CHAPTER 3

REASONING UNDER UNCERTAINITY

CHAPTER OUTLINE

3.0 REASONING UNDER UNCERTAINITY

3.1 CONDITIONAL PROBABILITY

3.2 PROBABILISTIC INFERENCE WITH EXAMPLE BAYES' THEOREM/RULE

3.3 BAYES NETWORKS / BAYES' BELIEF NETWORKS

3.4 TYPES OF INFERENCE IN BAYES NETWORKS

3.5 UNCERTAIN EVIDENCE

3.6 D-SEPARATION

3.7 PROBABILISTIC INFERENCE IN POLYTREES

3.8 HOW REASONING CAN BE DONE ON STATES AND ACTIONS

3.9 DIFFICULTIES IN REASONING WITH UNCERTAIN INFORMATION

3.10 GENERATING PLANS

conditional Probability

$P(A|B)$ → Probability of a given B

(or)

$P(B|A)$ → Probability of B is given A

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

$$\frac{P(A|B)}{P(B|A)} = \frac{P(A)}{P(B)}$$

$P(A \cap B) = P(A) \cdot P(B)$ → for exclusive

3.0 REASONING UNDER UNCERTAINITY

Decision-making under uncertainty requires the understanding of the underlying uncertainties and assumptions within the probabilistic models or the data.

Till now, we have learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A!B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

So to represent uncertain knowledge, where we are not sure about the predicates, we need uncertain reasoning or probabilistic reasoning.

Following are some leading causes of uncertainty to occur in the real world.

- Information occurred from unreliable sources.
- Experimental Errors.
- Equipment fault.
- Temperature variation.
- Climate change.

In this chapter, we use Bayes' Theorem and Bayes' Networks to deal with uncertain information.

Before getting into the chapter, lets understand the few rules of probability / axioms of probability.

AXIOM-1

Probability : Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

$0 \le P(A) \le 1$, where P(A) is tge probability of an event A.
$P(A) = 0$, indicates total uncertainty in an event A.
$P(A) = 1$, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

$$\text{Probability of occurance} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

- $P(\neg A)$ = probability of a not happening event.
- $P(\neg A) + P(A) = 1$.

Event : Each possible outcome of a variable is called an event.

AXIOM-2

Sample Space : The collection of all possible events is called sample space. The Probability for entire sample space is 1. That is $P(S) = 1$.

AXIOM-3 :

Mutually Exclusive Event : These Mutually exclusive events mean that such events cannot occur together or in other words, they don't have common values or we can say their intersection is zero/null. We can also represent such events as follows :

$$P(A \cap B) = 0$$

Therefore, the probability of mutually exclusive events can be written as :

$$P(A \cup B) = P(A) + P(B) \text{ for mutually exclusive events}$$

Random Variables : Random variables are used to represent the events and objects in the real world.

Prior Probability : The prior probability of an event is probability computed before observing new information.

Posterior Probability : The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Axioms of Probability :

For any propositions A, B :

1. $0 \le P(A) \le 1$
2. P(True) = 1 (hence P(False) = 0)
3. $P(A \vee B) = P(A) - P(A \wedge B)$
4. Alternatively, if A and B are mutually exclusive ($A \wedge B = F$) then : $P(A \vee B) = P(A) + P(B)$

The axioms of propability limit the calss of functions that can be considered probability functions.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

3.1 / CONDITIONAL PROBABILITY

Conditional probability, in probability theory, is defined as the measure of the likelihood of an event occurring, assuming that another event or outcome has previously occurred. It is expressed as the multiplication of the probability of the previously occurred event with the probability of the conditional event that has occurred in succession.

So, if we have events **A** and **B** where $P(B) > 0$, we calculate the conditional probability of **A** when **B** has already occurred, $P(A \mid B)$ as :

$$P(A \mid B) = P(A \cap B)P(B)$$

While computing the conditional probability, it is assumed that we are aware of the outcome of event B. This is especially useful since the information of an experiment's outcome is often unknown.

Example :

- We have an event A where we assume that an individual who has applied to a university will be accepted. The probability of them getting accepted is 70%.
- We have another event B where there is a 50% chance that accepted students will be assigned dormitory housing.
- Hence, we calculate the conditional probability as,

  Probability (Students Accepted and Dormitory Assigned) = P (Dormitory Assigned | Students Accepted) × P(Students Accepted)

  = (0.50)*(0.70) = 0.35

Conditional probability finds extensive use in different fields such as insurance and calculus. It is also applicable in politics. Let's assume there is an expected re-election of a president. The results will depend on the preferences of those eligible to vote and the probability of the outcome of television advertising campaigns.

In another example, let's assume that the probability of rain in your area is 40% as specified by the weather. However, this outcome is largely dependent on :

- Whether there are clouds forming in your area.
- Whether there is the possibility of a cold front arriving in your area.
- Whether the clouds are being pushed away by another front.

The conditional probability will depend on each of the above events.

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

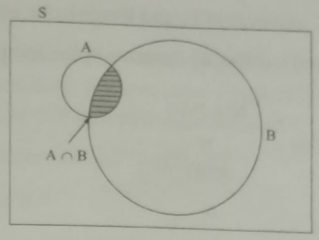


FIG. 3.1. Conditional Probability

As depicted by the above diagram, sample space is given by S, and there are two events A and B. In a situation where event B has already occurred, then our sample space S naturally gets reduced to B because now the chances of occurrence of an event will lie inside B.

As we have to figure out the chances of occurrence of event A, only a portion common to both A and B is enough to represent the probability of occurrence of A, when B has already occurred. The common portion of the events is depicted by the intersection of both the events A and B, i.e., $A \cap B$.

1. **Marginal Probability :** Marginal probability is the probability of an event happening, such as (p(A)), and it can be mentioned as an unconditional probability. It does not depend on the occurrence of another event. For example, the likelihood that a card is drawn from a deck of cards is black (P(black) = 0.5, which is independent event since the outcome of another event does not condition the result of one event.
2. **Joint Probability :** A joint probability is the probability of event A and event B happening, P(A and B). It is the likelihood of the intersection of two or more events. The probability of the intersection of A and B is written as $P(A \cap B)$. For example, the likelihood that a card is black and seven is equal to P(Black and Seven) = 2/52 = 1/26. (There are two Black-7 in a deck of 52: the 7 of clubs and the 4 of spades).

When the intersection of two events happens, then the formula for conditional probability for the occurrence of two events is given by :

$$P(A \mid B) = N(A \cap B)/N(B)$$

or

$$P(B \mid A) = N(A \cap B)/N(A)$$

where, $P(A \mid B)$ represents the probability of occurrence of A given B has occurred.

$N(A \cap B)$ is the number of elements common to both A and B.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

$N(B)$ is the number of elements in B, and it cannot be equal to zero.

Let N represent the total number of elements in the sample space.

$$P(A \mid B) = \frac{\frac{N(A \cap B)}{N}}{\frac{N(B)}{N}}$$

Therefore, $N(A \cap B)/N$ can be written as $P(A \cap B)$ and $N(B)/N$ as $P(B)$.

$\Rightarrow P(A \mid B) = P(A \cap B)/P(B)$

Therefore, $P(A \cap B) = P(B)\, P(A \mid B)$ if $P(B) \neq 0$

$= P(A)\, P(B \mid A)$ if $P(A) \neq 0$

Similarly, the probability of occurrence of B when A has already occurred is given by,

$$P(B \mid A) = P(B \cap A)/P(A)$$

EXAMPLE-1

Let us understand the Conditional Probability with an example.

Let there are 100 (47 boys, 53 girls) students in a classroom, the given table describes the number of students (boys and girls) who pass or fail in the exam.

	Pass	Fail	Total
Boys	15	32	47
Girls	29	24	53
Total	44	56	100

Let, A represent passed in the exam.

and B represent student is a Boy.

- Probability of a Boy to pass the exam :

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{15}{47}$$

- Probability that the student passes is a boy.

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} = \frac{15}{44}$$

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

EXAMPLE-2

In a group of 100 computer buyers, 40 bought CPU, 30 purchased monitor, and 20 purchased CPU and monitors. If a computer buyer chose at random and bought a CPU, what is the probability they also bought a Monitor?

As per the first event, 40 out of 100 bought CPU,

So, $P(A) = 40\%$ or 0.4

Now, according to the question, 20 buyers purchased both CPU and monitors. So, this is the intersection of the happening of two events. Hence,

$P(A \cap B) = 20\%$ or 0.2

By the formula of conditional probability we know;

$P(B \mid A) = P(A \cap B)/P(B)$

$P(B \mid A) = 0.2/0.4 = 2/4 = ½ = 0.5$

The probability that a buyer bought a monitor, given that they purchased a CPU, is 50%.

3.2 PROBABILISTIC INFERENCE WITH EXAMPLE

Probabilistic inference is the task of deriving the probability of one or more random variables taking a specific value or set of values.

Probabilistic reasoning/inference is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Need of Probabilistic Reasoning in AI :

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge :

- Bayes' rule
- Bayesian Statistics.

Baye's Theorem/Rule : Bayes' Theorem is an extension of the Conditional Probability. *Bayes' theorem defines the probability of occurrence of an event associated with any condition. It is considered for the case of conditional probability. Also, this is known as the formula for the likelihood of "causes".*

This simple equation underlies most modern AI systems for probabilistic inference.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

- **P(A | B)** is known as **posterior**, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.
- **P(B | A)** is called the **likelihood**, in which we consider that hypothesis is true, then we calculate the probability of evidence.
- **P(A)** is called the **prior probability**, probability of hypothesis before considering the evidence
- **P(B)** is called **marginal probability**, pure probability of an evidence.]

In general we can write, $P(B) = P(A) \times P(B \mid A_i)$, hence the Bayes' rule can be written as :

$$P(A_i \mid B) = \frac{P(A_i) \times P(B \mid A_i)}{\sum_{i=1}^{k} P(A_i) \times P(B \mid A_i)}$$

where, $A_1, A_2, A_3, \ldots\ldots, A_n$ is a set of mutually exclusive and exhaustive events.

Applying Bayes' Rule : Bayes' rule allows us to compute the single term $P(B \mid A)$ in terms of $P(A \mid B)$, $P(B)$, and $P(A)$. This is very useful in cases where we have a good probability of these three terms and want to determine the fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayes' rule becomes :

$$P(\text{cause} \mid \text{effect}) = \frac{P(\text{effect} \mid \text{cause})\, P(\text{cause})}{P(\text{effect})}$$

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

EXAMPLE-1

What is the probability that a patient has diseases meningitis with a stiff neck?

Given Data :

A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows :

- The known probability that a patient has meningitis disease is 1/30,000.
- The known probability that a patient has a stiff neck is 2%.

Let a be the proposition that patient has stiff neck and b be the proposition that patient has meningitis. , so we can calculate the following as :

$P(a \mid b) = 0.8$

$P(b) = 1/30000$

$P(a) = .02$

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)} = \frac{0.8 \times \left(\frac{1}{30000}\right)}{0.02} = 0.001333333.$$

Hence, we can assume that 1 patient out of 750 patients has meningitis disease with a stiff neck.

Application of Bayes' Theorem in Artificial Intelligence : Following are some applications of Bayes' theorem :

- To calculate the next step of the robot when the already executed step is given.
- Weather forecasting.
- To solve the Monty Hall problem.

3.3 BAYES NETWORKS / BAYES' BELIEF NETWORKS

A Bayesian network (also known as a Bayes network, Bayes net, belief network, or decision network) is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG).

It is also called a Bayes network, belief network, decision network, or Bayesian model. Bayesian networks are probabilistic, because these networks are built from a probability distribution, and also use probability theory for prediction and anomaly detection.

Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

Bayesian Network can be used for building models from data and experts opinions and it consists of two parts :

- Directed Acyclic Graph.
- Table of conditional probabilities.

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an Influence diagram.

A Bayesian Network Graph is Made up of Nodes and Arcs (Directed Links), Where

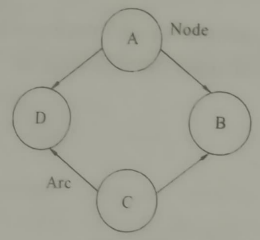


FIG 3.2 :

- Each node corresponds to the random variables, and a variable can be continuous or discrete.
- Arc or directed arrows represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.

These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other

- In the above diagram, **A**, **B**, **C**, and **D** are random variables represented by the nodes of the network graph.
- If we are considering node B, which is connected with node A by a directed arrow, then node A is called the **parent** of Node **B**.
- Node C is **independent** of node A.

The Bayesian network has mainly two components :

- Causal Component
- Actual numbers.

Each node in the Bayesian network has condition probability distribution P(Xi |Parent(Xi)), which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution :

1. Joint Probability Distribution :

If we have variables $x_1, x_2, x_3, \ldots, x_n$, then the probabilities of a different combination of $x_1, x_2, x_3, \ldots, x_n$, are known as Joint probability distribution.

$P[x_1, x_2, x_3, \ldots, x_n]$, it can be written as the following way in terms of the joint probability distribution.

$$= P[x_1 \mid x_2, x_3, \ldots, x_n]\, P[x_2, x_3, \ldots, x_n]$$

$$= P[x_1 \mid x_2, x_3, \ldots, x_n]\, P[x_2 \mid x_3, \ldots, x_n] \ldots P[x_{n-1} \mid x_n] P[x_n].$$

In general for each variable Xi, we can write the equation as :

$$\mathbf{P(X_i \mid X_{i-1}, \ldots, X_1) = P(X_i \mid Parents(Xi))}$$

Example of Bayes' Network : In this example, let us imagine that we are given the task of modeling a student's marks (m) for an exam he has just given. From the given Bayesian Network Graph below, we see that the marks depend upon two other variables. They are :

- **Exam Level (e) :** This discrete variable denotes the difficulty of the exam and has two values (0 for easy and 1 for difficult).
- **IQ Level (i) :** This represents the Intelligence Quotient level of the student and is also discrete in nature having two values (0 for low and 1 for high).

Additionally, the IQ level of the student also leads us to another variable, which is the Aptitude Score of the student (s). Now, with marks the student has scored, he can secure admission to a particular university. The probability distribution for getting admitted (a) to a university is also given below.

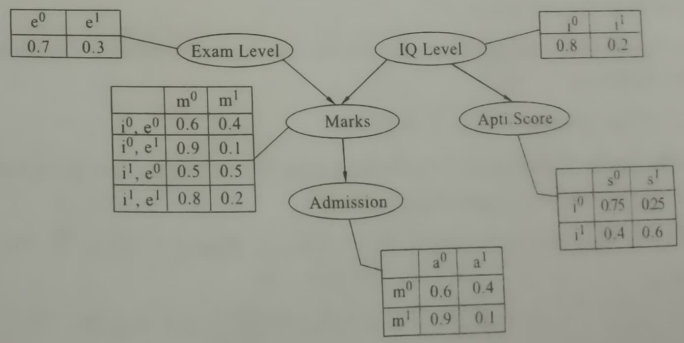


FIG 3.3 :

Let us first list all the possible events that are occurring in the above-given table

(i) Exam Level (e) (ii) IQ Level (i)

(iii) Aptitude Score (s) (iv) Marks (m)

(v) Admission (a)

These five variables are represented in the form of a Directed Acyclic Graph (DAG) in a Bayesian Network format with their Conditional Probability tables. Now, to calculate the Joint Probability Distribution of the 5 variables the formula is given by,

$$\mathbf{P[a, m, i, e, s] = P(a \mid m).P(m \mid i, e).P(i).P(e).P(s \mid i)}$$

From the above formula,

- P(a | m) denotes the conditional probability of the student getting admission based on the marks he has scored in the examination.
- P(m | i, e) represents the marks that the student will score given his IQ level and difficulty of the Exam Level.
- P(i) and P(e) represent the probability of the IQ Level and the Exam Level.
- P(s | i) is the conditional probability of the student's Aptitude Score, given his IQ Level.

Calculation of Joint Probability Distribution : Let us now calculate the JPD for two cases.

CASE-1

Calculate the probability that in spite of the exam level being difficult, the student having a low IQ level and a low Aptitude Score, manages to pass the exam and secure admission to the university.

From the above word problem statement, the Joint Probability Distribution can be written as below,

$$P[a=1, m=1, i=0, e=1, s=0]$$

From the above Conditional Probability tables, the values for the given conditions are fed to the formula and is calculated as below.

$$\begin{aligned} P[a=1, m=1, i=0, e=0, s=0] &= P(a=1 \mid m=1) \cdot P(m=1 \mid i=0, e=1) \cdot P(i=0) \cdot P(e=1) \\ &\quad P(s=0 \mid i=0) \\ &= 0.1 \times 0.1 \times 0.8 \times 0.3 \times 0.75 \\ &= 0.0018 \end{aligned}$$

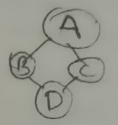
WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

CASE - 2

In another case, calculate the probability that the student has a High IQ level and Aptitude Score, the exam being easy yet fails to pass and does not secure admission to the university.

The formula for the JPD is given by

$$\mathbf{P[a=0, m=0, i=1, e=0, s=1]}$$



Thus,

$$\begin{aligned} P[a=0, m=0, i=1, e=0, s=1] &= P(a=0 \mid m=0) \cdot P(m=0 \mid i=1, e=0) \cdot P(i=1) \\ &\quad P(e=0) \cdot P(s=1 \mid i=1) \\ &= 0.6 \times 0.5 \times 0.2 \times 0.7 \times 0.6 \\ &= 0.0252 \end{aligned}$$

Hence, in this way, we can make use of Bayesian Networks and Probability tables to calculate the probability for various possible events that occur.

3.4 TYPES OF INFERENCE IN BAYES NETWORKS

The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of query nodes, given values for some evidence nodes. This task is called belief updating or probabilistic inference. Inference in Bayesian networks is very flexible, as evidence can be entered about any node while beliefs in any other nodes are updated.

There are two types of Bayes' Inference algorithms, they are :

1. Casual Inference (Top-Down Inference).
2. Diagnostic Inference (Bottom-Up Inference).

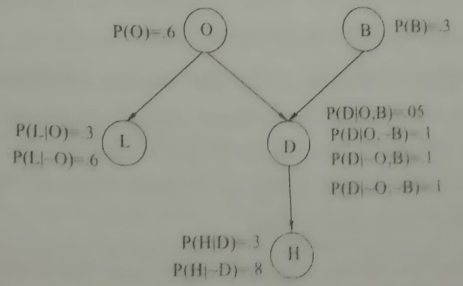


FIG 3.4 :

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

1. **Casual (Top-Down) Inference :** The algorithm for computing a conditional probabil... from a Bayesian Net is complicated, but it is easy when the query involves nodes that... are directly connected to each other. In this section we consider problems of the form... P(Q | E) and there is a link in the Bayesian Net from evidence E to query Q. We call this... case causal inference because we are reasoning in the same direction as the causal arc.

Consider our "home domain" and the problem of computing P(D | B), i.e., what is the probability that my dog is outside when it has bowel troubles? We can solve this problem as follows :

Apply the Product Rule and Marginalization

$$\begin{aligned} P(D \mid B) &= P(D,B)/P(B) && \text{by the Product Rule} \\ &= (P(D,B,O) + P(D,B,\sim O))/P(B) && \text{by marginalizing P(D,B)} \\ &= P(D,B,O)/P(B) + P(D,B,\sim O)/P(B) \\ &= P(D,O \mid B) + P(D,\sim O \mid B) \end{aligned}$$

Apply the conditionalized version of the chain rule, i.e., $P(A,B \mid C) = P(A \mid B,C)P(B \mid C)$, to obtain

$$P(D \mid B) = P(D \mid O,B)P(O \mid B) + P(D \mid \sim O,B)P(\sim O \mid B)$$

Since O and B are independent by the network, we know $P(O \mid B)=P(O)$ and $P(\sim O \mid B)=P(\sim O)$. This means we now have,

$$\begin{aligned} P(D \mid B) &= P(D \mid O,B)P(O) + P(D \mid \sim O,B)P(\sim O) \\ &= (.05)(.6) + (.1)(1 - .6) \\ &= 0.07 \end{aligned}$$

In general, for this case we first rewrite the goal conditional probability of query variable Q in terms of Q and all of its parents (that are not evidence) given the evidence. Second, re-express each joint probability back to the probability of Q given all of its parents. Third, look up in the Bayesian Net the required values.

2. **Diagnostic (Bottom-Up) Inference :** The last section considered simple causal inference. In this section we consider the simplest case of diagnostic inference. That is, the problem is to compute P(Q | E) and in the Bayesian Net there is an arc from query Q to evidence E. So, we are using a symptom to infer a cause. This is analogous to using the abduction rule of inference in FOL.

For Example : Consider the "home domain" again and the problem of computing $P(\sim B \mid \sim D)$. That is, if the dog is not outside, what is the probability that the dog has bowel troubles?

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

First, use Bayes's Rule :

$$P(\sim B \mid \sim D) = P(\sim D \mid \sim B)P(\sim B)/P(\sim D)$$

We can look up in the Bayesian Net the value of $P(\sim B) = 1 - .3 = .7$. Next, compute $P(\sim D \mid \sim B)$ using the causal inference method described above. Here we get,

$$\begin{aligned} P(\sim L \mid \sim B) &= P(\sim D,O \mid \sim B) + P(\sim D,\sim O \mid \sim B) \\ &= P(\sim D \mid O,\sim B)P(O \mid \sim B) + P(\sim D \mid \sim O,\sim B)P(\sim O \mid \sim B) \\ &= P(\sim D \mid O,\sim B)P(O) + P(\sim D \mid \sim O,\sim B)P(\sim O) \\ &= (.9)(.6) + (.8)(.4) \\ &= 0.86 \end{aligned}$$

So, $P(\sim B \mid \sim D) = (.86)(.7)/P(\sim D) = .602/P(\sim D)$.

To avoid computing the prior probability, $P(\sim D)$, of symptom D, we can use normalization, which requires computing $P(B \mid \sim D)$. That is, $P(B \mid \sim D) = P(\sim D \mid B)P(B)/P(\sim D)$ by Bayes's Rule, and $P(B)=.3$ from the Bayesian Net. Now compute $P(\sim D \mid B)$ as follows :

$$\begin{aligned} P(\sim D \mid B) &= P(\sim D,O \mid B) + P(\sim D,\sim O \mid B) \\ &= P(\sim D \mid O,B)P(O \mid B) + P(\sim D \mid \sim O,B)P(\sim O \mid B) \\ &= P(\sim D \mid O,B)P(O) + P(\sim D \mid \sim O,B)P(\sim O) \\ &= (.95)(.6) + (.9)(.4) \\ &= 0.93 \end{aligned}$$

So, $P(B \mid \sim D) = (.93)(.3)/P(\sim D) = .279/P(\sim D)$. Since $P(\sim B \mid \sim D) + P(B \mid \sim D) = 1$, we have $.602/P(\sim D) + .279/P(\sim D) = 1$, and so $P(\sim D) = .881$. Thus, $P(\sim B \mid \sim D) = .602/.881 = .683$.

In general, diagnostic inference problems are solved by converting them to causal inference problems using Bayes's Rule, and then proceeding as before.

3.5 / UNCERTAIN EVIDENCE

We consider the problem of performing Bayesian inference in probabilistic models where observations are accompanied by uncertainty, referred to as "uncertain evidence." We must be certain about the truth or falsity of the propositions they represent.

Each uncertain evidence node should have a child node, about which we can certain.

Ex : Suppose the Robot is not certain that its arm did not move

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

Causes for Uncertainty of Evidence are :

1. Observational Error
2. Unable to observe the precise state the world is in.

3.5.1 TYPES OF UNCERTAIN EVIDENCES

There are Two Types of Uncertain Evidences. They are :

1. **Virtual Evidence :** Virtual Evidence (VE), proposed by Pearl, provides a convenient way of incorporating evidence with uncertainty. A VE on a variable V is represented by a likelihood $\dot{E}_V$ where each $\dot{E}_V(v)$ is a real number in [0, 1]. Pearl's method extends the given BN by adding a binary virtual node which is a child of V. In our example, we add a node Sobs and a directed edge from S toward S_{obs}.

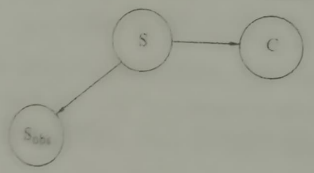


FIG 3.5 : Bayesian Network graph encoding a virtual evidence on S

2. **Soft Evidence** Soft evidence (SE), named by Valtorta, can be interpreted as evidence of uncertainty. SE is given as a probability distribution of one or several variables R(Y), Y ⊂ X, where X is the set of variables. Therefore, there is uncertainty about the specific state Y is in but we are sure of the probability distribution R(Y). Since R(Y) is a certain observation, this distribution should be preserved by updating belief. This is the main difference with virtual evidence for which this is not required.

3.6 D-SEPARATION

If G is a directed graph in which X, Y and Z are disjoint sets of vertices, then X and Y are d-connected by Z in G if and only if there exists an undirected path U between some vertex in X and some vertex in Y such that for every collider C on U, either C or a descendent of C is in Z, and no non-collider on U is in Z.

X and Y are d-separated by Z in G if and only if they are not d-connected by Z in G.

Evidence Nodes E d-separates V_i and V_j if for every undirected path in the Bayes Network between V_i and V_j, there is some node, V_b, on the path having one of the following three properties.

1. V_b is in E, and both arcs on the path lead out of V_b

WARNING XEROX PHOTOCOPYING OF THIS BOOK IS ILLEGAL

2. V_b is in E, and one arc on the path leads into V_b and one arc lead out
3. Neither V_b not any descendant of V_b is in E, and both arcs on the path lead into V_b

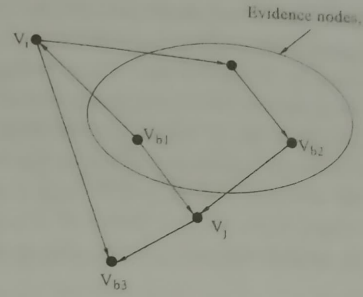


FIG 3.6 : d-Separation

V_i is independent of V_j given the evidence nodes because all the paths between them are blocked. The blocking nodes are :

(a) V_{b1} is an evidence node, and both arcs are lead out of V_{b1}.

(b) V_{b2} is an evidence node, and one arc lead into V_{b2} and one arc lead out.

(c) V_{b3} is not an evidence node, nor are any of its descendants, and both arcs lead into V_{b3}.

The "d" in d-separation and d-connection stands for dependence. Thus if two variables are d-separated relative to a set of variables Z in a directed graph, then they are independent conditional on Z in all probability distributions such a graph can represent. Roughly, two variables X and Y are independent conditional on Z if knowledge about X gives you no extra information about Y once you have knowledge of Z. In other words, once you know Z, X adds nothing to what you know about Y.

A path is active if it carries information, or dependence. Two variables X and Y might be connected by lots of paths in a graph, where all, some, or none of the paths are active. X and Y are d-connected, however, if there is any active path between them. So X and Y are d-separated if all the paths that connect them are inactive, or, equivalently, if no path between them is active.

A path is active when every vertex on the path is active. Paths, and vertices on these paths, are active or inactive relative to a set of other vertices Z. First lets examine when things are active or inactive relative to an empty Z. To make matters concrete, consider all of the possible undirected paths between a pair of variables A and B that go through a third variable C:

WARNING

1. $A \rightarrow C \rightarrow B$
2. $A \leftarrow C \leftarrow B$
3. $A \leftarrow C \rightarrow B$
4. $A \rightarrow C \leftarrow B$

The first is a directed path from A to B through C, the second a directed path from B to A through C, and the third a pair of directed paths from C to A and from C to B. If we interpret these paths causally, in the first case A is an indirect cause of B, in the second B is an indirect cause of A, and in the third C is a common cause of A and B. All three of these causal situations give rise to association, or dependence, between A and B, and all three of these undirected paths are active in the theory of d-separation. If we interpret the fourth case causally, then A and B have a common effect in C, but no causal connection between them. In the theory of d-separation, the fourth path is inactive. Thus, when the conditioning set is empty, only paths that correspond to causal connection are active.

In the first three, C is a non-collider on the path, and in the fourth C is a collider. When the conditioning set is empty, non-colliders are active. Intuitively, non-colliders transmit information (dependence).

3.7 PROBABILISTIC INFERENCE IN POLYTREES

- A polytree is a network in which there exists a node B such that some of the nodes are connected to it only through its parents.
- The nodes which are not connected by its parents are connected by its immediate successors.
- The nodes connected by parents are said to be above B.
- The nodes connected by its children are said to be below B.
- It has to be noted that in the polytree this B node is the only path between the above nodes and below nodes.
- The polytree network is shown in the Fig. 3.7.

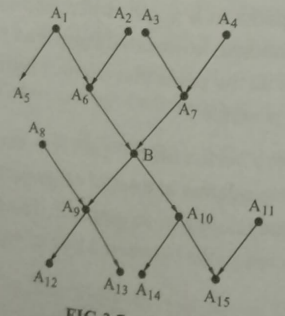


FIG 3.7 : Polytree

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

Evidence Above : If all evidence nodes are above Q. We calculate P(B | A5, A4) as an example. Here, bottom up recursive algorithm is used.

It is used to know the probability of every ancestor of B, if the evidence is given and till we reach the evidence or the evidence is below the ancestor.

The algorithm is as follows :

1. The parents of node B are taken,

$$P(B \mid A_5, A_5) = \sum_{A_6, A_7} P(B, A_6, A_7 \mid A_5, A_4)$$

(Here the summation is used to mention the four combinations, one is what is present i e., A_6, A_7, second one is substituting A6 for A6, the other is substituting $\neg A_7$ for A_7, and other is substituting $\neg A_6$. and – $\neg A_7$).

2. Conditional independence is used make the parents of Q as part of the evidence.

$$P(B, A_6, A_7 \mid A_5, A_4) = P(B, A_6, A_7, A_5, A_4) P(A_6, A_7 \mid A_5, A_4)$$

3. Substituting this in first step gives,

$$P(B \mid A_5, A_4) = \sum_{A_6, A_7} P(B, A_6, A_7, A_5, A_4) P(A_6, A_7 \mid A_5, A_4)$$

4. As the node is conditionally independent of non descendants if it parents are given,

$$P(B \mid A_5, A_4) = \sum_{A_6, A_7} P(B \mid A_6, A_7) P(A_6, A_7 \mid A_5, A_6)$$

5. By using D-separation we can split the parents,

$$P(B \mid A_5, A_4) = \sum_{A_6, A_7} P(B \mid A_6, A_7) P(A_6 \mid A_5, A_6) P(A_7 \mid A_5 \mid A_4)$$

6. As it is allowed by D-separation we ignore the evidence which is above one parent while we are calculating the probability of another parent.

$$P(Q \mid A_5, A_4) = \sum_{A_6, A_7} P(B \mid A_6, A_7) P(A_6 \mid A_5) P(A_7 \mid A_4)$$

In the above equation $P(B \mid A_6, A_7)$ is the probability, of the query node, where the value of its parents are given.

$P(A_6 \mid A_5)$ and $P(A_7 \mid A_4)$ are the probabilities of the parents if the part of the evidence above the parent is given.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

The recursive process of algorithm continues till we reach the node which has the evidence node as a parent.

In $P(A_7 | A_4)$ the A4 node is the evidence node and it is the parent of A_7.

Now the parents are involved.

Since,

$$P(A_7, A_3 | A_4) = P(A_7 | A_3, A_4)P(A_3 | A_4)$$

Then,

$$P(A_7, A_4) = \sum_{A_3} P(A_7 | A_3, A_4)P(A_3)$$

In $P(A_1 | A_5)$ calculation we have,

As in $P(A_1 | A_5)$ the evidence node is note above the query node this recursive procedure is terminated and as it is below evidence below method has to be used

$$P(A_1 | A_5) = \frac{P(A_5 | A_1)P(A_1)}{P(A_5)}$$

Now we can calculate $P(A_6 | A_5)$ and then $P(Q | A_5, A_4)$.

In the similar way, we can also use Evidence below and Evidence Above and below to inference the polytrees.

3.8 HOW REASONING CAN BE DONE ON STATES AND ACTIONS

- To investigate feature-based planning methods much more thoroughly, richer language to describe features and the constraints among them will be introduced.
- Generally, a goal condition can be described by any wff (Well Formed Formula) in the predicate calculus, and we can determine if a goal is satisfied in a world state described by formulas by attempting to prove the goal wff from those formulas.
- A predicate calculus formalization of states, actions, and the effects of actions on states
- Our knowledge about states and actions as formulas in the first-order predicate calculus.
- Then use a deduction system to ask questions such as *"Does there exist a state to satisfy certain properties, and if so, how can the present state be transformed into that state by actions"*.
- The situation calculus was used in some early AI planning systems.
- However, the formalism remains important for exposing and helping to clarify conceptual problems.

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

- In order to describe states in the situation calculus, we ***reify*** states.
- States can be denoted by constant symbols (S0, S1, S2, ...), by variables, or by functional expressions.
- ***Fluents:*** the atomic wff can denote relations over states.

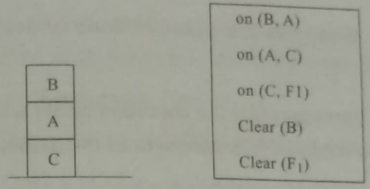


FIG. 3.8 : Block World Problem

- **First-Order Predicate Calculus :**

  $On(B,A) \wedge On(A,C) \wedge On(C, F1) \wedge Clear(B) \wedge .....$

  True Statement :

  $On(B,A,S0) \wedge On(A,C,S0) \wedge On(C,F1,S0) \wedge Clear(B,S0)$

  Preposition True of All States :

  $(\forall x,y,s)[On(x,y,s) \wedge \neg (y=F1) \supset Clear(y,s)]$ and $(\forall s)clear(F1, s)$

To Represent Actions and the Effects :

- ***Reify the Action :***
  - Actions can be denoted by constant symbols, by variables, or by functional expressions
  - Generally, we can represent a family of move actions by the schema, move(x,y,z), where x, y, and z are schema variables.
- Imagine a function constant, do, that denotes a function that maps actions and states into states.
- $Do(\alpha, \sigma)$ denotes a function that maps the state-action pair into the state obtained by performing the action denoted by α in the state denoted by σ.

Express the Effects of Actions by wffs :

There are two such wffs for each action-fluent pair.

For the pair {On, move}

$[On(x,y,s) \wedge Clear(x,s) \wedge Clear(z,s) \wedge (x \neq z) \supset On(x,z,do(move(x,y,z)s))]$

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

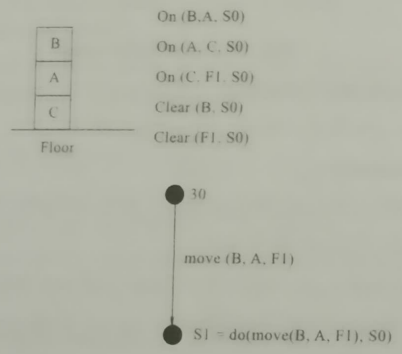
- We can also write the effect axioms for the (clear, move) pair

  $On(x,y,z) \wedge Clear(x,s) \wedge Clear(z,s) \wedge (x \neq z) \wedge (y \neq z) \supset Clear(y,do(move(x,y,z),s))$

  $On(x,y,z) \wedge Clear(x,s) \wedge Clear(z,s) \wedge (x \neq z) \wedge (z \neq F1) \supset \neg Clear(z,do(move(x,y,z),s))$

- The antecedents consist of two parts

  - One part expresses the preconditions under which the action can be executeed
  - The other part expresses the condition under which the action will have the effect expressed in the consequent of the axiom.



Inferred Using Effect Axioms :

On (B,F1,do (move (B,A,F1),S0))

¬ On (B,A,do(move(B,A,F1),S0))

Clear (A,do(move (B,A,F1), S0))

Inferred Using Frame Axioms :

On (A,C,do (move (B, A, F1),S0))

On (C,F1,do (move (B,A,F1), S0))

Clear (B,do (move (B,A,F1), S0))

True in all states

$(\forall s)$ Clear (F1, s)

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

Frame Axioms :

- Not all of the statements true about state do(move(B,A,F1),S0) can be inferred by the effects axioms
  - Before the move, such as that C was on the floor and that B was clear are also true of the state after the move
- In order to make inferences about these constancies, we need frame axioms for each action and for each fluent that doesn't change as a result of the action.
- The frame axioms for the pair, {(move, On)}

  $[On(x,y,s) \wedge (x \neq u) \supset On(x,y,do(move(u,y,z),s))$

  $(\neg On(x,y,s) \wedge [(x \neq u)(y \neq z)] \supset \neg On(x,y,do(move(u,y,z),s))$

- The frame axioms for the pair, {(move, clear)}

  $Clear(u,s) \wedge (u \neq z) \supset Clear(u,do(move(x, y,z),s))$

  $\neg Clear(u,s) \wedge (u \neq y) \supset \neg Clear(u,do(move(x,y,z),s))$

- Frame axioms are used to prove that a property of a state remains true if the state is changed by an action that doesnot affect that property
- ***Frame Problem :*** The various difficulties associated with dealing with fluent that are not affected by the actions.

Qualifications :

- The antecedent of the transition formula describing an action such as move gives the preconditions for a rather idealized case.
- To be more precise, adding other qualification such as $\neg Too\_heavy(x,s)$, $\neg Glued\_down(x,s)$, $\neg Armbroken(s)$, ...
- ***Qualification Problem :*** The difficulty of specifying all of the important qualifications.
- ***Ramification Problem:*** Keeping track of which derived formulas survive subsequent state transitions.

3.9 DIFFICULTIES IN REASONING WITH UNCERTAIN INFORMATION

- One of the main challenges for reasoning with uncertainty and time is how to represent and manipulate probabilistic and temporal knowledge in a computationally efficient and scalable way. There are many different formalisms and languages for expressing uncertainty and time, such as Bayesian networks, Markov decision processes, temporal logic, and fuzzy logic, but each of them has its own advantages and limitations.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

- Another challenge is how to deal with the trade-off between expressiveness and tractability, that is, how to balance the complexity and richness of the representation with the feasibility and speed of the inference and learning algorithms.
- A third challenge is how to handle the dynamic and interactive nature of the domain, where new information and feedback may become available, and where the system may need to adapt and update its beliefs and plans accordingly.

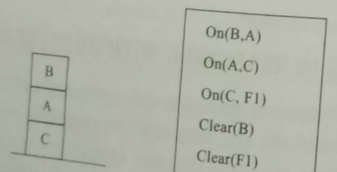
The following are the major difficulties experienced during reasoning under uncertainty :

- **Uncertain Data :** Missing data, unreliable, ambiguous, imprecise representation, inconsistent, subjective, derived from defaults, noisy.
- **Uncertain Knowledge :** Multiple causes lead to multiple effects, Incomplete knowledge of causality in the domain, laboratory data can be late, medical science can have an incomplete theory for some diseases.
- **Uncertain Knowledge Representation :** Restricted model of the real system, Limited expressiveness of the representation mechanism.
- **Inference Process :** Derived result is formally correct, but wrong in the real world, New conclusions are not well-founded, Incomplete, default reasoning methods.

3.10 GENERATING PLANS

To Generate a Plan that achieves some Goal $\gamma(s)$

- We attempt to prove $\exists(s)\ \gamma s)$
- And use the answer predicate to extract the state as a function of the nested actions that produce it.
- We want a plan that gets block B on the floor from the initial state, S_0 given in the below figure.



On(B,A)
On(A,C)
On(C, F1)
Clear(B)
Clear(F1)

FIG 3.9 : Block World Problem State

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

- Prove $\exists(s)$ ON (B,F1,s)
- We will prove by resolution refutation that the negation of $\exists(s)$ ON (B,F1,s), together with the formulas that describes S_0 and the effects of move are inconsistent.
  - Use an answer predicate to capture the substitutions made during the proof.
- On(A,C,S0)
- On(C,F1,S0)
- Clear(B,S0)
- Clear(F1,S0)
- $On(x,y,s) \wedge Clear(x,s) \wedge Clear(z,s) \wedge (x \neq z) \supset On(x,z,do(move(x,y,z),s))]$

Difficulties :

- If several actions are required to achieve a goal, the action functions would be nested.
- The proof effort is too large for even simple plan.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

CHAPTER 4

LEARNING FROM OBSERVATIONS

CHAPTER OUTLINE

4.1 LEARNING

4.2 FORMS OF LEARNING

4.3 INDUCTIVE LEARNING WITH EXAMPLE

4.4 DECISION TREE

4.5 ADVANTAGES OF THE DECISION TREE

4.6 HOW DECISION TREE IS USED TO DESCRIBE THE DOMAIN WITH EXAMPLE

4.7 PROCEDURE OF SELECTING ATTRIBUTE TESTS

4.8 DECISION TREE LEARNING ALGORITHM WITH EXAMPLE

4.9 OVER-FITTING PROBLEM

4.10 TECHNIQUE EMPLOYED IN HANDLING OVER-FITTING PROBLEM

4.11 ISSUES THAT ARE TO BE CONSIDERED FOR EXTENDING THE APPLICABILITY OF DECISION TREE

4.12 DEFINE FALSE NEGATIVE AND FALSE POSITIVE

4.13 PROCEDURE TO CURRENT-BEST-HYPOTHESIS TEST / SEARCH

4.14 VERSION SPACE LEARNING

4.1 LEARNING

Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more effectively the next time.

(or)

A computer program learns if it improves its performance at some task through experience.

(or)

Learning results in changes in the agent (or mind) that improve its competence and/or efficiency.

(or)

Learning is the process of converting experience into expertise or knowledge.

(or)

A machine is said to be learning from past Experiences (data feed-in) with respect to some class of tasks if its Performance in a given Task improves with the Experience.

4.2 FORMS OF LEARNING

There are three types of learning are. They are :

1. **Supervised Learning :** The machine has a "teacher" who guides it by providing sample inputs along with the desired output. The machine then maps the inputs and the outputs. This is similar to how we teach very young children with picture books.
2. **Unsupervised Learning :** Means to act without anyone's supervision or direction. Unsupervised learning, the model is given a dataset which is neither labeled nor classified. The model explores the data and draws inferences from datasets to define hidden structures from unlabelled data.
3. **Reinforcement Learning (RL) :** It is a a sub-field of Machine Learning where the aim is create agents that learn how to operate optimally in a partially random environment by directly interacting with it and observing the consequences of its actions.

4.2.1 SUPERVISED LEARNING

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (X) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(x)$$

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (Y) for that data.

Learning takes place in the presence of a supervisor or a teacher. A supervised learning algorithm learns from labeled training data, helps you to predict outcomes for unforeseen data.

Now a days, almost all learning is supervised. Your data has known labels as output. It involves a supervisor that is more knowledgeable than the neural network itself.

Types of Supervised Learning Algorithms are :

- Regression.
- Logistic Regression.
- Classification.
- Naive Bayes Classifiers.
- K-NN (k nearest neighbors).
- Decision Trees.
- Support Vector Machine.

Supervised learning problems can be further grouped into **regression** and **classification** problems.

Classification : A classification problem is when the output variable is a category, such as "red" or "blue" or "disease" and "no disease".

Regression: A regression problem is when the output variable is a real value, such as "dollars" or "weight". Regression is a ML algorithm that can be trained to predict real numbered outputs; like temperature, stock price, etc. Regression models are used to predict a continuous value. Predicting prices of a house given the features of house like size, price etc is one of the common examples of Regression. It is a supervised technique.

Why Supervised Learning?

- Supervised learning allows you to collect data or produce a data output from the previous experience.
- Helps you to optimize performance criteria using experience.
- Supervised machine learning helps you to solve various types of real-world computation problems.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

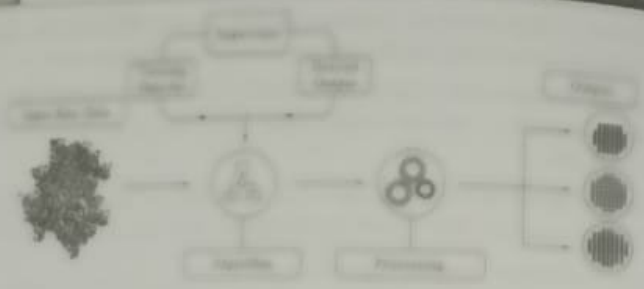


FIG. 4.1 : Supervised Learning

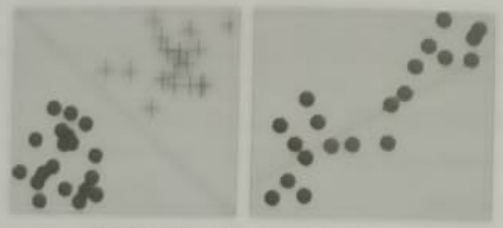


FIG. 4.2 : Types of Supervised Learning

How Supervised Learning works?

For example, you want to train a machine to help you predict how long it will take you to drive home from your workplace. Here, you start by creating a set of labeled data. This data includes

- Weather condition
- Time of the day
- Holidays

The output is the amount of time it took to drive back home on that specific day.

If it's raining outside, then it will take you longer to drive home. But the machine needs data and statistics.

This training set will contain the total commute time and corresponding factors like whether time etc. Based on this training set, your machine might see there's a direct relationship between the amount of rain and time you will take to get home.

So, it ascertains that the more it rains, the longer you will be driving to get back to you home. It might also seen the connection between the time you leave work and the time you'll be on the road.

The closer you're to 6 p.m the longer time it takes for you to get home. Your machine may find some of the relationships with your labeled data.

Advantages :

- Supervised learning allows collecting data and produces data output from previous experiences.
- Helps to optimize performance criteria with the help of experience.
- Supervised machine learning helps to solve various types of real-world computation problems.
- It performs classification and regression tasks.
- It allows estimating or mapping the result to a new sample.
- We have complete control over choosing the number of classes we want in the training data.

Disadvantages :

- Classifying big data can be challenging.
- Training for supervised learning needs a lot of computation time. So, it requires a lot of time.
- Supervised learning cannot handle all complex tasks in Machine Learning.
- Computation time is vast for supervised learning.
- It requires a labelled data set.
- It requires a training process.

4.2.2 UNSUPERVISED LEARNING

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.

Here the task of the machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself. The objective of unsupervised learning is to restructure the input record into new features or a set of objects with some pattern.

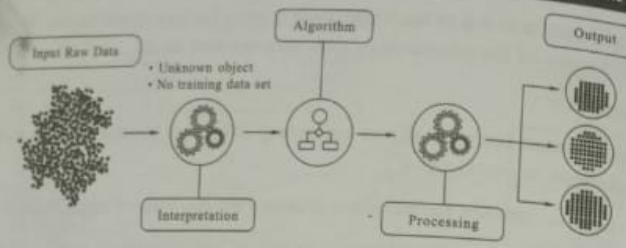


FIG 4.3 : Unsupervised Learning

Example-1 :

FIG 4.4 :

For instance, suppose it is given an image having both dogs and cats which is has never seen. Thus the machine has no idea about the features of dogs and cats so we can't categorize it as 'dogs and cats'. But it can categorize them according to their similarities, paterns and differences, i.e., we can easily categorize the above picture into two parts. The first may contain all pictures having cats in them. Here you didn't learn anything before, which means no training data or examples.

Example-2 :

Google is an instance of clustering that needs unsupervised learning to group news items depends on their contents. Google has a set of millions of news items written on multiple topics are their clustering algorithm necessarily groups these news items into a small number that are same or associated to each other by using multiple attributes, including word frequency, sentence length, page content, etc.

It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data.

Clustering : A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

The different types of clustering are :

- Centroid-based clustering.
- Density-based clustering.
- Distribution-based clustering.
- Hierarchical clustering.

- **Association :** An association rule learning problem is hwere you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y. Association rule learning can be divided into three types of algorithms :
  - Apriori.
  - Eclat (Equvalence class Tranformation).
  - F-P Growth Algorithm (Frequent pattern).

It is employed in :

- Market Basket analysis.
- Web usage mining.
- Continous production.

There are various examples of Unsupervised learning which are as follows :

- Organize computing clusters.
- Social network analysis.
- Market segmentation.
- Astronomical data analysis.

Advantages of Unsupervised Learning :

- It does not require a training data to be labeled.
- Dimensionality reduction can be easily accomplished using unsupervised learning.
- Capable of finding previously unknown patterns in data.

Disadvantages of Unsupervised Learning :

- Difficult to measure accuracy or effectiveness due to lack of predefined answers during training.
- The results often have lesser accuracy.
- The user needs to spend time interpreting and label the classes which follow that classification.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

4.2.3 REINFORCEMENT LEARNING

Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty. In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.

Since there is no labeled data, so the agent is bound to learn by its experience only. RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.

The agent interacts with the environment and explores it by itself. *The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.*

The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that ***"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."*** How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.

It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.

Working of Reinforcement Learning : The working of reinforcement learning is as follows,

- First you need to prepare an agent with some specific set of strategies.
- Now leave the agent to observe the current state of the environment.
- Based on the agent's observation, select the optimal policy, and perform suitable action.
- Based on the action taken, the agent will get reward or penalty.
- Update the set of strategies used in step-1, if needed. Repeat the process from step-1-4 until the agent learns and adopts the optimal policy as well.

Example : Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.

The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment.

The agent learns that what actions lead to positive rd, the agent gets a positive point, and as a penalty, it gets a negative point.

Supervised learning methods, as we know, take both training data and its associated output during the training process. But the unsupervised learning methods do not require any labels or responses along with the training data and they learn patterns and relationships from the given raw data. Whereas, in reinforcement learning methods the agent interacts with a specific environment in discrete steps.

Applications of Reinforcement Learning :

- Manufacturing.
- Finance.
- Dynamic Pricing.
- Medical Industry.
- Inventory Management.
- Delivery Management.
- E-Commerce Personalization.

4.3 INDUCTIVE LEARNING WITH EXAMPLE

Inductive learning is a type of machine learning that uses data to make predictions or generalizations about a given problem. It is based on the idea that if a set of data points have certain characteristics, then future data points will also have those characteristics.

This type of learning is used in many areas of machine learning and data science, such as supervised learning, unsupervised learning, and reinforcement learning. For example, in supervised learning, inductive learning can be used to build a model that can predict the outcome of a given problem based on the data it has been trained on.

In unsupervised learning, inductive learning can be used to identify patterns in data and make predictions about future data points. In reinforcement learning, inductive learning can be used to identify the best action to take in a given situation.

Inductive learning also called Concept Learning is a way in which AI systems try to use a generalized rule to carry out observations. The data is obtained as a result of machine learning or from domain experts (humans) where it is used to drive algorithms often called the Inductive Learning Algorithms (ALIs) that are used to generate a set of classification rules. These rules that are generated are in the **"If this then that"** form. They are rules that determine the state of an entity at every single iteration step

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

in learning and how the learning can be effectively changed by appending more rules to the already existing ruleset. The goal of inductive learning is to learn the function for new data when the output and the examples of the function are input into the AI system.

Inductive Learning may be helpful in the following four situations :

- **Problems in Which no Human Expertise is Available :** People cannot write a program to solve a problem if they do not know the answer. These are areas ripe for exploration.
- **Humans Can Complete the Task, But No One Knows How to Do It :** There are situations in which humans can do things that computers cannot or do not do well. Riding a bike or driving a car are two examples.
- **Problems Where the Desired Function is Frequently Changing :** Humans could describe it and write a program to solve it, but the problem changes too frequently. It is not economical. The stock market is one example.
- **Problems Where Each User Requires a Unique Function :** Writing a custom program for each user is not cost-effective. Consider Netflix or Amazon recommendations for movies or books.

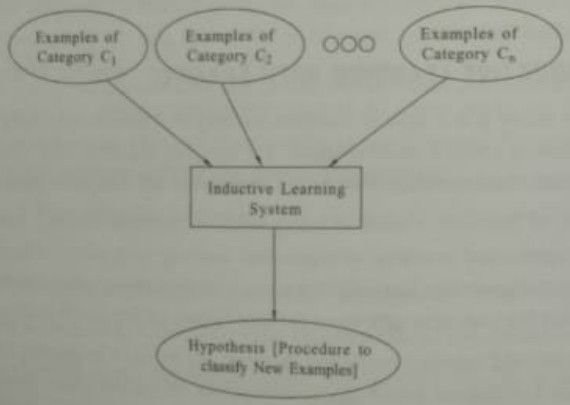


FIG 4.5 : Inductive Learning

Inductive Learning : *Inductive Learning problem is to learn a function from example. Let* ***f*** *be the* ***target*** *function, an example is a pair* ***(x, f(x)),*** *Then the Inductive Learning problem is to find a hypothesis* ***h,*** *such that* ***hH"f,*** *given a training set of examples.*

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

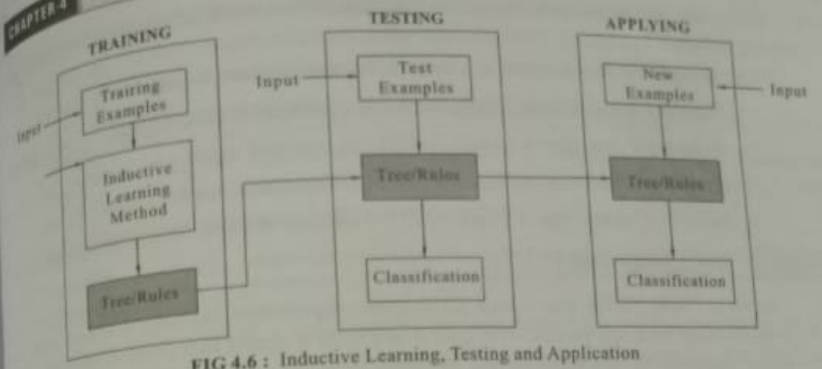


FIG 4.6 : Inductive Learning, Testing and Application

Example : Suppose an example set having attributes Place type, weather, location, decision and seven examples, our task is to generate a set of rules that under what condition what is the decision.

Example No.	Place Type	Wheather	Location	Decision
1.	Hilly	Winter	Kullu	Yes
2.	Mountain	Windy	Mumbai	No
3.	Mountain	Windy	Shimla	Yes
4.	Beach	Windy	Mumbai	No
5.	Beach	Warm	Goa	Yes
6.	Beach	Windy	Goa	No
7.	Beach	Warm	Shimla	Yes

From the above Table, the subset-1 is given by :

S.No.	Place Type	Wheather	Location	Decision
1.	Hilly	Winter	Kullu	Yes
2.	Mountain	Windy	Shimla	Yes
3.	Beach	Warm	Goa	Yes
4.	Beach	Warm	Shimla	Yes

Subset-2 is given by :

S.No.	Place Type	Wheather	Location	Decision
5.	Mountain	Windy	Mumbai	No
6.	Beach	Windy	Mumbai	No
7.	Beach	Windy	Goa	No

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

From the above subsets the following rules are formed using Inductive Learning algorithms :

- **Rule-1** : IF the weather is warm THEN the decision is yes.
- **Rule-2** : IF place type is hilly THEN the decision is yes.
- **Rule-3** : IF location is Shimla THEN the decision is yes.
- **Rule-4** : IF location is Mumbai THEN the decision is no.
- **Rule-5** : IF place type is beach AND the weather is windy THEN the decision is no.

4.4 DECISION TREE

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

A decision tree is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data. It is a tree-like structure where each internal node represents a decision or test on a specific feature or attribute, each branch represents the outcome of that decision, and each leaf node represents the final decision or prediction.

It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

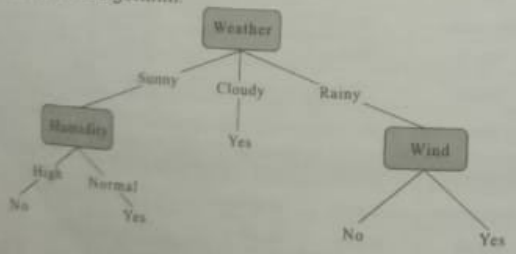


FIG 4.7 : Decision Tree

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

Decision trees in artificial intelligence are used to arrive at conclusions based on the data available from decisions made in the past. Further, these conclusions are assigned values, deployed to predict the course of action likely to be taken in the future.

4.4.1 IMPORTANT TERMINOLOGIES RELATED TO DECISION TREES

1. **Root Node** : The root node is the very first node in the tree. This node initiates the whole decision-making process by further getting divided into 2 or more sets of nodes and each node consists of a feature in the data set.
2. **Splitting** : A node is broken down into sub-nodes at every level and this process is called splitting.
3. **Decision Node** : When a node can be broken down into 2 or more nodes then it's called a decision node. This breaking down happens according to the number of different decisions that can be made from that particular node.
4. **Leaf / Terminal Node** : Nodes that cannot be broken down into sub-nodes anymore.
5. **Pruning** : This is the process of removing an unwanted part of a tree. To be precise it can be a node (or) a branch in the tree too.
6. **Branch/Sub-Tree** : When a tree is split into different sub-parts it is known to be a branch in a tree (or) a sub tree.
7. **Parent and Child Node** : When a node is divided into sub-nodes the sub-nodes are called the child nodes and the node that got split is called the parent node of all these child nodes.

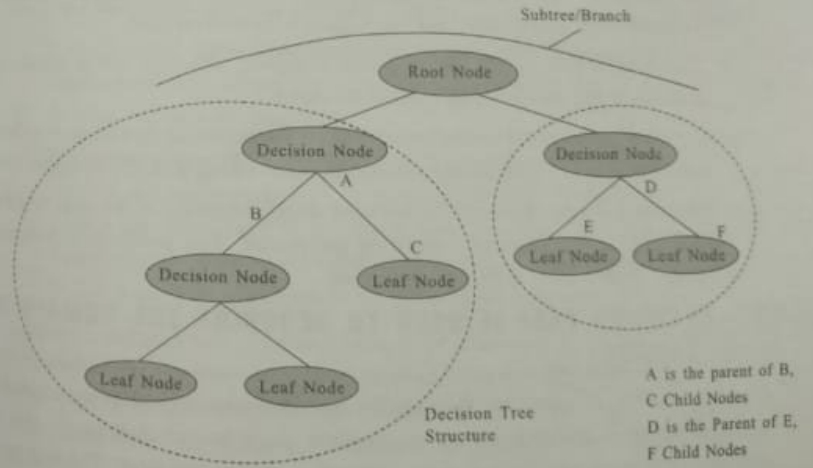


FIG 4.8 : Decision Tree - Terminology

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

4.5 ADVANTAGES AND DISADVANTAGES OF THE DECISION TREE

Advantages of Decision Trees :

- A decision tree is easy to understand and interpret.
- Expert opinion and preferences can be included, as well as hard data.
- Can be used with other decision techniques.
- New scenarios can easily be added.
- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data.
- Able to handle multi-output problems.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

Disadvantages of Decision Trees :

- If a decision tree is used for categorical variables with multiple levels, those variables with more levels will have more information gain.
- Calculations can quickly become very complex, although this is usually only a problem if the tree is being created by hand.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.

4.6 DECISION TREE IS USED TO DESCRIBE THE DOMAIN WITH EXAMPLE

A decision tree is a sequential diagram-like tree structure, where every internal node (non-leaf node) indicates a test on an attribute, each branch defines a result of the test, and each leaf node (or terminal node) influences a class label. The highest node in a tree is the root node.

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

Decision trees are a type of supervised machine learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

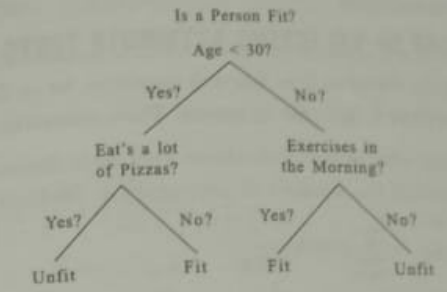


FIG 4.9 : Decision Tree

An example of a decision tree can be explained using above binary tree. Let's say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like 'What's the age?', 'Does he exercise?', 'Does he eat a lot of pizzas'? And the leaves, which are outcomes like either 'fit', or 'unfit'. In this case this was a binary classification problem (a yes no type problem).

There are two main types of Decision Trees :

1. **Classification Trees (Yes/No types) :** What we've seen above is an example of classification tree, where the outcome was a variable like 'fit' or 'unfit'. Here the decision variable is categorical.
2. **Regression Trees (Continuous Data Types) :** Here the decision or the outcome variable is Continuous. **Eg :** A number like 123. Working Now that we know what a Decision Tree is, we'll see how it works internally. There are many algorithms out there which construct Decision Trees, but one of the best is called as ID3 Algorithm. ID3 Stands for *Iterative Dichotomiser 3*.

In general, decision tree classifiers have good efficiency. However, successful use can be based on the data at hand. Decision tree induction algorithms have been used for classification in several application areas, including medicine, manufacturing and production, monetary analysis, astronomy, and molecular biology. Decision trees are based on multiple commercial rule induction systems.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

During tree construction, attribute selection measures are used to choose the attribute that best partitions the tuples into different classes. When decision trees are constructed, some branches can reflect noise or outliers in the training records. Tree pruning tries to recognize and eliminate such branches, with the aim of improving classification accuracy on unviewed data.

4.7 PROCEDURE OF SELECTING ATTRIBUTE TESTS

Before discussing the Decision Tree learning algorithm, let us discuss few measures to select the best attribute to split the examples. Those measures are:

Entrophy : Entropy, also called as Shannon Entropy is denoted by H(S) for a finite set S, is the measure of the amount of uncertainty or randomness in data.

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

Intuitively, it tells us about the predictability of a certain event. Example, consider a coin toss whose probability of heads is 0.5 and probability of tails is 0.5. Here the entropy is the highest possible, since there's no way of determining what the outcome might be. Alternatively, consider a coin which has heads on both the sides, the entropy of such an event can be predicted perfectly since we know beforehand that it'll always be heads. In other words, this event has no randomness hence it's entropy is zero. In particular, lower values imply less uncertainty while higher values imply high uncertainty.

Suppose a feature has 8 "yes" and 4 "no" initially, after the first split the left node gets 5 'yes' and 2 'no' where as right node gets 3 'yes' and 2 'no'.

We see here the split is not pure, why?. Because we can still see some negative classes in both the nodes. In order to make a decision tree, we need to calculate the impurity of each split, and when the purity is 100%, we make it as a leaf node.

To check the impurity of feature 2 and feature 3 we will take the help for Entropy formula.

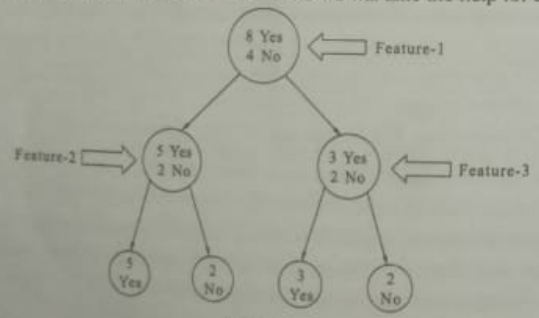


FIG 4.10 :

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

$$\Rightarrow -\left(\frac{5}{7}\right)\log_2\left(\frac{5}{7}\right)-\left(\frac{2}{7}\right)\log_2\left(\frac{2}{7}\right)$$

$$\Rightarrow -(0.71 \times -0.49) - (0.28 \times -1.83)$$

$$\Rightarrow -(-0.34) - (-0.51)$$

$$\Rightarrow 0.34 + 0.51$$

$$\Rightarrow 0.85$$

For feature 3,

$$\Rightarrow -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right)-\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right)$$

$$\Rightarrow -(0.6 \times -0.73) - (0.4 \times -1.32)$$

$$\Rightarrow 0.438 + 0.528$$

$$\Rightarrow 0.966$$

We can clearly see from the tree itself that left node has low entropy or more purity than right node since left node has a greater number of "yes" and it is easy to decide here.

Always remember that the higher the Entropy, the lower will be the purity and the higher will be the impurity.

As mentioned earlier the goal of machine learning is to decrease the uncertainty or impurity in the dataset, here by using the entropy we are getting the impurity of a particular node, we don't know if the parent entropy or the entropy of a particular node has decreased or not.

For this, we bring a new metric called ***"information gain"*** which tells us how much the parent entropy has decreased after splitting it with some feature.

Information Gain : Information gain measures the reduction of uncertainty given some feature and it is also a deciding factor for which attribute should be selected as a decision node or root node.

$$\textbf{Information Gain} = E(Y) - E(Y \mid X)$$

It is just entropy of the full dataset – entropy of the dataset given some feature.

To understand this better let's consider an example: Suppose our entire population has a total of 30 instances. The dataset is to predict whether the person will go to the gym or not. Let's say 16 people go to the gym and 14 people don't.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

Now we have two features to predict whether he/she will go to the gym or not.

Feature-1 is "**Energy**" which takes two values "high" and "low".

Feature-2 is "**Motivation**" which takes 3 values "No motivation", "Neutral" and "Highly motivated".

Let's see how our decision tree will be made using these 2 features. We'll use information gain to decide which feature should be the root node and which feature should be placed after the split.

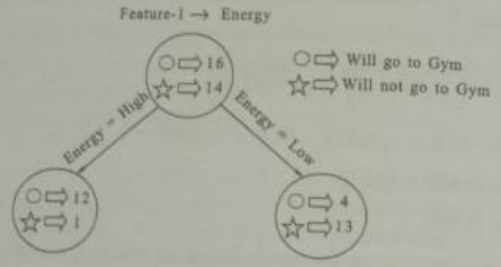


FIG 4.11 :

Let's calculate the entropy :

$$E(\text{Parent}) = -\left(\frac{16}{30}\right)\log_2\left(\frac{16}{30}\right)-\left(\frac{14}{30}\right)\log_2\left(\frac{14}{30}\right)=0.99$$

$$E(\text{Parent} \mid \text{Energy} = \text{"high"}) = -\left(\frac{12}{13}\right)\log_2\left(\frac{12}{13}\right)-\left(\frac{1}{13}\right)\log_2\left(\frac{1}{13}\right)=0.39$$

$$E(\text{Parent} \mid \text{Energy} = \text{"low"}) = -\left(\frac{4}{17}\right)\log_2\left(\frac{4}{17}\right)-\left(\frac{13}{17}\right)\log_2\left(\frac{13}{17}\right)=0.79$$

To see the weighted average of entropy of each node we will do as follows :

$$E(\text{Parent} \mid \text{Motivation}) = \frac{8}{30}\times 0.54+\frac{10}{30}\times 0.97+\frac{12}{30}\times 0.98 = 0.86$$

Now we have the value of E(Parent) and E(Parent | Motivation), information gain will be :

$$\begin{aligned}\text{Information gain} &= E(\text{Parent}) - E(\text{Parent} \mid \text{Movtivation})\\ &= 0.99 - 0.86\\ &= 0.13\end{aligned}$$

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

We now see that the **"Energy"** feature gives more reduction which is 0.37 than the "Motivation" feature. Hence we will select the feature which has the highest information gain and then split the node based on that feature.

In this example **"Energy"** will be our root node and we'll do the same for sub-nodes. Here we can see that when the energy is "high" the entropy is low and hence we can say a person will definitely go to the gym if he has high energy, but what if the energy is low? We will again split the node based on the new feature which is "Motivation".

Gini Index : It is a measure of purity or impurity while creating a decision tree. *It is calculated by subtracting the sum of the squared probabilities of each class from one.* It is the same as entropy but is known to calculate quicker as compared to entropy. CART (Classification and regression tree) uses the Gini index as an attribute selection measure to select the best attribute/feature to split. The attribute with a lower Gini index is used as the best attribute to split.

$$\text{Gini} = 1-\sum_j p^2 j$$

Here 'i' is the no. of classes.

Gain Ratio : The Gain Ratio solves the problem of Bias in Information gain. Information gain shows bias towards the nodes that have a large number of values i.e if an attribute has a large set of values in the data set then makes it a root node. This is something that isn't appreciated while designing a model, so to solve this problem of bias, the number of branches that a node would split into is something that is taken under consideration by the gain ratio while splitting a node. The gain ratio is defined as the ratio of information gain and intrinsic information (or) split info.

Here Intrinsic information/split info refers to the entropy of sub-dataset proportions.

The formula for gain ratio is :

$$\text{Gain Ratio} = \frac{\text{Information gain}}{\text{Split info (or) Instrinsic info}}$$

Reduction in Variance : It is used when the data we have is continuous in nature. Variance simply refers to the change in the model when using different subparts of the training data. The split with lower variance is taken into consideration. To use variance as an attribute selection measure, firstly calculate variance for each node followed by calculating variance for each split.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

$$\text{Variance} = \frac{\sum (X - \bar{X})^2}{n}$$

X bar is the mean of the values, X is actual and n is the number of values.

Chi-Square : Chi-Square is a comparison between observed results and expected results. This statistical method is used in CHAID(Chi-square Automatic Interaction Detector). CHAID in itself is a very old algorithm that is not used much these days. The higher the value of chi-square, higher is the difference between the current node and its parent.

The formula for chi-square is :

$$x^2 = \sum \frac{(O - E)^2}{E}$$

where, x^2 = Chi square obtained

Σ = The Sum of

O = Observed score

E = Expected score

4.8 DECISION TREE LEARNING ALGORITHM WITH EXAMPLE

A decision tree can be constructed using following algorithms:

ID3 : ID3 stands for *Iterative Dichotomiser 3* . This algorithm iteratively divides features into two or more groups at each step. Here "Iterative" means continuous/repeated and "dichotomiser" means dividing. ID3 follows a top-down approach i.e the tree is built from the top and at every step greedy approach is applied. The greedy approach means that at each iteration we select the best feature at the present moment to create a node and this node is again split after applying some statistical methods. ID3 generally is not a very ideal algorithm as it overfits when the data is continuous.

C4.5 : It is considered to be better than the ID3 algorithm as it can handle both discrete and continuous data. In C4.5 splitting is done based on Information gain (attribute selection measure) and the feature with the highest Information gain is made the decision node and is further split. C4.5 handles overfitting by the method of pruning i.e it removes the branches/subpart of the tree that does not hold much importance (or) is redundant. To be specific, C4.5 follows post pruning i.e removing branches after the tree is created.

CART : CART stands for classification and regression trees. As the name suggests CART can also perform both classification and regression-based tasks. CART uses Gini's impurity index as an attribute selection method while splitting a node into further nodes when it's a classification-based use case and uses sum squared error as an

WARNING XEROX ! PHOTOCOPYING OF THIS BOOK IS ILLEGAL

attribute selection measure when the use case is regression-based. While CART uses Gini Index as an ASM (attribute selection measure), C4.5 and ID3 use information gain as an ASM.

CHAID : CHAID stands for Chi-square Automatic Interaction Detector. It is known to be the oldest of all three algorithms in history and is used very less these days. In CHAID chi-square is the attribute selection measure to split the nodes when it's a classification-based use case and uses F-test as an attribute selection measure when it is a regression-based use case. Higher the chi-square value higher is the preference given to that feature. The major difference between CHAID and CART is, CART splits one node into two nodes whereas CHAID splits one node into 2 or more nodes.

MARS : MARS stands for Multivariate adaptive regression splines. It is an algorithm that was specifically designed to handle regression-based tasks, provided, the data is non-linear.

In this chapter we will discuss ID3 algorithm to construct a Decision Tree for a given problem domain. Consider a piece of data collected over the course of 14 days where the features are Outlook, Temperature, Humidity, Wind and the outcome variable is whether Golf was played on the day. Now, our job is to build a predictive model which takes in above 4 parameters and predicts whether Golf will be played on the day. We'll build a decision tree to do that using ID3 algorithm.

Day	Outlook	Temperature	Humidity	Wind	Play Golf
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

XEROX ! PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

ID3 Algorithm will perform following tasks recursively :

- *Create root node for the tree*
- *If all examples are positive, return leaf node 'positive'*
- *Else if all examples are negative, return leaf node 'negative'*
- *Calculate the entropy of current state H(S)*
- *For each attribute, calculate the entropy with respect to the attribute 'x' denoted by H(S, x)*
- *Select the attribute which has maximum value of IG(S, x)*
- *Remove the attribute that offers highest IG from the set of attributes*
- *Repeat until we run out of all attributes, or the decision tree has all leaf nodes.*

Now, let's go ahead and grow the decision tree. The initial step is to calculate H(S), the Entropy of the current state. In the above example, we can see in total there are 5 No's and 9 Yes's.

Yes	No	Total
9	5	14

$$\text{Entropy (S)} = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

$$\text{Entropy (S)} = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)$$

$$= 0.940$$

Remember that the Entropy is 0 if all members belong to the same class, and 1 when half of them belong to one class and other half belong to other class that is perfect randomness. Here it's 0.94 which means the distribution is fairly random. Now, the next step is to choose the attribute that gives us highest possible Information Gain which we'll choose as the root node. Let's start with 'Wind'.

$$IG(S, \text{Wind}) = H(S) - \sum_{i=0}^{n} p(x) \times H(x)$$

where 'x' are the possible values for an attribute. Here, attribute 'Wind' takes two possible values in the sample data, hence x = {Weak, Strong} We'll have to calculate :

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

1. $H(S_{weak})$
2. $H(S_{strong})$
3. $P(S_{weak})$
4. $P(S_{strong})$
5. $H(S) = 0.94$ Which we had already calculated in the previous example.

Amongst all the 14 examples we have 8 places where the wind is weak and 6 where the wind is Strong.

Wind = Weak	Wind = Strong	Total
8	6	14

$$P(S_{weak}) = \frac{\text{Number of Weak}}{\text{Total}}$$

$$= \frac{8}{14}$$

$$P(S_{strong}) = \frac{\text{Number of Strong}}{\text{Total}}$$

$$= \frac{6}{14}$$

Now, out of the 8 Weak examples, 6 of them were 'Yes' for Play Golf and 2 of them were 'No' for 'Play Golf'. So, we have,

$$\text{Entropy } (S_{weak}) = -\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right)$$

$$= 0.811$$

Similarly, out of 6 Strong examples, we have **3 examples where the outcome was 'Yes' for Play Golf and 3 where we had 'No' for Play Golf**.

$$\text{Entropy } (S_{strong}) = -\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right)$$

$$= 1.000$$

Remember, here half items belong to one class while other half belong to other. Hence we have perfect randomness. Now we have all the pieces required to calculate the Information Gain,

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

$$IG(S, Wind) = H(S) - \sum_{i=0}^{n} P(x) \times H(x)$$

$$IG(S, Wind) = H(S) - P(S_{weak}) \times H(S_{weak}) - P(S_{strong}) \times H(S_{strong})$$

$$= 0.940 - \left(\frac{8}{14}\right)(0.811) - \left(\frac{6}{14}\right)(1.00)$$

$$= 0.048$$

Which tells us the Information Gain by considering 'Wind' as the feature and give us information gain of 0.048. Now we must similarly calculate the Information Gain for all the features.

IG(S, Outlook) = 0.246

IG(S, Temperature) = 0.029

IG(S, Humidity) = 0.151

IG(S, Wind) = 0.048 (Previous example)

We can clearly see that IG(S, Outlook) has the highest information gain of 0.246, hence we chose Outlook attribute as the root node. At this point, the decision tree looks like.

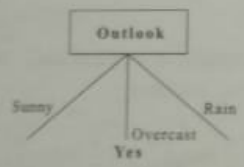


FIG 4.12 :

Here we observe that whenever the outlook is Overcast, Play Golf is always 'Yes', it's no coincidence by any chance, the simple tree resulted because of **the highest information gain is given by the attribute Outlook**. Now how do we proceed from this point? We can simply apply recursion, you might want to look at the algorithm steps described earlier. Now that we've used Outlook, we've got three of them remaining Humidity, Temperature, and Wind. And, we had three possible values of Outlook: Sunny, Overcast, Rain. Where the Overcast node already ended up having leaf node 'Yes', so we're left with two subtrees to compute: Sunny and Rain.

Next step would be computing $H(S_{sunny})$.

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

Table where the value of Outlook is Sunny looks like

Temperature	Humidity	Wind	PlayGolf
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

$$H(S_{sunny}) = -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.96$$

In the similar fashion, we compute the following values

$IG(S_{sunny}, Humidity) = 0.96$

$IG(S_{sunny}, Temperature) = 0.57$

$IG(S_{sunny}, Wind) = 0.019$

As we can see the highest Information Gain is given by Humidity. Proceeding in the same way with S_{rain} will give us Wind as the one with highest information gain. The final Decision Tree looks something like this.

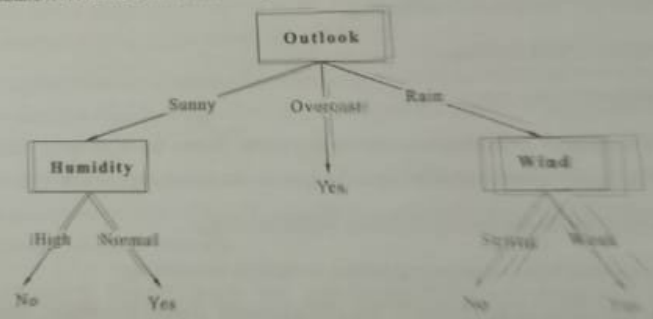


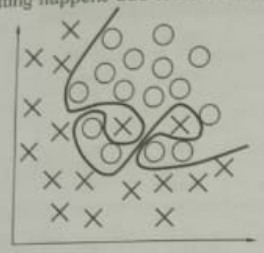
FIG 4.13 :

4.9 OVER-FITTING PROBLEM

A statistical model is said to be over fitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

Overfitting occurs when the model cannot generalize and fits too closely to the training dataset instead. Overfitting happens due to several reasons, such as :



Over-Fitting
(Forcefitting--too Good to be True)

FIG 4.14 : Over-fitting

- The training data size is too small and does not contain enough data samples to accurately represent all possible input data values.
- The training data contains large amounts of irrelevant information, called noisy data.
- The model trains for too long on a single sample set of data.
- The model complexity is high, so it learns the noise within the training data.

Techniques to Reduce Overfitting :

- Increase training data.
- Reduce model complexity.
- Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
- Ridge Regularization and Lasso Regularization
- Use dropout for neural networks to tackle overfitting.

4.10 TECHNIQUE EMPLOYED IN HANDLING OVER-FITTING PROBLEM

You can prevent overfitting by diversifying and scaling your training data set or using some other data science strategies, like below.

Early Stopping : Early stopping pauses the training phase before the machine learning model learns the noise in the data. However, getting the timing right is important; else the model will still not give accurate results.

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

1. **Pruning :** You might identify several features or parameters that impact the final prediction when you build a model. Feature selection– or pruning – identifies the most important features within the training set and eliminates irrelevant ones. For example, to predict if an image is an animal or human, you can look at various input parameters like face shape, ear position, body structure, etc. You may prioritize face shape and ignore the shape of the eyes.
2. **Regulatization :** Regularization is a collection of training/optimization techniques that seek to reduce overfitting. These methods try to eliminate those factors that do not impact the prediction outcomes by grading features based on importance. For example, mathematical calculations apply a penalty value to features with minimal impact. Consider a statistical model attempting to predict the housing prices of a city in 20 years. Regularization would give a lower penalty value to features like population growth and average annual income but a higher penalty value to the average annual temperature of the city.
3. **Ensembling :** Ensembling combines predictions from several separate machine learning algorithms. Some models are called weak learners because their results are often inaccurate. Ensemble methods combine all the weak learners to get more accurate results. They use multiple models to analyze sample data and pick the most accurate outcomes. The two main ensemble methods are bagging and boosting. Boosting trains different machine learning models one after another to get the final result, while bagging trains them in parallel.
4. **Data Augmentaiton :** Data augmentation is a machine learning technique that changes the sample data slightly every time the model processes it. You can do this by changing the input data in small ways. When done in moderation, data augmentation makes the training sets appear unique to the model and prevents the model from learning their characteristics. For example, applying transformations such as translation, flipping, and rotation to input images.

4.11 ISSUES THAT ARE TO BE CONSIDERED FOR EXTENDING THE APPLICABILITY OF DECISION TREE

Practical issues in learning decision trees are Determining how deep to grow the decision tree, handling continuous attributes, choosing an appropriate attribute selection measure, and handling training data with missing attribute values, handling attributes with different costs, and improving computational efficiency are all practical issues in learning decision trees.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

4.12 FALSE NEGATIVE AND FALSE POSITIVE

A false positive (+) describes that the results states you have the condition that were tested for, but you don not really have it. A false negative (-) means that the results states that you do not have a condition, but you actually do.

A false positive is where you receive a positive result for a test, when you should have received a negative results. It's sometimes called a "**false alarm**" or "false positive error." It's usually used in the medical field, but it can also apply to other arenas (like software testing). Some examples of false positives:

- A pregnancy test is positive, when in fact you aren't pregnant.
- A cancer screening test comes back positive, but you don't have the disease.
- A prenatal test comes back positive for Down's Syndrome, when your fetus does not have the disorder(1).
- Virus software on your computer incorrectly identifies a harmless program as a malicious one.

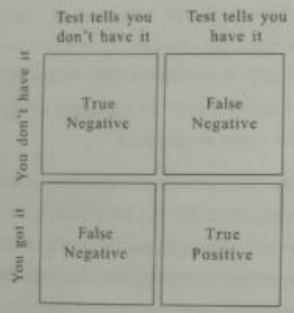


FIG 4.15 : False Positives and False Negatives

A false negative is where a negative test result is wrong. In other words, you get a negative test result, but you should have got a positive test result.

False negatives can happen in areas, like :

- Quality control in manufacturing; a false negative in this area means that a defective item passes through the cracks.
- In software testing, a false negative would mean that a test designed to catch something (i.e. a virus) has failed.

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

4.13 PROCEDURE TO CURRENT-BEST-HYPOTHESIS TEST/SEARCH

Current Best Hypothesis Search (CBHS) is a machine learning algorithm that focuses on finding the best hypothesis (or solution) to a problem, given a set of observed data.

CBHS is commonly used in situations where there is a large search space, and it is not possible to exhaustively search the entire space to find the optimal solution.

In CBHS, the algorithm begins by generating a set of initial hypotheses. It then evaluates each hypothesis by comparing it to the observed data and computing a score that measures the fit between the hypothesis and the data.

The hypothesis with the best score is selected as the "current best hypothesis," and the algorithm continues to generate and evaluate new hypotheses until a stopping criterion is met.

The idea behind the current best hypothesis search is to maintain a single hypothesis and to adjust it as a new example arise in order to maintain consistency.

The hypothesis space H is a set of hypothesis that learning algorithm is designed to entertain. The learning algorithm believes that one hypothesis is correct, that is, it believes the sentence :

$$h_1 \vee h_2 \vee h_3 \vee \ldots \vee h_n$$

Hypothesis that is not consistent with the example can be ruled out.

There are two possible ways to be inconsistent with an example :

(i) **False Negative :** In this hypothesis, the example should be negative but in fact it is positive.

(ii) **False Positive :** In this type of hypothesis, the example should be positive but in fact it is negative.

If the example is consistent with the hypothesis then do not change it. If the example is false negative then, generalize the hypothesis and if the example is false positive then specialize the hypothesis.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

```
function CURRENT-BEST-LEARNING(examples) return a hypothesis
H<- any hypothesis consistent with the first example in examples
for each remaining example in examples do
    if e is a false positive for H then
        H← choose a specialization of H is consistent with examples
    else if e is a false negative for H then
        H ← choose a generalization of H is consistent with examples
    if no consistent specialization/generalization can be found
    then fail
return H
```

Disadvantages :

- Checking all the previous instances over again for each modification is very expensive.
- The search process may involve a great deal of backtracking. Hypothesis space can be a doubly exponentially large place.

4.14 VERSION SPACE LEARNING

The set of all valid hypotheses provided by an algorithm is called Version Space (VS) with respect to the hypothesis space H and the given example set D.

Definition : The Version Space, denoted $VS_{H,D}$, with respect to Hypothesis space H and training examples D, is the subset of hypothesis from H consistent with the training examples in D.

$$VS_{H,D} \equiv [h \in H \mid \text{Consistent } (h, D)]$$

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. Version spaces are more prominent in Boolean function learning.

The version space method is a concept learning process accomplished by managing multiple models within a version space.

Let's assume that we are given a list of inputs X and their corresponding outputs f(X), along with this we also have a per-defined Hypothesis Space H_v. With version space learning, we will look at each pair of input-output and rule out hypothesis from H_v that are not consistent, finally reaching at a subset of H_v. This subset of Hv, let's name

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

it "h" is consistent with our data. How do we say h is consistent with our data-set you ask? h is said to consistent with our dataset if h(x) = f(x) for all x in X.

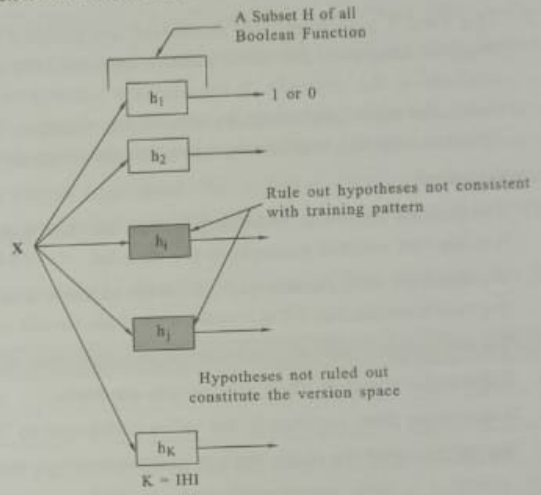


FIG 4.16: Hypothesis Space Consistent with given Example X

FIND-S Algorithm: (*Finding a Maximally Specific Hypothesis*) : Before getting into the Find-S Algorithm, let us know about the following concepts.

- **Concept Learning :** Concept Learning can be viewed as the task of searching through a large space of hypothesis implicitly defined by the hypothesis representation. The goal of the concept learning search is to find the hypothesis that best fits the learning examples.
- **General Hypothesis :** The general hypothesis basically states the general relationship between the major variables. For example, a general hypothesis for ordering food would be ***I want a burger***.

$$G = [\text{'?', '?', '?', ...'?'}]$$

- **Specific Hypothesis :** The specific hypothesis fills in all the important details about the variables given in the general hypothesis. The more specific details into the example given above would be ***I want a cheeseburger with a chicken pepperoni filling with a lot of lettuce.***

$$S = [\text{'}\phi\text{', '}\phi\text{', '}\phi\text{', ..., '}\phi\text{'}]$$

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

The Find-S algorithm follows the steps written below :

1. Initialize 'h' to the most specific hypothesis.
2. The Find-S algorithm only considers the positive examples and eliminates negative examples. For each positive example, the algorithm checks for each attribute in the example. If the attribute value is the same as the hypothesis value, the algorithm moves on without any changes. But if the attribute value is different than the hypothesis value, the algorithm changes it to '?'.

How it works?

1. The process starts with initializing 'h' with the most specific hypothesis, generally, it is the first positive example in the data set.
2. We check for each positive example. If the example is negative, we will move on to the next example but if it is a positive example we will consider it for the next step.
3. We will check if each attribute in the example is equal to the hypothesis value.
4. If the value matches, then no changes are made.
5. If the value does not match, the value is changed to '?'.
6. We do this until we reach the last positive example in the data set.

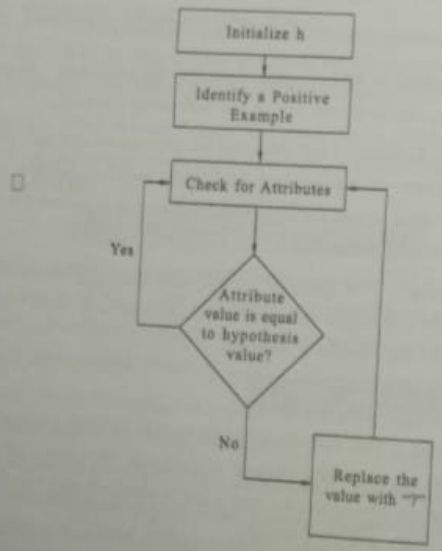


FIG 4.17 : Find-S Algorithm

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

Limitations of Find-S Algorithm : There are a few limitations of the Find-S algorithm listed down below:

- There is no way to determine if the hypothesis is consistent throughout the data.
- Inconsistent training sets can actually mislead the Find-S algorithm, since it ignores the negative examples.
- Find-S algorithm does not provide a backtracking technique to determine the best possible changes that could be done to improve the resulting hypothesis.

Implementation of Find-S Algorithm : To understand the implementation, let us try to implement it to a smaller data set with a bunch of examples to decide if a person wants to go for a walk.

The concept of this particular problem will be on what days does a person likes to go on walk.

Time	Weather	Temperature	Company	Humidity	Wind	Goes
Morning	Sunny	Warm	Yes	Mild	Strong	Yes
Evening	Rainy	Cold	No	Mild	Normal	No
Morning	Sunny	Moderate	Yes	Normal	Normal	Yes
Evening	Sunny	Cold	Yes	High	Strong	Yes

Looking at the data set, we have six attributes and a final attribute that defines the positive or negative example. In this case, yes is a positive example, which means the person will go for a walk.

So now, the general hypothesis is:

h0 = {'Morning', 'Sunny', 'Warm', 'Yes', 'Mild', 'Strong'}

This is our general hypothesis, and now we will consider each example one by one, but only the positive examples.

h1 = {'Morning', 'Sunny', '?', 'Yes', '?', '?'}

h2 = {'?', 'Sunny', '?', 'Yes', '?', '?'}

We replaced all the different values in the general hypothesis to get a resultant hypothesis.

To overcome the limitations of the Find-S algorithm, we will look into another version space learning algorithm called as Candidate Elimination Algorithm.

Candidate Elimination Algorithm : Unlike Find-S algorithm, the Candidate Elimination algorithm considers not just positive but negative samples as well. It relies on the concept of version space.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING

The candidate-elimination algorithm manipulates the boundary-set representation of a version space to create boundary sets that represent a new version space consistent with all the previous instances plus the new one. In case of positive examples, the algorithm generalizes the elements of the [sbs] (i.e. Specific Boundary set) as little as possible so that they cover the new instance yet remain consistent with past data, and removes those elements of the [gbs] that do not cover the new instance. And for a negative instance the algorithm specializes elements of the [gbs] (i.e. General Boundary Set) so that they no longer cover the new instance yet remain consistent with past data, and removes from the [sbs] those elements that mistakenly cover the new, negative instance.

A hypothesis is sufficient if it is 1 for for all training samples labelled 1 and is said to be necessary if it is 0 for all training samples labelled 0. A hypothesis that is both necessary and sufficient is said to be consistent with our dataset.

Algorithm :

For each training example d, do:
  If d is positive example
    Remove from G any hypothesis h inconsistent with d
    For each hypothesis s in S not consistent with d:
      Remove s from S
      Add to S all minimal generalizations of s consistent with d and having a generalization in G
      Remove from S any hypothesis with a more specific h in S
  If d is negative example
    Remove from S any hypothesis h inconsistent with d
For each hypothesis g in G not consistent with d:
      Remove g from G
      Add to G all minimal specializations of g consistent with d and having a specialization in S
      Remove from G any hypothesis having a more general hypothesis in G

Let's consider the below example and trace the Candidate Elimination Algorithm:

Example	Size	Color	Shape	Class/Label
1	Big	Red	Circle	No
2	Small	Red	Triangle	No
3	Small	Red	Circle	Yes
4	Big	Blue	Circle	No
5	Small	Blue	Circle	Yes

WARNING XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

Let's start with defining

S0 : (0, 0, 0) Most Specific Boundary

G0 : (?, ?, ?) Most Generic Boundary

The first example is negative, the hypothesis at the specific boundary is consistent, hence we retain it, and the hypothesis at the generic boundary is inconsistent hence we write all consistent hypotheses by removing one "?" at a time.

S1 : (0, 0, 0)

G1 : (Small, ?, ?), (?, Blue, ?), (?, ?, Triangle)

The second example is negative, the hypothesis at the specific boundary is consistent, hence we retain it, and the hypothesis at the generic boundary is inconsistent hence we write all consistent hypotheses by removing one "?" at a time.

S2 : (0, 0, 0)

G2 : (Small, Blue, ?), (Small, ?, Circle), (?, Blue, ?), (Big, ?, Triangle), (?, Blue, Triangle)

The third example is positive, the hypothesis at the specific boundary is inconsistent, hence we extend the specific boundary, and the consistent hypothesis at the generic boundary is retained and inconsistent hypotheses are removed from the generic boundary.

S3 : (Small, Red, Circle)

G3 : (Small, ?, Circle)

The fourth example is negative, the hypothesis at the specific boundary is consistent, hence we retain it, and the hypothesis at the generic boundary is inconsistent hence we write all consistent hypotheses by removing one "?" at a time.

S4 : (Small, Red, Circle)

G4 : (Small, ?, Circle)

The fifth example is positive, the hypothesis at the specific boundary is inconsistent, hence we extend the specific boundary, and the consistent hypothesis at the generic boundary is retained and inconsistent hypotheses are removed from the generic boundary.

S5 : (Small, ?, Circle)

G5 : (Small, ?, Circle)

Learned Version Space by Candidate Elimination Algorithm for given data set is:

S: G : (Small, ?, Circle)

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL WARNING