CSCE 642: DEEP REINFORCEMENT LEARNING

PROJECT PROPOSAL

Personalized Healthcare Appointment Scheduling via Deep Reinforcement Learning

1. Team Members

Subhiksha Saravanasundaram
Gauri Pawar

2. Proposed Starting Point

The project will be developed entirely in Python, using an OpenAI Gym-style environment to simulate patient arrivals, appointment slots, and consultation durations. The environment will be implemented using NumPy and Pandas for queue management and reward computation. No external simulator is required. However, the implementation may draw structural inspiration from standard RL templates such as the gym.Env interface.

3. Project Overview and Objective

The objective of this project is to design and train a reinforcement learning (RL) agent capable of dynamically scheduling healthcare appointments to minimize patient waiting time and doctor idle time, thereby improving clinic efficiency. Traditional scheduling methods such as FIFO or fixed-time slots are often inadequate for handling real-world variability in patient severity, consultation duration, and no-show probability. By framing this as a Markov Decision Process (MDP), the RL agent will learn adaptive scheduling policies that optimize both patient and provider outcomes.

4. Problem Formulation

4.1. State Space (S): Represents key features such as patient queue length, doctor availability status, current time slot, estimated consultation duration, patient severity score, and predicted no-show probability. Together, these capture both operational efficiency and patient variability.

4.2. State Representation and Feature Computation:

4.2.1 Feature Estimation:

- Estimated Consultation Duration will be computed using a regression model trained on simulated or historical appointment data, accounting for patient type, severity, and visit category.
- Predicted No-Show Probability will be estimated using logistic regression or gradient-boosted models, based on features like prior attendance history, appointment lead time, and time of day.

4.2.2. Tabular Q-Learning Feasibility: To keep the state space tractable for the tabular Q-learning baseline, continuous variables will be discretized into bins such as:

- Queue length ∈ {0, 1–2, 3–5, 6+}
- Consultation duration ∈ {short, medium, long}
- No-show probability ∈ {low, medium, high}
- Time of day ∈ {morning, afternoon, evening}

This reduces the total number of possible states to a few thousand, ensuring computational feasibility and stable convergence.

4.2.3 Transition to DQN: For the Deep Q-Network phase, these same features will be used in continuous form, with numerical features normalized and categorical variables one-hot encoded. This allows the DQN to learn more granular scheduling patterns while building on the validated tabular setup.

4.3. Action Space (A): Consists of assigning a patient to one of the upcoming available time slots (e.g., next slot, +1 hour, +2 hours, or defer to next day).

4.4. Transition Function (P): Simulates the dynamics of new patient arrivals, consultation completions, and occasional cancellations, reflecting stochastic variability in clinic operations.

4.5. Reward Function (R): Rt = - (Waiting Time+ Idle Time) - α (No Show Penalty), where higher rewards are obtained when patient throughput is high and doctor idle time is low.

4.6. Episode Definition: Each episode represents a simulated clinic day or week of appointment scheduling decisions.

5. Algorithms and Approach

The project will proceed in two stages to ensure both interpretability and scalability of the reinforcement learning framework:

5.1 *Baseline - Tabular Q-Learning*: This stage will establish a baseline for improvement, providing insights into state–action correlations and reward sensitivity.

- The discretized state space will be used to keep the Q-table manageable and computationally efficient.
- The agent will learn to balance patient waiting time and doctor idle time through iterative exploration using the ε-greedy policy.

5.2 *Scalable Model - Deep Q-Network (DQN):* A neural network approximator will replace the Q-table, enabling generalization across unseen states.

- Techniques such as experience replay and target network updates will be incorporated to stabilize learning and prevent divergence.
- Hyperparameters including learning rate, discount factor (γ), and batch size will be tuned to ensure convergence.

6. Evaluation Metrics

The performance of the agent will be assessed using the following metrics:

- Both Q-learning and DQN models will be compared against heuristic scheduling policies such as First-In-First-Out (FIFO) and Shortest-Job-First (SJF).
- Average patient waiting time
- Doctor utilization rate (percentage of active consultation time)
- No-show and reschedule rates
- Cumulative episodic reward over training episodes

7. Expected Outcomes

The reinforcement learning agent is expected to achieve a 10–15% reduction in average waiting time and an increase in doctor utilization relative to baseline scheduling heuristics. The trained policy should generalize to different patient arrival distributions and clinic setups, demonstrating robust adaptability in dynamic scheduling contexts.

8. Stretch Goal

As an advanced extension, the project aims to develop a multi-doctor scheduling system using Multi-Agent Reinforcement Learning (MARL), allowing agents to coordinate across shared appointment resources. Another enhancement involves integrating realistic patient arrival patterns from publicly available datasets such as MIMIC-III to evaluate real-world performance.

9. Deliverables

- Python-based simulation environment for healthcare appointment scheduling
- Implementation of Q-learning and DQN algorithms
- Training and evaluation scripts with performance visualizations
- Comparative analysis between RL-based and heuristic scheduling approaches
- Final report documenting the MDP formulation, algorithm design, experimental results, and discussion