Models, States, Actions, Dynamics, Rewards, Policies, and Value Functions

Chris Atkeson

Models

In AI we often have religious wars about models: whether an agent should have them, what form they should take, and what they should be used for.

I will take a Omnistic approach.

A model is a representation of a system.

The representation could be mathematical, pictorial, physical, metaphorical, analogical, …

It is a good idea for an agent to have many types of models (representations).

The questions that will be asked determine which types of models are most useful.

So, a good attitude is that a model is something other than “the system” that can be used to provide information about “the system”, and we should have many types of models.

States

The state of the system is all the information needed at any time to predict the future of the system. If more currently available information would improve the prediction, then you don't have a state. You have some features. Folks in AI and Machine Learning are often sloppy about this. For example, a single image is not a state. It is missing information about velocity.

For mechanical systems, the state vector usually consists of the position (and orientation) and velocity (and angular velocity) of all the parts. **State** = [pos, vel]

Engineers use **x** for state and AI folks use **s**. I was trained as an engineer.

The derivative of the state (vector) for mechanical systems is made up of the derivatives of the components of **x** = [position, velocity] so $\dot{\mathbf{x}}$ = [velocity, acceleration]

Actions

Actions are things that can be set arbitrarily at any instant, such as the command to a motor or valve or power amplifier.

For mechanical systems, ignoring actuator dynamics, the action vector usually consists of the forces and torques that the actuators can produce.

Engineers use **u** for action, and AI folks use **a**.

A **control law** (engineering) or **policy** (AI) is a function from a state to an action:

$$\mathbf{u} = \mathbf{U}(\mathbf{x}) \qquad \mathbf{a} = \pi(\mathbf{s}) \qquad \mathbf{u} = \pi(\mathbf{x})$$

Control laws/Policies

A **control law** (engineering) or **policy** (AI) is a function from a state to an action:

$$\mathbf{u} = \mathbf{U}(\mathbf{x}) \qquad \mathbf{a} = \pi(\mathbf{s}) \qquad \mathbf{u} = \pi(\mathbf{x})$$

Dynamics

Continuous time dynamics generate the derivative of the state vector

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

Discrete time dynamics generate the state at the next time instant

$$\mathbf{x}_{i+1} = \mathbf{F}(\mathbf{x}_i, \mathbf{u}_i)$$

Types of tasks

- Regulator tasks: want to stay at x_d
- Trajectory tasks: go from A to B in time T, or attain goal set G
- Periodic tasks: cyclic behavior such as walking

How do you say what you want? This is really hard

King Midas story

Desired state (goal): $\mathbf{x}_d$

Desired volume (goal set): $g(\mathbf{x}) = 1$

Terminal cost $L(\mathbf{x}_N)$

One step cost $L_i(\mathbf{x}_i, \mathbf{u}_i)$ Total cost: $L_N(\mathbf{x}_N) + \sum_i L_i(\mathbf{x}_i, \mathbf{u}_i)$

The cost of a policy:

$$C(\pi, \mathbf{x}_0) = L_N(\mathbf{x}_N) + \sum_i L_i(\mathbf{x}_i, \pi(\mathbf{x}_i))$$

Costs vs. rewards joke.

Value Functions

Given a policy $\mathbf{u} = \pi(\mathbf{x})$, we can calculate the lifetime cost of starting in state $\mathbf{x}$

$V^{\pi}(\mathbf{x}) = \sum L_i(\mathbf{x}_i,\pi(\mathbf{x}_i)) + L_N(\mathbf{x}_N),\ \ \mathbf{x}_{i+1} = \mathbf{F}(\mathbf{x}_i,\pi(\mathbf{x}_i)),\ \mathbf{x}_0 = \mathbf{x}$

$V^{\pi}(\mathbf{x})$ is the value function for the policy π.
The value function for the optimal policy satisfies Bellman's equation:

$V^*_i(\mathbf{x}) = \min_{\mathbf{u}}(L_i(\mathbf{x},\mathbf{u}) + V^*_{i+1}(\mathbf{F}_i(\mathbf{x},\mathbf{u})))$, where the solution $\mathbf{u}$ is the optimal action.

For infinite lifetimes, V(x) becomes independent of time:

$V^*(\mathbf{x}) = \min_{\mathbf{u}}(L(\mathbf{x},\mathbf{u}) + V^*(\mathbf{F}(\mathbf{x},\mathbf{u}))),\quad \pi^*(\mathbf{x}) = \operatorname{argmin}_{\mathbf{u}}(L(\mathbf{x},\mathbf{u}) + V^*(\mathbf{F}(\mathbf{x},\mathbf{u})))$

A useful concept is a Q function, which is the value of one free action, then $\pi^*(\mathbf{x})$

$Q(\mathbf{x},\mathbf{u}) = L(\mathbf{x},\mathbf{u}) + V^*(\mathbf{F}(\mathbf{x},\mathbf{u}))$ so $V(x) = \min_{\mathbf{u}} Q(\mathbf{x},\mathbf{u})$ and $\pi^*(\mathbf{x}) = \operatorname{argmin}_{\mathbf{u}} Q(\mathbf{x},\mathbf{u})$

How can we represent a control law/policy or a value function?

Formula (analytic): u = -(c1*position + c2*velocity)

A lookup table

A polynomial: u = c1, u = -(c1*position + c2*velocity)

Arbitrary functions (basis functions), each with a weight

$$\Sigma_i(w_i * g_i(x,x\_dot))$$

“Neural” networks

A bunch of examples

Anything else you can think of …

How do you design or learn a feedback controller (policy)?

- Parameterize the policy: $\pi(\mathbf{x}, \mathbf{p})$
- Trial and error: guess a **p**, try it out, score it. If it is better keep it. Otherwise try a new **p**.
- Automate trial and error: Use your favorite function optimizer to optimize **p**. Each function evaluation is a simulation or actual execution of a controller for some period of time.
- Learning: Use your favorite function optimizer to optimize **p**. Each function evaluation is a simulation or actual execution of a controller for some period of time.
- Dynamic programming provides an efficient way to compute value functions and corresponding policies.
- Other things we will discuss later.