Theory and Methods for Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 1: Introduction to Reinforcement Learning

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-618 (Spring 2020)

License Information for Reinforcement Learning Slides

Logistics

- **Credits:** 3
- **Prerequisites:** Previous coursework in optimization, probability theory, and linear algebra is required. Familiarity with deep learning and programming in python is useful.
- **Grading:** Course project and presentation
- **Moodle:** https://moodle.epfl.ch/course/view.php?id=15887
- **TA's**: Kamalaruban Parameswaran, Paul Rolland, Igor Krawczuk, and Cheng Shi

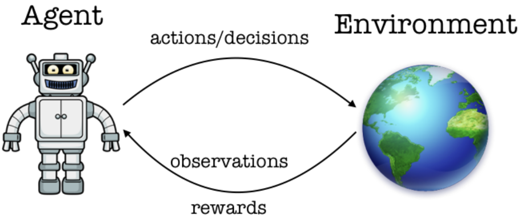
Outline

- This class:
  1. Reinforcement Learning: A basic introduction
  2. Markov Decision Process
- Next class:
  1. Dynamic Programming

Recommended reading

- Chapter 3 in S. Sutton, and G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

What is Reinforcement Learning?



Reinforcement learning is learning what to do — how to map situations to actions — so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics — trial-and-error search and delayed reward — are the two most important distinguishing features of reinforcement learning.

Richard S. Sutton and Andrew G. Barto
Reinforcement Learning: An Introduction, 1998

Applications: Humanoid (walking) Robot

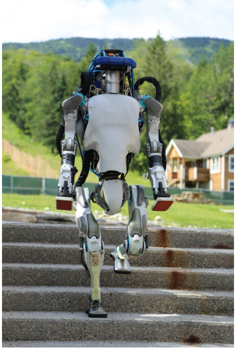


Figure: https://www.bostondynamics.com/atlas

- see in action ▸ Link
- choose the forces applied at the joints based on the current walking position
- reward: $(+)$ for forward motion; $(-)$ for falling over

Applications: Autonomous Helicopter Flight

Figure: http://heli.stanford.edu/

- see in action Link
- choose the speed of the motors based on the current helicopter position
- reward: $(+)$ for following desired trajectory; $(-)$ for crashing

Applications: Playing Games Better Than Humans — Atari 2600

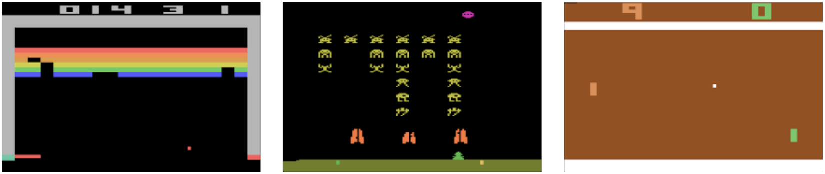


Figure: http://www.cs.toronto.edu/ vmnih/docs/dqn.pdf

- see in action Link
- choose the next move based on the current game position
- reward: $(+)$ for increasing score; $(-)$ for decreasing score

Applications: Playing Games Better Than Humans — AlphaGo

Figure: https://deepmind.com/research/alphago/

- see in action Link
- choose the next move based on the current board position
- reward: $(+)$ for winning the game; $(-)$ for losing the game

Applications: Playing Games Better Than Humans — AlphaStar

Figure: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/

- see in action [Link]
- multi-agent RL

Applications: Portfolio Management

Figure: https://medium.com

- make buy/sell decisions according to market conditions
- reward: $(+)$ for each \$ in the bank

More Applications

- wireless communication [1]
- display ads [2]
- energy management [3]
- chatbots [4]

Reinforcement Learning vs. Regular Machine Learning

- Distinguishing features:
  - lack of a supervisor, only a reward signal
  - delayed feedback
  - non *i.i.d* data
  - actions affect the subsequent observations

Reinforcement Learning vs. Imitation Learning

Reward hypothesis

Preferred behavior/goal can be described by the maximization of expected cumulative reward.

- Reinforcement Learning:
  - receives a feedback signal: reward
  - learning from interaction
- Imitation Learning:
  - no feedback signal available
  - learning from (expert) demonstrations
  - behavioral cloning and inverse reinforcement learning

Motivation

Key question

How do we model the reinforcement learning problem?

Markov Property

Definition (Markov property)

Consider a sequence of random variables $S_1, S_2, \ldots, S_t, \ldots$ (called as states). A state S_t is *Markov* if and only if

$$P(S_{t+1} \mid S_t) \;=\; P(S_{t+1} \mid S_1, \ldots S_t)$$

- future $\perp\!\!\!\perp$ past | present
- the state summarizes the "past (history)" so as to retain all "essential" information
- once the state is known, the history may be thrown away
- the state is a sufficient statistic of the history

Markov Process

- Sequence of random states with Markov property
- Elements of a Markov Process:

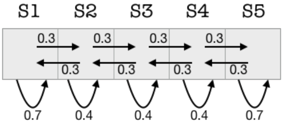
$\mathcal{S}$ state space, S_t represents the state at time t

P state transition probability, $S_{t+1} \sim P(\cdot \mid S_t)$

P_0 initial state distribution, $S_0 \sim P_0(\cdot)$

Markov Process

- Pong 1D: https://mathsimulationtechnology.wordpress.com/pong/
- Example:



  - $\mathcal{S} = \{S1, S2, S3, S4, S5\}$
  - $P(S1 \mid S2) = P(S3 \mid S2) = 0.3, \; P(S2 \mid S2) = 0.4$
  - sample episode: $S2, S1, S1, S2, S3, S4, S3, S4, S5, \ldots$

Markov Reward Process (MRP)

- MRP = Markov Process + Reward
- Elements of a Markov Reward Process:

$\mathcal{S}$ state space, S_t represents the state at time t

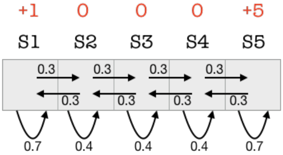
R reward function, $R_t = R(S_t) \in \mathcal{R} \subseteq \mathbb{R}$

P state transition probability, $S_{t+1} \sim P(\cdot \mid S_t)$

P_0 initial state distribution, $S_0 \sim P_0(\cdot)$

Markov Reward Process (MRP)

- Example:



  - $\mathcal{S} = \{S1, S2, S3, S4, S5\}$
  - $R(S1) = +1,\ R(S2) = R(S3) = R(S4) = 0,\ R(S5) = +5$
  - $P(S1 \mid S2) = P(S3 \mid S2) = 0.3,\ P(S2 \mid S2) = 0.4$
  - sample episode: $S2, S1, S1, S2, S3, S4, S3, S4, S5, \ldots$
    $(0, +1, +1, 0, 0, 0, 0, 0, +5, \ldots)$

Markov Decision Process (MDP)

- MDP = MRP + Action
- Elements of a Markov Decision Process:

  $\mathcal{S}$ state space, S_t represents the state at time t

  $\mathcal{A}$ action space, A_t represents the action taken at time t

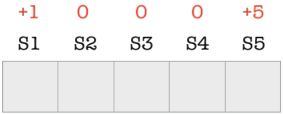
  R reward function, $R_{t+1} = R(S_t, A_t, S_{t+1}) \in \mathcal{R} \subseteq \mathbb{R}$

  P state transition probability, $S_{t+1} \sim P(\cdot \mid S_t, A_t)$

  P_0 initial state distribution, $S_0 \sim P_0(\cdot)$

Markov Decision Process (MDP)

- Example:



  - $\mathcal{S} = \{S1, S2, S3, S4, S5\}$
  - $\mathcal{A} = \{\text{left}, \text{right}\}$
  - $R(S2, \text{left}, S1) = +1,\ R(S3, \text{left}, S2) = 0,\ R(S4, \text{right}, S5) = +5$
  - deterministic dynamics: $P(S1 \mid S2, \text{left}) = P(S1 \mid S1, \text{left}) = 1$, $P(S3 \mid S2, \text{right}) = 1$
  - sample episode: $S2, \text{left}, S1, {\color{red}+1}, \text{left}, S1, {\color{red}+1}, \text{right}, S2, {\color{red}0}, \ldots$

Policy

Definition (Policy)

A policy π is a distribution over actions given states,

$$\pi(a \mid s) \;=\; P(A_t = a \mid S_t = s)$$

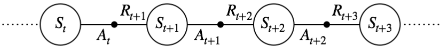
- a policy specifies what action to take in each state
- MDP policies depend on the current state (not the history)
- policies are stationary (time-independent), $A_t \sim \pi(\cdot \mid S_t), \forall t > 0$
- example: $\pi(\text{left} \mid s) = 1, \forall s \in \mathcal{S}$, a policy that always chooses left for all states.

Reinforcement Learning Game

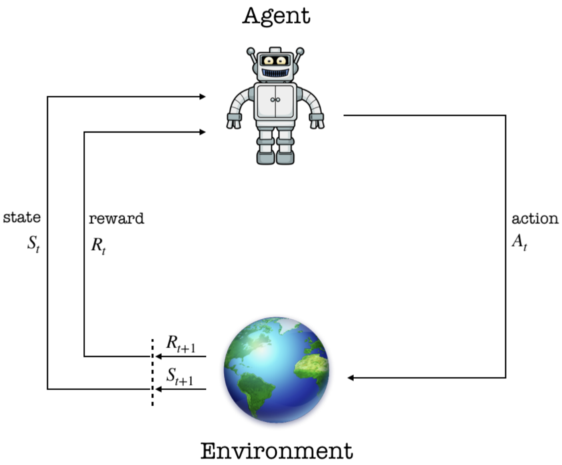
- Players:
  - environment: MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, P_0)$
  - agent: deterministic or random policy $\pi(a \mid s)$

- At time step $t = 0$: $S_0 \sim P_0(\cdot)$
- At each time step $t = 1, 2, \ldots$
  - agent receives some representation of the environment's state $S_t \in \mathcal{S}$
  - agent chooses an action $A_t \in \mathcal{A}(S_t)$ based on S_t or $(S_{0:t}, A_{0:t-1})$
  - agent receives a reward $R_{t+1} \in \mathcal{R} \subseteq \mathbb{R}$, and finds itself in a new state S_{t+1}

- Trajectory:



Agent Environment Interaction



Dynamics of the Environment

- Probability of the next state and reward given the current state and action:

$$p(s', r \mid s, a) \;:=\; P(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a)$$

State transition probability

$$p(s' \mid s, a) \;:=\; P(S_t = s' \mid S_{t-1} = s, A_{t-1} = a) \;=\; \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

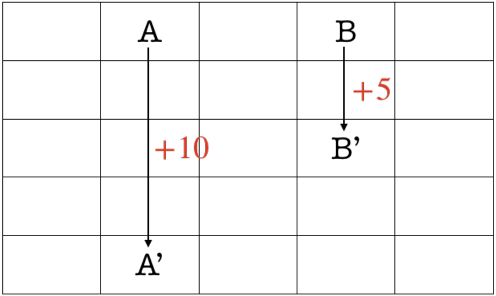
Expected immediate reward

$$r(s, a) \;:=\; \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] \;=\; \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

$$r(s, a, s') \;:=\; \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] \;=\; \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

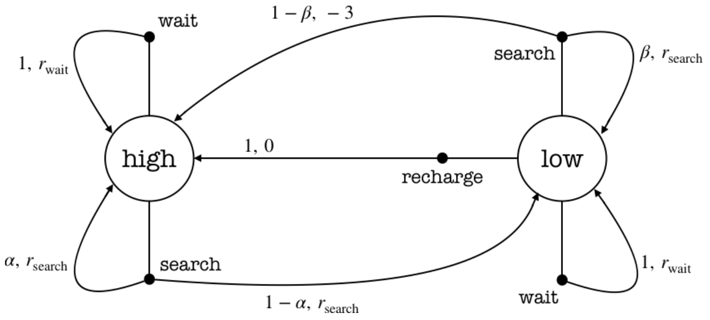
Example: Gridworld

- Problem specification:
  - state space: $\mathcal{S} = \{\text{cells in the grid}\}$; $|\mathcal{S}| = 25$
  - action space: $\mathcal{A} = \{\text{north}, \text{south}, \text{east}, \text{west}\}$; $|\mathcal{A}| = 4$
  - dynamics: deterministic
  - if the action takes the agent off the grid: no move, but reward -1
  - from state A, all four actions yield a reward of $+10$ and take the agent to A'
  - from state B, all four actions yield a reward of $+5$ and take the agent to B'
  - other actions result in a reward of 0



Example: Recycling Robot

- Problem specification:
  - state space: $\mathcal{S} = \{\text{high}, \text{low}\}$; $|\mathcal{S}| = 2$
  - action space: $\mathcal{A}(\text{high}) = \{\text{search}, \text{wait}\}$, and $\mathcal{A}(\text{low}) = \{\text{search}, \text{wait}, \text{recharge}\}$
  - rewards: r_{search} = expected number of cans while searching, and r_{wait} = expected number of cans while waiting ($r_{\text{search}} > r_{\text{wait}}$).



Returns

- Episodic tasks:
  - interaction breaks naturally into episodes
  - example: plays of a game, trip through a maze
  - terminal state
  - non-terminal states $\mathcal{S}$
  - set of all states plus the terminal state $\mathcal{S}^+$
  - return $G_t := R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$, where at time step T terminal state is reached

Returns

- Continuing tasks:
  - interaction does not have natural episodes, but just goes on and on
  - example: controlling a power plant
  - discount rate $\gamma \in [0, 1]$
  - humans prefer $\gamma < 1$, *e.g.*, financial returns
  - $\gamma = 0$: only care about immediate reward
  - $\gamma = 1$: future reward is beneficial as immediate reward
  - discounted return $G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
  - if $R_t \in [0, 1]$, then $G_t \leq \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$

Returns

- Unified notation (for both episodic and continuing tasks):
  - absorbing state for episodic tasks
  - return $G_t := \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
  - $T = \infty$ or $\gamma = 1$ (but not both)

Recursive relationship

$$
\begin{aligned}
G_t &:= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\
&= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\
&= R_{t+1} + \gamma G_{t+1}.
\end{aligned}
$$

Planning and Learning

- Markov Decision Process:
  - fully observable state and reward
  - known reward distribution and transition probabilities
  - A_t function of $(S_{1:t}, A_{1:t-1}, R_{1:t})$

- Reinforcement learning:
  - observable state and reward
  - unknown reward distribution
  - unknown transition probabilities
  - A_t function of $(S_{1:t}, A_{1:t-1}, R_{1:t})$

Resources

- https://github.com/ShangtongZhang/reinforcement-learning-an-introduction
- https://github.com/openai/spinningup
- https://github.com/openai/baselines

References

[1] Timothy X Brown.
Low power wireless communication via reinforcement learning.
In *Advances in Neural Information Processing Systems*, pages 893–899, 2000.

[2] Han Cai, Kan Ren, Weinan Zhang, Kleanthis Malialis, Jun Wang, Yong Yu, and Defeng Guo.
Real-time bidding by reinforcement learning in display advertising.
In *Proceedings of International Conference on Web Search and Data Mining*, pages 661–670. ACM, 2017.

[3] Sunyong Kim and Hyuk Lim.
Reinforcement learning based energy management algorithm for smart energy buildings.
Energies, 11(8):2010, 2018.

[4] Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al.
A deep reinforcement learning chatbot.
arXiv preprint arXiv:1709.02349, 2017.