Decision Trees

Connectionist and Statistical Language Processing

Frank Keller

keller@coli.uni-sb.de

Computerlinguistik
Universität des Saarlandes

Overview

- motivating examples
- appropriate problems for decision trees
- decision tree induction
- entropy and information gain
- the ID3 algorithm
- features of ID3: search strategy, inductive bias
- Occam's razor
- avoiding overfitting

Literature: Witten and Frank (2000: ch. 4), Mitchell (1997: ch. 3).

Revision: Machine Learning Problems

Main classes of machine learning problems:

- *Classification:* learn to put instances into pre-defined classes, e.g., **decision tree, Bayesian classifiers**.
- *Association:* learn relationships between attributes (note dealt with in this course).
- *Numeric prediction:* learn to predict a numeric quantity instead of a class, e.g., **linear models**.
- *Clustering:* discover classes of instances that belong together, e.g., ***k*-means clustering**.

A Sample Data Set

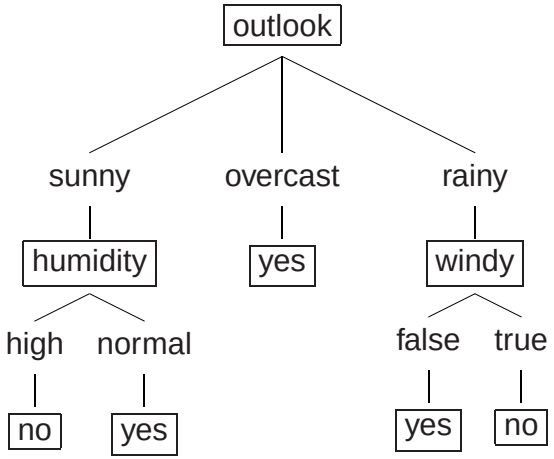
Fictional data set that describes the weather conditions for playing some unspecified game.

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes

outlook	temp.	humidity	windy	play
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

A Sample Decision Tree

Example for a decision tree for the weather data:



Decision Trees: Appropriate Problems

- *Instances represented by attribute-value pairs:* each instance consists of an attribute (outlook, humidity, etc.) with discrete values (sunny, overcast, rainy).
- *Target function with discrete output values:* the decision tree assigns a discrete classification.
- *Disjunctive descriptions may be required:* each path in the tree represents a disjunction of attribute combinations.
- *Training data may contain errors:* decision trees are robust to classification errors and attribute errors in training data.
- *Training data may contain missing values:* decision trees can be used even if instances have missing attributes.

Decision Tree Induction

Key idea: determine the attribute that best classifies the training data; use this attribute at the root of the tree. Repeat this process at for each branch.

This means we are performing **top-down, greedy** search through the space of possible decision trees.

Key problem: How do we choose the best attribute? Answer: use the attribute with the highest **information gain.**

The resulting algorithm is called **ID3** (induction of decision trees based on information gain) and is due to Quinlan (1986).

Entropy

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

S: training data set
c: number of target classes
p_i: proportion of examples in S belonging to target class i

Information theoretic interpretation: number of bits required to encode the classification of an arbitrary member of S.

If all instances in S belong to the same class, then $E(S) = 0$.

If S contains the same number of instances for each class, then $E(S) = 1$.

Entropy: Example

Compute the entropy the weather data set: binary classification into the target classes yes and no.

Out of 14 instances, 9 are classified as yes, and 5 as no.

$p_{\text{yes}} = -(9/14)\log_2(9/14) = 0.41$
$p_{\text{no}} = -(5/14)\log_2(5/14) = 0.53$

$E(S) = p_{\text{yes}} + p_{\text{no}} = 0.94$

Information Gain

$$\text{Gain}(S,A) = E(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} E(S_v) \quad (2)$$

$\text{Values}(A)$: set of all possible values of attribute A
S_v: subset of S for which A has value v
$|S|$: size of S; $|S_v|$: size of S_v

The information gain $\text{Gain}(I,A)$ is the *expected reduction in entropy* caused by knowing the value of the attribute A.

Information Gain: Example

Compute the information gain for the attribute windy in the weather data set:

$$\begin{aligned}\text{Gain}(S,\text{windy}) &= E(S) - \tfrac{|S_{\text{false}}|}{|S|}E(S_{\text{false}}) - \tfrac{|S_{\text{true}}|}{|S|}E(S_{\text{true}}) \\ &= 0.94 - (8/14)0.81 - (6/14)1.00 \\ &= 0.048\end{aligned}$$

$$\begin{aligned}\text{Gain}(S,\text{humidity}) &= E(S) - \tfrac{|S_{\text{high}}|}{|S|}E(S_{\text{high}}) - \tfrac{|S_{\text{normal}}|}{|S|}E(S_{\text{normal}}) \\ &= 0.94 - (7/14)0.96 - (7/14)0.59 \\ &= 0.15\end{aligned}$$

Humidity provides greater information gain than windy, relative to the target classification.

ID3 Algorithm

Informal formulation of ID3:

- Determine the attribute that has the highest information gain on the training set.
- Use this attribute as the root of the tree, create a branch for each of the values that the attribute can take.
- for each of the branches, repeat this process with the subset of the training set that is classified by this branch.

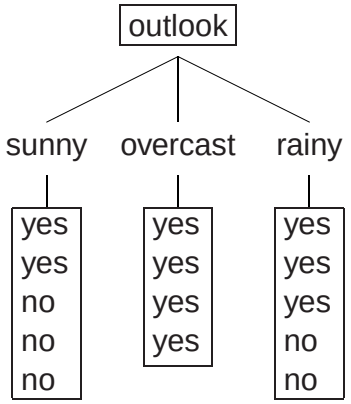
On the next slide, we give the pseudocode for ID3 (slightly simplified).

ID3 Algorithm

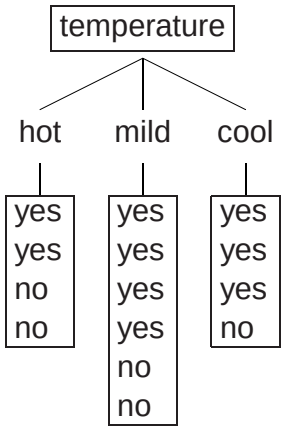
```
ID3(Examples,Target_attribute,Attributes)
  create a new node Root for the tree;
  if all members of Examples are in the same class C
    Root = single-node tree with label = C;
  else if Attributes is empty
    Root = single node tree with label = most common
      value of Target_attribute in Examples;
  else
    A := member of Attributes that maximizes Gain(Examples,A);
    A is decision attribute for Root;
    for each possible value v of A
      add a new branch below Root, testing for A = v;
      Examples_v := subset of Examples with A = v;
      if Examples_v is empty
        below the new branch add a leaf with label = most
          common value of Target_attribute in Examples;
      else
        below the bew branch add subtree
          ID3(Examples_v,Target_attribute,Attributes - {A});
  return Root;
```


ID3: Example

Comparing Root nodes for the weather data set:



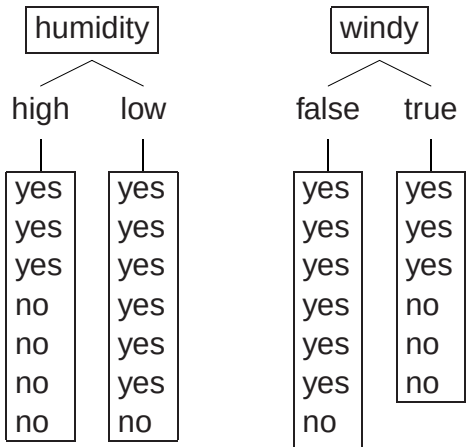
$\text{Gain}(S, \text{outlook}) = 0.25$



$\text{Gain}(S, \text{temperature}) = 0.03$

ID3: Example

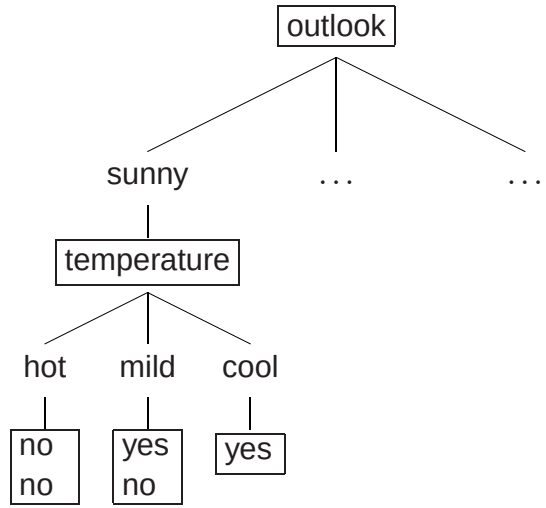
Comparing Root nodes for the weather data set:



$\text{Gain}(S, \text{humidity}) = 0.15$
$\text{Gain}(S, \text{windy}) = 0.05$

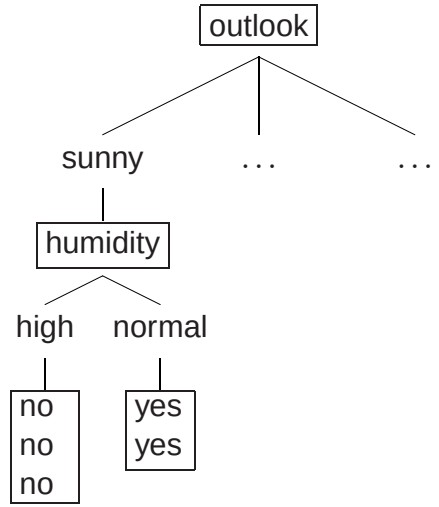
outlook achieves highest information gain $\Rightarrow$ should be used as root for decision tree

ID3: Example



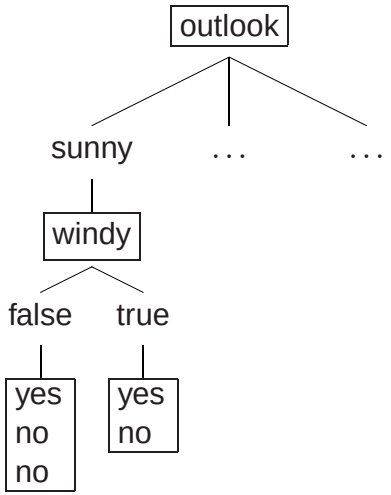
$\text{Gain}(S, \text{temperature}) = 0.57$

ID3: Example



$\text{Gain}(S, \text{humidity}) = 0.97$

ID3: Example

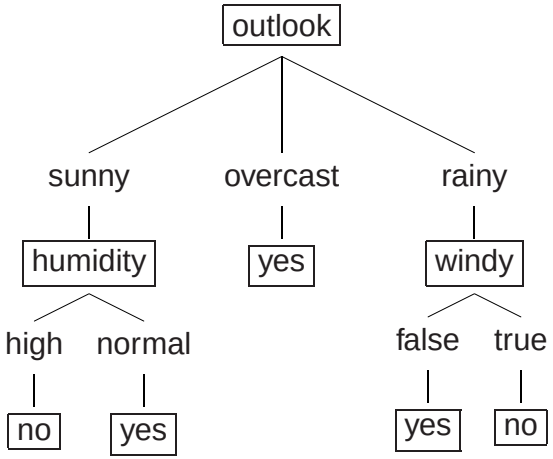


$\text{Gain}(S, \text{windy}) = 0.02$

humidity achieves highest information gain $\Rightarrow$ should be chosen for outlook = sunny

ID3: Example

Final decision tree for the weather data:



Note: all leaf nodes are associated with training examples from the same class (entropy = 0).

Note: the attribute temperature is not used.

Search Strategy in ID3

- *Complete hypothesis space:* any finite discrete-valued function can be expressed.
- *Incomplete search:* searches incompletely through the hypothesis space until the tree is consistent with the data.
- *Single hypothesis:* only one current hypothesis (the best one) is maintained.
- *No backtracking:* once an attribute is selected, this can't be changed. Problem: might not find global maximum.
- *Full training set:* attributes are selection by computing information gain on the full training set. Advantage: robust to errors. Problem: not incremental.

Inductive Bias in ID3

Inductive bias (learning bias): strategy used to generalize from seen data (training set) to unseen data (test set).

ID3 performs (approximately) a *top-down, breadth-first, greedy* search through the search space. Its inductive bias is:

- Shorter trees are preferred over longer trees.
- Trees that place high information gain attributes close to the root are preferred over those that don't.

The inductive bias of ID3 follows from its *search bias*, not from a *language bias* (it can represent any target function expressed as disjunction of conjunctions).

Why Prefer Short Hypotheses?

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

This seems to be an inductive bias that reflects a key strategy in scientific inquiry.

Justification: there are fewer short hypotheses than long ones. Hence we are less likely to find one that fits the data by chance.

Problems:

- There are many small sets of hypotheses that once can define (e.g., all tree with exactly ten nodes).
- The size of a hypothesis is relative to the representation that the learner uses.

Solution: apply Occam's razor to the representation language.

Avoiding Overfitting

ID3 tries to grow the tree until it *perfectly* fits the training data.

Problematic if there is noise (erroneous instances) in the training set, or when the training set is too small. ID3 *overfits*, i.e., fails to generalize to unseen data.

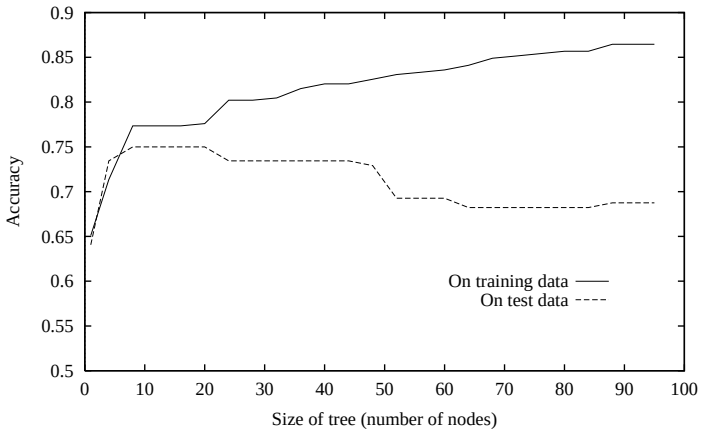
Example: add the an incorrect instance to the weather data:

outlook	temp.	humidity	windy	play
sunny	hot	normal	true	no

ID3 will create a new node (under humidity = normal) to accommodate this instance. This will increase the error rate on the test data.

Avoiding Overfitting

Example: learning which patients have a form of diabetes (Mitchell 1997: p. 67).



Avoiding Overfitting

Two approaches to avoiding overfitting:

- Stop growing the tree before it perfectly classifies the training data. Problem: to know when to stop.
- Allow the tree to overfit the data, and then post-prune it.

The *post-pruning* approach is more common. It requires to split the data into a training set and a *validation set*, which is used for pruning.

Rationale: validation set is unlikely to exhibit the same errors and random fluctuations as the training set.

Often a split 2/3 training and 1/3 validation is used.

Reduced-error Pruning

Informal algorithm:

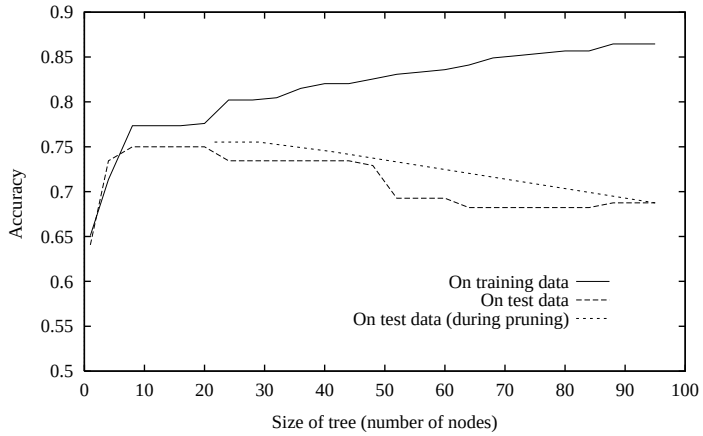
- Pruning a node: remove subtree rooted at the node, make it a leaf node, assign it the most common classification.
- Remove a node only if it doesn't decrease performance on the validation set.
- Prune interactively, start with the node that leads to the most increase in performance.

Drawback: data has to be split in three set (training, validation, testing); requires a large dataset.

Many other approaches for pruning exist, some don't require a validation set (e.g., rule post-pruning).

Reduced-error Pruning

Example: diabetes data with reduced-error pruning (Mitchell 1997: p. 70).



Summary

- Decision tree induction is a method for learning concepts defined by attributes with discrete values.
- ID3 infers a decision tree by growing it top-down, breadth-first, greedily based on information gain.
- the inductive bias of ID3 is for shorter tree with high-information gain attributes close to the root.
- Overfitting is a problems for decision tree induction; can be caused by noise in the training data, or by small training sets.
- A number of techniques exist for post-pruning the tree to avoid overfitting, an example is reduced-error pruning.

References

Mitchell, Tom. M. 1997. *Machine Learning*. New York: McGraw-Hill.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1(1): 81–106.

Witten, Ian H., and Eibe Frank. 2000. *Data Mining: Practical Machine Learing Tools and Techniques with Java Implementations*. San Diego, CA: Morgan Kaufmann.