Overview

- Concept Learning
- Representation
- Inductive Learning Hypothesis
- Concept Learning as Search
- The Structure of the Hypothesis Space
- Find-S Algorithm
- Version Space
- List-Eliminate Algorithm
- Candidate-Elimination Algorithm
- Using Partially Learned Concepts
- Inductive Bias
- An Unbiased Learner
- The Futility of Bias-Free Learning
- Inductive and Equivalent Deductive Systems
- Three Learners with Different Bias
The Example

Windsurfing can be fun, but this depends on the weather. The following data is collected:

<table>
<thead>
<tr>
<th>Sky</th>
<th>AirTemp</th>
<th>Humidity</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>

Is there a pattern here - what is the general concept?
Concept Learning

*Concept* = boolean-valued function defined over a large set of objects or events.

*Concept learning* = Inferring a boolean-valued function from training examples of input and output of the function.

How can we represent a concept?
Representation (1/2)

Many possible representations, but here:

**Instance** $x$: A collection of attributes ($Sky$, $AirTemp$, $Humidity$, etc.)

**Target function** $c$: $EnjoySport : X \rightarrow \{0,1\}$

**Hypothesis** $h$: A conjunction of constraints on the attributes.
A constraint can be:
- a specific value (e.g. $Water = Warm$)
- don’t care (e.g. $Water = ?$)
- no value allowed (e.g. $Water = \emptyset$)

**Training example** $d$: An instance $x_i$ paired with the target function $c$, $\langle x_i, c(x_i) \rangle$

$c(x_i) = 0$ negative example

$c(x_i) = 1$ positive example
Instances $X$: The set of possible instances $x$.

Hypotheses $H$: The set of possible hypotheses $h$.

Training examples $D$: The set of possible training examples $d$. 

$x = \text{instance}$

$h = \text{hypothesis}$

$d = \text{training example}$
Inductive Learning Hypothesis

Thus, concept learning is to find a \( h \in H \) so that \( h(x) = c(x) \) for all \( x \in X \).

**Problem:** We might not have access to all \( x \)!

**Solution:** Find a \( h \) that fits the examples, and take the achieved function as an approximation of the true \( c(x) \).

This principle is called **induction**.

*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.
Concept Learning as Search

Concept learning is to find a $h \in H$ so that $h(x) = c(x)$ for all $x \in X$.

A search problem: search through the hypothesis space $H$ after the best $h$. 

Concept learning = Inferring a boolean-valued function from training examples of input and output of the function.

$\in$ - "member of"

$x$ = instance

$h$ = hypothesis

$d$ = training example

$c$ = target function
Many algorithms for concept learning organize the search by using a useful structure in the hypothesis space: **general-to-specific ordering**!

An example:

\[ h_1 = \langle \text{Sunny}, ?, ?, \text{Strong}, ?, ? \rangle \]
\[ h_2 = \langle \text{Sunny}, ?, ?, ?, ?, ? \rangle \]
\[ h_3 = \langle \text{Sunny}, ?, ?, ?, \text{Cool}, ? \rangle \]

Because \( h_2 \) imposes fewer constraints, it will cover more instances \( x \) in \( X \) than both \( h_1 \) and \( h_3 \).

\( h_2 \) is more **general** than \( h_1 \) and \( h_3 \).

In a similar manner, we can define more-specific-than.
Find-S Algorithm

Finds the most specific hypothesis matching the training example (hence the name).

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 $h$

$x = \text{instance}$
$h = \text{hypothesis}$
$d = \text{training example}$
$c = \text{target function}$
Find-S algorithm example

\[ h \leftarrow \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle \]
\[ x = \langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle \]
\[ h \leftarrow \langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle \]
\[ x = \langle \text{Sunny, Warm, High, Strong, Warm, Same} \rangle \]
\[ h \leftarrow \langle \text{Sunny, Warm, ?, Strong, Warm, Same} \rangle \]
\[ x = \langle \text{Rainy, Cold, High, Strong, Warm, Change} \rangle \]
\[ h \leftarrow \langle \text{Sunny, Warm, ?, Strong, Warm, Same} \rangle \]
\[ x = \langle \text{Sunny, Warm, High, Strong, Cool, Change} \rangle \]
\[ h \leftarrow \langle \text{Sunny, Warm, ?, Strong, ?, ?} \rangle \]

- \( x \) = instance
- \( h \) = hypothesis
- \( d \) = training example
- \( c \) = target function

Concept Learning as...
The Structure of \( H \)
Find-S Algorithm
Version Space
List-Eliminate...
Problems with Find-S

There are several problems with the Find-S algorithm:

It cannot determine if it has learnt the concept.
There might be several other hypotheses that match as well – has it found the only one?

It cannot detect when training data is inconsistent.
We would like to detect and be tolerant to errors and noise.

Why do we want the most specific hypothesis?
Some other hypothesis might be more useful.

Depending on $H$, there might be several maximally consistent hypotheses, and there is no way for Find-S to find them. All of them are equally likely.
Definition: A hypothesis $h$ is consistent with a set of training examples $D$ if and only if $h(x) = c(x)$ for each example $\langle x, c(x) \rangle$ in $D$.

Version space = the subset of all hypotheses in $H$ consistent with the training examples $D$.

Definition: The version space, denoted $VS_{H,D}$, with respect to hypothesis space $H$ and training examples $D$, is the subset of hypotheses from $H$ consistent with the training examples in $D$. 
Representation of Version Space

**Option 1:** List all of the members in the version space. Works only when the hypothesis space $H$ is finite!

**Option 2:** Store only the set of most general members $G$ and the set of most specific members $S$. Given these two sets, it is possible to generate any member of the version space as needed.

$x =$ instance  
$h =$ hypothesis  
$d =$ training example  
$c =$ target function  
$\text{VS}_{H,D} =$ version space

The Structure of $H$  
Find-S Algorithm  
Version Space  
List-Eliminate...  
Candidate-Elimination...
List-Eliminate Algorithm

1. $VS_{H,D} \leftarrow$ a list containing every hypothesis in $H$
2. For each training example, $\langle x, c(x) \rangle$
   remove from $VS_{H,D}$ any hypothesis that is inconsistent with the training example $h(x)?c(x)$
3. Output the list of hypotheses in $VS_{H,D}$

**Advantage:** Guaranteed to output all hypotheses consistent with the training examples.

But **inefficient**! Even in this simple example, there are $1+4\cdot3\cdot3\cdot3\cdot3 = 973$ semantically distinct hypotheses.
Candidate-Elimination Algorithm

\[ G \leftarrow \text{maximally general hypothesis in } H \]
\[ S \leftarrow \text{maximally specific hypothesis in } H \]

For each training example \( d = \langle x, c(x) \rangle \)

modify \( G \) and \( S \) so that \( G \) and \( S \) are consistent with \( d \)

\( x = \text{instance} \)
\( h = \text{hypothesis} \)
\( d = \text{training example} \)
\( c = \text{target function} \)
\( \text{VS}_{H,D} = \text{version space} \)
Candidate-Elimination Algorithm (detailed)

\[ G \leftarrow \text{maximally general hypotheses in } H \]
\[ S \leftarrow \text{maximally specific hypotheses in } H \]

For each training example \( d = \langle x, c(x) \rangle \)

If \( d \) is a positive example
- Remove from \( G \) any hypothesis that is 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
    - \( h \) consistent with \( d \)
    - Some member of \( G \) is more general than \( h \)
- Remove from \( S \) any hypothesis that is more general than another hypothesis in \( S \)

If \( d \) is a negative example
- Remove from \( S \) any hypothesis that is 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
    - \( h \) consistent with \( d \)
    - Some member of \( S \) is more specific than \( h \)
- Remove from \( G \) any hypothesis that is less general than another hypothesis in \( G \)

\( x = \text{instance} \)
\( h = \text{hypothesis} \)
\( d = \text{training example} \)
\( c = \text{target function} \)
\( \text{VS}_{H,D} = \text{version space} \)
Example of Candidate-Elimination

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

\[
G : \{?, ?, ?, ?, ?\}
\]

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

\[
S : \{\text{Sunny, Warm, Normal, Strong, Warm, Same}\}
\]

\[
G : \{?, ?, ?, ?, ?\}
\]

\[
x = \langle \text{Sunny, Warm, High, Strong, Warm, Same} \rangle
\]

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

\[
G : \{?, ?, ?, ?, ?\}
\]

\[
x = \langle \text{Rainy, Cold, High, Strong, Warm, Change} \rangle
\]

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

\[
\]

\[
x = \langle \text{Sunny, Warm, High, Strong, Cool, Change} \rangle
\]

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

\[
\]
Properties of Candidate-Elimination

Converges to target concept when
  • No error in training examples
  • Target concept is in $H$

Converges to an empty version space when
  • Inconsistency in training data
  • Target concept cannot be described by hypothesis representation

$x = \text{instance}$
$h = \text{hypothesis}$
$d = \text{training example}$
$c = \text{target function}$
$V S_{H,D} = \text{version space}$
$G = \text{set of most general members in } H$
$S = \text{set of most specific members in } H$
Using Partially Learned Concepts

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

\[ \langle Sunny, ?, ?, Strong, ?, ? \rangle \quad \langle Sunny, Warm, ?, ?, ?, ? \rangle \quad \langle ?, Warm, ?, Strong, ?, ? \rangle \]

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

<table>
<thead>
<tr>
<th>New instance</th>
<th>Votes: Pos</th>
<th>Neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = \langle Sunny, Warm, Normal, Strong, Cool, Change \rangle )</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>( x = \langle Rainy, Cold, Normal, Light, Warm, Same \rangle )</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>( x = \langle Sunny, Warm, Normal, Light, Warm, Same \rangle )</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( x = \langle Sunny, Cold, Normal, Strong, Warm, Same \rangle )</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

\( x = \text{instance} \)
\( h = \text{hypothesis} \)
\( d = \text{training example} \)
\( c = \text{target function} \)
\( \text{VS}_{H,D} = \text{version space} \)
\( G = \text{set of most general members in H} \)
\( S = \text{set of most specific members in H} \)
Inductive Bias

New example: Each instance is a single integer [1,4]

Training examples:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

There is no way to represent this simple example by using our representation.

Why?

The hypotheses space $H$ is biased, since it only allow conjunctions of attribute values.
An Unbiased Learner

Solution(?) : Let $H$ be the set of all possible subsets of $X$ – the power set of $X$ – denoted as $P(X)$. It contains $2^{|X|}$ hypotheses, where, $|X|$ is the number of distinct instances.

For our example, where each instance is a single integer $[1,4]$ :

$$P(X) = \begin{cases} \langle \emptyset \rangle, \\ \langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle, \\ \langle 1,2 \rangle, \langle 1,3 \rangle, \langle 1,4 \rangle, \langle 2,3 \rangle, \langle 2,4 \rangle, \langle 3,4 \rangle, \\ \langle 1,2,3 \rangle, \langle 1,2,4 \rangle, \langle 1,3,4 \rangle, \langle 2,3,4 \rangle, \\ \langle 1,2,3,4 \rangle \end{cases}$$
The Futility of Bias-free Learning

For our example, we get after three training examples:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

$S : \{1,3\}$

$G : \{\neg 2 = \{1,3,4\}\}$

How do we classify a new instance $x=4$?

The only examples that are classified are the training examples themselves. In order to learn the target concept one would have to present every single instance in $X$ as a training example!

Each unobserved instance will be classified positive by precisely half the hypotheses in $VS_{H,D}$ and negative by the other half.
Inductive Bias Revisited

Inductive bias is necessary in order to be able to generalize!

Definition: The inductive bias of $L$ is any minimal set of assertions $B$ such that for any target concept $c$ and corresponding training data $D$

$$(\forall x \in X) [(B \land D \land x) \Rightarrow L(x, D)]$$

Where $y \Rightarrow z$ means that $z$ deductively follows from $y$. 
Inductive and equivalent deductive systems

Inductive system

Candidate Elimination Algorithm

Using Hypothesis Space $H$

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

Training examples

New instance

Equivalent deductive system

Theorem Prover

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

Training examples

New instance

Assertion "$H$ contains the target concept"

Inductive bias made explicit

$x = \text{instance}$

$h = \text{hypothesis}$

$d = \text{training example}$

$c = \text{target function}$

$V S_{H,D} = \text{version space}$

$G = \text{set of most general members in } H$

$S = \text{set of most specific members in } H$

The Futility of...

Inductive Bias Revisited

Inductive and ...

Three Learners...

Summary
Three Learners with different Biases

1. Rote learner – simply stores instances
   No bias – unable to generalize

2. Candidate-Elimination
   Bias: Target concept is in hypothesis space $H$

3. Find-S
   Bias: Target concept is in hypothesis space $H$
   +
   All instances are negative unless taught otherwise

$x = \text{instance}$
$h = \text{hypothesis}$
$d = \text{training example}$
$c = \text{target function}$
$VS_{H,D} = \text{version space}$
$G = \text{set of most general members in } H$
$S = \text{set of most specific members in } H$
• Concept learning as search through $H$

• Many algorithms use general-to-specific ordering of $H$

• Find-S algorithm

• Version space and version space representations

• List-Eliminate algorithm

• Candidate-Elimination algorithm

• Inductive leaps possible only if learner is biased

• Inductive learners can be modelled by equivalent deductive systems