UNIT-5 - PART 1

UNCERTAIN KNOWLEDGE AND REASONING

UNCERTAINTY:-

↳ The term uncertainity means doubtful (or) not sure.

↳ In this chapter we see what an agent should do when the information is not clear.

ACTING UNDER UNCERTAINITY:-

The logical agents discussed so far make the epistemological commitment that the propositions are true, false or unknown.

↳ When an agent knows enough information about its environment, the logical approach enables it to derive plans that are guaranteed to work.

↳ Unfortunately, agents almost never have access to the whole truth about their environment. Therefore agents must act under uncertainity.

⇒ For a logical agent, it might be impossible to construct a complete and correct description of how its actions will work.

Example:

↳ Consider the agent want to drive someone to the airport to catch a flight and the agent is considering a plan A90.

↳ The plan A90 involves leaving home 90 minutes before the flight departs and driving at a reasonable speed.

↳ Eventhough the airport is only about 15 miles away, the agent will ~~not~~ ~~be~~ not be conclude with certainity that plan A90 will get us to the airport in time.

↳ Instead, it reaches the weaker conclusion that "Plan A90 will get us to the airport in time, as long as the car doesn't break down or run out of gas and the agent don't get into an accident and there are no accidents on the bridge".

None of the above conditions can be inferred (deduced), so the plan's success can't be inferred.

⇨ If a logical agent can't conclude that any particular course of action achieves its goal, then it will be unable to act.

⇨ The following approaches can overcome uncertainity to some extent.

1. Conditional planning can overcome uncertainity to some extent only if the agent's sensing actions can provide the required information.
2. Provide the agent with a simple but incorrect theory of the environment that does enable it to derive a plan. But problems arise when events contradict the agent's theory.

⇨ Let us suppose that A_{90} is in fact the right thing to do, i.e. out of all plans, A_{90} is expected to maximize the agent's performance measure.

The performance measure includes:

i. Getting to the airport in time for the flight
ii. Avoiding a long wait at the airport and
iii. Avoiding speeding tickets along the way.

↳ The information the agent has can't guarantee any of these outcomes for A_{90}, but it can provide some degree of belief that they will be achieved.

Handling uncertain knowledge:-

↳ Consider a simple diagnosis example to illustrate the nature of uncertain knowledge. Here diagnosis is a task that almost always involves uncertainity.

↳ Let us try to write the rules for dental diagnosis using first-order logic. We can see how the logical approach breaks down.

Consider the following rule:

"$\forall p$ Symptom$(P, \text{Toothache}) \Rightarrow$ Disease(P, Cavity)".

This rule is wrong. Not all patients with tooaches have cavities; some of them have gum disease, an abscess or one of several other problems.

↳ Unfortunately, in order to make the rule true, we have to add an unlimited list of possible causes.

$\forall p$ Symptom(P, Toothache) $\Rightarrow$ Disease(P, cavity) $\vee$ Disease(P, GumDisease) $\vee$ Disease(P, Abscess) .....

↳ Consider the following rule:

$\forall p$ Disease(P, cavity) $\Rightarrow$ Symptom(P, Toothache)

This rule is not right because not all cavities causes pain. The only way to fix the rule is to make it logically exhaustive. i.e. In left-hand side we can specify th all the causes (or) qualifications required for a cavity to cause a toothache.

⇒ Trying to use first-order logic for medical diagnosis fails for 3 reasons:

1. Laziness:-
   ↳ It is too much work to list the complete set of antecedents or consequents to ensure an exceptionless rule
   ↳ It is too hard to use such rules.

2. Theoretical Ignorance:-
   ↳ Medical science has no complete theory for the domain.

3. Practical Ignorance:-
   ↳ Even if we know all the rules, we might be uncertain about a particular patient because not all the necessary tests have been done.

⇒ The agent's knowledge can at best provide only a degree of belief in the relevant sentences.

↳ The main tool for dealing with degrees of belief will be probability theory

↳ Probability theory assigns to each sentence a numerical degree of belief between 0 and 1

↳ Probability provides a way of summarizing the uncertainity that comes from our laziness and ignorance.

↳ we may not sure for a particular patient, but we believe that there is an 80% chance i.e. a probability of 0.8 that the patient has a cavity if he or she has a toothache.

The belief could be derived from statistical data or from some general rules or from a combination of evidence sources.

⇒ Assigning probability of 0 to a given sentence means that the sentence is false.

⇒ Assigning a probability of 1 to a given sentence means that the sentence is true.

⇒ Probabilities between 0 and 1 corresponds to intermediate degrees of belief in the truth of a sentence. The sentences itself is in fact either true or false.

⇒ A degree of belief is different from a degree of truth. A probability of 0.8 doesn't mean "80% true" but rather an 80% degree of belief.

⇒ All probability statements indicate the evidence with respect to which the probability is being assessed.

↳ As the agent receives new percepts, its probability assessments are updated to reflect the new evidence.

UNCERTAINITY AND RATIONAL DECISIONS:-

⇒ The term rational means sensible (or) reasonable (or) clever

↳ The presence of uncertainity changes the way an agent make decisions.

↳ A logical agent typically has a goal and executes any plan that is guaranteed to achieve it.

↳ An action/plan can be selected or rejected on the basis of whether it achieves the goal, regardless of what other actions might achieve.

↳ When uncertainity enters the picture, this is no longer the case.

Ex: Consider the A_{90} plan for getting to the airport. Suppose it has 95% chance of succeeding, it doesn't mean that it is a good choice. There might be other plans such as A_{120} with higher probabilities of success. Consider the plan A_{1440} that involves leaving home 24 hours in advance. In most cases, this is not a good choice, because it involves an intolerable wait.

⇒ To make a choice, an agent must have preferences between the different possible outcomes of the various plans.

↳ A particular outcome is a completely specified state, including the factors such as whether the agent arrives on time and the length of the wait at the airport.

⇒ Utility theory is used to represent and reason with preferences.

⇒ Utility theory:-

↳ Here utility means "the quality of being useful".

↳ Utility theory says that every state has a degree of usefulness, or utility or benefits.

↳ The agent will prefer states with higher utility.

⇒ Decision-theory:-

↳ Decision theory = Probability theory + utility theory ✓

↳ The fundamental idea of decision theory is that "an agent is rational iff it chooses the action that yields the highest expected utility". This is called the principle of "Maximum Expected Utility (MEU)".

Design for a decision-theoretic agent:-

↳ The following function shows the structure of an agent that uses decision theory to select actions.

```
function DT-AGENT(percept) returns an action
{
  static: belief_state, probabilistic beliefs about the current state of the world
          action, the agent's action.

  update belief_state based on action and percept
  Calculate outcome probabilities for actions,
    Given action descriptions and current belief_state
      Select action with highest expected utility
    Given probabilities of outcomes and utility information
      return action
}
```

⇒ The difference between logical agent and decision-theoritic agent is that the decision-theoritic agent's knowledge of the current state is uncertain.

BASIC PROBABILITY NOTATION:-

⇨ We need a formal language for representing and reasoning with uncertain knowledge.

⇨ Any notation for describing degrees of belief must be able to deal with two main issues:

1. The nature of the sentences to which degrees of belief are assigned.
2. Dependence of the degree of belief on the agent's experience.

⇨ Probability theory uses an extension of propositional logic for its sentences.

Propositions:-

↳ Degrees of belief are always applied to propositions.

↳ Probability theory typically uses a language that is slightly more expressive than propositional logic.

↳ The basic element of the language is the random variable.

↳ Random variable refers to a "part" of the world whose status is unknown.

↳ Capital letters are used to represent random variables.

Ex: Cavity might refer to whether the tooth has a cavity.

↳ Each random variable has a domain of values that it can take on.

Ex: Domain of cavity might be {true, false}.

↳ The simplest kind of proposition asserts that a random variable has a particular value drawn from its domain.

Ex: Cavity = true

↳ Random variables are divided into 3 kinds, depending on the type of the domain:
1. Boolean random variable
2. Discrete random variable
3. Continuous random variable.

1. Boolean Random Variable:-

↳ Boolean Random variables have the domain {true, false}

Ex: Domain of cavity might be {true, false}

↳ Cavity = true proposition is abbreviated as cavity

↳ Cavity = false proposition is abbreviated as ¬cavity

2. Discrete Random variables:-

↳ Discrete random variables include Boolean random variables as a special case, take on values from a countable domain.

Ex: The domain of weather might be {sunny, rainy, cloudy, snow}.

↳ The values in the domain must be mutually exclusive and exhaustive.

3. Continuous Random variables:-

↳ Take on values from the real numbers.

↳ The domain can be either entire real number or some subset such as the interval [0, 1].

Ex: The proposition $X = 4.02$ assents that the random variable X has the exact value 4.02.

↳ Propositions concerning continuous random variables can also be inequalities.

Ex: $X \leq 4.02$ ✓

⇒ The elementary propositions such as Cavity = true and Toothache = false can be combined to form complex propositions using all the standard logical connectives.

Ex: Cavity = true ∧ Toothach = false ✓

The above proposition can also be written as cavity ∧ ¬toothache ✓

Atomic Events:-

↳ Atomic event is useful in understanding the foundations of probability theory.

↳ An atomic event is a "complete specification of the state of the world about which the agent is uncertain.

↳ It can be thought of assigning particular values to all the variable of which the world is composed.

Ex: If the world consists of only the Boolean variables Cavity and Toothache, then there are just 4 distinct atomic events.

Cavity = false ∧ Toothache = false
Cavity = false ∧ Toothache = true
Cavity = true ∧ Toothache = false ✓
Cavity = true ∧ Toothache = true

⇒ Properties of atomic events:-

1. They are mutually exclusive - atmost one can actually be the case. ie. We can take only one truth value at a time but not both.

   Ex: Cavity ∧ toothache and cavity ∧ ¬toothache cannit both be the case.

2. The set of all possible atomic events is exhaustive - atleast one must be the case i.e. the disjunction of all atomic events is logically equivalent to true.

3. Any particular atomic event entails the truth or falsehood of every proposition whether simple or complex.

   Ex: The atomic event cavity ∧ ¬toothache entails the truth of cavity and the falsehood of toothache.

4. Any proposition is logically equivalent to the disjunction of all atomic events that entails the truth of the proposition.

   Ex: The proposition cavity is equivalent to disjunction of the atomic events cavity ∧ toothache and cavity ∧ ¬toothache.

PRIOR PROBABILITY:-

⇒ Prior probability is also called unconditional probability.

⇒ Prior probability or unconditional probability associated with a proposition 'a' is the degree of belief accorded to it in the absence of any other information.

⇒ It is written as P(a) → Prior probability of proposition 'a'.

Ex: If the prior probability that a person have a cavity is 0.1, then it can be written as P(cavity) = 0.1.

⇒ P(a) can be used only when there is no other information. As soon as otherwise new information is known, we must reason with the conditional probability

⇒ Sometimes we will also use probabilities of all possible values of a random variable.

Ex: P(Weather) denotes a vector of possible probabilities of each individual state of the weather.

Thus instead of writing 4 equations

P(Weather = sunny) = 0.7

P(Weather = rainy) = 0.2

P(Weather = cloudy) = 0.08

P(Weather = snow) = 0.02

we may simply write

$$P(\text{Weather}) = \langle 0.7, 0.2, 0.08, 0.02 \rangle$$

The above statement defines a prior probability distribution for the random variable weather.

Joint-Probability Distribution:-

Denoting the probabilities of all the combinations of the values of a set of random variables is called joint probability distribution.

Ex: P(Weather, Cavity) is called the joint probability distribution of weather and cavity. This can be represented by a 4×2 table of probabilities as shown below:

Weather	Sunny	rainy	Cloudy	Snow
Cavity = true	0.144	0.02	0.016	0.02
cavity = false	0.576	0.08	0.064	0.08

Figure: Joint-probability distribution of weather and cavity.

⇒ Notice that the probabilities in the joint distribution sum to 1.

i.e. $0.144 + 0.02 + 0.016 + 0.02 + 0.576 + 0.08 + 0.064 + 0.08 = 1$

Full-Joint Probability Distribution:-

A joint-probability distribution that covers the complete set of random variables used to describe the world is known as full-joint probability distribution.

Ex: If the world consists of just 3 random variables Cavity, Toothache, and ~~Weather~~ Catch then the full-joint distribution is given by

P(Cavity, Toothache, Catch) i.e. P(Cavity, Toothache, Catch)

This joint probability can be represented as a 2x2x4 table with 16 entries as shown below.

	toothache		¬toothache	
	Catch	¬Catch	Catch	¬Catch
Cavity	0.108	0.012	0.072	0.008
¬Cavity	0.016	0.064	0.144	0.576

Figure: A full joint distribution for Toothache, Cavity and Catch.

Prior probability for continuous variables:-

For continuous variables, it is not possible to write out the entire distribution as a table, because there are infinitely many values.

For continuous variables, define the probability that a random variable takes on some value x as a parameterized function of x.

Ex: $P(X=x) = U[18,26](x)$

The above statement expresses the belief that X is distributed uniformly between 18 and 26.

Probability distribution for continuous variables are called probability density functions.

POSTERIOR PROBABILITY (OR) CONDITIONAL PROBABILITY:-

⇒ Once the agent has obtained some evidence about the previously unknown random variables making up the domain, prior probabilities are no longer applicable. We must use conditional or posterior probabilities.

↳ Conditional probability is denoted by P(a|b), where a and b are any propositions. This is read as "the probability of a, given that all we know is b".

Ex: P(Cavity|toothache) = 0.8

indicates that if a patient is observed to have a toothache, then the probability of patient's having cavity is 0.8.

⇒ Conditional probabilities can be defined interms of unconditional probabilities.

i.e. $P(a|b) = \frac{P(a \wedge b)}{P(b)}$ holds whenever $P(b) > 0$. ✓

This equation can also be written as

$P(a \wedge b) = P(a|b) \cdot P(b)$ which is called the product rule. ✓

↳ we can also write the above equation in another form as follows:

$P(a \wedge b) = P(b|a) \cdot P(a)$ ✓

⇒ We can also use the notation P for conditional distributions.

↳ $P(X/Y)$ gives the values of $P(X=x_i | Y=y_j)$ for each possible i, j. ✓

↳ Apply the product rule to each case where the propositions a and b assert particular values of X and Y respectively.

We obtain the following equations:

$P(X=x_1 \wedge Y=y_1) = P(X=x_1 | Y=y_1) \cdot P(Y=y_1)$

$P(X=x_2 \wedge Y=y_2) = P(X=x_2 | Y=y_2) \cdot P(Y=y_2)$

⋮

We can combine all these into the single equation

$P(X,Y) = P(X/Y) \cdot P(Y)$

⇒ We know that $P(a|b)$ is the probability of a, given that all we know is b. When additional information 'c' is available, the degree of belief in a is $P(a|b \wedge c)$, c may tell us directly whether a is true or false.

Ex: If a patient who complains of toothache is examined and discover a cavity, then we have additional evidence cavity.

∴ we conclude that $P(\text{cavity} | \text{toothache} \wedge \text{cavity}) = 1.0$ ✓

THE AXIOMS OF PROBABILITY:-

⇒ Axioms are used to define the probability scale and its endpoints.

↳ The basic axioms are:

1. All probabilities are between 0 and 1.

   i.e. For any proposition 'a' $0 \leq P(a) \leq 1$

2. Necessarily true i.e. valid propositions have probability 1 and necessarily false i.e. unsatisfiable propositions have probability 0.

   $P(true) = 1$ ✓

   $P(false) = 0$ ✓

3. The probability of a disjunction is given by

   $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$ ✓

This rule is easily remembered by noting that the case where 'a' holds, together with the cases where 'b' holds, certainly cover all the cases where $a \vee b$ holds.

⇒ The above 3 axioms are often called "kolmogorov's axioms" in honor of the Russian mathematician Andrei kolmogorov.

⇒ The axioms deal only with prior probabilities rather than conditional probabilities.

Using the axioms of probability:-

⇒ We can derive a variety of useful facts from the basic axioms.

↳ Consider the familiar axiom 3

$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$ ✓

We can derive the familiar rule for negation by substituting $\neg a$ for b in axiom 3.

i.e. $P(a \vee \neg a) = P(a) + P(\neg a) - P(a \wedge \neg a)$ (by axiom 3 with $b = \neg a$)

$P(true) = P(a) + P(\neg a) - P(false)$ (by logical equivalence)

✓ $1 = P(a) + P(\neg a) - 0$ (by axiom 2)

$\therefore P(\neg a) = 1 - P(a)$ (by algebra)

⇒ Let the discrete variable D have the domains $\langle d_1, d_2, \ldots d_n \rangle$.

Then $\boxed{\sum_{i=1}^{n} P(D=d_i) = 1}$ ✓

i.e. any probability distribution on a single variable must sum to 1.

⇒ It is also true that any joint probability distribution on any set of variables must sum to 1

⇒ We know that any proposition 'a' is equivalent to the disjunction of all atomic events in which 'a' is true. Call this set of events e(a)

↳ We also know that atomic events are mutually exclusive. so the probability of any conjunction of atomic events is zero by axiom2.

↳ Hence, from axiom 3, we can derive the following simple relationship:

"The probability of a proposition is equal to the sum of the probabilities of the atomic events in which it holds"..

i.e. $\boxed{P(a) = \sum_{e_i \in e(a)} P(e_i)}$ ✓

The above equation provides a simple method for computing the probability of any proposition, given a full-joint distribution that specifies the probabilities of all atomic events.

Why the axioms of probability are reasonable:-

$P(a) = \sum_{e_i \in e(a)} P(e_i)$ $\qquad$ $\sum_{i=1}^{n} P(D=d_i) = 1$ ✓

$P(a \vee \neg a) = P(a) + P(\neg a) - P(a \wedge \neg a)$

$P(true) = P(a) + P(\neg a) - P(false)$ →

$1 = P(a) + P(\neg a) - 0$ →

$1 - P(a) = P(\neg a)$ →

⇒ If the complete set of variables can be divided into independent subsets, then the full-joint can be factored into seperate joint distributions on those subsets.

Ex: $P(C_1, \dots C_n) = P(C_1) \cdots P(C_n)$

⇒ Independence assertions can help in reducing the size of the domain representation and the complexity of the inference problems.

BAYES' RULE AND ITS USE:-

⇒ We know that $\boxed{P(a|b) = \frac{P(a \wedge b)}{P(b)}}$

From this we can obtain the product rule in two forms

$$P(a \wedge b) = P(a|b) \cdot P(b)$$

$$P(a \wedge b) = P(b|a) \cdot P(a)$$

↳ Equate the two right-hand sides

$$P(a|b) \cdot P(b) = P(b|a) \cdot P(a)$$

↳ Divide by $P(a)$, we get

$$\frac{P(a|b) \cdot P(b)}{P(a)} = \frac{P(b|a) \cdot P(a)}{P(a)}$$

$$\therefore \boxed{P(b|a) = \frac{P(a|b) \cdot P(b)}{P(a)}}$$

The above equation is known as Bayes' rule or Bayes' law or Bayes' theorem. This simple equation underlies all modern AI systems for probabilistic inference.

⇒ Bayes' rule for multivalued variables can be written as

$$\boxed{P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}}$$

⇒ The more general version of Bayes' rule conditionalized on some background evidence e is:

$$\boxed{P(Y|X,e) = \frac{P(X|Y,e) \cdot P(Y|e)}{P(X|e)}}$$

Applying the bayes' rule : The simple case :-

⇒ Bayes' rule requires one conditional probability and two unconditional probabilities to calculate one conditional probability.

↳ We can use Bayes' rule in medical diagnosis. In medical diagnosis we have conditional probabilities on casual relationships and we want to derive a diagnosis.

Example:

↳ Let a doctor knows that the disease "meningitis" causes the patient to have "stiffneck" problem.

↳ Doctor knows that 50% of the time the patient have stiffneck because of meningitis.

↳ Doctor also knows some unconditional facts:

The prior probability of a patient has a stiffneck is $1/20$

The prior probability of a patient has a meningitis is $\frac{1}{50,000}$

↳ Let the symbol 's' denotes stiffneck and
'm' denotes meningitis

↳ We have $P(s) = \frac{1}{20}$

$$P(m) = \frac{1}{50,000}$$

$$P(s|m) = 50\% = 0.5$$

$$\therefore P(m|s) = \frac{P(s|m) \cdot P(m)}{P(s)}$$

$$= \frac{0.5 \times 1/50,000}{1/20}$$

$$= \frac{0.5 \times 20}{50,000}$$

$$= \frac{10}{50,000}$$

$$= \frac{1}{5,000} = 0.0002$$

It means that 1 in 5000 patients with a stiffneck have meningitis

⇒ We can also apply normalization concept to Bayes' rule and we have

$$P(m|s) = \frac{P(s|m) \cdot P(m)}{P(s)}$$

$$= \frac{1}{P(s)} (P(s|m) \cdot P(m))$$

$$= \alpha \cdot \langle P(s|m) \cdot P(m), P(s|\neg m) P(\neg m) \rangle$$

↳ The general form of Bayes' rule with normalization is:

$$\boxed{P(Y|X) = \alpha \cdot P(X|Y) \cdot P(Y)}$$

Using Bayes' rule : Combining evidence

⇒ We have seen that Bayes' rule can be useful for answering probabilistic queries conditioned on one evidence for example stiffneck.

↳ In practice, the probabilistic information is often available in the form P(effect | cause) i.e. Because of cause we have effect.

⇒ What happens if we have two or more evidence variables

Ex: $P(\text{cavity} | \text{toothache} \wedge \text{catch})$

Here we have two evidence variables toothache and catch.

↳ If we know the full-joint distribution, one can read off the answer:

$P(\text{cavity} | \text{toothache} \wedge \text{catch}) = \alpha \langle 0.108, 0.016 \rangle \simeq (0.871, 0.129)$

However, this approach will not scale up to large number of evidence variables.

↳ We can use Bayes' rule to reformulate the problem

$$P(\underset{b}{\text{cavity}} | \underset{a}{\text{toothache} \wedge \text{catch}}) = \frac{P(\text{toothache} \wedge \text{catch} | \text{cavity}) \cdot P(\text{cavity})}{P(\text{toothache} \wedge \text{catch})}$$

$$= \frac{1}{P(\text{toothache} \wedge \text{catch})} \cdot P(\text{toothache} \wedge \text{catch} | \text{cavity}) \cdot P(\text{cavity})$$

$$= \alpha \cdot P(\text{toothache} \wedge \text{catch} | \text{cavity}) \cdot P(\text{cavity}) \rightarrow (1)$$

Here, we need to know the conditional probabilities of the conjunction of toothache and catch for each value of cavity.

This might be feasible for just two evidence variables, but again it will not scale up. ie. if there are 'n' possible evidence variables then we need to know 2^n conditional probabilities.

So we need to find some additional assertions about the domain that will enable us to simplify the expressions.

↳ The notion of independence provides a clue.

The variables toothache and catch are independent given the presence or absence of a cavity.

Mathematically this property written as

$$P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) = P(\text{toothache} \mid \text{cavity}) \cdot P(\text{catch} \mid \text{cavity}) \rightarrow (2)$$

This equation expresses the conditional independence of toothache and catch given cavity.

Substitute equation (2) in equation (1), we get

$$P(\text{cavity} \mid \text{toothache} \wedge \text{catch}) = \alpha \cdot P(\text{toothache} \mid \text{cavity}) \cdot P(\text{catch} \mid \text{cavity}) \cdot P(\text{cavity}) \rightarrow (3)$$

⇒ The general definition of conditional independence of two variables X and Y given a third variable Z is:

$$\boxed{P(X, Y \mid Z) = P(X \mid Z) \cdot P(Y \mid Z)}$$

As with absolute independence, the equivalent forms

$P(X \mid Y, Z) = P(X \mid Z)$ (or) $P(Y \mid X, Z) = P(Y \mid Z)$ can also be used.

⇒ Absolute independence assertions allow a decomposition of the full-joint distribution into smaller pieces. The same is true for conditional independence assertions.

i.e. $P(\text{Toothache, Catch, Cavity}) = P(\text{Toothache, Catch} \mid \text{cavity}) \cdot P(\text{Cavity})$ (using product rule)

$= P(\text{Toothache} \mid \text{cavity}) P(\text{Toothach}$ ~~(struck through)~~

$= P(\text{Toothache} \mid \text{cavity}) \cdot P(\text{catch} \mid \text{cavity}) \cdot P(\text{cavity})$

(using equation (3))

MYCIN – Expert System

⇒ Using conditional independence for n symptoms, the size of the representation grows as O(n) instead of $O(2^n)$

Thus, conditional independence assertions can allow probabilistic systems to scale up and are much more commonly available than absolute independent assertions.

⇒ The dentistry example illustrates a commonly occuring pattern in which a single cause directly influences a number of effects, all of which are conditionally independent, given the cause.

The full-joint distribution can be written as

$$P(\text{Cause}, \text{Effect}_1, \text{Effect}_2, \ldots \text{Effect}_n) = P(\text{Cause}) \cdot \prod_i P(\text{Effect}_i | \text{Cause})$$

This probability distribution is called a naive Bayes' model.

"Naive" because it is often used (as a simplifying assumption) in cases where the "effect" variables are not conditionally independent given the "cause" variable.

"Bayesian classifier" → somewhat careless usage.

Idiot Bayes model.

P(Cause)

UNIT-VI - PART-2

LEARNING

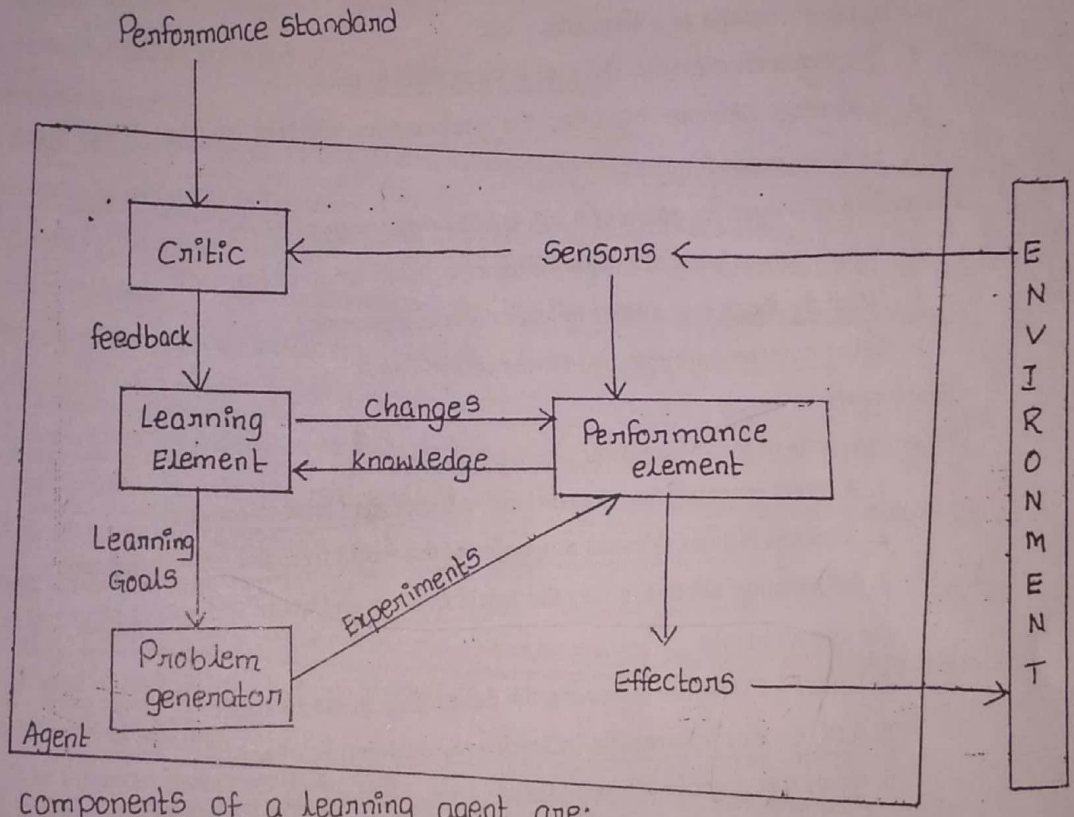
INTRODUCTION:-

⇒ Learning is essential for unknown environments.

↳ Learning modifies the agent's decision mechanisms to improve performance.

Learning Agents:-

⇒ Following figure shows the structure of learning agents.



⇒ Basic components of a learning agent are:

1. Learning Element: Responsible for making improvements.
   ↳ The learning element uses feedback from the Critic on how the agent is doing and determines how the performance element should be modified to do better in the future.
2. Performance Element:-
   ↳ Responsible for selecting external actions.
   ↳ It takes percepts and decides on actions.
3. Critic:-
   ↳ Tells the learning element how well the agent is doing with respect to a fixed performance standard.

4. Problem generator:-

↳ Responsible for suggesting actions that will lead to a new and informative experience.

LEARNING FROM OBSERVATIONS

The idea behind learning is that percepts should be used not only for acting, but also for improving the agent's ability to act in future.

Learning takes place as the agent observes its interactions with the world and its own decision-making processes.

Learning takes many forms, depending on the nature of the performance element, the component to be improved, and the available feedback.

FORMS OF LEARNING :

Learning agent contains two elements:

1. ***Performance element:*** Decides what actions to take
2. ***Learning element:*** Modifies the performance element so that better decisions can be taken in the future.

The design of a learning element is affected by three major issues:

- Which *components* of the performance element are to be learned?
- What *feedback* is available to learn these components?
- What *representation* is used for the components?

Components:

The components of these agents include the following:

1. A direct mapping from conditions on the current state to actions.
2. A means to infer relevant properties of the world from the percept sequence.
3. Information about the way the world evolves and about the results of possible actions the agent can take.
4. Utility information indicating the desirability of world states.
5. Action-value information indicating the desirability of actions.
6. Goals that describe classes of states whose achievement maximizes the agent's utility.

Each of these components can be learned from appropriate feedback.

Example: Consider an agent training to become a taxi driver.

1. Every time the instructor says "Brake", the agent can learn a condition-action rule for when to brake (component1).
2. By seeing many camera images, it can learn to recognize them (component2).

Ammu chellam om

3. By trying actions agent can learn to recognize them- Braking on wet road it can learn the effects of actions (component3)

Feedback:

The type of feedback available for learning is usually the most important factor in determining the nature of the learning problem that the agent faces. There are 3 types of feedback/learning:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**

1. Supervised learning

A correct answer for each example or instance is available. *It involves learning a function from examples of its inputs and outputs.*

Cases 1, 2 and 3 are instances of supervised learning. In (1), the agent learns a condition-action rule for braking- this is a function from states to a Boolean output (i.e. brake or not brake). In (2), the agent can learn a function from images to a Boolean output.

For fully observable environments, the agent can observe the effects of its actions and hence the agent can use supervised learning methods.

For partially observable environments, the problem is more difficult because the immediate effects might be invisible.

2. Unsupervised learning

It involves learning a pattern in the input when no specific output values are supplied. It is mainly used in probabilistic learning system.

3. Reinforcement learning

It is the most general of 3 categories. Here learning pattern is rather than being told by a teacher, *it learns from reinforcement, i.e. by occasional rewards.*

Example: The lack of a tip at the end of the journey, gives the agent some indication that its behavior is undesirable.

Representation

The representation of the learned information also plays a very important role in determining how the learning algorithm must work.

There are several representation schemes.

1. Linear-weighted polynomials
2. Propositional and first-order logical sentences for all of the components in a logical agent.
3. Probabilistic descriptions.

The last major factor in the design of learning systems is the availability of prior knowledge.

2. INDUCTIVE LEARNING :-

↳ Simplest form : Learning a function from examples.

↳ The input for deterministic supervised learning is the correct value of the unknown function for particular inputs.

We must try to recover the unknown function or something close to it.

We can define an example is a pair $(x, f(x))$, where x is the input and $f(x)$ is the output of the function applied to x.

⇒ The task of pure inductive inference (or) induction is:

" Given a set of examples of f, return a function h that approximates f i.e. which is close to f ".

$h \cong f$ ✓

The function h is called a hypothesis.

↳ Finding a suitable hypothesis can be difficult. The reason that learning is difficult from a conceptual point of view is that it is not easy to tell whether any particular h is a good approximation of f.

↳ The hypothesis space describes the set of hypotheses under consideration.

Ex: Polynomials, Sinusoidal functions, Propositional logic, Predicate logic etc.

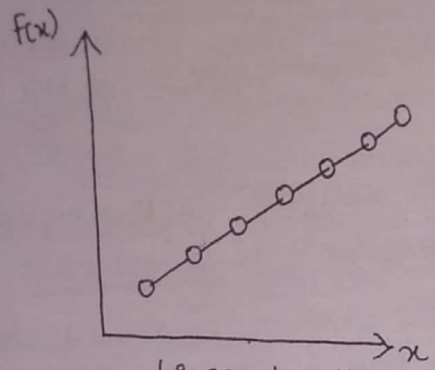
Example:

Consider fitting a function of a single variable to some data points.

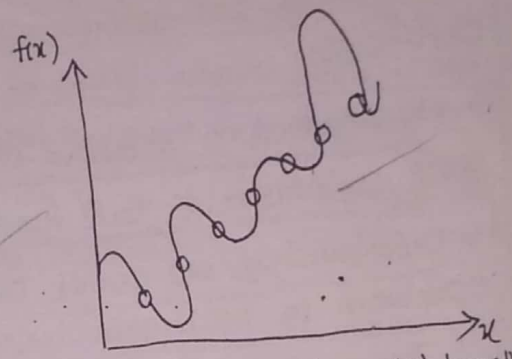
↳ Examples are $(x, f(x))$ pairs, where both x and $f(x)$ are real numbers.

↳ Hypothesis space H is the set of polynomial of degree atmost K such as $3x^2+2$, x^7-4x^3 and so on.

↳ The following figure shows some data with an exact fit by a straightline and a degree-7 polynomial.



Linear hypothesis



degree-7 polynomial hypothesis.

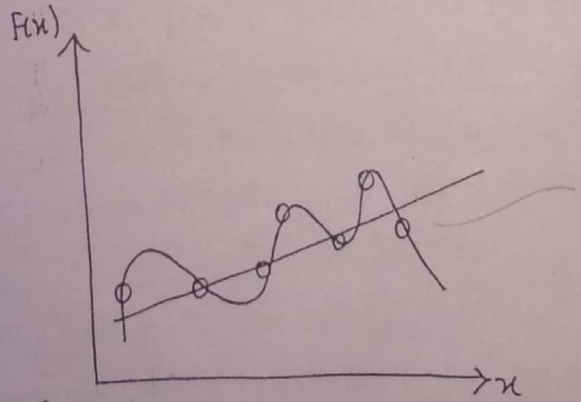
↳ A hypothesis which agrees with all data is called a consistent hypothesis.

↳ First issue in the inductive learning

" How do we choose from among multiple consistent hypotheses?

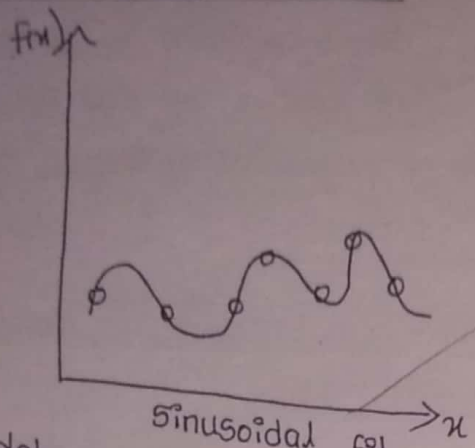
Answer is: Ockham's razor. i.e. prefer the simplest hypothesis consistent with the data.

↳ The following figure shows a hypothesis for some other kind of data.



There is no consistent straight line for this data set. It requires degree-6 polynomial for an exact fit.

↳ The choice of hypothesis space can strongly influence the task of finding a suitable function.



Sinusoidal fit.

The data can be fit exactly by a simple function of the form $ax+b+c\cdot\sin x$.

↳ We say that a learning problem is realizable if the hypothesis space contains the true function. otherwise it is unrealizable.

↳ Unfortunately, we can't always tell whether a given learning problem is realizable because the true function is unknown.

Two ways to fix this problem is.

1. Use prior knowledge to derive a hypothesis space.
2. Use largest possible hypothesis space.

LEARNING DECISION-TREES:-

⇒ Decision tree learning is one of the simplest and most successful forms of learning algorithms.

Decision trees as performance elements:-

⇒ A decision tree takes a set of attributes as input and returns a "decision" - the predicted output value for the input.

↳ The input attributes can be discrete or continuous.

↳ The output values can also be discrete or continuous.

Learning a discrete-valued function is called classification learning.

Learning a continuous-valued function is called regression.

⇒ Here we will use Boolean-classification in which each example is classified as true (positive) or false (negative).

→ Making decisions:-

↳ A decision tree reaches its decision by performing a sequence of tests.

↳ Each internal node in the tree corresponds to testing the value of the one of the attribute.

↳ Branches from the nodes are labeled with the possible values of the test.

↳ Each leaf node in the tree specifies the value to be returned if that leaf is reached.

Example:-

⇒ A simpler example is provided by the problem of whether to wait for a table at a restaurant.

The aim is to learn a definition for the goal predicate Willwait.

↳ The problem contains the following list of examples:-

1. Alternate : Whether there is a suitable alternative restaurant nearby.
2. Bar : Whether the restaurant has a suitable bar area to wait in.
3. Fri/Sat : True on Fridays and Saturdays
4. Hungry : Whether we are Hungry
5. Patrons: How many people are in the restaurant (None, Some, Full).

6. Price : The restaurant's price range ($, $$, $$$)

7. Raining: Whether it is raining outside.

8. Reservation: Whether we made a reservation.

9. Type : The kind of restaurant (French, Italian, Thai or burger)

10. Wait Estimate : The wait estimated by the host (0-10m, 10-30, 30-60, >60)

The decision-tree for the above attributes is shown in the following figure:

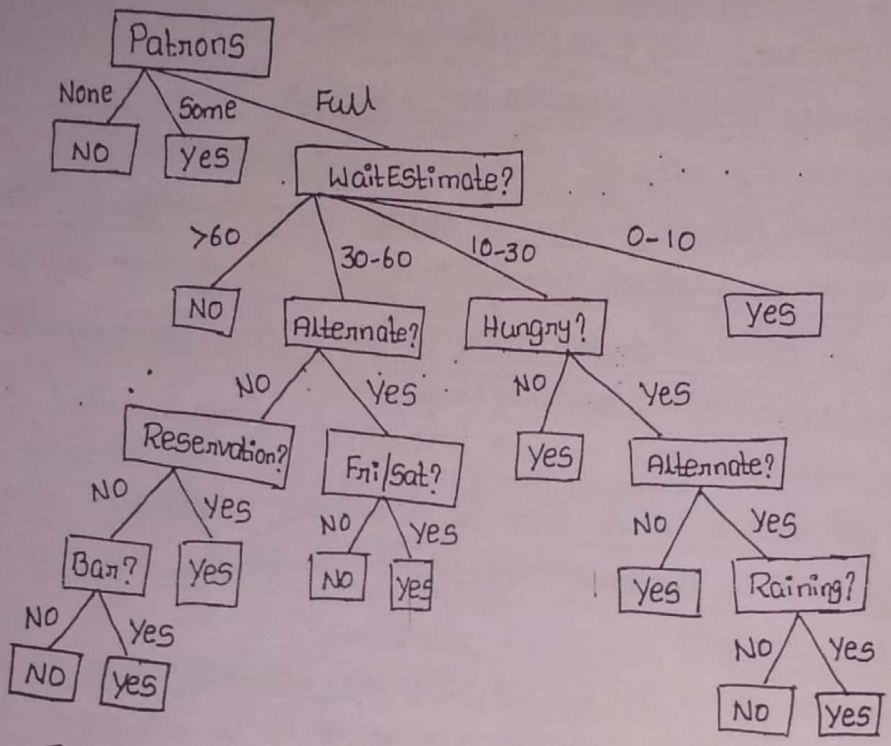


Figure: A decision-tree for deciding whether to wait for a table.

NOTE: The tree doesn't use the price and Type attributes, because they are considered as irrelevant.

⇒ Examples are processed by the tree starting at the root and following the appropriate branch until a leaf is reached.

Ex: An example with patrons = Full and WaitEstimate = 0-10 will be classified as positive i.e. Yes we will wait for a table.

EXPRESSIVENESS OF DECISION TREES:-

⇨ Any particular decision tree hypothesis for the "will wait" predicate can be defined as

$\forall s,\ \text{willwait}(s) \Leftrightarrow (P_1(s) \vee P_2(s) \vee \cdots \vee P_n(s))$

where each $P(s)$ is a conjunction of tests corresponding to a path from the root of the tree to a leaf with a positive outcome.

⇨ Decision tree describes a relationship between willwait and some logical combination of attribute values.

⇨ We can't use decision trees to represent tests that refer to two or more different objects.

⇨ Decision trees are fully expressive within the class of propositional languages i.e. any boolean function can be written as a decision-tree.

⇨ Decision trees can represent many functions with much smaller trees.

↳ For some kind of functions, however this is a real problem.

1. For example if the function is the parity function, which returns 1 iff an even number of inputs are 1, then exponentially large decision trees will be needed.
2. It is also difficult to use a decision tree to represent a majority function, which returns 1 if more than half of its inputs are 1.

↳ In other words, decision trees are good for some kinds of functions and bad for others.

⇨ Consider the set of boolean functions on 'n' attributes.

How many different functions are in this set?

Ans: The function is defined by its truth table. The truth table has 2^n rows. We can consider the "answer" column of the table as a 2^n-bit number that defines the function.

If it takes 2^n bits to define the function, there are 2^{2^n} different functions on 'n' attributes.

INDUCING (OR) LEARNING DECISION TREES FROM EXAMPLES:-

⇨ An example for a Boolean decision tree consists of a vector of input attributes 'x' and a single boolean output value 'y'.

↳ A set of examples $(x_1, y_1), (x_2, y_2), \ldots (x_{12}, y_{12})$ is shown below.

	Attributes										
Example	Alternate	Bar	Fri/Sat	Hungry	Patrons	Price	Rain	Reservation	Type	Wait Estimate	Goal WillWait
x_1	yes	NO	NO	yes	Some	$$$	NO	yes	French	0-10	yes
x_2	yes	NO	No	Yes	Full	$	NO	NO	Thai	30-60	NO
x_3	NO	yes	NO	NO	Some	$	NO	NO	Burger	0-10	Yes
x_4	yes	NO	yes	Yes	Full	$	yes	NO	Thai	10-30	yes
x_5	yes	No	yes	No	Full	$$$	NO	yes	French	>60	NO
x_6	NO	yes	No	yes	Some	$$	yes	yes	Italian	0-10	yes
x_7	NO	yes	NO	NO	None	$	yes	NO	Burger	0-10	NO
x_8	NO	No	No	yes	Some	$$	yes	yes	Thai	0-10	yes
x_9	No	yes	yes	No	Full	$	yes	NO	Burger	>60	NO
x_{10}	yes	yes	yes	yes	Full	$$$	NO	yes	Italian	10-30	NO
x_{11}	NO	NO	NO	NO	None	$	NO	NO	Thai	0-10	NO
x_{12}	yes	yes	yes	yes	Full	$	NO	NO	Burger	30-60	yes

Figure: Examples for the restaurant domain.

Positive examples:-

Positive examples are the one in which goal Willwait is true (i.e. Willwait = yes).

Ex: $x_1, x_3, x_4, x_6, x_8, x_{12}$

Negative examples:-

Negative examples are the one in which goal Willwait is false i.e. Willwait = No.

Ex: $x_2, x_5, x_7, x_9, x_{10}, x_{11}$

Training set:

Complete set of examples is called the training set.

⇒ The problem of finding a decision-tree that agrees with the training set might be difficult.

↳ One trivial solution is: Construct a tree with one branch for each example of the training set.

This approach works perfectly for the samples in the training set. This may not work for new examples.

↳ Better solution is: Find a smallest decision-tree that is consistent with the examples. Unfortunately finding smallest decision tree is an intractable problem.

⇒ Many algorithms exist for constructing reasonably small trees.

⇒ The basic idea behind the DECISION-TREE-LEARNING algorithm is:

"Test the most important attribute first".

↳ The most important or good attribute splits the examples into subsets that are "all positive" (or) "all negative".

⇒ The DECISION-TREE-LEARNING algorithm is recursive:

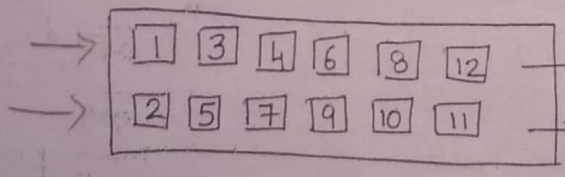
↳ There are 4 cases to consider for these recursive problems

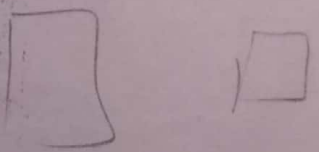
1. If there are some positive and some negative examples left, then choose the best attribute to split them.
2. If all the remaining examples are positive or negative then we are done. ie. We can answer yes or NO.
3. If there are no examples left, it means that no such examples has been observed, return a default value calculated from the majority classification.
4. If we have positive and negative examples left but no attributes to split them, we are in trouble. This may be caused by incorrect (noise) data or by lack of information or by truly non-deterministic domain.

One way to solve this problem is to use a majority function.

Example:

Step1 : Classify the examples into positive and negative sets.





Step 2: Now we have to select the most important attribute first. Consider the attributes 'Type' and 'patrons'

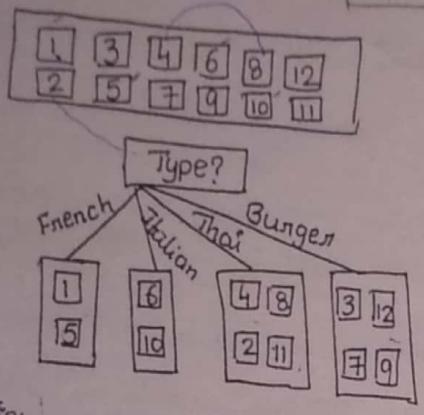


Figure: Splitting the examples by testing 'type' attribute

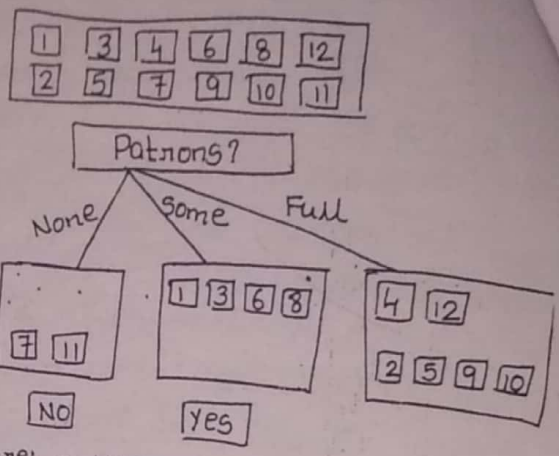


Figure: Splitting the examples by testing "Patrons" attribute.

⇒ "Type" attribute is a poor attribute because it splits the examples into 4 sets, each of which has the same number of positive and negative examples.

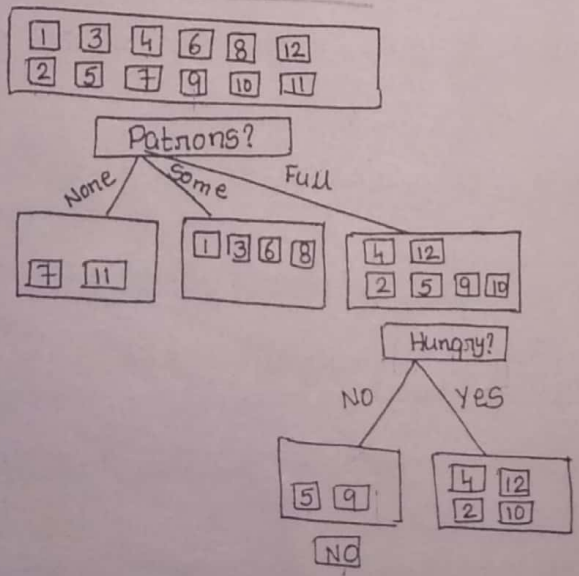
⇒ "Patrons" is a fairly important attribute because:

When Patrons = None, we are left with only negative examples, we can answer NO.

When Patrons = Some, we are left with only positive examples, we can answer yes.

When Patrons = Full, we are left with a mixed set of examples, which needs further classification.

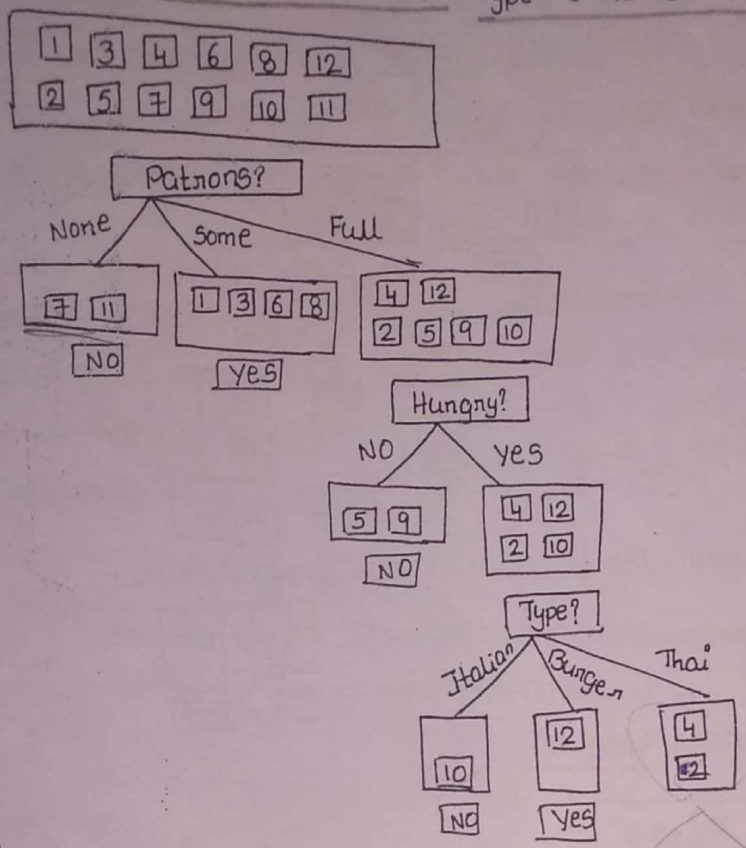
Step 3: Consider the next best attribute for classification. "Hungry" is the best choice.



Patrons
↓
Hungry
↓
Type
↓
Fri/Sat

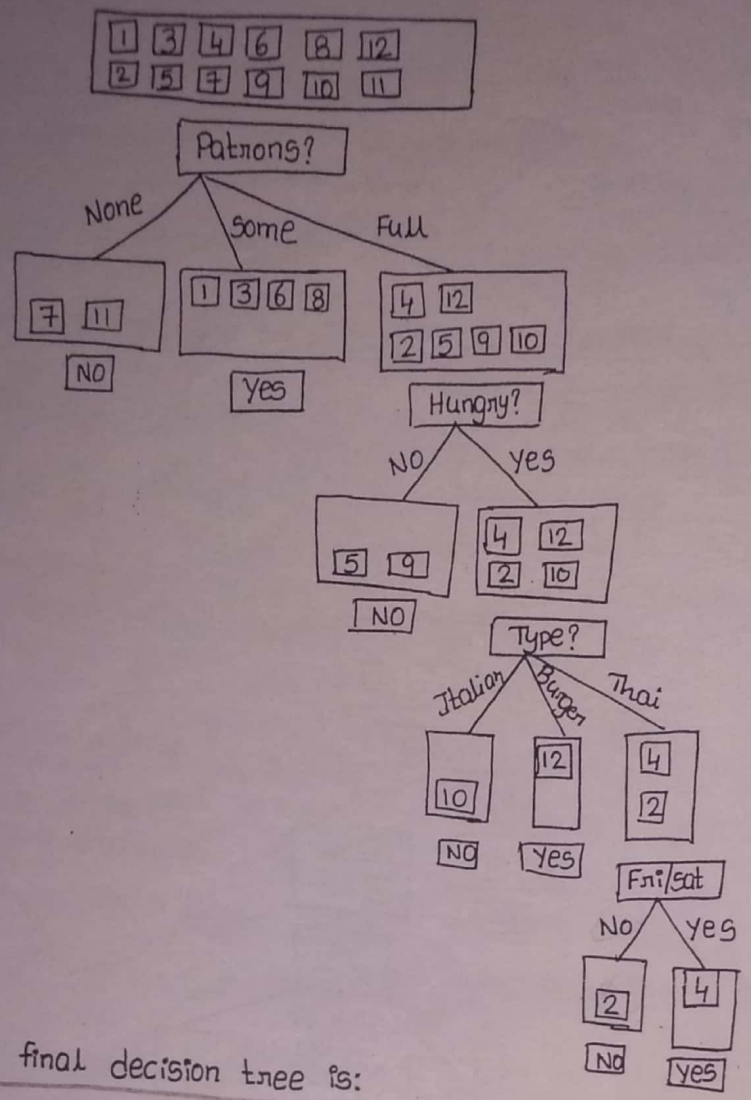
↳ When Hungry = NO, we are left with negative examples, we can answer No.
↳ When Hungry = YES, we are left with mixed set of examples, so it needs further classification.
↳ The next step will be performed on the reamining samples of Hungry=yes

Step4:- Select the next best attribute. "Type" is the next best choice.

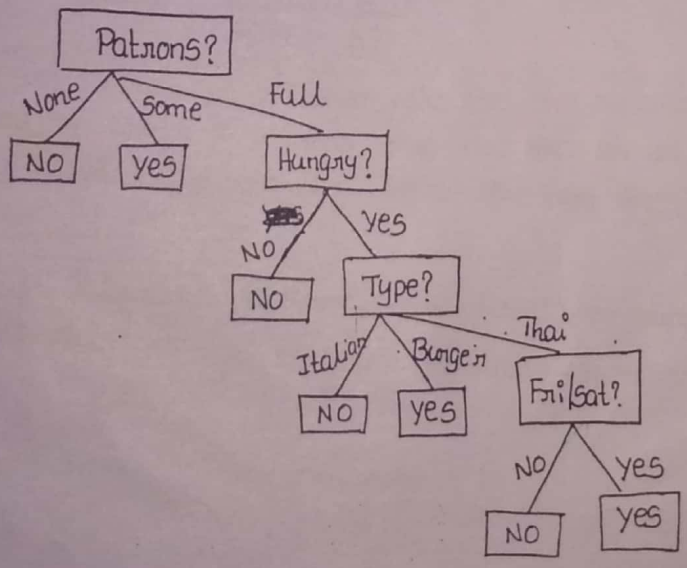


↳ When type = Italian, we are left with negative example, we can answer NO.
↳ When type = Burger, we are left with positive examples, we can answer yes.
↳ When type = Thai, we are left with mixed set of examples, which needs further classification.

Step5:- Now we have to classify the remaining examples of "Type = Thai".
Select the best attribute. Now it is "Fri/Sat".



⇨ The final decision tree is:



⇒ The DECISION-TREE-LEARNING algorithm is shown below.

```
function DECISION-TREE-LEARNING(examples, attributes, default) returns a decision tree.
{
  inputs: examples, set of examples
          attributes, set of attributes
          default, default value for the goal predicate.
  if examples is empty then return default
  else if all examples have the same classification then return the classification.
  else if attributes is empty then return MAJORITY_VALUE(examples)
  else
  {
     best ← CHOOSE-ATTRIBUTE(attributes, examples)
     tree ← a new decision tree with root test best
     m ← MAJORITY_VALUES(examples)
     for each value v_i of best do
     {
        examples_i ← {elements of examples with best = v_i}
        subtree ← DECISION-TREE-LEARNING(examples_i, attributes - best, m)
        add a branch to tree with label v_i and subtree
     }
  }
  return tree
}
```

CHOOSING ATTRIBUTE SETS:-

⇒ The scheme used in decision tree learning for selecting attribute is designed to minimize the depth of the final tree.

↳ The idea is to pick the attribute that provides an exact classification of the examples.

↳ A perfect attribute divides the examples into sets that are all positive or all negative.

↳ A really useless attribute leaves the example sets with the same proportion of positive and negative examples as the original set.

↳ All we need is a measure of "fairly good" and "really useless" attributes

⇒ The measure should have its maximum value when the attribute is perfect and its minimum value when the attribute is useless.

↳ One suitable measure is the expected amount of "information" provided by the attribute.

↳ Information theory measures information content in bits.

↳ If possible answers v_i have probabilities $P(v_i)$ then the information content 'I' of the actual answer is given by

$$I(P(v_1), P(v_2), \ldots P(v_n)) = \sum_{i=1}^{n} -P(v_i)\log_2(v_i)$$

⇒ For a training set containing 'p' positive examples and 'n' negative examples,

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = \frac{-p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

↳ A chosen attribute 'A' divides the training set E into subsets $E_1, E_2, \ldots E_v$ according to their values for A, where 'A' has 'v' distinct values.

$$\text{remainder}(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} \cdot I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

∴ Information Gain:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{remainder}(A)$$

⇒ Choose the attribute with largest Information gain.

Example:-

⇒ For a training set with 6 positive examples and 6 negative examples

i.e. $p=n=6$

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = I\left(\frac{6}{12}, \frac{6}{12}\right) = \frac{-6}{12}\log_2\frac{6}{12} - \frac{6}{12}\log_2\frac{6}{12}$$

$$= 1 \text{ bit}$$

⇒ Calculate the information gain for the attributes "Patrons" and "type" (9)

$IG(\text{Patrons}) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{remainder(Patrons)}$

$= I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} \cdot I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$

$= 1 - \left[\frac{0+2}{12} \cdot I\left(\frac{0}{0+2}, \frac{2}{0+2}\right) + \frac{4+0}{12} \cdot I\left(\frac{4}{4+0}, \frac{0}{4+0}\right) + \frac{2+4}{12} I\left(\frac{2}{2+4}, \frac{4}{2+4}\right)\right]$

$= 1 - \left[\frac{2}{12} \cdot I\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} \cdot I\left(\frac{4}{4}, \frac{0}{4}\right) + \frac{6}{12} \cdot I\left(\frac{2}{6}, \frac{4}{6}\right)\right]$

$= 1 - \left[\frac{2}{12} \cdot I(0,1) + \frac{4}{12} \cdot I(1,0) + \frac{6}{12} \cdot I\left(\frac{1}{3}, \frac{2}{3}\right)\right]$

$= 0.541$ bits

$\therefore IG(\text{Patrons}) = 0.541$ bits $\rightarrow$ (1) ✓

$IG(\text{Type}) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{remainder(Type)}$

$= I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} \cdot I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$

$= 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right)\right]$

$= 0$

$\therefore IG(\text{Type}) = 0$ bits $\rightarrow$ (2) ✓

⇒ From equations (1) and (2) patrons has the highest information gain value.

∴ Patrons is chosen as the best attribute.

Assessing the performance of the learning algorithm:-

⇒ A learning algorithm is good if it produces hypotheses that do a good job of predicting the classification of unseen examples.

↳ We can assess the quality of a hypotheses by checking its predictions against the correct classification once we know it.

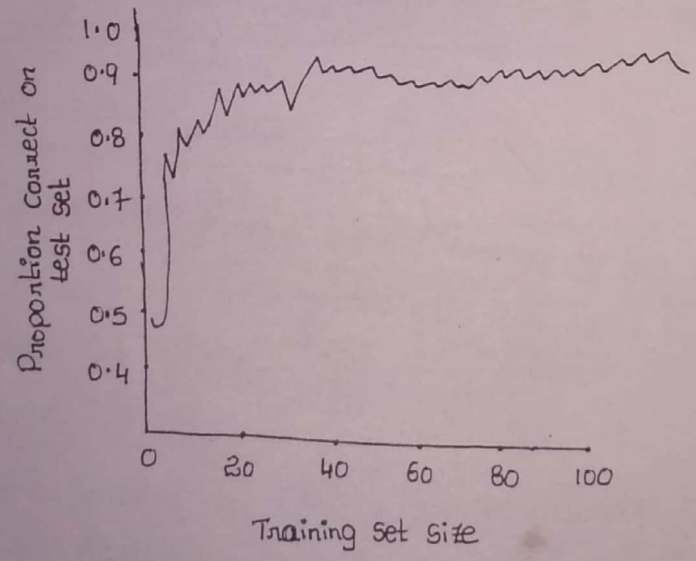
⇒ Procedure:

1. Collect a large set of examples
2. Divide it into two disjoint sets: the training set and the test set.
3. Apply the learning algorithm to the training set, generating a hypothesis 'h'
4. Measure the percentage of examples in the test set that are correctly classified by 'h'.
5. Repeat steps 1 to 4 for different sizes of training sets.

↳ The result of this procedure is a set of data that can be processed to give the average prediction quality as a function of the size of the training set.

This function can be plotted on a graph, giving what is called the "learning curve" for the algorithm on the particular domain.

↳ The learning curve for DECISION-TREE-LEARNING with the restaurant examples is shown below:-



↳ As the learning set grows, the prediction quality increases.

Broadening the applicability of decision trees:-

In order to extend decision tree induction to a wide variety of problems, a number of issues must be addressed.

1. Missing data:-

In many domains, not all the attributes values will be known for every example. The values might have gone unrecorded or they might be too expensive to obtain.

This gives rise to the following problem:

"How should one modify the information gain formula when some examples have unknown values for the attributes?".

2. Multi-valued attributes:- age = minor/youth/middle age/senior

When an attribute has many possible values, the information gain measure gives an inappropriate indication of the attribute's usefulness.

3. Continuous and Integer-valued input attributes:-

Continuous or integer-valued attributes such as Height and Weight have an infinite set of possible values.

Rather than generating infinitely many branches, decision-tree learning algorithm typically find a split-point that gives highest information gain.

Efficient dynamic programming methods exists for finding good split point, but it is still more expensive.

4. Continuous-valued output attributes:-

If we are trying to predict a continuous value rather than a discrete classification, we need a regression tree.

Such a tree has at each leaf a linear function of some subset of numerical attributes rather than a single value.

It is difficult to construct regression trees.