Machine Learning and Data Mining
(IT4242E)

Quang Nhat NGUYEN

quang.nguyennhat@hust.edu.vn

Hanoi University of Science and Technology

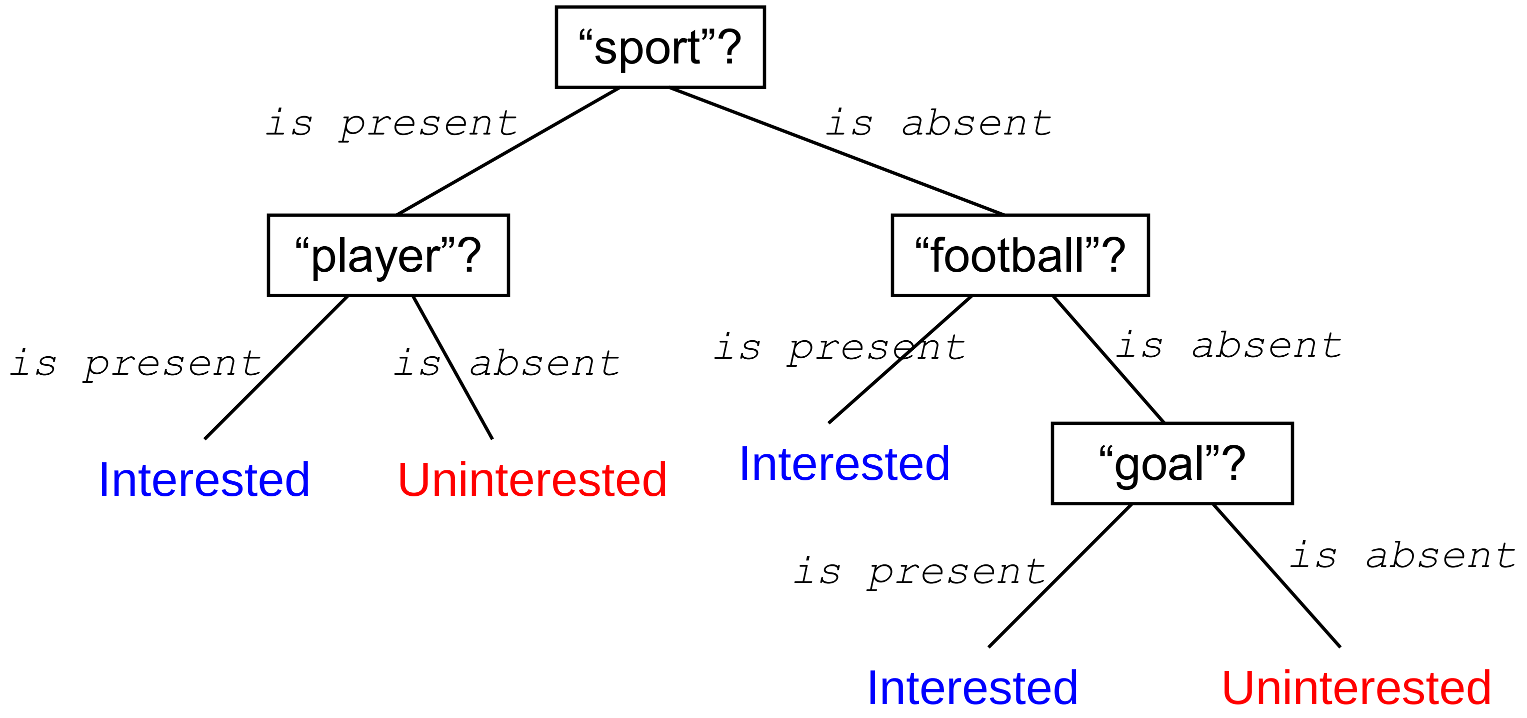
School of Information and Communication Technology

Academic year 2021-2022

The course's content:

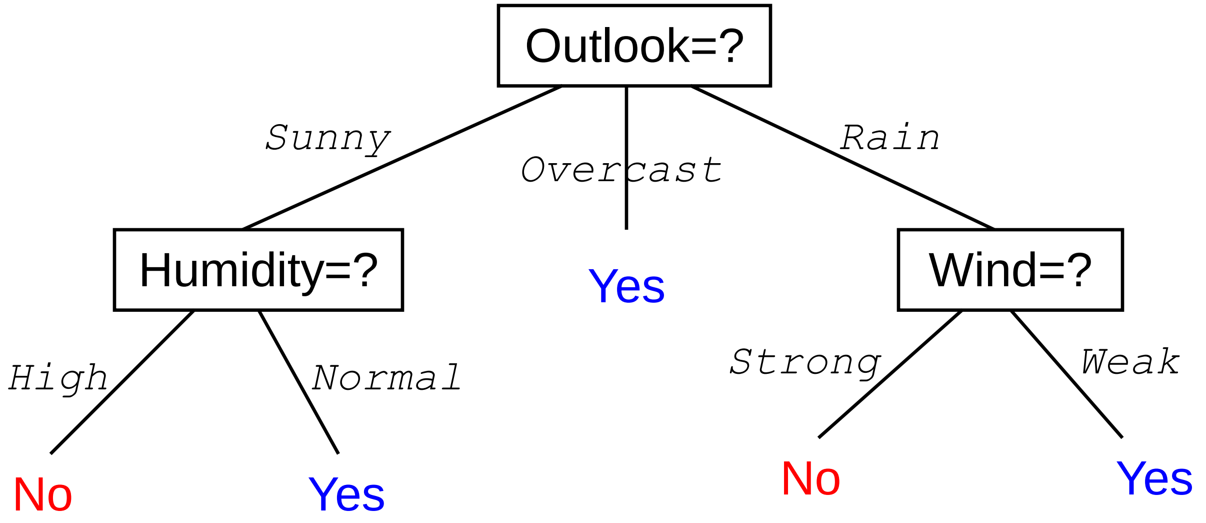
- Introduction
- Performance evaluation of the ML/DM system
- Regression problem
- **Classification problem**
  - **Decision tree learning**
- Clustering problem
- Association rule mining problem

Example of a DT: Which documents are of my interest?



- (…,"sport",…,"player",…) → Interested
- (…,"goal",…) → Interested
- (…,"sport",…) → Uninterested

Example of a DT: Does a person play tennis?



- (Outlook=`Overcast`, Temperature=`Hot`, Humidity=`High`, Wind=`Weak`) → Yes
- (Outlook=`Rain`, Temperature=`Mild`, Humidity=`High`, Wind=`Strong`) → No
- (Outlook=`Sunny`, Temperature=`Hot`, Humidity=`High`, Wind=`Strong`) → No

Decision tree – Introduction

- Decision tree (DT) learning
  - To approximate a ***discrete-valued target function***
  - The target function is represented by a decision tree
- A DT can be represented (interpreted) as a set of IF-THEN rules (i.e., easy to read and understand)
- Capable of learning disjunctive expressions
- DT learning is robust to noisy data
- One of the most widely used methods for inductive inference
- Successfully applied to a range of real-world applications

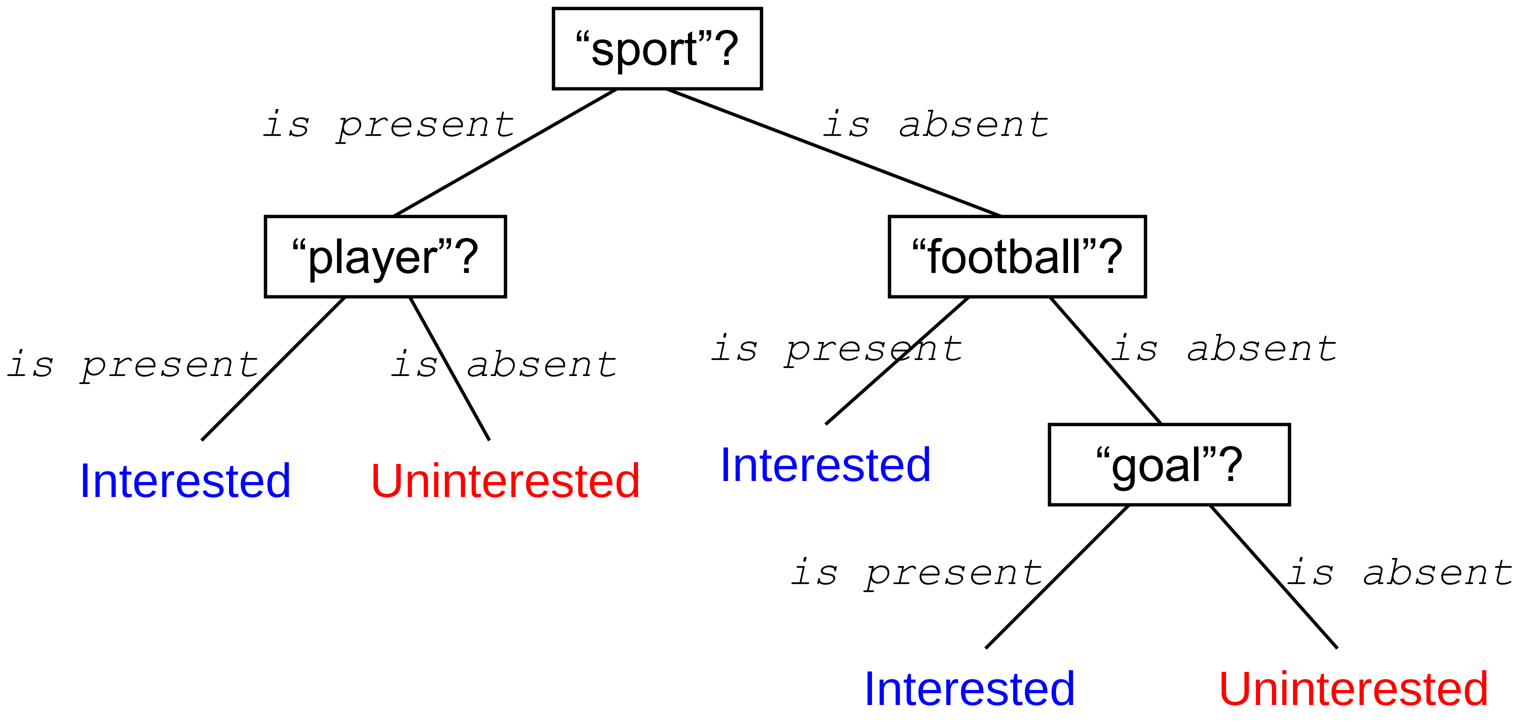
Decision tree – Representation (1)

- Each **internal node** represents an **attribute** *to be tested* by instances
- Each **branch** from a node corresponds to *a possible value of the attribute* associated with that node
- Each **leaf node** represents a *classification* (e.g., a class label)
- **A learned DT classifies an instance** by sorting it down the tree, from the root to some leaf node
  - → The classification associated with the leaf node is used for the instance

Decision tree – Representation (2)

- A DT represents a disjunction of conjunctions of constraints on the attribute values of instances
- Each **path** from the root to a leaf corresponds to a conjunction of attribute tests
- The tree itself is a disjunction of these conjunctions
- Examples
  - → Let's consider the two previous example DTs…

Which documents are of my interest?

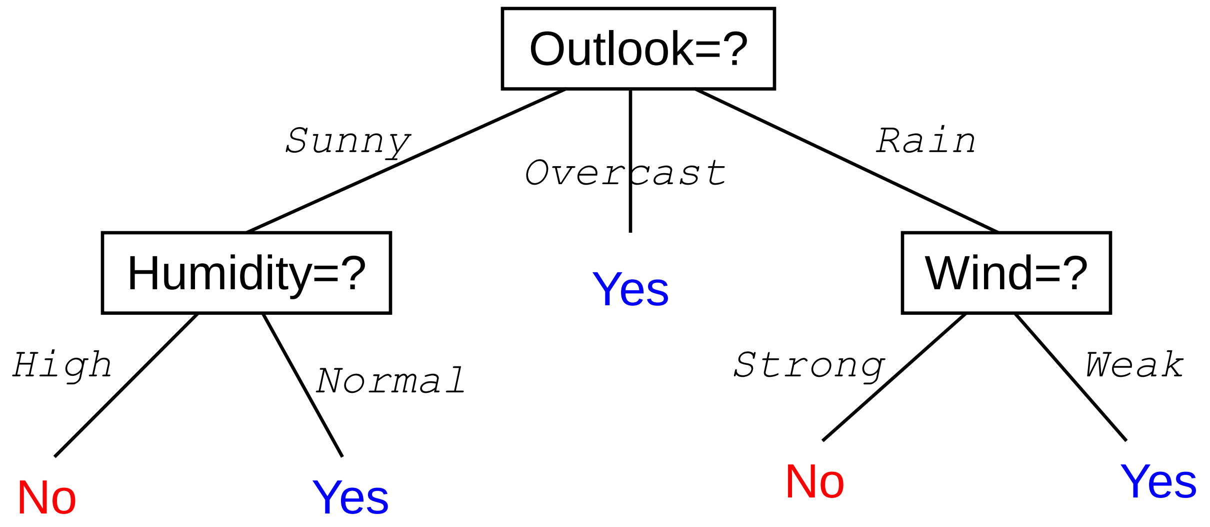


[("sport" is present) ∧ ("player" is present)] ∨

[("sport" is absent) ∧ ("football" is present)] ∨

[("sport" is absent) ∧ ("football" is absent) ∧ ("goal" is present)]

Does a person play tennis?



[(Outlook=`Sunny`) $\wedge$ (Humidity=`Normal`)] $\vee$

(Outlook=`Overcast`) $\vee$

[(Outlook=`Rain`) $\wedge$ (Wind=`Weak`)]

Decision tree learning – ID3 alg. (1)

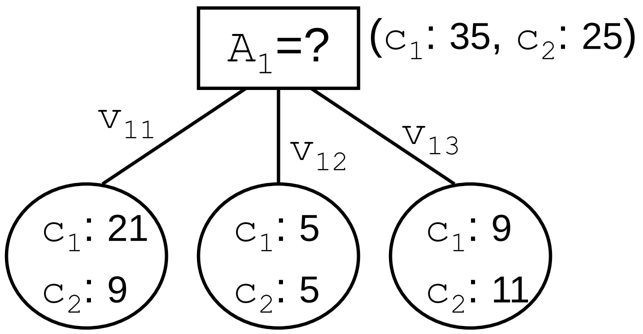
- Perform a **greedy search** through the space of possible DTs
- Construct (i.e., learn) a DT in a **top-down** fashion, starting from its root node
- At each node, the **test attribute** is the one (of the candidate attributes) that best classifies the training instances associated with the node
- A descendant (sub-tree) of the node is created **for each possible value of the test attribute**, and the training instances are sorted to the appropriate descendant node
- Every attribute can **appear at most once** along any path of the tree
- The tree growing process continues
    - Until the (learned) DT perfectly classifies the training instances, or
    - Until all the attributes have been used

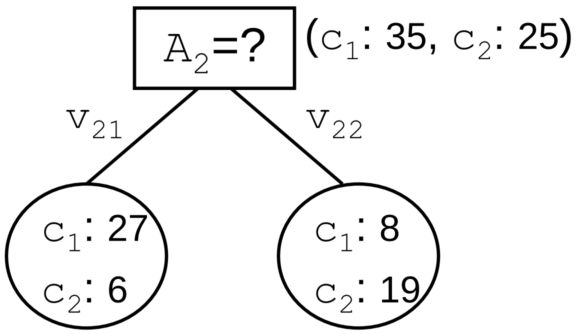
Decision tree learning – ID3 alg. (2)

```
ID3_alg(Training_Set, Class_Labels, Attributes)
Create a node Root for the tree
If all instances in Training_Set have the same class label c, Return the tree of the
    single-node Root associated with class label c
If the set Attributes is empty, Return the tree of the single-node Root associated with
    class label ≡ Majority_Class_Label(Training_Set)
A ← The attribute in Attributes that "best" classifies Training_Set
The test attribute for node Root ← A
For each possible value v of attribute A
    Add a new tree branch under Root, corresponding to the test: "value of attribute A is v"
    Compute Training_Set_v = {instance x | x ⊆ Training_Set, x_A=v}
    If (Training_Set_v is empty) Then
        Create a leaf node with class label ≡ Majority_Class_Label(Training_Set)
        Attach the leaf node to the new branch
    Else  Attach to the new branch the sub-tree  ID3_alg(Training_Set_v,
        Class_Labels, {Attributes \ A})
Return  Root
```

Selection of the test attribute

- A very important task in DT learning: at each node, how to choose the test attribute?
- To select the attribute that is most useful for classifying the training instances associated with the node
- How to measure an attribute's capability of separating the training instances according to their target classification
  - → Use a statistical measure – *Information Gain*
- Example: A two-class (c_1, c_2) classification problem
  - → Which attribute, A_1 or A_2, should be chosen to be the test attribute?





Entropy

- A commonly used measure in the Information Theory field
- To measure the impurity (inhomogeneity) of a set of instances
- The entropy of a set `S` relative to a `c`-class classification

$$Entropy(S) = \sum_{i=1}^{c} - p_i . \log_2 p_i$$

  where p_i is the proportion of instances in `S` belonging to class `i`, and $0.\log_2 0 = 0$

- The entropy of a set `S` relative to a two-class classification

$$Entropy(S) = -p_1.\log_2 p_1 - p_2.\log_2 p_2$$

- Interpretation of entropy (in the Information Theory field)
  → The entropy of `S` specifies the expected number of bits needed to encode class of a member randomly drawn out of `S`
    - Optical length code assigns $-\log_2 p$ bits to message having probability `p`
    - The expected number of bits needed to encode a class: $p.\log_2 p$

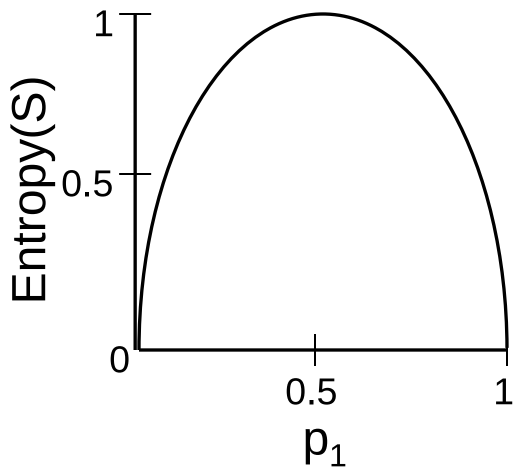
Entropy – Two-class example

- S contains 14 instances, where 9 belongs to class c_1 and 5 to class c_2
- The entropy of S relative to the two-class classification:

  $$\text{Entropy}(S) = -(9/14).\log_2(9/14) - (5/14).\log_2(5/14) \approx 0.94$$

- Entropy =0, if all the instances belong to the same class (either c_1 or c_2)

  →Need 0 bit for encoding (no message need be sent)



- Entropy =1, if the set contains equal numbers of c_1 and c_2 instances

  → Need 1 bit per message for encoding (whether c_1 or c_2)
- Entropy = some value in (0,1), if the set contains unequal numbers of c_1 and c_2 instances

  → Need on average <1 bit per message for encoding

Information gain

- Information gain of an attribute relative to a set of instances is
  - the expected reduction in entropy
  - caused by partitioning the instances according to the attribute
- Information gain of attribute `A` relative to set `S`

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

where `Values(A)` is the set of possible values of attribute `A`, and

$S_v = \{x \mid x \in S, x_A = v\}$

- In the above formula, the second term is the expected value of the entropy after `S` is partitioned by the values of attribute `A`
- Interpretation of `Gain(S,A)`: The number of bits saved (reduced) for encoding class of a randomly drawn member of `S`, by knowing the value of attribute `A`

Example training set

Let’s consider the following dataset (of a person) `S`:

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

`[Mitchell, 1997]`

Information gain – Example

- What is the information gain of attribute `Wind` relative to the training set `S` – `Gain(S,Wind)`?
- Attribute `Wind` have two possible values: `Weak` and `Strong`
- `S` = {9 positive and 5 negative instances}
- S_{weak} = {6 pos. and 2 neg. instances having `Wind=Weak`}
- S_{strong} = { 3 pos. and 3 neg. instances having `Wind=Strong`}

$$Gain(S,Wind) = Entropy(S) - \sum_{v \in \{Weak,Strong\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

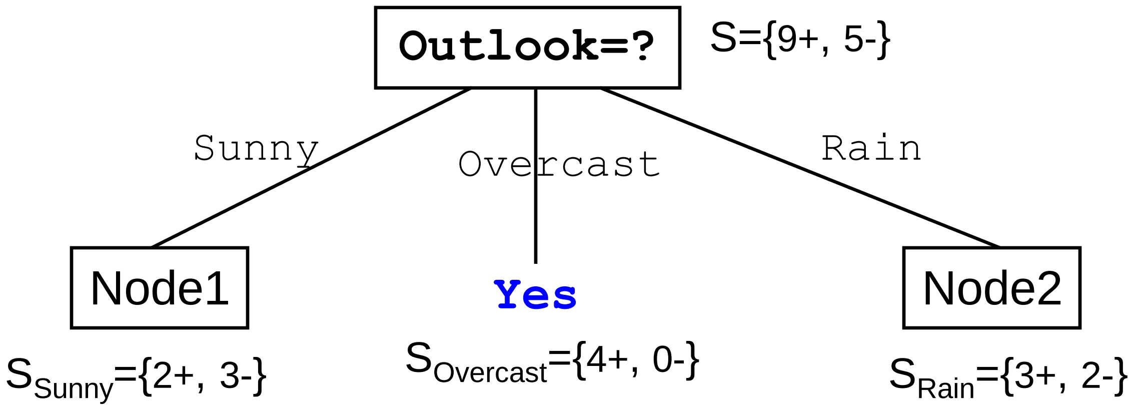
$$= Entropy(S) - (8/14).Entropy(S_{Weak}) - (6/14).Entropy(S_{Strong})$$

$$= 0.94 - (8/14).(0.81) - (6/14).(1) = 0.048$$

Decision tree learning – Example (1)

- At the root node, which attribute of {`Outlook`, `Temperature`, `Humidity`, `Wind`} should be the test attribute?
  - `Gain(S, `**`Outlook`**`) = ... = `**`0.246`**
  - `Gain(S, Temperature) = ... = 0.029`
  - `Gain(S, Humidity) = ... = 0.151`
  - `Gain(S, Wind) = ... = 0.048`

→So, `Outlook` is chosen as the test attribute for the root node!



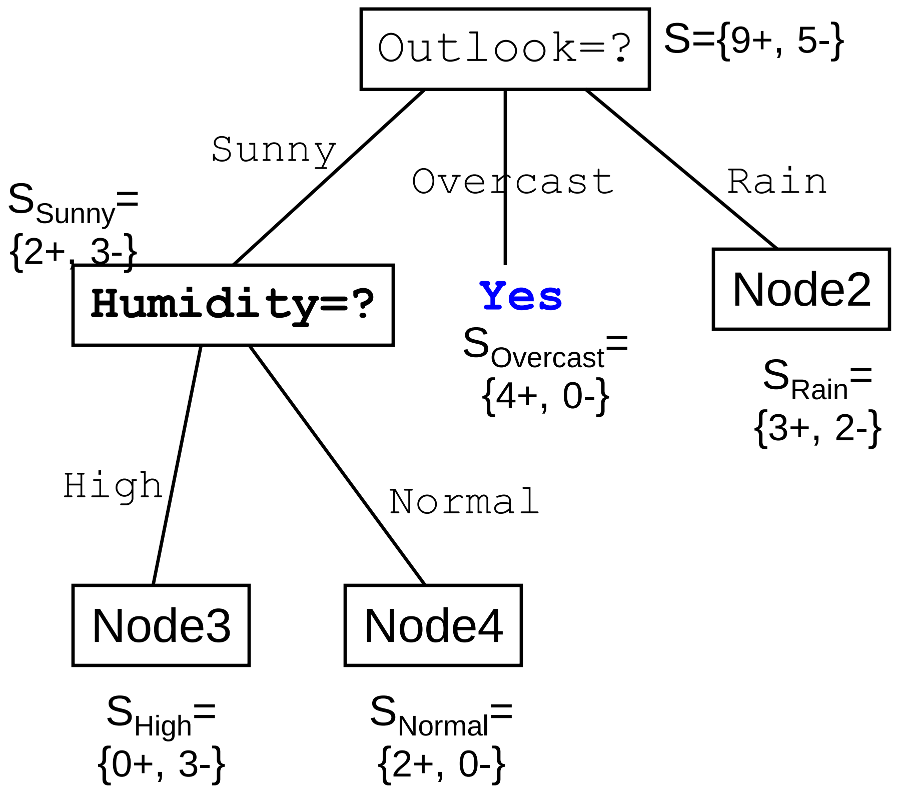
Decision tree learning – Example (2)

- At `Node1`, which attribute of {`Temperature`, `Humidity`, `Wind`} should be the test attribute?

  Note! Attribute `Outlook` is excluded, since it has been used by `Node1`'s parent (i.e., the root node)

  - Gain(S_{Sunny}, `Temperature`) =...= 0.57
  - Gain(S_{Sunny}, **`Humidity`**) = ... = **0.97**
  - Gain(S_{Sunny}, `Wind`) = ... = 0.019

→So, `Humidity` is chosen as the test attribute for `Node1`!



DT learning – Search strategy (1)

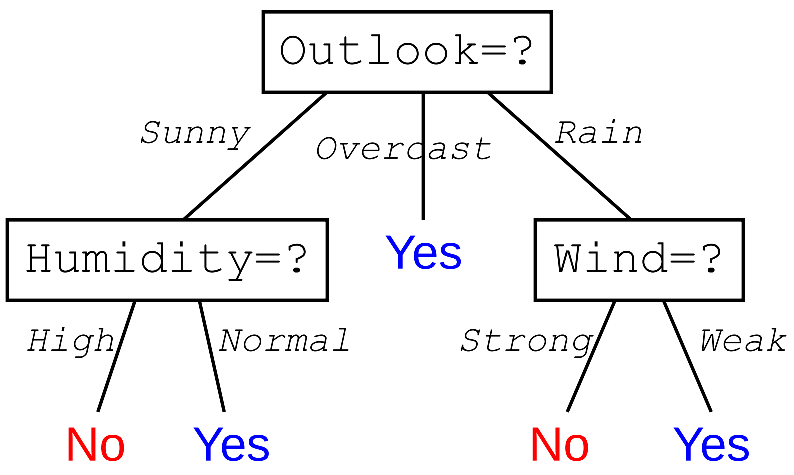
- ID3 searches in the hypotheses space (i.e., possible decision trees) for a decision tree that fits the training examples
- ID3 implements a search strategy from simple to comlex, starting with an empty tree
- ID3’s search process is controlled (guided) by the Information Gain evaluation metric
- ID3 searches *just one* (not all possible) decision tree that fits the training examples

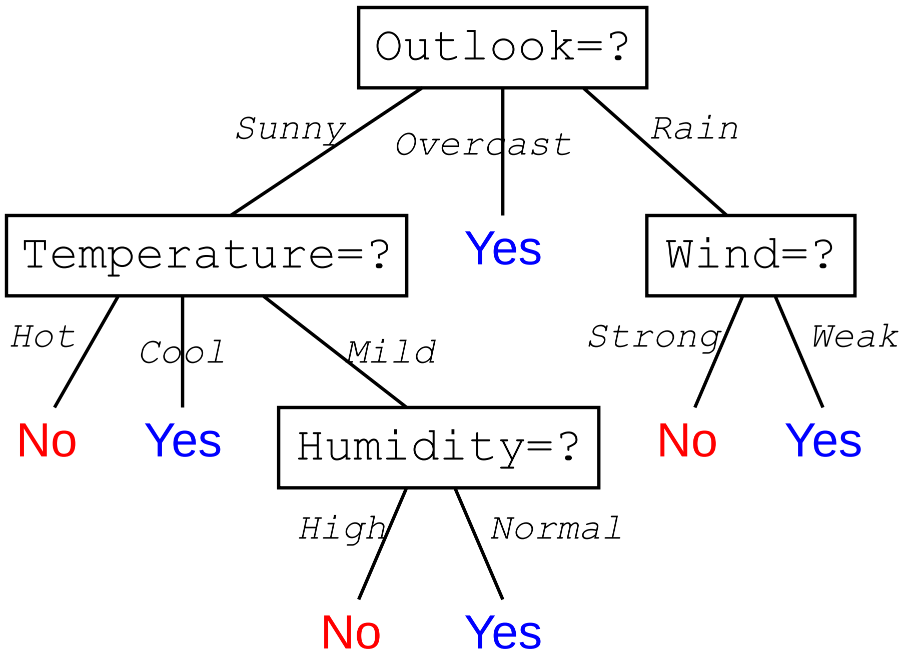
DT learning – Search strategy (2)

- In the search process, ID3 does not backtrack
  - → Guaranteed to find a locally optimal solution – Not guaranteed to find the globally optimal solution
  - → Once an attribute is selected as the test attribute for a node, ID3 never backtracks to reconsider that selection
- At each step of the search process, ID3 uses a statistical evaluation (i.e., Information Gain) to improve the current hypothesis
  - → The search process (for a solution) is less influenced by the errors of a few training examples (if there are)

Inductive bias in DT learning (1)

- Both the two DTs below are consistent with the given training dataset
- So, which one is preferred (i.e., selected) by the ID3 algorithm?





Inductive bias in DT learning (2)

- Given a set of training instances, there may be many DTs consistent with these training instances
- So, which of these candidate DTs should be chosen?
- ID3 chooses the **first acceptable DT** it encounters in its simple-to-complex, hill-climbing search
  - →Recall that ID3 searches incompletely through the hypothesis space (i.e., **without backtracking**)
- ID3's search strategy
  - Select in favor of **shorter trees** over longer ones
  - Select trees that place the **attributes with highest information gain** closest to the root node

Issues in DT learning

- Over-fitting the training data
- Handling continuous-valued (i.e., real-valued) attributes
- Choosing appropriate measures for test attribute selection
- Handling training data with missing-value attributes
- Handling attributes with differing costs

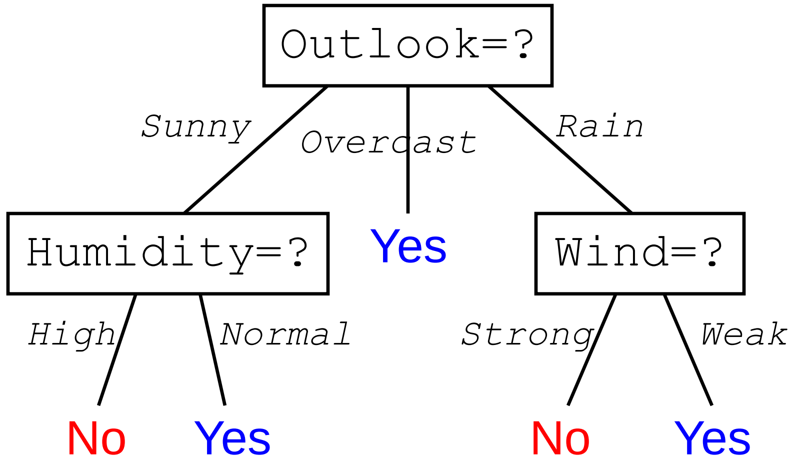
→ An extension of the ID3 algorithm with the above mentioned issues resolved results in the C4.5 algorithm

Over-fitting in DT learning (1)

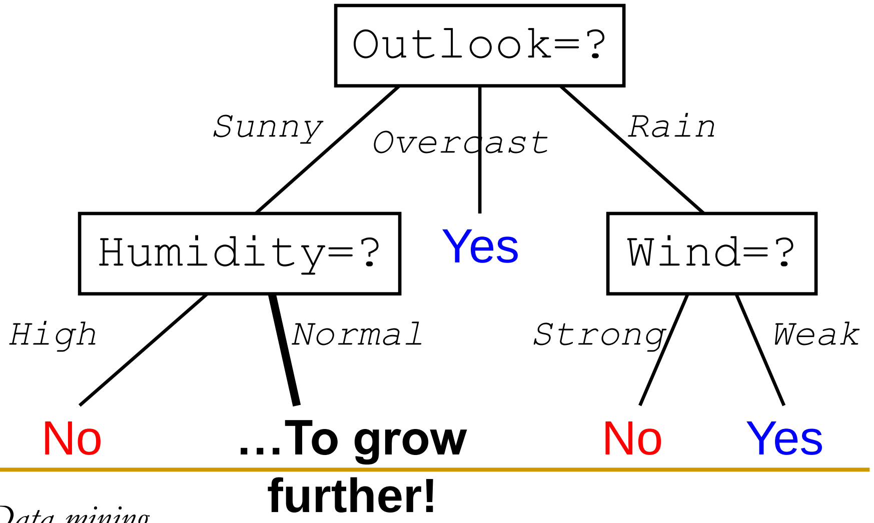
- Is a decision tree, which perfectly fits the training set, an optimal solution?
- If the training set contains some noise/error…?

Example: A noise/error example (i.e., this example has the true label Yes, but is wrongly annotated *No*):

(Outlook=Sunny, Temperature=Hot, Humidity=Normal, Wind=Strong, PlayTennis=**No**)

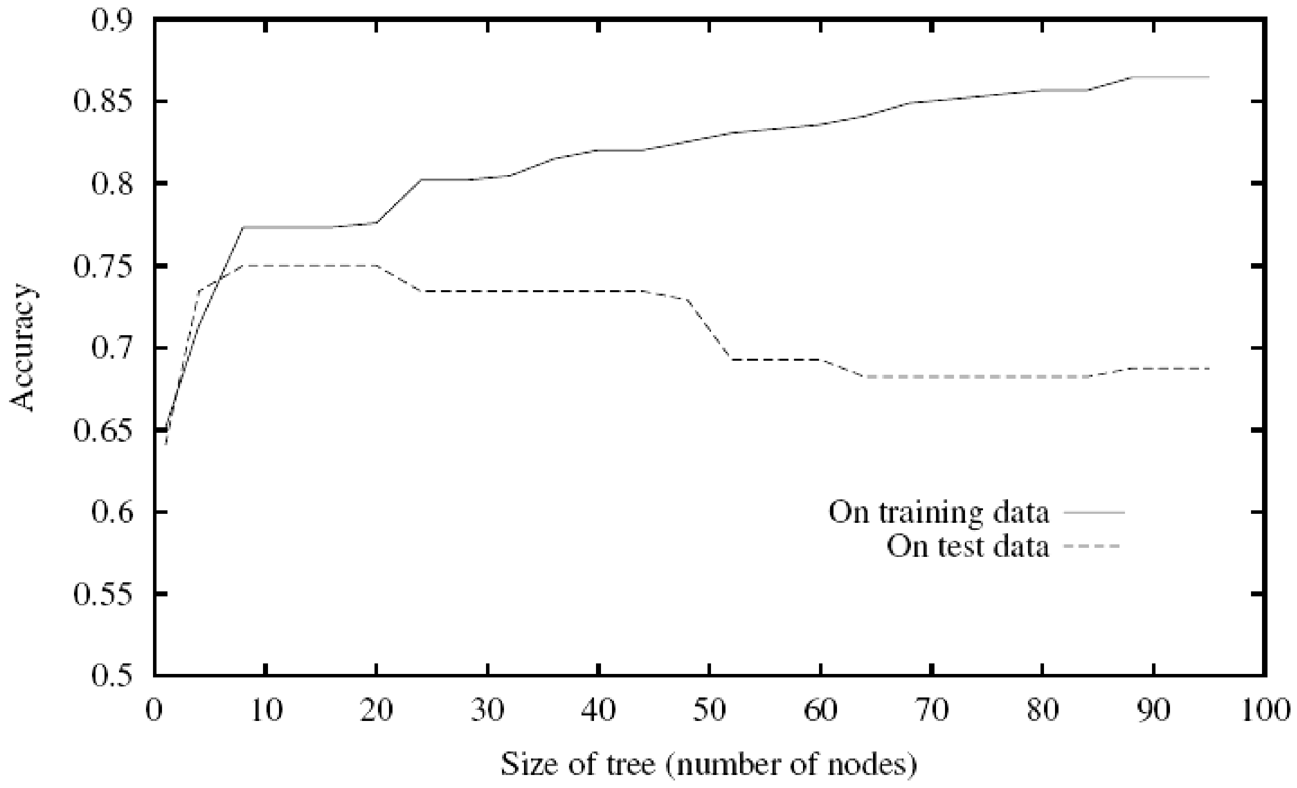


To learn a *more complex* decision tree!
(just because of the noise/error example)



Over-fitting in DT learning (2)

Extending the DT learning process will decrease the accuracy on the test set despite increasing the accuracy on the training set



[*Mitchell, 1997*]

Solutions to over-fitting (1)

- 2 strategies
  - Stopping the learning (i.e., growing) of the decision tree earlier, prior to reaching a decision tree that perfectly fits (i.e., classifies) the training set
  - To learn (i.e., grow) a complete tree (i.e., corresponding to a DT that perfectly fits the training set), and then to post-prune the (complete) tree
- The strategy of post-pruning over-fit trees is often more effective in practice

  → Reason: For the "early-stopping" strategy, the DT learning needs to determine precisely *when to stop the learning* (i.e., growing) of the tree – Difficult to determine!

Solutions to over-fitting (2)

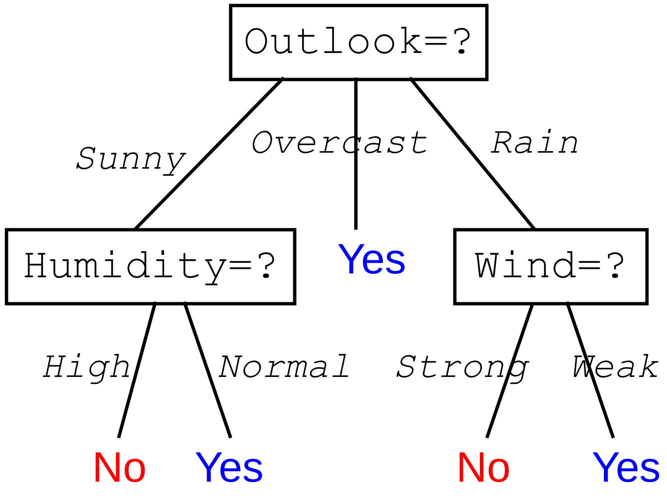
- How to select an "appropriate" size of the decision tree?
  - To evaluate the classification performance on a validation set
    - This is the most often used approach
    - There are 2 main approaches: *Reduced-error pruning* and *Rule post-pruning*
  - To apply a statistical evaluation (e.g., chi-square test) to check if an expansion (or a pruning) of a tree node can improve the performance
  - To evaluate the complexity of the encoding (i.e., representation of) the training examples and the decision tree, and stop the learning (i.e., growing) the decision tree when the length of this encoding is minimum
    - Based on the Minimum Description Length (MDL) theory
    - To minimize: `size`(tree) + `size`(misclassifications(tree))

Reduced-error pruning

- Each node of the (perfectly fit) tree is checked for pruning
- A node is removed if the tree after removing that node achieves a performance <u>not worse than</u> the original tree for a validation set
- Pruning a node consists of the following tasks:
  - To prune completely the sub-tree associating to the pruned node,
  - To convert the pruned node to a leaf node (with a class label),
  - To associate with this leaf node a class label that most occurs amongst the training examples associated with that node
- To repeat the pruning nodes
  - Always select a node whose pruning maximizes the DT's classification capability on a validation set
  - Stop the pruning when a further pruning decreases the DT's classification capability von a validation set

Rule post-pruning

- To learn (i.e., grow) a decision tree that perfectly fits the training set
- Convert the representation of the learned DT to a corresponding set of rules (i.e., each rule for each path from the root node to a leaf node)
- To reduce (i.e., to generalize) each rule (i.e., independently to the other rules), by removing any conditions (in the IF part) that results in an improvement on classification efficiency of that rule
- To order the reduced rules according to classification efficiency, and to use this order for classification of future examples



IF (Outlook=`Sunny`) ∧ (Humidity=`Normal`)

THEN (PlayTennis=**`Yes`**)

Continuous-valued attributes

- To be converted to discrete-valued attributes, by partitioning the continuous values range into a set of non-overlapping intervals
- For a continuous-valued attribute `A`, create a new binary-value attribute A_v such that: A_v is true if `A>v`, and is false if otherwise
- How to determine the “best” threshold value `v`?
  - → To select a threshold value `v` that results in the highest value of *Information Gain*
- Example:
  - To arrange the training examples in an increasing order of `Temperature`
  - To determine those adjacent-but-different-class-label training examples
  - 2 possible threshold values: $\texttt{Temperature}_{54}$ and $\texttt{Temperature}_{85}$
  - The new binary-value attribute $\texttt{Temperature}_{54}$ is selected, because Gain($\texttt{S}$,$\texttt{Temperature}_{54}$) > Gain($\texttt{S}$,$\texttt{Temperature}_{85}$)

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

Alternative measures for test attribute sel.

- The measure *Information Gain* tends to

  $\rightarrow$ Prioritize attributes that have more values than those with less values

  Example: The attribute `Date` has a very large number of values

  - This attribute has a very high value of Information Gain
  - This attribute alone can perfectly classifies the training set (i.e., this attribute partitions the training examples into many (very) small-sized subsets
  - This attribute is selected as the test attribute for the root node (of a decision tree that has the depth of 1, but it is very wide and has lots of branches)

- Alternative measures: Gain Ratio

  $\rightarrow$To reduce the effect of multi-valued attributes

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

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

(where `Values(A)` a set of possible values for the attibute `A`, and S_v={x| x∈S, x_A=v})

Missing-valued attributes (1)

- Assume that an attribute `A` is a candidate for the test attribute of node `n`
- How to handle an example `x` that does not have value for the attribute `A` (i.e., x_A =null/undefined)?
- Let's denote S_n the set of the training examples that associates to the node `n` and have value for attribute `A`

→Solution 1: x_A is the most frequent value for the attribute `A` amongst the training examples in the set S_n

→Solution 2: x_A is the most frequent value for the attribute `A` amongst the training examples in the set S_n that have the same class label of the example `x`

Missing-valued attributes (2)

→Solution 3:

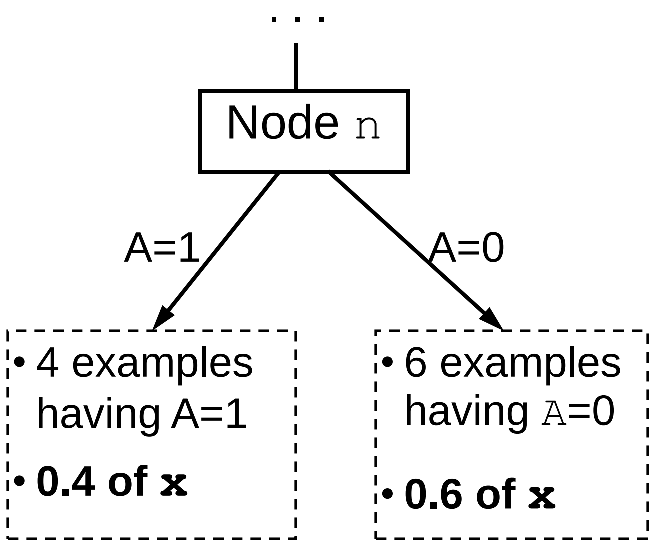
- First, to compute the probability p_v for each possible value v of the attribute A
- Then, to assign *this fraction* p_v *of the example* x for the corresponding branch of the node n
- *This fractional examples* are used to compute *Information Gain*

Example:

- A binary-valued (0/1) attribute A
- Node n consists of:
  - An example x missing value of A
  - 4 examples having A=1, and
  - 6 examples having A=0

$p(x_A=1) = 4/10 = 0.4$

$p(x_A=0) = 6/10 = 0.6$



Attributes having differing costs

- In some ML or DM problems, attributes may associate with differing costs (i.e., importance degrees)
  - Example: In a problem of learning to classify medical diseases, `BloodTest` costs $150, whereas `TemperatureTest` costs $10
- Trend of learning DTs
  - Use as many as possible low-cost attributes
  - Only use high-cost attributes if it is a must (i.e., in order to achieve reliable classifications)
- How to learn a DT employing lost-cost attributes?

  → To use alternative measures to IG for the selection of test attributes

$$\frac{Gain^2(S,A)}{Cost(A)}$$

[Tan and Schlimmer, 1990]

$$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$$

[Nunez, 1988; 1991]

(w ($\in$[0,1]) is a weight that balances the importance degrees between *Attribute cost* and *Information Gain*)

DT learning – When?

- Training examples are represented by pairs of attribute-value
  - Suitable for discrete-valued attributes
  - For continuous-valued attributes, to be discretized
- The target function's output is a discrete value
  - For example: To classify examples into appropriate class labels
- Very suitable if the target function is represented in a disjunctive form
- The training set may contain noise/error
  - Error in the class labels of the training examples
  - Error in the values of the attributes that represent the training examples
- The training set may contain missing-value examples
  - For some training examples, their values of a certain attribute is undefined/unknown

References

- T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- M. Nunez. *Economic induction: A case study*. In Proceedings of the 3rd European Working Session on Learning, EWSL-88, pp.139-145. California: Morgan Kaufmann, 1988.
- M. Nunez. *The use of background knowledge in decision tree induction*. Machine Learning, 6(3): 231-250, 1991.
- M. Tan and J. C. Schlimmer. *Two case studies in cost-sensitive concept acquisition*. In Proceedings of the 8th National Conference on Artificial Intelligence, AAAI-90, pp.854-860, 1990.