Searching for Single Rules

- Introduction
  - Concept Learning
  - Generality Relations
  - Refinement Operators
  - Structured Hypothesis Spaces
- Simple algorithms
  - Find-S
  - Find-G
- Version Spaces
  - Version Spaces
  - Candidate-Elimination Algorithm
- Batch Learning

Concept Learning

- Given:
  - Positive Examples E^+
    - examples for the concept to learn (e.g., days with golf)
  - Negative Examples E^-
    - counter-examples for the concept (e.g., days without golf)
  - Hypothesis Space H
    - a (possibly infinite) set of candidate hypotheses
    - e.g., rules, rule sets, decision trees, linear functions, neural networks, ...
- Find:
  - Find the target hypothesis $h \in H$
  - the target hypothesis is the hypothesis that was used (could have been used) to generate the training examples

Correctness

- What is a good rule?
  - Obviously, a correct rule would be good
  - Other criteria: interpretability, simplicity, efficiency, ...
- Problem:
  - We cannot compare the learned hypothesis to the target hypothesis because we don't know the target
    - Otherwise we wouldn't have to learn...
- Correctness on training examples
  - ***completeness***: Each positive example should be covered by the target hypothesis
  - ***consistency***: No negative example should be covered by the target hypothesis
- But what we want is correctness on *all* possible examples!

Conjunctive Rule

$$\underbrace{\texttt{if}\ (\text{att}_i = \text{val}_{iI})\ \texttt{and}\ (\text{att}_j = \text{val}_{jJ})}_{\text{Body of the rule (IF-part)}}\quad \texttt{then}\ \underbrace{+}_{\text{Head of the rule (THEN-part)}}$$

Body of the rule (IF-part)

- contains a conjunction of conditions
- a condition typically consists of comparison of attribute values

Head of the rule (THEN-part)

- contains a prediction
- typically + if object belongs to concept, – otherwise

- Coverage
  - A rule is said to ***cover*** an example if the example satisfies the conditions of the rule.
- Prediction
  - If a rule covers an example, the rule's head is predicted for this example.

Propositional Logic

- simple logic of propositions
  - combination of simple facts
  - no variables, no functions, no relations (→ predicate calculus)
  - Operators:
    - conjunction ∧, disjunction ∨, negation ¬, implication →, ...

p	q	$p \rightarrow q$ $\neg p \vee q$
T	T	T
T	F	F
F	T	T
F	F	T

- rules with attribute/value tests may be viewed as statements in propositional logic
  - because all statements in the rule implicitly refer to the same object
  - each attribute/value pair is one possible condition
- Example:
  - `if` windy = false `and` outlook = sunny `then` golf
  - in propositional logic: $\neg$ windy $\wedge$ sunny_outlook $\rightarrow$ golf

Generality Relation

- Intuitively:
  - A statement is more general than another statement if it refers to a superset of its objects
- Examples:

more general

All students are good.
All students are good in Machine Learning.
All students who took a course in Machine Learning and Data Mining are good in Machine Learning
All students who took course ML&DM at the TU Darmstadt are good in Machine Learning
All students who took course ML&DM at the TU Darmstadt and passed with 2 or better are good in Machine Learning.

more specific

Generality Relation for Rules

- Rule r_1 is *more general* than r_2 $\quad r_1 \geq r_2$
  - if it covers all examples that are covered by r_2.
- Rule r_1 is *more specific* than r_2 $\quad r_1 \leq r_2$
  - if r_2 is more general than r_1.
- Rule r_1 is *equivalent* to r_2 $\quad r_1 \equiv r_2$
  - if it is more specific and more general than r_2.

Examples:

`if` size > 5 `then` +
`if` size > 3 `then` +

`if` animal = mammal `then` +
`if` feeds_children = milk `then` +

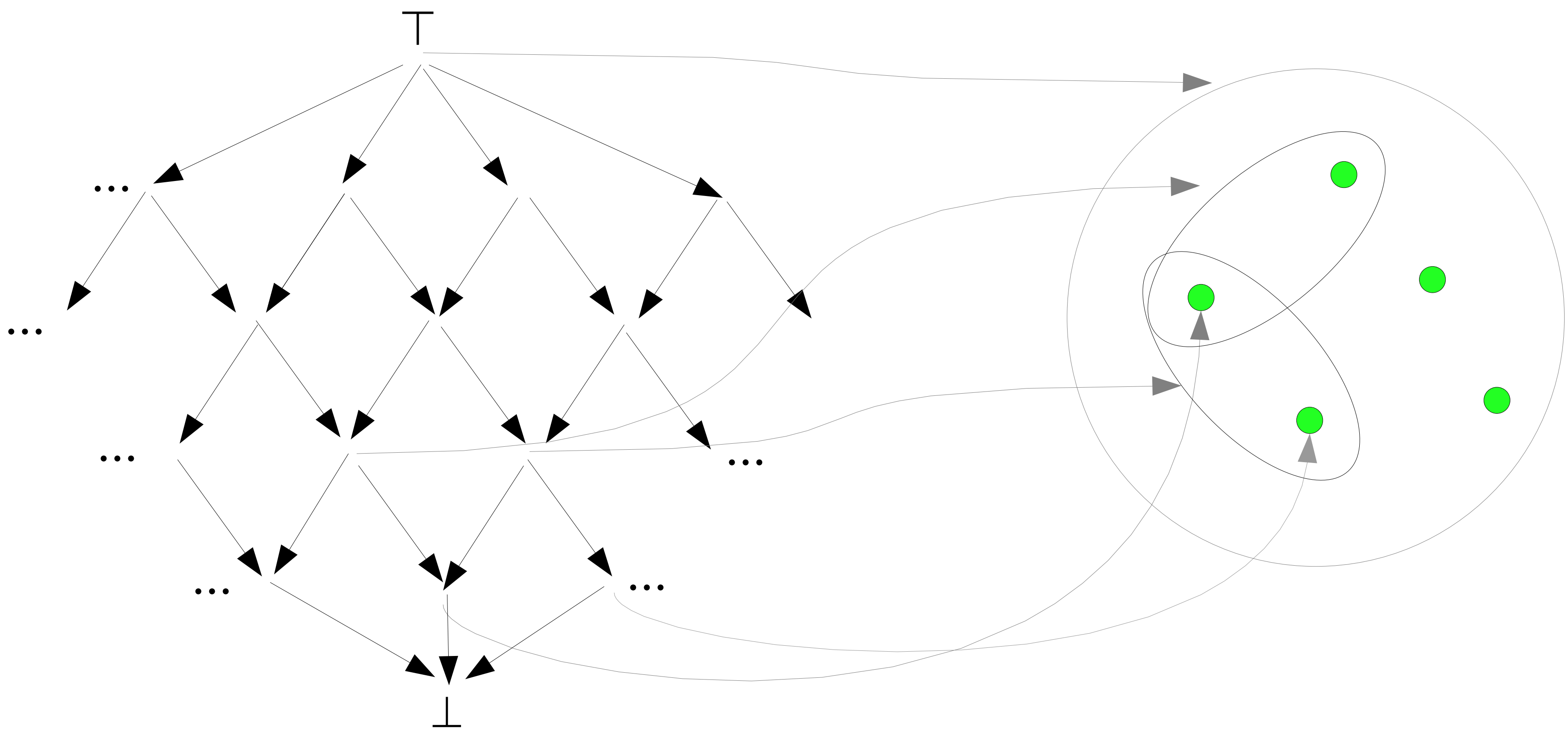
`if` outlook = sunny `then` +
`if` outlook = sunny `and` windy = false `then` +

Special Rules

- Most general rule $\top$
  - typically the rule that covers all examples
    - the rule with the body true
    - if disjunctions are allowed: the rule that allows all possible values for all attributes
- Most specific rule $\perp$
  - typically the rule that covers no examples
    - the rule with the body false
    - the conjunction of all possible values of each attribute
      - evaluates to false (only one value per attribute is possible)
- Each training example can be interpreted as a rule
  - body: all attribute-value tests that appear inside the example
  - the resulting rule is an immediate generalization of $\perp$
    - covers only a single example

Structured Hypothesis Space

- The availability of a generality relation allows to structure the hypothesis space:



Structured Hypothesis Space
arrows represent „is more general than“

Instance Space

Testing for Generality

- In general, we cannot check the generality of hypotheses
  - We do not have all examples of the domain available (and it would be too expensive to generate them)
- For single rules, we can approximate generality via a *syntactic generality check*:
  - **Example:** Rule r_1 is *more general* than r_2 if the set of conditions of r_1 forms a *subset* of the set of conditions of r_2.
  - Why is this only an approximation?
- For the general case, computable generality relations will rarely be available
  - E.g., rule sets
- Structured hypothesis spaces and version spaces are also a theoretical model for learning

Refinement Operators

- A refinement operator modifies a hypothesis
  - can be used to search for good hypotheses
- Generalization Operator:
  - Modify a hypothesis so that it becomes more general
    - e.g.: remove a condition from the body of a rule
  - necessary when a positive example is uncovered
- Specialization Operator:
  - Modify a hypothesis so that it becomes more specific
    - e.g., add a condition to the body of a rule
  - necessary when a negative examples is covered
- Other Refinement Operators:
  - in some cases, the hypothesis is modified in a way that neither generalizes nor specializes
    - e.g., stochastic or genetic search

Generalization Operators for Symbolic Attributes

There are different ways to generalize a rule, e.g.:

- **Subset Generalization**
  - a condition is removed
  - used by most rule learning algorithms

$$\text{shape} = \text{square} \;\&\; \text{color} = \text{blue} \rightarrow +$$
$$\Rightarrow$$
$$\text{color} = \text{blue} \rightarrow +$$

- **Disjunctive Generalization**
  - another option is added to the test

$$\text{shape} = \text{square} \;\&\; \text{color} = \text{blue} \rightarrow +$$
$$\Rightarrow$$
$$\text{shape} = (\text{square} \vee \text{rectangle}) \;\&\; \text{color} = \text{blue} \rightarrow +$$

- **Hierarchical Generalization**
  - a generalization hierarchy is exploited

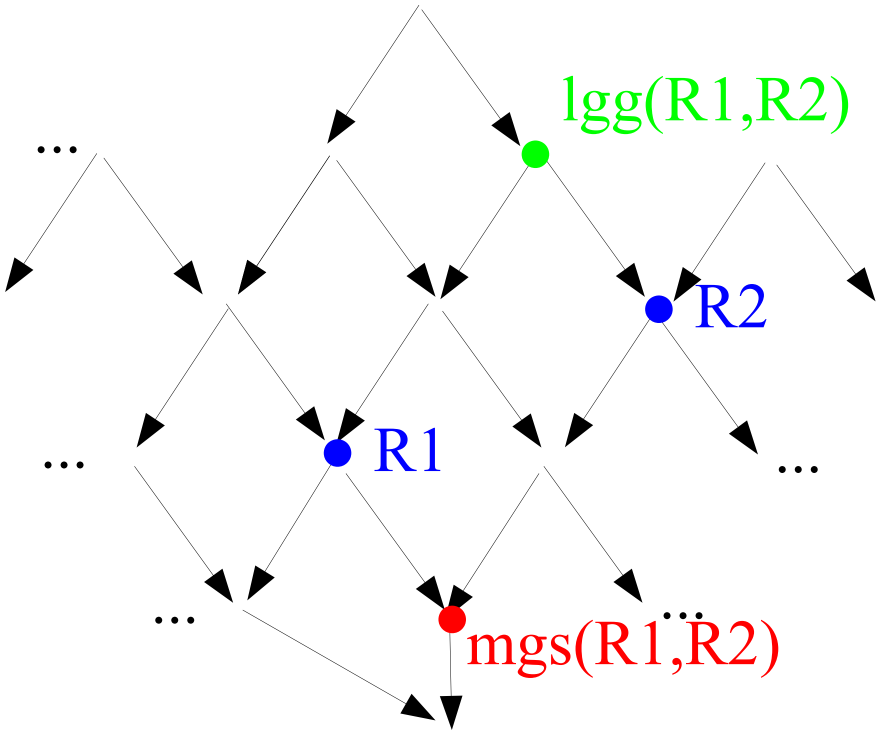
$$\text{shape} = \text{square} \;\&\; \text{color} = \text{blue} \rightarrow +$$
$$\Rightarrow$$
$$\text{shape} = \text{quadrangle} \;\&\; \text{color} = \text{blue} \rightarrow +$$

Minimal Refinement Operators

- In many cases it is desirable, to only make minimal changes to a hypothesis
  - specialize only so much as is necessary to uncover a previously covered negative example
  - generalize only so much as is necessary to cover a previously uncovered positive example
- Minimal Generalization relative to an example:
  - Find a generalization g of a rule r and an example e so that
    - g covers example e (r did not cover e)
    - there is no other rule g' so that $e \leq g' < g$ and $g' \geq r$
  - need not be unique
- Minimal Specialization relative to an example:
  - analogously

Subset Generalization of Rules

- least general generalization (lgg) of two rules
  - the intersection of the conditions of the rules
    (or a rule and an example)
- most general specialization (mgs) of two rules
  - the union of the conditions of the rules



minimal specialization relative to a rule/example

may be viewed as the lgg of the rule and the negation of the example

note that the negation of a conjunctive rule turns into a disjunction of several rules with one negated condition

Algorithm Find-S

The hypothesis
`if` **false** `then +`

I. h = most specific hypothesis in H
   (covering no examples)

II. for each training example e

   a) if e is negative
      - do nothing

   b) if e is positive
      - for each condition c in h
        - if c does not cover e
          - delete c from h

III. return h

Minimal Subset generalization
(other generalizations possible)

Note: when the first positive examples is encountered, step II.b) amounts to converting the example into a rule
(Recall that the most specific hypothesis can be written as a conjunction of all possible values of each attribute.)

Example

No.	Sky	Temperature	Humidity	Windy	Water	Forecast	sport?
1	sunny	hot	normal	strong	warm	same	yes
2	sunny	hot	high	strong	warm	same	yes
3	rainy	cool	high	strong	warm	change	no
4	sunny	hot	high	strong	cool	change	yes

H_0: `if` false `then +`

`if` (sky = sunny & sky = rainy & ... & forecast = same & forecaset = change) `then +`

{ <Ø,Ø,Ø,Ø,Ø,Ø> }

H_1: { <sunny, hot, normal, strong, warm,same> }

H_2: { <sunny, hot, ?, strong, warm,same> }

H_3: { <sunny, hot, ?, strong, warm,same> }

H_4: { <sunny, hot, ?, strong, ?, ? > }

Short-hand notation:

- only body (head is +)
- one value per attribute
- ⊘ for `false` (full conjunction)
- ? for `true` (full disjunction)

Algorithm Find-G

The hypothesis
`if` true `then +`

I. h = most general hypothesis in H
(covering all examples)

II. for each training example e

a) if e is positive
- do nothing

b) if e is negative
- for some condition c in e
  - if c is part of h
    - add a condition that negates c and covers all previous positive examples to h

III. return h

Minimal Subset specialization
(other specializations possible)

Example

No.	Sky	Temperature	Humidity	Windy	Water	Forecast	sport?
1	sunny	hot	normal	strong	warm	same	yes
2	sunny	hot	high	strong	warm	same	yes
3	rainy	cool	high	strong	warm	change	no
4	sunny	hot	high	strong	cool	change	yes

H_0: `if` true `then` +

`if` (sky = sunny || sky = rainy) & ... & (forecast = same || forecaset = change) `then` +

{ <?, ?, ?, ?, ?, ?> }

H_1: { <?, ?, ?, ?, ?, ?> }

H_2: { <?, ?, ?, ?, ?, ?> }

H_3: { <sunny, ?, ?, ?, ?, ?> }

H_4: { <sunny, ?, ?, ?, ?, ?> }

Other possibilities:

- <?, hot, ?, ?, ?, ?>
- <?, ?, ?, ?, ?, same>

Properties of Find-S and Find-G

- completeness:
  - h covers all positive examples
- consistency:
  - h will not cover any negative training examples
  - but only if the hypothesis space contains a target concept
    (i.e., there is a single conjunctive rule that describes the target concept)
- Properties:
  - no way of knowing whether it has found the target concept
    (there might be more than one theory that are complete and consistent)
  - Find-S prefers more specific hypotheses (hence the name)
    (it will never generalize unless forced by a training example)
  - Find-G prefers more general hypotheses (hence the name)
    (it will never specialize unless forced by a training example)
  - it only maintains one specific hypothesis
    (in other hypothesis languages there might be more than one)

Uniqueness of Refinement Operators

- Subset Specialization is not unique
  - we could specialize any condition in the rule that currently covers the example
  - we could specialize it to any value other than the one that is used in the example

→ a wrong choice may lead to an impasse

- Possible Solutions:
  - more expressive hypothesis language (e.g., disjunctions of values)
  - backtracking
  - remember all possible specializations and remove bad ones later
- Note: Generalization operators also need to be unique!

Algorithm Find-GSet

I. h = most general hypothesis in H (covering all examples)

II. $G = \{ h \}$

III. for each training example e

  a) if e is positive
    - remove all $h \in G$ that do not cover e

  b) if e is negative
    - for all hypotheses $h \in G$ that cover e
      - $G = G \setminus \{h\}$
      - for every condition c in e
        - for all conditions c' that negate c
          - $h' = h \cup \{c'\}$
          - if h' covers all previous positive examples
            - $G = G \cup \{h'\}$

IV. return G

Correct Hypotheses

- Find-GSet:
  - finds most general hypotheses that are correct on the data

  → has a bias towards general hypotheses
- Find-SSet:
  - can be defined analogously
  - finds most specific hypotheses that are correct on the data

  → has a bias towards specific hypotheses
- Thus, the hypotheses found by Find-GSet or Find-SSet are not necessarily identical!
- Could there be hypotheses that are correct but are neither found by GSet nor by SSet?

Version Space

- The Version Space V is the set of all hypotheses that
  - cover all positive examples (*completeness*)
  - do not cover any negative examples (*consistency*)
- For structured hypothesis spaces there is an efficient representation consisting of
  - the general boundary G
    - all hypotheses in V for which no generalization is in V
  - the specific boundary S
    - all hypotheses in V for which no specialization is in V
- a hypothesis that is neither in G nor in S is
  - a generalization of at least one hypothesis in S
  - a specialization of at least one hypothesis in G

Candidate Elimination Algorithm

- G = set of maximally general hypotheses
  S = set of maximally specific hypotheses
- For each training example e
  - if e is positive
    - For each hypothesis g in G that <u>does not cover</u> e
      - remove g from G
    - For each hypothesis s in S that <u>does not cover</u> e
      - remove s from S
      - $S = S \cup$ all hypotheses h such that
        - h is a minimal <u>generalization</u> of s
        - h covers e
        - some hypothesis in G is more general than h
      - remove from S any hypothesis that is more general than another hypothesis in S

Candidate Elimination Algorithm (Ctd.)

- if e is negative
  - For each hypothesis s in S that covers e
    - remove s from S
  - For each hypothesis g in G that covers e
    - remove g from G
    - $G = G \cup$ all hypotheses h such that
      - h is a minimal specialization of g
      - h does not cover e
      - some hypothesis in S is more specific than h
    - remove from G any hypothesis that is less general than another hypothesis in G

Example

No.	Sky	Temperature	Humidity	Windy	Water	Forecast	sport?
1	sunny	hot	normal	strong	warm	same	yes
2	sunny	hot	high	strong	warm	same	yes
3	rainy	cool	high	strong	warm	change	no
4	sunny	hot	high	strong	cool	change	yes

S_0: { <Ø,Ø,Ø,Ø,Ø,Ø> }
G_0: { <?, ?, ?, ?, ?, ?> }

S_1: { <sunny, hot, normal, strong, warm,same> }
G_1: { <?, ?, ?, ?, ?, ?> }

S_2: { <sunny, hot, ?, strong, warm,same> }
G_2: { <?, ?, ?, ?, ?, ?> }

S_3: { <sunny, hot, ?, strong, warm,same> }
G_3: { <sunny, ?, ?, ?, ?, ? >
<?, hot, ?, ?, ?, ? >
<?, ?, ?, ?, ?, same > }

S_4: { <sunny, hot, ?, strong, ?, ? > }
G_4: { <sunny, ?, ?, ?, ?, ? >
<?, hot, ?, ?, ?, ? >}

How to Classify these Examples?

- Version Space:

G <sunny, ?, ?, ?, ?, ? > <?, hot, ?, ?, ?, ? >

<sunny, ?, ?, strong, ?, ?> <sunny, hot, ?, ?, ?, ?> <?, hot, ?, strong, ?, ?>

S <sunny, hot, ?, strong, ?, ? >

- How to Classify these Examples?

No.	Sky	Temperature	Humidity	Windy	Water	Forecast	sport?
5	sunny	hot	normal	strong	cool	change	yes
6	rainy	cool	normal	light	warm	same	no
7	sunny	hot	normal	light	warm	same	?
8	sunny	cool	normal	strong	warm	same	maybe no

Properties

- Convergence towards target theory
  - convergence if $S = G$
- Reliable classification with partially learned concepts
  - an example that matches all elements in S must be a member of the target concept
  - an example that matches no element in G cannot be a member of the target concept
  - examples that match parts of S and G are undecidable
- no need to remember examples (*incremental* learning)
- Assumptions
  - no errors in the training set
  - the hypothesis space contains the target theory
  - practical only if generality relation is (efficiently) computable

Other Generality Relations

- First-Order
  - generalize the arguments of each pair of literals of the same relation
- Hierarchical Values
  - generalization and specialization for individual attributes follows the ontology

Generalization Operators for Numerical Attributes

- Subset Generalization
    - generalization works as in symbolic case
    - specialization is difficult as there are infinitely different values to specialize to
- Disjunctive Generalization
    - specialization and generalization as in symbolic case
    - problematic if no repetition of numeric values can be expected
        - generalization will only happen on training data
        - no new unseen examples are covered after a generalization
- Interval Generalization
  - the range of possible values is represented by an open or closed intervals
    - generalize by widening the interval to include the new point
    - specialize by shortening the interval to exclude the new point

Batch induction

- So far we looked at
  - all theories at the same time (implicitly through the version space)
  - and processed examples incrementally
- We can turn this around:
  - work on the theories incrementally
  - and process all examples at the same time
- Basic idea:
  - try to quickly find a complete and consistent rule
  - need not be in either S or G (but in the version space)
- Algorithm like FindG:
  - successively refine rule by adding conditions:
    - evaluate all refinements and pick the one that looks best
  - until the rule is consistent

Algorithm Batch-FindG

I. h = most general hypothesis in H
   C = set of all possible conditions

II. while h covers negative examples

   I. for each possible condition $c \in C$

      a) $h' = h \cup \{c\}$

      b) if h' covers

         - all positive examples
         - and fewer negative examples than h_{best}

         then $h_{best} = h'$

   II. $h = h_{best}$

III. return h

Scan through all examples in database:
- count covered positives
- count covered negatives

Properties

- General-to-Specific (Top-Down) Search
  - similar to FindG:
    - **FindG** makes an arbitrary selection among possible refinements, taking the risk that it may lead to an consistency later
    - **Batch-FindG** selects next refinement based on all training examples
- Heuristic algorithm
  - among all possible refinements, we select the one that leads to the fewest number of covered negatives
    - IDEA: the more negatives are excluded with the current condition, the less have to be excluded with subsequent conditions
- Converges towards some theory
  - not necessarily towards a theory in G
- Not very efficient, but quite flexible
  - criteria for selecting conditions could be exchanged