1.1 Introduction

Dynamic Programming : Sequential decision making

two principal features :

* discrete - time dynamic system
* cost additive overtime.

"state" :

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \cdots, N-1.$$

* k : times
* x_k : state , summarize past information that is relevent for future optimization
* u_k : control / decision variable to be selected at time k.
* w_k : random parameter
* N : number of times control is applied.
* f_k : help the system state to update.

"cost" :

in period k : $g_k(x_k, u_k, w_k)$

total cost : $g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)$

expected cost (optimize) : $\underset{w_0 \cdots w_{N-1}}{E}\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)\right]$

Take Expectation over $\{w_0, \cdots, w_{N-1}\}$.

"optimization" : $\{u_0, \cdots, u_{N-1}\}$.

u_k is selected with some knowledge of x_k.

Example 1.1.1 (Inventory control).

Background : * order quantity at each of N periods to meet stochastic demand

* target to minimize expected cost

DP : x_k : stock available at the beginning of period k.

u_k : stock ordered (immediately delivered) at the beginning of period k.

w_k : demand in $k \sim F_k$.

$w_0, \cdots, w_{N-1}$ ind. exceeded demand $\Rightarrow$ backlogged.

state transition :

$x_{k+1} = x_k + u_k - w_k$ (backlogged). $x_{k+1} = (x_k + u_k - w_k)^+$ (lost sale)

cost :

i) $r(x_k) = \left[\begin{array}{ll} \text{holding cost} & x_k > 0 \\ \text{backlog cost} & x_k < 0 \end{array}\right.$ (lost sale : lost sale panelty).

ii) purchase cost : cu_k , c : cost per unit ordered

expected total cost : $E\{ R(x_N) + \sum_{i=1}^{N-1} (r(x_k) + cu_k) \}$.

$\xrightarrow{\text{optimize}}$ $(u_0, \cdots, u_{N-1})$

Open-loop optimization : at time 0 : select $u_0, \cdots, u_{N-1}$.

closed-loop optimization : postpone u_k until x_k is known (additional information)

DP : closed-loop optimization

cost-to-go function :

$\mu_k(x_k)$: $x_k \to u_k$: amount order at time k if stock is x_k.

$\pi : \{\mu_0, \mu_1, \cdots, \mu_{N-1}\}$: a policy.

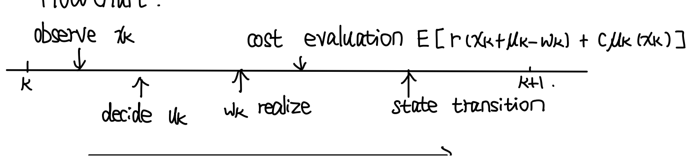
$J_\pi(x_0) = E\{ R(x_N) + \sum_{k=0}^{N-1} (r(x_k) + c\mu_k(x_k)) \}$.

DP : $\underset{\pi \in \Pi}{\text{minimize}}\ J_\pi(x_0)$.

$\Rightarrow \quad \mu_k(x_k) = \left[\begin{array}{ll} S_k - x_k & \text{if } x_k < S_k. \\ 0 & \text{otherwise.} \end{array}\right.$

S_k : threshold $\sim F$, holding penality, c, ...

DP Flow chart :



$\xrightarrow{\hspace{3cm}}$

sequence of events.

1.2. Discrete – State and Finite – State Problems:

x_k : discrete and finite.

$P_{ij}(u,k)$: prob at time k. next state will be j given current state is i and control is u.

$$P_{ij}(u,k) = P(x_{k+1}=j \mid x_k = i,\ u_k = u).$$

$$x_{k+1} = w_k$$

$$\Rightarrow P(w_k = j \mid x_k = i,\ u_k = u) = P_{ij}(u,k)$$

1.3 Discrete time Dynamic System

$$x_{k+1} = f_k(x_k,\ u_k,\ w_k). \qquad k = 0, 1, \cdots, N-1.$$

$$x_k \in S_k \ ,\ u_k \in C_k \ ,\ w_k \in D_k.$$

u_k : constrainted $u_k \in U(x_k) \subset C_k$.

w_k : characterized by a probability distribution $P_k(\cdot \mid x_k, u_k)$. depend on x_k. u_k. but not on $w_{k-1}, \cdots, w_0$

$\pi : \{\mu_0, \cdots, \mu_{N-1}\}$: policy / control law.

$\mu_k : x_k \longmapsto u_k = \mu_k(x_k) \in U(x_k)$. (admissible).

Given initial state x_0 and an admissible policy π:

$$x_{k+1} = f_k(x_k.\ u_k.\ w_k). \quad k = 0, 1, \cdots, N-1.$$

Given cost function : g_k , $k = 0, 1, \cdots, N$, expected loss of π starting at x_0 :

$$J_\pi(x_0) = E\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k.\ \mu_k(x_k)\ ,\ w_k) \}.$$

Expectation takes over w_k. x_k.

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0).$$

Π : set of all admissible policies.

π^* : optimal policy given x_0 (target).

In many cases. we want to find π^* for all intial states.

$J^*(x_0) = J_{\pi^*}(x_0)$: optimal cost function

1.4. Principle of Optimality.

$\pi^* = \{\mu_0^*, \mu_1^*, \cdots, \mu_{N-1}^*\}$. an optimal policy (不一定只有一个).

Assume using π^*. x_i occur with positive probability. then we have subproblem:

we are at x_i at time i, and we wish to minimize cost-to-go from time i to time N: $\min_{\mu_i \cdots \mu_{N-1}} E\{g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\}$.

the truncated policy of $\pi^* = \{\mu_i^*, \mu_{i+1}^*, \cdots, \mu_{N-1}^*\}$. is optimal for this subproblem.

1.5. The DP Algorithm.

$*$ Prop 1.3.1. For every state x_0, $J^*(x_0) = J_0(x_0)$, given by the last step of the Algorithm, which proceeds the backward in time for $N-1$ to 0.

$$J_N(x_N) = g_N(x_N).$$

$$J_k(x_k) = \min_{u_k \in U(x_k)} E_{w_k}\{g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))\}. \quad (*)$$

$$k = 0, 1, \cdots, N-1.$$

Expectation taken over w_k which depends on x_k, u_k. Furthermore, $u_k^* = \mu_k^*(x_k)$ minimize R.H.S of $(*)$ for each x_k and k, then $\pi^* = (\mu_0^*, \cdots, \mu_{N-1}^*)$ is optimal.

Example 1.1.5 (Optimize a chess Match strategy).

Backgrounds: $*$ chess player plays two rounds of chess match with an oppent. we need to maximize his chance of winning.

$*$ Each game has one of two outcomes:

a) A win by one of the players (1 point for the winner, 0 for the loser)

b) A draw (0.5 point for each player).

$*$ If score tied at 1-1 at the end of two games, match goes into sudden death mode. Continue to play while first time one of the player wins game.

$*$ The player has two playing styles, and can choose one in each game independently. ($P_d > P_w$)

1). Timid play. draw w.p. $P_d > 0$ } never win.
lose w.p. $1-P_d > 0$

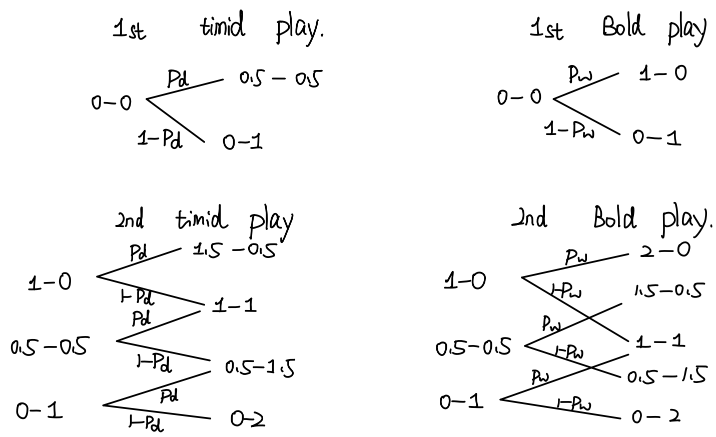
2) Bold play. win w.p. $P_w > 0$ } never draw.
lose w.p. $1-P_w > 0$

Once in sudden-death mode, player's optimal Policy: Bold.

* Player has two decisions to make:

1. what to play in 1st game
2. what to play in 2nd game.

Analysis:



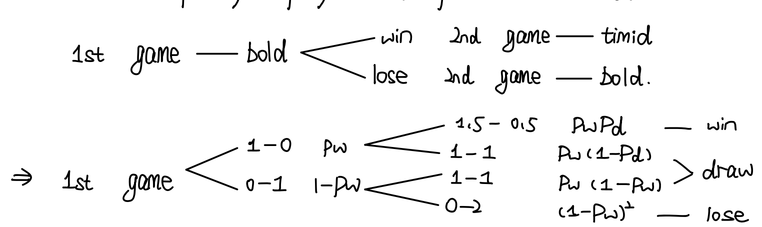
2-0, 1.5-0.5. ⇒ win ⇒ cost -1

0.5-1.5, 0-2 ⇒ lose ⇒ cost 0.

1-1 ⇒ draw ⇒ cost $-P_w$.

⇒ minimize $-P(2\text{-}0,\ 1.5\text{-}0.5) - P_w(1\text{-}1) \Leftrightarrow$ maximize $P(2\text{-}0,\ 1.5\text{-}0.5) + P_w P(1\text{-}1)$

Now Consider policy: play timid if he is ahead in score.



Prob(win match) $= P_w P_d + [P_w(1-P_d) + P_w(1-P_w)]\, P_w$

$= P_w^2 (2-P_w) + P_w(1-P_w) P_d.$

$\Rightarrow$ If $P_w < \frac{1}{2}$: chance of winning a game < 0.5

but not mean lose the math w.p. > 0.5

so long as $P_w \to 0.5$. $P_d \to 1$.

Value of information :

* open-loop solution :

1) TT : prob to win : $P_d^2 P_w$

2) BB : prob to win : $P_w^3 + 2P_w^2(1-P_w)$

3) BT / TB : prob to win : $P_w P_d + P_w^2 (1-P_d)$.

$\Rightarrow$ prob to win : $\max\{P_w^2(3-2P_w),\ P_w P_d + P_w^2(1-P_d)\}$.

if $P_d > 2P_w$, then TB is optimal

DP Algorithm to show optimality.

state : net score = player's point $-$ opponent's point

$J_k(x_k) = \max[P_d J_{k+1}(x_k) + (1-P_d) J_{k+1}(x_k-1),\ P_w J_{k+1}(x_k+1) + (1-P_w) J_{k+1}(x_k-1)]$

If.

$P_w J_{k+1}(x_k+1) + (1-P_w) J_{k+1}(x_k-1) > P_d J_{k+1}(x_k) + (1-P_d) J_{k+1}(x_k-1) \Rightarrow$ Bold.

$\Leftrightarrow \quad \frac{P_w}{P_d} \geqslant \frac{J_{k+1}(x_k) - J_{k+1}(x_k-1)}{J_{k+1}(x_{k+1}) - J_{k+1}(x_k-1)}$

$$J_N(x_N) = \begin{cases} 1 & \text{if } x_N > 0 \\ P_w & \text{if } x_N = 0 \\ 0 & \text{if } x_N < 0 \end{cases}$$

$J_{N-1}(x_{N-1}) = 1$ if $x_{N-1} > 1$. optimal play either.

$J_{N-1}(1) = \max\{P_d + (1-P_d)P_w,\ P_w + (1-P_w)P_w\}$.

$= P_d(1-P_d)P_w \quad \Rightarrow$ optimal play : timid.

$J_{N-1}(0) = P_w \quad \Rightarrow$ optimal : Bold.

$J_{N-1}(-1) = P_w^2 \Rightarrow$ optimal : Bold.

$J_{N-1}(x_{N-1}) = 0$, $x_{N-1} < -1$. optimal : either.

Given $J_{N-1}(x_{N-1})$

$$J_{N-2}(0) = \max\{ P_d P_w + (1-P_d) P_w^2,\ P_w(P_d + (1-P_d)P_w) + (1-P_w)P_w^2 \}.$$

$$= P_w(P_w + (P_w + P_d)(1-P_w))$$

If score even with 2 games remaining, opt: Bold.

Thus, 2-game match optimal policy for both period:

B to play. T iff player is ahead in the score.

The region of pairs (P_w, P_d) for which the player has a bettern than 50% chance to win a 2-game match is

$R = \{(P_w, P_d) \mid J_0(0) = P_w(P_w + (P_w + P_d)(1-P_w)) > \frac{1}{2}\}$. (includes point $P_w < \frac{1}{2}$).