Reinforcement Learning

Feedback on the 1. Assignment

The marks for the first RL assignment have been returned last week. Here are now some comments. I was very impressed by the very good reports I have had the pleasure to read. Almost all reports showed a good understanding of the algorithms and most of you proved able to demonstrate your understanding in an accessible and very convincing form. Well done.

Intuition and the incorporation of background knowledge (Q1) and the use of visualisation (in Q2 and Q3) are very important when attempting at an efficient use of RL algorithms. A good intuition is also needed to choose appropriate quantities to be visualised. In the present problem it is of course useful to look at the paths of the taxi in same example trials, because it is quite obvious whether the taxi takes most of the actions correctly (Q2.2), which was done nicely in a few of your reports. An equally good idea about the performance can be obtained from the mean reward – at least if the optimal mean reward is known. To find the optimal mean reward (see below on Q2.3) was not easy here and is often not possible, but in many cases we have an idea what level of reward we can consider as acceptable, i.e. we would choose a good-enough solution instead of an optimal one.

Q1: What is a “time horizon”? A horizon is how far you can look, not how far you can go. A time horizon is therefore the number of time steps the algorithm should look ahead in order to get a good reward. The given reward structure was such that the agent could but did not have to take into account the full episode, i.e. looking ahead only to the pick-up works just as well as looking ahead from the initial position of the taxi beyond the pick-up to drop-off. An interesting variant of the problem might have been to always reward the pick-up location with a small reward even after the passenger has been picked up already such that the taxi has to decide whether to stay at the pick-up place for a small reward (waiting for more than one passenger?) or whether to move to the drop-off location for a bigger reward at correct drop-off. Reasonable numbers were anything between 5 and 30. Discussing the discount parameter was also acceptable. In cases when “time horizon” was understood differently I look for the solution from other parts of Q1.

What is a good time-out? I have to admit the “$S*S$” statement in the sample program (2nd tutorial) is not really clear. This expression could make sense for a 1D problem: If the agent is performing fully random exploration then it would move on average a distance proportional to sqrt(S) in S time steps (as implied by the theory of random walks or diffusion theory). So in $S*S$ time steps it can be expected to traverse a large part of the environment and thus to find the goal. Note, however, that a random initialisation of the value function does not have this effect without exploration (i.e. ε=0) the agent can be easily trapped between two states (e.g. the left one says “go right” and vice versa). In 2D the distance from the start grows (on average) also like sqrt(t) such that for an environment of X times X you need to wait (proportional to) $X*X$ to reach the boundary and you have a chance that the robot moved to every point in $S*S$, where $S=X*X$. The question is, however, whether such results on random walks, i.e. diffusion theory, apply to the case X=5.

The required number of trials depends on the exploration, but also other factors have an effect, obviously also the problem complexity. It is generally hard to predict, but it is usually possible to increase either the number of trials (if the algorithm did not converge) or to increase the exploration (if the algorithm converged, but not anywhere near a good solution). In academic RL problems you should always try a few more trials than necessary and never too few.

How does the trial length enter? A trial achieves the spreading of the information one step backwards, so in order to get the information all the way to starting state the algorithm has to run through an episode of length T, about T times, i.e. you will need as many episodes as you need training steps for the individual state action pairs (unless you are using eligibility traces) or you have very long episodes that go through some state-action pairs several times.

Most of you discussed this question in terms of total number of states. On the other hand when discussing from the point of view of the total number of states, one should keep in mind that the distribution of the states is not homogeneous such that more trails are (theoretically) necessary than a few times "states times actions times path length". There were a few very good reports that mentioned that only 12 (or 16) routes are actually relevant which explains that the algorithm can converge more quickly than estimated by the total number of states. Still it is good to plan generously w.r.t to the number of trial, at least if other termination criteria are in place (see Q2.3).

The remaining tasks in Q1 are discussed below together with Q2.3.

The most important problem in RL is the exploration-exploitation dilemma, which means, if we are working with an ε-greedy policy, how to control ε. The problem is due to the fact that the theoretical guarantees for convergence towards an optimal solution are not directly applicable to practical problems, such that a compromise must be found that leads in a reasonable time to a good solution.

Good solutions (just like optimal solutions) require a greedy policy. If ε decay as a power law, i.e. like $((T\text{-}t)/T)^a$, then the power (a) should not be too small nor too large, and also the starting value of ε should be chosen appropriately: If a typical path (time horizon) is about 10 steps, then in all but the first few trials ε should be smaller than 0.1 such that the taxi now and then can finish it's course without interruption. E.g. for T=10000 steps, we could set $\varepsilon=0.2\ ((T\text{-}t)/T)^2)$. Then for the last 100 trials ($t>9899$) we have $\varepsilon<0.00002$ such that probably no exploratory action is taken during this evaluation period. If $a=4$, then already after about half the trials exploration becomes ineffective.

It should be noted that such decay scheme for ε are needed in complex problems, but in the present case it is sufficient to run the algorithm with an ε-greedy policy with fixed ε, if ε is set to zero for a final (or perhaps also for an intermediate) testing period. But while even for Q-learning it is not easy to set a good ε, for SARSA it is more difficult to do this, because SARSA requires soft-policies: If now ε is suddenly set to zero, then a solution that has learned to avoid "cliffs" will have no chance to not change and may not be optimal, even for the present problem.

Q4: What is better: SARSA or Q-learning? You came to various conclusions, as the difference between Q-learning and SARSA will depend on the exploration strategy as some of you correctly described: If the exploration rate decays over trials in the same way both will perform more or less identical. For a constant ε, Q-learning is likely to be better if tested with $\varepsilon=0$ in the final period, but SARSA will be better if tested with keeping $\varepsilon>0$, which, however, is not a good idea. For stochastic problems, however, this may be different, i.e. SARSA can have an advantage.

Q2: The idea of the problem was to learn a strategy that can deal with any combination of pick-up and drop-off locations. For the illustration of your result, you were asked to use one specific pair, i.e. some part of the Q- or V-values that you have obtained for the full problem.

Q2.3 asked for a criterion for convergence, and not merely for a specific number of trials to be read off of a curve. This is not an easy question and there are several acceptable answers. Convergence means that no further changes are to be expected, as the results (not the problem as such) are stochastic, we will not see a convergence towards a single value, but a rather a convergence towards a stationary behaviour, i.e. the distribution of the considered quantity does not change. This implies e.g. that the mean of this distribution converges. In the context of RL this is often sufficient, but also the standard deviation deserves attention.

Note that it is important to be clear about the procedure for obtaining the mean reward. Whether the reward in one trial is summed or averaged over time steps does not make a difference. Better is to average a few (e.g. 100) trials to smooth the fluctuations due to the random starting position and the random choice of pick-up and drop-off location. Still better is to run the algorithm several times (if a serious evaluation is required or the confidence is required that a good-enough solution is really good enough), but this additional effort was not expected here. Setting a threshold for convergence (i.e. that the “mean” reward does not leave a certain interval) is obviously dependent on the way a mean reward is defined.

The convergence depends on a number of factors: Decreasing the learning rate (η) can force the algorithm to converge. If this is done slowly (Robbins-Monro conditions), then this will (at sufficient exploration) still leave time for the algorithm to find the optimal solution, but if we have anyway only a few thousands of iterations then this will be on the cost of the achievable performance (premature convergence). The exploration itself is also critical: If ε is set to zero after some trials the algorithm will converge quickly by adjusting the value function.

These considerations aside, what I was hoping for was a discussion of convergence in terms of value (or reward) vs. in terms of action changes. Most of you have shown a good understanding how long it takes to perform all actions in all states at least a few times, this is necessary, but not sufficient for convergence. If, however, after this exploration the actions don't change any more, the policy has converged, while the value function, especially at low learning rate (η) may take much longer to reach it final form. A good test it running a set of greedy trials, then increasing ε substantially, and then performing another greedy test. If the actions have not changed then we can consider the policy to have converged. The problem with this in the taxi task is that there are many optimal solutions (i.e. the taxi can e.g. go E then N or N then E at the same cost), such that the preferred actions in some states can continue to change forever, although this wouldn't have an effect on the value. Complexity theory for the optimal policy is often specified in terms of the difference in the value between the best and the second best action. This difference is here often zero, such that there seems to be no advantage for policy iteration. One solution would be to identify equivalent action sequences, but this would be cumbersome. A better way is to check the whether the values of the actions have changed more than a minimal amount. This amount depends on the choice of γ to the power of the longest optimal path times the cost of the smallest detour (2 steps). We don't have to be precise in this estimate if we make sure that we underestimate this value. Then we can decide whether the best action if it has changed at all, has now a value that differs less from the previous value than this threshold. Values for the other actions can still fluctuate such that we may have an advantage over checking the convergence of the value function. Quite obviously, this procedure is too complicated for the modest taxi problem, such that a good answer would include just a mentioning of the value vs. policy convergence, the identification of either one of these two as an indicator and a hint how fluctuations are dealt with (i.e. averaging over trials). A practical way of identifying convergence is the consideration of two sliding averages with different averaging intervals: If both averages are “long” enough for the problem (here e.g. 20 and 200) and intersect more than once then we can assume convergence (we should still make sure that the algorithm has reached a good performance and did not get stuck somewhere, i.e. some visualisation is always helpful.

Q5 was relatively easy (and should have been answerable even if you did not have the time or patience to run actual simulations). A problem which you may have liked to mention is the fact that the rewards are different for the path towards the pick-up (when the taxi driver may not be that much under pressure to be efficient) and towards the drop-off (when the passenger would complain about addition costs). If would have been nice if it was indicated that inferences from the second part of the journey are less critical then from the first part, although a well-working algorithm would not make differences. There are several other aspects which could have been mentioned here:

- The equivalence of paths could be used (see above)
- hierarchical representation would become useful at least if the problem gets bigger than 5x5 squares
- some of the goods paths are partially identical such that options (I guess only of 2 or 3 steps) might be useful, e.g. straight or “diagonal” paths are useful
- eligibility trace would be useful here as well (this is a bit off-topic)
- some squares don't need to be visited expect immediately after initialisation, such that a grouping of states according to visiting frequency may be possible.

As none of these was expected to be implemented (as the problem is too simple to require this) some additional speculations should have been easy, so at least one remark in addition to the pick-up and drop-off similarity was expected for full marks, otherwise I gave 5 to 7 marks in dependence on the quality of the explanation.

Here are a few more comments each of which applies to only some of the reports

- It is also important to mention the initialisation of the Q-function (optimistic, random, zero). With an optimistic initialisation (you were asked for the maximal reward uin Q1) a good exploration was possible here even at a greedy strategy. Otherwise, ε-greedy action selection was a reasonable choice, although Boltzmann exploration would be more useful here as it suppresses the few low-reward actions while still exploring the remaining actions.
- Average reward can be calculated per step or per episode (see above). Both can be useful: The episode average is also implicit in the value function (at lease for γ close to 1) such that it can be used to check the values. The step average is also interesting, see e.g. the definition of the rewards that is used in policy gradients.
- The learning rate (η) needs to be small in stochastic problems. Here it was (as we were dealing with a deterministic problem) less critical. Why not use learning rate of 1 in the present case? A learning rate of 1 means that the new value ($r+\gamma\ V$) is considered to be reliable and is used to replace the old one (in the Q function) This can actually work in a deterministic problem: As soon you have an update, you put it in the look-up table. It is still discouraged because we have either a stochastic policy as in SARSA or a discrepancy between what the agent sees later and what is in the Q-function in Q-learning. So it is a good idea to have large learning rates in the beginning and smaller later on.
  For η=0.5 in a sense an average of two new values is used (admittedly not in a strict sense), η=0.1 averages about ten rewards. This will be useful in a stochastic problem (although in order to arrive at a good estimate, η should go to zero slowly, cf. the Robbins-Munro

conditions).

- Numbers should be given in a reasonable format: If you define a criterion for convergence, you usually cannot take all of 5 digits of a number seriously. On the other hand, I should say, if you are rounding too much, e.g. give a 10000 instead of a 15000, you may loose some information that is already significant. This depends on the stochasticity of the process that created this number. E.g. a comparison of the performance of two algorithms should consider how precisely the performance was measured: So if the performance is known to, say, 5% accuracy, the difference (more precise calculations not withstanding) should be (about) 10% of the value in order to legitimate the statement that one algorithm is better. Note that if they are, say, just 5% different you cannot conclude that they are equal, but you can say that they are indistinguishable based on your results.
- If there was more time for the assignment, you should have tried to run all your simulations many times (e.g. 100 times) and to take the mean of the repetitions, this would allow you to obtain the results on performance with more accuracy.
- A plot of numerical results should improve clarity. Please avoid using lettering below a size of 8 point. Label the axes (adding a caption that explains the axes is an alternative in less formal settings). If you are using colour make sure that a black-and-white version of the plot is still understandable if you submit a black-and-white print-out. Most importantly, when you use plots try to make a point. There is nothing wrong with stating, e.g. "The two curves are nearly identical" in the caption after explaining the meaning of the two curves.
- Concatenating a matrix' rows into a vector for the purpose of display, if possible. Such a representation may be somewhat useful for understanding what the algorithm does on a plain square, where the pattern is still discernible. Here with the walls and the complex state space this was not really helpful in order to visually verify the results of your work.
  As mentioned in the beginning, the purpose of the graphs is to give an idea whether the algorithm works well. I have to admit that they were also asked for to provide evidence that you have actually solved and understood the problem. Ideally, the graphs should show both: A sliding (or exponential) average (bright colour) and the scattered data (not occluding the average. If your algorithm is running fast, then it is possible to add an outer loop over the critical parameters, which give the final performance for all reasonable values of the parameter. In Q2.4 also other representations were given good marks.
- Generally, you got half of the marks for a reasonable result, e.g. in terms of a graph and the remaining for the explanation. An implausible result could still get marks in this way if explained well.