Markov Decision Processes

In reinforcement learning I want an evaluation of the answer. I cannot show the answer to my model for the input(difference between supervised learning and reinforcement learning) I can only give a feedback to the model(Good/Bad).

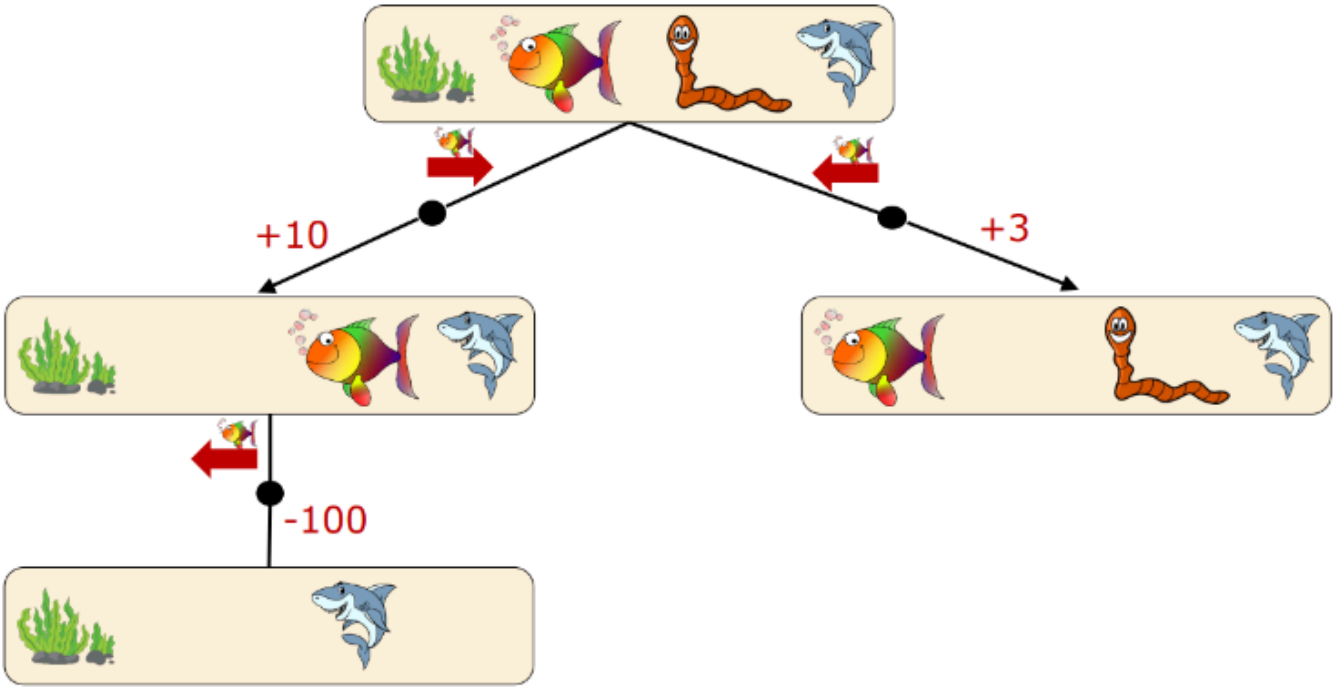
Agent-Environment Interface

The goal is to model simple problem(in terms of interactions) but also models problems where the model needs to take a sequence of decision to reach our goals. The optimality of the actions depends on the input and we don't have examples on the correct sequences. Each actions/decision taken by the model will have consequences.

Sequential Decision Making

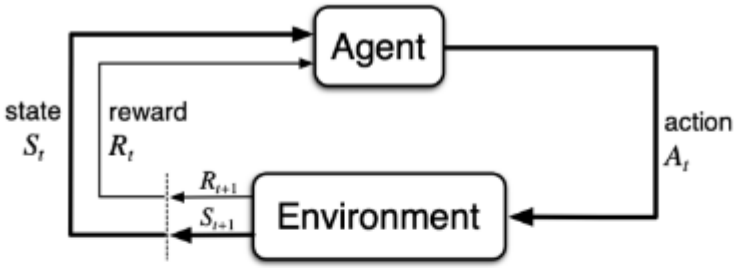
In sequential decision making...

- we take a sequence of decisions (or actions) to reach the goal
- the optimal actions are context-dependent
- we generally do not have examples of correct actions
- actions may have long-term consequences
- short-term consequences of optimal actions might seem negative



The above example take the instance where a fish search for food and gives feedback on the decisions the fish takes in terms of the value above the choice. I want the model with an optimal behaviour even if the surroundings changes. Nobody is telling me the best actions but they give me a numerical feedback. Given a specific input there will be a sequence of feedback and result and the model will use them to evaluate the best sequence of action to take.

Agent-Environment Interface



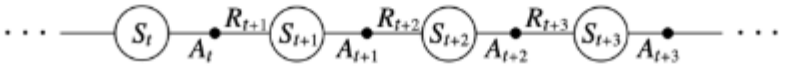
Agent and environment interact at discrete time steps: $t = 0, 1, 2, \mathrm{K}$

Agent observes state at step t: $S_t \in \mathcal{S}$

produces action at step t: $A_t \in \mathcal{A}(S_t)$

gets resulting reward: $R_{t+1} \in \mathcal{R}$

and resulting next state: $S_{t+1} \in \mathcal{S}$



Markov Decision Process

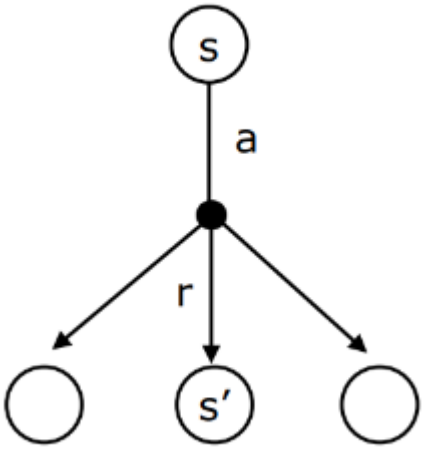
Markov Decision Process: One-Step Dynamics

Markov Property: future state (s') and reward (r) only depend on current state (s) and action (a)
(It is not a limiting assumption, it can be seen as a property of state)

In a Markov Decision Process (MDP), the one-step dynamic can be described as: $p(s', r|s, a)$

$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$

$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1 \; \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$



When holds the Markov Property and the state and action sets are finite, the problem is a finite Markov Decision Process. To define a finite MDP, you need to define:

- state and action sets

- one-step dynamics: $p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\}$
- we can also derive the next state distribution and expected reward as(they are useful for convenience):

$$p(s'|s, a) \doteq Pr\{S_{t+1} = s'|S_t = s, A_t = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a)$$

$$r(s, a) \doteq \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a)$$

Return

Agent should not choose actions on the basis of immediate reward. In fact, long-term consequences are more important than short-term reward. So, we need to take into account the sequence of future reward. We define **return**, G_t, as a function of the sequence of future rewards: $G_t \doteq f(R_{t+1} + R_{t+2} + R_{t+3} + \dots)$

To succeed, the agent will have to maximize the expected return $\mathbb{E}[G_t]$

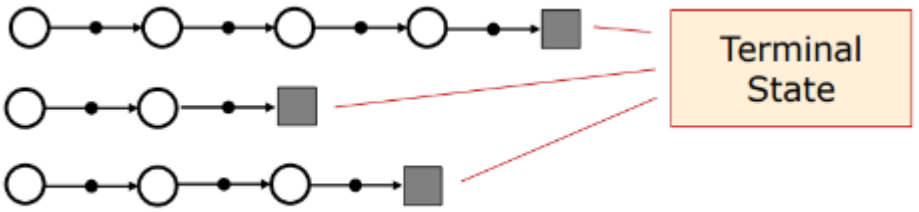
I can use as the function f():

- The sum of all the rewards(Total rewards)
- Average of the rewards(similar to the sum)
- Discounted rewards(for rewards very far in the view)

Episodic Task

For some problems we have a clear start and a clear end.

In episodic task the agent-environment interaction naturally breaks into chunks called episodes.



It is possible to maximize the expected total reward: $\mathbb{E}[G_t] = \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_T]$

Continuing Tasks

In continuing task the agent-environment interaction goes on continually and there are no terminal state. The total reward is a sum over an infinite sequence and might not be finite:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_{t+k} + \dots = \infty$$

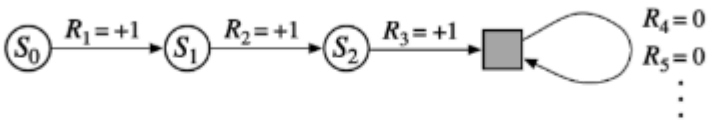
To solve this issue we can discount the future rewards by a factor of γ ($0 < \gamma < 1$):

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_{t+k} + \dots < \infty$$

Thus, the expected reward to maximize will be defined as: $\mathbb{E}[G_t] = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}]$

Unify Notation for Returns

In episodic tasks, we number from zero the time steps for each episode
We can design terminal state as **absorbing states** that always produce zero reward:



We can use the same definition of expected reward for episodic and continuing tasks:

$$\mathbb{E}[G_t] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right]$$

- where $\gamma = 1$ can be used if an absorbing state is always reached
- if γ =0, agent would only care about immediate reward
- As $\gamma \to 1$, agent would take future rewards into account more strongly

Goal and Reward
A goal should specify what we want to achieve, not how we want to achieve it
The Reward Hypotesis: That all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward).
Examples of reward design:

- goal-reward representation: 1 for goal, 0 otherwise
- action-penalty representation: -1 for not goal, 0 once goal reached
  If I am not using a discount factor the action-penalty representation will arrive at the solution faster, otherwise the goal-reward representation can arrive at the result faster for some values of γ
  Challenges to reward hypothesis
- How to represent risk-sensitive behavior?
- How to capture diversity in behavior?

Policy

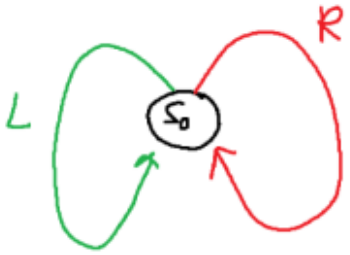
A policy, at any given point in time, decides which action the agent selects
A policy fully defines the behavior of an agent
Policies can be:

- Markovian / Non Markovian(We will focus on Markovian)
- Deterministic / Stochastic
- Stationary / Non Stationary(We will focus on Stationary)
  Deterministic Policy
  In the simplest case the policy can be modeled as a function $(\pi : \mathcal{S} \to \mathcal{A})$: $\pi(s) = a$
  Accordingly, the policy maps each state into an action
  This type of policy can be conveniently represented using a table

State	Action
s_0	a_1
s_1	a_0
s_2	a_0

Stochastic Policy
A more general approach is to model policy as function that maps each state to a probability distribution over the actions: $\pi(a|s)$

- $\sum_{a \in \mathcal{A}(s)} \pi(a|s) = 1$
- $\pi(a|s) \geq 0$
  A stochastic policy can be used to represent also a deterministic policy
  It is not much interesting to search an optimal stochastic policy as from it it can be derive a deterministic policy.
  Markovian/Non Markovian Policy



π_1: choose 50% R and 50% L

π_2: alternate R and L

Are π_1 and π_2 both markovian?
No! π_2 does not depend only from current state, so it is not a valid policy for us!
However, we can overcome this limitation by extending the state definition (e.g., in this case including previous action)

Value Functions

How can I evaluate the performance of my policy?
To evaluate the quality of different policies I have to introduce the state-value function:

$$V_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s]$$

It is the return value for what happens in average for the return on the status S. I am following policy pies for the expected return.
The action value function give an extra degree of freedom in the form of giving the first action:

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a]$$

it represent the expected return from a given state s, when a given action a is selected and then policy π is followed

Bellman Expecation Equation

The state-value function can again be decomposed into immediate reward plus discounted value of successor state:

$$V_\pi = \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s] = \sum_{a \in \mathcal{A}} \pi(a|a)(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V_\pi(s'))$$

The action-value function can be similarly decomposed:

$$Q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma V_p i(S_{t+1})|S_t = s, A_t = a]$$

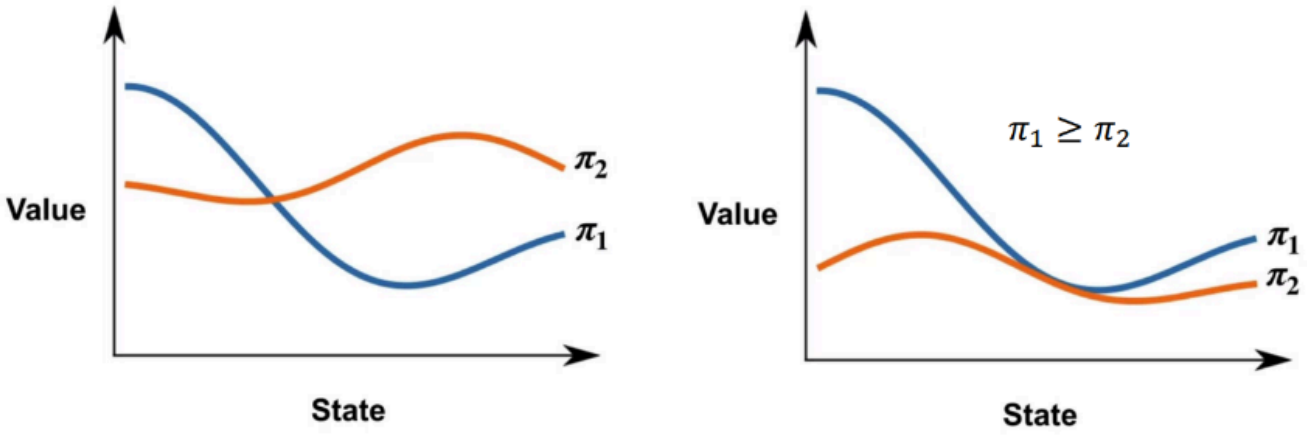
$$= r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V_\pi(s')$$

$$= r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \sum_{a' \in \mathcal{A}} \pi(a'|s')Q_\pi(s',a')$$

This helps as it take out from the calculation the value due to the prediction of the next values. I can do just the first step.
The limitation comes to the scalability as for each new state we need to add a new equation to be calculated(Unfeasible for larger number of states).

Optimality

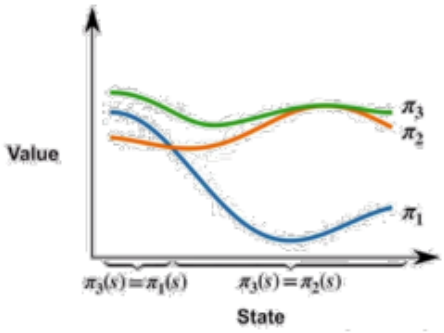
We want to find the optimal policy(better performance than the others)

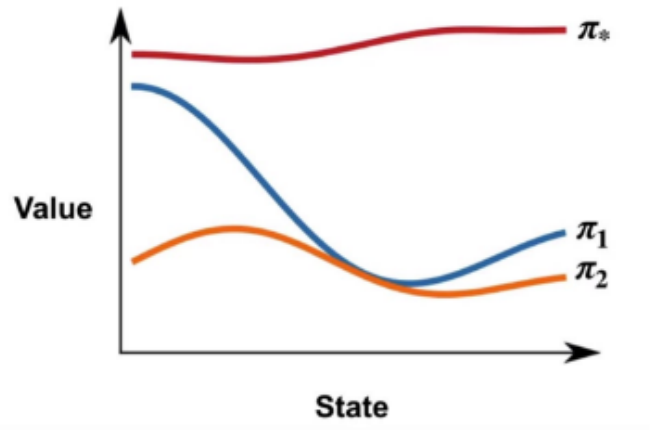


- We say that $\pi \geq \pi'$ if and only if $V_\pi(s) \geq V_{\pi'}(s), \quad \forall s \in \mathcal{S}$

For the evaluation I can use the value function. If two Values have some state with the same value we can choose either of them as they are equal in that specific case.
There is always a optimal deterministic policy π^* that is better or equal to all the others($\pi^* \geq \pi, \forall \pi$) So we can choose hte best policy for each state.

to all the others ($\pi^* \geq \pi, \ \forall \pi$)
ı of proof





We need to consider the optimal value function that is the function where in every state the Value function is optimal:

$$V^*(s) \doteq max_\pi V_\pi(s), \forall s \in \mathcal{S}$$

$$Q^*(s,a) \doteq max_\pi Q_\pi(s,a), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

The value function is the same(I there are some divergences the optimal actions are related to two different optimal policies). All these policies will have the same value and action function. V^*, Q^* I cn write the Bellman Expectation Equation, but in this case π^* is unknown but having a sum in respect of a choice on an action and knowing that π^* is maximising the value of the functions.

Computing the Optimal Value Function

$$V^*(s) = \max_a \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^*(s') \right\}$$

r, p, γ ⟶ Linear System Solver ⟶ V*

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^\pi(s') \right)$$

? π*, r, p, γ ⟶ Linear System Solver ⟶ V*

⟹ Dynamic Programming and Reinforcement Learning Algorithms

Using these formulas I can resolve the formula without knowing π but remove the linear property of the equations(recursive and presence of max).