Dynamic Programming II

CMPT 882
Feb. 25
Dynamic Programming: Discrete Time

• Discrete time model: $x_{k+1} = f_d(x_k, u_k)$, $u_k \in U(x_k)$
  • May be obtained through discretizing continuous time model (eg. Forward Euler: $x_{k+1} = x_k + \Delta t f(x_k, u_k)$)
  • Cost: $J_N(x_N) = l(x_N) + \sum_{k=0}^{N-1} c(x_k, u_k)$

• Find optimal cost:
  
  $$J_0^*(x_0) = \min_{u} \left\{ l(x_N) + \sum_{k=0}^{N-1} c(x_k, u_k) \right\}$$

• Strategy: start at $k = N$ and work backwards to obtain $J_k(x)$
  • $J_N^*(x_N) = h_N(x_N) = l(x_N)$
  • $J_k^*(x_k) = \min_{u_k \in U(x_k)} \{ c_k(x_k, u_k) + J_{k+1}^*(f(x_k, u_k)) \}$
Example: Linear Quadratic Regulator (LQR)

- From before:
  \[ J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \frac{1}{2} \{x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + (A x_{N-1} + B u_{N-1})^T L (A x_{N-1} + B u_{N-1}) \} \]
  - Decision variable: \( u_{N-1} \)

- Take derivatives to find minimum:
  \[ \frac{\partial J_{N-1}(x_{N-1})}{\partial u_{N-1}} = Ru_{N-1} + B^T L (A x_{N-1} + B u_{N-1}) \]
  - Set to zero to obtain \( u_{N-1}^* \)
  - Plug in \( u_{N-1}^* \) into \( J_{N-1}(x_{N-1}) \)

- Positive semidefinite
- First order condition is sufficient

\[ u_{N-1}^* = F x_{N-1} \]

where \( F = -(R + B^T L B)^{-1} B^T L A \)

\[ J_{N-1}^*(x_{N-1}) = \frac{1}{2} x_{N-1}^T P x_{N-1} \]
where \( P = Q + F^T R F + (A + B F)^T L (A + B F) \)
Example: Linear Quadratic Regulator (LQR)

• Set derivative to zero to obtain control:

\[ Ru_{N-1} + B^T L (Ax_{N-1} + Bu_{N-1}) = 0 \]
\[ Ru_{N-1} + B^T L A x_{N-1} + B^T L B u_{N-1} = 0 \]
\[ (R + B^T L B) u_{N-1} + B^T L A x_{N-1} = 0 \]

\[ u_{N-1}^* = F x_{N-1}, \text{ where } F = -(R + B^T L B)^{-1} B^T L A \]

• Plug \( u_{N-1}^* \) into for \( J_{N-1} \)

\[ J_{N-1}^*(x_{N-1}) = \frac{1}{2} \{ x_{N-1}^T Q x_{N-1} + u_{N-1}^* R u_{N-1}^* + (Ax_{N-1} + Bu_{N-1})^T L (Ax_{N-1} + Bu_{N-1}) \} \]
\[ J_{N-1}^*(x_{N-1}) = \frac{1}{2} \{ x_{N-1}^T Q x_{N-1} + x_{N-1}^T F^T R F x_{N-1} + (Ax_{N-1} + BF x_{N-1})^T L (Ax_{N-1} + BF x_{N-1}) \} \]
\[ J_{N-1}^*(x_{N-1}) = \frac{1}{2} x_{N-1}^T (Q + F^T R F + (A + BF)^T L (A + BF)) x_{N-1} \]
\[ J_{N-1}^*(x_{N-1}) = \frac{1}{2} x_{N-1}^T P x_{N-1}, \text{ where } P = Q + F^T R F + (A + BF)^T L (A + BF) \]
Example: Linear Quadratic Regulator (LQR)

- Look for a pattern

\[ J_N^*(x_N) = \frac{1}{2} x_N^T L x_N \]

- \( u_{N-1}^* = F x_{N-1} \), where \( F = -(R + B^T L B)^{-1} B^T L A \)

- \( J_{N-1}^*(x_{N-1}) = \frac{1}{2} x_{N-1}^T P x_{N-1} \), where \( P = Q + F^T R F + (A + B F)^T L (A + B F) \)
Example: Linear Quadratic Regulator (LQR)

- Look for a pattern
  
  $J_N^*(x_N) = \frac{1}{2} x_N^T L x_N$
  
  $u_{N-1}^* = F x_{N-1}$, where $F = -(R + B^T L B)^{-1} B^T L A$
  
  $J_{N-1}^*(x_{N-1}) = \frac{1}{2} x_{N-1}^T P x_{N-1}$, where $P = Q + F^T R F + (A + BF)^T L (A + BF)$
Example: Linear Quadratic Regulator (LQR)

- Look for a pattern

\[ J_N^* (x_N) = \frac{1}{2} x_N^T P_N x_N, \] where \( P_N = L \)

\[ u_{N-1}^* = F_{N-1} x_{N-1} \] where \( F_{N-1} = -(R + B^T P_N B)^{-1} B^T P_N A \)

\[ J_{N-1}^* (x_{N-1}) = \frac{1}{2} x_{N-1}^T P_{N-1} x_{N-1}, \] where \( P_{N-1} = Q + F_{N-1}^T R F_{N-1} + (A + B F_{N-1})^T P_N (A + B F_{N-1}) \)
Example: Linear Quadratic Regulator (LQR)

• Proceed by induction

\[ J_N^*(x_N) = \frac{1}{2} x_N^T P_N x_N, \text{ where } P_N = L \]

\[ u_k^* = F_k x_k, \text{ where } F_k = -(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A \]

\[ J_{N-1}^*(x_k) = \frac{1}{2} x_k^T P_k x_k, \text{ where } P_k = Q + F_k^T R F_k + (A + B F_k)^T P_{k+1} (A + B F_k) \]
Example: Linear Quadratic Regulator (LQR)

- Proceed by induction

\[ J_N^*(x_N) = \frac{1}{2} x_N^T P_N x_N, \text{ where } P_N = L \]

\[ u_k^* = F_k x_k, \text{ where } F_k = -(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A \]

\[ J_{N-1}^*(x_k) = \frac{1}{2} x_k^T P_k x_k, \text{ where } P_k = Q + F_k^T R F_k + (A + B F_k)^T P_{k+1} (A + B F_k) \]

- Eventually,

\[ J_0(x_0) = \frac{1}{2} x_0^T P_0 x_0 \]
Comments

• No control constraint

• What if there is control constraint?
  • Practically, let controllers saturate
  • Explicitly treat it in the minimization of $J$ $\leftarrow$ more difficult

• MATLAB commands
  • Discrete time: d1qr; continuous time: lqr

• In general, need to solve
  • $J_k(x_k) = \min_{u_k \in U(x_k)} \{c_k(x_k, u_k) + J_{k+1}(f(x_k, u_k))\}$, $J_N(x_N) = l(x_N)$
  • If $x \in \mathbb{R}^n$, then $(x_k)$ is an $n + 1$ dimensional array
Dynamic Programming: Continuous Time

\[
\text{minimize } \quad J(x(t_f), t_f) + \int_0^{t_f} c(x(t), u(t)) \, dt
\]

subject to \( \dot{x}(t) = f(x(t), u(t)) \)

\[ x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, x(0) = x_0 \]

- Let \( J(x(t), t) = l(x(t_f), t_f) + \int_t^{t_f} c(x(t), u(t)) \, dt \)
- \( J^*(x(0), 0) \) is what we want

- Strategy:
  - make a “discrete time” argument with \( \Delta t \)
  - Let \( \Delta t \to 0 \)
Dynamic Programming: Continuous Time

• Let \( J(x(t), t) = \int_t^T C(x(s), u(s))ds + l(x(t_f)) \) “Cost to go”

\[
V(x(t), t) := J^*(x(t), t) = \min_{u_{[t,T]}} \left[ \int_t^T C(x(s), u(s))ds + l(x(T)) \right]
\]

“Value function”, “\( J^*(x(t), t) \)”

Write out time interval explicitly for clarity

• Dynamic programming principle:

\[
V(x(t), t) = \min_{u_{[t,t+\delta]}} \left[ \int_t^{t+\delta} C(x(s), u(s))ds + V(x(t+\delta), t+\delta) \right]
\]

• Approximate integral and Taylor expand \( V(x(t + \delta), t + \delta) \)

• Derive Hamilton-Jacobi partial differential equation (HJ PDE)
Dynamic Programming: Continuous Time

- Approximations for small $\delta$:

$$
V(x(t), t) = \min_{u_{[t, t+\delta]}(\cdot)} \left[ \int_{t}^{t+\delta} C(x(s), u(s)) \, ds + V(x(t+\delta), t+\delta) \right]
$$

- Omit $t$ dependence...

$$
V(x, t) = \min_u \left[ C(x, u) \delta + V(x, t) + \frac{\partial V}{\partial x} \cdot \delta f(x, u) + \frac{\partial V}{\partial t} \delta \right]
$$

- $V(x, t)$ does not depend on $u$

$$
V(x, t) = V(x, t) + \min_u \left[ C(x, u) \delta + \frac{\partial V}{\partial x} \cdot \delta f(x, u) + \frac{\partial V}{\partial t} \delta \right]
$$

Optimization over a vector, not a function!
Dynamic Programming: Continuous Time

• Approximations for small $\delta$:

$$V(x(t), t) = \min_{u_{[t, t+\delta]}} \left[ \int_{t}^{t+\delta} C(x(s), u(s)) ds + V(x(t+\delta), t+\delta) \right]$$

\[\begin{align*}
V(x(t), t) &= C(x(t), u(t))\delta \\
&+ x(t) + \delta f(x, u) \\
&+ V(x(t), t) + \frac{\partial V}{\partial x} \cdot \delta f(x, u) + \frac{\partial V}{\partial t} \delta
\end{align*}\]

• Omit $t$ dependence...

\[V(x, t) = \min_u \left[ C(x, u)\delta + V(x, t) + \frac{\partial V}{\partial x} \cdot \delta f(x, u) + \frac{\partial V}{\partial t} \delta \right]\]

Optimization over a vector, not a function!

• $V(x, t)$ does not depend on $u$

$$V(x, t) = V(x, t) + \frac{\partial V}{\partial t} \delta + \min_u \left[ C(x, u)\delta + \frac{\partial V}{\partial x} \cdot \delta f(x, u) \right]$$
Dynamic Programming: Continuous Time

• Approximations for small $\delta$:

$$V(x(t), t) = \min_{u \in [t, t+\delta]} \left[ \int_t^{t+\delta} C(x(s), u(s)) ds + V(x(t+\delta), t+\delta) \right]$$

$V(x(t), t)$ does not depend on $u$

$$x(t) + \delta f(x, u)$$

$V(x(t), t) + \frac{\partial V}{\partial x} \cdot \delta f(x(t), u(t)) + \frac{\partial V}{\partial t} \delta$

• Omit $t$ dependence...

$$\begin{align*}
V(x, t) &= \min_u \left[ C(x, u) \delta + V(x, t) + \frac{\partial V}{\partial x} \cdot \delta f(x, u) + \frac{\partial V}{\partial t} \delta \right] \\
0 &= \frac{\partial V}{\partial t} \delta + \min_u \left[ C(x, u) \delta + \frac{\partial V}{\partial x} \cdot \delta f(x, u) \right]
\end{align*}$$

Optimization over a vector, not a function!
Dynamic Programming: Continuous Time

• **Approximations for small $\delta$:**

\[
V(x(t), t) = \min_{u(\cdot)} \left[ \int_{t}^{t+\delta} C(x(s), u(s)) ds + V(x(t+\delta), t+\delta) \right]
\]

\[
= C(x(t), u(t)) \delta + x(t) + \delta f(x(t), u(t)) + \frac{\partial V}{\partial t} \delta
\]

• **Omit $t$ dependence...**

\[
V(x, t) = \min_u \left[ C(x, u) \delta + V(x, t) + \frac{\partial V}{\partial x} \cdot \delta f(x, u) + \frac{\partial V}{\partial t} \delta \right]
\]

Optimization over a vector, not a function!

• **$V(x, t)$ does not depend on $u$**

\[
\frac{\partial V}{\partial t} + \min_u \left[ C(x, u) + \frac{\partial V}{\partial x} \cdot f(x, u) \right] = 0
\]
Comments

• Hamilton-Jacobi partial differential equation
  \[
  \frac{\partial V}{\partial t} + \min_u \left[ C(x,u) + \frac{\partial V}{\partial x} \cdot f(x,u) \right] = 0, \quad V(x, t_f) = l(x)
  \]

• Terminology:
  • Pre-Hamiltonian: \( H(x, u, \lambda) = C(x, u) + \lambda^\top f(x, u) \)
  • Hamiltonian: \( H^*(x, \lambda) = C(x, u^*) + \lambda^\top f(x, u^*) \)
  \[
  \Rightarrow \frac{\partial V}{\partial t} + H^*(x, \lambda) = 0
  \]
Comments

• Hamilton-Jacobi partial differential equation
  \[ \frac{\partial V}{\partial t} + \min_u \left[ C(x, u) + \frac{\partial V}{\partial x} \cdot f(x, u) \right] = 0, \quad V(x, t_f) = l(x) \]

• Minimization over \( u \) is typically easy
  • Most systems are control affine: \( f(x, u) \) has the form \( f(x) + g(x)u \)
  • Control constraints are typically “box” constraints, e.g. \( |u_i| \leq 1 \)

• PDE is solved on a grid
  • \( x \in \mathbb{R}^n \) means \( V(t, x) \) is computed on an \((n + 1)\)-dimensional grid

• \( V(x, t) \) is often not differentiable
  • Viscosity solutions
  • Lax Friedrichs numerical method
Comments

• Hamilton-Jacobi partial differential equation
  \[ \frac{\partial V}{\partial t} + \min_u \left[ C(x, u) + \frac{\partial V}{\partial x} \cdot f(x, u) \right] = 0, \quad V(x, t_f) = l(x) \]

• Minimization over \( u \) is typically easy
  • Most systems are control affine: \( f(x, u) \) has the form \( f(x) + g(x)u \)
  • Control constraints are typically “box” constraints, e.g. \( |u_i| \leq 1 \)

• PDE is solved on a grid
  • \( x \in \mathbb{R}^n \) means \( V(t, x) \) is computed on an \((n + 1)\)-dimensional grid

• \( V(x, t) \) is often not differentiable
  • Viscosity solutions
  • Lax Friedrichs numerical method
• Hamilton-Jacobi partial differential equation
  \[ \frac{\partial V}{\partial t} + \min_u \left[ C(x, u) + \frac{\partial V}{\partial x} \cdot f(x, u) \right] = 0, \quad V(x, t_f) = l(x) \]

• Minimization over \( u \) is typically easy
  • Most systems are control affine: \( f(x, u) \) has the form \( f(x) + g(x)u \)
  • Control constraints are typically “box” constraints, e.g. \( |u_i| \leq 1 \)

• PDE is solved on a grid
  • \( x \in \mathbb{R}^n \) means \( V(t, x) \) is computed on an \((n + 1)\)-dimensional grid

• \( V(x, t) \) is often not differentiable
  • Viscosity solutions
  • Lax Friedrichs numerical method