بسم الله الرحمن الرحيم

In the Name of Allah, the Most Gracious, the Most Merciful

قال الم ١٦ | طٰہٰ ٢٠

قَالَ رَبِّ اشْرَحْ لِي صَدْرِي ﴿٢٥﴾

٢٥۔ (موسیٰ علیہ السلام نے) عرض کیا: اے میرے رب! میرے لئے میرا سینہ کشادہ فرما دے۔

وَيَسِّرْ لِي أَمْرِي ﴿٢٦﴾

٢٦۔ اور میرا کار (رسالت) میرے لئے آسان فرما دے۔

وَاحْلُلْ عُقْدَةً مِّن لِّسَانِي ﴿٢٧﴾

٢٧۔ اور میری زبان کی گرہ کھول دے۔

يَفْقَهُوا قَوْلِي ﴿٢٨﴾

٢٨۔ کہ لوگ میری بات (آسانی سے) سمجھ سکیں۔

Surah Taha with Urdu Translation

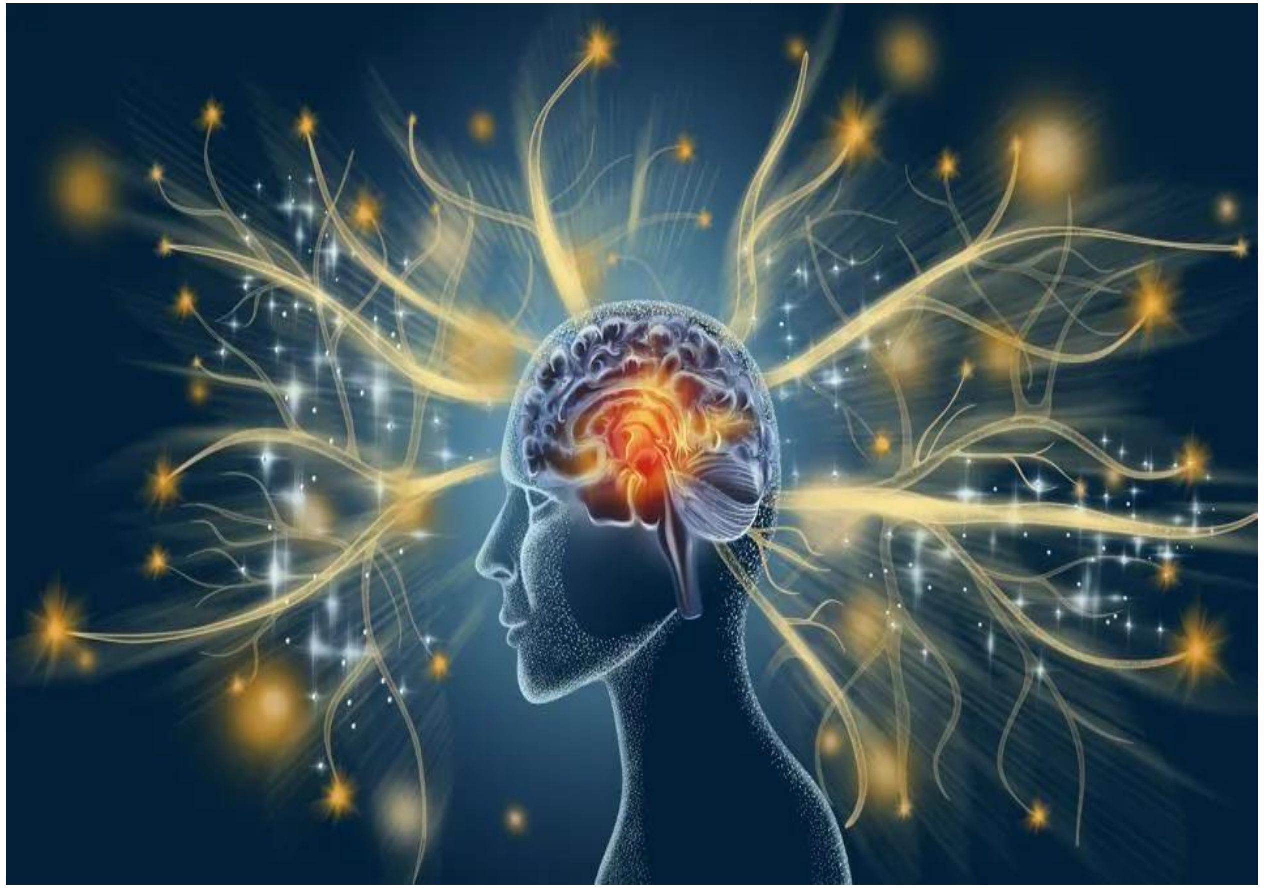
رَبِّ إِنِّي لِمَا أَنزَلْتَ إِلَيَّ مِنْ خَيْرٍ فَقِيرٌ

اے میرے رب! بے شک میں، جو بھلائی
بھی تو میری طرف نازل فرمائے، اس کا محتاج ہوں۔

﴿سورۃ القصص: آیت نمبر 24﴾

CS4152: Deep Learning and Neural Networks

Lecture 3 (Introduction to Neural Networks)



Dr. Jameel Ahmad

Assistant Professor

Jameel.ahmad@umt.edu.p k

Department of Computer Science

School of Systems and Technology

University of Management and Technology, Lahore

Lecture 3

Neural Networks
An Overview
Jameel Ahmad, PhD

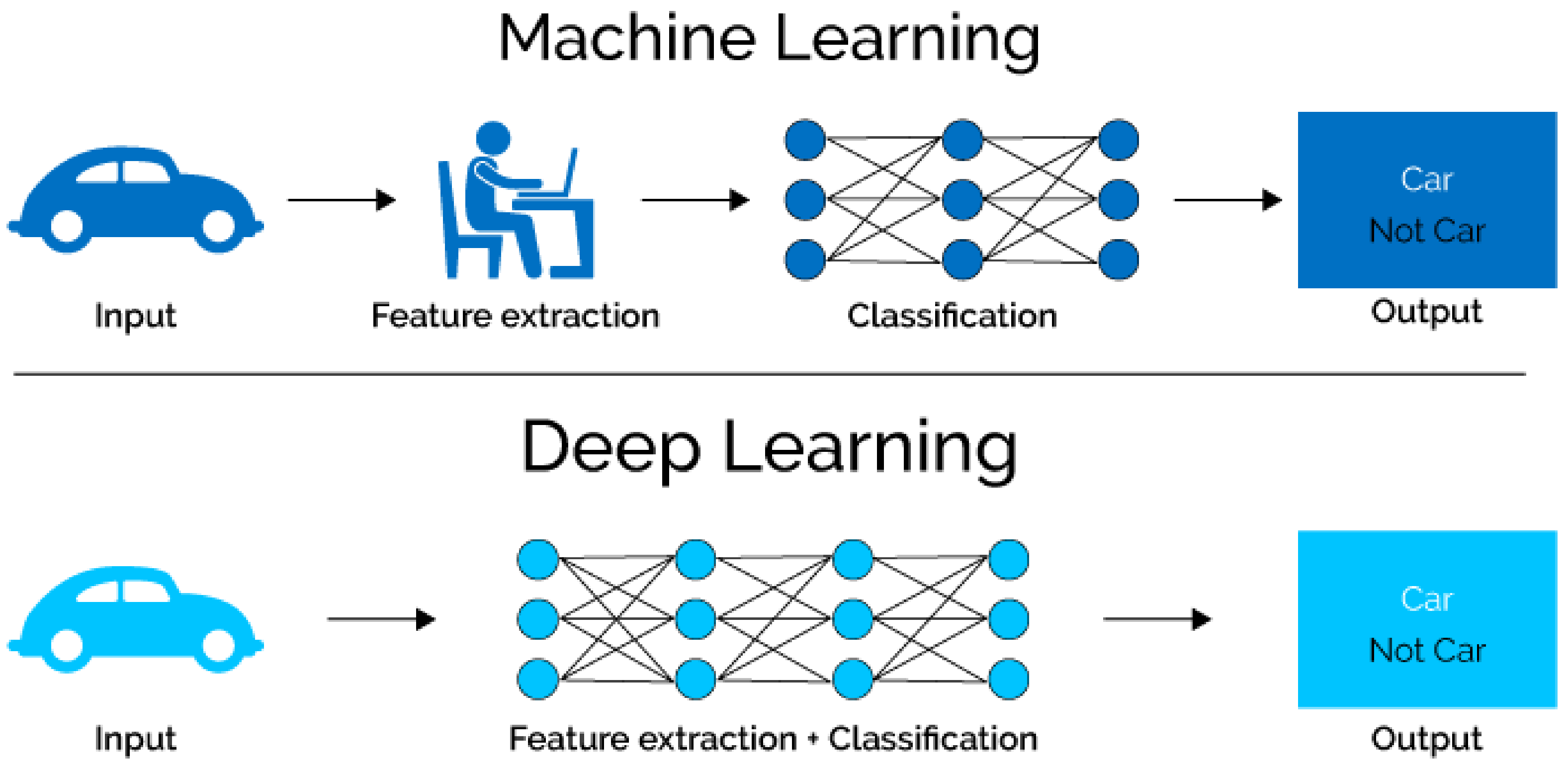
Lecture Outline

- Elements of neural networks (NNs)
  - Activation functions
  - Single and Multilayer Perceptron

ML vs. Deep Learning (DL)

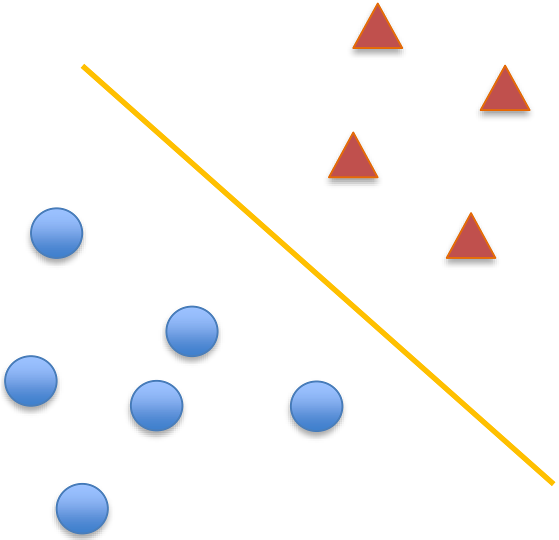
Difference between Machine Learning and Deep Learning

- ***Deep learning*** (DL) is a machine learning subfield that uses multiple layers for learning data representations
  - DL is exceptionally effective at learning patterns



Neural Network

Logistic Regression

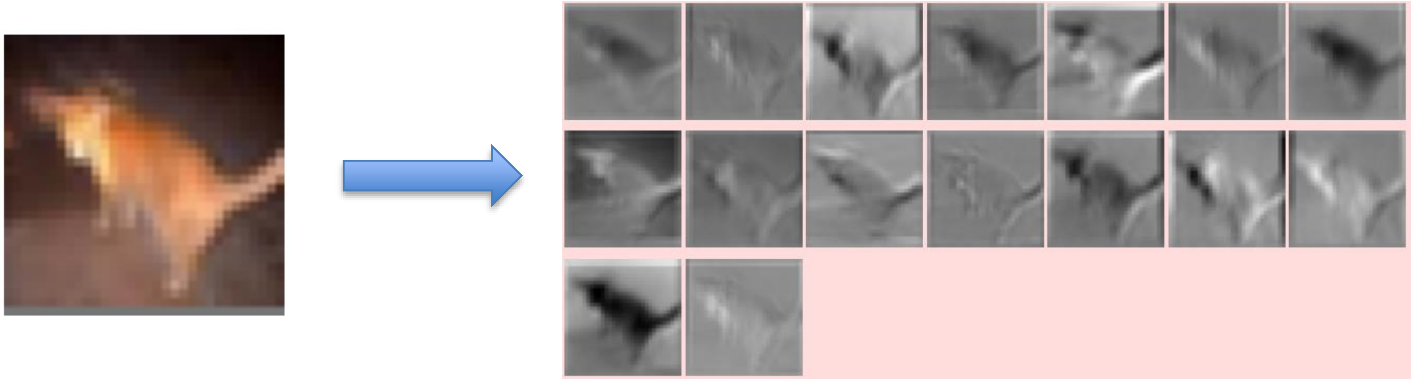


Neural Networks

Neural Network

- Non-linear score function $\boldsymbol{f} = \ldots(\max(\mathbf{0}, \boldsymbol{W}_{\mathbf{1}}\boldsymbol{x}))$

On CIFAR-10



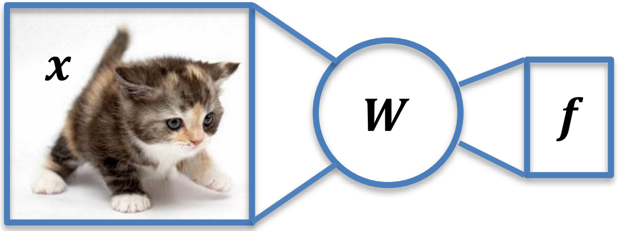
Visualizing activations of the first layer.

Source: ConvNetJS

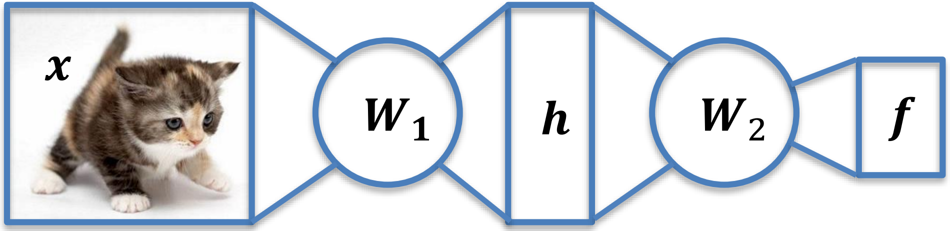
Neural Network

1-layer network: $\boldsymbol{f} = \boldsymbol{W}\boldsymbol{x}$

2-layer network: $\boldsymbol{f} = \boldsymbol{W_2} \max(\boldsymbol{0}, \boldsymbol{W_1}\boldsymbol{x})$



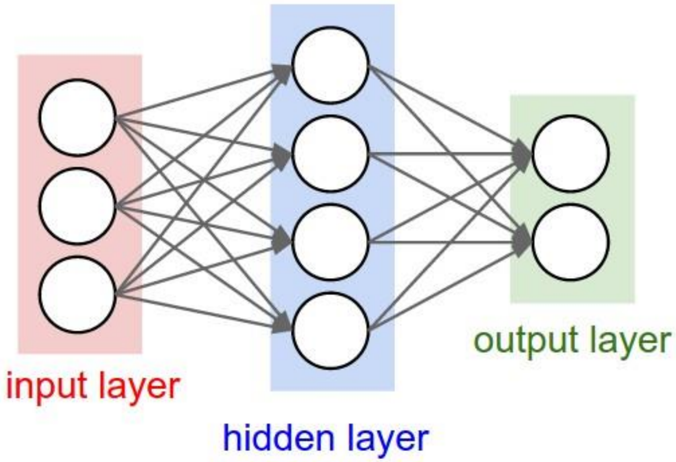
$128 \times 128 = 16384$ 10



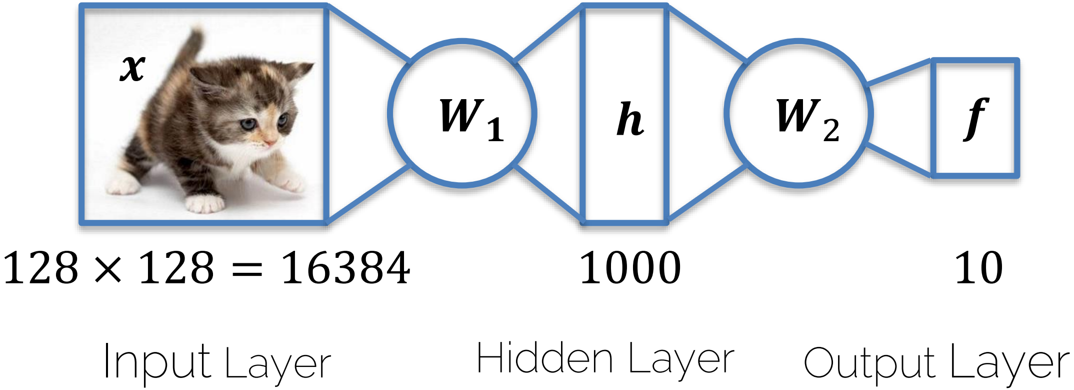
$128 \times 128 = 16384$ 1000 10

Why is this structure useful?

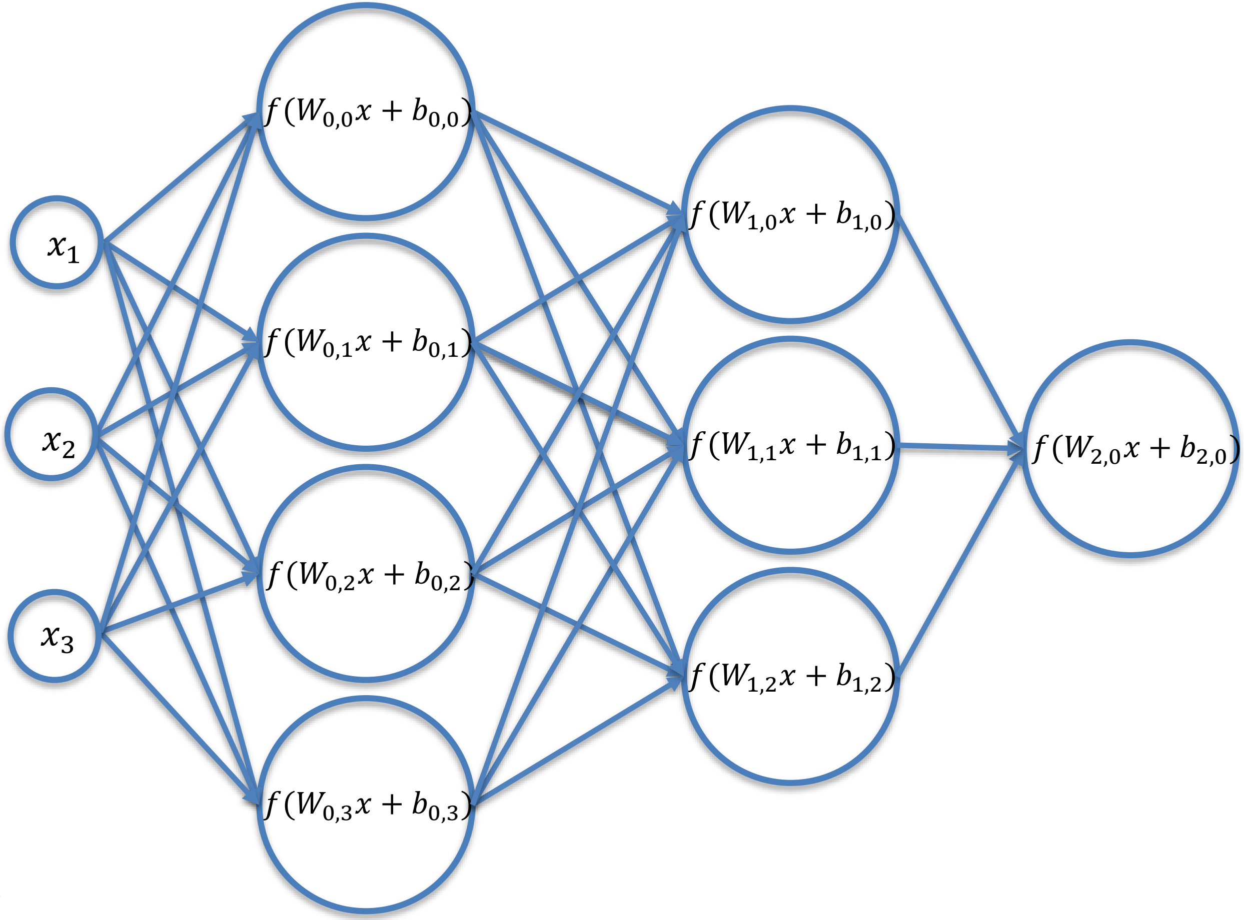
Neural Network



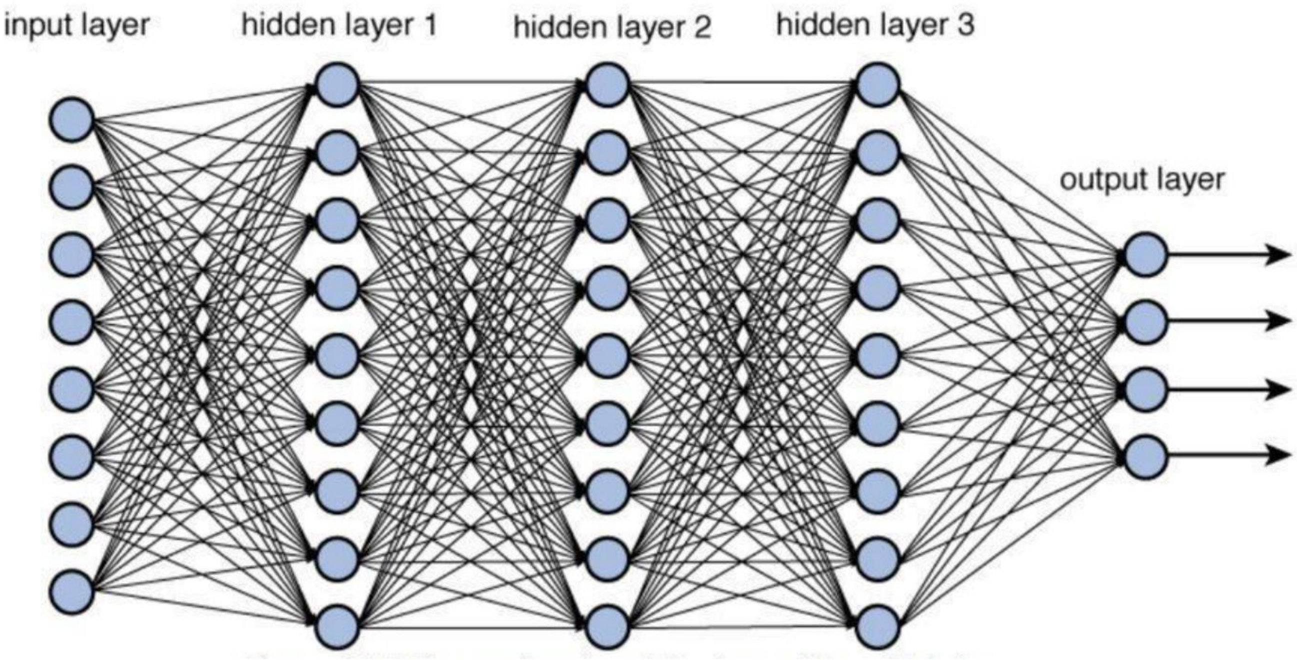
2-layer network: $\boldsymbol{f} = \boldsymbol{W_2} \max(\boldsymbol{0}, \boldsymbol{W_1 x})$



Net of Artificial Neurons



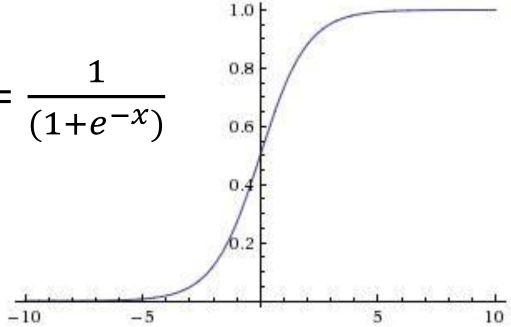
Neural Network



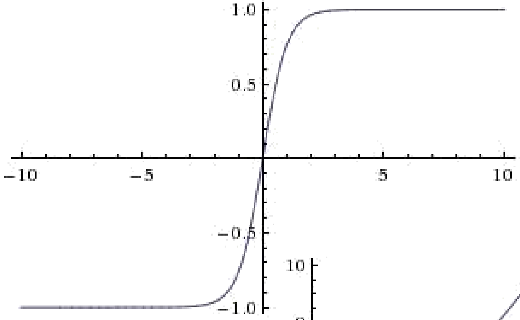
Source: https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

Activation Functions

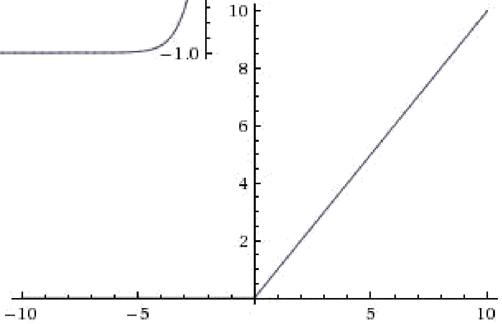
Sigmoid: $\sigma(x) = \frac{1}{(1+e^{-x})}$



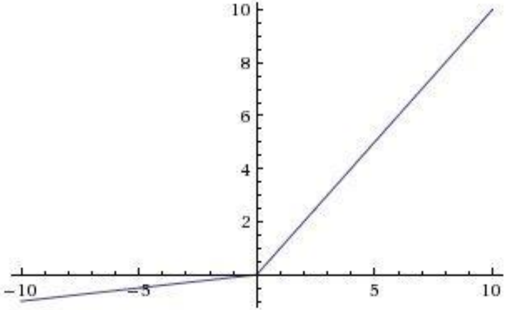
tanh: $\tanh(x)$



ReLU: $\max(0, x)$



Leaky ReLU: $\max(0.1x, x)$



Parametric ReLU: $\max(\alpha x, x)$

Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$

Neural Network

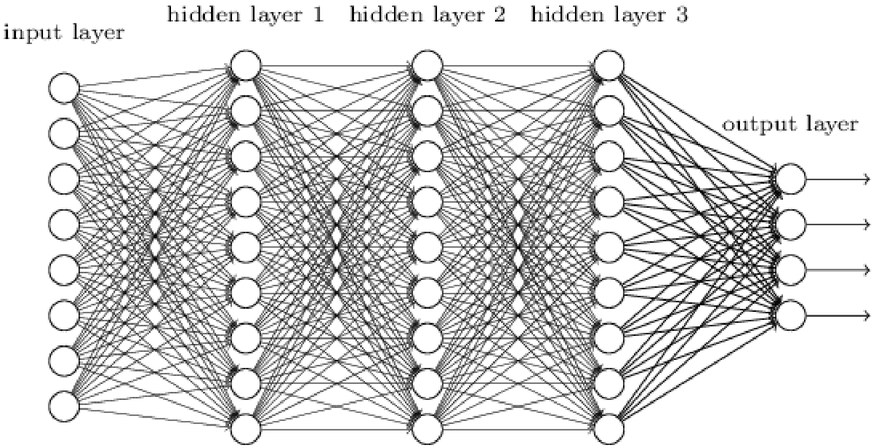
$$\boldsymbol{f} = \boldsymbol{W_3} \cdot (\boldsymbol{W_2} \cdot (\boldsymbol{W_1} \cdot \boldsymbol{x}))$$

Why activation functions?

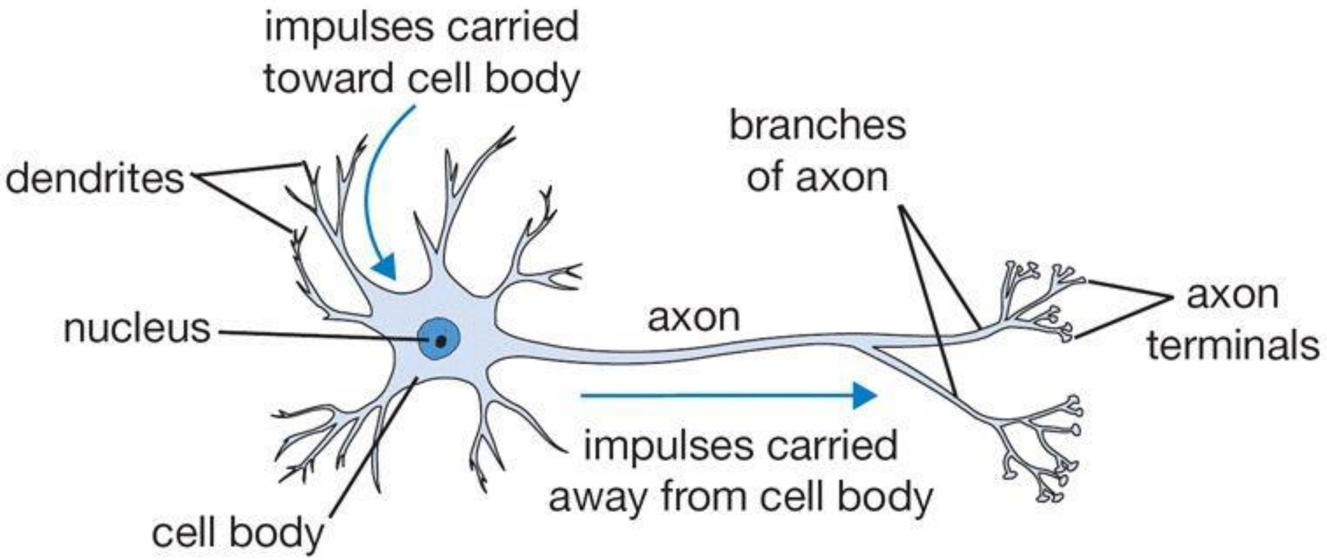
Simply concatenating linear layers would be so much cheaper...

Neural Network

Why organize a neural network into layers?

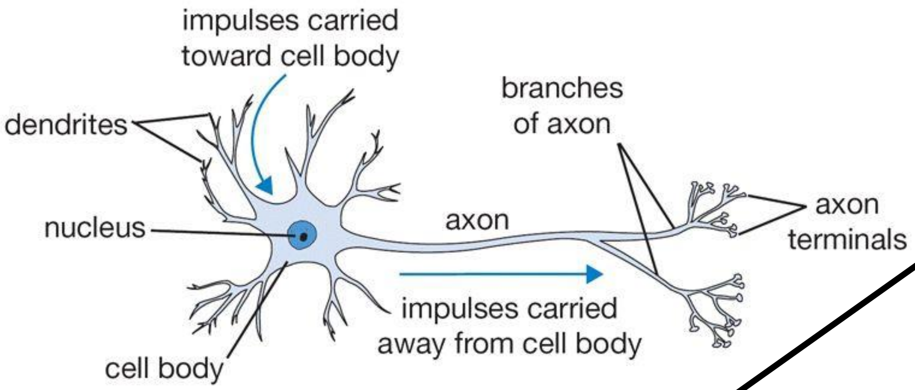


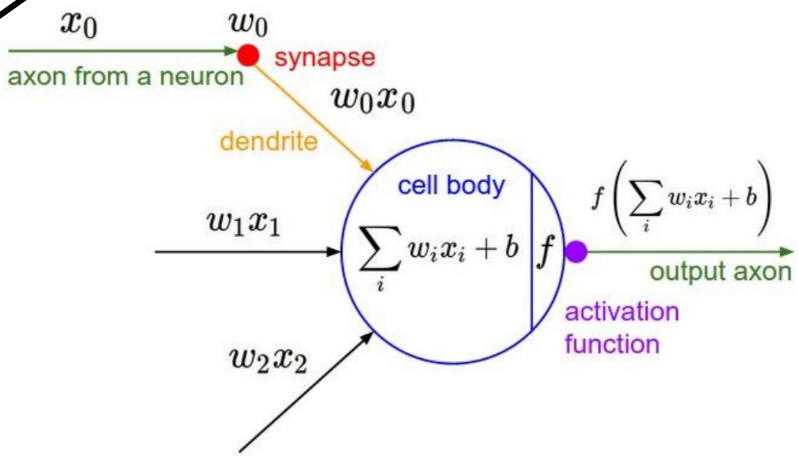
Biological Neurons



Credit: Stanford CS 231n

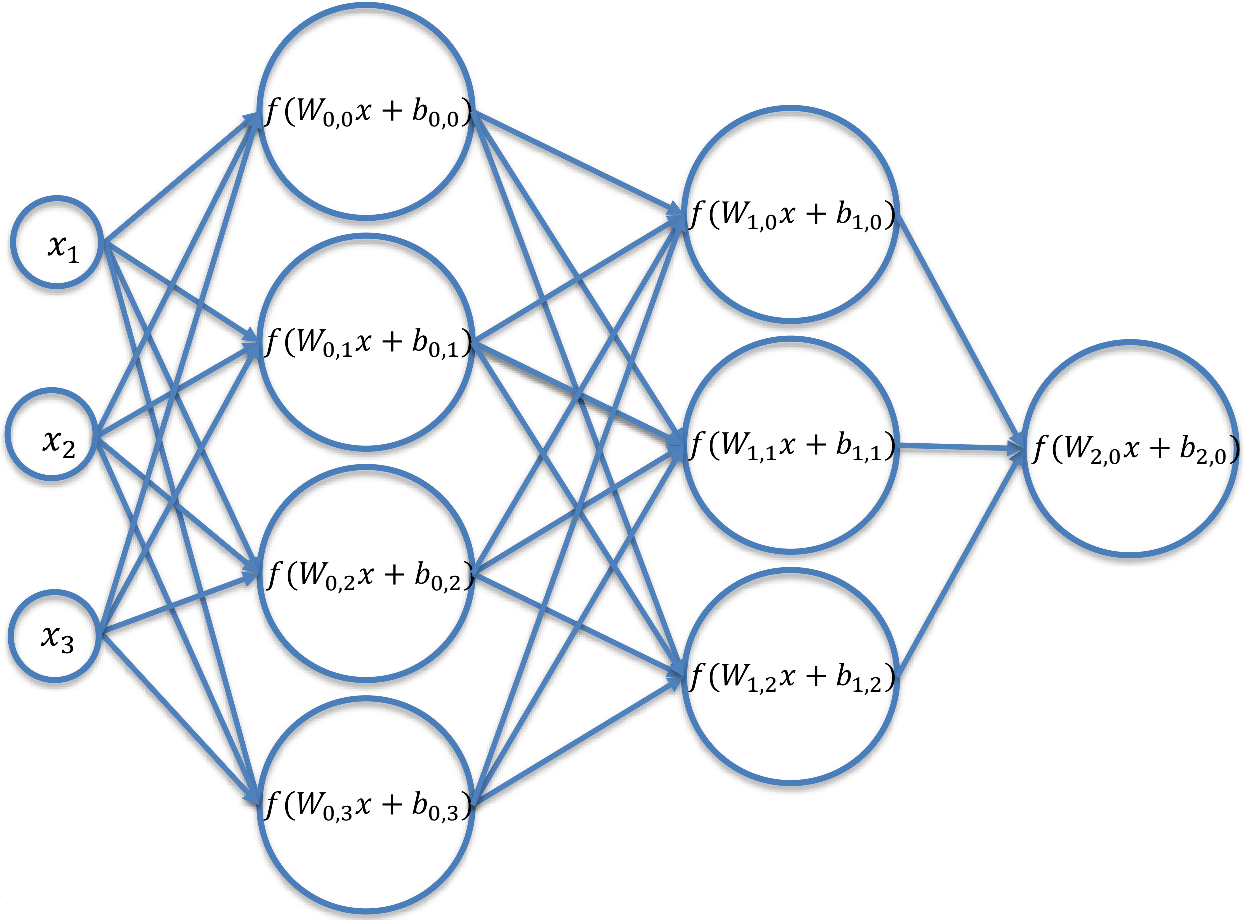
Biological Neurons





Credit: Stanford CS 231n

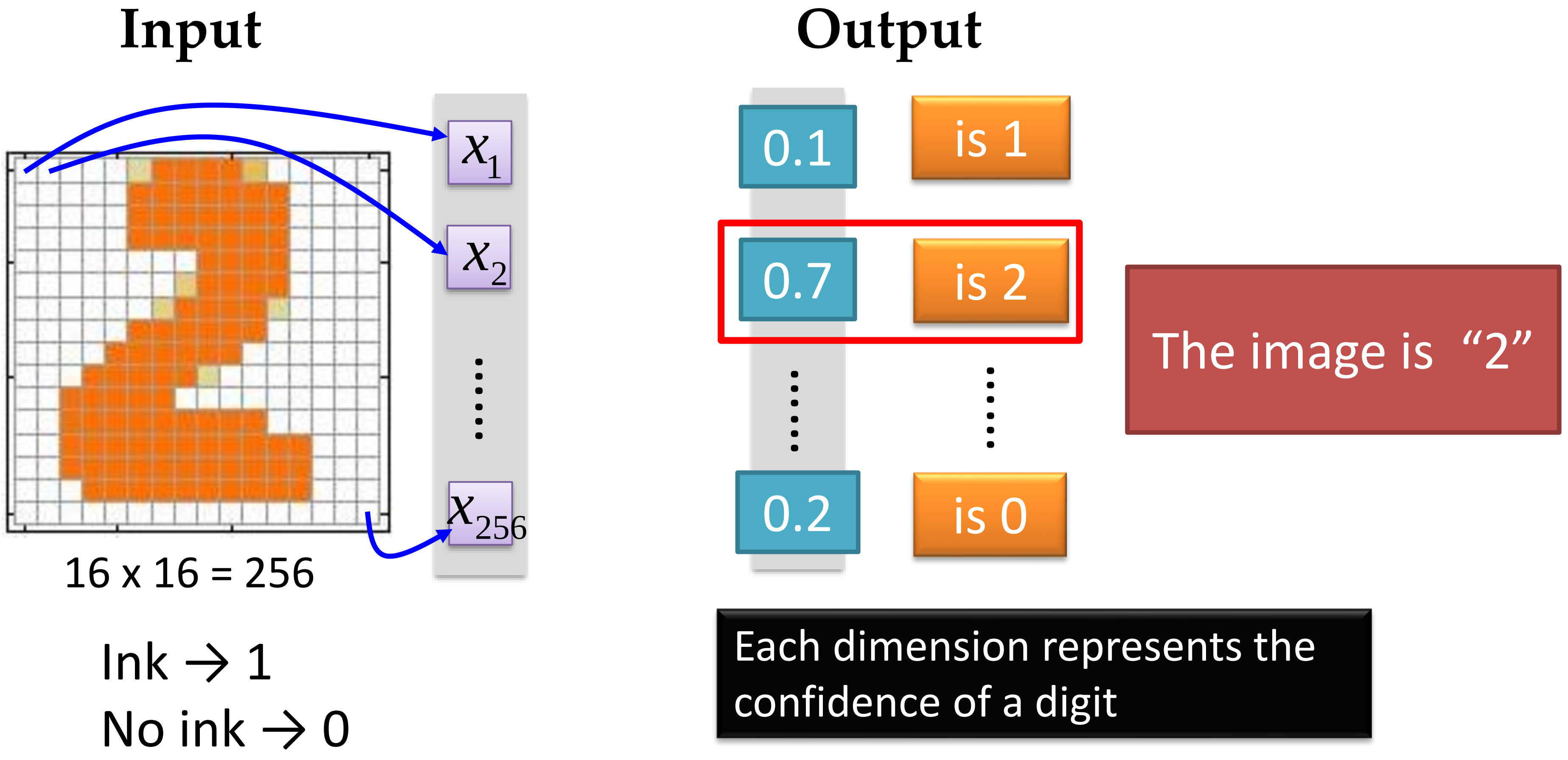
Artificial Neural Network



Neural Networks

Introduction to Neural Networks

- Handwritten digit recognition (MNIST dataset)
  - The intensity of each pixel is considered an input element
  - Output is the class of the digit

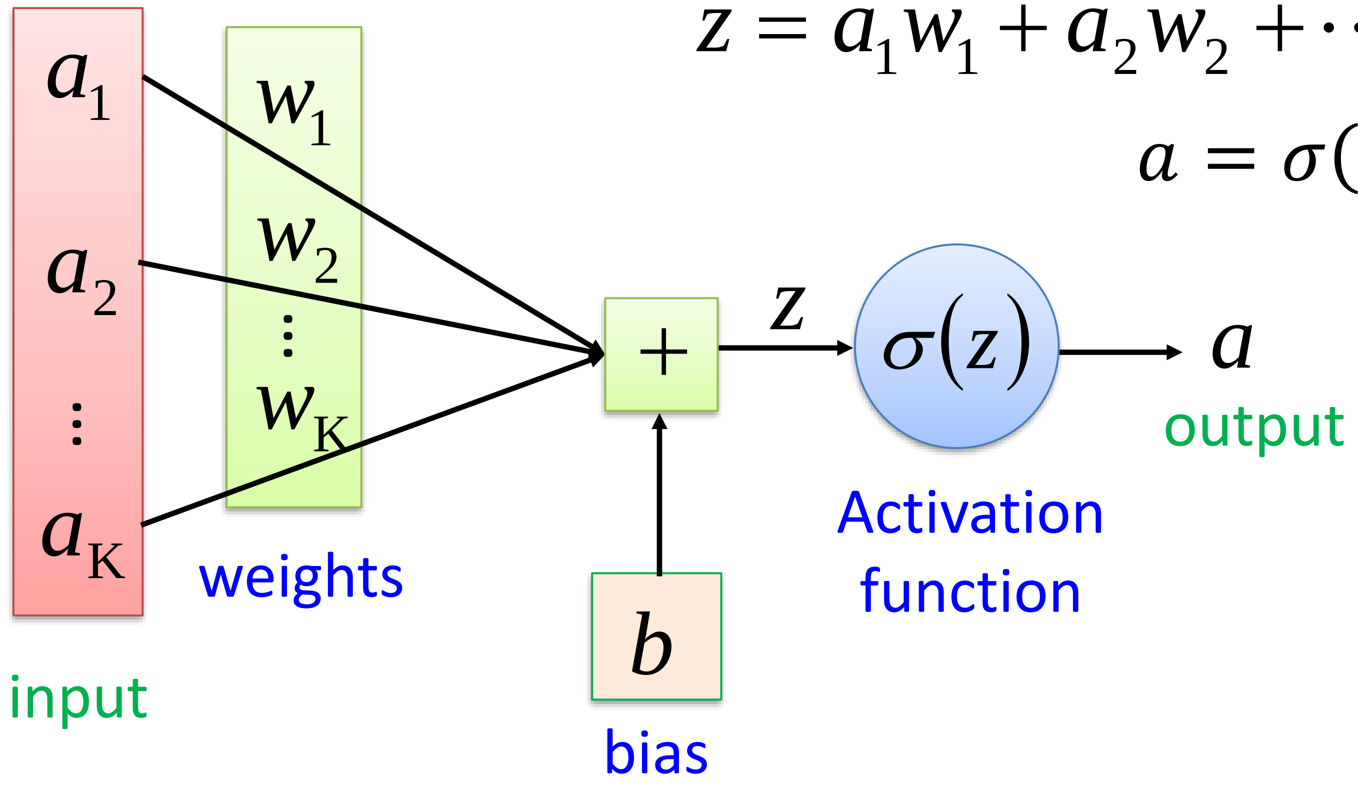


Elements of Neural Networks

- NNs consist of hidden layers with neurons (i.e., computational units)
- A single neuron maps a set of inputs into an output number, or $f: R^K \to R$

$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$

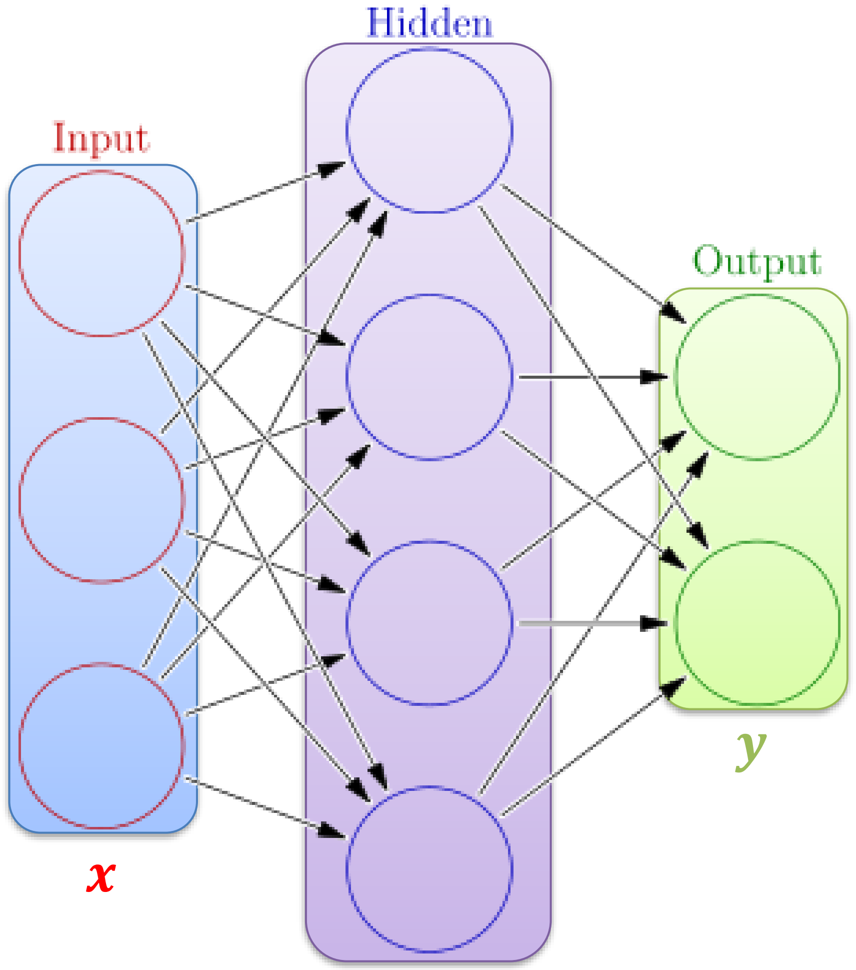
$$a = \sigma(z)$$



Elements of Neural Networks

Introduction to Neural Networks

- A NN with one hidden layer and one output layer



Weights Biases

$$\text{hidden layer } h = \sigma(\mathbf{W}_1 x + b_1)$$

$$\text{output layer } y = \sigma(W_2 h + b_2)$$

Activation functions

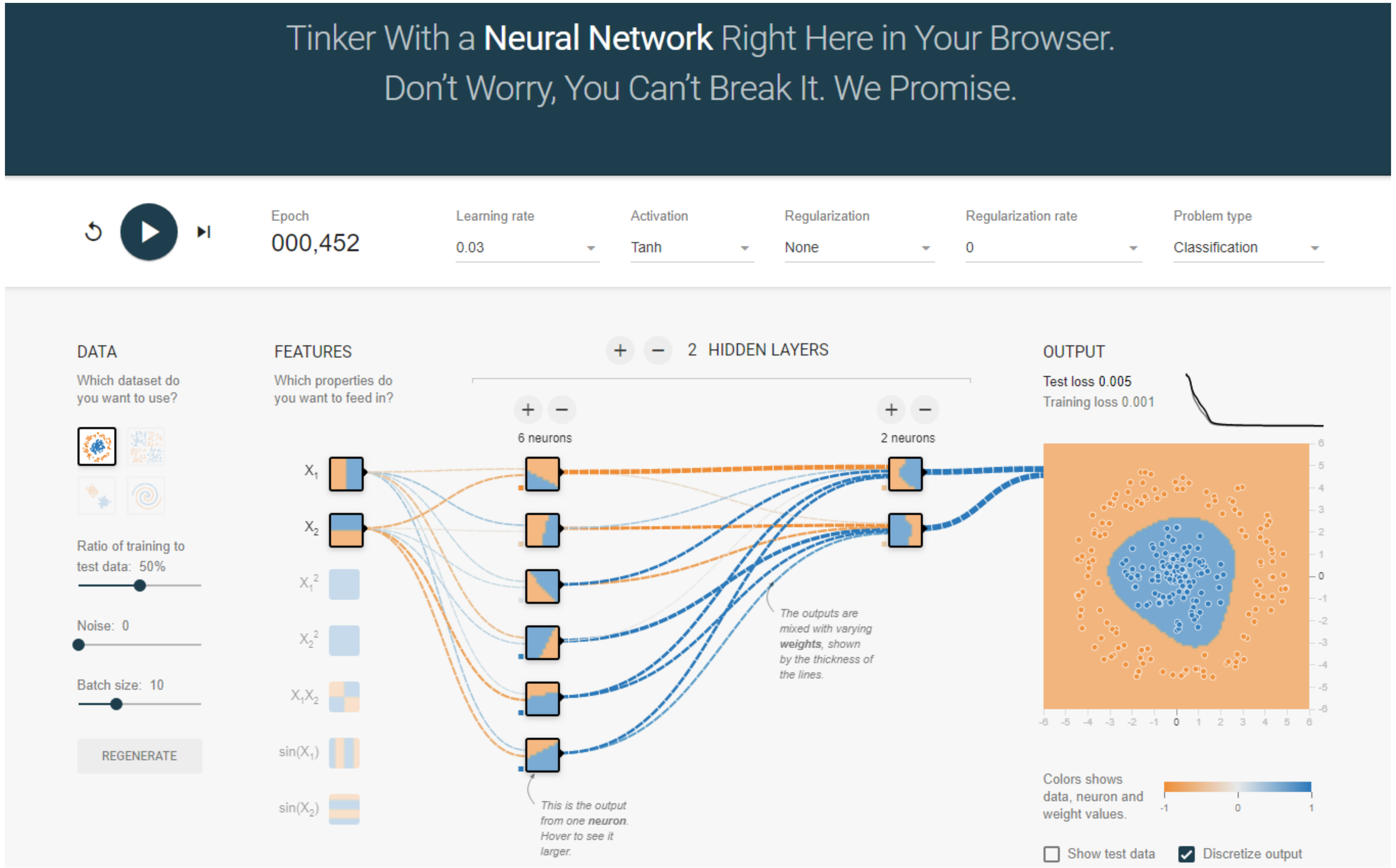
4 + 2 = 6 neurons (not counting inputs)
[3 × 4] + [4 × 2] = 20 weights
4 + 2 = 6 biases

26 learnable parameters

Elements of Neural Networks

Introduction to Neural Networks

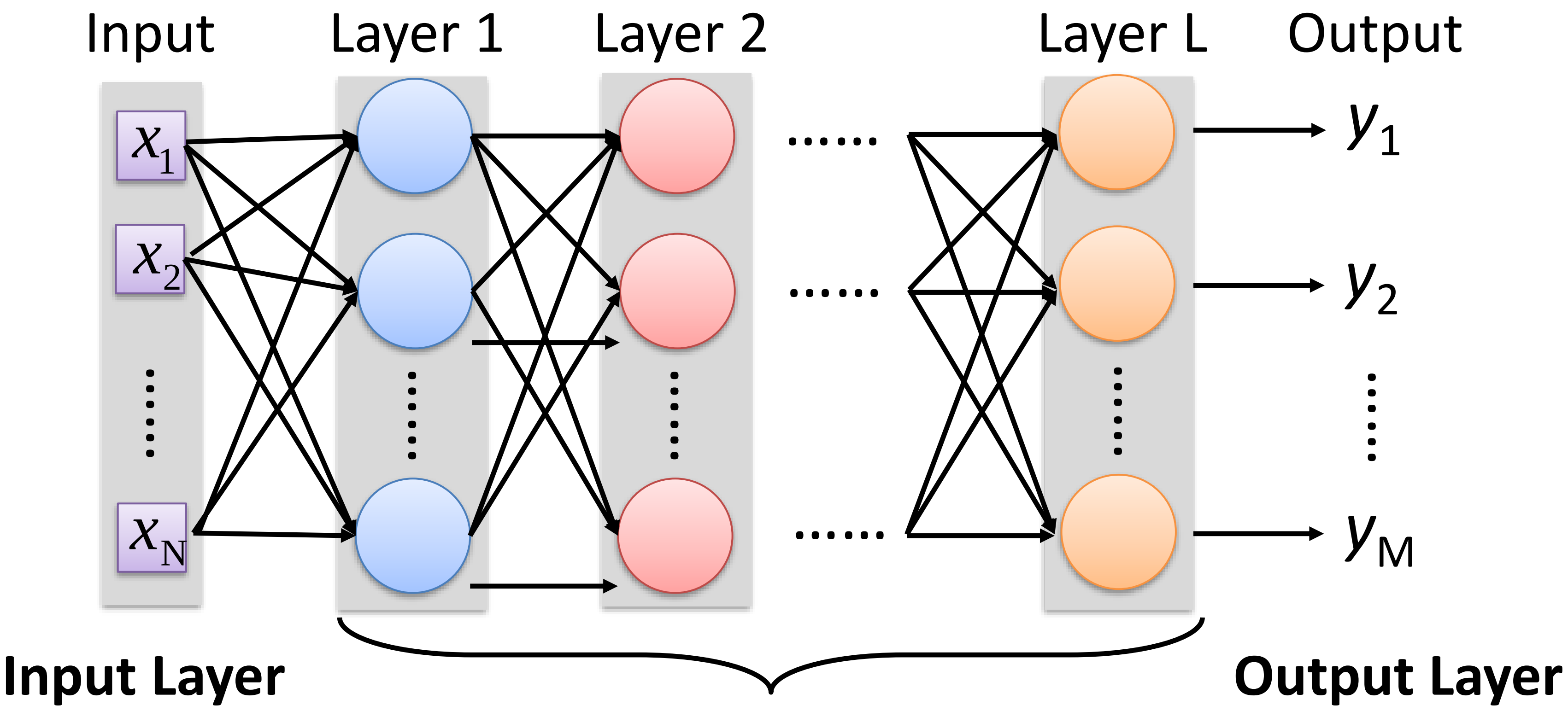
- A neural network playground [link]



Elements of Neural Networks

Introduction to Neural Networks

- Deep NNs have many hidden layers
  - Fully-connected (dense) layers (a.k.a. Multi-Layer Perceptron or MLP)
  - Each neuron is connected to all neurons in the succeeding layer

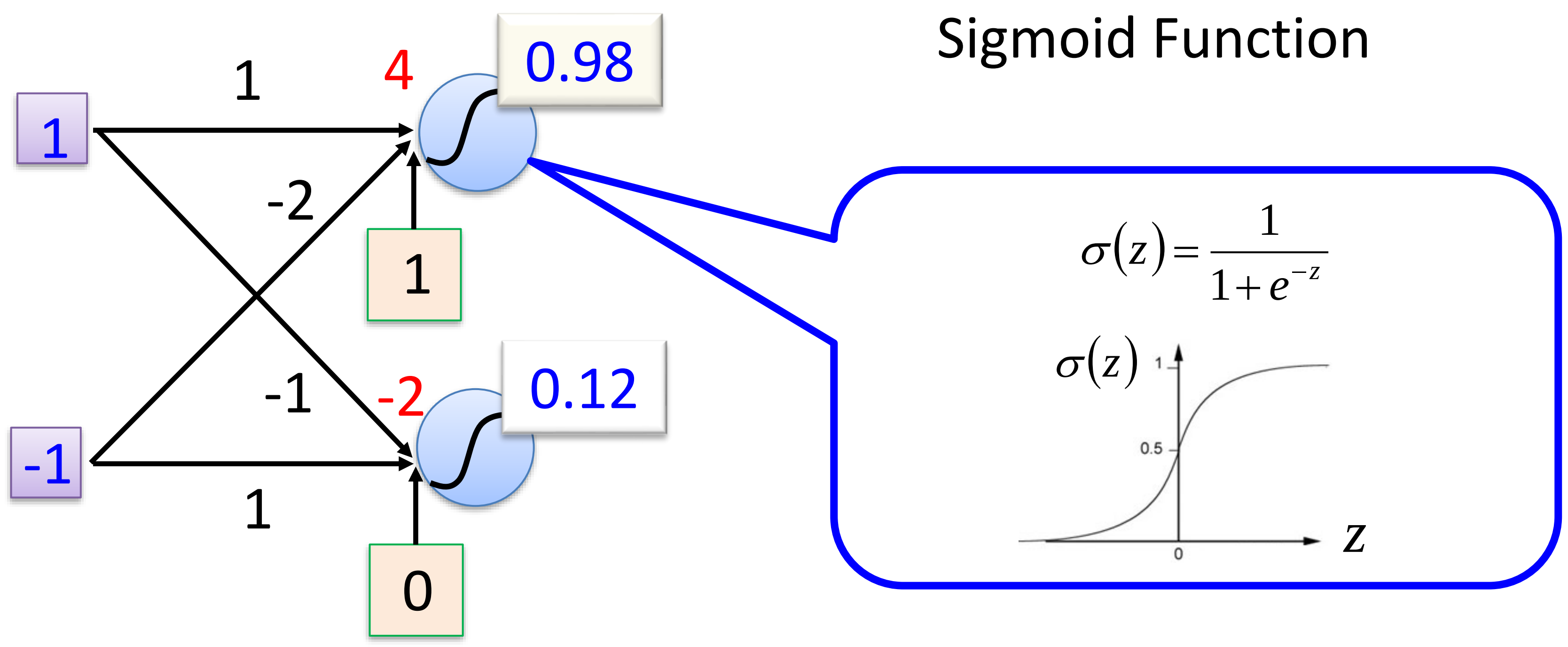


Elements of Neural Networks

Introduction to Neural Networks

- A simple network, toy example

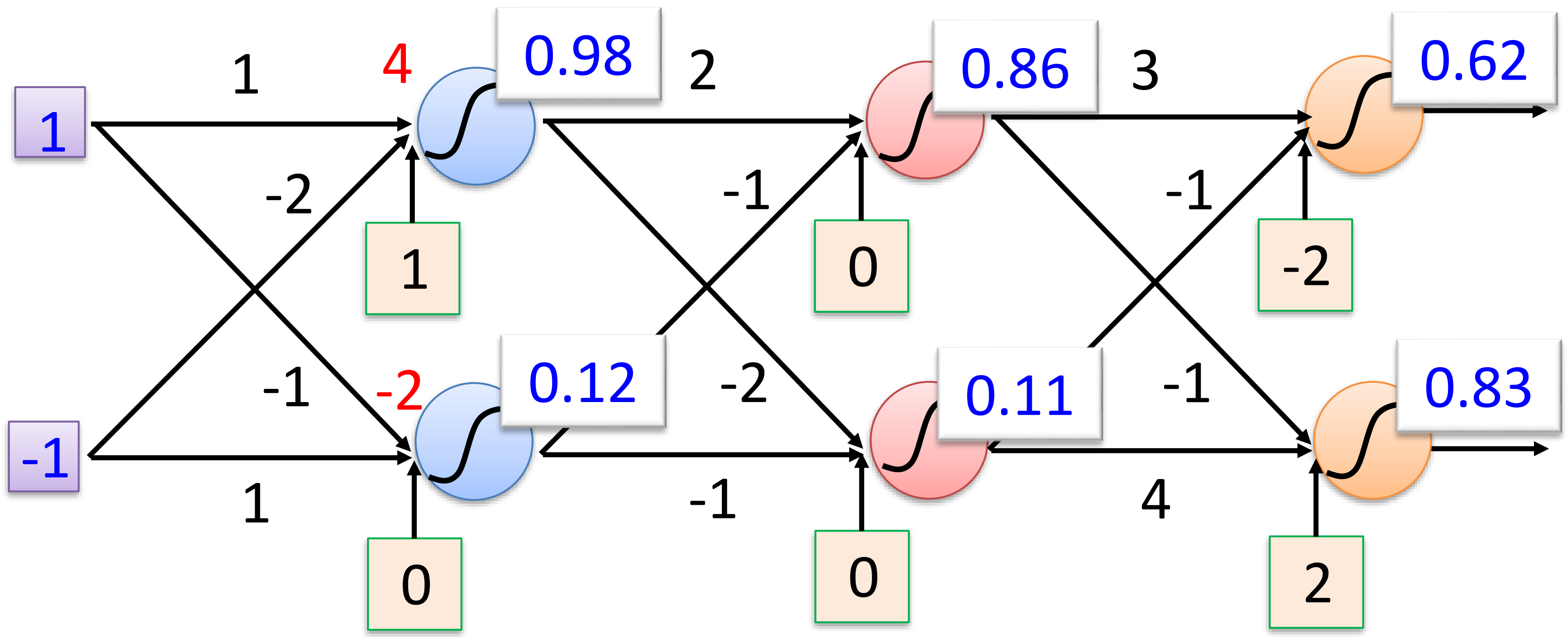
$$(1 \cdot 1) + (-1) \cdot (-2) + 1 = 4$$



$$1 \cdot (-1) + (-1) \cdot 1 + 0 = \text{-}2$$

Elements of Neural Networks

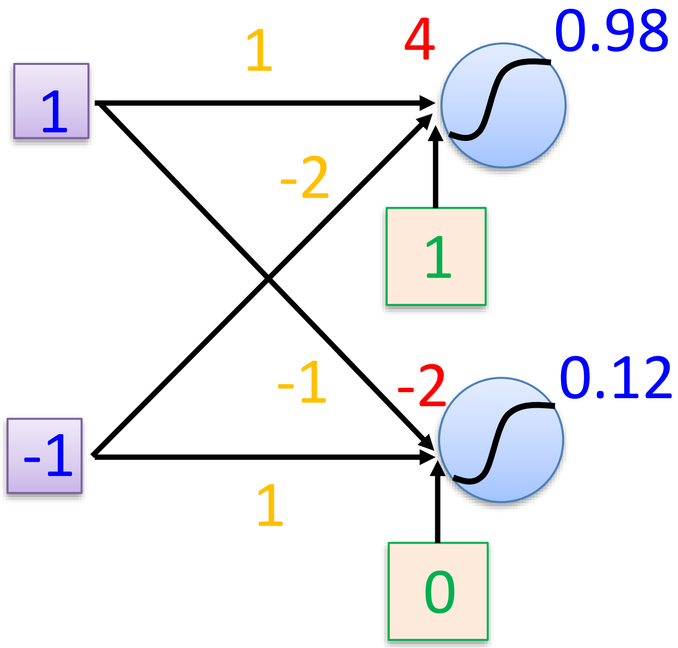
- A simple network, toy example (cont'd)
  - For an input vector $[1 \quad -1]^T$, the output is $[0.62 \quad 0.83]^T$



$$f: R^2 \rightarrow R^2 \qquad f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix}$$

Matrix Operation

- Matrix operations are helpful when working with multidimensional inputs and outputs

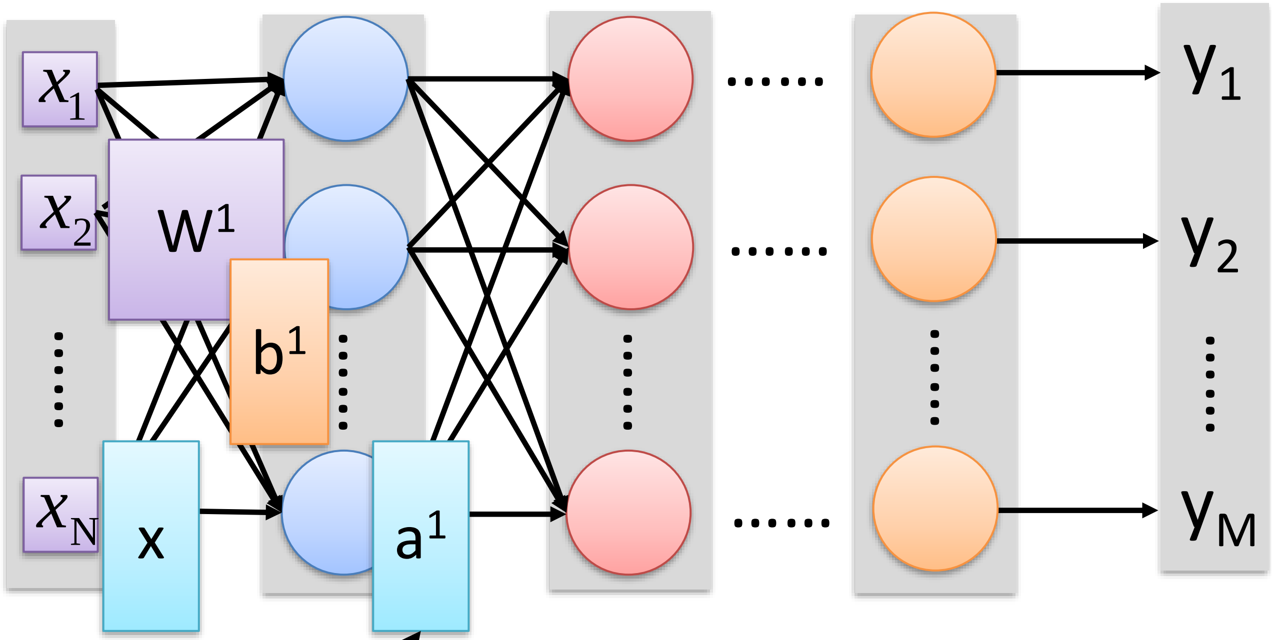


$$\sigma(W \; x + b) = a$$

$$\sigma\Big(\underbrace{\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix}}_{\begin{bmatrix}4\\ -2\end{bmatrix}}\Big) = \begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

Matrix Operation

- Multilayer NN, matrix calculations for the first layer
  - Input vector x, weights matrix W^1, bias vector b^1, output vector a^1

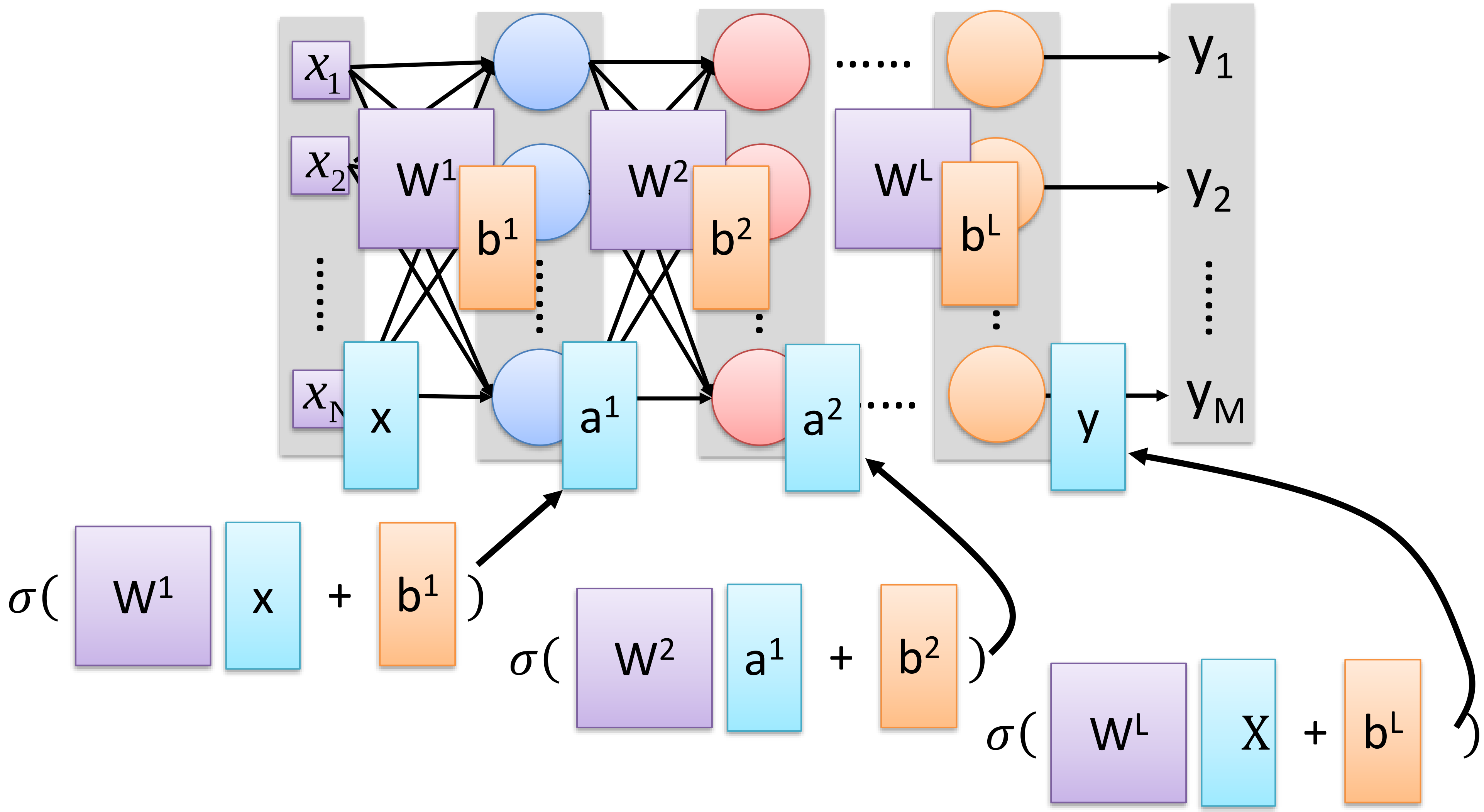


$$a^1 = \sigma(W^1 x + b^1)$$

Matrix Operation

Introduction to Neural Networks

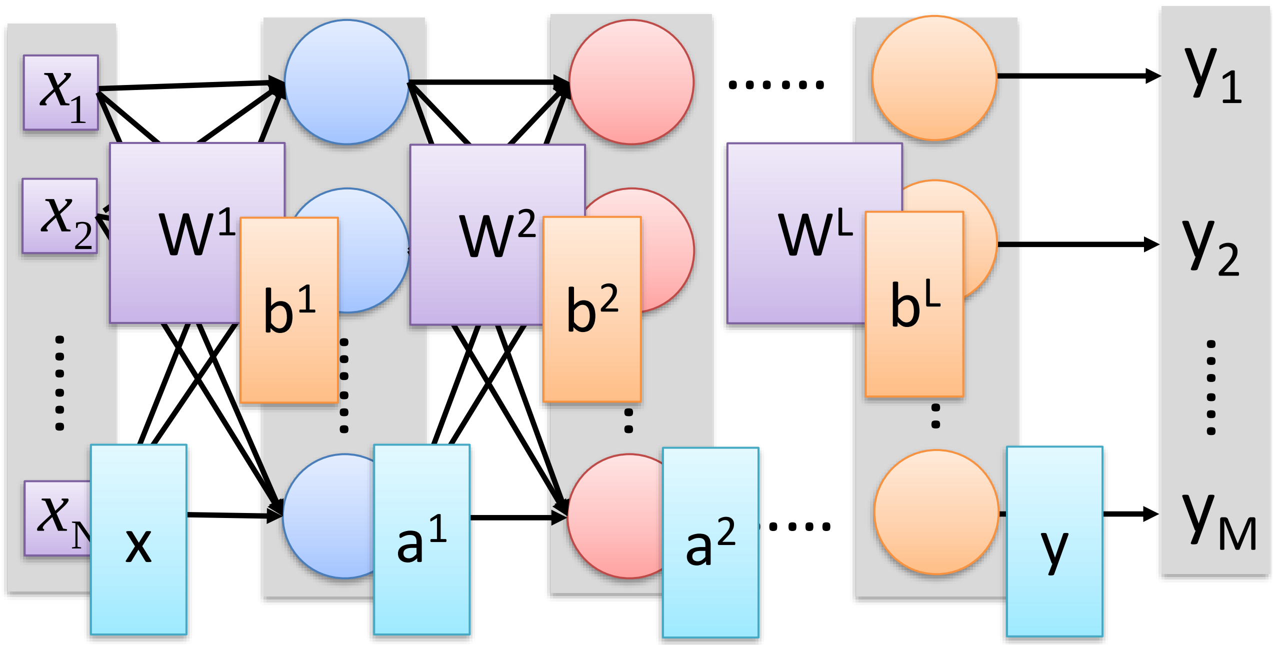
- Multilayer NN, matrix calculations for all layers



Matrix Operation

Introduction to Neural Networks

- Multilayer NN, function f maps inputs x to outputs y, i.e., $y = f(x)$

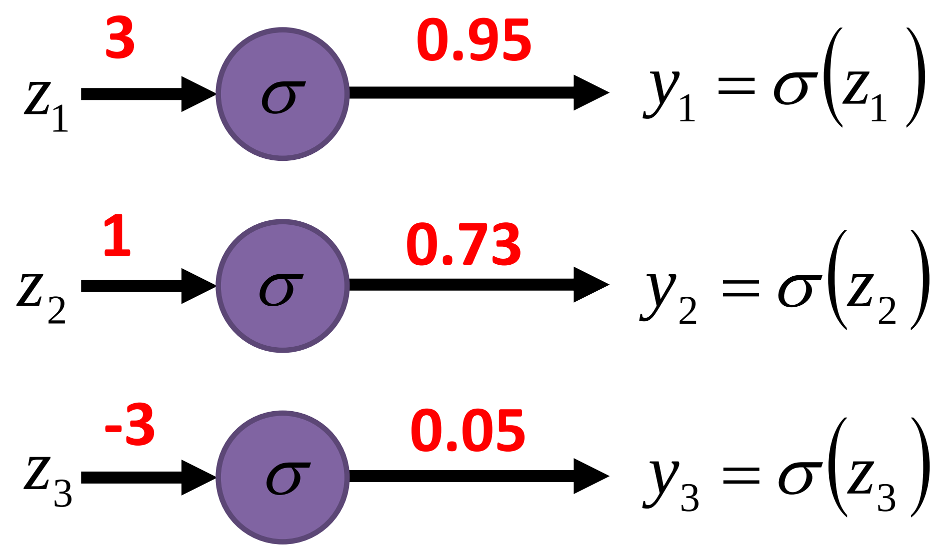


$$y = f(x) = \sigma(W^L \cdots \sigma(W^2 \, \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

Softmax Layer

- In multi-class classification tasks, the output layer is typically a ***softmax layer***
  - I.e., it employs a ***softmax activation function***
  - If a layer with a sigmoid activation function is used as the output layer instead, the predictions by the NN may not be easy to interpret
    - Note that an output layer with sigmoid activations can still be used for binary classification

A Layer with Sigmoid Activations



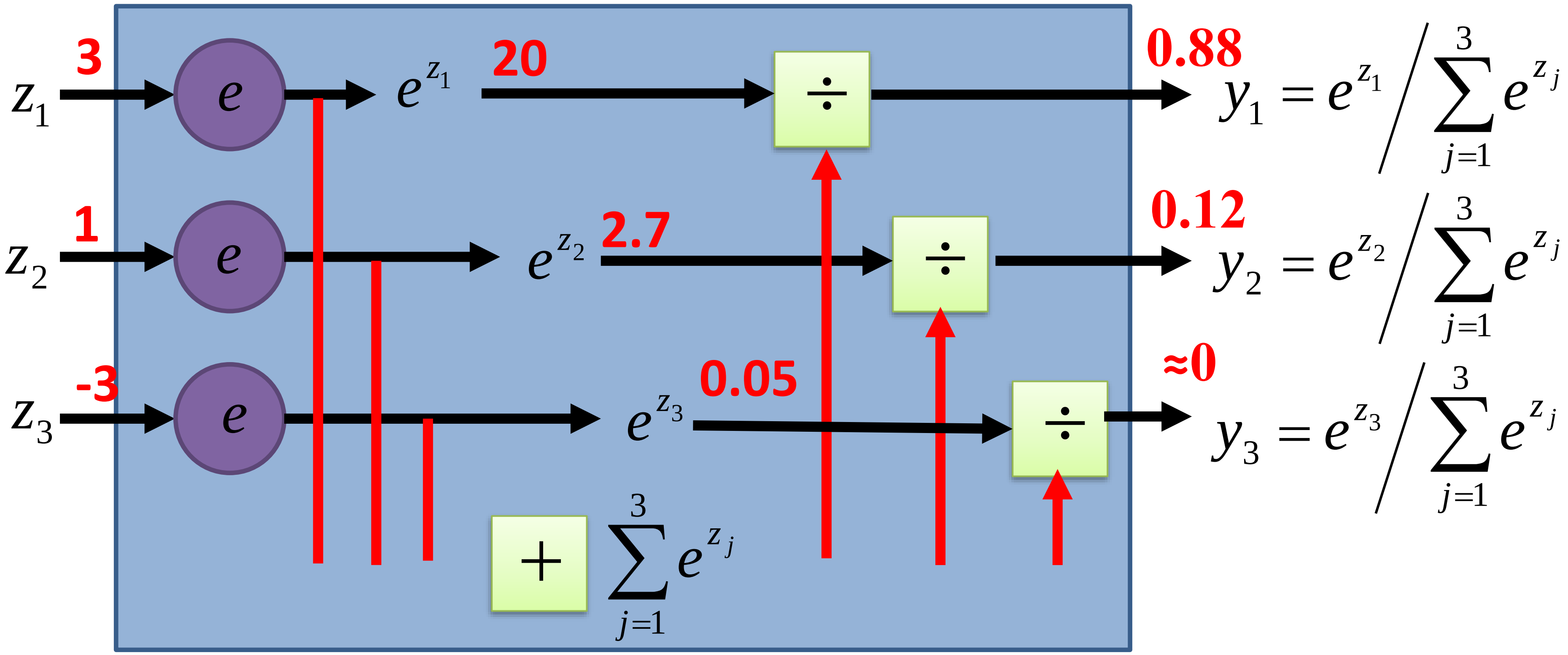
Softmax Layer

- The softmax layer applies softmax activations to output a probability value in the range [0, 1]
  - The values z inputted to the softmax layer are referred to as ***logits***

Probability:

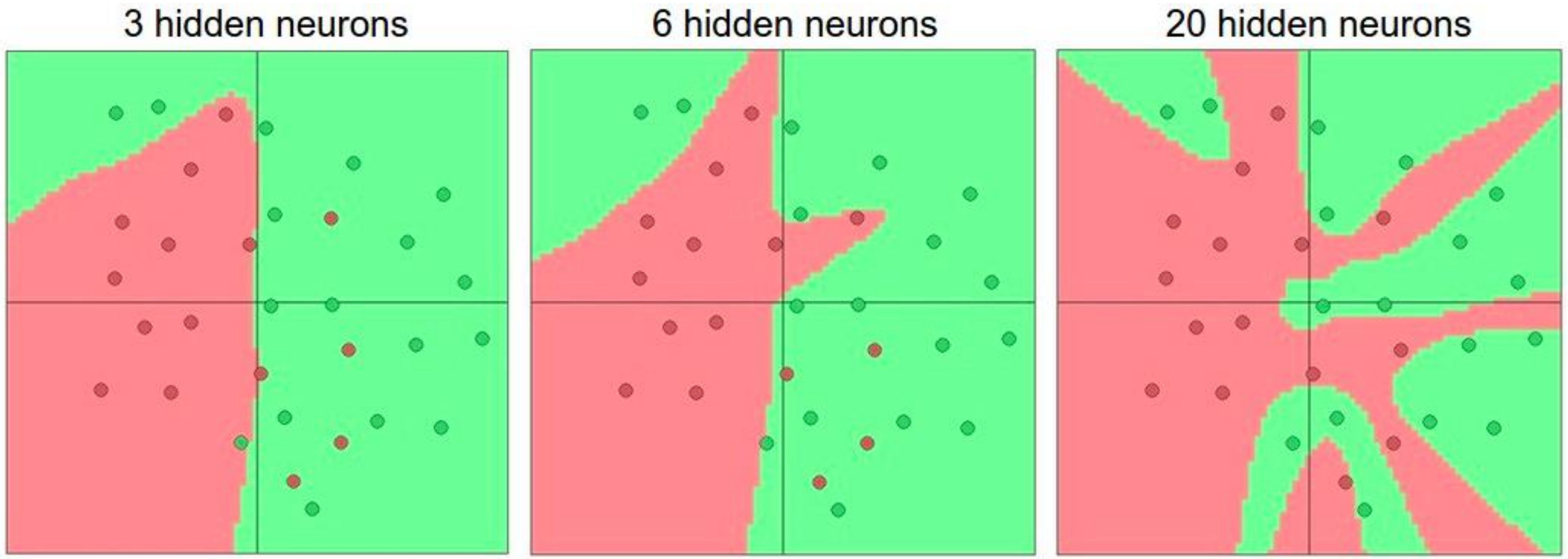
- $0 < y_i < 1$
- $\sum_i y_i = 1$

A Softmax Layer



Activation Functions

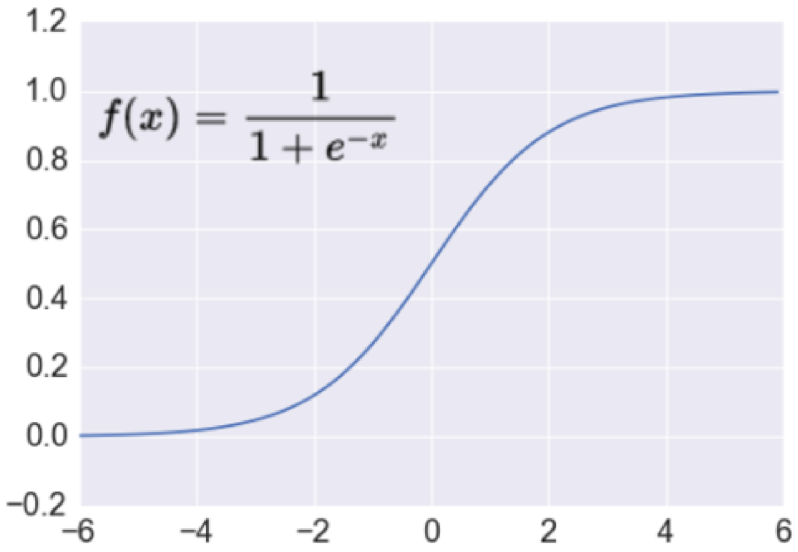
- Non-linear activations are needed to learn complex (non-linear) data representations
  - Otherwise, NNs would be just a linear function (such as $W_1 W_2 x = Wx$)
  - NNs with large number of layers (and neurons) can approximate more complex functions
    - Figure: more neurons improve representation (but, may overfit)



Activation: Sigmoid

- ***Sigmoid function*** σ: takes a real-valued number and "squashes" it into the range between 0 and 1
  - The output can be interpreted as the firing rate of a biological neuron
    - Not firing = 0; Fully firing = 1
  - When the neuron's activation are 0 or 1, sigmoid neurons saturate
    - Gradients at these regions are almost zero (almost no signal will flow)
  - Sigmoid activations are less common in modern NNs

$f(x)$

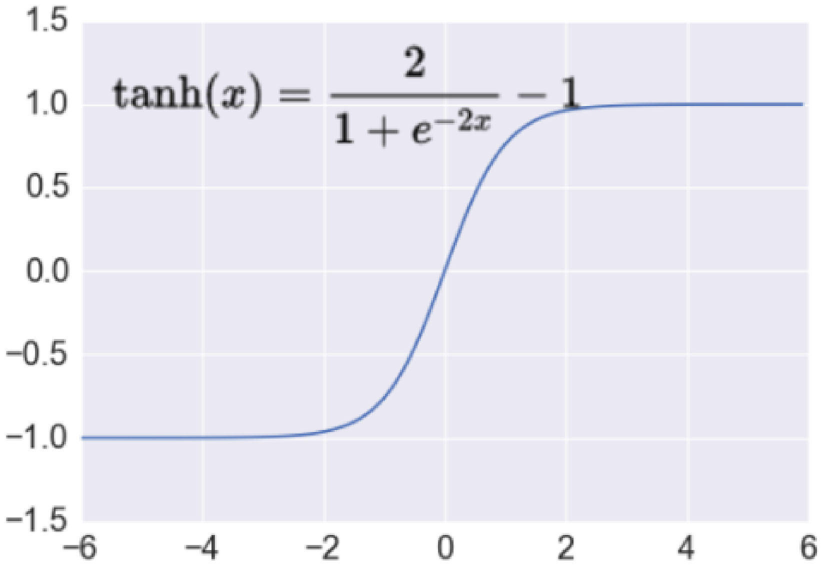


$\mathbb{R}^n \rightarrow [0,1]$

Activation: Tanh

- ***Tanh function***: takes a real-valued number and "squashes" it into range between -1 and 1
  - Like sigmoid, tanh neurons saturate
  - Unlike sigmoid, the output is zero-centered
    - It is therefore preferred than sigmoid
  - Tanh is a scaled sigmoid: $\tanh(x) = 2 \cdot \sigma(2x) - 1$

$f(x)$



$\mathbb{R}^n \rightarrow [-1,1]$

Activation: ReLU

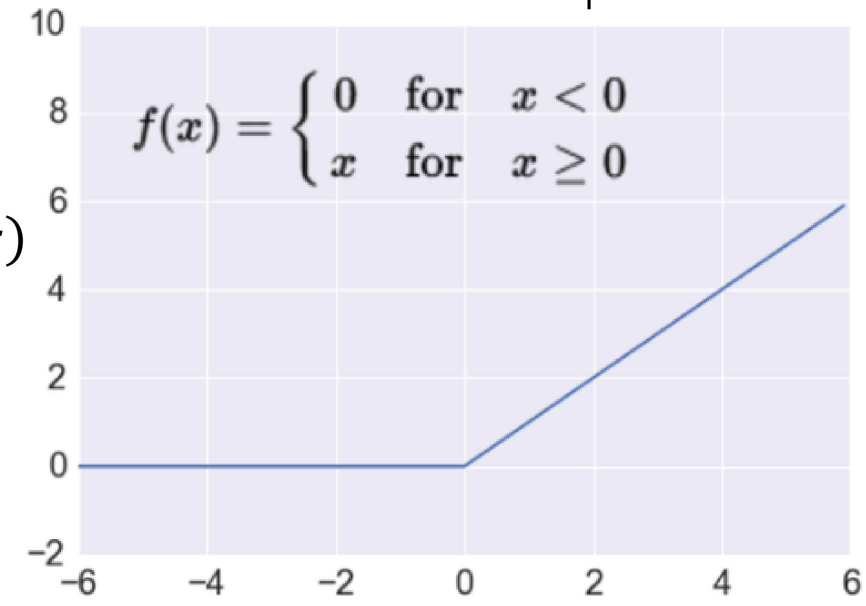
- ***ReLU*** (Rectified Linear Unit): takes a real-valued number and thresholds it at zero

$$f(x) = \max(0, x)$$

  - Most modern deep NNs use ReLU activations
  - ReLU is fast to compute
    - Compared to sigmoid, tanh
    - Simply threshold a matrix at zero
  - Accelerates the convergence of gradient descent
    - Due to linear, non-saturating form
  - Prevents the gradient vanishing problem

$\mathbb{R}^n \rightarrow \mathbb{R}^n_+$

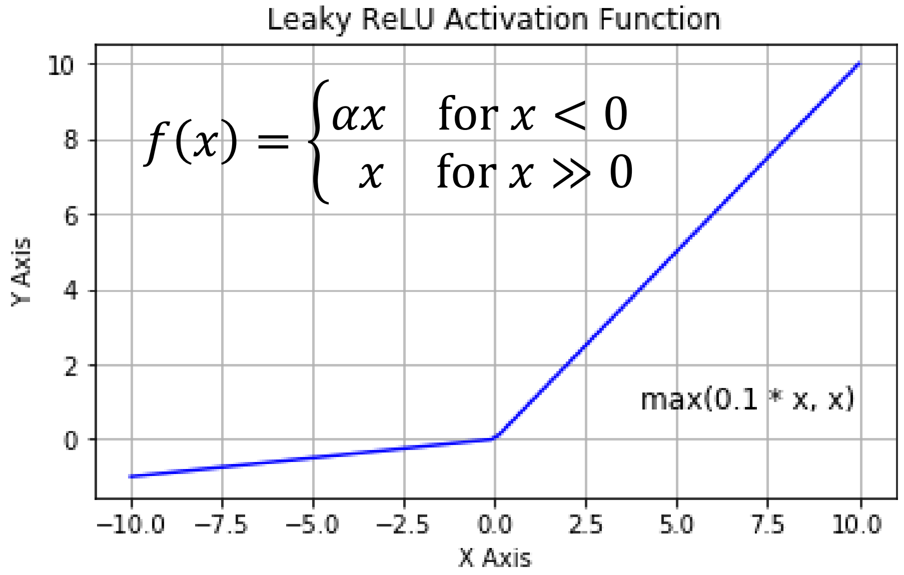
$f(x)$



x

Activation: Leaky ReLU

- The problem of ReLU activations: they can "die"
  - ReLU could cause weights to update in a way that the gradients can become zero and the neuron will not activate again on any data
  - E.g., when a large learning rate is used
- ***Leaky ReLU*** activation function is a variant of ReLU
  - Instead of the function being 0 when $x < 0$, a leaky ReLU has a small negative slope (e.g., $\alpha = 0.01$, or similar)
    - This resolves the dying ReLU problem
    - Most current works still use ReLU
      - With a proper setting of the learning rate, the problem of dying ReLU can be avoided



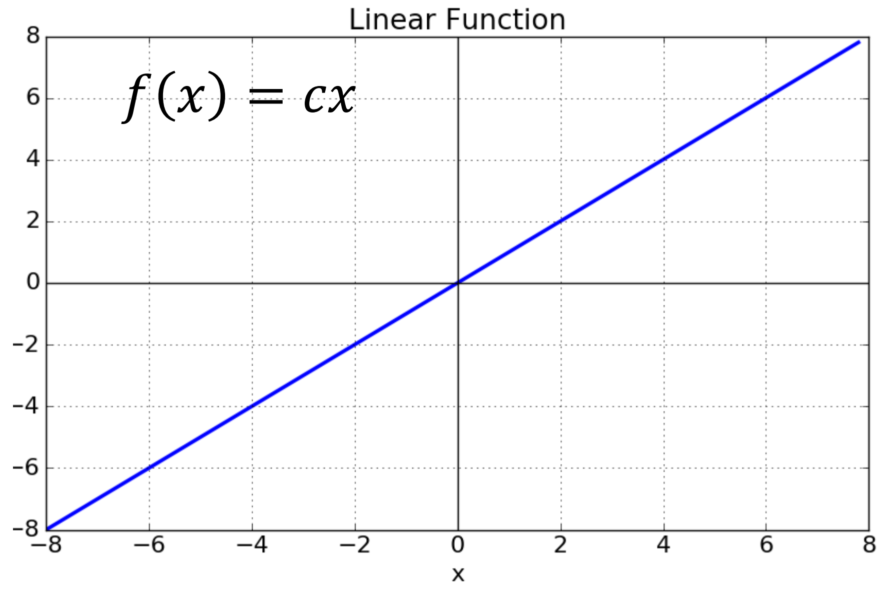
Activation: Linear Function

Introduction to Neural Networks

- ***Linear function*** means that the output signal is proportional to the input signal to the neuron

$$\mathbb{R}^n \rightarrow \mathbb{R}^n$$

  - If the value of the constant c is 1, it is also called identity activation function
  - This activation type is used in regression problems
    - E.g., the last layer can have linear activation function, in order to output a real number (and not a class membership)



References

1. Hung-yi Lee – Deep Learning Tutorial
2. Ismini Lourentzou – Introduction to Deep Learning
3. CS231n Convolutional Neural Networks for Visual Recognition (Stanford CS course) (link)
4. James Hays, Brown – Machine Learning Overview
5. Param Vir Singh, Shunyuan Zhang, Nikhil Malik – Deep Learning
6. Sebastian Ruder – An Overview of Gradient Descent Optimization Algorithms (link)