– SUS 2020–
Lecture 11: Reinforcement Learning

Hung Son Nguyen

MIM
University of Warsaw

Based on slides by Zoubin Ghahramani and Carl Edward Rasmussen

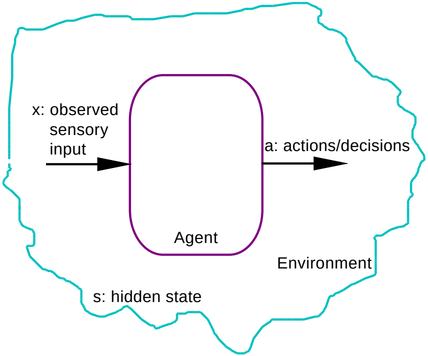
Outline

1. **Introduction**
2. Markov Decision Problems (MDPs)
3. Basic Approach

Intelligent Behaviour?

Imagine a creature/agent (human/animal/machine) which receives sensory inputs and can take some actions in an environment:

- Assume that the creature also receives rewards (or penalties/losses) from the environment.
- The goal of the creature is to maximise the rewards it receives (or equivalently minimise the losses).



A theory for choosing actions that minimize losses is a theory for how to behave optimally...

Bayesian Decision Theory

Bayesian decision theory deals with the problem of making optimal decisions—that is, decisions or actions that minimize an expected loss.

- Let's say we have a choice of taking one of k possible **actions** $a_1 \ldots a_k$.
- Assume that the world can be in one of m different **states** $s_1, \ldots, s_m$.
- If we take action a_i and the world is in state s_j we incur a **loss** ℓ_{ij}
- Given all the observed data $\mathcal{D}$ and prior background knowledge $\mathcal{B}$, our **beliefs** about the state of the world are summarized by $p(s|\mathcal{D}, \mathcal{B})$.
- *The optimal action is the one which is expected to minimize loss (or maximize utility)*:

$$a^* = \arg\min_{a_i} \sum_{j=1}^{m} \ell_{ij}\, p(s_j|\mathcal{D}, \mathcal{B})$$

Bayesian sequential decision theory	(statistics)
Optimal control theory	(engineering)
Reinforcement learning	(computer science / psychology)

A simple example

Assume we have two actions:

a_1 : play

a_2 : don't play

And two outcomes:

s_1 : win lottery

s_2 : don't win lottery

transition	loss
$p(s_1 \vert a_1, \mathcal{B}) = 0.000001$	$\ell_{11} = -100000$
$p(s_2 \vert a_1, \mathcal{B}) = 0.999999$	$\ell_{12} = +0.9$
$p(s_1 \vert a_2, \mathcal{B}) = 0$	$\ell_{21} = 0$
$p(s_2 \vert a_2, \mathcal{B}) = 1$	$\ell_{22} = 0$

Optimal action:

$$a^* = \arg\min_{a_i} \sum_{j=1}^{m} \ell_{ij}\, p(s_j|a_i, \mathcal{B})$$

What is the optimal action for this decision problem?

Comments about the above framework

The optimal action is the one which is expected to minimize loss (or maximize utility):

$$a^* = \arg\min_{a_i} \sum_{j=1}^{m} \ell_{ij}\, p(s_j|\mathcal{D}, \mathcal{B})$$

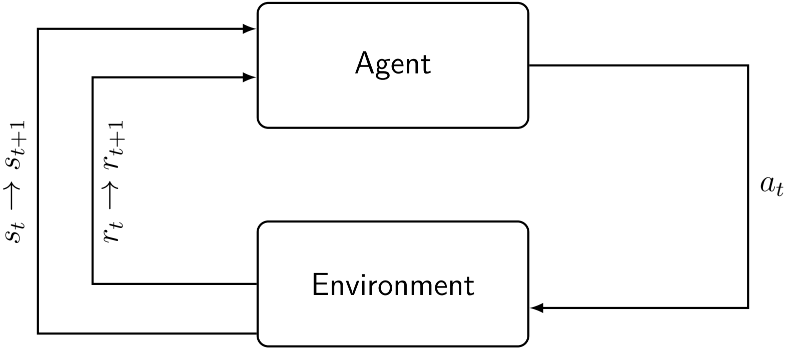
- This is a theory for how to make a *single decision*. How do we make a *sequence of decisions* in order to achieve some long-term goals/rewards?
- This assumes that we *know* what the losses are for each action-state pair. The losses may in fact have to be learned from experience.
- We need a model for how the observed data $\mathcal{D}$ relates to the states of the world s.
- It may be impossible to *enumerate* all possible actions and states. What about continuous state and action spaces?

Basics

- Reinforcement learning is an area concerned with how an agent ought to take actions in an environment so as to maximize some notion of reward.
- "A way of programming agents by reward and punishment without needing to specify how the task is to be achieved."
- Specify what to do, but not how to do it.
  - Only formulate the reward function.
  - Learning "fills in the details".
- Compute better final solutions for a task.
  - Based on actual experiences, not on programmer assumptions.
- Less (human) time needed to find a good solution.

Outline

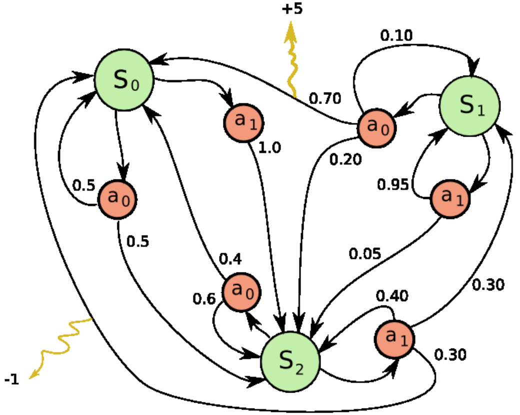
Intelligent agent systems



Markov Decision Problems (MDPs)

MDP is a 5-tuple $(S, A, \mathcal{P}, \mathcal{R}, \gamma)$ where:

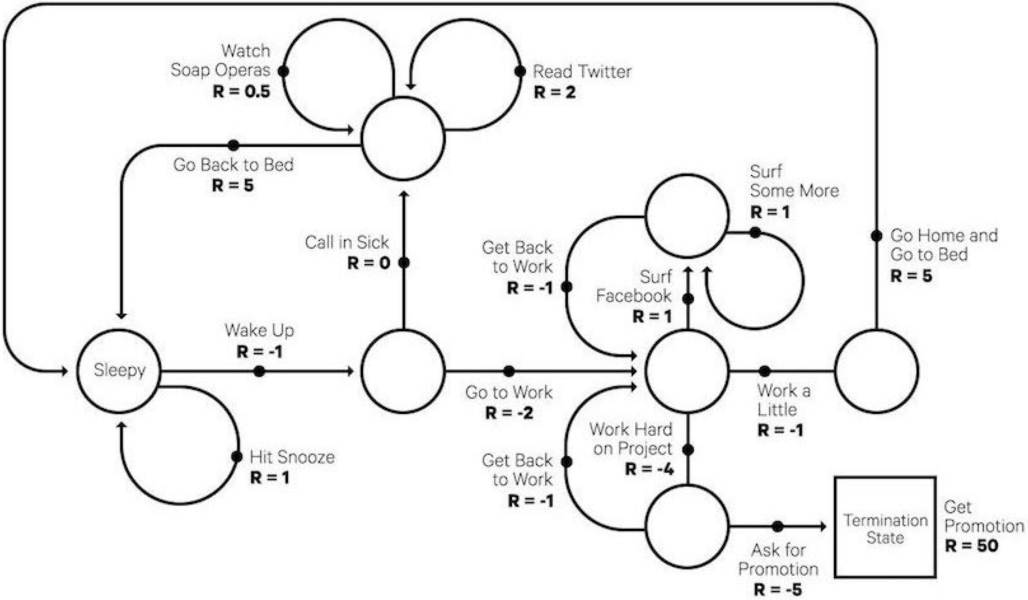
S = finite set of states
A = finite set of actions
$\mathcal{P}$ = state-transition probability matrix
$\mathcal{R}$ = reward function
$\gamma \in [0, 1]$ = discount factor



At the time step t we denote:
States: s_t
Actions: a_t
Rewards: r_t

Discounted sum of rewards from timestep t: $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$

Markov Decision Problems (MDPs)



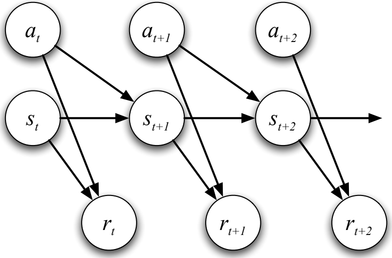
Markov Decision Problems (MDPs)

States: s_t
Actions: a_t
Rewards: r_t

The variable s_t is the state of the world and agent at time t

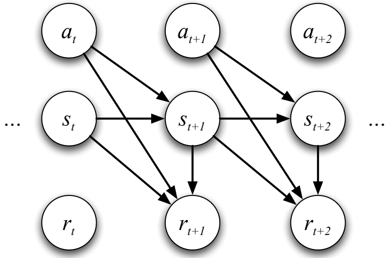
The agent takes action a_t and receives reward r_t (or loss, if you like to think negatively...)

The reward is assumed to depend on the state and the action.



Markov property: $p(s_{t+1}, r_t | s_t, a_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots) = p(s_{t+1}, r_t | s_t, a_t)$

Markov Decision Problems (MDPs)



The world is characterized by

Transition Probabilities: $\mathcal{P}^a_{ss'} = p(s_{t+1} = s' | s_t = s, a_t = a)$

Expected rewards: $\mathcal{R}^a_{ss'} = \mathbb{E}[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s']$

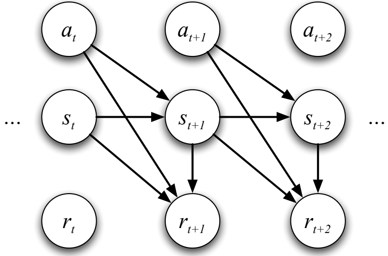
The agent is characterized by

Policy: $\pi(s, a) = p(a_t = a | s_t = s)$

Why is the action at time t only dependent on the state at time t?

Markov Decision Problems (MDPs)

Why is the action at time t only dependent on the state at time t?



Actions a_t should be chosen to maximize sum of (discounted) future rewards R_t.

By the Markov properties in the graph (i.e. conditional independence), future rewards and states are independent of past rewards, actions, and states given s_t and a_t:

$$p(s_{t+1}, r_{t+1}, s_{t+2}, \ldots | s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = p(s_{t+1}, r_{t+1}, s_{t+2}, \ldots | s_t, a_t)$$

If s_t is known, the *expected* value of the return R_t depends only on a_t, so previous states and actions are irrelevant.

Value Functions

Value Function: how good is it to be in a given state? This obviously depends on the agent's policy:

$$V^\pi(s) \;=\; \mathbb{E}_\pi[R_t|s_t = s] \;=\; \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle|s_t = s\right].$$

State-action value function: how good is it to be in a given state and take a given action, and then follow policy π:

$$Q^\pi(s,a) \;=\; \mathbb{E}_\pi[R_t|s_t = s, a_t = a] \;=\; \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle|s_t = s, a_t = a\right].$$

The relation between the state value function and the state-action value function:

$$V^\pi(s) \;=\; \sum_a \pi(s,a)Q^\pi(s,a)$$

Self-Consistency of Value Functions

A fundamental property of value functions is that they satisfy a set of recursive consistency equations. V^π is the unique solution to these equations.

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi[R_t | s_t = s] = \mathbb{E}_\pi\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s \right] \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_{t+1} = s' \right] \right) \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right)
\end{aligned}
$$

We can solve them using a "backup operation" from $s' \to s$ (or other means). Linear system of $N \equiv |s|$ equations in N unknowns.

$$
\mathbf{v} = \left(I - \gamma \sum_a \mathrm{diag}(\boldsymbol{\pi}_a) \mathcal{P}^a \right)^{-1} \left(\sum_a \boldsymbol{\pi}_a \odot \mathrm{diag}(\mathcal{P}^a \mathcal{R}^{a\top}) \right)
$$

There is a similar equation for $Q^\pi(s,a)$

Optimal Policies and Values

Optimal Policy: π^* such that $V^{\pi^*}(s) \geq V^{\pi}(s)\ \forall s$.

There may be more than one optimal policy.

Question: Is there always at least one optimal policy? YES

Optimal state value function: $V^*(s) = \max_\pi V^\pi(s)\ \forall s$

Optimal state-action value function: $Q^*(s,a) = \max_\pi Q^\pi(s,a)\ \forall s$. This is the expected return of action a in state s, thereafter following optimal policy.

$$Q^*(s,a) \ = \ \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a].$$

Bellman Optimality Equation

N ***nonlinear*** **equations in** N **unknowns for** V^***.**

$$
\begin{aligned}
V^*(s) &= \max_a Q^{\pi^*}(s,a) = \max_a \mathbb{E}_{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right] \\
&= \max_a \mathbb{E}_{\pi^*}\left[r_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t = s, a_t = a\right] \\
&= \max_a \sum_{s'} \mathcal{P}^a_{ss'}\left(\mathcal{R}^a_{ss'} + \gamma V^*(s')\right)
\end{aligned}
$$

NA ***nonlinear*** **equations in** NA **unknowns for** Q^*

$$
\begin{aligned}
Q^*(s,a) &= \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \middle| s_t = s, a_t = a\right] \\
&= \sum_{s'} \mathcal{P}^a_{ss'}\left(\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\right)
\end{aligned}
$$

Outline

Main Notions: Policies

- **Policy**: The function that allows us to compute the next action for a particular state.

$$\pi(s,a) = p(a_t = a | s_t = s)$$

- An **optimal Policy** is a policy that maximizes the expected reward/reinforcement/feedback of a state:

$$\pi^* \text{ such that } V^{\pi^*}(s) \geq V^{\pi}(s) \text{ for all } s.$$

Optimal state value function: $V^*(s) = \max_\pi V^\pi(s) \; \forall s$
Optimal state-action value function: $Q^*(s,a) = \max_\pi Q^\pi(s,a) \; \forall s, a$.
Property of the **optimal policy**

$$Q^*(s,a) \;=\; \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \text{ for all } a \text{ and } s$$

- Thus, the task of RL is to use observed rewards to find an optimal policy for the environment.

Main Notions: Modes of Learning

- *Passive Learning*: Agents policy is fixed and our task is to learn how good the policy is.
- *Active Learning*: Agents must learn what actions to take.
- *Off-policy learning*: learn the value of the optimal policy independently of the agent's actions.
- *On-policy learning*: learn the value of the policy the agent actually follows.

Main Notions: Exploration vs. Exploitation

- *Exploitation* Use the knowledge already learned on what the next best action is in the current state.
- *Exploration* In order to improve policies the agent must explore a number of states. I.e., select an action different of the one that it currenlty thinks is best.

Difficulties of Reinforcement learning

- *Blame attribution problem*: The problem of determining which action was responsible for a reward or punishment.
  - Responsible action may have occurred a long time before the reward was received.
  - A combination of actions might have lead to a reward.
- *Recognising delayed rewards*: What seem to be poor actions now might lead to much greater rewards in the future than what appears to be good actions.
  - Future rewards need to be recognised and back-propagated.
  - Problem complexity increases if the world is dynamic.
- *Explore-exploit dilemma*: If the agent has worked out a good course of actions, should it continue to follow these actions or should it explore to find better actions?
  - Agent that never explores can not improve its policy.
  - Agent that only explores never uses what it has learned.

Solving MDPs

Given the optimal value function, V^*, it is easy to get optimal policy π^*: be **greedy** w.r.t. V^*.

If you have V^*, the actions that appear best after a one-step search will be optimal.

V^* turns a long-term reward into a quantity that is locally and immediately available.

Using Q^* it is even easier to get the optimal policy:

$$\pi^*(s,a) = 0 \;\; \forall a \;\; s.t. \;\; Q^*(s,a) \neq \max_{a'} Q^*(s,a')$$

Policy Improvement Theorem

Policy Evaluation $$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}_{ss'}^{a} \Big(\mathcal{R}_{ss'}^{a} + \gamma V_k^{\pi}(s') \Big)$$

assumes $\mathcal{P}$ known, $\mathcal{R}$ known, and a full backup (we can also sweep in place)

Policy Improvement Theorem

$$Q^{\pi}(s, \pi'(s)) \ \geq \ V^{\pi}(s) \ \forall s \quad \Longrightarrow \quad V^{\pi'}(s) \ \geq \ V^{\pi}(s)$$

Proof:

$$\begin{aligned}
V^{\pi}(s) \ &\leq \ Q^{\pi}(s, \pi'(s)) \\
&= \ \mathbb{E}_{\pi'}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s] \\
&\leq \ \mathbb{E}_{\pi'}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi'(s_{t+1})) | s_t = s] \\
&= \ \mathbb{E}_{\pi'}\Big[r_{t+1} + \gamma \mathbb{E}_{\pi'}[r_{t+2} + \gamma V^{\pi}(s_{t+2})] | s_t = s\Big] \\
&= \ \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 V^{\pi}(s_{t+2}) | s_t = s] \\
&\vdots \\
&\leq \ V^{\pi'}(s)
\end{aligned}$$

Policy Iteration

The policy improvement theorem suggests a way of improving policies:

$$\begin{aligned}\pi'(s) &\leftarrow \arg\max_a Q^\pi(s,a) \;\; \forall s \\ &= \arg\max_a \mathbb{E}[r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a]\end{aligned}$$

This procedure converges to an optimal policy by policy improvement theorem and Bellman optimality.

$$V^{\pi'}(s) \geq \arg\max_a Q^\pi(s,a) \geq \sum_a \pi(s,a) Q^\pi(s,a) = V^\pi(s)$$

Policy Iteration: Iterates between evaluation and improvement

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \ldots \pi^*$$

Problem with Policy Iteration: Evaluation step can be really slow...

Dynamic programming (DP) algorithm

This applied the Generalized Policy Iteration (GPI):

$$\pi_0 \xrightarrow{\text{eval}} V_{\pi_0} \xrightarrow{\text{imp}} \pi_1 \xrightarrow{\text{eval}} V_{\pi_1} \xrightarrow{\text{imp}} \pi_2 \xrightarrow{\text{eval}} \ldots \xrightarrow{\text{imp}} \pi_* \xrightarrow{\text{eval}} V_*$$

Policy improvement step:

$$\begin{aligned} \pi'(s) &\leftarrow \arg\max_a Q^\pi(s,a) \;\; \forall s \\ &= \arg\max_a \mathbb{E}[r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a] \end{aligned}$$

Value evaluation step:

$$\begin{aligned} V_{k+1}(s) &\leftarrow \max_a \mathbb{E}[r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s, a_t = a] \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'} \Big(\mathcal{R}^a_{ss'} + \gamma V_k(s')\Big) \end{aligned}$$

Value Iteration

- Do we really need to wait until convergence of the evaluation step?
- In fact, we can improve after **one** sweep of evaluation!

$$\begin{aligned} V_{k+1}(s) &\leftarrow \max_a \mathbb{E}[r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s, a_t = a] \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s') \right) \end{aligned}$$

  converges: $V_k \longrightarrow V^*$.
  At each step we also have a policy.
- Problem: it is still not feasible to update the value of every single state.
  E.g. backgammon has 10^{20} states!
- Bellman called this the **curse of dimensionality**

Asynchronous dynamic programming

These are in-place iterated dynamic programming (DP) algorithms that are not organized in terms of systematic sweeps over all the states.

- *States are backed-up in order visited or randomly.*
  To converge the algorithms must continue to visit every state.
- Key idea in RL: We can run the DP algorithm at the same time as the agent is *actually experiencing* the MDP.
- This leads to an **exploration vs exploitation tradeoff**: act so as to visit new parts of state space or exploit already visited part of state-space?
- An example of a simple exploration strategy are ε-greedy policies:

  $$\pi_\varepsilon(s, a) = (1 - \varepsilon)\pi(s, a) + \varepsilon \cdot u(a)$$

  where $u(a)$ is a uniform distribution over actions.
- *Can you think of anything wrong with this?*

Some Other Algorithms (next lecture)

- Temporal Difference learning
- Q-learning
- SARSA
- Monte Carlo Method
- Evolutionary Algorithms