Deep Reinforcement Learning

A brief survey

Kai Arulkumaran   Marc Peter Deisenroth   Miles Brundage
Anil Anthony Bharath

December 7, 2017

Introduction

Reinforcement learning

The essence of RL is learning through interaction: an RL agent interacts with its environment and, upon observing the consequences of its actions, can learn to alter its own behavior in response to rewards received.

This paradigm of trial-and-error learning has its roots in behaviorist psychology and is one of the main foundations of RL.

Introduction

Reinforcement learning

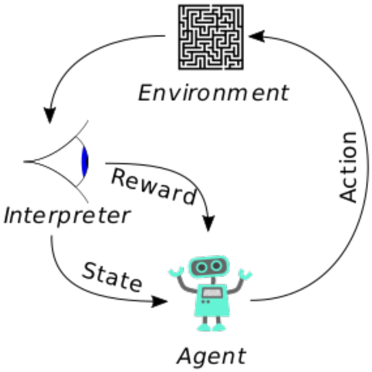

Figure: The typical framing of an RL scenario

Background

Markov decision process

RL can be described as a a Markov decision process, which is a quadruple (S, A, T, R):

- a set of state S
- a set of actions A
- transition probability $T : S \times A \times S \mapsto \mathbb{R}$
- reward function $R : X \times A \times X \mapsto \mathbb{R}$ or $R : X \times X \mapsto \mathbb{R}$

Background

Markov decision process

A policy π is a map from a state to an action

$$\pi : S \mapsto A$$

if it is deterministic; or a map from a state and an action to a probability

$$\pi : S \times A \mapsto [0, 1]$$

with $\sum_a \pi(s, a) = 1$ if it is stochastic. The goal of RL is to find an optimal policy π^* that maximize the expected return from all states

$$\pi^* = \arg\max_{\pi} \mathbb{E}[R|\pi]$$

Background

Markov decision process

A key concept underlying RL is the Markov property — only the current state affects the next state. This means that any decision made at s_t can be based solely on current state, rather than former states.

Background

Markov decision process

A key concept underlying RL is the Markov property — only the current state affects the next state. This means that any decision made at s_t can be based solely on current state, rather than former states.

Although this assumption is held by the majority of RL algorithms, it is somewhat unrealistic, as it requires the states to be fully observable. A generalization of MDPs are partially observable MDPs, in which the agents receives an observation $o \in \Omega$ depending on the current state and the previous action:

$$O : S \times A \mapsto \Omega$$

.

RL Algorithms

Value functions

Value function methods are based on estimating the value of being in a state. The state-value function $V^{\pi}(s)$ is the expected return when starting in state s and following π subsequently:

$$V^{\pi}(s) = \mathbb{E}[R|s, \pi]$$

The optimal policy has a corresponding state-value function $V^{*}(s)$, and vice versa. If the optimal state-value function is known, we can simply pick the action that maximizes the next state's value function:

$$a = \arg\max_{a} \mathbb{E}_{s_{t+1} \sim T(s_{t+1}|s_t, a)}[V^{*}(s_{t+1})]$$

RL Algorithms

Value functions

Since the transition function T is unknown, we may construct another function, the state-action value $Q^{\pi}(s, a)$:

$$Q^{\pi}(s, a) = \mathbb{E}[R|s, a, \pi]$$

the best policy can be found by greedily choose a state given $Q^{\pi}(s, a)$:

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

and thus

$$V^{\pi}(s) = \max aQ^{\pi}(s, a)$$

RL Algorithms

Dynamic programming

The state-action function can be written as

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))]$$

This means that Q^{π} can be improved by bootstrapping:

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha\delta$$

where α is the learning rate and $\delta = Y - Q^{\pi}(s_t, a_t)$ is the temporal difference error.

Deep Reinforcement Learning

Many DRL algorithms are based on scaling up prior work in RL to high-dimensional problems.

Deep Reinforcement Learning

Many DRL algorithms are based on scaling up prior work in RL to high-dimensional problems.

However, in high-dimensional problems, state space is too large that it is impossible to calculate the Q function for each state-action pairs.

Deep Reinforcement Learning

Many DRL algorithms are based on scaling up prior work in RL to high-dimensional problems.

However, in high-dimensional problems, state space is too large that it is impossible to calculate the Q function for each state-action pairs.

The deep Q-network (DQN), uses a deep neural network to represent the Q function, instead of Q-table. This compresses the Q function. Furthermore, information of a state-action pair can be propagated to other state-action pairs.

Deep Reinforcement Learning

Experience replay

Experience replay stores transitions of the form $(s_t, a_t, s_{t+1}, r_{t+1})$ in a cyclic buffer.

- RL agents can sample from and train on data offline;
- The temporal correlations that affects RL algorithms are broken;
- From a practical perspective, batches of data can be efficiently processed in parallel.

Deep Reinforcement Learning

Target networks

Target network is a network that initially contains the weights of the network enacting the policy but is kept frozen for a large period of time. Rather than having to calculate the TD error based on its own rapidly fluctuating estimates of the Q-values, the policy network uses the fixed target network. During training, the weights of the target network are updated to match the policy network after a fixed number of steps.

Q & A