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

When reading Chapter 12: Implementing a Multilayer Artificial Neural Perceptron from Scractch, Dr. Raschka used the following formula to compute the partial derivative of the loss function with respect to the all the weight in $W^{(out)}$ when backpropagating:

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

I did not understand it at first, but after many trials I now do. In this document, I want to walk you through how I computed the partial derivative of the loss function with respect to all the weights in $W^{(out)}$, and got to the result Sebastian presented in his book.

In chapter 12, Dr. Raschka used the following as the cost function:

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

In $J(W)$:

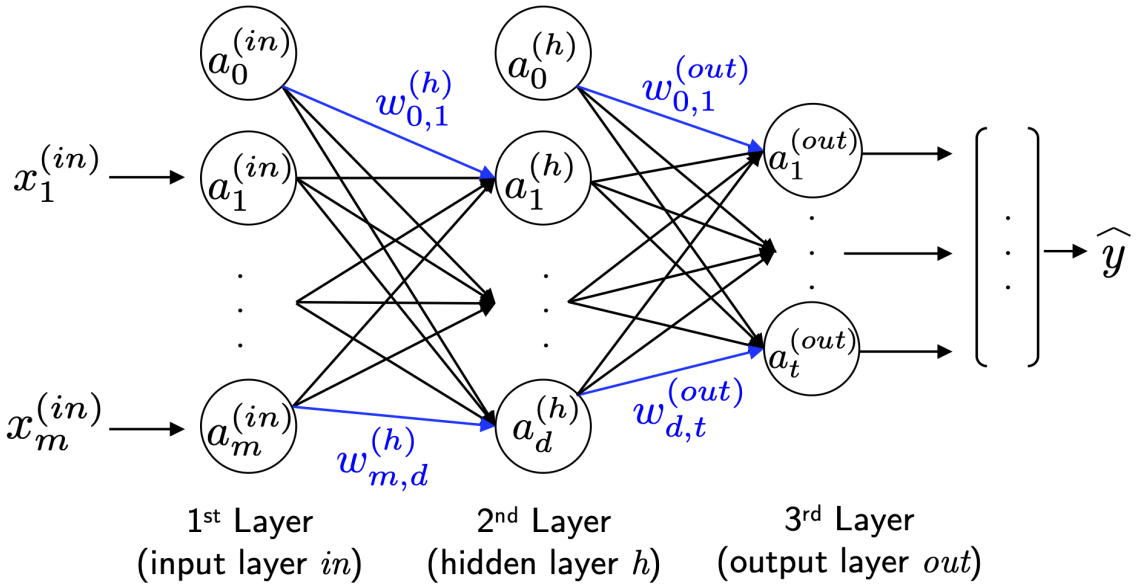
- n is the number of training examples.
- Superscript $[i]$ refers to a specific training example.
- t is the number of activations in the last (outptut) layer.
- Subscript j refers to a specific output neuron.
- $J(W)$ is the 'loss' (or 'error') value. It tells how well the network is performing. Given a training example, the function uses the activation values in the output layer and returns the error.

$a_j^{[i]}$ refers to *the activation value of the j^{th} neuron in the output layer, after the i^{th} example is forward propagated through the network.*

$y_j^{[i]}$ on the other hand, is *the target value for the neuron j in the output layer, given a training example i.* In other words, during training ... we may forward propagate an example i through the network and obtain different activations values in the output layer; we want activation $a_j^{[i]}$ to get as close as possible to $y_j^{[i]}$.

2 A general $m - d - t$ MLP

Let's consider a general MLP consisting of m neurons in the input, d neurons in the hidden, and t neurons in the output layers:



Also, let's imagine ONE training example goes through the network. This is to what its journey looks like from the input layer to the output layer:

$$[a_0^{(in)}, a_1^{(in)}, a_2^{(in)}, \cdots, a_m^{(in)}]$$
$$\downarrow$$
$$W^{(h)}$$
$$\downarrow$$
$$[z_1^{(h)}, z_2^{(h)}, \cdots, z_d^{(h)}]$$
$$\downarrow$$
$$\phi(\bullet)$$
$$\downarrow$$
$$[a_0^{(h)}, a_1^{(h)}, a_2^{(h)}, \cdots, a_d^{(h)}]$$
$$\downarrow$$
$$W^{(out)}$$
$$\downarrow$$
$$[z_1^{(out)}, z_2^{(out)}, \cdots, z_t^{(out)}]$$
$$\downarrow$$
$$\phi(\bullet)$$
$$\downarrow$$
$$[a_1^{(out)}, a_2^{(out)}, \cdots, a_t^{(out)}]$$

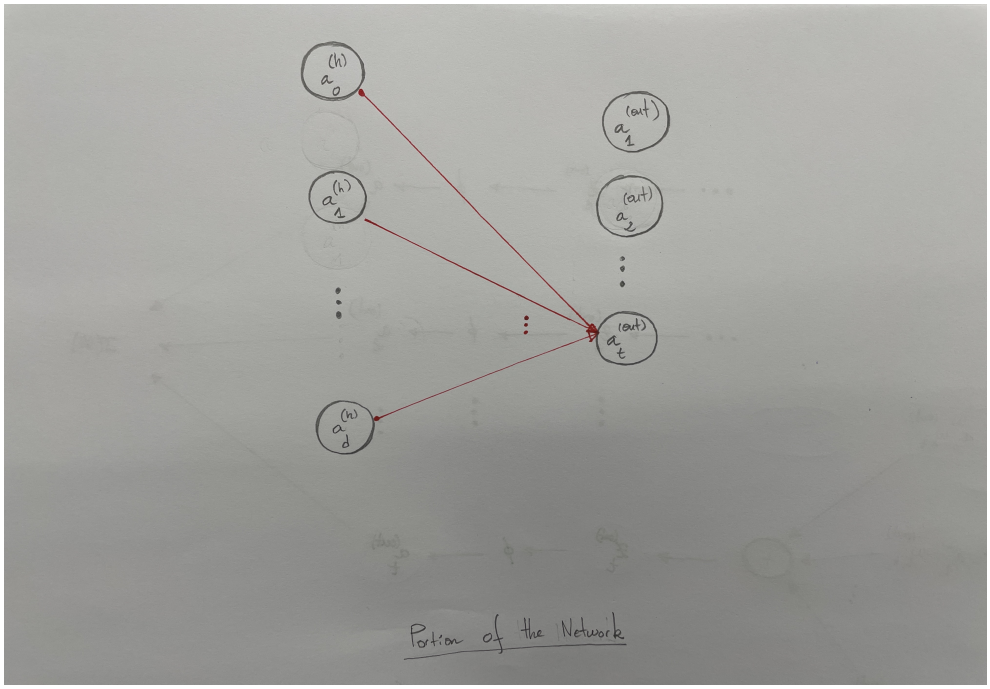
Note: I represent training examples and weight vectors as row vectors

With that out of the way, let's come back to our original purpose: Deriving the formula Dr. Raschka used. This document is concerned with $W^{(out)}$ (the weights between the hidden and output layer). In the following three sections, I will manually derive the gradients.

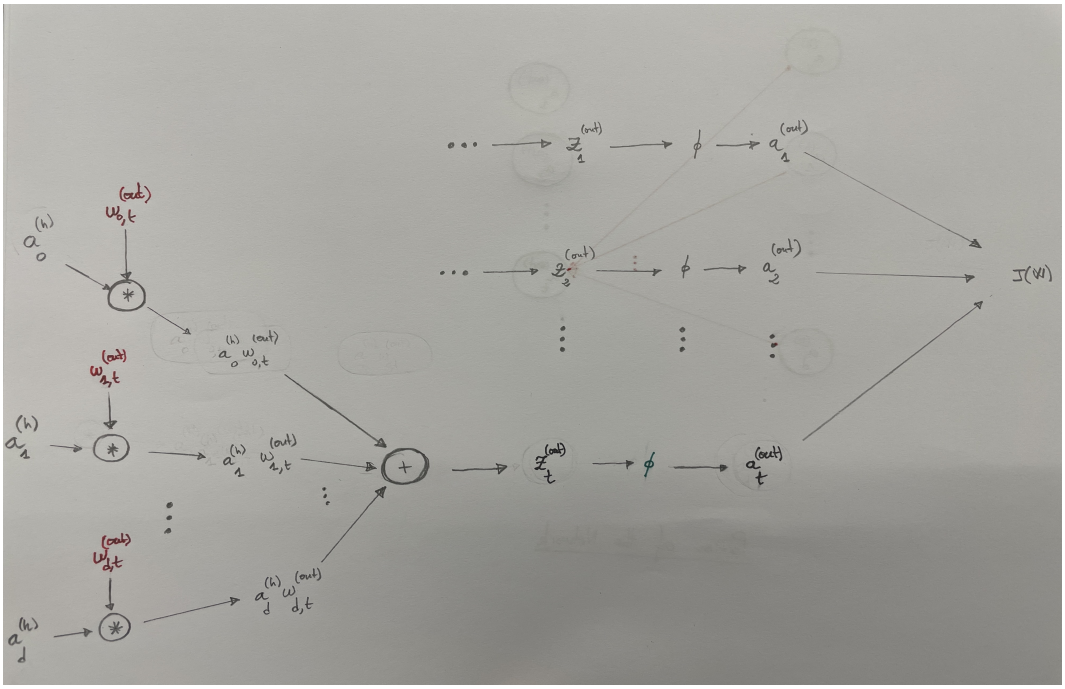
Let's first consider ONE training example, then expand to n examples.

3 Gradients of $w_{0,t}^{(out)}$, $w_{1,t}^{(out)}$, ..., $w_{d,t}^{(out)}$

Consider this portion of $W^{(out)}$:



The connections $w_{0,t}^{(out)}$, $w_{1,t}^{(out)}$, ..., $w_{d,t}^{(out)}$ are highlighted in red. Let's look at them in detail...



On the image above, you are looking at graph representation of the portion we are focusing on. You can see how the weights $w_{0,t}^{(out)}$, $w_{1,t}^{(out)}$, ... $w_{d,t}^{(out)}$ get multiplied with the activations of the hidden layer and how subsequent operations result in activation value $a_t^{(out)}$ activation unit in the output layer of our MLP.

To get to the formula Dr. Raschka used, we must **backpropagate**. In other words, we must compute the gradient of $J(W)$ w.r.t *all* the weights in $W^{(out)}$.

Since we focused our attention on $w_{0,t}^{(out)}$, $w_{1,t}^{(out)}$, ... $w_{d,t}^{(out)}$. Let's manually compute their gradients!

We start at $J(W)$. When we take the partial derivative of the loss w.r.t itself. We get 1

$$\frac{\partial J(W)}{\partial J(W)} = 1$$

I continue with $a_t^{(out)}$

$$\frac{\partial J(W)}{\partial a_t^{(out)}} = \frac{a_t^{(out)} - y_t}{a_t^{(out)}(1 - a_t^{(out)})}$$

In case you wonder how I came up with this result, I wrote another document about it [LINK IT]

I move on with $z_t^{(out)}$

$$\frac{\partial J(W)}{\partial z_t^{(out)}} = \frac{\partial a_t^{(out)}}{\partial z_t^{(out)}} \times \frac{\partial J(W)}{\partial a_t^{(out)}} = a_t^{(out)}(1 - a_t^{(out)}) \bullet \frac{\partial J(W)}{\partial a_t^{(out)}} = a_t^{(out)} - y_t$$

Let's continue with $a_0^{(h)} w_{0,t}^{(out)}$, $a_1^{(h)} w_{1,t}^{(out)}$, ..., $a_d^{(h)} w_{d,t}^{(out)}$

$$\frac{\partial J(W)}{\partial a_0^{(h)} w_{0,t}^{(out)}} = \frac{\partial J(W)}{\partial a_1^{(h)} w_{1,t}^{(out)}} = \cdots = \frac{\partial J(W)}{\partial a_d^{(h)} w_{d,t}^{(out)}} = a_t^{(out)} - y_t$$

Conveniently, they all have the same gradients. I encourage you to compute the chain rule and see why.

We can now determine the gradient of $w_{0,t}^{(out)}$, $w_{1,t}^{(out)}$, ... $w_{d,t}^{(out)}$. I will ignore the bias weight $w_{0,t}^{(out)}$, but will come to later in the document.

$$\frac{\partial J(W)}{\partial w_{1,t}^{(out)}} = \frac{\partial a_1^{(h)} w_{1,t}^{(out)}}{\partial w_{1,t}^{(out)}} \times \frac{\partial J(W)}{\partial a_1^{(h)} w_{1,t}^{(out)}} = a_1^{(h)}(a_t^{(out)} - y_t)$$

$$\vdots$$

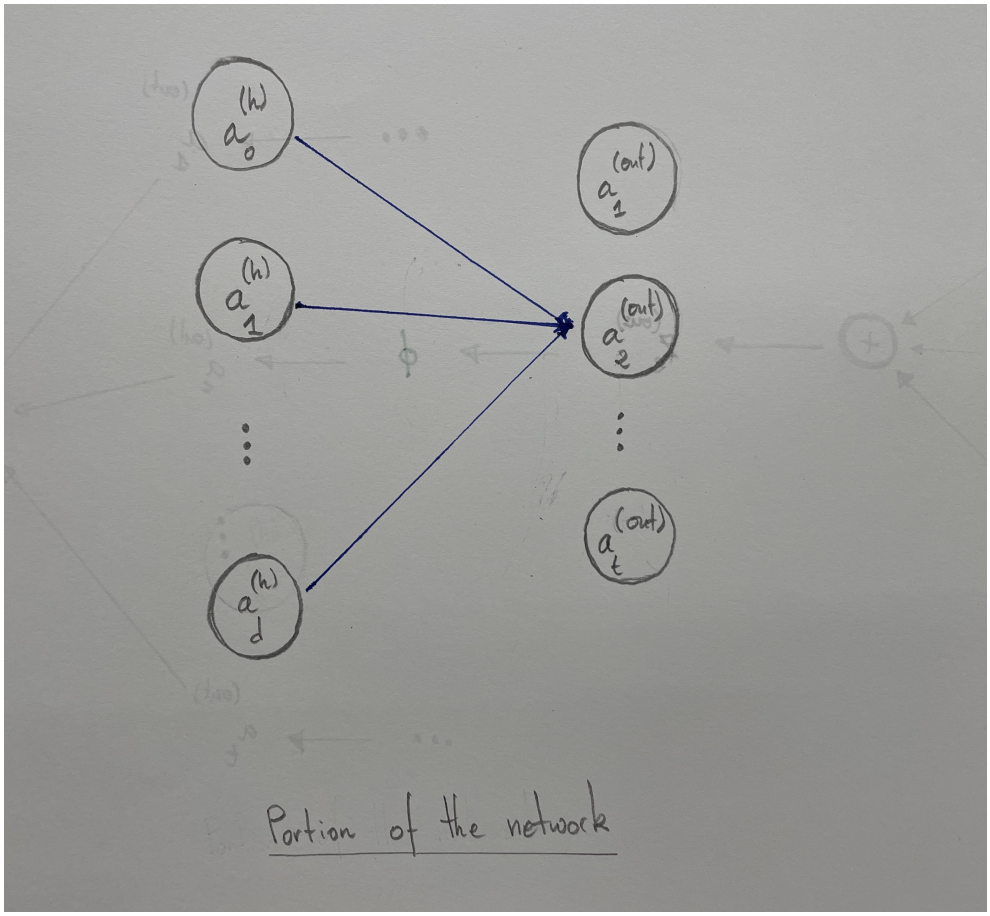
$$\frac{\partial J(W)}{\partial w_{d,t}^{(out)}} = \frac{\partial a_d^{(h)} w_{d,t}^{(out)}}{\partial w_{d,t}^{(out)}} \times \frac{\partial J(W)}{\partial a_d^{(h)} w_{d,t}^{(out)}} = a_d^{(h)}(a_t^{(out)} - y_t)$$

Notice, how $a_t^{(out)} - y_t$ repeats in the results we found. Let's call this expression $\delta_t^{(out)}$.

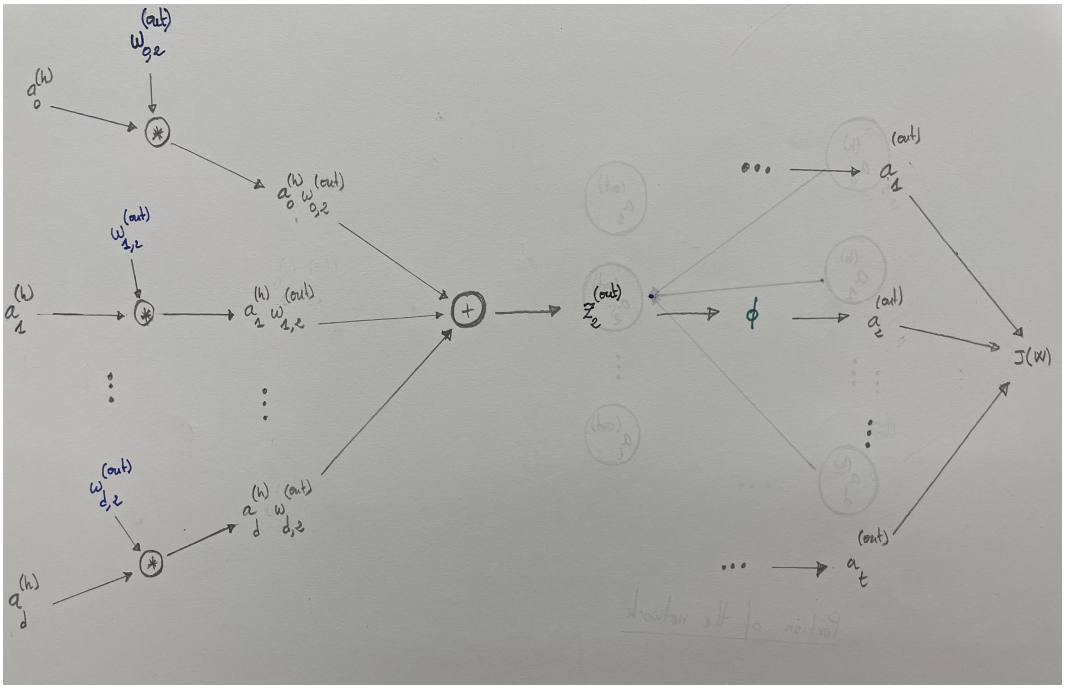
We now know what the gradients of $w_{0,t}^{(out)}$, $w_{1,t}^{(out)}$, ... $w_{d,t}^{(out)}$ are. Let's move our attention to another part of the network.

4 Gradients of $w_{0,2}^{(out)}$, $w_{1,2}^{(out)}$, ..., $w_{d,2}^{(out)}$

Consider this other portion of $W^{(out)}$:



The connections $w_{0,2}^{(out)}$, $w_{1,2}^{(out)}$, ..., $w_{d,2}^{(out)}$ are highlighted in blue. Again, let's look at them in detail... just like we did in the previous section.



Let's compute the gradients, starting with the loss value $J(W)$. As you read, notice how similar the result we are about to find are to the ones we previously found.

$$\frac{\partial J(W)}{\partial J(W)} = 1$$

I continue with $a_2^{(out)}$

$$\frac{\partial J(W)}{\partial a_2^{(out)}} = \frac{a_2^{(out)} - y_2}{a_2^{(out)}(1 - a_2^{(out)})}$$

Moving on with $z_2^{(out)}$

$$\frac{\partial J(W)}{\partial z_2^{(out)}} = \frac{\partial a_2^{(out)}}{\partial z_2^{(out)}} \times \frac{\partial J(W)}{\partial a_2^{(out)}} = a_2^{(out)}(1 - a_2^{(out)}) \bullet \frac{\partial J(W)}{\partial a_2^{(out)}} = a_2^{(out)} - y_2$$

Let's continue with: $a_0^{(h)} w_{0,2}^{(out)}$, $a_1^{(h)} w_{1,2}^{(out)}$, ..., $a_d^{(h)} w_{d,2}^{(out)}$

$$\frac{\partial J(W)}{\partial a_0^{(h)} w_{0,2}^{(out)}} = \frac{\partial J(W)}{\partial a_1^{(h)} w_{1,2}^{(out)}} = \cdots = \frac{\partial J(W)}{\partial a_d^{(h)} w_{d,2}^{(out)}} = a_2^{(out)} - y_2$$

Conveniently, they also have the same gradients.

We can now determine the gradients of $w_{0,2}^{(out)}$, $w_{1,2}^{(out)}$, ..., $w_{d,2}^{(out)}$. Here again, I will ignore the bias weight $w_{0,2}^{(out)}$, but will come to it later. I promise.

$$\frac{\partial J(W)}{\partial w_{1,2}^{(out)}} = \frac{\partial a_1^{(h)} w_{1,2}^{(out)}}{\partial w_{1,2}^{(out)}} \times \frac{\partial J(W)}{\partial a_1^{(h)} w_{1,2}^{(out)}} = a_1^{(h)} (a_2^{(out)} - y_2)$$

$$\vdots$$

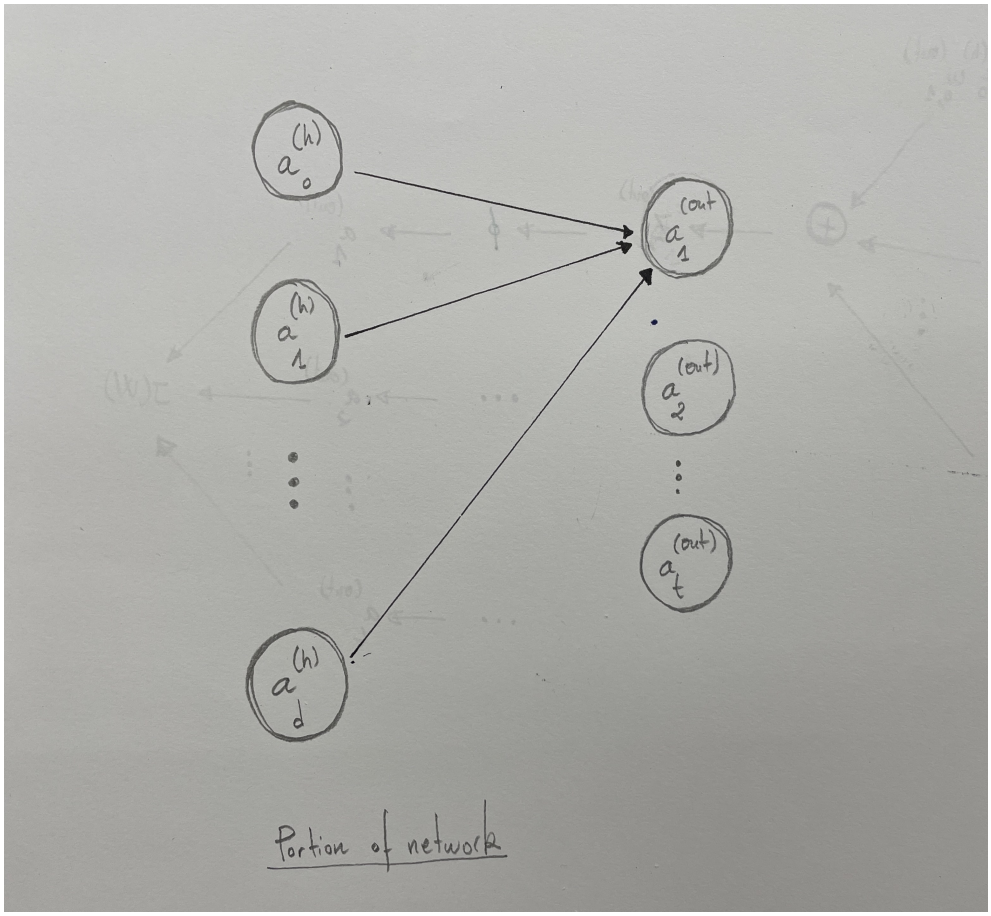
$$\frac{\partial J(W)}{\partial w_{d,2}^{(out)}} = \frac{\partial a_d^{(h)} w_{d,2}^{(out)}}{\partial w_{d,2}^{(out)}} \times \frac{\partial J(W)}{\partial a_d^{(h)} w_{d,2}^{(out)}} = a_d^{(h)} (a_2^{(out)} - y_2)$$

Please notice once again, how $a_2^{(out)} - y_2$ repeats in the results we found. Let's call it $\delta_2^{(out)}$.

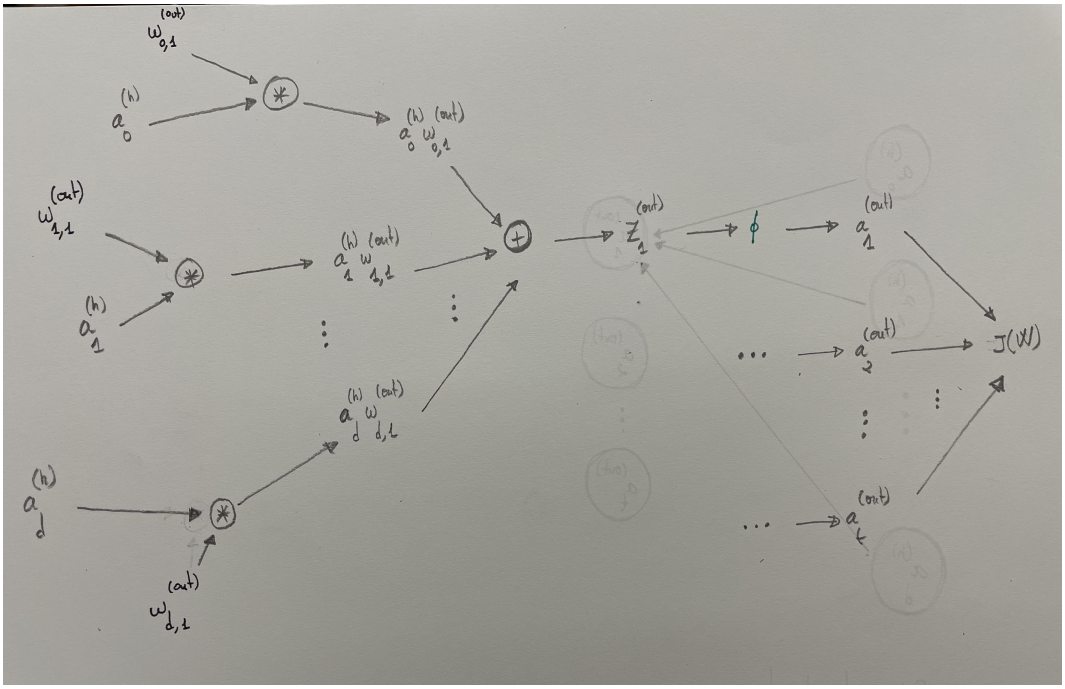
We now know the gradients of $w_{0,2}^{(out)}$, $w_{1,2}^{(out)}$, ..., $w_{d,2}^{(out)}$. Let's move our attention to another part of the network.

5 Gradients of $w_{0,1}^{(out)}$, $w_{1,1}^{(out)}$, ..., $w_{d,1}^{(out)}$

Let's now consider this porttion of $W^{(out)}$. It's the last one.



The connections $w_{0,1}^{(out)}$, $w_{1,1}^{(out)}$, ..., $w_{d,1}^{(out)}$ are highlighted in black this time. Again, let's look at them in detail:



Just like the previous times, let's compute the gradients one at time until we find the gradients of the weights we are interested in. Let's start with the loss value $J(W)$.

$$\frac{\partial J(W)}{\partial J(W)} = 1$$

I continue with $a_1^{(out)}$

$$\frac{\partial J(W)}{\partial a_1^{(out)}} = \frac{a_1^{(out)} - y_1}{a_1^{(out)}(1 - a_1^{(out)})}$$

Moving on with $z_1^{(out)}$

$$\frac{\partial J(W)}{\partial z_1^{(out)}} = \frac{\partial a_1^{(out)}}{\partial z_1^{(out)}} \times \frac{\partial J(W)}{\partial a_1^{(out)}} = a_1^{(out)}(1 - a_1^{(out)}) \bullet \frac{\partial J(W)}{\partial a_1^{(out)}} = a_1^{(out)} - y_1$$

Let's continue with: $a_0^{(h)} w_{0,1}^{(out)}$, $a_1^{(h)} w_{1,1}^{(out)}$, ..., $a_d^{(h)} w_{d,1}^{(out)}$

$$\frac{\partial J(W)}{\partial a_0^{(h)} w_{0,1}^{(out)}} = \frac{\partial J(W)}{\partial a_1^{(h)} w_{1,1}^{(out)}} = \cdots = \frac{\partial J(W)}{\partial a_d^{(h)} w_{d,1}^{(out)}} = a_1^{(out)} - y_1$$

The same gradients. Again.

Let's now determine the gradients of $w_{0,1}^{(out)}$, $w_{1,1}^{(out)}$, ..., $w_{d,1}^{(out)}$. Here again, I will ignore the bias weight $w_{0,1}^{(out)}$, but will come back to it later.

$$\frac{\partial J(W)}{\partial w_{1,1}^{(out)}} = \frac{\partial a_1^{(h)} w_{1,1}^{(out)}}{\partial w_{1,1}^{(out)}} \times \frac{\partial J(W)}{\partial a_1^{(h)} w_{1,1}^{(out)}} = a_1^{(h)}(a_1^{(out)} - y_1)$$

$$\vdots$$

$$\frac{\partial J(W)}{\partial w_{d,1}^{(out)}} = \frac{\partial a_d^{(h)} w_{d,1}^{(out)}}{\partial w_{d,1}^{(out)}} \times \frac{\partial J(W)}{\partial a_d^{(h)} w_{d,1}^{(out)}} = a_d^{(h)}(a_1^{(out)} - y_1)$$

Notice here again, how $a_1^{(out)} - y_1$ repeats in the results we found. Let's call it $\delta_1^{(out)}$.

Using the chain rule, we computed the gradients of $w_{0,1}^{(out)}$, $w_{1,1}^{(out)}$, ..., $w_{d,1}^{(out)}$ starting with $J(W)$.

6 Let's recap

We considered ONE training example. When *this* example is forward propagated through the network $a_1^{(out)}$, $a_2^{(out)}$, ..., $a_t^{(out)}$ get computed.

To backpropagate the error for this example from the output to the hidden layer, we must know the gradients of the weights in the $W^{(out)}$ matrix.

The gradients of weights in $W^{(out)}$ are given by the following:

$$\begin{array}{lll}
\frac{\partial J(W)}{\partial w_{1,1}^{(out)}} = a_1^{(h)}\delta_1^{(out)} & \frac{\partial J(W)}{\partial w_{1,2}^{(out)}} = a_1^{(h)}\delta_2^{(out)} & \frac{\partial J(W)}{\partial w_{1,t}^{(out)}} = a_1^{(h)}\delta_t^{(out)} \\
\frac{\partial J(W)}{\partial w_{2,1}^{(out)}} = a_2^{(h)}\delta_1^{(out)} & \frac{\partial J(W)}{\partial w_{2,2}^{(out)}} = a_2^{(h)}\delta_2^{(out)} & \frac{\partial J(W)}{\partial w_{2,t}^{(out)}} = a_2^{(h)}\delta_t^{(out)} \\
\vdots & \vdots & \vdots \\
\frac{\partial J(W)}{\partial w_{d,1}^{(out)}} = a_d^{(h)}\delta_1^{(out)} & \frac{\partial J(W)}{\partial w_{d,2}^{(out)}} = a_d^{(h)}\delta_2^{(out)} & \frac{\partial J(W)}{\partial w_{d,t}^{(out)}} = a_d^{(h)}\delta_t^{(out)}
\end{array}$$

Conveniently enough, these gradients can be put in a matrix. Also known as **Jacobian matrix**. It gives the following:

$$\begin{bmatrix}
\frac{\partial J(W)}{\partial w_{1,1}^{(out)}} & \frac{\partial J(W)}{\partial w_{1,2}^{(out)}} & \cdots & \frac{\partial J(W)}{\partial w_{1,t}^{(out)}} \\
\frac{\partial J(W)}{\partial w_{2,1}^{(out)}} & \frac{\partial J(W)}{\partial w_{2,2}^{(out)}} & \cdots & \frac{\partial J(W)}{\partial w_{1,t}^{(out)}} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial J(W)}{\partial w_{d,1}^{(out)}} & \frac{\partial J(W)}{\partial w_{d,2}^{(out)}} & \cdots & \frac{\partial J(W)}{\partial w_{d,t}^{(out)}}
\end{bmatrix} = \begin{bmatrix}
a_1^{(h)}\delta_1^{(out)} & a_1^{(h)}\delta_2^{(out)} & \cdots & a_1^{(h)}\delta_t^{(out)} \\
a_2^{(h)}\delta_1^{(out)} & a_2^{(h)}\delta_2^{(out)} & \cdots & a_2^{(h)}\delta_t^{(out)} \\
\vdots & \vdots & \ddots & \vdots \\
a_d^{(h)}\delta_1^{(out)} & a_d^{(h)}\delta_2^{(out)} & \cdots & a_d^{(h)}\delta_t^{(out)}
\end{bmatrix}$$

The above matrix is obtained after multiplying the following two vectors:

$$\begin{bmatrix} a_1^{(h)} \\ a_2^{(h)} \\ \vdots \\ a_d^{(h)} \end{bmatrix} \times \begin{bmatrix} \delta_1^{(out)} & \delta_2^{(out)} & \cdots & \delta_t^{(out)} \end{bmatrix}$$

If you followed closely, you'd notice that the above vectors are 1) the activation values in the hidden layer after the training example we considered goes through and 2) the δ values we determined along the way. We can call this vector $\delta^{(out)}$. So the above can be written as:

$$(a^{(h)})^T \delta^{(out)}$$

7 Okay, but what if we have n examples?

The Jacobian matrix we wrote earlier contains the gradients of all the weights in $W^{(out)}$ for *only one* example. If a second training example goes through the network, a new loss value is computed... and a new gradient is computed for *each* weight in $W^{(out)}$ during backpropagation. The new gradient of *each* weight is **added** to the gradients of the same weight when backpropagation happened for the previous example. It's like they accummulate.

It looks like this matrix form, when n training example go through the network:

$$\begin{bmatrix} \sum_{i=1}^{n} a_1^{[i]}\delta_1^{[i]} & \sum_{i=1}^{n} a_1^{[i]}\delta_2^{[i]} & \dots & \sum_{i=1}^{n} a_1^{[i]}\delta_t^{[i]} \\ \sum_{i=1}^{n} a_2^{[i]}\delta_1^{[i]} & \sum_{i=1}^{n} a_2^{[i]}\delta_2^{[i]} & \dots & \sum_{i=1}^{n} a_2^{[i]}\delta_t^{[i]} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} a_d^{[i]}\delta_1^{[i]} & \sum_{i=1}^{n} a_d^{[i]}\delta_2^{[i]} & \dots & \sum_{i=1}^{n} a_d^{[i]}\delta_t^{[i]} \end{bmatrix}$$

The above matrix can be deconstructed in two seperate matrices giving the following

$$\begin{bmatrix} a_1^{[1]} & a_1^{[2]} & \dots & a_1^{[n]} \\ a_2^{[1]} & a_2^{[2]} & \dots & a_2^{[n]} \\ \vdots & \vdots & \ddots & \vdots \\ a_d^{[1]} & a_d^{[2]} & \dots & a_d^{[n]} \end{bmatrix} \begin{bmatrix} \delta_1^{[1]} & \delta_2^{[1]} & \dots & \delta_t^{[1]} \\ \delta_1^{[2]} & \delta_2^{[2]} & \dots & \delta_t^{[2]} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_1^{[n]} & \delta_2^{[n]} & \dots & \delta_t^{[n]} \end{bmatrix} = (A^{(h)})^T \delta^{(out)}$$

Note: the superscript $[i]$ refers to a specific training example.

This is the formula Dr. Raschka used in the book. I hope me walking you through my reasonning helped you reach a better understanding.

8 Wait a minute! What about the bias?

Let's way we have ONE example again. What would the Jacobian matrix, look like, if we only looked at the bias weights? It would like this:

$$\begin{bmatrix} \frac{\partial J(W)}{\partial w_{0,1}^{(out)}} & \frac{\partial J(W)}{\partial w_{0,2}^{(out)}} & \dots & \frac{\partial J(W)}{\partial w_{0,t}^{(out)}} \end{bmatrix} = \begin{bmatrix} a_0^{(h)}\delta_1^{(out)} & a_0^{(h)}\delta_2^{(out)} & \dots & a_0^{(h)}\delta_t^{(out)} \end{bmatrix}$$

But since $a_0^{(h)} = 1$:

$$\begin{bmatrix} a_0^{(h)}\delta_1^{(out)} & a_0^{(h)}\delta_2^{(out)} & \dots & a_0^{(h)}\delta_t^{(out)} \end{bmatrix} = \begin{bmatrix} \delta_1^{(out)} & \delta_2^{(out)} & \dots & \delta_t^{(out)} \end{bmatrix}$$

So, when backpropagating ONE example through $W^{(out)}$, the gradients in the bias weights are given by the above

We saw that as training examples go through the network, the gradients in the weights accumulate. Similarly, after n examples those δ values accumulate. In vector form, that gives the following:

$$\begin{bmatrix} \sum_{i=1}^{n} \delta_1^{[i]} & \sum_{i=1}^{n} \delta_2^{[i]} & \dots & \sum_{i=1}^{n} \delta_t^{[i]} \end{bmatrix}$$