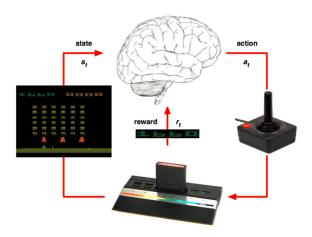
Deep Q-Learning in a nutshell

Bar Hilleli

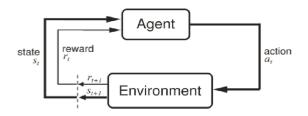
barhilleli5@gmail.com

Oct 15, 2017

Reinforcement Learning - background (0)



Reinforcement Learning - background (1)



- Reinforcement Learning (RL) describes a set of learning problems where an agent interacts with an environment.
- RL algorithms seek to find a policy, π , that **maximize the reward** received over time from the environment.

Markov Decision Process - MDP

Definition

The MDP framework consists of four elements: (S, A, R, P)

- \circ S is a discrete set of states
- ullet \mathcal{A} is a discrete set of actions
- ullet \mathcal{R} is a reward model $\mathcal{R}: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- $\mathcal{P}: \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a transition probability matrix

$$\mathcal{P}_{ss'}^{\mathsf{a}} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = \mathsf{a}]$$

Policy

Definition

A (deterministic) **policy** is a mapping from a given world state *s*, to a desired action *a*:

$$\pi: \mathcal{S} \to \mathcal{A}$$

5 / 20

Objective

Given an MDP (S, A, P, R) we wish to find a policy so as to **maximize** the **long term sum of the discounted immediate rewards**

$$\sum_{t=0}^{\infty} \gamma^t r_t$$

where γ is called the *discount factor* in range [0, 1].

State-Value Function

Definition

The state-value function for policy π , denoted by $V^{\pi}(s,a)$, is the expected return when starting in state s and following policy π thereafter, and is formally defined as,

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s
ight]$$

Action-Value Function

Definition

Similarly, The action-value function for policy π , denoted by $Q^{\pi}(s,a)$, is the expected reward of taking action a in state s and following policy π thereafter, and is formally defined as,

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a
ight]$$

Optimal Policy

Considering finite MDPs, an **optimal policy**, π^* , is better than all other policies in the sense that its expected return is greater than or equal to that of all other policies for all states. Meaning,

$$V^{\pi^\star}(s) \geq V^\pi(s)$$

for all $s \in \mathcal{S}, \pi \in \Pi$.

9 / 20

Optimal State-Value Function

Definition

The value functions of π^* is called **optimal state-value function**, denoted by V^* and defined as,

$$V^{\star}(s) = \max_{\pi} \ V^{\pi}(s)$$

for all $s \in \mathcal{S}$.

Optimal Action-Value Function

Definition

Similarly the **optimal action-value function**, denoted Q^* , is defined as

$$Q^{\star}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

Q^* in terms of V^*

 $Q^{\star}(s,a)$ gives the expected return taking action a in state s and thereafter following π^{\star} , therefore we can write,

$$Q^{\star}(s, a) = \mathbb{E}_{s_{t+1}}[r_{t+1} + \gamma V^{\star}(s_{t+1}) \mid s_t = s, a_t = a]$$

Bellman Optimality Equation

Since the value of a state s under π^* must be equal to the expected return of the best action from that state we have that,

$$V^{\star}(s) = \max_{a \in \mathcal{A}} Q^{\star}(s, a)$$

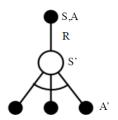
Therefore, we can write the **Bellman optimality equation** for Q^* ,

$$Q^{\star}(s, a) = \mathbb{E}_{s_{t+1}} \left[r_{t+1} + \gamma \max_{a'} \ Q^{\star}(s_{t+1}, a') | s_t = s, a_t = a \right]$$

Q-Learning

- The Q-learning algorithm aims to find the optimal action-value function.
- The Q-learning update is:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$



Q-Learning Algorithm

Q-learning: An off-policy TD control algorithm

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$ Repeat (for each episode):

Initialize S

Repeat (for each step of episode):

Choose A from S using policy derived from Q (e.g., ϵ -greedy)

Take action A, observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_{a} Q(S', a) - Q(S, A)]$$

$$S \leftarrow S'$$

until S is terminal

Q-Learning Properties

- Model-Free: no knowledge of MDP
- TD(0): bootstrapping updates a guess towards a guess
- Off-Policy: learn about optimal policy while following exploratory policy

Theorem

Q-learning control converges to the optimal action-value function,

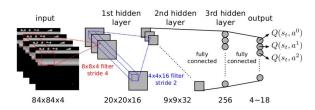
$$Q(s,a) \rightarrow Q^{\star}(s,a)$$

Approximating $Q^*(s, a)$ using a Neural Network

• Dealing with a very large state space of images, we **approximate** the optimal action-value function using a **deep Q-network** (DQN) with **parameters** θ :

$$Q_{\theta}(s,a) \approx Q^{\star}(s,a).$$

 A deep Q-network is a neural network that for a given state outputs a vector of action values



DQN Algorithm

The **DQN** algorithm:

- Take action a_t according to ϵ -greedy policy.
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory \mathcal{B} .
- Sample random mini-batch of transitions (s, a, r, s') from \mathcal{B} .
- ullet Compute Q-learning targets w.r.t. old, fixed parameters $ilde{ heta}.$
- Optimize MSE between Q-network and Q-learning targets

$$L(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{B}} \left[\left(r + \gamma \max_{a'} Q_{\tilde{\theta}}(s', a') - Q_{\theta}(s, a) \right)^{2} \right]$$

• Using variant of stochastic gradient descent.

Toward Deep Reinforcement Learning without a Simulator: An Autonomous Steering Example

Video

Toward Deep Reinforcement Learning without a Simulator: An Autonomous Steering Example

Thank you