University of San Francisco

Physics 302 – Scientific Computation and Machine Learning
Spring, 2020

HW02

Due: Tuesday, October 20 at 11 PM

**XOR, Fully-connected ANN, and Nonlinear Boundaries**

Part I

Using the XOR gate truth table as the training set for the 2-layer-3-neuron network. Please use the notation in LectureSlides Week8-1 to implement this neural network.

The class `NeuralNetwork` should have five methods:

```
def __init__(self, layers, activation='sigmoid'):
     '''
     docstring
     '''
     ...


def fit(self, X, y, learning_rate=0.2, steps=100000, tol=
1e-2):
     '''
     docstring
     '''
     ...


def find_RMS_error(self, X, y):
     '''
     docstring
     '''
     ...
     return RMS_err
```

```
    def predict(self, x):
        '''
        docstring
        '''
        ...

        return prediction

    def visual_NN_boundaries(self, Nsamp=2000):
        '''
        docstring
        '''
        ...
```

Separately, outside the class definition, you should write four other functions:

```
sigmoid(x)

sigmoid_prime(x)

tanh(x)

tanh_prime(z)
```

The main program should look like this. In fact this is how I will test your code:

```
nn = NeuralNetwork([2,2,1], activation='sigmoid')

X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])

y = np.array([0, 1, 1, 0])

nn.fit(X, y, epochs=20000)
```

<u>About the RMS error and the tolerance</u>

Find the "average error" by finding the RMS error for all the training data (first adding the square of the error for each entry in `X`, then dividing the sum by the number of entries in `y` (the target), and finally taking square root). If at some point the RMS error is less than `tol`, the training is considered a success, and you can break out of the `for` loop. The following statement should be printed at the end:

```
NN training succeeded!
```

Otherwise, print

```
NN training failed.
```

<u>Reporting Training Progress</u>

For Every 1/10 of the specified number of training steps, check the RMS error by applying the neural net to the training data, and print (for this example, I specified 20,000 steps):

```
step: 0
Training Results(data, prediction, expected):
[0 0], 0.79458, 0
[0 1], 0.80081, 1
[1 0], 0.79900, 1
[1 1], 0.82609, 0
RMS_err: 0.59031

step: 2000
Training Results(data, prediction, expected):
[0 0], 0.00309, 0
[0 1], 0.95839, 1
[1 0], 0.95579, 1
[1 1], 0.00241, 0
RMS_err: 0.03042

step: 4000
Training Results(data, prediction, expected):
[0 0], 0.00120, 0
[0 1], 0.97355, 1
[1 0], 0.97089, 1
[1 1], 0.00008, 0
RMS_err: 0.01968

step: 6000
Training Results(data, prediction, expected):
[0 0], -0.00043, 0
[0 1], 0.97812, 1
[1 0], 0.97795, 1
[1 1], -0.00026, 0
RMS_err: 0.01553
```

```
step: 8000
Training Results(data, prediction, expected):
[0 0], 0.00025, 0
[0 1], 0.98151, 1
[1 0], 0.98126, 1
[1 1], 0.00033, 0
RMS_err: 0.01317

step: 10000
Training Results(data, prediction, expected):
[0 0], 0.00014, 0
[0 1], 0.98384, 1
[1 0], 0.98327, 1
[1 1], -0.00037, 0
RMS_err: 0.01163

step: 12000
Training Results(data, prediction, expected):
[0 0], 0.00021, 0
[0 1], 0.98536, 1
[1 0], 0.98500, 1
[1 1], 0.00043, 0
RMS_err: 0.01048

step: 14000
Training Results(data, prediction, expected):
[0 0], 0.00023, 0
[0 1], 0.98652, 1
[1 0], 0.98633, 1
[1 1], 0.00020, 0
RMS_err: 0.00960

NN training succeeded!
```

Depending on your implementation (e.g., learning rate), the number of training steps for you may or may not agree with mine.

Finally, apply the trained neural network to 2000 test data points. Let's call the *i*th data point `x`; `x[0]` and `x[1]` should be between [0, 1] for all values of *i*. Use the results to visualize the boundary.

Part II

Now use this training set:

```
X = np.array([[0, 0],
              [0, 1],
              [0.5, 1],
              [0, 0.5],
              [1, 0],
              [1, 1]])

y = np.array([0, 0, 0, 1, 1, 1])
```

You may consider increase the complexity of your architecture, e.g., to `[2,2,2,1]`.

Apply your trained neural net to 2000 test data points (all coordinates should be between 0 and 1) to visualize the boundary.

Bonus Part

Live update the training process.