# Build an AI Agent for Medical Case Queue Management

Varanasi Satya Sreekanth
(varanasi_sreekanth@yahoo.com)
Phone:+91-9182495028

March 8, 2025

## 1 Introduction

This document describes the implementation of a reinforcement learning (RL) agent using Proximal Policy Optimization (PPO) for medical case assignment. The goal is to optimize efficiency, reduce misassignments, and ensure Service Level Agreement (SLA) compliance while dynamically adapting to changes in hospital policies. The learned AI agent will use Deep Reinforcement Learning (DRL), specifically Proximal Policy Optimization (PPO), to optimize medical case assignments. It will consider factors like doctor availability, case urgency, specialization requirements, and compliance with SLAs. The agent will also adapt to changes in hospital policies in real-time.

1. Reinforcement Learning: Uses PPO to optimize case assignments dynamically.
2. Doctor Availability: The model learns to prioritize available doctors.
3. Urgency  SLA Compliance: It optimizes assignments based on urgency and SLA rules.
4. Adaptability: Adjusts to evolving policies and real-time availability changes.

## 2 Methodology

### 2.1 Overview

PPO is a policy gradient method introduced by OpenAI that improves upon earlier algorithms like Trust Region Policy Optimization (TRPO). PPO is widely used due to its simplicity, efficiency, and robustness.

### 2.2 Need for Choosing PPO

Traditional policy gradient methods, like REINFORCE, suffer from high variance and instability. Trust Region Policy Optimization (TRPO) improved stability by enforcing a constraint on policy updates but was computationally expensive. PPO simplifies TRPO while maintaining stability by using a clipped surrogate objective function.

### 2.3 Mathematical Formulation of PPO

#### 2.3.1 Policy Gradient Theorem

Policy gradient methods optimize the policy by maximizing the expected reward:

$$J(\theta) = \mathbb{E}\tau \sim \pi\theta \left[ \sum_{t=0}^{T} r_t \right] \tag{1}$$

where is a trajectory sampled from the policy.

The policy is updated using the gradient of the objective function:

$$\nabla_\theta J(\theta) = \mathbb{E}\tau \sim \pi\theta \left[ \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A_t \right] \tag{2}$$

where is the advantage function.

### 2.3.2 PPO Objective Function

PPO introduces a clipped objective function to prevent large policy updates:

$$L(\theta) = \mathbb{E}_t \left[ \min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \right] \tag{3}$$

where is the probability ratio and is a small positive constant (e.g., 0.2).

### 2.3.3 Clipping Mechanism

The clip function ensures that updates are within a reasonable range, avoiding instability.

# 3 Medical Case Queue Management in View of Proximal Policy Optimization

The RL environment models the hospital system as a Markov Decision Process (MDP).

1. **Action Space:** The agent selects a doctor from a fixed number of available doctors:

$$\mathcal{A} = \{0, 1, \ldots, N_d - 1\} \tag{4}$$

   where $N_d$ is the number of doctors.

2. **Observation Space:** Each patient case is represented by a 9-dimensional feature vector:

$$s = [\text{age}, \text{gender}, \text{medical history}, \text{urgency}, \text{availability}, \text{SLA compliance}, \text{experience}, \text{ratings}, \text{priority}] \tag{5}$$

3. **Loading Patient Data** Patient details are loaded from a CSV file, ensuring scalability.

4. **Reward Function** The PPO agent maximizes a reward function based on assignment quality. The reward function includes:

   (a) Penalty for assigning unavailable doctors: $-10$

   (b) Penalty for exceeding max workload: $-5$

   (c) Matching medical history with experience:

$$R_{\text{match}} = 10 \times (1 - |\text{history} - \text{experience}|) \tag{6}$$

   (d) Reward for handling urgent cases: $+5 \times$ case urgency

   (e) Reward for SLA compliance: $+10 \times$ SLA compliance

   (f) Reward for assigning highly rated doctors: $+5 \times$ ratings

   (g) Penalty for repeated misassignments: $-2 \times$ misassignments

5. **Mapping Variables to PPO** PPO is a policy-gradient-based reinforcement learning algorithm. The mapping of RL variables to PPO is shown in Table 1.

| PPO Concept | Code Variables |
|---|---|
| State (Observation) | $s$ (9-dimensional feature vector) |
| Action Selection | $a \in \mathcal{A}$ (choosing a doctor) |
| Policy Network | $\pi_\theta(s)$ (learned policy) |
| Reward Function | $r$ (as defined above) |
| Advantage Estimation | PPO estimates $A(s, a)$ |
| Value Function (Critic) | $V_\theta(s)$ |
| Exploration-Exploitation | PPO clipping mechanism |

Table 1: Mapping between PPO concepts and RL environment variables.

# 4 Training the PPO Agent

The PPO agent is trained using Stable-Baselines3:

```
model = PPO("MlpPolicy", env, verbose=1)
model.learn(total_timesteps=10000)
```

# 5 Evaluation and Performance Metrics

Performance is measured using the following metrics:

- **Average Patient Wait Time** $\mathbb{E}[T_{\text{wait}}]$
- **Doctor Utilization Rate** $\frac{\text{assigned cases}}{\text{max cases per doctor}}$
- **Misassignment Rate** $\mathbb{E}[\text{misassignments}]$
- **SLA Compliance Rate** $\mathbb{E}[\text{SLA compliance}]$

# 6 References


1. Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal policy optimization algorithms.” arXiv preprint arXiv:1707.06347 (2017).
2. Gu, Yang, Yuhu Cheng, CL Philip Chen, and Xuesong Wang.“Proximal policy optimization with policy feedback.” IEEE Transactions on Systems, Man, and Cybernetics: Systems 52, no. 7 (2021): 4600-4610.
3. Zhang, Junwei, Zhenghao Zhang, Shuai Han, and Shuai Lü. “Proximal policy optimization via enhanced exploration efficiency.” Information Sciences 609 (2022): 750-765.
4. OpenAI’s PPO Implementation: https://spinningup.openai.com/en/latest/algorithms/ppo.html