AA203
Optimal and Learning-based Control

Discrete LQR, stochastic DP, value iteration, policy iteration
Roadmap

Control
- Feedback control
- Adaptive control

Optimal and learning control
- Adaptive optimal control
- Model-free RL
- Model-based RL

Open-loop
- Indirect methods
- Direct methods

MPC

Closed-loop
- DP
- HJB / HJI
Dynamic programming

• Model: $x_{k+1} = f(x_k, u_k, k), \quad u_k \in U(x_k)$

• Cost: $J(x_0) = h_N(x_N) + \sum_{k=0}^{N-1} g(x_k, \pi_k(x_k), k)$

DP Algorithm: For every initial state $x_0$, the optimal cost $J^*(x_0)$ is equal to $J_0^*(x_0)$, given by the last step of the following algorithm, which proceeds backward in time from stage $N - 1$ to stage 0:

$$J_N^*(x_N) = h_N(x_N)$$

$$J_k^*(x_k) = \min_{u_k \in U(x_k)} g(x_k, u_k, k) + J_{k+1}^*(f(x_k, u_k, k)), \quad k = 0, \ldots, N - 1$$

Furthermore, if $u_k^* = \pi_k^*(x_k)$ minimizes the right hand side of the above equation for each $x_k$ and $k$, the policy \{\pi_0^*, \pi_1^*, \ldots, \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

\[
J_0(x_0) = \frac{1}{2} x_N^T Q_N x_N + \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q_k x_k + u_k^T R_k u_k + 2x_k^T S_k u_k)
\]

subject to the dynamics

\[
x_{k+1} = A_k x_k + B_k u_k, \quad k \in \{0, 1, \ldots, N - 1\}
\]

assuming

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

\[ J_0(x_0) = \frac{1}{2} x_N^T Q_N x_N + \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q_k x_k + u_k^T R_k u_k + 2x_k^T S_k u_k) \]

subject to dynamics

\[ x_{k+1} = A_k x_k + B_k u_k, \quad k \in \{0, 1, \ldots, N - 1\} \]

as...
Discrete LQR – trajectory optimization

\[
\begin{align*}
\min_{x_k, u_k} & \quad \frac{1}{2} \begin{bmatrix} x_0^T & S_0 & S_0^T & R_0 \\ S_0 & Q_1 & S_1 & \vdots \\ S_1^T & Q_1 & S_1 & \vdots \\ \vdots & \vdots & \vdots & \ddots \\ x_{N-1} & x_{N-1} & x_{N-1} & x_{N-1} \end{bmatrix} \\
\text{s.t.} & \quad \begin{bmatrix} -I & A_0 & B_0 & -I \\ A_1 & B_1 & -I \end{bmatrix} \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ x_N \end{bmatrix} + \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ x_N \end{bmatrix} = 0
\end{align*}
\]
Discrete LQR – trajectory optimization

Defining suitable notation, this is

$$\min_z \quad \frac{1}{2} z^T W z$$

s.t. \hspace{1cm} Cz + d = 0

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

$$\begin{bmatrix} z^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} W & C^T \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -d \end{bmatrix}$$
Discrete LQR – dynamic programming

First step:

\[ J^*_N(x_N) = \frac{1}{2} x_N^T Q_N x_N = \frac{1}{2} x_N^T P_N x_N \]

Proceeding backward in time:

\[ J^*_{N-1}(x_{N-1}) = \min_{u_{N-1}} \frac{1}{2} \left( [x_{N-1}]^T [Q_{N-1} \quad S_{N-1} \quad S_{N-1}^T \quad R_{N-1}] [x_{N-1}] + x_{N-1}^T P_N x_N \right) \]

\[ = \min_{u_{N-1}} \frac{1}{2} \left( [x_{N-1}]^T [Q_{N-1} \quad S_{N-1} \quad S_{N-1}^T \quad R_{N-1}] [x_{N-1}] + \right. \]

\[ \left. (A_{N-1}x_{N-1} + B_{N-1}u_{N-1})^T P_N (A_{N-1}x_{N-1} + B_{N-1}u_{N-1}) \right) \]
Discrete LQR – dynamic programming

Unconstrained NOC:

\[ \nabla_{u_{N-1}} J_{N-1}(x_{N-1}) = R_{N-1} u_{N-1} + S_{N-1}^T x_{N-1} + B_{N-1}^T P_N (A_{N-1} x_{N-1} + B_{N-1} u_{N-1}) = 0 \]

\[ \implies u_{N-1}^* = -(R_{N-1} + B_{N-1}^T P_N B_{N-1})^{-1} (B_{N-1}^T P_N A_{N-1} + S_{N-1}^T) x_{N-1} \]

\[ := F_{N-1} x_{N-1} \]

Note also that SOC hold:

\[ \nabla_{u_{N-1}}^2 J_{N-1}(x_{N-1}) = R_{N-1} + B_{N-1}^T P_N B_{N-1} \succ 0 \]
Discrete LQR – dynamic programming

Plugging in the optimal policy:

\[
J_{N-1}^*(x_{N-1}) = \frac{1}{2} x_{N-1}^T (Q_{N-1} + A_{N-1}^T P_N A_{N-1} - \\
A_{N-1}^T P_N B_{N-1} + S_{N-1})(R_{N-1} + B_{N-1}^T P_N B_{N-1})^{-1} (B_{N-1}^T P_N A_{N-1} + S_{N-1}^T) x_{N-1} \\
:= \frac{1}{2} x_{N-1}^T P_{N-1} x_{N-1}
\]

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. \( P_k = Q_k + A_k^T P_{k+1} A_k - (A_k^T P_{k+1} B_k + S_k)(R_k + B_k^T P_{k+1} B_k)^{-1}(B_k^T P_{k+1} A_k + S_k) \)
4. \( \pi_k^*(x_k) = F_k x_k \)
5. \( J_k^*(x_k) = \frac{1}{2} x_k^T P_k x_k \)

Compute policy backwards in time, apply policy forward in time.
Stochastic optimal control problem: Markov Decision Problem (MDP)

• System: \( x_{k+1} = f_k(x_k, u_k, w_k), k = 0, \ldots, N - 1 \)

• Control constraints: \( u_k \in U(x_k) \)

• Probability distribution: \( w_k \sim P_k(\cdot | x_k, u_k) \)

• Policies: \( \pi = \{\pi_0, \ldots, \pi_{N-1}\} \), where \( u_k = \pi_k(x_k) \)

• Expected Cost:

\[
J_\pi(x_0) = E_{w_k, k=0,\ldots,N-1} \left[ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \pi_k(x_k), w_k) \right]
\]

• Stochastic optimal control problem

\[
J^*(x_0) = \min_{\pi} J_\pi(x_0)
\]
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

Principle of optimality

• Let \( \pi^* = \{\pi_0^*, \pi_1^*, \ldots, \pi_{N-1}^*\} \) be an optimal policy

• Consider tail subproblem

\[
E \left[ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \pi_k(x_k), w_k) \right]
\]

and the tail policy \( \{\pi_i^*, \ldots, \pi_{N-1}^*\} \)

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

\[ J_N(x_N) = g_N(x_N) \]

and go backwards using

\[ J_k(x_k) = \min_{u_k \in U(x_k)} E_{w_k} [g_k(x_k, u_k, w_k) + J_{k+1}(f(x_k, u_k, w_k))] \]

for \( k = 0, 1, \ldots, N - 1 \)

• Then \( J^*(x_0) = J_0(x_0) \) and optimal policy is constructed by setting

\[ \pi_k^*(x_k) = \arg\min_{u_k \in U(x_k)} E_{w_k} [g_k(x_k, u_k, w_k) + J_{k+1}(f(x_k, u_k, w_k))] \]
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

\[
E \left[ g_3(x_3) + \sum_{k=0}^{2} \left( u_k + (x_k + u_k - w_k)^2 \right) \right]
\]
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

$$E \left[ 0 + \sum_{k=0}^{2} \left( u_k + (x_k + u_k - w_k)^2 \right) \right]$$

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

\[
E \left[ \frac{1}{2} x_N^T Q x_N + \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q_k x_k + u_k^T R_k u_k) \right]
\]

subject to

\* dynamics \( x_{k+1} = A_k x_k + B_k u_k + w_k \)

with \( x_0 \sim \mathcal{N}(\overline{x}_0, \Sigma_{x_0}) \), \( \{w_k \sim \mathcal{N}(0, \Sigma_{w_k})\} \) independent and Gaussian vectors
Stochastic LQR

As before, let’s suppose $J_{k+1}^*(x_{k+1}) = \frac{1}{2} x_{k+1}^T P_k x_{k+1}$. Then:

$$J_k^*(x_{k+1}) = \min_{u_k} \mathbb{E}_{w_k} \left[ g_k(x_k, u_k, w_k) + J_{k+1}^*(f(x_k, u_k, w_k)) \right]$$

$$= \min_{u_k} \frac{1}{2} \mathbb{E}_{w_k} \left[ x_k^T Q_k x_k + u_k^T R_k u_k + (A_k x_k + B_k u_k + w_k)^T P_{k+1} (A_k x_k + B_k u_k + w_k) \right]$$

$$= \min_{u_k} \frac{1}{2} \mathbb{E}_{w_k} \left[ x_k^T Q_k x_k + u_k^T R_k u_k + (A_k x_k + B_k u_k)^T P_{k+1} (A_k x_k + B_k u_k) \right.$$  

$$+ 2(A_k x_k + B_k u_k)^T P_{k+1} w_k + w_k^T P_{k+1} w_k \right]$$

$$= \min_{u_k} \frac{1}{2} \left( x_k^T Q_k x_k + u_k^T R_k u_k + (A_k x_k + B_k u_k)^T P_{k+1} (A_k x_k + B_k u_k) + \text{tr}(P_{k+1} \Sigma_w) \right)$$
Stochastic LQR

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

$$J^*_k(x_{k+1}) = \min_{u_k} \mathbb{E}_{w_k} \left[ g_k(x_k, u_k, w_k) + J^*_{k+1}(f(x_k, u_k, w_k)) \right]$$

$$= \min_{u_k} \frac{1}{2} \mathbb{E}_{w_k} \left[ x_k^TQ_kx_k + u_k^TR_ku_k + (A_kx_k + B_ku_k + w_k)^TP_{k+1}(A_kx_k + B_ku_k + w_k) \right]$$

$$= \min_{u_k} \frac{1}{2} \mathbb{E}_{w_k} \left[ x_k^TQ_kx_k + u_k^TR_ku_k + (A_kx_k + B_ku_k)^TP_{k+1}(A_kx_k + B_ku_k) + 2(A_kx_k + B_ku_k)^TP_{k+1}w_k + w_k^TP_{k+1}w_k \right]$$

$$= \min_{u_k} \frac{1}{2} \left( x_k^TQ_kx_k + u_k^TR_ku_k + (A_kx_k + B_ku_k)^TP_{k+1}(A_kx_k + B_ku_k) + \text{tr}(P_{k+1}\Sigma_{w_k}) \right)$$

\[ \Rightarrow \text{optimal policy is the same as in the deterministic case; cost-to-go is increased by some constant related to magnitude of noise} \]
Infinite Horizon MDPs

State: \( x \in \mathcal{X} \) \hspace{1cm} (often \( s \in \mathcal{S} \))

Action: \( u \in \mathcal{U} \) \hspace{1cm} (often \( a \in \mathcal{A} \))

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

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

Discount Factor: \( \gamma \)

**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

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

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

• The optimal value function $V^*(x)$ satisfies Bellman’s equation

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

• For any stationary policy $\pi$, the value $V_\pi(x) := E\left[ \sum_{t \geq 0} \gamma^t R(x_t, \pi(x_t)) \right]$ is the unique solution to the equation

$$V_\pi(x) = R(x, \pi(x)) + \gamma \sum_{x' \in X} T(x'|x, \pi(x)) V_\pi(x')$$
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:

$$V_{k+1}(x) = \max_u \left( R(x, u) + \gamma \sum_{x' \in X} T(x'|x, u) V_k(x') \right)$$
State-action value functions (Q functions)

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

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

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

• Value iteration for $Q$ functions

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

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

1. **Policy Evaluation**
   
   Compute $V_{\pi_k}(x)$ as the solution of
   
   $$V_{\pi}(x) = R(x, \pi(x)) + \gamma \sum_{x' \in X} T(x'|x, \pi(x)) V_{\pi}(x')$$

2. **Policy Improvement**
   
   Define $\pi_{k+1}(x) = \arg\max_u \left( R(x, u) + \gamma \sum_{x' \in X} T(x'|x, u) V_{\pi_k}(x') \right)$

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

   Inequality is strict if $\pi_k$ 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