



MITx 6.86x

Machine Learning with Python-From Linear Models to Deep Learning[Course](#)[Progress](#)[Dates](#)[Discussion](#)[Resources](#)[Course](#) / [Unit 3. Neural networks \(2.5 weeks\)](#) / [Project 3: Digit recognition \(Part 2\)](#)[< Previous](#)

3. Activation Functions

[Bookmark this page](#)

Project due Apr 5, 2023 08:59 -03 Completed

The first step is to design the activation function for each neuron. In this problem, we will use weights to 1, use **ReLU** for the activation function of the hidden layers, and use an identity output neuron. The hidden layer has a bias but the output layer does not. Complete the `neural_networks.py`, including `rectified_linear_unit` and `rectified_linear_unit_derivative` you to use in the `NeuralNetwork` class, and implement them below.

You will be working in the file `part2-nn/neural_nets.py` in this problem

Correction note (Nov 1): In the `part2-nn/neural_nets.py`, in the definition of `Class` the initialization of weights has now been changed to an initialization as float rather than integer. You can either re-download the updated project release [mnist.tar.gz](https://learning.edx.org/course/course-v1/edX/6.034x/2022S2/part2-nn/mnist.tar.gz), or change the corresponding `nn/neural_nets.py` to the following, where we have added decimal points to all numbers.

```
class NeuralNetwork():

    def __init__(self):

        # DO NOT CHANGE PARAMETERS (Initialized to floats instead of integers)
        self.input_to_hidden_weights = np.matrix('1. 1.; 1. 1.; 1. 1.')
        self.hidden_to_output_weights = np.matrix('1. 1. 1.')
        self.biases = np.matrix('0.; 0.; 0.')
```

Rectified Linear Unit

2.0/2.0 points (graded)

First implement the ReLU activation function, which computes the ReLU of a scalar.

Note: Your function does not need to handle a vectorized input

Available Functions: You have access to the NumPy python library as `np`

```
1 def rectified_linear_unit(x):
2     """ Returns the ReLU of x, or the maximum between 0 and x."""
3     return np.maximum(0, x)
4
```

Note: Your function does not need to handle a vectorized input

Available Functions: You have access to the NumPy python library as np

```
1 def rectified_linear_unit_derivative(x):
2     """ Returns the derivative of ReLU."""
3     return np.greater(x, 0).astype(int)
4
```

< Previous

Next >

Press ESC then TAB or click outside of the code editor to exit

Correct



edX

[About](#)

[Affiliates](#)

[edX for Business](#)

[Open edX](#)

[Careers](#)

[News](#)

Legal

[Terms of Service & Honor Code](#)

[Privacy Policy](#)

[Accessibility Policy](#)

[Trademark Policy](#)

[Sitemap](#)

[Cookie Policy](#)

[Do Not Sell My Personal Information](#)



© 2023 edX LLC. All rights reserved.

深圳市恒宇博科技有限公司 [粤ICP备17044299号-2](#)