Lecture 3: November 4

Lecturer: Yishay Mansour *Scribe: Orit Mazor, Sigal Korczyn, Nimrod Hoofien*

3.1 Introduction

3.1.1 The Problem

In the following course, we will refrain mainly to discrete time problems, that is, at any given point in time $T = 1 \ldots N$ and only in those points the agent may perform one single action. If N is finite, we shall refer to these problems as having a *finite horizon*. When N is infinite, these problems will be referred to as having an *infinite horizon*. When discussing finite horizon problems, at time $T = N$ the agent is not allowed to perform an action and instead will be rewarded with an immediate end point profit according to the position it is in.

3.1.2 states and actions

Lets define S to be the set of states of the system. For each state $s \in S$ we define the set of actions A_s as the actions that the agent may perform in a state s. For simplicity, we will assume S and A are discrete and not time dependent.At any given state, an action can be performed deterministically (using a function mapping states to actions) or stochastically (randomly). When choosing an action stochastically we will define a distribution q(a) for every $a \in A_s$.

3.1.3 Immediate reward and the probability of transition

As a result of performing an action $a \in A_s$ in state $s \in S$ at time t:
1. The agent is rewarded an immediate reward $R_t(s, a)$. We define the expectation of R_t as $r_t(s, a) = E[R_t(s, a)]$.
2.The system transfers to a new state s', determined according to transition probability $P_t(s'|s, a)$. We assume that P_t is well defined, that is, that for every $s \in S$ and $a \in A_s$, $\sum_j P(j|s, a) = 1$.
We will not discuss how or when the immediate reward reach the agent. It may be accumulated in the time frame $[t, t+1]$, or alternatively, it can be given in a single point in time between t and $t+1$. In any case all that matters to the agent is that the immediate reward reaches it before t+1.

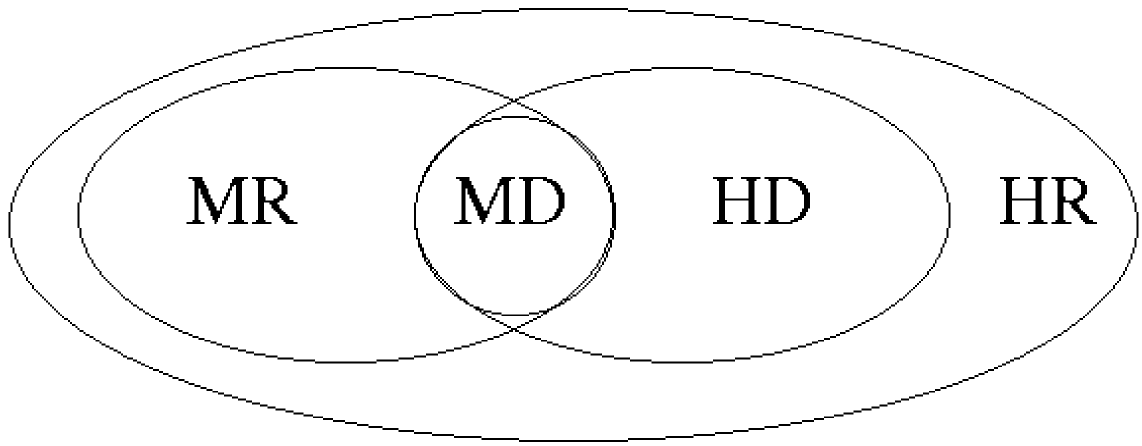

Figure 3.1: model diagram

A Markovian process is define as a process in which the only information needed from history is the current state. We define a Markovian process as (T, S, A, $P_t(.|s,a)$, $R_t(s,a)$). The process defined here is a Markovian one since the following states and immediate rewards (and therefore the whole continuation of the process) depends only on the current state and operation chosen, and not on the history.

3.1.4 Policy and Decision Rules

A decision rule may need memory of the whole history to determine the most profitable course of action, or it may need only the current state. It may also be a deterministic rule (resulting in one singe operation) or a stochastic one (resulting in a distribution on a set of operations). We will define the following set of rules:
MD - a deterministic Markovian rule (having no memory):
$d_t : S \to A_s$, $d_t(s) \in A_s$
HD - a deterministic history dependent rule:
$d_t : H_t \to A_s$
Where we define the history as: $H_1 = s$, and $H_t = H_{t-1} \times A \times S$, $H = \bigcup H_t$
MR - a stochastic Markovian rule:
$d_t : S \to P(A)$,where $P(A)$ is the set of distributions on A.
HR - a stochastic history dependent rule:
$d_t : H_t \to P(A)$
We define a stationary rule to be a rule that is independent on time, i.e. $\forall t, d_t = d$, for some d.
SD - a deterministic, stationary rule.

SR - a stochastic, stationary rule.

3.1.5 Policy

A policy, at any given point in time, decides which action the agent selects. Let D_t^k be all the options at time t, then we define $P_i^k = D_1^k \times ... \times D_{n-1}^k$. We also define Π^{SD} and Π^{DT} to be a stationary, random or deterministic rule, respectively.

3.1.6 Example 1

The first example is a Markovian decision making problem with one time phase. This implies that $N = 2$, $T = \{1, 2\}$ and $r(s')$ is the value of state s' in the end.
When the system is started at state $s_0 \in S$, the goal of the agent is to choose an action $a \in A$ that maximizes the value of $v(a) = E[R_1(s_0, a_1) + r(s')]$
We will choose a deterministic policy π that selects an action as follows:
$u(a) = E[R_1(s, a)] + E[R(x_2)]$, when x_2 is a random variable describing the state reached after action a. Writing it explicitly, $u(a) = r_1(s, a) + \sum_{j \in S} P(j|s, a) * r_2(j)$
The agent will choose an action a^*, that maximizes $V(a)$, i.e.
$a^* \in \{a_1 | u(a_1) = \max_{a \in A}\{U(a)\}\}$
In the above example, there is no stochastic rule that is better then the deterministic policy presented. If there was, then such a stochastic rule would have an action distribution of $q(a)$ and the return of such a policy would be $\sum q(a)U(a)$, but since we chose a^* so that $U(a) \leq U(a^*)$ for any action a, then $\sum q(a)U(a) \leq \sum q(A)U(a^*) \leq U(a^*)$ (in simpler words, the best deterministic choice is always at least as good as any average).

3.1.7 Example 2

Lets define a state machine with two states s_1, s_2.
Formally, in the example we have:
time: $T = \{1...N\}$
states: $S = \{s_1, s_2\}$
actions: $A_{s_1} = \{a_{11}, a_{12}\}, A_{s_2} = \{a_{21}\}$
reward: $r_t(s_1, a_{11}) = 5, r_t(s_1, a_{12}) = 10, r_t(s_2, a_{21}) = -1$
final reward: $r_N(s_1) = 0, r_N(s_2) = 0$
transition function: $P_t(s_1|s_1, a_{11}) = 0.5, P_t(s_2|s_1, a_{11}) = 0.5$
$P_t(s_2|s_1, a_{12}) = 1, P_t(s_2|s_2, s_{21}) = 1$
Now we want to maximize the return. First we compare two deterministic policies:
π_1- always chooses a_{11} when in state s_1.
π_2 - always chooses a_{12} when in state s_2.

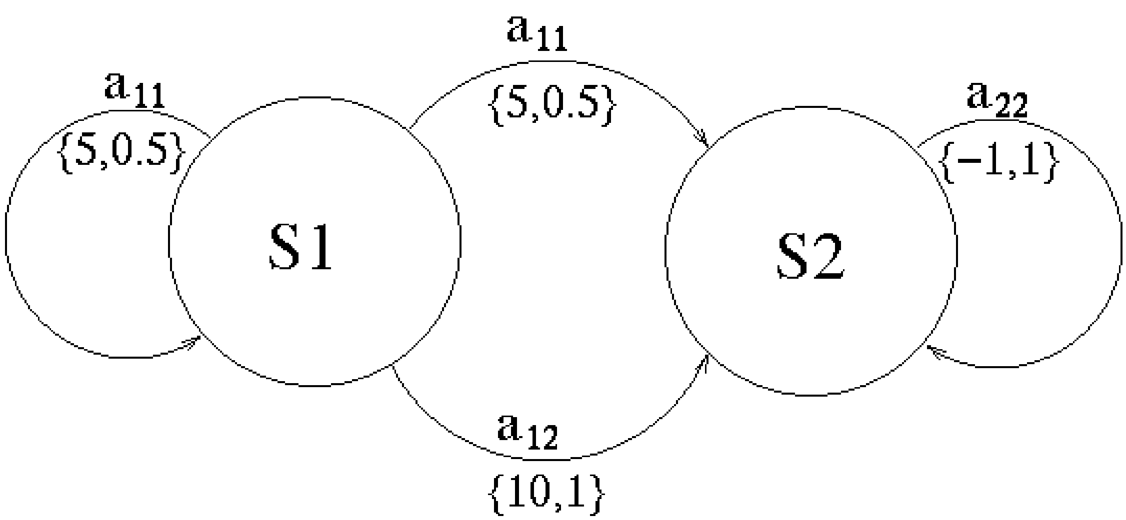

Figure 3.2: State Diagram

Recall that $V_N(\Pi)$ is the expected return of policy Π in N time steps.

$$\begin{aligned}
V_3(\pi_1) &= \frac{1}{2}(5-1+0)+\frac{1}{2}(5+5+0)=7 \\
V_3(\pi_2) &= 10-1=9 \\
V_5(\pi_1) &= \frac{1}{2}(5-3)+\frac{1}{4}(10-2)+\frac{1}{8}(20)=7.25 \\
V_5(\pi_2) &= 10-3=7
\end{aligned}$$

Notice how in shorter time frames π_2 is the preferable policy, while in longer ones, π_1 brings on the higher returns.
Lets try determining the best policy. We do this by looking first at the reward gained in the last step, for N=2:

$$\begin{aligned}
Q_2(s_1,a_{11}) &= 5 \\
Q_2(s_1,a_{12}) &= 10
\end{aligned}$$

In this case:

$$\begin{aligned}
V_2(s_1) &= 10 \\
V_2(s_2) &= -1
\end{aligned}$$

When N=2, The optimal action from s_1 is a_{12}.

For N=3, we use the optimal policy for N=2, after we do one action.

$$\begin{aligned} Q_3(s_1, a_{11}) &= 5 + \frac{1}{2}V_2(s_1) + \frac{1}{2}V_2(s_2) = 5 + \frac{1}{2} * 10 + \frac{1}{2} * (-1) = 9.5 \\ Q_3(s_1, a_{12}) &= 10 + V_2(s_2) = 10 - 1 = 9 \end{aligned}$$

Hence, when N=3 the optimal action from s_1 is a_{11}.

3.2 Finite Horizon

3.2.1 Expected Total Reward

$V_N^\pi(s) = E_s^\pi[\sum_{t=1}^{N-1} R_t(X_t, Y_t) + R_n(X_N)]$ where,
X_t- State in time t
Y_t- Action in time t
Under the assumption that $M > |r_t(s,a)|$ for each $t \leq N$ and that A and S are discrete then $V_n^\pi(s)$ is well defined for all $\pi \in \Pi^{HR}$. (Π^{HR} is the group of stochastic, history-depended decision rules). Namely, for each decision rule there is a well defined value.

3.2.2 Return Estimation of a Given Policy

Let $\pi \in \Pi^{HR}$ be a given policy. We would like to calculate it's value $V_N^\pi(s)$. An inefficient way of calculating $V_N^\pi(s)$ is creating all possible histories, finding the return value for each one and calculating the expectation of it. Then we can compute the value by taking weighted sum. An alternative calculation method is calculating return of the last step first, and recurse for the previous steps, until reaching the first step. Let us define a set of variables $U_t^\pi(h_t)$ to represent the expected return starting in time t until time N where h_t notes the history (previous steps and actions) until time t. We can write U_t^π as:
$U_t^\pi = E_{h_t}^\pi[\sum_{n=t}^{N-1} r_n(X_n, Y_n) + r_N(X_N)]$ where $h_t = (h_{t-1}, a_{t-1}, s_t)$
We can define U_t^π as a function of U_{t+1}^π as follows:
$U_t^\pi(h_t) = E^\pi h_t[r_t(X_t, Y_t) + U_{t+1}(h_t, Y_t, X_{t+1})]$
Note that X_t is known from the history and Y_t is known if π is deterministic. X_{t+1} is unknown for any policy since the environment is stochastic.
The idea: Calculate the values of U_t^π from $t = N$ to $t = 1$ using a dynamic programming algorithm.

Dynamic Programming Algorithm for $\pi \in \Pi^{HD}$

1. $t \leftarrow N$
$\forall h_N = (h_{N-1}, a_{N-1}, s_N) : U_N^\pi(h_N) \leftarrow r_N(s_N)$
2. Repeat $t \leftarrow t-1$ until t=1
$\forall h_t = (h_{t-1}, a_{t-1}, s_t) \in H_t$
$U_t^\pi(h_t) \leftarrow r_t(s_t, d_t(h_t)) + \sum_{j \in S} P_t(j|s_t, d_t(h_t)) * U_{t+1}^\pi(h_t, d_t(h_t), j)$
The next lemma states the correction of the algorithm.

Lemma 3.1 *for* $\pi \in \Pi^{HD}, U_1^\pi(s) = V_N^\pi(s)$

Proof: We will use a reversed induction to show that $U_t^\pi(h_t)$ equals the expected return of π given history h_t.
Basis: for t=N the lemma is obviously true (by the definition of V).
Induction Step: $U_t^\pi(h_t)$ can be written as:
(1) $U_t^\pi(h_t) = E_{h_t}^\pi[r_t(X_t, Y_t) + U_{t+1}(h_t, Y_t, X_{t+1)}]$
$X_t = s_t$ is known from h_t
$Y_t = d_t(h_t)$ is known from π since the policy is deterministic.
(2) $U_t^\pi(h_t) = r_t(s_t, d_t(h_t)) + E_{h_t}^\pi[U_{t+1}^\pi(h_t, d_t(h_t), x_{t+1)}]$ and at stage 2 of the algorithm we calculate exactly that expectation. Therefore, by using the induction hypothesis, it holds also for t. This implies that at the end, we have computed the expected return of π from the start state, which is by definition $V_N^\pi(s)$. □
If the policy was a stochastic one ($\pi \in \Pi^{HR}$), we would change stage 2 of the algorithm and get:
$(2) U_t^\pi(h_t) \leftarrow \sum_{a \in A} q_{h_t}(a)\{r_t(s_t, d_t(h_t)) + \sum_{j \in S} P_t(j|s_t, d_t(h_t)) * U_{t+1}^\pi(h_t, d_t(h_t), j)\}$

Computational Complexity

Let K be the number of states and L the number of actions. Then $|H_t| = K^{t+1}L^t$ and the computation time would be: $\sum_{t=0}^{N-1} K|H_t| \leq K^2 \sum_{t=0}^{N-1}(KL)^t$
The above value is for a general history dependent policy. If the policy is Markovian, then $|H_t| = K$ and we get a computation time of $K^2(N-1)$

3.2.3 The Optimality Principal

Let us define: $U_t^*(h_t) = \max_{\pi \in \Pi^{HR}}\{U_t^\pi(h_t)\}$. For the simplicity of the discussion we assume that both S and A are finite. The optimality equations (also known as Bellman equations) are:
$U_t(h_t) = \max_{a \in A}\{r_t(s_t, a) + \sum_{j \in S} P_t(j|s_t, a) U_{t+1}(h_t, a, j)\}$,
Where $U_N(h_N) = r_N(s_N)$ for $h_N = (h_{N-1}, a_{N-1}, s_N)$.
Note that replacing the max operation in the equation with the action taken by policy π, we get the equation for computing the return of policy π.

Theorem 3.2 *Let U_t be the solution for the optimality equations for $t = 1$ up to $N-1$, then:*
(1) $\forall t \leq N-1, \forall h_t \in H_t,\ u_t(h_t) = u_t^*(h_t)$
(2) $\forall s_1 \in S,\ u_1(s_1) = V_N^*(s_1)$

Proof: First, we will show by induction that $u_t(h_t) \geq u_t^*(h_t)$ for $1 \leq t \leq N$ and for every $h_t \in H_t$. Then we exhibit a specific policy π for which $V_t^\pi(h_t) = u_t(h_t)$ and that this will conclude the proof.
Basis: When $t = N$, by definition $u_N(h_N) = r_N(s_N) = u_N^\pi(h_N)$ for every policy π and therefore $u_N(h_N) = u_N^*(h_N)$.
Induction Step: Let us assume that $u_t(h_t) \geq u_t^*(h_t)$ for $n+1 \leq t \leq N$ and $h_t \in H_t$. Let π' be a policy in Π^{HR} (that performs the operation d_n' on step n). for $t = n$:
$u_n(h_n) = \max_{a\in A}\{r(s_n,a) + \sum_{j\in S} P_n(j|s_n,a)u_{n+1}(h_n,a,j)\}$
Consider an arbitrary policy π', that for history h_n uses stochastic action $q(a)$. By the induction assumption,

$$\begin{aligned}
u_n(h_n) &\geq \max_{a\in A}\{r(s_n,a) + \sum_{j\in S} P_n(j|s_n,a)u_{n+1}^*(h_n,a,j)\} \\
&\geq \max_{a\in A}\{r(s_n,a) + \sum_{j\in S} P_n(j|s_n,a)u_{n+1}^{\pi'}(h_n,a,j)\} \\
&\geq \sum_{a\in A} q(a)\{r(s_n,a) + \sum_{j\in S} P_n(j|s_n,a)u_{n+1}^{\pi'}(h_n,a,j)\} \\
&= u_n^{\pi'}(h_n)
\end{aligned}$$

Therefore, $u_n(h_n) \geq U_n^{\pi'}(h_n)$ for every π' and in particular to the optimal policy, $u_n(h_n) \geq u_n^*(h_n) = \max_{\pi'} u_n^{\pi'}(h_n)$. It is left to show that there exists a policy π' for which $V_N^{\pi'}(h_n) = u_n(h_n)$. Let us define π' in the following manner:
$\pi'(h_n) = argmax_{a\in A}\{r(s_n,a) + \sum_{j\in S} P_n(j|s_n,a)u_{n+1}(h_n,a,j)\}$
We will now show that $u_n(h_n) = u_n^{\pi'}(h_n)$ using reversed induction on n.
Basis: for $n = N$ this is obviously true.
Induction Step:

$$\begin{aligned}
u_n^{\pi'}(h_n) &= r(s_n,\pi'(h_n)) + \sum_{j\in S} P_n(j|s_n,\pi'(h_n))u_{n+1}^{\pi'}(h_n,\pi'(h_n),j) \\
&= r(s_n,\pi'(h_n)) + \sum_{j\in S} P_n(j|s_n,\pi'(h_n))u_{n+1}(h_n) \\
&= u_n(h_n)
\end{aligned}$$

The second equality is by the induction hypothesis, that $u_t^{\pi'} = u_t(h_t)$ for $t \geq n+1$. Therefore $u_n^{\pi'}(h_n) = u_n(h_n)$ and since $u_n^*(h_n) \geq u_n^{\pi'}(h_n)$ it is clear that $u_n(h_n) = u_n^*(h_n)$ as was required. □