CMSC726 Spring 2006: Concept Learning (aka Logical Learning 101)

readings: Mitchell, Ch. 2
sources: course slides are based on material from a variety of sources, including Tom Dietterich, Rich Maclin, Ray Mooney, Andrew Moore, Andrew Ng, Jude Shavlik, and others.

<table>
<thead>
<tr>
<th>Concept Learning Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning from examples</td>
</tr>
<tr>
<td>General-to-specific ordering over hypotheses</td>
</tr>
<tr>
<td>Version Spaces and candidate elimination algorithm</td>
</tr>
<tr>
<td>Picking new examples</td>
</tr>
<tr>
<td>The need for inductive bias</td>
</tr>
</tbody>
</table>
Some Examples for SmileyFaces

<table>
<thead>
<tr>
<th>Eyes</th>
<th>Nose</th>
<th>Head</th>
<th>Fcolor</th>
<th>Hair?</th>
<th>Smile?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>Purple</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>Green</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>Yellow</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>Green</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>Yellow</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Features from Computer View

<table>
<thead>
<tr>
<th>Eyes</th>
<th>Nose</th>
<th>Head</th>
<th>Fcolor</th>
<th>Hair?</th>
<th>Smile?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round</td>
<td>Triangle</td>
<td>Round</td>
<td>Purple</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Square</td>
<td>Square</td>
<td>Square</td>
<td>Green</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Square</td>
<td>Triangle</td>
<td>Round</td>
<td>Yellow</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Round</td>
<td>Triangle</td>
<td>Round</td>
<td>Green</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Square</td>
<td>Square</td>
<td>Round</td>
<td>Yellow</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Representing Hypotheses

Many possible representations for hypotheses $h$

Idea: $h$ as conjunctions of constraints on features

Each constraint can be:
- a specific value (e.g., $Nose = Square$)
- don’t care (e.g., $Eyes = ?$)
- no value allowed (e.g., $Water = \emptyset$)

For example,

$Eyes$ $Nose$ $Head$ $Fcolor$ $Hair$?

$<Round, ?, Round, ?, No>$

Prototypical Concept Learning Task

Given:
- Instances $X$: Faces, each described by the attributes $Eyes$, $Nose$, $Head$, $Fcolor$, and $Hair$?
- Target function $c$: $Smile?: X \rightarrow \{ \text{no, yes} \}$
- Hypotheses $H$: Conjunctions of literals such as $<x_1, c(x_1)>, <x_2, c(x_2)>, ..., <x_m, c(x_m)>
  <?, Square, Square, Yellow, ?>$
- Training examples $D$: Positive and negative examples of the target function
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.

- What are the implications?
- Is this reasonable?
- What (if any) are our alternatives?
- What about concept drift (what if our views/tastes change over time)?
**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 next more general constraint satisfied by \( x \)

3. Output hypothesis \( h \)

---

**Hypothesis Space Search by Find-S**

![Diagram showing instances and hypotheses](attachment:image.png)

Instances \( X \) and Hypotheses \( H \) are depicted with a hierarchy of specific to general hypotheses, illustrating the search process through positive training instances.
Complaints about Find-S

- Cannot tell whether it has learned concept
- Cannot tell when training data inconsistent
- Picks a maximally specific $h$ (why?)
- Depending on $H$, there might be several!

How do we fix this?

The List-Then-Eliminate Algorithm

1. Set $VersionSpace$ equal to 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 $VersionSpace$

- But is listing all hypotheses reasonable?
- How many different hypotheses in our simple problem?
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 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 $D$.

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

The **General boundary**, $G$, of version space $V_{S_{H,D}}$ is the set of its maximally general members.

The **Specific boundary**, $S$, of version space $V_{S_{H,D}}$ is the set of its maximally specific members.

Every member of the version space lies between these boundaries

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

where $x \geq y$ means $x$ is more general or equal to $y$

---

Candidate Elimination Algorithm

$G = \text{maximally general hypotheses in } H$
$S = \text{maximally specific hypotheses in } H$

For each training example $d$, do

**If $d$ is a positive example**

Remove from $G$ any hypothesis that does not include $d$
For each hypothesis $s$ in $S$ that does not include $d$
    Remove $s$ from $S$
    Add to $S$ all minimal generalizations $h$ of $s$ such that
    1. $h$ includes $d$, and
    2. Some member of $G$ is more general than $h$
Remove from $S$ any hypothesis that is more general than another hypothesis in $S$
Candidate Elimination Algorithm (cont)

For each training example \( d \), do (cont)

If \( d \) is a negative example
- Remove from \( S \) any hypothesis that does include \( d \)
- For each hypothesis \( g \) in \( G \) that does include \( d \)
  - Remove \( g \) from \( G \)
  - Add to \( G \) all minimal generalizations \( h \) of \( g \) such that
    1. \( h \) does not include \( d \), 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 \( G \) or \( S \) ever becomes empty, data not consistent (with \( H \))

Example Trace

\[ S_0 : \{ <\emptyset,\emptyset,\emptyset,\emptyset,\emptyset> \} \]
\[ G_0 : \{ <?,?,?,?,?> \} \]
\[ X_1 = <R,T,R,P,Y> + \]
\[ S_1 : \{ <R,T,R,P,Y> \} \]
\[ G_1 : \{ <?,?,?,?,?> \} \]

\[ G_2 : \{ <R,?,?,?,?>, <?,?,?,?,?>, <?,?,?,?,?>, <?,?,?,?,?> \} \]
\[ X_2 = <S,S,G,Y> - \]
\[ S_2 : \{ <S,S,R,Y,Y> \} \]
\[ G_3 : \{ <?,T,?,?,?>, <?,?,R,?,?>, <?,?,?,P,?> \} \]

\[ S_3 : \{ <?,T,R,?,Y> \} \]
\[ S_4 : \{ <?,T,?,Y> \} \]
\[ X_3 = <S,T,R,Y,> + \]

\[ S_5 : \{ <?,?,R,?,Y> \} \]
\[ X_4 = <R,T,R,G,N> - \]
\[ S_5 = <S,S,R,Y,Y> + \]

\[ S_5 \]
| What Training Example Next? |

G:  \{ <?,?,Round,?,?> <?,Triangle,?,?,?> \}

S:  \{ <?,Triangle,?,Round,?,Yes> \}

| How Should These Be Classified? |

G:  \{ <?,?,Round,?,?> <?,Triangle,?,?,?> \}

S:  \{ <?,Triangle,?,Round,?,Yes> \}
What Justifies this Inductive Leap?

+ < Round, Triangle, Round, Purple, Yes >
+ < Square, Triangle, Round, Yellow, Yes >

S: < ?, Triangle, Round, ?, Yes >

Why believe we can classify the unseen?
< Square, Triangle, Round, Purple, Yes > ?

An UN-Biased Learner

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

Consider $H' = \text{disjunctions, conjunctions, negations over previous } H$.
For example:

$< ?, \text{Triangle, Round, ?}, \text{Yes} > \lor < \text{Square, Square, ?}, \text{Purple}, ? >$

What are S, G, in this case?
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 \( D_c \)
\[
(\forall x_i \in X)[(B \land D_c \land x_i) \vdash L(x_i, D_c)]
\]
where \( A \vdash B \) means \( A \) logically entails \( B \)

Inductive Systems and Equivalent Deductive Systems

<table>
<thead>
<tr>
<th>Inductive System</th>
<th>Equivalent Deductive System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training examples</td>
<td>Classification of new instance, or &quot;don't know&quot;</td>
</tr>
<tr>
<td>New instance</td>
<td>Using Hypothesis Space ( H )</td>
</tr>
<tr>
<td>Candidate Elimination Algorithm</td>
<td>Classification of new instance, or &quot;don't know&quot;</td>
</tr>
<tr>
<td>Using Hypothesis Space ( H )</td>
<td>Theorem Prover</td>
</tr>
<tr>
<td>Assertion &quot;( H ) contains hypothesis&quot;</td>
<td>Classification of new instance, or &quot;don't know&quot;</td>
</tr>
</tbody>
</table>
Three Learners w/ Different Biases

1. *Rote learner*: store examples, classify new instance iff it matches previously observed example (don’t know otherwise).

2. *Version space candidate elimination algorithm*.

3. *Find-S*

Summary

- **Key Concepts for Logical Learners**
  - Concept learning as search through hypothesis space to find hypothesis consistent with training set
  - Definition of **Hypothesis Space** and general-to-specific partial ordering over hypothesis
  - Focus on **Consistency**
  - Use of (declarative) **Inductive Bias**