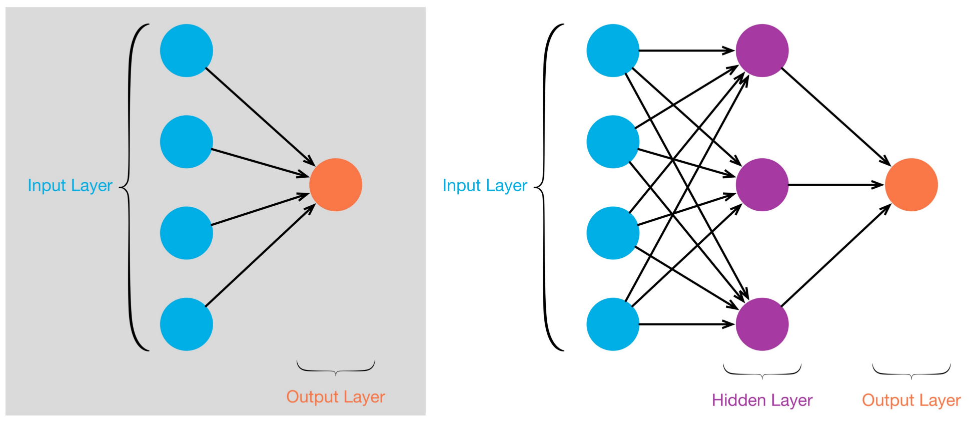
1 Intro

Neural networks became increasingly popular in the last few years as showed by the research publication topics. The main advantage of those models is that they can learn complex structures and are highly customizable. However, one significant drawback is their poor explanability power. Neural networks are often categorized in the so-called "black box models". Fortunately, there are some very good tutorials on understanding how a neural network actually works. Today, every data scientist knows the tutorials of Andrew Ng.

After having done those tutorials, I personally found out that a 1-layer network is relatively easy to understand, however it gets tricky when increasing the number of layers. This article is an attempt to understand in details the 2-layer neural network through the building of a small playground. We will even publish the playground online! To enjoy fully this article, the reader must be familiar with the working of a 1-layer neural network.

There are already an elephantic number of articles on the maths behind neural networks, so I will focus more on the building app as I believe is less frequent.

The difference between a 1-layer network and a 2-layer network relies in the following picture:



Note that the input layer is usually ignored when counting the layers.

I won't give much details on the 1-layer, I strongly recommend Andrew Ng's tutorial for more details or this article for a wrap up.

Let's now go into deeper details on the working on a 2-layer network.

2 2-layer NN

For the sake of simplicity, we will use a binary classification. Thus, the cost function is:

$$\mathcal{L}(y,\widehat{y}) = -[y\log(\widehat{y}) + (1-y)\log(1-\widehat{y})]$$

We notice that the loss function decreases when $y \neq \widehat{y}$, which is expected as we want to penalize wrong classification.

Forward propagation (with derivatives)

$$\begin{tikzcd}
x \arrow[dr] & & W^{[2]} \arrow[dr] & & & \\
W^{[1]} \arrow[r] & z^{[1]} = W^{[1]}x + b^{[1]} \arrow[r] & a^{[1]} = \sigma(z^{[1]}) \arrow[r] & z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \arrow[r] & a^{[2]} = \sigma(z^{[2]}) \arrow[r] & \mathcal{L}(a^{[2]}, y) \\
b^{[1]} \arrow[ur] & & b^{[2]} \arrow[ur] & & &
\end{tikzcd}$$

As we can see here, we use two sets of weights and biases: $W^{[1]}$, $b^{[1]}$, $W^{[2]}$ and $b^{[2]}$.

Backward propagation (with derivatives)

$$\begin{tikzcd}
x & & W^{[2]} & & & \\
W^{[1]} & z^{[1]} = W^{[1]}x + b^{[1]} \arrow[ul] \arrow[l] \arrow[dl] & a^{[1]} = \sigma(z^{[1]}) \arrow[l] & z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \arrow[l] \arrow[ul] \arrow[dl] & a^{[2]} = \sigma(z^{[2]}) \arrow[l] & \mathcal{L}(a^{[2]}, y) \arrow[l] \\
b^{[1]} & & b^{[2]} & & &
\end{tikzcd}$$

For the full details on the derivative computation, I explain everything on my website (hopefully I didn't do too many mistakes).

3 Building the app