Systems Biology – Exercises

Week 10: Neural Network Part (1)

Neural Network

- As the name implies, we investigate how to implement a network of conceptualized neurons. This idea of neuron is derived from the human brain, which has roughly 86 billion neurons with connections amounting to between 1014 and 1015, called synapses. Today, we will work with four neurons and 16 synapses.
- The exercise today is about a simple neural network (one Neuron) and propagation mechanism. The information flow is firstly from input to output, and weight adjustments are performed for each new iteration of a training process. The neural network training is done by comparing the output of the neurons with a reference data set. The discrepancy is calculated and used to determine the weights for the input layer of the next training iteration.

Mathematical Representation of a Neuron

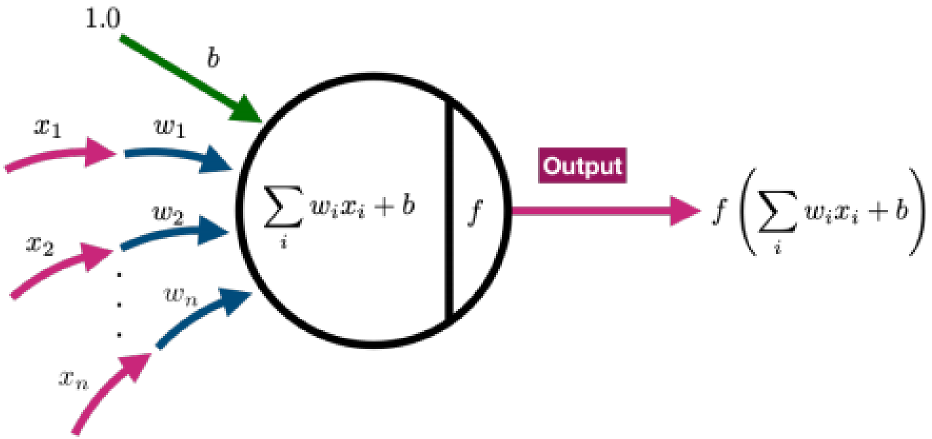


Figure 1: Mathematical Representation of a Neuron

Activation functions

- In Figure 1, we can see the mathematical representation of one neuron. To form an output, an activation function is needed (*f*). A few common types of these functions are shown below.

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$$

Equation 1

$$f(x) = \tanh(x) = 2\sigma(2x) - 1$$

Equation 2

$$f(x) = max(0, x) = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases}$$

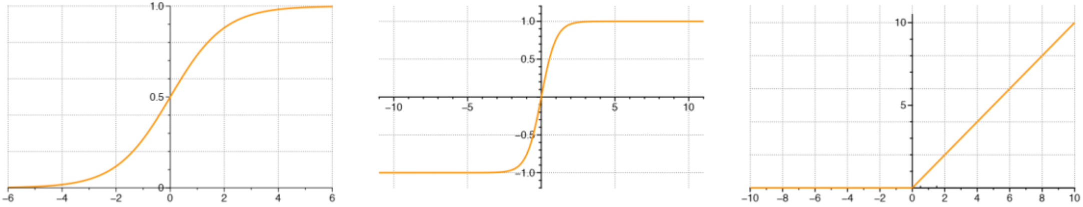
Equation 3

$$f(x) = \begin{cases} x & if\ x > 0 \\ 0.01x & otherwise \end{cases}$$

Equation 4

Visualization of Activation functions

- Equation 1: Sigmoid function (Figure 2a)
- Equation 2: Hyperbolic tangent (Figure 2b)
- Equation 3: Rectifier Linear Unit, or ReLU (Figure 2c)
- Equation 4: Leaky ReLu (used to compensate the ReLU for neurons becoming inactive during training)

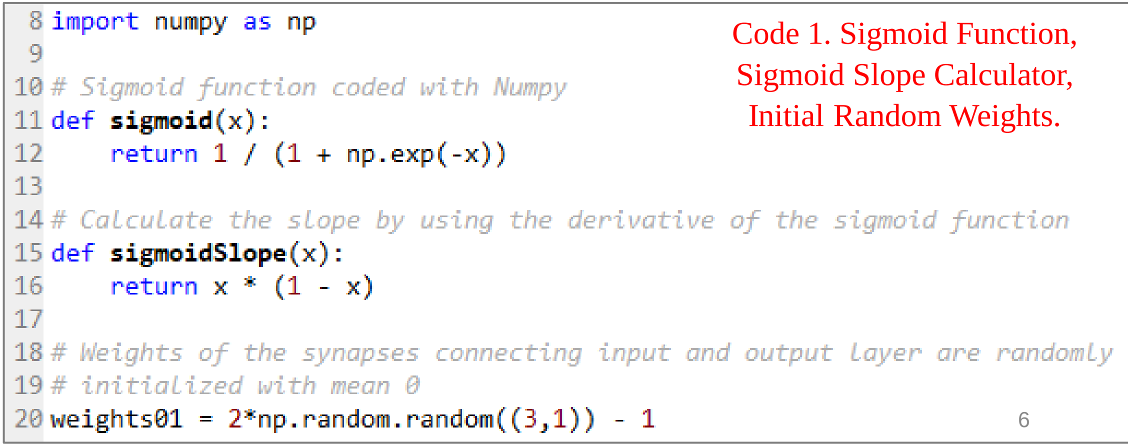


(a) Sigmoid Function (b) Hyperbolic Tangent Function (c) Rectifier Linear Unit Function

Figure 2: Different Activation Functions (Equations 1 to 3)

Setting the sigmoid function

- Begin by writing the following lines of code into a new file. These few lines on this page give you a starting point for setting the sigmoid function, its slope for training, and the initial random weights of the input layer.



```
import numpy as np

# Sigmoid function coded with Numpy
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Calculate the slope by using the derivative of the sigmoid function
def sigmoidSlope(x):
    return x * (1 - x)

# Weights of the synapses connecting input and output layer are randomly
# initialized with mean 0
weights01 = 2*np.random.random((3,1)) - 1
```

Code 1. Sigmoid Function, Sigmoid Slope Calculator, Initial Random Weights.

Exercise 1

- Provide an input matrix and a reference output vector in the form of Equation 5. Use np.array() for the input and np.array().T for the vector output. Write both the matrix and the vector independently of one-another.

Equation 5:

$$\text{inputData} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \text{referenceOutput}$$

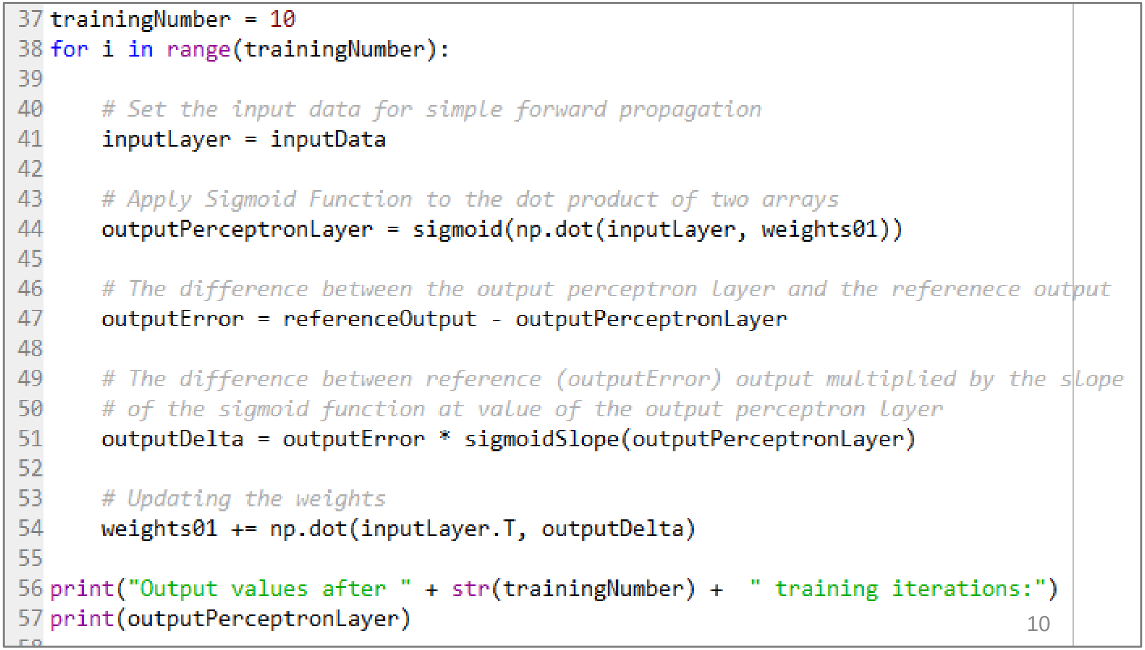
Exercise 2

- Set a seed of 1 for the random function to get consistent values for **weights01**. Note that we are using Numpy’s random generator.

The Neural Network as Python Code

- Let’s have a look at the following code, implementing the very simple neural networks. It passes an input matrix and tries to match it to the given output. It has in effect one layer of neurons (perceptron layer), which is the output.
- The Sigmoid function is used to convert weight values into what we can interpret as probabilities (between 0 and 1). The values in outputError denotes the difference between the reference data and the actual output and uses the slope of the Sigmoid function to calculate outputDelta, the values to update the weights for the next iteration.

The Neural Network as Python Code



```python
trainingNumber = 10
for i in range(trainingNumber):

    # Set the input data for simple forward propagation
    inputLayer = inputData

    # Apply Sigmoid Function to the dot product of two arrays
    outputPerceptronLayer = sigmoid(np.dot(inputLayer, weights01))

    # The difference between the output perceptron layer and the referenece output
    outputError = referenceOutput - outputPerceptronLayer

    # The difference between reference (outputError) output multiplied by the slope
    # of the sigmoid function at value of the output perceptron layer
    outputDelta = outputError * sigmoidSlope(outputPerceptronLayer)

    # Updating the weights
    weights01 += np.dot(inputLayer.T, outputDelta)

print("Output values after " + str(trainingNumber) +  " training iterations:")
print(outputPerceptronLayer)
```

Exercise 3

- Once you have successfully included the input and output to run the code, find a value of trainingNumber so that the output equals that of the reference data, when rounding for two decimal points. Use np.round(array, decimals = 2) for easier inspection.

Exercise 4

- Run the same code with different sets of outputs as in Equation 6 and 7.

Equation 6:

$$\text{inputData} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \text{referenceOutput}$$

Equation 7:

$$\text{inputData} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \text{referenceOutput}$$

Homework

- Due next Wednesday (17:00, 8th, Dec. 2021) electronically to manaba+R
- File format: YourStudentID_Week10_n.py/pdf (ID without hyphen, e.g., 12345678901_Week10_n.py).
- Your code must include your own comments for all code sections. Go line-by-line, if necessary. Comments in your program must be full sentences and reflect your understanding of the code.

Q1. Complete Python code for the simple neural network (one Neuron) and propagation mechanism (Exercises 1, 2, and 3). → YourStudentID_Week10_01.py

Q2. How can you change the slope of the Sigmoid function in the source code of Q1 to reduce the number of steps in trainingNumber to get to a result satisfying conditions in Exercise 3? In this question, you should refer to Figure 3, indicate the chosen Sigmoid function using the comments in *.py file, and modify the source code to incorporate your chosen Sigmoid function.
Q2 → YourStudentID_Week10_02.py

- Q3. What happens if the Sigmoid function has too steep of a slope (e.g.$\sigma(5x), \sigma(10x)$, $\sigma(100x)$)? Write a report for Q3, the text should have about (but no less than) 100 words.
Q3→YourStudentID_Week10_03.pdf

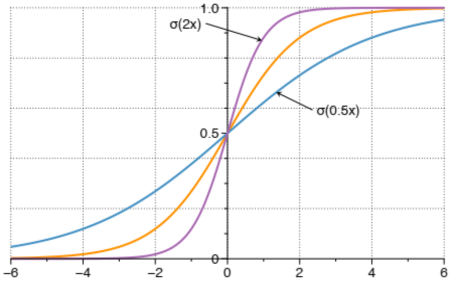


Figure 3: Sigmoid Function with Varying Steepness

Systems Biology – Exercises

Week 11: Neural Network Part (2)