Dynamic Programming II

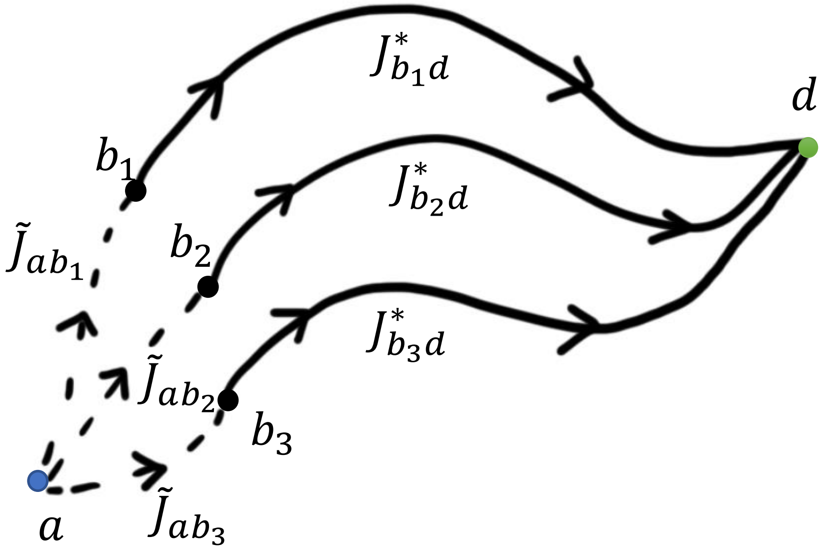
CMPT 882

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:

$$J^*_{N-1}(x_{N-1}) = \min_{u_{N-1}} \overbrace{\frac{1}{2}\{x_{N-1}^\top Q x_{N-1} + u_{N-1}^\top R u_{N-1} + (Ax_{N-1} + Bu_{N-1})^\top L (Ax_{N-1} + Bu_{N-1})\}}^{J_{N-1}(x_{N-1})}$$

  - Decision variable: u_{N-1}

- Take derivatives to find minimum:

$$\frac{\partial^2 J_{N-1}(x_{N-1})}{\partial u_{N-1}^2} = R + B^\top L B \geq 0$$

- Positive semidefinite
- First order condition is sufficient

$$\frac{\partial J_{N-1}(x_{N-1})}{\partial u_{N-1}} = R u_{N-1} + B^\top L (A x_{N-1} + B u_{N-1})$$

- Set to zero to obtain u^*_{N-1} ➡ $u^*_{N-1} = F x_{N-1}$, where $F = -(R + B^\top L B)^{-1} B^\top L A$
- Plug in u^*_{N-1} into $J_{N-1}(x_{N-1})$ ➡ $J^*_{N-1}(x_{N-1}) = \frac{1}{2} x_{N-1}^\top P x_{N-1}$, where $P = Q + F^\top R F + (A + BF)^\top L (A + BF)$

Example: Linear Quadratic Regulator (LQR)

- Set derivative to zero to obtain control:

$$Ru_{N-1} + B^\top L(Ax_{N-1} + Bu_{N-1}) = 0$$
$$Ru_{N-1} + B^\top LAx_{N-1} + B^\top LBu_{N-1} = 0$$
$$(R + B^\top LB)u_{N-1} + B^\top LAx_{N-1} = 0$$

  - $u^*_{N-1} = Fx_{N-1}$, where $F = -(R + B^\top LB)^{-1}B^\top LA$

- Plug u^*_{N-1} into for J_{N-1}

$$J^*_{N-1}(x_{N-1}) = \frac{1}{2}\{x^\top_{N-1}Qx_{N-1} + u^{*\top}_{N-1}Ru^*_{N-1} + (Ax_{N-1} + Bu^*_{N-1})^\top L(Ax_{N-1} + Bu^*_{N-1})\}$$
$$J^*_{N-1}(x_{N-1}) = \frac{1}{2}\{x^\top_{N-1}Qx_{N-1} + x^\top_{N-1}F^\top RFx_{N-1} + (Ax_{N-1} + BFx_{N-1})^\top L(Ax_{N-1} + BFx_{N-1})\}$$
$$J^*_{N-1}(x_{N-1}) = \frac{1}{2}x^\top_{N-1}\big(Q + F^\top RF + (A + BF)^\top L(A + BF)\big)x_{N-1}$$
$$J^*_{N-1}(x_{N-1}) = \frac{1}{2}x^\top_{N-1}Px_{N-1}, \text{ where } P = Q + F^\top RF + (A + BF)^\top L(A + BF)$$

Example: Linear Quadratic Regulator (LQR)

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

Example: Linear Quadratic Regulator (LQR)

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

Example: Linear Quadratic Regulator (LQR)

- Look for a pattern
- $J_N^*(x_N) = \frac{1}{2} x_N^\top P_N x_N$, where $P_N = L$
  - $u_{N-1}^* = F_{N-1} x_{N-1}$, where $F_{N-1} = -(R + B^\top P_N B)^{-1} B^\top P_N A$
  - $J_{N-1}^*(x_{N-1}) = \frac{1}{2} x_{N-1}^\top P_{N-1} x_{N-1}$, where $P_{N-1} = Q + F_{N-1}^\top R F_{N-1} + (A + B F_{N-1})^\top P_N (A + B F_{N-1})$

Example: Linear Quadratic Regulator (LQR)

- Proceed by induction
- $J_N^*(x_N) = \frac{1}{2} x_N^\top P_N x_N$, where $P_N = L$
  - $u_k^* = F_k x_k$, where $F_k = -(R + B^\top P_{k+1} B)^{-1} B^\top P_{k+1} A$
  - $J_{N-1}^*(x_k) = \frac{1}{2} x_k^\top P_k x_k$, where $P_k = Q + F_k^\top R F_k + (A + BF_k)^\top P_{k+1} (A + BF_k)$

Example: Linear Quadratic Regulator (LQR)

- Proceed by induction
- $J_N^*(x_N) = \frac{1}{2} x_N^\top P_N x_N$, where $P_N = L$
  - $u_k^* = F_k x_k$, where $F_k = -(R + B^\top P_{k+1} B)^{-1} B^\top P_{k+1} A$
  - $J_{N-1}^*(x_k) = \frac{1}{2} x_k^\top P_k x_k$, where $P_k = Q + F_k^\top R F_k + (A + BF_k)^\top P_{k+1}(A + BF_k)$

- Eventually,
  - $J_0(x_0) = \frac{1}{2} x_0^\top P_0 x_0$

Comments

- No control constraint

- What if there is control constraint?
  - Practically, let controllers saturate
  - Explicitly treat it in the minimization of J ← more difficult

- MATLAB commands
  - Discrete time: `dlqr`; continuous time: `lqr`

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

Dynamic Programming: Continuous Time

$$\underset{u(\cdot)}{\text{minimize}} \ \overbrace{l\big(x(t_f), t_f\big)}^{\text{Final cost}} + \overbrace{\int_0^{t_f} c\big(x(t), u(t)\big)dt}^{\text{Running cost}}$$

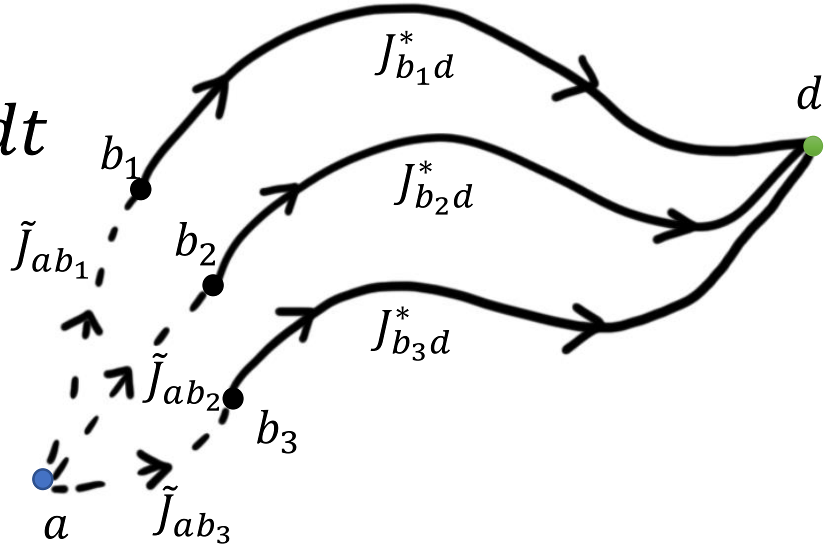
Cost functional, $J\big(x(\cdot), u(\cdot)\big)$

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

Dynamic model

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

- Let $J(x(t), t) = l\big(x(t_f), t_f\big) + \int_t^T C\big(x(t), u(t)\big)dt$
    - $J^*(x(0), 0)$ is what we want
- Strategy:
    - make a “discrete time” argument with Δt
    - Let $\Delta t \to 0$



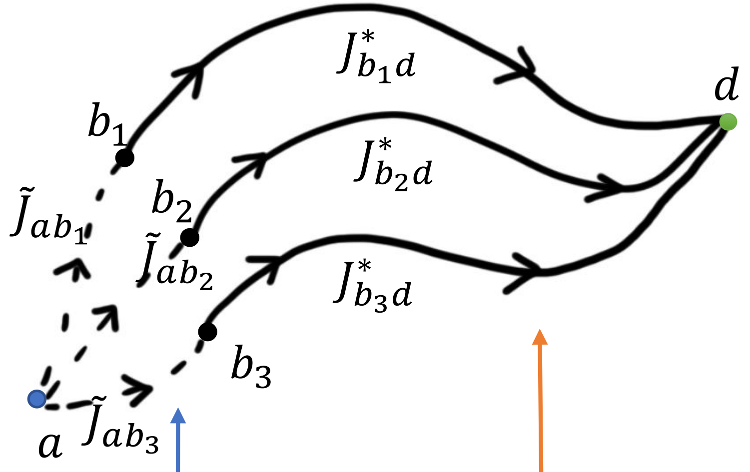
Dynamic Programming: Continuous Time

- Let $J(x(t),t) = \int_t^T C\big(x(s),u(s)\big)ds + l\Big(x\big(t_f\big)\Big)$ "Cost to go"

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

Write out time interval explicitly for clarity

"Value function", "$J^*(x(t),t)$"



- Dynamic programming principle:

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

  - 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 δ:

$$V(x(t),t) = \min_{u_{[t,t+\delta]}(\cdot)} \left[\underbrace{\int_t^{t+\delta} C\big(x(s),u(s)\big)ds}_{C\left(x(t),u(t)\right)\delta} + \underbrace{V(\overbrace{x(t+\delta)}^{x(t)+\delta f(x,u)}, t+\delta)}_{V(x(t),t) + \frac{\partial V}{\partial x}\cdot \delta f(x(t),u(t)) + \frac{\partial V}{\partial t}\delta} \right]$$

  - Omit t dependence…

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

Optimization over a vector, not a function!

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

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

Dynamic Programming: Continuous Time

- Approximations for small δ:

$$V(x(t),t) = \min_{u_{[t,t+\delta]}(\cdot)} \left[\underbrace{\int_t^{t+\delta} C\big(x(s),u(s)\big)ds}_{C(x(t),u(t))\delta} + \underbrace{V\big(\overbrace{x(t+\delta)}^{x(t)+\delta f(x,u)},t+\delta\big)}_{V(x(t),t)+\frac{\partial V}{\partial x}\cdot\delta f(x(t),u(t))+\frac{\partial V}{\partial t}\delta} \right]$$

  - Omit t dependence…

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

Optimization over a vector, not a function!

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

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

Dynamic Programming: Continuous Time

- Approximations for small δ:

$$V(x(t),t) = \min_{u_{[t,t+\delta]}(\cdot)} \left[\underbrace{\int_t^{t+\delta} C\big(x(s),u(s)\big)ds}_{C\big(x(t),u(t)\big)\delta} + \underbrace{V\big(\overbrace{x(t+\delta)}^{x(t)+\delta f(x,u)}, t+\delta\big)}_{V(x(t),t) + \frac{\partial V}{\partial x}\cdot \delta f(x(t),u(t)) + \frac{\partial V}{\partial t}\delta} \right]$$

  - Omit t dependence…

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

Optimization over a vector, not a function!

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

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

Dynamic Programming: Continuous Time

- Approximations for small δ:

$$V(x(t),t) = \min_{u_{[t,t+\delta]}(\cdot)} \left[\underbrace{\int_t^{t+\delta} C\big(x(s),u(s)\big)ds}_{C\left(x(t),u(t)\right)\delta} + \underbrace{V\big(\overbrace{x(t+\delta)}^{x(t)+\delta f(x,u)},t+\delta\big)}_{V(x(t),t) + \frac{\partial V}{\partial x}\cdot \delta f(x(t),u(t)) + \frac{\partial V}{\partial t}\delta} \right]$$

  - Omit t dependence…

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

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

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

- 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

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

- 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

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

- 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

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

- 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