Unit 4

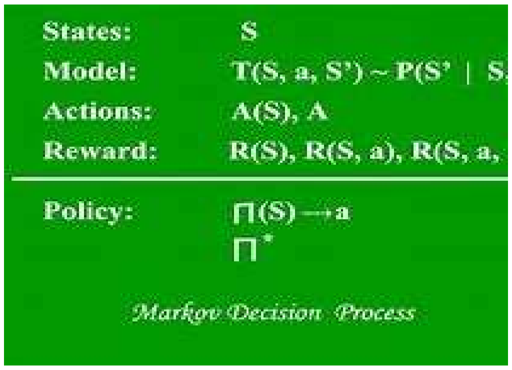
Markov Decision Process

Reinforcement Learning is a type of Machine Learning. It allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behavior; this is known as the reinforcement signal.

There are many different algorithms that tackle this issue. As a matter of fact, Reinforcement Learning is defined by a specific type of problem and all its solutions are classed as Reinforcement Learning algorithms. In the problem, an agent is supposed to decide the best action to select based on his current state. When this step is repeated, the problem is known as a **Markov Decision Process**.

A **Markov Decision Process (MDP)** model contains:

- A set of possible world states S.
- A set of Models.
- A set of possible actions A.
- A real-valued reward function R(s,a).
- A policy is a solution to **Markov Decision Process**.



States:	S
Model:	T(S, a, S') ~ P(S' | S
Actions:	A(S), A
Reward:	R(S), R(S, a), R(S, a,
Policy:	$\Pi(S) \rightarrow a$ Π^*

Markov Decision Process

<u>State</u>

A **State** is a set of tokens that represent every state that the agent can be in.

<u>Model</u>

A **Model** (sometimes called Transition Model) gives an action's effect in a state. In particular, T(S, a, S') defines a transition T where being in state S and taking an action 'a' takes us to state S' (S and S' may be the same). For stochastic actions (noisy, non-deterministic) we also define a probability P(S'|S,a) which represents the probability of reaching a state S' if action 'a' is taken in state S. Note Markov property states that the effects of an action taken in a state depend only on that state and not on the prior history.

Actions

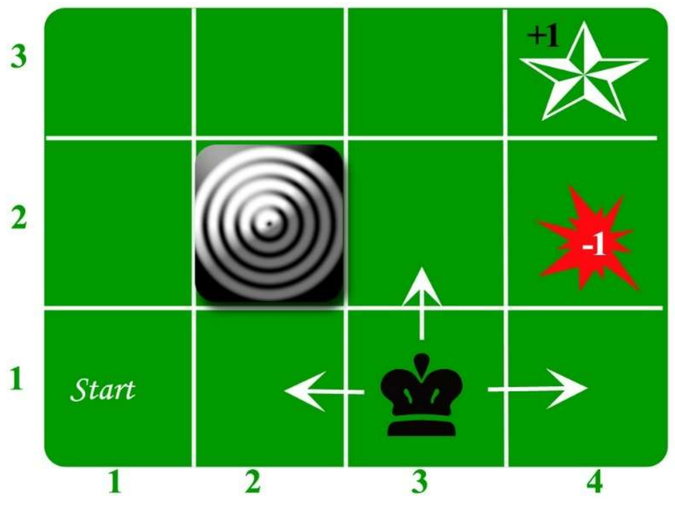
Action A is a set of all possible actions. A(s) defines the set of actions that can be taken being in state S.

Reward

A **Reward** is a real-valued reward function. R(s) indicates the reward for simply being in the state S. R(S,a) indicates the reward for being in a state S and taking an action 'a'. R(S,a,S') indicates the reward for being in a state S, taking an action 'a' and ending up in a state S'.

Policy

A **Policy** is a solution to the Markov Decision Process. A policy is a mapping from S to a. It indicates the action 'a' to be taken while in state S.
Let us take the example of a grid world:



An agent lives in the grid. The above example is a 3*4 grid. The grid has a START state(grid no 1,1). The purpose of the agent is to wander around the grid to finally reach the Blue Diamond (grid no 4,3). Under all circumstances, the agent should avoid the Fire grid (orange color, grid no 4,2). Also, the grid no 2,2 is a blocked grid, it acts as a wall hence the agent cannot enter it.

The agent can take any one of these actions: **UP, DOWN, LEFT, RIGHT**

Walls block the agent's path, i.e., if there is a wall in the direction the agent would have taken, the agent stays in the same place. So for example, if the agent says LEFT in the START grid he would stay put in the START grid.

First Aim: To find the shortest sequence getting from START to the Diamond. Two such sequences can be found:

- **RIGHT RIGHT UP UPRIGHT**
- **UP UP RIGHT RIGHT RIGHT**

Let us take the second one (UP UP RIGHT RIGHT RIGHT) for the subsequent discussion.
The move is now noisy. 80% of the time the intended action works correctly. 20% of the time the action agent takes causes it to move at right angles. For example, if the agent says UP the probability of going UP is 0.8 whereas the probability of going LEFT is 0.1, and the probability of going RIGHT is 0.1 (since LEFT and RIGHT are right angles to UP).

Utility theory

Utility theory is a fundamental concept in economics and decision theory. This theory provides a framework for understanding how individuals make choices under uncertainty. The aim of this agent is not only to achieve the goal but the best possible way to reach the goal. This idea suggests that people give a value to each possible result of a choice showing how much they like or are happy with that result. The aim is to get the highest expected value, which is the average of the values of all possible results taking into account how likely each one is to happen.

1. Utility Function: Definition and Purpose

The utility function is a core element of utility-based agents, serving as a mathematical representation of the agent's preferences. It assigns a numerical value (utility) to each possible outcome, reflecting the desirability or satisfaction associated with that outcome.

- **Definition**: A utility function U(s) is a mapping from states sss to real numbers, indicating the utility or value of each state.
- **Purpose**: The primary purpose of the utility function is to quantify the agent's preferences, allowing it to compare and evaluate different states. By maximizing utility, the agent can choose actions that lead to the most desirable outcomes according to its objectives.

For example, in an autonomous vehicle, the utility function might consider factors such as safety, speed, fuel efficiency, and passenger comfort. Each possible driving state would be assigned a utility value based on these criteria.

Difference between Value Iteration and Policy Iteration

Aspect	**Value Iteration**	**Policy Iteration**
Methodology	Iteratively updates value functions until convergence	Alternates between policy evaluation and improvement
Goal	Converges to optimal value function	Converges to the optimal policy
Execution	Directly computes value functions	Evaluate and improve policies sequentially
Complexity	Typically simpler to implement and understand	Involves more steps and computations
Convergence	May converge faster in some scenarios	Generally converges slower but yields better policies

Partially Observable Markov Decision Process (POMDP)

A POMDP models decision-making tasks where an agent must make decisions based on incomplete or uncertain state information. It is particularly useful in scenarios where the agent cannot directly observe the underlying state of the system but rather receives observations that provide partial information about the state.

Components of a POMDP

A POMDP is formally defined by the following elements:

- **States (S)**: A finite set of states representing all possible conditions the system can be in.

- **Actions (A)**: A finite set of actions available to the agent.
- **Transition Model (T)**: A function $T(s,a,s')=P(s'|s,a)$ that defines the probability of transitioning from state *s* to state *s*' under action ?*a*.
- **Observations (O)**: A finite set of observations that the agent can perceive.
- **Observation Model (Z)**: A function $Z(s',a,o)=P(o|s',a)$ that defines the probability of observing ? after taking action *a* and ending up in state *s*'.
- **Rewards (R)**: A function $R(s,a)$ that assigns a numerical reward to taking action *a* in state *s*.
- **Discount Factor (*γ*)**: A factor between 0 and 1 that discounts future rewards, reflecting the preference for immediate rewards over future gains.

Mathematical Framework of Partially Observable Markov Decision Process

The decision process in a POMDP is a cycle of states, actions, and observations. At each time step, the agent:

1. Observes a signal that partially reveals the state of the environment.
2. Chooses an action based on the accumulated observations.
3. Receives a reward dependent on the action and the underlying state.
4. Moves to a new state based on the transition model.

The key challenge in a POMDP is that the agent does not know its exact state but has a belief or probability distribution over the possible states. This belief is updated using the Bayes' rule as new observations are made, forming a belief update rule:

Bel(s’) =\frac{ P(o|s’,a) \sum_s P(s’|s,a) Bel(s)}{P(o|a, Bel)}

Where:

- Bel(*s*) is the prior belief of being in state *s*.
- Bel(*s*') is the updated belief after observing *o* and taking action *a*.

Strategies for Solving Partially Observable Markov Decision Processes

Partially Observable Markov Decision Processes (POMDPs) pose significant challenges in environments where agents have incomplete information. Solving POMDPs involves optimizing decision-making strategies under uncertainty, crucial in many real-world applications. This overview highlights key strategies and methods for addressing these challenges.

Belief State Representation:

In POMDPs, agents maintain a belief state—a probability distribution over all possible states—to manage uncertainty. This belief updates dynamically with actions and observations via Bayes' rule.

Solving Techniques:

1. **Value Iteration**: Extends traditional value iteration to belief states, using a piecewise linear and convex value function to calculate the expected rewards and update beliefs accordingly.
2. **Point-Based Methods**: These methods, such as Perseus and Point-Based Value Iteration (PBVI), focus on a select set of belief points to simplify computations and efficiently approximate the value function.
3. **Policy Search Methods**: Methods like QMDP and Finite-state controllers (FIB) search for optimal policies, sometimes simplifying the problem by assuming full observability post-action or using a finite set of controller states.
4. **Monte Carlo Methods**: Techniques like Partially Observable Monte Carlo Planning (POMCP) and Despot leverage Monte Carlo simulations within a tree search framework to estimate policy values under uncertainty, focusing on key scenarios to reduce complexity.

These methods illustrate the ongoing advancements in computational techniques to manage and solve the complexities of POMDPs, enhancing decision-making in uncertain environments.