Stochastic Processes

Lecture 7: value function, Monto Carlo method, first-step method

Hailun Zhang@SDS of CUHK-Shenzhen

February 22, 2021

Discount factor $0 \le \gamma < 1$

Expected total discounted profit over an infinite horizon

$$
\begin{aligned}
v(i) &= \mathbb{E}\Big[g(X_0) + \gamma g(X_1) + \gamma^2 g(X_2) + \dots | X_0 = i\Big] \\
&= \mathbb{E}\Big[\sum_{n=0}^{\infty} \gamma^n g(X_n) | X_0 = i\Big] \\
&= \mathbb{E}\Big[\sum_{n=0}^{\infty} \gamma^n g(X_{n+k}) | X_k = i\Big], \quad i \in \mathcal{S}.
\end{aligned}
$$

Theorem (Bellman equation)

(a) Assume that g is bounded.

$$v = g + \gamma P v. \tag{1}$$

(b) When g is bounded, solution v to (1) exists and is unique.

The theorem holds even if $|S|$ is infinite.

Proof of the theorem

Fix point theorem

- Suppose that $f : x \in \mathbb{R}^d \to f(x) \in \mathbb{R}^d$.
- f is a contraction mapping: there exists an $0 < L < 1$ such that

$$\|f(x) - f(y)\| \leq L\|x - y\| \quad \text{for any } x, y \in \mathbb{R}^m,$$

  where $\|x\|$ is any norm on $\mathbb{R}^m$. For example,
  - $\|x\|_2 = \sqrt{\sum_{i=1}^d x_i^2}$
  - $\|x\|_1 = \sum_{i=1}^d |x_i|$
  - $\|x\|_\infty = \max_{i=1}^d |x_i|$
- Then equation

$$x = f(x)$$

  has a unique fixed point x^*.

Algorithms to compute v when $|S|$ is finite

- When $v = (I - \gamma P)^{-1} g$: Gauss elimination $O(m^3)$
- Value iteration:

$$\begin{aligned} &\text{initialize } v^0 \\ &v^k = g + \gamma P v^{k-1}, \quad k = 1, 2, \ldots \end{aligned}$$

- For any vector u, Pu needs m^2 operations

Discounted total cost: $\gamma = .95$

- Python code:

```
import numpy as np
P= np.matrix("0 .8 .2;
              .5 0 .5;
              1 0 0")
g =np.array([-5,1, 10])
beta =.95
v= ((np.eye(3)- beta* P)**(-1)).dot(g)

v =
   10.991
   15.930
   20.441
```

Algorithm to compute x^*

- Initialize $x^0 \in \mathbb{R}^d$ with any value.
- Define

$$
\begin{aligned}
x^1 &= f(x^0), \\
x^2 &= f(x^1), \\
&\dots \\
x^{k+1} &= f(x^k), \\
&\dots
\end{aligned}
$$

- $\{x^k\}$ is bounded

$$
\begin{aligned}
\|x^{k+1} - x^k\| &\le L^k \|x^1 - x^0\|, \\
\|x^{k+1} - x^0\| &\le \left(1 + L + \dots L^k\right) \|x^1 - x^0\|,
\end{aligned}
$$

- $\{x^k\}$ converges as $k \to \infty$.

Rate of convergence

- $$\|x^{k+1} - x^*\| \leq L^k \|x^1 - x^*\|$$

Revisit: computing the value function

- Suppose that $g(1) = -\$5$, $g(2) = \$1$, $g(3) = \$10$.

- $\mathbb{E}\Big[g(X_1) + g(X_2) + g(X_3) | X_0 = 1\Big]$

$$v^3(i) = \mathbb{E}\Big[g(X_0) + g(X_1) + g(X_2) + g(X_3) | X_0 = i\Big], \quad i = 1, 2, 3.$$

- value function

$$\begin{pmatrix} v^3(1) \\ v^3(2) \\ v^3(3) \end{pmatrix} = (I + P + P^2 + P^3) \begin{pmatrix} -5 \\ 1 \\ 10 \end{pmatrix} = \begin{pmatrix} -1.52 \\ 4.30 \\ 8.80 \end{pmatrix}$$

When P is unknown or too large

- but repreated episodes can be observed ....
- $X_0 = 1$

episode	X_1	X_2	X_3
1	2	3	1
2	2	1	3
3	3	1	3
4	2	1	2
$\vdots$			
100	2	1	3

Monte Carlo method to estimate $v^3(1)$

- Episode 1: $x_0 = 1, X_1 = 2, X_2 = 3, X_3 = 1,$

$$\text{Episode(1) profit} = -5 + 1 + 10 - 5 = 1.$$

- Episode 2: $x_0 = 1, X_1 = 2, X_2 = 1, X_3 = 3,$

$$\text{Episode(2) profit} = -5 + 1 - 5 + 10 = 1.$$

- Episode 3: $x_0 = 1, X_1 = 3, X_2 = 1, X_3 = 3,$

$$\text{Episode(3) profit} = -5 + 10 - 5 + 10 = 10.$$

- Episode 4: $x_0 = 1, X_1 = 2, X_2 = 1, X_3 = 2,$

$$\text{Episode(4) profit} = -5 + 1 - 5 + 1 = -8.$$

Python code for one episode

- Python code:

```
import numpy as np
def next_state(x):
    P =  np.array([[0., 0.8, 0.2],
                   [0.5, 0., 0.5],
                   [1., 0., 0.]]);
    return np.random.choice(np.arange(3), p=P[x,:])

g = np.array([-5, 1, 10]);
x = 0; v = 0; T=4; episode = [];
for t in range(T):
    episode.append(x);
    reward = g[x];
    v = v + reward;
    x = next_state(x)

print(episode); print(v)
```

Monte Carlo method

- Monte Carlo (MC) estimate

$$v^3(1) \approx \hat{v}^L(1) = \frac{1}{L}\sum_{\ell=1}^{L} \text{episode}(\ell)\ \text{profit}$$

- How many episodes?
- Build a confidence interval $[\hat{v}^L(1) - \epsilon, \hat{v}^L(1) + \epsilon]$, where

$$\epsilon = 1.96(\hat{\sigma}^L)/\sqrt{L},$$

and $\hat{\sigma}^L$ is the sample standard deviation

$$\hat{\sigma}^L = \sqrt{\frac{1}{L-1}\sum_{\ell=1}^{L}\Big(\text{episode }(\ell)\ \text{cost} - \hat{v}^L(1)\Big)^2}$$

Python code for multiple episodes (page 1)

- Python code:

```
import numpy as np
def next_state(x):
    P =  np.array([[0., 0.8, 0.2],
                   [0.5, 0., 0.5],
                   [1., 0., 0.]])
    return np.random.choice(np.arange(3), p=P[x,:])

g = np.array([-5, 1, 10]);
value_estimates = [];
T = 4; # time horizon
K = 10000; # number of episodes
gamma = 0.95;  x_0=1;
```

Python code for multiple episodes (page 2: CI)

- Python code:

```
for k in range(K):
    x = x_0;   v = 0;
    for t in range(T):
        reward = g[x];
        v = v + reward;
        x = next_state(x);
    value_estimates.append(v);
value_estimates= np.array(value_estimates)
v = np.mean(value_estimates)
epsilon = 1.96* np.std(value_estimates)/np.sqrt(K)
```

Monto Carlo method for infinite horizon problem

$$v(1) \approx \hat{v}^L(1) = \frac{1}{L} \sum_{\ell=1}^{L} \text{episode}(\ell) \text{ profit}$$

$$\text{episode}(\ell) \text{ profit } =$$

Python code for infinite horizion problem

```
import numpy as np
def next_state(x):
    P =  np.array([[0., 0.8, 0.2], [0.5, 0., 0.5], [1., 0., 0.
    return np.random.choice(np.arange(3), p=P[x,:])

g = np.array([-5, 1, 10]); value_estimates = [];
T = 100;  K = 10000;  gamma = 0.95;  x_0=1;
for k in range(K):
    x = x_0;   v = 0;
    for t in range(T):
        reward = g[x];
        v = v + gamma**t *reward;
        x = next_state(x);
    value_estimates.append(v);
value_estimates= np.array(value_estimates)
v = np.mean(value_estimates)
epsilon = 1.96* np.std(value_estimates)/np.sqrt(K)
```

The first step method

Gambler's ruin problem

- Consider a DTMC with state space $S = \{0, 1, 2, 3, 4\}$ and transition probabilities

$$P_{00} = P_{44} = 1, P_{i,i+1} = .2, \quad P_{i,i-1} = .8.$$

- States 0 and 4 are absorbing states.
- Compute the probability that starting from state 3, the DTMC is eventually absorbed into state 0.

The first-step method

- Let P_i be the probability that starting from state i, the DTMC eventually is absorbed into state 0.
- First step method:

$$P_3 = .8P_2 + .2(0) \tag{2}$$

$$P_2 = .8P_1 + .2P_3 \tag{3}$$

$$P_1 = .8 + .2P_2 \tag{4}$$

- In vector form,

$$\begin{pmatrix} 1 & -.2 & 0 \\ -.8 & 1 & -.2 \\ 0 & -.8 & 1 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0 \\ 0 \end{pmatrix}, \Rightarrow \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} 0.98824 \\ 0.94118 \\ 0.75294 \end{pmatrix}$$

Recursions

- set $P_0 = 1$ and $P_4 = 0$, equations (2)-(4) become

$$P_i = .8P_{i-1} + .2P_{i+1}, \quad i = 1, 2, 3, \tag{5}$$

which is equivalent to

$$.2(P_i - P_{i+1}) = .8(P_{i-1} - P_i), \quad i = 1, 2, 3.$$

- Thus,

$$\begin{aligned}
&P_1 - P_2 = 4(1 - P_1) \\
&P_2 - P_3 = 4(P_1 - P_2) = 4^2(1 - P_1) \\
&(P_3 - 0) = 4(P_2 - P_3) = 4^3(1 - P_1) \\
&P_1 = \left(4 + 4^2 + 4^3\right)(1 - P_1) \Rightarrow P_1 = 1 - \frac{1}{1 + 4 + 4^2 + 4^3} = 0.98824.
\end{aligned}$$