CS6700 : Reinforcement Learning
Programming Assignment 2

Deadline : 12 Apr, 2019, 11.55 pm

- This is an individual assignment. Collaborations and discussions are strictly prohibited. *Do **not** put up the code for any part of the assignment in public repositories online.*
- You have to turn in the well-documented code along with a detailed report of the results of the experiments. Typeset your report in LaTeX.
- Be precise with your explanations. Unnecessary verbosity will be penalized.
- Check the Moodle discussion forums regularly for updates regarding the assignment.
- *Any kind* of plagiarism will be dealt with extremely seriously. Acknowledge any and every resource used.
- **Please start early.**

We will be using python and OpenAI Gym to solve the problems in this assignment. The above link leads to the page on 'Getting started with Gym'. Please install OpenAI Gym before starting.

1 Puddle World

1.1 Environment details

1. This is a typical grid world, with 4 stochastic actions. The actions might result in movement in a direction other than the one intended with a probability of 0.1. For example, if the selected action is N (north), it will transition to the cell one above your current position with probability 0.9. It will transition to one of the other neighbouring cells with probability 0.1/3.
2. Transitions that take you off the grid will not result in any change.
3. There is also a gentle Westerly blowing, that will push you one additional cell to the east, regardless of the effect of the action you took, with a probability of 0.5 [1].
4. The episodes start in one the start states in the first column, with equal probability.
5. There are three variants of the problem, A, B, and C, in each of which the goal is in the square marked with the respective alphabet. There is a reward of +10 on reaching the goal.
6. There is a puddle in the middle of the gridworld which the agent would like to avoid. Every transition into a puddle cell gives a negative reward depending on the depth of the puddle at that point, as indicated in the figure.

[1] You might want to turn this off for Goal C

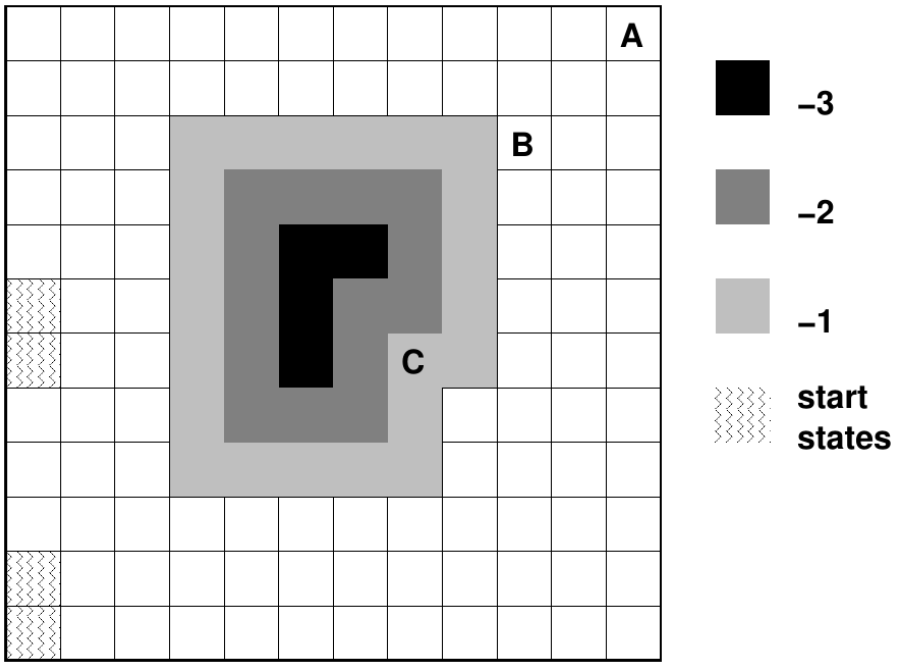

Figure 1: The puddle-world.

1.2 SARSA

1. Create this gridworld in OpenAI-gym (documentation and code) We highly recommend getting used to Gym since it is offers a rich suite of flexible, open-source environments that you might be using for any potential future research.

1.2.1 SARSA

2. Implement *SARSA* to solve each of the three variants of the problem. For each variant run experiments with a gamma value of 0.9. Compute the averages over 50 independent runs. The deliverables are the following:

   (a) Turn in two learning curves for each experiment, one that shows how the average number of steps to goal changes over learning trials and the other that shows how the average reward per episode changes.

   (b) Indicate the optimal policies arrived at in each of the experiments.

3. Now solve this with *Sarsa*(λ), for values of $\lambda = \{0, 0.3, 0.5, 0.9, 0.99, 1.0\}$. Apart from the plots mentioned above, also turn in a plot/graph that shows the average performance of the algorithms (in terms of average steps and reward obtained), for the various values of λ after 25 learning trials.

1.3 Policy Gradient

Consider the following parameterization of a policy: There is a "preference" for each action, for each column and for each row. Thus the set of preferences can be denoted by $\theta_x(N,0), \theta_y(N,0), \theta_x(S,0), \theta_y(S,0), \theta_x(E,0), \theta_y(E,0), ..., \theta_x(W,9), \theta_y(W,9)$, for a total of 80 preference values. The total preference for an action a in a state (i, j) is given by $\theta_x(a,i) + \theta_y(a,j)$. The action probabilities are generated by a soft-max function using these preferences.

4. Implement a MC policy gradient algorithm. Choose appropriate learning rates, and turn in two curves for each variant as indicated in the first part as well as the optimal policies learnt.

5. Explicitly derive the update equations. Also, comment on the suitability of this policy parameterization for the given task, and for gridworld problems in general.

1.4 Function Approximation

Consider a puddle world exactly like in the previous question, but blown up by 1000 times. While it is still possible to run your earlier code on this version, the aim of this exercise is to get you to play around with function approximation.

6. Repeat from subsection 1.2.1 with a linear function approximator. Report results for at least 2 values of λ.

Submission Guidelines

Submit a single tar/zip file containing the following files in the specified directory structure.
Use the following naming convention: `rollno_PA_2.tar.gz` or `rollno_PA_2.zip`
A sample submission would look like this: