Statement of the Practical Work

The goal of this practical work is to construct several Neural Networks models to play the classical Pac-Man game (`https://en.wikipedia.org/wiki/Pac-Man`). It is based on the Pac-Man projects developed for UC Berkeley's introductory artificial intelligence course, CS 188 (`http://ai.berkeley.edu/project_overview.html`), that we have adapted to Python 3.7 and particularized for the goals of the course.

In the file *practical_work.tar.gz* you will find the basic software needed to develop the practical work:

1. The original (and adapted) files in the Pac-Man projects: *game.py, ghostAgents.py, graphicsDisplay.py, graphicsUtils.py, keyboardAgents.py, layout.py, pacmanAgents.py, pacman.py, textDisplay.py* and *util.py*, together with the *layouts/* folder containing several layouts of the game.
2. The *MLNN_Agents.py* file, containing the implementation of two agents:
   (a) *MLNN_RandomAgent*, that performs random moves (just to give an example of how can an heuristic agent be implemented).
   (b) *MLNN_LearnedFromDataAgent*, that will play with the model that you have to construct from some data.
3. The *MLNN_ConstructModel_MLP.py* file, containing the (incomplete) implementation of the class *MLP_Keras*, used by *MLNN_LearnedFromDataAgent*. **This is the only file you will have to modify!**
4. The *Data/* folder, containing the data needed to train your models.
5. The *Models/* folder, where you can store your final models.

How does the Pac-Man software work?

1. In its basic form, simply type *"python3.7 pacman.py"*, and you will be able to play with the keyboard (implemented in *keyboardAgents.py*).
2. If you want to play with an implemented agent, then you will need to indicate it with option *-p*: *"python3.7 pacman.py -p MyImplementedAgent"*. For example, when you want to play with *MLNN_RandomAgent*, you will need to type *"python3.7 pacman.py -p MLNN_RandomAgent -a nf=NF,fm=FM,sa=True"*, where the *-a* option is used to inform the parameters of the agent (see below).
3. Typing *"python3.7 pacman.py -h"* you can see all the options of the program (the *README.TXT* file also contains useful information and examples of use), where you can change:

(a) The layout, with the *-l* option

(b) The type of ghost agent, with the *-g* option

(c) The number of ghosts, with the *-k* option

(d) Several useful options for testing your model, such as the number of games (*-n* option) or running with minimal output and no graphics (*-q* option)

What are you expected to do?

1. Complete the implementation of the *MLNN_ConstructModel_MLP.py* file, setting the architecture of the MLP, the training parameters, and implementing the model selection and construction of the final model parts (function *constructModel_MLP_Keras*). Follow the instructions in the file.

2. Run *MLNN_ConstructModel_MLP.py* in "model selection mode" (*-s* option) in order to obtain your best architecture and parameters for every data set in *Data/*. You may obtain different optimal parameters for every data set. Type *python3.7 MLNN_ConstructModel_MLP.py -h* to see the parameters of the program.

3. Obtain the final models training (with the parameters found in the model selection step) on every data set in *Data/*, indicating the file name where you want to save your model (*-m* option).

4. Test your final models running *pacman.py* with the already implemented agent that takes its decisions from your trained model:
   python3.7 pacman.py -p MLNN_LearnedFromDataAgent -a nf=NF,fm=FM,sa=True,
   where *NF* if the number of features in the input data (4 for *Data-004F* files, 100 for *Data-100F* files and 500 for *Data-500F* files), *FM* is the model file name, and *sa* indicates whether you allow the pacman to stop or not (*sa=True/False*).

5. Obtain statistics about the performance of your models (see options *-n* and *-q* above), by running them with different pacman configurations:

   (a) Changing the layout (-l smallClassic / -l mediumClassic / -l originalClassic, for example)

   (b) Changing the types of ghosts (-g RandomGhost / -g DirectionalGhost)

   (c) Changing the number of ghosts

   (d) etc

   It is expected that the results with *Data-004F* files will be better than those with *Data-100F* files, which will be better than those with *Data-500F* files.

IMPORTANT: It is a good idea to test your hardware+software environment and make a good estimation of the execution times with sufficient advance.

You must deliver:

1. A brief document with the description of the experiments performed, a summary of the problems encountered, the final parameters selected for every data set, the results obtained, your conclusions, discussion, etc.

2. The final version of your software (remember that, in principle, only the *MLNN_ConstructModel_MLP.py* file should have been modified)

3. The files with the final models for every data set

Optional extensions:

1. Study how the results change as a function of the number of examples.

2. Test a model *XXX* different from MLP. To that end, construct a file named *MLNN_ConstructModel_XXX.py*, with the same structure than the original *MLNN_ConstructModel_MLP.py*, where you include the specific implementation for *XXX*. Does *XXX* work better than MLP?

3. Implement a heuristic agent that works better than the best model constructed with the provided data, generate new data based on this heuristic and train a new MLP (or any other model *XXX*) with these new data. It is expected that the new model will work similar to the heuristic agent, and therefore better than the best model constructed with the provided data.

The practical work should be delivered by January 7th 2020.