Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/79/20241669

Optimizing hospital outpatient services: A comparative study of backward induction and Q-learning techniques

Shilin Zhang

Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, Guangzhou, 511442, China

3278184231@qq.com

Abstract. This study addresses the critical issue of optimizing outpatient services in high-capacity hospitals, focusing on developing cost-effective management strategies. Utilizing a simulated model of outpatient services, this research incorporates real data from the National Health Service (NHS) to tackle practical challenges in hospital management. The methodology encompasses the application of backward induction, Q-learning, and Deep Q-Network (DQN) algorithms to formulate solutions. The findings indicate that backward induction effectively resolves simpler scenarios within the assumed conditions. In contrast, Q-learning offers a viable approach, with DQN demonstrating superior performance in addressing more complex, realistic problems. The conclusion drawn from this study is that each algorithm exhibits unique strengths in its respective operational environment. While direct comparison between the models based on output analysis is not feasible due to the variation in environmental settings, it is evident that all three algorithms significantly contribute to resolving the targeted issues in outpatient service management. This research not only provides valuable insights into hospital outpatient service optimization but also opens avenues for further exploration in the application of advanced computational techniques in healthcare management.

Keywords: Hospital Outpatient Services, Cost-Effective Strategies, Backward Induction, Q-Learning, Deep Q-Network (DQN), Healthcare Service Optimization.

1. Introduction

With the advancement of medical technology and the increasing number of patients, it has become especially important to analyze medical data rationally and make decisions about outpatient situations. We are faced with many issues to address, such as how to allocate doctors for outpatient services rationally and predict the number of patients. Therefore, we are eager to use cutting-edge approaches to analyze and address these challenges.

In outpatient services, cost management encompasses various aspects, including the waiting costs for patients, treatment costs, and the costs of hiring doctors. To deliver efficient and affordable healthcare, we need a method to balance these three costs, aiming not only to compute the lowest cost but also to identify the optimal solution to achieve it. Regarding disease issues, we attempt to analyze available public medical datasets, seeking to uncover the causes of such diseases and their related factors.

In real-world outpatient hospital services, effective cost management not only conserves hospital resources but also enhances patient satisfaction and delivers higher-quality medical care. Also, with

DOI: 10.54254/2755-2721/79/20241669

appropriate analysis and prediction of diseases, we can channel medical resources to where they are more needed, optimizing the medical surveys and implementation methods currently in use [1].

Throughout the problem-solving process, we encountered an array of challenges. Some of the most significant and demanding hurdles we faced included:

1.Data Accessibility: Medical data, given its sensitive nature, is often challenging to obtain. For our analysis, we required detailed information on the hourly patient count and individual physician salaries. Such specific data is rarely available to the public, posing a significant obstacle in our endeavors.

2.Dynamic Nature of the Environments: Both environments we're addressing evolve over time. This dynamic nature means we can't merely preset values and solve corresponding problems. We must select representative data over an extended period and adjust our strategies for each unique environment post-analysis.

3.Disease Analysis: Not all diseases are suitable for our study. While many diseases strongly correlate with single factors like climate, gender, or region, nearly all diseases have multiple inducing factors. Identifying the right diseases and the associated data for analysis poses a substantial challenge. Additionally, the accuracy of our analysis requires rigorous scientific validation.

4.Real-world Implementation Factors: Even with the most advanced analysis and optimal strategies, implementing these changes in a real-world hospital setting entails navigating bureaucratic roadblocks, ensuring comprehensive staff training, and managing potential resistance to change. Moreover, our pursuit of minimized costs must not jeopardize the quality of medical services. Ensuring our cost-saving measures do not negatively impact patient care [2,3].

2. Literature Review

2.1. Former researchers' work

The intricacies of outpatient chemotherapy appointment scheduling are multifaceted, especially when factoring in the unpredictability of patient deferrals. Such deferrals can arise due to various reasons, including patients' health conditions or unforeseen emergencies, adding layers of complexity to an already challenging scheduling task. The paper titled "Daily outpatient chemotherapy appointment scheduling with random deferrals" by Thierry Garaix, Salim Rostami, and Xiaolan Xie delves into these complexities, aiming to devise efficient scheduling strategies to accommodate such uncertainties.

The algorithm they use is GRASP (Greedy Randomized Adaptive Search Procedure) [4]. GRASP is a heuristic specifically designed to tackle combinatorial optimization problems. In the context of this paper, it's adeptly utilized to compute treatment sequences for patients in outpatient chemotherapy centers.

The core strength of GRASP lies in its constructive procedure. This procedure incrementally builds a list of patients, ensuring the generation of high-quality sequences even for instances with a large number of patients. The algorithm incorporates a double justification technique, a known method to improve the quality of scheduling solutions. This technique enhances the efficiency of the scheduling process without significantly increasing computational efforts.

There are also some limitations to GRASP. One of the main limitations of the GRASP algorithm is the time-consuming nature of its evaluation process. This evaluation, which is done through simulation, can be particularly lengthy, especially for larger datasets. Also, the performance of the GRASP algorithm can be influenced by external factors, such as the number of replications used in the simulation [5]. This means the algorithm's efficiency might vary based on the specific parameters set during execution.

2.2. How to differentiate your approach from existing works

It's imperative to differentiate our endeavors from the work of predecessors. One standout distinction is our strategy of beginning with a basic model and a straightforward algorithm. As we progressively introduce more determinative factors, we evolve our model and the theoretical techniques we employ. Our approach to handling medical data also diverges. Given that available public datasets in the

DOI: 10.54254/2755-2721/79/20241669

healthcare domain are quite limited, we start our analysis with smaller datasets. After constructing our model, deriving results, and validating its feasibility, we then escalate the dataset size and volume, subsequently delving deeper into theoretical computations. Due to the absence of permissions and conditions for practical implementation, our theories will be verified in the virtual environment we've designed. This will serve to evaluate the efficacy of our efforts.

3. Main problem / Model

3.1. Define the problem

Now, we assume that there's a local hospital providing outpatient services. Every day, the hospital opens for 8 hours, from 9 a.m. to 17 p.m. At the beginning of the day and each hour, there will be some patients entering the hospital, and the hospital can hire extra doctors to deal with the patients [6,7]. At the beginning of the day, the hospital have 10 free doctors in total, and each extra doctor cost $500 per hour, even if the doctor is not at work for this hour. If a patient is treated, the treatment cost will be $30; if a patient was left untreated for an hour, $300 should be added to the cost amount. Each patient can be treated in an hour, and each doctor can heal 2 patients each hour. Our goal is to find the minimum cost for the whole day. The trend of patient visits to the outpatient clinic corresponds to Figure1:

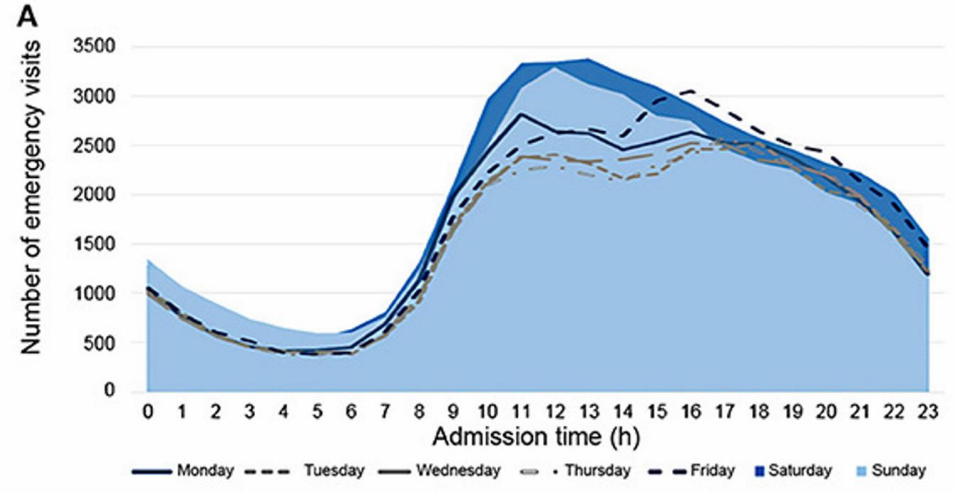

Figure 1. Number of emergency visits to the outpatient through time [8].

Regarding the NHS data analysis, we can conceptualize our model as a dynamic interplay between patients and the hospital. Within a hospital setting, daily visitors comprise both scheduled appointments and walk-ins. Additionally, there are instances of patients who, despite having an appointment, fail to show up. For subsequent visits, there's a mix of patients who skip follow-up appointments and those who attend them as scheduled. Our objective is to optimize resource allocation, be it minimizing bed utilization or employing the least number of doctors, without compromising patient care quality. To navigate this complexity, we initially abstract these variables into a unified concept of "resources." This abstraction allows us to modify and adapt our approach across diverse scenarios.

Mathematical formulations:

Stages: $t = 1, \ldots, T$ hours of the day

States: x_t : number of patients in the waiting room at time t

Decisions: u_t: number of additional doctors staffed in hour t

Parameter: d_t: patient demand during hour t

Transition function:

$$f_t(x_t, u_t) = \max\left[x_t + d_t - 2 \times (10 + u_t), 0\right] \quad (1)$$

DOI: 10.54254/2755-2721/79/20241669

Cost function:

$$g_t(x_t, u_t) = w \times x_t + 500 \times u_t \tag{2}$$

$$g_{T+1}(x_{T+1}) = W \times x_{T+1} \tag{3}$$

Bellman equation: Cost of state x_t at time t

$$J_t(x_t) = \min_{u_t \geq 0}\{w \times x_t + 500 \times u_t + J_{t+1}(\max[x_t + d_t - 2 \times (10 + u_t), 0])\} \tag{4}$$

$$J_{T+1}(x_{T+1}) = W \times x_{T+1} \tag{5}$$

4. Data

4.1. Obtaining the data

Obtaining data from hospital outpatient clinics can be a cumbersome process. As we don't have direct access to actual data from real-world outpatient settings, our data source is confined to the internet and publicly available archives. After some research, we procured a line chart depicting the number of outpatient visits to a particular hospital as it varied throughout the day. This chart provides an intuitive reflection of the actual trend in patient visitation numbers and serves as the foundational dataset for our study. Obtaining comprehensive patient data presents a distinct set of challenges. Given the sensitive nature of such information, we opted for datasets from the NHS hospitals in the UK, which are suitable for our research endeavors. We sourced this dataset from publicly available online platforms, making it accessible and feasible for our analysis.

4.2. Clean the data

The process of cleaning the data is totally different. For the first situation, we want some random data but also like the trend of the chart given. So, in each node, we take an average number consistent with the chart and a random number that satisfies a normal distribution. Cleaning the data from NHS is another one. Most of the numbers we need are around 50k to 200k, so we have to ensure all the data we need for calculation is in this range and eliminate the wrong data using a classifier.

5. Solution Approach

5.1. Intuition of the solution

In the environments presented by these two problems, we face a common dilemma: finding the best decision-making strategy. One feasible approach is to compute all possible policies and their resulting returns, allowing us to make a selection. Such computation is typically intense but can easily find the theoretically optimal return. We map all decisions, returns, and states into a grid and compute backward from the end state—this method is called backward induction. It offers us the possibility to address these two issues. However, when the amount of data reaches a relatively high level, we cannot use backward induction, as we lack the computational power to process a matrix representing all states with a dimension of 200k by 200k. At this point, Q-learning becomes our preferred method. Q-learning is a model-free learning method. We only need to provide the state, decision, and reward function, and the algorithm will attempt to find a way to achieve the maximum return. The results obtained by Q-learning are much better than those of backward induction, especially in the second problem. We don't need to traverse all possible scenarios. With a certain number of training steps, the program returns a result with considerable credibility.

5.2. Precise statement

The pseudo code of backward induction of the first problem is shown in Algorithm 1:

DOI: 10.54254/2755-2721/79/20241669

Algorithm 1: Pseudo code for problem1

Input:
Daily data of the outpatient
Output:
The best choice for doctor management
Begin

```
DEFINE function d_t(t, open_time)
    RETURN new patient count at time t based on a predefined distribution
DEFINE function f(x_t, u_t, t, open_time, capacity)
    COMPUTE and RETURN queue length at time t+1
DEFINE function g_t(x_t, u_t)
    COMPUTE and RETURN cost at time t
DEFINE function g_T(x_T)
    RETURN end state cost
DEFINE function backward_induction(state_space, action_space, T, open_time, capacity)
    INITIALIZE matrices U_t and J_t
    SET initial values for the final column of J_t
    FOR each time t from T-1 to 0 in reverse
        FOR each state x_t
            FOR each action u_t
                CALCULATE current cost
                IF current cost is lesser than previously noted cost
                    UPDATE cost and policy values
    RETURN U_t and J_t
MAIN program starts
    SET parameters
    INVOKE backward_induction to get U_t and J_t
    PRINT U_t and J_t
VISUALIZE the results
```

End

The code simulates patient arrival and waiting scenarios in a hospital setting, with the aim to determine the optimal policy for hiring on-demand doctors such that the overall cost is minimized. This cost encapsulates patient waiting cost and the cost of hiring temporary doctors.

The code sets up several core functions:

d_t(t, open_time): Returns the number of new patients arriving at the hospital at time t.

f(x_t, u_t, t, open_time, capacity): Transition function which yields the queue length at time t+1.

g_t(x_t, u_t): Cost function that provides the cost at time t given a queue length x_t and the number of on-demand doctors u_t.

g_T(x_T): End state cost function detailing the cost associated with untreated patients.

backward_induction(state_space, action_space, T, open_time, capacity): Uses backward induction to determine the optimal policy and its associated cost.

For Q-Learning, the pseudocode is shown in Algorithm 2.

Algorithm 2: Pseudo code for problem1, Q-learning

Input:
Daily data of the outpatient, randomly
Output:
The best choice for doctor management
Begin

```
1.Class HospitalEnvironment:
2.  Initialize with parameters distribution, capacity, open_time
3.      Set distribution, capacity, open_time
4.      Reset the environment
```

DOI: 10.54254/2755-2721/79/20241669

```
    Function reset:
        Reset queue length to 0
        Set current time to open time
        Initialize new patients to an array of zeros
    Function step with parameter action:
        Simulate new patients arriving from a Poisson distribution
        Adjust queue length based on new patients and action taken
        Compute cost based on queue length and action
        Advance time by 1 hour
        Return new queue length and cost
Class QLearningAgent:
            Initialize with parameters state_space, action_space, learning_rate, discount_factor, exploration_rate
        Set state space, action space, learning rate, discount factor, exploration rate
        Initialize Q-table with zeros
    Function learn with parameters current_state, action, reward, next_state:
        Calculate max future Q-value for next state
        Update Q-value for the current state-action pair using the Q-learning formula
    Function act with parameter state:
        With probability exploration_rate:
            Return random action
        Else:
            Return action with the highest Q-value for the given state
Main Execution:
    Initialize HospitalEnvironment and QLearningAgent
    Set the number of episodes for learning
    For each episode:
        Reset the environment
        For each time step (hour) within the episode:
            Decide on an action using the act function
            Execute the action in the environment
            Receive the next state and cost from the environment
            Convert the cost into a reward (negative cost)
            Update the Q-table using the learn function

    After all episodes, visualize the average cost per episode and Q-table
    Test the learned Q-table over a day and visualize the results
End
```

The program simulates a hospital environment where patients arrive at random intervals, and the hospital decides how to allocate doctors based on the current length of the queue.

The Hospital Environment class represents the dynamics of the hospital. It computes the new patients arriving every hour and adjusts the queue based on the available doctors. The cost incurred each hour is also calculated here.

The QLearningAgent represents an agent that learns to make decisions using the Q-learning algorithm. The learn method updates the Q-table using the Q-learning update rule, while the act method decides which action to take. Sometimes, it explores by choosing a random action, and other times, it exploits by choosing the best-known action from its Q-table [9,10].

The main execution sets up and simulates the environment and the learning agent. Throughout the simulation, the agent tries to learn the best action to take in each state (queue length) and timestep (hour of the day) to minimize the total cost. It tracks and periodically displays the average cost produced by the agent. After learning, it tests the Q-table over a day to see how well the agent performs and visualizes the results using a heatmap and plots.

DOI: 10.54254/2755-2721/79/20241669

The code updates these values using a formula based on the reward received and expected future rewards. The Epsilon-greedy policy balances exploration (trying out new actions) and exploitation (using the best-known action). The agent takes a random action with some probability (epsilon) and otherwise chooses the best-known action. The Q-table is a table the agent uses to store values for each state-action pair. It uses this table to decide which action to take in a given state. Over time and with learning, this table gradually converges to the true values, enabling the agent to make optimal decisions [11]. For the second question, the pseudo code of backward induction is shown in Algorithm 3:

Algorithm 3: Pseudo code for problem2, backward induction

Input:
Out patient data from NHS
Output:
The best choice for doctor management
Begin
1.Initialize data for elective_total, non_elective, first_attendances, subsequent_attendances, first_dna, subsequent_dna for three months.
2.Set total_resources = 200000
3.Define reward function:
4. Compute total reward based on resources allocated and actual demand.
5. Penalize deviations from total_resources.
6. Return computed reward
7.Initialize an empty list for best_allocations
8.For each month starting from the last to the first:
9. Initialize best_reward to negative infinity and best_allocation to (0, 0, 0, 0)
10. For each possible allocation to elective procedures:
11. For each possible allocation to non-elective procedures:
12. For each possible allocation to first-time attendances:
13. Compute the allocation for subsequent attendances by subtracting from total_resources.
14. Compute the reward for the current allocation.
15. If current reward is better than best_reward:
16. Update best_reward and best_allocation
17. Add the best_allocation for this month to the front of best_allocations list
18.For each month, display the best allocation.
End

The code set monthly data for different medical services, including elective and non-elective procedures, first-time attendances, subsequent attendances, and their corresponding non-attendances for three months. With a fixed resource count, a reward function is defined to calculate benefits from resource allocation. It summarizes resources for elective, non-elective, first-time, and subsequent attendances, adjusting for non-attendances and applying penalties for over or under-allocation. The main logic employs backward induction, starting from the third month and moving to the first, iterating through all possible resource combinations, computing rewards, and identifying the best allocation for each month based on maximum reward. Finally, the optimal allocation for each month is displayed. Algorithm 4 shows the pseudo-code of Q-learning for the second problem:

Algorithm 4:Pseudo code for problem2, Q-learning

Input:
Out patient data from NHS
Output:
The best choice for doctor management
Begin
1.Initialize medical data for three months
2.Set parameters (total_resources, alpha, gamma, epsilon, etc.)
3.Initialize Q-table with zeros

DOI: 10.54254/2755-2721/79/20241669

```
4.DEFINE reward function based on allocations and penalties
5.FOR each episode DO:
6.    Set current month to 0
7.    Randomly initialize allocations for each service
8.    FOR each step within an episode DO:
9.        IF random value < epsilon THEN
10.           Choose random action
11.       ELSE
12.           Choose the best action from Q-table
13.       Update allocations based on chosen action
14.       Clamp allocations within allowed limits
15.       Calculate reward for new allocations
16.       IF not last month THEN
17.           Update Q-value using the Q-learning formula considering future rewards
18.       ELSE
19.           Update Q-value considering only current reward
20.       Move to the next month
21.       IF last month THEN
22.           Break out of the loop
23.   Print episode progress after every 100 episodes
24.Identify best allocations for each month from Q-table
25.Print best allocations for each month
End
```

The code employs Q-learning for optimal resource allocation across three months for various medical services. Initially, it sets up monthly data and parameters such as total resources, alpha, gamma, epsilon, and more. A Q-table, shaped as a five-dimensional array, is created and initialized to zeros. The reward function then assesses the allocations, accounting for over or under-allocations. In the Q-learning phase, each episode begins with randomized allocations, and for each step, the system determines if it should explore or exploit. Based on this decision, allocations are adjusted, rewards are computed, and the Q-values are updated. This continues until the last month or when the episode hits its step limit. Post-completion of all episodes, the algorithm pinpoints the optimal allocation by referencing the highest Q-value for every month. Finally, these optimal allocations are displayed.

6. Experiments

After a lengthy process of coding and debugging, we are finally ready to apply the code to attempt to solve the problem. The solution for the first issue visualized using backward induction is shown in Figure 2:

DOI: 10.54254/2755-2721/79/20241669

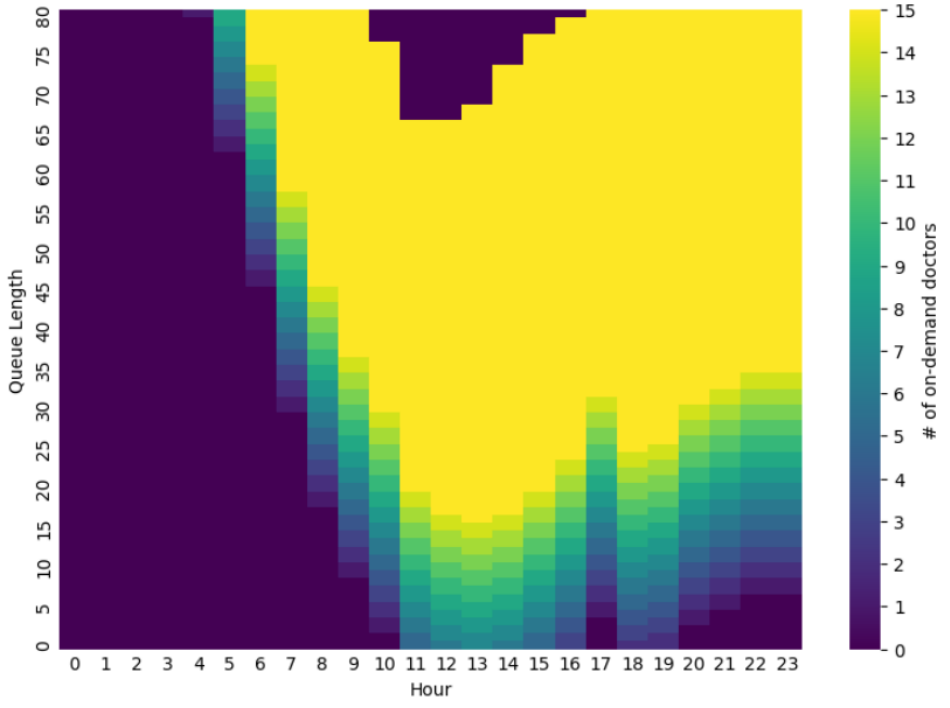

Figure 2. The relationship between queue length and demand of doctors.

In it, different colors represent the number of doctors that the program believes should be hired in the decision-making process. The program also provides the minimum cost we can achieve:Total cost for the day: 24610.0. For the deployment of the Q-learning method, there is a different Q-table for each hour. Figure3 shows one of the Q-tables:

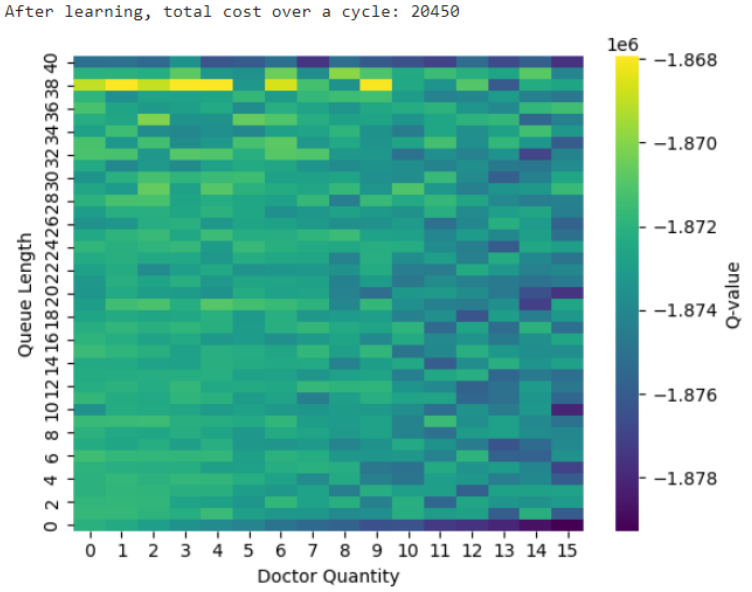

Figure 3. One of the Q-tables of the whole process.

Similarly, after applying Q-learning, the program calculates the minimum cost we can achieve. Comparing the backward induction and Q-learning methods, the results of backward induction are

DOI: 10.54254/2755-2721/79/20241669

undoubtedly the best in theory. However, Q-learning significantly reduces the time needed to achieve a good result, and its performance can be further improved by increasing the number of learning steps. For the second problem, the output result of backward induction is shown as Figure4:

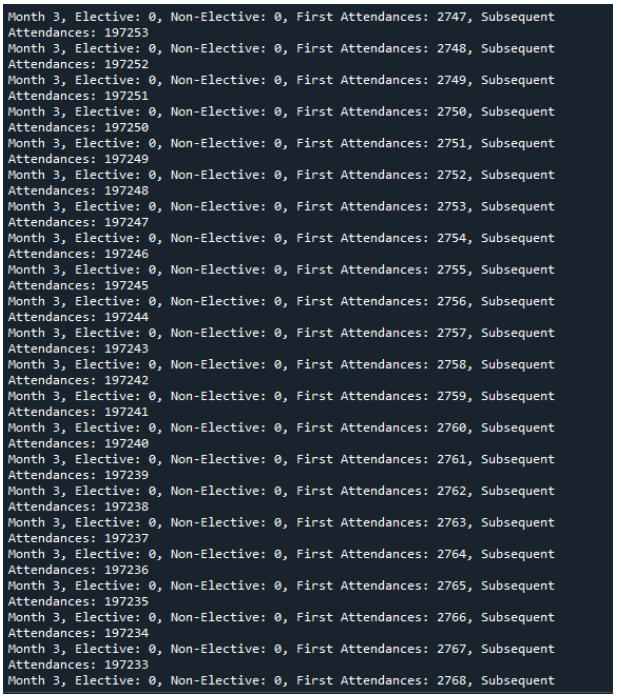

Figure 4. The output results are not as expected.

Since we did not obtain an output result long after setting the range, we tried using Q-learning to address this issue. In the end, it quickly provided some results.

7. Conclusion

In this research, we defined two distinct environments, proposed two different problems, and adopted similar approaches to attempt their resolution. In the first environment, we sought to address the issue of outpatient doctor hiring for a hospital, utilizing both backward induction and Q-learning to determine the minimal cost attainable. Ultimately, we successfully navigated this environment, identifying various strategies for hiring doctors based on different patient visitation numbers. The process of resolving this challenge afforded me invaluable experience and offered practical solutions to this real-world issue. In the second environment, we endeavored to discern the relationship between scheduled and unscheduled patients and follow-up patients, aiming to plan resource allocation for these patients within the hospital setting. In tackling this challenge, we observed the performance disparities between backward induction and Q-learning when addressing large-scale problems, which offers valuable insights for addressing such issues in real-world settings. In summary, this memorable research journey deepened my understanding of various deep learning algorithms and inspired me to apply deep learning techniques to issues arising in different medical system environments.

References

[1] Garaix, T., Rostami, S., & Xie, X. (2018). Optimizing outpatient appointment system using machine learning algorithms and scheduling rules: A prescriptive analytics framework. Expert Systems with Applications, 102, 245-261.

[2] Chong, L.R., Tsai, K.T., Lee, L.L., Foo, S.G., & Chang, P.C. (2020). Artificial Intelligence Predictive Analytics in the Management of Outpatient MRI Appointment No-Shows. American Journal of Roentgenology, 215(5), 1155-1162.

DOI: 10.54254/2755-2721/79/20241669

[3] Babayoff, O., Shehory, O., Geller, S., Shitrit-Niselbaum, C., Weiss-Meilik, A., & Sprecher, E. (2022). Improving Hospital Outpatient Clinics Appointment Schedules by Prediction Models. Journal of Medical Systems, 47(1):5.

[4] S. Binato, G.C. de Oliveira, J.L. de Araujo. "A greedy randomized adaptive search procedure for transmission expansion planning" , IEEE Transactions on Power Systems, 2001

[5] Ershang Tian, Juntae Kim. "Improved Vehicle Detection Using Weather Classification and Faster R-CNN with Dark Channel Prior" , Electronics, 2023

[6] Garaix, T., Rostami, S., & Xie, X. (2023). Using machine learning techniques to reduce uncertainty for outpatient appointment scheduling practices in outpatient clinics. Operations Research for Health Care, 31, 100380.

[7] Garaix, T., Rostami, S., & Xie, X. (2022). Simulation modelling of hospital outpatient department: A review of the literature. Simulation Modelling Practice and Theory, 112, 102377.

[8] P. Zhang. CIS_L3_DP.pdf, page3.

[9] Garaix, T., Rostami, S., & Xie, X. (2019). Improving healthcare operations management with machine learning. Nature Machine Intelligence, 1(5), 261-267.

[10] Garaix, T., Rostami, S., & Xie, X. (2021). A simulation-based evaluation of machine learning models for clinical decision support: Application and analysis using hospital readmission. npj Digital Medicine, 4, 158.

[11] Philpott-Morgan, S., Thakrar, D.B., Symons, J., Ray, D., Ashrafian, H., & Darzi, A. (2021). Characterising the nationwide burden and predictors of unkept outpatient appointments in the National Health Service in England: A cohort study using a machine learning approach. PLoS Medicine, 18(10):e1003783.