3. Activation Functions

Project due Nov 5, 2020 05:29 IST *Completed*

The first step is to design the activation function for each neuron. In this problem, we will initialize the network weights to 1, use **ReLU** for the activation function of the hidden layers, and use an identity function for the output neuron. The hidden layer has a bias but the output layer does not. Complete the helper functions in `neural_networks.py`, including `rectified_linear_unit` and `rectified_linear_unit_derivative`, for 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 NeuralNetwork()`, the initialization of weights has now been changed to an initialization as `float` rather than `int`. You could either re-download the updated project release mnist.tar.gz, or change the corresponding lines in `part2-nn/neural_nets.py` to the following, where we have added decimal points to all numbers in the initialization:

```
class NeuralNetwork():

    def __init__(self):

        # DO NOT CHANGE PARAMETERS (Initialized to floats instead of ints)
        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

```
def rectified_linear_unit(x):
    """ Returns the ReLU of x, or the maximum between 0 and x."""
    # TODO
    if x > 0:
        return x
    else:
        return 0
```

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

Correct

Test results

See full output

```
CORRECT
```

See full output

Submit You have used 2 of 50 attempts

Taking the Derivative

2.0/2.0 points (graded)
Now implement its derivative so that we can properly run backpropagation when training the net. Note: we will consider the derivative at zero to have the same value as the derivative at all negative points.

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

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

```
def rectified_linear_unit_derivative(x):
    """ Returns the derivative of ReLU."""
    # TODO
    if x<=0:
        return 0
    else:
        return 1
```

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

Correct

Test results

See full output

CORRECT

See full output

Submit You have used 2 of 25 attempts

Discussion

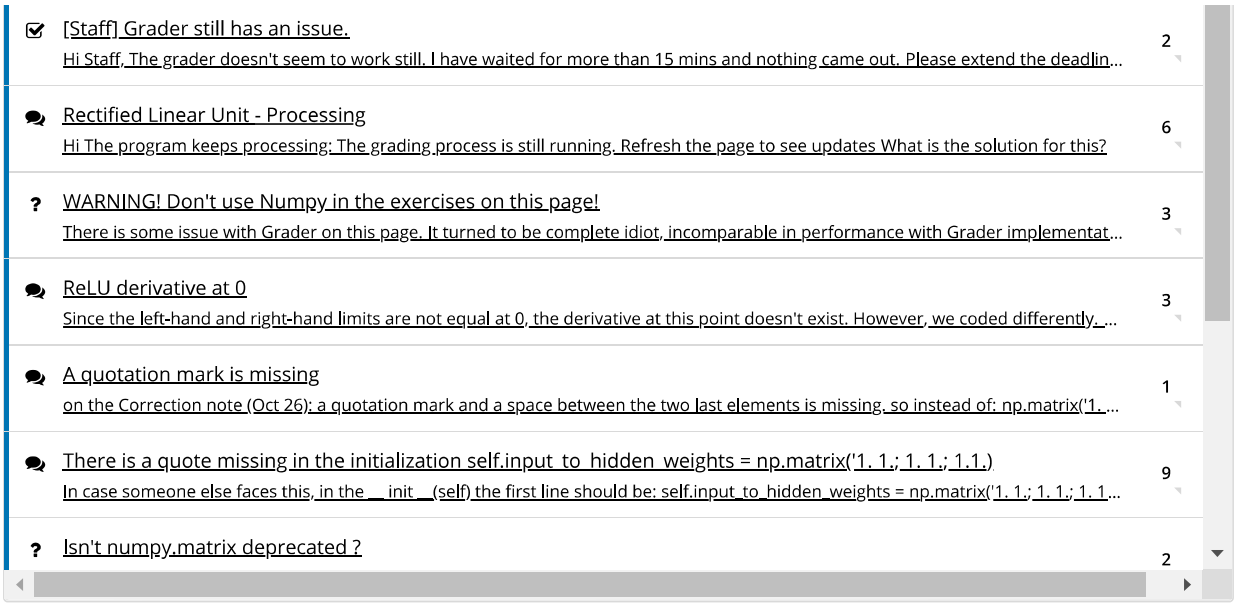
Hide Discussion

Topic: Unit 3 Neural networks (2.5 weeks):Project 3: Digit recognition (Part 2) / 3. Activation Functions

Add a Post

Show all posts by recent activity

- Tip for Taking the Derivative — Community TA — 5
- 25th Submission keeps processing — Dear staff, For Quest 2 on this page, its final submission 25th is not being taken even after correct grading for 24 submissions — 17
- HINT: if grader processing for a long time — Hi, The issue with the grader taking a long time to process the answer is related to the numeric type used in the comparison statem... — 1 new
- Grader is continoulsy processing for last 30 mins...for 2 question on this page.Please check for grader — Dear Staff, Grader is continuously processing for last 30 mins...for 2 question on this page.Please check for grader — 25
- Strange behavior on RELU grader — For my code I can literally write "return True" and then the submission results say that my output is all "False" Similarly, I can try sub... — 4
- Grader stuck at Processing — I have tried resetting, refreshing everything. Still its stuck at processing. — 3



Previous Next