AA203
Optimal and Learning-based Control

Dynamic programming
Optimal control

- Open-loop
  - Indirect methods
  - Direct methods
- MPC
- Closed-loop
  - Adaptive optimal control
  - Model-based RL
    - Linear methods
    - Non-linear methods
    - DP
    - HJB / HJI

Adaptive optimal control

Model-based RL
Principle of optimality

The **key** concept behind the dynamic programming approach is the **principle of optimality**

Suppose optimal path for a multi-stage decision-making problem is

- first decision yields segment $a - b$ with cost $J_{ab}$
- remaining decisions yield segments $b - e$ with cost $J_{be}$
- optimal cost is then $J_{ae}^* = J_{ab} + J_{be}$
Principle of optimality

• Claim: If \( a - b - e \) is optimal path from \( a \) to \( e \), then \( b - e \) is optimal path from \( b \) to \( e \)

• Proof: Suppose \( b - c - e \) is the optimal path from \( b \) to \( e \). Then

\[
J_{bce} < J_{be}
\]

and

\[
J_{ab} + J_{bce} < J_{ab} + J_{be} = J_{ae}^*
\]

Contradiction!
Principle of optimality

Principle of optimality (for discrete-time systems): Let $\pi^*: = \{\pi^*_0, \pi^*_1, \ldots, \pi^*_N\}$ be an optimal policy. Assume state $x_k$ is reachable. Consider the subproblem whereby we are at $x_k$ at time $k$ and we wish to minimize the cost-to-go from time $k$ to time $N$. Then the truncated policy $\{\pi^*_k, \pi^*_{k+1}, \ldots, \pi^*_{N-1}\}$ is optimal for the subproblem.

• tail policies optimal for tail subproblems
• notation: $\pi^*_k(x_k) = \pi^*(x_k, k)$
Applying the principle of optimality

Principle of optimality: if $b - c$ is the initial segment of the optimal path from $b$ to $f$, then $c - f$ is the terminal segment of this path.

Hence, the optimal trajectory is found by comparing:

$$C_{bcf} = J_{bc} + J_{cf}^*$$
$$C_{bdf} = J_{bd} + J_{df}^*$$
$$C_{bef} = J_{be} + J_{ef}^*$$
Applying the principle of optimality

• need only to compare the concatenations of immediate decisions and optimal decisions → significant decrease in computation / possibilities

• in practice: carry out this procedure backward in time
Example

Optimal cost: 18
Optimal path: $a \rightarrow d \rightarrow e \rightarrow f \rightarrow g \rightarrow h$
DP Algorithm

• 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.
Comments

• discretization (from differential equations to difference equations)
• quantization (from continuous to discrete state variables / controls)
• interpolation
• global minimum
• constraints, in general, simplify the numerical procedure
• optimal control in closed-loop form
• curse of dimensionality
Example: discrete LQR

• In most cases, DP algorithm needs to be performed numerically
• A few cases can be solved analytically

**Discrete LQR:** select control inputs to minimize

\[
J(x_0) = \frac{1}{2} x'_N H x_N + \frac{1}{2} \sum_{k=0}^{N-1} [x'_k Q x_k + u'_k R u_k]
\]

subject to the dynamics

\[
x_{k+1} = A_k x_k + B_k u_k
\]

Assumption: \( H = H' \geq 0, Q = Q' \geq 0, R = R' > 0 \)
Example: discrete LQR

First step:

\[
J^*_N(x_N) = \frac{1}{2} x'_N H x_N := \frac{1}{2} x'_N P_N x_N
\]

Going backward

\[
J_{N-1}(x_{N-1}) = \min_{u_{N-1}} \frac{1}{2} \left\{ x'_{N-1} Q x_{N-1} + u'_{N-1} R u_{N-1} + x'_N H x_N \right\}
\]

\[
\min_{u_{N-1}} \frac{1}{2} \left\{ x'_{N-1} Q x_{N-1} + u'_{N-1} R u_{N-1} + (A_{N-1} x_{N-1} + B_{N-1} u_{N-1})' H (A_{N-1} x_{N-1} + B_{N-1} u_{N-1}) \right\}
\]
Example: discrete LQR

Taking derivative

\[
\frac{\partial J^*_N\left(\mathbf{x}_{N-1}\right)}{\partial \mathbf{u}_{N-1}} = R\mathbf{u}_{N-1} + B'_{N-1}H(A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}) = 0
\]

and

\[
\frac{\partial^2 J^*_N\left(\mathbf{x}_{N-1}\right)}{\partial \mathbf{u}_{N-1}^2} = R + B'_{N-1}H B_{N-1} > 0
\]
DP for discrete LQR

Hence, the optimizer satisfies

$$( R + B'_{N-1} H B_{N-1} ) u^*_{N-1} + B'_{N-1} H A_{N-1} x_{N-1} = 0$$

so

$$u^*_{N-1} = - ( R + B'_{N-1} H B_{N-1} )^{-1} B'_{N-1} H A_{N-1} x_{N-1} := F_{N-1} x_{N-1}$$
DP for discrete LQR

Plugging in

\[ J_{N-1}(x_{N-1}) = \frac{1}{2} x'_{N-1} \left\{ Q + F'_{N-1} R F_{N-1} + \right. \]
\[ \left. (A_{N-1} + B_{N-1} F_{N-1})' H (A_{N-1} + B_{N-1} F_{N-1}) \right\} x_{N-1} \]

\[ := x'_{N-1} P_{N-1} x_{N-1} \]

\[ F_{N-1} = \left. - (R + B'_{N-1} P_N B_{N-1})^{-1} B'_{N-1} P_N A_{N-1} \right. \]
DP for discrete LQR

Proceeding by induction, the solution is given by

1. \( J_N (\mathbf{x}_N) = \frac{1}{2} \mathbf{x}'_N P_N \mathbf{x}_N \), where \( P_N = H \)

2. \( \mathbf{u}_k^* = F_k \mathbf{x}_k \), where \( F_k = -(R + B_k' P_{k+1} B_k)^{-1} B_k' P_{k+1} A_k \)

3. \( J_k (\mathbf{x}_k) = \frac{1}{2} \mathbf{x}'_k P_k \mathbf{x}_k \), where

\[
P_k = Q + F_k' R F_k + (A_k + B_k F_k)' P_{k+1} (A_k + B_k F_k)
\]

At the end, \( J_0 (\mathbf{x}_0) = \frac{1}{2} \mathbf{x}'_0 P_0 \mathbf{x}_0 \)
Next time

• iLQR, DDP, and LQG

\[
x_{k+1} = A_k x_k + B_k u_k + w_k \\
y_k = C_k x_k + v_k
\]