Lecture 21 :

Dynamic Programming dim < 4 or 5 , $c()$, $f()$, x, u known

$$\hat{J}^*(x_i) \leftarrow \min_{u \in U} \left[c(x_i, u) + \hat{J}^*(f(x_i, u)) \right]$$

$$\pi^*(x_i) = \underset{u \in U}{\operatorname{argmin}} \left[c(x_i, u) + J^*(f(x_i, u)) \right]$$

Stability analysis via SOS $f()$ vector-valued poly

$f()$, x, u, known

find α s.t. $V_\alpha(x)$ is SOS

$$-\dot{V}_\alpha(x) = -\frac{\partial V_\alpha}{\partial x} f(x) \text{ is SOS}$$

Trajectory Optimization $c()$, $f()$, x, u known , local solutions

$$\underset{x_1, \ldots, x_N, u_0, \ldots, u_{N-1}}{\text{minimize}} \quad \sum_{n=0}^{N-1} c(x_n, u_n)\, dt$$

$$\text{s.t.} \quad x_{n+1} = x_n + f(x_n, u_n)\, dt$$

Pendulum DP

Airplane verification SOS

Rocket landing, walking / backflip robot traj opt

Flying a quadrotor in unknown env traj opt . LQR

A clarification

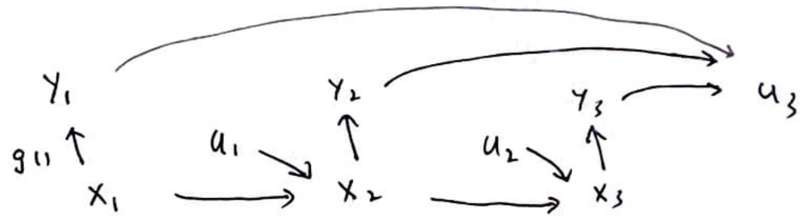
$$x = [x_{robot}, x_{world}]$$

Motion planning for cars inference other drivers motion

Sushi making ??? f, x, u, c

Markov decision process $\mathcal{M} = \{X, U, f(), c()\}$

Partially observed MDP $\mathcal{M} = \{X, U, Y, f(), g(), c()\}$



Estimate $f()$

"System identification"

"model learning"

Estimate $g^{-1}()$, estimate x

"state estimation"

Estimate what X (the space of x) should even be

latent space learning ?

Estimate $c()$ from demonstrations of an expert

"Imitation Learning"

$$\min_{\substack{x_0, \dots, x_N \\ u_0, \dots, u_N}} \sum_{i=0}^{N} c(x_i, u_i) \quad \text{s.t. } x_{t+1} = f(x_t, u_t)$$

non-linear programming

Given not much known in manipulation

options:

1. find prob domains where most things are known
2. assume some subset is known
3. tackle hardest, general problem

know dynamics → unknown dynamics → unknown dynamics

known state → known state → partially observed state

"Copy-demo" MIT 1970

Ishikawa Komuro lab, 2009

Dense Object Nets visual object descriptors CoRL 2018

one image → class of images description

key point affordances for category level manipulation

Imitation Learning

π_{expert}

↓

$D = \{T_0, T_1, \dots, T_m\}$ → $\pi_{learner}$

$T_i = ((y_0, u_0), (y_1, u_1), \dots, (y_{N_i}, u_{N_i}))$

1. Behavior Cloning

Fit from data

$u = \pi(y_{-T}, y_{-T+1}, \dots, y)$

2. Inverse Optimal Control / Inverse Reinforcement Learning

a. infer $c()$

b. optimize π

3. Generative Adversarial Imitation Learning

Make $\pi_{Learner}$

indistinguishable

from π_{Expert}

Contact Rich Manipulation Tasks

Solution Approaches

Model Based:

controllers with good generalization

provides insides

- hard to hybrid, unknown dynamics

Learning based:

model-based RL

model-free RL

Model-based RL

fit a model

generate samples

improve the policy

Motivation

Goal: controller for contact rich manipulation

Challenges: unknown dynamics

partially observable states

Overview

1. collect samples from robot $\{\tau_i^j\}$
2. fit time-varying linear-Gaussian dynamics
3. optimize local controllers using iLQG
4. Repeat 1-3 until convergence
5. train global policy by imitating local policies

$$
\begin{array}{ccc}
\text{collect samples from } p_i & \longrightarrow & \{\tau_i^j\} \\
\uparrow & & \downarrow \\
\text{optimize local controller } p_i & \longleftarrow & \text{train global policy } \pi_\theta \text{ to match local controllers } p_i
\end{array}
$$

$\tau_j = \{x_i, u_i\}_{i=1}^{T_j}$

iLQG with known dynamics

locally-optimal feedback control of nonlinear stochastic systems

$$\min_{\pi} E\left[h(x(T)) + \int_t^T \ell(\tau, x(\tau), \pi(\tau, x(\tau)))\, d\tau \right]$$

$$dx = f(x,u)\,dt + F(x,u)\,\underline{dw}$$

random variable?

iLQG

approach: $\xrightarrow{\text{convert to}}$ LQG, something we know how to solve

1. current trajectory $\bar{x}(t), \bar{u}(t)$

2. approximate costs and trajectories around current trajectory

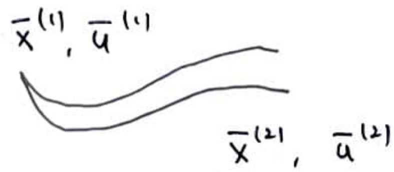
   quadratic approximation of costs

   linear approximation of dynamics

3. Solve linear quadratic problem to get local feedback controller $\delta u = I_k + L_k \delta x$

4. use controller to update current trajectory

5. Repeat 1-4 until convergence



Linear - Quadratic Approximation

Start with nominal trajectory $\bar{x}(t), \bar{u}(t)$

Discrete time dynamics $\bar{x}_{k+1} = \bar{x}_k + \Delta t \cdot f(\bar{x}_k, \bar{u}_k)$

Linearized Dynamics $\delta x_k \triangleq x_k - \bar{x}_k$, $\delta u_k = u_k - \bar{u}_k$

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k + \underbrace{\mathcal{C}_k(\delta u_k)\xi_k}_{\text{noise}}$$

Quadratic approximation to cost

$$\text{cost}_k = q_k + \delta x_k^T q_k + \frac{1}{2}\delta x_k^T Q_k \delta x_k + \delta u_k^T r_k + \frac{1}{2}\delta u_k^T R_k \delta u_k + \delta u_k^T P_k \delta x_k$$

Solving single LQG step

Very similar to LQR

Cost-to-go function is quadratic under any affine control law

$$\delta u = I_k + L_k \delta x \qquad V_k(\delta x) = S_k + \delta x^T s_k + \frac{1}{2}\delta x^T S_k \delta x$$

Can show via induction that

$$S_k = Q_k + A_k^T S_{k+1} A_k - G^T H^{-1} G$$

$$s_k = q_k + A_k^T s_{k+1} - G^T H^{-1} g$$

$$S_k = q_k + S_{k+1} + \frac{1}{2}\sum_i C_{i,k}^T S_{k+1} C_{i,k} - \frac{1}{2} g^T H^{-1} g$$

Extract optimal control law $\delta u = I_k + L_k \delta x$ by recursively minimizing cost to go

Bellman Equation

$$V_k(\delta x) = \text{immediate cost} + E[V_{k+1}(\text{next state})]$$

$$= q_k + \delta x^T (q_k + \frac{1}{2} Q_k \delta x) + \pi^T (r_k + \frac{1}{2} R_k \pi)$$

$$+ \pi^T P_k \delta x + E[V_{k+1}(A_k \delta x + B_k \pi + C_k \xi_k)]$$

$$V_k(\delta x) = q_k + S_{k+1} + \frac{1}{2}\sum_i C_i^T S_{k+1} C_i + \delta x^T (q_k + A_k^T s_{k+1})$$

$$+ \frac{1}{2}\delta x^T (Q_k + A_k^T S_{k+1} A_k)\delta x$$

$$+ \pi^T (g + G \delta x) + \frac{1}{2}\pi^T H \pi$$

9. G, H depend only on known running cost, dynamics and cost-to-go at $k+1$

$$g \triangleq r_k + B_k^T s_{k+1} + \Sigma_i C_{i,k}^T S_{k+1} C_{i,k}$$

$$G \triangleq P_k + B_k^T S_{k+1} A_k$$

$$H \triangleq R_k + B_k^T S_{k+1} B_k + \Sigma_i C_{i,k}^T S_{k+1} C_{i,k}$$

Minimized control law $\delta u = -H^{-1}(g + G\delta x)$

note other terms not related to π

But $H \succeq 0$ may not be true, no control constraints

note δu may be large step, need to be in approximate dynamics region

$$\nabla_{\delta u} = g + G\delta x$$

$$\delta u = -\varepsilon \hat{H}^{-1}(g + G\delta x)$$

$$\hat{H} > 0$$

can restrict δu to a region, distance small

Peg-in-hole

Cost function: distance of known points on rigidly grasped object to their desired location

$$r_e(\| P_t - P_t^* \|)$$

State vector

joint angles and velocities

cartesian velocity on manipulated object

$P_t - P_{target}$

previously commanded torque

command = joint torque

iLQG with unknown dynamics

Problem statement: extend iLQG to settings where we don't know the dynamics

Approach:

1. Iteratively refit locally linear models to the dynamics
2. With locally linear models in hand run iLQG as before
3. Run step 1-2 until convergence

run $P(u_t | x_t)$
on robot
collect $D = \{\tau_i\}$

$\{\tau_i\}$

next iteration

fit dynamics
$P(x_{t+1} | x_t, u_t)$

improve $P(u_t | x_t)$

Fit locally - linear dynamics

$$p(x_{t+1} | x_t, u_t) = \cancel{N} \mathcal{N}(A_t x_t + B_t u_t + c_t, N_t)$$

$$\{x_t, u_t, x_{t+1}\}$$

need to have a good priori to make this step works

Time varying → capture discontinuous dynamics ?

average the ~~dynamics~~ dynamics from both sides of a discontinuity

problem: can only work in a region around trajectory

Guided Policy Search with Local Models

run $p(u_t | x_t)$ on robot collect $D = \{\tau_i\}$

$\{\tau_i\}$

fit dynamics $p(x_{t+1} | x_t, u_t)$

train $\pi_\theta(u_t | o_t)$

improve $p(u_t | x_t)$

next iteration

local policies

controller 1

controller 2

controller 3

train a global function approximator

$$\pi_\theta(x_t) = u_t$$

↑ can be image