Neural Networks (NN)

NON-LINEAR MODEL

Logistic regression is good if we have 2 features in our problem, for example:

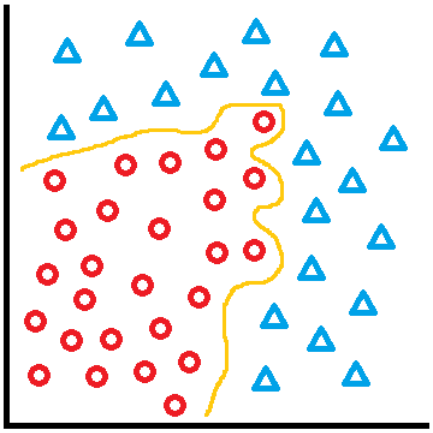


Figure 1. Example of Logistic Regression Problem with 2 Features

However, if we have lots of features, we'll need to come up with hypothesis that include all of these features.

With too many features, it might lead to overfitting problem and it can also be computationally expensive. Because of that, we'll need a new machine learning algorithm (Neural networks).

MODEL REPRESENTATION

Neural networks was developed to simulate the networks of neurons in the brain. Some components in neural network include:

- Input wires
- Output wires

Neuron model: logistic unit:

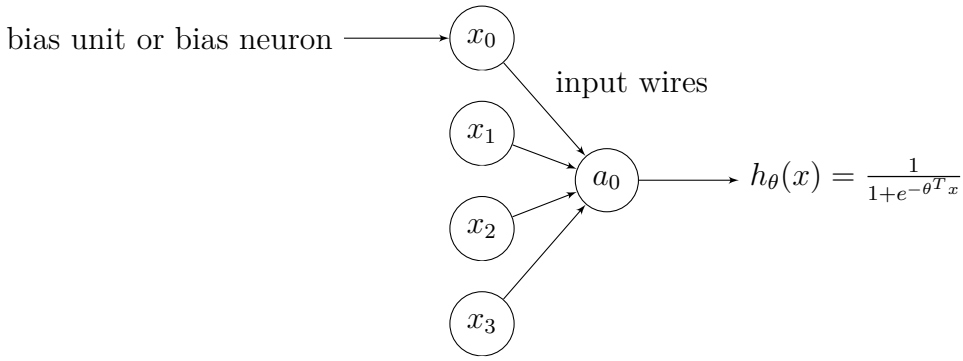


Figure 2. An example of a neuron model

The neuron in figure 2 is called an artificial neuron with a sigmoid activation function.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

The vector θ is called "parameters" or "weights"

Neural network:

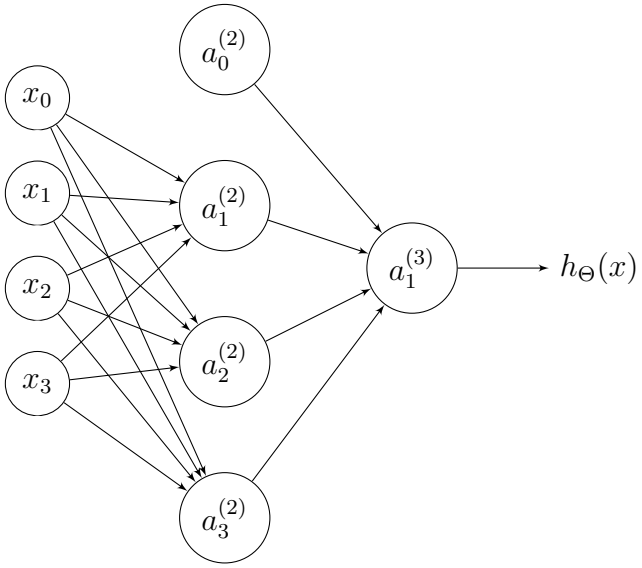


Figure 3. An example of a neuron network

Some new notations:

- $a_i^{(j)}$: "activation" of unit i in layer j
- $\Theta^{(j)}$: matrix of weights controlling function mapping from layer j to layer $j+1$

For example:

- $\Theta^{(1)}$: the matrix of weights controlling function mapping from layer 0 to layer 1
- $\Theta_{12}^{(4)}$: the weight of the mapping from neuron 2 in layer 3 to neuron 1 in layer 4

Consider the example in figure 3, we will have:

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3) \qquad \textbf{Equation } 1.$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3) \qquad \textbf{Equation } 2.$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3) \qquad \textbf{Equation } 3.$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}) \quad \textbf{Equation } 4.$$

Consequently, we can see the matrix of the weights Θ as a 3-D matrix (sort of) as depicted in figure 4 below.

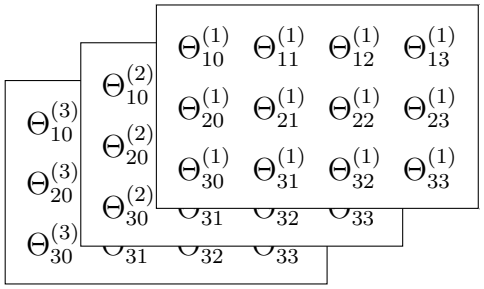
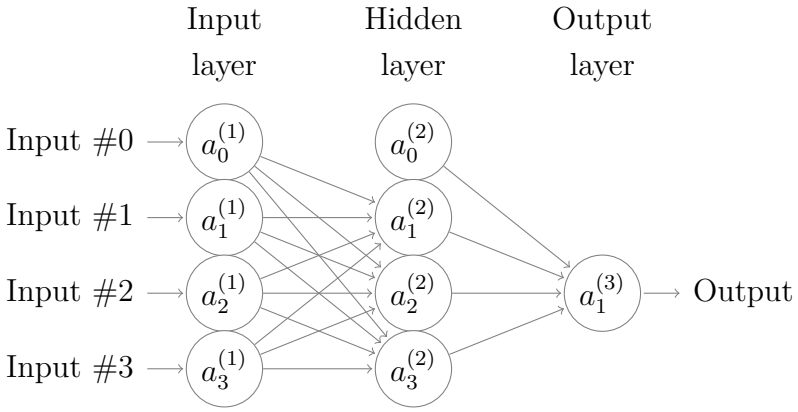


Figure 4. Visualization of matrix Θ for example in figure 3

Consider an example of a neural network with 1 hidden layer having artificial neurons with a sigmoid activation function, we can write the activation functions in the hidden layer as follows:



$$a_1^{(2)} = g(\Theta_{10}^{(1)} a_0^{(1)} + \Theta_{11}^{(1)} a_1^{(1)} + \Theta_{12}^{(1)} a_2^{(1)} + \Theta_{13}^{(1)} a_3^{(1)}) = g(z_1^{(2)}) \quad \textbf{Equation } 5.$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} a_0^{(1)} + \Theta_{21}^{(1)} a_1^{(1)} + \Theta_{22}^{(1)} a_2^{(1)} + \Theta_{23}^{(1)} a_3^{(1)}) = g(z_2^{(2)}) \quad \textbf{Equation } 6.$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} a_0^{(1)} + \Theta_{31}^{(1)} a_1^{(1)} + \Theta_{32}^{(1)} a_2^{(1)} + \Theta_{33}^{(1)} a_3^{(1)}) = g(z_3^{(2)}) \quad \textbf{Equation } 7.$$

$$h_\Theta(a^{(1)}) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}) = g(z^{(3)}) \quad \textbf{Equation } 8.$$

When we consider $a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$ as a whole, we can see the vectorized computation of the layer 2 (hidden layer) of the neural network as $\Theta^{(1)} a^{(1)}$ with $\Theta^{(1)} = \begin{pmatrix} \Theta_{10}^{(1)} & \Theta_{11}^{(1)} & \Theta_{12}^{(1)} & \Theta_{13}^{(1)} \\ \Theta_{20}^{(1)} & \Theta_{21}^{(1)} & \Theta_{22}^{(1)} & \Theta_{23}^{(1)} \\ \Theta_{30}^{(1)} & \Theta_{31}^{(1)} & \Theta_{32}^{(1)} & \Theta_{33}^{(1)} \end{pmatrix}$ and $a^{(1)} = \begin{pmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix}$

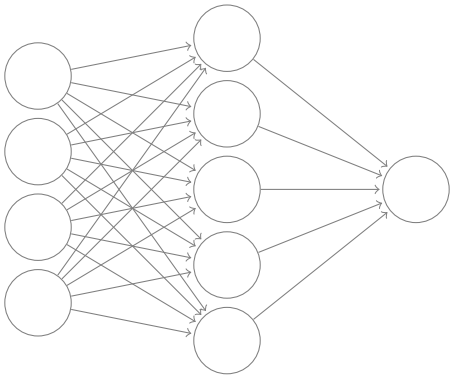
Consider $a^{(1)} = \begin{pmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix}$ and $z^{(2)} = \begin{pmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{pmatrix}$, we'll have $z^{(2)} = \Theta^{(1)} a^{(1)}$. Therefore, the activation functions of layer 2 would be $a^{(2)} = g(z^{(2)})$

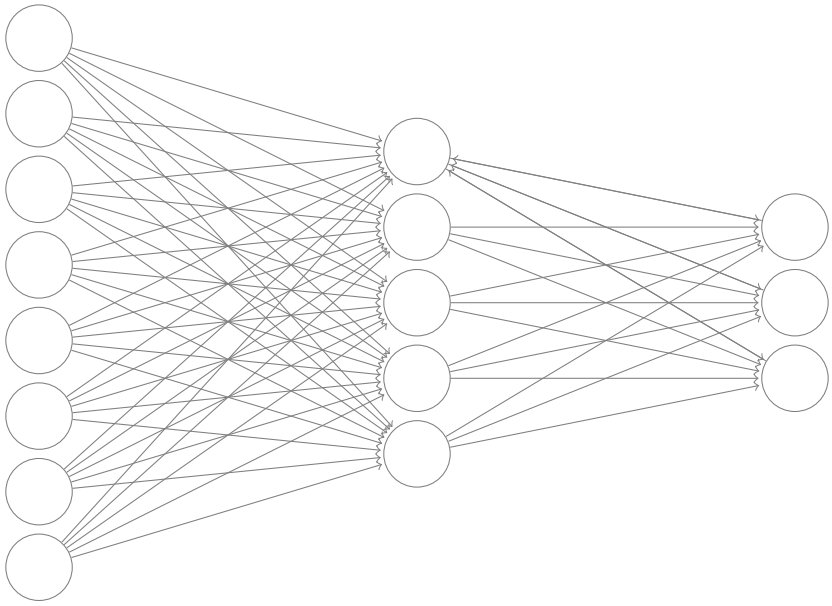
After that, since we define the bias neuron in each layer as 1 by default, we'll add $a_0^{(2)} = 1$ into the newly computed vector $a^{(2)}$.

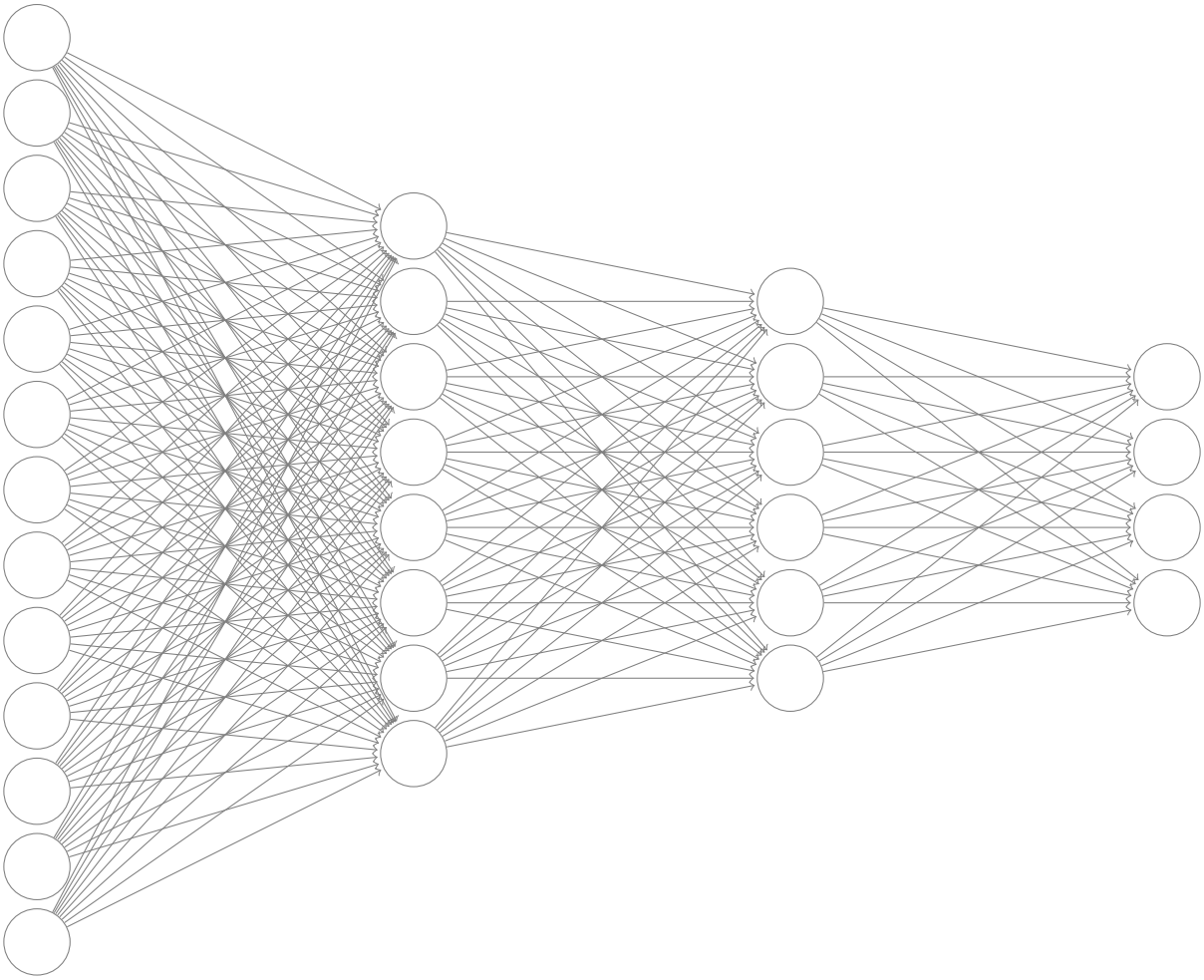
Finally, the hypothesis would be the activation function(s) of the output layer, which can be computed as follows: $z^{(3)} = \Theta^{(2)} a^{(2)}$ and $h_\Theta(a^{(1)}) = a^{(3)} = g(z^{(3)})$

This process of computing $h_\Theta(x)$ is called forward propagation: we start with the activations of the input units and forward propagate that to the hidden layers and compute the activations of the hidden layers and then forward propagate to compute activation(s) of the output layer.

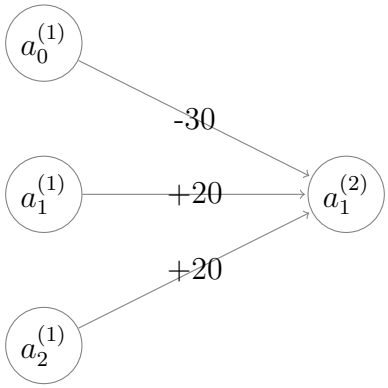
Some other neural network architectures:







Consider an example of a neural network that compute boolean operator AND with two inputs of values representing TRUE or FALSE $a_1^{(1)}, a_2^{(1)} \in \{0, 1\}$ and the output $y = a_1^{(1)} \wedge a_2^{(1)}$:



Therefore, we will have the hypothesis $h_\Theta(x) = g(-30 + 20a_1^{(1)} + 20a_2^{(1)})$ with g is the sigmoid function $g(z) = \frac{1}{1+e^{-z}}$. Consider the truth table for boolean operator AND:

$a_1^{(1)}$	$a_2^{(1)}$	$h_\Theta(a_1^{(1)})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

Consider a neural network that compute the boolean operator $(\neg a_1^{(1)}) \wedge (\neg a_2^{(1)})$ and the boolean operator $a_1^{(1)} \vee a_2^{(1)}$ respectively: