UNIT - I

# 1 Introduction

## 1.1 : Introduction : Well Posed Learning Problems

**Q.1 Define learning.** ☞ [JNTU : Dec.-17, Marks 2]

**Ans. :** Learning is a phenomenon and process which has manifestations of various aspects. Learning process includes gaining of new symbolic knowledge and development of cognitive skills through instruction and practice. It is also discovery of new facts and theories through observation and experiment.

**Q.2 Define machine learning.**

**Ans. :** 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.

**Q.3 What is need of machine learning in this era ?** ☞ [JNTU : Dec.-16, Marks 3]

**Ans. :** • Main goal of machine learning is to devise learning algorithms that do the learning automatically without human intervention or assistance.

- The machine learning paradigm can be viewed as "programming by example." Another goal is to develop computational models of human learning process and perform computer simulations.
- The goal of machine learning is to build computer systems that can adapt and learn from their experience.
- Machine learning algorithms can figure out how to perform important tasks by generalizing from examples.
- Machine Learning provides business insight and intelligence. Decision makers are provided with greater insights into their organizations. This adaptive technology is being used by global enterprises to gain a competitive edge.
- Machine learning algorithms discover the relationships between the variables of a system (input, output and hidden) from direct samples of the system.

**Q.4 What are T, P, E ? How do we formulate a machine learning problem ?**

**Ans. :** • In general, to have a well-defined learning problem, we must identity these three features : the class of tasks, the measure of performance to be improved, and the source of experience.

- A Robot Driving Learning Problem
  1. Task T : Driving on public, 4-lane highway using vision sensors.
  2. Performance measure P : Average distance traveled before an error (as judged by human overseer).
  3. Training experience E : A sequence of images and steering commands recorded while observing a human driver.
- A Handwriting Recognition Learning Problem
  1. Task T : Recognizing and classifying handwritten words within images
  2. Performance measure P : Percent of words correctly classified.
  3. Training experience E : A database of handwritten words with given classifications.
- Text Categorization Problem
  1. Task T : Assign a document to its content category.
  2. Performance measure P : Precision and Recall.
  3. Training experience E : Example pre-classified documents.

**Q.5 What are the reasons for using machine learning ?**

**Ans. :** Following are some of the reasons :

1. Some tasks cannot be defined well, except by examples. For example : recognizing people.
2. Relationships and correlations can be hidden within large amounts of data. To solve these problems, machine learning and data mining may be able to find these relationships.
3. Human designers often produce machines that do not work as well as desired in the environments in which they are used.
4. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans.
5. Environments change time to time.
6. New knowledge about tasks is constantly being discovered by humans.

**Q.6 List the phases of machine learning ?**

**Ans. :** Typically follows three phases :

1. **Training :** A training set of examples of correct behaviour is analysed and some representation of the newly learnt knowledge is stored. This is some form of rules.
2. **Validation :** The rules are checked and, if necessary, additional training is given. Sometimes additional test data are used, but instead, a human expert may validate the rules, or some other automatic knowledge - based component may be used. The role of the tester is often called the opponent.
3. **Application :** The rules are used in responding to some new situation.

**Q.7 What is meant by machine learning ? What is its need to today's society ? Explain successful applications of machine learning ?**

**☞ [JNTU : Dec.-17, Marks 10]**

**Ans. :** • Examples of successful applications of machine learning :

1. Learning to recognize spoken words.
2. Learning to drive an autonomous vehicle.
3. Learning to classify new astronomical structures.
4. Learning to play world-class backgammon.
5. Spoken language understanding: within the context of a limited domain, determine the meaning of something uttered by a speaker to the extent that it can be classified into one of a fixed set of categories

**Face Recognition**

- Face recognition task is effortlessly and every day we recognize our friends, relative and family members. We also recognition by looking at the photographs. In photographs, they are in different pose, hair styles, background light, makeup and without makeup.
- We do it subconsciously and cannot explain how we do it. Because we can't explain how we do it, we can't write an algorithm.
- Face has some structure. It is not a random collection of pixel. It is symmetric structure. It contains predefined components like nose, mouth, eye, ears. Every person face is a pattern composed of a particular combination of the features. By analyzing sample face images of a person, a learning program captures the pattern specific to that person and uses it to recognize if a new real face or new image belongs to this specific person or not.
- Machine learning algorithm creates an optimized model of the concept being learned based on data or past experience.

**Q.8 Explain the difference between machine learning and data mining.**

**Ans. :**

| Machine Learning | Data Mining |
|---|---|
| In machine learning the main goal is to learn a model, which can be used to predict future events. | In data mining, the main goal is to discover new interesting information which describes the current data set. |
| It considered data as secondary. | It considered data as primary. |
| Machine learning uses relatively complex and global models. | Data mining uses simple models or local patterns. |

| | |
|---|---|
| Only hundreds or thousands of examples in a training data set. | Huge data sets, even millions of rows. |
| To learn one or few carefully defined models, this can be used to predict future events. | To find all interesting patterns which describe the data set. |

**Q.9 What are the ingredients of machine learning ?**

**Ans. :** The ingredients of machine learning are as follows :

1. **Tasks :** The problems that can be solved with machine learning. A task is an abstract representation of a problem. The standard methodology in machine learning is to learn one task at a time. Large problems are broken into small, reasonably independent sub-problems that are learned separately and then recombined.
   - Predictive tasks perform inference on the current data in order to make predictions. Descriptive tasks characterize the general properties of the data in the database
2. **Models :** The output of machine learning. Different models are geometric models, probabilistic models, logical models, grouping and grading.
   - The model-based approach seeks to create a modified solution tailored to each new application. Instead of having to transform your problem to fit some standard algorithm, in model-based machine learning you design the algorithm precisely to fit your problem.
   - Model is just made up of set of assumptions, expressed in a precise mathematical form. These assumptions include the number and types of variables in the problem domain, which variables affect each other, and what the effect of changing one variable is on another variable.
   - Machine learning models are classified as : Geometric model, Probabilistic model and Logical model.
3. **Features :** The workhorses of machine learning. A good feature representation is central to achieving high performance in any machine learning task.
   - Feature extraction starts from an initial set of measured data and builds derived values intended to be informative, non redundant, facilitating the subsequent learning and generalization steps.
   - Feature selection is a process that chooses a subset of features from the original features so that the feature space is optimally reduced according to a certain criterion.



## 1.2 : Designing Learning System

**Q.10 What is an influence of information theory on machine learning ?** [JNTU : Dec.-17, Marks 3]

**Ans. :** Information theory is measures of entropy and information content. Minimum description length approaches to learning. Optimal codes and their relationship to optimal training sequences for encoding a hypothesis.

**Q.11 What is meant by target function of a learning program ?** [JNTU : Dec.-16, Marks 2]

**Ans. :** Target function is a method for solving a problem that an AI algorithm parses its training data to find. Once an algorithm finds its target function, that function can be used to predict results. The function can then be used to find output data related to inputs for real problems where, unlike training sets, outputs are not included.

**Q.12 Explain basic design issue and approaches of machine learning.**

**Ans. : Designing a Learning System**

- Goal : Design a system to learn how to play checkers and enter it into the world checkers tournament.
  1) Choose the training experience
  2) Choose the target function
  3) Choose a representation for the target function
  4) Choose a function approximation algorithm

**Training Experience**

- How training experience influences performance goal ?
  1. Type of feedback : Direct vs Indirect.
  2. Learning strategy : Have a teacher or not ? Exploration vs Exploitation ?

3. Diversity of training : Is the training data representative of the task ? How many peers should we play with? How many tactics should we try when playing with self ?

- Let us decide that our program will learn by playing with itself and formulate the learning problem.
- Choosing the training experience :
  1. Direct or indirect feedback
  2. Degree of learner's control
  3. Representative distribution of examples
- **Learning Goal is to :** Define precisely a class of problems that forms interesting forms of learning, explore algorithms to solve such problems, understand fundamental structure of learning problems and processes.
- Design choice 1 : The problem for selecting type of training experience from which our system will learn. Direct training examples. Just a bunch of board states together with a correct move.
- Design Choice 2 : Indirect training. A bunch of recorded games, where the correctness of the moves is inferred by the result of the game.
- Learning is most reliable when the training examples follow a distribution similar to that of future test examples.

**Q.13 How to choose and represent the target function ?**

**Ans. :** • It determines exactly what type of knowledge will be learned and how this will be used by the performance program.

- Choosing a representation for the target function :
  1. Expressive representation for a close function approximation
  2. Simple representation for simple training data and learning algorithms

$$V(b) = w_0 + w_1X_1 + ... + w_6X_6$$

$X_{1,2}$ : Number of black/red pieces on the board

$X_{3,4}$ : Number of black/red kings on the board

$X_{5,6}$ : Number of black/red pieces threatened can be captured on red/black next turn)

- Consider the chess board program.
- ChooseMove : B -> M where B is any legal board state and M is a legal move (hopefully the "best" legal move)
- Alternatively, function V : B -> $\Re$ which maps from B to some real value where higher scores are assigned to better board states.
- Now use the legal moves to generate every subsequent board state and use V to choose the best one and therefore the best legal move.
  1. V(b) = 100, if b is a final board state that is won
  2. V(b) = –100, if b is a final board state that is lost
  3. V(b) = 0, if b is a final board state that is a draw
  4. V(b) = V(b'), if b is not a final state where b' is the best final board state starting from b assuming both players play optimally
- While this recursive definition specifies a value of V(b) for every board state b, this definition is not usable by our checkers player because it is not efficiently computable.
- For representation
  1. Use a large table with an entry specifying a value for each distinct board state.
  2. Collection of rules that match against features of the board state.
  3. Quadratic polynomial function of predefined board features.

**Q.14 How to adjust the weights ?**

**Ans. :** • Choose the weights $w_i$ to best fit the set of training examples.

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

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

- Require an algorithm that will incrementally refine weights as new training examples become available and it will be robust to errors in these estimated training values.
- Least Mean Squares (LMS) is one such algorithm.

## 1.3 : Perspectives and Issues in Machine Learning

**Q.15 Define useful perspective on machine learning.**

**Ans. :** 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

**Q.16 Describe the issues in machine learning ?**

**Ans. :** Issues of machine learning are as follows :

- What learning algorithms to be used ?
- How much training data is sufficient ?
- When and how prior knowledge can guide the learning process ?
- What is the best strategy for choosing a next training experience ?
- What is the best way to reduce the learning task to one or more function approximation problems ?
- How can the learner automatically alter its representation to improve its learning ability ?

## 1.4 : Concept Learning and the General to Specific Ordering

**Q.17 What is hypothesis ?**

**Ans. :** • A hypothesis is a vector of constraints for each attribute

1. Indicate by a "?" that any value is acceptable for this attribute
2. Specify a single required value for the attribute
3. Indication by a "Ø" that no value is acceptable

- If some instance x satisfies all the constraints of hypothesis h, then h classifies x as a positive example (h(x) = 1).

**Q.18 What is the inductive learning hypothesis ?**

**Ans. :** 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.

**Q.19 Explain with example concept learning task.**

**Ans. :** • 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.
- Formal Definition for Concept Learning: Inferring a boolean-valued function from training examples of its input and output.
- 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.
- Concept learning involves determining a mapping from a set of input variables to a Boolean value. Such methods are known as inductive learning methods.
- If a function can be found which maps training data to correct classifications, then it will also work well for unseen data. This process is known as generalization.
- Example : Learn the "days on which my friend enjoys his favorite water sport"

| Example | Sky | Air Temp | Humidity | Wind | Water | Forecast | Enjoy Sport |
|---|---|---|---|---|---|---|---|
| 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 |

- A set of example days, and each is described by six attributes. The task is to learn to predict the value of EnjoySport for arbitrary day, based on the values of its attribute values.
- **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.
- 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.
- Hypothesis representation (constraints on instance attributes) :
  **<Sky, AirTemp, Humidity, Wind, Water, Forecast>**

1. Any value is acceptable is represented by ?
2. No value is acceptable is represented by Φ

**Q.20 Illustrate general to specific ordering of hypotheses in concept learning ?** ☞ **[JNTU : Dec.-17, Marks 5]**

**Ans. :**
- Many algorithms for concept learning organize the search through the hypothesis space by relying on a general-to-specific ordering of hypotheses.
- By taking advantage of this naturally occurring structure over the hypothesis space, we can design learning algorithms that 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.
- One learning method is to determine the most specific hypothesis that matches all the training data.
- 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.

$h_j \geq h_k$ iff $\forall x \in X : h_k(X) = 1 \Rightarrow h_j(x) = 1$

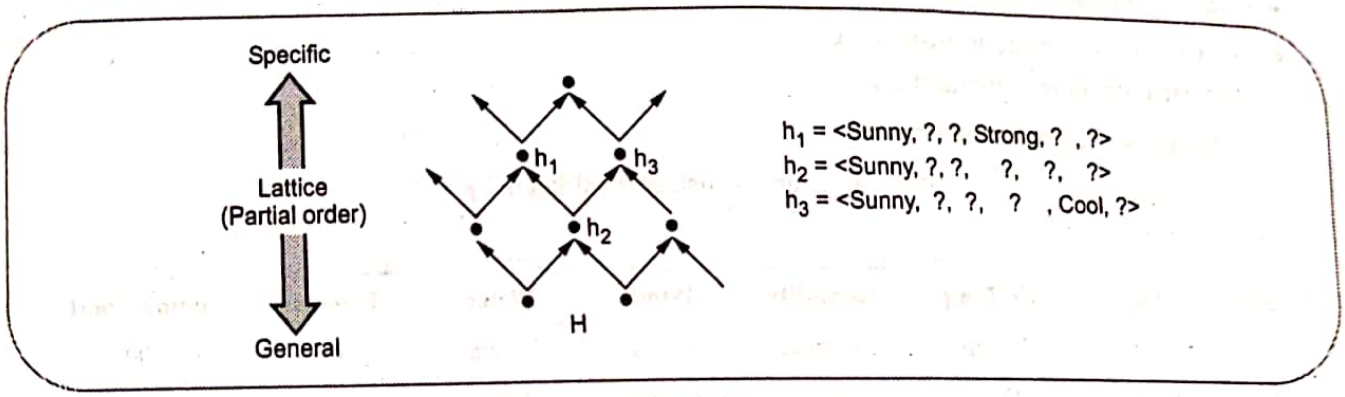



Fig. Q.20.1

## 1.5 : FIND-S Algorithm

**Q.21 Explain the key properties of FIND-S algorithm for concept learning with necessary example.**

☞ [JNTU : Dec.-17, Marks 5]

**Ans. :** • The key property is that for hypothesis spaces described by conjunctions of attribute constraints, FIND-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples.

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

| Example | Sky | Air Temp | Humidity | Wind | Water | Forecast | Enjoy Sport |
|---|---|---|---|---|---|---|---|
| 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 |

h = < ϕ, ϕ, ϕ, ϕ, ϕ, ϕ>

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

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

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

**Algorithm :**

- Initialize h to the most specific hypothesis in H :
- For each attribute training instance x :

  For each attribute constraint a in h :

  If the constraint is not satisfied by x.

  Then replace a, by the next more general constraint satisfied by x.
- Output hypothesis h

| Example | Sky | Air Temp | Humidity | Wind | Water | Forecast | Enjoy Sport |
|---|---|---|---|---|---|---|---|
| 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 |

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

**Prediction :**

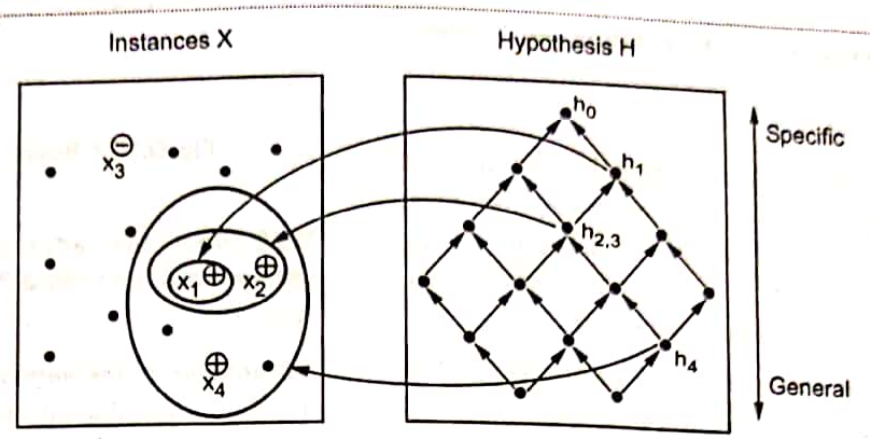
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | Rainy | Cold | High | Strong | Warm | Change | NO |
| 6 | Sunny | Warm | Normal | Strong | Warm | Same | YES |
| 7 | Sunny | Warm | Low | Strong | Cool | Same | YES |

**Q.22 Describe hypothesis space search by FIND-S algorithm.** ☞ [JNTU : Dec.-16, Marks 10]

**Ans. :** • The FIND-S algorithm illustrates one way in which the more-general than partial ordering can be used to organize the search for an acceptable hypothesis.

- The search moves from hypothesis to hypothesis, searching from the most specific to progressively more general hypotheses along one chain of the partial ordering.
- The hypothesis space search performed by FIND-S.

- The search begins ($h_o$) with the most specific hypothesis in H, then considers increasingly general hypotheses ($h_1$ through $h_4$) as mandated by the training examples.
- In the instance space diagram, positive training examples are denoted by "+," negative by "-," and instances that have not been presented as training examples are denoted by a solid circle.
- At each stage the hypothesis is the most specific hypothesis consistent with the training examples observed up to this point.




$x_1$ = <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>, +

$h_0$ = <∅, ∅, ∅, ∅, ∅, ∅>
$h_1$ = <Sunny warm normal strong warm same>
$h_2$ = <Sunny warm ? strong warm same>
$h_3$ = <Sunny warm ? strong warm same>
$h_4$ = <Sunny warm ? strong ? ?>

**Fig. Q.22.1**

## 1.6 : Version Space and Candidate Elimination Algorithm

**Q.23 What is version space ?**

**Ans. :**
- **Version space :** A set of all hypotheses that are consistent with the training examples.
- The **version space**, denoted VS_H,D, with respect to hypothesis space H and training examples D, is the subset of from H consistent with the training examples in D.
- 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.
- The version space method is a concept learning process accomplished by managing multiple models within a version space.

**Q.24 Explain characteristics of version space.**

**Ans. :** **Characteristics :**

1. Tentative heuristics are represented using version spaces.
2. A version space represents all the alternative plausible descriptions of a heuristic.
3. A plausible description is one that is applicable to all known positive examples and no known negative example.

4. A version space description consists of two complementary trees :
   i. One that contains nodes connected to overly general models, and
   ii. One that contains nodes connected to overly specific models.
5. Node values/attributes are discrete.

**Q.25 Describe compact representation for version spaces.**

**Ans. :**

- Instead of enumerating all the hypotheses consistent with a training set, we can represent its most specific and most general boundaries. The hypotheses included in-between these two boundaries can be generated as needed.
- Definition: 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.
- Definition : The specific boundary (S) with respect to hypothesis space H and training data D, is the set of minimally general (i.e., maximally specific) members of H consistent with D.
- Fig. Q.25.1 shows general and specific hypothesis.

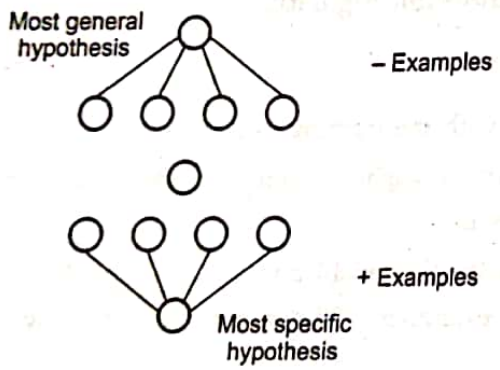



**Fig. Q.25.1 General and specific hypothesis**

- Each specialization must be a generalization of some specific concept description. No specialization can be a specialization of another general concept description. Fig. Q.25.2 shows boundary set with hypothesis

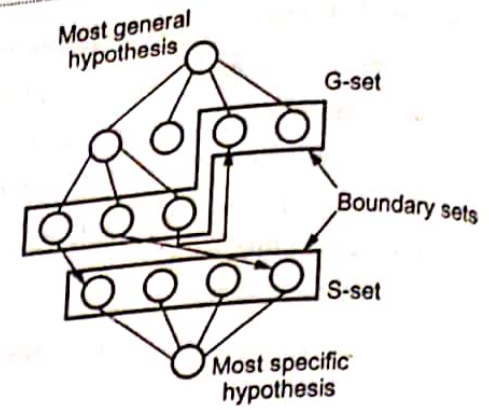



**Fig Q.25.2 Boundary set with hypothesis**

**Q.26 What are advantages and disadvantages of version space method ?**

**Ans. :**

**Advantages of the version space method:**

1. Can describe all the possible hypotheses in the language consistent with the data.
2. Fast (close to linear).

**Disadvantages of the version space method:**

1. Inconsistent data (noise) may cause the target concept to be pruned.
2. Learning disjunctive concepts is challenging

**Q.27 Describe List-Then-Eliminate Algorithm.**

**Ans. :** • 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.
- 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 spa?ces

**Q.28 Define candidate elimination algorithm.**

**Ans. :** The candidate-Elimination algorithm computes the version space containing all (and only those) hypotheses from H that are consistent with an observed sequence of training examples

**Q.29 Write algorithm for Candidate-Elimination.**

**Ans. :**

**Algorithm :**

**Given :**

- A representation language.
- 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.

**Method :**

- Initialize G, the set of maximally general hypotheses, to contain one element : the null description (all features are variables).
- Initialize S, the set of maximally specific hypotheses, to contain one element : the first positive example.
- Accept a new training example.
- If the example is positive:

1. Generalize all the specific models to match the positive example, but ensure the following :
   - The new specific models involve minimal changes.
   - Each new specific model is a specialization of some general model.
   - No new specific model is a generalization of some other specific model.
2. Prune away all the general models that fail to match the positive example.
   - If the example is negative :

1. Specialize all general models to prevent match with the negative example, but ensure the following :
   - The new general models involve minimal changes.
   - Each new general model is a generalization of some specific model.
   - No new general model is a specialization of some other general model.
2. Prune away all the specific models that match the negative example.
   - If S and G are both singleton sets, then :
   - if they are identical, output their value and halt.
   - if they are different, the training cases were inconsistent. Output this result and halt.
   - else continue accepting new training examples.

**The algorithm stops when :**

1. It runs out of data.
2. The number of hypotheses remaining is :

0 - No consistent description for the data in the language.

1 - Answer (version space converges).

$2^+$ - All descriptions in the language are implicitly included.

**Q.30 Explain candidate elimination algorithm with example.**

**Ans. :** • The candidate-Elimination algorithm computes the version space containing all (and only those) hypotheses from H that are consistent with an observed sequence of training examples.

- Example : Learning the concept of "Japanese Economy Car"
- Features : Country of Origin, Manufacturer, Color, Decade, Type

| Origin | Manufacturer | Color | Decade | Type | Example Type |
|---|---|---|---|---|---|
| Japan | Honda | Blue | 1980 | Economy | Positive |
| Japan | Toyota | Green | 1970 | Sports | Negative |
| Japan | Toyota | Blue | 1990 | Economy | Positive |
| USA | Chrysler | Red | 1980 | Economy | Negative |
| Japan | Honda | White | 1980 | Economy | Positive |
| Japan | Toyota | Green | 1980 | Economy | Positive |
| Japan | Honda | Red | 1990 | Economy | Negative |

- **Positive Example 1 :** (Japan, Honda, Blue, 1980, Economy)
- Initialize G to a singleton set that includes everything.

$$G = \{ (?, ?, ?, ?, ?) \}$$

- Initialize S to a singleton set that includes the first positive example.

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

- **Negative Example 2 :** (Japan, Toyota, Green, 1970, Sports)
- Specialize G to exclude the negative example.

$$G = \{ (?, Honda, ?, ?, ?), (?, ?, Blue, ?, ?), (?, ?, ?, 1980, ?), (?, ?, ?, ?, Economy) \}$$

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

- **Positive Example 3** : (Japan, Toyota, Blue, 1990, Economy)
- Prune G to exclude descriptions inconsistent with the positive example.

$$G = \{ (?, ?, Blue, ?, ?), (?, ?, ?, ?, Economy) \}$$

- Generalize S to include the positive example :

$$S = \{ (Japan, ?, Blue, ?, Economy) \}$$

- **Negative Example :** (USA, Chrysler, Red, 1980, Economy)
- Specialize G to exclude the negative example (but stay consistent with S)

$$G = \{ (?,?, Blue, ?, ?), (Japan, ?, ?, ?, Economy) \}$$

$$S = \{ (Japan, ?, Blue, ?, Economy) \}$$

**Negative Example :** (Japan, Honda, Red, 1990, Economy)

- Example is inconsistent with the version-space.
  **G cannot be specialized.**
  **S cannot be generalized.**
- The version space **collapses.**
- Conclusion : No conjunctive hypothesis is consistent with the data set

## 1.7 : Inductive Bias

**Q.31 What is an inductive bias ?**

**Ans. :**
- Consider a concept learning algorithm L for the set of instances X. Let c be an arbitrary concept defined over X, and let Dc = {<x , c(x)>} be an arbitrary set of training examples of c.
- Let L(xi, Dc) denote the classification assigned to the instance xi by L after training on the data Dc.
- The inductive bias of L is any minimal set of assertions B such that for any target concept c and corresponding training examples Dc the following formula holds.

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

**Q.32 Explain fundamental property of Inductive Inference.**

**Ans. :**
- 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 Leap : A learner should be able to generalize training data using prior assumptions in order to classify unseen instances.
- The generalization is known as inductive leap and our prior assumptions are the inductive bias of the learner.
- Inductive Bias (prior assumptions) of Candidate-Elimination Algorithm is that the target concept can be represented by a conjunction of attribute values, the target concept is contained in the hypothesis space and training examples are correct.

**Q.33 Explain with example inductive bias.**

**Ans. :**
- 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.
- 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 ?
- In EnjoySport example, we restricted the hypothesis space to include only conjunctions of attribute values. Because of this restriction, the hypothesis space is unable to represent even simple disjunctive target concepts such as "Sky = Sunny or Sky = Cloudy."

| Example | Sky | Air Temp | Humidity | Wind | Water | Forecast | Enjoy Sport |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Cool | Change | YES |
| 2 | Cloudy | Warm | Normal | Strong | Cool | Change | YES |
| 3 | Rainy | Warm | Normal | Strong | Cool | Change | NO |

- From first two examples : S2 : <?, Warm, Normal, Strong, Cool, Change>
- This is inconsistent with third examples, and there are no hypotheses consistent with these three examples PROBLEM : We have biased the learner to consider only conjunctive hypotheses. We require a more expressive hypothesis space.
- 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.

**Q.34 Which are the three learning algorithms from weakest to strongest bias ?**

**Ans. :**
- 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. **Inductive Bias : No inductive bias**
- CANDIDATE-ELIMINATION : 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. ***Inductive Bias : the target concept can be represented in its hypothesis space.***
- FIND-S : This algorithm, described earlier, finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances. ***Inductive Bias: the target concept can be represented in its hypothesis space, and all instances are negative instances unless the opposite is entailed by its other knowledge.***

## 1.8 : Decision Tree Learning : Introduction and Representation

**Q.35 What is decision tree ?**

**Ans. :**
- Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.
- A decision tree is a tree where each node represents a feature(attribute), each link(branch) represents a decision(rule) and each leaf represents an outcome(categorical or continues value).
- A decision tree or a classification tree is a tree in which each internal node is labeled with an input feature. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature.

**Q.36 What are the nodes of decision tree ?**

**Ans. :**
- A decision tree has two kinds of nodes

1. Each leaf node has a class label, determined by majority vote of training examples reaching that leaf.
2. Each internal node is a question on features. It branches out according to the answers.

- Decision tree learning is a method for approximating discrete-valued target functions. The learned function is represented by a decision tree

**Q.37 How to represents decision tree ?**

**Ans. :** • Goal : Build a decision tree for classifying examples as positive or negative instances of a concept

- Supervised learning, batch processing of training examples, using a preference bias.
- A decision tree is a tree where
  a. Each non-leaf node has associated with it an attribute (feature)
  b. Each leaf node has associated with it a classification (+ or -)
  c. Each arc has associated with it one of the possible values of the attribute at the node from which the arc is directed.
- Internal node denotes a test on an attribute. Branch represents an outcome of the test. Leaf nodes represent class labels or class distribution.
- A decision tree is a flow-chart-like tree structure, where each node denotes a test on an attribute value, each branch represents an outcome of the test, and tree leaves represent classes or class distributions. Decision trees can easily be converted to classification rules.

**Q.38 Write decision tree algorithm.**

**Ans. :** • To generate decision tree from the training tuples of data partition D.

**Input :**

1. Data partition ( D)
2. Attribute list
3. Attribute selection method

**Algorithm :**

1. Create a node (N)
2. If tuples in D are all of the same class then
3. Return node (N )as a leaf node labeled with the class C.
4. If attribute list is empty then return N as a leaf node labeled with the majority class in D
5. Apply attribute selection method(D, attribute list) to find the "best" splitting criterion;
6. Label node N with splitting criterion;
7. If splitting attribute is discrete-valued and multiway splits allowed
8. Then attribute list -> attribute list -> splitting attribute
9. For (each outcome j of splitting criterion )
10. Let $D_j$ be the set of data tuples in D satisfying outcome j;
11. If $D_j$ is empty then attach a leaf labeled with the majority class in D to node N;
12. Else attach the node returned by Generate decision tree ($D_j$ , attribute list) to node N;
13. End of for loop
14. return N;

**Q.39 Describe characteristics for appropriate problems for decision tree learning.**

**Ans. :** • Decision tree learning is generally best suited to problems with the following characteristics :

1. Instances are represented by attribute-value pairs. Fixed set of attributes, and the attributes take a small number of disjoint possible values.
2. The target function has discrete output values. Decision tree learning is appropriate for a boolean classification, but it easily extends to learning functions with more than two possible output values.
3. Disjunctive descriptions may be required. Decision trees naturally represent disjunctive expressions.
4. The training data may contain errors. Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
5. The training data may contain missing attribute values. Decision tree methods can be used even when some training examples have unknown values.
6. Decision tree learning has been applied to problems such as learning to classify

**Q.40 Define the following :**
**a. Input variable b. Leaf node c. Internal nodes d. Depth**

**Ans. :**

a. **Input variable :** Each member of the set { $x_1$, $x_2$, ... $x_n$} is called an input variable.

b. **Leaf node :** A node without further branches is called a leaf node. The leaf nodes return class

labels and, in some implementations, they return the probability scores.

c. **Internal nodes** are the decision or test points. Each internal node refers to an input variable or an attribute. The top internal node is called the root.

d. **Depth** : The depth of a node is the minimum number of steps required to reach the node from the root.

**Q.41 List the advantages and disadvantages of decision tree.**

**Ans. : Advantages :**

1. Rules are simple and easy to understand.
2. Decision trees can handle both nominal and numerical attributes.
3. Decision trees are capable of handling datasets that may have errors.
4. Decision trees are capable of handling datasets that may have missing values.
5. Decision trees are considered to be a nonparametric method.
6. Decision trees are self-explanatory.

**Disadvantages :**

1. Most of the algorithms require that the target attribute will have only discrete values.
2. Some problem are difficult to solve like XOR.
3. Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
4. Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.

**Q.42 How to evaluate a decision tree ?**

**Ans. :** • A decision tree can help you make tough choices between different paths and outcomes, but only if you evaluate the model correctly.

• Decision trees are graphic models of possible decisions and all related possible outcomes in a tree form, with the outcomes shown as "branches" off each choice. You can use a decision tree to help you make all kinds of business decisions, including new product development, new marketing strategies and workforce changes.

• Decision tree is evaluated as follows :

1. First, evaluate whether the splits of the tree make sense. Conduct sanity checks by validating the decision rules with domain experts, and determine if the decision rules are sound.

• Next, look at the depth and nodes of the tree. Having too many layers and obtaining nodes with few members might be signs of overfitting.

• In overfitting, the model fits the training set well, but it performs poorly on the new samples in the testing set.

• Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship.

• Overfitting is when a classifier fits the training data too tightly. Such a classifier works well on the training data but not on independent test data. It is a general problem that plagues all machine learning methods.

• Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

• To prevent over-fitting we have several options :

1. Restrict the number of adjustable parameters the network has - e.g. by reducing the number of hidden units, or by forcing connections to share the same weight values.
2. Stop the training early, before it has had time to learn the training data too well.
3. Add some form of regularization term to the error/cost function to encourage smoother network mappings.
4. Add noise to the training patterns to smear out the data points.

## 1.9 : Basic Decision Tree Learning Algorithm

**Q.43 Define information gain.**

**Ans. :** • Entropy measures the impurity of a collection. Information Gain is defined in terms of Entropy.

• Information gain tells us how important a given attribute of the feature vectors is.

- Information gain of attribute A is the reduction in entropy caused by partitioning the set of examples S.

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

where Values (A) is the set of all possible values for attribute A and $S_v$ is the subset of S for which attribute A has value v

**Q.44 What is the role of information gain in decision tree learning ?**

☞ [JNTU : Dec.-16, Marks-3]

**Ans. :** • Information gain measures how well a given attribute separates the training examples according to their target classification

- ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.
- Entropy is used for measuring information gain.
- Putting together a decision tree is all a matter of choosing which attribute to test at each node in the tree.
- We shall define a measure called information gain which will be used to decide which attribute to test at each node.
- Information gain is itself calculated using a measure called entropy, which we first define for the case of a binary decision problem and then define for the general case

**Q.45 Explain Gini Index and Entropy of dicision tree algorithm.**

**Ans. :** • One of the decision tree algorithms is CART (Classification and Regression Tree).

- **Classification Tree** : When decision or target variable is categorical, the decision tree is classification decision tree.
- **Regression Tree** : When the decision or target variable is continuous variable, the decision tree is called regression decision tree.
- **CART algorithm can be used for building both Classification and Regression** Decision Trees. The impurity measure used in building decision tree in CART is Gini Index. The decision tree built by CART algorithm is always a binary decision tree.
- Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
- Gini index, entropy and twoing rule are some of the frequently used impurity measures.
- Gini Index for a given node t :

$$\text{GINI}(t) = \sum_j p(j \mid t)(1 - p(j \mid t)) = \sum_j p(j \mid t)^2$$

Maximum of $1-1/n_c$ (number of classes) when records are equally distributed among all classes = maximal impurity.

- Minimum of 0 when all records belong to one class = complete purity.
- Entropy at a given node by :

$$\text{Entropy}(t) = \sum_j p(j \mid t) \log p(j \mid t)$$

- Maximum ($\log n_c$) when records are equally distributed among all classes(maximal impurity).
- Minimum (0.0) when all records belongs to one class (maximal purity).
- Entropy is the only function that satisfies all of the following three properties
  1. When node is pure, measure should be zero
  2. When impurity is maximal (i.e. all classes equally likely), measure should be maximal
  3. Measure should obey multistage property
- When a node p is split into k partitions (children), the quality of the split is computed as a weighted sum :

$$\text{GINI}_{\text{split}} = \sum_{i=1}^{k} \frac{n_i}{n} \text{GINI}(i) = \sum_j p(j \mid t)^2$$

where $n_i$ = number of records at child i, and n = number of records at node p.

- A problem with all impurity measures is that they depend only on the number of (training) patterns of different classes on either side of the hyperplane. Thus, if we change the class regions without changing the effective areas of class regions on

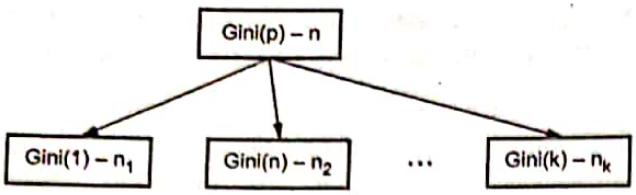



Fig. Q.45.1

either side of a hyperplane, the impurity measure of the hyperplane will not change.

- Thus the impurity measures do not really capture the geometric structure of class distributions. Also, all the algorithms need to optimize on some average of impurity of the child nodes and often it is not clear what kind of average is proper.

**Q.46 Which attribute is best ? How to select best attributes ?**

**Ans. :** • We would like to select the attribute that is most useful for classifying examples.

- Information gain measures how well a given attribute separates the training examples according to their target classification.
- ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.
- In order to define information gain precisely, we use a measure commonly used in information theory, called entropy
- Entropy characterizes the impurity of an arbitrary collection of examples.
- Putting together a decision tree is all a matter of choosing which attribute to test at each node in the tree.
- We shall define a measure called information gain which will be used to decide which attribute to test at each node.
- Information gain is itself calculated using a measure called entropy, which we first define for the case of a binary decision problem and then define for the general case.
- Given a binary categorization, C, and a set of examples, S, for which the proportion of examples categorized as positive by C is p+ and the proportion of examples categorized as negative by C is p-, then the entropy of S is :

$$\text{Entropy}(S) = -p_+ \log_2 (p_+) - p - \log_2 (p_-)$$

Where

S is a sample of training examples

$p_+$ is the proportion of positive examples

$p_-$ is the proportion of negative examples

- Imagine having a set of boxes with some balls in. If all the balls were in a single box, then this would be nicely ordered, and it would be extremely easy to find a particular ball.
- If, however, the balls were distributed amongst the boxes, this would not be so nicely ordered, and it might take quite a while to find a particular ball.
- If we were going to define a measure based on this notion of purity, we would want to be able to calculate a value for each box based on the number of balls in it, then take the sum of these as the overall measure.
- We would want to reward two situations: nearly empty boxes, and boxes with nearly all the balls in. This is the basis for the general entropy measure, which is defined as follows.
- Given an arbitrary categorization, C into categories $c_1, \ldots, c_n$ and a set of examples, S, for which the proportion of examples in $c_i$ is $p_i$, then the entropy of S is :

$$\text{Entropy}(S) = \sum_{i-1}^{n} -p_i \log_2 (p_i)$$

**Q.47 Explain ID3 algorithm.**

**Ans. :** • The calculation for information gain is the most difficult part of this algorithm.

- ID3 performs a search whereby the search states are decision trees and the operator involves adding a node to an existing tree. It uses information gain to measure the attribute to put in each node, and performs a greedy search using this measure of worth.

- The algorithm goes as follows : Given a set of examples (S), categorised in categories $c_i$, then :

1. Choose the root node to be the attribute, A, which scores the highest for information gain relative to S.
2. For each value v that A can possibly take, draw a branch from the node.
3. For each branch from A corresponding to value v, calculate $S_v$. Then :
   i. If $S_v$ is empty, choose the category c default which contains the most examples from S, and put this as the leaf node category which ends that branch.
   ii. If $S_v$ contains only examples from a category c, then put c as the leaf node category which ends that branch.
   iii. Otherwise, remove A from the set of attributes which can be put into nodes. Then put a new node in the decision tree, where the new attribute being tested in the node is the one which scores highest for information gain relative to $S_v$.

The following diagram should explain the ID3 algorithm further.

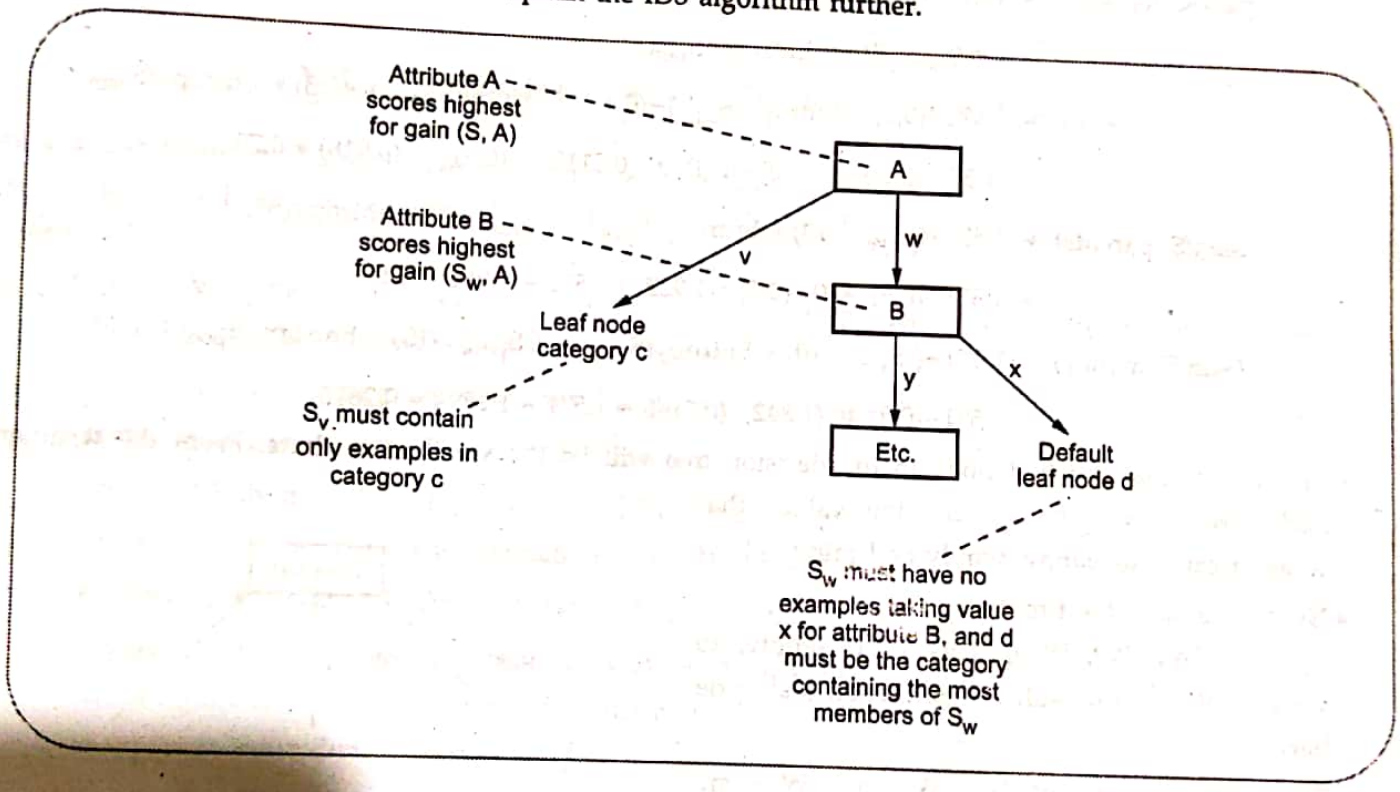



**Fig. Q.47.1**

**Q.48 Suppose we want to train a decision tree using the following instances :**

| Weekend (Example) | Weather | Parents | Money | Decision (Category) |
|---|---|---|---|---|
| W1 | Sunny | Yes | Rich | Cinema |
| W2 | Sunny | No | Rich | Tennis |
| W3 | Windy | Yes | Rich | Cinema |
| W4 | Rainy | Yes | Poor | Cinema |
| W5 | Rainy | No | Rich | Stay in |
| W6 | Rainy | Yes | Poor | Cinema |

| | | | | |
|---|---|---|---|---|
| W7 | Windy | No | Poor | Cinema |
| W8 | Windy | No | Rich | Shopping |
| W9 | Windy | Yes | Rich | Cinema |
| W10 | Sunny | No | Rich | Tennis |

**Ans. :**

$$\text{Entropy}(S) = -p_{\text{cinema}} \log_2(p_{\text{cinema}}) - p_{\text{tennis}} \log_2(p_{\text{tennis}}) - p_{\text{shopping}} \log_2(p_{\text{shopping}}) - p_{\text{stay\_in}} \log_2(p_{\text{stay\_in}})$$

$$= -(6/10) * \log_2(6/10) - (2/10) * \log_2(2/10) - (1/10) * \log_2(1/10) - (1/10) * \log_2(1/10)$$

$$= -(6/10) * -0.737 - (2/10) * -2.322 - (1/10) * -3.322 - (1/10) * -3.322$$

$$= 0.4422 + 0.4644 + 0.3322 + 0.3322 = 1.571$$

and we need to determine the best of :

$$\text{Gain}(S, \text{weather}) = 1.571 - (|S_{\text{sun}}|/10) * \text{Entropy}(S_{\text{sun}}) - (|S_{\text{wind}}|/10) * \text{Entropy}(S_{\text{wind}}) - (|S_{\text{rain}}|/10) * \text{Entropy}(S_{\text{rain}})$$

$$= 1.571 - (0.3) * \text{Entropy}(S_{\text{sun}}) - (0.4) * \text{Entropy}(S_{\text{wind}}) - (0.3) * \text{Entropy}(S_{\text{rain}})$$

$$1.571 - (0.3) * (0.918) - (0.4) * (0.81125) - (0.3) * (0.918) = 0.70$$

$$\text{Gain}(S, \text{parents}) = 1.571 - (|S_{\text{yes}}|/10) * \text{Entropy}(S_{\text{yes}}) - (|S_{\text{no}}|/10) * \text{Entropy}(S_{\text{no}})$$

$$= 1.571 - (0.5) * 0 - (0.5) * 1.922 = 1.571 - 0.961 = 0.61$$

$$\text{Gain}(S, \text{money}) = 1.571 - (|S_{\text{rich}}|/10) * \text{Entropy}(S_{\text{rich}}) - (|S_{\text{poor}}|/10) * \text{Entropy}(S_{\text{poor}})$$

$$1.571 - (0.7) * (1.842) - (0.3) * 0 = 1.571 - 1.2894 = 0.2816$$

- This means that the first node in the decision tree will be the weather attribute. From the weather node, we draw a branch for the values that weather can take: sunny, windy and rainy :

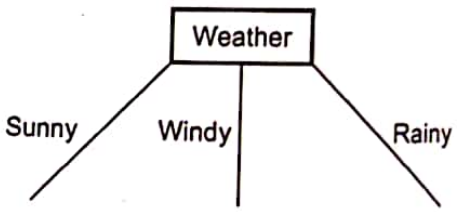



**Fig. Q.48.1**

- Now we look at the first branch. $S_{\text{sunny}}$ = {W1, W2, W10}. This is not empty, so we do not put a default categorization leaf node here.
- The categorizations of W1, W2 and W10 are Cinema, Tennis respectively. As these are not all the same, we cannot put a categorization leaf node here. Hence we put an attribute node here, which we will leave blank for the time being.
- Looking at the second branch, $S_{\text{windy}}$ = {W3, W7, W8, W9}. Again, this is not empty, and they do not all belong to the same class, so we put an attribute node here, left blank for now. The same situation happens with the third branch, hence our amended tree looks like this :

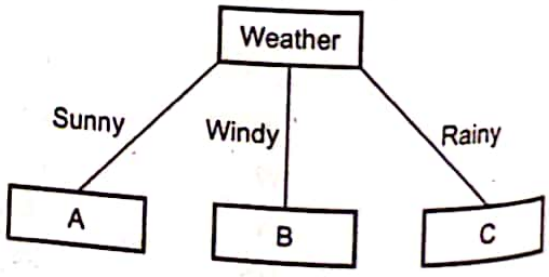



**Fig. Q.48.2**

- In effect, we are interested only in this part of the table :

| Weekend (Example) | Weather | Parents | Money | Decision (Category) |
|---|---|---|---|---|
| W1 | Sunny | Yes | Rich | Cinema |
| W2 | Sunny | No | Rich | Tennis |
| W10 | Sunny | No | Rich | Tennis |

Hence we can calculate :

$$\text{Gain}(S_{sunny}, \text{parents}) = 0.918 - (|S_{yes}|/|S|) * \text{Entropy}(S_{yes}) - (|S_{no}|/|S|) * \text{Entropy}(S_{no})$$
$$= 0.918 - (1/3) * 0 - (2/3) * 0 = 0.918$$

$$\text{Gain}(S_{sunny}, \text{money}) = 0.918 - (|S_{rich}|/|S|) * \text{Entropy}(S_{rich}) - (|S_{poor}|/|S|) * \text{Entropy}(S_{poor})$$
$$= 0.918 - (3/3) * 0.918 - (0/3) * 0 = 0.918 - (3/3) * 0.918 - (0/3) * 0 = 0.918 - 0.918 = 0$$

## 1.10 : Hypothesis Space Search in Decision Tree Learning

**Q.49 Discuss hypothesis space search in decision tree learning.**

**Ans. :** • The hypothesis space searched by ID3 is the set of possible decision trees. ID3 performs a simple-to complex, hill-climbing search through this hypothesis space.

- Begins with the empty tree, then considers progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data.
- The information gain measure guides the hill-climbing search.
- **Hypothesis Space :** Set of possible decision trees
- **Search Method :** Simple-to-Complex Hill-Climbing Search (only a single current hypothesis is maintained (from candidate-elimination method)). No Backtracking!!!
- **Evaluation Function :** Information Gain Measure
- **Batch Learning :** ID3 uses all training examples at each step to make statistically-based decisions (from candidate-elimination method which makes decisions incrementally). The search is less sensitive to errors in individual training examples.
- **By viewing ID3 in terms of its search space and search strategy, following are the capabilities and limitations :**
  1. **ID3 hypothesis space of all decision trees is a complete space of finite discrete-valued functions, relative to the available attributes.**
  2. **It maintains only a single current hypothesis as it searches through the space of decision trees.**
  3. **ID3 in its pure form performs no backtracking in its search**
  4. **ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis**

## 1.11 : Inductive Bias in Decision Tree Learning

**Q.50 Compare ID3 with Candidate-Elimination algorithm.**

**Ans. :**

| ID3 Algorithm | Candidate-Elimination algorithm |
|---|---|
| ID3 searches a complete hypothesis space | The version space CANDIDATE-ELIMINAATION searches an incomplete hypothesis space |
| Its hypothesis space introduces no additional bias | Its search strategy introduces no additional bias. |
| Its inductive bias is solely a consequence of the ordering of hypotheses by its search strategy. | Its inductive bias is solely a consequence of the expressive power of its hypothesis representation |
| It searches incompletely | It searches space completely |

**Q.51 Why Prefer Short Hypotheses ?**

**Ans. : Argument in favor:**

1. Fewer short hypotheses than long hypotheses
2. A short hypothesis that fits the data is unlikely to be a coincidence
3. A long hypothesis that fits the data might be a coincidence

**Argument opposed :**

1. There are many ways to define small sets of hypotheses
2. What is so special about small sets based on size of hypothesis
   - **OCCAM'S RAZOR :** Prefer the simplest hypothesis that fits the data.
   - **Occam's razor** was shown experimentally to be a successful strategy.
   - The term Occam's razor refers to the philosophical idea or scientific principle that of any given set of explanations for an event occurring, it is most likely that the simplest one is the correct one.
   - Occam's razor does not seek to offer complete and absolute proof, but to find the simplest probable answer to a question of why an event happened

**Q.52 State Occam's razor principle.**

☞ **[JNTU : Dec.-16, Marks 2]**

**Ans. :**
- **Prefer the simplest** hypothesis that fits the data
- **Occam's razor** will produce two different hypotheses from the same training examples when it is applied by two learners that perceive these examples in terms of different internal representations

## 1.12 : Issues in Decision Tree Learning



**Q.53 Define pre pruning and post pruning.**

**Ans. :** • In prepruning, a tree is "pruned" by halting its construction early. Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.

• In the postpruning, it removes subtrees from a "fully grown" tree. A subtree at a given node is pruned by removing its branches and replacing it with a leaf. The leaf is labeled with the most frequent class among the subtree being replaced.

**Q.54 Why tree pruning useful in decision tree induction ?**

**Ans. :** When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers. Tree pruning methods address this problem of **overfitting** the data. Such methods typically use statistical measures to remove the least reliable branches.

**Q.55 What is tree pruning ?**

**Ans. :** Tree pruning attempts to identify and remove such branches, with the goal of improving classification accuracy on unseen data

**Q.56 Explain overfitting. What are the reason for overfitting ?**

**Ans. :** • Training error can be reduced by making the hypothesis more sensitive to training data, but this may lead to overfitting and poor generalization.

• Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship.

• Overfitting is when a classifier fits the training data too tightly. Such a classifier works well on the training data but not on independent test data. It is a general problem that plagues all machine learning methods.

• Because of overfitting, low error on training data and high error on test data.

• Overfitting occurs when a model begins to memorize training data rather than learning to generalize from trend.

• The more difficult a criterion is to predict, the more noise exists in past information that need to be ignored. The problem is determining which part to ignore.

• Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

• We can determine whether a predictive model is underfitting or overfitting the training data by looking at the prediction error on the training data and the evaluation data.

• Fig. Q.56.1 shows overfiting.

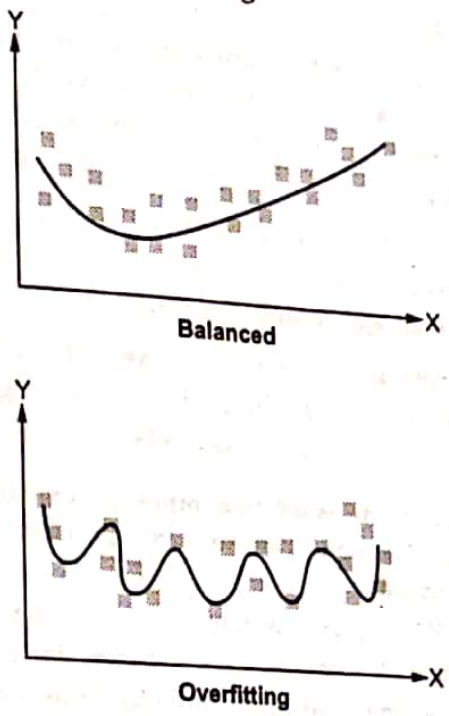



**Fig. Q.56.1 Overfitting**

• Reasons for overfitting

1. Noisy data 2. Training set is too small
3. Large number of features

• Method to avoid overfittings :

1. Limit the number of hidden nodes.
2. Stop training early to avoid a perfect explanation of the training set, and
3. Apply weight decay to limit the size of the weights, and thus of the function class implemented by the network.

**Q.57 Why is tree pruning useful in decision tree induction ? What is a drawback of using a separate set of tuples to evaluate pruning ?**

**Ans. :** • The decision tree built may overfit the training data. There could be too many branches, some of which may reflect anomalies in the training data due to noise or outliers.

- Tree pruning addresses this issue of overfitting the data by removing the least reliable branches.
- This generally results in a more compact and reliable decision tree that is faster and more accurate in its classification of data.
- The drawback of using a separate set of tuples to evaluate pruning is that it may not be representative of the training tuples used to create the original decision tree.
- If the separate set of tuples are skewed, then using them to evaluate the pruned tree would not be a good indicator of the pruned tree's classification accuracy.
- Furthermore, using a separate set of tuples to evaluate pruning means there are less tuples to use for creation and testing of the tree.
- While this is considered a drawback in machine learning, it may not be so in data mining due to the availability of larger data sets.

**Q.58 What is decision tree pruning? Explain error based pruning and reduced error pruning method for tree pruning.**

**Ans. :** • Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances.

- Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting
- Pruning means to change the model by deleting the child nodes of a branch node. The pruned node is regarded as a leaf node. Leaf nodes cannot be pruned.
- A decision tree consists of a root node, several branch nodes, and several leaf nodes.

1. The root node represents the top of the tree. It does not have a parent node, however, it has different child nodes.
2. Branch nodes are in the middle of the tree. A branch node has a parent node and several child nodes.
3. Leaf nodes represent the bottom of the tree. A leaf node has a parent node. It does not have child nodes.

**Error based pruning :**

- Error based pruning can be used to prune a decision tree and it does not require the use of validation data.
- Error-based pruning considers the E errors among the N training examples at a leaf of the tree to give an estimate of the error probability for that node.
- The assumption is that these are E events in N independent trials which is, of course, not perfectly true.
- Using the binomial theorem, condence limits can be calculated for the probability of error for a given condence level
- EBP also performs subtree raising. In the case of subtree raising, an internal node can be replaced by the subtree of one of its children rather than a leaf.
- EBP will estimate that more errors are made than actually observed on the validation set for certainty factors lower than 100. Hence, if we ignore subtree raising, then EBP applied to a validation set will tend to prune more than reduced error pruning (REP) for all CF < 100 and will certainly prune more of the tree for smaller CFs.

**Reduced Error Pruning**

- It is simplest and most understandable method in decision tree pruning.
- This method considers each of the decision nodes in the tree to be candidates for pruning, consist of removing the subtree rooted at that node, making it a leaf node.
- The available data is divided into three parts : the training examples, the validation examples used for pruning the tree, and a set of test examples used to provide an unbiased estimate of accuracy over future unseen examples.
- If the error rate of the new tree would be equal to or smaller than that of the original tree and that subtree contains no subtree with the same property, then subtree is replaced by leaf node, means pruning is done.

- Otherwise don't prune it. The advantage of this method is its linear computational complexity.
- When the test set is much smaller than the training set, this method may lead to over pruning.

**Q.59 What is RULE POST-PRUNING ?**

**Ans. :** • It is method for finding high accuracy hypotheses.

- Rule post-pruning involves the following steps :
  1. Infer decision tree from training set
  2. Convert tree to rules - one rule per branch
  3. Prune each rule by removing preconditions that result in improved estimated accuracy
  4. Sort the pruned rules by their estimated accuracy and consider them in this sequence when classifying unseen instances

**Q.60 Why convert the decision tree to rules before pruning ?**

**Ans. :**

- Converting to rules allows distinguishing among the different contexts in which a decision node is used.
- Converting to rules removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves.
- Converting to rules improves readability. Rules are often easier for to understand

**Q.61 Explain alternative measures for selecting attributes.**

**Ans. :** • One of the decision tree algorithms is CART Classification and Regression Tree).

Classification Tree : When decision or target variable is categorical, the decision tree is classification decision tree.

Regression Tree : When the decision or target variable is continuous variable, the decision tree is called regression decision tree.

CART algorithm can be used for building both Classification and Regression Decision Trees. The impurity measure used in building decision tree in CART is Gini Index. The decision tree built by CART algorithm is always a binary decision tree.

- Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
- Gini index, entropy and twoing rule are some of the frequently used impurity measures.
- Gini Index for a given node t :

$$GINI(t) = \sum_j p(j|t)(1-p(j|t)) = \sum_j p(j|t)^2$$

Maximum of $1-1/n_c$ (number of classes) when records are equally distributed among all classes = maximal impurity.

- Minimum of 0 when all records belong to one class = complete purity.
- Entropy at a given node by :

$$\text{Entropy}(t) = \sum_j p(j|t) \log p(j|t)$$

- Maximum ($\log n_c$) when records are equally distributed among all classes(maximal impurity).
- Minimum (0.0) when all records belongs to one class (maximal purity).
- Entropy is the only function that satisfies all of the following three properties
  1. When node is pure, measure should be zero
  2. When impurity is maximal (i.e. all classes equally likely), measure should be maximal
  3. Measure should obey multistage property
- When a node p is split into k partitions (children), the quality of the split is computed as a weighted sum :

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i) = \sum_j p(j|t)^2$$

where $n_i$ = number of records at child i, and n = number of records at node p.

- A problem with all impurity measures is that they depend only on the number of (training) patterns of different classes on either side of the hyperplane. Thus, if we change the class regions without changing the effective areas of class regions on either side of a hyperplane, the impurity measure of the hyperplane will not change.

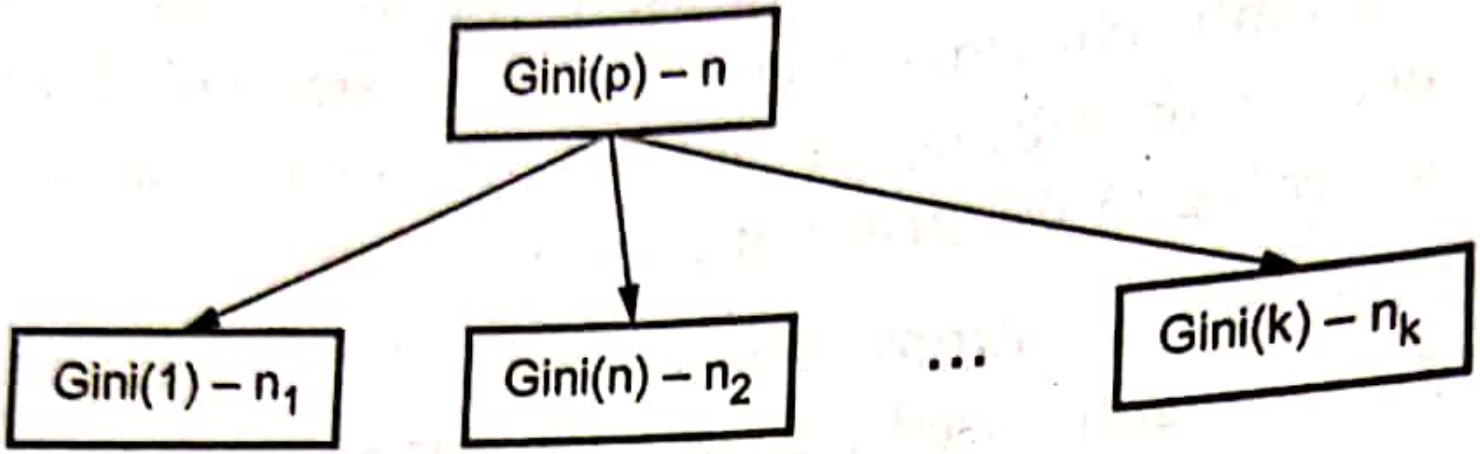



**Fig. Q.61.1**

- Thus the impurity measures do not really capture the geometric structure of class distributions. Also, all the algorithms need to optimize on some average of impurity of the child nodes and often it is not clear what kind of average is proper.