


# Intro to Neural Networks


**BA865 – Mohannad Elhamod**

# Basic Linear Algebra

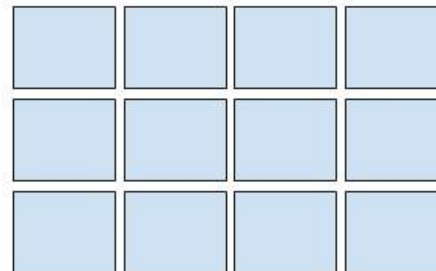
# What is a Tensor?

- A tensor is a matrix of rank 3 or higher.
- Examples?
- You can think about it as “groups of vectors”

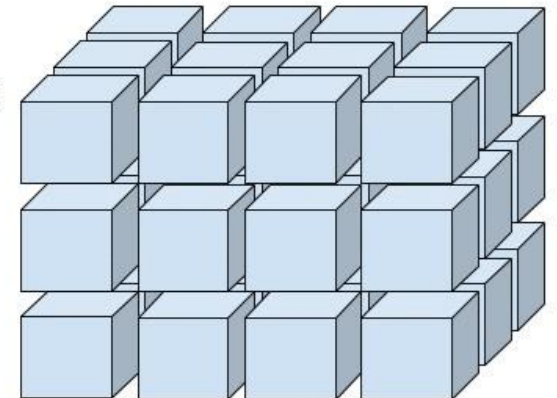
Rank 0:   
(scalar)

Rank 1:   
(vector)

Rank 2: (matrix)



Rank 3:



© Gordon Burtch, 2022

# Addition

- Shape of the Two Tensors Needs to Conform.
- Sum Element-wise
  - Replicate B until it matches A's dimensions, then perform element-wise addition.

`np.arange(3) + 5`

0	1	2
---	---	---

+

5	5	5
---	---	---

=

5	6	7
---	---	---

`np.ones((3, 3)) + np.arange(3)`

1	1	1
1	1	1
1	1	1

+

0	1	2
0	1	2
0	1	2

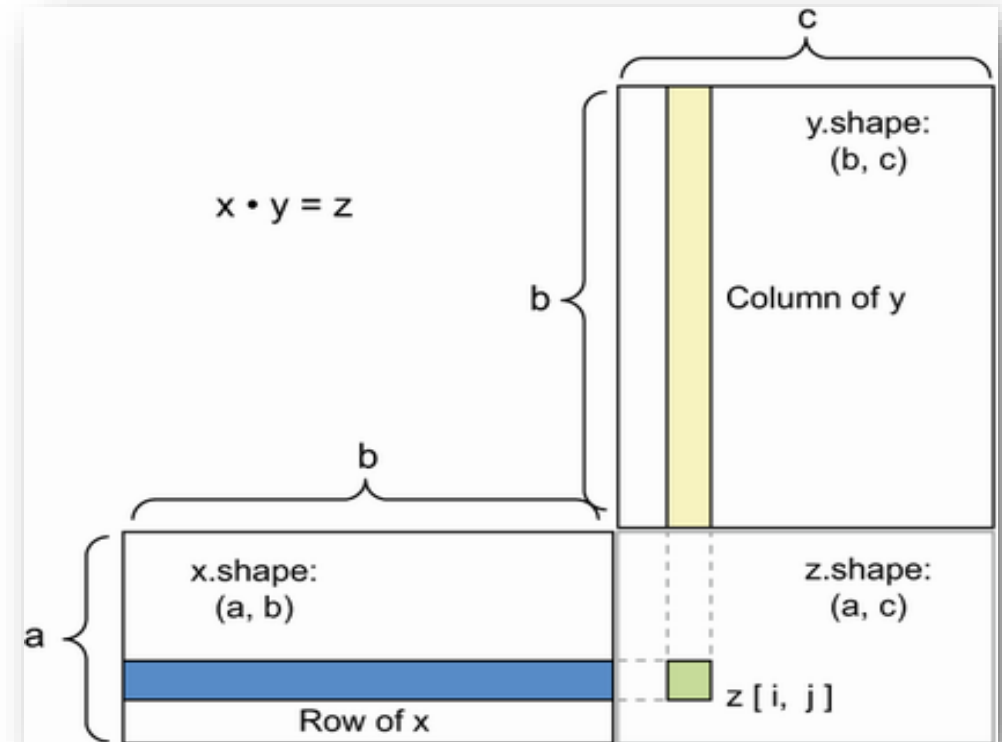
=

1	2	3
1	2	3
1	2	3

© Gordon Burtch, 2023

# Multiplication (Dot Product)

- $x.shape[-1]$  should equal  $y.shape[0]$
- This is different from element-wise multiplication!



© Gordon Burtch, 2023

# Linear Transformation

- Matrix multiplication is a “linear transformation”.
- What does that mean? [Demo](#)
  - The outputs are a linear combination of the inputs.
  - A line remains a line and relative positions are preserved (No warping!).
  - Reflection, scaling, rotation, and shear.
  - Not translation!

# Enter Non-Linearities

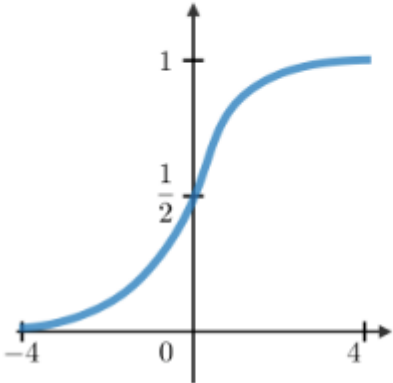
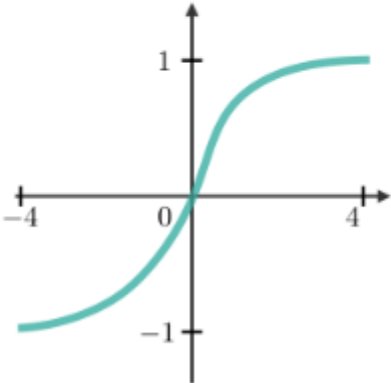
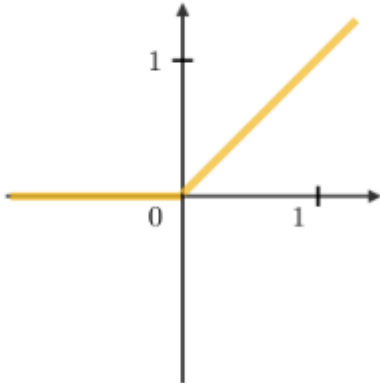
# What is a Non-Linearity?

- A non-linearity will lead to warping and “mangling” of the space.
- [Demo](#)
- Examples:
  - Quadratic function.
  - Exponential function.
  - The step function



# Popular Examples

- Demo

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

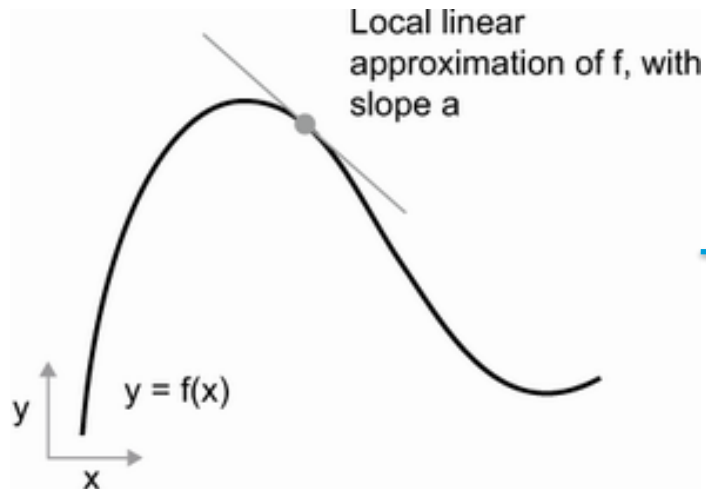
<https://studymachinelearning.com>

# Basic Calculus

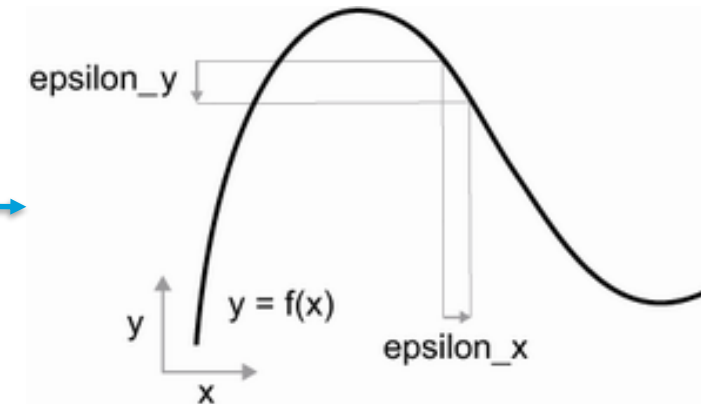
# The Derivative

- It is the **local** rate of change

- $\lim_{\epsilon_x \rightarrow 0} \frac{\epsilon_y}{\epsilon_x} = \frac{\partial y}{\partial x}$

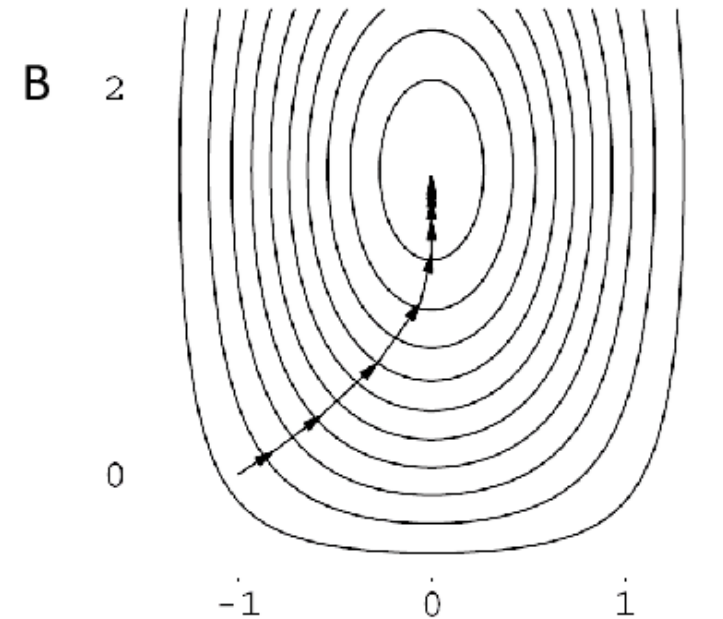
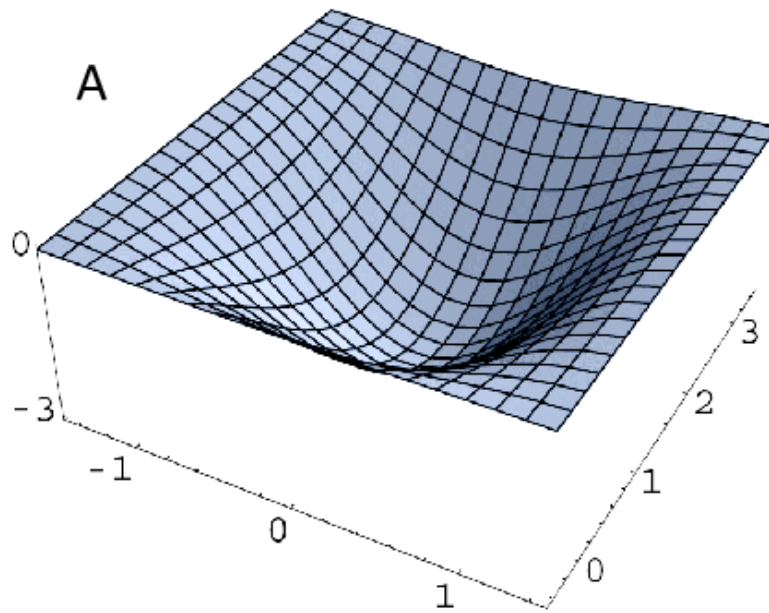


© Gordon Burtch, 2022



# The Gradient

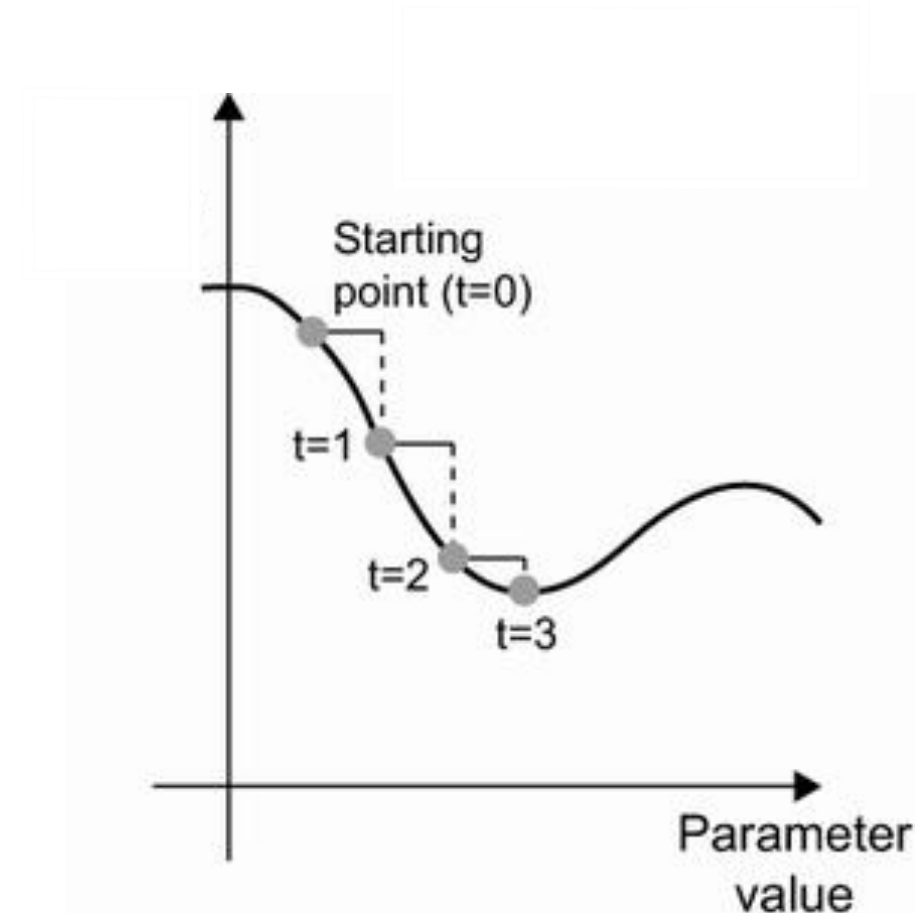
- The equivalent of the derivative in multi-dimensional space:  $\nabla_{x,y} z$



© Gordon Burtch, 2022

# Gradient Descent

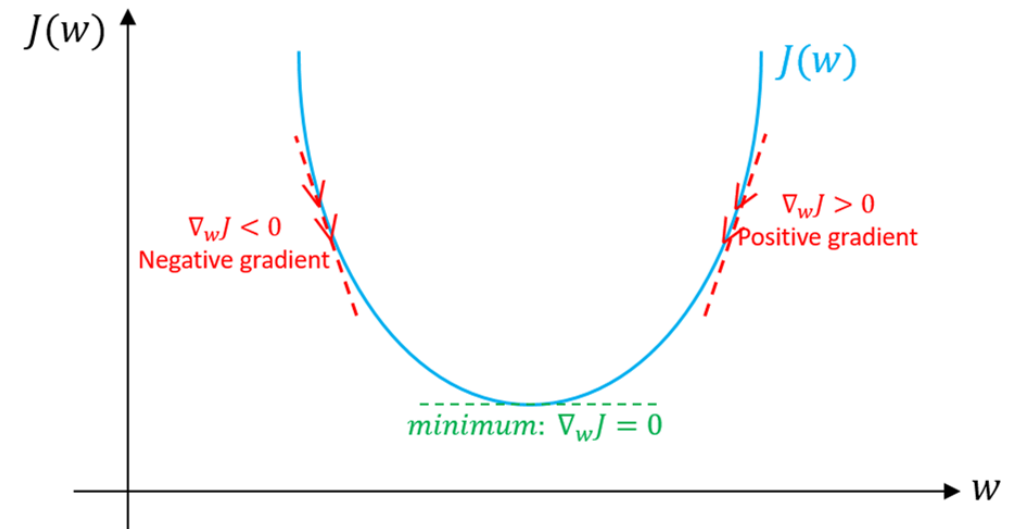
- Since the gradient points in the steepest direction, following it will lead us to minimizing (or maximizing) the function's value.



© Gordon Burtch, 2022

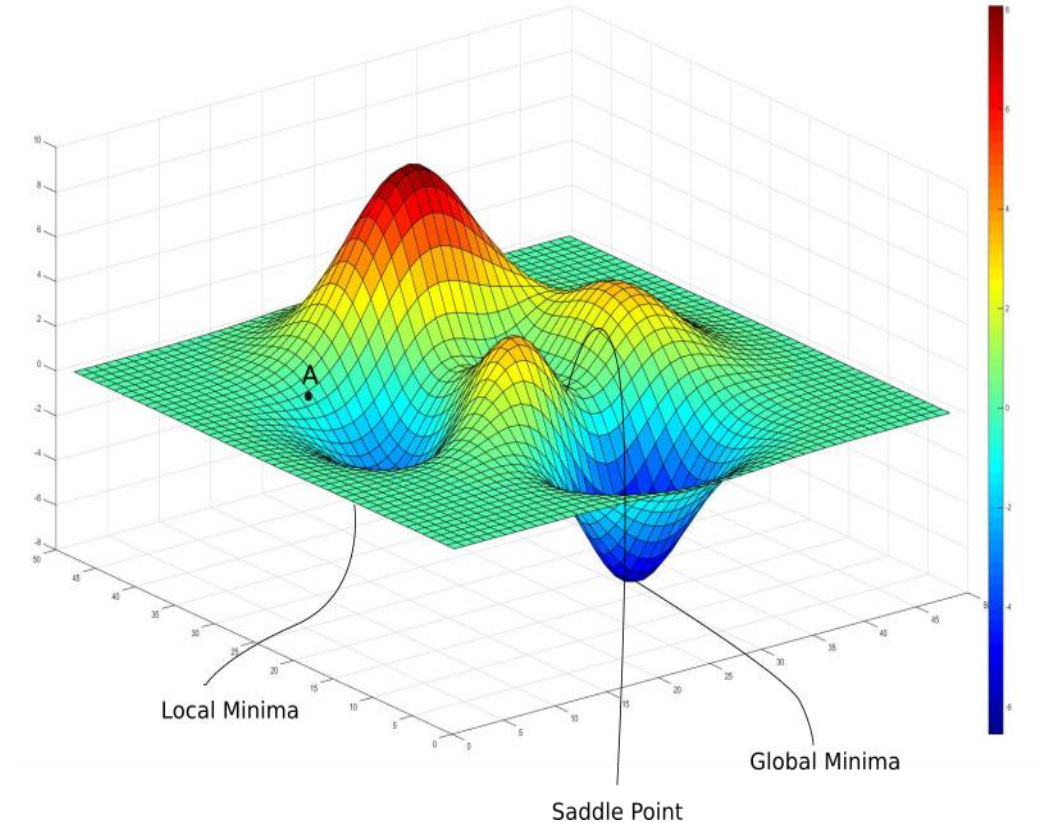
# Optimization

- By following the direction of the gradient, you can minimize an error function. This is called optimization.
  - $\frac{\partial J}{\partial w}$  for one-dimensional parameter.
  - becomes  $\nabla_w J$  when  $w$  is high-dimensional



# Optimization

- Can we always achieve lowest error?
- Demo



TechTalks

# Chain Rule

- To get the gradient of a function of a function:
  - Chain (multiply) the gradient of the first in terms of the second, ... until you reach the independent parameter.

$$z = f(x, y) \quad x = x(u, v)$$

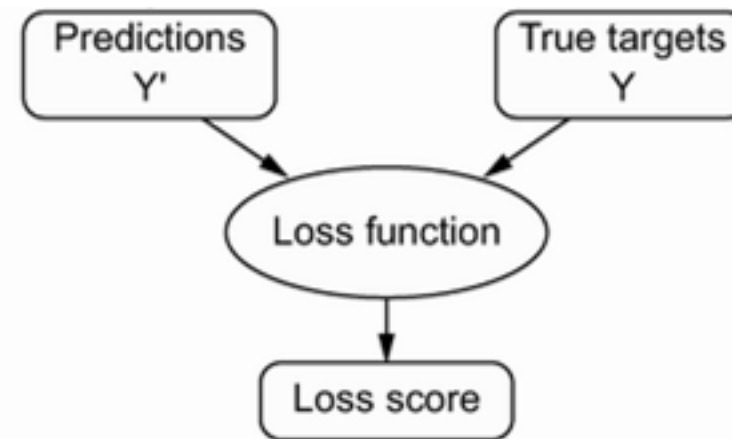
$$\frac{\partial z}{\partial u} = \frac{\partial z}{\partial x} \cdot \frac{\partial x}{\partial u} + \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial u}$$



# Calculating Error

# Calculating Error

- The terms **error**, **loss**, and **cost** are almost used interchangeably.
- A loss is generally a measure of how “different” the ground truth is from the model’s predictions.

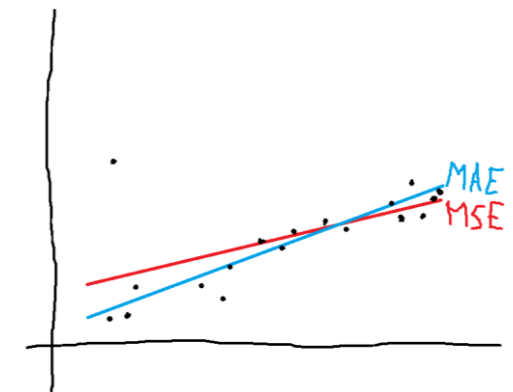
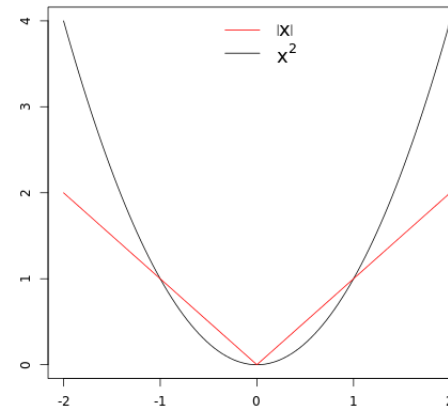


# Types of Loss Functions

- Mean Absolute Error (MAE/ L1 loss)
  - Large and small differences are penalized proportionally.
- Mean Squared Error (MSE/ L2 loss)
  - Larger differences are penalized exponentially more than smaller ones

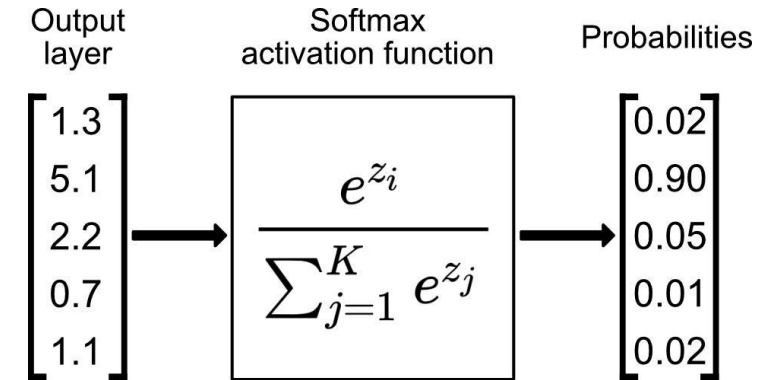
$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}$$

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

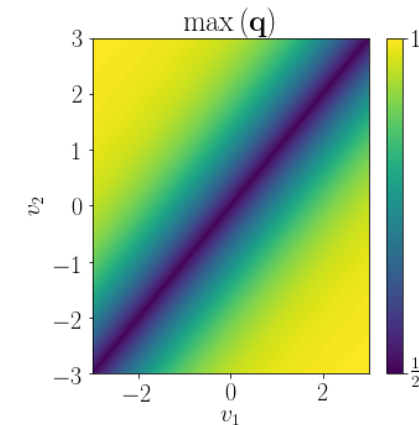


# Obtaining Probabilities

- Probabilities occur frequently in the context of classification.
- Given an array of numbers, a **softmax** converts that array into a probability distribution
  - This is not the only way to convert an array into a probability distribution. However, it is the most popular way in deep learning. Why?



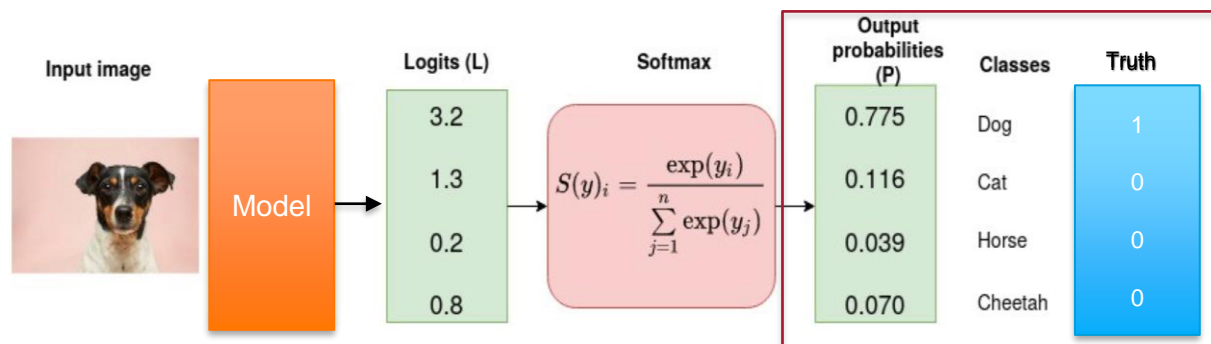
<https://towardsdatascience.com/>



<https://charlielehman.github.io>

# Measuring Error in Probabilities

- It is generally used to measure the error between the ground truth and a probability distribution.
- Given the output of some classification mode, you can turn it into probabilities using *softmax*, and then measure the error using **cross-entropy**



Input image source: Photo by [Victor Grabarczyk](#) on [Unsplash](#). Diagram by author.

CSDN @Everloob

$$H(P^* | P) = - \sum_i \underbrace{P^*(i)}_{\text{TRUE CLASS DISTRIBUTION}} \log \underbrace{P(i)}_{\text{PREDICTED CLASS DISTRIBUTION}}$$

# The Perceptron

**The Primordial Cell of Neural Networks**

# The building block: The Perceptron

- Demo

---

**Algorithm:** Perceptron Learning Algorithm
 

---

```

 $P \leftarrow \text{inputs with label } 1;$ 
 $N \leftarrow \text{inputs with label } 0;$ 
Initialize  $\mathbf{w}$  randomly;
while !convergence do
    Pick random  $\mathbf{x} \in P \cup N$  ;
    if  $\mathbf{x} \in P$  and  $\mathbf{w} \cdot \mathbf{x} < 0$  then
        |  $\mathbf{w} = \mathbf{w} + \mathbf{x}$  ;
    end
    if  $\mathbf{x} \in N$  and  $\mathbf{w} \cdot \mathbf{x} \geq 0$  then
        |  $\mathbf{w} = \mathbf{w} - \mathbf{x}$  ;
    end
end
//the algorithm converges when all the
  inputs are classified correctly
  
```

---

Figure courtesy of Akshay L.Chandra

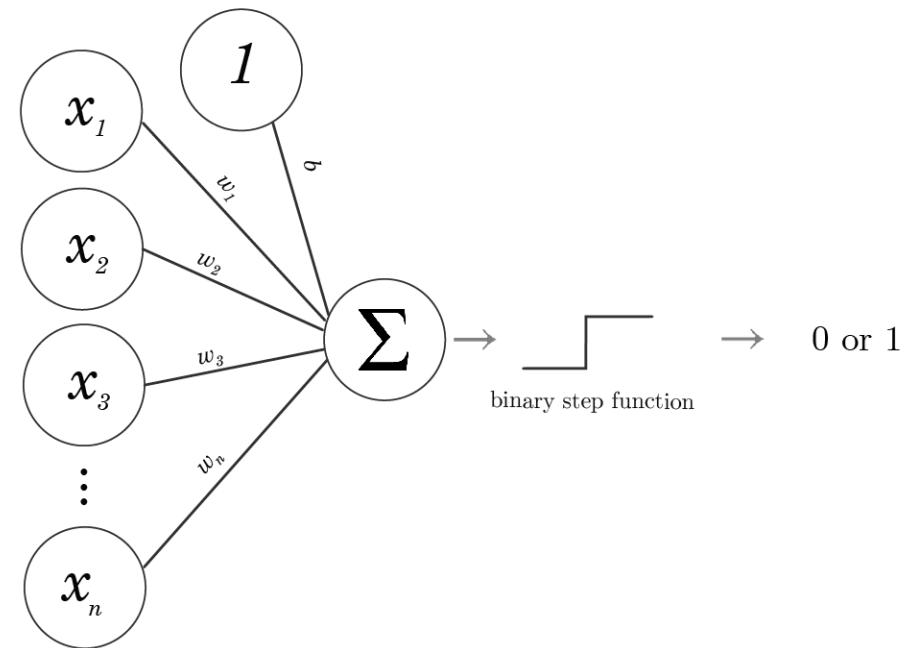
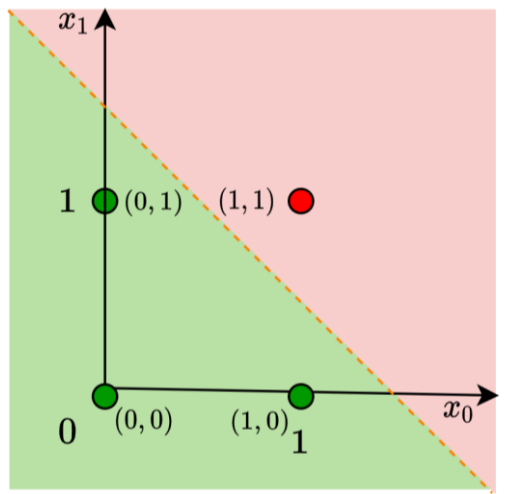


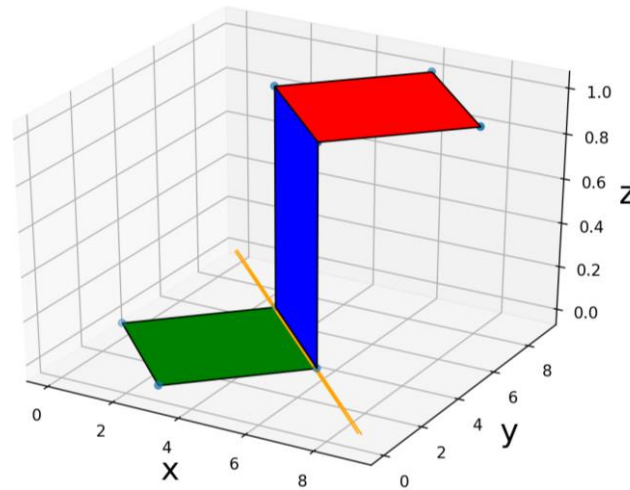
Figure courtesy of Akshay L.Chandra

# The building block: The Perceptron

- Does the perceptron remind you of something?
  - The perceptron uses a step function. Logistic Regression uses a sigmoid.

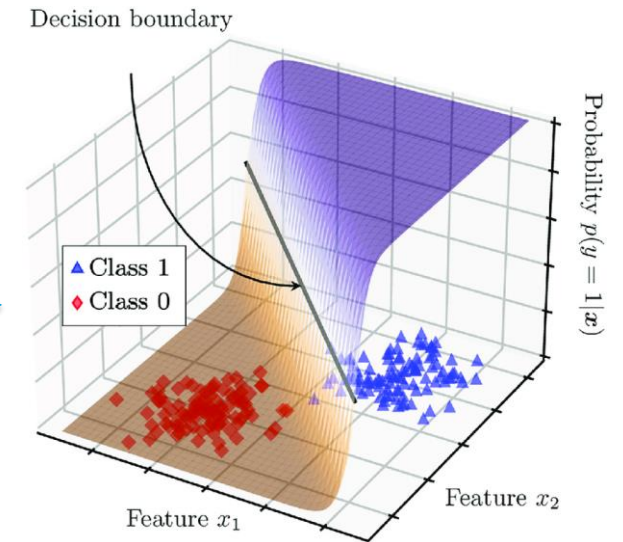


Perceptron - Top view



Perceptron - 3D view

<https://livebook.manning.com>



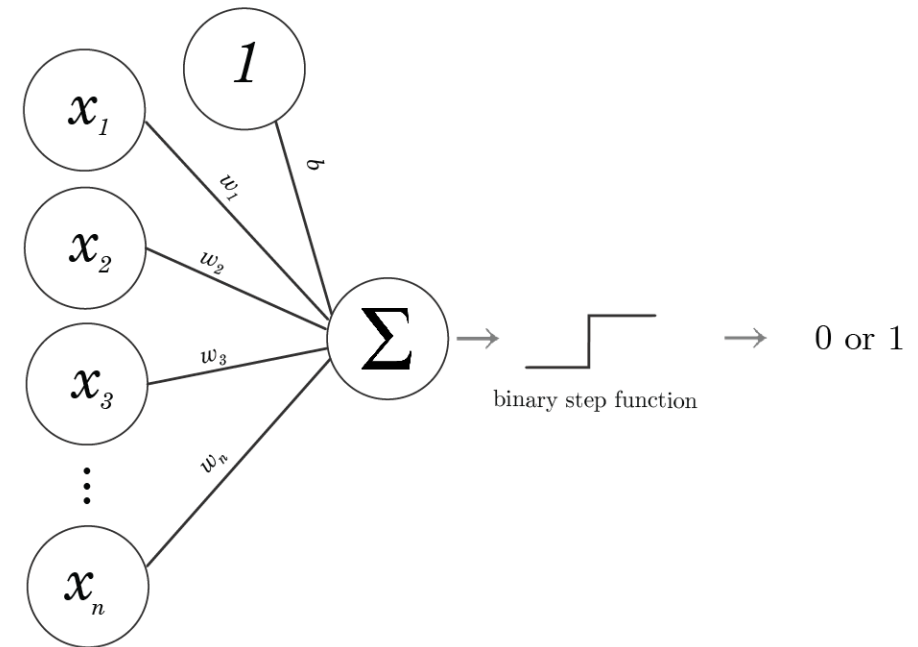
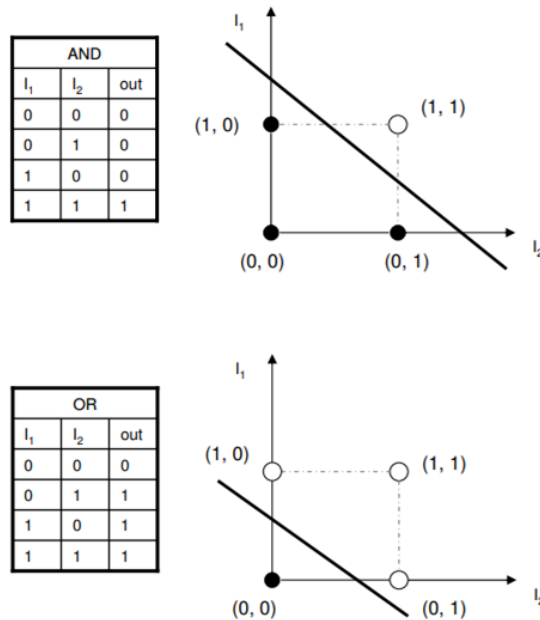
Logistic Regression - 3D view

<https://www.researchgate.net>



# The building block: The Perceptron

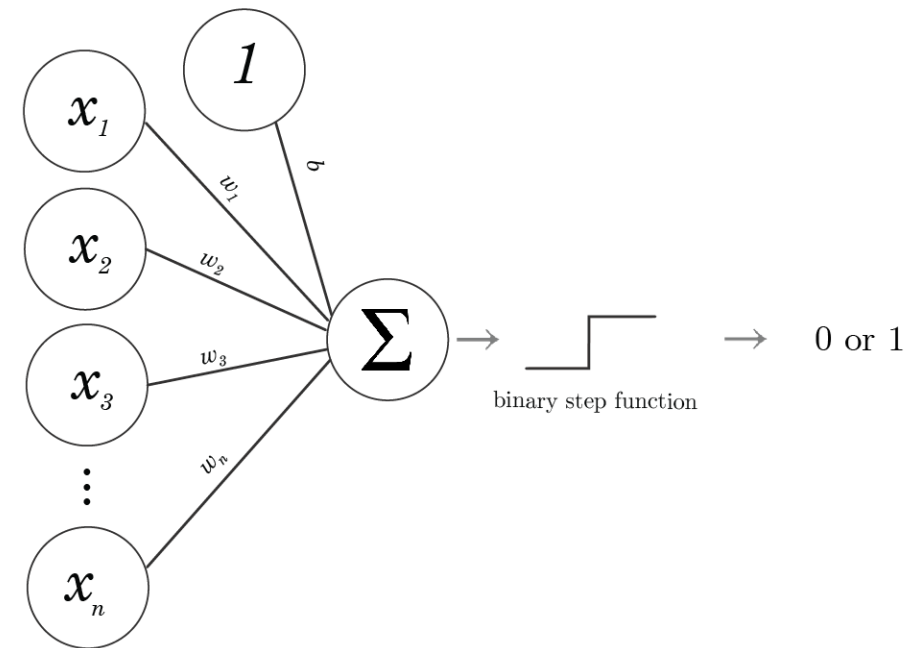
- Works well for linearly separable cases.



Adam Dhala

# The building block: The Perceptron

- Can we work around the “linear separability” issue?

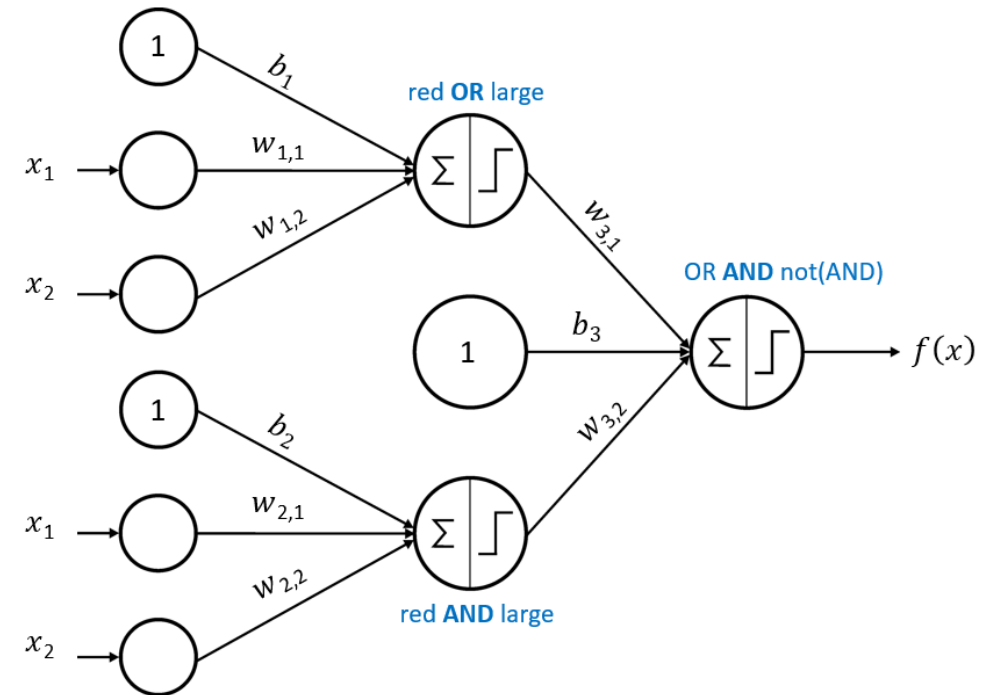
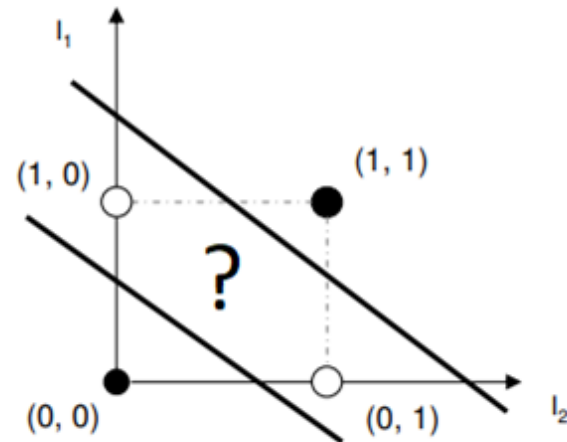


Adam Dhala

# Power in Numbers: Multiple Perceptrons

- Demo

XOR		
$I_1$	$I_2$	out
0	0	0
0	1	1
1	0	1
1	1	0

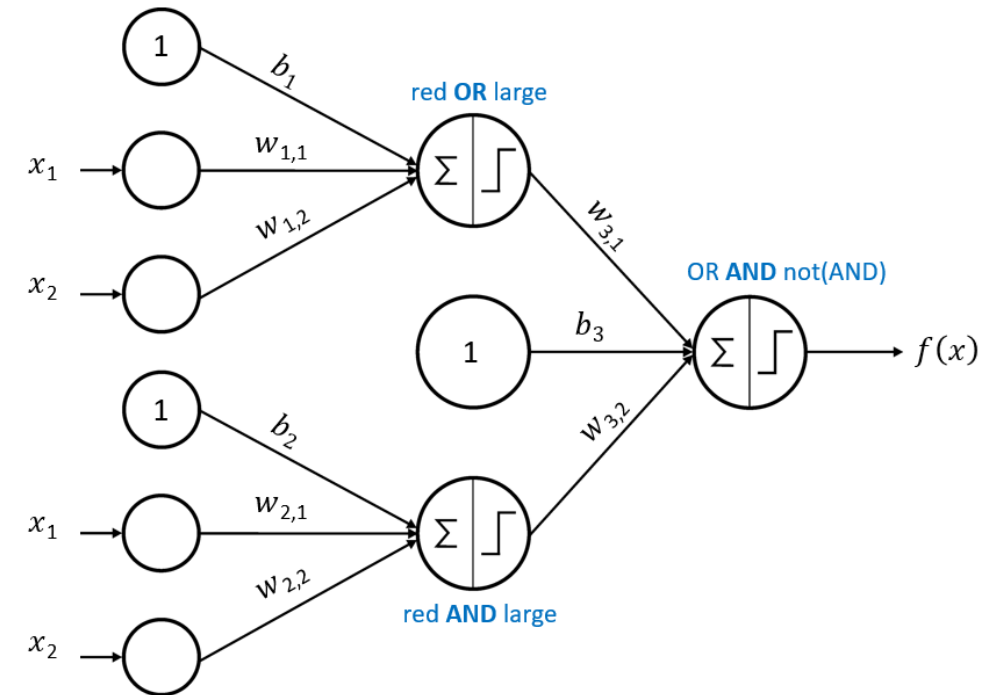


[western-neuralnets.ca](http://western-neuralnets.ca)

# Power in Numbers: Multiple Perceptrons



The Multi-Layer Perceptron  
(MLP) is born!



[western-neuralnets.ca](http://western-neuralnets.ca)

# More Layers!

- In [theory](#), a single hidden layer is sufficient to learn any function. In practice however, networks with more layers are easier to optimize than networks larger width.
- The more layers a model has, the higher its capacity and the more complex decision boundaries it can learn.


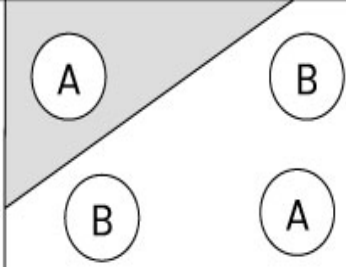
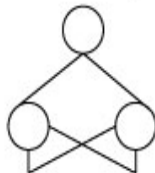
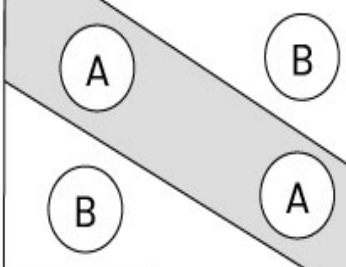
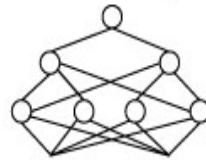
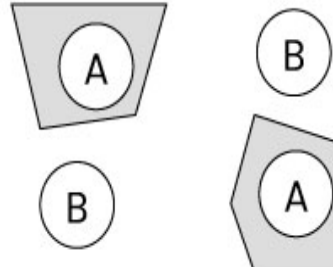
	Types of Decision Regions	Exclusive-OR Problem
<b>Single-Layer</b> 	Half Plane Bounded by Hyperplane	
<b>Two-Layer</b> 	Convex Open or Closed Regions	
<b>Three-Layer</b> 	Arbitrary (Complexity Limited by No. of Nodes)	

Figure courtesy of [Very Engineering Team](#)