Reinforcement learning and the Policy gradient method. a fast introduction

Antonio Massaro [1]

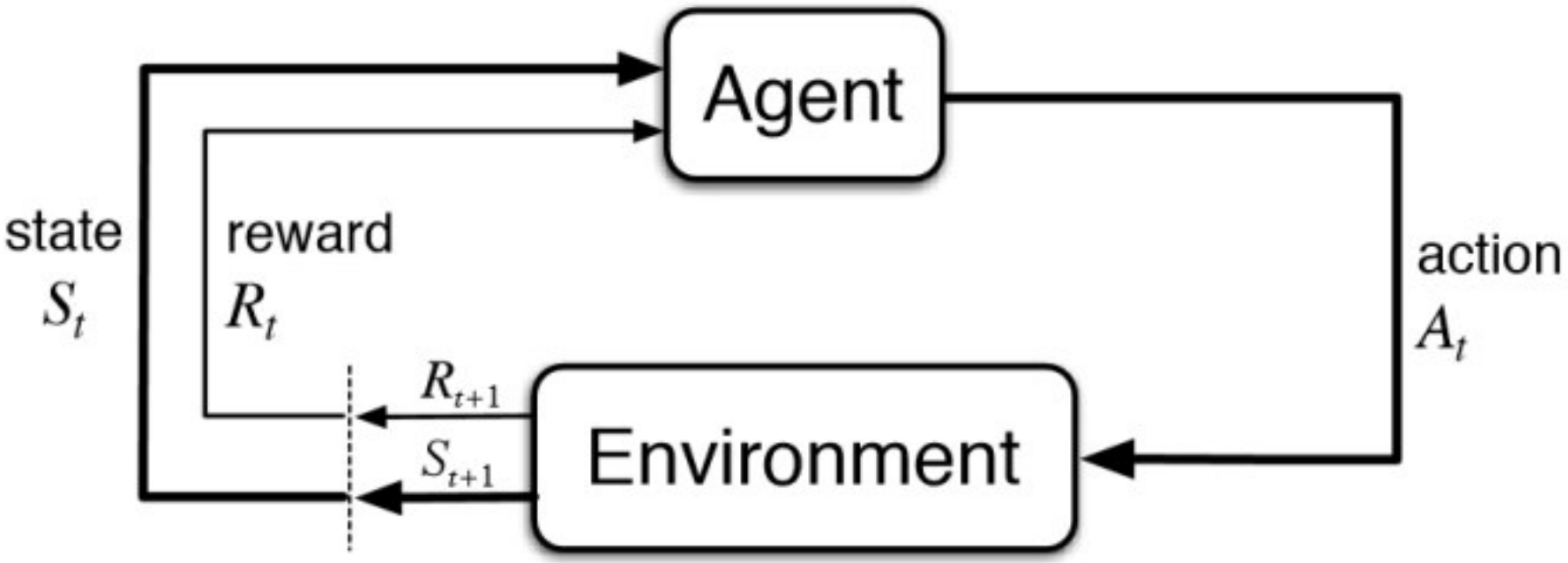
[1] Nokia Bell Labs, Paris.

07/11/2019

Markov decision problems

MDPs are meant to be a straightforward framing of the problem of **learning** from **interaction** to achieve a **goal**.
[Sutton, Reinforcement learning: an introduction]

The RL model



- Finite states
- Finite actions
- Discrete time-steps

Objective: maximize the reward

States, Actions, Rewards,Transition probabilities

Interacting with the environment the agent produces a *trajectory*

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, S_3, A_3, ...$$

S_t and R_t are markovian stochastic processes.

$$Pr(S_t = s', R_t = r | S_{<t}, R_{<t}, A_{<t}) =$$

$$Pr(S_t = s', R_t = r | S_{t-1} = s_{t-1}, A_{t-1} = a_{t-`1}) = p(s', r|s, a)$$

Transition probabilities:

$$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$$

$$(s', r, s, a) \longmapsto p(s', r|s, a)$$

States, Actions, Rewards,Transition probabilities

Interacting with the environment the agent produces a *trajectory*

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, S_3, A_3, ...$$

S_t and R_t are markovian stochastic processes.

$$Pr(S_t = s', R_t = r | S_{<t}, R_{<t}, A_{<t}) =$$

$$Pr(S_t = s', R_t = r | S_{t-1} = s_{t-1}, A_{t-1} = a_{t-`1}) = p(s', r|s, a)$$

Transition probabilities:

$$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$$

$$(s', r, s, a) \longmapsto p(s', r|s, a)$$

States, Actions, Rewards,Transition probabilities

Interacting with the environment the agent produces a *trajectory*

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, S_3, A_3, ...$$

S_t and R_t are markovian stochastic processes.

$$Pr(S_t = s', R_t = r | S_{<t}, R_{<t}, A_{<t}) =$$

$$Pr(S_t = s', R_t = r | S_{t-1} = s_{t-1}, A_{t-1} = a_{t-`1}) = p(s', r|s, a)$$

Transition probabilities:

$$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$$

$$(s', r, s, a) \longmapsto p(s', r|s, a)$$

Policies

A policy is a function from states to probability distributions over actions available at such state.

$$\pi(a|s) = \text{probability of selecting action } a \text{ at state } s$$

Policies

A policy is a function from states to probability distributions over actions available at such state.

$$\pi(a|s) = \text{probability of selecting action } a \text{ at state } s$$

Values function

Total discounted reward

$$G_t = \sum_{k=t+1}^{\infty} \gamma^{k-t-1} R_k$$

Value function

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s \right]$$

Q-value

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right]$$

Objective function, optimal policies

Maximize the total expected discounted reward

$$\pi^* = \arg\max_{\pi \in policies} v_\pi(s), \forall s \in \mathcal{S}$$

π^* exists and its value functions satisfy the Bellman equations.

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

$$q_*(s, a) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]$$

Objective function, optimal policies

Maximize the total expected discounted reward

$$\pi^* = \arg\max_{\pi \in policies} v_\pi(s), \forall s \in \mathcal{S}$$

π^* exists and its value functions satisfy the Bellman equations.

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

$$q_*(s, a) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]$$

Known transition probabilities

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

1. Solve in $v_*(s)$ (Caccioppoli-Banach theorem)

2. define $\pi^*(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')]$.

Unknown transition probabilities: Q-learning

- initialize $q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$
- initialize $s \in \mathcal{S}$
- repeat
  1. randomly choose action a
  2. rake action a, observe r, s'
  3. $q(s, a) \leftarrow q(s, a) + \alpha[r + \gamma \max'_a[q(s', a') - q(s, a)]$
  4. s$\leftarrow s'$

It converges to $q^*(s, a)$!

Unknown transition probabilities: Q-learning

- initialize $q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$
- initialize $s \in \mathcal{S}$
- repeat
  1. randomly choose action a
  2. rake action a, observe r, s'
  3. $q(s, a) \leftarrow q(s, a) + \alpha[r + \gamma \max'_a[q(s', a') - q(s, a)]$
  4. s$\leftarrow s'$

It converges to $q^*(s, a)$!

The policy gradient theorem

Suppose policy $\pi_\theta(a|s)$ is a differentiable function of parameter θ

$$\nabla_\theta v^\pi = \sum_{s \in S} p^\pi(s) \sum_{a \in A(s)} \nabla_\theta \pi(a|s) q^\pi(s,a)$$

- $p^\pi(s)$ is the stationary distribution under π.

Reinforce[Williams, 1992]

$$\theta_{t+1} \leftarrow \theta_t + \alpha G_t \frac{\nabla_\theta \pi(a_t|s_t, \theta_t)}{\pi(a_t|s_t, \theta_t)}$$

- $\alpha < 1$ is the learning rate
- G_t is the sample cumulative reward from time t on

Advantages of policy gradient methods

- A policy may be a simpler that state-action values to approximate
- If so, policy-gradient will learn faster and yield a superior asymptotic performance
- Policy parametrization is a good way of injecting prior knowledge about the desired form of the policy.