Knowledge Engineering

Semester 2, 2004-05

Michael Rovatsos
mrovatsso@inf.ed.ac.uk

Lecture 2 – Inductive Learning: Decision Trees
14th January 2005

Where are we?

- Last time ...
  - we defined knowledge, KBS and KE
  - looked at KE process
  - identified important building blocks of KE process.
- Today ...
  - marks the beginning of the "Knowledge Acquisition" (KA) part of the module
  - we will discuss methods for automating KA
  - in particular: **Decision Tree Learning**

Knowledge Acquisition

- Knowledge Acquisition generally considered bottleneck in KE process
- Informal methods:
  - Expert interviews (today developers $\neq$ experts)
  - Analysis of organisational databases and documents
  - Independent analysis of domain knowledge (textbooks, online documents, etc.)
- (Although inevitable) these methods are complex, costly, and inflexible ➡ automation desirable
- Discussion of machine learning methods, in particular: **inductive (symbolic) learning**

Inductive Learning

- Idea: we are provided with examples $(x, f(x))$ where $f(x)$ is the correct value of the **target function** f for input x and we want to learn f
- Task of inductive inference:

  Given a collection of examples of f, return a function h that approximates f
- h is a **hypothesis** taken from a **hypothesis space** H
- (Pure) inductive inference assumes no prior knowledge
- Validation: construct/adjust h using a **training set**, evaluate generalisation capabilities on **test set**

Inductive Learning

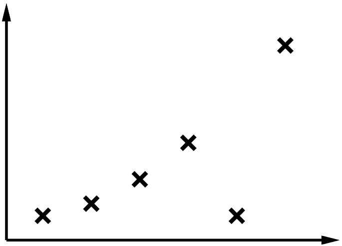
- Inductive learning (IL) is a form of **supervised learning**: information about the output value $f(x)$ of x is explicit
- Art of inductive learning: given a set of training examples, choose the best hypothesis
- h **consistent**: agrees with all example data seen so far (not all learning algorithms return consistent hypotheses)
- H defines the range of functions we can use and determines expressiveness of hypothesis
- Learning problem **realisable** if $f(x) \in H$ (often this is not known in advance)

Choosing Hypotheses

- **Ockham's razor**: prefer the simplest hypothesis consistent with the data
- Why is this a reasonable policy?
  - Intuitively, why choose complex hypothesis if simple one does the job?
  - There exist more long (i.e. more complex) hypotheses than short ones
    ➡ accidental choice of bad hypothesis that is consistent with data is more unlikely if the hypothesis is simple
- Problem: identifying what simple hypotheses are
- Trade-off: the more expressive the hypothesis space, the more examples are needed (and the more the complex learning algorithm)

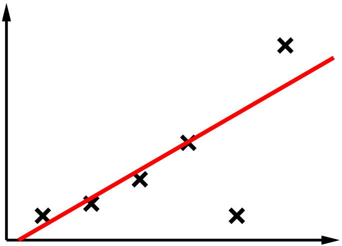
Example

- Curve fitting: consider real numbers x and $f(x)$ as data points (examples)
- Assume H is the set of polynomials, e.g. $5x$, $3x^2 + 2$, $x^5 - 3x^4 + 2$, etc.
- Construct h such that it agrees with f on **training set**



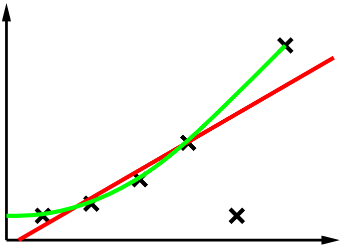
Example

- Curve fitting: consider real numbers x and $f(x)$ as data points (examples)
- Assume H is the set of polynomials, e.g. $5x$, $3x^2 + 2$, $x^5 - 3x^4 + 2$, etc.
- Construct h such that it agrees with f on **training set**



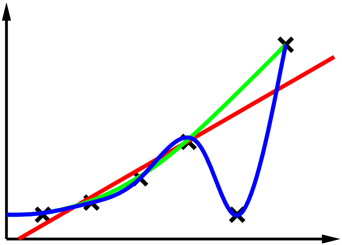
Example

- Curve fitting: consider real numbers x and $f(x)$ as data points (examples)
- Assume H is the set of polynomials, e.g. $5x$, $3x^2 + 2$, $x^5 - 3x^4 + 2$, etc.
- Construct h such that it agrees with f on **training set**



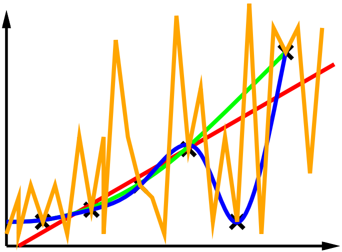
Example

- Curve fitting: consider real numbers x and $f(x)$ as data points (examples)
- Assume H is the set of polynomials, e.g. $5x$, $3x^2 + 2$, $x^5 - 3x^4 + 2$, etc.
- Construct h such that it agrees with f on **training set**



Example

- Curve fitting: consider real numbers x and $f(x)$ as data points (examples)
- Assume H is the set of polynomials, e.g. $5x$, $3x^2 + 2$, $x^5 - 3x^4 + 2$, etc.
- Construct h such that it agrees with f on **training set**



Describing IL Methods

- What kind of information do the examples offer?
  - How much training data is available? All at once?
  - What are their attributes and those attributes' domains (boolean, discrete, continuous) ?
  - What is the range of possible classifications?
  - Do we have to consider **noise** in the data?
- The hypothesis space:
  - Choice of right representation
  - Questions of expressiveness vs. complexity
  - How can the learning result be used after learning?
- Choosing hypotheses:
  - Incremental vs. batch processing of examples
  - Refining an initial hypothesis vs. starting with none
  - What kind of **inductive bias** is applied?

Decision Trees

- Attribute-based **classification** learning:
  - Example input x: situation/object described in terms of attribute values
  - Example output $f(x)$: a discrete-valued classification decision
- Here: Boolean classification, each example is classified as positive (**true**) or negative (**false**)
- Alternatively: f describes an unknown concept, and all values of x for which $f(x) = \text{true}$ describe the instances of this concept
- Hypothesis = a **decision tree** (DT) whose nodes correspond to tests on attribute values to decide whether $f(x)$ is true or false

Example

Assume we are given a set of situations in which a customer will or will not wait in a restaurant (examples), i.e. the **goal predicate** is *WillWait*(x).

	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*

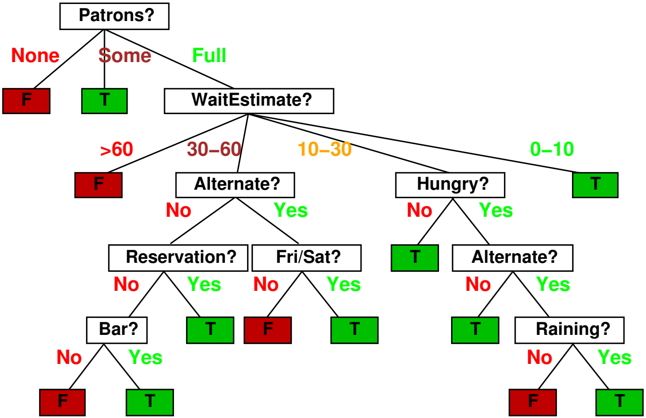
Example

Attributes:

- Alternate: Is there an alternative restaurant nearby?
- Bar: Is there a bar that makes waiting comfortable?
- Fri/Sat: True if current day is Friday or Saturday
- Patrons: None or some people in the restaurant, or is it full?
- Raining: Is it raining outside?
- Reservation: Was a reservation made?
- Estimate: How long is the estimated waiting time?
- . . . and some other (self-explanatory)

Example

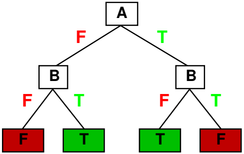
Assume this is the actual decision tree used by the person in question:



Expressiveness

- *What kind of logical constraints can DTs express?*
- Consider conjunction P_i of attribute values on each path leading to "Yes" and disjunction $G = P_1 \vee \ldots P_n$ over these conjunctions
  - ➡ DTs can represent any formula of propositional logic
- Example: Each truth table row corresponds to one path

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



- Easy to build a tree that is consistent with all examples, but will it be able to generalise?

Decision Tree Learning Algorithm

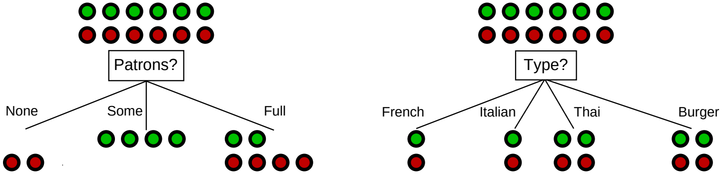
- Iteratively build a tree by selecting the "best" attribute and adding descendant nodes for all its values
- If all examples on some branch have the same classification, then no more decision steps are necessary (add leaf node with this classification)
- If some examples are positive and some negative, choose a new attribute to discriminate between them
- If we run out of attributes, examples have same description but different classification (noise)
  ➡ use majority vote as a workaround
- If we run out of examples then no data is available for current attribute value; use majority value of parent node

The Algorithm

```
DECISION-TREE-LEARNING(examples, attribs, default)
inputs  : examples, a set of examples , attribs, a set of attributes
          default, default value for the goal predicate
if examples is empty then return default
else if all examples have same classification
        then return this classification
else if attribs is empty then return MAJORITY-VALUE(examples)
else
   best ← CHOOSE-ATTRIBUTE(attribs, examples)
   tree ← a new decision tree with root test best
   m ← MAJORITY-VALUE(examples)
   for each value v_i of best do
      examples_i ← { elements of examples with best = v_i }
      subtree ← DECISION-TREE-LEARNING(examples_i, attribs − best, m)
      add a branch to tree with label v_i and subtree subtree
   return tree
```

Attribute Selection Heuristics

- Best way to obtain compact decision tree: find attributes that split example set into positive/negative examples
- Example:



Entropy-Based Measures

- **Information-theoretic entropy** can be used as a measure for amount of information
- If $v_1, \ldots v_n$ attribute values with probabilities $P(v_i)$, information content

$$I(P(v_1), \ldots P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i)$$

- For example: I(0.5,0.5)=1 (bit), I(0.01,0.99)=0.08 (bits)
- Assume we have p positive and n negative examples
  ➡ classifying a given example correctly requires $I(\frac{p}{p+n}, \frac{n}{p+n})$ bits of information

Information Gain

- Attribute A splits example set into n subsets E_i containing p_i positive and n_i negative examples
- How much information do we still need after this test?
- Assumption: an example has value v_i for the attribute in question with probability $\frac{p_i+n_i}{p+n}$
  ➡ measure for remaining "information-to-go":

$$Remainder(A) = \sum_{i=1}^{n} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- $Gain(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - Remainder(A)$ provides a measure for the **information gain** provided by A
- Heuristics: choose A that maximises $Gain(A)$

Overfitting

- Problem: If hypothesis space is large enough, there is a probability of finding “meaningless” regularities
- Example: Date of birth data as a predictor for getting an MSc in Informatics
- If the hypothesis “overfits” the learning data, it may be consistent with examples but useless for generalisation purposes
- A general problem of all learning algorithms
- One way of dealing with overfitting: **decision tree pruning** (e.g. use significance tests to determine irrelevance of attributes)

Validation

Typical validation for inductive learning methods:

- Split example data into training set and test set
- Train system with example data
- Evaluate prediction accuracy on test set
- Optionally: use **cross-validation** to prevent overfitting
  - Set a portion (e.g. $1/k$ of the data) aside
  - Conduct k experiments using the "left out" examples as test set (and remaining data as training set)
  - Average performance over k runs

Critique

- Many functions not easy to represent with DTs (e.g. majority function or mathematical functions)
- Best for problems with limited number of attributes and attribute values
- Assumes examples are unambiguously and completely (no missing data) described/classified (deterministic and fully observable environment)
- No use of prior knowledge ➡ learning can be very slow
- Is DTL an (1) an incremental and/or (2) an anytime algorithm?
- Is this an adequate model of real learning?

Summary

- Inductive Learning: Inference of knowledge from examples
- Decision Trees: A simple yet effective method for attribute-based inductive inference
- Expressiveness vs. complexity, Ockham's Razor
- Entropy-based heuristics for attribute selection
- Problems of noise and overfitting
- Next lecture: **Version space learning**