Machine Learning
COS6026-B

Prof. Rami Qahwaji
rsrqahwa@Bradford.ac.uk

10th Lab Session: Neural Networks in Python

For this week we will implement the KNN tutorial from:
https://iamtrask.github.io/2015/07/12/basic-python-network

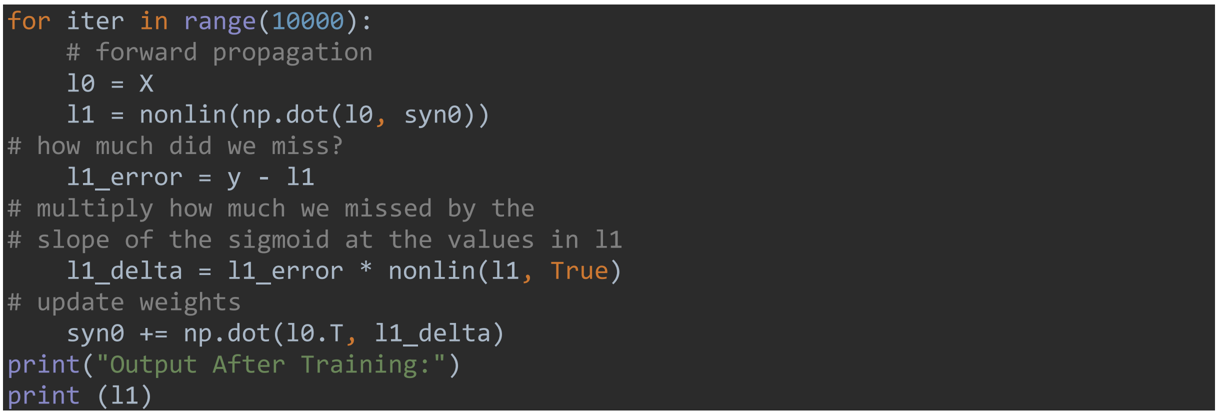
A neural network trained with backpropagation is attempting to use input to predict output. We will try to predict the output column from the 3-input columns.

Inputs			Output
0	0	1	0
1	1	1	1
1	0	1	1
0	1	1	0

The Python Code (NN with input layer and output layer – No hidden layer)

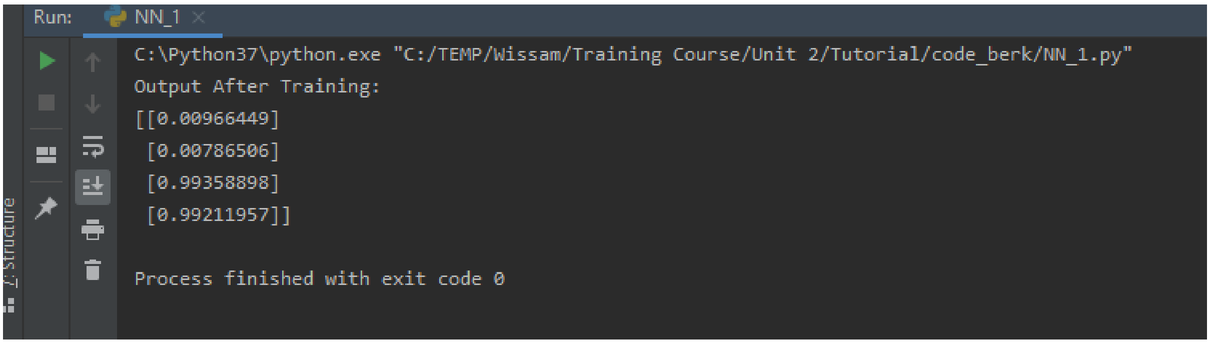
```
import numpy as np

# sigmoid function
def nonlin(x, deriv=False):
    if (deriv == True):
        return x * (1 - x)
    return 1 / (1 + np.exp(-x))
# input dataset
X = np.array([[0, 0, 1],
              [0, 1, 1],
              [1, 0, 1],
              [1, 1, 1]])
# output dataset
y = np.array([[0, 0, 1, 1]]).T
# seed random numbers to make calculation
# deterministic (just a good practice)
np.random.seed(1)
# initialize weights randomly with mean 0
syn0 = 2 * np.random.random((3, 1)) - 1
```



```python
for iter in range(10000):
    # forward propagation
    l0 = X
    l1 = nonlin(np.dot(l0, syn0))
# how much did we miss?
    l1_error = y - l1
# multiply how much we missed by the
# slope of the sigmoid at the values in l1
    l1_delta = l1_error * nonlin(l1, True)
# update weights
    syn0 += np.dot(l0.T, l1_delta)
print("Output After Training:")
print (l1)
```

Output:



```
C:\Python37\python.exe "C:/TEMP/Wissam/Training Course/Unit 2/Tutorial/code_berk/NN_1.py"
Output After Training:
[[0.00966449]
 [0.00786506]
 [0.99358898]
 [0.99211957]]

Process finished with exit code 0
```

Variable	Definition
X	Input dataset matrix where each row is a training example
y	Output dataset matrix where each row is a training example
l0	First Layer of the Network, specified by the input data
l1	Second Layer of the Network, otherwise known as the hidden layer
syn0	First layer of weights, Synapse 0, connecting l0 to l1.
*	Elementwise multiplication, so two vectors of equal size are multiplying corresponding values 1-to-1 to generate a final vector of identical size.
-	Elementwise subtraction, so two vectors of equal size are subtracting corresponding values 1-to-1 to generate a final vector of identical size.

x.dot(y)	If x and y are vectors, this is a dot product. If both are matrices, it's a matrix-matrix multiplication. If only one is a matrix, then it's vector matrix multiplication.

For detailed explanation of the code, check:
https://iamtrask.github.io/2015/07/12/basic-python-network

Tasks:

- Compare l1 after the first iteration and after the last iteration.
- Check out how l1_error changes as you iterate.
- Modify the code to add a hidden layer, send me your code to post it on Canvas!