LECTURE SLIDES - DYNAMIC PROGRAMMING

BASED ON LECTURES GIVEN AT THE

MASSACHUSETTS INST. OF TECHNOLOGY

CAMBRIDGE, MASS

FALL 2015

DIMITRI P. BERTSEKAS

These lecture slides are based on the two-volume book: "Dynamic Programming and Optimal Control" Athena Scientific, by D. P. Bertsekas (Vol. I, 3rd Edition, 2005; Vol. II, 4th Edition, 2012); see

http://www.athenasc.com/dpbook.html

Two related reference books:

(1) "Abstract Dynamic Programming," by D. P. Bertsekas, Athena Scientific, 2013

(2) "Neuro-Dynamic Programming," Athena Scientific, by D. P. Bertsekas and J. N. Tsitsiklis, 1996

6.231: DYNAMIC PROGRAMMING

LECTURE 1

LECTURE OUTLINE

- Problem Formulation
- Examples
- The Basic Problem
- Significance of Feedback

DP AS AN OPTIMIZATION METHODOLOGY

- Generic optimization problem:

$$\min_{u \in U} g(u)$$

where u is the optimization/decision variable, $g(u)$ is the cost function, and U is the constraint set

- Categories of problems:
  - Discrete (U is finite) or continuous
  - Linear (g is linear and U is polyhedral) or nonlinear
  - Stochastic or deterministic: In stochastic problems the cost involves a stochastic parameter w, which is averaged, i.e., it has the form

$$g(u) = E_w\{G(u, w)\}$$

where w is a random parameter.

- DP can deal with complex stochastic problems where information about w becomes available in stages, and the decisions are also made in stages and make use of this information.

BASIC STRUCTURE OF STOCHASTIC DP

- Discrete-time system

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

  - k: Discrete time
  - x_k: State; summarizes past information that is relevant for future optimization
  - u_k: Control; decision to be selected at time k from a given set
  - w_k: Random parameter (also called disturbance or noise depending on the context)
  - N: Horizon or number of times control is applied

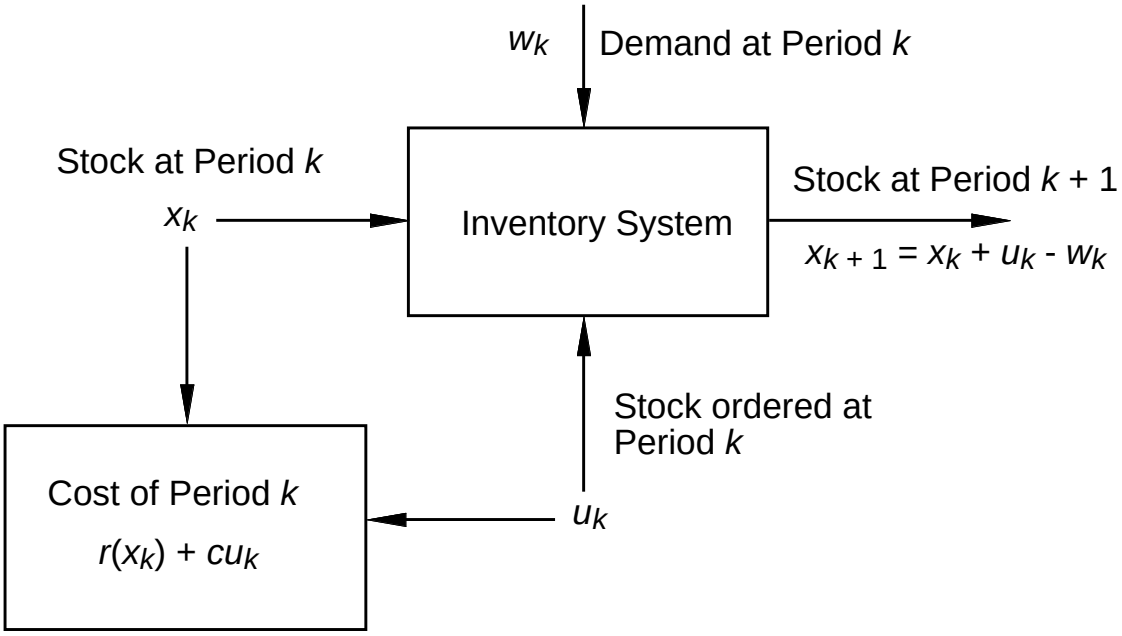
- Cost function that is additive over time

$$E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\}$$

- Alternative system description: $P(x_{k+1} \mid x_k, u_k)$

$$x_{k+1} = w_k \quad \text{with } P(w_k \mid x_k, u_k) = P(x_{k+1} \mid x_k, u_k)$$

INVENTORY CONTROL EXAMPLE



- Discrete-time system

$$x_{k+1} = f_k(x_k, u_k, w_k) = x_k + u_k - w_k$$

- Cost function that is additive over time

$$E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\}$$

$$= E\left\{ \sum_{k=0}^{N-1} \big(cu_k + r(x_k + u_k - w_k) \big) \right\}$$

- Optimization over policies: Rules/functions $u_k = \mu_k(x_k)$ that map states to controls

ADDITIONAL ASSUMPTIONS

- The set of values that the control u_k can take depend at most on x_k and not on prior x or u
- Probability distribution of w_k does not depend on past values $w_{k-1}, \ldots, w_0$, but may depend on x_k and u_k
  - Otherwise past values of w or x would be useful for future optimization
- Sequence of events envisioned in period k:
  - x_k occurs according to

$$x_k = f_{k-1}\big(x_{k-1}, u_{k-1}, w_{k-1}\big)$$

  - u_k is selected with knowledge of x_k, i.e.,

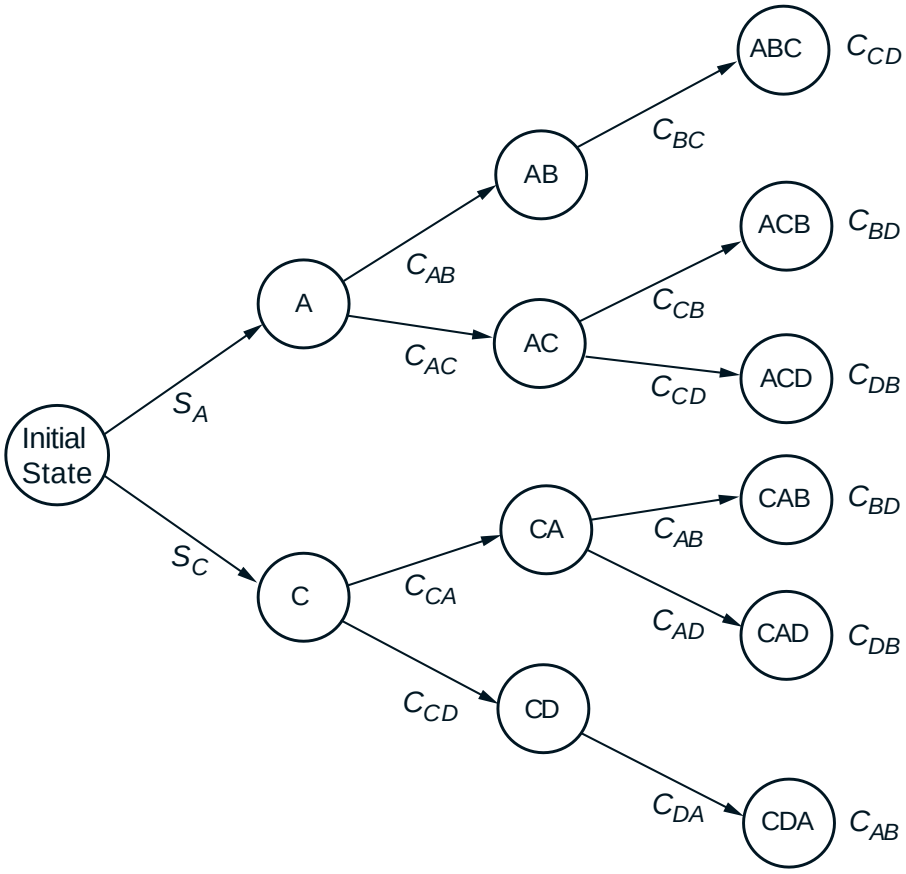
$$u_k \in U_k(x_k)$$

  - w_k is random and generated according to a distribution

$$P_{w_k}(x_k, u_k)$$

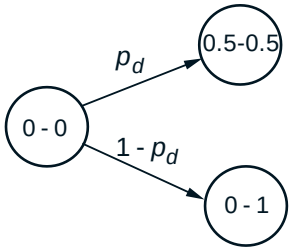
DETERMINISTIC FINITE-STATE PROBLEMS

- Scheduling example: Find optimal sequence of operations A, B, C, D
- A must precede B, and C must precede D
- Given startup cost S_A and S_C, and setup transition cost C_{mn} from operation m to operation n

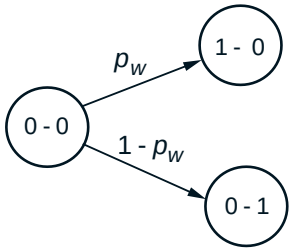


STOCHASTIC FINITE-STATE PROBLEMS

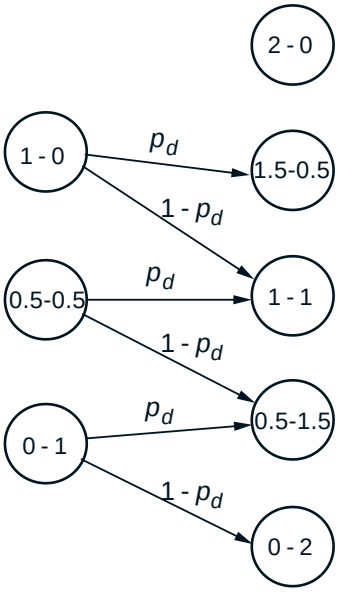
- Example: Find two-game chess match strategy
- *Timid* play draws with prob. $p_d > 0$ and loses with prob. $1 - p_d$. *Bold* play wins with prob. $p_w < 1/2$ and loses with prob. $1 - p_w$



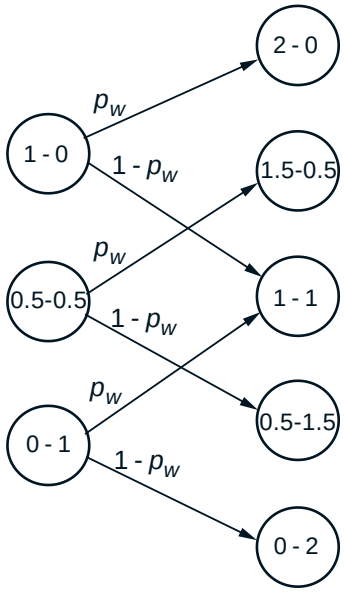
1st Game / Timid Play



1st Game / Bold Play



2nd Game / Timid Play



2nd Game / Bold Play

BASIC PROBLEM

- System $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \ldots, N-1$
- Control contraints $u_k \in U_k(x_k)$
- Probability distribution $P_k(\cdot \mid x_k, u_k)$ of w_k
- Policies $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all x_k
- Expected cost of π starting at x_0 is

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

- Optimal cost function

$$J^*(x_0) = \min_\pi J_\pi(x_0)$$

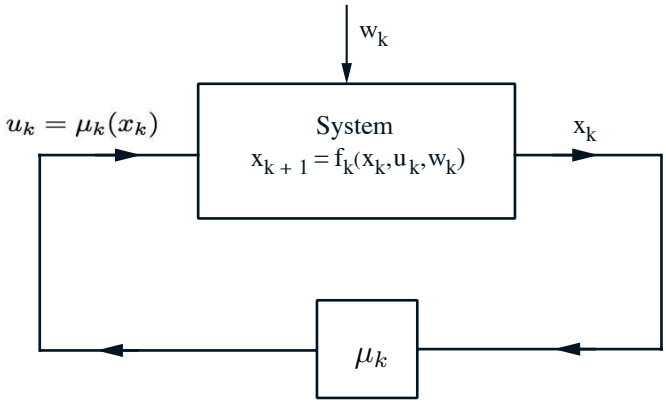
- Optimal policy π^* satisfies

$$J_{\pi^*}(x_0) = J^*(x_0)$$

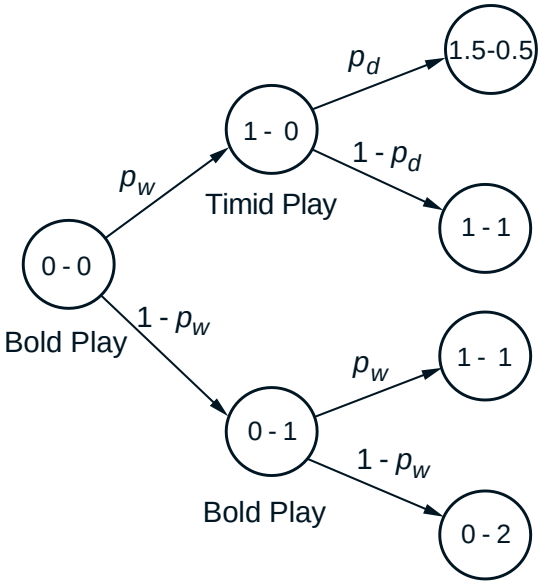
When produced by DP, π^* is independent of x_0.

SIGNIFICANCE OF FEEDBACK

- Open-loop versus closed-loop policies



- In deterministic problems open loop is as good as closed loop
- Value of information; chess match example
- Example of open-loop policy: Play always bold
- Consider the closed-loop policy: Play timid if and only if you are ahead



VARIANTS OF DP PROBLEMS

- Continuous-time problems
- Imperfect state information problems
- Infinite horizon problems
- Suboptimal control

LECTURE BREAKDOWN

- Finite Horizon Problems (Vol. 1, Ch. 1-6)
  - Ch. 1: The DP algorithm (2 lectures)
  - Ch. 2: Deterministic finite-state problems (1 lecture)
  - Ch. 4: Stochastic DP problems (2 lectures)
  - Ch. 5: Imperfect state information problems (2 lectures)
  - Ch. 6: Suboptimal control (2 lectures)
- Infinite Horizon Problems - Simple (Vol. 1, Ch. 7, 3 lectures)

****

- Infinite Horizon Problems - Advanced (Vol. 2)
  - Chs. 1, 2: Discounted problems - Computational methods (3 lectures)
  - Ch. 3: Stochastic shortest path problems (2 lectures)
  - Chs. 6, 7: Approximate DP (6 lectures)

COURSE ADMINISTRATION

- Homework ... once a week or two weeks (30% of grade)
- In class midterm, near end of October ... will cover finite horizon and simple infinite horizon material (30% of grade)
- Project (40% of grade)
- Collaboration in homework allowed but individual solutions are expected
- Prerequisites: Introductory probability, good gasp of advanced calculus (including convergence concepts)
- Textbook: Vol. I of text is required. Vol. II is strongly recommended, but you may be able to get by without it using OCW material (including videos)

A NOTE ON THESE SLIDES

- These slides are a teaching aid, not a text
- Don't expect a rigorous mathematical development or precise mathematical statements
- Figures are meant to convey and enhance ideas, not to express them precisely
- Omitted proofs and a much fuller discussion can be found in the textbook, which these slides follow

6.231 DYNAMIC PROGRAMMING

LECTURE 2

LECTURE OUTLINE

- The basic problem
- Principle of optimality
- DP example: Deterministic problem
- DP example: Stochastic problem
- The general DP algorithm
- State augmentation

BASIC PROBLEM

- System $x_{k+1} = f_k(x_k, u_k, w_k)$, $k = 0, \ldots, N-1$
- Control constraints $u_k \in U_k(x_k)$
- Probability distribution $P_k(\cdot \mid x_k, u_k)$ of w_k
- Policies $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$ and is such that $\mu_k(x_k) \in U_k(x_k)$ for all x_k
- Expected cost of π starting at x_0 is

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

- Optimal cost function

$$J^*(x_0) = \min_\pi J_\pi(x_0)$$

- Optimal policy π^* is one that satisfies

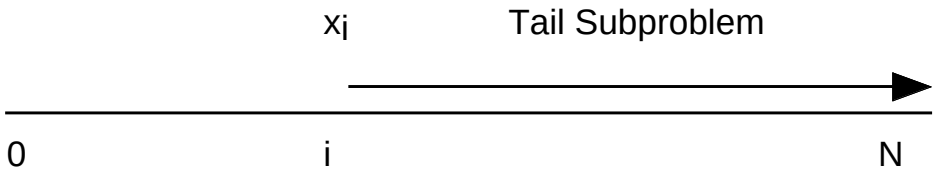
$$J_{\pi^*}(x_0) = J^*(x_0)$$

PRINCIPLE OF OPTIMALITY

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be optimal policy
- Consider the "tail subproblem" whereby we are at x_i at time i and wish to minimize the "cost-to-go" from time i to time N

$$E\left\{ g_N(x_N) + \sum_{k=i}^{N-1} g_k\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

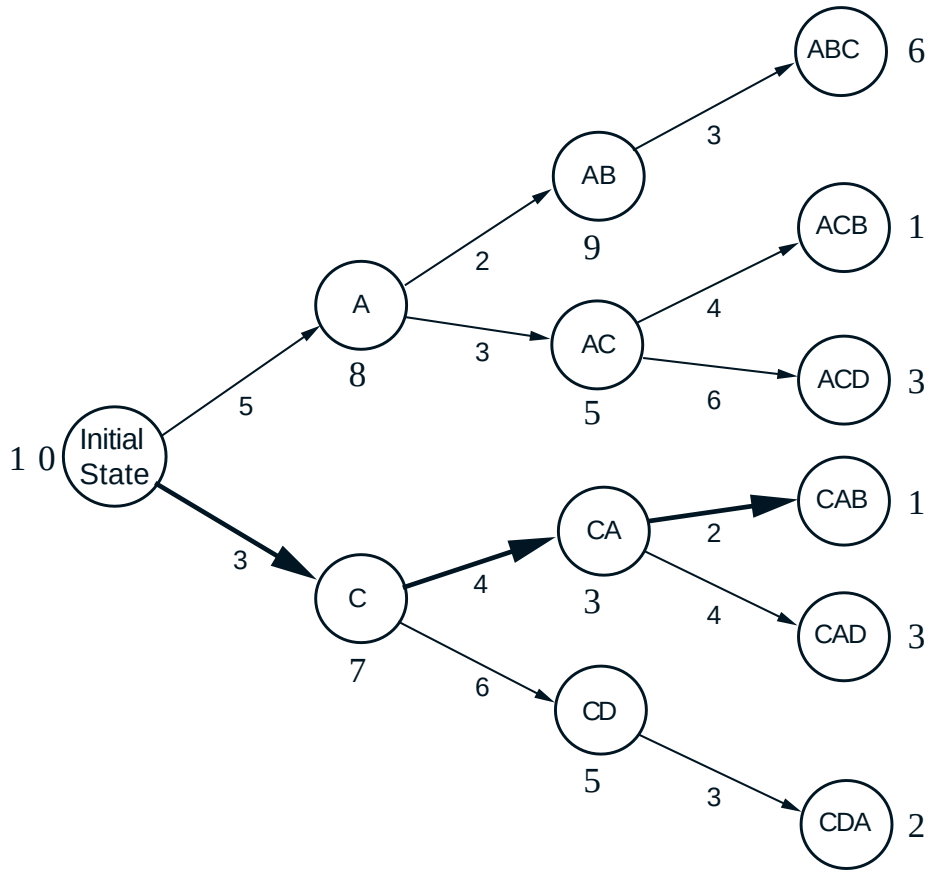
and the "tail policy" $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$



- *Principle of optimality*: The tail policy is optimal for the tail subproblem (optimization of the future does not depend on what we did in the past)
- DP first solves ALL tail subroblems of final stage
- At the generic step, it solves ALL tail subproblems of a given time length, using the solution of the tail subproblems of shorter time length

DETERMINISTIC SCHEDULING EXAMPLE

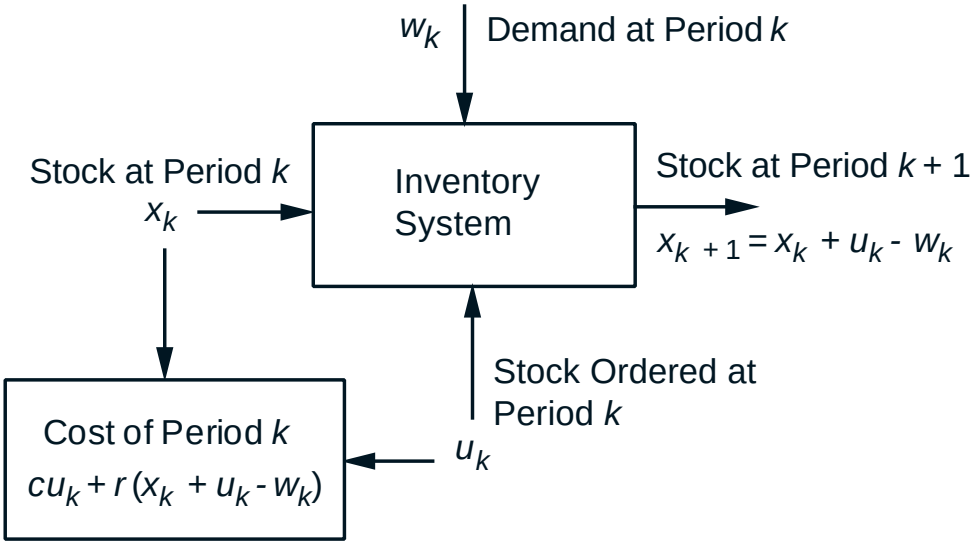
- Find optimal sequence of operations A, B, C, D (A must precede B and C must precede D)



- Start from the last tail subproblem and go backwards

- At each state-time pair, we record the optimal cost-to-go and the optimal decision

STOCHASTIC INVENTORY EXAMPLE



- Tail Subproblems of Length 1:

$$J_{N-1}(x_{N-1}) = \min_{u_{N-1} \geq 0} E_{w_{N-1}} \Big\{ cu_{N-1} + r(x_{N-1} + u_{N-1} - w_{N-1}) \Big\}$$

- Tail Subproblems of Length $N - k$:

$$J_k(x_k) = \min_{u_k \geq 0} E_{w_k} \Big\{ cu_k + r(x_k + u_k - w_k) + J_{k+1}(x_k + u_k - w_k) \Big\}$$

- $J_0(x_0)$ is opt. cost of initial state x_0

DP ALGORITHM

- Start with

$$J_N(x_N) = g_N(x_N),$$

and go backwards using

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k} \Big\{ g_k(x_k, u_k, w_k) + J_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\}, \quad k = 0, 1, \ldots, N-1.$$

- Then $J_0(x_0)$, generated at the last step, is equal to the optimal cost $J^*(x_0)$. Also, the policy

$$\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$$

where $\mu_k^*(x_k)$ minimizes in the right side above for each x_k and k, is optimal

- Justification: Proof by induction that $J_k(x_k)$ is equal to $J_k^*(x_k)$, defined as the optimal cost of the tail subproblem that starts at time k at state x_k

- Note:
  - ALL the tail subproblems are solved (in addition to the original problem)
  - Intensive computational requirements

PROOF OF THE INDUCTION STEP

- Let $\pi_k = \{\mu_k, \mu_{k+1}, \ldots, \mu_{N-1}\}$ denote a tail policy from time k onward
- Assume that $J_{k+1}(x_{k+1}) = J^*_{k+1}(x_{k+1})$. Then

$$J^*_k(x_k) = \min_{(\mu_k, \pi_{k+1})} \underset{w_k, \ldots, w_{N-1}}{E} \Bigg\{ g_k\big(x_k, \mu_k(x_k), w_k\big) + g_N(x_N) + \sum_{i=k+1}^{N-1} g_i\big(x_i, \mu_i(x_i), w_i\big) \Bigg\}$$

$$= \min_{\mu_k} \underset{w_k}{E} \Bigg\{ g_k\big(x_k, \mu_k(x_k), w_k\big) + \min_{\pi_{k+1}} \Bigg[\underset{w_{k+1}, \ldots, w_{N-1}}{E} \Bigg\{ g_N(x_N) + \sum_{i=k+1}^{N-1} g_i\big(x_i, \mu_i(x_i), w_i\big) \Bigg\} \Bigg] \Bigg\}$$

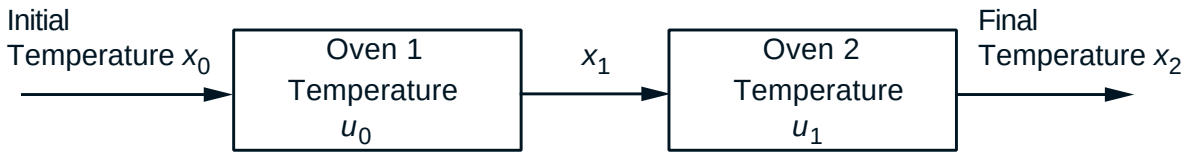
$$= \min_{\mu_k} \underset{w_k}{E} \Big\{ g_k\big(x_k, \mu_k(x_k), w_k\big) + J^*_{k+1}\Big(f_k\big(x_k, \mu_k(x_k), w_k\big)\Big) \Big\}$$

$$= \min_{\mu_k} \underset{w_k}{E} \Big\{ g_k\big(x_k, \mu_k(x_k), w_k\big) + J_{k+1}\Big(f_k\big(x_k, \mu_k(x_k), w_k\big)\Big) \Big\}$$

$$= \min_{u_k \in U_k(x_k)} \underset{w_k}{E} \Big\{ g_k(x_k, u_k, w_k) + J_{k+1}\Big(f_k(x_k, u_k, w_k)\Big) \Big\}$$

$$= J_k(x_k)$$

LINEAR-QUADRATIC ANALYTICAL EXAMPLE



- System

$$x_{k+1} = (1-a)x_k + au_k, \qquad k = 0, 1,$$

where a is given scalar from the interval $(0, 1)$

- Cost

$$r(x_2 - T)^2 + u_0^2 + u_1^2$$

where r is given positive scalar

- DP Algorithm:

$$J_2(x_2) = r(x_2 - T)^2$$

$$J_1(x_1) = \min_{u_1} \Big[u_1^2 + r\big((1-a)x_1 + au_1 - T\big)^2 \Big]$$

$$J_0(x_0) = \min_{u_0} \big[u_0^2 + J_1\big((1-a)x_0 + au_0\big) \big]$$

STATE AUGMENTATION

- When assumptions of the basic problem are violated (e.g., disturbances are correlated, cost is nonadditive, etc) reformulate/augment the state
- DP algorithm still applies, but the problem gets BIGGER
- Example: Time lags

$$x_{k+1} = f_k(x_k, x_{k-1}, u_k, w_k)$$

- Introduce additional state variable $y_k = x_{k-1}$. New system takes the form

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} f_k(x_k, y_k, u_k, w_k) \\ x_k \end{pmatrix}$$

View $\tilde{x}_k = (x_k, y_k)$ as the new state.

- DP algorithm for the reformulated problem:

$$J_k(x_k, x_{k-1}) = \min_{u_k \in U_k(x_k)} \underset{w_k}{E} \Big\{ g_k(x_k, u_k, w_k) + J_{k+1}\big(f_k(x_k, x_{k-1}, u_k, w_k), x_k\big) \Big\}$$

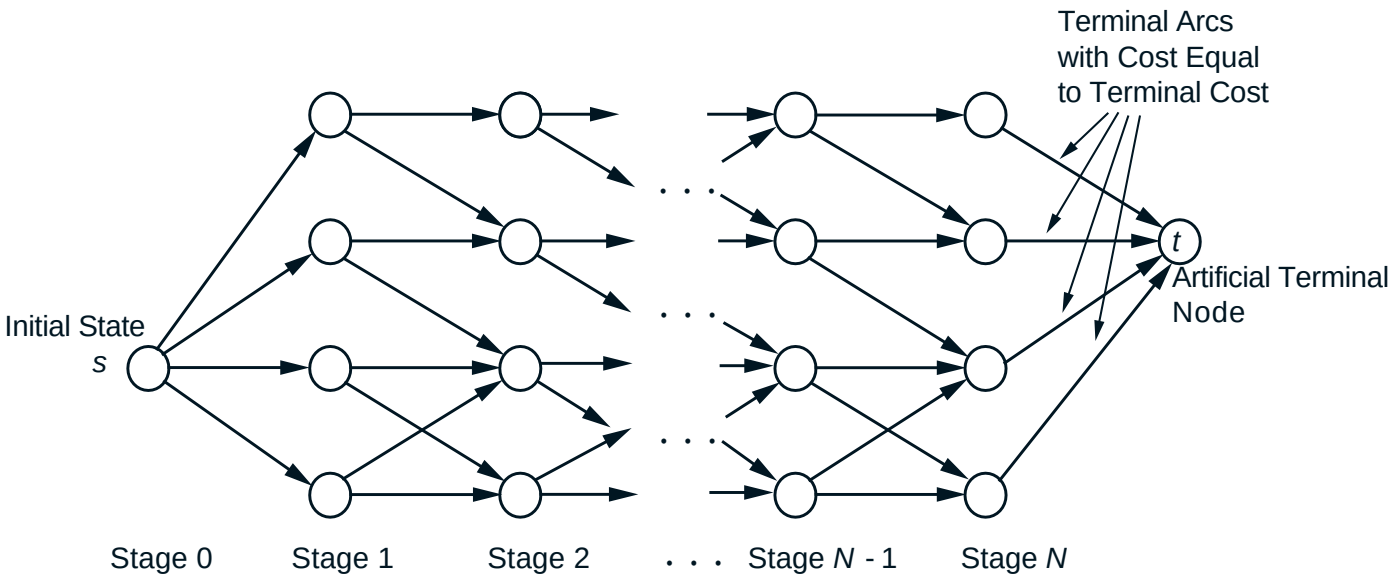
6.231 DYNAMIC PROGRAMMING

LECTURE 3

LECTURE OUTLINE

- Deterministic finite-state DP problems
- Backward shortest path algorithm
- Forward shortest path algorithm
- Shortest path examples
- Alternative shortest path algorithms

DETERMINISTIC FINITE-STATE PROBLEM



- States <==> Nodes
- Controls <==> Arcs
- Control sequences (open-loop) <==> paths from initial state to terminal states
- a_{ij}^k: Cost of transition from state $i \in S_k$ to state $j \in S_{k+1}$ at time k (view it as "length" of the arc)
- a_{it}^N: Terminal cost of state $i \in S_N$
- Cost of control sequence <==> Cost of the corresponding path (view it as "length" of the path)

BACKWARD AND FORWARD DP ALGORITHMS

- DP algorithm:

$$J_N(i) = a_{it}^N, \quad i \in S_N,$$

$$J_k(i) = \min_{j \in S_{k+1}} \left[a_{ij}^k + J_{k+1}(j) \right], \quad i \in S_k, \ k = 0, \ldots, N-1$$

The optimal cost is $J_0(s)$ and is equal to the length of the shortest path from s to t

- Observation: An optimal path $s \to t$ is also an optimal path $t \to s$ in a "reverse" shortest path problem where the direction of each arc is reversed and its length is left unchanged

- Forward DP algorithm (= backward DP algorithm for the reverse problem):

$$\tilde{J}_N(j) = a_{sj}^0, \quad j \in S_1,$$

$$\tilde{J}_k(j) = \min_{i \in S_{N-k}} \left[a_{ij}^{N-k} + \tilde{J}_{k+1}(i) \right], \quad j \in S_{N-k+1}$$

The optimal cost is $\tilde{J}_0(t) = \min_{i \in S_N} \left[a_{it}^N + \tilde{J}_1(i) \right]$

- View $\tilde{J}_k(j)$ as optimal cost-to-arrive to state j from initial state s

A NOTE ON FORWARD DP ALGORITHMS

- There is no forward DP algorithm for stochastic problems

- Mathematically, for stochastic problems, we cannot restrict ourselves to open-loop sequences, so the shortest path viewpoint fails

- Conceptually, in the presence of uncertainty, the concept of "optimal-cost-to-arrive" at a state x_k does not make sense. For example, it may be impossible to guarantee (with prob. 1) that any given state can be reached

- By contrast, even in stochastic problems, the concept of "optimal cost-to-go" from any state x_k makes clear sense

GENERIC SHORTEST PATH PROBLEMS

- $\{1, 2, \ldots, N, t\}$: nodes of a graph (t: the *destination*)
- a_{ij}: cost of moving from node i to node j
- Find a shortest (minimum cost) path from each node i to node t
- Assumption: All cycles have nonnegative length. Then an optimal path need not take more than N moves
- We formulate the problem as one where we require exactly N moves but allow degenerate moves from a node i to itself with cost $a_{ii} = 0$

$J_k(i)$ = opt. cost of getting from i to t in $N-k$ moves

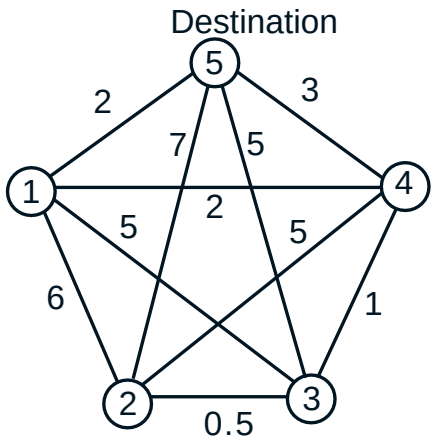
$J_0(i)$: Cost of the optimal path from i to t.

- DP algorithm:

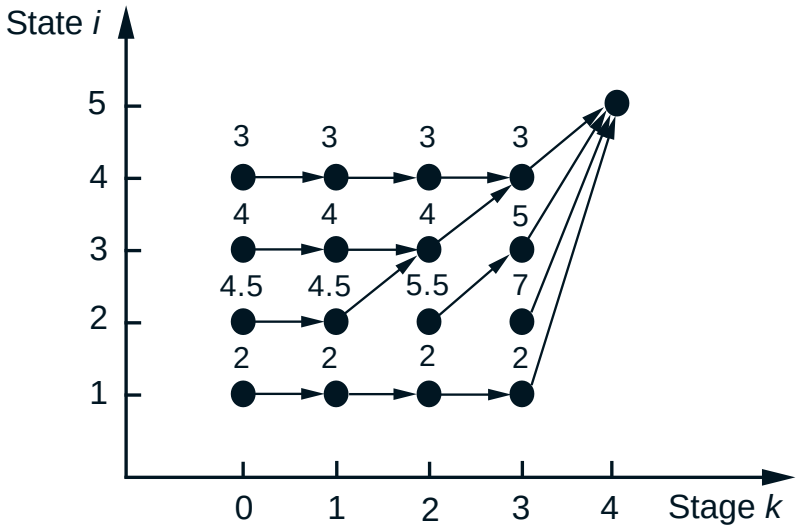
$$J_k(i) = \min_{j=1,\ldots,N} \big[a_{ij} + J_{k+1}(j)\big], \qquad k = 0, 1, \ldots, N-2,$$

with $J_{N-1}(i) = a_{it}$, $i = 1, 2, \ldots, N$

EXAMPLE



(a)



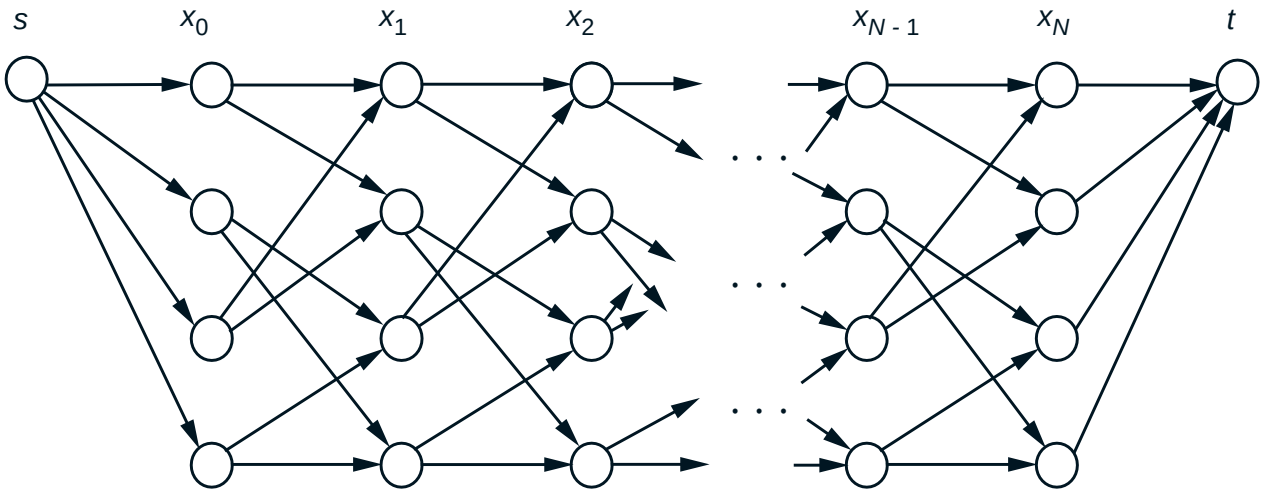
(b)

$$J_{N-1}(i) = a_{it}, \qquad i = 1, 2, \ldots, N,$$

$$J_k(i) = \min_{j=1,\ldots,N} \left[a_{ij} + J_{k+1}(j) \right], \qquad k = 0, 1, \ldots, N-2.$$

ESTIMATION / HIDDEN MARKOV MODELS

- Markov chain with transition probabilities p_{ij}
- State transitions are hidden from view
- For each transition, we get an (independent) observation
- $r(z; i, j)$: Prob. the observation takes value z when the state transition is from i to j
- Trajectory estimation problem: Given the observation sequence $Z_N = \{z_1, z_2, \ldots, z_N\}$, what is the "most likely" state transition sequence $\hat{X}_N = \{\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_N\}$ [one that maximizes $p(X_N \mid Z_N)$ over all $X_N = \{x_0, x_1, \ldots, x_N\}$].



VITERBI ALGORITHM

- We have

$$p(X_N \mid Z_N) = \frac{p(X_N, Z_N)}{p(Z_N)}$$

where $p(X_N, Z_N)$ and $p(Z_N)$ are the unconditional probabilities of occurrence of (X_N, Z_N) and Z_N

- Maximizing $p(X_N \mid Z_N)$ is equivalent with maximizing $\ln(p(X_N, Z_N))$

- We have (using the "multiplication rule" for cond. probs)

$$p(X_N, Z_N) = \pi_{x_0} \prod_{k=1}^{N} p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k)$$

so the problem is equivalent to

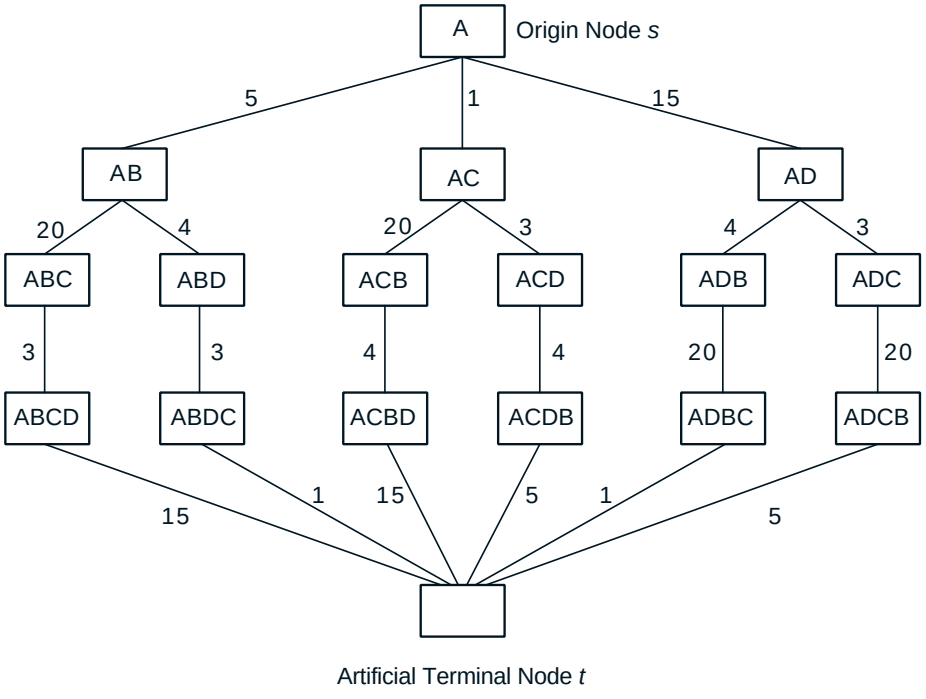
$$\text{minimize} - \ln(\pi_{x_0}) - \sum_{k=1}^{N} \ln\big(p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k)\big)$$

over all possible sequences $\{x_0, x_1, \ldots, x_N\}$.

- This is a shortest path problem.

GENERAL SHORTEST PATH ALGORITHMS

- There are many nonDP shortest path algorithms. They can all be used to solve deterministic finite-state problems
- They may be preferable than DP if they avoid calculating the optimal cost-to-go of EVERY state
- Essential for problems with HUGE state spaces.
- Combinatorial optimization is prime example (e.g., scheduling/traveling salesman)



	5	1	15
5		20	4
1	20		3
15	4	3	

LABEL CORRECTING METHODS

- Given: Origin s, destination t, lengths $a_{ij} \geq 0$.
- Idea is to progressively discover shorter paths from the origin s to every other node i
- Notation:
  - d_i (label of i): Length of the shortest path found (initially $d_s = 0$, $d_i = \infty$ for $i \neq s$)
  - UPPER: The label d_t of the destination
  - OPEN list: Contains nodes that are currently active in the sense that they are candidates for further examination (initially OPEN=$\{s\}$)

Label Correcting Algorithm

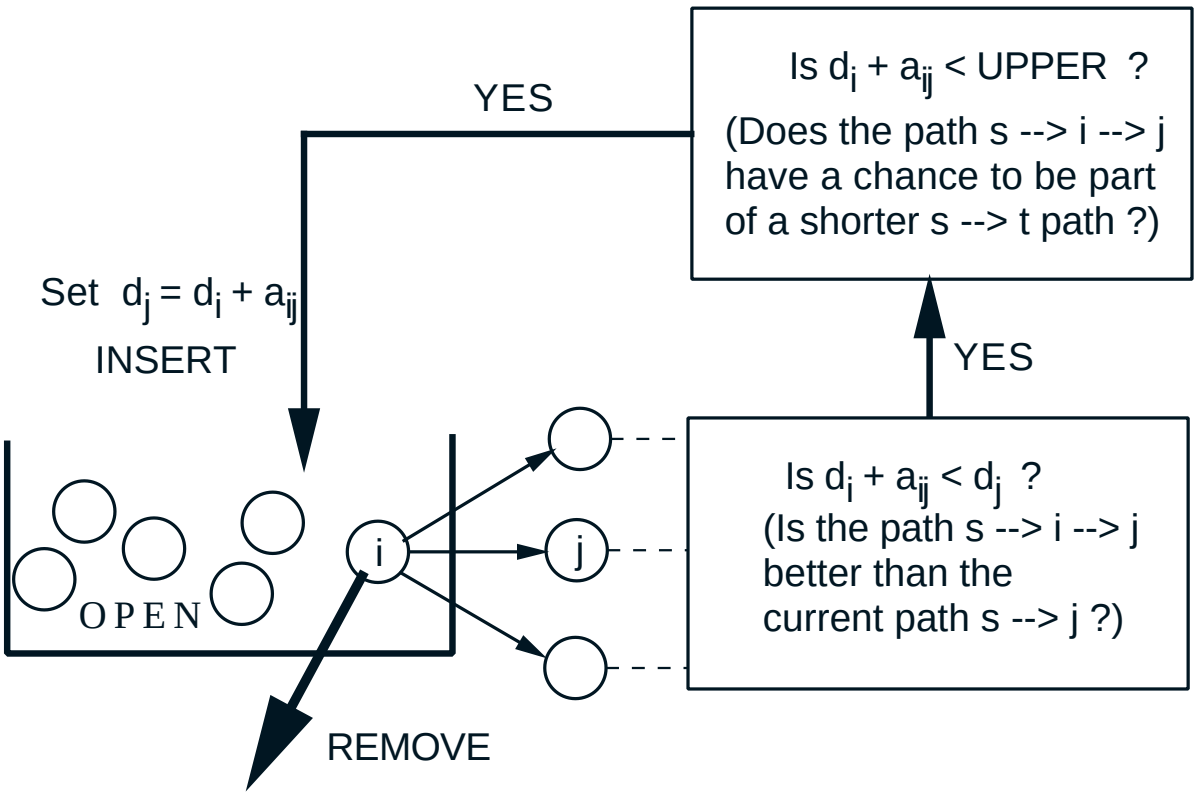
Step 1 (Node Removal): Remove a node i from OPEN and for each child j of i, do step 2

Step 2 (Node Insertion Test): If $d_i + a_{ij} < \min\{d_j, \text{UPPER}\}$, set $d_j = d_i + a_{ij}$ and set i to be the parent of j. In addition, if $j \neq t$, place j in OPEN if it is not already in OPEN, while if $j = t$, set UPPER to the new value $d_i + a_{it}$ of d_t

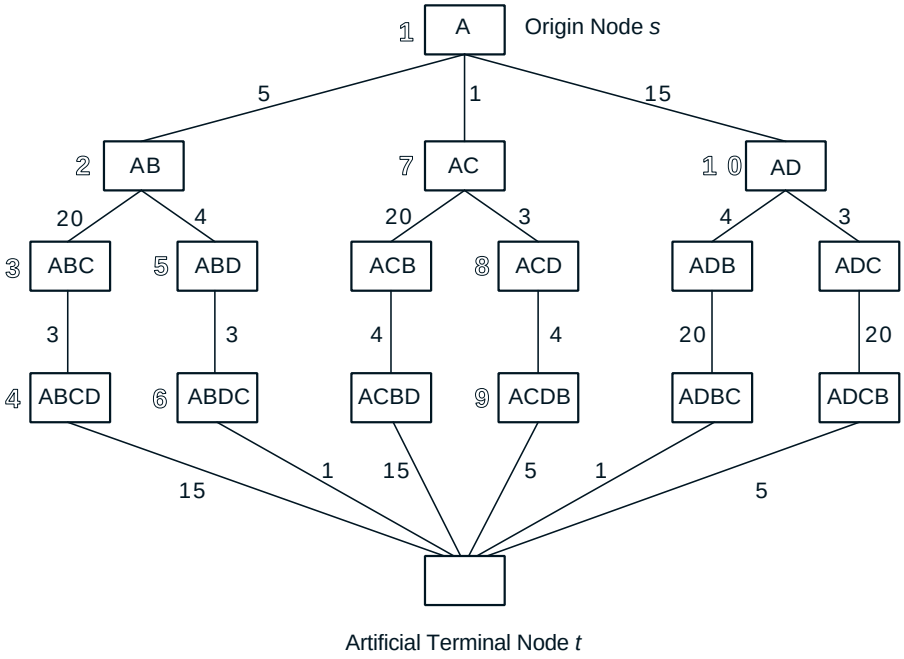
Step 3 (Termination Test): If OPEN is empty, terminate; else go to step 1

VISUALIZATION/EXPLANATION

- Given: Origin s, destination t, lengths $a_{ij} \geq 0$
- d_i (label of i): Length of the shortest path found thus far (initially $d_s = 0$, $d_i = \infty$ for $i \neq s$). The label d_i is implicitly associated with an $s \to i$ path
- UPPER: The label d_t of the destination
- OPEN list: Contains "active" nodes (initially OPEN=$\{s\}$)



EXAMPLE



Iter. No.	Node Exiting OPEN	OPEN after Iteration	UPPER
0	-	1	∞
1	1	2, 7,10	∞
2	2	3, 5, 7, 10	∞
3	3	4, 5, 7, 10	∞
4	4	5, 7, 10	43
5	5	6, 7, 10	43
6	6	7, 10	13
7	7	8, 10	13
8	8	9, 10	13
9	9	10	13
10	10	Empty	13

- Note that some nodes never entered OPEN

VALIDITY OF LABEL CORRECTING METHODS

Proposition: If there exists at least one path from the origin to the destination, the label correcting algorithm terminates with UPPER equal to the shortest distance from the origin to the destination

Proof: (1) Each time a node j enters OPEN, its label is decreased and becomes equal to the length of some path from s to j

(2) The number of possible distinct path lengths is finite, so the number of times a node can enter OPEN is finite, and the algorithm terminates

(3) Let $(s, j_1, j_2, \ldots, j_k, t)$ be a shortest path and let d^* be the shortest distance. If UPPER $> d^*$ at termination, UPPER will also be larger than the length of all the paths $(s, j_1, \ldots, j_m)$, $m = 1, \ldots, k$, throughout the algorithm. Hence, node j_k will never enter the OPEN list with d_{j_k} equal to the shortest distance from s to j_k. Similarly node j_{k-1} will never enter the OPEN list with $d_{j_{k-1}}$ equal to the shortest distance from s to j_{k-1}. Continue to j_1 to get a contradiction

6.231 DYNAMIC PROGRAMMING

LECTURE 4

LECTURE OUTLINE

- Examples of stochastic DP problems
- Linear-quadratic problems
- Inventory control

LINEAR-QUADRATIC PROBLEMS

- System: $x_{k+1} = A_k x_k + B_k u_k + w_k$
- Quadratic cost

$$\underset{\substack{w_k \\ k=0,1,\ldots,N-1}}{E} \left\{ x'_N Q_N x_N + \sum_{k=0}^{N-1} (x'_k Q_k x_k + u'_k R_k u_k) \right\}$$

where $Q_k \geq 0$ and $R_k > 0$ [in the positive (semi)definite sense].

- w_k are independent and zero mean
- DP algorithm:

$$J_N(x_N) = x'_N Q_N x_N,$$

$$J_k(x_k) = \min_{u_k} E\big\{ x'_k Q_k x_k + u'_k R_k u_k + J_{k+1}(A_k x_k + B_k u_k + w_k) \big\}$$

- Key facts:
  - $J_k(x_k)$ is quadratic
  - Optimal policy $\{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is linear:

$$\mu_k^*(x_k) = L_k x_k$$

  - Similar treatment of a number of variants

DERIVATION

- By induction verify that

$$\mu_k^*(x_k) = L_k x_k, \qquad J_k(x_k) = x_k' K_k x_k + \text{constant},$$

where L_k are matrices given by

$$L_k = -(B_k' K_{k+1} B_k + R_k)^{-1} B_k' K_{k+1} A_k,$$

and where K_k are symmetric positive semidefinite matrices given by

$$K_N = Q_N,$$

$$K_k = A_k' \big(K_{k+1} - K_{k+1} B_k (B_k' K_{k+1} B_k + R_k)^{-1} B_k' K_{k+1} \big) A_k + Q_k$$

- This is called the discrete-time Riccati equation
- Just like DP, it starts at the terminal time N and proceeds backwards.
- Certainty equivalence holds (optimal policy is the same as when w_k is replaced by its expected value $E\{w_k\} = 0$).

ASYMPTOTIC BEHAVIOR OF RICCATI EQ.

- Assume stationary system and cost per stage, and technical assumptions: controlability of (A, B) and observability of (A, C) where $Q = C'C$

- The Riccati equation converges $\lim_{k\to-\infty} K_k = K$, where K is pos. definite, and is the unique (within the class of pos. semidefinite matrices) solution of the algebraic Riccati equation

$$K = A'\big(K - KB(B'KB + R)^{-1}B'K\big)A + Q$$

- The optimal steady-state controller $\mu^*(x) = Lx$

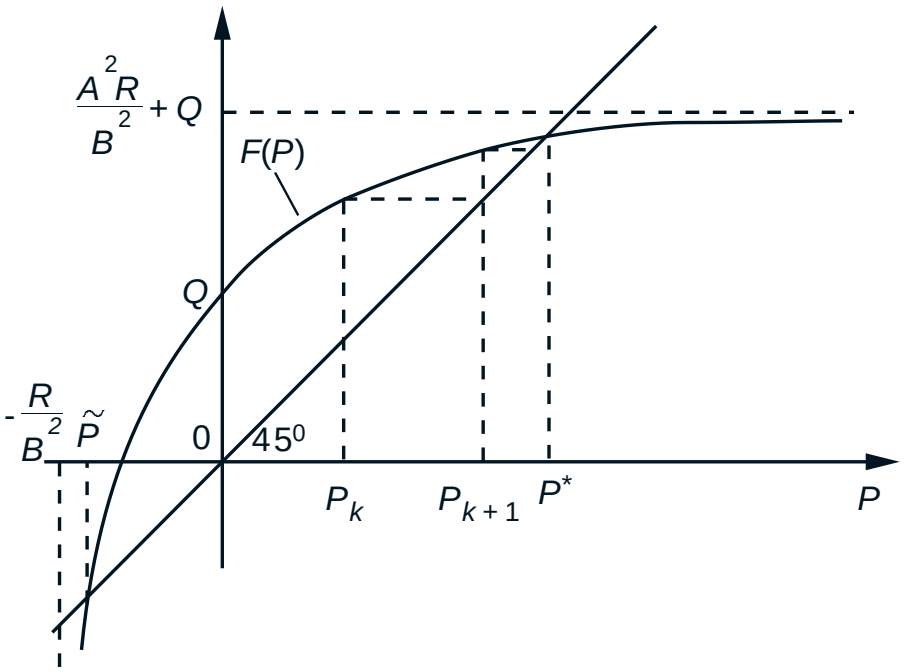
$$L = -(B'KB + R)^{-1}B'KA,$$

is stable in the sense that the matrix $(A + BL)$ of the closed-loop system

$$x_{k+1} = (A + BL)x_k + w_k$$

satisfies $\lim_{k\to\infty}(A + BL)^k = 0$.

GRAPHICAL PROOF FOR SCALAR SYSTEMS



- Riccati equation (with $P_k = K_{N-k}$):

$$P_{k+1} = A^2 \left(P_k - \frac{B^2 P_k^2}{B^2 P_k + R} \right) + Q,$$

or $P_{k+1} = F(P_k)$, where

$$F(P) = A^2 \left(P - \frac{B^2 P^2}{B^2 P + R} \right) + Q = \frac{A^2 R P}{B^2 P + R} + Q$$

- Note the two steady-state solutions, satisfying $P = F(P)$, of which only one is positive.

RANDOM SYSTEM MATRICES

- Suppose that $\{A_0, B_0\}, \ldots, \{A_{N-1}, B_{N-1}\}$ are not known but rather are independent random matrices that are also independent of the w_k
- DP algorithm is

$$J_N(x_N) = x_N' Q_N x_N,$$

$$J_k(x_k) = \min_{u_k} \underset{w_k, A_k, B_k}{E} \Big\{ x_k' Q_k x_k + u_k' R_k u_k + J_{k+1}(A_k x_k + B_k u_k + w_k) \Big\}$$

- Optimal policy $\mu_k^*(x_k) = L_k x_k$, where

$$L_k = -\big(R_k + E\{B_k' K_{k+1} B_k\}\big)^{-1} E\{B_k' K_{k+1} A_k\},$$

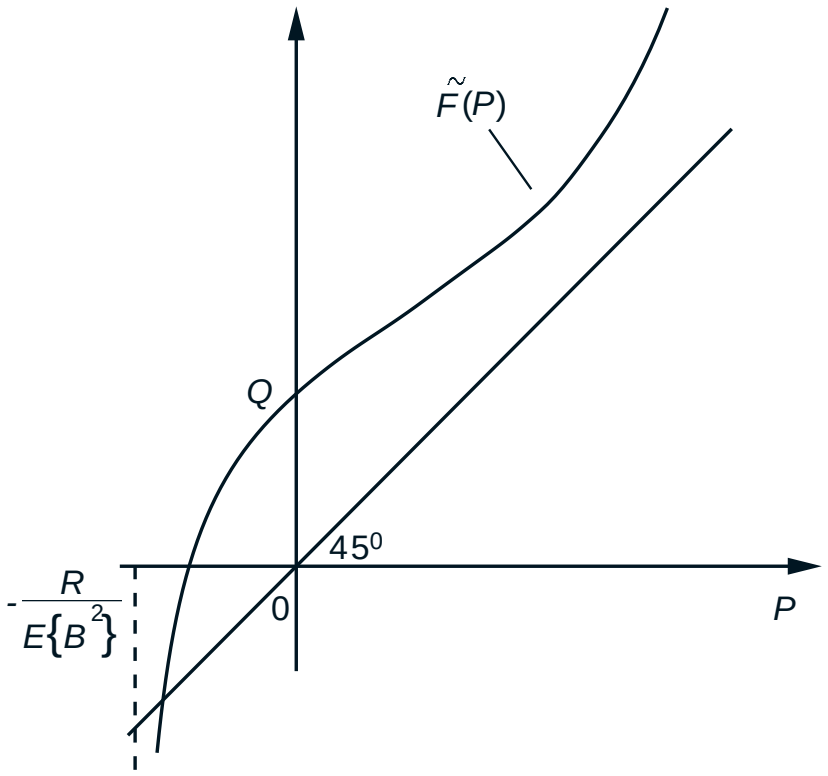
and where the matrices K_k are given by

$$K_N = Q_N,$$

$$K_k = E\{A_k' K_{k+1} A_k\} - E\{A_k' K_{k+1} B_k\} \big(R_k + E\{B_k' K_{k+1} B_k\}\big)^{-1} E\{B_k' K_{k+1} A_k\} + Q_k$$

PROPERTIES

- Certainty equivalence may not hold
- Riccati equation may not converge to a steady-state



- We have $P_{k+1} = \tilde{F}(P_k)$, where

$$\tilde{F}(P) = \frac{E\{A^2\}RP}{E\{B^2\}P + R} + Q + \frac{TP^2}{E\{B^2\}P + R},$$

$$T = E\{A^2\}E\{B^2\} - \big(E\{A\}\big)^2\big(E\{B\}\big)^2$$

INVENTORY CONTROL

- x_k: stock, u_k: stock purchased, w_k: demand

$$x_{k+1} = x_k + u_k - w_k, \qquad k = 0, 1, \ldots, N-1$$

- Minimize

$$E\left\{\sum_{k=0}^{N-1}\big(cu_k + H(x_k + u_k)\big)\right\}$$

where

$$H(x+u) = E\{r(x+u-w)\}$$

is the expected shortage/holding cost, with r defined e.g., for some $p > 0$ and $h > 0$, as

$$r(x) = p\max(0, -x) + h\max(0, x)$$

- DP algorithm:

$$J_N(x_N) = 0,$$

$$J_k(x_k) = \min_{u_k \geq 0}\Big[cu_k + H(x_k+u_k) + E\big\{J_{k+1}(x_k+u_k-w_k)\big\}\Big]$$

OPTIMAL POLICY

- DP algorithm can be written as $J_N(x_N) = 0$,

$$J_k(x_k) = \min_{u_k \geq 0} \Big[cu_k + H(x_k + u_k) + E\Big\{ J_{k+1}(x_k + u_k - w_k) \Big\} \Big]$$

$$= \min_{u_k \geq 0} G_k(x_k + u_k) - cx_k = \min_{y \geq x_k} G_k(y) - cx_k,$$

where

$$G_k(y) = cy + H(y) + E\Big\{ J_{k+1}(y - w) \Big\}$$

- If G_k is convex and $\lim_{|x| \to \infty} G_k(x) \to \infty$, we have

$$\mu_k^*(x_k) = \begin{cases} S_k - x_k & \text{if } x_k < S_k, \\ 0 & \text{if } x_k \geq S_k, \end{cases}$$

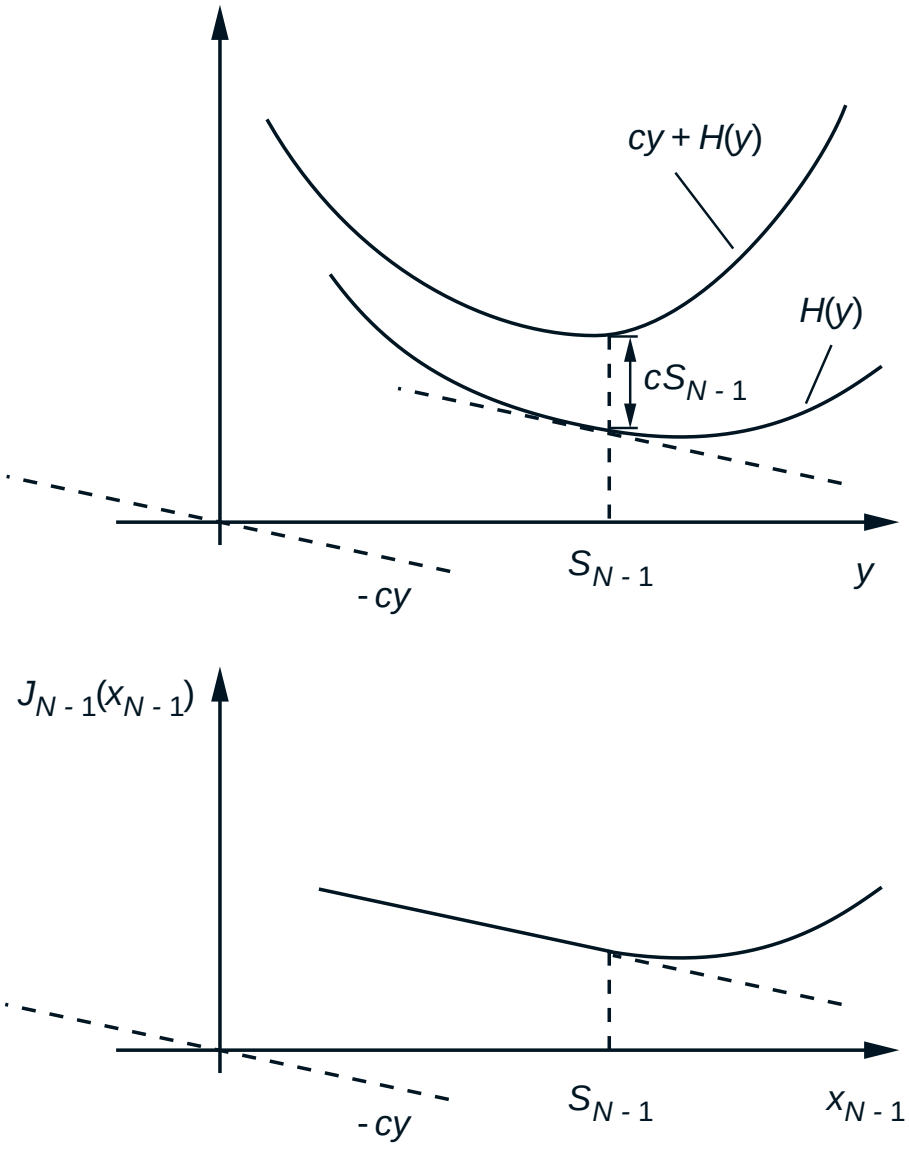
where S_k minimizes $G_k(y)$.

- This is shown, assuming that H is convex and $c < p$, by showing that J_k is convex for all k, and

$$\lim_{|x| \to \infty} J_k(x) \to \infty$$

JUSTIFICATION

- Graphical inductive proof that J_k is convex.



6.231 DYNAMIC PROGRAMMING

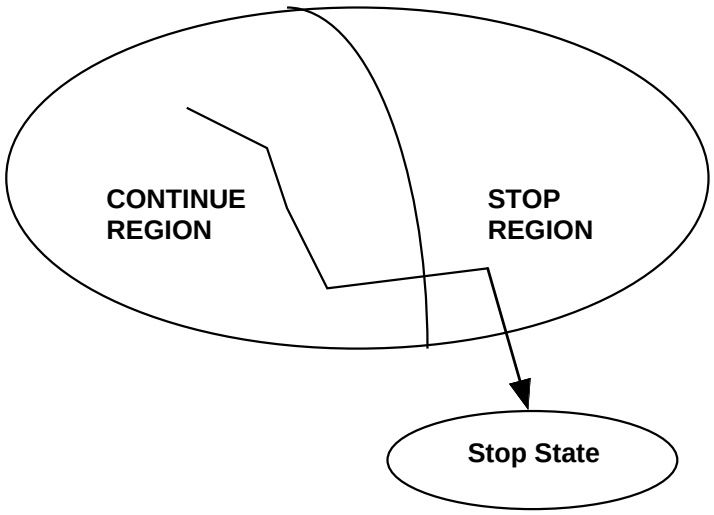
LECTURE 5

LECTURE OUTLINE

- Stopping problems
- Scheduling problems
- Minimax Control

PURE STOPPING PROBLEMS

- Two possible controls:
  - Stop (incur a one-time stopping cost, and move to cost-free and absorbing stop state)
  - Continue [using $x_{k+1} = f_k(x_k, w_k)$ and incurring the cost-per-stage]
- Each policy consists of a partition of the set of states x_k into two regions:
  - Stop region, where we stop
  - Continue region, where we continue



EXAMPLE: ASSET SELLING

- A person has an asset, and at $k = 0, 1, \ldots, N-1$ receives a random offer w_k
- May accept w_k and invest the money at fixed rate of interest r, or reject w_k and wait for w_{k+1}. Must accept the last offer w_{N-1}
- DP algorithm (x_k: current offer, T: stop state):

$$J_N(x_N) = \begin{cases} x_N & \text{if } x_N \neq T, \\ 0 & \text{if } x_N = T, \end{cases}$$

$$J_k(x_k) = \begin{cases} \max\Big[(1+r)^{N-k} x_k,\ E\Big\{J_{k+1}(w_k)\Big\}\Big] & \text{if } x_k \neq T, \\ 0 & \text{if } x_k = T. \end{cases}$$

- Optimal policy;

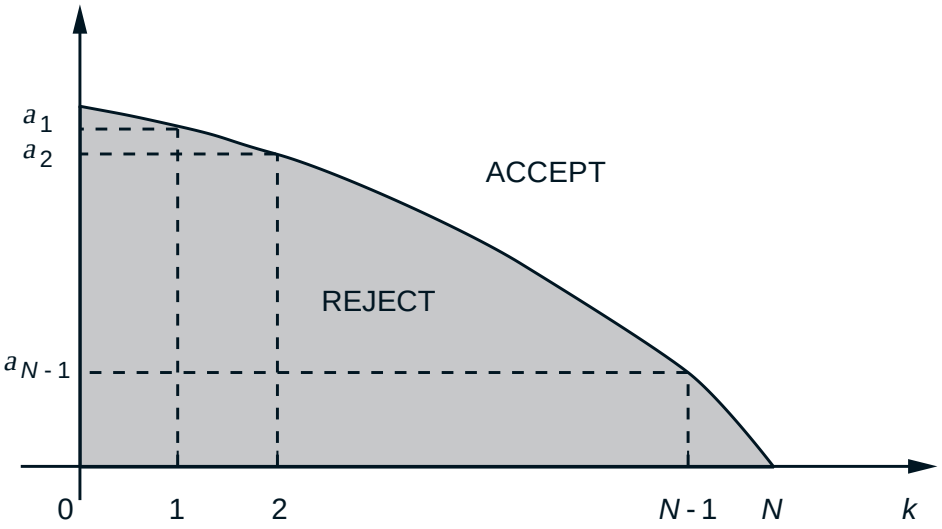
$$\text{accept the offer } x_k \qquad \text{if } x_k > \alpha_k,$$

$$\text{reject the offer } x_k \qquad \text{if } x_k < \alpha_k,$$

where

$$\alpha_k = \frac{E\Big\{J_{k+1}(w_k)\Big\}}{(1+r)^{N-k}}.$$

FURTHER ANALYSIS



- Can show that $\alpha_k \geq \alpha_{k+1}$ for all k
- Proof: Let $V_k(x_k) = J_k(x_k)/(1+r)^{N-k}$ for $x_k \neq T$. Then the DP algorithm is

$$V_N(x_N) = x_N, \quad V_k(x_k) = \max\left[x_k, \, (1+r)^{-1} \underset{w}{E}\{V_{k+1}(w)\}\right]$$

We have $\alpha_k = E_w\{V_{k+1}(w)\}/(1+r)$, so it is enough to show that $V_k(x) \geq V_{k+1}(x)$ for all x and k. Start with $V_{N-1}(x) \geq V_N(x)$ and use the monotonicity property of DP. **Q.E.D.**

- We can also show that if w is bounded, $\alpha_k \to \overline{a}$ as $k \to -\infty$. Suggests that for an infinite horizon the optimal policy is stationary.

GENERAL STOPPING PROBLEMS

- At time k, we may stop at cost $t(x_k)$ or choose a control $u_k \in U(x_k)$ and continue

$$J_N(x_N) = t(x_N),$$

$$J_k(x_k) = \min\Big[t(x_k), \min_{u_k \in U(x_k)} E\Big\{g(x_k, u_k, w_k) + J_{k+1}\Big(f(x_k, u_k, w_k)\Big)\Big\}\Big]$$

- Optimal to stop at time k for x in the set

$$T_k = \left\{ x \;\middle|\; t(x) \leq \min_{u \in U(x)} E\Big\{g(x, u, w) + J_{k+1}\Big(f(x, u, w)\Big)\Big\} \right\}$$

- Since $J_{N-1}(x) \leq J_N(x)$, we have $J_k(x) \leq J_{k+1}(x)$ for all k, so

$$T_0 \subset \cdots \subset T_k \subset T_{k+1} \subset \cdots \subset T_{N-1}.$$

- Interesting case is when all the T_k are equal (to T_{N-1}, the set where it is better to stop than to go one step and stop). Can be shown to be true if

$$f(x, u, w) \in T_{N-1}, \qquad \text{for all } x \in T_{N-1},\ u \in U(x),\ w.$$

SCHEDULING PROBLEMS

- We have a set of tasks to perform, the ordering is subject to optimal choice.
- Costs depend on the order
- There may be stochastic uncertainty, and precedence and resource availability constraints
- Some of the hardest combinatorial problems are of this type (e.g., traveling salesman, vehicle routing, etc.)
- Some special problems admit a simple quasi-analytical solution method
  - Optimal policy has an "index form", i.e., each task has an easily calculable "cost index", and it is optimal to select the task that has the minimum value of index (multi-armed bandit problems - to be discussed later)
  - Some problems can be solved by an "interchange argument" (start with some schedule, interchange two adjacent tasks, and see what happens). They require existence of an optimal policy which is open-loop.

EXAMPLE: THE QUIZ PROBLEM

- Given a list of N questions. If question i is answered correctly (given probability p_i), we receive reward R_i; if not the quiz terminates. Choose order of questions to maximize expected reward.

- Let i and j be the kth and $(k+1)$st questions in an optimally ordered list

$$L = (i_0, \ldots, i_{k-1}, i, j, i_{k+2}, \ldots, i_{N-1})$$

$$\begin{aligned} E\{\text{reward of } L\} &= E\big\{\text{reward of } \{i_0, \ldots, i_{k-1}\}\big\} \\ &+ p_{i_0} \cdots p_{i_{k-1}} (p_i R_i + p_i p_j R_j) \\ &+ p_{i_0} \cdots p_{i_{k-1}} p_i p_j E\big\{\text{reward of } \{i_{k+2}, \ldots, i_{N-1}\}\big\} \end{aligned}$$

Consider the list with i and j interchanged

$$L' = (i_0, \ldots, i_{k-1}, j, i, i_{k+2}, \ldots, i_{N-1})$$

Since L is optimal, $E\{\text{reward of } L\} \geq E\{\text{reward of } L'\}$, so it follows that $p_i R_i + p_i p_j R_j \geq p_j R_j + p_j p_i R_i$ or

$$p_i R_i/(1-p_i) \geq p_j R_j/(1-p_j).$$

MINIMAX CONTROL

- Consider basic problem with the difference that the disturbance w_k instead of being random, it is just known to belong to a given set $W_k(x_k, u_k)$.
- Find policy π that minimizes the cost

$$J_\pi(x_0) = \max_{\substack{w_k \in W_k(x_k, \mu_k(x_k)) \\ k=0,1,\ldots,N-1}} \Big[g_N(x_N) + \sum_{k=0}^{N-1} g_k\big(x_k, \mu_k(x_k), w_k\big) \Big]$$

- The DP algorithm takes the form

$$J_N(x_N) = g_N(x_N),$$

$$J_k(x_k) = \min_{u_k \in U(x_k)} \max_{w_k \in W_k(x_k, u_k)} \Big[g_k(x_k, u_k, w_k) + J_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big]$$

(Section 1.6 in the text).

DERIVATION OF MINIMAX DP ALGORITHM

- Similar to the DP algorithm for stochastic problems. The optimal cost $J^*(x_0)$ is

$$J^*(x_0) = \min_{\mu_0} \cdots \min_{\mu_{N-1}} \max_{w_0 \in W[x_0, \mu_0(x_0)]} \cdots \max_{w_{N-1} \in W[x_{N-1}, \mu_{N-1}(x_{N-1})]}$$

$$\left[\sum_{k=0}^{N-1} g_k\Big(x_k, \mu_k(x_k), w_k\Big) + g_N(x_N) \right]$$

$$= \min_{\mu_0} \cdots \min_{\mu_{N-2}} \left[\min_{\mu_{N-1}} \max_{w_0 \in W[x_0, \mu_0(x_0)]} \cdots \max_{w_{N-2} \in W[x_{N-2}, \mu_{N-2}(x_{N-2})]} \right.$$

$$\left[\sum_{k=0}^{N-2} g_k\Big(x_k, \mu_k(x_k), w_k\Big) + \max_{w_{N-1} \in W[x_{N-1}, \mu_{N-1}(x_{N-1})]} \right.$$

$$\left. \left. \Big[g_{N-1}\Big(x_{N-1}, \mu_{N-1}(x_{N-1}), w_{N-1}\Big) + J_N(x_N) \Big] \right] \right]$$

- Interchange the min over μ_{N-1} and the max over $w_0, \ldots, w_{N-2}$, and similarly continue backwards, with $N-1$ in place of N, etc. After N steps we obtain $J^*(x_0) = J_0(x_0)$.

- Construct optimal policy by minimizing in the RHS of the DP algorithm.

UNKNOWN-BUT-BOUNDED CONTROL

- For each k, keep the x_k of the controlled system

$$x_{k+1} = f_k\big(x_k, \mu_k(x_k), w_k\big)$$

inside a given set X_k, the *target set at time* k.

- This is a minimax control problem, where the cost at stage k is

$$g_k(x_k) = \begin{cases} 0 & \text{if } x_k \in X_k, \\ 1 & \text{if } x_k \notin X_k. \end{cases}$$

- We must reach at time k the set

$$\overline{X}_k = \big\{x_k \mid J_k(x_k) = 0\big\}$$

in order to be able to maintain the state within the subsequent target sets.

- Start with $\overline{X}_N = X_N$, and for $k = 0, 1, \ldots, N-1$,

$$\overline{X}_k = \big\{x_k \in X_k \mid \text{ there exists } u_k \in U_k(x_k) \text{ such that } f_k(x_k, u_k, w_k) \in \overline{X}_{k+1}, \text{ for all } w_k \in W_k(x_k, u_k)\big\}$$

6.231 DYNAMIC PROGRAMMING

LECTURE 6

LECTURE OUTLINE

- Problems with imperfect state info
- Reduction to the perfect state info case
- Linear quadratic problems
- Separation of estimation and control

BASIC PROBL. W/ IMPERFECT STATE INFO

- Same as basic problem of Chapter 1 with one difference: the controller, instead of knowing x_k, receives at each time k an observation of the form

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k(x_k, u_{k-1}, v_k), \quad k \geq 1$$

- The observation z_k belongs to some space Z_k.
- The random observation disturbance v_k is characterized by a probability distribution

$$P_{v_k}(\cdot \mid x_k, \ldots, x_0, u_{k-1}, \ldots, u_0, w_{k-1}, \ldots, w_0, v_{k-1}, \ldots, v_0)$$

- The initial state x_0 is also random and characterized by a probability distribution P_{x_0}.
- The probability distribution $P_{w_k}(\cdot \mid x_k, u_k)$ of w_k is given, and it may depend explicitly on x_k and u_k but not on $w_0, \ldots, w_{k-1}, v_0, \ldots, v_{k-1}$.
- The control u_k is constrained to a given subset U_k (this subset does not depend on x_k, which is not assumed known).

INFORMATION VECTOR AND POLICIES

- Denote by I_k the information vector, i.e., the information available at time k:

$$I_k = (z_0, z_1, \ldots, z_k, u_0, u_1, \ldots, u_{k-1}), \quad k \geq 1,$$

$$I_0 = z_0$$

- We consider policies $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, where each μ_k maps I_k into a u_k and

$$\mu_k(I_k) \in U_k, \qquad \text{for all } I_k,\ k \geq 0$$

- We want to find a policy π that minimizes

$$J_\pi = \mathop{E}_{\substack{x_0, w_k, v_k \\ k=0,\ldots,N-1}} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k\big(x_k, \mu_k(I_k), w_k\big) \right\}$$

subject to the equations

$$x_{k+1} = f_k\big(x_k, \mu_k(I_k), w_k\big), \qquad k \geq 0,$$

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k\big(x_k, \mu_{k-1}(I_{k-1}), v_k\big), \quad k \geq 1$$

REFORMULATION AS PERFECT INFO PROBL.

- System: We have

$$I_{k+1} = (I_k, z_{k+1}, u_k), \quad k = 0, 1, \ldots, N-2, \quad I_0 = z_0$$

View this as a dynamic system with state I_k, control u_k, and random disturbance z_{k+1}

- Disturbance: We have

$$P(z_{k+1} \mid I_k, u_k) = P(z_{k+1} \mid I_k, u_k, z_0, z_1, \ldots, z_k),$$

since $z_0, z_1, \ldots, z_k$ are part of the information vector I_k. Thus the probability distribution of z_{k+1} depends explicitly only on the state I_k and control u_k and not on the prior "disturbances" $z_k, \ldots, z_0$

- Cost Function: Write

$$E\Big\{g_k(x_k, u_k, w_k)\Big\} = E\left\{\underset{x_k, w_k}{E}\Big\{g_k(x_k, u_k, w_k) \mid I_k, u_k\Big\}\right\}$$

so the cost per stage of the new system is

$$\tilde{g}_k(I_k, u_k) = \underset{x_k, w_k}{E}\Big\{g_k(x_k, u_k, w_k) \mid I_k, u_k\Big\}$$

DP ALGORITHM

- Writing the DP algorithm for the (reformulated) perfect state info problem:

$$J_k(I_k) = \min_{u_k \in U_k} \Big[\underset{x_k,\, w_k,\, z_{k+1}}{E} \Big\{ g_k(x_k, u_k, w_k) + J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \Big\} \Big]$$

for $k = 0, 1, \ldots, N-2$, and for $k = N-1$,

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1} \in U_{N-1}} \Big[\underset{x_{N-1},\, w_{N-1}}{E} \Big\{ g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) + g_N\big(f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})\big) \mid I_{N-1}, u_{N-1} \Big\} \Big]$$

- The optimal cost J^* is given by

$$J^* = \underset{z_0}{E}\Big\{ J_0(z_0) \Big\}$$

LINEAR-QUADRATIC PROBLEMS

- System: $x_{k+1} = A_k x_k + B_k u_k + w_k$
- Quadratic cost

$$\underset{\substack{w_k \\ k=0,1,\ldots,N-1}}{E} \left\{ x_N' Q_N x_N + \sum_{k=0}^{N-1} (x_k' Q_k x_k + u_k' R_k u_k) \right\}$$

where $Q_k \geq 0$ and $R_k > 0$

- Observations

$$z_k = C_k x_k + v_k, \qquad k = 0, 1, \ldots, N-1$$

- $w_0, \ldots, w_{N-1}$, $v_0, \ldots, v_{N-1}$ indep. zero mean
- Key fact to show:
  - Optimal policy $\{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is of the form:

$$\mu_k^*(I_k) = L_k E\{x_k \mid I_k\}$$

    L_k: same as for the perfect state info case
  - Estimation problem and control problem can be solved separately

DP ALGORITHM I

- Last stage $N-1$ (supressing index $N-1$):

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1}} \Big[E_{x_{N-1},w_{N-1}} \big\{ x'_{N-1} Q x_{N-1} + u'_{N-1} R u_{N-1} + (Ax_{N-1} + Bu_{N-1} + w_{N-1})' \cdot Q(Ax_{N-1} + Bu_{N-1} + w_{N-1}) \mid I_{N-1}, u_{N-1} \big\} \Big]$$

- Since $E\{w_{N-1} \mid I_{N-1}, u_{N-1}\} = E\{w_{N-1}\} = 0$, the minimization involves

$$\min_{u_{N-1}} \Big[u'_{N-1}(B'QB + R)u_{N-1} + 2E\{x_{N-1} \mid I_{N-1}\}' A'QBu_{N-1} \Big]$$

The minimization yields the optimal μ^*_{N-1}:

$$u^*_{N-1} = \mu^*_{N-1}(I_{N-1}) = L_{N-1} E\{x_{N-1} \mid I_{N-1}\}$$

where

$$L_{N-1} = -(B'QB + R)^{-1} B'QA$$

DP ALGORITHM II

- Substituting in the DP algorithm

$$J_{N-1}(I_{N-1}) = \underset{x_{N-1}}{E}\Big\{x'_{N-1}K_{N-1}x_{N-1} \mid I_{N-1}\Big\}$$
$$+ \underset{x_{N-1}}{E}\Big\{\Big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\Big)'$$
$$\cdot P_{N-1}\Big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\Big) \mid I_{N-1}\Big\}$$
$$+ \underset{w_{N-1}}{E}\{w'_{N-1}Q_N w_{N-1}\},$$

where the matrices K_{N-1} and P_{N-1} are given by

$$P_{N-1} = A'_{N-1}Q_N B_{N-1}(R_{N-1} + B'_{N-1}Q_N B_{N-1})^{-1}$$
$$\cdot B'_{N-1}Q_N A_{N-1},$$
$$K_{N-1} = A'_{N-1}Q_N A_{N-1} - P_{N-1} + Q_{N-1}$$

- Note the structure of J_{N-1}: in addition to the quadratic and constant terms, it involves a (≥ 0) quadratic in the estimation error

$$x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}$$

DP ALGORITHM III

- DP equation for period $N-2$:

$$J_{N-2}(I_{N-2}) = \min_{u_{N-2}} \Big[E_{x_{N-2},w_{N-2},z_{N-1}} \{x'_{N-2}Qx_{N-2} + u'_{N-2}Ru_{N-2} + J_{N-1}(I_{N-1}) \mid I_{N-2}, u_{N-2}\}\Big]$$

$$= E\Big\{x'_{N-2}Qx_{N-2} \mid I_{N-2}\Big\} + \min_{u_{N-2}} \Big[u'_{N-2}Ru_{N-2} + E\Big\{x'_{N-1}K_{N-1}x_{N-1} \mid I_{N-2}, u_{N-2}\Big\}\Big]$$

$$+ E\Big\{\Big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\Big)' \cdot P_{N-1}\Big(x_{N-1} - E\{x_{N-1} \mid I_{N-1}\}\Big) \mid I_{N-2}, u_{N-2}\Big\}$$

$$+ E_{w_{N-1}}\{w'_{N-1}Q_N w_{N-1}\}$$

- Key point: We have excluded the estimation error term from the minimization over u_{N-2}
- This term turns out to be independent of u_{N-2}

QUALITY OF ESTIMATION LEMMA

- Current estimation error is unaffected by past controls: For every k, there is a function M_k s.t.

$$x_k - E\{x_k \mid I_k\} = M_k(x_0, w_0, \ldots, w_{k-1}, v_0, \ldots, v_k),$$

independently of the policy being used

- Consequence: Using the lemma,

$$x_{N-1} - E\{x_{N-1} \mid I_{N-1}\} = \xi_{N-1},$$

where

ξ_{N-1}: function of $x_0, w_0, \ldots, w_{N-2}, v_0, \ldots, v_{N-1}$

- Since ξ_{N-1} is independent of u_{N-2}, the conditional expectation of $\xi'_{N-1} P_{N-1} \xi_{N-1}$ satisfies

$$E\{\xi'_{N-1} P_{N-1} \xi_{N-1} \mid I_{N-2}, u_{N-2}\} = E\{\xi'_{N-1} P_{N-1} \xi_{N-1} \mid I_{N-2}\}$$

and is independent of u_{N-2}.

- So minimization in the DP algorithm yields

$$u^*_{N-2} = \mu^*_{N-2}(I_{N-2}) = L_{N-2}\, E\{x_{N-2} \mid I_{N-2}\}$$

FINAL RESULT

• Continuing similarly (using also the quality of estimation lemma)

$$\mu_k^*(I_k) = L_k E\{x_k \mid I_k\},$$

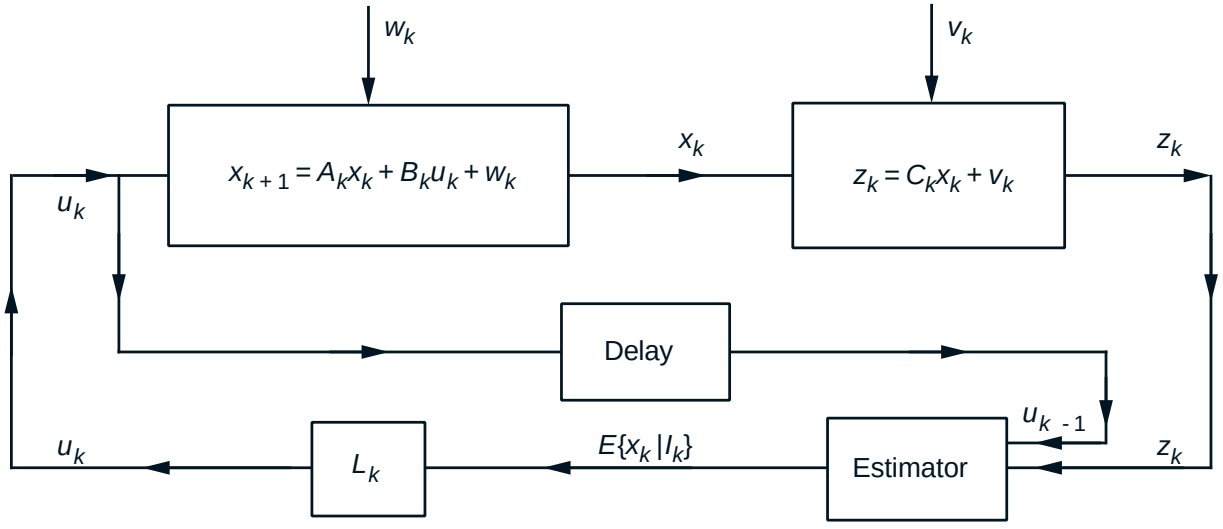
where L_k is the same as for perfect state info:

$$L_k = -(R_k + B_k' K_{k+1} B_k)^{-1} B_k' K_{k+1} A_k,$$

with K_k generated using the Riccati equation:

$$K_N = Q_N, \qquad K_k = A_k' K_{k+1} A_k - P_k + Q_k,$$

$$P_k = A_k' K_{k+1} B_k (R_k + B_k' K_{k+1} B_k)^{-1} B_k' K_{k+1} A_k$$



SEPARATION INTERPRETATION

- The optimal controller can be decomposed into
  - (a) An estimator, which uses the data to generate the conditional expectation $E\{x_k \mid I_k\}$.
  - (b) An actuator, which multiplies $E\{x_k \mid I_k\}$ by the gain matrix L_k and applies the control input $u_k = L_k E\{x_k \mid I_k\}$.
- Generically the estimate $\hat{x}$ of a random vector x given some information (random vector) I, which minimizes the mean squared error

$$E_x\{\|x - \hat{x}\|^2 \mid I\} = \|x\|^2 - 2E\{x \mid I\}\hat{x} + \|\hat{x}\|^2$$

is $E\{x \mid I\}$ (set to zero the derivative with respect to $\hat{x}$ of the above quadratic form).

- The estimator portion of the optimal controller is optimal for the problem of estimating the state x_k assuming the control is not subject to choice.
- The actuator portion is optimal for the control problem assuming perfect state information.

STEADY STATE/IMPLEMENTATION ASPECTS

- As $N \to \infty$, the solution of the Riccati equation converges to a steady state and $L_k \to L$.
- If x_0, w_k, and v_k are Gaussian, $E\{x_k \mid I_k\}$ is a linear function of I_k and is generated by a nice recursive algorithm, the Kalman filter.
- The Kalman filter involves also a Riccati equation, so for $N \to \infty$, and a stationary system, it also has a steady-state structure.
- Thus, for Gaussian uncertainty, the solution is nice and possesses a steady state.
- For nonGaussian uncertainty, computing $E\{x_k \mid I_k\}$ maybe very difficult, so a suboptimal solution is typically used.
- Most common suboptimal controller: Replace $E\{x_k \mid I_k\}$ by the estimate produced by the Kalman filter (act as if x_0, w_k, and v_k are Gaussian).
- It can be shown that this controller is optimal within the class of controllers that are linear functions of I_k.

6.231 DYNAMIC PROGRAMMING

LECTURE 7

LECTURE OUTLINE

- DP for imperfect state info
- Sufficient statistics
- Conditional state distribution as a sufficient statistic
- Finite-state systems
- Examples

REVIEW: IMPERFECT STATE INFO PROBLEM

- Instead of knowing x_k, we receive observations

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k(x_k, u_{k-1}, v_k), \quad k \geq 0$$

- I_k: information vector available at time k:

$$I_0 = z_0, \quad I_k = (z_0, z_1, \ldots, z_k, u_0, u_1, \ldots, u_{k-1}), \quad k \geq 1$$

- Optimization over policies $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, where $\mu_k(I_k) \in U_k$, for all I_k and k.
- Find a policy π that minimizes

$$J_\pi = \mathop{E}_{\substack{x_0, w_k, v_k \\ k=0,\ldots,N-1}} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k\Big(x_k, \mu_k(I_k), w_k\Big) \right\}$$

subject to the equations

$$x_{k+1} = f_k\Big(x_k, \mu_k(I_k), w_k\Big), \qquad k \geq 0,$$

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k\Big(x_k, \mu_{k-1}(I_{k-1}), v_k\Big), \quad k \geq 1$$

DP ALGORITHM

- DP algorithm:

$$J_k(I_k) = \min_{u_k \in U_k} \Big[\underset{x_k,\, w_k,\, z_{k+1}}{E} \Big\{ g_k(x_k, u_k, w_k) + J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \Big\} \Big]$$

for $k = 0, 1, \ldots, N-2$, and for $k = N-1$,

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1} \in U_{N-1}} \Big[\underset{x_{N-1},\, w_{N-1}}{E} \Big\{ g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) + g_N\big(f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})\big) \mid I_{N-1}, u_{N-1} \Big\} \Big]$$

- The optimal cost J^* is given by

$$J^* = \underset{z_0}{E}\Big\{ J_0(z_0) \Big\}.$$

SUFFICIENT STATISTICS

- Suppose there is a function $S_k(I_k)$ such that the min in the right-hand side of the DP algorithm can be written in terms of some function H_k as

$$\min_{u_k \in U_k} H_k\Big(S_k(I_k), u_k\Big)$$

- Such a function S_k is called a sufficient statistic.
- An optimal policy obtained by the preceding minimization can be written as

$$\mu_k^*(I_k) = \overline{\mu}_k\Big(S_k(I_k)\Big),$$

where $\overline{\mu}_k$ is an appropriate function.

- Example of a sufficient statistic: $S_k(I_k) = I_k$
- Another important sufficient statistic

$$S_k(I_k) = P_{x_k|I_k},$$

assuming that v_k is characterized by a probability distribution $P_{v_k}(\cdot \mid x_{k-1}, u_{k-1}, w_{k-1})$

DP ALGORITHM IN TERMS OF $P_{X_K|I_K}$

- Filtering Equation: $P_{x_k|I_k}$ is generated recursively by a dynamic system (estimator) of the form

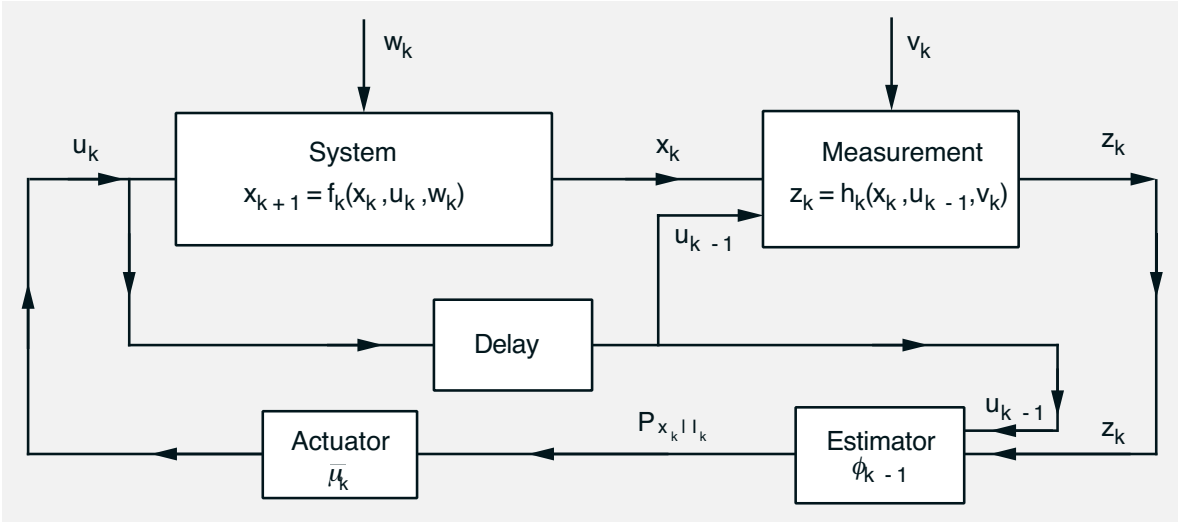
$$P_{x_{k+1}|I_{k+1}} = \Phi_k\left(P_{x_k|I_k}, u_k, z_{k+1}\right)$$

for a suitable function Φ_k

- DP algorithm can be written as

$$\overline{J}_k(P_{x_k|I_k}) = \min_{u_k \in U_k} \Big[\underset{x_k, w_k, z_{k+1}}{E} \Big\{ g_k(x_k, u_k, w_k) + \overline{J}_{k+1}\Big(\Phi_k(P_{x_k|I_k}, u_k, z_{k+1})\Big) \mid I_k, u_k \Big\} \Big]$$

- It is the DP algorithm for a new problem whose state is $P_{x_k|I_k}$ (also called belief state)



EXAMPLE: A SEARCH PROBLEM

- At each period, decide to search or not search a site that may contain a treasure.
- If we search and a treasure is present, we find it with prob. β and remove it from the site.
- Treasure's worth: V. Cost of search: C
- States: treasure present & treasure not present
- Each search can be viewed as an observation of the state
- Denote

p_k : prob. of treasure present at the start of time k

with p_0 given.

- p_k evolves at time k according to the equation

$$p_{k+1} = \begin{cases} p_k & \text{if not search,} \\ 0 & \text{if search and find treasure,} \\ \dfrac{p_k(1-\beta)}{p_k(1-\beta)+1-p_k} & \text{if search and no treasure.} \end{cases}$$

This is the filtering equation.

SEARCH PROBLEM (CONTINUED)

- DP algorithm

$$\overline{J}_k(p_k) = \max\left[0,\ -C + p_k\beta V + (1 - p_k\beta)\overline{J}_{k+1}\left(\frac{p_k(1-\beta)}{p_k(1-\beta)+1-p_k}\right)\right],$$

with $\overline{J}_N(p_N) = 0$.

- Can be shown by induction that the functions $\overline{J}_k$ satisfy

$$\overline{J}_k(p_k)\begin{cases} = 0 & \text{if } p_k \leq \frac{C}{\beta V}, \\ > 0 & \text{if } p_k > \frac{C}{\beta V}. \end{cases}$$

- Furthermore, it is optimal to search at period k if and only if

$$p_k\beta V \geq C$$

(expected reward from the next search $\geq$ the cost of the search - a myopic rule)

FINITE-STATE SYSTEMS - POMDP

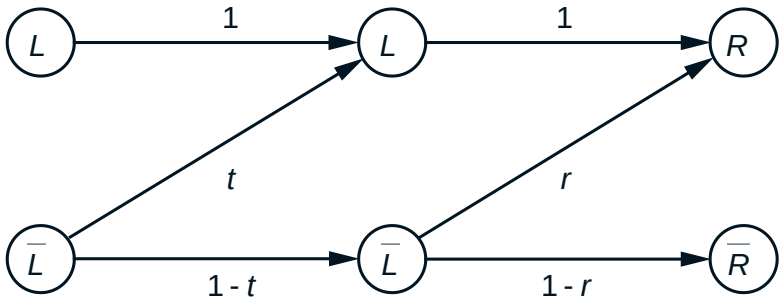
- Suppose the system is a finite-state Markov chain, with states $1, \ldots, n$.
- Then the conditional probability distribution $P_{x_k|I_k}$ is an n-vector

$$\big(P(x_k = 1 \mid I_k), \ldots, P(x_k = n \mid I_k)\big)$$

- The DP algorithm can be executed over the n-dimensional simplex (state space is not expanding with increasing k)
- When the control and observation spaces are also finite sets the problem is called a POMDP (Partially Observed Markov Decision Problem).
- For POMDP it turns out that the cost-to-go functions $\overline{J}_k$ in the DP algorithm are piecewise linear and concave (Exercise 5.7)
- Useful in practice both for exact and approximate computation.

INSTRUCTION EXAMPLE I

- Teaching a student some item. Possible states are L: Item learned, or $\overline{L}$: Item not learned.
- Possible decisions: T: Terminate the instruction, or $\overline{T}$: Continue the instruction for one period and then conduct a test that indicates whether the student has learned the item.
- Possible test outcomes: R: Student gives a correct answer, or $\overline{R}$: Student gives an incorrect answer.
- Probabilistic structure



- Cost of instruction: I per period
- Cost of terminating instruction: 0 if student has learned the item, and $C > 0$ if not.

INSTRUCTION EXAMPLE II

- Let p_k: prob. student has learned the item given the test results so far

$$p_k = P(x_k = L \mid z_0, z_1, \ldots, z_k).$$

- Filtering equation: Using Bayes' rule

$$\begin{aligned} p_{k+1} &= \Phi(p_k, z_{k+1}) \\ &= \begin{cases} \frac{1-(1-t)(1-p_k)}{1-(1-t)(1-r)(1-p_k)} & \text{if } z_{k+1} = R, \\ 0 & \text{if } z_{k+1} = \overline{R}. \end{cases} \end{aligned}$$

- DP algorithm:

$$\overline{J}_k(p_k) = \min \left[(1-p_k)C,\ I + \underset{z_{k+1}}{E} \left\{ \overline{J}_{k+1}\Big(\Phi(p_k, z_{k+1})\Big) \right\} \right]$$

starting with

$$\overline{J}_{N-1}(p_{N-1}) = \min\Big[(1-p_{N-1})C,\ I+(1-t)(1-p_{N-1})C\Big].$$

INSTRUCTION EXAMPLE III

- Write the DP algorithm as

$$\overline{J}_k(p_k) = \min\Big[(1-p_k)C,\ I + A_k(p_k)\Big],$$

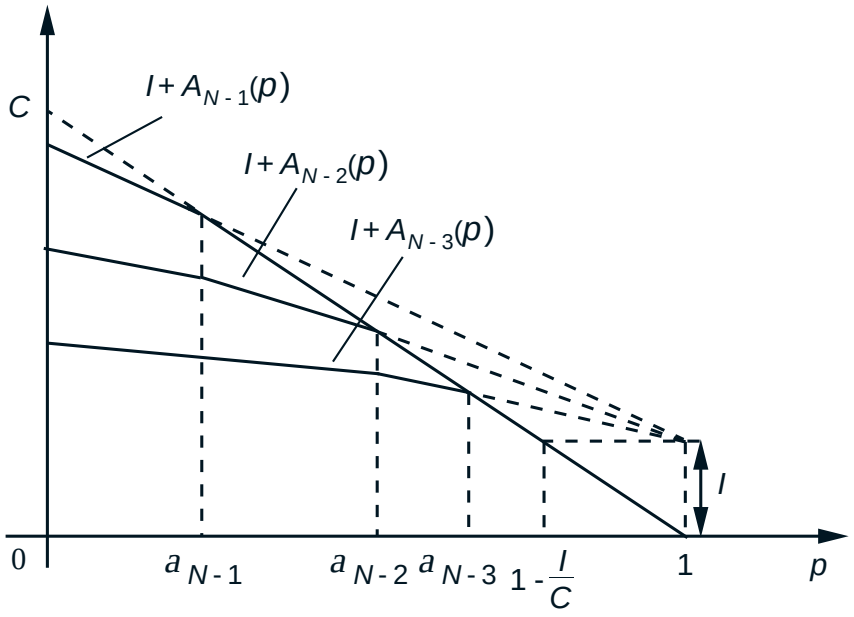
where

$$\begin{aligned} A_k(p_k) = {} & P(z_{k+1} = R \mid I_k)\overline{J}_{k+1}\Big(\Phi(p_k, R)\Big) \\ & + P(z_{k+1} = \overline{R} \mid I_k)\overline{J}_{k+1}\Big(\Phi(p_k, \overline{R})\Big) \end{aligned}$$

- Can show by induction that $A_k(p)$ are piecewise linear, concave, monotonically decreasing, with

$$A_{k-1}(p) \leq A_k(p) \leq A_{k+1}(p), \qquad \text{for all } p \in [0,1].$$

(The cost-to-go at knowledge prob. p increases as we come closer to the end of horizon.)



6.231 DYNAMIC PROGRAMMING

LECTURE 8

LECTURE OUTLINE

- Suboptimal control
- Cost approximation methods: Classification
- Certainty equivalent control: An example
- Limited lookahead policies
- Performance bounds
- Problem approximation approach
- Parametric cost-to-go approximation

PRACTICAL DIFFICULTIES OF DP

- The curse of dimensionality
  - Exponential growth of the computational and storage requirements as the number of state variables and control variables increases
  - Quick explosion of the number of states in combinatorial problems
  - Intractability of imperfect state information problems
- The curse of modeling
  - Mathematical models
  - Computer/simulation models
- There may be real-time solution constraints
  - A family of problems may be addressed. The data of the problem to be solved is given with little advance notice
  - The problem data may change as the system is controlled – need for on-line replanning

COST-TO-GO FUNCTION APPROXIMATION

- Use a policy computed from the DP equation where the optimal cost-to-go function J_{k+1} is replaced by an approximation $\tilde{J}_{k+1}$. (Sometimes $E\{g_k\}$ is also replaced by an approximation.)
- Apply $\overline{\mu}_k(x_k)$, which attains the minimum in

$$\min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\}$$

- There are several ways to compute $\tilde{J}_{k+1}$:
  - Off-line approximation: The entire function $\tilde{J}_{k+1}$ is computed for every k, before the control process begins.
  - On-line approximation: Only the values $\tilde{J}_{k+1}(x_{k+1})$ at the relevant next states x_{k+1} are computed and used to compute u_k just after the current state x_k becomes known.
  - Simulation-based methods: These are off-line and on-line methods that share the common characteristic that they are based on Monte-Carlo simulation. Some of these methods are suitable for are suitable for very large problems.

CERTAINTY EQUIVALENT CONTROL (CEC)

- Idea: Replace the stochastic problem with a deterministic problem
- At each time k, the future uncertain quantities are fixed at some "typical" values
- On-line implementation for a perfect state info problem. At each time k:

(1) Fix the w_i, $i \geq k$, at some $\overline{w}_i$. Solve the deterministic problem:

$$\text{minimize} \quad g_N(x_N) + \sum_{i=k}^{N-1} g_i\big(x_i, u_i, \overline{w}_i\big)$$

where x_k is known, and

$$u_i \in U_i, \quad x_{i+1} = f_i\big(x_i, u_i, \overline{w}_i\big).$$

(2) Use the first control in the optimal control sequence found.

- Equivalently, we apply $\bar{\mu}_k(x_k)$ that minimizes

$$g_k\big(x_k, u_k, \overline{w}_k\big) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, \overline{w}_k)\big)$$

where $\tilde{J}_{k+1}$ is the optimal cost of the corresponding deterministic problem.

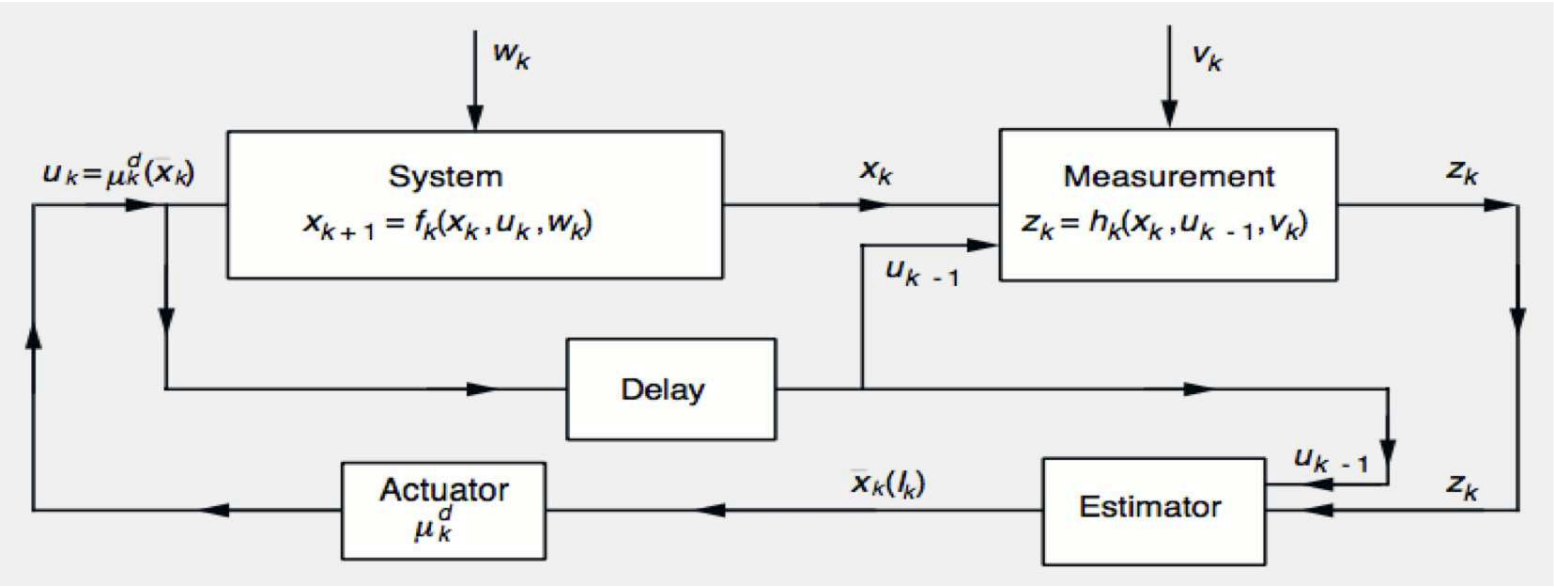
EQUIVALENT OFF-LINE IMPLEMENTATION

- Let $\{\mu_0^d(x_0), \ldots, \mu_{N-1}^d(x_{N-1})\}$ be an optimal controller obtained from the DP algorithm for the deterministic problem

$$\text{minimize} \quad g_N(x_N) + \sum_{k=0}^{N-1} g_k\big(x_k, \mu_k(x_k), \overline{w}_k\big)$$

$$\text{subject to} \quad x_{k+1} = f_k\big(x_k, \mu_k(x_k), \overline{w}_k\big), \quad \mu_k(x_k) \in U_k$$

- The CEC applies at time k the control input $\mu_k^d(x_k)$.
- In an imperfect info version, x_k is replaced by an estimate $\overline{x}_k(I_k)$.



PARTIALLY STOCHASTIC CEC

- Instead of fixing *all* future disturbances to their typical values, fix only some, and treat the rest as stochastic.

- Important special case: Treat an imperfect state information problem as one of perfect state information, using an estimate $\overline{x}_k(I_k)$ of x_k as if it were exact.

- Multiaccess communication example: Consider controlling the slotted Aloha system (Example 5.1.1 in the text) by optimally choosing the probability of transmission of waiting packets. This is a hard problem of imperfect state info, whose perfect state info version is easy.

- Natural partially stochastic CEC:

$$\tilde{\mu}_k(I_k) = \min\left[1, \frac{1}{\overline{x}_k(I_k)}\right],$$

where $\overline{x}_k(I_k)$ is an estimate of the current packet backlog based on the entire past channel history of successes, idles, and collisions (which is I_k).

GENERAL COST-TO-GO APPROXIMATION

- One-step lookahead (1SL) policy: At each k and state x_k, use the control $\overline{\mu}_k(x_k)$ that

$$\min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\},$$

where
  - $\tilde{J}_N = g_N$.
  - $\tilde{J}_{k+1}$: approximation to true cost-to-go J_{k+1}

- Two-step lookahead policy: At each k and x_k, use the control $\tilde{\mu}_k(x_k)$ attaining the minimum above, where the function $\tilde{J}_{k+1}$ is obtained using a 1SL approximation (solve a 2-step DP problem).

- If $\tilde{J}_{k+1}$ is readily available and the minimization above is not too hard, the 1SL policy is implementable on-line.

- Sometimes one also replaces $U_k(x_k)$ above with a subset of "most promising controls" $\overline{U}_k(x_k)$.

- As the length of lookahead increases, the required computation quickly explodes.

PERFORMANCE BOUNDS FOR 1SL

- Let $\overline{J}_k(x_k)$ be the cost-to-go from (x_k, k) of the 1SL policy, based on functions $\tilde{J}_k$.
- Assume that for all (x_k, k), we have

$$\hat{J}_k(x_k) \leq \tilde{J}_k(x_k), \qquad (*)$$

where $\hat{J}_N = g_N$ and for all k,

$$\hat{J}_k(x_k) = \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\},$$

[so $\hat{J}_k(x_k)$ is computed along with $\overline{\mu}_k(x_k)$]. Then

$$\overline{J}_k(x_k) \leq \hat{J}_k(x_k), \qquad \text{for all } (x_k, k).$$

- Important application: When $\tilde{J}_k$ is the cost-to-go of some heuristic policy (then the 1SL policy is called the rollout policy).
- The bound can be extended to the case where there is a δ_k in the RHS of (*). Then

$$\overline{J}_k(x_k) \leq \tilde{J}_k(x_k) + \delta_k + \cdots + \delta_{N-1}$$

COMPUTATIONAL ASPECTS

- Sometimes nonlinear programming can be used to calculate the 1SL or the multistep version [particularly when $U_k(x_k)$ is not a discrete set]. Connection with stochastic programming (2-stage DP) methods (see text).

- The choice of the approximating functions $\tilde{J}_k$ is critical, and is calculated in a variety of ways.

- Some approaches:

  (a) Problem Approximation: Approximate the optimal cost-to-go with some cost derived from a related but simpler problem

  (b) Parametric Cost-to-Go Approximation: Approximate the optimal cost-to-go with a function of a suitable parametric form, whose parameters are tuned by some heuristic or systematic scheme (Neuro-Dynamic Programming)

  (c) Rollout Approach: Approximate the optimal cost-to-go with the cost of some suboptimal policy, which is calculated either analytically or by simulation

PROBLEM APPROXIMATION

- Many (problem-dependent) possibilities
  - Replace uncertain quantities by nominal values, or simplify the calculation of expected values by limited simulation
  - Simplify difficult constraints or dynamics

- Enforced decomposition example: Route m vehicles that move over a graph. Each node has a "value." First vehicle that passes through the node collects its value. Want to max the total collected value, subject to initial and final time constraints (plus time windows and other constraints).

- Usually the 1-vehicle version of the problem is much simpler. This motivates an approximation obtained by solving single vehicle problems.

- 1SL scheme: At time k and state x_k (position of vehicles and "collected value nodes"), consider all possible kth moves by the vehicles, and at the resulting states we approximate the optimal value-to-go with the value collected by optimizing the vehicle routes one-at-a-time

PARAMETRIC COST-TO-GO APPROXIMATION

- Use a cost-to-go approximation from a parametric class $\tilde{J}(x, r)$ where x is the current state and $r = (r_1, \ldots, r_m)$ is a vector of "tunable" scalars (weights).

- By adjusting the weights, one can change the "shape" of the approximation $\tilde{J}$ so that it is reasonably close to the true optimal cost-to-go function.

- Two key issues:
  - The choice of parametric class $\tilde{J}(x, r)$ (the approximation architecture).
  - Method for tuning the weights ("training" the architecture).

- Successful application strongly depends on how these issues are handled, and on insight about the problem.

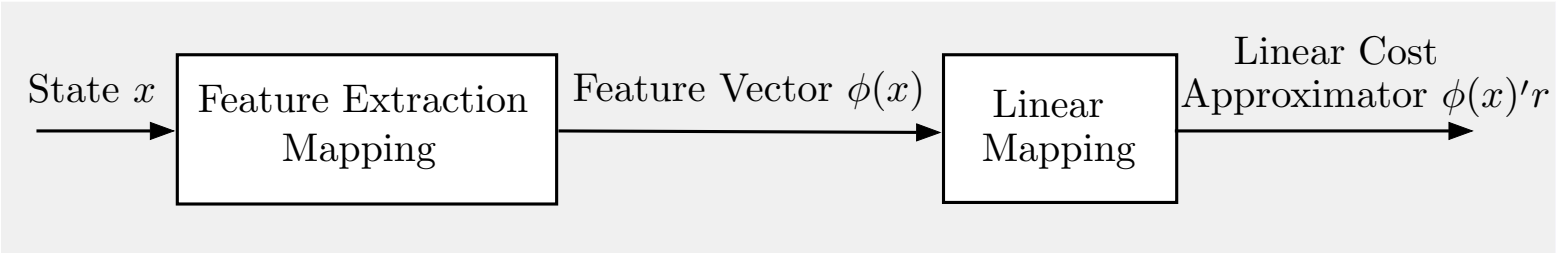
- Sometimes a simulation-based algorithm is used, particularly when there is no mathematical model of the system.

- We will look in detail at these issues after a few lectures.

APPROXIMATION ARCHITECTURES

- Divided in linear and nonlinear [i.e., linear or nonlinear dependence of $\tilde{J}(x, r)$ on r]
- Linear architectures are easier to train, but nonlinear ones (e.g., neural networks) are richer
- Linear feature-based architecture: $\phi = (\phi_1, \ldots, \phi_m)$

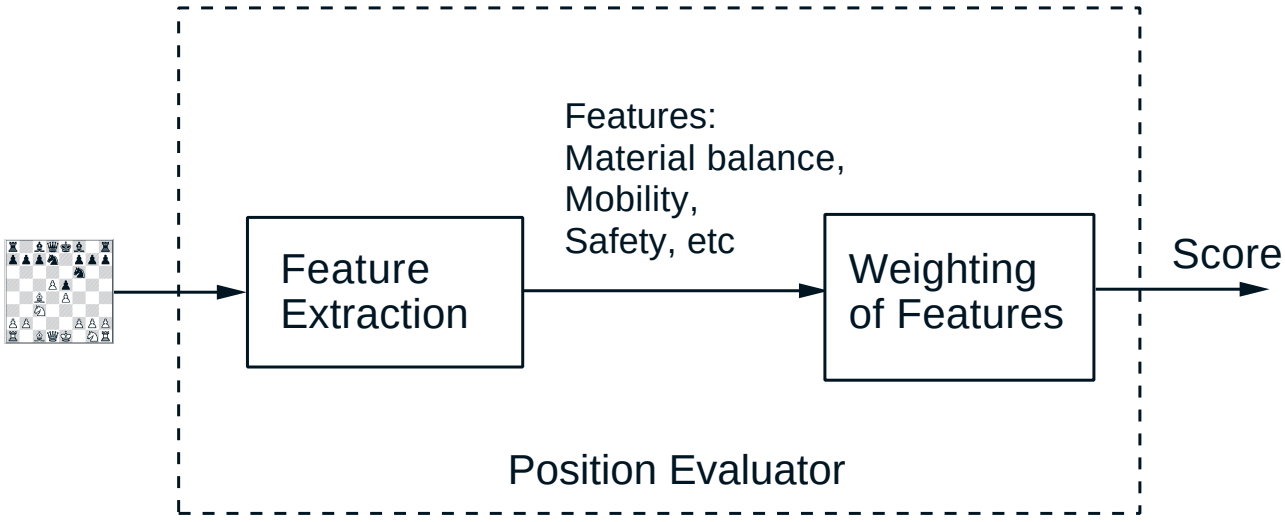
$$\tilde{J}(x, r) = \phi(x)'r = \sum_{j=1}^{m} \phi_j(x) r_j$$



- Ideally, the features will encode much of the nonlinearity that is inherent in the cost-to-go approximated, and the approximation may be quite accurate without a complicated architecture
- Anything sensible can be used as features. Sometimes the state space is partitioned, and "local" features are introduced for each subset of the partition (they are 0 outside the subset)

AN EXAMPLE - COMPUTER CHESS

- Chess programs use a feature-based position evaluator that assigns a score to each move/position



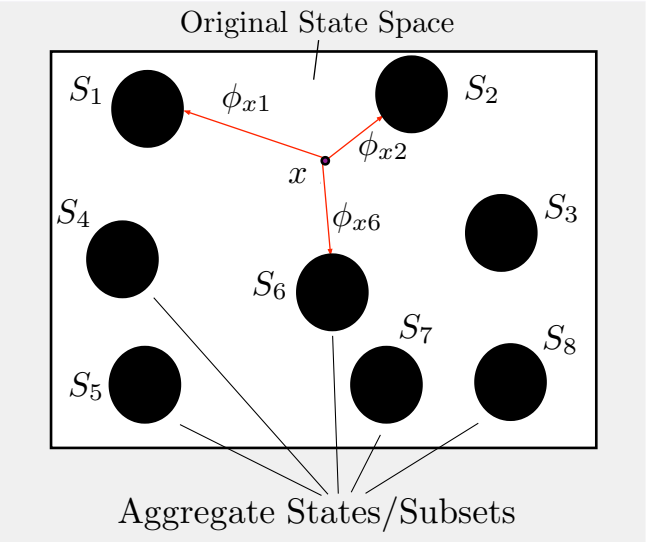
- Many context-dependent special features.
- Most often the weighting of features is linear but multistep lookahead is involved.
- Most often the training is done "manually," by trial and error.

ANOTHER EXAMPLE - AGGREGATION

- Main elements (in a finite-state context):
  - Introduce "aggregate" states $S_1, \ldots, S_m$, viewed as the states of an "aggregate" system
  - Define transition probabilities and costs of the aggregate system, by relating original system states with aggregate states (using so called "aggregation and disaggregation probabilities")
  - Solve (exactly or approximately) the "aggregate" problem by any kind of method (including simulation-based) ... more on this later.
  - Use the optimal cost of the aggregate problem to approximate the optimal cost of each original problem state as a linear combination of the optimal aggregate state costs
- This is a linear feature-based architecture (the optimal aggregate state costs are the features)
- Hard aggregation example: Aggregate states S_j are a partition of original system states (each original state belongs to one and only one S_j).

AN EXAMPLE: REPRESENTATIVE SUBSETS

- The aggregate states S_j are disjoint "representative" subsets of original system states



- Common case: Each S_j is a group of states with "similar characteristics"

- Compute a "cost" r_j for each aggregate state S_j (using some method)

- Approximate the optimal cost of each original system state x with $\sum_{j=1}^{m} \phi_{xj} r_j$

- For each x, the ϕ_{xj}, $j = 1, \ldots, m$, are the "aggregation probabilities" ... roughly the degrees of membership of state x in the aggregate states S_j

- Each ϕ_{xj} is prespecified and can be viewed as the jth feature of state x

6.231 DYNAMIC PROGRAMMING

LECTURE 9

LECTURE OUTLINE

- Rollout algorithms
- Policy improvement property
- Discrete deterministic problems
- Approximations of rollout algorithms
- Model Predictive Control (MPC)
- Discretization of continuous time
- Discretization of continuous space
- Other suboptimal approaches

ROLLOUT ALGORITHMS

- One-step lookahead policy: At each k and state x_k, use the control $\overline{\mu}_k(x_k)$ that

$$\min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\},$$

where
  - $\tilde{J}_N = g_N$.
  - $\tilde{J}_{k+1}$: approximation to true cost-to-go J_{k+1}
- Rollout algorithm: When $\tilde{J}_k$ is the cost-to-go of some heuristic policy (called the base policy)
- Policy improvement property (to be shown): The rollout algorithm achieves no worse (and usually much better) cost than the base heuristic starting from the same state.
- Main difficulty: Calculating $\tilde{J}_k(x_k)$ may be computationally intensive if the cost-to-go of the base policy cannot be analytically calculated.
  - May involve Monte Carlo simulation if the problem is stochastic.
  - Things improve in the deterministic case.

EXAMPLE: THE QUIZ PROBLEM

- A person is given N questions; answering correctly question i has probability p_i, reward v_i. Quiz terminates at the first incorrect answer.

- Problem: Choose the ordering of questions so as to maximize the total expected reward.

- Assuming no other constraints, it is optimal to use the index policy: Answer questions in decreasing order of $p_i v_i/(1-p_i)$.

- With minor changes in the problem, the index policy need not be optimal. Examples:
  - A limit ($< N$) on the maximum number of questions that can be answered.
  - Time windows, sequence-dependent rewards, precedence constraints.

- Rollout with the index policy as base policy: Convenient because at a given state (subset of questions already answered), the index policy and its expected reward can be easily calculated.

- Very effective for solving the quiz problem and important generalizations in scheduling (see Bertsekas and Castanon, J. of Heuristics, Vol. 5, 1999).

COST IMPROVEMENT PROPERTY

- Let

$\overline{J}_k(x_k)$: Cost-to-go of the rollout policy

$H_k(x_k)$: Cost-to-go of the base policy

- We claim that $\overline{J}_k(x_k) \leq H_k(x_k)$ for all x_k, k
- Proof by induction: We have $\overline{J}_N(x_N) = H_N(x_N)$ for all x_N. Assume that

$$\overline{J}_{k+1}(x_{k+1}) \leq H_{k+1}(x_{k+1}), \quad \forall\ x_{k+1}.$$

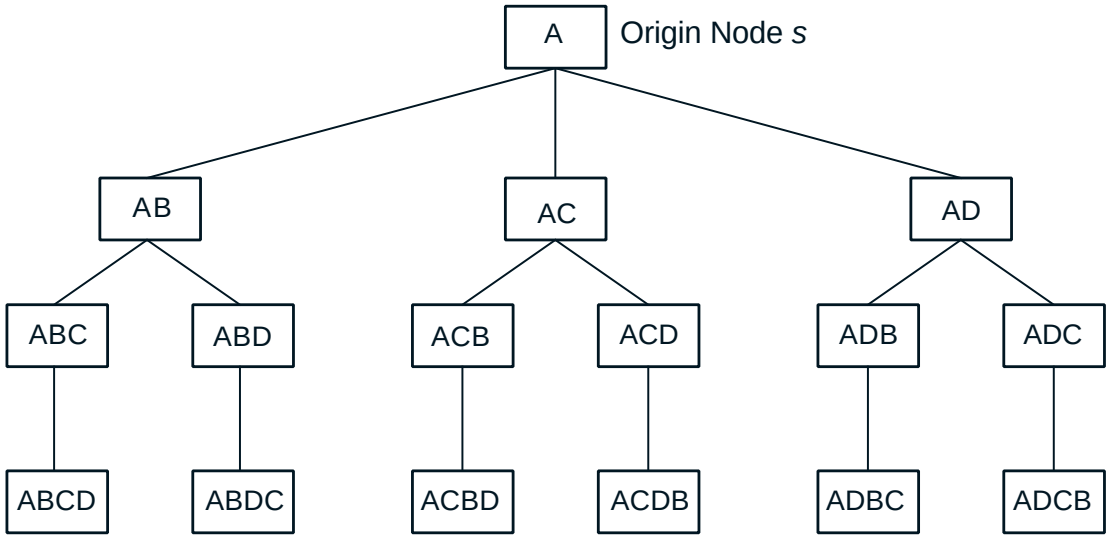
Let $\overline{\mu}_k(x_k)$ and $\mu_k(x_k)$ be the controls applied by rollout and heuristic at x_k. Then, for all x_k

$$\begin{aligned}
\overline{J}_k(x_k) &= E\Big\{g_k\Big(x_k, \overline{\mu}_k(x_k), w_k\Big) + \overline{J}_{k+1}\Big(f_k\Big(x_k, \overline{\mu}_k(x_k), w_k\Big)\Big)\Big\} \\
&\leq E\Big\{g_k\Big(x_k, \overline{\mu}_k(x_k), w_k\Big) + H_{k+1}\Big(f_k\Big(x_k, \overline{\mu}_k(x_k), w_k\Big)\Big)\Big\} \\
&\leq E\Big\{g_k\Big(x_k, \mu_k(x_k), w_k\Big) + H_{k+1}\Big(f_k\Big(x_k, \mu_k(x_k), w_k\Big)\Big)\Big\} \\
&= H_k(x_k)
\end{aligned}$$

  - Induction hypothesis ==> 1st inequality
  - Min selection of $\overline{\mu}_k(x_k)$ ==> 2nd inequality
  - Definition of H_k, μ_k ==> last equality

DISCRETE DETERMINISTIC PROBLEMS

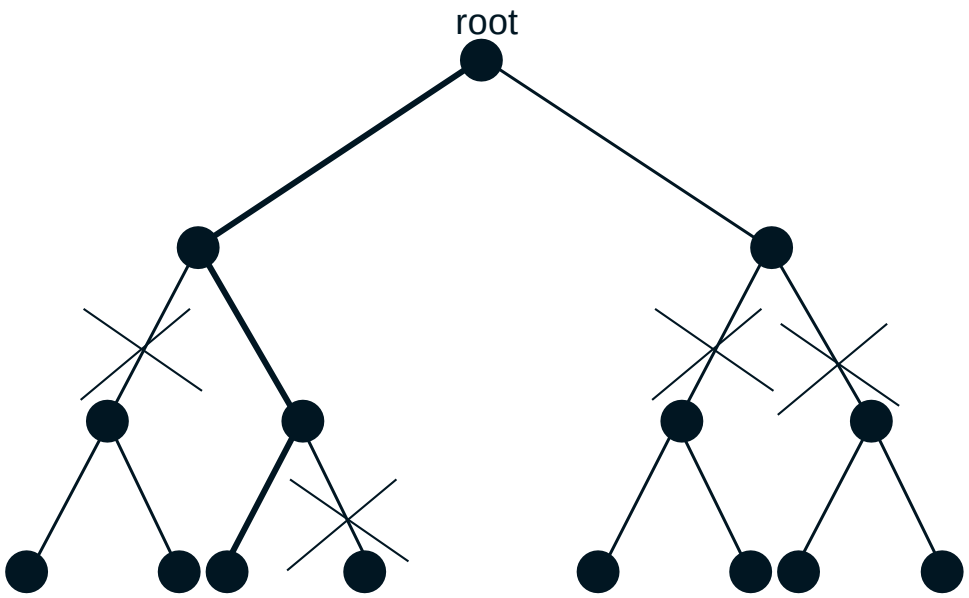
- Any discrete optimization problem can be represented sequentially by breaking down the decision process into stages.
- A tree/shortest path representation. The leaves of the tree correspond to the feasible solutions.
- Example: Traveling salesman problem. Find a minimum cost tour through N cities.



Traveling salesman problem with four cities A, B, C, D

- Complete partial solutions, one stage at a time
- May apply rollout with any heuristic that can complete a partial solution
- No costly stochastic simulation needed

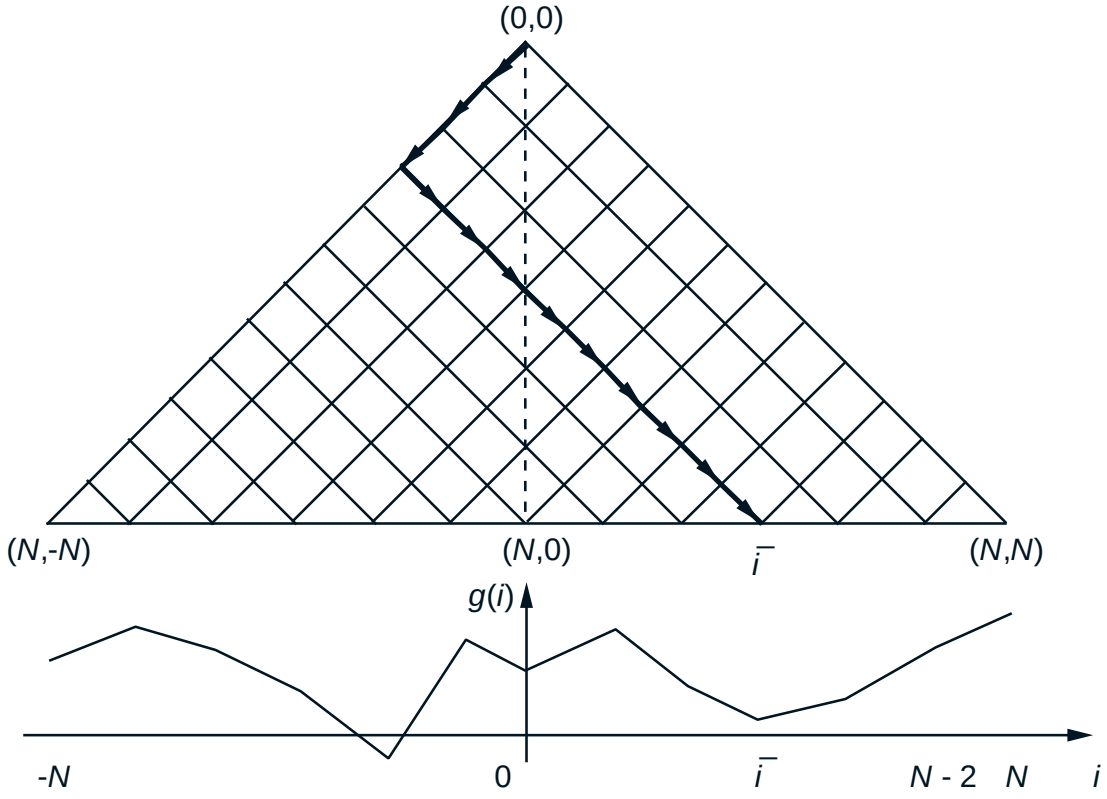
EXAMPLE: THE BREAKTHROUGH PROBLEM



- Given a binary tree with N stages.
- Each arc is free or is blocked (crossed out)
- Problem: Find a free path from the root to the leaves (such as the one shown with thick lines).
- Base heuristic (greedy): Follow the right branch if free; else follow the left branch if free.
- This is a rare rollout instance that admits a detailed analysis.
- For large N and given prob. of free branch: the rollout algorithm requires $O(N)$ times more computation, but has $O(N)$ times larger prob. of finding a free path than the greedy algorithm.

DET. EXAMPLE: ONE-DIMENSIONAL WALK

- A person takes either a unit step to the left or a unit step to the right. Minimize the cost $g(i)$ of the point i where he will end up after N steps.



- Base heuristic: Always go to the right. Rollout finds the rightmost local minimum.

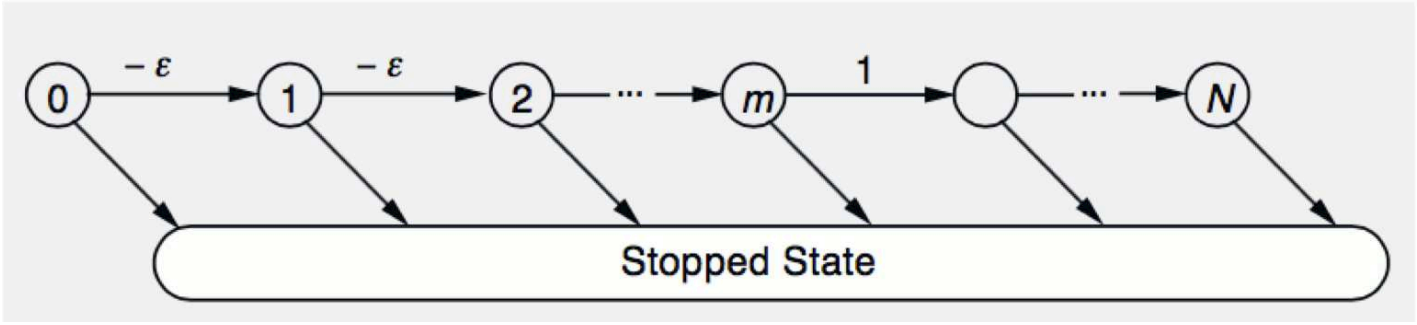
- Alternative base heuristic: Compare always go to the right and always go the left. Choose the best of the two. Rollout finds a global minimum.

A ROLLOUT ISSUE FOR DISCRETE PROBLEMS

- The base heuristic need not constitute a policy in the DP sense.
- Reason: Depending on its starting point, the base heuristic may not apply the same control at the same state.
- As a result the cost improvement property may be lost (except if the base heuristic has a property called sequential consistency; see the text for a formal definition).
- The cost improvement property is restored in two ways:
  - The base heuristic has a property called sequential improvement which guarantees cost reduction at each step (see the text for a formal definition).
  - A variant of the rollout algorithm, called fortified rollout, is used, which enforces cost improvement. Roughly speaking the "best" solution found so far is maintained, and it is followed whenever at any time the standard version of the algorithm tries to follow a "worse" solution (see the text).

ROLLING HORIZON WITH ROLLOUT

- We can use a rolling horizon approximation in calculating the cost-to-go of the base heuristic.
- Because the heuristic is suboptimal, the rationale for a long rolling horizon becomes weaker.
- Example: N-stage stopping problem where the stopping cost is 0, the continuation cost is either $-\epsilon$ or 1, where $0 < \epsilon << 1$, and the first state with continuation cost equal to 1 is state m. Then the optimal policy is to stop at state m, and the optimal cost is $-m\epsilon$.



- Consider the heuristic that continues at every state, and the rollout policy that is based on this heuristic, with a rolling horizon of $\ell \leq m$ steps.
- It will continue up to the first $m - \ell + 1$ stages, thus compiling a cost of $-(m-\ell+1)\epsilon$. The rollout performance improves as l becomes shorter!
- Limited vision may work to our advantage!

MODEL PREDICTIVE CONTROL (MPC)

- Special case of rollout for linear deterministic systems (similar extensions to nonlinear/stochastic)
  - System: $x_{k+1} = Ax_k + Bu_k$
  - Quadratic cost per stage: $x_k'Qx_k + u_k'Ru_k$
  - Constraints: $x_k \in X$, $u_k \in U(x_k)$

- Assumption: For any $x_0 \in X$ there is a feasible state-control sequence that brings the system to 0 in m steps, i.e., $x_m = 0$

- MPC at state x_k solves an m-step optimal control problem with constraint $x_{k+m} = 0$, i.e., finds a sequence $\bar{u}_k, \ldots, \bar{u}_{k+m-1}$ that minimizes

$$\sum_{\ell=0}^{m-1} \left(x_{k+\ell}'Qx_{k+\ell} + u_{k+\ell}'Ru_{k+\ell}\right)$$

subject to $x_{k+m} = 0$

- Then applies the first control $\bar{u}_k$ (and repeats at the next state x_{k+1})

- MPC is rollout with heuristic derived from the corresponding $m-1$-step optimal control problem

- Key Property of MPC: Since the heuristic is stable, the rollout is also stable (suggested by policy improvement property; see the text).

DISCRETIZATION

- If the time, and/or state space, and/or control space are continuous, they must be discretized.

- Consistency issue, i.e., as the discretization becomes finer, the cost-to-go functions of the discretized problem should converge to those of the original problem.

- Pitfall with discretizing continuous time: The control constraint set may change a lot as we pass to the discrete-time approximation.

- Example: Consider the system $\dot{x}(t) = u(t)$, with control constraint $u(t) \in \{-1, 1\}$. The reachable states after time δ are $x(t + \delta) = x(t) + u$, with $u \in [-\delta, \delta]$.

- Compare it with the reachable states after we discretize the system naively: $x(t+\delta) = x(t)+\delta u(t)$, with $u(t) \in \{-1, 1\}$.

- "Convexification effect" of continuous time: a discrete control constraint set in continuous-time differential systems, is equivalent to a continuous control constraint set when the system is looked at discrete times.

SPACE DISCRETIZATION

- Given a discrete-time system with state space S, consider a finite subset $\overline{S}$; for example $\overline{S}$ could be a finite grid within a continuous state space S.
- Difficulty: $f(x,u,w) \notin \overline{S}$ for $x \in \overline{S}$.
- We define an approximation to the original problem, with state space $\overline{S}$, as follows:
- Express each $x \in S$ as a convex combination of states in $\overline{S}$, i.e.,

$$x = \sum_{x_i \in \overline{S}} \phi_i(x) x_i \quad \text{where } \phi_i(x) \geq 0, \ \sum_i \phi_i(x) = 1$$

- Define a "reduced" dynamic system with state space $\overline{S}$, whereby from each $x_i \in \overline{S}$ we move to $\overline{x} = f(x_i,u,w)$ according to the system equation of the original problem, and then move to $x_j \in \overline{S}$ with probabilities $\phi_j(\overline{x})$.
- Define similarly the corresponding cost per stage of the transitions of the reduced system.
- Note application to finite-state POMDP (discretization of the simplex of the belief states).

SPACE DISCRETIZATION/AGGREGATION

- Let $\overline{J}_k(x_i)$ be the optimal cost-to-go of the "reduced" problem from each state $x_i \in \overline{S}$ and time k onward.

- Approximate the optimal cost-to-go of any $x \in S$ for the original problem by

$$\tilde{J}_k(x) = \sum_{x_i \in \overline{S}} \phi_i(x) \overline{J}_k(x_i),$$

and use one-step-lookahead based on $\tilde{J}_k$.

- The coefficients $\phi_i(x)$ can be viewed as features in an aggregation scheme.

- Important question: Consistency, i.e., as the number of states in $\overline{S}$ increases, $\tilde{J}_k(x)$ should converge to the optimal cost-to-go of the original prob.

- Interesting observation: While the original problem may be deterministic, the reduced problem is always stochastic.

- Generalization: The set $\overline{S}$ may be any finite set (not a subset of S) as long as the coefficients $\phi_i(x)$ admit a meaningful interpretation that quantifies the degree of association of x with x_i (a form of aggregation).

OTHER SUBOPTIMAL APPROACHES

- Minimize the DP equation error (Fitted Value Iteration): Approximate $J_k(x_k)$ with $\tilde{J}_k(x_k, r_k)$, where r_k is a parameter vector, chosen to minimize some form of error in the DP equations
  - Can be done sequentially going backwards in time (approximate J_k using an approximation of J_{k+1}, starting with $\tilde{J}_N = g_N$).
- Direct approximation of control policies: For a subset of states x^i, $i = 1, \ldots, m$, find

$$\hat{\mu}_k(x^i) = \arg \min_{u_k \in U_k(x^i)} E\Big\{ g(x^i, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x^i, u_k, w_k), r_{k+1}\big) \Big\}$$

Then find $\tilde{\mu}_k(x_k, s_k)$, where s_k is a vector of parameters obtained by solving the problem

$$\min_s \sum_{i=1}^{m} \|\hat{\mu}_k(x^i) - \tilde{\mu}_k(x^i, s)\|^2$$

- Approximation in policy space: Do not bother with cost-to-go approximations. Parametrize the policies as $\tilde{\mu}_k(x_k, s_k)$, and minimize the cost function of the problem over the parameters s_k (random search is a possibility).

6.231 DYNAMIC PROGRAMMING

LECTURE 10

LECTURE OUTLINE

- Infinite horizon problems
- Stochastic shortest path (SSP) problems
- Bellman's equation
- Dynamic programming – value iteration
- Discounted problems as special case of SSP

TYPES OF INFINITE HORIZON PROBLEMS

- Same as the basic problem, but:
  - The number of stages is infinite.
  - Stationary system and cost (except for discounting).
- Total cost problems: Minimize

$$J_\pi(x_0) = \lim_{N\to\infty} \underset{\substack{w_k \\ k=0,1,\dots}}{E} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

(if the lim exists - otherwise lim sup).
  - Stochastic shortest path (SSP) problems ($\alpha = 1$, and a termination state)
  - Discounted problems ($\alpha < 1$, bounded g)
  - Undiscounted, and discounted problems with unbounded g
- Average cost problems

$$\lim_{N\to\infty} \frac{1}{N} \underset{\substack{w_k \\ k=0,1,\dots}}{E} \left\{ \sum_{k=0}^{N-1} g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

- Infinite horizon characteristics: Challenging analysis, elegance of solutions and algorithms (stationary optimal policies are likely)

PREVIEW OF INFINITE HORIZON RESULTS

- Key issue: The relation between the infinite and finite horizon optimal cost-to-go functions.
- For example, let $\alpha = 1$ and $J_N(x)$ denote the optimal cost of the N-stage problem, generated after N DP iterations, starting from some J_0

$$J_{k+1}(x) = \min_{u \in U(x)} E_w \left\{ g(x,u,w) + J_k\big(f(x,u,w)\big) \right\}, \ \forall\, x$$

- Typical results for total cost problems:
  - Convergence of value iteration to J^*:

$$J^*(x) = \min_\pi J_\pi(x) = \lim_{N \to \infty} J_N(x), \quad \forall\ x$$

  - Bellman's equation holds for all x:

$$J^*(x) = \min_{u \in U(x)} E_w \left\{ g(x,u,w) + J^*\big(f(x,u,w)\big) \right\}$$

  - Optimality condition: If $\mu(x)$ minimizes in Bellman's Eq., $\{\mu, \mu, \ldots\}$ is optimal.
- Bellman's Eq. holds for all deterministic problems and "almost all" stochastic problems.
- Other results: True for SSP and discounted; exceptions for other problems.

"EASY" AND "DIFFICULT" PROBLEMS

- Easy problems (Chapter 7, Vol. I of text)
  - All of them are finite-state, finite-control
  - Bellman's equation has unique solution
  - Optimal policies obtained from Bellman Eq.
  - Value and policy iteration algorithms apply
- Somewhat complicated problems
  - Infinite state, discounted, bounded g (contractive structure)
  - Finite-state SSP with "nearly" contractive structure
  - Bellman's equation has unique solution, value and policy iteration work
- Difficult problems (w/ additional structure)
  - Infinite state, $g \geq 0$ or $g \leq 0$ (for all x, u, w)
  - Infinite state deterministic problems
  - SSP without contractive structure
- Hugely large and/or model-free problems
  - Big state space and/or simulation model
  - Approximate DP methods
- Measure theoretic formulations (not in this course)

STOCHASTIC SHORTEST PATH PROBLEMS

- Assume finite-state system: States $1, \ldots, n$ and special cost-free termination state t
  - Transition probabilities $p_{ij}(u)$
  - Control constraints $u \in U(i)$ (finite set)
  - Cost of policy $\pi = \{\mu_0, \mu_1, \ldots\}$ is

$$J_\pi(i) = \lim_{N \to \infty} E\left\{ \sum_{k=0}^{N-1} g\big(x_k, \mu_k(x_k)\big) \,\Big|\, x_0 = i \right\}$$

  - Optimal policy if $J_\pi(i) = J^*(i)$ for all i.
  - Special notation: For stationary policies $\pi = \{\mu, \mu, \ldots\}$, we use $J_\mu(i)$ in place of $J_\pi(i)$.

- Assumption (termination inevitable): There exists integer m such that for all policies π:

$$\rho_\pi = \max_{i=1,\ldots,n} P\{x_m \neq t \mid x_0 = i, \pi\} < 1$$

- Note: We have $\rho = \max_\pi \rho_\pi < 1$, since ρ_π depends only on the first m components of π.

- Shortest path examples: Acyclic (assumption is satisfied); nonacyclic (assumption is not satisfied)

FINITENESS OF POLICY COST FUNCTIONS

- View

$$\rho = \max_{\pi} \rho_\pi < 1$$

as an upper bound on the non-termination prob. during 1st m steps, regardless of policy used

- For any π and any initial state i

$$P\{x_{2m} \neq t \mid x_0 = i, \pi\} = P\{x_{2m} \neq t \mid x_m \neq t,\, x_0 = i, \pi\} \times P\{x_m \neq t \mid x_0 = i, \pi\} \leq \rho^2$$

and similarly

$$P\{x_{km} \neq t \mid x_0 = i, \pi\} \leq \rho^k, \qquad i = 1, \ldots, n$$

- So $E\{$Cost between times km and $(k+1)m - 1$ $\}$

$$\leq m\rho^k \max_{\substack{i=1,\ldots,n \\ u \in U(i)}} \left|g(i,u)\right|$$

and

$$\left|J_\pi(i)\right| \leq \sum_{k=0}^{\infty} m\rho^k \max_{\substack{i=1,\ldots,n \\ u \in U(i)}} \left|g(i,u)\right| = \frac{m}{1-\rho} \max_{\substack{i=1,\ldots,n \\ u \in U(i)}} \left|g(i,u)\right|$$

MAIN RESULT

- Given any initial conditions $J_0(1), \ldots, J_0(n)$, the sequence $J_k(i)$ generated by value iteration,

$$J_{k+1}(i) = \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J_k(j) \right], \ \forall \ i$$

converges to the optimal cost $J^*(i)$ for each i.

- Bellman's equation has $J^*(i)$ as unique solution:

$$J^*(i) = \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J^*(j) \right], \ \forall \ i$$

$$J^*(t) = 0$$

- A stationary policy μ is optimal if and only if for every state i, $\mu(i)$ attains the minimum in Bellman's equation.

- Key proof idea: The "tail" of the cost series,

$$\sum_{k=mK}^{\infty} E\left\{ g\big(x_k, \mu_k(x_k)\big) \right\}$$

vanishes as K increases to ∞.

OUTLINE OF PROOF THAT $J_N \to J^*$

- Assume for simplicity that $J_0(i) = 0$ for all i. For any $K \geq 1$, write the cost of any policy π as

$$J_\pi(x_0) = \sum_{k=0}^{mK-1} E\left\{g\Big(x_k, \mu_k(x_k)\Big)\right\} + \sum_{k=mK}^{\infty} E\left\{g\Big(x_k, \mu_k(x_k)\Big)\right\}$$

$$\leq \sum_{k=0}^{mK-1} E\left\{g\Big(x_k, \mu_k(x_k)\Big)\right\} + \sum_{k=K}^{\infty} \rho^k m \max_{i,u} |g(i,u)|$$

Take the minimum of both sides over π to obtain

$$J^*(x_0) \leq J_{mK}(x_0) + \frac{\rho^K}{1-\rho} m \max_{i,u} |g(i,u)|.$$

Similarly, we have

$$J_{mK}(x_0) - \frac{\rho^K}{1-\rho} m \max_{i,u} |g(i,u)| \leq J^*(x_0).$$

It follows that $\lim_{K\to\infty} J_{mK}(x_0) = J^*(x_0)$.

- $J_{mK}(x_0)$ and $J_{mK+k}(x_0)$ converge to the same limit for $k < m$ (since k extra steps far into the future don't matter), so $J_N(x_0) \to J^*(x_0)$.

- Similarly, $J_0 \neq 0$ does not matter.

EXAMPLE

- Minimizing the $E\{\text{Time to Termination}\}$: Let

$$g(i,u) = 1, \qquad \forall\ i = 1,\ldots,n, \quad u \in U(i)$$

- Under our assumptions, the costs $J^*(i)$ uniquely solve Bellman's equation, which has the form

$$J^*(i) = \min_{u \in U(i)} \left[1 + \sum_{j=1}^{n} p_{ij}(u) J^*(j) \right], \quad i = 1,\ldots,n$$

- In the special case where there is only one control at each state, $J^*(i)$ is the mean first passage time from i to t. These times, denoted m_i, are the unique solution of the classical equations

$$m_i = 1 + \sum_{j=1}^{n} p_{ij} m_j, \qquad i = 1,\ldots,n,$$

which are seen to be a form of Bellman's equation

6.231 DYNAMIC PROGRAMMING

LECTURE 11

LECTURE OUTLINE

- Review of stochastic shortest path problems
- Computational methods for SSP
  - Value iteration
  - Policy iteration
  - Linear programming
- Computational methods for discounted problems

STOCHASTIC SHORTEST PATH PROBLEMS

- Assume finite-state system: States $1, \ldots, n$ and special cost-free termination state t
  - Transition probabilities $p_{ij}(u)$
  - Control constraints $u \in U(i)$ (finite set)
  - Cost of policy $\pi = \{\mu_0, \mu_1, \ldots\}$ is

$$J_\pi(i) = \lim_{N \to \infty} E \left\{ \sum_{k=0}^{N-1} g\big(x_k, \mu_k(x_k)\big) \,\Big|\, x_0 = i \right\}$$

  - Optimal policy if $J_\pi(i) = J^*(i)$ for all i.
  - Special notation: For stationary policies $\pi = \{\mu, \mu, \ldots\}$, we use $J_\mu(i)$ in place of $J_\pi(i)$.

- Assumption (Termination inevitable): There exists integer m such that for every policy and initial state, there is positive probability that the termination state will be reached after no more that m stages; for all π, we have

$$\rho_\pi = \max_{i=1,\ldots,n} P\{x_m \neq t \mid x_0 = i, \pi\} < 1$$

MAIN RESULT

- Given any initial conditions $J_0(1), \ldots, J_0(n)$, the sequence $J_k(i)$ generated by value iteration

$$J_{k+1}(i) = \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J_k(j) \right], \ \forall \ i$$

converges to the optimal cost $J^*(i)$ for each i.

- Bellman's equation has $J^*(i)$ as unique solution:

$$J^*(i) = \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J^*(j) \right], \ \forall \ i$$

- For a stationary policy μ, $J_\mu(i)$, $i = 1, \ldots, n$, are the unique solution of the linear system of n equations

$$J_\mu(i) = g\big(i, \mu(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big) J_\mu(j), \ \ \forall \ i = 1, \ldots, n$$

- A stationary policy μ is optimal if and only if for every state i, $\mu(i)$ attains the minimum in Bellman's equation.

BELLMAN'S EQ. FOR A SINGLE POLICY

- Consider a stationary policy μ
- $J_\mu(i)$, $i = 1, \ldots, n$, are the unique solution of the linear system of n equations

$$J_\mu(i) = g\big(i, \mu(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big) J_\mu(j), \quad \forall\, i = 1, \ldots, n$$

- The equation provides a way to compute $J_\mu(i)$, $i = 1, \ldots, n$, but the computation is substantial for large n $[O(n^3)]$
- For large n, value iteration may be preferable. (Typical case of a large linear system of equations, where an iterative method may be better than a direct solution method.)
- For VERY large n, exact methods cannot be applied, and approximations are needed. (We will discuss these later.)

POLICY ITERATION

- It generates a sequence $\mu^1, \mu^2, \ldots$ of stationary policies, starting with any stationary policy μ^0.

- At the typical iteration, given μ^k, we perform a policy evaluation step, that computes the $J_{\mu^k}(i)$ as the solution of the (linear) system of equations

$$J(i) = g\big(i, \mu^k(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big) J(j), \quad i = 1, \ldots, n,$$

in the n unknowns $J(1), \ldots, J(n)$. We then perform a policy improvement step,

$$\mu^{k+1}(i) = \arg \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J_{\mu^k}(j) \right], \ \forall \ i$$

- Terminate when $J_{\mu^k}(i) = J_{\mu^{k+1}}(i)$ $\forall$ i. Then $J_{\mu^{k+1}} = J^*$ and μ^{k+1} is optimal, since

$$\begin{aligned} J_{\mu^{k+1}}(i) &= g(i, \mu^{k+1}(i)) + \sum_{j=1}^{n} p_{ij}(\mu^{k+1}(i)) J_{\mu^{k+1}}(j) \\ &= \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J_{\mu^{k+1}}(j) \right] \end{aligned}$$

JUSTIFICATION OF POLICY ITERATION

- We can show that $J_{\mu^k}(i) \geq J_{\mu^{k+1}}(i)$ for all i, k
- Fix k and consider the sequence generated by

$$J_{N+1}(i) = g\big(i, \mu^{k+1}(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu^{k+1}(i)\big) J_N(j)$$

where $J_0(i) = J_{\mu^k}(i)$. We have

$$J_0(i) = g\big(i, \mu^k(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big) J_0(j)$$

$$\geq g\big(i, \mu^{k+1}(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu^{k+1}(i)\big) J_0(j) = J_1(i)$$

- Using the monotonicity property of DP,

$$J_0(i) \geq J_1(i) \geq \cdots \geq J_N(i) \geq J_{N+1}(i) \geq \cdots, \qquad \forall\ i$$

Since $J_N(i) \to J_{\mu^{k+1}}(i)$ as $N \to \infty$, we obtain policy improvement, i.e.

$$J_{\mu^k}(i) = J_0(i) \geq J_{\mu^{k+1}}(i) \qquad \forall\ i, k$$

- A policy cannot be repeated (there are finitely many stationary policies), so the algorithm terminates with an optimal policy

LINEAR PROGRAMMING

- We claim that J^* is the "largest" J that satisfies the constraint

$$J(i) \leq g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J(j), \qquad (1)$$

for all $i = 1, \ldots, n$ and $u \in U(i)$.

- Proof: If we use value iteration to generate a sequence of vectors $J_k = \big(J_k(1), \ldots, J_k(n)\big)$ starting with a J_0 that satisfies the constraint, i.e.,

$$J_0(i) \leq \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J_0(j) \right], \quad \forall\ i$$

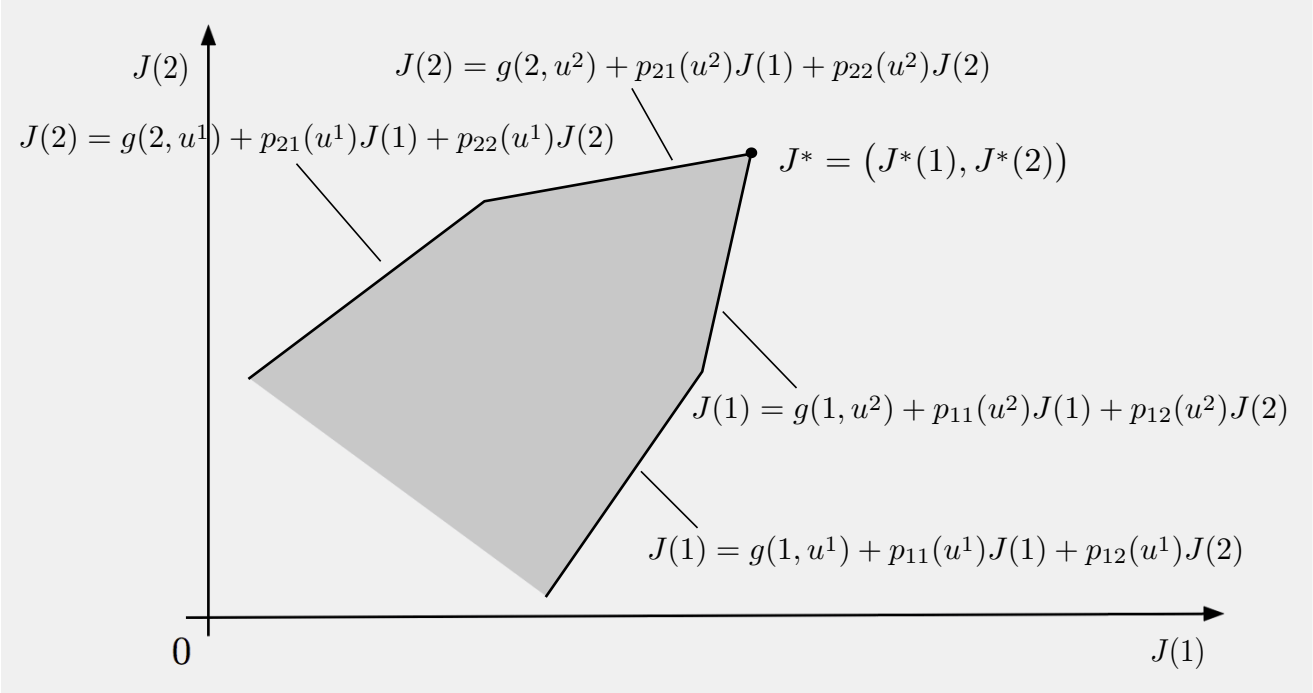
then, $J_k(i) \leq J_{k+1}(i)$ for all k and i (monotonicity property of DP) and $J_k \to J^*$, so that $J_0(i) \leq J^*(i)$ for all i.

- So $J^* = \big(J^*(1), \ldots, J^*(n)\big)$ is the solution of the linear program of maximizing $\sum_{i=1}^{n} J(i)$ subject to the constraint (1).

LINEAR PROGRAMMING (CONTINUED)

- Obtain J^* by Max $\sum_{i=1}^n J(i)$ subject to

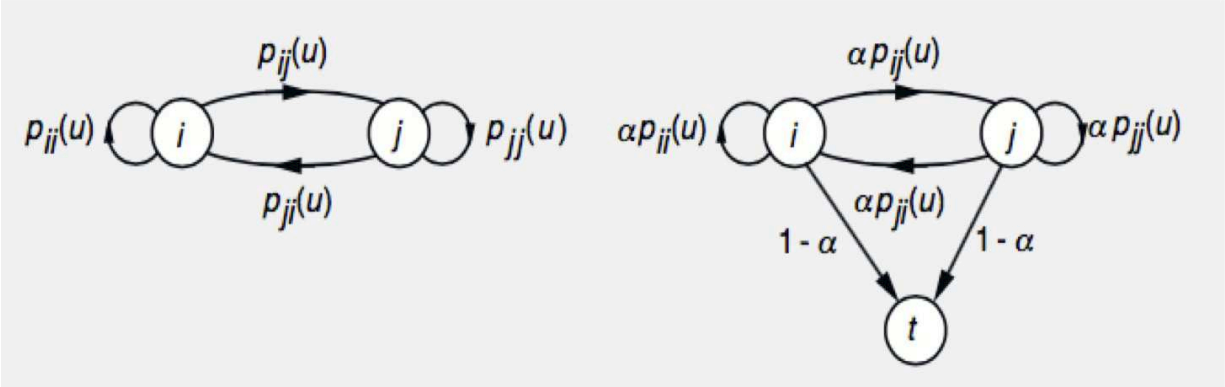
$$J(i) \leq g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J(j), \quad i = 1, \ldots, n, \ u \in U(i)$$



- Drawback: For large n the dimension of this program is very large. Furthermore, the number of constraints is equal to the number of state-control pairs.

DISCOUNTED PROBLEMS

- Assume a discount factor $\alpha < 1$.
- Conversion to an SSP problem.



- kth stage cost is the same for both problems
- Value iteration converges to J^* for all initial J_0:

$$J_{k+1}(i) = \min_{u \in U(i)} \left[g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J_k(j) \right], \ \forall \ i$$

- J^* is the unique solution of Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \left[g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J^*(j) \right], \ \forall \ i$$

- Policy iteration terminates with an optimal policy, and linear programming works.

DISCOUNTED PROBLEM EXAMPLE

- A manufacturer at each time:
  - Receives an order with prob. p and no order with prob. $1-p$.
  - May process all unfilled orders at cost $K > 0$, or process no order at all. The cost per unfilled order at each time is $c > 0$.
  - Maximum number of orders that can remain unfilled is n.
  - Find a processing policy that minimizes the α-discounted cost per stage.
  - State: Number of unfilled orders at the start of a period $(i = 0, 1, \ldots, n)$.
- Bellman's Eq.:

$$J^*(i) = \min \Big[K + \alpha(1-p)J^*(0) + \alpha p J^*(1),$$
$$ci + \alpha(1-p)J^*(i) + \alpha p J^*(i+1)\Big],$$

for the states $i = 0, 1, \ldots, n-1$, and

$$J^*(n) = K + \alpha(1-p)J^*(0) + \alpha p J^*(1)$$

for state n.

- Analysis: Argue that $J^*(i)$ is mon. increasing in i, to show that the optimal policy is a threshold policy.

6.231 DYNAMIC PROGRAMMING

LECTURE 12

LECTURE OUTLINE

- Average cost per stage problems
- Connection with stochastic shortest path problems
- Bellman's equation
- Value iteration
- Policy iteration

AVERAGE COST PER STAGE PROBLEM

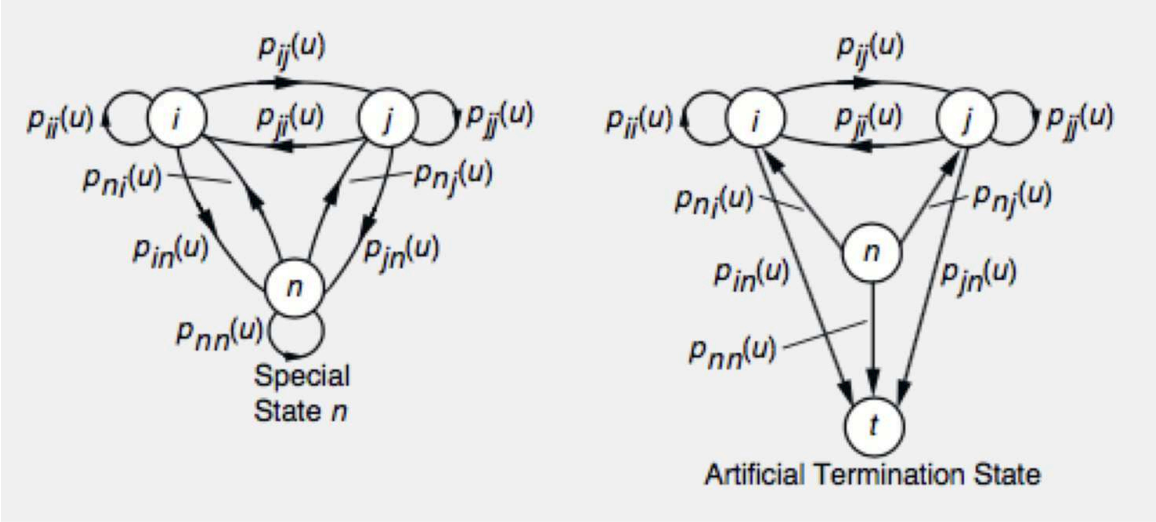
- Assume a stationary system with finite number of states and controls.
- Minimize over policies $\pi = \{\mu_0, \mu_1, ...\}$

$$J_\pi(x_0) = \lim_{N\to\infty} \frac{1}{N} \underset{\substack{w_k \\ k=0,1,...}}{E} \left\{ \sum_{k=0}^{N-1} g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

- Important characteristics (not shared by other types of infinite horizon problems).
  - For any fixed T, the cost incurred up to time T does not matter (only the state that we are at time T matters)
  - If all states "communicate" the optimal cost is independent of initial state [if we can go from i to j in finite expected time, we must have $J^*(i) \leq J^*(j)$]. So $J^*(i) \equiv \lambda^*$ for all i.
  - Because "communication" issues are so important, the methodology relies heavily on Markov chain theory.
  - The theory depends a lot on whether the chains corresponding to policies have a single or multiple recurrent classes. We will focus on the simplest version, using SSP theory.

CONNECTION WITH SSP

- Assumption: State n is special, in that for all initial states and all policies, n will be visited infinitely often (with probability 1).
- Then we expect that $J^*(i) \equiv$ some λ^*
- Divide the sequence of generated states into cycles marked by successive visits to n.
- Let's focus on a single cycle: It can be viewed as a state trajectory of an SSP problem with n as the termination state.



- Let the cost at i of the SSP be $g(i,u) - \lambda^*$
- We will argue (informally) that

Av. Cost Probl. $\equiv$ A Min Cost Cycle Probl. $\equiv$ SSP Probl.

CONNECTION WITH SSP (CONTINUED)

- Consider a minimum cycle cost problem: Find a stationary policy μ that minimizes the expected cost per transition within a cycle

$$\frac{C_{nn}(\mu)}{N_{nn}(\mu)},$$

where for a fixed μ,

$C_{nn}(\mu)$: $E\{$cost from n up to the first return to $n\}$

$N_{nn}(\mu)$: $E\{$time from n up to the first return to $n\}$

- Intuitively, $C_{nn}(\mu)/N_{nn}(\mu)$ = average cost of μ, and optimal cycle cost = λ^*, so

$$C_{nn}(\mu) - N_{nn}(\mu)\lambda^* \geq 0,$$

with equality if μ is optimal.

- Consider SSP with stage costs $g(i,u) - \lambda^*$. The cost of μ starting from n is $C_{nn}(\mu) - N_{nn}(\mu)\lambda^*$, so the optimal/min cycle μ is also optimal for the SSP.

- Also: Optimal SSP cost starting from $n = 0$.

BELLMAN'S EQUATION

- Let $h^*(i)$ the optimal cost of this SSP problem when starting at the nontermination states $i = 1, \ldots, n$. Then $h^*(1), \ldots, h^*(n)$ solve uniquely the corresponding Bellman's equation

$$h^*(i) = \min_{u \in U(i)} \left[g(i,u) - \lambda^* + \sum_{j=1}^{n-1} p_{ij}(u) h^*(j) \right], \ \forall \, i$$

- If μ^* is an optimal stationary policy for the SSP problem, we have

$$h^*(n) = C_{nn}(\mu^*) - N_{nn}(\mu^*)\lambda^* = 0$$

- Combining these equations, we have

$$\lambda^* + h^*(i) = \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) h^*(j) \right], \ \forall \, i$$

$$h^*(n) = 0$$

- If $\mu^*(i)$ attains the min for each i, μ^* is optimal.
- There is also Bellman Eq. for a single policy μ.

MORE ON THE CONNECTION WITH SSP

- Interpretation of $h^*(i)$ as a relative or differential cost: It is the minimum of

$E\{\text{cost to reach } n \text{ from } i \text{ for the first time}\}$

$-\ E\{\text{cost if the stage cost were } \lambda^* \text{ and not } g(i,u)\}$

- Algorithms: We don't know λ^*, so we can't solve the average cost problem as an SSP problem. But similar value and policy iteration algorithms are possible, and will be given shortly.

- Example: A manufacturer at each time
  - Receives an order with prob. p and no order with prob. $1-p$.
  - May process all unfilled orders at cost $K > 0$, or process no order at all. The cost per unfilled order at each time is $c > 0$.
  - Maximum number of orders that can remain unfilled is n.
  - Find a processing policy that minimizes the total expected cost per stage.

EXAMPLE (CONTINUED)

- State = number of unfilled orders. State 0 is the special state for the SSP formulation.
- Bellman's equation: For states $i = 0, 1, \ldots, n-1$

$$\lambda^* + h^*(i) = \min \Big[K + (1-p)h^*(0) + ph^*(1), \\ ci + (1-p)h^*(i) + ph^*(i+1)\Big],$$

and for state n

$$\lambda^* + h^*(n) = K + (1-p)h^*(0) + ph^*(1)$$

Also $h^*(0) = 0$.

- Optimal policy: Process i unfilled orders if

$$K+(1-p)h^*(0)+ph^*(1) \leq ci+(1-p)h^*(i)+ph^*(i+1)$$

- Intuitively, $h^*(i)$ is monotonically nondecreasing with i (interpret $h^*(i)$ as optimal costs-to-go for the associate SSP problem). So a threshold policy is optimal: process the orders if their number exceeds some threshold integer m^*.

VALUE ITERATION

- Natural VI method: Generate optimal k-stage costs by DP algorithm starting with any J_0:

$$J_{k+1}(i) = \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) J_k(j) \right], \ \forall \ i$$

- Convergence: $\lim_{k \to \infty} J_k(i)/k = \lambda^*$ for all i.

- Proof outline: Let J_k^* be so generated starting from the opt. differential cost, i.e., the initial condition $J_0^* = h^*$. Then, by induction,

$$J_k^*(i) = k\lambda^* + h^*(i), \qquad \forall i, \ \forall \ k.$$

On the other hand,

$$\left| J_k(i) - J_k^*(i) \right| \leq \max_{j=1,\ldots,n} \left| J_0(j) - h^*(j) \right|, \qquad \forall \ i$$

since $J_k(i)$ and $J_k^*(i)$ are optimal costs for two k-stage problems that differ only in the terminal cost functions, which are J_0 and h^*.

RELATIVE VALUE ITERATION

- The VI method just described has two drawbacks:
  - Since typically some components of J_k diverge to ∞ or $-\infty$, calculating $\lim_{k\to\infty} J_k(i)/k$ is numerically cumbersome.
  - The method will not compute a corresponding differential cost vector h^*.
- We can bypass both difficulties by subtracting a constant from all components of the vector J_k, so that the difference, call it h_k, remains bounded.
- Relative VI algorithm: Pick any state s, and iterate according to

$$h_{k+1}(i) = \min_{u\in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) h_k(j) \right] - \min_{u\in U(s)} \left[g(s,u) + \sum_{j=1}^{n} p_{sj}(u) h_k(j) \right], \quad \forall\ i$$

- Convergence: We can show $h_k \to h^*$ (under an extra assumption; see Vol. II).

POLICY ITERATION

- At iteration k, we have a stationary μ^k.
- Policy evaluation: Compute λ^k and $h^k(i)$ of μ^k, using the $n+1$ equations $h^k(n)=0$ and

$$\lambda^k + h^k(i) = g\big(i,\mu^k(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big)h^k(j), \ \forall\ i$$

- Policy improvement: (For the λ^k-SSP) Find

$$\mu^{k+1}(i) = \arg\min_{u\in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u)h^k(j) \right], \quad \forall\, i$$

- If $\lambda^{k+1}=\lambda^k$ and $h^{k+1}(i)=h^k(i)$ for all i, stop; otherwise, repeat with μ^{k+1} replacing μ^k.
- Result: For each k, we either have $\lambda^{k+1}<\lambda^k$ or we have policy improvement for the λ^k-SSP:

$$\lambda^{k+1}=\lambda^k, \qquad h^{k+1}(i)\le h^k(i), \quad i=1,\ldots,n.$$

The algorithm terminates with an optimal policy.

6.231 DYNAMIC PROGRAMMING

LECTURE 13

LECTURE OUTLINE

- Control of continuous-time Markov chains – Semi-Markov problems
- Problem formulation – Equivalence to discrete-time problems
- Discounted problems
- Average cost problems

CONTINUOUS-TIME MARKOV CHAINS

- Stationary system with finite number of states and controls
- State transitions occur at discrete times
- Control applied at these discrete times and stays constant between transitions
- Time between transitions is random
- Cost accumulates in continuous time (may also be incurred at the time of transition)
- Example: Admission control in a system with restricted capacity (e.g., a communication link)
  - Customer arrivals: a Poisson process
  - Customers entering the system, depart after exponentially distributed time
  - Upon arrival we must decide whether to admit or to block a customer
  - There is a cost for blocking a customer
  - For each customer that is in the system, there is a customer-dependent reward per unit time
  - Minimize time-discounted or average cost

PROBLEM FORMULATION

- $x(t)$ and $u(t)$: State and control at time t
- t_k: Time of kth transition ($t_0 = 0$)
- $x_k = x(t_k)$; $\quad x(t) = x_k$ for $t_k \leq t < t_{k+1}$.
- $u_k = u(t_k)$; $\quad u(t) = u_k$ for $t_k \leq t < t_{k+1}$.
- No transition probabilities; instead transition distributions (quantify the uncertainty about both transition time and next state)

$$Q_{ij}(\tau, u) = P\{t_{k+1} - t_k \leq \tau,\ x_{k+1} = j \mid x_k = i,\ u_k = u\}$$

- Two important formulas:

(1) Transition probabilities are specified by

$$p_{ij}(u) = P\{x_{k+1} = j \mid x_k = i,\ u_k = u\} = \lim_{\tau \to \infty} Q_{ij}(\tau, u)$$

(2) The Cumulative Distribution Function (CDF) of τ given i, j, u is (assuming $p_{ij}(u) > 0$)

$$P\{t_{k+1} - t_k \leq \tau \mid x_k = i,\ x_{k+1} = j,\ u_k = u\} = \frac{Q_{ij}(\tau, u)}{p_{ij}(u)}$$

Thus, $Q_{ij}(\tau, u)$ can be viewed as a "scaled CDF"

EXPONENTIAL TRANSITION DISTRIBUTIONS

- Important example of transition distributions:

$$Q_{ij}(\tau, u) = p_{ij}(u)\big(1 - e^{-\nu_i(u)\tau}\big),$$

where $p_{ij}(u)$ are transition probabilities, and $\nu_i(u)$ is called the transition rate at state i.

- Interpretation: If the system is in state i and control u is applied
  - the next state will be j with probability $p_{ij}(u)$
  - the time between the transition to state i and the transition to the next state j is exponentially distributed with parameter $\nu_i(u)$ (independently of j):

$$P\{\text{transition time interval } > \tau \mid i, u\} = e^{-\nu_i(u)\tau}$$

- The exponential distribution is memoryless. This implies that for a given policy, the system is a continuous-time Markov chain (the future depends on the past through the current state).

- Without the memoryless property, the Markov property holds only at the times of transition.

COST STRUCTURES

- There is cost $g(i,u)$ per unit time, i.e.

$$g(i,u)dt = \quad \text{the cost incurred in time } dt$$

- There may be an extra "instantaneous" cost $\hat{g}(i,u)$ at the time of a transition (let's ignore this for the moment)

- Total discounted cost of $\pi = \{\mu_0, \mu_1, \ldots\}$ starting from state i (with discount factor $\beta > 0$)

$$\lim_{N\to\infty} E\left\{\sum_{k=0}^{N-1}\int_{t_k}^{t_{k+1}} e^{-\beta t} g\big(x_k,\mu_k(x_k)\big)dt \;\Big|\; x_0 = i\right\}$$

- Average cost per unit time

$$\lim_{N\to\infty} \frac{1}{E\{t_N\}} E\left\{\sum_{k=0}^{N-1}\int_{t_k}^{t_{k+1}} g\big(x_k,\mu_k(x_k)\big)dt \;\Big|\; x_0 = i\right\}$$

- We will see that both problems have equivalent discrete-time versions.

DISCOUNTED CASE - COST CALCULATION

- For a policy $\pi = \{\mu_0, \mu_1, \ldots\}$, write

$$J_\pi(i) = E\{\text{1st transition cost}\} + E\{e^{-\beta\tau} J_{\pi_1}(j) \mid i, \mu_0(i)\}$$

where $E\{\text{1st transition cost}\} = E\left\{\int_0^\tau e^{-\beta t} g(i, \mu_0(i))dt\right\}$ and $J_{\pi_1}(j)$ is the cost-to-go of $\pi_1 = \{\mu_1, \mu_2, \ldots\}$

- We calculate the two costs in the RHS. The $E\{\text{1st transition cost}\}$, if u is applied at state i, is

$$\begin{aligned} G(i,u) &= E_j\Big\{E_\tau\{\text{1st transition cost} \mid j\}\Big\} \\ &= \sum_{j=1}^n p_{ij}(u) \int_0^\infty \left(\int_0^\tau e^{-\beta t} g(i,u)dt\right) \frac{dQ_{ij}(\tau,u)}{p_{ij}(u)} \\ &= g(i,u) \sum_{j=1}^n \int_0^\infty \frac{1-e^{-\beta\tau}}{\beta} dQ_{ij}(\tau,u) \end{aligned}$$

- Thus the $E\{\text{1st transition cost}\}$ is

$$G\big(i,\mu_0(i)\big) = g\big(i,\mu_0(i)\big) \sum_{j=1}^n \int_0^\infty \frac{1-e^{-\beta\tau}}{\beta} dQ_{ij}\big(\tau,\mu_0(i)\big)$$

(The summation term can be viewed as a "discounted length of the transition interval $t_1 - t_0$".)

COST CALCULATION (CONTINUED)

- Also the expected (discounted) cost from the next state j is

$$
\begin{aligned}
&E\big\{e^{-\beta\tau} J_{\pi_1}(j) \mid i, \mu_0(i)\big\} \\
&\quad = E_j\Big\{E\{e^{-\beta\tau} \mid i, \mu_0(i), j\} J_{\pi_1}(j) \mid i, \mu_0(i)\Big\} \\
&\quad = \sum_{j=1}^{n} p_{ij}\big(\mu_0(i)\big) \left(\int_0^\infty e^{-\beta\tau} \frac{dQ_{ij}(\tau, \mu_0(i))}{p_{ij}(\mu_0(i))} \right) J_{\pi_1}(j) \\
&\quad = \sum_{j=1}^{n} m_{ij}\big(\mu_0(i)\big) J_{\pi_1}(j)
\end{aligned}
$$

where $m_{ij}(u)$ is given by

$$
m_{ij}(u) = \int_0^\infty e^{-\beta\tau} dQ_{ij}(\tau, u) \quad \left(< \int_0^\infty dQ_{ij}(\tau, u) = p_{ij}(u) \right)
$$

and can be viewed as the "effective discount factor" [the analog of $\alpha p_{ij}(u)$ in discrete-time case].

- So $J_\pi(i)$ can be written as

$$
J_\pi(i) = G\big(i, \mu_0(i)\big) + \sum_{j=1}^{n} m_{ij}\big(\mu_0(i)\big) J_{\pi_1}(j)
$$

i.e., the (continuous-time discounted) cost of 1st period, plus the (continuous-time discounted) cost-to-go from the next state.

COST CALCULATION (CONTINUED)

- Also the expected (discounted) cost from the next state j is

$$\begin{aligned}
&E\big\{e^{-\beta\tau} J_{\pi_1}(j) \mid i, \mu_0(i)\big\} \\
&\quad = E_j\Big\{E\{e^{-\beta\tau} \mid i, \mu_0(i), j\} J_{\pi_1}(j) \mid i, \mu_0(i)\Big\} \\
&\quad = \sum_{j=1}^{n} p_{ij}\big(\mu_0(i)\big) \left(\int_0^\infty e^{-\beta\tau} \frac{dQ_{ij}(\tau, \mu_0(i))}{p_{ij}(\mu_0(i))}\right) J_{\pi_1}(j) \\
&\quad = \sum_{j=1}^{n} m_{ij}\big(\mu_0(i)\big) J_{\pi_1}(j)
\end{aligned}$$

where $m_{ij}(u)$ is given by

$$m_{ij}(u) = \int_0^\infty e^{-\beta\tau} dQ_{ij}(\tau, u) \quad \left(< \int_0^\infty dQ_{ij}(\tau, u) = p_{ij}(u)\right)$$

and can be viewed as the "effective discount factor" [the analog of $\alpha p_{ij}(u)$ in discrete-time case].

- So $J_\pi(i)$ can be written as

$$J_\pi(i) = G\big(i, \mu_0(i)\big) + \sum_{j=1}^{n} m_{ij}\big(\mu_0(i)\big) J_{\pi_1}(j)$$

i.e., the (continuous-time discounted) cost of 1st period, plus the (continuous-time discounted) cost-to-go from the next state.

EQUIVALENCE TO AN SSP

- Similar to the discrete-time case, introduce an "equivalent" stochastic shortest path problem with an artificial termination state t
- Under control u, from state i the system moves to state j with probability $m_{ij}(u)$ and to the termination state t with probability $1-\sum_{j=1}^{n} m_{ij}(u)$
- Bellman's equation: For $i=1,\ldots,n$,

$$J^*(i) = \min_{u\in U(i)} \left[G(i,u) + \sum_{j=1}^{n} m_{ij}(u) J^*(j) \right]$$

- Analogs of value iteration, policy iteration, and linear programming.
- If in addition to the cost per unit time g, there is an extra (instantaneous) one-stage cost $\hat{g}(i,u)$, Bellman's equation becomes

$$J^*(i) = \min_{u\in U(i)} \left[\hat{g}(i,u) + G(i,u) + \sum_{j=1}^{n} m_{ij}(u) J^*(j) \right]$$

MANUFACTURER'S EXAMPLE REVISITED

- A manufacturer receives orders with interarrival times uniformly distributed in $[0, \tau_{\max}]$.
- He may process all unfilled orders at cost $K > 0$, or process none. The cost per unit time of an unfilled order is c. Max number of unfilled orders is n.
- The nonzero transition distributions are

$$Q_{i1}(\tau, \text{Fill}) = Q_{i(i+1)}(\tau, \text{Not Fill}) = \min\left[1, \frac{\tau}{\tau_{\max}}\right]$$

- The one-stage expected cost G is

$$G(i, \text{Fill}) = 0, \qquad G(i, \text{Not Fill}) = \gamma\, c\, i,$$

where

$$\gamma = \sum_{j=1}^{n} \int_0^{\infty} \frac{1 - e^{-\beta\tau}}{\beta} dQ_{ij}(\tau, u) = \int_0^{\tau_{\max}} \frac{1 - e^{-\beta\tau}}{\beta\tau_{\max}} d\tau$$

- There is an "instantaneous" cost

$$\hat{g}(i, \text{Fill}) = K, \qquad \hat{g}(i, \text{Not Fill}) = 0$$

MANUFACTURER'S EXAMPLE CONTINUED

- The "effective discount factors" $m_{ij}(u)$ in Bellman's Equation are

$$m_{i1}(\text{Fill}) = m_{i(i+1)}(\text{Not Fill}) = \alpha,$$

where

$$\alpha = \int_0^\infty e^{-\beta\tau} dQ_{ij}(\tau, u) = \int_0^{\tau_{\max}} \frac{e^{-\beta\tau}}{\tau_{\max}} d\tau = \frac{1 - e^{-\beta\tau_{\max}}}{\beta\tau_{\max}}$$

- Bellman's equation has the form

$$J^*(i) = \min\big[K + \alpha J^*(1),\ \gamma c i + \alpha J^*(i+1)\big], \quad i = 1, 2, \ldots$$

- As in the discrete-time case, we can conclude that there exists an optimal threshold i^*:

fill the orders <==> their number i exceeds i^*

AVERAGE COST

- Minimize $\lim_{N\to\infty} \frac{1}{E\{t_N\}} E\left\{\int_0^{t_N} g\big(x(t),u(t)\big)dt\right\}$ assuming there is a special state that is "recurrent under all policies"
- Total expected cost of a transition

$$G(i,u) = g(i,u)\overline{\tau}_i(u),$$

where $\overline{\tau}_i(u)$: Expected transition time.

- We apply the SSP argument used for the discrete-time case.
  - Divide trajectory into cycles marked by successive visits to n.
  - The cost at (i,u) is $G(i,u) - \lambda^*\overline{\tau}_i(u)$, where λ^* is the optimal expected cost per unit time.
  - Each cycle is viewed as a state trajectory of a corresponding SSP problem with the termination state being essentially n.
- So Bellman's Eq. for the average cost problem:

$$h^*(i) = \min_{u\in U(i)} \left[G(i,u) - \lambda^*\overline{\tau}_i(u) + \sum_{j=1}^{n} p_{ij}(u)h^*(j)\right]$$

MANUFACTURER EXAMPLE/AVERAGE COST

- The expected transition times are

$$\overline{\tau}_i(\text{Fill}) = \overline{\tau}_i(\text{Not Fill}) = \frac{\tau_{\max}}{2}$$

the expected transition cost is

$$G(i, \text{Fill}) = 0, \qquad G(i, \text{Not Fill}) = \frac{c\, i\, \tau_{\max}}{2}$$

and there is also the "instantaneous" cost

$$\hat{g}(i, \text{Fill}) = K, \qquad \hat{g}(i, \text{Not Fill}) = 0$$

- Bellman's equation:

$$h^*(i) = \min\Big[K - \lambda^* \frac{\tau_{\max}}{2} + h^*(1), \\ ci\frac{\tau_{\max}}{2} - \lambda^* \frac{\tau_{\max}}{2} + h^*(i+1)\Big]$$

- Again it can be shown that a threshold policy is optimal.

6.231 DYNAMIC PROGRAMMING

LECTURE 14

LECTURE OUTLINE

- We start a ten-lecture sequence on advanced infinite horizon DP and approximation methods
- We allow infinite state space, so the stochastic shortest path framework cannot be used any more
- Results are rigorous assuming a finite or countable disturbance space
  - This includes deterministic problems with arbitrary state space, and countable state Markov chains
  - Otherwise the mathematics of measure theory make analysis difficult, although the final results are essentially the same as for finite disturbance space
- We use Vol. II of the textbook, starting with discounted problems (Ch. 1)
- The central mathematical structure is that the DP mapping is a contraction mapping (instead of existence of a termination state)

DISCOUNTED PROBLEMS/BOUNDED COST

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots$$

- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

$$J_\pi(x_0) = \lim_{N\to\infty} \underset{\substack{w_k \\ k=0,1,\ldots}}{E} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

with $\alpha < 1$, and for some M, we have

$$|g(x, u, w)| \leq M, \qquad \forall\ (x, u, w)$$

- We have

$$\big|J_\pi(x_0)\big| \leq M + \alpha M + \alpha^2 M + \cdots = \frac{M}{1-\alpha}, \qquad \forall\ x_0$$

- The "tail" of the cost $J_\pi(x_0)$ diminishes to 0
- The limit defining $J_\pi(x_0)$ exists

WE ADOPT "SHORTHAND" NOTATION

- Compact pointwise notation for functions:
  - If for two functions J and J' we have $J(x) = J'(x)$ for all x, we write $J = J'$
  - If for two functions J and J' we have $J(x) \leq J'(x)$ for all x, we write $J \leq J'$
  - For a sequence $\{J_k\}$ with $J_k(x) \to J(x)$ for all x, we write $J_k \to J$; also $J^* = \min_\pi J_\pi$
- Shorthand notation for DP mappings (operate on functions of state to produce other functions)

$$(TJ)(x) = \min_{u \in U(x)} \underset{w}{E} \Big\{ g(x,u,w) + \alpha J\Big(f(x,u,w)\Big) \Big\}, \ \forall\, x$$

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ.

- For any stationary policy μ

$$(T_\mu J)(x) = \underset{w}{E} \Big\{ g\Big(x, \mu(x), w\Big) + \alpha J\Big(f(x, \mu(x), w)\Big) \Big\}, \ \forall\, x$$

- For finite-state problems:

$$T_\mu J = g_\mu + \alpha P_\mu J, \qquad TJ = \min_\mu T_\mu J$$

"SHORTHAND" COMPOSITION NOTATION

- Composition notation: T^2J is defined by $(T^2J)(x) = (T(TJ))(x)$ for all x (similar for T^kJ)
- For any policy $\pi = \{\mu_0, \mu_1, \ldots\}$ and function J:
  - $T_{\mu_0}J$ is the cost function of π for the one-stage problem with terminal cost function αJ
  - $T_{\mu_0}T_{\mu_1}J$ (i.e., T_{μ_0} applied to $T_{\mu_1}J$) is the cost function of π for the two-stage problem with terminal cost $\alpha^2 J$
  - $T_{\mu_0}T_{\mu_1}\cdots T_{\mu_{N-1}}J$ is the cost function of π for the N-stage problem with terminal cost $\alpha^N J$
- For any function J:
  - TJ is the optimal cost function of the one-stage problem with terminal cost function αJ
  - T^2J (i.e., T applied to TJ) is the optimal cost function of the two-stage problem with terminal cost $\alpha^2 J$
  - T^NJ is the optimal cost function of the N-stage problem with terminal cost $\alpha^N J$

"SHORTHAND" THEORY – A SUMMARY

- Cost function expressions [with $J_0(x) \equiv 0$]

$$J_\pi(x) = \lim_{k\to\infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0)(x), \quad J_\mu(x) = \lim_{k\to\infty} (T_\mu^k J_0)(x)$$

- Bellman's equation: $J^* = TJ^*, \quad J_\mu = T_\mu J_\mu$
- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

- Value iteration: For any (bounded) J and all x,

$$J^*(x) = \lim_{k\to\infty} (T^k J)(x)$$

- Policy iteration: Given μ^k:
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

  - Policy improvement: Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$$

SOME KEY PROPERTIES

• Monotonicity property: For any functions J and J' such that $J(x) \leq J'(x)$ for all x, and any μ

$$(TJ)(x) \leq (TJ')(x), \qquad \forall\ x,$$

$$(T_\mu J)(x) \leq (T_\mu J')(x), \qquad \forall\ x.$$

Also

$$J \leq TJ \quad \Rightarrow \quad T^k J \leq T^{k+1} J, \qquad \forall\ k$$

• Constant Shift property: For any J, any scalar r, and any μ

$$\big(T(J + re)\big)(x) = (TJ)(x) + \alpha r, \qquad \forall\ x,$$

$$\big(T_\mu(J + re)\big)(x) = (T_\mu J)(x) + \alpha r, \qquad \forall\ x,$$

where e is the unit function [$e(x) \equiv 1$] (holds for most DP models).

• A third important property that holds for some (but not all) DP models is that T and T_μ are contraction mappings (more on this later).

CONVERGENCE OF VALUE ITERATION

- If $J_0 \equiv 0$,

$$J^*(x) = \lim_{N\to\infty} (T^N J_0)(x), \qquad \text{for all } x$$

Proof: For any initial state x_0, and policy $\pi = \{\mu_0, \mu_1, \ldots\}$,

$$\begin{aligned} J_\pi(x_0) &= E\left\{ \sum_{k=0}^{\infty} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\} \\ &= E\left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\} \\ &\quad + E\left\{ \sum_{k=N}^{\infty} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\} \end{aligned}$$

from which

$$J_\pi(x_0) - \frac{\alpha^N M}{1-\alpha} \le (T_{\mu_0} \cdots T_{\mu_{N-1}} J_0)(x_0) \le J_\pi(x_0) + \frac{\alpha^N M}{1-\alpha},$$

where $M \ge |g(x,u,w)|$. Take the min over π of both sides. **Q.E.D.**

BELLMAN'S EQUATION

- The optimal cost function J^* satisfies Bellman's Eq., i.e. $J^* = TJ^*$.

Proof: For all x and N,

$$J^*(x) - \frac{\alpha^N M}{1-\alpha} \leq (T^N J_0)(x) \leq J^*(x) + \frac{\alpha^N M}{1-\alpha},$$

where $J_0(x) \equiv 0$ and $M \geq |g(x,u,w)|$.

- Apply T to this relation and use Monotonicity and Constant Shift,

$$(TJ^*)(x) - \frac{\alpha^{N+1} M}{1-\alpha} \leq (T^{N+1} J_0)(x) \leq (TJ^*)(x) + \frac{\alpha^{N+1} M}{1-\alpha}$$

- Take limit as $N \to \infty$ and use the fact

$$\lim_{N \to \infty} (T^{N+1} J_0)(x) = J^*(x)$$

to obtain $J^* = TJ^*$. **Q.E.D.**

THE CONTRACTION PROPERTY

• Contraction property: For any bounded functions J and J', and any μ,

$$\max_x \left|(TJ)(x) - (TJ')(x)\right| \leq \alpha \max_x \left|J(x) - J'(x)\right|,$$

$$\max_x \left|(T_\mu J)(x) - (T_\mu J')(x)\right| \leq \alpha \max_x \left|J(x) - J'(x)\right|.$$

Proof: Denote $c = \max_{x \in S} \left|J(x) - J'(x)\right|$. Then

$$J(x) - c \leq J'(x) \leq J(x) + c, \qquad \forall\ x$$

Apply T to both sides, and use the Monotonicity and Constant Shift properties:

$$(TJ)(x) - \alpha c \leq (TJ')(x) \leq (TJ)(x) + \alpha c, \qquad \forall\ x$$

Hence

$$\left|(TJ)(x) - (TJ')(x)\right| \leq \alpha c, \qquad \forall\ x.$$

Similar for T_μ. **Q.E.D.**

IMPLICATIONS OF CONTRACTION PROPERTY

- We can strengthen our earlier result:
- Bellman's equation $J = TJ$ has a unique solution, namely J^*, and for any bounded J, we have

$$\lim_{k\to\infty} (T^k J)(x) = J^*(x), \qquad \forall\ x$$

Proof: Use

$$\max_x \left|(T^k J)(x) - J^*(x)\right| = \max_x \left|(T^k J)(x) - (T^k J^*)(x)\right|$$
$$\leq \alpha^k \max_x \left|J(x) - J^*(x)\right|$$

- Special Case: For each stationary μ, J_μ is the unique solution of $J = T_\mu J$ and

$$\lim_{k\to\infty} (T_\mu^k J)(x) = J_\mu(x), \qquad \forall\ x,$$

for any bounded J.

- Convergence rate: For all k,

$$\max_x \left|(T^k J)(x) - J^*(x)\right| \leq \alpha^k \max_x \left|J(x) - J^*(x)\right|$$

NEC. AND SUFFICIENT OPT. CONDITION

- A stationary policy μ is optimal if and only if $\mu(x)$ attains the minimum in Bellman's equation for each x; i.e.,

$$TJ^* = T_\mu J^*.$$

Proof: If $TJ^* = T_\mu J^*$, then using Bellman's equation ($J^* = TJ^*$), we have

$$J^* = T_\mu J^*,$$

so by uniqueness of the fixed point of T_μ, we obtain $J^* = J_\mu$; i.e., μ is optimal.

- Conversely, if the stationary policy μ is optimal, we have $J^* = J_\mu$, so

$$J^* = T_\mu J^*.$$

Combining this with Bellman's equation ($J^* = TJ^*$), we obtain $TJ^* = T_\mu J^*$. **Q.E.D.**

COMPUTATIONAL METHODS - AN OVERVIEW

- Typically must work with a finite-state system. Possibly an approximation of the original system.
- Value iteration and variants
  - Gauss-Seidel and asynchronous versions
- Policy iteration and variants
  - Combination with (possibly asynchronous) value iteration
  - "Optimistic" policy iteration
- Linear programming

$$\text{maximize} \quad \sum_{i=1}^{n} J(i)$$

$$\text{subject to} \quad J(i) \leq g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J(j), \quad \forall \ (i,u)$$

- Versions with subspace approximation: Use in place of $J(i)$ a low-dim. basis function representation, with state features $\phi_m(i)$, $m = 1, \ldots, s$

$$\tilde{J}(i,r) = \sum_{m=1}^{s} r_m \phi_m(i)$$

and modify the basic methods appropriately.

USING Q-FACTORS I

- Let the states be $i = 1, \ldots, n$. We can write Bellman's equation as

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u) \qquad i = 1, \ldots, n,$$

where

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big)$$

for all (i,u)

- $Q^*(i,u)$ is called the optimal Q-factor of (i,u)
- Q-factors have optimal cost interpretation in an "augmented" problem whose states are i and (i,u), $u \in U(i)$ - the optimal cost vector is (J^*, Q^*)
- The Bellman Eq. is $J^* = TJ^*$, $Q^* = FQ^*$ where

$$(FQ^*)(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \min_{v \in U(j)} Q^*(j,v) \right)$$

- It has a unique solution.

USING Q-FACTORS II

- We can equivalently write the VI method as

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i,u), \qquad i = 1, \ldots, n,$$

where Q_{k+1} is generated for all i and $u \in U(i)$ by

$$Q_{k+1}(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \min_{v \in U(j)} Q_k(j,v) \right)$$

or $J_{k+1} = TJ_k$, $Q_{k+1} = FQ_k$.

- Equal amount of computation ... just more storage.

- Having optimal Q-factors is convenient when implementing an optimal policy on-line by

$$\mu^*(i) = \min_{u \in U(i)} Q^*(i,u)$$

- Once $Q^*(i,u)$ are known, the model [g and $p_{ij}(u)$] is not needed. Model-free operation.

- Stochastic/sampling methods can be used to calculate (approximations of) $Q^*(i,u)$ [not $J^*(i)$] with a simulator of the system.

6.231 DYNAMIC PROGRAMMING

LECTURE 15

LECTURE OUTLINE

- Review of basic theory of discounted problems
- Monotonicity and contraction properties
- Contraction mappings in DP
- Discounted problems: Countable state space with unbounded costs
- Generalized discounted DP
- An introduction to abstract DP

DISCOUNTED PROBLEMS/BOUNDED COST

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots$$

- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

$$J_\pi(x_0) = \lim_{N\to\infty} \underset{\substack{w_k \\ k=0,1,\ldots}}{E} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

with $\alpha < 1$, and for some M, we have $|g(x, u, w)| \leq M$ for all (x, u, w)

- Shorthand notation for DP mappings (operate on functions of state to produce other functions)

$$(TJ)(x) = \min_{u\in U(x)} \underset{w}{E} \left\{ g(x, u, w) + \alpha J\big(f(x, u, w)\big) \right\}, \ \forall\, x$$

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ.

- For any stationary policy μ

$$(T_\mu J)(x) = \underset{w}{E} \left\{ g\big(x, \mu(x), w\big) + \alpha J\big(f(x, \mu(x), w)\big) \right\}, \ \forall\, x$$

"SHORTHAND" THEORY – A SUMMARY

- Cost function expressions [with $J_0(x) \equiv 0$]

$$J_\pi(x) = \lim_{k\to\infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0)(x), \quad J_\mu(x) = \lim_{k\to\infty} (T_\mu^k J_0)(x)$$

- Bellman's equation: $J^* = TJ^*, \quad J_\mu = T_\mu J_\mu$
- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

- Value iteration: For any (bounded) J and all x:

$$J^*(x) = \lim_{k\to\infty} (T^k J)(x)$$

- Policy iteration: Given μ^k,
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

  - Policy improvement: Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$$

MAJOR PROPERTIES

- Monotonicity property: For any functions J and J' on the state space X such that $J(x) \leq J'(x)$ for all $x \in X$, and any μ

$$(TJ)(x) \leq (TJ')(x), \qquad \forall\ x \in X,$$

$$(T_\mu J)(x) \leq (T_\mu J')(x), \qquad \forall\ x \in X.$$

- Contraction property: For any bounded functions J and J', and any μ,

$$\max_x \big|(TJ)(x) - (TJ')(x)\big| \leq \alpha \max_x \big|J(x) - J'(x)\big|,$$

$$\max_x \big|(T_\mu J)(x) - (T_\mu J')(x)\big| \leq \alpha \max_x \big|J(x) - J'(x)\big|.$$

- Shorthand writing of the contraction property

$$\|TJ - TJ'\| \leq \alpha \|J - J'\|, \quad \|T_\mu J - T_\mu J'\| \leq \alpha \|J - J'\|,$$

where for any bounded function J, we denote by $\|J\|$ the sup-norm

$$\|J\| = \max_{x \in X} \big|J(x)\big|.$$

CONTRACTION MAPPINGS

- Given a real vector space Y with a norm $\|\cdot\|$ (see text for definitions).

- A function $F : Y \mapsto Y$ is said to be a *contraction mapping* if for some $\rho \in (0, 1)$, we have

$$\|Fy - Fz\| \leq \rho \|y - z\|, \qquad \text{for all } y, z \in Y.$$

ρ is called the *modulus of contraction* of F.

- Linear case, $Y = \Re^n$: $Fy = Ay + b$ is a contraction (for some norm $\|\cdot\|$) if and only if all eigenvalues of A are strictly within the unit circle.

- For $m > 1$, we say that F is an *m-stage contraction* if F^m is a contraction.

- Important example: Let X be a set (e.g., state space in DP), $v : X \mapsto \Re$ be a positive-valued function. Let $B(X)$ be the set of all functions $J : X \mapsto \Re$ such that $J(s)/v(s)$ is bounded over s.

- The *weighted sup-norm* on $B(X)$:

$$\|J\| = \max_{s \in X} \frac{|J(s)|}{v(s)}.$$

- Important special case: The discounted problem mappings T and T_μ [for $v(s) \equiv 1$, $\rho = \alpha$].

A DP-LIKE CONTRACTION MAPPING

- Let $X = \{1, 2, \ldots\}$, and let $F : B(X) \mapsto B(X)$ be a linear mapping of the form

$$(FJ)(i) = b(i) + \sum_{j \in X} a(i,j)\, J(j), \qquad \forall\ i$$

where $b(i)$ and $a(i,j)$ are some scalars. Then F is a contraction with modulus ρ if

$$\frac{\sum_{j \in X} |a(i,j)|\, v(j)}{v(i)} \leq \rho, \qquad \forall\ i$$

[Think of the special case where $a(i,j)$ are the transition probs. of a policy].

- Let $F : B(X) \mapsto B(X)$ be the mapping

$$(FJ)(i) = \min_{\mu \in M} (F_\mu J)(i), \qquad \forall\ i$$

where M is parameter set, and for each $\mu \in M$, F_μ is a contraction from $B(X)$ to $B(X)$ with modulus ρ. Then F is a contraction with modulus ρ.

CONTRACTION MAPPING FIXED-POINT TH.

- Contraction Mapping Fixed-Point Theorem: If $F : B(X) \mapsto B(X)$ is a contraction with modulus $\rho \in (0, 1)$, then there exists a unique $J^* \in B(X)$ such that

$$J^* = FJ^*.$$

Furthermore, if J is any function in $B(X)$, then $\{F^k J\}$ converges to J^* and we have

$$\|F^k J - J^*\| \leq \rho^k \|J - J^*\|, \qquad k = 1, 2, \ldots.$$

- Similar result if F is an m-stage contraction mapping.

- This is a special case of a general result for contraction mappings $F : Y \mapsto Y$ over normed vector spaces Y that are complete: every sequence $\{y_k\}$ that is Cauchy (satisfies $\|y_m - y_n\| \to 0$ as $m, n \to \infty$) converges.

- The space $B(X)$ is complete [see the text (Section 1.5) for a proof].

GENERAL FORMS OF DISCOUNTED DP

- Monotonicity assumption: If $J, J' \in R(X)$ and $J \leq J'$, then

$$H(x,u,J) \leq H(x,u,J'), \qquad \forall\ x \in X,\ u \in U(x)$$

- Contraction assumption:
  - For every $J \in B(X)$, the functions $T_\mu J$ and TJ belong to $B(X)$.
  - For some $\alpha \in (0,1)$ and all $J, J' \in B(X)$, H satisfies

$$\big|H(x,u,J) - H(x,u,J')\big| \leq \alpha \max_{y \in X} \big|J(y) - J'(y)\big|$$

    for all $x \in X$ and $u \in U(x)$.

- We can show all the standard analytical and computational results of discounted DP based on these two assumptions (with identical proofs!)

- With just the monotonicity assumption (as in shortest path problem) we can still show various forms of the basic results under appropriate assumptions (like in the SSP problem)

EXAMPLES

- Discounted problems

$$H(x,u,J) = E\big\{g(x,u,w) + \alpha J\big(f(x,u,w)\big)\big\}$$

- Discounted Semi-Markov Problems

$$H(x,u,J) = G(x,u) + \sum_{y=1}^{n} m_{xy}(u)J(y)$$

where m_{xy} are "discounted" transition probabilities, defined by the transition distributions

- Deterministic Shortest Path Problems

$$H(x,u,J) = \begin{cases} a_{xu} + J(u) & \text{if } u \neq t, \\ a_{xt} & \text{if } u = t \end{cases}$$

where t is the destination

- Minimax Problems

$$H(x,u,J) = \max_{w \in W(x,u)} \big[g(x,u,w) + \alpha J\big(f(x,u,w)\big)\big]$$

RESULTS USING CONTRACTION

- The mappings T_μ and T are sup-norm contraction mappings with modulus α over $B(X)$, and have unique fixed points in $B(X)$, denoted J_μ and J^*, respectively (cf. Bellman's equation). *Proof*: From contraction assumption and fixed point Th.

- For any $J \in B(X)$ and $\mu \in \mathcal{M}$,

$$\lim_{k\to\infty} T_\mu^k J = J_\mu, \qquad \lim_{k\to\infty} T^k J = J^*$$

(cf. convergence of value iteration). *Proof*: From contraction property of T_μ and T.

- We have $T_\mu J^* = TJ^*$ if and only if $J_\mu = J^*$ (cf. optimality condition). *Proof*: $T_\mu J^* = TJ^*$, then $T_\mu J^* = J^*$, implying $J^* = J_\mu$. Conversely, if $J_\mu = J^*$, then $T_\mu J^* = T_\mu J_\mu = J_\mu = J^* = TJ^*$.

- Useful bound for J_μ: For all $J \in B(X)$, $\mu \in \mathcal{M}$

$$\|J_\mu - J\| \le \frac{\|T_\mu J - J\|}{1-\alpha}$$

Proof: Take limit as $k \to \infty$ in the relation

$$\|T_\mu^k J - J\| \le \sum_{\ell=1}^{k} \|T_\mu^\ell J - T_\mu^{\ell-1} J\| \le \|T_\mu J - J\| \sum_{\ell=1}^{k} \alpha^{\ell-1}$$

RESULTS USING MON. AND CONTRACTION I

- Existence of a nearly optimal policy: For every $\epsilon > 0$, there exists $\mu_\epsilon \in \mathcal{M}$ such that

$$J^*(x) \leq J_{\mu_\epsilon}(x) \leq J^*(x) + \epsilon v(x), \qquad \forall\ x \in X$$

Proof: For all $\mu \in \mathcal{M}$, we have $J^* = TJ^* \leq T_\mu J^*$. By monotonicity, $J^* \leq T_\mu^{k+1} J^* \leq T_\mu^k J^*$ for all k. Taking limit as $k \to \infty$, we obtain $J^* \leq J_\mu$.

Also, choose $\mu_\epsilon \in \mathcal{M}$ such that for all $x \in X$,

$$\|T_{\mu_\epsilon} J^* - J^*\| = \big\|(T_{\mu_\epsilon} J^*)(x) - (TJ^*)(x)\big\| \leq \epsilon(1-\alpha)$$

From the earlier error bound, we have

$$\|J_\mu - J^*\| \leq \frac{\|T_\mu J^* - J^*\|}{1-\alpha}, \qquad \forall\ \mu \in \mathcal{M}$$

Combining the preceding two relations,

$$\frac{\big|J_{\mu_\epsilon}(x) - J^*(x)\big|}{v(x)} \leq \frac{\epsilon(1-\alpha)}{1-\alpha} = \epsilon, \qquad \forall\ x \in X$$

- Optimality of J^* over stationary policies:

$$J^*(x) = \min_{\mu \in \mathcal{M}} J_\mu(x), \qquad \forall\ x \in X$$

Proof: Take $\epsilon \downarrow 0$ in the preceding result.

RESULTS USING MON. AND CONTRACTION II

- Nonstationary policies: Consider the set Π of all sequences $\pi = \{\mu_0, \mu_1, \ldots\}$ with $\mu_k \in \mathcal{M}$ for all k, and define for any $J \in B(X)$

$$J_\pi(x) = \limsup_{k\to\infty} (T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J)(x), \qquad \forall\, x \in X,$$

(the choice of J does not matter because of the contraction property).

- Optimality of J^* over nonstationary policies:

$$J^*(x) = \min_{\pi\in\Pi} J_\pi(x), \qquad \forall\, x \in X$$

Proof: Use our earlier existence result to show that for any $\epsilon > 0$, there is μ_ϵ such that $\|J_{\mu_\epsilon} - J^*\| \leq \epsilon(1-\alpha)$. We have

$$J^*(x) = \min_{\mu\in\mathcal{M}} J_\mu(x) \geq \min_{\pi\in\Pi} J_\pi(x)$$

Also

$$T^k J \leq T_{\mu_0} \cdots T_{\mu_{k-1}} J$$

Take limit as $k \to \infty$ to obtain $J \leq J_\pi$ for all $\pi \in \Pi$.

6.231 DYNAMIC PROGRAMMING

LECTURE 16

LECTURE OUTLINE

- Review of computational theory of discounted problems
- Value iteration (VI), policy iteration (PI)
- Optimistic PI
- Computational methods for generalized discounted DP
- Asynchronous algorithms

DISCOUNTED PROBLEMS

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots$$

- Bounded g. Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

$$J_\pi(x_0) = \lim_{N\to\infty} \mathop{E}_{\substack{w_k \\ k=0,1,\ldots}} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

- Shorthand notation for DP mappings (n-state Markov chain case)

$$(TJ)(x) = \min_{u\in U(x)} E\big\{g(x,u,w)+\alpha J\big(f(x,u,w)\big)\big\}, \ \forall\, x$$

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ.

- For any stationary policy μ

$$(T_\mu J)(x) = E\big\{g(x,\mu(x),w)+\alpha J\big(f(x,\mu(x),w)\big)\big\}, \ \forall\, x$$

Note: T_μ is linear [in short $T_\mu J = P_\mu(g_\mu + \alpha J)$].

"SHORTHAND" THEORY – A SUMMARY

- Cost function expressions (with $J_0 \equiv 0$)

$$J_\pi = \lim_{k\to\infty} T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0, \qquad J_\mu = \lim_{k\to\infty} T_\mu^k J_0$$

- Bellman's equation: $J^* = TJ^*, \;\; J_\mu = T_\mu J_\mu$
- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

- Contraction: $\|TJ_1 - TJ_2\| \le \alpha \|J_1 - J_2\|$
- Value iteration: For any (bounded) J

$$J^* = \lim_{k\to\infty} T^k J$$

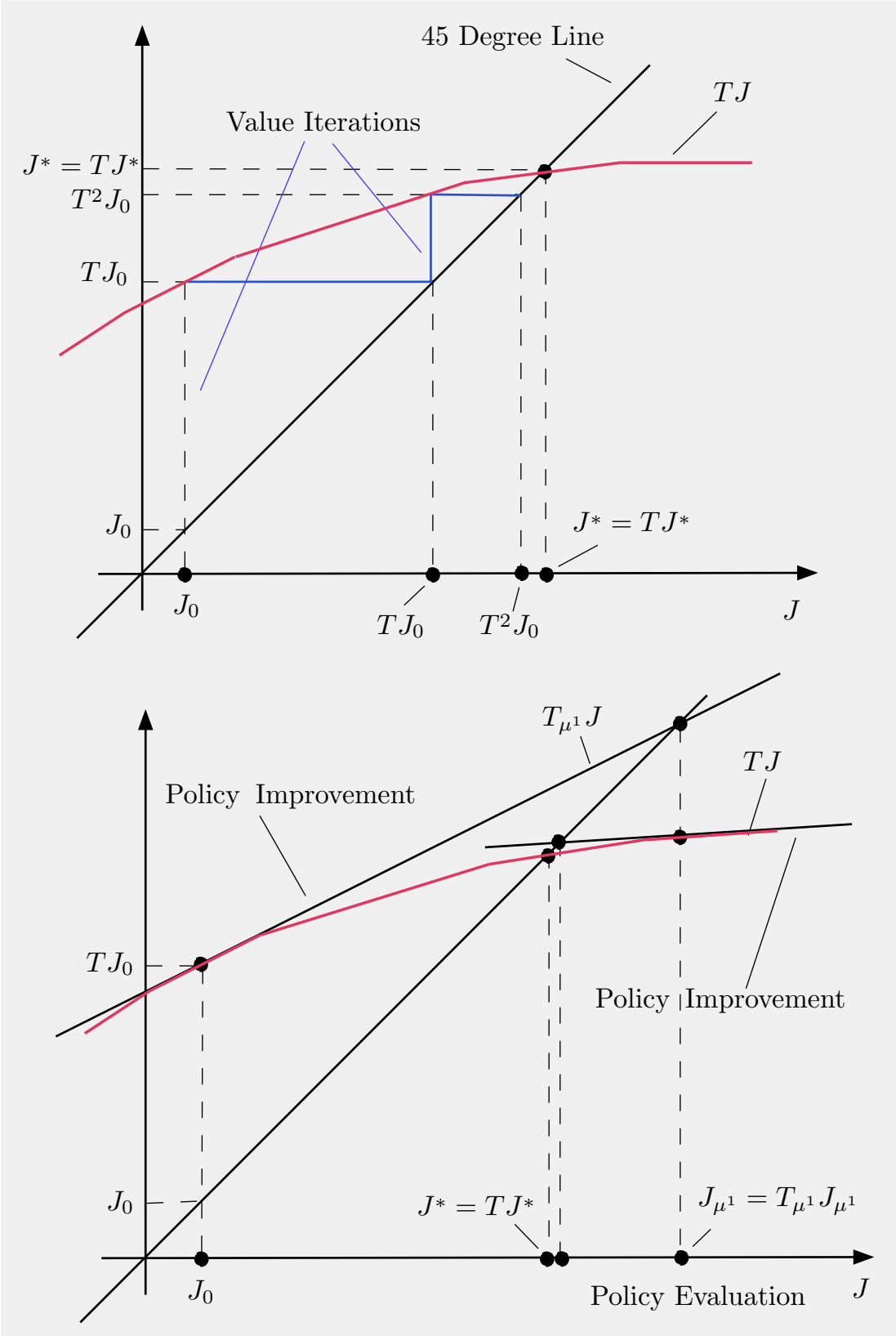
- Policy iteration: Given μ^k,
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

  - Policy improvement: Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$$

INTERPRETATION OF VI AND PI



VI AND PI METHODS FOR Q-LEARNING

- We can write Bellman's equation as

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u) \qquad i = 1, \ldots, n,$$

where Q^* is the vector of optimal Q-factors

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big)$$

- VI and PI for Q-factors are mathematically equivalent to VI and PI for costs.

- They require equal amount of computation ... they just need more storage.

- For example, we can write the VI method as

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i,u), \qquad i = 1, \ldots, n,$$

where Q_{k+1} is generated for all i and $u \in U(i)$ by

$$Q_{k+1}(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{v \in U(j)} Q_k(j,v)\right)$$

APPROXIMATE PI

- Suppose that the policy evaluation is approximate, according to,

$$\max_x |J_k(x) - J_{\mu^k}(x)| \leq \delta, \qquad k = 0, 1, \ldots$$

and policy improvement is approximate, according to,

$$\max_x |(T_{\mu^{k+1}} J_k)(x) - (T J_k)(x)| \leq \epsilon, \qquad k = 0, 1, \ldots$$

where δ and ϵ are some positive scalars.

- Error Bound: The sequence $\{\mu^k\}$ generated by approximate policy iteration satisfies

$$\limsup_{k \to \infty} \max_{x \in S} \big(J_{\mu^k}(x) - J^*(x)\big) \leq \frac{\epsilon + 2\alpha\delta}{(1-\alpha)^2}$$

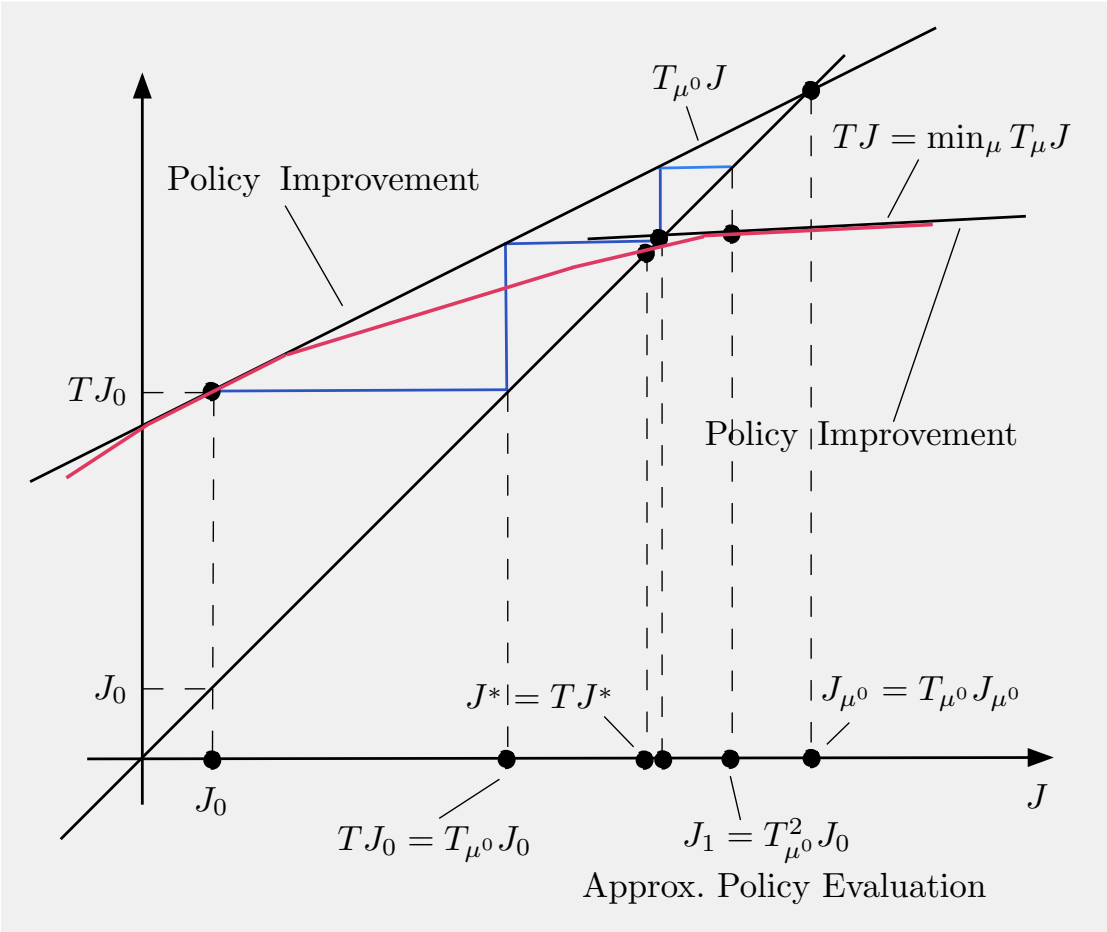
- Typical practical behavior: The method makes steady progress up to a point and then the iterates J_{μ^k} oscillate within a neighborhood of J^*.

OPTIMISTIC PI

- This is PI, where policy evaluation is carried out by a finite number of VI
- Shorthand definition: For some integers m_k

$$T_{\mu^k} J_k = TJ_k, \qquad J_{k+1} = T_{\mu^k}^{m_k} J_k, \qquad k = 0, 1, \ldots$$

  - If $m_k \equiv 1$ it becomes VI
  - If $m_k = \infty$ it becomes PI
  - For intermediate values of m_k, it is generally more efficient than either VI or PI



EXTENSIONS TO GENERALIZED DISC. DP

- All the preceding VI and PI methods extend to generalized/abstract discounted DP.
- Summary: For a mapping $H : X \times U \times R(X) \mapsto \Re$, consider

$$(TJ)(x) = \min_{u \in U(x)} H(x, u, J), \qquad \forall\ x \in X.$$

$$(T_\mu J)(x) = H\big(x, \mu(x), J\big), \qquad \forall\ x \in X.$$

- We want to find J^* such that

$$J^*(x) = \min_{u \in U(x)} H(x, u, J^*), \qquad \forall\ x \in X$$

and a μ^* such that $T_{\mu^*} J^* = TJ^*$.

- Discounted, Discounted Semi-Markov, Minimax

$$H(x, u, J) = E\big\{g(x, u, w) + \alpha J\big(f(x, u, w)\big)\big\}$$

$$H(x, u, J) = G(x, u) + \sum_{y=1}^{n} m_{xy}(u) J(y)$$

$$H(x, u, J) = \max_{w \in W(x,u)} \big[g(x, u, w) + \alpha J\big(f(x, u, w)\big)\big]$$

ASSUMPTIONS AND RESULTS

- Monotonicity assumption: If $J, J' \in R(X)$ and $J \leq J'$, then

$$H(x,u,J) \leq H(x,u,J'), \qquad \forall\ x \in X,\ u \in U(x)$$

- Contraction assumption:
  - For every $J \in B(X)$, the functions $T_\mu J$ and TJ belong to $B(X)$.
  - For some $\alpha \in (0,1)$ and all $J, J' \in B(X)$, H satisfies

$$\big|H(x,u,J) - H(x,u,J')\big| \leq \alpha \max_{y \in X} \big|J(y) - J'(y)\big|$$

    for all $x \in X$ and $u \in U(x)$.

- Standard algorithmic results extend:
  - Generalized VI converges to J^*, the unique fixed point of T
  - Generalized PI and optimistic PI generate $\{\mu^k\}$ such that

$$\lim_{k\to\infty} \|J_{\mu^k} - J^*\| = 0, \qquad \lim_{k\to\infty} \|J_k - J^*\| = 0$$

- Analytical Approach: Start with a problem, match it with an H, invoke the general results.

ASYNCHRONOUS ALGORITHMS

- Motivation for asynchronous algorithms
  - Faster convergence
  - Parallel and distributed computation
  - Simulation-based implementations

- General framework: Partition X into disjoint nonempty subsets $X_1, \ldots, X_m$, and use separate processor ℓ updating $J(x)$ for $x \in X_\ell$.

- Let J be partitioned as $J = (J_1, \ldots, J_m)$, where J_ℓ is the restriction of J on the set X_ℓ.

- Synchronous algorithm: Processor ℓ updates J for the states $x \in X_\ell$ at all times t,

$$J_\ell^{t+1}(x) = T(J_1^t, \ldots, J_m^t)(x), \quad x \in X_\ell, \ \ell = 1, \ldots, m$$

- Asynchronous algorithm: Processor ℓ updates J for the states $x \in X_\ell$ only at a subset of times $\mathcal{R}_\ell$,

$$J_\ell^{t+1}(x) = \begin{cases} T\big(J_1^{\tau_{\ell 1}(t)}, \ldots, J_m^{\tau_{\ell m}(t)}\big)(x) & \text{if } t \in \mathcal{R}_\ell, \\ J_\ell^t(x) & \text{if } t \notin \mathcal{R}_\ell \end{cases}$$

where $t - \tau_{\ell j}(t)$ are communication "delays"

ONE-STATE-AT-A-TIME ITERATIONS

- Important special case: Assume n "states", a separate processor for each state, and no delays
- Generate a sequence of states $\{x^0, x^1, \ldots\}$, generated in some way, possibly by simulation (each state is generated infinitely often)
- Asynchronous VI: Change any one component of J^t at time t, the one that corresponds to x^t:

$$J^{t+1}(\ell) = \begin{cases} T\big(J^t(1), \ldots, J^t(n)\big)(\ell) & \text{if } \ell = x^t, \\ J^t(\ell) & \text{if } \ell \neq x^t, \end{cases}$$

- The special case where

$$\{x^0, x^1, \ldots\} = \{1, \ldots, n, 1, \ldots, n, 1, \ldots\}$$

is the Gauss-Seidel method

- More generally, the components used at time t are delayed by $t - \tau_{\ell j}(t)$
- Flexible in terms of timing and "location" of the iterations
- We can show that $J^t \to J^*$ under assumptions typically satisfied in DP

ASYNCHRONOUS CONV. THEOREM I

- Assume that for all $\ell, j = 1, \ldots, m$, the set of times $\mathcal{R}_\ell$ is infinite and $\lim_{t\to\infty} \tau_{\ell j}(t) = \infty$

- Proposition: Let T have a unique fixed point J^*, and assume that there is a sequence of nonempty subsets $\{S(k)\} \subset R(X)$ with $S(k+1) \subset S(k)$ for all k, and with the following properties:

(1) Synchronous Convergence Condition: Every sequence $\{J^k\}$ with $J^k \in S(k)$ for each k, converges pointwise to J^*. Moreover, we have

$$TJ \in S(k+1), \qquad \forall\, J \in S(k),\ k = 0, 1, \ldots.$$

(2) Box Condition: For all k, $S(k)$ is a Cartesian product of the form

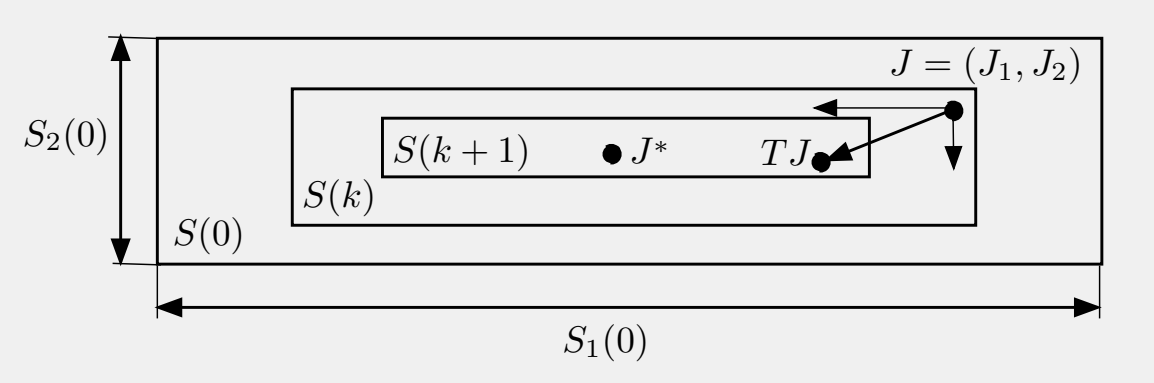
$$S(k) = S_1(k) \times \cdots \times S_m(k),$$

where $S_\ell(k)$ is a set of real-valued functions on X_ℓ, $\ell = 1, \ldots, m$.

Then for every $J \in S(0)$, the sequence $\{J^t\}$ generated by the asynchronous algorithm converges pointwise to J^*.

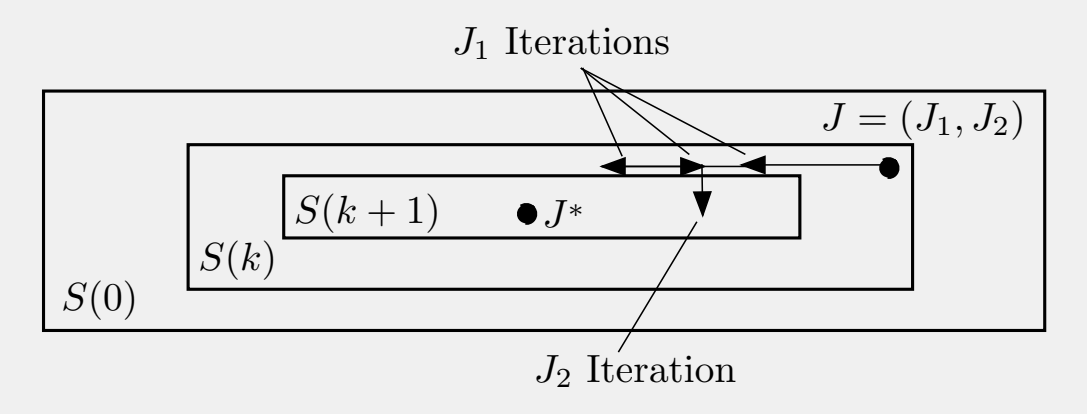
ASYNCHRONOUS CONV. THEOREM II

- Interpretation of assumptions:



A synchronous iteration from any J in $S(k)$ moves into $S(k+1)$ (component-by-component)

- Convergence mechanism:



Key: "Independent" component-wise improvement. An asynchronous component iteration from any J in $S(k)$ moves into the corresponding component portion of $S(k+1)$ permanently!

PRINCIPAL DP APPLICATIONS

- The assumptions of the asynchronous convergence theorem are satisfied in two principal cases:
  - When T is a (weighted) sup-norm contraction.
  - When T is monotone and the Bellman equation $J = TJ$ has a unique solution.
- The theorem can be applied also to convergence of asynchronous optimistic PI for:
  - Discounted problems (Section 2.6.2 of the text).
  - SSP problems (Section 3.5 of the text).
- There are variants of the theorem that can be applied in the presence of special structure.
- Asynchronous convergence ideas also underlie stochastic VI algorithms like Q-learning.

6.231 DYNAMIC PROGRAMMING

LECTURE 17

LECTURE OUTLINE

- Undiscounted problems
- Stochastic shortest path problems (SSP)
- Proper and improper policies
- Analysis and computational methods for SSP
- Pathologies of SSP
- SSP under weak conditions

UNDISCOUNTED PROBLEMS

- System: $x_{k+1} = f(x_k, u_k, w_k)$
- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

$$J_\pi(x_0) = \limsup_{N\to\infty} \underset{\substack{w_k \\ k=0,1,\ldots}}{E} \left\{ \sum_{k=0}^{N-1} g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

Note that $J_\pi(x_0)$ and $J^*(x_0)$ can be $+\infty$ or $-\infty$

- Shorthand notation for DP mappings

$$(TJ)(x) = \min_{u\in U(x)} \underset{w}{E} \Big\{ g(x,u,w) + J\big(f(x,u,w)\big) \Big\}, \ \forall\ x$$

$$(T_\mu J)(x) = \underset{w}{E} \Big\{ g\big(x,\mu(x),w\big) + J\big(f(x,\mu(x),w)\big) \Big\}, \ \forall\ x$$

- T and T_μ need not be contractions in general, but their monotonicity is helpful (see Ch. 4, Vol. II of text for an analysis).

- SSP problems provide a "soft boundary" between the easy finite-state discounted problems and the hard undiscounted problems.
  - They share features of both.
  - Some nice theory is recovered thanks to the termination state, and special conditions.

SSP THEORY SUMMARY I

- As before, we have a cost-free term. state t, a finite number of states $1,\dots,n$, and finite number of controls.

- Mappings T and T_μ (modified to account for termination state t). For all $i=1,\dots,n$:

$$(T_\mu J)(i) = g\big(i,\mu(i)\big) + \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)J(j),$$

$$(TJ)(i) = \min_{u\in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u)J(j) \right],$$

or $T_\mu J = g_\mu + P_\mu J$ and $TJ = \min_\mu [g_\mu + P_\mu J]$.

- Definition: A stationary policy μ is called proper, if under μ, from every state i, there is a positive probability path that leads to t.

- Important fact: (To be shown) If μ is proper, T_μ is contraction w. r. t. some weighted sup-norm

$$\max_i \frac{1}{v_i}|(T_\mu J)(i)-(T_\mu J')(i)| \le \rho_\mu \max_i \frac{1}{v_i}|J(i)-J'(i)|$$

- T is similarly a contraction if all μ are proper (the case discussed in the text, Ch. 7, Vol. I).

SSP THEORY SUMMARY II

- The theory can be pushed one step further. Instead of all policies being proper, assume that:

  (a) There exists at least one proper policy

  (b) For each improper μ, $J_\mu(i) = \infty$ for some i

- Example: Deterministic shortest path problem with a single destination t.
  - States <=> nodes; Controls <=> arcs
  - Termination state <=> the destination
  - Assumption (a) <=> every node is connected to the destination
  - Assumption (b) <=> all cycle costs > 0
- Note that T is not necessarily a contraction.
- The theory in summary is as follows:
  - J^* is the unique solution of Bellman's Eq.
  - μ^* is optimal if and only if $T_{\mu^*} J^* = TJ^*$
  - VI converges: $T^k J \to J^*$ for all $J \in \Re^n$
  - PI terminates with an optimal policy, if started with a proper policy

SSP ANALYSIS I

- For a proper policy μ, J_μ is the unique fixed point of T_μ, and $T_\mu^k J \to J_\mu$ for all J (holds by the theory of Vol. I, Section 7.2)

- Key Fact: A μ satisfying $J \geq T_\mu J$ for some $J \in \Re^n$ must be proper - true because

$$J \geq T_\mu^k J = P_\mu^k J + \sum_{m=0}^{k-1} P_\mu^m g_\mu$$

since $J_\mu = \sum_{m=0}^{\infty} P_\mu^m g_\mu$ and some component of the term on the right blows up as $k \to \infty$ if μ is improper (by our assumptions).

- Consequence: T can have at most one fixed point within $\Re^n$.

Proof: If J and J' are two fixed points, select μ and μ' such that $J = TJ = T_\mu J$ and $J' = TJ' = T_{\mu'} J'$. By preceding assertion, μ and μ' must be proper, and $J = J_\mu$ and $J' = J_{\mu'}$. Also

$$J = T^k J \leq T_{\mu'}^k J \to J_{\mu'} = J'$$

Similarly, $J' \leq J$, so $J = J'$.

SSP ANALYSIS II

- We first show that T has a fixed point, and also that PI converges to it.
- Use PI. Generate a sequence of proper policies $\{\mu^k\}$ starting from a proper policy μ^0.
- μ^1 is proper and $J_{\mu^0} \geq J_{\mu^1}$ since

$$J_{\mu^0} = T_{\mu^0} J_{\mu^0} \geq T J_{\mu^0} = T_{\mu^1} J_{\mu^0} \geq T_{\mu^1}^k J_{\mu^0} \geq J_{\mu^1}$$

- Thus $\{J_{\mu^k}\}$ is nonincreasing, some policy $\bar{\mu}$ is repeated and $J_{\bar{\mu}} = T J_{\bar{\mu}}$. So $J_{\bar{\mu}}$ is fixed point of T.
- Next show that $T^k J \to J_{\bar{\mu}}$ for all J, i.e., VI converges to the same limit as PI. (Sketch: True if $J = J_{\bar{\mu}}$, argue using the properness of $\bar{\mu}$ to show that the terminal cost difference $J - J_{\bar{\mu}}$ does not matter.)
- To show $J_{\bar{\mu}} = J^*$, for any $\pi = \{\mu_0, \mu_1, \ldots\}$

$$T_{\mu_0} \cdots T_{\mu_{k-1}} J_0 \geq T^k J_0,$$

where $J_0 \equiv 0$. Take lim sup as $k \to \infty$, to obtain $J_\pi \geq J_{\bar{\mu}}$, so $\bar{\mu}$ is optimal and $J_{\bar{\mu}} = J^*$.

SSP ANALYSIS III

- Contraction Property: If all policies are proper (cf. Section 7.1, Vol. I), T_μ and T are contractions with respect to a weighted sup norm.

Proof: Consider a new SSP problem where the transition probabilities are the same as in the original, but the transition costs are all equal to -1. Let $\hat{J}$ be the corresponding optimal cost vector. For all μ,

$$\hat{J}(i) = -1 + \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\hat{J}(j) \leq -1 + \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\hat{J}(j)$$

For $v_i = -\hat{J}(i)$, we have $v_i \geq 1$, and for all μ,

$$\sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)v_j \leq v_i - 1 \leq \rho\, v_i, \qquad i = 1, \ldots, n,$$

where

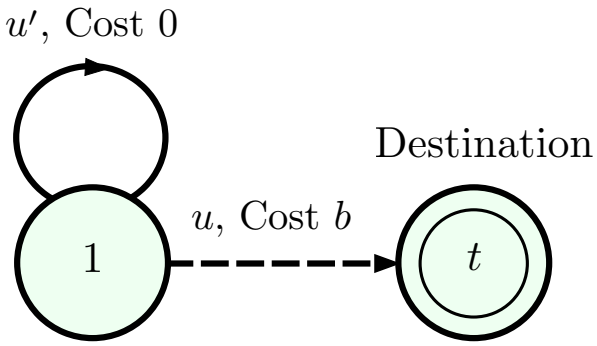
$$\rho = \max_{i=1,\ldots,n} \frac{v_i - 1}{v_i} < 1.$$

This implies T_μ and T are contractions of modulus ρ for norm $\|J\| = \max_{i=1,\ldots,n} |J(i)|/v_i$ (by the results of earlier lectures).

SSP ALGORITHMS

- All the basic algorithms have counterparts under our assumptions; see the text (Ch. 3, Vol. II)
- "Easy" case: All policies proper, in which case the mappings T and T_μ are contractions
- Even with improper (infinite cost) policies all basic algorithms have satisfactory counterparts
  - VI and PI
  - Optimistic PI
  - Asynchronous VI
  - Asynchronous PI
  - Q-learning analogs
- ** THE BOUNDARY OF NICE THEORY **
- Serious complications arise under any one of the following:
  - There is no proper policy
  - There is improper policy with finite cost $\forall\ i$
  - The state space is infinite and/or the control space is infinite [infinite but compact $U(i)$ can be dealt with]

PATHOLOGIES I: DETERM. SHORTEST PATHS



- Two policies, one proper (apply u), one improper (apply u')
- Bellman's equation is

$$J(1) = \min\big[J(1), b\big]$$

Set of solutions is $(-\infty, b]$.

- Case $b > 0$, $J^* = 0$: VI does not converge to J^* except if started from J^*. PI may get stuck starting from the inferior proper policy
- Case $b < 0$, $J^* = b$: VI converges to J^* if started above J^*, but not if started below J^*. PI can oscillate (if started with u' it generates u, and if started with u it can generate u')

PATHOLOGIES II: BLACKMAILER'S DILEMMA

- Two states, state 1 and the termination state t.
- At state 1, choose $u \in (0, 1]$ (the blackmail amount demanded) at a cost $-u$, and move to t with prob. u^2, or stay in 1 with prob. $1 - u^2$.
- Every stationary policy is proper, but the control set in not finite (also not compact).
- For any stationary μ with $\mu(1) = u$, we have

$$J_\mu(1) = -u + (1 - u^2)J_\mu(1)$$

from which $J_\mu(1) = -\frac{1}{u}$

- Thus $J^*(1) = -\infty$, and there is no optimal stationary policy.
- A nonstationary policy is optimal: demand $\mu_k(1) = \gamma/(k + 1)$ at time k, with $\gamma \in (0, 1/2)$.
  - Blackmailer requests diminishing amounts over time, which add to ∞.
  - The probability of the victim's refusal diminishes at a much faster rate, so the probability that the victim stays forever compliant is strictly positive.

SSP UNDER WEAK CONDITIONS I

- Assume there exists a proper policy, and J^* is real-valued. Let

$$\hat{J}(i) = \min_{\mu: \text{ proper}} J_\mu(i), \qquad i = 1, \ldots, n$$

Note that we may have $\hat{J} \neq J^*$ [i.e., $\hat{J}(i) \neq J^*(i)$ for some i].

- It can be shown that $\hat{J}$ is the unique solution of Bellman's equation within the set $\{J \mid J \geq \hat{J}\}$

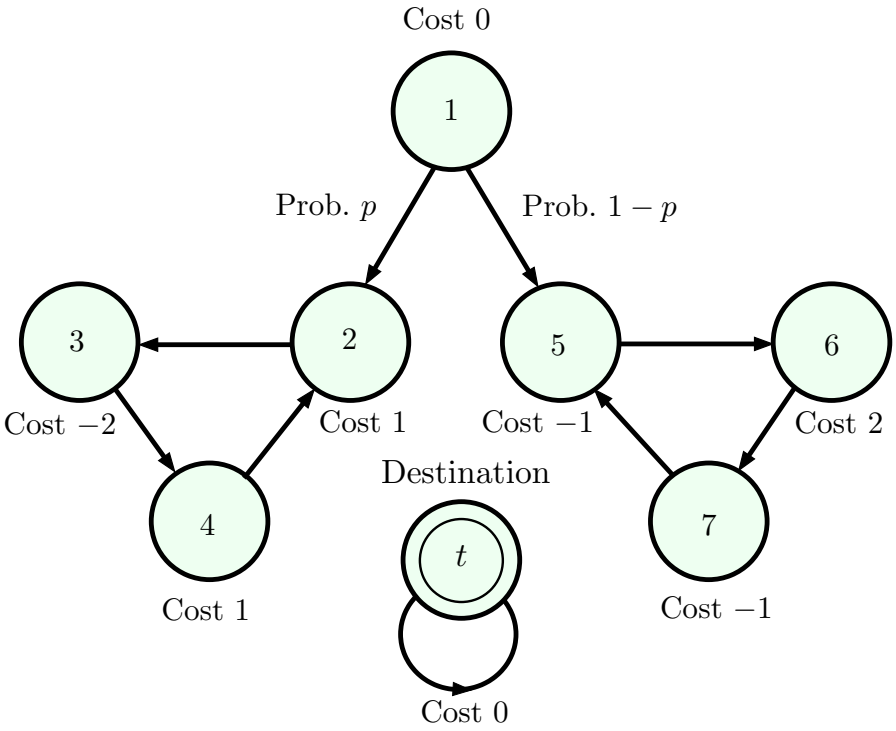
- Also VI converges to $\hat{J}$ starting from any $J \geq \hat{J}$

- The analysis is based on the δ-perturbed problem: adding a small $\delta > 0$ to g. Then:
  - All improper policies have infinite cost for some states in the δ-perturbed problem
  - All proper policies have an additional $O(\delta)$ cost for all states
  - The optimal cost J_δ^* of the δ-perturbed problem converges to $\hat{J}$ as $\delta \downarrow 0$

- There is also a PI method that generates a sequence $\{\mu^k\}$ with $J_{\mu^k} \to \hat{J}$. Uses sequence $\delta_k \downarrow 0$, and policy evaluation based on the δ_k-perturbed problems with $\delta_k \downarrow 0$.

SSP UNDER WEAK CONDITIONS II

- J^* need not be a solution of Bellman's equation! Also J_μ for an improper policy μ.



- For $p = 1/2$, we have

$$J_\mu(1) = 0,\ J_\mu(2) = J_\mu(5) = 1,\ J_\mu(3) = J_\mu(7) = 0,\ J_\mu(4) = J_\mu(6) = 2,$$

Bellman Eq. at state 1, $J_\mu(1) = \frac{1}{2}\big(J_\mu(2)+J_\mu(5)\big)$, is violated.

- References: Bertsekas, D. P., and Yu, H., 2015. "Stochastic Shortest Path Problems Under Weak Conditions," Report LIDS-2909; Math. of OR, to appear. Also the on-line updated Ch. 4 of the text.

6.231 DYNAMIC PROGRAMMING

LECTURE 18

LECTURE OUTLINE

- Undiscounted total cost problems
- Positive and negative cost problems
- Deterministic optimal cost problems
- Adaptive (linear quadratic) DP
- Affine monotonic and risk sensitive problems

Reference:

Updated Chapter 4 of Vol. II of the text:

Noncontractive Total Cost Problems

On-line at:

http://web.mit.edu/dimitrib/www/dpchapter.html

Check for most recent version

CONTRACTIVE/SEMICONTRACTIVE PROBLEMS

- Infinite horizon total cost DP theory divides in
  - "Easy" problems where the results one expects hold (uniqueness of solution of Bellman Eq., convergence of PI and VI, etc)
  - "Difficult" problems where one of more of these results do not hold

- "Easy" problems are characterized by the presence of strong contraction properties in the associated algorithmic maps T and T_μ

- A typical example of an "easy" problem is discounted problems with bounded cost per stage (Chs. 1 and 2 of Voll. II) and some with unbounded cost per stage (Section 1.5 of Voll. II)

- Another is semicontractive problems, where T_μ is a contraction for some μ but is not for other μ, and assumptions are imposed that exclude the "ill-behaved" μ from optimality

- A typical example is SSP where the improper policies are assumed to have infinite cost for some initial states (Chapter 3 of Vol. II)

- In this lecture we go into "difficult" problems

UNDISCOUNTED TOTAL COST PROBLEMS

- Beyond problems with strong contraction properties. One or more of the following hold:
  - No termination state assumed
  - Infinite state and control spaces
  - Either no discounting, or discounting and unbounded cost per stage
  - Risk-sensitivity/exotic cost functions (e.g., SSP problems with exponentiated cost)
- Important classes of problems
  - SSP under weak conditions (e.g., the previous lecture)
  - Positive cost problems (control/regulation, robotics, inventory control)
  - Negative cost problems (maximization of positive rewards - investment, gambling, finance)
  - Deterministic positive cost problems - Adaptive DP
  - A variety of infinite-state problems in queueing, optimal stopping, etc
  - Affine monotonic and risk-sensitive problems (a generalization of SSP)

POS. AND NEG. COST - FORMULATION

- System $x_{k+1} = f(x_k, u_k, w_k)$ and cost

$$J_\pi(x_0) = \lim_{N\to\infty} \underset{\substack{w_k \\ k=0,1,\dots}}{E} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

Discount factor $\alpha \in (0, 1]$, but g may be unbounded

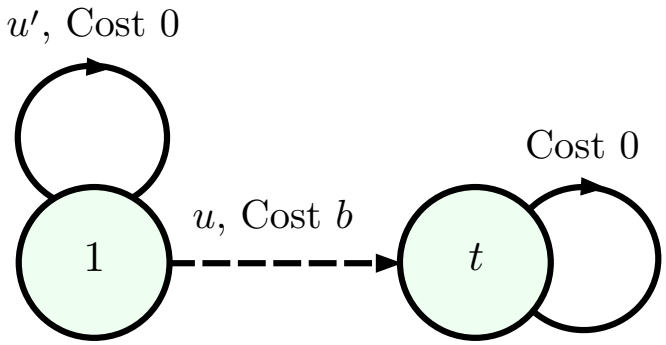
- Case P: $g(x, u, w) \geq 0$ for all (x, u, w)
- Case N: $g(x, u, w) \leq 0$ for all (x, u, w)
- Summary of analytical results:
  - Many of the strong results for discounted and SSP problems fail
  - Analysis more complex; need to allow for J_π and J^* to take values $+\infty$ (under P) or $-\infty$ (under N)
  - However, J^* is a solution of Bellman's Eq. (typically nonunique)
  - Opt. conditions: μ is optimal if and only if $T_\mu J^* = TJ^*$ (P) or if $T_\mu J_\mu = TJ_\mu$ (N)

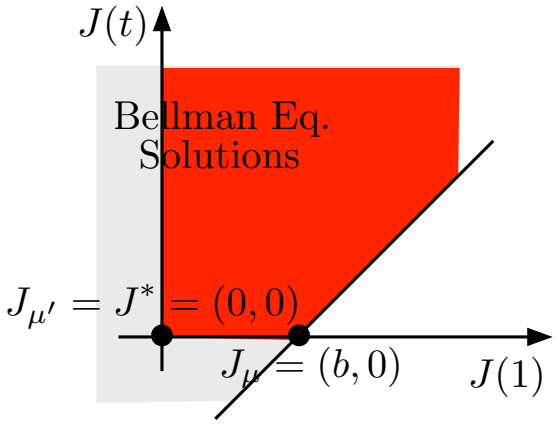
SUMMARY OF ALGORITHMIC RESULTS

- Neither VI nor PI are guaranteed to work
- Behavior of VI
  - P: $T^k J \to J^*$ for all J with $0 \leq J \leq J^*$, if $U(x)$ is finite (or compact plus more conditions - see the text)
  - N: $T^k J \to J^*$ for all J with $J^* \leq J \leq 0$
- Behavior of PI
  - P: J_{μ^k} is monotonically nonincreasing but may get stuck at a nonoptimal policy
  - N: J_{μ^k} may oscillate (but an optimistic form of PI converges to J^* - see the text)
- These anomalies may be mitigated to a greater or lesser extent by exploiting special structure, e.g.
  - Presence of a termination state
  - Proper/improper policy structure in SSP
- Finite-state problems under P can be transformed to equivalent SSP problems by merging (with a simple algorithm) all states x with $J^*(x) = 0$ into a termination state. They can then be solved using the powerful SSP methodology (see updated Ch. 4, Section 4.1.4)

EXAMPLE FROM THE PREVIOUS LECTURE

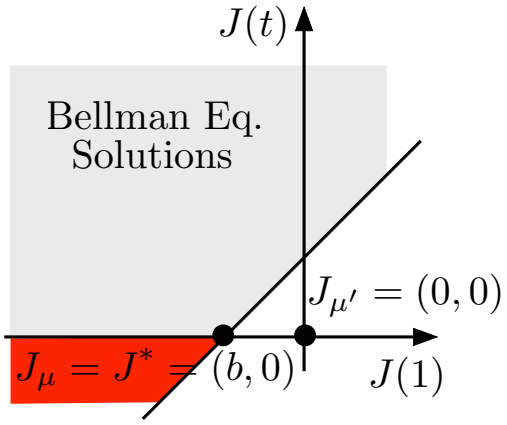
- This is essentially a shortest path example with termination state t





Case P

VI fails starting from
$J(1) \neq 0$, $J(t) = 0$
PI stops at μ



Case N

VI fails starting from
$J(1) < J^*(1)$, $J(t) = 0$
PI oscilllates between μ and μ'

- Bellman Equation:

$$J(1) = \min\big[J(1), b + J(t)\big], \qquad J(t) = J(t)$$

DETERM. OPT. CONTROL - FORMULATION

- System: $x_{k+1} = f(x_k, u_k)$, arbitrary state and control spaces X and U
- Cost positivity: $0 \leq g(x, u)$, $\forall\ x \in X,\ u \in U(x)$
- No discounting:

$$J_\pi(x_0) = \lim_{N \to \infty} \sum_{k=0}^{N-1} g\big(x_k, \mu_k(x_k)\big)$$

- "Goal set of states" X_0
  - All $x \in X_0$ are cost-free and absorbing
- A shortest path-type problem, but with possibly infinite number of states
- A common formulation of control/regulation and planning/robotics problems
- Example: Linear system, quadratic cost (possibly with state and control constraints), $X_0 = \{0\}$ or X_0 is a small set around 0
- Strong analytical and computational results

DETERM. OPT. CONTROL - ANALYSIS

- Bellman's Eq. holds (for not only this problem, but also all deterministic total cost problems)

$$J^*(x) = \min_{u \in U(x)} \big\{ g(x,u) + J^*\big(f(x,u)\big) \big\}, \quad \forall\, x \in X$$

- Definition: A policy π terminates starting from x if the state sequence $\{x_k\}$ generated starting from $x_0 = x$ and using π reaches X_0 in finite time, i.e., satisfies $x_{\bar{k}} \in X_0$ for some index $\bar{k}$

- Assumptions: The cost structure is such that
  - $J^*(x) > 0$, $\forall\, x \notin X_0$ (termination incentive)
  - For every x with $J^*(x) < \infty$ and every $\epsilon > 0$, there exists a policy π that terminates starting from x and satisfies $J_\pi(x) \leq J^*(x) + \epsilon$.

- Uniqueness of solution of Bellman's Eq.: J^* is the unique solution within the set

$$\mathcal{J} = \big\{ J \mid 0 \leq J(x) \leq \infty,\ \forall\, x \in X,\ J(x) = 0,\ \forall\, x \in X_0 \big\}$$

- Counterexamples: Earlier SP problem. Also linear quadratic problems where the Riccati equation has two solutions (observability not satisfied).

DET. OPT. CONTROL - VI/PI CONVERGENCE

- The sequence $\{T^k J\}$ generated by VI starting from a $J \in \mathcal{J}$ with $J \geq J^*$ converges to J^*
- If in addition $U(x)$ is finite (or compact plus more conditions - see the text), the sequence $\{T^k J\}$ generated by VI starting from any function $J \in \mathcal{J}$ converges to J^*
- A sequence $\{J_{\mu^k}\}$ generated by PI satisfies $J_{\mu^k}(x) \downarrow J^*(x)$ for all $x \in X$
- PI counterexample: The earlier SP example
- Optimistic PI algorithm: Generates pairs $\{J_k, \mu^k\}$ as follows: Given J_k, we generate μ^k according to

$$\mu^k(x) = \arg \min_{u \in U(x)} \big\{ g(x,u) + J_k\big(f(x,u)\big) \big\}, \quad x \in X$$

and obtain J_{k+1} with $m_k \geq 1$ VIs using μ^k:

$$J_{k+1}(x_0) = J_k(x_{m_k}) + \sum_{t=0}^{m_k - 1} g\big(x_t, \mu^k(x_t)\big), \quad x_0 \in X$$

If $J_0 \in \mathcal{J}$ and $J_0 \geq T J_0$, we have $J_k \downarrow J^*$.

- Rollout with terminating heuristic (e.g., MPC).

LINEAR-QUADRATIC ADAPTIVE CONTROL

- System: $x_{k+1} = Ax_k + Bu_k, \quad x_k \in \Re^n, u_k \in \Re^m$
- Cost: $\sum_{k=0}^{\infty}(x_k' Q x_k + u_k' R u_k)$, $Q \geq 0$, $R > 0$
- Optimal policy is linear: $\mu^*(x) = Lx$
- The Q-factor of each linear policy μ is quadratic:

$$Q_\mu(x,u) = \begin{pmatrix} x' & u' \end{pmatrix} K_\mu \begin{pmatrix} x \\ u \end{pmatrix} \qquad (*)$$

- We will consider A and B unknown
- We use as basis functions all the quadratic functions involving state and control components

$$x^i x^j, \qquad u^i u^j, \qquad x^i u^j, \qquad \forall\ i,j$$

These form the "rows" $\phi(x,u)'$ of a matrix Φ

- The Q-factor Q_μ of a linear policy μ can be exactly represented within the subspace spanned by the basis functions:

$$Q_\mu(x,u) = \phi(x,u)' r_\mu$$

where r_μ consists of the components of K_μ in (*)

- Key point: Compute r_μ by simulation of μ (Q-factor evaluation by simulation, in a PI scheme)

PI FOR LINEAR-QUADRATIC PROBLEM

- Policy evaluation: r_μ is found (exactly) by least squares minimization

$$\min_r \sum_{(x_k,u_k)} \Big|\phi(x_k,u_k)'r - \big(x_k'Qx_k + u_k'Ru_k + \phi\big(x_{k+1},\mu(x_{k+1})\big)'r\big)\Big|^2$$

where (x_k, u_k, x_{k+1}) are "enough" samples generated by the system or a simulator of the system.

- Policy improvement:

$$\overline{\mu}(x) \in \arg\min_u \big(\phi(x,u)'r_\mu\big)$$

- Knowledge of A and B is not required

- If the policy evaluation is done exactly, this becomes exact PI, and convergence to an optimal policy can be shown

- The basic idea of this example has been generalized and forms the starting point of the field of adaptive DP

- This field deals with adaptive control of continuous-space (possibly nonlinear) dynamic systems, in both discrete and continuous time

FINITE-STATE AFFINE MONOTONIC PROBLEMS

- Generalization of positive cost finite-state stochastic total cost problems where:
  - In place of a transition prob. matrix P_μ, we have a general matrix $A_\mu \geq 0$
  - In place of 0 terminal cost function, we have a more general terminal cost function $\bar{J} \geq 0$
- Mappings

$$T_\mu J = b_\mu + A_\mu J, \qquad (TJ)(i) = \min_{\mu \in \mathcal{M}} (T_\mu J)(i)$$

- Cost function of $\pi = \{\mu_0, \mu_1, \ldots\}$

$$J_\pi(i) = \limsup_{N \to \infty} (T_{\mu_0} \cdots T_{\mu_{N-1}} \bar{J})(i), \quad i = 1, \ldots, n$$

- Special case: An SSP with an exponential risk-sensitive cost, where for all i and $u \in U(i)$

$$A_{ij}(u) = p_{ij}(u) e^{g(i,u,j)}, \quad b(i,u) = p_{it}(u) e^{g(i,u,t)}$$

- Interpretation:

$$J_\pi(i) = E\{e^{(\text{length of path of } \pi \text{ starting from } i)}\}$$

AFFINE MONOTONIC PROBLEMS: ANALYSIS

- The analysis follows the lines of analysis of SSP
- Key notion (generalizes the notion of a proper policy in SSP): A policy μ is stable if $A_\mu^k \to 0$; else it is called unstable
- We have

$$T_\mu^N J = A_\mu^N J + \sum_{k=0}^{N-1} A_\mu^k b_\mu, \quad \forall\, J \in \Re^n,\ N = 1, 2, \ldots,$$

- For a stable policy μ, we have for all $J \in \Re^n$

$$J_\mu = \limsup_{N\to\infty} T_\mu^N J = \limsup_{N\to\infty} \sum_{k=0}^{\infty} A_\mu^k b_\mu = (I - A_\mu)^{-1} b_\mu$$

- Consider the following assumptions:

  (1) There exists at least one stable policy

  (2) For every unstable policy μ, at least one component of $\sum_{k=0}^{\infty} A_\mu^k b_\mu$ is equal to ∞

- Under (1) and (2) the strong SSP analytical and algorithmic theory generalizes
- Under just (1) the weak SSP theory generalizes.

6.231 DYNAMIC PROGRAMMING

LECTURE 19

LECTURE OUTLINE

- We begin a lecture series on approximate DP.
- Reading: Chapters 6 and 7, Vol. 2 of the text.
- Today we discuss some general issues about approximation and simulation
- We classify/overview the main approaches:
  - Approximation in policy space (policy parametrization, gradient methods, random search)
  - Approximation in value space (approximate PI, approximate VI, Q-Learning, Bellman error approach, approximate LP)
  - Rollout/Simulation-based single policy iteration (will not discuss this further)
  - Approximation in value space using problem approximation (simplification - forms of aggregation - limited lookahead) - will not discuss much

GENERAL ORIENTATION TO ADP

- ADP (late 80s - present) is a breakthrough methodology that allows the application of DP to problems with many or infinite number of states.
- Other names for ADP are:
  - "reinforcement learning" (RL)
  - "neuro-dynamic programming" (NDP)
- We will mainly adopt an n-state discounted model (the easiest case - but think of HUGE n).
- Extensions to other DP models (continuous space, continuous-time, not discounted) are possible (but more quirky). We will set aside for later.
- There are many approaches:
  - Problem approximation and 1-step lookahead
  - Simulation-based approaches (we will focus on these)
- Simulation-based methods are of three types:
  - Rollout (we will not discuss further)
  - Approximation in policy space
  - Approximation in value space

WHY DO WE USE SIMULATION?

- One reason: Computational complexity advantage in computing expected values and sums/inner products involving a very large number of terms
  - Speeds up linear algebra: Any sum $\sum_{i=1}^{n} a_i$ can be written as an expected value

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} \xi_i \frac{a_i}{\xi_i} = E_\xi \left\{ \frac{a_i}{\xi_i} \right\},$$

    where ξ is any prob. distribution over $\{1, \ldots, n\}$
  - It is approximated by generating many samples $\{i_1, \ldots, i_k\}$ from $\{1, \ldots, n\}$, according to ξ, and Monte Carlo averaging:

$$\sum_{i=1}^{n} a_i = E_\xi \left\{ \frac{a_i}{\xi_i} \right\} \approx \frac{1}{k} \sum_{t=1}^{k} \frac{a_{i_t}}{\xi_{i_t}}$$

  - Choice of ξ makes a difference. Importance sampling methodology.

- Simulation is also convenient when an analytical model of the system is unavailable, but a simulation/computer model is possible.

APPROXIMATION IN POLICY SPACE

- A brief discussion; we will return to it later.
- Use parametrization $\mu(i;r)$ of policies with a vector $r = (r_1, \ldots, r_s)$. Examples:
  - Polynomial, e.g., $\mu(i;r) = r_1 + r_2 \cdot i + r_3 \cdot i^2$
  - Multi-warehouse inventory system: $\mu(i;r)$ is threshold policy with thresholds $r = (r_1, \ldots, r_s)$
- Optimize the cost over r. For example:
  - Each value of r defines a stationary policy, with cost starting at state i denoted by $\tilde{J}(i;r)$.
  - Let $(p_1, \ldots, p_n)$ be some probability distribution over the states, and minimize over r

$$\sum_{i=1}^{n} p_i \tilde{J}(i;r)$$

  - Use a random search, gradient, or other method
- A special case: The parameterization of the policies is indirect, through a cost approximation architecture $\hat{J}$, i.e.,

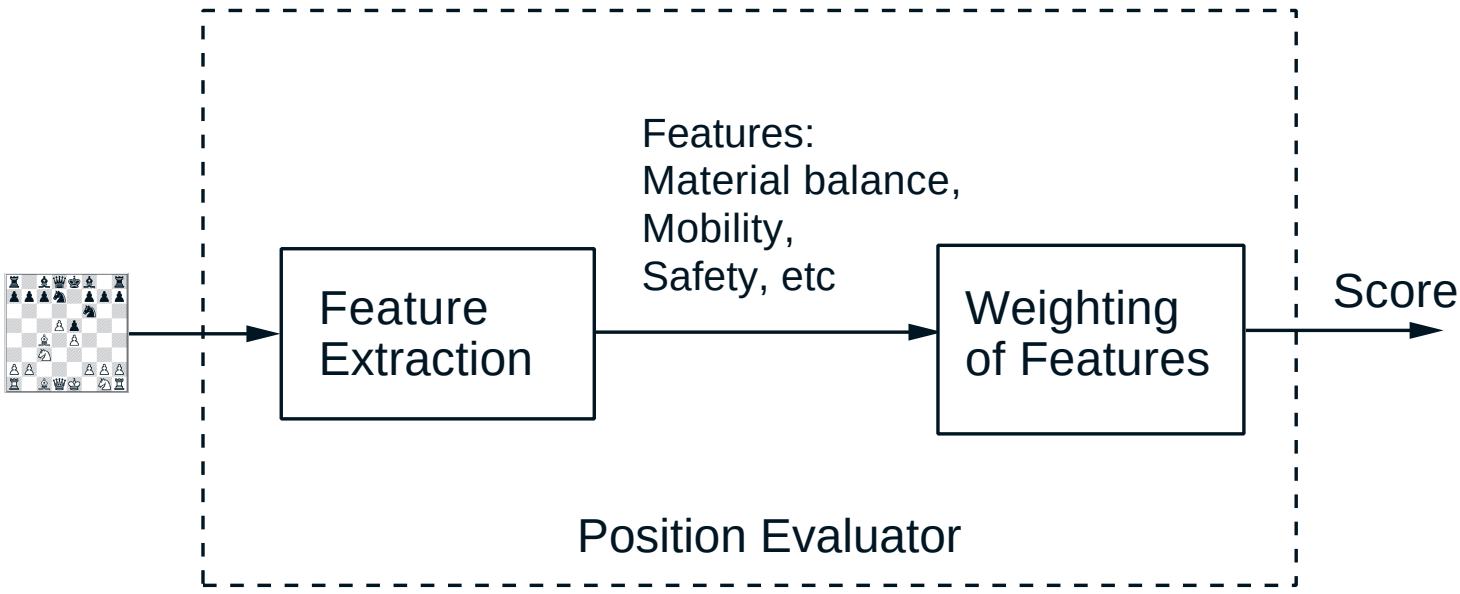
$$\mu(i;r) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \hat{J}(j;r)\big)$$

APPROXIMATION IN VALUE SPACE

- Approximate J^* or J_μ from a parametric class $\tilde{J}(i;r)$ where i is the current state and $r = (r_1, \ldots, r_m)$ is a vector of "tunable" scalars weights
- Use $\tilde{J}$ in place of J^* or J_μ in various algorithms and computations (VI, PI, LP)
- Role of r: By adjusting r we can change the "shape" of $\tilde{J}$ so that it is "close" to J^* or J_μ
- Two key issues:
  - The choice of parametric class $\tilde{J}(i;r)$ (the approximation architecture)
  - Method for tuning the weights ("training" the architecture)
- Success depends strongly on how these issues are handled ... also on insight about the problem
- A simulator may be used, particularly when there is no mathematical model of the system
- We will focus on simulation, but this is not the only possibility
- We may also use parametric approximation for Q-factors

APPROXIMATION ARCHITECTURES

- Divided in linear and nonlinear [i.e., linear or nonlinear dependence of $\tilde{J}(i;r)$ on r]
- Linear architectures are easier to train, but nonlinear ones (e.g., neural networks) are richer
- Computer chess example:
  - Think of board position as state and move as control
  - Uses a feature-based position evaluator that assigns a score (or approximate Q-factor) to each position/move



- Relatively few special features and weights, and multistep lookahead

LINEAR APPROXIMATION ARCHITECTURES

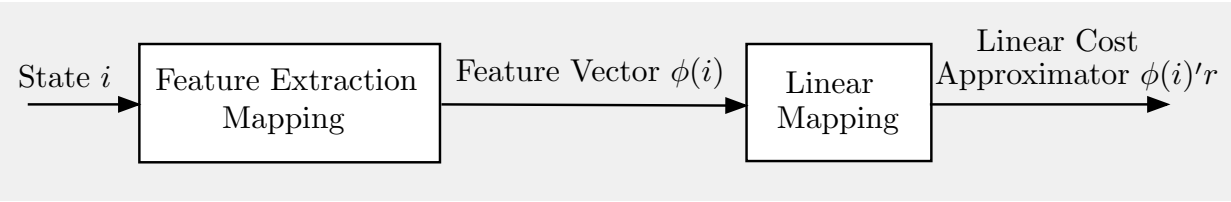
- Often, the features encode much of the nonlinearity inherent in the cost function approximated

- Then the approximation may be quite accurate without a complicated architecture. (Extreme example: The ideal feature is the true cost function)

- With well-chosen features, we can use a linear architecture:

$$\tilde{J}(i;r) = \phi(i)'r, \ \forall \ i \quad \text{or} \quad \tilde{J}(r) = \Phi r = \sum_{j=1}^{s} \Phi_j r_j$$

Φ: the matrix whose rows are $\phi(i)'$, $i = 1, \ldots, n$, Φ_j is the jth column of Φ



- This is approximation on the subspace

$$S = \{\Phi r \mid r \in \Re^s\}$$

spanned by the columns of Φ (basis functions)

- Many examples of feature types: Polynomial approximation, radial basis functions, domain specific, etc

ILLUSTRATIONS: POLYNOMIAL TYPE

• Polynomial Approximation, e.g., a quadratic approximating function. Let the state be $i = (i_1, \ldots, i_q)$ (i.e., have q "dimensions") and define

$$\phi_0(i) = 1, \quad \phi_k(i) = i_k, \quad \phi_{km}(i) = i_k i_m, \quad k, m = 1, \ldots, q$$

Linear approximation architecture:

$$\tilde{J}(i; r) = r_0 + \sum_{k=1}^{q} r_k i_k + \sum_{k=1}^{q} \sum_{m=k}^{q} r_{km} i_k i_m,$$

where r has components r_0, r_k, and r_{km}.

• Interpolation: A subset I of special/representative states is selected, and the parameter vector r has one component r_i per state $i \in I$. The approximating function is

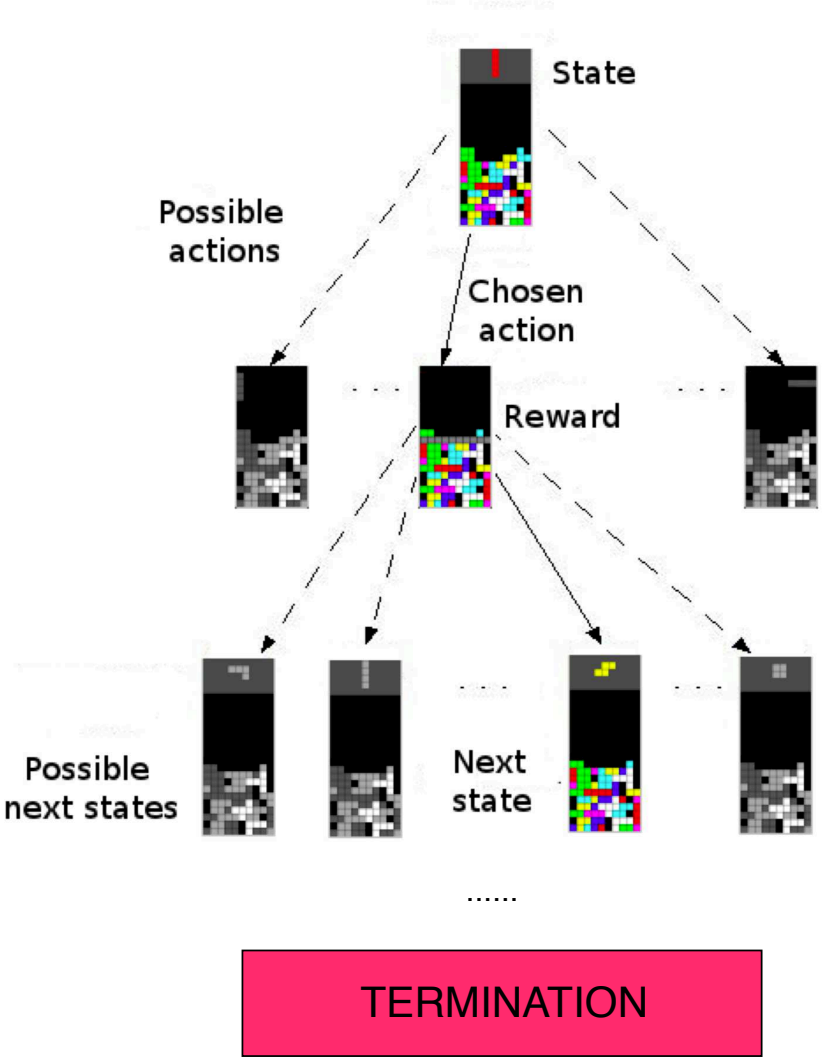
$$\tilde{J}(i; r) = r_i, \qquad i \in I,$$

$\tilde{J}(i; r) =$ interpolation using the values at $i \in I$, $i \notin I$

For example, piecewise constant, piecewise linear, more general polynomial interpolations.

A DOMAIN SPECIFIC EXAMPLE

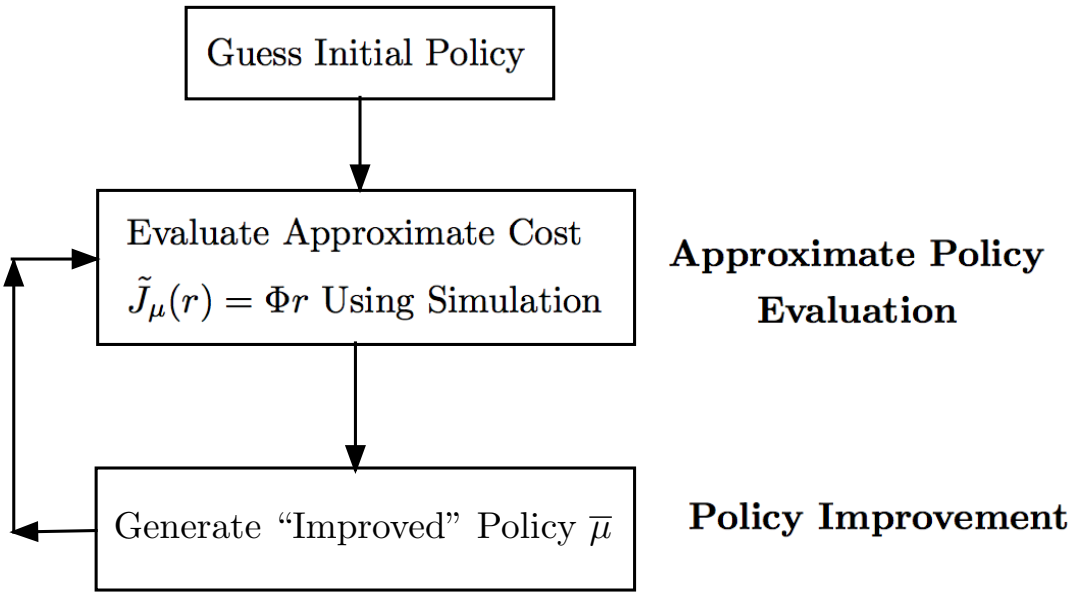
- Tetris game (used as testbed in competitions)



- $J^*(i)$: optimal score starting from position i
- Number of states $> 2^{200}$ (for 10×20 board)
- Success with just 22 features, readily recognized by tetris players as capturing important aspects of the board position (heights of columns, etc)

APPROX. PI - OPTION TO APPROX. J_μ OR Q_μ

- Use simulation to approximate the cost J_μ of the current policy μ
- Generate "improved" policy $\overline{\mu}$ by minimizing in (approx.) Bellman equation



- Altenatively approximate the Q-factors of μ
- A survey reference: D. P. Bertsekas, "Approximate Policy Iteration: A Survey and Some New Methods," J. of Control Theory and Appl., Vol. 9, 2011, pp. 310-335.

DIRECTLY APPROXIMATING J^* OR Q^*

- Approximation of the optimal cost function J^* directly (without PI)
  - Q-Learning: Use a simulation algorithm to approximate the Q-factors

$$Q^*(i,u) = g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J^*(j);$$

    and the optimal costs

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u)$$

  - Bellman Error approach: Find r to

$$\min_r E_i \Big\{ \big(\tilde{J}(i;r) - (T\tilde{J})(i;r) \big)^2 \Big\}$$

    where $E_i\{\cdot\}$ is taken with respect to some distribution over the states
  - Approximate Linear Programming (we will not discuss here)

- Q-learning can also be used with approximations

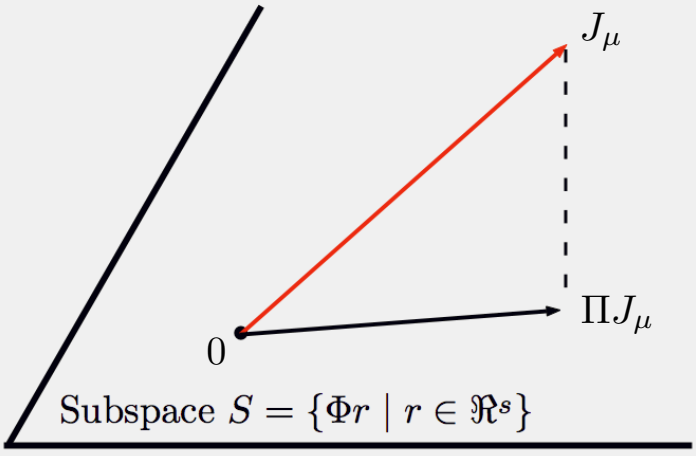
- Q-learning and Bellman error approach can also be used for policy evaluation

DIRECT POLICY EVALUATION

- Can be combined with regular and optimistic policy iteration
- Find r that minimizes $\|J_\mu - \tilde{J}(\cdot, r)\|_\xi^2$, i.e.,

$$\sum_{i=1}^{n} \xi_i \big(J_\mu(i) - \tilde{J}(i,r)\big)^2, \qquad \xi_i\text{: some pos. weights}$$

- Nonlinear architectures may be used
- The linear architecture case: Amounts to projection of J_μ onto the approximation subspace



Direct Method: Projection of cost vector J_μ

- Solution by linear least squares methods

POLICY EVALUATION BY SIMULATION

- Projection by Monte Carlo Simulation: Compute the projection ΠJ_μ of J_μ on subspace $S = \{\Phi r \mid r \in \Re^s\}$, with respect to a weighted Euclidean norm $\|\cdot\|_\xi$
- Equivalently, find Φr^*, where

$$r^* = \arg\min_{r\in\Re^s} \|\Phi r - J_\mu\|_\xi^2 = \arg\min_{r\in\Re^s} \sum_{i=1}^n \xi_i \big(J_\mu(i) - \phi(i)'r\big)^2$$

- Setting to 0 the gradient at r^*,

$$r^* = \left(\sum_{i=1}^n \xi_i \phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^n \xi_i \phi(i) J_\mu(i)$$

- Generate samples $\big\{(i_1, J_\mu(i_1)), \ldots, (i_k, J_\mu(i_k))\big\}$ using distribution ξ
- Approximate by Monte Carlo the two "expected values" with low-dimensional calculations

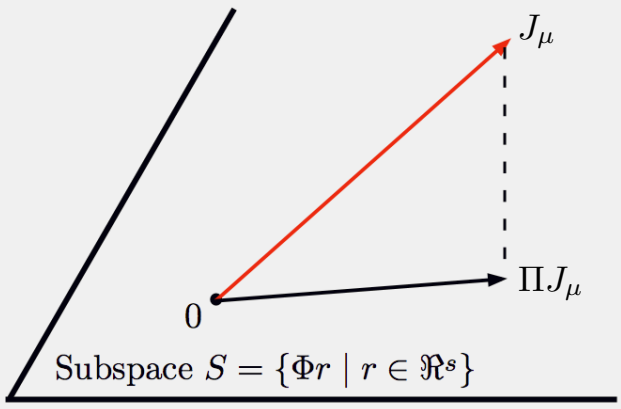
$$\hat{r}_k = \left(\sum_{t=1}^k \phi(i_t)\phi(i_t)'\right)^{-1} \sum_{t=1}^k \phi(i_t) J_\mu(i_t)$$

- Equivalent least squares alternative calculation:

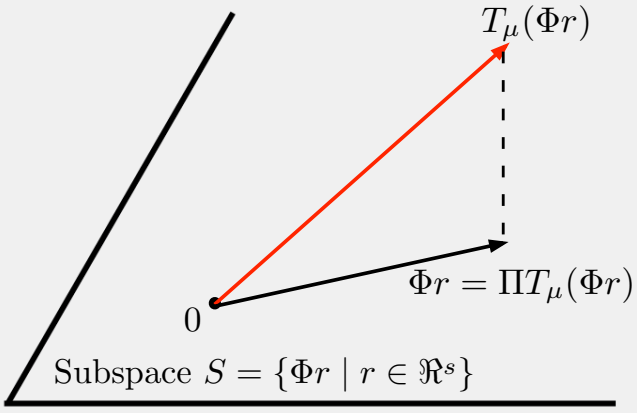
$$\hat{r}_k = \arg\min_{r\in\Re^s} \sum_{t=1}^k \big(\phi(i_t)'r - J_\mu(i_t)\big)^2$$

INDIRECT POLICY EVALUATION

- An example: Solve the projected equation $\Phi r = \Pi T_\mu(\Phi r)$ where Π is projection w/ respect to a suitable weighted Euclidean norm (Galerkin approx.



Direct Method: Projection of cost vector J_μ



Indirect Method: Solving a projected form of Bellman's equation

- Solution methods that use simulation (to manage the calculation of Π)
  - TD(λ): Stochastic iterative algorithm for solving $\Phi r = \Pi T_\mu(\Phi r)$
  - LSTD(λ): Solves a simulation-based approximation w/ a standard solver
  - LSPE(λ): A simulation-based form of projected value iteration; essentially

$$\Phi r_{k+1} = \Pi T_\mu(\Phi r_k) + \text{ simulation noise}$$

BELLMAN EQUATION ERROR METHODS

- Another example of indirect approximate policy evaluation:

$$\min_r \|\Phi r - T_\mu(\Phi r)\|_\xi^2 \qquad (*)$$

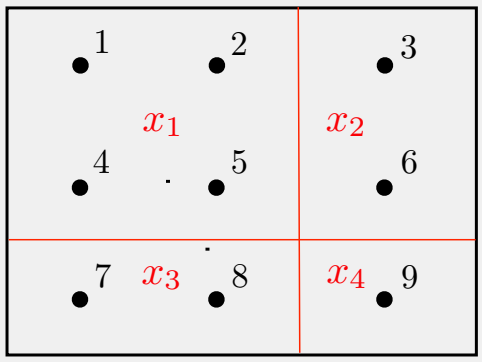
where $\|\cdot\|_\xi$ is Euclidean norm, weighted with respect to some distribution ξ

- It is closely related to the projected equation approach (with a special choice of projection norm)

- Several ways to implement projected equation and Bellman error methods by simulation. They involve:
  - Generating many random samples of states i_k using the distribution ξ
  - Generating many samples of transitions (i_k, j_k) using the policy μ
  - Form a simulation-based approximation of the optimality condition for projection problem or problem (*) (use sample averages in place of inner products)
  - Solve the Monte-Carlo approximation of the optimality condition
- Issues for indirect methods: How to generate the samples? How to calculate r^* efficiently?

ANOTHER INDIRECT METHOD: AGGREGATION

- An example: Group similar states together into "aggregate states" $x_1, \ldots, x_s$; assign a common cost r_i to each group x_i. A linear architecture called hard aggregation.



$$\Phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

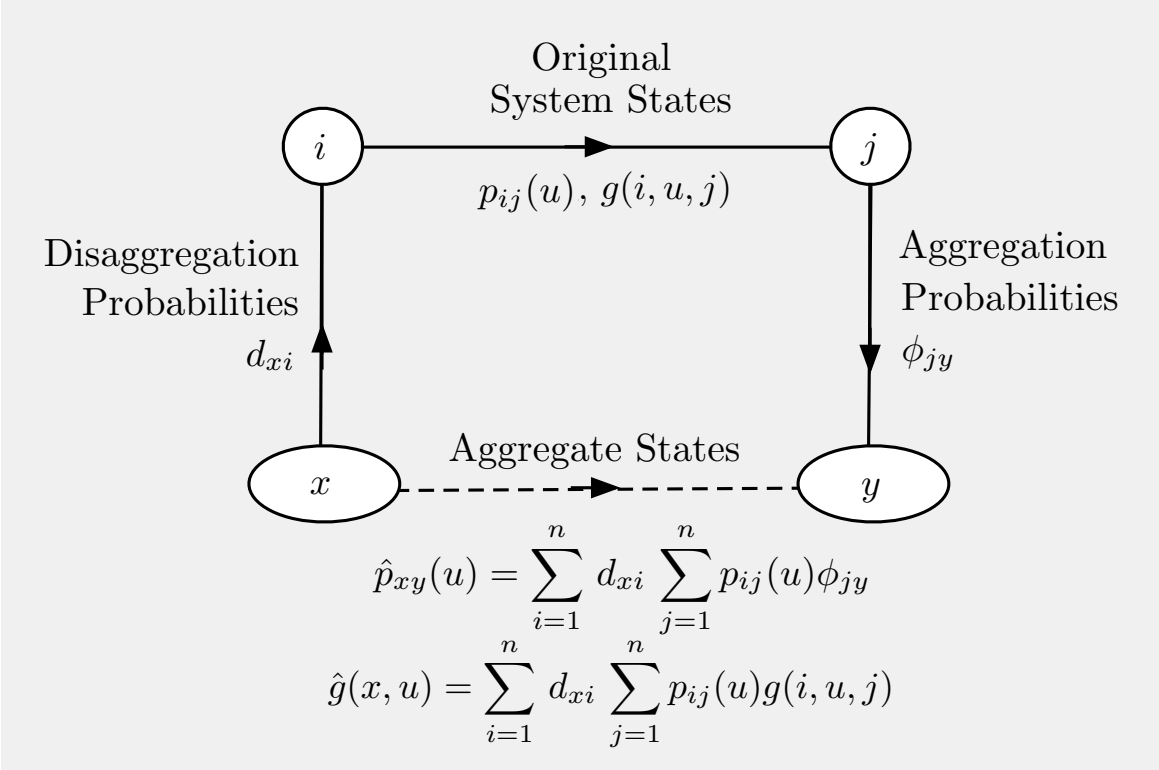
- Solve an "aggregate" DP problem to obtain $r = (r_1, \ldots, r_s)$.
- More general/mathematical view: Solve

$$\Phi r = \Phi D T_\mu(\Phi r)$$

where the rows of D and Φ are prob. distributions (e.g., D and Φ "aggregate" rows and columns of the linear system $J = T_\mu J$)

- Compare with projected equation $\Phi r = \Pi T_\mu(\Phi r)$. Note: ΦD is a projection in some interesting cases

AGGREGATION AS PROBLEM APPROXIMATION



- Aggregation can be viewed as a systematic approach for problem approx. Main elements:
  - Solve (exactly or approximately) the "aggregate" problem by any kind of VI or PI method (including simulation-based methods)
  - Use the optimal cost of the aggregate problem to approximate the optimal cost of the original problem

- Because an exact PI algorithm is used to solve the approximate/aggregate problem the method behaves more regularly than the projected equation approach

THEORETICAL BASIS OF APPROXIMATE PI

- If policies are approximately evaluated using an approximation architecture such that

$$\max_i |\tilde{J}(i, r_k) - J_{\mu^k}(i)| \leq \delta, \qquad k = 0, 1, \ldots$$

- If policy improvement is also approximate,

$$\max_i |(T_{\mu^{k+1}} \tilde{J})(i, r_k) - (T\tilde{J})(i, r_k)| \leq \epsilon, \qquad k = 0, 1, \ldots$$

- Error bound: The sequence $\{\mu^k\}$ generated by approximate policy iteration satisfies

$$\limsup_{k \to \infty} \max_i \left(J_{\mu^k}(i) - J^*(i)\right) \leq \frac{\epsilon + 2\alpha\delta}{(1-\alpha)^2}$$

- Typical practical behavior: The method makes steady progress up to a point and then the iterates J_{μ^k} oscillate within a neighborhood of J^*.
- Oscillations are quite unpredictable.
  - Bad examples of oscillations are known.
  - In practice oscillations between policies is probably not the major concern.
  - In aggregation case, there are no oscillations

THE ISSUE OF EXPLORATION

- To evaluate a policy μ, we need to generate cost samples using that policy - this biases the simulation by underrepresenting states that are unlikely to occur under μ
- Cost-to-go estimates of underrepresented states may be highly inaccurate
- This seriously impacts the improved policy $\overline{\mu}$
- This is known as inadequate exploration - a particularly acute difficulty when the randomness embodied in the transition probabilities is "relatively small" (e.g., a deterministic system)
- Some remedies:
  - Frequently restart the simulation and ensure that the initial states employed form a rich and representative subset
  - Occasionally generate transitions that use a randomly selected control rather than the one dictated by the policy μ
  - Other methods: Use two Markov chains (one is the chain of the policy and is used to generate the transition sequence, the other is used to generate the state sequence).

APPROXIMATING Q-FACTORS

- Given $\tilde{J}(i;r)$, policy improvement requires a model [knowledge of $p_{ij}(u)$ for all $u \in U(i)$]
- Model-free alternative: Approximate Q-factors

$$\tilde{Q}(i,u;r) \approx \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J_\mu(j)\big)$$

and use for policy improvement the minimization

$$\overline{\mu}(i) \in \arg\min_{u \in U(i)} \tilde{Q}(i,u;r)$$

- r is an adjustable parameter vector and $\tilde{Q}(i,u;r)$ is a parametric architecture, such as

$$\tilde{Q}(i,u;r) = \sum_{m=1}^{s} r_m \phi_m(i,u)$$

- We can adapt any of the cost approximation approaches, e.g., projected equations, aggregation
- Use the Markov chain with states (i,u), so $p_{ij}(\mu(i))$ is the transition prob. to $(j,\mu(i))$, 0 to other (j,u')
- Major concern: Acutely diminished exploration

STOCHASTIC ALGORITHMS: GENERALITIES

- Consider solution of a linear equation $x = b + Ax$ by using m simulation samples $b + w_k$ and $A + W_k$, $k = 1, \ldots, m$, where w_k, W_k are random, e.g., "simulation noise"

- Think of $x = b + Ax$ as approximate policy evaluation (projected or aggregation equations)

- Stoch. approx. (SA) approach: For $k = 1, \ldots, m$

$$x_{k+1} = (1 - \gamma_k)x_k + \gamma_k\big((b + w_k) + (A + W_k)x_k\big)$$

- Monte Carlo estimation (MCE) approach: Form Monte Carlo estimates of b and A

$$b_m = \frac{1}{m}\sum_{k=1}^{m}(b + w_k), \qquad A_m = \frac{1}{m}\sum_{k=1}^{m}(A + W_k)$$

Then solve $x = b_m + A_m x$ by matrix inversion

$$x_m = (1 - A_m)^{-1} b_m$$

or iteratively

- TD(λ) and Q-learning are SA methods

- LSTD(λ) and LSPE(λ) are MCE methods

6.231 DYNAMIC PROGRAMMING

LECTURE 20

LECTURE OUTLINE

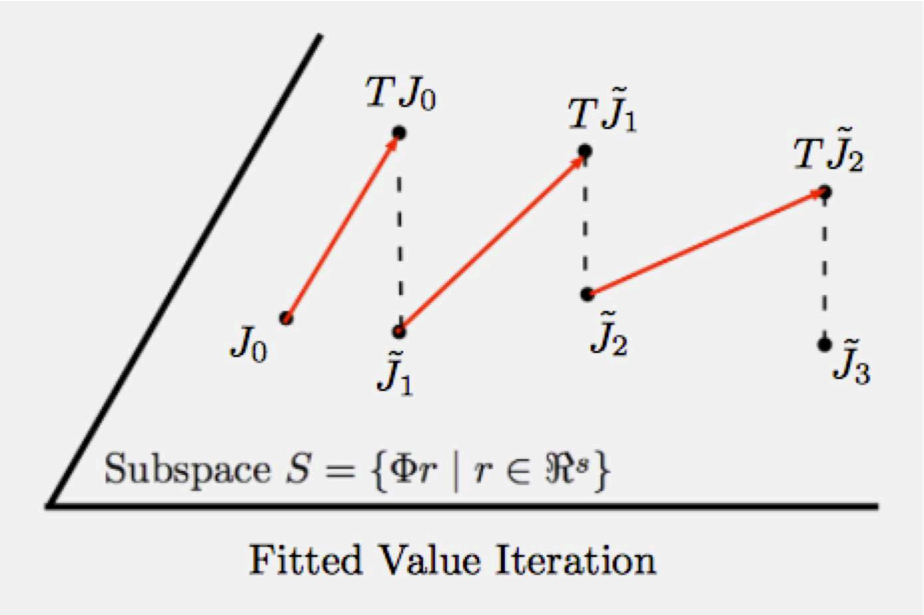
- Discounted problems - Approximation on subspace $\{\Phi r \mid r \in \Re^s\}$
- Approximate (fitted) VI
- Approximate PI
- The projected equation
- Contraction properties - Error bounds
- Matrix form of the projected equation
- Simulation-based implementation
- LSTD and LSPE methods

REVIEW: APPROXIMATION IN VALUE SPACE

- Finite-spaces discounted problems: Defined by mappings T_μ and T ($TJ = \min_\mu T_\mu J$).
- Exact methods:
  - VI: $J_{k+1} = TJ_k$
  - PI: $J_{\mu^k} = T_{\mu^k} J_{\mu^k}$, $\quad T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$
  - LP: $\min_J c'J$ subject to $J \leq TJ$
- Approximate versions: Plug-in subspace approximation with Φr in place of J
  - VI: $\Phi r_{k+1} \approx T\Phi r_k$
  - PI: $\Phi r_k \approx T_{\mu^k} \Phi r_k$, $\quad T_{\mu^{k+1}} \Phi r_k = T\Phi r_k$
  - LP: $\min_r c'\Phi r$ subject to $\Phi r \leq T\Phi r$
- Approx. onto subspace $S = \{\Phi r \mid r \in \Re^s\}$ is often done by projection with respect to some (weighted) Euclidean norm.
- Another possibility is aggregation. Here:
  - The rows of Φ are probability distributions
  - $\Phi r \approx J_\mu$ or $\Phi r \approx J^*$, with r the solution of an "aggregate Bellman equation" $r = DT_\mu(\Phi r)$ or $r = DT(\Phi r)$, where the rows of D are probability distributions

APPROXIMATE (FITTED) VI

- Approximates sequentially $J_k(i) = (T^k J_0)(i)$, $k = 1, 2, \ldots$, with $\tilde{J}_k(i; r_k)$
- The starting function J_0 is given (e.g., $J_0 \equiv 0$)
- Approximate (Fitted) Value Iteration: A sequential "fit" to produce $\tilde{J}_{k+1}$ from $\tilde{J}_k$, i.e., $\tilde{J}_{k+1} \approx T\tilde{J}_k$ or (for a single policy μ) $\tilde{J}_{k+1} \approx T_\mu \tilde{J}_k$



Fitted Value Iteration

- After a large enough number N of steps, $\tilde{J}_N(i; r_N)$ is used as approximation to $J^*(i)$
- Possibly use (approximate) projection Π with respect to some projection norm,

$$\tilde{J}_{k+1} \approx \Pi T \tilde{J}_k$$

WEIGHTED EUCLIDEAN PROJECTIONS

- Consider a weighted Euclidean norm

$$\|J\|_\xi = \sqrt{\sum_{i=1}^{n} \xi_i \big(J(i)\big)^2},$$

where $\xi = (\xi_1, \ldots, \xi_n)$ is a positive distribution ($\xi_i > 0$ for all i).

- Let Π denote the projection operation onto

$$S = \{\Phi r \mid r \in \Re^s\}$$

with respect to this norm, i.e., for any $J \in \Re^n$,

$$\Pi J = \Phi r^*$$

where

$$r^* = \arg\min_{r \in \Re^s} \|\Phi r - J\|_\xi^2$$

- Recall that weighted Euclidean projection can be implemented by simulation and least squares, i.e., sampling $J(i)$ according to ξ and solving

$$\min_{r \in \Re^s} \sum_{t=1}^{k} \big(\phi(i_t)'r - J(i_t)\big)^2$$

FITTED VI - NAIVE IMPLEMENTATION

- Select/sample a "small" subset I_k of representative states
- For each $i \in I_k$, given $\tilde{J}_k$, compute

$$(T\tilde{J}_k)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}_k(j;r)\big)$$

- "Fit" the function $\tilde{J}_{k+1}(i;r_{k+1})$ to the "small" set of values $(T\tilde{J}_k)(i)$, $i \in I_k$ (for example use some form of approximate projection)
- "Model-free" implementation by simulation
- Error Bound: If the fit is uniformly accurate within $\delta > 0$, i.e.,

$$\max_i |\tilde{J}_{k+1}(i) - T\tilde{J}_k(i)| \leq \delta,$$

then

$$\limsup_{k \to \infty} \max_{i=1,\ldots,n} \big(\tilde{J}_k(i,r_k) - J^*(i)\big) \leq \frac{\delta}{1-\alpha}$$

- But there is a potential serious problem!

AN EXAMPLE OF FAILURE

- Consider two-state discounted MDP with states 1 and 2, and a single policy.
  - Deterministic transitions: $1 \to 2$ and $2 \to 2$
  - Transition costs $\equiv 0$, so $J^*(1) = J^*(2) = 0$.
- Consider (exact) fitted VI scheme that approximates cost functions within $S = \{(r, 2r) \mid r \in \Re\}$ with a weighted least squares fit; here $\Phi = (\,1, 2\,)'$
- Given $\tilde{J}_k = (r_k, 2r_k)$, we find $\tilde{J}_{k+1} = (r_{k+1}, 2r_{k+1})$, where $\tilde{J}_{k+1} = \Pi_\xi(T\tilde{J}_k)$, with weights $\xi = (\xi_1, \xi_2)$:

$$r_{k+1} = \arg\min_r \Big[\xi_1\big(r-(T\tilde{J}_k)(1)\big)^2+\xi_2\big(2r-(T\tilde{J}_k)(2)\big)^2\Big]$$

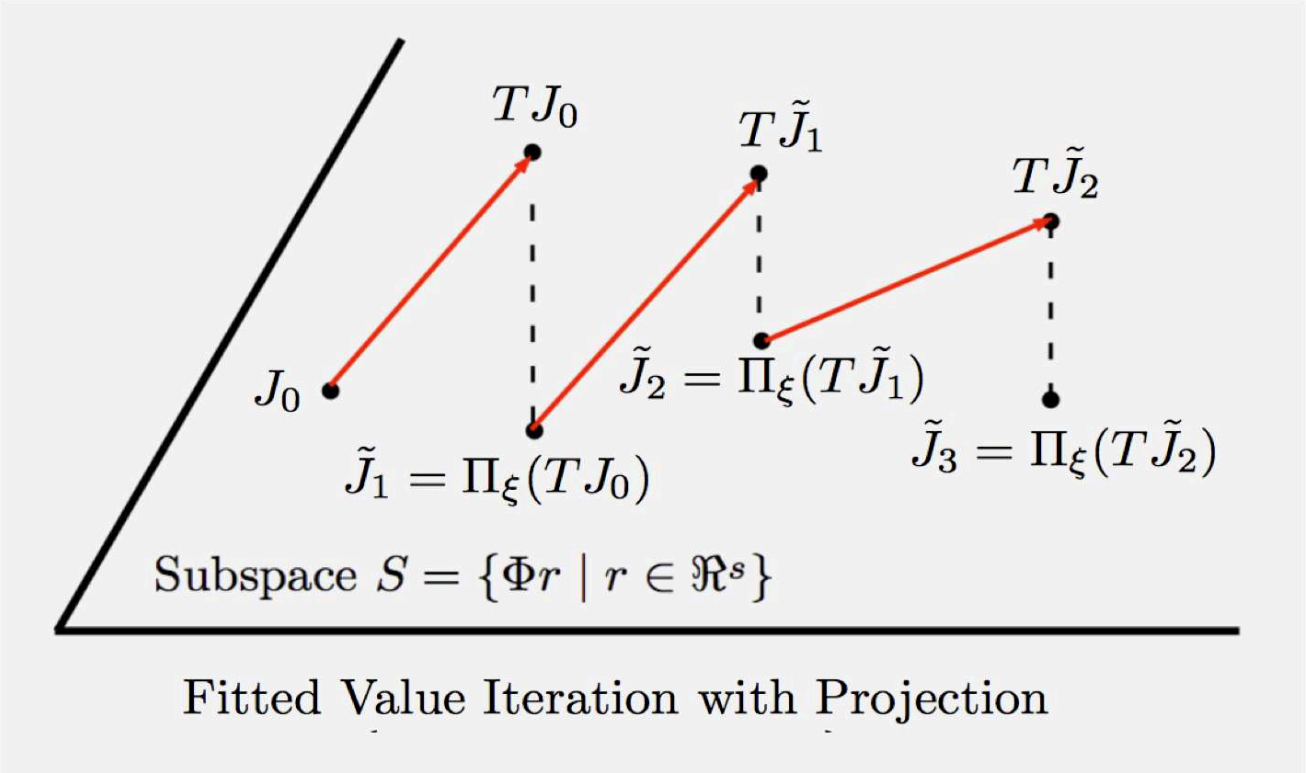
- With straightforward calculation

$$r_{k+1} = \alpha\beta r_k, \qquad \text{where } \beta = 2(\xi_1+2\xi_2)/(\xi_1+4\xi_2) > 1$$

- So if $\alpha > 1/\beta$ (e.g., $\xi_1 = \xi_2 = 1$), the sequence $\{r_k\}$ diverges and so does $\{\tilde{J}_k\}$.
- Difficulty is that T is a contraction, but $\Pi_\xi T$ (= least squares fit composed with T) is not.

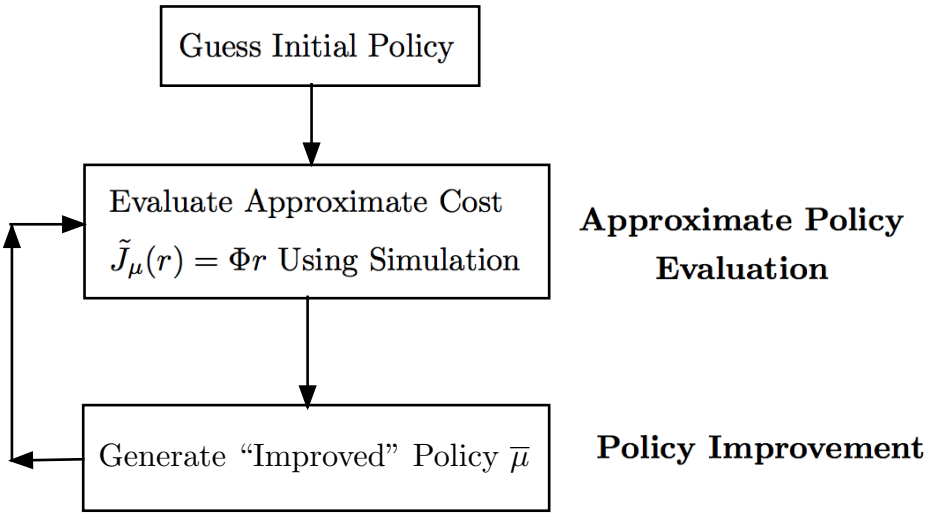
NORM MISMATCH PROBLEM

- For fitted VI to converge, we need $\Pi_\xi T$ to be a contraction; T being a contraction is not enough



- We need a ξ such that T is a contraction w. r. to the weighted Euclidean norm $\|\cdot\|_\xi$
- Then $\Pi_\xi T$ is a contraction w. r. to $\|\cdot\|_\xi$
- We will come back to this issue, and show how to choose ξ so that $\Pi_\xi T_\mu$ is a contraction for a given μ

APPROXIMATE PI



- Evaluation of typical μ: Linear cost function approximation $\tilde{J}_\mu(r) = \Phi r$, where Φ is full rank $n \times s$ matrix with columns the basis functions, and ith row denoted $\phi(i)'$.

- Policy "improvement" to generate $\overline{\mu}$:

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \phi(j)' r\big)$$

- Error Bound (same as approximate VI): If

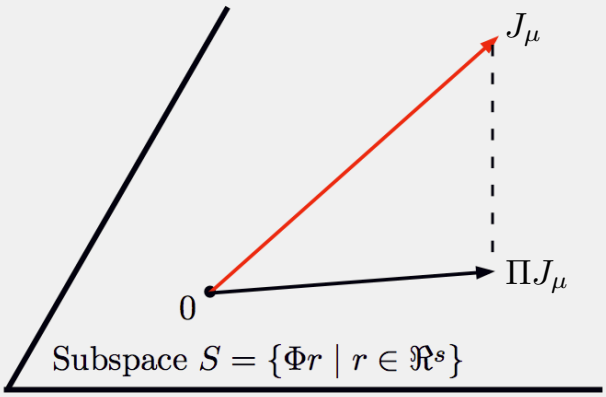
$$\max_i |\tilde{J}_{\mu^k}(i, r_k) - J_{\mu^k}(i)| \leq \delta, \qquad k = 0, 1, \ldots$$

the sequence $\{\mu^k\}$ satisfies

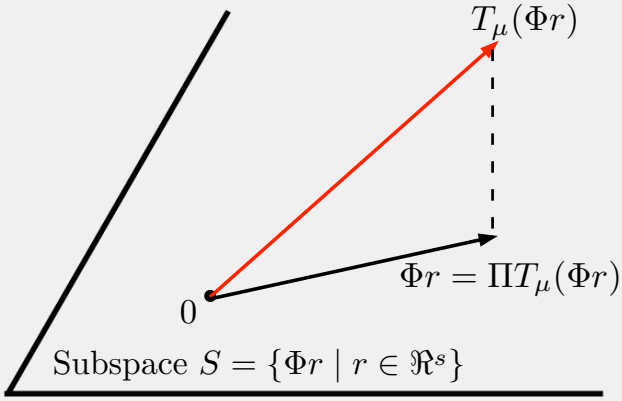
$$\limsup_{k \to \infty} \max_i \big(J_{\mu^k}(i) - J^*(i)\big) \leq \frac{2\alpha\delta}{(1-\alpha)^2}$$

APPROXIMATE POLICY EVALUATION

- Consider approximate evaluation of J_μ, the cost of the current policy μ by using simulation.
  - Direct policy evaluation - generate cost samples by simulation, and optimization by least squares
  - Indirect policy evaluation - solving the projected equation $\Phi r = \Pi T_\mu(\Phi r)$ where Π is projection w/ respect to a suitable weighted Euclidean norm



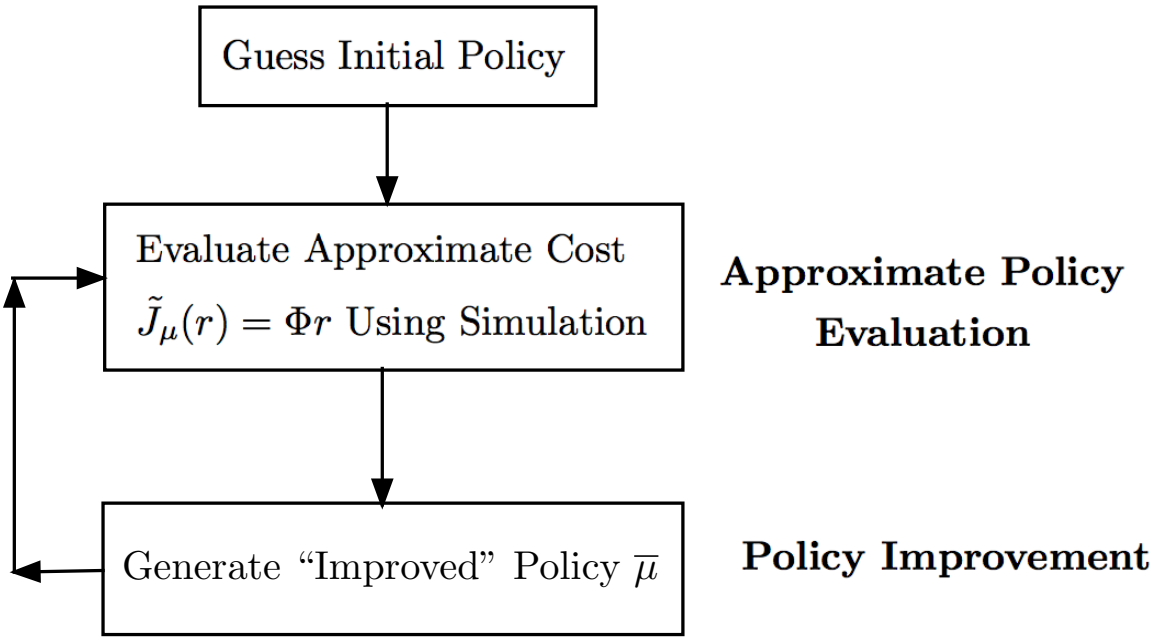
Direct Method: Projection of cost vector J_μ



Indirect Method: Solving a projected form of Bellman's equation

- Recall that projection can be implemented by simulation and least squares

PI WITH INDIRECT POLICY EVALUATION



- Given the current policy μ:
  - We solve the projected Bellman's equation

$$\Phi r = \Pi T_\mu(\Phi r)$$

  - We approximate the solution J_μ of Bellman's equation

$$J = T_\mu J$$

    with the projected equation solution $\tilde{J}_\mu(r)$

KEY QUESTIONS AND RESULTS

- Does the projected equation have a solution?
- Under what conditions is the mapping ΠT_μ a contraction, so ΠT_μ has unique fixed point?
- Assumption: The Markov chain corresponding to μ has a single recurrent class and no transient states, with steady-state prob. vector ξ, so that

$$\xi_j = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} P(i_k = j \mid i_0 = i) > 0$$

Note that ξ_j is the long-term frequency of state j.

- Proposition: (Norm Matching Property) Assume that the projection Π is with respect to $\|\cdot\|_\xi$, where $\xi = (\xi_1, \ldots, \xi_n)$ is the steady-state probability vector. Then:

  (a) ΠT_μ is contraction of modulus α with respect to $\|\cdot\|_\xi$.

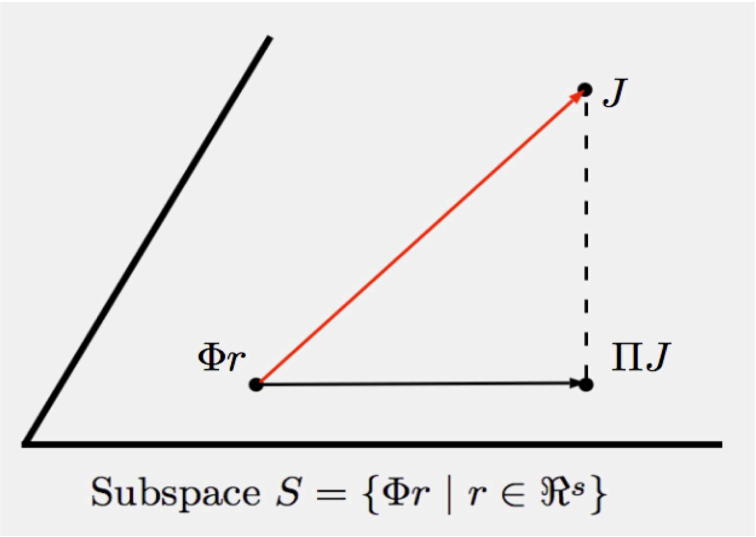
  (b) The unique fixed point Φr^* of ΠT_μ satisfies

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi$$

PRELIMINARIES: PROJECTION PROPERTIES

- Important property of the projection Π on S with weighted Euclidean norm $\|\cdot\|_\xi$. For all $J \in \Re^n$, $\Phi r \in S$, the Pythagorean Theorem holds:

$$\|J - \Phi r\|_\xi^2 = \|J - \Pi J\|_\xi^2 + \|\Pi J - \Phi r\|_\xi^2$$



- The Pythagorean Theorem implies that the projection is nonexpansive, i.e.,

$$\|\Pi J - \Pi \bar{J}\|_\xi \leq \|J - \bar{J}\|_\xi, \qquad \text{for all } J, \bar{J} \in \Re^n.$$

To see this, note that

$$\begin{aligned} \left\|\Pi(J - \overline{J})\right\|_\xi^2 &\leq \left\|\Pi(J - \overline{J})\right\|_\xi^2 + \left\|(I - \Pi)(J - \overline{J})\right\|_\xi^2 \\ &= \|J - \overline{J}\|_\xi^2 \end{aligned}$$

PROOF OF CONTRACTION PROPERTY

- Lemma: If P is the transition matrix of μ,

$$\|Pz\|_\xi \leq \|z\|_\xi, \qquad z \in \Re^n,$$

where ξ is the steady-state prob. vector.
Proof: For all $z \in \Re^n$

$$\|Pz\|_\xi^2 = \sum_{i=1}^n \xi_i \left(\sum_{j=1}^n p_{ij} z_j \right)^2 \leq \sum_{i=1}^n \xi_i \sum_{j=1}^n p_{ij} z_j^2$$

$$= \sum_{j=1}^n \sum_{i=1}^n \xi_i p_{ij} z_j^2 = \sum_{j=1}^n \xi_j z_j^2 = \|z\|_\xi^2.$$

The inequality follows from the convexity of the quadratic function, and the next to last equality follows from the defining property $\sum_{i=1}^n \xi_i p_{ij} = \xi_j$

- Using the lemma, the nonexpansiveness of Π, and the definition $T_\mu J = g + \alpha P J$, we have

$$\|\Pi T_\mu J - \Pi T_\mu \bar{J}\|_\xi \leq \|T_\mu J - T_\mu \bar{J}\|_\xi = \alpha \|P(J - \bar{J})\|_\xi \leq \alpha \|J - \bar{J}\|_\xi$$

for all $J, \bar{J} \in \Re^n$. Hence ΠT_μ is a contraction of modulus α.

PROOF OF ERROR BOUND

- Let Φr^* be the fixed point of ΠT. We have

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi.$$

Proof: We have

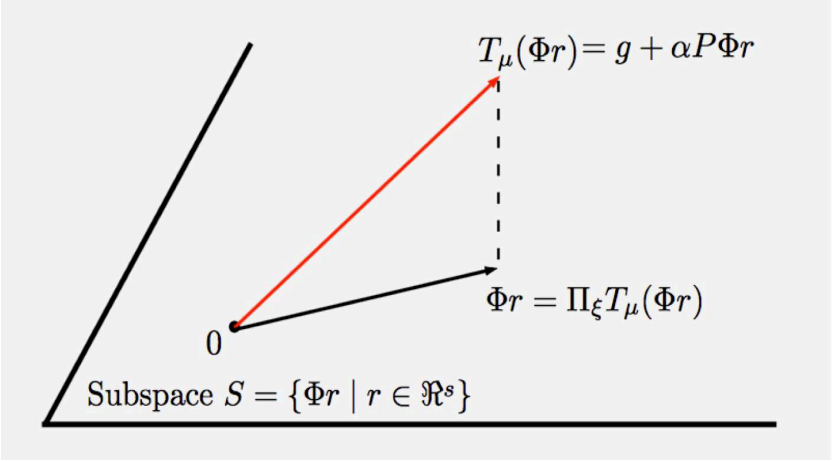
$$\begin{aligned}
\|J_\mu - \Phi r^*\|_\xi^2 &= \|J_\mu - \Pi J_\mu\|_\xi^2 + \left\|\Pi J_\mu - \Phi r^*\right\|_\xi^2 \\
&= \|J_\mu - \Pi J_\mu\|_\xi^2 + \left\|\Pi T J_\mu - \Pi T(\Phi r^*)\right\|_\xi^2 \\
&\leq \|J_\mu - \Pi J_\mu\|_\xi^2 + \alpha^2 \|J_\mu - \Phi r^*\|_\xi^2,
\end{aligned}$$

where

  - The first equality uses the Pythagorean Theorem
  - The second equality holds because J_μ is the fixed point of T and Φr^* is the fixed point of ΠT
  - The inequality uses the contraction property of ΠT.

Q.E.D.

MATRIX FORM OF PROJECTED EQUATION



- The solution Φr^* satisfies the orthogonality condition: The error

$$\Phi r^* - (g + \alpha P \Phi r^*)$$

is "orthogonal" to the subspace spanned by the columns of Φ.

- This is written as

$$\Phi' \Xi \big(\Phi r^* - (g + \alpha P \Phi r^*) \big) = 0,$$

where Ξ is the diagonal matrix with the steady-state probabilities $\xi_1, \ldots, \xi_n$ along the diagonal.

- Equivalently, $Cr^* = d$, where

$$C = \Phi' \Xi (I - \alpha P) \Phi, \qquad d = \Phi' \Xi g$$

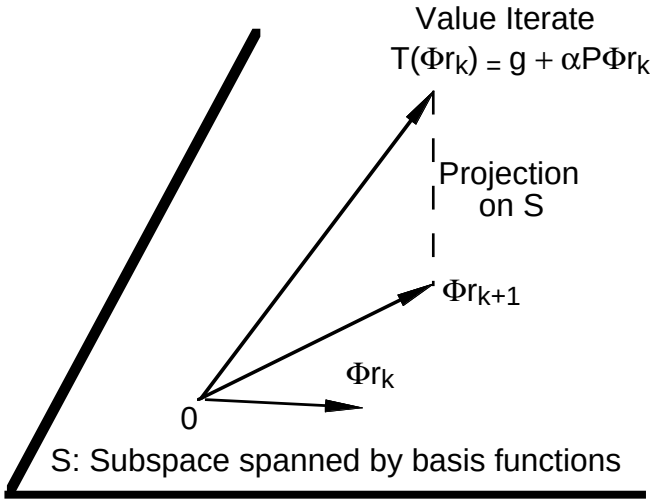
but computing C and d is HARD (high-dimensional inner products).

SOLUTION OF PROJECTED EQUATION

- Solve $Cr^* = d$ by matrix inversion: $r^* = C^{-1}d$
- Alternative: Projected Value Iteration (PVI)

$$\Phi r_{k+1} = \Pi T(\Phi r_k) = \Pi(g + \alpha P \Phi r_k)$$

Converges to r^* because ΠT is a contraction.



- PVI can be written as:

$$r_{k+1} = \arg\min_{r \in \Re^s} \left\| \Phi r - (g + \alpha P \Phi r_k) \right\|_\xi^2$$

By setting to 0 the gradient with respect to r,

$$\Phi' \Xi \big(\Phi r_{k+1} - (g + \alpha P \Phi r_k) \big) = 0,$$

which yields

$$r_{k+1} = r_k - (\Phi' \Xi \Phi)^{-1} (C r_k - d)$$

SIMULATION-BASED IMPLEMENTATIONS

- Key idea: Calculate simulation-based approximations based on k samples

$$C_k \approx C, \qquad d_k \approx d$$

- Approximate matrix inversion $r^* = C^{-1}d$ by

$$\hat{r}_k = C_k^{-1} d_k$$

This is the LSTD (Least Squares Temporal Differences) method.

- PVI method $r_{k+1} = r_k - (\Phi' \Xi \Phi)^{-1}(Cr_k - d)$ is approximated by

$$r_{k+1} = r_k - G_k(C_k r_k - d_k)$$

where

$$G_k \approx (\Phi' \Xi \Phi)^{-1}$$

This is the LSPE (Least Squares Policy Evaluation) method.

- Key fact: C_k, d_k, and G_k can be computed with low-dimensional linear algebra (of order s; the number of basis functions).

SIMULATION MECHANICS

- We generate an infinitely long trajectory $(i_0, i_1, \ldots)$ of the Markov chain, so states i and transitions (i, j) appear with long-term frequencies ξ_i and p_{ij}.

- After generating each transition (i_t, i_{t+1}), we compute the row $\phi(i_t)'$ of Φ and the cost component $g(i_t, i_{t+1})$.

- We form

$$d_k = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)g(i_t, i_{t+1}) \approx \sum_{i,j}\xi_i p_{ij}\phi(i)g(i,j) = \Phi'\Xi g = d$$

$$C_k = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\big(\phi(i_t)-\alpha\phi(i_{t+1})\big)' \approx \Phi'\Xi(I-\alpha P)\Phi = C$$

Also in the case of LSPE

$$G_k = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\phi(i_t)' \approx \Phi'\Xi\Phi$$

- Convergence based on law of large numbers.

- C_k, d_k, and G_k can be formed incrementally. Also can be written using the formalism of temporal differences (this is just a matter of style)

OPTIMISTIC VERSIONS

- Instead of calculating nearly exact approximations $C_k \approx C$ and $d_k \approx d$, we do a less accurate approximation, based on few simulation samples
- Evaluate (coarsely) current policy μ, then do a policy improvement
- This often leads to faster computation (as optimistic methods often do)
- Very complex behavior (see the subsequent discussion on oscillations)
- The matrix inversion/LSTD method has serious problems due to large simulation noise (because of limited sampling) - particularly if the C matrix is ill-conditioned
- LSPE tends to cope better because of its iterative nature (this is true of other iterative methods as well)
- A stepsize $\gamma \in (0, 1]$ in LSPE may be useful to damp the effect of simulation noise

$$r_{k+1} = r_k - \gamma G_k(C_k r_k - d_k)$$

6.231 DYNAMIC PROGRAMMING

LECTURE 21

LECTURE OUTLINE

- Review of approximate policy iteration
- Projected equation methods for policy evaluation
- Issues related to simulation-based implementation
- Multistep projected equation methods
- Bias-variance tradeoff
- Exploration-enhanced implementations
- Oscillations

REVIEW: PROJECTED BELLMAN EQUATION

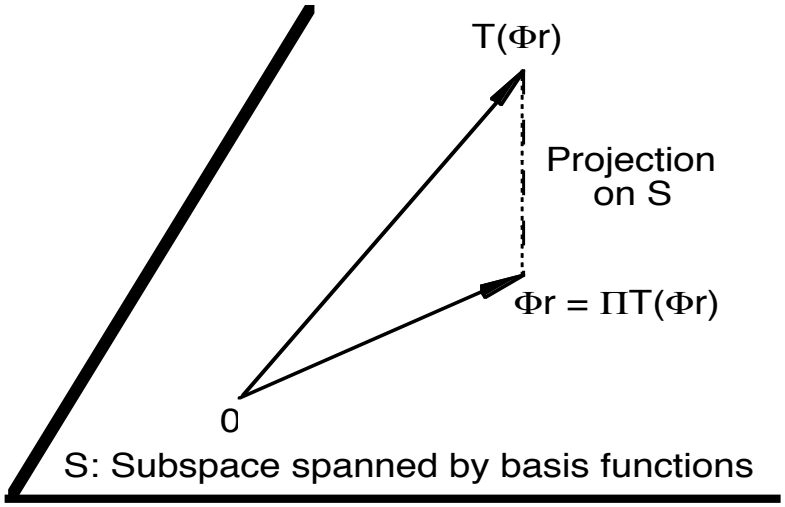
- For a fixed policy μ to be evaluated, consider the corresponding mapping T:

$$(TJ)(i) = \sum_{i=1}^{n} p_{ij}\big(g(i,j)+\alpha J(j)\big), \qquad i = 1, \ldots, n,$$

or more compactly, $TJ = g + \alpha PJ$

- Approximate Bellman's equation $J = TJ$ by $\Phi r = \Pi T(\Phi r)$ or the matrix form/orthogonality condition $Cr^* = d$, where

$$C = \Phi' \Xi (I - \alpha P)\Phi, \qquad d = \Phi' \Xi g.$$



Indirect method: Solving a projected form of Bellman's equation

PROJECTED EQUATION METHODS

- Matrix inversion: $r^* = C^{-1}d$
- Iterative Projected Value Iteration (PVI) method:

$$\Phi r_{k+1} = \Pi T(\Phi r_k) = \Pi(g + \alpha P \Phi r_k)$$

Converges to r^* if ΠT is a contraction. True if Π is projection w.r.t. steady-state distribution norm.

- Simulation-Based Implementations: Generate $k+1$ simulated transitions sequence $\{i_0, i_1, \ldots, i_k\}$ and approximations $C_k \approx C$ and $d_k \approx d$:

$$C_k = \frac{1}{k+1}\sum_{t=0}^{k} \phi(i_t)\big(\phi(i_t) - \alpha\phi(i_{t+1})\big)' \approx \Phi'\Xi(I - \alpha P)\Phi$$

$$d_k = \frac{1}{k+1}\sum_{t=0}^{k} \phi(i_t) g(i_t, i_{t+1}) \approx \Phi'\Xi g$$

- LSTD: $\hat{r}_k = C_k^{-1} d_k$
- LSPE: $r_{k+1} = r_k - G_k(C_k r_k - d_k)$ where

$$G_k \approx G = (\Phi'\Xi\Phi)^{-1}$$

Converges to r^* if ΠT is contraction.

ISSUES FOR PROJECTED EQUATIONS

- Implementation of simulation-based solution of projected equation $\Phi r \approx J_\mu$, where $C_k r = d_k$ and

$$C_k \approx \Phi' \Xi (I - \alpha P)\Phi, \qquad d_k \approx \Phi' \Xi g$$

- Low-dimensional linear algebra needed for the simulation-based approximations C_k and d_k (of order s; the number of basis functions).

- Very large number of samples needed to solve reliably nearly singular projected equations.

- Special methods for nearly singular equations by simulation exist; see Section 7.3 of the text.

- Optimistic (few sample) methods are more vulnerable to simulation error

- Norm mismatch/sampling distribution issue

- The problem of bias: Projected equation solution $\neq \Pi J_\mu$, the "closest" approximation of J_μ

- Everything said so far relates to policy evaluation. How about the effect of approximations on policy improvement?

- We will next address some of these issues

MULTISTEP METHODS

- Introduce a multistep version of Bellman's equation $J = T^{(\lambda)} J$, where for $\lambda \in [0, 1)$,

$$T^{(\lambda)} = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^{\ell} T^{\ell+1}$$

Geometrically weighted sum of powers of T.

- T^{ℓ} is a contraction with mod. α^{ℓ}, w. r. to weighted Euclidean norm $\| \cdot \|_{\xi}$, where ξ is the steady-state probability vector of the Markov chain.

- Hence $T^{(\lambda)}$ is a contraction with modulus

$$\alpha_{\lambda} = (1 - \lambda) \sum_{\ell=0}^{\infty} \alpha^{\ell+1} \lambda^{\ell} = \frac{\alpha(1 - \lambda)}{1 - \alpha\lambda}$$

Note $\alpha_{\lambda} \to 0$ as $\lambda \to 1$ - affects norm mismatch

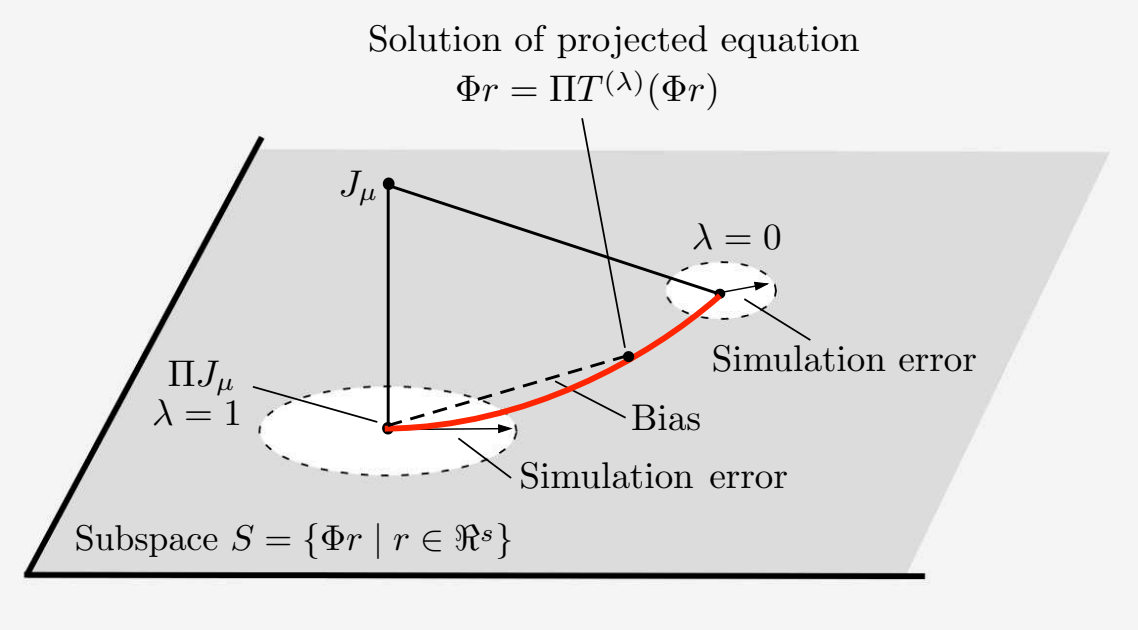
- T^{ℓ} and $T^{(\lambda)}$ have the same fixed point J_{μ} and

$$\|J_{\mu} - \Phi r_{\lambda}^{*}\|_{\xi} \leq \frac{1}{\sqrt{1 - \alpha_{\lambda}^{2}}} \|J_{\mu} - \Pi J_{\mu}\|_{\xi}$$

where Φr_{λ}^{*} is the fixed point of $\Pi T^{(\lambda)}$.

- Φr_{λ}^{*} depends on λ.

BIAS-VARIANCE TRADEOFF



- From $\|J_\mu - \Phi r_{\lambda,\mu}\|_\xi \leq \frac{1}{\sqrt{1-\alpha_\lambda^2}} \|J_\mu - \Pi J_\mu\|_\xi$ error bound
- As $\lambda \uparrow 1$, we have $\alpha_\lambda \downarrow 0$, so error bound (and quality of approximation) improves:

$$\lim_{\lambda \uparrow 1} \Phi r_{\lambda,\mu} = \Pi J_\mu$$

- But the simulation noise in approximating

$$T^{(\lambda)} = (1-\lambda) \sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}$$

increases

- Choice of λ is usually based on trial and error

MULTISTEP PROJECTED EQ. METHODS

- The multistep projected Bellman equation is

$$\Phi r = \Pi T^{(\lambda)}(\Phi r)$$

- In matrix form: $C^{(\lambda)} r = d^{(\lambda)}$, where

$$C^{(\lambda)} = \Phi' \Xi \big(I - \alpha P^{(\lambda)}\big)\Phi, \qquad d^{(\lambda)} = \Phi' \Xi g^{(\lambda)},$$

with

$$P^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \alpha^\ell \lambda^\ell P^{\ell+1}, \quad g^{(\lambda)} = \sum_{\ell=0}^{\infty} \alpha^\ell \lambda^\ell P^\ell g$$

- The LSTD(λ) method is $\big(C_k^{(\lambda)}\big)^{-1} d_k^{(\lambda)}$, where $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ are simulation-based approximations of $C^{(\lambda)}$ and $d^{(\lambda)}$.

- The LSPE(λ) method is

$$r_{k+1} = r_k - \gamma G_k \big(C_k^{(\lambda)} r_k - d_k^{(\lambda)}\big)$$

where G_k is a simulation-based approx. to $(\Phi' \Xi \Phi)^{-1}$

- TD(λ): An important simpler/slower iteration [similar to LSPE(λ) with $G_k = I$ - see the text].

MORE ON MULTISTEP METHODS

- The simulation process to obtain $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ is similar to the case $\lambda = 0$ (single simulation trajectory $i_0, i_1, \ldots$, more complex formulas)

$$C_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \sum_{m=t}^{k} \alpha^{m-t} \lambda^{m-t} \big(\phi(i_m) - \alpha \phi(i_{m+1})\big)'$$

$$d_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \sum_{m=t}^{k} \alpha^{m-t} \lambda^{m-t} g_{i_m}$$

- In the context of approximate policy iteration, we can use optimistic versions (few samples between policy updates).
- Many different versions (see the text).
- Note the λ-tradeoffs:
  - As $\lambda \uparrow 1$, $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ contain more "simulation noise", so more samples are needed for a close approximation of $r_{\lambda,\mu}$
  - The error bound $\|J_\mu - \Phi r_{\lambda,\mu}\|_\xi$ becomes smaller
  - As $\lambda \uparrow 1$, $\Pi T^{(\lambda)}$ becomes a contraction for arbitrary projection norm

APPROXIMATE PI ISSUES - EXPLORATION

- 1st major issue: exploration. Common remedy is the off-policy approach: Replace P of current policy with

$$\overline{P} = (I - B)P + BQ,$$

where B is a diagonal matrix with $\beta_i \in [0, 1]$ on the diagonal, and Q is another transition matrix.

- Then LSTD and LSPE formulas must be modified ... otherwise the policy associated with $\overline{P}$ (not P) is evaluated (see the textbook, Section 6.4).

- Alternatives: Geometric and free-form sampling

- Both of these use multiple short simulated trajectories, with random restart state, chosen to enhance exploration (see the text)

- Geometric sampling uses trajectories with geometrically distributed number of transitions with parameter $\lambda \in [0, 1)$. It implements LSTD(λ) and LSPE(λ) with exploration.

- Free-form sampling uses trajectories with more generally distributed number of transitions. It implements method for approximation of the solution of a generalized multistep Bellman equation.

APPROXIMATE PI ISSUES - OSCILLATIONS

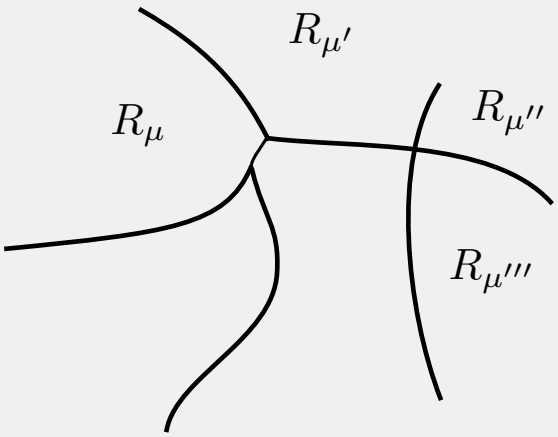
- Define for each policy μ

$$R_\mu = \big\{ r \mid T_\mu(\Phi r) = T(\Phi r) \big\}$$

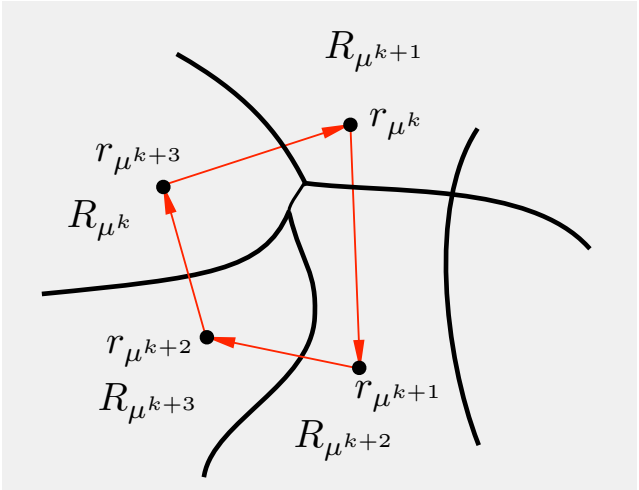
- These sets form the greedy partition of the parameter r-space

$$R_\mu = \big\{ r \mid T_\mu(\Phi r) = T(\Phi r) \big\}$$

For a policy μ, R_μ is the set of all r such that policy improvement based on Φr produces μ

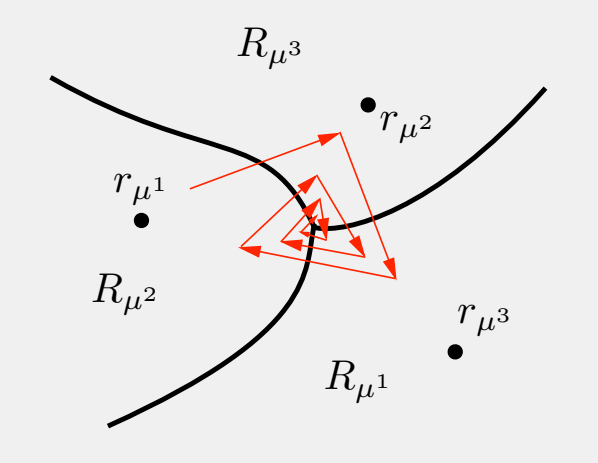


- Oscillations of nonoptimistic approx.: r_μ is generated by an evaluation method so that $\Phi r_\mu \approx J_\mu$



MORE ON OSCILLATIONS/CHATTERING

- For optimistic PI a different picture holds



- Oscillations are less violent, but the "limit" point is meaningless!

- Fundamentally, oscillations are due to the lack of monotonicity of the projection operator, i.e., $J \leq J'$ does not imply $\Pi J \leq \Pi J'$.

- If approximate PI uses policy evaluation

$$\Phi r = (WT_\mu)(\Phi r)$$

with W a monotone operator, the generated policies converge (to an approximately optimal limit).

- The operator W used in the aggregation approach has this monotonicity property.

6.231 DYNAMIC PROGRAMMING

LECTURE 22

LECTURE OUTLINE

- Aggregation as an approximation methodology
- Aggregate problem
- Examples of aggregation
- Simulation-based aggregation
- Q-Learning

PROBLEM APPROXIMATION - AGGREGATION

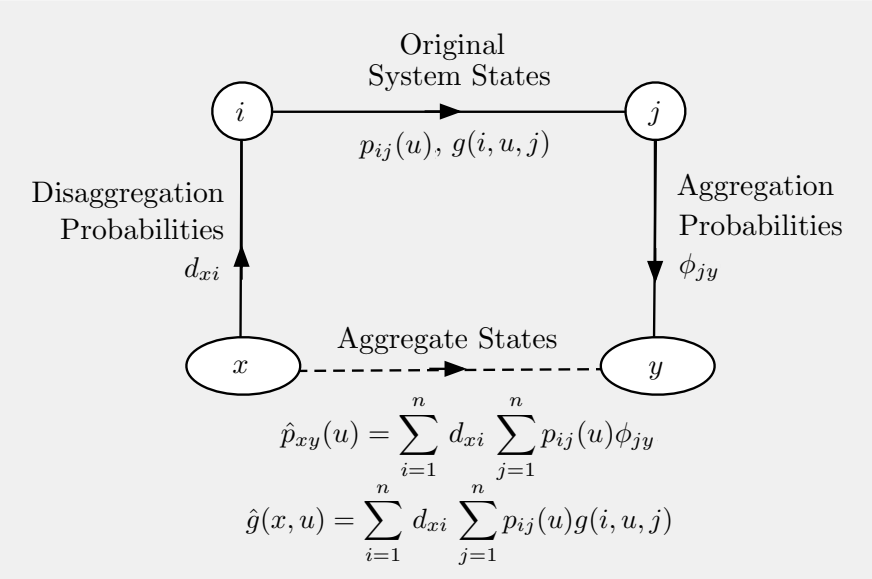
- Another major idea in ADP is to approximate the cost-to-go function of the problem with the cost-to-go function of a simpler problem. The simplification is often ad-hoc/problem dependent.

- Aggregation is a systematic approach for problem approximation. Main elements:
  - Introduce a few "aggregate" states, viewed as the states of an "aggregate" system
  - Define transition probabilities and costs of the aggregate system, by relating original system states with aggregate states
  - Solve (exactly or approximately) the "aggregate" problem by any kind of value or policy iteration method (including simulation-based methods)
  - Use the optimal cost of the aggregate problem to approximate the optimal cost of the original problem

- Hard aggregation example: Aggregate states are subsets of original system states, treated as if they all have the same cost.

AGGREGATION/DISAGGREGATION PROBS

- The aggregate system transition probabilities are defined via two (somewhat arbitrary) choices



- For each original system state j and aggregate state y, the aggregation probability ϕ_{jy}
  - The "degree of membership of j in the aggregate state y."
  - In hard aggregation, $\phi_{jy} = 1$ if state j belongs to aggregate state/subset y.

- For each aggregate state x and original system state i, the disaggregation probability d_{xi}
  - The "degree of i being representative of x."
  - In hard aggregation, one possibility is all states i that belongs to aggregate state/subset x have equal d_{xi}.

AGGREGATE PROBLEM

- The transition probability from aggregate state x to aggregate state y under control u

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}, \quad \text{or } \hat{P}(u) = DP(u)\Phi$$

where the rows of D and Φ are the disaggr. and aggr. probs.

- The aggregate expected transition cost is

$$\hat{g}(x,u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)g(i,u,j), \quad \text{or } \hat{g} = DPg$$

- The optimal cost function of the aggregate problem, denoted $\hat{R}$, is

$$\hat{R}(x) = \min_{u \in U} \left[\hat{g}(x,u) + \alpha \sum_{y} \hat{p}_{xy}(u)\hat{R}(y) \right], \qquad \forall\, x$$

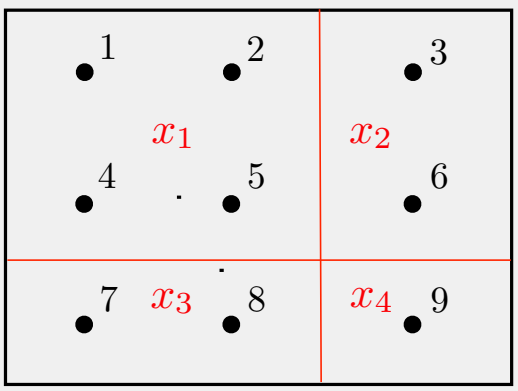
or $\hat{R} = \min_u[\hat{g} + \alpha \hat{P}\hat{R}]$ - Bellman's equation for the aggregate problem.

- The optimal cost J^* of the original problem is approximated using interpolation, $J^* \approx \tilde{J} = \Phi\hat{R}$:

$$\tilde{J}(j) = \sum_{y} \phi_{jy}\hat{R}(y), \qquad \forall\, j$$

EXAMPLE I: HARD AGGREGATION

- Group the original system states into subsets, and view each subset as an aggregate state
- Aggregation probs: $\phi_{jy} = 1$ if j belongs to aggregate state y.

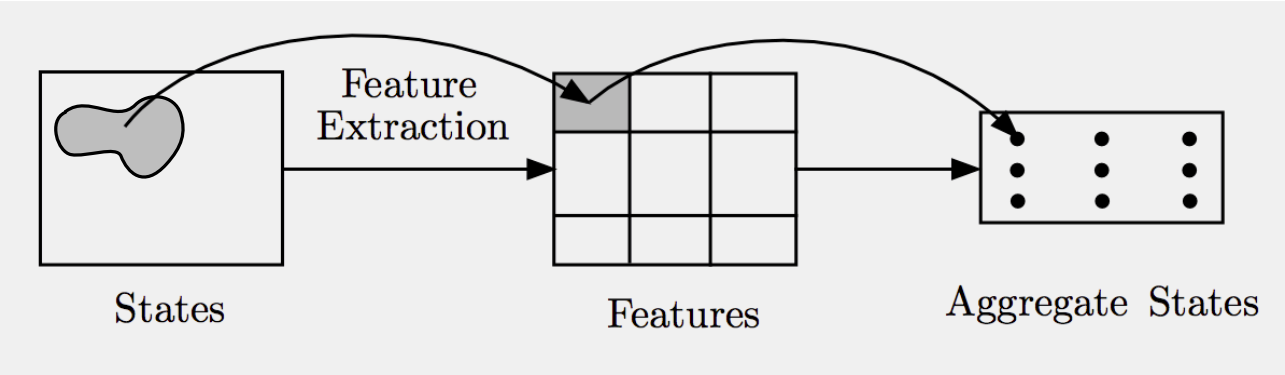


$$\Phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Disaggregation probs: There are many possibilities, e.g., all states i within aggregate state x have equal prob. d_{xi}.
- If optimal cost vector J^* is piecewise constant over the aggregate states/subsets, hard aggregation is exact. Suggests grouping states with "roughly equal" cost into aggregates.
- Soft aggregation (provides "soft boundaries" between aggregate states).

EXAMPLE II: FEATURE-BASED AGGREGATION

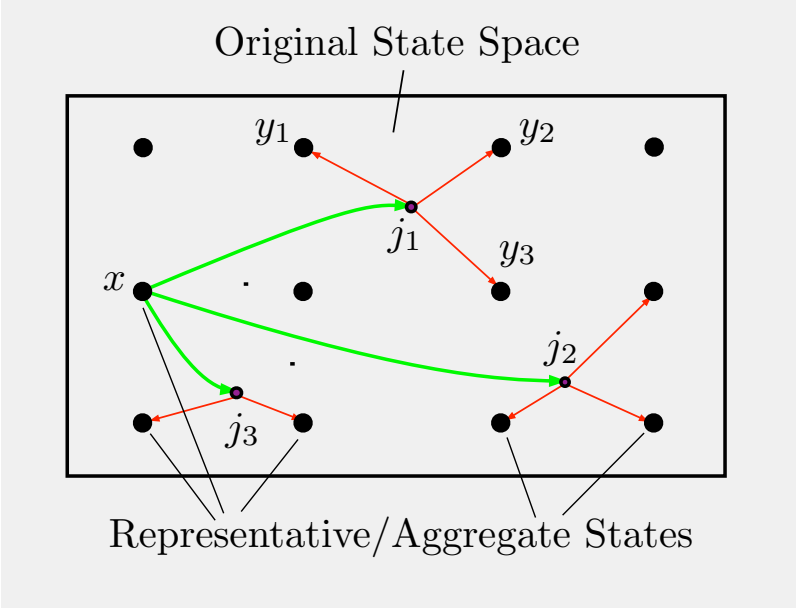
- If we know good features, it makes sense to group together states that have "similar features"
- Essentially discretize the features and assign a weight to each discretization point



- A general approach for passing from a feature-based state representation to an aggregation-based architecture
- Hard aggregation architecture based on features is more powerful (nonlinear/piecewise constant in the features, rather than linear)
- ... but may require many more aggregate states to reach the same level of performance as the corresponding linear feature-based architecture

EXAMPLE III: REP. STATES/COARSE GRID

- Choose a collection of "representative" original system states, and associate each one of them with an aggregate state. Then "interpolate"



- Disaggregation probs. are $d_{xi} = 1$ if i is equal to representative state x.

- Aggregation probs. associate original system states with convex combinations of rep. states

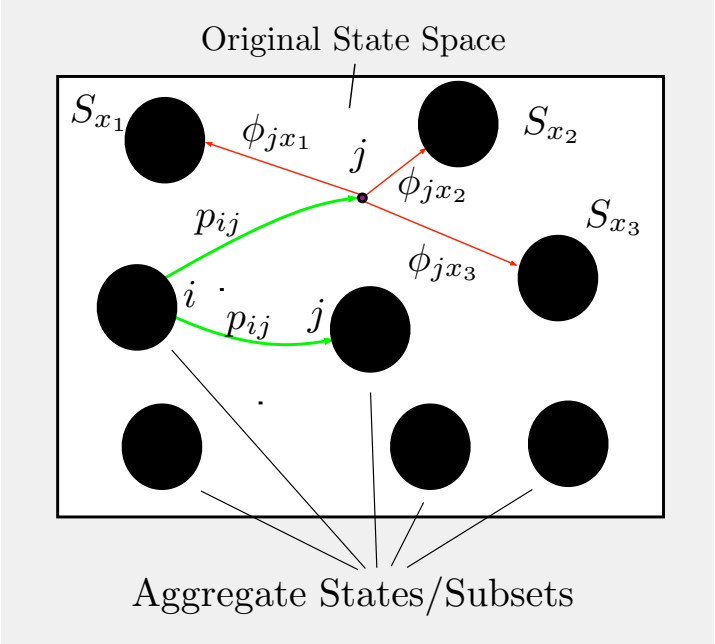
$$j \sim \sum_{y \in \mathcal{A}} \phi_{jy} y$$

- Well-suited for Euclidean space discretization

- Extends nicely to continuous state space, including belief space of POMDP

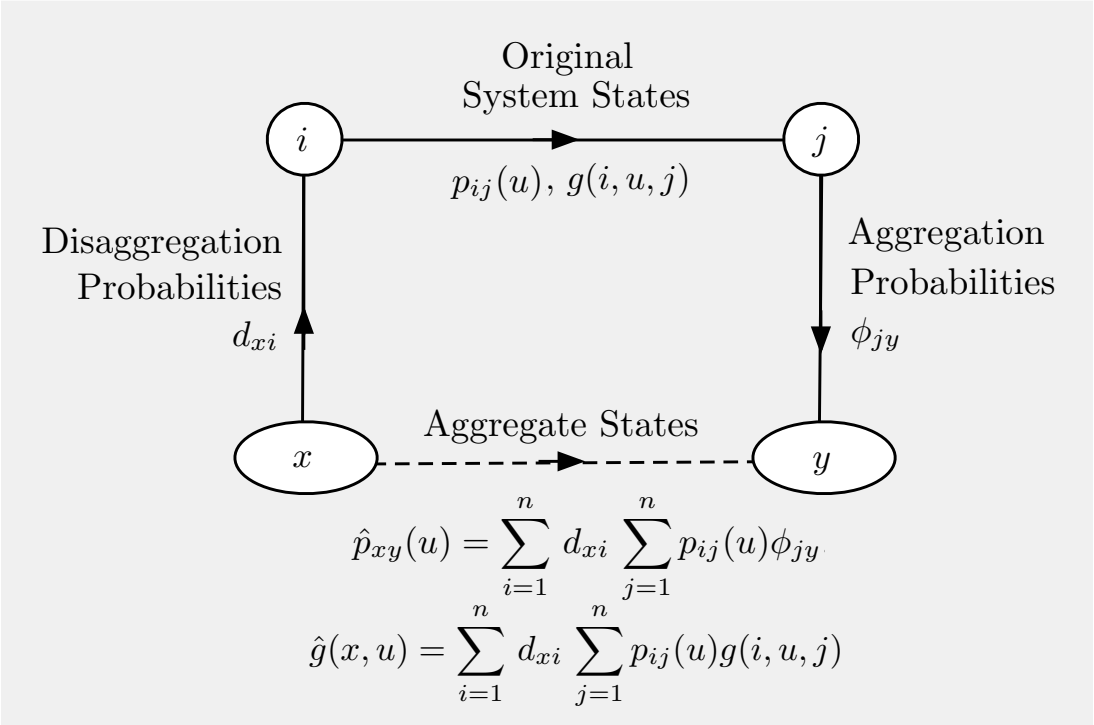
EXAMPLE IV: REPRESENTATIVE FEATURES

- Choose a collection of "representative" subsets of original system states, and associate each one of them with an aggregate state



- Common case: S_x is a group of states with "similar features"
- Hard aggregation is special case: $\cup_x S_x = \{1, \ldots, n\}$
- Aggregation with representative states is special case: S_x consists of just one state
- With rep. features, aggregation approach is a special case of projected equation approach with "seminorm" projection. So the TD methods and multistage Bellman Eq. methodology apply

APPROXIMATE PI BY AGGREGATION



- Consider approximate PI for the original problem, with evaluation done using the aggregate problem (other possibilities exist - see the text)

- Evaluation of policy μ: $\tilde{J} = \Phi R$, where $R = DT_\mu(\Phi R)$ (R is the vector of costs of aggregate states corresponding to μ). May use simulation.

- Similar form to the projected equation $\Phi R = \Pi T_\mu(\Phi R)$ (ΦD in place of Π).

- Advantages: It has no problem with exploration or with oscillations.

- Disadvantage: The rows of D and Φ must be probability distributions.

Q-LEARNING I

- Q-learning has two motivations:
  - Dealing with multiple policies simultaneously
  - Using a model-free approach [no need to know $p_{ij}(u)$, only be able to simulate them]
- The Q-factors are defined by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \quad \forall\ (i,u)$$

- Since $J^* = TJ^*$, we have $J^*(i) = \min_{u \in U(i)} Q^*(i,u)$ so the Q factors solve the equation

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{u' \in U(j)} Q^*(j,u')\right)$$

- $Q^*(i,u)$ can be shown to be the unique solution of this equation. Reason: This is Bellman's equation for a system whose states are the original states $1,\ldots,n$, together with all the pairs (i,u).
- Value iteration: For all (i,u)

$$Q(i,u) := \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{u' \in U(j)} Q(j,u')\right)$$

Q-LEARNING II

- Use some randomization to generate sequence of pairs (i_k, u_k) [all pairs (i, u) are chosen infinitely often]. For each k, select j_k according to $p_{i_k j}(u_k)$.

- Q-learning algorithm: updates $Q(i_k, u_k)$ by

$$Q(i_k, u_k) := \big(1 - \gamma_k(i_k, u_k)\big)Q(i_k, u_k) + \gamma_k(i_k, u_k)\left(g(i_k, u_k, j_k) + \alpha \min_{u' \in U(j_k)} Q(j_k, u')\right)$$

- Stepsize $\gamma_k(i_k, u_k)$ must converge to 0 at proper rate (e.g., like $1/k$).

- Important mathematical point: In the Q-factor version of Bellman's equation the order of expectation and minimization is reversed relative to the ordinary cost version of Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha J^*(j)\big)$$

- Q-learning can be shown to converge to true/exact Q-factors (sophisticated stoch. approximation proof).

- Major drawback: Large number of pairs (i, u) - no function approximation is used.

Q-FACTOR APPROXIMATIONS

- Basis function approximation for Q-factors:

$$\tilde{Q}(i,u,r) = \phi(i,u)'r$$

- We can use approximate policy iteration and LSPE/LSTD/TD for policy evaluation (exploration issue is acute).

- Optimistic policy iteration methods are frequently used on a heuristic basis.

- Example (very optimistic). At iteration k, given r_k and state/control (i_k, u_k):

  (1) Simulate next transition (i_k, i_{k+1}) using the transition probabilities $p_{i_k j}(u_k)$.

  (2) Generate control u_{k+1} from

$$u_{k+1} = \arg \min_{u \in U(i_{k+1})} \tilde{Q}(i_{k+1}, u, r_k)$$

  (3) Update the parameter vector via

$$r_{k+1} = r_k - (\text{LSPE or TD-like correction})$$

- Unclear validity. Solid basis for aggregation case, and for case of optimal stopping (see text).

6.231 DYNAMIC PROGRAMMING

LECTURE 23

LECTURE OUTLINE

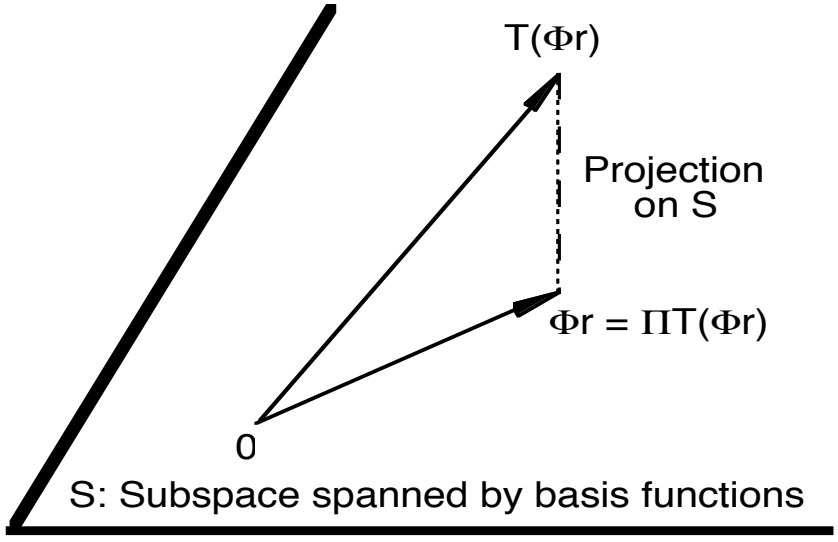
- Additional topics in ADP
- Stochastic shortest path problems
- Average cost problems
- Generalizations
- Basis function adaptation
- Gradient-based approximation in policy space
- An overview

REVIEW: PROJECTED BELLMAN EQUATION

- Policy Evaluation: Bellman's equation $J = TJ$ is approximated the projected equation

$$\Phi r = \Pi T(\Phi r)$$

which can be solved by a simulation-based methods, e.g., LSPE(λ), LSTD(λ), or TD(λ). Aggregation is another approach - simpler in some ways.



Indirect method: Solving a projected form of Bellman's equation

- These ideas apply to other (linear) Bellman equations, e.g., for SSP and average cost.

- Important Issue: Construct simulation framework where ΠT [or $\Pi T^{(\lambda)}$] is a contraction.

STOCHASTIC SHORTEST PATHS

- Introduce approximation subspace

$$S = \{\Phi r \mid r \in \Re^s\}$$

and for a given proper policy, Bellman's equation and its projected version

$$J = TJ = g + PJ, \qquad \Phi r = \Pi T(\Phi r)$$

Also its λ-version

$$\Phi r = \Pi T^{(\lambda)}(\Phi r), \qquad T^{(\lambda)} = (1-\lambda)\sum_{t=0}^{\infty} \lambda^t T^{t+1}$$

- Question: What should be the norm of projection? How to implement it by simulation?

- Speculation based on discounted case: It should be a weighted Euclidean norm with weight vector $\xi = (\xi_1, \ldots, \xi_n)$, where ξ_i should be some type of long-term occupancy probability of state i (which can be generated by simulation).

- But what does "long-term occupancy probability of a state" mean in the SSP context?

- How do we generate infinite length trajectories given that termination occurs with prob. 1?

SIMULATION FOR SSP

- We envision simulation of trajectories up to termination, followed by restart at state i with some fixed probabilities $q_0(i) > 0$.

- Then the "long-term occupancy probability of a state" of i is proportional to

$$q(i) = \sum_{t=0}^{\infty} q_t(i), \qquad i = 1, \ldots, n,$$

where

$$q_t(i) = P(i_t = i), \qquad i = 1, \ldots, n, \ t = 0, 1, \ldots$$

- We use the projection norm

$$\|J\|_q = \sqrt{\sum_{i=1}^{n} q(i) \big(J(i)\big)^2}$$

[Note that $0 < q(i) < \infty$, but q is not a prob. distribution.]

- We can show that $\Pi T^{(\lambda)}$ is a contraction with respect to $\|\cdot\|_q$ (see the next slide).

- LSTD(λ), LSPE(λ), and TD(λ) are possible.

CONTRACTION PROPERTY FOR SSP

- We have $q = \sum_{t=0}^{\infty} q_t$ so

$$q'P = \sum_{t=0}^{\infty} q_t'P = \sum_{t=1}^{\infty} q_t' = q' - q_0'$$

or

$$\sum_{i=1}^{n} q(i)p_{ij} = q(j) - q_0(j), \qquad \forall\ j$$

- To verify that ΠT is a contraction, we show that there exists $\beta < 1$ such that $\|Pz\|_q^2 \leq \beta \|z\|_q^2$ for all $z \in \Re^n$.

- For all $z \in \Re^n$, we have

$$\begin{aligned}
\|Pz\|_q^2 &= \sum_{i=1}^{n} q(i) \left(\sum_{j=1}^{n} p_{ij} z_j \right)^2 \leq \sum_{i=1}^{n} q(i) \sum_{j=1}^{n} p_{ij} z_j^2 \\
&= \sum_{j=1}^{n} z_j^2 \sum_{i=1}^{n} q(i)p_{ij} = \sum_{j=1}^{n} \big(q(j) - q_0(j)\big) z_j^2 \\
&= \|z\|_q^2 - \|z\|_{q_0}^2 \leq \beta \|z\|_q^2
\end{aligned}$$

where

$$\beta = 1 - \min_j \frac{q_0(j)}{q(j)}$$

AVERAGE COST PROBLEMS

- Consider a single policy to be evaluated, with single recurrent class, no transient states, and steady-state probability vector $\xi = (\xi_1, \ldots, \xi_n)$.
- The average cost, denoted by η, is

$$\eta = \lim_{N\to\infty} \frac{1}{N} E\left\{ \sum_{k=0}^{N-1} g\big(x_k, x_{k+1}\big) \;\Big|\; x_0 = i \right\}, \quad \forall\ i$$

- Bellman's equation is $J = FJ$ with

$$FJ = g - \eta e + PJ$$

where e is the unit vector $e = (1, \ldots, 1)$.

- The projected equation and its λ-version are

$$\Phi r = \Pi F(\Phi r), \qquad \Phi r = \Pi F^{(\lambda)}(\Phi r)$$

- A problem here is that F is not a contraction with respect to any norm (since $e = Pe$).
- $\Pi F^{(\lambda)}$ is a contraction w. r. to $\|\cdot\|_\xi$ assuming that e does not belong to S and $\lambda > 0$ (the case $\lambda = 0$ is exceptional, but can be handled); see the text. LSTD(λ), LSPE(λ), and TD(λ) are possible.

GENERALIZATION/UNIFICATION

- Consider approx. solution of $x = T(x)$, where

$$T(x) = Ax + b, \qquad A \text{ is } n \times n, \quad b \in \Re^n$$

by solving the projected equation $y = \Pi T(y)$, where Π is projection on a subspace of basis functions (with respect to some Euclidean norm).

- We can generalize from DP to the case where A is arbitrary, subject only to

$$I - \Pi A : \text{ invertible}$$

Also can deal with case where $I - \Pi A$ is (nearly) singular (iterative methods, see the text).

- Benefits of generalization:
  - Unification/higher perspective for projected equation (and aggregation) methods in approximate DP
  - An extension to a broad new area of applications, based on an approx. DP perspective
- Challenge: Dealing with less structure
  - Lack of contraction
  - Absence of a Markov chain

GENERALIZED PROJECTED EQUATION

- Let Π be projection with respect to

$$\|x\|_\xi = \sqrt{\sum_{i=1}^n \xi_i x_i^2}$$

where $\xi \in \Re^n$ is a probability distribution with positive components.

- If r^* is the solution of the projected equation, we have $\Phi r^* = \Pi(A\Phi r^* + b)$ or

$$r^* = \arg\min_{r\in\Re^s} \sum_{i=1}^n \xi_i \left(\phi(i)'r - \sum_{j=1}^n a_{ij}\phi(j)'r^* - b_i \right)^2$$

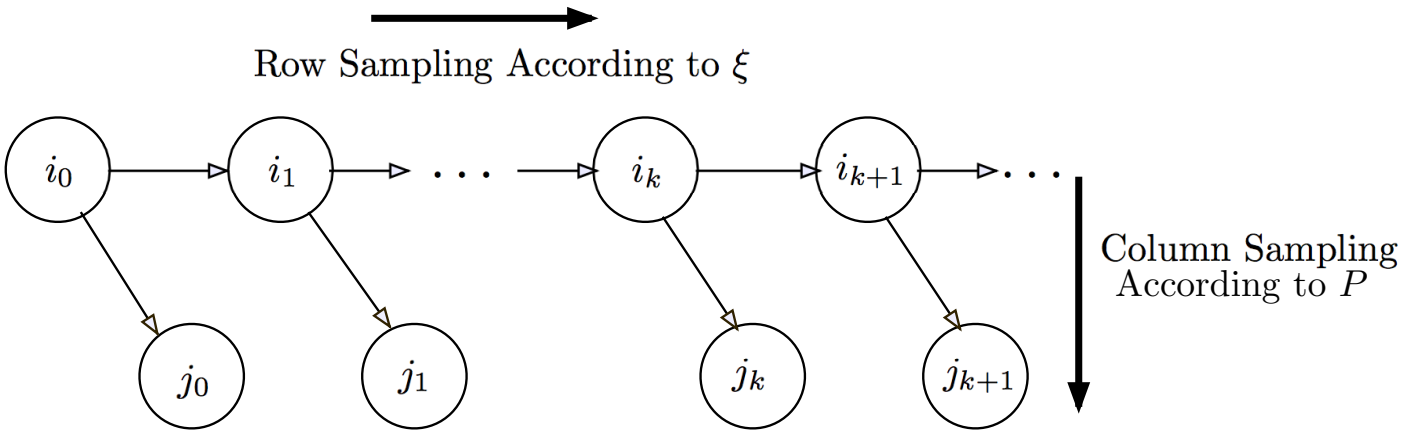
where $\phi(i)'$ denotes the ith row of the matrix Φ.

- Optimality condition/equivalent form:

$$\sum_{i=1}^n \xi_i \phi(i) \left(\phi(i) - \sum_{j=1}^n a_{ij}\phi(j) \right)' r^* = \sum_{i=1}^n \xi_i \phi(i) b_i$$

- The two expected values can be approximated by simulation

SIMULATION MECHANISM



- Row sampling: Generate sequence $\{i_0, i_1, \ldots\}$ according to ξ, i.e., relative frequency of each row i is ξ_i

- Column sampling: Generate $\{(i_0, j_0), (i_1, j_1), \ldots\}$ according to some transition probability matrix P with

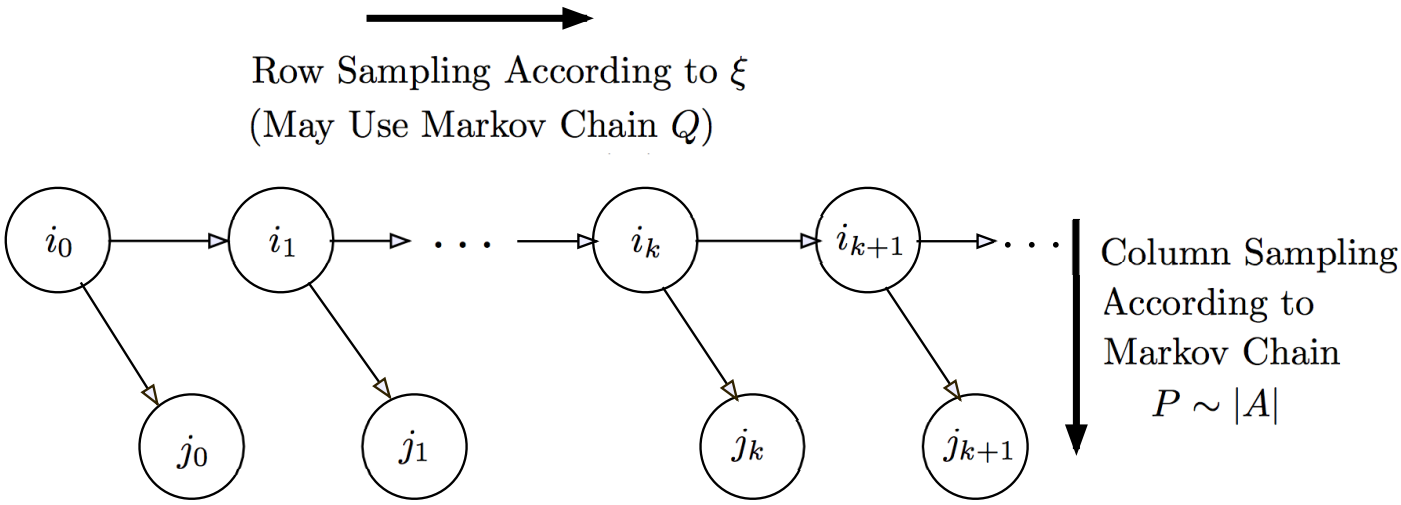
$$p_{ij} > 0 \qquad \text{if} \qquad a_{ij} \neq 0,$$

i.e., for each i, the relative frequency of (i, j) is p_{ij} (connection to importance sampling)

- Row sampling may be done using a Markov chain with transition matrix Q (unrelated to P)

- Row sampling may also be done without a Markov chain - just sample rows according to some known distribution ξ (e.g., a uniform)

ROW AND COLUMN SAMPLING



- Row sampling $\sim$ State Sequence Generation in DP. Affects:
  - The projection norm.
  - Whether ΠA is a contraction.
- Column sampling $\sim$ Transition Sequence Generation in DP.
  - Can be totally unrelated to row sampling. Affects the sampling/simulation error.
  - "Matching" P with $|A|$ is beneficial (has an effect like in importance sampling).
- Independent row and column sampling allows exploration at will! Resolves the exploration problem that is critical in approximate policy iteration.

LSTD-LIKE METHOD

- Optimality condition/equivalent form of projected equation

$$\sum_{i=1}^{n} \xi_i \phi(i) \left(\phi(i) - \sum_{j=1}^{n} a_{ij} \phi(j) \right)' r^* = \sum_{i=1}^{n} \xi_i \phi(i) b_i$$

- The two expected values are approximated by row and column sampling (batch $0 \to t$).

- We solve the linear equation

$$\sum_{k=0}^{t} \phi(i_k) \left(\phi(i_k) - \frac{a_{i_k j_k}}{p_{i_k j_k}} \phi(j_k) \right)' r_t = \sum_{k=0}^{t} \phi(i_k) b_{i_k}$$

- We have $r_t \to r^*$, regardless of ΠA being a contraction (by law of large numbers; see next slide).

- Issues of singularity or near-singularity of $I - \Pi A$ may be important; see the text.

- An LSPE-like method is also possible, but requires that ΠA is a contraction.

- Under the assumption $\sum_{j=1}^{n} |a_{ij}| \leq 1$ for all i, there are conditions that guarantee contraction of ΠA; see the text.

JUSTIFICATION W/ LAW OF LARGE NUMBERS

- We will match terms in the exact optimality condition and the simulation-based version.
- Let $\hat{\xi}_i^t$ be the relative frequency of i in row sampling up to time t.
- We have

$$\frac{1}{t+1}\sum_{k=0}^{t}\phi(i_k)\phi(i_k)' = \sum_{i=1}^{n}\hat{\xi}_i^t\phi(i)\phi(i)' \approx \sum_{i=1}^{n}\xi_i\phi(i)\phi(i)'$$

$$\frac{1}{t+1}\sum_{k=0}^{t}\phi(i_k)b_{i_k} = \sum_{i=1}^{n}\hat{\xi}_i^t\phi(i)b_i \approx \sum_{i=1}^{n}\xi_i\phi(i)b_i$$

- Let $\hat{p}_{ij}^t$ be the relative frequency of (i,j) in column sampling up to time t.

$$\begin{aligned}\frac{1}{t+1}\sum_{k=0}^{t}\frac{a_{i_k j_k}}{p_{i_k j_k}}\phi(i_k)\phi(j_k)' &= \sum_{i=1}^{n}\hat{\xi}_i^t\sum_{j=1}^{n}\hat{p}_{ij}^t\frac{a_{ij}}{p_{ij}}\phi(i)\phi(j)' \\ &\approx \sum_{i=1}^{n}\xi_i\sum_{j=1}^{n}a_{ij}\phi(i)\phi(j)'\end{aligned}$$

BASIS FUNCTION ADAPTATION I

- An important issue in ADP is how to select basis functions.

- A possible approach is to introduce basis functions parametrized by a vector θ, and optimize over θ, i.e., solve a problem of the form

$$\min_{\theta \in \Theta} F\big(\tilde{J}(\theta)\big)$$

where $\tilde{J}(\theta)$ approximates a cost vector J on the subspace spanned by the basis functions.

- One example is

$$F\big(\tilde{J}(\theta)\big) = \sum_{i \in I} |J(i) - \tilde{J}(\theta)(i)|^2,$$

where I is a subset of states, and $J(i)$, $i \in I$, are the costs of the policy at these states calculated directly by simulation.

- Another example is

$$F\big(\tilde{J}(\theta)\big) = \big\|\tilde{J}(\theta) - T\big(\tilde{J}(\theta)\big)\big\|^2,$$

where $\tilde{J}(\theta)$ is the solution of a projected equation.

BASIS FUNCTION ADAPTATION II

- Some optimization algorithm may be used to minimize $F\big(\tilde{J}(\theta)\big)$ over θ.

- A challenge here is that the algorithm should use low-dimensional calculations.

- One possibility is to use a form of random search (the cross-entropy method); see the paper by Menache, Mannor, and Shimkin (Annals of Oper. Res., Vol. 134, 2005)

- Another possibility is to use a gradient method. For this it is necessary to estimate the partial derivatives of $\tilde{J}(\theta)$ with respect to the components of θ.

- It turns out that by differentiating the projected equation, these partial derivatives can be calculated using low-dimensional operations. See the references in the text.

APPROXIMATION IN POLICY SPACE I

• Consider an average cost problem, where the problem data are parametrized by a vector r, i.e., a cost vector $g(r)$, transition probability matrix $P(r)$. Let $\eta(r)$ be the (scalar) average cost per stage, satisfying Bellman's equation

$$\eta(r)e + h(r) = g(r) + P(r)h(r)$$

where $h(r)$ is the differential cost vector.

• Consider minimizing $\eta(r)$ over r. Other than random search, we can try to solve the problem by a policy gradient method:

$$r_{k+1} = r_k - \gamma_k \nabla \eta(r_k)$$

• Approximate calculation of $\nabla\eta(r_k)$: If $\Delta\eta$, Δg, ΔP are the changes in η, g, P due to a small change Δr from a given r, we have

$$\Delta\eta = \xi'(\Delta g + \Delta P h),$$

where ξ is the steady-state probability distribution/vector corresponding to $P(r)$, and all the quantities above are evaluated at r.

APPROXIMATION IN POLICY SPACE II

• Proof of the gradient formula: We have, by "differentiating" Bellman's equation,

$$\Delta\eta(r)\cdot e+\Delta h(r) = \Delta g(r)+\Delta P(r)h(r)+P(r)\Delta h(r)$$

By left-multiplying with ξ',

$$\xi'\Delta\eta(r)\cdot e+\xi'\Delta h(r) = \xi'\Big(\Delta g(r)+\Delta P(r)h(r)\Big)+\xi' P(r)\Delta h(r)$$

Since $\xi'\Delta\eta(r)\cdot e = \Delta\eta(r)$ and $\xi' = \xi' P(r)$, this equation simplifies to

$$\Delta\eta = \xi'(\Delta g + \Delta P h)$$

• Since we don't know ξ, we cannot implement a gradient-like method for minimizing $\eta(r)$. An alternative is to use "sampled gradients", i.e., generate a simulation trajectory $(i_0, i_1, \ldots)$, and change r once in a while, in the direction of a simulation-based estimate of $\xi'(\Delta g + \Delta P h)$.

• Important Fact: $\Delta\eta$ can be viewed as an expected value!

• Much research on this subject, see the text.

6.231 DYNAMIC PROGRAMMING

OVERVIEW-EPILOGUE

- Finite horizon problems
  - Deterministic vs Stochastic
  - Perfect vs Imperfect State Info
- Infinite horizon problems
  - Stochastic shortest path problems
  - Discounted problems
  - Average cost problems

FINITE HORIZON PROBLEMS - ANALYSIS

- Perfect state info
  - A general formulation - Basic problem, DP algorithm
  - A few nice problems admit analytical solution
- Imperfect state info
  - Reduction to perfect state info - Sufficient statistics
  - Very few nice problems admit analytical solution
  - Finite-state problems admit reformulation as perfect state info problems whose states are prob. distributions (the belief vectors)

FINITE HORIZON PROBS - EXACT COMP. SOL.

- Deterministic finite-state problems
  - Equivalent to shortest path
  - A wealth of fast algorithms
  - Hard combinatorial problems are a special case (but # of states grows exponentially)
- Stochastic perfect state info problems
  - The DP algorithm is the only choice
  - Curse of dimensionality is big bottleneck
- Imperfect state info problems
  - Forget it!
  - Only small examples admit an exact computational solution

FINITE HORIZON PROBS - APPROX. SOL.

- Many techniques (and combinations thereof) to choose from
- Simplification approaches
  - Certainty equivalence
  - Problem simplification
  - Rolling horizon
  - Aggregation - Coarse grid discretization
- Limited lookahead combined with:
  - Rollout
  - MPC (an important special case)
  - Feature-based cost function approximation
- Approximation in policy space
  - Gradient methods
  - Random search

INFINITE HORIZON PROBLEMS - ANALYSIS

- A more extensive theory
- Bellman's equation
- Optimality conditions
- Contraction mappings
- A few nice problems admit analytical solution
- Idiosynchracies of problems with no underlying contraction
- Idiosynchracies of average cost problems
- Elegant analysis

INF. HORIZON PROBS - EXACT COMP. SOL.

- Value iteration
  - Variations (Gauss-Seidel, asynchronous, etc)
- Policy iteration
  - Variations (asynchronous, based on value iteration, optimistic, etc)
- Linear programming
- Elegant algorithmic analysis
- Curse of dimensionality is major bottleneck

INFINITE HORIZON PROBS - ADP

- Approximation in value space (over a subspace of basis functions)
- Approximate policy evaluation
  - Direct methods (fitted VI)
  - Indirect methods (projected equation methods, complex implementation issues)
  - Aggregation methods (simpler implementation/many basis functions tradeoff)
- Q-Learning (model-free, simulation-based)
  - Exact Q-factor computation
  - Approximate Q-factor computation (fitted VI)
  - Aggregation-based Q-learning
  - Projected equation methods for opt. stopping
- Approximate LP
- Rollout
- Approximation in policy space
  - Gradient methods
  - Random search