Modelling of Cognitive Processes: Test 1 - October 2019
You have 1 hour for this test. You can look up info online during the test.

Comment your code and make sure it runs. A code that does some of the parts of the exam but runs, is better than a code that "can potentially" do everything but does not run. Also, even if does not run properly, send us the last version of your script with comments so we can evaluate what you were trying to do.

Take the test1_starting_script_**2nd_version**.py script, and make a new file surname_firstname_mcp_test1.py. When finished, upload to Assignments → Test 1, and come to sign at the front of the room.

The diagram below shows what the arrays `cat_prototype`, `dog_prototype` and `test_inputs` represent in the model.

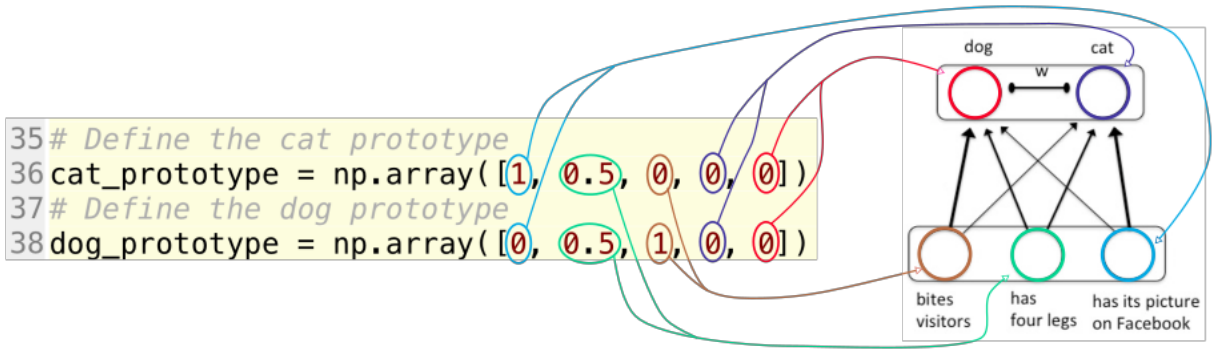

Figure 1. Correspondence between the model and the arrays in the script.

1. The script currently trains the network on 10 samples of each category, then it presents 20 test inputs (10 of each category) in the test phase and prints the activation of the output units at the end of the activation optimization for each test input. The standard deviation (sigma) of random, normally distributed noise in the test_inputs is currently set at 3.0.
   What is the proportion of test inputs for which the activation optimization finishes at a higher activation for the correct output unit than for the incorrect output unit (i.e. does a correct pet detection)? See how the array test_inputs is built to know what is the correct detection for each sample.
   Then decrease that sigma to 0.5. What is the proportion of test inputs for which the activation optimization finishes at a higher activation for the correct output unit?
   Explain why the performance in the two cases is different. Put your explanation in the initial comments section of the code.

2. Add 3 new input units to the network to represent these new concepts: "Is Allergenic", "Humans' best friend" and "Makes noise". Don't forget to change everything throughout the script that has something to do with the number of units.

3. Now apply Hebbian learning with this new input format. In the training phase, present 30 cats and 30 dogs. The cat prototype is now (1, 0.5, 0, 1, -0.9, -0.3). Hence, the prototypical cat has a value of 1 on the "Is Allergenic" variable, a value of -0.9 at the "Humans' best friend" variable and a value of -0.3 at "Makes noise". The dog prototype is now (0, 0.5, 1, -0.9, 1, 1.5). Add random, normally distributed noise to the training samples and keep the samples_noise_std variable as it was in the original script (0.01). Present the training samples in a random order during training.

4. After the learning phase comes a test phase. Present these new cat and dog prototypes (with some noise as before) to the network (10 of each like before). Change sigma back to 3.0. What is the proportion of test inputs for which the activation optimization finishes at a higher activation for the correct output unit (i.e. does a correct pet detection)?

5. Is the result different than with only 3 input units (question 1)? If it is different explain why in the comments code at the top of your script.