2 Concept Learning and the General-to-Specific Ordering

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

**Note:** This is a simple approach assuming no noise and illustrating key concepts.
A Concept Learning Task

- **Concept Learning** is the process of inferring a boolean-valued function from training examples of its input and output.

- **Example:** Target concept: “days on which Aldo enjoys his favorite water sport”.

- **Training Examples for EnjoySport:**

<table>
<thead>
<tr>
<th>Sky</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>EnjoySport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cold</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Change</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Cool</td>
<td>Change</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- **Possible values:**

<table>
<thead>
<tr>
<th>Sky</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>EnjoySport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Cloudy</td>
<td>Rainy</td>
<td>Sunny</td>
<td>Cool</td>
<td>Same</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Representing Hypotheses

- Many possible representations.
- Here, \( h \) is a conjunction of constraints on attributes.
- Each constraint can be either
  - a specific value (e.g., \( Water = Warm \)),
  - don’t care (e.g., \( Water = ? \)), or
  - no value allowed (e.g., \( Water = \emptyset \)).

- Hypotheses are represented as vectors, e.g.,

  \[
  \langle \text{Sky} \quad \text{AirTemp} \quad \text{Humid} \quad \text{Wind} \quad \text{Water} \quad \text{Forecast} \rangle
  \]
  \[
  \langle \text{Sunny} \quad ? \quad ? \quad \text{Strong} \quad ? \quad \text{Same} \rangle
  \]

- Most general hypothesis: \( \langle ?, ?, ?, ?, ?, ? \rangle \).
- Most specific hypothesis: \( \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle \).
Prototypical Concept Learning Task

Given:

- **Instances** $X$: Possible days, each described by the attributes $Sky$, $AirTemp$, $Humidity$, $Wind$, $Water$, $Forecast$
- **Target function** $c$: $EnjoySport : X \rightarrow \{0, 1\}$
- **Hypotheses** $H$: Conjunctions of literals. E.g.,
  $$\langle ?, Cold, High, ?, ?, ?, ? \rangle.$$  
- **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 $D$.

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.
Instances, Hypotheses, and the More-General-Than Ordering

**Instances X**

- $x_1 = \langle \text{Sunny, Warm, High, Strong, Cool, Same} \rangle$
- $x_2 = \langle \text{Sunny, Warm, High, Light, Warm, Same} \rangle$

**Hypotheses H**

- $h_1 = \langle \text{Sunny, ?, ?, Strong, ?, ?} \rangle$
- $h_2 = \langle \text{Sunny, ?, ?, ?, ?, ?} \rangle$
- $h_3 = \langle \text{Sunny, ?, ?, ?, Cool, ?} \rangle$
More-General-Than Ordering – Formal Definition

Let \( h_j \) and \( h_k \) be boolean-valued functions defined over \( X \).

\( h_j \) is more-general-than-or-equal-to \( h_k \) (written \( h_j \geq_{g} h_k \)) iff
\[
(\forall x \in X) \ (h_k(x) = 1 \rightarrow h_j(x) = 1).
\]

\( h_j \) is (strictly) more-general-than \( h_k \) (written \( h_j >_{g} h_k \)) iff
\[
h_j \geq_{g} h_k \land h_k \not\geq_{g} h_j.
\]

\( h_k \) is more-specific-than \( h_j \) iff \( h_j \) is more-general-than \( h_k \).

\( \geq_{g} \) is a partial ordering over \( H \),
i.e., it is reflexive, antisymmetric, transitive and some pairs may not be ordered.
**Find-S Algorithm**

- Initialize $h$ to be the most specific hypothesis in $H$.

- For each positive training instance $x$ do:
  - For each attribute constraint $a_i$ in $h$ do:
    - 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 $x$.

- Output hypothesis $h$. 
Hypothesis Space Search by *FIND-S*

\[
\begin{align*}
x_1 &= \langle \text{Sunny Warm Normal Strong Warm Same} \rangle, + \\
x_2 &= \langle \text{Sunny Warm High Strong Warm Same} \rangle, + \\
x_3 &= \langle \text{Rainy Cold High Strong Warm Change} \rangle, - \\
x_4 &= \langle \text{Sunny Warm High Strong Cool Change} \rangle, + \\
\end{align*}
\]

\[
\begin{align*}
h_0 &= \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle \\
h_1 &= \langle \text{Sunny Warm Normal Strong Warm Same} \rangle \\
h_2 &= \langle \text{Sunny Warm ? Strong Warm Same} \rangle \\
h_3 &= \langle \text{Sunny Warm ? Strong Warm Same} \rangle \\
h_4 &= \langle \text{Sunny Warm ? Strong ? ?} \rangle \\
\end{align*}
\]
Complaints about FIND-S

- Can’t tell whether it has learned a concept.
- Can’t tell when training data inconsistent.
- Picks a most specific $h$. Why?
- Depending on $H$, there might be several most specific hypothesis!
Complaints about FIND-S

- Can’t tell whether it has learned a concept.
- Can’t tell when training data inconsistent.
- Picks a most specific $h$. Why?
- Depending on $H$, there might be several most specific hypothesis!

$\Rightarrow$ Is it possible to describe all hypothesis consistent with the training data?

$\Rightarrow$ Version spaces and the candidate-elimination algorithm.
Version Spaces

**Idea:** Compute the set of all hypothesis consistent with the training examples.
Version Spaces

- **Idea:** Compute the set of all hypothesis consistent with the training examples.

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

  \[ \text{Consistent}(h, D) \equiv (\forall \langle x, c(x) \rangle \in D) \; h(x) = c(x) \]

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

  \[ V_{H,D} \equiv \{ h \in H \mid \text{Consistent}(h, D) \} \]
Version Spaces

▶ Idea: Compute the set of all hypothesis consistent with the training examples.

▶ hypothesis \( h \) is consistent with a set of training examples \( D \) of target concept \( c \) iff \( h(x) = c(x) \) for each training example \( \langle x, c(x) \rangle \) in \( D \).

\[
\text{Consistent}(h, D) \equiv (\forall \langle x, c(x) \rangle \in D) \ h(x) = c(x)
\]

▶ The version space \( V S_{H,D} \) with respect to hypothesis space \( H \) and training examples \( D \) is the subset of hypotheses from \( H \) consistent with all training examples in \( D \).

\[
V S_{H,D} \equiv \{ h \in H \mid \text{Consistent}(h, D) \}
\]

▶ How can we represent a version space?
The **List-Then-Eliminate Algorithm:**

- \( \text{VersionSpace} \leftarrow \text{a list containing every hypothesis in } H \).

- For each training example, \( \langle x, c(x) \rangle \) do:
  - remove from \( \text{VersionSpace} \) any hypothesis \( h \) for which \( h(x) \neq c(x) \).

- Output the list of hypotheses in \( \text{VersionSpace} \).
The **List-Then-Eliminate Algorithm**:  

- \( \text{VersionSpace} \leftarrow \) a list containing every hypothesis in \( H \).

- For each training example, \( \langle x, c(x) \rangle \) do:
  
  - remove from \( \text{VersionSpace} \) any hypothesis \( h \) for which \( h(x) \neq c(x) \).

- Output the list of hypotheses in \( \text{VersionSpace} \).

- We need to find a more compact representation.
Example Version Space

<table>
<thead>
<tr>
<th>Sky</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>EnjoySport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cold</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Change</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Cool</td>
<td>Change</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Its version space:

  \[
  S: \{ <\text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ?> \} \\
  \]
Representing Version Spaces

- The general boundary $G$ of version space $V S_{H,D}$ is the set of its most general members.
- The specific boundary $S$ of version space $V S_{H,D}$ is the set of its most specific members.
- Every member of the version space lies between these boundaries

$$V S_{H,D} = \{ h \in H \mid (\exists s \in S)(\exists g \in G)(g \geq_g h \geq_g s)\}$$

where $x \geq_g y$ means $x$ is more general or equal to $y$. 
Candidate Elimination Algorithm

- Initialize $G$ to be the most general hypotheses in $H$.
  Initialize $S$ to be the most 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
        - $h$ is consistent with $d$, and
        - some member of $G$ is more general than $h$
      - Remove from $S$ any hypothesis that is more general than another hypothesis in $S$
Candidate Elimination Algorithm – Continued

► Remember: For each training example $d$ do:

▷ If $d$ is a negative example

- 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
    - $h$ is consistent with $d$, and
    - 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

- Final version space:

\[ S: \{ <\text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ?> \} \]


- How should new instances be classified?
Example Trace

Final version space:


How should new instances be classified?

Example Trace

Final version space:

\[ S: \{ <\text{Sunny, Warm, ?, Strong, ?, ?> > \} \]

\[ G: \{ <\text{Sunny, ?, ?, ?, ?, ?> >, <\text{?, Warm, ?, ?, ?, ?> > \} \]

How should new instances be classified?

- \( <\text{Sunny, Warm, Normal, Strong, Cool, Change}> \)
- \( <\text{Rainy, Cold, Normal, Light, Warm, Same}> \)
Example Trace

Final version space:

\[
S: \{ <\text{Sunny, Warm, ?, Strong, ?, ?> > \}
\]

\[
\]

How should new instances be classified?

\[
\langle \text{Sunny, Warm, Normal, Strong, Cool, Change} \rangle
\]

\[
\langle \text{Rainy, Cold, Normal, Light, Warm, Same} \rangle
\]

\[
\langle \text{Sunny, Warm, Normal, Light, Warm, Same} \rangle
\]
An Un-Biased Learner

- **Idea:** Choose $H$ that expresses every teachable concept (i.e., $H$ is the power set of $X$).
- Consider $H' = \text{disjunctions, conjunctions, negations over previous } H$. E.g.,

$$\langle \text{Sunny, Warm, Normal, ?, ?, ?} \rangle \lor \neg \langle ?, ?, ?, ?, ?, ?, \text{Change} \rangle$$

- What are $S, G$ in this case?
An Un-Biased Learner

► **Idea:** Choose $H$ that expresses every teachable concept (i.e., $H$ is the power set of $X$).

► Consider $H' =$ disjunctions, conjunctions, negations over previous $H$. E.g.,

$$\langle Sunny, Warm, Normal, ?, ?, ? \rangle \lor \neg \langle ?, ?, ?, ?, ?, ?, Change \rangle$$

► What are $S$, $G$ in this case?

► $S$ will consist of all positive examples.

► $G$ will consist of all negative examples.
An Un-Biased Learner

- **Idea:** Choose $H$ that expresses every teachable concept (i.e., $H$ is the power set of $X$).
- Consider $H' = \text{disjunctions, conjunctions, negations over previous } H$. E.g.,

$$\langle \text{Sunny, Warm, Normal, ?}, \text{?}, \text{?}, \text{?} \rangle \lor \neg \langle \text{?, ?}, \text{?, ?}, \text{?, ?}, \text{Change} \rangle$$

- What are $S, G$ in this case?
  - $S$ will consist of all positive examples.
  - $G$ will consist of all negative examples.

$\rightsquigarrow$ No generalization.
Inductive Bias

- Consider
  - concept learning algorithm $L$,
  - instances $X$, target concept $c$,
  - training examples $D_c = \{ (x_j, c(x_j)) \mid 1 \leq j \leq n \}$ and
  - let $L(x, D_c)$ denote the classification assigned to the instance $x$ 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 $D_c$

  $$(\forall x_i \in X)[(B \land D_c \land x_i)] \models L(x_i, D_c).$$
Inductive Systems and Equivalent Deductive Systems

Inductive system

Training examples

Candidate Elimination Algorithm Using Hypothesis Space $H$

New instance

Classification of new instance, or "don’t know"

Equivalent deductive system

Training examples

Theorem Prover

New instance

Classification of new instance, or "don’t know"

Assertion "$H$ contains the target concept"

Inductive bias made explicit
Three Learners with Different Biases

▸ Rote learner:
  Store examples, classify \( x \) iff it matches previously observed example.
  ▸ No inductive bias.

▸ Version space candidate elimination algorithm
  ▸ Inductive bias: Target concept can be represented in its hypothesis space.

▸ Find-S
  ▸ Inductive bias: Target concept can be represented in its hypothesis space and all instances are negative instances unless the opposite is entailed by its other knowledge.
Summary Points

- Concept learning as search through $H$.
- General-to-specific ordering over $H$.
- Version space candidate elimination algorithm.
- $S$ and $G$ boundaries characterize learner’s uncertainty.
- Learner can generate useful queries.
- Inductive leaps possible only if learner is biased.
- Inductive learners can be modelled by equivalent deductive systems.
Exercises

▶ Exercise 1.3 in Mitchell’s book.
▶ Consider a learning scenario where each object (example) is described by a list of $n$ attributes, each of which can take $k$ possible discrete values.

▷ What is the size of the instance space?
▷ What is the size of the concept space?
▷ What is the size of the hypothesis space in the following concept representation languages:

- Purely conjunctive expressions of conditions “attribute = value” where only those conditions appear which are required, e.g.,

  $$C \leftrightarrow (A2 = \text{small}) \land (A5 = \text{high}).$$

- Purely conjunctive expressions of conditions “attribute in set”, e.g.,

  $$C \leftrightarrow A2 \in \{\text{small, medium}\} \land A5 \in \{\text{red, blue, yellow}\}.$$  

- Unrestricted disjunctive normal form.
Exercises – Continued

► Exercise 2.2 in Mitchell’s book.
► Exercise 2.4 in Mitchell’s book.