

1 Introduction

1.1 Rewards, Observations, History, State

No supervisor, only reward signal. A **reward** R_t is a scalar feedback signal: indicates how well agent is doing at step t . The agent's job is to select actions to maximize (expected) cumulative reward!!

- actions may have long term consequences, rewards may be delayed;
- time really matters: data sequential, not *i.i.d.* like in supervised learning setting;
- it may be better to sacrifice *immediate reward* to gain more *long-term reward*;
- agent's actions affect data it receives.

Note

What if the goal is to complete a challenge in the shortest amount of time. Define reward on each step to be -1 and have a terminal state "goal completed".

The **history** is all the observed variables up to time t : $H_t = O_1A_1, R_2O_2A_2, \dots, R_tO_tA_t$. It is used by agent to generate next action. But it may be enormous amount of data... Instead we use **state** S_t which is the information used to model what to do next.

State is any function of history: $S_t = f(H_t)$. But bad choice of state may lose relevant information contained in history.

The **environment state** S_t^e is the environment's private internal representation, i.e. whatever it uses to generate next observation and reward. Usually it is not observed, so agent policies cannot depend on it.

The **agent state** S_t^a is the agent's internal representation, i.e. whatever information it uses to pick the next action. Formally, state is a function of the history $S_t^a = \underbrace{f}_{\text{our choice}}(H_t)$, where

$H_t = O_1A_1, R_2O_2A_2, \dots, A_{t-1}, R_tO_t$, i.e. summary of all observable variables up to time t . And it's our decision of what to remember, what to throw away and how to summarize the history!

An **information state** (a.k.a. **Markov state**) contains all useful information from the history. S_t is Markov iff

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_t, S_{t-1}, \dots, S_1]$$

- The future is independent of the past, given present: $H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$;
- The state is sufficient statistics of the future (i.e. fully characterizes the future) - once the state is known the history can be thrown away;
- When we model about distribution of some variable in the future, conditioning on the state is the same as conditioning on the history.

Note

By definition...

- ... environment state S_t^e is Markov! But we don't observe it.
- ... full history H_t is Markov! But it is too much.

So there is always a Markov state, it is just whether we can find a useful state.

Full observability (\Rightarrow **MDP**) is a particular case when agent directly observes environment:

$$O_t = S_t^e = S_t^a \quad (1)$$

Partial observability (\Rightarrow **POMDP**): agent *indirectly* observes environment $S_t^a \neq S_t^e$ and has to build his own state, for example:

- **Beliefs** of environment state: $S_t^a = \left(\mathbb{P}[S_t^e = s^1, \dots, S_t^e = s^m] \right)$
- dynamic updates of most probable state estimate: $S_t^a = w_s S_{t-1}^a + w_o O_t$

1.1.1 RL agent

An **RL agent** may include one or more of these components:

- **Policy**: agent's behavior function
 - map from state to action
 - **deterministic**: $a = \pi(s)$
 - **stochastic(probabilistic)**: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$
- **Value function**: how good is each state and/or action, prediction of expected future reward
 - State-value function $v(s) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} | S_t = s]$
 - Action-value function $q(s, a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} | S_t = s, A_t = a]$
- **Model**: how the agent thinks the environment works
 - **dynamics**: $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$ predicts the next state
 - **rewards**: $\mathcal{R}_s^a = \mathbb{E}_\pi[R_{t+1} | S_t = s, A_t = a]$ predicts the next immediate reward
 - it is optional to build model of environment, there are model-free methods

Note

The way to interpret γ is that we effectively look $\frac{1}{1-\gamma}$ steps into the future.

- $\gamma = 0.9 \rightarrow 10$ steps
- $\gamma = 0.99 \rightarrow 100$ steps

1.1.2 Taxonomy of RL agent

Not all of the components above are necessary. Below is taxonomy of agents based on the presence of certain elements.

- **Value Based**:
 - no explicit policy;
 - value function;
 - $\pi(s) = \operatorname{argmax}_{a \in A} \mathbb{E}_{s'|s,a} V(s')$.
- **Policy Based**:
 - Policy;
 - no value function;
- Actor Critic (to be discussed later)

Model Free vs **Model Based**. Reminder: Model = attempt to understand the environment.

1.2 RL vs Planning

- **Reinforcement learning**
 - The environment is initially unknown
 - but agent improves the policy by interacting with environment
- **Planning**
 - The model of the environment is (fully) known
 - agent interacts with this model (emulator of reality/environment) without external interaction;
 - agent improves policy from only interaction with the model/emulator.

These are different formulations, but of course related, as one way to solve RL problem is to learn model of environment first then do planning.

RL is like trial-and-error learning

1.3 Prediction vs Control

- **Prediction**: evaluate the future. Given fixed policy, evaluate value function.
- **Control**: optimize the future, i.e. find the best policy.

Usually we need to solve Prediction problem to solve control problem. We need to be able to evaluate all of our policies in order to find the best one.

2 Markov-Decision Process

MDP formally describes environment, where environment is fully observable, i.e. current (agent's) state completely characterizes that process, how the process unfolds.

Almost all RL problems can be formalized as MDPs:

- Optimal control primarily deals with continuous MDPs (with continuous actions).
- POMDP can be converted to MDP .
- *Bandits* are MDPs with one state.

Markov process State chain	→	Markov Reward Process +reward function	→	Markov Decision Process +actions
-------------------------------	---	---	---	-------------------------------------

2.1 Markov Process (MP)

Markov Property: future is independent of the past given present.

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_t, S_{t-1}, \dots, S_0] \quad (2)$$

It means that we can define a **state-transition probability matrix**

$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s] \quad (3)$$

Markov Process (Markov Chain) is a memory-less process defined as a tuple $\mathcal{MP} = (\mathcal{S}, \mathcal{P})$:

- \mathcal{S} is a (finite) set of states;
- \mathcal{P} is state transition probability matrix: $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$

This fully defines dynamics of the system.

Note

What to do if probability matrix change in time? Non-stationary Markov process (or non-stationary MDP in general).

1. You can use the same algorithms we use in the stationary case but incrementally adjust your solution to track the best solution you've found so far.
2. or you can reduce non-stationary dynamics to a more complex Markov process (see 12:00 of the video with Facebook example, where probability of staying on Facebook reduces with time or number of visits)

2.2 Markov Reward Process

Markov Reward Process (MRP) is a Markov chain with values defined as a tuple $\mathcal{MRP} = (\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$:

- \mathcal{R} is a **reward function**: $\mathcal{R}_s = \mathbb{E}[R_{t+1} | S_t = s]$
 - how much (expected) reward do I get from simply being in that state;
- γ is a **discount factor** $\gamma \in [0, 1]$

Reward function allows us to value how much reward I accumulate *across a particular sequence* sampled from this Markov process.

The **return** G_t is the total discounted reward from time-step t (across entire chain).

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (4)$$

Note

Why use discount factor γ ?

- admit the fact that our model is imperfect, hence future is more uncertain than present;
 - we build \mathcal{MRP} to represent environment, but it is imperfect model of the environment;
 - if we build a long plan that says I have to wait for many steps before collecting a large reward, I have to really trust my model and really believe that things will turn out as I plan to wait for so long...
- γ helps to define G_t to converge to finite!!
 - avoids infinite returns G in cyclic \mathcal{MRP} ;
 - makes sure there is no infinite evaluation of a state.
- *Myopic* ($\gamma = 0$) vs *far-sighted* ($\gamma = 1$)
 - we introduce a judgement here - we prefer a short-term reward to delayed reward and amount is given by γ ;
 - the way to interpret γ is that we effectively look $\frac{1}{1-\gamma}$ steps into the future.
- if the reward is financial, then money now worth more than money later (interest);
- it is sometimes possible to use *undiscounted* ($\gamma = 1$) MRPs, e.g. if we know that all sequences terminate.
 - approach: average reward formulation

2.2.1 Value Function and Bellman Equation for MRPs

Value Function $v(s)$ of \mathcal{MRP} gives long-term value of a state s

$$v(s) = \mathbb{E}[G_t | S_t = s] \quad (5)$$

- If I drop you into the state s of this MRP, what is the expected total reward I am going to collect?
- We use expectation, because environment is stochastic.

The value function can be decomposed into two parts:

- expected immediate reward R_{t+1} ;
- expected discounted value of successor state $\gamma v(S_{t+1})$.

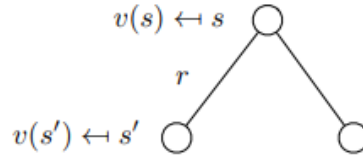
$$\begin{aligned}
 v(s) &= \mathbb{E}[G_t | S_t = s] = \mathbb{E}[R_{t+1} + \underbrace{\gamma R_{t+2} + \gamma^2 R_{t+3} + \dots}_{\gamma G_{t+1}} | S_t = s] \\
 &= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \underbrace{\mathbb{E}[\mathbb{E}[G_{t+1} | S_{t+1} = s', S_t = s] | S_t = s]}_{v(s')} \\
 &= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[v(S_{t+1}) | S_t = s]
 \end{aligned}$$

Note

Why immediate reward when we exit state S_t is R_{t+1} ? That's just a convention about when tick-counter gets updated. In this convention tick is updated at when environment takes control and emits reward and observation.

Bellman equation is a tautological definition of value function and can be thought as a **one step look-ahead search** (backup diagram).

$$\begin{aligned}
 v(s) &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \\
 &\Updownarrow \\
 v(s) &= \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')
 \end{aligned}$$



In matrix form

$$\begin{aligned}
 v &= \mathcal{R} + \gamma \mathcal{P}v \\
 \begin{bmatrix} v(s_1) \\ \vdots \\ v(s_n) \end{bmatrix} &= \begin{bmatrix} \mathcal{R}_{s_1} \\ \vdots \\ \mathcal{R}_{s_n} \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \vdots \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(s_1) \\ \vdots \\ v(s_n) \end{bmatrix}
 \end{aligned} \tag{6}$$

It is a linear equation and can be solved directly

$$v = (I - \gamma \mathcal{P})^{-1} \mathcal{R} \tag{7}$$

- Complexity is $\mathcal{O}(n^3)$, so can only be solved for small problems.
- For larger problems there are other methods
 - Dynamic programming;
 - Monte-Carlo evaluation;
 - Temporal-Difference learning.

2.3 Markov Decision Process (MDP)

This is the thing we are actually solving in RL

MDP = MRP + actions

Markov Decision Process (MDP) is a tuple $\mathcal{MDP} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$

- \mathcal{S} is a finite set of states;
- \mathcal{A} is a finite set of actions;
- \mathcal{P} is a state transition prob matrix (may depend on action)

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- \mathcal{R} is a reward function (may depend on action):

$$R_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

- γ is a discount factor.

There is much more control now for agent to control the path over the states (agency)

A (stochastic) policy is a distribution over actions given state

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s] \quad (8)$$

- This is something that agent controls!
- A policy fully defines *behavior* of an agent;
- it is convenient to make it stochastic to later introduce *exploration*.
- **Markov property**: in MDP, policy depends only on the current state, not on the history^a. as a result we consider *stationary policies*: $\forall t : A_t \sim \pi(\cdot | s_t)$
- to follow a policy π means we sample actions from $A_t \sim \pi(a | S_t)$

^aReminder: Markov assumption says that future is fully characterized by current state

Important reductions

If I have an $\mathcal{MDP} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ and a (fixed) policy $\pi(a|s)$, then

- $MDP \rightarrow MP^\pi$: the state sequence S_1, S_2, \dots defines a $\mathcal{MP}^\pi = (\mathcal{S}, \mathcal{P}^\pi)$
- $MDP \rightarrow MRP^\pi$: the state-reward sequence $S_1, R_2, S_2, R_3, \dots$ defines a $\mathcal{MRP}^\pi = (\mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma)$

where we just average over the policy:

$$\mathcal{P}_{ss'}^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^a = \sum_{a \in \mathcal{A}} \mathbb{P}[A_t = a | S_t = s] \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] \quad (9)$$

$$\mathcal{R}_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a \text{ (see 2.2)} \quad (10)$$

2.3.1 State-value and Action-value functions for MDP

Our previous definition of value function worked for a \mathcal{MRP} , where we just flew through the states but did not have any control (agency). There is no single expectation, expectation depends on behavior we choose. We need to redefine notion of value function. We can do that if we fix the policy π .

The **state-value function** $v_\pi(s)$ of an \mathcal{MDP} is the expected return given I start at state s

and then follow the policy π :

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] \quad (11)$$

- meaning: how good it is to be in state s if we then follow policy π ;
- it is same as the value function of policy-induced \mathcal{MRP}^{π} π

The **action-value function** $q_{\pi}(s, a)$ of an \mathcal{MDP} is the expected return given I start at state s , make action a and then follow the policy π :

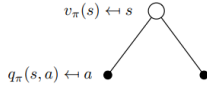
$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \quad (12)$$

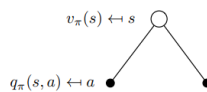
- meaning: how good it is to take action a from state s if we then follow policy π ;
- it is the quantity we are intuitively most interested in when selecting optimal action

2.3.2 Bellman equation

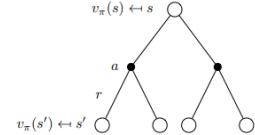
Again, same idea, the value function can be decomposed into expected immediate reward plus expected value of successor state given we follow the policy π . Same works for state- and action- value functions.

One-step lookaheads give us relation between v and q functions.

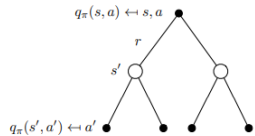
$$\begin{aligned} v_{\pi}(s) &= \mathbb{E}_{\pi}[G_t | S_t = s] \\ &= \mathbb{E}_{A_t \sim \pi(a|s)} \left[\underbrace{\mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]}_{q_{\pi}(s, a)} \right] \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_{\pi}(s, a) \end{aligned}$$


$$\begin{aligned} q_{\pi}(s, a) &= \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \mathcal{R}_s^a + \gamma \mathbb{E}_{s' \sim S_{t+1} | S_t = s, A_t = a} \left[\underbrace{\mathbb{E}_{\pi}[G_{t+1} | S_{t+1} = s', S_t = s, A_t = a]}_{v_{\pi}(s')} \right] \\ &= \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_{\pi}(s') \end{aligned}$$


Combining the two we get

$$v_{\pi}(s) = \underbrace{\sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a}_{\mathbb{E}_{\pi}[R_{t+1} | S_t = s]} + \gamma \underbrace{\sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_{\pi}(s')}_{\mathbb{E}_{\pi}[v(S_{t+1}) | S_t = s]}$$


and

$$q_{\pi}(s, a) = \underbrace{\mathcal{R}_s^a}_{\mathbb{E}_{\pi}[R_{t+1} | S_t = s, A_t = a]} + \gamma \underbrace{\sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a' | s') q_{\pi}(s', a')}_{\mathbb{E}_{\pi}[q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]}$$


The Bellman equation can be expressed in matrix form using π -induced MRP:

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

with direct solution

$$v_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

2.3.3 Optimal Value Function

The **optimal state-value function** $v_*(s)$ is the maximum state-value function over all policies:

$$v_*(s) = \max_{\pi} v_\pi(s)$$

The **optimal action-value function** $q_*(s, a)$ is the maximum action-value function over all policies:

$$q_*(s, a) = \max_{\pi} q_\pi(s, a)$$

Once we know $q_*(s, a)$, we immediately get π_* !

$$\pi_*(s) = \operatorname{argmax}_{a \in \mathcal{A}} q_*(s, a)$$

Note

When drawing state transition diagram, it is convenient to:

- assign $v_\pi(s)$ to nodes (states);
- assign $q_\pi(s, a)$ to edges/arcs (action from a particular state).

2.3.4 Bellman Optimality Equation

- Non-linear due to max operator;
- no closed form solution (in general);
- many iterative solution methods:
 - Value iteration
 - Policy iteration
 - Q-learning
 - Sarsa

Principle of optimality: to behave optimally, I should make an optimal step and then behave optimally from the new state.