

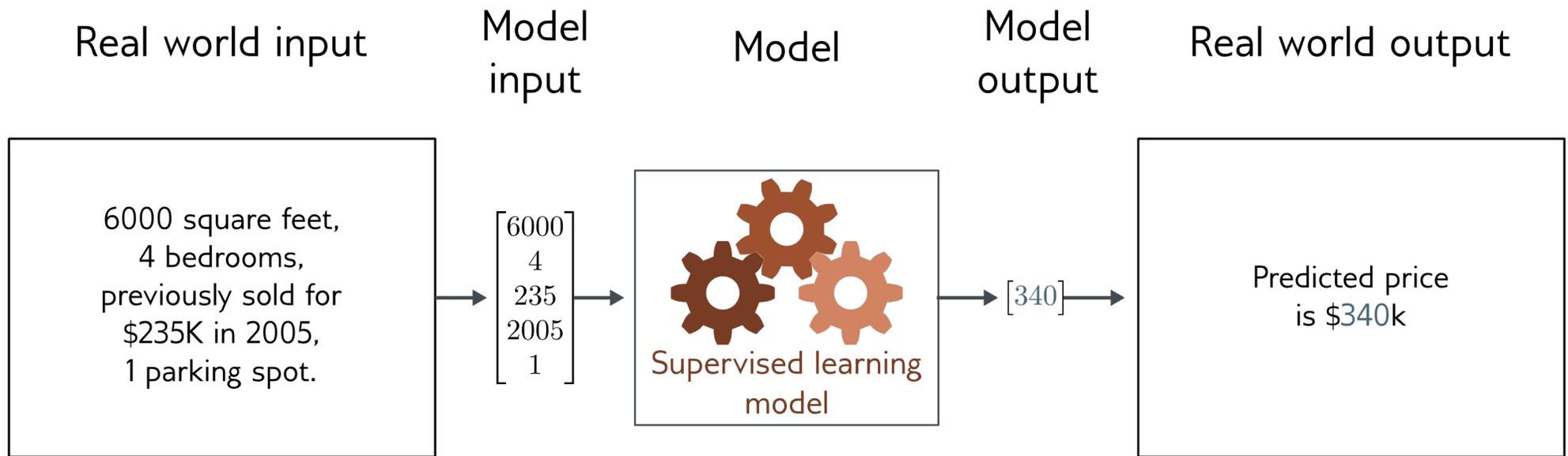


# Lecture 03

# Shallow Networks

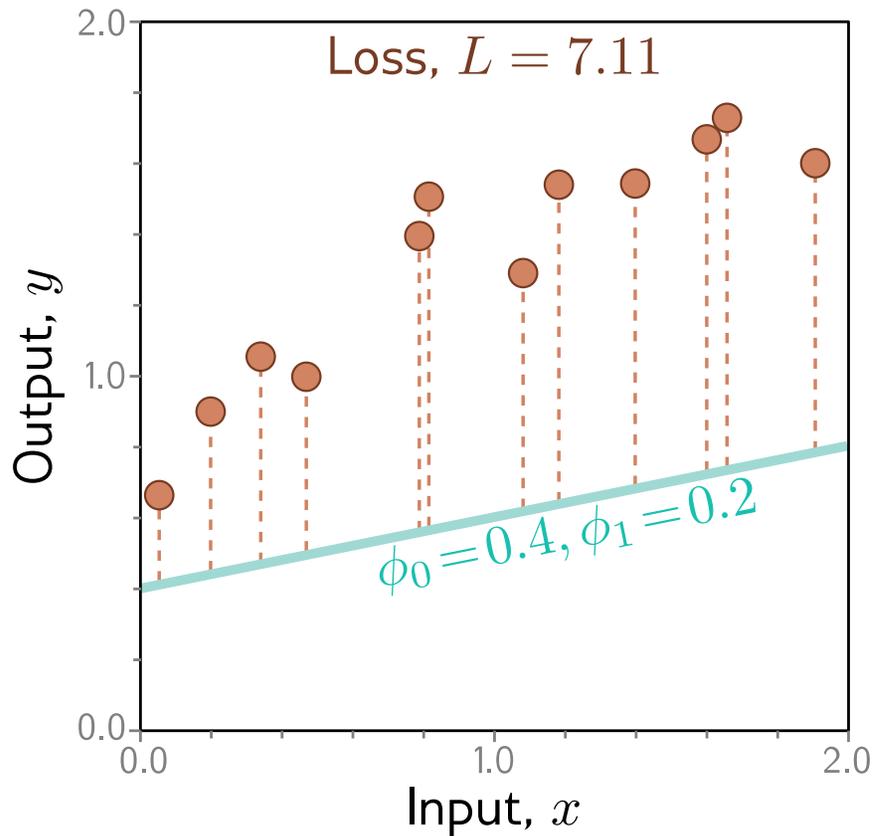
DL4DS – Spring 2026

# Recap: Regression



- Univariate regression problem (one output, real value)
- Fully connected network

# Recap: 1D Linear regression loss function

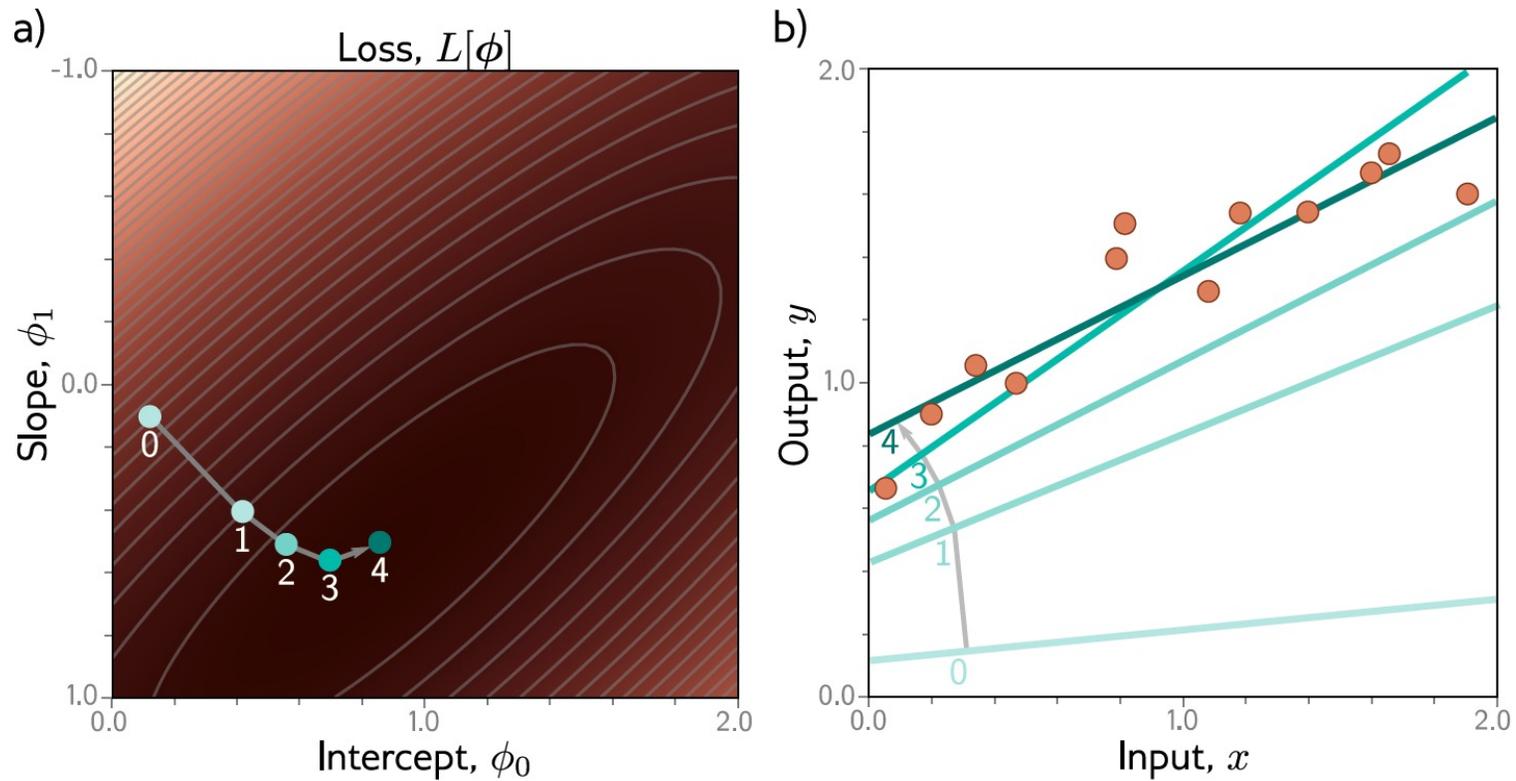


Loss function:

$$L[\phi] = \sum_{i=1}^I (f[x_i, \phi] - y_i)^2$$
$$= \sum_{i=1}^I (\phi_0 + \phi_1 x_i - y_i)^2$$

“Least squares loss function”

# Recap: 1D Linear regression training



This technique is known as **gradient descent**

# Shallow neural networks

- 1D regression model is obviously limited
  - Want to be able to describe input/output that are not lines
  - Want multiple inputs
  - Want multiple outputs
- Shallow neural networks
  - Flexible enough to describe arbitrarily complex input/output mappings
  - Can have as many inputs as we want
  - Can have as many outputs as we want

# This lecture we'll cover...

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology

# 1D Linear Regression

$$\begin{aligned}y &= f[x, \phi] \\ &= \phi_0 + \phi_1 x\end{aligned}$$

## Example shallow network

$$\begin{aligned}y &= f[x, \phi] \\ &= \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x]\end{aligned}$$

## Example shallow network

$$\begin{aligned}y &= f[x, \phi] \\ &= \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x]\end{aligned}$$

---

# Example shallow network

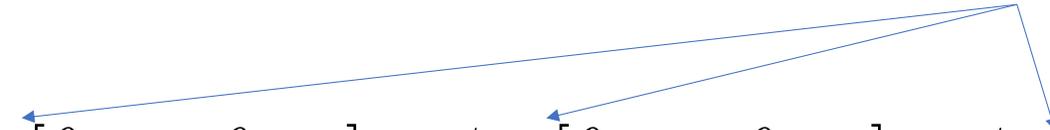
Activation function

$$y = f[x, \phi]$$
$$= \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x]$$

---

# Example shallow network

Activation function

$$y = f[x, \phi]$$
$$= \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x]$$


$$a[z] = \text{ReLU}[z] = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}.$$

**Rectified Linear Unit**  
(one type of activation function)

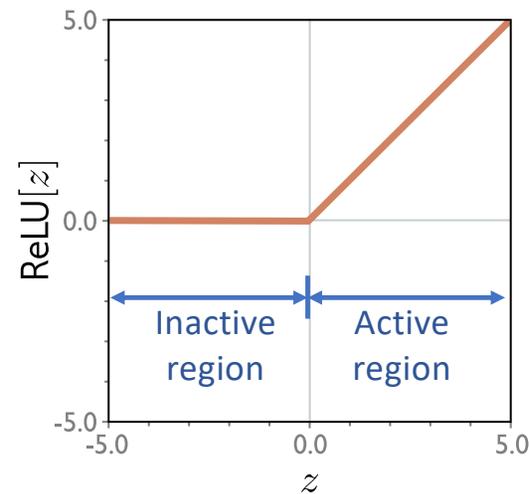
# Example shallow network

$$y = f[x, \phi] \\ = \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x]$$

Activation function

$$a[z] = \text{ReLU}[z] = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}.$$

**Rectified Linear Unit**  
(particular kind of activation function)



# Example shallow network

$$y = f[x, \phi]$$
$$= \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x]$$

---

This model has 10 parameters:

$$\phi = \{\phi_0, \phi_1, \phi_2, \phi_3, \theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}\}$$

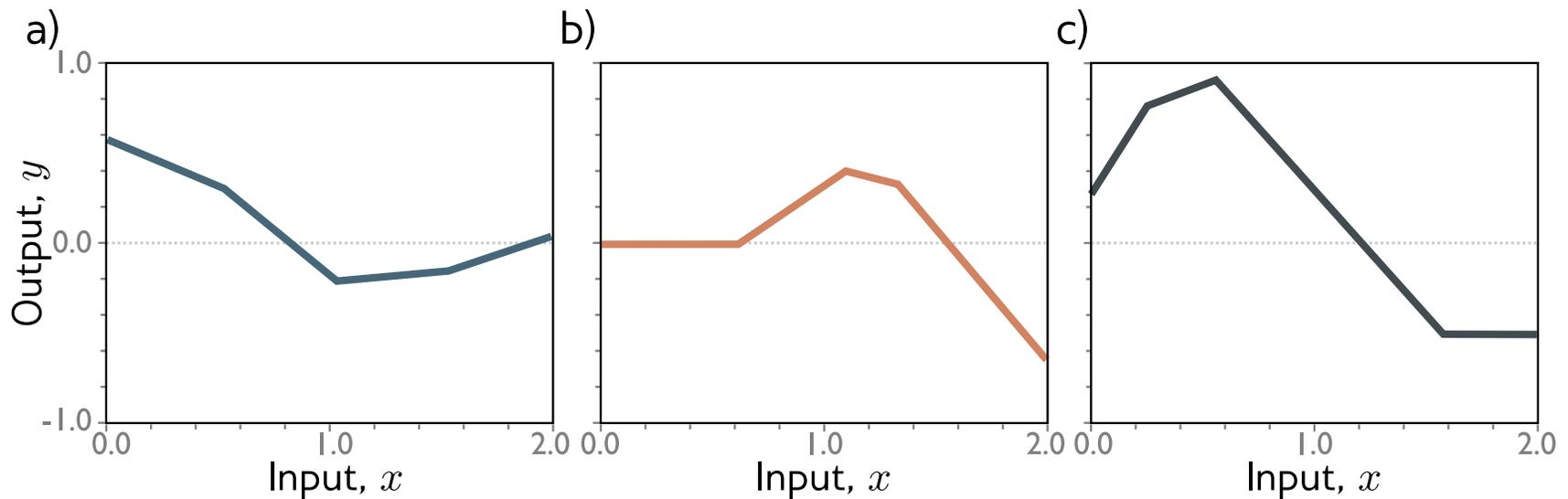
- Represents a family of functions
- Parameters determine a particular function
- Given the parameters, we can perform inference (evaluate the equation)
- Given training dataset  $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^I$
- Define loss function  $L[\phi]$  (least squares)
- Change parameters to minimize loss function

## Example shallow network

$$y = \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x].$$

# Example shallow network

$$y = \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x].$$



Piecewise linear functions with three joints

# Hidden units

$$y = \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x].$$

---

Break down into two parts:

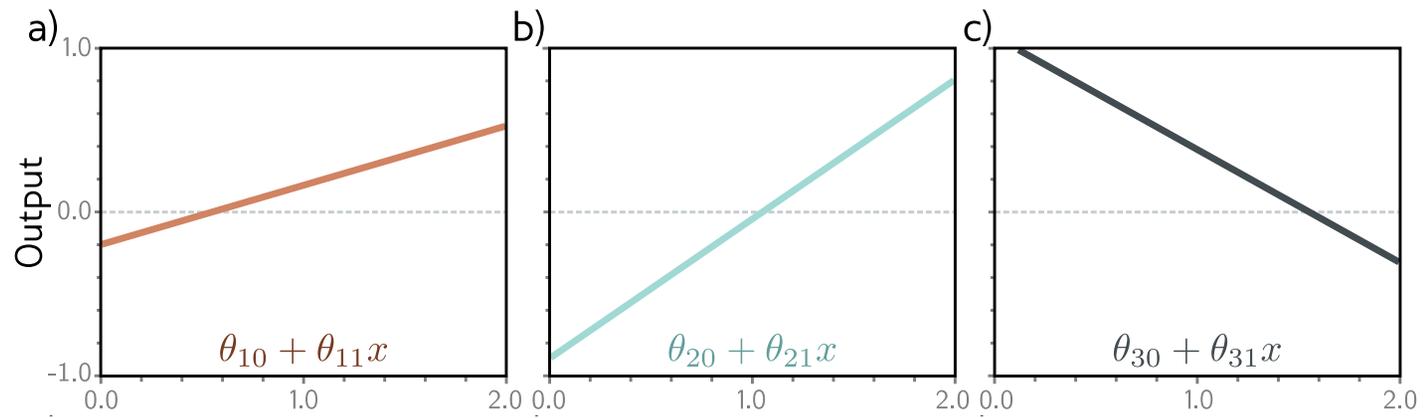
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

where:

$$\text{Hidden units} \left\{ \begin{array}{l} h_1 = a[\theta_{10} + \theta_{11}x] \\ h_2 = a[\theta_{20} + \theta_{21}x] \\ h_3 = a[\theta_{30} + \theta_{31}x] \end{array} \right.$$

1. compute three linear functions

Linear Functions



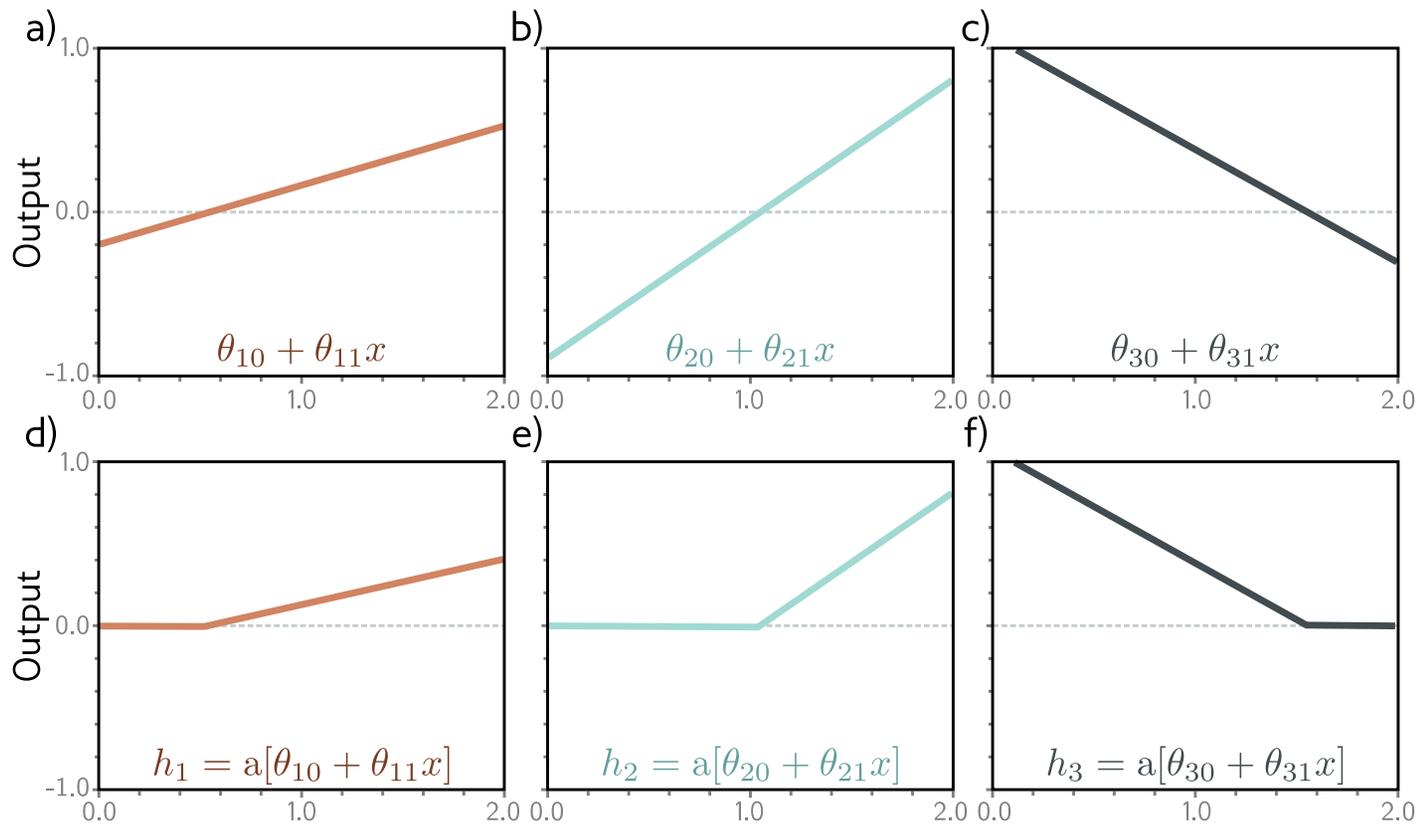
2. Pass through ReLU functions (creates hidden units)

$$h_1 = a[\theta_{10} + \theta_{11}x]$$

$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x],$$

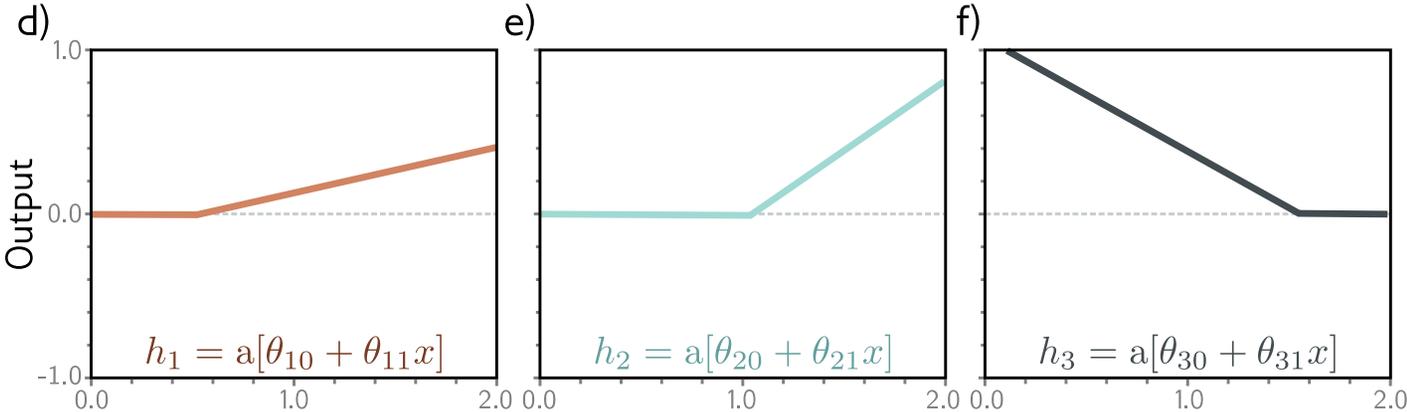
Linear Functions



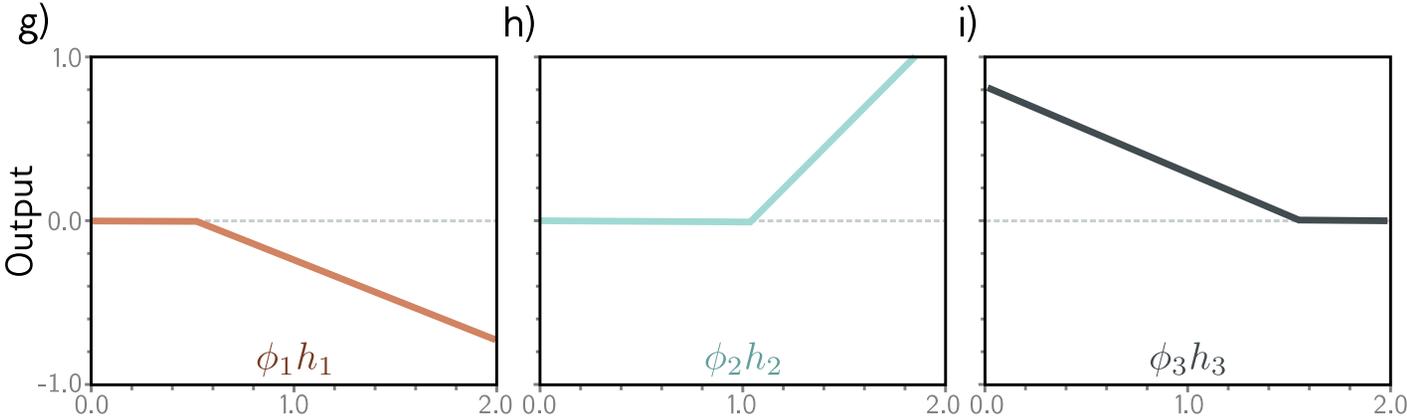
After Activation

## 2. Weight the hidden units

After Activation



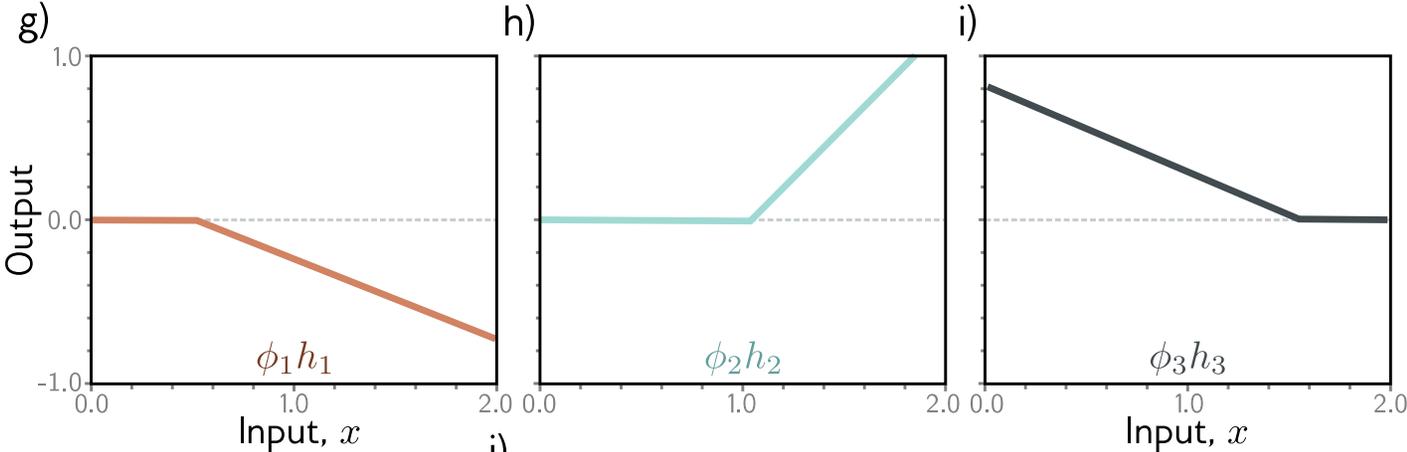
Weight the Hidden units



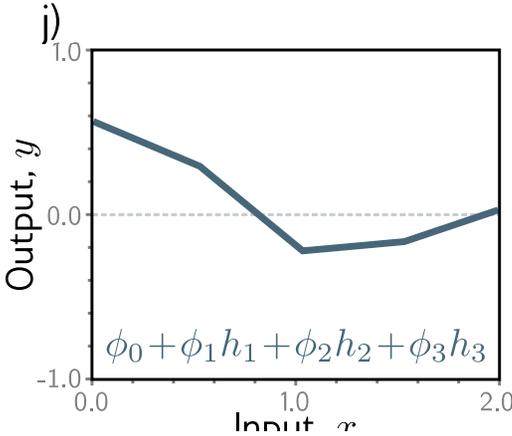
4. Sum the weighted hidden units to create output

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

Weight the hidden units

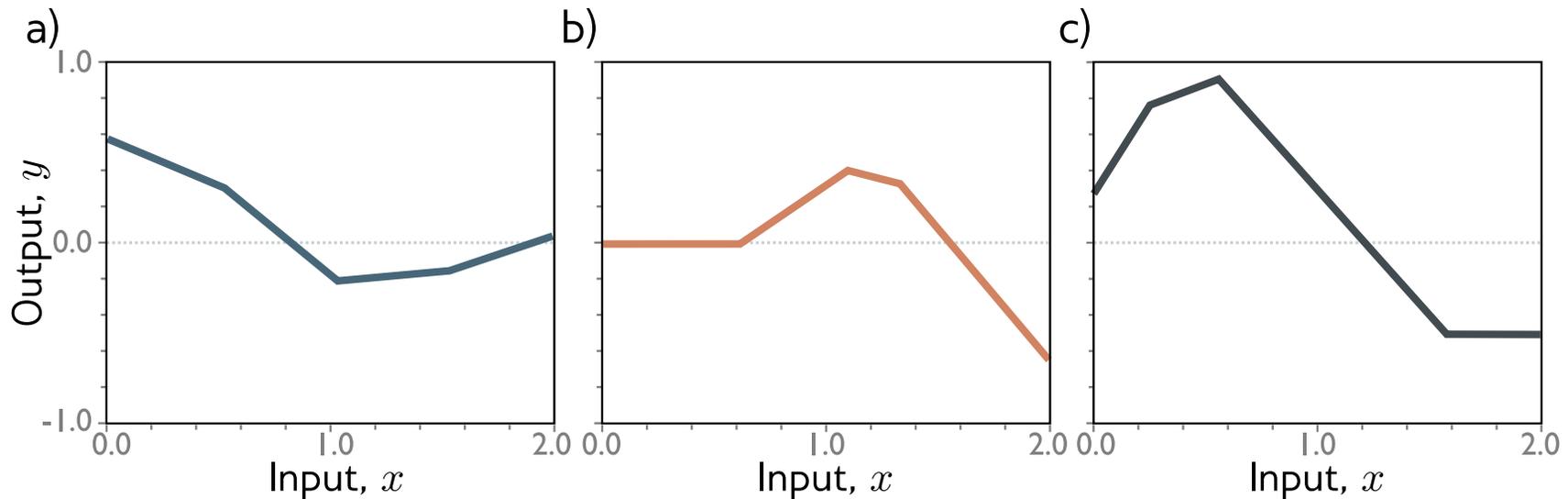


Sum the weighted hidden units



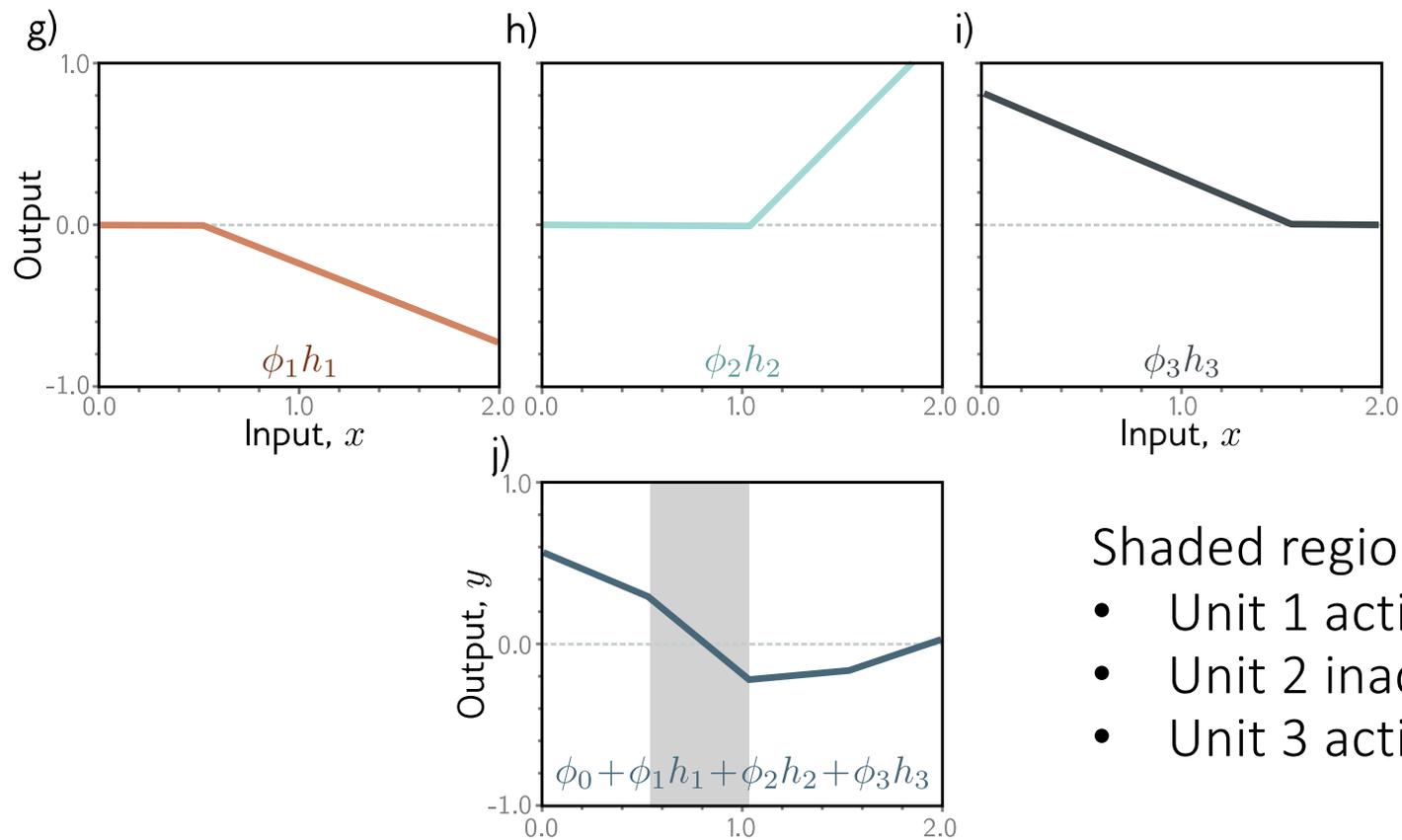
## Example: 3 different shallow networks

$$y = \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x].$$

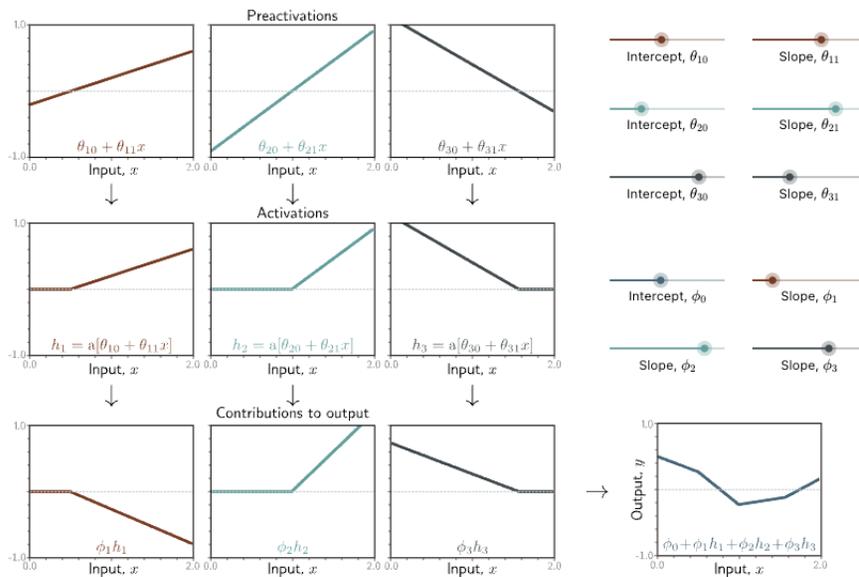


Example shallow network = piecewise linear functions  
1 "joint" per ReLU function

Activation pattern = which hidden units are activated?



# Interactive Figure 3.3a: 1D Shallow Network (ReLU)



<https://udlbook.github.io/udlfigures/>

**Figure 3.3** Computation for function in figure 3.2a. (Top row) The input  $x$  is passed through three linear functions, each with a different  $y$ -intercept  $\theta_{\bullet 0}$  and slope  $\theta_{\bullet 1}$ . (Center row) Each line is passed through the ReLU activation function. (Bottom row) The three resulting functions are then weighted (scaled) by  $\phi_1$ ,  $\phi_2$ , and  $\phi_3$ , respectively. (Bottom right) Finally, the weighted functions are summed, and an offset  $\phi_0$  that controls the height is added.

Move the sliders to modify the parameters of the shallow network.

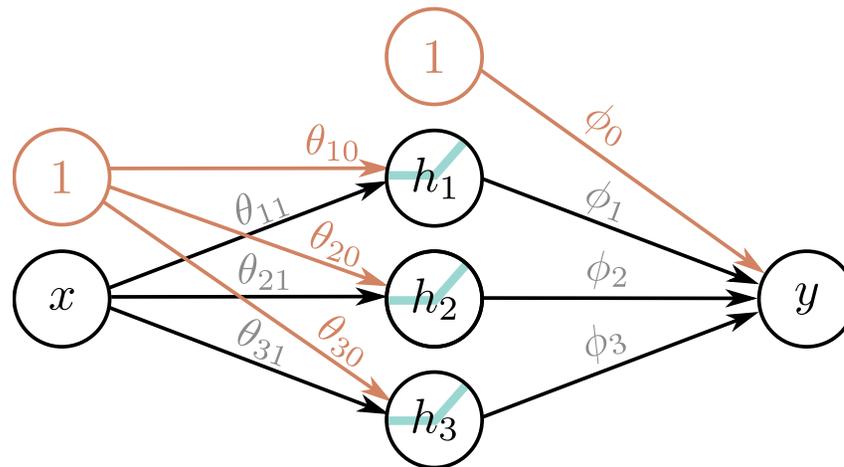
# Depicting neural networks

$$h_1 = a[\theta_{10} + \theta_{11}x]$$

$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x]$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$



Each parameter multiplies its source and adds to its target

# Depicting neural networks

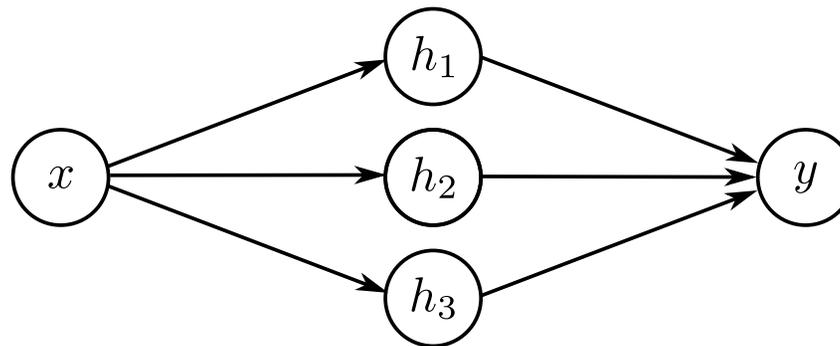
*Usually don't show the bias terms*

$$h_1 = a[\theta_{10} + \theta_{11}x]$$

$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x]$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$



# Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology

With 3 hidden units:

$$h_1 = a[\theta_{10} + \theta_{11}x]$$

$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x]$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

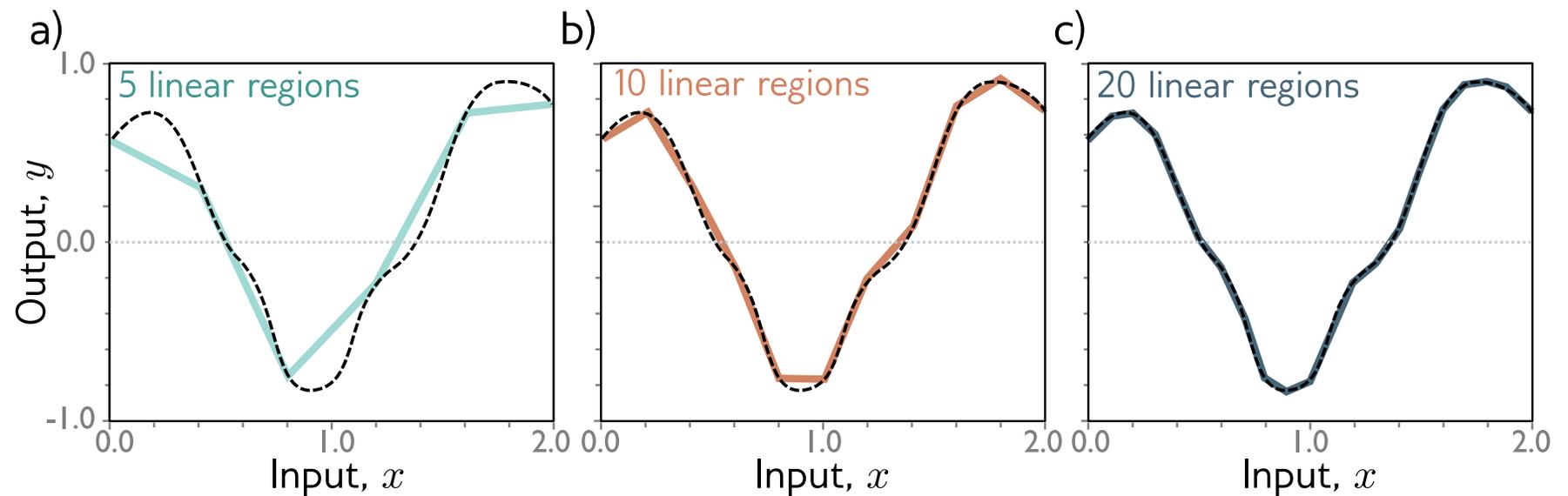
With D hidden units:

$$h_d = a[\theta_{d0} + \theta_{d1}x]$$

$$y = \phi_0 + \sum_{d=1}^D \phi_d h_d$$

# With enough hidden units...

... we can describe any 1D function to arbitrary accuracy



# Universal approximation theorem

“a formal proof that, with enough hidden units, a shallow neural network can describe any continuous function in  $\mathbb{R}^D$  to arbitrary precision”

# Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology

# Two outputs

- 1 input, 4 hidden units, 2 outputs

$$h_1 = a[\theta_{10} + \theta_{11}x]$$

$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x]$$

$$h_4 = a[\theta_{40} + \theta_{41}x]$$

$$y_1 = \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4$$

$$y_2 = \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4$$

# Two outputs

- 1 input, 4 hidden units, 2 outputs

$$h_1 = a[\theta_{10} + \theta_{11}x]$$

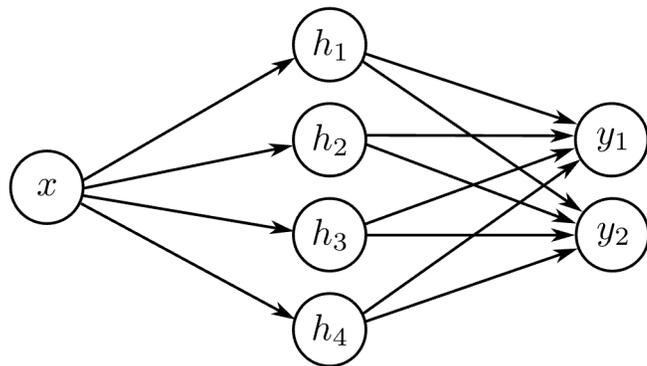
$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x]$$

$$h_4 = a[\theta_{40} + \theta_{41}x]$$

$$y_1 = \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4$$

$$y_2 = \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4$$



# Two outputs

- 1 input, 4 hidden units, 2 outputs

$$h_1 = a[\theta_{10} + \theta_{11}x]$$

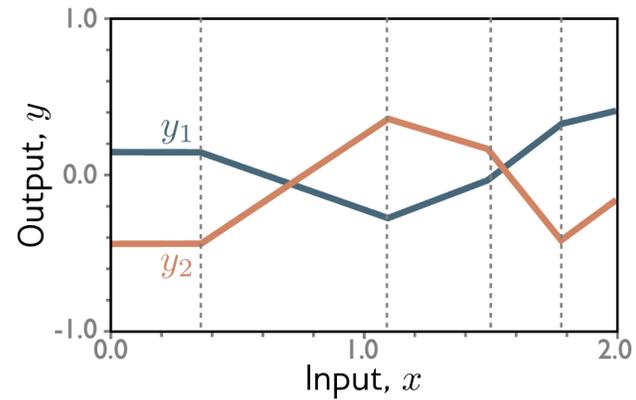
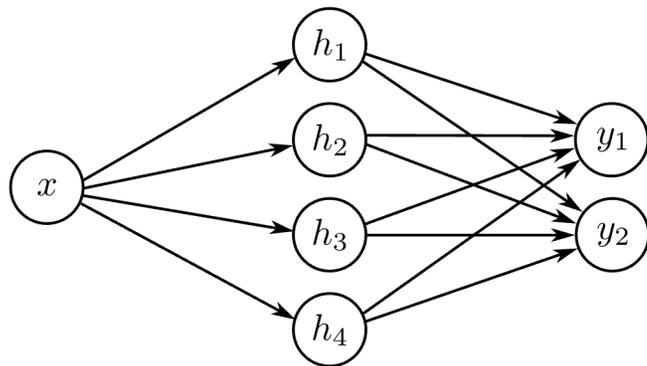
$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x]$$

$$h_4 = a[\theta_{40} + \theta_{41}x]$$

$$y_1 = \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4$$

$$y_2 = \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4$$



# Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology

# Two inputs

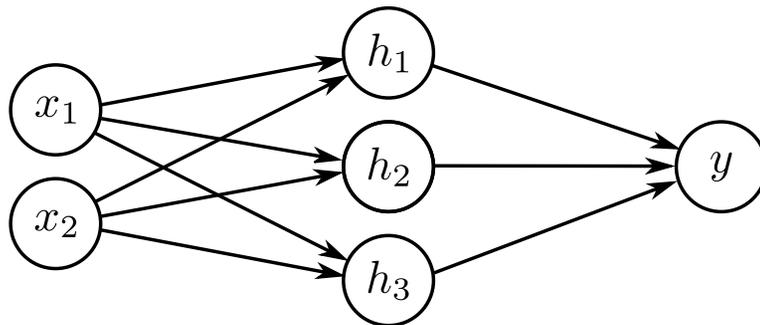
- 2 inputs, 3 hidden units, 1 output

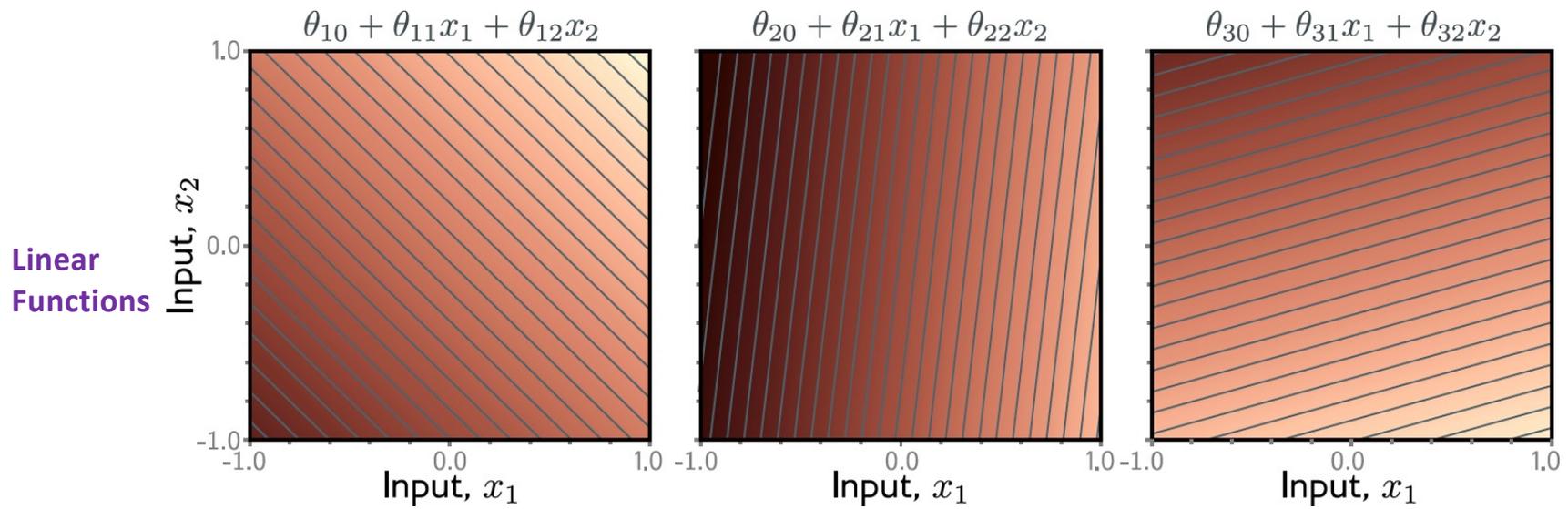
$$h_1 = a[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$$

$$h_2 = a[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$$

$$h_3 = a[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$$

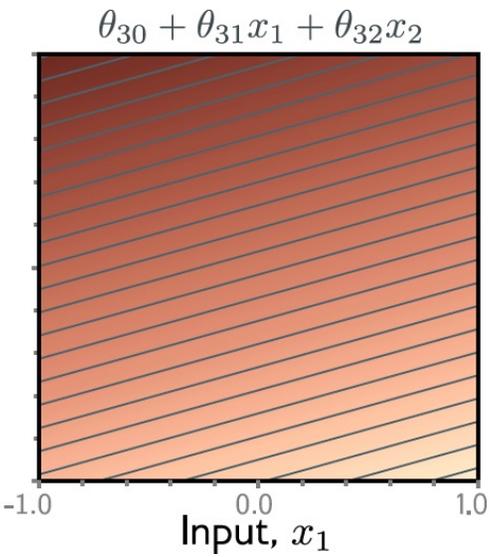
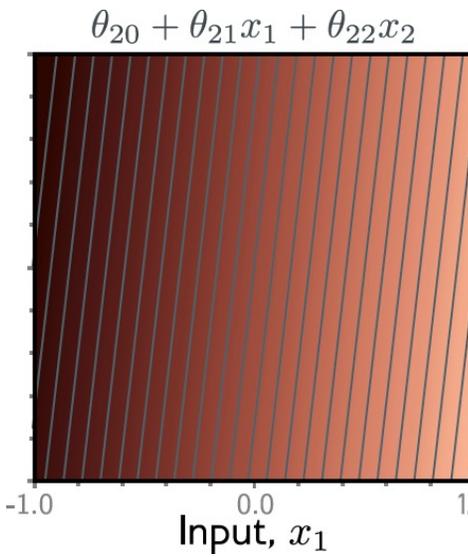
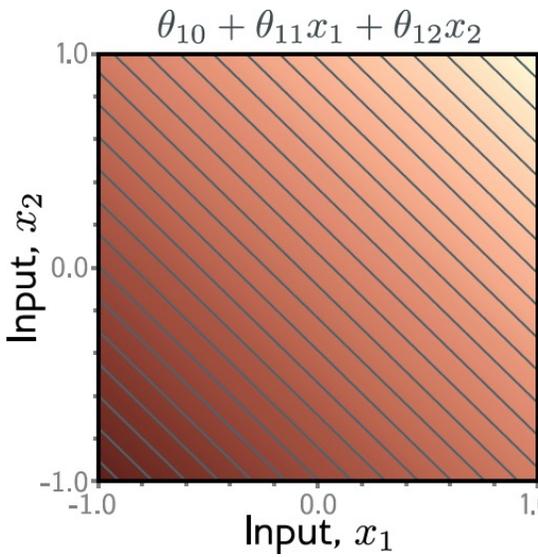
$$y = \phi_0 + \phi_1h_1 + \phi_2h_2 + \phi_3h_3$$



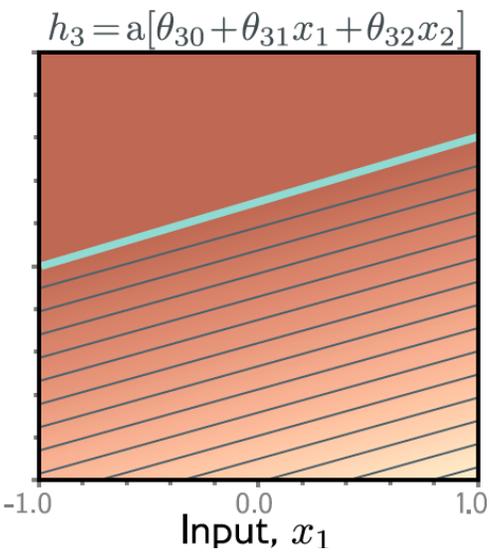
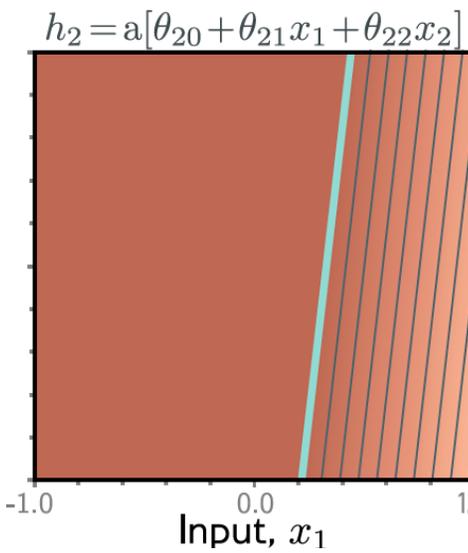
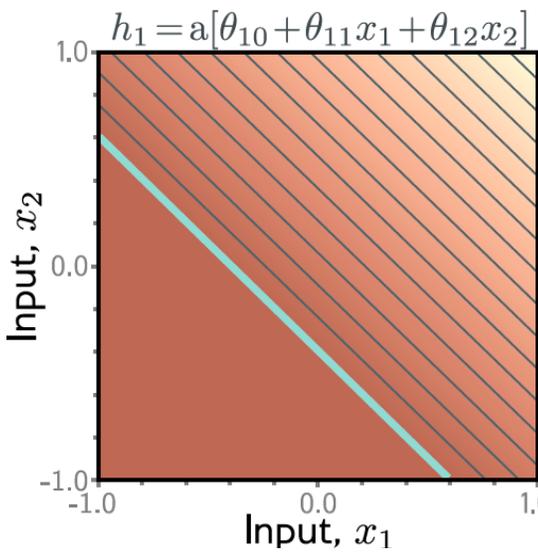


[See Interactive Figure 3.8a https://udlbook.github.io/udlfigures/](https://udlbook.github.io/udlfigures/)

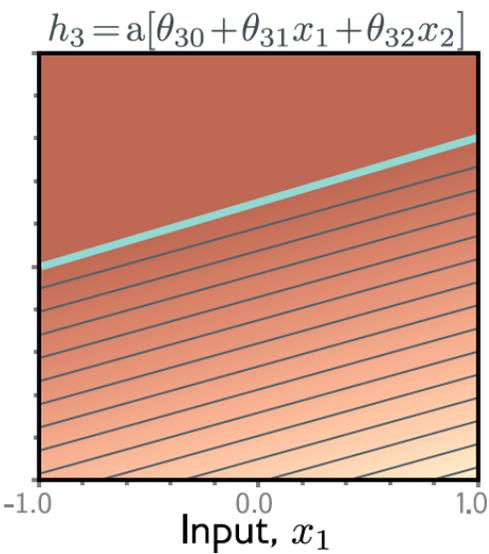
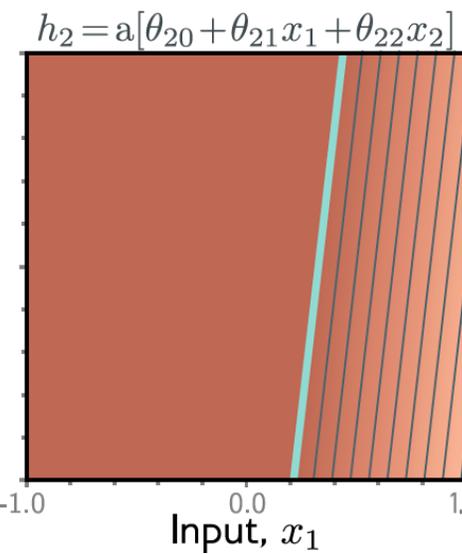
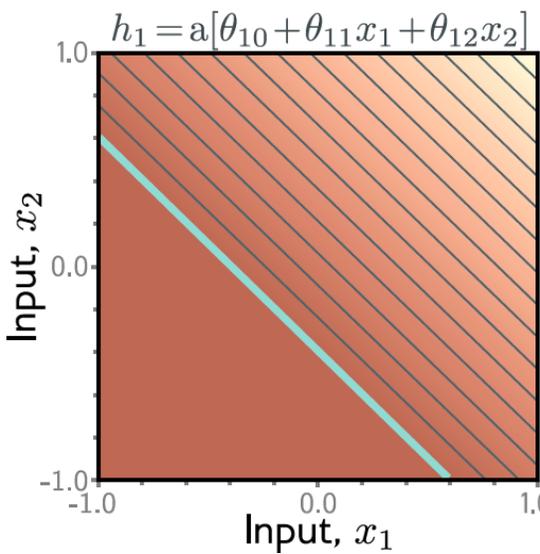
Linear Functions



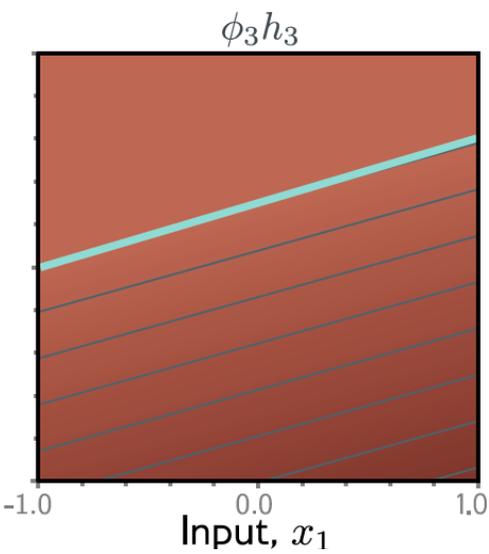
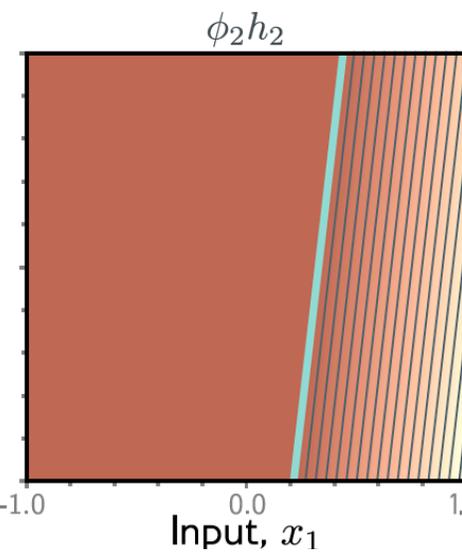
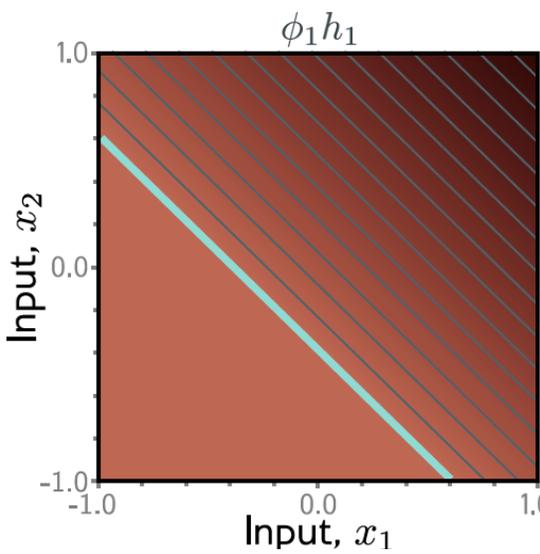
After Activation



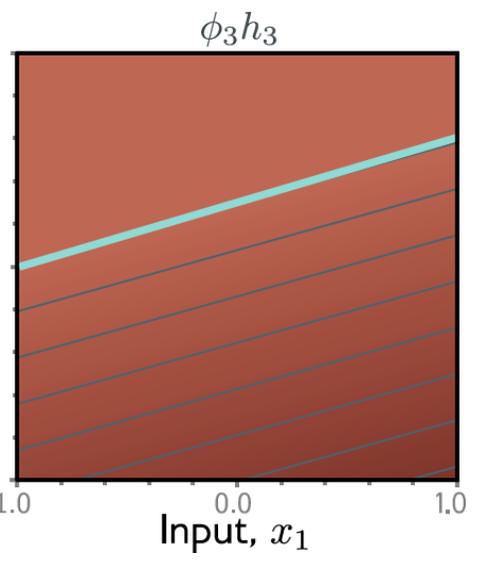
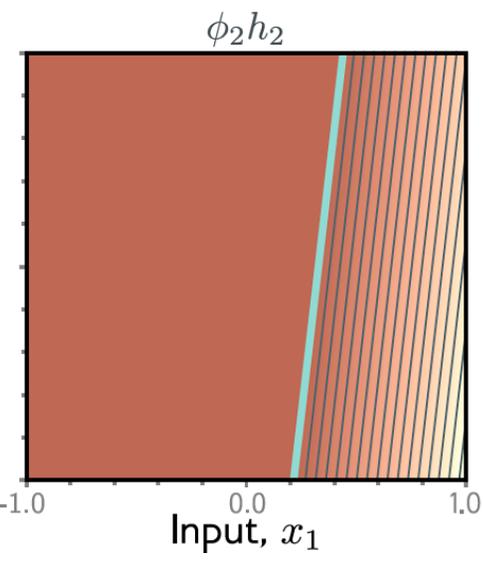
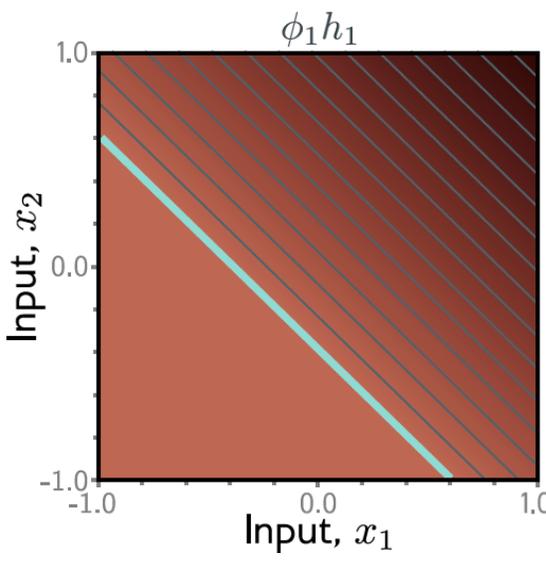
After  
Activation



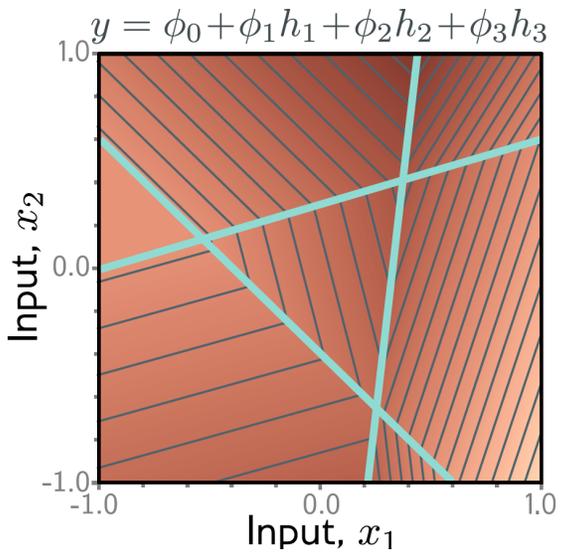
Weight the  
Hidden units

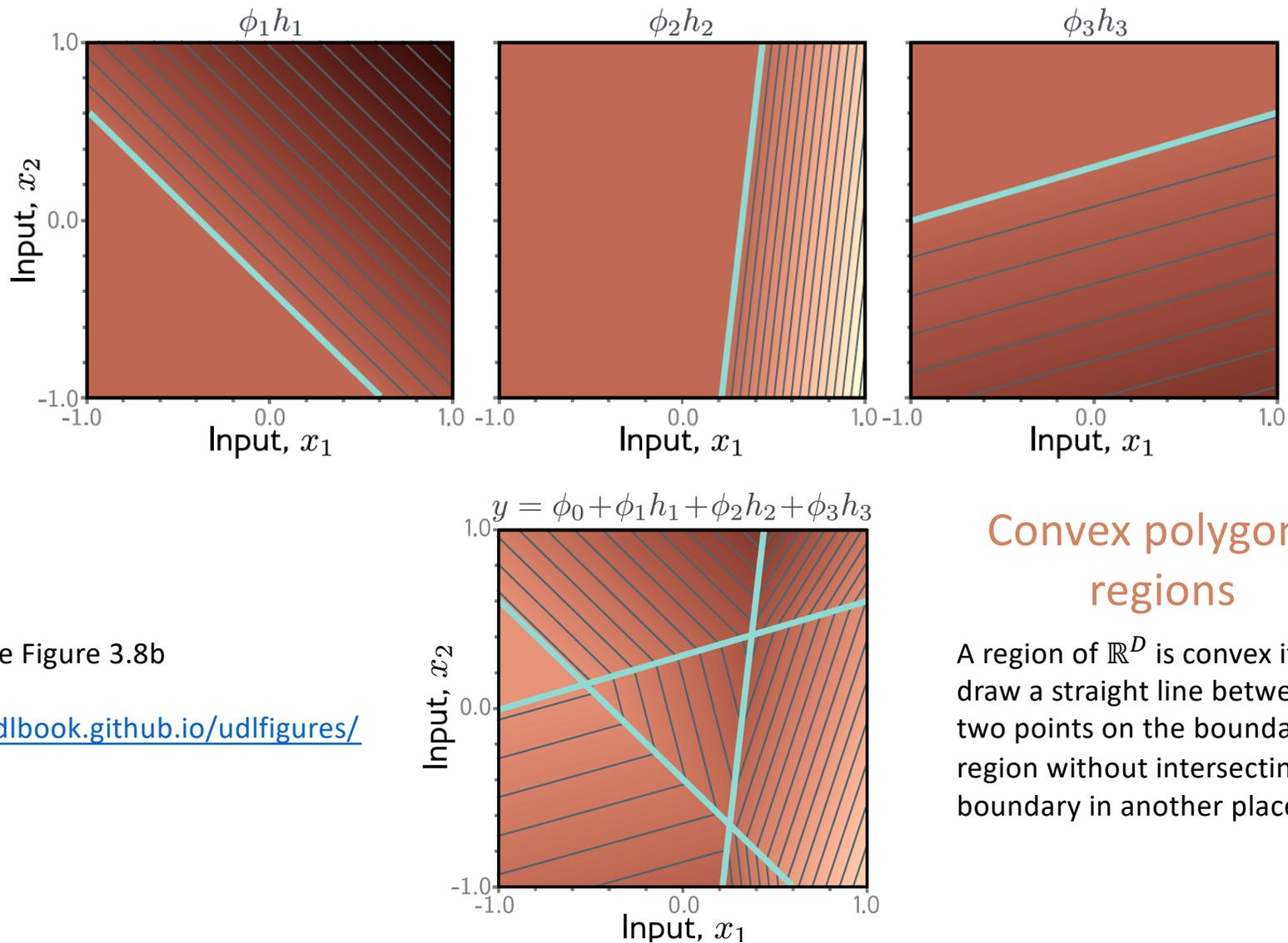


Weight the hidden units



Sum the weighted hidden units





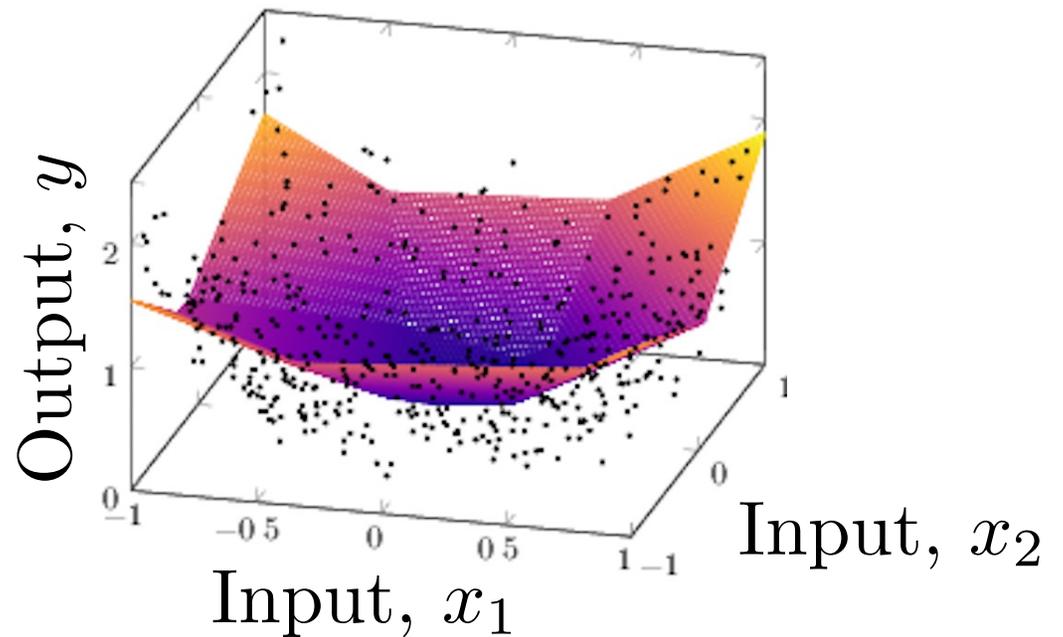
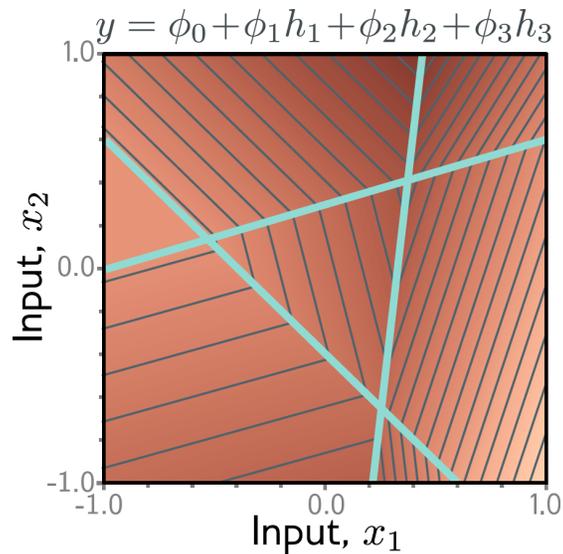
Interactive Figure 3.8b

<https://udlbook.github.io/udlfigures/>

## Convex polygonal regions

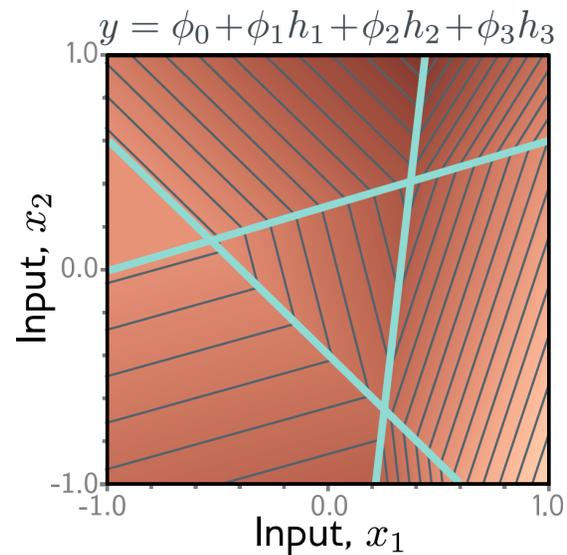
A region of  $\mathbb{R}^D$  is convex if we can draw a straight line between any two points on the boundary of the region without intersecting the boundary in another place.

Fitting a dataset where:  
each sample has 2 inputs and 1 output



## Question:

- For the 2D case, what if there were two outputs?
- If this is one of the outputs, what would the other one look like?



(slide poll on next slide)

**slido**

Please download and install the Slido app on all computers you use



**For the 2D case with 2 outputs, what would the other output look like?**

① Start presenting to display the poll results on this slide.

# Shallow neural networks

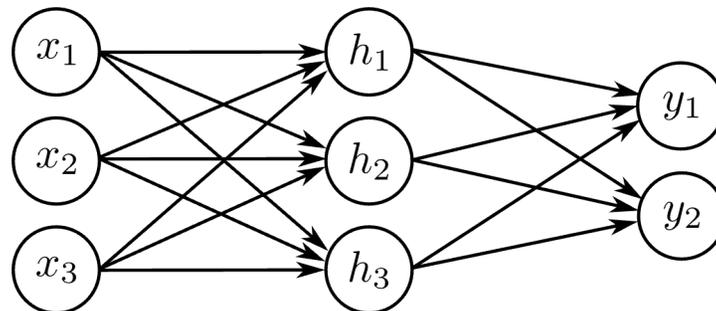
- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- **General case**
- Number of regions
- Terminology

# Arbitrary inputs, hidden units, outputs

- $D_i$  inputs,  $D$  hidden units, and  $D_o$  Outputs

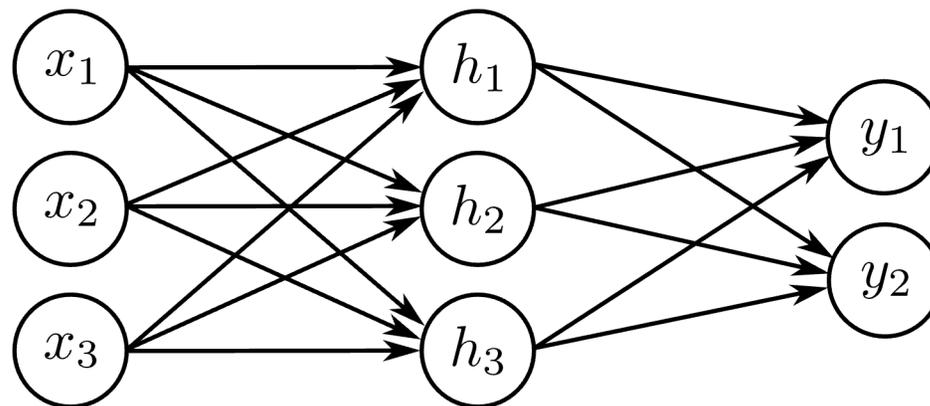
$$h_d = a \left[ \theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i \right] \quad y_j = \phi_{j0} + \sum_{d=1}^D \phi_{jd} h_d$$

- e.g., Three inputs, three hidden units, two outputs



Question:

- How many parameters does this model have?



Poll on next slide

**Do not edit**  
*How to change the design*

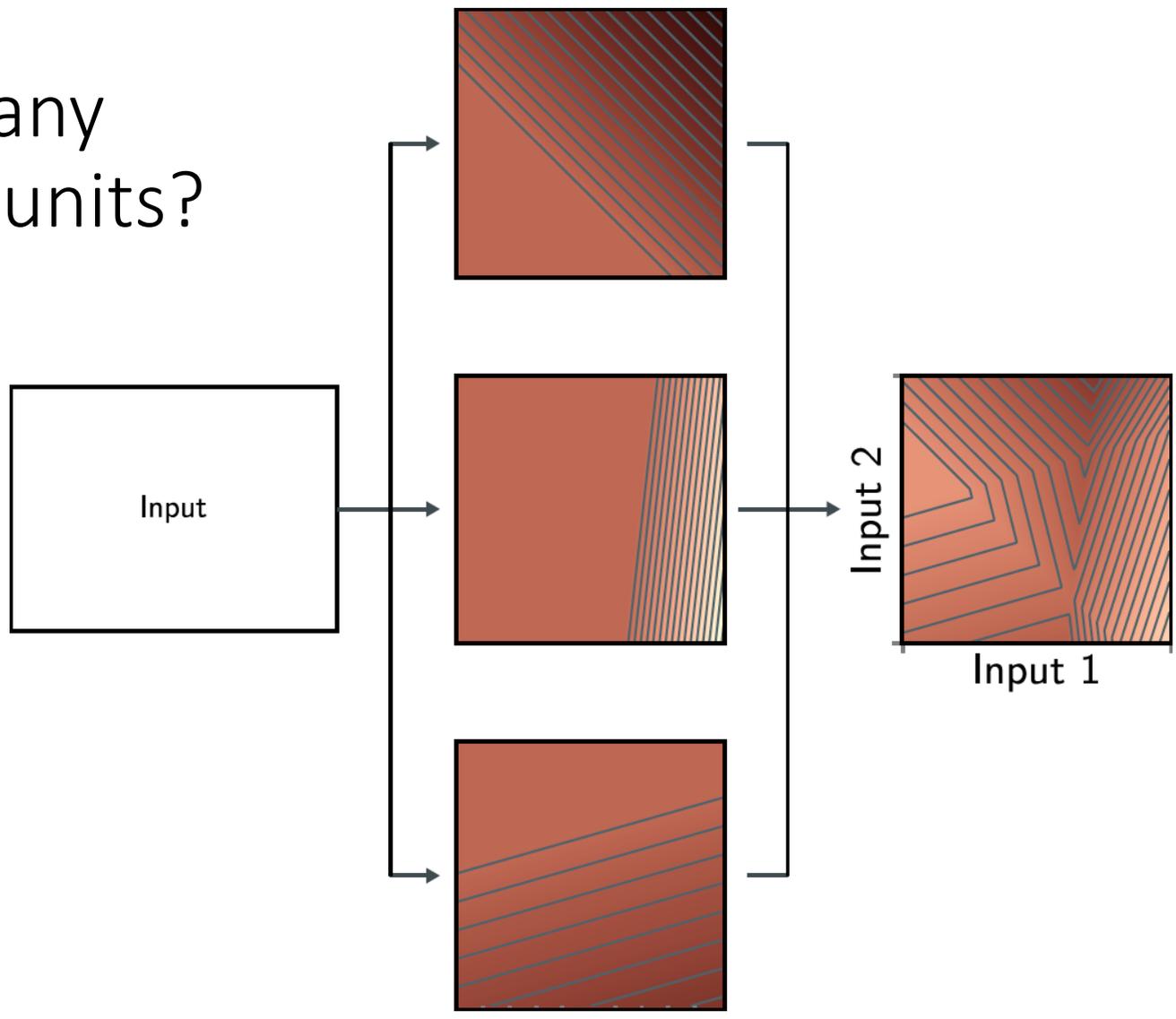


## How many parameters in the model?

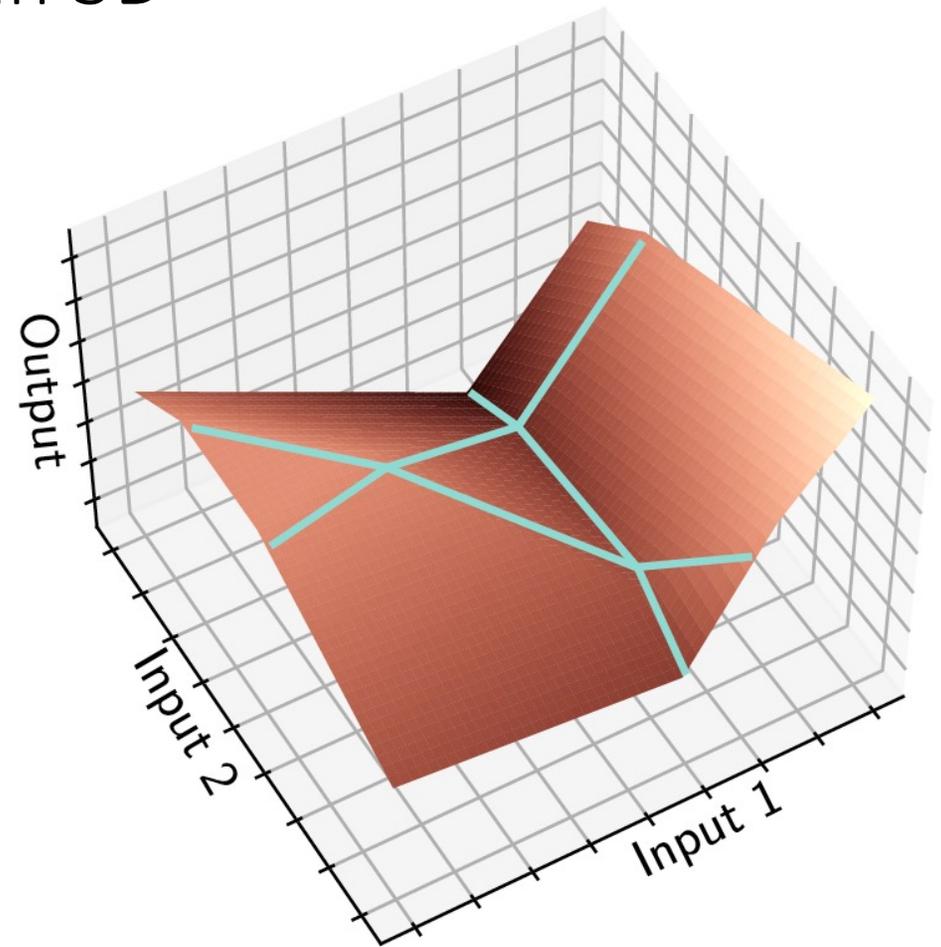
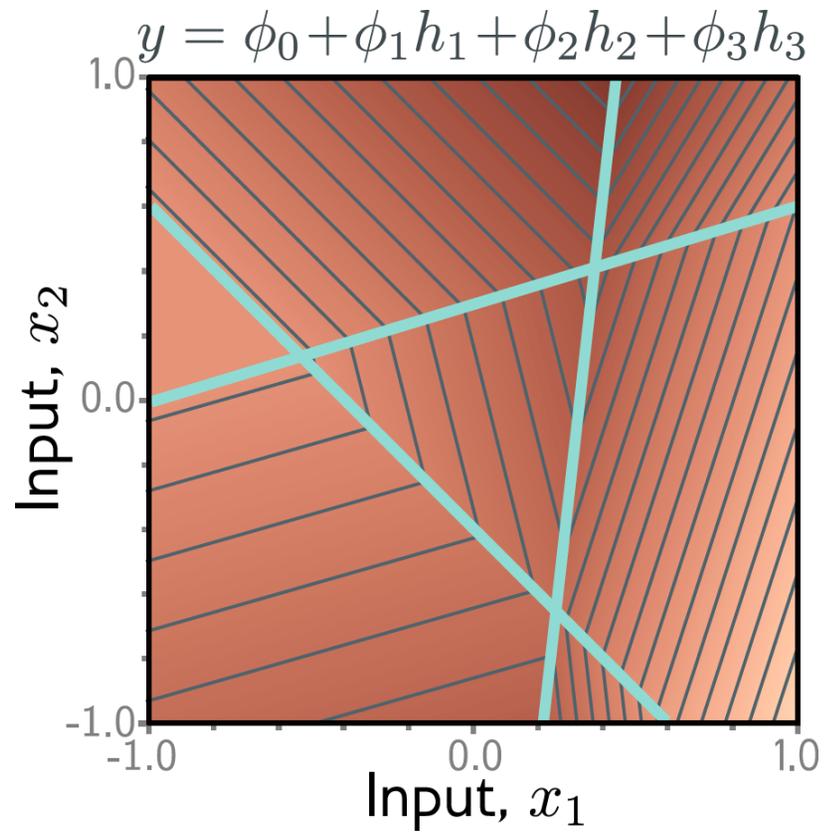
① The Slido app must be installed on every computer you're presenting from

**slido**

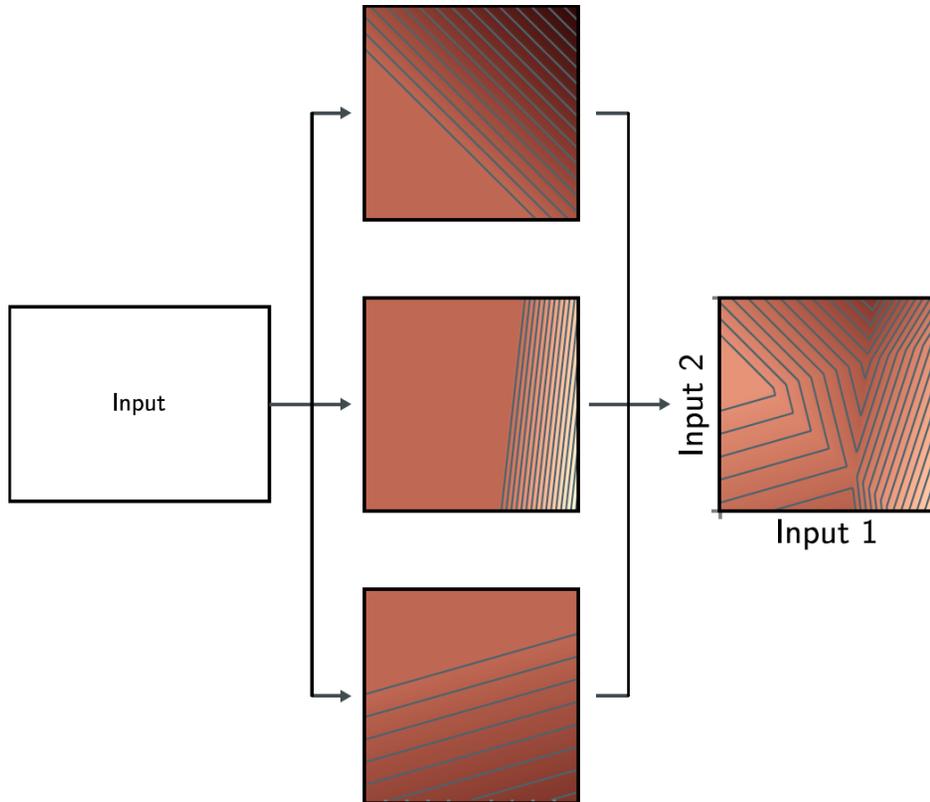
How many hidden units?



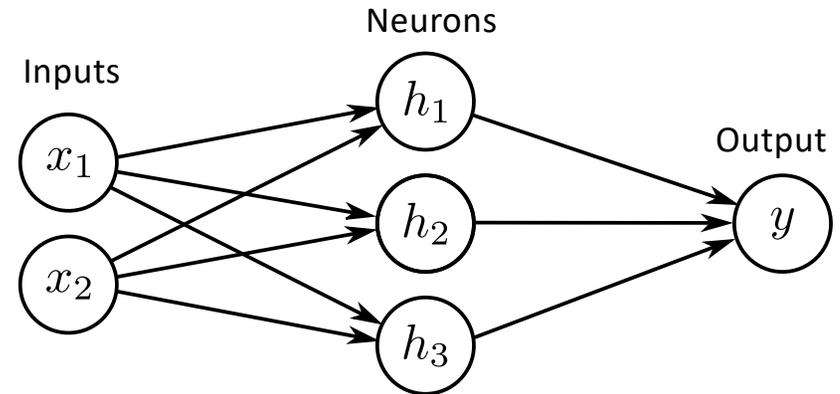
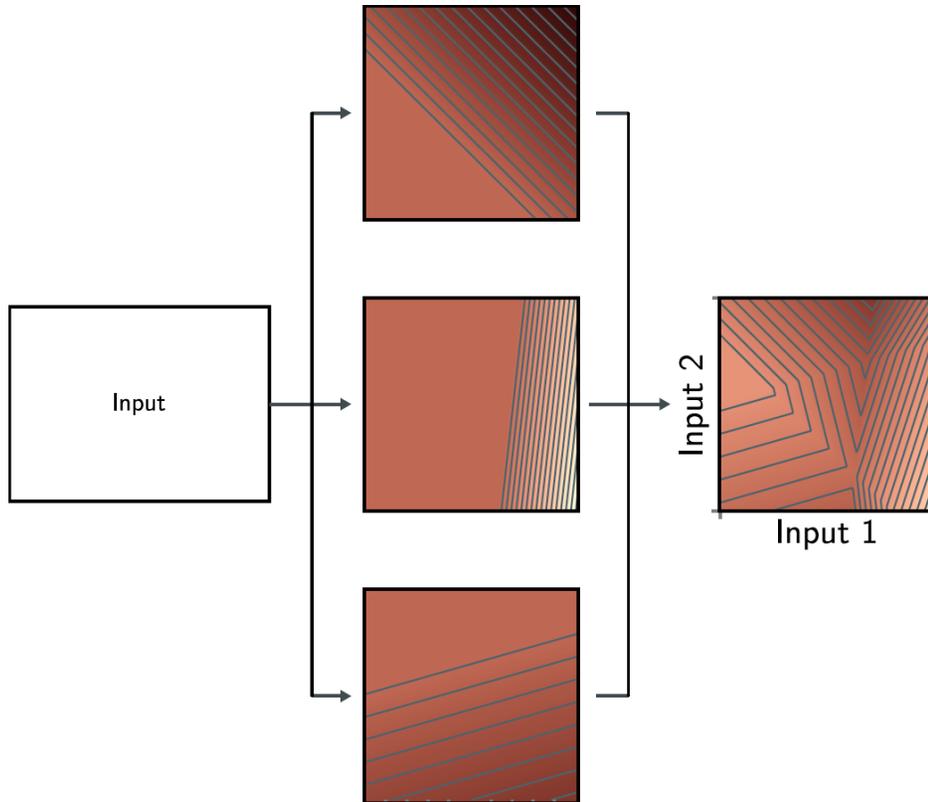
# Output with boundaries and in 3D



How would you draw and write this neural network?



How would you draw and write this neural network?



“neural network”

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

$$h_1 = a[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$$

$$h_2 = a[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$$

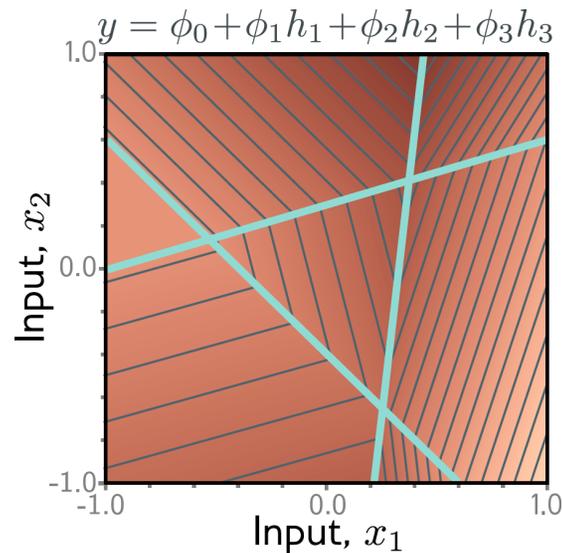
$$h_3 = a[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$$

# Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology

# Number of output regions

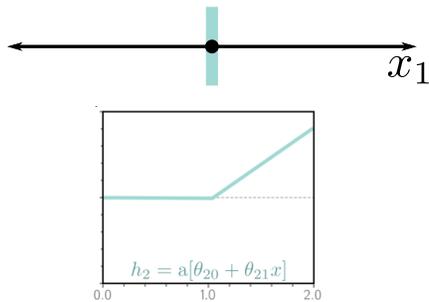
- With ReLU activations, each output consists of multi-dimensional **piecewise linear hyperplanes**
- With two inputs, and three hidden units, we saw there were seven polygons for each output:



# Example with $D = D_i \rightarrow 2^{D_i}$ regions

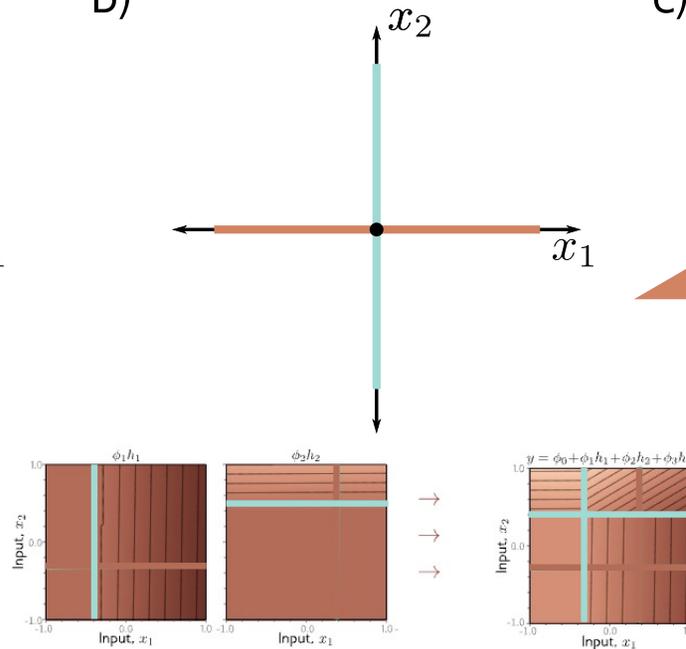
$D_i$  : # of inputs  
 $D$  : # of hidden units  
 $D_o$  : # of outputs

a)



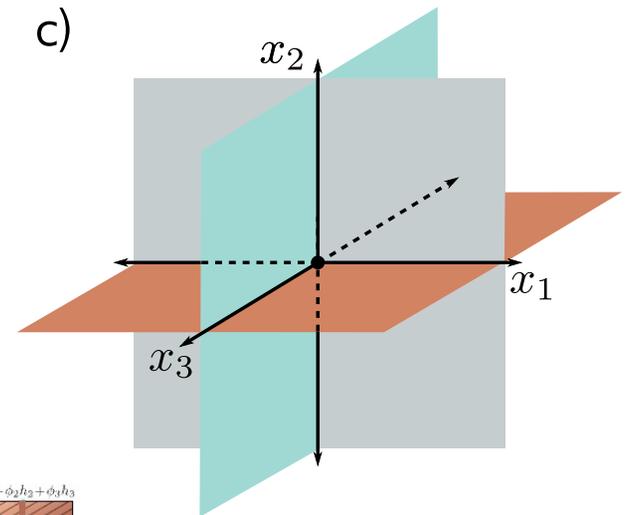
- 1 input (1-dimension)
- 1 hidden unit
- creates two regions (one joint)

b)



- 2 input (2-dimensions) with
- 2 hidden units
- creates four regions (two lines)

c)



- 3 inputs (3-dimensions) with
- 3 hidden units
- creates eight regions (three planes)

$D_i$  : # of inputs  
 $D$  : # of hidden units  
 $D_o$  : # of outputs

## Number of regions:

- Number of regions created by  $D > D_i$  hyper-planes in  $D_i$  dimensions was proved by Zaslavsky (1975) to be:

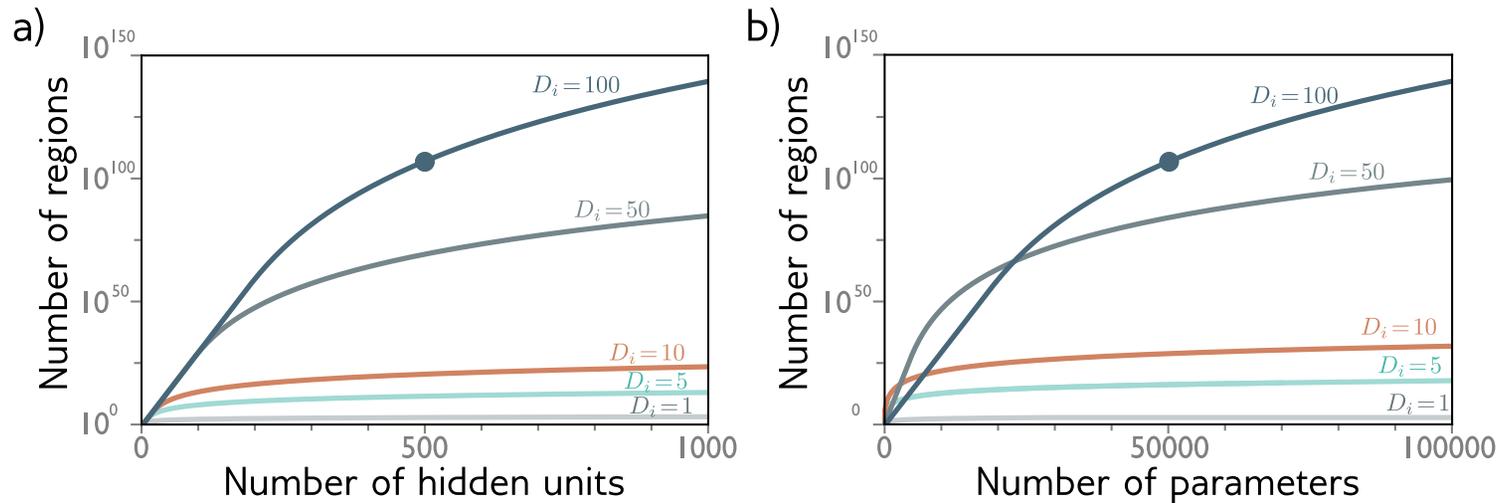
$$\sum_{j=0}^{D_i} \binom{D}{j} = \frac{D!}{j!(D-j)!} \quad \leftarrow \text{Binomial coefficients!}$$

- How big is this? It's greater than  $2^{D_i}$  but less than  $2^D$ .

# Number of output regions

$D_i$  : # of inputs  
 $D$  : # of hidden units  
 $D_o$  : # of outputs

- In general, each output consists of  $D$  dimensional **convex polytopes**
- How many?

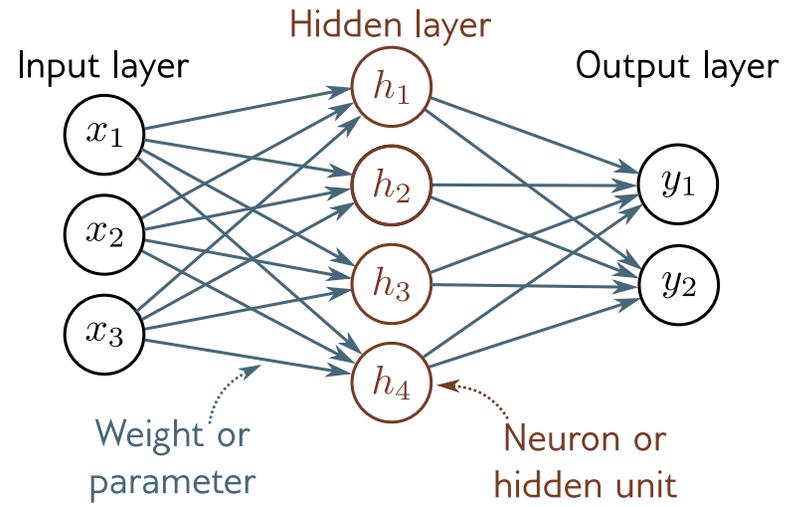


Highlighted point = 500 hidden units or 51,001 parameters

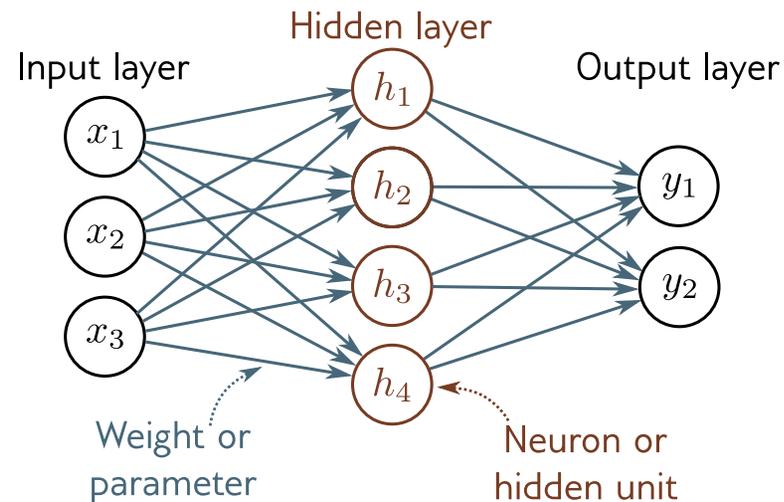
# Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology

# Nomenclature

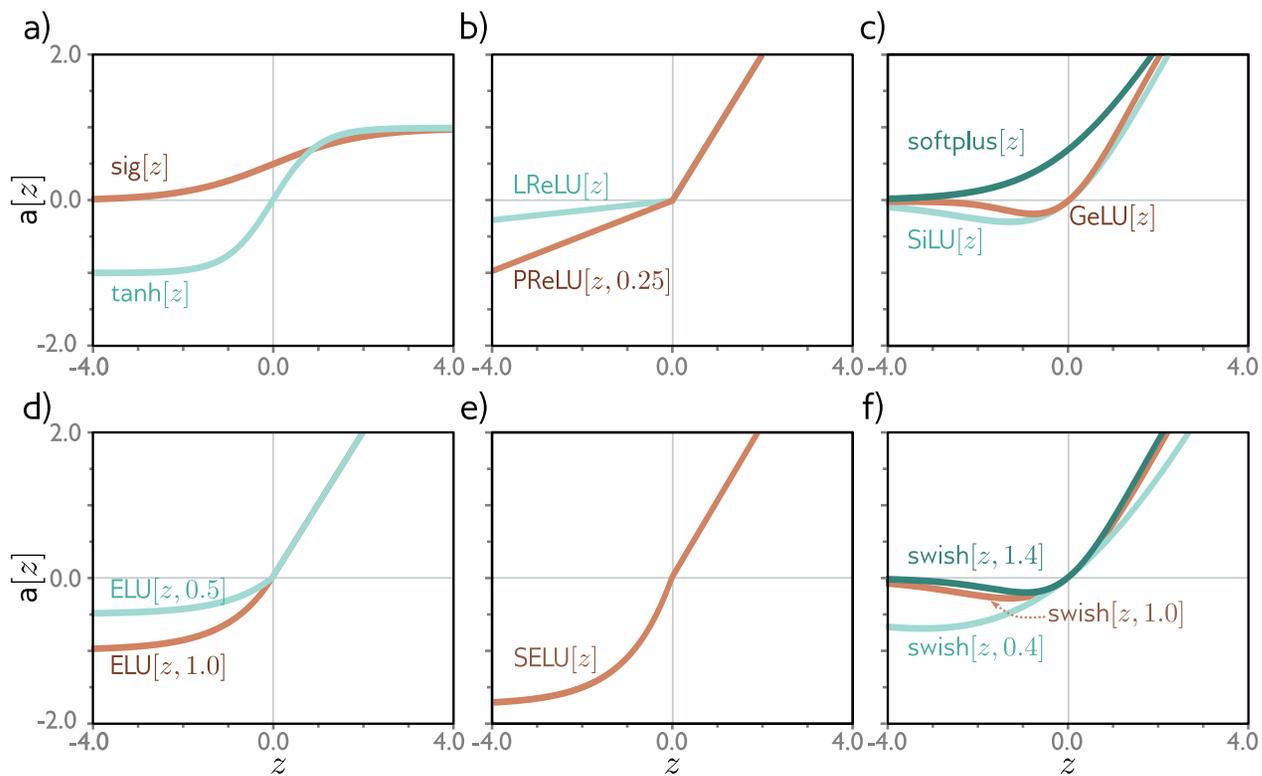


# Nomenclature



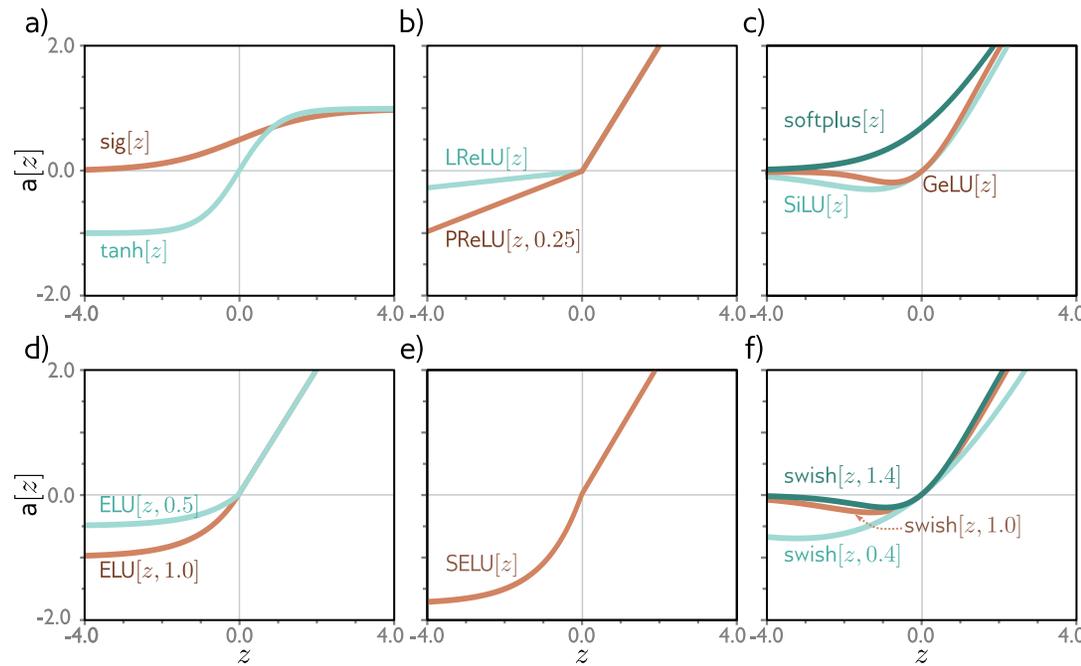
- Y-offsets = **biases**
- Slopes = **weights**
- Everything in one layer connected to everything in the next = **fully connected network (multi-layer perceptron)**
- No loops = **feedforward network**
- Values after ReLU (activation functions) = **activations**
- Values before ReLU = **pre-activations**
- One hidden layer = **shallow neural network**
- More than one hidden layer = **deep neural network**
- Number of hidden units  $\approx$  **capacity**

# Other activation functions



Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. [arXiv:1710.05941](https://arxiv.org/abs/1710.05941).

# Interactive Figures 3.3b and 3.3c



Also look at

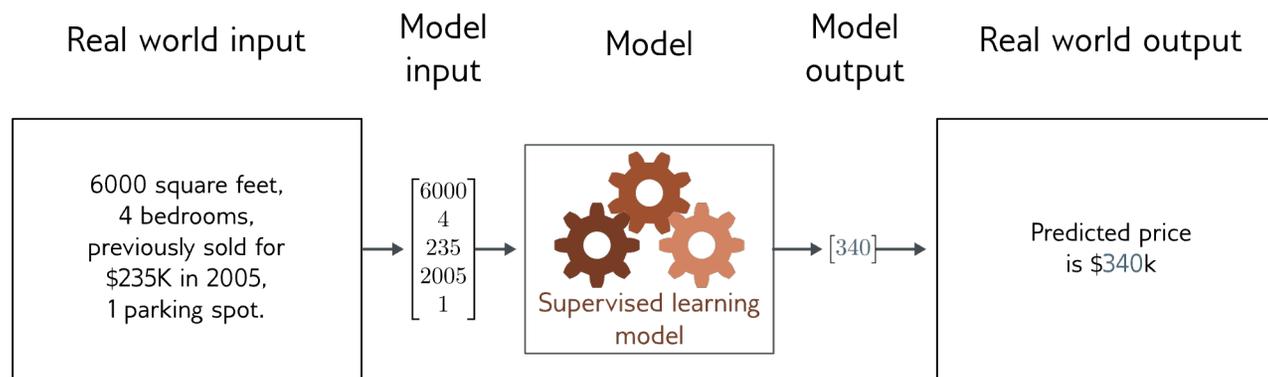
3.3b – 1D shallow network (sigmoid)

3.3c – 1D shallow network (Heaviside/Step)

$$\text{heaviside}[z] = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$$

<https://udlbook.github.io/udlfigures/>

# Regression



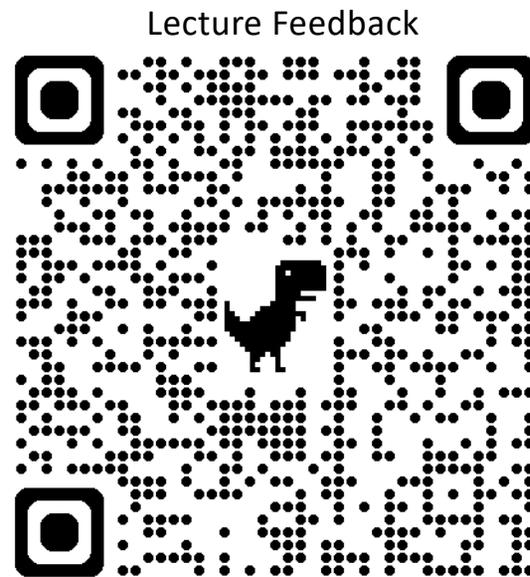
We have built a model that can:

- take an arbitrary number of inputs
- output an arbitrary number of outputs
- model a function of arbitrary complexity between the two

$$h_d = a \left[ \theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i \right] \quad y_j = \phi_{j0} + \sum_{d=1}^D \phi_{jd} h_d$$

## Next time:

- What happens if we feed one neural network into another neural network?



<https://forms.gle/pXHM5nx1Ti9aFmpw6>