Knowledge Engineering

Semester 2, 2004-05

Michael Rovatsos
mrovatso@inf.ed.ac.uk

Lecture 3 – Inductive Learning: Version Spaces
18th January 2005

Where are we?

- Last time ...
  - we started talking about Knowledge Acquisition
  - suggested methods for automating it
  - in particular: Decision Tree Learning
- Today ...
  - we will discuss another inductive learning method
  - look at inductive learning with a knowledge representation touch
  - **Version Space Learning**

Knowledge Representation & Learning

- Interfacing between Knowledge Acquisition & Knowledge Representation:
  - Using results from KA in KR systems
  - Using knowledge from the KR system in the KA process (will be discussed in "Knowledge Evolution" part)
- Methods such as decision tree learning cannot be integrated in a KR system directly
- Would like to define learning algorithms that operate on generic representations, e.g. logic

Example

- Recall decision tree learning examples:

	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*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- View e.g. example X_1 as a logical formula: $Alternate(X_1) \wedge \neg Bar(X_1) \wedge \neg Fri/Sat(X_1) \wedge Hungry(X_1) \ldots$
- Call this formula the **description** $D(X_i)$ of X_i

Example: Describing DTL in First-Order Logic

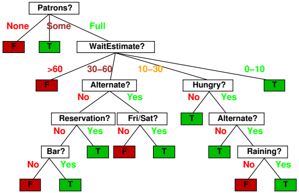
- Classification: $WillWait(X_1)$
- Use generalised notation $Q(X_i)/\neg Q(X_i)$ for classification of positive/negative examples
- Training set = conjunction of all description and classification sentences

$$D(X_1) \wedge Q(X_1) \wedge D(X_2) \wedge \neg Q(X_2) \wedge D(X_3) \wedge Q(X_3) \ldots$$

- Each hypothesis H_i is equivalent to a **candidate definition** $C_i(x)$ such that $\forall x Q(X) \Leftrightarrow C_i(x)$

Example

Recall decision tree from last lecture:



Example

This is equivalent to the disjunction of all branchens that lead to a "true" node (formula for each branch = conjunction of attribute values on branch)

$$\forall r \overbrace{WillWait(r)}^{Q(r)} \Leftrightarrow \overbrace{Patrons(r, Some)}^{C_i(r)}$$
$$\vee \ (Patrons(r,\ Full) \wedge Hungry(r) \wedge Type(r, French))$$
$$\vee \ (Patrons(r,\ Full) \wedge Hungry(r) \wedge Type(r, Thai) \wedge Fri/Sat(r))$$
$$\vee \ (Patrons(r,\ Full) \wedge Hungry(r) \wedge Type(r, Burger))$$

Hypotheses and Hypothesis Spaces

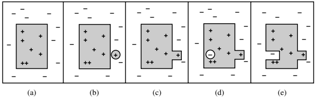
- Set of examples that satisfy a candidate definition = **extension** of the respective hypothesis
- In the learning process, we can rule out hypotheses that are not **consistent** with examples
- Two cases:
  - **False negative**: hypothesis predicts negative outcome but classification of example is positive
  - **False positive**: hypothesis predicts positive outcome but classification of example is negative

Hypotheses and Hypothesis Spaces

- Learning algorithm believes that one of its hypotheses is true, i.e. $H_1 \vee H_2 \vee H_3 \vee \ldots$
- Each false positive/false negative could be used to rule out inconsistent hypotheses from the hyp. space
  ➡ general model of inductive learning
- But not practicable if hyp. space is vast, e.g. all formulae of first-order logic
- Have to look for simpler methods:
  - Current-best hypothesis search
  - Version space learning

Current-Best Hypothesis Search

- Idea very simple: adjust hypothesis to maintain consistency with examples
- Uses **specialisation**/**generalisation** of current hypothesis to exclude false positives/include false negatives



- Assumes "more general than" and "more specific than" relations to search hypothesis space efficiently

Current-Best Hypothesis Search

CURRENT-BEST-LEARNING(*examples*)
$H \leftarrow$ any hypothesis consistent with the first example in *examples*
for each remaining example e in *examples* **do**
if e is a false positive for H **then**
  $H \leftarrow$ choose a specialisation of H consistent with *examples*
else if e is a false negative for H **then**
  $H \leftarrow$ choose a generalisation of H consistent with *examples*
if no consistent specialisation/generalisation can be found **then fail**
return H

Things to note:

- Non-deterministic choice of specialisation/generalisation
- Does not provide rules for spec./gen.
- One possibility: add/drop conditions

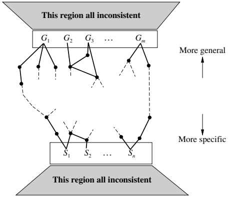
Version Space Learning

- Problems of current-best learning:
  - Have to check all examples again after each modification
  - Involves great deal of backtracking
- Alternative: maintain set of all hypotheses consistent with examples
- **Version space** = set of remaining hypotheses
- Algorithm:
  VERSION-SPACE-LEARNING(*examples*)
  $V \leftarrow$ set of all hypotheses
  for each example e in *examples* **do**
   if V is not empty
    then $V \leftarrow \{h \in V : h \text{ is consistent with } e\}$
  return V

Version Space Learning

- Advantages:
  - incremental approach (don't have to consider old examples again)
  - **least-commitment** algorithm
- Problem: How to write down disjunction of all hypotheses?
  ➡ think of interval notation $[1, 2]$
- Exploit ordering on hypotheses and boundary sets
  - **G-set** most general boundary (no more general hypotheses are consistent with all examples)
  - **S-set** most specific boundary (no more specific hypotheses are consistent with all examples)

Version Space Learning

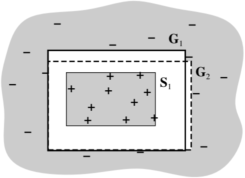


Version Space Learning

- Everything between G and S (version space) is consistent with examples and represented by boundary sets
- Initially: $G = \{True\}$, $S = \{False\}$
- How to prove that this is a reasonable representation?
- Need to show two properties:
  - Every consistent H not in the boundary sets is more specific than some G_i and more general than some S_j (follows from definition)
  - Every H more specific than some G_i and more general than some S_j is consistent.
    Any such H rejects all negative examples rejected by each member of G and accepts all positive examples accepted by any member of S ➡ H consistent

Version space learning

There are no known examples "between" S and G, i.e. outside S but inside G:



Updating the Version Space

- Final issue: how to update the version space?
- Assume S_i and G_i members of S-/G-sets.
  Each example can be a false positive (FP)/false negative (FN) for each of them:
  1. FP for S_i ➡ S_i too general ➡ throw S_i out (no consistent specialisations of S_i exist by definition)
  2. FN for S_i ➡ S_i too specific ➡ replace it by all its immediate generalisations
  3. FP for G_i ➡ G_i too general ➡ replace it by all its immediate specilisations
  4. FN for G_i ➡ G_i too specific ➡ throw S_i out (no consistent generalisations of G_i exist by definition)

Remarks/Problems

- After termination of the algorithm:
  - Only one concept left ➡ unique hypothesis or
  - S/G becomes empty ➡ version space collapses (no consistent hypothesis exists) or
  - we run out of examples with several hypotheses remaining ➡ use disjunction or e.g. majority vote
- Drawbacks of version space learning:
  - Noise/insufficient attributes ➡ VS collapses
  - Allowing unlimited disjunction ➡ G will always contain disjunction of negation of examples, S will contain disjunction of positive examples (but use **generalisation hierarchy**)
  - Number of elements in S and G may grow exponentially

Summary

- How to deal with knowledge-based representations of inductive learning?
- Described DTL in terms of logic
- Introduced current-best learning (problems: backtracking, non-incremental)
- Version spaces as an incremental method of inductive learning
- Next time: **Knowledge Representation & Reasoning**

Announcements

- There will be no lecture on the 28th January! (Friday next week)
- Prepared a preliminary listing of all necessary AIMA chapters for those who want to copy them
- Paper copies of previous KE notes available from the ITO (if "4up" format is too small to read)