Reinforcement Learning

Kristian Wichmann

January 28, 2018

1 Basics of reinforcement learning

Reinforcement learning deals with an *agent* making *decisions* over time in dealing with some *environment*, to maximize some cumulative, scalar *reward*.

An example would be playing a video game. Here, the player is the agent. The control inputs are the decisions. The game and its internal logic is the environment. And the score is the reward.

Sometimes, a reward will be greater if it is postponed - it is not always advantagegous to reap any immediate reward. In other words, reinforcement learning algorithms will not benefit from being greedy.

1.1 The reward hypothesis

The basis of all reinforcement learning is the *reward hypothesis*, which states:

> All goals can be described by the maximization of some cumulative, expected reward.

1.2 Observation, action, and reward

At at given timestep t, a reinforcement learning agent gets some input; an *observation* O_t about the environment. It then takes an *action* A_t. And finally it gets a scalar *reward* R_t.

1.3 History and state

At any given time step t, the agent has a *history* H_t, which simply consist of all the observations, actions, and rewards that have happened so far:

$$H_t = O_1, A_1, R_1, O_2, A_2, R_2, \cdots, O_t, A_t, R_t, \tag{1.1}$$

A *state* is a way for to parse this history into a more meaningful form. Formally, the state representation is simply a function of the history:

$$S_t = f(H_t) \tag{1.2}$$

It is very important to note, that this is generally distinct from the *environment state* H_t^e. The environment state contains complete information about the environment, including data and mechanisms that may be hidden to the agent. Hence H_t^e can depend on other things than just the historySuch information is *private* to the environment.

The *agent state* S_t^a on the other hand is the internal representation the agent uses to decide on which actions to take. Once again, it can be any function of history:

$$S_t^a = f(H_t) \tag{1.3}$$

1.4 Markov states

A state S_{t+1} is called a *Markov state* or an *information state* if it only depends on the state of the previous time step. Expressed probabilistically:

$$\mathbb{P}[S_{t+1}|S_1, S_2, \cdots S_t] = \mathbb{P}[S_{t+1}|S_t] \tag{1.4}$$

In other words, we don't need the entire history to decide on an action: Knowing the present is enough. Or put another way:

The future is independent of the past, given the present.

Or:

The state is a *sufficient statistic* of the future.

The environment state is always Markov, as by definition it contains all information about what can happen next. Similarly, the state consisting of the entire history is trivially Markov as well.

1.5 Full observability

This is the case where, in fact, we can observe everything about the environment and its inner workings. So that:

$$O_t = S_t^a = S_t^e \tag{1.5}$$

Sometimes this is reasonable. Sometimes not. But it will be a useful theoretical situation. This is known as a *Markov decision process* or MDP for short.

When this condition is not fulfilled, we speak about *partial observability* or a *partially observable evironment*. Here the agent only indirectly observes the invironment. This situation is known as a *partially observable Markov decision process* or POMDP for short.

1.6 State example: Bayesian beliefs

This state representation can be seen as a current best bet at what the actual environment state is. In other words, it is represented by Bayesian probabilities, which may then be updated over time using Bayes' rule:

$$S_t^a = (\mathbb{P}[S_t^e = s_1], \mathbb{P}[S_t^e = s_2], \cdots, \mathbb{P}[S_t^e = s_n]) \tag{1.6}$$

1.7 State example: Recurrent neural net

Here, the state S_t^a is a linear combination of the observation O_t and the state of the last time step S_{t-1}^a, followed by a non-linear *activation function* σ:

$$S_t^a = \sigma(W_O O_t + W_S S_{t-1}^a) \tag{1.7}$$

Here, the W's are weight matrices with sizes corresponding to the dimenstionality of observations and state vectors. Typical choices for σ are sigmoid, tanh, or rectified linear unit.

2 Components of an agent

A reinforcement learning agent may contain one or more of the following components:

- *Policy*: The agent's behaviour function. Shows how the agent gets from its state to deciding on an action.
- *Value function*: A measure of how desirable it is to be in a given state, or perform a given action.
- *Model*: The agent's representation of the environment.

We'll examine each of these in greater detail below:

2.1 Policy

A policy π is a mapping from state to action. For a *deterministic* policy, this is an ordinary function:

$$\pi(s) = a \tag{2.1}$$

So the state s is always mapped to the action a. We should ideal choose π so that the reward is maximized.

But a policy can also be *stochastic*, i.e. probabilistic. In this case π takes the form of conditional probabilities:

$$\pi(a|s) = \mathbb{P}[A = a|S = s] \tag{2.2}$$

2.2 Value function

A value function V_π is a prediction of the future reward for state s under a given policy π:

$$V_\pi(s) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots |S_t = s] \tag{2.3}$$

Here γ is a *discounting factor*, between 0 and 1. 0 means that we only care about the immediate reward, while values approaching 1 means that we care progressively more about long run rewards.

2.3 Model

In this context, a model tries to predict the evolution of the environment. These can take two different forms:

- *Transition prediction* $\mathcal{P}$ models the change of state of the environment:

$$\mathcal{P}^a_{ss'} = \mathbb{E}[S_{t+1} = s'|S_t = s, A_t = a] \tag{2.4}$$

- *Reward prediction* $\mathcal{R}$ models the change in (immediate) reward:

$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] \tag{2.5}$$

2.4 Categorization of agents

Based on the above, we can broadly categorize agents into several categories:

- *Value based agents* are based on a value function - with the policy being implicit. I.e. in each step we can greedily pick the action with the highest future reward, based directly on the value function.

- *Policy based agents* on the other hand has the policy as its key ingredient. Here, the policy mapping directly shows what action to take in each state.
- *Actor critic agents* takes both a poicy and a value function into account. Ideally the "best of both worlds".
- *Model free agents* may by policy and/or value function based, but has no model! I.e. it doesn't try to make predictions about the environment.
- *Model agents* may by policy and/or value function based, but includes modelling.

3 Reinforcement learning and planning

There's two fundamental types of problems in sequential decision making: *Reinforcement learning* and *planning.*

Reinforcement learning can be viewed as dumping the agent into an alien environment it initially knows nothing about. The agent then begins to interact with the environment, eventually improving its policy for doing so.

Planning is the situation where the agent starts with a perfect model of the environment. The agent then uses this model to make calculations without any external interaction, which then informs policy.

So in planning, we can in principle look many steps ahead, determining states and rewards (or expectation values thereof) and use this to pick optimal actions. In other words, a tree search.

3.1 Exploration versus exploitation