mrp_3

천정민

서울과학기술대학교
SEOULTECH SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

I. Motivation

Recap

- A Markov chain is a stochastic process with the specification of
  - a state space S
  - a transition probability matrix $\mathbf{P}$

- A Markov reward process is a Markov chain with the specification of
  - a reward r_t with the reward function $R(s)$
  - a time horizon H, which is the duration we are interested in cumulative sum of rewards.

- If H is finite, then we call finite-horizon MRP.
- IF H is infinite, then we call infinite-horizon MRP.
  - Sometimes, the stochastic process is non-terminating infinite horizon MRP.
  - Sometimes, the stochastic process is terminating infinite horizon MRP. In this case, we may treat them as non-terminating, but the chain is absorbed into a absorbing state whose reward is zero.

Formulating an infinite horizon MRP

- In the previous lecture, we dealt with the following question.

'Given I drink coke today, what is likely my consumption for upcoming `10 days`? (Pepsi is \$1 and Coke is \$1.5)'

- Infinite horizon problem is such as following.

'I am to live eternally. Given I drink coke today, what is likely my consumption for `my upcoming forever life`? (Pepsi is \$1 and Coke is \$1.5)'

'(In this case, how to model her death?)'

- It may seem unrealistic on this soda problem to have an infinite time horizon. But infinite horizon model is indeed more common for MRP due to following reasons.

1. Time horizon may be finite, but uncertain how long.
2. The horizon may be belived to be a long time.
3. In accounting principle, all businesses are assumed to be perpetual.
4. Oftentimes, each time step is very small such as minute ,or even millisecond, making the number of total time step as a very large number.
5. Really long finite time horizon can be approximated by infinite time.

Return for infinite horizon

- Return for finite horizon in the previous lecture was

$$G_t = \sum_{i=t}^{H-1} r_i = r_t + r_{t+1} + \cdots + r_{H-1}$$

- If extended for infinite horizon, it becomes

$$G_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} + \cdots$$

Convergence of G_t

$$G_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} + \cdots$$

- Is G_t a convergent series, thus measurable??
  - Even if r_i is a small number, it may diverge unless r_t decays drastically over time.
  - What does it mean drastically?
    - cf) $\sum 1/n = \infty$ and $\sum 1/n^2 < \infty$

Discount factor

Introducing discount factor

- A mathematically convenient way to guarantee is to introduce discount factor, $\gamma < 1$
- Using a discount factor, the return, G_t, is newly defined as

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$$

- As long as r_t is bounded, i.e. $|r_t| < M$ for some $M > 0$ and for all t, G_t is convergent.
- G_t can be written as

$$G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$$

Note that this generalizes the previous notation with $\gamma = 1$.

Is discount factor practical?

- Many real problems indeed should be modelled with discount factor.
- Humans behave in much the same way, putting more importance in the near future.
- Interest rate is generally positive, making today's money worth more than tomorrow's money.
- Future is risky to some degree, making future's reward less valuable than today's reward.
- ~~If you die today, there is no tomorrow.~~

State-value function

- Like before, the state-value function $V_t(s)$ for a MRP and a state s is defined as the expected return starting from state s at time t, namely,

$$V_t(s) = \mathbb{E}[G_t | S_t = s]$$

- For infinite horizon problem, are the following two quantity different?
  1. $V_t(s) = \mathbb{E}[G_t | S_t = s]$
  2. $V_0(s) = \mathbb{E}[G_0 | S_0 = s]$

- It is not! This is because the lengths of remaing time where returns are summed are equally infinite.
- This makes our life easier, and allowing us to drop the time subscript for the state-value function when necessary.
- Namely, $V_t(s) = V_0(s) = V(s)$.

```
"Dream as if you'll live forever. Live as if you'll die today. - James Dean"
```

Summary

$$
\begin{aligned}
G_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots && (1)\\
G_t &= \sum_{i=t}^{\infty} \gamma^{i-t} r_i && (2)\\
V(s) &= \mathbb{E}[G_t | S_t = s] && (3)
\end{aligned}
$$

II. Analytic solution

Development

- For a finite horizon MRP, the goal was to find $V_t(s)$ for all states s for $0 \leq t \leq H$.
- Since $V_0(s) = V_t(s) = V(s)$, the goal is only to find $V(s)$ for all states s.

$$
\begin{aligned}
V(s) &= V_t(s) = \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | S_t = s] \\
&= \mathbb{E}[r_t | S_t = s] + \gamma \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | S_t = s] \\
&= R(s) + \gamma \mathbb{E}[G_{t+1} | S_t = s] \\
&= R(s) + \gamma \sum_{\forall s'} \mathbb{P}[S_{t+1} = s' | S_t = s] \mathbb{E}[G_{t+1} | S_t = s, S_{t+1} = s'] \\
&= R(s) + \gamma \sum_{\forall s'} \mathbf{P}_{ss'} \mathbb{E}[G_{t+1} | S_{t+1} = s'] \ (\because \text{Markov property}) \\
&= R(s) + \gamma \sum_{\forall s'} \mathbf{P}_{ss'} V_{t+1}(s') \\
&= R(s) + \gamma \sum_{\forall s'} \mathbf{P}_{ss'} V(s') \qquad (4)
\end{aligned}
$$

- The Eq in previous lecture note was

$$V_t(s) = R(s) + \sum_{\forall s'} \mathbf{P}_{ss'} V_{t+1}(s'), \ \forall s$$

- The Eq (4) in the previous page was

$$V(s) = R(s) + \gamma \sum_{\forall s'} \mathbf{P}_{ss'} V(s'), \ \forall s$$

- Difference
  - The former had $\gamma = 1$.
  - The latter dropped the time subscripts.
- Similarity
  - Both are understood as

$$\text{(Expected return at time } t) = (\text{reward at time } t) + (\text{Expected return at time } t+1)$$

- The above equation is called Bellman's equation, named after Richard R. Bellman, who introduced dynamic programming in 1953.

Analytic formula

$$V(s) = R(s) + \gamma \sum_{\forall s'} \mathbf{P}_{ss'} V(s'), \ \forall s$$

- Once again, the strategy is
  - Column vector v for $V(s)$
  - Column vector R for $R(s)$
  - $\gamma \mathbf{P} v$ for $\gamma \sum_{\forall s'} \mathbf{P}_{ss'} V(s')$, where $\mathbf{P}$ is a transition matrix
- It follows $v = R + \gamma \mathbf{P} v$
- This can be solved as:

$$\begin{aligned} & v = R + \gamma \mathbf{P} v \\ \Rightarrow \quad & Iv = R + \gamma \mathbf{P} v \\ \Rightarrow \quad & Iv - \gamma \mathbf{P} v = R \\ \Rightarrow \quad & (I - \gamma \mathbf{P}) v = R \\ \Rightarrow \quad & v = (I - \gamma \mathbf{P})^{-1} R \end{aligned}$$

Example

'I am to live eternally. Given I drink coke today, what is likely my consumption for `my upcoming forever life`? (Pepsi is \$1 and Coke is \$1.5)'

- We need information regarding the discount rate. Let's assume $\gamma = 0.95$.
- (equivalent to having daily interest rate of 5%)
- We have

$$\begin{aligned} v &= R + \gamma \mathbf{P} v \\ \begin{pmatrix} v(c) \\ v(p) \end{pmatrix} &= \begin{pmatrix} R(c) \\ R(p) \end{pmatrix} + \gamma \begin{pmatrix} \mathbf{P}_{cc} & \mathbf{P}_{cp} \\ \mathbf{P}_{pc} & \mathbf{P}_{pp} \end{pmatrix} \begin{pmatrix} v(c) \\ v(p) \end{pmatrix} \\ \begin{pmatrix} v(c) \\ v(p) \end{pmatrix} &= \begin{pmatrix} 1.5 \\ 1.0 \end{pmatrix} + 0.95 \begin{pmatrix} 0.7 & 0.3 \\ 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} v(c) \\ v(p) \end{pmatrix} \end{aligned} \tag{5}$$

```python
import numpy as np

# Transition matrix P
P = np.array([[0.7, 0.3],
              [0.5, 0.5]])

# Reward vector R
R = np.array([[1.5],
              [1.0]])

# Discount factor
gamma = 0.95

# Compute v = (I - gamma*P)^(-1) * R
I = np.eye(2)
v = np.linalg.inv(I - gamma * P) @ R

print(v)
```

```
## [[26.48148148]
##  [25.86419753]]
```

III. Iterative solution - by fixed point theorem

Recap

- The previous approach was based on the following two formula.

$$v = R + \gamma \mathbf{P} v \tag{6}$$
$$v = (I - \gamma \mathbf{P})^{-1} R \tag{7}$$

- The Eq. (6) is a Bellman's equation.
- The Eq. (7) is used to find a analytic solution.
- Using the Eq (7), there are two concerns that you should have. (This is the suggested solution to Exercise **??**)
  1. The matrix $I - \gamma \mathbf{P}$ may not be invertible.
  2. Even if invertible, it may be prohibitive if for a big matrix.
- We are free from the first concern. The matrix $I - \gamma \mathbf{P}$ can be proved to be invertible always as long as $\mathbf{P}$ is stochastic.
- We are not free from the second concern. So, this section introduces an alternative, numerical, and iterative approach.

Iterative algorithm

- Using the fixed-point theorem along with Eq. (6), we apply the following iterative algorithm to find v. (Warning: The subscript i is not state index, not time index, but the iteration index)

$$v_{i+1} \leftarrow R + \gamma \mathbf{P} v_i$$

```
Let epsilon <- 10^{-8} # or some small number
Let v_0 <- zero vector
i <- 1
While ||v_i-v_{i-1}|| > epsilon # may use any norm
  v_{i+1} <- R + \gamma*P*v_{i}
  i <- i+1
Return v_{i+1}
```

Fixed point theorem

Definition 1 (Fixed point)

For a function $f(\cdot)$, x^* is called a fixed point if $f(x^*) = x^*$ holds.

Remark 1

- For example, $x^* = 2$ is a fixed point for $f(x) = x^2 - 3x + 4$.
- Not all functions have fixed points. For example, $f(x) = x + 1$.
- In graphical terms, a fixed point x means the point $(x, f(x))$ is on the line $y = x$.
- In other words the graph of f has a point in common with that line.

- The above mentioned fixed point method has a few common properties.
  1. It is characterized as a iterative process (such as $x_0 \to x_1 \to x_2 \to \cdots$).
  2. In each iteration, the current candidate for the solution gets closer to the true value.
  3. It converges. That is, it is theoretically reach the exact value up to tolerance.

Implementation

```python
import pandas as pd

# Transition matrix P (column-major to match R)
P = np.array([0.7, 0.5, 0.3, 0.5]).reshape((2, 2),
order="F")
# Reward vector R
R = np.array([[1.5],
              [1.0]])

gamma = 0.95
epsilon = 1e-8

# Initial value vector v_old = [0,0]^T
v_old = np.zeros((2, 1))

# Store results in a list
results = [v_old.flatten()]

while True:
    v_new = R + gamma * (P @ v_old)   # Bellman update
    results.append(v_new.flatten())
    if np.max(np.abs(v_new - v_old)) < epsilon:  # convergence check
        break
    v_old = v_new
```

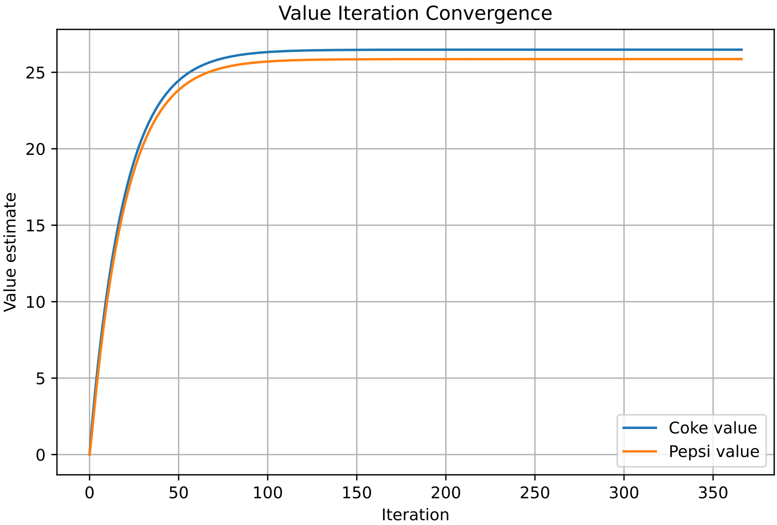
```python
# Convert to DataFrame for nicer output
df = pd.DataFrame(results, columns=["coke", "pepsi"])

print(df.head())   # show first few iterations
```

```
##        coke     pepsi
## 0  0.000000  0.000000
## 1  1.500000  1.000000
## 2  2.782500  2.187500
## 3  3.973800  3.360750
## 4  5.100391  4.483911
```

```python
print(df.tail())   # show last few iterations
```

```
##           coke      pepsi
## 362  26.481481  25.864197
## 363  26.481481  25.864197
## 364  26.481481  25.864197
## 365  26.481481  25.864197
## 366  26.481481  25.864197
```

visualize


Value Iteration Convergence
Value estimate
Iteration
0
5
10
15
20
25
0
50
100
150
200
250
300
350
Coke value
Pepsi value

```
"withus KNN"
```