Module 5

CHAPTER 5 Planning and Learning

Syllabus

The planning problem, Planning with state space search, Partial order planning, Hierarchical planning, Conditional Planning.

Learning : Forms of Learning, Theory of Learning, PAC learning. Introduction to statistical learning (Introduction only)

Introduction to reinforcement learning: Learning from Rewards, Passive Reinforcement Learning, Active reinforcement Learning

5.1 COMPONENTS OF PLANNING

GQ. Explain the different components of planning system. **OR** Given the components of planning system and briefly explain them. **OR** Explain the various components of planning system. How can you represent a planning action?

Different components of planning system are as follows :

1. Choose best rule to apply
2. Apply the chosen rule
3. Detecting when a solution has been found
4. Detecting dead ends
5. Repairing an almost correct solution

- **1. Choose best rule :** Isolate set of differences between the desired goal state and the current state, identify those rules that are relevant to reducing those differences (means-end analysis). If several rules found then choose the best using heuristic information.
- **2. Apply rules :** In simple systems, applying rules is easy. Each rule simply has specified the problem state that would result from its application. In complex systems, we must be able to deal with rules that specify only a small part of the complete problem state. One way is to describe, for each action, each of the changes it makes to the state description.
- **3. Detecting a solution :** A planning system has succeeded in finding a solution to a problem when it has found a sequence of operators that transform the initial problem state into the goal state. In simple problem-solving systems we know the solution by a straightforward match of the state description. But, in complex problem different reasoning mechanisms can be used to describe the problem states, that reasoning mechanisms could be used to discover when a solution had been found.
- **4. Detecting dead ends :** A planning system must be able to detect when it is exploring a path that can never lead to a solution. Above same reasoning mechanism can be used to detect dead ends. In search process reasoning forward from initial state, it can prune any path that lead to a state from which goal state cannot be reached. Similarly backward reasoning, some states can be pruned from the search space.
- **5. Repairing an almost correct solution :** In completely decomposable problems can be solve the sub problems and combine the sub solutions yield a solution to the original problem. But try to solving nearly decomposable problems one way is use means-ends analysis technique to minimize the difference between initial states to goal state. One of the better ways to represent knowledge about what went wrong and then apply a direct patch.

5.1.1 Problem Solving Vs. Planning

GQ. Compare problem solving and planning.

Sr. No.	Planning	Problem Solving
1.	The task coming up with a sequence of actions that will achieve a goal is called planning.	Problem solving is the systematic search through a range of possible actions in order to reach to some predefined goal solution.
2.	Planning is typically viewed as a generic term of problem solving because it deals with search in an abstract level.	Problem solving comprises standard search process.
3.	Planning is involved in plan generation.	Problem solving is more concerned with plan execution.
4.	Planner is viewed as the producer or generator of the solution.	Problem solving just demonstrates one specific solution.
5.	Backward planning is typical for planning.	In problem solving usually forward approach is adopted.
6.	The pre-requirements are almost same for variety of problems in planning.	Problem solving is of two types – special purpose and general purpose. A special purpose is for one problem. General purpose is for variety of problems.
7.	STRIPS (Standard Research Institute problem solver), PDDL (Planning domain definition language) are some examples for planning systems.	A* algorithm is an example.

Module 5

- A simple planning agent is very similar to problem-solving agents in that it constructs plans that achieve its goals, and then executes them. The limitations of the problem solving approach motivates the design of planning systems.

☞ **To solve a planning problem using a state-space search approach we would let the**

- Initial state = initial situation
- Goal-test predicate = goal state description
- Successor function computed from the set of operators.
- Once a goal is found, solution plan is the sequence of operators in the path from the start node to the goal node.

In searches, operators are used simply to generate successor states and we cannot look "inside" an operator to see how it's defined. The goal-test predicate also is used as a "black box" to test if a state is a goal or not. The search cannot use properties of how a goal is defined in order to reason about finding path to that goal.

Planning is considered different from problem solving because of the difference in the way they represent states, goals, actions, and the differences in the way they construct action sequences.

☞ **Remember the search-based problem solver had four basic elements**

- **Representations of actions :** Programs that develop successor state descriptions which represent actions.
- **Representation of state :** Every state description is complete. This is because a complete description of the initial state is given, and actions are represented by a program.
- **Representation of goals :** A problem solving agent has only information about its goal which is in terms of a goal test and the heuristic function.
- **Representation of plans :** In problem solving, the solution is a sequence of actions.

☞ **Cordless drill for a problem solving exercise we need to specify**

1. **Initial state :** The agent is at home without any objects that he is wanting.
2. **Operator set :** Everything the agent can do.

5.2 PROBLEMS ASSOCIATED WITH PLANNING AGENT

GQ. Write short notes on : (i) Planning agent (ii) State goal and action representation.

5.2.1 Planning Agent

- The actual branching factor would be in the thousands or millions. The heuristic evaluation function can only choose states to determine which one is closer to the goal. It cannot eliminate actions from consideration.
- The agent makes guesses by considering actions and the evaluation function ranks those guesses. The agent picks the best guess, but then has no idea what to try next and therefore starts guessing again.
- It considers sequences of actions beginning from the initial state. The agent is forced to decide what to do in the initial state first, where possible choices are to go to any of the next places. Until the agent decides how to acquire the objects, it can't decide where to go. Planning emphasizes what is in operator and goal representations.

5.2.2 Three Key Ideas behind Planning

There are Three Key Ideas behind Planning

- To "open up" the representations of state, goals, and operators so that a reasoner can more intelligently select actions when they are needed.
- The planner is free to add actions to the plan wherever they are needed, rather than in an incremental sequence starting at the initial state.
- Most parts of the world are independent of most other parts which makes it feasible to take a conjunctive goal and solve it with a divide-and-conquer strategy.
- An intelligent agent can act independently and has well-defined goals. It can adapt its behavior to its environment "a general-purpose system that, like a human, can perform a variety of different tasks under conditions that may not be known a priori." An agent must be aware of the user's goals and may have to behave in a changing world.

5.2.3 Two Types of Agents

1. Physical agents (usually known as robots)
2. Software agents (sometimes known as softbots)

1. Physical agents

These are physical artefacts and act in a physical environment, e.g. send a physical agent into a dangerous building. It must be able to see, it must know where it is, it must be able to move and search, plan its goals, execute its goals (physically), re-plan if necessary, communicate with other (possibly human) agents.

☞ **Consist of some or all of the following :**

(a) **Computers**

Top-level controller

Low-level controller, e.g. to manipulate a hand

(b) **Sensors**

Establish contact / non-contact with objects in the environment

To "see"

(c) **Effectors**

"arms", "hands", "feet"

(d) **Auxiliary equipment**

Tools, containers to put things in, tables to put things on

Stages of development of physical agents :

- Slave manipulator operated by human master
- Limited sequence manipulator (hard to adjust)
- Teach-replay robot
- Computer-controlled robot
- Intelligent robot

Incentives for development of physical agents

- Job undesirability
- Lethal (radiation)
- Harmful (paint spraying, chemical handling)
- Risky (fire-fighting, combat)
- Strenuous (heavy loads, visual inspection)
- Noisy (riveting, hammering, forging)
- Boring (sorting, assembling)

2. Software agents

These are software programs and act in computers or computer networks.

Simple examples are programs that sort your incoming mail according to a given priority, that store documents in the correct folder (previously done by a human agent, a secretary).

☞ **Three Laws of Robotics**

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.
- A robot may not harm humanity, or, by inaction, allow humanity to come to harm.

☞ **Agents and goals**

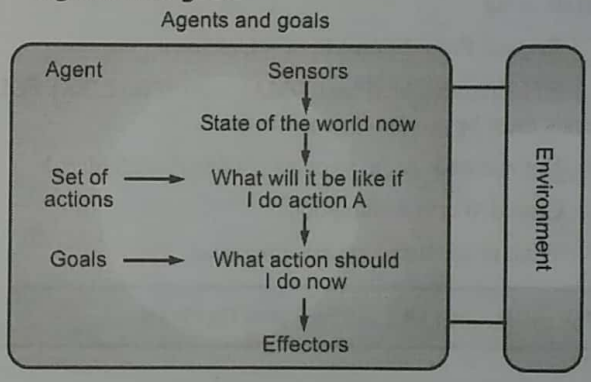


Fig. 5.2.1 : Software agents

(i) **Machine translation (MT)** is the task of automatically converting one natural language into another, preserving the meaning of the input text, and producing fluent text in the output language.

(ii) **State, goal and action representation :** Describe state, goal and action represents to plan, one should be able to represent the problem properly. Representation of the planning problem is **mapping of the states, actions and the goals**. For the representation, the language used should be concrete, understandable and expressive. In a broader perspective, there are different representation methods or ways that are followed like propositional, first order, state variable. Fig. 5.2.2 depicts the high-level diagram for planning.

Module 5

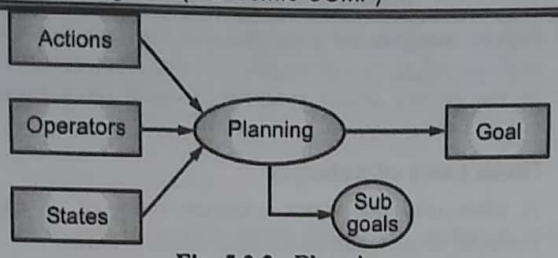


Fig. 5.2.2 : Planning

Planning essentially needs the representation in terms of **time**. This is required so that we are able to reason regarding the actions that are to be taken along with the reactions that we get back.

(iii) State Representation : States are the representation of the facts. States are represented as conjunction. It comprises the positive literals that specify the state.

A state is represented with a conjunction of positive literals using :

Logical Propositions: Poor ∧ Unknown

FOL literals: At (Plan1,OMA) ∧ At (Plan2,JFK) FOL literals must be ground & function-free

Not allowed: At (x, y) or At (Father(Fred),Sydney)

Closed World Assumption :

What is not stated are assumed false.

5.3 GOAL REPRESENTATION

Goal is most often a partially specified State. A state or say proposition is said to achieve or satisfy the given goal if it consists of all the objects required for the goal or may by some other too. As an example, if the goal kind ∧ hardworking, then a state that has kind ∧ hardworking ∧ pretty fulfils the goal.

Example : Rich ∧ Famous ∧ Miserable satisfies the goal Rich ∧ Famous

☞ Action representation

Whenever we decide to do something, we are aware of the state we are in and what possibly the effects are. When it comes to the mapping of the actions for an agent or in case of robots, the pre and the post situations need to be specified. These can also be called as **preconditions** and the after effects are called **post-conditions**.

For example, an action to drive from one place to another can be mapped as follows :

- Action<drive(c,from.to)
- Pre-condition: at(c,from)^car(c)
- Post-condition: ~at<c, from)^at(c, to)]

Action Representation in this case is :

- Action Schema
  → Action name
  → Precondition
  → Effects
- **Example :**
  Action(Fly (p. from to),
  →Pre-condition: At (p from)∧plane (p) ∧Airport(from) ∧Airport (to)
  → Effect: At (p, from) ∧At(p.to))
- Sometimes, Effects are split into ADD list and DELETE list

  At (WHI, LNK), Plane(WHI)
  Airpost (LNK) Airport (OHA)
  Fly(WHI, LNK OHA)
  At (WHI, OHA)-At (WHI,LNK)

Here, in the post-condition, where we have written **~at(c, from)** which indicates that; the state is deleted or is to be removed. In some cases, add and delete list can also be used.

In case of a slate variable representation, the state comprises of different state variables. The action here is defined as a partial function on the states.

☞ The actions can be actually applied using following criteria

1. A substitution is to be identified for the variables. That is, for the current state that exists, identify the action with a pre-condition that satisfies the current state (the current state can be subset of the pre-condition).
2. Apply this substitution (for whatever part of the current state it is applicable).
3. Add the post-condition (effects) to the remaining subset of the current state if any.

5.4 PLANNING WITH STATE SPACE RESEARCH

We highlight the main planning method to solve AI problems

State space search is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or states of an instance are considered, with **the intention of finding a goal state with the desired property**.

For finding the solution one can make use of explicit search tree that is generated by the initial state and the successor function that together define the state space

Construction of State Space

(i) The root of search tree is a search node corresponding to initial state. In this state we can check whether the goal is reached.

(ii) If goal is not reached we consider another state. This can be done by expanding from the current state by applying successor function which generates new state. And from this, we get multiple states.

(iii) For each one of these, again we need to check goal test or repeat expansion of each state.

(iv) The choice of which state to expand is determined by the search strategy.

(v) It is possible that some state cannot lead to goal state such a state we should not expand.

Example : state-search example :

Making coffee

(1) Take some of the boiled water in a cup and add necessary amount of instant coffee powder to make decoction.

(2) Add milk powder to the remaining boiling water to make milk.

(3) Mix decoction and milk.

(4) Add sufficient quantify of sugar and the coffee is ready.

The most straight forward approach is to use state-space search.

Since the descriptions of actions in a planning problem specify both pre conditions and effects, it is just possible to each along both the directions :

(1) Forward from the initial state or

(2) backward from the goal

5.4.1 Forward State-Space Search

This search is also called as progressive planning, because it moves in the forward direction. It is similar to the problem-solving approach.

We begin with the problem's initial state, considering sequences of actions until we reach a goal state.

The formulation of state-space search planning problem is as follows

(a) Forward search is **an algorithm that searches forward from the initial state of the world to try to find a state that satisfies the goal formula.**

(b) The initial state of the search is the initial state from the planning problem here, each state will be set of position ground literals; not appearing literals are taken as false.

(c) The actions which are applicable to a state are all those whose preconditions are satisfied. Adding the positive effect literals and deleting the negative effect literals, the successor sate is generated from the action.

(d) The goal test chocks whether the state satisfies the goal of the planning problem.

(e) The step cost of each action is taken as 1. Different cost for different actions may be allowed.

(f) Forward state-space search is not very practicable. It is because of a big branching factor. Forward search considers all applicable actions, (i.e. all relevant and non-relevant actions are considered).

(g) A forward planner searches the state-space graph from the initial state to the goal-description.

State Representation

- A state space essentially consists of a set of nodes representing each state of the problem, arcs between nodes represent the moves from one state to another, an initial state and a goal state.
- Each state space takes the form of a tree or a graph.
- Factors that determine which search algorithm or technique will be used include the type of the problem and how the problem can be represented.
- We have presented state-space search as a forward search method, but it is also possible to search backward from the set of states that satisfy the goal to the initial state.

Module 5

5.4.2 Backward State-Space Search

(a) Backward state space search planning is we want to generate possible predecessors of a given goal state, work backwards toward the initial state.

(b) If a solution exists, it should be found by a backward search which **allows only relevant action**

(c) The meaning or restriction to relevant actions only means that backward search often has a much lower branching factor than forward search.

(d) Goal states are often incompletely specified. It expresses only what is desired in the final state, rather than a complete description of the final state.

(e) It is also called as Regression planning because Backward state-space search will find the solution from goal to the action.

(f) If there are many known states possible to be reached from goal state. Reaching to any one of those nodes, we can reach the start node.

(g) In general, backward search works only when we know how to regress form a state description to the predecessor state description. For example, it is hard to search backwards for a solution to the n-queens problem because these is no easy way to describe the states that are one move away from the goal.

(h) To obtain full advantage of backward search, we need to deal with partially uninstantiated actions and states.

For example, suppose the goal is to **deliver a specific piece of cargo to Delhi** :

This suggests the action unload (C, P, Delhi) Thus, Action (unload (C, P. Delhi))

Precondition : In (C, P) $\wedge$ (P, Delhi) $\wedge$ cargo (C) $\wedge$ plane (P) $\wedge$ Airport (Delhi)

Effect : At (C, Delhi) $\wedge \neg$ In (C, P)

5.5 PARTIAL ORDER PLANNING

UQ. Explain a partial order planner with an example.
(MU - Q. 5(c), Dec. 15, Q. 4(b), May 17, Q. 5(a), May 2019, 10 Marks)

- Partial order planning is an approach to automated planning that maintains a partial ordering between actions and only commits ordering between actions when the actions are partial. Also the planning does not specify which action will come out first when two actions are processed.
- A partial order plan or partial plan which specifies all actions that need to be taken, but only specifies the order between actions when necessary, is the result of a partial order planner.
- This is sometimes also called **a non-linear planner**, which is a misnomer because such planners often produce a linear plan.
- A partial ordering is a less-than relation that is transitive and asymmetric.

Partial order planner components

1. **A set of actions** (also known as operators).
2. **A partial order :** A partial-order plan is a set of actions together with a partial ordering, representing a "before" relation on actions, such that any total ordering of the actions, consistent with the partial ordering, will solve the goal from the initial state.
3. **A set of causal links :** It specifies which actions meet which preconditions of other actions. Alternately, a set of bindings between the variables in action. Write $\mathbf{act_o < act_1}$ if action act_0 is before action act_1 occurs in the partial order.
4. **A set of open preconditions** : For **uniformity**, treat **start** as an action that achieves the relations that are true in the initial state, and treat **finish** as an action whose precondition is the goal to be solved. The **pseudo action start** is before every other action, and finish is after every other action. The use of these as actions means that; the algorithm does not require special cases for the initial situation and for the goals. When the preconditions of finish hold, the goal is solved.
5. In order to keep the possible orders of the actions as open as possible, the set of other conditions and **causal links** must be as small as possible. A plan is itself a solution if the set of open preconditions is empty.
6. An action, other than start or finish, will be in a **partial-order plan** to achieve a precondition of an action in the plan. Each precondition of an action in the plan is either true in the initial state or so achieved by start, or there will be an action in the plan that achieves it.
7. We must ensure that the actions achieve the conditions they were assigned to achieve. Each **precondition P** of an **action** $\mathbf{act_1}$ in a plan will have an **action** $\mathbf{act_0}$ associated with it such that $\mathbf{act_0}$ **achieves precondition P for** $\mathbf{act_1}$.

8. The triple (act_0, **P**, act_1) is a **causal link**. The partial order specifies that action act_0 occurs before action act_1, which is written as $act_0 < act_1$. Any other action A that makes P false must either be before act_0 or after act_1.

☞ **Linearization of partial order plan**

A linearization of partial order plan is a **total order plan** derived from the particular partial order plan, in other words, both order plans consist of the same actions, with the order in the **linearization** being a **linear extension** of the partial order in the original partial order plan.

For example : A plan for baking a cake might start as follows :

- Go to store,
- Obtain flour, get milk, etc.
- Pay for all the goods,
- Go to kitchen.

This is a partial plan because the order for finding flour and milk is not specified; the agent can wander around the store, accumulating all the items as its shopping list is complete.

5.5.1 Causal Link in Partial Order Planning

- Each causal link specifies a **pair of steps and a proposition**, where the proposition is a post condition of the first step and precondition of the second step. The first step is ordered before the second step.
- If a precondition of a step is not supported by a causal link, then it is a flaw in a partial-order plan.
- Any planning algorithm that can place two actions into a plan without specifying which should come first is called partial-order planner.
- **POP (Partial-order-plan) :** is a regression planner; it uses problem decomposition; it searches plan space rather than state space; it builds partially-ordered plans; and it operates by the principle of least commitment.

5.5.2 Working of a Partial Order Planner

GQ. How does partial order planner work ?

1. Begin with the actions start and finish and the partial order **start < finish**.
2. The planner maintains an agenda that is a set of **(P, A)** pairs, where A is an action in the plan and P is an atom that is a precondition of A that must be achieved. Initially the agenda contains pairs (**G, finish**), where G is an atom that must be true in the goal state.
3. At each stage in the planning process, a pair (G, act_1) is selected from the agenda, where P is a precondition for action act_1.
4. Then an action, act_0, is chosen to achieve P. That action is either already in the plan it could be the start action, or it is a new action that is added to the plan. Action act_0 must happen before act_1 in the partial order. It adds a causal link that records that act_0 achieves **P for action act_1**. Any action in the plan that deletes P must happen either before act_0 or after act_1.
5. If act_0 is a new action, its preconditions are added to the agenda, and the process continues until the agenda is empty. This is a non-deterministic procedure. The "**choose**" and the "**either ...or ...**" form choices that must be searched over.
6. **POP :** is a **regression planner**; it uses problem decomposition; it searches plan space rather than state space; it builds partially-ordered plans; and it operates by the principle of least-commitment.

☞ **A plan in POP (whether it be a finished one or an unfinished one) comprises of following**

- **A set of plan steps:** Each of these is a STRIPS operator, but with the variables instantiated.
- **A set of ordering constraints:** $S_i < S_j$ means step Si must occur sometime before S_j (not necessarily immediately before).
- **A set of causal links:** S_i c $\rightarrow$ S_j means step Si achieves precondition c of step S_j.

So, it comprises actions (steps) with constraints (for ordering and causality) on them. The algorithm needs to start off with an initial plan. This is an unfinished plan, which we will refine until we reach a solution plan.

The initial plan comprises two dummy steps, called **Start and Finish**. Start is a step with no preconditions, only effects: the effects are the initial state of the world. Finish is a step with no effects, only preconditions: the **preconditions** are the goal.

5.5.3 Example for Partial Order Planner

GQ. Given an example for partial order planner.

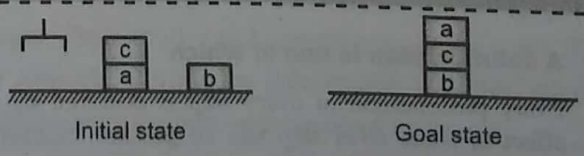


Fig. 5.5.1 : Partial order planner

Module 5

The above initial state can be represented in POP as the following initial plan :

Plan(STEPS: {S1: Op(ACTION: Start, EFFECT: clear(b) ∧ clear(c) ∧ on(c, a) ∧ ONTABLE(a) ∧ ONTABLE(b) ∧ ARMEMPTY), S2: Op(ACTION: Finish, PRECOND: on(c, b) ∧ on(a, c))}, ORDERINGS: {S1 < S2}, LINKS: {})

This initial plan is refined using POP's plan refinement operators. As we apply them, they will take us from an unfinished plan to a less and less unfinished plan, and ultimately to a solution plan.

☞ **There are four operators, falling into two groups**

1. **Goal achievement operators – Step addition** : Add a new **step S_i** which has an effect c that can achieve an as yet unachieved precondition of an existing step Sj . Also add the following constraints**: Si < Sj and Si c– → Sj and Start < Si < Finish**. – Use an effect c of an existing step Si to achieve an as yet unachieved precondition of another existing step Sj . And add just two constraints: **Si < Sj and Si c – → Sj** .
2. **Causal links:** must be protected from threats, i.e. steps that delete (**or negate or clobber**) the protected condition. If S threatens link **Si c – → Sj : – PROMOTE:** add the **constraint S < Si;** or – **DEMOTE:** add the constraint Sj < S the goal achievement operators ought to be obvious enough. They find preconditions of steps in the unfinished plan that is not yet achieved.

The two goal achievement operators remedy this either by adding a new step whose effect achieves the precondition, or by exploiting one of the effects of a step that is already in the plan.

The promotion and demotion operators may be less clear. Why are these needed? POP uses problem-decomposition: faced with a conjunctive precondition, it uses goal achievement on each conjunct separately. But, as we know, this brings the risk that the steps we add when achieving one part of a precondition might interfere with the achievement of another precondition.

And the idea of promotion and demotion is to add ordering constraints so that the step cannot interfere with the achievement of the precondition. Finally, we have to be able to recognize when we have reached a solution plan : a finished plan.

☞ **A solution plan is one in which**

- Every precondition of every step is achieved by the effect of some other step and all possible clobberers have been suitably demoted or promoted.
- There are no contradictions in the ordering constraints, e.g. disallowed is **Si < Sj and Sj < Si** ; also disallowed is **Si < Sj , Sj < Sk** and **Sk < Si** .

The solutions may still be **partially-ordered**. This retains flexibility for as long as possible. Only immediately prior to execution will the plan need linearization, i.e. the imposition of arbitrary ordering constraints on steps that are not yet ordered. (In fact, if there's more than one agent, or if there's a single agent but it is capable of multitasking, then some linearization can be avoided: steps can be carried out in parallel.)

⏭ 5.6 HIERARCHICAL PLANNING

- Take an unachieved precondition from the plan; achieve it. Resolve any threats using promotion or demotion. But, what the above fails to show is that planning involves search.
- At certain points in the algorithm, the planner will be faced with choices (alternative ways of refining the current unfinished plan). POP must try one of them but have the option of returning to explore the others.

☞ **There are basically two main 'choice points' in the algorithm**

- In **goal achievement**, a condition c might be achievable by any one of a number of new steps and/or existing steps. For each way of achieving c, a new version of the plan must be created and placed on the agenda.
- **Question :** A condition c might be achievable by new steps or existing steps. When placing these alternatives on the agenda, might be we arrange for the latter to come off the agenda before the former?
- When resolving threats, POP must choose between demotion and promotion. All preconditions must eventually be achieved, and so these aren't alternatives. The choice can be made irrevocably.
- Provided implementation of POP uses a complete and optimal search strategy, then POP itself is complete and optimal. However, **POP's branching factor** can still be high and the unfinished plans that we store on the agenda can be quite large data structures, so we typically abandon completeness/optimality to keep time and space more manageable.

Search strategies that are more like depth-first search might be preferable. And we might use heuristics to order alternatives or even to prune the agenda.

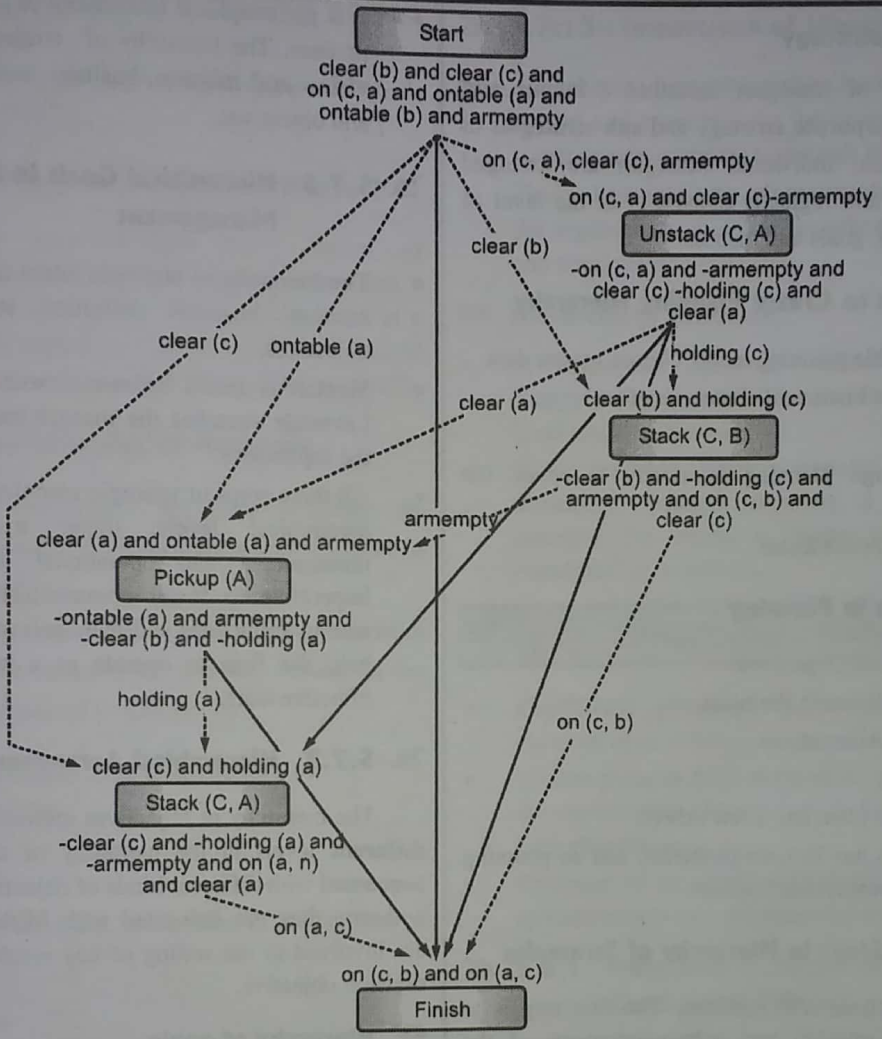


Fig. 5.6.1 : Hierarchical planning

5.7 HIERARCHICAL TASK NETWORK PLANNING

- In an Artificial Intelligence hierarchical task network planning is an approach to **automated planning** in which the dependency among actions can be given in the form of hierarchically structured networks.
- A hierarchical planning represents the **organisational levels** and units in the company for which one wants to plan. A planning hierarchy is a **combination of characteristic values** based on the characteristics of one information structure. You can create only one planning hierarchy for an information structure.
- Planning formed and executed by several subsystems organised according to a layered structure for the system goal.

System

System when applied to production and operations management, the process is called a **hierarchical production planning and control** process, which is the essence of a manufacturing planning and control (**MPC**) system. Also, these decisions are usually made by the upper levels of the management hierarchy.

Hierarchical plan consists of :

There are 3 types of hierarchical plans. They are strategic, administrative and operating (technical core) .The three hierarchical plans are independent, and they support the fulfilment of the three organisational needs.

Module 5

Hierarchical strategy

The hierarchy of strategies describes a **layout and relations** of **corporate strategy** and **sub strategies** of the organisation. Individual strategies are arranged hierarchically and logically consistent at the level of vision, mission, goals and metrics.

5.7.1 Steps to Create Planning Hierarchy

1. From the flexible planning menu, choose master data.
2. Leave all fields blank and choose 'confirm'
3. Choose 'save'
4. Choose: Change planning Hierarchy to create the hierarchy.
5. To continue, choose Enter

5.7.2 Steps in Planning

1. Determination of Objectives,
2. Constructing Planning Premises,
3. Evaluation of Alternatives.
4. Selecting plan,
5. Controlling the plan, and a few others.

Every business has its own problems, and so planning details differ from business to business.

5.7.3 Key Steps in Hierarchy of Strategies

Strategic objectives and Analysis. The first step is to define the vision, mission and values-statements of the organisation.

1. Strategic formulation.
2. Strategic implementation
3. Strategic Evaluation and control.

5.7.4 Three Levels of Strategy

1. **Corporate level of strategy**. This level answers the foundational question of what you want to achieve.
2. **Business unit level strategy**. This level focuses on how you are going to compete.
3. **Market level strategy**. This strategy focuses on how you are going to grow.

5.7.5 Strategy Intent

- Strategy intent refers to the **purpose** for which the organisation strives for.
- It is philosophical framework of strategic management process. The hierarchy of strategic intent covers the **vision and mission**, business definition and the goals and objectives.

5.7.6 Hierarchical Goals in Strategic Management

- The hierarchy of strategic intent covers the vision and mission, business definition and the goals and objectives.
- **Stretch** is misfit between resources and aspirations. Leverage stretches the meagre resource base to meet the aspirations.
- All three steps in strategic planning occur within three hierarchical levels; upper management, middle management and operational levels. Thus, it is imperative to foster communication and interaction among employees and managers at all levels. So as to help the firm to operate as a more functional and effective team.

5.7.7 Hierarchical Arrangement of Objective

The hierarchy of objectives indicates that **managers at different levels** in the hierarchy of the organisation are concerned with different kinds of objectives according to the authority they are delegated with **Middle level managers** are involved in the setting of key result area objective and division objective.

Hierarchy of goals

The hierarchy of goals is a **goal-oriented iterative method** that can be used to define the scope of the given project. Thus, by setting goals that can be brought up to date as knowledge expands. The method consists of a hierarchy with a main goal on top, followed by sub-goals, project goals, deliveries and success criterion.

Hierarchy of objectives

The hierarchy of objectives is a **tool** that helps analyse and communicate a **project's objectives**. The hierarchy of objective organises the objectives of a project into different levels of hierarchy or tree.

5.7.8 Advantages of Hierarchy of Objectives

1. Clear lines of authority and reporting within the business.
2. Clearer understanding of employee roles and responsibilities.

3. Accountability for actions or decisions at different management levels.
4. Clear career paths and development prospects which can motivate employees.

5.7.9 How Hierarchical Structure Works ?

A hierarchical structure has many layers of management, and business with the structure often use a **'top down'** approach with a long chain of command in a hierarchical structure, managers will have a narrow span of control and a relatively small number of subordinates (staff).

5.7.10 Some Examples of Hierarchy

1. An example of hierarchy is the corporate ladder. An example of hierarchy is the various levels of priests in catholic churches.
2. A structure that has a predetermined ordering from high to low. For example, all files and folders on the hard disk are organised in a hierarchy.

5.7.11 Advantages and Limitations of Hierarchy

Advantage

Clear chain of command, clear paths of Advancement, Specialisation.

Disadvantage

Poor flexibility, communication barriers, organisational disunity.

5.7.12 Characteristics of Hierarchy Data Model

1. **Structure :** The main characteristic of a hierarchical data model is the **tree like** structure. One-to-many and Redundancy, Navigation, local parent pointers.
2. **Hierarchical relation :** Hierarchical relationships are based on degree or levels of superordination and subordination, where the superordinate term represents a class or a whole, and subordinate term refers to its members or parts.

5.7.13 Importance of Hierarchy

- An effective hierarchy makes leaders accountable for results, and provisions for their replacing failures, with someone new-sometimes through internal promotion.
- This is how hierarchy ultimately serves the success of the organisation as whole-including owners, managers and employees.

Hierarchy of concepts

- A conceptual hierarchy defines a sequence of mappings from a set of low-level concepts to higher level managerial concepts.
- These mappings form a concept hierarchy for the dimension location, mapping a set of low-level concepts (i.e. cities) to higher-level, more general concepts (i.e. countries).

5.8 CONDITIONAL PLANNING

- Conditional planning has to work regardless of outcome of an action.
- It takes place in fully **observable environment** where the current state of the agent in known environment is fully observable.
- Outcome of an action cannot be determined so the environment is said to be non-deterministic.

5.8.1 Difference in Hierarchical and Conditional Planning

1. Hierarchical Planning

(a) In hierarchical planning, at each level of hierarchy the objective functions are reduced to a small number of activities at the next lower level.

(b) The computational cost of finding the correct way to arrange these activities for the current problem is small.

(c) Hierarchical methods can result in linear time.

(d) The initial plan of hierarchical planning describes the complete problem which is a very high level description.

(e) The plans are refined by applying action decomposition.

(f) Each action decomposition reduces a high level description to some of the individual lower level descriptions.

Module 5

(g) The action decomposers describe how to implement the actions.

2. Conditional Planning

(a) It deals with the planning by some appropriate conditions.

(b) The agents plan first and then execute the plan that was produced.

(c) The agents find out which part of the plan to execute by including sensing actions in the plan to test for the appropriate conditions.

(d) Conditional planning works regardless of the outcome of an agent.

(e) It takes place in fully observable Environment where the current state of the agent in known environment is fully observable.

(f) The outcome of actions cannot be determined so the environment is said to be non-deterministic.

(g) A "state node" is represented with a "square" and "chance node" is represented with a "circle".

(h) Here, we can check what is happening in environment at predetermined points of the plan to deal with ambiguous action.

(i) It needs to take some actions at every state and must be able to handle every outcome for the action it takes.

Hierarchical methods can result in linear time. The initial plan of hierarchical planning describes the complete problem which is a very high level description.

Conditional planning deals with the planning by some appropriate conditions. The agent plans first and then executes the plan that was produced.

As an alternative to outright refusal the planning authority can grant permission subject to one or more condition. Planning condition sometimes limit the use or occupation of land or premises to a named person or company.

5.9 FORMS OF LEARNING

Learning is one of the fundamental building blocks of Artificial Intelligence solutions. From a conceptual stand point, learning is a process that improves the knowledge of an Artificial Intelligence program by making observations about its environment.

Learning process can be effected as :

(1) **Skill refinement :** Repeatedly practicing one can attain **good skill** refined skill; e.g. learning to drive a car.

(2) **Knowledge acquisition :** Knowledge is acquired through learning. One can develop learning based on type of knowledge representation used (e.g. inductive, predictive logic, concepts, predictions etc.)

5.10 LEARNING ALGORITHMS

With the constant advancements in artificial intelligence, the field has become too big to specialize in all together. There are countless problems that can be solved with countless methods. Knowledge of an experienced AI researcher specialized in one field may mostly be useless for another field. Understanding the nature of different machine learning problems is very important. Even though the list of machine learning problems is very long, these problems can be grouped into three different learning approaches:

1. Supervised Learning;
2. Unsupervised Learning;
3. Reinforcement Learning.

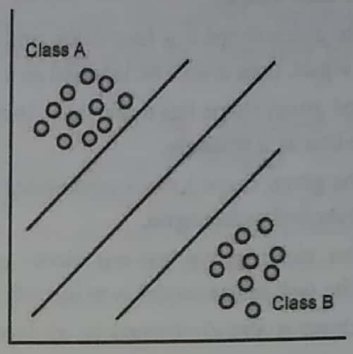
Top machine learning approaches are categorized depending on the nature of their feedback mechanism for learning. Most of the machine learning problems may be addressed by adopting one of these approaches.

5.11 SUPERVISED LEARNING

UQ. What is supervised learning ? Give example of each. **(MU - Q. 1(e), May 17, Q. 1(e), May 19, 5 Marks)**

- Learning that takes place based on a class of examples is referred to as supervised learning. It is learning based on labeled data. In short, while learning, the system has knowledge of a set of labeled data. This is one of the most common and frequently used learning methods
- The supervised learning method is comprised of a series of algorithms that build mathematical models of certain data sets that are capable of containing both inputs and the desired outputs for that particular machine.
- The data being inputted into the supervised learning method is known as training data, and essentially consists of training examples which contain one or more inputs and typically only one desired output. This output is known as a "supervisory signal."
- In the training examples for the supervised learning method, the training example is represented by an array, also known as a vector or a feature vector, and the training data is represented by a matrix.

- The algorithm uses the iterative optimization of an objective function to predict the output that will be associated with new inputs.
- Ideally, if the supervised learning algorithm is working properly, the machine will be able to correctly determine the output for the inputs that were not a part of the training data.
- Supervised learning uses classification and regression techniques to develop predictive models. Classification techniques predict categorical responses,
- Regression techniques predict continuous responses, for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.
- Let us begin by considering the simplest machine-learning task : supervised learning for classification. Let us take an example of classification of documents. In this particular case a learner learns based on the available documents and their classes. This is also referred to as labeled data.
- The program that can map the input documents to appropriate classes is called a classifier, because it assigns a class (i.e., document type) to an object (i.e., a document). The task of supervised learning is to construct a classifier given a set of classified training examples. A typical classification is depicted in Fig. 5.11.1.
- Fig. 5.11.1 represents a hyperplane that has been generated after learning, separating two classes - class A and class B in different parts. Each input point presents input–output instance from sample space. In case of document classification, these points are documents.

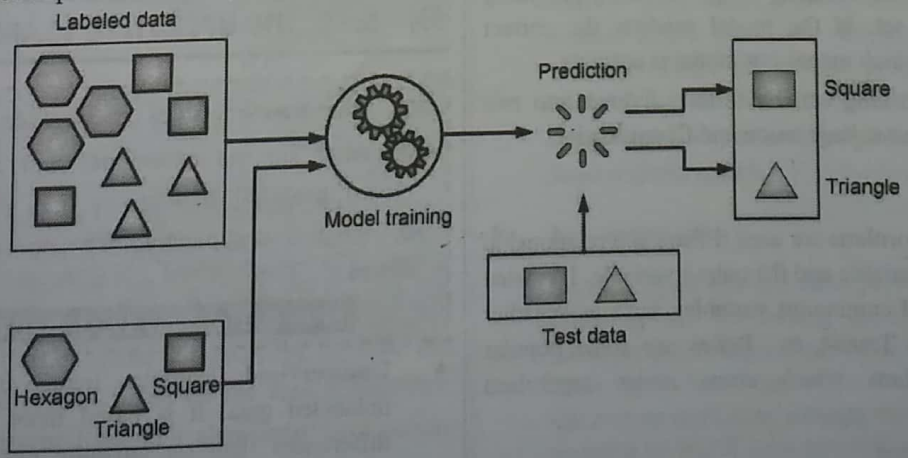


(1D1)**Fig. 5.11.1 : Supervised learning**

- Learning computes a separating line or hyperplane among documents. An unknown document type will be decided by its position with respect to a separator.
- There are a number of challenges in supervised classification such as generalization, selection of right data for learning, and dealing with variations. Labeled examples are used for training in case of supervised learning. The set of labeled examples provided to the learning algorithm is called the *training set*.
- Supervised learning is not just about classification, but it is the overall process that with guidelines maps to the most appropriate decision.

5.11.1 How Supervised Learning Works?

- In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.
- The working of Supervised learning can be easily understood by the below example and diagram (Fig. 5.11.2).



(1D2)**Fig. 5.11.2 : How Supervised learning works?**

Module 5

- Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.
  - If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
  - If the given shape has three sides, then it will be labelled as a **triangle**.
  - If the given shape has six equal sides then it will be labelled as **hexagon**.
- Now, after training, we test our model using the test set, and the task of the model is to identify the shape.
- The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.
- Following are the steps involved in Supervised Learning :
  - First Determine the type of training dataset
  - Collect/Gather the labelled training data.
  - Split the training dataset into training dataset, test dataset, and validation dataset.
  - Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
  - Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
  - Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
  - Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.
- Supervised learning can be further divided into two types of problems: Regression and Classification.

☞ Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning :

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

☞ Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Random Forest
- Logistic Regression
- Decision Trees
- Support vector Machines

5.11.2 Advantages of Supervised Learning

1. With the help of supervised learning, the model can predict the output on the basis of prior experiences.
2. In supervised learning, we can have an exact idea about the classes of objects.
3. Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

5.11.3 Disadvantages of Supervised Learning

1. Supervised learning models are not suitable for handling the complex tasks.
2. Supervised learning cannot predict the correct output if the test data is different from the training dataset.
3. Training required lots of computation times.
4. In supervised learning, we need enough knowledge about the classes of object.

5.12 UNSUPERVISED LEARNING

GQ. What are the types of unsupervised learning?

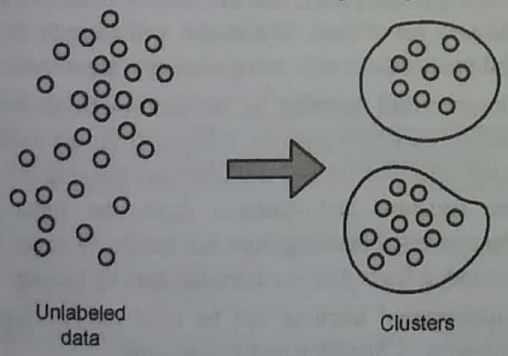
GQ. What are the advantages and disadvantages of unsupervised learning?

UQ. What is unsupervised learning? Give example of each.

(MU - Q. 1(e), May 17, Q. 1(e), May 19, 5 Marks)

- Unsupervised learning refers to learning from unlabeled data. It is based more on similarity and differences than on anything else. In this type of learning, all similar items are clustered together in a particular class where the label of a class is not known.

- It is not possible to learn in a supervised way in the absence of properly labeled data. In these scenarios there is need to learn in an unsupervised way. Here the learning is based more on similarities and differences that are visible. These differences and similarities are mathematically represented in unsupervised learning.
- Given a large collection of objects, we often want to be able to understand these objects and visualize their relationships. For an example based on similarities, a kid can separate birds from other animals. It may use some property or similarity while separating, such as the birds have wings.
- The criterion in initial stages is the most visible aspects of those objects. Linnaeus devoted much of his life to arranging living organisms into a hierarchy of classes, with the goal of arranging similar organisms together at all levels of the hierarchy. Many unsupervised learning algorithms create similar hierarchical arrangements based on similarity-based mappings.
- The task of hierarchical clustering is to arrange a set of objects into a hierarchy such that similar objects are grouped together. Nonhierarchical clustering seeks to partition the data into some number of disjoint clusters. The process of clustering is depicted in Fig. 5.12.1.
- A learner is fed with a set of scattered points, and it generates two clusters with representative centroids after learning. Clusters show that points with similar properties and closeness are grouped together.



(1D3)**Fig. 5.12.1 : Unsupervised learning**

- Unsupervised learning is a set of algorithms where the only information being uploaded is inputs. The device itself, then, is responsible for grouping together and creating ideal outputs based on the data it discovers. Often, unsupervised learning algorithms have certain goals, but they are not controlled in any manner.
- Instead, the developers believe that they have created strong enough inputs to ultimately program the machine to create stronger results than they themselves possibly could. The idea here is that the machine is programmed to run flawlessly to the point where it can be intuitive and inventive in the most effective manner possible.
- The information in the algorithms being run by unsupervised learning methods is not labeled, classified, or categorized by humans. Instead, the unsupervised algorithm rejects responding to feedback in favor of identifying commonalities in the data. It then reacts based on the presence, or absence, of such commonalities in each new piece of data that is being inputted into the machine itself.
- It is used to draw inferences from datasets consisting of input data without labelled responses. Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data. Applications for clustering include gene sequence analysis, market research, and object recognition.

5.12.1 Types of Unsupervised Learning Algorithm

The unsupervised learning algorithm can be further categorized into two types of problems:

1. Clustering
2. Association

1. Clustering

Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

2. Association

An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Module 5

Below is the list of some popular unsupervised learning algorithms :

- K-means clustering
- KNN (k-nearest neighbors)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Apriori algorithm
- Singular value decomposition

5.12.2 Advantages of Unsupervised Learning

1. Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
2. Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

5.12.3 Disadvantages of Unsupervised Learning

1. Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
2. The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

In practical scenarios there is always need to learn from both labeled and unlabeled data. Even while learning in an unsupervised way, there is the need to make the best use of labeled data available. This is referred to as semisupervised learning. Semisupervised learning is making the best use of two paradigms of learning - that is, learning based on similarity and learning based on inputs from a teacher. Semisupervised learning tries to get the best of both the worlds.

5.12.4 Difference between Supervised and Unsupervised Learning

- Supervised and Unsupervised learning are the two techniques of machine learning. But both the techniques are used in different scenarios and with different datasets. Below the explanation of both learning methods along with their difference table is given.
- Supervised learning is a machine learning method in which models are trained using labeled data. In supervised learning, models need to find the mapping function to map the input variable (X) with the output variable (Y).

$$Y = f(X)$$

- Supervised learning needs supervision to train the model, which is similar to as a student learns things in the presence of a teacher. Supervised learning can be used for two types of problems: Classification and Regression.
- **Example :** Suppose we have an image of different types of fruits. The task of our supervised learning model is to identify the fruits and classify them accordingly. So to identify the image in supervised learning, we will give the input data as well as output for that, which means we will train the model by the shape, size, color, and taste of each fruit. Once the training is completed, we will test the model by giving the new set of fruit. The model will identify the fruit and predict the output using a suitable algorithm.
- Unsupervised learning is another machine learning method in which patterns inferred from the unlabeled input data. The goal of unsupervised learning is to find the structure and patterns from the input data. Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own.
- Unsupervised learning can be used for two types of problems : Clustering and Association.
- **Example :** To understand the unsupervised learning, we will use the example given above. So unlike supervised learning, here we will not provide any supervision to the model. We will just provide the input dataset to the model and allow the model to find the patterns from the data. With the help of a suitable algorithm, the model will train itself and divide the fruits into different groups according to the most similar features between them.

GQ. What is the difference between supervised learning and unsupervised learning?

The main differences between Supervised and Unsupervised learning are given below :

Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in Classification and Regression problems.	Unsupervised Learning can be classified in Clustering and Associations problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

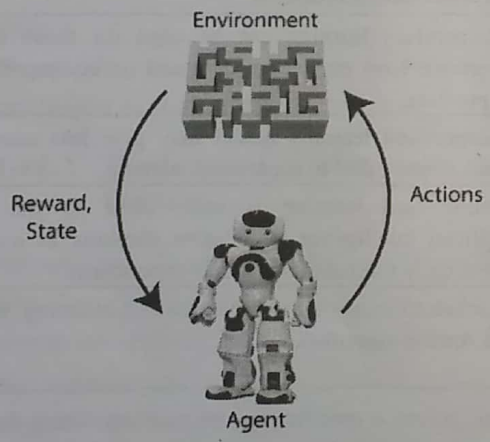
5.13 REINFORCEMENT LEARNING

GQ. What is Reinforcement Learning? Explain with an example.

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that "Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that." How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.

Module 5

- It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.
- **Example :** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.
- The agent continues doing these three things (take action, change state/remain in the same state, and get feedback), and by doing these actions, he learns and explores the environment.
- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.



(1D4) **Fig. 5.13.1**

- For machine learning, the environment is typically represented by an "MDP" or Markov Decision Process. These algorithms do not necessarily assume knowledge, but instead are used when exact models are infeasible. In other words, they are not quite as precise or exact, but they will still serve a strong method in various applications throughout different technology systems.
- The key features of Reinforcement Learning are mentioned below.
  - In RL, the agent is not instructed about the environment and what actions need to be taken.
  - It is based on the hit and trial process.
  - The agent takes the next action and changes states according to the feedback of the previous action.
  - The agent may get a delayed reward.
  - The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

5.13.1 Approaches to Implement Reinforcement Learning

GQ. What are the approaches for Reinforcement Learning?

There are mainly three ways to implement reinforcement-learning in ML, which are :

1. **Value-based :** The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.
2. **Policy-based :** Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward. The policy-based approach has mainly two types of policy :

   Deterministic : The same action is produced by the policy (π) at any state.

   Stochastic : In this policy, probability determines the produced action.
3. **Model-based :** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

Here are important characteristics of reinforcement learning

- There is no supervisor, only a real number or reward signal
- Sequential decision making
- Time plays a crucial role in Reinforcement problems
- Feedback is always delayed, not instantaneous

- Agent's actions determine the subsequent data it receives

RL can be used in almost any application. It is a learning based on experience algorithm, a decision maker algorithm, an algorithm that learns autonomously, an optimization algorithm that over time learns to maximize its reward, the reward can be defined by the engineer to reach the objective of the problem.

5.13.2 Challenges of Reinforcement Learning

Here are the major challenges you will face while doing Reinforcement earning :

1. Feature/reward design which should be very involved
2. Parameters may affect the speed of learning.
3. Realistic environments can have partial observability.
4. Too much Reinforcement may lead to an overload of states which can diminish the results.
5. Realistic environments can be non-stationary.

5.13.3 Applications of Reinforcement Learning

Here are applications of Reinforcement Learning :

1. Robotics for industrial automation.
2. Business strategy planning
3. Machine learning and data processing
4. It helps you to create training systems that provide custom instruction and materials according to the requirement of students.
5. Aircraft control and robot motion control

5.13.4 Reinforcement Learning vs. Supervised Learning

GQ. What is the difference between Reinforcement Learning and Supervised Learning ?

Parameters	Reinforcement Learning	Supervised Learning
Decision style	Reinforcement learning helps you to take your decisions sequentially.	In this method, a decision is made on the input given at the beginning.
Works on	Works on interacting with the environment.	Works on examples or given sample data.
Dependency on decision	In RL method learning decision is dependent. Therefore, you should give labels to all the dependent decisions.	Supervised learning the decisions which are independent of each other, so labels are given for every decision.
Best suited	Supports and work better in AI, where human interaction is prevalent.	It is mostly operated with an interactive software system or applications.
Example	Chess game	Object recognition

5.14 GENERAL LEARNING MODEL

We develop general learning model and note the factors affecting learning performance.

(i) Learning can be done in several ways. But learning requires that new knowledge structure be created from some form of input stimulus.

This type of new knowledge must be assimilated into a knowledge base and be tested in some way for its utility.

(ii) Testing means that the knowledge should be used in the performance of some task from which meaningful feedback can be obtained. The feedback provides some measure of accuracy and usefulness of the newly acquired knowledge. We exhibit the general learning model in the figure.

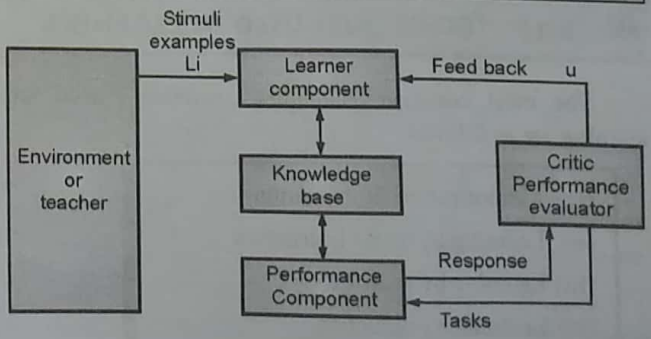


Fig. 5.14.1 : General Learning Model

(iii) In the figure, environment represents overall learner system. The environment may be regarded as one which produces random stimuli or a training source such as a teacher which provides carefully selected training examples to the learner component.

Module 5

(iv) Some representation language for communication between environment and the learner component must be used. The language scheme must be same as that used in the knowledge base. When they are same, we say that 'single representation' is used.

(v) Inputs to the learner component may be physical stimuli of some type or descriptive, symbolic training examples.

The information conveyed to the learner component is used to create and modify knowledge structures in the knowledge base.

The performance component uses this knowledge to carry out some tasks, such as solving a problem, playing a game etc.

(vi) When a task is given to performance component, it describes its actions in performing the task and produces a response. Then critic module evaluates this response **relative** to an **optimal** response.

(vii) Critic module sends the response to the learner component to check whether or not the performance is acceptably and if so required modifying the structures in the knowledge base.

If the proper learning is achieved, then the performance of the system will be improved by the knowledge base component.

(viii) The cycle described in the above figure is to be repeated a number of times (i) until the performance of the system reaches some acceptable level, (ii) or a known learning goal is achieved, or (iii) after some repetitions no change **occurs** in the knowledge base.

5.15 TECHNIQUES USED IN LEARNING

The most common techniques (methods) used for learning are as follows :

(i) Memorization (Rote learning)
(ii) Learning by direct instruction
(iii) Learning by analogy,
(iv) Learning by induction
(v) Learning by deduction
(vi) Learning using neural network

5.15.1 Learning by Memorization (Rote Learning)

- It is the simplest form of learning. It requires the least amount of inference. Here, learning is achieved by simply copying the knowledge that is used in the knowledge base. For example, for memorizing multiplication table, we use this type of learning.
- When a computer stores a piece of data, it performs a rudimentary form of learning. It is a simple case of 'data caching'. We store computed values so that we do not have to recompute them later.
- When computation is more expensive then this strategy can save a significant amount of time.
- Caching is used in AI programs to produce some suprising performance improvements. Such caching is known as rote learning.

Remark : Data Caching

- Data Caching is a technique of storing frequently used data in memory, so that when the same data is needed next time, it can be directly obtained from the memory, instead of being generated by the application.
- The Android Phone's cache comprises stores of small bits of information that your apps and web browser use to speed up performance. But cached files can become corrupted and overloaded and cause performance issues. Cache need not be constantly cleared, but a periodic clean-out can be helpful.

5.15.2 Learning by Direct Instruction

- This type of learning is slightly more complex. This requires more inference than rote learning. Here to integrate knowledge into 'knowledge base', it must be transformed into an operational form.
- When a number of facts are presented to us directly in a well-organised manner, we use this type of learning.

5.15.3 Learning by Analogy

- This is a process of learning a new concept or solution by using the similar known concepts or solutions. For example, in an examination previously learned examples help one to solve new problems (in an exam)
- We make frequent use of analogical learning. This form of learning requires more inferring than either of the previous forms.
- It is because 'difficult transformations' must be made between the known and unknown situations.

5.15.4 Learning by Induction

- This is a powerful form of teaming which also requires more interring than the first two methods. This form of learning is a form of **invalid** but **useful** inference.
- Here we formulate a general concept after seeing a number of instances or examples of the concept. For example, we learn the concepts of sweet taste or color after experiencing the sensations associated with several examples of sweet foods or coloured objects.

5.15.5 Learning by Deduction

- This form of Learning is achieved through a sequence of deductive inference steps using known facts. From the known facts, new facts or relationships are derived logically. For example, we could learn deductively that sita is the cousin of Ramesh, if we have knowledge of sita and Ramesh's parents and Rules for cousin relationship.
- Deductive learning requires more interence than the other methods

5.15.6 Neural Network

- Neural Networks can be loosely separated into neural methods and learning rules.
- Mculloch and pitts developed the first network models to explain how the signals passed from one neuron to another within **the network.**

5.16 STATISTICAL LEARNING IN ARTIFICIAL INTELLIGENCE

- Statistical learning theory is a branch of Artificial Intelligence. It provides us with the theoretical basis for many of today's machine learning algorithms. The basic premise of learning is to use a set of observations to uncover an underlying process. Thus statistical learning in AI is a set of tools for machine learning that uses statistics and functional analysis.
- The main goal of statistical learning theory is to provide a framework for studying the problem if inference, that is of gaining knowledge, making predictions, making decisions or constructing models from a set of data.

5.17 THEORY OF LEARNING

The theory of AI learning includes the study of

(i) Neural like elements,
(ii) Multi-dimensional neural –like growing networks
(iii) Temporary and long term memory
(iv) The study of functional organization of the brain of AI systems of the sensor systems, modulating system, motor system etc.

5.17.1 Machine-Learning Theory

- Computational learning theory or statistical learning theory refers to mathematical frameworks for qualifying learning tasks and algorithms.
- These are the subfields of machine learning that a machine-learning practitioner does not **needs to know** in great depth in order to achieve good results on wide range of problems.

5.17.2 Artificial Learning and Machine Learning

- Learning in machine learning refers to a machine's ability to learn and is based on data; and machine-learning algorithms ability to train a model, evaluate its performance or accuracy and then make predictions.
- AI is a technology that enables a machine to simulate human behaviour. Machine-learning is a subset of Artificial Intelligence, which allows a machine to learn automatically from past data without programming explicity. The goal of AI is to make a smart compute system like human to solve complex problems.

5.17.3 A.I and Deep Learning

- A.I means getting a computer to copy human behaviour in some ways. Deep learning is a subset of machine learning that enables computers to solve more complex problems.
- Deep Learning uses huge neural networks with many layers of processing units. It takes advantage of many improved training techniques in computing power to learn complex patterns in large amounts of data. Common applications include image and speech recognition.

Module 5

5.17.4 Techniques of Deep Learning

- Deep learning theory is a subfield of AI and devoted to studying the design and analysis of machine algorithm.
- In statistical learning, algorithm is given samples that are labelled in some useful way. For example, the samples might be description of mushrooms and labels could be whether or not the mushrooms are edible.
- The algorithm takes these previously labelled samples and uses them to induce a classifier. This classifier is a function that assigns labels to samples, that have not been **seen** previously by the algorithm.
- The goal of statistical learning algorithm is to optimise some measure of performance such as minimising the number of mistakes made on new samples.
- In addition to performance bounds, computational learning theory studies the time complexity and feasibility of learning.
- In computational learning theory a computation is considered feasible if it can be done in polynomial time.

  There are two types of complexity results :

  (1) **Positive results :** It shows that a certain class of functions is learnable in polynomial time.

  (2) **Negative results :** It shows that certain classes cannot be learned in polynomial time.

Remark : Polynomial time

An algorithm is said to be of polynomial time if its running time is upper bound by a polynomial expression in the size of the input algorithm, that is

$T(n) = 0 (n^k)$; for some positive constant K

Cobhom's thesis states that polynomial time is a synonym for 'tractable', 'feasible' 'efficient' or 'fast'.

5.18 PAC LEARNING

- In computational learning theory, 'Probably Approximately Correct' (PAC) learning is a framework for mathematical analysis of machine learning. PAC analyses the generalization error of a learning algorithm in terms of its error on a training set. The goal is to show that an algorithm achieves low generalization error with high probability.
- Probably approximately correct (PAC) learning theory helps analyse whether and under what conditions a learner L will probably output an approximately correct classifier.

5.18.1 Analysis of PAC

PAC analysis is used to compute transfer functions for circuits that exhibit frequency translation. It is a small signal analysis lie AC analysis, except that the circuits is first linearised about a periodically varying operating point as oppoced to a simple DC operating point.

5.18.2 Epsilon in PAC Learning

- Probability [error (h) > $\in$] < δ ; we are now in a position to say when a learned concept is good , [i.e. degree of goodness]
- 'When the [probability that its error is greater than accuracy $\in$] is less than the confidence level δ'
- Different degrees of 'goodness' will correspond to different values of epsilon and delta. The smaller epsilon and delta are, the better the learned concept will be.
- Our problem, for a given concept to be learned, and given epsilon and delta, is to determine the size of the training set. This may or may not depend on the algorithm used to derive the learned concept.
- Now, we want to learn the concept "medium-built person" from examples. We are given the height and weight of 'm' individuals, the training set. We are told for each [height, weight] pair if it is or not of medium built.
- We want to study this concept. We produce an algorithm that answers correctly if a pair [height, weight] represents/not a medium built person. We want to find the value of m to use if we want to learn the concepts.
- First to understand the concept; the probability of error of the learned concept is at most epsilon.
- Suppose C is the (true) concept we are learning, h is the concept we have learned, then

  error [h] = probability [C(x) < > h(x) for an individual x] < epsilon

Now, we set a bound delta on the probability that this error is greater than epsilon, that is,

Probability [error (h) > epsilon] < delta

Now, we assume that the concept is represented as a rectangle, with sides parallel to the axes height/weight, and with dimensions height-min, height-max; weight-min, weight-max.

Now the hypothesis will take the form of a rectangle with the sides parallel to the axes. To build the learned concept, we build the algorithm.

(i) If there are no positive individuals in the training set, the learned concept is null.

(ii) We choose epsilon and delta and determine a value for m that will satisfy the PAC learning conditions.

5.18.3 Agnostic PAC Learning

- A learner that does not assume that it contains an error-free hypothesis and that simply finds the hypothesis with minimum training error is called as an agnostic learner.
- An important innovation of PAC frame work is the introduction of computational complexity theory concepts to machine learning.
- In particular the learner is expected to find efficient functions (time and space requirement bounded to a polynomials of the example size) , and the learner itself must implement an efficient procedure.

5.19 REINFORCEMENT LEARNING (R.L)

- We have already seen the meaning of Reinforcement learning, in nutshell. Now we discuss R.L. in detail.
- Reinforcement learning is an area of machine learning. In R.L., reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it has to learn from its experience.

5.19.1 Example : We Consider a Problem

- We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best solution path to reach the reward.
- The following range shows the robot, diamond and fire. The goal of the robot is to get the reward that is the diamond and avoid the hurdles that are fire. The robot learns by trying all the possible paths and then choosing the path which gives him the reward with the least hurdles. Each right step will subtract the reward of the robot. The total reward will be calculated when it reaches the final reward that is the diamond.

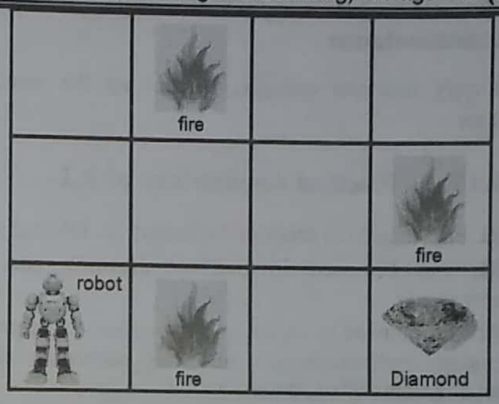


Fig. 5.19.1

5.19.2 Method of R.L.

(i) **Input :** The input should be an initial state from which the model will start.

(ii) **Output :** There are many possible output as there are varieties of solution to a given problem.

(iii) **Training :** The training is based upon the input. The model will return a state and the user will decide to reward or punish the model based on its output.

(iv) The model keeps on learning

(v) The best solution is decided based on the maximum reward.

5.19.3 Types of Reinforcement

There are two types of Reinforcement

(1) **Positive :** Positive reinforcement is defined as when an event occurs due to a particular behaviour, increases the strength and the frequency of the behaviour. In other words, it has a positive effect on behaviour.

Advantages of RL

(i) Maximise performance

(ii) Sustain change for a long period of time

Disadvantages of RL

Too much Reinforcement can lead to overload of states which can diminish the results.

(2) **Negative :** Negative reinforcement is defined as strengthening of a behaviour because a negative condition is stopped or avoided.

Advantages

(i) Increases behaviour

(ii) Provide defiance to minimum standard of performance

Module 5

Disadvantages

It only provides enough to meet up the minimum behaviour.

5.19.4 Practical Applications of R.L

(1) RL can be used in robotics for industrial automation.

(2) RL can be used in machine learning and data processing.

(3) RL can be used to create training systems that provide custom instruction and materials according to the requirements of students.

RL can be used in large environment in the following situations :

(i) A model of the environments is known, but an analytic solution is not available.

(ii) Only a simulation model of the environment is given (the subject of simulation based optimisation)

(iii) The only way to collect information about the environment is to interact with it.

(4) Training the models that control autonomous cars is an excellent example of a potential application of reinforcement learning.

- In an ideal situation, the computer should get no instructions on driving the car. The programmer would avoid hard-wiring anything connected with the task and allow the machine to learn the from its own errors. In a perfect situation, the only hard-wired element would be the reward function.
- **For example** in usual circumstances we would require an autonomous vehicle to put safety first, minimize ride time, reduce pollution offer passengers comfort and obey the rules of law.
- With an autonomous race car, we would emphasis, speed much more than the driver's comfort. The programmer cannot predict everything that could happen on the road. Instead of building lengthy "if-then" instructions, the programmer prepares the reinforcement learning agent to be capable of learning from the system of rewards and penalties. The agent, it is the another name for reinforcement learning algorithms performing the task gets rewards for reaching specific goals.
- In particular, if AI is going to drive a car, learning to play some Atari classics can be considered a meaningful intermediate milestone. A potential application of reinforcement learning in autonomous vehicles is the following interesting case :
- A developer is unable to predict all future road situations, so letting the model train itself with a system of penalties and rewards in a varied environment is possibly the most effective way for the AI to broaden the experience it both has and collects.

5.19.5 Challenges with RL

- The main challenges in RL is in preparing the simulation environment, which is highly dependent on the task to be performed. When the model has to go superhuman in chess, Go or Atari games, preparing the simulation environment is relatively simple. When it comes to building a model capable of driving on autonomous car, building a realistic simulator is crucial before letting the car ride on the street.
- The model has to figure out how to brake or avoid a collision in a safe environment, where sacrificing even a thousand cars comes at a minimum cost. Transferring the model out of the training environment and into the real world is where things get tricky. Scaling the neural network controlling the agent is another challenge. There is no way to communicate with the network other than through the system of rewards and penalties. This in particular may lead to catastrophic forgetting, where acquiring new knowledge causes some of the old to be erased from the network.
- Yet another challenge is reaching a local optimum that is the agent performs the task as it is; but not in the optimal or required way.

5.19.6 Challenges in Deep Reinforcement Learning Research and Applications

The existing challenges in deep reinforcement learning research and applications, including :

(1) The sample efficiency problem

(2) Stability of training

(3) The catastrophic interference problem

(4) The exploration problem

(5) Meta-learning and representation learning for the generality of reinforcement learning methods across tasks;

(6) Multi-agent reinforcement learning with other agents an part of the environment

(7) sim-to-real transfer for bridging the gaps between simulated environments and the real world.

(8) Large-scale reinforcement learning with parallel training frameworks to shorten the wall-clock time for training etc.

Some key words to be noted

- Sample efficiency – stability
- Catastrophic interference - Exploration
- Meta learning – Representation learning Generality
- Multi agent reinforcement learning
- Sim2real-scalability

5.20 LEARNING FROM REWARDS AND PUNISHMENT

- The concept of Rewards and punishment was developed by D.F. skinner.
- Operant conditioning is a way of learning by means of rewards and punishments. This type of conditioning holds that a certain behaviour and a consequence, either a reward or punishment have a connection which brings about learning.
- Rewards and punishment motivate behaviour, but it is not clear how they impact skill performance and whether the effect varies across skills. Collectively these results suggest that punishment impacts skilled behaviour more than reward in a complex, task dependent fashion.
- If we are doing something that is rewarding, we continue to do it; and if it is punishing, we stop doing it. Therefore the reward and punishment centrers undoubtedly constitute one of the most important of all the controllers of our bodily activities, our drives, our aversions, our motivations.

5.20.1 Effectiveness of Rewards and Punishments

- Neuroscience suggests that it comes to motivating action (e.g. getting people to work longer hours or producing star reports etc.) rewards may be more effective than punishment.
- So our brain has evolved to accomodate our environment in which often the best way to gain rewards, is to take actions.
- The agent is rewarded for correct moves and punished for the wrong ones. In doing so, the agent tries to minimize wrong moves and maximize the right ones.

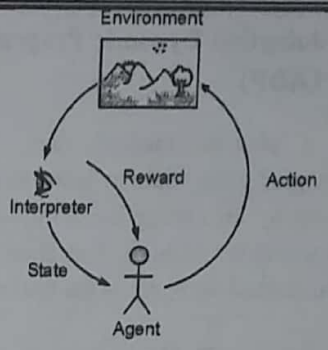


Fig.5.20.1

5.21 PASSIVE REINFORCEMENT LEARNING AND ACTIVE REINFORCEMENT LEARNING

- In case of passive reinforcement learning, the agent's policy is fixed which means that it is told what to do. Hence the goal of passive RL agent is to execute a fixed policy (i.e. sequence of actions) and evaluate it while that of an active RL agent is to act and learn an optimal policy.
- In contrast to this, in active RL an agent needs to decide what to do an there is no fixed policy that it can act on.

5.22 PASSIVE REINFORCEMENT LEARNING

As the goal of the agent is to evaluate how good an optimal policy is the agent needs to learn the expected utility u π(s) for each state s. This can be done in 3 ways :

5.22.1 Direct Utility Estimation

- In this method, the agent executes a sequence of trials or runs (sequence of states action transitions that continue until the agent reaches the terminal state). Each trial gives a sample value and the agent estimates the utility based on the sample value. And it can be calculated as running averages of sample values.
- The main drawback is that this method makes a wrong assumption that state utilities are independent while in reality they are Markovian. Also it is slow to converge.

Module 5

5.22.2 Adaptive Dynamic Programming (ADP)

- ADP is a smarter method than Direct Utility Estimation as it runs trials to learn the model of the environment by estimating the utility of a state as a sum of rewards for being in that state. And expected discounted reward for being in the next state :

$$U^{\pi}(s) = R(s) + r \sum_{s'} P(s' \mid s, \pi(s))\, U^{\pi}(s')$$

Where R(s) = reward for being instate s
$P(S'/S, \pi(s))$ = transition model,
ν = discounted factor
And $U^{\pi}(s)$ = utility of being in state S

- It can be solved using value-iteration algorithm. The algorithm converges fast but can become quite costly to compute for large state spaces.
- ADP is a model based approach and requires the transition model of the environment.
- A model free approach is temporal difference learning.

5.22.3 Temporal Difference Learning (TDL)

- TDL does not require the agent to learn the transition model. The updates occur between successive states and agents only updates states that are directly affected

Here , $u^{\pi}(s) \leftarrow u^{\pi}(s') + \alpha\,[R(s) + \nu\,(u^{\pi}(s') - u^{\pi}(s))]$

Where,

α = learning rate which determines the convergence to true utilization

- While ADP adjusts the utility of s with all its successor states, TD learning adjusts it with that of a single successor state s′. TDL is slower in convergence of computation.

5.22.4 Active Reinforcement Learning

- As the goal of an active agent is to learn an optimal policy, the agent needs to learn the expected utility of each state and update its policy.
- It can be done using a passive ADP agent and then using value or policy iteration, it can learn optimal actions. Bu this approach results into a greedy agent.
- Hence, we use an approach that gives higher weights to unexplored actions and lower weights to actions with lower utilities.

$$U_{i+1}(s) \leftarrow R(s) + \nu \max_{a \in A} f\left(\sum_{s'} P(s'/s,a)\, u_i(s'), N(s,a)\right)$$

Where f(u,n) is the exploration function that increases with expected value u and decreases with number of trails n.

$$f(u, n) = \begin{cases} R^{+}, & \text{if } n < Ne \\ u, & \text{otherwise} \end{cases}$$

where R^{+} is an optimistic reward and N_e is the number of times we want an agent to be forced to pick an action in every state. The exploration function converts a passive agent into active one.

While ADP adjusts the utility of s with all its successive states, TD learning adjusts it with that of a single successor state S′.

TD is slower in convergence of computation.

5.23 Q-LEARNING

- Q-learning is a model free reinforced learning algorithm to learn the value of an action in a particular state.
- It does not require a model of the environment (hence 'model-free') and it can handle problems with stochastic transitions and rewards without requiring adaptations.
- For any finite Markov decision process (FMDP), Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state.
- Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly-random policy. 'Q' refers to the function that the algorithm computes – the expected rewards for an action taken in a given state.

5.23.1 Goal of the Agent

- R.L. involves an agent, a set of states S and a set A of actions per state. By performing an action $a \in A$, the agent transitions from state to state. Executing an action in a specific state provides the agent with a reward (a numerical score)
- The goal of the agent is to maximise its total reward. It does this by adding the maximum reward attainable from future states to the reward for achieving its current state, effectively influencing the current action by the potential future reward. This potential reward is

a weighted sum of the expected values of the rewards of all future steps starting from the current state.

- As an example, consider the process of boarding a train in which the reward is measured by the negative of the total time spent boarding (alternatively, the cost of boarding the train is equal to the boarding time)
- One strategy is to enter the train door as soon as they open, minimising the initial wait time for yourself. If the train is crowded then you will have a slow entry after the initial action of entering the door as people are fighting you to depart the train as you attempt to board. The total boarding time, or cost, is then 'O seconds wait time +15 seconds fight time'.
- On the next day, by random chance (exploration), you decide to wait and let other people depart first. This results in a larger wait time initially. But the time fighting other people is less. Overall, this path has a higher reward than that of the previous day, since the total boarding time is now.

'5 seconds wait time +0 seconds fight time'

- Through exploration, despite the initial action resulting in a larger cost (or negative reward) than in the forceful strategy, the overall cost is lower, and it reveals a more rewarding strategy.

5.23.2 Influence of Variables : Learning Rate

- The learning rate or step size determines to what extent acquired information overrides old information.
- A factor of O makes the agent learn nothing (exclusively exploiting prior knowledge) while a factor 1 makes the agent consider only the most recent information (ignoring prior knowledge to explore possibilities). In fully deterministic environments, a learning rate of $\alpha_t = 1$ is optimal. When the problem is stochastic, the algorithm converges under some technical conditions on the learning rate that requires it to decrease to zero. In practice, often a constant learning rate is used, such as $\alpha_t = 0.1$ for all t.

5.23.3 Discount Factor

- The discount factor ν determines the importance of future rewards. A factor of O will make the agent short-sighted (myopic) by only considering current rewards, i.e. T_1 while a factor approaching 1 will make it strive for a long term high reward. If the discount factor meets or exceeds 1, the action values may diverge, for $\gamma = 1$, without a terminal state, all environment histories become infinitely long and utilities with additive, undiscounted rewards generally become infinite.
- Even with a discount factor only slightly lower than 1, Q-function learning leads to propagation of errors and instabilities when the value function is approximated with an artificial neural network.
- In that case, starting with a lower discount factor and increasing it towards its final value accelerates learning.

5.23.4 Initial Conditions (Q_0)

- Since Q-learning is an iterative algorithm, it assumes an initial condition before the first update occurs. High initial conditions before the first update occurs. High initial values, also known as **'optimistic initial conditions'** can encourage exploration.
- No matter what action is selected, the update rule will cause it to have lower values than the other alternative and that increases the choice of probability.
- The first reward γ can be used to reset the initial conditions. According to this idea, the first time an action is taken the reward is used to set the value of Q. This allows immediate learning in case of fixed deterministic rewards. A model that incorporates reset of initial conditions (RIC) is expected to predict participants behaviour better than a model that assumes any arbitrary initial condition (AIC). RIC seems to be consistent with human behaviour in repeated binary choice experiment.

5.23.5 Implementation

- Q-learning can be combined with function approximation. This makes it possible to apply the algorithm to larger problems, even when the state-space is continuous.
- One solution is to use an artificial neural network as a function approximator. Function approximation may speed up learning in finite problems, due to the fact that the algorithm can generalise earlier experiences to previously unseen states.

Module 5

5.23.6 Quantization

- Another technique to decrease the state/action space quantizes possible values.
- Consider an example of learning to balance a stick on a finger. To describe a state at a certain point in time involves the position of the finger in space, its velocity, the angle of the stick, and the angular velocity of the stick. This gives a four element vector that describes **one state,** i.e. a snapshot of one state encoded into four values.
- The problem is that infinitely many possible states are present.
- The exact distance of the finger from its starting position ($-\infty$ to ∞) is not known. But rather it is far away or not can be located

5.23.7 Variants (Deep Q-learning)

- When a non-linear function approximator such as a neural network is used to represent Q, then reinforced learning becomes unstable or divergent. The reason for this instability is due to the correlation present in the sequence of observations.
- The small updates to Q may change the policy of the agent and the data distribution, and the correlations between Q and the target values.
- Here the technique uses a random sample of prior actions instead of the most recent action to proceed. This removes correlation in the observation sequence and smooths changes in the data distribution.
- Iterative updates adjust Q towards target values that are only periodically updated further reducing correlations with the target.

5.23.8 Delayed Q-learning

- Delayed Q-learning is an alternative implementation of the online Q-learning algorithm, with probably approximately correct (PAC) learning.
- (Greedy) GQ is a variant of Q-learning to use in combination with (linear) function approximation. The advantages of Greedy GQ is that convergence is guaranteed even when function approximation is used to estimate the action values.
- Distributed Q-learning is a variant of Q-learning which seeks to model the distribution of returns rather than the expected return of each action. It has been observed to facilitate estimate by deep neural networks and can enable alternative control methods, such as risk sensitive control.

5.23.9 Limitations

- The standard Q-learning algorithm (using Q-table) applies only to discrete action and state spaces. Discretisation of these values leads to inefficient learning, largely due to the dimensionality.
- However, there are adaptations of Q-learning that attempts to solve this problem such as wire-fitted Neural Q-learning.

5.23.10 Temporal Difference Learning (TD)

- TD learning refers to a class of model-free reinforcement learning methods which learn by bootstrapping from the current estimate of the value function. These methods sample from the environment, like Monte Carlo methods, and perform updates based on current estimates, like dynamic programming methods.
- While Monte-Carlo methods only adjust their estimates once the final outcome is known, TD methods adjust predictions to match later, more accurate, predictions about the future before the final outcomes is known. This is a form of bootstrapping. We illustrate this with the following example :

"Suppose you wish to predict the weather for Saturday, and you have some model that predicts Saturday's weather given the weather of each day in the week. In the standard case, you would wait until Saturday and then adjust all your models. However, when it is, for example, Friday, you should have a pretty good idea of what the weather would be on Saturday- and thus be able to change, say, Saturdays model before Saturday arrives. "

5.23.11 TD Algorithm in Neuroscience

- Researchers have discovered that the firing rate of dopamine neurorns in the ventral tegmental area (VTA) and substantia nigra (SIVc) appear to reduce the error function in the algorithm. The error function reports back the difference between the estimated reward at any given state or time step and the actual reward received.

- The larger the error function, the larger the difference between the expected and actual reward. When this is paired with a stimulus that accurately reflects a future reward, the error can be used to associate the stimulus with the future reward.
- The relationship between the model and potential neurological function has produced research attempting to use TD to explain many aspects of behavioural research.

☞ **Remark**

(1) The 'q' in q-learning stands for quality. Quality in this case represents how useful a given action is in gaining some future reward

(2) Gamma in Q-learning is the discount factor. It quantifies how much importance we give for future rewards. It is also handy to approximate the noise in future rewards Gamma varies from 0 to 1. If Gamma is closer to 0, the agent will tend to consider only immediate rewards.

5.24 INTRODUCTION TO STATISTICAL LEARNING

- In statistical learning a set of rules is used to understand the given data.
- Statistical learning theory is a framework for machine learning. It is developed using statistics and functional analysis.
- Statistical learning theory finds predictive function using given data to solve the problem.
- Statistical learning is understanding from training data and predicting on unseen data.
- Statistical learning is estimating a function f from the given data. Using f_c we can better grasp the relation between the dependent variable Y on X. Thus f gives a the systematic information that X provides about Y. Generally X is a vector value quantity of the types $X = (X_1, X_2, \ldots, X_m)$
- Statistical learning theory has practical applications in the fields of computer vision, speech recognition and bioinformatics.
- Statistical learning theory begins with a class of hypothesis and uses empirical data to select one hypothesis from that class. If the data generating mechanism is soft, then the difference between the training error of a hypothesis from the class is small (and hence may be negligible).
- Statistical learning theory comes under different classes : supervised learning and unsupervised learning on line learning and reinforcement learning.
- Supervised learning is predicting an output based on one or more inputs. Supervised learning involves learning from a training set of data. Every point in the training set is an input-output pair, where input maps the output. The learning method infers the function that maps between the input and the output and the learned function is used to predict the output from future input.

☞ **Types of supervised learning problems**

- Supervised learning problems are problems of regression or problems of classification. And that depends on the type of output.If the output takes continuous values, it is a regression problem. For example, consider Ohm's law. If voltage is taken as input and current as output then regression will be given by the functional relation V = I-R.
- If the output takes discrete set of values then it is a classification problem. Classification is very common for machine learning applications.In facial recognition, we regard a person's face as input and person's name as output. Here the input is a large multidimensional vector whose elements are pixels in the picture.

Module 5

Chapter Ends...

❑❑❑