Systems Biology – Exercises

Week 11: Neural Network Part (2)

Neural networks with hidden layers

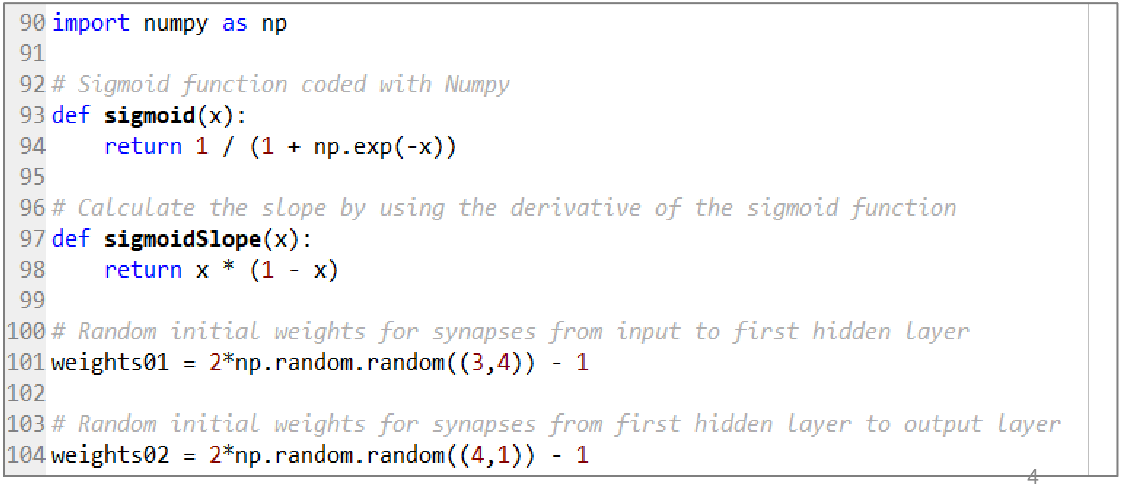
- Today we will be looking at the follow up of last week's simple neural network. Previously, the neural network only contained a single layer of neurons. Our next step in more complex neural networks introduce hidden layers.
- In this case, errors are calculated at the output (same as before), but then distributed backwards through the network layers. As we have only two neuron layers (the hidden layer and the output layer) in our example, the backward propagation is performed from the outputPerceptronLayer to the hiddenLayer.

Implementing a Multilayer Perceptron

- Create a new file for the multilayer neural network.
- These few lines written on the next page give you a starting point for setting the sigmoid function, its slope for training, and the initial random weights of the input layer.
- We are using the same sigmoid function, but have to change how the weights are calculated. Note that the dimensions of the weights also change.

Code 1

- Sigmoid Function, Sigmoid Slope Calculator, Initial Random Weights



```python
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)

# Random initial weights for synapses from input to first hidden layer
weights01 = 2*np.random.random((3,4)) - 1

# Random initial weights for synapses from first hidden layer to output layer
weights02 = 2*np.random.random((4,1)) - 1
```

Exercise 1

- Provide an input matrix and a reference output vector in the form of Equation 1. 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. The implementation is exactly the same as last week.

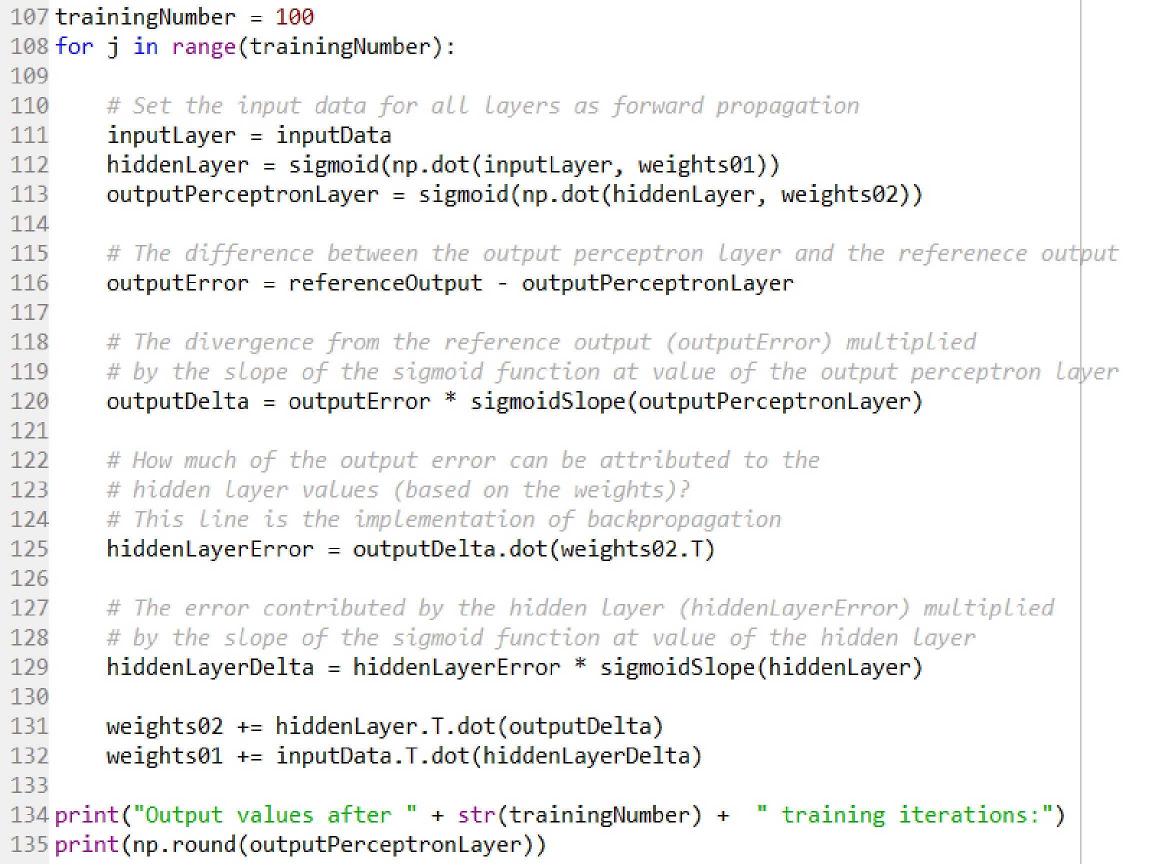
$$\text{inputData} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \text{referenceOutput} \tag{1}$$

Exercise 2

- Set a seed of 42 for the random function to get consistent values for weights01 and weights02.
- Note that we are using Numpy’s random generator. You can again use the same line of code from last week.

A Multilayer Neural Network as Python Code

- The following code (on the next page) implements one of the very simplest forms of a multilayer neural network. It passes one input variables (12 values) and tries to match it to the given output. It has in effect two perceptron layers, the hidden layer and the output layer.
- 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 error of the hidden layer is updated based on the divergence of the output from the reference data.



```python
trainingNumber = 100
for j in range(trainingNumber):

    # Set the input data for all layers as forward propagation
    inputLayer = inputData
    hiddenLayer = sigmoid(np.dot(inputLayer, weights01))
    outputPerceptronLayer = sigmoid(np.dot(hiddenLayer, weights02))

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

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

    # How much of the output error can be attributed to the
    # hidden layer values (based on the weights)?
    # This line is the implementation of backpropagation
    hiddenLayerError = outputDelta.dot(weights02.T)

    # The error contributed by the hidden layer (hiddenLayerError) multiplied
    # by the slope of the sigmoid function at value of the hidden layer
    hiddenLayerDelta = hiddenLayerError * sigmoidSlope(hiddenLayer)

    weights02 += hiddenLayer.T.dot(outputDelta)
    weights01 += inputData.T.dot(hiddenLayerDelta)

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

Exercise 3

- The simple neural network from last week had some difficulties approximating the following output (Equation 2). Can this problem be solved by the Multilayer Neural Network?

$$\text{inputData} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \text{referenceOutput} \qquad (2)$$

Exercise 4

- Rewrite the loop used in your example to stop at a given error level. The type of loop used to run a fixed number of times is called counter-based loop, the type of loop that stops when a condition is met is called sentinel-based or event-based loop. Use the following code snipped to prompt the sentinel: np.mean(np.abs(outputError))

Homework

- Due next Wednesday (17:00, 15th, Dec. 2021) electronically to manaba+R
- File format: YourStudentID_Week11_n.py/pdf (ID without hyphen, e.g., 12345678901_Week11_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. Draw the multilayer neural network as a diagram. The naming convention must follow the one used in the Python code. The final diagram should only include variable and function names. The diagram must be drawn with a computer.

Q2. How does the multilayer neural network compare to the single layer neural network? Take two problems (two different referenceOutput), one solvable by both and one only by multilayer neural network. Modify and extend the example source code provided in the slide to get the values of the mean output error (np.mean(np.abs(outputError))) for 900 iterations (both problems and both neural networks) and plot all four variables as a graph (using source code).

Q1: YourStudentID_Week11_01.pdf,

Q2: YourStudentID_Week11_02.py

Systems Biology – Exercises

Week 12: Swarm Intelligence