Lecture 12
Decision Trees

Marco Chiarandini

Deptartment of Mathematics & Computer Science
University of Southern Denmark

Slides by Stuart Russell and Peter Norvig

Course Overview

- ✔ Introduction
  - ✔ Artificial Intelligence
  - ✔ Intelligent Agents
- ✔ Search
  - ✔ Uninformed Search
  - ✔ Heuristic Search
- ✔ Adversarial Search
  - ✔ Minimax search
  - ✔ Alpha-beta pruning
- ✔ Knowledge representation and Reasoning
  - ✔ Propositional logic
  - ✔ First order logic
  - ✔ Inference
- ✔ Uncertain knowledge and Reasoning
  - ✔ Probability and Bayesian approach
  - ✔ Bayesian Networks
  - ✔ Hidden Markov Chains
  - ✔ Kalman Filters
- Learning
  - Decision Trees
  - Maximum Likelihood
  - EM Algorithm
  - Learning Bayesian Networks
  - Neural Networks
  - Support vector machines

Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- Learning method depends on type of performance element, available feedback, type of component to be improved, and its representation
- For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples

Inductive learning

Simplest form: learn a function from examples
f is the target function

An example is a pair x, $f(x)$, e.g., $\begin{array}{c|c|c} O & O & X \\ \hline & X & \\ \hline X & & \end{array}, \quad +1$

Problem: find a(n) hypothesis h
such that $h \approx f$
given a training set of examples

Inductive learning

Simplest form: learn a function from examples
f is the target function

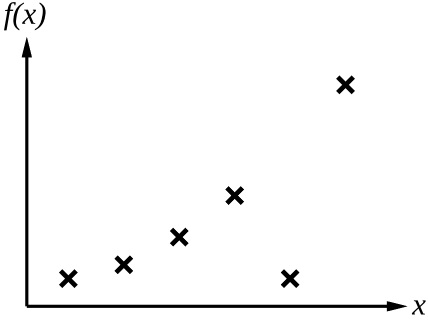
An example is a pair x, $f(x)$, e.g., $\begin{array}{c|c|c} O & O & X \\ \hline & X & \\ \hline X & & \end{array}$ $, \quad +1$

Problem: find a(n) hypothesis h
such that $h \approx f$
given a training set of examples

(**This is a highly simplified model of real learning:**
- **Ignores prior knowledge**
- **Assumes a deterministic, observable "environment"**
- **Assumes examples are given**
- **Assumes that the agent wants to learn f—why?**)

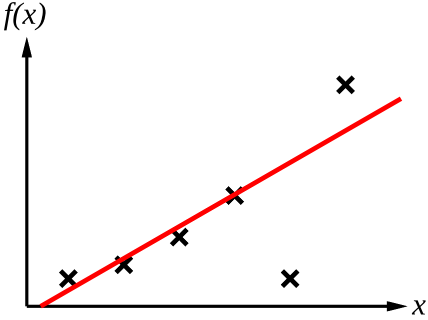
Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)
E.g., curve fitting:



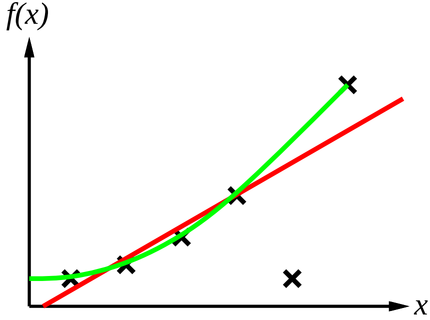
Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)
E.g., curve fitting:



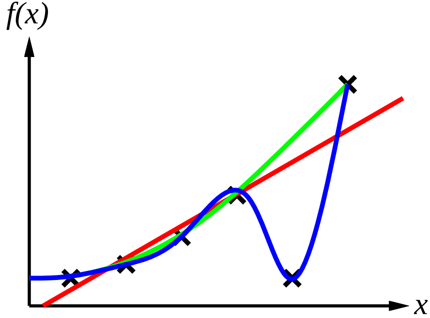
Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)
E.g., curve fitting:



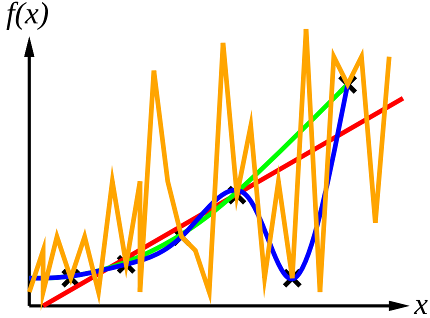
Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)
E.g., curve fitting:



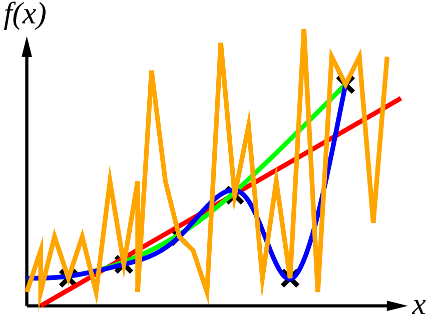
Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)
E.g., curve fitting:



Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)
E.g., curve fitting:



Ockham's razor: maximize a combination of consistency and simplicity

Attribute-based representations

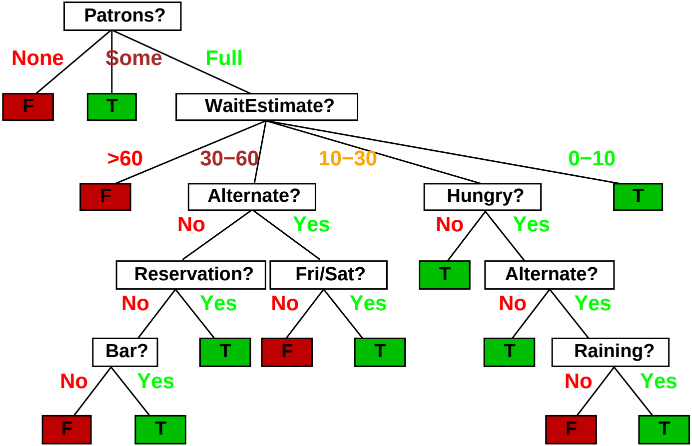
Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	>60	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	>60	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Classification of examples is positive (T) or negative (F)

Decision trees

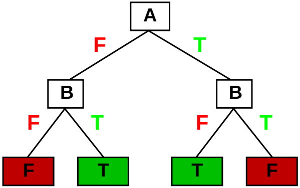
One possible representation for hypotheses
E.g., here is the “true” tree for deciding whether to wait:



Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row $\longrightarrow$ path to leaf:

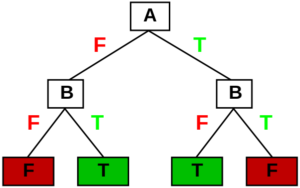
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

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



Trivially, there is a consistent decision tree for any training set
w/ one path to leaf for each example (unless f nondeterministic in x)
but it probably won't generalize to new examples
Prefer to find more **compact** decision trees

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??
$=$ number of Boolean functions
$=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., $Hungry \land \neg Rain$)??

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)??

Each attribute can be in (positive), in (negative), or out

$\implies$ 3^n distinct conjunctive hypotheses

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., $Hungry \land \neg Rain$)??

Each attribute can be in (positive), in (negative), or out

$\implies$ 3^n distinct conjunctive hypotheses

More expressive hypothesis space

– increases chance that target function can be expressed

– increases number of hypotheses consistent w/ training set

$\implies$ may get worse predictions

Decision tree learning

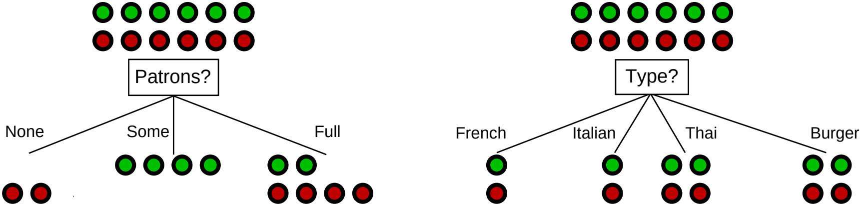
Aim: find a small tree consistent with the training examples
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

function DTL(*examples, attributes, default*) **returns** a decision tree

if *examples* is empty **then return** *default*
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** Mode(*examples*)
else
best $\leftarrow$ Choose-Attribute(*attributes*, *examples*)
tree $\leftarrow$ a new decision tree with root test *best*
for each value v_i of *best* **do**
$examples_i \leftarrow \{$elements of *examples* with $best = v_i\}$
subtree $\leftarrow$ DTL($examples_i$, *attributes* $-$ *best*, Mode(*examples*))
add a branch to *tree* with label v_i and subtree *subtree*
return *tree*

Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice—gives **information** about the classification

Information

Information answers questions
The more clueless I am about the answer initially, the more information is contained in the answer
Scale: 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$
Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} -P_i \log_2 P_i$$

(also called entropy of the prior)

Information contd.

- Suppose we have p positive and n negative examples at the root
  $\implies H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example
  information of the table
  E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

Information contd.

- Suppose we have p positive and n negative examples at the root
  $\implies H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example
  information of the table
  E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

- An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification

Information contd.

- Suppose we have p positive and n negative examples at the root
  $\implies H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example
  information of the table
  E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

- An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification

- Let E_i have p_i positive and n_i negative examples
  $\implies H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example
  $\implies$ **expected** number of bits per example over all branches is

$$\sum_i \frac{p_i+n_i}{p+n} H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

Information contd.

- Suppose we have p positive and n negative examples at the root
  $\implies H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example
  information of the table
  E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

- An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification

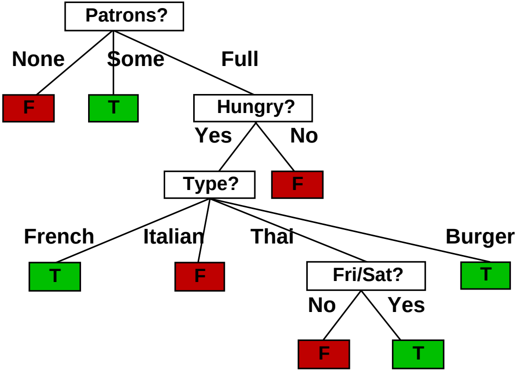
- Let E_i have p_i positive and n_i negative examples
  $\implies H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example
  $\implies$ **expected** number of bits per example over all branches is

$$\sum_i \frac{p_i + n_i}{p+n} H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

- For *Patrons?*, this is 0.459 bits, for *Type* this is (still) 1 bit
  $\implies$ choose the attribute that minimizes the remaining information needed

Example contd.

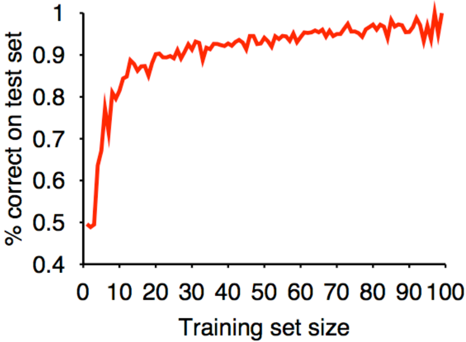
Decision tree learned from the 12 examples:



Substantially simpler than "true" tree—a more complex hypothesis isn't justified by small amount of data

Performance measurement

How do we know that $h \approx f$? (Hume's **Problem of Induction**)
1) Use theorems of computational/statistical learning theory
2) Try h on a new test set of examples
   (use **same distribution over example space** as training set)
Learning curve = % correct on test set as a function of training set size

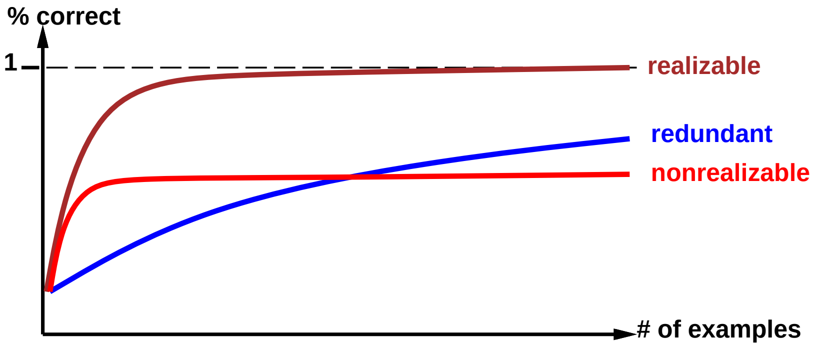


Performance measurement contd.

Learning curve depends on

– realizable (can express target function) vs. non-realizable
  non-realizability can be due to missing attributes
  or restricted hypothesis class (e.g., thresholded linear function)

– redundant expressiveness (e.g., loads of irrelevant attributes)



Decision Tree Types

- Classification tree analysis is when the predicted outcome is the class to which the data belongs. Iterative Dichotomiser 3 (ID3), C4.5, (Quinlan, 1986)
- Regression tree analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).
- Classification And Regression Tree (CART) analysis is used to refer to both of the above procedures, first introduced by (Breiman et al., 1984)
- CHi-squared Automatic Interaction Detector (CHAID). Performs multi-level splits when computing classification trees. (Kass, G. V. 1980).
- A Random Forest classifier uses a number of decision trees, in order to improve the classification rate.
- Boosting Trees can be used for regression-type and classification-type problems.

Used in data mining (most are included in R, see `rpart` and `party` packages, and in Weka, Waikato Environment for Knowledge Analysis)