7 CHAPTER LEARNING

7.0 INTRODUCTION

Learning is done by viewing, listening, interactions, studying and by experience. Learning provides us the power to reason, ability to handle new situations and enables us to act in an intelligent way. Human beings are intelligent as they possess knowledge of world. Similarly, making a machine intelligent means it should have the power of learning. So, **Machine learning** is a challenge now. An **intelligent machine** should be able to learn new things and to adapt to new situations rather than simply doing steps as they are made to do. **Please understand that a machine cannot be called intelligent, if it does not have power of learning.** In literature, many definitions of learning have been given. Let us give some here too—

1. **Herbert Simon (1983)**

"Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks more efficiently and more effectively the next time."

2. **Marvin Minsky (1986)**

"Learning is making useful changes in the working of our mind."

3. **Ryszard Michalski (1986)**

"Learning is constructing or modifying representations of what is being experienced."

4. **Mitchell (1997)**

"A computer program is said to learn from experience (E) w.r.t. some class of tasks (T) and performance measure (P), if its performance at tasks in T, as measured by P, improves with experience E".

The learning problem is divided into three types—

(*a*) Unsupervised learning

It finds a model that matches a stream of input experiences and is able to predict what new experience to expect.

(*b*) Supervised learning

It **classifies** (be able to determine) what category something belongs to after seeing a number of examples of things from each category or regression (given a set of numerical input/output examples, discover a continuous function that would generate the outputs from the inputs).

(*c*) Reinforcement learning

The agent is rewarded for good responses and punished for bad ones. They can be analyzed in terms of decision theory, using concepts like utility. The mathematical analysis of machine

learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory.

7.1 CONCEPT OF LEARNING

The major paradigms of machine learning are as follows—

(a) **Rote learning** by memorization (saving knowledge so that it can be used again).

(b) **Induction** involves the process of learning by example where a system tries to induce a general rule from a set of observed instances.

(c) **Analogy.** It can recognize similarities in information already stored. It can determine correspondence between two different representations.

(d) **Genetic Algorithms (GA)** are a part of evolutionary computing, a way of solving problems by mimicking processes. Nature uses selection, crosses over, mutation and accepting to evolve a solution to a problem.

(e) **Reinforcement.** It assigns rewards, positive or negative, act end of a sequence of steps, it learns which actions are good or bad.

Let us now discuss the general block diagram model of learning.

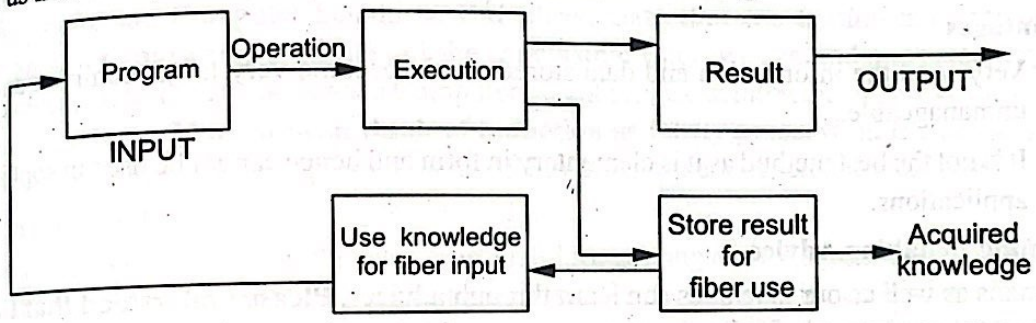


Fig. 7.1 General block diagram of a learning system.

As shown in Fig. 7.1, the input program is any general program executed for solution of a problem. The results are reported as output. These results are also stored back to acquire knowledge for future use. This simply means that, if a similar problem is required to be solved next time, the results can automatically be taken from previously acquired knowledge. **Please note that learning involves generalization from experience.** A system is said to be learning if it not only does the repetition of same task more effectively but also the similar tasks of the related domain. It covers a wide range of phenomenon:

(a) Knowledge acquisition.

(b) Skill refinement.

Knowledge acquisition refers to a situation in which computer executes one program and remembers the process for future use.

Skill refinement means the ability because of which, by doing some task repeatedly people tend to solve the task in lesser time. By practice skill tends to get refined.

Same capabilities are expected out of machines also.

7.2 FIVE TYPES OF LEARNING

There are five main methods of learning in context of machines. They are as follows—

(a) Rote learning

(b) Learning by taking advice

(*c*) Learning by induction

(*d*) Learning by deduction (or Relevance-based Learning).

(*e*) Learning by analogy

I. Rote Learning/Learning through memorization

Rote learning is based on the principle of **caching**. Caching means to store data of a particular task.

For example, if say, a chess is played then the winning sequence of moves will simply be remembered by the players involved. Next time, instead of arbitrarily selecting from all possible moves, they will select a move from among the winning moves. **Please note that by repeatedly remembering the winning situation, the choice of move becomes improved.** This is what humans can do and hence we expect this much from our machines also. If our machine uses this technique of data caching then a significant amount of time can be saved as our intermediate results are already stored and are being only reused here.

Advantages

1. It enables machine to store organized information.
2. It is faster to reuse information stored than to collect it.
3. We get generalized information stored in the database using Rote learning.

Disadvantages

1. Very soon the information and data stored would become very large, complex and hence unmanageable.
2. It is not the best method as it is elementary in form and hence cannot be used in sophisticated applications.

II. Learning by taking Advice

Both humans as well as our machines can learn through advices. **Please understand that this kind of learning takes place when a computer runs a particular program by taking advice from its programmer (or creator of that program).**

Here the knowledge provided by the teacher is at a level more general than that usable by the performance element. The system has to particularize this general advice to specific situations.

For example: The advice often given to novices trying to learn card games.

Also note that much of our education involves this type of learning. The main problem to be tackled by the learning element here is transforming the general purposes knowledge into specific rules and heuristics which can be used by the performance element. This process is called as **operationalisation**. For a realistic model we have to assume that the advice is given in natural language which further complicates the problem. This is a difficult learning problem and there are a few interesting models proposed.

III. Learning by Induction

Induction involves drawing general conclusions from specific observations. Learning from examples and **learning by discovery** are instances of inductive learning. Most of the knowledge that the human race has acquired about its environment is through the process of induction.

For example: Observing the specific facts that the sun has risen in the east on day 1, day 2 etc., we infer a general statement, namely, sun always rises in the east.

IV. Learning by Deduction/Relevance-based Learning

Another method of inferring some conclusions from a given set of statements is deduction. Here, we start with a set of so called **axioms** and using rules of logic, derive conclusion. Provided that the logic we use is consistent, all the derived statements are true if the original axioms are true. **Please note**

were that **the deduction is a truth preserving process whereas induction is not.** This is because from a given set of observations there are infinitely many inductive conclusions possible. Hence, while learning from examples, we have to pay special attention to the problem of how far should the observations be generalized. Presence of counter examples helps us from **overgeneralization.** But even then there may be many possible generalizations and hence the learning system should be provided with enough background knowledge for it to prefer **good** generalizations.

For example: Starting with the statements—

S1: All people attending workshops on expert systems are computer scientists.

S2: Mr. X attended a workshop on expert systems.

We can derive conclusion—

Mr. X is a Computer Scientist.

If the first two statements are true then so is the derived conclusion. **This is because in deduction we go from the general to the specific.** On the other hand, in induction we go from the specific **to general** and hence it is not possible to guarantee, in general, the correctness of the conclusions.

For example, observing that Mr. X, Mr. Y and so on, all of whom are Computer Scientists, attend workshops on expert systems, we can inductively conclude that all people attending workshops on expert systems are Computer Scientists. **But please note that the truthness of the starting premises does not guarantee truthness of the conclusion.** But if we can find one person attending workshop on expert systems who is not a Computer Scientist, we can immediately prove the inductive conclusion to be false. **Thus, we can think of induction as falsity preserving.** Incidentally, this is the reason why counter examples play a crucial role in development of scientific theories.

V. Learning by Analogy

Here, the knowledge provided by the teacher is useful for solving an analogous though not identical problem. Hence, the learning element needs to make some inferences to decide how this knowledge needs to be transformed so that it is useful for the problem at hand. There is not much work done in this area.

7.3 OTHER TYPES OF LEARNING

There are various other types of learning as follows—

(*a*) Concept Learning (CL).

(*b*) Explanation Based Learning (EBL).

(*c*) Symbol Based Learning (SBL).

(*d*) Relevance Based Learning (RBL)/Deductive Learning.

Let us discuss them now.

I. Concept Learning

The problem of learning a single concept from examples and counter examples is called as **concept learning.**

Associated with each concept learning problem is a domain of objects which is characterized by a set of attributes. Each object in the domain is described by a tuple of values – one for each attribute. A subset of the domain is the concept. The learning system is provided (by the teacher) with some objects from the domain and is also told whether each object belongs to the concept or not. All objects belonging to the concept are called **positive examples** and all other objects from the domain are called **negative examples**. The system is expected to learn a representation of the concept which can be used by it for deciding whether a given object belongs to the concept.

Concept learning is the simplest case of learning from examples. To understand its relev let us consider an example. Suppose we need an expert system for diagnosing some diseases. T data needed for diagnosis will be some symptoms of the patient (e.g., temperature, whether he headache etc.) and results of clinical tests (e.g., presence of pus cells in urine etc.). Suppose system should be rule-based and for simplicity we assume that each rule has a condition part is a Boolean function of the data and an action part that gives the diagnosis. We want the syste acquire the rules by interacting with an expert. The expert will show the system various cases and diagnosis. If we restrict our attention to learning rules for diagnosing a single disease, what we h is a concept learning problem. Here the attributes are the symptoms and clinical tests. Positive negative examples are cases provided by the expert. **The objective is to learn a decision proced for diagnosing the diseases. This type of learning is useful even if the initial knowledge b is hand coded. The system can periodically interact with expert to check correctness of diagnosis.** In areas where the system is making mistakes, the expert can supply the system some typical cases and the system will automatically update its knowledge base. Since the exper generally not a computer scientist, it is easier for him to discuss typical cases than to examine modify the knowledge base of the system. **It may be noted that for the system to learn it m possess some minimal knowledge regarding the various attributes etc.**

The first issue to be addressed while developing algorithms for concept learning is the choice representation for the concept. The representation should be usable by the decision procedure an should be suitable for the learning algorithm. We will be considering two representations – **decisi trees and logic expressions.**

In a **decision tree** all nodes except the leaf nodes are labeled by a question (involving, genera the value of one attribute). All the edges from an node are labeled by possible answers for question. The leaf nodes are labeled by decisions or the class labels. Suppose we have a conc represented as a decision tree and we have to use this for deciding whether a given object belo to the concept. We start at the root node. At each node we answer the question at that node for current object and thus take the appropriate edge from the node. Finally when we come to a l node its label will be the decision.

A **logic expression** is directly usable as the condition in an if-then rule for making decisi By using simple structures for the logic expression it is possible to learn the concept as a lo expression. Consider a concept learning problem with N attributes Y_i, taking values in s v_i, $1 \le i \le N$. We distinguish between two logic expressions for representing a concept.

Defintion 2.1 A logic expression of the form

$$[Y_1 \in v_1] /\backslash \ldots /\backslash [Y_N \in v_N],\ v_i \subseteq v_i,\ v,\ i,$$

is said to be in Conjunctive Normal Form (CNF).

Here each expression of the form $[Y_i \in v_i]$ is called a **selector.** It is true if the value of attribute is in the corresponding set. Since this can be viewed as a disjunction of equality predica each of which tests if Y_i is equal to some element of v_i, a selector is said to have internal disjunctio

Defintion 2.1 A logic expression of the form

$$C_1 \backslash/ C_2 \backslash/ \ldots \backslash/ Cm,\ m \ge 1,$$

where each C_i is a CNF expression, is said to be in Disjunctive Normal Form (DNF).

In the next section we will consider algorithms for learning concepts as CNF or DNF expressio and also as decision trees.

The representation chosen for the concept determines the set of all possible concepts, the called concept space. For example, for CNF concepts the concept space will be the set of all possi tuples of the form $(A_1, \ldots, A_N)$, where A_i V_i. The set of all possible examples will be called t

instance space. The input to the learning system will be points in the instance space using which the system searches over the concept space to find best concept. If we use logic expressions for concepts then all examples (which are N-tuples of values for all attributes) are CNF concepts. Hence all examples will be points in the concept space. This is referred to as single representation trick which is useful because the examples can serve as starting points for the search over the concept space. To define the goal of this search process we need to decide what is the 'best' concept. A concept is said to be consistent if it explains all the examples. That is, if it satisfies all positive examples and none of the negative examples. But consistency by itself is not sufficient; the disjunction of all +ve examples is a consistent concept though it has no generalisation capability. If our concept space contains only CNF concepts, then we cannot any way converge to disjunction of all examples. Then the way we define the notation of generalization determines the type of concepts that the learning system converges to. **In general if we use DNF concepts then we need to employ some heuristic evaluation function to make the system prefer 'good' generalizations.**

Another feature that affects the search process is whether we are required to process one example at a time or we are allowed to store all the examples and process them together. In the former case we say that the learning is incremental. While incremental learning is very much desired, often it is difficult to learn incrementally, especially while learning DNF concepts or decision trees.

Before we complete this section it may be remarked that the concept learning problem we have discussed here is very similar to the problem of learning discriminant functions in pattern recognition. The main difference is that here the attributes may not be numerical values and there may not be any algebraic structure on the sets where the attributes take values. Hence the traditional pattern recognition algorithms that assume real valued features, are not very useful for concept learning. Also since the subjective is knowledge acquisition for expert systems, we want the learnt concepts to be represented in a form easily comprehensible to humans. That is why we use logic expressions and decision trees rather than discriminant functions to represent the learnt knowledge.

II. Explanation Based Learning (EBL)

Learning is of 2 types—

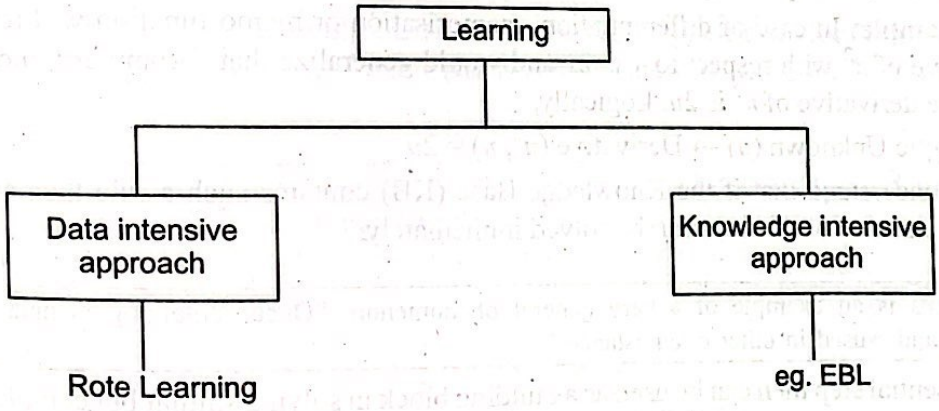


Fig. 7.2 EBL

In **data intensive** approach, the main focus is on given **data**. Large number of positive and negative examples are given to the machine and based on the outcomes of these examples, the machine learns. **For example:** Rote learning.

In **knowledge intensive** approach, the main focus is on **knowledge.** It is a kind of **analogical learning**. It is also called as **situation based learning.**

For example: EBL uses this approach.

In games like chess, medical diagnosis etc., knowledge plays a vital role. In these systems, a critical situation is handled and the machine is given the example to handle this situation This is

EBL. So, here one example in form of a knowledge solution or past experience is sufficient f learning a new solution. **In general, in EBL, the emphasis is given to learn more by taking t critical examples rather than by having more examples.** In EBL, one positive training example i given to machine in order to perform the learning. **Please note that in EBL, the learned knowledg is valid knowledge.** It is obtained by performing a **deductive reasoning process** on a set of facts. I EBL, while problem solving, its salient features are abstracted. These features are as follows—

(*a*) Formal statement about goal.

(*b*) Minimum one training example.

(*c*) Domain theory that relates to the concept and training example.

(*d*) Criteria for applying an operator.

This information is stored in the KB. When a new problem is to be solved using EBL, a matchin is performed between the new problem and existing problem details. If partial matching occurs, th solution is taken from the KB and modified according to new problem. Thus, we can conclude th EBL program attempts to learn from a single example by explaining why it is an example of targ concept. When program formalizes the explanation, it generalizes this explanation and stores i further use. The system learns this way and its performance improves.

EBL is a form of deductive learning where the learner develops a logical explanation of ho the positive training example defines a concept. In EBL, first of all, the concept is formulated usin training example and domain theory and then it is generalized by regression formulae.

EBL

An EBL system attempts to learn from a single example x by explaining why x is an exampl of the target concept (predicate) instead of learning through a large number of examples. Th explanation is then generalized and the system's performance is improved through the availability background knowledge. The process is known as **memorisation** (or memo function). **It is define as the phenomenon to accumulate a database of input/output pairs in which when a function called, it first checks the database to see whether it can avoid solving the problem.** EBL furth creates general rules depending on the problem to be solved which cover an entire class of cases.

For example: In case of differentiation, memorisation or memo function will remember th the derivative of x^2 with respect to x is $2x$ and would generalize that for any arithmetic unknow function; the derivative of u^2 is $2u$. Logically,

Arithmetic Unknown (u) → Derivative (u^2, u) = $2u$

Please understand that if the Knowledge Base (KB) contains such a rule then any new ca which is an instance of this rule can be solved immediately.

NOTE: This is an example of a very general phenomenon: "Once something is understood, it i generalized and reused in other circumstances".

This essential step then can be used as a building block in solving similar but complex problem

EBL Algorithm

EBL follows the following algorithm:

Step 1: Given an example (say of fork or of differentiation), construct a proof such that the go predicate applies to the example using the available background knowledge.

Step 2: In parallel, construct a generalized proof tree for the variablised goal using the same inferen steps as in the original proof.

Step 3: Construct a new rule whose left side consists of the leaves of the proof tree and whose rig hand side is the variablised goal.

Step 4: Drop any condition(s) which are true regardless of the values of the variables in the goal

fficiency

BL basically works in two steps: **explain** and **generalize.** Sometimes a rule can be generalized n a number of ways, which reflects the efficiency of the algorithm. Three factors contribute to its fficiency:

1. Adding large number of rules can slow down the reasoning process because the inference mechanism must still check those rules even in cases where they do not yield a solution. **Please note that it increases the branching factor in the search space.**
2. To compensate for the slowdown in reasoning process, the derived rules must offer significant increase in speed for the cases which they cover. This increase arises mainly because the derived rules avoid dead ends which would otherwise be taken.
3. Derived rules should be as general as possible so that they apply to the largest possible set of cases.

A common approach to ensure that derived rules are efficient is to insist on the **operationality** of each subgoal in the rule. **Also note that a subgoal is operational if it is easy to solve.** Unfortunately, there is unusually a trade–off between operationality and generality. **More specific subgoals are easier to solve but cover fewer cases only.** Also operationality is a matter of degree, one or 2 steps is definitely operational but may not hold good for 10 or 100 steps. Finally, the cost of solving a given subgoal depends on what other rules are available in the knowledge base. It can expand (up or down) as more and more rules are added. So, EBL systems face a really complex optimization problem in trying to maximize the efficiency of the given initial KB. It is also possible to derive a mathematical model of the effect on overall efficiency of adding a given rule and to use this model to select the best rule to add.

Please understand that when the recursive rules are involved, the analysis can become very complicated. One approach that is promising can be to address the problem of efficiency empirically, simply by adding several rules and then identifying which are useful and actually would speed up the process.

Conclusions

By generalizing from the past examples, EBL makes the KB more efficient for the kind of problems to which it can be applied. Hence, if this EBL system is carefully designed then it is possible to obtain better speed-ups.

NOTE: In 1991, Samuelson and Rayner showed that a reasonably large PROLOG-based natural language system designed for speech-to-speech translation between Swedish and English was able to achieve real-time performance only by the application of EBL to the parsing processes.

Advantages

1. EBL allows learning from a single training instance.
2. EBL forms generalizations that are relevant to specific goals.
3. Learner can select relevant aspects of training examples.

III. Symbol Based Learning (SBL)

SBL involves 3 tasks mainly—

1. **Specification of goal** and data.
2. **Representation of learned knowledge.** In learning problems, the concepts are represented as conjunctive sentences containing variables.
3. **Set of operations.** Once the training session is over, the learner should generate a heuristic rule or plan objecting its goal. This requires the ability to manipulate representations. The operations, here, include—

- Generalizing symbolic expressions.
- Specializing symbolic expressions.
- Adjusting weights in Neural Network (NN)
- Or Modifying the program's representation.

We create **generalizations** by replacing the constants with variables and **specializations** by replacing variables with constants.

IV. RBL/Deductive Learning

An American agent comes to India as a visitor. He/she firstly meets an Indian named Rajiv. On hearing his speech in Hindi, he concludes that all Indians speak Hindi. Note that this agent doesn't conclude that all Indian men are called as Rajiv. Hence, we can generalize this observation—

All Indians speak Hindi: Herein, the visitor-agent has some background knowledge like people in a given country, usually, speak the same language. In First-Order logic, this can be represented as–

$$\text{Nationality}(x, n) \wedge \text{Nationality}(y, n) \wedge \text{Language}(x, l) \rightarrow \text{language}(y, L) \quad \ldots(1)$$

i.e., if 'x', and 'y' have the same nationality 'n' and 'x' speaks language l, then y also speaks 'l'. But the observation, Nationality (Rajiv, Indian) ∧ Language (Rajiv, Hindi) along with the given knowledge shown in sentence 1, the following implication holds:

$$\text{Nationality}(x, \text{Indian}) \rightarrow \text{Language}(x, \text{Hindi})$$

Please understand that the sentences (as in equation (1)) **express a strict form of relevance that nationality of a person determines his/her language** or we can say that the language is a function of nationality. These sentences are called as **functional dependencies** (or FDs): In a syntax (given by Davis in 1985), it can be rewritten as—

$$\text{Nationality}(x, n) > — \text{Language}(x, l) \quad \ldots(2)$$

NOTE: In English, it means that nationality determines language

Equation (2) shows that the FD (or determination) is really a relationship between the **predicates—Nationality** and **Language.**

From equations (1) and (2) it is crystal clear that it becomes rudimentary that the prior, background knowledge concerns the relevance of a set of features to the goal predicate. This knowledge together with the observations, allows the agent to infer a new general rule, based on observations:

$$\text{Hypothesis} \wedge \text{Description} \in \text{Classifications}$$

$$\text{Background} \wedge \text{Descriptions} \wedge \text{Classification} \in \text{Hypothesis} \quad \ldots(3)$$

Please note that this type of generalization is called Relevance-based Learning (RBL). Also note that although RBL does make use of the content of the observation in creating hypothesis yet it does not generate hypotheses which go beyond the logical content of the background knowledge and the observations. This is a **deductive form of learning.** Because it cannot by itself amount for the creation of the new knowledge from a scratch, so it is also called as **relevance-based** or **deductive learning.**

FDs or determinations are very vital as they specify a sufficient basic vocabulary from which hypotheses concerning the goal predicate can be constructed. This can be proved by showing that a given FD is logically equivalent to a statement that the correct definition of the goal predicate is one of the set of definitions expressible using the predicates on the left hand side of the FD. Hence, reduction in the hypothesis space size should make it easier to learn the goal predicate. The gains can be qualified using a simple analysis—

If $|H|$ is the size of hypothesis space then for Boolean functions log ($|H|$) examples are required to arrive at a reasonable hypothesis. For a learner having n Boolean features participating in the construction of hypothesis—

$|H| = O(\log 2^{2^n})$. This means that the examples required will be $O(2^n)$. **Also note that if the determination contains d predicates in the left hand side, the learner will require only $O(2^d)$ examples.** This shows a **reduction of** $O(2^{n-d})$ as compared to learning through classification of examples.

7.4 GENETICS ALGORITHM (GA)

There is a need for a tool or a technique which reduces human dependency and decision making should be done based on same algorithm or a mathematical formula. Genetic Algorithms (GA) are **intrinsically parallel.** Most other algorithms are **serial** and can only explore the solution space to a problem in one direction at a time. GA has **multiple offspring** i.e., they can explore solution space in multiple directions at once. If one path turns out to be a dead end, they can easily eliminate it. GAs perform well in problems for which the fitness landscape is complex—ones where the fitness function is discontinuous, noisy, changes over time or has many **local optima.** Many search algorithms can become trapped by local optima if they reach the top of a hill on the fitness landscape. However, crossover is the key element that distinguishes GAs from other methods such as hill-climbers and simulated annealing . **Please note that without crossover, each individual solution is on its own, exploring the search space in its immediate vicinity without reference to what other individuals may have discovered. Also note that with crossover in place, there is a transfer of information between successful candidates.** Individuals can benefit from what others have learned and schemata can be mixed and combined with the potential to produce an offspring that has the strengths of both its parents. GAs can manipulate many parameters simultaneously. GAs use parallelism and hence can produce multiple equally good solutions to the same problems possibly with one candidate solution optimizing one parameter and another candidate optimizing a different one. A human overseer can then select one of these candidates to use.

Do you know? GAs know nothing about the problems they are deployed to solve. GAs make random changes to their candidate solutions and then use the fitness function to determine whether those changes produce an improvement.

7.5 NEURAL NETS

Neural computing is the study of networks of adaptable nodes which, through a process of learning from task examples, store experimental knowledge and make it available for use. The configurations and algorithms in neural computing are biologically inspired structures that perform useful tasks. **Please understand that the human nervous system consists of about 10^{11} neurons and 10^{15} interconnection. Also note that these numbers cannot be matched by an artificial NN, the tasks that can be performed by ANN are also severely limited.**

Fig. 7.3 shows the structure of a pair of biological neurons. **Dendrites extend from the cell body to other neurons where they receive signals at a connection point called a synapse.** On the receiving side of the synapse, these inputs are conducted to the cell body where they are summed, some inputs tending to excite the cell, others tending to inhibit its firing. When the cumulative excitation in the cell body exceeds a threshold, the cell fires, sending a signal down the axon to other neurons.

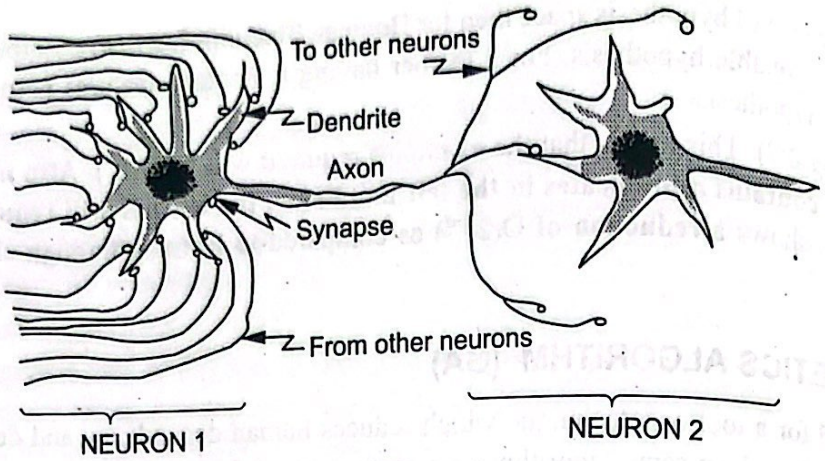


Fig. 7.3 Structure of a pair of biological neurons.

Basic computing unit

We consider a simple adaptable node (Fig. 7.4) which we will use to explain the features of ANN.

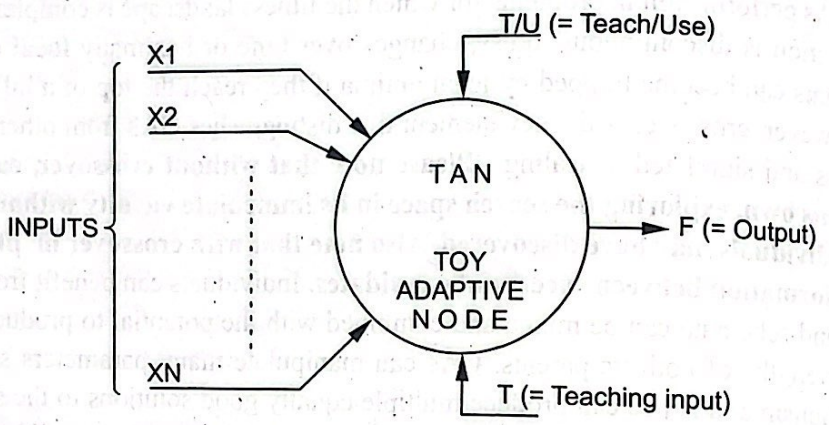
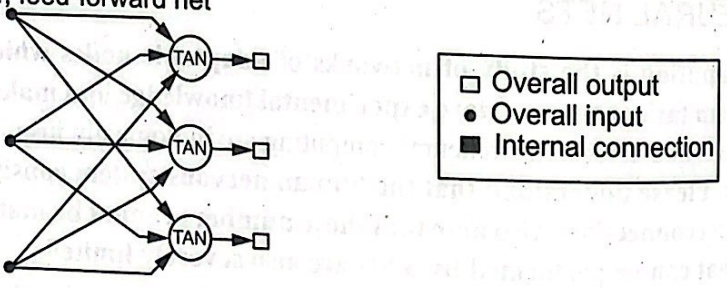


Fig. 7.4 A Toy Adaptive Node (TAN)

Network Structures

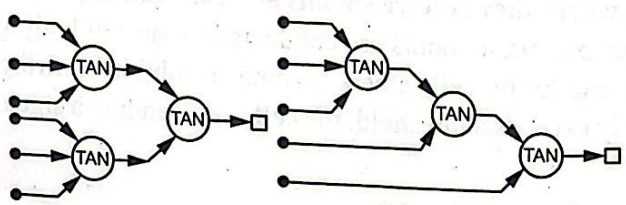
Fig. 7.5 shows some network structures using the basic computing elements (TANs). We use these structures to introduce some key ideas of NN, namely, associative nets, hidden layers, hard learning visible and hidden units. We also use them to explain the main features of NN, namely, distributed and located information, association and recall of information, and storing of knowledge.

(i) A single-layr, feed-forward net



(a)

(ii) Feed-forward nets



(b) (c)

(iii) Feedback nets

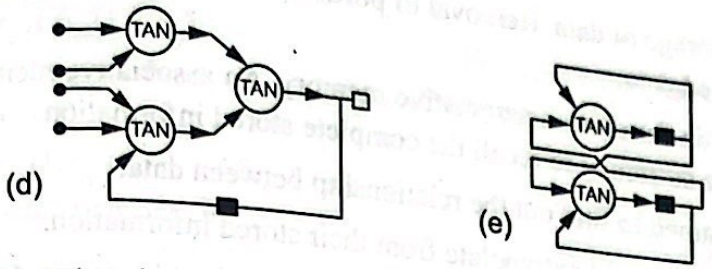


Fig. 7.5 Some network structures using the basic computing elements (TANs).

Neural net is a widely used technique for learning by charging the weights assigned to the nodes or arcs of a network. However, their name is a misnomer since they bear little resemblance to actual neural mechanisms. Fig. 7.6 shows a typical neural net—.

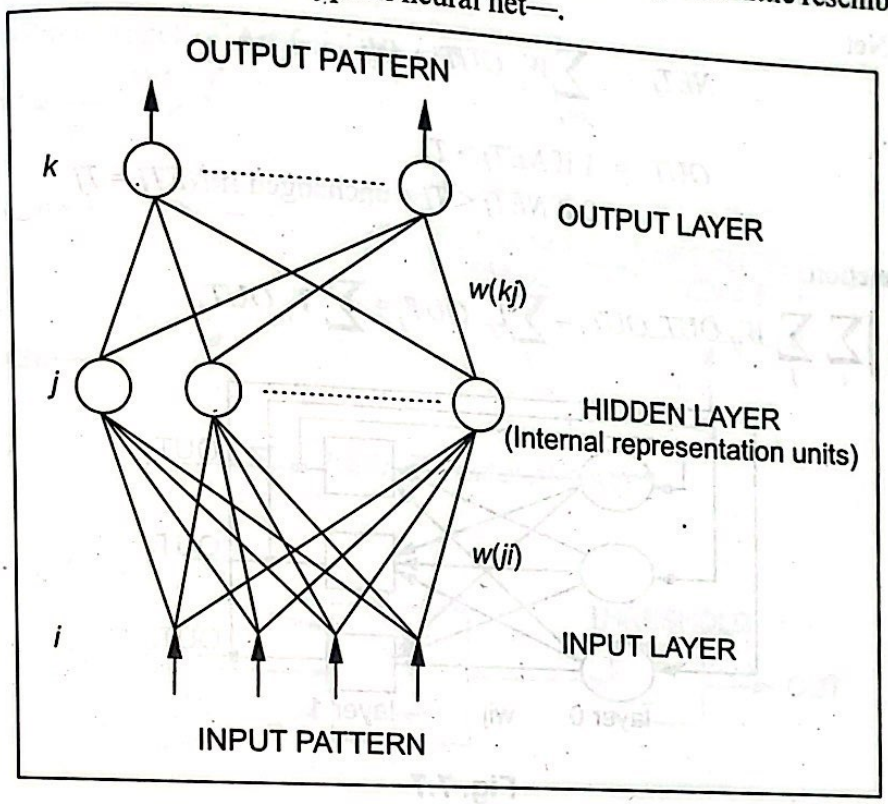


Fig. 7.6 Typical neural net.

Its input is a sequence of number that indicates the relative proposition of some selected features and whose output is another sequence of number that indicates the most likely concept characterized by that combination of features.

For example: in case of OCR-output character Recognizer, the **features** might represent lines, curves and angles and the concept might represent the letters that have those features.

Do you know? In a NN, the structure of nodes and arcs is fixed and the only changes that may occur are the assignment of weights to the arcs.

When a new input is presented, the weights on the arcs are combined with the weights on the outputs. In learning stage, the system is told whether the predicted weights are correct and various methods of back propagation are used to adjust the weight on the arcs that leads to the result.

Neural networks support digital computers with the ability to make sensible decision to learn by experience.

Main features of ANNs are as follows:

1. Distributed Storage of data. Removal of portions of net is possible. It does not degrade the quality of stored data.
2. Data is stored in form of an associative memory. An associative memory is one in which partial data are sufficient to recall the complete stored information.
3. Nets can be trained to find out the relationship between data.
4. Nets can interpolate and extrapolate from their stored information.
5. Quality of stored images degrades as some portion of net is removed.

APPLICATION OF NN

Travelling Salesman Problem

Hopfield Net

$$NETj = \sum_{i=j} W_{ij}\, OUTi + INj$$

$$OUT = 1 \text{ if } NETj > Tj$$
$$= 0 \text{ if } NETj < Tj \quad \text{unchanged if } NETj = Tj$$

Energy function

$$E = \left(-\frac{1}{2}\right)\sum_i \sum_j W_{ij}\, OUT_i\, OUT_j - \sum_j I_j\, OUT_j + \sum_j T_j\, OUT_j$$

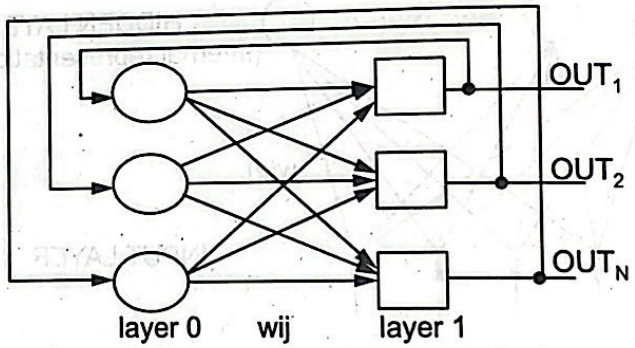


Fig. 7.7

Travelling Salesman Problem:

Order visited

City	1	2	3	4
A	0	1	0	0
B	0	0	0	1
C	1	0	0	0
D	0	0	1	0

E_1 –Energy fucntion must be low for only those solutions for which there is a single 1 in a column and row.

E_2 –Energy function must favour solutions having short distances.

$$E_1 = \frac{A}{2}\sum_x \sum_i \sum_{j \neq i} OUT_{xj} + \frac{B}{2}\sum_x \sum_i \sum_{y \neq x} OUT_{xi} + \frac{C}{2}\left[\left(\sum_x \sum_i OUT_{xi}\right)\right] - r$$

$$E_2 = \left(\frac{1}{2}\right) D \sum_x \sum_{y \neq x} \sum_i d_{xy}\, OUT_{xi}\left(OUT_{yi+1} + OUT_{yi-1}\right)$$

Weights:

$$W_{xi,yi} = A\,\delta_{xy}(1-\delta_{ij}) \quad \text{(Prevents more than a single 1 within a row)}$$
$$= B\delta_{ij}(1-\delta xy) \quad \text{(Prevents more than a single 1 within a column)}$$
$$= -C \quad \text{(Global inhibition)}$$
$$= D\,d_{xy}(\delta_{ji} + i + \delta_{ji-1}) \quad \text{(Distance term)}$$

Bias weight for each neuron Cn.

Activation function:

$$OUT = \left(\frac{1}{2}\right)\left[1+\tanh\left(\frac{NET}{U_0}\right)\right]$$

Perception

McCulloch-Pitts Model of Artificial Neuron

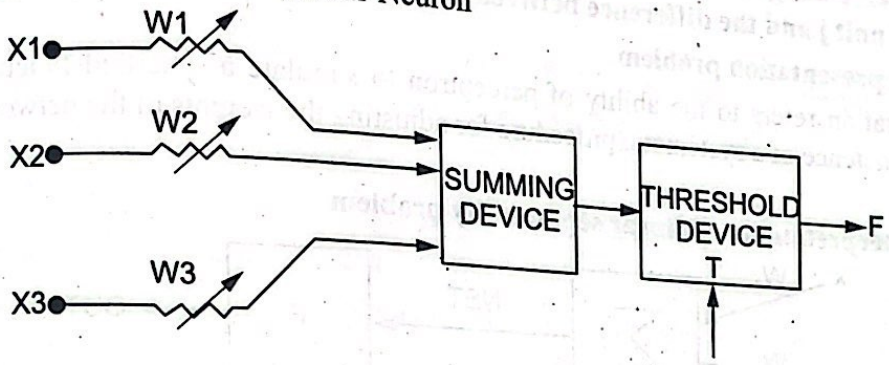


Fig. 7.8 M-P model of neuron.

Perceptron

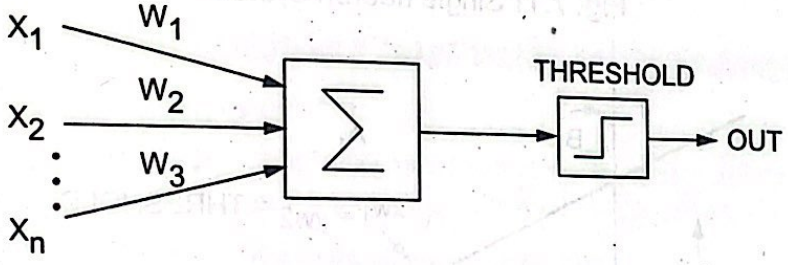


Fig. 7.9 Perceptron-M-P node with some additional, fixed preprocessing

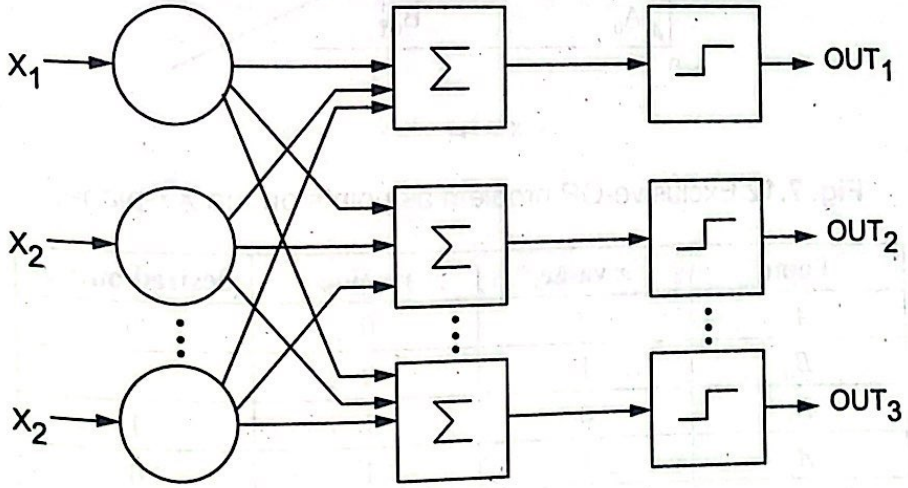


Fig. 7.10 Multioutput perceptron.

Perceptron classification problem

Given k vectors vi, $i = 1, \ldots k$, each of dimension n, suppose they are partitioned into two subsets I and I_1. The perceptron classification problem is:

Build a network consisting of threshold switching elements such that output is 1 for all input vectors in I+ and 0 for all input vectors in I.

Perceptron learning (training) problem

The **perceptron learning (training) problem** is:

Change the weights and thresholds of the various neurons adaptively until classification occurs.

Hebb's rule: Change in weight connecting unit i to unit j is proportional to the product of the output of unit j and the activation value of unit i.

Delta rule: Change in weight connecting unit j to unit i is proportional to the product of the output of unit j and the difference between the desired and actual outputs of unit i.

Perceptron representation problem

Representation refers to the ability of perceptron to simulate a specified function. Learning refers to the existence of a systematic procedure for adjusting the weights of the network to produce that function.

Geometric interpretation and linear separability problem

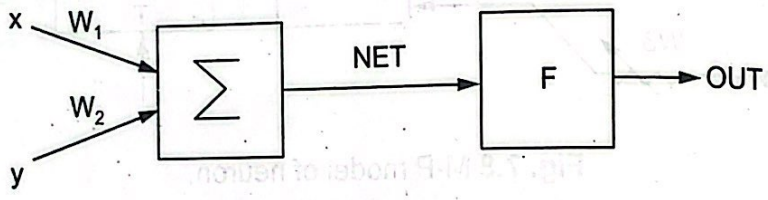


Fig. 7.11 Single neuron system.

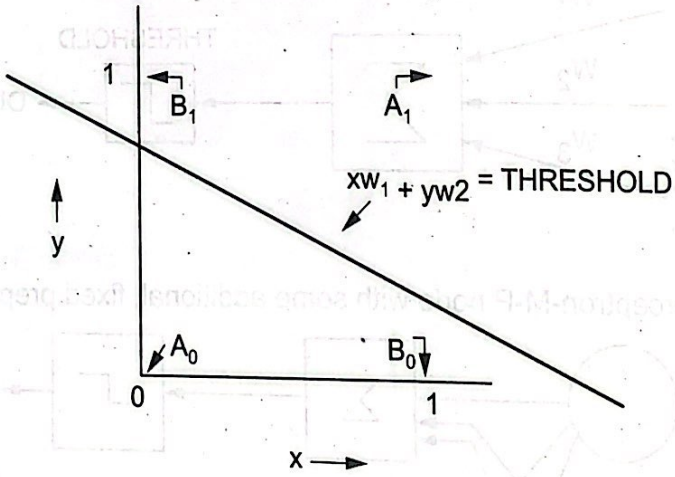


Fig. 7.12 Exclusive-OR problem as points on the XY-plane.

Point	x value	y value	desired output
A_0	0	0	0
B_0	1	0	1
B_1	0	1	1
A_1	1	1	0

Fig. 7.13 Exclusive-OR truth table.

SUMMARY

Learning is an incessant process. An intelligent machine should be able to learn new things. There are various ways.of learning means change in the system or its KB brought about its interaction with a teacher or external environment so that the system improves its performance on a class of tasks.

MULTIPLE CHOICE QUESTIONS [MCQS]

1. Saving knowledge so that it can be used again is known as
   (*a*) Rote learning (*b*) Induction (*c*) Deduction (*d*) None of the above.
2. Rote learning is based on the principle of
   (*a*) Caching (b) Forwarding (*c*) Backtracking (*d*) None of the above.
3. EBL uses
   (*a*) Data intensive approach (*b*) Knowledge intensive approach
   (*c*) Divide and conquer (*d*) None of the above.
4. Human nervous system consists of
   (*a*) 10^{10} neurons (*b*) 10^{11} neurons (*c*) 10^{12} neurons (*d*) 10^{14} neurons
5. The rule used to change weight is
   (*a*) Kirchoff's rule (*b*) Hebb's rule (*c*) Boehm's rule (*d*) None of the above.

ANSWERS

1. (*a*) 2. (*a*) 3. (*b*) 4. (*b*) 5. (*b*)

CONCEPTUAL SHORT QUESTIONS WITH ANSWERS

Q. 1. What are identification trees?

Ans. It is a learning method that uses tree representation of any problem indicating all possibilities. It is a sort of decision tree only. For example, for medical diagnoses, surveys etc it is used.

Q. 2. Draw the block diagram of a general learning system.

[RTM Nagpur Univ. B.Tech IT-7th Sem., 2009]

Ans. Figure below shows the block diagram of a general learning system.

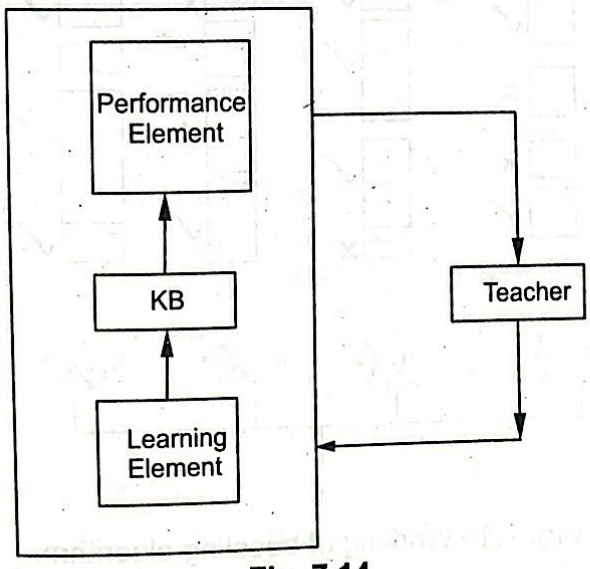


Fig. 7.14

There is some class of tasks that the system is expected to learn. The teacher watches the performance on some problem from this class and provides the system with an evaluation. Using this, the learning element modifies the current KB. Through this process of interaction, the system acquires the needed knowledge to improve its performance. **Please note that the performance element must be capable of solving the class of problems.**

Q. 3. What is learning by discovery?

Ans. A form of learning from examples where the teacher does not supply the answers. The learning system has to look for inherent regularities in the mass of data for inferring general rules.

For example: Discovery of Kepler's law. Kepler analysed the mass of data on planetary motion and came out with some 'beautiful' laws which generalise all the examples.

Q. 4. What is learning through clustering?

Ans. Clustering is the process of grouping or classifying objects on the basis of a close association or shared characteristics. It is essentially a discovery learning process in which similarities patterns are found among a group of objects.

Q. 5. What is matching?

Ans. The process of comparing two structures or patterns for equality is known as matching. It may serve to control the sequence of operations, to identify or classify objects or to retrieve items from a database.

Q. 6. Discuss Boosting Algorithm of Ensemble Learning.

Ans. The most widely used ensemble method is called **boosting.** It works on the concept of a **weighted training set.** In such a training set, each example has an associated weight, $w_i \geq 0$. **Please note that the higher is the weight of an example, the higher is the importance attached to it during the learning of a hypothesis.**

Working Boosting starts with $w_j = 1$ for all examples, i.e., training set. From this the first hypothesis, h_f, is generated. This hypothesis will classify some of the training examples correctly and some incorrectly. In order for the next hypothesis to perform better on the misclassified example, the weights of the misclassified ones are increased. From this new weighted training set, the hypothesis, h_2, is generated.

The process is continued until M hypothesis have been generated where M is the input of the boosting algorithm. **The final ensemble hypothesis** is a weighted–majority combination of M hypothesis, each weighted according to how it will perform on the training set. The process is shown below (Fig. 7.15).

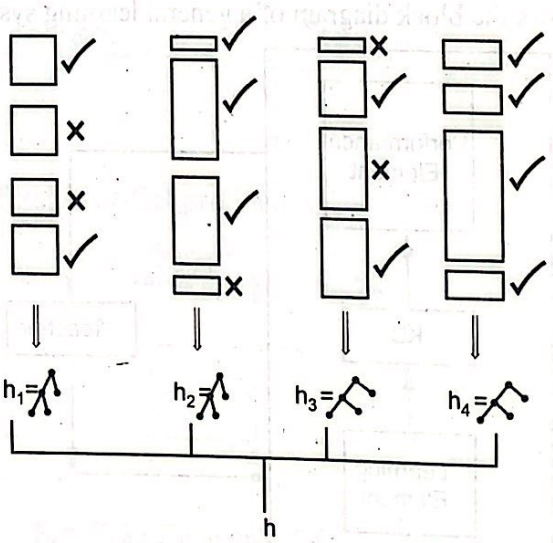


Fig. 7.15 Working of boosting algorithm.

1. Each **rectangle** in the figure corresponds to an example.
2. The **height** of the rectangle corresponds to the **weight.**
3. The **ticks** and **crosses** indicate whether the example is classified correctly by the current hypothesis in the final ensemble.

It has been observed that the predictions improve as the ensemble hypothesis gets more plex.

Next, we review the premise with which we started, i.e., the example pair $(x, f(x))$, i.e., how do now that the hypothesis, h, is close to the target function $f(x)$ if we do not know f. The answer to is based on **Computational Learning Theory,** a field that includes AI, statistics and theoretical puter science.

Principle: "Any hypothesis which is seriously wrong will almost certainly be found out high probability after a small number of examples because it will make on incorrect diction".

Please understand that any hypothesis which is consistent with a sufficiently large set of ining examples is unlikely to be seriously wrong, i.e., it must be probably approximately correct C). **Any learning algorithm which returns hypotheses which are probably approximately rect is called as a PAC–learning algorithm. Also understand that this leads to one important terion on the connection between the training and the test examples, the hypothesis is to be proximately correct on the test set, not on just training set. The basic assumption** is that the ning and test sets are drawn randomly and independently from the same population of examples h the same probability distribution. **This is known as a stationary assumption.** This assumption mportant **because without this assumption the ensemble learning cannot predict at all.**

EXERCISE QUESTIONS

Q.1. (*a*) Explain artificial neurons. Use necessary diagram.

(*b*) Calculate the logic function 'AND' or "OR" using McCulloch-Pitts neurons.

(*c*) Explain the Perceptron Training algorithm.

[UPSAT, B. Tech., (CSE)-8th Sem., Oct. 2004]

Q.2. What is learning? Explain Rote Learning and Induction Learning.

[RTM Nagpur Univ., BE (CSE)- 7th Sem., Summer 2009]

Q.3. (*a*) What is declarative knowledge?

(*b*) What is baring by induction? **[PTU., M.C.A., 4th Sem., 2009]**

Q.4. Design a learning automation that selects TV channels based on day of week and time of day (3 evening hours only) based on preference of your family. (Specify the preferences of your family clearly). **[GGSIPU., B.Tech. (CSE)- 8th Sem., May 2008]**

Q.5. What are different types of learning. **[GGSIPU., B.Tech. (CSE)- 8th Sem., May-June 2009]**

Q.6. (*a*) Give a short description of process of learning using Back Propagation NN.

(*b*) Give a flow graph for developing a genetic algorithm. **[GGSIPU., B.Tech. (CSE)- 8th Sem., May 2010]**

Q. 7. (*a*) Describe an Automation. Extend it to learning automation.

(*b*) Pick one problem area that you feel would justify the energy required to design an expert system solution and spell out the problem in some detail. Based on your intuition, which aspects of this solution would be most difficult to automate?

Q. 8. (*a*) What are the different types of learning? Explain with an example.

(*b*) Why is the process of knowledge acquisition so tedious?

[GGSIPU., B.Tech. (CSE)- 8th Sem., May 2011]