Decision Trees - Part I

Machine Learning, A.Y. 2022/23, Padova

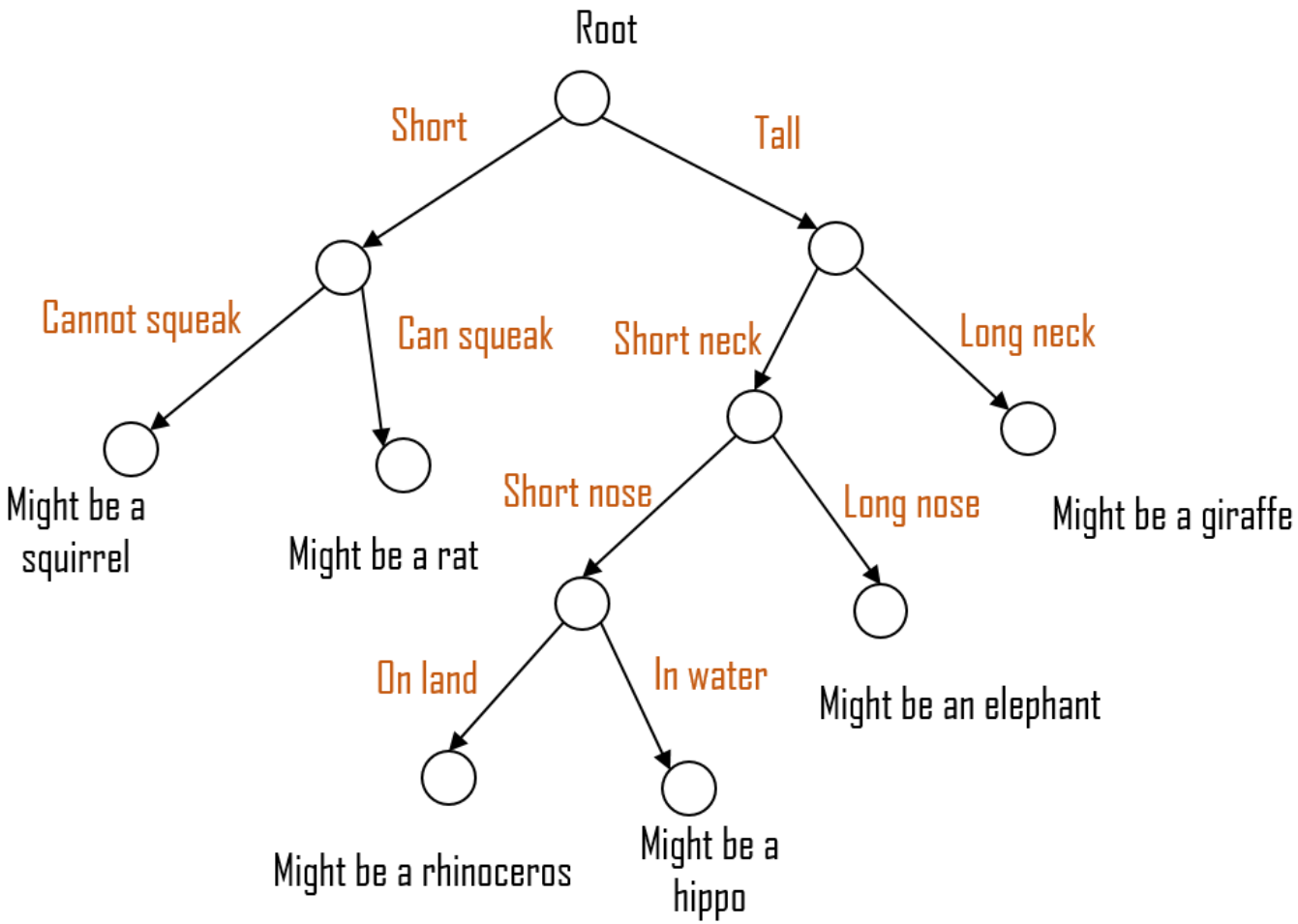
Fabio Aiolli

October 17th, 2022

Decision Trees

- Decision trees (DTs) allow to learn discrete decision functions/rules that are representable by a tree.
- Decision trees can be easily reduced to a series of `if-then` rules making this representation of the hypotheses comprehensible/interpretable by humans.
- This last feature (absent in most of the techniques that we will see later on) makes DTs particularly interesting for medical, biological, and financial applications, i.e., where interpreting the model learned by the algorithm is particularly important for an expert user.

Decision trees



Decision trees: definition

In a decision tree:

- An inner node specifies a test of some attribute of an instance;
- A branch descending from a node corresponds to one of the possible values the attribute can assume;
- A leaf node assigns a classification.

The classification of an instance works as in the following:

1. Start from the root;
2. Select the attribute attached to the current node;
3. Move down the tree branch corresponding to the value of that attribute in the given instance;
4. If a leaf node is reached, the label associated to the node is returned, else go to step 2 and repeat.

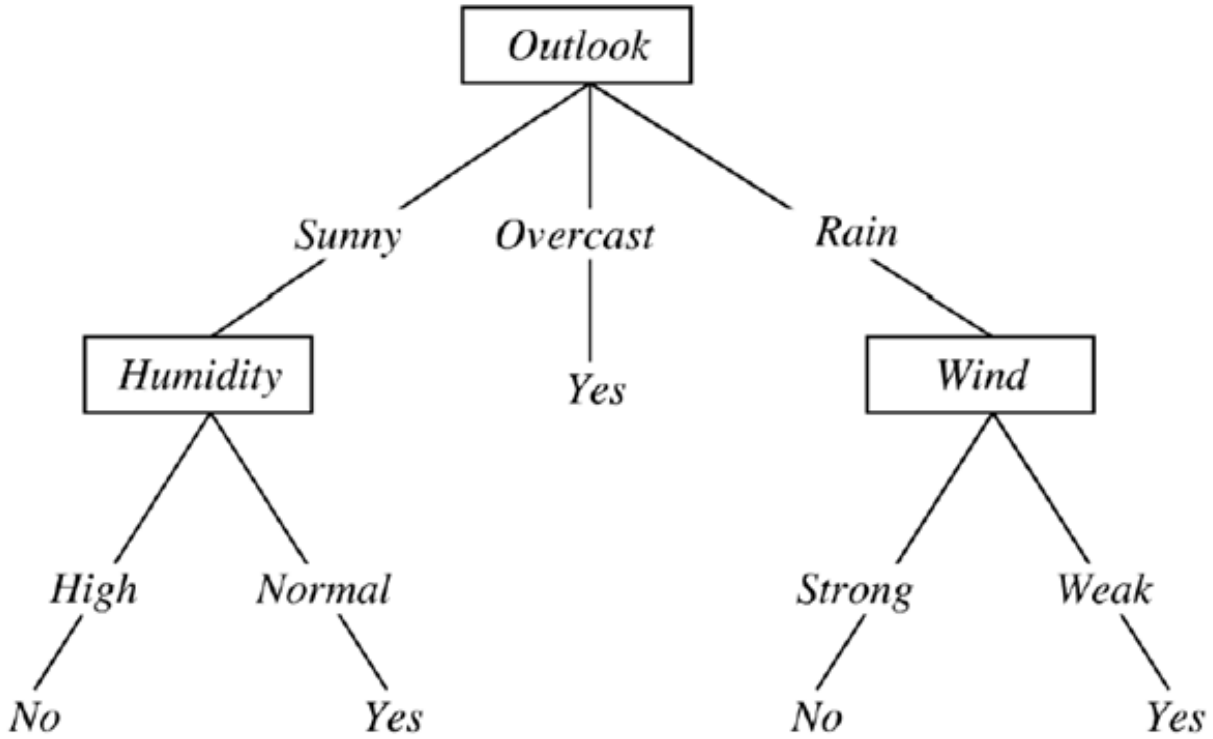
Is it a suitable day for playing tennis? (1)

Let's predict if a day is suitable to play tennis given its outlook, temperature, humidity, and wind.

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

Is it a suitable day to play tennis? (2)

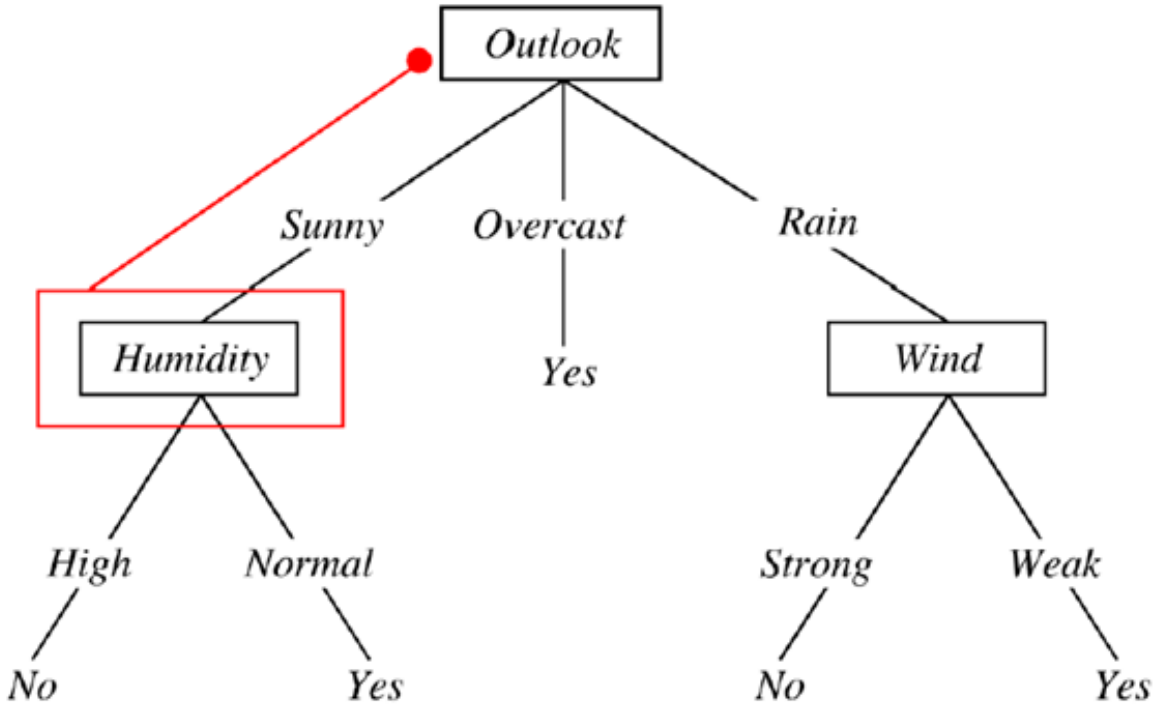
How can we decide if a day is a suitable day to play tennis?



Example: `[O=Sunny, T=Hot, H=High, W=Strong]` ?

Is it a suitable day to play tennis? (3)

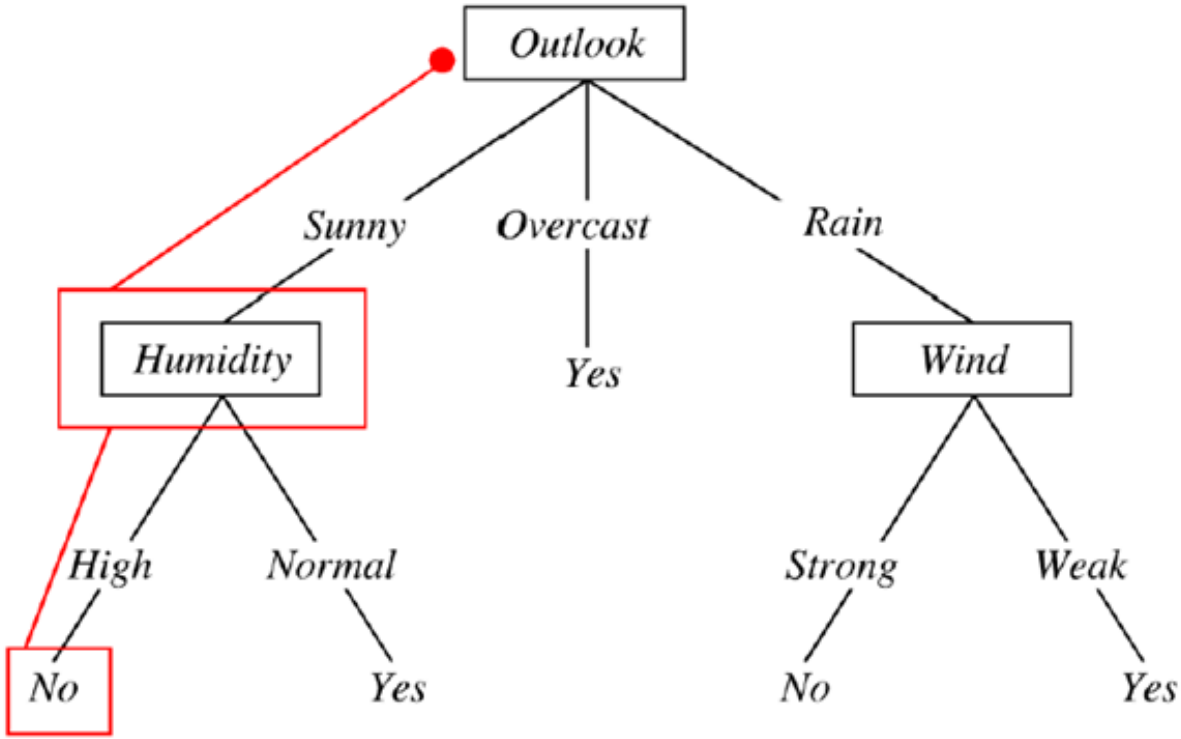
The attibute `Outlook (O)` is atteched to the root. Then, being `Outlook = Sunny` in the instance, the branch `Sunny` is followed:



Esempio: `[O=Sunny, T=Hot, H=High, W=Strong]` ?

Is it a suitable day to play tennis? (4)

The attibute `Humidity (H)` is attached to the reached node.Then, being `Humidity = High` in the instance, the branch `High` is followed and the classification `NO` is returned:

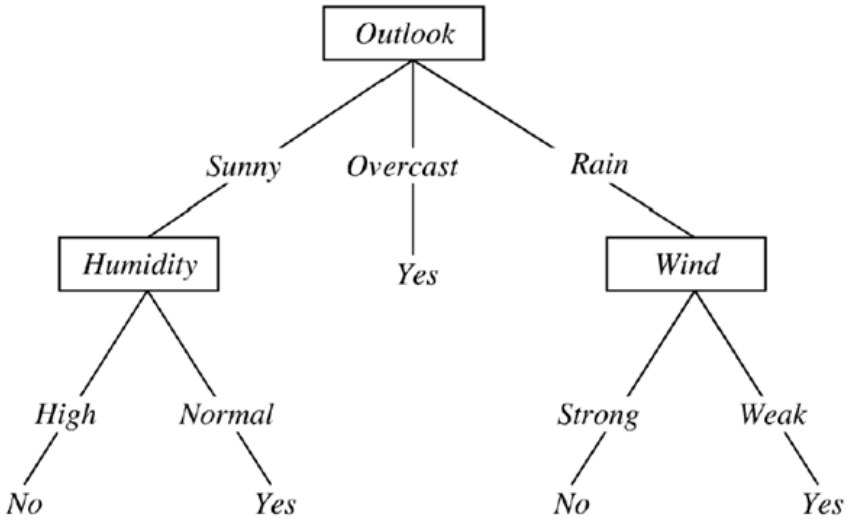


Esempio: `[O=Sunny, T=Hot, H=High, W=Strong]` ? NO

Decision trees and Boolean functions

A decision tree can be represented as a Boolean function:

- Each path from the root to a leaf codifies a conjunction of constraints on the attribute values of instances;
- Different paths that lead to a same classification codify disjunctions of conjunctions;
- The above mentioned rules define a series of DNF (disjunctive normal form) formulas, one DNF for each class.



DNF corresponding to YES

(O=Sunny **and** H=Normal)

or

(O=Overcast)

or

(O=Rain **and** W=Week)

Decision trees: when to use them

Decision trees are particularly useful when dealing with problems having the following characteristics:

- Instances represented as attribute-value pairs;
  - A fixed set of attributes and values;
  - Few possible values for the attributes;
  - Discrete values for the attributes (extensions to real-valued)
- Target functions with discrete output values (two or more values);
- Target concepts can be described by disjunctions of boolean functions;
- Training examples can contain noise and missing values.

Learning algorithms for decision trees are generally very efficient. For this reason, decision trees are (still) popular in practical applications.

ID3: DTs learning algorithm (Quinlan 1986)

The most popular learning algorithm for DTs (a.k.a. ID3) is based on a top-down, greedy search procedure through the space of possible DTs. A recursive implementation is the following:

ID3(S: Sample, A: Attributes)

- Create a root node T
- If examples in S are all of the same class c, return the leaf node T with label c;
- If A is empty, return the leaf node T with label the majority class in S;
- Select $a \in A$, the *optimal* attribute in A (details later);
- Partition S according to the values that a can take: $S_{a=v_1}, \ldots, S_{a=v_m}$ where m is the number of distinct values of the attribute a;
- Return the tree T having sub-trees the trees obtained by recursively calling **ID3**($S_{a=v_j}$,$A - a$), for each j.

ID3: optimal attribute choice (1)

Different learning algorithms for DTs mainly differ in how they select the optimal attributes for the nodes: ID3 uses the concepts of Entropy and Information Gain.

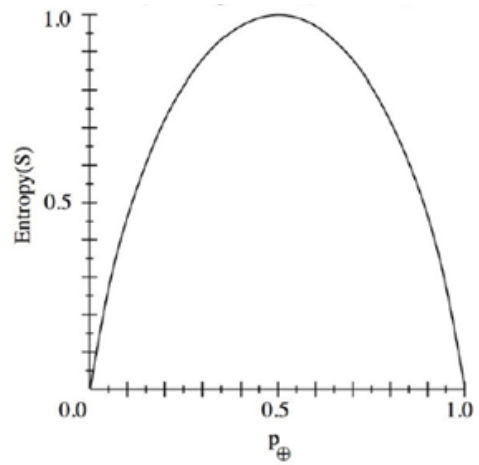
The entropy is a measure of the "degree of impurity" of a sample S. Let C be the number of classes and S_c the subset of S of samples of class c, then the entropy is computed with the formula:

$$E(S) = -\sum_{c=1}^{m} p_c \log(p_c), \text{ where } p_c = \frac{|S_c|}{|S|}$$

that, in the case of binary classification, it becomes:

$$E(S) = -p_- \log(p_-) - p_+ \log(p_+)$$

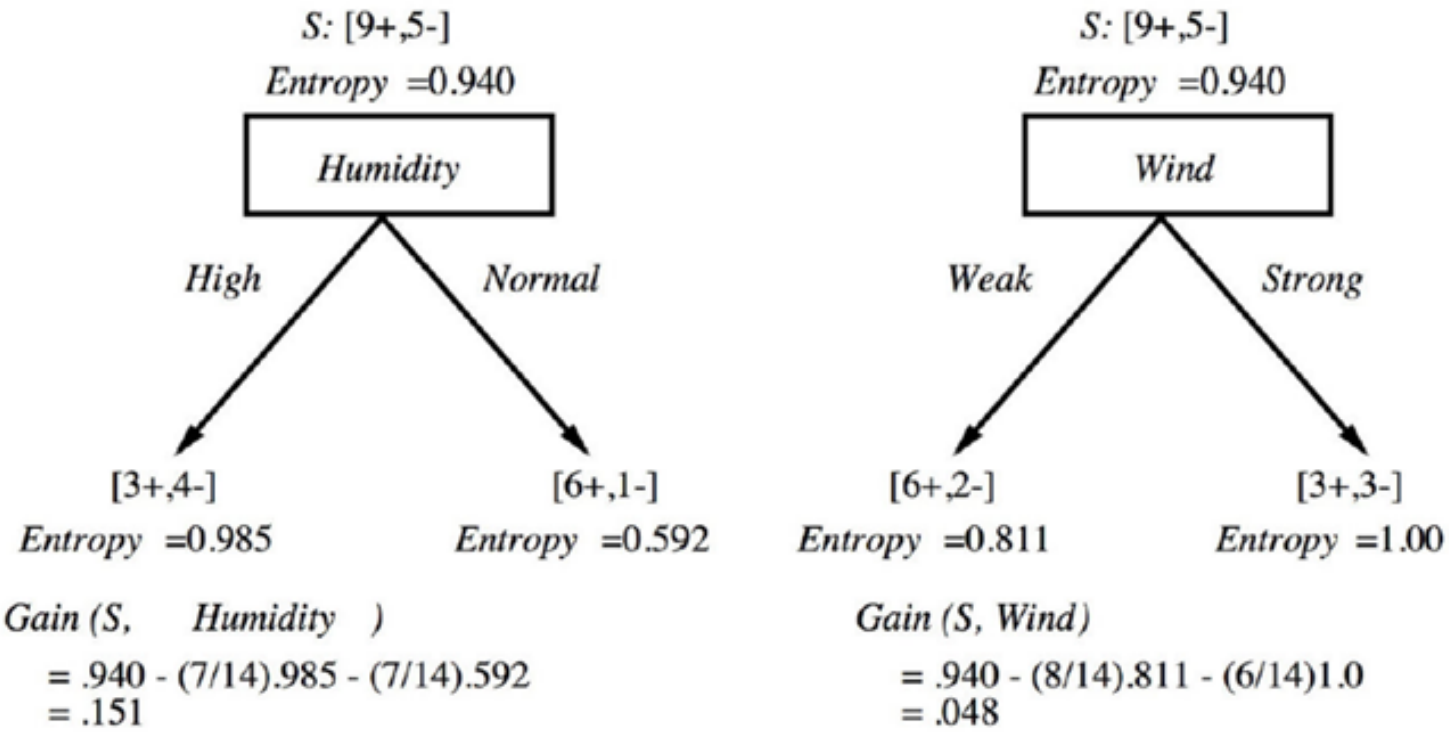
where p_-, p_+ are the proportion of negative (resp. positive) examples in S.



ID3: optimal attribute choice (2)

The selected attribute will be the one that maximizes the Information Gain $G(S, a)$, i.e. the *expected* reduction of the entropy obtained when partitioning the examples in S according to the value of attribute a:

$$G(S, a) = E(S) - \sum_{v \in V(a)} \frac{|S_{a=v}|}{|S|} E(S_{a=v})$$



Alternative criteria for the optimal attribute choice

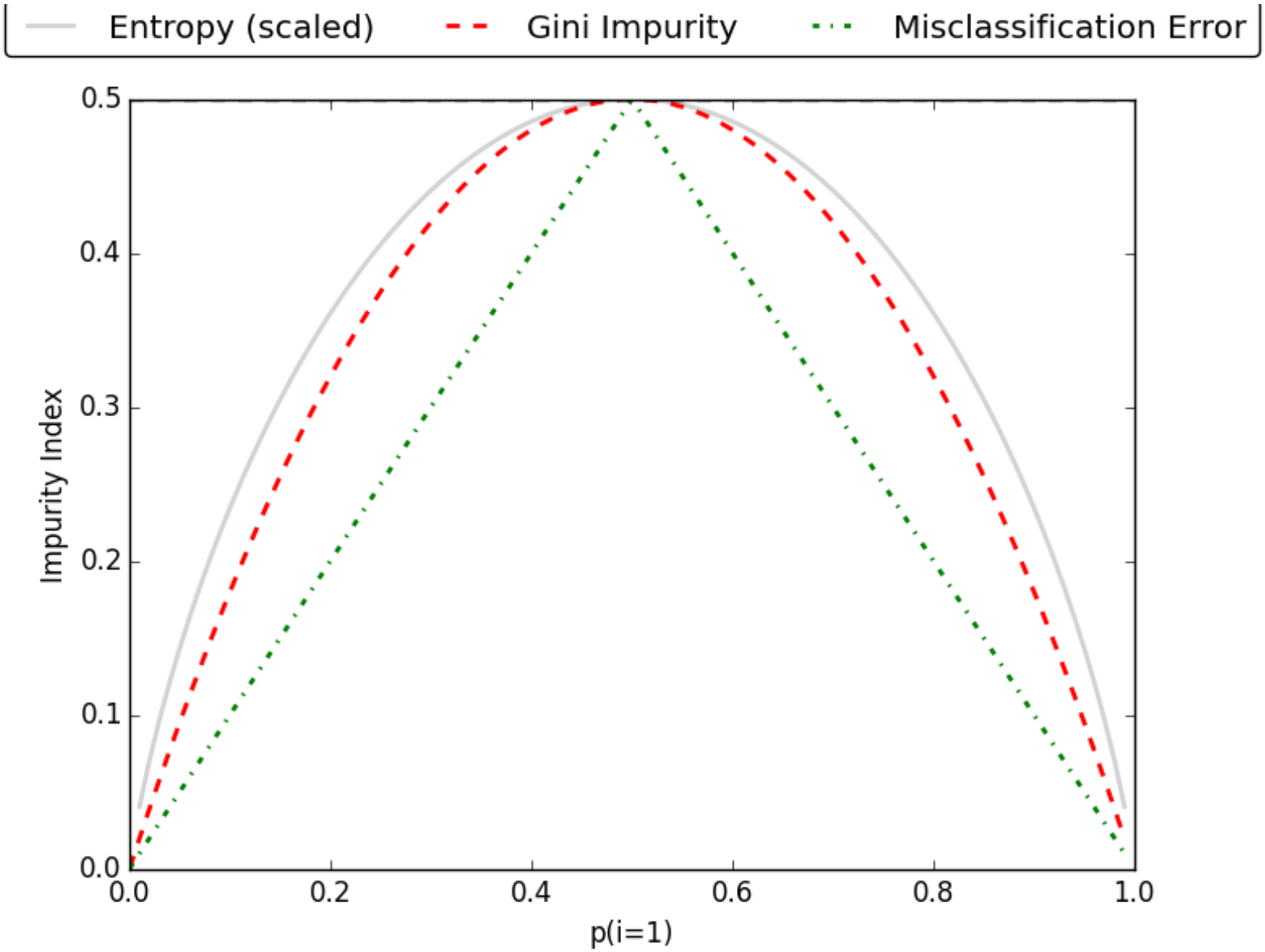
We can generalize the notion of Information Gain to other impurity measures:

- Cross-Entropy: $I_H = -\sum_{c=1}^{m} p_c \log(p_c)$
- Gini Index: $I_G = 1 - \sum_{c=1}^{m} p_c^2$
- Misclassification: $I_E = 1 - \max_c(p_c)$

Specifically, let I be any of the impurity criteria above, the information gain can be defined as:

$$G(S, a) = I(S) - \sum_{v \in V(a)} \frac{|S_{a=v}|}{|S|} I(S_{a=v})$$

Alternative criteria for the optimal attribute choice



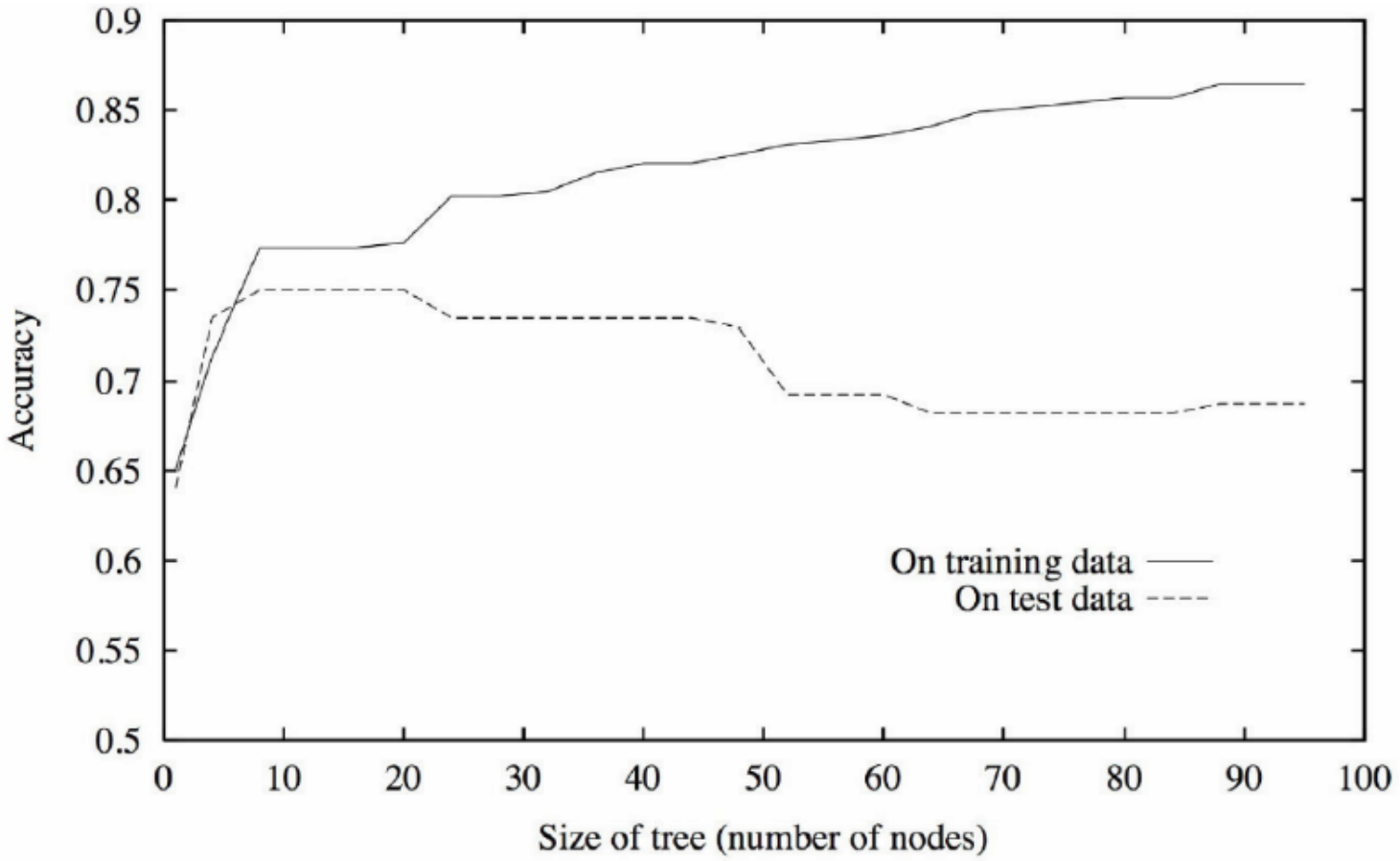
Information gain, contraindication

Problem. The information gain tend to favor too much those attributes that can assume many possible values.

Example. If an attribute consisting of the date of the considered day is added to the problem of deciding when to play tennis (e.g. Date = "November 11"), then the Date attribute is likely going to be the one with maximum gain (in fact, each subset will constitute a different and probably pure subset, therefore with zero impurity), even if that attribute is clearly not significant.

Learning of DTs: overfitting

Problem: Overfitting



Partial solution (sklearn): setting a minimal number of examples to accept in leaf nodes or limiting the maximal deepth of the tree.

Recap

Notions

- Decision Trees: when to use them?
- Hypothesis space of DTs (DNF)
- ID3 Algorithm
- Selection of the optimal attribute (Info Gain, impurity measures)

Exercizes

- Using the algorithm ID3, compute (manually) the decision tree corresponding to the task of realizying simple Boolean formulas (AND, OR, XOR, ...) in n variables. Hint: Consider the truth table of the formula as your training sample.