CSE 4392 - Assignments - Assignment 2

List of assignment due dates.

The assignment should be submitted via Canvas. Submit a file called assignment2.zip, containing the following files:

- answers.pdf, for your answers to the written tasks, and for the output that the programming task asks you to include. Only PDF files will be accepted. All text should be typed, and if any figures are present they should be computer-generated. Scans of handwriten answers will NOT be accepted.
- All files containing your Python code for the programming tasks. Feel free to implement your solutions in multiple files, and to reuse code from those files in multiple tasks, as long as the executable files follow the specified naming conventions.

These naming conventions are mandatory, non-adherence to these specifications can incur a penalty of up to 20 points.

Your name and UTA ID number should appear on the top line of both documents.

Task 1 (30 points, programming)

In this task you will implement the "test module" part of perceptrons. Your code will create a perceptron with specified weights, and will compute the output of that perceptron for a specified input vector

Arguments

You must implement a Python executable file called `perceptron`, that is invoked as follows:

```
perceptron <weights_file> <input_file> <activation>
```

The arguments provide to the function the following information:

- The first argument, <weights_file>, is the path name of the file where the weights of the perceptron are specified. The path name can specify any file stored on the local computer. The file contains one number per line. The first line is the bias weight. Examples of such weight files are weights1.txt and weights2.txt. Note that the weights in weights1.txt specify the AND perceptron as discussed in the lecture slides.
- The second argument, <input_file>, is the path name of the file where the input vector is specified. The path name can specify any file stored on the local computer. The file contains one number per line. Examples of such files are
  - input1_00.txt, input1_01.txt, and input1_11.txt. These input files can be used together with weight file weights1.txt, as the dimensionalities match.
  - input2a.txt and input2b.txt. These input files can be used together with weight file weights2.txt, as the dimensionalities match.
- The third argument, <activation> can have two possible values: `step` or `sigmoid`. If the value is `step`,

the perceptron should use the step function as its activation function. If the value is `sigmoid`, the perceptron should use the sigmoid function as its activation function.

Output

The program should compute and print the values of:

- `a`, which denotes the result of the first step (dot product of weights and inputs, plus bias weight).
- `z`, which denotes the result of the second step (activation function applied to the result of the first step).

You can refer to slide 9 from the introductory neural network slides for the full definitions of `a` and `z`.

Please use the following line for printing, to make sure that printing outputs are formatted in a uniform way and can be automatically graded:

```
print("a = %.4f\nz = %.4f" % (a, z))
```

These are some examples of output from my solution:

```
perceptron weights1.txt input1_00.txt step
a = -1.5000
z = 0.0000

perceptron weights1.txt input1_11.txt step
a = 0.5000
z = 1.0000

perceptron weights1.txt input1_11.txt sigmoid
a = 0.5000
z = 0.6225

perceptron weights2.txt input2a.txt sigmoid
a = -8.1900
z = 0.0003
```

Task 2 (40 points, programming)

In this task you will implement the "test module" part of layered feedforward neural networks. Your code will create a neural network with specified number of layers, number of units in each layer, and weights, and then your code will compute the output of that neural network for a specified input vector. All layers in your neural network are assumed to be fully connected.

Arguments

You must implement a Python executable file called `nn_forward`, that is invoked as follows:

```
nn_forward <nn_file> <input_file> <activation>
```

The arguments provide to the function the following information:

- The first argument, <nn_file>, is the path name of the file where the complete specifications of the neural network are provided. The path name can specify any file stored on the local computer. Examples of such files are nn_xor.txt and nn2.txt. Note that the weights in nn_xor.txt specify the XOR network as discussed in the lecture slides.

  To minimize the amount of time you spend writing code that deals with these files, you are provided with python code in nn_load.py, which you are free to use and modify as you wish. You can use the provided `nn_load` function to read <nn_file>. Please refer to the documentation for that function to understand its return values. The return values of `nn_load` provide all the information that you need to create the neural network.

  Although it is not necessary for you to complete this program, if you want you can read the format specifications that <nn_file> has to follow.

- The second argument, <input_file>, is the path name of the file where the input vector is specified. The path name can specify any file stored on the local computer. The file contains one number per line. Examples of such files are
  - input1_00.txt, input1_01.txt, and input1_11.txt. These input files can be used together with weight file nn_xor.txt, as the dimensionalities match.
  - input2a.txt and input2b.txt. These input files can be used together with weight file nn2.txt, as the dimensionalities match.
- The third argument, <activation> can have two possible values: `step` or `sigmoid`. If the value is `step`, the perceptron should use the step function as its activation function. If the value is `sigmoid`, the perceptron should use the sigmoid function as its activation function.

Output

The program should compute print the `a` values and `z` values of all units in all layers. These values should be printed layer by layer, starting from the input layer, and finishing with the output layer. All values should be rounded to four decimal digits. These are some examples of output from my solution. Your output should be formatted to match EXACTLY the format that you see below:

```
nn_forward nn_xor.txt input1_00.txt step
layer 1, a values: [  -0.5000  -1.5000 ]
layer 2, a values: [  -0.5000 ]

layer 0, z values: [   0.0000   0.0000 ]
layer 1, z values: [   0.0000   0.0000 ]
layer 2, z values: [   0.0000 ]


nn_forward nn_xor.txt input1_01.txt step
layer 0, z values: [   0.0000   1.0000 ]
layer 1, z values: [   1.0000   0.0000 ]
layer 2, z values: [   1.0000 ]


nn_forward nn_xor.txt input1_01.txt sigmoid
layer 1, a values: [   0.5000  -0.5000 ]
layer 2, a values: [  -0.2551 ]
```

```
layer 0, z values: [   0.0000   1.0000 ]
layer 1, z values: [   0.6225   0.3775 ]
layer 2, z values: [   0.4366 ]


nn_forward nn2.txt input2a.txt step
layer 1, a values: [  -0.0620  -0.2270 ]
layer 2, a values: [  -0.2500  -0.2000   0.1000 ]

layer 0, z values: [  -0.3000   0.6000   0.7000   0.4000 ]
layer 1, z values: [   0.0000   0.0000 ]
layer 2, z values: [   0.0000   0.0000   1.0000 ]


nn_forward nn2.txt input2b.txt sigmoid
layer 1, a values: [   0.2940  -0.2845 ]
layer 2, a values: [  -0.1998   0.0177   0.0198 ]

layer 0, z values: [   0.3000   0.1500  -0.1000   0.2000 ]
layer 1, z values: [   0.5730   0.4294 ]
layer 2, z values: [   0.4502   0.5044   0.5050 ]
```

Task 3 (10 points).

Note: In this question you should assume that the activation function of a perceptron is the step function. More specifically, this function:

- outputs 0 if the weighted sum of inputs is LESS THAN 0 (not less than or equal).
- outputs 1 if the weighted sum of inputs is greater than or equal to 0.

Design a perceptron that takes three Boolean inputs (i.e., inputs that are equal to 0 for false, and 1 for true), and outputs: 1 if at least two of the three inputs are true, 0 otherwise. You should NOT worry about what your perceptron does when the input values are not 0 or 1.

Task 4 (10 points).

Note: In this question you should assume that the activation function of a perceptron is the same as for Task 3.

Design a neural network that:

- takes two inputs, A and B.
- outputs 1 if 2A + 3B = 4.
- outputs 0 otherwise.

Your solution should include a drawing of the network, that shows all edges connecting outputs of one layer to inputs in the next layer, and that also shows the values of all weights (including bias weights) of all perceptrons.

Task 5 (10 points).

Note: In this question you should assume that the activation function of a perceptron is the same as for Task 3.

Is it possible to design a neural network (which could be a single perceptron or a larger network), that satisfies these specs?

- Takes a single input called X, which can be any real number.
- If $X < 3$, the network outputs 0.
- If $3 < X < 7$, the network outputs 1.
- If $X > 7$, the network outputs 0.

We don't care what the network outputs when $X = 3$ or $X = 7$.

If your answer is no, explain why not. If your answer is yes, your solution should include a drawing of the network, that shows all edges connecting outputs of one layer to inputs in the next layer, and that also shows the values of all weights (including bias weights) of all perceptrons.

CSE 4309 - Assignments - Assignment 2