Pretty darn good control: when are approximate solutions better than approximate models*

Abstract

The text of your abstract. 150 – 250 words.

Introduction

Figures brainstorm

Figure 1: 1-D and 3-D model conceptual figure. something about the objective / decision Figure 2: Stability / multistability in the 1D and 3D models. state space + time views

Figure 3: The 1-D optimal management solution. 'constant escapement' intuition etc

Results figures: - timeseries example of management under the RL policy. probably compare to managing under 1-D solution / rule-of-thumb methods - state-space view of management doughnut - visualization / encapsulation of the policy heatmaps, slices, policy vs position along ellipse - reward plot over time (comparing methods)

Fisheries: approximate models and approximate solutions

Reinforcement learning

Reinforcement learning (RL) is, in a nutshell, a way of approaching *control problems* through machine learning. An RL algorithm can be conceptually separated into two parts: an *agent*, and an *environment* which the agent can interact with. That is, the agent may act on the environment and thus change its state, while the environment gives a *reward* to the agent in return (see Fig.

*This material is based upon work supported by the National Science Foundation under Grant No. DBI-1942280.

??). The rewards encode the agent's goal. The main part of an RL algorithm is then to progressively improve the agent's *policy*, in order to maximize the cumulative reward received. This is done by aggregating experience and learning from it.

For our use case, the environment will be a computational model of the population dynamics of a fishery, with the environment state being a vector of all the fish populations, $S = (V_1, V_2, H)$. At each time step, the agent harvests one of the populations, V_1. This changes the state as

$$(V_1,\ V_2,\ H) \mapsto ((1-q)V_1,\ V_2,\ H),$$

where q is a fishing *quota* set by the agent. This secures a reward of qV_1. Afterwards, the environment undergoes a timestep under its natural dynamics given by (**??**).

Mathematical framework for RL

Mathematically, RL may be formulated using a discrete time *partially observable Markov decision process (POMDP)*. This formalization is rather flexible and allows one, e.g., to account for situations where the agent may not fully observe the environment state, or where the only observations available to the agent are certain functions of the underlying state. For the sake of clarity, we will present here only the class of POMDPs which are relevant to our work: *fully observable MDPs with a trivial emission function* (FMDPs for short). An FMDP may be defined by the following data:

- $\mathcal{S}$: *state space*, the set of states of the environment,
- $\mathcal{A}$: *action space*, the set of actions which the agent may choose from,
- $T(s_{t+1}|s_t, a_t)$: *transition operator*, a conditional distribution which describes the dynamics of the system,
- $r(s_t, a_t)$: *reward function*, the reward obtained after performing action $a_t \in \mathcal{A}$ in state s_t,
- $d(s_0)$: *initial state distribution*, the initial state of the environment is sampled from this distribution,
- $\gamma \in [0, 1]$: *discount factor*.

At a time t, the FMDP agent observes the full state s_t of the environment and chooses an action based on this observation according to a *policy function* $\pi(a_t|s_t)$. In return, it receives a discounted reward $\gamma^t r(a_t, s_t)$. The discount factor helps regularize the agent, helping the optimization algorithm find solutions which pay off within a timescale of $t \sim \log(\gamma^{-1})^{-1}$.

With any fixed policy function, the agent will traverse a path τ =

$(s_0,\ a_0,\ s_1,\ a_1\ \ldots,\ s_{t_{\text{fin.}}})$ sampled randomly from the distribution

$$p_\pi(\tau) = d(s_0) \prod_{t=0}^{t_{\text{fin.}}-1} \pi(a_t|s_t) T(s_{t+1}|s_t, a_t).$$

Reinforcement learning seeks to optimize π such that the expected rewards are maximal,

$$\pi^* = \text{argmax}\ \mathbb{E}_{\tau\sim p_\pi}[R(\tau)],$$

where

$$R(\tau) = \sum_{t=0}^{t_{\text{fin.}}-1} \gamma^t r(a_t, s_t)$$

is the cumulative reward of path τ. The function $J(\pi) := \mathbb{E}_{\tau\sim p_\pi}[R(\tau)]$ is called the *value function*.

Deep Reinforcement Learning

The policy function π often lives in a high- or infinite-dimensional space. This makes it unfeasible to directly optimize π. In practice, an alternative approach is used: π is optimized over a much lower-dimensional parametrized family of functions. Deep reinforcement learning uses this strategy, focusing on function families parametrized by neural networks. (See Fig. **??**.)

Acknowledgements

The title of this piece references a mathematical biology workshop at NIMBioS organized by Paul Armsworth, Alan Hastings, Megan Donahue, and Carl Towes in 2011 that first posed the question addressed here.

References