Concept Learning

TM Chapter 2

Outline

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

Why Study Concept Learning?

- Simple (e.g., assumes error-free, noise-free data) and practically useless!
- Easier to fully understand and explain the general challenges, issues, and concepts in ML
- White-box model: Prediction is interpretable & explainable
- Relates well to your earlier modules: Discrete math (CS1231), logic, search (CS3243)
- Principled and rigorously grounded: Proofs, proofs, and more proofs!

What is Concept Learning?

- What is a concept? A boolean-valued function over a set of input instances (each comprising input attributes)
- Concept learning is a form of supervised learning. Infer an unknown boolean-valued function from training examples

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Positive (+ve) and negative (–ve) training examples for target concept *EnjoySport*

How to Represent a Hypothesis?

- Many possible hypothesis representations: Trade-off between expressive power vs. smaller hypothesis space
- Consider a simple representation: Hypothesis h is a conjunction of constraints on input attributes
- Each constraint can be
  - a specific value (e.g., "*Water* = *Warm*")
  - don't care (e.g., "*Water* = ?")
  - no value allowed (e.g., "*Water* = Ø")

Sky	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*
$\langle$*Sunny*,	?,	?,	*Strong*,	?,	*Same*$\rangle$

Concept Learning for *EnjoySport*

Given

- Input instances X: Each instance $x \in X$ is represented by the following input attributes describing the day:
  - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*)
  - *AirTemp* (with values *Warm* and *Cold*)
  - *Humidity* (with values *Normal* and *High*)
  - *Wind* (with values *Strong* and *Weak*)
  - *Water* (with values *Warm* and *Cool*)
  - *Forecast* (with values *Same* and *Change*)

Concept Learning for *EnjoySport*

Given

- Hypothesis space H: Each hypothesis $h \in H$ ($h : X \rightarrow \{0, 1\}$) is represented by a conjunction of constraints (see page 5) on input attributes (e.g., $\langle \mathit{Sunny}, ?, ?, \mathit{Strong}, ?, \mathit{Same} \rangle$).

Definition. An input instance $x \in X$ **satisfies** (all constraints of) a hypothesis $h \in H$ iff $h(x) = 1$.

In other words, h classifies x as a +ve example.

Concept Learning for *EnjoySport*

Given

- Unknown target concept/function *EnjoySport*: $c : X \rightarrow \{0, 1\}$
- Noise-free training examples D: +ve and −ve examples of the target function (e.g., $D = \{\langle x_k, c(x_k)\rangle\}_{k=1,\ldots,n}$)

Determine a hypothesis $h \in H$ that is **consistent** with D

Definition. A hypothesis h is **consistent** with a set of training examples D iff $h(x) = c(x)$ for all $\langle x, c(x)\rangle \in D$.

How is saying 'h is **consistent** with training example $\langle x, c(x)\rangle$' different from 'saying instance x **satisfies** h'? Implication?

Inductive Learning Assumption

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.

Concept Learning is Search

Goal. Search for a hypothesis $h \in H$ that is **consistent** with D

For *EnjoySport* task, H contains

- $5 \times 4 \times 4 \times 4 \times 4 \times 4 = 5120$ syntactically distinct hypotheses
- $1 + 4 \times 3 \times 3 \times 3 \times 3 \times 3 = 973$ semantically distinct hypotheses
- Every hypothesis containing 1 or more Ø symbols represents an empty set of input instances, hence classifying every instance as a –ve example

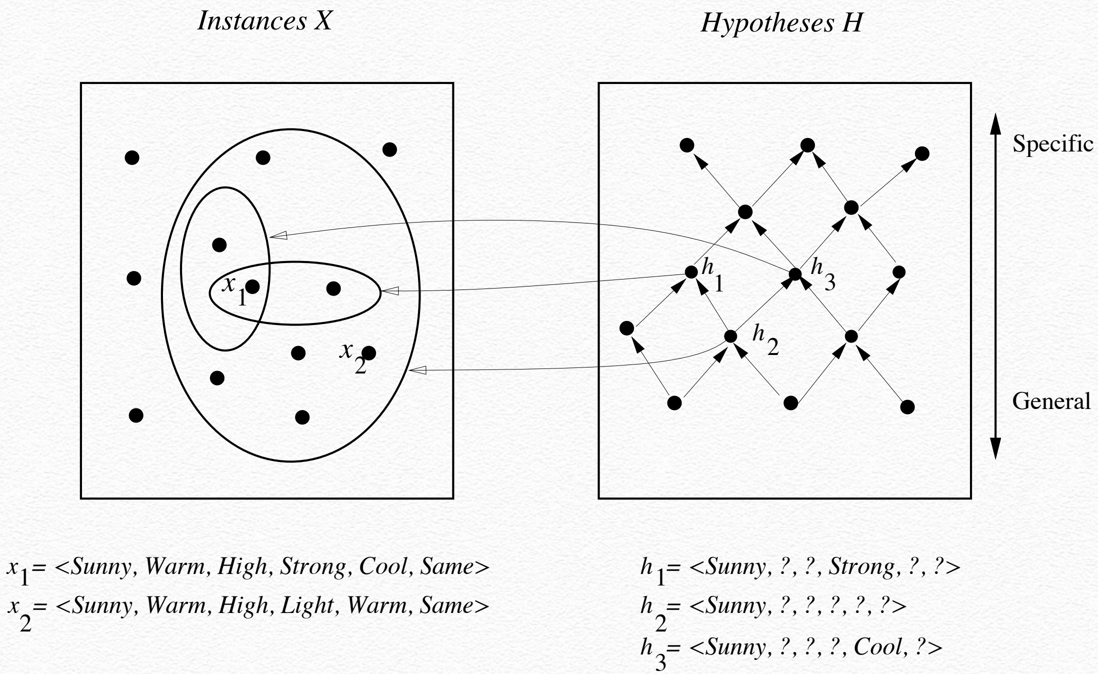
In practice, hypothesis space H is much larger and possibly infinite, hence motivating the need to exploit structure for searching efficiently.

Exploit Structure in Concept Learning

Definition. h_j is **more general than or equal to** h_k (denoted by $h_j \geq_g h_k$) iff any input instance x that satisfies h_k also satisfies h_j:

$$\forall x \in X \ \ (h_k(x) = 1) \rightarrow (h_j(x) = 1) \ .$$

$\geq_g$ relation defines a partial order (reflexive, antisymmetric, transitive) over H & not total order (e.g., $h_1 \ngeq_g h_3$ and $h_3 \ngeq_g h_1$)



Exploit Structure in Concept Learning

Definition. h_j is **more general than or equal to** h_k (denoted by $h_j \geq_g h_k$) iff any input instance x that satisfies h_k also satisfies h_j:

$$\forall x \in X \ \ (h_k(x) = 1) \rightarrow (h_j(x) = 1) \ .$$

Definition. h_j is **more general than** h_k (denoted by $h_j >_g h_k$) iff $h_j \geq_g h_k$ and $h_k \ngeq_g h_j$.

Definition. h_j is **more specific than** h_k iff h_k is **more general than** h_j.

Definition. h_j is **more specific than or equal to** h_k iff h_k is **more general than or equal to** h_j.

FIND-S Algorithm

Idea. Start with most specific hypothesis. Whenever it wrongly classifies a +ve training example as −ve, "minimally" generalize it to satisfy its input instance.

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

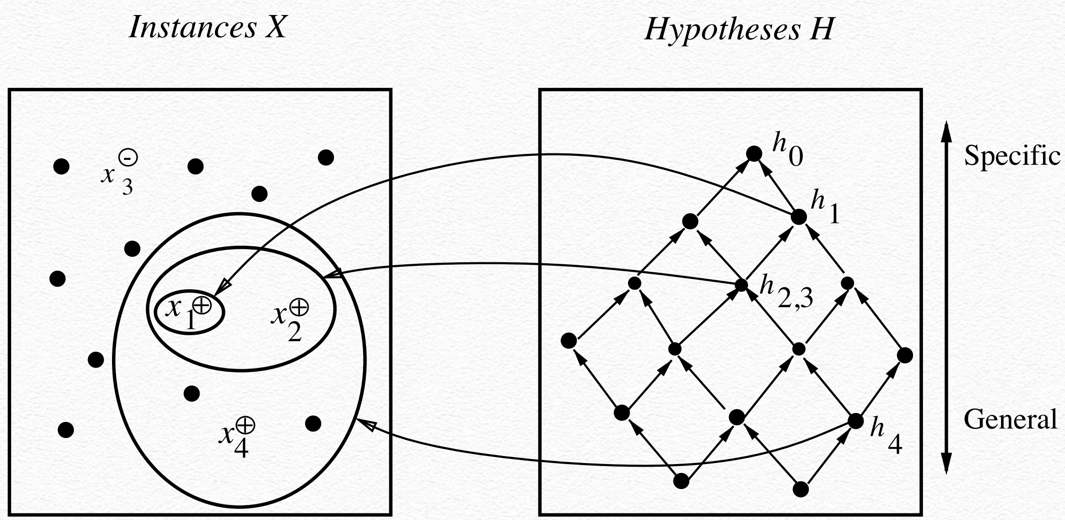
     If x satisfies constraint a_i in h

     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

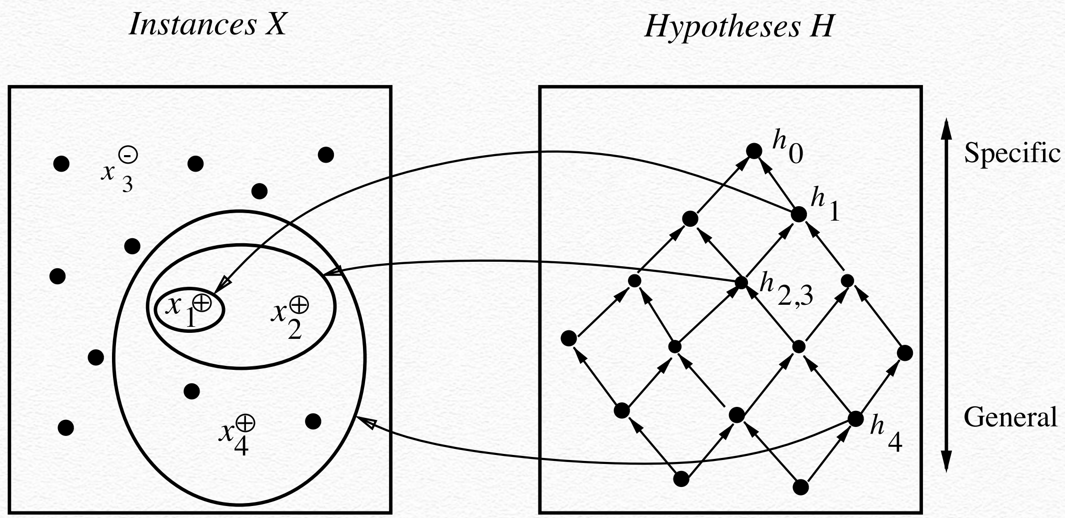
Proposition 1. h is consistent with D iff every +ve training instance satisfies h and every −ve training instance does not satisfy h.



	h_0 = <∅, ∅, ∅, ∅, ∅, ∅>
x_1 = <*Sunny Warm Normal Strong Warm Same*>, +	h_1 = <*Sunny Warm Normal Strong Warm Same*>
x_2 = <*Sunny Warm High Strong Warm Same*>, +	h_2 = <*Sunny Warm ? Strong Warm Same*>
x_3 = <*Rainy Cold High Strong Warm Change*>, -	h_3 = <*Sunny Warm ? Strong Warm Same*>
x_4 = <*Sunny Warm High Strong Cool Change*>, +	h_4 = <*Sunny Warm ? Strong ? ?* >

Hypothesis Space Search by FIND-S

Proposition 2. Suppose that $c \in H$. Then, h_n is consistent with $D = \{\langle x_k, c(x_k)\rangle\}_{k=1,\ldots,n}$.



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 Same>
h_2 = <Sunny Warm ? Strong Warm Same>
h_3 = <Sunny Warm ? Strong Warm Same>
h_4 = <Sunny Warm ? Strong ? ? >

Limitations of FIND-S

- Can't tell whether Find-S has learned target concept
- Can't tell when training examples are inconsistent (i.e., contain errors or noise)
- Picks a maximally specific h (why?)
- Depending on H, there might be several!

Version Spaces

Definition. The **version space** $VS_{H,D}$ wrt hypothesis space H and training examples D, is the subset of hypotheses from H consistent with D:

$$VS_{H,D} = \{h \in H \mid h \text{ is consistent with } D\} \ .$$

- If $c \in H$, then a large enough D can reduce $VS_{H,D}$ to $\{c\}$
- If D is insufficient, then $VS_{H,D}$ represents the uncertainty of what the target concept is
- $VS_{H,D}$ contains all consistent hypotheses, including maximally specific hypotheses

LIST-THEN-ELIMINATE Algorithm

Idea. List all hypotheses in H. Then, eliminate any hypothesis found inconsistent with any training example.

1. *VersionSpace* $\leftarrow$ a list containing every hypothesis in H

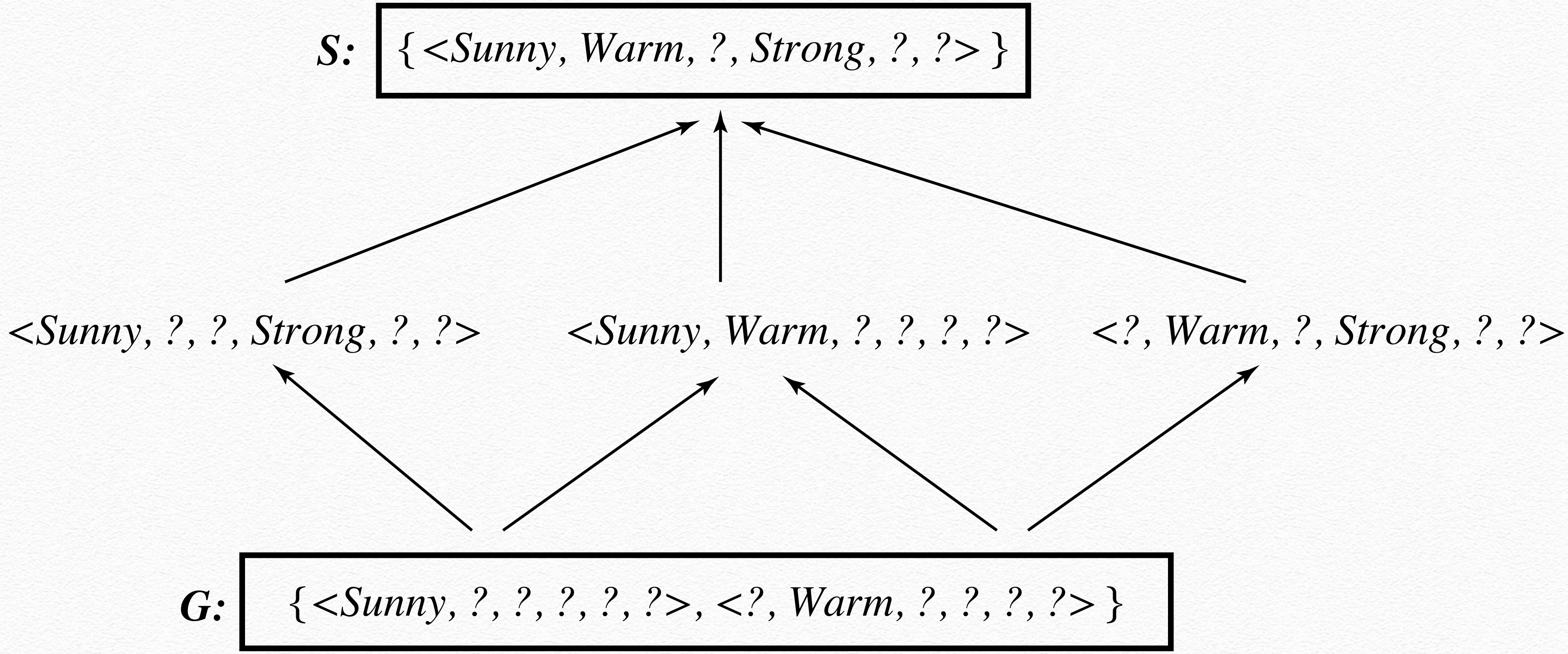
2. For each training example $\langle x, c(x) \rangle$

   Remove from *VersionSpace* any hypothesis h for which $h(x) \neq c(x)$

3. Output the list of hypotheses in *VersionSpace*

Limitation. Prohibitively expensive to exhaustively enumerate all hypotheses in finite H

Version Space for *EnjoySport*



Compact Representation of Version Space

Definition. The **general boundary** G of $VS_{H,D}$ is the set of maximally general members of H consistent with D:

$$G = \{g \in H \mid g \text{ consistent with } D \wedge (\neg\exists g' \in H \;\; g' >_g g \wedge g' \text{ consistent with } D)\}.$$

Definition. The **specific boundary** S of $VS_{H,D}$ is the set of maximally specific members of H consistent with D:

$$S = \{s \in H \mid s \text{ consistent with } D \wedge (\neg\exists s' \in H \;\; s >_g s' \wedge s' \text{ consistent with } D)\}.$$

Every member of version space lies between these boundaries:

Version space representation theorem (VSRT).

$$VS_{H,D} = \{h \in H \mid \exists s \in S \; \exists g \in G \;\; g \geq_g h \geq_g s\} \; .$$

Proof of Version Space Representation Theorem

$\Leftarrow$ Every h satisfying RHS is in $VS_{H,D}$.

1. Choose arbitrary $g \in G$, $s \in S$, $h \in H$ s.t. $g \geq_g h \geq_g s$
2. Every +ve training instance satisfies s, by Def. of S and Prop. 1
3. Since $h \geq_g s$, every +ve training instance satisfies h
4. Every −ve training instance does not satisfy g, by Def. of G and Prop. 1
5. Since $g \geq_g h$, every −ve training instance does not satisfy h
6. h is consistent with D, by Prop. 1 and steps 3 and 5
7. $h \in VS_{H,D}$

$\Rightarrow$ Every member of $VS_{H,D}$ satisfies RHS. DIY.

CANDIDATE-ELIMINATION Algorithm

Idea. Start with most general and specific hypotheses. Each training example "minimally" generalizes S and specializes G to remove inconsistent hypotheses from version space.

1. $G \leftarrow$ maximally general hypotheses in H

2. $S \leftarrow$ maximally specific hypotheses in H

CANDIDATE-ELIMINATION Algorithm

3. For each training example *d*
   - If *d* is a +ve example
     - Remove from *G* any hypothesis inconsistent with *d*
     - For each $s \in S$ not consistent with *d*
       - Remove *s* from *S*
       - Add to *S* all minimal generalizations *h* of *s* s.t.

         h is consistent with *d*, and

         some member of *G* is more general than or equal to *h*
       - Remove from *S* any hypothesis that is more general than another hypothesis in *S*

CANDIDATE-ELIMINATION Algorithm

/* Influence of +ve and −ve examples on S and G are dual */

- If d is a −ve example
  - Remove from S any hypothesis inconsistent with d
  - For each $g \in G$ not consistent with d
    - Remove g from G
    - Add to G all minimal specializations h of g s.t.

      h is consistent with d, and

      some member of S is more specific than or equal to h
    - Remove from G any hypothesis that is more specific than another hypothesis in G

CANDIDATE-ELIMINATION Trace 1

- Remove from *G* any hypothesis inconsistent with *d*
- For each $s \in S$ not consistent with *d*
  - Remove *s* from *S*
  - Add to *S* all minimal generalizations *h* of *s* s.t.

    h is consistent with *d*, and

    some member of *G* is more general than or equal to *h*

S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}

S_1: {<*Sunny, Warm, Normal, Strong, Warm, Same*>}

S_2: {<*Sunny, Warm, ?, Strong, Warm, Same*>}

G_0, G_1, G_2: {<*?, ?, ?, ?, ?, ?*>}

Training examples:

1 . <*Sunny, Warm, Normal, Strong, Warm, Same*>, *Enjoy Sport = Yes*

2 . <*Sunny, Warm, High, Strong, Warm, Same*>, *Enjoy Sport = Yes*

CANDIDATE-ELIMINATION Trace 2

- Remove from *S* any hypothesis inconsistent with *d*
- For each $g \in G$ not consistent with *d*
  - Remove *g* from *G*
  - Add to *G* all minimal specializations *h* of *g* s.t.

    h is consistent with *d*, and

    some member of *S* is more specific than or equal to *h*

S_2, S_3: { <*Sunny, Warm, ?, Strong, Warm, Same*> }

G_3: {<*Sunny, ?, ?, ?, ?, ?*> <*?, Warm, ?, ?, ?, ?*> <*?, ?, ?, ?, ?, Same*>}

G_2: {<*?, ?, ?, ?, ?, ?*> }

Training Example:

3. <*Rainy, Cold, High, Strong, Warm, Change*>, *EnjoySport=No*

CANDIDATE-ELIMINATION Trace 3

- Remove from *G* any hypothesis inconsistent with *d*
- For each $s \in S$ not consistent with *d*
  - Remove *s* from *S*
  - Add to *S* all minimal generalizations *h* of *s* s.t.

    h is consistent with *d*, and

    some member of *G* is more general than or equal to *h*

S_3: {<*Sunny, Warm, ?, Strong, Warm, Same*>}

S_4: {<*Sunny, Warm, ?, Strong, ?, ?*>}

G_4: {<*Sunny, ?, ?, ?, ?, ?*> <*?, Warm, ?, ?, ?, ?*>}

G_3: {<*Sunny, ?, ?, ?, ?, ?*> <*?, Warm, ?, ?, ?, ?*> <*?, ?, ?, ?, ?, Same*>}

Training Example:

4.<*Sunny, Warm, High, Strong, Cool, Change*>, *EnjoySport = Yes*