Lecture 1: Dynamic Programming

General problem
Multivariable derivative
Discrete-time LQ

Dynamic programming (DP)

introduction:

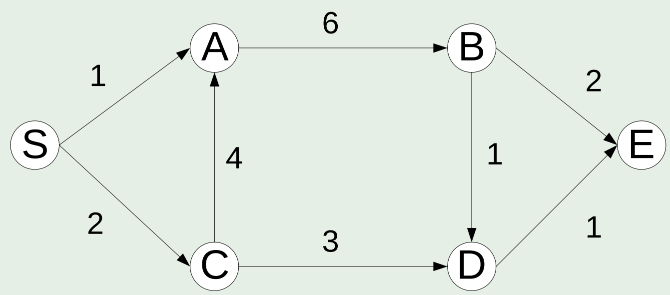
- history: developed in the 1950's by Richard Bellman
- "programming": ∼"planning" (has nothing to do with computers)
- a useful concept with lots of applications
- IEEE Global History Network: "A breakthrough which set the stage for the application of functional equation techniques in a wide spectrum of fields..."

Essentials of dynamic programming

- key idea: solve a complex and difficult problem via solving a collection of sub problems

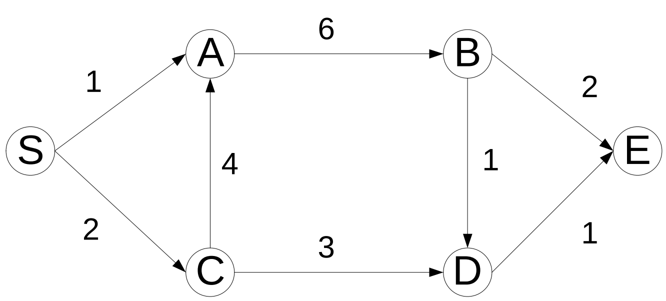
Example (Path planning)

goal: obtain minimum cost path from S to E



- observation: if node C is on the optimal path, the then path from node C to node E must be optimal as well

Essentials of dynamic programming



$$dist\,(E) \triangleq \text{minimum cost } S \to E$$

- solution:

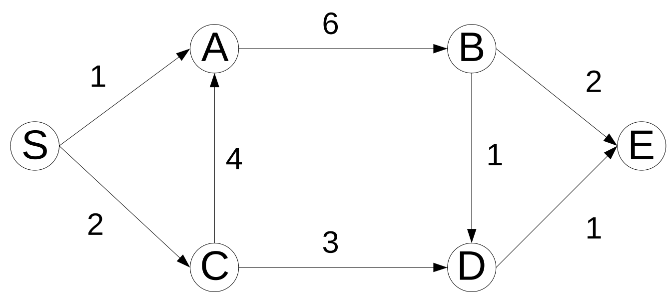
backward analysis

$$\begin{aligned}
dist\,(E) &= \min\{dist\,(B)+2, dist\,(D)+1\}\\
dist\,(B) &= dist\,(A)+6\\
dist\,(D) &= min\{dist\,(B)+1, dist\,(C)+3\}\\
dist\,(C) &= 2\\
dist\,(A) &= \min\{1, dist\,(C)+4\}
\end{aligned}$$

forward computation

$$\begin{aligned}
dist\,(C) &= 2\\
dist\,(A) &= 1\\
dist\,(B) &= 1+6=7\\
dist\,(D) &= 5\\
dist\,(E) &= 6
\end{aligned}$$

Essentials of dynamic programming



- summary (Bellman's principle of optimality): "From any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point."

General optimal control problems

- general discrete-time plant:

$$
\begin{aligned}
x(k+1) &= f(x(k), u(k), k) \\
\text{state constraint: } x(k) &\in X \subset \mathbf{R}^n \\
\text{input constraint: } u(k) &\in U \subset \mathbf{R}^m
\end{aligned}
$$

- performance index:

$$J = S(x(N)) + \sum_{k=0}^{N-1} L(x(k), u(k), k)$$

S & L–real, scalar-valued functions; N–final time (optimization horizon)

- goal: obtain the optimal control sequence

$$\{u^o(0), u^o(1), \ldots, u^o(N-1)\}$$

Dynamic programming for optimal control

- define: $U_k \triangleq \{u(k), u(k+1), \dots, u(N-1)\}$
- optimal cost to go at time k:

$$
\begin{aligned}
J_k^o(x(k)) &\triangleq \min_{U_k} \left\{ S(x(N)) + \sum_{j=k}^{N-1} L(x(j), u(j), j) \right\} \\
&= \min_{u(k)} \min_{U_{k+1}} \left\{ L(x(k), u(k), k) + \left[S(x(N)) + \sum_{j=k+1}^{N-1} L(x(j), u(j), j) \right] \right\} \\
&= \min_{u(k)} \left\{ L(x(k), u(k), k) + \min_{U_{k+1}} \left[S(x(N)) + \sum_{j=k+1}^{N-1} L(x(j), u(j), j) \right] \right\} \\
&= \min_{u(k)} \left\{ L(x(k), u(k), k) + J_{k+1}^o(x(k+1)) \right\} \qquad (1)
\end{aligned}
$$

- boundary condition: $J_N^o(x(N)) = S(x(N))$
- The problem can now be solved by solving a sequence of problems $J_{N-1}^o,\ J_{N-2}^o, \dots, J_1^o,\ J^o$.

Solving discrete-time finite-horizon LQ via DP

- system dynamics:

$$
x(k+1) = A(k)x(k) + B(k)u(k),\ x(k_0) = x_o \qquad (2)
$$

- performance index:

$$
J = \frac{1}{2}x^T(N)Sx(N) + \frac{1}{2}\sum_{k=k_0}^{N-1}\left\{ x^T(k)Q(k)x(k) + u^T(k)R(k)u(k) \right\}
$$

$$
Q(k) = Q^T(k) \succeq 0,\ S = S^T \succeq 0,\ R(k) = R^T(k) \succ 0
$$

- optimal cost to go:

$$
J_k^o(x(k)) = \min_{u(k)} \left\{ \frac{1}{2}x^T(k)Q(k)x(k) + \frac{1}{2}u^T(k)R(k)u(k) + J_{k+1}^o(x(k+1)) \right\}
$$

with boundary condition: $J_N^o(x(N)) = \frac{1}{2}x^T(N)Sx(N)$

Facts about quadratic functions

- consider
$$f(u) = \frac{1}{2}u^T M u + p^T u + q,\ M = M^T \tag{3}$$
- optimality (maximum when M is negative definite; minimum when M is positive definite) is achieved when
$$\frac{\partial f}{\partial u} = Mu^o + p = 0 \Rightarrow u^o = -M^{-1}p \tag{4}$$
- and the optimal cost is
$$f^o = f(u^o) = -\frac{1}{2}p^T M^{-1} p + q \tag{5}$$

From J_N^o to J_{N-1}^o in discrete-time LQ

- by definition:
$$J_{N-1}^o(x(N-1)) = \min_{u(N-1)} \left\{ \frac{1}{2}x^T(N)Sx(N) \right.$$
$$\left. + \frac{1}{2}\left[x^T(N-1)Q(N-1)x(N-1) + u^T(N-1)R(N-1)u(N-1)\right] \right\}$$
- using the system dynamics (2) gives
$$J_{N-1}^o(x(N-1)) = \frac{1}{2}\min_{u(N-1)}\{x^T(N-1)Q(N-1)x(N-1)$$
$$+ u^T(N-1)R(N-1)u(N-1) + [A(N-1)x(N-1) + B(N-1)u(N-1)]^T$$
$$\times S[A(N-1)x(N-1) + B(N-1)u(N-1)]\}$$
- optimal control by letting $\partial J_{N-1}/\partial u(N-1) = 0$:
$$u^o(N-1) = -\underbrace{\left[R(N-1) + B^T(N-1)SB(N-1)\right]^{-1} B^T(N-1)SA(N-1)}_{\text{state feedback gain: } K(N-1)}x(N-1)$$

$\star$Optimality at N and $N-1$

at time N: optimal cost is

$$J_N^o(x(N)) = \frac{1}{2}x^T(N)Sx(N) \triangleq \frac{1}{2}x^T(N)P(N)x(N)$$

at time $N-1$:

$$J_{N-1}^o(x(N-1)) = \frac{1}{2}\min_{u(N-1)}\{x^T(N-1)Q(N-1)x(N-1)$$
$$+u^T(N-1)R(N-1)u(N-1)+[A(N-1)x(N-1)+B(N-1)u(N-1)]^T$$
$$\times S[A(N-1)x(N-1)+B(N-1)u(N-1)]\}$$

optimal cost to go [by using (5)] is

$$J_{N-1}^o(x(N-1)) = \frac{1}{2}x^T(N-1)\Big\{Q(N-1)+A^T(N-1)SA(N-1)$$
$$-(\underline{\ldots})^T\left[R(N-1)+B^T(N-1)SB(N-1)\right]^{-1}\underline{B^T(N-1)SA(N-1)}\Big\}x(N-1)$$
$$\triangleq \frac{1}{2}x^T(N-1)P(N-1)x(N-1)$$

Summary: from N to $N-1$

at N:

$$J_N^o(x(N)) = \frac{1}{2}x^T(N)Sx(N) = \frac{1}{2}x^T(N)P(N)x(N)$$

at $N-1$:

$$J_{N-1}^o(x(N-1)) = \frac{1}{2}x^T(N-1)P(N-1)x(N-1)$$

with (S has been replaced with $P(N)$ here)

$$P(N-1) = Q(N-1)+A^T(N-1)P(N)A(N-1)$$
$$-(\underline{\ldots})^T\left[R(N-1)+B^T(N-1)P(N)B(N-1)\right]^{-1}\underline{B^T(N-1)P(N)A(N-1)}$$

and state-feedback law

$$u^o(N-1) = -\left[R(N-1)+B^T(N-1)P(N)B(N-1)\right]^{-1}$$
$$\times B^T(N-1)P(N)A(N-1)x(N-1)$$

Induction from $k+1$ to k

- assume at $k+1$:

$$J_{k+1}^{o}\left(x\left(k+1\right)\right) = \frac{1}{2}x^{T}\left(k+1\right)P\left(k+1\right)x\left(k+1\right)$$

- analogous as the case from N to $N-1$, we can get, at k:

$$J_{k}^{o}\left(x\left(k\right)\right) = \frac{1}{2}x^{T}\left(k\right)P\left(k\right)x\left(k\right)$$

  with Riccati equation

$$P\left(k\right) = A^{T}\left(k\right)P\left(k+1\right)A\left(k\right) + Q\left(k\right)$$
$$-A^{T}\left(k\right)P\left(k+1\right)B\left(k\right)\left[R\left(k\right)+B^{T}\left(k\right)P\left(k+1\right)B\left(k\right)\right]^{-1}B^{T}\left(k\right)P\left(k+1\right)A\left(k\right)$$

  and state-feedback law

$$u^{o}\left(k\right) = -\left[R\left(k\right)+B^{T}\left(k\right)P\left(k+1\right)B\left(k\right)\right]^{-1}B^{T}\left(k\right)P\left(k+1\right)A\left(k\right)x\left(k\right)$$

Implementation

- optimal state-feedback control law:

$$u^{o}\left(k\right) = -\left[R\left(k\right)+B^{T}\left(k\right)P\left(k+1\right)B\left(k\right)\right]^{-1}B^{T}\left(k\right)P\left(k+1\right)A\left(k\right)x\left(k\right)$$

- Riccati equation:

$$P\left(k\right) = A^{T}\left(k\right)P\left(k+1\right)A\left(k\right) + Q\left(k\right)$$
$$-A^{T}\left(k\right)P\left(k+1\right)B\left(k\right)\left[R\left(k\right)+B^{T}\left(k\right)P\left(k+1\right)B\left(k\right)\right]^{-1}B^{T}\left(k\right)P\left(k+1\right)A\left(k\right)$$

  with the boundary condition $P\left(N\right) = S$.
- $u^{o}(k)$ depends on
  - the state vector $x\left(k\right)$
  - system matrices $A(k)$ and $B(k)$ and the cost matrix $R(k)$
  - $P(k+1)$, which depends on $Q(k+2)$, $A(k+1)$, $B(k+1)$, and $P(k+2)$...
- iterating gives: $u(0)$ depends on $\{A(k), B(k), R(k), Q(k+1)\}_{k=0}^{N-1}$
  In practice, $P(k)$ can be computed offline since they do not require information of $x(k)$.