Modelling of Cognitive Processes

Test 2, December 2019

You have 1 hour for this test. You can look up info online during the test. You are also allowed to use code from the practical sessions, or code you wrote earlier. Communication with people is not allowed during the test.

When finished, upload your file **lastname_firstname_test2.py** to Ufora, and come to the front of the class, with this paper, signed.

Comment your code, and make sure it runs. A code that does some of the parts of the test 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.

Note that the problem statement consists of 2 pages.

Problem statement

Human children learn throughout their childhood what makes a sentence grammatical. This process is quite interesting, but we are not quite sure whether learning grammar is a linearly separable problem or not. Therefore, we ask you to help us out.

In the world this test was made in, a sentence is considered grammatical when the first and the last word are the same. For simplicity, we will assume that we only have two words: A and B. Further, sentences are always composed of three words. So, grammatical sentences are BBB or BAB (for example); ungrammatical ones would be AAB or ABB (for example).

The steps you should take:

1. Generate all possible sentences that can be formed. You should have 2^3 (=8) different sentences. Make sure that your sentences are translated into appropriate patterns that can be presented to your model. Do *not* code your sentences as strings (['A', 'B', 'A'] to represent the sentence 'ABA' will not work). Work with arrays of 0's and 1's to represent words, and their position in the sentence. Higher scores are awarded if your code is easy to generalize to larger grammars, such as larger numbers of words, and larger numbers of words per sentence. However, we advise to initially hard-code the possible sentences, and then get back to this step when you carried out the next steps.
2. Make an input array by repeating each possible input 50 times. So, your final input array should have a shape[0] of 400 (50 x 8). Make an associated output array (1 if the sentence is considered grammatical, 0 if otherwise).
3. Train a two-layered Perceptron 40 times on the data. For each of the 40 simulations, use a different training- and test set. Present stimuli in random order during training. Training set size should be 60% of the total data set each time. Print the average accuracy for your Perceptron (across the 40 data sets). At the top of your script (i.e., before any actual code!), write down what you conclude from your output, and why you find this logical (or not). Also check using your code if it makes a difference whether your Perceptron has a bias (intercept) unit or not; also explain at the top of the code, why this is so. (Clearly write "Q3: Response" at the top of your code.)
4. Fit a three-layered model using a module we used in class. Find the minimal model: the model with the lowest number of hidden units that is able to achieve at least 95% accuracy across 40 simulations. Here also, the training- and testing set should be different for the 40 simulations, training set size should be 60%, and stimuli must be presented in random order.
5. Based on your results from step 4, describe why the three-layered model has a worse / equal / better performance than the two-layered model from Step 3. (Clearly write "Q5: Response" in your code).
6. If your minimal model has more than one hidden unit, construct a new model with the same number of hidden units, but with two hidden layers (instead of one). If the minimal model from Step 4 has just one hidden unit, then add one hidden unit. Does this new model perform better / the same / worse than the minimal model from Step 4? Did you expect this? (Clearly write "Q6: Response" in your code.)