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 0: contrast 1D vs 3D strategies (see commented out diagram).

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

RL figures: - schematic of RL (a-la previous marcus paper Fig 1) - Neural network optimization figure

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

Here we review classical strategies for sustainable fishery management and we provide a birds-eye view of the alternative approach we propose. A summary of the contrast between the two strategies is given in Fig. **??**.

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

Classical approaches to sustainable fisheries

There are several strategies that have been used to manage fisheries: *escapement*, *maximum sustainable yield (MSY)*, *total allowable catch (TAC)*, among others. We collectively refere to these as *classical*, and will compare their performance to RL-based management strategies. As shown in Fig. **??**, classical strategies have the common aspect of reducing the complex dynamics of the fishery ecosystem to a single equation governing the harvested population (say, F). A common example is using a logistic growth equation,

$$F_{t+1} - F_t = rF_t(1 - F_t/K) - h_t =: L(F_t) - h_t,$$

where the interaction between F and its environment is summarized to two parameters, the growth rate r, and the carrying capacity K. In the equation above, h_t is the *harvest* at timestep t. The goal is to choose the harvest policy $h : F_t \mapsto h_t$ such that long-term profits are maximized.

An advantage of one dimensional approaches is that the optimal policy is often known exactly, and, moreover, is intuitive. For example, in the logistic equation pointed out above, the maximal sustainable yield of the system is attained at $F = F_{MSY} := K/2$. The optimizer is an *escapement* policy:

$$h_t = \begin{cases} F_t - K/2, & \text{if, } F_t > K/2 \\ 0, & \text{else.} \end{cases}$$

This corresponds to keeping the system at its optimal growth rate as much as possible.

Escapement policies, or more generally *bang bang* policies, tend to be the optimal solution for these types of control problems. A drawback of these solutions, in the fishery context, is the presence of several timesteps with zero harvest which can arise. To mend this, certain suboptimal solutions have been constructed for fishery management.

One ubiquitous solution is simply called maximum sustainable yield (MSY). It consists on letting $h(F) = rF/2$, so that

$$h(F_{MSY}) = L(F_{MSY}) = rK/4.$$

That is, at the MSY biomass, the logistic growth of F is cancelled exactly by the harvest.

The MSY rule fixes the drawback in the escapement policy by having $h(F) > 0$ for all $F > 0$. It, however, has its own drawbacks. It is particularly sensitive to misestimates of the parameter r, as we will discuss in Sec. **??**. Due to this, similar but more conservative policies have been used.

Total allowable catch (TAC) is one such policy. It consists on reducing the inclination of the line defined by $h(F)$, for example using $h(F) = 0.8 \times r/2$.

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. **??**). 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 $\tau = (s_0,\ a_0,\ s_1,\ a_1\ \ldots,\ s_{t_{\text{fin.}}})$ 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 even 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. **??**.)

Since our state and observation spaces are continuous, we will focus on deep reinforcement learning throughout this paper. Specifically, we parametrize π using a neural network with two hidden layers of 64 neurons.

We use the *proximal policy optimization (PPO)* algorithm to optimize π. Within the RL literature there is a wealth of algorithms from which to choose from, each with its pros and cons. Here we have focused on a single one of these for the sake of particularity—to draw a clear comparison between the RL-based and the classical fishery management approaches. In practice, further improvements can be expected by a careful selection of the optimization algorithm.

Model-free reinforcement learning

Within control theory, the usual setup is one where we use as much information from the model as possible in order to derive an optimal solution. **(Wax poetic a bit about Bellmann eq. approaches here?)**

The classical sustainable fishery management approaches summarized in Sec. are model-based controls. As we saw in that section, these controls may run

into trouble in the case where there are inaccuracies in the model parameter estimates.

More generally, there are many situations in which the exact model of the system is not known or not tractable. This is a standard situation in ecology: mathematical models capture the most prominent aspects of the ecosystem's dynamics, while ignoring or summarizing most of its complexity. In this case, it is clear, model-based controls run a grave danger of mismanaging the system.

Reinforcement learning is, on the other hand, typically a model-free approach to control theory.[1] While the model is certainly used to generate training data, it is not directly used by the optimization algorithm. This provides more flexibility to use model-free RL in instances where the model of the system is not accurately known. In fact, it has been shown to generally be preferable to use model-free RL in such instances.

Mention CL here? Where solutions to other related problems in the curriculum are used to build solutions to the target problem.

This context provides a motivation for this paper. Indeed, models for ecosystem dynamics are only ever approximate and incomplete descriptions of reality. This way, it is plausible that model-free RL controls outperform currently used model-based controls in ecological management problems.

Results

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

[1] There are, however, approaches to perform model-based reinforcement learning. While we will not focus on these in this paper they are discussed in [**?**, **?**].