Learning

Outlines

- What is Learning?
- Rote learning
- Induction
- Explanation based learning
- Discovery analogy

What is Learning?

- Most often heard criticisms of AI is that machines cannot be called intelligent until they are able to learn to do new things and adapt to new situations, rather than simply doing as they are told to do.
- **Definitions of Learning [Simon 1983]**: changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.
- Learning covers a wide range of phenomenon:
  - **Skill refinement** : Practice makes skills improve. More you play tennis, better you get
  - **Knowledge acquisition**: Knowledge is generally acquired through experience

Various Learning Mechanisms

- Simple storing of computed information or *rote learning*, is the most basic learning activity.

  -Many computer programs ie., database systems can be said to learn in this sense.

- Another way we learn if through taking advice from others.

  -Advice taking is similar to rote learning, but high-level advice may not be in a form simple enough for a program to use directly in problem solving. The advice need to be *operationalized*

- People also learn through their own problem-solving experience.

- **Learning from examples** : we often learn to classify things in the world without being given explicit rules.

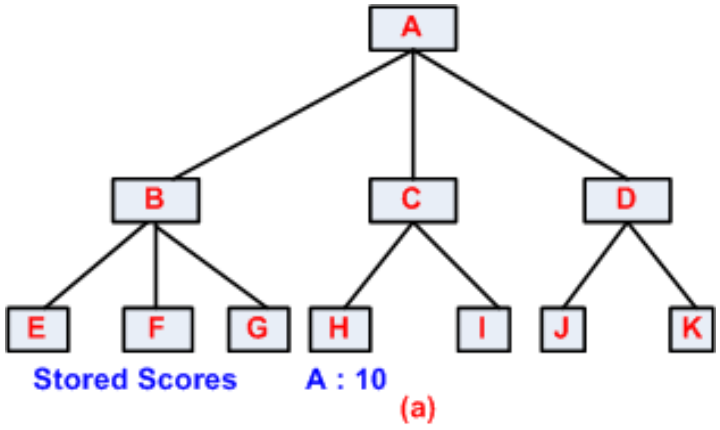
  -Learning from examples usually involves a teacher who helps us classify things by correcting us when we are wrong.

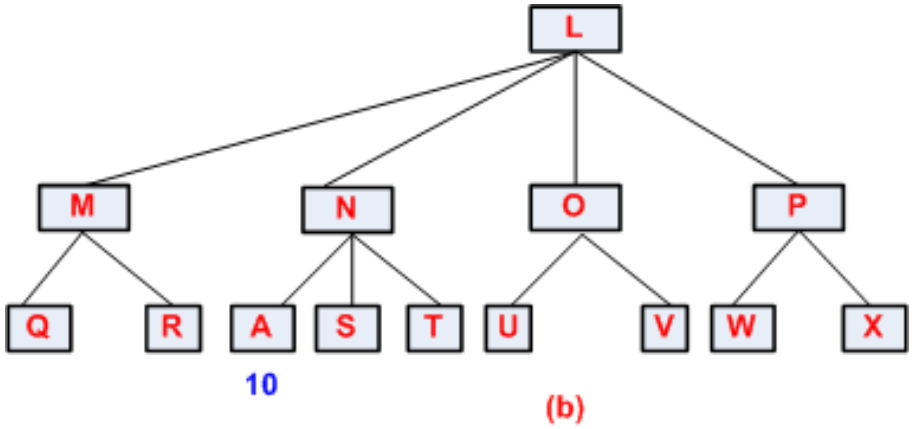
Rote Learning

- When a computer stores a piece of data, it is performing a rudimentary form of learning.
- **Pros**
  - In case of data caching, we store computed values so that we do not have to recompute them later.
  - When computation is more expensive than recall, this strategy can save a significant amount of time.
- Caching has been used in AI programs to produce some surprising performance improvements.
- Such caching is known as rote learning.

Rote Learning: Example

- Use static evaluation function & used that score to continue search
- When finished searching, Propagating the values backward, it had a score for the position represented by the root.
- It also recorded the board position at the root & backed up score that had just been computed for it.
- Instead of using the static evaluation function to compute a score for **A**, the stored values for **A** can be used.





Storing Backed-up Values

Need Capabilities

- Rote learning does not appear to involve any sophisticated problem-solving capabilities.

Needs 02 important capabilities

- *Organized storage of information*: in order for it to be faster to use a stored value than it would be to recompute it, there must be a way to access the appropriate stored value quickly
- *Generalization*: the number of distinct objects that might potentially be stored can be very large. To keep the number of stored objects down to a manageable level, some kind of generaliztion is necessary

Learning from examples: Induction

- ***Classification*** is the process of assigning, to a particular input, the name of a class to which it belongs.
- The classes from which the classification procedure can choose can be described in a variety of ways.
- Their definition will depend on the use to which they are put.
- *Classification is an important component of many problem solving tasks*.
- Simplest: straightforward recognition task

  If: the current goal is to get from place **A** to place **B**, and there is a **WALL** separating the two places

  then: look for a **DOORWAY** in the **WALL** and go through it

  - To use this rule successfully, the system's matching routine must be able to identify an object as a wall
  - Without this, the rule can never be invoked
  - Then, to apply the rule, the system must be able to recognize a doorway

Ways of classification

Before classification can be done, the classes it will use must be defined:

1. Isolate a set of features that are relevant to the task domain. Define each class by a weighted sum of values of these features. Each class is then defined by a scoring function

   $$c_1t_1 + c_2t_2 + c_3t_3 + \ldots\ldots\ldots$$

   Each **t** corresponds to a value of a relevant parameter, & each **c** represents the weight to be attached to the corresponding **t**

   Ex: task is weather prediction, the parameters can be measurements such as rainfall, location of cold fronts etc.

2. Isolate a set of features that are relevant to the task domain. Define each class as a structure composed of these features.

   Ex: classifying animals, various features can be such things as color, length of neck , feathers, etc.

Concept Learning

- The idea of producing a classification program that can evolve its own class definitions is called ***concept learning or induction***.
- The techniques used for this task must depend on the way that classes (concepts) are described
- If classes are described by scoring functions, the concept learning can be done using the technique of coefficient adjustment

Version spaces

- Mitchell [1977; 1978]
- Goal: to produce a description that is consistent with all positive examples but no negative examples in the training set.
- Version spaces work by maintaining a set of possible descriptions and evolving that set as new examples and near misses are presented.
- The version space is simply a set of descriptions, so an initial idea is to keep an explicit list of those descriptions.

Frame representing an individual car

Car023

Origin:	Japan
Manufacturer:	Honda
Color:	Blue
Decade:	1970
Type:	Economy

An example of the concept car

- Each slot contain only the discrete values
- Features & values: bias
- We can learn concepts that have to do with Car manufactures, but not car owners
- A clear statement of the bias of a learning system Is very important to its evaluation

Origin	**∈**	**{Japan, USA, Britain, Germany, Italy}**
Manufacturer	**∈**	**{Honda, Toyota, Ford, Chrysler, Jaguar, BMW, Fiat}**
Color	**∈**	**{Blue, Green, Red, White}**
Decade	**∈**	**{1950, 1960, 1970, 1980, 1990, 2000}**
Type	**∈**	**{Economy, Luxury, Sports}**

Representation Language for Cars

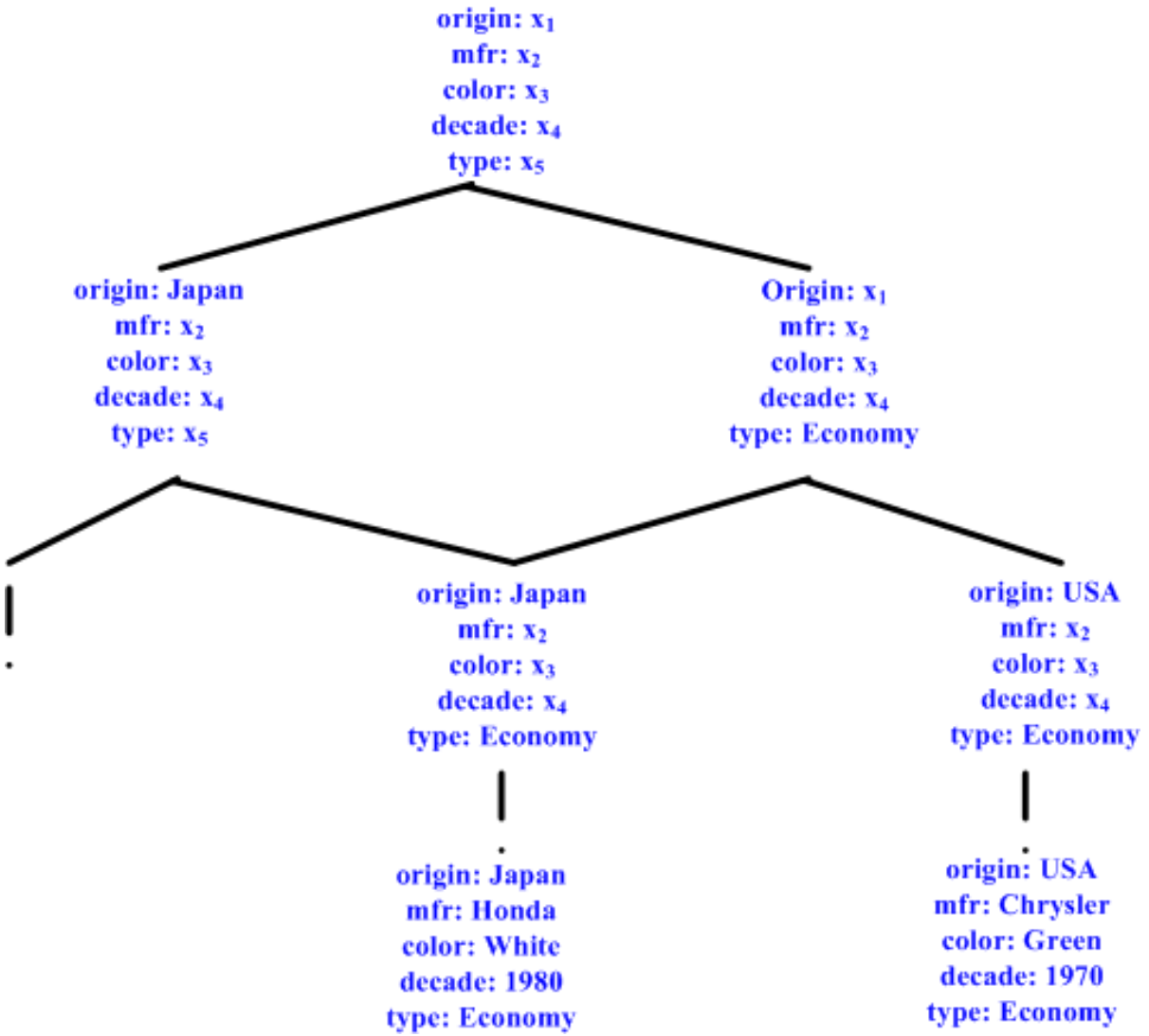
Concept Descriptions

- Concept descriptions/training examples, can be stated in terms of these slots and values
  - **Japanese economy car**
  - X_1, X_2, X_3 are variables
  - X_2: the color of a car is not relevant to whether the car is a Japanese economy car

Origin:	**Japan**
Manufacturer:	x_1
Color:	x_2
Decade:	x_3
Type:	Economy

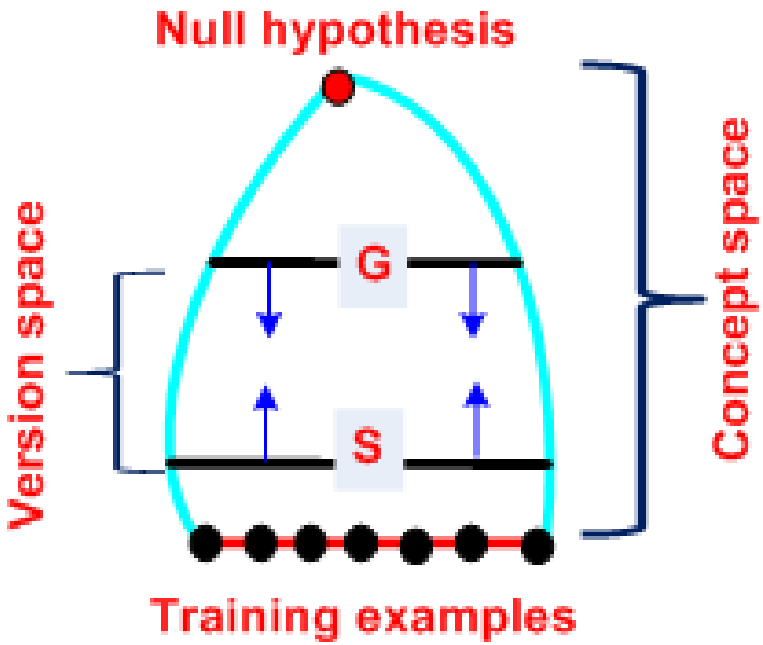
Concept "Japanese economy car"

Partial Ordering of Concepts Specified by The Representation Language



Concept & Version Spaces

- The entire partial ordering: **concept space**
- At the top of the concept space is the null description, consisting only of variables
- At bottom, are all possible training instances which contains no variables
- Before we receive any training examples, the target concept lies somewhere in the concept space

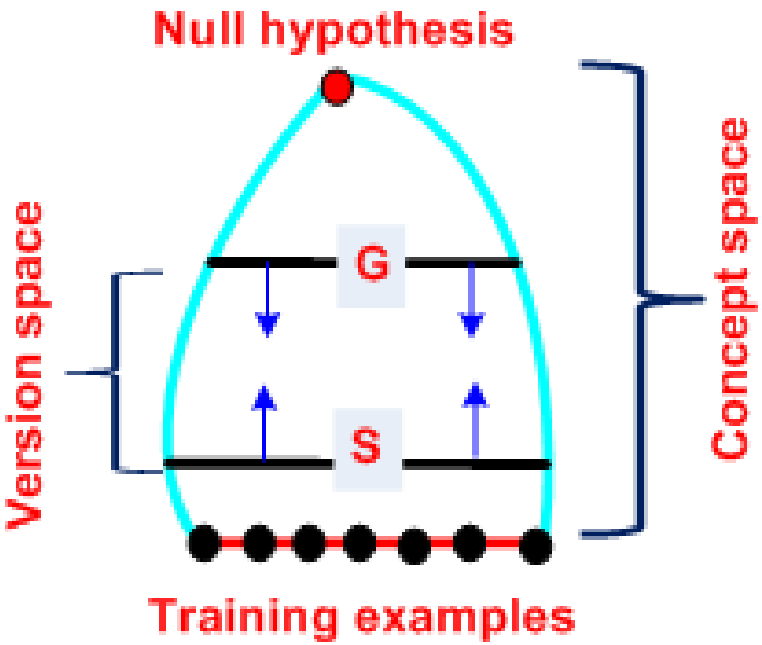


Concept & Version Spaces

Ex: if every possible description is an instance of the intended concept, then the null description is the concept definition since it matches everything

-On the other hand, if the target concept includes only a single example, then one of the descriptions at the bottom of the concept space is the desired concept definition.

•Most target concepts lies somewhere in between these two extremes



Concept & Version spaces

- As we process training examples, we want to refine our notion of where the target concept might lie
- Concept hypotheses can be represented as a subset of the concept space called *version space*
- The version space is the largest collection of descriptions that is consistent with all the training examples seen so far

How can we represent version space?

- The version space is simply a set of descriptions, so an initial idea is to keep an explicit list of those descriptions
- Unfortunately, the number of descriptions in the concept space is exponential in the number features & values
- So, enumerating them is prohibitive
- **Version space consists of 02 subsets of the concept space:**

1. One subset, G contains the most general descriptions consistent with the training examples
2. The other subset, S contains the most specific descriptions consistent with the training examples.

The version space is the set of all descriptions that lie between

some element of G & some element of S in the partial order of the

concept space

How can we represent version space?

- This representation of the version space is not only efficient for storage, but also for modification.
- Each time we receive a +ve training example, we want to make the S set more general
- -ve training examples serve to make the G set more specific
- If S & G set converge, our range of hypotheses will narrow to a single concept description
- The algorithm for narrowing the version space is called the *Candidate elimination algorithm*.

Algorithm: Candidate Elimination

Given: A representation language and a set of positive and negative examples expressed in that language.

Compute : A concept description that is consistent with all the positive examples and none of the negative examples.

1. Initialize G to contain one element
2. Initialize S to contain one element: the first positive element.
3. Accept new training example. If it is a positive example, first remove from G any descriptions that do not cover the example. Then update the set S to contain most specific set of descriptions in the version space that cover the example and the current elements of the S set. Inverse actions for negative example
4. If S and G are both singleton sets, then if they are identical, output their values and halt.

   -if they are both singleton sets but they are different, then the training cases were inconsistent. Output this result & halt. Otherwise go to step 3

Candidate Elimination

Origin:	**Japan**
Mfr:	Honda
Color:	Blue
Decade:	1980
Type:	Economy
(+)	

Origin:	**Japan**
Mfr:	Toyota
Color:	Green
Decade:	1970
Type:	sports
(-)	

Origin:	**Japan**
Mfr:	Toyota
Color:	Blue
Decade:	1990
Type:	Economy
(+)	

Origin:	**USA**
Mfr:	Chrysler
Color:	Red
Decade:	1980
Type:	Economy
(-)	

Origin:	**Japan**
Mfr:	Honda
Color:	White
Decade:	1980
Type:	Economy
(+)	

Positive & Negative examples of the concept “Japanese economy car”

Candidate Elimination

- G & S both start out as singleton sets
- G contains null description & S contains the first positive training example
- The version space now contains all descriptions that are consistent with first example:

$G = \{(x_1, x_2, x_3, x_4, x_5)\}$

S = {(Japan, Honda, Blue, 1980, Economy)}

- Process 2nd example (-)
- The G set must be specialized in such a way that the negative example is no longer in the version space
- Specialization involves replacing variables with constants
- The G set must be specialized only to descriptions that are within the current version space, not outside of it

Candidate Elimination

- Available specialization:

G = {(x_1,Honda, x_3, x_4, x_5), (x_1, x_2, Blue, x_4, x_5),
(x_1, x_2,x_3, 1980, x_5), (x_1, x_2, x_3, x_4,Economy)}

- The S set is unaffected by the negative example.
- Process 3rd example (+)
- The first order is to remove from G set any descriptions that are inconsistent with positive example:

G = {(x_1, x_2, Blue, x_4, x_5), (x_1, x_2, x_3, x_4,Economy)}

- We must generalize S set to include the new example. This involves replacing constants with variables:

S = {(Japan, x_2, Blue, x_4, Economy)}

Candidate Elimination

- At this point, the S & G sets specify a version space (a space of candidate descriptions) that can be translated roughly into English as: "the target concept may be specific as 'Japanese economy car' or as general as either 'blue car' or 'economy car'"
- Process 4th negative example (-)
- The G set will be specialized to avoid covering new example:

G = {(Japan, x_2, Blue, x_4, x_5), (Japan, x_2, x_3, x_4,Economy)}

- We know that the car must be Japanese, because all of the descriptions in the version space contain Japan as origin

Candidate Elimination

- Process 5^{th} example (+)
- First remove from G set any descriptions that are inconsistent with it:

G = {(Japan, x_2, x_3, x_4,Economy)}

- Generalize S set to include new example:

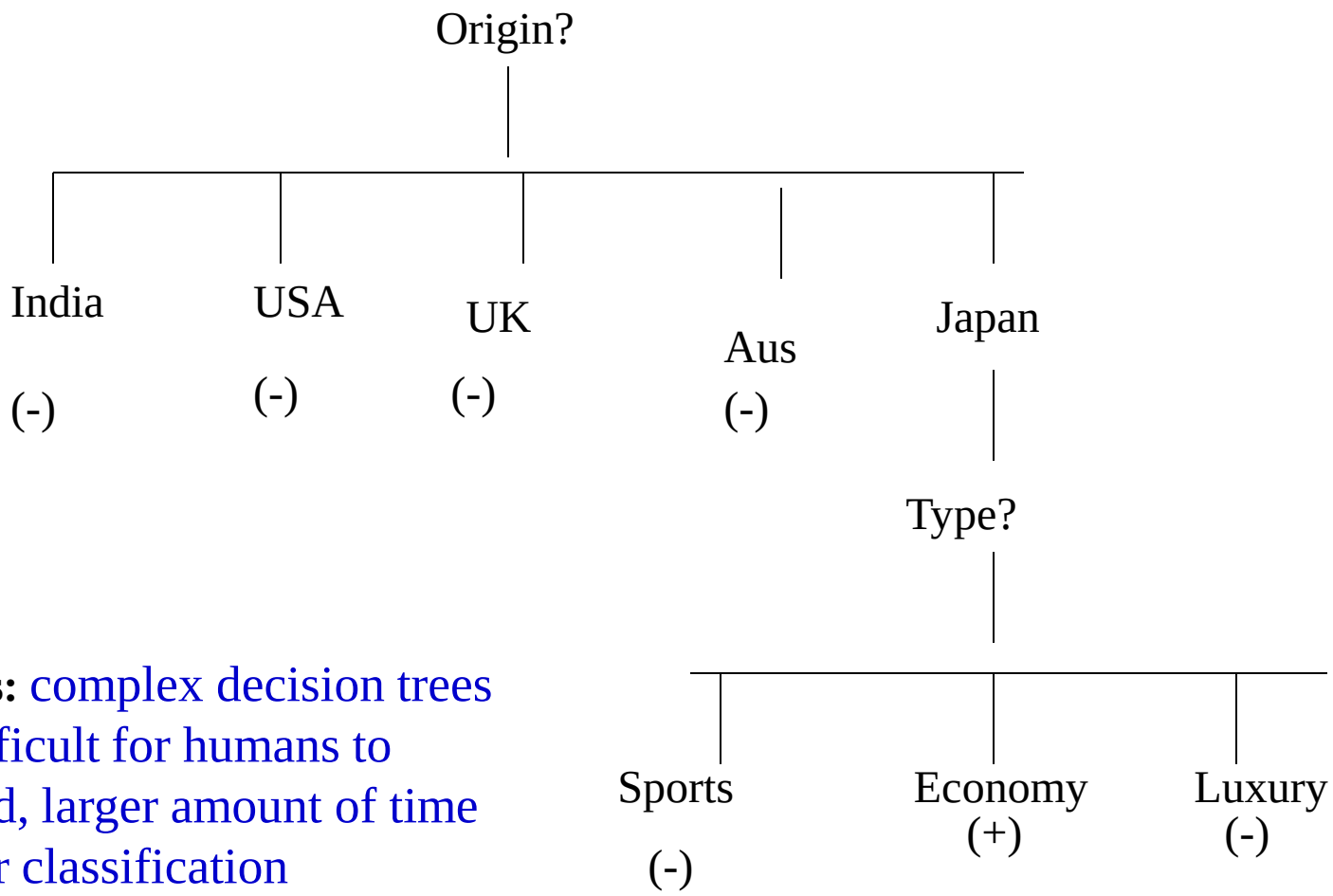
S = {(Japan, x_2, x_3, x_4,Economy)}

- S and G are both singletons, so the algorithm has converged on the target concept. No more examples are needed.

Decision Trees

- **Induction decision tree (ID)**
  - To classify a particular input, we start at the top of the tree and answer questions until we reach a leaf, where the specification is stored.
  - ID3 is a program example for Decision Trees.
  - ID3 uses iterative method to build up decision trees, preferring simple trees over complex ones, on the theory that simple trees are more accurate classifiers of future inputs.
  - It begins by choosing a random subset of the training examples.
  - This subset is called the window.
  - The algorithm builds a decision tree that correctly classifies all examples in the window.

Decision tree for “Japanese economy car”



Drawbacks: complex decision trees
Can be difficult for humans to
Understand, larger amount of time
Needed for classification

Study Yourself !!!!

- Analogy
  - Transformational Analogy
  - Derivational Analogy
- Winston's Learning Program

Best Of Luck

- Don’t Panic!!
- Study heard!!
- Follow the instructions/suggestions/comments during conducting this course