Dynamic Programming (DP)

Katta G. Murty Lecture slides

DP deals with sequence of decisions, one after the other. DP deals with Sequetial Decision Processes.

Considers system that can be in one of a Finite number of states each point of time. State Space = set of all possible states of system.

In each state, One of a finite no. of actions or decisions, has to be taken. Set of actions for each state is different.

For each action, either an Immediate cost has to be paid (in minimization problems), or an Immediate Profit or Reward obtained (in maximization problems), and system moves to another state. Whole process stops when system reaches a Specified Final State.

Markovian Property: Immediate cost incurred, & next state system moves to, always depend only on present state & decision taken there, and not on past states through which system arrived at current state, i.e., **given present state, future is independent of the past.**

Additivity of Objective Func.: Objective func. is either Total cost or the Total reward.

Representation using a Network

A Network is set of Nodes & set of Directed arcs, each arc directed from one node (called Tail node of arc) to another (called Head node of arc). Arc directed from node i to node j is denoted by (i, j).

For sequential decision process, we represent each state by a node.

At a node i suppose there are 3 different decisions possible, and taking these decisions leads to states j_1, j_2, j_3. Then we include 3 arcs $(i, j_1), (i, j_2), (i, j_3)$ incident out of node i.

Thus each arc represents a possible decision at tail node of arc.

Length of each arc = immediate cost (or reward) of corresponding decision.

Sequence of states visited by system called a Realization. It depends on initial state & decisions made at various states along sequence.

Each realization corresponds to a path in network from initial node to terminal node. Total cost of realization is sum of lengths of arcs on path.

Policies, Optimal Policy

A policy is a rule that specifies the decision to take in each possible state of system.

An optimum policy is one that minimizes total cost (or maximizes total reward).

Multi-Stage Problems

A multi-stage problem is a sequential decision problem with state space partitioned into subsets called Stages. Each stage has ≥ 1 states. Stages themselves arranged in some order, and numbered $1, \ldots, n$. System always moves from one stage to next until terminal stage reached. So every decision taken in stage r

leads to stage $r + 1$.

Multistage models arise in situations where decisions are taken on a periodic basis, say every time period.

Some sequetial decision processes are staged; others have no stages at all.

Example: Driving home from work in Shortest Time

States are various street intersections between home and work. Network is street network with street intersections as nodes, and various street segments as arcs.

An example without stages: Solving Nonnegative Integer Knapsack Problem by DP

Type j	Weight w_j	Value v_j
1	3	600
2	4	700
3	5	500
4	2	180
5	20	2000
6	30	6000

Knapsack weight capacity $w = 12$

An example of staged DP: Solving 0–1 knapsack problem by DP

Only one copy of each object is available for loading into knapsack.

Here, remaining knapsack's weight capacity does not completely characterize state of system, we also need to know which object types are already loaded, and which are still available for loading into it.

So, a staged representation is convenient here. Process has n stages. In rth stage we consider loading object type r only. In each stage we have states representing the remining weight capacity of knapsack.

The Optimum Value Function (OVF)

Let S = State space.For each $s \in S$ define

$f(s) =$ Minimum total cost incurred (or maximum total reward obtained) by pursuing an optimal policy beginning with s as initial state.

$f(s)$ is called the OVF

Boundary Conditions

Terminal states are specified, process terminates whenever system reaches one of terminal states. So future cost (or reward) in each terminal state is 0, i.e.,

$f(s) = 0 \quad$ if s is a terminal state.

These are called Boundary conditions that the OVF must satisfy.

The Principle of Optimality

If i_r is a node on the shortest path from a node i_0 to a node i_k, then the portion of this path between nodes i_r to i_k must be a shortest path from i_r to i_k.

This simple observation is basis for principle of Optimality used to develop Recursive Method for DPs. Here is one version of this principle:

VERSION 1: If s is a state encountered in an optimum realization beginning in state s_0 and following an optimal policy, then the portion of this realization from s to the end is an optimum realization if process begins in state s.

The Functional Equations Satisfied by OVF

Consider cost minimization problem. Let

s_0	$=$	Current state
k	$=$	no. possible decisions in state s_0 with immediate costs $c_1, \ldots, c_k$ and state transitions $s_1, \ldots, s_k$
$f(s_t)$	$=$	Min. cost incurred by pursuing optimum policy beginning with s_t as initial state, for $t = 1$ to k
$c_t + f(s_t)$	$=$	Total cost from now till end, if we take decision t in current state s_0 but follow an optimum policy from next state onwards.

Hence optimum decision in current state s_0 is the one which minimizes $c_t + f(s_t)$, i.e.,

$$f(s_0) = \min\{c_t + f(s_t) : \quad t = 1 \text{ to } k\}$$

And optimum decision in state s_0 is t that minimizes RHS.

Above eq. known as Functional eq. or Optimality Eq. satisfied by OVF $f(s)$. If we know the values of $f(s_1), \ldots, f(s_k)$, we can use it to find $f(s_0)$.

We know $f(s) = 0$ for all terminal states. Using this and functional eqs. we can compute OVF at all states by moving backwards from terminal states one state at a time.

This method of evaluating OVF called Recursive Technique or Backwards Recursion.

Final output from this consists of OVF and optimum decision to take in each possible state of system.

To find Shortest paths from every node to a destination node in acyclic staged network

Finding Shortest paths from every node to a destination node in an acyclic network that is not staged, by DP

A directed network said to be Acyclic if it has no circuits (i.e., directed cycles). The nodes in such a network can be numbered in such a way that on every arc the

number of tail $<$ the number of head.

Such numbering called Acyclic numbering of nodes.

How to find acyclic numbering of nodes?

1 Look for un-numbered nodes which are not "head" of any remaining arc.

if none, network not acyclic, terminate.

Otherwise number all such nodes serially in any order beginning with next unused integer. Then go to 2.

2 If all nodes numbered, you have acyclic numbering, terminate.

Otherwise, consider all newly numbered nodes and arcs incident at them as deleted, and go back to 1 with remaining network.

Examples:

DP Algorithm to find Shortest Paths to a destination node in an acyclic network

Find acyclic numbering of nodes. Suppose there are n nodes, and node n is destination node. Define OVF

$f(i)$ = Length of shortest path from i to node n.

Boundary condition is $f(n) = 0$. The functional eq. is:

$f(i) = \min\{c_{ij} + f(j) : j \text{ such that } (i, j) \text{ is an arc}\}$

The j that attains the minimum above is the next node to go to when at node i.

Beginning with $f(n) = 0$ use backwards recursion to find $f(i)$ in order $n-1, n-2, \ldots, 1$.

Finding Shortest Paths from an Orign node to all other nodes in an acyclic network by DP

1 Find acyclic numbering of nodes. Let node 1 be origin.

2 Set up OVF and boundary conditions:

$f(i) =$ Length of shortest path from node 1 to node i

Boundary condition $\quad f(1) = 0$.

3 Write functional equations.

$f(i) = \min\{c_{ji} + f(j) : j$ such that (j, i) is an arc$\}$.

The j that attains the minimum here is the node from which you come to node i.

4 Beginning with $f(1) = 0$, use this recursion to find $f(i)$ in the order $i = 1, 2, \ldots, n-1, n$.

Solving Nonnegative Integer Knapsack Problems by DP

Object i	Weight w_i	Value v_i
1	3	12
2	4	12
3	3	9
4	3	15
5	7	42
6	9	18

Knapsack weight capacity $w_0 = 12$

The state of the system is characterized by the remaining knapsack weight capacity, which can take integer values between 0 to 12 in this problem.

The OVF in state w is:

$f(w) =$ Max possible value that can be loaded into knapsack if weight capacity is w.

The functional equations are:

$$f(w) = \max\{v_i + f(w - w_i) : i = 1 \text{ to } n \text{ s. th. } w_i \leq w\}$$

and the boundary conditions are: $f(0) = f(1) = f(2) = 0.$

Solving 0–1 Knapsack problem by DP

Object i	Weight w_i	Value v_i
1	3	12
2	4	12
3	3	15
4	7	42
5	9	18

Knapsack weight capacity $w_0 = 12$

Only one copy of each object available

For $k = 1$ to 5, decision whether to include object k or not, will be made only in stage k. So, states in the system are: $(k, w) : k = 1$ to 5, $w = 0$ to 12. The OVF is:

$f(k, w) =$ Max possible value that can be loaded into knapsack if weight capacity is w, and only objects $\{k, k+1, \dots, n\}$ are available (i.e., beginning in stage k in state (k, w) and going upto stage n).

Only two possible actions available in stage k, they are:

1 Not to include object k. If this action taken in state (k, w) transition will occur to state $(k+1, w)$ with immediate reward of 0.

2 To include object k. This action available in state (k, w) only if $w \geq w_k$, and in this case transition will take place to state $(k+1, w - w_k)$ with immediate reward of v_k.

The recursive eqs. are:

$$f(k, w) = \begin{cases} f(k+1, w) & \text{if } w < w_k \\ \max\{f(k+1, w), v_k + f(k+1, w - w_k)\} & \text{if } w \geq w_k \end{cases}$$

And the boundary conditions are:

$$f(n, w) = \begin{cases} 0 & \text{if } w < w_n \\ v_n & \text{if } w \geq w_n \end{cases}$$

Resource Allocation Problems Solved by DP

- Consider single resource with $\leq k$ units availab.
- Resource can be allocated to n different activities, but only in integer quantities.
- For $i = 1$ to n, $y = 0$ to k, $r_i(y)$ = reward (profit) obtained by allocating y units of resource to activity i , is data given.
- The objective is to find optimum allocation of available resource among activities to maximize total reward.

Examples of Applications

1 How many inspectors to allocate to each river (or region) to monitor pollution?

2 How many patrol cars to allocate to different highway segments to catch speeding drivers?

3 How many campaign volunteers should a politician allocate to each district to get maximum total votes?

4 How many machines to allocate to different products?

5 How many \$ million to allocate to different regions to maximize benefit?

etc.

For $i = 1$ to n let x_i = number of units of resource allocated to i th activity.

Aim: find optimum $(x_1, \ldots, x_n)$ that maximizes total reward.

Can solve using staged DP. n stages. In stage i we decide only x_i, the number of units (integer) of resource allocated to ith activity only.

So state in this system is of form (i, ℓ) where i = stage, and ℓ = total resource units available. $i = 1$ to n, $\ell = 0$ to k.

The OVF is:

$f(i, \ell) =$ Maximum total reward that can be obtained from activities i to n only with ℓ units of resource available to allocate among them.

The boundary conditions are:

$$f(n, \ell) = \max\{r_i(t) : 0 \leq t \leq \ell\}$$

If $\bar{t}$ attains the maximum in above, opt. decision in state (n, ℓ) is to allot $\bar{t}$ units of resource to activity n.

The recurrence eqs. are:

$$f((i, \ell)) = \max\{r_i(x_i) + f((i+1, \ell - x_i)) : 0 \leq x_i \leq \ell\}$$

If $\bar{x}_i$ attains max. in above, optimum decision in state (i, ℓ) is to allot $\bar{x}_i$ units of resource to activity i and continue as in state $(i+1, \ell - \bar{x}_i)$.

Example:

EPA wants to max. no. tests of dioxin contamination of Michigan rivers. State divided into 3 regions. Table gives data on the no. tests that can be conducted in each region by allotting some inspectors.

EPA willing to appoint total 5 inspectors. Determine how many to allot to each region to max. total number tests conducted over whole state per month.

Region	No. of tests/month if r inspectors allotted $r = 1$	2	3	4	5
1	25	50	80	117	125
2	20	70	130	150	160
3	10	20	35	40	45

Homeworks:

1. US govt. plans to spend money in an economically depressed region to increase employment, over a period of 3 years. Funds can only be spent in integer no. of units of $1 mil. in any year. Here is estimated returns.

	New jobs created if \$$r$ mil. are spent in year								
Year	$r = 0$	1	2	3	4	5	6	7	8
1	0	5	15	40	80	90	95	98	100
2	0	5	15	40	60	70	73	74	75
3	0	4	26	40	45	50	51	52	53

Find an optimum policy for spending the funds over the planning horizon, which maximizes the total number of additional jobs created, using DP.

2. There are 4 types of investments. Each accepts investment only in integer multiples of certificates. We have 30 units of money to invest (1 unit = $1000). Following table provides data on rewards obtained from investments in the different types.

Investment	Cost	Reward for buying r certificates				
type	(units/certificate)	$r = 1$	2	3	4	5
1	3	2	3	8	16	23
2	2	1	2	4	7	12
3	4	4	8	15	24	30
4	6	4	9	23	36	42

At least one certificate of each type must be purchased. Use DP to determine the optimum number of certificates of each type to buy to maximize total reward.

3. There are 4 objects available for loading into a knapsack of unlimited weight capacity. Data on the objects is given below.

Object i	1	2	3	4
Value v_i	7	16	19	15
Weight w_i	3	6	7	5

An unlimited number of copies of each object are available for loading into the knapsack. Define

$g(t)$ = the minimum total weight of items needed in order to achieve a total value of at least t in the knapsack.

Find $g(t)$ and the associated (complete) optimal policy for $t = 25$.