Week 4: 5/31/17

4. Neural Networks

Neural networks is a model inspired by how the brain works. It is widely used today in many applications: when your phone interprets and understand your voice commands, it is likely that a neural network is helping to understand your speech; when you cash a check, the machines that automatically read the digits also use neural networks.

Although Neural Networks were developed in the 1940s, much of the research stagnated. However, with the availability of parallel computing, Neural Networks gained a resurgence, and have since become rather popular as a learning algorithm. The motivation for this can be illustrated via the following example:

Example 4.1 : *Non-linear Classification*

Performing linear regression with a complex set of data with many features is very unwieldy. Say you wanted to create a hypothesis from three (3) features that included all the quadratic terms:

$$g(\theta_0 + \theta_1 x_1^2 + \theta_2 x_1 x_2 + \theta_3 x_1 x_3 + \theta_4 x_2^2 + \theta_5 x_2 x_3 \theta_6 x_3^2)$$

This gives us 6 features. If we were to have 100 features, using a quadratic polynomial, we would have 5050 new features.

We can approximate the growth of the number of new features we get with all quadratic terms with $\mathcal{O}(n^2/2)$. And if you wanted to include all cubic terms in your hypothesis, the features would grow asymptotically at $\mathcal{O}(n^3)$. These are very steep growths, so as the number of our features increase, the number of quadratic or cubic features increase very rapidly and becomes quickly impractical.

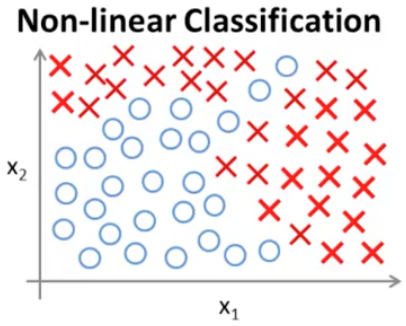


Figure 1. Classification example with 2 features.

4.1. Neurons & Model Representation.

Neural networks are limited imitations of how our own brains work. They've had a big recent resurgence because of advances in computer hardware. There is evidence that the brain uses only *"one learning algorithm"* for all its different functions. Scientists have tried cutting (in an animal brain) the connection between the ears and the auditory cortex and rewiring the optical nerve with the auditory cortex to find that the auditory cortex literally learns to see.

This principle is called "neuroplasticity" and has many examples and experimental evidence.

In this figure, we can think of the *dendrite* as the "input wires" and the *axon* as the "output wires." The *nucleus* is where the neuron performs any necessary computations. In general, these mechanisms form the basis for how the neuron communicates with others. In essence, the neuron is a computation unit that takes an input, performs some calculation, and then returns an output. All sensors and muscles in our bodies follow the same concept.

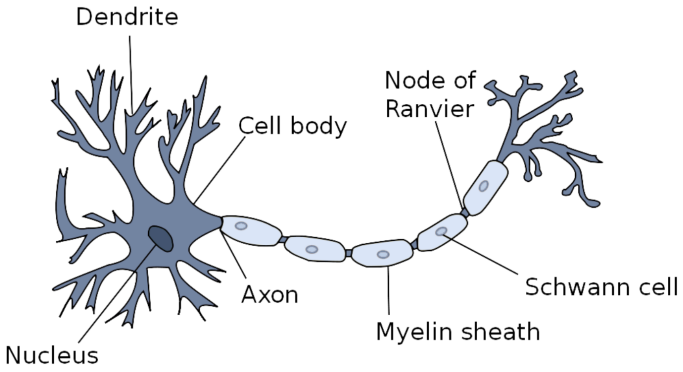


Figure 2. Illustration of a biological neuron with the components discussed in this text

In an *artificial* neural system, a single neuron is modeled as a single *logistic unit*:

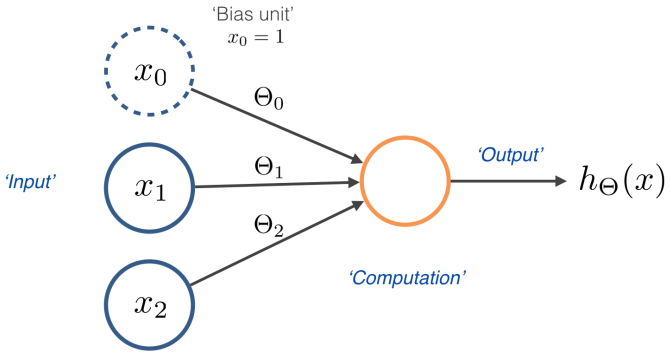


Figure 3. Single *neuron* or *'Logistic unit'*

Whenever a diagram like this is drawn, it represents the computation of the familiar *sigmoid* or *logistic* function:

$$h_\Theta(x) = \frac{1}{1 + e^{-\Theta^T x}}$$

In which the *input values*, x, and the *weights*, Θ, (or *parameters*) are defined as:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \quad \Theta = \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \end{bmatrix}$$

Definitions:

- ***Bias unit***: The zero-index in each layer of the neural network. Generally always 1, and may or may not be included in the diagram. However, this must be included in the calculations to ensure that the matrix math is possible.
- ***Activation Function***: Term for the non-linear hypothesis function, or computation.

A neural network, is a group of these neurons strung together:

Introducing a few more definitions:

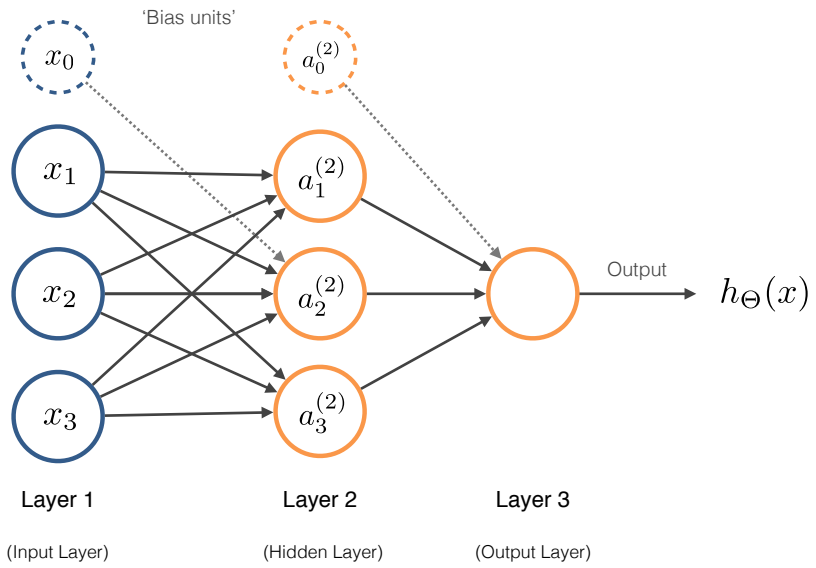


Figure 4. Neural Network

Definitions:

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

For 3 input units and 3 hidden units, the computations represented by this diagram are:

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

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

Where $\Theta^{(1)} \in \mathbb{R}^{3\times 4}$ matrix. In general, we can say:

- If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$. Or: *(next layer) × (this layer + 1)*

4.1.1. *Forward Propagation: Vectorized implementation.*

Now, the above set of equations may be rewritten in a vectorized notation. First,

If we rewrite the activation values (rows above) to:

$$a_1^{(2)} = g(z_1^{(2)})$$
$$a_2^{(2)} = g(z_2^{(2)})$$
$$a_3^{(2)} = g(z_3^{(2)})$$

In other words, for layer $j = 2$ and node k, the variable z will be:

$$z_k^{(2)} = \Theta_{k,0}^{(1)}x_0 + \Theta_{k,1}^{(1)}x_1 + \cdots + \Theta_{k,n}^{(1)}x_n$$

Representing x and z^j as a vector:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix}$$

Now, setting $x = a^{(1)}$ and $j = 2$, we can rewrite the equation as:

$$\begin{aligned} z^{(2)} &= \Theta^{(1)} x \\ &= \Theta^{(1)} a^{(1)} \end{aligned}$$

or, more generally:

$$z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$$

Now we can get a vector of our activation nodes for layer j as follows:

$$a^{(j)} = g(z^{(j)})$$

Where our function g can be applied element-wise to our vector $z^{(j)}$. Then, adding in the Bias unit, $a_0^{(2)} = 1$, from the second layer. Then, $\rightarrow a^{(2)} \in R^4$. The z vector is calculated as:

$$z^{(j+1)} = \Theta^{(j)} a^{(j)}$$

This last theta matrix $\Theta^{(j)}$ will have only one row which is multiplied by one column, $a^{(j)}$., so that your result is a single number. Finally, our result is:

$$h_\Theta(x) = a^{(j+1)} = g(z^{(j+1)})$$

This process is generally called *forward propagation.*

Note: The last layer (RHS) of Fig. 4 corresponds to *logistic regression.*

$$h_\Theta(x) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

The features are $\{a_1^{(2)}, a_2^{(2)}, a_3^{(2)}\}$, rather than $\{x_1, x_2, x_3\}$, which are *learned* from the parameters $\Theta^{(1)}$. Hence, the system learns it's own features before sending them to the logistic regression.

Note: ***Additional Architectures***
In Fig. 5 below, we have a multi-layer neural network.

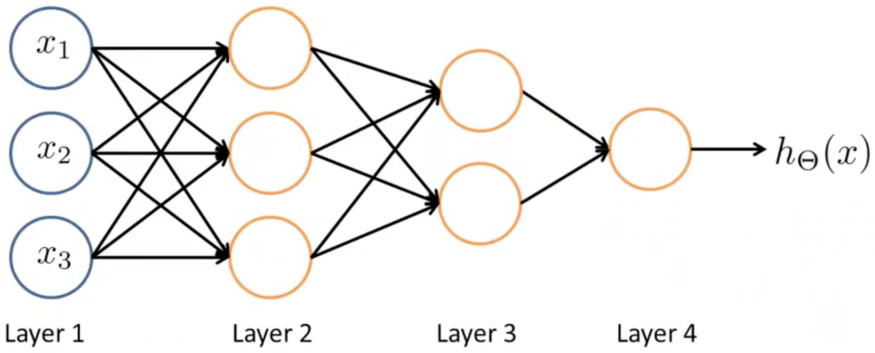


Figure 5. Other network architectures

4.2. Applications. In this section we take a look at several examples to understand why neural networks can be used to understand complex non-linear hypothesis.

Example 4.2 : ***Non-linear Classification***: XOR / XNOR

Assume x_1, x_2 are binary values (0 or 1), as seen in the simplified figure below:

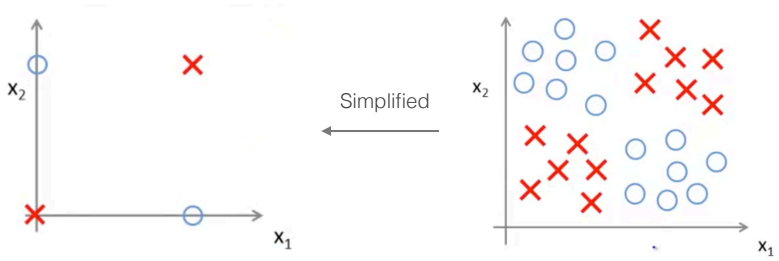


Figure 6. Simplified classification example

For this example, we can assume that this is a simplification of a more complex problem. This becomes a set of logical problem:

$$y = x_1 \ XOR \ x_2$$

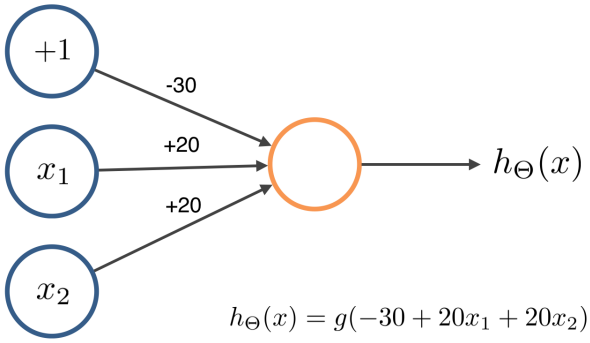
and the logical compliment:

$$\begin{aligned} y &= x_1 \ XNOR \ x_2 \\ &= NOT \ (x_1 \ XOR \ x_2) \end{aligned}$$

Now, we want to know if we can get a neural network to fit to this sort of training set.

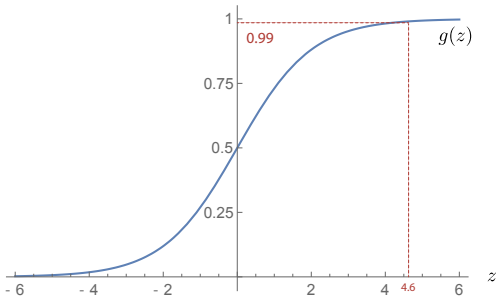
Example 4.3 : ***Non-linear Classification***: AND

Assume: $x_1, x_2 \in \{0, 1\}$, we want to model: $y = x_1 \ AND \ x_2$.



Assuming the same *logistic* function, we can create a *'truth table'*:

x_1	x_2	$h_\Theta(x)$
0	0	$g(-10) = 0$
0	1	$g(10) = 0$
1	0	$g(-10) = 0$
1	1	$g(10) = 1$

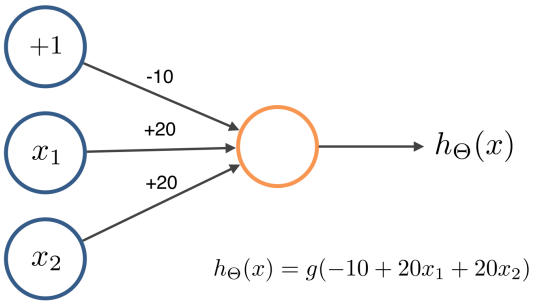


Then, from this table, we can see that $h_\Theta(x)$ is only positive for:

$$h_\Theta(x) \approx x_1 \; AND \; h_\Theta(x) \approx x_2$$

Example 4.4 : OR Function

Now we take a look at the logical OR function:

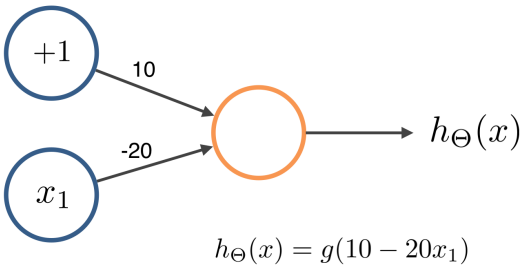


x_1	x_2	$h_\Theta(x)$
0	0	$g(-10) = 0$
0	1	$g(10) = 1$
1	0	$g(10) = 1$
1	1	$g(30) = 1$

Hence, we have the logical OR function.

Example 4.5 : Negation: NOT

Next, we consider a binary case, or the logical *'Negation'*, NOT. More specifically, we are computing $NOTx_1$.



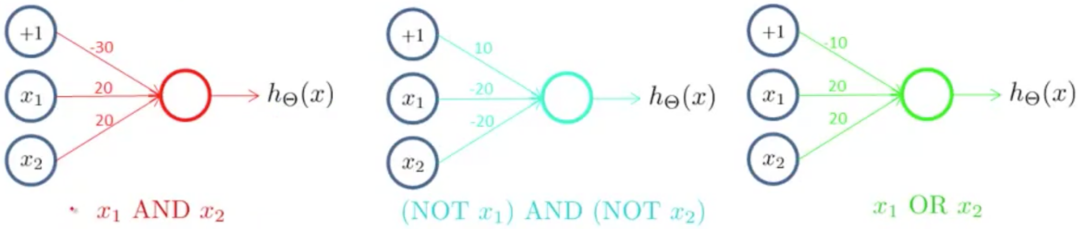
$h_\Theta(x) = g(10 - 20x_1)$

x_1	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

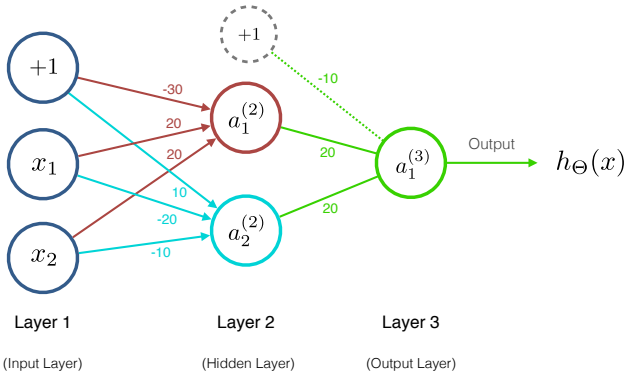
General Idea: The key for this type of negation problem is to place a fairly strong negative weight on the variable in which we are negtating.

Example 4.6 : *Putting it all together*: XNOR

Now, taking the three pieces that we have put together, we can now compute the XNOR function.



assembling this into one system:



x_1	x_2	$a_1^{(2}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

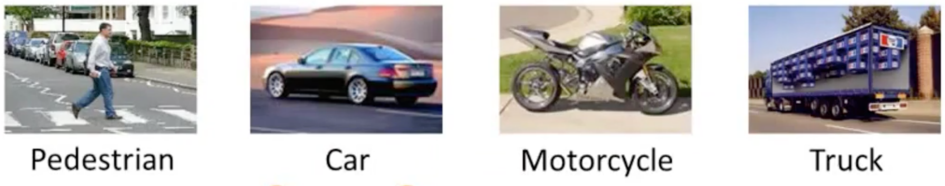
4.3. Multi-Class Classification.

Now, we may have more than one category to distinguish from. An example of this is the *handwriting recognition* problem, in which we had *ten* categories to choose from (0-9).

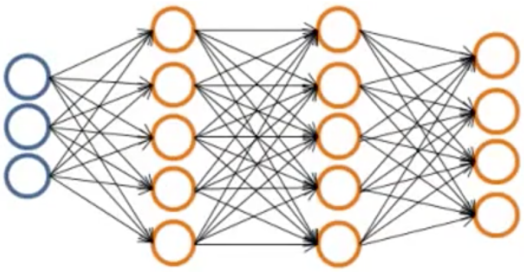
To classify data into multiple classes, we let our hypothesis function return a vector of values. Say we wanted to classify our data into one of four categories. We will use the following example to see how this classification is done. This algorithm takes as input an image and classifies it accordingly:

Example 4.7 : *Multiple output units*: One-vs-all

Lets say we have a computer vision problem, in which we would like to identify four categories of objects:



then, our output will have four objects as well:



$$h_\Theta(x) \in \mathbb{R}^4$$

in which our output is expected to be:

$$\text{Want } h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad \text{etc.}$$

when pedestrian — when car — when motorcycle

for each object. For this problem, our training set will look as follows:

$$\text{Training set: } (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$$

$$y^{(i)} \text{ one of } \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

pedestrian car motorcycle truck

We can define our set of resulting classes as y:

$$y^{(i)} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Each $y^{(i)}$ represents a different image corresponding to either a car, pedestrian, truck, or motorcycle. The inner layers, each provide us with some new information which leads to our final hypothesis function. The setup looks like:

$$\begin{bmatrix}x_0\\x_1\\x_2\\\cdots\\x_n\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(2)}\\a_1^{(2)}\\a_2^{(2)}\\\cdots\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(3)}\\a_1^{(3)}\\a_2^{(3)}\\\cdots\end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix}h_\Theta(x)_1\\h_\Theta(x)_2\\h_\Theta(x)_3\\h_\Theta(x)_4\end{bmatrix}$$

Our resulting hypothesis for one set of inputs may look like:

$$h_\Theta(x) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

In which case our resulting class is the third one down, or $h_\Theta(x)_3$, which represents the motorcycle.