CmSc310  
Artificial Intelligence  

Machine Learning  

1. What is learning?  
   - A computer program learns if it improves its performance at some task through experience (T. Mitchell, 1997)  
   - Any change in a system that allows it to perform better (Simon 1983)  

What do we learn:  
   - Descriptions  
   - Relations  
   - Procedures  

Relations and procedures are learned in the form of rules how to recognize/classify objects, states, events, and rules how to transform an initial situation to achieve a goal (final state)  

How do we learn:  
   - Rote learning - storage of computed information.  
   - Taking advice from others. (Advice may need to be operationalized.)  
   - Learning from problem solving experiences - remembering experiences and generalizing from them. (May add efficiency but not new knowledge.)  
   - Learning from examples. (May or may not involve a teacher.)  
   - Learning by experimentation and discovery. (Decreasing burden on teacher, increasing burden on learner.)  

2. Approaches to Machine Learning  
   - Symbol-based  
   - Connectionist Learning  
   - Evolutionary learning  

3. Concept Learning - Inductive Symbol-Based Machine Learning  
   - Decision trees: ID3 algorithm  
   - Version space search  
   - Explanation-based learning  
   - Supervised learning  
   - Reinforcement learning
3.1. Version space search for concept learning

- Concepts – describe classes of objects
- Concepts consist of feature sets
- Operation on concept descriptions
  - Generalization: Replace a feature with a variable
  - Specialization: Instantiate a variable with a feature

Positive and Negative examples of a concept

- The concept description has to match all positive examples
- The concept description has to be false for the negative examples

The VERSION SPACE represents all the alternative PLAUSIBLE DESCRIPTIONS of the concept. A plausible description is one that is applicable to all known positive examples and no known negative example.

The version space contains two sets of hypotheses:
  - \( G \) – the most general hypotheses that match the training data
  - \( S \) – the most specific hypotheses that match the training data

Each hypothesis is represented as a vector of values of the known attributes

Example:
Consider the task to obtain a description of the concept: Japanese Economy car.
The attributes under consideration are: Origin, Manufacturer, Color, Decade, Type

Assume that the training data are:
- Positive example: (Japan, Honda, Blue, 1980, Economy)
- Positive example: (Japan, Honda, White, 1980, Economy)
- Negative example: (Japan, Toyota, Green, 1970, Sports)

The most general hypothesis that matches these data is:
\((?, \text{Honda}, ?, ?, ?)\), the symbol ‘?’ means that the attribute may take any value

The most specific hypothesis that matches the examples is:
\((\text{Japan}, \text{Honda}, ?, 1980, \text{Economy})\)

General-to-Specific Ordering

Concept learning: Search through a search space that consists of all possible hypotheses, where the goal is the hypothesis that most closely represents the concept.
Partial ordering in the search space:

Most general hypothesis: \( h_g = < ?, ?, \ldots, ? > \)

Most specific hypothesis: \( h_s = < \emptyset, \emptyset, \ldots > \)

Relation “more general than or as general as”

\( h_1 \geq_g h_2 \)

All instances matched by \( h_2 \) are also matched by \( h_1 \).

**Algorithm: CANDIDATE ELIMINATION**

Given: - A representation language  
- A set of positive and negative examples expressed in that language

Compute: A concept description that is consistent with all the positive examples and none of the negative examples.

Method: - Initialize \( G \) to contain one element: all features are variables.  
- Initialize \( S \) to contain one element: the first positive example.  
- Accept a new training example.

- If it is a positive example:
  - Remove from \( G \) any descriptions that do not cover the example.  
  - Update \( S \) to contain the most specific set of descriptions in the version space that cover the example and the current elements of the \( S \) set (i.e., generalize the elements of \( S \) as little as possible so that they cover the new training example).

- If it is a negative example:
  - Remove from \( S \) any descriptions that cover the example.  
  - Update \( G \) to contain the most general set of descriptions in the version space that do not cover the example (i.e., specialize the elements of \( G \) as little as possible so that the negative example is no longer covered by any of the elements of \( G \)).

- If \( S \) and \( G \) are both singleton sets, then:
  - if they are identical, output their value and stop.  
  - if they are different, the training cases were inconsistent. Output this result and stop.

- Else continue accepting new training examples.
**Example:** (from Rich & Knight): Learning the concept of "Japanese economy car"

Features: Origin, Manufacturer, Color, Decade, Type

**POSITIVE EXAMPLE:** (Japan, Honda, Blue, 1980, Economy)

Initialize G to singleton set that includes everything
Initialize S to singleton set that includes first positive example

\[
G = \{ (?, ?, ?, ?, ?) \} \\
S = \{ (Japan, Honda, Blue, 1980, Economy) \}
\]

**NEGATIVE EXAMPLE:** (Japan, Toyota, Green, 1970, Sports)

Specialize G to exclude negative example

\[
G = \{ (?, Honda, ?, ?, ?), \\
(?, ?, Blue, ?, ?,) \\
(?, ?, ?, 1980, ?) \\
(?, ?, ?, ?, Economy) \} \\
S = \{ (Japan, Honda, Blue, 1980, Economy) \}
\]

**POSITIVE EXAMPLE:** (Japan, Toyota, Blue, 1990, Economy)

Remove from G descriptions inconsistent with positive example
Generalize S to include positive example

\[
G = \{ (?, ?, Blue, ?, ?) \\
(?, ?, ?, ?, Economy) \} \\
S = \{ (Japan, ?, Blue, ?, Economy) \}
\]

**NEGATIVE EXAMPLE:** (USA, Chrysler, Red, 1980, Economy)

Specialize G to exclude negative example
(but staying within version space, i.e., staying consistent with S)

\[
G = \{ (?, ?, Blue, ?, ?) \\
( Japan, ?, ?, ?, Economy) \} \\
S = \{ (Japan, ?, Blue, ?, Economy) \}
\]
POSITIVE EXAMPLE: (Japan, Honda, White, 1980, Economy)

Remove from G descriptions inconsistent with positive example
Generalize S to include positive example

G = {(Japan, ?, ?, ?, Economy)}
S = {(Japan, ?, ?, ?, Economy)}

S = G, both singleton => done!