Feedforward-OneCase-IncompleteCode1

March 25, 2021

Nicholas Paisley

[1]:
```
import numpy as np

#Create loop.

# 5  [H1 H2] 2

# +
# +  [+
# +  [+  +]
# +  [+  +]  +
# +  [+  +]  +
#      ^
#      |
#   black box

def feed_forward(features,w1,b1,w2,b2,w3,b3,act):
    Olayer1 = np.matmul(w1, features) + b1 #Computing next row of nodes/
 ↪features (4) [Hidden Layer 1]
    Olayer2 = np.matmul(w2, Olayer1) + b2  #Computing next row of nodes/
 ↪features (3) [Hidden Layer 2]
    Olayer3 = np.matmul(w3, Olayer2) + b3  #Computing next row of nodes/
 ↪features (2) Node/Features
    #Giving the option to choose either sigmoid and/or RELU
    if int(act) > 0:
        Output1 = sigmoid(Olayer3)
    else:
        Output1 = RELU(Olayer3)
    return(Output1)

#Definitions for the Sigmoid or RELU

def sigmoid(x): #Creating a definition for Sigmoid function
    return 1/(1+np.exp(-x)) #Sigmoid function

def RELU(x):
```

```python
    if all(x) > 0: #all() function returns True if all items in an iterable are␣
↪true
        return np.around(x,decimals=7) #around() function rounds the given␣
↪number to the amount of decimals given
    else:
        return 0

#=====================================================

    return targets_predicted

#======================================================

print('Starting ...')

## Set up training data with just one case

#Given the features (the 5 starting X1,X2,X3,X4,X5). We are starting out with a␣
↪(5x1) matrix (Input layer).
features = np.array([[0],
                     [0],
                     [0],
                     [0],
                     [0]])

targets_observed = np.array([0,1]) #The numbers we want to achieve after black␣
↪box calculations. The final answers for the last 2 features.
number_of_features,number_of_cases = 5,1 #Starting with 5 nodes (features) and␣
↪1 case (basically stating we have a (5x1)).

weights_1 = np.random.rand(4,5) #Setting the dimensions of the matrix for our␣
↪random weights (rows by columns)
biases_1 =  np.random.rand(4,1) #Setting the dimensions of the matrix for our␣
↪baises (rows by columns)

#print("Weights_1 = " , weights_1)

#Input Layer E R5 ----> Output Layer E R4

#       (4 x 5)                  (5 x 1)    (4 x 1)
#w1,1 w1,2 w1,3 w1,4 w1,5        x1         b1        w1,1x1 + w1,2x2 + w1,3x3 +␣
↪w1,4x4 + w1,5x5 + b1
#w2,1 w2,2 w2,3 w2,4 w2,5   *    x2    +    b2    =   w2,1x1 + w2,2x2 + w2,3x3 +␣
↪w2,4x4 + w2,5x5 + b2 = (4 x 1) matrix with new features after calculations␣
↪for Hidden Layer 1
```

```
#w3,1 w3,2 w3,3 w3,4 w3,5        x3        b3        w3,1x1 + w3,2x2 + w3,3x3 +␣
↪w3,4x4 + w3,5x5 + b3
#w4,1 w4,2 w4,3 w4,4 w4,5        x4        b4        w4,1x1 + w4,2x2 + w4,3x3 +␣
↪w4,4x4 + w4,5x5 + b4
#                                x5

#This process continues until we hit our destination of 2 nodes/features.

weights_2 = np.random.rand(3,4)
biases_2 = np.random.rand(3,1)

#(3 x 4) weights * (4 x 1) features + (3 x 1) biases = (3 x 1) matrix for new␣
↪features after calculations for Hidden Layer 2

#      (3 x 4)           (4 x 1)   (3 x 1)
#w1,1 w1,2 w1,3 w1,4       x1        b1       w1,1x1 + w1,2x2 + w1,3x3 + w1,4x4␣
↪+ b1
#w2,1 w2,2 w2,3 w2,4    *  x2    +   b2   =   w2,1x1 + w2,2x2 + w2,3x3 + w2,4x4␣
↪+ b2 = (3 x 1) matrix with new features after calculations for Hidden Layer 2
#w3,1 w3,2 w3,3 w3,4       x3        b3       w3,1x1 + w3,2x2 + w3,3x3 + w3,4x4␣
↪+ b3
#                          x4

weights_3 = np.random.rand(2,3)
biases_3 = np.random.rand(2,1)

#print("Weight_3 =" , weights_3)

#(2 x 3) weights * (3 x 1) features + (2 x 1) = (2 x 1) matrix for new features␣
↪after calculations for 2 features (our ending)

#   (2 x 3)        (3 x 1)   (2 x 1)
#w1,1 w1,2 w1,3      x1        b1       w1,1x1 + w1,2x2 + w1,3x3 + b1
#w2,1 w2,2 w2,3   *  x2    +   b2   =   w2,1x1 + w2,2x2 + w2,3x3 + b2 = (2 x 1)␣
↪matrix with new features after calculations for Hidden Layer 2
#                    x3

act = input("sigmoid(1) or RELU(-1)?") #Using an input to allow you to call␣
↪either Sigmoid or RELU

#Calling the "feed_foward" definition and saving the result as␣
↪"Targets_Predicted"
Targets_Predicted = feed_forward(features,weights_1,biases_1,
                                      weights_2,biases_2,
                                      weights_3,biases_3,act)
```

```
print('Features : ',features) #our beginning features that we are given to us
print(' Targets : ', targets_observed) #printing the "target_observed"␣
↪variables [0,1] <- This is what we want our results to be.
print(' Targets predicted : ', Targets_Predicted) #Prints out our predict␣
↪values from either the Sigmoid and/or RELU function.
```

```
Starting …

sigmoid(1) or RELU(-1)? 1

Features :  [[0]
 [0]
 [0]
 [0]
 [0]]
 Targets :  [0 1]
 Targets predicted :  [[0.97627951]
 [0.98872001]]
```