ORIE Final Project 2020

This project centers on a standard inventory model. It has three parts. The first part concerns Markov chain modeling, coding, and analysis. The second part is writing and coding an optimal control (dynamic programming) problem and the last touches on a basic reinforcement learning algorithm.

Problem

In the morning of every day, the manager looks at the remaining inventory (or backlog) of inventory and places an order for items. We will assume that the order arrives immediately and before any demand is realized (it arrives before the store opens). For now, let us assume that the manager follows the policy: order enough items to bring the inventory up to 50. If the inventory is already at or above 50, the manager orders nothing.

Demand arrives after the order is placed and received. Let D_t be the demand arriving at time t. We will assume that this demand follows a Poisson distribution with mean $\lambda=10$:

$$E[D_t] = 10, \;\; P(D_t = k) = \frac{e^{-10}10^k}{k!}$$

If demand is larger than available inventory, the excess orders are backlogged which means that the customer will get the items as soon as more inventory arrives. Thus, **inventory can be negative.** If the inventory, for example, is -3 at the beginning of the day it means that there are three customers to whom we owe items. When the order arrives, the first three items will go to these customers so that---under the policy of ordering up to 50---the inventory available for the next-period customers will be 50-3=47.

We assume throughout that the inventory cannot go below -30. If there are already 30 customers to whom we owe products, any new orders are lost. The inventory, thus, can never go below -30.

Part 1: Markov chain modeling

Let X_t be the inventory at the beginning of day t (before orders are placed and demand is realized).

(1) Write down a (discrete time) Markov chain model for X_t. This requires you to specify fully the transition probability matrix.
(2) Code, in Gurobi, a linear program that gives as output the stationary distribution of this Markov chain and use that to answer:
    a. What is the long-run average number of units in inventory
    b. What is the service level: what is the fraction of days where some customers are backlogged (not all demand is satisfied).

Part 2: Optimization via Dynamic Programming

In the model description we assumed that the manager follows the policy of ordering-up-to 50 units but this is not necessarily the optimal thing to do. We are now going to derive the optimal policy.

To make our life a bit simpler, we will that there is limited space in the warehouse. The manager can never bring the inventory to more than 100 units. The state space is then [-30,-29,….,100].

Suppose that the company pays \$c per unit ordered (i.e., if you order 10 units, you pay \$10c). The company also pays a capital cost of \$h per unit in inventory per period of time (i.e, if there are 20 unit in inventory at the end of a day, the company incurs \$20h of inventory costs). Suppose that the company loses \$b for each customer that is backlogged per unit of time. So if the inventory is -3 at the end of a period, that cost is \$3b for that period.

The manager wants to minimize the expected infinite-horizon discounted cost. The discount factor is 0.9.

(1) Write the Bellman equation for this problem.
(2) Write a value iteration code in python. The code should take as input the parameters λ (the mean per-period demand), c, b and h.
(3) Fix c=1 and $\lambda = 15$. Choose 5 different combinations of parameters b,h (large h and small b, small h and large b and so on) and for each
    a. report the optimal value starting at time 0 with x=20 units in inventory,
    b. What is the optimal strategy, i.e., what is the optimal order quantity or order up to level for each state.

<u>Part 3: Simulation-based optimization and reinforcement learning</u>

(1) Read on the Q-Learning algorithm
(2) Code the Q-learning algorithm for the inventory problem
(3) Picking one set of inputs (λ, c, h, b) from part 2, provide visuals that capture the convergence of the Q-learning algorithm to the true value function and strategy (that you know from part 2).