**Virtual Pi2Go Programming: Exploration versus Exploitation in Machine Learning**

**AIM:** After completing this worksheet you should be able to explain how to use a learning rate to gradually move a machine learning program from exploring behaviours to selecting behaviours that are likely to yield a reward.

**You Need:** To complete this worksheet you need to have a virtual Pi2Go simulator (see WS1), understand how to control the robot’s motors and sensors (WS3&WS4), be able to use files to store programs (WS5), use control structures (WS7-10), Data Types (WS12-14), functions (WS16) and the time module (WS6). You should also have completed WS22 (Introduction to Machine Learning)

**If the simulator isn’t already running: Start the Simulator, Select the Pi2Go Simulation and oval.xml, then start IDLE (open a *new IDLE window* if you have used IDLE to start the simulator).**

So far you have written programs to that attempt to learn the best action to take in any situation by taking an action at random. However, when writing such a program, you want to gradually move from a phase in which the program *explores* different actions to see which one works best in a state and towards a phase in which it *exploits* it learning by selecting the best action. We can do this by having a number, let us call it *epsilon*, that gradually reduces as the program progresses.

Each time the program must select an action it generates a random number. If the number is higher than epsilon, then the program selects the action which currently has the highest expected reward. If the number is lower than epsilon, then the program selects an action at random.

We gradually reduce epsilon (for instance we can start with epsilon at 1 and reduce it by 0.05 each time the program earns a reward of 2).

**Exercise 1:** Introduce epsilon into your program from WS22. This program should attempt to learn how to move around the oval by gradually moving to a state where it has successfully learned what to do.

**Hints:**

* You will need to write an algorithm for selecting the action with the highest reward that you can use when the random number generated is higher than epsilon.
* When testing your program, you may need to move the robot back to the oval if it moves too far away from it.
* This program takes a while to run so you may find it useful to use print statements to show when epsilon reduces and whether an action is being selected at random or because it is the current best choice.



University of Liverpool, 2019

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).