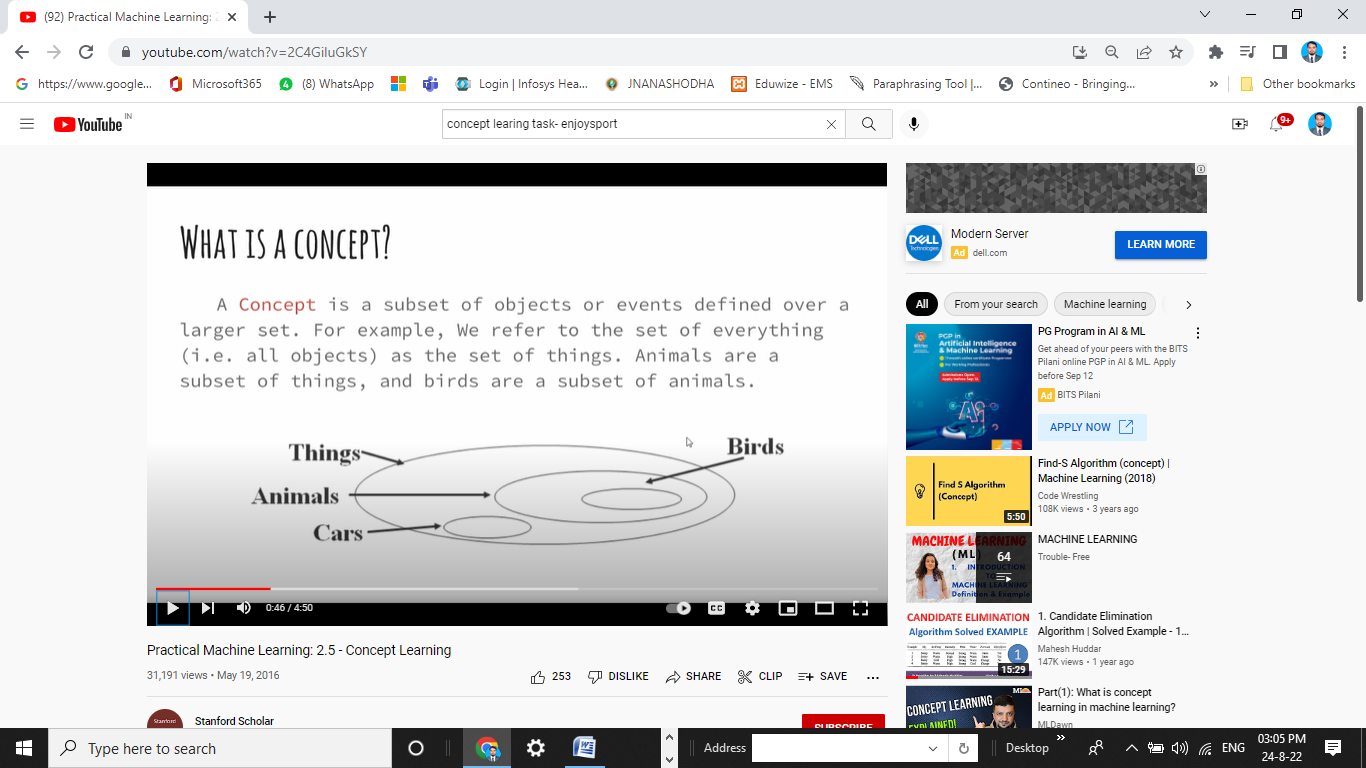
**Concept:**

concept is a **set of instances or objects sharing common characteristics.** It represents a category or class of items that share certain features or properties.

For example, in a simple scenario of learning about animals, "bird" can be considered a concept.

It **includes all instances of animals that have wings, feathers, lay eggs,** and typically can fly.

Each individual bird, such as a sparrow or eagle, is an instance of the concept "bird".



Similarly, in a more complex scenario like medical diagnosis, "healthy patient" and "disease" could be considered concepts.

Instances labeled as "healthy patient" would include individuals who exhibit typical vital signs and absence of symptoms, while instances labeled as "disease" would include individuals with specific symptoms and medical conditions.

**Concept Learning:**

Concept learning is the **process of learning a general definition of a concept from a set of examples**, where each example is labeled as either a member or a non-member of the concept.

It's a fundamental approach to automated learning from examples.

This process involves identifying common patterns or characteristics among the examples labeled as members of the concept in order to generalize and accurately classify new instances as members or non-members of the concept.

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

**Example:** Consider the task of learning the concept "Days on which a **PERSON** enjoys his favorite **water sport”.**

**The task is to learn to predict the value of EnjoySport for an arbitrary day**, **based on the values of its other attributes.**

Hypotheses are used in concept learning to represent and generalize the learned concept based on the provided examples. Each hypothesis representation consists of a conjunction of constraints on the instance attributes.

Let each hypothesis be a vector of six constraints, specifying the values of the six attributes

Sky, AirTemp, Humidity, Wind, Water, and Forecast.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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: Positive and negative training examples for the target concept EnjoySport.**

The value of each attribute of the hypotheses is indicated by either? or Φ

? – indicates the any value is acceptable for this attribute

Φ – means no value is acceptable

**Representation of Hypothesis:**

**Example: Hypothesis representation**  that person enjoys his favorite sport only on cold days with high humidity is **(?, Cold, High, ?, ?, ?)**

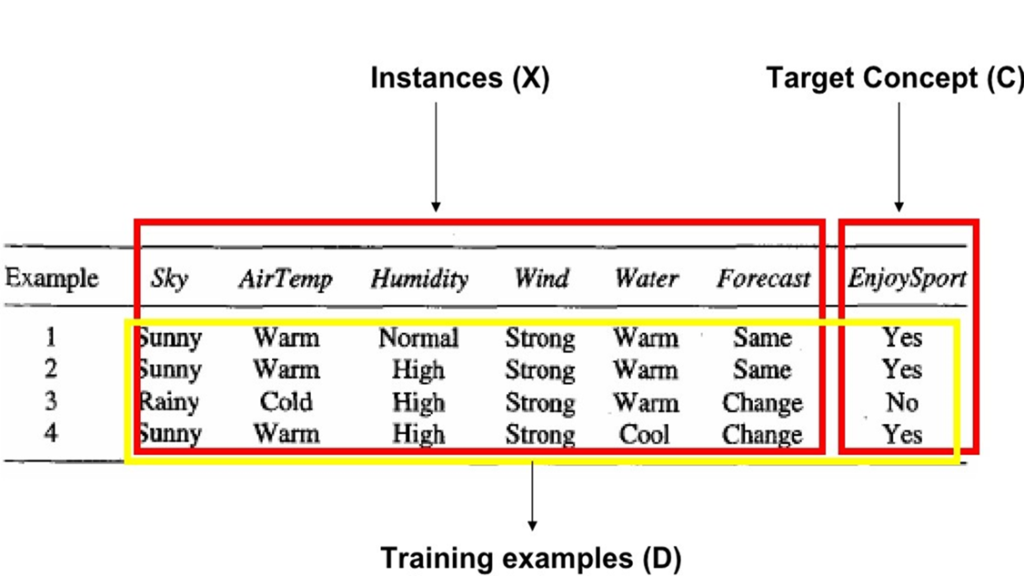
**Goal:** To infer the “best” concept-description from the set of all possible hypothesis.

**“best”** means “which **best generalizes** to all elements (known or unknown) of the instance space”

**Most general hypothesis:** Everyday is good day for water sports **<?, ?, ?, ?, ?, ?>**

**Most specific hypothesis**: no day is a good day for water sports **<Φ, Φ, Φ, Φ, Φ, Φ>**

**Notation**



Training examples (D) : <x, c(x)>

**Set of Instances (X):** Set of items over which the concept is defined is called the **set of instances**, which is denoted by X.

**Example:** X is the set of all possible days, each represented by the attributes: Sky, AirTemp, Humidity, Wind, Water, and Forecast

**Target concept (C):** The concept or function to be learned is called the **target concept**, which is denoted by c.

C can be any Boolean valued function defined over the instances X

c: X→ {O, 1}

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.

**Positive Examples or members of the target concept**: Instances: for which **c(x) = 1**

**Negative Examples or non-members of the target concept:** Instances for which **c(x) = 0**

**Training example representation D : <x, c(x)>**

The ordered pair **<x, c(x)>** is used to describe the training example consisting of the instance x and its target concept value **c(x)**. Represents set of available training examples

**H** – represents set of all possible hypotheses. H is determined by the human designer's choice of hypothesis representation. In general, each hypothesis **h in H represents a boolean-valued function defined over X that**

**is, h : X 🡪{O, 1}.**

**The goal of the of the concept-learning (learner) is to find a hypothesis h such that h (x) = c (x) for a x in X.**

**FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS**

**FIND-S Algorithm**

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x

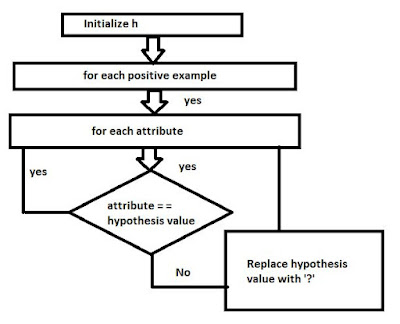
For each attribute constraint ai in h

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

3. Output hypothesis h



To illustrate this algorithm, assume the learner is given the sequence of training examples from the **EnjoySport** task

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |

Step1: Initialize h to the most specific hypothesis h: (Ø, Ø, Ø, Ø, Ø, Ø)

Step2: Consider the **first training example**  x1 = <Sunny Warm Normal Strong Warm Same>, +

hypothesis **h is too specific**. None of the "Ø" constraints in h are satisfied by this example, so each **is replaced by the next more general constraint** that fits the example [i.e. specific to general approach]

⸫ h1 = <Sunny Warm Normal Strong Warm Same>

Consider the second training example x2 = <Sunny, Warm, High, Strong, Warm, Same>, +

By considering second training example the algorithm further generalizes h, substituting a "?" in place of any attribute value in h that is not satisfied by the new example.

⸫ h2 = <Sunny Warm ? Strong Warm Same>

Consider the third training example x3 = <Rainy, Cold, High, Strong, Warm, Change>, -

Upon encountering the third training the algorithm makes no change to h. The FIND-S algorithm simply ignores every negative example.

⸫ h3 = < Sunny Warm ? Strong Warm Same>

Consider the fourth training example x4 = <Sunny Warm High Strong Cool Change>, +

The fourth example leads to a further generalization of h

[

⸫ h4 = < Sunny Warm ? Strong ? ? >

**Drawbacks [limitations/disadvantages] of Find-S algorithm**

**1 Does not consider negative examples as the result inconsistent training sets can actually mislead find-S**

**2. Hypothesis h return by Find-S may not be the only h that fits the data we can actually generate many hypothesis h1, h2,…..h**

**!!! To overcome these drawbacks Candidate elimination algorithm is used**