# Neural Networks

Shyue Ping Ong

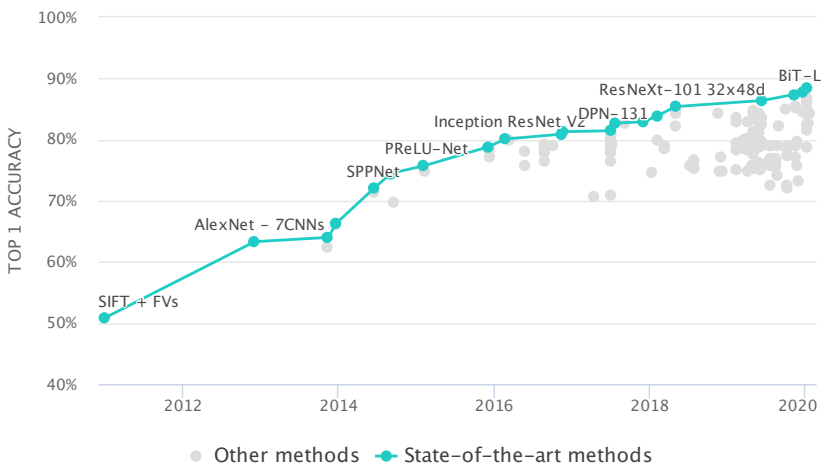
University of California, San Diego

NANO281

# Overview

# Preliminaries

- Neural networks/deep learning has gotten a lot of hype in recent years.
- In many areas, they have outperformed many traditional ML methodologies.

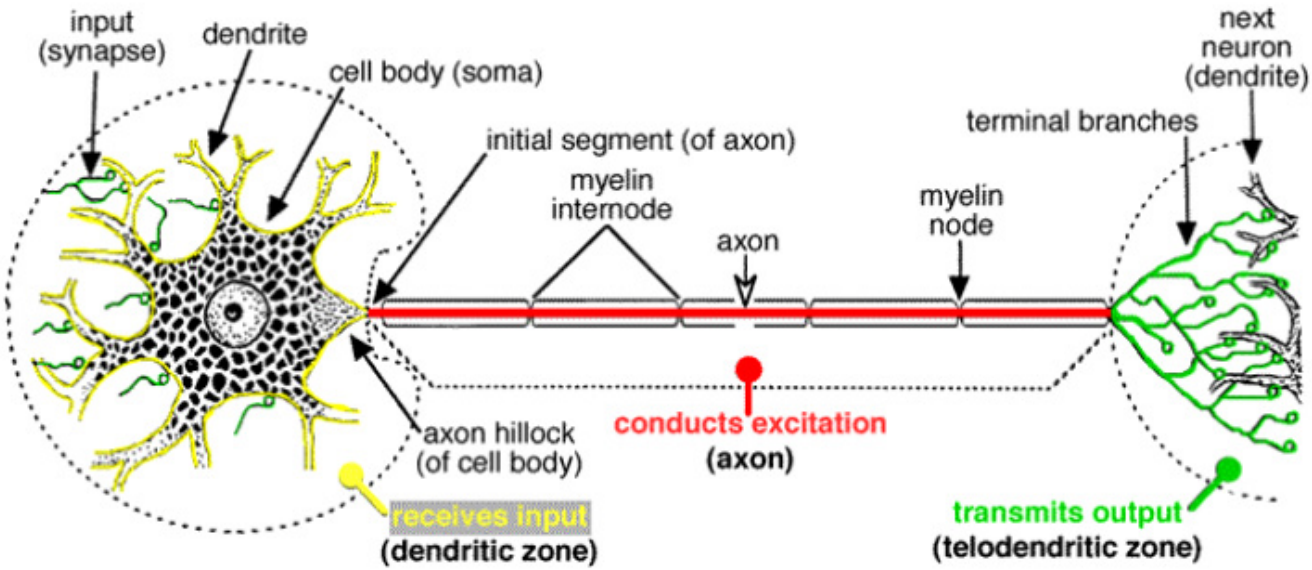



# Artificial Neural Network

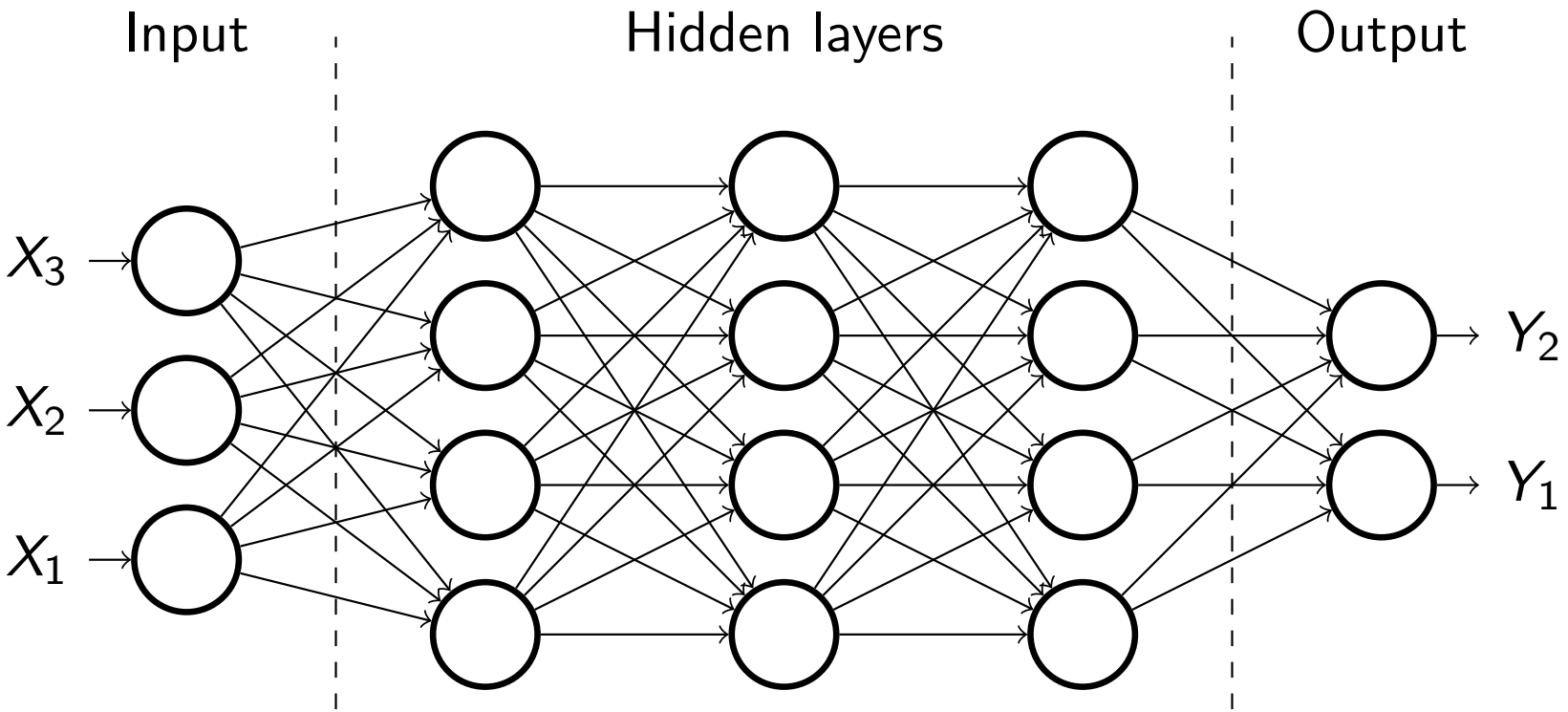
- An artificial neural network (NN) is a learning algorithm that is (very) loosely based on the structure of the brain.

## Universal Approximation Theorem[1]

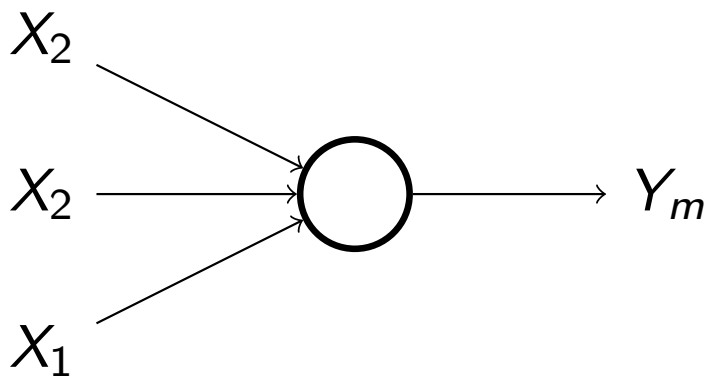
A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions under mild assumptions on the activation function.



# Neural Networks



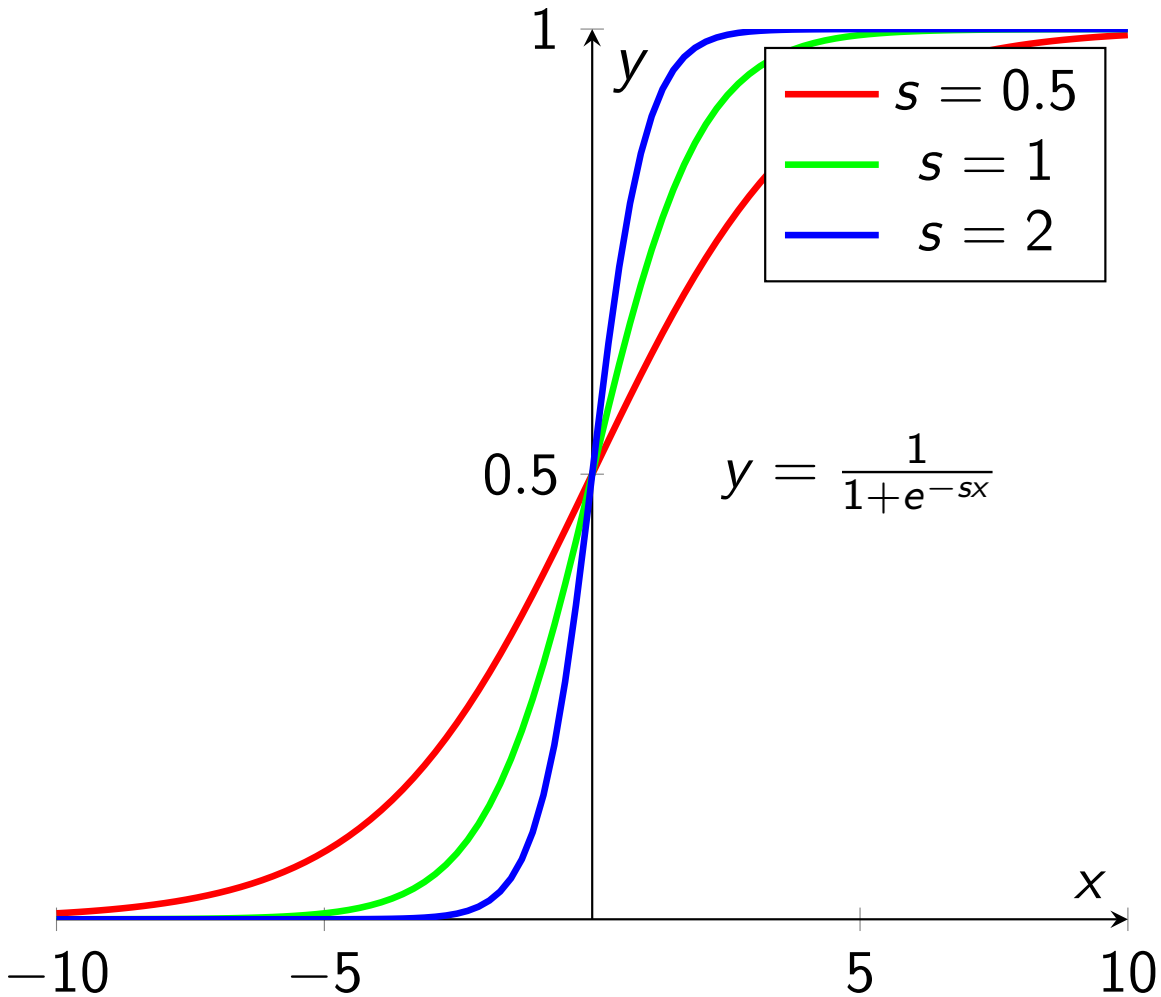
# Neuron




$$Y_m = \sigma(\alpha_{0m} + \alpha_m^T \mathbf{X})$$

- Output of each neuron is a linear function of the inputs.
- In hidden layers, the output is passed through an *activation function* $\sigma$.

# Sigmoid Activation Function



# Fitting a Neural Network

- Model *weights* ($\alpha$ in the linear functions) are fitted by *back-propogation*, basically a form of gradient descent.
- Loss functions: squared error for regression, squared error or cross entropy for classification.
- To avoid overfitting, regularization (similar to ridge regression) is typically applied. E.g., *weight decay*:

$$J = \sum \alpha^2$$

# Decisions for Neural Networks

- Number of hidden units and layers: generally error on the side of having too many hidden units than too few - flexibility is needed to capture non-linearities in the data.
- Extra weights can be shrunk to zero with appropriate regularization.
- Learning rate is a key parameter in NN as well as other model fitting. The learning rate controls the rate of gradient descent.
- The error function is non-convex and contains multiple minima, i.e., final solution obtained depends on initial weights. Typical approach is to start with a number of random starting configurations and choose solution with lowest penalized error.
- Alternative is to average predictions over a number of NN, i.e., ensemble models.

# Software packages for Neural Networks

- Scikit-learn has a basic implementation of a multi-layer perceptron. For this course, we will use this to avoid overloading students with different software packages.
- However, you should be aware that if you are into serious work with neural network, alternative software implementations are available that offer more features and efficiency. Note that these packages are not limited to NNs, but NNs are what their most common use case.
  - Tensorflow. Open-source package by Google. Very powerful and efficient, but a very steep learning curve. Highly recommend that you use the Keras package (already intergrated into TF2) which provides a high level API to it similar to scikit-learn in some ways.
  - Pytorch. Open-source package by Facebook. Equally powerful as TF but also steep learning curve.

# Bibliography


Balázs Csanád Csáji.
*Approximation with Artificial Neural Networks*.
PhD thesis.

# The End