Lab Session 4 - Scientific Writing

Viet Vu, Fredrik Kortetjärvi, Rohullah Khorami, Mariaguadaloppe Farah

1 Task 1

-0.04	-0.04	-0.04	1
-0.04	W	-0.04	-1
-0.04	-0.04	-0.04	-0.04

Table 1: Grid 1

-0.04	1	-1	-0.04
-0.04	-0.04	W	-0.04
-0.04	-0.04	-0.04	-0.04

Table 2: Grid 2

-0.04	-0.04	-0.04	-0.04
-0.04	W	-1	-0.04
-0.04	-0.04	W	1

Table 3: Grid 3

2 Task 2

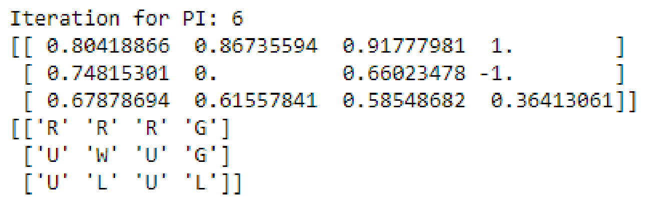
Policy iteration and TD learning has been implemented.

3 Task 3

3.0.1 Question 1

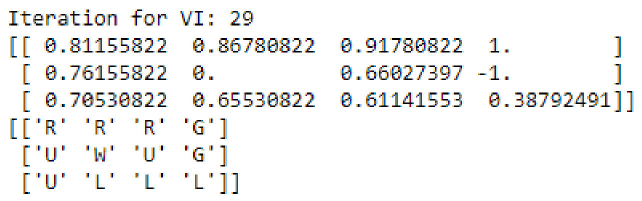
Compare value and policy iterations in terms of convergence time. Relate it to computational complexity, current implementation and number of iterations needed.

Looking at the comparison between Policy iteration(PI) and Value iteration(VI), we can see that PI converge faster compared to VI based on the iteration.



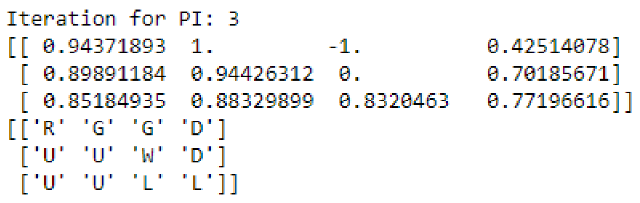
```
Iteration for PI: 6
[[ 0.80418866  0.86735594  0.91777981  1.        ]
 [ 0.74815301  0.          0.66023478 -1.        ]
 [ 0.67878694  0.61557841  0.58548682  0.36413061]]
[['R' 'R' 'R' 'G']
 ['U' 'W' 'U' 'G']
 ['U' 'L' 'U' 'L']]
```

Figure 1: Policy iteration Grid 1



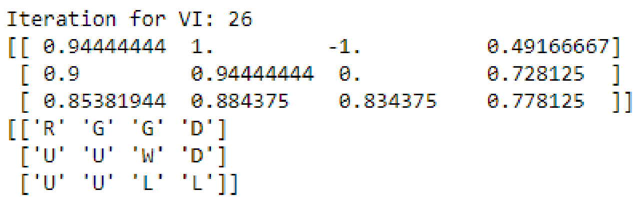
```
Iteration for VI: 29
[[ 0.81155822  0.86780822  0.91780822  1.        ]
 [ 0.76155822  0.          0.66027397 -1.        ]
 [ 0.70530822  0.65530822  0.61141553  0.38792491]]
[['R' 'R' 'R' 'G']
 ['U' 'W' 'U' 'G']
 ['U' 'L' 'L' 'L']]
```

Figure 2: Value iteration Grid 1



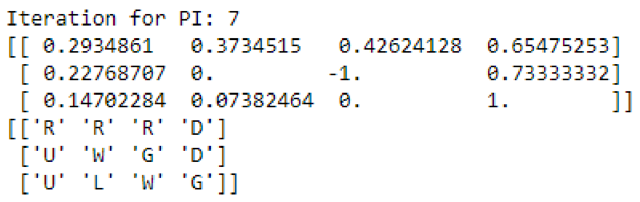
```
Iteration for PI: 3
[[ 0.94371893  1.         -1.          0.42514078]
 [ 0.89891184  0.94426312  0.          0.70185671]
 [ 0.85184935  0.88329899  0.8320463   0.77196616]]
[['R' 'G' 'G' 'D']
 ['U' 'U' 'W' 'D']
 ['U' 'U' 'L' 'L']]
```

Figure 3: Policy Iteration Grid 2



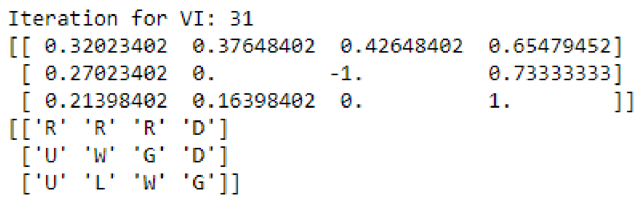
```
Iteration for VI: 26
[[ 0.94444444  1.         -1.          0.49166667]
 [ 0.9         0.94444444  0.          0.728125  ]
 [ 0.85381944  0.884375    0.834375    0.778125  ]]
[['R' 'G' 'G' 'D']
 ['U' 'U' 'W' 'D']
 ['U' 'U' 'L' 'L']]
```

Figure 4: Value Iteration Grid 2



```
Iteration for PI: 7
[[ 0.2934861   0.3734515   0.42624128  0.65475253]
 [ 0.22768707  0.         -1.          0.73333332]
 [ 0.14702284  0.07382464  0.          1.        ]]
[['R' 'R' 'R' 'D']
 ['U' 'W' 'G' 'D']
 ['U' 'L' 'W' 'G']]
```

Figure 5: Policy Iteration Grid 3



```
Iteration for VI: 31
[[ 0.32023402  0.37648402  0.42648402  0.65479452]
 [ 0.27023402  0.         -1.          0.73333333]
 [ 0.21398402  0.16398402  0.          1.        ]]
[['R' 'R' 'R' 'D']
 ['U' 'W' 'G' 'D']
 ['U' 'L' 'W' 'G']]
```

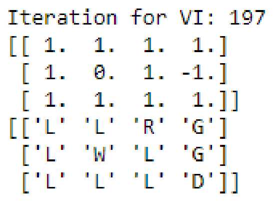
Figure 6: Value Iteration Grid 3

3.1 Question 2

How are optimal policies change with immediate reward values? Show some examples (similar to Figure 17.2b in AIMA).

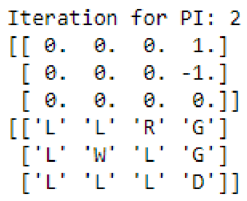
With no immediate rewards, shows that the iteration for VI is 197 and it shows that it doesn't care about where it goes since all states are good even if wrong as long as it reaches the goal. Comparing it to PI with no immediate reward which doesn't take any steps, this is because there's no changes in the next state thus there's no changes.

When the immediate rewards is -0.8, the VI take less iteration and since every states is low it's because most of the time not taking a step would be better since each new state would yield less reward. As for PI, going the top path most of the time to get to the goal is rewarding.



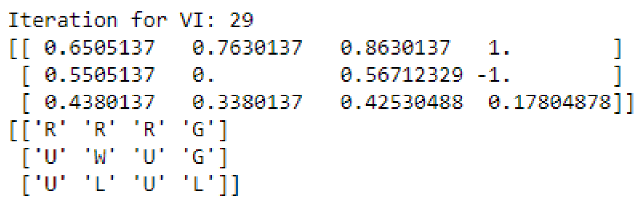
```
Iteration for VI: 197
[[ 1.  1.  1.  1.]
 [ 1.  0.  1. -1.]
 [ 1.  1.  1.  1.]]
[['L' 'L' 'R' 'G']
 ['L' 'W' 'L' 'G']
 ['L' 'L' 'L' 'D']]
```

Figure 7: Value Iteration no immediate reward



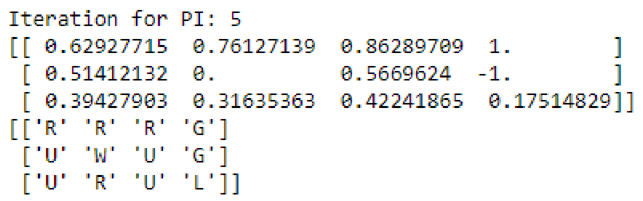
```
Iteration for PI: 2
[[ 0.  0.  0.  1.]
 [ 0.  0.  0. -1.]
 [ 0.  0.  0.  0.]]
[['L' 'L' 'R' 'G']
 ['L' 'W' 'L' 'G']
 ['L' 'L' 'L' 'D']]
```

Figure 8: Policy Iteration no immediate reward



```
Iteration for VI: 29
[[ 0.6505137   0.7630137   0.8630137   1.        ]
 [ 0.5505137   0.          0.56712329 -1.        ]
 [ 0.4380137   0.3380137   0.42530488  0.17804878]]
[['R' 'R' 'R' 'G']
 ['U' 'W' 'U' 'G']
 ['U' 'L' 'U' 'L']]
```

Figure 9: Value Iteration -0.8 reward



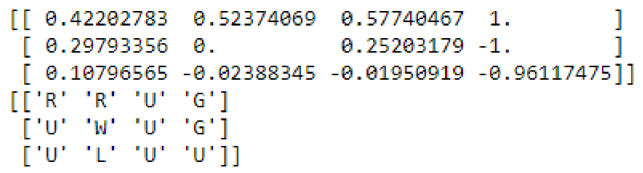
```
Iteration for PI: 5
[[ 0.62927715  0.76127139  0.86289709  1.        ]
 [ 0.51412132  0.          0.5669624  -1.        ]
 [ 0.39427903  0.31635363  0.42241865  0.17514829]]
[['R' 'R' 'R' 'G']
 ['U' 'W' 'U' 'G']
 ['U' 'R' 'U' 'L']]
```

Figure 10: Policy Iteration -0.8 reward

3.2 Question 3

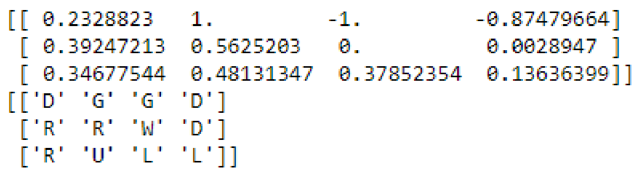
Compare TD-Learning and Q-Learning results with each other. Also with value and policy iterations. Remember that value and policy iteration solve Markov Decision Processes where we know the model (T and Rewards). TD-Learning and Q-Learning are (passive and active, respectively) Reinforcement Learning methods that don't have the model but use data from simulations. Data are simulated from the model we know, but the model is not used in TD or Q-Learning.

In all the grids, TD would avoid going close to the pit(-1) and the nodes next to it as to minimize the risk of falling into it, an example would be in the grid 2 and grid 3, where some nodes had result of 0 since those nodes was next to the pit. While Q would try to explore all the alternatives.



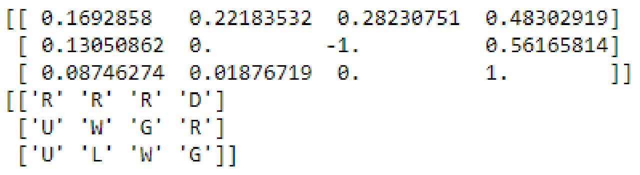
```
[[ 0.42202783  0.52374069  0.57740467  1.        ]
 [ 0.29793356  0.          0.25203179 -1.        ]
 [ 0.10796565 -0.02388345 -0.01950919 -0.96117475]]
[['R' 'R' 'U' 'G']
 ['U' 'W' 'U' 'G']
 ['U' 'L' 'U' 'U']]
```

Figure 11: TD Learning Grid 1



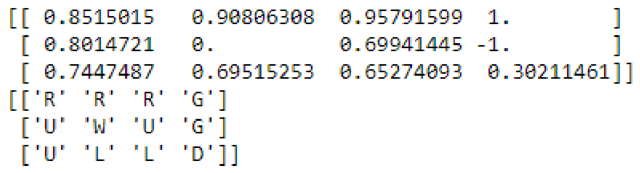
```
[[ 0.2328823   1.         -1.         -0.87479664]
 [ 0.39247213  0.5625203   0.          0.0028947 ]
 [ 0.34677544  0.48131347  0.37852354  0.13636399]]
[['D' 'G' 'G' 'D']
 ['R' 'R' 'W' 'D']
 ['R' 'U' 'L' 'L']]
```

Figure 12: TD Learning Grid 2



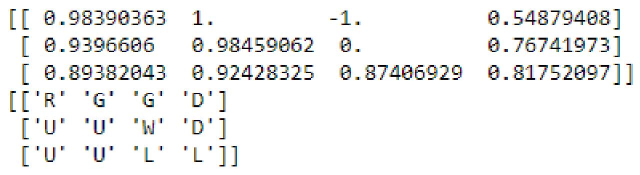
```
[[ 0.1692858   0.22183532  0.28230751  0.48302919]
 [ 0.13050862  0.         -1.          0.56165814]
 [ 0.08746274  0.01876719  0.          1.        ]]
[['R' 'R' 'R' 'D']
 ['U' 'W' 'G' 'R']
 ['U' 'L' 'W' 'G']]
```

Figure 13: TD Learning Grid 3



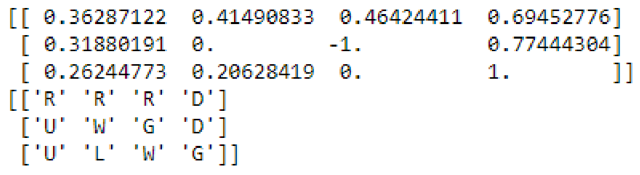
```
[[ 0.8515015   0.90806308  0.95791599  1.        ]
 [ 0.8014721   0.          0.69941445 -1.        ]
 [ 0.7447487   0.69515253  0.65274093  0.30211461]]
[['R' 'R' 'R' 'G']
 ['U' 'W' 'U' 'G']
 ['U' 'L' 'L' 'D']]
```

Figure 14: Q Learning Grid 1



```
[[ 0.98390363  1.         -1.          0.54879408]
 [ 0.9396606   0.98459062  0.          0.76741973]
 [ 0.89382043  0.92428325  0.87406929  0.81752097]]
[['R' 'G' 'G' 'D']
 ['U' 'U' 'W' 'D']
 ['U' 'U' 'L' 'L']]
```

Figure 15: Q Learning Grid 2



```
[[ 0.36287122  0.41490833  0.46424411  0.69452776]
 [ 0.31880191  0.         -1.          0.77444304]
 [ 0.26244773  0.20628419  0.          1.        ]]
[['R' 'R' 'R' 'D']
 ['U' 'W' 'G' 'D']
 ['U' 'L' 'W' 'G']]
```

Figure 16: Q Learning Grid 3

3.3 Question 4

What are the effects of epsilon and alpha values and how they are modified?

Epsilon affects the whether the action should be based on exploration or exploitation. These grids' epsilon value is 0.5 which means 50% is exploring the surrounding nodes and the other 50% is exploiting what it already knows. Could say that Epsilon affects if the action will be random or not. Alpha affects the learning rates of the algorithm. They usually are valued ¡1 as to approach 0 at infinity. These grids, the alpha's value is 0.9 as to let it learn slow while putting it at 0.2 will make it learn faster but at the cost of good paths but since the grid is small it is better to go with lower numbers to go fast because the results would be somewhat the same.

3.4 Question 5

What is the effect of number of episodes?

More number of episodes will give each iteration more attempts to learn what's the optimal path.