Example code SARSA

December 7, 2021

1 Lena Feiler - i6246119

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
%pylab inline
import random

# parameters
gridSize = 4
states_terminal = [[0,0], [gridSize-1, gridSize-1]] # these are the goal␣
↪states, we have 2
valid_actions = [[-1, 0], [1, 0], [0, 1], [0, -1]] # we can go left, right, up,␣
↪down
gamma = 0.1 # discount rate
currentReward = -1 # changed to 0 to fit our previous lecture examples
numIterations = 100
alpha = 0.1 #exploration factor

# initialization
Q = np.zeros((gridSize, gridSize)) #only stores value function
states = [[i, j] for i in range(gridSize) for j in range(gridSize)] # 4x4 grid␣
↪--> 16 states
```

```
Populating the interactive namespace from numpy and matplotlib
```

[2]:
```
def generateInitialState():
    #generate a random initial state
    xCoord = np.random.randint(gridSize)
    yCoord = np.random.randint(gridSize)
    initialState = [xCoord, yCoord]  # state will be defined as tuple of␣
↪coordinates
    return initialState

def generateNextAction():
    #generate a random action from the valid set of actions
    action = np.random.randint(len(valid_actions))
    nextAction = valid_actions[action]
    return nextAction
```

```python
# define the transition function from a given state and action
def getNextState(state, action):

    #define what happens when reaching a terminal state
    if state in states_terminal:
        return 100, state

    # here you should complete this step, the transition step
    nextState = np.add(np.array(state), np.array(action))
    nextState = [nextState[0], nextState[1]]

    # if the agent reaches a wall
    if -1 in nextState or gridSize in nextState:
        nextState = state

    return currentReward, nextState
```

[3]:
```python
for it in range(numIterations):
    currentState = generateInitialState()

    while True:
        currentAction = generateNextAction()
        reward, nextState = getNextState(currentState, currentAction) #r, s'

        #complete the stop action if the agent reached the goal state
        if currentState in states_terminal:
            #print(nextState)
            break

        #update the Q-value function Q
        #  (s,a)← (1-alpha)*Q(s,a) + alpha*(( +gamma ^( ', '))
        Q[currentState[0], currentState[1]]= (1 - alpha) * Q[currentState[0],␣
↪currentState[1]]+ alpha * (reward + gamma * Q[nextState[0], nextState[1]])

        #assign as current state the next state
        currentState = nextState
```

[4]:
```python
print(Q)
```

```
[[ 0.         -1.08938553 -1.11050311 -1.11108094]
 [-1.0675988  -1.10971526 -1.11101916 -1.11066288]
 [-1.11009984 -1.11101814 -1.10955141 -1.09199067]
 [-1.1110829  -1.11096884 -1.08566385  0.        ]]
```

[ ]: