Il concetto di stato nella Programmazione Dinamica ed Equazione di Bellman
a.a. 2024/2025

Mario Sassano

Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università di Roma Tor Vergata

Nelle lezioni precedenti...

- Abbiamo introdotto la Programmazione Dinamica
- Abbiamo visto che, per il Principio di Ottimalità, particolari *sotto-porzioni* della soluzione ottima devono costituire le soluzioni ottime corrispondenti a problemi *ristretti*
- Abbiamo interpretato il Problema del Cammino Minimo come un caso speciale di DP distinguendo tra l'equazione funzionale di DP e gli algoritmi che permettono di risolverla

Il problema dello "zainetto"

Knapsack

- consideriamo "zainetto" di capacità (volume) K
- possiamo riempire lo zaino con un numero intero di oggetti x^i, $i = 1, \ldots, N$ (N tipi)
- ogni oggetto ha volume v^i e valore c^i

Obiettivo: massimizzare il valore dello zainetto

$$\mathcal{P}(N, K) := \begin{cases} \max & \sum_{i=1}^{N} c^i x^i \\ s.t. & \sum_{i=1}^{N} v^i x^i \leqslant K \\ & x^i \geqslant 0, i = 1, \ldots, N \\ & x^i \in \mathbb{N}, i = 1, \ldots, N \end{cases}$$

Interpretazione come ottimizzazione dinamica

1. si decide incrementalmente se aggiungere o meno l'oggetto x^1, x^2, ...
2. aggiungendo l'oggetto x^i si occupa spazio nello zainetto
3. si ripete dal passo 1.

la scelta di aggiungere l'oggetto x^i influenza la capacità di inserire oggetti "in futuro"

Knapsack

- consideriamo "zainetto" di capacità (volume) K
- possiamo riempire lo zaino con un numero intero di oggetti x^i, $i = 1, \dots, N$ (N tipi)
- ogni oggetto ha volume v^i e valore c^i

Obiettivo: massimizzare il valore dello zainetto

$$\mathcal{P}(N,K) := \begin{cases} \max & \sum_{i=1}^{N} c^i x^i \\ s.t. & \sum_{i=1}^{N} v^i x^i \leqslant K \\ & x^i \geqslant 0, i = 1, \dots, N \\ & x^i \in \mathbb{N}, i = 1, \dots, N \end{cases}$$

La soluzione *algoritmica* classica prevede di risolvere NK sottoproblemi $\mathcal{P}(r,\lambda)$ con $r \leqslant N$ oggetti e volume $\lambda \leqslant K$...

Per formulare il problema con DP, introduciamo una variabile di **stato**

y : **volume libero** dello zainetto

Inizialmente[a]

$y_0 = K$ (zaino vuoto con tutto lo spazio a disposizione...)

Mano mano che vengono aggiunti oggetti ($v_k \in \{v^1, v^2, \dots, v^N\}$)

$y_{k+1} = y_k - v_k$ (a patto che ci sia spazio sufficiente, $v_k \leqslant y_k$, ...)

[a] Notazione: **apice** indica tipologia di oggetto $x^1, x^2, \dots$, **pedice** indica evoluzione temporale y_k.

Sistema sostituisce *dinamicamente* il vincolo di capacità massima

Scriviamo l'equazione funzionale di DP

$V(y)$: **valore massimo ottenibile** dato il volume y (quindi $V(0) = 0$)

$$V(y) = \max_{v_k \leqslant y}\{ \underbrace{c_k}_{\text{guadagno attuale}} + \underbrace{V(y - v_k)}_{\text{effetto su futuro}} \}, \quad \forall y \in \{1, \dots, K\}$$

Massimizzare il valore aggiungendo un oggetto alla volta (c_k) ma considerando anche l'effetto futuro ($V(y - v_k)$) della scelta

Esempio Knapsack

- x^0: spazio vuoto, $v^0 = 1$, $c^0 = 0$
- x^1: cibo, $v^1 = 2$, $c^1 = 2$
- x^2: attrezzatura, $v^2 = 3$, $c^2 = 1$

$$\begin{cases} \max & 2x^1 + x^2 \\ s.t. & x^0 + 2x^1 + 3x^2 \leqslant 9 \\ & x^i \geqslant 0, i = 1, \ldots, N \\ & x^i \in \mathbb{N}, i = 1, \ldots, N \end{cases}$$

Risolviamo l'equazione funzionale di DP per determinare $V(9)$:

$$V(0) = 0$$
$$V(1) = \max_{k \in \{0,1,2\}, v_k \leqslant 1} \{c_0 + V(1 - v_0)\} = 0$$
$$V(2) = \max_{v_k \leqslant 2} \{c_0 + V(2-1), c_1 + V(2-2)\} = 2$$
$$V(3) = \max_{v_k \leqslant 3} \{0 + V(3-1), 2 + V(3-2), 1 + V(3-3)\} = 2$$
$$V(4) = \max_{v_k \leqslant 4} \{0 + V(4-1), 2 + V(4-2), 1 + V(4-3)\} = 4$$
$$V(5) = \max_{v_k \leqslant 5} \{0 + V(5-1), 2 + V(5-2), 1 + V(5-3)\} = 4$$
$$V(6) = \max_{v_k \leqslant 6} \{0 + V(6-1), 2 + V(6-2), 1 + V(6-3)\} = 6$$
$$V(7) = \max_{v_k \leqslant 7} \{0 + V(7-1), 2 + V(7-2), 1 + V(7-3)\} = 6$$
$$V(8) = \max_{v_k \leqslant 8} \{0 + V(8-1), 2 + V(8-2), 1 + V(8-3)\} = 8$$
$$V(9) = \max_{v_k \leqslant 9} \{0 + V(9-1), 2 + V(9-2), 1 + V(9-3)\} = 8$$

Formulazione generale

Processo decisionale multi-stage

$$\begin{cases} \min\limits_{u_1,\dots,u_{N-1}} & \sum\limits_{k=0}^{N-1} g_k(x_k, u_k) + g_N(x_N) \\ s.t. & x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \dots, N-1 \\ & u_k \in U_k \\ & x_k \in X \end{cases}$$

- k indice temporale
- N orizzonte temporale
- x_k stato del sistema
- u_k controllo da selezionare
- g_k costo corrente, g_N costo terminale
- U_k insieme di controlli ammissibili al tempo k
- X di cardinalità finita (per il momento...) $\Rightarrow$ numero finito di possibili stati

Introduciamo la **Funzione Valore**

$V_k(x_k)$: *costo migliore ottenibile dal passo k fino ad N*
data la condizione "iniziale" x_k al tempo k

Equazione di Bellman - equazione funzionale per V

$$\begin{cases} V_k(x_k) & = \min\limits_{u_k \in U_k} \{ g_k(x_k, u_k) + V_{k+1}(\underbrace{f(x_k, u_k)}_{x_{k+1}}) \}, \forall^a x_k \in X, k = \{0, 1, \ldots, N-1\} \\ V_N(x_N) & = g_N(x_N), \forall x_N \in X \end{cases}$$

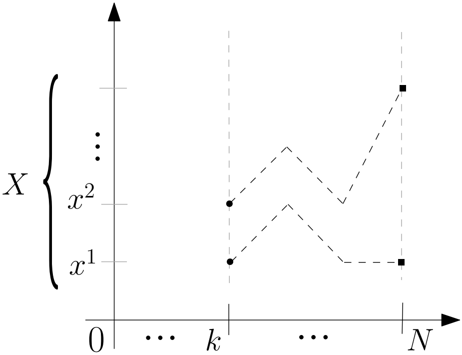
[a] *a priori* non sappiamo in quale stato si troverà la soluzione ottima all'istante k...

Dopo aver calcolato il *valore di lungo periodo* di ciascuno stato, l'azione ottima al tempo k è quella che trasferisce il sistema nello stato a "minor costo" al tempo $k+1$

$$u_k^\star(x_k) = \arg \min_{u_k \in U_k} \{ g_k(x_k, u_k) + V_{k+1}(x_{k+1}) \},$$

Soluzione ricorsiva dell'equazione di Bellman (1/4)

Proviamo a risolvere l'equazione ricorsivamente "in avanti"

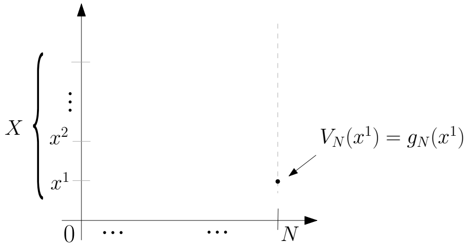


$\Rightarrow$ due sottoproblemi da x^1 e x^2 al tempo k, identici al problema originale su orizzonte temporale ristretto

$\Rightarrow$ non possiamo riutilizzare nulla di quello che calcoliamo da x^1 per x^2

Proviamo a risolvere l'equazione ricorsivamente "all'indietro" con principio di ottimalità

Consideriamo l'istante terminale N...

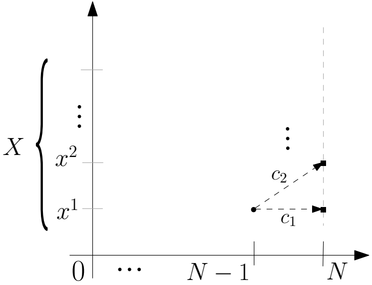


$\Rightarrow$ Il costo migliore a partire da x^1 al tempo N è semplicemente pari al costo terminale g_N valutato in x^1 (non abbiamo più decisioni da prendere...)

$\Rightarrow$ Assegniamo il costo ottimo al tempo N a ciascuno stato in X, calcolando $V_N(\cdot)$

Soluzione ricorsiva dell'equazione di Bellman (3/4)

Proviamo a risolvere l'equazione ricorsivamente "all'indietro" con principio di ottimalità
facciamo un passo indietro ad $N-1$...



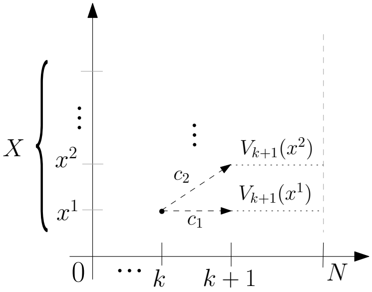
⇒ Il costo della scelta 1 è $c_1 + V_N(x^1)$ (riutilizzando il calcolo di $V_N(x^1)$)

⇒ Conoscendo tutti i costi $V_N(x^i)$, possiamo determinare la migliore scelta da $x_{N-1} = x^1$ al tempo $N-1$:

$$V_{N-1}(x^1) = \min_i \{c_i + V_N(x^i)\}$$

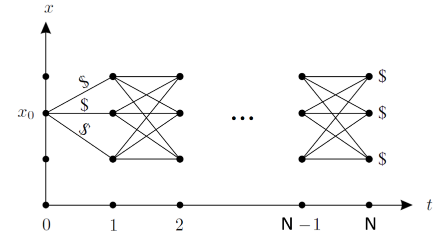
⇒ Assegniamo il costo ottimo al tempo $N-1$ a ciascuno stato in X, calcolando $V_{N-1}(\cdot)$

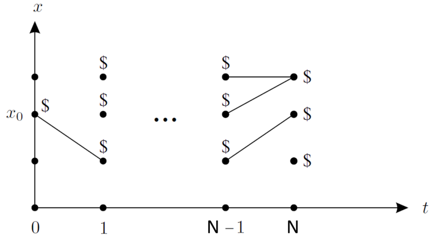
Proviamo a risolvere l'equazione ricorsivamente "all'indietro" con principio di ottimalità **al generico istante k...**



⇒ Per il principio di ottimalità, la "coda" (da $k+1$ e stato x^1) della soluzione ottima da k deve necessariamente coincidere con $V_{k+1}(x^1)$ (non possiamo fare meglio semplicemente perchè partiamo da k...) ⇒ $V_k(x^1) = \min_i\{c_i + V_{k+1}(x^i)\}$

⇒ **Scelte greedy (basate solo su considerazioni istantanee) rispetto ad una funzione che racchiude anche le conseguenze future!**





$O(|U|^N N)$: esplorazione di $|U|^N$ possibili scelte con N somme per calcolarne il costo

$O(|X||U|N)$: per ogni istante, per ogni stato e ogni azione, dobbiamo aggiungere il costo della scelta alla funzione valore già calcolata

Commenti

- se N diventa molto grande, DP ha un costo computazionale molto minore
- DP diventa oneroso al crescere del numero degli stati
- per il Principio di Ottimalità, all'indietro possiamo *scartare* percorsi
- DP trova la soluzione ottima *per ogni* x_0*!*

$$\begin{cases} \min\limits_{u_1,\ldots,u_{N-1}} & \sum_{k=0}^{N-1} pu_k + A\eta(u_k) + hx_k + g_N(x_N) \\ s.t. & x_{k+1} = x_k + u_k - d_k, \quad k = 0, 1, \ldots, N-1 \\ & u_k \in U_k \\ & x_k \geqslant 0 \end{cases}$$

- **capacità di immagazzinamento finita** M_m

$$x_k + u_k - d_k \leqslant M_m \qquad \Rightarrow \qquad u_k \leqslant M_m + d_k - x_k$$

- **capacità di produzione finita** M_p, $u_k \leqslant M_p$
- **Vincolo di domanda**

$$x_k + u_k - d_k \geqslant 0 \qquad \Rightarrow \qquad u_k \geqslant d_k - x_k$$

$\Rightarrow U_k = [d_k - x_k, \min\{M_p, M_m + d_k - x_k\}]$, $X = \{0, \ldots, M_m\}$

Equazione di Bellman

$$V_k(x_k) = \min_{u_k \in U_k} \{pu_k + A\eta(u_k) + hx_k + V_{k+1}(x_{k+1})\}, \qquad k = 0, 1, \ldots, N-1, \quad x_k = 0, \ldots, M_m$$

$$V_N(x_N) = g_N(x_N), \quad x_N = 0, \ldots, M_m$$

Nelle prossime lezioni...

Siamo interessati ad estendere la *tabella* che definisce $V_k(x)$ nell'esempio ad un numero infinito di righe e colonne

Prossimi passi:

- cominciamo con le colonne, $x_k \in \mathbb{R}^n$ (infinite scelte)
- studiamo una situazione in cui la minimizzazione rispetto ad u_k possa essere eseguita analiticamente (e non per enumerazione...)