JB INSTITUTE OF ENGINEERING & TECHNOLOGY
UGC Autonomous
Accredited by NAAC & NBA, Approved by AICTE & Permanently Affiliated to JNTUH

Department of Artificial Intelligence &Machine Learning

ARTIFICIAL INTELLIGENCE AND ITS APPLICATIONS

STUDY MATERIAL
UNIT 4

B.Tech II Year – II Sem (Section A & B)Academic Year 2022-23

PREPARED & COMPILED BY

Mrs. Beulah J Karthikeyan Assistant Professor, Department of AI&ML
JBIET
Bhaskar Nagar, Yenkapally (V), Moinabad (M), Ranga Reddy, District Hyderabad., Moinabad, Hyderabad 500075Telangana, India.

MODULE 4

Learning: What Is Learning? Rote Learning, Learning by Taking Advice, Learning in Problem Solving. Learning from Examples, Winston's Learning Program, Decision Trees.
Expert Systems: Representing and Using Domain Knowledge. Shell, Explanation, Knowledge Acquisition.

What is Learning?

Learning process is the basis of knowledge acquisition process. Knowledge acquisition is the expanding the capabilities of a system or improving its performance at some specified task. So we can say knowledge acquisition is the goal oriented creation and refinement of knowledge. The acquired knowledge may consist of various facts, rules, concepts, procedures, heuristics, formulas, relationships or any other useful information. Knowledge can be acquired from various sources like, domain of interests, text books, technical papers, databases, reports. The terms of increasing levels of abstraction, knowledge includes data, information and Meta knowledge. Meta knowledge includes the ability to evaluate the knowledge available, the additional knowledge required and the systematic implied by the present rules. Learning involves generalization from experience. Computer system is said to have learning if it is able to not only do the "repetition of same task" more effectively, but also the similar tasks of the related domain. Learning is possible due to some factors like the skill refinement and knowledge acquisition. Skill refinement refers to the situation of improving the skill by performing the same task again and again. If machines are able to improve their skills with the handling of task, they can be said having skill of learning. On the other hand, as we are able to remember the experience or gain some knowledge by handling the task, so we can improve our skill.

We would like our learning algorithms to be efficient in three respects:
(1) Computational: Number of computations during training and during recognition.
(2) Statistical: Number of examples required for good generalization, especially labeled data.
(3) Human Involvement: Specify the prior knowledge built into the model before training.

Design of learning element is dictated by the followings.

(1) What type of performance element is used?
(2) Which functional component is to be learned?
(3) How that functional component is represented?
(4) What kind of feedback is available?
(5) How can be compared between the existing feedbacks with the new data?
(6) What are the levels of comparisons? Etc.

Any system designed to create new knowledge and thereby improve its performance must include a set of data structures that represents the system's present level of expertise and a task algorithm that uses the rules to guide the system's problem solving activity.

Hence the inputs may be any types of inputs, those are executed for solution of a problem. Those inputs are processed to get the corresponding results. The learning element learns some sort of

knowledges by the knowledge acquisition techniques. The acquired knowledge may be required for a same problem in future, for which that problem can be easily solved. Every learning model must contain implicit or explicit restrictions on the class of functions that can learn. Among the set of all possible functions, we are particularly interested in a subset that contains all the tasks involved in intelligent behaviour. Examples of such tasks include visual perception, auditory perception, planning, control etc. The set does not just include specific visual perception tasks, but the set of all the tasks that an intelligent agent should be able to learn. Although we may like to think that the human brain is some what general purpose, it is extremely restricted in its ability to learn high dimensional functions.

CLASSIFICATION OF LEARNING

The process of learning may be of various types. One can develop learning taxonomies based on the type of knowledge representation used (predicate calculus, rules, frames, scripts etc), the type of knowledge learned (game playing, problem solving) or by areas of application (medical diagnosis, engineering etc). Generally learning may be of two types like single agent learning and multi-agent learning.

Single Agent Learning

Over the last four decades, machine learning's primary interest has been single agent learning. Single agent learning involves improving the performance or increasing the knowledge of a single agent. An improvement in performance or an increase in knowledge allows the agent to solve past problems with better quality or efficiency. An increase in knowledge may also allow the agent to solve new problems. An increase in performance is not necessarily due to an increase in knowledge. It may be brought about simply by rearranging the existing knowledge or utilizing it in a different manner. Single agent learning systems may be classified according to their underlying learning strategies. These strategies are classified as follows.

Rote Learning

This strategy does not require the learning system to transform or infer knowledge. It is the simplest form of learning. It requires the least amount of inference and is accomplished by simply copying the knowledge in the same form that it will be used directly into the knowledge base. It includes learning my imitation, simple memorization and learning by being performed. For example we may use this type of learning when we memorize multiplication tables. In this method we store the previous computed values, for which we do not have for recomputed them later. Also we can say rote learning is one type of existing or base learning. For example, in our childhood, we have the knowledge that "sun rises in the east". So in our later stage of learning we can easily memorize the thing. Hence in this context, a system may simply memorize previous solutions and recall them when confronted with the same problem. Generally access of stored value must be faster than it would be to recompute. Methods like hashing, indexing and sorting can be employed to enable this. One drawback of rote learning is it is not very effective in a rapidly changing environment. If the environment does change then we must detect and record exactly what has changed. Also this technique must not decrease the efficiency of the system. We must be able to decide whether it is worth storing the value in the first place.

Learning from Instruction

This strategy also known as learning by being told or learning by direct instruction. It requires the learning system to select and transform knowledge into a usable form and then integrate it into the existing knowledge of the system. It is a more complex form of learning. This learning technique requires more inference than rote learning. It includes learning from teachers and learning by using books, publications and other types of instructions.

Learning by Deduction

This process is accomplished through a sequence of deductive inference steps using known facts. From the known facts, new facts or relationships are logically derived. Using this strategy, the learning system derives new facts from existing information or knowledge by employing deductive inference. It requires more inferences than other techniques. The inference method used is a deductive type, which is a valid form of inference. For example we can say x is the cousin of y if we have the knowledge of x's and y's parents and the rules for cousin relationships. The learner draws deductive inferences from the knowledge and reformulates them in the form of useful conclusions which preserve the information content of the original data. Deductive learning includes knowledge reformulation, compilation and organizational procedures that preserve the truth of the original formulation.

Learning by Analogy

It is a process of learning a new concept or solution through the use of similar known concepts or solutions. We make frequent use of analogical learning. The first step is inductive inference, required to find a common substructure between the problem domain and one of the analogous domains stored in the learner's existing knowledge base. This form of learning requires the learning system to transform and supplement its existing knowledge from one domain or problem area into new domain. This strategy requires more inferencing by the learning system than previous strategies. Relevant knowledge must be found in the systems existing knowledge by using induction strategies. This knowledge must then be transformed to the new problem using deductive inference. Example of learning by analogy may include the driving technique of vehicles. If we know the driving procedure of a bike, then when we will drive a car then some sort of previous learning procedures we may employ. Similarly for driving a bus or truck, we may use the procedure for driving a car.

Learning from Examples

In this process of learning it includes the learning through various interactive and innovative examples. This strategy, also called concept acquisition. It requires the learning system to induce general class or concept descriptions from examples. Since the learning system does not have prior or analogous knowledge of the concept area, the amount of inferencing is greater than both learning by deduction and analogy. For solving a newly designed problem we may use its corresponding old examples.

Learning from Observations and Discovery

Using this strategy, the learning system must either induce class descriptions from observing the environment or manipulate the environment to acquire class descriptions or concepts. This is an unsupervised learning technique. It requires the greatest amount of inferencing among all of the

different forms of learning. From an existing knowledge base, some new forms of discovery of knowledge may formed. The learning discovery process is very important in the respect of constructing new knowledge base.

Learning by Induction

Inductive learning is the system that tries to induce a general rule based on observed instances. In other words, the system tries to infer an association between specific inputs and outputs. In general, the input of the program is a set of training instance where the output is a method of classifying subsequent instance. For example, the input of the program may be color of types of fruits where the output may be the types of fruits those are useful for protein. Induction method involves the learning by examples, experimentation, observation and discovery. The search spaces encountered in learning tend to be extremely large, even by the standards of search based problem solving. This complexity of problems is cleared by choosing a problem among the different generalizations supported by any given training data. Inductive bias refers to any method a learning program uses to constrain the space of possible generalizations. A no. of inductive learning algorithms have been developed like Probably Approximately Correct (PAC), Version spaces etc. Probably Approximately Correct learning was proposed concerning the situation that cannot be deductive. Approximately correct is recognized whenever the program can get most of the problems right. In order to increase performance of the program, learning algorithms should restrict the size of hypothesis space. On the other hand, the goal of version space is to produce a description that uses only positive examples. The program in practice, produce a description of all acceptable concepts. In detail, we may conclude that there are two sets of concepts that are produced during learning. Firstly, the most specific concept describes what the target set should be. Secondly, the least specific concept describes what should not be in the target group. Generally inductive learning is frequently used by humans. This form of learning is more powerful than the others. We use this learning when we formulate a general concept after seeing a number of instances. For example, we can say the taste of sugar is sweet if we have the knowledge about sweetness.

Learning from Advices

In this process we can learn through taking advice from others. The idea of advice taking learning was proposed in early 1958 by McCarthy. In our daily life, this learning process is quite common. Right from our parents, relatives to our teachers, when we start our educational life, we take various advices from others. All most all the initial things and all type of knowledges we acquire through the advices of others. We know the computer programs are written by programmers. When a programmer writes a computer program he or she gives many instructions to computer to follow, the same way a teacher gives his/her advice to his students. The computer follows the instructions given by the programmer. Hence, a kind of learning takes place when computer runs a particular program by taking advice from the creator of the program.

Learning by Clustering

This process is similar to the inductive learning. Clustering is a process of grouping or classifying objects on the basis of a close association or shared characteristics. The clustering process is essentially required in a learning process in which similarity patterns are found among a group of objects. The program must discover for itself the natural classes that exist for the objects, in addition to a method for classifying instances. AUTOCLASS (Cheeseman et al., 1988) is one program that

accepts a number of training cases and hypothesizes a set of classes. For any given case, the program provides a set of probabilities that predict into which classes the case is likely to be fall.

Winston's Learning Program

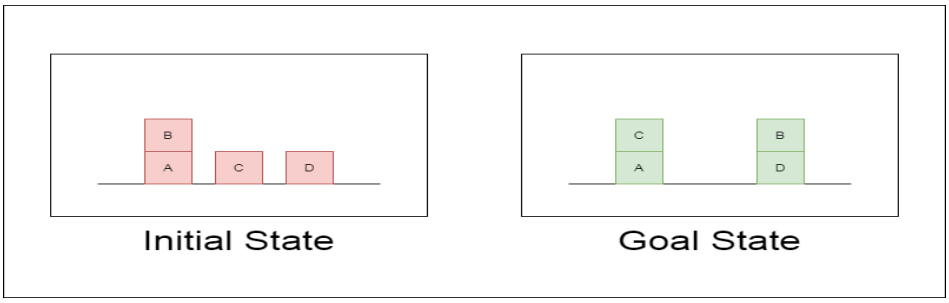
Winston's doctoral thesis at MIT entitled "Learning Structural Descriptions from Examples" (1970) was a major step towards a clarification of how concepts involving complex structural relationships might be learned.

His program is presented with line drawings of scenes containing children's toy blocks, such as bricks, cubes, pyramids, and wedges. The program forms descriptive networks for these scenes, which shows the properties and relationships of the objects appearing in them. Using these structural descriptions, the program can learn structural concepts such as "pedestal", "arch" or "arcade" on the basis of examples and counterexamples of the concepts.

While the content of the scenes is severly restricted, the methods employed in the program seem quite general, or at least readily generalisible to more realistic concept-formation tasks (consider "chair", "table", "house", etc.) Certainly this type of structural learning is a long way off from learning by parameter adjustment, which has been the dominant paradigm in pattern recognition research (Uhr & Vossler's program and some others excepted).

Winston's program uses Guzman's algorithm to determine the bodies in a scene; it then determines which edges belong to which object and fills in partially occluded edges. Then it infers the types of objects (brick, wedge, etc.) from the shapes and adjacency relationships of the viable faces. The sizes and orientation are then readily available.

Goal Stack Planning for Blocks World Problem



What is Blocks World Problem?

This is how the problem goes — There is a table on which some blocks are placed. Some blocks may or may not be stacked on other blocks. We have a robot arm to pick up or put down the blocks. The robot arm can move only one block at a time, and no other block should be stacked on top of the block which is to be moved by the robot arm.

Our aim is to change the configuration of the blocks from the Initial State to the Goal State, both of which have been specified in the diagram above.

What is Goal Stack Planning?

Goal Stack Planning is one of the earliest methods in artificial intelligence in which we work **backwards from the goal state to the initial state.**

We start at the goal state and we try fulfilling the preconditions required to achieve the initial state. These preconditions in turn have their own set of preconditions, which are required to be satisfied first. We keep solving these "goals" and "sub-goals" until we finally arrive at the Initial State. **We make use of a stack to hold these goals that need to be fulfilled as well the actions that we need to perform for the same**.

Apart from the "Initial State" and the "Goal State", we maintain a **"World State"** configuration as well. Goal Stack uses this world state to work its way from Goal State to Initial State. World State on the other hand starts off as the Initial State and ends up being transformed into the Goal state.

At the end of this algorithm we are left with an empty stack and a set of actions which helps us navigate from the Initial State to the World State.

Representing the configurations as a list of "predicates"

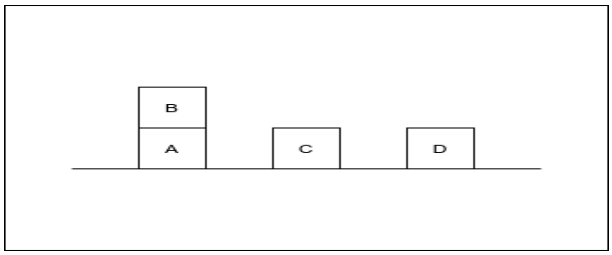
Predicates can be thought of as a statement which helps us convey the information about a configuration in Blocks World.

Given below are the list of predicates as well as their intended meaning

1. ON(A,B) : Block A is on B
2. ONTABLE(A) : A is on table
3. CLEAR(A) : Nothing is on top of A
4. HOLDING(A) : Arm is holding A.
5. ARMEMPTY : Arm is holding nothing

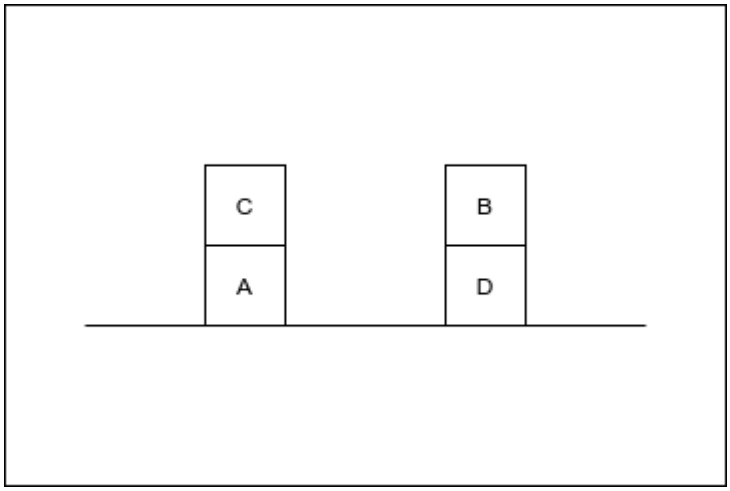
Using these predicates, we represent the Initial State and the Goal State in our example like this:

Initial State — ON(B,A) ∧ ONTABLE(A) ∧ ONTABLE(C) ∧ ONTABLE(D) ∧ CLEAR(B) ∧ CLEAR(C) ∧ CLEAR(D) ∧ ARMEMPTY



Initial State

Goal State — ON(C,A) ∧ ON(B,D) ∧ ONTABLE(A) ∧ ONTABLE(D) ∧ CLEAR(B) ∧ CLEAR(C) ∧ ARMEMPTY



Goal State

Thus a configuration can be thought of as a list of predicates describing the current scenario.

"Operations" performed by the robot arm

The Robot Arm can perform 4 operations:

1. STACK(X,Y) : Stacking Block X on Block Y
2. UNSTACK(X,Y) : Picking up Block X which is on top of Block Y
3. PICKUP(X) : Picking up Block X which is on top of the table
4. PUTDOWN(X) : Put Block X on the table
5. All the four operations have certain preconditions which need to be satisfied to perform the same. These preconditions are represented in the form of predicates.

The effect of these operations is represented using two lists **ADD** and **DELETE**. DELETE List contains the predicates which will cease to be true once the operation is performed. ADD List on the other hand contains the predicates which will become true once the operation is performed.

The Precondition, Add and Delete List for each operation is rather intuitive and have been listed below.

OPERATORS	PRECONDITION	DELETE	ADD
STACK(X,Y)	CLEAR(Y)∧ HOLDING(X)	CLEAR(Y) HOLDING(X)	ARMEMPTY ON(X,Y)
UNSTACK(X,Y)	ARMEMPTY∧ ON(X,Y)∧ CLEAR(X)	ARMEMPTY∧ ON(X,Y)	HOLDING(X) ∧CLEAR(Y)
PICKUP(X)	CLEAR(X)∧ ONTABLE(X)∧ ARMEMPTY	ONTABLE(X)∧ ARMEMPTY	HOLDING(X)
PUTDOWN(X)	HOLDING(X)	HOLDING(X)	ONTABLE(X)∧ ARMEMPTY

Operations performed by the Robot Arm

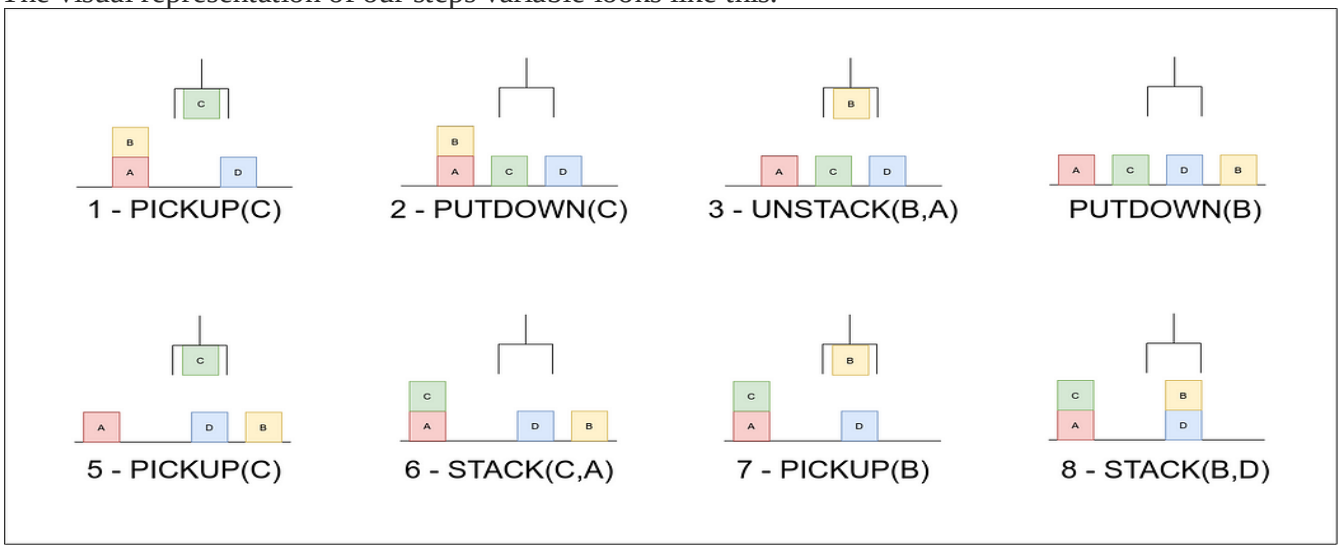
For example, to perform the **STACK(X,Y)** operation i.e. to Stack Block X on top of Block Y, No other block should be on top of Y **(CLEAR(Y))** and the Robot Arm should be holding the Block X (**HOLDING(X)**).

Once the operation is performed, these predicates will cease to be true, thus they are included in **DELETE List** as well. (Note : It is not necessary for the Precondition and DELETE List to be the exact same).

On the other hand, once the operation is performed, The robot arm will be free (**ARMEMPTY**) and the block X will be on top of Y (**ON(X,Y)**).The other 3 Operators follow similar logic, and this part is the cornerstone of Goal Stack Planning.

In this example, steps = [PICKUP(C), PUTDOWN(C), UNSTACK(B,A), PUTDOWN(B), PICKUP(C), STACK(C,A), PICKUP(B), STACK(B,D)]

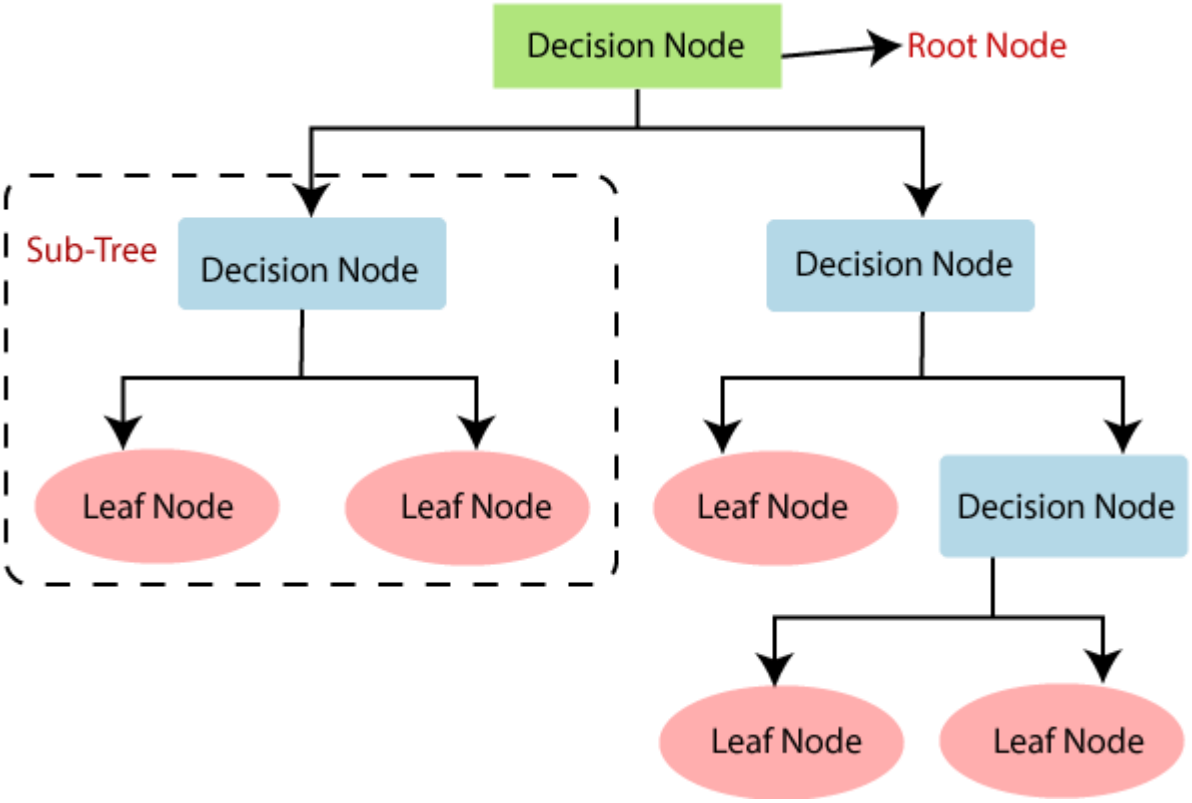
The visual representation of our steps variable looks like this.



Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

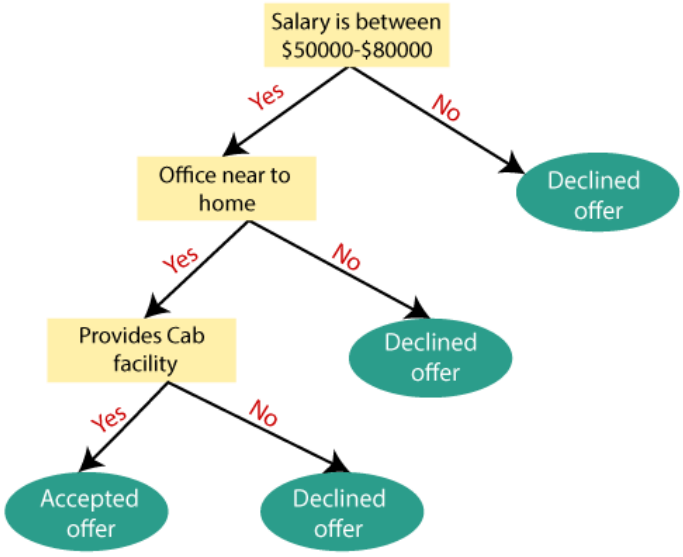
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

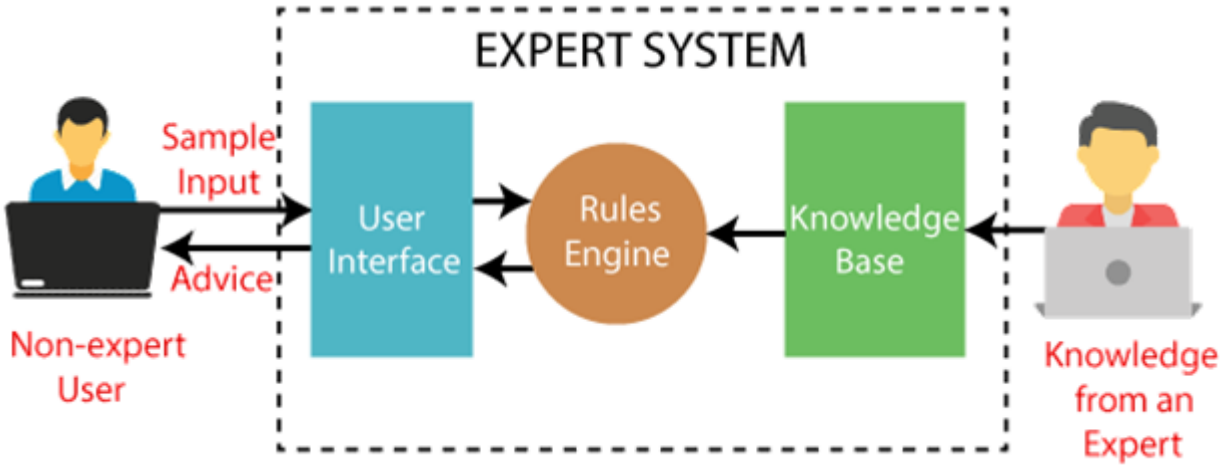
What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for compsex problems using **both facts and heuristics like a human expert**. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as **medicine, science,** etc.

The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

Below is the block diagram that represents the working of an expert system:



Note: It is important to remember that an expert system is not used to replace the human experts; instead, it is used to assist the human in making a complex decision. These systems do not have human capabilities of thinking and work on the basis of the knowledge base of the particular domain.

Below are some popular examples of the Expert System:

- **DENDRAL:** It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.
- **MYCIN:** It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.
- **PXDES:** It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.
- **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer at early stages.

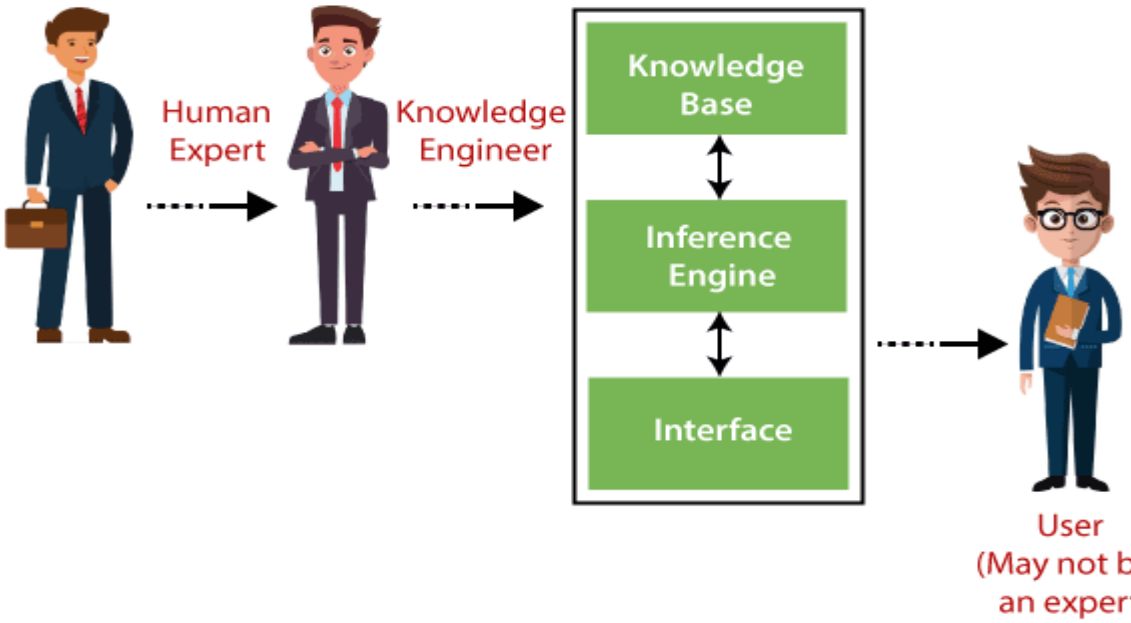
Characteristics of Expert System

- **High Performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.
- **Understandable:** It responds in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.
- **Reliable:** It is much reliable for generating an efficient and accurate output.
- **Highly responsive:** ES provides the result for any complex query within a very short period of time.

Components of Expert System

An expert system mainly consists of three components:

- **User Interface**
- **Inference Engine**
- **Knowledge Base**



1. User Interface

With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, **it is an interface that helps a non-expert user to communicate with the expert system to find a solution**.

2. Inference Engine(Rules of Engine)

- The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.
- With the help of an inference engine, the system extracts the knowledge from the knowledge base.
- There are two types of inference engine:
- **Deterministic Inference engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on **facts** and **rules**.
- **Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

- **Forward Chaining:** It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.
- **Backward Chaining:** It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

3. Knowledge Base

- The knowledgebase is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.
- It is similar to a database that contains information and rules of a particular domain or subject.
- One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Components of Knowledge Base

- **Factual Knowledge:** The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.
- **Heuristic Knowledge:** This knowledge is based on practice, the ability to guess, evaluation, and experiences.

Knowledge Representation: It is used to formalize the knowledge stored in the knowledge base using the If-else rules.

Knowledge Acquisitions: It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

Development of Expert System

Here, we will explain the working of an expert system by taking an example of MYCIN ES. Below are some steps to build an MYCIN:

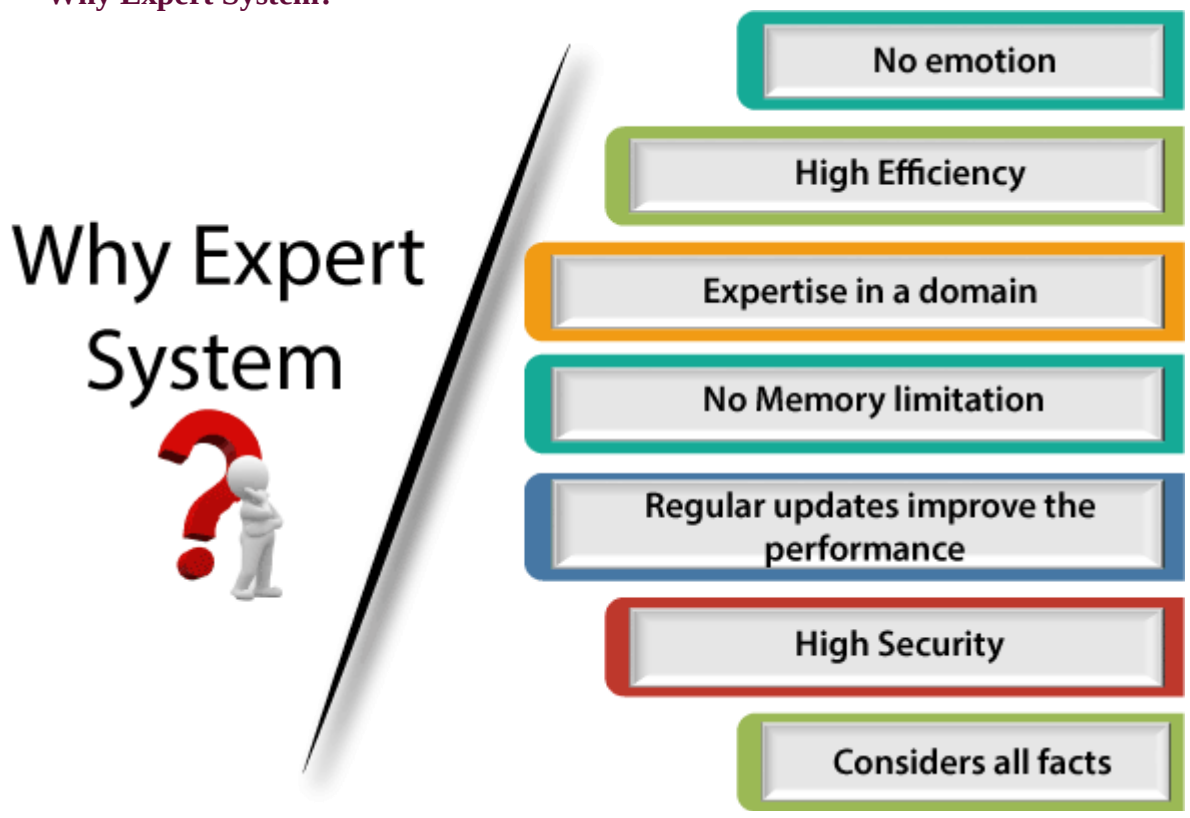
- Firstly, ES should be fed with expert knowledge. In the case of MYCIN, human experts specialized in the medical field of bacterial infection, provide information about the causes, symptoms, and other knowledge in that domain.
- The KB of the MYCIN is updated successfully. In order to test it, the doctor provides a new problem to it. The problem is to identify the presence of the bacteria by inputting the details of a patient, including the symptoms, current condition, and medical history.
- The ES will need a questionnaire to be filled by the patient to know the general information about the patient, such as gender, age, etc.
- Now the system has collected all the information, so it will find the solution for the problem by applying if-then rules using the inference engine and using the facts stored within the KB.
- In the end, it will provide a response to the patient by using the user interface.

Participants in the development of Expert System

There are three primary participants in the building of Expert System:

1. **Expert:** The success of an ES much depends on the knowledge provided by human experts. These experts are those persons who are specialized in that specific domain.
2. **Knowledge Engineer:** Knowledge engineer is the person who gathers the knowledge from the domain experts and then codifies that knowledge to the system according to the formalism.
3. **End-User:** This is a particular person or a group of people who may not be experts, and working on the expert system needs the solution or advice for his queries, which are complex.

Why Expert System?



Before using any technology, we must have an idea about why to use that technology and hence the same for the ES. Although we have human experts in every field, then what is the need to develop a computer-based system. So below are the points that are describing the need of the ES:

1. **No memory Limitations:** It can store as much data as required and can memorize it at the time of its application. But for human experts, there are some limitations to memorize all things at every time.
2. **High Efficiency:** If the knowledge base is updated with the correct knowledge, then it provides a highly efficient output, which may not be possible for a human.
3. **Expertise in a domain:** There are lots of human experts in each domain, and they all have different skills, different experiences, and different skills, so it is not easy to get a final output for the query. But if we put the knowledge gained from human experts into the expert system, then it provides an efficient output by mixing all the facts and knowledge
4. **Not affected by emotions:** These systems are not affected by human emotions such as fatigue, anger, depression, anxiety, etc.. Hence the performance remains constant.
5. **High security:** These systems provide high security to resolve any query.
6. **Considers all the facts:** To respond to any query, it checks and considers all the available facts and provides the result accordingly. But it is possible that a human expert may not consider some facts due to any reason.
7. **Regular updates improve the performance:** If there is an issue in the result provided by the expert systems, we can improve the performance of the system by updating the knowledge base.

Capabilities of the Expert System

Below are some capabilities of an Expert System:

- **Advising:** It is capable of advising the human being for the query of any domain from the particular ES.
- **Provide decision-making capabilities:** It provides the capability of decision making in any domain, such as for making any financial decision, decisions in medical science, etc.
- **Demonstrate a device:** It is capable of demonstrating any new products such as its features, specifications, how to use that product, etc.
- **Problem-solving:** It has problem-solving capabilities.
- **Explaining a problem:** It is also capable of providing a detailed description of an input problem.
- **Interpreting the input:** It is capable of interpreting the input given by the user.
- **Predicting results:** It can be used for the prediction of a result.
- **Diagnosis:** An ES designed for the medical field is capable of diagnosing a disease without using multiple components as it already contains various inbuilt medical tools.

Advantages of Expert System

- These systems are highly reproducible.
- They can be used for risky places where the human presence is not safe.
- Error possibilities are less if the KB contains correct knowledge.
- The performance of these systems remains steady as it is not affected by emotions, tension, or fatigue.
- They provide a very high speed to respond to a particular query.

Limitations of Expert System

- The response of the expert system may get wrong if the knowledge base contains the wrong information.
- Like a human being, it cannot produce a creative output for different scenarios.

- Its maintenance and development costs are very high.
- Knowledge acquisition for designing is much difficult.
- For each domain, we require a specific ES, which is one of the big limitations.
- It cannot learn from itself and hence requires manual updates.

Applications of Expert System

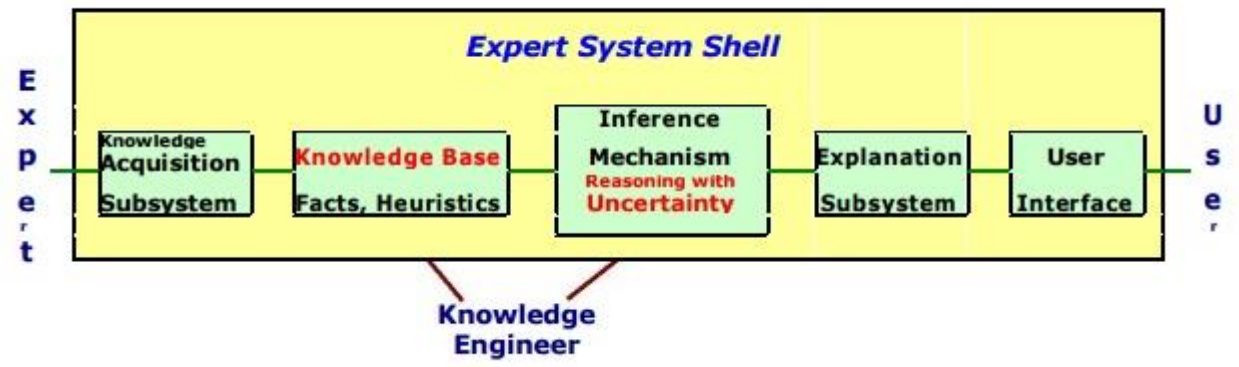
- **In designing and manufacturing domain**
  It can be broadly used for designing and manufacturing physical devices such as camera lenses and automobiles.
- **In the knowledge domain**
  These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax advisor.
- **In the finance domain**
  In the finance industries, it is used to detect any type of possible fraud, suspicious activity, and advise bankers that if they should provide loans for business or not.
- **In the diagnosis and troubleshooting of devices**
  In medical diagnosis, the ES system is used, and it was the first area where these systems were used.
- **Planning and Scheduling**
  The expert systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.

Expert System Shells

- An Expert system shell is a software development environment. It contains the basic components of expert systems. A shell is associated with a prescribed method for building applications by configuring and instantiating these components.

Shell components and description

- The generic components of a shell : the knowledge acquisition, the knowledge Base, the reasoning, the explanation and the user interface are shown below. The knowledge base and reasoning engine are the core components.



All these components are described in the next slide.

- **■ Knowledge Base**
- A store of factual and heuristic knowledge. Expert system tool provides one or more knowledge representation schemes for expressing knowledge about the application domain. Some tools use

both Frames (objects) and IF-THEN rules. In PROLOG the knowledge is represented as logical statements.

■ **Reasoning Engine**

- Inference mechanisms for manipulating the symbolic information and knowledge in the knowledge base form a line of reasoning in solving a problem. The inference mechanism can range from simple modus ponens backward chaining of IF-THEN rules to Case-Based reasoning.

Knowledge Acquisition subsystem

- A subsystem to help experts in build knowledge bases. However, collecting knowledge, needed to solve problems and build the knowledge base, is the biggest bottleneck in building expert systems.

Explanation subsystem

- A subsystem that explains the system's actions. The explanation can range from how the final or intermediate solutions were arrived at justifying the need for additional data.

- **User Interface**

- A means of communication with the user. The user interface is generally not a part of the expert system technology. It was not given much attention in the past. However, the user interface can make a critical difference in the perceived utility of an Expert system.

Explanation

Most expert systems have explanation facilities that allow the user to ask questions - why and how it reached some conclusion.

The questions are answered by referring to the system goals, the rules being used, and existing problem solving. The rules typically reflect empirical, or "compiled" knowledge. They are codes of an expert's rules of thumb, not the expert's deeper understanding.

Example :

Dialog with an expert system designed to give advice on car problems.

System Does the car start ?

User No.

System Does the engine turn over ?

User Yes.

System Do you smell gas ?

User Yes

System Recommendation - Wait 5 minutes and try again.

User Why ?

System I used the rule:

If not start, and engine_turn_over, and smell_gas

Then recommendation is 'Wait 5 minutes and try again'

User

Note : The rule gives the correct advice for a flooded car, and knows the questions to be ask to determine if the car is flooded, but it does not contain the knowledge of what a flooded car is and why waiting will help.

Types of Explanation

There are four types of explanations commonly used in expert systems.

Rule trace reports on the progress of a consultation;

‡ Explanation of how the system reached to the given conclusion;

KNOWLEDGE ACQUISITION

Knowledge acquisition is the gathering or collecting knowledge from various sources. It is the process of adding new knowledge to a knowledge base and refining or improving knowledge that was previously acquired. Acquisition is the process of expanding the capabilities of a system or improving its performance at some specified task. So it is the goal oriented creation and refinement of knowledge. Acquired knowledge may consist of facts, rules, concepts, procedures, heuristics, formulas, relationships, statistics or any other useful information. Source of these knowledges may be experts in the domain of interest, text books, technical papers, database reports, journals and the environments. The knowledge acquisition is a continuous process and is spread over entire lifetime. Example of knowledge acquisition is machine learning. It may be process of autonomous knowledge creation or refinements through the use of computer programs. The newly acquired knowledge should be integrated with existing knowledge in some meaningful way. The knowledge should be accurate, non-redundant, consistent and fairly complete. Knowledge acquisition supports the activities like entering the knowledge and maintaining knowledge base. The knowledge acquisition process also sets dynamic data structures for existing knowledge to refine the knowledge.

The role of knowledge engineer is also very important with respect to develop the refinements of knowledge. Knowledge engineers may be the professionals who elicit knowledge from experts. They integrate knowledge from various sources like creates and edits code, operates the various interactive tools, build the knowledge base etc.

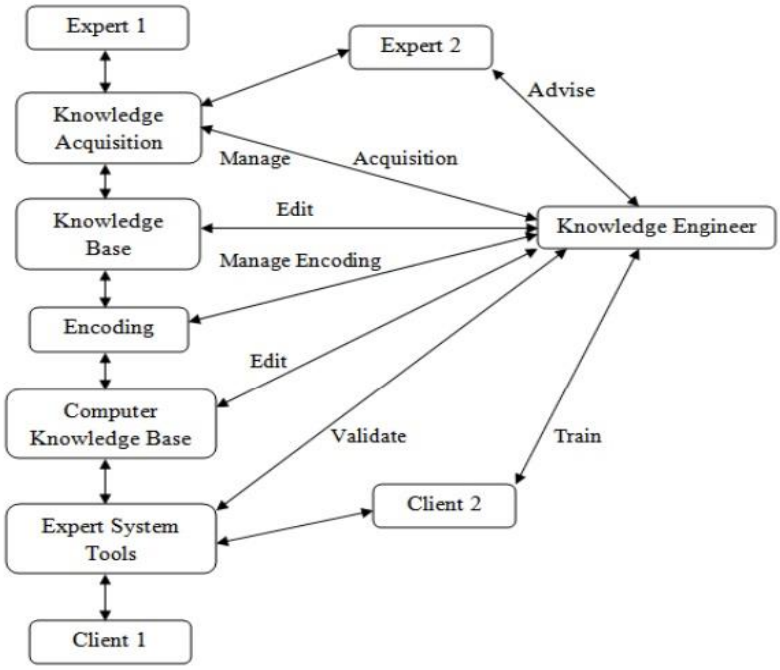


Figure : Knowledge Engineer's Roles in Interactive Knowledge Acquisition

Knowledge Acquisition Techniques

Many techniques have been developed to deduce knowledge from an expert. They are termed as knowledge acquisition techniques. They are:

a) Diagram Based Techniques

b) Matrix Based Techniques

c) Hierarchy-Generation Techniques

d) Protocol Analysis Techniques

e) Protocol Generation Techniques

f) Sorting Techniques

In diagram based techniques the generation and use of concept maps, event diagrams and process maps. This technique captures the features like "why, whe n, who, how and where". The matrix based techniques involve the construction of grids indicating such things as problems encountered against possible solutions. Hierarchical techniques are used to build hierarchical structures like trees. Protocol analysis technique is used to identify the type of knowledge like goals, decisions, relationships etc. The protocol generation techniques include various types of interviews like structured, semi-structured and unstructured.

The most common knowledge acquisition technique is face-to-face interview. Interview is a very important technique which must be planned carefully. The results of an interview must be verified and validated. Some common variations of an unstructured interview are talk through, teach through and read through. The knowledge engineer slowly learns about the problem. Then can build a representation of the knowledge. In unstructured interviews, seldom provides complete or well-organized descriptions of cognitive processes because the domains are generally complex. The experts usually find it very difficult to express some more important knowledge. Data acquired are often unrelated, exists at varying levels of complexity, and are difficult for the knowledge engineer to review, interpret and integrate. But on the other hand structured interviews are systematic goal oriented process. It forces an organized communication between the knowledge engineer and the expert. In structured interview, inter personal communication and analytical skills are important.