# Lecture 27

## Inductive Logic Programming (ILP)

Thursday, December 2, 1999

William H. Hsu
Department of Computing and Information Sciences, KSU
http://www.cis.ksu.edu/~bhsu

Readings:
Sections 10.6-10.8, Mitchell
Section 21.4, Russell and Norvig



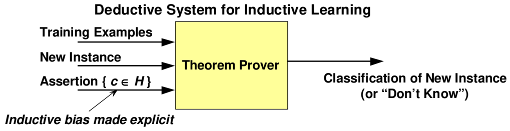
# Lecture Outline

- Readings: Sections 10.6-10.8, Mitchell; Section 21.4, Russell and Norvig
- Suggested Exercises: 10.5, Mitchell
- Induction as Inverse of Deduction
  - Problem of inductive learning revisited
  - Operators for automated deductive inference
    - Resolution rule for deduction
    - First-order predicate calculus (FOPC) and resolution theorem proving
  - Inverting resolution
    - Propositional case
    - First-order case
- Inductive Logic Programming (ILP)
  - Cigol
  - Progol



# Induction as Inverted Deduction: Design Principles

Deductive System for Inductive Learning




- Recall: Definition of Induction
  - Induction: finding $h$ such that $\forall \langle x_i, f(x_i)\rangle \in D \,.\, (B \wedge D \wedge x_i) \vdash f(x_i)$
    - $A \vdash B$ means *A logically entails B*
    - $x_i \equiv i$th target instance
    - $f(x_i)$ is the target function value for example $x_i$ (data set $D = \{\langle x_i, f(x_i)\rangle\}$)
    - Background knowledge $B$ (e.g., *inductive bias* in inductive learning)
- Idea
  - Design inductive algorithm by *inverting operators for automated deduction*
  - Same deductive operators as used in theorem proving



# Induction as Inverted Deduction: Example

- Deductive Query
  - "Pairs $\langle u, v\rangle$ of people such that $u$ is a child of $v$"
  - Relations (predicates)
    - *Child* (target predicate)
    - *Father, Mother, Parent, Male, Female*
- Learning Problem
  - Formulation
    - Concept learning: target function $f$ is Boolean-valued
    - i.e., target predicate
  - Components
    - Target function $f(x_i)$: *Child* (*Bob*, *Sharon*)
    - $x_i$: *Male* (*Bob*), *Female* (*Sharon*), *Father* (*Sharon*, *Bob*)
    - $B$: {*Parent* ($x$, $y$) $\leftarrow$ *Father* ($x$, $y$). *Parent* ($x$, $y$) $\leftarrow$ *Mother* ($x$, $y$).}
  - What satisfies $\forall \langle x_i, f(x_i)\rangle \in D \,.\, (B \wedge D \wedge x_i) \vdash f(x_i)$?
    - $h_1$: *Child* ($u$, $v$) $\leftarrow$ *Father* ($v$, $u$). - doesn't use $B$
    - $h_2$: *Child* ($u$, $v$) $\leftarrow$ *Parent* ($v$, $u$). - uses $B$



# Perspectives on Learning and Inference

- Jevons (1874)
  - First published insight that induction can be interpreted as inverted deduction
  - "Induction is, in fact, the inverse operation of deduction, and cannot be conceived to exist without the corresponding operation, so that the question of relative importance cannot arise. Who thinks of asking whether addition or subtraction is the more important process in arithmetic? But at the same time much difference in difficulty may exist between a direct and inverse operation; … it must be allowed that inductive investigations are of a far higher degree of difficulty and complexity that any questions of deduction…"
- Aristotle (*circa* 330 B.C.)
  - Early views on learning from observations (examples) and interplay between induction and deduction
  - "… scientific knowledge through demonstration [i.e., deduction] is impossible unless a man knows the primary immediate premises… we must get to know the primary premises by induction; for the method by which even sense-perception implants the universal is inductive…"



# Induction as Inverted Deduction: Operators

- Deductive Operators
  - Have mechanical operators ($F$) for finding *logically entailed conclusions* ($C$)
  - $F(A, B) = C$ where $A \wedge B \vdash C$
    - $A$, $B$, $C$: logical formulas
    - $F$: deduction algorithm
  - Intuitive idea: apply deductive inference (*aka* sequent) rules to $A$, $B$ to generate $C$
- Inductive Operators
  - Need operators $O$ to find *inductively inferred hypotheses* ($h$, "primary premises")
  - $O(B, D) = h$ where $\forall \langle x_i, f(x_i)\rangle \in D \,.\, (B \wedge D \wedge x_i) \vdash f(x_i)$
    - $B$, $D$, $h$: logical formulas describing observations
    - $O$: induction algorithm

## Induction as Inverted Deduction: Advantages and Disadvantages

- Advantages (Pros)
  - Subsumes earlier idea of finding *h* that "fits" training data
  - Domain theory *B* helps define meaning of "fitting" the data: $B \wedge D \wedge x_i \vdash f(x_i)$
  - Suggests algorithms that search *H* guided by *B*
    - Theory-guided constructive induction [Donoho and Rendell, 1995]
    - *aka* Knowledge-guided constructive induction [Donoho, 1996]
- Disadvantages (Cons)
  - Doesn't allow for noisy data
    - Q: Why not?
    - A: Consider what $\forall <x_i, f(x_i)> \in D \,.\, (B \wedge D \wedge x_i) \vdash f(x_i)$ stipulates
  - First-order logic gives a *huge* hypothesis space *H*
    - Overfitting…
    - Intractability of calculating all acceptable *h*'s



## Deduction: Resolution Rule

- Intuitive Idea
  - Suppose we know
    - $P \vee L$
    - $L \rightarrow R$
  - Can infer: $P \vee R$
  - Use this to reason over logical statements (propositions, first-order clauses)
- Resolution Rule
  - Sequent rule

$$\begin{array}{c} P \vee L \\ \underline{\neg L \vee R} \\ P \vee R \end{array}$$

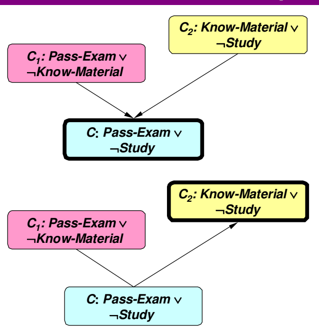
  - 1. Given: initial clauses $C_1$ and $C_2$, find literal *L* from clause $C_1$ such that $\neg L$ occurs in clause $C_2$
  - 2. Form the resolvent *C* by including all literals from $C_1$ and $C_2$, except *L* and $\neg L$
    - Set of literals occurring in conclusion *C* is

$$C = (C_1 - \{L\}) \cup (C_2 - \{\neg L\})$$

    - $\cup$ denotes set union, "-" denotes set difference



## Inverting Resolution: Example




Resolution

Inverse Resolution



## Inverted Resolution: Propositional Logic

- Problem Definition
  - Given: initial clauses $C_1$ and *C*
  - Return: clause $C_2$ such that *C* is resolvent of $C_1$ and $C_2$
- Intuitive Idea
  - Reason from consequent and partial set of premises to unknown premises
  - *Premise* ≡ hypothesis
- Inverted Resolution Procedure
  - 1. Find literal *L* that occurs in $C_1$ but not in *C*
  - 2. Form second clause $C_2$ by including the following literals:

$$C_2 = (C - (C_1 - \{L\})) \cup \{\neg L\}$$



## Quick Review: First-Order Predicate Calculus (FOPC)

- Components of FOPC Formulas: Quick Intro to Terminology
  - Constants: e.g., *John*, *Kansas*, 42
  - Variables: e.g., *Name*, *State*, *x*
  - Predicates: e.g., *Father-Of*, *Greater-Than*
  - Functions: e.g., *age*, *cosine*
  - Term: constant, variable, or *function*(*term*)
  - Literals (atoms): *Predicate*(*term*) or negation (e.g., $\neg$*Greater-Than* (*age* (*John*), 42)
  - Clause: disjunction of literals with implicit universal quantification
  - Horn clause: at most one positive literal ($H \vee \neg L_1 \vee \neg L_2 \vee \ldots \vee \neg L_n$)
- FOPC: Representation Language for First-Order Resolution
  - *aka* First-Order Logic (FOL)
  - Applications
    - Resolution using Horn clauses: logic programming (Prolog)
    - Automated deduction (deductive inference), theorem proving
  - Goal: learn first-order rules by *inverting first-order resolution*



## First-Order Resolution

- Intuitive Idea: Same as For Propositional Resolution
- Resolution Rule
  - Sequent rule: also same as for propositional resolution

$$\begin{array}{c} P \vee L \\ \underline{\neg L \vee R} \end{array}$$

  - 1. Given: initial clauses $C_1$ and $C_2$, find literal $L_1$ from clause $C_1$, literal $L_2$ from clause $C_2$, and *substitution* θ such that $L_1\theta = \neg L_2\theta$ (found using *unification*)
  - 2. Form the resolvent *C* by including all literals from $C_1\theta$ and $C_2\theta$, except $L_1\theta$ and $\neg L_2\theta$
    - Set of literals occurring in conclusion *C* is

$$C = (C_1 - \{L_1\})\theta \cup (C_2 - \{\neg L_2\})\theta$$

    - $\cup$ denotes set union, "-" denotes set difference
    - Substitution θ applied to sentences with matched literals removed

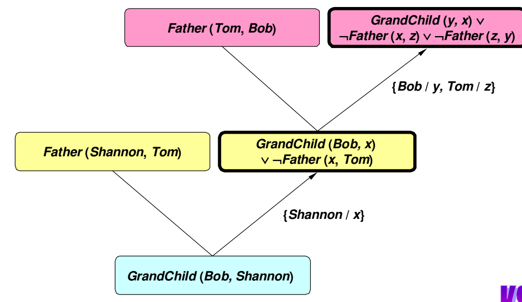
## Inverted Resolution: First-Order Logic

- **Problem Definition**
  - As for inverted propositional resolution
    - Given: initial clauses $C_1$ and $C$
    - Return: clause $C_2$ such that $C$ is resolvent of $C_1$ and $C_2$
  - Difference: must find, apply substitutions *and inverse substitutions*
- **Inverted Resolution Procedure**
  - 1. Find literal $L_1$ that occurs in $C_1$ but doesn't match any literal in $C$
  - 2. Form second clause $C_2$ by including the following literals:

$$C_2 = (C - (C_1 - \{L_1\})\theta_1)\theta_2^{-1} \cup \{\neg L_1 \theta_1 \theta_2^{-1}\}$$



## Inverse Resolution Algorithm (Cigol): Example






## Progol

- **Problem: Searching Resolution Space Results in Combinatorial Explosion**
- **Solution Approach**
  - Reduce explosion by generating *most specific acceptable h*
  - Conduct general-to-specific search (cf. *Find-G*, *CN2* ≡ *Learn-One-Rule*)
- **Procedure**
  - 1. User specifies $H$ by stating predicates, functions, and forms of arguments allowed for each
  - 2. *Progol* uses sequential covering algorithm
    - FOR each $<x_i, f(x_i)>$ DO

      Find most specific hypothesis $h_i$ such that $B \wedge h_i \wedge x_i \vdash f(x_i)$
    - Actually, considers only entailment within $k$ steps
  - 3. Conduct general-to-specific search bounded by specific hypothesis $h_i$, choosing hypothesis with minimum description length



## Learning First-Order Rules: Numerical versus Symbolic Approaches

- **Numerical Approaches**
  - Method 1: learning classifiers and extracting rules
    - Simultaneous covering: decision trees, ANNs
    - *NB: extraction methods may not be simple enumeration of model*
  - Method 2: learning rules directly using numerical criteria
    - Sequential covering algorithms and search
    - Criteria: MDL (information gain), accuracy, m-estimate, other heuristic evaluation functions
- **Symbolic Approaches**
  - Invert forward inference (deduction) operators
    - Resolution rule
    - Propositional and first-order variants
  - Issues
    - Need to control search
    - Ability to tolerate noise (contradictions): paraconsistent reasoning



## Terminology

- **Induction and Deduction**
  - Induction: finding $h$ such that $\forall <x_i, f(x_i)> \in D\ .\ (B \wedge D \wedge x_i) \vdash f(x_i)$
  - Inductive learning: $B$ ≡ background knowledge (inductive bias, etc.)
  - Developing *inverse deduction operators*
    - Deduction: finding entailed logical statements $F(A, B) = C$ where $A \wedge B \vdash C$
    - Inverse deduction: finding hypotheses $O(B, D) = h$ where $\forall <x_i, f(x_i)> \in D\ .\ (B \wedge D \wedge x_i) \vdash f(x_i)$
  - Resolution rule: deductive inference rule ($P \vee L, \neg L \vee R \vdash P \vee R$)
    - Propositional logic: boolean terms, connectives ($\wedge, \vee, \neg, \rightarrow$)
    - First-order predicate calculus (FOPC): well-formed formulas (WFFs), *aka* clauses (defined over literals, connectives, implicit quantifiers)
  - Inverse entailment: inverse of resolution operator
- **Inductive Logic Programming (ILP)**
  - Cigol: ILP algorithm that uses inverse entailment
  - Progol: sequential covering (general-to-specific search) algorithm for ILP



## Summary Points

- **Induction as Inverse of Deduction**
  - Problem of induction revisited
    - Definition of induction
    - Inductive learning as specific case
    - Role of induction, deduction in automated reasoning
  - Operators for automated deductive inference
    - Resolution rule (*and operator*) for deduction
    - First-order predicate calculus (FOPC) and resolution theorem proving
  - Inverting resolution
    - Propositional case
    - First-order case (inverse entailment operator)
- **Inductive Logic Programming (ILP)**
  - Cigol: inverse entailment (very susceptible to combinatorial explosion)
  - Progol: sequential covering, general-to-specific search using inverse entailment
- **Next Week: Knowledge Discovery in Databases (KDD), Final Review**