1

EL2805, Exercise 3 (Solving MDPs).

Robert Mattila

Today's central concepts

Two phases in solving an MDP.

i, Modeling. Define all the quantities (state-space, transitions, etc.)

ii, Use algorithm to obtain optimal policy

The second step is usually rather mechanical if the first is done well.

$\Longrightarrow$ Don't rush step i, !!!

Finite-horizon, $T < \infty$:

(Stochastic) Dynamic Programming/Backward Induction:

- $u_T^*(s) = r_T(s)$ ← the terminal reward-to-go

- $$\underbrace{u_t^*(s)}_{\text{optimal expected reward-to-go at time } t} = \underbrace{\max_{a \in A_s}}_{\text{optimal action}} \Big\{ \underbrace{r_t(s,a)}_{\text{immediate reward}} + \underbrace{\sum_{s' \in S} p_t(s'|s,a)\, u_{t+1}^*(s')}_{\text{expected reward-to-go if we pick action } a} \Big\}$$

- $a_t^*(s) \in \arg\max_{a \in A_s} \{ \text{—— " ——} \}$

if there are multiple optimal actions, then it's arbitrary which one we pick

Discounted ∞-horizon:

- Optimality conditions / Bellman's equation:

$$u^*(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{s' \in S} p(s'|s,a) u^*(s') \right\}$$

- Value iteration (VI) algorithm:

$$u_{k+1}(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{s' \in S} p(s'|s,a) u_k(s') \right\}$$

will converge to $u^*(s)$ for any initial condition $u_0(s)$.

Note, this is essentially DP with a longer and longer horizon.

- Policy iteration (PI) algorithm:

  0, Guess a policy π_0

  i, (Policy evaluation)
  Compute the value of policy π_k by solving:
$$u^{\pi_k}(s) = r(s, \pi_k(s)) + \lambda \sum_{s' \in S} p(s'|s, \pi_k(s)) u^{\pi_k}(s')$$

  ii, (Policy improvement)
  Update the policy:
$$\pi_{k+1}(s) \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{s' \in S} p(s'|s,a) u^{\pi_k}(s') \right\}$$

  Stop if $\pi_k = \pi_{k+1}$.

Remark:

For the systems we consider, deterministic policies are sufficient for optimality.

That is, a policy π is a function that takes a state $s \in S$ and maps it into an action $a \in A_s$.

Remark:

These algorithms are also valid for other objective functions. There are variations of these algorithms that are more efficient.

Remark:

It is possible to solve MDPs via linear programming, which allows constraints to be enforced.

Ex. 3.2 | Machine replacement problem.

Machine breaks down with prob. θ. Cost R to replace machine, and cost c of not being able to utilize it. Find optimal policy for T time-steps.

a, Model as MDP.

b, Solve for $\theta = 0.1$, $R = 10$, $c = 5$ and $T = 3$.

Solution:

a,

State-space: $\mathcal{S} = \{\text{Functional}, \text{Broken}\} = \{F, B\}$.

Actions: $\mathcal{A} = \{\text{Continue}, \text{Replace}\} = \{C, R\}$.

Time-horizon and objective:

Finite-horizon $T < \infty$:

$$\mathbb{E}\left\{ \sum_{t=0}^{T-1} r_t(s_t, a_t) + r_T(s_T) \right\}.$$

Note: we allow for replacing a functional machine, and DP will let show that this would be nonsense.

We could have incorporated prior knowledge by: $\mathcal{A}(F) = \{C\}$, $\mathcal{A}(B) = \{C, R\}$.

Rewards:

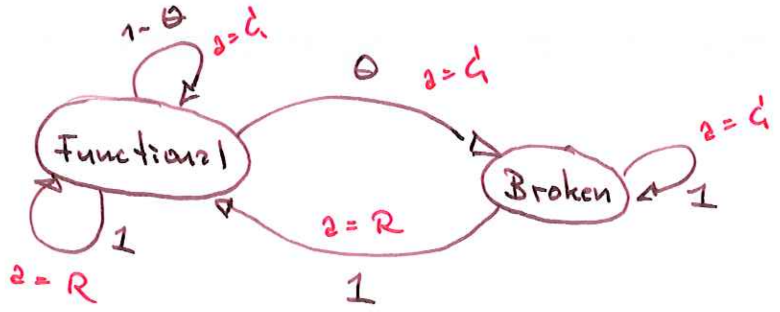
Terminal:

$$r_T(\cdot) = 0$$

Non-terminal:

- $r_t(s = \cdot, a = R) = -R$ ← choses to replace
- $r_t(s = B, a = C) = -c$ ← not able to utilize the machine
- $r_t(s = F, a = C) = 0$ ← normal operation

Transitions:



- $P_t(s' = B \mid s = B, a = C) = 1$ neglect to replace
- $P_t(s' = F \mid s = B, a = R) = 1$ replace
- $P_t(s' = F \mid s = F, a = C) = 1 - \theta$ does not break down
- $P_t(s' = B \mid s = F, a = C) = \theta$ breaks down
- $P_t(s' = F \mid s = F, a = R) = 1$ replace functional machine.

(Note:
We assume the machine cannot break down immediately when it has been replaced.)

b, To solve the MDP, recall the backward induction:

- $u_T^*(s_T) = r_T(s_T)$
- $u_t^*(s_t) = \max_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{s' \in S'} P_t(s' \mid s_t, a)\, u_{t+1}^*(s') \right\}$
- $a_t^*(s_t) \in \arg\max_{a \in A_{s_t}} \left\{ \text{———} \;\; " \;\; \text{———} \right\}$

We are given that
$\theta = 0.1$, $R = 10$, $c = 5$ and $T = 3$.

<u>Base-case:</u>

- $u_3^*(F) = r_3(F) = 0$
- $u_3^*(B) = r_3(B) = 0$

It is convenient to record the results in two tables:

u^*:

$t =$	0	1	2	3
$s = F$				0
$s = B$				0

a^*:

$t =$	0	1	2	3
$s = F$				$\cdot$
$s = B$				$\cdot$

At $t = T$, we are indifferent to what action is picked.

<u>Recursion:</u>

- $$u_2^*(F) = \max_{a \in \mathcal{A}} \Big\{ r(F,a) + \sum_{s' \in \mathcal{S}} p_t(s' \mid F, a)\, u_3^*(s') \Big\}$$

$(\mathcal{A} = \{C_1, R\})$

indicates what action the term corresponds to

$$= \max \Big\{ \overset{C_1}{r(F,C_1) + p(F \mid F, C_1)\, u_3^*(F) + p(B \mid F, C_1)\, u_3^*(B)},$$

$$\overset{R}{r(F,R) + p(F \mid F,R)\, u_3^*(F) + p(B \mid F,R)\, u_3^*(B)} \Big\}$$

$$= \max \big\{ \overset{C_1}{0 + (1-\theta)\cdot 0 + \theta \cdot 0},\ \overset{R}{-R + 1\cdot 0 + 0 \cdot 0} \big\}$$

$$= \max \big\{ \overset{C_1}{0},\ \overset{R}{-R} \big\} = 0,$$

with $a_2^*(F) = C_1$.

- $$u_2^*(B) = \max_{a \in \mathcal{A}} \left\{ r(B,a) + \sum_{s' \in \mathcal{S}} p_t(s' \mid B, a)\, u_3^*(s') \right\}$$

$$= \max \left\{ \overbrace{r(B,C_1) + p(F \mid B, C_1)\, u_3^*(F) + p(B \mid B, C_1)\, u_3^*(B)}^{C_1}, \right.$$

$$\left. \overbrace{r(B,R) + p(F \mid B,R)\, u_3^*(F) + p(B \mid B,R)\, u_3^*(B)}^{R} \right\}$$

$$= \max \left\{ \overbrace{-C + 0 \cdot 0 + 1 \cdot 0}^{C_1},\ \overbrace{-R + 1 \cdot 0 + 0 \cdot 0}^{R} \right\}$$

$$= \max \left\{ \overset{C_1}{-C},\ \overset{R}{-R} \right\} = \max \left\{ \overset{C_1}{-5},\ \overset{R}{-10} \right\} = -5,$$

with $a_2^*(B) = C_1$.

Continue the recursion one more step:

- $$u_1^*(F) = \max \left\{ \overbrace{r(F,C_1) + p(F \mid F, C_1)\, u_2^*(F) + p(B \mid F, C_1)\, u_2^*(B)}^{C_1}, \right.$$

$$\left. \overbrace{r(F,R) + p(F \mid F,R)\, u_2^*(F) + p(B \mid F,R)\, u_2^*(B)}^{R} \right\}$$

$$= \max \left\{ \overbrace{0 + (1-\theta) \cdot 0 + \theta \cdot (-5)}^{C_1},\ \overbrace{-R + 1 \cdot 0 + 0 \cdot (-5)}^{R} \right\}$$

$$= \max \left\{ \overset{C_1}{-5\theta},\ \overset{R}{-R} \right\} = \max \left\{ \overset{C_1}{-0.5},\ \overset{R}{-10} \right\} = -0.5,$$

with $a_1^*(F) = C_1$.

- $$u_1^*(B) = \max \left\{ \overbrace{r(B,C_1) + p(F \mid B, C_1)\, u_2^*(F) + p(B \mid B, C_1)\, u_2^*(B)}^{C_1}, \right.$$

$$\left. \overbrace{r(B,R) + p(F \mid B,R)\, u_2^*(F) + p(B \mid B,R)\, u_2^*(B)}^{R} \right\}$$

$$= \max \left\{ \overbrace{-C + 0 \cdot 0 + 1 \cdot (-5)}^{C_1},\ \overbrace{-R + 1 \cdot 0 + 0 \cdot (-5)}^{R} \right\}$$

$$= \max \left\{ \overset{C_1}{-C-5},\ \overset{R}{-R} \right\} = \max \left\{ \overset{C_1}{-10},\ \overset{R}{-10} \right\} = -10,$$

with $a_1^*(B) \in \{C_1, R\}$.

Note that both actions yields the same value, so we are free to chose one.

Finally, the last stage in the recursion:

- $$u_0^*(F) = \max\Big\{ \overbrace{r(F,c_1) + p(F|F,c)\,u_1^*(F) + p(B|F,c_1)\,u_1^*(B)}^{c_1},$$
$$\overbrace{r(F,R) + p(F|F,R)\,u_1^*(F) + p(B|F,R)\,u_1^*(B)}^{R} \Big\}$$
$$= \max\Big\{ \overbrace{0 + (1-\theta)(-0.5) + \theta\cdot(-10)}^{c_1},$$
$$\overbrace{-R + 1\cdot(-0.5) + 0\cdot(-10)}^{R} \Big\}$$
$$= \max\Big\{ \overbrace{-0.9\times 0.5 - 1}^{c_1}, \overbrace{-10 - 0.5}^{R} \Big\}$$
$$= \max\Big\{ \overbrace{-1.45}^{c_1}, \overbrace{-10.5}^{R} \Big\} = -1.45,$$
with $a_0^*(F) = c_1$.

- $$u_0^*(B) = \max\Big\{ \overbrace{r(B,c_1) + p(F|B,c_1)\,u_1^*(F) + p(B|B,c_1)\,u_1^*(B)}^{c_1},$$
$$\overbrace{r(B,R) + p(F|B,R)\,u_1^*(F) + p(B|B,R)\,u_1^*(B)}^{R} \Big\}$$
$$= \max\Big\{ \overbrace{-c + 0\cdot(-0.5) + 1\cdot(-10)}^{c_1}, \overbrace{-R + 1\cdot(-0.5) + 0\cdot(-10)}^{R} \Big\}$$
$$= \max\Big\{ \overbrace{-c-10}^{c_1}, \overbrace{-R-0.5}^{R} \Big\} = \max\Big\{ \overbrace{-15}^{c_1}, \overbrace{-10.5}^{R} \Big\} = -10.5,$$
with $a_0^*(B) = R$.

In summary, we have that

u^*:

$t =$	0	1	2	3
$s = F$	-1.45	-0.5	0	0
$s = B$	-10.5	-10	-5	0

a^*:

$t =$	0	1	2	3
$s = F$	G	G	G	•
$s = B$	R	G/R	G	•

either action is optimal

<u>Note:</u> The computations above are essentially matrix-vector multiplications and additions. As such, they can be computed much more succinctly then above. Try to figure out how on your own! This is also good to keep in mind when implementing on a computer.

Ex 3.3 Need to sell apartment within N days. We receive an offer w_t every evening that we need to accept or reject the next day. Every offer is a multiple of 10 000 SEK, and they are i.i.d., positive and bounded. Fixed daily interest rate $\rho > 0$ once we sell. Compute an optimal policy.

Solution:

We derived the MDP-model in the previous session.

State-space:

Let's assume we always talk in terms of multiples of 10 000 SEK, and that the maximum bid is 10 000 $\cdot$ b_{max} SEK. Then:

$$S = \{0, 1, \ldots, b_{max}\} \cup \{\text{Sold}\}$$

current bid under consideration

Actions:

$$A_b = \{\underbrace{R}_{\text{reject}}, \underbrace{A}_{\text{accept}}\}$$: the actions available when we are considering a bid b

$$A_{\text{Sold}} = \{\underbrace{C}_{\text{continue}}\}$$: ——— " ——— we have sold.

Time-horizon and objective:

Finite-horizon, N:

$$\mathbb{E}\left\{\sum_{t=0}^{N-1} r_t(s_t, a_t) + r_N(s_N)\right\}$$

Rewards:

Terminal:

- $r_N(\text{Sold}) = 0$
- $r_N(b) = b$

we are forced to sell at time N if we haven't sold

Non-terminal:

- $r_t(s=b,\ a=\mathcal{A}) = b(1+\rho)^{N-t}$

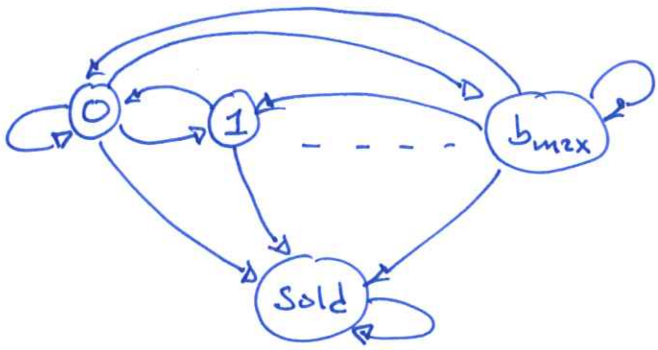
the current offer

the future interest that we will collect

- $r_t(s=b,\ a=R) = 0$
- $r_t(s=\text{Sold},\ a=C) = 0$

Note: We could take A_{Sold} to be $\{\mathcal{A}, R\}$ as we did last time. In that case, the choice of action in the policy will be arbitrary.

Transitions:



- $P_t(s'=b' \mid s=b,\ a=R) = \Pr\{w_t = b'\}$

if we reject, tomorrow's offer is drawn i.i.d.

- $P_t(s'=\text{Sold} \mid s=b,\ a=\mathcal{A}) = 1$ — we decide to sell
- $P_t(s'=\text{Sold} \mid s=\text{Sold},\ a=C) = 1$ — if we have sold

To solve the MDP, recall the backward induction:

- $u_N^*(s_N) = r_N(s_N)$
- $u_t^*(s_t) = \max\limits_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum\limits_{s' \in S}' P_t(s' | s_t, a) u_{t+1}^*(s') \right\}$
- $a_t^*(s_t) \in \arg\max\limits_{a \in A_{s_t}} \left\{ \text{———} \;\; '' \;\; \text{———} \right\}$

Base case:

- $u_N^*(\text{sold}) = r_N(\text{sold}) = 0$
- $u_N^*(b) = r_N(b) = b$

Recursion:

For $s_t = \text{sold}$:

$$u_t^*(\text{sold}) = \max_{a \in \underbrace{A_{\text{sold}}}_{=\{d\}}} \left\{ \underbrace{r_t(\text{sold}, a)}_{=0} + \sum_{s' \in S}' \underbrace{P_t(s' | \text{sold}, a)}_{= \begin{cases} 1 & \text{if } s' = \text{sold} \\ 0 & \text{o.w.} \end{cases}} u_{t+1}^*(s') \right\}$$

$$= 0 + 1 \cdot u_{t+1}^*(\text{sold})$$

$$= u_{t+1}^*(\text{sold}).$$

By induction, we conclude that

$$u_t^*(\text{sold}) = u_{t+1}^*(\text{sold}) = \ldots = u_N^*(\text{sold}) = 0. \qquad (*)$$

For $s_t = b$:

$$u_t^*(b) = \max_{a \in A_b} \left\{ r_t(b,a) + \sum_{s' \in S} p_t(s'|b,a) \underbrace{u_{t+1}^*(s')}_{\substack{\text{this is } = 0 \text{ for} \\ s' = \text{Sold according} \\ \text{to } (*)}} \right\}$$

$$= \max_{a \in \{\text{st}, R\}} \left\{ r_t(b,a) + \sum_{i=0}^{b_{max}} p_t(i|b,a)\, u_{t+1}^*(i) \right\} \qquad (**)$$

Note: This is a sum over $S \setminus \{\text{Sold}\} = \{0, 1, \dots, b_{max}\}$.

It's easiest to evaluate the two actions separately:

If $a = \text{st}$:

$$\underbrace{r_t(b, \text{st})}_{= b(1+p)^{N-t}} + \sum_{i=0}^{b_{max}} \underbrace{p_t(i|b, \text{st})}_{= 0 \text{ for all } i} u_{t+1}^*(i) = b(1+p)^{N-t}$$

If $a = R$:

$$\underbrace{r_t(b, R)}_{= 0} + \sum_{i=0}^{b_{max}} \underbrace{p_t(i|b,R)}_{= \Pr\{w_t = i\}} u_{t+1}^*(i) =$$

$$\sum_{i=0}^{b_{max}} \Pr\{w_t = i\}\, u_{t+1}^*(i) = \mathbb{E}_w \left\{ u_{t+1}^*(w) \right\}$$

Taken together in $(**)$, we obtain:

$$u_t^*(b) = \max\Big\{ \underbrace{b(1+p)^{N-t}}_{\mathcal{A}} , \underbrace{\sum_{i=0}^{b_{max}} \Pr\{w_t = i\} u_{t+1}^*(i)}_{R} \Big\}$$

$$= \max\Bigg\{ \underbrace{b}_{\mathcal{A}} , \underbrace{\frac{\sum_{i=0}^{b_{max}} \Pr\{w_t = i\} u_{t+1}^*(i)}{(1+p)^{N-t}}}_{R} \Bigg\} \times (1+p)^{N-t} \quad \overset{\text{def.}}{=} \alpha_t$$

$$\overset{\text{def.}}{=} \max\{\underbrace{b}_{\mathcal{A}}, \underbrace{\alpha_t}_{R}\} \times (1+p)^{N-t}.$$

Hence, the optimal policy is:

If we receive an offer b at time t, then we should:

$\mathcal{A}$(ccept) if $b > \alpha_t$,

R(eject) if $b \leq \alpha_t$,

if $b = \alpha_t$, the action is arbitrary.

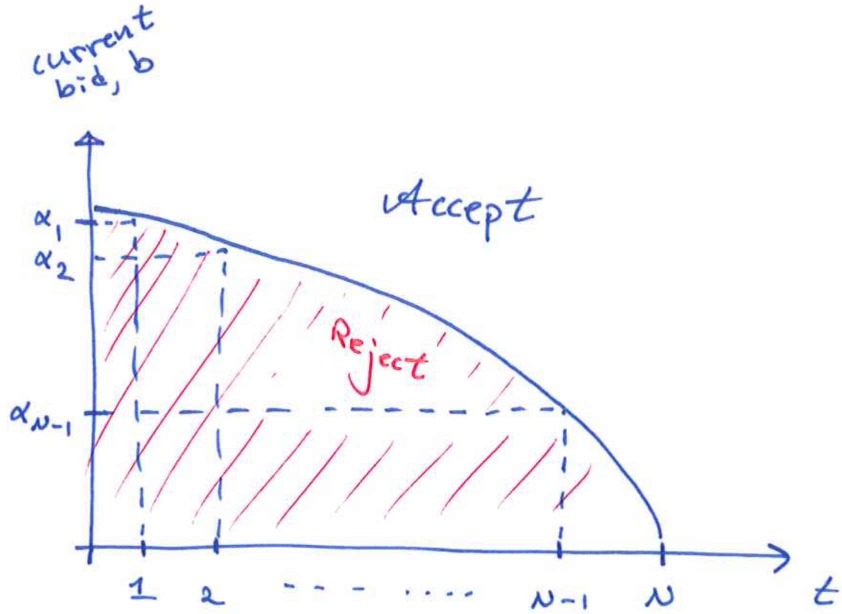
where

$$\alpha_t = \frac{\sum_{i=0}^{b_{max}} \Pr\{w_t = i\} u_{t+1}^*(i)}{(1+p)^{N-t}} = \mathbb{E}_{\omega}\{u_{t+1}^*(\omega)\} \cdot (1+p)^{t-N}$$

expected reward-to-go

discount by interest

This is a time-dependent threshold policy:



The more time we have ahead of us, the more greedy we can be with the bid we choose to accept.

Ex 3.6 At every time a job, with salary from the set $\{w_1, \dots, w_N\}$, is offered and has to be accepted or rejected. The offers are iid. An unemployment compensation c is given at every time (if unemployed). Assume future discount factor λ.

a, Show that there is a threshold $\bar{w}$ over which offers should be accepted. Characterize $\bar{w}$.

b, Assume the worker is fired from job i wp. p_i. Show that a, holds if $p_i = p$ for all i.

Solution:

State-space:

$$\mathcal{S} = \{s_1, \dots, s_N\} \cup \{s_1', \dots, s_N'\}$$

s_i: not employed, considering an offer from job i with salary w_i

s_i': employed at job i with salary w_i

Actions:

$$A_{s_i} = \{\mathcal{A}, R\}$$

accept reject

$$A_{s_i'} = \{C\}$$

continue working

Time-horizon and objective:

∞-horizon, discounted:

$$\mathbb{E}\left\{\sum_{t=0}^{\infty} \lambda^t r_t(s_t, a_t)\right\}$$

Remark:

How do we interpret λ?

One interpretation is that we are maximizing

$$\mathbb{E}\left\{ \sum_{t=0}^{T} r_t(s_t, a_t) \right\}$$

for a random T: $\Pr\{T = k\} = \lambda^{k-1}(1-\lambda)$.

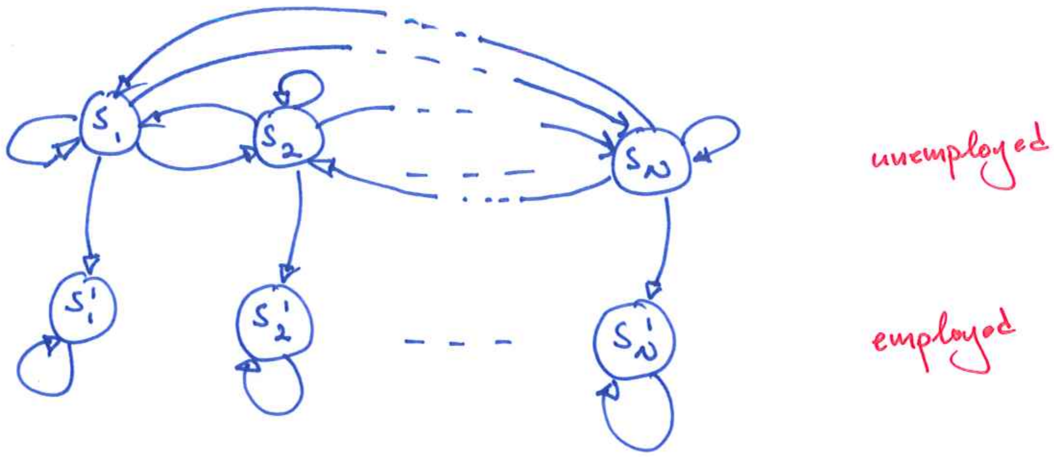
$1-\lambda$ can be interpretted as the worker's risk of dying each time unit. The worker's expected life-time is $\mathbb{E}\{T\} = \frac{1}{1-\lambda}$ time units. The worker tries to maximize the total money earned, subject to knowing that she might die.

Rewards:

- $r_t(s_i, \mathcal{A}) = w_i$ ← she accepts the offer from job i
- $r_t(s_i, R) = c$ ← she gets the unemployment compensation
- $r_t(s_i', C_i) = w_i$ ← she works at job i

Transitions:

Let q_i denote the probability that the worker receives an offer from job i.



The non-zero transitions are:

- $P_t(s_i' \mid s_i', C) = 1$
- $P_t(s_i' \mid s_i, \mathcal{A}) = 1$
- $P_t(s_j \mid s_i, R) = q_j$

Bellman equation:

$$u^*(s) = \max_{a \in A_s} \left\{ r_t(s,a) + \lambda \sum_{s' \in S} p_t(s' \mid s,a)\, u^*(s') \right\}$$

(**Remark:**

Everything is stationary, so we drop the time indices on r_t and p_t.)

Let's evaluate this for the different states:

If $s = s_i'$:

$$u^*(s_i') = \max_{a \in \underbrace{A_{s_i'}}_{=\{c_i'\}}} \left\{ r(s_i', a) + \lambda \sum_{s' \in S'} \underbrace{p(s' | s_i', a)}_{= \begin{cases} 1 & \text{if } s' = s_i' \\ 0 & \text{o.w.} \end{cases}} u^*(s') \right\}$$

$$= \underbrace{r(s_i', c_i')}_{= \omega_i} + \lambda \cdot 1 \cdot u^*(s_i')$$

$$= \omega_i + \lambda u^*(s_i')$$

$$\Longrightarrow \quad u^*(s_i') - \lambda u^*(s_i') = \omega_i$$

$$\Longrightarrow \quad u^*(s_i') = \frac{\omega_i}{1 - \lambda} \qquad (*)$$

If $s = s_i$:

$$u^*(s_i) = \max_{a \in A_{s_i}} \left\{ r(s_i, a) + \lambda \sum_{s' \in S} p(s' | s_i, a) u^*(s') \right\}$$

Let's consider the two actions separately:

If $a = \mathcal{A}$:

$$\underbrace{r(s_i, \mathcal{A})}_{= \omega_i} + \lambda \sum_{s' \in S} \underbrace{p(s' | s_i, \mathcal{A})}_{= \begin{cases} 1 & \text{if } s' = s_i' \\ 0 & \text{o.w.} \end{cases}} u^*(s') = \omega_i + \lambda u^*(s_i')$$

If $a = R$:

$$\underbrace{r(s_i, R)}_{=c} + \lambda \sum_{s' \in S} \underbrace{p(s' \mid s_i, R)}_{= \begin{cases} q_j & \text{if } s' = s_j \\ 0 & \text{o.w.} \end{cases}} u^*(s') = c + \lambda \sum_{j=0}^{N} q_j u^*(s_j)$$

Taken together, we have that

$$u^*(s_i) = \max \Big\{ \underset{A}{\omega_i + \lambda u^*(s_i')} , \underset{R}{c + \lambda \sum_{j=0}^{N} q_j u^*(s_j)} \Big\}$$

$$= \left\{ \begin{array}{c} \text{From } (*) \text{ we know that} \\ u^*(s_i') = \dfrac{\omega_i}{1-\lambda} \end{array} \right\}$$

$$= \max \Big\{ \underset{A}{\omega_i + \lambda \frac{\omega_i}{1-\lambda}} , \underset{R}{c + \lambda \sum_{j=0}^{N} q_j u^*(s_j)} \Big\}$$

$$= \max \Big\{ \underset{A}{\frac{\omega_i}{1-\lambda}} , \underset{R}{c + \lambda \sum_{j=0}^{N} q_j u^*(s_j)} \Big\}$$

$$= \max \Big\{ \underset{A}{\omega_i} , \underbrace{(1-\lambda) \Big[c + \lambda \sum_{j=0}^{N} q_j u^*(s_j) \Big]}_{\overset{\text{def.}}{=} \bar{\omega}} \Big\} \cdot \frac{1}{1-\lambda}$$

$$= \max \{ \underset{A}{\omega_i} , \underset{R}{\bar{\omega}} \} \cdot \frac{1}{1-\lambda} .$$

Since $\bar{\omega}$ is a constant (does not depend on i), this is threshold policy:

if we are offered a salary ω_i, then we should:

A(ccept) if $\omega_i > \bar{\omega}$,

R(eject) if $\omega_i \le \bar{\omega}$, ← arbitrary if $\omega_i = \bar{\omega}$.

where

$$\bar{\omega} = (1-\lambda)\left[c + \lambda \sum_{j=0}^{N} q_j u^*(s_j)\right].$$

Remark:

How does this help us — $u^*(s)$ is still unknown...? There are only $N+1$ possible threshold policies.

This limits the policy space we need to search over vastly. If N is reasonably large, we can simply compute u for each threshold policy: u^* is the best one.

Part b) : The worker gets fired w.p. p_i.
We need to remodel the problem:

State-space:

$$\mathcal{S} = \{s_1, \ldots, s_N\} \cup \{s_1', \ldots, s_N'\}$$

unemployed employed

Actions:

$$A_{s_i} = \{\mathcal{A}, R\} \qquad A_{s_i'} = \{C_i\}$$

Rewards:

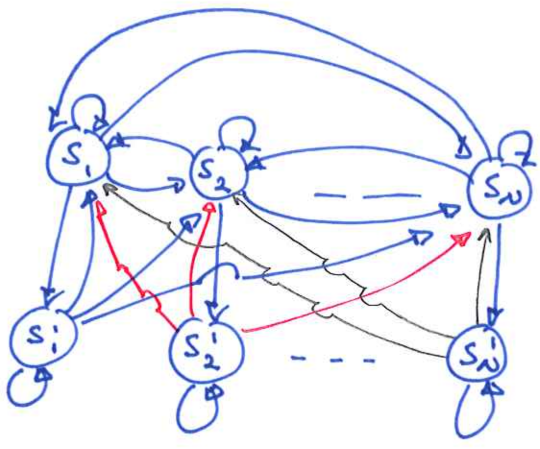
- $r(s_i, \mathcal{A}) = \omega_i$
- $r(s_i, R) = c$
- $r(s_i', C_i) = \omega_i$

Time-horizon and objective:

∞-horizon, discounted:

$$\mathbb{E}\left\{\sum_{t=0}^{\infty} \lambda^t r(s_t, a_t)\right\}$$

Transitions:



- $p(s_i' \mid s_i', C_i) = 1 - p_i$
  not fired
- $p(s_j \mid s_i', C_i) = p_i \cdot q_j$
  fired, and offered job j
- $p(s_j \mid s_i, R) = q_j$
  reject, and be offered job j
- $p(s_i' \mid s_i, \mathcal{A}) = 1 - p_i$
  accept job i, and not fired this time period (e.g., month)
- $p(s_j \mid s_i, \mathcal{A}) = p_i\, q_j$
  accept job i, but fired and then offered job j.

Bellman equation:

$$u^*(s) = \max_{a\in A_s}\left\{ r(s,a) + \sum'_{s'\in S'} p(s'|s,a)\,u^*(s')\right\}$$

Again, we consider the states separately:

If $s = s_i'$:

$$u^*(s_i') = \max_{\underbrace{a\in A_{s_i'}}_{=\{C\}}}\left\{ \underbrace{r(s_i',a)}_{=\omega_i} + \lambda \sum'_{s'\in S'} p(s'|s_i',a)\,u^*(s')\right\}$$

$$= \omega_i + \lambda\left[(1-p_i)\,u^*(s_i') + \sum_{j=0}^{N} p_i q_j\, u^*(s_j)\right]$$

If we solve for $u^*(s_i')$:

$$u^*(s_i') - \lambda(1-p_i)\,u^*(s_i') = \omega_i + \lambda p_i \sum'^{N}_{j=0} q_j u^*(s_j) \implies$$

$$u^*(s_i') = \frac{\omega_i + \lambda p_i \sum'^{N}_{j=0} q_j u^*(s_j)}{1-\lambda(1-p_i)} \qquad (*)$$

If $s = s_i$:

$$u^*(s_i) = \max_{\underbrace{a\in A_{s_i}}_{=\{\mathcal{A},R\}}}\left\{ r(s_i,a) + \lambda \sum'_{s'\in S'} p(s'|s_i,a)\,u^*(s')\right\}$$

If $a = \mathcal{A}$:

$$\underbrace{r(s_i,\mathcal{A})}_{=\omega_i} + \lambda \sum'_{s'\in S'} p(s'|s_i,\mathcal{A})\,u^*(s') =$$

$$\omega_i + \lambda\left[(1-p_i)\,u^*(s_i') + \sum_{j=0}^{N} p_i q_j\, u^*(s_j)\right]$$

<u>If $a = R$:</u>

$$\underbrace{r(s_i, R)}_{= C} + \lambda \sum_{s' \in S'} p(s' | s_i, R)\, u^*(s') = c + \lambda \sum_{j=0}^{N} q_j\, u^*(s_j)$$

Taken together, we have:

$$u^*(s_i) = \max \begin{cases} \overset{A}{\omega_i} + \lambda \left[(1-p_i)\, u^*(s_i') + \sum_{j=0}^{N} p_i\, q_j\, u^*(s_j) \right], \\ \overset{R}{c} + \lambda \sum_{j=0}^{N} q_j\, u^*(s_j) \end{cases} \qquad (**)$$

Let's analyze this. Assume (WLOG) that the salaries are sorted:

$$\omega_1 < \omega_2 < \dots < \omega_N .$$

It is also given that we should assume $p_i = p$.

Then $(*)$:

$$u^*(s_i') = \frac{\omega_i + \overbrace{\lambda p \sum_{j=0}^{N} q_j\, u^*(s_j)}^{\text{constant (in } i)}}{1 - \lambda (1-p)} \propto \omega_i$$

is an increasing function in i.

(That is,

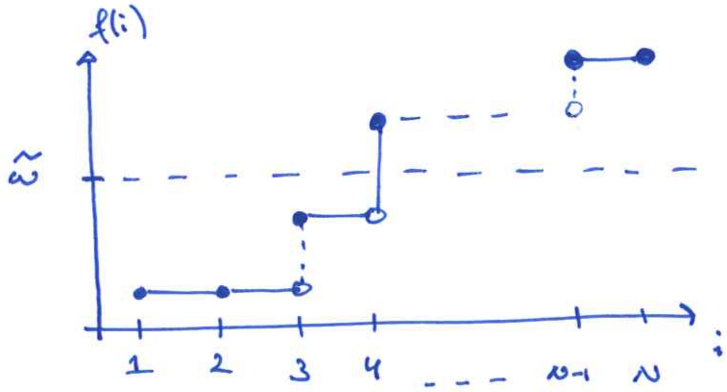
$$u^*(s_1') < u^*(s_2') < \dots < u^*(s_N').)$$

And from $(**)$:

$$u^*(s_i) = \max \begin{cases} \overset{A}{\omega_i} + \lambda \Big[\underbrace{(1-p)u^*(s_i')}_{\text{increasing in } i} + \underbrace{\sum_{j=0}^{N} p q_j u^*(s_j)}_{\text{constant (in } i)} \Big], \\ \underbrace{\overset{R}{c} + \lambda \sum_{j=0}^{N} q_j u^*(s_j)}_{\overset{\text{def}}{=} \tilde{\omega}, \text{ constant (in } i)} \end{cases}$$

$$\overset{\text{def.}}{=} \max \Big\{ \underset{A}{f(i)}, \underset{R}{\tilde{\omega}} \Big\},$$

where $f(i)$ is an increasing function in i.



The optimal policy is to A(ccept) if $f(i) > \tilde{\omega}$.
Since $f(i)$ is increasing, this means that if
$f(i) > \tilde{\omega} \implies f(i+1) > \tilde{\omega}$.

In other words, if there is a salary w_i such that $f(i) > \tilde{w}$, then it should be accepted. But then so should the salary w_{i+1} since $f(i+1) > \tilde{w}$, etc.

This implies that the solution is a threshold policy: