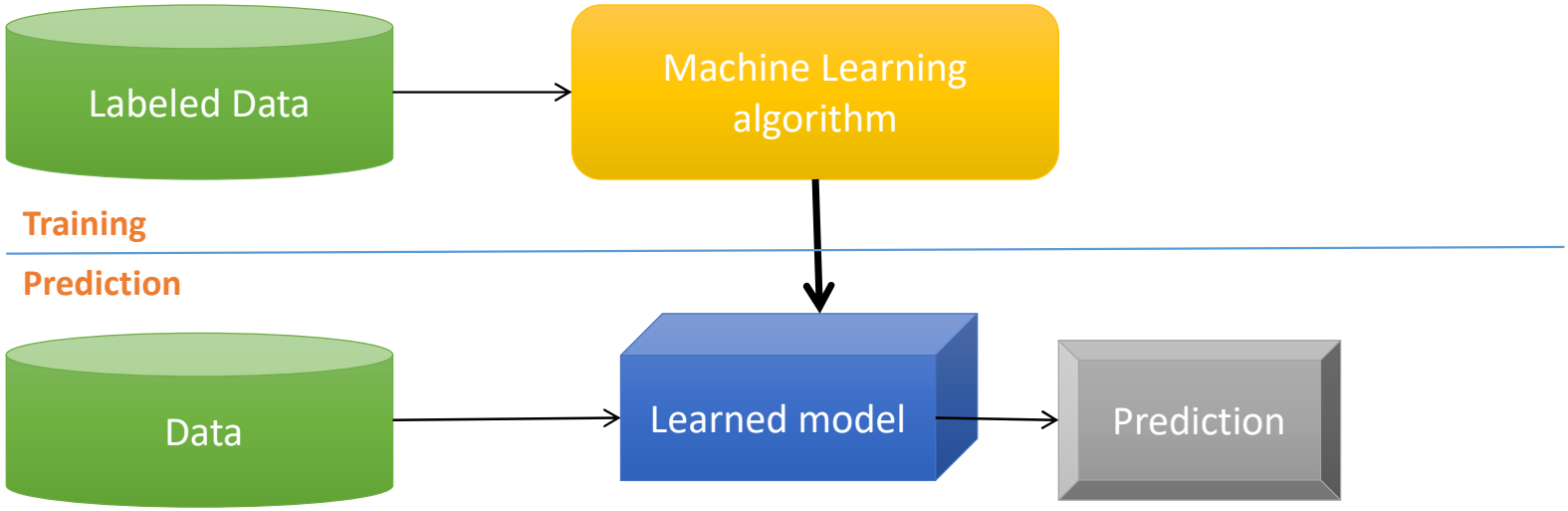
Neural Networks

Dr. Jameel Malik

muhammad.jameel@seecs.edu.pk

Machine Learning Basics -- Recap

Machine learning is a field of computer science that gives computers the ability to **learn without being explicitly programmed**

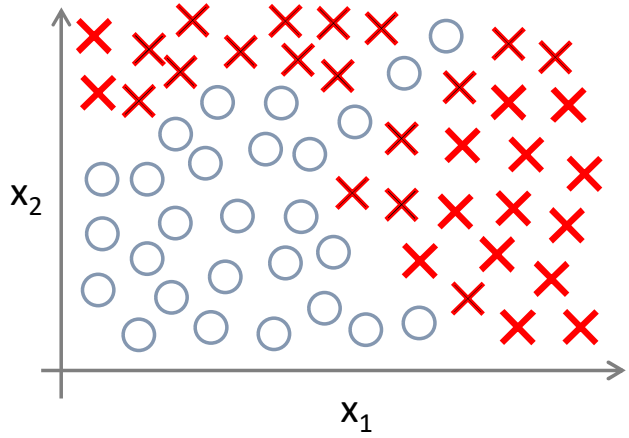


Methods that can learn from and make predictions on data

Why Neural Networks ?

- Let's say we need to learn complex non-linear hypothesis

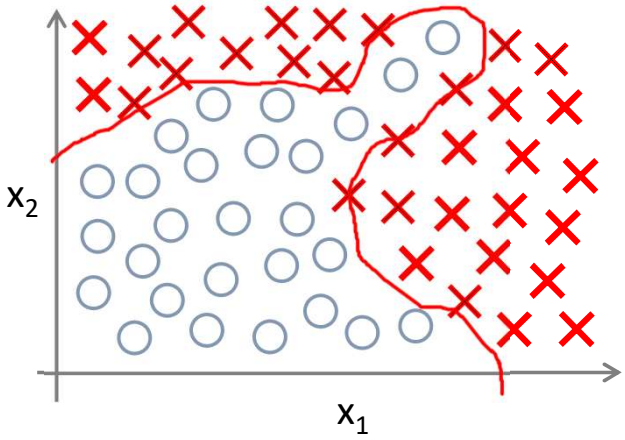
Classification Problem



Why Neural Networks ?

- Suppose we need to learn complex non-linear hypothesis

Classification Problem



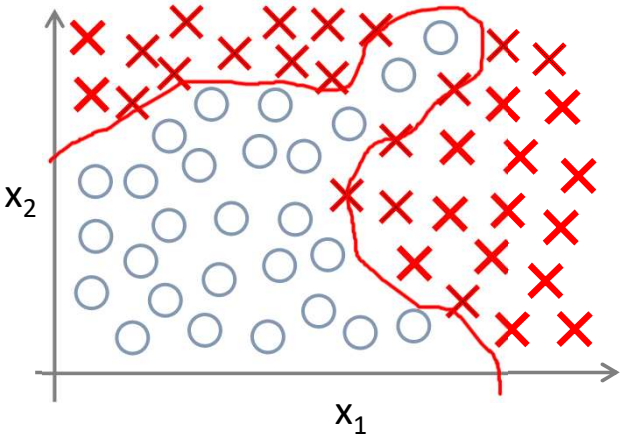
Assume the following Logistic Regression Hypothesis

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

Why Neural Networks ?

- Suppose we need to learn complex non-linear hypothesis

Classification Problem



$x_1 =$ size
$x_2 =$ # bedrooms
$x_3 =$ # floors
$x_4 =$ age
$\cdots$
x_{100}

Assume the following Logistic Regression Hypothesis

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

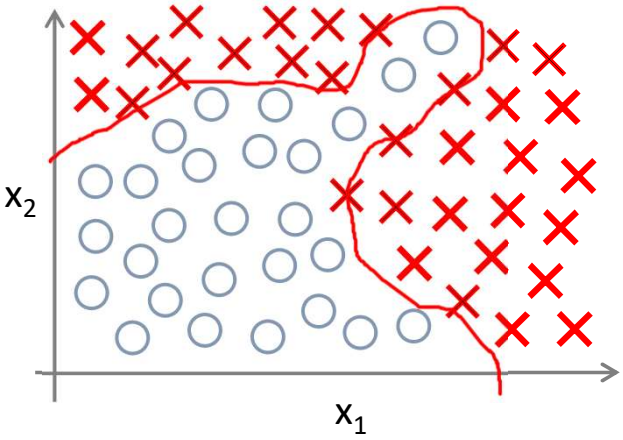
- What should be the hypothesis when we have lots of features?

Andrew ng

Why Neural Networks ?

- Suppose we need to learn complex non-linear hypothesis

Classification Problem



x_1 = size
x_2 = # bedrooms
x_3 = # floors
x_4 = age
$\dots$
x_{100}

Assume the following Logistic Regression Hypothesis

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

- What should be the hypothesis when we have lots of features?
  - may be we should assume a non-linear hypothesis with 1000's of features..
  - This can really blow up the features space
  - And we do not know which features can be more useful

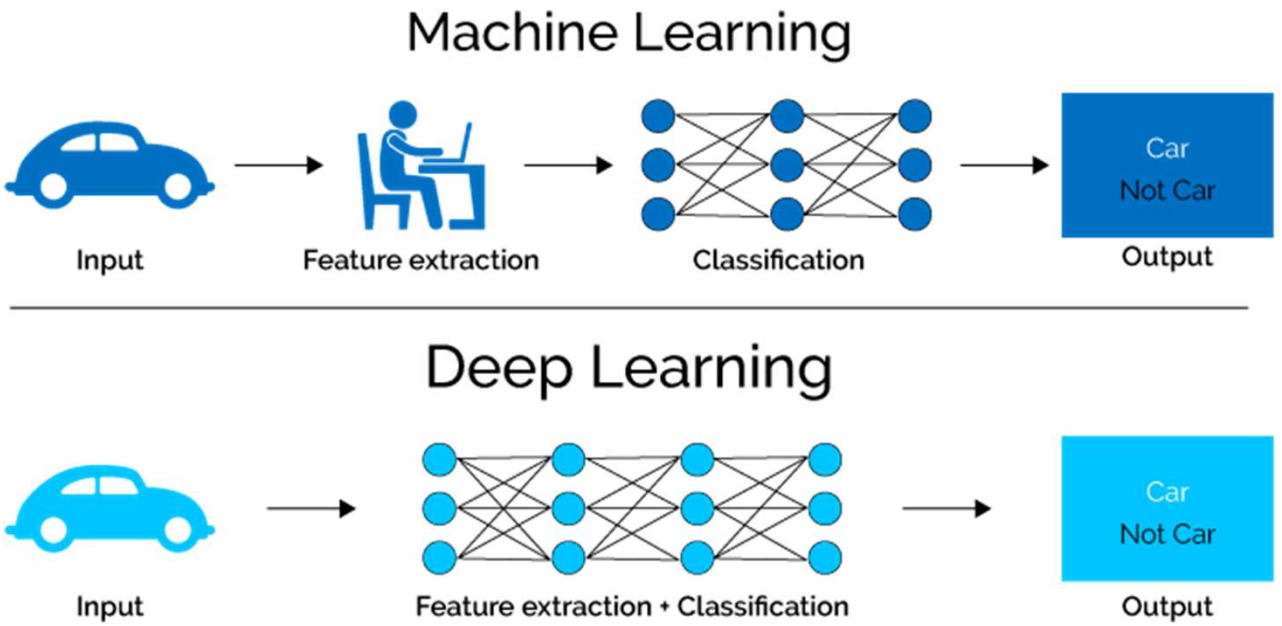
Introduction to Neural Networks

- Neural networks (NNs) is a type of machine learning algorithm that became popular in the 1980s.
- Lots of successes, hype, and great conferences: NeurIPS, ICML etc.
- Then along came SVMs, Random Forests and Boosting in the 1990s, and Neural Networks took a back seat.
- Re-emerged around 2010 as **Deep Learning.**
- By 2020s very dominant and successful.
- Part of success due to vast improvements in computing power, larger training sets, and software: Tensorflow and PyTorch.

- Much of the credit goes to three pioneers and their students: Yann LeCun, Geoffrey Hinton and Yoshua Bengio, who received the 2019 ACM Turing Award for their work in Neural Networks.

What is Deep Learning (DL) ?

Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**



https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

Why is DL useful?

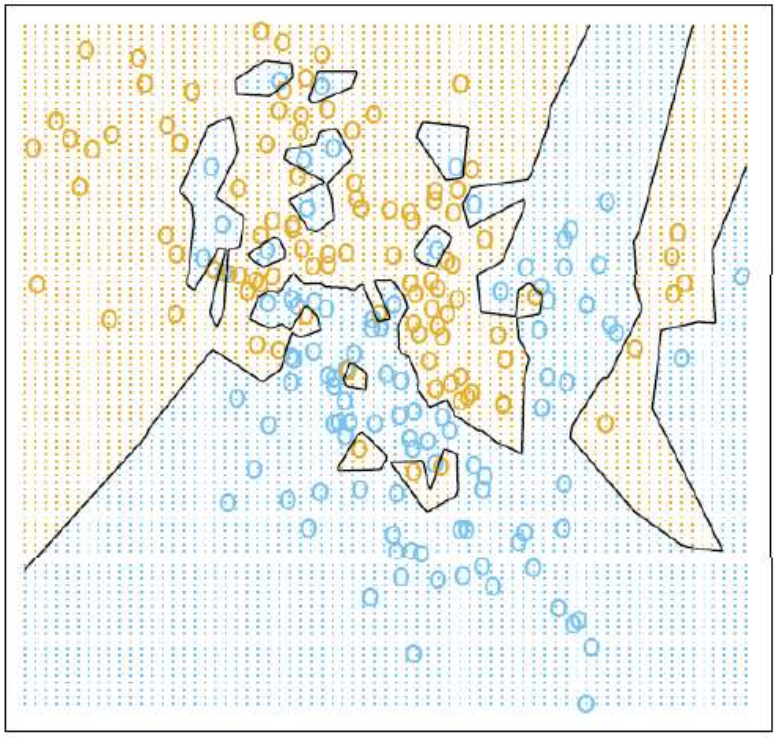
- Manually designed features are often **over-specified**, **incomplete** and take a **long time to design** and validate
- Automatically learned Features are **easy to adapt**, **fast** to learn
- Deep learning provides a very **flexible** learnable framework for representing world, visual and linguistic information
- Can learn from both unsupervised and supervised data
- Effective **end-to-end** system learning
- Utilize large amounts of training data

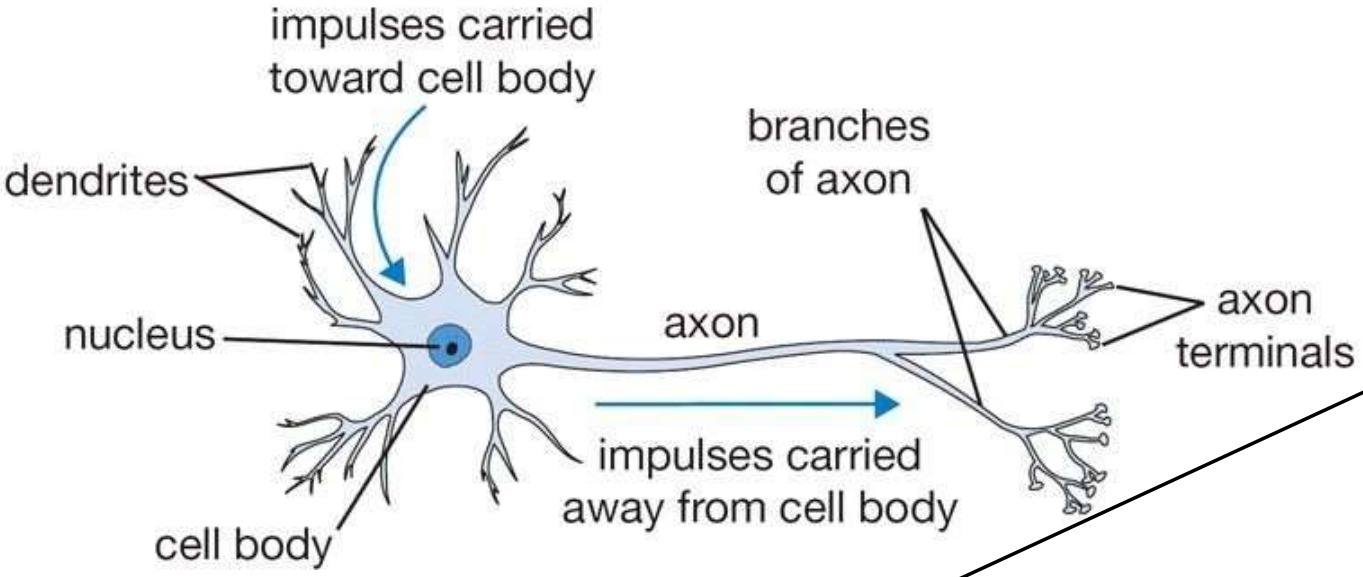
In ~2010 DL started outperforming other ML techniques first in speech and computer vision, then Natural Language Processing (NLP).

Representational Power of NNs

- NNs with at least one hidden layer are universal approximators
  i.e., NN can approximate any arbitrary complex continuous function

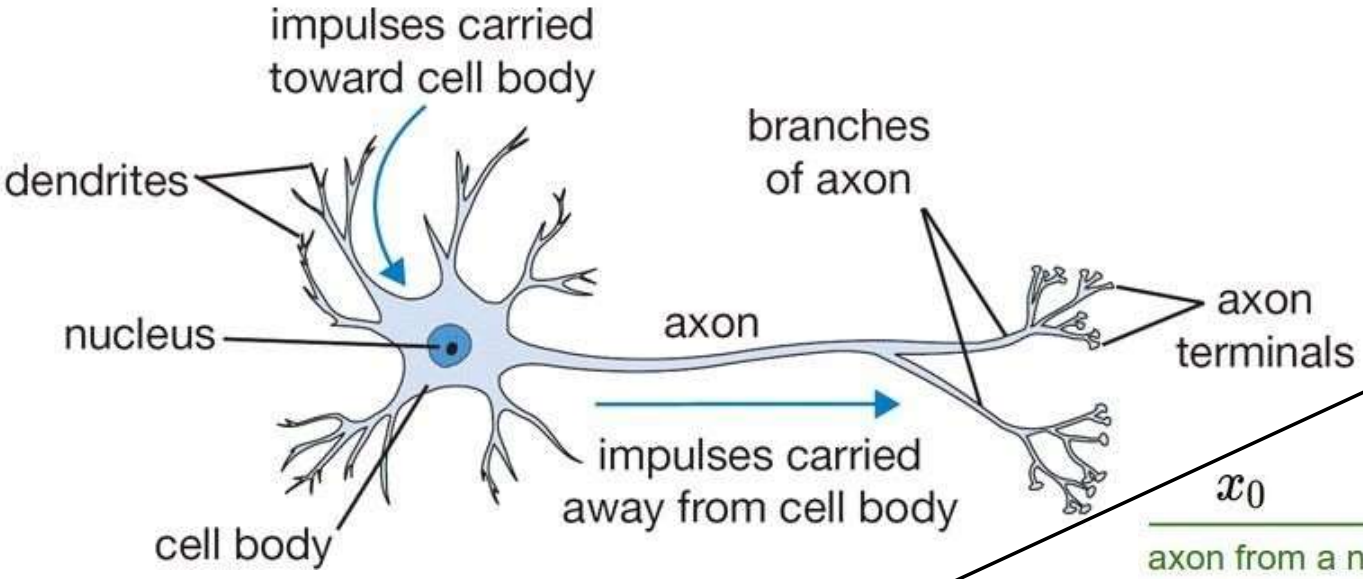
- NNs use **nonlinear mapping** of the inputs x to the outputs $f(x)$ to compute complex decision boundaries
- But then, why use **deeper NNs**?
  - The fact that deep NNs work better is an empirical observation

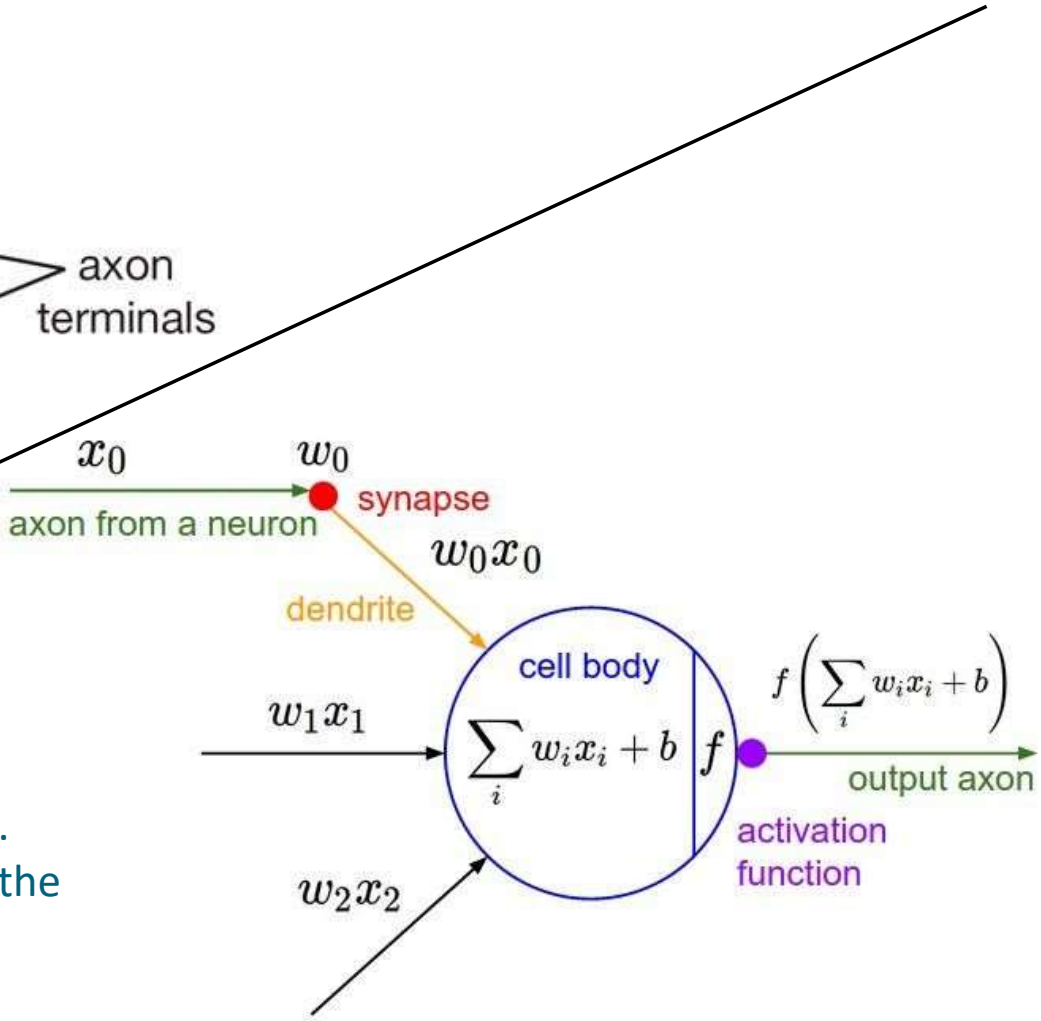




Perceptron

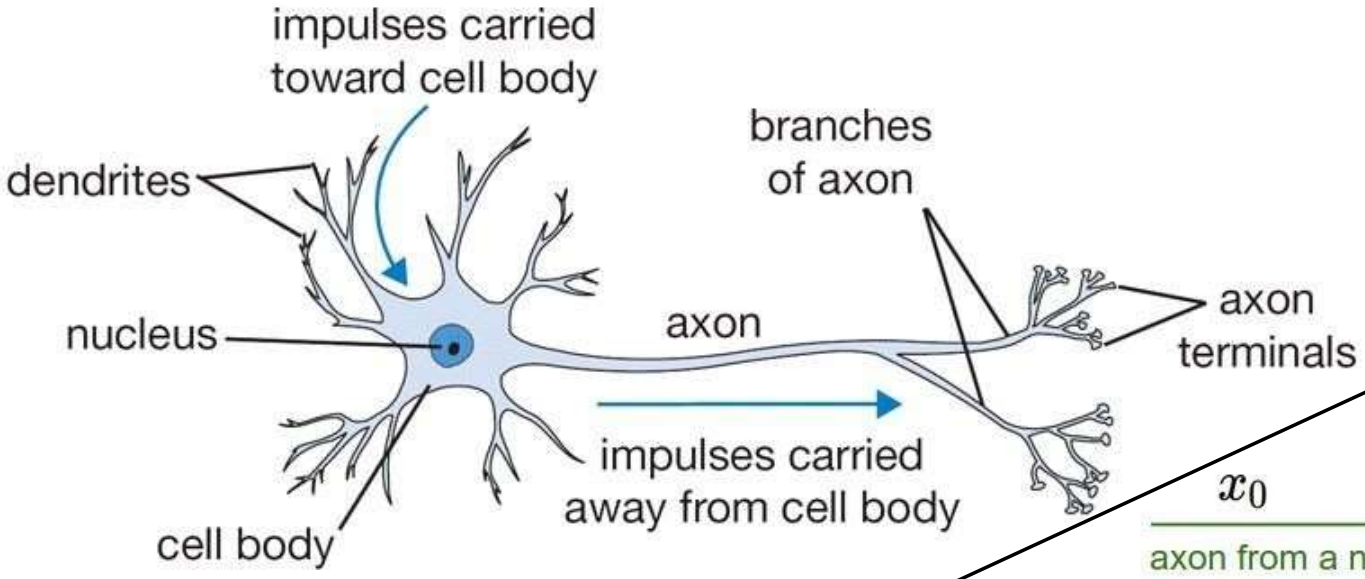
- The *perceptron* is the basic processing element of NN.
- It has inputs that may come from a sensor or may be the outputs of other perceptrons.

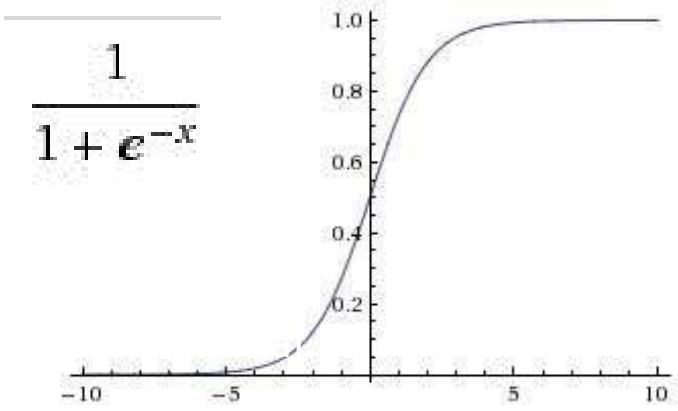


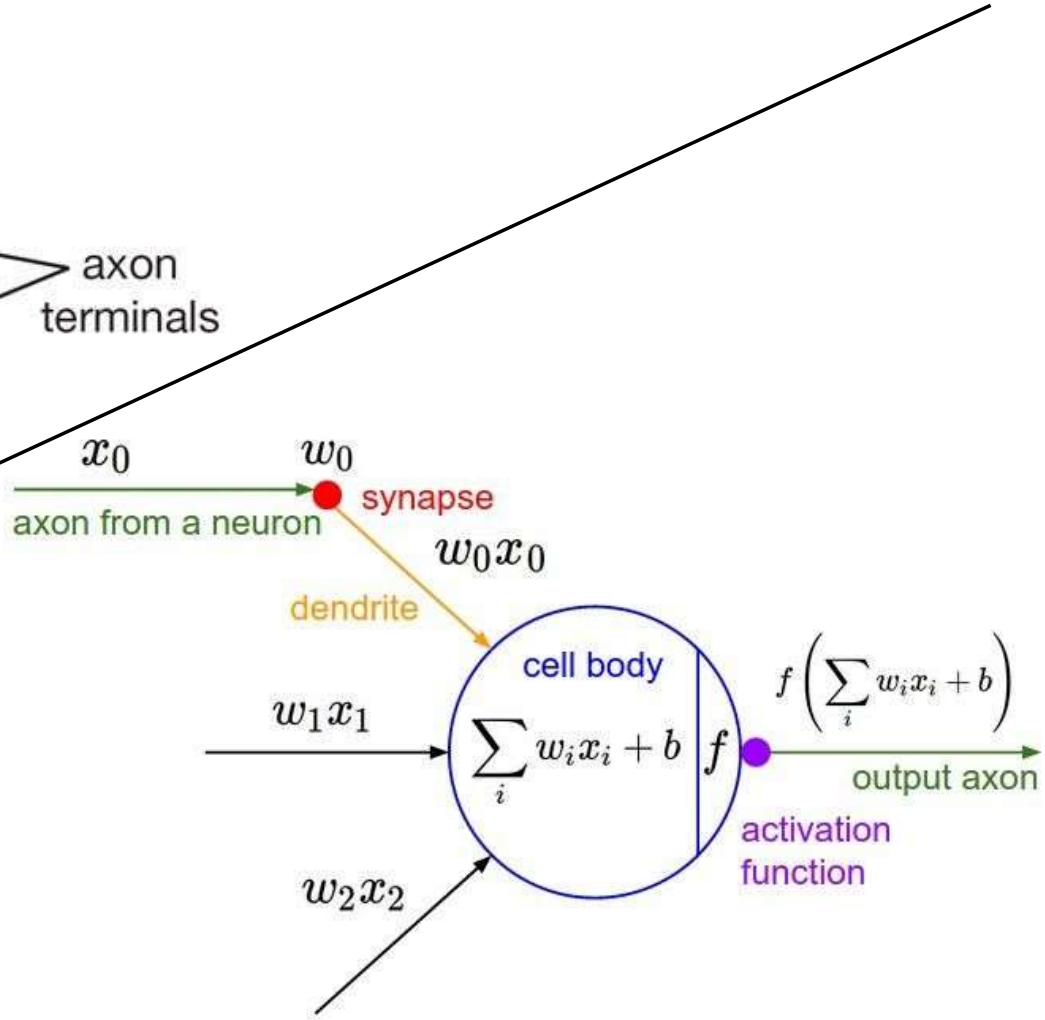


Perceptron

- The *perceptron* is the basic processing element of NN.
- It has inputs that may come from a sensor or may be the outputs of other perceptrons.





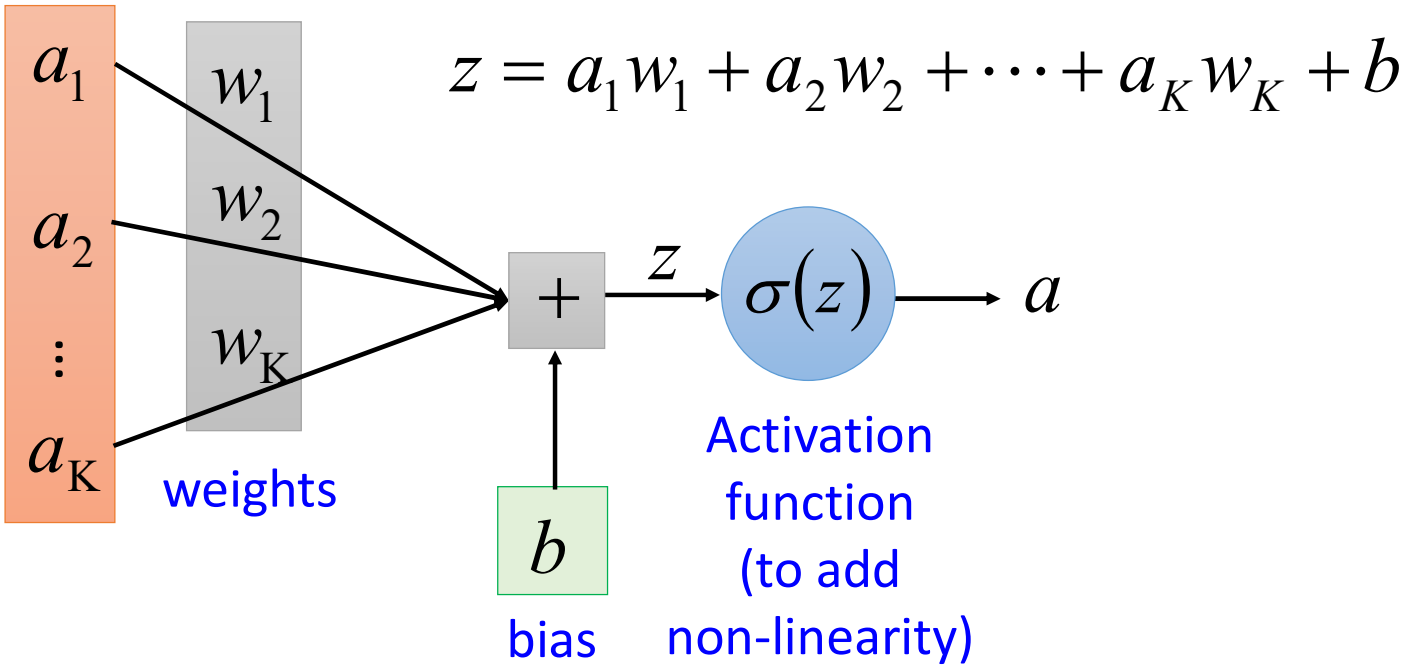


Nonlinear **sigmoid activation** function (f)

Elements of Neural Network

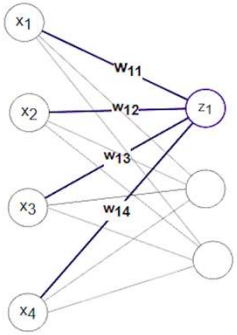
A single neuron maps a set of inputs into an output number, or $f: R^K \rightarrow R$

Neuron $f: R^K \rightarrow R$



A Simple Linear Layer

- **Memory Requirements:**
  - Parameters (weights, biases)
    - **12 + 3 = 15**



4 Input Neurons, 3 output Neurons

TIME

Thread 1

Calculate Weighted Sum

Thread 1

$z_1 = w_{11} * x_1 + w_{12} * x_2 + w_{13} * x_3 + w_{14} * x_4 + b$

Calculate Activation Function

z1 sigmoid a1

a1 = sigmoid(z1)

Thread 2

Calculate Weighted Sum

Thread2

$z_2 = w_{21} * x_1 + w_{22} * x_2 + w_{23} * x_3 + w_4 * x_4 + b$

Calculate Activation Function

z2 sigmoid a2

a2 = sigmoid(z2)

Thread 3

Calculate Weighted Sum

Thread3

$z_3 = w_{31} * x_1 + w_{32} * x_2 + w_{33} * x_3 + w_{34} * x_4 + b$

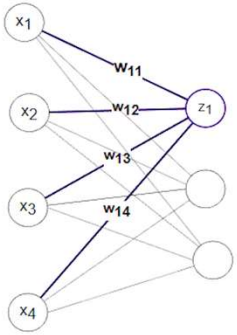
Calculate Activation Function

z3 sigmoid a3

a3 = sigmoid(z3)

A Simple Linear Layer – CPU execution?

- **Memory Requirements:**
  - Parameters (weights, biases)
    - **12 + 3 = 15**



4 Input Neurons, 3 output Neurons

TIME

Thread 1

Calculate Weighted Sum

Thread 1

$z_1 = w_{11} * x_1 + w_{12} * x_2 + w_{13} * x_3 + w_{14} * x_4 + b$

Calculate Activation Function

z1 sigmoid a1

a1 = sigmoid(z1)

Thread 2

Calculate Weighted Sum

Thread2

$z_2 = w_{21} * x_1 + w_{22} * x_2 + w_{23} * x_3 + w_4 * x_4 + b$

Calculate Activation Function

z2 sigmoid a2

a2 = sigmoid(z2)

Thread 3

Calculate Weighted Sum

Thread3

$z_3 = w_{31} * x_1 + w_{32} * x_2 + w_{33} * x_3 + w_{34} * x_4 + b$

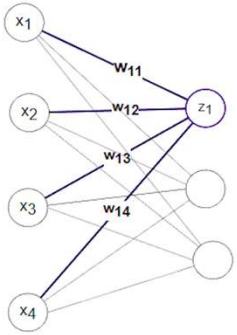
Calculate Activation Function

z3 sigmoid a3

a3 = sigmoid(z3)

A Simple Linear Layer – GPU execution?

- **Memory Requirements:**
  - Parameters (weights, biases)
    - **12 + 3 = 15**



4 Input Neurons, 3 output Neurons

TIME

Thread 1

Calculate Weighted Sum

Thread 1

$z_1 = w_{11} * x_1 + w_{12} * x_2 + w_{13} * x_3 + w_{14} * x_4 + b$

Calculate Activation Function

z1 sigmoid a1

a1 = sigmoid(z1)

Thread 2

Calculate Weighted Sum

Thread2

$z_2 = w_{21} * x_1 + w_{22} * x_2 + w_{23} * x_3 + w_4 * x_4 + b$

Calculate Activation Function

z2 sigmoid a2

a2 = sigmoid(z2)

Thread 3

Calculate Weighted Sum

Thread3

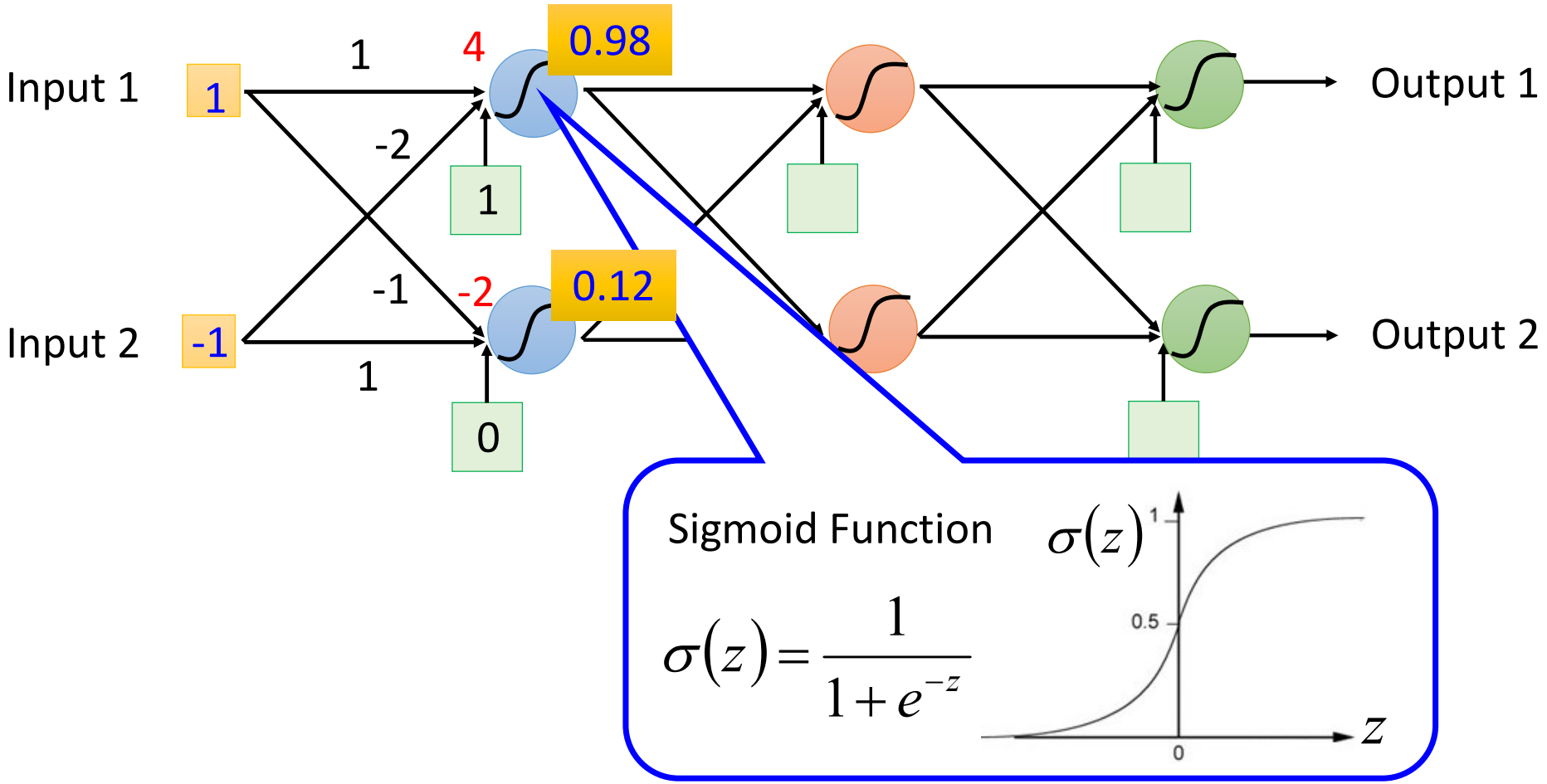
$z_3 = w_{31} * x_1 + w_{32} * x_2 + w_{33} * x_3 + w_{34} * x_4 + b$

Calculate Activation Function

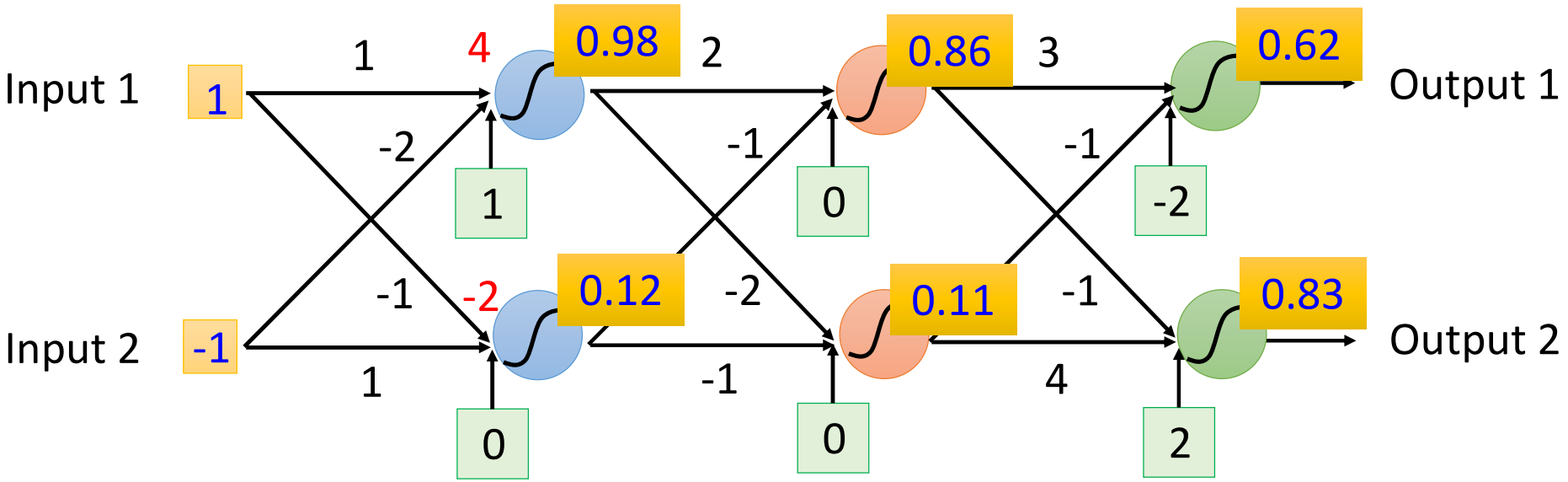
z3 sigmoid a3

a3 = sigmoid(z3)

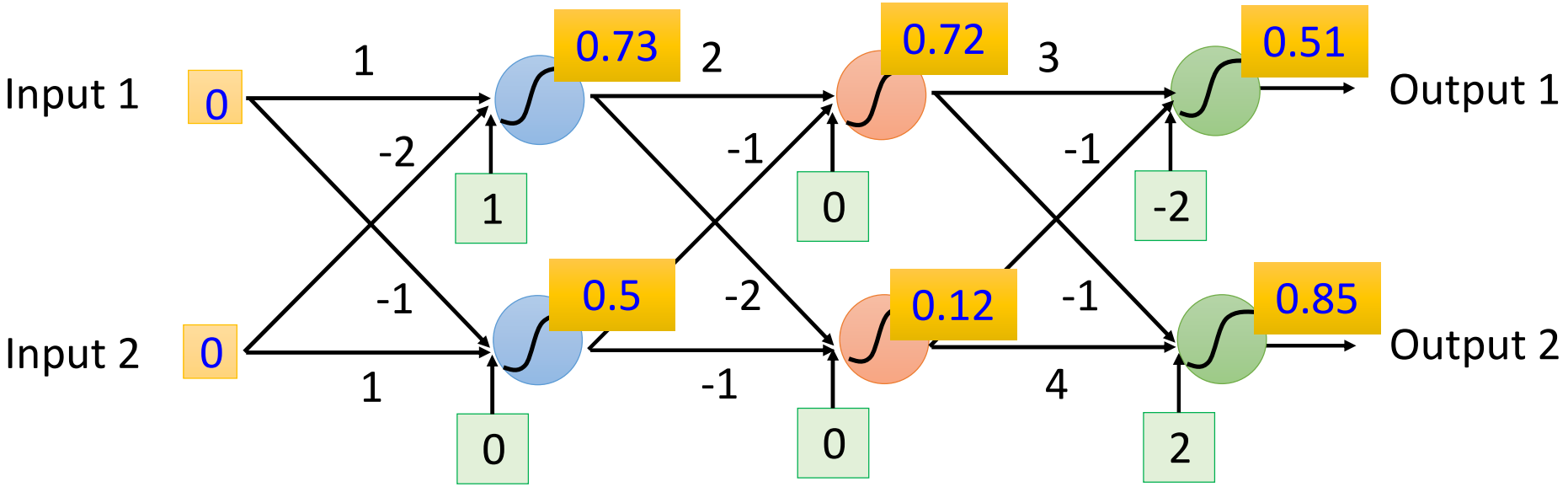
Example of Neural Network -- Prediction



Example of Neural Network -- Prediction



Example of Neural Network -- Prediction

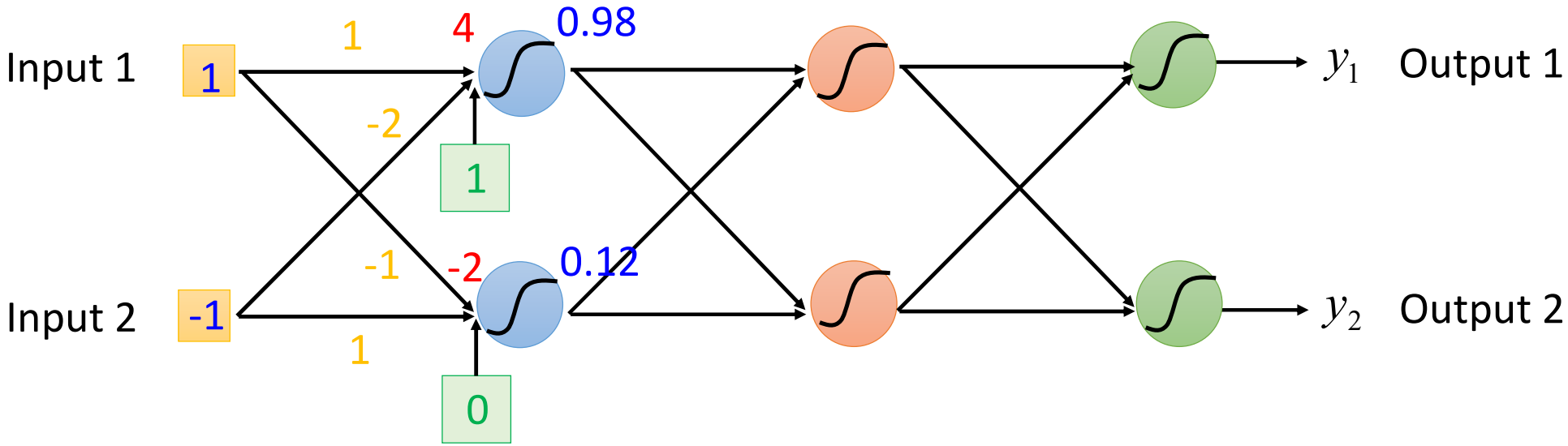


$f: R^2 \rightarrow R^2$

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right) = \begin{bmatrix}0.62\\0.83\end{bmatrix} \quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right) = \begin{bmatrix}0.51\\0.85\end{bmatrix}$$

Different parameters define different function

Matrix Operation

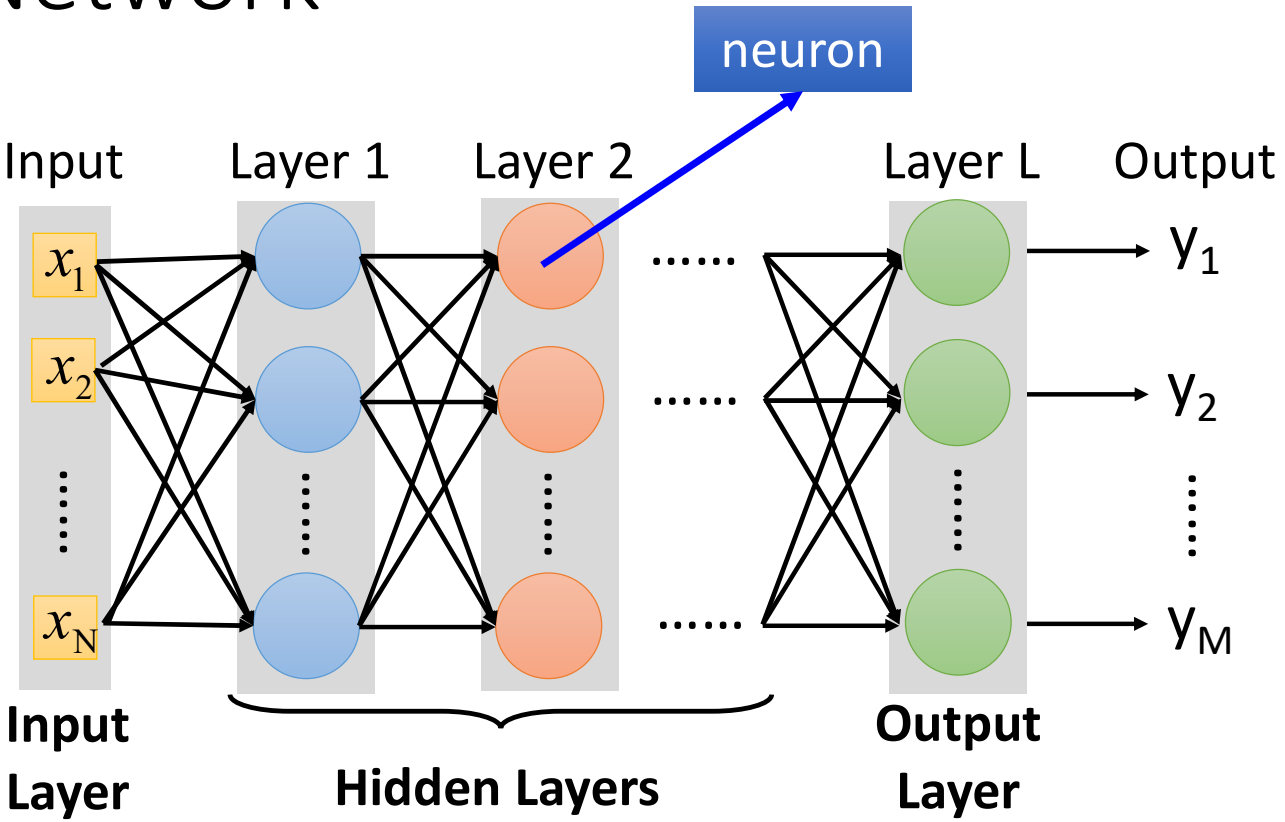


$$\sigma\left(\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

$$\begin{bmatrix}4\\ -2\end{bmatrix}$$

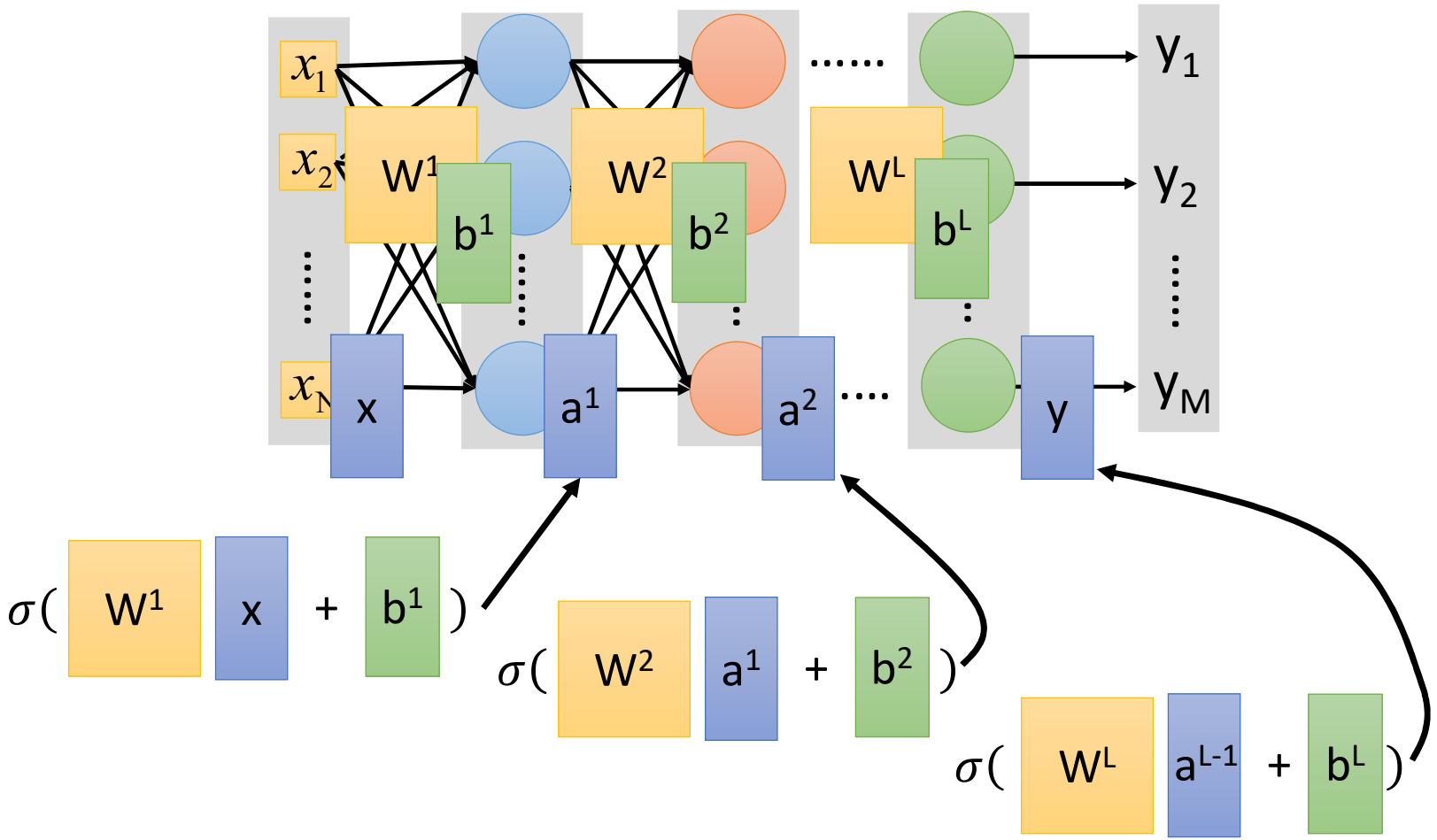
$$\sigma(\ W\ \ x\ +\ b\ )=\ a$$

Neural Network

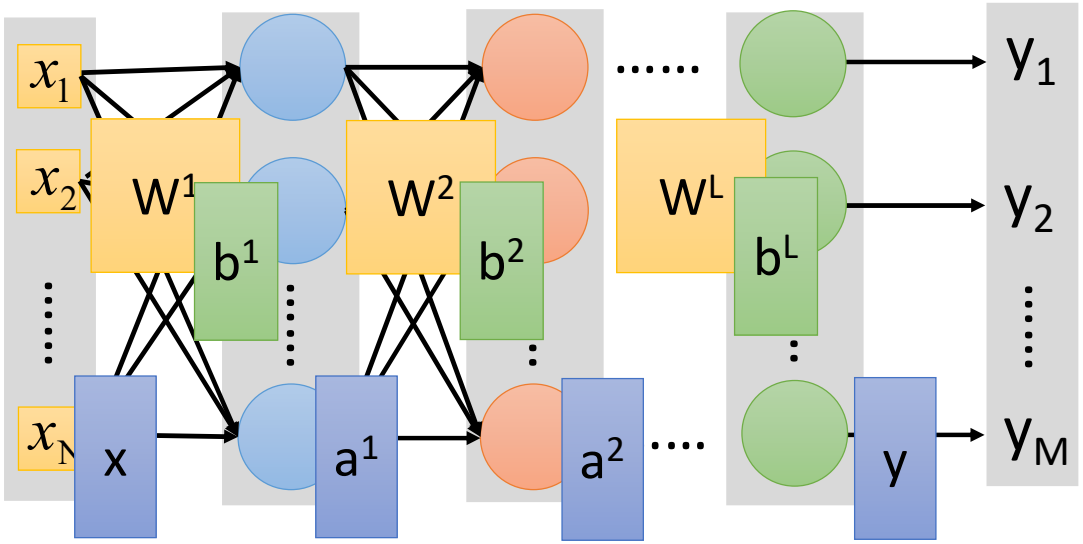


Deep means many hidden layers

Neural Network



Neural Network



$$y = f(x)$$

Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$