Homework-02

Neural Networks - Feedforward pass

Background

This homework teaches how to code the internals of a simple neural network. While there are a number of tools for this, it is important to understand the main operations in a neural network and code your very own neural network library.

Deliverables

Small write up in *Markdown*.
Source code in `neural_network.py`, `logic_gates.py`, `test.py` description in `README.md`.

API (Class: `NeuralNetwork`) Points: 6

```
-- create the dictionary of matrices 0
[nil] __init__(([int] in, [int] h1, [int] h2, …, [int] out))
-- returns 0(layer)
[2D DoubleTensor] getLayer([int] layer)
-- feedforward pass single vector
[1D DoubleTensor] forward([1D DoubleTensor] input)
-- feedforward pass transposed design matrix
[2D DoubleTensor] forward([2D DoubleTensor] input)
```

When I import your library (`from neural_network import NeuralNetwork`) I should see **only** three (yes, 3) above mentioned methods defined by you as `NeuralNetwork`'s attributes. No global variables, no other helper functions (which must be `local`, if needed). Input will be `2xn` and corresponding output will be `1xn`.

Secondary API (`logic_gates`) Points: 4

```
Class: [boolean] AND([boolean] x, [boolean] y)
Class: [boolean]  OR([boolean] x, [boolean] y)
Class: [boolean] NOT([boolean] x)
Class: [boolean] XOR([boolean] x, [boolean] y)
```

No global variables, no other helper functions (which must be `private`, if needed).
Note to self: split API in 4 classes with three methods each: `__init__`, `__call__` and `forward`, so that one can use the given logic function without setting the parameters every time.

Instructions

Part A:

1. Create a Python script and create an object `model` of class `NeuralNetwork`.
2. Initializing class using `__init__()`, with the list `(in, h1, h2, …, out)` as argument, will populate the `network` dictionary with the `0(layer)` matrices (which are mapping layer `layer` to `layer + 1`), initialised to random values (`0` mean, `1/sqrt(layer_size)` standard deviation). The size of the input layer is `in`, the size of the hidden layers are `h1`, `h2`, …, and the size of the output layer is `out`.

3. `getLayer(layer)` will return `0(layer)`.
4. By running `forward(input)` the script will perform the forward propagation pass on the network previously built using sigmoid nonlinearities.

Part B:

1. Use the API in `NeuralNetwork` to create an `AND`, `OR`, `NOT` and `XOR` networks that perform logic operation on `boolean` values.
2. logic_gates.py will have four classes as per the second API. Each class constructor will call `NeuralNetwork` class and then set the weights of neural network using `getLayer([int] layer)`.
3. Calling forward function of any logic operation will call `forward()` of `NeuralNetwork` and return the output of the logic operation.

Format

Submit ZIP file containing all the deliverables with instructions to run your script using. Your folder name should be as following:
`<your_purdue_username>_HW<0X>`
For example, my folder name for this homework will look like `aabhish_HW02`.

Suggestions/Recommendations

- I will test if the forward propagation is computed correctly against my own neural network, which will be sharing the same matrices `0` of your network.
- You should build the `AND`, `OR`, `NOT` and `XOR` networks using your own `NeuralNetwork` library. You will need to convert `boolean`s to the integers `0` and `1` and vice versa, in order to comply with the given API.

Sample Test: (This is just a pseudo code)

```
from logic_gates import AND

from logic_gates import NOT


And = AND()

Not = NOT()


print(And(True, True))

print(Not(True))
```

Notice how `And()` is supposed to automatically call `And.forward()`. Hint: look into usage of `__call__()`.