EL2800, 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

- $u^*_t(s) = \max_{a \in A_s} \left\{ r_t(s,a) + \sum_{s' \in S} p_t(s'|s,a) u^*_{t+1}(s') \right\}$

optimal expected reward-to-go at time t

optimal action

immediate reward

expected reward-to-go if we pick action a

- $a^*_t(s) \in \arg\max_{a \in A_s} \left\{ \text{———} \, \| \, \text{———} \right\}$

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 3.10 | 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 · 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

$$A_{Sold} = \{\underbrace{C}_{\text{continue}}\}$$: the actions available when 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:

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

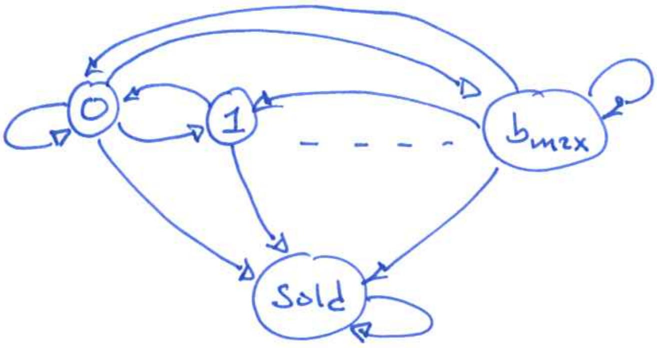
the current offer

the future interest that we will collect

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

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
- $P_t(s'=\text{Sold} \mid s=\text{Sold},\ a=\mathcal{C}) = 1$ 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 $s_t = \text{sold}$:

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

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

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

By induction, we conclude that

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

For $s_t = b$:

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

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

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

It's easiest to evaluate the two actions separately:

If $a = \mathrm{st}$:

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

If $a = R$:

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

$$\sum_{i=0}^{b_{max}} Pr\{\omega_t = i\} u_{t+1}^*(i) \qquad = \mathbb{E}_\omega \{ u_{t+1}^*(\omega) \}$$

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

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

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

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

Hence, the optimal policy is:

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

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

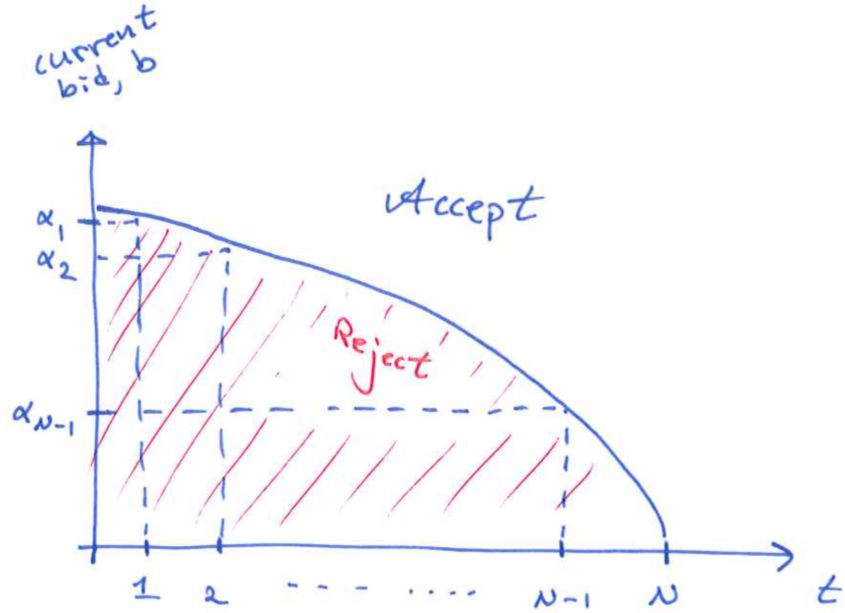
R(eject) if $b \leq \alpha_t$,

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

where

$$\alpha_t = \frac{\sum_{i=0}^{b_{max}} \Pr\{\omega_t = i\} u_{t+1}^*(i)}{(1+p)^{N-t}} = \underbrace{\mathbb{E}_{\omega}\{u_{t+1}^*(\omega)\}}_{\text{expected reward-to-go}} \cdot \underbrace{(1+p)^{t-N}}_{\text{discount by interest}}$$

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 3.2 | 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:

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

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

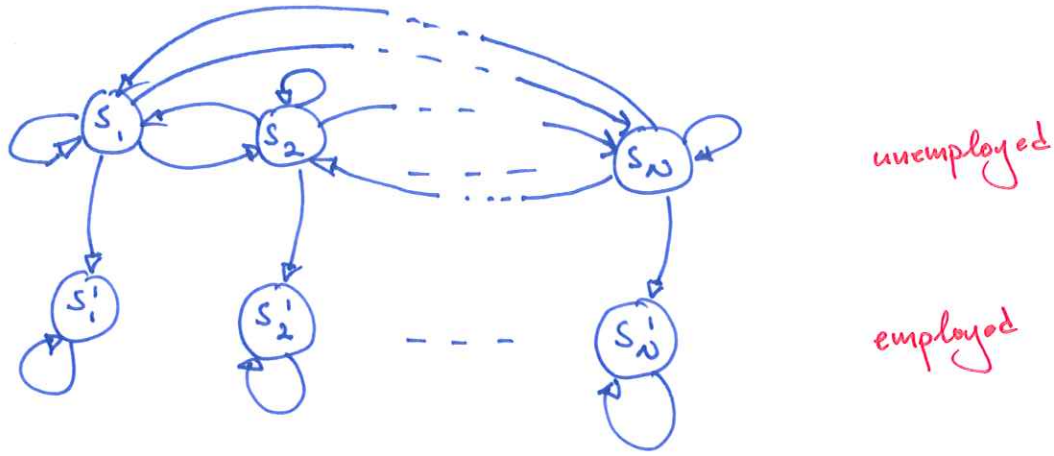
$1-\lambda$ can be interpretted as the worker's risk of dying each time unit. The worker's expected life-time is $\mathbb{E}\{T\} = \frac{1}{1-\lambda}$ time units. The worker tries to maximize the total money earned, subject to knowing that she might die.

Rewards:

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

- $p_t(s_i' \mid s_i', C) = 1$
- $p_t(s_i' \mid s_i, A) = 1$
- $p_t(s_j \mid s_i, R) = q_j$

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:

<u>If $s = s_i'$:</u>

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

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

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

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

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

<u>If $s = s_i$:</u>

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

Let's consider the two actions separately:

<u>If $a = \mathcal{A}$:</u>

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

If $a = R$:

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

Taken together, we have that

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

$$= \left\{ \text{From } (*) \text{ we know that } u^*(s_i') = \frac{\omega_i}{1-\lambda} \right\}$$

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

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

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

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

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:

$\mathcal{A}$(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:

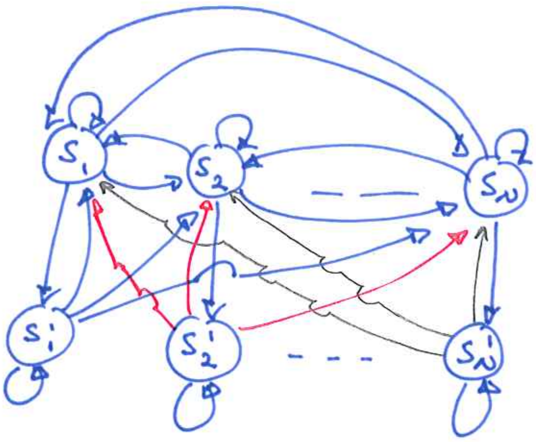
- $r(s_i, \mathcal{A}) = \omega_i$
- $r(s_i, R) = c$
- $r(s_i', C_i) = \omega_i$

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 $s = s_i'$:

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

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

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

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

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

If $s = s_i$:

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

If $a = ct$:

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

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

If $a = R$:

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

Taken together, we have:

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

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,

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

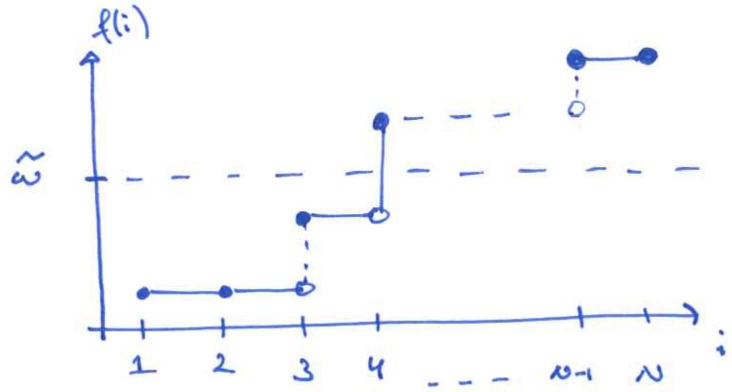
And from (* *):

$$u^*(s_i) = \max\left\{ \overset{A}{\omega_i + \lambda\Big[\underbrace{(1-p)u^*(s_i')}_{\text{increasing in } i} + \underbrace{\sum_{j=0}^{N} p\, q_j\, u^*(s_j)}_{\text{constant (in } i)}\Big]}, \right.$$

$$\left. \underbrace{\overset{R}{c + \lambda \sum_{j=0}^{N} q_j\, u^*(s_j)}}_{\overset{\text{def}}{=}\ \tilde{\omega},\ \text{constant (in } i)} \right\}$$

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

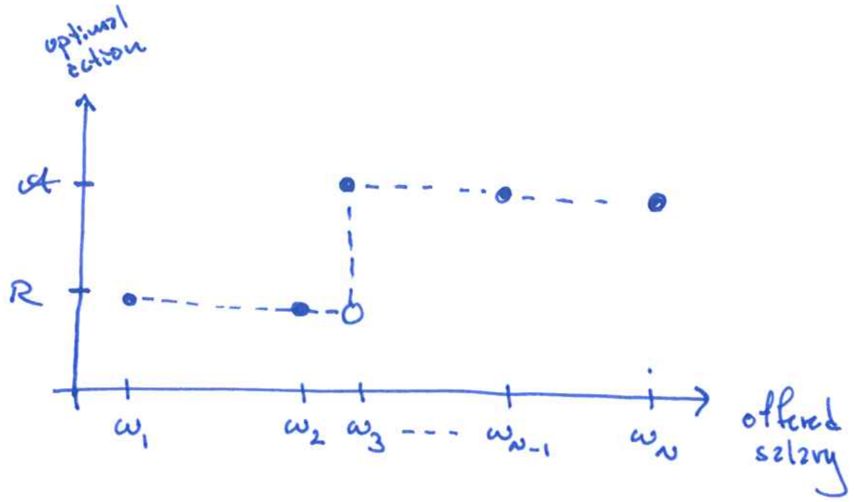
where $f(i)$ is an increasing function in i.



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 ω_i such that $f(i) > \tilde{\omega}$, then it should be accepted. But then so should the salary ω_{i+1} since $f(i+1) > \tilde{\omega}$, etc.

This implies that the solution is a threshold policy:



Ex 3.3 An order is received w.p. p at each time step. One can choose to process all orders (setup cost $K>0$), or wait (costs $c>0$ per order on hold). The maximum number of unfilled orders is n. There is a discount factor λ. Characterize an optimal processing policy.

Solution:

State-space: $S = \{0, 1, \dots, n\}$

\# of unfilled orders when starting the period

Actions:

$A_n = \{P\}$ $\qquad A_i = \{W, P\}$ $\qquad i = 1, \dots, n-1$

process all orders

wait

process all orders

Rewards:

In this case, we deal with costs:

- $r(i, P) = -K$
- $r(i, W) = -ci$ $\qquad i = 1, \dots, n-1$
- $r(n, P) = -K$

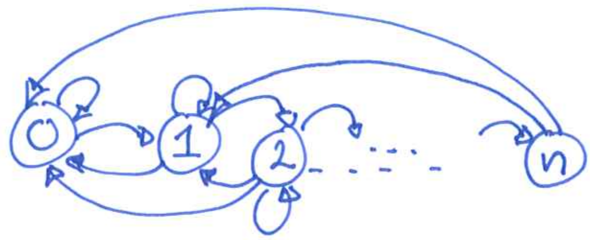
Time-horizon and objective:

∞-horizon, discounted:

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

(Remark: we can interpret $1-\lambda$ as the probability of "going out of business" (e.g., bankruptcy).)

Transitions:



- $p(i \mid i, W) = 1-p$

  wait, and no new order

- $p(i+1 \mid i, W) = p$ $\qquad\qquad i = 1, \ldots, n-1$

  wait, and a new order

- $p(0 \mid i, P) = 1-p$

  process everything, and no new order

- $p(1 \mid i, P) = p$

  process everything, and receive a new order

- $p(0 \mid n, P) = 1-p$
- $p(1 \mid n, P) = p$

Bellman equation:

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

If $s = n$:

$$u^*(n) = \max_{\underbrace{a \in A_n}_{=\{P\}}} \left\{ \underbrace{r(n,a)}_{=-K} + \lambda \sum_{s' \in S} p(s' \mid n, a) u^*(s') \right\}$$

$$= -K + \lambda \left[(1-p) u^*(0) + p u^*(1) \right] \qquad (*)$$

If $s = i < n$:

$$u^*(i) = \max_{\underbrace{a \in A_i}_{=\{P,W\}}} \left\{ r(i,a) + \lambda \sum_{s' \in S} p(s'|i,a)\, u^*(s') \right\}$$

Evaluate the actions separately:

If $a = P$:

$$\underbrace{r(i,P)}_{=-K} + \lambda \sum_{s' \in S} p(s'|i,P)\, u^*(s') =$$

$$-K + \lambda \left[(1-p) u^*(0) + p u^*(1) \right] \stackrel{\text{def.}}{=} \gamma$$

If $a = W$:

$$\underbrace{r(i,W)}_{=-ci} + \lambda \sum_{s' \in S} p(s'|i,W)\, u^*(s') =$$

$$-ci + \lambda \left[(1-p) u^*(i) + p u^*(i+1) \right]$$

So that:

$$u^*(i) = \max \Big\{ \underset{P}{\gamma},\ \underset{W}{-ci + \lambda \left[(1-p) u^*(i) + p u^*(i+1) \right]} \Big\}. \qquad (**)$$

It is reasonable to assume that the optimal policy is of threshold type. Can we prove that it is?

A threshold policy is such that

$$a^*(i) = P \implies a^*(i+1) = P.$$

This is equivalent to

$$\gamma > -ci + \lambda\left[(1-p)u^*(i) + pu^*(i+1)\right] \implies$$
$$\gamma > -c(i+1) + \lambda\left[(1-p)u^*(i+1) + pu^*(i+2)\right]$$

Note that this holds if

$$u^*(i) \geq u^*(i+1), \qquad (\square)$$

since $c > 0$.

Because then:

$$\gamma > -ci + \lambda[(1-p)u^*(i) + pu^*(i+1)] \quad \text{since } c>0$$
$$> -c(i+1) + \lambda[(1-p)\underbrace{u^*(i)}_{\geq u^*(i+1)} + p\underbrace{u^*(i+1)}_{\geq u^*(i+2)}]$$
$$\geq -c(i+1) + \lambda[(1-p)u^*(i+1) + pu^*(i+2)]$$

Can we prove that $(\square)$ holds?

We will use value iteration to prove $(\square)$.

Base case:

Let $u_0(s) = 0$ for all $s \in \mathcal{S}$.

Then $u_0(i) \geq u_0(i+1)$ holds trivially.

Induction:

Assume that $u_k(i) \geq u_k(i+1)$.

(We will show that then $u_{k+1}(i) \geq u_{k+1}(i+1)$ holds.)

If $s = i+1 < n$:

It is clear that the following holds:

$$\underbrace{-c(i+1)}_{\substack{\leq -ci \\ \text{since} \\ c>0}} + \lambda\left[(1-p)\underbrace{u_k(i+1)}_{\substack{\leq u_k(i) \\ \text{by assumption}}} + p\underbrace{u_k(i+2)}_{\substack{\leq u_k(i+1) \\ \text{by} \\ \text{assumption}}}\right] \leq -ci + \lambda\left[(1-p)u_k(i) + pu_k(i+1)\right] \quad (\Delta)$$

Let $F_k(x)$ be defined as

$$F_k(x) = \max\left\{-K + \lambda\left[(1-p)u_k(0) + pu_k(1)\right],\ x\right\}.$$

This function is increasing in x.

One iteration of value iteration is (compare the computations that led to $(**)$):

$$\begin{aligned} u_{k+1}(i+1) &= \max\Big\{-K + \lambda\left[(1-p)u_k(0) + pu_k(1)\right], \\ &\qquad -c(i+1) + \lambda\left[(1-p)u_k(i+1) + pu_k(i+2)\right]\Big\} \\ &= F_k\left(-c(i+1) + \lambda\left[(1-p)u_k(i+1) + pu_k(i+2)\right]\right) \end{aligned}$$

$$\leq F_k\left(-ci + \lambda\left[(1-p)u_k(i) + pu_k(i+1)\right)\right)$$

by (Δ) and that $F_k(x) \uparrow x$

$$= \max\left\{-K + \lambda\left[(1-p)u_k(0) + pu_k(1)\right],\right.$$
$$\left. -ci + \lambda\left[(1-p)u_k(i) + pu_k(i+1)\right]\right\}$$

$$= u_{k+1}(i).$$

by definition of one iteration of value iteration, compare (**).

If $s = n$:

Again, one iteration of value iteration (compare computation of (*)) yields:

$$u_{k+1}(n) = -K + \lambda\left[(1-p)u_k(0) + pu_k(1)\right]$$

obviously smaller than the max of itself and something else

$$\leq \max\left\{-K + \lambda\left[(1-p)u_k(0) + pu_k(1)\right],\right.$$
$$\left. -c(n-1) + \lambda\left[(1-p)u_k(n-1) + pu_k(n)\right]\right\}$$

compare (**)

$$= u_{k+1}(n-1).$$

In summary, we have now shown that

$$u_k(i) \geq u_k(i+1) \implies u_{k+1}(i) \geq u_{k+1}(i+1)$$

for all $i = 0, \ldots, n-1$.

Taking the limit:

We have shown that

- $u_0(i) \geq u_0(i+1)$
- $u_k(i) \geq u_k(i+1) \Rightarrow u_{k+1}(i) \geq u_{k+1}(i+1)$.

Since it is well known that

$$u^*(s) = \lim_{k\to\infty} u_k(s),$$

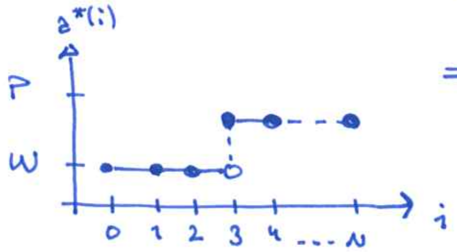
when u_k is computed using value iteration for discounted problems, we conclude that

$$u^*(i) \geq u^*(i+1),$$

See, e.g., p. 84 in Vol. 2 of Bertsekas' "Dynamic Program. and Optimal Control".

which is $(\square)$.

In summary, this proves that $a^*(i) = P \Rightarrow a^*(i+1) = P$



$\Rightarrow$ To find the optimal policy, we just need to check which of the $N+1$ possible threshold policies is the best.

(This entails performing the policy evaluation step of policy iteration $N+1$ times.)