Advanced Python for Neuroscientists
Lecture 8: Reinforcement learning

Summer 2022

Princeton Neuroscience Institute
Instructors: Yisi Zhang & Bichan Wu

July 21, 2022

Recap

Lecture 5: Neural network I

- Feedforward neural network
- Gradient descent

Lecture 6: Neural network II

- Backpropagation
- Stochastic Gradient Descent
- Application

Lecture 7: Convonlutional neural network

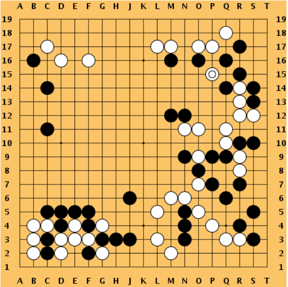
- Motivation & concept
- Overview
- Architectures

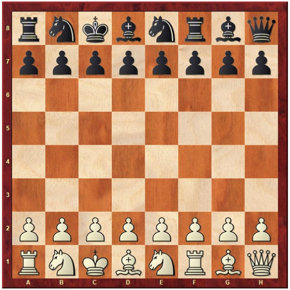
Outline

- Motivation & concept
- Formalization
- Deep Q-learning

8.1 Motivation & concept

How do you train a machine to play games?





How to train this model using supervised learning?
In this case, what is the input and what is the output?

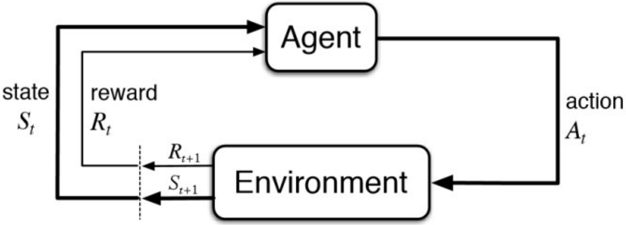
8.1 Motivation & concept

Issues with SL:
- Too many possible solutions (ground truth not well defined)
- Cost function not well defined
- Cannot generalize

Why RL?
- Learn from interaction
- Does not require a knowledgeable external supervisor
- Make sequence of decisions

8.1 Motivation & concept

Reinforcement learning

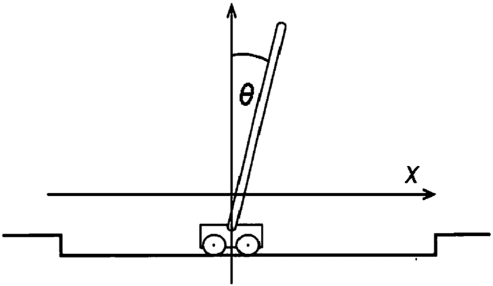


An **agent** interacting with an **environment**, which provides numeric **reward**.

Goal: Learn how to take actions to maximize reward

8.1 Motivation & concept

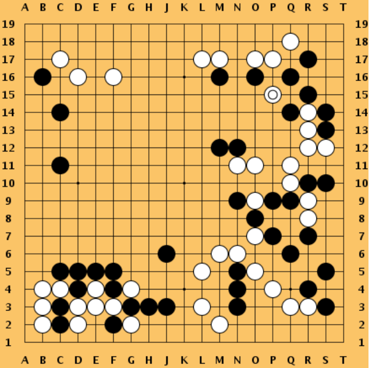
Cart-pole problem



Objective: Balance a pole on top of a moving cart
State: Angle, angular speed, position, horizontal velocity
Action: Horizontal force applied to the cart
Reward: 1 at each time step if the pole is upright

8.1 Motivation & concept

Go



Objective: Win the game
State: Position of all pieces
Action: Where to put the next piece
Reward: 1 if win

8.2 Formalization

Markov decision process

- Markov: the environment's response at $t+1$ depends only on the state and action representations at t

Transition probability

$$\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

Expected reward

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$$

8.2 Formalization

Markov decision process

- At time step t = 0, environment samples initial state $s_0 \sim p(s_0)$
- Then, for t = 0 until done:
  - Agent selects action a_t
  - Environment samples reward $r_t \sim R(.|s_t, a_t)$
  - Environment samples next state $s_{t+1} \sim P(.|s_t, a_t)$
  - Agent receives reward r_t and next states s_{t+1}

- A policy π is a function from S to A that specifies what action to take in each state
- Objective: find policy π^* that maximizes the expected return

8.2 Formalization

Expected return

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_k \gamma^k r_{t+k+1}$$

where $0 \le \gamma \le 1$, called discount rate.

8.2 Formalization

Example: Grid world

states

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

actions = {

1. right $\rightarrow$
2. left $\leftarrow$
3. up $\uparrow$
4. down $\downarrow$

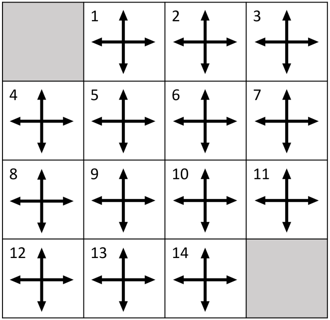
}

Objective: reach the terminal grids in least number of actions

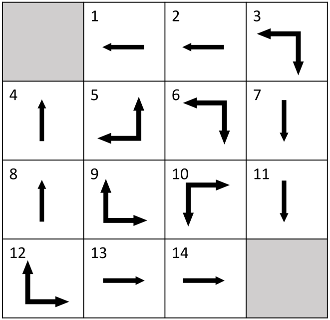
8.2 Formalization

Example: Grid world

Random policy



Optimal policy



8.2 Formalization

The optimal policy π^*

We want to find the optimal policy to maximize the expected return (sum of rewards).

$$\pi^* = \operatorname*{argmax}_{\pi} \mathbb{E}[\sum_{t \geq 0} \gamma^t r_t | \pi]$$

with $s_0 \sim p(s_0)$, $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim p(\cdot|s_t, a_t)$

8.2 Formalization

Value functions

To evaluate how good a state is:
State-value function for policy π

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

8.2 Formalization

Value functions

To evaluate how good an action is in state s:
Action-value function for policy π

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

8.2 Formalization

Bellman equation

The optimal Q-value function Q^* is one which yields maximum expected return achievable from (s, a):

$$Q^*(s,a) = E\{r_{t+1} + \gamma \max_{a'} Q^*(s', a')|s_t = s, a_t = a\}$$

8.2 Formalization

Q-learning

Temporal difference (TD)

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha\underbrace{(R(s,a) + \gamma \max_{a'} Q(s',a') - Q_{t-1}(s,a))}_{\text{TD}}$$

8.2 Formalization

Grid world: find the shortest path to terminals (coding time)

Q-table

	up	down	left	right
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

8.3 Deep Q-learning

Use a function approximator to estimate the action-value function.

$$Q(s, a; \theta) \approx Q^*(s, a)$$

If the approximator is a deep neural network, it is called Deep Q-learning

8.3 Deep Q-learning

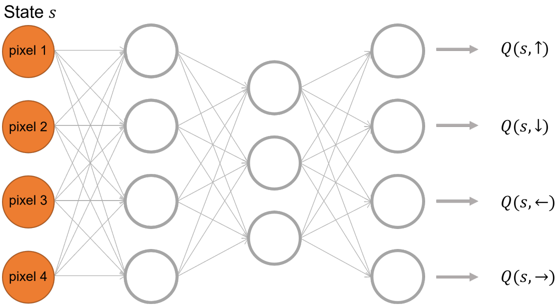
We want to find
$Q^*(s,a) = E\{r + \gamma \max_{a'} Q^*(s', a')|s, a\}$

Forward pass
Cost function $J_i(\theta_i) = [y_i - Q(s,a;\theta_i)]^2$
where $y_i = E\{r + \gamma \max_{a'} Q^*(s', a';\theta_{i-1})|s, a\}$

Backward pass
Gradient descent with respect to Q-function parameters θ
$\nabla_{\theta_i} J_i(\theta_i) =$
$(E\{r + \gamma \max_{a'} Q^*(s', a';\theta_{i-1})\} - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)$

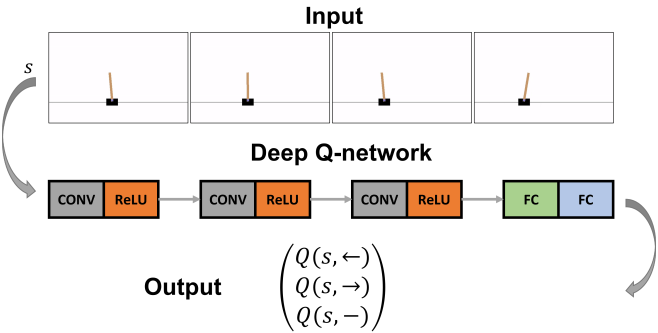
8.3 Deep Q-learning

Implement neural network



8.3 Deep Q-learning

Example: cartpole (coding time)



Homework

- Make sure you understand all the exercises above
- Run through the codes here that should replicate all the figures
  https://github.com/yisiszhang/AdvancedPython/blob/main/colab/Lecture8.ipynb
- Try to improve the Q-learning code