Never Stand Still

Tree Learning

COMP9417 Machine Learning & Data Mining

Term 1, 2020

Adapted from slides by Dr Michael Bain

Aims

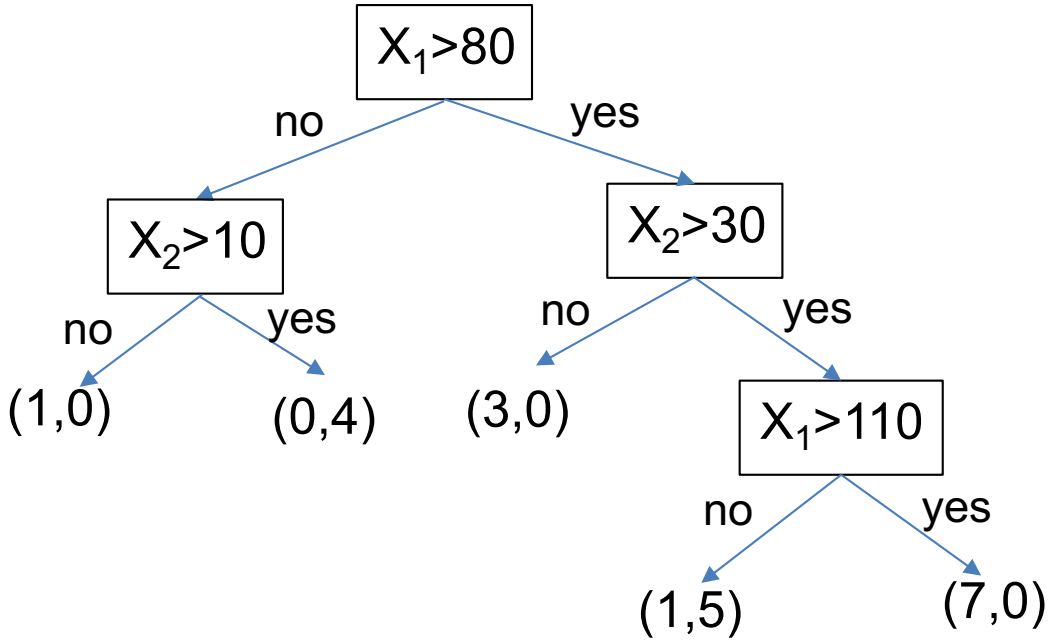
决策树

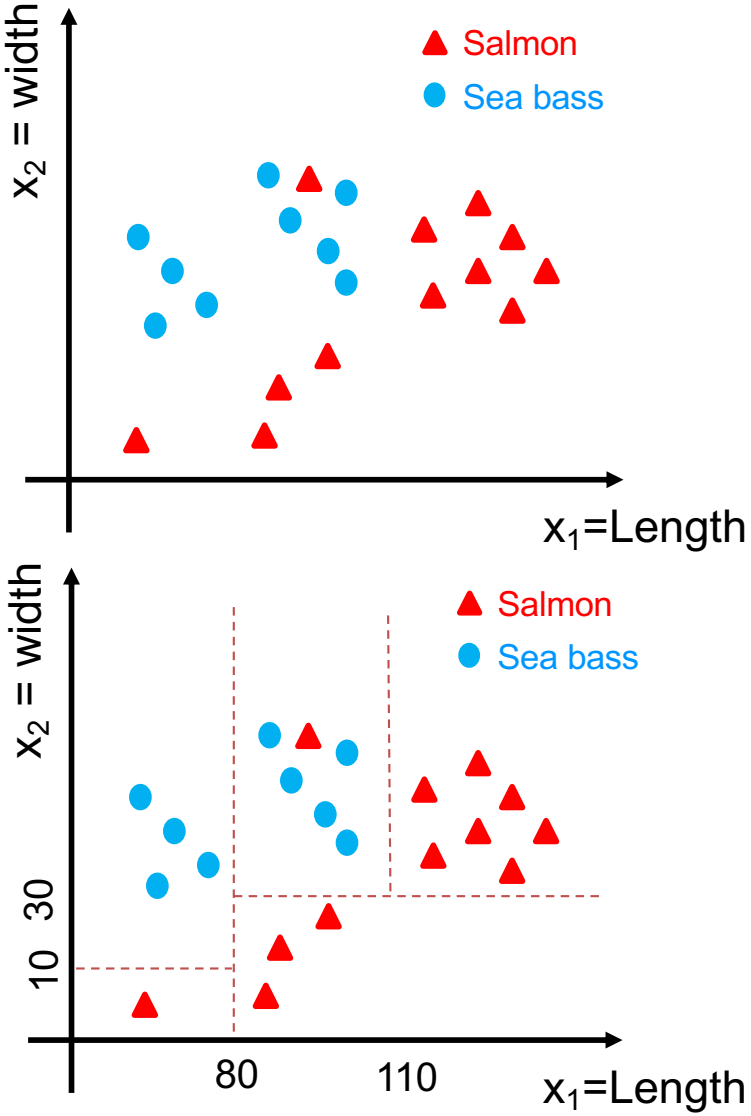
This lecture will enable you to describe decision tree learning, the use of entropy and the problem of overfitting. Following it you should be able to:

- define the decision tree representation
- list representation properties of data and models for which decision trees are appropriate
- reproduce the basic top-down algorithm for decision tree induction (TDIDT)
- define entropy in the context of learning a classifier from examples
- define overfitting of a training set by a hypothesis
- describe developments of the basic TDIDT algorithm: pruning, rule generation, numerical attributes, many-valued attributes, costs, missing values
- describe the inductive bias of the basic TDIDT algorithm
- describe regression trees

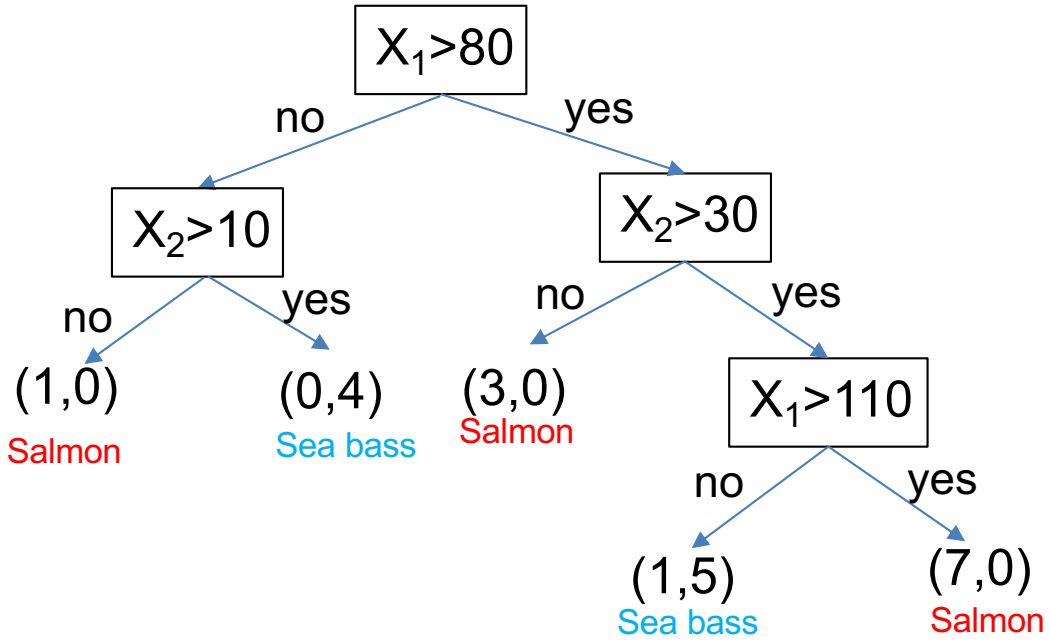
Classification: Decision trees

Example: Let's go back to the fish example with two types of "*salmon*" and "*sea bass*" and assume we have two features $length$ and $width$ to classify fish type





Classification: Decision trees



For any new sample, we start from the top of the tree and answer the questions and follow the appropriate road to the bottom and then decide about the label

Why use decision trees?

- Decision trees are probably the single most popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap (efficient, even on big data)
- There are some drawbacks, though — e.g., high variance
- They do classification, i.e., predict a categorical output from categorical and/or real inputs, or regression

Decision Tree for PlayTennis

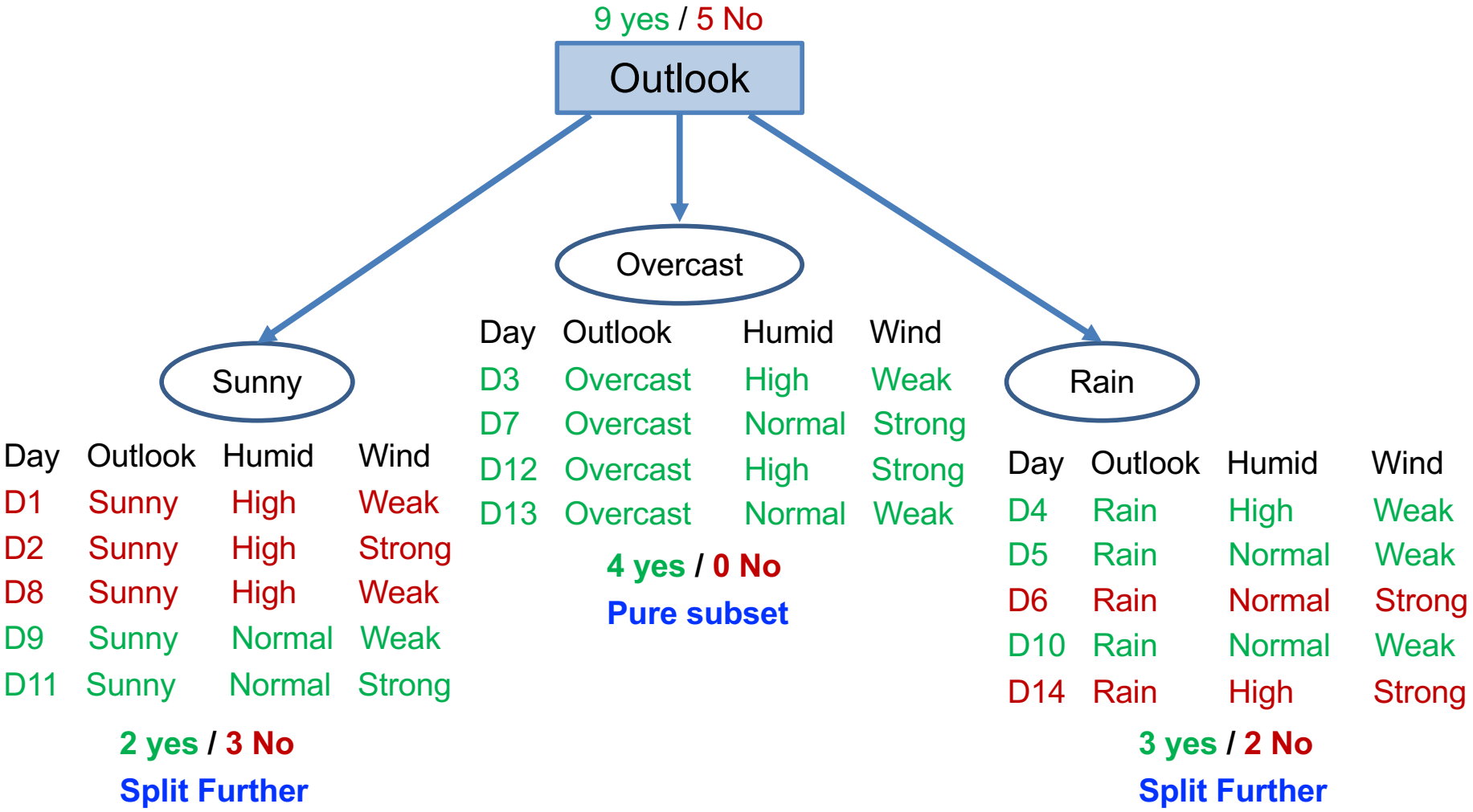
Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes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

Decision Tree for PlayTennis

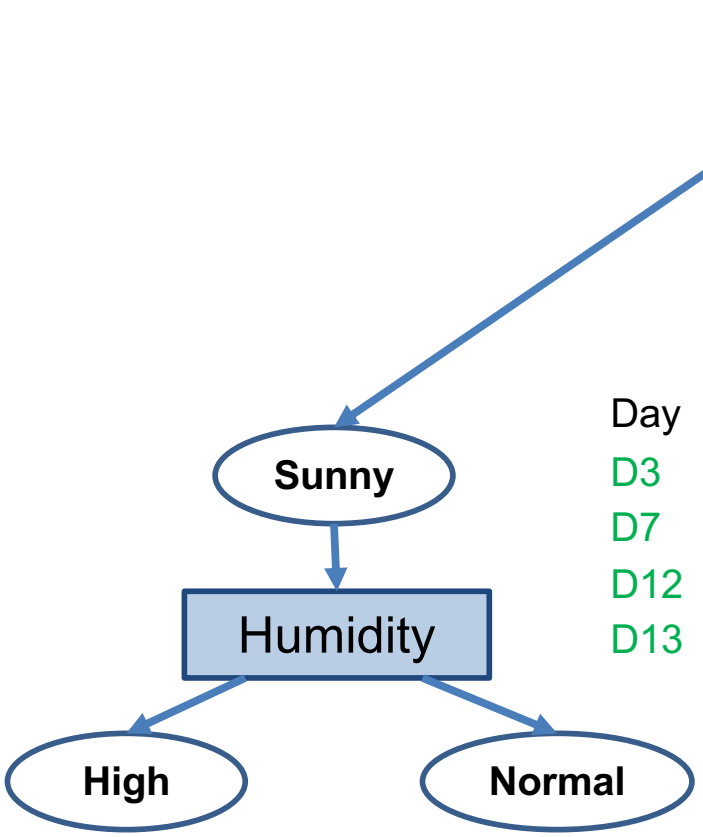
Decision tree works in a divide and conquer fashion:

- Split into subsets
- Are they pure?
  - If yes: stop
  - If not: repeat
- For a new data, find the subset the data falls into

Decision Tree for PlayTennis



Decision Tree for PlayTennis



Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

4 yes / 0 No

Pure subset

Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

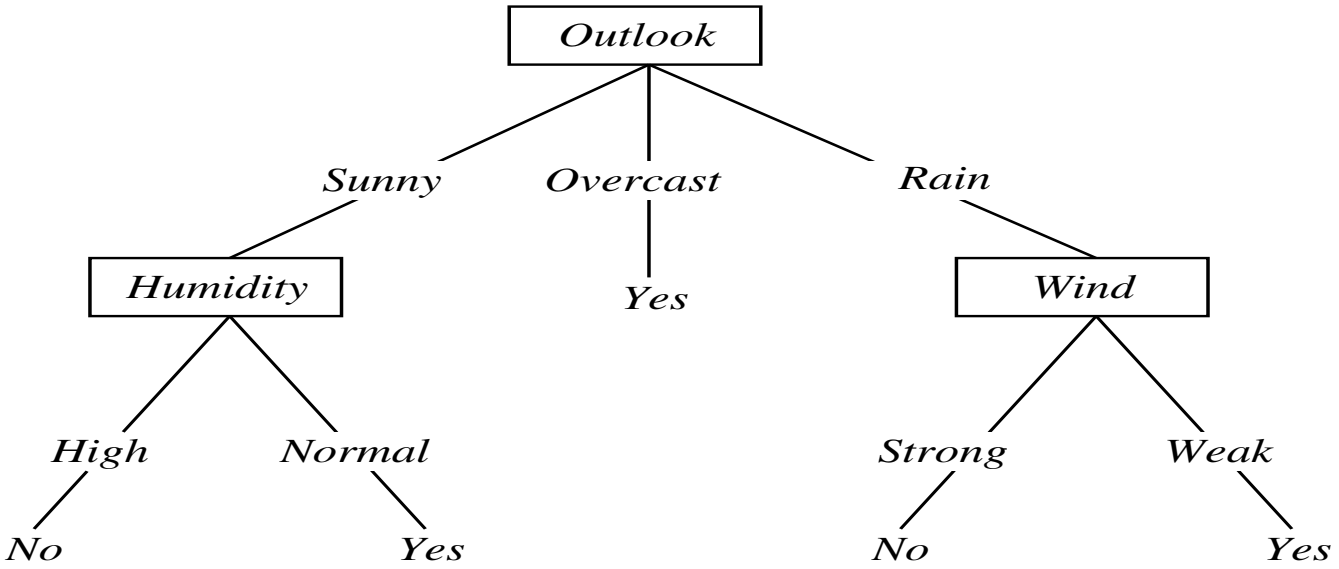
3 yes / 2 No

Split Further

Day	Humid	Wind
D1	High	Weak
D2	High	Strong
D8	High	Weak

Day	Humid	Wind
D9	Normal	Weak
D11	Normal	Strong

Decision Tree for PlayTennis



What would be the decision for a new data:

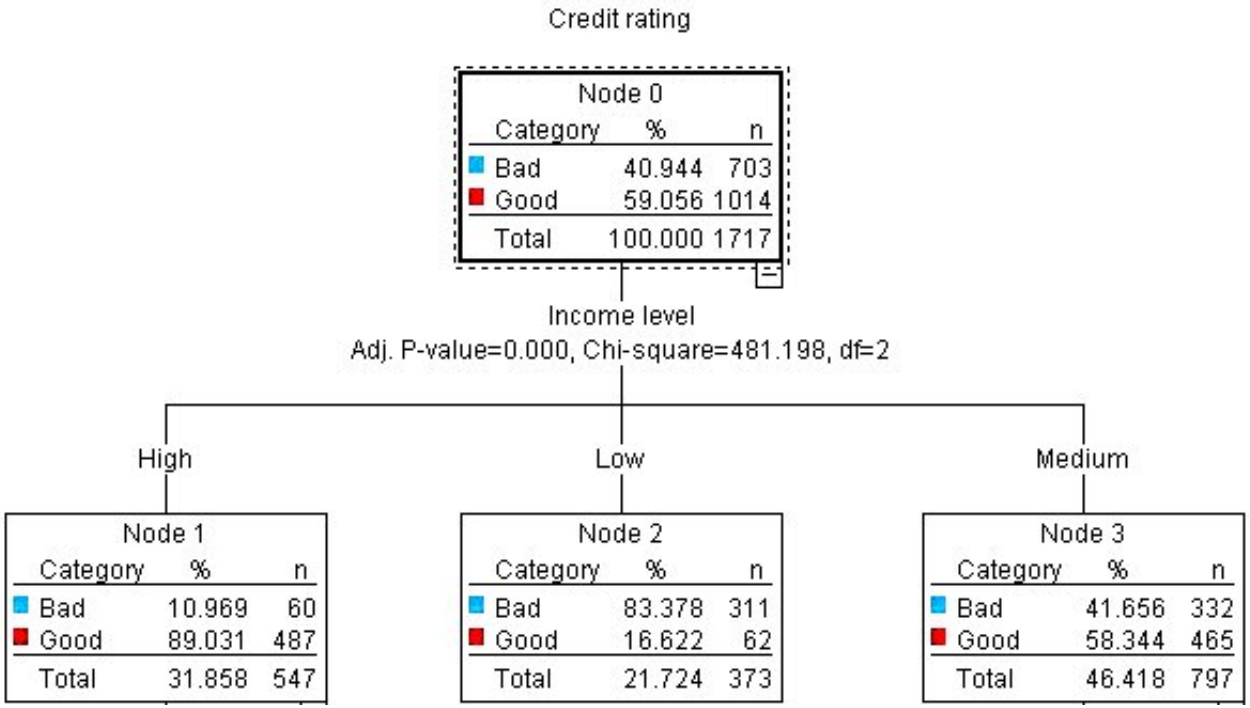
<Outlook = Rain, Humidity = High , Wind = Weak> ?

A Tree to Predict C-Section Risk

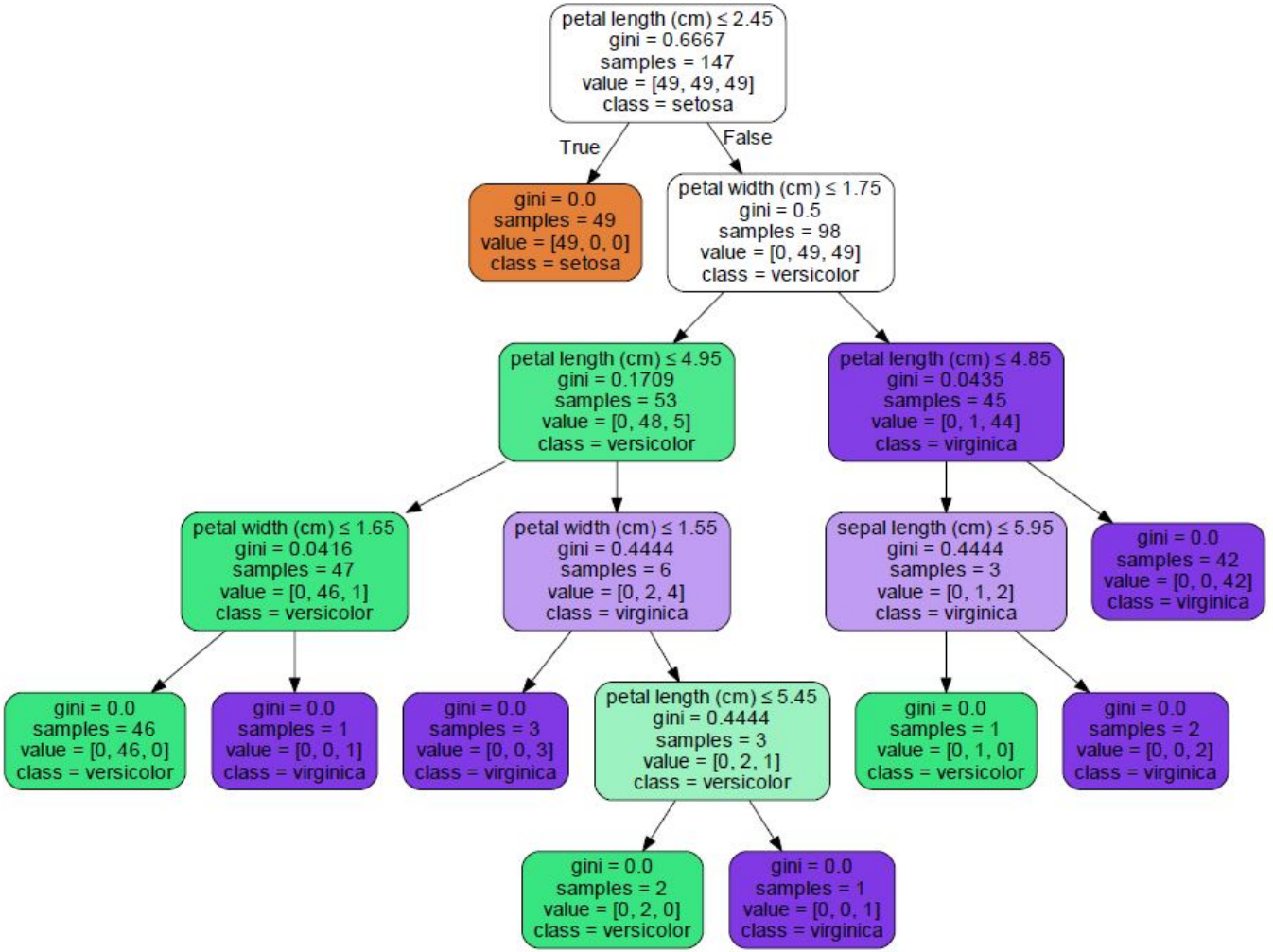
Learned from medical records of 1000 women. Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05-
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Decision Tree for Credit Rating



Decision Tree for Fisher's Iris data



Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

Decision Tree: Expressiveness

Using decision trees to represent Boolean functions like AND (∧), OR (∨), XOR (⊕)

$\boldsymbol{X \wedge Y}$

$X = t$:

| $Y = t$: $true$

| $Y = f$: $false$

$X = f$: $false$

$\boldsymbol{X \vee Y}$

$X = t$: $true$

$X = f$:

| $Y = t$: $true$

| $Y = f$: $false$

Decision Tree: Expressiveness

$\boldsymbol{X \oplus Y}$

$X = t$:

| $Y = t: false$

| $Y = f: true$

$X = f$:

| $Y = t: true$

| $Y = f: false$

"In general, decision trees represent a disjunction of conjunctions of constraints on the attribute-values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions" (Mitchell, 1997)

When to Consider Decision Trees?

- Instances described by a mix of numeric features and discrete attribute–value pairs
- Target function is discrete valued (otherwise use regression trees)
- Possibly noisy training data
- Interpretability is an advantage

Examples are extremely numerous, including:

- Equipment or medical diagnosis
- Credit risk analysis
- Modelling calendar scheduling preferences
- etc.

Top-Down Induction of Decision Trees (TDIDT)

Main loop:

- $A \leftarrow$ the "best" decision attribute for next node to split examples
- Assign A as decision attribute for node
- For each value of A, create new descendant of node (child node)
- Split training examples to child nodes
- If training examples perfectly classified (pure subset), Then $STOP$, Else iterate over new child nodes

Discovered by two people independently:

- Ross Quinlan (ID3: 1986), (C4.5: 1993)
- Breimanetal (CaRT: 1984) from statistics

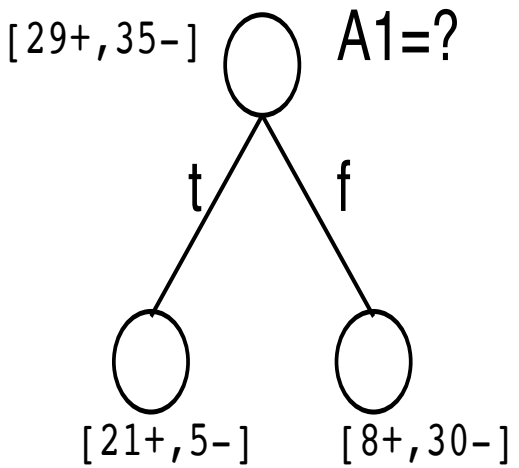
Top-Down Induction of Decision Trees (TDIDT)

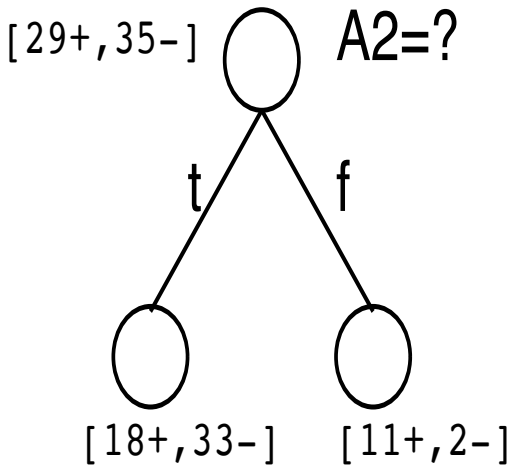
- ID3 (Iterative Dichotomiser 3)
  - Uses categorical attributes/features
  - For categorical targets
  - Is precursor of C4.5 (without restriction of categorical attributes)

- CaRT (Classification & Regression Tree)
  - Uses both categorical and numerical attributes/features
  - Supports both categorical & numerical targets

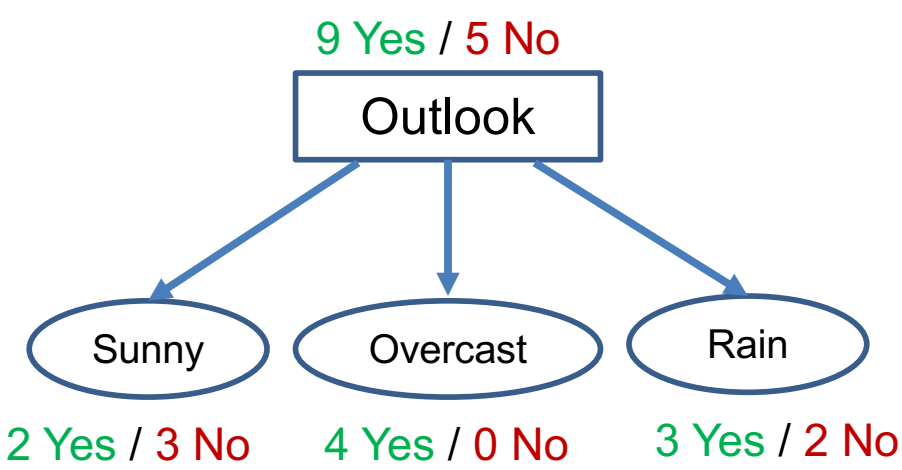
Here, the focus is on ID3 for classification.

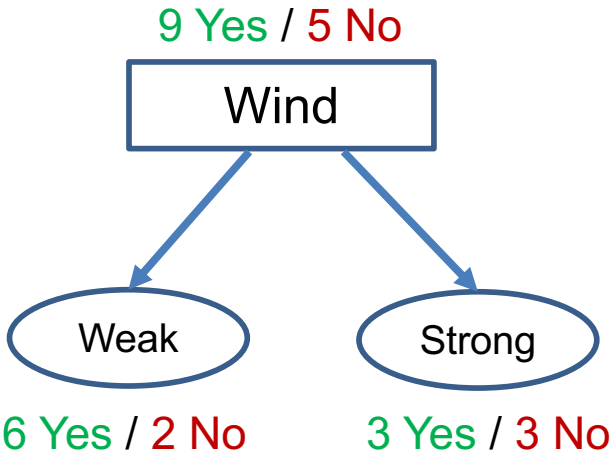
Which attribute is best?



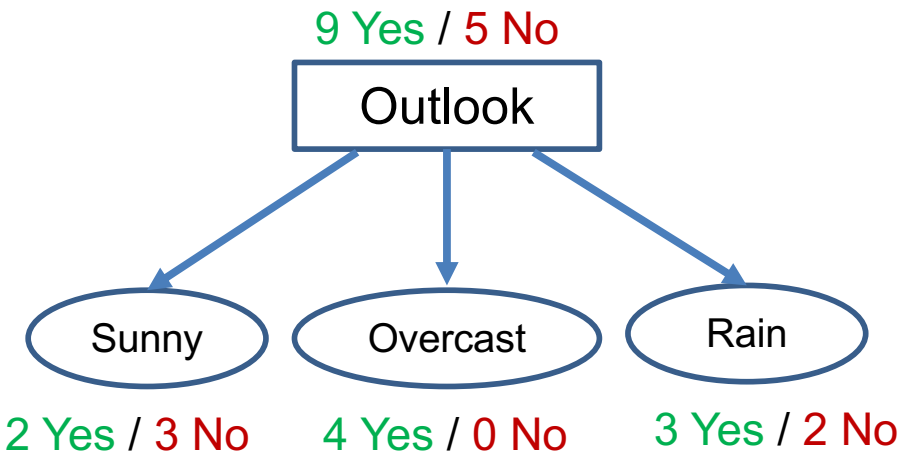


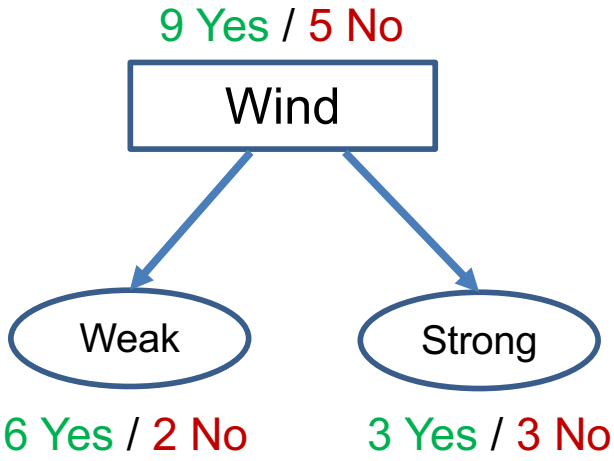
Which attribute is best?





Which attribute is best?





- We are looking for a split whit higher “purity”
- We need to measure the purity of the split
  - More certain about our classes after split
    - A set with all examples belonging to one class is 100% pure
    - A set with 50% examples in one class and 50% in the other is 100% uncertain and impure

Which attribute is best?

There are many different ways to measure the purity of a subset, but one good measure for it is Entropy.

（信息）熵

Entropy: 信息不确定性程度的度量

- From statistical point of view: is a measure of uncertainty
- From information theory point of view: The amount of information (in the Shannon sense) needed to specify the full state of a system

Entropy

Entropy measures the "impurity" of S

[0,1]

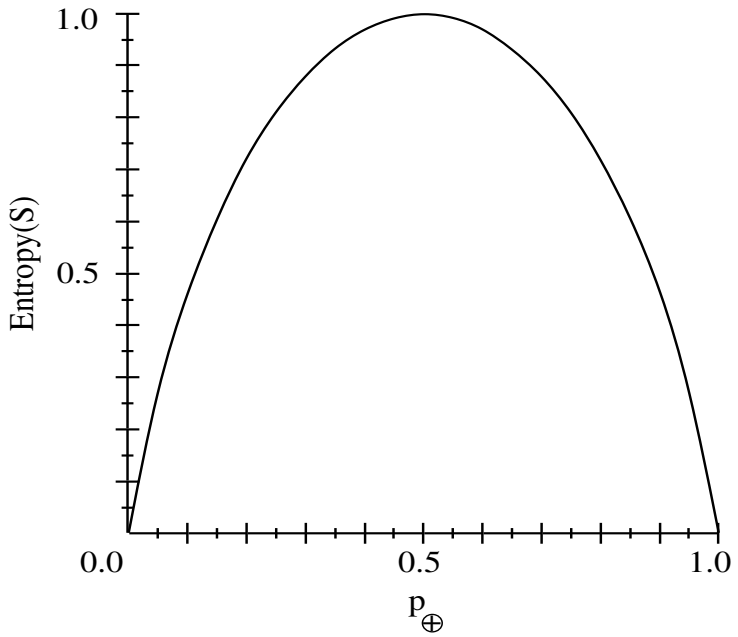
$$Entropy(S) = H(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

- S is subset of training examples
- $p_{\oplus}, p_{\ominus}$ are the portion(%) of positive and negative examples in S

Interpretation: if item x belongs to S, how many bits are needed to tell if x is positive or negative?

A "pure" sample set is one in which all examples are of the same class.

Entropy



Where:

S is a sample of training examples

$p_{\oplus}$ is the proportion of positive examples in S

$p_{\ominus}$ is the proportion of negative examples in S

General Case

From information theory, the optimal number of bits to encode a symbol with probability p is $-\log_2 p$ ...

Suppose X can have one of k values ... $v_1, v_2, \ldots, v_k$

$$P(X = v_1) = p_1, P(X = v_2) = p_2, \ldots, P(X = v_k) = p_k$$

What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from X's distribution ? It's

$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \cdots - p_k \log_2 p_k$$

$$= -\sum_{j=1}^{k} p_j \log_2 p_j$$

$H(X)$ is the *entropy* of X

Entropy

$Entropy(S)$ is the expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of S:

$$p_{\oplus}\,(-\log_2 p_{\oplus}) + p_{\ominus}\,(-\log_2 p_{\ominus})$$

$$Entropy(S) = -p_{\oplus}\,\log_2 p_{\oplus} - p_{\ominus}\,\log_2 p_{\ominus}$$

And we want to minimize this entropy.

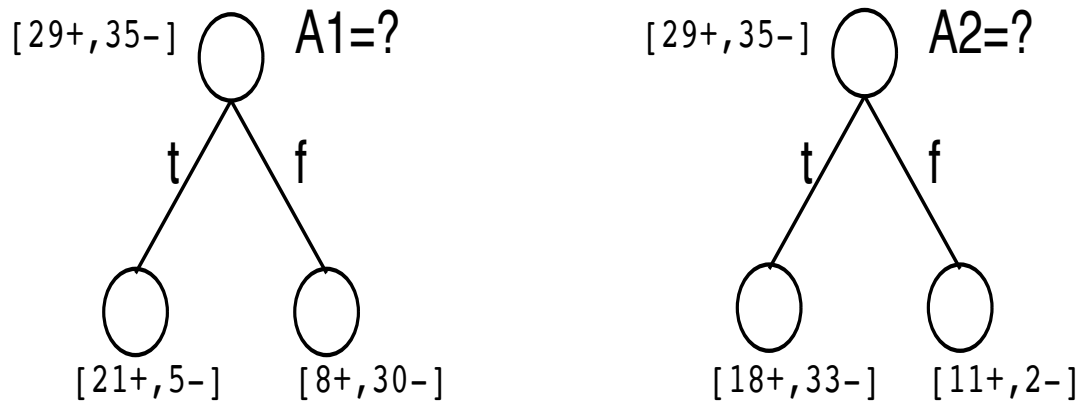
Entropy

- "High entropy"/"impure set" means X is very uniform and boring
  - Example: (3 samples from $\oplus$, 3 samples from $\ominus$)
  - $H(S) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1$ (can be interpreted as 1 bits)

- "Low entropy"/"pure set" means X is not uniform and interesting
  - Example: (6 samples from $\oplus$, 0 samples from $\ominus$)
  - $H(S) = -\frac{6}{6}\log_2\frac{6}{6} - \frac{0}{6}\log_2\frac{0}{6} = 0$ (can be interpreted as 0 bits)

Entropy

Entropy is a measure in just one subset.



$$H(S) = -\frac{29}{64}\log_2\frac{29}{64} - \frac{35}{64}\log_2\frac{35}{64} = 0.9936$$

But, we are interested in expected drop in entropy after split, to decide which attribute is the best. How?

Information Gain

- $Gain(S, A)$ is the expected reduction in entropy due to sorting on A

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

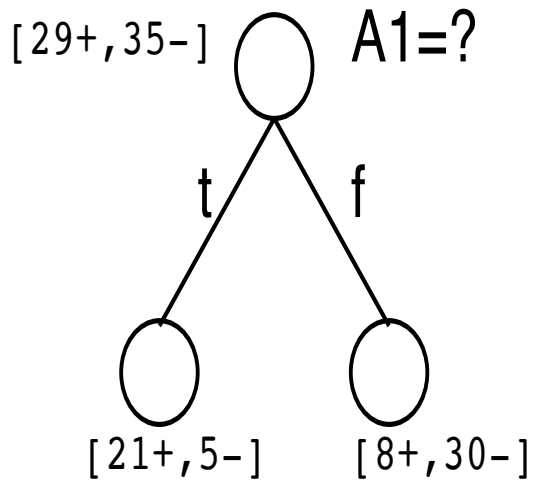
  - v is the possible values of attribute A
  - S is the set of examples we want to split
  - S_v is the subset of examples where $X_A = v$

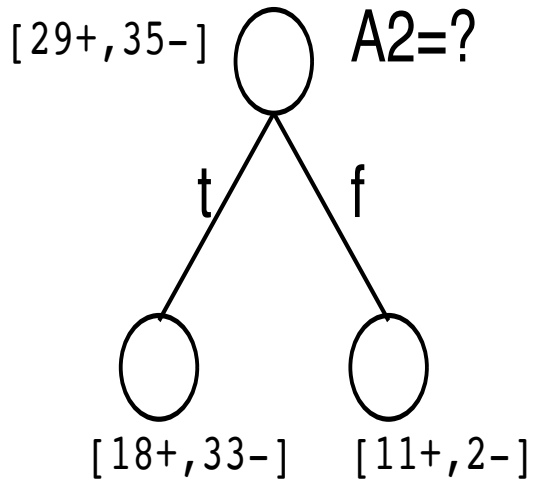
- We want to find the attribution which maximizes the gain

This is also called "mutual information" between attribute A and class labels of S

Information Gain

What is the information gain for attribute $A1$ and $A2$?





Information Gain

$$Gain(S, A1) = Entropy(S) - (\frac{|S_t|}{|S|} Entropy(S_t) + \frac{|S_f|}{|S|} Entropy(S_f)$$

$$= 0.9936 - ((\frac{26}{64}(-\frac{21}{26}\log_2\left(\frac{21}{26}\right) - \frac{5}{26}\log_2(\frac{5}{26}))) +$$

$$(\frac{38}{64}(-\frac{8}{38}\log_2\left(\frac{8}{38}\right) - \frac{30}{38}\log_2(\frac{30}{38}))))$$

$$= 0.9936 - (0.2869 + 0.4408)$$

$$= 0.2658$$

$$Gain(S, A2) = 0.9936 - (0.7464 + 0.0828)$$

$$= 0.1643$$

Information Gain

So in this example, we choose $A1$, since it gives a larger expected reduction in entropy.

To select best attribute at each branch:

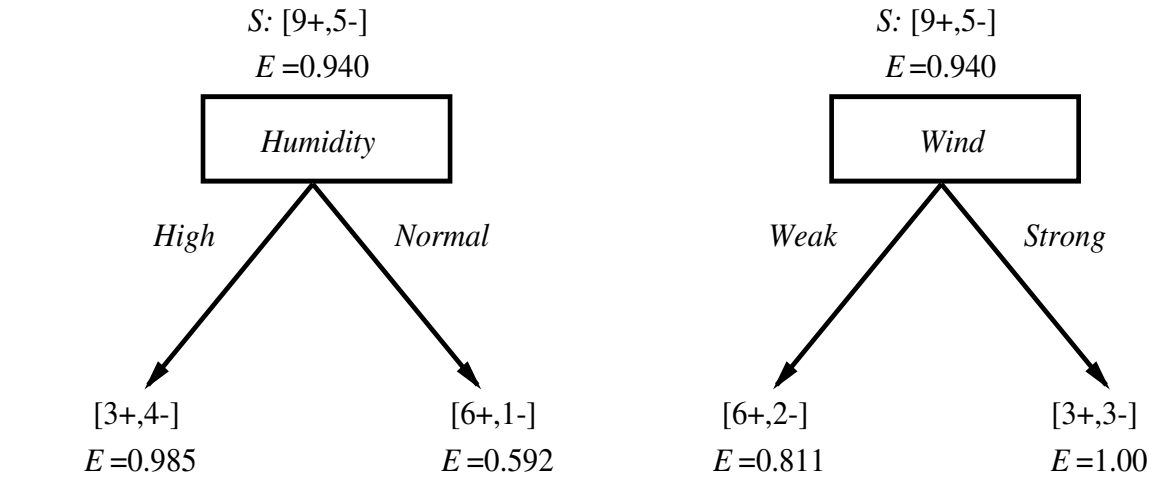
- take every/remaining attributes in your data
- Compute information gain for that attribute
- Select the attribute that has the highest information gain

Training Examples

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

Information gain once more

Which attribute is the best classifier?



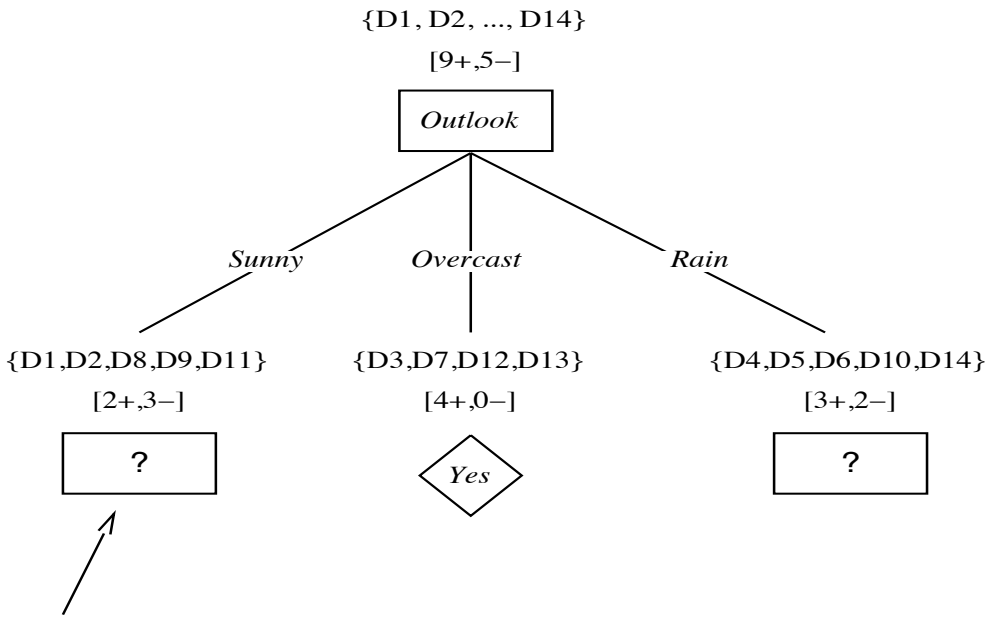
Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Information gain once more



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny} , *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny} , *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny} , *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Information Gain

Will information gain always finds the prefect solution?

Information Gain Limitations

- Information gain is more biased towards attributes with large number of values/categories
    - Subsets are more likely to be pure if there is a large number of values
    - This can result overfitting (selection of an attribute which is non-optimal for prediction) which does not generalize well to unseen data
- Suggested solution: gain ratio
    - A modification of information gain that reduces the bias
    - Takes number and size of branches into account

Gain ratio

$$SplitEntropy(S, A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

Where:

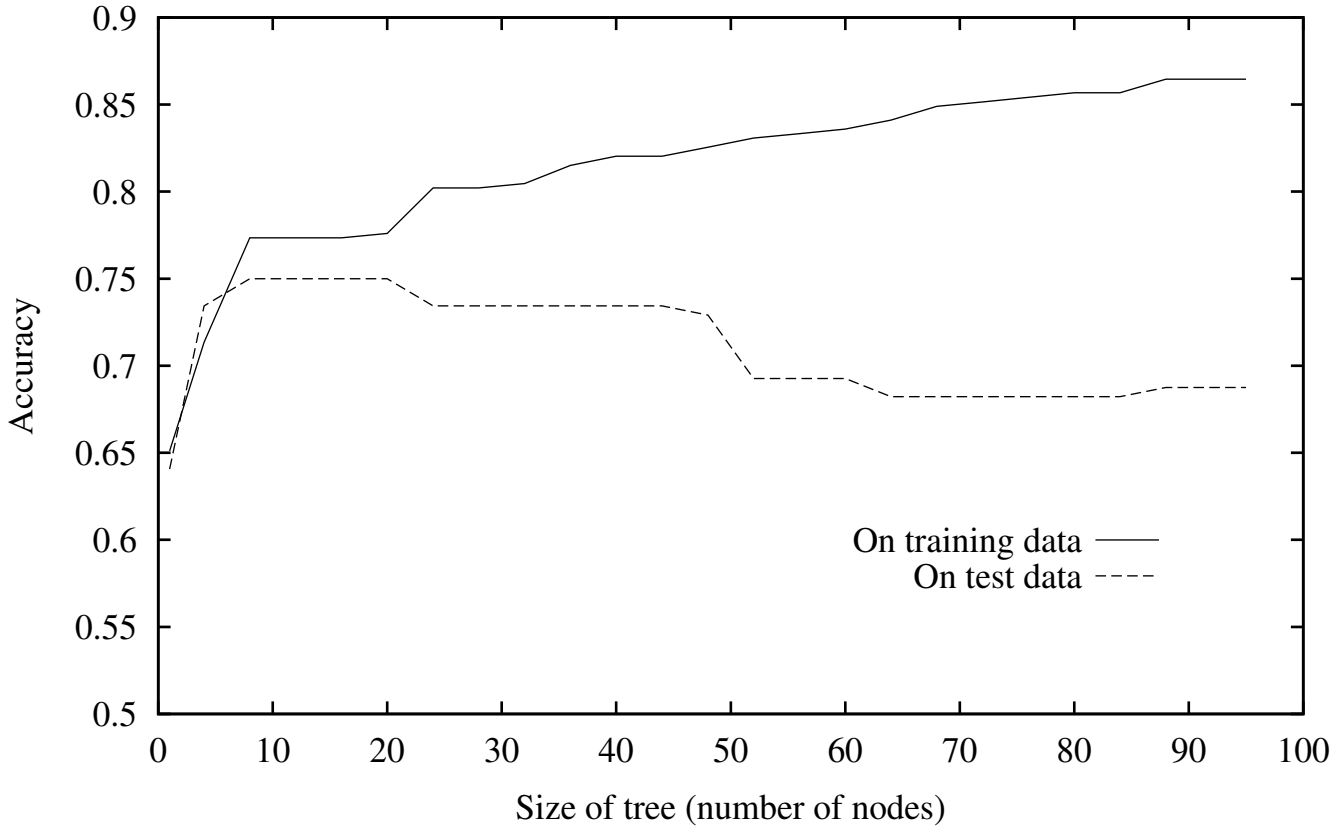
- A: candidate attribute
- v: possible values of A
- S: Set of examples $\{X\}$ at the node
- S_v: subset where $X_A = v$

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

Overfitting in Decision Trees

- Can always classify training examples perfectly
  - If necessary, keep splitting until each node contains 1 example
  - Singleton subset (leaf nodes with one example) which is by definition pure
- But this is not always ideal
  - In leaf nodes with singleton subsets, you have no confidence in your decision

Overfitting in Decision Tree Learning



Overfitting in General

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Definition
Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

And

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Avoiding Overfitting

修剪

How can we avoid overfitting? **Pruning**

- **pre-pruning:** stop growing when data split not statistically significant
- **post-pruning:** grow full tree, then remove sub-trees which are overfitting (based on validation set)

Post-pruning avoids problem of “early stopping”

Avoiding Overfitting

Pre-pruning

- Can be based on statistical significance test
- Stops growing the tree when there is no statistically significant association between any attribute and the class at a particular node
- For example, in ID3: chi-squared test plus information gain
  - only statistically significant attributes were allowed to be selected by information gain procedure

Avoiding Overfitting

Pre-pruning

- Simplest approach: stop growing the tree when fewer than some lower-bound on the number of examples at a leaf
- In C4.5, this parameter is the m parameter
- In sklearn, this parameter is $\min\_samples\_leaf$
- In sklearn, the parameter $\min\_impurity\_decrease$ enables stopping when this falls below a lower-bound

Avoiding Overfitting

- Other pre-pruning approaches:
  - Maximum number of leaf nodes (in *sklearn*, $\max\_leaf\_nodes)$
  - Minimum number of samples to split (in *sklearn* $\min\_samples\_split\ )$
  - Maximum depth (in *sklearn*, $\max\_depth)$

Avoiding Overfitting

Early stopping

- Pre-pruning may suffer from early stopping: may stop the growth of tree prematurely
- Classic example: XOR/Parity-problem
    - No individual attribute exhibits a significant association with the class
    - Target structure only visible in fully expanded tree
    - Pre-pruning won't expand the root node
- But: XOR-type problems not common in practice
- And: pre-pruning faster than post-pruning

Avoiding Overfitting

Post-pruning

- Builds full tree first and prunes it afterwards
    - Attribute interactions are visible in fully-grown tree
- Problem: identification of subtrees and nodes that are due to chance effects
- Subtree replacement
- Possible strategies: error estimation, significance testing, MDL principle
- We examine reduced-error Pruning

Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

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

Reduced-Error Pruning

- **Good** produces smallest version of most accurate subtree
- **Not so good** reduces effective size of training set

Effect of Reduced-Error Pruning