This is CS5045!

GCR:ioc7cdl

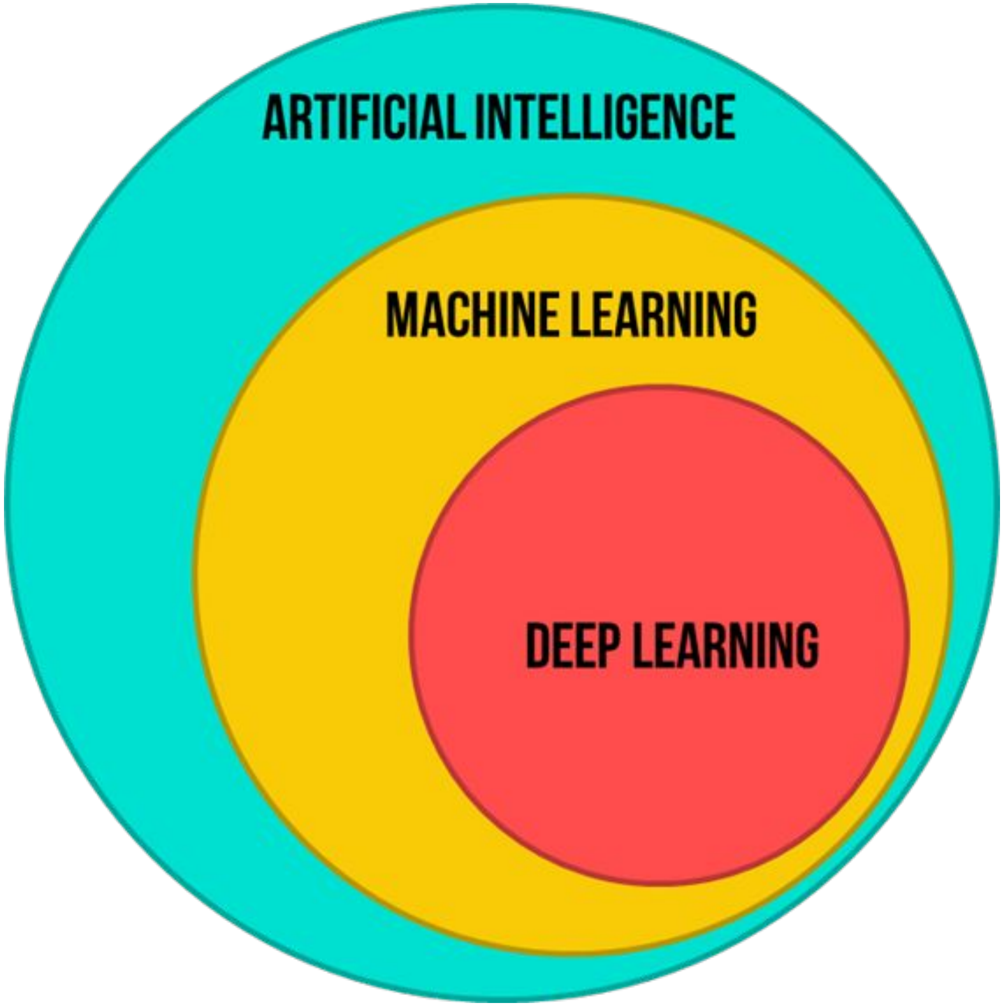
Hi,
I am Sumaiyah

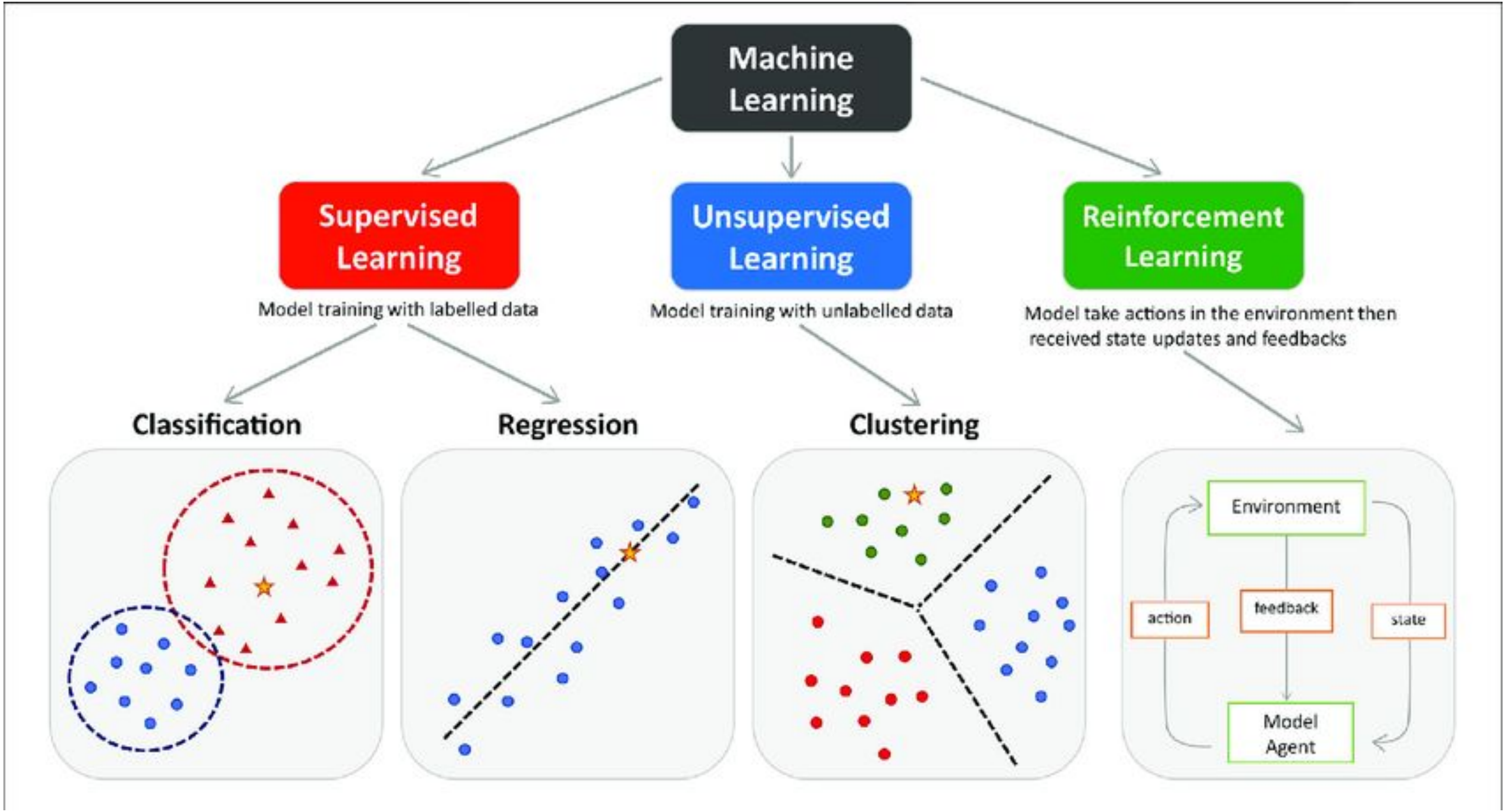
Email : Sumaiyah@nu.edu.pk

Office : In front of CS Secretariat

P.S. These slides are useless if you do not attend classes

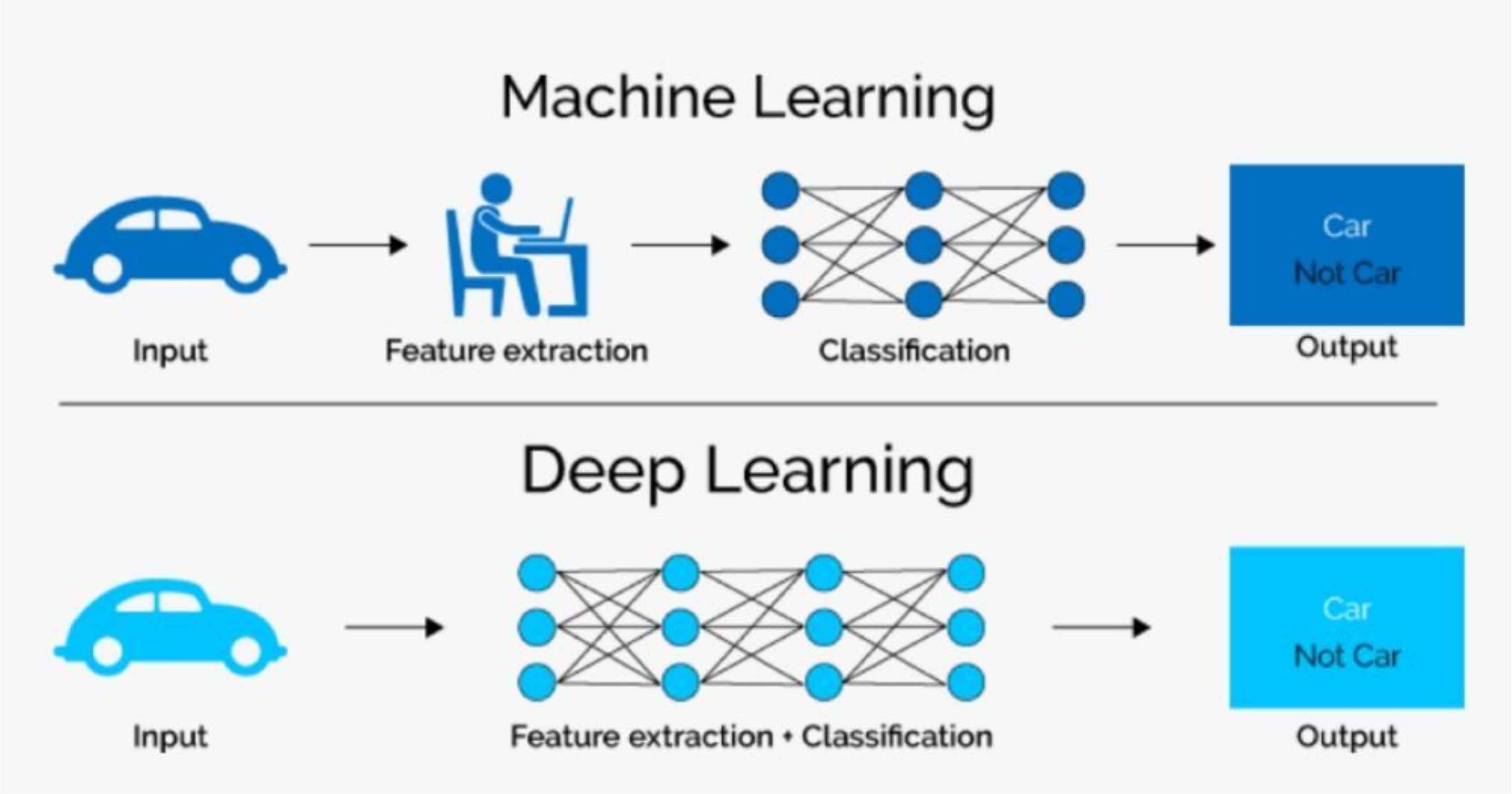
Machine Learning Recap


ARTIFICIAL INTELLIGENCE
MACHINE LEARNING
DEEP LEARNING


Machine Learning
Supervised Learning
Model training with labelled data
Unsupervised Learning
Model training with unlabelled data
Reinforcement Learning
Model take actions in the environment then received state updates and feedbacks
Classification
Regression
Clustering
Environment
action
feedback
state
Model Agent

Machine Learning Terms

- Overfitting
- Underfitting
- Training / Validation / Testing
- Cross Validation

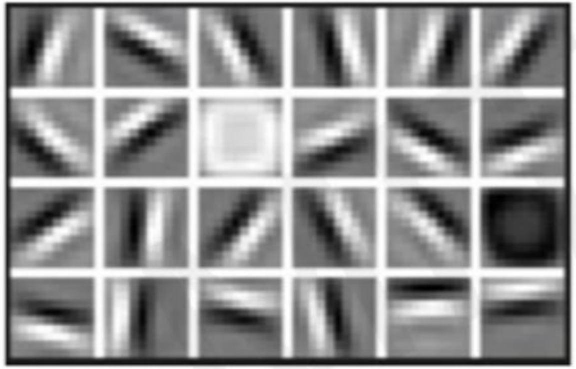

Machine Learning
Car
Not Car
Input
Feature extraction
Classification
Output
Deep Learning
Car
Not Car
Input
Feature extraction + Classification
Output

Why Deep Learning?

Hand engineered features are time consuming, brittle, and not scalable in practice

Can we learn the **underlying features** directly from data?

Low Level Features



Lines & Edges

Mid Level Features

Eyes & Nose & Ears

High Level Features

Facial Structure

Why Now?

1952	Stochastic Gradient Descent
1958	Perceptron • Learnable Weights
⋮	
1986	Backpropagation • Multi-Layer Perceptron
1995	Deep Convolutional NN • Digit Recognition
⋮	

Neural Networks date back decades, so why the resurgence?

1. Big Data

- Larger Datasets
- Easier Collection & Storage

IMAGENET

WIKIPEDIA
The Free Encyclopedia

2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable

3. Software

- Improved Techniques
- New Models
- Toolboxes

TensorFlow

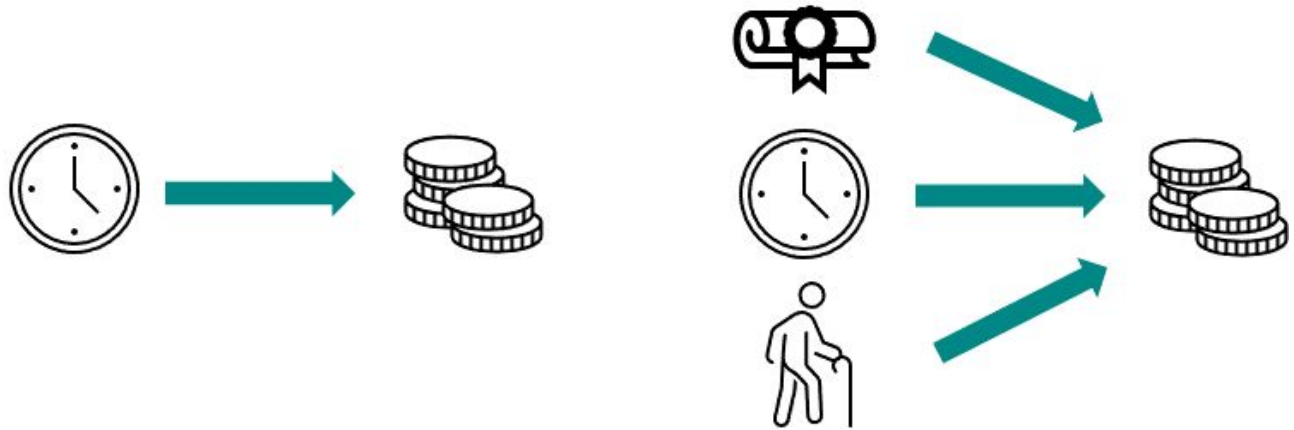
Deep Learning Applications

Discussed in class

Linear Regression

Simple Linear Regression

Multiple Linear Regression



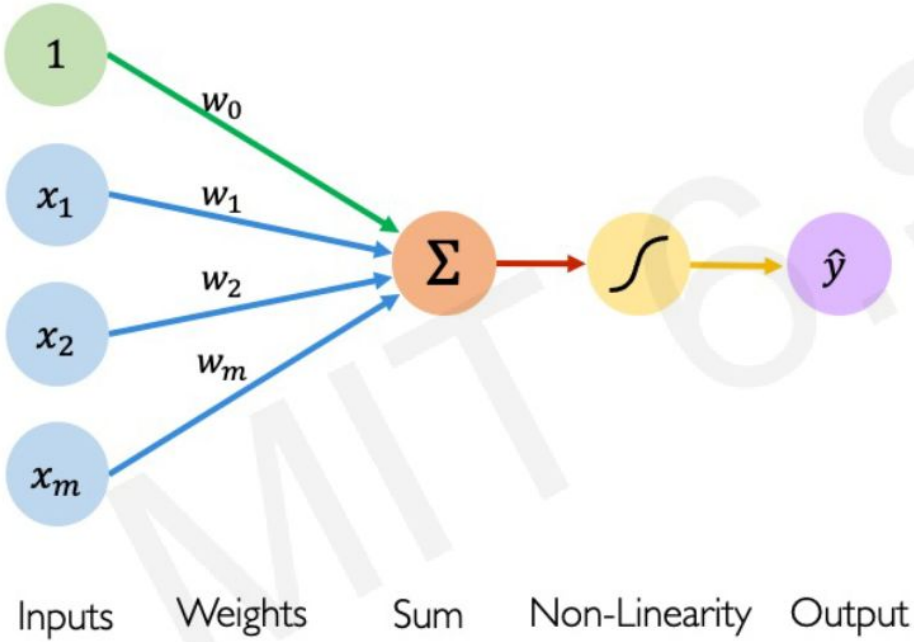
Simple Linear Regression

$$\hat{y} = b \cdot x + a$$

Multiple Linear Regression

$$\hat{y} = b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_k \cdot x_k + a$$

The Perceptron: Forward Propagation



Output

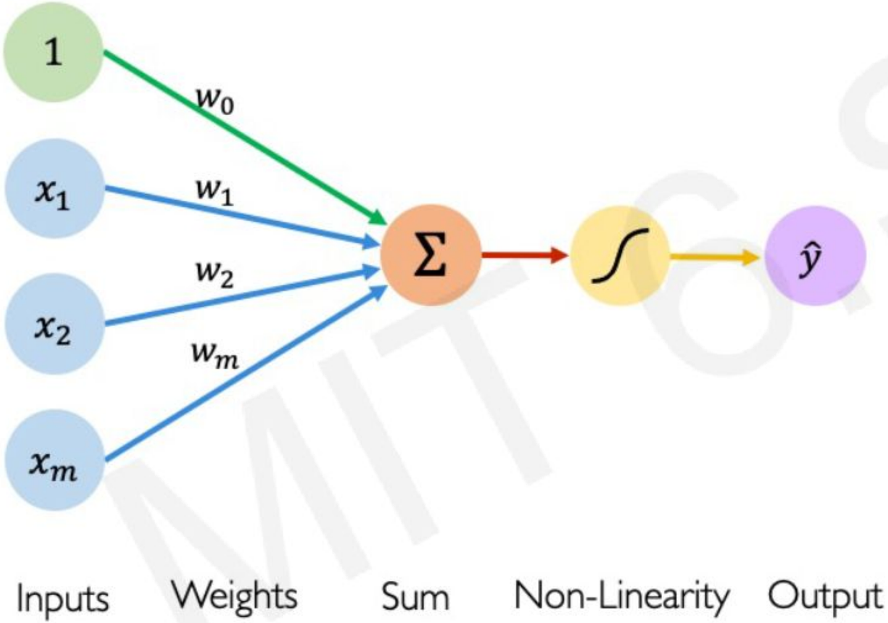
Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i \right)$$

Non-linear activation function

Bias

The Perceptron: Forward Propagation

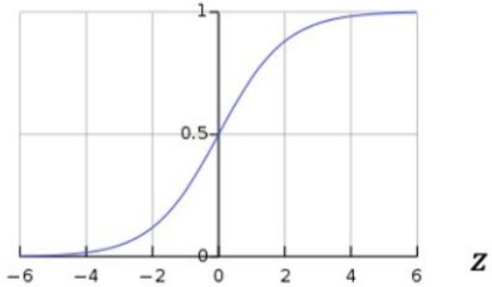


Activation Functions

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

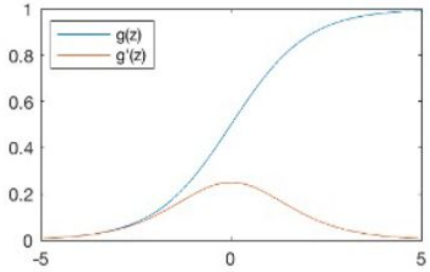
- Example: sigmoid function

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



Common Activation Functions

Sigmoid Function

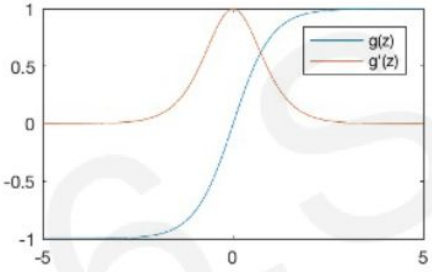


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

```
tf.math.sigmoid(z)
```

Hyperbolic Tangent

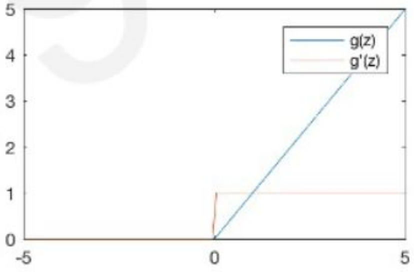


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

```
tf.math.tanh(z)
```

Rectified Linear Unit (ReLU)



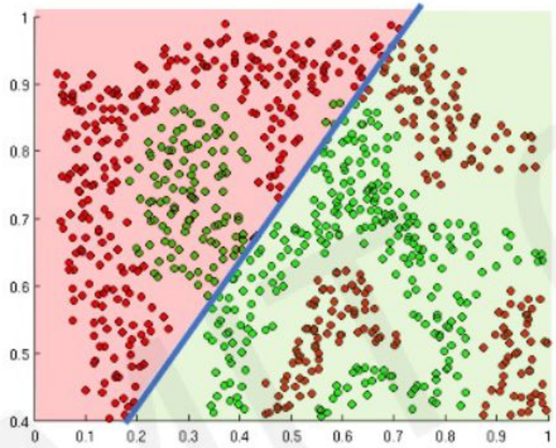
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

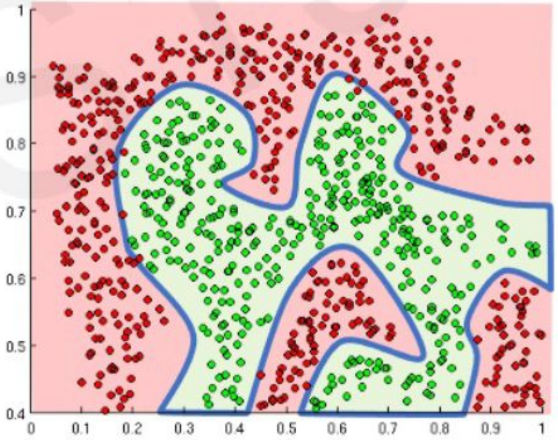
```
tf.nn.relu(z)
```

Importance of Activation Functions

The purpose of activation functions is to ***introduce non-linearities*** *into the network*



Linear activation functions produce linear decisions no matter the network size

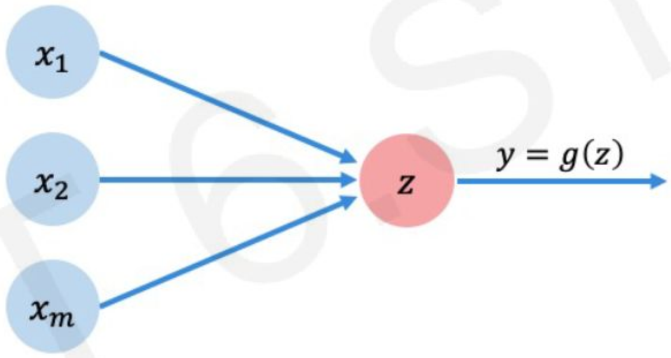


Non-linearities allow us to approximate arbitrarily complex functions

Deep Learning Terms

- Input Layer
- Output Layer
- Hidden Layer
- Neurons/ Nodes
- Shallow Neural Network
- Deep Neural Network
- Epoch
- Hyperparameter vs Parameters
- Activation Function

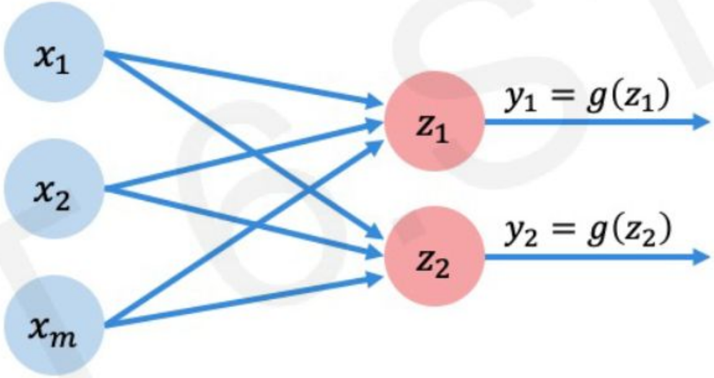
The Perceptron: Simplified



$$z = w_0 + \sum_{j=1}^{m} x_j \, w_j$$

Multi Output Perceptron

Because all inputs are densely connected to all outputs, these layers are called **Dense** layers



$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

Dense layer from scratch

```python
class MyDenseLayer(tf.keras.layers.Layer):
  def __init__(self, input_dim, output_dim):
    super(MyDenseLayer, self).__init__()

    # Initialize weights and bias
    self.W = self.add_weight([input_dim, output_dim])
    self.b = self.add_weight([1, output_dim])

  def call(self, inputs):
    # Forward propagate the inputs
    z = tf.matmul(inputs, self.W) + self.b

    # Feed through a non-linear activation
    output = tf.math.sigmoid(z)

    return output
```

References

- https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
- https://arunaddagatla.medium.com/maximum-likelihood-estimation-in-logistic-regression-f86ff1627b67
- https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e
- https://www.marktechpost.com/2021/04/08/logistic-regression-with-keras/
- https://www.datasciencecentral.com/logistic-regression-as-a-neural-network/
- https://github.com/SSaishruthi/LogisticRegression_Vectorized_Implementation/blob/master/Logistic_Regression.ipynb
- https://towardsdatascience.com/where-did-the-binary-cross-entropy-loss-function-come-from-ac3de349a715
- https://www.youtube.com/watch?v=nzNp05AyBM8