Lecture Scribe for Machine Learning (CS60050)

Spring 2020-2021

Instructor: Prof. Aritra Hazra

CONCEPT LEARNING (Date: 8th January,2021 and 13th January, 2021)

- **Learning Diagram**

  So far in the previous lecture we have learnt about what is learning and is learning feasible? In that context Learning diagram is recapitulated in this lecture.

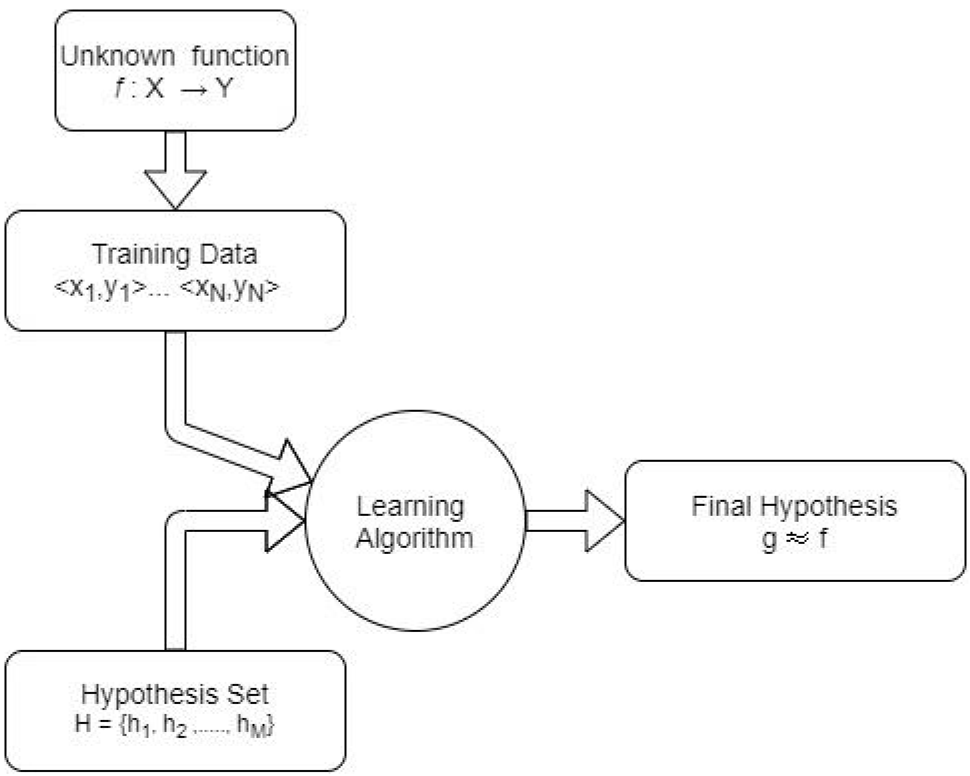
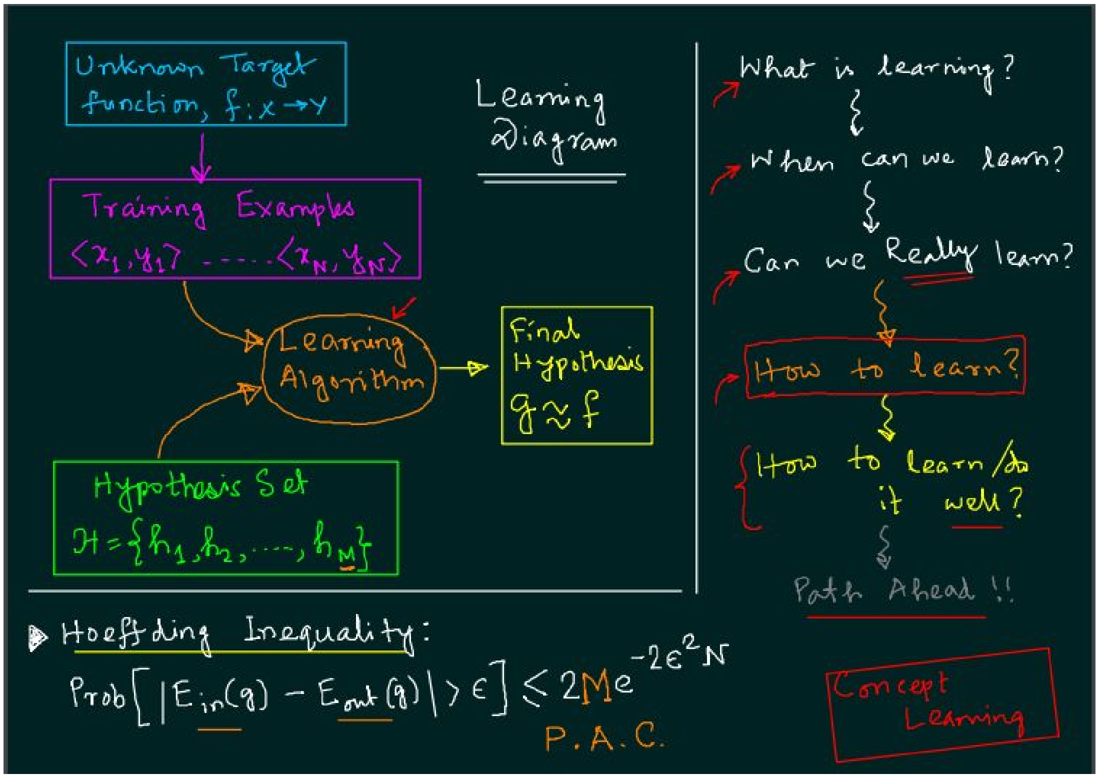


Fig1: Learning Diagram

We need to learn a function $f : X \to Y$, the unknown target. In our hand, we have the training data set, i.e., $\{<x_1, y_1>, <x_2, y_2>, \cdots <x_N, y_N>\}$. We are now trying to write our learning algorithm, which uses these training data and tries to figure out a function, say g, which relates f approximately , and we denote this g as the hypothesis. Specifically, we want to choose the function g from the hypothesis set H, where H = $h_1, h_2, \ldots h_M$. Therefore, we may say $g = h_i$, where $i \in 1$ to M.



Refer Handout-02-a slide-1

- **Concept Learning**

In a concept learning task, a human or machine learner is trained to classify objects by being shown a set of example objects along with their class labels. The learner simplifies what has been observed by condensing it in the form of an example. This simplified version of what has been learned is then applied to future examples.

In Concept Learning our task is to derive a Boolean function from training examples.

We can have many "hypothetical" boolean functions that form our Hypotheses set; from that we have to figure out a certain hypothesis h such that h = c where c is our target concept that defines the system.

Although we can have many complex examples where we have to derive non boolean functions, for simplicity we will consider boolean function examples.

Example: Suppose we are having a scenario where based on certain attribute inputs we are deciding some event "EnjoySport" as Yes or No.

The attributes are ***Sky, AirTemp, Humid, Wind, Water, Forecast*** .

Suppose we are having the following 4 training examples. Now our task is to find an appropriate set of hypotheses based on the concept we can acquire from the given training data. Based on the hypothesis set we will gradually move to our goal of having a hypothesis say h, such that $\forall x \in X$, $h(x) = c(x)$.

Sky	AirTemp	Humid	Wind	Water	Forecast	EnjoySport
Sunny	**Warm**	**Normal**	**Strong**	**Warm**	**Same**	**Yes**
Sunny	**Warm**	**High**	**Strong**	**Warm**	**Same**	**Yes**
Rainy	**Cold**	**High**	**Strong**	**Warm**	**Change**	**No**
Sunny	**Warm**	**High**	**Strong**	**Cool**	**Change**	**Yes**

Now in this particular example, let's count the number of possible instances of the input attributes.
The Sky attribute can have values within this set *{Sunny, Rainy, Cloudy}* So, |Sky| = 3
Similarly we are having, |Temp| = |Humid| = |Wind| = |Water| = |Forecast| = 2

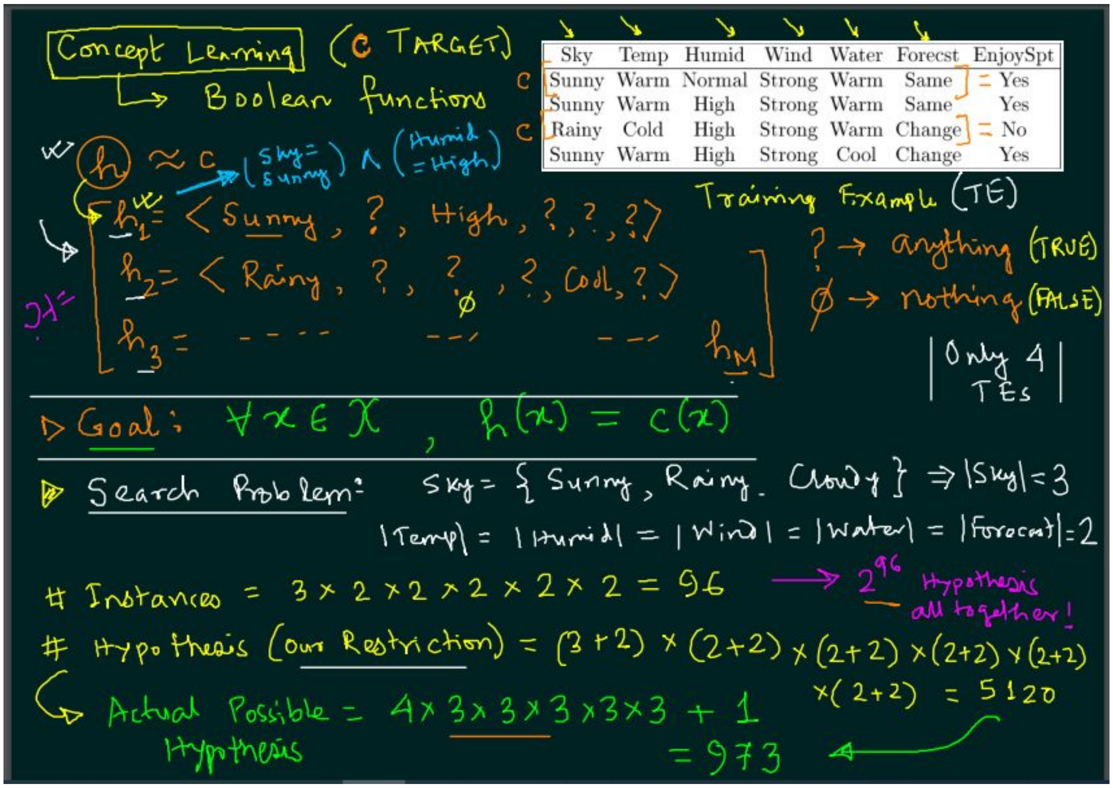
Thus total possible distinct instances are: $|\boldsymbol{X}| = 3 * 2 * 2 * 2 * 2 * 2 = 96$
Now from these distinct instances EnjoySport can be either 0 or 1. So the maximum number of hypotheses will be $|\mathbf{H}| = 2^{96}$ which is highly inefficient even for a small EnjoySport example.

To tackle this scenario the idea of **Inductive Learning Hypothesis** is important.

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.
So the choice of the hypothesis space can reduce the number of hypotheses. So let's see how we should represent the hypothesis.

Representing Hypotheses:

Represent hypothesis as **Conjunction of constraints** of the following form:
– Values possible in any hypothesis
  Specific value : Water Warm
  Don't-care value: Water: ? (anything permissible value)
  No value allowed : Water: Φ (nothing permissible value)
– Use a vector of such values as hypothesis:
Attributes: < Sky AirTemp Humid Wind Water Forecast >
For example : < Sunny ? ? Strong ? Same >
Based on our constraint (including ?s & Φs) and total number of possible hypothesis will be,
$|\boldsymbol{H}| = (3+2)(2+2)(2+2)(2+2)(2+2)(2+2)] = 5 * 4 * 4 * 4 * 4 * 4 = 5120 << 2^{96}$
Further, when we consider the hypothesis with Φs
semantically distinct h's will be $|\boldsymbol{H}| = 4 * 3 * 3 * 3 * 3 + 1 = 973$
Thus, with our constraint we have reduced the actual number of hypotheses much from the inefficient number 2^{96}.

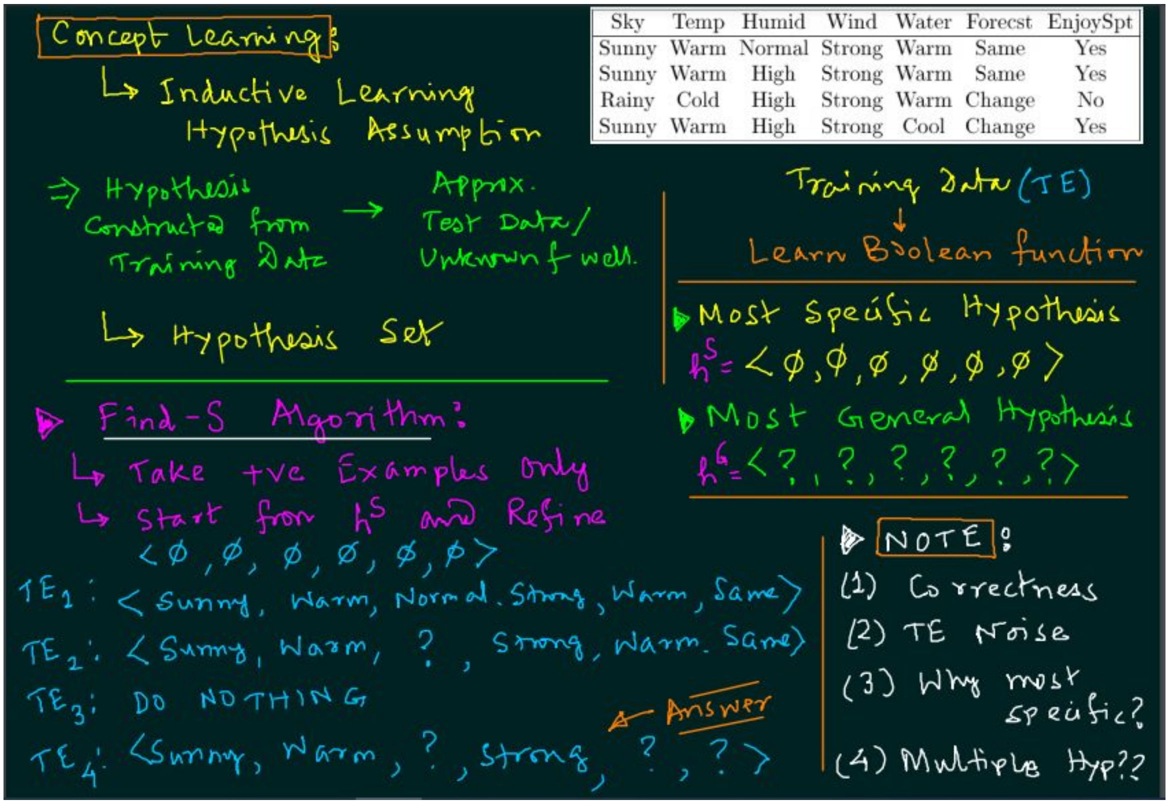


Refer Handout-02-a slide-2

- **Find-S algorithm** is a basic concept learning algorithm in machine learning. Find-S algorithm finds the most specific hypothesis that fits all the positive examples. We have to note here that the algorithm considers only those positive training examples. Find-S algorithm starts with the most specific hypothesis and generalizes this hypothesis each time it fails to classify an observed positive training data. Hence, the Find-S algorithm moves from the most specific hypothesis to the most general hypothesis.
  Here we take the assumption that there is hypothesis h in H describing target function c and there are no errors in the Training Examples (TEs).
  The most **general hypothesis** is represented by: $h^G = <?, ?, ?, ?, ?, ?>$
  The most **specific hypothesis** is represented by : $h^S = \boldsymbol{<\Phi, \Phi, \Phi, \Phi, \Phi, \Phi>}$

  Steps Involved In Find-S :

  1. Start with the most specific hypothesis. $h^S = \boldsymbol{<\Phi, \Phi, \Phi, \Phi, \Phi, \Phi>}$
  2. Take the next example and if it is negative, then no changes occur to the hypothesis.
  3. If the example is positive and we find that our initial hypothesis is too specific then we update our current hypothesis to general condition.
  4. Keep repeating the above steps till all the training examples are complete.
  5. After we have completed all the training examples we will have the final hypothesis when can be used to classify the new examples.



Refer Handout-02-a slide-3

So let's start with **<Φ, Φ, Φ, Φ, Φ, Φ>**

TE$_1$ <Sunny, Warm, Normal, Strong, Warm, Same> → YES

So, h$_1$ = <Sunny, Warm, Normal, Strong, Warm, Same>

TE$_2$ <Sunny, Warm, High, Strong, Warm, Same> → YES

So, h$_2$ = <Sunny, Warm, ?, Strong, Warm, Same>

TE$_3$ <Cold, Warm, High, Strong, Warm, Change> → NO

Same as h$_2$

TE$_2$ <Sunny, Warm, High, Strong, Cool, Change> → YES

So, h$_3$ = <Sunny, Warm, ? , Strong, ?, ?>

Thus based on find S Algorithm the most specific Hypothesis is **<Sunny,Warm,?, Strong, ?, ?>**

Problems with find S Algorithm:

1. Throws away information! (Negative examples)
2. Can't tell whether it has learned the concept (Depending on H, there might be several h's that fit TEs!)
3. Can't tell when training data is inconsistent (Since ignores negative TEs)

By keeping all consistent hypotheses we can tackle these problems.

Consistent Hypotheses

A hypothesis h is consistent with a set of training examples *D* of target concept *c if h(x) = c(x)* for each training example *<x, c(x)>* in *D*.

Notation: *Consistent(h, D)* ≡ ∀ *<x, c(x)>* ∈ *D, h(x) = c(x)*

- ## VERSION SPACE

A version space is a hierarchical representation of knowledge that enables you to keep track of all the useful information supplied by a sequence of learning examples without remembering any of the examples.

Version space basically represents the intermediate hypotheses between the most specific and most general hypothesis.

The version space, $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with D.

Notation: $VS_{H,D} = \{h \mid h \in H \text{ and Consistent } (h, D)\}$

The **general boundary**, G, of version space $VS_{H,D}$ is the set of its maximally general members consistent with D. It Summarizes the negative examples. Anything more general will cover a negative TE •

The **specific boundary**, S, of version space $VS_{H,D}$ is the set of its maximally specific members consistent with D. It Summarizes the positive examples. Anything more specific will fail to cover a positive TE.

Representing Version Spaces

- Store most/least general boundaries of space
- Generate all intermediate h's in VS
- Idea that any h in VS must be consistent with all TE's
- Generalize from most specific boundaries
- Specialize from most general boundaries

In the previous example we have the most specific hypothesis is **<Sunny,Warm,?, Strong, ?, ?>**

General hypotheses for the given examples will be <Sunny, ?, ?, ?, ?, ? >, <?, Warm, ?, ?, ?, ?>

Now gradually we will move from the general hypotheses towards the most specific hypotheses by specialising the features so that it gets consistent with the target concept.

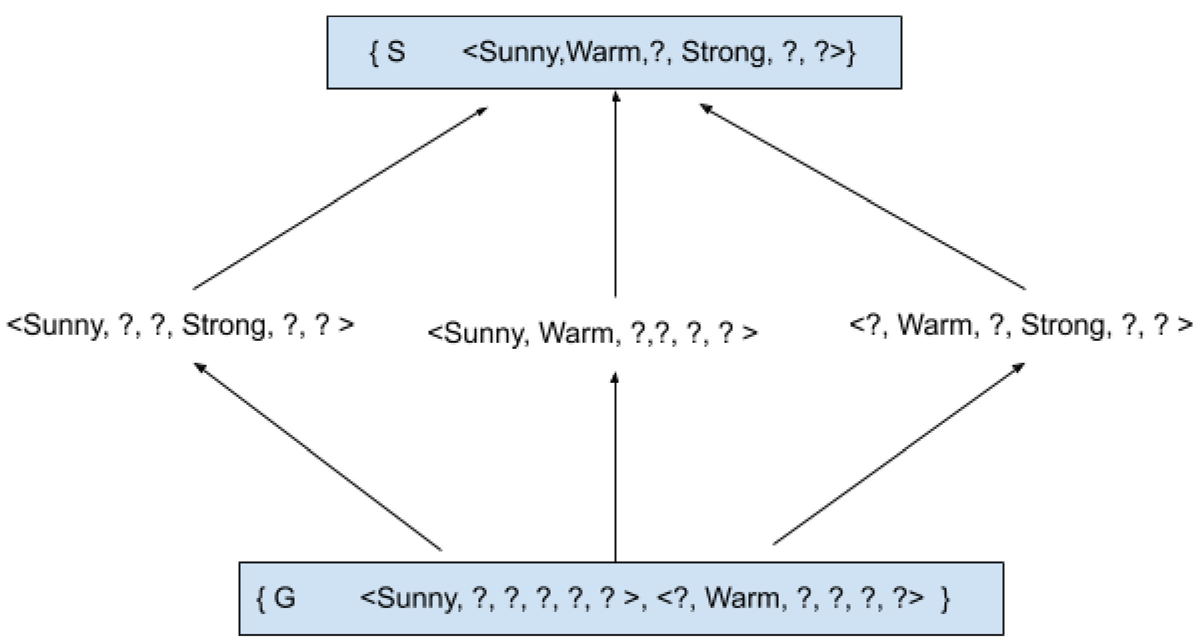
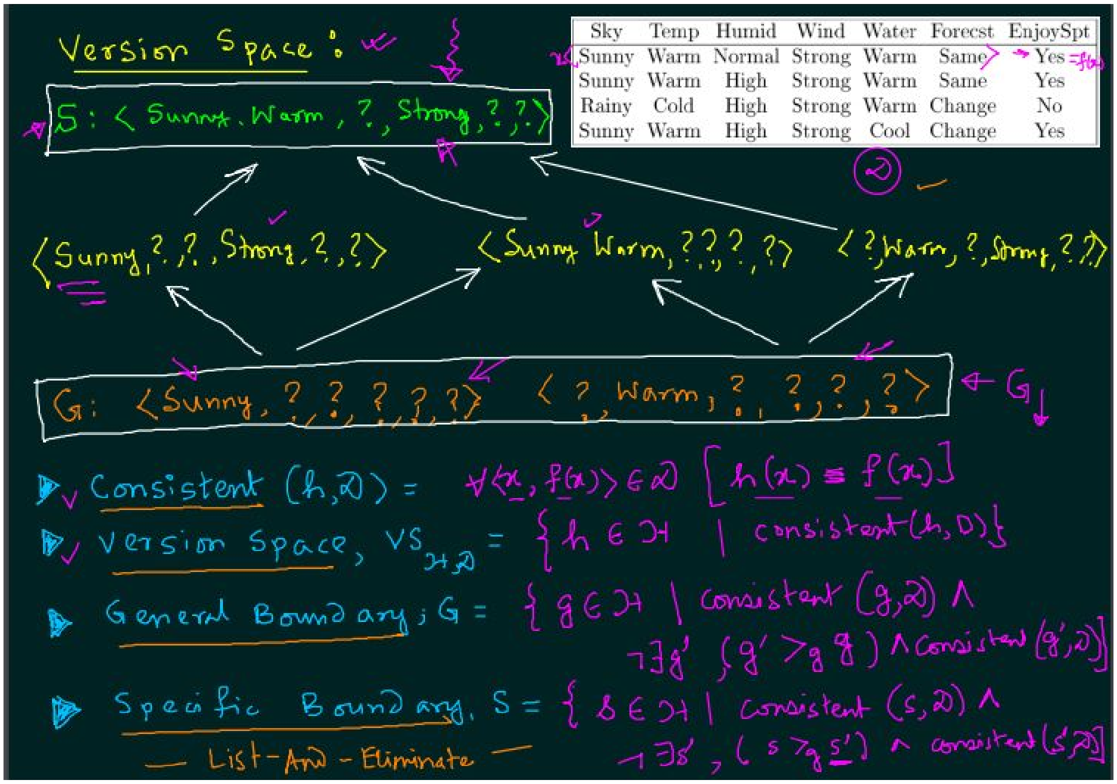


Fig 2: Version Space for this Example



Refer Handout-02-b Slide - 5

- **Candidate Elimination Algorithm:**

The candidate elimination algorithm incrementally builds the version space given a hypothesis space H and a set E of examples. The examples are added one by one; each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example. The candidate elimination algorithm does this by updating the general and specific boundary for each new example.

- We can consider this as an extended form of Find-S algorithm.
- It considers both positive and negative examples.
- Actually, positive examples are used here as the Find-S algorithm (Basically they are generalizing from the specification).
- While the negative example is specified from generalize form.

Steps Involved:

Step1: Load Training Examples

Step2: Initialize General Hypothesis and Specific Hypothesis.

Step3: For each training example

Step4: If example is positive example

if attribute_value == hypothesis_value:

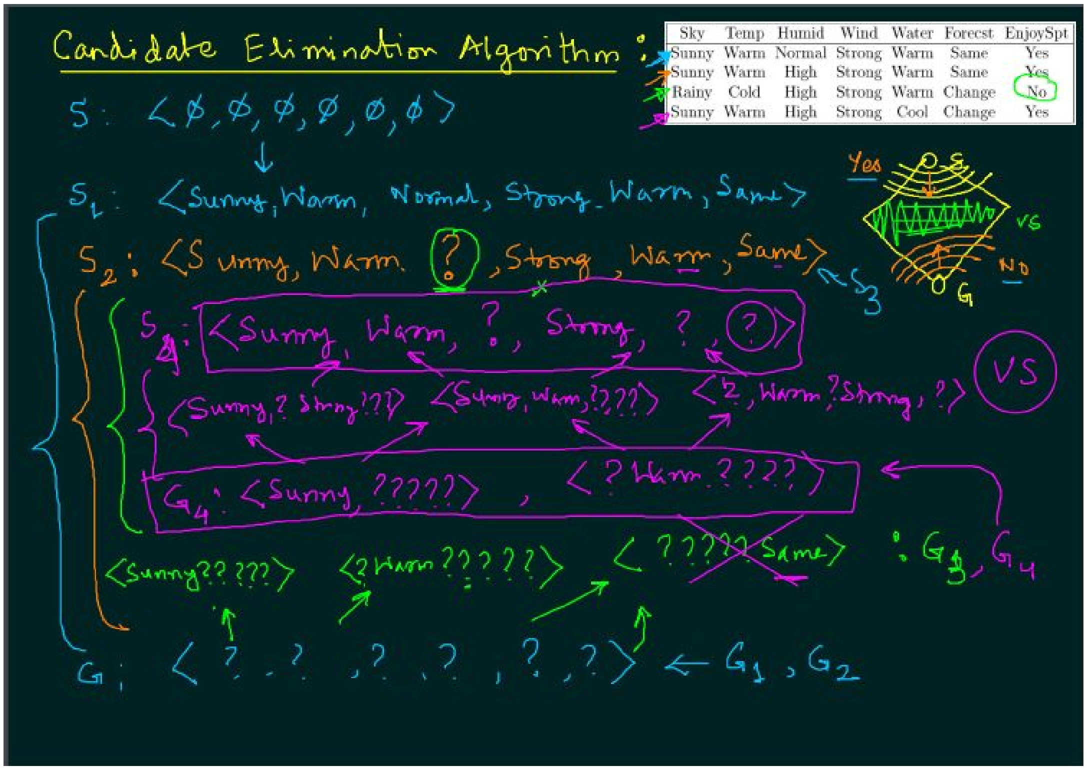
Do nothing

else:

replace attribute value with '?' (Basically generalizing it)

Step5: If example is Negative example

Make the generalized hypothesis more specific.



Refer Handout-02-b Slide-6

Initially : G = {<?, ?, ?, ?, ?, ?>}

S = [**Φ**, **Φ**, **Φ**,**Φ**, **Φ**, **Φ**]

For TE_1 : <'sunny','warm','normal','strong','warm ','same'> and positive output.

G_1 = G

S_1 = <'sunny','warm','normal','strong','warm ','same'>

For TE_2 : <'sunny','warm','high','strong','warm ','same'> and positive output.

G_2 = G

S_2 = <'sunny','warm',?,'strong','warm ','same'>

For TE_3 : <'rainy','cold','high','strong','warm ','change'> and negative output.

G_3 = {<'sunny', ?, ?, ?, ?, ?>, <?, 'warm', ?, ?, ?, ?>, <?, ?, ?, ?, ?, ?>,
<?, ?, ?, ?, ?, ?>, <?, ?, ?, ?, ?, ?>, <?, ?, ?, ?, ?, 'change'>}

$S_3 = S_2$

For TE_4 : <'sunny','warm','high','strong','cool','change'> and positive output.

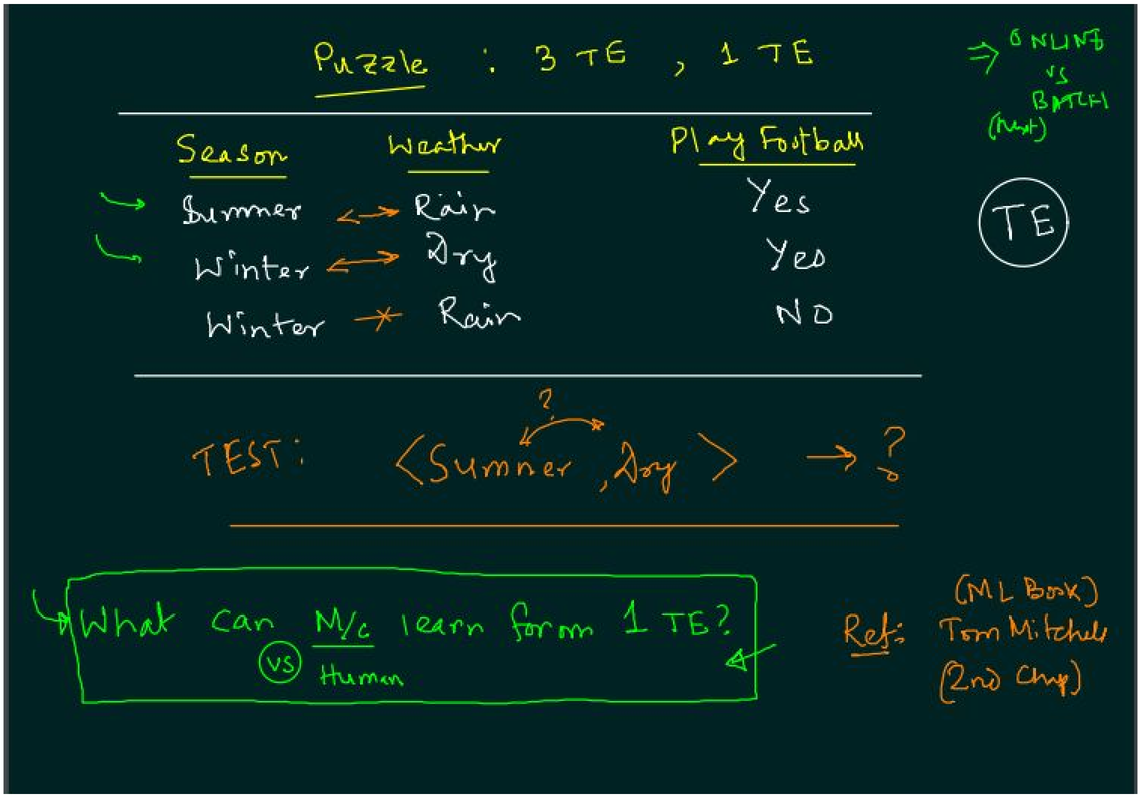
$G_4 = G_3$

S_4 = <'sunny','warm',?,'strong', ?, ?>

At last, by synchronizing the G_4 and S_4 algorithms produce the output.

G = {<'sunny', ?, ?, ?, ?, ?>, <?, 'warm', ?, ?, ?, ?>}

S = <'sunny','warm',?,'strong', ?, ?>



Refer Handout-02b Slide-9

- **PUZZLE**

  Q1. Suppose we are having the following Training Examples

Season	**Weather**	**Play Football**
Summer	Rain	Yes
Winter	Dry	Yes
Winter	Rain	No

Figure out what will be the output for the test inputs <Summer, Dry>

Q2. What can machines learn with respect to humans from one Training Example?
Refer Machine Learning, Tom Mitchell

Prepared by ARGHA SEN