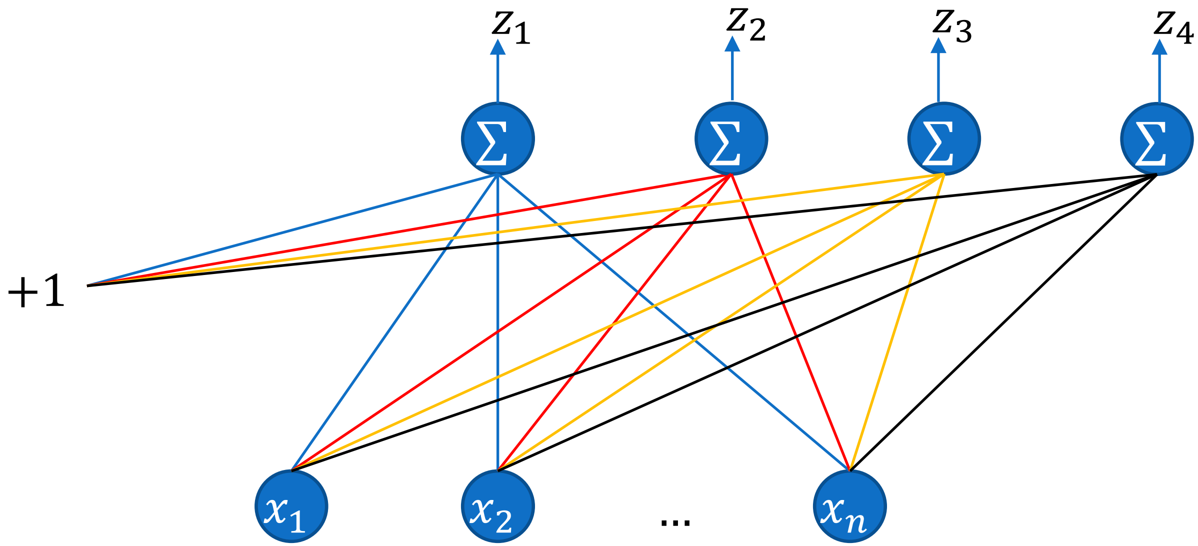
Review

SOFTMAX

Softmax: A Visual Perspective

Compute Error

$$S(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$



Loss Function

Actual labels are one-hot-encoded. $\rightarrow$ $y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \hat{y} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix}$

$$\boxed{Loss(y, \hat{y}) = -\sum_{j=1}^{C} y_j log \hat{y}_j}$$

$$Loss(y, \hat{y}) = -(0.\, log\hat{y}_1 + 1.\, log\hat{y}_2 + 0.\, log\hat{y}_3 + 0.\, log\hat{y}_4)$$

$$Loss(y, \hat{y}) = -(1.\, log\hat{y}_2) = -log\hat{y}_2$$

$$\boxed{J(y, \hat{y}) = \frac{1}{m}\sum_{i=1}^{m} Loss(y, \hat{y})}$$

Use gradient descent to adjust weights once you have cost.

One Hot Encoding (aka categorical encoding)

x_1	x_2	y
5	9	0
6	8	0
1	2	1
11	12	2

y *One Hot Encoded*
$[1, 0, 0]$
$[1, 0, 0]$
$[0, 1, 0]$
$[0, 0, 1]$

$\hat{y}$ *After Softmax*
$[0.9, 0.1, 0]$
$[0.8, 0.2, 0]$
$[0.1, 0.75, 0.15]$
$[0, 0.05, 0.95]$

$$\boxed{Loss(y, \hat{y}) = -\sum_{j=1}^{C} y_j log \hat{y}_j}$$

$$\boxed{J(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^{m} Loss(y, \hat{y})}$$

Project

- Idea Discussion!
- Dataset Creation
- Model Training
- Model Evaluation
  - Kaggle

Perceptron

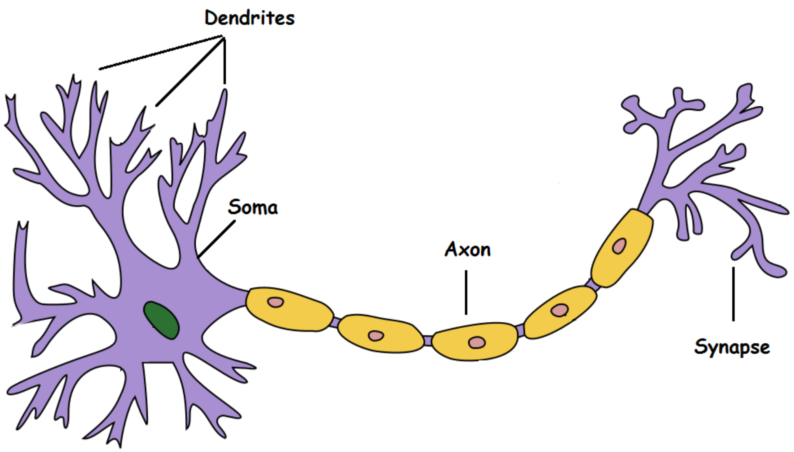
References

- Akshay L Chandra – medium.com
  - McCulloch-Pitts Neuron — Mankind's First Mathematical Model Of A Biological Neuron: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1
  - Perceptron: The Artificial Neuron (An Essential Upgrade To The McCulloch-Pitts Neuron): https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d
  - Perceptron Learning Algorithm: A Graphical Explanation Of Why It Works: https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975
- Prof. Mitesh M. Khapra (https://www.cse.iitm.ac.in/~miteshk/) on NPTEL's (http://nptel.ac.in/) Deep Learning course (https://onlinecourses.nptel.ac.in/noc18_cs41/preview)
- Machine Learning for Intelligent Systems, Kilian Weinberger, Cornell, Lectures 3-6, https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html
- Perceptrons. An Introduction to Computational Geometry. Marvin Minsky and Seymour Papert. M.I.T. Press, Cambridge, Mass., 1969. https://science.sciencemag.org/content/165/3895/780

McCulloch-Pitts Neuron

- The fundamental unit of ANNs – An Artificial Neuron
- 1943 by McCulloch and Pitts: Mimicking the functionality of a biological neuron

Dendrites receive signals from other neurons
Soma processes the information
Axons transmit the output
Synapses are the connections to other neurons

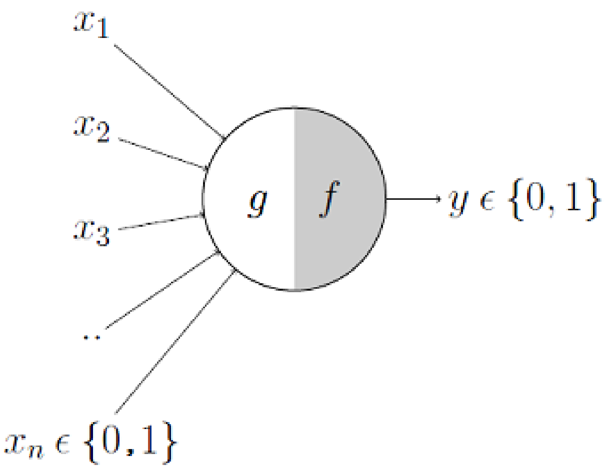


- About 86 billion of these in our brains on average!
- Each neuron gets activated/fired when its firing criteria is met
  - Based on the aggregation of signals from the inputs

Credit: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

McCulloch-Pitts Neuron

- The first computational model of a neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.



$$g(x_1, x_2, \dots, x_n) = g(X) = \sum_{i=1}^{n} x_i$$

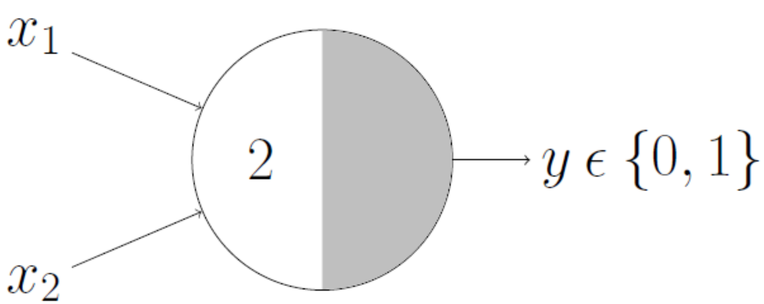
$$y = f(g(X)) = 1 \quad if\ g(X) \geq \theta$$
$$= 0 \quad if\ g(X) < \theta$$

- $\boldsymbol{g}$ aggregates inputs, $\boldsymbol{f}$ makes decisions based on θ
- $\boldsymbol{\theta}$ is a hand-coded threshold
- All x_i are binary inputs and y_i is a binary output

Credit: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

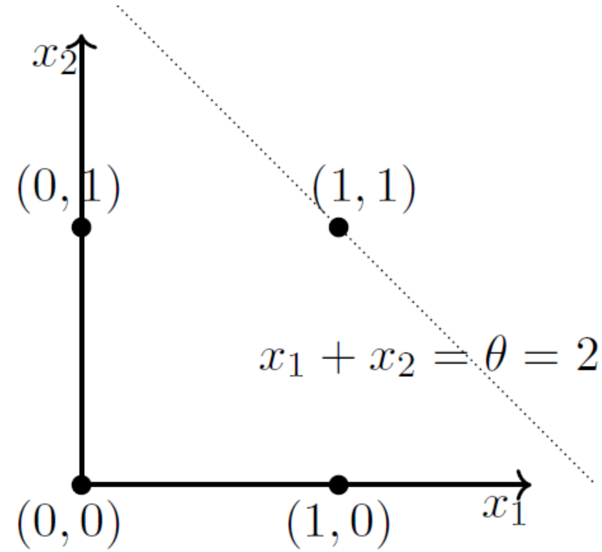
Boolean Functions

- AND Function



AND function

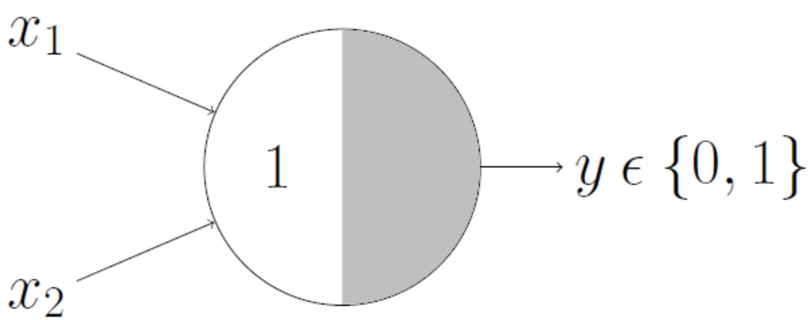
$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$



Credit: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

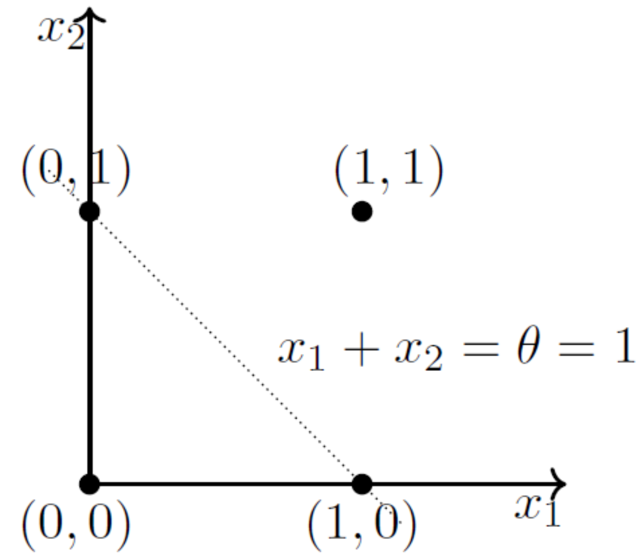
Boolean Functions

- OR Function



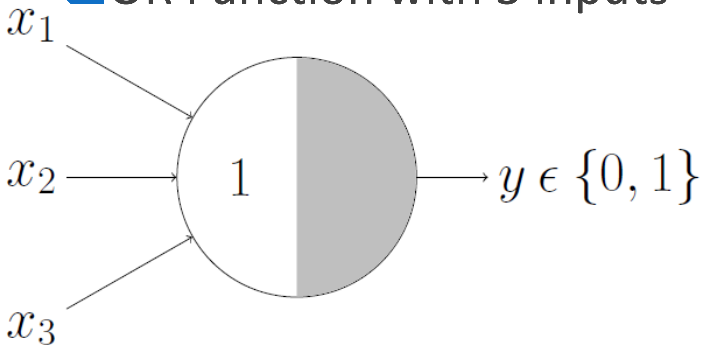
OR function

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$$



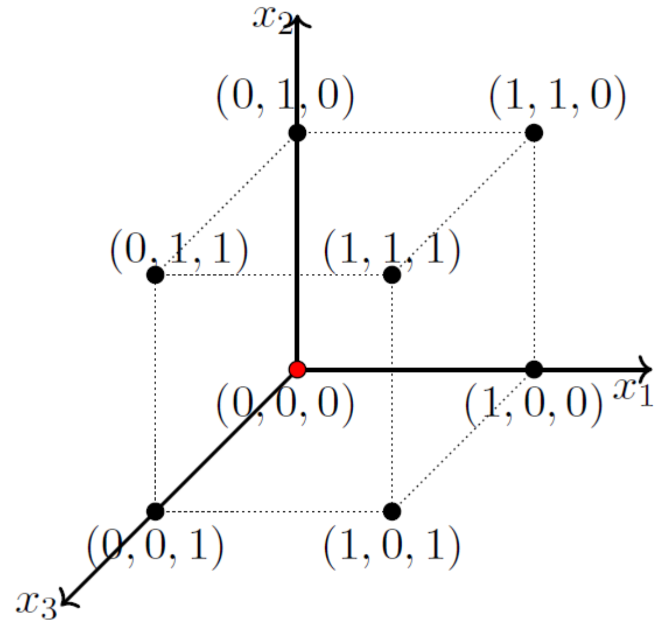
Boolean Functions

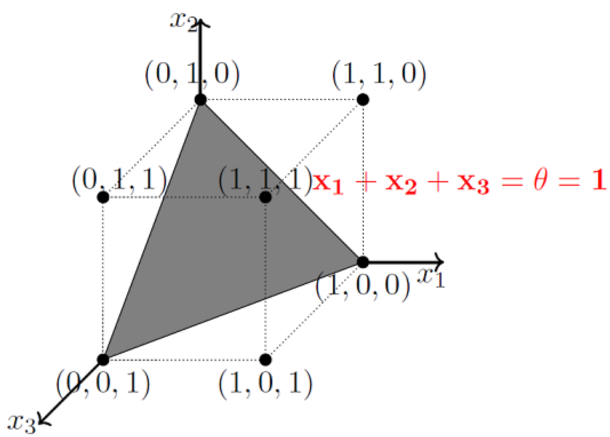
- OR Function with 3 inputs



OR function

$$x_1 + x_2 + x_3 = \sum_{i=1}^{3} x_i \geq 1$$





The decision boundary would be a plane!

Limitations of M-P Neuron

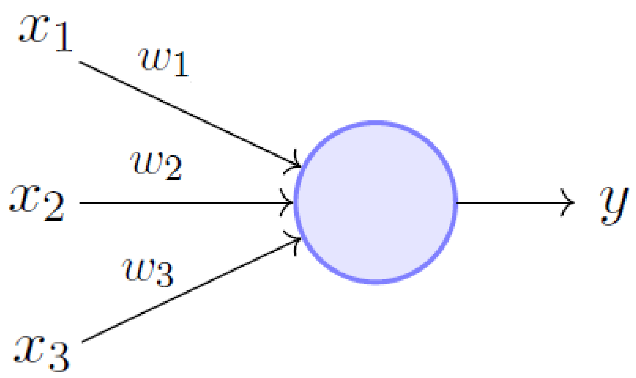
- Limited to Boolean inputs
- Hand-coded thresholds
- All inputs are equally important
  - Are all inputs born equal?
  - Is number of legs as important for a **human vs cat** classifier as number of ears?

- What about functions that are not linearly separable? E.g., the XOR function?

- In 1958, Fran Rosenblatt, an American Psychologist, proposed the perceptron model
  - Added **weights and thresholds that could be learned.**
  - Allowed **real-numbered inputs.**

Credit: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

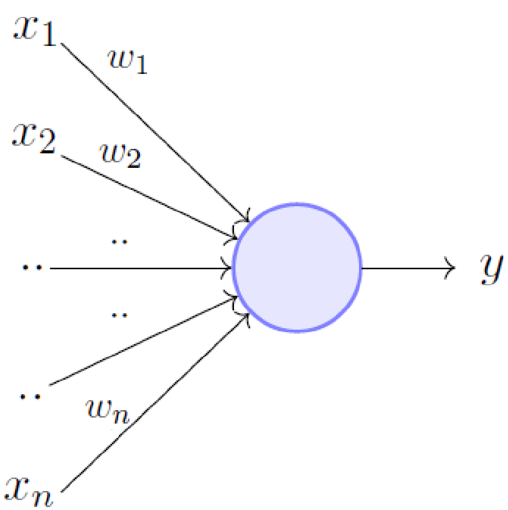
The Perceptron



Perceptron Model (Minsky-Papert in 1969)

- Numerical weights associated with inputs
- No longer limited to Boolean inputs
- A learning mechanism to train the weights and threshold.

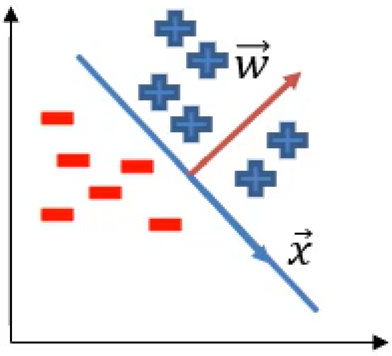
The Perceptron

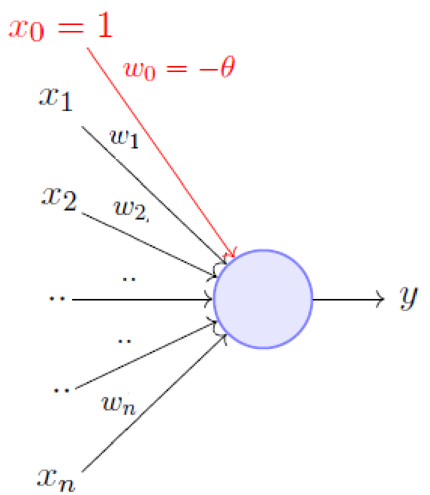


$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$





A more accepted convention,

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$
$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

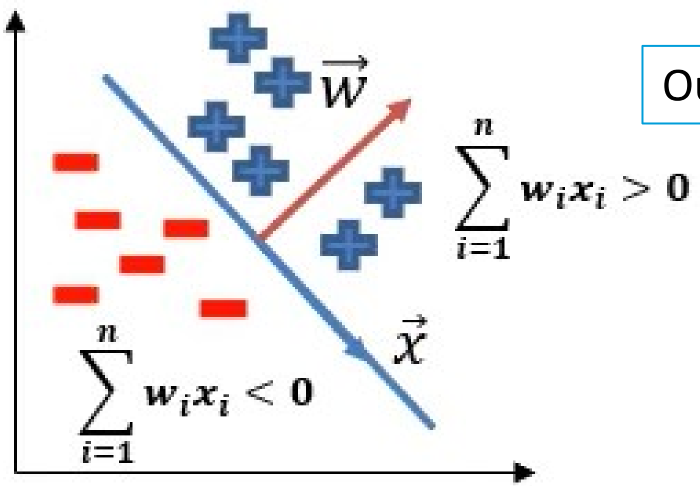
$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$

Credit: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

The Perceptron

- If we set $\theta = 0$, then the decision is simply: $h(x_i) = \text{sgn}(w^T x_i + \theta)$
- After absorbing θ, this becomes: $h(x_i) = \text{sgn}(w^T x_i)$

Output is either +ve or -ve



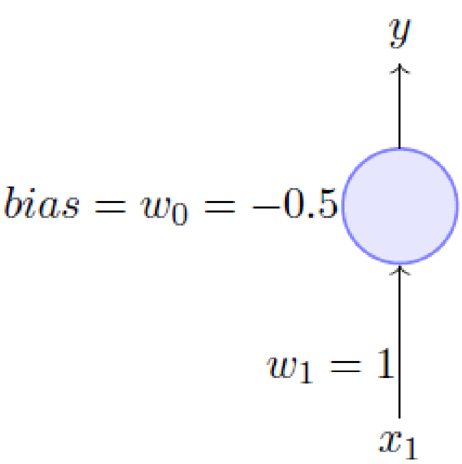
- This also simplifies defining correct/incorrect classification:

$$y_i(w^T x_i) > 0 \iff x_i \text{ is classified correctly}$$
$$y_i > 0 \text{ and } (w^T x_i) > 0 \text{ OR } y_i < 0 \text{ and } (w^T x_i) < 0$$

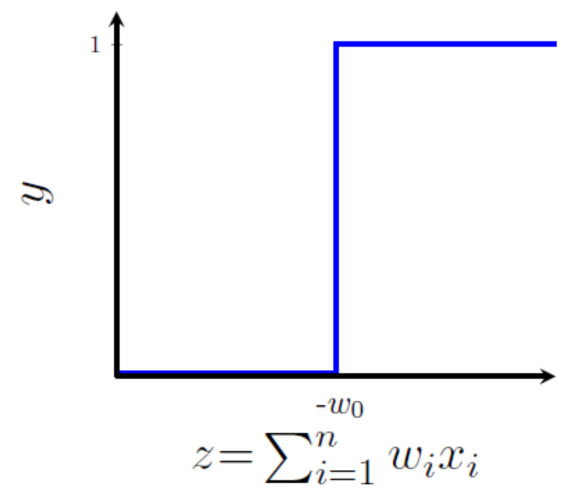
Credit: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

Connecting the dots

- The perceptron employs very harsh thresholds!
- E.g., if the threshold is 0.5, input = 0.49 to the thresholding function would yield a negative and input=0.51 would yield a positive output



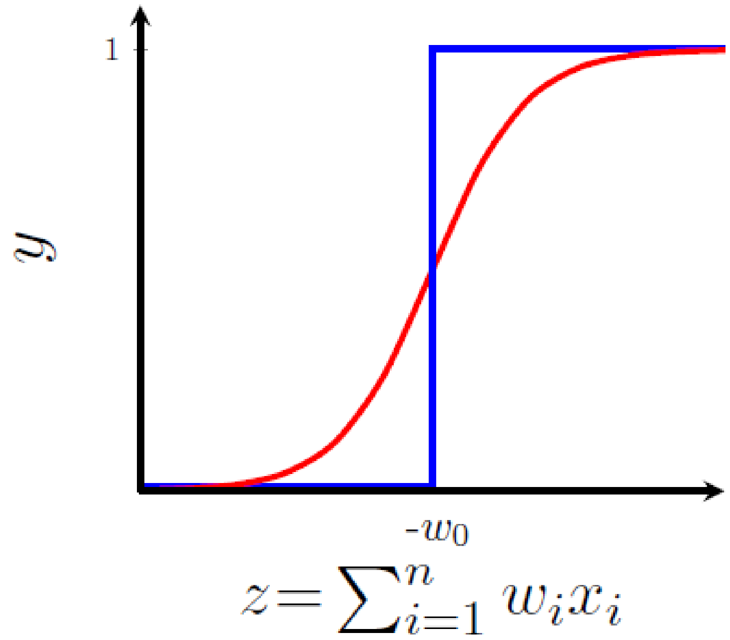
$$
\begin{aligned}
y &= 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0 \\
&= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0
\end{aligned}
$$



Connecting the dots – Using other activation functions

- We can use a smoother function like sigmoid!

$$y = \frac{1}{1 + e^{-(w_0 + \sum_{i=1}^{n} w_i x_i)}}$$



- The output is no longer binary but a real value between 0 and 1, which can be interpreted as a probability
- So instead of yes/no decision, we get the probability of yes.
- The output is **smooth, continuous,** and **differentiable.**

The Perceptron

- A perceptron separates the input space into two halves, positive and negative.
- All the inputs that produce ***true*** lie on one side (positive half) and all the inputs that produce ***false*** lie on the other side (negative half space)

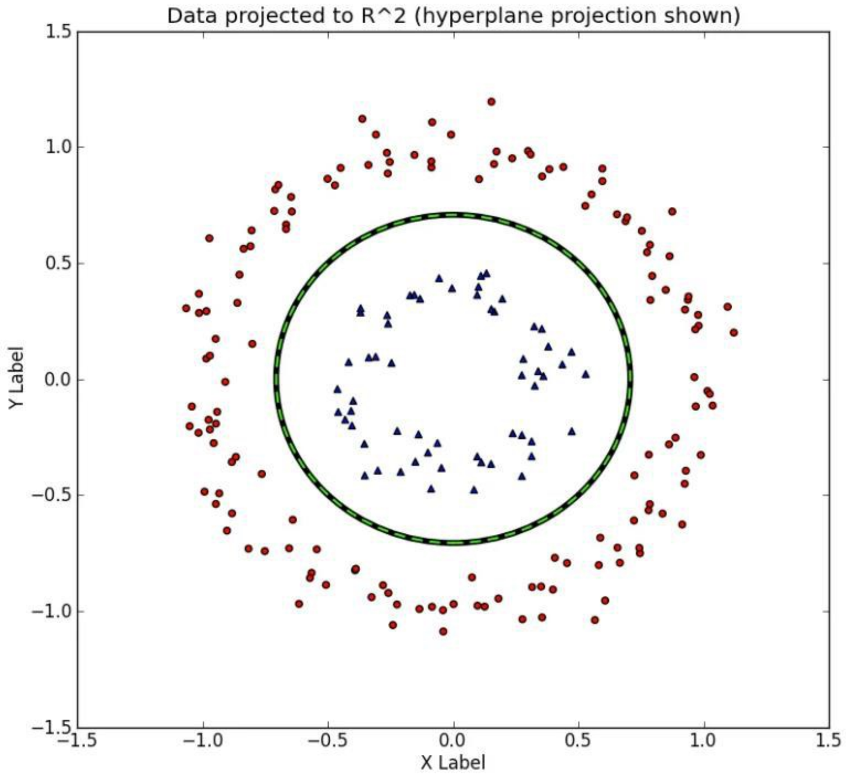
- **A single Perceptron can only be used to implement linearly separable functions**
  - Just like M-P Neuron

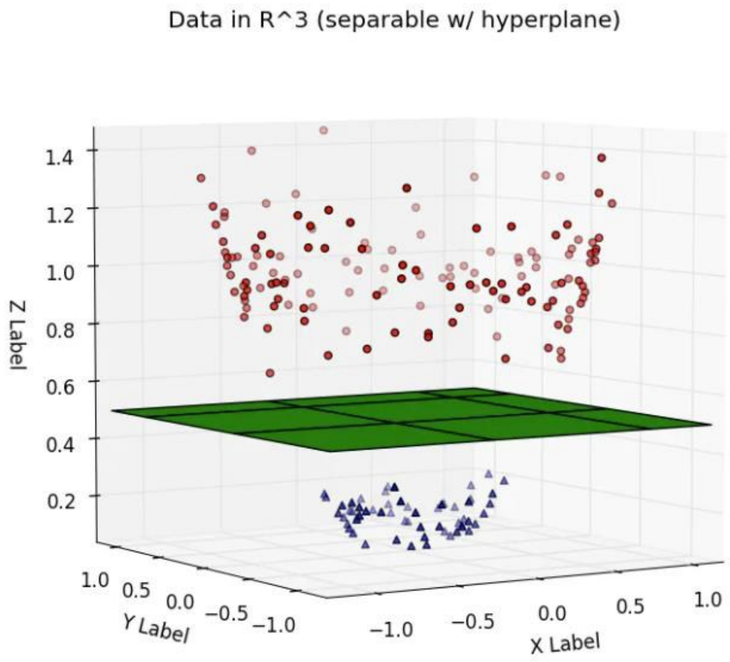
- **How Perceptron is different than M-P Neuron?**
  - The inputs can be assigned different importance
  - The weights and the thresholds can be learned.
  - The inputs can be real values

How to make linearly separable decision boundary? What should be changed in Perceptron?

Credit: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

One way: Adding Dimensions to Achieve Linear Separability





Second Way: Use Hidden Layers

Credit: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f)

Book Reading

- Jurafsky – Chapter 7
- Tom Mitchel – Chapter 4