ML Section 2: Reinforcement Learning

Adam Hawley

March 14, 2019

Contents

1 Psychology & Engineering of Reinforcement Learning

See lecture for history.

2 Agent Paradigm

A common setup for the agent paradigm of RL involves having an agent which carries out **actions** on an **environment** through the use of **effectors**. These actions are based on the **reward signal** and **states** from the environment which are read using **sensors**.

3 RL vs. Supervised Learning

What makes RL different?

- No supervisor feedback: learning from reward signal only which is often delayed, not instantaneous.
- Agent influences the selection of training experience.
- Sequential aspect of decision making.

4 The Markov Decision Process

The Markov decision process is used to model the environment.

A (single-agent) MDP is a tuple (S,A,p,R) where:

S A set of states

A A set of actions

p: S x A x S → [0,1] Specifies the transition probabilities between states.

R: S x A → R Specifies the reward for each state-action pair.

Note: deterministic transition function $\delta : S \times A \implies S$.

5 Agent State vs. Environment State

It is very important to recognise the difference between these two. Agents have their own representation of the environment state. Agents may have **partial observability** so they are not able to observe everything in the environment. This may be benefitial as they may want to focus in on relevant parts of the environment. Ideally only the information necessary to make an optimal decision is contained in the agent state.

6 Markov States

A state S_t is Markov if and only if S_t contains all relevant information to determine the next state.

Note: any state can be made into a Markov state by incorporating the complete history.

7 RL Output

Goal: learn an *optimal policy* π: S $\rightarrow$ A

Evaluation of policy via discounted cumulative reward:

$$V^{\pi}(S_t) = \sum_{i \leq 0} \gamma^i r_{t+1} \tag{1}$$

where:

- $0 \leq \gamma < 1$ is a discount factor ($\gamma = 0$ means that only immediate reward is considered).
- r_{t+1} is the reward at time t+1 determined by performing actions specified by policy π.

The goal previously mentioned can now be redefined as: Agent learns policy π that maximises $V^{\pi}(s)$ for all states s. So the optimal policy π^* = argmax_{π} V^{π}(S) for all S. **Notation**: $V^{\pi^*}(S) = V^*(S)$.

The optimal action in state s is the action a that maximises the sum of the immediate reward r(s,a) plus the value V^* of the immedaite successor state, discounted by γ:

$$\pi * (S) = argmax_a[r(s,a) + \gamma V^*(\delta(s,a))] \tag{2}$$

If functions r and δ are known then the agent can acquire π^* by computing V^*.

7.1 Bellman Equation

For computing V^*

$$V^*(s) := r(s) + \gamma max_a \sum_s T(s,a,s')V^*(s') \tag{3}$$

For n states: n equations with n unknowns. **But**: n equations are non-linear (because of the *max* operator).

7.2 Value Iteration

1. For all states s: $\mathrm{V_0(s)} := 0$
2. t := 0
3. Update values of all states s based on successor states: $V_{t+1}(s) := r(s) + \gamma max_a V_t(\delta(s,a))$ (the Bellman equation for deterministic MVPs)
4. t := t + 1;
5. Repeat steps 3 and 4 until convergence (or had enough)

This algorithm is **guaranteed** to converge to the optimal solution. See lecture for formal proof on this.

7.3 Model-Based vs. Model Free

What if δ and r are unknown?

- Solution 1: Learn them both from experience (model-based).
- Solution 2: Learn quality function Q directly (model-free).

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a)) \tag{4}$$

This is another version of the Bellman Equation. This means that the optimal policy computation: $\pi^*(s) = argmax_a Q(s,a)$. And it is then possible to learn Q even if δ and r are unknown. **SEE LECTURE FOR Q LEARNING ALGORITHM + PROOF**

7.4 Policy Iteration

(An alternative to Value Iteration)

1. Start with arbitrary policy π_0
2. i := 0
3. Evaluate: Given π_i calculate corresponding $\mathrm{V_i}$
4. Improve: $\pi_{\mathrm{i+1}}\mathrm{(s)} := \mathrm{argmax_a(V_i(}\delta\mathrm{(s,a)))}$
5. Repeat steps 3 and 4 until convergence(or had enough)

7.4.1 Evaluating a Policy

Given π_{i}: $\mathrm{V_i(s) = r(s)} + \gamma\ \mathrm{V_i(}\delta\mathrm{(s,a))}$ for all states s. If there are n states, this yields n linear equations in n unknowns since there is no longer an argmax term. Therefore it is *easy* to solve.

7.5 Reminder of RL vs. Supervised Learning

- Supervised learning: instructive feedback.
- Reinforcement learning: evaluative feedback.

Therefore in RL, there is a need for active exploration: trial-and-error search.

7.5.1 Exploration vs. Exploitation

- Exploration: choose actions that gather information on the environment
- Exploitation: choose actions which have (so far) shown to lead to large rewards.

The **exploration (action selection) strategy** is therefore controlling the trade-off between exploration and exploitation. Generally we shift gradually with time from exploration to exploitation. A couple of examples of these strategies include ϵ-*greedy* and *Boltzmann.*

SEE LECTURE FOR DETAILS OF THESE EXAMPLES