Introduction to Inductive Logic Programming Lectures 1 and 2

Stephen Muggleton
Department of Computing
Imperial College, London

14th March, 2011

Overview

Lecture 1	Generalisation
Lecture 2	Refinement and Inverting Entailment

Suggested reading

- S-H. Nienhuys-Cheng and R. de Wolf, "Foundations of Inductive Logic Programming", Springer-Verlag, 1997.
- L. De Raedt, P. Frasconi, K. Kersting, and S.H. Muggleton. "Probabilistic Inductive Logic Programming." Springer-Verlag, 2008.

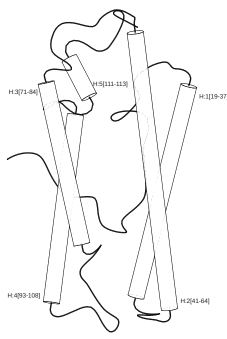
Machine learning

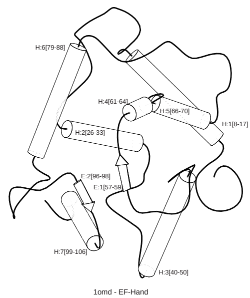
Machine Learning is the study of computer programs that improve automatically through experience.

Tom Mitchell, "Machine Learning", 1997.

Logical	Probabilistic	Mixed
Decision trees	Neural nets	Bayes' nets
Grammars	HMMs	SCFGs
Logic Programs	POMDPs	SLPs

Inducing a model
“4-helical up-and-down bundle”

<table>
<tr><th>Positive(12)</th><th>Negative(12)</th></tr>
<tr><td>


</td><td>


</td></tr>
</table>

The protein P has fold class “Four-helical up-and-down bundle” if it contains a long helix H1 at a secondary structure position between 1 and 3, and H1 is followed by a second helix H2.
[JMB, 2001; MLJ, 2001]

Inductive Logic Programming

Background knowledge. Protein sequence, partial grammar, domain constraints.

Examples. Molecules, annotated sentences.

Hypothesis. Explanation of molecular 3-D shape, new clauses in a grammar.

What is generalisation?

Statement A	Daffy Duck can fly
Statement B	All ducks can fly
Statement C	Marek lives in London
Statement D	Marek lives in England

Terms, atoms and literals

Function symbols	eg. f, g
Predicate symbols	eg. p, q
Constants	eg. c, d
Variables	eg. x, y, z
Terms	eg. $c, 3, x, f(c, g(x))$
Atoms	eg. $\forall x, y.p(x, f(3, y))$
	or $p(x, f(3, y))$
Literals	eg. $\neg p(x, f(3, y)), q(z, d)$

Clauses and Clausal Theories

Clause - disjunction of literals	eg. $l_1 \vee \ldots \vee l_m$
	or $\{l_1, \ldots, l_m\}$
Definite Clause - one positive literal	eg. $a_0 \vee \neg a_1 \ldots \vee \neg a_m$
	or $a_0, \leftarrow a_1, \ldots, a_m$
Theory - conjunction of clauses	eg. $C_1 \wedge \ldots \wedge C_n$
	or $\{C_1, \ldots, C_n\}$
Logic Program - conjunction of	eg. $C_1 \wedge \ldots \wedge C_n$
Horn clauses	or $\{C_1, \ldots, C_n\}$

Simple generalisation
Atom and Clause Subsumption

Given a substitution $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ and formula F. $F\theta$ is formed by replacing every variable v_i in F by t_i.

Atom A subsumes atom B, $A \succeq B$, iff there exists a substitution θ such that $A\theta = B$.

Clause C subsumes clause D, $C \succeq D$, iff there exists a substitution θ such that $C\theta \subseteq D$.

Generalisation example revisited

Daffy Duck can fly	$can\_fly(\text{daffy})$
All ducks can fly	$can\_fly(x)$

$$can\_fly(x) \succeq can\_fly(\text{daffy})$$

$$\theta = \{x/\text{daffy}\}$$

Least general generalisation (lgg) [Plotkin/Reynolds]

Atom A' is a common generalisation of atoms A and B iff $A' \succeq A$ and $A' \succeq B$.

A' is a least general generalisation of atoms A and B iff all common generalisations of A and B subsume A'.

Atoms A and B are compatible iff they have the same predicate symbol and sign.

lgg example

A	B	lgg(A,B)
can_fly(daffy)	can_fly(donald)	$can\_fly(x)$
conn(n1,n1)	conn(n2,n2)	conn(x,x)

Generalisation - harder example

C	Marek lives in London	lives(marek,london)
D	Marek lives in England	lives(marek,england)

Background knowledge

lives(x,england) $\leftarrow$ lives(x,london)

Interpretations

Ground formula	Formula containing no variables
Herbrand Universe, U	Set of all ground terms constructed from given predicate, function symbols and constants
Herbrand Base	Set of all ground atoms over Herbrand Universe
Interpretation	Subset of Herbrand Base Atoms assigned True

Models and Entailment

Interpretation M is a model of formula (atom/literal/clause/theory) F iff F evaluates to True given M. To evaluate F for each variable x replace $\forall x.P(x)$ throughout by $\bigwedge_{t \in U} P(t)$ and then each atom in M by True and apply logical connective truth tables throughout.

$$F \models G \text{ iff every model of } F \text{ is a model of } G$$

Generalisation as entailment

Entailment

C more general than D iff $C \models D$

Relative Entailment

C more general than D wrt B iff $B, C \models D$

ILP general logical setting

B	Background Knowledge - Logic Program
E	Examples - Set of ground unit clauses
H	Hypothesis - Logic Program

Given B, E find H such that

$$B, H \models E$$

Search and refinement

Given B, E find H such that

$$B, H \models E$$

Q : Algorithmically how do we find H given B, E?

A : Search space of clauses from simple to complex (general to specific) or complex to simple (specific to general). This process is called Clause Refinement .

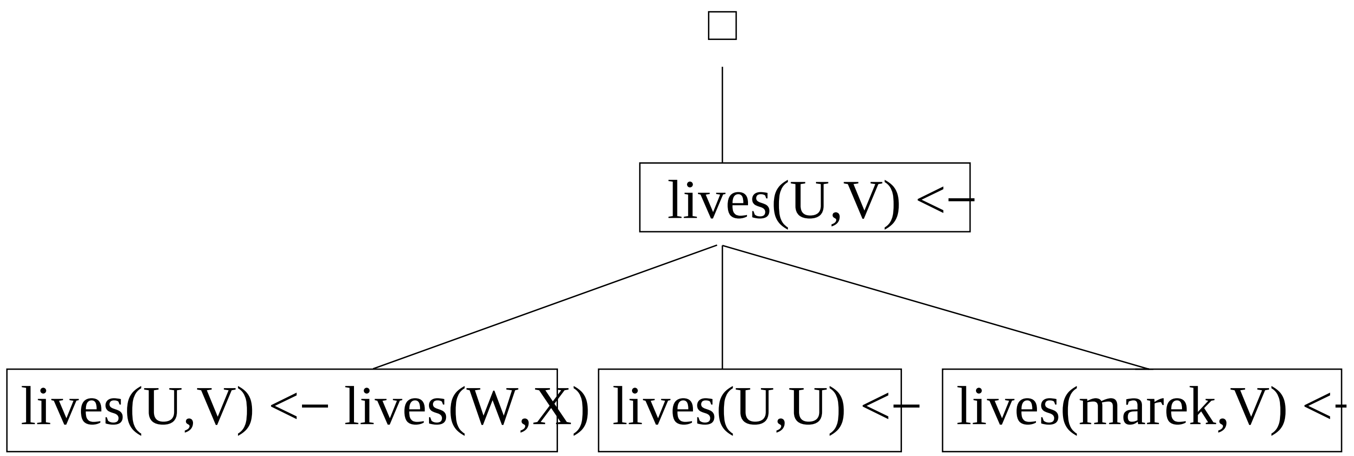
Refinement operator [Shapiro] ρ

Predicate symbols P, Function symbols F

Clause $D \in \rho(C)$ iff one of the following.

	C		
1.	$\{l_1, \ldots, l_n\}$	$C \cup \{p(v_1, \ldots, v_m)\}$	$p_m \in P$
2.	$\{l_1, \ldots, l_i, \ldots, l_n\}$	$\{l_1, \ldots, l'_i, \ldots, l_n\}$	$l'_i = l_i\{u/v\}$
3.	$\{l_1, \ldots, l_i, \ldots, l_n\}$	$\{l_1, \ldots, l'_i, \ldots, l_n\}$	$l'_i = l_i\{u/t\}$ $t = f(v_1, \ldots, v_m)$ $f_m \in F$

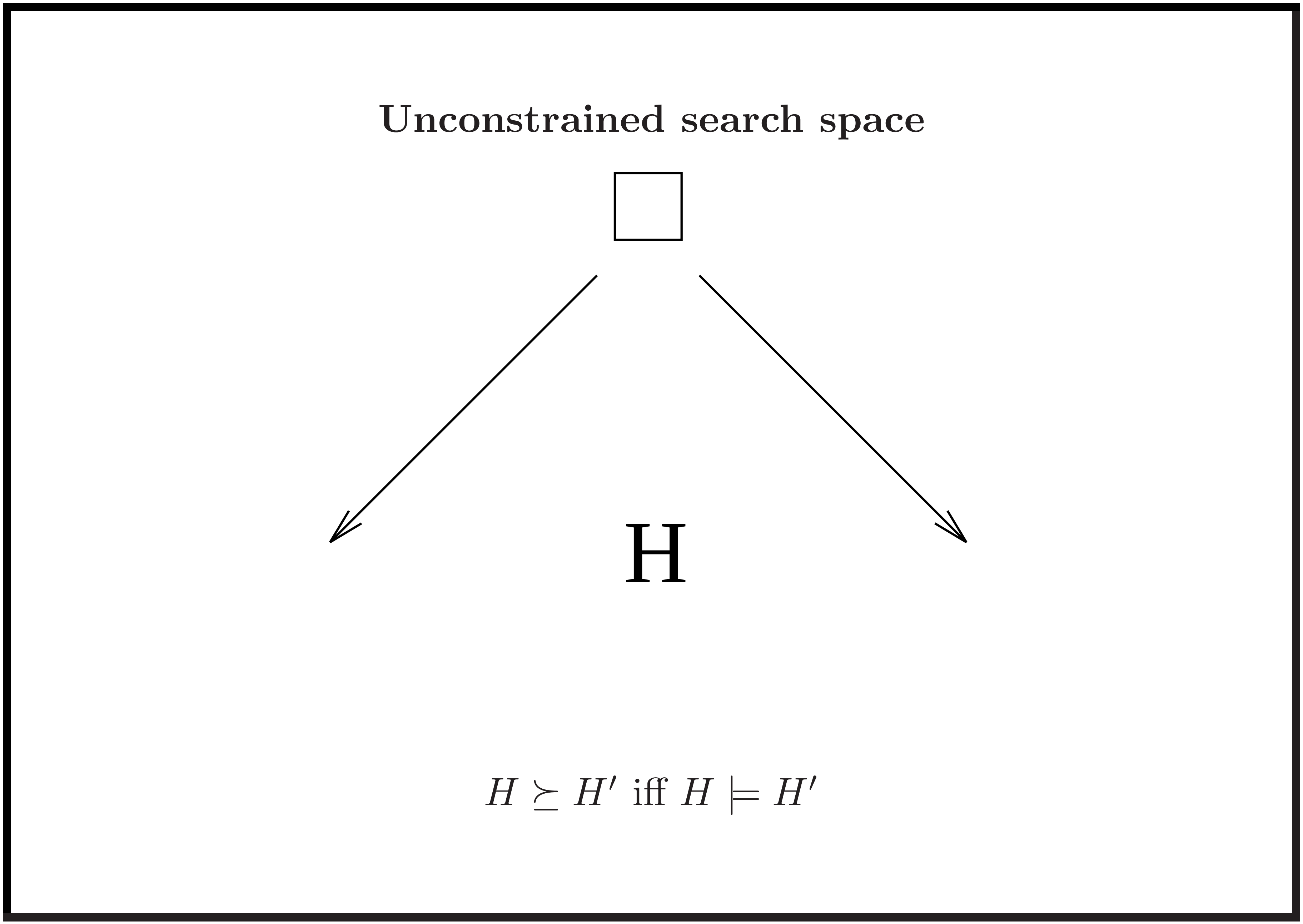
Refinement graph ρ



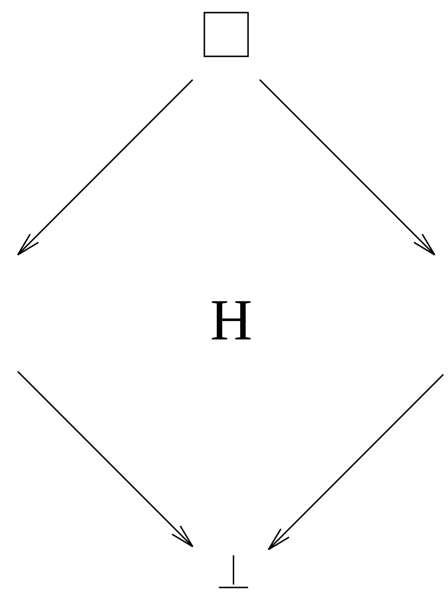
Refinement operator properties [Nienhuys-Cheng]

Theorem. There is no refinement operator ρ which has all the following properties.

Finite	$\forall C.\ \lvert\rho(C)\rvert$ is finite
Proper	$\forall C, D.\ D \in \rho(C)$ only if $C \succ D$
Complete	$\forall C \exists n.\ C \in \rho^n(\Box)$

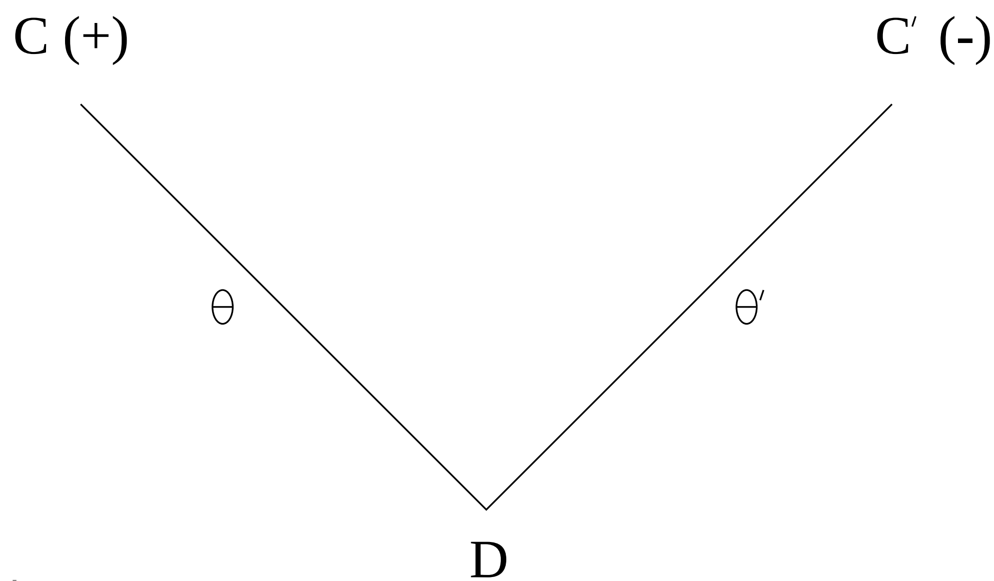

Unconstrained search space
H
$H \succeq H'$ iff $H \models H'$

Constrained search space



$$C \succeq D \text{ iff } \exists\theta . C\theta \subseteq D$$

Inverse resolution (first-order)



Inverting entailment

$$B \wedge H \models E$$

$$B \wedge \overline{E} \models \overline{H}$$

$$B \wedge \overline{E} \models \overline{\bot} \models \overline{H}$$

$$H \models \bot$$

Inverting Entailment Examples (1)

B	**E**	$\perp$
anim(X)← pet(X). pet(X)← dog(X).	nice(X)← dog(X).	nice(X) ← dog(X), pet(X), anim(X).

hasbeak(X)←
bird(X).
bird(X)←
vulture(X).
	hasbeak(tweety).	hasbeak(tweety);
		bird(tweety);
		vulture(tweety).

Inverting Entailment Examples (2)

B	**E**	$\perp$
white(swan1).	$\leftarrow$ black(swan1).	$\leftarrow$ black(swan1), white(swan1).
sentence([],[]).	sentence([a,a,a],[]).	sentence([a,a,a],[]) $\leftarrow$ sentence([],[]).