Ceng461 HW2 Report

Ali Görkem Yalçın

January 12, 2020

Introduction

I used 0 index for north, 1 index for east, 2 for south, 3 for west.

Value Iteration

Value iteration calculates the expected utility of each state using the utilities of the neighbouring states until the for some iteration count. At the end utility matrix's values is converged, thus the policy. To implement the algorithm I needed the following:
Transitional matrix: This matrix will have the information of probabilities of actually going from one state to another state with the chosen action.
From state i, with an action k, probability of moving to jth state.
It is a very big function to write because except for s5 and s9, there are no identical states with the identical probabilites of going from one state to another(for example, s5 and s9's north state is available and it has the same probabilitiy but s5 has a state in its west side but s4 does not. So the probabilites change for the next states. To write a more generic code, I needed more of the *same* states).
An example can be checked by printing the transitional matrix[:,:,0]. The 0 index is the action index. This print function will print something like this:

0.9	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1	0.8	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0.1	0.8	0.1	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
0.8	0	0	0	0.2	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	0.8	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0
0	0	0	0.8	0	0	0.1	0.1	0	0	0	0	0	0	0	0
0	0	0	0	0.8	0	0	0	0.1	0.1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.1	0.8	0.1	0	0	0	0
0	0	0	0	0	0	0.8	0	0	0.1	0	0.1	0	0	0	0
0	0	0	0	0	0	0	0	0.8	0	0	0	0.1	0.1	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

Which is the probabilities of going from one state to another using the north action.

State reward matrix: This matrix is a simple matrix containing the rewards(r) and the state rewards such as the terminal states and the wall state(5th indexed state)

Then I find the utility matrix by iterating over all the states, finding the utility of that state, updating the utility matrix. This operation is done for N times which is the iteration count.

Then from this utility matrix, I create a policy.

Policy iteration

In the policy iteration algorithm, I used a dummy-all north always-policy to converge the algorithm into creating an optimal policy, same as I got from the value iteration. In policy iteration we pick an initial policy, then we iteratively perform state value determination, which calculates the utility of each state given the current policy, and policy improvement, which updates the current policy if any improvement is possible. The algorithm terminates when the policy stabilizes meaning that the two consecutive state value determinations do not change any values.
I used the transitional matrix here again for the same purpose of calculating the utility of states.
I used the reward matrix in the same way.
I created 2 arrays(u1, u2) to check for differences of utility values between iterations.

Q learning

In the Q learning algorithm, agent discovers the environment by exploring or exploiting the environment depending on the exploration rate.
Depending on the reward agent gets from visiting the state and the previous reward it got from the same state, it updates the Q table.
At the end, for every state's maximum valued action is the policy agent creates from Q learning algorithm.

Experiment results

1) a) Reward = 0
Discount Factor = 1
P = 1
Value Iteration Utility Matrix

10	10	10	1
10.	0	10	10
10.	10	10	10
10.	1	−10	10

Policy created by the value iteration:
> > v 1
v X > >
> > > >
$^1 - 1010$

Policy Iteration Utility Matrix

10	10	10	1
10.	0	10	10
10.	10	10	10
10.	1	−10	10

Policy created by the policy iteration:

> > v 1
v X > >
> > > >
$^1 - 1010$

Since at the converged utility matrix every state offers a reward of 10, policy runs away from every state that has a reward that is less than 10. It also does not go to 15th indexed terminal state with the reward of 10 because it does differentiate between that state and any other state. This is a problem since policy does not go towards the terminal state. Optimal policy would be going south at the 11th indexed state to reach a terminal state.

b) Q learning algorithm for e = 0, a = 0.1, N = 1000

Q table:

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

Best values for each state:

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Policy of the Q table

$\uparrow\uparrow\uparrow 1$

$\uparrow X \uparrow\uparrow$

$\uparrow\uparrow\uparrow\uparrow$

$\uparrow 1 - 1010$

Since exploration rate is 0, agent looks for the best reward in q table to get the maximum reward from exploiting the environment. Since all rewards are 0 no exploiting can be done, increasing N does not help because agent always chose the maximum rewarded state which results in a loop for N times without improving anything.

a. Utility matrix of Value Iteration Algorithm for:

P = 1

Iteration count = 1000

State reward = -0.01

Discount factor = 1

9.94	9.95	9.96	1
9.95.	0	9.97	9.98
9.96.	9.97	9.98	9.99
9.95.	1	-10	10

Policy created by the value iteration:

$> >$ v 1
v X $>$ v
$> > >$ v
${}^{1} - 1010$

Utility matrix of Policy Iteration Algorithm for:
P = 1
State reward = -0.01
Discount factor = 1

9.94	9.95	9.96	1
9.95.	0	9.97	9.98
9.96.	9.97	9.98	9.99
9.95.	1	-10	10

Policy created by the policy iteration:
$> >$ v 1
v X $>$ v
$> > >$ v
$\uparrow 1 - 1010$

Now we have a minus reward for moving around in the environment which results in the agent trying to go to the high rewarded terminal state. This time agent does not randomly roam and avoid the states with a reward less than 10 but now it tries to go to the best rewarded state. Since the reward is not that low, it is not really punishing the agent for roaming around the environment. For example if the reward was something like -5, agent would try to reach the terminal state at all costs since roaming gives a lot of penalty. But not in this parameter set.

Q table for: P = 1 Discount factor = 1 Learning rate = 1/count Exploration probability = 0 Iteration count = 1000

$$\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0. & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0. & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0. & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0. & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array}$$

Best values for each state:

$$\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0. & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array}$$

Policy of the Q table

$\uparrow\uparrow\uparrow 1$
$\uparrow X \uparrow\uparrow$
$\uparrow\uparrow\uparrow\uparrow$
$\uparrow\ 1 - 1010$

Q learning still did not converge to the optimal policy since the agent still can not explore the environment but try to exploit with a non existent data set.
b)

Utility matrix of Value Iteration Algorithm for:
P = 1
Iteration count = 1000
State reward = -0.01
Discount factor = 0.2

$$\begin{array}{cccc}
-0.0044 & 0.028 & 0.19 & 1 \\
-0.0044. & 0 & 0.0676 & 0.388 \\
0.028. & 0.19 & 0.388 & 1.99 \\
0.19. & 1 & -10 & 10
\end{array}$$

Policy created by the value iteration:

$> > > 1$
v X $>$ v
$>$ v $>$ v
$>$ 1 -10 10

Utility matrix of Policy Iteration Algorithm for: P = 1 State reward = -0.01 Discount factor = 0.2

$$\begin{array}{cccc}
-0.0044 & 0.028 & 0.19 & 1 \\
-0.0044. & 0 & 0.0676 & 0.388 \\
0.028. & 0.19 & 0.388 & 1.99 \\
0.19. & 1 & -10 & 10
\end{array}$$

Policy created by the policy iteration:

```
> > > 1
v X > v
> v > v
> 1 -10 10
```

Discount factor represents the wanting the short-term reward over long-term rewards. So there are 3 policy changes from the solution before. At state2, policy says to go to(at least try to) terminal state with a reward of 1. The other 2 changes are for the same reward. This means that trying to go for the reward of 10 is less appealing than getting the reward of 1. Since the distance between s2 and the last terminal state is at least 5, discount factor removes the advantage of 10, thus the policy converges to getting the short term reward.

c) Utility matrix of Value Iteration Algorithm for:
P = 1
Iteration count = 1000
State reward = 5
Discount factor = 1

$$\begin{array}{cccc}
9.9950e+03 & 1.0000e+04 & 1.0005e+04 & 1.0000e+00 \\
10000. & 0 & 10010 & 10015 \\
10005. & 10010 & 10015 & 10020 \\
1.001e+04. & 1.000e+00 & -1.000e+01 & 1.000e+01
\end{array}$$

Policy created by the value iteration:

```
> > v 1
v X > v
> > > >
< 1 -10 10
```

Utility matrix of Policy Iteration Algorithm for:
P = 1
State reward = 5
Discount factor = 1

Utility matrix did not converge, matrix's values increased with every iteration.

So in this parameters set, visiting each state will give the agent + rewards so agent tries to not reach the terminal states. The only reason why value iteration found an *optimal* policy is that there is a iteration count. The policy runs away from the terminal states. Policy iteration did not even converge because every iteration increased the values of utility matrix, thus the algorithm ran once again, then values increased again etc..

d) Utility matrix of Value Iteration Algorithm for:
P = 0.5
Iteration count = 1000
State reward = -0.01
Discount factor = 1

$$\begin{array}{cccc}
9.692 & 9.722 & 9.742 & 1 \\
9.672. & 0 & 9.822 & 9.8772 \\
9.6832. & 9.7288 & 9.8144 & 9.9248 \\
9.6432 & 1 & -10 & 10
\end{array}$$

Policy created by the value iteration:
$>$ $>$ v 1
$\uparrow X > v$
$>>> v$
$\uparrow$ $1 - 1010$

Utility matrix of Policy Iteration Algorithm for:
P = 0.5
State reward = -0.01
Discount factor = 1

$$\begin{array}{cccc}
9.692 & 9.722 & 9.742 & 1 \\
9.672. & 0 & 9.822 & 9.8772 \\
9.6832. & 9.7288 & 9.8144 & 9.9248 \\
9.6432 & 1 & -10 & 10
\end{array}$$

Policy created by the policy iteration:
$>$ $>$ v 1
$\uparrow X > v$
$>>> v$
$\uparrow$ $1 - 1010$

So now agent does not care about long term reward or short term reward, which is better, agent will go for that. Reward for each state is -0.01 so agent will make moves with more carefully. But the stochasticism is increased with the p being 0.5. Now half of the time agent goes to an unwanted state. But the important part is that the discount factor is now 1. Because of that agent always tries to reach the maximum rewarded state. And as a result the utility matrix is filled with higher numbers.
e) Utility matrix of Value Iteration Algorithm for:
P = 0.8
Iteration count = 1000

State reward = -0.01
Discount factor = 0.9

3.75177637	4.32661808	4.94142615	1
3.43279424	0	6.2111535	7.3044334
3.9234601.	4.63501623	5.7470344	8.46948692
3.19218821	1	-10	10

Policy created by the value iteration:
$>$ $>$ v 1
v X $>$ v
$>$ $>$ $>$ v
$^1 - 1010$

Utility matrix of Policy Iteration Algorithm for:
P = 0.8
State reward = -0.01
Discount factor = 0.9

3.75177637	4.32661808	4.94142615	1
3.43279424	0	6.2111535	7.3044334
3.9234601.	4.63501623	5.7470344	8.46948692
3.19218821	1	-10	10

Policy created by the policy iteration:
$>$ $>$ v 1
v X $>$ v
$>$ $>$ $>$ v
$^1 - 1010$

Now the agent has a *normal* discount factor, not much but a punishment for roaming around the environment and most of the time agent goes to the place it wants.

i. Q table for:
P = 0.8
Discount factor = 0.9
Learning rate = 0.1

Exploration probability = 0
Iteration count = 1000

−0.0030349	−0.0029701	−0.002881	−0.00199]
−0.0039404	−0.0041851	−0.00385219	−0.00385219]
−0.00220951	0.0991	−0.0011791	−0.00125929]
0.	0.	0.	0.
−0.0029701	−0.00305831	−0.00199	−0.00199
0.	0.	0.	0.
−0.001	0.55041435	0.	0.
0.85788282	0.	0.	0.
−0.0036829	−0.0042661	0.13299775	−0.0039404
−0.0021439	0.36407065	−0.001	−0.001171
0.2887685	−0.001	−1.	−0.9809149
0.04437109	0.	0.	0.
0.22006368	−0.00109	−0.001	−0.0011791
0.	0.	0.	0.
0.	0.	0.	0.
0.	0.	0.	0.

Best values for each state:

−0.00199	−0.00385219	0.0991	0.
−0.00199	0.	0.55041435	0.85788282
0.13299775	0.36407065	0.2887685	0.04437109
0.22006368	0.	0.	0.

Policy of the Q table
< v > 1
v X > ↑
v >↑↑
↑ −1010

So now the exploration probability is 0 so the agent always exploits the environment by taking the maximum values from the q table. But the problem is that the iteration count, learning rate and the exploration rate is so low that agent fails to come up with a good policy. Iteration count and learning rate is so low that even if in the small amount of actions that the agent finds a good reward, it does not learn much from it.

ii) Q table for:
P = 0.8
Discount factor = 0.9
Learning rate = 0.1
Exploration probability = 0
Iteration count = 10000

−0.0030349	−0.0029701	−0.002881	−0.00199
−0.0039404	−0.0041851	−0.00385219	−0.00385219
−0.00220951	0.96120975	−0.0011791	−0.00125929
0.	0.	0.	0.
−0.0029701	−0.00305831	−0.00199	−0.00199
0.	0.	0.0.	
−0.001	0.86046194	0.	0.
0.9815956	0.	0.	0.
−0.0036829	−0.0042661	0.41075574	−0.0039404
−0.0021439	0.6898233	−0.001	−0.001171
0.72841637	−0.001	−1.	−0.9809149
0.8047888	0.	0.	0.
0.35300288	−0.00109	−0.001	−0.0011791
0.	0.	0.	0.
0.	0.	0	.0.
0.	0.	0.	0.

Best values for each state:

−0.00199	−0.00385219	0.96120975	0.
−0.00199	0.	0.86046194	0.9815956
0.41075574	0.6898233	0.72841637	0.8047888
0.35300288	0.	0.	0.

Policy of the Q table
$<$ v $>$ 1
v X $>$ $\uparrow$
$v >\uparrow\uparrow$
$\uparrow$ 1 $-$ 1010

Now the iteration rate is increased an the values of the utility matrix increased in average. The policy is still not optimal because agent does not learn much from going to the terminal states and still agent can not explore the environment. It tries to exploit the environment without knowing everything in the environment.

iii) Q table for:
P = 0.8
Discount factor = 0.9
Learning rate = 0.1
Exploration probability = 0.1
Iteration count = 10000

$3.16854824e-02$	$-2.09368000e-03$	$3.26048092e-01$	$-1.99000000e-03$
$9.86606657e-02$	$8.36948281e-02$	$-1.00000000e-03$	$-1.93877479e-03$
$-1.00000000e-03$	$9.64318108e-01$	$9.75064744e-02$	$8.42675290e-03$
$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$
$4.55474457e-02$	$2.85664699e-01$	$7.15036356e-01$	$2.24769596e-01$
$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$
$3.85287494e-01$	$3.75405418e-01$	$4.45671573e-01$	$5.39453723e-01$
$-1.00000000e-03$	$-1.00000000e-03$	$1.69636980e+00$	$0.00000000e+00$
$6.68095578e-01$	$8.21243538e-01$	$8.67868134e-01$	$7.58475812e-01$
$6.13419486e-01$	$6.77202709e-01$	$9.87782776e-01$	$7.44794973e-01$
$6.44647784e-01$	$-4.84925454e-02$	$-2.46527549e+00$	$-4.11048485e-01$
$-1.97525978e-03$	$-1.00000000e-03$	$7.79351817e+00$	$4.28766695e-02$
$7.76944682e-01$	$9.80960892e-01$	$8.76548192e-01$	$8.37434943e-01$
$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$
$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$
$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$	$0.00000000e+00$

Best values for each state:

0.32604809	0.09866067	0.96431811	0.
0.71503636	0.	0.53945372	1.6963698
0.86786813	0.98778278	0.64464778	7.79351817
0.98096089	0.	0.	0.

Policy of the Q table
v $\uparrow > 1$
$vX < v$
$vv \uparrow v$
$> 1 - 1010$

Now since the agent has some exploration ability, it finds more rewards in the unvisited states and updates the q table with those values more and if it is not exploring new states, it is exploiting the statees it visisted before. Now we can see some convergence in going to the 10 rewarded states and running away from the -10 rewarded states. This is still not optimal because still learning rate is not in the place we want it to be.
iv) Q table for:
P = 0.8
Discount factor = 0.9
Learning rate = 1
Exploration probability = 0.1
Iteration count = 10000

0.42612659	0.791	0.4845851	0.42612659
0.549539	0.89	0.62171	0.549539
0.62171	0.89	7.2629	0.62171
0.	0.	0.	0.
0.4845851	0.70564108	0.79515675	0.79515675
0.	0.	0.	0.
3.42165225	7.2629	7.2629	0.89
1.	4.24771882	8.99	7.2629
0.70564108	5.863949	0.89	0.89
2.46728449	6.52661	1.	1.
7.2629	$-10.$	5.863949	3.06948702
2.46728449	7.2629	10.	6.52661
0.62171	1.	0.89461861	0.89461861
0.	0.	0.	0.
0.	0.	0.	0.
0.	0.	0.	0.

Best values for each state:

0.791	0.89	7.2629	0.
0.79515675	0.	7.2629	8.99
5.863949	6.52661	7.2629	10.
1.	0.	0.	0.

Policy of the Q table
$>>$ v 1
v X $>$ v
$>>\uparrow v$
$> 1 - 1010$

Now we are getting close to the optimal policy. But we increased the learning rate to 1 which means the agent is learning too much from going to the states. It learns that last terminal state has a 10 reward and it updates the value accordingly in the q table with a 10 for that state,action pair. Since it learns too quickly, it knows to run away from -10 and go to the 10 rewarded states. It also runs away from the top right 1 rewarded state because it knows that the 10 is nearby. Also it goes to the 1 for the bottom left because it knows that -10 is nearby and it is decreasing the reward of the nearby states so the agent wants to go to the 1 rather than falling to the -10 pit. This is still not optimal because the learning rate is too much. Agent always learns new information but it does not value the past information that much. Information goes from one ear and out the other.

v) Q table for:
P = 0.8
Discount factor = 0.9
Learning rate = 1/count
Exploration probability = 0.1

Iteration count = 10000

0.25929945	0.42741973	0.	−0.01
0.40182071	0.75805485	0.29353297	0.18990595
0.495	0.99225815	0.34163883	0.3259666
0.	0.	0.	0.
0.25601429	0.31195408	0.69205974	0.32241673
0.	0.	0.	0.
0.41657075	5.12702282	0.96179672	3.1371023
2.076913	4.1631201	7.15826114	3.56111179
0.53513803	0.76549919	0.84241613	0.71406904
0.79574569	1.83617238	0.99597571	0.80553503
2.83059137	−4.48852899	−7.82731717	−0.66293652
0.26368974	5.68406552	8.82310996	4.45848425
0.78187302	0.95850154	0.86396192	0.83859978
0.	0.	0.	0.
0.	0.	0.	0.
0.	0.	0.	0.

Best values for each state:

0.42741973	0.75805485	0.99225815	0.
0.69205974	0.	5.12702282	7.15826114
0.84241613	1.83617238	2.83059137	8.82310996
0.95850154	0.	0.	0.

Policy of the Q table
$> > > 1$
v X $>$ v
v $> \uparrow v$
$> 1 - 1010$

This 1/count learning rate makes the utility matrix - policy much better looking because agent now values the past information more than the current information in a harmonic series way. As a result agent still learns more but the agent does not trust the new information as it sees them. Agent needs more proof that the new information is true before the q table is converged to the new information. Policy is much better looking from where we started. But this is still not converged to the optimal policy because the iterating more with this parameter set still improves the policy.

vi) Q table for:
P = 0.8
Discount factor = 0.9
Learning rate = 1/count
Exploration probability = 0.1
Iteration count = 100000

```
0.36716895   1.39498842       0.          −0.01
0.40691144   3.23508689   1.55040863   0.61138458
3.14896824   1.13706009   4.79345501   2.16635405
    0.           0.           0.           0.
0.74250244   1.62411576   2.58038515   1.90534535
    0.           0.           0.           0.
4.04765917   6.40827766   3.20977516    5.139142
1.92742139   6.24002509   7.88133415   5.28452536
1.87584449   3.60146553   0.97486906   2.29968511
3.96527713   4.84494058   1.29628384    2.7577377
5.00836171   6.24930313  −6.78800261    2.359864
6.91899743   8.34619153   9.37190689   5.96789675
2.60134865   0.98992948   1.17592227   1.26011547
    0.           0.           0.           0.
    0.           0.           0.           0.
    0.           0.           0.           0.
```

Best values for each state:

```
1.39498842   3.23508689   4.79345501       0.
2.58038515       0.       6.40827766   7.88133415
3.60146553   4.84494058   6.24930313   9.37190689
2.60134865       0.           0.           0.
```

Policy of the Q table

```
> > v 1
v X > v
> > > v
↑ 1 − 1010
```

Now this parameter set is the best of the q learning parameter set we tested before. It allows for exploration, it lets agent learn new information whilist holding the past information valuable, it values short term rewards more but not in the way of declining long term rewards and it has a high count of iterations which makes the algorithm converge much better. This parameter creates the optimal policy we got from the VI/PI algorithm

vii) I managed to get to the optimal policy in 39850 iterations with the following parameter set
learning rate = 1/count
e = 0.1
iteration count = 39850