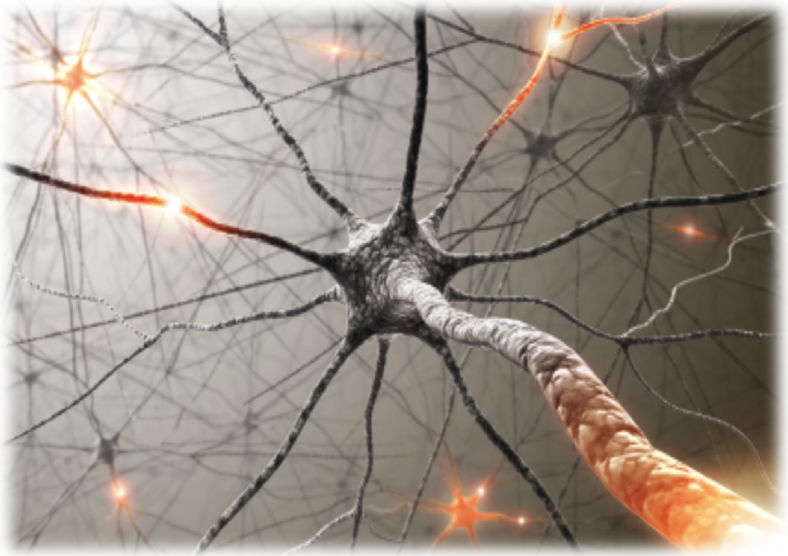
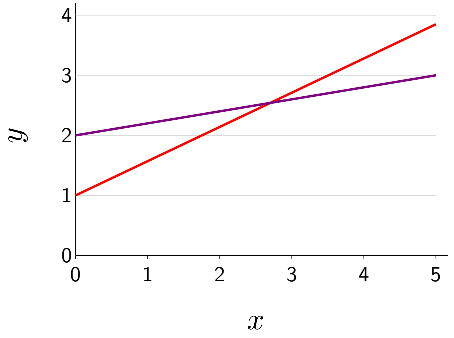
Machine learning: neural networks



Non-linear predictors

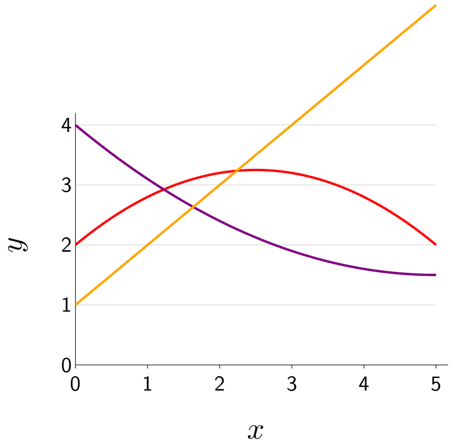
Linear predictors:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x), \ \phi(x) = [1, x]$$



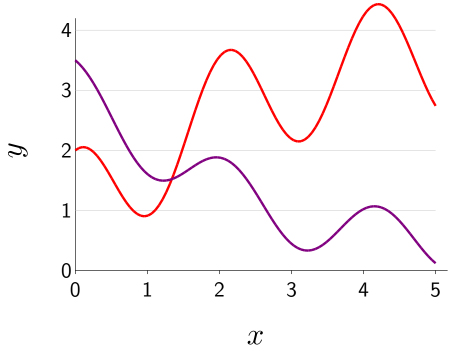
Non-linear (quadratic) predictors:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x), \ \phi(x) = [1, x, x^2]$$



Non-linear neural networks:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \sigma(\mathbf{V}\phi(x)), \ \phi(x) = [1, x]$$



Motivating example

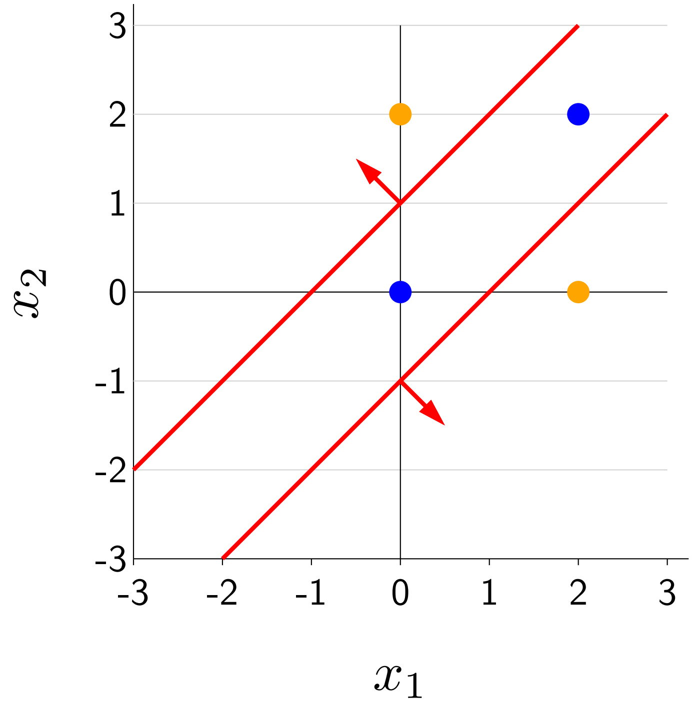
Example: predicting car collision

Input: positions of two oncoming cars $x = [x_1, x_2]$

Output: whether safe ($y = +1$) or collide ($y = -1$)

Unknown: safe if cars sufficiently far: $y = \text{sign}(|x_1 - x_2| - 1)$

x_1	x_2	y
0	2	1
2	0	1
0	0	-1
2	2	-1



Decomposing the problem

Test if car 1 is far right of car 2:

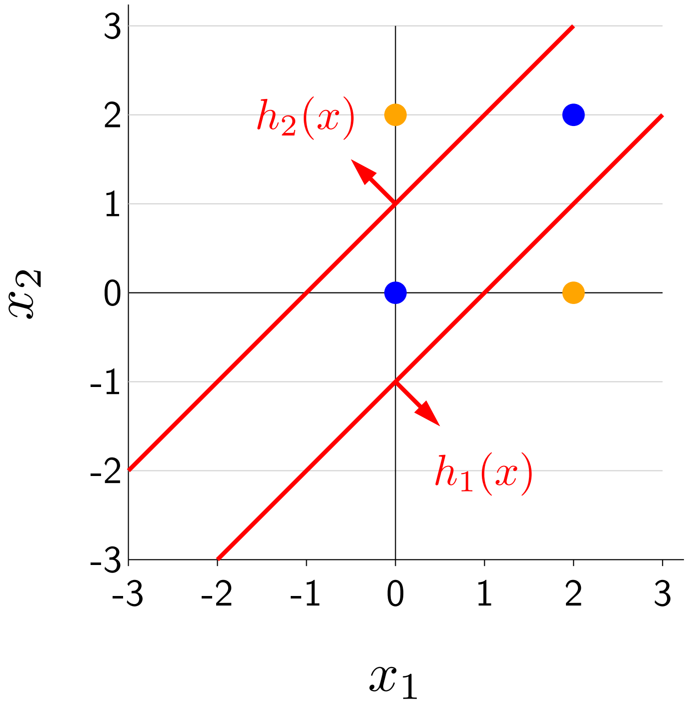
$$h_1(x) = \mathbf{1}[x_1 - x_2 \geq 1]$$

Test if car 2 is far right of car 1:

$$h_2(x) = \mathbf{1}[x_2 - x_1 \geq 1]$$

Safe if at least one is true:

$$f(x) = \text{sign}(h_1(x) + h_2(x))$$



x	$h_1(x)$	$h_2(x)$	$f(x)$
$[0, 2]$	0	1	$+1$
$[2, 0]$	1	0	$+1$
$[0, 0]$	0	0	-1
$[2, 2]$	0	0	-1

Rewriting using vector notation

Intermediate subproblems:

$$h_1(x) = \mathbf{1}[x_1 - x_2 \geq 1] = \mathbf{1}[[-1, +1, -1] \cdot [1, x_1, x_2] \geq 0]$$

$$h_2(x) = \mathbf{1}[x_2 - x_1 \geq 1] = \mathbf{1}[[-1, -1, +1] \cdot [1, x_1, x_2] \geq 0]$$

$$\mathbf{h}(x) = \mathbf{1}\left[\begin{bmatrix} -1 & +1 & -1 \\ -1 & -1 & +1 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \geq 0\right]$$

Predictor:

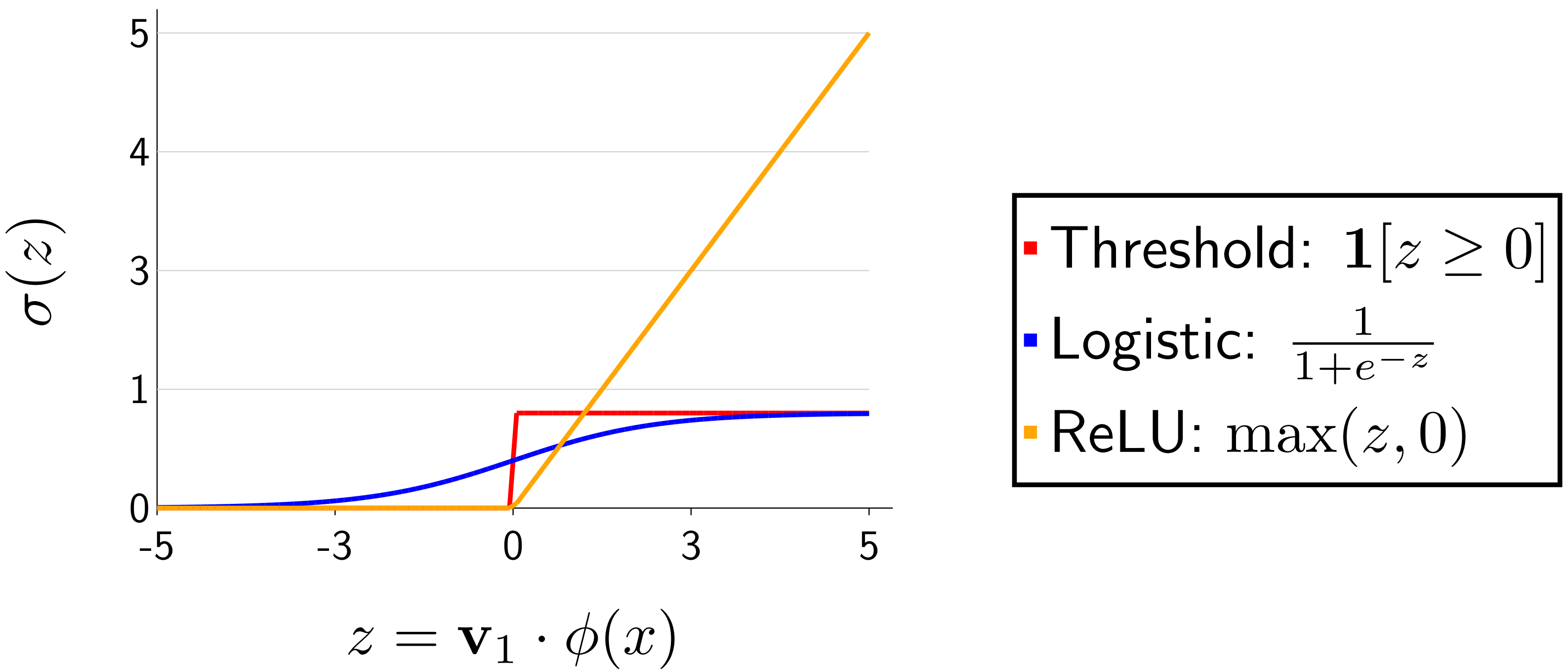
$$f(x) = \mathsf{sign}(h_1(x) + h_2(x)) = \mathsf{sign}([1, 1] \cdot \mathbf{h}(x))$$

Avoid zero gradients

Problem: gradient of $h_1(x)$ with respect to $\mathbf{v}_1$ is 0

$$h_1(x) = \mathbf{1}[\mathbf{v}_1 \cdot \phi(x) \geq 0]$$

Solution: replace with an **activation function** σ with non-zero gradients



$$h_1(x) = \sigma(\mathbf{v}_1 \cdot \phi(x))$$

Two-layer neural networks

Intermediate subproblems:

$$\mathbf{h}(x) = \sigma(\mathbf{V}\phi(x))$$

Predictor (classification):

$$f_{\mathbf{V},\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \mathbf{h}(x))$$

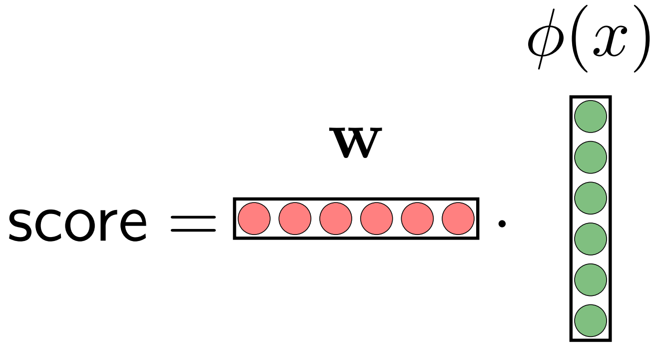
Interpret $\mathbf{h}(x)$ as a learned feature representation!

Hypothesis class:

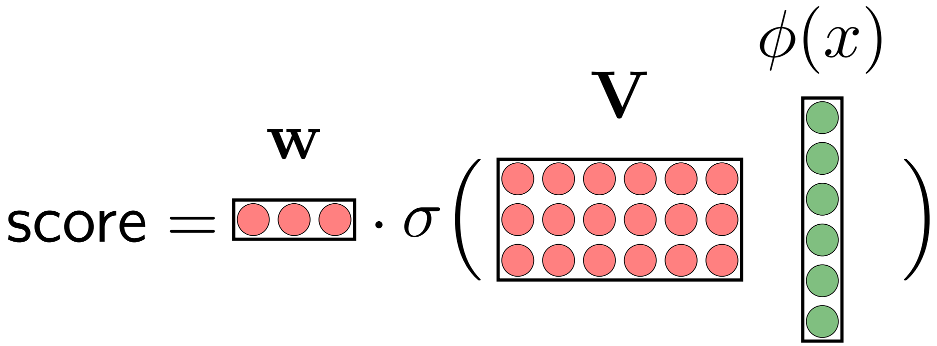
$$\mathcal{F} = \{f_{\mathbf{V},\mathbf{w}} : \mathbf{V} \in \mathbb{R}^{k \times d}, \mathbf{w} \in \mathbb{R}^k\}$$

Deep neural networks

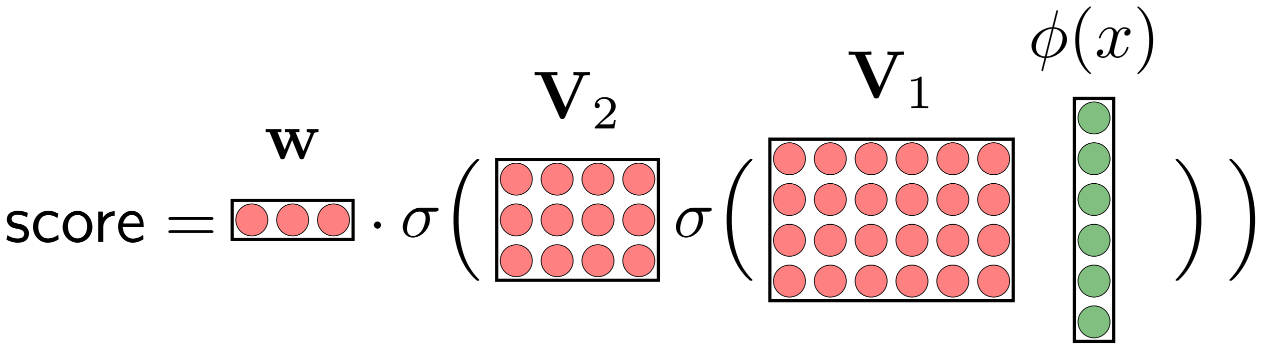
1-layer neural network:



2-layer neural network:

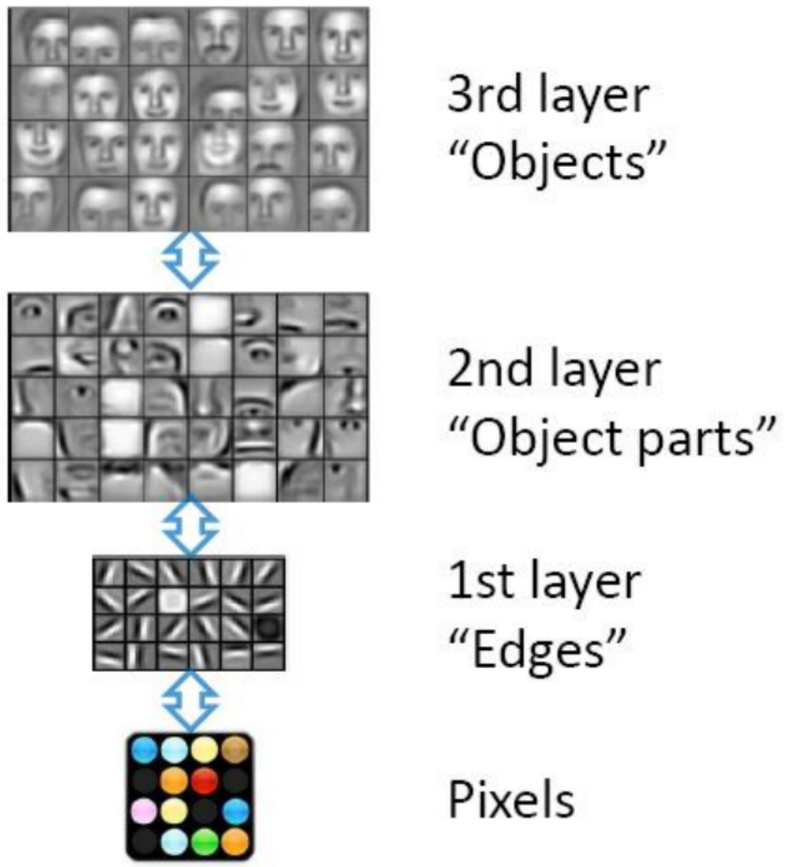


3-layer neural network:

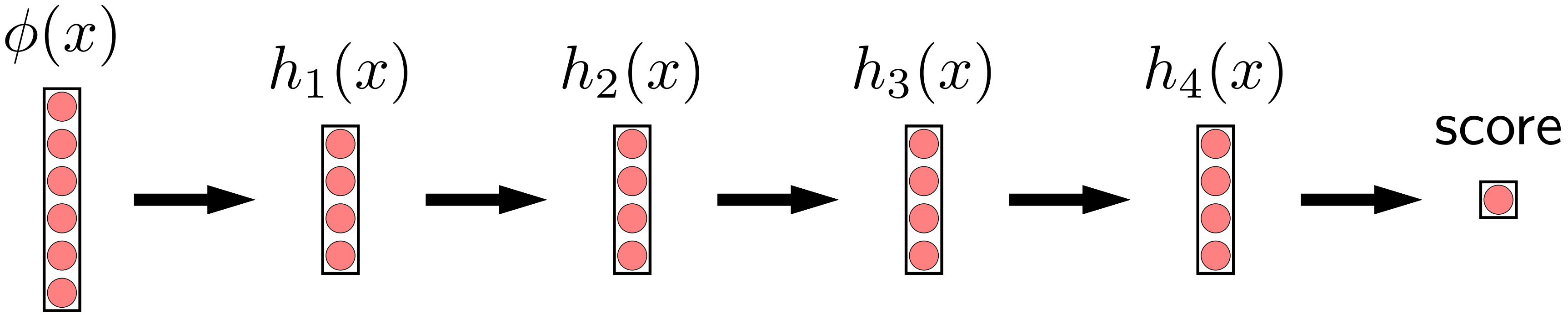


[figure from Honglak Lee]

Layers represent multiple levels of abstractions



Why depth?



Intuitions:

- Multiple levels of abstraction
- Multiple steps of computation
- Empirically works well
- Theory is still incomplete

Summary

$$\text{score} = \mathbf{w} \cdot \sigma(\mathbf{V}\,\phi(x))$$

- Intuition: decompose problem into intermediate parallel subproblems
- Deep networks iterate this decomposition multiple times
- Hypothesis class contains predictors ranging over weights for all layers
- Next up: learning neural networks