Reinforcement Learning

—

Exercise 2: Value Functions, Dynamic Programming and Optimal Policies

Alexander Mattick

Fraunhofer-Institut für Integrierte Schaltungen IIS

Overview

Exercise Content

Week	*Date*	*Topic*	*Material*	*Who?*
0			*no exercises*	
1	23.04.	MDPs		Nico
2	30.04.	Dynamic Programming		Alex
3	07.05.	OpenAI Gym, PyTorch-Intro		Alex
4	14.05.	TD-Learning		Nico
5	22.05.	Practical Session (zoom@home)	**Attention: Lecture Slot!**	Nico + Alex
6	28.05.	TD-Control		Nico
7	04.06.	DQN		Nico
8	11.06.	VPG		Alex
9	18.06.	A2C		Nico
10	25.06.	Multi-armed Bandits		Alex
11	02.07.	RND/ICM		Alex
12	09.07.	MCTS		Alex
13	16.07.	BCQ		Nico

Fraunhofer IIS

Dynamic Programming

Markov Decision Processes
Recap

- We need a controller that helps us select the actions to maximize expected cumulative reward
  - So-called: **Expected return** or **value**

- A policy π represents this controller:
  - π determines the agent's behavior, i.e., its way of acting
  - π is a mapping from state space $\mathcal{S}$ to action space $\mathcal{A}$, i.e., $\pi : \mathcal{S} \mapsto \mathcal{A}$
  - Two types of policies:
    - Deterministic policy: $a = \pi(s)$.
    - Stochastic policy: $\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$

- **Goal**: find a policy that maximizes the expected return!
  - We denote the optimal policy π for a given MDP as π^*

Markov Decision Processes

The Value Function

(State-)Value function

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s\,] = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s\right]$$

- "Expected return following policy π from state s"

Action-value function/Q-function

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s, a_t = a\right]$$

- "Expected return of doing action a in state s and following policy π afterwards"

Dynamic Programming
Introduction

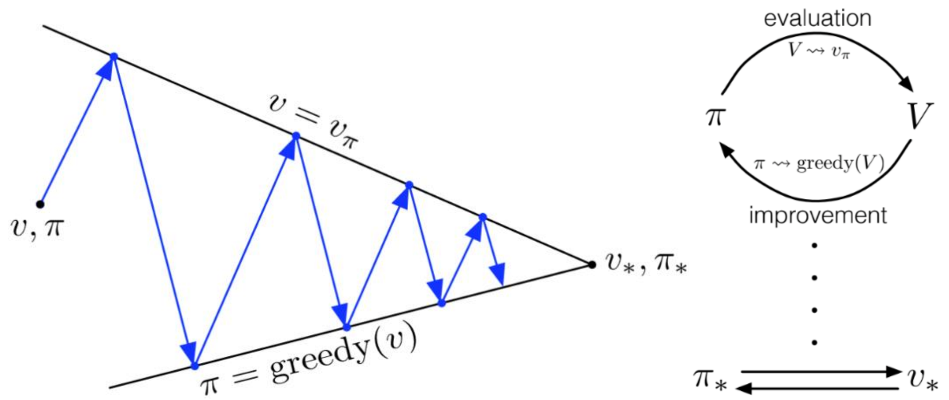
- Limited utility in practical reinforcement learning, but theoretical importance
  - Why?
- **Idea:** Use value functions to organize and structure the search for good policies
  - We can easily obtain optimal policies once we have found the optimal value function (and vice versa)
  - Founded on the Bellman optimality equation(s)

Bellman-Optimality Equation

$$V_{\pi^*}(s) = \max_a Q_{\pi^*}(s,a) = \max_a \mathbb{E}_{\pi^*}[r_t + \gamma V_{\pi^*}(s)]$$

Four key concepts

- Policy Evaluation
- Policy Improvement
- (Generalized) Policy Iteration
- Value Iteration



Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Fraunhofer IIS

Policy Evaluation

- Given a policy π and the environment dynamics, we can easily compute the value of state:

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s] = \ \dots = \sum_a \pi(a|s) \sum_{s'\,r} P(s', r \mid s, a)[r + \gamma V_\pi(s')]$$

- System of $\#states$ linear equations with $\#states$ unknowns
  - Can be solved straightforwardly
  - For our purposes, we solve it iteratively

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$ arbitrarily, for $s \in \mathcal{S}$, and $V(terminal)$ to 0

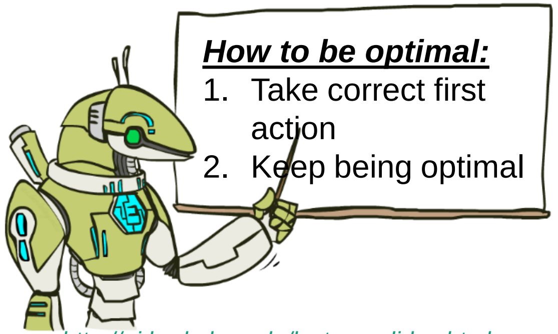
Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a) \left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Policy Improvement

- Given a policy π and its value function (and the environement dynamics), greedily take the action that looks good in the short term

$$\pi'(s) = arg\max_a Q_\pi\,(s,a)$$

- Suppose $\pi' = \pi$, then π' fullfils the Bellman optimality equation in all states
  - Therefore: We found the optimal policy



http://ai.berkeley.edu/lecture_slides.html

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$; $V(\mathit{terminal}) \doteq 0$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop for each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s') \big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s') \big]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration

- **Drawback of Policy Iteration:** We must do a full Policy Evaluation procedure for every step, which is costly!
- We can also truncate this:
  - If we stop the policy evaluation after just one sweep, this is called **Value Iteration**
  - Surprisingly, this corresponds to translating the Bellman optimality equation into an update rule
  - We can also drop the policy improvement step because we are only interested in the final policy
  - Downside of this: More iteration to convergence needed

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

Dynamic Programming

Summary

- Policy Evaluation
  - Given policy π, compute its (approximate) value function for (part of) the state space
- Policy Improvement
  - Given value function $V_\pi(s)$, extract the greedy policy $\pi'$with $V_{\pi'}(s) \geq V_\pi(s)$
- (Generalized) Policy Iteration
  - Repeat until convergence (policy doesn't change after improvement, i.e., Bellman optimality equation holds)
    - Do x steps of Policy Evaluation
    - Do Policy Improvement
- Value Iteration
  - Special case of Policy Iteration with 1 Policy Evaluation step
  - Converged when change in value estimates smaller than some threshold
  - Policy Improvement step only as the last step

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

Thank you for your attention!