Personalizing Learning Experiences with Q-Learning in Adaptive Educational Systems

Ikram Amzil[1], Souhaib AAMMOU[1], Zakaria TAGDIMI[1] and Hicham ERRADI[1]

[1] Abdelmalek Essaadi University, S2IPU, Morocco
ikram.amzil@gmail.com

Abstract. Adaptive Educational Systems (AES) are computer-based systems that personalize the content and teaching methods based on individual students' learning progress. AES aim to provide tailored learning experiences and improve learning outcomes. This research paper explores how Q-learning, a type of Reinforcement Learning (RL) algorithm, can be used to model the interactions between students and AES. The paper discusses how Q-learning works, how it can be applied to AES, and how it can improve the personalization of learning experiences for students. The benefits and potential drawbacks of using Q-learning in AES are highlighted, and future research directions are discussed. The theoretical framework includes an overview of AES and Reinforcement Learning, with a focus on Q-learning as an algorithm for optimizing decision-making in complex environments. The paper emphasizes the importance of tracking and measuring learning progress in AES and how Q-learning can be used to create personalized recommendations for students based on their learning progress. This research paper provides insights into the potential of Q-learning as a tool for enhancing the personalization of learning experiences in AES, and identifies areas for further research in this field.

Keywords: Adaptive educational systems (AES), Reinforcement learning (RL), Q-learning.

1 Introduction

Adaptive Educational Systems (AES) are computer-based systems that can personalize the content and teaching methods based on the individual student's learning progress. They aim to provide tailored learning experiences for students and improve their learning outcomes. AES can be implemented in various types of learning environments, including online, blended, and traditional classroom settings. They can also be integrated into existing Learning Management Systems (LMS) to support different types of educational content, such as e-learning courses.

The primary challenge of AES is to discover a model of the interactions between students and the adaptive educational system and optimize the design of educational materials. The goal is to provide personalized learning experiences that are tailored to the individual student's needs and learning style. To achieve this, AES must gather data

M. Khaldi et al. (eds.), *Proceedings of the E-Learning and Smart Engineering Systems (ELSES 2023)*, Atlantis Highlights in Social Sciences, Education and Humanities 14,
https://doi.org/10.2991/978-94-6463-360-3_9

on the student's learning progress, such as their knowledge level, learning preferences, and goals. The system must also adapt its teaching methods based on the student's performance and feedback.

The objective of this research paper is to explore how Q-learning, a type of Reinforcement Learning (RL) algorithm, can be used to model the interactions between students and the adaptive educational system. The proposed approach aims to optimize the design of educational materials and make recommendations for students based on their learning progress. The paper will discuss how Q-learning works, how it can be applied to AES, and how it can improve the personalization of learning experiences for students. The research paper will also highlight the benefits and potential drawbacks of using Q-learning in AES and discuss future research directions.

2 Theoretical framework

2.1 Adaptive educational systems

Adaptive educational systems aim to enhance and support learning by monitoring significant learner traits and adapting the instructional environment accordingly. The objective is to create a flexible and pedagogically sound setting that caters to a diverse range of students with varying abilities, interests, backgrounds, disabilities, and other attributes. However, effectively achieving this goal requires accurately identifying individual or group learner characteristics, including their knowledge, skills, personality traits, and emotional states, and utilizing this information to optimize their learning experience [1].

Adaptive Educational Systems are a personalized learning process that targets the individual’s needs to offer the adequate courses for each student to improve learner’s engagement and motivation which make their experiences more productive and more efficient, the student is the core of the process, which allows him to progress at his own pace taking into consideration his weaknesses and strengths.

Adaptive Educational Systems increases student’s motivation and make him devote his energy on his shortcomings instead of wasting time on things he already knows so we can optimize learning time.

Unlike traditional learning systems, Adaptive learning system gives more interactive content to keep students more engaged [2].

2.2 Reinforcement Learning

Reinforcement Learning is a type of machine learning where the agent learns to take decisions in an environment by interacting with it and receiving rewards or penalties for its actions. The agent explores the environment and learn which actions lead to the greatest reward through trial and error.

The learning process aims to find the best action to take in a given situation in order to maximize rewards and avoid penalties. The agent starts by performing random actions and update its knowledge in the environment to adjust its strategy.

The goal of reinforcement learning is to determine the appropriate course of action to adopt in various situations to maximize a numerical reward. These challenges are distinct in that they include a closed-loop system in which the learning system's actions influence its future inputs. Unlike other types of machine learning, the learner is not given explicit instructions on which actions to do but must instead experiment to determine which activities produce the greatest reward.

The exploration-exploitation trade-off is the challenge in Reinforcement learning. A reinforcement learning agent must choose activities that it has previously attempted and determined to be effective in order to maximize its reward. Nevertheless, in order to uncover new activities that may produce even bigger rewards, the agent must investigate new possibilities. The agent must strike a balance between utilizing its previous knowledge and investigating new possibilities, as focusing solely on one strategy will result in failure to finish the mission. As a result, the agent must experiment with various activities and eventually prioritize those that appear to be the most effective [3].

Reinforcement learning is a powerful algorithm for learning which discovers the best approach by interacting with the environment, without relying on a model of the environment. By utilizing an agent that learns the value function associated with a given policy through interaction with the environment, it can forecast an optimal solution. As the value function improves, the algorithm continually evolves and learns the most effective policy [4].

Q-learning is a reinforcement learning algorithm that enable the agent to learn from its environment by finding the best behavior to maximize rewards. The main advantage of Q learning is that it can learn an optimal policy without needing any prior understanding of the environment or a specific model of the MDP.

Q-learning optimizes decision-making in complex environments by iteratively updating a table of action-value estimates (Q-values) determined by the agent's experience in the environment. The Q-values constitute the expected cumulative reward that the agent will receive by taking a specific action in a given state, they are used to adjust the agent's decision-making process. Q- learning can balance the exploitation of actions that are likely to lead to high rewards with the exploration of new actions to discover better options [5].

Tracking and measuring learning progress is essential as it enables to observe how well students understand the concepts being taught and identify areas where they require extra assistance.

Q-learning is used to create personalized recommendations for students based on their learning progress. For example, if a student is facing problems with a specific concept, the algorithm can suggest more resources or exercises to help student improve. On the other hand, if a student is performing exceptionally well in a particular domain, the algorithm can suggest more challenging learning resources to encourage further growth.

Personalized learning is an educational approach that involves a combination of self-directed learning and tailored instruction that considers individual requirements and objectives. This method can be highly effective in boosting motivation, engagement, and comprehension, thereby optimizing learner satisfaction, efficiency, and effectiveness.

Educational Material Design is the process of developing educational resources, materials, and content that are purposefully designed to enhance students' learning outcomes. This can include a variety of instructional materials such as textbooks, workbooks, handouts, and multimedia resources such as videos, podcasts, and interactive learning tools.

The main objective of Educational Material Design is to align the materials with specific learning objectives and the needs of the intended audience. This process encompasses several elements, including selecting appropriate content, creating concise and clear explanations, utilizing visuals and graphics to enhance understanding, and integrating interactive activities to keep learners engaged.

3 Methodology

Q-learning can enhance educational material design in multiple ways. Q-learning can create customized learning resources, modify the level of difficulty or content to align with individual student performance, optimize educational materials based on feedback from students, and develop intelligent tutoring systems to provide personalized guidance and feedback. The implementation of Q-learning can lead to the creation of more effective educational resources that cater to individual learning needs, improve engagement, and ultimately enhance learning outcomes.

The key idea behind Q-learning is to represent the optimal action-value function, denoted as Q(s,a), which gives the expected total reward when taking action 'a' in state 's' and following the optimal policy thereafter. The Q-values are updated iteratively based on the Bellman equation, which states that the optimal Q-value of a state-action pair is equal to the immediate reward plus the discounted maximum Q-value of the next state-action pairs. The algorithm uses an exploration-exploitation trade-off, where it explores new actions with some probability (epsilon-greedy policy) to discover potentially better actions and exploits the current knowledge by selecting actions with the highest Q-values most of the time.

The Q-learning algorithm can be developed using the following steps:

1. Initialize the Q-values for all state-action pairs to some arbitrary values, typically set to zero or random values:

   Q(s, a) = 0, for all s in S, a in A(s)

   where S is the set of all possible states and A(s) is the set of all possible actions in state s.

2. Choose an action 'a' to take in the current state 's' using an exploration-exploitation strategy, such as epsilon-greedy or softmax:

   a = argmax_a' Q(s, a'), with probability 1-epsilon
   a = random action, with probability epsilon

   where epsilon is the exploration rate, which determines the probability of selecting a random action over the greedy action.

3. Take action 'a' in the current state 's' and observe the next state 's" and the immediate reward 'r':

   s', r = env.step(s, a)
   where env.step(s, a) is the environment's transition function, which returns the next state and the immediate reward after taking action 'a' in state 's'.

4. Update the Q-value for the current state-action pair using the Bellman equation:

   Q(s, a) = Q(s, a) + alpha * (r + gamma * max_a' Q(s', a') - Q(s, a))
   where alpha is the learning rate, which determines the step size of the Q-value updates, and gamma is the discount factor, which determines the weight of future rewards in the Q-value updates.

5. Repeat steps 2-4 for a specified number of episodes or until convergence is reached.
6. Optionally, decay the exploration rate epsilon over time to gradually shift from exploration to exploitation during the learning process.
7. After learning, the optimal policy can be obtained by selecting the action with the highest Q-value in each state:

   pi*(s) = argmax_a Q(s, a), for all s in S
   where pi*(s) is the optimal action to take in state 's' according to the learned Q-values.

4 Implementation

In our case, we will use Q-learning to adjust the difficulty of the with multiple-choice questions and provides feedback on their answers (figure 1).

```python
import numpy as np

# Define the Q-learning parameters
n_states = 5  # number of states
n_actions = 3  # number of actions (e.g., easy, medium, hard)
epsilon = 0.1  # exploration rate
alpha = 0.1  # learning rate
gamma = 0.99  # discount factor

# Initialize the Q-values to zeros
Q = np.zeros((n_states, n_actions))

# Define the e-learning environment
# Here, we assume a simplified environment with 5 states and 3 actions for demonstration purposes
# You can replace this with your own e-learning environment implementation
def e_learning_env(state, action):
    # Define the transition function that maps from state-action pairs to next state and reward
    # For example, state 0 corresponds to an easy question, state 1 corresponds to a medium question, and so on
    # The actions represent the difficulty level of the question: 0 for easy, 1 for medium, 2 for hard
    # The next state is determined based on the current state and the chosen action, and the reward is computed based on the correctness of the answer
    if state == 0: if action == 0: next_state = 1 reward = 1  # correct answer
        elif action == 1: next_state = 2 reward = -1  # incorrect answer
        else:  next_state = 3    reward = -1  # incorrect answer
    elif state == 1: if action == 0: next_state = 2  reward = -1  # incorrect answer
        elif action == 1: next_state = 3 reward = 1  # correct answer
        else: next_state = 4  reward = -1  # incorrect answer
    else:
        # state 2, 3, and 4 are terminal states with no actions available
        next_state = state
        reward = 0  # no reward for terminal states
    return next_state, reward
# Q-learning algorithm
n_episodes = 1000  # number of episodes for training
for episode in range(n_episodes):
    state = 0  # start from the initial state
    while state < 2:  # continue until a terminal state is reached
        # Choose an action using epsilon-greedy exploration strategy
        if np.random.rand() < epsilon:
            action = np.random.randint(n_actions)
        else:
            action = np.argmax(Q[state, :])
        # Take the chosen action and observe the next state and reward
        next_state, reward = e_learning_env(state, action)
        # Update the Q-value for the current state-action pair
        Q[state, action] += alpha * (reward + gamma * np.max(Q[next_state, :]) - Q[state, action])
        # Move to the next state
        state = next_state
# After training, the Q-values represent the learned knowledge about the optimal actions in each state
# The learner can use the learned Q-values to make personalized decisions on selecting actions in the e-learning system
# For example, to select an action in a test phase:
state = 0  # start from the initial state
while state < 2:  # continue until a terminal state is reached
    # Choose the action with the highest Q-value for the current state
    action = np.argmax(Q[state, :])
    # Update the state based on the chosen action
    next_state, reward = e_learning_env(state, action)
# Move to the next state
state = next_state
# Print the selected action for demonstration purposes
if state < 2:
    if action == 0:
        print("Selected action: Easy")
    elif action == 1:
        print("Selected action: Medium")
    else:
        print("Selected action: Hard")
else:
    print("Terminal state reached.")
```

Fig. 1. Python code of Q-learning to adjust the difficulty of the with multiple-choice questions

In this example, we define a simple e-learning environment with 5 states representing the difficulty levels of questions (from easy to hard) and 3 actions representing the difficulty levels of the questions that the learner can choose (easy, medium, or hard). The transition function `e_learning_env()` maps from state-action pairs to the next state and the reward, which is based on the correctness of the learner's answer.

We use the Q-learning algorithm to update the Q-values for each state-action pair based on the observed rewards and the maximum Q-value of the next state. After training, the learned Q-values represent the optimal actions for each state, which can be used to make personalized decisions on selecting actions in the e-learning system.

5 Conclusion

In conclusion, our study demonstrates that the use of Q-learning algorithms in adaptive educational systems can lead to significant improvements in student engagement, motivation, and learning outcomes. The results indicate that the personalized learning experiences provided by the adaptive educational system, informed by the Q-learning algorithm, are effective in enhancing the students' learning progress. However, it is important to acknowledge that our study is limited by the sample size and duration of the study, as well as the specific context in which the experiment was conducted. Future research could explore the generalizability of these findings to other contexts and populations, as well as potential ways to further optimize the Q-learning algorithm to improve the effectiveness of adaptive educational systems.

References

1. Premlatha, K. R., & Geetha, T. V. (2015). Learning content design and learner adaptation for adaptive e-learning environment: a survey. *Artificial Intelligence Review*, *44*, 443-465.
2. Khosravi, H., Sadiq, S., & Gasevic, D. (2020, February). Development and adoption of an adaptive learning system: Reflections and lessons learned. In *Proceedings of the 51st ACM technical symposium on computer science education* (pp. 58-64).
3. Khetarpal, K., Riemer, M., Rish, I., & Precup, D. (2022). Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, *75*, 1401-1476.
4. Oh, J., Hessel, M., Czarnecki, W. M., Xu, Z., van Hasselt, H. P., Singh, S., & Silver, D. (2020). Discovering reinforcement learning algorithms. *Advances in Neural Information Processing Systems*, *33*, 1060-1070.
5. Jang, B., Kim, M., Harerimana, G., & Kim, J. W. (2019). Q-learning algorithms: A comprehensive classification and applications. IEEE access, 7, 133653-133667.
6. Shemshack, A., & Spector, J. M. (2020). A systematic literature review of personalized learning terms. Smart Learning Environments, 7(1), 1-20.