Reinforcement Learning and Autonomous Systems (CS4122)

Lecture 17 (24/09/2024)
Lecture 18 (25/09/2024)

NOTE: The topics in this note were actually discussed in lectures 16 and 17 (23^{rd} and 24^{th} September). In lecture 18 (25^{th} September), I discussed policy iteration in class. But while preparing the notes, I have included policy iteration in lecture 16 to give the notes more structure. Accordingly, I had to shift what was actually discussed in lectures 16 and 17 to the notes of lectures 17 and 18.

Instructor: Gourav Saha

1 Introduction

In this note we will take a problem and formulate it as a MDP. After formulating the problem as an MDP, we will also write the corresponding

1. Bellman equation,
2. Equation for iterative policy evaluation,
3. Bellman optimality equation, and
4. Equation for value iteration.

for the given problem. Finally, we will also implement policy iteration for this problem in Python.

2 Machine Repair Problem

2.1 Problem Statement

Consider a machine in a factory. The health of the machine is denoted using $h \in \{1, 2, \ldots, N\}$ where $h = N$ implies the best health and $h = 1$ implies the worst health. Let $f(h)$ denote the monetary value of the products produced by the machine in one time slot if the health level is h and the machine is operating. Obviously, $f(h)$ is monotonically increasing function of h.

In every time slot, the factory owner has to decide whether to repair the machine or not:

1. *Repair:* The time to repair the machine is τ time slots. This means that if the factory owner decides to repair the machine in time slot t, then the machine is not operational from time slot t to $t + \tau - 1$. Once the repair starts, it can't be stopped until the repair is finished. Also, during repair, the monetary value of the products produced by the machine is zero. After the repair is finished, the health level of the machine becomes N.
2. *Don't repair:* Let the health level in time slot t be h. If the factory owner decides to NOT repair the machine in time slot t, then the health level of the machine in time slot $t + 1$ will be h with probability p_h and $\max(1, h - 1)$ with probability $(1 - p_h)$.

The objective of the factory owner is to maximize the β-discounted reward where the reward is the monetary value of the products produced in a time slot.

2.2 MDP Formulation

2.2.1 State and State Space

The states of the system are:

1. h: The health of the machine in the current time slot.
2. c: It is a countdown counter to capture the time after which the machine can be used after its repairing starts.

Finally, the state of the system is,

$$x = (h, c)$$

Now, we have to find the state space of x. Now:

1. $h \in \{1, 2, \ldots, N\}$.
2. $c \in \{0, 1, \ldots, \tau - 1\}$. $c = 0$ means that the machine is not undergoing repairing OR repairing just started in the current time slot. When repairing starts, the counter is set to $c = \tau - 1$ from which it counts down to $c = 0$ after which time it can start producing goods again.

Finally, $x \in \mathcal{S}$ where,

$$\mathcal{S} = \{1, 2, \ldots, N\} \times \{0, 1, \ldots, \tau - 1\} \tag{1}$$

is the state space of x.

NOTE: Compared to lectures 11 to 13 notes, we are not EXPLICITLY mentioning the time subscript related to states, actions, an rewards, i.e. instead of writting x_t, h_t, and c_t, we write x, h, and c. While it is completely correct to mention the time subscript, it may give a wrong impression of non-stationarity.

2.2.2 Action and Action Space

Let a denote the action in the current time slot. $a = 0$ means don't repair the machine while $a = 1$ means to repair the machine. The action space is,

$$\mathcal{A}(h, c) = \begin{cases} \{0, 1\} & , \ c = 0 \\ \{-1\} & ; \ c > 0 \end{cases}$$

Note that the decision to repair or not to repair can only be taken when the machine is not in reparing mode, i.e. $c = 0$. Once the repairing starts, i.e. $c > 0$, we can't decide to repair or not repair; we simply got to wait until the repairing finishes. $a = -1$ means waiting when the machine is in repair mode.

2.2.3 Reward and Reward Probability

Reward r in the current time slot is the monetary value of the goods produced by the machine in the current time slot. The equation governing reward is as follows,

$$r = \begin{cases} f(h) & ; c = 0, a = 0 \\ 0 & ; c = 0, a = 1 \\ 0 & ; c > 0 \end{cases} \tag{2}$$

In (2):

1. Case $c = 0, a = 0$: This implies that the machine is not in repair mode ($c = 0$) and also repairing didn't start in the current time slot ($a = 0$). In other words, the machine is operational and hence the monetary value of goods produced by it is $f(h)$.
2. Case $c = 0, a = 1$: This implies that the machine is not in repair mode ($c = 0$) but the repair just began in the current time slot ($a = 1$). In other words, the machine is NOT operational and hence the monetary value of goods produced by it is 0.
3. Case $c > 0$: Machine is undergoing repair and hence the monetary value of goods produced by it is 0.

One may ask that where is the reward probability. Actually, equation (2) implictly contains the reward probability. It is just that equation (2) is a deterministic function and hence it is better represented in the form given in (2). But, we can also write equation (2) as reward probability as shown below,

$$P[r \mid h, c, a] = \begin{cases} 1 & , \ c = 0, a = 0, r = f(h) \\ 0 & , \ c = 0, a = 0, r \neq f(h) \\ 1 & , \ c = 0, a = 1, r = 0 \\ 0 & , \ c = 0, a = 1, r \neq 0 \\ 1 & , \ c > 0, r = 0 \\ 0 & , \ c > 0, r \neq 0 \end{cases} \tag{3}$$

NOTE: Equations (2) and (3) are same. Don't write both in exam.

2.2.4 State Transition Probability

Let $x^{'} = \left(h^{'}, c^{'}\right)$ denote the state in the next time slot (or simply speaking the "next state"). Here, $h^{'}$ is the next health state and $c^{'}$ is the next counter state. *NOTE: By default, the symbol $'$ as an upper script to a state implies the enxt state. So, $x^{'}$ is same as x_{t+1}.*

The equation governing state transition of the health state is,

$$h^{'} = \begin{cases} \max\left(1, h-1\right) & \text{, w.p. } \left(1-p_h\right) \text{ if } c=0, a=0 \\ h & \text{, w.p. } p_h \text{ if } c=0, a=0 \\ h & \text{, } c=0, a=1 \\ h & \text{, } c>1 \\ N & \text{, } c=1 \end{cases} \tag{4}$$

In (4):

1. Case $c=0, a=0$: This implies that the machine is not in repair mode ($c=0$) and also repairing didn't start in the current time slot ($a=0$). In this case, the health of the machine will degrade with probability $1-p_h$ ot remain the same with probability p_h.
2. Case $c=0, a=1$: This implies that the machine is not in repair mode ($c=0$) but the repair just began in the current time slot ($a=1$). The health state of the machine will not degrade when the repairing starts. *Acutally, it does not matter in this problem as to what the next state $h^{'}$ is because the reward will be zero anyway.*
3. Case $c>1$: Machine is undergoing repair. Health state remains the same like the previous case.
4. Case $c=1$: In this case, the machine is one time slot away from getting repaired. Hence, the next health state is N (full health).

The equation governing state transition of the counter state is,

$$c^{'} = \begin{cases} 0 & \text{, } c=0, a=0 \\ \tau - 1 & \text{, } c=0, a=1 \\ c-1 & \text{, } c>0 \end{cases} \tag{5}$$

2.3 Designing Optimal Policy

After we formulate a problem as an MDP, the next step is to find the optimal policy. Bellman optimality equation, value iteration, Bellman equation, and iterative policy evaluation are the concepts associated with finding the optimal policy. The **generic** equations corresponding to these four concepts are as follows:

1. Bellman optimality equation.

$$\begin{aligned} V^{\star}\left(x\right) &= \max_{a \in \mathcal{A}(x)} q^{\star}\left(x,a\right) \text{ , } \forall x \in \mathcal{S} \\ q^{\star}\left(x,a\right) &= r\left(x,a\right) + \beta \sum_{x^{'} \in \mathcal{S}} P\left[x^{'} \mid x, a\right] V^{\star}\left(x^{'}\right) \text{ , } \forall x \in \mathcal{S}, \forall a \in \mathcal{A}\left(x\right) \end{aligned}$$

2. Value iteration.

$$\begin{aligned} V_{k+1}\left(x\right) &= \max_{a \in \mathcal{A}(x)} q_{k+1}\left(x,a\right) \text{ , } \forall x \in \mathcal{S} \\ q_{k+1}\left(x,a\right) &= r\left(x,a\right) + \beta \sum_{x^{'} \in \mathcal{S}} P\left[x^{'} \mid x, a\right] V_k\left(x^{'}\right) \text{ , } \forall x \in \mathcal{S}, \forall a \in \mathcal{A}\left(x\right) \end{aligned}$$

3. Bellman equation.

$$\begin{aligned} V^{\pi}\left(x\right) &= \sum_{a \in \mathcal{A}(x)} \pi\left(a \mid x\right) q^{\pi}\left(x,a\right) \text{ , } \forall x \in \mathcal{S} \\ q^{\pi}\left(x,a\right) &= r\left(x,a\right) + \beta \sum_{x^{'} \in \mathcal{S}} P\left[x^{'} \mid x, a\right] V^{\pi}\left(x^{'}\right) \text{ , } \forall x \in \mathcal{S}, \forall a \in \mathcal{A}\left(x\right) \end{aligned}$$

4. Iterative policy evaluation.

$$\begin{aligned} V_{k+1}(x) &= \sum_{a \in \mathcal{A}(x)} \pi(a \mid x)\, q_{k+1}(x,a) \ , \ \forall x \in \mathcal{S} \\ q_{k+1}(x,a) &= r(x,a) + \beta \sum_{x' \in \mathcal{S}} P\left[x' \mid x, a\right] V_{k+1}\left(x'\right) \ , \ \forall x \in \mathcal{S}, \forall a \in \mathcal{A}(x) \end{aligned}$$

As it is evident from the above equations, all of them have similar structure. So, if we write one, the others will follow.

In order to custom tailor the above equations to our particular problem we need to define the following entities for our problem:

1. states (and state space),
2. actions (and action space),
3. reward probability that decides $r(x,a)$ in the above equations, and
4. state transition probability $P\left[x' \mid x, a\right]$

But, defining these four entities is exactly what MDP formualtion is all about which we have already finished. So, **one may think that MDP formualtion and writing the generic form of the above four equations is all that is needed to custom tailor the above equations for our particular problem. Though in theory this argument is correct. In practice, it is wrong!** This is because for most problems, $r(x,a)$ and $P\left[x' \mid x, a\right]$ are **sparse**, i.e. for most (x,a) and $\left(x', x, a\right)$, $r(x,a)$ and $P\left[x' \mid x, a\right]$ are zero. If we don't use this sparsity, we will not be memory efficient while implementing value/policy iteration (say as a Python code).

2.3.1 Bellman Optimality Equation and Value Iteration

We first write the Bellman Optimality Equation for this problem which is as follows,

$$V^{\star}(h,c) = \max_{a \in \mathcal{A}(x)} q^{\star}(h,c,a) \ , \ \forall (h,c) \in \mathcal{S} \tag{6}$$

$$q^{\star}(h,0,0) = f(h) + \beta\left(p_h V^{\star}(h,0) + (1-p_h) V^{\star}(\max(1,h-1),0)\right) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{7}$$

$$q^{\star}(h,0,1) = 0 + \beta V^{\star}(h,\tau-1) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{8}$$

$$q^{\star}(h,c,-1) = 0 + \beta V^{\star}(h,c-1) \ , \ \forall h \in \{1,2,\ldots,N\}, c > 1 \tag{9}$$

$$q^{\star}(h,1,-1) = 0 + \beta V^{\star}(N,0) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{10}$$

In (2), the state space $\mathcal{S}$ is given by (1). We now explain equations (7) to (10):

1. *Equation (7):* This equation is when in the current time slot, the machine is not in repair mode ($c=0$) and also repairing didn't start in the current time slot ($a=0$). In this case, there will be an immediate reward of $f(h)$. In the next time slot:
   (a) The machine will NOT degrade with probability p_h and then incur an expected return of $V^{\star}(h,c)$ from the next time slot. This explains the term $p_h V^{\star}(h,c)$.
   (b) The machine will degrade with probability $1-p_h$ and then incur an expected return of $V^{\star}(\max(1,h-1),c)$ from the next time slot. This explains the term $(1-p_h) V^{\star}(\max(1,h-1),c)$.
2. *Equations (8) to (10):* All these equations is for the case when the machine is in repair mode in the current time slot. Hence, the immediate reward is 0. The state transition and hence the expected value of the future return can be easily understood by refering to equations (4) and (5).

Having written the Bellman optimality equation, writting value iteration is straightforward and is as follows,

$$V_{k+1}(h,c) = \max_{a \in \mathcal{A}(x)} q_{k+1}(h,c,a) \ , \ \forall (h,c) \in \mathcal{S} \tag{11}$$

$$q_{k+1}(h,0,0) = f(h) + \beta \left(p_h V_k(h,0) + (1-p_h) V_k(\max(1,h-1),0) \right) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{12}$$

$$q_{k+1}(h,0,1) = 0 + \beta V_k(h,\tau-1) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{13}$$

$$q_{k+1}(h,c,-1) = 0 + \beta V_k(h,c-1) \ , \ \forall h \in \{1,2,\ldots,N\}, c > 1 \tag{14}$$

$$q_{k+1}(h,1,-1) = 0 + \beta V_k(N,0) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{15}$$

2.3.2 Bellman Equation and Iterative Policy Evaluation

Let's say that we have a deterministic policy $\pi(x)$. So, the Bellman equation is as follows,

$$V^{\pi}(h,c) = q^{\pi}(h,c,\pi(x)) \ , \ \forall (h,c) \in \mathcal{S} \tag{16}$$

$$q^{\pi}(h,0,0) = f(h) + \beta \left(p_h V^{\pi}(h,0) + (1-p_h) V^{\pi}(\max(1,h-1),0) \right) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{17}$$

$$q^{\pi}(h,0,1) = 0 + \beta V^{\pi}(h,\tau-1) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{18}$$

$$q^{\pi}(h,c,-1) = 0 + \beta V^{\pi}(h,c-1) \ , \ \forall h \in \{1,2,\ldots,N\}, c > 1 \tag{19}$$

$$q^{\pi}(h,1,-1) = 0 + \beta V^{\pi}(N,0) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{20}$$

Similarly, iterative policy evaluation equations are as follows,

$$V_{k+1}(h,c) = q_{k+1}(h,c,\pi(x)) \ , \ \forall (h,c) \in \mathcal{S} \tag{21}$$

$$q_{k+1}(h,0,0) = f(h) + \beta \left(p_h V_k(h,0) + (1-p_h) V_k(\max(1,h-1),0) \right) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{22}$$

$$q_{k+1}(h,0,1) = 0 + \beta V_k(h,\tau-1) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{23}$$

$$q_{k+1}(h,c,-1) = 0 + \beta V_k(h,c-1) \ , \ \forall h \in \{1,2,\ldots,N\}, c > 1 \tag{24}$$

$$q_{k+1}(h,1,-1) = 0 + \beta V_k(N,0) \ , \ \forall h \in \{1,2,\ldots,N\} \tag{25}$$

2.4 Python Code for Policy Iteration

The code for policy iteration is included in this folder. I implemented policy iteration because I thought it more involved compared to value iteration. It is your job to implement value iteration for this problem.

3 Modified Machine Repair Problem

In this modified version of the machine repair problem, the machine can breakdown. After breakdown, the machine can't be repaired anymore. If the machine is in operation and its health is h, then the following are possible:

1. The machine's health does not degrade with probability p_h.

2. The machine's health degrade, i.e. goes to $\max(1, h-1)$, with probability θ_h.
3. The machine breaks down with probability $1 - p_h - \theta_h$.

The objective is to maximize the operation time of the machine.

This problem is almost same as the last problem. The only difference is that this is an episodic task while the last one is continuing task. In this note, I will briefly discuss this problem. It is your job to fill in the blanks!

The states remains the same. But the state space becomes,

$$\mathcal{S} = \{1, 2, \ldots, N\} \times \{0, 1, \ldots, \tau - 1\} \bigcup \{end\}$$

where the *end* state is when the machine breaks down. The action and action space does not change[1].

The Bellman optimality equation can be divided into two parts:

1. When the episode ended (break down already occured). In this case,

$$V^\star(end) = 0 + \beta V^\star(end) \tag{26}$$

   The above equation implies $V^\star(end) = 0$. What this essentially means is that once the machine breaks down, the net return is 0.

2. When the episode is not over (machine did not break down). In this case,

$$V^\star(h, c) = \max_{a \in \mathcal{A}(x)} q^\star(h, c, a) \ , \ \forall (h, c) \in \mathcal{S} \tag{27}$$

$$q^\star(h, 0, 0) = f(h) + \beta\left(p_h V^\star(h, 0) + \theta_h V^\star(\max(1, h-1), 0) + (1 - p_h - \theta_h) V^\star(end)\right) \ , \ \forall h \in \{1, 2, \ldots, N\} \tag{28}$$

$$q^\star(h, 0, 1) = 0 + \beta V^\star(h, \tau - 1) \ , \ \forall h \in \{1, 2, \ldots, N\} \tag{29}$$

$$q^\star(h, c, -1) = 0 + \beta V^\star(h, c - 1) \ , \ \forall h \in \{1, 2, \ldots, N\}, c > 1 \tag{30}$$

$$q^\star(h, 1, -1) = 0 + \beta V^\star(N, 0) \ , \ \forall h \in \{1, 2, \ldots, N\} \tag{31}$$

To simplify equation (28), we can simply substitute $V^\star(end) = 0$ from (26).

4 Homework

Write the following for the fishing in gridworld (lecture 10), automatic scheduler for WiFi and mobile data (lectures 11 to 13), and investing in stock market (lectures 11 to 13):

1. Bellman equation,
2. Equation for iterative policy evaluation,
3. Bellman optimality equation, and
4. Equation for value iteration.

[1] If you want to be rigourous, you can an action "do nothing" for the *end* state