Machine Learning

(Học máy – IT3190E)

Khoat Than

School of Information and Communication Technology

Hanoi University of Science and Technology

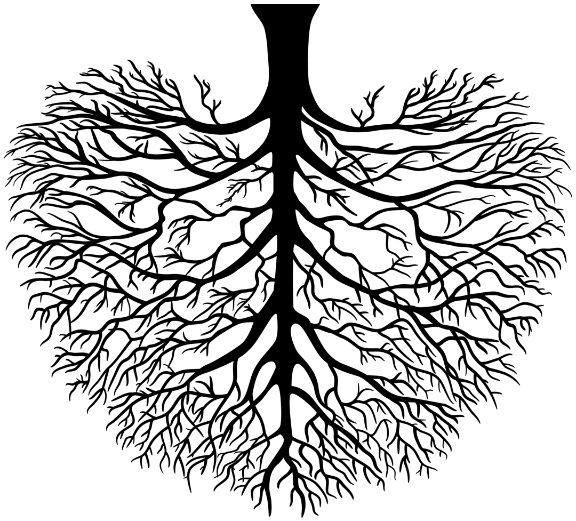
2023

Contents

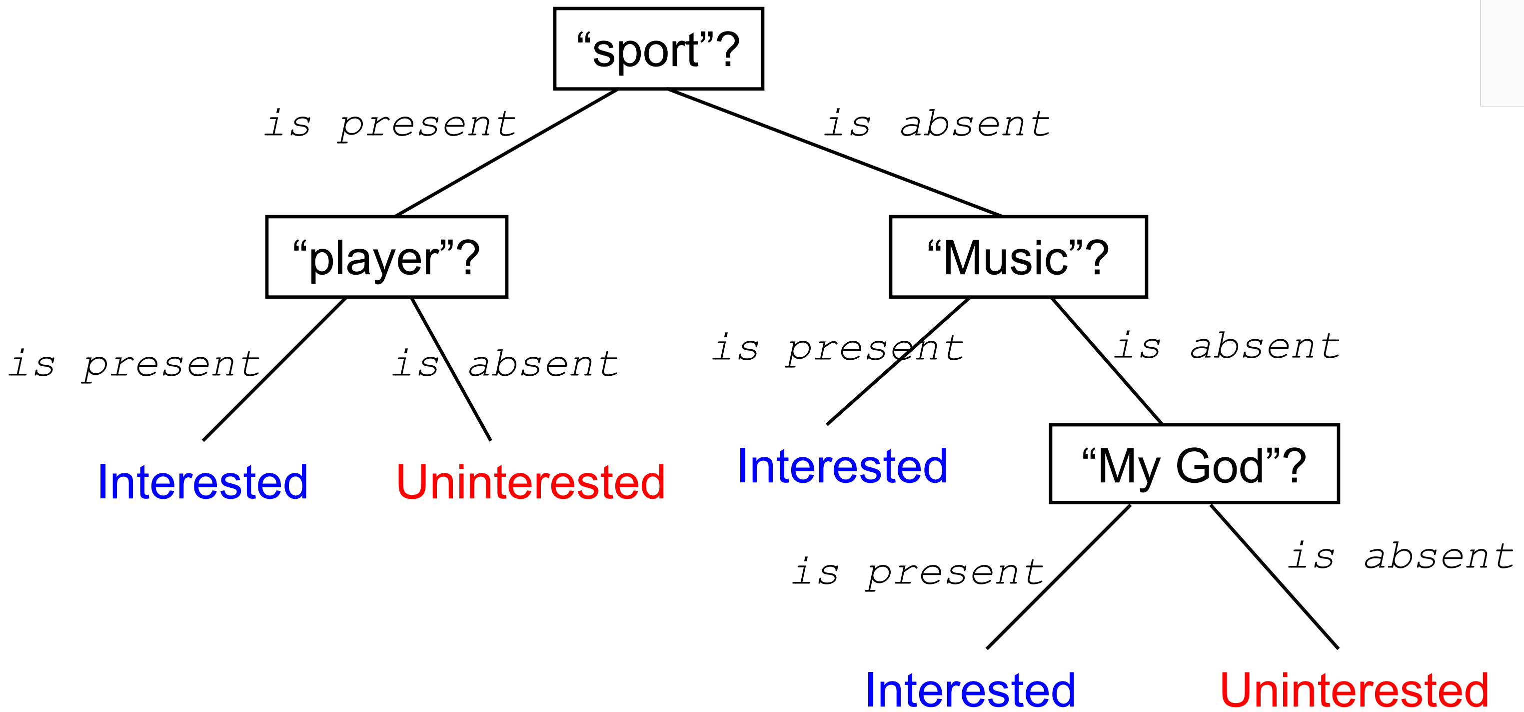
- Introduction to Machine Learning
- **Supervised learning**
  - **Decision tree and Random forest**
- Unsupervised learning
- Reinforcement learning
- Practical advice

1. Decision tree

- Decision tree
  - To represent a function by using a tree.
- Each decision tree can be interpreted as a set of rules of the form: IF-THEN
- Decision trees have been used in many practical applications.

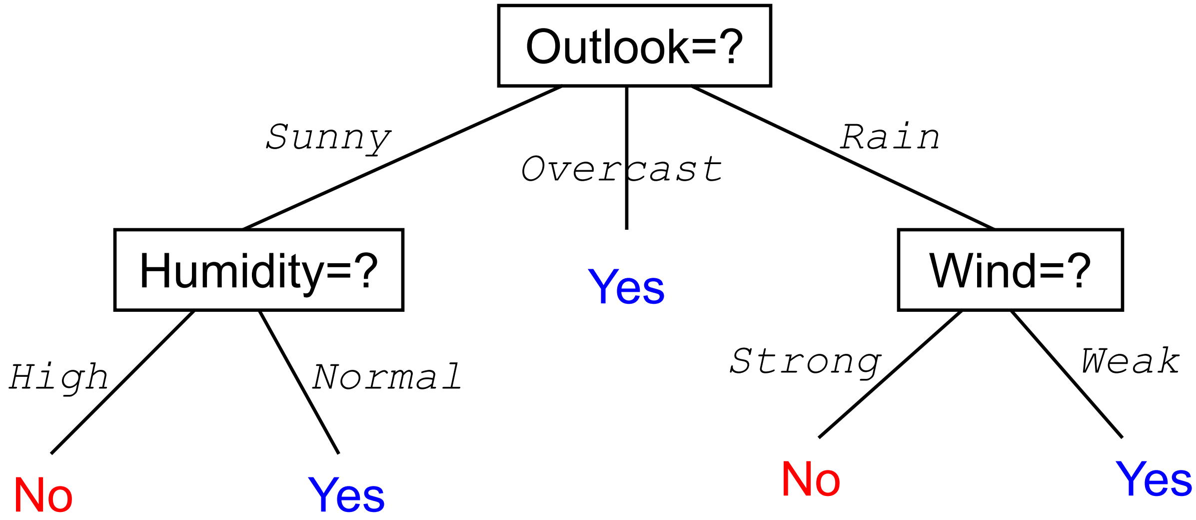


Examples of a decision tree (1)



- (…,"sport",…,"player",…) → Interested
- (…,"My God",…) → Interested
- (…,"sport",…) → Uninterested

Examples of a decision tree (2)



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

Classification problem

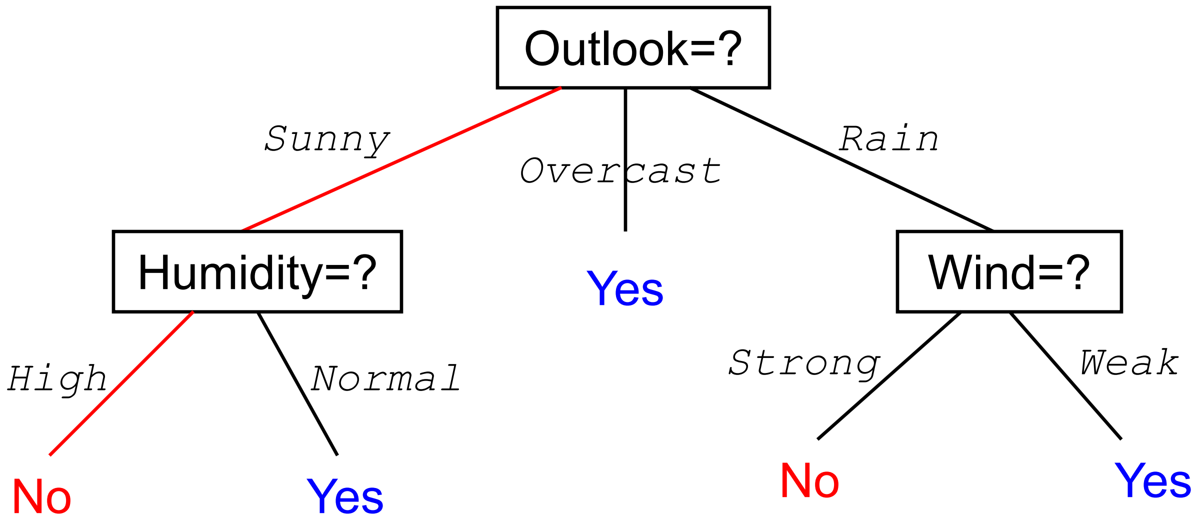
- Data representation:
  - Each observation is represented by n attributes/features, e.g., $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$.
  - Each attribute is **nominal/categorical**, i.e., represents names, labels or categories, e.g.,
    $x_{i1} \in \{high, normal\}$, $x_{i2} \in \{male, female, other\}$
  - There is a set C of predefined labels.
- We have to learn a function from a training dataset: $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$

Tree representation (1)

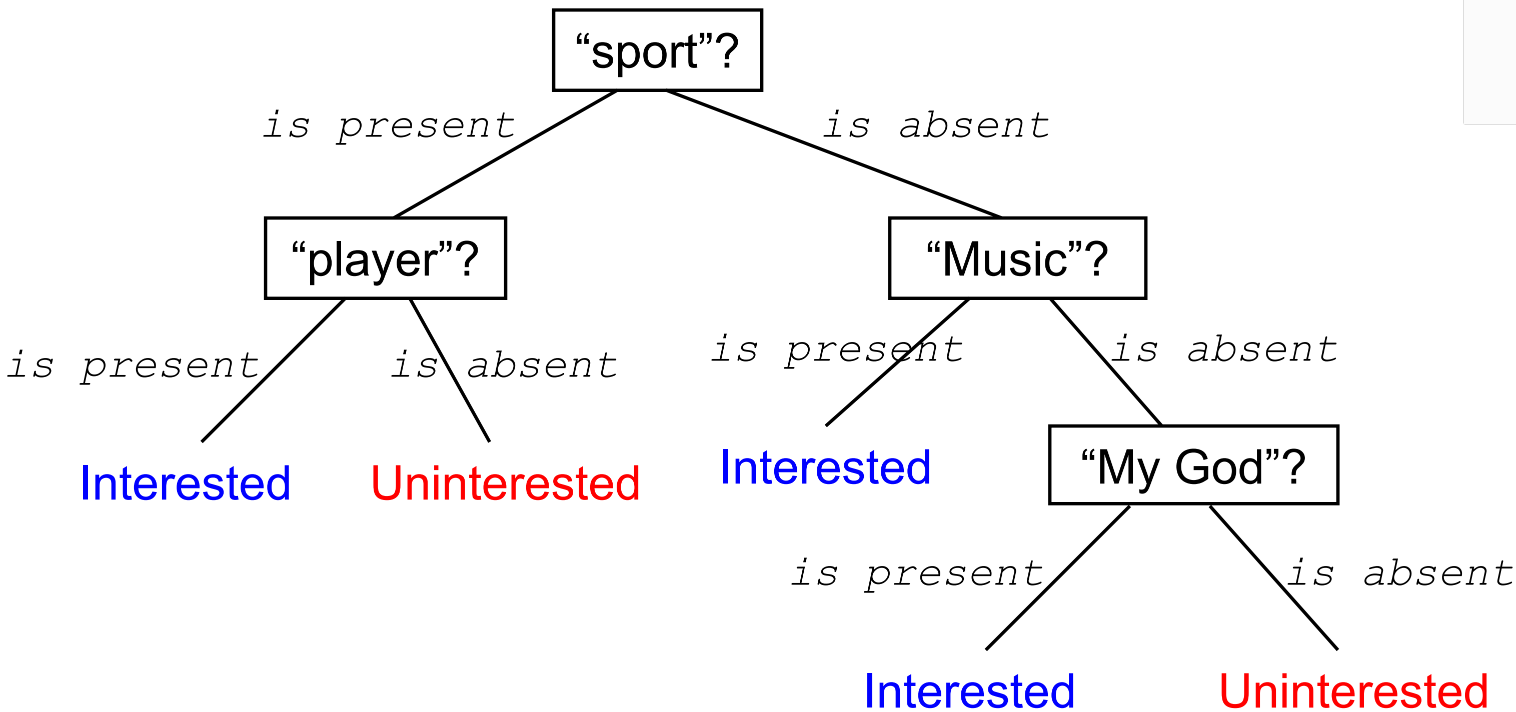
- *Each internal node* represents an attribute for testing the incoming data.
- Each *branch/subtree of a node* corresponds to a value of the attribute of that node.
- Each leaf node represents a class label.
- Once a tree has been learned, *we can predict the label for a new instance by using its attributes to travel from the root downto a leaf.*
  - The label of the leaf will be used to assign to the new instance.

Tree representation (2)

- Each path from the root to a leaf is a *conjunction/AND* of the attribute tests.
- A decision tree itself is a *disjunction/OR of those conjunctions*.



Representation by a disjunction



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

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

[("sport" is absent) ∧ ("Music" is absent) ∧ ("My God" is present)]

2. Learning a decision tree by ID3

- ID3 (Iterative Dichotomiser 3) is a greedy algorithm which was proposed by Ross Quinlan in 1986.
- It uses the top-down scheme.
- At each node N, select a test attribute A which can help us best do classification for the data in N.
  - *Generate a branch for each value of A, and then separate the data into its branches accordingly.*
- Grow the tree until:
  - *It classifies correctly all the training data; or*
  - *All the attributes are used.*
- Note: each attribute can only appear at most once in any path of the tree.

The ID3 algorithm

ID3_alg(*Training_Set, Class_Labels, Attributes*)

Generate the Root of the tree

If all of *Training_Set* belong to class c, then Return Root as leaf with label c

If *Attributes* is empty, then
Return Root as leaf with label c = **Majority_Class_Label**(*Training_Set*)

A ← a set of *Attributes* that are best discriminative for *Training_Set*

Let A be the test attributes of Root

For each value v of A

Generate a branch of Root which corresponds with v.

Determine $Training\_Set_v$ = { x in *Training_Set* | $x_A = v$}

If ($Training\_Set_v$ is empty) Then

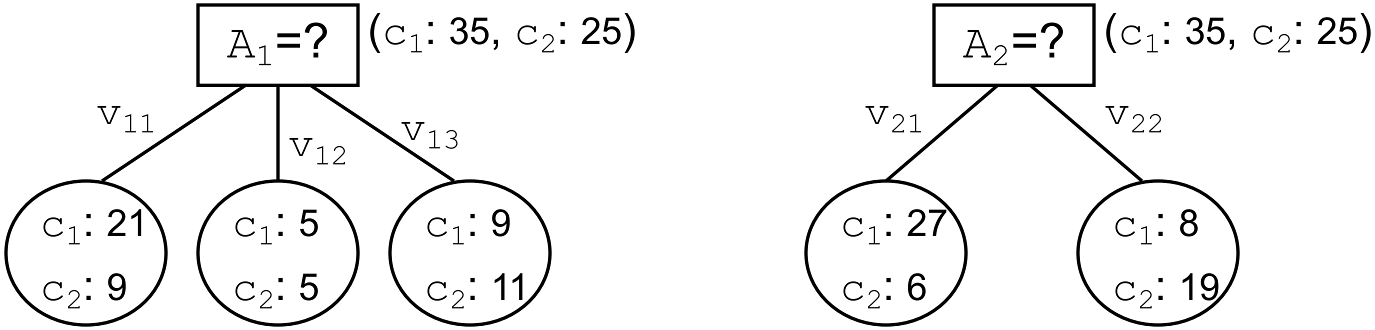
Generate a leaf with class label c = **Majority_Class_Label**(*Training_Set*)

Else
Generate a subtree by **ID3_alg**($Training\_Set_v$, Class_Labels, Attributes \{A})

Return Root

How to choose the test attributes?

- At each node, how can we choose a set of test attributes?
  - These attributes should be ***discriminative***, i.e., can help us classify well the data inside that node.
- How to know an attribute to be discriminative?
- Ex: assuming 2 classes in the data, which of A_1 and A_2 should be selected as the test attribute?



- ***Information gain*** can help.

Information gain: entropy

- Entropy measures the impurity/inhomogeneity of a set.
- Entropy of a set S with c classes can be defined as:

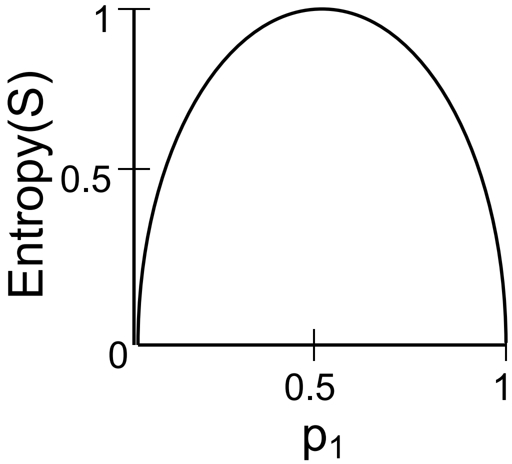
$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

  - Where p_i is the proportion of instances with class label i in S; and $0.\log_2 0 = 0$ as a convention; $p_1+p_2+\ldots+p_c = 1$
- For 2 classes: $entropy(S) = -\ p_1\log_2 p_1 - p_2\log_2 p_2$
- Meanings of entropy in Information Theory:
  - Entropy shows the *number of bits on average* to encode a class of S.
  - Entropy of a message measures the *average amount of information* contained in that message.
  - Entropy of a random variable x measures the *unpredictability* of x.

Information gain: entropy example

- S consists of 14 examples for which 9 belong to class c_1 and 5 belong to class c_2.
- So the entropy of S is:

  Entropy(S)
  $= -(9/14).\log_2(9/14) - (5/14).\log_2(5/14)$
  ≈ 0.94



- Entropy = 0 if all examples in S have the same label.
- Entropy = 1 if the two classes in S are equal in size.
- Otherwise, entropy will always belong to (0, 1).

Information gain

- *Information gain* of an attribute in S:
  - Measures the reduction of entropy if we divide S into subsets according to that attribute.
- Information gain of attribute A in S is defined as:

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

  - Where Values(A) is the set of all values of A, and $S_v = \{\mathbf{x} \mid \mathbf{x} \text{ in } S, \text{ and } x_a = v\}$
- The **second term** in Gain(S,A) measures the *information remained* when S is divided into subsets according to the values of A.
- Meaning of Gain(S,A): *the average amount of information is lost when dividing S according to A.*

Information gain: example (1)

- A set S of observations about a person playing tennis.

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 (2)

- What is Gain(S, Wind)?
- Wind has two values: Strong & Weak
- S = {9 examples with label Yes, 5 examples with label No}
- S_{Weak} = {6 examples with label Yes and 2 examples with label No, having Wind=Weak}
- S_{Strong} = {3 examples with label Yes, 3 examples with label No, having Wind=Strong}
- So:

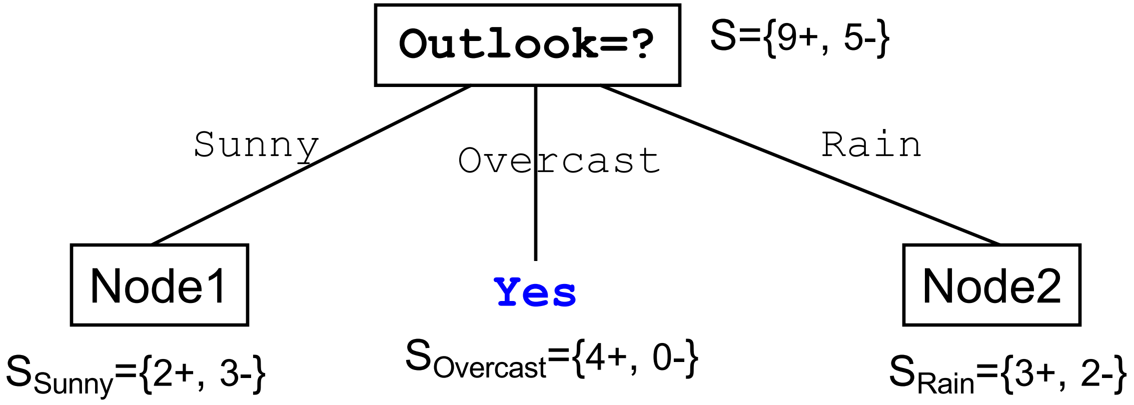
$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - \frac{8}{14} Entropy(S_{Weak}) - \frac{6}{14} Entropy(S_{Strong})$$

$$= 0.94 - \frac{8}{14} * 0.81 - \frac{6}{14} * 1 = 0.048$$

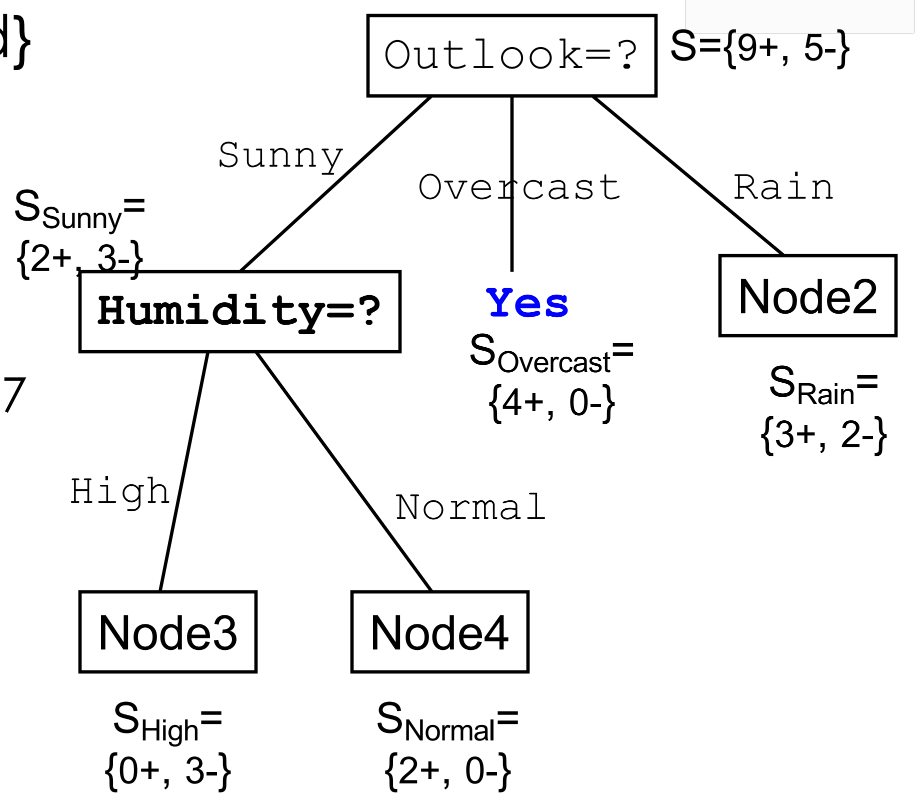
ID3: example (1)

- At the root, which one 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 selected as the test attribute.



ID3: example (2)

- At Node1, which one of {Temperature, Humidity, Wind} should be the test attribute?
  - Note: Outlook is left out
  - Gain(S_{Sunny}, Wind) = ... = 0.019
  - Gain(S_{Sunny}, Temperature) =...= 0.57
  - Gain(S_{Sunny}, **Humidity**) = ... = **0.97**
- So, Humidity is selected to divide Node1.

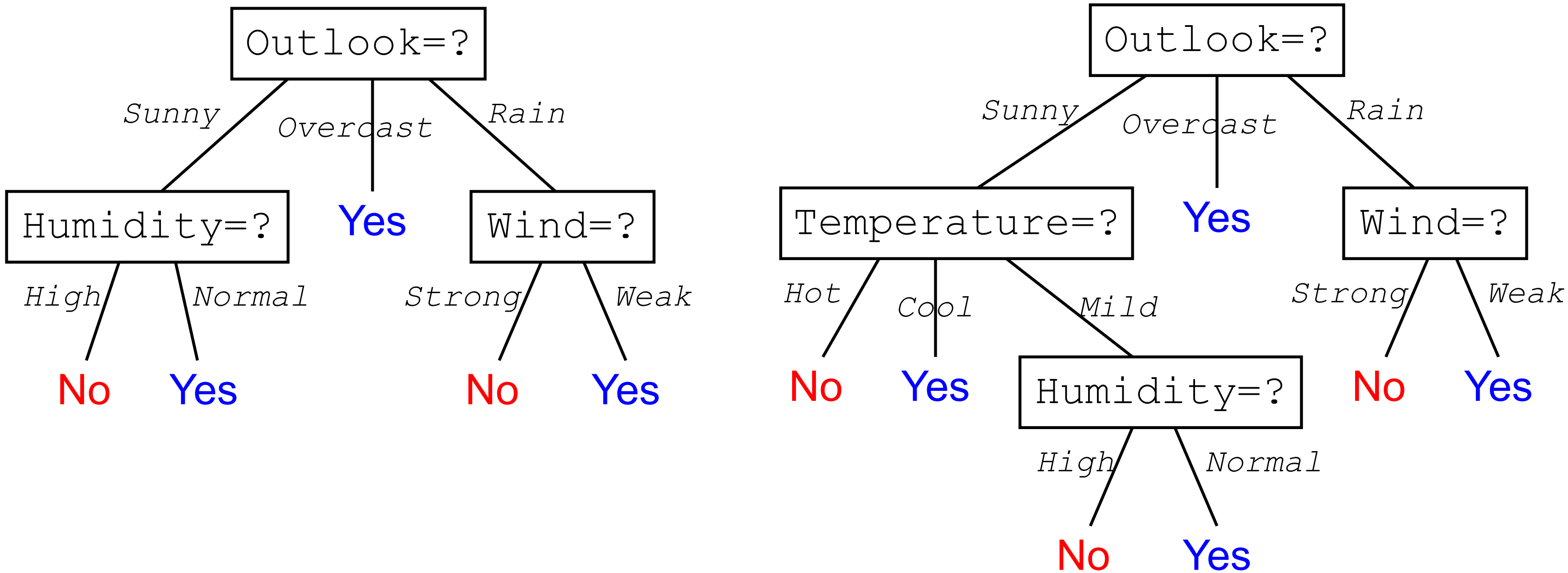


ID3: searching scheme (1)

- ID3 searches for a tree that fits well with the training data.
  - By growing the tree gradually.
- Information Gain decides the search direction of ID3.
- ID3 just searches for only one tree.
- ID3 never backtracks, as a consequence:
  - It can find a local optimal solution/tree.
  - Once an attribute has been selected, ID3 never rethinks of this choice.

ID3: searching scheme (2)

- For a training dataset, there might be many trees that fit well with it.
  - Which tree will be selected by ID3?



ID3: searching scheme (3)

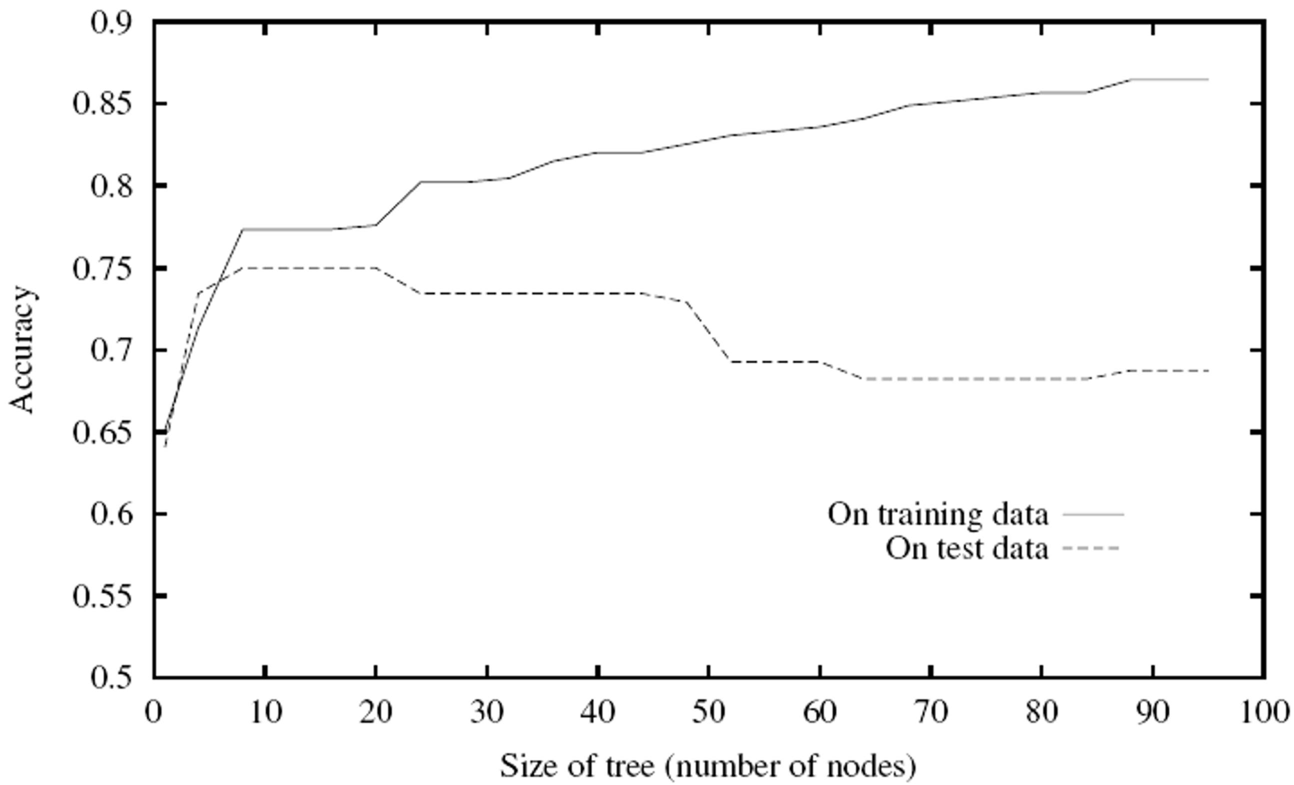
- ID3 selects the first tree that fits the training data,
  - Because it never reconsiders its choices when growing a tree.
- So, the searching scheme of ID3:
  - Prefers simple trees.
  - Prefers trees in which the attributes with higher information gain will be placed closer to the roots.

3. Some issues of ID3

- The learnt trees may overfit the training data.
- How to work with real attributes?
  - Many applications have real inputs.
- Is there any better measure than information gain?
- How to deal with missing values?
  - Missing-value is an inherent problem in many practical applications.
- How to enclose the cost of attributes in ID3?

Overfitting in ID3 (2)

- An example: continuing to grow the tree can improve the accuracy on the training data, but perform badly on the test data.



[*Mitchell, 1997*]

Overfitting: solutions

- 2 solutions:
  - *Stop learning early:* prevent the tree before it fits the training data perfectly.
  - *Prune the full tree:* grow the tree to its full size, and then post prune the tree.
- It is hard to decide when to stop learning.
- Post-pruning the tree empirically results in better performance. But
  - How to decide the good size of a tree?
  - When to stop pruning?
- We can use a validation set to do pruning, such as, *reduced-error pruning*, and *rule-post pruning*.

ID3: attribute selection

- Information gain:
  - Prefers the attribute that has more unique values.
  - Attributes with more unique values will be placed closer to the root than the other attribute.
- We can use some other measures, such as Gain Ratio

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

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

ID3: missing or real values

- How to work with real attributes?
  - Real attributes/features are popular in practice.
  - One way is to ***discretization***, i.e., transforming a real attribute into a discrete one by dividing the domain of that attribute into a set of intervals.
    Ex: [0, 1] → {[0, 0.25); [0.25, 0.5); [0.5, 0.75); [0.75, 1]}
- How to deal with missing values?
  - Missing values are inherent in practical applications.
  - An observation **x** may not have a value x_A.
  - *Solution 1:* fill in x_A as the most popular value of A in the training data.
  - *Solution 2:* fill in x_A as the most popular value of A in the training data which belong to the same class with **x**.

5. Random forests

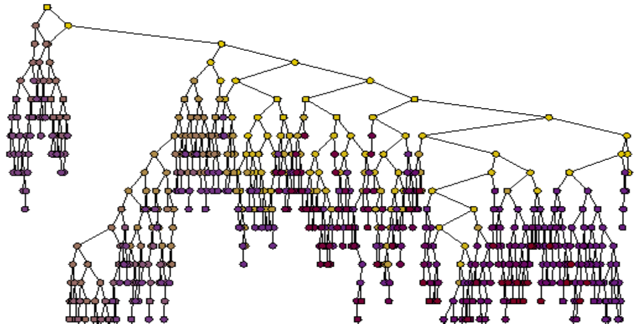
- Random forests (RF) is a method by Leo Breiman (2001) for both classification and regression.
- **Main idea:** prediction is based on combination of many decision trees, by *taking the average of all individual predictions*.
- Each tree in RF is simple but random.
- Each tree is grown differently, depending on the choices of the attributes and training data.

5. Random forests

- RF currently is one of the most popular and accurate methods [Fernández-Delgado et al., 2014]
  - It is also very general.
- RF can be implemented easily and efficiently.
- It can work with problems of very high dimensions, without overfitting ☺
- However, little is known about its theoretical properties ☹

5. RF: three basic ingredients

- **Randomization and no pruning:**
  - For each tree and at each node, we select randomly a subset of attributes.
  - Find the best split, and then grow appropriate subtrees.
  - Every tree will be grown to its largest size without pruning.
- **Combination:** each prediction later is made by taking the average of all predictions of individual trees.
- **Bagging:** the training set for each tree is generated by sampling (with replacement) from the original data.



5. RF: algorithm

- **Input**: training data D, number K of trees
- **Learning**: grow K trees as follows
  - Generate a training set D_i by sampling with replacement from D.
  - Learn the i^{th} tree from D_i:
    - At each node:
      - Select randomly a subset S of attributes.
      - Split the node into subtrees according to S.
    - Grow this tree upto its largest size without pruning.
- **Prediction**: taking the average of all predictions from the individual trees.

5. RF: practical performance

- RF is extensively compared with other methods
  - By Fernández-Delgado et al. (2014).
  - Using 55 different problems.
  - Using average accuracy (μ^P) as a measure.

No.	Classifier	μ^P	No.	Classifier	μ^P
1	rf_t	91.1	11	Bagging_LibSVM_w	89.9
2	parRF_t	91.1	12	RandomCommittee_w	89.9
3	svm_C	90.7	13	Bagging_RandomTree_w	89.8
4	RRF_t	90.6	14	MultiBoostAB_RandomTree_w	89.8
5	RRFglobal_t	90.6	15	MultiBoostAB_LibSVM_w	89.8
6	LibSVM_w	90.6	16	MultiBoostAB_PART_w	89.7
7	RotationForest_w	90.5	17	Bagging_PART_w	89.7
8	C5.0_t	90.5	18	AdaBoostM1_J48_w	89.5
9	rforest_R	90.3	19	Bagging_REPTree_w	89.5
10	treebag_t	90.2	20	MultiBoostAB_J48_w	89.4

References

- L. Breiman. *Random forests*. Machine learning, 45(1), 5-32, 2001.
- Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim. *Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?* Journal of Machine Learning Research, 15(Oct):3133−3181, 2014.
- T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- M. Nunez. *The use of background knowledge in decision tree induction*. Machine Learning, 6(3): 231-250, 1991.
- Quinlan, J. R. *Induction of Decision Trees*. Mach. Learn. 1, 1 (Mar. 1986), 81-106, 1986
- Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.
- Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, *34*(1), 1-47.