Machine Learning: Lecture 9

Rule Learning / Inductive Logic Programming

Learning Rules

- One of the most ***expressive*** and ***human readable*** representations for learned hypotheses is sets of ***production rules*** (***if-then*** rules).
- Rules can be derived from other representations (e.g., decision trees) or they can be learned ***directly***. Here, we are concentrating on the direct method.
- An important aspect of direct rule-learning algorithms is that they can learn sets of ***first-order rules*** which have much more representational power than the ***propositional*** rules that can be derived from decision trees. Learning first-order rules can also be seen as automatically inferring ***PROLOG programs*** from examples.

Propositional versus First-Order Logic

- ***Propositional Logic*** does not include variables and thus cannot express general relations among the values of the attributes.
- ***Example 1:*** in Propositional logic, you can write:
  IF $(Father_1=Bob) \wedge (Name_2=Bob) \wedge (Female_1=True)$ ***THEN*** $Daughter_{1,2}=True$.

  This rule applies only to a specific family!
- ***Example 2:*** In First-Order logic, you can write:
  IF $Father(y,x) \wedge Female(y)$, ***THEN*** $Daughter(x,y)$

  This rule (which you cannot write in Propositional Logic) applies to any family!

Learning Propositional versus First-Order Rules

- Both approaches to learning are useful as they address different types of learning problems.
- Like Decision Trees, Feedforward Neural Nets and IBL systems, Propositional Rule Learning systems are suited for problems in which no substantial relationship between the values of the different attributes needs to be represented.
- In First-Order Learning Problems, the hypotheses that must be represented involve relational assertions that can be conveniently expressed using first-order representations such as ***horn clauses*** (H <- L_1 ^…^L_n).

Learning Propositional Rules: Sequential Covering Algorithms

Sequential-Covering(*Target_attribute*, *Attributes*, *Examples*, *Threshold*)

- ***Learned_rules*** **<-- { }**
- ***Rule*** **<-- Learn-one-rule(*Target_attribute*, *Attributes*, *Examples*)**
- **While Performance(*Rule*, *Examples*) > *Threshold*, do**
  - ***Learned_rules*** **<-- *Learned_rules* + *Rule***
  - ***Examples*** **<-- *Examples* -{examples correctly classified by *Rule*}**
  - ***Rule*** **<-- Learn-one-rule(*Target_attribute*, *Attributes*, *Examples*)**
- ***Learned_rules*** **<-- sort *Learned_rules* according to Performance over *Examples***
- **Return *Learned_rules***

Learning Propositional Rules: Sequential Covering Algorithms

- The algorithm is called a ***sequential covering algorithm*** because it sequentially learns a set of rules that together cover the whole set of positive examples.
- It has the advantage of reducing the problem of learning a disjunctive set of rules to a sequence of simpler problems, each requiring that a single conjunctive rule be learned.
- The final set of rules is sorted so that the most accurate rules are considered first at classification time.
- However, because it does not backtrack, this algorithm is not guaranteed to find the smallest or best set of rules ---> Learn-one-rule must be very effective!

Learning Propositional Rules: *Learn-one-rule*

General-to-Specific Search:

- Consider the most general rule (hypothesis) which matches every instances in the training set.
- Repeat
  - Add the attribute that most improves rule performance measured over the training set.
- Until the hypothesis reaches an acceptable level of performance.

General-to-Specific Beam Search (CN2):

- Rather than considering a single candidate at each search step, keep track of the ***k*** best candidates.

Comments and Variations regarding the Basic Rule Learning Algorithms

- ***Sequential versus Simultaneous covering***: sequential covering algorithms (CN2) make a larger number of independent choices than simultaneous covering ones (ID3).
- ***Direction of the search:*** CN2 uses a general-to-specific search strategy. Other systems (GOLEM) uses a specific to general search strategy. General-to-specific search has the advantage of having a single hypothesis from which to start.
- ***Generate-then-test versus example-driven:*** CN2 is a generate-then-test method. Other methods (AQ, CIGOL) are example-driven. Generate-then-test systems are more robust to noise.

Comments and Variations regarding the Basic Rule Learning Algorithms,Cont'd

- ***Post-Pruning:*** pre-conditions can be removed from the rule whenever this leads to improved performance over a set of pruning examples distinct from the training set.
- Performance measure: different evaluation can be used. **Example:** ***relative frequency*** (AQ), ***m-estimate of accuracy*** (certain versions of CN2) and ***entropy*** (original CN2).

Learning Sets of First-Order Rules: FOIL (Quinlan, 1990)

FOIL is similar to the Propositional Rule learning approach except for the following:

- FOIL accommodates ***first-order*** rules and thus needs to accommodate ***variables*** in the rule pre-conditions.
- FOIL uses a special ***performance measure*** (FOIL-GAIN) which takes into account the different ***variable bindings***.
- FOILS seeks only rules that predict when the target literal is True (instead of predicting when it is True or when it is False).
- FOIL performs a simple ***hillclimbing*** search rather than a ***beam search***.

Induction as Inverted Deduction

- Let ***D*** be a set of training examples, each of the form $\langle x_i, f(x_i) \rangle$. Then, learning is the problem of discovering a hypothesis ***h***, such that the classification $f(x_i)$ of each training instance x_i follows deductively from the hypothesis ***h***, the description of x_i and any other background knowledge ***B*** known to the system.

Example:

- x_i: ***Male(Bob), Female(Sharon), Father(Sharon, Bob)***
- ***f(xi): Child(Bob, Sharon)***
- ***B: Parent(u,v) <-- Father(v,u)***
- we want to find ***h*** s.t., ***(B^h^xi) |-- f(xi).***

h1: Child(u,v) <-- Father(v,u)

h2: Child(u,v) <-- Parents(v,u)

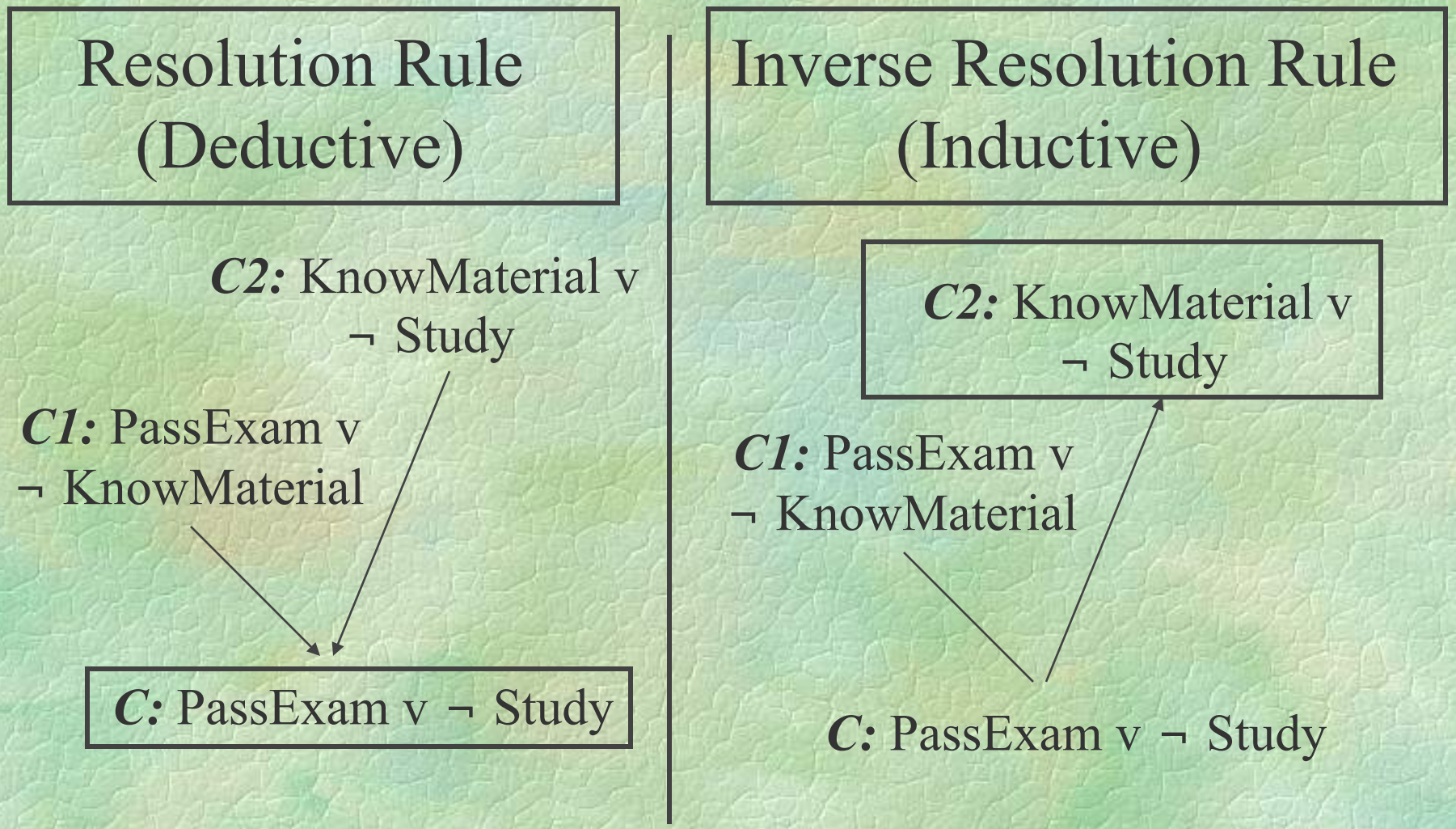
Induction as Inverted Deduction: Attractive Features

- The formulation subsumes the common definition of learning as finding some general concept that matches a given set of examples (with no background knowledge B, available)
- Incorporating the notion of Background information allows for a richer definition of when a hypothesis may be said to fit the data. It allows the introduction of domain-specific information.
- Background information can help guide the search for h, rather than merely searching the space of syntactically legal hypotheses.

Induction as Inverted Deduction: Difficulties

- The formal definition of learning does not naturally accommodate noisy data.
- The language of first-order logic is so expressive that the search through the space of hypotheses is intractable in the general case. Recent work has sought restricted forms of first-order expressions to improve search tractability.
- Despite the intuition that background knowledge should help constrain the search for a hypothesis, in most ILP systems, the complexity of the hypothesis space search increases as background knowledge increases. [Chapters 11 and 12, however, discuss how background knowledge can decrease this complexity]

An Example: CIGOL



This idea can also be applied to First-Order Resolution!

First-Order Multi-Step Inverse Resolution Example