# Neural Network Discussion

> "Machine learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving confidence intervals around these functions."

>

> -- Ian Goodfellow, et al. [^51]

[^51]:Ian Goodfellow, Yoshua Bengio, Aaron Courville, 'Deep Learning,' MIT Press, 2016, http://www.deeplearningbook.org

## Introduction

If we discuss Neural Networks, we should first consider the system we hope to emulate. Let us start with a simple count of neuronal cells in various organisms along the earth's phylogenetic or evolutionary tree. We might get a better idea of the type of "computing power" these living creatures possess. See table 5.1.

#### Table 5.1: Organisms Vs Number of Neurons In Each ([Wikipedia](https://en.wikipedia.org/wiki/List\_of\_animals\_by\_number\_of\_neurons)) {-}

| Organism | Common Name | Approximate Number of Neurons |

| :--------------------- | ----------: | ----------------------------: |

| C. elegans | roundworm | 302 |

| Chrysaora fuscescens | jellyfish | 5,600 |

| Apis linnaeus | honey bee | 960,000 |

| Mus musculus | mouse | 71,000,000 |

| Felis silvestris | cat | 760,000,000 |

| Canis lupus familiaris | dog | 2,300,000,000 |

| Homo sapien sapien | humans | 100,000,000,000 |

This table portrays a high-level overview of the computing power of neuronal clusters and brains produced throughout evolution. However, there is one missing number worth noting. The table above does not describe the connectivity between neurons. The inter-connectivity of neurons varies greatly from lower to higher organisms. For example, some elementary animals have "where some neurons can have four to eight separate branches" [^52], per nerve cell. While human neurons may have approximately 10^4 inter-connected synaptic junctions per neuron, thus resulting in a total of approximately 600 trillion synapses per human brain.[^53]

[^52]:https://www.wormatlas.org/hermaphrodite/nervous/Neuroframeset.html

[^53]:Shepherd, G. M. (2004), The synaptic organization of the brain (5th ed.), Oxford University Press, New York.

Although neurons have differing morphologies, neurons in the human brain are extremely diverse. Indeed, size and shape may not be the definitive way of classifying neurons but instead by what neurotransmitters the cells secrete. "Neurotransmitters can be classified as either excitatory or inhibitory." [^58] Currently the [NeuroPep (version 1.0, 2014-11-26)](http://isyslab.info/NeuroPep/home.jsp) database "holds 5949 non-redundant neuropeptide entries originating from 493 organisms belonging to 65 neuropeptide families."[^59]

\newpage

[^58]:https://www.kenhub.com/en/library/anatomy/neurotransmitters

[^59]:http://isyslab.info/NeuroPep/home.jsp

[^54]

[^54]:https://www.howstuffworks.com/

Given an order of operation via:

Dendrite(s) $\Longrightarrow$ Cell body $\Longrightarrow$ Fibrous Axon $\Longrightarrow$ Synaptic Junction or Synaptic Gap $\Longrightarrow$ Dendrite(s) ... Ad infinitum.

However, nature is more subtle and intricate than to have neurons in a series, only blinking on and off, firing or not. Neural networks are often programmed to classify dangerous road objects, as is the case of Tesla cars. The goal of a Tesla auto-piloted car is to use all available sensors to correctly classify all the conceivable circumstances on the road. On the road, a Tesla automobile uses dozens of senors which the computer needs to evaluate and weigh the values of all these sensors to formulate a 'decision.' The altitude of the auto, derived from the GPS, may weigh less heavily than the speed of the vehicle or Lidar estimates on how close objects are. However, that goal can be thwarted when an artificial intelligence system decides a truck is a sign and does not apply the brakes.[^55]

```{r fig.width=3, fig.height=1.2 ,fig.cap="Goal of a Tesla Neural Networks is to generate the correct repsonses for its environment.", fig.align="center", echo=FALSE}

img <- readPNG("../00-data/10-images/nn.black.box.png")

grid.raster(img)

```

[^55]:https://arstechnica.com/cars/2019/05/feds-autopilot-was-active-during-deadly-march-tesla-crash/

\newpage

## The One Neuron System

If we investigate a one neuron system, \*our\* neuron could be diagrammed in four sections.

[^56]

```{r fig.width=4, fig.cap="One Neuron Schema", fig.asp="125%", fig.align="center", echo=FALSE}

img <- readPNG("../00-data/10-images/one.neuron.schema.2.png")

grid.raster(img)

```

[^56]:Tom Mitchell, Machine Learning, McGraw-Hill, 1997, ISBN: 0070428077

If we investigate one neuron for a moment, we find two separate mathematical functions incorporated into a single nerve cell.

### Summation Function

The first portion is a summation function. It receives the real number values from $x\_1$ to $x\_n$, all the branches of the dendritic trees, and multiplies them by a set of weights ($w\_1$ to $w\_n$). An analogy I prefer is of small or large rivers joining. The current moves through the branches giving a total signal or current of sodium ions. Interestingly the summation in each neuron, while dealing with the vectors of inputs and weights, is carrying out the [dot product](https://www.khanacademy.org/math/linear-algebra/vectors-and-spaces/dot-cross-products/v/vector-dot-product-and-vector-length) of these vectors, such that;

\begin{equation}

\large Y ~=~ X^T \cdot W - Bias ~~\equiv~~ \sum\_{i=1}^n x\_i w\_i - T

\end{equation}

### Activation Functions

The second function is called an Activation Function. These inputs are multiplied by a set of corresponding unique weights from $w\_1$ to $w\_n$. Giving in the simplest case;

Once the Summation Function yields a value, its result is sent to the \*Activation Function\* or \*Threshold Function\*.

Initially, the neural networks used the Heaviside-Threshold Function, as shown in figure 5.4, the 'One Neuron System.' The benefits of step functions were their simplicity and high signal to noise ratio. While the detriments were, it is a discontinuous function, therefore not able to be differentiated and a mathematical problem.

Let us take into account the product, $x\_0 \cdot w\_0$. If we assign $x\_0 = T$ and $w\_0 = -1$ this simply becomes a bias. This bias allows us the ability to shift our Activation Function and its inflection point in the positive or negative x-direction.

\begin{equation}

\large {Z}^{(1)} = f \left( \sum\_{i=1}^n x\_i w\_i - T\right) = \{0, 1\}

\end{equation}

The function displayed in figure #5.4 is a step function. However, this step function has a problem; namely, it is a discontinuous function and, therefore, not differentiable. This fact is essential soon.

Several functions may be used in place of the step function. One is the hyperbolic tangent function and the second is a sigmoidal function;

\begin{equation}

\large Z^{(2)} ~=~ tanh(x) = \frac{1 - e^{-{\alpha}}}{1 + e^{-{\alpha}}} ~~~:~~~ \large where ~~~ \large \alpha = \sum\_{i=1}^n x\_i w\_i - T

\end{equation}

\begin{equation}

\large Z^{(3)} ~=~ sigmoid(x) ~=~ \frac{1}{1 + e^{-{\alpha}}}

\end{equation}

\begin{equation}

\large Z^{(4)} ~=~ Hard ~ Tanh (x) ~=~ \large \left\{ \begin{array}{rcl} 1 & x > 1 \\ x & -1 \leq x \leq 1 \\ -1 & x < -1 \end{array}\right.

\end{equation}

Several alternative functions are useful for various reasons. The most common of which are Softmax and reLU functions.

Rectified Linear Activation Unit, (ReLU):

\begin{equation}

\large Z^{(5)} ~=~ \large ReLU ~= \begin{cases} x \geqq 0 ~~~~y = x\\ x < 0 ~~~~y = 0 \end{cases}

\end{equation}

### Binary Output Or Probability

In the case of real neurons, the output is off or on, zero or one. However, in the case of our electronic model, it is advantageous to calculate a probability for greater interpretability.

>The Softmax function may appear like the Sigmoid function from above, but it differs in major ways.[^57]

>

>\* The softmax activation function returns the probability distribution over mutually exclusive output classes.

>\* The calculated probabilities [are] in the range of 0 to 1.

>\* The sum of all the probabilities is 1.

[^57]:Josh Patterson, Adam Gibson, Deep Learning; A Practitioner's Approach, 2017, O'Reilly

Typically the Softmax Function is used in binary or multiple classification logistic regression models and in building the final output layer of neural networks.

\begin{equation}

\large Z^{(6)} ~=~ Softmax(x) = \frac {e^{\alpha\_i}}{\sum\_{i=1}^n e^{\alpha\_i}}

\end{equation}

---

The benefit of these activation functions is that they are now differentiable. This fact becomes very important for \*Back-Propagation\*, which is discussed later.

\newpage

## The Two Neuron System

Building up in complexity, let us could consider our first Neural Network by using \*only\* two neurons. In two neuron systems, let us first generalize a bit more by adding that $X$ is an array of all the inputs as is $W\_1$ and $W\_2$ is also an array of weights for each neuron. See figure #5.5.

```{r fig.width=4, fig.cap="A Two Neuron System", fig.align = "center", echo=FALSE}

img <- readPNG("../00-data/10-images/two.neuron.system.png")

grid.raster(img)

```

### Feed-Forward of a Two Neuron Network (to find $Z$)

In our two neuron network, we can now write out the mathematics for each step as it progresses in a "Forward" (left to right) direction.

Step #1: To move from $X$ to $P\_1$:

\begin{equation}

f^1( \overrightarrow{x}, \overrightarrow{w}) \equiv~~ P\_1 = \left( X^T \cdot W\_1 - T \right)

\end{equation}

Step #2: $P\_1$ feeds forward to $Y$:

\begin{equation}

f^2(P\_1) ~~\equiv~ Y = \left( \frac{1}{1 + e^{- \alpha}} \right) ~~:~~ where ~~~ \alpha = P\_1

\end{equation}

Step #3: $Y$ feeds forward to $P\_2$:

\begin{equation}

f^3(\overrightarrow{y}, \overrightarrow{w}) ~~\equiv~ P\_2 = \left( Y^T \cdot W\_2 - T \right)

\end{equation}

Step #4: $P\_2$ feeds forward to $Z$:

\begin{equation}

f^4(P\_2) ~~\equiv~ Z = \left( \frac{1}{1 + e^{- \large \alpha}} \right) ~~~:~~~ where ~~ \alpha = P\_2

\end{equation}

Our complicated function is simply a matter of chaining one result used for the next step.

\begin{equation}

Z ~=~ f^4 \left( f^3 \left( f^2 \left( f^1 \left( X, W \right) \right) \right) \right)

\end{equation}

In our \*\*Feed-Forward Model\*\*, we can now take the values from any numerical system and produce zeros or ones. Remember, in this set of experiments, we are using the concentrations of the amino acids to provide a categorical or binary output, belongs to Myoglobin protein family, or does not.

### Back-propagation (to find $X$)

Now that we have learned to calculate the output of our neurons using the Feed-Forward process, what if our answer is incorrect.

The process for determining the weights is known as Back-Propagation. Back-Propagation is crucial to understanding and tuning a neural network.

$x\_1$ to $x\_n$, all the branches of the dendritic trees, and multiplies them by a set of weights ($w\_1$ to $w\_n$)

In essence, we are asking \*'How much does $\mathbf{Z}$ wiggle (read: change) when $\mathbf{X, W}$ is wiggled (read: changed)'.\*

{---This now gives the building blocks and the formulas necessary to calculate $X, W$ via an iterative process that can be run until we reach a value that we desire with given criteria.

Performance function equals negative the max magnitude of the vector of D minus the vector of Z magnitude squared. The performance function is no more than a slightly modified mean squared error (MSE).}

\begin{equation}

\mathbf{Performance} ~~=~~ c \cdot (d - z)^2

\end{equation}

\begin{equation}

\frac{d ~ Perf}{d x} ~~=~~ \frac{d \left \{ f^4 \left( f^3 \left( f^2 \left( f^1 \left( X, W \right) \right) \right) \right) \right \}}{dx}

\end{equation}

```{r fig.height=1, fig.cap="A Two Neuron System", fig.align = "center", echo=FALSE}

img <- readPNG("../00-data/10-images/two.neuron.system.noframe.png")

grid.raster(img)

```

\begin{equation}

Neuron ~2 \Rightarrow 1 :~~~~ \frac{\delta ~ Perf}{\delta w\_1} ~=~ \frac{\delta Perf}{\delta z} \cdot \frac{\delta z}{\delta P\_2} \cdot \frac{\delta P\_2}{\delta y} \cdot \frac{\delta y}{\delta P\_1} \cdot \frac{\delta P\_2}{\delta w\_1}

\end{equation}

\begin{equation}

\frac{\delta Perf}{\delta z} ~~=~~ \frac{ \left\{ \frac{1}{2} \large \| \overrightarrow{d} - \overrightarrow{z} \|^2 \right\}} {\delta z} ~~=~~ \mathbf{\overrightarrow{d} - \overrightarrow{z}}

\end{equation}

\begin{equation}

if ~~ we ~~ substitute ~~P\_2=\alpha ~~:~~\frac{\delta z}{\delta P\_2} ~~=~~ \frac{\delta~ ((1 + e^{-\alpha})^{-1})}{\delta \alpha} ~~=~~ e^{-\alpha} \cdot (1 + e^{-\alpha})^{-2}

\end{equation}

However, if we rearrange the expression:

\begin{equation}

\frac{ e^{-\alpha} }{ (1 + e^{-\alpha})^{-2} } ~~=~~ \frac{e^{-\alpha}}{1 + e^{-\alpha}} \cdot \frac{1}{1 + e^{-\alpha}}

\end{equation}

Then \*\*both\*\* add and subtract 1 to the right-hand side:

\begin{equation}

= ~~ \frac{ (1+ e^{-\alpha}) -1 }{1 + e^{-\alpha}} \cdot \frac{1}{1 + e^{-\alpha}}

\end{equation}

Rearrange to find:

\begin{equation}

= ~~ \left( \frac{ 1+ e^{-\alpha} }{1 + e^{-\alpha}} ~-~ \frac{ 1 }{1 + e^{-\alpha}} \right) \left( \frac{1}{1 + e^{-\alpha}} \right) ~~=~~ \left(1- \frac{1}{1 + e^{-\alpha}} \right) \left( \frac{1}{1 + e^{-\alpha}} \right)

\end{equation}

Therefore:

\begin{equation}

\frac{\delta z}{\delta \alpha} ~~=~~ \frac{\delta~ ((1 + e^{-\alpha})^{-1})}{\delta \alpha} ~~=~~ \left(1- \frac{1}{1 + e^{-\alpha}} \right) \left( \frac{1}{1 + e^{-\alpha}} \right)

\end{equation}