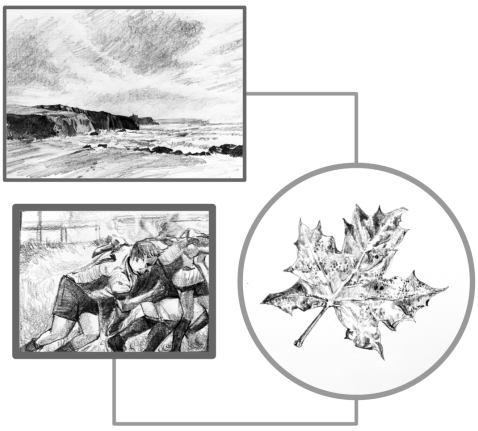

13

Optimising actions for control objectives

Concept. The idea here is

13.1 States, actions and attributing rewards

Up to this point, we have only considered actions which were either scheduled up front through some fixed process or through user interaction via a game interface. In order to start creating algorithms to act on the system state for us, we now need to develop a formalism which 'closes the loop' by feeding information back from the stochastic process to another decision-making process. Note that in most cases, the state of real-world phenomena cannot be measured perfectly. So in order to enable any agent trained on simulated phenomena to potentially act in the real world, we will need to model this measurement process as part of the information retrieval step.

Let's now define the concept of an 'environment state' $\mathcal{S}_{\mathsf{t}+1}$ at timestep $\mathsf{t}+1$; this is a new vector that doesn't have to share the same length as the measured state vector $X_{\mathsf{t}+1}$. We will then say generally that this environment state is 'observed' by the agent using the following observation function

$$\mathcal{S}^i_{\mathsf{t}+1} = O^i_{\mathsf{t}+1}(X', \mathcal{Z}_{\mathsf{t}+1}, \mathsf{t})\,, \tag{13.1}$$

where we have also introduced a new vector $\mathcal{Z}_{\mathsf{t}+1}$ which we will use to store all of the relevant parameters to the agent[1] at timestep $\mathsf{t}+1$.

If we are now given the conditional probability that an action vector element $\mathcal{A}_{\mathsf{t}+1} = a$ is chosen given that state vector $\mathcal{S}_{\mathsf{t}+1} = s$ has been measured $\pi(a, s) = p(a|s)$, we can use this to draw new actions for the agent with a newly defined action-generating function

$$\mathcal{A}^i_{\mathsf{t}+1} = \Pi^i_{\mathsf{t}+1}(\mathcal{S}_{\mathsf{t}+1}, \mathcal{Z}_{\mathsf{t}+1})\,. \tag{13.2}$$

[1] This vector is intended to include parameters for measurement, policy specification and ultimately the learning algorithm as well.

From this point on we'll call $\pi(a, s)$ the 'policy' adoped by the agent. A Markov Decision Process (MDP) defines an algorithm in which the agent uses a single state measurement vector and its given policy π to draw actions $\mathcal{A}_{\mathsf{t}+1}$ at timestep $\mathsf{t}+1$. It then performs these actions in its environment, which we have previously formalised through defining the iteration $X_{\mathsf{t}+1} = \mathcal{F}_{\mathsf{t}+1}(X', Z_{\mathsf{t}}, \mathcal{A}_{\mathsf{t}+1}, \mathsf{t})$.

In order to assess the quality of an agent's actions, we might later attribute a reward value $\mathcal{R}_{\mathsf{t}}$ for actions that were taken at timestep t. Using a series of these rewards, a return value R can also be computed using a future discount factor γ like so

$$R = \sum_{\mathsf{t}=0}^{\infty} \gamma^{\mathsf{t}} \mathcal{R}_{\mathsf{t}} \,. \tag{13.3}$$

A state-value function V_π is defined as the expectation (under policy π) of return R, given state vector $\mathcal{S}_{\mathsf{t}} = s$, i.e.,

$$V_\pi(s) = \mathrm{E}_\pi(R|s) \,. \tag{13.4}$$

Similarly, an action-value function Q_π is defined as the expectation (again, under policy π) of return R, given state vector $\mathcal{S}_{\mathsf{t}} = s$ and action vector $\mathcal{A}_{\mathsf{t}} = a$, i.e.,

$$Q_\pi(s, a) = \mathrm{E}_\pi(R|s, a) \,. \tag{13.5}$$

Follow-up this bit with the model-based approach that we're going to take in this book.

- Talk through value and policy learning - in this book we will be doing the value learning with our generalised stochastic model and then the policy learning bit is more nuanced.
- The value learning can be facilitated in software using a predictive model which is able to roll forecast rewards forward in time in a Monte Carlo fashion up to a window from a certain point given an input prior distribution of policies.
- This input prior distribution of policies can itself be optimised by maximising expected utility in a Bayesian design framework!

Bibliography