LABREPORT

David van Balen, Joris van Gool, Daan van Laar
5513588 4270126 5518741

1.

a

ValueIterationAgents assign values to states, whereas qlearning agents assign values to combinations of a state and an action.

b

In reinforcement learning, an agent has to experimentally find out how good actions are. If these methods would be available, it would be possible to make an offline planning based on the results, and the entire learning would be unnecessary

c

When a state is left, you get the reward for that state. The living reward is only earned if you don't earn another reward, which makes sense because it would be a constant reduction anyway, so you might as well combine it in the reward by math. A terminal state rewards 0 when it is left.
The rewards earned are only dependent on the state that is left by convention.

2.

An online planner makes choices while the actions are being executed, and uses feedback from previous actions to choose the next action. An offline planner plans ahead, and does not depend on realtime feedback.

3.

a

A terminal state.

b

Finite problem

c

It means that future rewards must always outweigh the living reward because there is a possibility to end the getting of rewards if that is more favourable.

4

a

with noise = 0.2:

Discount 0 - failure
discount 0.2 - failure
discount 0.4 - failure
discount 0.6 - failure
discount 0.8 - failure

with discount = 0.8:

noise 0 - yay
noise 0.2 - failure
noise 0.4 - failure
noise 0.6 - failure

noise 0.8 - failure

b
We changed the noise to 0

c
We did this because when the noise is 0, the agent won't accidentally fall off the bridge.
This made the agent cross the bridge, because it didn't accidentally fall off.

5
a
- (1, 0.001, -5)
- (0.5, 0.3, -1)
- (1, 0.001, -0.1)
- (1, 0.3, -1)
- (42, 0, 42)

b
-The first one works because it has relatively low noise, so taking a risky part pays off.
The emmidiate reward causes the AI to try and finish the level as fast as possible,
without falling off the cliff.
-The second one has quite a bit of noise and a high discount rate, the AI takes the safe path
to the closest reward.
-The third: Because of the low noise, no discount and midly negative reward the AI takes the
dangerous path to the furthest away reward.
-The fourth: because of the somewhat present noise the safe path will be taken, the very mellow
living
reward together with the discount rate cause the long route to be more rewarding.
-The fifth: 42 is the answer to everything and therefore it is correct.
No noise combined with a positive living reward and a >1 discount causes the AI to never end.
42 is the answer to everything therefore it is correct.

Exercise 6
a.
After starting up crawler.py using the console, we changed the value of alpha as quickly as we could. With low values of alpha, we observed that the crawler wasn't really making any advances. When we chose high values of alpha, we observed a quick learning process resulting in the crawler quickly learning to crawl. This is because the new measurements are worth a lot and the new measurements typically are better since the crawler has learned what good actions are. In this case, a high learning rate results in a quick good crawler.
b.
A low value of epsilon means that the crawler doesn't explore enough states and just doesn't know what the best strategy is. In our case, the crawler only alternated between 2 states and didn't make any progress at all. However, if we choose epsilon too high, the crawler will only do random actions.
c.
We experimented with quite a few different combinations of alpha and epsilon, but we didn't really find any new results compared to the results of question a and b.
d.

If the discount rate is really high, it will definitely have an effect on the effect of alpha, because the new results have a discounted reward. It does not have any effect on epsilon.

Exercise 7
a.
No, this does not return an optimal policy. This is because 50 training episodes is not nearly enough to learn the entire map. There are plenty of states that have a very slim chance of being visited and 50 training episodes doesn't guarantee all those states being visited. That means that not all states have gotten a value, and therefore the found policy can't assure optimality.
b.
With high values of epsilon, the states that are close to the starting state have a far greater chance of being visited than states further away. A low value of epsilon means that the up till then optimal policy will be followed most often, which means exiting the bridge on the left and not knowing what is on the right side.
c.
The only thing really affected by the changing of the value of alpha is the amount in the states. The ratio between states stays the same, but the actual number depends on the value of alpha
d.
After experimenting with different values, we have found that no combination finds an optimal policy for this problem within 50 training episodes. It is just too little for this problem.

Exercise 8
a.
We have experimented with quite a few different values and found that the values that are close to (0.05, 0.8, 0.2) are the most optimal for Pacman on the smallGrid problem. Other values showed a win:loss ratio which was lower than the win:loss ratio of (0.05, 0.8, 0.2)
b.
In the states, there are values of the x-coordinate, y-coordinate and a few other values like the distance to the nearest food and how many ghosts are 1 space away. Pacman uses these to find the optimal strategy to win. The rewards are what influence the score, like eating a piece of food gives a positive effect on the score, but just walking endlessly or getting eaten is a negative influence.
c.
Training Games: 2000
Learning Parameters: (0.05, 0.8, 0.2)
Average Reward: -67.93
Test Games: 10
Win Rate: 1.0
Average Score: 501.0
These are great results for Pacman. The only flaw in this is that Pacman doesn't really know what to do when there is no direct path to a food and the ghost is far away. We think this is mainly due to the fact that in the state, there is only an entry for how many ghosts there are 1 space away.

Exercise 9
a.

The problem with not describing states as features is that two very very similar states could be different in the eye of Pacman, while the actions that it would take are problably the same. This means that Pacman's learing would be very ineffecient and therefore feature based representations for a state are more beneficial for Pacman and Q-learing in general.

b.

The IdentityExtractor returns 1 for all combination of state and actions

The CoordinateExtractor gives a value of 1 for the coordinates and action of a (state, action) pair.

The SimpleExtractor gets a feature for the number of ghosts that are 1 step away, it also has a feature for if it is eating food, then there is also a feature describing where the nearest food is.

c.

PacmanQAgent:
Average Reward All: -76.17
Average Reward Last: 285.83
Average Score: 498.3
Win Rate: 1.00

ApproximateQAgent:
Average Reward All: -92.96
Average Reward Last: 305.62
Average Score: 500.6
Win Rate: 1.00

d.

MediumGrid:
Average Score: 527.8
Win Rate: 1.00

MediumClassic
Average Score: 1325.1
Win Rate: 1.00

10.

```
Provisional grades
==================
Question q1: 6/6
Question q2: 1/1
Question q3: 5/5
Question q4: 5/5
Question q5: 3/3
Question q6: 1/1
Question q7: 1/1
Question q8: 3/3
------------------
Total: 25/25
```