Last Time:

- How to drive a car

Today:

- Data-driven / model-free "behavioral" control
- Behavior Cloning

What if we don't have a model?

- Given input-output data, standard approach is to fit a model (system ID)
- Let's assume we have data $u_{1:T}, y_{1:T}$ and we fit a linear state-space model:

$$x_{n+1} = Ax_n + Bu_n$$

$$y_n = Cx_n$$

- For an arbitrary change of coordinates $x = Q\tilde{x}$:

$$\tilde{x}_{n+1} = \underbrace{Q^{-1}AQ}_{\tilde{A}}\tilde{x} + \underbrace{Q^{-1}B}_{\tilde{B}}u_n$$

$$y_n = \underbrace{CQ}_{\tilde{C}}\tilde{x}_n$$

$\Rightarrow$ State definition is arbitrary!

* Can we just skip the state entirely?

A Different Take on Linear Systems:

- What does it mean to be linear?

$$f(ax_1 + bx_2) = af(x_1) + bf(x_2)$$

(vectors: x_1, x_2; scalars: a, b)

- "Superposition" $\Rightarrow$ linear combination of inputs $\downarrow$ " " " outputs

- For a linear dynamical systems, superposition holds for trajectories!

$$\begin{bmatrix} y_1 \\ u_1 \\ y_2 \\ u_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} C & & & \\ CI & & & \\ CA & CB & & \\ 0 & 0 & I & \\ CA^2 & CAB & CB & \cdots \end{bmatrix} \begin{bmatrix} x_1 \\ u_1 \\ u_2 \\ \vdots \end{bmatrix}$$

z $\quad$ F $\quad$ v

$$\Rightarrow F(av^1 + bv^2) = aFv^1 + bFv^2 = az^1 + bz^2$$

- Given enough trajectory data, we can span the space of all possible input-output trajectories of a given length:

$$z = az^1 + bz^2 + cz^3 + \cdots$$

$$= \begin{bmatrix} \vdots & \vdots & \vdots & \\ z^1 & z^2 & z^3 & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ \vdots \end{bmatrix}$$

H $\quad$ w

- We can now use H in place of a state-space model (A, B, C) to do prediction + control.

* Example

- State prediction for damped SHO:

$$\ddot{q} = -Kq - C\dot{q} \Rightarrow \underbrace{\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 1 \\ -K & -C \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} q \\ \dot{q} \end{bmatrix}}_{x}$$

$$z = \begin{bmatrix} x_1 \\ \vdots \\ x_{50} \end{bmatrix} = Hw$$

$$\underbrace{\min_w \frac{1}{2} \| EHw - x_1 \|_2^2}_{\text{least-squares projection}}, \quad E = [I \; 0 \; 0 \; \cdots]$$

- Closed-form solution:

$$w = (H^T E^T E H)^{-1} H^T E^T x_1, \quad z = Hw$$

- We can do the same thing with only $y = Cx$

- We can also generate new dynamically feasible trajectories by sampling a random vector and projecting onto the data manifold ("de-noising")

Behavior Cloning:

- Let's assume we have input-output data from a closed-loop system being controlled by an "expert policy"

$$z^i = \begin{bmatrix} y_1^i \\ y_2^i \\ \vdots \end{bmatrix} \Rightarrow H = \begin{bmatrix} \vdots & \vdots & \\ z^1 & z^2 & \cdots \\ \vdots & \vdots & \end{bmatrix}$$

- Given some history of observations $y_{1:m}$, we can predict the rest of the input-output trajectory

- We can get a feedback policy by taking u_{m+1} from the predicted z:

$$\min_w \frac{1}{2}\| YHw - y_{1:m} \|_2^2, \quad Y = \begin{bmatrix} I & 0 & \cdots \\ 0 & I & 0 \cdots \end{bmatrix}$$

Picks out $y_{1:m}$

$$u_{m+1} = UHw, \quad U = [0\ 0 \cdots I\ 0 \cdots 0]$$

* Example

- Clone an LQR policy from random demonstrations

Data-Driven MPC:

- We can use the data matrix as our "dynamics model" in an MPC controller.

- Note that we need data with "sufficient excitation". It can't be just closed-loop "on-policy" rollouts like in the prediction/cloning case.

* Safe bet: inject noise into the system during data collection

- Given a data matrix:

$$Z = \begin{bmatrix} y_1 \\ u_1 \\ y_2 \\ u_2 \\ \vdots \end{bmatrix} = Hw$$

- We can solve the following QP:

$$\min_{z,w} \ \frac{1}{2} z^T Q z + \frac{1}{2} \alpha w^T w \quad \leftarrow \text{regularizer on weights}$$

$$\text{s.t.} \quad z = Hw$$

$$y = Yz \quad \leftarrow \text{picks out first few outputs}$$

y: observations

$$\Rightarrow L = \frac{1}{2} z^T Q z + \frac{1}{2} \alpha w^T w + \lambda^T (Hw - z) + \mu^T (Yz - y)$$

$$\nabla L = 0 \Rightarrow \begin{bmatrix} Q & 0 & -I & Y^T \\ 0 & \alpha I & H^T & 0 \\ -I & H & 0 & 0 \\ Y & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ w \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ y \end{bmatrix}$$

- Note that this is an output feedback controller (uses y instead of x). There's no separate state estimator.

What about nonlinear systems?

- In the linear case, the data manifold is a (hyper-) plane / subspace

- For nonlinear systems, it is a more complex curved manifold

  ⇒ superposition doesn't hold anymore, but we can still define projection onto the manifold.

- Diffusion methods essentially learn this projection