Data Mining

Lesson 3

Decision Trees

MSc in Computer Science

University of New York Tirana

Assoc. Prof. Dr. Marenglen Biba

Data Mining: Content

- Introduction to data mining and machine learning
- Inductive learning
- Decision trees
- Rule induction
- Instance-based learning
- Bayesian learning
- Neural networks
- Support vector machines
- Other machine learning models
- Engineering data mining tasks

Lesson outline

- Decision Trees
- The ID3 Algorithm
- Inductive Bias of ID3
- Language and Preference Bias
- Overfitting

- Lab Session
  - Decision Trees in Weka
  - Decision Trees in Oracle

Introduction

- Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.

- Learned trees can also be re-represented as sets of if-then rules to improve human readability.

- These learning methods are among the most popular of inductive inference algorithms and have been successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants.

Decision Trees Representation

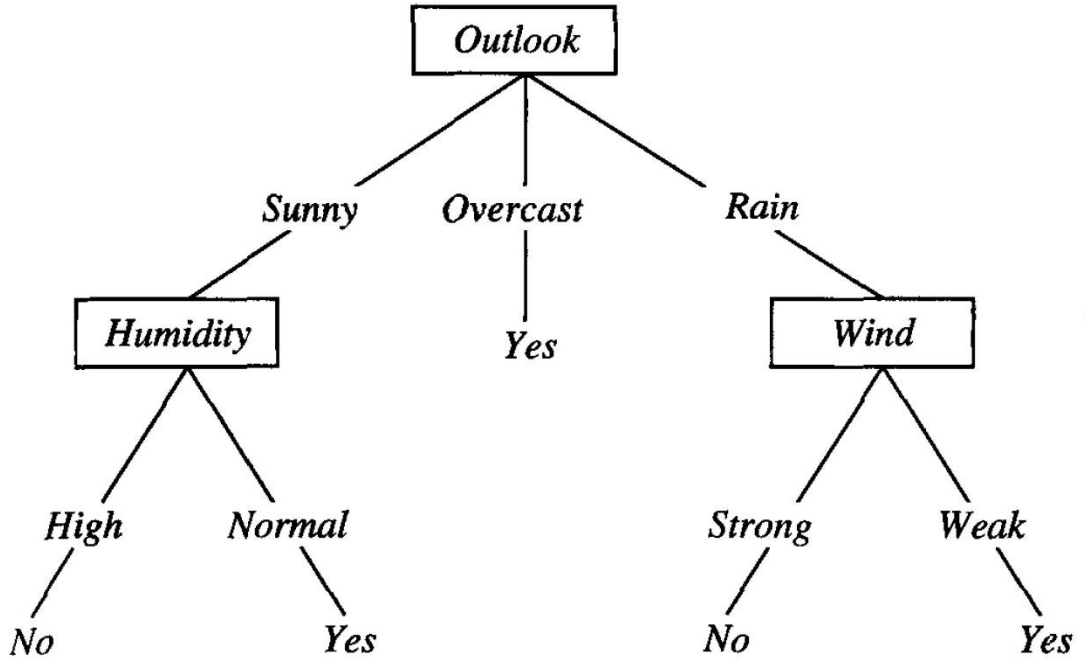


FIGURE 3.1
A decision tree for the concept *PlayTennis*. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, *Yes* or *No*). This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

Decision Trees: Example

- For example, the instance

  {Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong)

  would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative

  instance (i.e., the tree predicts that PlayTennis = no).

Disjunction of conjuctions

- In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.
- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.
- For example, the decision tree shown in Figure 3.1 corresponds to the expression

$$\{\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal})$$

$$\vee\ (\text{Outlook} = \text{Overcast})$$

$$\vee\ (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})$$

Appropriate Problems For Decision Tree Learning

1. *Instances are represented by attribute-value pairs.*
   - Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot).
   - The easiest situation for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., Hot, Mild, Cold).
   - However, extensions to the basic algorithm allow handling real-valued attributes as well (e.g., representing Temperature numerically).

Appropriate Problems For Decision Tree Learning

2. *The target function has discrete output values*.
   - The decision tree in Figure 3.1 assigns a boolean classification (e.g., yes or no) to each example.
   - Decision tree methods easily extend to learning functions with more than two possible output values.
   - A more substantial extension allows learning target functions with real-valued outputs, though the application of decision trees in this setting is less common.

Appropriate Problems For Decision Tree Learning

3. *Disjunctive descriptions may be required.*
   - As noted above, decision trees naturally represent disjunctive expressions.
4. *The training data may contain errors.*
   - Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
5. *The training data may contain missing attribute values.*
   - Decision tree methods can be used even when some training examples have unknown values

     (e.g., if the Humidity of the day is known for only some of the training examples).

The Basic Decision Tree Learning Algorithm

- Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees.

- This approach is exemplified by the ID3 algorithm (Quinlan 1986) and its successor C4.5 (Quinlan 1993).

- www.rulequest.com

The ID3 algorithm

- ID3 learns decision trees by constructing them top-down, beginning with the question "which attribute should be tested at the root of the tree?"

- To answer this question, each instance attribute is evaluated using a statistical test to determine how well it alone classifies the training examples.

- The best attribute is selected and used as the test at the root node of the tree.

The ID3 algorithm

- A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node (i.e., down the branch corresponding to the example's value for this attribute).

- The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree.

- This forms a greedy search for an acceptable decision tree, in which the algorithm never backtracks to reconsider earlier choices.

The 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*

* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

TABLE 3.1
Summary of the ID3 algorithm specialized to learning boolean-valued functions. ID3 is a greedy algorithm that grows the tree top-down, at each node selecting the attribute that best classifies the local training examples. This process continues until the tree perfectly classifies the training examples, or until all attributes have been used.

Which Attribute Is the Best Classifier?

- The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree.
  - We would like to select the attribute that is most useful for classifying examples.
- What is a good quantitative measure of the worth of an attribute?
  - We will define a statistical property, called information gain, that measures how well a given attribute separates the training examples according to their target classification.
- ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.

Entropy Measures Homogeneity Of Examples

- In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy, that characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is:

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

- where p_+ is the proportion of positive examples in S and p_- is the proportion of negative examples in S.
- In all calculations involving entropy we define 0log0 to be 0.

Example of Entropy

- To illustrate, suppose S is a collection of 14 examples of some boolean concept, including 9 positive and 5 negative examples (we adopt the notation [9+, 5-] to summarize such a sample of data).
- Then the entropy of S relative to this boolean classification is

  Entropy{[9+,5-]} = $-(9/14)\ \log_2(9/14) - (5/14)\ \log_2(5/14) = 0.940$
- Notice that the entropy is 0 if all members of S belong to the same class.
- For example, if all members are positive ($p_+ = 1$), then p_- is 0, and Entropy(S) = $– 1 * \log_2(1) – 0 * \log_2 0 = -1* 0 – 0 * \log_2 0 = 0$.
- Note the entropy is 1 when the collection contains an equal number of positive and negative examples.

Entropy

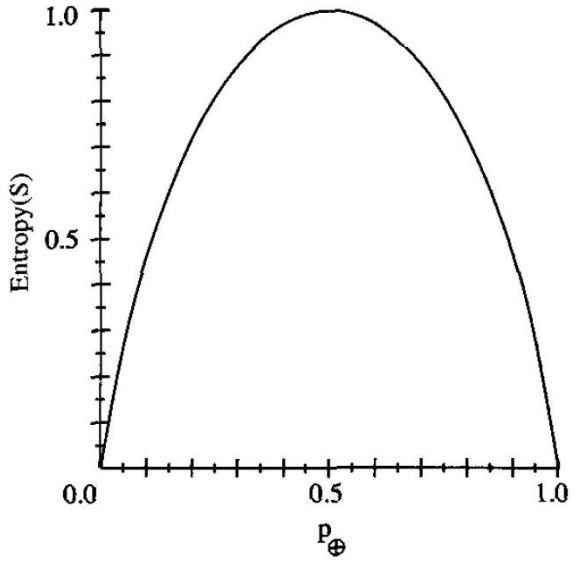


FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_{\oplus}$, of positive examples varies between 0 and 1.

Entropy in information theory

- One interpretation of entropy from information theory is that it specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of S (i.e., a member of S drawn at random with uniform probability).
- For example, if p_+ is 1, the receiver knows the drawn example will be positive, so no message need be sent, and the entropy is zero.
- On the other hand, if p_+ is 0.5, one bit is required to indicate whether the drawn example is positive or negative.

General case

- Thus far we have discussed entropy in the special case where the target classification is boolean.
- More generally, if the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as:

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

where p_i is the proportion of S belonging to class i.

- Note the logarithm is still base 2 because entropy is a measure of the expected encoding length measured in bits.
- Note also that if the target attribute can take on c possible values, the entropy can be as large as $\log_2 c$.

Information Gain Measures The Expected Reduction in Entropy

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data.
- The measure we will use, called information gain, is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.
- More precisely, the information gain, Gain(S, A) of an attribute A, relative to a collection of examples S, is defined as:

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

- where Values(A) is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v (i.e., $S_v = \{s \in S | A(s) = v\}$.

Reduction in Entropy

- Note the first term in the Equation is just the entropy of the original collection S, and the second term is the expected value of the entropy after S is partitioned using attribute A.
- The expected entropy described by this second term is simply the sum of the entropies of each subset S_V, weighted by the fraction of examples that belong to S_v.

- *Gain(S, A) is therefore the expected reduction in entropy caused by knowing the value of attribute A.*

- Put another way, Gain(S, A) is the information provided about the target function value, given the value of some other attribute A.

Example of information gain

- For example, suppose S is a collection of training-example days described by attributes including Wind, which can have the values Weak or Strong. As before, assume S is a collection containing 14 examples, [9+, 5-].
- Of these 14 examples, suppose 6 of the positive and 2 of the negative examples have Wind = Weak, and the remainder have Wind = Strong. The information gain due to sorting the original 14 examples by the attribute Wind may then be calculated as:

$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

Which attribute is the best classifier?

- Information gain is precisely the measure used by ID3 to select the best attribute at each step in growing the tree.

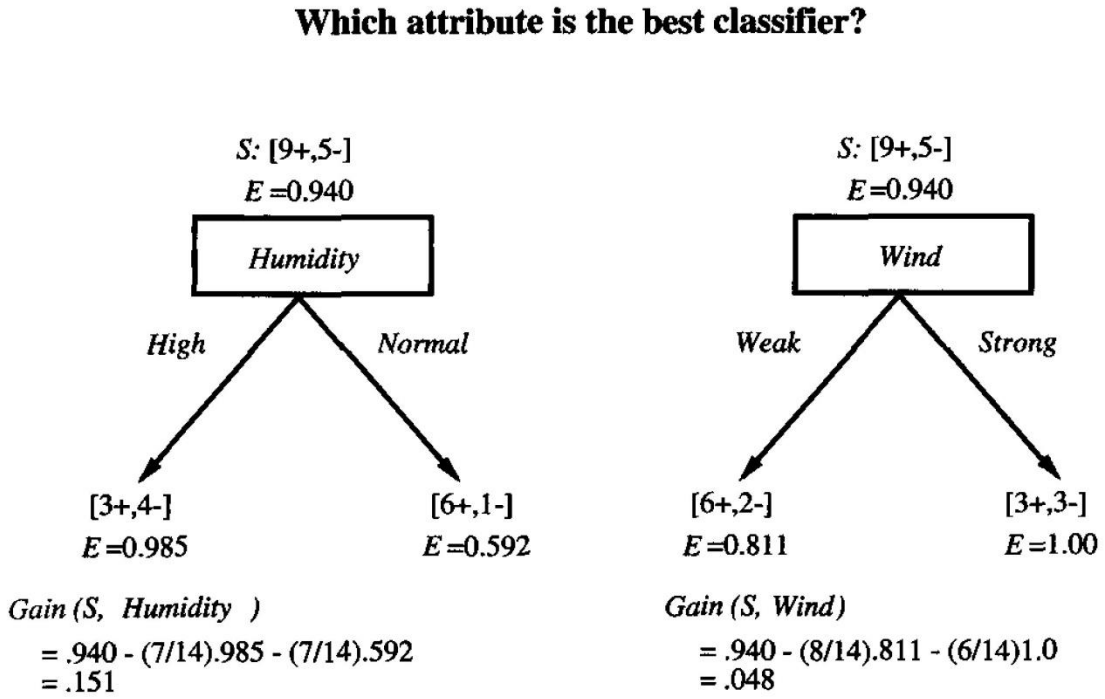


FIGURE 3.3
Humidity provides greater information gain than *Wind*, relative to the target classification. Here, E stands for entropy and S for the original collection of examples. Given an initial collection S of 9 positive and 5 negative examples, [9+, 5−], sorting these by their *Humidity* produces collections of [3+, 4−] ($Humidity = High$) and [6+, 1−] ($Humidity = Normal$). The information gained by this partitioning is .151, compared to a gain of only .048 for the attribute *Wind*.

Illustration of ID3

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

TABLE 3.2
Training examples for the target concept *PlayTennis*.

Illustration of ID3

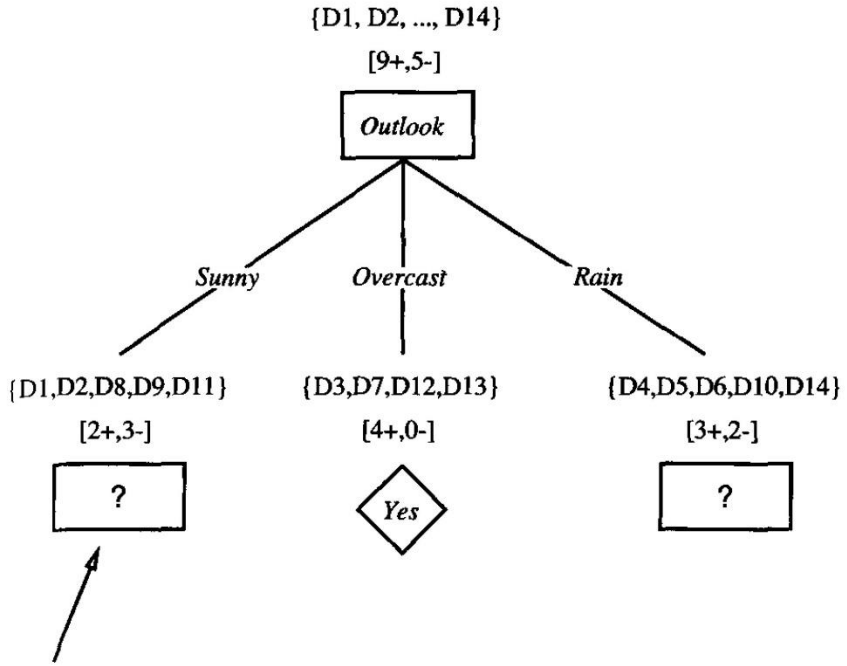
- ID3 determines the information gain for each candidate attribute (i.e., Outlook, Temperature, Humidity, and Wind), then selects the one with highest information gain.
- The computation of information gain for two of these attributes is shown in Figure 3.3.
- The information gain values for all four attributes are:
  - Gain(S, Outlook) = 0.246
  - Gain(S, Humidity) = 0.151
  - GainiS, Wind) = 0.048
  - Gain(S, Temperature) = 0.029

Illustration of ID3

- According to the information gain measure, the Outlook attribute provides the best prediction of the target attribute PlayTennis, over the training examples.

- Therefore, Outlook is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values (i.e., Sunny, Overcast, and Rain).

Illustration of ID3



S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 - (2/5) 1.0 - (3/5) .918 = .019

FIGURE 3.4
The partially learned decision tree resulting from the first step of ID3. The training examples are sorted to the corresponding descendant nodes. The *Overcast* descendant has only positive examples and therefore becomes a leaf node with classification *Yes*. The other two nodes will be further expanded, by selecting the attribute with highest information gain relative to the new subsets of examples.

Constructing the tree

- The process of selecting a new attribute and partitioning the training examples is now repeated for each nonterminal descendant node, this time using only the training examples associated with that node.
- Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree.
- This process continues for each new leaf node until either of two conditions is met:
  - (1) every attribute has already been included along this path through the tree, or
  - (2) the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).

Final constructed tree

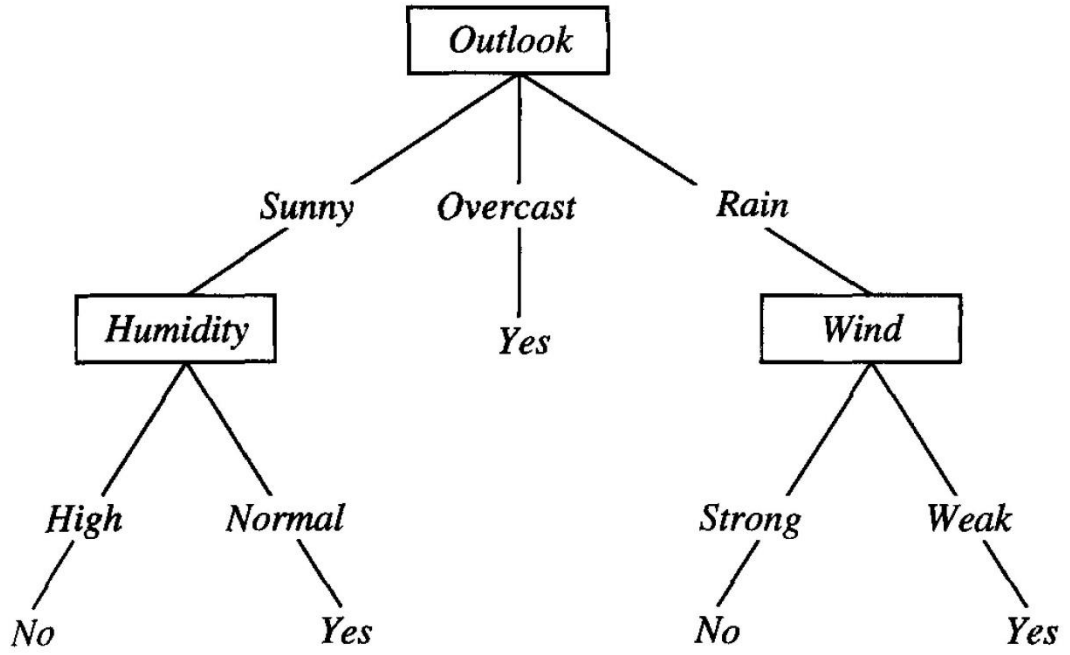


FIGURE 3.1

Hypothesis Space Search in Decision Tree Learning

- As with other inductive learning methods, ID3 can be characterized as searching a space of hypotheses for one that fits the training examples.
- The hypothesis space searched by ID3 is the set of possible decision trees.
- ID3 performs a simple-to-complex, hill-climbing search through this hypothesis space, beginning with the empty tree, then considering progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data.
- The evaluation function that guides this hill-climbing search is the information gain measure.

Hypothesis search space

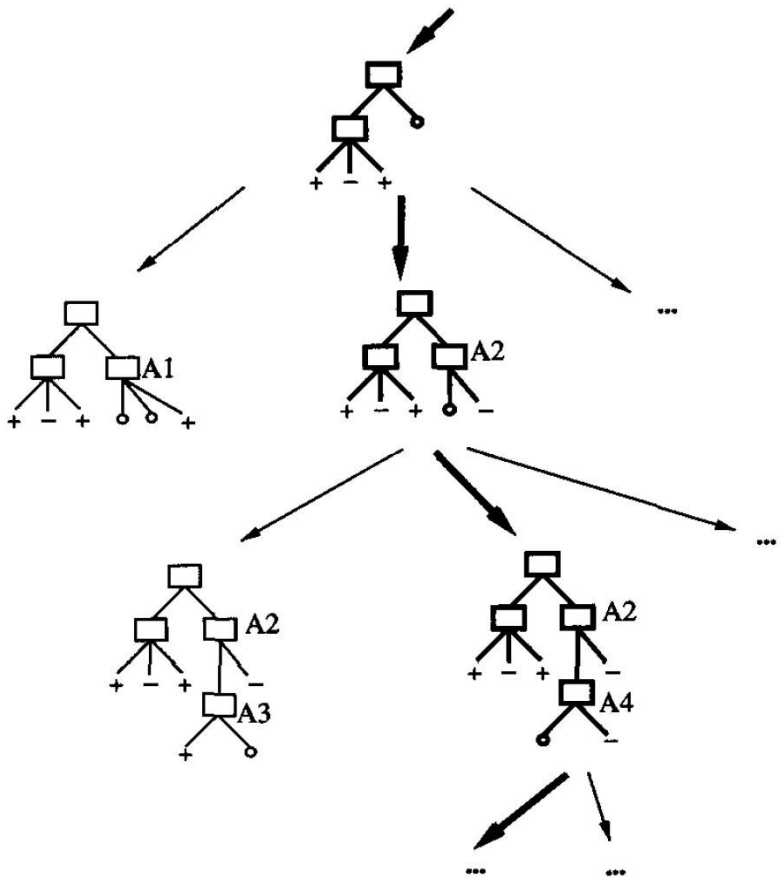


FIGURE 3.5
Hypothesis space search by ID3. ID3 searches through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic.

Complete space of finite discrete-valued functions

- ID3's hypothesis space of all decision trees is a complete space of finite discrete-valued functions, relative to the available attributes.

- Because every finite discrete-valued function can be represented by some decision tree, ID3 avoids one of the major risks of methods that search incomplete hypothesis spaces (such as methods that consider only conjunctive hypotheses): that the hypothesis space might not contain the target function.

Single hypothesis

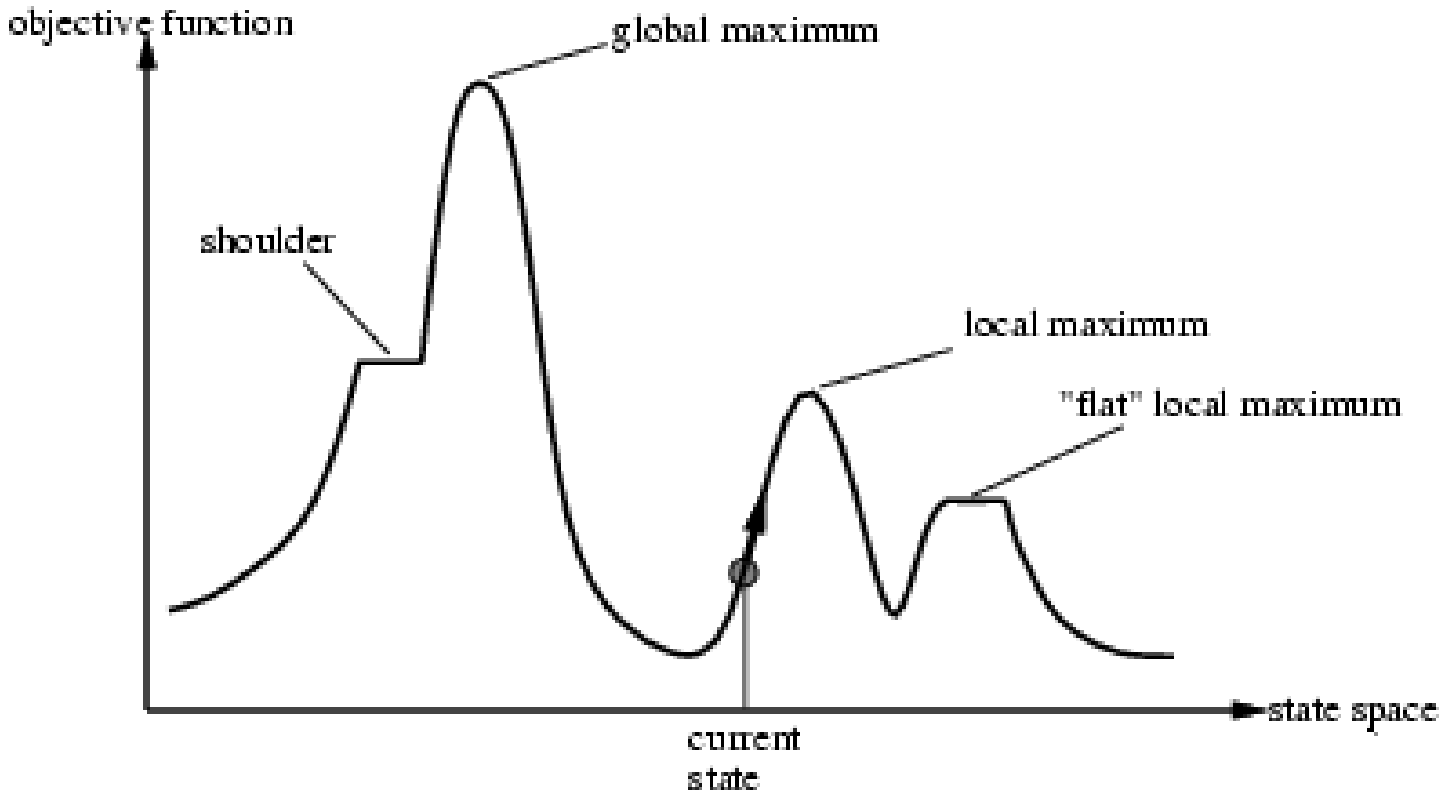
- ID3 maintains only a single current hypothesis as it searches through the space of decision trees.
- This contrasts, for example, with the earlier version space Candidate-Elimination method, which maintains the set of all hypotheses consistent with the available training examples.
- By determining only a single hypothesis, ID3 loses the capabilities that follow from explicitly representing all consistent hypotheses.
- For example, it does not have the ability to determine how many alternative decision trees are consistent with the available training data.

Local optima

- ID3 in its pure form performs no backtracking in its search.
- Once it selects an attribute to test at a particular level in the tree, it never backtracks to reconsider this choice.
- Therefore, it is susceptible to the usual risks of hill-climbing search without backtracking: converging to locally optimal solutions that are not globally optimal.
- In the case of ID3, a locally optimal solution corresponds to the decision tree it selects along the single search path it explores.
- However, this locally optimal solution may be less desirable than trees that would have been encountered along a different branch of the search.

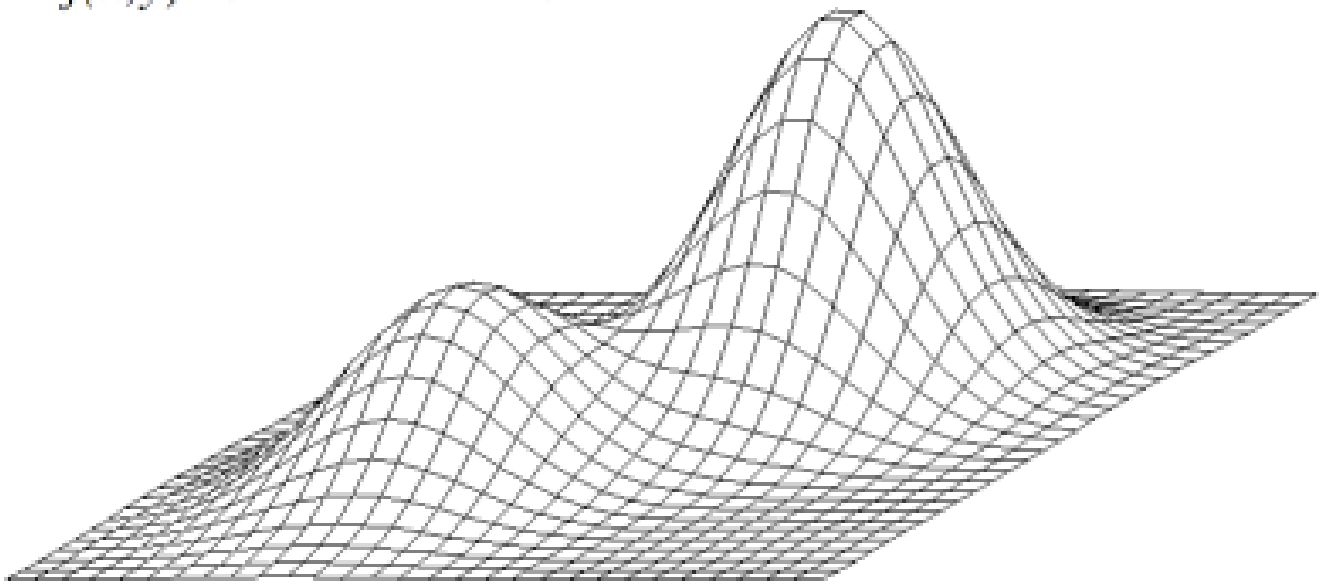
Hill-climbing search

- Problem: depending on initial state, can get stuck in local maxima



Hill Climbing and Local maxima

$$f(x,y)=e^{-(x^2+y^2)} + 2e^{-((x-1.7)^2+(y-1.7)^2)}$$



Use of all training examples

- ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis.

- This contrasts with methods that make decisions incrementally, based on individual training examples (e.g., Find-S or Candidate-Elimination).

- One advantage of using statistical properties of all the examples (e.g., information gain) is that the resulting search is much less sensitive to errors in individual training examples.

- ID3 can be easily extended to handle noisy training data by modifying its termination criterion to accept hypotheses that imperfectly fit the training data.

Inductive Bias In Decision Tree Learning

- What is the policy by which ID3 generalizes from observed training examples to classify unseen instances?

- In other words, what is its inductive bias?

- Recall from Lesson 2 that inductive bias is the set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to future instances.

Inductive Bias In Decision Tree Learning

- Given a collection of training examples, there are typically many decision trees consistent with these examples.
- Describing the inductive bias of ID3 therefore consists of describing the basis by which it chooses one of these consistent hypotheses over the others.
  - 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.
- Roughly speaking, then, the ID3 search strategy:
  - (a) selects in favor of shorter trees over longer ones, and
  - (b) selects trees that place the attributes with highest information gain closest to the root.

Inductive Bias In Decision Tree Learning

- Because of the subtle interaction between the attribute selection heuristic used by ID3 and the particular training examples it encounters, it is difficult to characterize precisely the inductive bias exhibited by ID3.

- However, we can approximately characterize its bias as a preference for short decision trees over complex trees.

A closer approximation to the inductive bias of ID3

- *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

- There is an interesting difference between the types of inductive bias exhibited by ID3 and by the Candidate-Elimination algorithm discussed in Lesson 2.

1. Consider the difference between the hypothesis space search in these two approaches:
   - ID3 searches a complete hypothesis space (i.e., one capable of expressing any finite discrete-valued function).
   - It searches incompletely through this space, from simple to complex hypotheses, until its termination condition is met (e.g., until it finds a hypothesis consistent with the data).
   - Its inductive bias is solely a consequence of the ordering of hypotheses by its search strategy.
   - Its hypothesis space introduces no additional bias.

Restriction Biases and Preference Biases

2. The version space Candidate-Elimination algorithm searches an incomplete hypothesis space (i.e., one that can express only a subset of the potentially teachable concepts).
   - It searches this space completely, finding every hypothesis consistent with the training data.
   - Its inductive bias is solely a consequence of the expressive power of its hypothesis representation.
   - Its search strategy introduces no additional bias.

Language Bias

- In brief, the inductive bias of ID3 follows from its search strategy, whereas the inductive bias of the Candidate-Elimination algorithm follows from the definition of its search space.
- 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).
- In contrast, 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).

Which bias to use?

- Given that some form of inductive bias is required in order to generalize beyond the training data, which type of inductive bias shall we prefer; a preference bias or restriction bias?
- Typically, a preference bias is more desirable than a restriction bias, because it allows the learner to work within a complete hypothesis space that is assured to contain the unknown target function.
- In contrast, a restriction bias that strictly limits the set of potential hypotheses is generally less desirable, because it introduces the possibility of excluding the unknown target function altogether.

Combining biases

- Whereas ID3 exhibits a purely preference bias and Candidate-Elimination a purely restriction bias, some learning systems combine both.
- Consider, for example, the program described in Lesson 1 for learning a numerical evaluation function for game playing.
- In this case, the learned evaluation function is represented by a linear combination of a fixed set of board features, and the learning algorithm adjusts the parameters of this linear combination to best fit the available training data.
- In this case, the decision to use a linear function to represent the evaluation function introduces a restriction bias (nonlinear evaluation functions cannot be represented in this form).
- At the same time, the choice of a particular parameter tuning method (the LMS algorithm in this case) introduces a preference bias stemming from the ordered search through the space of all possible parameter values.

Why Prefer Short Hypotheses?

- Is ID3's inductive bias of favoring shorter decision trees a sound basis for generalizing beyond the training data?
- Philosophers and others have debated this question for centuries, and the debate remains unresolved to this day.
- William of Occam was one of the first to discuss the question, around the year 1320, so this bias often goes by the name of Occam's razor.

- *Occam's razor: Prefer the simplest hypothesis that fits the data.*

Preferring simple hypothesis

- Giving an inductive bias a name does not justify it. Why should one prefer simpler hypotheses?
- Notice that scientists sometimes appear to follow this inductive bias. Physicists, for example, prefer simple explanations for the motions of the planets, over more complex explanations. Why?
- One argument is that because there are fewer short hypotheses than long ones (based on straightforward combinatorial arguments), it is less likely that one will find a short hypothesis that coincidentally fits the training data.
- In contrast there are often many very complex hypotheses that fit the current training data but fail to generalize correctly to subsequent data.

Preferring simple hypothesis

- Consider decision tree hypotheses.
- There are many more 500-node decision trees than 5-node decision trees.
- Given a small set of 20 training examples, we might expect to be able to find many 500-node decision trees consistent with these, whereas we would be more surprised if a 5-node decision tree could perfectly fit this data.
- We might therefore believe the 5-node tree is less likely to be a statistical coincidence and prefer this hypothesis over the 500-node hypothesis.

Issues In Decision Tree Learning

- Practical issues in learning decision trees include:
  - determining how deeply to grow the decision tree
  - handling continuous attributes
  - choosing an appropriate attribute selection measure
  - handling training data with missing attribute values
  - handling attributes with differing costs
  - improving computational efficiency.

- ID3 has itself been extended to address most of these issues, with the resulting system renamed C4.5 (Quinlan 1993).

Overfitting

Avoiding Overfitting the Data

- *Definition: Given a hypothesis space H, a hypothesis h ε H is said to overfit the training data if there exists some alternative hypothesis h' ε H, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances (not just training ones).*

Overfitting

- In this case, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes.

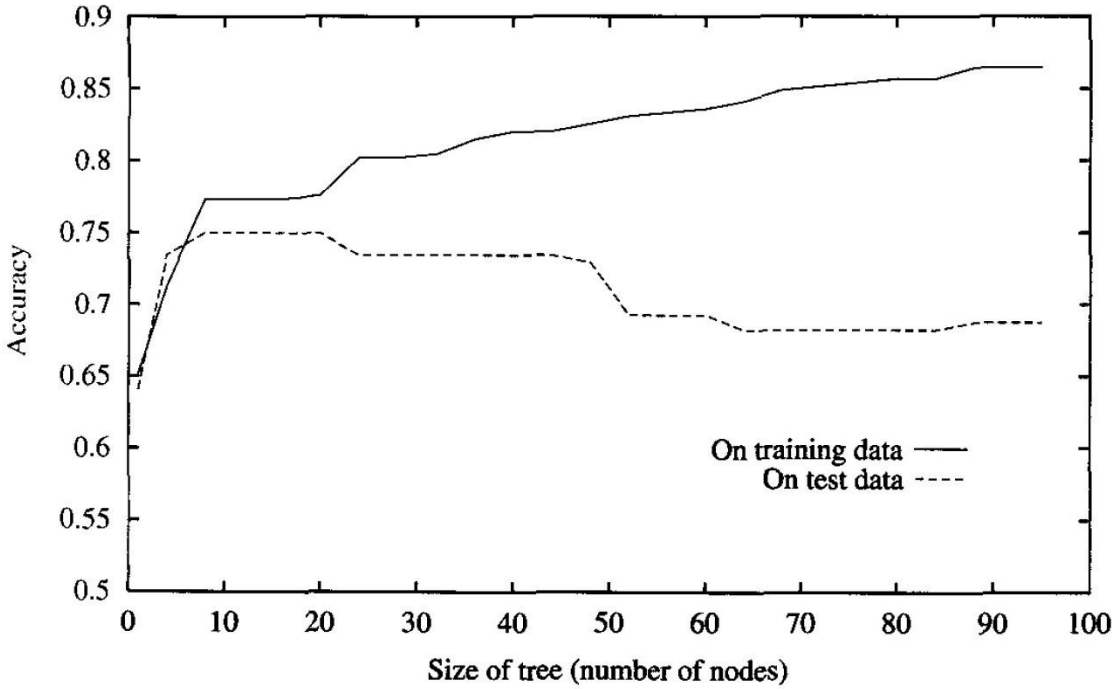


FIGURE 3.6
Overfitting in decision tree learning. As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically. However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases. Software and data for experimenting with variations on this plot are available on the World Wide Web at http://www.cs.cmu.edu/~tom/mlbook.html.

Overfitting

- How can it be possible for tree h to fit the training examples better than h', but for it to perform more poorly over subsequent examples?
- One way this can occur is when the training examples contain random errors or noise.
- In fact, overfitting is possible even when the training data are noise-free, especially when small numbers of examples are associated with leaf nodes.
- In this case, it is quite possible for coincidental regularities to occur, in which some attribute happens to partition the examples very well, despite being unrelated to the actual target function.
- Whenever such coincidental regularities exist, there is a risk of overfitting.

Impact of Overfitting

- Overfitting is a significant practical difficulty for decision tree learning and many other learning methods.

- For example, in one experimental study of ID3 involving five different learning tasks with noisy, nondeterministic data (Mingers 1989b), overfitting was found to decrease the accuracy of learned decision trees by 10-25% on most problems.

Approaches against overfitting

- There are several approaches to avoiding overfitting in decision tree learning.
- These can be grouped into two classes:
  - approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,
  - approaches that allow the tree to overfit the data, and then post-prune the tree.
- Although the first of these approaches might seem more direct, the second approach of post-pruning overfit trees has been found to be more successful in practice.
  - This is due to the difficulty in the first approach of estimating precisely when to stop growing the tree.

Determining the correct final tree size

- Regardless of whether the correct tree size is found by stopping early or by post-pruning, a key question is what criterion is to be used to determine the correct final tree size.

Approaches include:

1. Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
2. Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set.
    - For example, Quinlan (1986) uses a chi-square test to estimate whether further expanding a node is likely to improve performance over the entire instance distribution, or only on the current sample of training data.

Determining the correct final tree size

3. Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized.
   - This approach, based on a heuristic called the Minimum Description Length principle, is discussed later in this course, as well as in Quinlan and Rivest (1989) and Mehta et al. (1995).

Validation Set

- The first of the three approaches is the most common and is often referred to as a training and validation set approach.
- We discuss the two main variants of this approach below.
- In this approach, the available data are separated into two sets of examples:
  - a training set, which is used to form the learned hypothesis
  - a separate validation set, which is used to evaluate the accuracy of this hypothesis over subsequent data and, in particular, to evaluate the impact of pruning this hypothesis.

Validation Set: Motivation

- The motivation is this: Even though the learner may be misled by random errors and coincidental regularities within the training set, the validation set is unlikely to exhibit the same random fluctuations.
- Therefore, the validation set can be expected to provide a safety check against overfitting the spurious characteristics of the training set.
- Of course, it is important that the validation set be large enough to itself provide a statistically significant sample of the instances.
- One common heuristic is to withhold one-third of the available examples for the validation set, using the other two-thirds for training.

Reduced Error Prunning

- How exactly might we use a validation set to prevent overfitting?

- One approach, called reduced-error pruning (Quinlan 1987), is to consider each of the decision nodes in the tree to be candidates for pruning.

- Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node.

Reduced Error Prunning

- Nodes are removed only if the resulting pruned tree performs better or equal than the original over the validation set.
- This has the effect that any leaf node added due to coincidental regularities in the training set is likely to be pruned because these same coincidences are unlikely to occur in the validation set.
- Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the validation set.
- Pruning of nodes continues until further pruning is harmful (i.e., decreases accuracy of the tree over the validation set).

Reduced Error Prunning

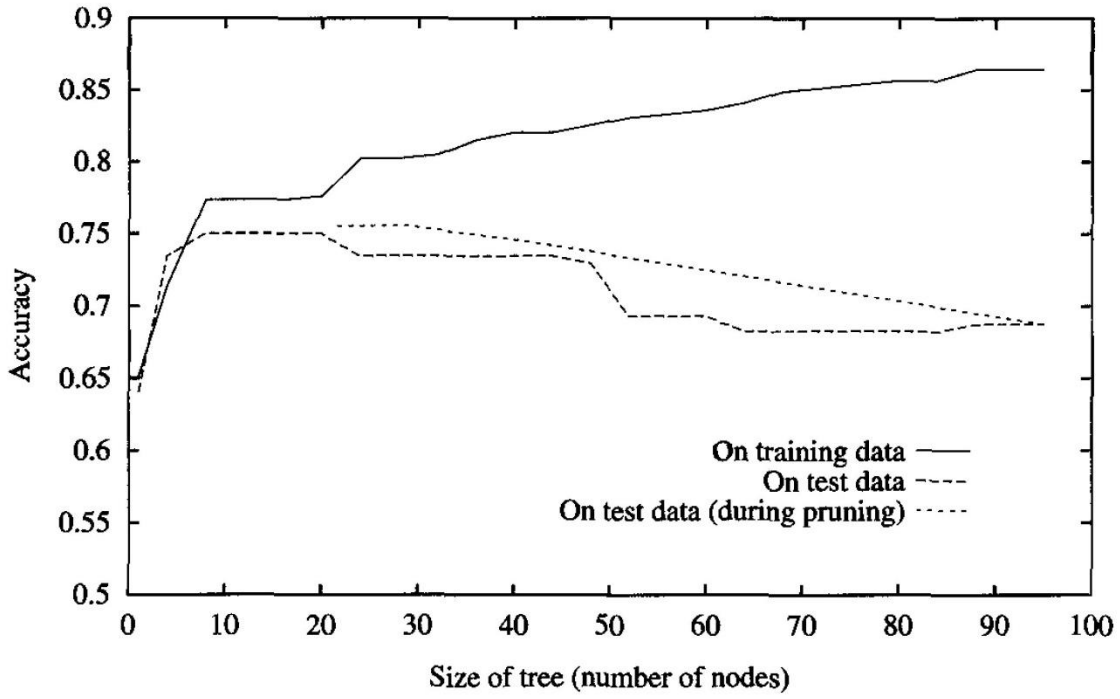


FIGURE 3.7
Effect of reduced-error pruning in decision tree learning. This plot shows the same curves of training and test set accuracy as in Figure 3.6. In addition, it shows the impact of reduced error pruning of the tree produced by ID3. Notice the increase in accuracy over the test set as nodes are pruned from the tree. Here, the validation set used for pruning is distinct from both the training and test sets.

Rule Post-Pruning

- In practice, one quite successful method for finding high accuracy hypotheses is a technique we shall call rule post-pruning.
- A variant of this pruning method is used by C4.5 (Quinlan 1993), which is an outgrowth of the original ID3 algorithm.

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.

Incorporating Continuous-Valued Attributes

- Our initial definition of ID3 is restricted to attributes that take on a discrete set of values.
  - First, the target attribute whose value is predicted by the learned tree must be discrete valued.
  - Second, the attributes tested in the decision nodes of the tree must also be discrete valued. This second restriction can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree.
- This can be accomplished by dynamically defining new discrete- valued attributes that partition the continuous attribute value into a discrete set of intervals.
- This is called discretization and it is available in Weka also.

Handling Training Examples with Missing Attribute Values

- In certain cases, the available data may be missing values for some attributes.
- Consider the situation in which Gain(S, A) is to be calculated at node n in the decision tree to evaluate whether the attribute A is the best attribute to test at this decision node.
- Suppose that {x, c(x)} is one of the training examples in S and that the value A(x) is unknown.
- One strategy for dealing with the missing attribute value is to assign it the value that is most common among training examples at node n.
- Alternatively, we might assign it the most common value among examples at node n that have the classification c(x).
- The elaborated training example using this estimated value for A(x) can then be used directly by the existing decision tree learning algorithm.
- Filters in Weka allow to do this.

Handling Attributes with Differing Costs

- In some learning tasks the instance attributes may have associated costs.
- For example, in learning to classify medical diseases we might describe patients in terms of attributes such as Temperature, BiopsyResult, Pulse, BloodTestResults, etc.
- These attributes vary significantly in their costs, both in terms of monetary cost and cost to patient comfort.
- In such tasks, we would prefer decision trees that use low-cost attributes where possible, relying on high-cost attributes only when needed to produce reliable classifications.

Attributes with Differing Costs in ID3

- ID3 can be modified to take into account attribute costs by introducing a cost term into the attribute selection measure.

- For example, we might divide the Gain by the cost of the attribute, so that lower-cost attributes would be preferred.

- While such cost-sensitive measures do not guarantee finding an optimal cost-sensitive decision tree, they do bias the search in favor of low-cost attributes.

Readings for this part

- Machine Learning. T. Mitchell
  - Chapter 3

Lab Session

- Decision Trees in Weka
- Decision Trees in Oracle

Decision Trees in Weka

Examples with DTs

- In all the examples no dataset engineering is performed, just the parameters of the algorithms are changed in order to show the impact of changing only the parameters of the model.

Iris

- Iris dataset
  - **Attribute Information:**
    1. sepal length in cm
    2. sepal width in cm
    3. petal length in cm
    4. petal width in cm
    5. class:
    -- Iris Setosa
    -- Iris Versicolour
    -- Iris Virginica
- http://archive.ics.uci.edu/ml/datasets/Iris
- In Weka Format from Weka website

- Default parameters
  - Accuracy 96%

- Other parameter changes
  - No accuracy improvement

Colic

- Colic
  - Classification task: whether lesion is surgical
  - http://archive.ics.uci.edu/ml/datasets/Horse+Colic
- Default parameters (some are already optimized by default)
  - Rule post-pruning of C4.5 is used by default
  - Accuracy 85.32%
- Optimized parameters
  - Confidence Factor: used for pruning
  - Conf. Factor: 0.2, accuracy 85.59
  - Smaller Conf. Factor, accuracy 85.59: no more improvement is possible
- If you use reduced-error pruning
  - Conf. Factor: 0.25, accuracy 84.51
- Unpruned (true in Weka): 82.33

Credit card approval

- Classification: approve or not credit card
- This file concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.
- This dataset is interesting because there is a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

- Australian and German credit dataset
- http://archive.ics.uci.edu/ml/datasets/Statlog+%28Australian+Credit+Approval%29

Credit card approval - AU

- Default params
  - Acc. 86.08%

- Optimized parameters
  - Conf.factor 0.4 (less pruning), Acc.: 86.23%
- Reduced error pruning
  - Accuracy: 83.33%.
- Unpruned (true in Weka)
  - Accuracy: 81.88%.

Diabetes

- **Attribute Information:**

  1. Number of times pregnant
  2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
  3. Diastolic blood pressure (mm Hg)
  4. Triceps skin fold thickness (mm)
  5. 2-Hour serum insulin (mu U/ml)
  6. Body mass index (weight in kg/(height in m)^2)
  7. Diabetes pedigree function
  8. Age (years)
  9. Class variable (0 or 1)

- Default parameters
  - Acc. 73.82%
- Other parameters
  - Conf. Fact 0.2, Acc. 74.08%

Heart Disease

- **Attribute Information:**

Only 14 attributes used:

1. #3 (age)
2. #4 (sex)
3. #9 (cp)
4. #10 (trestbps)
5. #12 (chol)
6. #16 (fbs)
7. #19 (restecg)
8. #32 (thalach)
9. #38 (exang)
10. #40 (oldpeak)
11. #41 (slope)
12. #44 (ca)
13. #51 (thal)
14. #58 (num) (the predicted attribute)

- Default parameters
  - Acc. 77.55%
- Other parameters
  - Conf. Fact. 0.5, Acc. 78.54%

Glass Identification

- **Attribute Information:**
- 1. Id number: 1 to 214
  2. RI: refractive index
  3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
  4. Mg: Magnesium
  5. Al: Aluminum
  6. Si: Silicon
  7. K: Potassium
  8. Ca: Calcium
  9. Ba: Barium
  10. Fe: Iron
  11. Type of glass: (class attribute)
  -- 1 building_windows_float_processed
  -- 2 building_windows_non_float_processed
  -- 3 vehicle_windows_float_processed
  -- 4 vehicle_windows_non_float_processed (none in this database)
  -- 5 containers
  -- 6 tableware
  -- 7 headlamps

Hepatitis

- Attributes
- 1. Class: DIE, LIVE
  2. AGE: 10, 20, 30, 40, 50, 60, 70, 80
  3. SEX: male, female
  4. STEROID: no, yes
  5. ANTIVIRALS: no, yes
  6. FATIGUE: no, yes
  7. MALAISE: no, yes
  8. ANOREXIA: no, yes
  9. LIVER BIG: no, yes
  10. LIVER FIRM: no, yes
  11. SPLEEN PALPABLE: no, yes
  12. SPIDERS: no, yes
  13. ASCITES: no, yes
  14. VARICES: no, yes
  15. BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00
  -- see the note below
  16. ALK PHOSPHATE: 33, 80, 120, 160, 200, 250
  17. SGOT: 13, 100, 200, 300, 400, 500,
  18. ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0, 6.0
  19. PROTIME: 10, 20, 30, 40, 50, 60, 70, 80, 90
  20. HISTOLOGY: no, yes

Thyroid Disease

- # From Garavan Institute
  # Documentation: as given by Ross Quinlan
  # 6 databases from the Garavan Institute in Sydney, Australia
  # Approximately the following for each database:

- 2800 training (data) instances and 972 test instances
- Plenty of missing data
- 29 or so attributes, either Boolean or continuously-valued

Labor relation

- **Attribute Information:**
- 1. dur: duration of agreement
  [1..7]
  2 wage1.wage : wage increase in first year of contract
  [2.0 .. 7.0]
  3 wage2.wage : wage increase in second year of contract
  [2.0 .. 7.0]
  4 wage3.wage : wage increase in third year of contract
  [2.0 .. 7.0]
  5 cola : cost of living allowance
  [none, tcf, tc]
  6 hours.hrs : number of working hours during week
  [35 .. 40]
  7 pension : employer contributions to pension plan
  [none, ret_allw, empl_contr]
  8 stby_pay : standby pay
  [2 .. 25]
  9 shift_diff : shift differencial : supplement for work on II and III shift
  [1 .. 25]
  10 educ_allw.boolean : education allowance
  [true false]
  11 holidays : number of statutory holidays
  [9 .. 15]
  12 vacation : number of paid vacation days
  [ba, avg, gnr]
  13 lngtrm_disabil.boolean : employer's help during employee longterm disability
  [true , false]
  14 dntl_ins : employers contribution towards the dental plan
  [none, half, full]
  15 bereavement.boolean : employer's financial contribution towards the covering the costs of bereavement
  [true , false]
  16 empl_hplan : employer's contribution towards the health plan
  [none, half, full]

Letter Recognition

Primary Tumor

- **Attribute Information:**
- --- NOTE: All attribute values in the database have been entered as numeric values corresponding to their index in the list of attribute values for that attribute domain as given below.
  1. class: lung, head & neck, esophasus, thyroid, stomach, duoden & sm.int, colon, rectum, anus, salivary glands, pancreas, gallblader, liver, kidney, bladder, testis, prostate, ovary, corpus uteri, cervix uteri, vagina, breast
  2. age: <30, 30-59, >=60
  3. sex: male, female
  4. histologic-type: epidermoid, adeno, anaplastic
  5. degree-of-diffe: well, fairly, poorly
  6. bone: yes, no
  7. bone-marrow: yes, no
  8. lung: yes, no
  9. pleura: yes, no
  10. peritoneum: yes, no
  11. liver: yes, no
  12. brain: yes, no
  13. skin: yes, no
  14. neck: yes, no
  15. supraclavicular: yes, no
  16. axillar: yes, no
  17. mediastinum: yes, no
  18. abdominal: yes, no

Sonar

- **Abstract**: The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.

Soybean

- R.S. Michalski and R.L. Chilausky
  "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis", International Journal of Policy Analysis and Information Systems, Vol. 4, No. 2, 1980.

Vehicles

- The purpose is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette.
- The vehicle may be viewed from one of many different angles. attribute 'Class' {opel,saab,bus,van}

Vote

- **Abstract**: 1984 United Stated Congressional Voting Records; Classify as Republican or Democrat
- Attribute Information:
- 1. Class Name: 2 (democrat, republican)
- 2. handicapped-infants: 2 (y,n)
- 3. water-project-cost-sharing: 2 (y,n)
- 4. adoption-of-the-budget-resolution: 2 (y,n)
- 5. physician-fee-freeze: 2 (y,n)
- 6. el-salvador-aid: 2 (y,n)
- 7. religious-groups-in-schools: 2 (y,n)
- 8. anti-satellite-test-ban: 2 (y,n)
- 9. aid-to-nicaraguan-contras: 2 (y,n)
- 10. mx-missile: 2 (y,n)
- 11. immigration: 2 (y,n)
- 12. synfuels-corporation-cutback: 2 (y,n)
- 13. education-spending: 2 (y,n)
- 14. superfund-right-to-sue: 2 (y,n)
- 15. crime: 2 (y,n)
- 16. duty-free-exports: 2 (y,n)
- 17. export-administration-act-south-africa: 2 (y,n)

Vowel

- Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of lpc derived log area ratios.

Zoo

- **Abstract**: Artificial, 7 classes of animals
- Attribute Information: (name of attribute and type of value domain)
- 1. animal name: Unique for each instance
- 2. hair Boolean
- 3. feathers Boolean
- 4. eggs Boolean
- 5. milk Boolean
- 6. airborne Boolean
- 7. aquatic Boolean
- 8. predator Boolean
- 9. toothed Boolean
- 10. backbone Boolean
- 11. breathes Boolean
- 12. venomous Boolean
- 13. fins Boolean
- 14. legs Numeric (set of values: {0,2,4,5,6,8})
- 15. tail Boolean
- 16. domestic Boolean
- 17. catsize Boolean
- 18. type Numeric (integer values in range [1,7])

Decision Trees in Oracle

Decision Trees in Oracle

- Oracle Tutorial on classification models
  - Describe the use of classification models when applying supervised learning techniques
  - Create a workflow that uses classification models
  - Train, test, and score data
  - Create output for predictive results

End of class