Dense Neural Networks

Tadeu Silva (Deds)

July 22, 2021

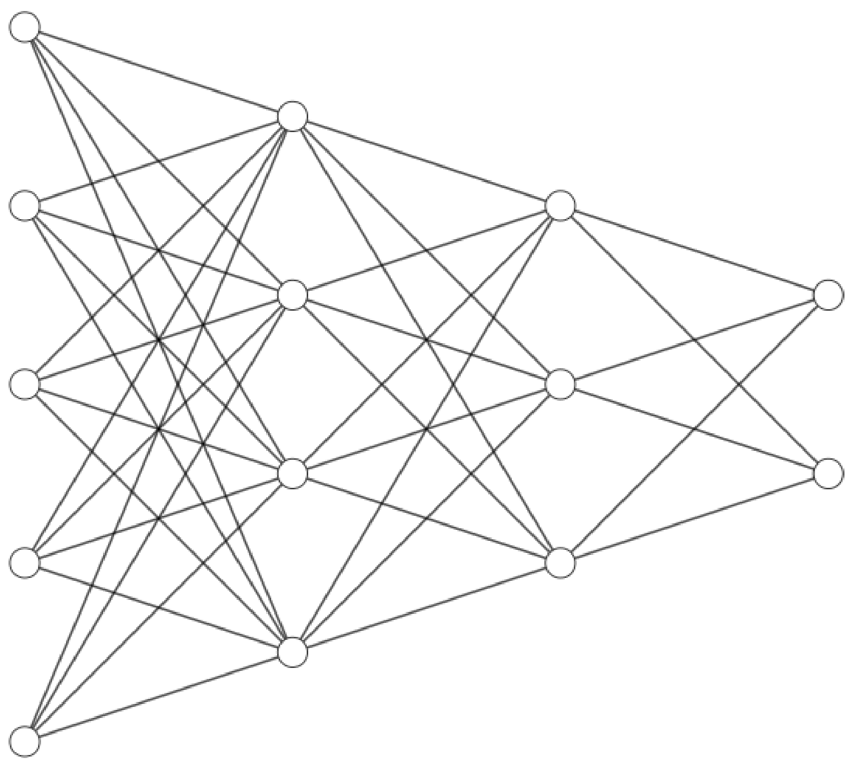

Figure 1: Architecture of Neural Network

1 Introduction

A Neural Network is made by two steps: **Forward** pass and **Backward** pass. In the Forward pass, the next layer is a linear transformationa between the previous one, with an activation function in the end of each result. In the Backward pass, we asses the **loss** (how our prediction is wrong from the actual data) and update the parameters (weights and bias) according to an **Optimizer** (in this case Gradient Descent).

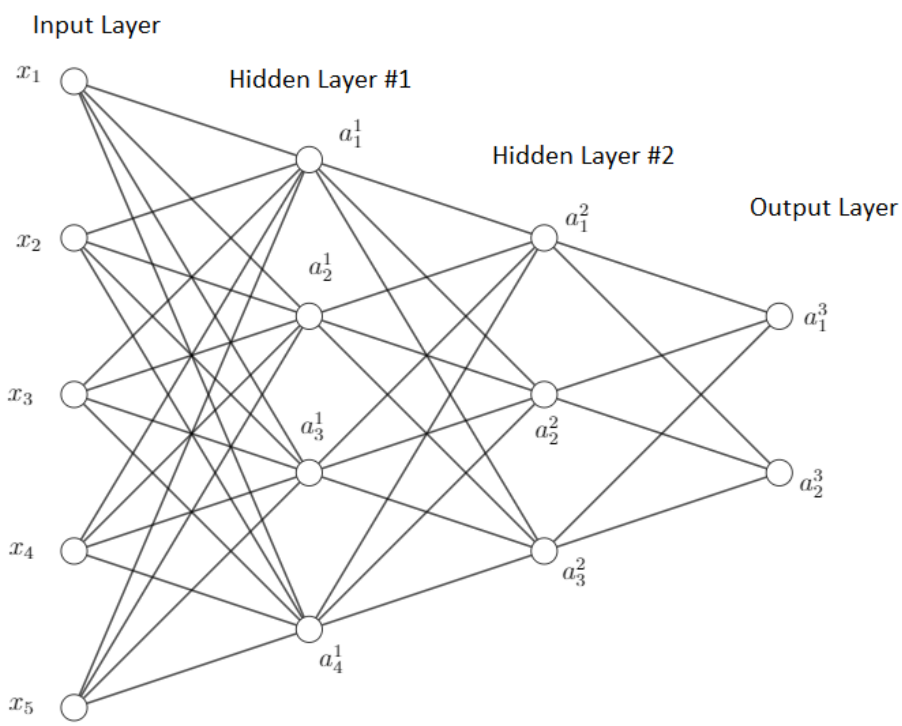

Figure 2: Dense Neural Network: Activation

2 Forward Pass

Given an input $\overrightarrow{x}$ in the Input Layer, the Hidden Layer #1 has output:

$$z^1 = W^1 \cdot \overrightarrow{x} + b^1 \tag{1}$$

where W is the **Weights** matrix, b is the **bias** vector and $\overrightarrow{x}$ is the **output** vector.
Obs:

- $\overrightarrow{x}$ is a $(m, 1)$ vector
- W is a (n, m) matrix
- b is a $(n, 1)$ matrix

Then we **activate** the next layer by an Activation Funtion, to simulate nonlinear behavior (otherwise we are only doing linear regression). The final result for the next layer is:

$$a^1 = \sigma(z^1) = \sigma(W^1 \cdot \overrightarrow{x} + b^1) \tag{2}$$

Obs: The input is called $\overrightarrow{x}$, but the other layers are called **a**

So, in general, the activation of layer l is given by:

$$a^l = \sigma(z^l) = \sigma(W^l \cdot a^{l-1} + b^l) \tag{3}$$

2.1 Example

In our example at figure 2 we have:

Hidden Layer # 1

$$\begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_4^1 \end{bmatrix} = \sigma\left(\begin{bmatrix} z_1^1 \\ z_2^1 \\ z_3^1 \\ z_4^1 \end{bmatrix}\right) = \sigma\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \cdot \begin{bmatrix} W_{11}^1 & W_{12}^1 & W_{13}^1 & W_{14}^1 & W_{15}^1 \\ W_{21}^1 & W_{22}^1 & W_{23}^1 & W_{24}^1 & W_{25}^1 \\ W_{31}^1 & W_{32}^1 & W_{33}^1 & W_{34}^1 & W_{35}^1 \\ W_{41}^1 & W_{42}^1 & W_{43}^1 & W_{44}^1 & W_{45}^1 \end{bmatrix} + \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \\ b_4^1 \end{bmatrix}\right) \tag{4}$$

Hidden Layer # 2

$$\begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix} = \sigma\left(\begin{bmatrix} z_1^2 \\ z_2^2 \\ z_3^2 \end{bmatrix}\right) = \sigma\left(\begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_4^1 \end{bmatrix} \cdot \begin{bmatrix} W_{11}^2 & W_{12}^2 & W_{13}^2 & W_{14}^2 \\ W_{21}^2 & W_{22}^2 & W_{23}^2 & W_{24}^2 \\ W_{31}^2 & W_{32}^2 & W_{33}^2 & W_{34}^2 \end{bmatrix} + \begin{bmatrix} b_1^2 \\ b_2^2 \\ b_3^2 \end{bmatrix}\right) \tag{5}$$

Output Layer

$$\begin{bmatrix} a_1^3 \\ a_2^3 \end{bmatrix} = \sigma\left(\begin{bmatrix} z_1^3 \\ z_2^3 \end{bmatrix}\right) = \sigma\left(\begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix} \cdot \begin{bmatrix} W_{11}^3 & W_{12}^3 & W_{13}^3 \\ W_{21}^3 & W_{22}^3 & W_{23}^3 \end{bmatrix} + \begin{bmatrix} b_1^3 \\ b_2^3 \end{bmatrix}\right) \tag{6}$$

3 Backward Pass

3.1 Loss

In the Backward pass, we first compute our **loss** (aka the error of our output) and average along the data, to get a measure of our total uncertainty. So:

$$loss = (\frac{1}{n})\sum_{i=0}^{n} loss(a_i^2, y_i) \tag{7}$$

where n is the dimension of the output layer. One option is to use the **Mean Squared Error** as the loss. So in our example we have:

$$loss = (\frac{1}{n})\sum_{i=0}^{n}(a_i^2 - y_i)^2 \tag{8}$$

3.2 Backpropagation

The second part is to calculate how each weight and bias had influence in the loss. So we want to calculate:

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial W} \\ \\ \frac{\partial C}{\partial b} \end{bmatrix} \tag{9}$$

We do that using the Backpropagation (chain rule). Remembering from calculus, the derivative of a composite function $f(g(x))$ can be written as:

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx} \tag{10}$$

Another thing that is going to be useful is remember how to calculate the derivative of matrices. So, given a scalar function C (even do our loss depends of **y** and **a**, we can treat it as scalar), the derivative with respect to the matrix W, is given by:

$$\frac{\partial C}{\partial W} = \begin{pmatrix} \frac{\partial C}{\partial W_{11}} & \frac{\partial C}{\partial W_{12}} & \cdots & \frac{\partial C}{\partial W_{1m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial C}{\partial W_{n1}} & \frac{\partial C}{\partial W_{n2}} & \cdots & \frac{\partial C}{\partial W_{nm}} \end{pmatrix} \tag{11}$$

Similarly to the bias b:

$$\frac{\partial C}{\partial b} = \begin{pmatrix} \frac{\partial C}{\partial b_1} \\ \vdots \\ \frac{\partial C}{\partial b_n} \end{pmatrix} \tag{12}$$

Following that same logic, we have $a(z(W))$, then we start from the Output layer and go backwards to the Input layer, as follows:

$$\frac{\partial C}{\partial W} = \left(\frac{\partial C}{\partial a}\right)\left(\frac{\partial a}{\partial z}\right)\left(\frac{\partial z}{\partial W}\right) \tag{13}$$

$$\frac{\partial C}{\partial b} = \left(\frac{\partial C}{\partial a}\right)\left(\frac{\partial a}{\partial z}\right)\left(\frac{\partial z}{\partial b}\right) \tag{14}$$

For the <u>third</u> term, note that:

$$z^l = W^l \cdot a^{l-1} + b^l \Rightarrow \frac{\partial z^l}{\partial W^l} = a^{l-1}$$
$$z^l = W^l \cdot a^{l-1} + b^l \Rightarrow \frac{\partial z^l}{\partial b^l} = 1$$

For the <u>second</u> term:

$$\frac{\partial a}{\partial z} = \frac{d\sigma}{\partial z}$$

The activation function σ depends on the problem at hand. But let's suppose we are at a **classification** problem, so we'll go with the **Softmax** function, defined as:

$$\sigma(z) = \frac{e^z}{\sum_{j=0}^{n} e^{z_j}} \tag{15}$$

So the equation is:

$$\frac{\partial a^l}{\partial z^l} = \frac{d\sigma}{\partial z^l} = z^l(1 - z^l)$$

For the <u>first</u> term, we have two options:

1. Output Layer

For the Output Layer, we just have the loss as our main metric. So the equation is:

$$\frac{\partial C}{\partial a} = \frac{dloss}{da}$$

In our example loss is **MSE**, so:

$$\frac{\partial C}{\partial a^2} = 2(a_2^2 - y_i)$$

2. Other Layers
In other layers, we have $a^l = a^{l+1}(z^{l+1}(a^l))$
This means that the error in the layer l influences the error of layer l+1, and that's why the chain rule is so powerfull in keeping track of those links. So the equations goes as follows:

$$\frac{\partial C}{\partial a^l} = \left(\frac{\partial C}{\partial a^{l+1}}\right)\left(\frac{\partial a^{l+1}}{\partial z^{l+1}}\right)\left(\frac{\partial z^{l+1}}{\partial a^l}\right)$$

3.2.1 Example

Let's denote the following two operations:

- $\times$: element-wise multiplication, i.e. same dimensions only
- $\cdot$: dot product, i.e (m,k) $\cdot$ (k,n) = (m,n)

This means that if:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \Rightarrow AXB = \begin{bmatrix} a_{11} \cdot b_{11} & a_{12} \cdot b_{12} \\ a_{21} \cdot b_{21} & a_{22} \cdot b_{22} \end{bmatrix}$$

Analogously, if:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, B = \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} \Rightarrow A \cdot B = \begin{bmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} \end{bmatrix}$$

In our example at figure 1 we have:

Output Layer - Weights

$$\frac{\partial C}{\partial W^3} = \left(\frac{\partial C}{\partial a^3}\right)\left(\frac{\partial a^3}{\partial z^3}\right)\left(\frac{\partial z^3}{\partial W^3}\right) \tag{16}$$

$$\frac{\partial C}{\partial W^3} = \left[2\left(\begin{bmatrix} a_1^3 - y_1 \\ a_2^3 - y_2 \end{bmatrix}\right) \times \begin{bmatrix} z_1^3(1-z_1^3) \\ z_2^3(1-z_2^3) \end{bmatrix}\right] \cdot \left(\begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix}\right)^T \tag{17}$$

Output Layer - Bias

$$\frac{\partial C}{\partial b^3} = \left(\frac{\partial C}{\partial a^3}\right)\left(\frac{\partial a^3}{\partial z^3}\right) \tag{18}$$

$$\frac{\partial C}{\partial b^3} = 2\left(\begin{bmatrix} a_1^3 - y_1 \\ a_2^3 - y_2 \end{bmatrix}\right) \times \begin{bmatrix} z_1^3(1-z_1^3) \\ z_2^3(1-z_2^3) \end{bmatrix} \tag{19}$$

Hidden Layer # 2 - Weights

$$\frac{\partial C}{\partial W^2} = \left(\frac{\partial C}{\partial a^2}\right)\left(\frac{\partial a^2}{\partial z^2}\right)\left(\frac{\partial z^2}{\partial w^2}\right) \tag{20}$$

where:

$$\frac{\partial C}{\partial a^2} = \left(\frac{\partial C}{\partial a^3}\right)\left(\frac{\partial a^3}{\partial z^3}\right)\left(\frac{\partial z^3}{\partial a^2}\right) \tag{21}$$

Note that:

$$z^3 = W^3 \cdot a^2 + b^3 \Rightarrow \frac{\partial z^3}{\partial a^2} = W^3$$

So:

$$\frac{\partial C}{\partial a^2} = \left(\left[2\left(\begin{bmatrix} a_1^3 - y_1 \\ a_2^3 - y_2 \end{bmatrix}\right) \times \begin{bmatrix} z_1^3(1-z_1^3) \\ z_2^3(1-z_2^3) \end{bmatrix}\right]^T \cdot \begin{bmatrix} W_{11}^3 & W_{12}^3 & W_{13}^3 \\ W_{21}^3 & W_{22}^3 & W_{23}^3 \end{bmatrix}\right)^T \tag{22}$$

And finally:

$$\frac{\partial C}{\partial W^2} = \left(\frac{\partial C}{\partial a^2}\right) \times \begin{bmatrix} z_1^2(1-z_1^2) \\ z_2^2(1-z_2^2) \\ z_3^2(1-z_3^2) \end{bmatrix}\Bigg] \cdot \left(\begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_4^1 \end{bmatrix}\right)^T \tag{23}$$

Hidden Layer # 2 - Bias

$$\frac{\partial C}{\partial W^2} = \left(\frac{\partial C}{\partial a^2}\right) \times \begin{bmatrix} z_1^2(1-z_1^2) \\ z_2^2(1-z_2^2) \\ z_3^2(1-z_3^2) \end{bmatrix} \tag{24}$$

Hidden Layer # 1 - Weights

$$\frac{\partial C}{\partial W^1} = \left(\frac{\partial C}{\partial a^1}\right)\left(\frac{\partial a^1}{\partial z^1}\right)\left(\frac{\partial z^1}{\partial w^1}\right) \tag{25}$$

where:

$$\frac{\partial C}{\partial a^1} = \left(\frac{\partial C}{\partial a^2}\right)\left(\frac{\partial a^2}{\partial z^2}\right)\left(\frac{\partial z^2}{\partial a^1}\right) \tag{26}$$

Analogously as the Hidden Layer # 2 we get the influence from every layer, starting from the output layer, remembering now that:

$$z^2 = W^2 \cdot a^1 + b^2 \Rightarrow \frac{\partial z^2}{\partial a^1} = W^2$$

So the final equation is:

$$\frac{\partial C}{\partial W^1} = \left(\frac{\partial C}{\partial a^1}\right) \times \begin{bmatrix} z_1^1(1-z_1^1) \\ z_2^1(1-z_2^1) \\ z_3^1(1-z_3^1) \\ z_4^1(1-z_4^1) \end{bmatrix} \Bigg] \cdot \left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right)^T \tag{27}$$

Hidden Layer # 1 - Bias

$$\frac{\partial C}{\partial W^1} = \left(\frac{\partial C}{\partial a^1}\right) \times \begin{bmatrix} z_1^1(1-z_1^1) \\ z_2^1(1-z_2^1) \\ z_3^1(1-z_3^1) \\ z_4^1(1-z_4^1) \end{bmatrix} \tag{28}$$

3.3 Optimizing

For the last step, we optimize the weights accordingly to the changes calculated on the previous section. There are a few optimizers, but the two most used are: **SGD** (Stochastic Gradient Descent) and **Adam**.

SGD - Stochastic Gradient Descent
The SGD is based in the simple idea that: the gradient shows the growth direction. Because we want to **decrease** our error/loss, we simple get the opposite direction adding the minus sign. So the equation is:

$$W^l = W^l - lr \times \frac{\partial C}{\partial W^l} \tag{29}$$

$$b^l = b^l - lr \times \frac{\partial C}{\partial b^l} \tag{30}$$

where lr is the **learning rate**, a parameter $(lr < 1)$ just to make sure that we don't make great jumps and 'miss' the local minima.

Depending in the size of our data, we usually don't go through **every** data available through each iteration (usually called epochs). The Stochastic Gradient Descent takes a **batch** size, where we do the forward/backwards pass only in a sample of the data, to minimize computation costs.