

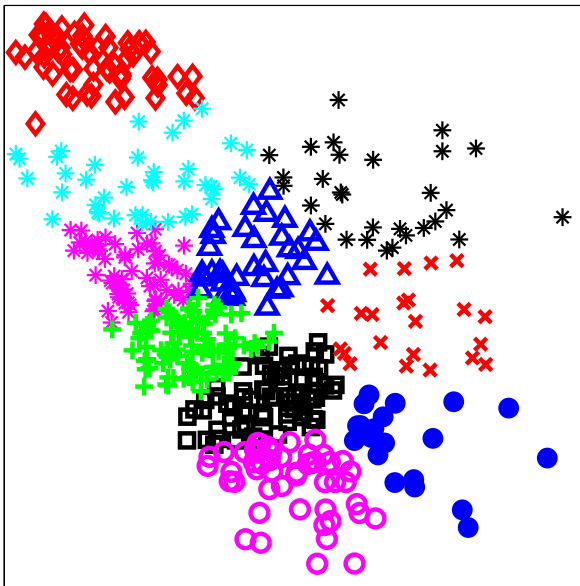
**Learning From Data**  
**Lecture 20**  
**Multilayer Perceptron**

Multiple layers  
Universal Approximation  
The Neural Network

**M. Magdon-Ismail**  
CSCI 4100/6100

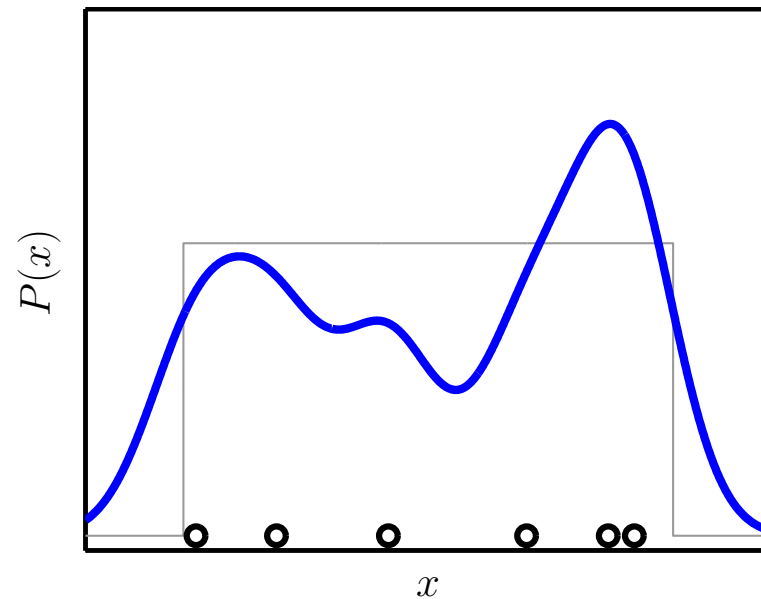
# RECAP: Unsupervised Learning

## $k$ -Means Clustering



‘Hard’ partition into  $k$ -clusters

## Gaussian Mixture Model



‘Soft’ probability density estimation

# The Neural Network - Biologically Inspired



[Krizhevsky, Sutskever and Hinton, 2012]

A brief history of Neural Networks.

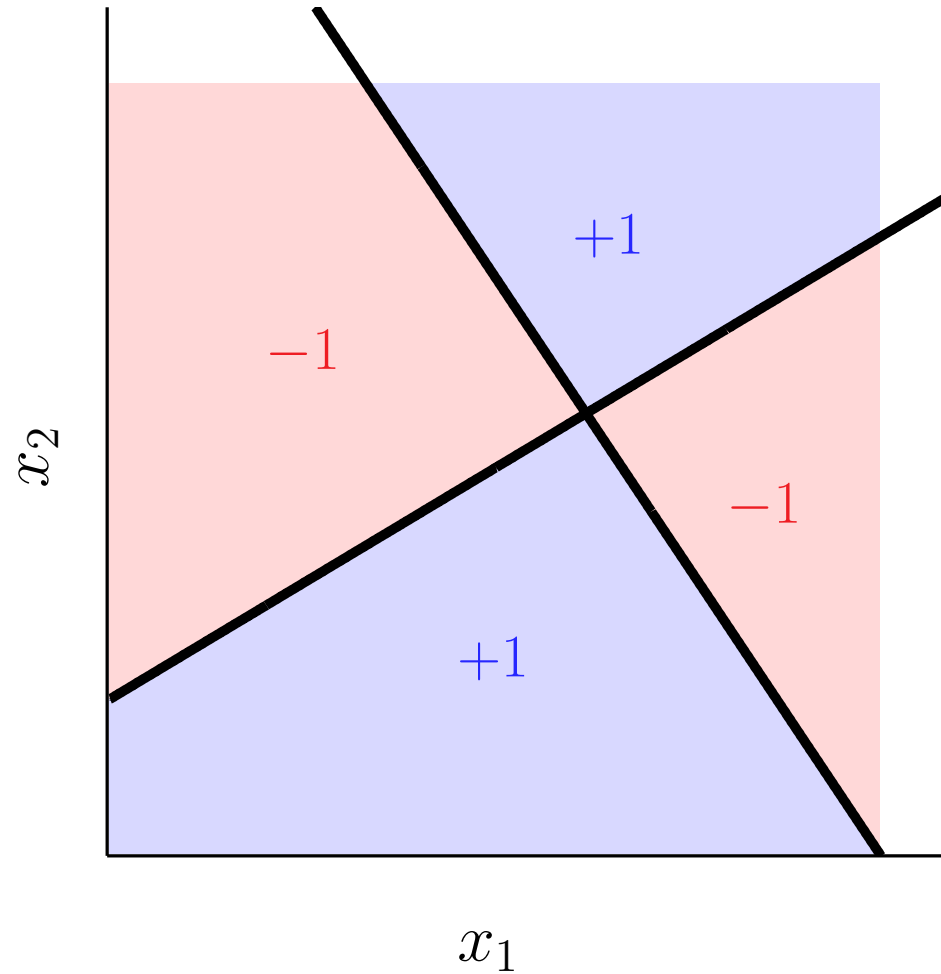
1943	1958	1969	1982	1986	1992-1995	2006	2012
McCulloch & Pitt (Neural Nets)	Rosenblatt (Perceptron)	Minsky & Pappert XOR, fitting	Hopfield Network	Hinton et. al (MLP, Backprop.)	Vapnik (SVM)	Hinton et. al. (Deep Belief Nets)	Hinton et. al. (DeepNets, Vision)

# Planes Don't Flap Wings to Fly

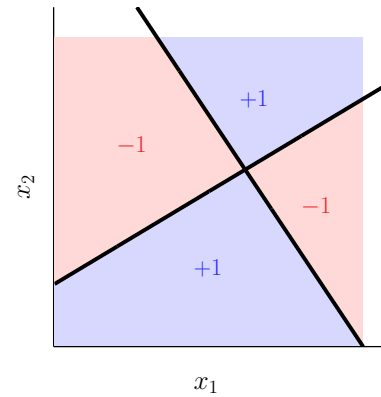


Engineering success may start with biological inspiration, but then take a totally different path.

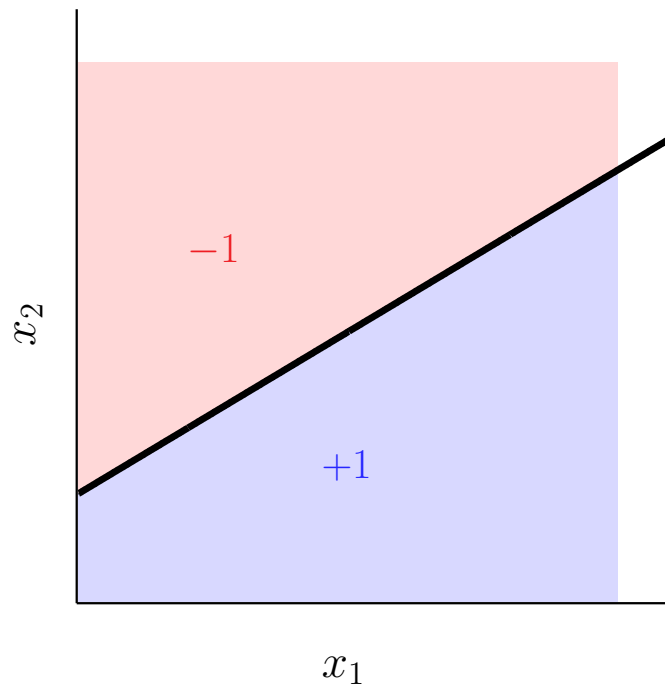
# XOR: A Limitation of the Linear Model



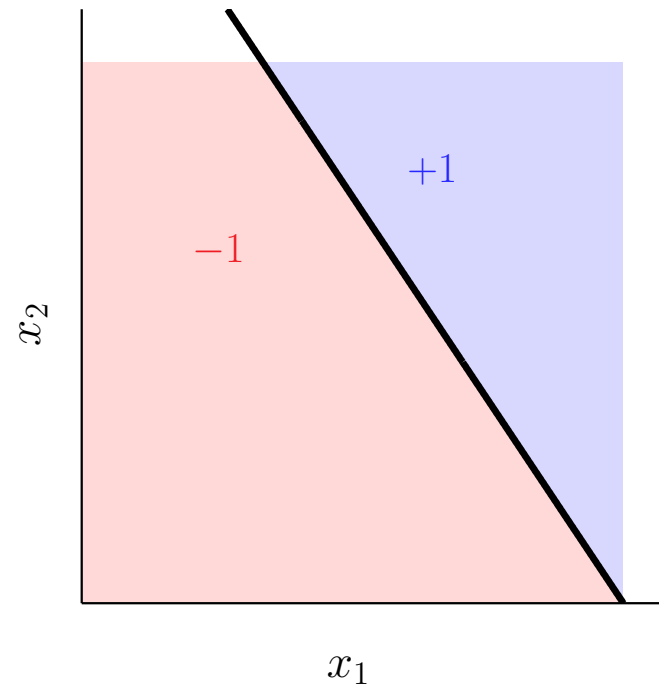
# Decomposing XOR



$$f = h_1 \bar{h}_2 + \bar{h}_1 h_2$$



$$h_1(\mathbf{x}) = \text{sign}(\mathbf{w}_1^T \mathbf{x})$$

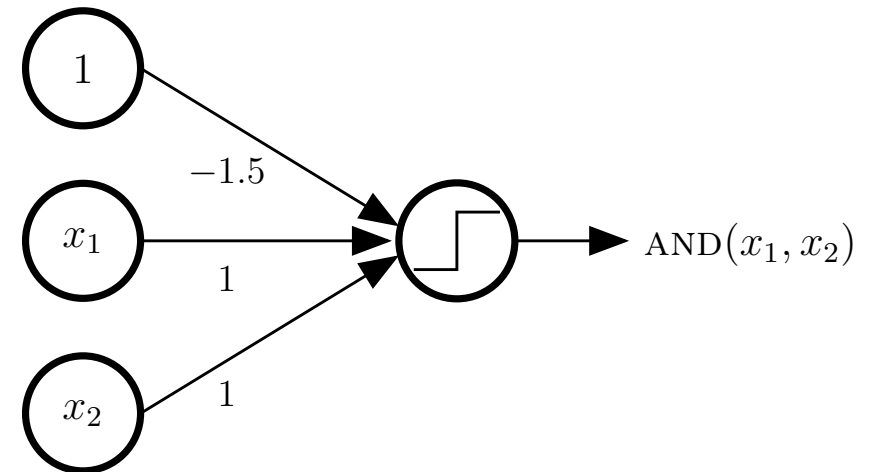
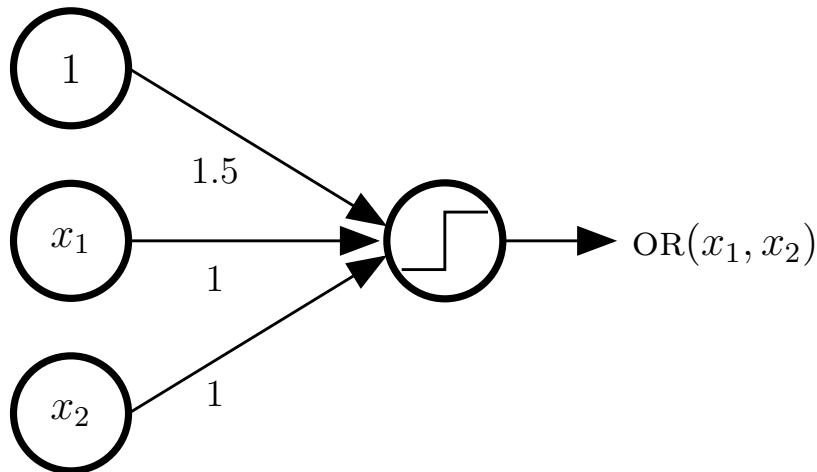


$$h_2(\mathbf{x}) = \text{sign}(\mathbf{w}_2^T \mathbf{x})$$

# Perceptrons for OR and AND

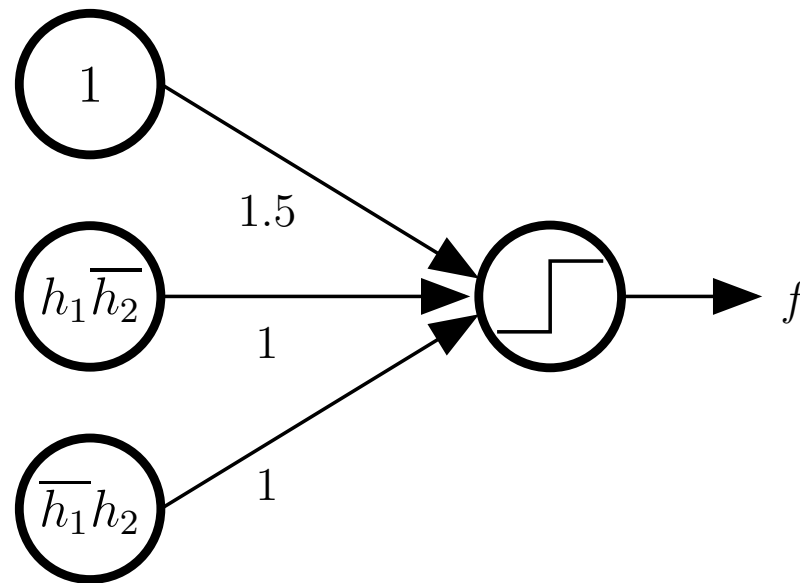
$$\text{OR}(x_1, x_2) = \text{sign}(x_1 + x_2 + 1.5)$$

$$\text{AND}(x_1, x_2) = \text{sign}(x_1 + x_2 - 1.5)$$



# Representing $f$ Using OR and AND

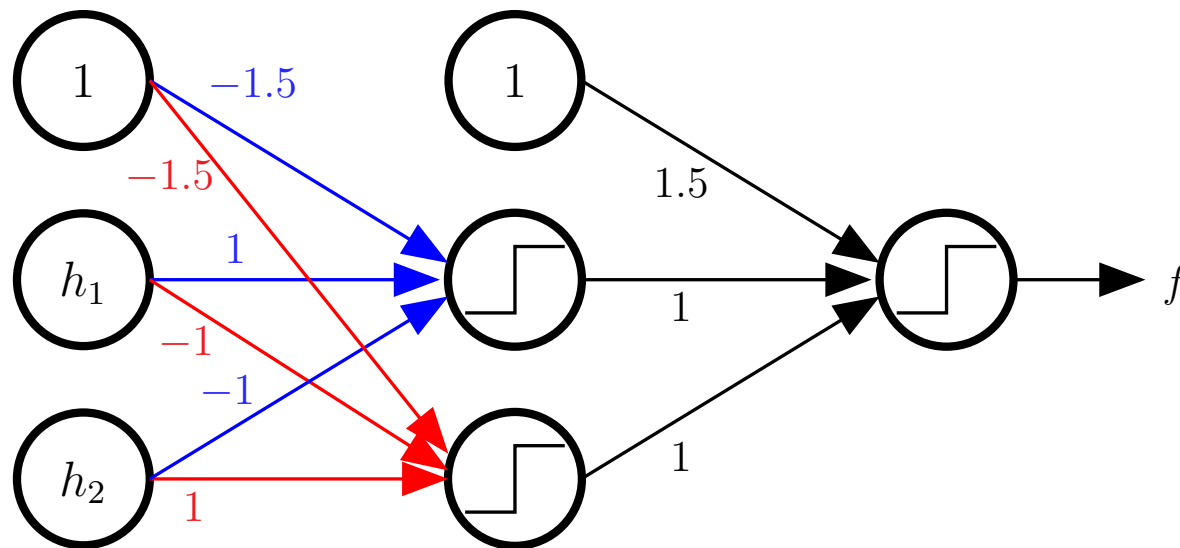
$$f = h_1\bar{h}_2 + \bar{h}_1h_2$$





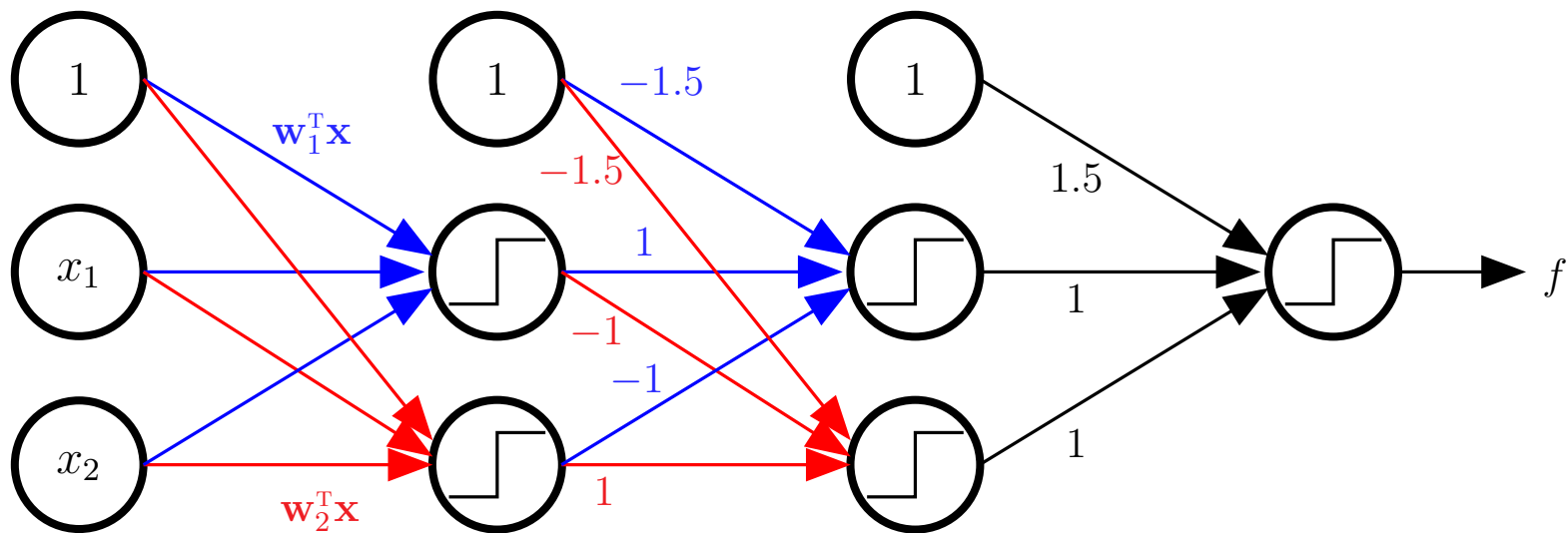
# Representing $f$ Using OR and AND

$$f = h_1\overline{h_2} + \overline{h_1}h_2$$

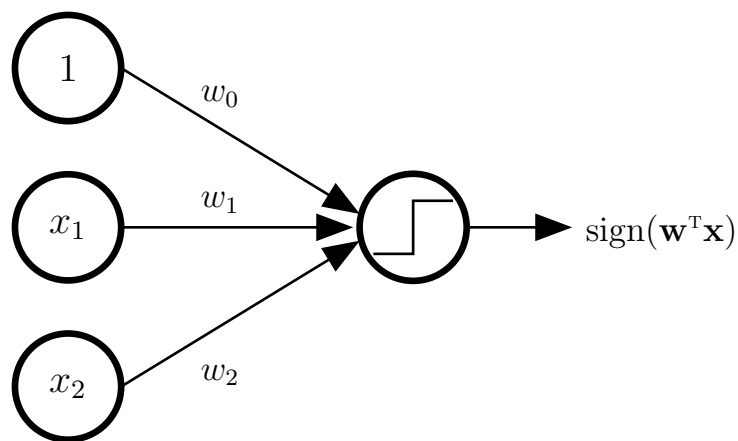
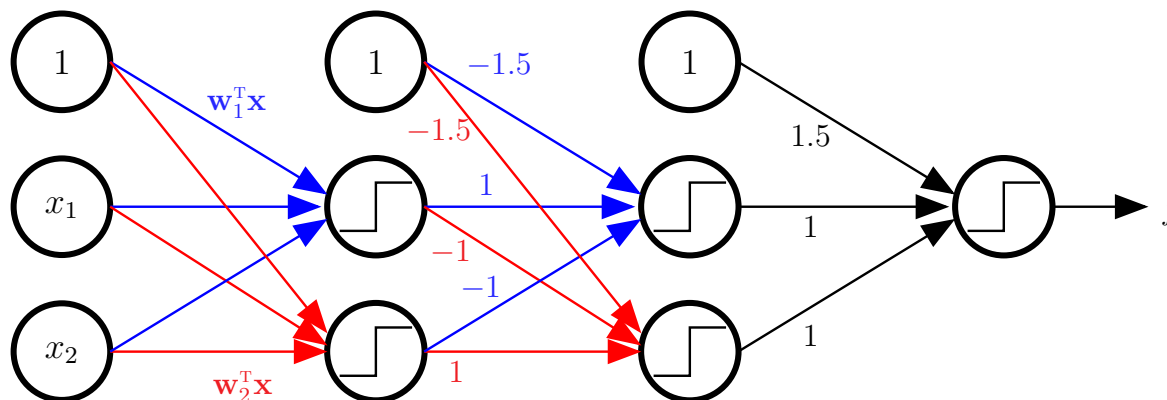


# Representing $f$ Using OR and AND

$$f = h_1 \bar{h}_2 + \bar{h}_1 h_2$$



# The Multilayer Perceptron (MLP)



More layers allow us to implement  $f$

These additional layers are called *hidden layers*

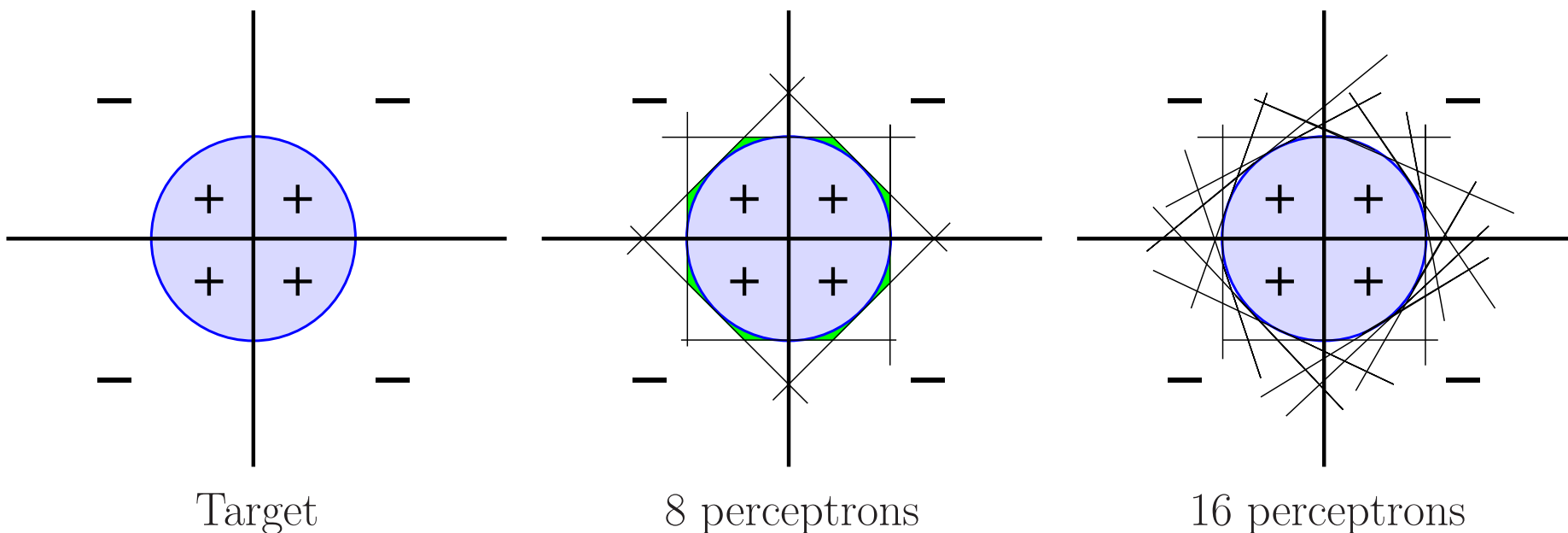
---

# Universal Approximation

Any target function  $f$  that can be decomposed into linear separators can be implemented by a 3-layer MLP.

# Universal Approximation

A sufficiently smooth separator can “essentially” be decomposed into linear separators.



---

# Approximation Versus Generalization

The size of the MLP controls the approximation-generalization tradeoff.

More nodes per hidden layer  $\implies$  approximation $\uparrow$  and generalization $\downarrow$

---

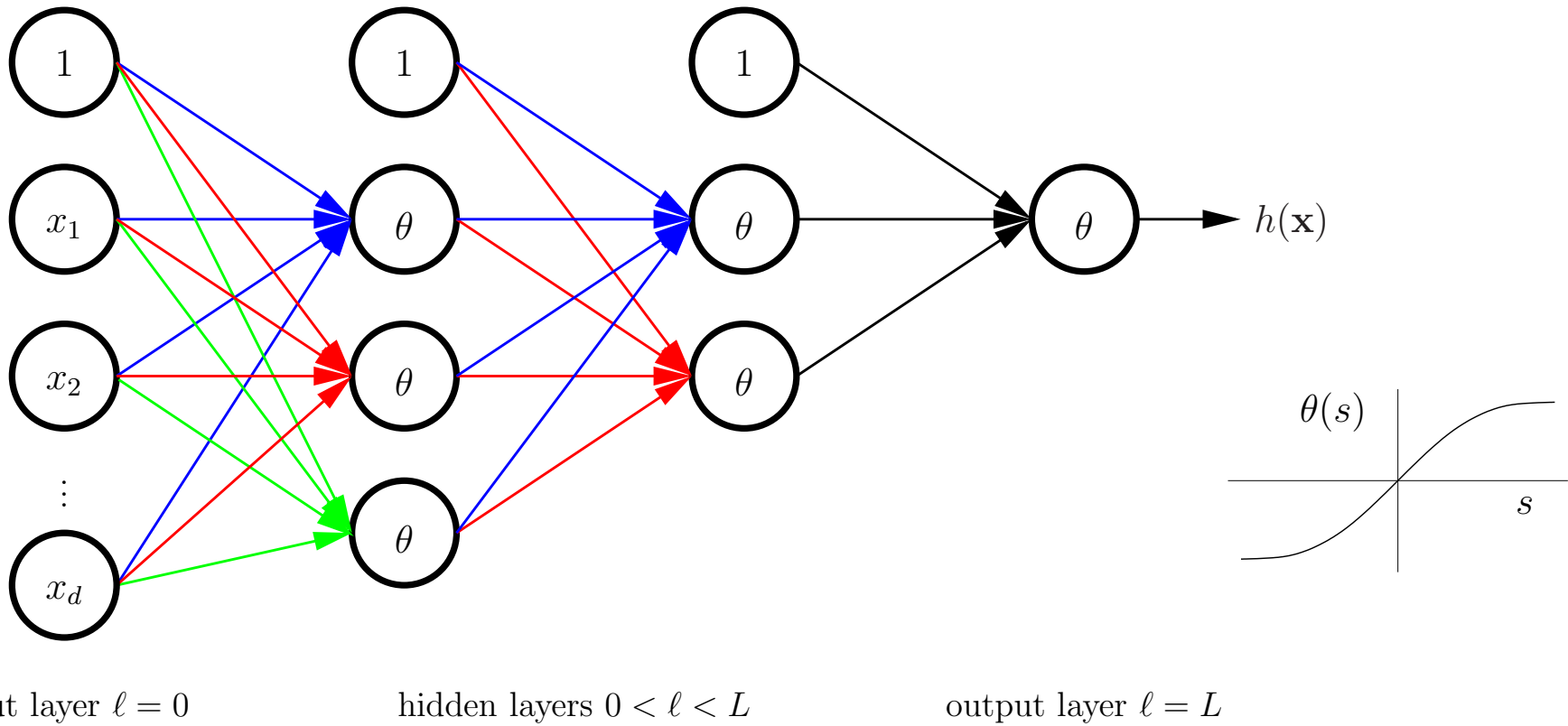
# Minimizing $E_{\text{in}}$

A combinatorial problem even harder with the MLP than the Perceptron.

$E_{\text{in}}$  is not smooth (due to sign function), so cannot use gradient descent.

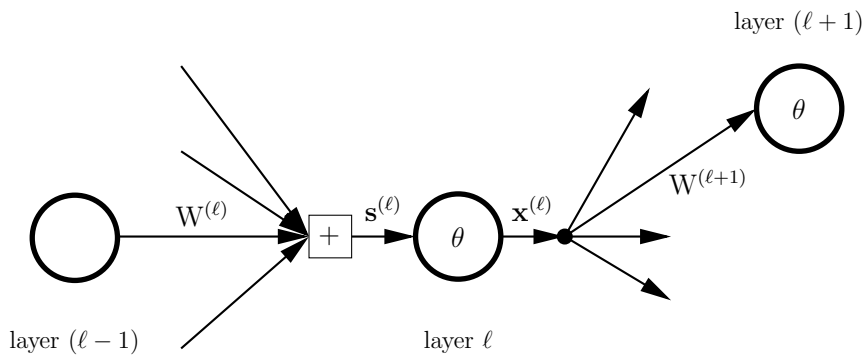
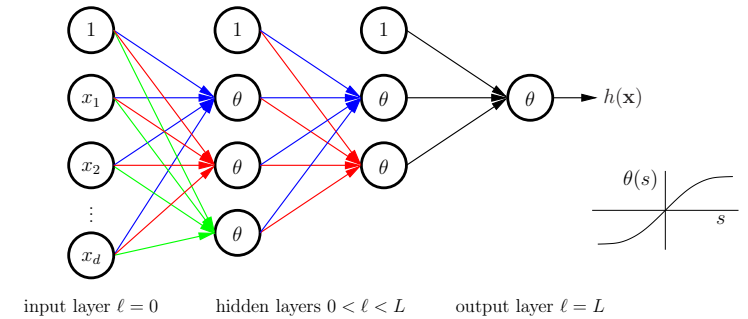
$\text{sign}(x) \approx \tan(x) \longrightarrow$  gradient descent to minimize  $E_{\text{in}}$ .

# The Neural Network





# Zooming into a Hidden Node



layer  $\ell$  parameters

signals in	$\mathbf{s}^{(\ell)}$	$d^{(\ell)}$ dimensional input vector
outputs	$\mathbf{x}^{(\ell)}$	$d^{(\ell)} + 1$ dimensional output vector
weights in	$\mathbf{W}^{(\ell)}$	$(d^{(\ell-1)} + 1) \times d^{(\ell)}$ dimensional matrix
weights out	$\mathbf{W}^{(\ell+1)}$	$(d^{(\ell)} + 1) \times d^{(\ell+1)}$ dimensional matrix

layers  $\ell = 0, 1, 2, \dots, L$

layer  $\ell$  has "dimension"  $d^{(\ell)} \implies d^{(\ell)} + 1$  nodes

$$\mathbf{W}^{(\ell)} = \begin{bmatrix} \mathbf{w}_1^{(\ell)} & \mathbf{w}_2^{(\ell)} & \cdots & \mathbf{w}_{d^{(\ell)}}^{(\ell)} \\ | & | & \vdots & | \end{bmatrix}$$

# The Neural Network

