ORIE 5510 Engineering Stochastic Processes Spring 2022

Jim Dai & Zachary Katz [1]

Course Project

Due: April 11 11:59PM, 2022

We strongly encourage you to complete this project using Python with Google Colab[2]. You are required to use Python packages for neural networks[3]. You should submit your .ipynb notebook file, with each question clearly marked. Each of you is expected to do this project independently. You may discuss this project with your classmates, TAs, and instructors; the primary contact person is Zachary Katz (zdk4@cornell.edu). Document the specific help from others in your submission.

Consider a perishable product with a maximum lifetime of L periods. Thus, at the end of period t, there may be leftover inventory of this product with a remaining lifetime j such that $1 \le j \le L-1$. Any leftover items with 0 remaining life is considered "expired" and will be disposed. Let X_{tj} be the amount of inventory with remaining lifetime less or equal to j (after the "expired items" are disposed), then the system state by the end of period t is $X_t = (X_{t1}, X_{t2}, \cdots, X_{t,L-1})$[4]. Note that the sequence $(X_{t1}, X_{t2}, \cdots, X_{t,L-1})$ is non-decreasing and X_{t1} is the number of items with remaining lifetime 1 and $X_{tj} - X_{t,j-1}$ is the number of items with remaining lifetime exactly $j \in \{2, 3, \cdots, L-1\}$.

By the end of period $t-1$, after disposing expired items, an order for $Q - X_{t-1,L-1}$ units of new items is placed. At the beginning of period t, $Q - X_{t-1,L-1}$ units of new items arrive, bringing the total inventory level to Q. The variable cost of c_v per unit is paid when new inventory arrives. A random demand D_t then occurs and is met immediately as much as possible with inventory *from the oldest to the newest*. Demand across different periods are assumed to be i.i.d. and follow Poisson distribution with mean λ. Unmet demand is lost. At the end of period t, expired inventory is disposed at a cost c_d per unit, and leftover inventory incurs a holding cost h per unit. It can be checked that $X = \{X_t, t = 0, 1, \ldots, \}$ is a DTMC.

We use $\mathcal{S}$ to denote the state space of the Markov chain. The objective is to compute the discounted value function $v(x) = \mathbb{E}[\sum_{n=0}^{\infty} \beta^n g(X_n) | X_0 = x]$ for different initial states $x \in \mathcal{S}$, where $g(X_n)$ is the expected profit **during period $n+1$ (i.e., X_n is the system state by the end of period n).** That is, to compute the expected profit, you need to take into account the variable cost of items purchased at the beginning of period $n+1$;

[1] Adopted from the work of Jim Dai, Mark Gluzman, Lily Liu, & Hailun Zhang

[2] A detailed guide for using Google Colab will be posted for reference.

[3] Please refer to the demo code and "Regression with neural networks in Python" for an example. Google Colab already includes all necessary packages to run the demo code and complete the project.

[4] You may let X_{tj} be the amount of inventory with remaining lifetime equal to j, it is easy to check that these two representations of the system state are equivalent.

the revenues earned on sales based on demand during period $n+1$; and the holding and disposal costs associated with the leftover items at the end of period $n+1$. Let β denote the discount factor. In practice, we use $\mathbb{E}[\sum_{n=0}^{t} \beta^n g(X_n)|X_0 = x]$ to approximate the discounted value function $v(x)$. Here t is the length of an episode and its value depends on the discounted factor β.

Some parameters: selling price $c_p = \$1.$, variable cost $c_v = \$0.4$, disposal cost $c_d = \$0.1$, holding cost $h = \$0.1$ and demand rate $\lambda = 17$.

1. In homework 2, 3 and 4, we only considered $L = 2$; in recitations, we considered examples with $L \geq 3$. Now consider a problem with parameters $Q = 90, L = 7, \beta = 0.8$. Sample a subset $\bar{X} = \{\bar{x}_k\}_{k=1}^{K}$ of the whole state space $\mathcal{S}$ as follows: starting from the initial state $x_0 = (0, ..., 0)$, simulate a full path of the DTMC $\{X_0 = x_0, X_1 = x_1, \cdots, X_T = x_T\}$ for some appropriately chosen T. Then we choose K unique states of interest as our sampled subset.

   After obtaining a subset $\bar{X}$ of K states, for each state $x \in \bar{X}$, we use the Monte Carlo method to estimate its corresponding discounted value function $\hat{v}(x)$, including its 95%-level confidence interval, where x is the initial state. For each state $x \in \bar{X}$, we need to simulate for N episodes, each with length t, to obtain an estimate $\hat{v}(x)$ and its associated confidence interval.

   Specifically, you need to show your work for the following:

   (a) For each $K = 20, 200$, sample a full path of the DTMC $\{X_0 = x_0, X_1 = x_1, \cdots, X_T = x_T\}$ where $T = 1000$. Choose K states to form a subset $\bar{X} = \{\bar{x}_k\}_{k=1}^{K}$ where $\bar{x}_k = x_{1000}$ for $k = 1, \ldots, K$. That is, for each of the K full paths you run, select the 1000^{th} (which is the last) state to form the subset of states. Hint: Create helper function(s) that for any given state determine the next state and compute the expected profit.

   (b) For K=20 and N=100, using the 20 sample states from part (a) to estimate $\hat{v}(\bar{x}_k)$, where $\bar{x}_k$ is the initial state, for $k = 1, \ldots, 20$. Clearly show each initial state $\bar{x}_k$, the estimated value $\hat{v}(\bar{x}_k)$, and its 95%-level confidence interval.[5]
   Note that you will need to choose the period of each episode t. There are many possibilities, but note that your selection depends on the discount factor β. Please specify the length of the episode t you chose and briefly reason (1-2 sentences) why.

   (c) For each value of $K = 20, 200$ (using one case $N = 100$ and the other case $N = 1000$ as the number of episodes), plot the estimated value $\hat{v}(\bar{x}_k)$ with its 95%-level confidence interval, with $k = 1, \ldots, K$ as the x-coordinates. (You should plot $2 * 2 = 4$ figures).[6]

[5] You can use a table with three columns to show them or print out the values in Jupyter notebook. Either way, be sure to make it clear of each state and its corresponding values.

[6] You can look more into matplotlib.pyplot.errorbar for details to plot points with confidence interval.

Please use the $K = 20$ and $K = 200$ sampled states from part a as your initial states. Note that the values needed to create the plot for the first case, $K = 20$ and $N = 100$, were already computed in Question 1(b).

2. As you can see, the state space of the problem is large. In many applications, the state space is even larger. Therefore, storing $\hat{v}(x)$ for *all* $x \in \mathcal{S}$ in computer memory may be infeasible. Instead of remembering the pre-computed value $\hat{v}(x)$, one can compute in *real time* every time a state x is observed. However, the Monte Carlo method that we use above can sometimes be too slow to estimate $v(x)$.

   One possible solution to the preceding challenge is to find a family of functions

   $$f_\theta : x \in \mathcal{S} \to f_\theta(x) \in R,$$

   where $\theta \in \mathbb{R}^d$ is a vector of d parameters to be computed in advance (known as offline computation) and d is some positive integer. Assume that θ has been computed and is fixed. For each given $x \in \mathcal{S}$, $f_\theta(x)$ can be computed quickly (online or in real time). We use $f_\theta(x)$ to estimate $v(x)$. Namely,

   $$\tilde{v}(x) = f_\theta(x), \quad x \in \mathcal{S}$$

   provides another estimate of $v(x)$. Unlike the Monte Carlo estimate $\hat{v}(x)$, $\tilde{v}(x)$ can be computed much faster. Conceptually, $\theta \in \mathbb{R}^d$ can be computed in advance using a cloud computing facility for many days or weeks; with the given θ, $f_\theta(x)$ can be computed, for example, from the dashboard computer in a car, quickly for each input x.

   One can use regression to find a value of $\theta \in \mathbb{R}^d$. For that, we assume

   $$\big(x_k \in \mathcal{S}, \quad \hat{v}(x_k) \in \mathbb{R}\big), \quad k = 1, \ldots, m$$

   is given. Then $\theta \in \mathbb{R}^d$ can be computed to minimize

   $$\sum_{k=1}^{m} \big(f_\theta(x_k) - \hat{v}(x_k)\big)^2 \tag{1}$$

   In this project, you will need to use a neural network with weight vector θ (see the demo code for the construction of a neural network) to represent $f_\theta(x)$. Specifically, we will choose $m = K$, where K is the number of initial states in Problem 1; thus, x_k and the Monte Carlo estimate $\hat{v}(x_k)$ are available for $k = 1, \ldots, K$ as obtained from Problem 1. We will use

   $$\big(\bar{x}_k, \hat{v}(\bar{x}_k)\big) \quad k = 1, \ldots, K$$

   to train the neural network parameter θ by solving the optimization problem (1).

   Once θ is computed,

(a) Please build a neural network and train it using the 200 sampled states from Question 1(a). Use the corresponding value estimates $\hat{v}(x)$ from Question 1(c) as training values.

Next, for the initial state $x = (10, 20, 30, 40, 50, 80)$, please list $\tilde{v}(x)$, $\hat{v}(x)$, and the 95% confidence interval (CI) of $\hat{v}(x)$. To estimate $\hat{v}(x)$ for this initial state, use the number of episodes $N = 1000$ each with length $t = 100$.

A demo code (demo.ipynb) will be posted for reference, together with a detailed document "Regression with neural networks in Python".

(b) Run the DTMC again for 50000 periods starting from state $(10, 20, 30, 40, 50, 80)$. Record $X_{1000}, X_{2000}, \ldots X_{50000}$ into a test set of 50 states. Plot $\tilde{v}(x)$ and $\hat{v}(x)$ for all states x in this test set. To compute $\hat{v}(x)$, use $N = 1000$ episodes with $t = 100$ episode length *for each of the 50 states*. Use your neural network from part a to compute $\tilde{v}(x)$.

Your plot should have state # (1-50 from the 50 states you just obtained) on the x-axis and the resulting value estimates on the y-axis. $\tilde{v}(x)$ and $\hat{v}(x)$ should be included on the same plot so that you can compare the value estimates. Briefly comment (1-2 sentences) on the results.

(c) Finally, please use as test sets the $k = 20$ and $k = 200$ initial states from Question 1(a), as well as their respective $N = 1000$ value estimates from Question 1(c). Compute the mean squared error (MSE) between the estimates from Question 1(c), $\hat{v}(x)$, and the estimates from the neural network, $\tilde{v}(x)$. Your final answer for this part should be two MSEs, one for the $K = 20$ initial states and the other for the $K = 200$ initial states.

You may write your own function to compute MSE, or you may use the scikit-learn implementation [7].

[7] scikit-learn MSE implementation