DAT405 Assignment 5 - Group 52

Hampus Jansson - (4 hrs)

Erik Johannesen - (4 hrs)

May 3, 2023

Task 1

a) the optimal path is unique, and it is EENNN.

b) E E F

N/E N/E N

N/E/S N/E/S N

E E N

```
    (0,0) - E, (1, 0) - E, (2,0) - N, (0,1) - N/E/S, (1,1) - N/E/S,
    (2,1) - N, (0,2) - N/E, (1,2) - N/E, (2,2) - N,
    (0,3) - E, (1,3) - E, (2,3) - F
    <p>
```

c) 0.

Task 2

Result for 2a

In [2]:
```python
#creating the reward array
rewards = []
for i in range(3):
    rewards.append([])
    for j in range(3):
        rewards[i].append(0)
rewards[1][1] = 10

#discount factor is 0.9
discount_factor = 0.9

#creating the start array, V0 - we started one step further
grid = []
for i in range(3):
    grid.append([])
    for j in range(3):
        grid[i].append(6)
```

```
grid[1][0] = grid[0][1] = grid[1][2] = grid[2][1] = 2
grid[1][1] = 8

#creating the variables that will break the algoritm when it converges
epsilon = 0.5
biggest_difference = 1

#algoritm that breakes when it converges
while biggest_difference > epsilon:
    #saving previous V
    previous = []
    for i in range(3):
        previous.append([])
        for j in range(3):
            previous[i].append(grid[i][j])

    biggest_difference = 0

    #the algoritm for deciding value for each state
    for i in range(3):
        for j in range(3):

            #checking each direction to see which one is bigger

            if j == 0:
                s = 0
            else:
                s = 0.8*(rewards[i][j-1]+discount_factor*previous[i][j-1])+0.2*(re

            if j == 2:
                n = 0
            else:
                n = 0.8*(rewards[i][j+1]+discount_factor*previous[i][j+1])+0.2*(re

            if i == 0:
                w = 0
            else:
                w = 0.8*(rewards[i-1][j]+discount_factor*previous[i-1][j])+0.2*(re

            if i == 2:
                e = 0
            else:
                e = 0.8*(rewards[i+1][j]+discount_factor*previous[i+1][j])+0.2*(re

            #assigning the max value of the directions to the grid
            grid[i][j] = max(s, e, w, n)

            #updating biggest difference
            biggest_difference = max(biggest_difference,  abs(grid[i][j] - previou

#printing result
print(grid)
```

```
[[41.1814610154055, 47.51657647469815, 41.1814610154055], [47.51657647469815, 43.6
20485405649404, 47.51657647469815], [41.1814610154055, 47.51657647469815, 41.18146
10154055]]
[41.1814610154055, 47.51657647469815, 41.1814610154055]
```

2a

The optimal value function is presented above.

The optimal policy based on that: (0,0) - N/E, (1,0) - N, (2,0) - N/W, (0,1) - E, (1,1) - N/E/S/W, (2,1) - W, (0,2) - S/E, (1,2) - S, (2,2) - W/S

2b

The result does not depend on V0 because the weight of V0 becomes very small with the more iterations that is run. Each time the weight is 0.9 less. In the end only the reward affects the result.

2C

The higher the discount factor, the higher factor does the previous V matter in the next iteration. If the discount factor is 0, then the grid will be equal to the rewards. If the discount factor is 1, then the grid will move towards infinity.

4a

Without exploration the algoritm might miss many states that could be more optimal, after finding certain states that gives a immediate rewards. Exploration is necessary to find out if there are actions that provide better rewards.

4b

Supervised learning tasks requires you to train the model first, and then the model tries to mimic the training process. With reinforcement learning you define what counts as a success, and then the model tests by itself and finds out what leads to success.

5a

In a decision tree, you're looking at several variables to try to classify something. Using the training data, you descend the tree to the outcome that gives you the highest probablity, based on the training data.

The weakness of decision trees is that they usually only work well for the training data, and often do not work well for the other data. In a random forest, you create several decision trees based on picking a few random lines from the dataset, and from the new dataset you use a random selection of variables to create the random tree. Then this step is repeated with different random selections everytime. Then when you use the random forest, you put the data through each of the trees and see which classification you got from the majority.

5b

The advantage with using random forests over decision trees is that they are usually more accurate. The drawback is that they are slower, because you need to check more trees everytime.