Feedforward Neural Networks in Depth, Part 1: Forward and Backward Propagations

Dec 10, 2021

This post is the first of a three-part series in which we set out to derive the mathematics behind feedforward neural networks. They have

- an input and an output layer with at least one hidden layer in between,
- fully-connected layers, which means that each node in one layer connects to every node in the following layer, and
- ways to introduce nonlinearity by means of activation functions.

We start with forward propagation, which involves computing predictions and the associated cost of these predictions.

Forward Propagation

Settling on what notations to use is tricky since we only have so many letters in the Roman alphabet. As you browse the Internet, you will likely find derivations that have used different notations than the ones we are about to introduce. However, and fortunately, there is no right or wrong here; it is just a matter of taste. In particular, the notations used in this series take inspiration from Andrew Ng's Standard notations for Deep Learning. If you make a comparison, you will find that we only change a couple of the details.

Now, whatever we come up with, we have to support

- multiple layers,
- several nodes in each layer,
- various activation functions,
- various types of cost functions, and
- mini-batches of training examples.

As a result, our definition of a node ends up introducing a fairly large number of notations:

$$
\begin{aligned}
z_{j,i}^{[l]} &= \sum_k w_{j,k}^{[l]} a_{k,i}^{[l-1]} + b_j^{[l]}, &(1)\\
a_{j,i}^{[l]} &= g_j^{[l]}(z_{1,i}^{[l]}, \dots, z_{j,i}^{[l]}, \dots, z_{n^{[l]},i}^{[l]}). &(2)
\end{aligned}
$$

Does the node definition look intimidating to you at first glance? Do not worry. Hopefully, it will make more sense once we have explained the notations, which we shall do next:

Entity	Description
l	The current layer $l = 1, \dots, L$, where L is the number of layers that have weights and biases. We use $l = 0$ and $l = L$ to denote the input and output layers.
$n^{[l]}$	The number of nodes in the current layer.
$n^{[l-1]}$	The number of nodes in the previous layer.
j	The jth node of the current layer, $j = 1, \dots, n^{[l]}$.
k	The kth node of the previous layer, $k = 1, \dots, n^{[l-1]}$.
i	The current training example $i = 1, \dots, m$, where m is the number of training examples.
$z_{j,i}^{[l]}$	A weighted sum of the activations of the previous layer, shifted by a bias.
$w_{j,k}^{[l]}$	A weight that scales the kth activation of the previous layer.
$b_j^{[l]}$	A bias in the current layer.
$a_{j,i}^{[l]}$	An activation in the current layer.
$a_{k,i}^{[l-1]}$	An activation in the previous layer.
$g_j^{[l]}$	An activation function $g_j^{[l]} \colon \mathbb{R}^{n^{[l]}} \to \mathbb{R}$ used in the current layer.

To put it concisely, a node in the current layer depends on every node in the previous layer, and the following visualization can help us see that more clearly:

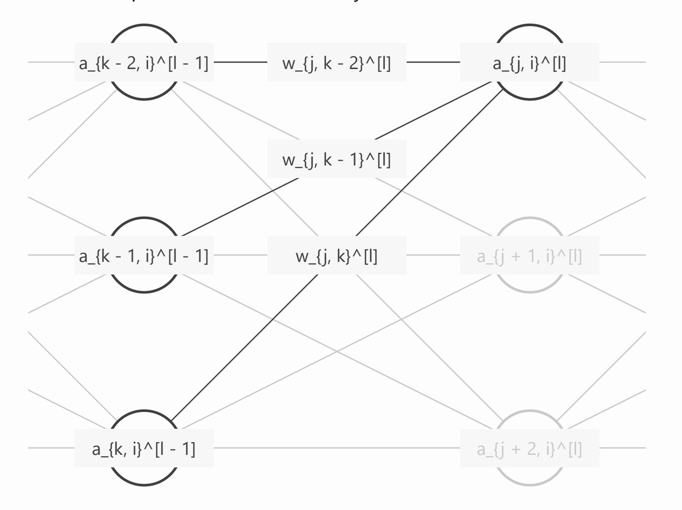

Figure 1: A node in the current layer.

Moreover, a node in the previous layer affects every node in the current layer, and with a change in highlighting, we will also be able to see that more clearly:

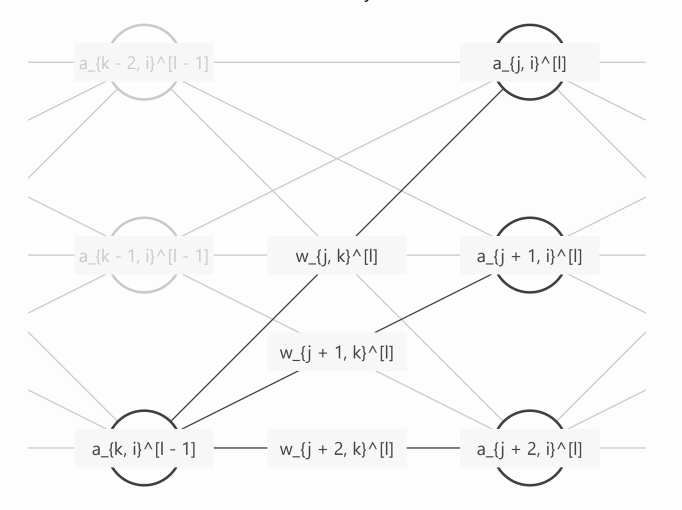

Figure 2: A node in the previous layer.

In the future, we might want to write an implement from scratch in, for example, Python. To take advantage of the heavily optimized versions of vector and matrix operations that come bundled with libraries such as NumPy, we need to vectorize (1) and (2).

To begin with, we vectorize the nodes:

$$
\begin{bmatrix} z_{1,i}^{[l]} \\ \vdots \\ z_{j,i}^{[l]} \\ \vdots \\ z_{n^{[l]},i}^{[l]} \end{bmatrix}
=
\begin{bmatrix}
w_{1,1}^{[l]} & \dots & w_{1,k}^{[l]} & \dots & w_{1,n^{[l-1]}}^{[l]} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
w_{j,1}^{[l]} & \dots & w_{j,k}^{[l]} & \dots & w_{j,n^{[l-1]}}^{[l]} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
w_{n^{[l]},1}^{[l]} & \dots & w_{n^{[l]},k}^{[l]} & \dots & w_{n^{[l]},n^{[l-1]}}^{[l]}
\end{bmatrix}
\begin{bmatrix} a_{1,i}^{[l-1]} \\ \vdots \\ a_{k,i}^{[l-1]} \\ \vdots \\ a_{n^{[l-1]},i}^{[l-1]} \end{bmatrix}
+
\begin{bmatrix} b_1^{[l]} \\ \vdots \\ b_j^{[l]} \\ \vdots \\ b_{n^{[l]}}^{[l]} \end{bmatrix},
$$

$$
\begin{bmatrix} a_{1,i}^{[l]} \\ \vdots \\ a_{j,i}^{[l]} \\ \vdots \\ a_{n^{[l]},i}^{[l]} \end{bmatrix}
=
\begin{bmatrix}
g_1^{[l]}(z_{1,i}^{[l]}, \dots, z_{j,i}^{[l]}, \dots, z_{n^{[l]},i}^{[l]}) \\
\vdots \\
g_j^{[l]}(z_{1,i}^{[l]}, \dots, z_{j,i}^{[l]}, \dots, z_{n^{[l]},i}^{[l]}) \\
\vdots \\
g_{n^{[l]}}^{[l]}(z_{1,i}^{[l]}, \dots, z_{j,i}^{[l]}, \dots, z_{n^{[l]},i}^{[l]})
\end{bmatrix},
$$

which we can write as

$$
\begin{aligned}
\mathbf{z}_{:,i}^{[l]} &= \mathbf{W}^{[l]} \mathbf{a}_{:,i}^{[l-1]} + \mathbf{b}^{[l]}, &(3)\\
\mathbf{a}_{:,i}^{[l]} &= \mathbf{g}^{[l]}(\mathbf{z}_{:,i}^{[l]}), &(4)
\end{aligned}
$$

where $\mathbf{z}_{:,i}^{[l]} \in \mathbb{R}^{n^{[l]}}$, $\mathbf{W}^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}$, $\mathbf{b}^{[l]} \in \mathbb{R}^{n^{[l]}}$, $\mathbf{a}_{:,i}^{[l]} \in \mathbb{R}^{n^{[l]}}$, $\mathbf{a}_{:,i}^{[l-1]} \in \mathbb{R}^{n^{[l-1]}}$, and lastly, $\mathbf{g}^{[l]} \colon \mathbb{R}^{n^{[l]}} \to \mathbb{R}^{n^{[l]}}$. We have used a colon to clarify that $\mathbf{z}_{:,i}^{[l]}$ is the ith column of $\mathbf{Z}^{[l]}$, and so on.

Next, we vectorize the training examples:

$$
\begin{aligned}
\mathbf{Z}^{[l]} &= \begin{bmatrix} \mathbf{z}_{:,1}^{[l]} & \dots & \mathbf{z}_{:,i}^{[l]} & \dots & \mathbf{z}_{:,m}^{[l]} \end{bmatrix} &(5)\\
&= \mathbf{W}^{[l]} \begin{bmatrix} \mathbf{a}_{:,1}^{[l-1]} & \dots & \mathbf{a}_{:,i}^{[l-1]} & \dots & \mathbf{a}_{:,m}^{[l-1]} \end{bmatrix} + \begin{bmatrix} \mathbf{b}^{[l]} & \dots & \mathbf{b}^{[l]} & \dots & \mathbf{b}^{[l]} \end{bmatrix} \\
&= \mathbf{W}^{[l]} \mathbf{A}^{[l-1]} + \operatorname{broadcast}(\mathbf{b}^{[l]}), \\
\mathbf{A}^{[l]} &= \begin{bmatrix} \mathbf{a}_{:,1}^{[l]} & \dots & \mathbf{a}_{:,i}^{[l]} & \dots & \mathbf{a}_{:,m}^{[l]} \end{bmatrix}, &(6)
\end{aligned}
$$

where $\mathbf{Z}^{[l]} \in \mathbb{R}^{n^{[l]} \times m}$, $\mathbf{A}^{[l]} \in \mathbb{R}^{n^{[l]} \times m}$, and $\mathbf{A}^{[l-1]} \in \mathbb{R}^{n^{[l-1]} \times m}$. In addition, have a look at the NumPy documentation if you want to read a well-written explanation of broadcasting.

We would also like to establish two additional notations:

$$
\begin{aligned}
\mathbf{A}^{[0]} &= \mathbf{X}, &(7)\\
\mathbf{A}^{[L]} &= \hat{\mathbf{Y}}, &(8)
\end{aligned}
$$

where $\mathbf{X} \in \mathbb{R}^{n^{[0]} \times m}$ denotes the inputs and $\hat{\mathbf{Y}} \in \mathbb{R}^{n^{[L]} \times m}$ denotes the predictions/outputs.

Finally, we are ready to define the cost function:

$$
J = f(\hat{\mathbf{Y}}, \mathbf{Y}) = f(\mathbf{A}^{[L]}, \mathbf{Y}), \tag{9}
$$

where $\mathbf{Y} \in \mathbb{R}^{n^{[L]} \times m}$ denotes the targets and $f \colon \mathbb{R}^{2n^{[L]}} \to \mathbb{R}$ can be tailored to our needs.

We are done with forward propagation! Next up: backward propagation, also known as backpropagation, which involves computing the gradient of the cost function with respect to the weights and biases.

Backward Propagation

We will make heavy use of the chain rule in this section, and to understand better how it works, we first apply the chain rule to the following example:

$$
\begin{aligned}
u_i &= g_i(x_1, \dots, x_j, \dots, x_n), &(10)\\
y_k &= f_k(u_1, \dots, u_i, \dots, u_m). &(11)
\end{aligned}
$$

Note that x_j may affect $u_1, \dots, u_i, \dots, u_m$, and y_k may depend on $u_1, \dots, u_i, \dots, u_m$; thus,

$$
\frac{\partial y_k}{\partial x_j} = \sum_i \frac{\partial y_k}{\partial u_i} \frac{\partial u_i}{\partial x_j}. \tag{12}
$$

Great! If we ever get stuck trying to compute or understand some partial derivative, we can always go back to (10), (11), and (12). Hopefully, these equations will provide the clues necessary to move forward. However, be extra careful not to confuse the notation used for the chain rule example with the notation we use elsewhere in this series. The overlap is unintentional.

Now, let us concentrate on the task at hand:

$$
\begin{aligned}
\frac{\partial J}{\partial w_{j,k}^{[l]}} &= \sum_i \frac{\partial J}{\partial z_{j,i}^{[l]}} \frac{\partial z_{j,i}^{[l]}}{\partial w_{j,k}^{[l]}} = \sum_i \frac{\partial J}{\partial z_{j,i}^{[l]}} a_{k,i}^{[l-1]}, &(13)\\
\frac{\partial J}{\partial b_j^{[l]}} &= \sum_i \frac{\partial J}{\partial z_{j,i}^{[l]}} \frac{\partial z_{j,i}^{[l]}}{\partial b_j^{[l]}} = \sum_i \frac{\partial J}{\partial z_{j,i}^{[l]}}. &(14)
\end{aligned}
$$

Vectorization results in

$$
\begin{aligned}
&\begin{bmatrix}
\frac{\partial J}{\partial w_{1,1}^{[l]}} & \dots & \frac{\partial J}{\partial w_{1,k}^{[l]}} & \dots & \frac{\partial J}{\partial w_{1,n^{[l-1]}}^{[l]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial J}{\partial w_{j,1}^{[l]}} & \dots & \frac{\partial J}{\partial w_{j,k}^{[l]}} & \dots & \frac{\partial J}{\partial w_{j,n^{[l-1]}}^{[l]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial J}{\partial w_{n^{[l]},1}^{[l]}} & \dots & \frac{\partial J}{\partial w_{n^{[l]},k}^{[l]}} & \dots & \frac{\partial J}{\partial w_{n^{[l]},n^{[l-1]}}^{[l]}}
\end{bmatrix} \\
&= \begin{bmatrix}
\frac{\partial J}{\partial z_{1,1}^{[l]}} & \dots & \frac{\partial J}{\partial z_{1,i}^{[l]}} & \dots & \frac{\partial J}{\partial z_{1,m}^{[l]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial J}{\partial z_{j,1}^{[l]}} & \dots & \frac{\partial J}{\partial z_{j,i}^{[l]}} & \dots & \frac{\partial J}{\partial z_{j,m}^{[l]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial J}{\partial z_{n^{[l]},1}^{[l]}} & \dots & \frac{\partial J}{\partial z_{n^{[l]},i}^{[l]}} & \dots & \frac{\partial J}{\partial z_{n^{[l]},m}^{[l]}}
\end{bmatrix} \\
&\quad \cdot \begin{bmatrix}
a_{1,1}^{[l-1]} & \dots & a_{k,1}^{[l-1]} & \dots & a_{n^{[l-1]},1}^{[l-1]} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
a_{1,i}^{[l-1]} & \dots & a_{k,i}^{[l-1]} & \dots & a_{n^{[l-1]},i}^{[l-1]} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
a_{1,m}^{[l-1]} & \dots & a_{k,m}^{[l-1]} & \dots & a_{n^{[l-1]},m}^{[l-1]}
\end{bmatrix},
\end{aligned}
$$

$$
\begin{bmatrix} \frac{\partial J}{\partial b_1^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial b_j^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial b_{n^{[l]}}^{[l]}} \end{bmatrix}
=
\begin{bmatrix} \frac{\partial J}{\partial z_{1,1}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial z_{j,1}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial z_{n^{[l]},1}^{[l]}} \end{bmatrix}
+ \dots +
\begin{bmatrix} \frac{\partial J}{\partial z_{1,i}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial z_{j,i}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial z_{n^{[l]},i}^{[l]}} \end{bmatrix}
+ \dots +
\begin{bmatrix} \frac{\partial J}{\partial z_{1,m}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial z_{j,m}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial z_{n^{[l]},m}^{[l]}} \end{bmatrix},
$$

which we can write as

$$
\begin{aligned}
\frac{\partial J}{\partial \mathbf{W}^{[l]}} &= \sum_i \frac{\partial J}{\partial \mathbf{z}_{:,i}^{[l]}} \mathbf{a}_{:,i}^{[l-1]\intercal} = \frac{\partial J}{\partial \mathbf{Z}^{[l]}} \mathbf{A}^{[l-1]\intercal}, &(15)\\
\frac{\partial J}{\partial \mathbf{b}^{[l]}} &= \sum_i \frac{\partial J}{\partial \mathbf{z}_{:,i}^{[l]}} = \underbrace{\sum_{\text{axis}=1} \frac{\partial J}{\partial \mathbf{Z}^{[l]}}}_{\text{column vector}}, &(16)
\end{aligned}
$$

where $\partial J/\partial \mathbf{z}_{:,i}^{[l]} \in \mathbb{R}^{n^{[l]}}$, $\partial J/\partial \mathbf{Z}^{[l]} \in \mathbb{R}^{n^{[l]} \times m}$, $\partial J/\partial \mathbf{W}^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}$, and $\partial J/\partial \mathbf{b}^{[l]} \in \mathbb{R}^{n^{[l]}}$.

Looking back at (13) and (14), we see that the only unknown entity is $\partial J/\partial z_{j,i}^{[l]}$. By applying the chain rule once again, we get

$$
\frac{\partial J}{\partial z_{j,i}^{[l]}} = \sum_p \frac{\partial J}{\partial a_{p,i}^{[l]}} \frac{\partial a_{p,i}^{[l]}}{\partial z_{j,i}^{[l]}}, \tag{17}
$$

where $p = 1, \dots, n^{[l]}$.

Next, we present the vectorized version:

$$
\begin{bmatrix} \frac{\partial J}{\partial z_{1,i}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial z_{j,i}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial z_{n^{[l]},i}^{[l]}} \end{bmatrix}
=
\begin{bmatrix}
\frac{\partial a_{1,i}^{[l]}}{\partial z_{1,i}^{[l]}} & \dots & \frac{\partial a_{j,i}^{[l]}}{\partial z_{1,i}^{[l]}} & \dots & \frac{\partial a_{n^{[l]},i}^{[l]}}{\partial z_{1,i}^{[l]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial a_{1,i}^{[l]}}{\partial z_{j,i}^{[l]}} & \dots & \frac{\partial a_{j,i}^{[l]}}{\partial z_{j,i}^{[l]}} & \dots & \frac{\partial a_{n^{[l]},i}^{[l]}}{\partial z_{j,i}^{[l]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial a_{1,i}^{[l]}}{\partial z_{n^{[l]},i}^{[l]}} & \dots & \frac{\partial a_{j,i}^{[l]}}{\partial z_{n^{[l]},i}^{[l]}} & \dots & \frac{\partial a_{n^{[l]},i}^{[l]}}{\partial z_{n^{[l]},i}^{[l]}}
\end{bmatrix}
\begin{bmatrix} \frac{\partial J}{\partial a_{1,i}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial a_{j,i}^{[l]}} \\ \vdots \\ \frac{\partial J}{\partial a_{n^{[l]},i}^{[l]}} \end{bmatrix},
$$

which compresses into

$$
\frac{\partial J}{\partial \mathbf{z}_{:,i}^{[l]}} = \frac{\partial \mathbf{a}_{:,i}^{[l]}}{\partial \mathbf{z}_{:,i}^{[l]}} \frac{\partial J}{\partial \mathbf{a}_{:,i}^{[l]}}, \tag{18}
$$

where $\partial J/\partial \mathbf{a}_{:,i}^{[l]} \in \mathbb{R}^{n^{[l]}}$ and $\partial \mathbf{a}_{:,i}^{[l]}/\partial \mathbf{z}_{:,i}^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l]}}$.

We have already encountered

$$
\frac{\partial J}{\partial \mathbf{Z}^{[l]}} = \begin{bmatrix} \frac{\partial J}{\partial \mathbf{z}_{:,1}^{[l]}} & \dots & \frac{\partial J}{\partial \mathbf{z}_{:,i}^{[l]}} & \dots & \frac{\partial J}{\partial \mathbf{z}_{:,m}^{[l]}} \end{bmatrix}, \tag{19}
$$

and for the sake of completeness, we also clarify that

$$
\frac{\partial J}{\partial \mathbf{A}^{[l]}} = \begin{bmatrix} \frac{\partial J}{\partial \mathbf{a}_{:,1}^{[l]}} & \dots & \frac{\partial J}{\partial \mathbf{a}_{:,i}^{[l]}} & \dots & \frac{\partial J}{\partial \mathbf{a}_{:,m}^{[l]}} \end{bmatrix}, \tag{20}
$$

where $\partial J/\partial \mathbf{A}^{[l]} \in \mathbb{R}^{n^{[l]} \times m}$.

On purpose, we have omitted the details of $g_j^{[l]}(z_{1,i}^{[l]}, \dots, z_{j,i}^{[l]}, \dots, z_{n^{[l]},i}^{[l]})$; consequently, we cannot derive an analytic expression for $\partial a_{j,i}^{[l]}/\partial z_{j,i}^{[l]}$, which we depend on in (17). However, since the second post of this series will be dedicated to activation functions, we will instead derive $\partial a_{j,i}^{[l]}/\partial z_{j,i}^{[l]}$ there.

Furthermore, according to (17), we see that $\partial J/\partial z_{j,i}^{[l]}$ also depends on $\partial J/\partial a_{j,i}^{[l]}$. Now, it might come as a surprise, but $\partial J/\partial a_{j,i}^{[l]}$ has already been computed when we reach the lth layer during backward propagation. How did that happen, you may ask. The answer is that every layer paves the way for the previous layer by also computing $\partial J/\partial a_{k,i}^{[l-1]}$, which we shall do now:

$$
\frac{\partial J}{\partial a_{k,i}^{[l-1]}} = \sum_j \frac{\partial J}{\partial z_{j,i}^{[l]}} \frac{\partial z_{j,i}^{[l]}}{\partial a_{k,i}^{[l-1]}} = \sum_j \frac{\partial J}{\partial z_{j,i}^{[l]}} w_{j,k}^{[l]}. \tag{21}
$$

As usual, our next step is vectorization:

$$
\begin{aligned}
&\begin{bmatrix}
\frac{\partial J}{\partial a_{1,1}^{[l-1]}} & \dots & \frac{\partial J}{\partial a_{1,i}^{[l-1]}} & \dots & \frac{\partial J}{\partial a_{1,m}^{[l-1]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial J}{\partial a_{k,1}^{[l-1]}} & \dots & \frac{\partial J}{\partial a_{k,i}^{[l-1]}} & \dots & \frac{\partial J}{\partial a_{k,m}^{[l-1]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial J}{\partial a_{n^{[l-1]},1}^{[l-1]}} & \dots & \frac{\partial J}{\partial a_{n^{[l-1]},i}^{[l-1]}} & \dots & \frac{\partial J}{\partial a_{n^{[l-1]},m}^{[l-1]}}
\end{bmatrix} \\
&= \begin{bmatrix}
w_{1,1}^{[l]} & \dots & w_{j,1}^{[l]} & \dots & w_{n^{[l]},1}^{[l]} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
w_{1,k}^{[l]} & \dots & w_{j,k}^{[l]} & \dots & w_{n^{[l]},k}^{[l]} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
w_{1,n^{[l-1]}}^{[l]} & \dots & w_{j,n^{[l-1]}}^{[l]} & \dots & w_{n^{[l]},n^{[l-1]}}^{[l]}
\end{bmatrix} \\
&\quad \cdot \begin{bmatrix}
\frac{\partial J}{\partial z_{1,1}^{[l]}} & \dots & \frac{\partial J}{\partial z_{1,i}^{[l]}} & \dots & \frac{\partial J}{\partial z_{1,m}^{[l]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial J}{\partial z_{j,1}^{[l]}} & \dots & \frac{\partial J}{\partial z_{j,i}^{[l]}} & \dots & \frac{\partial J}{\partial z_{j,m}^{[l]}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial J}{\partial z_{n^{[l]},1}^{[l]}} & \dots & \frac{\partial J}{\partial z_{n^{[l]},i}^{[l]}} & \dots & \frac{\partial J}{\partial z_{n^{[l]},m}^{[l]}}
\end{bmatrix},
\end{aligned}
$$

which we can write as

$$
\frac{\partial J}{\partial \mathbf{A}^{[l-1]}} = \mathbf{W}^{[l]\intercal} \frac{\partial J}{\partial \mathbf{Z}^{[l]}}, \tag{22}
$$

where $\partial J/\partial \mathbf{A}^{[l-1]} \in \mathbb{R}^{n^{[l-1]} \times m}$.

Summary

Forward propagation is seeded with $\mathbf{A}^{[0]} = \mathbf{X}$ and evaluates a set of recurrence relations to compute the predictions $\mathbf{A}^{[L]} = \hat{\mathbf{Y}}$. We also compute the cost $J = f(\hat{\mathbf{Y}}, \mathbf{Y}) = f(\mathbf{A}^{[L]}, \mathbf{Y})$.

Backward propagation, on the other hand, is seeded with $\partial J/\partial \mathbf{A}^{[L]} = \partial J/\partial \hat{\mathbf{Y}}$ and evaluates a different set of recurrence relations to compute $\partial J/\partial \mathbf{W}^{[l]}$ and $\partial J/\partial \mathbf{b}^{[l]}$. If not stopped prematurely, it eventually computes $\partial J/\partial \mathbf{A}^{[0]} = \partial J/\partial \mathbf{X}$, a partial derivative we usually ignore.

Moreover, let us visualize the inputs we use and the outputs we produce during the forward and backward propagations:

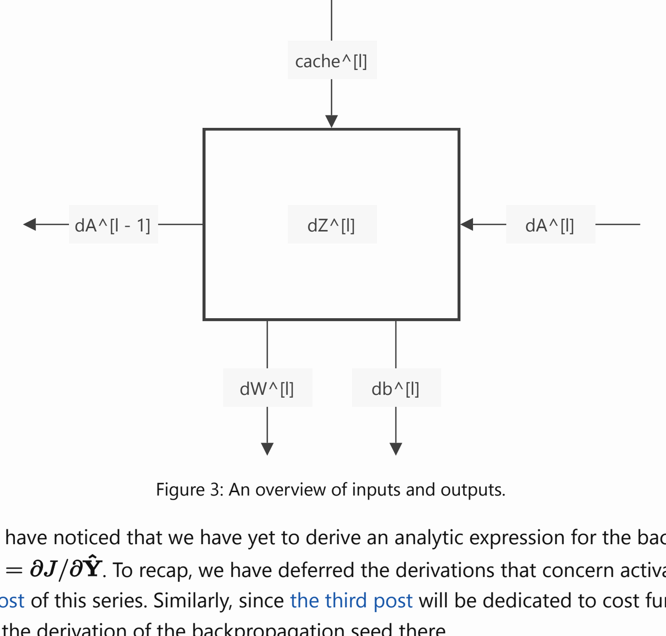

Figure 3: An overview of inputs and outputs.

Now, you might have noticed that we have yet to derive an analytic expression for the backpropagation seed $\partial J/\partial \mathbf{A}^{[L]} = \partial J/\partial \hat{\mathbf{Y}}$. To recap, we have deferred the derivations that concern activation functions to the second post of this series. Similarly, since the third post will be dedicated to cost functions, we will instead address the derivation of the backpropagation seed there.

Last but not least: congratulations! You have made it to the end (of the first post).