Decision Tree

Outline

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting

Representation of Concepts

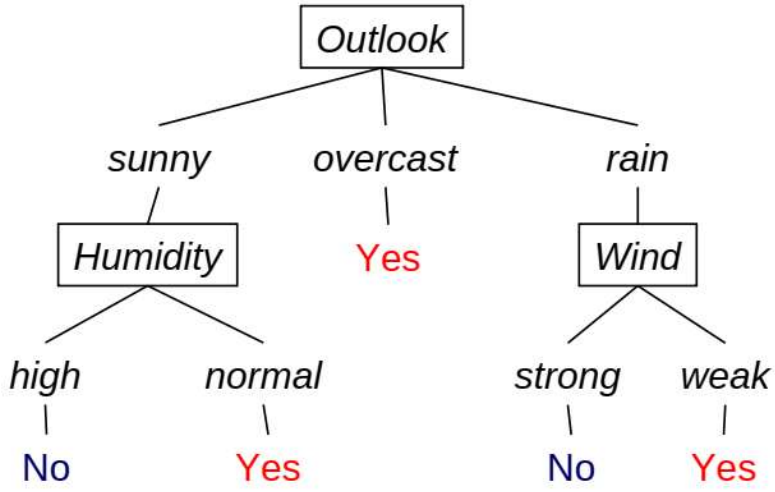
- Concept learning: conjunction of attributes
  - (Sunny AND Hot AND Humid AND Windy) +
- Decision trees: disjunction of conjunction of attribues
  - (Sunny AND Normal) OR (Overcast) OR (Rain AND Weak) +
- More powerful representation
- Larger hypothesis space H
- Can be represented as a tree
- Common form of decision making in humans

Decision Trees

- Decision tree is a classifier with a tree representation:
  - Each internal node is a decision node which specifies a test of some attribute
  - Each branch corresponds to attribute value
  - Each leaf node assigns a classification
- Can be represented by logical formulas
- Example of representing rule in DT's:

  if outlook = sunny AND humidity = normal
  OR
  if outlook = overcast
  OR
  if outlook = rain AND wind = weak
  then playtennis

Decision tree for Play Tennis



Decision Trees and Rules

- Rules can represent a decision tree:

  if item1 then subtree1

  elseif item2 then subtree2

  elseif...

- There are as many rules as there are leaf nodes in the decision tree.

- Advantage of rules over decision trees:
  - Rules are a widely-used and well-understood representation formalism for knowledge in expert systems;
  - Rules are easier to understand, modify and combine; and
  - Rules can significantly improve classification performance by eliminating unnecessary tests.
  - Rules make it possible to combine different decision trees for the same task.

Advantages of Decision Trees

- **Decision trees are simple to understand.**

  People are able to understand decision trees model after a brief explanation.

- **Decision trees have a clear evaluation strategy**

  Decision trees are easily interpreted

- **Decision trees are able to handle both nominal and categorical data**.

  Other techniques are usually specialised in analysing datasets that have only one type of variable. Ex: neural networks can be used only with numerical variables

Advantages of Decision Trees (1)

- **Decision trees are a white box model.**

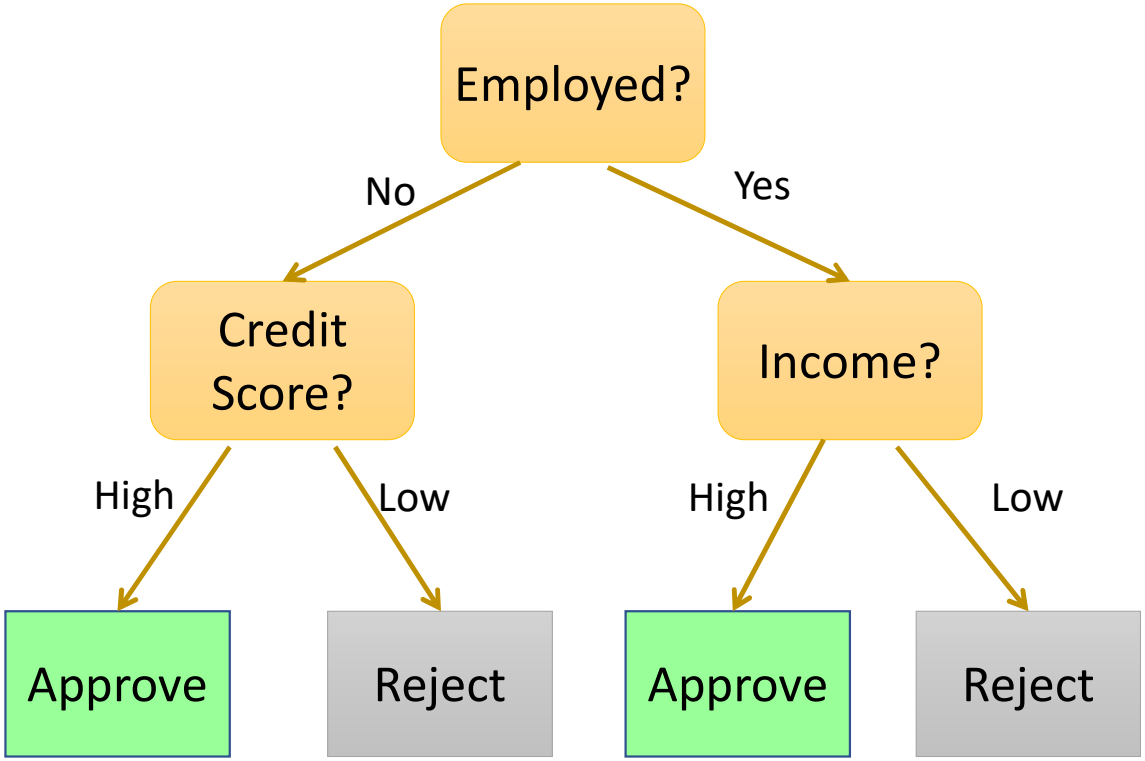
  If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. An example of a black box model is an artificial neural network since the explanation for the results is excessively complex to be comprehended.

- **Decision trees are robust, perform well with large data in a short time.**

  Large amounts of data can be analysed using personal computers in a time short enough to enable stakeholders to take decisions based on its analysis.

Decision Tree Another Example

Whether to approve a loan



Applications of Decision Trees

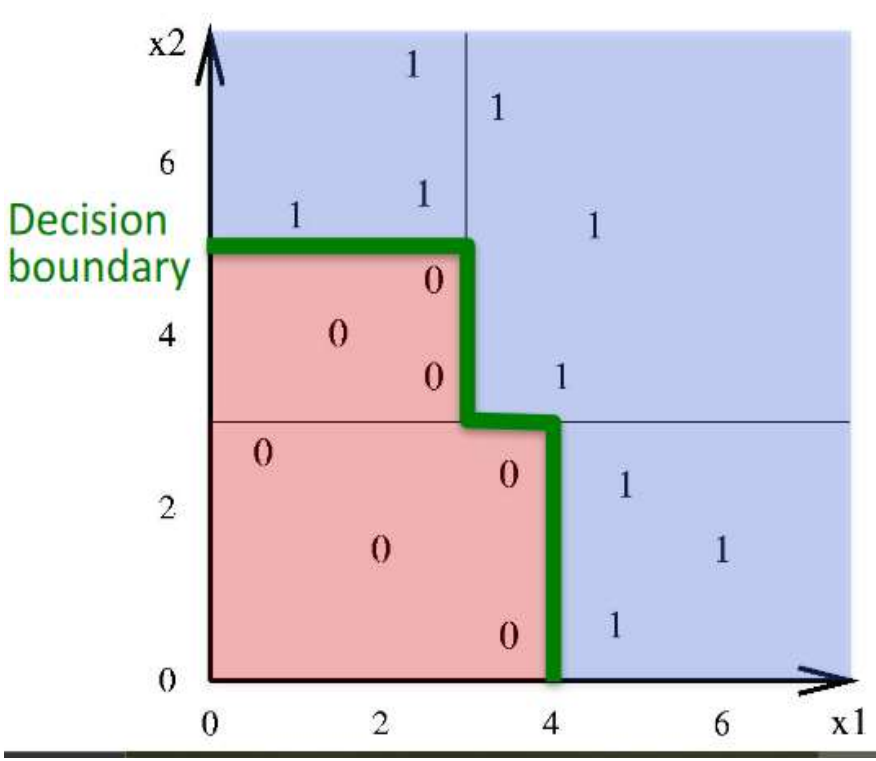
- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

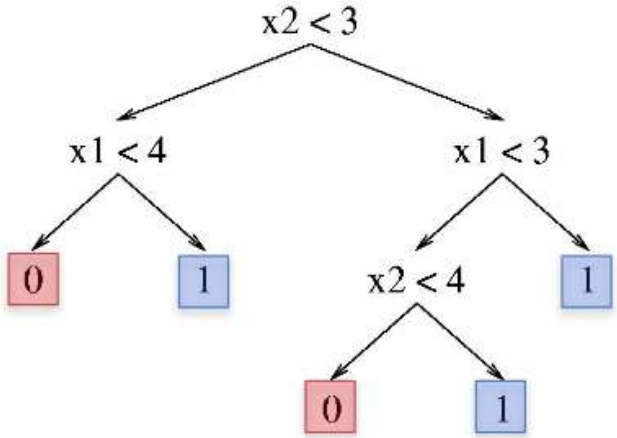
Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preference

Decision Tree –Decision Boundary

•Decision trees divide the feature space into axis-parallel rectangles
•Each rectangular region is labeled with one label–or a probability distribution over labels



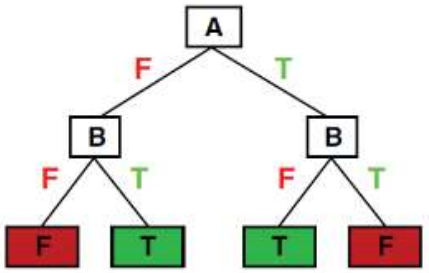


Expressiveness

Decision trees can represent any function of the input attributes

- –Boolean operations (and, or, xor, etc.)
- –All Boolean functions

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Issues

- Given some training examples, what decision tree should be generated?
- One proposal: prefer the smallest tree that is consistent with the data (Bias)
- Possible method:
  - search the space of decision trees for the smallest decision tree that fits the data

Searching for a good tree

- The space of decision trees is too big for systematic search.
- Stop and
  - return the a value for the target feature or
  - a distribution over target feature values
- Choose a test (e.g. an input feature) to split on.
  - For each value of the test, build a subtree for those examples with this value for the test.

Top-Down Induction of Decision Trees ID3

1. Which node to proceed with?

1. A ← the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

2. When to stop?

The Basic ID3 Algorithm

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree
        ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {A}))
- End
- Return *Root*

Choices

- When to stop
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split

- Which test to split on
  - split gives smallest error.
  - With multi-valued features
    - split on all values or
    - split values into half.

Which Attribute is ”best”? Principled Criterion

- Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.
- information gain
  - measures how well a given attribute separates the training examples according to their target classification
  - This measure is used to select among the candidate attributes at each step while growing the tree
  - Gain is measure of how much we can reduce uncertainty (Value lies between 0,1)

Entropy

- A measure for
  - uncertainty
  - purity
  - information content

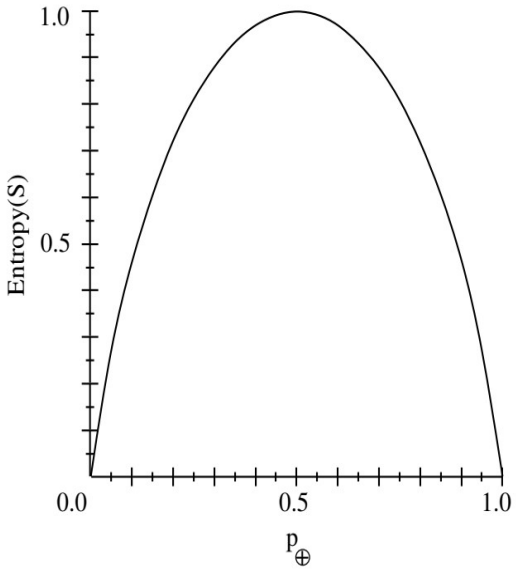
$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

Info Gain = Entropy(Parent)- (Weighted Avg) Entropy(Children)

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

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

- The entropy is 0 if the outcome is ``certain".
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).



Entropy

- Entropy characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection S (containing positive and negative examples), the entropy of S relative to this boolean classification is

- $\mathbf{p}_+/\mathbf{p}_-$ is the proportion of positive/negative examples in S.
- Entropy is 0 if all members of S belong to the same class; entropy is 1 when the collection contains an equal number of positive and negative examples.
- More general:

Information Gain

- Information gain measures the expected reduction in entropy caused by partitioning the examples according to an attribute:

- First term: entropy of the original collection S; second term: expected value of entropy after S is partitioned using attribute A (S_v subset of S).

- Gain(S,A): The expected reduction in entropy caused by knowing the value of attribute A.

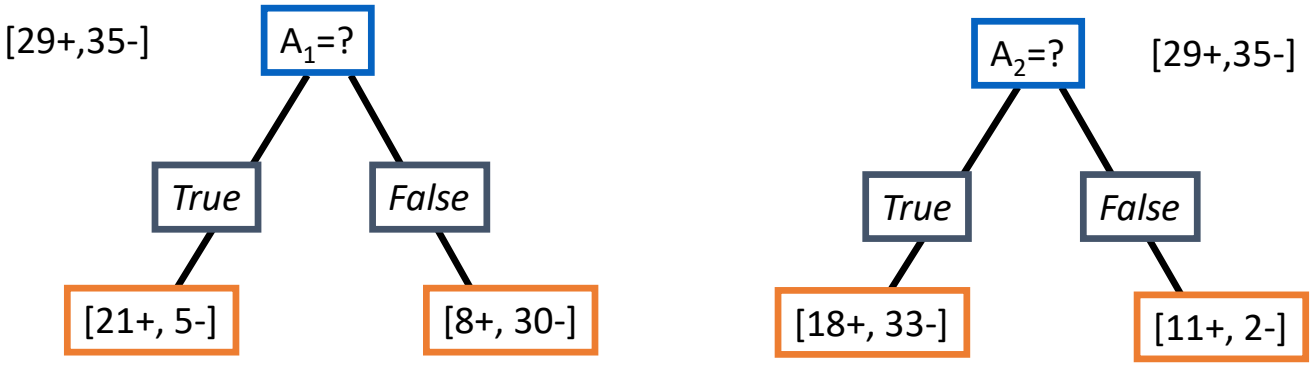
- ID3 uses information gain to select the best attribute at each step in growing the tree.

Information Gain

Gain(S,A): expected reduction in entropy due to partitioning S on attribute A

$$\text{Gain}(S,A)=\text{Entropy}(S) - \sum_{v \in \text{values}(A)} |S_v|/|S| \ \text{Entropy}(S_v)$$

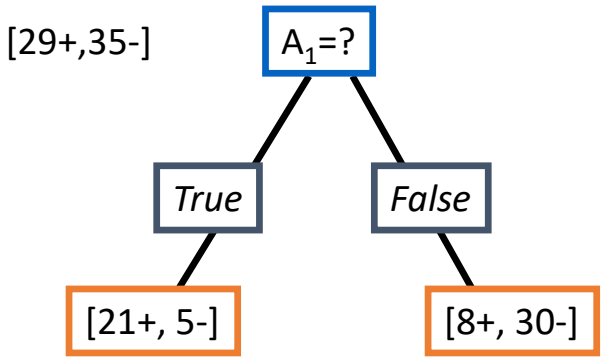
Entropy([29+,35-]) = $-29/64 \log_2 29/64 - 35/64 \log_2 35/64$
= 0.99

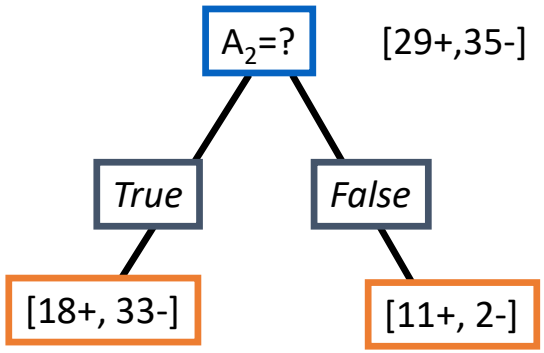


Information Gain

Entropy([21+,5-]) = 0.71
Entropy([8+,30-]) = 0.74
Gain(S,A_1)=Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
=0.27

Entropy([18+,33-]) = 0.94
Entropy([8+,30-]) = 0.62
Gain(S,A_2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12

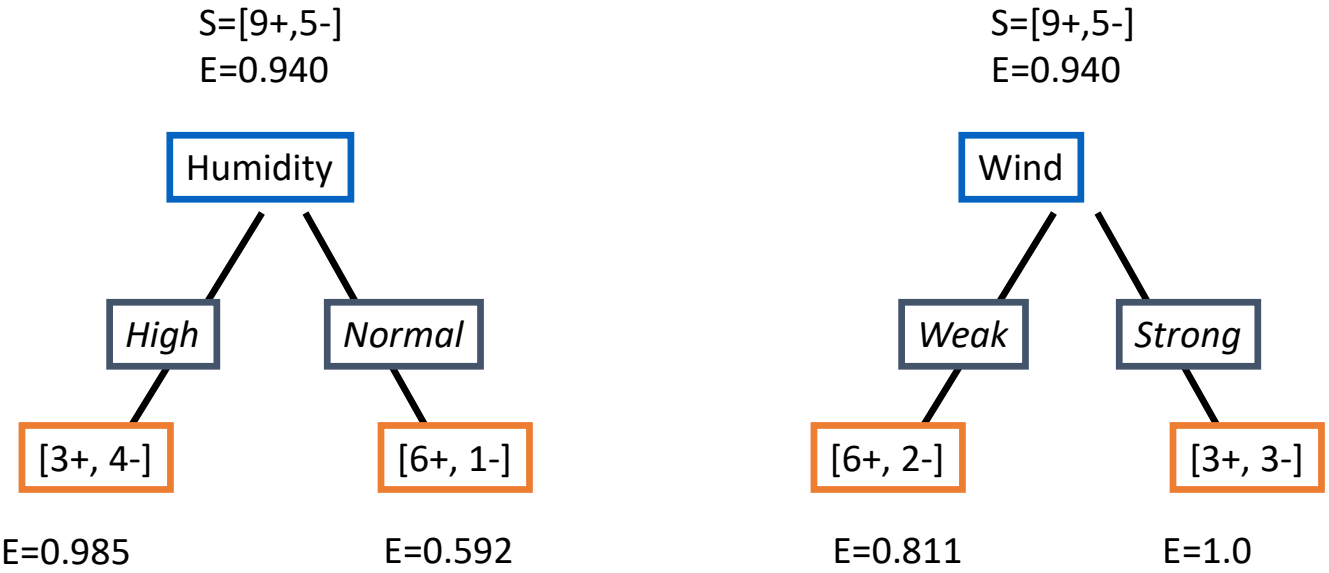




Training Examples

Day	Outlook	Temp	Humidity	Wind	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*

Selecting the Next Attribute



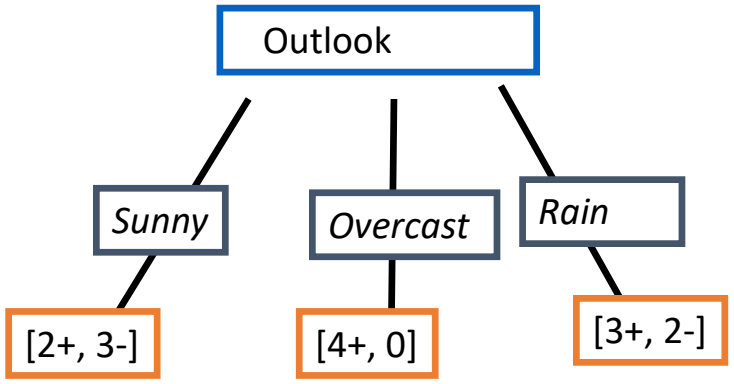
Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t target classification.

Selecting the Next Attribute

S=[9+,5-]
E=0.940



Gain(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

Selecting the Next Attribute

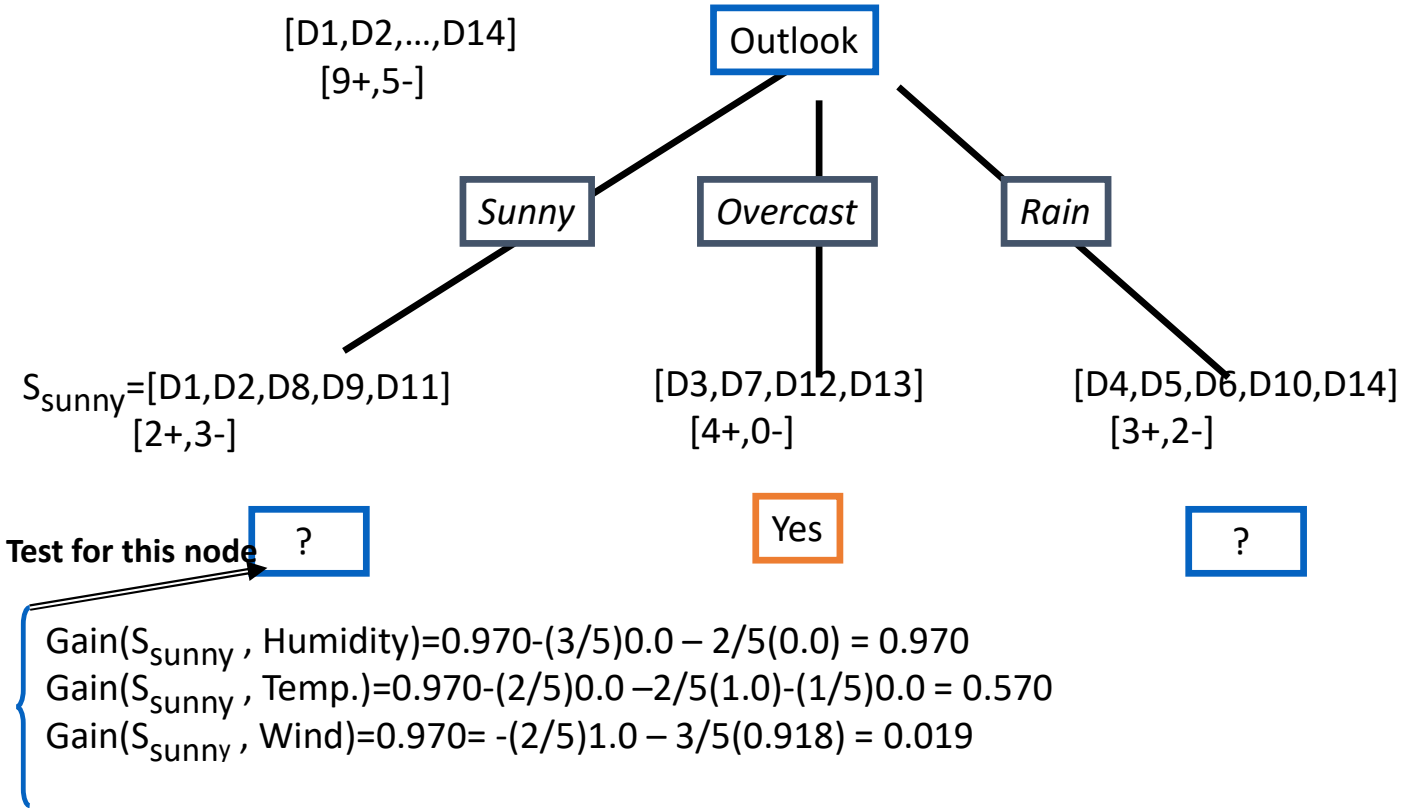
The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

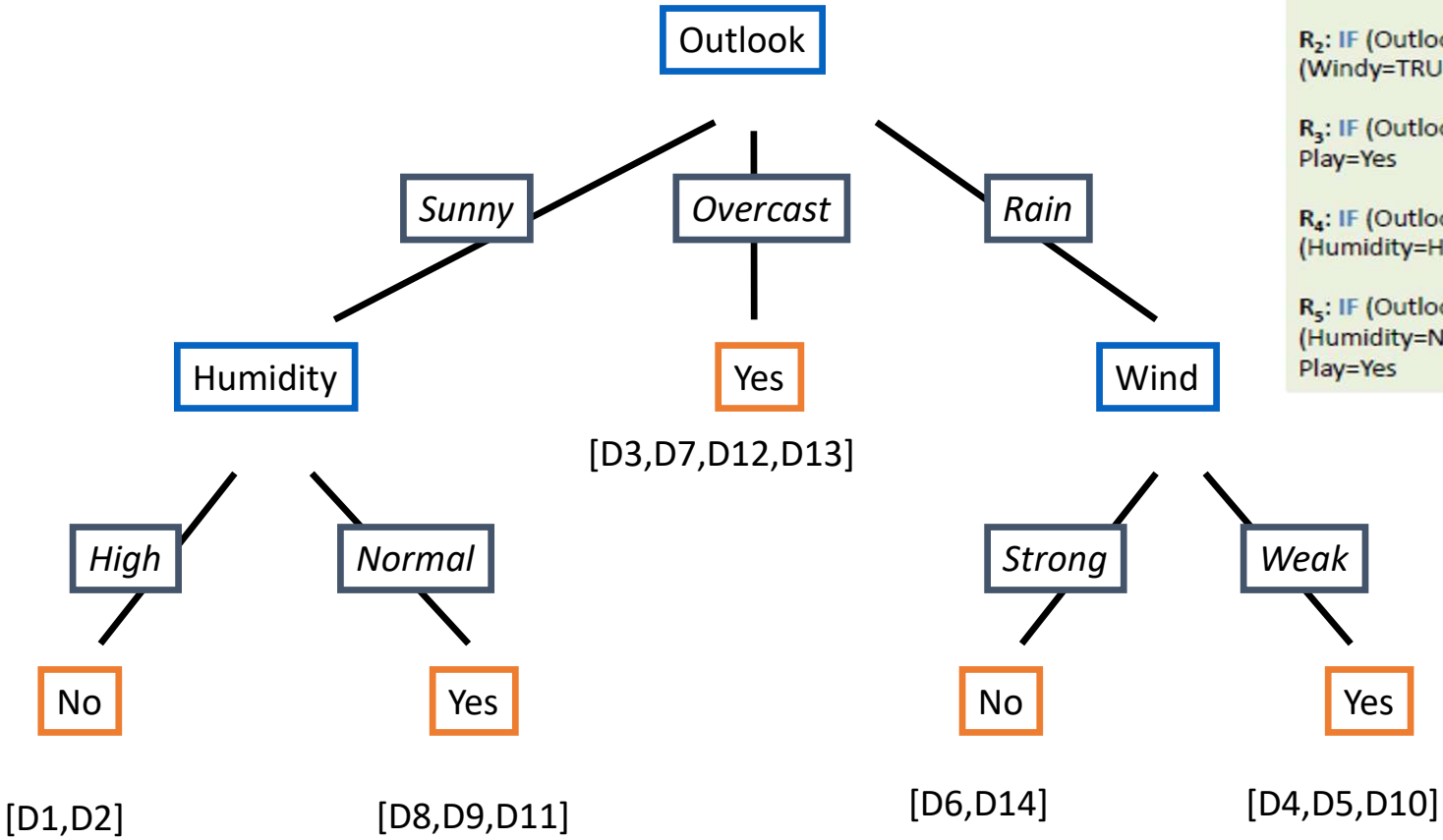
where S denotes the collection of training examples

Note: $0\text{Log}_2 0 = 0$

ID3 Algorithm



ID3 Algorithm



R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes

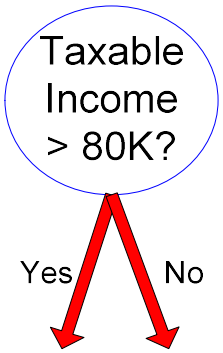
Splitting Rule: GINI Index

- GINI Index
  - Measure of node impurity

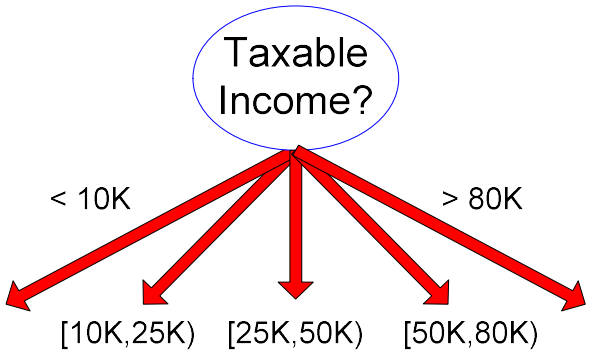
$$GINI_{node}(Node) = 1 - \sum_{c \in classes} [p(c)]^2$$

$$GINI_{split}(A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GINI(N_v)$$

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

Continuous Attribute – Binary Split

- For continuous attribute
  - Partition the continuous value of attribute A into a discrete set of intervals. Temperature = 82:5
  - Create a new boolean attribute A_c , looking for a threshold c,

$$A_c = \begin{cases} true & \text{if } A_c < c \\ false & \text{otherwise} \end{cases}$$

- consider How to choose c ? finds the best cut

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

Hypothesis Space Search in Decision Trees

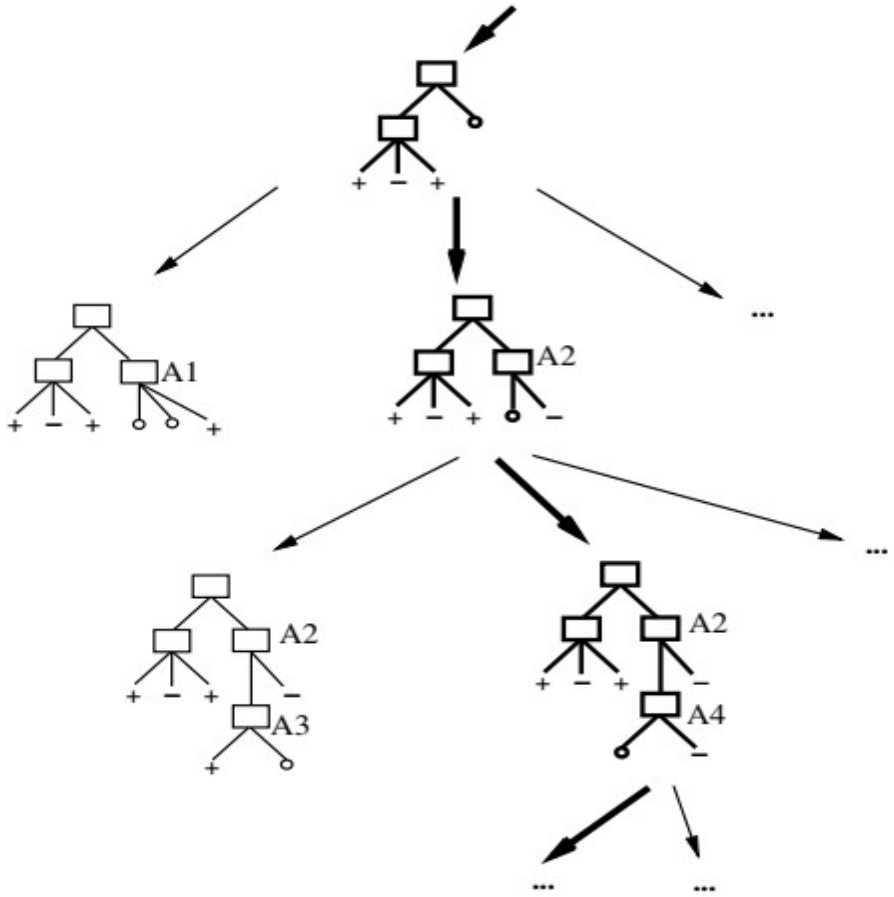
The ID3 algorithm

- searches hypotheses space which is set of possible hypotheses
- Performs simple-to-complex, Hill-Climbing search
- Starts with empty tree
- Depends on Information Gain which guides hill-climbing search

Hypothesis Space Search in Decision Trees

ID3	candidate-Elimination
ID3 maintains only a single current hypothesis	candidate-Elimination method, which maintains the set of all hypotheses consistent with the available training examples
It does not know how many alternative decision trees are consistent with the available training data	
performs no backtracking in its search	
ID3 uses all training examples at each step in the search to make statistically based decisions	FIND-S or CANDIDATE-ELIMINANATION make decisions incrementally, based on individual training examples
Goal: to find the best decision tree	

Hypothesis Space Search in Decision Trees



Hypothesis Space Search in Decision Trees

- ID3 searches a *complete* hypothesis space but does so *incompletely* since once it finds a good hypothesis it stops (cannot find others).
- Candidate-Elimination searches an *incomplete* hypothesis space (it can only represent some hypothesis) but does so *completely*.
- A ***preference bias*** is an inductive bias where some hypothesis are preferred over others.
- A ***restriction bias*** is an inductive bias where the set of hypothesis considered is restricted to a smaller set.

INDUCTIVE BIAS IN DECISION TREE LEARNING

- Inductive bias is the set of assumptions that, together with the training data
- Which of these decision trees does ID3 choose?
- It chooses the first acceptable tree it encounters in its simple-to-complex, hill climbing search through the space of possible trees
- the ID3 search strategy
  - selects in favor of shorter trees over longer ones
  - selects trees that place the attributes with highest information gain closest to the root

Approximate inductive bias of ID3: Shorter trees are preferred over larger trees.

- Consider breadth-first search algorithm BFS-ID3 which finds a shortest decision tree and thus exhibits precisely the bias "shorter trees are preferred over longer trees."
- BFS-ID3 conducts the entire breadth-first search through the hypothesis space
- ID3 can be viewed as an efficient approximation to BFS-ID3
- Because ID3 uses the information gain heuristic and a hill climbing strategy, it exhibits a more complex bias than BFS-ID3
- it does not always find the shortest consistent tree
- it is biased to favor trees that place attributes with high information gain closest to the root.

A closer approximation to the inductive bias of ID3: Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

Restriction Biases and Preference Biases

types of inductive bias

- preference bias or search bias
- restriction bias or language bias

- The inductive bias of ID3 is thus a preference for certain hypotheses over others (e.g., for shorter hypotheses), with no hard restriction on the hypotheses that can be eventually enumerated. This form of bias is typically called a preference bias (or, alternatively, a search bias).
- the bias of the CANDIDATE ELIMINATION algorithm is in the form of a categorical restriction on the set of hypotheses considered. This form of bias is typically called a restriction bias (or, alternatively, a language bias).

a preference bias is more desirable than a restriction bias

Bias and Occam's Razor

Bias: Searches a complete hypothesis space incompletely

Inductive bias is solely a consequence of the ordering of hypotheses by its search strategy

Inductive bias often goes by the name of Occam's razor

Occam's Razor: Prefer shorter hypotheses that fits the data

Argument in favor:

- Fewer short hypotheses than long hypotheses
- A short hypothesis that fits the data is unlikely to be a coincidence
- A long hypothesis that fits the data might be a coincidence

Issues in Decision Tree Learning

- Determine how deeply to grow the decision tree, underfitting and overfitting
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with differing costs
- Improving computational efficiency

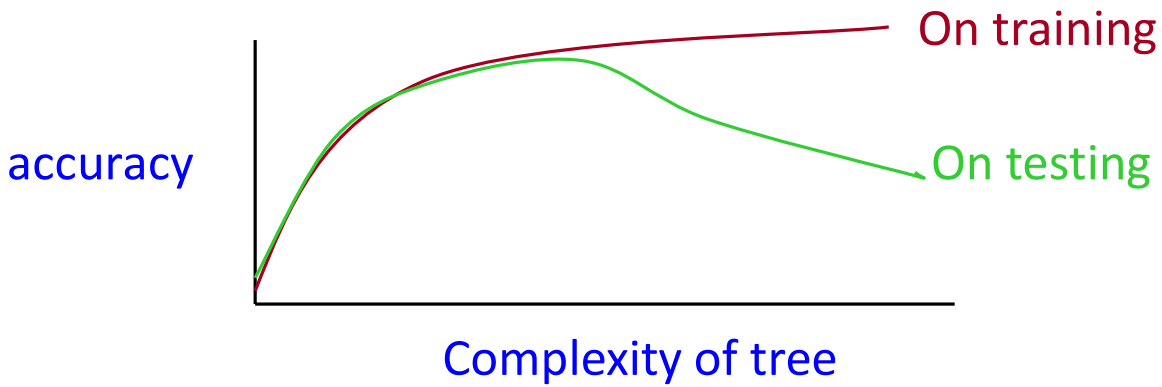
Overfitting

An algorithm can produce trees that overfit the training examples in the following two cases

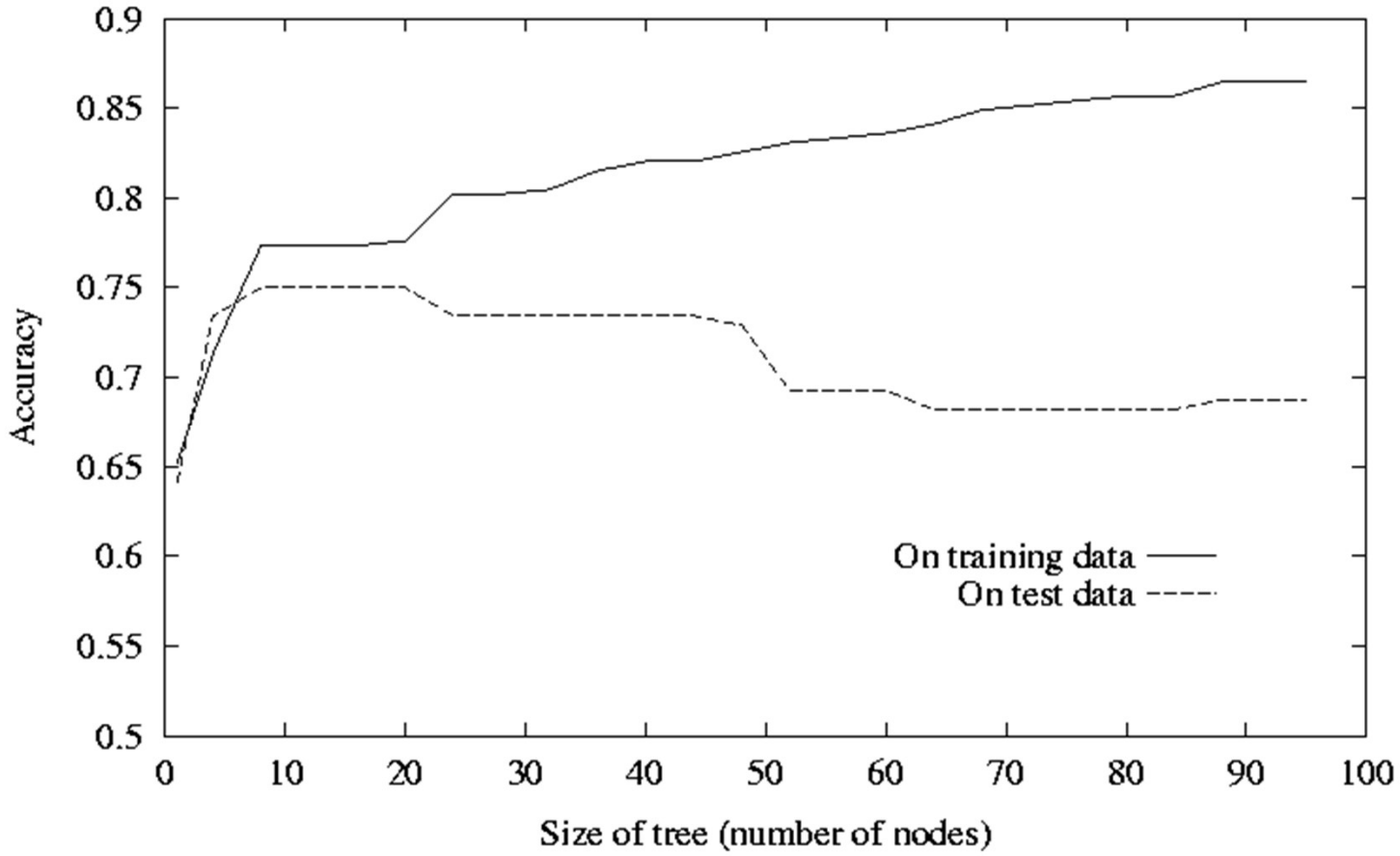
- There is noise in the data
- The number of training examples is too small to produce a representative sample of the true target function

Overfitting

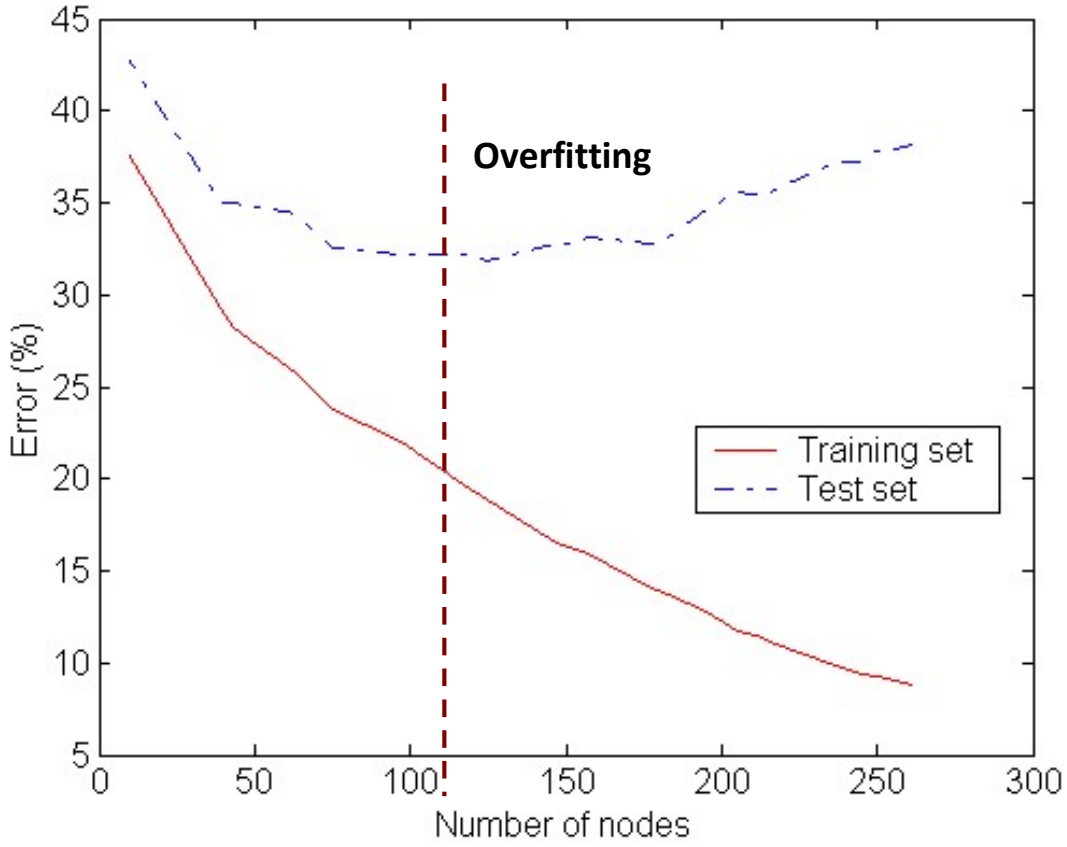
- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data
  - May be based on insufficient data
- A hypothesis *h* is said to overfit the training data if there is another hypothesis, h′, such that *h* has smaller error than *h*′ on the training data but *h* has larger error on the test data than *h*′.



Overfitting

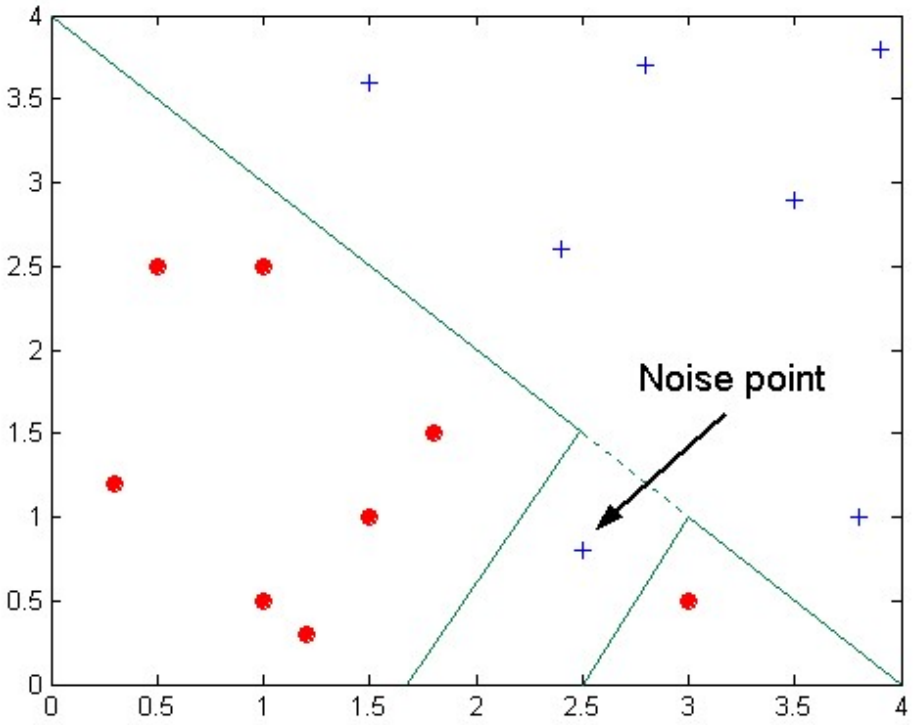


Underfitting and Overfitting



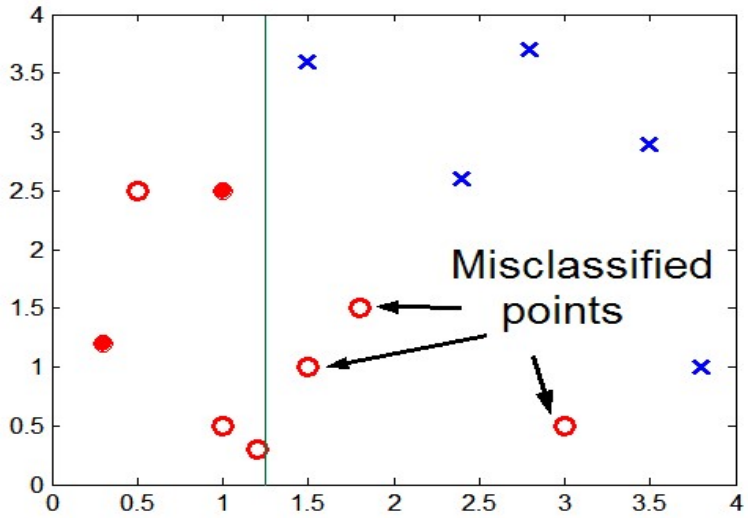
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Lack of data points makes it difficult to predict correctly the class labels of that region

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Overfitting happens when a model is capturing idiosyncrasies of the data rather than generalities.
- Often caused by too many parameters relative to the amount of training data.
- E.g. an order-N polynomial can intersect any N+1 data points

Avoid Overfitting

- There are several approaches to avoiding overfitting in decision tree learning. These can be grouped into two classes:
- Prepruning: Stop growing when data split not statistically significant.
- Postpruning: Grow full tree then remove nodes

- Methods for evaluating subtrees to prune:
  - Minimum description length (MDL):
    Minimize: size(tree) + size(misclassifications(tree))
  - Measure performance over training data
  - Measure performance over separate validation data set

Pre-Pruning (Early Stopping)

- It is the difficult in the this approach of estimating precisely when to stop growing the tree
- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Reduced-error Pruning

Split data into training and validation set.
We use a validation set to prevent overfitting.

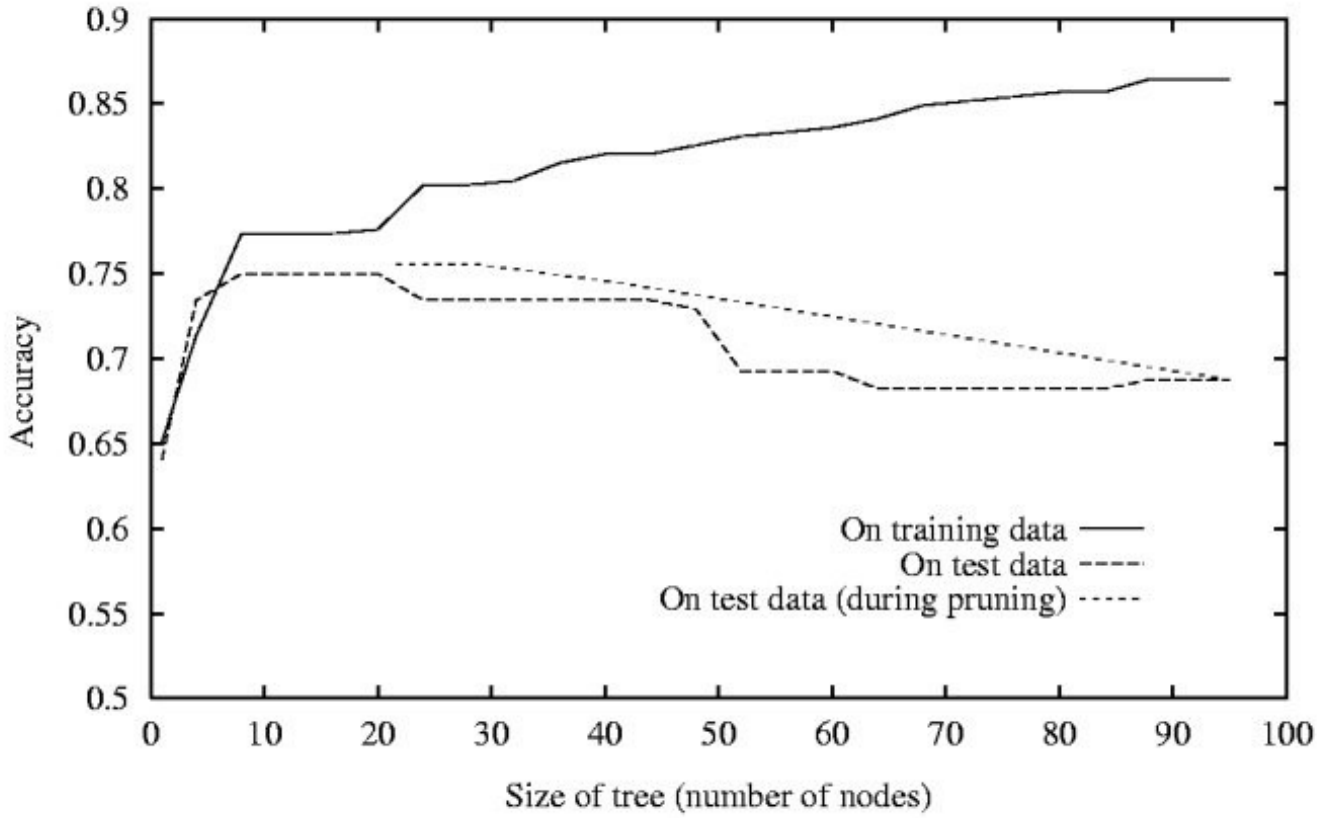
Do until further pruning is harmful(i.e., decreases accuracy of the tree over the validation set):

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Always choose the node whose removal most increases the decision tree accuracy over the validation set

produces smallest version of most accurate subtree
What if data is limited?

General Strategy: Overfit and Simplify

Reduced Error Pruning



Rule Post Pruning

one quite successful method for finding high accuracy hypotheses is a technique we shall call rule post-pruning.

Rule post-pruning involves the following steps:

1. Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.
2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

Rule Post Pruning

There are three main advantages of converting the decision tree to rules before pruning:

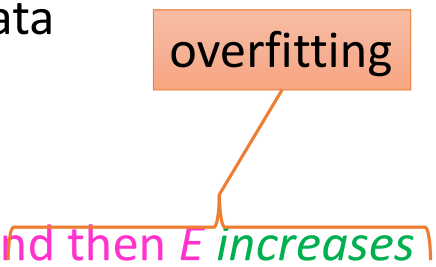
- Converting to rules allows distinguishing among the different contexts in which a decision node is used.
- Converting to rules removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves
- Converting to rules improves readability

Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about H
- Generalization: How well a model performs on new data
- Overfitting: H more complex than C or f
- Underfitting: H less complex than C or f

Triple Trade-Off

- There is a trade-off between three factors:
  - Complexity of H, *c* (H),
  - Training set size, *N,*
  - Generalization error, *E* on new data



- As *N* *increases*, *E* decreases
- As *c* (H) *increases*, first *E* *decreases* and then *E* *increases*
- As *c* (H) *increases*, the training error *decreases* for some time and then stays constant (frequently at 0)

References:

1. Machine Learning: Tom Mitchel book
2. Introduction to Machine Learning (IITKGP)
   Prof. Sudeshna Sarkar