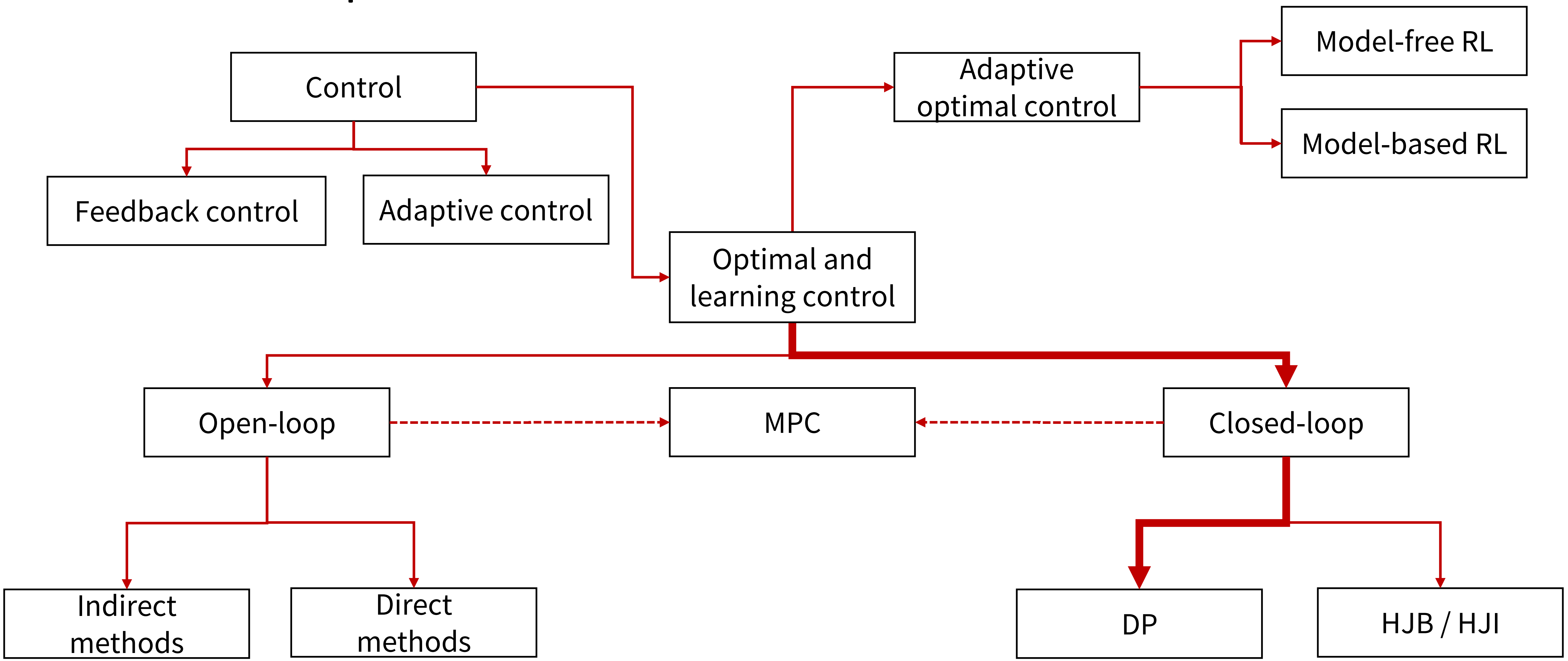
AA203
Optimal and Learning-based Control

Discrete LQR, stochastic DP, value iteration, policy iteration

Stanford University

Autonomous Systems Lab

Roadmap


Control
Feedback control
Adaptive control
Optimal and learning control
Adaptive optimal control
Model-free RL
Model-based RL
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI

Dynamic programming

- Model: $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, k), \quad \mathbf{u}_k \in U(\mathbf{x}_k)$
- Cost: $J(\mathbf{x}_0) = h_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} g(\mathbf{x}_k, \pi_k(\mathbf{x}_k), k)$

DP Algorithm: For every initial state $\mathbf{x}_0$, the optimal cost $J^*(\mathbf{x}_0)$ is equal to $J_0^*(\mathbf{x}_0)$, given by the last step of the following algorithm, which proceeds backward in time from stage $N-1$ to stage 0:

$$J_N^*(\mathbf{x}_N) = h_N(\mathbf{x}_N)$$

$$J_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_k \in U(\mathbf{x}_k)} g(\mathbf{x}_k, \mathbf{u}_k, k) + J_{k+1}^*\big(f(\mathbf{x}_k, \mathbf{u}_k, k)\big), \qquad k = 0, \dots, N-1$$

Furthermore, if $\mathbf{u}_k^* = \pi_k^*(\mathbf{x}_k)$ minimizes the right hand side of the above equation for each $\mathbf{x}_k$ and k, the policy $\{\pi_0^*, \pi_1^*, \dots, \pi_{N-1}^*\}$ is optimal

Discrete LQR

- Canonical application of dynamic programming for control
- One case where DP can be solved analytically (in general, DP algorithm must be performed numerically)

Discrete (Deterministic) LQR: select control inputs to minimize

$$J_0(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_N^T Q_N \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + 2\mathbf{x}_k^T S_k \mathbf{u}_k\right)$$

subject to the dynamics

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k, \qquad k \in \{0, 1, \ldots, N-1\}$$

assuming

$$Q_k = Q_k^T \succeq 0, \quad R_k = R_k^T \succ 0, \quad \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} \succeq 0 \quad \forall k$$

Discrete LQR

Many important extensions, some of which we'll cover later in this class

- Tracking LQR: $\mathbf{x}_k, \mathbf{u}_k$ represent small deviations ("errors") from a nominal trajectory (possibly with nonlinear dynamics)
- Cost with linear terms, affine dynamics: can consider today's analysis with augmented dynamics

$$\mathbf{y}_{k+1} = \begin{bmatrix} \mathbf{x}_{k+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A_k & c_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \mathbf{u}_k = \tilde{A}\mathbf{y}_k + \tilde{B}\mathbf{u}_k$$

Discrete LQR – trajectory optimization

Rewrite the minimization of

$$J_0(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_N^T Q_N \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1} \left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + 2\mathbf{x}_k^T S_k \mathbf{u}_k\right)$$

subject to dynamics

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k, \qquad k \in \{0, 1, \dots, N-1\}$$

as…

Discrete LQR – trajectory optimization

$$\min_{\mathbf{x}_k,\mathbf{u}_k} \quad \frac{1}{2}\begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_0\\ \mathbf{x}_1\\ \mathbf{u}_1\\ \vdots\\ \mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\\ \mathbf{x}_N\end{bmatrix}^T \begin{bmatrix}Q_0 & S_0 & & & & & & \\ S_0^T & R_0 & & & & & & \\ & & Q_1 & S_1 & & & & \\ & & S_1^T & R_1 & & & & \\ & & & & \ddots & & & \\ & & & & & Q_{N-1} & S_{N-1} & \\ & & & & & S_{N-1}^T & R_{N-1} & \\ & & & & & & & Q_N\end{bmatrix}\begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_0\\ \mathbf{x}_1\\ \mathbf{u}_1\\ \vdots\\ \mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\\ \mathbf{x}_N\end{bmatrix}$$

$$\text{s.t.} \quad \begin{bmatrix}-I & & & & & & \\ A_0 & B_0 & -I & & & & \\ & & A_1 & B_1 & -I & & \\ & & & & \ddots & & \\ & & & & A_{N-1} & B_{N-1} & -I\end{bmatrix}\begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_0\\ \mathbf{x}_1\\ \mathbf{u}_1\\ \mathbf{x}_2\\ \vdots\\ \mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\\ \mathbf{x}_N\end{bmatrix} + \begin{bmatrix}\mathbf{x}_0\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix} = \mathbf{0}$$

Discrete LQR – trajectory optimization

Defining suitable notation, this is

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\mathbf{z}^T W \mathbf{z} \\ \text{s.t.} \quad & C\mathbf{z} + \mathbf{d} = \mathbf{0} \end{aligned}$$

with solution from applying NOC (also SOC in this case, due to problem convexity):

$$\begin{bmatrix} \mathbf{z}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} W & C^T \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ -\mathbf{d} \end{bmatrix}$$

Discrete LQR – dynamic programming

First step:

$$J_N^*(\mathbf{x}_N) = \frac{1}{2} x_N^T Q_N x_N = \frac{1}{2} x_N^T P_N x_N$$

Proceeding backward in time:

$$J_{N-1}^*(\mathbf{x}_{N-1}) = \min_{\mathbf{u}_{N-1}} \frac{1}{2} \left(\begin{bmatrix} \mathbf{x}_{N-1} \\ \mathbf{u}_{N-1} \end{bmatrix}^T \begin{bmatrix} Q_{N-1} & S_{N-1} \\ S_{N-1}^T & R_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{N-1} \\ \mathbf{u}_{N-1} \end{bmatrix} + \mathbf{x}_N^T P_N \mathbf{x}_N \right)$$

$$= \min_{\mathbf{u}_{N-1}} \frac{1}{2} \left(\begin{bmatrix} \mathbf{x}_{N-1} \\ \mathbf{u}_{N-1} \end{bmatrix}^T \begin{bmatrix} Q_{N-1} & S_{N-1} \\ S_{N-1}^T & R_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{N-1} \\ \mathbf{u}_{N-1} \end{bmatrix} + \right.$$

$$\left. (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1})^T P_N (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}) \right)$$

Discrete LQR – dynamic programming

Unconstrained NOC:

$$
\begin{aligned}
\nabla_{u_{N-1}} J_{N-1}(\mathbf{x}_{N-1}) &= R_{N-1}\mathbf{u}_{N-1} + S_{N-1}^T \mathbf{x}_{N-1} + \\
&\qquad B_{N-1}^T P_N (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}) = \mathbf{0} \\
\implies \mathbf{u}_{N-1}^* &= -(R_{N-1} + B_{N-1}^T P_N B_{N-1})^{-1}(B_{N-1}^T P_N A_{N-1} + S_{N-1}^T)\mathbf{x}_{N-1} \\
&:= F_{N-1} x_{N-1}
\end{aligned}
$$

Note also that SOC hold:

$$
\nabla_{u_{N-1}}^2 J_{N-1}(\mathbf{x}_{N-1}) = R_{N-1} + B_{N-1}^T P_N B_{N-1} \succ 0
$$

Discrete LQR – dynamic programming

Plugging in the optimal policy:

$$
\begin{aligned}
J^*_{N-1}(\mathbf{x}_{N-1}) &= \frac{1}{2}\mathbf{x}_{N-1}^T \big(Q_{N-1} + A_{N-1}^T P_N A_{N-1} - \\
&\quad (A_{N-1}^T P_N B_{N-1} + S_{N-1})(R_{N-1} + B_{N-1}^T P_N B_{N-1})^{-1}(B_{N-1}^T P_N A_{N-1} + S_{N-1}^T)\big)\,\mathbf{x}_{N-1} \\
&:= \frac{1}{2}\mathbf{x}_{N-1}^T P_{N-1}\mathbf{x}_{N-1}
\end{aligned}
$$

Algebraic details aside:

- Cost-to-go (equivalently, "value function") is a quadratic function of the state at each step
- Optimal policy is a time-varying linear feedback policy

Discrete LQR – dynamic programming

Proceeding by induction, we derive the Riccati recursion:

1. $P_N = Q_N$
2. $F_k = -(R_k + B_k^T P_{k+1} B_k)^{-1}(B_k^T P_{k+1} A_k + S_k^T)$
3. $P_k = Q_k + A_k^T P_{k+1} A_k -$
   $$(A_k^T P_{k+1} B_k + S_k)(R_k + B_k^T P_{k+1} B_k)^{-1}(B_k^T P_{k+1} A_k + S_k^T)$$
4. $\pi_k^*(\mathbf{x}_k) = F_k \mathbf{x}_k$
5. $J_k^*(\mathbf{x}_k) = \frac{1}{2}\mathbf{x}_k^T P_k \mathbf{x}_k$

Compute policy backwards in time, apply policy forward in time.

Stochastic optimal control problem: Markov Decision Problem (MDP)

- System: $\boldsymbol{x}_{k+1} = f_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k), k = 0, \dots, N-1$
- Control constraints: $\boldsymbol{u}_k \in U(\boldsymbol{x}_k)$
- Probability distribution: $\boldsymbol{w}_k \sim P_k(\cdot \mid \boldsymbol{x}_k, \boldsymbol{u}_k)$
- Policies: $\pi = \{\pi_0 \dots, \pi_{N-1}\}$, where $\boldsymbol{u}_k = \pi_k(\boldsymbol{x}_k)$
- Expected Cost:

$$J_\pi(\boldsymbol{x}_0) = E_{\boldsymbol{w}_k, k=0,\dots,N-1}\left[g_N(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} g_k(\boldsymbol{x}_k, \pi_k(\boldsymbol{x}_k), \boldsymbol{w}_k)\right]$$

- Stochastic optimal control problem

$$J^*(x_0) = \min_\pi J_\pi(\boldsymbol{x}_0)$$

Key points

- Discrete-time model
- Markovian model
- Objective: find optimal closed-loop policy
- Additive cost (central assumption)
- Risk-neutral formulation

Key points

- Discrete-time model
- Markovian model
- Objective: find optimal closed-loop policy
- Additive cost (central assumption)
- Risk-neutral formulation

Other communities use different notation: Powell, W. B. *AI, OR and control theory: A Rosetta Stone for stochastic optimization*. Princeton University, 2012.

Principle of optimality

- Let $\pi^* = \{\pi_0^*, \pi_1^*, \dots, \pi_{N-1}^*\}$ be an optimal policy
- Consider tail subproblem

$$E\left[g_N(\boldsymbol{x}_N) + \sum_{k=i}^{N-1} g_k(\boldsymbol{x}_k, \pi_k(\boldsymbol{x}_k), \boldsymbol{w}_k)\right]$$

and the tail policy $\{\pi_i^*, \dots, \pi_{N-1}^*\}$

Principle of optimality: The tail policy is optimal for the tail subproblem

The DP algorithm (stochastic case)

Intuition

- DP first solves ALL tail subproblems at the final stage
- At generic step, it solves ALL tail subproblems of a given time length, using solution of tail subproblems of shorter length

The DP algorithm (stochastic case)

The DP algorithm

- Start with

$$J_N(\boldsymbol{x}_N) = g_N(\boldsymbol{x}_N)$$

and go backwards using

$$J_k(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k \in U(\boldsymbol{x}_k)} E_{\boldsymbol{w}_k}\left[g_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) + J_{k+1}\left(f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k)\right)\right]$$

for $k = 0, 1, \ldots, N-1$

- Then $J^*(\boldsymbol{x}_0) = J_0(\boldsymbol{x}_0)$ and optimal policy is constructed by setting
$$\pi_k^*(\boldsymbol{x}_k) = \underset{\boldsymbol{u}_k \in U(\boldsymbol{x}_k)}{\operatorname{argmin}} E_{\boldsymbol{w}_k}\left[g_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) + J_{k+1}\left(f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k)\right)\right]$$

Example: Inventory Control Problem

- Stock available $x_k \in \mathbb{N}$, inventory $u_k \in \mathbb{N}$, and demand $w_k \in \mathbb{N}$
- Dynamics: $x_{k+1} = \max(0, x_k + u_k - w_k)$
- Constraints: $x_k + u_k \leq 2$
- Probabilistic structure: $p(w_k = 0) = 0.1, p(w_k = 1) = 0.7$, and $p(w_k = 2) = 0.2$
- Cost

$$E\Bigg[\underbrace{0}_{g_3(x_3)} + \sum_{k=0}^{2} \underbrace{(u_k + (x_k + u_k - w_k)^2)}_{g_k(x_k, u_k, w_k)}\Bigg]$$

Example: Inventory Control Problem

- Stock available $x_k \in \mathbb{N}$, inventory $u_k \in \mathbb{N}$, and demand $w_k \in \mathbb{N}$
- Dynamics: $x_{k+1} = \max(0, x_k + u_k - w_k)$
- Constraints: $x_k + u_k \leq 2$
- Probabilistic structure: $p(w_k = 0) = 0.1, p(w_k = 1) = 0.7$, and $p(w_k = 2) = 0.2$
- Cost

$$E\Big[\underbrace{0}_{g_3(x_3)} + \sum_{k=0}^{2} \underbrace{\left(u_k + (x_k + u_k - w_k)^2\right)}_{g_k(x_k, u_k, w_k)}\Big]$$

More generally, could imagine costs:

- $H(x_k)$ – holding inventory
- $B(u_k)$ – buying inventory
- $S(x_k, u_k, w_k)$ – selling (matching stock with demand)

Example: Inventory Control Problem

- Algorithm takes form

$$J_k(x_k) = \min_{0 \leq u_k \leq 2 - x_k} E_{w_k}[u_k + (x_k + u_k - w_k)^2 + J_{k+1}(\max(0, \; x_k + u_k - w_k))]$$

for $k = 0, 1, 2$

- For example

$$J_2(0) = \min_{u_2 = 0,1,2} E_{w_2}[u_2 + (u_2 - w_2)^2] =$$

$$\min_{u_2 = 0,1,2} u_2 + 0.1(u_2)^2 + 0.7(u_2 - 1)^2 + 0.2(u_2 - 2)^2$$

which yields $J_2(0) = 1.3$, and $\pi_2^*(0) = 1$

Example: Inventory Control Problem

Final solution:

- $J_0(0) = 3.7,$
- $J_0(1) = 2.7,$ and
- $J_0(2) = 2.818$

(see this spreadsheet)

Stochastic LQR

Find control policy that minimizes

$$E\left[\frac{1}{2}\boldsymbol{x}_N^T Q \boldsymbol{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \boldsymbol{u}_k^T R_k \boldsymbol{u}_k\right)\right]$$

subject to

- dynamics $\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{w}_k$

with $\boldsymbol{x}_0 \sim \mathcal{N}(\overline{\boldsymbol{x}_0}, \Sigma_{\boldsymbol{x}_0}), \{\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \Sigma_{\boldsymbol{w}_k})\}$ independent and Gaussian vectors

Stochastic LQR

As before, let's suppose $J^*_{k+1}(\mathbf{x}_{k+1}) = \frac{1}{2}\mathbf{x}^T_{k+1} P_k \mathbf{x}_{k+1}$. Then:

$$\begin{aligned}
J^*_k(\mathbf{x}_{k+1}) &= \min_{\mathbf{u}_k} \mathbb{E}_{\mathbf{w}_k}\left[g_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) + J^*_{k+1}(f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k))\right] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\mathbb{E}_{\mathbf{w}_k}\left[\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + (A_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k)^T P_{k+1}(A_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k)\right] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\mathbb{E}_{\mathbf{w}_k}\left[\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + (A_k\mathbf{x}_k + B_k\mathbf{u}_k)^T P_{k+1}(A_k\mathbf{x}_k + B_k\mathbf{u}_k)\right. \\
&\qquad\qquad \left. 2(A_k\mathbf{x}_k + B_k\mathbf{u}_k)^T P_{k+1}\mathbf{w}_k + \mathbf{w}_k^T P_{k+1}\mathbf{w}_k\right] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + (A_k\mathbf{x}_k + B_k\mathbf{u}_k)^T P_{k+1}(A_k\mathbf{x}_k + B_k\mathbf{u}_k) + \mathrm{tr}(P_{k+1}\Sigma_{\mathbf{w}_k})\right)
\end{aligned}$$

Stochastic LQR

As before, let's suppose $J^*_{k+1}(\mathbf{x}_{k+1}) = \frac{1}{2}\mathbf{x}_{k+1}^T P_k \mathbf{x}_{k+1}$. Then:

$$
\begin{aligned}
J^*_k(\mathbf{x}_{k+1}) &= \min_{\mathbf{u}_k} \mathbb{E}_{\mathbf{w}_k}\left[g_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) + J^*_{k+1}(f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k))\right] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\mathbb{E}_{\mathbf{w}_k}\left[\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + (A_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k)^T P_{k+1}(A_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k)\right] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\mathbb{E}_{\mathbf{w}_k}\Big[\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + (A_k\mathbf{x}_k + B_k\mathbf{u}_k)^T P_{k+1}(A_k\mathbf{x}_k + B_k\mathbf{u}_k) \\
&\qquad 2(A_k\mathbf{x}_k + B_k\mathbf{u}_k)^T P_{k+1}\mathbf{w}_k + \mathbf{w}_k^T P_{k+1}\mathbf{w}_k\Big] \\
&= \min_{\mathbf{u}_k} \frac{1}{2}\left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + (A_k\mathbf{x}_k + B_k\mathbf{u}_k)^T P_{k+1}(A_k\mathbf{x}_k + B_k\mathbf{u}_k) + \operatorname{tr}(P_{k+1}\Sigma_{\mathbf{w}_k})\right)
\end{aligned}
$$

➔ optimal policy is the same as in the deterministic case; cost-to-go is increased by some constant related to magnitude of noise

Infinite Horizon MDPs

State: $x \in \mathcal{X}$ (often $s \in \mathcal{S}$)

Action: $u \in \mathcal{U}$ (often $a \in \mathcal{A}$)

Transition Function: $T(x_t \mid x_{t-1}, u_{t-1}) = p(x_t \mid x_{t-1}, u_{t-1})$

Reward Function: $r_t = R(x_t, u_t)$

Discount Factor: γ

MDP: $\mathcal{M} = (\mathcal{X}, \mathcal{U}, T, R, \gamma)$

Infinite Horizon MDPs

MDP: $\mathcal{M} = (\mathcal{X}, \mathcal{U}, T, R, \gamma)$

Stationary policy: $u_t = \pi(x_t)$

Goal: Choose policy that **maximizes cumulative (discounted) reward**

$$V^* = \max_{\pi} E\left[\sum_{t \geq 0} \gamma^t R\big(x_t, \pi(x_t)\big)\right];$$

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t \geq 0} \gamma^t R\big(x_t, \pi(x_t)\big)\right]$$

Infinite Horizon MDPs

- The optimal value function $V^*(x)$ satisfies Bellman's equation

$$V^*(x) = \max_u \left(R(x,u) + \gamma \sum_{x' \in \mathcal{X}} T(x'|x,u)\, V^*(x') \right)$$

- For any stationary policy π, the value $\mathrm{V}_\pi(x) \coloneqq E\left[\sum_{t\geq 0} \gamma^t R\big(x_t, \pi(x_t)\big)\right]$ is the unique solution to the equation

$$\mathrm{V}_\pi(x) = R(x, \pi(x)) + \gamma \sum_{x' \in \mathcal{X}} T(x'|x, \pi(x))\, \mathrm{V}_\pi(x')$$

Solving infinite-horizon MDPs

If you know the model, use DP-ideas

- Value Iteration / Policy Iteration

RL: Learning from interaction

- Model-Based
- Model-free
    - Value based
    - Policy based

Value Iteration

- Initialize $V_0(x) = 0$ for all states x
- Loop until finite horizon / convergence:

$$V_{k+1}(x) = \max_u \left(R(x,u) + \gamma \sum_{x' \in \mathcal{X}} T(x' | x, u) V_k(x') \right)$$

State-action value functions (Q functions)

- The expected cumulative discounted reward starting from x, applying u, and following the optimal policy thereafter

$$V^*(x) = \max_u \underbrace{\left(R(x,u) + \gamma \sum_{x' \in \mathcal{X}} T(x'|x,u)\, V^*(x') \right)}$$

$$V^*(x) = \max_u Q^*(x,u)$$

- Value iteration for Q functions

$$Q_{k+1}(x,u) = R(x,u) + \gamma \sum_{x' \in \mathcal{X}} T(x'|x,u) \max_{u'} Q_k(x',u')$$

Policy Iteration

Starting with a policy $\pi_k(x)$, alternate two steps:

1. <u>Policy Evaluation</u>
   Compute $V_{\pi_k}(x)$ as the solution of
   $$\mathrm{V}_\pi(x) = R(x, \pi(x)) + \gamma \sum_{x' \in \mathcal{X}} T(x' | x, \pi(x))\, \mathrm{V}_\pi(x')$$
2. <u>Policy Improvement</u>
   Define $\pi_{k+1}(x) = \arg\max_u \left(R(x, u) + \gamma \sum_{x' \in \mathcal{X}} T(x' | x, u)\, \mathrm{V}_{\pi_k}(x') \right)$

Proposition: $V_{\pi_{k+1}}(x) \geq V_{\pi_k}(x)\ \forall\ x \in \mathcal{X}$

Inequality is strict if π_k is suboptimal

Use this procedure to iteratively improve policy until convergence

Recap

- Value Iteration
  - Estimate optimal value function
  - Compute optimal policy from optimal value function
- Policy Iteration
  - Start with random policy
  - Iteratively improve it until convergence to optimal policy
- Require **model of MDP** to work!

Next time

- Intro to reinforcement learning
- Belief space MDPs
- Dual control
- LQG