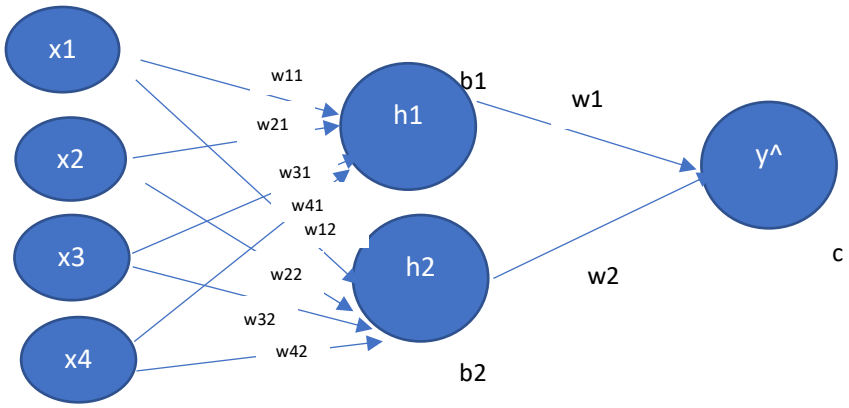
1) Let's use this NN just as an example to look at. We will use differing values of n, but we will stick with 4 columns and 2 hidden units.




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

**X** = [ [x1, x2, x3, x4] ]

X is n by c

**Y** = [ [y ] ]

**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 = [ z1=x1(**w11**) + x2(w21) + x3(w31) + x4(w41) + b1 z2=x1(w12) + x2(w22) + x3(w32) + x4(w42) + b2 ]

H1 = Sig(Z1) = (shape of H1 is n by h)

H1= [ h1=Sig(z1) h2=Sig(z2) ]

H1 shape is n by h

Z2 = [ z2 = h1w1 + h2w2 + c ]

Z2 shape: n by o

Y^ = [Sig( 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 **$1/2(y^ - y)^2$**

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

dL/dw11 = dz1/dw11 * dh1/dz1 * dz2/dh1 * **dy^/dz2* dL/dy^** →
**x1 * (h1)(1 – h1) * w1 * (y^)(1 – y^) * (y^– y)**
dL/dw21 = dz1/dw21 * dh1/dz1 * dz2/dh1 * **dy^/dz2 * dL/dy^** →
**x2 * (h1)(1 – h1) * w1 * (y^)(1 – y^) * (y^ – y)**
dL/dw31 = dzi1/dw31 * dhi1/dzi1 * dz2_i/dhi1 * **dy^i/dz2_i * dLi/dy^i** →
**x3 * (h1)(1 – h1) * w1 * (y^)(1 – y^) * (y^ – y)**
dL/dw41 = dzi1/dw31 * dhi1/dzi1 * dz2_i/dhi1 * **dy^i/dz2_i * dLi/dy^i** →
**x4 * (h1)(1 – h1) * w1 * (y^)(1 – y^) * (y^– y)**

dL/dw12 = dz2/dw12 * dh2/dz2 * dz2/dh2 * **dy^/dz2 * dL/dy^**
**x1 * (h2)(1 – h2) * w2 * (y^)(1 – y^) * (y^ – y)**
dL/dw22 = dz2/dw22 * dh2/dz2 * dz2/dh2 * **dy^/dz2 * dL/dy^**
**x2 * (h2)(1 – h2) * w2 * (y^)(1 – y^) * (y^ – y)**
dL/dw32 = dz2/dw32 * dh2/dz2 * dz2/dh2 * **dy^/dz2 * dL/dy^**
**x3 * (h2)(1 – h2) * w2 * (y^)(1 – y^) * (y^ – y)**
dL/dw42 = dz2/dw42 * dh2/dz2 * dz2/dh2 * **dy^/dz2 * dL/dy^**
**x4 * (h2)(1 – h2) * w2 * (y^)(1 – y^) * (y^ – y)**

W1 = X mult Sig(H) mult W2 mult D_Error

**LOOK AT THE SHAPES and BUILD what you need:**

1) Build **(y^)(1 – y^) * (y^ – y)** this shape will be n by o and is Sig(Y^)*(Error)

Call this **D_Error**

In general, Y^ is n by o
Sig(Y^) must be n by o
Y is n by o
so Y^ - Y must be n by o

2) Now, we need the W2 matrix to multiply by D_Error correctly.

Here, we need w1 times D_Error and we also need w2 times D_Error, where W2 = [ [w1], [w2 ] ] recall W2 is h by o
D_Error is n by o
So D_Error @ W2.T will be

[ (y^)(1 – y^) * (y^ – y) ] @ [ w1 w2 ]

n by o o by h (why o by h? because we transposed it)

→ n by h!

[ (y^)(1 – y^) * (y^ – y) * w1 (y^)(1 – y^) * (y^ – y) * w2 ]

3) Next, we need to multiply by h1(1 – h1) and h2(1 – h2) properly

Directly multiply the derivative of the sig of h by what we built above.
[ h1(1 – h1) h2(1 – h2) ] * [ (y^)(1 – y^) * (y^ – y) * w1 (y^)(1 – y^) * (y^ – y) * w2 ]
We get this….
[ h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1 h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2 ]

4) Now- we have almost all the parts for the derivatives of the W1 (which include w11, w12, w13, w14, w21, w22, w23, w24)
The final step is to multiply the RIGHT X values….
Notice also that we need to end up with 8 weight values here so that we can use them to update the weight matrix. Remember, these are the derivatives that we will use to make updates with. They are part of our GRADIENT.
How can we use
[ h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1 h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2 ]
and
X = [ [1, 2, 3, 4] ] to create what we need?? We are trying to create a matrix that matches the shape of W1 – right?
The shape of W1 is c by h (which in this example is 4 by 2). This gives us a HINT. To create a 4 by 2 matrix, we can multiply a n by 2 @ 4 by n →

[ x1,
x2,
x3, @ [ h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1 h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2 ]
x4 ]]

[ x1 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1 x1 * h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2
x2 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1 x2 * h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2
x3 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1 x3 * h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2
x4 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1 x4 * h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2 ]

YAY!! This is what we need. It is the same shape as W1 and it is exactly the eight W1 derivatives as we have above.
Note, for example, this: x1 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1 is the gradient (the derivative) for w11. It is dL/dw11.

Question – does this work for 2 inputs or 10 inputs or 10,000 inputs? HINT: Yes 😉
Question – If you do this for 3 inputs, how does this affect the gradient for W1?

**Next, we need to get the derivatives for b, c, and W2.**

dL/db1 = dz1/db1 * dh1/dz1 * dz2/dh1 * dy^/dz2 * dL/dy^

(h1)(1 – h1) * **w1** * **(y^i)(1 – y^i) * (y^i – yi)**

dL/db2 = dz2/db2 * dh2/dz1 * dz2/dh2 * **dy^/dz2 * dL/dy^**

(hi2)(1 – hi2) *** w2 * (y^i)(1 – y^i) * (y^i – yi)**

We already have this!

[ h1(1 – h1) h2(1 – h2) ] * ([ (y^)(1 – y^) * (y^ – y) ] @ [ w1 w2 ] ) →

[ h1(1 – h1) (y^)(1 – y^) * (y^ – y) *** w1** h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2 ] ← THIS is the gradient (derivatives) for b1 and b2

This is perfect because the shape of the b biases is [ b1 b2 ] and this matches our shape so that we can use it to update the b's.

---

Finally, we need the derivatives for W2 and for c

dL/dw1 = dz2/dw1 * **dy^/dz2 * dL/dy^**

**h1 *(y^)(1 – y^) * (y^ – y)**

dL/dw2 = dz2_i/dw2 * **dy^i/dz2_i * dLi/dy^i**

**h2 *(y^)(1 – y^) * (y^ – y)**

This is simply H.T @ D_Error →

[ h1 @ [ (y^)(1 – y^)(y^ – y) ]

h2]

=

[ h1(y^)(1 – y^)(y^ – y)

h2(y^)(1 – y^)(y^ – y) ]

Lastly:

dL/dc = dz2/dc * **dy^/dz2 * dL/dy^**

(1) ***(y^i)(1 – y^i) * (y^i – yi)**

→ [ (y^)(1 – y^)(y^ – y) ]

**This is our final gradient when we use one input vector……**

[ dw11 = x1 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1
dw12 = x1 * h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2
dw21 = x2 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1
dw22 = x2 * h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2
dw31 = x3 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1
dw32 = x3 * h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2
dw41 = x4 * h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1
dw42 = x4 * h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2

db1 = h1(1 – h1) (y^)(1 – y^) * (y^ – y) * w1
db2 = h2(1 – h2) (y^)(1 – y^) * (y^ – y) * w2

dw1 = h1(y^)(1 – y^) * (y^ – y)
dw2 = h2(y^)(1 – y^) * (y^ – y)

dc = (y^)(1 – y^) * (y^ – y)
]