Neuron Network Basis

Chun-Hsiang Chan

Undergraduate Program in Intelligent Computing and Big Data, Chung Yuan Christian University
Master Program in Intelligent Computing and Big Data, Chung Yuan Christian University

Outline

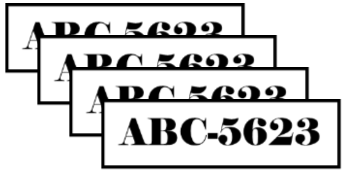
- Model Formulation
- Single Neuron
- Hidden Layer
- Activation Function
- Linearity and Non-linearity
- Model Parameters
- Loss Function

Model Formulation

- Imagine that you have a problem, such as a digit recognition, object detection, or numerical prediction.

Input **Function** **Output**

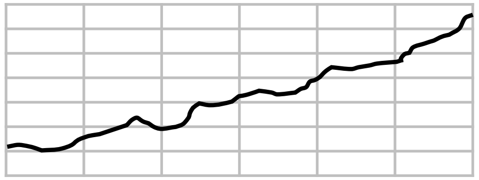
Digit recognition

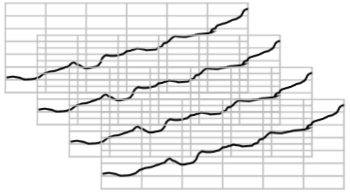


$f(x)$

> "ABC-5623"

Numerical prediction



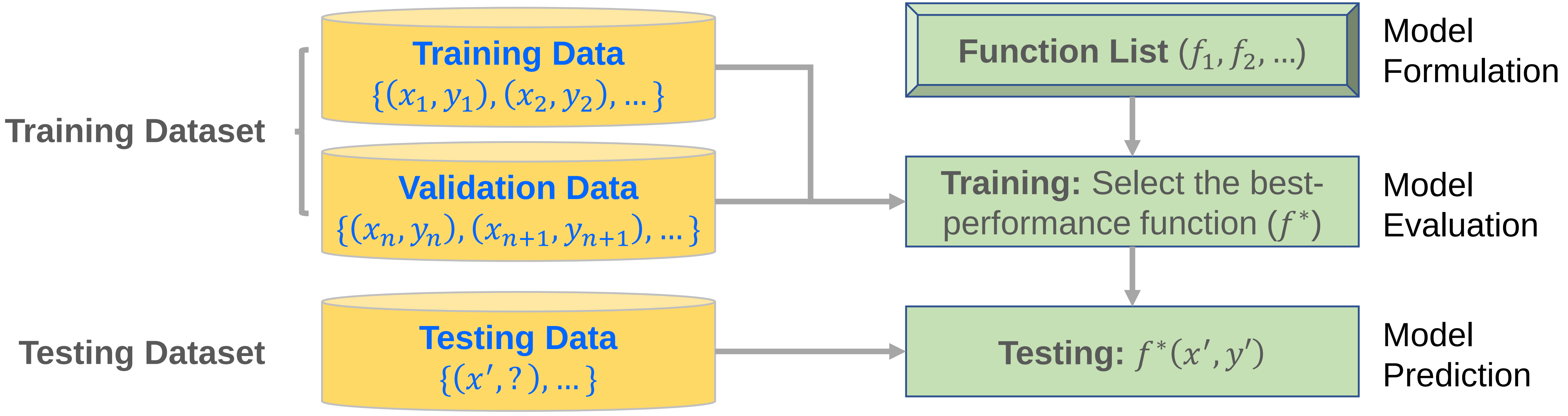


$f(x)$

> 234, 235, …

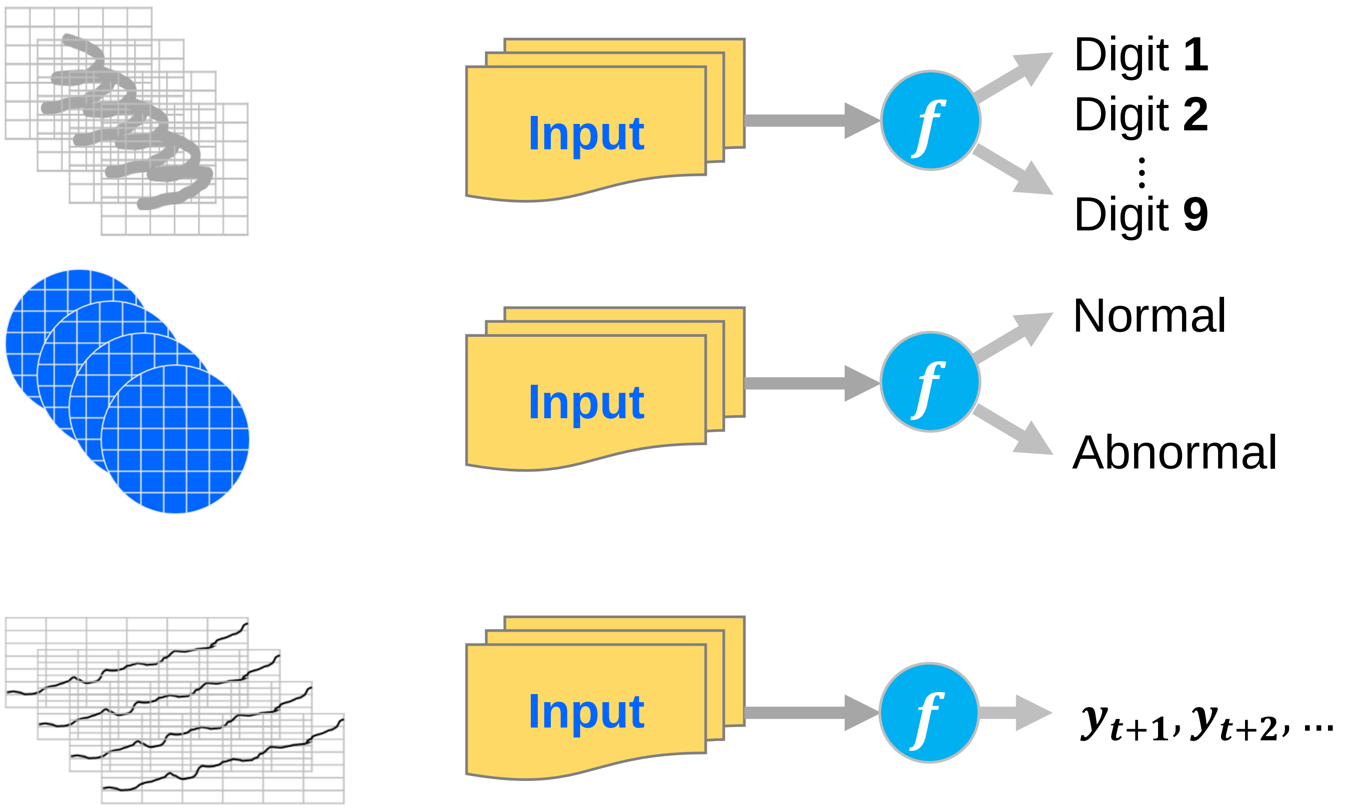
Model Formulation

- In machine learning and deep learning, we usually divide our input data into training and validation datasets for the training model.



Model Formulation

- **Prediction Tasks**
  - Handwritten Recognition
  - Abnormal Chip Testing
  - Stock Prediction

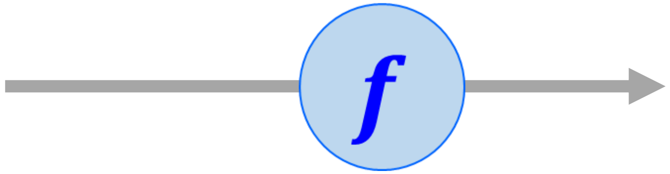


Model Formulation

- **Handwritten Recognition**: alphabet and number

x: *image*

7 x 6 array



y: *label*

$$f: R^N \rightarrow R^M$$

1 indicates the element contains ink
0 indicates the element contains nothing

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow 7 \times 6 = 42\ elements$$

10+26 dimension for label recognition

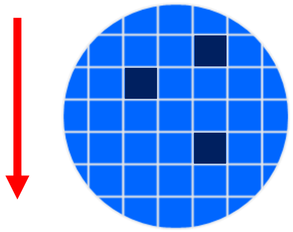
$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{matrix} a \\ b \\ c \\ \vdots \\ 8 \\ 9 \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Detect　　Ground Truth

Model Formulation

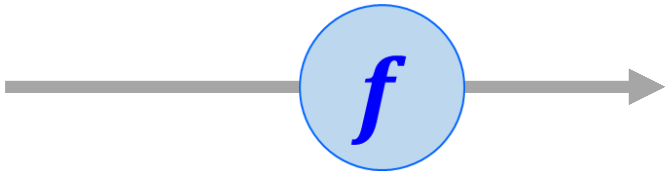
- **Abnormal Chip Testing**

x: $\boldsymbol{image}$



1 indicates a defect chip
0 indicates a normal chip

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow dimensions = size(chips)$$



y: $\boldsymbol{label}$

2 dimensions

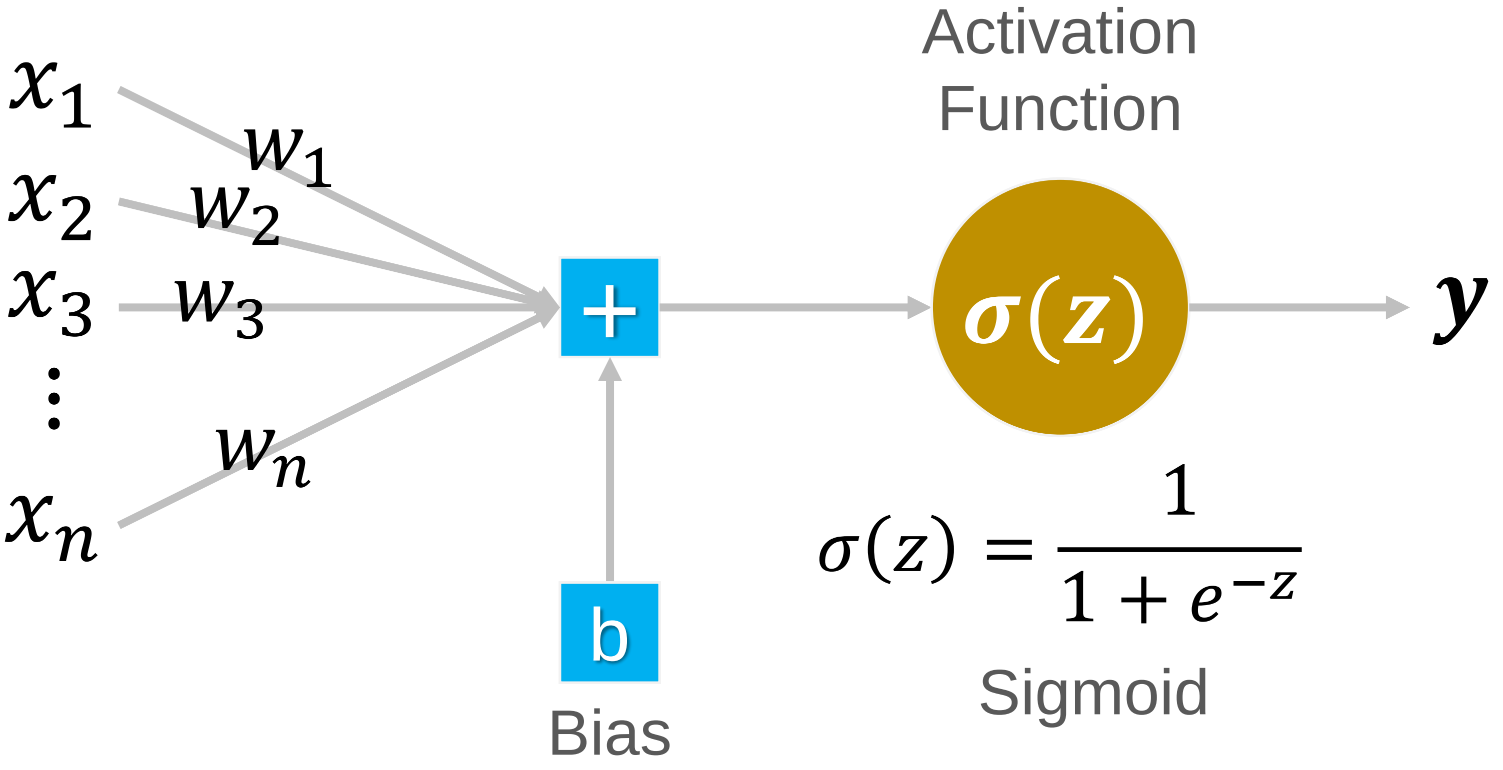
Normal chip or Defect chip

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{matrix} \text{V} \\ \text{V} \\ \times \\ \vdots \\ \text{V} \\ \text{V} \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Confirm

Detect Ground Truth

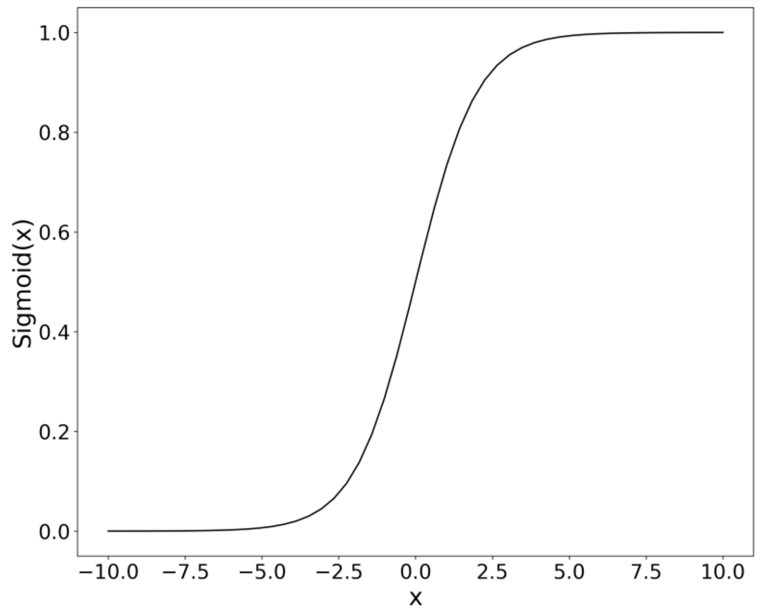
Single Neuron



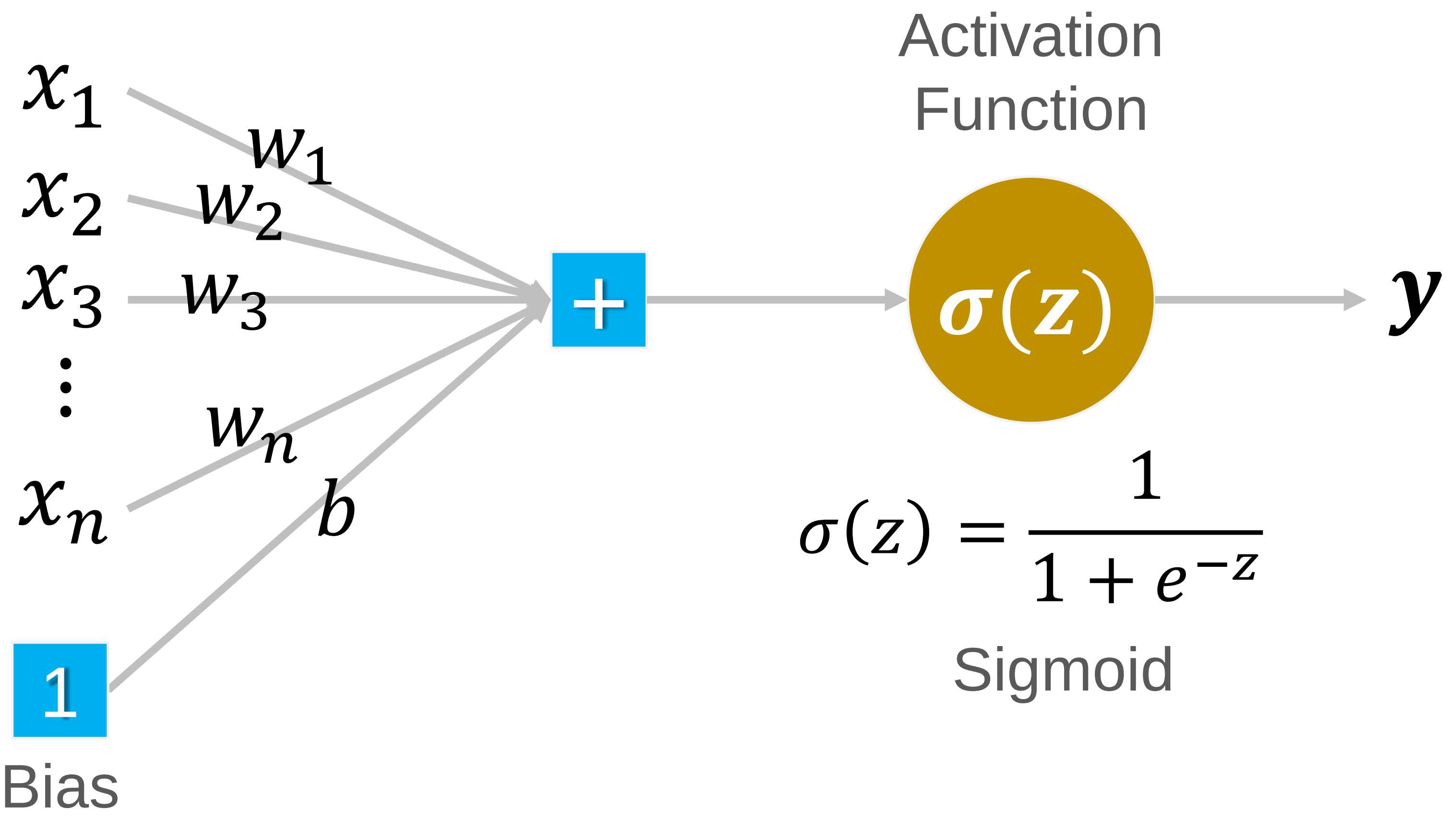
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid

$\sigma(z)$



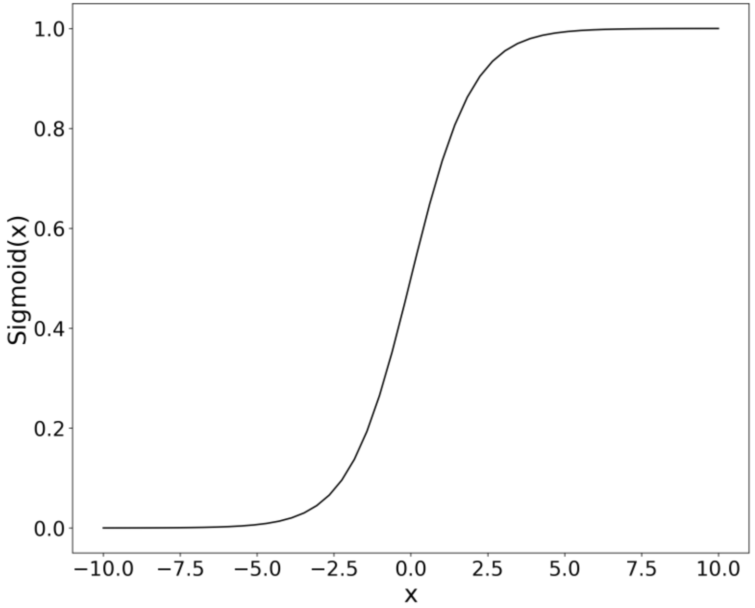
Single Neuron



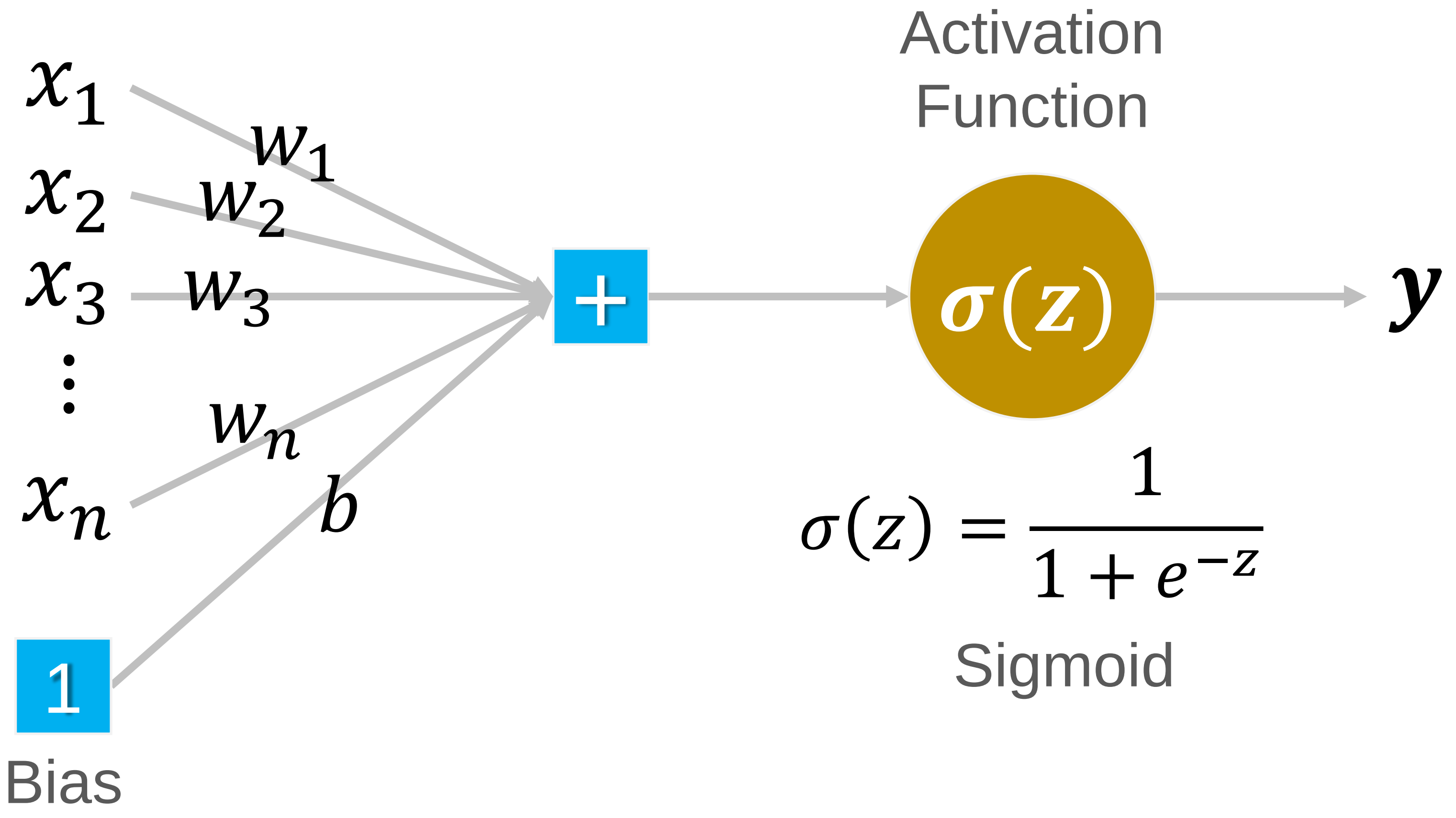
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid

$\sigma(z)$



Single Neuron

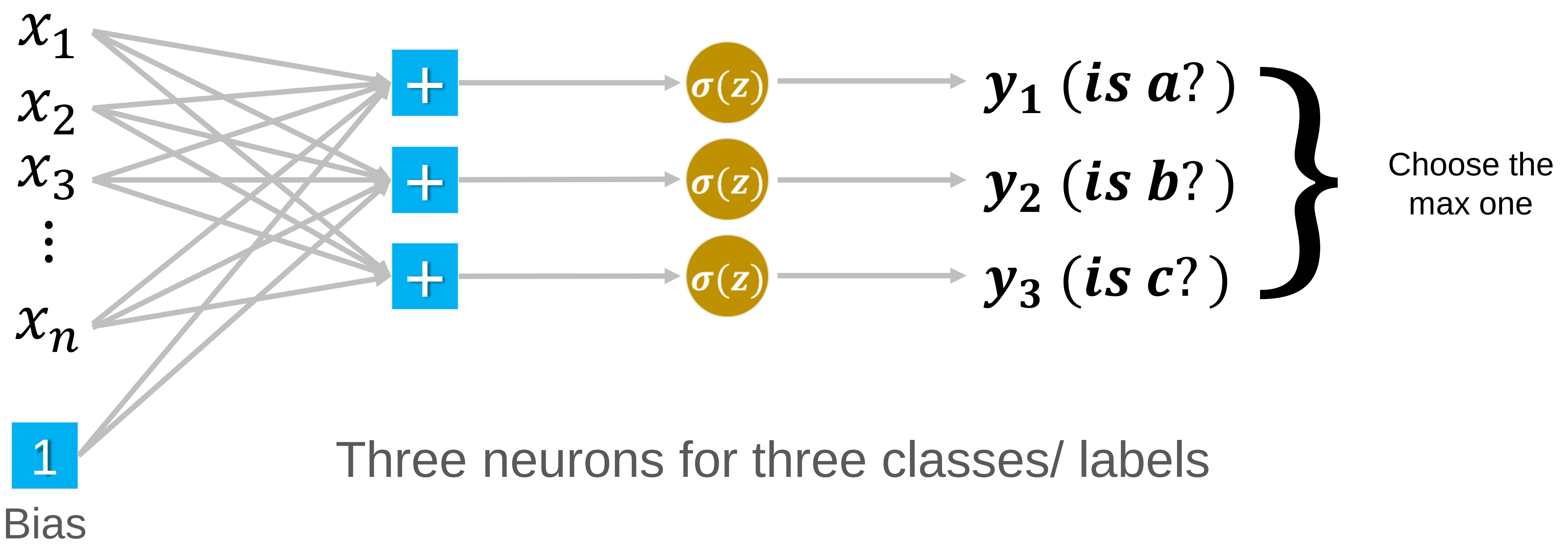


$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid

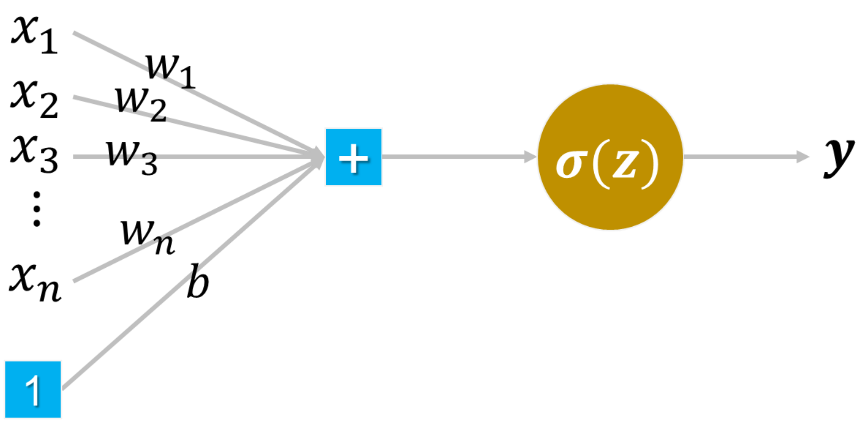
$$y = h_{w,b}(x) \\ = \sigma(w^T x + b)$$

A Layer of Neurons – Perceptron



Three neurons for three classes/ labels

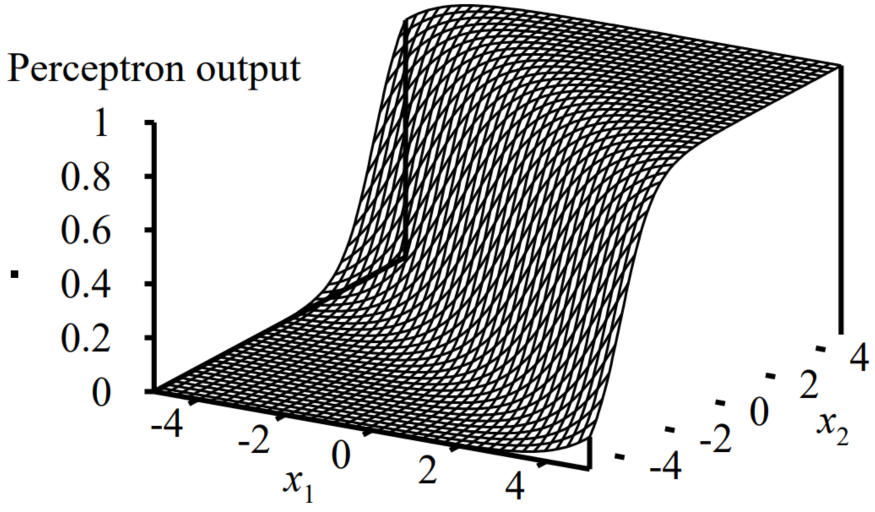
A Layer of Neurons – Perceptron



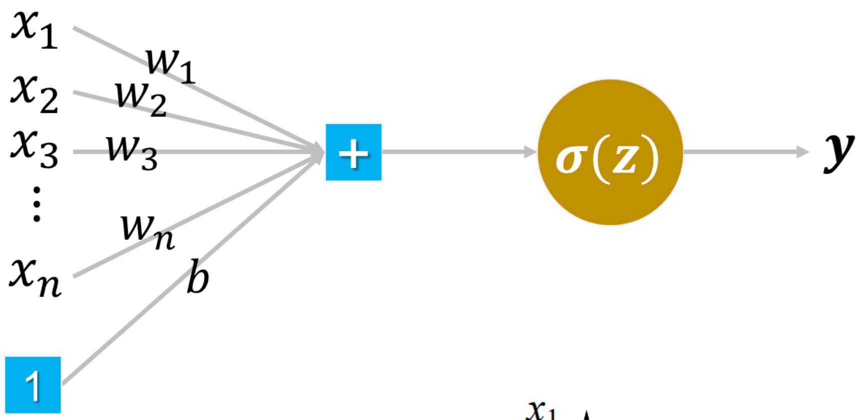
$$\begin{cases} true & y \geq \theta \\ false & y < \theta \end{cases}$$

θ *is threshold*

$y = \sigma(w_1 x_1 + w_2 x_2 + b)$ is a linear formula…

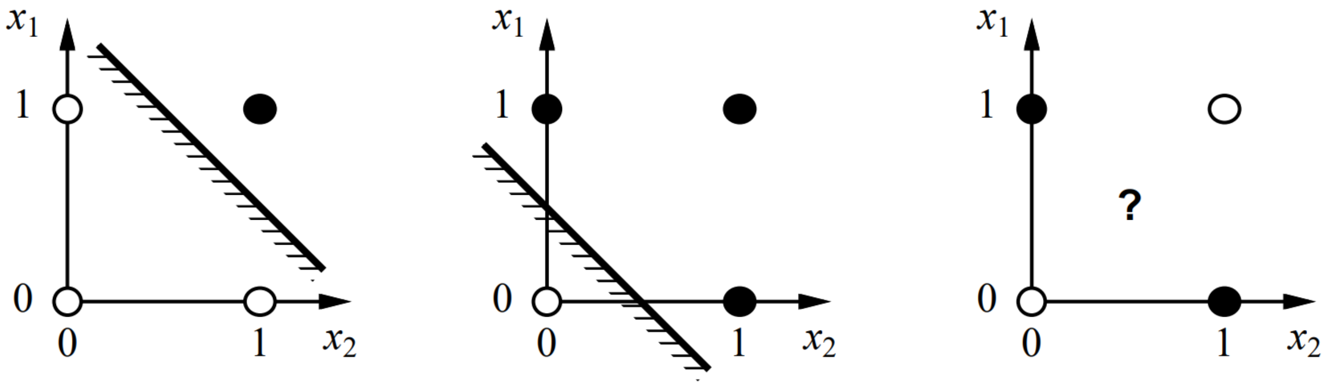


A Layer of Neurons – Perceptron



$$\begin{cases} true & y \geq \theta \\ false & y < \theta \end{cases}$$

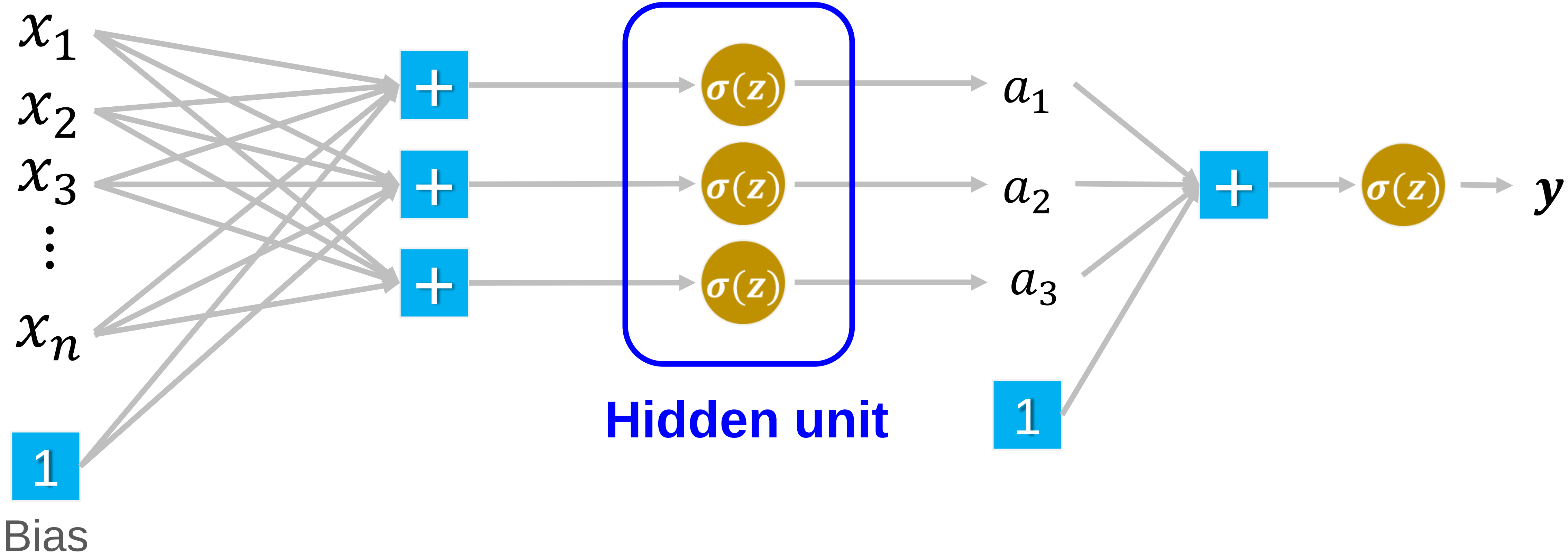
θ *is threshold*



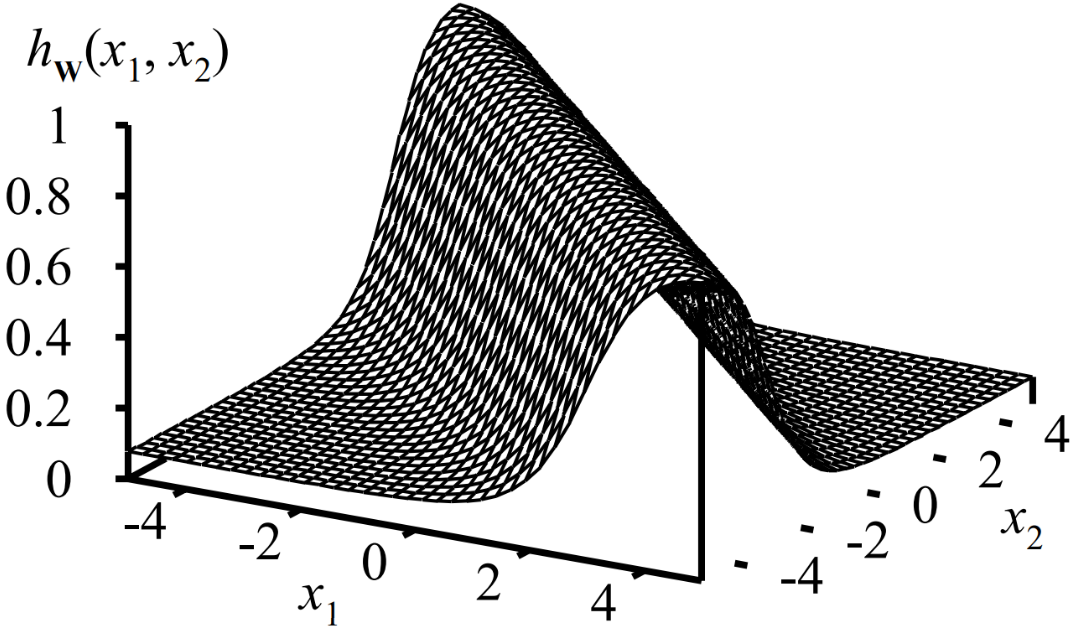
(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

Minsky & Papert (1969) pricked the neural network balloon

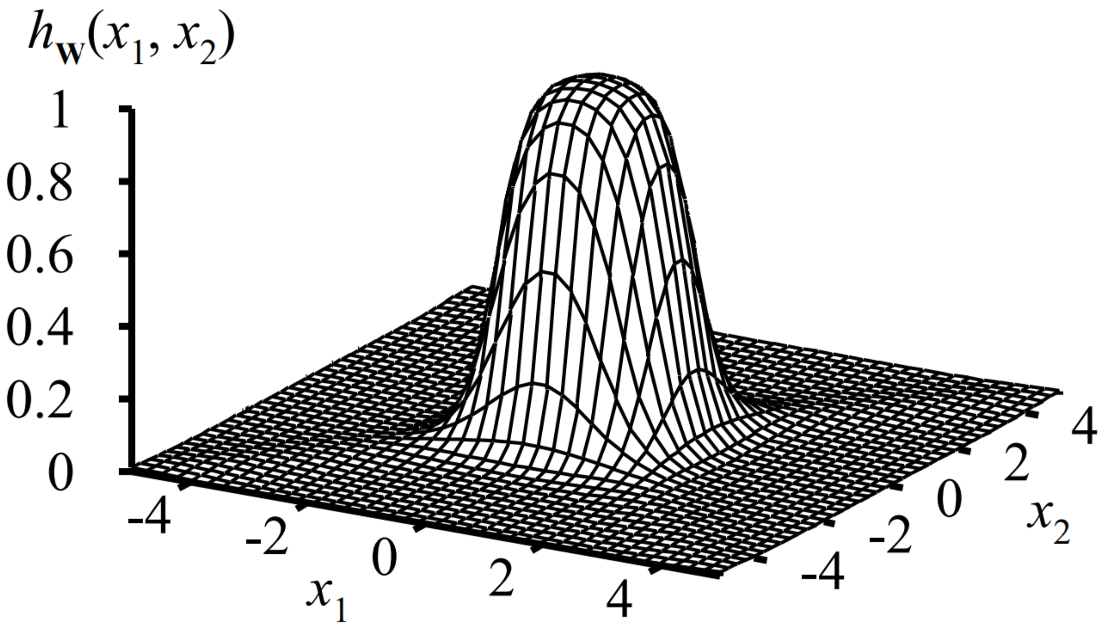
A Layer of Neurons – Perceptron



A Layer of Neurons – Perceptron



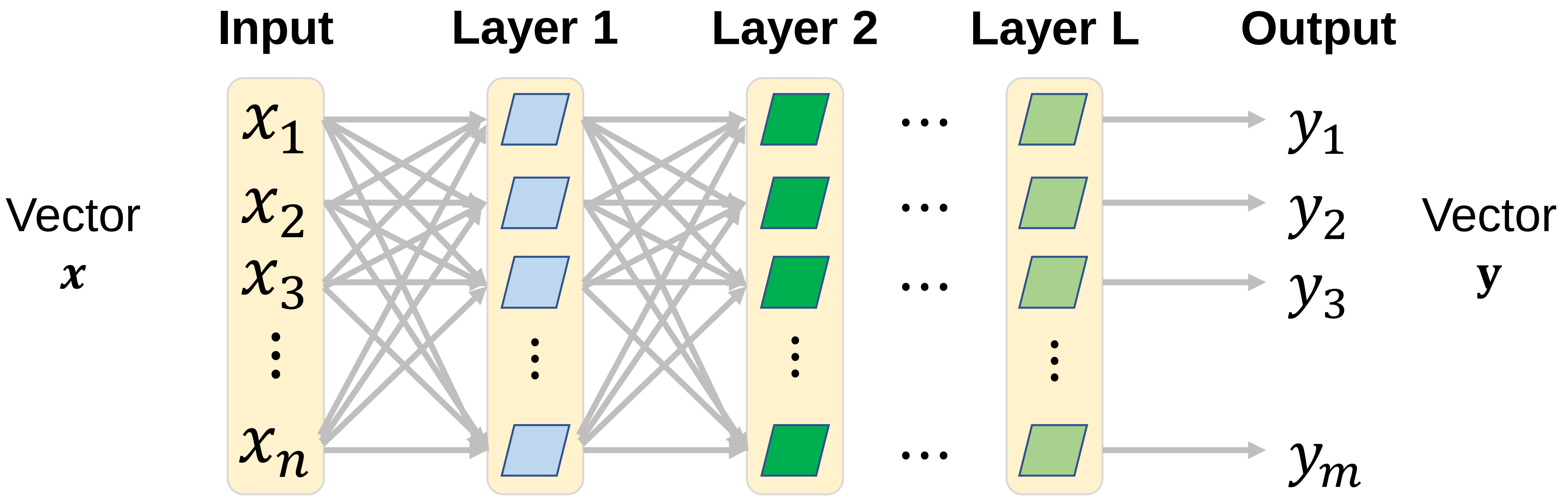
Combine two opposite-facing threshold functions to make a ridge



Combine two perpendicular ridges to make a bump

Fully Connected Feedforward Network

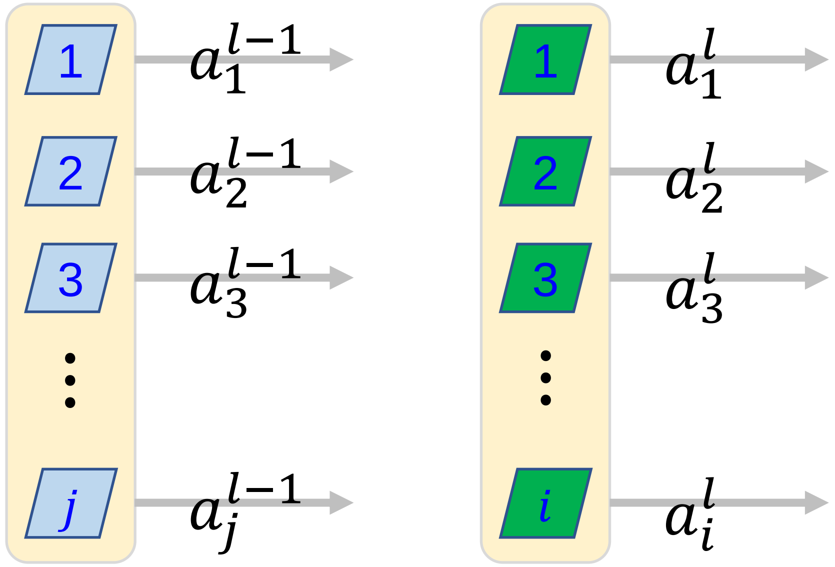
Deep Neural Network (DNN)



DNN: multiple hidden layers

Hidden Layer – Layer

Layer $l-1$ **Layer l**



N_{l-1} nodes **N_l nodes**

A Neuron Output

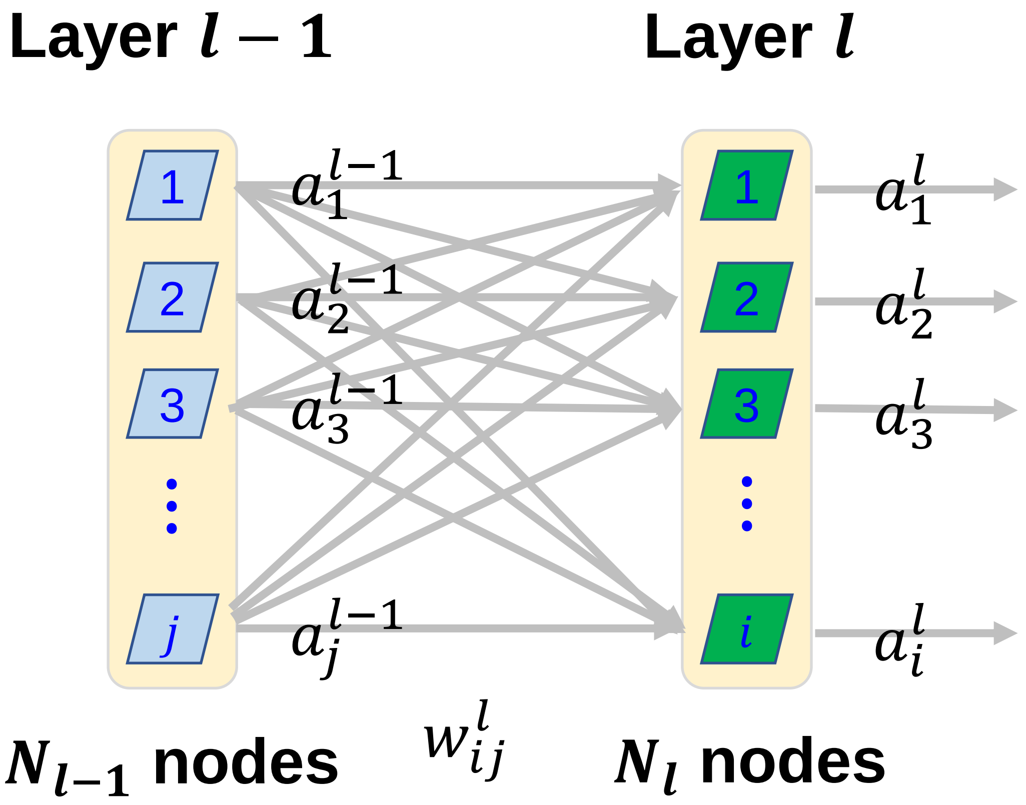
a_1^l → Layer l

→ neuron i

$$a^l = \begin{bmatrix} \vdots \\ a_i^l \\ \vdots \end{bmatrix}$$

One layer ➔ One Vector

Hidden Layer – Weights



A Weight between two neurons/nodes

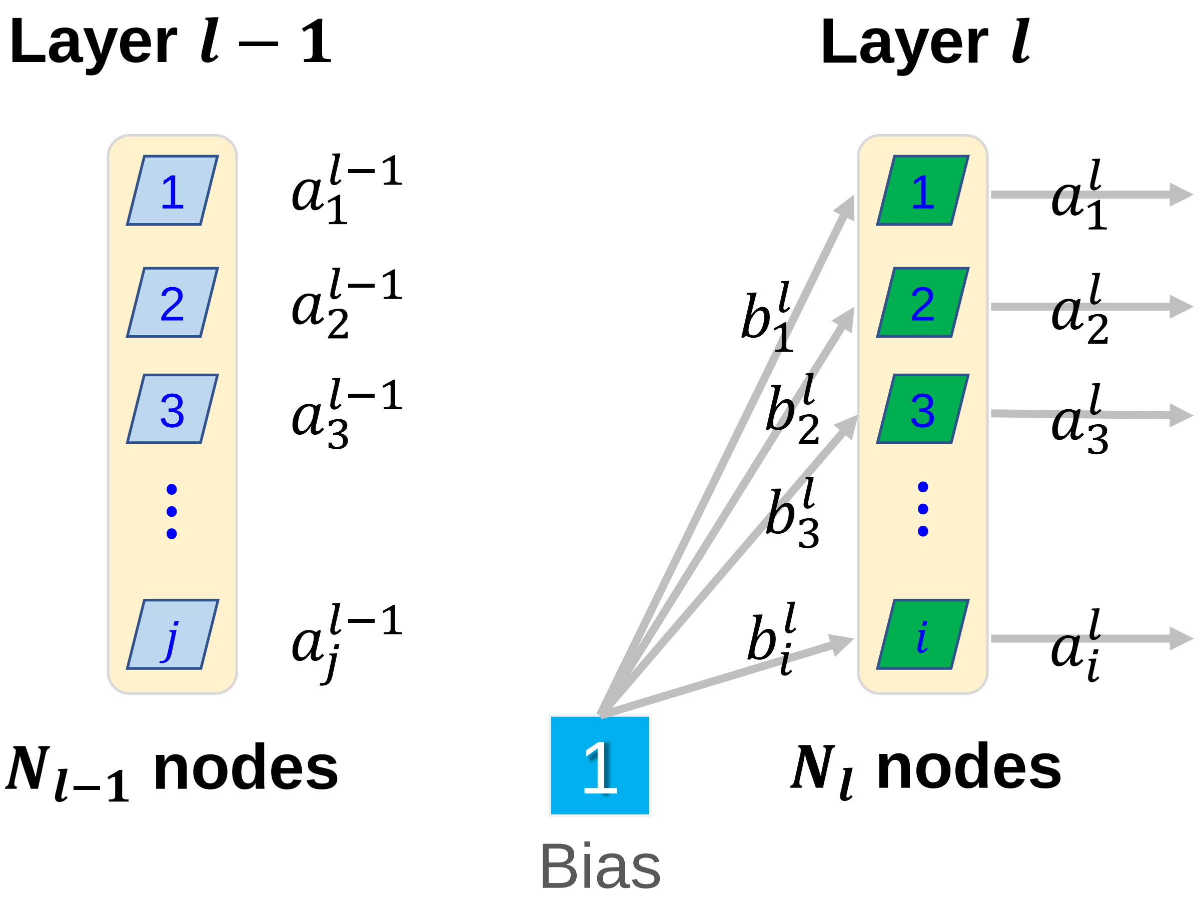
w_{ij}^l → Layer $l-1$ to Layer l

→ From neuron j (layer $l-1$) to neuron i (layer l)

$$w^l = \begin{array}{c} N_{l-1} \\ \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array} N_l$$

Weights between layers ➔ One Matrix

Hidden Layer – Bias

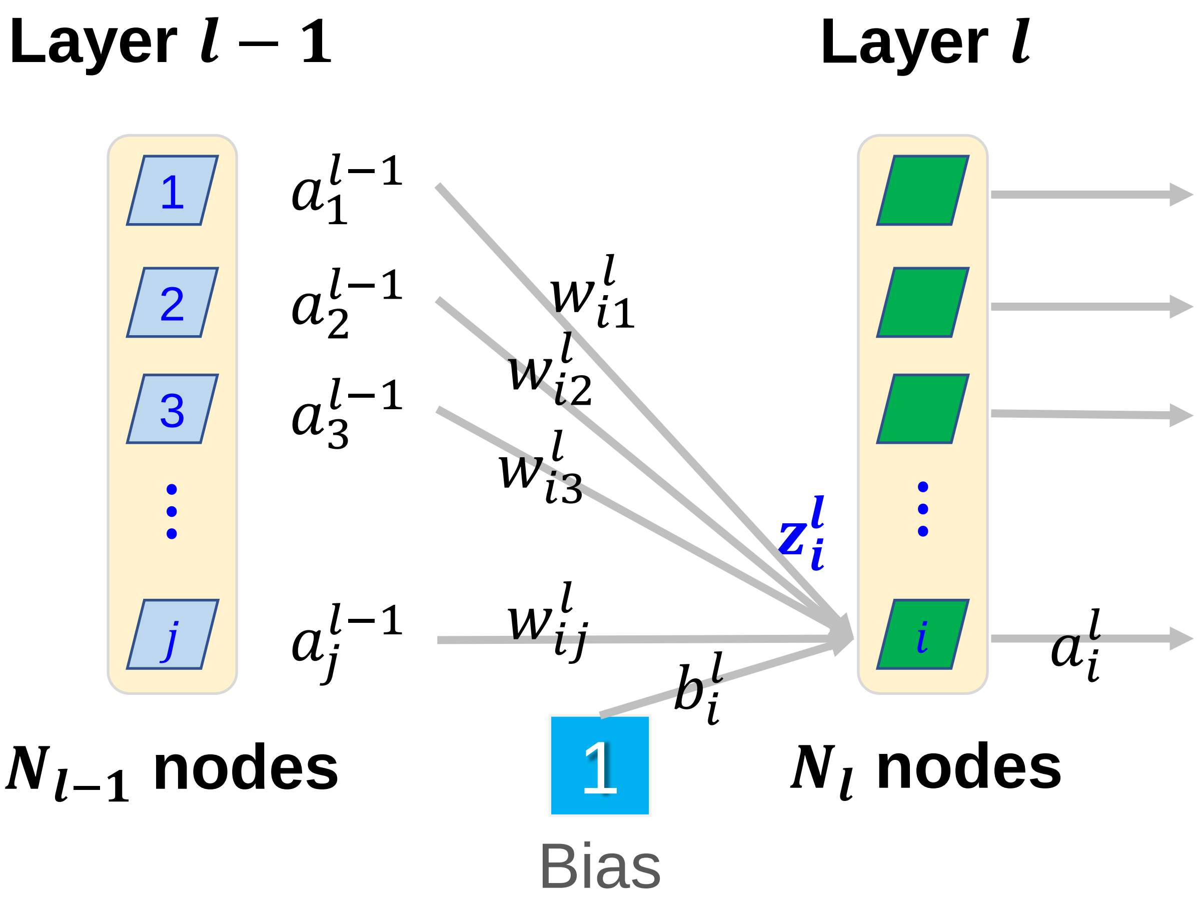


b_i^l Bias for neuron i at layer l

$$b^l = \begin{bmatrix} \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

Bias of each layer ➔ One Vector

Hidden Layer – Activation Function



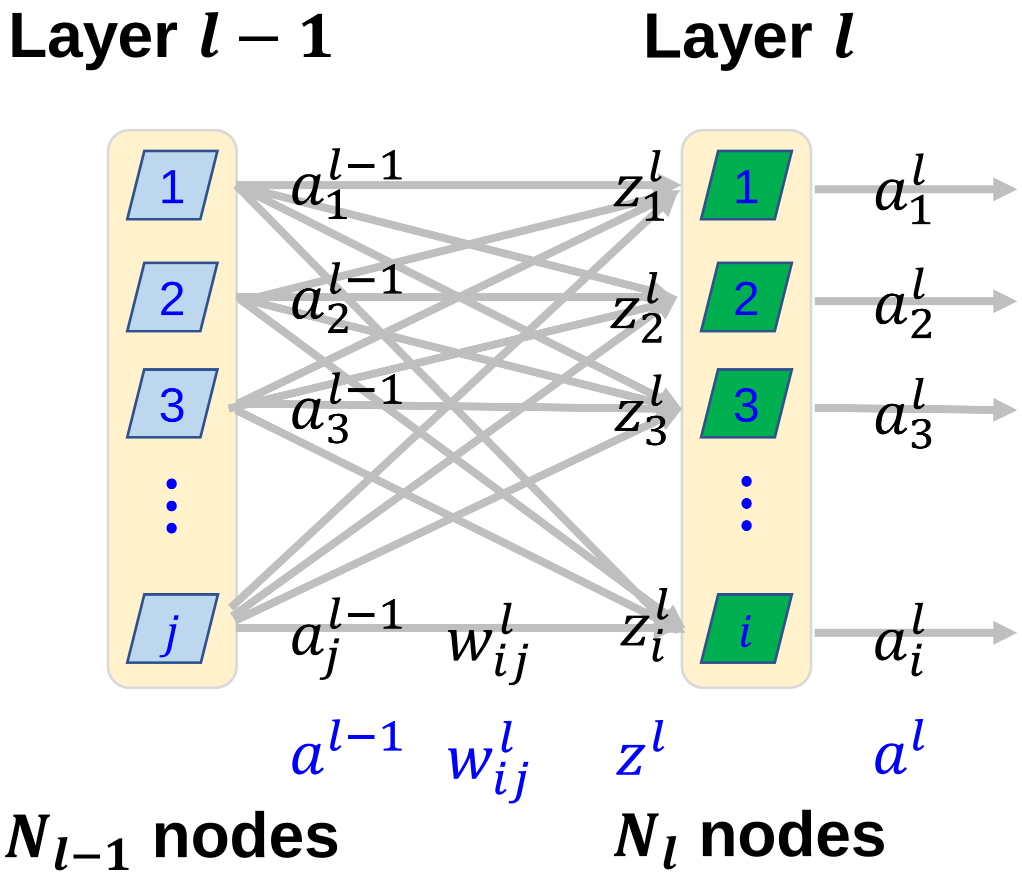
$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \cdots + b_i^l$$

$$z_i^l = \sum_{j=1}^{N_{l-1}} w_{ij}^l a_j^{l-1} + b_i^l$$

$$z^l = \begin{bmatrix} \vdots \\ z_i^l \\ \vdots \end{bmatrix}$$

Activation function at each layer ➔ One Vector

Hidden Layer – Overall



$$z_1^l = w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \cdots + b_1^l$$

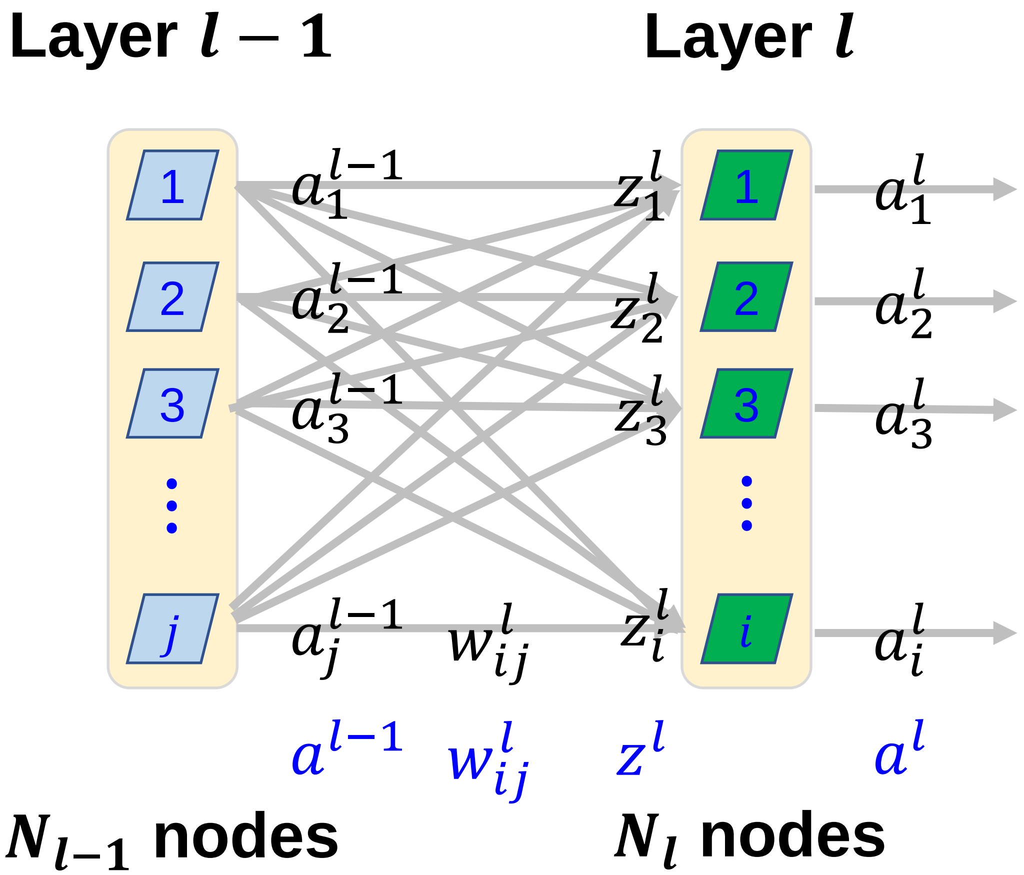
$$\vdots$$

$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \cdots + b_i^l$$

$$\begin{bmatrix} \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$z^l = W^l a^{l-1} + b^l$$

Hidden Layer – Overall

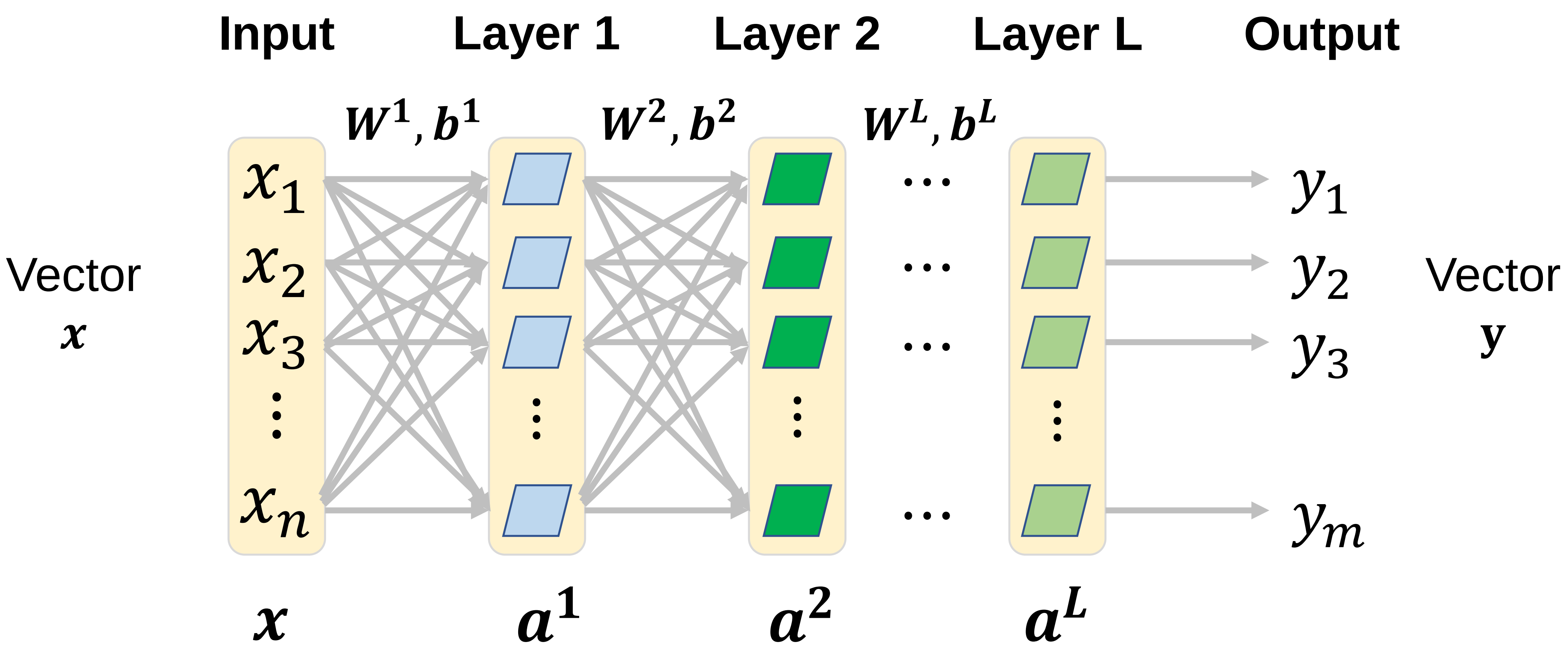


$$a_i^l = \sigma\left(z_i^l\right)$$

$$\begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_i^l \end{bmatrix} = \begin{bmatrix} \sigma\left(z_1^l\right) \\ \sigma\left(z_2^l\right) \\ \vdots \\ \sigma\left(z_i^l\right) \end{bmatrix}$$

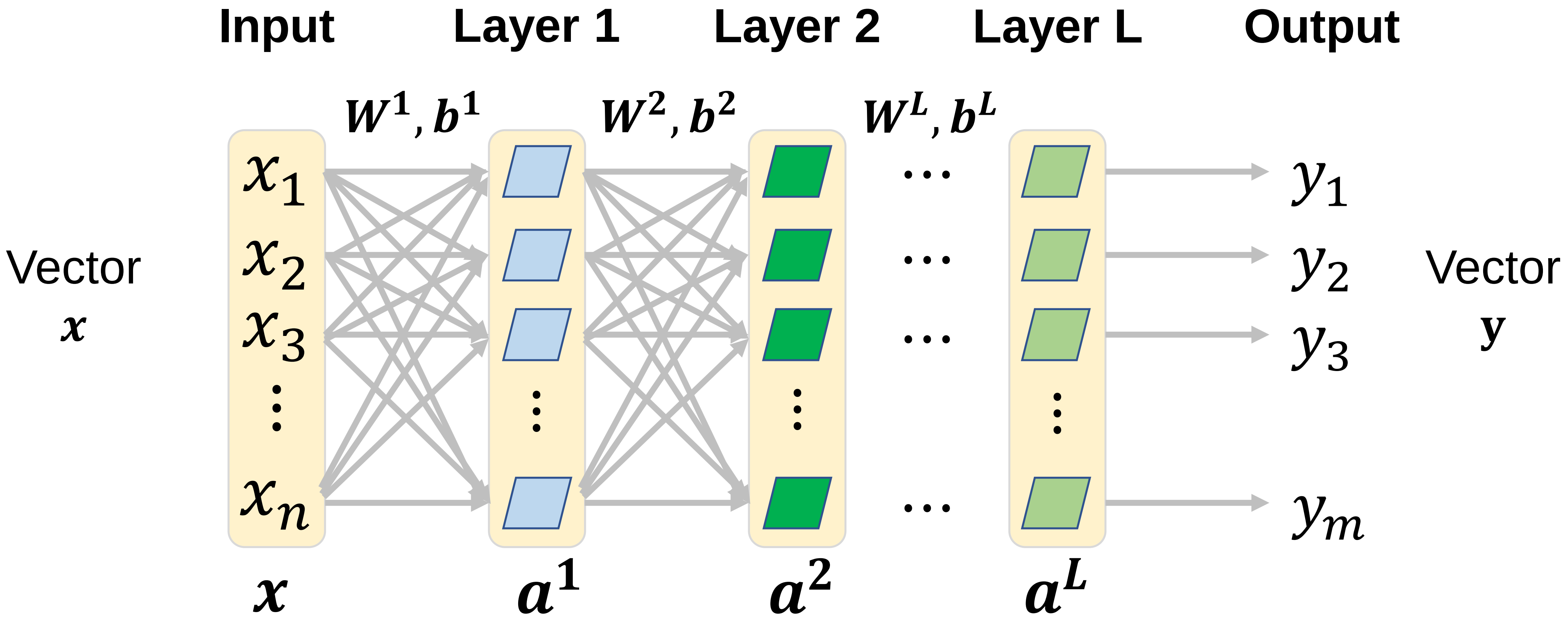
$$a^l = \sigma\left(z^l\right)$$

Hidden Layer – Overall



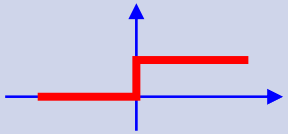
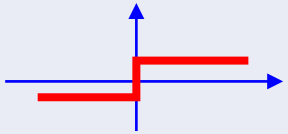
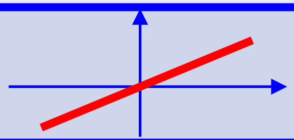
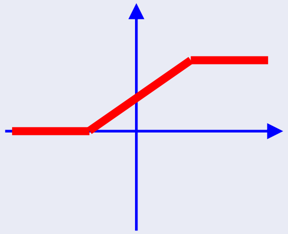
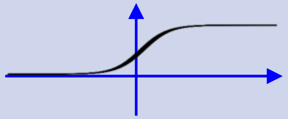
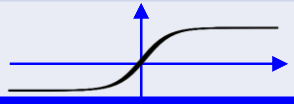
$\sigma\left(W^1 x + b^l\right)$ $\quad \sigma(W^2 a^1 + b^2)$ $\quad \sigma(W^L a^{L-1} + b^L) = y$

Hidden Layer – Overall

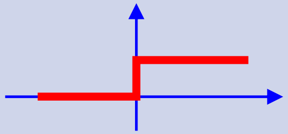
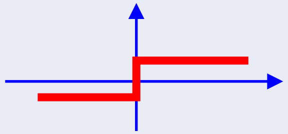
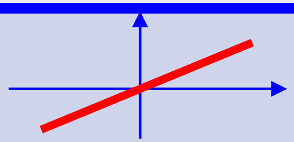
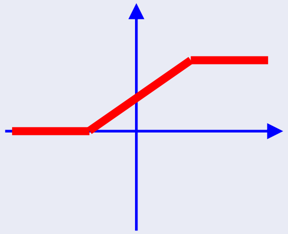
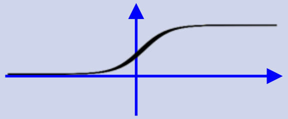
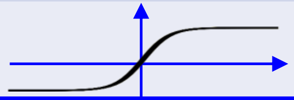


$$y = f(x) = \sigma(W^L \ldots \sigma(W^2\sigma(W^1 x + b^1) + b^2) \ldots + b^L)$$

Activation Function $\sigma(\cdot)$

	Activation Function	Equation	Example	1D Graph
Bounded function	Unit step (Heatviside)	$\phi(z)=\begin{cases}0 & z<0\\ 0.5 & z=0\\ 1 & z>0\end{cases}$	Perceptron variant	
	Sign (Signum)	$\phi(z)=\begin{cases}-1 & z<0\\ 0 & z=0\\ 1 & z>0\end{cases}$	Perceptron variant	
	Linear	$\phi(z)=z$	Adaline, linear regression	
Bounded function	Piece-wise Linear	$\phi(z)=\begin{cases}1 & z\geq\frac{1}{2}\\ z+\frac{1}{2} & -\frac{1}{2}<z<\frac{1}{2}\\ 0 & z\leq-\frac{1}{2}\end{cases}$	Support vector machine	
	Logistic (sigmoid)	$\phi(z)=\frac{1}{1+e^{-z}}$	Logistic regression, multi-layer NN	
	Hyperbolic Tangent	$\phi(z)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	Multi-layer NN	

Activation Function $\boldsymbol{\sigma}(\cdot)$

	Activation Function	Equation	Example	1D Graph
Boolean	Unit step (Heatviside)	$\phi(z)=\begin{cases}0 & z<0\\ 0.5 & z=0\\ 1 & z>0\end{cases}$	Perceptron variant	
	Sign (Signum)	$\phi(z)=\begin{cases}-1 & z<0\\ 0 & z=0\\ 1 & z>0\end{cases}$	Perceptron variant	
Linear	Linear	$\phi(z)=z$	Adaline, linear regression	
	Piece-wise Linear	$\phi(z)=\begin{cases}1 & z\geq\frac{1}{2}\\ z+\frac{1}{2} & -\frac{1}{2}<z<\frac{1}{2}\\ 0 & z\leq-\frac{1}{2}\end{cases}$	Support vector machine	
Non-linear	Logistic (sigmoid)	$\phi(z)=\frac{1}{1+e^{-z}}$	Logistic regression, multi-layer NN	
	Hyperbolic Tangent	$\phi(z)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	Multi-layer NN	

Activation Function (non-linearity)

- **Sigmoid**

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

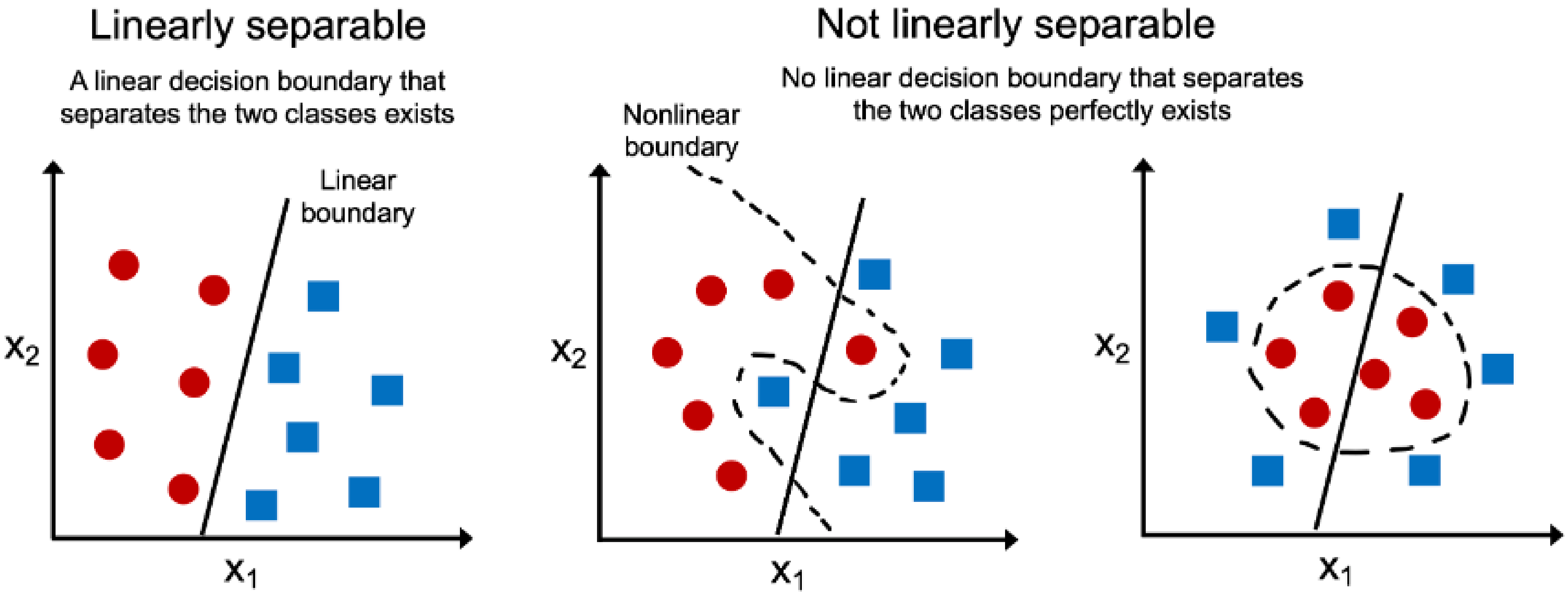
- **Tanh**

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- **Rectified Linear Unit (ReLU)**

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

Linearity and Non-linearity



https://vitalflux.com/how-know-data-linear-non-linear/

Linearity and Non-linearity

- With linearity, the deep neural network is the same as linear transform.

$$W_1(W_2 \cdot x) = (W_1 W_2)x = Wx$$

- With non-linearity, the deep neural network with multiple layers could have a more complex function.

Model Parameters

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

Function set

Different parameters W and b → different functions

Formal definition

$$f(x;\theta) \Rightarrow model\ parameter\ set$$
$$\theta = \{W^1, b^1, W^2, b^2, \ldots, W^L, b^L\}$$

Loss Function & Reward Function

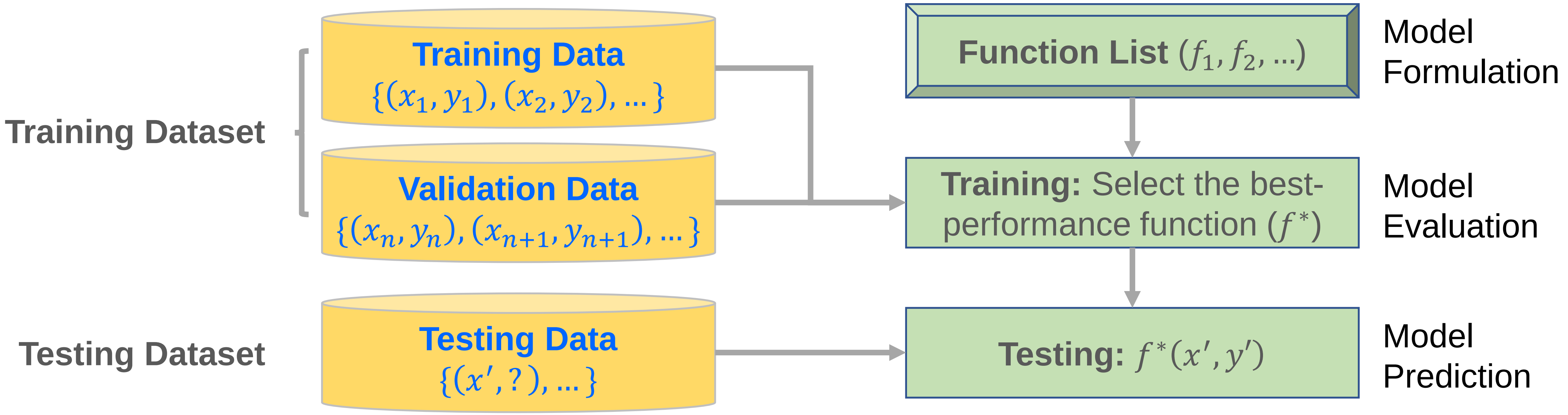
- Define a function to measure the quality of parameters set θ
  - Evaluating by a loss/cost/error function $C(\theta)$ ➔ how bad

$$\theta^* = \arg\min_{\theta} C(\theta)$$

  - Evaluating by an objective/reward function $O(\theta)$ ➔ how good

$$\theta^* = \arg\max_{\theta} O(\theta)$$

Loss Function



Best function: $f(x;\theta) \sim \hat{y} \rightarrow \|\hat{y} - f(x;\theta)\| \approx 0$

Define a loss function: $C(\theta) = \sum_k \|\hat{y}_k - f(x_k;\theta)\|$

Loss Function

Table 1 List of losses analyzed in this paper. y is true label as one-hot encoding, $\hat{y}$ is true label as $+1/-1$ encoding, o is the output of the last layer of the network, $\cdot\ (j)$ denotes $j-th$ dimension of a given vector, and $\sigma(\cdot)$ denotes probability estimate.

Symbol	Name	Equation
$\mathcal{L}_1$	L_1 loss	$\|y-o\|_1$
$\mathcal{L}_2$	L_2 loss	$\|y-o\|_2^2$
$\mathcal{L}_1 \circ \sigma$	Expectation loss	$\|y-\sigma(o)\|_1$
$\mathcal{L}_2 \circ \sigma$	Regularized expectation loss	$\|y-\sigma(o)\|_2^2$
$\mathcal{L}_\infty \circ \sigma$	Chebyshev loss	$max_j\left\|\sigma(o)^j - y^j\right\|$
$Hinge$	Hinge (margin) los	$\sum_j \max\left(0, \frac{1}{2} - \hat{y}^j o^j\right)$
$Hinge^2$	Squared hinge (margin) loss	$\sum_j \max\left(0, \frac{1}{2} - \hat{y}^j o^j\right)^2$
$Hinge^3$	Cubed hinge (margin) loss	$\sum_j \max\left(0, \frac{1}{2} - \hat{y}^j o^j\right)^3$

Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*.

Loss Function

Table 1 List of losses analyzed in this paper. y is true label as one-hot encoding, $\hat{y}$ is true label as $+1/-1$ encoding, o is the output of the last layer of the network, $\cdot\ (j)$ denotes $j-th$ dimension of a given vector, and $\sigma(\cdot)$ denotes probability estimate.

Symbol	Name	Equation
$\log$	Log (cross entropy) loss	$-\sum_j y^j log\sigma(o)^j$
$\log^2$	Squared log loss	$-\sum_j \left[y^j log\sigma(o)^j\right]^2$
tan	Tanimoto loss	$\frac{-\sum_j \sigma(o)^j y^j}{\lVert\sigma(o)\rVert_2^2 + \lVert y\rVert_2^2 - \sum_j \sigma(o)^j y^j}$

Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*.

References

- 臺大電機系李宏毅教授講義
- 臺大資工系陳縕儂教授講義
- NVIDIA Deep Learning Tutorial
- https://aima.cs.berkeley.edu/slides-pdf/chapter20b.pdf
- Janocha and Czarnecki (2017) On Loss Functions for Deep Neural Networks in Classification. *Computer Science ArXiv*. abs/1702.05659.

Thank you for your attention!