Assessment 3

SIT 796

Introduction

This article is specific to why policy iteration is infeasible for the environment "**CarRacing-v0**". The article explains the challenges related to mapping the continuous action-spaces and state-spaces specific to "**CarRacing-v0**" and concludes with arguing infeasibility of assigning transition probabilities for this environment for the purpose of policy iteration. Finally, the article concludes with possible computer vision/deep learning techniques which can be used to assign transition probabilities to state/action space found for this environment by providing the agent information of the terrain ahead.

Discretization of action space

The action space of this environment is continuous, with three floating point values [steering, throttle, brake]:

Steering: Takes values from [-1,1]: With -1 extreme left and +1 extreme right.

Throttle: Takes values from [0,1]: With 0 no Throttle and 1 max intensity Throttle.

Brake: Takes values from [0,1]: With 0 no Brake and 1 max intensity Brake.

Although the action space is continuous, it is possible to map continuous actions to discrete actions using hard discretization [1]

Turn left → [-1.0, 0.0, 0.0]

Turn right → [+1.0, 0.0, 0.0]

Brake → [0.0, 0.0, 0.8]

Accelerate → [0.0, 1.0, 0.8]

Do-Nothing → [0.0, 0.0, 0.0]

This shows how the continuous actions can be mapped to discrete actions. While the discretization of action space was possible, the same was not feasible for observation space, hence making the problem infeasible for policy iteration [discussed in next section].

Dimensionality reduction of state space

The observation space for the environment is initialization of a randomly deformed circular track. This track is represented by an RGB frame of dimension 96 x 96 x 3. This gives a huge number of ***27648 values*** representing a single observation space. While this can be reduced to some extent as shown below, even then manual mapping of action-state space is infeasible.

Here are few of the approaches which I used to reduce the observation space of the environment.

Dividing the frames into bins: This approach first converts the frame into grayscale colorspace, then the frame is divided into 12 x 12 grids. These individual grids are then considered a single state, this approach reduces the observation space drastically, but does not capture the complexities of the environment as because of the random initialization there are several sub grids which are redundant and do not contribute to action taken by the agent.

Using only Green Channel from RGB frames: This approach builds on the above approach of using a single channel instead of RGB channels, but instead of grayscale, this approach uses green channel of the RGB frame. This provides better separation between road and grass pixels; this reduces the observation space to 96 x 96 x 1. Even this approach fails to reduce the observation state to a manageable state.

Using 10 x 10 cropped frame from 96 x 96 frame [2]: This approach is slightly modified from [2] and uses green channel image and a cropped region around the agent of area 10 x 10. This brings the dimension from 96 x 96 x 3 to 10 x 10 x 1. This results in 100 values representing the observation space. While this is a very low number as compared to the initial dimension, the random initialization of the track and the random turns associated with the environment makes it impossible to map action-state space to reasonable transition probabilities. The degree of turns makes it difficult to classify a turn into discrete turns using traditional computer vision techniques.

None of the proposed approaches could reduce the observation space to a manageable dimension. Because of the huge observation space and random track initialization, it is infeasible to manually map action state pairs and assign transition probabilities.

Conclusion

For the reasons stated above this problem is unsuitable for policy iteration algorithms as all the components of MDP are not known for this environment. Few approaches could be used to make this feasible but are beyond the scope of this task:

1. A classifier (ML or deep learning) predicting the upcoming track nature (straight, left turn, right turn, U-turn) could be trained to give discrete states which could then be mapped to discrete actions. This would require track data to train the classifier to identify the upcoming terrain. This would help the agent gain the knowledge of upcoming track and make policy iteration feasible.
2. Traditional computer vision techniques can also be explored to achieve discretization of observation state into states mentioned in previous step, although this would suffer from poor generalization as the track generated is random and the degree of turns would make assigning a label a tough task.

This environment can be considered a perfect example of arguing the case for model-based Reinforcement Learning and where Dynamic Programming based methods would be considered infeasible to achieve the optimal solution.

References

[1] https://notanymike.github.io/Solving-CarRacing/

[2] https://scientific-python.readthedocs.io/en/latest/notebooks_rst/6_Machine_Learning/04_Exercices/02_Practical_Work/02_RL_2_CarRacing.html