Tutorial 2 (Neural Computing)
Pole Balancing exercise

Artur d'Avila Garcez
Department of Computer Science – City, University of London

1. Start MATLAB.

2. Copy the file containing the pole balancing examples (pole_balancing.txt) from the module's webpage on Moodle to your working directory.

3. Load the file on MATLAB with "load pole_balancing.txt".

4. Inspect the file which should contain 200 examples, each line with 2 input values (angle and angular velocity) and a target output (force).

5. Create an input data structure and a target data structure from the "pole_balancing" file:

   5.1 pole_input = pole_balancing(:,1:2);

   5.2 pole_target = pole_balancing(:,3);

6. Start the neural network tool: nnstart

7. Create, configure and initialize a multilayer neural network with 2 input neurons and 1 output neuron (how many hidden neurons are necessary to solve this problem?):

   7.1 The easiest way to do this is to click on "fitting tool", select "pole_input" and "pole_target", select samples as "rows", click "next" and "train" the network.

   7.2 Inspect the network's "performance"

   7.3 Click "next" twice and under "save results" generate an "advanced script"

   7.4 Now you can edit the script, save it as a "pole.m" file and press F5 to run it.

8. What happens if you change the network's activation functions by adding the following lines of code at a suitable place in pole.m?

```
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'logsig';
```

   8.1 What should be the output's activation function given that the target outputs are real numbers not restricted to the interval [0,1]?

9. What happens if you change the learning rate (e.g. `net.trainParam.mu = 10;`) or the number of hidden neurons, say, to 2 or 200? Why should you use a small learning rate?

10. Check if the network generalises well to unseen examples by creating a training set with 150 examples and a test set with 50 examples, and comparing the network's *training set performance* and *test set performance*.