3. Decision Trees

Problem: Given a dataset D for a target function c, compute consistent hypothesis wrt D

We can see concept learning as a search in the hypothesis space. The solution approach is the following:

- Define hypothesis space H
- Implement an algorithm to search h and H that are consistent with D.

DECISION TREES: the hypothesis space is a SET of DECISION TREES, so every hypothesis is a decision tree.

- We have internal nodes that are interpreted as test of attributes
- edges are values of attributes
- leaves are the classification categories

We can see a decision tree also as a set of rules and any path to leaf node can be described as conjunction of constraints.

The problem is the following:

We have a dataset and we build the tree consistent with the dataset. Given a tree is easy to classify an instance, we have only to travel the path to the leaf.

A TREE IS CONSISTENT WITH THE DATASET IF CLASSIFIES WITH SAME LABELS.

ID3 Algorithm:

INPUT: Examples, Target-attributes, Attributes

- Example: training examples
- Target-attribute: the attribute whose value has to be predicted by the tree
- Attributes: is a list of other attributes that may be tested by the learned decision tree.

OUTPUT: decision tree.

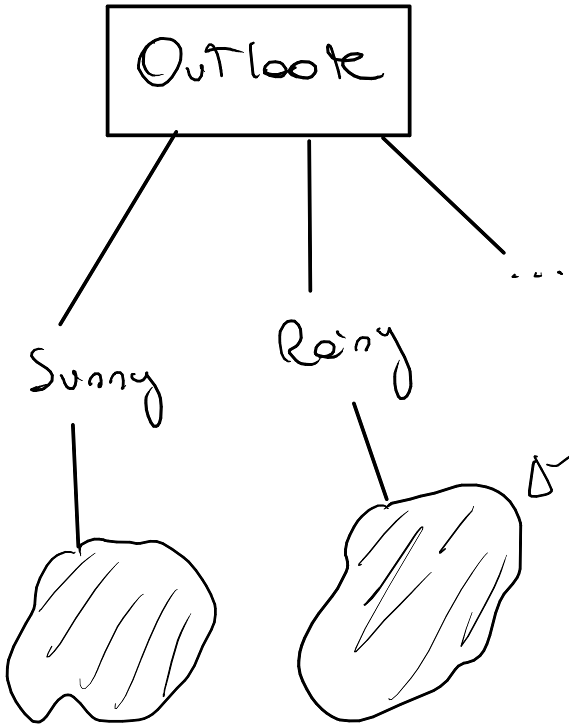
The algorithm compute a decision tree in a recursive way.

- Create a ROOT
- if all Examples are positive, return ROOT with label +
- if all Examples are −, return ROOT with label −
- if Attributes ≠ 0, return the ROOT with label = most common value of Target-Attribute in Examples.
- OTHERWISE: (CREATING INTERMEDIATE NODE)
  - A ← "best" decision attribute for Examples
  - Assign A as decision attribute for ROOT
  - For each $V_i \in A$
    - → add a branch from ROOT corresponding to $A = V_i$
    - → Examples_{V_i} = Subset of Examples that have V_i as value for A
    - → If $\text{Examples}_{V_i} = \emptyset$

      ADD a leaf node with label = most common value of Target-attribute in Example
    - → Else

      ADD THE TREE ID3 (Examples, Target-Attribute, Attributes − {A})
      (Recursive step)

We generate as many edges as the values of the attribute

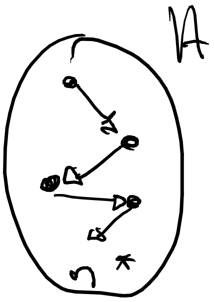


All data that has "outlook" = "Rainy"

In the case the data of the particular value is empty you generate a leaf node.

We will terminate because the set of attributes eventually will be Ø.

We need to understand which is the best decision attribute:



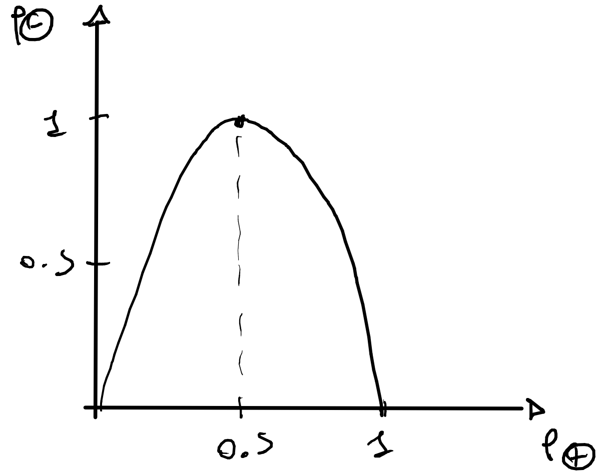
The way of moving within the hypotheses space depends on how we decide for the best attribute.

INFORMATION GAIN : the information gain measures how well a given attribute separates the training examples according to their target classification.

ID3 selects the attribute that has highest information gain and it is measured as a reduction in entropy.

ENTROPY: measures of mess (measures of impurity in S)

$p_\oplus$ = % + instances , $p_\ominus$ = % − instances



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

What is important is to reduce the entropy, in the base cases of the algorithm the entropy is 0

In case of multiple classes (multivalued target functions - c-wise classification)

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

The gain (A,S) is the expected reduction of entropy.

Given an attribute you generate a partition

$$S_v = \{ \underbrace{s}_{\text{subset of } S} \in S \mid A(s) = v \},$$

you make a weighted avg of entropies of the subsets:

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

The DT can change for many reasons (it can require additional nodes), for instance when you increase the dataset. When the tree is growing to accomodate new data in the dataset you increase the accuracy wrt the training set

(THE PROBLEM COMES FOR UNSEEN DATA).

OVERFITTING $\boxed{\uparrow \text{DT deeper} \Rightarrow \uparrow \text{accuracy on training data}}$

To avoid overfitting we can make some change to the algorithm C: we may produce the tree as big as possible and then cut off some parts, GROW FULL TREE and then POST-PRUNE.

PRUNING: pruning a (decision tree) decision node consists on REMOVING the subtree ROOTED at that node, making it a leaf node and assigning to it the most classification of the training examples affiliated with that node.

To determine the correct tree size:

- Use a separate set of examples (distinct from the training examples) to evaluate the utility of post pruning (K-FOLD - CROSS VALIDATION)

- apply a statistical test to estimate accuracy of a tree on the entire data distribution.

Reduced - Error Pruning

Split data into training and validation set.

Do until further pruning is harmful (reduce accuracy):

- Evaluate impact on validation set of pruning each possible node.

- Greedily remove the one that most improves validation set accuracy

When the dataset is limited, reducing the set of training examples can give bad results. The TESTS HAVE DIFFERENT COST IN TIME of how they affect on use case.

There are methods that use DT to do something more complex:

RANDOM FORES.

RANDOM FOREST: set of DT generated with randomness and integrate their values into a final result.

↓

Less sensitive to overfitting

Note on DT: the structure of DT is self explicable, that's not the case for other methods.

DT one of the baseline to compare results.