Automatic Differentiation

Background

Download the following and place in a single folder:

- `as04_q3.ipynb`
- `matad.py`
- `utils.py`.

You have seen the module `utils.py` before (week 03). The module `matad.py` is like `ad.py`, except that it works with matrices instead of scalars; it defines the `Mat` class, and associated `MatOperation` classes.

Note that the `MatOperation` functions return `Mat` objects, but their `backward` functions deal with NumPy arrays, not `Mat` arrays.

Question 3: Backprop using Auto-Differentiation

The jupyter notebook `as04_q3.ipynb` creates and tries to train a neural network on a simple dataset. However, some critical parts of the code are incomplete. Complete the implementation by doing the following:

(a) `backward`: Complete the implementation of the `backward` method in the `Mul` class. As the documentation states, the `Mul` class implements matrix-matrix multiplication. The `backward` method takes a 2D NumPy array as input, applies its own term to the chain of derivatives, and sends those derivatives (NumPy arrays) to the `backward` function of each of its arguments.

(b) `__call__`: Complete the implementation of the `__call__` function in the `Connection` class. This class represents the connection weights and biases between two `Population` layers. The `__call__` function takes the activity of the layer below, multiplies it by the connection weights, and adds the bias, and returns the resulting input current (as a `Mat` array).

Hint: Take advantage of the properties of the `Mat` and `MatOperation` classes. If you do it properly, your solution to part (c) will be much easier.

(c) `learn`: Complete the `learn` function in the `Network` class. To get full marks, you must use the automatic-differentiation functionality of the `Mat` and `MatOperation` classes. Notice that the `Network` class has a method called `parameters()` that returns a list of all the `Mat` objects in the network that correspond to connection weights and biases.

There is some code at the end of the notebook that creates a network and runs `learn` a the simple dataset. If your code works, you should see

Submit the updated notebook to Kritik. Be careful, and make sure you <u>submit the correct notebook</u>! Remember to ensure that nothing in the file can be used to identify you; the peer-assessment process is supposed to be anonymous.