Outline

[read Chapter 2]
[suggested exercises 2.2, 2.3, 2.4, 2.6]

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

Note: simple approach assuming no noise, illustrates key concepts

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>EnjoySpt</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>

What is the general concept?

Representing Hypotheses

Many possible representations

Here, $h$ is 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=”) 

For example,

Sky   AirTemp  Humid  Wind  Water  Forecast
\langle Sunny ?  ?  Strong  ?  Same \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.

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
Complaints about Find-S

- Can’t tell whether it has learned concept
- Can’t tell when training data inconsistent
- Picks a maximally specific \( h \) (why?)
- 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 \( (x, c(x)) \) in \( D \).

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

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 \( D \).

\[
VS_{H,D} \equiv \{ h \in H | \text{Consistent}(h, D) \}
\]

The List-Then-Eliminate **Algorithm**:

1. \( VersionSpace \leftarrow \) a list containing every hypothesis in \( H \)
2. For each training example, \( (x, c(x)) \)
   remove from \( VersionSpace \) any hypothesis \( h \) for which \( h(x) \neq c(x) \)
3. Output the list of hypotheses in \( VersionSpace \)

Example Version Space

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

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

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

where \( x \geq y \) means \( x \) is more general or equal to \( y \).

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 \( S \) is more specific than \( h \)
    * Remove from \( G \) any hypothesis that is less general than another hypothesis in \( G \)

- If \( d \) is a negative example

Example Trace

\[ S_0: \{\emptyset, \emptyset, \emptyset, \emptyset, \emptyset\} \]

\[ G_0: \{?, ?, ?, ?, ?\} \]
What Next Training Example?

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

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

How Should These Be Classified?

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

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

\text{<Sunny Warm Normal Strong Cool Change>}

\text{<Rainy Cool Normal Light Warm Same>}

\text{<Sunny Warm Normal Light Warm Same>}

What Justifies this Inductive Leap?

\[ + \langle \text{Sunny Warm Normal Strong Cool Change} \rangle \\
+ \langle \text{Sunny Warm Normal Light Warm Same} \rangle \\
\]

\[ S: \langle \text{Sunny Warm Normal ? ? ?} \rangle \]

Why believe we can classify the unseen

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

An UNBiased 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?

\[ S \leftarrow \]

\[ G \leftarrow \]
Inductive Bias

Consider

- concept learning algorithm $L$
- instances $X$, target concept $c$
- training examples $D_c = \{(x, c(x))\}$
- 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 $D_c$

$$(\forall x \in X)[(B \land D_c \land x) \vdash L(x, D_c)]$$

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

---

Inductive Systems and Equivalent Deductive Systems

Three Learners with Different Biases

1. **Rote learner**: Store examples, Classify $x$ if it matches previously observed example.
2. **Version space candidate elimination algorithm**
3. **Find-S**

Summary Points

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 modelled by equivalent deductive systems