STA4001: Stochastic Process
Project Report

李肖鹏 (116010114)

Due date: Nov. 7, 2019

1 Discrete Time Markov Chain Model

The DTMC $\{X_t\}_{t\geq 0}$ described in the instruction document can be defined in the following recursive manner, i.e.,

$$X_{t+1,i} = (X_{t,i+1} - D_{n+1})^+ - (X_{t,1} - D_{n+1})^+, \quad \forall\, i = 1, 2, \ldots, 5$$

$$X_{t+1,6} = (90 - D_{n+1})^+ - (X_{t,1} - D_{n+1})^+$$

where D_{n+1} is the demand of period $n+1$, and it satisfies Poisson(17) distribution. Therefore, using the above recursion formula, we can generate a sample path of length $10^3 K$, where $K = 20, 200, 2000$ and initial point $X_0 = (0, 0, 0, 0, 0, 0)$. Also, we can obtain the subset $\bar{X} = \{\bar{x}_k\}_{k=1}^{K}$.

To use Monte Carlo method, we first need to compute $g(\bar{x}_k)$ for all $\bar{x}_k \in \bar{X}$. This can be computed by

$$g(\bar{x}_k) = c_p \mathbb{E}[\min\{D, 90\}] - c_v(90 - \bar{x}_{k,6})^+ - c_d(\bar{x}_{k,1} - D)^+ - h\mathbb{E}\left[(90 - D)^+ - (\bar{x}_{k,1} - D)^+\right]$$

where $c_p = 1$, $c_v = 0.4$, $c_d = 0.1$, and $h = 0.1$. Therefore, by Monte Carlo method, we obtain

$$v(\bar{x}_k) = \frac{1}{N}\sum_{n=1}^{N} v^{(n)}(\bar{x}_k) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} \beta^t g\left(X_t^{(n)}, D_t^{(n)}\right), \quad X_0^{(n)} = \bar{x}_k,\ \forall\, n = 1, \ldots N$$

where n denotes the n-th episode and t denotes the period t. In Monte Carlo, $\beta = 0.8$ and

$$g\left(X, D\right) = c_p \min\{D, 90\} - c_v(90 - X_{k,6})^+ - c_d(X_{k,1} - D)^+ - h\left[(90 - D)^+ - (X_{k,1} - D)^+\right]$$

In this way, we can estimate the long run profit by $v(\bar{x}_k)$.

2 Monte Carlo Simulation

In my simulation, I always use period $T = 51$ in each episode for any sample points. This is because for $\beta = 0.8$, $\beta^{50} < 2 \times 10^{-5}$ and the profit is around 15, so a longer period will not give significant change on the total profit. I run six groups of simulation, with parameters (K, N) setting as $(20, 100)$, $(20, 1000)$, $(200, 100)$, $(200, 1000)$, $(2000, 100)$, and $(2000, 1000)$. The result for each

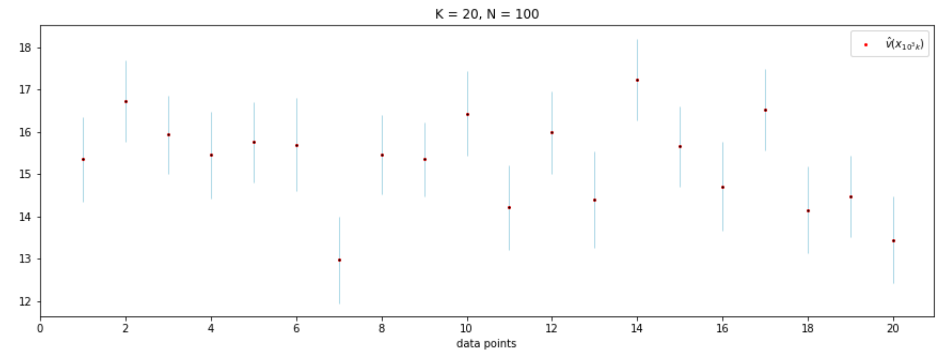
$\hat{v}(\bar{x}_k)$ is just the mean of all $\hat{v}^{(n)}(\bar{x}_k)$. The standard deviation of all $\hat{v}^{(n)}(\bar{x}_k)$ can be used to calculate the 95% confidence interval for each $\hat{v}(\bar{x}_k)$ by

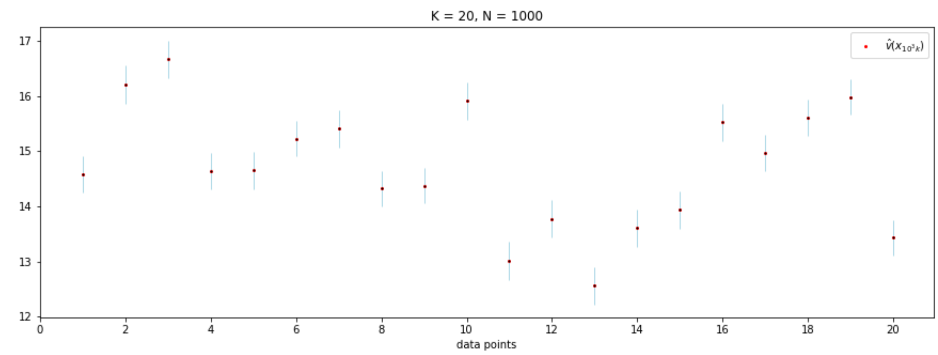
$$v(\bar{x}_k) \in \left[\hat{v}(\bar{x}_k) - 1.96\frac{\sigma_k}{\sqrt{N}}, \hat{v}(\bar{x}_k) + 1.96\frac{\sigma_k}{\sqrt{N}}\right] \tag{CI95}$$

where σ_k is the standard deviation over all $\hat{v}^{(n)}(\bar{x}_k)$ for fixed k. The data are saved in **.csv** files as the following table shows

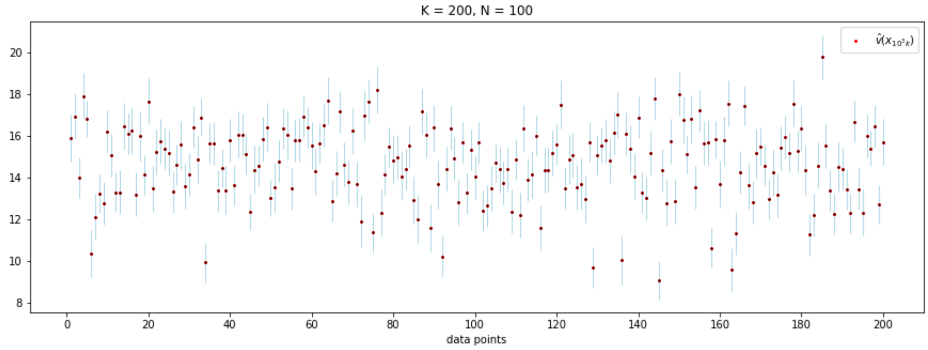
Content	Data type	File name
$\bar{x}_k$ for $K = 20, N = 100$	20×6 numpy array	K20N100X.csv
$\hat{v}(\bar{x}_k), \sigma_k$ for $K = 20, N = 100$	20×2 numpy array	K20N100Y.csv
$\bar{x}_k$ for $K = 20, N = 1000$	20×6 numpy array	K20N1000X.csv
$\hat{v}(\bar{x}_k), \sigma_k$ for $K = 20, N = 1000$	20×2 numpy array	K20N1000Y.csv
$\bar{x}_k$ for $K = 200, N = 100$	200×6 numpy array	K200N100X.csv
$\hat{v}(\bar{x}_k), \sigma_k$ for $K = 200, N = 100$	200×2 numpy array	K200N100Y.csv
$\bar{x}_k$ for $K = 200, N = 1000$	200×6 numpy array	K200N1000X.csv
$\hat{v}(\bar{x}_k), \sigma_k$ for $K = 200, N = 1000$	200×2 numpy array	K200N1000Y.csv
$\bar{x}_k$ for $K = 2000, N = 100$	2000×6 numpy array	K2000N100X.csv
$\hat{v}(\bar{x}_k), \sigma_k$ for $K = 2000, N = 100$	2000×2 numpy array	K2000N100Y.csv
$\bar{x}_k$ for $K = 2000, N = 1000$	2000×6 numpy array	K2000N1000X.csv
$\hat{v}(\bar{x}_k), \sigma_k$ for $K = 2000, N = 1000$	2000×2 numpy array	K2000N1000Y.csv

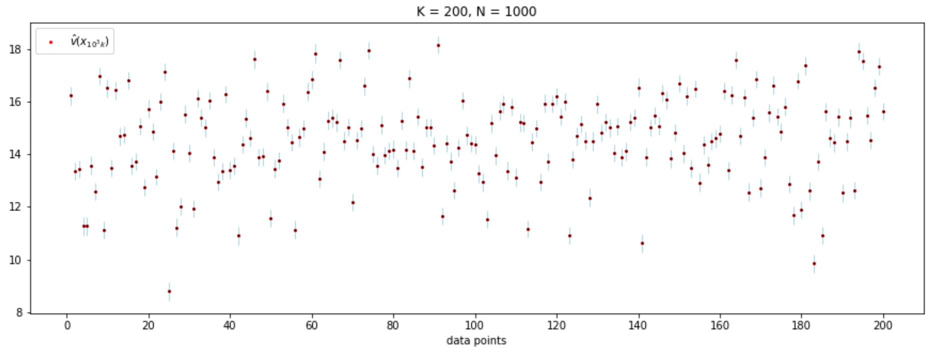
We can easily visualize the data above by the following graph. Notice that in the graph, red points denote data point, light blue lines denote 95% confidence interval.

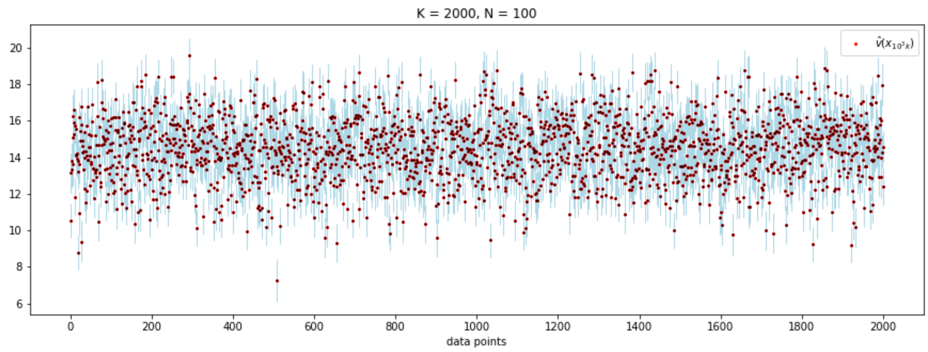


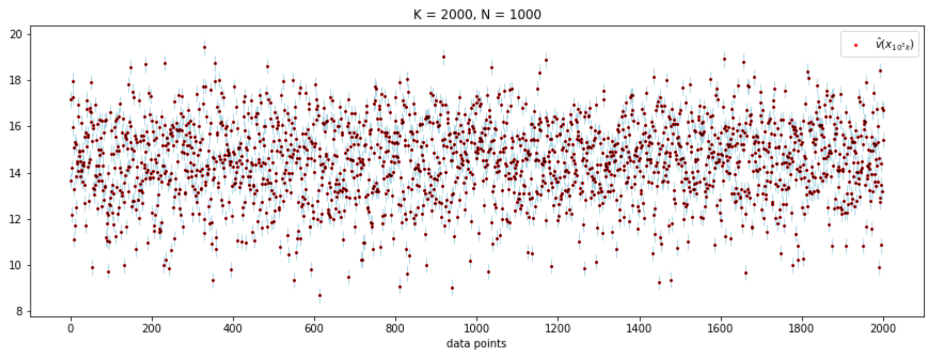


It is clearly that $N = 1000$ gives more narrow confidence interval compared with $N = 100$ case. This is because as the number of episodes increses, the mean expected profit of these epsiodes would gives a better estimation of the true expected profit. The other four graphs are given below.









Notice that the data of confidence interval is calculated by formula (CI95) with data in **.csv** file listed above in the code, so I did not create an extra **.csv** file to save them.

3 Neural Network Simulation

3.1 Size of State Space S

Before we go into the neural network part, we first compute the size of state space $|S|$. Recall that $X_t = (X_{t,1}, X_{t,2}, \ldots, X_{t,6})$ and $X_{t,i}$ denote the number of product with lifetime less than or equal to i. Thus, the constraint is that $X_{t,i} \le X_{t,i+1}$ for all $i = 1, 2, \ldots, 5$, and $X_{t,6} \le 90$. Since it is easy to see that this formulation has a one-to-one corresponding to the other formulation, i.e., $Z_t = (Z_{t,1}, Z_{t,2}, \ldots, Z_{t,6})$ and $Z_{t,i}$ denote the number of product with lifetime exactly equal to i. Then the constraint is $Z_{t,1} + Z_{t,2} + \ldots + Z_{t,6} = 90$ and $Z_{t,i}$ is nonnegative integer. Then the size of S is equal to the number of cases that you put 90 identical balls into 6 different boxes (equivalent to choosing 90 objects from 6 with repetition allowed and order does not matter), which can be solved by the following formula,

$$|S| = \binom{6+90-1}{90} = 57940519$$

It is also easy to see that all states are communicating with $(0,0,0,0,0,0)$. For illustration, consider how $(0,0,0,0,0,0)$ transforms to $(10,20,30,40,50,80)$, it is possible that

$$\begin{aligned}(0,0,0,0,0,0) &\xrightarrow[-10]{+90} (0,0,0,0,0,80) \xrightarrow[-10]{+10} (0,0,0,0,70,80) \xrightarrow[-10]{+10} (0,0,0,60,70,80) \\ &\xrightarrow[-10]{+10} (0,0,50,60,70,80) \xrightarrow[-30]{+10} (0,20,30,40,50,60) \xrightarrow[-10]{+30} (10,20,30,40,50,80)\end{aligned}$$

Therefore, all 57940519 states can be attained and are positve recurrent.

3.2 Neural Network estimation for a single point

For state $x = (10, 20, 30, 40, 50, 80)$, to compute $\hat{v}(x)$, I choose to use episode $N = 1000$ and time period $T = 51$ in each episode. To make the neural network estimation accurate, I use the $\bar{x}_k$ and $\hat{v}(\bar{x}_k)$ with $K = 1000$, $N = 1000$ obtained in part one as the training to train the profit estimating function $f_\theta(x)$. For this neural network, I run 100 epochs with batch size 5, and the result is concluded as follows,

State	$\tilde{v}(x)$	$\hat{v}(x)$	95%CI of $\hat{v}(x)$
$x = (10, 20, 30, 40, 50, 80)$	17.23	17.12	$[16.78, 17.45]$

Notice that this result can be obtained by calling **main1()** function in file **STA4001Project_Q2.py**.

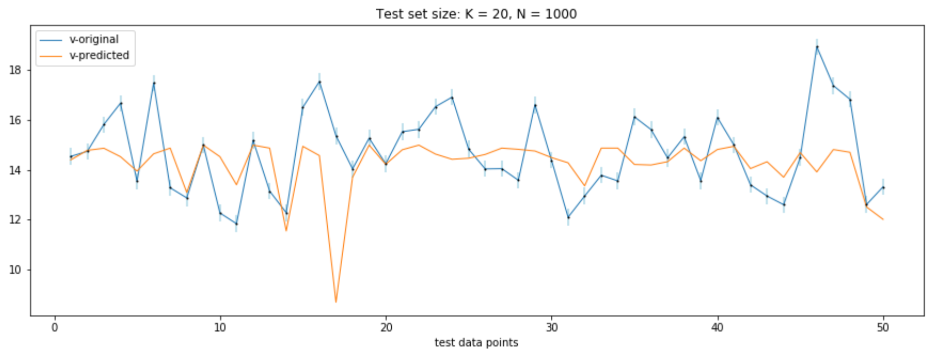
3.3 Neural Network estimation for test set with size 50

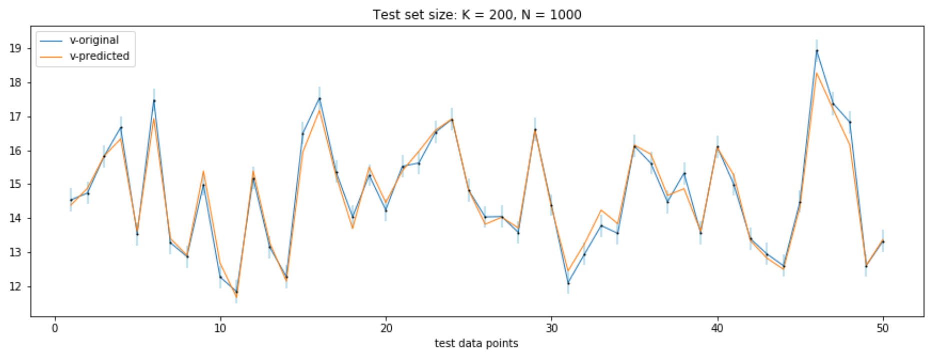
Now I generate a test set $X^* = \{x_j^*\}_{j=1}^{50}$ with size 50 to evaluate the training result of Neural Network. This set is generated by calling function **main2()** in file **STA4001Project_Q2.py**. In this function, we not only generate test set X^* but also using Monte Carlo in part one to calculate the expected long term profit $\hat{v}(x_j^*)$. The data are saved in **.csv** files as the following table shows

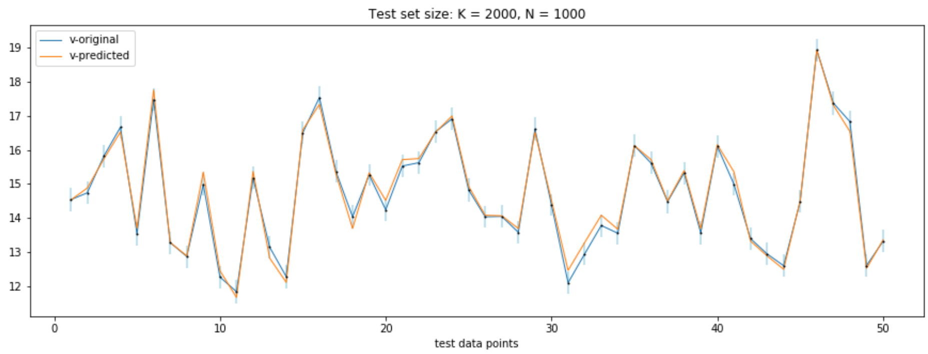
Content	Data type	File name
x_j^* for $j = 1, \ldots, 50$, $N = 1000$	50×6 numpy array	K50N1000X_test.csv
$\hat{v}(x_j^*), \sigma_j^*$ for $j = 1, \ldots, 50$, $N = 1000$	50×2 numpy array	K50N1000Y_test.csv

Here σ_j^* is the standard deviation over all $\hat{v}^{(n)}(x_j^*)$ which is defined similar to $v^{(n)}(\bar{x}_k)$.

Now I try to use three different training sets to train the neural network to compare their estimation accuracy on the test set obtained above. To make the result more robust, I choose the data obtained in part one, i.e., $\bar{x}_k$ and $\hat{v}(\bar{x}_k)$ with $N = 1000$, and $K = 20, 200, 2000$ respectively as the three training sets. Also, I set the epochs as 100 and batch size as 5 for $K = 20, 200, 2000$. The prediction result $\tilde{v}(x)$ for all states in the test set can be drawn in the same graph with $\hat{v}(x)$ as follows,







The mean square error (MSE) for $K = 20, 200, 2000$ is given in the following table,

Test set size	MSE
$K = 20, N = 1000$	3.15
$K = 200, N = 1000$	0.0731
$K = 2000, N = 1000$	0.0299

Notice that the result for prediction through neural network is saved in the following **.csv** files,

Content	Data type	File name
$\tilde{v}(x_j^*)$ with training set size 20	50×1 numpy array	K20N1000Y_train_result.csv
$\tilde{v}(x_j^*)$ with training set size 200	50×1 numpy array	K200N1000Y_train_result.csv
$\tilde{v}(x_j^*)$ with training set size 2000	50×1 numpy array	K2000N1000Y_train_result.csv

A final comment is that this result can be obtained by calling **main3()** function and other codes in the excutive part of file **STA4001Project_Q2.py**.