**Artificial Intelligence**

|  |
| --- |
| **UNIT – III:** |
| Strong slot and Filter structures, Conceptual Dependencies, Scripts. Introduction to Non monotonic reasoning ,Logics for Non monotonic reasoning, Implementation : Depth First Search, Dependency-Directed Back Tracking, Justification based Truth Maintenance System, Logic based Truth Maintenance Systems, Statistical Reasoning, Probability and Bayes Theorem,Certainty factors, Rule based Systems, Beyesian Networks, Dempster-Shaffer Theory. |
| **UNIT – IV:** |
| Minmax search, alpha-beta cutoffs, Planning system, Goal stack planning, Hierarchical Planning, Natural Language Processing., Syntactic Analysis, Semantic Analysis, Discuses and Pragmatic Processing. Introduction and Fundamentals of Artificial Neural Networks, Biological Prototype, Artificial Neuron, Single Layer Artificial Neural Networks, Multilayer Artificial Neural Networks, Training of Artificial Neural Networks |

**Unit-III**

**Strong Slot and Filler Structures**

*Strong Slot and Filler Structures* typically:

* Represent links between objects according to more *rigid* rules.
* Specific notions of what types of object and relations between them are provided.
* Represent knowledge about common situations.

1. **Conceptual Dependency (CD)**

Conceptual Dependency originally developed to represent knowledge acquired from natural language input.

The goals of this theory are:

* To help in the drawing of inference from sentences.
* To be independent of the words used in the original input.
* That is to say: *For any 2 (or more) sentences that are identical in meaning there should be only one representation of that meaning.*

It has been used by many programs that portend to understand English (*MARGIE, SAM, PAM*). CD provides:

* a structure into which nodes representing information can be placed
* a specific set of primitives
* at a given level of granularity.

Sentences are represented as a series of diagrams depicting actions using both abstract and real physical situations.

* The agent and the objects are represented
* The actions are built up from a set of primitive acts which can be modified by tense.

Examples of Primitive Acts are:

**ATRANS**

-- Transfer of an abstract relationship. *e.g. give*.

**PTRANS**

-- Transfer of the physical location of an object. *e.g. go*.

**PROPEL**

-- Application of a physical force to an object. *e.g. push*.

**MTRANS**

-- Transfer of mental information. *e.g. tell*.

**MBUILD**

-- Construct new information from old. *e.g. decide*.

**SPEAK**

-- Utter a sound. *e.g. say*.

**ATTEND**

-- Focus a sense on a stimulus. *e.g. listen, watch*.

**MOVE**

-- Movement of a body part by owner. *e.g. punch, kick*.

**GRASP**

-- Actor grasping an object. *e.g. clutch*.

**INGEST**

-- Actor ingesting an object. *e.g. eat*.

**EXPEL**

-- Actor getting rid of an object from body. *e.g. ????*.

Six primitive conceptual categories provide *building blocks* which are the set of allowable dependencies in the concepts in a sentence:

**PP**

-- Real world objects.

**ACT**

-- Real world actions.

**PA**

-- Attributes of objects.

**AA**

-- Attributes of actions.

**T**

-- Times.

**LOC**

-- Locations.

How do we connect these things together?

Consider the example:

*John gives Mary a book*

picture957

* Arrows indicate the direction of dependency. Letters above indicate certain relationships:

**o**

-- object.

**R**

-- recipient-donor.

**I**

-- instrument *e.g. eat with a spoon*.

**D**

-- destination *e.g. going home*.

* Double arrows (tex2html_wrap_inline7304) indicate *two-way* links between the actor (PP) and action (ACT).
* The actions are built from the set of primitive acts (see above).
  + These can be modified by *tense etc.*

The use of tense and mood in describing events is extremely important and schank introduced the following modifiers:

**p**

-- past

**f**

-- future

**t**

-- transition

**tex2html_wrap_inline7306**

-- start transition

**tex2html_wrap_inline7308**

-- finished transition

**k**

-- continuing

**?**

-- interrogative

**/**

-- negative

**delta**

-- timeless

**c**

-- conditional

the absence of any modifier implies the *present tense*.

So the *past tense* of the above example:

*John gave Mary a book* becomes:

picture997

The tex2html_wrap_inline7304 has an object (actor), PP and action, ACT. *I.e.* PP tex2html_wrap_inline7304 ACT. The triplearrow (tex2html_wrap_inline7316) is also a two link but between an object, PP, and its attribute, PA. *I.e.* PP tex2html_wrap_inline7316 PA.

It represents *isa* type dependencies. *E.g*

Dave tex2html_wrap_inline7316 lecturerDave is a lecturer.

*Primitive states* are used to describe many state descriptions such as height, health, mental state, physical state.

There are many more physical states than primitive actions. They use a numeric scale.

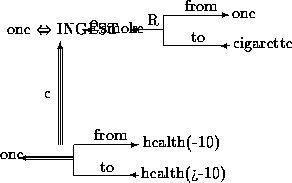
*E.g.* John tex2html_wrap_inline7316 height(+10) *John is the tallest* John tex2html_wrap_inline7316 height(< average) *John is short* Frank Zappa tex2html_wrap_inline7316 health(-10) *Frank Zappa is dead* Dave tex2html_wrap_inline7316 mental\_state(-10) *Dave is sad* Vase tex2html_wrap_inline7316 physical\_state(-10) *The vase is broken*

You can also specify things like the time of occurrence in the relation ship.

For Example: *John gave Mary the book yesterday*  
  
  
picture1049

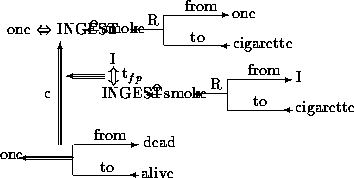
Now let us consider a more complex sentence: *Since smoking can kill you, I stopped*  Lets look at how we represent the inference that *smoking can kill*:

* Use the notion of *one* to apply the knowledge to.
* Use the *primitive act* of INGESTing smoke from a cigarette to one.
* Killing is a transition from being alive to dead. We use *triple arrow*s to indicate a transition from one state to another.
* Have a conditional, *c* causality link. The *triple arrow* indicates dependency of one concept on another.



To add the fact that *I stopped smoking*

* Use similar rules to imply that I smoke cigarettes.
* The qualification tex2html_wrap_inline7338 attached to this dependency indicates that the instance INGESTing smoke has stopped.



Advantages of CD:

* Using these primitives involves fewer inference rules.
* Many inference rules are already represented in CD structure.
* The holes in the initial structure help to focus on the points still to be established.

Disadvantages of CD:

* Knowledge must be decomposed into fairly low level primitives.
* Impossible or difficult to find correct set of primitives.
* A lot of inference may still be required.
* Representations can be complex even for relatively simple actions. Consider:

*Dave bet Frank five pounds that Wales would win the Rugby World Cup*.

Complex representations require a lot of storage

Applications of CD:

**MARGIE**

(*Meaning Analysis, Response Generation and Inference on English*) -- model natural language understanding.

**SAM**

(*Script Applier Mechanism*) -- Scripts to understand stories.

**PAM**

(*Plan Applier Mechanism*) -- Scripts to understand stories.

1. **Scripts**

A *script* is a structure that prescribes a set of circumstances which could be expected to follow on from one another.

It is similar to a thought sequence or a chain of situations which could be anticipated.

It could be considered to consist of a number of slots or frames but with more specialised roles.

Scripts are beneficial because:

* Events tend to occur in known runs or patterns.
* Causal relationships between events exist.
* Entry conditions exist which allow an event to take place
* Prerequisites exist upon events taking place. *E.g.* when a student progresses through a degree scheme or when a purchaser buys a house.

The components of a script include:

**Entry Conditions**

-- these must be satisfied before events in the script can occur.

**Results**

-- Conditions that will be true after events in script occur.

**Props**

-- Slots representing objects involved in events.

**Roles**

-- Persons involved in the events.

**Track**

-- Variations on the script. Different tracks may share components of the same script.

**Scenes**

-- The sequence of *events* that occur. *Events* are represented in *conceptual dependency* form.

Scripts are useful in describing certain situations such as robbing a bank. This might involve:

* Getting a gun.
* Hold up a bank.
* Escape with the money.

Here the *Props* might be

* Gun, *G*.
* Loot, *L*.
* Bag, *B*
* Get away car, *C*.

The *Roles* might be:

* Robber, *S*.
* Cashier, *M*.
* Bank Manager, *O*.
* Policeman, *P*.

The *Entry Conditions* might be:

* *S* is poor.
* *S* is destitute.

The *Results* might be:

* *S* has more money.
* *O* is angry.
* *M* is in a state of shock.
* *P* is shot.

There are 3 scenes: obtaining the gun, robbing the bank and the getaway.

The full Script could be described in the following figure

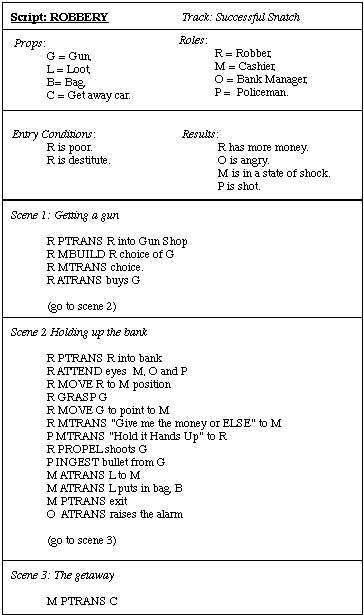


Fig.  Simplified Bank Robbing Script

Some additional points to note on Scripts:

* If a particular script is to be applied it must be activated and the activating depends on its significance.
* If a topic is mentioned in passing then a pointer to that script could be held.
* If the topic is important then the script should be opened.
* The danger lies in having too many active scripts much as one might have too many windows open on the screen or too many recursive calls in a program.
* Provided events follow a known trail we can use scripts to represent the actions involved and use them to answer detailed questions.
* Different trails may be allowed for different outcomes of Scripts (*e.g.* The bank robbery goes wrong).

Advantages of Scripts:

* Ability to predict events.
* A single coherent interpretation may be build up from a collection of observations.

Disadvantages:

* Less general than frames.
* May not be suitable to represent all kinds of knowledge.

**Note:** Students are advised to follow Conceptual Dependency (CD)[ all 14 rules] and Script for Riche Khight AI book.

**Questions**

1. Construct CD representation of the following:
   1. John begged Mary for a pencil.
   2. Jim stirred his coffee with a spoon.
   3. Dave took the book off Jim.
   4. On my way home, I stopped to fill my car with petrol.
   5. I heard strange music in the woods.
   6. Drinking beer makes you drunk.
   7. John killed Mary by strangling her.
   8. .
2. What is a CD ? Explain CD with examples.
3. Write the script of 1) Supper market 2) Eating in a restaurant
4. Write a script for enrolling as a student.
5. **Introduction to Non monotonic reasoning**

**Reasoning:**

The reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs. Or we can say, "**Reasoning is a way to infer facts from existing data**." It is a general process of thinking rationally, to find valid conclusions.

**Reasoning** is the act of deriving a conclusion from certain premises using a given methodology.

■ Any knowledge system must reason, if it is required to do something which has not been told explicitly .

■ For reasoning, the system must find out what it needs to know from what it already knows.

**Example :**

If we know : *Robins are birds*.

*All birds have wings*

Then if we ask: *Do robins have wings?*

To answer this question - some reasoning must go.

In artificial intelligence, the reasoning is essential so that the machine can also think rationally as a human brain, and can perform like a human.

Types of Reasoning

In artificial intelligence, reasoning can be divided into the following categories:

* **Monotonic Reasoning**
* **Non-monotonic Reasoning**

**1. Monotonic Reasoning:**

In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base.

In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.

To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.

Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.

Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic.

Any theorem proving is an example of monotonic reasoning.

**Example:**

* **Earth revolves around the Sun.**

It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like, "The moon revolves around the earth" Or "Earth is not round," etc.

**Advantages of Monotonic Reasoning:**

* In monotonic reasoning, each old proof will always remain valid.
* If we deduce some facts from available facts, then it will remain valid for always.

**Disadvantages of Monotonic Reasoning:**

* We cannot represent the real world scenarios using Monotonic reasoning.
* Hypothesis knowledge cannot be expressed with monotonic reasoning, which means facts should be true.
* Since we can only derive conclusions from the old proofs, so new knowledge from the real world cannot be added.

**2. Non-monotonic Reasoning**

In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.

Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.

Non-monotonic reasoning deals with incomplete and uncertain models.

"Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.

**Example:** Let suppose the knowledge base contains the following knowledge:

* **Birds can fly**
* **Penguins cannot fly**
* **Twetty is a bird**

So from the above sentences, we can conclude that **Twetty can fly**.

However, if we add one another sentence into knowledge base " **Twetty is a penguin**", which concludes " **Twetty cannot fly**", so it invalidates the above conclusion.

**Advantages of Non-monotonic reasoning:**

* For real-world systems such as Robot navigation, we can use non-monotonic reasoning.
* In Non-monotonic reasoning, we can choose probabilistic facts or can make assumptions.

**Disadvantages of Non-monotonic Reasoning:**

* In non-monotonic reasoning, the old facts may be invalidated by adding new sentences.
* It cannot be used for theorem **proving**.

**Non-Monotonic reasoning** is a generic name to a class or a specific theory of reasoning. Non-monotonic reasoning attempts to formalize reasoning with incomplete information by classical logic systems.The Non-Monotonic reasoning are of the type

**■ Default reasoning**

**■ Circumscription**

**■ Truth Maintenance Systems**

* **Default Reasoning**

This is a very common from of non-monotonic reasoning. The conclusions are drawn based on what is most likely to be true.

There are two approaches, both are logic type, to Default reasoning : one is Non-monotonic logic and the other is Default logic.

(A logic is **non-monotonic** if some conclusions can be invalidated by adding more knowledge. The logic of definite clauses with negation as failure is non-monotonic. Non-monotonic reasoning is useful for representing defaults. A **default** is a rule that can be used unless it overridden by an exception.

For example, to say that *b* is normally true if *c* is true, a knowledge base designer can write a rule of the form

b ←c ∧ ∼ aba.

where *aba* is an atom that means abnormal with respect to some aspect *a*. Given *c*, the agent can infer *b* unless it is told *aba*. Adding *aba* to the knowledge base can prevent the conclusion of *b*. Rules that imply *aba* can be used to prevent the default under the conditions of the body of the rule.)

**■ Non-monotonic logic**

It has already been defined. It says, "the truth of a proposition may change when new information (axioms) are added and a logic may be build to allows the statement to be retracted."

**Non-monotonic logic is predicate logic with one extension** called modal operator **M** which means “consistent with everything we know”.

The purpose of **M** is to allow consistency. A way to define consistency with PROLOG notation is :

To show that fact **P** is true, we attempt to prove **¬P**.

If we fail we may say that **P** is consistent since **¬P** is false.

**Example :**

∀**x : plays\_instrument(x)** ∧ **M manage(x)** → **jazz\_musician(x)**

States that for all **x**, the **x** plays an instrument and if the fact that **x** can manage is *consistent* with all other knowledge then we can conclude that **x** is a jazz musician.

* **Default logic**

**Default logic** initiates a new inference rule:

tex2html_wrap_inline7448

where

**A** is known as the prerequisite,

**B** as the justification, and

**C** as the consequent.

**‡** Read the above inference rule as:

" if **A**, and if it is consistent with the rest of what is known to assume that **B**, then conclude that **C** ".

**‡** The rule says that given the prerequisite, the consequent can be inferred, provided it is consistent with the rest of the data.

**Example** : Rule that "*birds typically fly*" would be represented as

**bird(x) : flies(x)/flies (x)**

which says

" If **x** is a bird and the claim that **x** flies is consistent with what we know, then infer that **x** flies".

**‡** Note : Since, all we know about Tweety is that :

Tweety is a bird, we therefore inferred that Tweety flies.

**‡** The idea behind non-monotonic reasoning is to reason with first

order logic, and if an inference can not be obtained then use the set

of default rules available within the first order formulation.

* **Circumscription**

Circumscription is a non-monotonic logic to formalize the common sense assumption.

Circumscription is a rule of conjecture that allows you to jump to the conclusion that the objects you can show that posses a certain property, *p*, are in fact all the objects that posses that property.

Circumscription can also cope with default reasoning.

Circumscription is a formalized rule of conjecture (guess) that can be used along with the rules of inference of first order logic.

Circumscription involves formulating rules of thumb with "*abnormality*" predicates and then restricting the extension of these predicates, circumscribing them, so that they apply to only those things to which they are currently known.

**■ Example : Take the case of Bird Tweety**

The rule of thumb is that "birds typically fly" is conditional. The predicate *"*Abnormal*"* signifies abnormality with respect to flying ability.

Observe that the rule ∀**x(Bird(x) & ¬ Abnormal(x)** → **Flies))** does not allow us to infer that *"Tweety flies"*, since we do not know that he is abnormal with respect to flying ability.

But if we add axioms which circumscribe the *abnormality predicate* to which they are currently known say *"Bird Tweety*" then the inference can be drawn. This inference is non-monotonic.

**Implementations: Truth Maintenance Systems**

A variety of *Truth Maintenance Systems* (TMS) have been developed as a means of implementing Non-Monotonic Reasoning Systems.

A truth maintenance system maintains consistency in knowledge representation of a knowledge base. The functions of TMS are to :

**■ Provide justifications for conclusions**

When a problem solving system gives an answer to a user's query, an explanation of that answer is required;

**Example** : An advice to a stockbroker is supported by an explanation of the reasons for that advice. This is constructed by the Inference Engine (IE) by tracing the justification of the assertion.

**■ Recognize inconsistencies**

The Inference Engine (IE) may tell the TMS that some sentences are contradictory. Then, TMS may find that all those sentences are believed true, and reports to the IE which can eliminate the inconsistencies by determining the assumptions used and changing them appropriately.

**Example :** A statement that either Abbott, or Babbitt, or Cabot is guilty together with other statements that Abbott is not guilty, Babbitt is not guilty, and Cabot is not guilty, form a contradiction.

**■ Support default reasoning**

In the absence of any firm knowledge, in many situations we want to reason from default assumptions.

**Example :** If "Tweety is a bird", then until told otherwise, assume that "Tweety flies" and for justification use the fact that "Tweety is a bird" and the assumption that "birds fly".

Basically TMSs:

* all do some form of dependency directed backtracking
* assertions are connected via a network of dependencies.

Reasoning Maintenance System (RMS) is a critical part of a reasoning system. Its purpose is to assure that inferences made by the reasoning system (RS) are valid.

The RS provides the RMS with information about each inference it performs, and in return the RMS provides the RS with information about the whole set of inferences. Several implementations of RMS have been proposed for non-monotonic reasoning. The important ones are the :

􀂃 Truth Maintenance Systems (TMS) and

􀂃 Assumption-based Truth Maintenance Systems (ATMS).

The TMS maintains the consistency of a knowledge base as soon as new knowledge is added. It considers only one state at a time so it is not possible to manipulate environment.

The ATMS is intended to maintain multiple environments.

There are three types of TMS

1. Justification-Based Truth Maintenance Systems (JTMS)
2. Logic-Based Truth Maintenance Systems (LTMS)
3. Assumption-Based Truth Maintenance Systems (ATMS)

**Justification-Based Truth Maintenance Systems (JTMS)**

* This is a simple TMS in that it does not know anything about the structure of the assertions themselves.
* Each supported belief (assertion) in has a justification.
* Each justification has two parts:
  + An *IN-List* -- which supports beliefs held.
  + An *OUT-List* -- which supports beliefs *not* held.
* An assertion is connected to its justification by an arrow.
* One assertion can *feed* another justification thus creating the network.
* Assertions may be labelled with a *belief status*.
* An assertion is *valid* if every assertion in the IN-List is believed and none in the OUT-List are believed.
* An assertion is non-monotonic is the OUT-List is not empty or if any assertion in the IN-List is non-monotonic.

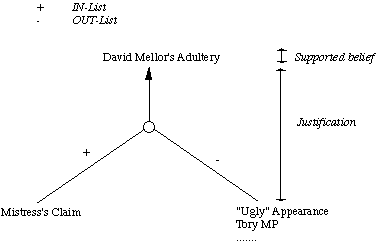


Fig.  A JTMS Assertion

**Logic-Based Truth Maintenance Systems (LTMS)**

Similar to JTMS except:

* Nodes (assertions) assume no relationships among them except ones explicitly stated in justifications.
* JTMS can represent P and tex2html_wrap_inline7182P simultaneously. An LTMS would throw a contradiction here.
* If this happens network has to be reconstructed.

**Assumption-Based Truth Maintenance Systems (ATMS)**

* JTMS and LTMS pursue a single line of reasoning at a time and backtrack (dependency-directed) when needed -- *depth first search*.
* ATMS maintain alternative paths in parallel -- *breadth-first search*
* Backtracking is avoided at the expense of maintaining multiple contexts.
* However as reasoning proceeds contradictions arise and the ATMS can be *pruned*
  + Simply find assertion with no valid justification.

**Statistical reasoning**

|  |
| --- |
| *Statistical reasoning* |
|  | Statistical reasoning is the way people reason with statistical ideas and make sense of statistical information. Statistical reasoning may involve connecting one concept to another (e.g., center and spread) or may combine ideas about data and chance. Reasoning means understanding and being able to explain statistical processes, and being able to fully interpret statistical results. To read more about statistical reasoning see Garfield (2002). |

**Symbolic versus statistical reasoning**

The (Symbolic) methods basically represent uncertainty belief as being

* True,
* False, *or*
* Neither True nor False.

Some methods also had problems with

* Incomplete Knowledge
* Contradictions in the knowledge.

Statistical methods provide a method for representing beliefs that are not certain (or uncertain) but for which there may be some supporting (or contradictory) evidence.

Statistical methods offer advantages in two broad scenarios:

**Genuine Randomness**

-- Card games are a good example. We may not be able to predict any outcomes with certainty but we have knowledge about the likelihood of certain items (*e.g.* like being dealt an ace) and we can exploit this.

**Exceptions**

-- Symbolic methods can represent this. However if the number of exceptions is large such system tend to break down. Many common sense and expert reasoning tasks for example. Statistical techniques can *summarise* large exceptions without resorting enumeration.

In the logic based approaches described, we have assumed that everything is either believed false or believed true.

However, it is often useful to represent the fact that we believe such that something is probably true, or true with probability (say) 0.65.

This is useful for dealing with problems where there is randomness and unpredictability (such as in games of chance) and also for dealing with problems where we could, if we had sufficient information, work out exactly what is true.

To do all this in a principled way requires techniques for probabilistic reasoning.

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic reasoning, let's understand some common terms:

**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.

1. 0 ≤ P(A) ≤ 1,   where P(A) is the probability of an event A.
2. P(A) = 0,  indicates total uncertainty in an event A.
3. P(A) =1, indicates total certainty in an event A.

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

C:\Users\Dr.JENA\Desktop\probabilistic.png

* P(¬A) = probability of a not happening event.
* P(¬A) + P(A) = 1.

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

**Sample space:** The collection of all possible events is called sample space.

**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.

**■ Probability Experiment :**

Process which leads to well-defined results call outcomes.

**■ Independent Events :**

Two events, **E1** and **E2**, are independent if the fact that **E1** occurs does not affect the probability of **E2** occurring.

**■ Mutually Exclusive Events :**

Events **E1**, **E2**, ..., **En** are said to be mutually exclusive if the occurrence of any one of them automatically implies the non-occurrence of the remaining **n − 1** events.

**■ Disjoint Events :**

Another name for mutually exclusive events.

**Conditional probability:**

Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

C:\Users\Dr.JENA\Desktop\probabilistic2.png

**Where P(*A*⋀*B*)= Joint probability of a and B**

**P(B)= Marginal probability of B.**

If the probability of A is given and we need to find the probability of B, then it will be given as:

C:\Users\Dr.JENA\Desktop\probabilistic3.png

It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of **P(A⋀*B*) by P( B )**.

**Example:**

In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

**Solution:**

Let, A is an event that a student likes Mathematics

B is an event that a student likes English.

C:\Users\Dr.JENA\Desktop\probabilistic5.png

**Hence, 57% are the students who like English also like Mathematics.**

**■ Joint probability :**

The probability of two events in conjunction. It is the probability of both events together. The joint probability of **A** and **B** is written **P(A ∩ B)** ; also written as **P(A, B)**.

**■ Marginal Probability :**

The probability of one event, regardless of the other event. The marginal probability of **A** is written **P(A)**, and the marginal probability of **B** is written **P(B)**.

**Mutually Exclusive Events (disjoint) :** means nothing in common Two events are mutually exclusive if they cannot occur at the same time.

(a) If two events are mutually exclusive, then probability of both occurring at same time is **P(A and B) = 0**

(b) If two events are mutually exclusive , then the probability of either occurring is **P(A or B) = P(A) + P(B)**

Given **P(A)= 0.20, P(B)= 0.70**, where **A** and **B** are disjoint then **P(A and B) = 0**

**Non-Mutually Exclusive Events**

The non-mutually exclusive events have some overlap.

When **P(A)** and **P(B)** are added, the probability of the intersection (ie. "and" ) is added twice, so subtract once.

**P(A or B) = P(A) + P(B) - P(A and B)**

**Summary of symbols & notations**

A U B (A union B) 'Either A or B occurs or both occur'

A ∩ B (A intersection B) 'Both A and B occur'

A ⊆ B (A is a subset of B) 'If A occurs, so does B'

A' Ā 'Event A does not occur'

Φ (the empty set) An impossible event

S (the sample space) An event that is certain to occur

A ∩ B = Φ Mutually exclusive Events

P(A) Probability that event A occurs

P(B) Probability that event B occurs

P(A U B) Probability that event A or event B occurs

P(A ∩ B) Probability that event A and event B occur

P(A ∩ B) = P(A) . P(B) Independent events

P(A ∩ B) = 0 Mutually exclusive Events

P(A U B) = P(A) + P(B) – P(AB) Addition rule;

P(A U B) = P(A) + P(B) – P(A) . P(B)

P(A U B) = P(A) + P(B) – P(A ∩ B)

P(A U B) = P(A) + P(B) – P(B|A).P(A)

Addition rule; independent events

P(A U B) = P(A) + P(B) Addition rule; mutually exclusive Events

A|B (A given B) "Event A will occur given that event B has occurred"

P(A|B) Conditional probability that event A will occur given that event B has occurred

already

P(B|A) Conditional probability that event B will occur given that event A has occurred

already

P(A ∩ B) = P(A|B).P(B) or

P(A ∩ B) = P(B|A).P(A)

Multiplication rule

P(A ∩ B) = P(A) . P(B) Multiplication rule; independent events;

ie probability of joint events A and B

P(A|B) = P(A ∩ B) / P(B) Rule to determine a conditional probability from unconditional probabilities.

**Bayes' theorem**

Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which determines the probability of an event with uncertain knowledge.

In probability theory, it relates the conditional probability and marginal probabilities of two random events.

The **Bayesian inference** is an application of Bayes' theorem, which is fundamental to Bayesian statistics.

It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.

**Example**: If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancer more accurately with the help of age.

Bayes' theorem can be derived using product rule and conditional probability of event A with known event B:

As from product rule we can write:

1. P(A ⋀ B)= P(A|B) P(B) or

Similarly, the probability of event B with known event A:

1. P(A ⋀ B)= P(B|A) P(A)

Equating right hand side of both the equations, we will get:

C:\Users\Dr.JENA\Desktop\bayes-theorem.png

The above equation (a) is called as **Bayes' rule** or**Bayes' theorem**. This equation is basic of most modern AI systems for **probabilistic inference**.

It shows the simple relationship between joint and conditional probabilities. Here,

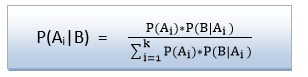
P(A|B) is known as **posterior**, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.

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 the equation (a), in general, we can write P (B) = P(A)\*P(B|Ai), hence the Bayes' rule can be written as:



Where A1, A2, A3,........, An is a set of mutually exclusive and exhaustive events.

**Applying Bayes' rule:**

Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|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:

C:\Users\Dr.JENA\Desktop\bayes-theorem3.png

**Example-1:**

**Question: 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|b) = 0.8

P(b) = 1/30000

P(a)= .02

C:\Users\Dr.JENA\Desktop\bayes-theorem4.png

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

**Example-2:**

**Question: From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King|Face), which means the drawn face card is a king card.**

**Solution:**

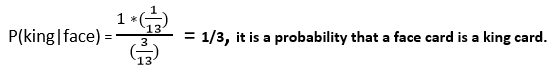
C:\Users\Dr.JENA\Desktop\bayes-theorem5.png

P(king): probability that the card is King= 4/52= 1/13

P(face): probability that a card is a face card= 3/13

P(Face|King): probability of face card when we assume it is a king = 1

Putting all values in equation (i) we will get:



**Example-3**

**Problem :** Marie's marriage is tomorrow, in recent years, each year it has rained only 5 days.

The weatherman has predicted rain for tomorrow, when it actually rains, the weatherman correctly forecasts rain 90% of the time. When it doesn't rain, the weatherman incorrectly forecasts rain 10% of the time.

**The question :** What is the probability that it will rain on the day of Marie's wedding?

**Solution :** The sample space is defined by two mutually exclusive events – *"*it rains" or *"*it does not rain*"*. Additionally, a third event occurs when the "weatherman predicts rain".

The events and probabilities are stated below.

**◊** Event A1 : rains on Marie's wedding.

**◊** Event A2 : does not rain on Marie's wedding

**◊** Event B : weatherman predicts rain.

**◊** P(A1) = 5/365 =0.0136985 [Rains 5 days in a year.]

**◊** P(A2) = 360/365 = 0.9863014 [Does not rain 360 days in a year.]

**◊** P(B|A1) = 0.9 [When it rains, the weatherman predicts rain 90% time.]

**◊** P(B|A2) = 0.1 [When it does not rain, weatherman predicts rain 10% time.]

We want to know **P(A1|B)**, the probability that it will rain on the day of Marie's wedding, given a forecast for rain by the weatherman.

The answer can be determined from Bayes' theorem, shown below.

**P(A1).P(B|A1) (0.014)(0.9)**

**P(A1|B) = P(A1).P(B|A1)+P(A2).P(B|A2) [(0.014)(0.9)+(0.986)(0.1)]**

**= 0.111**

So, despite the weatherman's prediction, there is a good chance that Marie will not get rain on at her wedding.

**Thus Bayes theorem is used to calculate conditional probabilities**

**Application of Bayes' theorem in Artificial intelligence:**

**Following are some applications of Bayes' theorem:**

* It is used to calculate the next step of the robot when the already executed step is given.
* Bayes' theorem is helpful in weather forecasting.
* It can solve the Monty Hall problem.

Thus Simple Bayes rule-based systems are not suitable for uncertain reasoning.

* Knowledge acquisition is very hard.
* Too many probabilities needed -- too large a storage space.
* Computation time is too large.
* Updating new information is difficult and time consuming.
* Exceptions like ``none of the above'' cannot be represented.
* Humans are not very good probability estimators.

However, Bayesian statistics still provide the core to reasoning in many uncertain reasoning systems with suitable enhancement to overcome the above problems.

We will look at three broad categories:

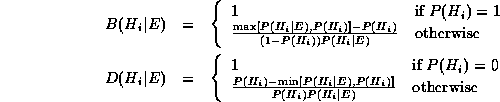
* Certainty factors,
* Dempster-Shafer models,
* Bayesian networks.

**Belief Models and Certainty Factors**

This approach has been suggested by Shortliffe and Buchanan and used in their famous medical diagnosis MYCIN system.

MYCIN is essentially and *expert system*. Here we only concentrate on the probabilistic reasoning aspects of MYCIN.

* MYCIN represents knowledge as a set of rules.
* Associated with each rule is a *certainty factor*
* A certainty factor is based on measures of belief *B* and disbelief *D* of an hypothesis tex2html_wrap_inline7536 given evidence *E* as follows:



where tex2html_wrap_inline7566 is the standard probability.

* The certainty factor *C* of some hypothesis tex2html_wrap_inline7536 given evidence*E* is defined as:  
  displaymath1566

**Rule based systems**

**■** A rule is an expression of the form **"if A then B"** where **A** is an assertion and **B** can be either an action or another assertion.

**Example** : Trouble shooting of water pumps

1. If pump failure then the pressure is low

2. If pump failure then check oil level

3. If power failure then pump failure

**■** Rule based system consists of a library of such rules.

**■** Rules reflect essential relationships within the domain.

**■** Rules reflect ways to reason about the domain.

**■** Rules draw conclusions and points to actions, when specific information about the domain comes in. This is called inference.

**■** The inference is a kind of chain reaction like : If there is a power failure then (see rules 1, 2, 3 mentioned above)

Rule 3 states that there is a pump failure, and

Rule 1 tells that the pressure is low, and

Rule 2 gives a (useless) recommendation to check the oil level.

■ It is very difficult to control such a mixture of inference back and forth in the same session and resolve such uncertainties

How to deal uncertainties in rule based system?

A problem with rule-based systems is that often the connections reflected by the rules are not absolutely certain (i.e. deterministic), and the gathered information is often subject to uncertainty.

In such cases, a certainty measure is added to the premises as well as the conclusions in the rules of the system.

A rule then provides a function that describes : how much a change in the certainty of the premise will change the certainty of the conclusion. In its simplest form, this looks like :

**If A (with certainty x) then B (with certainty f(x))**

**Dempster-Shafer Theory**

DST is a mathematical **theory of evidence** based on belief functions and plausible reasoning. It is used to combine separate pieces of information (evidence) to calculate the probability of an event.

DST offers an alternative to traditional probabilistic theory for the mathematical representation of uncertainty.

DST can be regarded as, a more general approach to represent uncertainty than the Bayesian approach.

Bayesian methods are sometimes inappropriate

**Example :**

Let **A** represent the proposition **"Moore is attractive"**.

Then the axioms of probability insist that **P(A) + P(¬A) = 1**.

Now suppose that Andrew does not even know who **"Moore"** is, then

**‡** We cannot say that Andrew believes the proposition if he has no

idea what it means.

**‡** Also, it is not fair to say that he disbelieves the proposition.

**‡** It would therefore be meaningful to denote Andrew's belief **B** of

**B(A)** and **B(¬A)** as both being **0**.

**‡** Certainty factors do not allow this.

This can be regarded as a more general approach to representing uncertainty than the Bayesian approach.

Bayesian methods are sometimes inappropriate:

[ The **theory of belief functions**, also referred to as **evidence theory** or **Dempster–Shafer theory** (**DST**), is a general framework for reasoning with uncertainty, with understood connections to other frameworks such as probability, possibility and [imprecise probability theories](https://en.wikipedia.org/wiki/Imprecise_probability). First introduced by [Arthur P. Dempster](https://en.wikipedia.org/wiki/Arthur_P._Dempster)[[1]](https://en.wikipedia.org/wiki/Dempster%E2%80%93Shafer_theory#cite_note-1) in the context of statistical inference, the theory was later developed by [Glenn Shafer](https://en.wikipedia.org/wiki/Glenn_Shafer) into a general framework for modeling epistemic uncertainty—a mathematical theory of [evidence](https://en.wikipedia.org/wiki/Evidence).[[2]](https://en.wikipedia.org/wiki/Dempster%E2%80%93Shafer_theory#cite_note-SH76-2)[[3]](https://en.wikipedia.org/wiki/Dempster%E2%80%93Shafer_theory#cite_note-3) The theory allows one to combine evidence from different sources and arrive at a degree of belief (represented by a mathematical object called *belief function*) that takes into account all the available evidence.

n this formalism a **degree of belief** (also referred to as a **mass**) is represented as a **belief function** rather than a [Bayesian](https://en.wikipedia.org/wiki/Bayesianism) [probability distribution](https://en.wikipedia.org/wiki/Probability_distribution). Probability values are assigned to *sets* of possibilities rather than single events: their appeal rests on the fact they naturally encode evidence in favor of propositions.

For example, suppose we have a belief of 0.5 and a plausibility of 0.8 for a proposition, say “the cat in the box is dead.” This means that we have evidence that allows us to state strongly that the proposition is true with a confidence of 0.5. However, the evidence contrary to that hypothesis (i.e. “the cat is alive”) only has a confidence of 0.2. The remaining mass of 0.3 (the gap between the 0.5 supporting evidence on the one hand, and the 0.2 contrary evidence on the other) is “indeterminate,” meaning that the cat could either be dead or alive. This interval represents the level of uncertainty based on the evidence in the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hypothesis** | **Mass** | **Belief** | **Plausibility** |
| Null (neither alive nor dead) | 0 | 0 | 0 |
| Alive | 0.2 | 0.2 | 0.5 |
| Dead | 0.5 | 0.5 | 0.8 |
| Either (alive or dead) | 0.3 | 1.0 | 1.0 |

Beliefs from different sources can be combined with various fusion operators to model specific situations of belief fusion,

## Dempster–Shafer as a generalisation of Bayesian theory[[edit](https://en.wikipedia.org/w/index.php?title=Dempster%E2%80%93Shafer_theory&action=edit&section=10)]

As in Dempster–Shafer theory, a Bayesian belief function {\displaystyle \operatorname {bel} :2^{X}\rightarrow [0,1]\,\!} has the properties {\displaystyle \operatorname {bel} (\emptyset )=0} and {\displaystyle \operatorname {bel} (X)=1}. The third condition, however, is subsumed by, but relaxed in DS theory:[[2]](https://en.wikipedia.org/wiki/Dempster%E2%80%93Shafer_theory#cite_note-SH76-2):p. 19

{\displaystyle {\text{If }}A\cap B=\emptyset ,{\text{ then}}\operatorname {bel} (A\cup B)=\operatorname {bel} (A)+\operatorname {bel} (B).}

For example, a Bayesian would model the color of a car as a probability distribution over (red, green, blue), assigning one number to each color. Dempster–Shafer would assign numbers to each of (red, green, blue, (red or green), (red or blue), (green or blue), (red or green or blue)) which do not have to cohere, for example Bel(red)+Bel(green) != Bel(red or green). This may be computationally more efficient if a witness reports "I saw that the car was either blue or green" in which case the belief can be assigned in a single step rather than breaking down into values for two separate colors. However this can lead to irrational conclusions.

Equivalently, each of the following conditions defines the Bayesian special case of the DS theory:[[2]](https://en.wikipedia.org/wiki/Dempster%E2%80%93Shafer_theory#cite_note-SH76-2):p. 37,45

* {\displaystyle \operatorname {bel} (A)+\operatorname {bel} ({\bar {A}})=1{\text{ for all }}A\subseteq X.}
* For finite *X*, all focal elements of the belief function are singletons.

Bayes' conditional probability is a special case of Dempster's rule of combination

**Dempster-Shafer Calculus**

The basic idea in representing uncertainty in this model is:

* Set up a confidence interval -- an interval of probabilities within which the true probability lies with a certain confidence -- based on the Belief *B* and plausibility *PL* provided by some evidence *E* for a proposition *P*.
* The belief brings together all the evidence that would lead us to believe in *P* with some certainty.
* The plausibility brings together the evidence that is compatible with *P* and is not inconsistent with it.
* This method allows for further additions to the set of knowledge and does not assume disjoint outcomes.

If tex2html_wrap_inline7598 is the set of possible outcomes, then a *mass probability*, *M*, is defined for each member of the set tex2html_wrap_inline7602 and takes values in the range [0,1].

The Null set, tex2html_wrap_inline7606, is also a member of tex2html_wrap_inline7602.

**NOTE:** This deals wit set theory terminology that will be dealt with in a tutorial shortly. Also see exercises to get experience of problem solving in this important subject matter.

M is a *probability density function* defined not just for tex2html_wrap_inline7598 but for **em all** subsets.

So if tex2html_wrap_inline7598 is the set { *Flu (F), Cold (C), Pneumonia (P)* } then tex2html_wrap_inline7602 is the set { tex2html_wrap_inline7606, {*F*}, {*C*}, {*P*}, {*F*, *C*}, {*F*, *P*}, {*C*, *P*}, {*F*, *C*, *P*} }

* The confidence interval is then defined as [*B*(*E*),*PL*(*E*)]  
  where  
  displaymath1649  
  where tex2html_wrap_inline7640 *i.e.* all the evidence that makes us believe in the correctness of *P*, and  
  eqnarray1652  
  where tex2html_wrap_inline7644 *i.e.* all the evidence that contradicts *P*.

**Bayesian networks**

A Bayesian network (or a belief network) is a probabilistic graphical model that represents a set of variables and their probabilistic independencies.

For example, a Bayesian network could represent the **probabilistic relationships** between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

Bayesian Networks are also called : Bayes nets, Bayesian Belief Networks (BBNs) or simply Belief Networks. Causal Probabilistic Networks (CPNs). Initially developed by Pearl (1988).

A Bayesian network consists of :

􀂃 a set of nodes and a set of directed edges between nodes.

􀂃 the edges reflect cause-effect relations within the domain.

􀂃 The effects are not completely deterministic (e.g. disease -> symptom).

􀂃 the strength of an effect is modeled as a probability.

The basic idea is:

* Knowledge in the world is *modular* -- most events are conditionally independent of most other events.
* Adopt a model that can use a more local representation to allow interactions between events that *only* affect each other.
* Some events may only be *unidirectional* others may be *bidirectional* -- make a distinction between these in model.
* Events may be causal and thus get chained together in a network.

**Implementation**

* A *Bayesian Network* is a *directed acyclic graph*:
  + A graph where the directions are links which indicate dependencies that exist between nodes.
  + Nodes represent propositions about events or events themselves.
  + Conditional probabilities quantify the strength of dependencies.

Consider the following example:

* The probability, tex2html_wrap_inline7674 that my car won't start.
* If my car won't start then it is likely that
  + The battery is flat or
  + The staring motor is broken.

In order to decide whether to fix the car myself or send it to the garage I make the following decision:

* If the headlights do not work then the battery is likely to be flat so i fix it myself.
* If the starting motor is defective then send car to garage.
* If battery and starting motor both gone send car to garage.

The network to represent this is as follows:

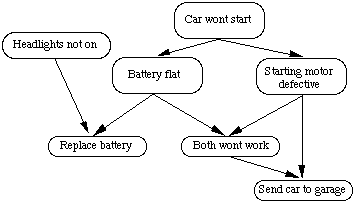


Fig. [21](https://users.cs.cf.ac.uk/Dave.Marshall/AI2/node95.html#figbnet1) A simple Bayesian network

**Reasoning in Bayesian nets**

* Probabilities in links obey standard conditional probability axioms.
* Therefore follow links in reaching hypothesis and update beliefs accordingly.
* A few broad classes of algorithms have bee used to help with this:
  + Pearls's message passing method.
  + Clique triangulation.
  + Stochastic methods.
  + Basically they all take advantage of *clusters* in the network and use their limits on the influence to constrain the search through net.
  + They also ensure that probabilities are updated correctly.
* Since information is *local* information can be readily added and deleted with minimum effect on the whole network. **ONLY** affected nodes need updating.

**Unit -IV**

**Mini-Max Algorithm**

* Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
* Mini-Max algorithm uses recursion to search through the game-tree.
* Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
* In this algorithm two players play the game, one is called MAX and other is called MIN.
* Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
* Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
* The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
* The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Working of Min-Max Algorithm:

* The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
* In this example, there are two players one is called Maximizer and other is called Minimizer.
* Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
* This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
* At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

**Step-1:** In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



**Step 2:** Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

* For node D         max(-1,- -∞) => max(-1,4)= 4
* For Node E         max(2, -∞) => max(2, 6)= 6
* For Node F         max(-3, -∞) => max(-3,-5) = -3
* For node G         max(0, -∞) = max(0, 7) = 7



**Step 3:** In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

* For node B= min(4,6) = 4
* For node C= min (-3, 7) = -3



**Step 3:** Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

* For node A max(4, -3)= 4



That was the complete workflow of the minimax two player game.

**Properties of Mini-Max algorithm:**

* **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
* **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
* **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **O(bm)**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
* **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

**Limitation of the mini-max Algorithm**:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from **alpha-beta pruning.**

**Alpha-Beta Cutoffs(Pruning)**

* Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
* As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**. This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
* Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
* The two-parameter can be defined as:
  1. **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
  2. **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.
* The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

Note: To better understand this topic, kindly study the minimax algorithm.

**Condition for Alpha-beta pruning:**

The main condition which required for alpha-beta pruning is:

1. α>=β

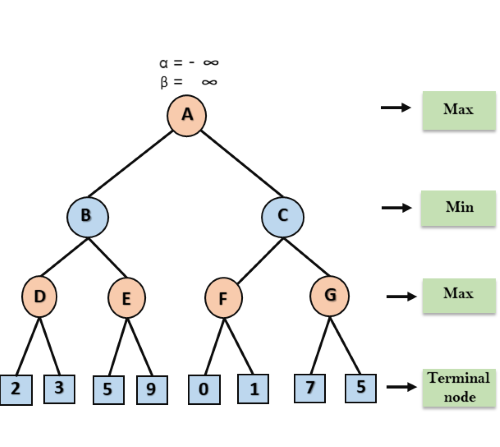
**Key points about alpha-beta cutoffs:**

* The Max player will only update the value of alpha.
* The Min player will only update the value of beta.
* While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
* We will only pass the alpha, beta values to the child nodes.

**Working of Alpha-Beta Pruning:**

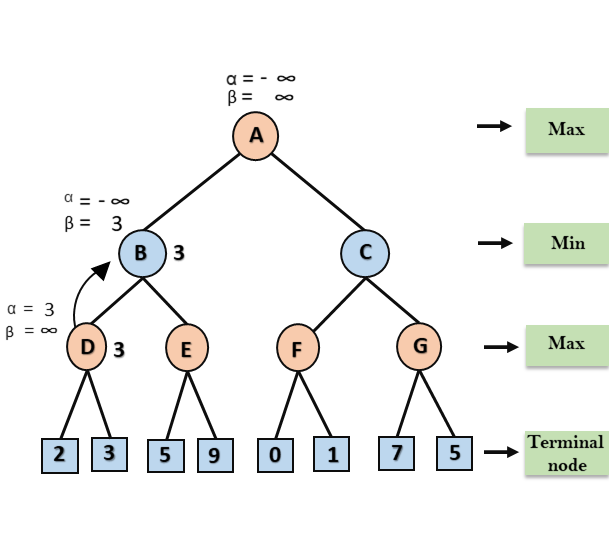
Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

**Step 1:** At the first step the, Max player will start first move from node A where α= -∞ and β= +∞, these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, and Node B passes the same value to its child D.



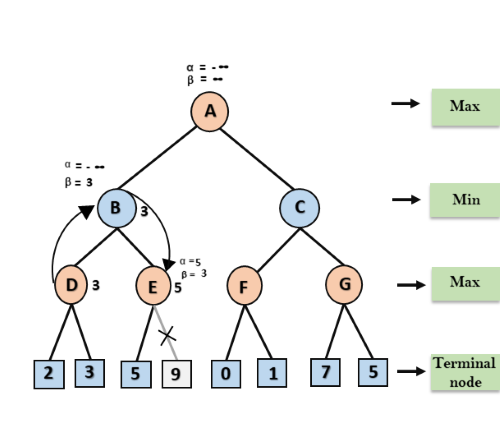
**Step 2:** At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

**Step 3:** Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.



In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.

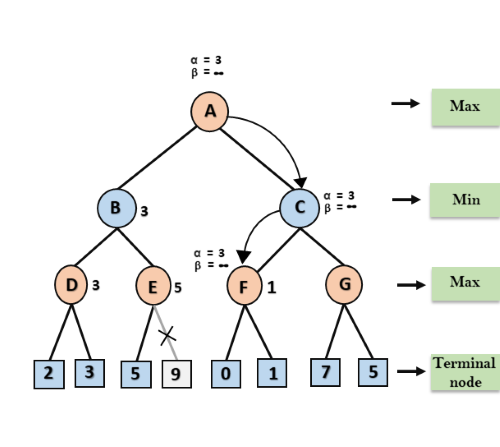
**Step 4:** At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



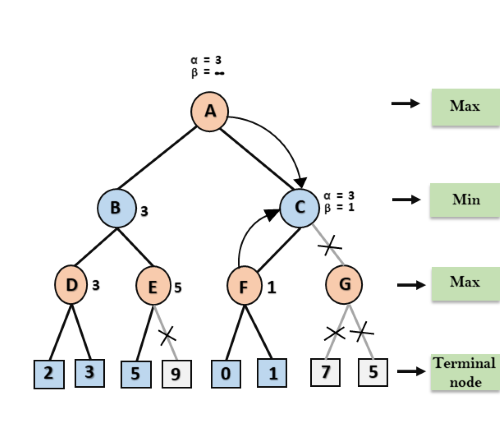
**Step 5:** At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.

At node C, α=3 and β= +∞, and the same values will be passed on to node F.

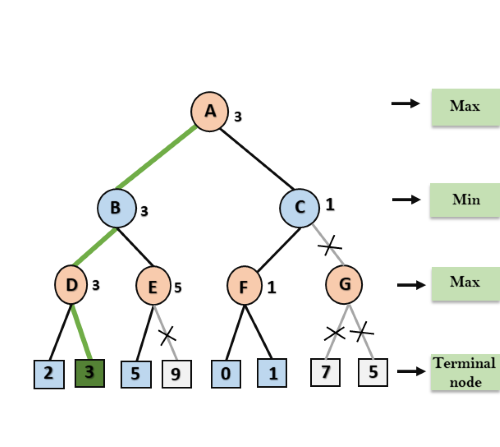
**Step 6:** At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



**Step 7:** Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



**Step 8:** C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



**Move Ordering in Alpha-Beta pruning:**

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning.

It can be of two types:

* **Worst ordering:** In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is O(bm).
* **Ideal ordering:** The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time. Complexity in ideal ordering is O(bm/2).

**Rules to find good ordering:**

Following are some rules to find good ordering in alpha-beta pruning:

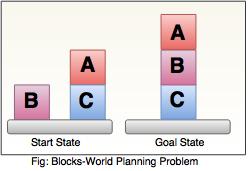
* Occur the best move from the shallowest node.
* Order the nodes in the tree such that the best nodes are checked first.
* Use domain knowledge while finding the best move. Ex: for Chess, try order: captures first, then threats, then forward moves, backward moves.
* We can bookkeep the states, as there is a possibility that states may repeat.

**Planning System**

* The planning in Artificial Intelligence is about the decision making tasks performed by the robots or computer programs to achieve a specific goal.
* The execution of planning is about choosing a sequence of actions with a high likelihood to complete the specific task.

**Blocks-World planning problem**

* The blocks-world problem is known as **Sussman Anomaly.**
* Non interleaved planners of the early 1970s were unable to solve this problem, hence it is considered as anomalous.
* When two subgoals G1 and G2 are given, a noninterleaved planner produces either a plan for G1 concatenated with a plan for G2, or vice-versa.
* In blocks-world problem, three blocks labeled as 'A', 'B', 'C' are allowed to rest on the flat surface. The given condition is that only one block can be moved at a time to achieve the goal.
* The start state and goal state are shown in the following diagram.



**Components of Planning System**

**The planning consists of following important steps:**

* Choose the best rule for applying the next rule based on the best available heuristics.
* Apply the chosen rule for computing the new problem state.
* Detect when a solution has been found.
* Detect dead ends so that they can be abandoned and the system’s effort is directed in more fruitful directions.
* Detect when an almost correct solution has been found.

**Goal stack planning**

This is one of the most important planning algorithms, which is specifically used by **STRIPS.**

Methods which focus on ways of decomposing the original problem into appropriate subparts and on ways of recording. And handling interactions among the subparts as they are detected during the problem-solving process are often called as planning.

Planning refers to the process of computing several steps of a problem-solving procedure before executing any of them

* The stack is used in an algorithm to hold the action and satisfy the goal.  A knowledge base is used to hold the current state, actions.
* Goal stack is similar to a node in a search tree, where the branches are created if there is a choice of an action.

**The important steps of the algorithm are as stated below:**  
  
**i.** Start by pushing the original goal on the stack. Repeat this  until the stack becomes empty. If stack top is a compound goal, then push its unsatisfied subgoals on the stack.  
**ii.** If stack top is a single unsatisfied goal then, replace it by an action and push the action’s precondition on the stack to satisfy the condition.  
**iii.** If stack top is an action, pop it from the stack, execute it and change the knowledge base by the effects of the action.  
**iv.** If stack top is a satisfied goal, pop it from the stack.

**Non-linear planning**

This planning is used to set a goal stack and is included in the search space of all possible subgoal orderings. It handles the goal interactions by interleaving method.  
  
**Advantage of non-Linear planning**

Non-linear planning may be an optimal solution with respect to plan length (depending on search strategy used).  
  
**Disadvantages of Nonlinear planning**

* It takes larger search space, since all possible goal orderings are taken into consideration.
* Complex algorithm to understand.

**Algorithm**  
1. Choose a goal 'g' from the goalset  
2. If 'g' does not match the state, then

* Choose an operator 'o' whose add-list matches goal g
* Push 'o' on the opstack
* Add the preconditions of 'o' to the goalset

3. While all preconditions of operator on top of opstack are met in state

* Pop operator o from top of opstack
* state = apply(o, state)
* plan = [plan; o]

**Hierarchical Planning**

In order to solve hard problems, a problem solver may have to generate long plans.

• It is important to be able to eliminate some of the details of the problem until a solution

• that addresses the main issues is found. Then an attempt can make to fill in the appropriate details.

• Early attempts to do this involved the use of macro operators, in which larger operators

• were built from smaller ones. In this approach, no details eliminated from actual descriptions of the operators.

**ABSTRIPS**

A better approach developed in ABSTRIPS systems which actually planned in a hierarchy of

abstraction spaces, in each of which preconditions at a lower level of abstraction ignored.

ABSTRIPS approach is as follows:

First solve the problem completely, considering only preconditions whose criticality value is the highest possible.

These values reflect the expected difficulty of satisfying the precondition.

To do this, do exactly what STRIPS did, but simply ignore the preconditions of lower than peak criticality.

Once this done, use the constructed plan as the outline of a complete plan and consider

preconditions at the next-lowest criticality level.

Augment the plan with operators that satisfy those preconditions.

Because this approach explores entire plans at one level of detail before it looks at the lower-level details of any one of them, it has called length-first approach.

The assignment of appropriate criticality value is crucial to the success of this hierarchical

planning method.

Those preconditions that no operator can satisfy are clearly the most critical.

Example, solving a problem of moving the robot, for applying an operator, PUSH-THROUGH

DOOR, the precondition that there exist a door big enough for the robot to get through is of high

criticality since there is nothing we can do about it if it is not true.

**Natural Language Processing**

Natural Language Processing (NLP) refers to AI method of communicating with an intelligent systems using a natural language such as English.

Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.

The field of NLP involves making computers to perform useful tasks with the natural languages humans use. The input and output of an NLP system can be −

* Speech
* Written Text

## Components of NLP

There are two components of NLP as given −

### Natural Language Understanding (NLU)

Understanding involves the following tasks −

* Mapping the given input in natural language into useful representations.
* Analyzing different aspects of the language.

### Natural Language Generation (NLG)

It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.

It involves −

* **Text planning** − It includes retrieving the relevant content from knowledge base.
* **Sentence planning** − It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
* **Text Realization** − It is mapping sentence plan into sentence structure.

The NLU is harder than NLG.

## Difficulties in NLU

NL has an extremely rich form and structure.

It is very ambiguous. There can be different levels of ambiguity −

* **Lexical ambiguity** − It is at very primitive level such as word-level.
* For example, treating the word “board” as noun or verb?
* **Syntax Level ambiguity** − A sentence can be parsed in different ways.
* For example, “He lifted the beetle with red cap.” − Did he use cap to lift the beetle or he lifted a beetle that had red cap?
* **Referential ambiguity** − Referring to something using pronouns. For example, Rima went to Gauri. She said, “I am tired.” − Exactly who is tired?
* One input can mean different meanings.
* Many inputs can mean the same thing.

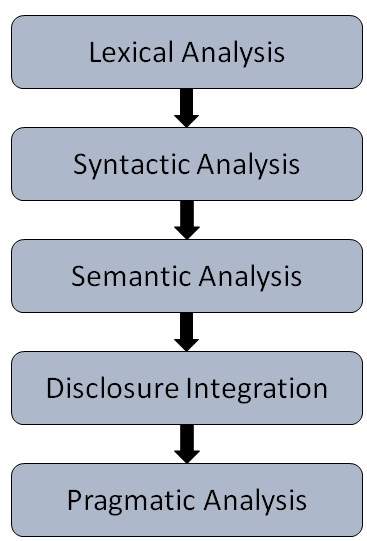
## NLP Terminology

* **Phonology** − It is study of organizing sound systematically.
* **Morphology** − It is a study of construction of words from primitive meaningful units.
* **Morpheme** − It is primitive unit of meaning in a language.
* **Syntax** − It refers to arranging words to make a sentence. It also involves determining the structural role of words in the sentence and in phrases.
* **Semantics** − It is concerned with the meaning of words and how to combine words into meaningful phrases and sentences.
* **Pragmatics** − It deals with using and understanding sentences in different situations and how the interpretation of the sentence is affected.
* **Discourse** − It deals with how the immediately preceding sentence can affect the interpretation of the next sentence.
* **World Knowledge** − It includes the general knowledge about the world.

## Steps in NLP

There are general five steps −

* **Lexical Analysis** − It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of txt into paragraphs, sentences, and words.
* **Syntactic Analysis (Parsing)** − It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as “The school goes to boy” is rejected by English syntactic analyzer.



* **Semantic Analysis** − It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as “hot ice-cream”.
* **Discourse Integration** − The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.
* **Pragmatic Analysis** − During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

## Implementation Aspects of Syntactic Analysis

There are a number of algorithms researchers have developed for syntactic analysis, but we consider only the following simple methods −

* Context-Free Grammar
* Top-Down Parser

Let us see them in detail −

### Context-Free Grammar

It is the grammar that consists rules with a single symbol on the left-hand side of the rewrite rules. Let us create grammar to parse a sentence −

“The bird pecks the grains”

**Articles (DET)** − a | an | the

**Nouns** − bird | birds | grain | grains

**Noun Phrase (NP)** − Article + Noun | Article + Adjective + Noun

= DET N | DET ADJ N

**Verbs** − pecks | pecking | pecked

**Verb Phrase (VP)** − NP V | V NP

**Adjectives (ADJ)** − beautiful | small | chirping

The parse tree breaks down the sentence into structured parts so that the computer can easily understand and process it. In order for the parsing algorithm to construct this parse tree, a set of rewrite rules, which describe what tree structures are legal, need to be constructed.

These rules say that a certain symbol may be expanded in the tree by a sequence of other symbols. According to first order logic rule, if there are two strings Noun Phrase (NP) and Verb Phrase (VP), then the string combined by NP followed by VP is a sentence. The rewrite rules for the sentence are as follows −

**S → NP VP**

**NP → DET N | DET ADJ N**

**VP → V NP**

**Lexocon −**

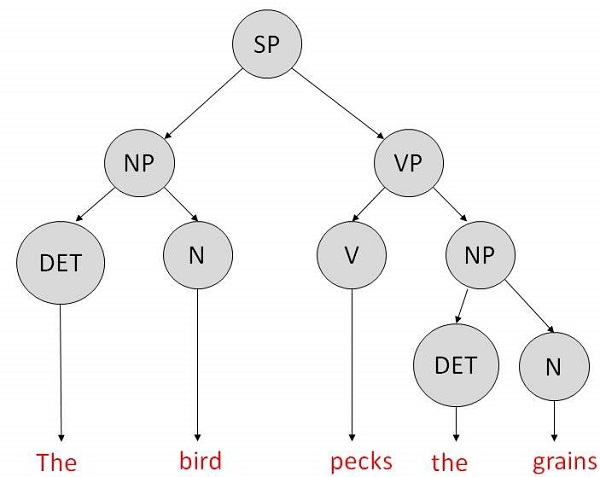
DET → a | the

ADJ → beautiful | perching

N → bird | birds | grain | grains

V → peck | pecks | pecking

The parse tree can be created as shown −



Now consider the above rewrite rules. Since V can be replaced by both, "peck" or "pecks", sentences such as "The bird peck the grains" can be wrongly permitted. i. e. the subject-verb agreement error is approved as correct.

**Merit** − The simplest style of grammar, therefore widely used one.

**Demerits −**

* They are not highly precise. For example, “The grains peck the bird”, is a syntactically correct according to parser, but even if it makes no sense, parser takes it as a correct sentence.
* To bring out high precision, multiple sets of grammar need to be prepared. It may require a completely different sets of rules for parsing singular and plural variations, passive sentences, etc., which can lead to creation of huge set of rules that are unmanageable.

### Top-Down Parser

Here, the parser starts with the S symbol and attempts to rewrite it into a sequence of *terminal symbols* that matches the classes of the words in the input sentence until it consists entirely of terminal symbols.

These are then checked with the input sentence to see if it matched. If not, the process is started over again with a different set of rules. This is repeated until a specific rule is found which describes the structure of the sentence.

**Merit** − It is simple to implement.

**Demerits −**

* It is inefficient, as the search process has to be repeated if an error occurs.
* Slow speed of working.

## Artificial Neural Networks (ANNs)

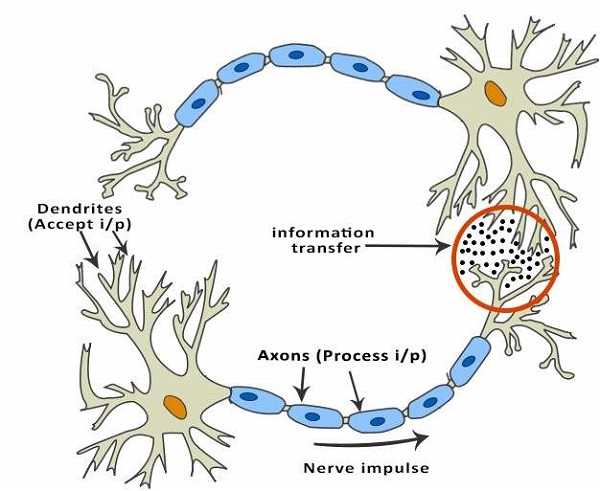
The inventor of the first neurocomputer, Dr. Robert Hecht-Nielsen, defines a neural network as −

"...a computing system made up of a number of simple, highly interconnected processing lements, which process information by their dynamic state response to external inputs.”

## Basic Structure of ANNs

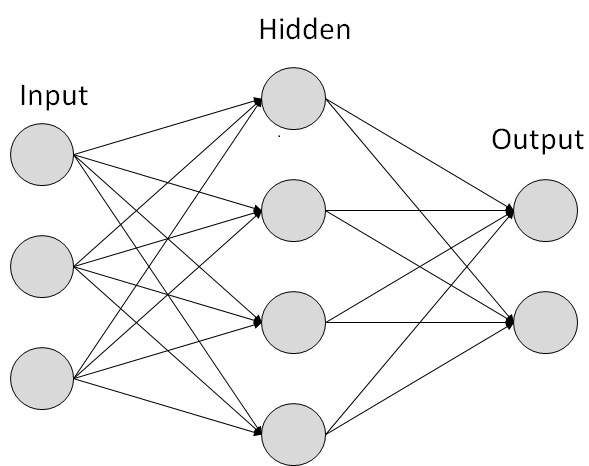
The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living **neurons** and **dendrites**.

The human brain is composed of 86 billion nerve cells called **neurons.** They are connected to other thousand cells by **Axons.** Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.



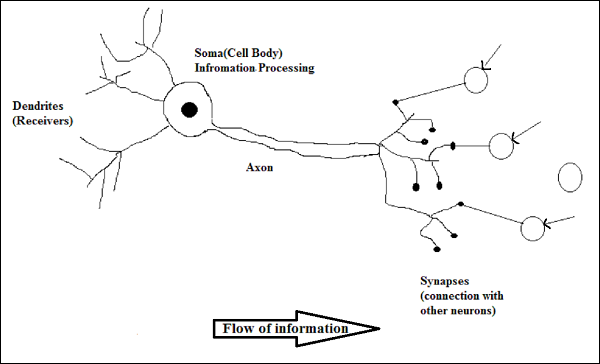
ANNs are composed of multiple **nodes**, which imitate biological **neurons** of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its **activation** or **node value.**

Each link is associated with **weight.** ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple ANN −



## Biological Neuron

A nerve cell neuronneuron is a special biological cell that processes information. According to an estimation, there are huge number of neurons, approximately 1011 with numerous interconnections, approximately 1015.



### Schematic Diagram

### Working of a Biological Neuron

As shown in the above diagram, a typical neuron consists of the following four parts with the help of which we can explain its working −

* **Dendrites** − They are tree-like branches, responsible for receiving the information from other neurons it is connected to. In other sense, we can say that they are like the ears of neuron.
* **Soma** − It is the cell body of the neuron and is responsible for processing of information, they have received from dendrites.
* **Axon** − It is just like a cable through which neurons send the information.
* **Synapses** − It is the connection between the axon and other neuron dendrites.

### ANN versus BNN

Before taking a look at the differences between Artificial Neural Network ANNANN and Biological Neural Network BNNBNN, let us take a look at the similarities based on the terminology between these two.

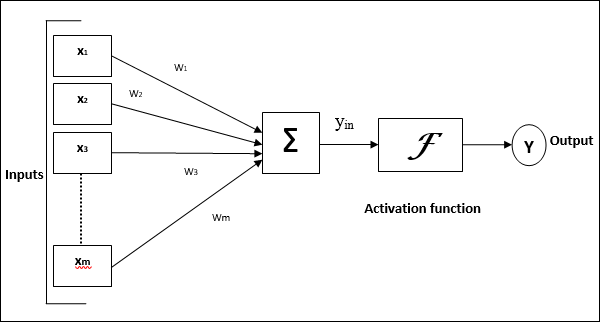
|  |  |
| --- | --- |
| **Biological Neural Network(**BNN) | **Artificial Neural Network**(ANN) |
| Soma | Node |
| Dendrites | Input |
| Synapse | Weights or Interconnections |
| Axon | Output |

The following table shows the comparison between ANN and BNN based on some criteria mentioned.

|  |  |  |
| --- | --- | --- |
| **Criteria** | **BNN** | **ANN** |
| **Processing** | Massively parallel, slow but superior than ANN | Massively parallel, fast but inferior than BNN |
| **Size** | 1011 neurons and 1015 interconnections | 102 to 104 nodes mainlydependsonthetypeofapplicationandnetworkdesignermainlydependsonthetypeofapplicationandnetworkdesigner |
| **Learning** | They can tolerate ambiguity | Very precise, structured and formatted data is required to tolerate ambiguity |
| **Fault tolerance** | Performance degrades with even partial damage | It is capable of robust performance, hence has the potential to be fault tolerant |
| **Storage capacity** | Stores the information in the synapse | Stores the information in continuous memory locations |

## Model of Artificial Neural Network

The following diagram represents the general model of ANN followed by its processing.



For the above general model of artificial neural network, the net input can be calculated as follows −

yin=x1.w1+x2.w2+x3.w3…xm.wmyin=x1.w1+x2.w2+x3.w3…xm.wm

i.e., Net input yin=∑mixi.wiyin=∑imxi.wi

The output can be calculated by applying the activation function over the net input.

Y=F(yin)Y=F(yin)

Output = function netinputcalculated

Processing of ANN depends upon the following three building blocks −

* Network Topology
* Adjustments of Weights or Learning
* Activation Functions

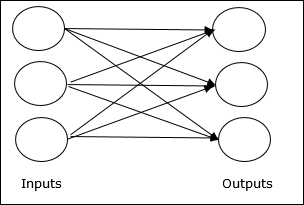
In this chapter, we will discuss in detail about these three building blocks of ANN

## Network Topology

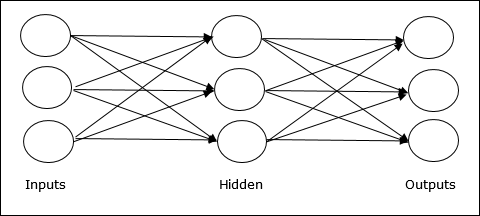
A network topology is the arrangement of a network along with its nodes and connecting lines. According to the topology, ANN can be classified as the following kinds −

### Feedforward Network

* **Single layer feedforward network** − The concept is of feedforward ANN having only one weighted layer. In other words, we can say the input layer is fully connected to the output layer.



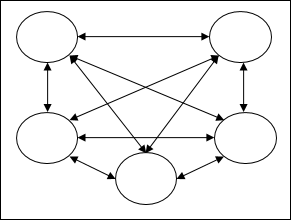
* **Multilayer feedforward network** − The concept is of feedforward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.



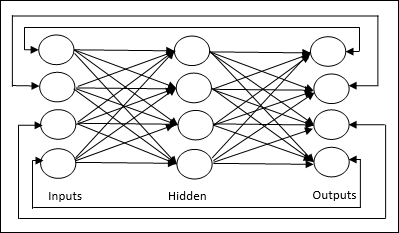
### Feedback Network

As the name suggests, a feedback network has feedback paths, which means the signal can flow in both directions using loops. This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium. It may be divided into the following types −

* **Recurrent networks** − They are feedback networks with closed loops. Following are the two types of recurrent networks.
* **Fully recurrent network** − It is the simplest neural network architecture because all nodes are connected to all other nodes and each node works as both input and output.



* **Jordan network** − It is a closed loop network in which the output will go to the input again as feedback as shown in the following diagram.



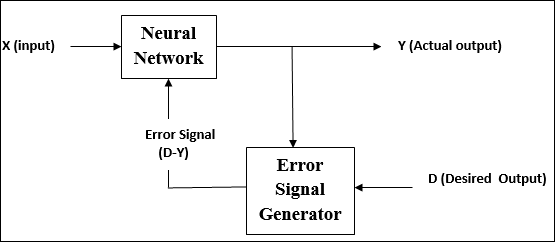
## Adjustments of Weights or Learning/Training

Learning, in artificial neural network, is the method of modifying the weights of connections between the neurons of a specified network. Learning in ANN can be classified into three categories namely supervised learning, unsupervised learning, and reinforcement learning.

### Supervised Learning

As the name suggests, this type of learning is done under the supervision of a teacher. This learning process is dependent.

During the training of ANN under supervised learning, the input vector is presented to the network, which will give an output vector. This output vector is compared with the desired output vector. An error signal is generated, if there is a difference between the actual output and the desired output vector. On the basis of this error signal, the weights are adjusted until the actual output is matched with the desired output.

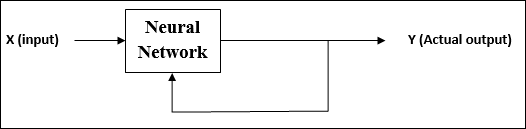


### Unsupervised Learning

As the name suggests, this type of learning is done without the supervision of a teacher. This learning process is independent.

During the training of ANN under unsupervised learning, the input vectors of similar type are combined to form clusters. When a new input pattern is applied, then the neural network gives an output response indicating the class to which the input pattern belongs.

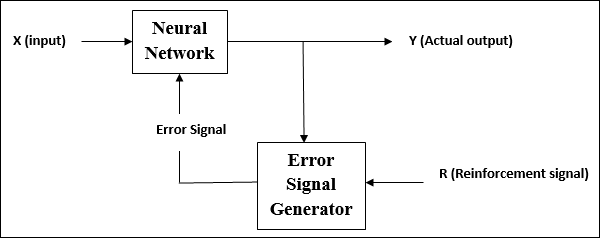
There is no feedback from the environment as to what should be the desired output and if it is correct or incorrect. Hence, in this type of learning, the network itself must discover the patterns and features from the input data, and the relation for the input data over the output.



### Reinforcement Learning

As the name suggests, this type of learning is used to reinforce or strengthen the network over some critic information. This learning process is similar to supervised learning, however we might have very less information.

During the training of network under reinforcement learning, the network receives some feedback from the environment. This makes it somewhat similar to supervised learning. However, the feedback obtained here is evaluative not instructive, which means there is no teacher as in supervised learning. After receiving the feedback, the network performs adjustments of the weights to get better critic information in future.



## Activation Functions

It may be defined as the extra force or effort applied over the input to obtain an exact output. In ANN, we can also apply activation functions over the input to get the exact output. Followings are some activation functions of interest −

### Linear Activation Function

It is also called the identity function as it performs no input editing. It can be defined as −

F(x)=xF(x)=x

### Sigmoid Activation Function

It is of two type as follows −

* **Binary sigmoidal function** − This activation function performs input editing between 0 and 1. It is positive in nature. It is always bounded, which means its output cannot be less than 0 and more than 1. It is also strictly increasing in nature, which means more the input higher would be the output. It can be defined as

F(x)=sigm(x)=11+exp(−x)F(x)=sigm(x)=11+exp(−x)

* **Bipolar sigmoidal function** − This activation function performs input editing between -1 and 1. It can be positive or negative in nature. It is always bounded, which means its output cannot be less than -1 and more than 1. It is also strictly increasing in nature like sigmoid function. It can be defined as

F(x)=sigm(x)=21+exp(−x)−1=1−exp(x)1+exp(x)

## Areas of Application

Followings are some of the areas, where ANN is being used. It suggests that ANN has an interdisciplinary approach in its development and applications.

### Speech Recognition

Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. In the present era, for communication with machines, humans still need sophisticated languages which are difficult to learn and use. To ease this communication barrier, a simple solution could be, communication in a spoken language that is possible for the machine to understand.

Great progress has been made in this field, however, still such kinds of systems are facing the problem of limited vocabulary or grammar along with the issue of retraining of the system for different speakers in different conditions. ANN is playing a major role in this area. Following ANNs have been used for speech recognition −

* Multilayer networks
* Multilayer networks with recurrent connections
* Kohonen self-organizing feature map

The most useful network for this is Kohonen Self-Organizing feature map, which has its input as short segments of the speech waveform. It will map the same kind of phonemes as the output array, called feature extraction technique. After extracting the features, with the help of some acoustic models as back-end processing, it will recognize the utterance.

### Character Recognition

It is an interesting problem which falls under the general area of Pattern Recognition. Many neural networks have been developed for automatic recognition of handwritten characters, either letters or digits. Following are some ANNs which have been used for character recognition −

* Multilayer neural networks such as Backpropagation neural networks.
* Neocognitron

Though back-propagation neural networks have several hidden layers, the pattern of connection from one layer to the next is localized. Similarly, neocognitron also has several hidden layers and its training is done layer by layer for such kind of applications.

### Signature Verification Application

Signatures are one of the most useful ways to authorize and authenticate a person in legal transactions. Signature verification technique is a non-vision based technique.

For this application, the first approach is to extract the feature or rather the geometrical feature set representing the signature. With these feature sets, we have to train the neural networks using an efficient neural network algorithm. This trained neural network will classify the signature as being genuine or forged under the verification stage.

### Human Face Recognition

It is one of the biometric methods to identify the given face. It is a typical task because of the characterization of “non-face” images. However, if a neural network is well trained, then it can be divided into two classes namely images having faces and images that do not have faces.

First, all the input images must be preprocessed. Then, the dimensionality of that image must be reduced. And, at last it must be classified using neural network training algorithm. Following neural networks are used for training purposes with preprocessed image −

* Fully-connected multilayer feed-forward neural network trained with the help of back-propagation algorithm.
* For dimensionality reduction, Principal Component Analysis PCAPCA is used.

## Advantages of Artificial Neural Networks ( ANN)

► **Storing information on the entire network** : Information such as in **traditional programming** is stored on the entire network, not on a database. The disappearance of a few pieces of information in one place does not prevent the network from functioning.

► **Ability to work with** incomplete knowledge **:** After ANN training, the data may produce output even with incomplete information. The loss of performance here depends on the importance of the missing information.

► **Having fault tolerance:** Corruption of one or more cells of ANN does not prevent it from generating output. This feature makes the networks fault tolerant.

► **Having a distributed memory:**In order for ANN to be able to learn, it is necessary to determine the examples and to teach the network according to the desired output by showing these examples to the network. The network's success is directly proportional to the selected instances, and if the event can not be shown to the network in all its aspects, the network can produce false output

► **Gradual corruption:** A network slows over time and undergoes relative degradation. The network problem does not immediately corrode immediately.

► **Ability to make machine learning:** Artificial neural networks learn events and make decisions by commenting on similar events.

► **Parallel processing capability:** Artificial neural networks have numerical strength that can perform more than one job at the same time.

## Disadvantages of Artificial Neural Networks (ANN)

► **Hardware dependence:** Artificial neural networks require processors with parallel processing power, in accordance with their structure. For this reason, the realization of the equipment is dependent.

► **Unexplained behavior of the network:** This is the most important problem of ANN. When ANN produces a probing solution, it does not give a clue as to why and how. This reduces trust in the network.

► **Determination of proper network structure:** There is no specific rule for determining the structure of artificial neural networks. Appropriate network structure is achieved through experience and trial and error.

► **Difficulty of showing the problem to the network:**ANNs can work with numerical information. Problems have to be translated into numerical values before being introduced to ANN. The display mechanism to be determined here  will directly influence the **performance of the** network . This depends on the user's ability.

► **The duration of the** **network is** **unknown: The** network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.