Lecture 2: Concept Learning
Version Space, Candidate Elimination, Inductive Bias

Ute Schmid

Cognitive Systems, Applied Computer Science, University of Bamberg
www.uni-bamberg.de/cogsys

Last change: October 14, 2019
Outline

- Definition of concept learning
- FIND-S
- Version Spaces
- Candidate Elimination
- Inductive Bias
- Summary
Definition of Concept Learning

- **Learning** involves acquiring general concepts from a specific set of training examples $D$

- Each **concept** $c$ can be thought of as a boolean-valued function defined over a larger set (i.e. a function defined over all animals, whose value is true for birds and false for other animals)

$\Rightarrow$ **Concept learning**: Inferring a boolean-valued function from training examples
A Concept Learning Task - Informal

- example target concept

  *Enjoy*: “*days on which Aldo enjoys his favorite sport*”

- set of example days $D$, each represented by a set of attributes

<table>
<thead>
<tr>
<th>Example</th>
<th>Sky</th>
<th>AirTemp</th>
<th>Humidity</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>Enjoy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Rainy</td>
<td>Cold</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Change</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Cool</td>
<td>Change</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- the task is to learn to predict the value of *Enjoy* for an arbitrary day, based on the values of its other attributes
A Concept Learning Task - Informal

Hypothesis representation

- Each hypothesis \( h \) consists of a **conjunction of constraints on the instance attributes**, that is, in this case a vector of six attributes

- Possible constraints:
  - \( ? \): any value is acceptable
  - \( \text{single required value} \): single required value for the attribute
  - \( \emptyset \): no value is acceptable

- if some instance \( x \) satisfies all the constraints of hypothesis \( h \), then \( h \) classifies \( x \) as a positive example \( (h(x) = 1) \)

  - **most general** hypothesis: \(<?, ?, ?, ?, ?, ?>\)
  - **most specific** hypothesis: \(<\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset>\)
A Concept Learning Task - Formal

Given:

- Instances $X$: Possible days, each described by the attributes
  - $Sky$ (with values Sunny, Cloudy and Rainy)
  - $AirTemp$ (with values Warm and Cold)
  - $Humidity$ (with values Normal and High)
  - $Wind$ (with values Strong and Weak)
  - $Water$ (with values Warm and Cool)
  - $Forecast$ (with values Same and Change)

- Hypotheses $H$ where each $h \in H$ is described as a conjunction of constraints on the above attributes

- Target Concept $c : Enjoy : X \Rightarrow \{0, 1\}$

- Training examples $D$: positive and negative examples of the table above

Determine:

- A hypothesis $h \in H$ such that $(\forall x \in X)[h(x) = c(x)]$
A Concept Learning Task - Example

- **example hypothesis** \( h_e = \langle Sunny, ?, ?, ?, ?, Warm, ? \rangle \)

  \( \Rightarrow \) According to \( h_e \) Aldo enjoys his favorite sport whenever the *sky* is sunny and the *water* is warm (independent of the other weather conditions!)

---

**example 1:** \( \langle Sunny, Warm, Normal, Strong, Warm, Same \rangle \)

This example satisfies \( h_e \), because the *sky* is sunny and the *water* is warm. Hence, Aldo would enjoy his favorite sport on this day.

---

**example 4:** \( \langle Sunny, Warm, High, Normal, Cool, Change \rangle \)

This example does **not** satisfy \( h_e \), because the *water* is cool. Hence, Aldo would not enjoy his favorite sport on this day.

\( \Rightarrow h_e \) is **not** consistent with the training examples \( D \)
Concept Learning as Search

• concept learning as search through the space of hypotheses $H$
  (implicitly defined by the hypothesis representation) with the goal of
  finding the hypothesis that best fits the training examples

• most practical learning tasks involve very large, even infinite
  hypothesis spaces

• many concept learning algorithms organize the search through the
  hypothesis space by relying on the general-to-specific ordering
FIND-S

- exploits general-to-specific ordering
- finds a maximally specific hypothesis $h$ consistent with the observed training examples $D$

**FIND-S Algorithm**

1. Initialize $h$ to the most specific hypothesis in $H$
2. For each positive training instance $x$
   - if the constraint $a_i$ is satisfied by $x$
     - then do nothing
   - else replace $a_i$ with the next more general constraint satisfied by $x$
3. Output hypothesis $h$
FIND-S – Example

- Initialize $h \leftarrow \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$

- example 1: $\langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle$
  $\Rightarrow h \leftarrow \langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle$

- example 2: $\langle \text{Sunny, Warm, High, Strong, Warm, Same} \rangle$
  $\Rightarrow h \leftarrow \langle \text{Sunny, Warm, ?, Strong, Warm, Same} \rangle$

- example 3: $\langle \text{Rainy, Cold, High, Strong, Warm, Change} \rangle$
  This example can be omitted because it is negative.
  Notice that the current hypothesis is already consistent with this example, because it correctly classifies it as negative!

- example 4: $\langle \text{Sunny, Warm, High, Strong, Cool, Change} \rangle$
  $\Rightarrow h \leftarrow \langle \text{Sunny, Warm, ?, Strong, ?, ?} \rangle$
FIND-S – Example

Instances $X$

Hypotheses $H$

$x_1 = \langle \text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same} \rangle, +$  
$x_2 = \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Strong}, \text{Warm}, \text{Same} \rangle, +$  
$x_3 = \langle \text{Rainy}, \text{Cold}, \text{High}, \text{Strong}, \text{Warm}, \text{Change} \rangle, -$  
$x_4 = \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Strong}, \text{Cool}, \text{Change} \rangle, +$  

$h_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$  
$h_1 = \langle \text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same} \rangle$  
$h_2 = \langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, \text{Warm}, \text{Same} \rangle$  
$h_3 = \langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, \text{Warm}, \text{Same} \rangle$  
$h_4 = \langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ? \rangle$
Remarks on FIND-S

• in each step, \( h \) is consistent with the training examples observed up to this point

• unanswered questions:
  
  ▶ Has the learner converged to the correct target concept?
    
    No way to determine whether FIND-S found the only consistent hypothesis \( h \) or whether there are many other consistent hypotheses as well

  ▶ Why prefer the most specific hypothesis?

  ▶ Are the training examples consistent?
    
    FIND-S is only correct if \( D \) itself is consistent. That is, \( D \) has to be free of classification errors.

  ▶ What if there are several maximally specific consistent hypotheses?
CANDIDATE-ELIMINATION addresses several limitations of the FIND-S algorithm

**key idea:** description of the set of all hypotheses consistent with $D$ without explicitly enumerating them

- performs poorly with noisy data
- useful conceptual framework for introducing fundamental issues in machine learning
Version Spaces

- to incorporate the key idea mentioned above, a compact representation of all consistent hypotheses is necessary

- Version space $V S_{H,D}$, with respect to hypothesis space $H$ and training data $D$, is the subset of hypotheses from $H$ consistent with $D$.

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

- $V S_{H,D}$ can be represented by the most general and the most specific consistent hypotheses in form of boundary sets within the partial ordering
Version Spaces

- The **general boundary set** $G$, with respect to hypothesis space $H$ and training data $D$, is the set of maximally general members of $H$ consistent with $D$.
  \[
  G \equiv \{ g \in H | \text{Consistent}(g, D) \land (\neg \exists g' \in H)[(g' >_g g) \land \text{Consistent}(g', D)]\}
  \]

- The **specific boundary set** $S$, with respect to hypothesis space $H$ and training data $D$, is the set of minimally general (i.e., maximally specific) members of $H$ consistent with $D$.
  \[
  S \equiv \{ s \in H | \text{Consistent}(s, D) \land (\neg \exists s' \in H)[(s >_g s') \land \text{Consistent}(s', D)]\}
  \]
Version Spaces

\[ S_4: \{<\text{Sunny}, \text{Warm}, ?, ?, \text{Strong}, ?, ?>\} \]

Candidate Elimination Algorithm

Initialize $G$ to the set of maximally general hypotheses in $H$
Initialize $S$ to the set of maximally specific hypotheses in $H$
For each training example $d \in D$, do

- **If $d$ is a positive example**
  - Remove from $G$ any hypothesis inconsistent with $d$
  - For each hypothesis $s$ in $S$ that is inconsistent 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 $S$

- **If $d$ is a negative example**
  - Remove from $S$ any hypothesis inconsistent with $d$
  - For each hypothesis $g$ in $G$ that is inconsistent 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 $G$
Illustrative Example

- Initialization of the Boundary sets
  - \( G_0 \leftarrow \{\langle ?, ?, ?, ?, ?, ? \rangle\} \)
  - \( S_0 \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\} \)

- example 1: \( \langle \text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same} \rangle \)

  \( S \) is overly specific, because it wrongly classifies example 1 as false. So \( S \) has to be revised by moving it to the least more general hypothesis that covers example 1 and is still more special than another hypothesis in \( G \).

  \[ \Rightarrow S_1 = \{\langle \text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same} \rangle\} \]
  \[ \Rightarrow G_1 = G_0 \]

- example 2: \( \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Strong}, \text{Warm}, \text{Same} \rangle \)

  \[ \Rightarrow S_2 = \{\langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, \text{Warm}, \text{Same} \rangle\} \]
  \[ \Rightarrow G_2 = G_1 = G_0 \]
Illustrative Example

Training Examples:
1. \(\langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle, \text{EnjoySport} = \text{Yes} \)
2. \(\langle \text{Sunny, Warm, High, Strong, Warm, Same} \rangle, \text{EnjoySport} = \text{Yes} \)
example 3: \(\langle \text{Rainy}, \text{Cold}, \text{High}, \text{Strong}, \text{Warm}, \text{Change} \rangle\)

\(G\) is overly general, because it wrongly classifies example 3 as true. So \(G\) has to be revised by moving it to the least more specific hypotheses that covers example 3 and is still more general than another hypothesis in \(S\).

There are several alternative minimally more specific hypotheses.

\[\Rightarrow S_3 = S_2\]
Illustrative Example

Trainings Examples:

3. \(\langle Rainy, Cold, High, Strong, Warm, Change \rangle, \text{EnjoySport} = No \)
Illustrative Example

- example 4: ⟨Sunny, Warm, High, Strong, Cool, Change⟩

$S_4 = \{\langle Sunny, Warm, ?, Strong, ?, ? \rangle\}$

$G_4 = \{\langle Sunny, ?, ?, ?, ?, ? \rangle, \langle ?, Warm, ?, ?, ?, ? \rangle\}$
Illustrative Example

\[ S_4: \{<\text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ?>\} \]

Remarks

• Will the algorithm converge to the correct hypothesis?
  ▶ convergence is assured provided there are no errors in $D$ and the $H$ includes the target concept
  ▶ $G$ and $S$ contain only the same hypothesis

• How can partially learned concepts be used?
  ▶ some unseen examples can be classified unambiguously as if the target concept had been fully learned
    • positive iff it satisfies every member of $S$
    • negative iff it doesn’t satisfy any member of $G$
  ▶ otherwise an instance $x$ is classified by majority (if possible)
Inductive Bias

- fundamental property of inductive learning
  - a learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying unseen examples
  - e.g., learning the *EnjoySport* concept was based on the assumption that the target concept could be represented as a conjunction of attribute values

- inductive bias $\approx$ policy by which the learner generalizes beyond the observed training data to infer the classification of new instances
Inductive Bias

Consider a concept learning algorithm $L$ for the set of instances $X$. Let $c$ be an arbitrary concept defined over $X$, and $D_c = \{<x, c(x)>\}$ an arbitrary set of training examples of $c$. Let $L(x_i, D_c)$ denote the classification assigned to the instance $x_i$ by $L$ after training on the data $D_c$.

The **inductive bias** of $L$ is any minimal set of assertions $B$ such that

$$(\forall x_i \in X)[(B \land D_c \land x_i) \vdash L(x_i, D_c)]$$
Kinds of Inductive Bias

- **Restriction Bias** (aka Language Bias)
  - entire $H$ is searched by learning algorithm
  - hypothesis representation **not expressive enough** to encompass all possible concepts
  - e.g. CANDIDATE-ELIMINATION: for the hypothesis language used in the “enjoy”-example $H$ only includes conjunctive concepts

- **Preference Bias** (aka Search Bias)
  - hypothesis representation encompasses all possible concepts
  - learning algorithm does not consider each possible hypothesis
  - e.g. use of heuristics, greedy strategies

⇒ Preference Bias more desirable, because it assures

\[(\exists h \in H)[(\forall x \in X)[h(x) = c(x)]]\]
An Unbiased Learner

- an unbiased $H = 2^{|X|}$ would contain every teachable function
- for such a $H$,
  - $G$ would always contain the **negation of the disjunction of observed negative examples**
  - $S$ would always contain the **disjunction of the observed positive examples**
- hence, only observed examples will be classified correctly
- in order to converge to a single target concept, every $x \in X$ has to be in $D$
- the learning algorithm is unable to generalize beyond observed training data
Inductive System vs. Theorem Prover

**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
- Theorem Prover
- Assertion "H contains" the target concept
- Classification of new instance, or "don't know"

*Inductive bias made explicit*
Summary

- Concept learning is defined as learning a boolean-valued function from training examples.
- For some hypothesis spaces, hypotheses can be ordered by generality. This allows for efficient representation as version spaces.
- The candidate-elimination algorithm exploits the general to specific ordering of hypotheses in version space.
- There is no bias-free learning.
- The hypothesis language defines a restriction bias (maybe some problems cannot be represented and therefore not learned).
- The search-strategy defines a preference bias (the “correct” hypothesis might be found, if the search strategy is “suitable”).
- If the inductive bias of a learner were represented explicitly, induction could be posed as a deduction problem: deriving a classification of a new example from bias and training examples.
## FIND-S / CANDIDATE ELIMINATION

<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>unsupervised learning</th>
</tr>
</thead>
</table>

### Approaches:

<table>
<thead>
<tr>
<th>Concept / Classification</th>
<th>Policy Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbolic</td>
<td>statistical / neuronal network</td>
</tr>
<tr>
<td>inductive</td>
<td>analytical</td>
</tr>
</tbody>
</table>

### Learning Strategy:

⇒ learning from examples