5

Optimising interactions with any system

Concept. To design and build software which enables the optimisation of automated control objectives over stochastic phenomena of any kind. The theory in this chapter will overlap significantly with that of Reinforcement Learning (RL), however, in contrast to more standard RL approaches, we shall be relying on all of the work from previous parts of this book to help agents characterise, measure and learn from their environment. The software which implements our generalised control optimisation algorithm will be implemented as an extension to the learnadex. For the mathematically-inclined, this chapter will cover how we formalise model-based automated control optimisation within the frameworks that we have already introduced in this book. For the programmers, the public Git repository for the code described in this chapter can be found here: https://github.com/worldsoop/worldsoop.

5.1 Formalising general interactions

Let's start by considering how we might adapt the mathematical formalism we have been using so far to be able to take actions which can manipulate the state at each timestep. Using the mathematical notation that we inherited from the stochadex, we may extend the formula for updating the state history matrix $X_{0:\mathtt{t}} \to X_{0:\mathtt{t}+1}$ to include a new layer of possible interactions which is facilitated by a new vector-valued 'take action' function $G_{\mathtt{t}}$. In doing so we shall be defining the domain of an acting entity in the stochastic process environment — which we shall hereafter refer to as simply the 'agent'.

During a timestep over which actions are performed by the agent, the stochadex state update formula can be extended to include interactions by composition with the original state update function like so

$$X^i_{\mathtt{t}+1} = G^i_{\mathtt{t}+1}[F_{\mathtt{t}+1}(X_{0:\mathtt{t}}, z, \mathtt{t}), A_{\mathtt{t}+1}] = \mathcal{F}^i_{\mathtt{t}+1}(X_{0:\mathtt{t}}, z, A_{\mathtt{t}+1}, \mathtt{t})\,, \tag{5.1}$$

where we have also introduced the concept of the 'actions' performed A_{t+1} on the system; some vector of parameters which define what actions are taken at timestep $t+1$. The code for the new iteration formula would look something like Fig. 5.1.

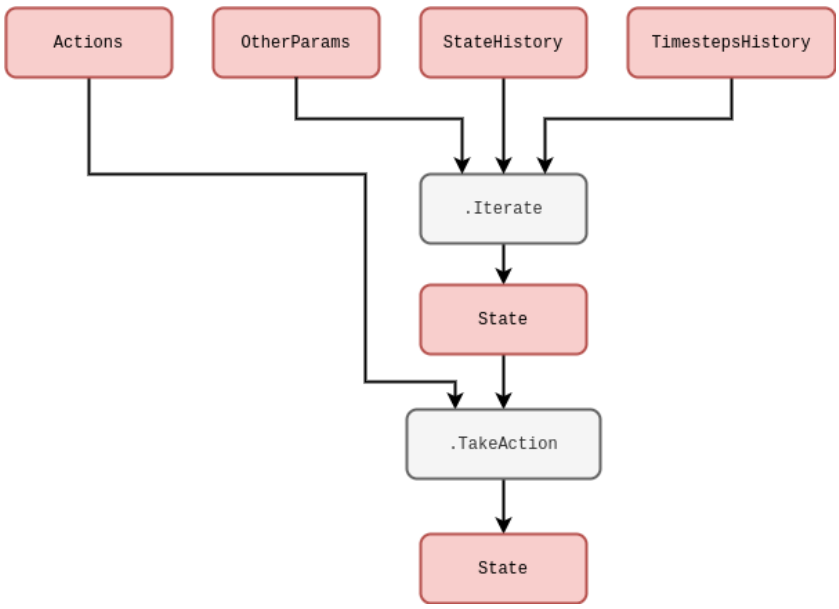

Figure 5.1: Code schematic of Eq. (5.1).

So far, Eq. (5.1) on its own will allow the agent to take actions that are scheduled up front through some fixed process or perhaps through user interaction via a game interface. So what's next? In order to start creating algorithms which will act on the system state for us, we need to develop a formalism which 'closes the loop' by feeding information back from the stochastic process to the agent's decision-making algorithm.

If we use $A_{0:t+1}$ a referring to the matrix of historically-taken actions which up to time $t+1$, we can build up a more generalised, non-Markovian picture of automated interactions with the system which matches the notation we are already using for $X_{0:t+1}$. Let us now define a Non-Markovian Decision Process (NMDP) as a probabilistic model which draws an actions matrix $A_{0:t+1} = A$ from a 'policy' distribution $\Pi_{(t+1)t}(A|X,\theta)$ given $X_{0:t} = X$ and a new vector of parameters which fully specify the automated interations. Using the probabilistic notation from the previous part of the book, the joint probability that $X_{0:t+1} = X$ and $A_{0:t+1} = A$ at time $t+1$ is

$$P_{t+1}(X, A|z, \theta) = P_t(X'|z, \theta)\, \Pi_{(t+1)t}(A|X', \theta) P_{(t+1)t}(x|X', z, A)\,, \tag{5.2}$$

where we recall that $P_{(t+1)t}(x|X', z, A)$ is the conditional probability of $X_{t+1} = x$ given $X_{0:t} = X'$ and z that we have encountered before, but it now requires $A_{0:t+1} = A$ as another given input. We have illustrated Eq. (5.1) and how it relates to the policy distribution of Eq. (5.2) with a new graph representation in Fig. 5.2.

For additional clarity, let's take a moment to think about what $\Pi_{(t+1)t}(A|X,\theta)$ represents and how generally descriptive it can be. If an agent is acting under and entirely deterministic policy,

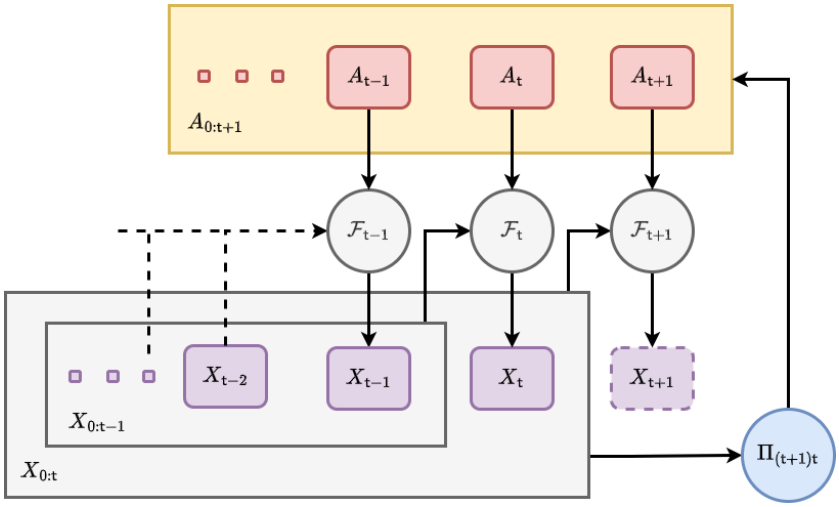

Figure 5.2: Graph representation of Eq. (5.1) with the policy distribution of Eq. (5.2).

then the policy distribution may be simplified to a direct function mapping which is parameterised by θ. At the other extreme, the distribution may also describe a fully stochastic policy where actions are drawn randomly in time. If we combine this consideration of noise with the observation that policies described by a distribution $\Pi_{(t+1)t}(A|X,\theta)$ permit a memory of past actions and states, it's easy to see that this structure can be used in a wide variety of different use cases.

By marginalising over Eq. (5.2) we find an updated probabilistic iteration formula for the stochastic process state which now takes the influence of agent actions into account

$$P_{t+1}(X|z,\theta) = \int_{\Xi_{t+1}} \mathrm{d}A\, P_t(X'|z,\theta)\, \Pi_{(t+1)t}(A|X',\theta) P_{(t+1)t}(x|X',z,A)\,. \tag{5.3}$$

This relationship will be very useful in the last part of this book when we begin to look at optimising control algorithms.

What are the main categories of action which are possible in the rows of A? Since the NMDP described by $\Pi_{(t+1)t}(A|X',\theta)$ is just another form of stochastic process, the main categories of action will fall into the same as those we covered in defining the stochadex formalism. The first, and perhaps most obvious, category would probably where the actions are defined in a continuous space and are continuously applied on every timestep. Some examples of these 'continuously-acting' decision processes include controlling the temperature of chemical reactions [1] (such as those in a brewery), spacecraft control [2] and guidance systems, as well as the driving of autonomous vehicles [3]. Within a kind of subset of the continuously-acting category; we can also find the event-based acting decision processes (where actions are not necessarily taken every timestep), e.g. controlling traffic through signal timings [4], managing disease spread through treatment intervals [5] and automated trading on stock markets [6].

Many of the examples we have given above have continuous action spaces, but we might also consider classes of decision processes where actions are defined discretely. Examples of these include the famous multi-armed bandit problem [7] (like choosing between website layouts for E-commerce [8]),

managing a sports team through player substitutions, sensor measurement scheduling [9] and the sequential design prioritisation of large-scale scientific experiments [10].

5.2 States, actions and attributing rewards

In the previous parts of this book we laid out the concept for a generalised framework to simulate and learn stochastic phenomena continually as data is received. Given that we have also introduced a framework for the automated control of these phenomena, we have all the ingredients we need to create optimal decision-making algorithms. The key question to answer then, is: *optimal with respect to what objective?*

The objective of an automated control algorithm could take many forms depending on the specific context. Since there is no loss in generality in doing so, it seems natural to follow the naming convention used by Markov Decision Processes (MDP) [11, 12] by referring to the objective outcome of an action at a particular point in time as having a 'reward' value r. Since the relationship between reward, actions and states may be stochastic, it makes sense to relate the reward outcome r given a state history X and action history A at timestep $\mathsf{t}+1$ through the probability distribution $P_{\mathsf{t}+1}(r|X,A)$. Hence, generally, this reward signal is non-Markovian — as is the case in many real-world problems [13].

We can use the reward probability distribution to derive a joint distribution over both state history X' and reward r at timestep $\mathsf{t}+1$ like so

$$P_{(\mathsf{t}+1)\mathsf{t}}(r,x'|X,z,\theta) = P_{\mathsf{t}+1}(r|X',A)\Pi_{(\mathsf{t}+1)\mathsf{t}}(A|X,\theta)P_{(\mathsf{t}+1)\mathsf{t}}(x'|X,z,A)\,. \tag{5.4}$$

In this expression, let's recall that we are using the policy distribution $\Pi_{(\mathsf{t}+1)\mathsf{t}}(A|X,\theta)$ for agent interactions and the fundamental state update conditional probability for the underlying stochastic process $P_{(\mathsf{t}+1)\mathsf{t}}(x'|X,z,A)$.

Note that in most use cases, the state of real-world phenomena cannot be measured perfectly. So to enable any agent trained on simulated phenomena to potentially act in the real world, we will need to include a measurement process as part of the information retrieval step. This is the part where we can leverage our work in a previous chapter which develops an online learning system for stochastic process models. But we're jumping ahead with this thinking and will return to this point later on.

Using Eq. (5.4), we can now define a 'state value function' V_{t} at timestep t which is the expected γ-discounted future reward given the current state history X and the other parameters like this[1]

$$\begin{aligned} V_{\mathsf{t}}(X,z,\theta) &= \mathrm{E}_{\mathsf{t}}(\mathsf{Discounted\ Return}|X,z,\theta) \\ &= \sum_{\mathsf{t}'=\mathsf{t}}^{\infty}\int_{\omega_{\mathsf{t}'+1}} \mathrm{d}^n x' \int_{\rho_{\mathsf{t}'+1}} \mathrm{d}r\, r\, \gamma^{\mathsf{t}'-\mathsf{t}} \prod_{\mathsf{t}''=\mathsf{t}}^{\mathsf{t}'} P_{(\mathsf{t}''+1)\mathsf{t}''}(r,x'|X,z,\theta)\,, \end{aligned} \tag{5.5}$$

[1] The discount factor in continuous time could also be explicitly dependent on the stepsize such that we would replace the discount factor in Eq. (5.5) with

$$\gamma^{\mathsf{t}'-\mathsf{t}} \longrightarrow \frac{1}{\gamma[\delta t(\mathsf{t}+1)]}\prod_{\mathsf{t}''=\mathsf{t}}^{\mathsf{t}'}\gamma[\delta t(\mathsf{t}''+1)]\,.$$

where $0 < \gamma < 1$. The idea behind this discount factor γ is to decrease the contribution of rewards to the optimisation objective (often called the 'expected discounted return' in RL) more and more as the prediction increases into the future. Note also that the state value function is inherently recursively defined, such that

$$V_{\mathtt{t}}(X, z, \theta) = \int_{\omega_{\mathtt{t}+1}} \mathrm{d}^n x \int_{\rho_{\mathtt{t}+1}} \mathrm{d}r \, P_{(\mathtt{t}+1)\mathtt{t}}(r, x'|X, z, \theta) \big[r + \gamma V_{\mathtt{t}+1}(X', z, \theta) \big] \,, \tag{5.6}$$

and the optimal θ can hence be derived from

$$\theta^*_{\mathtt{t}}(X, z) = \underset{\theta}{\mathrm{argmax}} \big[V_{\mathtt{t}}(X, z, \theta) \big] \,. \tag{5.7}$$

By deriving the optimal policy in terms of the parameters $\theta^*_{\mathtt{t}}(X, z)$, the optimal state value function and policy distribution can therefore be derived from

$$V^*_{\mathtt{t}}(X, z) = V_{\mathtt{t}}[X, z, \theta^*_{\mathtt{t}}(X, z)] \tag{5.8}$$

$$\Pi^*_{(\mathtt{t}+1)\mathtt{t}}(A|X, z) = \Pi_{(\mathtt{t}+1)\mathtt{t}}[A|X, \theta^*_{\mathtt{t}}(X, z)] \,. \tag{5.9}$$

Note that the type of decision process optimisation which we have introduced above differs from standard RL methodology. In the more conventional 'model-free' RL approaches, the state-action value function

$$Q_{\mathtt{t}}(X, A, z) = \mathrm{E}_{\mathtt{t}}(\mathsf{Discounted\ Return}|X, A, z) \,, \tag{5.10}$$

would be used to evaluate the optimal policy instead of the state value function $V_{\mathtt{t}}(X, z, \theta)$ that we are using above. We are able to use the latter here because the simulation model gives us explicit knowledge of the $P_{(\mathtt{t}+1)\mathtt{t}}(x'|X, z, A)$ distribution which is utilised by Eq. (5.4). When this model is not known, the state-action value function $Q_{\mathtt{t}}(X, A, z)$ must be learned explicitly through sample estimation from the measured state and experienced outcomes of actions taken by the agent.

When an agent takes an action to measure the state of the system (or when it is given measurements without needing to take action) there will typically be some uncertainty in how the history of measured real-world data Y maps to the latent states of the system X and its parameters z at time $\mathtt{t}+1$. It is natural, then, to represent this uncertainty with a posterior probability distribution $\mathcal{P}_{\mathtt{t}+1}(X, z|Y)$ as we did in the previous chapters of this book.

5.3 Algorithm designs

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

- Introduce broad concept of dynamic programming — partitioning a optimal global control into smaller optimal control segments/iterations.
- Talk about the utility of the model-based online learning approach in the case of partially observed systems [14].
- Look into the overlaps with this approach and Thompson sampling for exploration — discuss here.

- Looking at a stochastic policy iteration algorithm here combined with Monte Carlo rollouts.
- 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 discounted utility in a Bayesian design framework. Draw parallels.

5.4 Software design

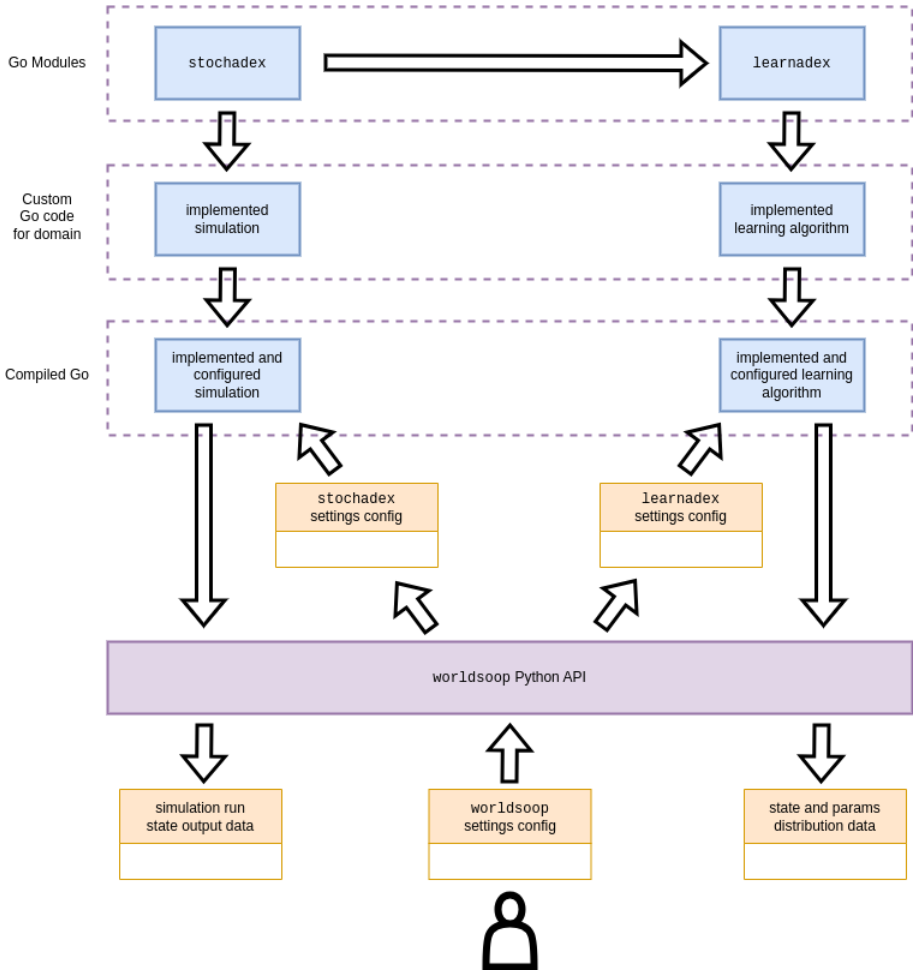

Figure 5.3: Diagram illustrating the high-level layer inter-dependencies of the Python API.

Bibliography

[1] C. Beeler, S. G. Subramanian, K. Sprague, N. Chatti, C. Bellinger, M. Shahen, N. Paquin, M. Baula, A. Dawit, Z. Yang *et al.*, "Chemgymrl: An interactive framework for reinforcement learning for digital chemistry," *arXiv preprint arXiv:2305.14177*, 2023.

[2] M. Tipaldi, R. Iervolino, and P. R. Massenio, "Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges," *Annual Reviews in Control*, 2022.

[3] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[4] D. Garg, M. Chli, and G. Vogiatzis, "Deep reinforcement learning for autonomous traffic light control," in *2018 3rd IEEE international conference on intelligent transportation engineering (ICITE)*. IEEE, 2018, pp. 214–218.

[5] A. Q. Ohi, M. Mridha, M. M. Monowar, and M. A. Hamid, "Exploring optimal control of epidemic spread using reinforcement learning," *Scientific reports*, vol. 10, no. 1, p. 22106, 2020.

[6] T. L. Meng and M. Khushi, "Reinforcement learning in financial markets," *Data*, vol. 4, no. 3, p. 110, 2019.

[7] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[8] Y. Liu and L. Li, "A map of bandits for e-commerce," *arXiv preprint arXiv:2107.00680*, 2021.

[9] A. S. Leong, A. Ramaswamy, D. E. Quevedo, H. Karl, and L. Shi, "Deep reinforcement learning for wireless sensor scheduling in cyber–physical systems," *Automatica*, vol. 113, p. 108759, 2020.

[10] T. Blau, E. V. Bonilla, I. Chades, and A. Dezfouli, "Optimizing sequential experimental design with deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 2107–2128.

[11] D. P. Bertsekas *et al.*, "Dynamic programming and optimal control 3rd edition, volume ii," *Belmont, MA: Athena Scientific*, vol. 1, 2011.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[13] M. Gaon and R. Brafman, "Reinforcement learning with non-markovian rewards," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 3980–3987.

[14] K. J. Åström, "Optimal control of markov processes with incomplete state information i," *Journal of mathematical analysis and applications*, vol. 10, pp. 174–205, 1965.