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.
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 my friend 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>Sunny, Cloudy, Rainy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp</td>
<td>Warm, Cold</td>
</tr>
<tr>
<td>Humid</td>
<td>Normal, High</td>
</tr>
<tr>
<td>Wind</td>
<td>Strong, Weak</td>
</tr>
<tr>
<td>Water</td>
<td>Warm, Cool</td>
</tr>
<tr>
<td>Forecast</td>
<td>Same, Change</td>
</tr>
</tbody>
</table>
Representing Hypotheses

- Many possible representations.
- Here, $h$ is a 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., $Water = \emptyset$).
- For example,
  
<table>
<thead>
<tr>
<th>Sky</th>
<th>AirTemp</th>
<th>Humid</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>?</td>
<td>?</td>
<td>Strong</td>
<td>?</td>
<td>Same</td>
</tr>
</tbody>
</table>

- Most general hypothesis: $\langle?,?,?,?,?\rangle$.
- Most specific hypothesis: $\langle\emptyset,\emptyset,\emptyset,\emptyset,\emptyset\rangle$. 
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.,
  $$\langle ?, Cool, High, ?, ?, ? \rangle.$$
- 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

\[ x_1 = \langle \text{Sunny, Warm, High, Strong, Cool, Same} \rangle \]
\[ x_2 = \langle \text{Sunny, Warm, High, Light, Warm, Same} \rangle \]
\[ h_1 = \langle \text{Sunny, ?, ?, Strong, ?, ?} \rangle \]
\[ h_2 = \langle \text{Sunny, ?, ?, ?, ?, ?} \rangle \]
\[ h_3 = \langle \text{Sunny, ?, ?, ?, Cool, ?} \rangle \]
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 not all pairs are 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

Instances $X$

Hypotheses $H$

$x_1 = <\text{Sunny Warm Normal Strong Warm Same}>, +$

$x_2 = <\text{Sunny Warm High Strong Warm Same}>, +$

$x_3 = <\text{Rainy Cold High Strong Warm Change}>, -$\n
$x_4 = <\text{Sunny Warm High Strong Cool Change}>, +$

$h_0 = <\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset>$

$h_1 = <\text{Sunny Warm Normal Strong Warm Same}>$

$h_2 = <\text{Sunny Warm ? Strong Warm Same}>$

$h_3 = <\text{Sunny Warm ? Strong Warm Same}>$

$h_4 = <\text{Sunny Warm ? Strong ? ?}>
Complaints about \textsc{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 \textsc{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!

- Is it possible to describe all hypothesis consistent with the training data?
- 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.

- A 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 $D$.

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

- 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 $D$.

  $$VS_{H,D} \equiv \{ h \in H \mid Consistent(h, D) \}$$
Version Spaces

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

► A 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:**

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

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

- Output the list of hypotheses in $VersionSpace$. 
The **List-Then-Eliminate Algorithm:**

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

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

- Output the list of hypotheses in $VersionSpace$.

- 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

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

  where $x \geq y$ 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:

\[
S: \{ \langle Sunny, Warm, ?, Strong, ?, ? \rangle \} 
\]

\[
G: \{ \langle Sunny, ?, ?, ?, ?, ? \rangle, \langle ?, Warm, ?, ?, ?, ? \rangle \} 
\]

- How should new instances be classified?
Example Trace

- Final version space:

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


- How should new instances be classified?

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

Final version space:

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


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

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

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

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

- How should new instances be classified?
  
  - \( <\text{Sunny, Warm, Normal, ?}, \text{Strong, ?}, ?> \)
  
  - \( <?, \text{Warm, ?}, \text{Strong, ?}, ?> \)

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

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

- 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 $H' = \text{disjunctions, conjunctions, negations over previous } H$. E.g.,

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

▶ What are $S, G$ in this case?

▷ $S$ 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 $H' =$ 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.

- No generalization.
Inductive Bias

Consider

- concept learning algorithm $L$
- instances $X$, target concept $c$
- training examples $D_c = \{\langle x_j, c(x_j) \rangle | 1 \leq j \leq n\}$
- 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
- New instance

Candidate Elimination Algorithm
Using Hypothesis Space $H$

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

Equivalent deductive system

- Training examples
- New instance
- Assertion "$H$ contains the target concept"

Theorem Prover

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

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.