Comparison and Modification of RL Agents for Parking

Mitchell Foo, Chloe Hong

***Abstract*—For our project, we explore the parking environment from *highway-env*, a collection of environments for autonomous driving and tactical decision-making tasks. Parking is a goal-conditioned continuous control task in which the ego-vehicle must park in a given space with the appropriate heading. We train policy-based, Q-learning based, and model-based reinforcement learning (RL) agents and compare their quantitative and qualitative results. We make modifications to the model-based agent's observation space and reward function as ways to improve its learning and customize the behavioral performance of the agent.**

I. Environment

The environment for our project comes from a collection of 2D environments called highway-env[1], of which we are using the parking environment. The environments are designed for decision-making in autonomous driving, where the primary agent is a car trying to navigate within a specific area or lane to avoid collisions or achieve particular goals. The parking environment consists of different parking spots where one is the designated spot for the agent to park in. The difficulty in this environment is for the agent to learn how to control the car's continuous actions of acceleration and steering.

The observations made by the agent in the environment are processed into a 6-length vector, which includes the x, y position and velocity of the agent and its angle of rotation described by $\sin\_h$ and $\cos\_h$. The episode ends when the agent vehicle is close enough to the goal, crashes, or the maximum time is reached. The agent has a continuous action space where it must control its acceleration and steering.

For our experiment, we explore ways to improve the agents against the vanilla model-based implementation by changing the rewards and observations to allow the agent more information about its environment.

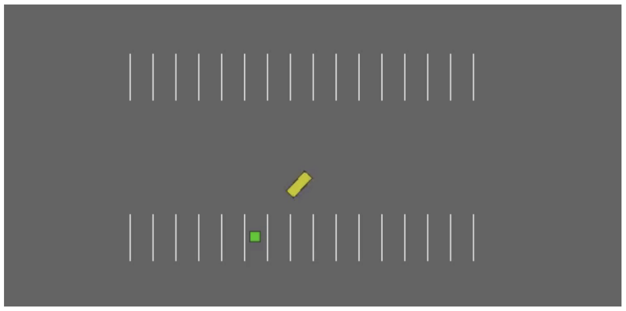

Fig. 1. Agent navigating in the parking environment

II. Reward Function

Rewards are given to the agent based on its proximity to the current parking goal. In order to calculate this distance as a reward, a weighted p-norm is used. If there are obstacles in the environment, the agent will also be given a negative collision reward.

For the reward function, let R be the total sum of rewards, let a be the agent's current position, let b be the goal's position, let w be the reward weights, and let c be the reward for obstacle collision.

$$R = \begin{cases} -\sqrt{|a-b| \cdot w} + c & \text{if colliding with an obstacle} \\ -\sqrt{|a-b| \cdot w} & \text{otherwise} \end{cases} \tag{1}$$

III. Methods

With the current agent parameters, we notice that the environment is designed such that the agent is only aware of itself and is guided to the goal via the reward function. Though we recognize that this is the original author's intention for the environment, we also notice that it limits the agent's ability to observe where it needs to go. We make modifications to explore the potential for an agent to be more efficient with the given goal-based task by modifying the agent's observations and reward weights.

A. Modification 1: Change the input domain

In the current input domain, the agent observes a six-length vector that includes its position, velocity, and rotation. Our proposed modification is to include the position of the goal within this observation, extending the observation to be an eight-length vector.

We hypothesize that information on the position of the goal can aid in the agent directing the vehicle more efficiently to the goal. Our current observation of an agent trained in the current environment is that because it does not observe the goal, it begins moving in a random direction at first to learn towards which direction it actually needs to move in. Being able to observe the goal directly could potentially eliminate this initial exploratory step.

To implement this addition, highway-env's **observation.py** handles all the different observation classes for its varying environments. The parking environment uses the *Kinematics-GoalObservation* class, which allows for calling the observation of the agent, the observation of the agent's achieved goal, and the observation of the desired goal. The new observation of the agent is implemented by concatenating the original observation vector with the x, y position of the desired goal.

B. Modification 2: Change the reward function

For our change in reward function, our plan is to adjust the weight of the rewards, which is factored by the p-norm distance between observation vectors of the agent's achieved goal and the desired goal. The reward weights are in a six-length vector that corresponds to the observation feature vector. Where f is the feature vector for the agent observation, and w the reward weights as a hyperparameter:

$$f = [x, y, vx, vy, cos\_h, sin\_h]$$
$$w = [1, 0.3, 0, 0, 0.02, 0.02]$$

The default weights prioritize the proximity of the car in the x-axis, less in the y-axis, and give little weight to the car's rotation. We recognize that this was strategic given the parking environment, to make sure the agent prioritizes moving in the x-axis because the environment is set up as a traditional parking lot. We propose modifications to these reward weights to see whether giving more weight to the rotation could aid the agent in aligning with the parking spot. We changed the weights as such:

$$w = [1, 0.2, 0, 0, 0.1, 0.1]$$

To implement this, each environment within highway-env has its own script. Within the script for the parking environment, one can change configurations including the reward weights.

IV. Training Parameters

We used the parking environment to test three reinforcement learning algorithms, PPO, an on-policy Policy-gradient method, SAC, an off-policy Q-function-based method, and CEM with model-based methods.

For our modifications, we decided to use model-based reinforcement learning and maintain consistent CEM parameters between runs.

A. PPO with Stable Baselines3

- network architecture = [256, 256]
- n steps = 2048
- batch size = 32
- n epochs = 10
- gamma = 0.0.8
- gae lambda = 0.95
- clip range = 0.2
- normalize advantage = True
- entropy coefficient = 0.0
- vf coefficient = 0.5
- max grad norm = 0.5

B. SAC with Stable Baselines3

- network architecture = [256, 256]
- learning rate = 0.0005
- buffer size = 1000000
- learning starts = 100
- batch size = 32
- tau = 0.005
- gamma = 0.8
- train frequency = 1
- gradient steps = 1

C. CEM

- collection size = 1000
- collection action repeat = 2
- network architecture = [64, 64]
- learning rate = 0.01
- epochs = 1500
- predict action repeat = 1
- cem horizon = 5
- cem population = 100
- cem selection = 10
- cem iterations = 5
- n iteration = 5

V. Results

We compared the average achieved scores from a random agent, PPO agent, SAC agent, and a model-based agent with the proposed modifications.

- Random agent score: -57.49 after 1 episode
- PPO score: -37.95 after 550 000 steps
- SAC score: -20.2 after 550 000 steps
- Model-Based score: -24.35 after 5 iterations
- Model-Based with Reward Modification: -24.43 after 5 iterations
- Model-Based with Observation Modification: -24.43 after 5 iterations

Important to note is the training time per agent. For PPO to train for 550 000 steps, would take around 1.5 hours. For SAC, would take around 4 hours. For CEM, training took around 30 minutes.

A. Base Algorithm Comparisons

Our findings were that in the same amount of training steps, SAC performed consistently better than PPO. SAC performed similarly to the model-based method, though it is noted that model-based methods are parameterized by trajectory iterations instead of training steps. All the methods were able to improve upon the random agent. SAC and CEM agents consistently were able to navigate to the goal whereas the PPO agent, could either achieve the goal, get close to it, or drive far from it.

A qualitative comparison between the agents reveals different agent behaviors between policy-based methods and model-based methods. We observe that in policy-based methods, the agents are better at picking the best route to the goal especially seen in the SAC agent. It is not to say that the model-based agent does not have a good path too, it is just that it corrects itself more and its movement is subtly more twitchy. That being said, policy-based agents struggle more with getting the acceleration right, often moving too fast and overshooting the goal, having to reverse back. This is where model-based methods prove better where the agent slowly makes its way to the target and stops when it needs to.

B. Evaluating Modifications

The general observation of the two modifications is that they did not necessarily improve the agent's total average return but rather in lessening the iterations it takes for the agent to reach convergence at an average reward of around -24. We notice the most substantial difference when expanding the observations of the agent to include the goal as well.

Qualitatively, the agent with modified observations behaved similarly to the original model-based agent, but changing the reward function to favor agent rotation, did have an impact on the way the agent drove to the goal. By favoring a rotation that matched the goal, we noticed that the agent would often drive in a curve towards the x-axis of the goal, and once aligned with it, would back into it. That being said, this tends to happen when there is enough distance between the agent and the goal, whereas if the agent is already close to the goal, it would still drive directly to it.

Video Links

- Random Agent Videos Link
- PPO Agent videos link
- SAC Agent videos link
- CEM Agent videos link
- CEM Agent with input modification videos link
- CEM Agent with reward modification videos link

VI. Conclusion

A. Discussion

From our exploration, we observe the effects of different RL approaches on highway-env's parking environment. Comparing On-Policy methods, Off-Policy methods, and model-based methods we can generalize that with more training we can potentially see better rewards from policy-based methods, but model-based methods are a lot more sample efficient and will still be successful in learning the task.

We believe that this MDP is best explored via model-based methods due to its sample efficiency as well as the nature of the task and action space. Because the rewards are goal-based and sparse, model-based dynamics are independent of the goal thus making it a simpler task to learn than policy-based methods. Also, intuitively, because the dynamics are simple (just steering and acceleration) but the optimal policy is comparatively complex, this problem well suits model-based methods.

For our first modifications, we found that expanding the observations of the agent to include the goal itself allowed for learning the task in fewer CEM iterations. This met our expected hypothesis about aiding the agent to train more efficiently. That being said, it does not improve the overall returns.

For our second modification, we found that the agent did converge one iteration faster than the original model-based agent as well as exhibited unique qualitative behaviors from the other two agents. We increased the rewards that weighted the agent's rotation alignment with the goal and though it did not necessarily always park in an aligned fashion as we had hoped, an interesting behavior in the agent did emerge where it would turn in front of the parking spot and then reverse into it. We observed that the reward function and its weight can be useful for suggesting a certain behavior in an agent.

In conclusion, our tests highlight the trade-offs of using different RL algorithms in a continuous goal-based environment. Our tests also show the effects of modifying reward and observations of the agent, affecting its behavior and efficiency. This show the advantages of exploring a problem in simulation, where we can change how we model the agent and environment based on how similar we want to realistic constraints.

B. Future Works

A key takeaway is how different reinforcement learning techniques are useful for different applications of robotics, or in this case, autonomous driving. What is interesting about the parking environment, in particular, is that though it is a simple goal, the task of parking as a whole suggests both getting to the spot and also in a certain safe and aligned manner. Where in this environment, the car can cross lanes and is encouraged to take the fastest and hence shortest path to the goal, an interesting way to further expand this environment is to instead prioritize safety and real-life parking behaviors over getting to the goal.

To encourage safety, a way to do this is to make the environment more complex with different types of obstacles. This could include adding parked cars to the environment and/or treating the parking lanes as solid walls. Colliding with an obstacle would give the agent a very large negative reward and end the episode. That being said, by making the environment more complex in this manner would also require making the agent's observation more complex to sense these obstacles. Some ideas could be allowing the agent to observe the position of the nearest car or have lidar-like capabilities of sensing objects in its proximity.

Lastly, another direction is to further explore a more gradual reward and goal-based system to encourage the agent to park in a more realistic manner. This could likely look like having a series of goals that the agent needs to get to in order to park. A simple example would be a goal in the agent's x-axis that it would first drive to and is positioned for the agent to efficiently turn into the last goal, which is the parking spot itself. This could also allow for exploring back-in versus front-in parking.

References

[1] Leurent, Edouar. "highway-env." Python, December 12, 2022. https://github.com/eleurent/highway-env.

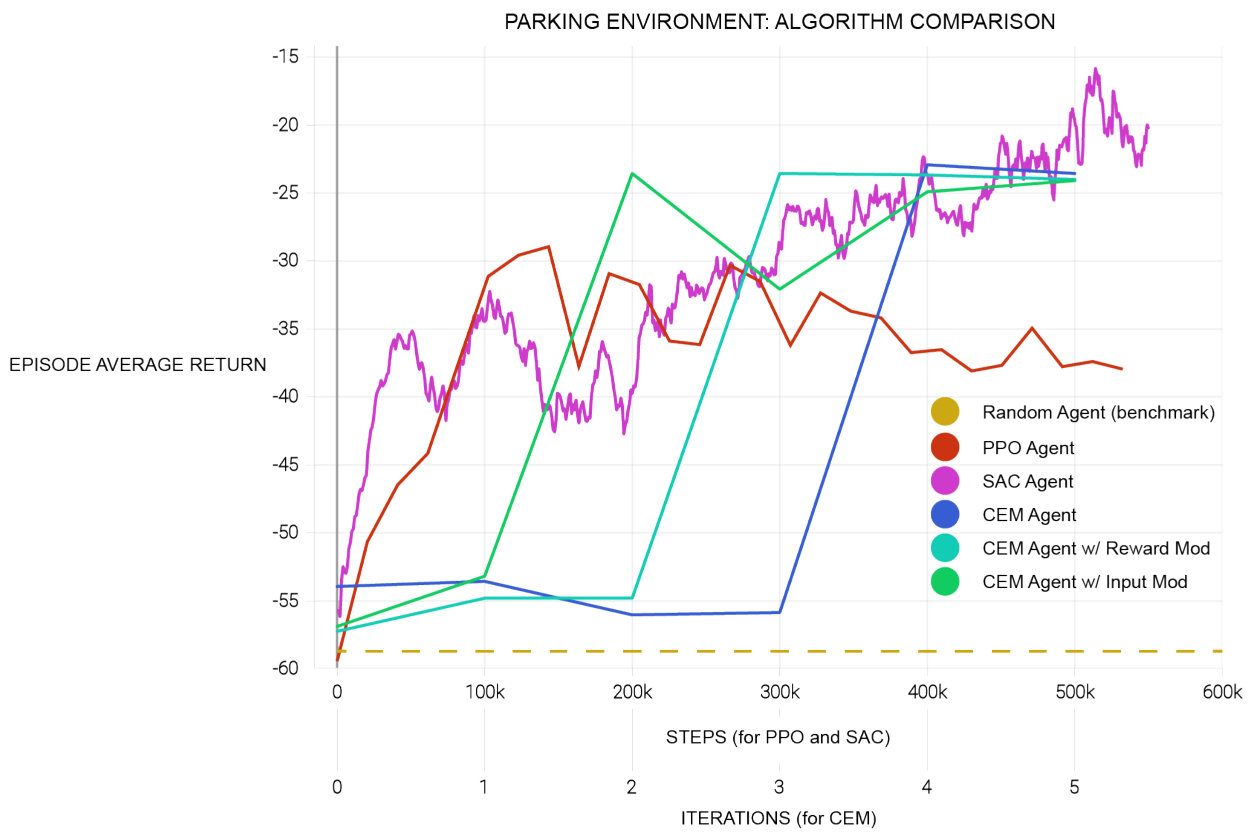

Fig. 2. Comparison of Agent Performance