UNIT-II

CONCEPT LEARNING AND THE GENERAL-TO-SPECIFIC ORDERING

2.1 INTRODUCTION

- Inducing general functions from specific training examples is a main issue of machine learning.
- Concept Learning: Acquiring the definition of a general category from given sample positive and negative training examples of the category.
- Concept Learning can be seen as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples.
- The hypothesis space has a *general-to-specific ordering* of hypotheses, and the search can be efficiently organized by taking advantage of a naturally occurring structure over the hypothesis space.
- An example for concept-learning is the learning of bird-concept from the given examples of birds (positive examples) and non-birds (negative examples).
- We are trying to learn the definition of a concept from given examples.

> **A Formal Definition for Concept Learning:** Inferring a boolean-valued function from training examples of its input and output.

2.2 A CONCEPT LEARNING TASK

To understand concept learning, consider the example of "EnjoySport". The task is to "learn to predict the value of EnjoySport for an arbitrary day", based on the values of its attribute values. Below table describes a set of example days, each represented by a set of *attributes.*

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	Warm	Change	**YES**

ATTRIBUTES   **CONCEPT**

2.2.1 Hypotheses representation of EnjoySport Example:

- In a simple representation of the hypothesis, each hypothesis consists of a conjunction of constraints on the instance attributes.
- For EnjoySport Example, each hypothesis will be a vector of six constraints, specifying the values of the six attributes:
    - (Sky, AirTemp, Humidity, Wind, Water, and Forecast)
- Each attribute will be:

    ? - indicating any value is acceptable for the attribute **(don't care)**

    Single value – specifying a single required value (ex. Warm) **(specific)**

    0 - indicating no value is acceptable for the attribute **(no value)**

- Example: The hypothesis that X enjoys his favorite sport only on cold days with high humidity (independent of the values of the other attributes) is represented by the expression

    <?, Cold, High, ?, ?, ?>

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

    <?,?,?,?,?,?>

- The most specific hypothesis – that no day is a positive example – is represented by

    <ø, ø, ø, ø, ø, ø>

2.2.2 Basic Terminology:

Instances: The set of items over which the concept is defined is called the set of instances, which we denote by X.

In our example EnjoySport, X is the set of all possible days, each represented by the attributes Sky, AirTemp, Humidity, Wind, Water, and Forecast.

Target Concept: The concept or function to be learned is called the target concept, which is denoted by c. In general, c can be any Boolean-valued function defined over the instances X; i.e., $c : X \rightarrow \{0, 1\}$.

In our current example, the target concept corresponds to the value of the attribute EnjoySport (i.e., $c(x) = 1$ if EnjoySport = Yes, and $c(x) = 0$ if EnjoySport = No).

Training Example: Each training example $<x, c(x)>$, consists of an instance x from X, along with its target concept value $c(x)$.

Positive Examples: Instances for which $c(x) = 1$ are called positive examples, or members of the target concept.

Negative Examples: Instances for which $c(x) = 0$ are called negative examples, or non-members of the target concept.

Hypotheses: H denotes the set of all possible hypotheses that the learner may consider regarding the identity of the target concept c. In general, each hypothesis h in H represents a Boolean-valued function defined over X; that is, $h : X \rightarrow \{0, 1\}$. The goal of the learner is to find a hypothesis h such that $h(x) = c(x)$ for all x in X.

The EnjoySport Concept Learning Task:
- Given:
  - Instances X: Possible days, each described by the attributes
    - Sky (with possible 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), and
    - Forecast (with values Same and Change).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes Sky, AirTemp, Humidity, Wind, Water, and Forecast.
  - Target concept c: EnjoySport : $X \rightarrow \{0,1\}$
  - Training examples D: Positive and negative examples of the target function

- Determine: A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

2.2.3 The Inductive Learning Hypothesis

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

– Inductive learning algorithms can at best guarantee that the output hypothesis fits the target concept over the training data.

– Lacking any further information, our assumption is that the best hypothesis regarding unseen instances is the hypothesis that best fits the observed training data. This is the fundamental assumption of inductive learning.

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.

2.3 CONCEPT LEARNING AS SEARCH

Concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.

• The goal of this search is to find the hypothesis that best fits the training examples.

• By selecting a hypothesis representation, the designer of the learning algorithm implicitly defines the space of all hypotheses that the program can ever represent and therefore can ever learn.

• For example consider **EnjoySport** learning task:

Sky has 3 possible values, and other 5 attributes have 2 possible values.

• There are 96 (= 3.2.2.2.2.2) distinct instances in X.

• There are 5120 (=5.4.4.4.4.4) *syntactically distinct hypotheses* in H.

– Two more values for attributes: ? and 0

• Every hypothesis containing one or more 0 symbols represents the empty set of instances; that is, it classifies every instance as negative.

• There are 973 (= 1 + 4.3.3.3.3.3) *semantically distinct hypotheses* in H.

– Only one more value for attributes: ?, and one hypothesis representing empty set of instances.

• Although EnjoySport has small, finite hypothesis space, most learning tasks have much larger (even infinite) hypothesis spaces.

– We need efficient search algorithms on the hypothesis spaces.

2.3.1 General-to-Specific Ordering of Hypotheses:

- Many algorithms for concept learning organize the search through the hypothesis in a **general-to-specific ordering of hypotheses**.

- By using this ordering of hypotheses, the learning algorithms can exhaustively search even infinite hypothesis spaces without explicitly enumerating every hypothesis.

- Consider two hypotheses

  h1 = (Sunny, ?, ?, Strong, ?, ?)

  h2 = (Sunny, ?, ?, ?, ?, ?)

  • Now consider the sets of instances that are classified positive by hl and by h2.

  – Because h2 imposes fewer constraints on the instance, it classifies more instances as positive.

  – In fact, any instance classified positive by hl will also be classified positive by h2.

– Therefore, we say that h2 is *more general than* hl.

More-General-Than Relation

• For any instance x in X and hypothesis h in H, we say that x satisfies h if and only if h(x) = 1.

• **More-General-Than-Or-Equal Relation:**

Let h1 and h2 be two boolean-valued functions defined over X.

Then h1 is ***more-general-than-or-equal-to*** h2 (written h1 ≥ h2) if and only if any instance that satisfies h2 also satisfies h1.

- h1 is ***more-general-than*** h2 (h1 > h2) if and only if h1≥h2 is true and h2≥h1 is false. We also say h2 is ***more-specific-than*** h1.

> **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
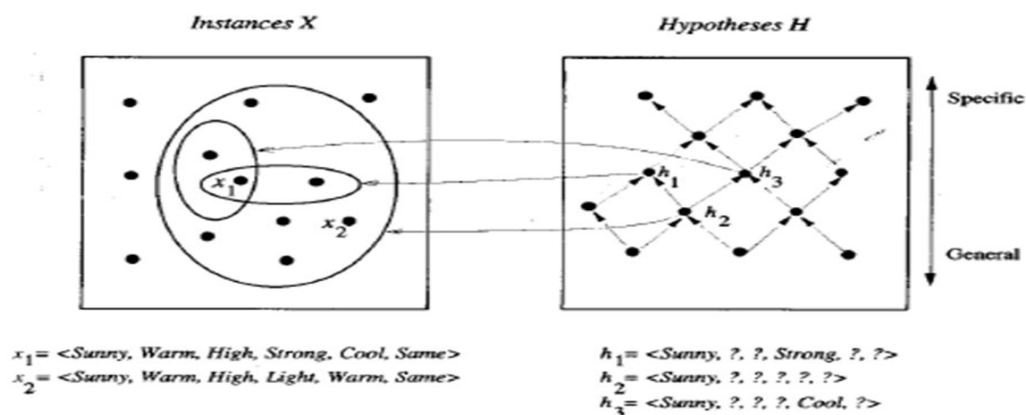> $$(\forall x \in X)[(h_k(x) = 1) \rightarrow (h_i(x) = 1)]$$

- For example, consider the two hypotheses:

    h_1 = <Sunny, ?, ?, Strong, ?, ?>

    h_2 = <Sunny, ?, ?, ?, ?, ?>

    Any instance classified positive by h_1 will also be classified positive by h_2. Therefore, we say that h_2 is more general than h_1.

- The $\geq_g$, relation defines a partial order over the hypothesis space H (the relation is reflexive, anti-symmetric and transitive).

x_1 = *<Sunny, Warm, High, Strong, Cool, Same>*
x_2 = *<Sunny, Warm, High, Light, Warm, Same>*

h_1 = *<Sunny, ?, ?, Strong, ?, ?>*
h_2 = *<Sunny, ?, ?, ?, ?, ?>*
h_3 = *<Sunny, ?, ?, ?, Cool, ?>*

Based on the above figure:

- h2 > h1 and h2 > h3

- But there is no more-general relation between h1 and h3

2.4 FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS

FIND-S Algorithm starts from the most specific hypothesis and generalize it by considering only positive examples.

• FIND-S algorithm ignores negative examples.

– As long as the hypothesis space contains a hypothesis that describes the true target concept, and the training data contains no errors, ignoring negative examples does not cause to any problem.

• FIND-S algorithm finds the most specific hypothesis within H that is consistent with the positive training examples.

– The final hypothesis will also be consistent with negative examples if the correct target concept is in H, and the training examples are correct.

- This algorithm uses more-general-than partial ordering to organize the search in the hypotheses space.

- **FIND-S Algorithm:**
    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

- **Example: Using Find-S algorithm for EnjoySport task**

    Step 1: The first step of FIND-S is to initialize *h* to the most specific hypothesis in H.

$$h \leftarrow <ø, ø, ø, ø, ø, ø>$$

    Step 2: For the First training example, hypothesis h is changed as initial h is too specific.

$$h \leftarrow <Sunny, Warm, Normal, Strong, Warm, Same>$$

    Step 3: For the second training example (also positive in this case) forces the algorithm to further generalize *h,* this time substituting a "?" in place of any attribute value in h that is not satisfied by the new example. The refined hypothesis in this case is

$$h \leftarrow <Sunny, Warm, ?, Strong, Warm, Same>$$

Step 4: With 3rd training example, which is negative in this case the algorithm makes no change to h.

Step 5: With 4th training example, h changes as below:

$$h \leftarrow \text{<Sunny, Warm, ?, Strong, ?, ?>}$$

Key property of FIND-S:

- FIND-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples.
- Also the final hypothesis will also be consistent with the negative examples provided the correct target concept is contained in H, and provided the training examples are correct.

$$h_0 = \text{<Ø, Ø, Ø, Ø, Ø, Ø>}$$

x_1 = *<Sunny Warm Normal Strong Warm Same>*, + h_1 = *<Sunny Warm Normal Strong Warm Same>*

x_2 = *<Sunny Warm High Strong Warm Same>*, + h_2 = *<Sunny Warm ? Strong Warm Same>*

x_3 = *<Rainy Cold High Strong Warm Change>*, - h_3 = *<Sunny Warm ? Strong Warm Same>*

x_4 = *<Sunny Warm High Strong Cool Change>*, + h_4 = *<Sunny Warm ? Strong ? ?>*
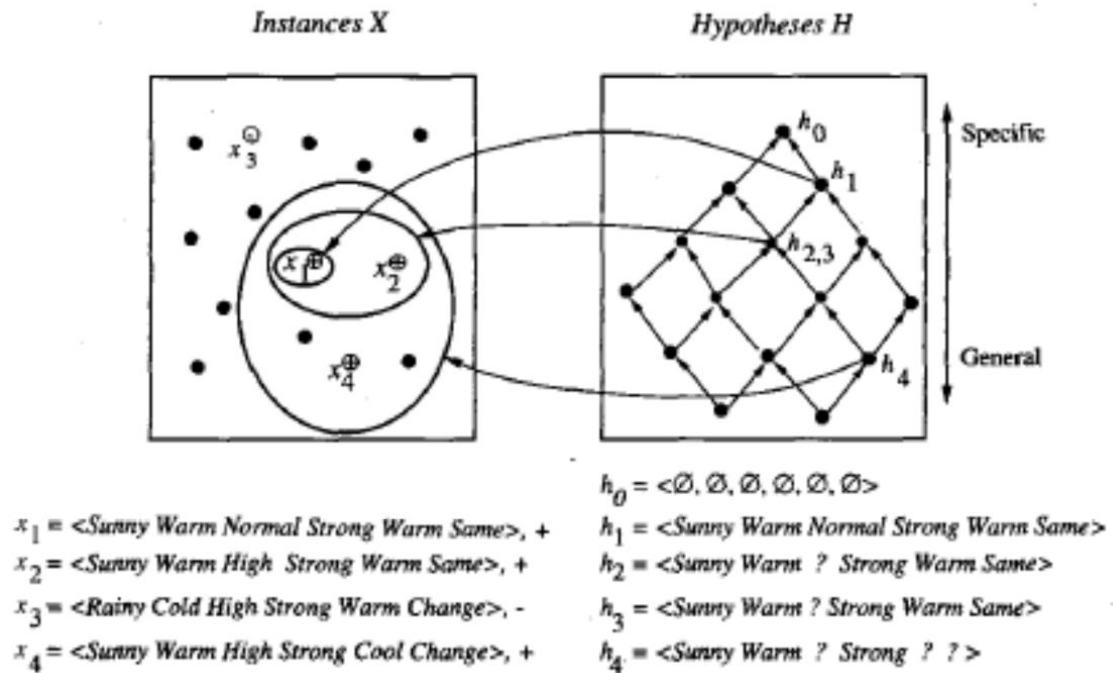
Limitations of FIND-S:

1. Has FIND-S converged to the correct target concept?

– Although FIND-S will find a hypothesis consistent with the training data, it has no way to determine whether it has found the only hypothesis in H consistent with the data (i.e., the correct target concept), or whether there are many other consistent hypotheses as well.

– We would prefer a learning algorithm that could determine whether it had converged and, if not, at least characterize its uncertainty regarding the true identity of the target concept.

2. Why prefer the most specific hypothesis?

– In case there are multiple hypotheses consistent with the training examples, FIND-S will find the most specific.

– It is unclear whether we should prefer this hypothesis over, say, the most general, or some other hypothesis of intermediate generality.

3. Are the training examples consistent?

– In most practical learning problems there is some chance that the training examples will contain at least some errors or noise.

– Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples.

– We would prefer an algorithm that could at least detect when the training data is inconsistent and, preferably, accommodate such errors.

4. What if there are several maximally specific consistent hypotheses?

– In the hypothesis language H for the EnjoySport task, there is always a unique, most specific hypothesis consistent with any set of positive examples.

– However, for other hypothesis spaces there can be several maximally specific hypotheses consistent with the data.

– In this case, FIND-S must be extended to allow it to backtrack on its choices of how to generalize the hypothesis, to accommodate the possibility that the target concept lies along a different branch of the partial ordering than the branch it has selected.

2.5 VERSION SPACES AND THE CANDIDATE-ELIMINATION ALGORITHM

FIND-S outputs a hypothesis from H, that is consistent with the training examples, this is just one of many hypotheses from H that might fit the training data equally well.

• The key idea in the Candidate-Elimination algorithm is to output a description of the set of all hypotheses consistent with the training examples.

2.5.1 Representation:

– Candidate-Elimination algorithm computes the description of this set without explicitly enumerating all of its members.

– This is accomplished by using the more-general-than partial ordering and maintaining a compact representation of the set of consistent hypotheses.

> *Consistent hypothesis:*
> - A hypothesis h is **consistent** with a set of training examples *D* if and only if *h(x) = c(x)* for each example <x, *c(x)*> in *D.*
>
> $$Consistent\,(h,\,D) \equiv (\forall \langle x, c(x) \rangle \in D)\; h(x) = c(x)$$

The key difference between this definition of **consistent** and **satisfies.**

• An example *x* is said to **satisfy** hypothesis **h** when **h(x)** = 1, regardless of whether x is a positive or negative example of the target concept.

• However, whether such an example is **consistent** with **h** depends on the target concept, and in particular, whether **h(x)** = **c(x).**

Version Spaces

• The Candidate-Elimination algorithm represents the set of all hypotheses consistent with the observed training examples.

• This subset of all hypotheses is called the **version space** with respect to the hypothesis space H and the training examples D, because it contains all plausible versions of the target concept.

> **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.
>
> $$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

2.5.2 List-Then-Eliminate Algorithm:

• List-Then-Eliminate algorithm 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 as more examples are observed, until ideally just one hypothesis remains that is consistent with all the observed examples.

– Presumably, this is the desired target concept.

– If insufficient data is available to narrow the version space to a single hypothesis, then the algorithm can output the entire set of hypotheses consistent with the observed data.

• 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 requirement for all but the most trivial hypothesis spaces.

- **The List-Then-Eliminate Algorithm:**
  - ✓ *VersionSpace* ← a list containing every hypothesis in H
  - ✓ For each training example, *<x, c(x)>*
    - ✓ remove from *VersionSpace* any hypothesis *h* for which *h(x) ≠c(x)*
  - ✓ Output the list of hypotheses in *VersionSpace*

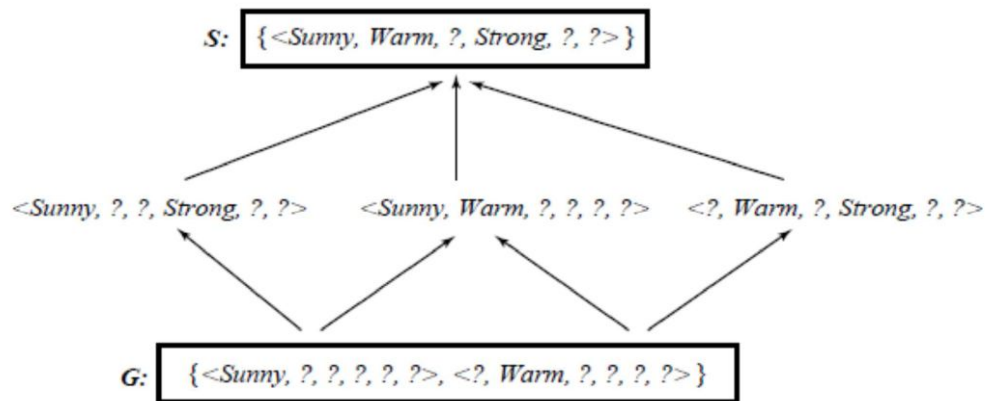### 2.5.3 A More Compact Representation for Version Spaces:

- A version space can be represented with its *general* and *specific boundary sets*.

- The Candidate-Elimination algorithm represents the version space by storing only its most general members G and its most specific members S.

- 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*.

$$G \equiv \{g \in H | Consistent(g, D) \land (\neg\exists g' \in H)[(g' >_g g) \land Consistent(g', D)]\}$$

- The **specific boundary** S, with respect to hypothesis space *H* and

t $$S \equiv \{s \in H | Consistent(s, D) \land (\neg\exists s' \in H)[(s >_g s') \land Consistent(s', D)]\}$$
ɪ

aining data *D*, is the set of minimally general (i.e., maximally specific) members of *H* consistent with *D*.

Example of version space:

S: {<Sunny, Warm, ?, Strong, ?, ?>}

<Sunny, ?, ?, Strong, ?, ?>   <Sunny, Warm, ?, ?, ?, ?>   <?, Warm, ?, Strong, ?, ?>

G: {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>}

- A version space with its general and specific boundary sets.

• The version space includes all six hypotheses shown here,

but can be represented more simply by S and **G.**

- **Version space representation theorem:** Let X be an arbitrary set of instances and let H be a set of Boolean-valued hypotheses defined over X. Let c : $X \rightarrow \{0,1\}$ be an arbitrary target concept defined over X, and let D be an arbitrary set of training examples <x, c(x)>. For all X, H, c, and D such that S and G are well defined,

$$VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G)(g \geq_g h \geq_g s)\}$$

2.5.4 Candidate-Elimination Learning Algorithm:

The Candidate-Elimination algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.

• It begins by initializing the version space to the set of all hypotheses in H; that is, by initializing the G boundary set to contain the most general hypothesis in H

$$G_0 \leftarrow \{ <?, ?, ?, ?, ?, ?> \}$$

and initializing the S boundary set to contain the most specific hypothesis

$$S_0 \leftarrow \{ <0, 0, 0, 0, 0, 0> \}$$

• These two boundary sets delimit the entire hypothesis space, because every other

hypothesis in H is both more general than S0 and more specific than G0.

• As each training example is considered, the S and G boundary sets are generalized and specialized, respectively, to eliminate from the version space any hypotheses found inconsistent with the new training example.

• After all examples have been processed, the computed version space contains all the hypotheses consistent with these examples and only these hypotheses.

Candidate-elimination algorithm using version spaces:

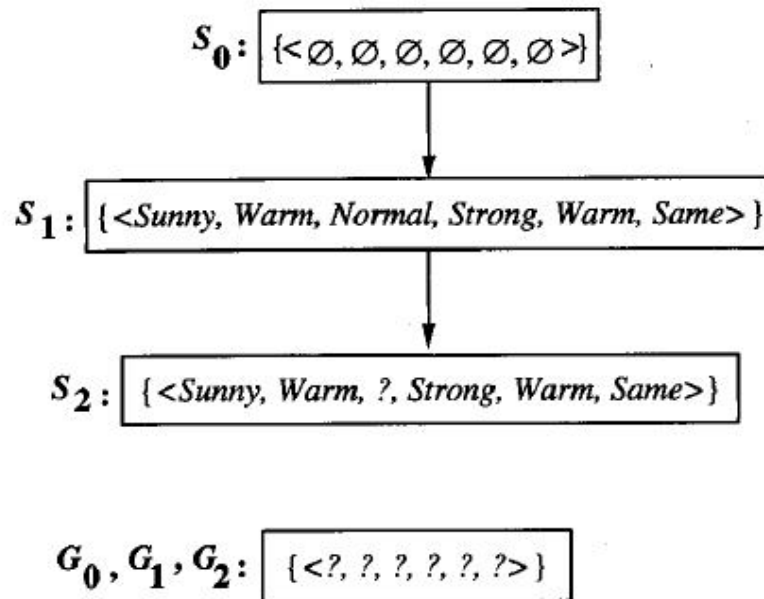Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of maximally specific hypotheses in H
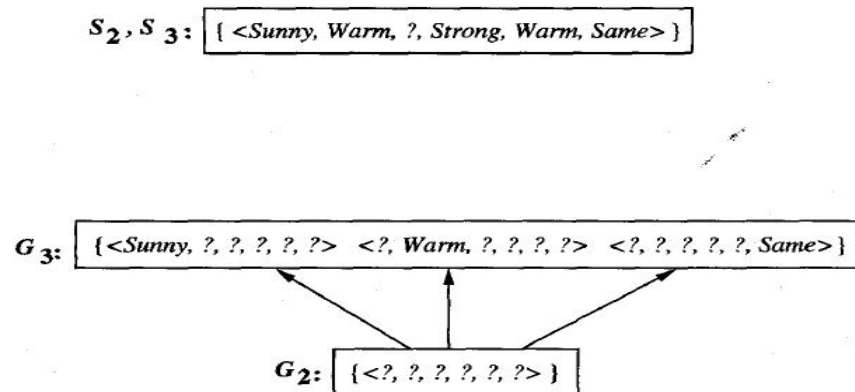For each training example d, do
- If d is a positive example
    a. Remove from G any hypothesis inconsistent with d ,
    b. For each hypothesis **s** in S that is not consistent with d ,
        i. Remove **s** from S
        ii. Add to S all minimal generalizations h of **s** such that
            ✓ h is consistent with d, and some member of G is more general than h
    c. Remove from S any hypothesis that is more general than another hypothesis in S
- If d is a negative example
    a. Remove from S any hypothesis inconsistent with d
    b. For each hypothesis g in G that is not consistent with d
        i. Remove g from G
        ii. 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
    c. Remove from G any hypothesis that is less general than another hypothesis in G

Example of Candidate-elimination algorithm using EnjoySport Task:

Trace 1: *S0* and *G0* are the initial boundary sets corresponding to the most specific and most general hypotheses. Training examples 1 and 2 force the *S* boundary to become more general. They have no effect on the *G* boundary. After observing the 1st and 2nd training examples, the S and G sets are as below:

$$S_0: \{<\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset>\}$$

$$S_1: \{<Sunny,\ Warm,\ Normal,\ Strong,\ Warm,\ Same>\}$$

$$S_2: \{<Sunny,\ Warm,\ ?,\ Strong,\ Warm,\ Same>\}$$

$$G_0, G_1, G_2: \{<?, ?, ?, ?, ?, ?>\}$$

Trace 2: With 3rd training example, the S and G boundaries are as below:

$$S_2, S_3: \{<Sunny,\ Warm,\ ?,\ Strong,\ Warm,\ Same>\}$$

$$G_3: \{<Sunny, ?, ?, ?, ?, ?>\quad <?, Warm, ?, ?, ?, ?>\quad <?, ?, ?, ?, ?, Same>\}$$
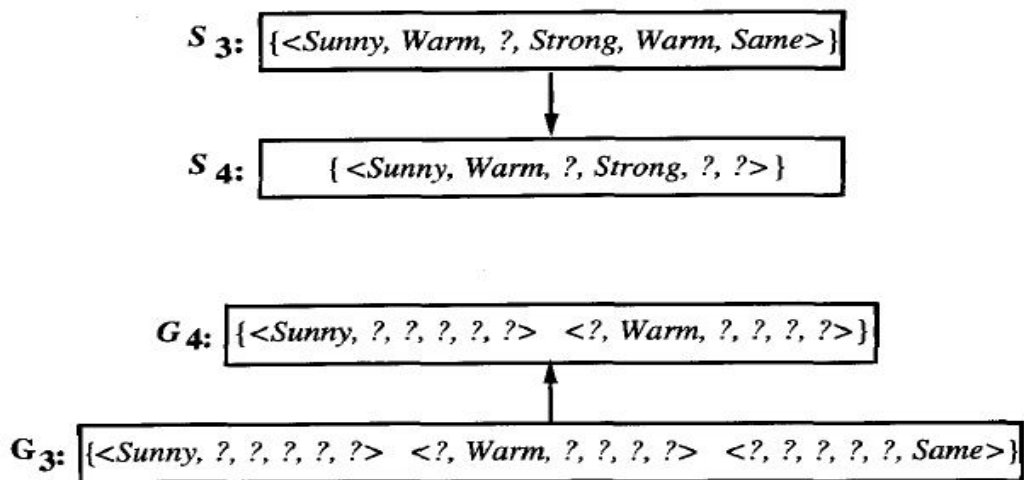
$$G_2: \{<?, ?, ?, ?, ?, ?>\}$$

The hypothesis in G incorrectly predicts that 3rd training example is a positive example. The hypothesis in the G boundary must therefore be specialized until it correctly classifies this new negative example. The members of G3 are shown in the above figure. With the six attributes of EnjoySport task, only three are consistent hypothesis w.r.t S2. For example, the hypothesis $h = <?, ?, Normal, ?, ?, ?>$ is a minimal specialization of G2 that correctly labels 3rd example as a negative example, but it is not included in G3 as it is inconsistent with S2.

Trace 3: With the 4th training example, the S and G boundaries are:

Given that there are six attributes that could be specified to specialize **G2,** why are there only three new hypotheses in **G3?**

• For example, the hypothesis **h** = <?, ?, **Normal,** ?, ?, ?> is a minimal specialization of **G2** that correctly labels the new example as a negative example, but it is not included in **G3.**

– The reason this hypothesis is excluded is that it is inconsistent with S2.

– The algorithm determines this simply by noting that **h** is not more general than the current specific boundary, **S2.**

• In fact, the **S** boundary of the version space forms a summary of the previously encountered positive examples that can be used to determine whether any given hypothesis is consistent with these examples.

• The G boundary summarizes the information from previously encountered negative examples. Any hypothesis more specific than G is assured to be consistent with past negative examples

S 3: {<Sunny, Warm, ?, Strong, Warm, Same>}

↓

S 4: {<Sunny, Warm, ?, Strong, ?, ?>}

G 4: {<Sunny, ?, ?, ?, ?, ?>   <?, Warm, ?, ?, ?, ?>}

↑

G 3: {<Sunny, ?, ?, ?, ?, ?>   <?, Warm, ?, ?, ?, ?>   <?, ?, ?, ?, ?, Same>}

With the 4th example one member of the G boundary is removed as that member fails to cover the new positive example.

The fourth training example further generalizes the S boundary of the version space.

• It also results in removing one member of the G boundary, because this

member fails to cover the new positive example.

– To understand the rationale for this step, it is useful to consider why the offending hypothesis must be removed from G.

– Notice it cannot be specialized, because specializing it would not make it cover the new example.

– It also cannot be generalized, because by the definition of G, any more general hypothesis will cover at least one negative training example.

– Therefore, the hypothesis must be dropped from the G boundary, thereby removing an entire branch of the partial ordering from the version space of hypotheses remaining under consideration

The final version space of EnjoySport task:

After processing the four examples, the boundary sets **S4** and **G4** delimit the version space of all hypotheses consistent with the set of incrementally observed training examples. The entire version space, including those hypotheses bounded by **S4** and **G4,** is shown in below:
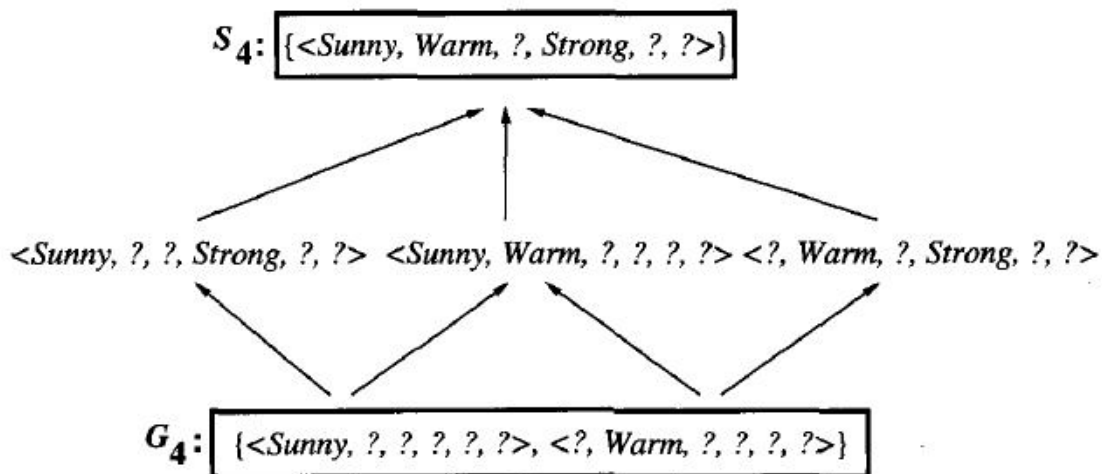
Key points about version space & Candidate-elimination algorithm:

• Positive training examples may force the **S** boundary of the version space to become increasingly general and Negative training examples play the complimentary role of forcing the **G** boundary to become increasingly specific.

- The **S** boundary of the version space forms a summary of the previously encountered positive examples that can be used to determine whether any given hypothesis is consistent with these examples.
- The G boundary summarizes the information from previously encountered negative examples. Any hypothesis more specific than G is assured to be consistent with past negative examples.
- The learned version space is independent of the sequence in which the training examples are presented.
- With more training data, the S and G boundaries will move monotonically closer to each other, delimiting a smaller and smaller version space of candidate hypotheses.

2.6 REMARKS ON VERSION SPACES AND CANDIDATE-ELIMINATION ALGORITHM

a. Will the Candidate-Elimination algorithm converge to the correct Hypothesis?

The version space learned by the Candidate-Elimination Algorithm will converge toward the hypothesis that correctly describes the target concept, provided

– There are no errors in the training examples, and

– there is some hypothesis in H that correctly describes the target concept.

• What will happen if the training data contains errors?

– The algorithm removes the correct target concept from the version space.

– S and G boundary sets eventually converge to an empty version space if sufficient additional training data is available.

– Such an empty version space indicates that there is no hypothesis in H consistent with all observed training examples.

• A similar symptom will appear when the training examples are correct, but the target concept cannot be described in the hypothesis representation.

– e.g., if the target concept is a disjunction of feature attributes and the hypothesis space supports only conjunctive descriptions

b. What Training Example Should the Learner Request Next?

We have assumed that training examples are provided to the learner by some external teacher.

• Suppose instead that the learner is allowed to conduct experiments in which it chooses the next instance, then obtains the correct classification for this instance from an external oracle (e.g., nature or a teacher).

– This scenario covers situations in which the learner may conduct experiments in nature or in which a teacher is available to provide the correct classification.

– We use the term query to refer to such instances constructed by the learner, which are then classified by an external oracle.

• Considering the version space learned from the four training examples of the EnjoySport concept.

– What would be a good query for the learner to pose at this point?

– What is a good query strategy in general?

-The learner should attempt to discriminate among the alternative competing hypotheses in its current version space.

– Therefore, it should choose an instance that would be classified positive by some of these hypotheses, but negative by others.

– One such instance is <Sunny, Warm, Normal, Light, Warm, Same>

– This instance satisfies three of the six hypotheses in the current version space.

– If the trainer classifies this instance as a positive example, the S boundary of the version space can then be generalized.

– Alternatively, if the trainer indicates that this is a negative example, the G boundary can then be specialized.

• In general, the optimal query strategy for a concept learner is to generate instances that satisfy exactly half the hypotheses in the current version space.

Instance	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
A	Sunny	Warm	Normal	Strong	Cool	Change	?
B	Rainy	Cold	Normal	Light	Warm	Same	?
C	Sunny	Warm	Normal	Light	Warm	Same	?
D	Sunny	Cold	Normal	Strong	Warm	Same	?

• When this is possible, the size of the version space is reduced by half with each new example, and the correct target concept can therefore be found with only

$\log_2 |VS|$ experiments.

c. How can Partially Learned Concepts be used?

Even though the learned version space still contains multiple hypotheses, indicating that the target concept has not yet been fully learned, it is possible to classify certain examples with the same degree of confidence as if the target concept had been uniquely identified.

Let us assume that the followings are new instances to be classified:

Instance A was is classified as a positive instance by every hypothesis in the current version space.

• Because the hypotheses in the version space unanimously agree that this is a positive instance, the learner can classify instance A as positive with the same confidence it

would have if it had already converged to the single, correct target concept.

• Regardless of which hypothesis in the version space is eventually found to be the correct target concept, it is already clear that it will classify instance A as a positive example.

• Notice furthermore that we need not enumerate every hypothesis in the version space in order to test whether each classifies the instance as positive.

– This condition will be met if and only if the instance satisfies every member of S.

– The reason is that every other hypothesis in the version space is at least as general as some member of S.

– By our definition of more-general-than, if the new instance satisfies all members of S it must also satisfy each of these more general hypotheses.

Instance B is classified as a negative instance by every hypothesis in the version space.

– This instance can therefore be safely classified as negative, given the partially learned concept.

– An efficient test for this condition is that the instance satisfies none of the members of G.

• Half of the version space hypotheses classify ***instance C*** as positive and half classify it as negative.

– Thus, the learner cannot classify this example with confidence until further training examples are available.

• ***Instance D*** is classified as positive by two of the version space hypotheses and negative by the other four hypotheses.

– In this case we have less confidence in the classification than in the unambiguous cases of instances A and B.

– Still, the vote is in favor of a negative classification, and one approach we could take would be to output the majority vote, perhaps with a confidence rating indicating how close the vote was.

2.7 INDUCTIVE BIAS

The **Candidate-Elimination Algorithm** will converge toward the true target concept provided it is given accurate training examples and provided its initial hypothesis space contains the target concept.

Fundamental Questions of Inductive Inference:
- ✓ What if the target concept is not contained in the hypothesis space?
- ✓ Can we avoid this difficulty by using a hypothesis space that includes every possible hypothesis?
- ✓ How does the size of this hypothesis space influence the ability of the algorithm to generalize to unobserved instances?
- ✓ How does the size of the hypothesis space influence the number of training examples that must be observed?

2.7.1 A Biased Hypothesis Space:

✓ Suppose we wish to assure that the hypothesis space contains the unknown target concept. The obvious solution is to enrich the hypothesis space to include *every possible* hypothesis.

✓ To illustrate, consider again the **EnjoySport** example in which we restricted the hypothesis space to include only conjunctions of attribute values.

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

✓ Because of this restriction, the hypothesis space is unable to represent even simple disjunctive target concepts such as "*Sky = Sunny* or *Sky = Cloudy*".

✓ In fact, given the three training examples (as in above table) of this disjunctive hypothesis, our algorithm would find that there are zero hypotheses in the version space.

✓ To see why there are no hypotheses consistent with these three examples, note that the most specific hypothesis consistent with the first two examples and representable in the given hypothesis space H is

$$S2 : <?, \textit{Warm, Normal, Strong, Cool, Change}>$$

✓ This hypothesis, although it is the maximally specific hypothesis from H that is consistent with the first two examples, is already overly general: it incorrectly covers the third (negative) training example.

✓ The problem is that we have biased the learner to consider only conjunctive hypotheses. In this case we require a more expressive hypothesis space.

2.7.2 An Unbiased Learner:

• The obvious solution to the problem of assuring that the target concept is in the hypothesis space H is to provide a hypothesis space

capable of representing **every teachable concept;** that is, it is capable of representing every possible subset of the instances X. In general, the set of all subsets of a set X is called the **power set** of X.

- In our initial **EnjoySport** learning task the size of the instance space X described by the six available attributes is 96. So, there are 2^{96}, or approximately 10^{28} distinct target concepts that could be defined over this instance space and that our learner might be called upon to learn.

- But the conjunctive hypothesis space is able to represent only 973 of these. Hence conjunctive hypothesis representation is very biased hypothesis space.

- To make our hypotheses space bias-free, consider the power set of instance space as hypotheses space.

- Then any Hypothesis can be represented with disjunctions, conjunctions, and negations of our earlier hypotheses.

- For instance, the target concept **"Sky = Sunny or Sky = Cloudy"** could then be described as

<center>**<Sunny,** ?, ?, ?, ?, ?> v **<Cloudy,** ?, ?, ?, ?, ?></center>

- This way of representing hypothesis space eliminates any problems of expressibility but it unfortunately raises a new, equally difficult problem - our concept learning algorithm is now completely unable to generalize beyond the observed examples.

- For instance, consider three positive examples (xl, x2, x3) and two negative examples (x4, x5) to the learner. Then S : { x1 V x2 V x3 } and G : { ⌐ (x4 V x5) }

- Therefore, the only examples that will be unambiguously classified by S and G are the observed training examples themselves.

- The classification of all the unobserved instances by taking a vote in this power set representation of hypotheses space will be futile as each unobserved instance will be classified positive by precisely half

the hypotheses in the version space and will be classified negative by the other half.

2.7.3 The Futility of Bias-Free Learning:

- The Fundamental property of inductive inference is: "*a learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances*".

- Inductive learning requires some form of prior assumptions, or inductive bias.

- General setting of an arbitrary learning problem:

    A learning algorithm L is provided by an arbitrary set of training data D_c = {<x, c(x)>} of some arbitrary target concept c. After training, L is asked to classify a new instance x_i. Let $L(x_i, D_c)$ denote the classification (e.g., positive or negative) that L assigns to x_i after learning from the training data D_c. We can describe this inductive inference step performed by L as follows:

$$(D_c \wedge x_i) \succ L(x_i, D_c)$$

where the notation $y \succ z$ indicates that z is inductively inferred from y .

- **Inductive Bias:**

    Consider a concept learning algorithm *L* for the set of instances X. Let c be an arbitrary concept defined over X, and let D_c = {< *x* , *c(x)*>} be an arbitrary set of training examples of c. Let *L(xi, 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 for any target concept c and corresponding training examples *D_c*

$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

- **Inductive bias of Candidate-Elimination algorithm:** The target concept c is contained in the given hypothesis space H.
- **Below figure shows modelling of Inductive systems by equivalent deductive systems:**
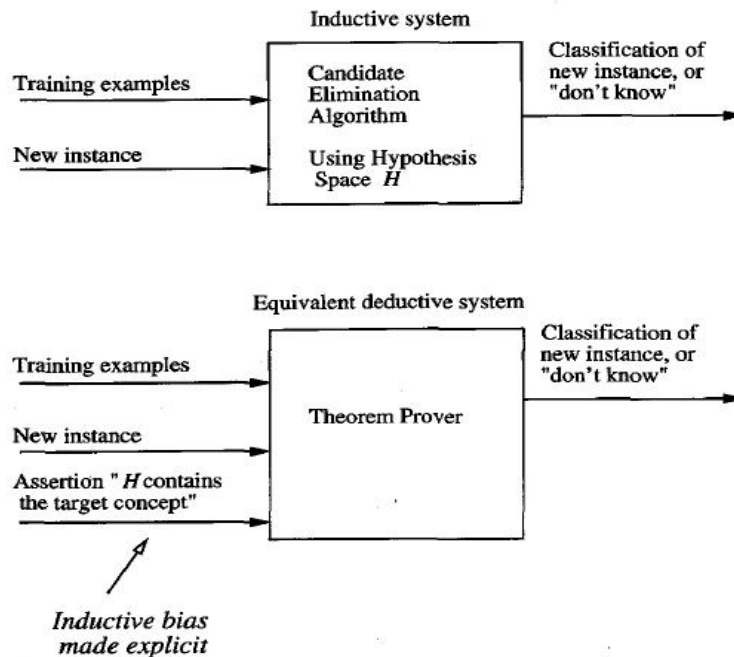
Advantages of viewing inductive inference systems in terms of inductive Bias:

- It provides a nonprocedural means of characterizing inductive inference systems policy for generalizing beyond the observed data.
- It allows comparison of different learners according to the strength of the inductive bias the inference systems employ.

Inductive Bias of Three learning Algorithms (weakest to strongest bias):

- **Rote-Learner:** Learning corresponds simply to storing each observed training example in memory. Subsequent instances are classified by looking them up in memory. If the instance is found in memory, the stored classification is returned. Otherwise, the system refuses to

classify the new instance. The Rote-Learner has no inductive bias and no additional assumptions required for classification.

- **Candidate-Elimination algorithm**: New instances are classified only in the case where all members of the current version space agree on the classification. Otherwise, the system refuses to classify the new instance. The **Candidate-Elimination algorithm** has a stronger inductive bias.

- **FIND-S**: This algorithm, finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances. The FIND-S algorithm has an even stronger inductive bias. The inductive bias is that the target concept can be represented in its hypothesis space, and all instances are negative instances unless the opposite is entailed by its other know1edge.

UNIT-II
SECTION-A

Objective Questions

1. Consider the following instances and hypothesis. identify the correct statement   [   ]

   x1= <Sunny, Warm, High, Strong, Cool, Same>
   x2= <Sunny, Warm, High, Light, Warm, Same>

   hl= <Sunny, ?, ?, Strong, ?, ?>
   h2= <Sunny, ?, ?, ?, ?, ?>
   h 3 = <Sunny, ?, ?, ?, Cool, ?>

A. h3 is more general than h1   B. h1 is more general than h2
C. h2 is more general than h1   D. All of the above

2. Which of the following is a negative example of EnjoySport learning task?
                                                              [   ]

A. h=<Sunny, Warm, Normal, Strong, Warm, Same>

B. h=<Sunny, Warm, ?, Strong, Warm, Same>

C. h=<Sunny, Warm,?, Strong, Warm, Same>

D. h=<Sunny, Warm, ?,Strong,?,?>

3. Which of the following is a generalization of h with respect to FIND-S algorithm?   [   ]

A. h=<Sunny, Warm, Normal, Strong, Warm, Same>

B. h=<Sunny, Warm, ?, Strong, Warm, Same>

C. h=<Sunny, Warm,?, Strong, Warm, Same>

D.. h=<Sunny, Warm, ?,Strong,?,?>

4.Identify   the   incorrect   statement   for   FIND-S   algorithm.
                                                              [   ]
A. Can't tell whether it has learned concept.
B. Can't tell when training data consistent.

C. Picks a maximally specific h

D. depending on H, there are several h.

5. Pick the correct hypothesis representation for the task: [   ]
Consider the set of all pairs of people where the first is a tall male(of any nationality and hair color) and the second is a Japanese female(of any hair color and height)
A.<<male ? tall ?><female ? ? Japanese>>
B. <<? ? tall ?><female ? ? Japanese>>
C.<< ? ? tall ?><female ? ? ?>>
D.<<male ? tall ? >< female ? ? ?>>

6. The size of hypothesis space in the given learning task is:
   Sex(male,female),Clor(black,brown,blonde),Height(tall,medium,short),Nationality(US,French,German,Trish,Indian,Japanese,Portuguese) [   ]
A. 384   B.385   C. 383   D. 386

7. Which of the following learning algorithm has no inductive bias
                                                                    [   ]
A. Rote Learner B. Candidate-Elimination C. FIND-S D. None of the above.

8. Which of the following statement is true about Candidate-elimination algorithm?   [   ]
A. Candidate-elimination algorithm uses general-to-specific ordering to find version space.
B. Candidate-elimination algorithm works well on noisy data.
C. Candidate-elimination algorithm searches the hypothesis space completely.
D. Both A and C

SECTION-B

Descriptive Questions

1. What is Concept Learning?

2. Elaborate on Find-S algorithm.

3. Write the steps for LIST-THEN-ELIMINATE algorithm.

4. Write the steps for CANDIDATE_ELIMINATION algorithm using version spaces.

5. Write the expression for defining version spaces.

6. Depict the version space for EnjoySport task with its general and specific boundary sets.

7. Explain why the size of the hypothesis space in the EnjoySport learning task is 973.

8. How would the number of possible instances and possible hypotheses increase with the addition of the attribute WaterCurrent, which can take on the values Light, Moderate or Strong ?

9. How the most specific hypothesis is represented? Explain with an example.