

CS-499 Computer Science Capstone

Module 3-2 Milestone Two: Enhancement One: Software Design and Engineering

Southern New Hampshire University

Stephen Owusu-Agyekum

March 20, 2024

Purpose

This narrative supports the artifact under the Software Engineering and Design category and justifies its inclusion in my ePortfolio. It highlights the learning process I went through to create the artifact and considers the approach utilized.

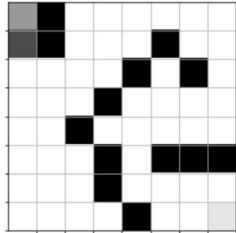
Prompt

The artifact I chose for this enhancement is “Treasure Hunt Game: Human Brain and Artificial Neural Network” from the CS-370 course. I took this course last year, 2023, around the May-June term. It is about creating an algorithm model to train an intelligent agent (Pirate) to hunt for a hidden treasure. I created deep Q-learning algorithms to build a model that trains agents to move through the maze to find the treasure. When the code is run, it will start the pirate at the top-left corner and play the game. Some blocks on the maze can prevent the pirate or the agent from moving through. The treasure is located in the bottom-right corner, and the agent should find a path from the starting position to the treasure.

I selected this artifact to include in my ePortfolio because it includes many features and characteristics of the Software Engineering and Design process. Developing the maze navigation path involves various software design and engineering aspects, including problem-solving, system architecture, modularity, testing, optimization, user experience design, and documentation. Also, the code provided achieves a well-structured, maintainable, and efficient implementation of a Q-learning-based maze navigation system. These principles contribute to developing high-quality software systems that are flexible, robust, and easy to understand and maintain.

```
[6]: qmaze = TreasureMaze(maze)
      canvas, reward, game_over = qmaze.act(DOWN)
      print("reward=", reward)
      show(qmaze)

reward= -0.04
[6]: <matplotlib.image.AxesImage at 0x1cefa992b08>
```



This function simulates a full game based on the provided trained model. The other parameters include the TreasureMaze object and the starting position of the pirate.

I improved the artifact through the following ways.

- The specific part of the artifact I enhanced is the “Q-Training Algorithm Code Block.”
- I added inline comments that explain the purpose (intent) and decision of each line or block of code that provide insight into the functionality of specific code segments.
- I improved the variable naming style to make it more descriptive and intuitive and to help understand the code without excessive commenting. For instance, I updated variable names like "**max_memory**," "**data_size**," and "**win_rate**" to make it self-explanatory, reducing the need for additional comments.
- I refactored the code by breaking it into smaller, more manageable functions to improve its modularity.
- I updated the deep Q-learning algorithm to make the Qtrain model run effectively.
- I updated the '**epoch**'s size to 1000, '**data_size**' to 32, and the '**max_memory**' to be eight times the maze size.
- We introduced parameters like '**opt**' in the Qtrain function to allow customization of training parameters, such as the number of epochs, maximum memory size, and data size

for training. This will provide flexibility in adapting the training process to different scenarios and problem settings.

#TODO: Complete the Q-Training Algorithm Code Block

This is your deep Q-learning implementation. The goal of your deep Q-learning implementation is to find the best possible navigation sequence that results in reaching the treasure cell while maximizing the reward. In your implementation, you need to determine the optimal number of epochs to achieve a 100% win rate.

You will need to complete the section starting with #pseudocode. The pseudocode has been included for you.

```
[1]: # Author: Stephen Owusu-Agyekum
# Date: 2024-04-15
# Version: V.2.1.0
# Description: This function implements the Q-training algorithm to train a model using reinforcement Learning.

def qtrain(model, maze, **opt):
    # Exploration factor for epsilon-greedy strategy
    # This is to clarify the purpose of the global variable 'epsilon' for adjusting exploration factor.
    # And document the significance of 'epsilon' in controlling the exploration factor for the agent.

    global epsilon

    # Number of epochs for training
    # To specify the default number of training epochs if not provided.
    # And provide a default value to ensure the training runs with a reasonable number of epochs.

    n_epoch = opt.get('n_epoch', 15000)

    # Maximum memory capacity to store experiences
    # Set the default maximum memory capacity for storing experiences if not provided.
    # Define a default value to ensure the model has enough memory to store experiences during training.

    max_memory = opt.get('max_memory', 1000)
```

```
# This is to initialize the win rate variable for monitoring training progress.
# This is to document the initialization of the 'win_rate' variable.

win_rate = 0.0

# Iterate over each epoch for training
for epoch in range(n_epoch):
    # Reset epoch-specific variables
    # To prepare epoch-specific variables for each training iteration.
    # To document the resetting of variables to their initial state at the beginning of each epoch.

    loss = 0.0
    game_over = False
    n_episodes = 0

    # Randomly select a free cell as the starting point for the agent
    # This is to choose a random starting position for the agent within the maze.
    # And to clarify the action of randomly selecting the agent's starting position.

    Agent_cell = random.choice(qmaze.free_cells)

    # Reset the maze with the agent positioned at the chosen cell
    # To reset the environment for a new epoch with the agent at the chosen starting position.
    # To document the action of resetting the maze environment.

    qmaze.reset(Agent_cell)

    # Obtain the current state of the environment (envstate)
    # Capture the current state of the environment.
    # Document the action of obtaining the environment state for the agent.

    envstate = qmaze.observe()
```

Course Outcome

I selected two-course outcomes or objectives during module one, which I planned to meet during my enhancement, and so far, I have met the two goals. I want to update the course objectives I selected with an additional two I have achieved. The initial outcomes I planned to meet were:

- “Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.” This outcome is achieved through the enhancement of the code comment. The addition of clear and concise comments throughout the codebase ensures that the purpose, functionality, and decision-making processes are well-documented. This makes the code coherent and accessible to other developers or stakeholders, meeting the outcome of professional-quality communications. Also, I ensured to provide detailed feedback on the training progress, including epoch details, loss, win count, and win rate, which enhances communication about the model's performance and convergence. Stakeholders can easily understand the progress and outcomes of the training process.
- “Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.” I achieved this outcome through code enhancement by implementing the experience replay mechanism through the GameExperience class demonstrates the use of a well-founded technique in reinforcement learning. This enhances the efficiency of model learning, showcasing proficiency in applying established techniques. Also, the dynamic adjustment of the exploration factor (epsilon) based on the win rate showcases an innovative approach to balancing exploration

and exploitation in reinforcement learning. This demonstrates the ability to apply creative solutions to solve complex problems effectively.

The two additional outcomes I have also met are as follows:

- “Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.” This objective is achieved through the enhancement by utilizing descriptive variable names such as `n_epoch`, `max_memory`, and `win_rate`, alongside detailed documentation, fosters a collaborative environment by making the codebase easily understandable and modifiable by multiple developers. This helps promotes collaboration by reducing ambiguity and enhancing code readability. Also, structuring the code into separate modules with well-defined responsibilities such as “`TreasureMaze.py`”, “`GameExperience.py`” and “`qtrain Function`” enables multiple developers to work on different components simultaneously without interference. This promotes collaboration and facilitates efficient development and maintenance of the codebase.
- “Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.” I achieved this objective through the enhancement of implementing the Q-learning algorithm in the `qtrain` function which adheres to the established algorithmic principles and standards in reinforcement learning. This demonstrates my proficiency in designing computing solutions based on rigorous algorithmic principles. Also, monitoring metrics such as loss, win rate, which were provided in the enhanced code, and completion criteria during training evaluates the effectiveness and convergence of the machine learning model. This adheres to established

standards for assessing model performance and ensures the reliability of the computing solution.

In the next code block, you will build your model and train it using deep Q-learning. Note: This step takes several minutes to fully run.

```
[12]: model = build_model(maze)
      qtrain(model, maze, epochs=1000, max_memory=8*maze.size, data_size=32)
```

Epoch: 109/14999	Loss: 0.0002	Episodes: 27	Win count: 105	Win rate: 1.000	time: 19.61 minutes
Epoch: 110/14999	Loss: 0.0003	Episodes: 11	Win count: 106	Win rate: 1.000	time: 19.65 minutes
Epoch: 111/14999	Loss: 0.0008	Episodes: 35	Win count: 107	Win rate: 1.000	time: 19.76 minutes
Epoch: 112/14999	Loss: 0.0008	Episodes: 14	Win count: 108	Win rate: 1.000	time: 19.80 minutes
Epoch: 113/14999	Loss: 0.0000	Episodes: 16	Win count: 109	Win rate: 1.000	time: 19.85 minutes
Epoch: 114/14999	Loss: 0.0003	Episodes: 43	Win count: 110	Win rate: 1.000	time: 19.97 minutes
Epoch: 115/14999	Loss: 0.0001	Episodes: 37	Win count: 111	Win rate: 1.000	time: 20.07 minutes
Epoch: 116/14999	Loss: 0.0001	Episodes: 31	Win count: 112	Win rate: 1.000	time: 20.16 minutes
Epoch: 117/14999	Loss: 0.0006	Episodes: 31	Win count: 113	Win rate: 1.000	time: 20.25 minutes
Epoch: 118/14999	Loss: 0.0003	Episodes: 28	Win count: 114	Win rate: 1.000	time: 20.33 minutes
Epoch: 119/14999	Loss: 0.0000	Episodes: 25	Win count: 115	Win rate: 1.000	time: 20.40 minutes
Epoch: 120/14999	Loss: 0.0005	Episodes: 23	Win count: 116	Win rate: 1.000	time: 20.47 minutes
Epoch: 121/14999	Loss: 0.0001	Episodes: 31	Win count: 117	Win rate: 1.000	time: 20.57 minutes
Epoch: 122/14999	Loss: 0.0003	Episodes: 18	Win count: 118	Win rate: 1.000	time: 20.62 minutes

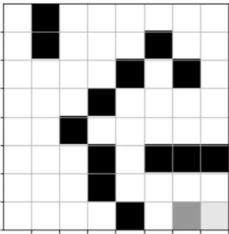
Reached 100% win rate at epoch: 122
n_epoch: 122, max_mem: 512, data: 32, time: 20.63 minutes

```
[12]: 1238.070556
```

This cell will check to see if the model passes the completion check. Note: This could take several minutes.

```
[14]: completion_check(model, qmaze)
      show(qmaze)
```


[14]: <matplotlib.image.AxesImage at 0x1ce84d99a48>



This cell will test your model for one game. It will start the pirate at the top-left corner and run `play_game`. The agent should find a path from the starting position to the target (treasure). The treasure is located in the bottom-right corner.

```
[16]: pirate_start = (0, 0)
      play_game(model, qmaze, pirate_start)
      show(qmaze)
```

[16]: <matplotlib.image.AxesImage at 0x1ce845dec08>



I updated the course objective to meet this enhancement by adding these two outcomes because, as a computer science student and upcoming IT Professional, I can employ these strategies to foster effective communication among diverse audiences. These include active listening, clear articulation of ideas, and creating opportunities for open dialogue. I could utilize collaborative platforms such as GitHub, where I host my ePortfolio, Slack, or Microsoft Teams to facilitate

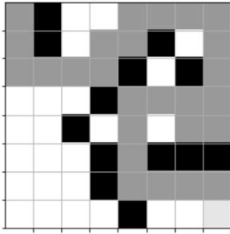
communication and information sharing. I have also learned to solve problems by thoroughly analyzing the problem statement to understand its requirements, constraints, and objectives. The enhancement has helped me adhere to the established industrial practices, standards, and guidelines throughout the design and implementation process, as well as best practices to maintain code quality and interoperability.

By implementing these objectives, I will create collaborative work environments using team members' varied expertise to aid in organizational decision-making. I can also create and assess computer solutions that tackle complicated issues while skillfully balancing trade-offs and adhering to industry standards and best practices. These skills are critical for success in the IT industry and are widely sought after by companies looking for ground-breaking and significant solutions. Through this enhancement, I have improved my written and oral communication skills. I can deliver clear and concise oral presentations that communicate technical concepts to diverse audiences, such as IT professionals and employers. Through the feedback I receive, I have enhanced my written communication skills, including producing coherent and well-organized technical documents, and paying grammar, spelling, and formatting to ensure professionalism and clarity.

```
This cell will test your model for one game. It will start the pirate at the top-left corner and run play_game. The agent should find a path from the starting position to the target (treasure). The treasure is located in the bottom-right corner.
```

```
[16]: pirate_start = (0, 0)
      play_game(model, qmaze, pirate_start)
      show(qmaze)
```

```
[16]: <matplotlib.image.AxesImage at 0x1ce845dec08>
```



Reflecting on enhancing and modifying the provided artifact involves considering various aspects of the development process. I started the process by evaluating the existing code to identify areas for enhancement or modification. This involved reviewing the code's functionality, structure, and performance to determine where improvements could be made. I checked the requirements and what we must achieve to create additional algorithms using deep Q-learning to model and train the intelligent agent. I implemented the planned changes to the artifact by writing new code, modifying existing code, and refactoring it to improve its structure and organization. During the implementation, I followed the best practices, adhered to coding standards, and maintained compatibility with the existing functionality.

The challenge I faced was when I installed the Python variables to install the Jupyter Notebook. I kept making errors that prevented me from completing the Notebook installation. It took me several hours to find out the cause of the problem. I had to install Anaconda Navigator, a desktop graphical user interface (GUI) with various applications, such as Jupyter Notebook, JupyterLab, Spyder, RStudio, and other tools and packages. Through Anaconda, I could use the Jupyter Notebook to start working on my artifact. Also, I encountered some errors within my code after updating the existing code and found fixing it a bit challenging. It took me more time to be able to fix it.

The enhancements helped me understand the process of reinforcement learning. Working with the Q-learning algorithm gave me valuable insights into reinforcement learning concepts. I better understood how agents navigate their environment and decide how best to maximize rewards. I learned to apply design patterns, such as the experience replay memory in the GameExperience class, to enhance the efficiency and effectiveness of the code.

Creating and improving this artifact gave me valuable hands-on experience in reinforcement learning, software design, testing, and collaboration. These learnings enhanced my technical skills and fostered a deeper appreciation for the iterative nature of software development and the importance of clear communication, modular design, and continuous improvement. Also, working with reinforcement learning algorithms like Q-learning requires algorithmic thinking and problem-solving skills and experience, which would enhance my ability to analyze problems, devise efficient algorithms, and evaluate trade-offs, which are fundamental software design and engineering skills.

References:

GfG. (2023, January 23). *Deep Q-Learning*. GeeksforGeeks.

<https://www.geeksforgeeks.org/deep-q-learning/>

Reinforcement Learning (DQN) Tutorial — PyTorch Tutorials 2.2.2+cu121 documentation.

(n.d.). https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

Staff, K. a. W. (2023, November 9). *How can AI and the human brain work together?*

Knowledge at Wharton. <https://knowledge.wharton.upenn.edu/article/how-can-ai-and-the-human-brain-work-together/>