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 ≠ 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 Diagram](image-url)
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) = 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:

<table>
<thead>
<tr>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
</tr>
<tr>
<td>X2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
</tr>
<tr>
<td>X3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
</tr>
<tr>
<td>X4</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
</tr>
<tr>
<td>X5</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$$</td>
<td>$</td>
<td>T</td>
<td>French</td>
</tr>
<tr>
<td>X6</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
</tr>
<tr>
<td>X7</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
</tr>
<tr>
<td>X8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
</tr>
<tr>
<td>X9</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
</tr>
<tr>
<td>X10</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$</td>
<td>$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
</tr>
<tr>
<td>X11</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
</tr>
<tr>
<td>X12</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
</tr>
</tbody>
</table>

Example

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

```
Patrons?
   None
   Some
   Full

WaitEstimate?
   >60
   30–60
   10–30

Hungry?
   No
   Yes

Reservation?
   No
   Yes

Fri/Sat?
   No
   Yes

Alternate?
   No
   Yes

Bar?
   No
   Yes

Raining?
   No
   Yes
```
Expressiveness

➤ What kind of logical constraints can DTs express?
➤ Consider conjunction \( P_i \) of attribute values on each path leading to “Yes” and disjunction \( G = P_1 \lor \ldots \lor P_n \) over these conjunctions
   ➤ DTs can represent any formula of propositional logic
   ➤ Example: Each truth table row corresponds to one path

\[
\begin{array}{c|c|c|c|c|c|c}
A & B & A \lor B & F & T & \text{F} & \text{T} \\
\hline
\text{F} & \text{F} & \text{F} & \text{F} & \text{F} & \text{F} & \text{F} \\
\text{F} & \text{T} & \text{T} & \text{T} & \text{F} & \text{F} & \text{F} \\
\text{T} & \text{F} & \text{T} & \text{T} & \text{T} & \text{F} & \text{F} \\
\text{T} & \text{T} & \text{T} & \text{T} & \text{T} & \text{F} & \text{F} \\
\end{array}
\]

➤ 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)
1 inputs : examples, a set of examples, attribs, a set of attributes
2 default, default value for the goal predicate
3 if examples is empty then return default
4 else if all examples have same classification
5 then return this classification
6 else if attribs is empty then return MAJORITY-VALUE(examples)
7 else
8 best ← CHOOSE-ATTRIBUTE(attribs, examples)
9 tree ← a new decision tree with root test best
10 m ← MAJORITY-VALUE(examples)
11 for each value \( v_i \) of best do
12 examples_i ← { elements of examples with best = \( v_i \)}
13 subtree ← DECISION-TREE-LEARNING(examples_i, attribs – best, m)
14 add a branch to tree with label \( v_i \) and subtree subtree
15 return tree
```

Attribute Selection Heuristics

➤ Best way to obtain compact decision tree: find attributes that split example set into positive/negative examples
➤ Example:
Entropic-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 \( P(v_i) \), info-

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

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

\[ I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) \]

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?

\[ \text{Assumption: an example has value } v_i \text{ for the attribute in question with probability } \frac{p_i+n_i}{p+n} \]

\[ \Rightarrow \text{measure for remaining "information-to-go":} \]

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

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

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