Reinforcement Learning

Johanni Brea

Introduction to Machine Learning

EPFL BIO322 2021

Examples of Reinforcement Learning

Examples: Theorem Proving

input: mathematical axioms and theorems (goals)

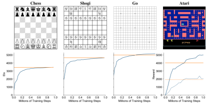
cnf(sos01,axiom, (product(A,A) = A)).
cnf(sos02,axiom, (product(A,product(B,C)) = product(product(A,B),product(
...
cnf(goals,negated_conjecture, (product(product(product(x0,x1),x1),product(x
product(product(x0,x1),product(product(x1,x0),x2)))).

desired output: steps to prove the theorem
learning task: learn solving strategies by trial-and-e
axioms and theorems.

`http://www.tptp.org/`, `https://deepmind.com/resea`
`Training-a-First-Order-Theorem-Prover-from-`

Examples: Games

input: rules of a game



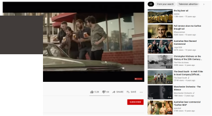
desired output: winning policy

learning task: learn winning strategies by trial-and-error from many games of the computer playing against itself.

`https://arxiv.org/abs/1911.08265`

Examples: Advertisement

input: user profile (currently viewed page & history)

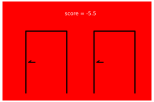


d **output**: attractive suggestions
ı by trial-and-error to display the suggestions
ɪrs will select with high probability.

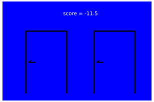
Learning by Trial-and-Error

Table of Contents

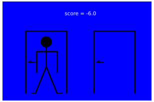
A Toy Problem: Chasse au Trésor



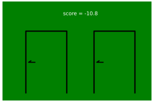
state 1



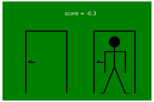
state 2



state 3



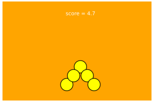
state 4



state 5

actions: (in each room)
open left door,
open right door

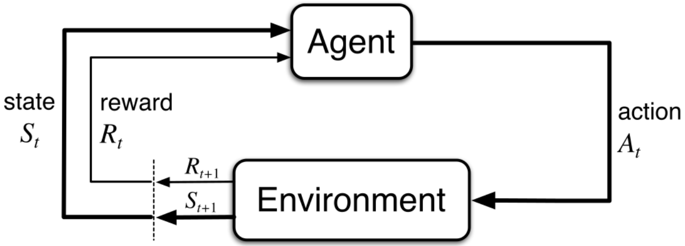
rewards: (depend on
state and action)
between -5 and 6



state 6

state 7

Agents and Environments: States, Actions, Rewards



The **agent** (a person, an animal, a computer program) observes at time t state S_t, reward R_t (for previous actions) and takes action A_t.

The **environment** (a game, a dynamical system, "the world") receives A_t and produces next state S_{t+1} and reward R_{t+1}.

- The dynamics of the environment can be stochastic, e.g. given by **transition probabilities** $P(S_{t+1}|S_t, A_t)$ and **reward probabilities** $P(R_{t+1}|S_t, A_t, S_{t+1})$.
- Usually, the agent starts with zero knowledge about the transition and reward probabilities.
- The agents goal is to maximize the expected cumulative reward.

Action Values

Action Values $Q(S_t, A_t)$ indicate the desirability of action A_t in state S_t by measuring the expected cumulative reward. They are also called **Q Values**.

Finite Horizon until T (episodic setting)

$$Q(S_t, A_t) = \mathrm{E}\left[R_{t+1} + R_{t+2} + \cdots + R_T\right]$$

Infinite Horizon with Discount Factor $\gamma \in [0, 1)$ (continual setting)

$$Q(S_t, A_t) = \mathrm{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots\right]$$

The expectation depends on the policy (action selection strategy) and the transition and reward probabilities. Because they are usually unknown to the RL agent, the agent cannot exactly compute these expectation but has to estimate them.

Learning Action Values with Monte Carlo Estimation

Compute the cumulative reward for every state-action visited in each episode (length T).
Average the result over all episodes where the same state-action pair was visited.

$Q(s, a)$: arbitrarily initialized
`CumulativeRewards`(s, a): an empty list
for all episodes **do**
    $G \leftarrow 0$
    for $t = T - 1, T - 2, \ldots, 1$ **do**
        $G \leftarrow G + R_{t+1}$
        append G to `CumulativeRewards`(S_t, A_t)
        $Q(S_t, A_t)$ = average(`CumulativeRewards`(S_t, A_t))
    end for
end for
return Q

- The estimated Q-values depend on the actions we take!
- One could also traverse the episode in forward order and compute $G = \sum_{i=t+1}^{T} R_i$ for each state-action pair.
- This algorithm can be further optimized by using a recursive update of the average (see notebook).

Policies: Exploration and Exploitation

- A **policy** is a mapping from perceived states s to actions a to be taken when in those states.
- A policy can be a **deterministic** function $a = \pi(s)$.
- In general a policy is **stochastic** and described by conditional probabilities $\pi = P(a|s)$.
- The policy serves two goals:
  1. **Exploitation**: Choose the (supposedly) best action to maximize the cumulative return.
  2. **Exploration**: Choose exploratory actions to learn more about the environment and know better which action is actually best.

  This is also called the **Exploration-Exploitation Dilemma** (or Trade-Off)

ϵ-Greedy Policies

- **ϵ-Greedy Policies** choose with probability $1 - \epsilon$ the (supposedly) best action for the current state (according to e.g. the current estimate of the Q-values) and with probability ϵ a random action.
- The best action is also called **greedy**; the random action **exploratory**.
- With $\epsilon = 1$ the agent only explores.
- With $\epsilon = 0$ the agent only exploits. This is the **greedy policy**.
- ϵ is a critical hyper-parameter affecting speed of learning! Common choices for simple problems are $\epsilon = 0.1$ at the beginning of learning.
- Over the course of learning, the agent gets more and more certain about the best actions; therefore one can gradually decrease ϵ, i.e. exploit more.

Quiz

1. Suppose an agent has experienced the two episodes $((S_1 = s_1, A_1 = a_2, R_1 = 0), (S_2 = s_5, A_2 = a_3, R_2 = 2))$ and $((S_1 = s_1, A_1 = a_2, R_1 = 1), (S_2 = s_4, A_2 = a_2, R_2 = 1))$. Action values are learnt with Monte Carlo Estimation. Which of the following Q-values is correct?
   A $Q(s_1, a_2) = 2$ **B** $Q(s_4, a_2) = \frac{1}{2}$ **C** $Q(s_5, a_3) = 2$
2. Suppose in a certain state s an agent can take actions a_1 or a_2. The Q-values are $Q(s, a_1) = 1$, $Q(s, a_2) = 4$ The agent uses an epsilon-greedy policy with $\epsilon = 0.5$. With which probability does the agent take action a_1?
   A 0 **B** 0.25 **C** 0.5 **D** 0.75 **E** 1.0
3. With which probability would a greedy agent take action a_1 in the setting above?
   A 0 **B** 0.25 **C** 0.5 **D** 0.75 **E** 1.0

Summary

Key Ingredients of Reinforcement Learning

- An **agent** performs **actions** A_t according to some **policy** in an **environment** and perceives **states** S_t and **rewards** R_t.
- The agent should choose a policy that trades off **exploitation** (acquire as much reward as possible) and **exploration** (learn more about potentially more rewarding parts of the environment).
- For exploitation the agent can rely on estimated **action values** $Q(s, a)$ that indicate the **expected cumulative reward** of action a in state s. The action values can be estimated with **Monte Carlo Estimation** (among many other methods not yet discussed).
- For exploitation the agent can occasionally take a random action. This is formalized in **epsilon-greedy** policies (among other exploration strategies not yet discussed).

Supervised vs. Unsupervised vs. Reinforcement Learning

	supervised	**unsupervised**	**reinforcement**
given	X, Y	X	an environment (the agent collects data)
goal	find $P(Y \mid X)$	find structure in data	find optimal policy
evaluation	test error	?	cumulative reward
approach	fit training data	use training data	interact with environment

Table of Contents

Q-Learning

Remember that action values are defined as

$$Q_\pi(S_t, A_t) = \mathrm{E}\left[R_{t+1} + R_{t+2} + \cdots + R_T\right]$$

The subscript π indicates that $Q_\pi(S_t, A_t)$ depends on the policy π used for future actions.

The equation above can also be written in recursive form as

$$Q_\pi(S_t, A_t) = \mathrm{E}\left[R_{t+1} + Q_\pi(S_{t+1}, A_{t+1})\right]$$

and for the optimal (greedy) policy π^* that takes in every state the action with maximal value

$$Q_{\pi^*}(S_t, A_t) = \mathrm{E}\left[R_{t+1} + \max_a Q_{\pi^*}(S_{t+1}, a)\right]$$

Therefore

$$\mathrm{E}\left[R_{t+1} + \max_a Q_{\pi^*}(S_{t+1}, a) - Q_{\pi^*}(S_t, A_t)\right] = 0$$

Q-Learning

$$\mathrm{E}\left[R_{t+1} + \max_a Q_{\pi^*}(S_{t+1}, a) - Q_{\pi^*}(S_t, A_t)\right] = 0$$

Question: Is there a way to start with an arbitrary $Q_0(S_t, A_t)$ and define an update rule $Q_k(S_t, A_t) \to Q_{k+1}(S_t, A_t)$ that depends on every observed transition $S_t, A_t \to R_{t+1}, S_{t+1}$ such that $\lim_{k\to\infty} Q_k(S_t, A_t) = Q_{\pi^*}(S_t, A_t)$?

Idea: The update rule should be such that the action value moves always in direction of the **Temporal Difference Error (TD-Error)**

$$\delta_k = R_{t+1} + \max_a Q_k(S_{t+1}, a) - Q_k(S_t, A_t)$$

i.e.

$$Q_{k+1}(S_t, A_t) = Q_k(S_t, A_t) + \lambda\delta_k$$

where λ is a learning rate, e.g. $\lambda = 0.1$.

Properties of Q-Learning

At convergence:

$$0 = \mathrm{E}[\delta_k] = \mathrm{E}\left[R_{t+1} + \max_a Q_k(S_{t+1}, a) - Q_k(S_t, A_t)\right]$$

i.e. Q-Learning stops when the action values Q_k satisfy the same equation as Q_{π^*} of the optimal greedy policy π^*.

Q-Learning is an **off-policy** method, because it estimates the Q-values for the greedy policy no matter what exploration policy is used; the Q-values of e.g. Monte Carlo Estimation depend on the exploration policy (Monte Carlo Estimation is an **on-policy** method).

Quiz

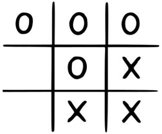
Suppose an agent experiences over and over an alternation of the two episodes
$(s_1, a_1, R_1 = 3, s_2, a_1, R_2 = 2, s_3, a_1, R_3 = 0)$ and
$(s_1, a_1, R_1 = 3, s_2, a_2, R_2 = 0, s_3, a_1, R_3 = 0)$.

Correct or wrong?

1. With learning rate $\lambda = 0.5$ and $Q_0(s, a) = 0$, Q learning finds $Q_1(s_1, a_1) = 1.5$.
2. With learning rate $\lambda = 0.5$ and $Q_0(s, a) = 0$, Q learning finds $\lim_{k\to\infty} Q_k(s_1, a_1) = 5$.
3. After experiencing every episode 5 times, Monte Carlo Estimation would find $Q(s_1, a_1) = 5$.

Table of Contents

Application: Tic-Tac-Toe



The player who succeeds in placing three of their marks in a diagonal, horizontal, or vertical row is the winner.

There are 255'168 different games, 131'184 end in a win of the first player, 77'904 end in a win of the second player and 46'080 end in a draw.

There are 5'478 different board positions (765 without rotations and mirroring).

- We use an enumerated representation of the states, i.e. $s \in \{1, 2, 3, \ldots 5478\}$.
- We use a Monte Carlo Estimator to learn the Q-values.
- Two epsilon-greedy players (cross and nought) play against each other in **self-play**.
- Each player stores all states (encountered when it is the players turn) together with the selected action in its own episode.
- Each episode ends with reward +1 for the winner, reward -1 for the looser and reward 0 in case of a draw.

Book

This lecture in inspired by the excellent text book

Reinforcement Learning: An Introduction
Richard S. Sutton and Andrew G. Barto
Second Edition
`http://incompleteideas.net/book/the-book-2nd.html`

See CS-456 for more Reinforcement Learning at EPFL
`https://lcnwww.epfl.ch/gerstner/VideoLecturesANN-Gerstner.html`