Why is $\frac{\partial J(W)}{\partial W_{i,j}^{(out)}} = \left(A^{(h)}\right)^T \delta^{(out)}$?

Kassi Bertrand

January 19^{th}, 2023

1 Introduction

In this LaTeX document, I want to show my step-by-step process to derive the partial derivative of the loss function with respect to of all the weights in $W^{(out)}$.

I'll start simple with a 2-2-2 MLP, which is a Multi-Layer Perceptron with 2 inputs, 2 hidden units, and 2 outputs, and then generalize what we learn to a general $m - d - t$ MLP.

In both cases, I will use the following loss/error function:

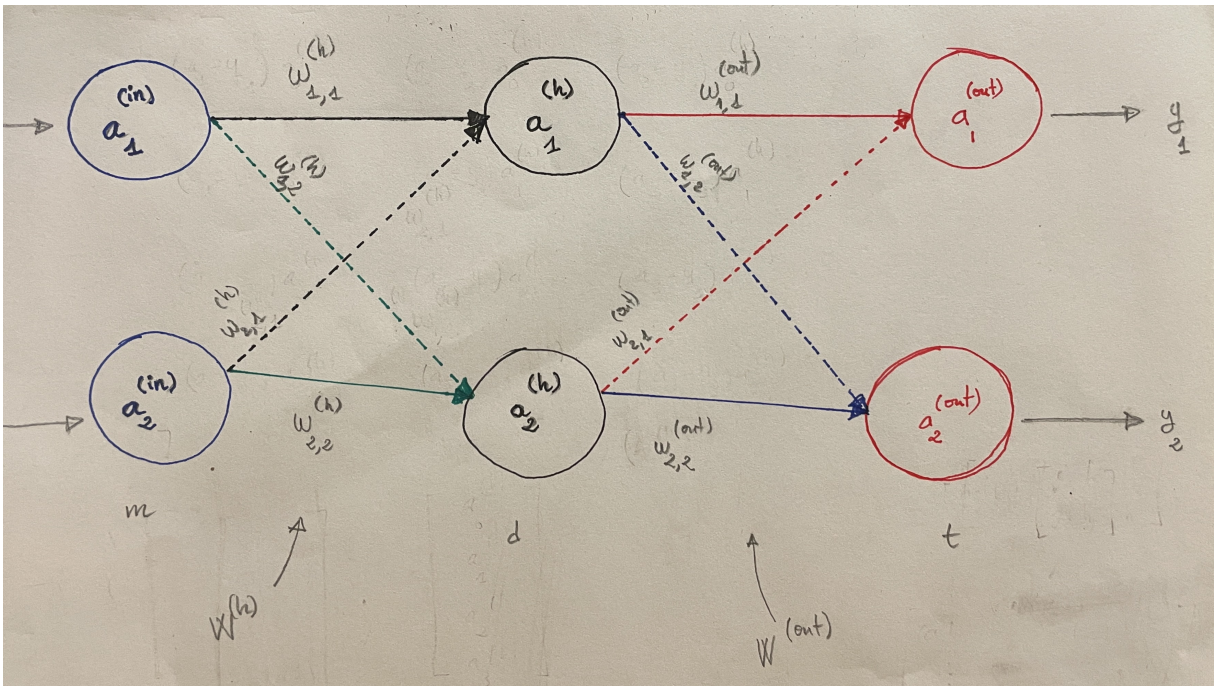
$$J(w) = -\sum_{i=1}^{n} \sum_{j=1}^{t} y_j^{[i]} ln(a_j^{[i]}) + (1 - y_j^{[i]}) ln(1 - a_j^{[i]})$$

Where the superscript $[i]$ is an index for training examples, and j is the number of output units.

Ready? Let's go!

2 With a 2-2-2 MLP

For this example, we will consider the following Neural network:



We'll ignore biase units in the input and hidden layers, and consider only ONE training example for simplicity purposes.

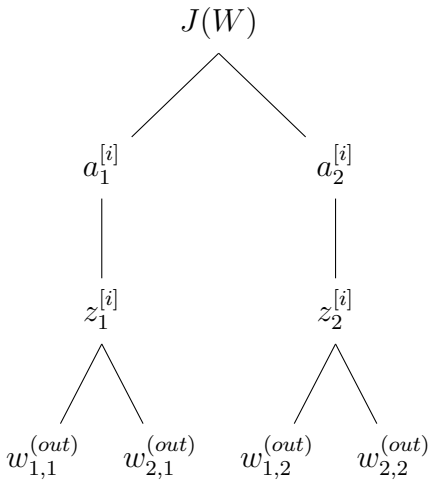
Since one training example is considered and the network has two activation units in its output layer, then $n = 1$ and $t = 2$. So, the loss function becomes like this:

$$J(w) = -\sum_{i=1}^{1}\sum_{j=1}^{2} y_j^{[i]} ln(a_j^{[i]}) + (1 - y_j^{[i]}) ln(1 - a_j^{[i]})$$

The journey of a SINGLE training example from the input layer to the output layer of our network goes like this:

$$[a_1^{(in)}, a_2^{(in)}]$$
$$\downarrow$$
$$W^{(h)}$$
$$\downarrow$$
$$[z_1^{(h)}, z_2^{(h)}]$$
$$\downarrow$$
$$\phi(\bullet)$$
$$\downarrow$$
$$[a_1^{(h)}, a_2^{(h)}]$$
$$\downarrow$$
$$W^{(out)}$$
$$\downarrow$$
$$[z_1^{(out)}, z_2^{(out)}]$$
$$\downarrow$$
$$\phi(\bullet)$$
$$\downarrow$$
$$[a_1^{(out)}, a_2^{(out)}]$$
$$\downarrow$$
$$[y_1, y_2]$$

Now, let's compute the partial derivative of $J(W)$ w.r.t to each weights in the $W^{(out)}$ matrix. To help myself, I drew the following tree diagram to see how the variables in $J(W)$ relate to the weights in $W^{(out)}$:



Our MLP has two outputs, and if you look at $J(W)$ closely, you'll see that it's a **sum**. So, we compute the cost for each output, then add the results together. And that's how we get the cost for the current training example. With that in mind, we can now compute the partial derivatives:

$$\frac{\partial J(W)}{\partial w_{1,1}^{(out)}} = \frac{\partial J(W)}{\partial a_1^{[i]}} \times \frac{\partial a_1^{[i]}}{\partial z_1^{[i]}} \times \frac{\partial z_1^{[i]}}{\partial w_{1,1}^{(out)}}$$

$$\frac{\partial J(W)}{\partial w_{2,1}^{(out)}} = \frac{\partial J(W)}{\partial a_1^{[i]}} \times \frac{\partial a_1^{[i]}}{\partial z_1^{[i]}} \times \frac{\partial z_1^{[i]}}{\partial w_{2,1}^{(out)}}$$

$$\frac{\partial J(W)}{\partial w_{1,2}^{(out)}} = \frac{\partial J(W)}{\partial a_2^{[i]}} \times \frac{\partial a_2^{[i]}}{\partial z_2^{[i]}} \times \frac{\partial z_2^{[i]}}{\partial w_{1,2}^{(out)}}$$

$$\frac{\partial J(W)}{\partial w_{2,2}^{(out)}} = \frac{\partial J(W)}{\partial a_2^{[i]}} \times \frac{\partial a_2^{[i]}}{\partial z_2^{[i]}} \times \frac{\partial z_2^{[i]}}{\partial w_{2,2}^{(out)}}$$

Notice, the following expressions are common in the first two and last two equations. Let's evaluate them:

$$\frac{\partial J(W)}{\partial a_1^{[i]}} \times \frac{\partial a_1^{[i]}}{\partial z_1^{[i]}} = (a_1^{[i]} - y_1) = \delta_1^{(out)}$$

$$\frac{\partial J(W)}{\partial a_2^{[i]}} \times \frac{\partial a_2^{[i]}}{\partial z_2^{[i]}} = (a_2^{[i]} - y_2) = \delta_2^{(out)}$$

$(a_1^{[i]} - y_1)$ and $(a_2^{[i]} - y_2)$ are the "**error**" terms in the output layer. Since our MLP has two outputs, we have two "error" terms as well. We use "δ" to denote that "error". $\delta_1^{(out)}$ for instance, is the error in the first activation unit in the output layer.

With this new knowledge, we can re-write the partial derivatives:

$$\frac{\partial J(W)}{\partial w_{1,1}^{(out)}} = \delta_1^{(out)} a_1^{(h)}$$

$$\frac{\partial J(W)}{\partial w_{2,1}^{(out)}} = \delta_1^{(out)} a_2^{(h)}$$

$$\frac{\partial J(W)}{\partial w_{1,2}^{(out)}} = \delta_2^{(out)} a_1^{(h)}$$

$$\frac{\partial J(W)}{\partial w_{2,2}^{(out)}} = \delta_2^{(out)} a_2^{(h)}$$

From the above, I can now introduce the $\delta^{(out)}$ matrix. It is our "error" matrix, and is of the shape $n \times t$ where n is the number of training examples and t the number of activation units in the output layer. Since we are dealing with ONE example and our MLP has 2 outputs, so the $\delta^{(out)}$ matrix is of the shape (1×2), and looks like this:

$$\delta^{(out)} = [\delta_1^{(out)}, \delta_2^{(out)}] = [(a_1^{[i]} - y_1), (a_2^{[i]} - y_2)]$$

Also, remember the $A^{(h)}$ matrix, obtained after training examples are forward propagated from the input to the hidden layer. It is a $(n \times d)$

matrix; where n is the number of training examples and d the number of activation units in the hidden layer. Since we have ONE training example and 2 units in the hidden layer, our $A^{(h)}$ matrix is of the shape (1×2), and looks like this:

$$A^{(h)} = [a_1^{[1]}, a_2^{[1]}]$$

We computed the partial derivatives with respect to the weights in $W^{(out)}$ above. But we can implement the same operations in vectorized form? Yes, of course! Look at this:

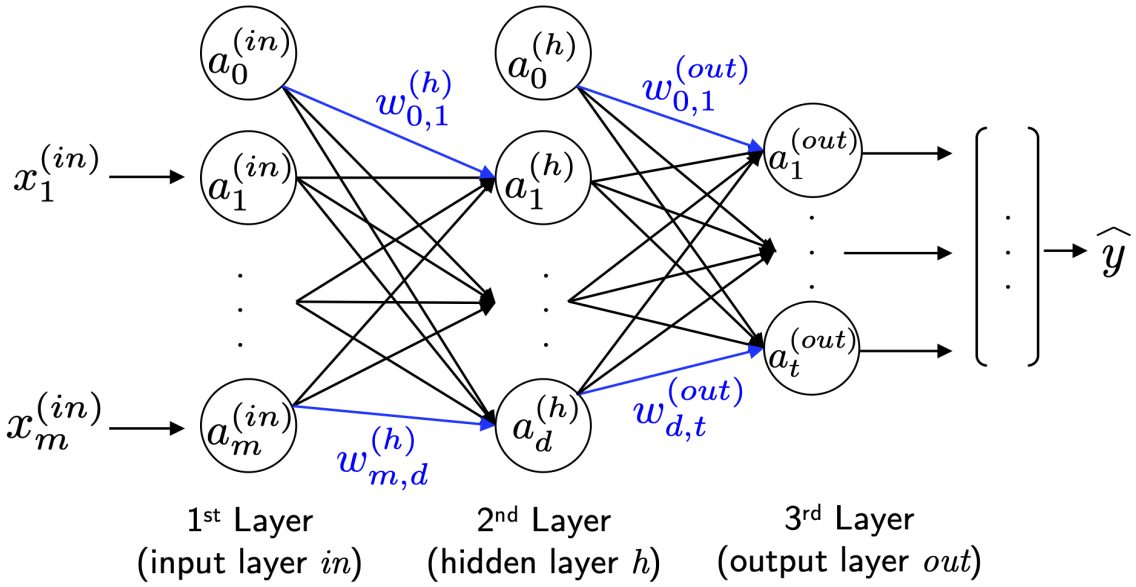
$$(A^{(h)})^T \delta^{(out)} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} \times \begin{bmatrix} \delta_1^{(out)} & \delta_2^{(out)} \end{bmatrix} = \begin{bmatrix} a_1^{(h)}\delta_1^{(out)} & a_2^{(h)}\delta_1^{(out)} \\ a_1^{(h)}\delta_2^{(out)} & a_2^{(h)}\delta_2^{(out)} \end{bmatrix}$$

See? By transposing the $A^{(h)}$ and multiplying it with $\delta^{(out)}$, we obtain a new matrix containing the partial derivatives of $J(W)$ w.r.t to each weights in $W^{(out)}$.

We can see that for our 2-2-2 MLP, $\frac{\partial J(W)}{\partial W_{i,j}^{(out)}} = (A^{(h)})^T \delta^{(out)}$. Can we show this relationship holds for a general MLP?

3 With a general $m - d - t$ MLP

Let's consider a general MLP like the following:



Here again, we will consider ONE training example. However, the network has "t" activation units in its output layer. So, the loss function becomes like this:

$$J(w) = -\sum_{i=1}^{1}\sum_{j=1}^{t} y_j^{[i]} ln(a_j^{[i]}) + (1 - y_j^{[i]}) ln(1 - a_j^{[i]})$$

Here is what the journey of SINGLE of a training example from the input layer to the output layer looks like this:

$$[a_1^{(in)}, a_2^{(in)}, \cdots, a_m^{(in)}]$$

$$\downarrow$$

$$W^{(h)}$$

$$\downarrow$$

$$[z_1^{(h)}, z_2^{(h)}, \cdots, z_d^{(h)}]$$

$$\downarrow$$

$$\phi(\bullet)$$

$$\downarrow$$

$$[a_1^{(h)}, a_2^{(h)}, \cdots, a_d^{(h)}]$$

$$\downarrow$$

$$W^{(out)}$$

$$\downarrow$$

$$[z_1^{(out)}, z_2^{(out)}, \cdots, z_t^{(out)}]$$

$$\downarrow$$

$$\phi(\bullet)$$

$$\downarrow$$

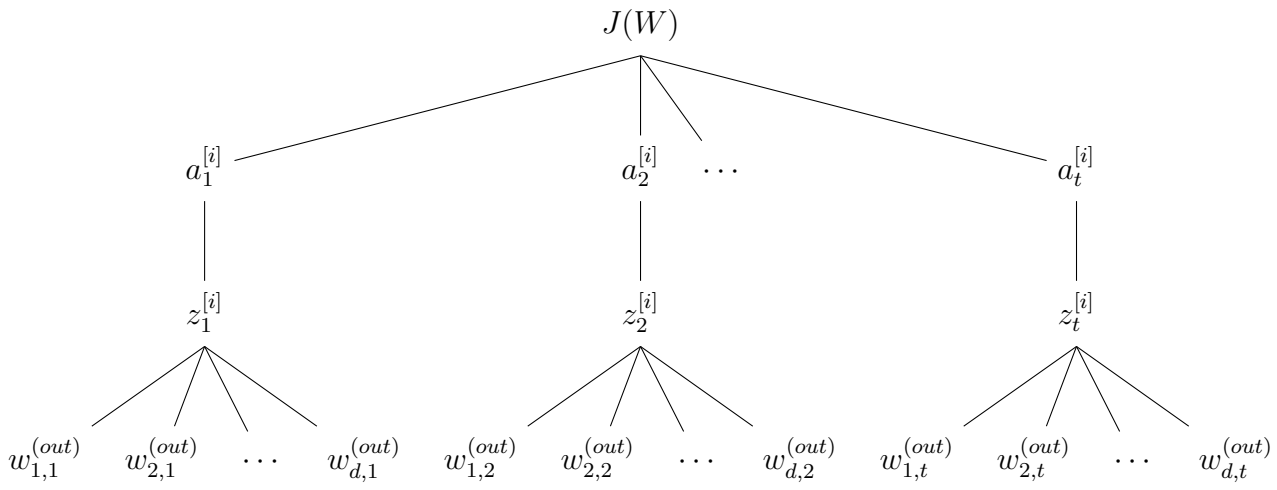
$$[a_1^{(out)}, a_2^{(out)}, \cdots, a_t^{(out)}]$$

$$\downarrow$$

$$[y_1, y_2, \cdots, y_t]$$

Notice, the biases have been ignored in the preceding figure.

Let's now draw a tree diagram once again to see how the variables in $J(W)$ in are related to the weights in $W^{(out)}$. As you probaby think already, it's pretty similar to what we had for the 2-2-2 MLP.



4 Challenge

Can you repeat everything we did, but considering n training examples?