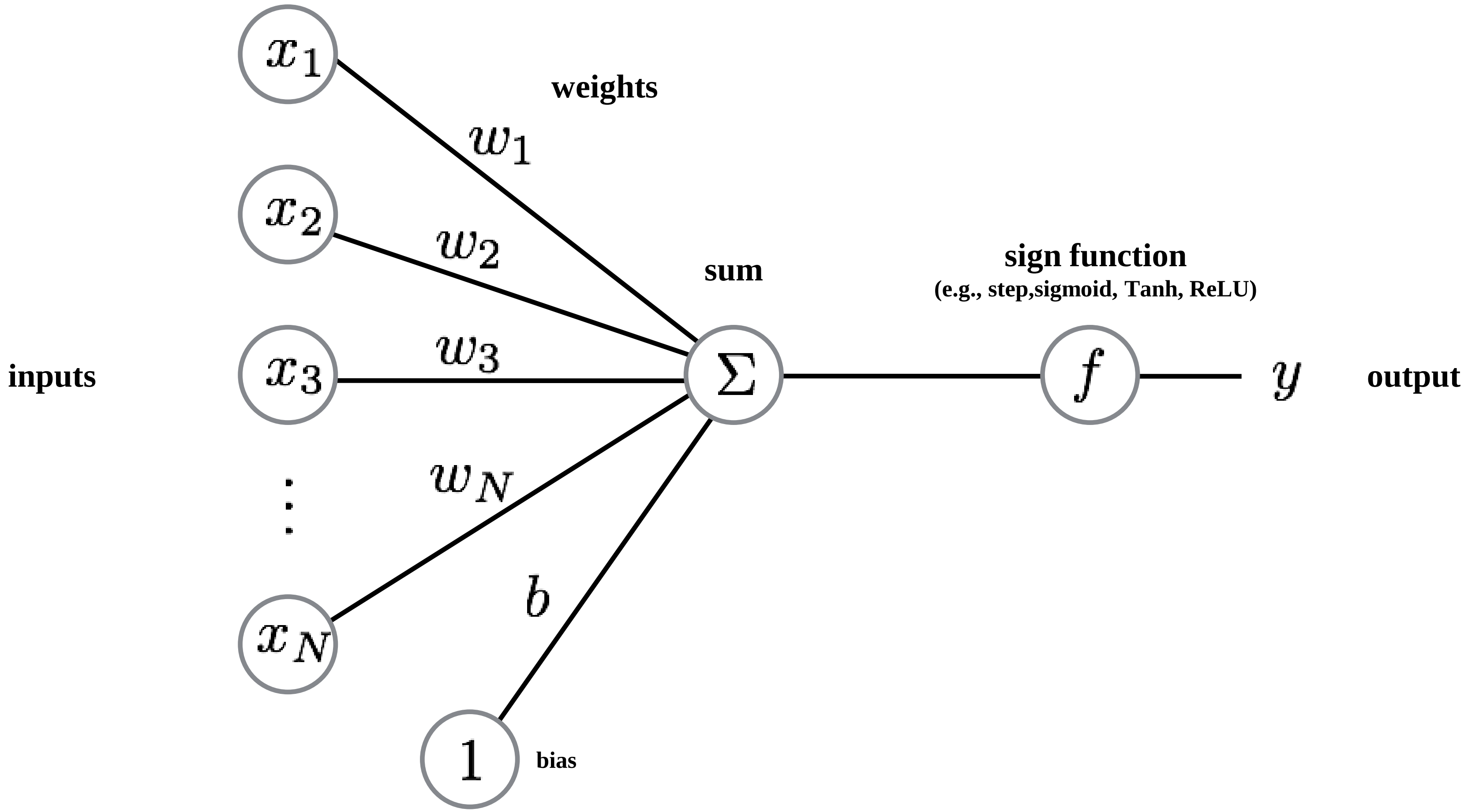
Electrical and Computer Engineering Department
Tarbiat Modares University

Deep Feedforward Networks

Foad Ghaderi, PhD

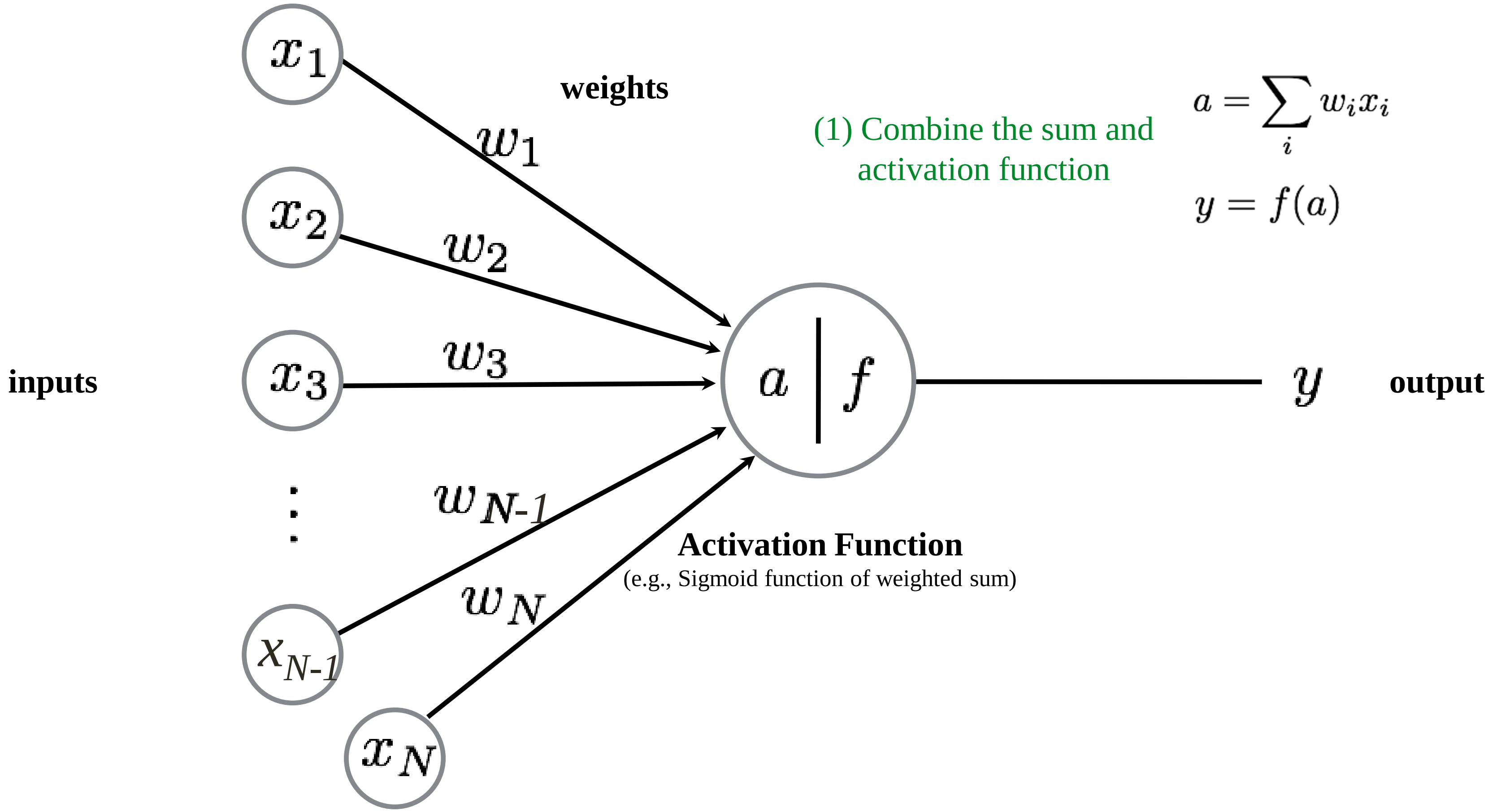
Feedforward networks

The Perceptron



Feedforward networks

The Perceptron



(1) Combine the sum and activation function

$$a = \sum_i w_i x_i$$

$$y = f(a)$$

(2) suppress the bias term (less clutter)

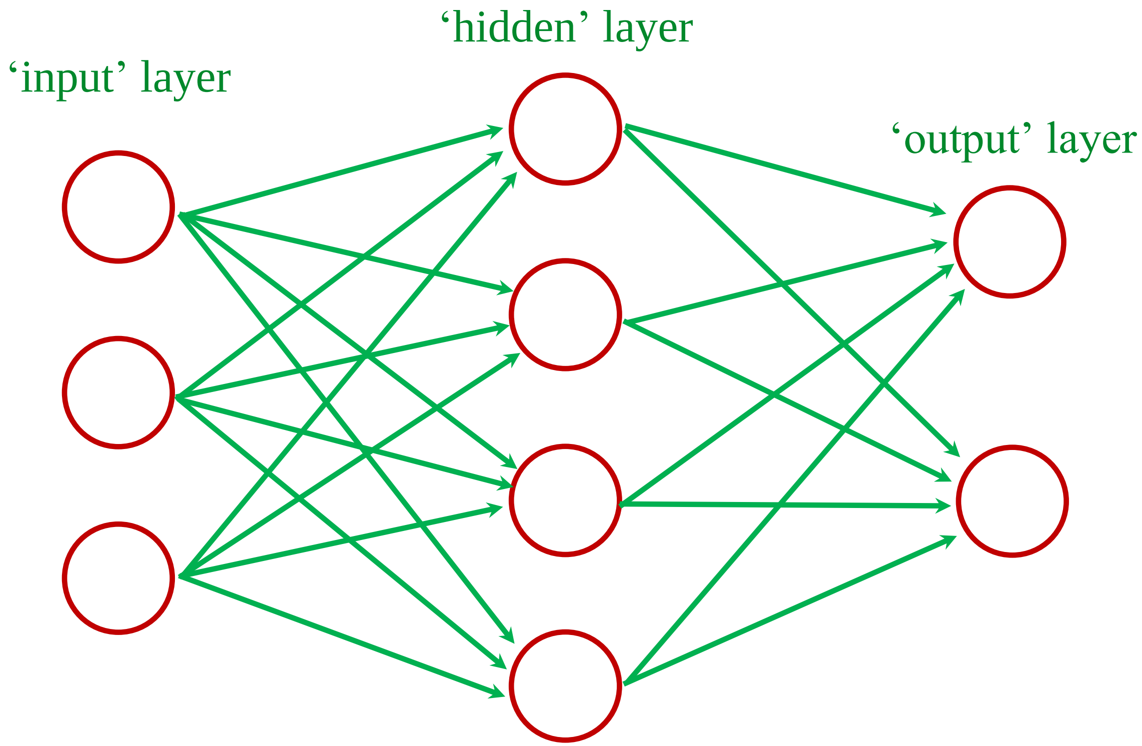
$$x_N = 1$$

$$w_N = b$$

Feedforward networks

A feedforward network (*neural networks*, or *multilayer perceptrons* (*MLPs*)) defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation.

… a collection of connected perceptrons

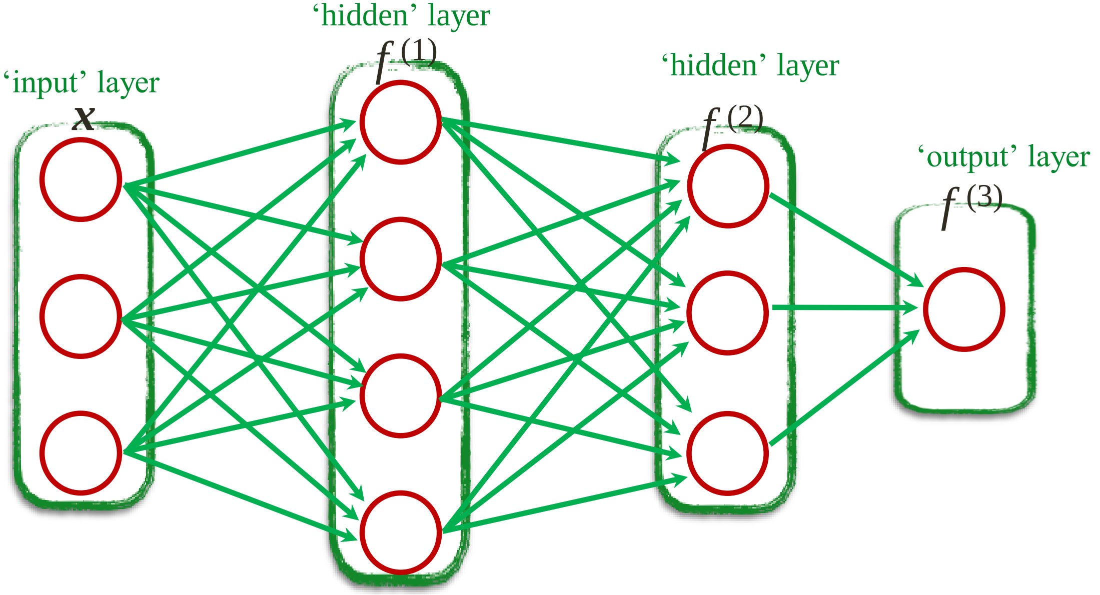


Feedforward networks

There are no *feedback* connections in which outputs of the model are fed back into itself.

The model is associated with a directed acyclic graph describing how the functions are composed together.

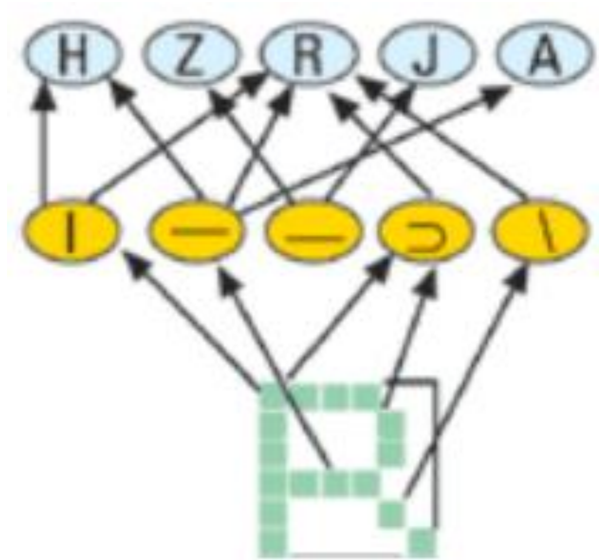
Example, we might have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$.



Feedforward networks

Example:

Optical Character Recognition (OCR)

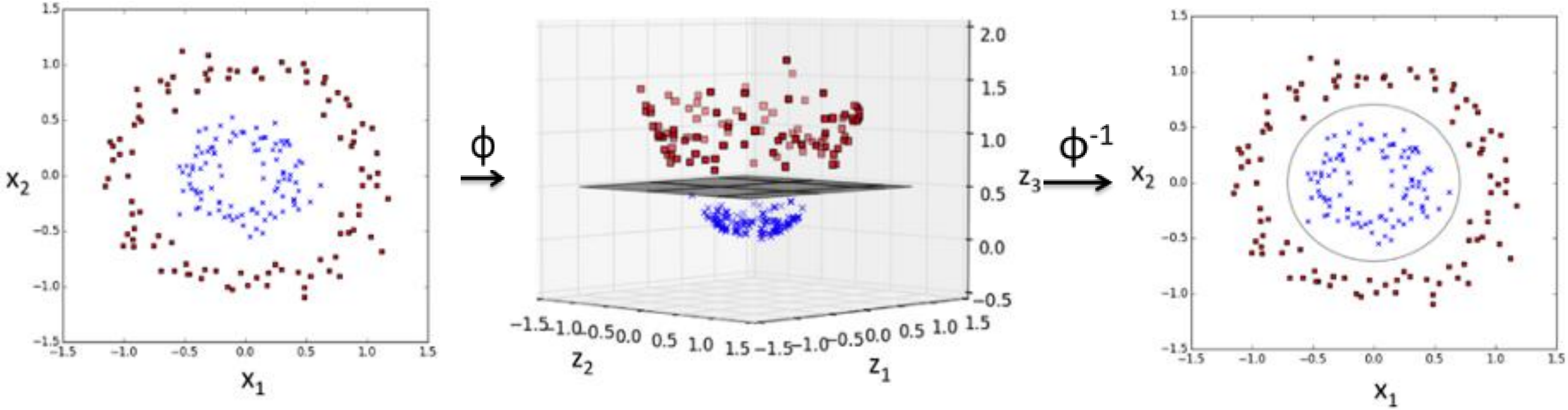


Feedforward networks

Simplest feedforward networks are linear models e.g. logistic regression and linear regression.

- ✓ May fit efficiently and reliably, either in closed form or with convex optimization.
- ✘ The model capacity is limited to linear functions.

To extend linear models to represent nonlinear functions of $\boldsymbol{x}$, we can apply the linear model not to $\boldsymbol{x}$ itself but to a transformed input $\phi(\boldsymbol{x})$, where ϕ is a nonlinear transformation.



Feedforward networks

The question is then how to choose the mapping ϕ.

1. One option is to use a very generic ϕ, such as the infinite-dimensional ϕ that is implicitly used by kernel machines based on the RBF kernel.
   - ✘ Generalization to the test set often remains poor
   - ✘ Do not encode enough prior information to solve advanced problems
2. To manually engineer ϕ
   - ✘ Domain specific
   - ✘ With little transfer between domains.
3. To learn ϕ (the strategy of deep learning)

$$y = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^\top \boldsymbol{w}$$

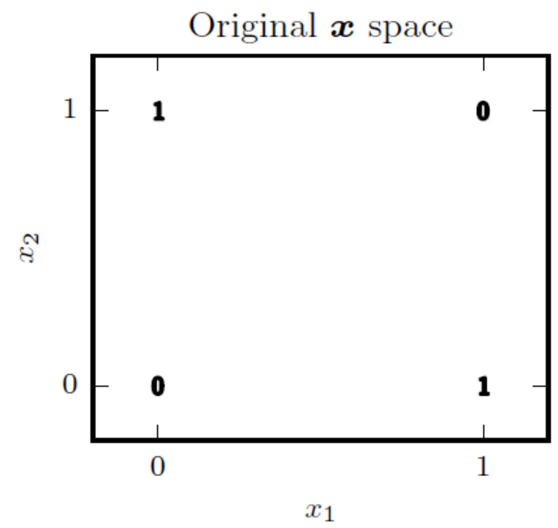
- ✘ Gives up on the convexity of the training problem
- ✓ The human designer only needs to find the right general function family

Parameters:

- ❑ $\boldsymbol{\theta}$ is used to learn ϕ from a broad class of functions,
- ❑ $\boldsymbol{w}$ maps from $\phi(\boldsymbol{x})$ to the desired output.

Feedforward networks

Example: Learning XOR



The XOR function provides the target function $y = f^*(\boldsymbol{x})$ that we want to learn.

Our model provides a function $y = f(\boldsymbol{x};\theta)$ and our learning algorithm will adapt the parameters θ to make f as similar as possible to f^*.

- ❑ No concerns about statistical generalization.
- ❑ Perform correctly on X = {$[0,0]^T$, $[0,1]^T$, $[1,0]^T$, and $[1,1]^T$}.
- ❑ The only challenge is to fit the training set.

Feedforward networks

Example: Learning XOR

Solution 1: Regression (mean squared error loss function)

$$J(\boldsymbol{\theta}) = \frac{1}{4} \sum_{\boldsymbol{x} \in \mathbb{X}} \left(f^*(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\theta})\right)^2$$

The proposed linear model for *f* is:

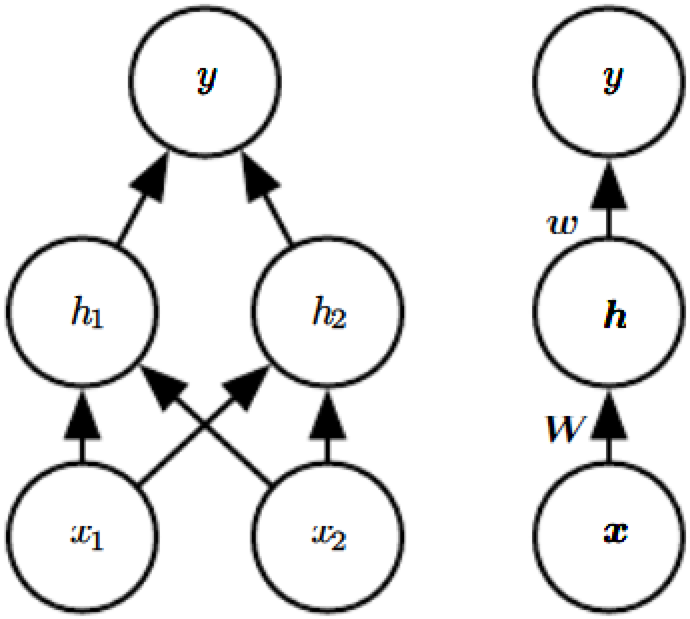
$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{x}^\top \boldsymbol{w} + b$$

Using normal equations we obtain ***w*** = 0 and b = 0.5.

Feedforward networks

Example: Learning XOR
Solution 2: A feedforward network with one hidden layer

The network contains two functions chained together:
$\boldsymbol{h} = f^{(1)}(\boldsymbol{x};\boldsymbol{W}, \boldsymbol{c})$ and
$y = f^{(2)}(\boldsymbol{h};\boldsymbol{w}, \mathrm{b})$,
with the complete model being
$f(\boldsymbol{x};\boldsymbol{W}, \boldsymbol{c},\boldsymbol{w}, \mathrm{b}) = f^{(2)}(f^{(1)}(\boldsymbol{x}))$.



A nonlinear function is required:
For example: an affine transformation controlled by learned parameters, followed by a fixed, nonlinear function (activation function)

$$h = g(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c})$$

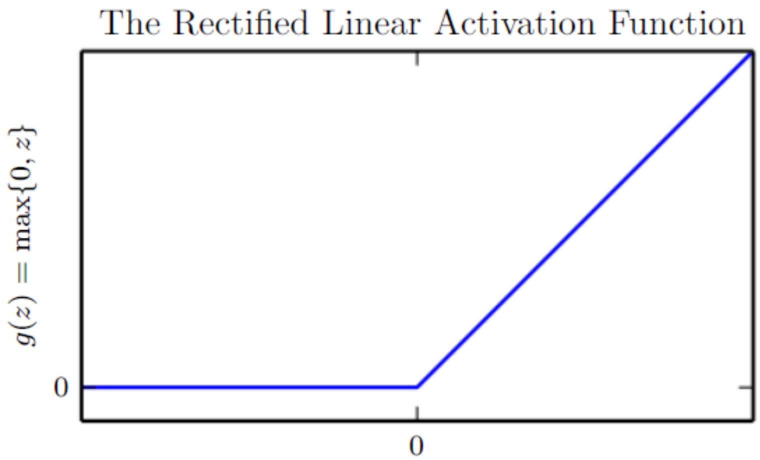
Feedforward networks

Example: Learning XOR

Solution 2: A feedforward network with one hidden layer

$h = g(\boldsymbol{W}^{\top}\boldsymbol{x} + \boldsymbol{c})$

g is the activation function (the default recommendation is to use the *rectified linear unit* or ReLU: $g(z) = max\{0, z\}$)



$$\rightarrow \quad f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^{\top} \max\{0, \boldsymbol{W}^{\top}\boldsymbol{x} + \boldsymbol{c}\} + b$$

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \quad b = 0$$

Feedforward networks

Example: Learning XOR
Solution 2: A feedforward network with one hidden layer

$$f(\boldsymbol{x};\boldsymbol{W},\boldsymbol{c},\boldsymbol{w},b)=\boldsymbol{w}^\top \max\{0,\boldsymbol{W}^\top\boldsymbol{x}+\boldsymbol{c}\}+b$$

$$\boldsymbol{W}=\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix} \quad \boldsymbol{c}=\begin{bmatrix}0\\ -1\end{bmatrix} \quad \boldsymbol{w}=\begin{bmatrix}1\\ -2\end{bmatrix} \quad b=0$$

Assume $\boldsymbol{X}$ is the vector of all possible inputs

$$\boldsymbol{X}=\begin{bmatrix}0 & 0\\ 0 & 1\\ 1 & 0\\ 1 & 1\end{bmatrix} \rightarrow \boldsymbol{XW}=\begin{bmatrix}0 & 0\\ 1 & 1\\ 1 & 1\\ 2 & 2\end{bmatrix} \xrightarrow{+c} \begin{bmatrix}0 & -1\\ 1 & 0\\ 1 & 0\\ 2 & 1\end{bmatrix}$$

all samples lie along a line with slope 1.

Feedforward networks

Example: Learning XOR

Solution 2: A feedforward network with one hidden layer

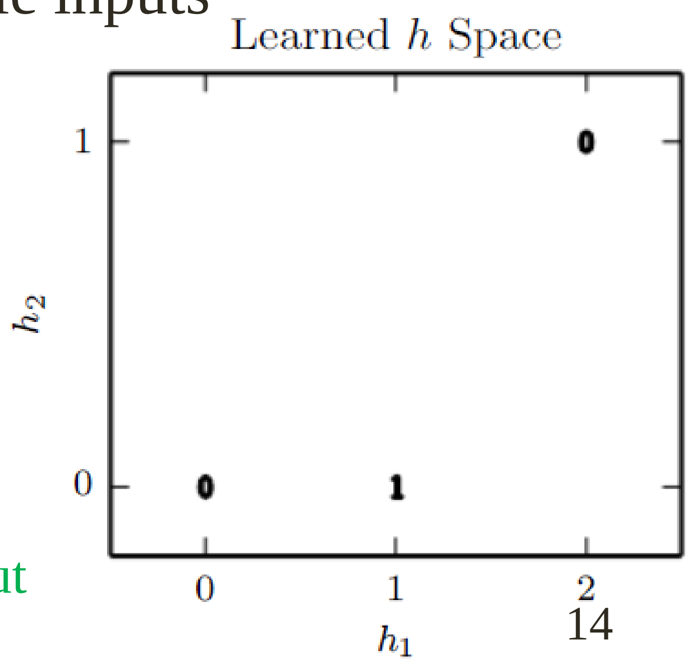
$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^\top \max\{0, \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c}\} + b$$

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \quad b = 0$$

Assume ***X*** is the vector of all possible inputs

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \xrightarrow{\text{apply ReLU}} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

$$\xrightarrow{\text{multiply by } \boldsymbol{w}} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$



For fixed h_2, output increases in h_1

Feedforward networks

Example: Learning XOR
Summary:

- Selecting alternative solution resulted in zero error
- In real world application there are many parameters. It is not easy to guess.
- Gradient descent optimization methods can estimate parameters with little errors