[12pt,a4paper]article amsfonts amsmath amssymb geometry indentfirst left=2.0cm,right=2.0cm,top=2.5cm

Markov Decision Processes (MDP)

esgl Hu

March 12, 2018

1 Background

aaaaaaaaa

2 Notation & Definition

A reinforcement learning task taht satisfies the Markov property is called a *Markov decision process*, or *MDP*. If the state and action spaces are finite, then it is called a *finite Markov decision process (finite MDP)*

Given any state and actin s and a, the probability of each possible pair of next state and reward, $s^{'}$, r, is denoted

$$p(s^{'}, r|s, a) \doteq Pr(\{S_{t+1} = s^{'}, R_{t+1} = r|S_t = s, A_t = a\}) \tag{1}$$

Given the dynamics as specified by (1), one can compute anything else one might want to know about the environment, such as the expected rewards for state-action pairs.

$$r(s, a) \doteq \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{r\in\mathcal{R}} r \sum_{s^{'}\in\mathcal{S}} p(s^{'}, r|s, a), \tag{2}$$

the *state-transition probabilities*,

$$p(s^{'}|s, a) \doteq Pr\{S_{t+1} = s^{'}|S_t = s, A_t = a\} = \sum_{r\in\mathcal{R}} p(s^{'}, r|s, a) \tag{3}$$

and the expected rewards for state-action-next-state triples,

$$r(s, a, s^{'}) \doteq \mathbb{E}[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s^{'}] = \frac{\sum_{r\in\mathcal{R}} rp(s^{'}, r|s, a)}{p(s^{'}|s, a)} \tag{4}$$

[12pt,a4paper]article amsfonts amsmath amssymb geometry indentfirst pifont ulem algorithm algorithmicx algpseudocode amsmath
left=2.0cm,right=2.0cm,top=2.5cm,bottom=2.5cm
Policy Gradient Methods esgl Hu

3 preface

Notations & Definitions

The methods have learned the values of actions and then selected actions based on their estimated action values, their policies would not even exist without the action-value estimates.

The methods have learned a *parameterized policy* that can select actions without consulting a value function. A value function may still be used to *learn* the policy weights, but is not required for action selection.

The notation $\theta \in \mathbb{R}^n$ is used for the primary learned weigth vector, $\pi(a|s,\theta) \doteq \Pr\{A_t = a|S_t = s, \theta_t = \theta\}$ for the probability that action a is taken at time t given that the agent is in state s at time t weight vector θ.

These methods seek to maximize preformance $\eta(\theta)$, so their updates approximate gradient ascent in η:

$$\theta_{t+1} \doteq \theta_t + \alpha \widehat{\nabla(\eta(\theta_t)})) \tag{5}$$

where $\widehat{\nabla(\eta(\theta_t)})$) is a stochastic estimate whose expectation approximates the gradients of the performance measure $\eta(\theta_t)$ with respect to its argument θ_t

Advantages & Disadvantages

Advantages

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

Disadvantages

- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

Definition of Policy Gradient

We let τ denote a state-action sequence $\{s_0, a_0, ..., s_H, a_H\}$. We overload nototation: $R(\tau) = \sum_{t=0}^{H} R(s_t, a_t)$.

$$\eta(\theta) = \mathbb{E}[\sum_{t=0}^{H} R(s_t, a_t), \pi_\theta] = \sum_{\tau} P(\tau;\theta) R(\tau) \tag{6}$$

In our new notation, our goal is to find θ:

$$\max_{\theta} \eta(\theta) = \max_{\theta} \sum_{\tau} P(\tau;\theta) R(\tau) \tag{7}$$

REINFORCE: A Monte-Carlo Policy-Gradient Method (episodic) a differentiable policy parameterization $\pi(a|s,\theta)$, $\forall a \in A, s \in S, \theta \in R^n$ Initialize policy weights θ True Generate an episode $\{S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T, S_T\}$ follow $\pi(\cdot|\cdot,\theta)$ each step of the episode $t = 0, 1, ..., T-1$ $G_t \leftarrow$ return from step t $\theta \leftarrow \theta + \alpha\gamma^t G_t \nabla_\theta log\pi(A_t|S_t,\theta)$