Inductive Bias & Concept Learning

1. Concept learning:

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

category. Concept learning can be formulated as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples. In many cases this search can be efficiently organized by taking

2. An example: EnjoySport.

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

- indicate by a "?" that any value is acceptable for this attribute,
- specify a single required value (e.g., *Warm*) for the attribute, or
- indicate by a "Ø" that no value is acceptable.

The most general hypothesis—that every day is a positive example—is represented by

$$\langle ?, ?, ?, ?, ?, ? \rangle$$

and the most specific possible hypothesis—that *no* day is a positive example—is represented by

$$\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$$

3. Inductive Learning:

Notice that although the learning task is to determine a hypothesis h identical to the target concept c over the entire set of instances X, the only information available about c is its value over the training examples. Therefore, inductive

- soft bias:

The 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.

- hard bias (representative power): depends on assumptions

4. General-to-Specific Hypotheses

Definition: Let h_j and h_k be boolean-valued functions defined over X. Then h_j is **more_general_than_or_equal_to** h_k (written $h_j \geq_g h_k$) if and only if

$$(\forall x \in X)[(h_k(x) = 1) \rightarrow (h_j(x) = 1)]$$

- Algorithm (Find-S)

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 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

TABLE 2.3
FIND-S Algorithm.

version space: the set of all hypotheses consistent with the training data

- hard bias: Conjunction hypotheses
  - instance space: $3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96$
  - hypothesis space: $1 + (4 \times 3 \times 3 \times 3 \times 3 \times 3) = 973$

The key property of the FIND-S algorithm is that for hypothesis spaces described by conjunctions of attribute constraints (such as H for the *EnjoySport* task), FIND-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples. Its final hypothesis will

5. Candidate-Elimination hypotheses

Represents all the hypothesis. No hard bias

- Hard Bias → In principle, the LIST-THEN-ELIMINATE algorithm can be applied whenever the hypothesis space H is finite. It has many advantages, including the fact that it is guaranteed to output all hypotheses consistent with the training data. Unfortunately, it requires exhaustively enumerating all hypotheses in H—an unrealistic

- Soft Bias → ELIMINATE algorithm, defined in Table 2.4.
The LIST-THEN-ELIMINATE algorithm first initializes the version space to contain all hypotheses in H, then eliminates any hypothesis found inconsistent with any training example. The version space of candidate hypotheses thus shrinks

Inductive bias of CANDIDATE-ELIMINATION algorithm. The target concept c is contained in the given hypothesis space H.

- Algorithm

The LIST-THEN-ELIMINATE Algorithm

1. *VersionSpace* ← a list containing every hypothesis in H
2. For each training example, $\langle x, c(x) \rangle$
   remove from *VersionSpace* any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypotheses in *VersionSpace*