Lecture D2. Markov Reward Process 2

Sim, Min Kyu, Ph.D., mksim@seoultech.ac.kr

SEOULTECH 서울과학기술대학교 데이터사이언스학과

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*.

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)

- 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 the horizon may be believed to be a long time and/or H is not certain.
2. In accounting principle, all businesses are assumed to be perpetual.
3. Really long finite time horizon can be approximated by infinite time.
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.

Return for infinite horizon

- Return for finite horizon in the previous lecture was

$$G_t = \sum_{i=t}^{H-1} r_i$$

- If extended for infinite horizon, it becomes

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

  - Even if r_i is a small number, the quantity is diverging as long as r_i does not decay drastically.
  - cf) $\sum 1/n = \infty$ and $\sum 1/n^2 < \infty$
  - In case r_i decaying drastically, the convergence is only guaranteed if the chain will eventually be absorbed to one of absorbing states whose rewards are zero.

Discount factor

- A mathematically convenient way to guarantee is to introduce *discount factor*, $\gamma < 1$
- Using a discount factor, the return becomes

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

- Or, it can be written as

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

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

- Other than for computational reason for return convergence, 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 MDP 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. $V_0(s) = \mathbb{E}[G_0 | S_t = 0]$
  2. $V_t(s) = \mathbb{E}[G_t | S_t = s]$

- It is not! This makes our life easier, and allowing us to drop the time subscript for the state-value function when necessary. Namely, $V_0(s) = V_t(s) = V(s)$.

Summary

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

II. Method 3 - 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.

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

- The section for Iterative solution in the previous lecture had the last equation of

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

- The Eq (4) was

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

- These are very similar except that the previous one had $\gamma = 1$.
- It is constructed as

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

- These are called *Bellman's equation*, named after Richard R. Bellman `(wiki link)` who introduced dynamic programming in 1953.

Analytic formula

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

- 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 $\gamma = 0.9$.
- We have

$$\begin{aligned} v &= R + \gamma \mathbf{P} v \\ \begin{pmatrix} v(c) \\ v(p) \end{pmatrix} &= \begin{pmatrix} R(c) \\ R(p) \end{pmatrix} + \gamma \begin{pmatrix} \mathbf{P}_{cc} & \mathbf{P}_{cp} \\ \mathbf{P}_{pc} & \mathbf{P}_{pp} \end{pmatrix} \begin{pmatrix} v(c) \\ v(p) \end{pmatrix} \\ \begin{pmatrix} v(c) \\ v(p) \end{pmatrix} &= \begin{pmatrix} 1.5 \\ 1.0 \end{pmatrix} + 0.9 \begin{pmatrix} 0.7 & 0.3 \\ 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} v(c) \\ v(p) \end{pmatrix} \end{aligned} \tag{5}$$

```r
P <- array(c(0.7,0.5,0.3,0.5), dim=c(2,2))
R <- array(c(1.5,1.0), dim=c(2,1))
gamma = .9
v <- solve(diag(2)-gamma*P)%*%R # v=(I-gamma P)^{-1}R
v
```

```
##          [,1]
## [1,] 13.35366
## [2,] 12.74390
```

Exercise 1

What is the relationship between the above vector v and stationary distribution?

Exercise 2

What are your concerns for this approach?

III. Method 4 - 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 2)
  1. The matrix $I - \gamma \mathbf{P}$ may not be invertible.
  2. Even if it's invertible, it may be prohibitive if the size of the matrix is big.
- We are free from the first concern. The matrix $I - \gamma \mathbf{P}$ can be proved to be invertible always.
- 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.

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

```
Let epsilon <- 10^{-8} # or some small number
Let v_0 <- zero vector
Let v_1 <- R + \gamma*P*v_0
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}
```

Math Review - Norm

Definition 1

For a length-n vector x, the norm of vector $||x||_p$ is defined as follows.

- 1-norm: $||x||_1 = \sum_{i=1}^{n} |x_i|$ *(sum of absolute value)*
- 2-norm: $||x||_2 = (\sum_{i=1}^{n} x_i^2)^{1/2}$ *(Euclidean distance, distance from the origin)*
- ∞-norm: $||x||_\infty = max_{1 \leq i \leq n} |x_i|$ *(farthest axis)*

- Throughout this course, we will use ∞-norm to guarantee that value functions (or any other quantities) are well approximated for every state.

Implementation

- The psedo code

```
Let epsilon <- 10^{-8} # or some small number
Let v_0 <- zero vector
Let v_1 <- R + \gamma*P*v_0
i <- 1
While ||v_i-v_{i-1}|| > epsilon # may use any n
  v_{i+1} <- R + \gamma*P*v_{i}
  i <- i+1
Return v_{i+1}
```

- The R-code
  - *(I wish there was a do-while loop in R)*

```
R <- array(c(1.5,1.0), dim=c(2,1))
P <- array(c(0.7,0.5,0.3,0.5), dim=c(2,2))
gamma <- 0.9
epsilon <- 10^(-8)
v_old <- array(rep(0,2), dim=c(2,1))
v_new <- R + gamma*P%*%v_old
while (max(abs(v_new-v_old)) > epsilon) { # inf-nor
  v_old <- v_new
  v_new <- R + gamma*P%*%v_old
}
print(v_new)
```

```
##          [,1]
## [1,] 13.35366
## [2,] 12.74390
```

- The full iteration process

```r
R <- array(c(1.5,1.0), dim=c(2,1))
P <- array(c(0.7,0.5,0.3,0.5), dim=c(2,2))
gamma <- 0.9
epsilon <- 10^(-8)
v_old <- array(rep(0,2), dim=c(2,1))
v_new <- R + gamma*P%*%v_old
results <- t(v_old)                  # to save
results <- rbind(results, t(v_new)) # to save
while (max(abs(v_new-v_old)) > epsilon) {
  v_old <- v_new
  v_new <- R + gamma*P%*%v_old
  results <- rbind(results, t(v_new)) # to save
}
```

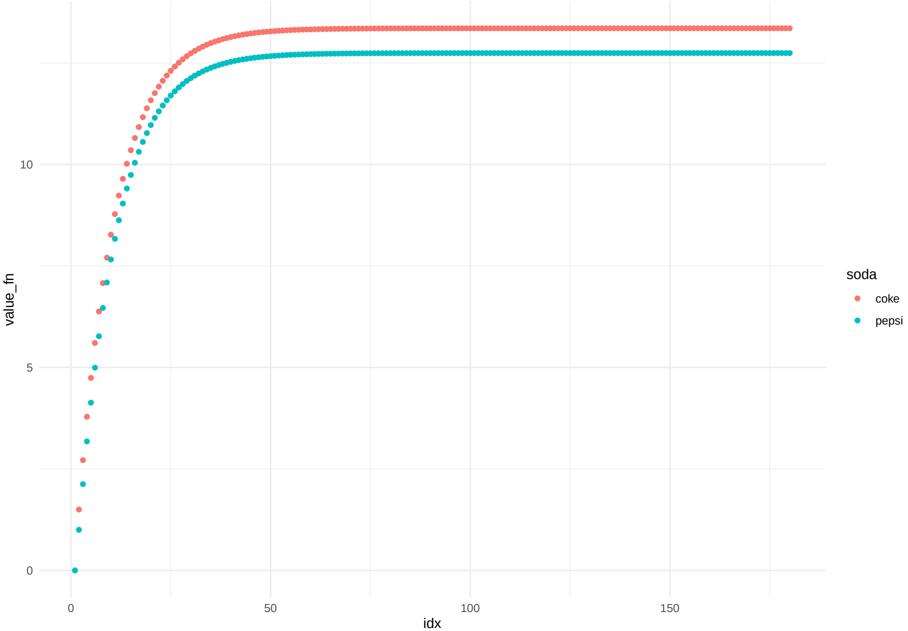
```r
results <- data.frame(results)
colnames(results) <- c("coke", "pepsi")
```

```r
head(results)
```

```
##       coke    pepsi
## 1 0.000000 0.000000
## 2 1.500000 1.000000
## 3 2.715000 2.125000
## 4 3.784200 3.178000
## 5 4.742106 4.132990
## 6 5.603434 4.993793
```

```r
tail(results)
```

```
##         coke   pepsi
## 175 13.35366 12.7439
## 176 13.35366 12.7439
## 177 13.35366 12.7439
## 178 13.35366 12.7439
## 179 13.35366 12.7439
## 180 13.35366 12.7439
```


value_fn
10
5
0
0
50
100
150
idx
soda
coke
pepsi

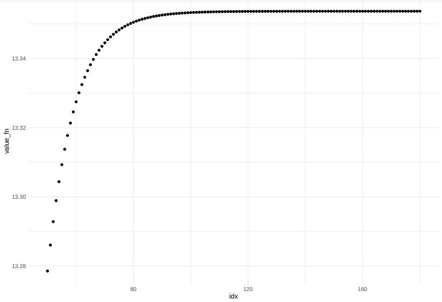
- The previous plot was generated by the following code.

```
library(tidyverse)
results$idx <- as.numeric(row.names(results))
results <- results %>%
  gather("coke", "pepsi", key="soda", value="value_fn")
ggplot(results, aes(x=idx, y=value_fn, group = soda, color = soda)) +
  geom_point() +
  theme_minimal()
```

- Note that there are quite convergence going on after many steps.

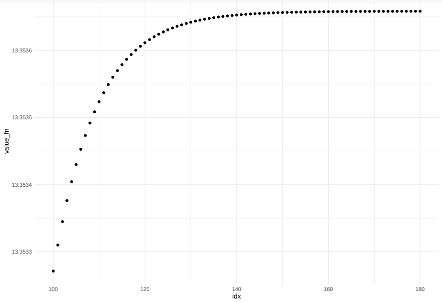
- After 50 steps (coke only)

```r
results %>% filter(idx >= 50, soda == "coke") %>%
  ggplot(aes(x=idx, y=value_fn)) +
  geom_point() +
  theme_minimal()
```



- After 100 steps (coke only)

```r
results %>% filter(idx >= 100, soda == "coke") %>%
  ggplot(aes(x=idx, y=value_fn)) +
  geom_point() +
  theme_minimal()
```



```
"Success isn't permarnent, and failure isn't fatal. - Mike Ditka"
```