Assignment 2: Semi gradient Sarsa

Student Srinivas Mukund: 202005966: x2020fvn@stfx.ca

Introduction

Semi gradient SARSA is an on policy approximation method which uses a differentiable parameterized function composed of a set of weights whose cardinality is equal to that of it's feature vector. The semi gradient version of SARSA differs from it's semi gradient TD(0) counter part in the sense that we have to also include the action somehow along with state in the features. The qhat values are calculated using a simple dot product of the weights with the feature matrix. We also need the gradient of the qhat which is nothing but the feature matrix itself when it comes to linear approximation functions. An action is taken using the epsilon greedy policy and the new weight matrix is simply calculated using the provided update method in the algorithm based on of the agent is in the terminal state or any state other than the terminal state. In our implementation, we have set self.done to False and plot the graph once agent is in terminal state. Otherwise we plot the evolution of the values after every 10 time steps in the terminal.

Implementation

As per the instructions of the assignment, we created a custom gym environment with a grid size of 10 x 10, composed of walls which is value 0 in the map, player is 2, floor is 1 and danger is 4. This is our state. For our actions, we defined 4 actions, up, down, right and left. Our reward function consists of +10 reward for reaching goal state, -5 for falling into danger state and -1 for every other state. If the agent runs into a wall, it results in the same state. We also randomize the player position every time the reset function is called.

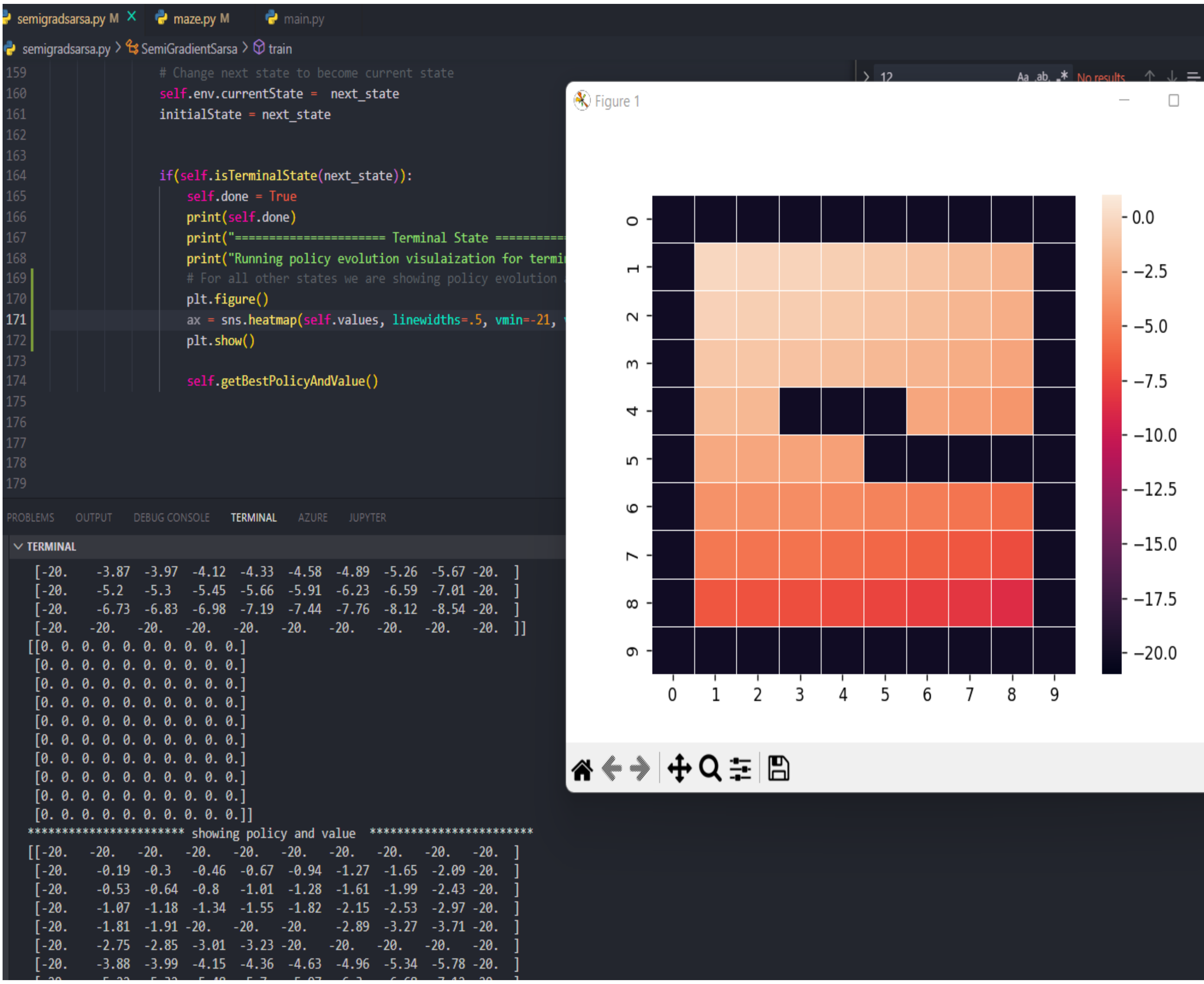
We then created a semigradientSarsa class which consists of methods to check if the agent is in terminal state, a function to return new weights, a get best policy and values function which is printed every 10 steps of the algorithm in the terminal as policy evolution, a train method, epsilon greedy action selection method and generateStateVector method to create the feature vector based on the player position. We implemented a graph based heatmap in seaborn in the last terminal step to show the plot. We decided to do this as the python console gets interrupted every 10 steps and also because we are already printing the results of values an policy to terminal.

For the feature vector, we decided to go with: [1,x,y,x**2, y**2, x+ a, y+a, one hot encoded action vector].

Results

We trained the agent for about 150 steps to obtain the optimized policy. One thing we want to mention is that we have obstacles in between the maze. So, wherever, there is a wall, we simply replaced the value of wall with -20. We did not calculate q hat values for walls as it did not make sense to do so because the agent can never be on a wall.

We are happy to state we are able to get very fats convergence and the policy values look perfect and correct. The goal state has the highest values while states across opposite to goal state had larger values. We used a learning rate of 1e-5 to make sure the agent does not overstep the gradients past the local minima and a gamma of 0.9.


semigradsarsa.py M
maze.py M
main.py
semigradsarsa.py > SemiGradientSarsa > train
# Change next state to become current state
self.env.currentState = next_state
initialState = next_state
if(self.isTerminalState(next_state)):
self.done = True
print(self.done)
print("==================== Terminal State =========
print("Running policy evolution visulaization for termi
# For all other states we are showing policy evolution
plt.figure()
ax = sns.heatmap(self.values, linewidths=.5, vmin=-21,
plt.show()
self.getBestPolicyAndValue()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
AZURE
JUPYTER
TERMINAL
[-20. -3.87 -3.97 -4.12 -4.33 -4.58 -4.89 -5.26 -5.67 -20.]
[-20. -5.2 -5.3 -5.45 -5.66 -5.91 -6.23 -6.59 -7.01 -20.]
[-20. -6.73 -6.83 -6.98 -7.19 -7.44 -7.76 -8.12 -8.54 -20.]
[-20. -20. -20. -20. -20. -20. -20. -20. -20. -20.]]
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
********************** showing policy and value ***********************
[[-20. -20. -20. -20. -20. -20. -20. -20. -20. -20.]
[-20. -0.19 -0.3 -0.46 -0.67 -0.94 -1.27 -1.65 -2.09 -20.]
[-20. -0.53 -0.64 -0.8 -1.01 -1.28 -1.61 -1.99 -2.43 -20.]
[-20. -1.07 -1.18 -1.34 -1.55 -1.82 -2.15 -2.53 -2.97 -20.]
[-20. -1.81 -1.91 -20. -20. -20. -2.89 -3.27 -3.71 -20.]
[-20. -2.75 -2.85 -3.01 -3.23 -20. -20. -20. -20. -20.]
[-20. -3.88 -3.99 -4.15 -4.36 -4.63 -4.96 -5.34 -5.78 -20.]
Figure 1
No results
0
1
2
3
4
5
6
7
8
9
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−15.0
−17.5
−20.0