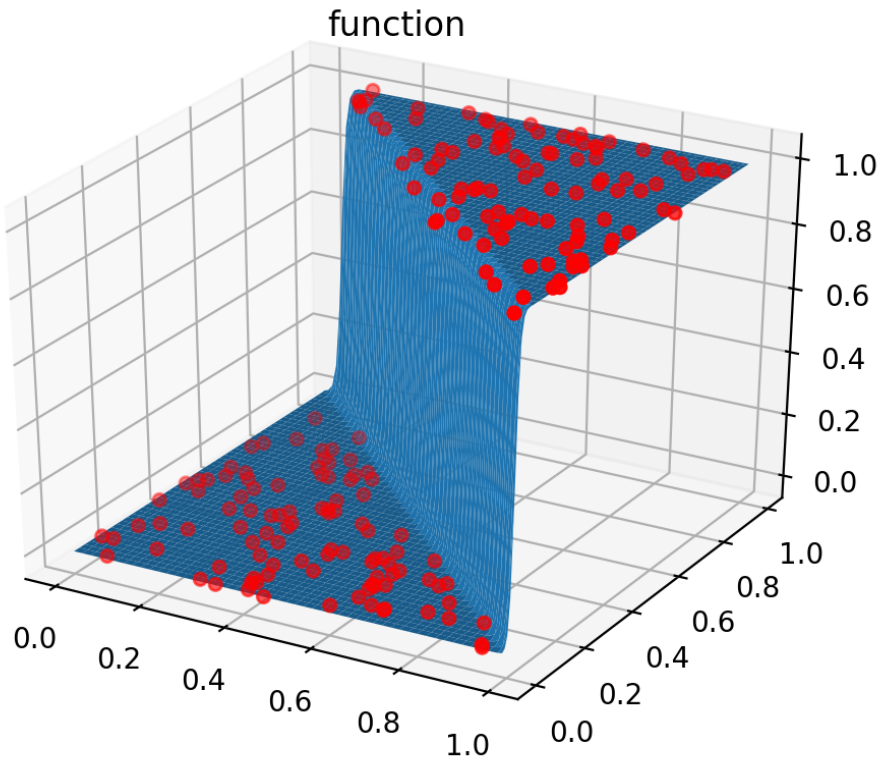
Homework 3

Download the code and data. You can run the program by putting all of the files into a common directory and typing

```
python3 my_nn.py
```

This program is a neural net, written in numpy, adopted from this tutorial.

The file `my_nn.py` itself merely contains the top-level code and the neural net configuration. Most of the code for the neural net is in `np_nn.py`. The file `surface_env.py` contains support for visualizing output. You should run your code from an environment that supports matplotlib, because most of your feedback will be graphical in nature. The output on dataset1 (how it is set up when you download it) looks something like this:



1. [10 points] The cost function is only briefly alluded to in the main backpropagation method of `np_nn.py`, via its first derivative. Comment this line out and add above it the line

```
# This is derivative of the original cost function, which is ...
```

And replace "`...`" with the common name of this cost function

2. (15 points) Add support for users to pass their own cost function to the trainer. That is, modify the method

```
train(X, Y, nn_architecture, epochs, learning_rate)
```

so that it takes an additional parameter:

```
train(X, Y, nn_architecture, epochs, learning_rate, df)
```

Where `df` is a function that returns the first derivative of a cost function. To do this, you must also add this parameter to any function that `train` calls on the way to invoking backprop.

3. [25 points] Define two functions: one that returns the first derivative of the original cost function (you can make that the default argument to `train` if you like), and another that returns the first derivative of the MSE cost function.

4. [30 points] Currently, to define a neural network, you define a list, e.g., as follows (see `my_nn.py`)

```
{"input_dim": 2, "output_dim": 4, "activation": "relu"},
{"input_dim": 4, "output_dim": 4, "activation": "relu"},
{"input_dim": 4, "output_dim": 1, "activation": "sigmoid"}
```

This creates a four-layer dense network. Modify this to support max-pooling by adding a new field to each layer definition, called type. And define the types according to this example:

```
{"input_dim": 2, "output_dim": 4, "activation": "relu", "type":
"dense"},
{"input_dim": 4, "output_dim": 6, "activation": "relu", "type":
"dense"},
{"input_dim": 6, "diameter": 2, "stride": 2, "type":
"maxpool"},
```

```
{"input_dim": 3, "output_dim": 1, "activation": "sigmoid",
"type": "dense"}
```

Where "dense" is the same type as before and "maxpool" is a max pool layer. Here "diameter" represents the number of pixels to the left and right of the current pixel to add to each pool (if they exist). So, for instance, in the maxpool layer above, the input has 6 nodes 1, 2, 3, 4, 5, 6. The diameter is 2, so the pooling at position 3 would take 1,2,3,4,5, and 6 it would be 4,5,6 and at two it would be 1,2,3,4. The stride is also two, so we would only compute the values for the pools at 1,3, and 5.

Note that, to support this fully, you must change nearly all the methods in np_nn.py, including forward and back propagation, and weight updating methods. Note, for instance, that the max pool layer itself has no weights or biases and so no parameters that need to be updated. It does, however, contribute its (piecewise) gradient to backpropagation.

5. [20 points] Write four modified versions of my_nn.py: my_nn_1.py -- my_nn4.py. Each one should be optimized for predicting the correspond datasets 1--4.

Submit only your versions of np_nn.py and my_nn.py: my_nn_1.py -- my_nn4.py.

28.57 % 2 of 7 topics complete

dataset1
CSV File

dataset2
CSV File

dataset3
CSV File

dataset4
CSV File

my_nn
PY File

np_nn
PY File
Updated ✓

surface_env
PY File
●