Concept and Concept Learning

- A Concept is a a subset of objects or events defined over a larger set

  [Example: The concept of a ***bird*** is the subset of all objects (i.e., the set of all ***things*** or all ***animals***) that belong to the category of bird.]
- Alternatively, a concept is a boolean-valued function defined over this larger set

  [Example: a function defined over all ***animals*** whose value is ***true*** for birds and ***false*** for every other animal].

Concept and Concept Learning

- Given a set of examples labeled as members or non-members of a concept, *concept-learning* consists of automatically inferring the general definition of this concept.

- In other words, *concept-learning* consists of approximating a boolean-valued function from training examples of its input and output.

Terminology and Notation

- The set of items over which the concept is defined is called the set of ***instances*** (denoted by X)
- The concept to be learned is called the *Target Concept* (denoted by c: X--> {0,1})
- The set of ***Training Examples*** is a set of instances, x, along with their target concept value c(x).
- Members of the concept (instances for which c(x)=1) are called ***positive examples***.
- Nonmembers of the concept (instances for which c(x)=0) are called ***negative examples***.
- H represents the set of ***all possible hypotheses***. H is determined by the human designer's choice of a hypothesis representation.
- **The goal of concept-learning is to find a hypothesis h:X --> {0,1} such that h(x)=c(x) for all x in X**

Concept Learning viewed as Search

- Concept Learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.
- Selecting a Hypothesis Representation is an important step since it restricts (or *biases*) the space that can be searched. [For example, the hypothesis "If the air temperature is cold **or** the humidity high then it is a good day for water sports" cannot be expressed in our chosen representation.]

Review of Concepts in Class

General to specific ordering of Hypotheses

- **Definition:** Let *hj* and *hk* be boolean-valued functions defined over X. Then *hj* is **more-general-than-or-equal-to** *hk* iff For all x in X, [(*hk*(x) = 1) --> (*hj*(x)=1)]
- **Example:**
  - *h1* = <Sunny,?,?,Strong,?,?>
  - *h2* = <Sunny,?,?,?,?,?>

  Every instance that are classified as positive by *h1* will also be classified as positive by *h2* in our example data set. Therefore *h2* is more general than *h1*.
- We also use the ideas of **“strictly”-more-general-than,** and **more-specific-than**

Find-S, a Maximally Specific Hypothesis Learning Algorithm

- Initialize *h* to the most specific hypothesis in *H*
- For each <u>positive</u> training instance *x*
  - For each attribute constraint *ai* in *h*

    If the constraint *ai* is satisfied by *x*

    then do nothing

    else replace *ai* in h by the next more general constraint that is satisfied by *x*
- Output hypothesis *h*

Although Find-S finds a hypothesis consistent with the training data, it does not indicate whether that is the only one available

Version Spaces and the Candidate-Elimination Algorithm

- **Definition:** A hypothesis *h* is **consistent** with a set of training examples *D* iff $h(x) = c(x)$ for each example $\langle x, c(x) \rangle$ in *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.*
- **NB:** While a Version Space can be exhaustively enumerated, a more compact representation is preferred.

A Compact Representation for Version Spaces

- Instead of enumerating all the hypotheses consistent with a training set, we can represent its **most specific** and **most general** boundaries. The hypotheses included in-between these two boundaries can be generated as needed.
- **Definition:** The **general boundary** *G*, with respect to hypothesis space *H* and training data *D*, is the set of maximally general members of *H* consistent with *D*.
- **Definition:** The **specific boundary** *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*.

Candidate Elimination Algorithm

- The candidate-Elimination algorithm computes the version space containing all (and only those) hypotheses from *H* that are consistent with an observed sequence of training examples.

Example 1

- **Learning the concept of "Japanese Economy Car“**

Features:

Country of Origin

Manufacturer

Color

Decade

Type

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive
Japan	Toyota	Green	1980	Economy	Positive
Japan	Honda	Red	1990	Economy	Negative

Positive Example 1
(Japan, Honda, Blue, 1980, Economy)

- Initialize G to a singleton set that includes everything.

G = { (?, ?, ?, ?, ?) }

- Initialize S to a singleton set that includes the first positive example.

S = { (Japan, Honda, Blue, 1980, Economy) }

Negative Example 2
(Japan, Toyota, Green, 1970, Sports)

- Specialize G to exclude the negative example.

- G =
  { (?, Honda, ?, ?, ?),
  (?, ?, Blue, ?, ?),
  (?, ?, ?, 1980, ?),
  (?, ?, ?, ?, Economy) } S = { (Japan, Honda, Blue, 1980, Economy) }

Positive Example 3(Japan, Toyota, Blue, 1990, Economy)

- Prune G to exclude descriptions inconsistent with the positive example.

G =

{ (?, ?, Blue, ?, ?),
(?, ?, ?, ?, Economy) }

- Generalize S to include the positive example.

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

Negative Example (USA, Chrysler, Red, 1980, Economy)

- Specialize G to exclude the negative example (but stay consistent with S)

G =

{ (?, ?, Blue, ?, ?),

(Japan, ?, ?, ?, Economy) }

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

Positive Example(Japan, Honda, White, 1980, Economy)

- Prune G to exclude descriptions inconsistent with positive example.

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

- Generalize S to include positive example.

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

Positive Example: (Japan, Toyota, Green, 1980, Economy)

- New example is consistent with version-space, so no change is made.

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

Negative Example: (Japan, Honda, Red, 1990, Economy)

- Example is inconsistent with the version-space.

  G cannot be specialized.
  S cannot be generalized.

- The version space **collapses**.
- Conclusion: No conjunctive hypothesis is consistent with the data set.

Remarks on Version Spaces and Candidate Elimination

- The version space learned by the Candidate-Elimination Algorithm will converge toward the hypothesis that correctly describes the target concept provided: (1) There are no errors in the training examples; (2) There is some hypothesis in *H* that correctly describes the target concept.
- Convergence can be speeded up by presenting the data in a strategic order. The best examples are those that satisfy exactly half of the hypotheses in the current version space.
- Version-Spaces can be used to assign certainty scores to the classification of new examples