Reinforcement Learning: Coursework Assignment 2 (*Semester 2, 2014/2015*)

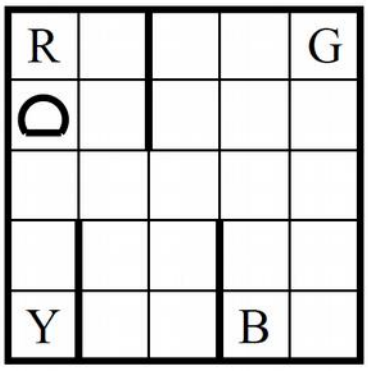
Instructions

- This homework assignment is to be done *individually*, without help from your classmates or others (except the RL tutors). Plagiarism will be dealt with strictly as per University policy.
- Solve all problems and provide your complete solutions (with adequate reasoning behind each step) in a report in computer-printed or *legibly* handwritten form.
- Before you start to write a program, read all questions below carefully.
- Only the report will be marked. The code itself will not be marked, but will used to clarify any questions arising from the report. If you are using code that you have not written yourself, then acknowledge this appropriately (you do not have to mention the code provided for the tutorials). Include references to books and papers that you have been using (you do not have to cite the RL lecture slides or reading material required for this course).
- Use graphical representations wherever suitable. If you use numerical output for demonstrating your results, make sure that the numbers are appropriately rounded and presented in an accessible way. If your problem involves randomness, explain why your result is representative for most of the possible realisations of the underlying random effects.
- Include your code in the submission (preferably Matlab). If you are presenting numerical results in your report, specify all major numerical parameters that were involved in the generation of the data shown.
- This assignment will count for 10% of your final course mark.
- Please submit your assignment by 4 pm on 26th March as a paper copy to ITO as well as an electronic version (including your code) via the submit system (directory "rl").

Questions

Similar to first assignment, you will study a variant of a benchmark problem for RL algorithms, namely the taxi problem (Dietterich, 2000). Below is the specification given in Ref. (Dutech, 2005)

- State variables: taxiLocation {1, ..., S}, passengerLocation {1, ..., L+1} (i.e. waiting at a pickup/drop-off location or being in the taxi), drop-offLocation {1, ..., L}.
- Initialisation of a trail: Taxi is uniformly randomly in any of the S grid squares, passengerLocation is uniformly randomly in one of the L passenger states, dropoffLocation is uniformly randomly one of the L-1 drop-off locations
- Termination of a trial: Passenger was successfully dropped-off (or after a sufficiently long time-out)
- Basic actions are: 1: go north, 2: go south, 3: go west, 4: go east, 5: pick up passenger, 6: drop off passenger.
- Reward: -1 for an attempted movement (whether it is successful or blocked by a wall), -1 for a successful pick-up, 0 for a successful drop-off, -10 for an attempted drop-off with no passenger or at the wrong location, -10 for an attempted pick-up at the wrong location (or if the passenger is already in the taxi).



1. Before you start programming, consider what domain knowledge and what properties of the algorithm may be useful for the solution.
2. Solve the problem using Q-learning[1] using four pick-up/drop-off-locations (L=4) for S=25 and S=100, i.e. the "world" is a 5x5 and a 10x10 grid (you may consider also larger grids, but this is not required). The obstacles (walls) in the larger problem should stay in the same relative positions (or remain comparable in another way). Note, however, that the pick-up/drop-off-locations are relatively smaller in the larger problem and that the actions are moving the taxi also in the larger world by one square only. Solve the problem using a hierarchical algorithm that reuses the results from the small problem in the larger one. There are several ways to do this, choose one and justify you choice. (30/100)
3. Solve the same problem with function approximation for the state-action value function.
    - A simple representation of the state-action value function could use one basis function (e.g. a Gaussian) for each of the squares in the 5x5 case. In the 10x10 case one basis function would then represent four squares. Would this be a reasonable representation for either of the two problem sizes and possibly for even larger ones? You may answer this question with or without a simulation. (20/100)
    - The idea is that the larger problem should be solved with only a moderate increase of computation time. Choose a set of basis functions (possibly using domain knowledge) that is useful for this purpose and study the effect of the problem size in a simulation. You may prefer to use function approximation for spatial states only, i.e. use a different set of parameters of each action and condition (other variants are possible, including continuous (spatial) actions). (50/100)
4. Add a paragraph of conclusions drawn from the solution of these problems, in particular reviewing the design choices you have made.

References:

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Intell. Res. (JAIR), 13*, 227-303.
Dutech, A., Edmunds, T., Kok, J., Lagoudakis, M., Littman, M., Riedmiller, M., Whiteson, S. (2005) Reinforcement learning benchmarks and bake-offs II. *Workshop at Advances in Neural Information Processing Systems conference*.

[1]You can also use a different reinforcement learning algorithm if you prefer.