Concept Learning

CS4881 Artificial Intelligence

Jay Urbain

Credits:

Machine Learning, Tom Mitchell

Concept Learning

- Learning from examples
- General-to-specific ordering over hypotheses
- Version spaces and candidate elimination algorithm
- Picking new examples
- The need for inductive bias

Note: Fundamental approach towards machine learning assuming no noise to illustrate key concepts.

Training Examples for EnjoySport

- What is the *general* concept?

Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Representing Hypothesis

- Many possible representations
  - Decision Tree, Neural Net, conditional probabilities
- Here, ***h*** is the conjunction of constraints on attributes
- Each constraint can be
  - A specific value, e.g. Water=Warm
  - Don’t care, e.g. Water=?
  - No value allowed, e.g., Water=0
- Example:

  Sky, AirTemp, Humid, Wind, Water, Forecast

<Sunny, ?, ?, Strong , ?, Same>

Prototypical Concept Learning Task

Given:

- Instances *X*: Possible days, each described by the feature attributes *Sky, AirTemp, Humidity, Wind, Water, Forecast.*

- Target function *c: EnjoySport: X* -> {0, 1}

- Hypothesis *H*: Conjunctions of literals. E.g.

  <?, Cold, High, ?, ?, ?>

- Training examples *D*: Positive and negative examples of the target function.

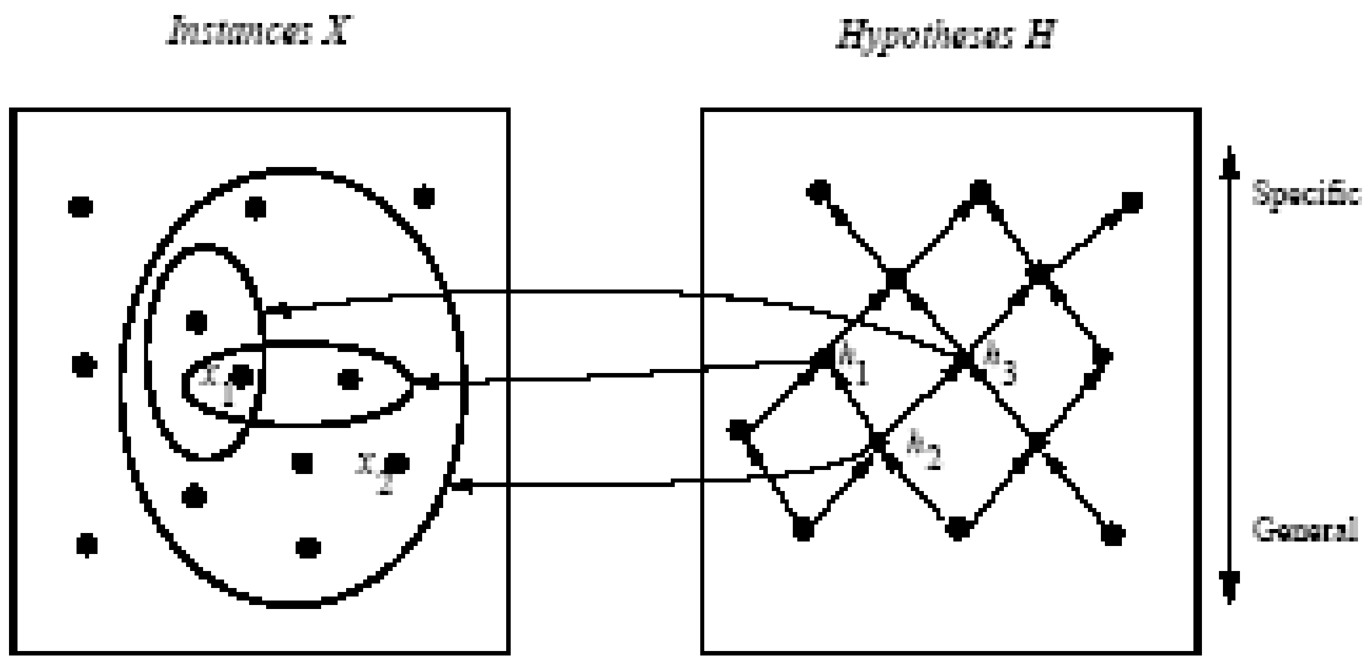
  $$\langle x_1, c(x_1)\rangle, \ldots \langle x_m, c(x_m)\rangle$$

- Determine: A hypothesis *h* in *H* such that *h(x) = c(x)* for all *x* in *D*.

Inductive learning hypothesis

The inductive learning hypothesis: Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

Instance, Hypotheses, and More-General-Than



x_1 = <Sunny, Warm, High, Strong, Cool, Same>
x_2 = <Sunny, Warm, High, Light, Warm, Same>

h_1 = <Sunny, ?, ?, Strong, ?, ?>
h_2 = <Sunny, ?, ?, ?, ?, ?>
h_3 = <Sunny, ?, ?, ?, Cool, ?>

Find-S Algorithm

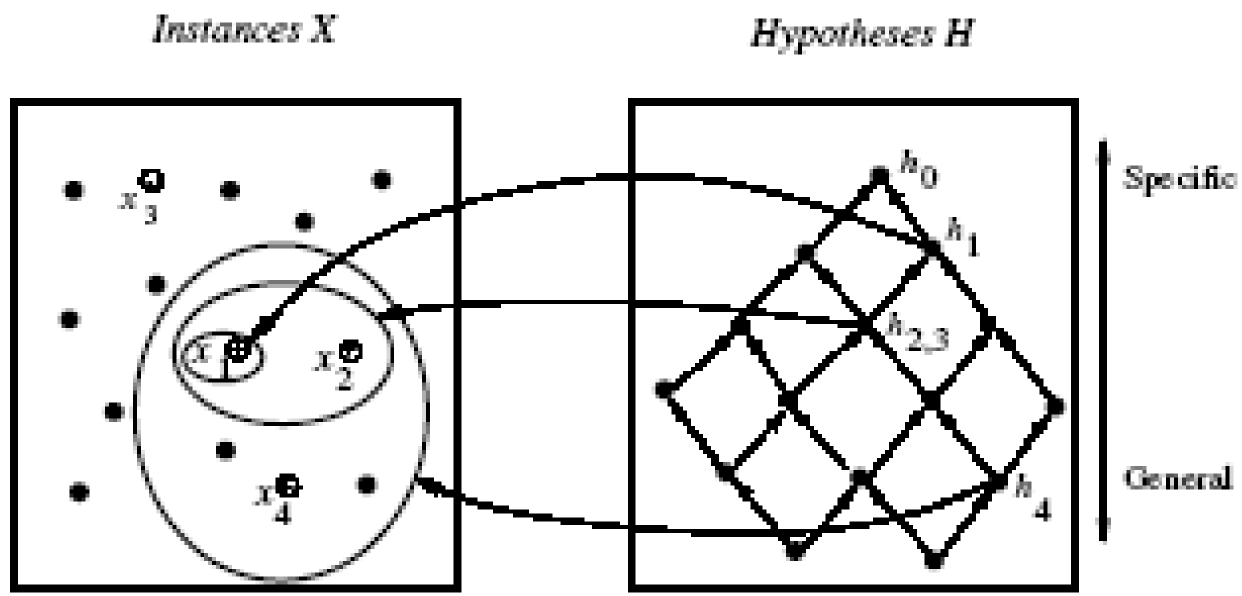
1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   - For each attribute constraint a_i in h

     If the constraint a_i in h is satisfied by x

     Then do nothing

     Else replace a_i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

Hypothesis Space Search by Find-S



x_1 = <Sunny Warm Normal Strong Warm Same>, +

x_2 = <Sunny Warm High Strong Warm Same>, +

x_3 = <Rainy Cold High Strong Warm Change>, -

x_4 = <Sunny Warm High Strong Cool Change>, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>

h_1 = <Sunny Warm Normal Strong Warm Sam

h_2 = <Sunny Warm ? Strong Warm Same>

h_3 = <Sunny Warm ? Strong Warm Same>

h_4 = <Sunny Warm ? Strong ? ? >

Complaints about Find-S

- Can’t tell whether it has learned the concept
- Can’t tell when training data is inconsistent
- Picks a maximally specific *h* (why?)
  - What about algorithm’s ability to generalize over unseen examples?
  - Depending on *H*, there might be several!

Version Spaces

A hypothesis h is **consistent** with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in D.

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D)\ h(x) = c(x)$$

The **version space**, $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D.

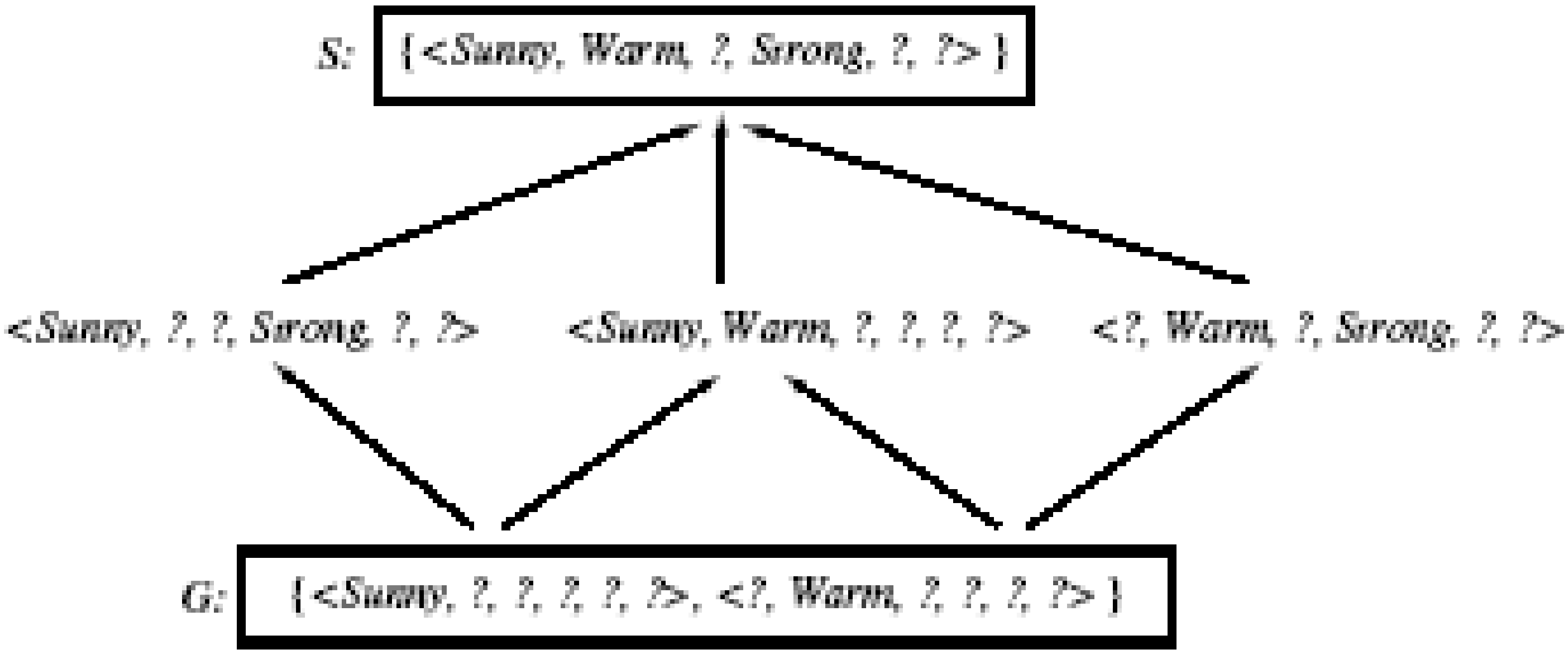
$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

List-then-Eliminate Algorithm

1. $VersionSpace \leftarrow$ a list containing every hypothesis in H
2. For each training example, $\langle x, c(x) \rangle$

   remove from $VersionSpace$ any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypotheses in $VersionSpace$

Example: Version Space



Representing Version Spaces

- The **General boundary G**, of version space $VS_{H,D}$ is the set of its maximally general members.
- The **Specific boundary S**, of version space $VS_{H,D}$ is the set of its maximally specific members.
- Every member of the version space lies between these boundaries.

$$VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$$

Candidate Elimination Algorithm

$G \leftarrow$ maximally general hypotheses in H
$S \leftarrow$ maximally specific hypotheses in H
For each training example d, do

- If d is a positive example
  - Remove from G any hypothesis inconsistent with d
  - For each hypothesis s in S that is not consistent with d
    - Remove s from S
    - Add to S all minimal generalizations h of s such that
      1. h is consistent with d, and
      2. some member of G is more general than h
    - Remove from S any hypothesis that is more general than another hypothesis in S
- If d is a negative example

Candidate Elimination Algorithm (cont.)

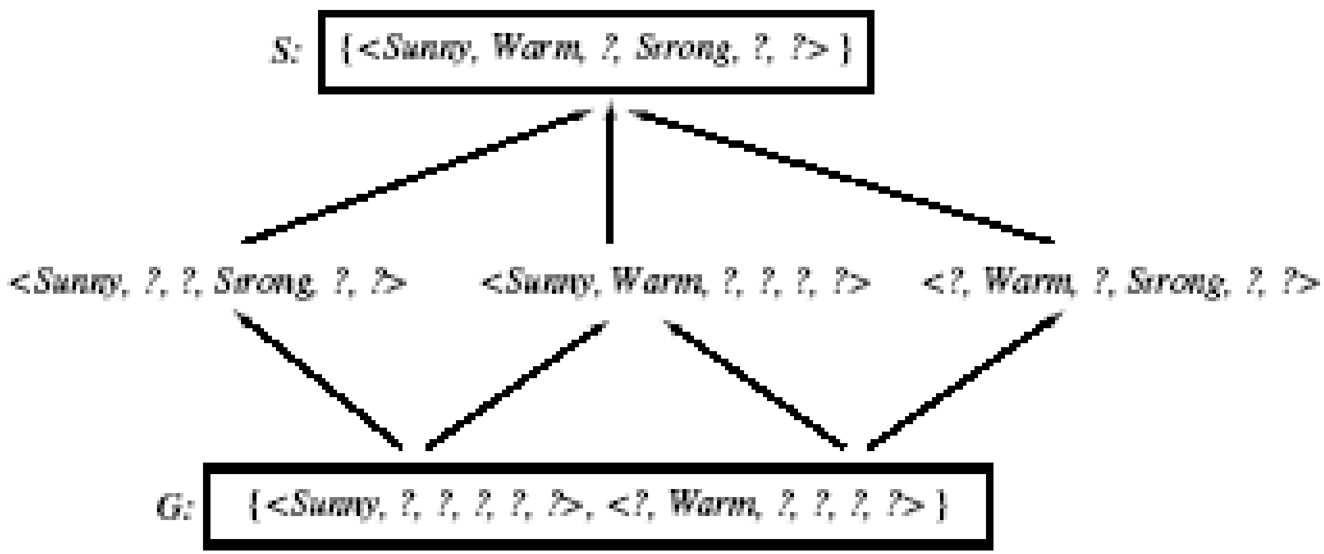
- Remove from S any hypothesis inconsistent with d
- For each hypothesis g in G that is not consistent with d
  - Remove g from G
  - Add to G all minimal specializations h of g such that
    1. h is consistent with d, and
    2. some member of S is more specific than h
  - Remove from G any hypothesis that is less general than another hypothesis in G

Example trace of candidate elimination

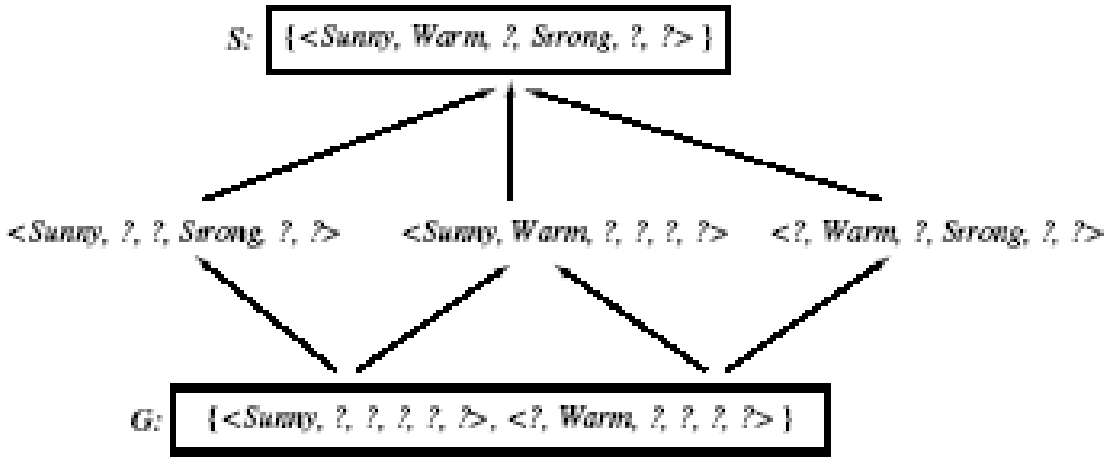
S_0: $\{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$

G_0: $\{\langle ?, ?, ?, ?, ?, ? \rangle\}$

Example trace of candidate elimination



How should this these be classified?



$$\langle Sunny\ Warm\ Normal\ Strong\ Cool\ Change\rangle$$

$$\langle Rainy\ Cool\ Normal\ Light\ Warm\ Same\rangle$$

$$\langle Sunny\ Warm\ Normal\ Light\ Warm\ Same\rangle$$

What justifies this inductive leap?

$$+ \ \langle Sunny\ Warm\ Normal\ Strong\ Cool\ Change \rangle$$

$$+ \ \langle Sunny\ Warm\ Normal\ Light\ Warm\ Same \rangle$$

$$S: \quad \langle Sunny\ Warm\ Normal\ ?\ ?\ ? \rangle$$

Why believe we can classify the unseen

$$\langle Sunny\ Warm\ Normal\ Strong\ Warm\ Same \rangle$$

Inductive Bias

- *CANDIDATE-ELIMINATION* algorithm will converge toward the true target concept.
  - Provided it is given accurate training examples
  - Provided its initial hypothesis space contains the target concept.

Biased Learner

- Assume we want to make sure the hypothesis space contains the unknown target concept.
- Consider a hypothesis space that only allows *conjunction* of attribute values.
- Because of this restriction we will not be able to represent *disjunctive* target concepts:

  <sky=cloudy OR sky=sunny>

We have biased the learner to only consider conjunctive hypotheses.

UnBiased Learner

- Obvious solution is to provide a hypothesis space with every teachable concept.
- For *EnjoySport* the instance space is 96.
- How many possible concepts can be defined over these instances?
  - The power set 2^{96}! (2^s set of all subsets s)

UnBiased Learner

- Assuming we can enumerate this hypothesis space of every teachable concept.
- We can represent this as arbitraty disjunctions, conjunctions and negations reflected from shown examples:

  {Sunny,?,?,?,?,?} V {Cloudy,?,?,?,?,?}
- ***We now have another problem: our concept learning algorithm can no longer generalize!***
- We would end up with:

  S:{x1vX2vX3) G:{not(X4vX5}

The futility of bias-free learning

- A learner that makes no *apriori* assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.

The futility of bias-free learning

- CANDIDATE-ELIMINATION algo. was able to generalize beyond the observed training examples because it was *biased* by the implicit assumption that the target concept could be represented by a conjunction of attribute values.

- What is the bias of a route learner?

- Decision tree?

- Neural Net?

Inductive bias

- The minimal set of assertions a concept learning algorithm makes concerning the hypothesis space containing the target concept.
- I.e., an assumption about how the hypothesis space is structured, and that it can represent the target concept.

Inductive bias

Consider

- concept learning algorithm L
- instances X, target concept c
- training examples $D_c = \{\langle x, c(x) \rangle\}$
- let $L(x_i, D_c)$ denote the classification assigned to the instance x_i by L after training on data D_c.

Definition:

The **inductive bias** of L is any minimal set of assertions B such that for any target concept c and corresponding training examples D_c

$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

where $A \vdash B$ means A logically entails B

The Futility of Bias-Free Learning

- **<u>Fundamental Property of Inductive Learning</u>**
  - A learner that makes no *apriori* assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.
- **<u>We constantly have recourse to inductive biases</u>**
  - *Example:* we all know that the sun will rise tomorrow. Although we cannot *deduce* that it will do so based on the fact that it rose today, yesterday, the day before, etc., we do take this **leap of faith** or use this **inductive bias**, naturally!

Three Learners with Different Biases

- *Rote learner*: Store examples, Classify x *iff* it matches previously observed example.
  - This system simply memorizes the training data and their classification - No generalization is involved.
- *Version space candidate elimination algorithm*
  - New instances are classified only if all the hypotheses in the version space agree on the classification.
- *Find-S*
  - New instances are classified using the most specific hypothesis consistent with the training data.

Three Learners with Different Biases

- *Remebmer!*
  - ***<u>An unbiased learner is not able to generalize beyond the observed examples!!!!</u>***

Summary

1. Concept learning as search through *H*
2. General-to-specific ordering over *H*
3. Version space candidate elimination algorithm
4. *S* and *G* boundaries characterize learner's uncertainty
5. Learner can generate useful queries
6. Inductive leaps possible only if learner is biased
7. Inductive learners can be modeled by equivalent deductive systems.