Reinforcement Learning!

Meichen Lu (ml574@cam.ac.uk)

April 26, 2018

Contents

1 Introduction

Often we do not have clear labels for explicit supervised learning. In the reinforcement learning (RL) framework, we will instead provide our algorithms only a reward function, which indicates to the learning agent when it is doing well, and when it is doing poorly. Reinforcement learning has been successful in applications as diverse as

- autonomous helicopter flight
- robot legged locomotion
- cell-phone network routing
- marketing strategy selection
- factory control
- efficient web-page indexing

In the operations research and control literature, reinforcement learning is called *approximate dynamic programming*, or *neuro-dynamic programming*.
In machine learning, the environment is typically formulated as a **Markov decision process (MDP)**, as many reinforcement learning algorithms for this context utilize dynamic programming techniques.The main difference is that the RL do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.

2 MDP and related algorithms

2.1 MDP definition

Components of MDP

- S is a set of environment and agent states;
- A a set of actions of the agent;
- P_{sa} are the state transition probabilities. i.e., the probability of transition from state s to state s' under action a.

$$P_{sa}(s, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a) \tag{1}$$

- $R : S \times A \mapsto \mathbb{R}$ is the reward function (or immediate reward)
- $\gamma \in [0, 1)$ is called the **discount factor**
- A **policy**, $\pi : S \mapsto A$, is any function mapping from the states to actions.
- Value function for a policy π is $V^\pi(s)$

MDP process is a Markov chain.

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} ...$$

The total payoff is given by

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots \tag{2}$$

Bellman equations Given a fixed policy π, its value function V^π satisfies the Bellman equations [Recursion]:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s') = R(s) + \gamma E_{s' \sim P_{s\pi(s)}}[V^\pi(s')] \tag{3}$$

Value iteration and policy iteration Value iteration find the maximum $V^\pi(s)$, V^*; policy iteration finds the policy (π^*) with maximum value function.
Convergence is guaranteed: after at most a finite number of iterations of this algorithm, V will converge to V^*, and π will converge to π^*.

2.2 Learning a model for an MDP

MLE estimates for the state transition probabilities:

$$P_{sa}(s') = \frac{\#\text{times we took action } a \text{ in state } s \text{ and got to } s'}{\#\text{times we took action } a \text{ in state } s} \tag{4}$$

We cal also learn the reward $R(s)$ as the average reward observed in state s.

2.3 Continuous state MDPs

2.3.1 Discretisation

Problem: piecewise constant; curse of dimensionality

2.3.2 Value function approximation

Model/simulator Sometimes, we have a physical model for the state transition probabilities.

Learnt model e.g. linear model (in analogy to LDS with control, $a_t \equiv$ the control term u_t) $s_{t+1} = As_t + Ba_t$ and learn the model by

$$\arg\min_{A,B} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left\| s_{t+1}^{(i)} - \left(As_t^{(i)} + Ba_t^{(i)} \right) \right\|^2 \tag{5}$$

Alternatively, we can build a stochastic model

$$s_{t+1} = As_t + Ba_t + \epsilon_t, \text{ where } \epsilon_t \sim \mathcal{N}(0, \sigma)$$

Fitted value iteration Learning the value function $V(s) = \theta^T \phi(s)$ where $\phi(s)$ is some appropriate feature mapping of the state. Given

$$s \text{ and } y(s) = V(s) = R(s) + \gamma max_a E_{s' \sim P_{s\pi(s)}}[V^\pi(s')]$$

We use supervised learning to learn the parameter $\theta = \arg\min_\theta \frac{1}{2} \left(\theta^T \phi(s^{(i)}) - y^{(i)} \right)^2$

3 Finite horizon MDPs

A finite horizon MDP will be defined as a tuple

$$(S, A, P_{sa}, T, R)$$

with $T > 0$ being the time horizon. The optimal policy π^* might be non-stationary, i.e. $\pi^{(t)} : S \mapsto A$.

3.1 Linear Quadratic Regulation (LQR)

3.2 Differential Dynamic Programming (DDP)

3.3 Linear Quadratic Gaussian (LQG)