Module1 Notes (Pointwise)

Syllabus: Introduction: Well posed learning problems, Designing a Learning system, Perspective and Issues in Machine Learning. Concept Learning: Concept learning task, Concept learning as search, Find-S algorithm, Version space, Candidate Elimination algorithm, Inductive Bias.

1.0 What is Machine Learning ?

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

• Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

• The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide.

• The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

1.0.1 Some successful applications of machine learning

- Learning to recognize spoken words (Lee, 1989; Waibel, 1989).
- Learning to drive an autonomous vehicle (Pomerleau, 1989).
- Learning to classify new astronomical structures (Fayyad et al., 1995).
- Learning to play world-class backgammon (Tesauro 1992, 1995).

1.0.2 Why is Machine Learning Important?

- Some tasks cannot be defined well, except by examples (e.g., recognizing people).
- Relationships and correlations can be hidden within large amounts of data. Machine Learning/Data Mining may be able to find these relationships.
- Human designers often produce machines that do not work as well as desired in the environments in which they are used.

- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans (e.g., medical diagnostic).
- Environments change over time.
- New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design systems “by hand”.

1.0.3 Some disciplines of their influence on machine learning

- Artificial intelligence
- Bayesian methods
- Computational complexity theory
- Control theory
- Information theory
- Philosophy
- Psychology and neurobiology
- Statistics

1.0.4 Areas of Influence for Machine Learning

Statistics: How best to use samples drawn from unknown probability distributions to help decide from which distribution some new sample is drawn? •

Brain Models: Non-linear elements with weighted inputs (Artificial Neural Networks) have been suggested as simple models of biological neurons.

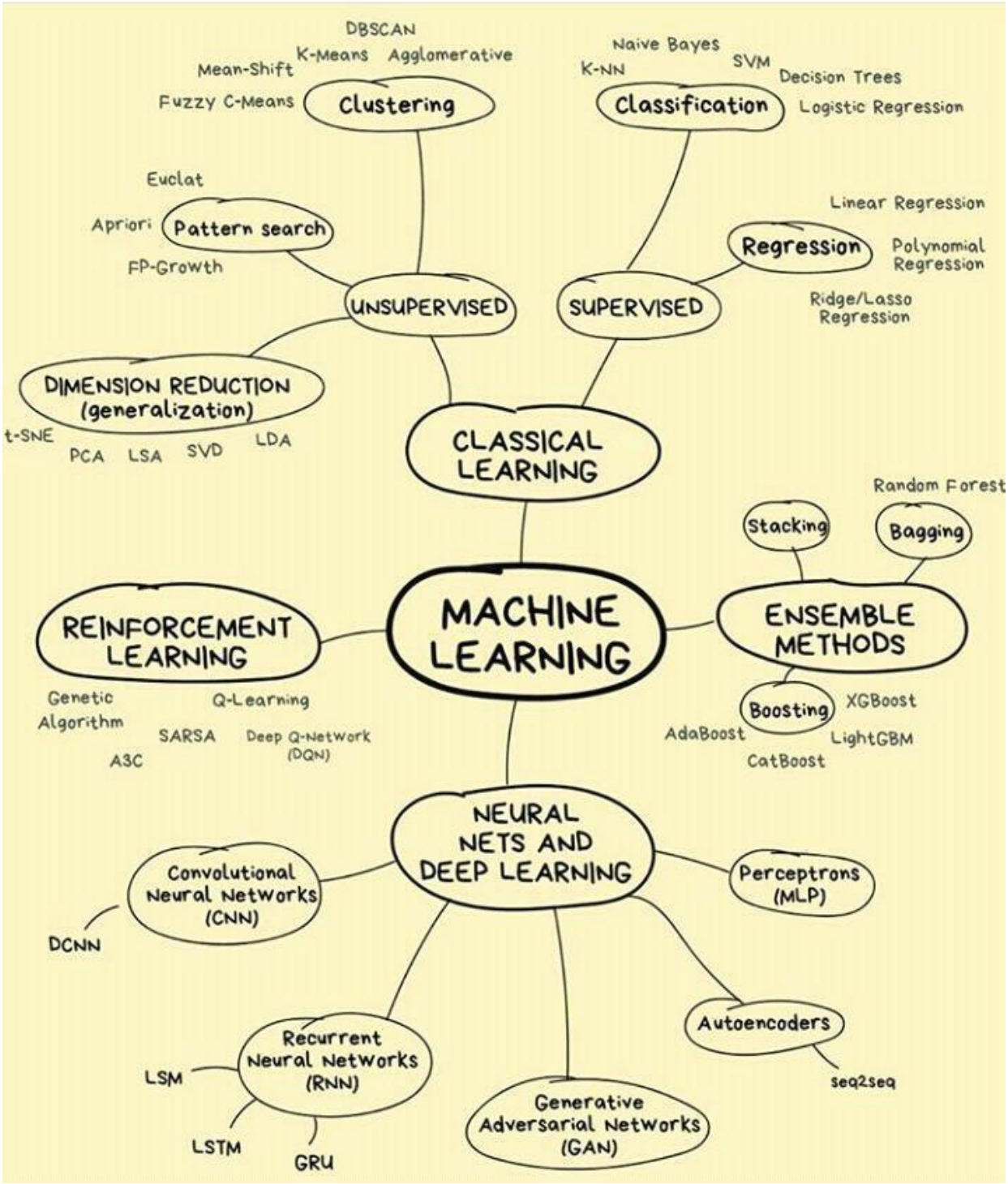
Adaptive Control Theory: How to deal with controlling a process having unknown parameters that must be estimated during operation?

Psychology: How to model human performance on various learning tasks?

Artificial Intelligence: How to write algorithms to acquire the knowledge humans are able to acquire, at least, as well as humans?

Evolutionary Models: How to model certain aspects of biological evolution to improve the performance of computer programs?

1.0.5 : Types of Machine Learning



1.2 Well-Posed Learning Problems:

The study of Machine learning is about writing software that improves its own performance with experience

Definition [Mitchell]: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Example1 : *A Checkers Learning Problem*

- **Task T** = Playing checkers
- **Performance Measure P** = Percentage of games won against opponent
- **Training Experience E** = Playing practice games against itself

Example2: *A Handwriting Recognition Learning Problem*

- **Task T** = Recognizing and classifying handwritten words
- **Performance Measure P** = Percentage of words correctly classified
- **Training Experience E** = A database of handwritten words with given classification

Example3 : A Robot driving learning program

- **Task T** = Driving on public four lane highways using vision sensors
- Performance Measure P = Average distance handled before error (as judged by human overseas)
- Training Experience E = A sequences of images and steering commands recorded while observing a human driver.

1.3. DESIGNING A LEARNING SYSTEM

Steps to design a learning system

0. Problem Description
1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm

1. ESTIMATING TRAINING VALUES
2. ADJUSTING THE WEIGHTS

5. The Final Design

1.3.0 Problem Description:

A Checker Learning Problem

- Task T: Playing Checkers
- Performance Measure P: Percent of games won against opponents
- Training Experience E: To be selected ==> Games Played against itself

1.3.1 Choosing the Training Experience

- Will the training experience provide direct or indirect feedback?
    - **Direct Feedback:** system learns from examples of individual checkers board states and the correct move for each
    - **Indirect Feedback**: Move sequences and final outcomes of various games played
        - **Credit assignment problem**: Value of early states must be inferred from the outcome
- Degree to which the learner controls the sequence of training examples
    - Teacher selects informative boards and gives correct move
    - Learner proposes board states that it finds particularly confusing. Teacher provides correct moves
    - Learner controls board states and (indirect) training classifications
- How well the training experience represents the distribution of examples over which the final system performance P will be measured
    - If training the checkers program consists only of experiences played against itself, it may never encounter crucial board states that are likely to be played by the human checkers champion
    - Most theory of machine learning rests on the assumption that the distribution of training examples is identical to the distribution of test examples

Partial Design of Checkers Learning Program

- A checkers learning problem:
    - Task T: playing checkers
    - Performance measure P: percent of games won in the world tournament
    - Training experience E: games played against itself

- Remaining choices
    - The exact type of knowledge to be learned
    - A representation for this target knowledge
    - A learning mechanism

1.3.2 Choosing the Target Function

- Assume that you can determine legal moves
- Program needs to learn the best move from among legal moves
    - Defines large search space known a priori
    - target function: ChooseMove: B → M
- ChooseMove is difficult to learn given indirect training
- Alternative target function
    - An evaluation function that assigns a numerical score to any given board state
    - V : B → R (where Ris the set of real numbers)
        - V(b) for an arbitrary board state b in B
        - if b is a final board state that is won, then V(b) = 100
        - if b is a final board state that is lost, then V(b) = -100
        - if b is a final board state that is drawn, then V(b) = 0
        - if b is not a final state, then V(b) = V(b '), where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game

- V(b) gives a recursive definition for board state b
    - Not usable because not efficient to compute except is first three trivial cases
    - nonoperational definition

• Goal of learning is to discover an operational description of V
• Learning the target function is often called function approximation,
• Referred to as V ˆ

1.3.3 Choosing a Representation for the Target Function

• Choice of representations involve trade offs
• Pick a very expressive representation to allow close approximation to the ideal target function V
• More expressive, more training data required to choose among alternative hypotheses
• Use linear combination of the following board features:
• x1: the number of black pieces on the board
• x2: the number of red pieces on the board
• x3: the number of black kings on the board
• x4: the number of red kings on the board
• x5: the number of black pieces threatened by red (i.e. which can be captured on red's next turn)
• x6: the number of red pieces threatened by black

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

Partial Design of Checkers Learning Program

A checkers learning problem:

• **Task T:** playing checkers
• **Performance measure P:** percent of games won in the world tournament
• **Training experience E:** games played against itself
• **Target Function:** V: Board → R
• **Target function representation**

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

3.4. Choosing a Function Approximation Algorithm

To learn we require a set of training examples describing the board b and the training value $V_{train}(b)$

• Ordered pair<b, $V_{train}(b)$>

$$\langle\langle x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0\rangle, +100\rangle$$

3.4.1 Estimating Training Values

Need to assign specific scores to intermediate board states

• Approximate intermediate board state b using the learner's current approximation of the next board state following b

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

• Simple and successful approach

• More accurate for states closer to end states

3.4.1 Adjusting the Weights

• Choose the weights wi to best fit the set of training examples

• Minimize the squared error E between the train values and the values predicted by the hypothesis

$$E \equiv \sum_{\langle b, V_{train}(b)\rangle \in training examples} \left(V_{train}(b) - \hat{V}(b)\right)^2$$

• Require an algorithm that

  • will incrementally refine weights as new training examples become available

  • will be robust to errors in these estimated training values

• Least Mean Squares (LMS) is one such algorithm

LMS Weight Update Rule

• For each train example $\langle b, V_{train}(b)\rangle$

  • Use the current weights to calculate $\hat{V}(b)$

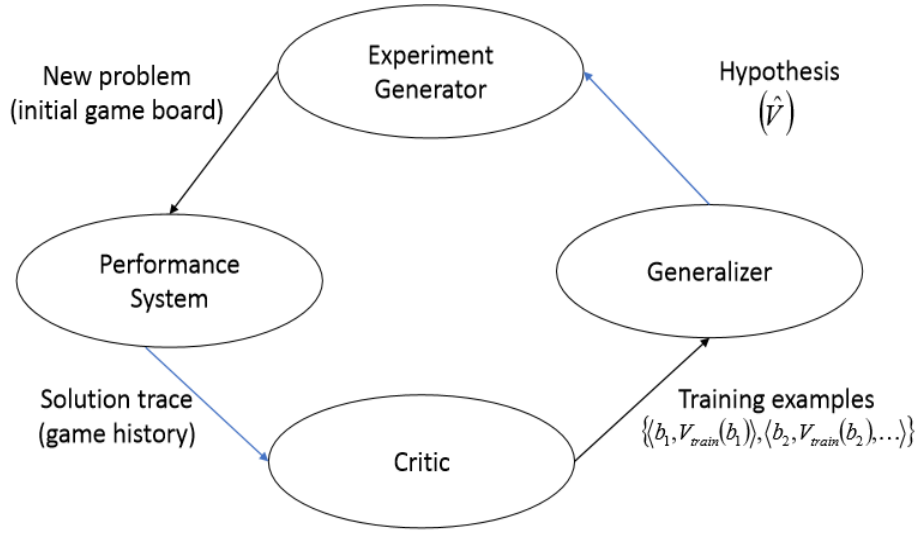
  • For each weight w_i, update it as

$$w_i \leftarrow w_i + \eta\left(V_{train}(b) - \hat{V}(b)\right)x_i$$

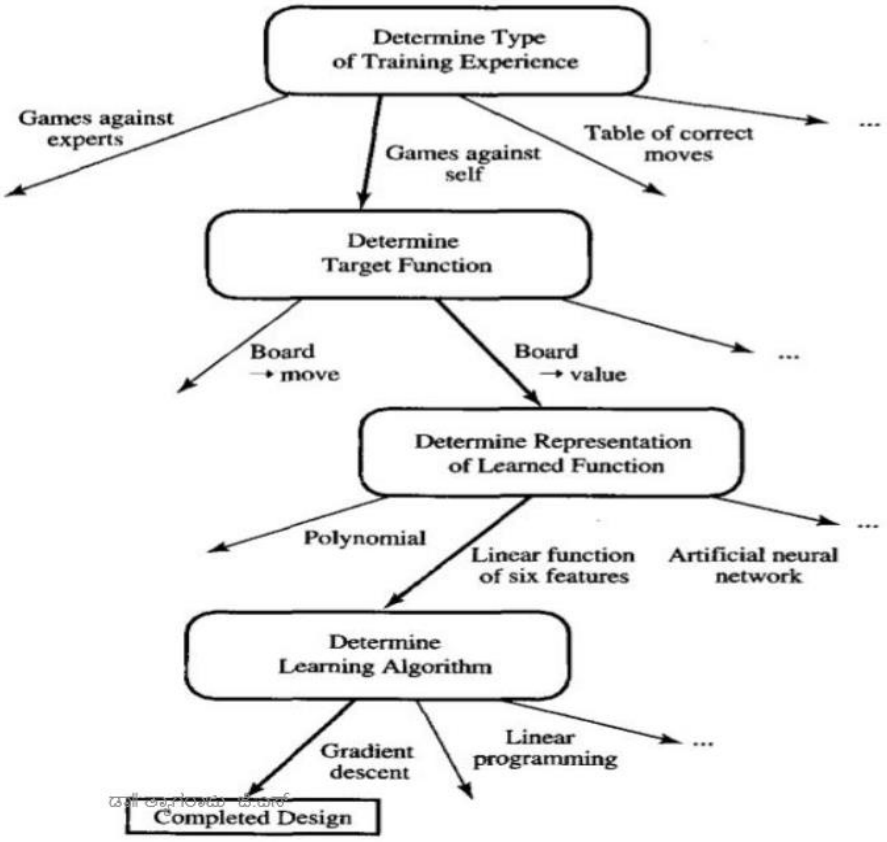
  • where

    • η is a small constant (e.g. 0.1)

3.5 Final Design



Summary of choices in designing the checkers learning program



1.4.0 Perspectives and Issues in Machine Learning

1.4.1 Perspectives in Machine Learning

• One useful perspective on machine learning is that it involves searching a very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by the learner.

• For example, consider the space of hypotheses that could in principle be output by the above checkers learner. This hypothesis space consists of all evaluation functions that can be represented by some choice of values for the weights wo through w6. The learner's task is thus to search through this vast space to locate the hypothesis that is most consistent with

• the available training examples. The LMS algorithm for fitting weights achieves this goal by iteratively tuning the weights, adding a correction to each weight each time the hypothesized evaluation function predicts a value that differs from the training value. This algorithm works well when the hypothesis representation considered by the learner defines a continuously parameterized space of potential hypotheses.

1.4.2 Issues in Machine Learning (i.e., Generalization)

• What algorithms exist for learning general target functions from specific training examples? In what settings will particular algorithms converge to the desired function, given sufficient training data? Which algorithms perform best for which types of problems and representations?

• How much training data is sufficient? What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space?

• When and how can prior knowledge held by the learner guide the process of generalizing from examples? Can prior knowledge be helpful even when it is only approximately correct? • What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?

• What is the best way to reduce the learning task to one or more function approximation problems? Put another way, what specific functions should the system attempt to learn? Can this process itself be automated?

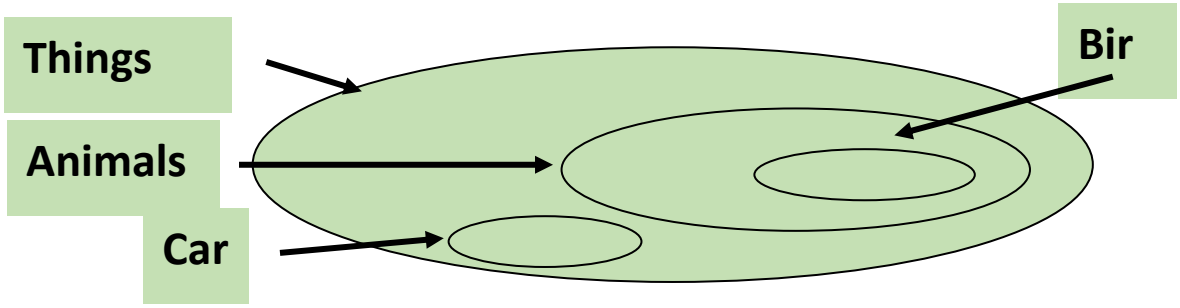
• How can the learner automatically alter its representation to improve its ability to represent and learn the target function?

Question Bank

1. Define Machine Learning. Discuss with examples Why Machine Learning is Important.
2. Discuss with examples some useful applications of machine learning
3. Explain how some areas/disciplines have influenced the Machine learning.
4. Define Learning Program for a given Problem. Describe the following problems with respect to Tasks, Performance and Experience:
    1. Checkers Learning Problems
    2. Handwritten Recognition Problem
    3. Robot Driving Learning Problem
5. Describe in detail all the steps involved in designing a Learning Systems
6. Discuss the Perspective and Issues in Machine Learning.

2.1 What is a Concept?

- A **Concept** is 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].

2.1 What is 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.

2.1 A CONCEPT LEARNING TASK

- Consider the example task of learning the target concept "*days on which my friend Aldo enjoys his favorite water sport*."
- Table describes a set of example days, each represented by a set of attributes.
- The attribute EnjoySport indicates whether or not Aldo enjoys his favorite water sport on this day.
- The task is to learn to predict the value of EnjoySport for an arbitrary day, based on the values of its other attributes.
- **Database:**

Day	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

Chosen Hypothesis Representation:

Conjunction of constraints on each attribute where:

- "?" means "any value is acceptable"
- "0" means "no value is acceptable"

- **Example of a hypothesis: <?, Cold, High, ?, ?, ?>** (If the air temperature is cold and the humidity high then it is a good day for water sports)
- **Goal:** To infer the "best" concept-description from the set of all possible hypotheses ("best" means "which best generalizes to all (known or unknown) elements of the instance space" concept-learning is an ill-defined task)
- **The most general hypothesis-that every day is a good day for water sports, positive example-is represented by <?, ?, ?, ?, ?, ?> and**
- **The most specific possible hypothesis-that no day is a positive example-is represented by <0,0,0,0,0,0>**

- **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*. The constraints may be "?" (any value is acceptable), "Ø" (no value is acceptable), or a specific value.
  - 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.

The *EnjoySport* concept learning task.

2.1 Example of a Concept Learning

- **Concept:** Good Days for Water Sports (values: Yes, No)
- **Attributes/Features:**
  - **Sky** (values: Sunny, Cloudy, Rainy)
  - **AirTemp** (values: Warm, Cold)
  - **Humidity** (values: Normal, High)
  - **Wind** (values: Strong, Weak)

- **Water** (Warm, Cool)
- **Forecast** (values: Same, Change)

- **Example of a Training Point:**

<Sunny, Warm, High, Strong, Warm, Same, Yes>

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.**

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.

Number of Instances, Concepts, Hypotheses

- Sky: Sunny, Cloudy, Rainy
- AirTemp: Warm, Cold
- Humidity: Normal, High
- Wind: Strong, Weak
- Water: Warm, Cold
- Forecast: Same, Change

#distinct instances : 3*2*2*2*2*2 = 96
#distinct concepts : 2^{96}
#syntactically distinct hypotheses : 5*4*4*4*4*4=5120
#semantically distinct hypotheses : 1+4*3*3*3*3*3=973

2.2 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.
- 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.]

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** (illustration [Mitchell, p. 25])

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

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

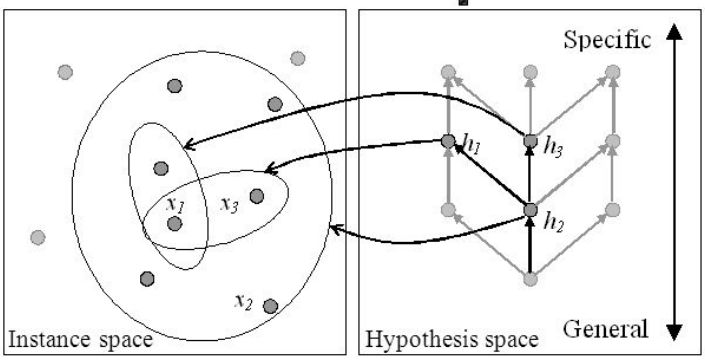
The Structure of *H*

x = instance
h = hypothesis
d = training example
c = target function

Many algorithms for concept learning organize the search by using a useful structure in the hypothesis space: **general-to-specific ordering**!

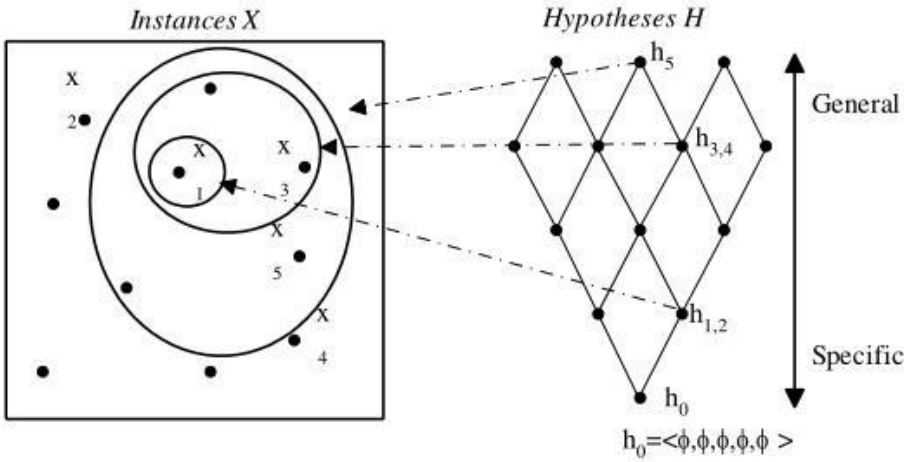
An example:

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



Because h_2 imposes fewer constraints, it will cover more instances x in X than both h_1 and h_3.

Hypothesis Space Search by Find-S

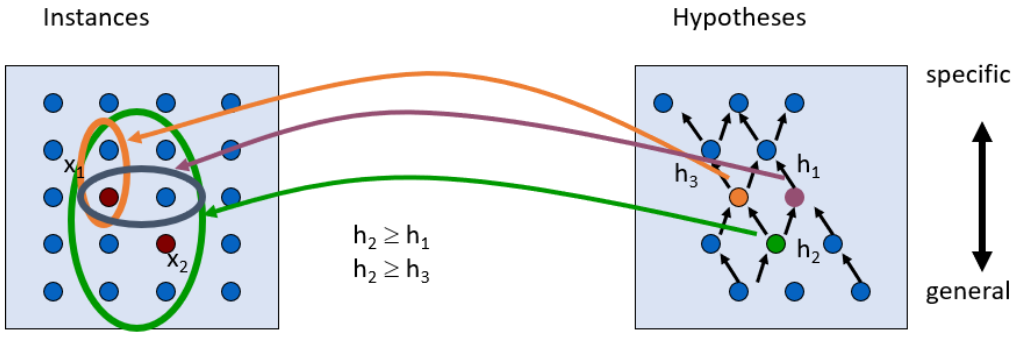


h_0=<ϕ,ϕ,ϕ,ϕ,ϕ >

x_1=<Round,Triangle,Round,Purple,Yes> +
x_2=<Square,Square,Square,Green,Yes> -
x_3=<Square,Triangle,Round,Yellow,Yes> +
x_4=<Round,Triangle,Round,Green,No> -
x_5=<Square,Square,Round,Yellow,Yes> +

h_1=<Round,Triangle,Round,Purple,Yes>
h_2=<Round,Triangle,Round,Purple,Yes>
h_3=<?,Triangle,Round,?,Yes>
h_4=<?,Triangle,Round,?,Yes>
h_5=<?,?,Round,?,Yes>

Instance, Hypotheses and ”more general”



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,?>

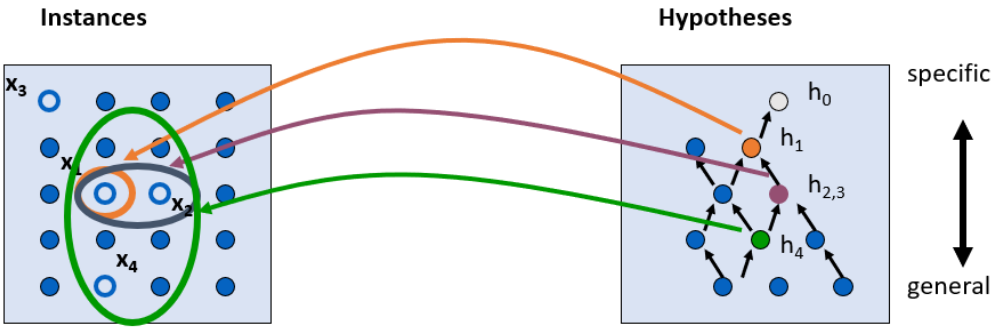
2.3 Find-S, a Maximally Specific Hypothesis Learning Algorithm

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 in h 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

Hypothesis Space Search by Find-S



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

x_2=<Sunny,Warm,High,Strong,Warm,Same> +

x_3=<Rainy,Cold,High,Strong,Warm,Change> -

x_4=<Sunny,Warm,High,Strong,Cool,Change> +

h_0 = < Ø, Ø, Ø, Ø, Ø, Ø,>

h_1=< Sunny,Warm,Normal,Strong,Warm,Same>

$h_{2,3}$=< Sunny,Warm,?,Strong,Warm,Same>

h_4=< Sunny,Warm,?, Strong,?,?>

Properties of Find-S

- Hypothesis space described by conjunctions of attributes
- Find-S will output the most specific hypothesis within H that is consistent with the positive training examples
- The output hypothesis will also be consistent with the negative examples, provided the target concept is contained in H.

Complaints about Find-S

- Can't tell if the learner has converged to the target concept, in the sense that it is unable to determine whether it has found the *only* hypothesis consistent with the training examples.
- Can't tell when training data is inconsistent, as it ignores negative training examples.
- Why prefer the most specific hypothesis?
- What if there are multiple maximally specific hypothesis?

2.4 Version Spaces

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

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

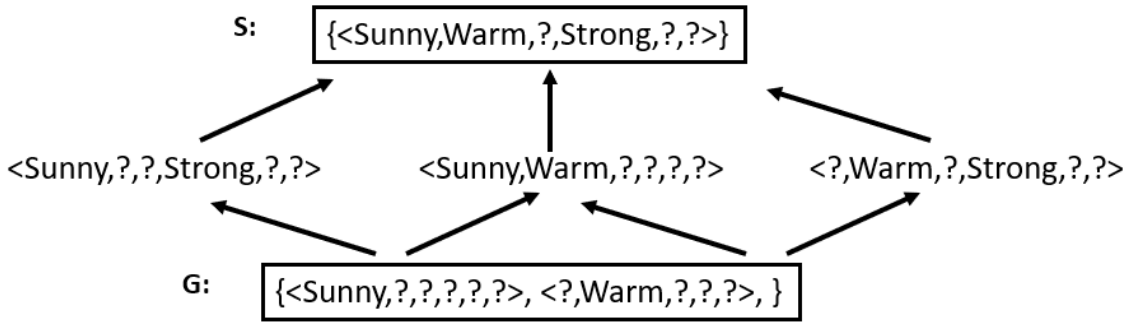
- **The version space, $VS_{H,D}$, with respect to hypothesis space H, and training set D, is the subset of hypotheses from H consistent with all training examples:**

$VS_{H,D}$ = {h ∈ H | Consistent(h,D) }

List-Then Eliminate Algorithm

1. *VersionSpace* ← a list containing every hypothesis in H
2. For each training example <x,c(x)>
remove from *VersionSpace* any hypothesis that is inconsistent with the training example h(x) ≠ c(x)
3. Output the list of hypotheses in *VersionSpace*

Example Version Space



x_1 = <Sunny Warm Normal Strong Warm Same> +
x_2 = <Sunny Warm High Strong Warm Same> +
x_3 = <Rainy Cold High Strong Warm Change> -
x_4 = <Sunny Warm High Strong Cool Change> +

Representing Version Spaces

- The general boundary, G, of version space $VS_{H,D}$ is the set of maximally general members.
- The specific boundary, S, of version space $VS_{H,D}$ is the set of maximally specific members.
- Every member of the version space lies between these boundaries

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

where x ≥ y means x is more general or equal than y

2.6 Candidate Elimination 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

  Go ← {(?, ?, ?, ?, ?, ?)}
- and initializing the S boundary set to contain the most specific (least general) hypothesis

  So ← {(ø, ø, ø, ø, ø, ø)}

Candidate Elimination Algorithm

G ← maximally general hypotheses in H

S ← maximally specific hypotheses in H

For each training example d=<x,c(x)>

Case 1 : **If d is a positive example**

Remove from G any hypothesis that is inconsistent with d

For each hypothesis s in S that is not consistent with d

- Remove s from S.
- Add to S all minimal generalizations h of s such that
  - h consistent with d
  - Some member of G is more general than h
- Remove from S any hypothesis that is more general than another hypothesis in S

Case 2: If d is a negative example

Remove from S any hypothesis that is inconsistent with d

For each hypothesis g in G that is not consistent with d

- remove g from G.
- Add to G all minimal specializations h of g such that
  - h consistent with d
  - Some member of S is more specific than h
- Remove from G any hypothesis that is less general than another hypothesis in G

An Illustrative Example

Figure traces the CANDIDATE-ELIMINATION algorithm applied to the first two training examples from table

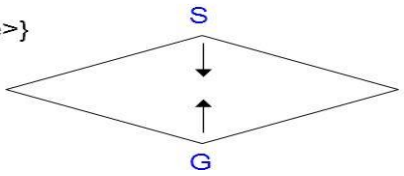
Candidate-Elimination Algorithm

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

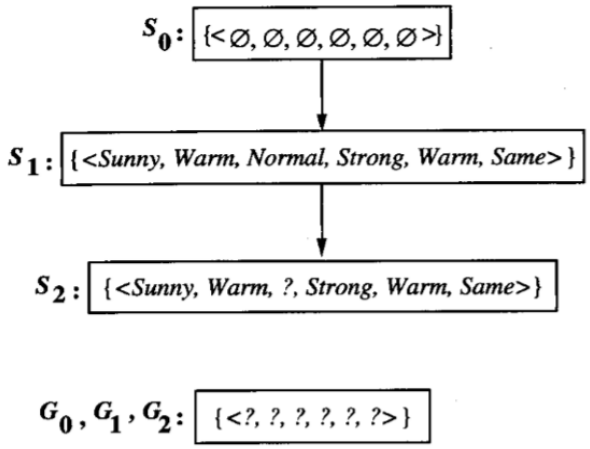
S_0 = {<∅, ∅, ∅, ∅, ∅, ∅>}
G_0 = {<?, ?, ?, ?, ?, ?>}

S_1 = {<Sunny, Warm, Normal, Strong, Warm, Same>}
G_1 = {<?, ?, ?, ?, ?, ?>}

S_2 = {<Sunny, Warm, ?, Strong, Warm, Same>}
G_2 = {<?, ?, ?, ?, ?, ?>}



Trace1

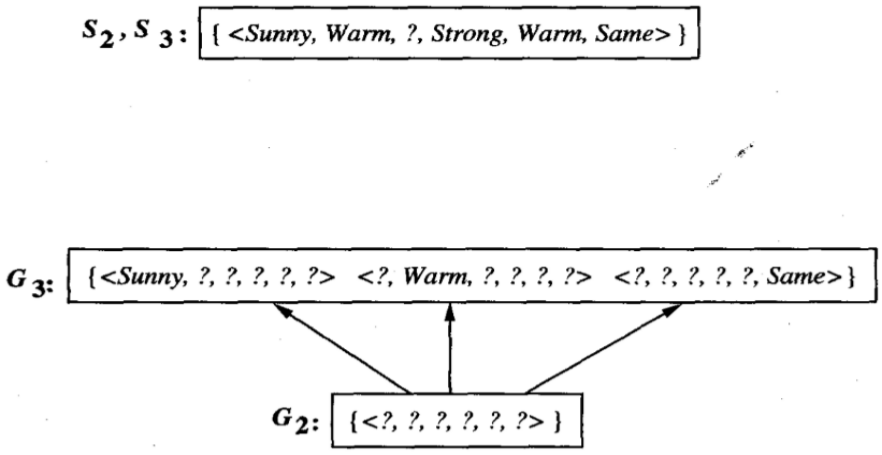


Training examples:

1. *<Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes*
2. *<Sunny, Warm, High, Strong, Warm, Same>, Enjoy Sport = Yes*

CANDIDATE-ELIMINATION Trace 1. S_0 and G_0 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, as in the FIND-S algorithm. They have no effect on the G boundary.

Trace 2:

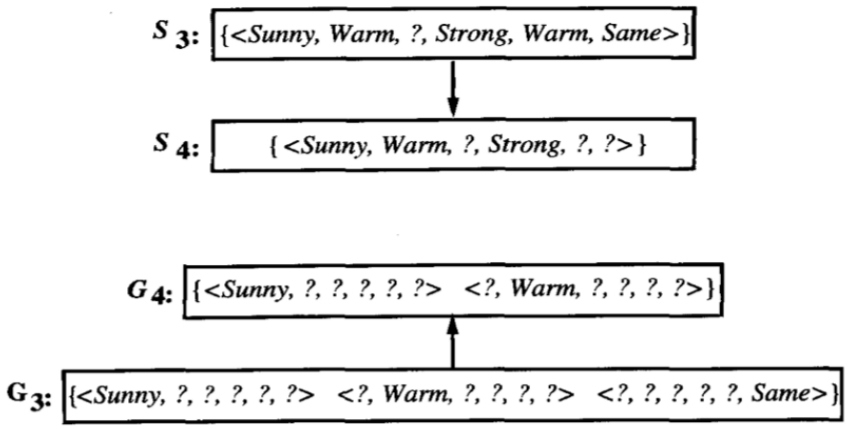


Training Example:

3. *<Rainy, Cold, High, Strong, Warm, Change>, EnjoySport=No*

CANDIDATE-ELIMINATION Trace 2. Training example 3 is a negative example that forces the G_2 boundary to be specialized to G_3. Note several alternative maximally general hypotheses are included in G_3.

Trace3

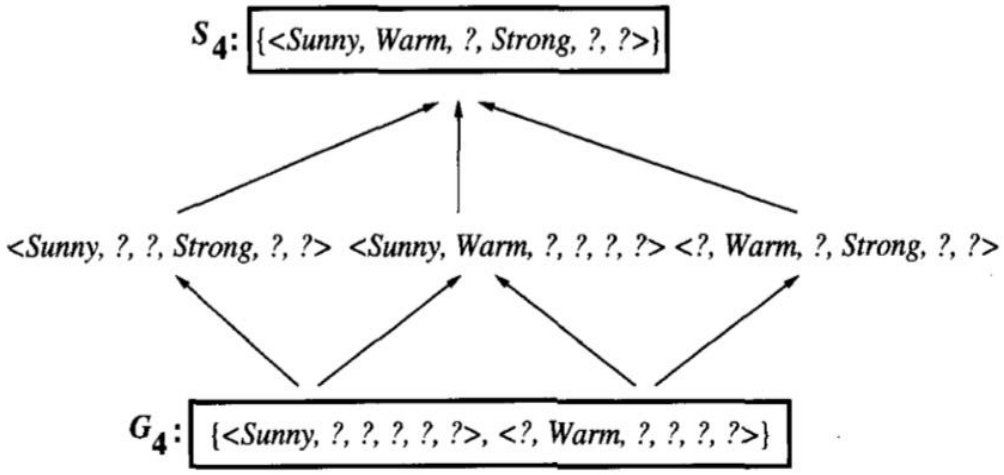


Training Example:

4.*<Sunny, Warm, High, Strong, Cool, Change>, EnjoySport = Yes*

CANDIDATE-ELIMINATION Trace 3. The positive training example generalizes the S boundary, from S_3 to S_4. One member of G_3 must also be deleted, because it is no longer more general than the S_4 boundary.

Final Version Space:



The final version space for the *EnjoySport* concept learning problem and training examples described earlier.

REMARKS ON VERSION SPACES AND CANDIDATE-ELIMINATION

1. **Will the CANDIDATE-ELIMINATION Algorithm Converge to the Correct Hypothesis?**
2. **What Training Example Should the Learner Request Next?**
3. **How Can Partially Learned Concepts Be Used?**

2.6 Inductive Bias

Inductive Bias I: A Biased Hypothesis Space

Database:

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

class

Given our previous choice of the hypothesis space representation, no hypothesis is consistent with the above database: we have *BIASED* the learner to consider only conjunctive hypotheses

Inductive Bias II: An Unbiased Learner

- **Idea: Choose H that expresses every teachable concept, that means H is the set of all possible subsets of X called the power set P(X)**
- **$|X|=96$, $|P(X)|=2^{96} \sim 10^{28}$ distinct concepts**
- **H = disjunctions, conjunctions, negations**
    - **e.g. <Sunny Warm Normal ? ? ?> v <? ? ? ? ? Change>**
- **H surely contains the target concept.**

Inductive Bias II: An Unbiased Learner

- **In order to solve the problem caused by the bias of the hypothesis space, we can remove this bias and allow the hypotheses to represent every possible subset of instances. The previous database could then be expressed as: <Sunny, ?,?,?,?,?> v <Cloudy,?,?,?,?,?,?>**
- ***However, such an unbiased learner is not able to generalize beyond the observed examples!!!!*** **All the non-observed examples will be well-classified by half the hypotheses of the version space and misclassified by the other half.**

Inductive Bias II: An Unbiased Learner
What are S and G in this case?

Assume positive examples (x_1, x_2, x_3) and negative examples (x_4, x_5)

S : { (x_1 v x_2 v x_3) }

G : { ¬ (x_4 v x_5) }

The only examples that are classified are the training examples themselves. In other words in order to learn the target concept one would have to present every single instance in X as a training example.
Each unobserved instance will be classified positive by precisely half the hypothesis in VS and negative by the other half.

Inductive Bias III: The Futility of Bias-Free Learning

- Fundamental Property of Inductive Learning A learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.
- We constantly have recourse to inductive biases *Example:* we all know that the sun will rise tomorrow. Although we cannot *deduce* that it will do so based on the fact that it rose today, yesterday, the day before, etc., we do take this leap of faith or use this inductive bias, naturally!

Inductive Bias

Consider:

- Concept learning algorithm L
- Instances X, target concept c
- Training examples D_c={<x,c(x)>}
- Let $L(x_i,D_c)$ denote the classification assigned to instance x_i by L after training on D_c.

Definition:

The inductive bias of L is any minimal set of assertions B such that for any target concept c and corresponding training data D_c

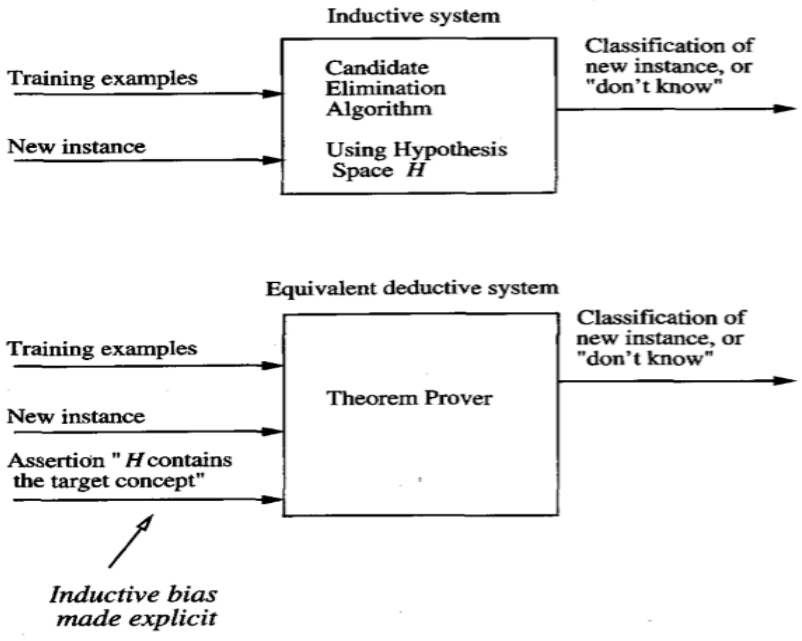
$$(\forall x_i \in X)[B \wedge D_c \wedge x_i] \mid\!\!- L(x_i, D_c)$$

Where A |-- B means that A logically entails B.

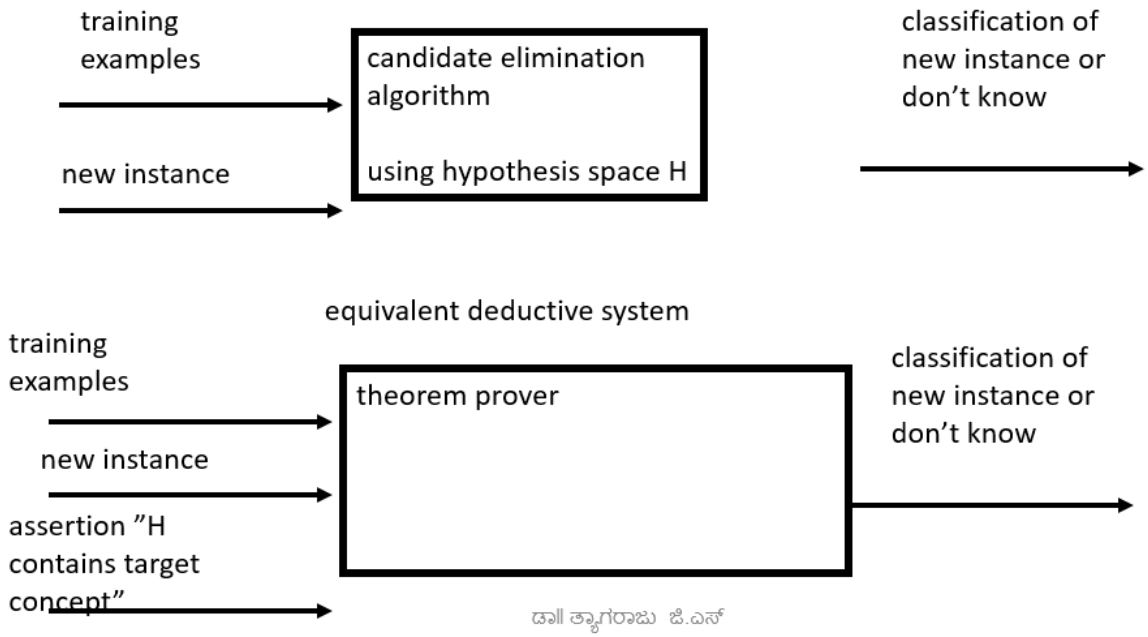
Inductive bias of CANDIDATE-ELIMINATION algorithm. The target concept c is contained in the given hypothesis space H.

Inductive Systems and Equivalent Deductive Systems

- Modeling inductive systems by equivalent deductive systems. The input-output behavior of the CANDIDATE-ELIMINATION Algorithm using a hypothesis space H is identical to that of a deductive theorem prover utilizing the assertion "*H contains the target concept*." This assertion is therefore called the inductive bias of the CANDIDATE-ELIMINATION Algorithm . Characterizing inductive systems by their inductive bias allows modeling them by their equivalent deductive systems. This provides a way to compare inductive systems according to their policies for generalizing beyond the observed training data.



Inductive Systems and Equivalent Deductive Systems



Ranking Inductive Learners according to their Biases

- **Rote-Learner: This system simply memorizes the training data and their classification--- No generalization is involved.**
- **Candidate-Elimination: New instances are classified only if all the hypotheses in the version space agree on the classification**
- **Find-S: New instances are classified using the most specific hypothesis consistent with the training data**

Module -1 Questions.

1. Define the following terms:

- **a. Learning**
- **b. LMS weight update rule**
- **c. Version Space**
- **d. Consistent Hypothesis**
- **e. General Boundary**
- **f. Specific Boundary**
- **g. Concept**

2. What are the important objectives of machine learning?

3. Explain find –S algorithm with given example. Give its application.

Table 1

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

4. What do you mean by a well –posed learning problem? Explain the important features that are required to well –define a learning problem.

5. Explain the inductive biased hypothesis space and unbiased learner

6. What are the basic design issues and approaches to machine learning?

7. How is Candidate Elimination algorithm different from Find-S Algorithm

8. How do you design a checkers learning problem

9. Explain the various stages involved in designing a learning system

10. Trace the Candidate Elimination Algorithm for the hypothesis space H' given the sequence of training examples from Table 1.

H'= < ?, Cold, High, ?,?,?>v<Sunny, ?, High, ?,?,Same>

11. Differentiate between Training data and Testing Data

12. Differentiate between Supervised, Unsupervised and Reinforcement Learning

13. What are the issues in Machine Learning

14. Explain the List Then Eliminate Algorithm with an example

15. What is the difference between Find-S and Candidate Elimination Algorithm

16. Explain the concept of Inductive Bias

17. With a neat diagram, explain how you can model inductive systems by equivalent deductive systems

18. What do you mean by Concept Learning?