Reinforcement Learning

HW Submission Guidelines

March 6, 2019

General Instructions

1. Write your answers in a **single PDF** file (not a doc).
2. Theory and programming assignments should be on different pages. **Indicate at the top your names, ID numbers and e-mails** - same PDF.
3. Do not submit images, include them in the PDF.
4. In the PDF write some explanations for **every question** (even if it's just python file name).
5. Attach also python files (in addition to putting them in nova).
6. Put the PDF and all python files in a **single ZIP** file and send it with the students details (no additional comments in the mail). Name the zip with the students IDs, i.e. 123456789_234567890.*zip*.
7. Assignments should include answers to theory questions, a report containing plots and statistics as asked for and the code.
8. Exercises should be submitted to the course e-mail: *rl.tau.2019@gmail.com*.
9. Assignments must be done in pairs.

Theoretical Assignments

1. Follow the exercise specifications exactly. Seek clarifications if needed.
2. Your proofs should be both rigorous and clear.

Programming Assignments

1. Make sure your names, ID numbers and e-mails are printed on the report. If the checker encounters a technical problem and has no way to contact you, this will result in a significant point reduction.

2. Follow the exercise specifications exactly.

3. Your code should run properly on Python 3.6 on linux (it doesn't have to run on Python 3 and above). It is required that it should run on *nova.cs.tau.ac.il* using */usr/local/lib/anaconda3-5.1.0/bin/python*. Test it before submitting it!

4. The one exception to the rule above is plots. You are not required to create your plots using Python on nova (although it is possible, using the savefig command from pylab). If you choose not to, then make sure your submission saves all the data required to generate the plot in a separate file, one file per plot.

5. For each programming assignment: In the printed submitted material indicate the path to the directory from which the program is executed. Make sure that the directory and files are read accessible by everyone (the output location of created/modified files should be a parameter to the program). This could be in your personal home directory, or elsewhere, as long as it is accessible.

6. The directory should contain a README file that has the students names and following detailed:

   (a) A description of each relevant source/input file.

   (b) A command to execute and its parameters if any. This command should also set any required environment. You can specify a different command per question or subquestion, as long as it is clear, and there are no dependencies between the commands (i.e., command #2 does not assume command #1 was run). If you need such a dependency - make one command for two questions, etc.

   (c) A description of each relevant output file.

7. Your program should not output warnings or errors when run correctly.

8. Make sure your program remains viable if your submitted files are moved (for example, don't refer to files you do not submit, and paths that are not necessarily accessible to others). In particular, watch for OS-dependent names (e.g., Windows-type path names).

9. Your code should be well documented and clear. Use meaningful variable names and sufficient comments to achieve this.

10. Your plots should be clear. Set meaningful labels and ranges on the axes, as well as meaningful titles and legends.

11. It is recommended that you use numpy vector and matrix operations instead of for loops with many iterations, whenever possible. This makes for faster, more compact programs.