Diagram

Description automatically generated

n = 5

X = Diagram

Description automatically generated

w = Graphical user interface, diagram

Description automatically generated

n by 4 (c)

z11 = w11x1 + w21x2 + w31x3 + b1

z12 = w12x1 + w22x2 + w32x3 + b2

z13 = w13x1 + w23x2 + w33x3 + b3

z14 = w14x1 + w24x2 + w34x3 + b4

h1 = S(z11), where S is the sigmoid activation function

h2 = S(z12)

h3 = S(z13)

h4 = S(z14)

Diagram

Description automatically generated @ Graphical user interface, diagram

Description automatically generated + b1 b2 b3 b4

The matrix of Z1:

zi1    zi2 zi3  zi4  
x11w11+x12w21+x13w31+b1  x11w12+x12w22+x13w32+b2  x11w13+x12w23+x13w33+b3  x11w14+x12w24+x13w34+b4  
x21w11+x22w21+x23w31+b1  x21w12+x22w22+x23w32+b2  x21w13+x22w23+x23w33+b3  x21w14+x22w24+x23w34+b4  
x31w11+x32w21+x33w31+b1  x31w12+x32w22+x33w32+b2  x31w13+x32w23+x33w33+b3  x31w14+x32w24+x33w34+b4  
x41w11+x42w21+x43w31+b1  x41w12+x42w22+x43w32+b2  x41w13+x42w23+x43w33+b3  x41w14+x42w24+x43w34+b4  
x51w11+x52w21+x53w31+b1  x51w12+x52w22+x53w32+b2  x51w13+x52w23+x53w33+b3  x51w14+x52w24+x53w34+b4

Z2 = S(Z1) @ W2 = S(Z1) @ [w1 w2 w3 w4] = w1h1 + w2h2 + w3h3 + w4h4 + c

y^ = S(Z2)

*# -\*- coding: utf-8 -\*-  
"""  
author: Xingyu Chen  
"""  
  
import* numpy *as* np  
  
X = np.array([[3, 4, 8], [6, -2, -1], [1, 2, 7], [-2, 5, -4], [-5, 9, -6]])  
print("The input is:\n", X)  
*# print(X.shape) ## 4 by 2 (n by c)*

Y = np.array([[0], [1], [0], [1], [0]])  
print("The target value are:\n", Y)  
*# print(Y.shape)  
  
## Size of each input x*X\_examples = X.shape[0] *## each x has 4 rows (examples)*print("The number of input vectors are:\n", X\_examples)  
X\_vectorSize = X.shape[1]  
  
numOutputs = 1  
NumHidden = 3  
  
W1\_x = np.array([[1, 2, 0, 1], [-1, -2, 1, 0], [0, 1, -1, 2]])  
print("The W1 weights for the x's are:\n", W1\_x)  
*# print(W1\_x.shape)  
## Here, W1\_x is 2 by 3 (c by h)*bs = np.array([[8, 9, 10, 11]]) *## bs shape should be 1 by h*print("The b are: \n", bs)  
*# print(bs.shape)*Z1 = X @ W1\_x + bs *# should be shape n by h*print("The Z1 are: \n", Z1)  
*# print(Z1.shape)*W2\_h = np.array([[1], [-2], [0], [-1]])  
*# print(W2\_h.shape)*print("The W2\_h are: \n", W2\_h)  
*##W2\_h FROM hidden units into the output.  
## The shape must be h by 1*c = 0 *## This is the shape of the output which is 1 by 1*A\_Z1 = np.maximum(0, Z1) *##ReLU, shape n by h*Z2 = (A\_Z1 @ W2\_h) + c *## n by h @ h by 1  
## Z2 shape is n by 1*y\_hat = np.maximum(0, Z2) *## ReLU #shape n by 1*print("The output is:\n", y\_hat)  
*# print(y\_hat.shape)*

The input is:

[[ 3 4 8]

[ 6 -2 -1]

[ 1 2 7]

[-2 5 -4]

[-5 9 -6]]

The target value are:

[[0]

[1]

[0]

[1]

[0]]

The number of input vectors are:

5

The W1 weights for the x's are:

[[ 1 2 0 1]

[-1 -2 1 0]

[ 0 1 -1 2]]

The b are:

[[ 8 9 10 11]]

The Z1 are:

[[ 7 15 6 30]

[ 16 24 9 15]

[ 7 14 5 26]

[ 1 -9 19 1]

[ -6 -25 25 -6]]

The W2\_h are:

[[ 1]

[-2]

[ 0]

[-1]]

The output is:

[[0]

[0]

[0]

[0]

[0]]