

## Lecture D2. Markov Reward Process 2

Sim, Min Kyu, Ph.D., [mksim@seoultech.ac.kr](mailto:mksim@seoultech.ac.kr)



서울과학기술대학교 데이터사이언스학과

- 1 I. Motivation
- 2 II. Method 3 - Analytic solution
- 3 III. Method 4 - Iterative solution - by fixed point theorem

# I. Motivation

# Recap

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

absorbing states  
✓ terminating  
✓ non-terminating

## 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)*

$$P \begin{array}{c} \begin{array}{c} C \\ P \\ \text{Etern.} \end{array} \\ \begin{bmatrix} .3 & .7-.8 & 1 \\ .5 & .5-.8 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

- 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.

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

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

$$\underline{G_t} = r_t + r_{t+1} + \dots$$

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

↓  
state  $\frac{1}{2}$  water


## Return for infinite horizon

- Return for finite horizon in the previous lecture was

$$G_t = \sum_{i=t}^{H-1} r_i \quad \left\{ = r_t + \dots + r_{H-1} \right.$$

- If extended for infinite horizon, it becomes

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

- 
- 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

for all  $r_i, |r_i| \leq M$   
bounded

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

- Or, it can be written as

$$< M + \gamma M + \gamma^2 M + \dots = \frac{M}{1-\gamma} < 10$$

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

: discounted sum of rewards

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

$G_t$



- 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?

time 0 or 2012th reward discounted sum

$$\begin{aligned} \textcircled{1} & V_0(s) = \mathbb{E}[G_0 | S_0 = 0] \\ \textcircled{2} & V_t(s) = \mathbb{E}[G_t | S_t = s] \end{aligned}$$

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

$$\begin{aligned} V(s) &= \mathbb{E}[G_t | S_t = s] \\ &\quad \uparrow \quad \uparrow \text{rew.} \\ &= \mathbb{E}[G_0 | S_0 = s] \end{aligned}$$

# Summary

$$\left( \begin{array}{lcl} G_t & = & r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots & (1) \\ G_t & = & \sum_{i=t}^{\infty} \gamma^{i-t} r_i & (2) \\ V_x(s) & = & \mathbb{E}[G_t | S_t = s] & (3) \end{array} \right.$$

## II. Method 3 - Analytic solution

## Development

 $V_0(s)$ 

- 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  $s$ .

$$\begin{aligned}
 \textcircled{V(s)} &= \underline{V_t(s)} = \mathbb{E}[G_t | S_t = s] \quad \checkmark && \mathbb{E}[r_t | S_t = s] \\
 &= \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots | S_t = s] \\
 &\stackrel{\mathbb{E}[r_t | S_t = s]}{=} \cancel{R(s)} + \gamma \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | S_t = s] && a_t P = a_{t+1} \\
 &= \underline{R(s)} + \gamma \mathbb{E}[G_{t+1} | S_t = s] = \underline{PV(s)} \\
 &= R(s) + \gamma \sum_{\forall s'} \mathbb{P}[S_{t+1} = \textcircled{s'} | S_t = s] \mathbb{E}[G_{t+1} | S_t = s, S_{t+1} = s'] \\
 &= R(s) + \gamma \sum_{\forall s'} \mathbf{P}_{ss'} \mathbb{E}[G_{t+1} | S_{t+1} = s'] \\
 &= R(s) + \gamma \sum_{\forall s'} \mathbf{P}_{ss'} V_{t+1}(s') \\
 &= \underline{R(s)} + \gamma \sum_{\forall s'} \mathbf{P}_{ss'} V(s')
 \end{aligned}$$

(4)

*Diagram illustrating the Bellman optimality equation:  $V = R + \gamma PV$ . It shows a state vector  $V$  (a column of circles) equal to the sum of a reward vector  $R$  (a column of squares) and a discounted next-state value vector  $\gamma PV$  (a column of squares). Arrows indicate the transition from  $V$  to  $PV$  and then to  $\gamma PV$ .*

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

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

for all  $s \in S$   
(D1, p19)

- The Eq (4) was

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

for all  $s \in S$   
(Bellman)

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

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

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

# Analytic formula

$$V(s) = R(s) + \gamma \sum_{s'} P_{ss'} V(s') \quad \text{for all } \underline{s} \in \mathcal{S}$$

*Handwritten diagram:*  $v \square = R \square + \gamma \square P \square$  with  $\square$  representing a column vector and  $\gamma$  a scalar.

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

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

*Handwritten note:*  $\rightarrow$  Invertible always

## 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

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



```

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,] 13.3536 ✓
## [2,] 12.7439 ✓

```

### Exercise 1

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

$$R = \begin{pmatrix} 1.5 & 1.0 \end{pmatrix}^t$$

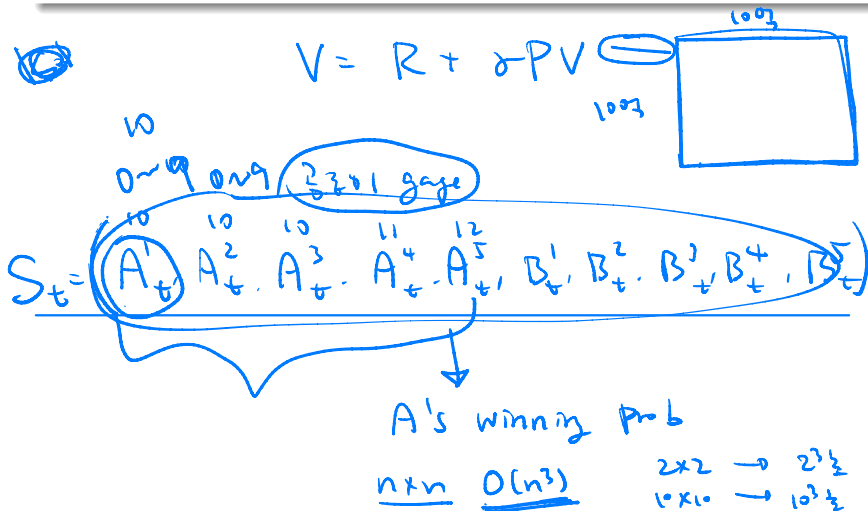
$$W = \begin{pmatrix} 5/8 & 3/8 \end{pmatrix}$$

$$✓ \quad \downarrow \quad 13.36 \times \frac{5}{8} + 12.74 \times \frac{3}{8} = \frac{5/8 \times 1.5 + 3/8 \times 1.0}{1 - 0.9}$$

## Exercise 2

$$|S| = 10^{10}$$

What are your concerns for this approach?



### III. Method 4 - Iterative solution - by fixed point theorem

↓  
AI math

## Recap

i)  $V = 0$

ii)  $\underline{V} \leftarrow R + \gamma P V$

iii)  $\underline{V} \leftarrow R + \gamma P V$

$$\underline{V_{\text{new}}} \leftarrow R + \gamma P \underline{V_{\text{old}}}$$

- The previous approach was based on the following two formula

$$\dot{v} = R + \gamma P v \quad \checkmark \quad (6)$$

$$v = (I - \gamma P)^{-1} R \quad \checkmark \quad (7)$$

- 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  $I - \gamma P$  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  $I - \gamma P$  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

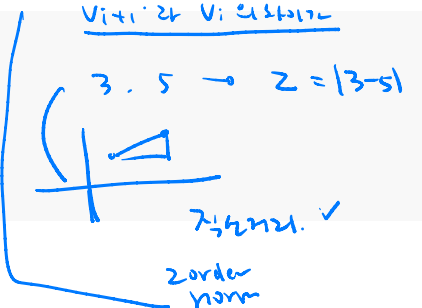
- Using the fixed-point theorem along with Eq. (6), we apply the following iterative algorithm to find  $v$ .

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

$i$  : iteration

```

1: Let epsilon <- 10-8 # or some small number
2: Let v_0 <- zero vector
3: Let v_1 <- R + \gamma P v_0
4: i <- 1
5: While ||v_i - v_{i-1}|| > epsilon # may use any norm
6:   v_{i+1} <- R + \gamma P v_i
7:   i <- i+1
8: Return v_{i+1}
  
```



# Math Review - Norm

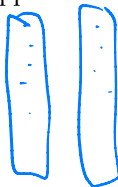
## Definition 1

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)
- $\infty$ -norm:  $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$  (farthest axis)

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



numpy

$$\begin{array}{l} U(-1, 1) \\ \hline 2U(0, 1) - 1 \end{array}$$

# Implementation

## • The psedo code

```

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

```

$$V_{\text{new}} \leftarrow R + \gamma P V_{\text{old}} \quad (\text{do while})$$

## • 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)) [0]
v_new <- R + gamma * P %*% v_old
while (max(abs(v_new - v_old)) > epsilon) { # inf-norm
  v_old <- v_new
  v_new <- R + gamma * P %*% v_old
}
print(v_new)

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

```

python  
do-while

## • The full iteration process

```
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
}
```

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

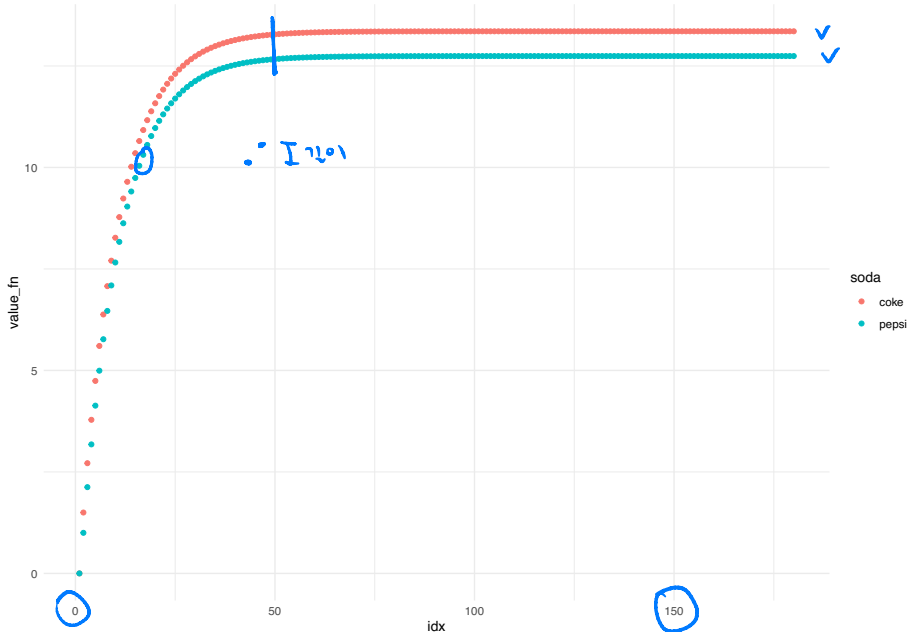
```
head(results)
```

```
##      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
```

```
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
```



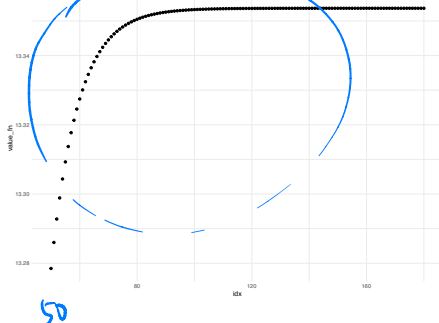


- The previous plot was generated by the following code.

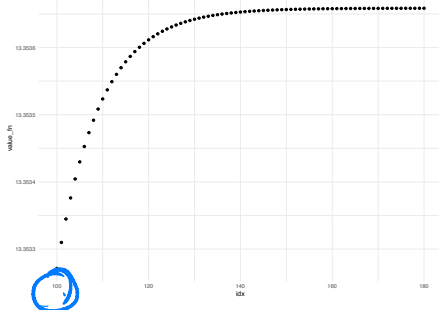
```
library(tidyverse)
results$idx <- as.numeric(row.names(results))
results <- results %>%
  gather("coke", "pepsi", key="soda", value="value_fn")
ggplot(results, aes(x=idx, y=value_fn, group = soda, color = soda)) +
  geom_point() +
  theme_minimal()
```

- Note that there are quite convergence going on after many steps.
- After 50 steps (coke only)
- After 100 steps (coke only)

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



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



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

$\frac{1}{2} / 2^k \gamma$  gamma  $\gamma$   
 gamma  $\in [0.9, 0.9993]$