Reinforcement Learning Assignment 3

Student Number:13064947 - John Goodacre

March 18, 2018

1 Basic Tabular Learning

Apologies for the duration of this assignment. However, it is actually only a few pages of prose and I simply did my best to fit in all the charts needed, I felt that putting in less would weaken my answers. Other charts may be found in the accompanying Jupyter notebook.

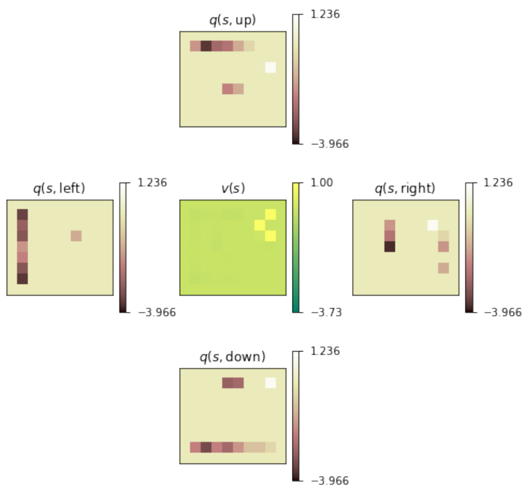
1.1 Why is the Experience Replay agent so much more data efficient than online Q-learning?

The action and state values for online Q-learning and Experience replay are shown in fig 1. The populated q values show that the Experience replay algo has learned much more. This makes logical sense.

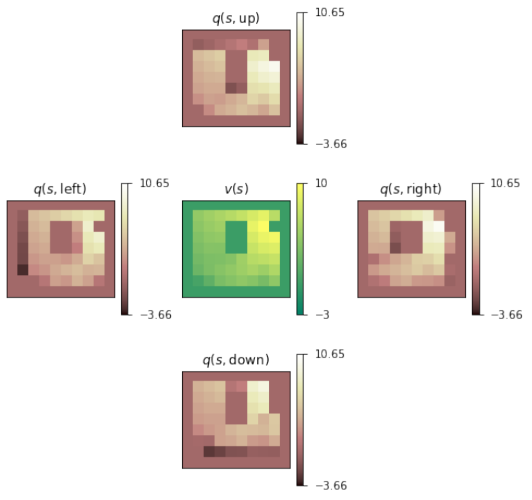
Although they may be grabbing the same amount of data (1e3 steps), the difference lies in the fact that the q-values for online learning are updated once at each step. Whereas the Experience replay algo also updates its q-values each step, but on top of this uses its replay buffer to replay its experiences and for each of these also updates its q-values. In this case each step of the Experience replay algorithm replays 30 times. It is simply updating its q-values many more times with the same data. This is clearly more data efficient in the stationary, deterministic environment as given.

1.2 If we run for number of updates rather than steps, which among online Q-learning and Experience Replay is better?

The action values for online Q-learning (3e4 steps) and experience replay with (1e3 steps but 30 offline updates), are given in fig 2. The results are more subtle as both algorithms have learned a reasonable set of action values. However the online algorithm is further ahead. The key difference in this case is that although the q-values have been updated a similar number of times, the online Q-learning algorithm has had more unique experiences and explored more of the environment. The experience replay algorithm will be more sure of the part



(a) Online Q learning 1e3



(b) Experience Replay - 1e3, 30 offline

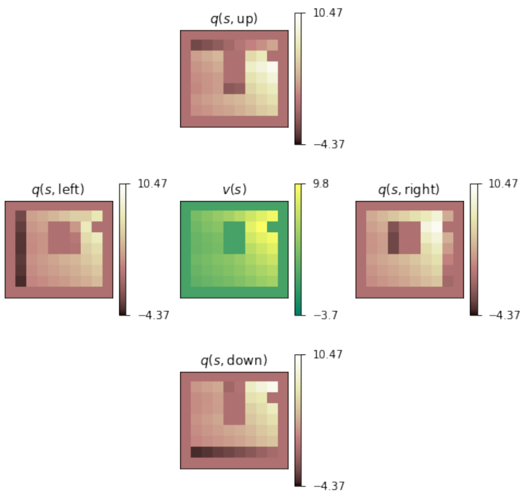
Figure 1: Tabular Online Q v Experience Replay, action values

of the environment it has explored (due to repetitive offline updates), but has far less 'real' experience.

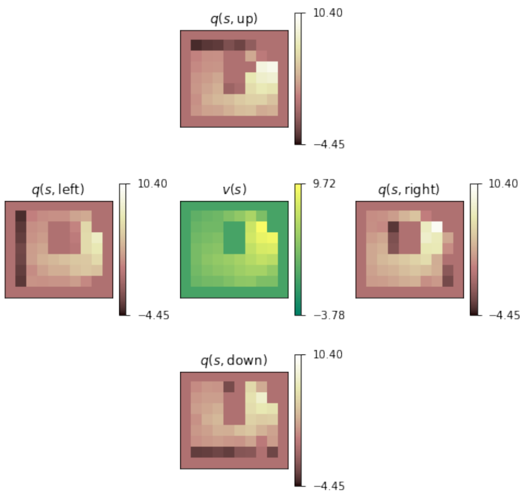
To illustrate this further I have included the learned greedy policies for both in fig 3. For experience replay, there are still a couple of states where the greedy policy is still sub-optimal (for example the top right state and second state on the top row).

1.3 Which among online Q-learning and Dyna-Q is more data efficient?

Fig 4, shows the learned action values for online Q-learning against Dyna-Q. Again, similar to the results with experience replay the Dyna-Q algorithm is more data efficient. The difference with Dyna-Q is that in this case we are 'learning' a model of the environment. So although we are replaying in a similar manner to experience replay. We are not directly replaying sampled experiences, but generating predicted rewards and next states given sampled states and actions. In this particular setting we have a tabular model and 'learning' comprises simply populating that model and indeed the setting is stationary and deterministic (so the differences with DynaQ and Experience replay are not as stark as they could be in other settings). But anyway Dyna-Q is more data efficient because the q-values are being updated a greater number of times but this time from generated model experiences.

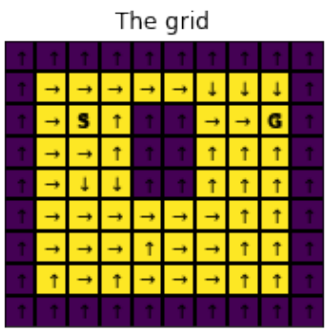


(a) Online Q-learning 3e4

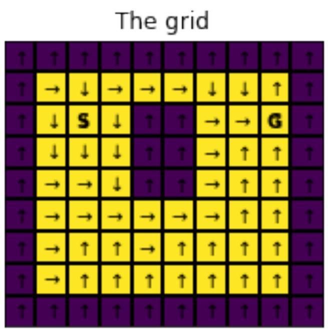


(b) Experience Replay 1e3, 30 offline

Figure 2: Comparison of updates Online Q v Experience Replay

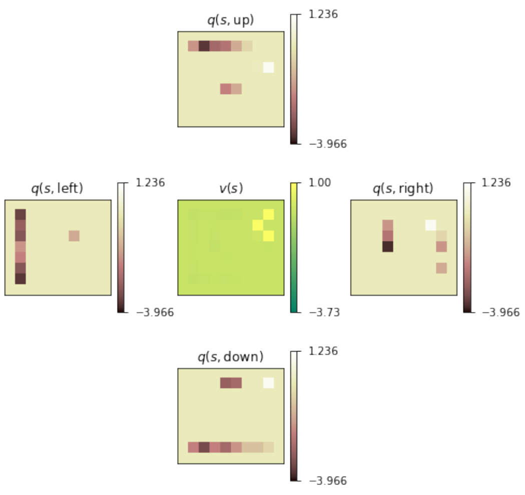


(a) Online Q-learning 3e4 - Greedy Policy

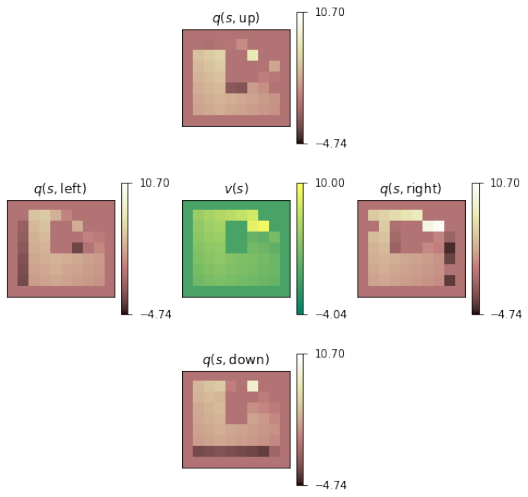


(b) Experience Replay 1e3, 30 offline - Greedy

Figure 3: Greedy Policy Online Q v Experience Replay (update comparison)



(a) Online Q learning 1e3

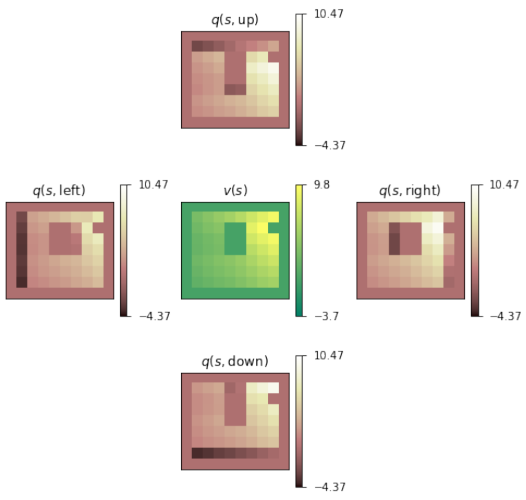


(b) DynaQ 1e3, 30 offline

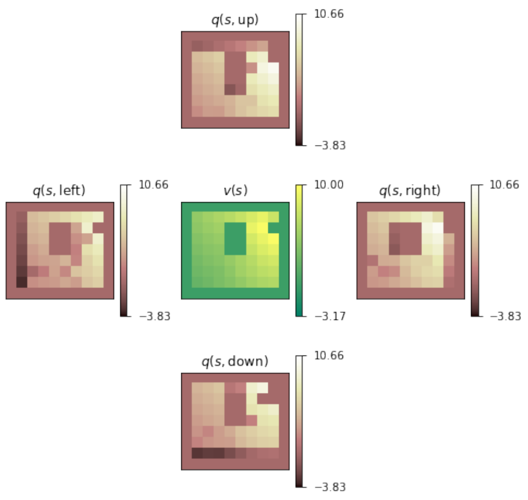
Figure 4: Data Efficiency - Online Q v DynaQ

1.4 If we run for number of updates rather than steps, which among online Q-learning and Dyna-Q is better?

Figures 5, 6, show the learned action values and learned greedy policies for online Q and DynaQ respectively. In this case both models have a similar number of updates but for online Q these updates come from direct experience with the environment, whereas for DynaQ it has less direct experience but also uses experience replay on its learned model of the environment. The online Q algorithm is better, it has more unique direct experience with the environment, whereas most of the DynaQ experience is from its learned model. From fig 6, we can see that for DynaQ there are still states in the environment where the greedy policy is still not optimal for DynaQ. The arguments are similar to experience replay, just with a learned model in this case.

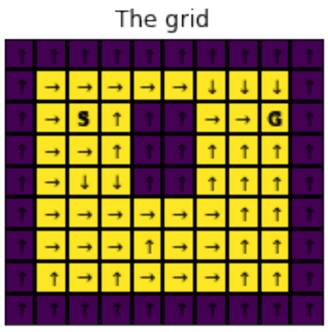


(a) Online Q-learning 3e4

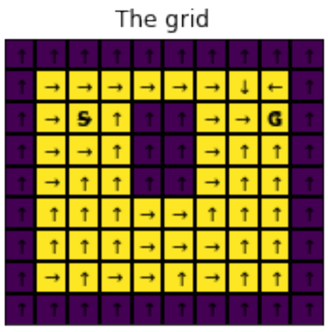


(b) DynaQ 1e3, 30 offline

Figure 5: Comparison of updates Online Q v DynaQ



(a) Online Q-learning 3e4 - Greedy Policy



(b) DynaQ 1e3, 30 offline - Greedy

Figure 6: Greedy Policy Online Q v DynaQ (update comparison)

2 Linear function approximation

2.1 The value estimates with function approximation are considerably more blurry, why?

Figure 7 illustrates the point the question is making. We can clearly see that the value estimates are more blurry (in this case I compared online Q against DynaQ with linear function approximation).

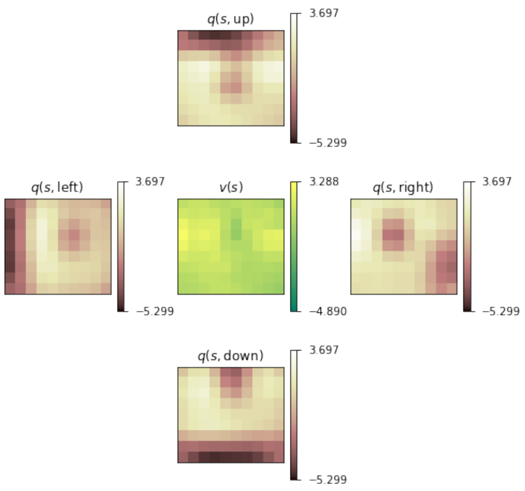
Here we have a function approximation of the q values. In most real world cases we will simply be unable to tabulate and learn the value of each state, action pair. Thus, we can instead use 'features'. These features give a lower dimensional representation of the environment. In the linear case each state can be represented as a vector of features and the state value can be calculated by a weighted sum of these features with a learned vector of weights. Clearly when we extend to q-values we also include action values. (Thus, for clarity the dot product of our 'Qtheta' weights, (now a matrix of dimension (features x actions)) with a state will now give estimated values for each of our actions).

The blurryness is natural. We have a lower dimensional representation and have effectively smoothed across states (in this case linearly), similar states will have similar feature representations (depending upon our features, I have also tried to answer this generically without talking about the particular feature representation given), thus this representation will for example find it more arduous to learn narrow passages around barriers (where hitting a barrier gives a negative penalty) as can be seen in fig 7

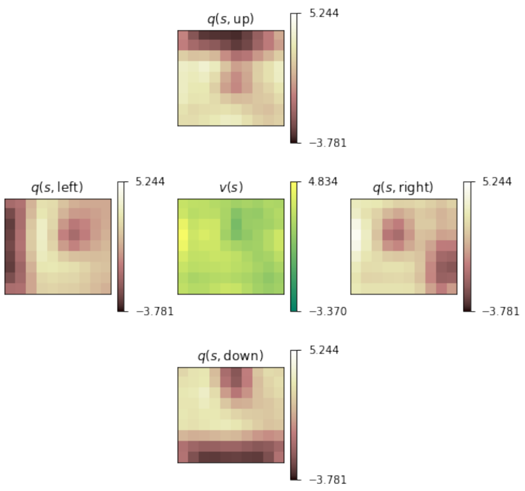
2.2 Inspect the policies with linear function approximation. How does it compare to optimal? Inconsistencies?

Figure 8, shows the learned greedy policies for Online-Q, experience replay and DynaQ using function approximation. None of them are yet optimal, but there are also differences. I highlighted above the potential difficulties with this representation in learning a path through a narrow passageway and fig 8 shows that only the experience replay algo has learned to do so (in this particular random run of the code). Also online and DynaQ have a initial strong tendency to move away from walls, even to the extent of moving away from the final goal.

In this particular implementation the key differences between the algos (bar the specifics about linear representations I mentioned above) are that online simply updates its q-value function approximation parameters each step (akin to stochastic gradient ascent). Experience replay does the same but gains more model updates and confidence through its replays. DynaQ now has an extra layer of abstraction in that not only is it replaying its experience from its learned



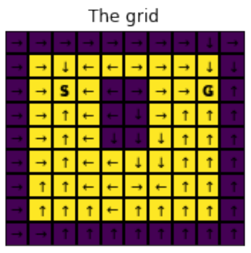
(a) Online Q-learning 1e5 - Linear function approx



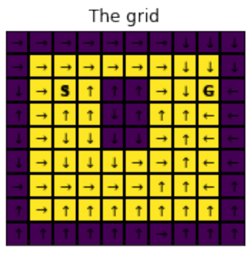
(b) DynaQ 1e5, 10 offline - Linear function approx

Figure 7: Linear function action values Online Q v DynaQ

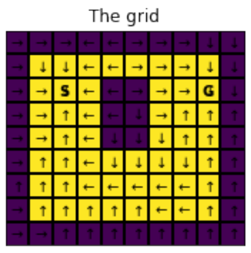
model as before, but that unlike the deterministic tabular case, this is a genuinely a learned model. The rewards, transition matrices and discounts are all learned through gradient descent. Thus, the learning rate will also significantly alter results in this case. Given the thrust of the question is about the linear function approximation I highlight my earlier answer about a lower dimensional 'smoothing' of states and indeed the results are completely dependent on the feature representation chosen, a bad one will give perverse results. (Again I have answered this generically rather than digging into the feature representation given, but given the number of training examples some of the behaviours through the narrow passageway and at the bottom of the maze look a little perverse and if I dug deeper, would examine if I could improve the features before deciding it is totally the fault of the linear set up).



(a) Linear Online Q 1e5



(b) ExpRep 1e5, 10 offline



(c) DynaQ 1e5, 10 offline

Figure 8: Greedy Policies, Linear - Online Q v ExpR v DynaQ

3 Non-Stationary

3.1 Compare Q-learning and Experience Replay, explain

We can see in figure 9, the value functions and greedy strategies learned in the non-stationary environment, for online Q and experience replay. The target has been moved during training to the bottom left of the maze. From the state value function alone it is a little difficult to see the impact of changing the target part way through, but one can see that online Q adapts quicker to the changed situation. This can be seen from a couple of brighter spots in the value function near the new goal and particularly from the greedy strategy, where states near the goal have found a greedy strategy moving to the new goal (in contrast to experience replay where almost all greedy actions continue to move away from the new goal).

The logic behind this is simply that experience replay gains online experience but also replays that experience multiple times. Thus, it increases confidence more quickly about existing states, however, this is a disadvantage in a non-stationary environment as it will be slower to adapt. Indeed we are sampling uniformly over ALL previous experiences (not the most recent), so until the experience data from the new target dominates the previous goal, there will be many cases where experience replay is simply misleading. The online Q algorithm does better. Its experience is only gained through online exploration (not replaying it).

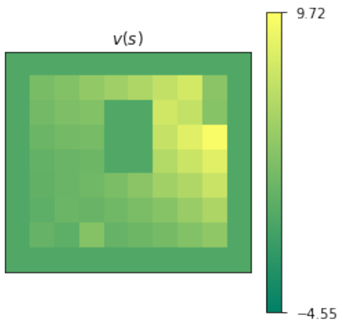
Note also in this case the interpretation of the tabular model itself has an impact. The simplest interpretation of 'learning' the model here is simply exploring and populating a look-up table with the latest values. If instead we interpreted learning to be taking an average over experiences, then although in an unchanging environment this would be equivalent, in a non-stationary environment, the learned model of the environment would clearly be slower to adapt to the change.

3.2 Compare Dyna-Q and Experience Replay, explain

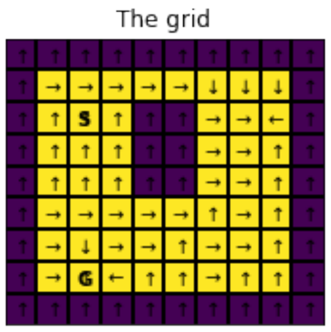
In figure 9, I have also shown a similar comparison of online Q learning with DynaQ. Here we see that DynaQ is better than online Q-learning and adapts more quickly. Indeed the greedy policy from start to goal is already optimal over the changed environment.

The difference is that although DynaQ also uses experience replay. First this experience is generated by its own learned model of the environment, and second this learned model is only updated by real experiences in an online fashion (the q-values are of course both updated online and from experience replay generated by this model). So although we are also sampling over previous states, unlike experience replay when the environment is non-stationary the uniform sampling is less likely to give misleading old examples of reward and state transitions, due to the fact that the experience replays are now being generated by a model which itself is being updated towards the new environment. Thus in this case it offers the advantages of online Q learning, but with the advantage of using replay (from the model) to gain faster confidence over existing states.

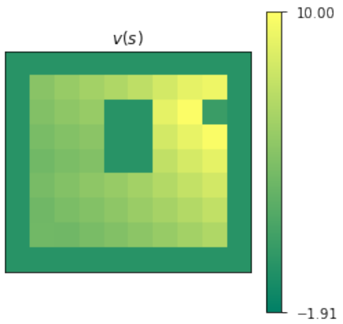
Note that if one chose not to sample uniformly in the previous experience replay example (thus including lots of 'bad' data), but using only recent experience. Then one could improve the performance towards DynaQ, but as it stands DynaQ is better in this setup for the reasons given above.



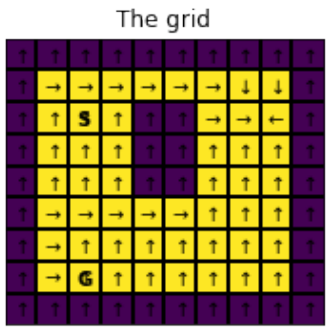
(a) Online Q-learning, Non-stat



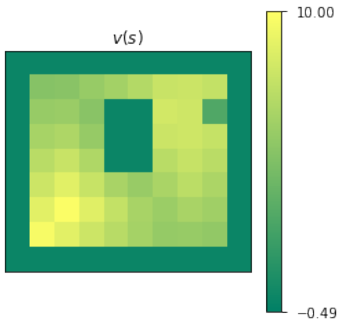
(b) Online Q, Non-stat - Greedy



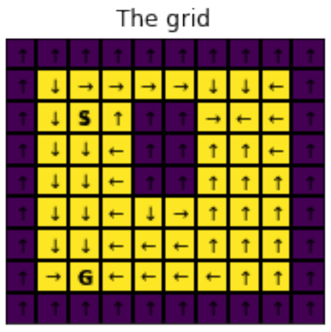
(c) Experience replay, Non-stat



(d) Exp Rep, Non-stat - Greedy



(e) DynaQ, Non-stat



(f) DynaQ, Non-stat- Greedy

Figure 9: Online Q v ExpRep v DynaQ, Non-stationary environment