International Journal of Computer Applications (0975 - 8887)
Volume 186 - No.4, January 2024

Optimization of Treatment Plans using Deep Reinforcement Learning with the Human-in-the-loop

M.A. El-dosuky
Computer Science Department, Faculty of Computers and Information, Mansoura University, Egypt
Computer Science Department, Arab East Colleges, Saudi Arabia

ABSTRACT

Human-Centered Artificial Intelligence (HCAI) is a philosophy that focuses on designing AI systems that prioritize human well-being and user experiences. Medical technologies driven by AI are developing quickly to provide useful solutions for clinical practice. Treatment plan optimization is a process that aims to improve the effectiveness and efficiency of a treatment plan for a specific medical condition. Combining Deep Reinforcement Learning (DRL) with human-in-the-loop (HITL) can optimize treatment plans by combining the expertise of human clinicians with deep reinforcement learning algorithms. This paper provides two approaches for treatment plan optimization with Proximal Policy Optimization (PPO) and Deep Q Learning (DQN).

General Terms

Computer Science, Artificial Intelligence

Keywords

Deep Reinforcement Learning, human-in-the-loop, Treatment Plans Optimization

1. INTRODUCTION

Human-Centered Artificial Intelligence (HCAI) is a philosophy that focuses on designing AI systems that prioritize human well-being and user experiences [1]. It aims to integrate AI technologies with human values, ethics, and user experiences, ensuring they align with human values and goals [2]. Key principles include user-centric design [3], human-AI collaboration [4], trust and transparency [5], and social impact [6]. Collaboration with AI systems fosters cooperation and leverages the complementary strengths of humans and machines.
Medical technologies driven by AI are developing quickly to provide useful solutions for clinical practice [7]. AI in medicine involves virtual and physical applications, with Machine Learning being the virtual component [8] and carebots being an example of the physical component [9].
Treatment plan optimization is a process that aims to improve the effectiveness and efficiency of a treatment plan for a specific medical condition. It involves a thorough patient assessment, evaluation of treatment options, use of decision support tools, personalized medicine, and follow-up and monitoring [10]. The goal is to optimize treatment outcomes, enhance quality of life, reduce disease burden, and optimize healthcare system resource utilization [11].

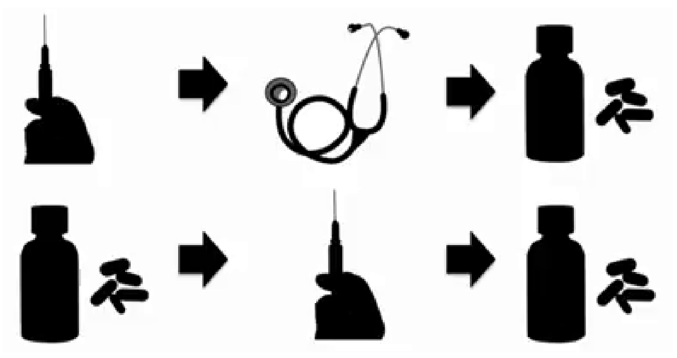
Fig. 1. Two Treatment Plans

Powerful systems, algorithms, and agents with amazing accomplishments have been developed as a result of combining Deep Learning and Reinforcement Learning, yielding Deep Reinforcement Learning (DRL) [12].
Human-in-the-loop (HITL) is a crucial area in AI research, as machine learning cannot replace human domain knowledge [13]. It aims to train accurate prediction models with minimal cost by integrating human knowledge and experience. Existing works are categorized into data processing, interventional training, and system-independent human-in-the-loop design.
DRL+HITL can optimize treatment plans by combining the expertise of human clinicians with deep reinforcement learning algorithms. HITL involves data collection and integration from various sources, policy initialization, reinforcement learning, and feedback from human experts. The agent learns from these inputs, generating personalized treatment recommendations based on factors like patient demographics, medical history, genetic information, and treatment outcomes. HITL allows for continuous improvement, updating policies based on new patient data and expert feedback. The optimized treatment plans can serve as decision support tools for clinicians, highlighting the most effective treatment options. By combining deep reinforcement learning algorithms with human clinicians' expertise, HITL can improve patient outcomes and healthcare delivery efficiency.

This paper provides two approaches for treatment plan optimization with Proximal Policy Optimization (PPO) and Deep Q Learning (DQN).
The structure of this paper beyond the introduction section is as follows. Section 2 provides the proposed methodology. Section 3 provides experiments and results. Section 4 concludes the paper, giving some possible future research directions.

2. PROPOSED METHODOLOGY

This section provides equations that are the basis for formulating DRL update process and the optimization objective in the context of treatment plan optimization using HITL. Then the section proposes an algorithm for optimizing treatment plans based on DRL and HITL.

2.1 Reinforcement Learning Update Equation

The reinforcement learning agent's policy can be updated using the Q-learning algorithm. The update equation for Q-learning is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \quad (1)$$

where: $Q(s,a)$ is the Q-value for state s and action a. α is the learning rate, determining the impact of new information on the agent's policy update. r is the immediate reward received after taking action a in state s. γ is the discount factor, balancing the importance of immediate and future rewards. s' is the next state after taking action a in state s. a' is the action chosen in the next state s' according to the agent's policy.

2.2 Treatment Plan Optimization Objective Function

The objective function for optimizing treatment plans can be defined as a combination of the expected rewards and costs associated with the treatment strategy. It can be formulated as:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right] - \lambda C(\pi) \quad (2)$$

where: $J(\pi)$ represents the objective function for policy π. $\mathbb{E}_\pi$ denotes the expectation over the states and actions visited under policy π. T is the time horizon. γ is the discount factor. $R(s_t, a_t)$ represents the reward obtained at time step t for state s_t and action a_t. $C(\pi)$ represents the cost associated with executing policy π. λ is a parameter that balances the trade-off between rewards and costs.

2.3 Proposed Algorithm

The first algorithm shows the construction of TreatmentPlanEnv class, which is the environment dedicated for treatment plan optimization. The algorithm utilizes gym Python package. The constructor of the class defines the action and observation spaces. Then, it Initializes the state and other variables. The step function takes action as input. It then updates the state based on the chosen action. After that it calculates the reward based on the state and action. Then it increments the current step. Finally, it checks if the episode is done or not.
The second algorithm utilizes PPO from stable_baselines3 Python package. It instantiates the treatment plan environment. Then it trains the DRL agent with multi-layer perceptron policy. Then it runs the treatment plan with human intervention. In an infinite loop until not done, the algorithm predicts the DRL agent action, get human expert input, takes a step in the environment, and output the reward.

Algorithm 1 Treatment Plan Environment

```
import gym
from gym import spaces

class TreatmentPlanEnv inherits gym.Env:
  Constructor:
    super(TreatmentPlanEnv, self)

    self.action_space ← spaces.Discrete(3)
    self.observation_space ← spaces.Box(low=0, high=1, shape=(4,), dtype=float32)

    self.state ← np.zeros((4,))
    self.current_step ← 0
    self.max_steps ← 10

  Function reset(self):
    self.state ← np.zeros((4,))
    self.current_step ← 0
    return self.state

  Function step(self, action):
    self.state[action] ← 1
    reward = self._calculate_reward()
    self.current_step ← self.current_step + 1
    done ← self.current_step >= self.max_steps
    return self.state, reward, done, {}

  Function _calculate_reward(self):
    reward ← np.sum(self.state)
    return reward
```

Algorithm 2 Treatment Plan Optimization with PPO

```
from stable_baselines3 import PPO

env ← TreatmentPlanEnv()

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

state ← env.reset()
done ← False

while not done:
  action, _ ← model.predict(state)
  print("DRL Agent suggests action:", action)

  human_action ← int(input("Enter action (0-2):"))

  action ← human_action

  state, reward, done, _ ← env.step(action)

  print("Reward:", reward)
  print("======================")
```

The third algorithm defines the treatment environment, the deep reinforcement learning agent, and the human expert. It also defines the learning parameters such as the episodes, maximum steps, epsilon, and learning rate. The algorithm iterates over episodes. It iterates over steps within each episode, applying exploration and exploitation steps. It updates the total reward and the epsilon. It prints the episode results. Finally it uses the trained agent to generate treatment plan recommendations.

3. EXPERIMENTS AND RESULTS

The implementation was done in Python 3.9.13, on HP Envy x360 laptop, with AMD Ryzen 7 processor, running Windows 11 Home 64-bit.
The following tables (Table 1, Table 2, Table 3, Table 4, and Table 5) trace the execution of the proposed methodology. The metrics are divided into three categories, namely rollout, time, and train. First, rollout has two metrics:

—**ep. len. mean**: mean episode length
—**ep. rew. mean**: mean episodic training reward

Second, time has four metrics:

—**fps**: frames per seconds, including gradient update time
—**iterations**: iterations number
—**time elapsed**: time elapsed from the start of training (in seconds)
—**total time steps**: total number of time steps

Third, train has ten metrics:

—**approx kl**: approximate average KL-divergence between new and old policy
—**clip fraction**: average fraction above clip range threshold
—**clip range**: PPO clipping factor
—**entropy loss**: average entropy loss
—**explained variance**: variance explained fraction
—**learning rate**: learning rate value
—**loss**: total loss
—**n updates**: number of gradient updates
—**policy gradient loss**: policy gradient loss
—**value loss**: error between output of the value function and Monte-Carlo estimation

Table 1.
Iteration
1

Metrics	rollout	ep len mean	10
		ep rew mean	24
	time	fps	932
		iterations	1
		time elapsed	2
		total timesteps	2048

There are two scenarios. The first scenario assumes that the human expert totally agrees with the action suggested by DRL agent as follows.

Algorithm 3 Treatment Plan Optimization with DQN

```
env ← gym.make('TreatmentPlanEnv')

agent ← DQNAgent(state_size=env.observation_space.shape[0],
action_size=env.action_space.n)

expert ← Expert()

episodes ← 1000
max_steps ← 100
epsilon ← 1.0
epsilon_decay ← 0.99
epsilon_min ← 0.01
learning_rate ← 0.001
gamma ← 0.99

for episode in range(episodes) do
    state ← env.reset()
    total_reward ← 0

    for step in range(max_steps) do
        if np.random.rand() ≤ epsilon then
            action ← env.action_space.sample()
        else
            action ← agent.act(state)
        end if

        next_state, reward, done, _ ← env.step(action)

        expert_feedback ← expert.get_feedback(state, action, reward, next_state, done)

        agent.update(state, action, reward, next_state, done, expert_feedback, learning_rate, gamma)

        state ← next_state
        total_reward ← total_reward + reward

        if done then
            break
        end if
    end for

    epsilon ← epsilon * epsilon_decay
    epsilon ← max(epsilon, epsilon_min)

    print("Episode:", episode + 1, "Reward:", total_reward)
end for

state ← env.reset()
done ← False
while not done do
    action ← agent.act(state)
    state, _, done, _ ← env.step(action)
end while

env.close()
```

Table 2. Iteration 2

Metrics	rollout	ep len mean	10
		ep rew mean	24.6
	time	fps	649
		iterations	2
		time elapsed	6
		total timesteps	4096
	train	approx kl	0.006527826
		clip fraction	0.0605
		clip range	0.2
		entropy loss	-1.1
		explained variance	0.0155
		learning rate	0.0003
		loss	18.4
		n updates	10
		policy gradient loss	-0.0104
		value loss	92.5

Table 3. Iteration 3

Metrics	rollout	ep len mean	10
		ep rew mean	25.4
	time	fps	603
		iterations	3
		time elapsed	10
		total timesteps	6144
	train	approx kl	0.0121835945
		clip fraction	0.101
		clip range	0.2
		entropy loss	-1.08
		explained variance	-0.267
		learning rate	0.0003
		loss	21.8
		n updates	20
		policy gradient loss	-0.00743
		value loss	51.5

Table 4. Iteration 4

Metrics	rollout	ep len mean	10
		ep rew mean	25.8
	time	fps	583
		iterations	4
		time elapsed	14
		total timesteps	8192
	train	approx kl	0.0076065226
		clip fraction	0.0736
		clip range	0.2
		entropy loss	-1.06
		explained variance	-0.0185
		learning rate	0.0003
		loss	31.7
		n updates	30
		policy gradient loss	-0.00457
		value loss	52.5

Table 5. Iteration 5

Metrics	rollout	ep len mean	10
		ep rew mean	26.2
	time	fps	570
		iterations	5
		time elapsed	17
		total timesteps	10240
	train	approx kl	0.011540119
		clip fraction	0.129
		clip range	0.2
		entropy loss	-1.04
		explained variance	-0.00387
		learning rate	0.0003
		loss	23.5
		n updates	40
		policy gradient loss	-0.00776
		value loss	54.8

```
DRL Agent suggests action: 0
Enter action (0-2):0
Reward: 1.0
============================
DRL Agent suggests action: 1
Enter action (0-2):1
Reward: 2.0
============================
DRL Agent suggests action: 1
Enter action (0-2):1
Reward: 2.0
============================
DRL Agent suggests action: 2
Enter action (0-2):2
Reward: 3.0
============================
DRL Agent suggests action: 2
Enter action (0-2):2
Reward: 3.0
============================
DRL Agent suggests action: 0
Enter action (0-2):0
Reward: 3.0
============================
DRL Agent suggests action: 2
Enter action (0-2):2
Reward: 3.0
============================
DRL Agent suggests action: 2
Enter action (0-2):2
Reward: 3.0
============================
DRL Agent suggests action: 0
Enter action (0-2):0
Reward: 3.0
============================
DRL Agent suggests action: 1
Enter action (0-2):1
Reward: 3.0
============================
```

The second scenario goes to the other extreme assuming that the human expert totally disagrees with the action suggested by DRL agent as follows.

```
DRL Agent suggests action: 2
Enter action (0-2):0
Reward: 1.0
============================
DRL Agent suggests action: 1
Enter action (0-2):0
Reward: 1.0
============================
DRL Agent suggests action: 1
Enter action (0-2):0
Reward: 1.0
============================
DRL Agent suggests action: 1
Enter action (0-2):0
Reward: 1.0
============================
DRL Agent suggests action: 1
Enter action (0-2):0
Reward: 1.0
============================
DRL Agent suggests action: 0
Enter action (0-2):1
Reward: 2.0
============================
DRL Agent suggests action: 2
Enter action (0-2):1
Reward: 2.0
============================
DRL Agent suggests action: 2
Enter action (0-2):1
Reward: 2.0
============================
DRL Agent suggests action: 1
Enter action (0-2):0
Reward: 2.0
============================
DRL Agent suggests action: 1
Enter action (0-2):0
Reward: 2.0
============================
```

4. CONCLUSION AND FUTURE WORK

This paper provides two approaches for treatment plan optimization with Proximal Policy Optimization (PPO) and Deep Q Learning (DQN). The paper traces the execution of the proposed algorithms. Many future directions are possible. One possible direction is generating personalized treatment recommendations for individual patients by considering factors like demographics, medical history, genetic information, and treatment outcomes. Another possible direction is to consider the optimized treatment plans as decision support tools, providing clinicians with recommendations on the most effective treatment options based on learned policies.

5. REFERENCES

References

[1] Bingley, W., Curtis, C., Lockey, S., Bialkowski, A., Gillespie, N., Haslam, S., Ko, R., Steffens, N., Wiles, J. & Worthy, P. Where is the human in human-centered AI? Insights from developer priorities and user experiences. *Computers In Human Behavior*. **141** pp. 107617 (2023)

[2] Gabriel, I. Artificial intelligence, values, and alignment. *Minds And Machines*. **30**, 411-437 (2020)

[3] Bu, L., Chen, C., Ng, K., Zheng, P., Dong, G. & Liu, H. A user-centric design approach for smart product-service systems using virtual reality: A case study. *Journal Of Cleaner Production*. **280** pp. 124413 (2021)

[4] Wang, D., Churchill, E., Maes, P., Fan, X., Shneiderman, B., Shi, Y. & Wang, Q. From human-human collaboration to Human-AI collaboration: Designing AI systems that can work together with people. *Extended Abstracts Of The 2020 CHI Conference On Human Factors In Computing Systems*. pp. 1-6 (2020)

[5] Schmidt, P., Biessmann, F. & Teubner, T. Transparency and trust in artificial intelligence systems. *Journal Of Decision Systems*. **29**, 260-278 (2020)

[6] Tomašev, N., Cornebise, J., Hutter, F., Mohamed, S., Picciariello, A., Connelly, B., Belgrave, D., Ezer, D., Haert, F., Mugisha, F. & Others AI for social good: unlocking the opportunity for positive impact. *Nature Communications*. **11**, 2468 (2020)

[7] Briganti, G. & Le Moine, O. Artificial intelligence in medicine: today and tomorrow. *Frontiers In Medicine*. **7** pp. 27 (2020)

[8] Hamet, P. & Tremblay, J. Artificial intelligence in medicine. *Metabolism*. **69** pp. S36-S40 (2017)

[9] Cornet, G. Robot companions and ethics: A pragmatic approach of ethical design. *Journal International De Bioéthique*. **24**, 49-58 (2013)

[10] Trofimov, A., Craft, D. & Unkelbach, J. Treatment-planning optimization. *Proton Therapy Physics. Series In Medical Physics And Biomedical Engineering. Boca Raton, FL: CRC Press/Taylor & Francis*. pp. 1 (2012)

[11] Wedenberg, M., Beltran, C., Mairani, A. & Alber, M. Advanced treatment planning. *Medical Physics*. **45**, e1011-e1023 (2018)

[12] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M., Pineau, J. & Others An introduction to deep reinforcement learning. *Foundations And Trends In Machine Learning*. **11**, 219-354 (2018)

[13] Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T. & He, L. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*. **135** pp. 364-381 (2022)