Part 4: Markov Decision Processes

Aim: This part covers discrete time Markov Decision processes whose state is completely observed. The key ideas covered is *stochastic dynamic programming*. We apply stochastic dynamic programming to solve fully observed Markov decision processes (MDPs). Later we will tackle Partially Observed Markov Decision Processes (POMDPs).

Issues such as general state spaces and measurability are omitted. Instead we focus on structural aspects of stochastic dynamic programming.

History

- *Classical control* (Freq. domain): Root locus, Bode diagrams, stability, PID, 1950s.
- State Space Theory - Kalman 1960s
  Modern Control (Time domain): State variable feedback, observability, controllability
- Optimal and Stochastic Control 1960s – 1990s
  - Dynamic Programming (Bellman)
  - LQ and Markov Decision Processes (1960s)
  - Partially observed Stochastic Control = Filtering + control
  - Stochastic Adaptive Control (1980s & 1990s)
  - Robust stochastic control H^∞ control (1990s)
  - Scheduling control of computer networks, manufacturing systems (1990s).
  - Neurodynamic programming (Re-inforcement learning) 1990s.

Applications

- Control in Telecom and Sensor Networks: Admission, Access and Power control – wireless networks, computer networks.
- Sensor Scheduling and Optimal search.
- Robotic navigation and Intelligent Control.
- Process scheduling and manufacturing
- Aeronautics: Auto-pilots, missile guidance systems, satellite navigation systems

1 Fully Observed MDP

1. **Discrete-time dynamic system**: State $\{x_k\} \in \mathcal{X}$. For time $k = 0, 1, \ldots, N$ evolves as

$$x_{k+1} = A_k(x_k, u_k, w_k) \quad x_0 \sim \pi_0(\cdot)$$

Observations: $y_k = x_k$

Control: $u_k \in U_k(x_k)$. Process Noise: w_k iid

2. **Policy class**: Consider *admissible* policy

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

where $u_k = \mu_k(x_k) \in U_k(x_k)$.

3. **Cost function**: additive cost function

$$J_\pi(x_0) = \mathbb{E}\left\{ c_N(x_N) + \sum_{k=0}^{N-1} c_k(x_k, \mu_k(x_k)) \right\} \qquad (1)$$

c_N denotes terminal cost.

Aim: Compute the optimal policy

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

Π is set of admissible policies.

$J^*(x_0)$ is called optimal cost (value) function.

Terminology

Finite Horizon: N finite.

Fully observed: $y_k = x_k$

Partially observed: $y_k = C_k(x_k, v_k)$ (next part)

Infinite horizon:

1. Average cost:

$$J_\pi(x_0) = \lim_{N\to\infty} \frac{1}{N}\mathbb{E}\left\{\sum_{k=0}^{N-1} c(x_k, \mu(x_k))\right\}$$

2 Discounted cost: $\rho \in (0,1)$

$$J_\pi(x_0) = \mathbb{E}\left\{\sum_{k=0}^{\infty} \rho^k\, c(x_k, \mu(x_k))\right\}$$

Remarks: 1. Average cost problems need more technical conditions and somewhat harder than discounted cost problems.

2. Stochastic dynamic optimization or Sequential decision making problem.

2 Application Examples

2.1 Finite state Markov Decision Processes (MDP)

x_k is a S state Markov chain.
Transition prob: $P_{ij}(u) = P(x_{k+1} = j | x_k = i, u_k = u)$,
$i, j \in \{1, \ldots, S\}$.
Cost function as in (1).

Numerous applications in OR, EE, Gambling theory.

Benchmark Example: Machine (or Sensor) Replacement
State: $x_k \in \{0, 1\}$ – machine state
$x_k = 0$ operational; $x_k = 1$ failed.

Control: $u_k \in \{0, 1\}$.
$u_k = 0$ keep machine; $u_k = 1$ replace by new one

Trans prob matrices: Let $\theta = P(x_{k+1} = 1 \mid x_k = 0)$.

$$P(0) = \begin{bmatrix} 1-\theta & \theta \\ 0 & 1 \end{bmatrix}, \quad P(1) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

Cost: Minimize $\mathbb{E}\{\sum_{k=0}^{N-1} c(x_k, u_k)\}$
where $c(0,0) = 0$, $c(1,0) = C$, $c(x,1) = R$

2.2 Other fully observed problems

1. Linear Quadratic (LQ) control Fully observed problem

$$x_{k+1} = A_k x_k + B_k u_k + w_k$$

$$J_\pi(x_0) = \mathbb{E}\left\{ x_N' \, Q_N \, x_N + \sum_{k=0}^{N-1} u_k' R_k u_k + x_k' Q_k x_k \right\}$$

$Q_k \geq 0$ and $R_k > 0$.
w_k is zero mean finite variance white noise (not necessarily Gaussian)

For detailed analysis and design examples, see Anderson and Moore.

1. Linear control methods have explicit solutions
2. Maybe applied to nonlinear systems operating on a *small signal basis* – linearization
3. Selection of Q and R involve engineering judgment.
4. Partially observed problem - LQG control is widely used.

2. Optimal Stopping Problems (termination control)

Asset Selling problem: You want to sell an asset.

Offers w_0, w_1, $\ldots$, w_{N-1} are iid.

If offer k accepted: invest w_k at fixed interest rate r.

If you reject offer, wait till next offer. Rejected offers cannot be renewed

Offer $N-1$ must be accepted if other offers were rejected.

Aim: What is optimal policy for accepting and rejecting?

Formulation: w_k: offer value – real valued (say).

Control space: accept (sell) u^1, reject (dont sell) u^2

State space: reals + T (termination state)

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

$$= \begin{cases} T & \text{if } x_k = T \text{ or } x_k \neq T \text{ and } u_k = u^1 \\ w_k & \text{otherwise} \end{cases}$$

$$\text{Reward: } \mathbb{E}\left\{ c_N(x_N) + \sum_{k=0}^{N-1} c_k(x_k, u_k, w_k) \right\}$$

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

$$c_k(x_k, u_k, w_k) = \begin{cases} (1+r)^{N-k} x_k & \text{if } x_k \neq T \\ 0 & \text{otherwise} \end{cases}$$

Rational Thief problem: Thief can chose night k to retire with earnings x_k or rob a house and bring w_k.

Thief is caught with probability p. If caught, lose all money.

Assume w_k are iid with mean $\bar{w}$. Compute optimal policy over N nights.

3. Scheduling Problems: Job scheduling on single processor: N jobs to be done in sequential order.

Job i requires time random T_i. T_i are iid.

If job i is completed at time t, reward is $\rho^t R_i$, $0 < \rho < 1$.

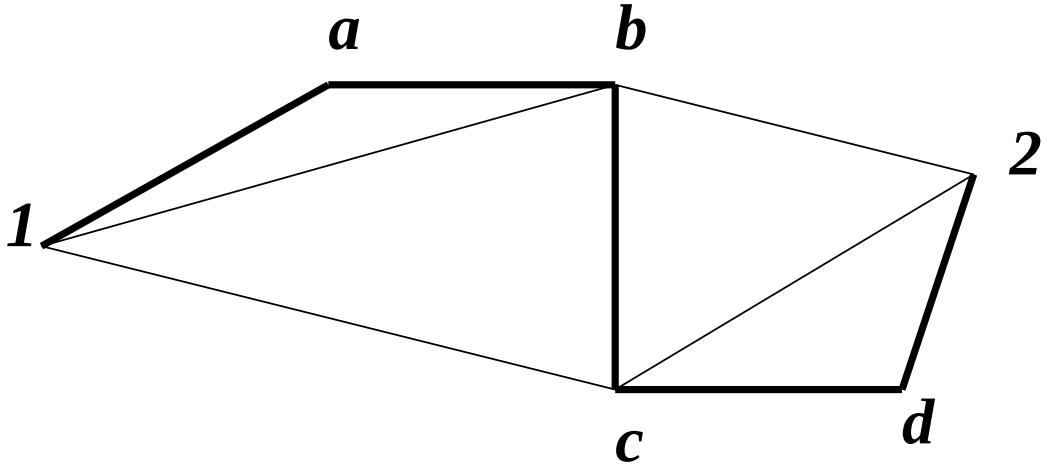
Find schedule that maximizes total reward.

See Bertsekas or Ross or Puterman for a wealth of examples. Journals such as IEEE Auto Control, Machine Learning, Annals of Operations Research.

3 Dynamic Programming (DP)

3.1 Principle of Optimality

(Bellman 1962). An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first selection.



If shortest distance from **1** *to* **2** *is via* **a,b,c,d,** *then shortest distance from* **b** *to* **2** *is via* **c,d**.

DP is an algorithm which uses the principle of optimality to determine optimal policy.
DP is widely used control, OR, discrete optimization.
DP yields a functional equation of the form:

$$J_k(x) = \min_u \left[c_k(x,u) + \int_{\mathbb{R}} J_{k+1}(A_k(x,u,w)) p(w) dw \right]$$

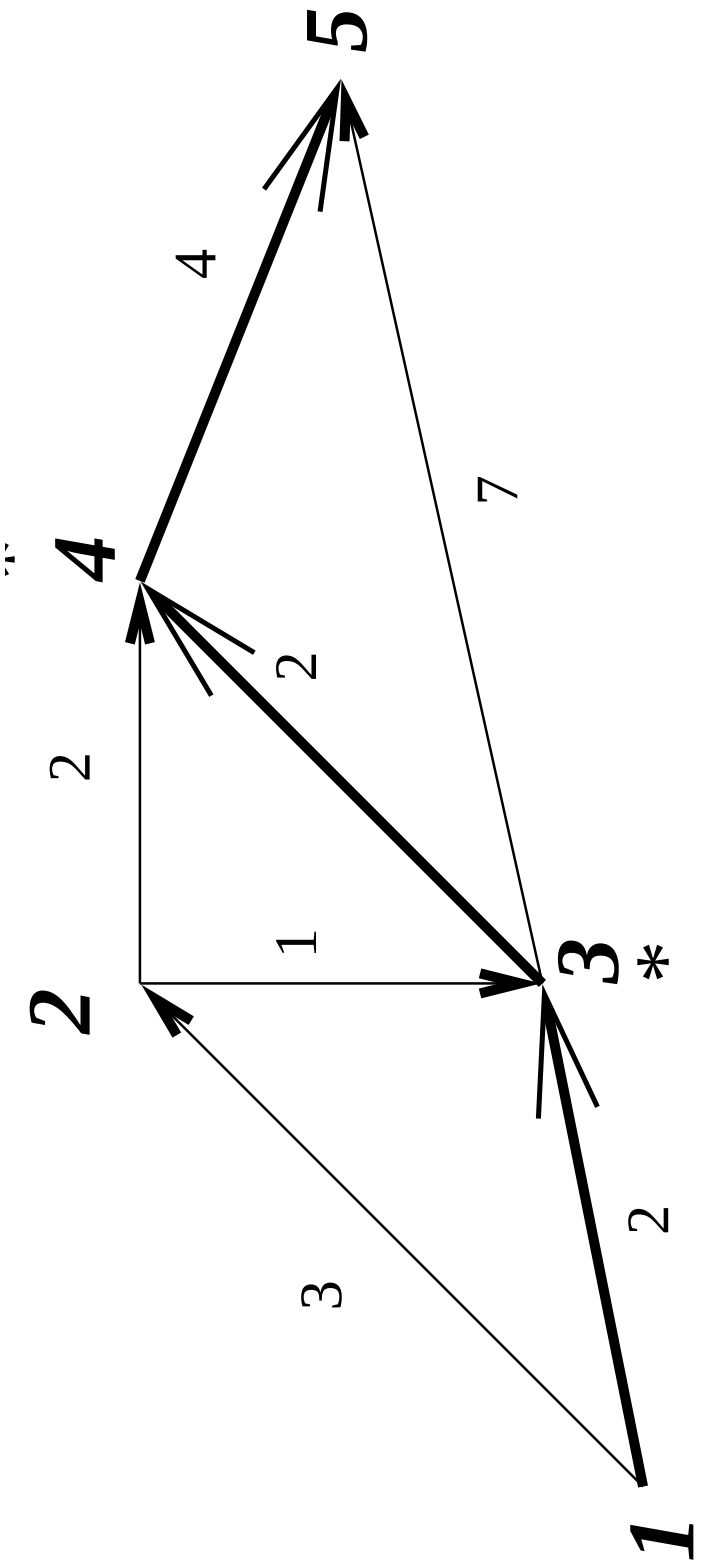


Figure 1: Deterministic Shortest path problem

3.2 DP for Shortest path problem

Compute shortest distance and path from node 1 to 5.

Let c_{ij} = distance from node i to j.

$c_{12} = 3$, $c_{13} = 2$, $c_{23} = 1$, $c_{24} = 2$,

$c_{34} = 2$, $c_{35} = 7$, $c_{45} = 4$.

Define:

J_i = min dist from node i to 5.

u_i^* = next point in min path from i to 5.

DP yields:

$$
\begin{aligned}
J_5 &= 0 \\
J_4 &= 4; \qquad u_4^* = 5. \\
J_3 &= \min_{u\in\{4,5\}} c_{3u} + J_u^* \\
&= \min\{c_{34} + J_4^*, c_{35} + J_5^*\} \\
&= \min\{2+4, 7\} = 6; \quad u_3^* = 4. \\
J_2 &= \min_{u\in\{3,4\}} c_{2u} + J_u^* \\
&= \min\{c_{23} + J_3^*, c_{24} + J_4^*\} \\
&= \min\{1+6, 2+4\} = 6; \quad u_2^* = 4. \\
J_1 &= \min\{c_{12} + J_2^*, c_{13} + J_3^*\} \\
&= \min\{3+6, 2+6\} = 8; \quad u_1^* = 3.
\end{aligned}
$$

Bactracking: shortest path from 1 to 5 is via $3, 4$.

Remarks on DP

1. We have worked backwards from node 5 to node 1. Called Backward DP
Forward DP (FDP) yields identical result.

2. Minimum path problem arises in optimal network flow design, critical path design, Viterbi algorithm.

3. It is important to note that the cost (distance) between nodes is path independent. If it is path dependent, DP does not yield optimal soln.

4. Backtracking is a characteristic of DP.
"Life can only be understood going backwards; but it must be lived going forwards"

5. We will focus on solving stochastic control problems via DP. (SDP)

6. Proof on principle of optimality is straightforward. (proof by contradiction).

4 Stochastic Dynamic Programming

4.1 Principle of Optimality

Stochastic Control version: Consider fully observed problem with cost function

$$J_\pi(x_0) = \mathbb{E}\left\{ c_N(x_N) + \sum_{k=0}^{N-1} c_k(x_k, \mu_k(x_k)) \mid x_0 \right\}$$

Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be optimal policy.
Suppose state at time i is x_i when using π^*.
Consider subproblem of minimizing

$$\mathbb{E}\left\{ c_N(x_N) + \sum_{k=i}^{N-1} c_k(x_k, \mu_k(x_k)) \mid x_i \right\}$$

Then $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$ is optimal for this subproblem.

4.2 Soln via Stochastic Dynamic Programming

Recall fully observed system is

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

Aim: Determine optimal policy to minimize $J_\pi(x_0)$

$$J_\pi(x_0) = \mathbb{E}\left\{ c_N(x_N) + \sum_{k=0}^{N-1} c_k(x_k, \mu_k(x_k)) \mid x_0 \right\}$$

Outline: The solution consists of two stages:

1. Backwards SDP to determine optimal policy – this is data independent (offline). Creates a lookup table – for state x_k it gives optimal u_k.
2. Forward implementation of controller: Given x_k pick optimal u_k – table lookup.

k	opt control if $x_k = e_1$	opt control if $x_k = e_2$
1	$u^*_{1,1}$	$u^*_{2,1}$
2	$u^*_{1,2}$	$u^*_{2,2}$
$\vdots$	$\vdots$	$\vdots$
$N-1$	$u^*_{1,N-1}$	$u^*_{2,N-1}$

SDP solution

Given

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

$$J_\pi(x_0) = \mathbb{E}\left\{ c_N(x_N) + \sum_{k=0}^{N-1} c_k(x_k, \mu_k(x_k)) \mid x_0 \right\}$$

For every initial state x_0, the optimal cost
$J^*(x_0) = \inf_\pi J_\pi(x_0) = J_0(x_0)$.
Here $J_0(x_0)$ is given by the last step of the backward DP algorithm for $k = N, N-1, \ldots, 0$:

$$J_N(x_N) = c_N(x_N) \qquad \text{(terminal cost)}$$

$$\begin{aligned} J_k(x) &= \min_{u_k \in U_k(x)} \left[c_k(x, u_k) + \mathbb{E}\left\{ J(x_{k+1}) | x_k = x \right\} \right], \\ &= \min_{u \in U_k(x)} \left[c_k(x, u) + \int_{\mathbb{R}} J_{k+1}(A_k(x, u, w)) p(w) dw \right] \end{aligned}$$

$$\mu_k^*(x) = \operatorname*{argmin}_{u_k \in U_k(x)} \left[c_k(x, u) + \int_{\mathbb{R}} J_{k+1}(A_k(x, u, w)) p(w) dw \right].$$

$$J_k(x) \stackrel{\text{defn}}{=} \min_{\mu_k, \ldots, \mu_N} \mathbb{E}\{ \sum_{t=k}^{N} c_t(x_t, u_t) | x_k = x \}$$

is called value- to-go function.

Remarks on SDP

Mathematical rigor: If $w_k \in D_k$ where D_k is countable, then above results are rigorous.

$$J_k(x) = \min_{u \in U_k(x)} \left[c_k(x,u) + \sum_{w \in D_k} J_{k+1}(A_k(x,u,w)) p(w) \right]$$

Otherwise, the results are "informal" in that we have not specified the function spaces to which u, x and w belong to. For general case measurable selection theorems are required.
(i) Need $c_k(x,u)$ to be a measurable function
(ii) For min to replace inf need compactness and lower semi-continuity.

General conditions (Hernandez Lerma & Lasserre).
(a) $U_k(x)$ compact, $c_k(x,\cdot)$ is lower semi-cont on $U_k(x)$ for all $x \in X$.
(b) $\int_{\mathbb{R}} J_{k+1}(A_k(x,u,w)) p(w) dw$ is lower semi-cont on $U_k(x)$ for every $x \in X$ and every cont bounded function J_{k+1} on X. (does not work for LQ!)
(a) can be replaced by inf-compactness of $c_k(x,u)$:
For $x \in X$, $r \in \mathbb{R}$, $\{u \in U_k(x) | c_k(x,u) \leq r\}$ compact.

5 Finite State Markov Decision Processes (MDP)

Markov chain $x_k \in \{1, 2, \ldots, S\}$
$P(x_{k+1} = j | x_k = i, u_k = u) = P_{ij}(u)$,
$i, j \in \{1, \ldots, S\}$.
Aim: Minimize

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

DP yields: For $i = 1, 2, \ldots, S$

$$J_N(i) = c_N(i),$$

$$J_k(i) = \min_{u_k} \left[c_k(i, u_k) + \mathbb{E}\left\{ J(x_{k+1}) | x_k = i \right\} \right],$$

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

$$= \min_{u_k} \left[c_k(i, u_k) + \sum_{j=1}^{S} J_{k+1}(j)\, P(x_{k+1} = j \mid x_k = i, u_k) \right],$$

$$= \min_{u_k} \left[c_k(i, u_k) + \sum_{j=1}^{S} J_{k+1}(j)\, P_{ij}(u_k) \right],$$

In matrix vector notation:

$$J_k = \min_u \left[c_k(u) + P(u)\, J_{k+1} \right]$$

Lookup Table

Dynamic programming creates a look-up table

$$J_k(i) = \min_{u_k} \left[c_k(i, u_k) + \sum_{j=1}^{S} J_{k+1}(j)\, P_{ij}(u_k) \right]$$

$$u^*_{i,k} = \mu^*_k(x_k = i) \stackrel{\text{defn}}{=} \arg\min_{u_k} \left[c_k(i, u_k) + \sum_{j=1}^{S} J_{k+1}(j)\, P_{ij}(u_k) \right]$$

Thus we have a look-up table

k	opt control if $x_k = 1$	opt control if $x_k = 2$
1	$u^*_{1,1}$	$u^*_{2,1}$
2	$u^*_{1,2}$	$u^*_{2,2}$
3	$u^*_{1,3}$	$u^*_{2,3}$
$\vdots$	$\vdots$	$\vdots$
$N-2$	$u^*_{1,N-2}$	$u^*_{2,N-2}$
$N-1$	$u^*_{1,N-1}$	$u^*_{2,N-1}$

Remarks: (i) Requires $O(N\,S)$ memory.

(ii) If $N = \infty$ (infinite horizon), and if $c_k(x_k, u_k) = \rho^k c(x_k, u_k)$, then entries $u^*_{i,k}$ in lookup table converge to values indpt of k. Steady (stationary) state policy; $O(S)$ memory

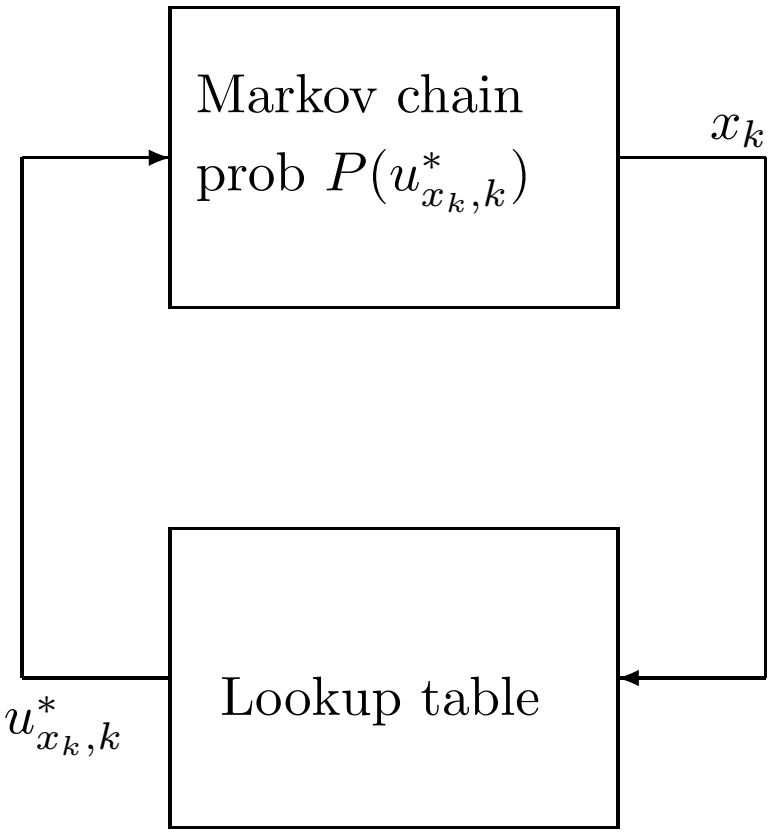
Controller Implementation:

1. Set $k = 0$, Initial condition: $x_0 = i$.

2. Select optimal control as $u^*_{x_k,k} = \mu_k(x_k)$ from DP lookup table
Instantaneous cost $= c_k(x_k, u^*_{x_k,k})$

3. Markov chain evolves randomly according to $P(u^*_{x_k,k})$. Generates new state x_{k+1}.

4. If $k = N - 1$, stop.
Else, set $k = k + 1$ and go to step 2.



Design Example: Machine Replacement

Recall that $x_k \in \{0,1\}$ and $u_k \in \{0,1\}$.

$$P(0) = \begin{bmatrix} 1-\theta & \theta \\ 0 & 1 \end{bmatrix}, \quad c(0) = \begin{bmatrix} 0 \\ c \end{bmatrix}$$

$$P(1) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad c(1) = \begin{bmatrix} R \\ R \end{bmatrix}$$

Cost: Minimize $\mathbb{E}\{\sum_{k=0}^{N-1} c(x_k, u_k)\}$

DP Soln for Control Policy:

$$\begin{bmatrix} J_N(1) \\ J_N(2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

For $k = N-1, N-2, \ldots, 1, 0$

$$\begin{bmatrix} J_k(1) \\ J_k(2) \end{bmatrix} = \min_{u \in \{0,1\}} [c(u) + P(u)\, J_{k+1}]$$

$$= \min \left\{ \begin{bmatrix} (1-\theta) J_{k+1}(1) + J_{k+1}(2) \\ c + J_{k+1}(2) \end{bmatrix}, \begin{bmatrix} R + J_{k+1}(1) \\ R + J_{k+1}(1) \end{bmatrix} \right\}$$

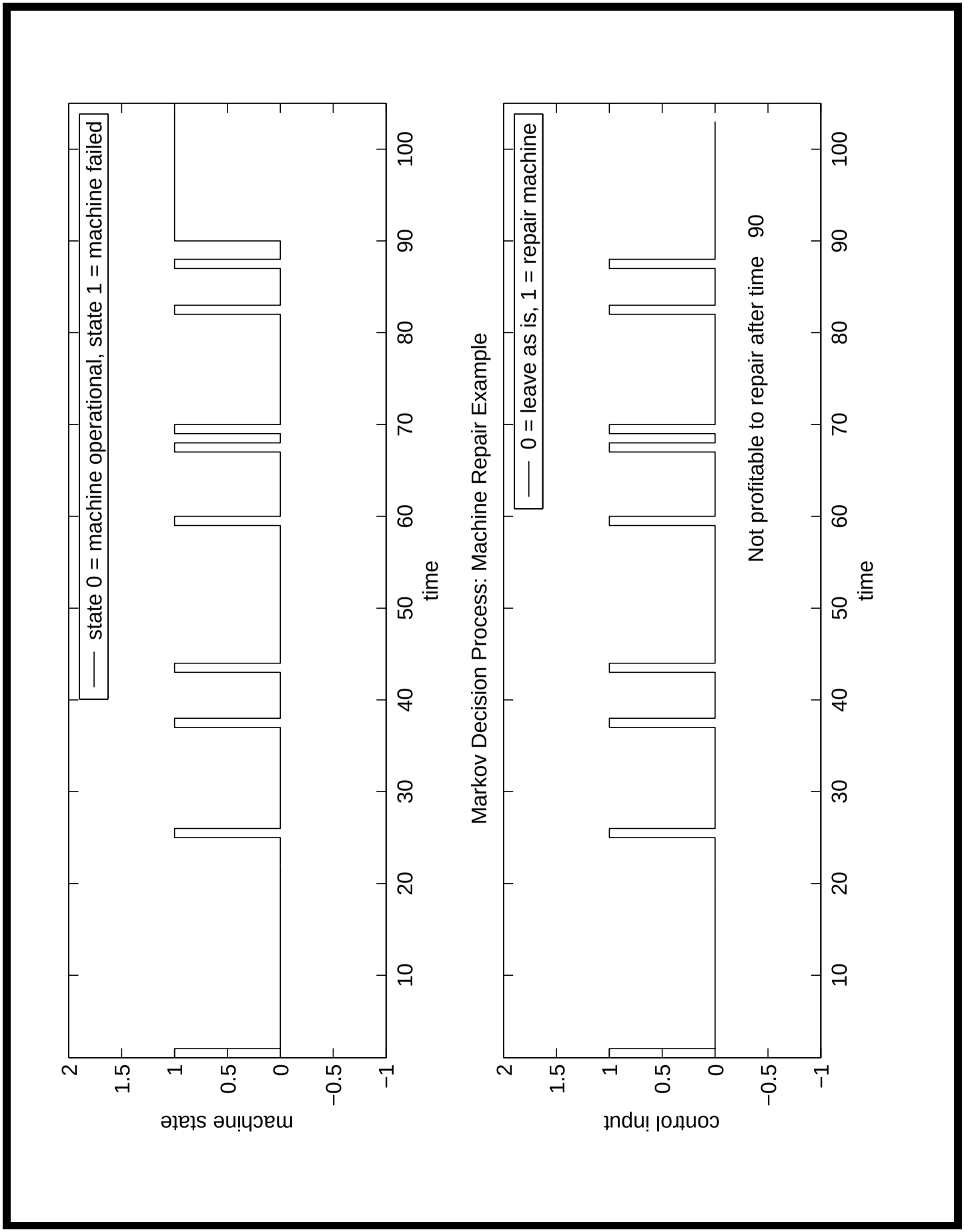
Design Example: Continued

e.g. if $N = 4$, $\theta = 0.1$, $c = 4$, $R = 3$:

k	0	1	2	3	4
$J_k(1)$	0.8430	0.5700	0.3000	0	0
$J_k(2)$	3.5700	3.3000	3.0000	3.0000	0
$u^*_{1,k}$	0	0	0	0	
$u^*_{2,k}$	1	1	1	1	

e.g. if $N = 4$, $\theta = 0.1$, $c = 4$, $R = 6$:

k	0	1	2	3	4
$J_k(1)$	1.4130	0.8700	0.3000	0	0
$J_k(2)$	6.8700	6.3000	6.0000	3.0000	0
$u^*_{1,k}$	0	0	0	0	
$u^*_{2,k}$	1	1	1	0	


Markov Decision Process: Machine Repair Example
state 0 = machine operational, state 1 = machine failed
machine state
time
0 = leave as is, 1 = repair machine
control input
Not profitable to repair after time 90
time

Remarks:

1. Note DP minimizes expected cost. Actual cost of a sample path is a random variable which may be lower than the expected value.

2. In the machine repair example, there is a threshold after which it becomes infeasible to repair the machine.

6 Perspective

1. DP is a widely used optimization algorithm in:
Stochastic control, combinatorial optimization,
operations research.
We have looked at Backward DP
Forward DP is similar – yields identical result
e.g. Viterbi algorithm is a shortest path algorithm.

2. DP for fully observed stochastic control.
LQ and MDP have explicit solutions.
Most other problems do not.

3. We considered additive cost functions.
Risk sensitive control considers exponential cost
functions, see Elliott et.al.

4. Why feedback control is essential (next slide)

Things not covered:
1. Engineering LQ control – Anderson & Moore
2. Detailed mathematics – Bertsekas & Shreve
3. Numerical approximations for solving DP.
4. Properties of value-to-go function – Ross

7 Why Feedback Control is essential

1. Open loop systems are a special case of closed loop systems for both deterministic and stochastic systems

2. In deterministic systems, for every closed loop systems, there is an equivalent open loop system,
Example: Linear case.

3. For stochastic systems closed loop (feedback) and open loop systems are not equivalent.
We show that for a stochastic system:
(i) no open loop system has the same properties of a feedback system
(ii) Feedback always achieves a better cost that open loop in optimal control.

Feedback is essential in stochastic systems

Consider stochastic system $x_{k+1} = x_k + u_k + w_k$ where x_0, w_k are white noise with variance σ^2.

Closed loop: Suppose feedback is $u_k = -x_k$.
Then $x_{k+1} = w_k$

Open loop: $x_{k+1} = x_k + u_k + w_k$
$= x_0 + \sum_{m=0}^{k} u_m + \sum_{m=0}^{k} w_m$
So mean is $\mathbb{E}\{x_k\} = \sum_{m=0}^{k-1} u_m$ and
variance is $\mathbb{E}\{x_0^2\} + \sum_m \mathbb{E}\{w_m^2\} = (k+1)\sigma^2$

No open loop system can produce $x_k = w_{k-1}$.

Cost: Suppose $J = \mathbb{E}\{\sum_{k=0}^{N} x_k^2\}$.
Closed loop: $J = \mathbb{E}\{x_0^2 + \sum_n w_k^2\} = (k+1)\sigma^2$
Open loop: $J = \sum_k \mathbb{E}\{(x_0 + \sum_m^k u_m + \sum_m^k w_m)^2\}$
$\geq \frac{1}{2}(k+1)(k+2)\sigma^2$

Feedback is superior to any open loop control

8 Infinite horizon results

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

$$J_\pi(x_0) = \lim_{N\to\infty} \mathbb{E}\left\{ \sum_{k=0}^{N-1} \rho^k c(x_k, \mu_k(x_k)) \right\}$$

$0 < \rho < 1$ is discount factor.

Admissible policies: $\pi = \{\mu_0, \mu_1, \ldots, \}$ where $u_k = \mu_k(x_k)$. Optimal cost is

$$J^*(x) = \min_{\pi\in\Pi} J_\pi(x)$$

Define class of *stationary* policies: $\pi = \{\mu, \mu, \ldots\}$.

To simplify notation call J_π as J_μ.

Require $J_\pi(x_0)$ to be finite. Examples include

(i) Stochastic shortest path: $\rho = 1$, cost free termination state. Termination is inevitable.

(ii) Discounted problems with bounded cost $|c(x, u)| \leq M$

We will only consider discounted problems. Then

$$J_\pi(x_0) = \mathbb{E}\left\{ \sum_{k=0}^{\infty} \rho^k c(x_k, \mu_k(x_k)) \right\}$$

8.1 DP for finite horizon version

Consider minimizing cost

$$\mathbb{E}\left\{\rho^n J(x_N) + \sum_{k=0}^{N-1} \rho^k c(x_k, u_k)\right\}$$

DP recursion yields for $k = 0, \ldots, N-1$:

$$J_k(x) = \min_u \mathbb{E}\{\rho^{N-k} c(x,u) + J_{k+1}(A(x,u,w))\}$$

initialized by $J_N(x) = \rho^N J(x)$. Define

$$V_k(x) = \frac{J_{N-k}(x)}{\rho^{N-k}}$$

Then DP can be written for $k = 0, 1, \ldots, N-1$ as

$$V_{k+1}(x) = \min_u \mathbb{E}\left\{c(x,u) + \rho V_k(A(x,u,w))\right\}$$

8.2 Main Result

$\lim_{k\to\infty} V_k(x) = V^*(x)$ where V^* is optimal value function for infinite horizon.

Bellman's equation holds

$$\boxed{V^*(x) = \min_u \mathbb{E}\left\{c(x,u) + \rho V^*(A(x,u,w))\right\}}$$

Define the operator

$$(TV)(x) \stackrel{\text{defn}}{=} \min_u \mathbb{E}\left\{c(x,u) + \rho V(A(x,u,w))\right\}$$

for any $V(x)$. Then Bellman's eqn is:

$$TV^* = V^*$$

T is monotonic: Suppose V and V' are s.t. $V(x) \leq V'(x)$ for all x. Then

$$\boxed{(TV)(x) \leq (TV')(x) \qquad \forall x}$$

Define also for any stationary policy μ

$$(T_\mu)V(x) = \mathbb{E}\left\{c(x,\mu(x)) + \rho V(A(x,\mu(x),w))\right\}$$

Result: (Bertsekas Vol.2, pg.12.) For every stationary policy μ, the associated cost satisfies

$$V_\mu = T_\mu V_\mu$$

This result means: For any stationary policy μ, the policy cost V_μ can be computed by solving $V_\mu = T_\mu V_\mu$. In finite state MDP case, V_μ can be computed exactly since $T_\mu V_\mu = c(\mu) + \rho P(\mu) V_\mu$. Thus

$$V_\mu = c(\mu) + \rho P(\mu) V_\mu \implies [I - \rho P(\mu)]\, V_\mu = c(\mu)$$

Note: Bellman's equation is a functional equation. It can rarely be solved explicitly.

8.3 Infinite Horizon MDPs: Numerical methods

Bellman's equation ($V^* = TV^*$) for finite state MDP is

$$V^*(i) = \min_u \left[c(i,u) + \rho \sum_{j=1}^{S} P_{ij}(u) V^*(j) \right]$$

In vector notation

$$V^* = \min_u \left[c(u) + \rho P(u) V^* \right]$$

where V^* and $c(u)$ are S dim vectors.

Recall optimal policy μ^* for MDP allocates $u_k^* = \mu^*(x_k)$, i.e. we need to construct the one row lookup table

k	opt control if $x_k = 1$	opt control if $x_k = 2$
Any k	u_1^*	u_2^*

1. *Linear Programming*: Since $\lim_{N\to\infty} T^N V = V^*$ for all V, we have using monotonicity of T:

$$V \le TV \implies V \le V^* = TV^*$$

Thus V^* is largest V that satisfies $V \le TV$.

$$\max \lambda' 1$$
$$\text{s.t. } \lambda \le c(u) + \rho P(u) \lambda$$

LP with $S|U|$ constraints. In queuing problems (that satisfy "conservation laws") these form a polymatroid.

2. *Value Iteration*: Successive iteration method for solving $V^* = TV^*$, i.e., a finite horizon approximation.
Initialize V_0. Then successive approximation procedure is

$$V_{k+1} = TV_k \text{ i.e. } V_{k+1} = \min_u [c(u) + \rho P(u) V_k]$$

Contraction mapping type proof of convergence

$$\text{One can show } \|V_k - V^*\|_\infty \leq \frac{2\rho}{1-\rho} \|V_k - V_{k-1}\|_\infty$$

3. *Policy Iteration*: For any stationary policy μ recall

$$T_\mu V = [c(u) + \rho P(u) V]$$

for any V. Recall cost function corresponding to μ, i.e., V_μ satisfies

$$T_\mu V_\mu = V_\mu$$

This means that for any stationary policy μ we can solve for V_μ:

$$V_\mu = c(\mu) + \rho P(\mu) V_\mu \implies (I - \rho P(\mu)) V_\mu = c(\mu)$$

Policy Iteration algorithm: Initialize μ^0 arbitrarily

Iterations: (i) Policy evaluation: V_{μ^k} is solution of linear equation

$$[I - \rho P(\mu^k)] V_{\mu^k} = c(\mu^k)$$

(ii) Policy improvement : $u^{k+1} = \min_u [c(u) + \rho P(u) V_k]$

Structural Results

When is optimal policy monotone in state?

Two concepts: submodularity, stochastic orders.

Submodular: $\phi(x,u)$ is submodular in (x,u) if

$$\phi(x,u+1)-\phi(x,u) \geq \phi(x+1,u+1)-\phi(x+1,u).$$

Examples: The following are submodular in (x,u)
(i): $\phi(x,u) = -xu$.
(ii) $\phi(x)$ or $\phi(u)$ is trivially submodular.
(iii) $\max(x,u)$
(iv) The sum of submodular functions is submodular.

Theorem [Topkis] Consider $\phi : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. If $\phi(x,u)$ is submodular, then $u^*(x) = \text{argmin}_u \, \phi(x,u) \uparrow x$.

First order stochastic dominance: Then π_1 first order stochastically dominates π_2 if $\sum_{i=j}^{X} \pi_1(i) \geq \sum_{i=j}^{X} \pi_2(i)$ for $j = 1, \ldots, X$. This is denoted as $\pi_1 \geq_s \pi_2$ or $\pi_2 \leq_s \pi_1$.

Example: $\pi_1 = [0.3,\ 0.2,\ 0.5]'$, $\pi_2 = [0.2,\ 0.4,\ 0.4]'$
not orderable.

Theorem: Let $\mathcal{V}$ denote the set of all X dimensional vectors v with nondecreasing components, i.e.,
$v_1 \leq v_2 \leq \cdots v_X$. Then $\pi_1 \geq_s \pi_2$ iff for all $v \in \mathcal{V}$,
$v'\pi_1 \geq v'\pi_2$.

Monotone Policies

(A1) $c(x, u, k) \downarrow x$.

(A2) $P_x(u) \leq_s P_{x+1}(u)$.

(A3) $c(x, u, k)$ is submodular in (x, u) That is:

$$c(x, u+1, k) - c(x, u, k) \downarrow x$$

(A4) $P(u)$ is tail supermodular:
$\sum_{j \geq l} (P_{xj}(u+1) - P_{xj}(u))$ is increasing in x.

Theorem: Assume that a finite horizon Markov decision process satisfies conditions (A1), (A2), (A3) and (A4).
Then $\mu_k^*(x) \uparrow x$.
Same proof applies for infinite horizon discounted cost and average cost.

$$Q_k(i,u) \stackrel{\text{defn}}{=} c(i,u,k) + J'_{k+1}P_i(u)$$

$$J_k(i) = \min_{u\in\mathcal{U}} Q_k(i,u), \quad \mu_k^*(i) = \operatorname*{argmin}_{u\in\mathcal{U}} Q_k(i,u)$$

where $J_{k+1} = \Big[J_{k+1}(1), \ldots, J_{k+1}(X)\Big]'$

Step 1. Assuming (A1) and (A2), $Q_k(i,u) \uparrow i$. Therefore $J_k(i) \uparrow i$.

Step 2. Assuming (A3) and (A4), $Q_k(i,u)$ is submodular.

Therefore $\mu_k^*(i) = \operatorname{argmin}_{u\in\mathcal{U}} Q_k(i,u) \uparrow i$.

Step 1: Use mathematical induction.
$Q_N(i,u) = c(i,N) \downarrow i$ by (A1). Suppose $Q_{k+1}(j,\bar{u}) \downarrow j$.
$J_{k+1}(j) = \min_{\bar{u}} Q_{k+1}(j,\bar{u}) \downarrow j$. Next $P_i(u) \leq_r P_{i+1}(u)$ by (A2). So $J'_{k+1}P_i(u) \geq J'_{k+1}P_{i+1}(u)$.
Finally $c(i,u,k) \downarrow i$ (A1),

$$c(i,u,k) + J'_{k+1}P_i(u) \geq c(i+1,u,k) + J'_{k+1}P_{i+1}(u).$$

Step 2: Consider $Q_k(i,u) = c(i,u,k) + J'_{k+1}P_i(u)$. By (A3), $c(i,u,k)$ is submodular. Applying (A4), since elements of J_{k+1} are decreasing, $J'_{k+1}P_i(u)$ is submodular.

How does Optimal Cost depend on Transition Matrix

Consider two MDPs with identical costs but different transition matrices P and $\bar{P}$.

(A1) $c(x,u,k) \downarrow x$.
(A2) $P_x(u) \leq_s P_{x+1}(u)$.
(A5) $P_x(u) \geq_s \bar{P}_x(u) \quad \forall x$.

Theorem. [Muller 1997]: Optimal cost incurred by policy $\mu^*(x;P)$ is smaller than that incurred by $\mu^*(x;\bar{P})$.

Proof:

$$Q_k(i,u) = c(i,u,k) + J'_{k+1}P_i(u)$$
$$\bar{Q}_k(i,u) = c(i,u,k) + \bar{J}'_{k+1}\bar{P}_i(u)$$

The proof is by induction. Clearly
$J_N(i) = \bar{J}_N(i) = c(i,N)$ for all $i \in \mathcal{X}$.

Suppose $J_{k+1}(i) \leq \bar{J}_{k+1}(i)$ for all $i \in \mathcal{X}$. Therefore $J'_{k+1}P_i(u) \leq \bar{J}'_{k+1}P_i(u)$. By (A1), (A2), $\bar{J}_{k+1}(i)$ is decreasing in i. By (A5), $P_i \geq_r \bar{P}_i$. Therefore $\bar{J}'_{k+1}P_i \leq \bar{J}'_{k+1}\bar{P}_i$. So
$c(i,u,k) + J'_{k+1}P_i(u) \leq c(i,u,k) + \bar{J}'_{k+1}\bar{P}_i(u)$ or equivalently, $Q_k(i,u) \leq \bar{Q}_k(i,u)$.

Neuro-Dynamic Programming Methods

The next two methods are simulation based. That is, although parameters are unknown, system can be simulated or observed under any choice of actions.

They form the core of re-inforcement learning or neuro-dynamic programming. The key idea in them is the Robbin's Munro stochastic approximation algorithm:

Result: Robbins Munro Algorithm.
Aim: Solve the algebraic equation $X = \mathbb{E}\{H(X)\}$ where H is a noisy function. That is we can measure samples $Y_n = H(X_n)$.

$$\textit{Algorithm} \quad \boxed{X_{n+1} = X_n + \gamma_n(Y_n - X_n)}$$

Key idea behind stochastic approx is to replace $\mathbb{E}\{H(X)\}$ by the sample $Y_n = H(X_n)$.

Remarks: The implicit assumption is that $\mathbb{E}\{H(X)\}$ cannot be computed in closed form – this is true when the density function is unknown. Step size $\gamma_n = 1/n$ typically.

Stochastic approximations are widely used in adaptive signal processing – e.g. adaptive filtering algorithms such as LMS and RLS algorithm. Recursive EM algorithm covered earlier is another example.

4. *Q-learning*: Simulation based. Define Q-factor

$$Q(i,u) = \left[c(i,u) + \rho \sum_{j=1}^{S} P_{ij}(u) V^*(j) \right]$$

From Bellman's equation this yields

$$Q(i,u) = \left[c(i,u) + \rho \sum_{j=1}^{S} P_{ij}(u) \min_{u'} Q(j,u') \right]$$

The trick above expresses Q as $\mathbb{E}\{\min(\cdot)\}$:

$$\boxed{Q(i,u) = \left[c(i,u) + \rho\, \mathbb{E}\{\min_{u'} Q(x_{k+1},u') | x_k = i\} \right]}$$

Hence can be solved via Robbins-Munro algorithm:

$$\begin{aligned} Q_{k+1}(i,u) = {} & Q_k(i,u) \\ & + \gamma \left(c(i,u) + \rho \min_{u'} \left(Q_k(j,u') \right) - Q_k(i,u) \right) \end{aligned}$$

Note: j is generated from (i,u) via simulation $\sim P_{ij}(u)$.

Remarks: (i) The above recursion does not require knowledge of $P(u)$.
(ii) Q learning is merely a stochastic approx algorithm!
(iii) NDP is widely used in Artificial intelligence where it is called Reinforcement learning.

5. *Temporal difference methods*: These can be used to compute by simulation the cost of a policy (details omitted).

Summary and Extensions

Stochastic Dynamic programming (SDP) involves solving a functional equation. This yields a (possibly infinite dimensional) lookup table.

There are 2 types of problems considered (i) Finite horizon
(ii) Infinite horizon – steady state controller.

For infinite horizon finite state MDPs there are several numerical algorithms: e.g. Policy iteration, value iteration, linear programming, neurodynamic programming.

We have not covered cont-time finite state MDPs. These arise in control of queuing systems – e.g. telecomms. By a process called "uniformization", a cont-MDP can be covered to an equivalent discrete-time MDP.

A generalization of cont-time MDPs are semi-Markov Decision processes. These are widely studied in discrete event systems.

Finally, MDPs with constraints can also be considered. Often the optimal policy is "randomized".