Lecture 1: Introduction

(INF8250AE: Introduction to Reinforcement Learning)

Amir-massoud Farahmand

Polytechnique Montréal & Mila

Adaptive Agents Lab (Adage)

Table of Contents

1. RL Problem
2. Markov Decision Process (MDP)
3. From Immediate to Long-Term Reward
4. Optimal Policy and Optimal Value Function
5. An Instance of an RL Algorithm: Q-Learning
    - Exploration vs. Exploitation
6. Logistics
7. References

Reinforcement Learning

Figure: An agent ...

Reinforcement Learning

Figure: ... observes the world ...

Reinforcement Learning

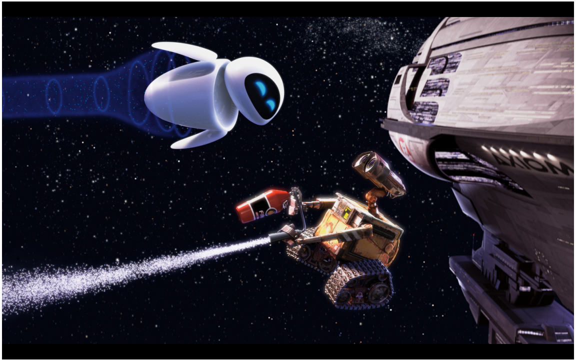


Figure: ... takes an action and its states changes ...

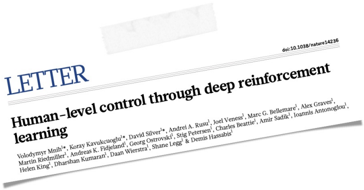
Reinforcement Learning

Figure: ... with the goal of achieving long-term rewards.

Reinforcement Learning: Examples

- Robot manipulator
  - Perceive: cameras, sensors for joint angles, force sensors, etc.
  - Act: Joint position or velocity
  - Goal: Build a car as fast as possible
- Smart HVAC (Heating, Ventilation, and Air Conditioning) system
  - Perceive: thermometers, humidity sensors, CO_2 meters, infrared cameras (temperature profiles)
  - Act: vary temperature, humidity, and airflow rates of vents
  - Goal: Maintain comfortable indoor environment efficiently

Reinforcement Learning in the News

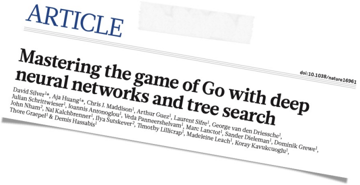


LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]



ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

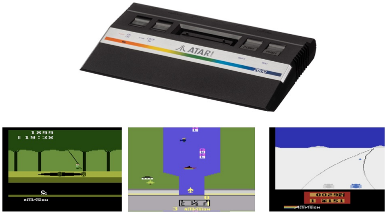


Figure: Some recent success stories!

(Potential) Applications of RL

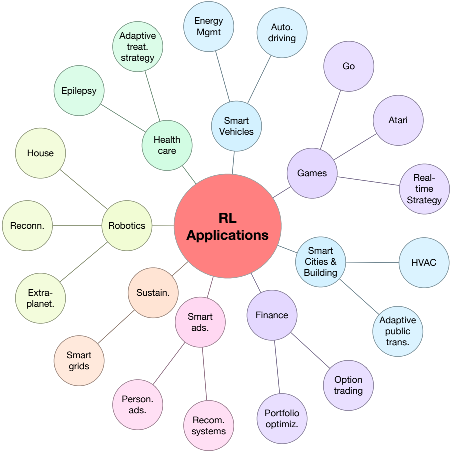


Figure: And a lot of potential applications

This course ...

This course is about reinforcement learning (RL) and sequential decision-making under uncertainty with an emphasis on theoretical understanding.

We build the foundation, step by step, prove many results, and try to gain an understanding of why many algorithms are designed the way they are, and why they work.

Reinforcement Learning: Problem and Methods

Reinforcement Learning (RL) refers to both a type of problem and a set of computational methods.

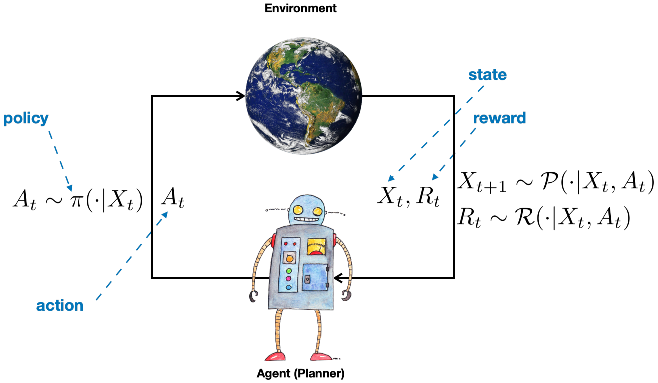
- **Problem:** How to act so that some notion of long-term performance is maximized?
- **Methods:** What kind of computation does an agent need to do in order to ensure that its actions lead to good (or even optimal) long-term performance?

Remark

Historically, only a subset of all computational methods that attempt to solve the RL problem are known as the RL methods. For example, Q-Learning is; evolutionary computation methods are not.

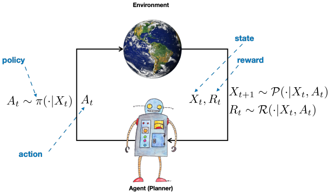
RL Problem and Agent-Environment Interaction

In RL, we often talk about an agent and its environment, and their interaction.


Environment
state
reward
policy
$A_t \sim \pi(\cdot|X_t)$
A_t
X_t, R_t
$X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
action
Agent (Planner)

RL Problem and Agent-Environment Interaction

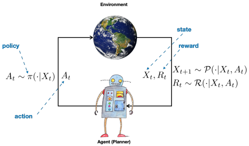
- Agent: decision maker and/or learner
  - robot
  - medical diagnosis and treatment system
  - air conditioning system
- Environment: anything outside the agent with which it interacts and attempts to control.
  - physical world outside the robot
  - patient's body
  - room


Environment
state
policy
reward
$A_t \sim \pi(\cdot|X_t)$
A_t
X_t, R_t
$X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
action
Agent (Planner)

RL Problem and Agent-Environment Interaction

At time $t = 1, 2, \ldots$, the interaction of the agent and the environment is as follows:

- the agent observes its state X_t in the environment.
  - Examples: position of the robot, vital information of a patient, the room temperature, etc.
- The agent picks an action A_t according to its policy π, e.g., $A_t = \pi(X_t)$ or $A_t \sim \pi(\cdot|X_t)$.
- The state of the agent in the environment changes and becomes X_{t+1} according to transition probability kernel (or distribution), i.e., $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$.
- The agent also receives a reward signal R_t, i.e., $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$.

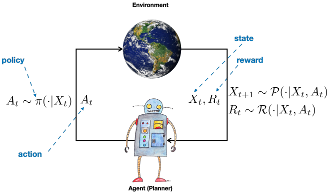

Environment
state
reward
policy
$A_t \sim \pi(\cdot|X_t)$
A_t
X_t, R_t
$X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
action
Agent (Planner)

RL Problem and Agent-Environment Interaction

State: A variable that summarizes whatever has happened to the agent so far.

Policy: Action selection mechanism. Usually a mapping from states to actions. It can be deterministic ($A_t = \pi(X_t)$) or stochastic ($A_t \sim \pi(\cdot|X_t)$).

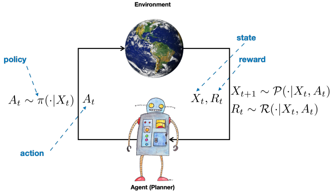
Transition probability kernel: Describes the dynamics. For example, a set of electromechanical equations describing how the position of the robot (including its joints) change when a certain command is sent to its motor. Or how the patient's physiology changes after the administration of the treatment.


Environment
state
reward
policy
$A_t \sim \pi(\cdot|X_t)$
A_t
X_t, R_t
$X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
action
Agent (Planner)

RL Problem and Agent-Environment Interaction

Reward: A real number specifying the *immediate* desirability of the choice of action A_t at the state X_t (possibly leading to state X_{t+1}) has been. Examples:

- Positive if robot successfully picks up an object, negative if it breaks the object.
- Infection subsides
- The room temperature becomes comfortable.


Environment
state
reward
policy
$A_t \sim \pi(\cdot|X_t)$
A_t
X_t, R_t
$X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
action
Agent (Planner)

Remark

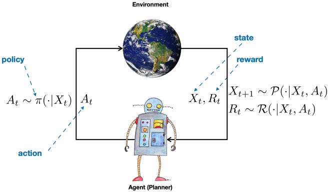
The reward signal/function/distribution only encodes the desirability of the action from the immediate perspective. A good action now may not be good in the long-term.

RL Problem and Agent-Environment Interaction

This process repeats and as a result, the agent receives a sequence of state, actions, and rewards:

$$X_1, A_1, R_1, X_2, A_2, R_2, \cdots .$$

This sequence might terminate after a fixed number of time steps (say, T), or until the agent gets to a certain region of the state space, or it might continue forever.


Environment
state
policy
reward
$A_t \sim \pi(\cdot|X_t)$
A_t
X_t, R_t
$X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
action
Agent (Planner)

Markov Decision Process (MDP)

Let us formally define some important concepts that we require throughout the course. Beforehand, some commonly used notations:
Given a space Ω.

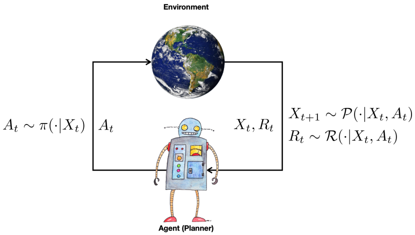
- $\mathcal{M}(\Omega)$: the space of all probability distributions defined over the space Ω.
- $B(\Omega)$: the space of all bounded functions defined over Ω

Examples: $\Omega = \{1, 2, \ldots, n\}, \mathbb{N}, \mathbb{R}, \mathbb{R}^d$, etc.
$\mathcal{M}(\mathbb{R})$: The space of distributions on the real line
$B(\mathbb{R})$: The space of bounded functions on the real line

Markov Decision Process (MDP)

Definition

A *discounted MDP* is a 5-tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{X}$ is a measurable state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X})$ is the transition probability kernel with domain $\mathcal{X} \times \mathcal{A}$, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathbb{R})$ is the immediate reward distribution, and $0 \leq \gamma < 1$ is the discount factor.


Environment
$A_t \sim \pi(\cdot|X_t)$
A_t
X_t, R_t
$X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
Agent (Planner)

Markov Decision Process (MDP)

MDPs encode the temporal evolution of a discrete-time stochastic process controlled by an *agent*.

- Initial state $X_1 \sim \rho$ with $\rho \in \mathcal{M}(\mathcal{X})$.
- Agent chooses action $A_t \in \mathcal{A}$.
- Agent goes to $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$ and receives reward $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$.
- The process repeats. The trajectory is $\xi = (X_1, A_1, R_1, X_2, A_2, R_2, \cdots)$, which is random.

Remark

The reward distribution could also depend on the next-state X_{t+1}. In that case, we would have a different reward kernel $\mathcal{R}'$ and the reward would be $R_t \sim \mathcal{R}'(\cdot|X_t, A_t, X_{t+1})$. But we can absorb the dynamics within $\mathcal{R}$, i.e., $\mathcal{R}(\cdot|x, a) = \int \mathcal{R}'(\cdot|x, a, x')\mathcal{P}(\mathrm{d}x'|x, a)$.

Markov Decision Process (MDP)

This is a general framework.
State space:

- Finite: $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ (or $\mathcal{X} = \{1, 2, \dots, n\}$) with $n < \infty$
- Infinite but countable: $\mathcal{X} = \{x_1, x_2, \dots\}$ (or $\mathcal{X} = \mathbb{N}$)
- Continuous: $\mathcal{X} \subset \mathbb{R}^d$

Dynamics:

- Stochastic
- Deterministic

Some Examples

- When $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$, the transition probability kernel $\mathcal{P}(\cdot|\cdot, a)$ is a matrix for any $a \in \mathcal{A}$. For example,

$$\mathcal{P}(\cdot|\cdot, a_1) = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}, \qquad \mathcal{P}(\cdot|\cdot, a_2) = \begin{bmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{bmatrix}.$$

- a dynamical system described by $x_{t+1} = f(x_t, a_t)$ where $x \in \mathbb{R}^m$, $a \in \mathbb{R}^n$, and $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$. For example,

$$f(x, a) = cx + a, \qquad f(x, a) = ax(1 - x)$$

  Q: What is $\mathcal{P}(\cdot|x, a)$?

Remark

A deterministic dynamical system always behave exactly the same given the same starting state and action. They can be described by the transition function $f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$ and $x_{t+1} = f(x_t, a_t)$.

Policy

Definition

A policy is a sequence $\bar{\pi} = \{\pi_1, \pi_2, \ldots\}$ such that for each t,

$$\pi_t(a_t | X_1, A_1, X_2, A_2, \ldots, X_{t-1}, A_{t-1}, X_t)$$

is a stochastic kernel on $\mathcal{A}$ given $\underbrace{\mathcal{X} \times \mathcal{A} \times \cdots \times \mathcal{X} \times \mathcal{A} \times \mathcal{X}}_{2t-1 \text{ elements}}$

satisfying

$$\pi_t(\mathcal{A} | X_1, A_1, X_2, A_2, \ldots, X_{t-1}, A_{t-1}, X_t) = 1$$

for every $(X_1, A_1, X_2, A_2, \ldots, X_{t-1}, A_{t-1}, X_t)$.

Policy

Definition

If π_t is parametrized only by X_t, that is

$$\pi_t(\cdot|X_1, A_1, X_2, A_2, \ldots, X_{t-1}, A_{t-1}, X_t) = \pi_t(\cdot|X_t),$$

$\bar{\pi}$ is a Markov policy.
If for each t and $(X_1, A_1, X_2, A_2, \ldots, X_{t-1}, A_{t-1}, X_t)$, the policy π_t assigns mass one to a single point in $\mathcal{A}$, $\bar{\pi}$ is called a *deterministic (nonrandomized)* policy; if it assigns a distribution over $\mathcal{A}$, it is called *stochastic* or *randomized* policy.
If $\bar{\pi}$ is a Markov policy in the form of $\bar{\pi} = (\pi, \pi, \ldots)$, it is called a stationary policy.

A policy $\pi(\cdot|x)$ is a stationary Markov policy. We often work with such policies. If it is also deterministic, we denote it by $\pi(x)$.

Policy-Induced Transition Kernels

An agent is “following” a Markov stationary policy π whenever A_t is selected according to the policy $\pi(\cdot|X_t)$, i.e., $A_t = \pi(X_t)$ (deterministic) or $A_t \sim \pi(\cdot|X_t)$ (stochastic).

The policy π induces two transition probability kernels $\mathcal{P}^\pi : \mathcal{X} \to \mathcal{M}(\mathcal{X})$ and $\mathcal{P}^\pi : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X} \times \mathcal{A})$. For a (measurable) subset A of $\mathcal{X}$ and a (measurable) subset B of $\mathcal{X} \times \mathcal{A}$ and a deterministic policy π, denote

$$(\mathcal{P}^\pi)(A|x) \triangleq \int_{\mathcal{X}} \mathcal{P}(\mathrm{d}y|x, \pi(x))\mathbb{I}_{\{y \in A\}},$$

$$(\mathcal{P}^\pi)(B|x, a) \triangleq \int_{\mathcal{X}} \mathcal{P}(\mathrm{d}y|x, a)\mathbb{I}_{\{(y, \pi(y)) \in B\}}.$$

Policy-Induced Transition Kernels

When we have a countable state-action space, we sometimes use summation instead of integrals. For example,

$$(\mathcal{P}^{\pi})(A|x) \triangleq \sum_{y\in\mathcal{X}} \mathcal{P}(y|x,\pi(x))\mathbb{I}_{\{y\in A\}} = \sum_{y\in A} \mathcal{P}(y|x,\pi(x)).$$

So for a particular $y \in \mathcal{X}$, we have $(\mathcal{P}^{\pi})(y|x) = \mathcal{P}(y|x,\pi(x))$. Also we can extend the definition of $\mathcal{P}^{\pi}$ to following a policy for m-steps $(m \geq 1)$ inductively. We use $(\mathcal{P}^{\pi})^{m}$ to denote such a transition kernel.

From Immediate to Long-Term Reward

RL problem: How to act so that some notion of long-term performance is maximized.
Q: What does long-term mean? How to quantify it?

Immediate Reward Problem

- At each round of interaction with its environment
  - An agent starts at a random state $X_1 \sim \rho \in \mathcal{M}(\mathcal{X})$
  - It chooses action $A_1 = \pi(X_1)$ (deterministic policy), and receives a reward of $R_1 \sim \mathcal{R}(\cdot|X_1, A_1)$.

We call each of these rounds an episode. Here the episode only lasts one time-step.

Q: How should this agent choose its policy in order to maximize its "performance"?

Q: What does performance mean?

Immediate Reward Problem

- At each round of interaction with its environment
  - An agent starts at a random state $X_1 \sim \rho \in \mathcal{M}(\mathcal{X})$
  - It chooses action $A_1 = \pi(X_1)$ (deterministic policy), and receives a reward of $R_1 \sim \mathcal{R}(\cdot|X_1, A_1)$.

We can talk about average (expected) reward that the agent receives within one episode as the measure of performance.
Average is over repeated interactions with the environment.
If we define the performance in this way, answering the question of how the agent should act to maximize this notion of performance is easy.

Immediate Reward Problem

Let us define expected reward as

$$r(x,a) \triangleq \mathbb{E}\left[R|X=x, A=a\right].$$

In order to maximize the expected reward, the best action depends on the state the agent initially starts with. At state x, it should choose

$$a^* \leftarrow \operatorname*{argmax}_{a\in\mathcal{A}} r(x,a).$$

This is the best, or *optimal*, action at state x.
The optimal policy $\pi^* : \mathcal{X} \to \mathcal{A}$:

$$\pi^*(x) \leftarrow \operatorname*{argmax}_{a\in\mathcal{A}} r(x,a). \tag{1}$$

Optimal policy is only a function of the agent's state x. It does not depend on the initial distribution ρ.

Finite Horizon Tasks

The agent interacts with the environment for a fixed $T \geq 1$ number of steps.

- At each round (episode),
  - The agent starts at $X_1 \sim \boxed{\rho} \in \mathcal{M}(\mathcal{X})$.
  - It chooses action $A_1 = \pi(X_1)$ (or $A_1 \sim \pi(\cdot|X_1)$ for a stochastic policy).
  - The agent goes to the next-state $X_2 \sim \mathcal{P}(\cdot|X_1, A_1)$ and receives reward $R_1 \sim \mathcal{R}(\cdot|X_1, A_1)$.
  - (this process repeats for several steps until ...)
  - The agent gets to the last state $X_T \sim \mathcal{P}(\cdot|X_{T-1}, A_{T-1})$, chooses action $A_T = \pi(X_T)$ (or $A_T \sim \pi(\cdot|X_T)$ for a stochastic policy), and receives $R_T \sim \mathcal{R}(\cdot|X_T, A_T)$.

So we receive a reward sequence $\boxed{(R_1, R_2, \ldots, R_T)}$.
Q: How should we evaluate the performance of the agent as a function of the reward sequence?

Finite Horizon Tasks

A common choice for performance is to compute the sum of rewards:

$$\underline{G^{\pi}} \triangleq R_1 + \ldots + R_T. \tag{2}$$

The r.v. G^{π} is called the return of following policy π.
Here the rewards received at all time steps are treated the same.
The agent just adds them together.

Finite Horizon Tasks

Another choice is to consider *discounted* sum of rewards. Given a discount factor $0 \leq \gamma \leq 1$, we define the return as

$$G^{\pi} \triangleq R_1 + \gamma R_2 + \ldots + \gamma^{T-1} R_T. \tag{3}$$

Whenever $\gamma < 1$, the reward that is received earlier contributes more to the return. Intuitively, this means that such a definition of return values earlier rewards more.

- A cookie today is better than a cookie tomorrow, and a cookie tomorrow is better than a cookie a week later.
- Financial interpretation (inflation rate).
- Marshmallow test (delayed gratification).

Smaller values of γ makes the agent more myopic.

Remark

The discount factor is a part of the problem definition.

From Return to Value Function

The return (3) (and (2) as a special case) is a random variable. To define a performance measure that is not random, we compute its expectation.

If $T = 1$, $V^{\pi}(x) = r^{\pi}(x) = \sum_a \pi(a|x) r(x,a)$

$= r(x, \pi(x))$.

$$V^{\pi}(x) \triangleq \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} R_t | X_1 = x\right].$$

This is the expected value of return if the agent starts at state x and follows policy π. The function $V^{\pi} : \mathcal{X} \to \mathbb{R}$ is called the value function of π.

state-

From Return to Value Function

Let us look at the case of $\underline{T = 1}$ a bit closer. The value function V^π at state x is

$$\underline{V^\pi}(x) = \mathbb{E}\left[\underline{R}_{\underline{1}} | \underline{X = x}\right].$$

This is similar to $\underline{r(x,}\, a) = \mathbb{E}\left[R | X = x, A = a\right]$ with the difference that $r(x,a)$ is conditioned on both x and a, whereas V^π is conditioned on x.

In V^π, the choice of action is determined by the policy π, i.e., at state x, $\underline{a = \pi(x})$ or $\underline{A \sim \pi(\cdot|x})$.

If we define

$$\underline{r^\pi}(x) \triangleq \underline{\mathbb{E}\left[R | X = x\right]}$$

with $\boxed{A \sim \pi(\cdot|x),}$ we get that $\boxed{r^\pi = V^\pi.}$

For $T > 1$, V^π captures the long-term (discounted) average of the rewards, instead of the expected immediate reward captures by r^π.

How to Get the Optimal Policy?

Let us focus on $T = 1$ again.
Recall from before that for the immediate reward maximization problem, the optimal policy was

$= r(x,a)$

$$\pi^*(x) \leftarrow \operatorname*{argmax}_{a \in \mathcal{A}} \mathbb{E}\left[R | X = x, A = a\right].$$

Getting the optimal policy from V^π "seems" less straightforward.

How to Get the Optimal Policy?

We need to search over the space of all deterministic or stochastic policies. If we denote the space of all stochastic policies by

$$\Pi = \{ \pi : \pi(\cdot|x) \in \mathcal{M}(\mathcal{A}), \forall x \in \mathcal{X} \}$$

we need to find

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} V^{\pi}.$$

It turns out that this problem is not too difficult when $T = 1$. As the values of V^{π} at two different states $x_1, x_2 \in \mathcal{X}$ do not have any interaction with each other, we find the optimal policy at each state separately.

How to Get the Optimal Policy?

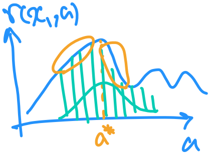
$$r(x,a) \longrightarrow a^* = \arg\max_{a} r(x,a)$$

For each $\underline{x \in \mathcal{X}}$,

$$\underline{V^\pi(x)} = \int \mathcal{R}(\mathrm{d}r|x,a)\pi(\mathrm{d}a|x) = \int \underline{\pi(\mathrm{d}a|x)}\,\underline{r(x,a)}.$$

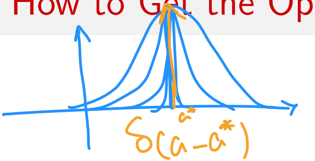
Find a $\underline{\pi(\cdot|x)}$ that maximizes $\underline{V^\pi(x)}$ means that

$$\rightarrow \sup_{\underline{\pi(\cdot|x)}\in\mathcal{M}(\mathcal{A})} \underbrace{\int \pi(\mathrm{d}a|x)r(x,a)}_{= r^\pi(x)}.$$



$$\pi(\cdot|x) = N(a^*, 1) \longrightarrow r^\pi(x) < r(x,a^*).$$

How to Get the Optimal Policy?



$\pi(a|x)r(x,a)da \leftarrow$

$$\Rightarrow \sup_{\pi(\cdot|x)\in\mathcal{M}(\mathcal{A})} \int \boxed{\pi(\mathrm{d}a|x)} r(x,a).$$

The maximizing distribution can concentrate all its mass at the action a^* that maximizes $r(x,a)$ (assuming it exists). Therefore,

$$\Rightarrow \pi^*(a|x) = \delta(a - \boxed{\operatorname*{argmax}_{a'\in\mathcal{A}} r(x,a')}) \; = a^* \quad \Rightarrow \pi^*(x) = a^*$$

(or equivalently, $\pi^*(x) = \operatorname{argmax}_{a'\in\mathcal{A}} r(x,a')$) is an optimal policy at state x. This is for $T = 1$.
For $T > 1$, the problem would be more complicated.

Episodic Tasks

In some scenarios, there is a final time T that the episode ends (or terminates), but it is not fixed a priori.

- Chess
- Finding a goal within a maze
- Robot successfully picks an object

The episode terminates whenever the agent reaches a certain state $\boxed{x_{\text{terminal}}}$ within the state space, i.e., it terminates whenever $\underline{X_T} = \underline{x_{\text{terminal}}}$. The length of the episode T is a random variable.

Episodic Tasks

$T = \infty$ $R_t = +1$

$\gamma < 1$

$\Rightarrow V^{\pi}(x) = 1 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \ldots$

The definition of the return and value functions is as before: For $0 \leq \gamma \leq 1$, we have

$$G^{\pi} \triangleq \sum_{t=1}^{T} \gamma^{t-1} R_t,$$

$$= \frac{1}{1-\gamma}.$$

and

$$V^{\pi}(x) \triangleq \mathbb{E}\left[G^{\pi} | X_1 = x\right].$$

Remark

If $\gamma < 1$, these definitions are always well-defined. If $\gamma = 1$, we need to ensure that the termination time T is finite. Otherwise, the summation might be divergent (just imagine that all R_t are equal to 1).

Continuing Tasks

Sometimes the interaction between the agent and its environment does not break into episodes that terminates. It goes on continually forever.

- Life-long robot
- Chemical plant that is supposed to work for a long time
- An approximate model for finite-horizon problem with very large T.

Continuing Tasks

Consider the sequence of rewards $(R_1, R_2, \dots)$ generated after the agent starts at state $\underline{X_1 = x}$ and follows policy $\underline{\pi}$. Given the discount factor $0 \underline{\leq \gamma < 1}$, the return is

$$G_{\tau}^{\pi} \triangleq \sum_{t \geq \tau} \underline{\gamma^{t-\tau} R_t}. \qquad (4)$$

$$G_1^{\pi} = \sum_{t \geq 1} \gamma^{t-1} R_t = R_1 + \gamma R_2 + \gamma^2 R_3 + \dots$$

Continuing Tasks

Definition (Value Functions)

The (state-)value function V^π and the action-value function Q^π for a policy π are defined as follows: Let $(R_t; t \geq 1)$ be the sequence of rewards when the process is started from a state X_1 (or (X_1, A_1) for the action-value function) drawn from a positive probability distribution over $\mathcal{X}$ (or $\mathcal{X} \times \mathcal{A}$) and follows the policy π ~~for $t \geq 1$ (or $t \geq 2$ for the action-value function)~~. Then,

$$V^\pi(x) \triangleq \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \middle| X_1 = x\right],$$

$A_1 \sim \pi(\cdot|X_1)$
$= \pi(\cdot|x)$

$$Q^\pi(x, a) \triangleq \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \middle| X_1 = x, A_1 = a\right].$$

Continuing Tasks

- The value function V^π evaluated at state x is the expected discounted return of following the policy π from state x.
- The action-value function Q^π evaluated at (x, a) is the expected discounted return when the agent starts at state x, takes action a, and then follows policy π.

Remark

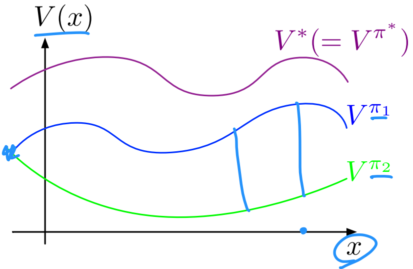
If $\gamma = 0$, $Q^\pi = \mathbb{E}\left[R_1 | X_1 = x, A_1 = a\right]$. This is the same as the expected immediate reward $r(x, a)$. The same way that we could easily compute the optimal action using $r(x, a)$ in the finite-horizon problem with $T = 1$, we can use Q^π (in continual task) in order to easily compute the optimal policy.

Optimal Policy and Optimal Value Function

Q: What does it mean for an agent to act optimally?

Let us compare two (Markov stationary) policies π and π':

- We say that π is better than or equal to π' (i.e., $\pi \geq \pi'$) iff $V^{\pi}(x) \geq V^{\pi'}(x)$ for all states.
- Optimal policy: If we can find a policy π^* that satisfies $\pi^* \geq \pi$ for any π, we call it an optimal policy.
- There may be more than one optimal policy, but their values are the same.



π_1^* & π_2^*

$V_{(x)}^{\pi_1^*} > V_{(x)}^{\pi_2^*}$

Optimal Policy and Optimal Value Function

If we denote Π as the space of all stationary Markov polices, this means that

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} V^\pi,$$

where one of the maximizers is selected in an arbitrary way.

The value function of this policy is the called the optimal (state-)value function, and is denoted by V^{π^*} or simply V^*.

We can also define the optimal policy based on Q^π, i.e.,

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} Q^\pi.$$

The optimal action-value function is denoted by Q^{π^*} or Q^*.

Optimal Policy and Optimal Value Function

For the immediate reward maximization problem (finite horizon with $\underline{T = 1}$), it is easy to find the optimal value function. Recall that $\pi^*(x) \leftarrow \underline{\operatorname{argmax}_{a \in \mathcal{A}}}\, r\underline{(x, a)}$ (1), so for any $x \in \mathcal{X}$,

$$\underline{V^*}(x) = \underline{V^{\pi^*}(x)} = \boxed{\max_{a \in \mathcal{A}} r(x, a).}$$

It is clear that for any $\pi : \mathcal{X} \to \mathcal{A}$,

$$V^*(x) = \underline{\max_{a \in \mathcal{A}} r(x, a)} \geq \underline{r(x, \pi(x))}.$$

The conclusion would be the same for stochastic policies.

Optimal Policy and Optimal Value Function

When we go to continual tasks (or even finite horizon with $T > 1$), we should ask several questions:

- Does any optimal policy exist? Maybe no single policy can dominate all others for all states.
  For example, it is imaginable that at best we can only hope to find a π^* that is better than any other policy π only on a proper subset of $\mathcal{X}$.
- Is the optimal policy necessarily a stationary policy?
  Isn't it possible to have a policy $\bar{\pi} = \{\pi_1, \pi_2, \ldots\}$ that depends on the time step and acts better than any stationary policy $\bar{\pi} = \{\pi, \pi, \ldots\}$?
- More pragmatic question: How can we find an optimal policy (if it exists)?

Optimal Policy and Optimal Value Function

When we go to continual tasks (or even finite horizon with $T > 1$), we can ask several questions:

- (Planning Problem) How can we find an optimal policy (if it exists) given the model $\mathcal{P}$ and $\mathcal{R}$?
- (RL Problem) How we can *learn* π^* (or a close approximation) without actually knowing the MDP, but only have samples coming from interacting with the MDP?

Q-Learning

Before diving into RL algorithms:

- Need to build necessary background
- Sneak peek: Q-Learning

Q-Learning

- Quintessential RL algorithm, introduced by Christopher Watkins [Watkins, 1989, Chapter 7 – Primitive Learning]
- Example of Temporal Difference (TD) learning [Sutton, 1988].

Q-Learning $Q^* \longrightarrow \pi^*(x) = \underset{a}{\arg\max}\, Q(x,a)$

```
Require: Step size $\alpha \in (0, 1]$
Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, except that for $x_{\text{terminal}}$, set $Q(x_{\text{terminal}}, \cdot) = 0$.
for each episode do
    Initialize $X_1 \sim \rho$
    for each step $t$ of episode do
        $A_t \sim \pi(\cdot|X_t)$,
        Take action $A_t$, observe $X_{t+1}$ and $R_t$
        Update $Q(X_t, A_t)$ using the following update rule

$$Q(X_t, A_t) \leftarrow Q(X_t, A_t) + \alpha \left[ R_t + \gamma \max_{a' \in \mathcal{A}} Q(X_{t+1}, a') - Q(X_t, A_t) \right].$$

    end for
end for
```

$\longrightarrow Q \longrightarrow Q^*$

Exploration vs. Exploitation

$\varepsilon = 0 \to$ greedy policy

Variety of policies can be selected as π.
A commonly-used one is ε-greedy policy:

$$A_t = \begin{cases} \operatorname{argmax}_{a \in \mathcal{A}} Q(X_t, a) & \text{w.p. } 1 - \varepsilon \\ \text{uniform}(\mathcal{A}) & \text{w.p. } \varepsilon \end{cases}$$

Usually the value of ε is small and may go to zero as the agent learns more about its environment.

Exploration vs. Exploitation

Another choice: Boltzmann (or softmax or Gibbs) policy:

$$\pi_\tau(a|\overset{x_t}{\cancel{S}};Q) = \frac{\boxed{\exp(Q(\overset{x_t}{\cancel{S}},a)/\tau)}}{\sum_{a'\in\mathcal{A}} \exp(Q(\underset{x_t}{S},a')/\tau)}$$

When the temperature $\tau \to 0$, it selects the action with highest value.

Both are methods to balance exploration vs. exploitation.

Exploration vs. Exploitation

Exploration-Exploitation tradeoff: the agent should collect as much information about the environment as possible (exploration), while benefitting from the knowledge that has been gathered so far in order to obtain a lot of rewards (exploitation).

Examples:

- Restaurant Selection.
  - Exploitation: Go to your favourite restaurant
  - Exploration: Try a new restaurant
- Online Banner Advertisements
  - Exploitation: Show the most successful advertisement
  - Exploration: Show a different advertisement
- Game Playing
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move

Exploration vs. Exploitation

- Without exploration, the agent may never find some good actions. Q: Can you come up with an example?
- Under certain conditions, including how the learning rate α should be selected, the Q-Learning algorithm on finite state-action MDPs can be guaranteed to converge to the optimal action-value function Q^*.
- The ε-greedy is one of the simplest and widely used methods for trading-off exploration and exploitation. Exploration-exploitation tradeoff is an important topic of research.

Logistics

This Course

- Introductory course on reinforcement learning.
- The goal is to help you understand the mathematical foundations of RL.
- What you need to be successful?
  - mathematical maturity (comfortable going through proofs)
  - Probability theory, linear algebra, basics of analysis, statistics, and supervised machine learning
  - Programming skills (Python)
  - Work hard!

Course Information

- Course Website: https://amfarahmand.github.io/IntroRL/
- Main source of information is the course webpage. Check regularly!
- We use Foundations of Reinforcement Learning as the main textbook. I will release chapters are we progress.
- YouTube videos from the previous offering of the course is available. There is no guarantee that they are exactly the same as what we cover this year, but they have enough overlap that you should be able to use them to review the material or keep up if you have to miss a class.
- We will also use Moodle for **announcements & grades & discussions**.

Course Information

- We have two sessions per week.
  - Mondays: Lectures
  - Fridays: Labs
    - Run by own wonderful TAs: Ali Saheb Pasand and Manoosh Samiei
    - Tutorials
    - TA Office Hours
    - Occasional lectures, if we are behind the schedule
    - (Possibly) Guest lectures
- If you require additional academic accommodations, please contact PolyMtl's accessibility services. (I do not know the link or how it exactly works yet.)
- I know that life can be difficult. I try to be as accommodating as possible.

In the Classroom

- Please ask questions. Don't be shy! Asking questions helps you learn better. I try to answer as many questions as I can.
- Actively answer my questions and participate in discussions. This is important for your learning.
- It is OK to use your laptop or tablet during the lectures. It might help your learning to annotate slides as we go through them. Please do not take a call, watch videos, etc. that distract you and others.
- Minimize talking with others in the middle of lecture, as it would be distracting to me and others. We will have breaks so you can discuss among yourselves.
- Recording or taking pictures in class is strictly prohibited without the consent of your instructor. Please ask before doing!

Course Evaluation

- Five (5) assignments (40%)
  - The lowest mark is dropped.
  - Combination of mathematical derivations, proofs, and programming exercises.
- Research Project (30%)
  - Proposal (5%), class presentation (5%), and written report (20%)
- Final Exam (20%)
  - December 5, 1:30PM
- Reading research papers (10%)
  - Short 1-paragraph summary and two questions on how the method(s) can be used or extended.
- Bonus (5%)
  - Finding typos, etc. in the textbook or slides
  - Active class participation, etc.

Collaboration and Assignments

- Collaboration on the Homework Assignments is allowed, under certain conditions:
    - You can discuss the assignment with another student (group of two).
    - You can work on the code and mathematical derivations together.
    - Each of you need to be able to explain the solution. For assignments, you should not follow the divide-and-conquer strategy: person A solving Q1 while person B solving Q2 is not allowed; person A and person B sitting together and solving Q1 and then Q2 is allowed.
    - In your submission, you need to be very clear about the contribution of each individual. For example, you should say we did a pair-programming or person A solved this part while person B solved another part. Person A came up with an about the proof, person B filled the gaps, and person A typed the proof.

Collaboration and Assignments

- You need to form a team of 3–4 members to work on your projects (the exact number will be determined after finalizing the number of students enrolled).
    - Similar to the homework assignments, you need to report the contribution of each collaborator.
    - You can use the help of a machine in writing the code for your project. You can also use it to revise your writing. But you should not delegate writing of the report to an LLM.
    - Instruction about the project will be posted.
- For Reading research papers, you can discuss them with others, but you need to read each paper yourself and write down the summary and research suggestions yourself. Do not delegate reading them or writing a summary to a machine.

Collaboration with the Machines

- LLMs are here and we cannot and do not want to ignore them.
- You do not want to completely rely on LLMs while you are at the learning stage. If you let LLMs do your work, you do not learn as effectively.
- You are discouraged to use LLMs in solving homework assignments. The coding questions are usually simple enough that completing them using LLMs would not give you any learning opportunities. Same for the derivations.
- You should not use LLMs to do the Reading assignments by asking it to read and summarize the paper for you. You can, however, discuss the paper with an LLM. I want you to actually read the paper and write your own summary and ideas.
- In the Projects, you can use LLM to help you in writing the code. But not the report, except in improving its writing, if needed.

References

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, University of Cambride, 1989.