Efficient Parking using Q-Learning

CMSC 421 Semester Project

Asher Anand, Pushkar Bhargiri, Ethan Lott, Patrick Marinich, Josh Pollack

Literature Review

Influence of Course Materials: The foundational understanding of Q-Learning for this project mainly sprouted from Homework 2 and related lecture slides. The base code that we used for HW2 proved to be instrumental in helping us create the frame for our parking lot simulator. Since we wanted to also train an AI agent, this knowledge of Q-learning fit perfectly. With the guidance provided in the relevant materials we were able to understand the concepts of state management and how to balance different rewards to accomplish a satisfactory outcome. The lecture slides were also an important resource as they taught us Q-learning principles such as exploration vs exploitation which helped us further refine our AI's decision making algorithm. We understood how to apply the Q-Learning concepts into code with the help of some of HW2's base code. The gymnasium documentation also played a key role in our understanding of training spaces and how an agent can travel through them. For our rendering, we used pygame so our introduction to the documentation in HW2 helped us figure out how we can properly render the visual portion of our final project.

Utilization of Online Resources: We were able to grow our project from the HW2 base code by reading the gymnasium documentation to see what else was possible with reinforcement learning and how we could use its capabilities to assist us in building an AI to that could explore multiple environments given the constant changing parameters of visualizing different size parking lots. For the visualization portion, we were able to render the simulation using pygame. Reading this documentation allowed us to create lines and shapes to give a clear indicator to potential users on what spot they would be parking at.

Inspirational Elements and Real World Observations: The practical inspiration for this project came as an extension from the parking management system at BWI Airport, as shown on their website. The concept of making parking easier by not having to pull all the way into a lane to locate a spot, made us think how could we take this a step further? We proposed that maybe you don't even have to enter the parking lot to find your solution, and thus our idea was born.

Introduction

The problem that we chose to investigate is an AI agent's ability to find the optimal parking spot in a crowded parking lot. In real life, a common problem faced when trying to park is getting the closest space possible to the destination without being able to physically see which of the parking spots are filled. Within a parking lot, there are a lot of different factors to consider, such as the parking lot size, the fullness of the lot, and where the 'best spot' is located. To emulate the ability for the agent to not be able to "see" each of the parking spots we chose Q-Learning for our learning method as during training the agent would have to explore the lot in an effort to learn where the best spot is.

Initial Goals

Initially, our goal was to have an agent capable of finding the best parking spot given the arrangement of the parking lot. In our development environment, we had the agent start at the top left of the parking lot, and the best parking spot would be located on the bottom right, the furthest away. In addition, there would be a random amount of parking spots filled up to simulate how a parking lot in real life may look. Given these conditions, our initial goal was to have the agent be able to find the best parking spot in a short amount of time. One of the reasons that we chose this to be our main goal is that it has practical applications for a parking lot that contains sensors in their parking lot. If we are able to download all this sensor information, we could train an AI agent using our algorithm and find the real-life driver a spot in a very short amount of time.

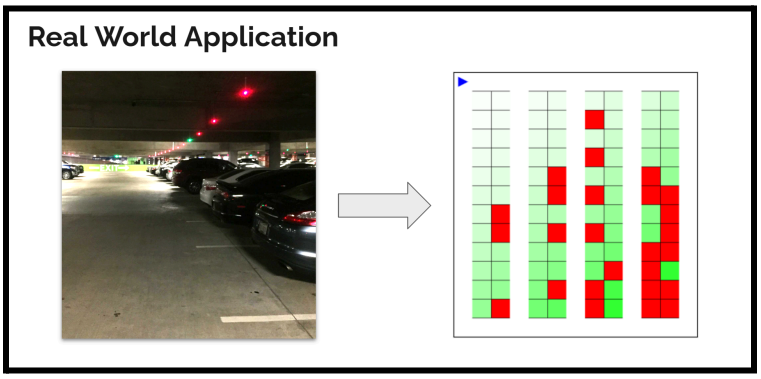

Figure 1 - This is an image from our presentation/poster describing a potential use case for our agent. The left picture is an image from a parking lot with light sensors which are used to detect if a car is in the parking spot. If this information were to be available it would be sufficient to train our agent into finding its way to the best available parking spot.

Our Solution

Our solution used Q-Learning and was able to accomplish our main goal of finding the best spot in the lot, in a short time frame, in a majority of the cases. There are a multitude of different ways that this solution could be expanded in future development to account for other aspects of different parking environments. Our implementation is sufficient for simple parking lots with uniform rows and columns, however, our solution would have to be altered for more complex parking lots for it to be as successful.

The environment that we built for our agent to learn provides a pivotal role in how the agent learns and reacts. To mimic the functionality of a parking lot, we defined two distinct space types, road spaces and parking spaces. The agent is only allowed to traverse through the road spaces to look for the best parking space and is not allowed to cut through the parking spaces, similar to real life. For parking spaces, there are additional pieces of information: whether or not the spot is occupied by another car and if not, how close it is to the optimal spot. In addition, to further simulate the parking lot experience our agent has three possible actions to take, similar to a real car, this would be forward, turn left, and turn right. This again mimics the limited

possible actions a real-life driver has when finding a parking spot. A final restriction that we incorporated is that the agent cannot enter a spot from the top or bottom lane, similar to real life. Our final implementation of our environment also includes a functionality to fill up the parking lot non-uniformly. In a real-life parking lot, the best spots would be taken first, and thus the lot would be more densely filled around the best spots, this is something that we wanted to emulate as well. Since our optimal spot is the bottom right spot, the lot will be more filled the closer we get to the bottom right.

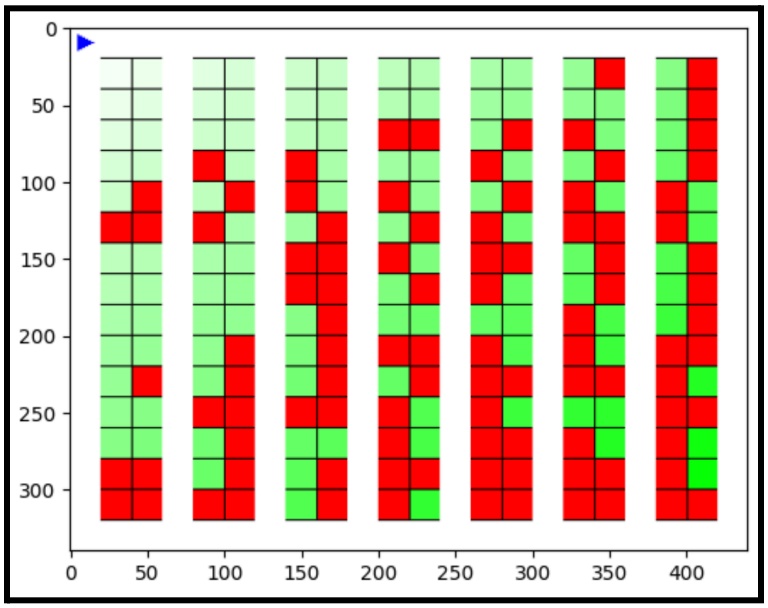

Figure 2 - This is an example environment that our agent can train on and solve. This specific example is 7 different columns of 15 double sided parking spots. The white spaces are the drivable roads, the green spaces are the empty parking spots, and the red spaces are the occupied spots. The shade of green determines the value of the parking spot, the closer to the bottom right the darker the green, which represents a better parking spot.

The basis for our agent's learning ability is Q-Learning. The main reason that we chose this for our implementation is that it is computationally quick and it simulates the agent's ability to initially not know the current environment. Q-Learning uses a balance between exploration and exploitation when learning the current environment. Exploration is the act of the agent trying something that it hasn't previously tried, whereas exploitation is the agent following the best action that it has found in a previous iteration. In our solution, Q-Learning is used to iteratively find the best remaining parking spot, through our experimentation it works well with smaller parking lots, but as the parking lots get larger then the algorithm has to run many more iterations to continue to find the best spot available.

The reward function that we settled on for our final implementation is a relatively simple one. The road spaces are given a reward of zero since we do not want the agent to end its traveling on a road space, but rather in a parking spot. Then the maximum possible score is determined by the Manhattan Distance from the upper left parking spot to the lower right parking spot,

skipping the road spaces. The maximum score is given to the bottom-right most parking space, and then the scores of every other space are given by the maximum score subtracted by the Manhattan Distance from the optimal location. Then after the positive rewards are described, any location that is deemed to be an occupied space is given a score of the negative maximum.

```
[[  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   1.   2.   0. -15.   4.   0.   5.   6.   0.   7.   8.   0.]
 [  0.   2.   3.   0.   4.   5.   0.   6. -15.   0.   8. -15.   0.]
 [  0.   3.   4.   0.   5. -15.   0.   7. -15.   0.   9. -15.   0.]
 [  0.   4. -15.   0. -15.   7.   0.   8.   9.   0. -15. -15.   0.]
 [  0.   5.   6.   0.   7.   8.   0.   9. -15.   0. -15.  12.   0.]
 [  0.   6. -15.   0.   8.   9.   0. -15. -15.   0. -15. -15.   0.]
 [  0. -15.   8.   0.   9. -15.   0. -15.  12.   0. -15. -15.   0.]
 [  0.   8. -15.   0.  10.  11.   0. -15.  13.   0. -15. -15.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]]
```

Figure 3 - This is an example of what our parking lot looks like inside our code. This lot has 4 columns of 8 double sided parking spots. The zeros represent the drivable road, the positive values represent the scores given if the agent parks in those unoccupied locations, and the negative values represent the parking spots where there is another car already parked there. In this example, the optimal spot would be the bottom right spot of the 9th column, with a score of 13.

Results

In general, our solution was successful in finding the optimal location in most of the parking lots that we tested it on. It worked the best on the smaller parking lots due to the nature of Q-Learning, since it is iterative and not necessarily complete, there is a chance that if the environment is too large the agent would not be able to find the optimal solution in the provided training iteration count.

Figure 4 - The left image is our Q-Learning agent finding the optimal location in a 4 column, 8 double row parking lot with 60000 training iterations. The right image is the same number of training iterations, but a much larger parking lot (7x15) and although the agent found a spot, it is not the optimal location, the optimal location would have been the furthest down green space in

the last column. This shows that as the parking lot size increases, the training iterations must also increase for the agent to continue to be optimal.

A side-effect of choosing Q-Learning as our learning algorithm is that the agent's failures are not detrimental. In *Figure 4*, the rightmost image shows a typical failure of our agent where it did not find the optimal location. Although it did not find the optimal location it still found a pretty decent parking spot, however when considering our goal, this is a failure. In a real life scenario, finding a sub-optimal open parking spot is a much better failure case than crashing into a car by attempting to occupy a parking spot that is already occupied. Since Q-Learning will follow the best policy that it finds during training, assuming that there is at least one open spot near the agent's starting position it will always have said open spot as a possible positive outcome as opposed to attempting to take an occupied spot. In an application setting, our agent's lack of catastrophic failure would be a potential strength of our agent.

Strengths, Weaknesses, Limitations

Due to the nature of Q-Learning, there are a multitude of different strengths, weaknesses, and limitations of our solution. Our model provides some key strengths that are critical to the success of a real life application of our model. Firstly, the training time for Q-Learning is relatively short. In the example shown in *Figure 4* with 60000 training iterations, the training would be complete in around a minute. In a real life application scenario, the application could have dedicated hardware which would allow for faster and more training iterations. Another strength of our model is that it avoids catastrophic failures since the Q-Learning agent will always follow the best policy that it finds. Due to the underlying structure of how our rewards are determined, the agent would always opt for a bad unoccupied spot rather than crashing into a car in an occupied spot.

There are some limitations of our current implementation which are important for discussion. The main limitation is that our environment is static, whereas in real life parking lots are typically dynamic. Our environment assumes that for the duration of the training time, no other cars enter the parking lot and no occupied spots are vacated. This is mitigated by our short training time as within a minute a parking lot very likely will be a static environment provided it is a small enough parking lot. Another limitation is that our parking lot design accounts for one specific type of parking lot, a flat symmetrical parking lot. There are other types of parking lots out there with different goal placements, such as parking garages where the parking lot is three-dimensional and the optimal parking spots are near the elevators. In its current form, our model does not take these into account and could be a source for future development to include more customization of the parking lot environment.

Finally, there have been some weaknesses observed with our Q-Learning model, specifically the idea that Q-Learning is not necessarily complete and optimal but probabilistic. Since Q-Learning is an iterative process, there is a chance that the optimal parking spot is never found by the agent during its training. This is a problem typically seen as the parking lots get larger because the probability of the agent exploring the bottom right portion of the parking lot

decreases the more moves the agent has to make to get there. This effect can be mitigated by tuning the hyperparameters of the Q-Learning algorithm or by increasing the training iterations. However, even with these mitigation strategies, there is no guarantee that the agent will find the optimal solution in a given training time.

Group Member Contributions

Asher Anand - I helped with the brainstorming of the project by discussing ideas such as an increased reward for being able to “pull through” a parking spot, though these were eventually determined to be beyond the scope of this initial effort. I modified the reward function to correct an error we saw with the agent entering a parking spot from the side, rather than pulling in parallel to the lines. I also modified the display to progress frame by frame on keypresses rather than on a timer to aid in the development process. I also contributed to the final presentation with content discussing the reward function issue we saw and potential future directions for the project.

Pushkar Bhargiri - I was able to help in the brainstorming part of this project by providing ideas such as having punishments for finding an occupied spot. I revised the final report, adding necessary details, revising sentence structure, and making grammatical changes. I also wrote the initial draft of the literature review. I helped format the slides for poster purposes and I helped design and create the poster presentation.

Ethan Lott - Because our idea involved learning an optimal path in a grid-type environment, my main contribution was incorporating the tools and strategies used for Homework 2 into our custom Parking Lot environment. This required further investigation of the Gymnasium and PyGame libraries so that we could increase the dimensionality of our state and action spaces and tune the agent’s behavior to our specifications. Additionally, I contributed to the code that computed the rewards of each parking spot and which spots should be filled by other cars.

Patrick Marinich - For this project, I was a part of the initial brainstorming of the initial idea and some of the features that we decided to include in our final project. I did some of the programming on the visual side of the project and created the parking spot lines that outline the parking spots. I spent time helping to put together the final presentation slides. In addition, I worked on experimenting with our model to find its shortcomings in the larger parking lots. Finally, I was responsible for the initial draft of the final paper report.

Josh Pollack - I came up with the original concept of using an AI in a parking lot simulator to find the optimal spot. I helped brainstorm ideas for better visual representation to make the lot clearer to the observers. I proofread the final report, ensuring that each section had the necessary information and aligned with the requirements of the guidelines. I worked on the

slides to explain what our goals were and our future directions. I wrote the final draft of the literature review, creating three relevant sections that each described the types of sources we utilized to build our knowledge base. Lastly, I gathered the materials for the poster board and helped create it.

Bibliography

Farama Foundation. “Gymnasium Documentation.” Gymnasium Documentation, 2023, gymnasium.farama.org/index.html.

Huslin, Anita. “Smoother Trip in BWI Garage.” Washington Post, 17 Jan. 2024, www.washingtonpost.com/archive/politics/2001/04/18/smoother-trip-in-bwi-garage/cdd3031b-af92-49cf-b712-cd62a1ea9769.

Kim, Sujeong. “MDPs 1”, CMSC 421, 20 Feb. 2024, University of Maryland, College Park. Class Lecture.

Kim, Sujeong. “MDPs 2”, CMSC 421, 22 Feb. 2024, University of Maryland, College Park. Class Lecture.

Kim, Sujeong.. "Homework 2: Reinforcement Learning." CMSC 421, University of Maryland, College Park, 2024.

Pygame developers. “Pygame Documentation.” Pygame Front Page - Pygame v2.6.0 Documentation, 2023, www.pygame.org/docs/.