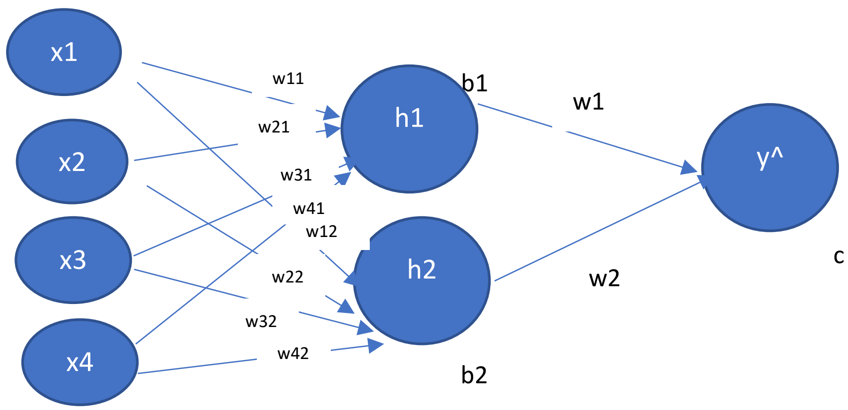
1) Let's use this NN just as an example to look at. We will use differing values of n, but we will stick with 4 columns and 2 hidden units.



Let's just start by looking at TWO input vectors only.

X = [[x11, x12, x13, x14],
[x21, x22, x23, x24]]

X is n by c

Y = [[y1] , [y2]]

W1 (c by h)
W1 = [[w11 , w12],
[w21 , w22],
[w31, w32],
[w41, w42]]

W2 = [[w1], [w2]] (h by o)

b = [b1 , b2] (1 by h)
c= [[c]]

Z1 = X@W1 which is shape n by h

Z1 = [z11=x11(**w11**) + x12(w21) + x13(w31) + x14(w41) + b1 z12=x11(w12) + x12(w22) + x13(w32) + x14(w42) + b2
z21 =x21(**w11**) + x22(w21) + x23(w31) + x24(w41) + b1 z22=x21(w12) + x22(w22) + x23(w32) + x24(w42) + b2]

shape n by h

H1= [h1**1**=Sig(z11) h1**2**=Sig(z12)
h2**1**=Sig(z21) h2**2**=Sig(z22)]

H1 shape is n by h

Z2 = [z(2)1 = h11w1 + h12w2 + c
z(2)2 = h21w1 + h22w2 + c]

Z2 shape: n by o

The Loss Function 1/2(y^ - y)^2

Average Loss: 1/n SUM **1/2(y^ - y)^2**
Total Loss: SUM **1/2(y^ - y)^2**

Y^ = [y^1 = Sig(z(2)1)
y^2 = Sig(z(2)2)] shape n by o

L1 = **1/2(y^1 – y1)^2**
L2 = **1/2(y^2 – y2)^2**
TotalLoss = L1 + L2 = **1/2(y^1 – y1)^2 +1/2(y^2 – y2)^2**

Gradient: Derivatives for W1, W2, B, and C

What does **w11** affect? Answer: **z11 and z21**
What does z11 affect? **h11**
What does h11 affect? **z(2)1**
What does z(2)1 affect? **y^1**
What does y^1 affect? TotalLoss (TL)

dTL/dw11 = dz11/**dw11** * dh11/dz11 * dz(2)1/dh11 * **dy^1/dz(2)1* dTL/dy^1**
+
dTL/dw11 = dz21/**dw11** * dh21/dz21 * dz(2)2/dh21 * **dy^2/dz(2)2* dTL/dy^2**

x11 * (h11)(1 – h11) * w1 * (y^1)(1 – y^1) * (y^1– y1)
+
x21 * (h21)(1 – h21) * w1 * (y^2)(1 – y^2) * (y^2– y2)

dTL/dw21 = dz11/**dw21** * dh11/dz11 * dz(2)1/dh11 * **dy^1/dz(2)1* dTL/dy^1**
+
dTL/dw21 = dz21/**dw21** * dh21/dz21 * dz(2)2/dh21 * **dy^2/dz(2)2* dTL/dy^2**

x12 * (h11)(1 – h11) * w1 * (y^1)(1 – y^1) * (y^1– y1)
+
x22 * (h21)(1 – h21) * w1 * (y^2)(1 – y^2) * (y^2– y2)

dTL/dw31 = dz11/**dw31** * dh11/dz11 * dz(2)1/dh11 * **dy^1/dz(2)1* dTL/dy^1**
+
dTL/dw31 = dz21/**dw31** * dh21/dz21 * dz(2)2/dh21 * **dy^2/dz(2)2* dTL/dy^2**

x13 * (h11)(1 – h11) * w1 * (y^1)(1 – y^1) * (y^1– y1)
+
x23 * (h21)(1 – h21) * w1 * (y^2)(1 – y^2) * (y^2– y2)

dTL/dw41 = dz11/**dw41** * dh11/dz11 * dz(2)1/dh11 * **dy^1/dz(2)1* dTL/dy^1**
+
dTL/dw41 = dz21/**dw41** * dh21/dz21 * dz(2)2/dh21 * **dy^2/dz(2)2* dTL/dy^2**

x14 * (h11)(1 – h11) * w1 * (y^1)(1 – y^1) * (y^1– y1)
+
x24 * (h21)(1 – h21) * w1 * (y^2)(1 – y^2) * (y^2– y2)

----

dTL/dw12 = dz12/**dw12** * dh12/dz12 * dz(2)1/dh12 * **dy^1/dz(2)1* dTL/dy^1**
+
dTL/dw12 = dz22/**dw12** * dh22/dz22 * dz(2)2/dh22 * **dy^2/dz(2)2* dTL/dy^2**

x11 * (h12)(1 – h12) * w2 * (y^1)(1 – y^1) * (y^1– y1)
+
x21 * (h22)(1 – h22) * w2 * (y^2)(1 – y^2) * (y^2– y2)

dTL/dw22 = dz12/**dw22** * dh12/dz12 * dz(2)1/dh12 * **dy^1/dz(2)1* dTL/dy^1**
+
dTL/dw22 = dz22/**dw22** * dh22/dz22 * dz(2)2/dh22 * **dy^2/dz(2)2* dTL/dy^2**

x12 * (h12)(1 – h12) * w2 * (y^1)(1 – y^1) * (y^1– y1)
+
x22 * (h22)(1 – h22) * w2 * (y^2)(1 – y^2) * (y^2– y2)

dTL/dw32 = dz12/**dw32** * dh12/dz12 * dz(2)1/dh12 * **dy^1/dz(2)1* dTL/dy^1**
+
dTL/dw32 = dz22/**dw32** * dh22/dz22 * dz(2)2/dh22 * **dy^2/dz(2)2* dTL/dy^2**

x13 * (h12)(1 – h12) * w2 * (y^1)(1 – y^1) * (y^1– y1)
+
x23 * (h22)(1 – h22) * w2 * (y^2)(1 – y^2) * (y^2– y2)

dTL/dw42 = dz12/**dw42** * dh12/dz12 * dz(2)1/dh12 * **dy^1/dz(2)1* dTL/dy^1**
+
dTL/dw42 = dz22/**dw42** * dh22/dz22 * dz(2)2/dh22 * **dy^2/dz(2)2* dTL/dy^2**

x14 * (h12)(1 – h12) * w2 * (y^1)(1 – y^1) * (y^1– y1)
+
x24 * (h22)(1 – h22) * w2 * (y^2)(1 – y^2) * (y^2– y2)

Build this
[(y^1)(1 – y^1) * (y^1– y1)
(y^2)(1 – y^2) * (y^2– y2)]

We know that
Y = [y1 Y^ = [y^1
y2] y^2]

Sig(Y) = Y_error = Y^ - Y =
[(y1)(1 – y1) [y1^ - y1
(y2) (1 – y2)] y2^ - y2]

→ Sig(Y) * (Y_error) which is
[(y1)(1 – y1) (y1^ - y1)
(y2) (1 – y2) (y2^ - y2)]

Next, build
w1 * (y^1)(1 – y^1) * (y^1– y1)
w1 * (y^2)(1 – y^2) * (y^2– y2)
w2 * (y^1)(1 – y^1) * (y^1– y1)
w2 * (y^2)(1 – y^2) * (y^2– y2)

We know that
W2 = [w1
w2]

So, to build this we need
Sig(Y) * (Y_error) @ W2.T

[(y1)(1 – y1) (y1^ - y1)
(y2) (1 – y2) (y2^ - y2)] @ [w1 w2]

This is 2 by 1 @ 1 by 2 so we get a 2 by 2 →

[(y1)(1 – y1) (y1^ - y1)(w1) (y1)(1 – y1) (y1^ - y1)(w2)

(y2)(1 – y2) (y2^ - y2)(w1) (y2)(1 – y2) (y2^ - y2)(w2)]

Now we need the H sigmoid.
We know that we have

H = [h11 h12
h21 h22]

Let's get the derivative of the sigmoid of each….

H = [h11(1 – h11) h12(1 – h12)
h21(1 – h21) h22 (1 – h22)]

Next, let's multiply this by the following so we get what we need.
BE CAREFUL OF SHAPE AND GOAL – look at the derivatives!

D_Error_W =
[(y1)(1 – y1) (y1^ - y1)(w1) (y1)(1 – y1) (y1^ - y1)(w2)
(y2)(1 – y2) (y2^ - y2)(w1) (y2)(1 – y2) (y2^ - y2)(w2)]

Direct multiply H * D_Error_W =
[h11(1 – h11) (y1)(1 – y1)(y1^ - y1)(w1) h12(1 – h12) (y1)(1 – y1) (y1^ - y1)(w2)
h21(1 – h21)(y2)(1 – y2) (y2^ - y2)(w1) h22(1–h22) (y2)(1 – y2) (y2^ - y2)(w2)]

Finally – we get to the x's!
We have [[x11 x12 x13 x14]
[x21 x22 x23 x24]]

We need to do two things here. We need to create a matrix that is the SAME shape as W1. Why?

Next, we need to properly represent all the derivatives to the left (and the sums).

The solution is X.T @H_D_Error_W

Let's check it!

H_D_Error_W =

[h11(1 – h11) (y1)(1 – y1)(y1^ - y1)(w1)   h12(1 – h12) (y1)(1 – y1) (y1^ - y1)(w2)
h21(1 – h21)(y2)(1 – y2) (y2^ - y2)(w1)   h22(1–h22) (y2)(1 – y2) (y2^ - y2)(w2)]

X =

We have [[x11 x12 x13 x14]
[x21 x22 x23 x24]]

X.T is [[x11   x21]
[x12 x22]   @   [h11(1 – h11) (y1)(1 – y1)(y1^ - y1)(w1)   h12(1 – h12) (y1)(1 – y1) (y1^ - y1)(w2)
[x13 x23]   h21(1 – h21)(y2)(1 – y2) (y2^ - y2)(w1)   h22(1–h22) (y2)(1 – y2) (y2^ - y2)(w2)]
[x14 x24]]

X.T is 4 by 2 and H_D_Error_W is 2 by 2 so the result will be 4 by 2. YAY!! Notice that W1 is 4 by 2 – so this is right (and this is how you know its right).

RESULT for the derivatives that will update W1:

[x11* h11(1 – h11) (y1)(1 – y1)(y1^ - y1)(w1) + **x21*** h21(1 – h21)(y2)(1 – y2) (y2^ - y2)(w1)   **x11*** h12(1 – h12) (y1)(1 – y1) (y1^ - y1)(w2) + **x21*** h22(1–h22) (y2)(1 – y2) (y2^ - y2)(w2)
x12* h11(1 – h11) (y1)(1 – y1)(y1^ - y1)(w1) + **x22*** h21(1 – h21)(y2)(1 – y2) (y2^ - y2)(w1)   **x12*** h12(1 – h12) (y1)(1 – y1) (y1^ - y1)(w2) + **x22*** h22(1–h22) (y2)(1 – y2) (y2^ - y2)(w2)
x13* h11(1 – h11) (y1)(1 – y1)(y1^ - y1)(w1) + **x23*** h21(1 – h21)(y2)(1 – y2) (y2^ - y2)(w1)   **x13*** h12(1 – h12) (y1)(1 – y1) (y1^ - y1)(w2) + **x23*** h22(1–h22) (y2)(1 – y2) (y2^ - y2)(w2)
x14* h11(1 – h11) (y1)(1 – y1)(y1^ - y1)(w1) + **x24*** h21(1 – h21)(y2)(1 – y2) (y2^ - y2)(w1)   **x14*** h12(1 – h12) (y1)(1 – y1) (y1^ - y1)(w2) + **x24*** h22(1–h22) (y2)(1 – y2) (y2^ - y2)(w2)

****Recall – the dTL/dw11 was this SUM. We now have this in the result!**
x11 * (h11)(1 – h11) * w1 * (y^1)(1 – y^1) * (y^1– y1)
+
x21 * (h21)(1 – h21) * w1 * (y^2)(1 – y^2) * (y^2– y2)

You can compare and check each of these with the derivatives we created above for W1.

What did we discover here?

When we have more than one input, we need to sum the derivatives.

NEXT STEPS

1) **Create matrices that can work in Python and that produce the above.**
2) **Complete this process for W2, b1 and b2, and c**