ECS7002P Artificial Intelligence in Games: Group Assignment 2

Suyog Pipliwal 210634338
Paul Sasha Mutawe 210715187
Animesh Devendra Chourey 210765551

Queen Mary University of London — January 9, 2022

Abstract

This report is a reflection of work done for group assignment-2 for module ECS7002P. The objective of this assignment is to implement tabular and non-tabular algorithms for finding optimal policy for the given environment.

1 Introduction

On a macro view we covered four main bases in the report, namely creating the frozen lake environment that consisted of created methods p and r plus corresponding methods that made p and r work to the description provided in the assignment brief. Method p returns the distributional probability of going from one state to another and the r method returns the reward for reaching the goal state. We also worked with tabular and non-tabular algorithms. Tabular methods refer to problems with state action pairs small enough to approximate value functions with tables or arrays which were used to implement policy evaluation, policy improvement, policy iteration, and value iteration. Non-tabular methods were used in the implementation of Sarsa control and Q-learning control using linear function approximation.

2 Question

1. The whole code of assignment is organized in one package. The main.py file contains the main function. It has two variables *lake and big_lake* which represent the two different starting states of the frozen lake environment used for testing. It runs on the default parameters and prints out the policy and values returned by different approaches. The different algorithms are implemented into their perspective files like sarsa.py containing code for the sarsa algorithm, policyEvaluation.py contains code for policy Evaluation etc. The chooseAction.py is the implementation of the epsilon greedy policy for choosing action used in Sarsa and Q-learning. The output.txt is the output of the main function.

2. For the policy iteration to find the optimal policy for the big frozen lake, the number of iterations it required are 6. Whereas, for the value iteration the number of iterations required are 20. Since the value iteration algorithm required more iterations, it was slower as compared to the policy iteration algorithm.

3. Under the same commom conditions, both converge to the real value functions but at different rates. Sarsa contol require 2000 episodes to find the optimal policy which can be further optimzie. whereas Q-learning control required around 500 episodes to find the optimal policy for the the small frozen lake.

4. Number of features are calculated by number of states * number of actions.These features are needed to calculate the dimensions of ϕ(s,a) vector as it's dimensions are number of actions by number of features. Parameter vector θ has same dimensions as number of features. One hot encoded feature vector ϕ(s,a) can be used as a selector using dot product to filter out the values of the unrelated (state,action) pair. Whike training the parameter vector θ can be updated to again utilize the feature vector to correctly update the encoded (state,value) and not altering the unrelated (state,action) values.

   Non tablular model free algorithm does not hold every (state,action) value and the policy in the memory, when the environment is finite. It just encodes the state in question without interrupting the run of the program. Parameter vector θ stores the general information regarding what values of the states and policies that must be taken. This can be decoded later to get the actual (state,action) values for the state using one hot encoded feature vector. Whereas the tablular mopdel free algorithm is a special case of non tabular model free algorithm becuase the table is created beforehand. It is a special form becuase the encoded values are individually kept in the memory.

5. We were not able to find optimal policy for the Sarsa control and the Q-learning control for the big frozen lake. This could be becuase the space is too large. Both the Sarsa and Q-learning take the updates the values of the path which they take whenever they reach the terminal state. After that, the values are discounted at every step from terminal to the initial state by both the algorithms. The states that are closer to the goal state have larger values. Therfore, if the path becomes too large, the states closer to the initial state will not be gathered by the algorithms and thus the steps will not be updated efficiently. The algorithm is not able to learn anything from the big frozen lake environment as all the values returned are zeros which are similar to the values of the first episode.

   Sarsa and Q-learning works effectively when the state space is not too large. When the state space is not too large, both the algorithms will be able to successfully construct a path using the (state,action) values, as the discounts affect the values of the states that are closer to the initial states considerably less.

3 Result and conslusion

There are a number of tabular and non-tabular algorithms for finding the optimal policy for the given environment. Each one has its advantage based on the environment. For this assignment, we are using give environment small frozen lake and a big frozen lake. We can find the optimal policy for the small frozen lake environment using policy iteration, value iteration, Sarsa, Q-learning and linear approximation of Sarsa and Q-learning. To find the optimal policy of a big frozen lake we can use policy iteration value but Sarsa and Q-learning can not find the optimal policy for a big frozen lake because of the large state space..