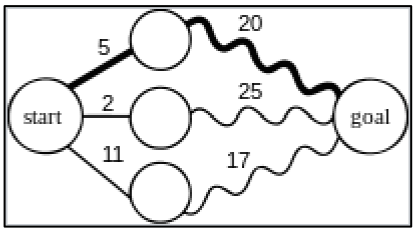
Summary of Intelligent Robotics class of 18/10/10

성기호, Peter Muschick, 김유진, 안정도

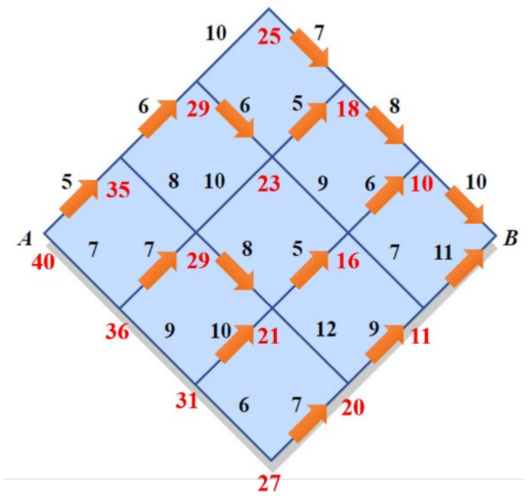
Dynamic Programming

Dynamic programming is both a mathematical optimization method and a computer programming method. The method was developed by Richard Bellman in the 1950s and has found applications in numerous fields, from aerospace engineering to economics. In both contexts it refers to simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner. While some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart recursively. Likewise, in computer science, if a problem can be solved optimally by breaking it into sub-problems and then recursively finding the optimal solutions to the sub-problems, then it is said to have optimal substructure [1].



Finding the shortest path in a graph using optimal substructure; a straight line indicates a single edge; a wavy line indicates a shortest path between the two vertices it connects (among other paths, not shown, sharing the same two vertices); the bold line is the overall shortest path from start to goal.

Ex) Multistage Optimization



- Need optimal path from A to B

① left-to-right only

② “stage” node location

③ “control” up, down (binary)

Sol) Calculate backward for each node

Limitation : Curse of dimensionality

Dynamic Programming

Input : State x, Control input (action) u, Horizon time T

$\dot{x} = f(\mathrm{x,u,t})$, Instantaneous cost $g(\mathrm{x,u,t})$, & Terminal cost $h(\mathrm{x})$

Output : Optimal policy (optimal control) $\prod^*(\mathrm{x,t})$ ex) up/down

Optimal cost-to-go $J^*(\mathrm{x,t})$ ex) 40, 35, 29, …

Expressed as a algorithm,

```
for each $x \in \underline{\overline{\mathrm{X}}}$, do
    $J^*(\mathrm{x,T}) \leftarrow h(\mathrm{x})$
end
for $\mathrm{t} \leftarrow T-1$ to O do
    for each x do
        $J^*(\mathrm{x}_t,\mathrm{t}) \leftarrow \min_u(\mathrm{g}(\mathrm{x}_t,\mathrm{u,t}) + J^*(\mathrm{x}_{t+1},\mathrm{t}+1))$
        $\prod^*(\mathrm{x}_t,\mathrm{t}) \leftarrow \arg\min_u(\mathrm{g}(\mathrm{x}_t,\mathrm{u,t}) + J^*(\mathrm{x}_{t+1},\mathrm{t}+1))$
    end
end
```

Optimal control formulation

$$\min_u \phi(\mathrm{x}(\mathrm{t}_1),\mathrm{t}_1) + \int_t^{t_1} L(\mathrm{x,u,t})\,\mathrm{dt}$$

$$s.t.\ \mathrm{x} = f(\mathrm{x,u,t}), \text{initial condition} : (\mathrm{t,x(t)})$$

$$\mathrm{V(x,t)} = \text{cost for optimal path starting at } (\mathrm{x,t})$$

$$= \min_u \int_t^{t_1} Ldt + \phi(\mathrm{x}(\mathrm{t}_1),\mathrm{t}_1)$$

$$= \left\{ \min_u \int_t^{t+\Delta t} Ldt + \underbrace{\int_{t+\Delta t}^{t_1} Ldt + \phi(\mathrm{x}(\mathrm{t}_1),\mathrm{t}_1)}_{V(x+\Delta x, t+\Delta t)} \right\}$$

$$= \left\{ L\Delta t + 0(\Delta\,\mathrm{t}) + \mathrm{V}(x+\Delta x, t+\Delta t) \right\}$$

$(\because \Delta\,\mathrm{t}$ is very small)

$$\cancel{V(x,t)} = \min_u \left\{ L\Delta t + \cancel{V(x,t)} + V_x \Delta t + V_t \Delta t + 0(\Delta t) \right\}$$

$$0 = \min_u \left\{ L + V_x \frac{\Delta x}{\Delta t} + V_t \frac{0(\Delta t)}{\Delta t} \right\}$$

$$\Delta t \to 0$$

$$-V_t = \min_u L + V_x \dot{x}$$

$$-\frac{\partial V}{\partial t} = \min_u L + \frac{\partial V}{\partial x} f$$

$\Rightarrow$ Hamilton-Jacobi-Bellman (HJB) eq.

with boundary condition, $V(x(t_1),t_1)=\phi(x(t_1),t_1)$

$$\min_u \frac{1}{2} X^T Q X \,|_{t_1} + \frac{1}{2}\int_{t_0}^{t_1} \left(X^T Q X + u^T R u \right) dt$$

s.t. $x = Ax + Bu$

$h(x) = X^T Q X \,|_{t_1} = V\left(x(t_1), t_1\right)$: terminal cost

$g(x,u,t) = \frac{1}{2}\left(X^T Q X + u^T R u \right)$: instantaneous cost

by HJB,

$$0 = \min_u \left[\frac{1}{2}\left(X^T Q X + u^T R u \right) + V_x^* \left(Ax + Bu \right) + V_t^* \right]$$

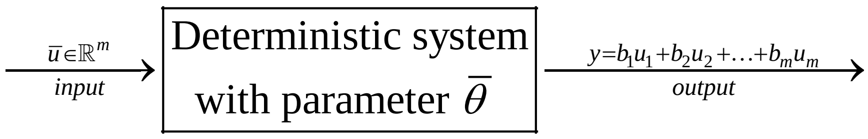
$$\left(\begin{array}{l} \frac{\partial}{\partial u} = 0, u^T R + V_x^* B = 0 \\ \Rightarrow u = -R^{-1} B^T V_x^{*T} = -R^{-1} B^T K x \end{array} \right)$$

$$0 = \frac{1}{2} X^T Q X - \frac{1}{2} V_x^* B R^{-1} B^T V_x^* + V_x^* A x + V_t^*$$

Let's try $V^*\left(x,t\right) = \frac{1}{2} x^T K(t) V,\ K = K^T > 0$

Least Squares Estimation

The method of **least squares** is a standard approach in regression analysis to approximate the solution of overdetermined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the residuals made in the results of every single equation [2].


$\bar{u} \in \mathbb{R}^m$
input
Deterministic system with parameter $\bar{\theta}$
$y = b_1 u_1 + b_2 u_2 + \ldots + b_m u_m$
output

Linearly parameterized model

$$\bar{\theta} = [b_1, \cdots, b_m] \in \mathbb{R}^m$$

where $\bar{\theta}$ is to be estimated.

System could be

① Linear dynamic system

e.g.

$$y(\mathrm{t}) = \mathrm{b}_1 u(\mathrm{t}-1) + \mathrm{b}_2 u(\mathrm{t}-2) + \cdots + \mathrm{b}_m u(\mathrm{t}-m)$$
$$u = [u(\mathrm{t}-1)\, u(\mathrm{t}-2) \cdots u(\mathrm{t}-m)]^T$$

② Nonlinear dynamic system

e.g.

$$y(\mathrm{t}) = \mathrm{b}_1 u(\mathrm{t}-1) + \mathrm{b}_2 u^2(\mathrm{t}-1) u(\mathrm{t}-2)$$
$$u = [u(\mathrm{t}-1)\ u(\mathrm{t}-1) u(\mathrm{t}-2)]^T$$

$\hat{y}(t) = \bar{u}^T \hat{\theta}$

$\mathrm{V}_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}\left(\hat{y}(\mathrm{t}) - \mathrm{y}(\mathrm{t})\right)^2$: square error

$\hat{\theta} = \arg\min_{\theta} \mathrm{V}_N(\theta)$

$\frac{\partial \mathrm{V}_N}{\partial \theta} = 0$

$\Rightarrow \frac{2}{N}\sum_{t=1}^{N}\left(\bar{u}^T(\mathrm{t})\bar{\theta} - \mathrm{y}(\mathrm{t})\right)\bar{u}(\mathrm{t}) = 0$

$\underbrace{\sum_{t=1}^{N}\left(u(\mathrm{t})\,\mathrm{M}^T(\mathrm{t})\bar{\theta}\right)}_{UU^T} = \sum_{t=1}^{N} y(t)\bar{u}(t)$

where $U = \left[u(1)\,u(2)\cdots u(N)\right]^{m\times N}$ $(m \leq \mathrm{N})$.

If $u(1),\cdots u(\mathrm{N})$ span the whole m-dim. vector space,

rankU = m : full rank

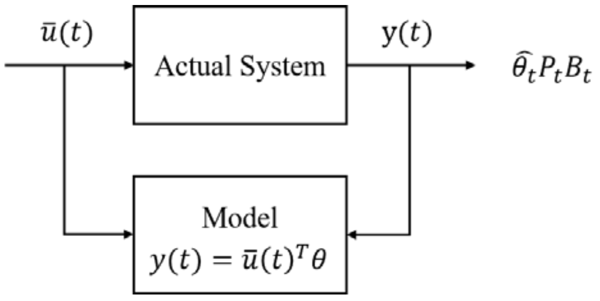
If $^{\exists}m$ linearly independent vectors in this U,

rankUU^T $= m$

$\hat{\theta} = P \cdot B$

where $P = \left[\sum_{t=1}^{N}\bar{u}\,\bar{u}^T\right]^{-1} = \left[UU^T\right]^{-1},\ b = \sum y\bar{u}.$

Recursive Least-Squares Alg.



$\hat{\theta} = PB$ Batch Processing

On-line estimation based on $\{y(t), u(t)\}$ the least data update the estimation $\hat{\theta}$.

Recursive formula

- Simple to complete with a given sample period
- No need to state the whole data

References

[1] https://en.wikipedia.org/wiki/Dynamic_programming

[2] https://en.wikipedia.org/wiki/Least_squares