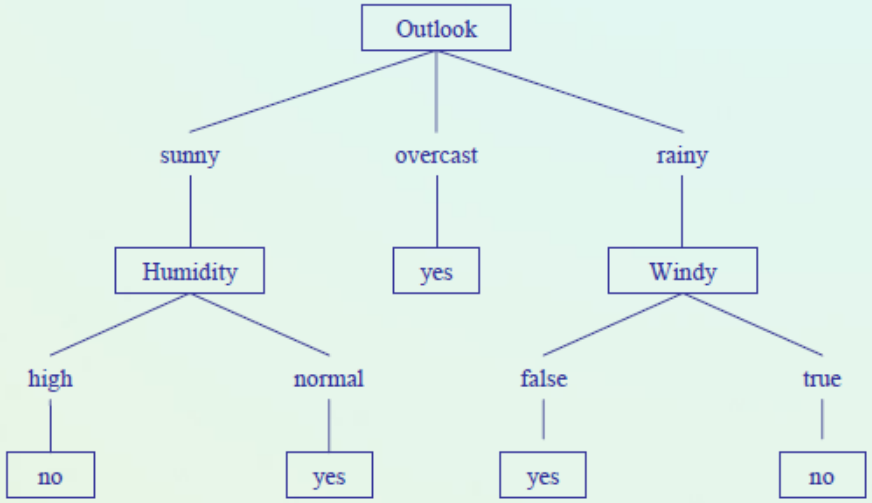
Split data into training and validation sets.

Pruning a decision node *d* consists of:

1. removing the subtree rooted at *d*.
2. making *d* a leaf node.
3. assigning *d* the most common classification of the training instances associated with *d*.

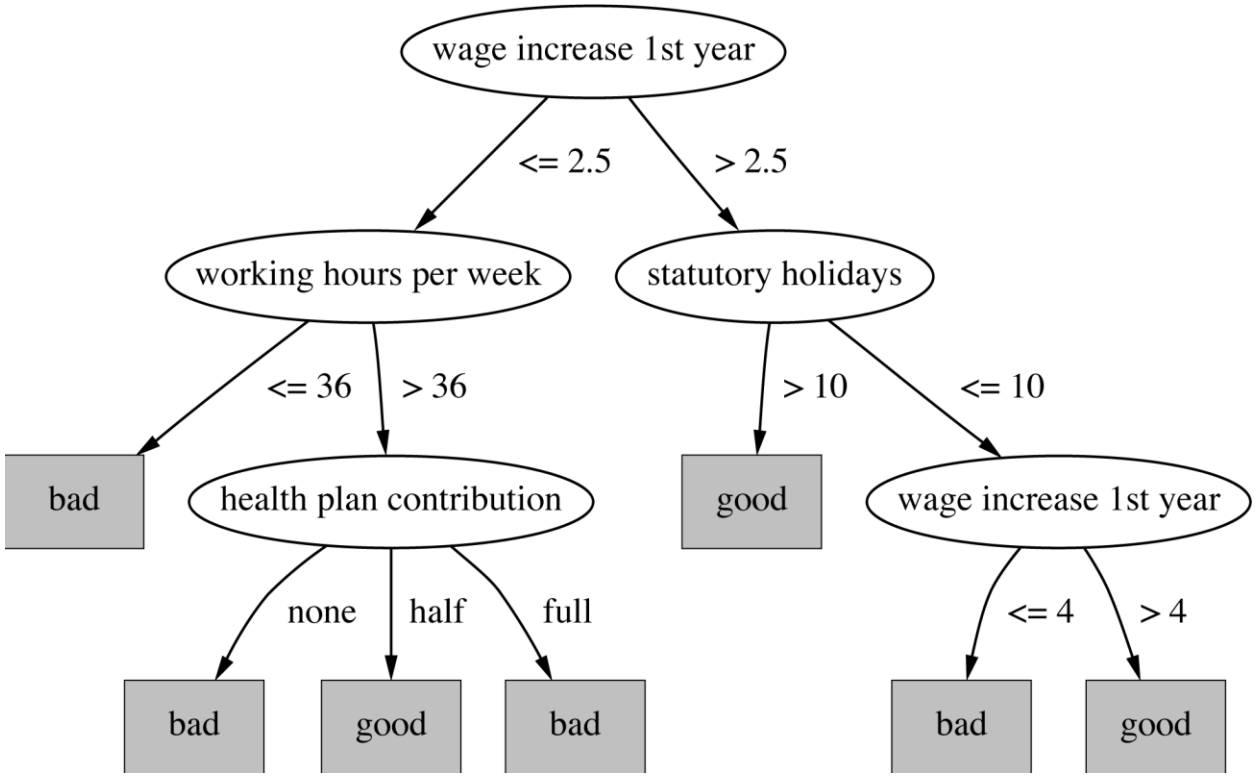
Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it).
2. Greedily remove the one that most improves validation set accuracy.



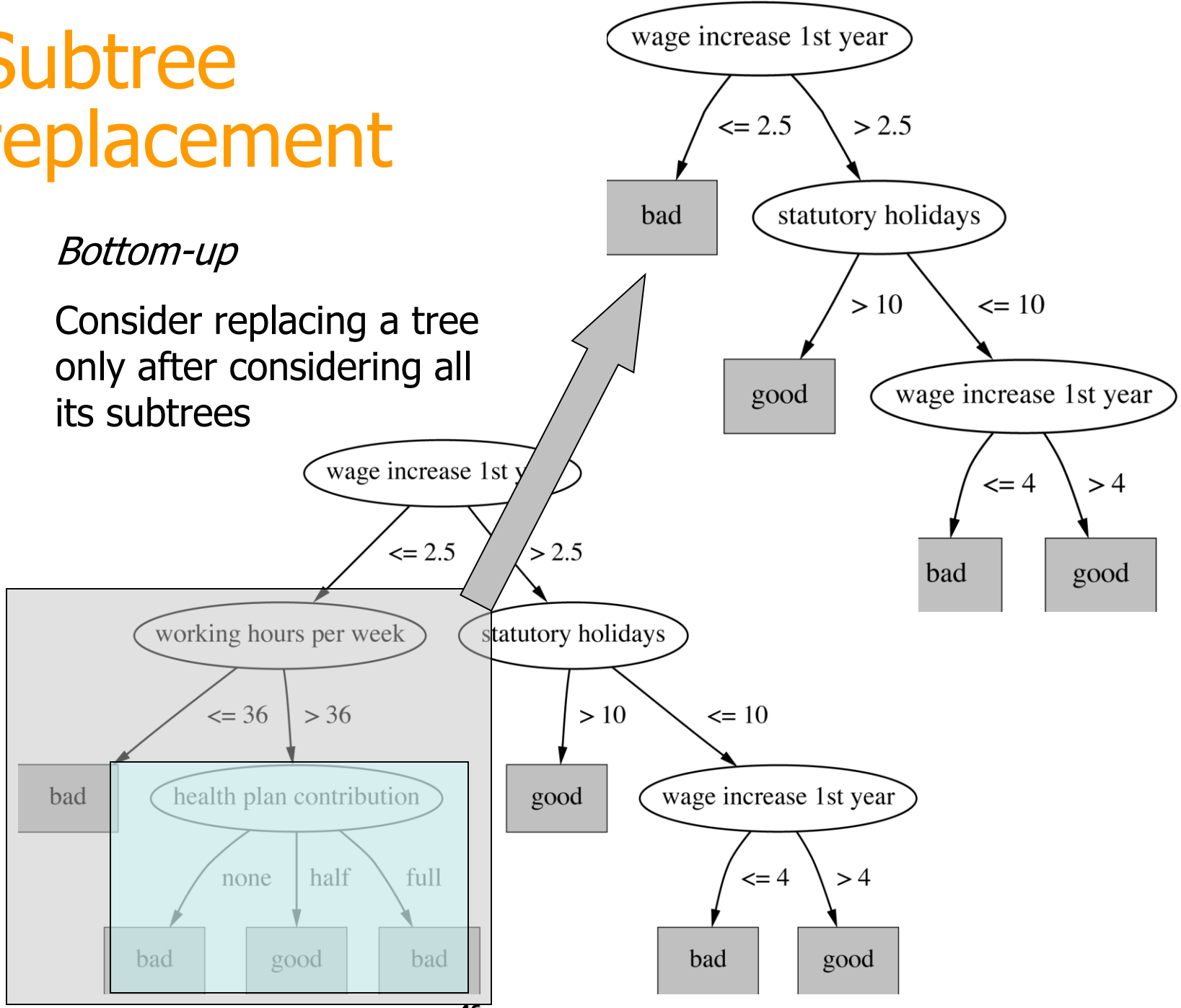
Subtree replacement

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees
- Ex: labor negotiations



Subtree replacement

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees



Estimating error rates

- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator
  Q: Why it would result in very little pruning?
- C4.5's method
  - Derive confidence interval from training data
  - Use a heuristic limit, derived from this, for pruning
  - Standard Bernoulli-process-based method
  - Shaky statistical assumptions (based on training data)

*Mean and variance

- Mean and variance for a Bernoulli trial: p, $p\,(1-p)$
- Expected success rate $f=S/N$
- Mean and variance for f: p, $p\,(1-p)/N$
- For large enough N, f follows a Normal distribution
- c% confidence interval $[-z \leq X \leq z]$ for random variable with 0 mean is given by:

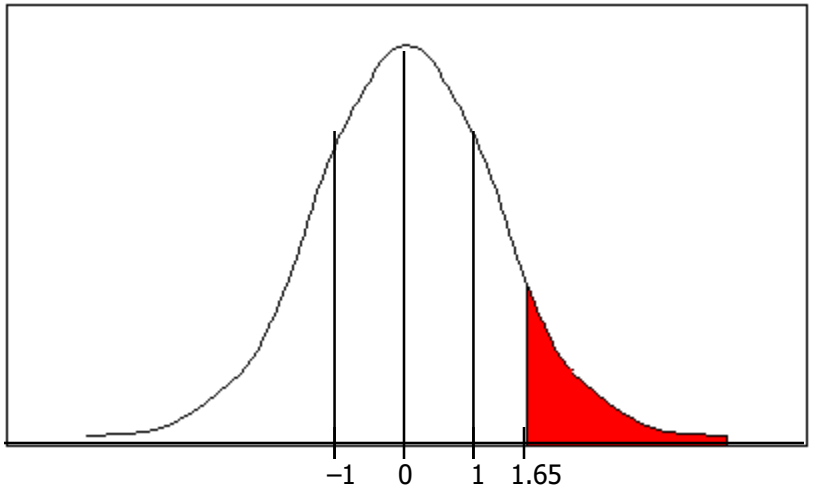
$$\Pr[-z \leq X \leq z] = c$$

- With a symmetric distribution:

$$\Pr[-z \leq X \leq z] = 1 - 2 \times \Pr[X \geq z]$$

*Confidence limits

- Confidence limits for the normal distribution with 0 mean and a variance of 1:



Pr[$X \geq z$]	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
25%	0.69
40%	0.25

- Thus:

$$\Pr[-1.65 \leq X \leq 1.65] = 90\%$$

- To use this we have to reduce our random variable f to have 0 mean and unit variance

*Transforming *f*

- Transformed value for *f*: $\frac{f-p}{\sqrt{p(1-p)/N}}$

  (i.e. subtract the mean and divide by the *standard deviation*)

- Resulting equation:

$$\Pr\left[-z \le \frac{f-p}{\sqrt{p(1-p)/N}} \le z\right] = c$$

- Solving for *p*:

$$p = \left(f + \frac{z^2}{2N} \pm z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \Big/ \left(1 + \frac{z^2}{N}\right)$$

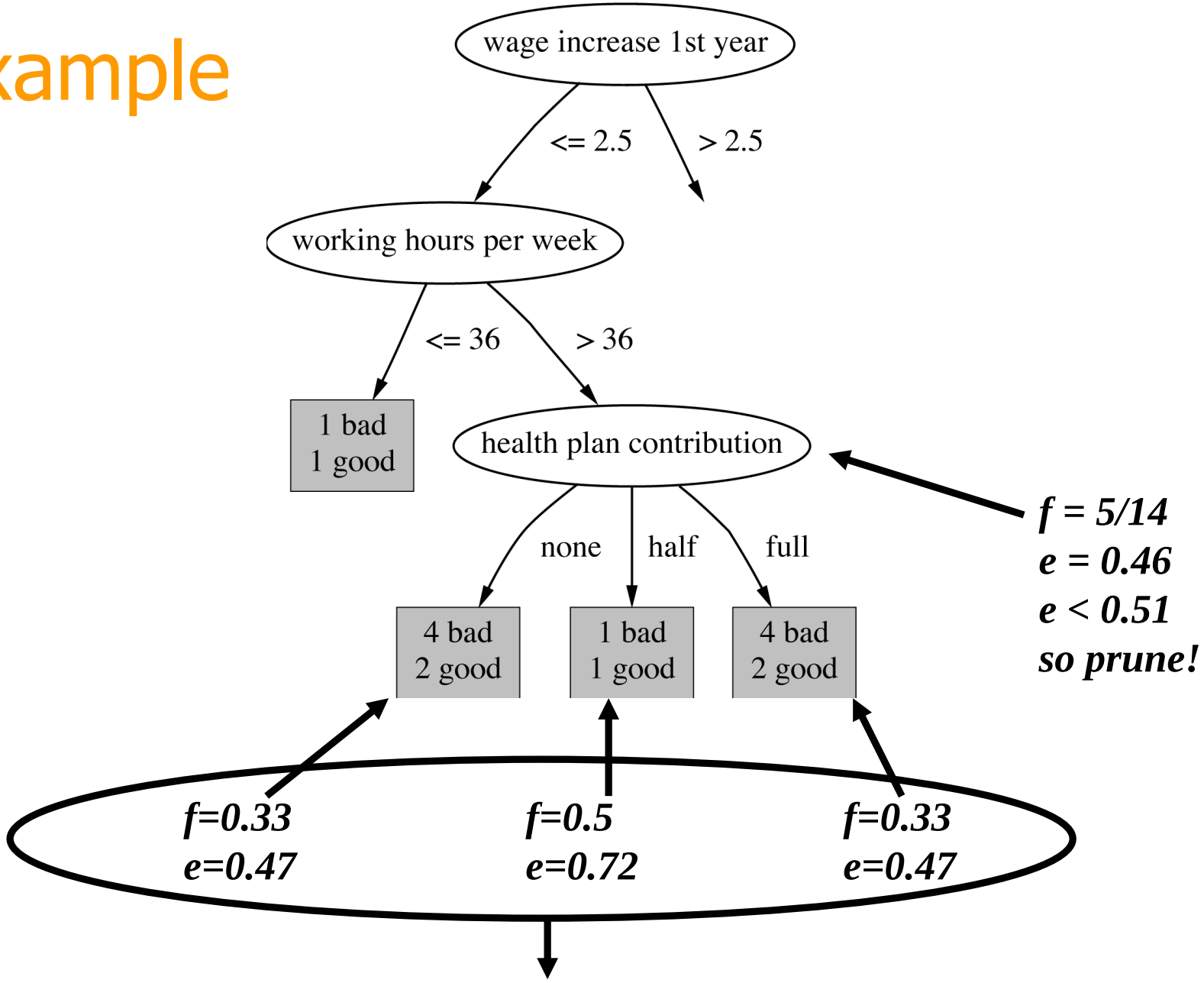
C4.5's method

- Error estimate for subtree is weighted sum of error estimates for all its leaves
- Error estimate for a node (upper bound):

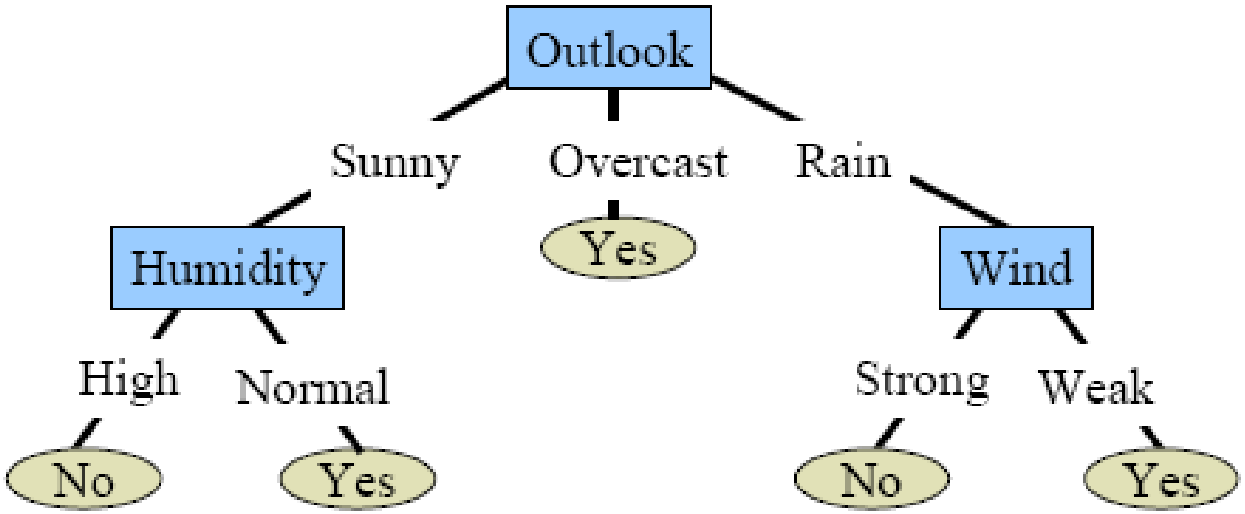
$$e = \left(f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \Big/ \left(1 + \frac{z^2}{N} \right)$$

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

Example


wage increase 1st year
<= 2.5
> 2.5
working hours per week
<= 36
> 36
1 bad
1 good
health plan contribution
none
half
full
4 bad
2 good
1 bad
1 good
4 bad
2 good
f = 5/14
e = 0.46
e < 0.51
so prune!
f=0.33
e=0.47
f=0.5
e=0.72
f=0.33
e=0.47
Combined using ratios 6:2:6 gives 0.51

From trees to rules



If	(outlook = sunny)	∧	(humidity=high)	then	PlayTennis=No
If	(outlook = sunny)	∧	(humidity=normal)	then	PlayTennis=Yes
If	(outlook = overcast)			then	PlayTennis=Yes
If	(outlook = rain)	∧	(wind=strong)	then	PlayTennis=No
If	(outlook = rain)	∧	(wind=weak)	then	PlayTennis=Yes

Randomly remove conditions in IF part, and evaluate