Introduction to Reinforcement learning

Moritz Wolter

March 18, 2025

The High-Performance Computing and Analytics Lab, Bonn University

Overview

Motivation

The Q-learning algorithm

Neural Q-Function approximations

Training and implementation

Motivation

Motivation

Thus far we have considered supervised learning problems and that's great, but

- how do we deal with situations where labels are unavailable?
- Animals and humans do not need to be told what to do all the time.
- Reinforcement learning (RL) aims to use environment-feedback.

Reinforcement learning examples

This lecture and the exercises deal with games. It is important to note, that RL powers robots, too [KK99; Wu+23].

Play video[1]

Currently companies are working on digital assistants, which are trained on humand feedback via RL [Bai+22].

[1] https://www.youtube.com/watch?v=xAXvfVTgqr0

The Frozen Lake

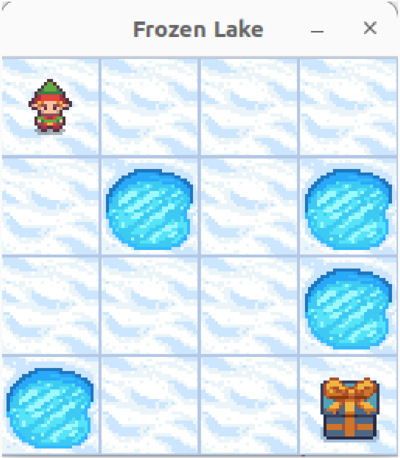

Figure: The frozen-lake setting is a standard educational reinforcement learning problem [Fou24].

Frozen Lake Rewards and Actions

The elf wants to reach the present. The only feedback we get is when we reach the present. More formally the reward is structured as [Fou24],

- Reach goal: $+1$
- Reach hole: 0
- Reach frozen: 0

To reach the present the agent is allowed to sample from the action space, which is

- 0: Move left
- 1: Move down
- 2: Move right
- 3: Move up

in this case [Fou24].

The RL problem

How do we deal with the fact, that we might have to perform many many actions until we potentially see a reward?

The Q-learning algorithm

The Q-Table idea

At every state (tile of the lake), we can think of the expected reward of taking an action. For example, if we stand next two a hole moving into it yields zero reward. The result is a table or function $Q(s, a) \in \mathbb{R}^{N_s, N_a}$ which returns a reward estimate for all states and actions. Here N_s measures the total number of states and N_a the number of possible actions.

The Q-learning algorithm [Mni+13]

Our agent lives in an environment $\mathcal{E}$, where it is allowed to perform actions a_t from the set of allowed actions $\mathcal{A} = \{1, \dots, \mathcal{K}\}$. The agent has access to an input x_t, which contains environment information. We record the sequence of actions and states, $s_t = x_1, a_1, x_2, \dots, a_{t-1}, x_t$. We denote the reward at time t as r_t. We expect to see a reward only at the end of a sequence $t = T$.

The Q-learning algorithm

$$Q(s_t, a_t)_{\text{update}} = Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a_m}(Q(s_{t+1}, a_m)) - Q(s_t, a_t)) \tag{1}$$

Here $Q \in \mathbb{R}^{N_s, N_a}$ denotes the Q-Table with N_s the number of states and N_a the number of possible actions. Reward at time t r_t learning rate α as well as the discount factor γ.

Q-learning in the frozen lake case

$$\begin{bmatrix} [0.,0.,0.,0.] & [0.,0.,0.,0.] & [0.,0.,0.,0.] & [0.,0.,0.,0.] \\ [0.,0.,0.,0.] & [0.,0.,0.,0.] & [0.,0.,0.,0.] & [0.,0.,0.,0.] \\ [0.,0.,0.,0.] & [0.,0.,0.,0.] & [0.,0.,0.,0.] & [0.,0.,0.,0.] \\ [0.,0.,0.,0.] & [0.,0.,0.,0.] & [0.,0.,0.9,0.] & [0.,0.,0.,0.] \end{bmatrix} \tag{2}$$

Q-learning in the frozen lake case

This process continues until a path is established. After 1000 episodes the table looks as follows

$$\begin{bmatrix} [0., 0.59049, 0., 0.] & [0., 0., 0., 0.] & [0., 0., 0., 0.] & [0., 0., 0., 0.] \\ [0., 0.6561, 0., 0.] & [0., 0., 0., 0.] & [0., 0., 0., 0.] & [0., 0., 0., 0.] \\ [0., 0., 0.729, 0.] & [0., 0.81, 0., 0.] & [0., 0., 0., 0.] & [0., 0., 0., 0.] \\ [0., 0., 0., 0.] & [0., 0., 0.9, 0.] & [0., 0., 1., 0.] & [0., 0., 0., 0.] \end{bmatrix}. \tag{3}$$

Action order: [Move left, Move down, Move right, Move up]

Deviating from $\max_a Q(s, a)$

The previous approach yielded only a single path. To explore more states we can choose to deviate from $\max_a Q(s, a)$ with a probability ϵ.

Deviating from $\max_a Q(s, a)$

$\epsilon = 0.2$, and 5k steps lead to

$$\begin{bmatrix} [0.53, 0.59, 0.59, 0.] & [0.53, 0., 0.6, 0.] & [0.59, 0.73, 0.59, 0.] & [0.66, 0., 0., 0.] \\ [0.59, 0.66, 0., 0.] & [0., 0., 0., 0.] & [0., 0.81, 0., 0.] & [0., 0., 0., 0.] \\ [0.66, 0., 0.73, 0.] & [0.66, 0.81, 0.81, 0.] & [0.729, 0.9, 0., 0.] & [0., 0., 0., 0.] \\ [0., 0., 0., 0.] & [0., 0.81, 0.9, 0.] & [0.81, 0.9, 1., 0.] & [0., 0., 0., 0.] \end{bmatrix}. \tag{4}$$

Neural Q-Function approximations

What if the state space is too large to explore?

- The number of possible states in games, but also in the real world grows very quickly.
- What if we fail to set up a complete table?
- How do we approximate missing table entries?

Neural approximation of the Q-Table [Mni+13]

Key Idea: Use a neural network $Q(s, a; \theta) \approx Q(s, a)$.

$$L(\theta) = \frac{1}{N_a} \sum_{i=1}^{N_a} (y_i - Q_n(s, a; \theta)_i)^2 \tag{5}$$

with $Q_n(s, a; \theta) \approx Q(s, a)$ the neural Q-Table approximator. And $\mathbf{y}$ the desired output at the current optimization step. Construct $\mathbf{y} \in \mathbb{R}^{3,3}$ by inserting

$$y_r = \begin{cases} r, & \text{if the game ended} \\ r + \gamma \max_a Q(s_{t+1,a;\theta}) & \text{else} \end{cases} \tag{6}$$

into $\mathbf{y}$ at the position of the move taken.

Training and implementation

Training TicTacToe Game-Agents

x	x	
	o	
o		

Table: TicTacToe board of an ongoing game. Player 1 (x) moves next. How should the distribution of future reward look like?

Training neural Q-Table approximators

Fully connected ReLU-layers,

$$\mathbf{h} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{7}$$

are a possible network architecture building block. Three layers per example are enough for an agent to play TicTacToe reasonably well.

Convergence and average reward

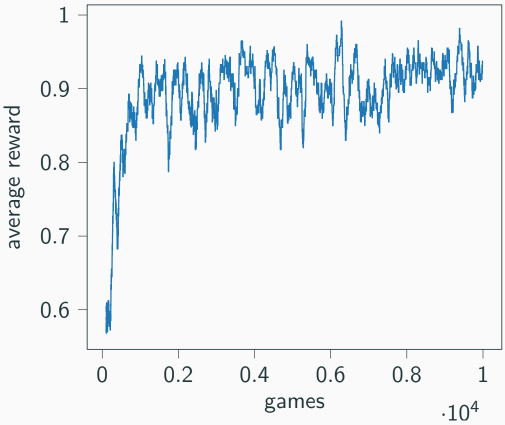

Figure: Average reward over 100 games for a neural TicTacToe Q-agent playing against a random opponent.

Conclusion

We have seen the core ideas required to build your own table and neural-network-powered agents.

Don't forget to play against your agent after finishing today's exercise.

References

[Bai+22] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. **"Training a helpful and harmless assistant with reinforcement learning from human feedback."** In: *arXiv preprint arXiv:2204.05862* (2022).

Literature ii

[Fou24] Farama Foundation. ***Frozen Lake.*** `https://gymnasium.farama.org/environments/toy_text/frozen_lake/`. [Online; accessed 08-March-2024]. 2024.

[KK99] Hajime Kimura and Shigenobu Kobayashi. **"Reinforcement learning using stochastic gradient algorithm and its application to robots."** In: *IEEJ Transactions on Electronics, Information and Systems* 119.8-9 (1999), pp. 931–934.

[Mni+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. **"Playing atari with deep reinforcement learning."** In: *arXiv preprint arXiv:1312.5602* (2013).

[Wu+23] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. **"Daydreamer: World models for physical robot learning."** In: *Conference on Robot Learning*. PMLR. 2023, pp. 2226–2240.