Neural networks

Florin Gogianu
Bitdefender ML team

* using slides, illustrations and ideas from prof. Simon Prince UDLB, Stanford CS231n, LeCun, Canziani, NYU Deep Learning

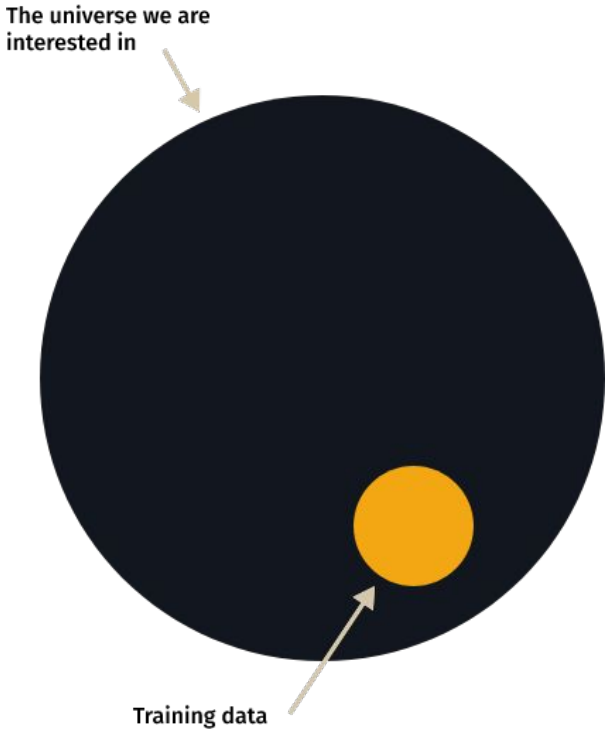
Reading

- Lecture 1: chapters 1, 2, 8.1-3, 9.1 from Prince, *Understanding Deep Learning*
- Lecture 2:
  - chapters 3 and 4 from Prince, UDL book
  - LeCun, Bengio, Hinton, *Deep Learning* (paper)

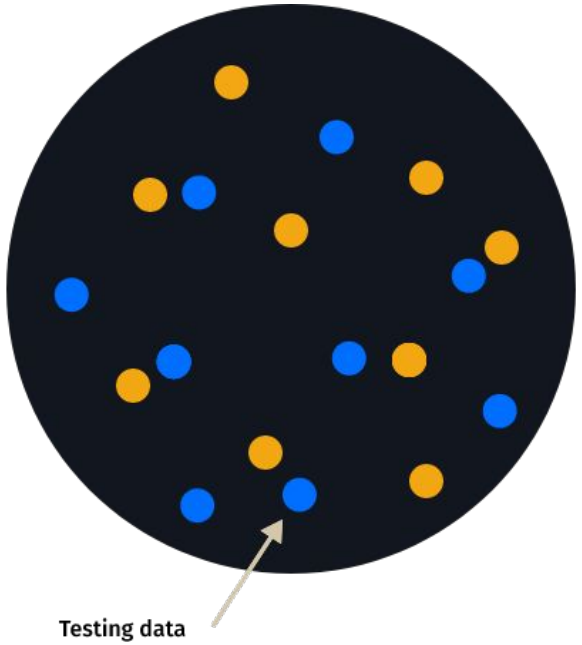
hide & seek: emergent behaviour from sparse rewards

Recap

Another picture of train / test



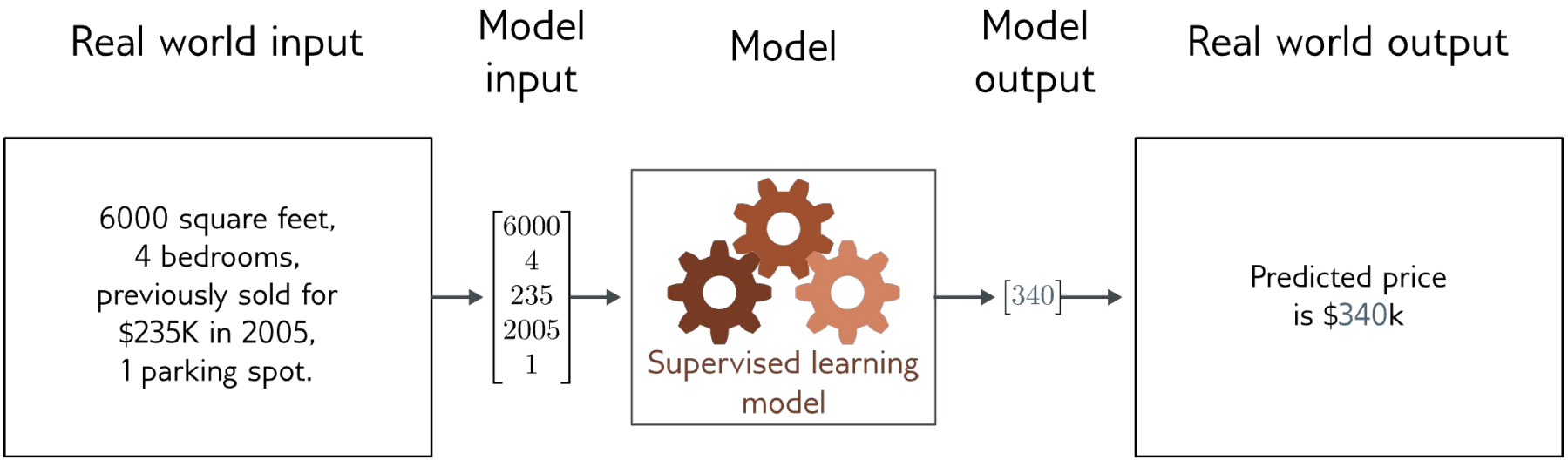
What we want



What we get

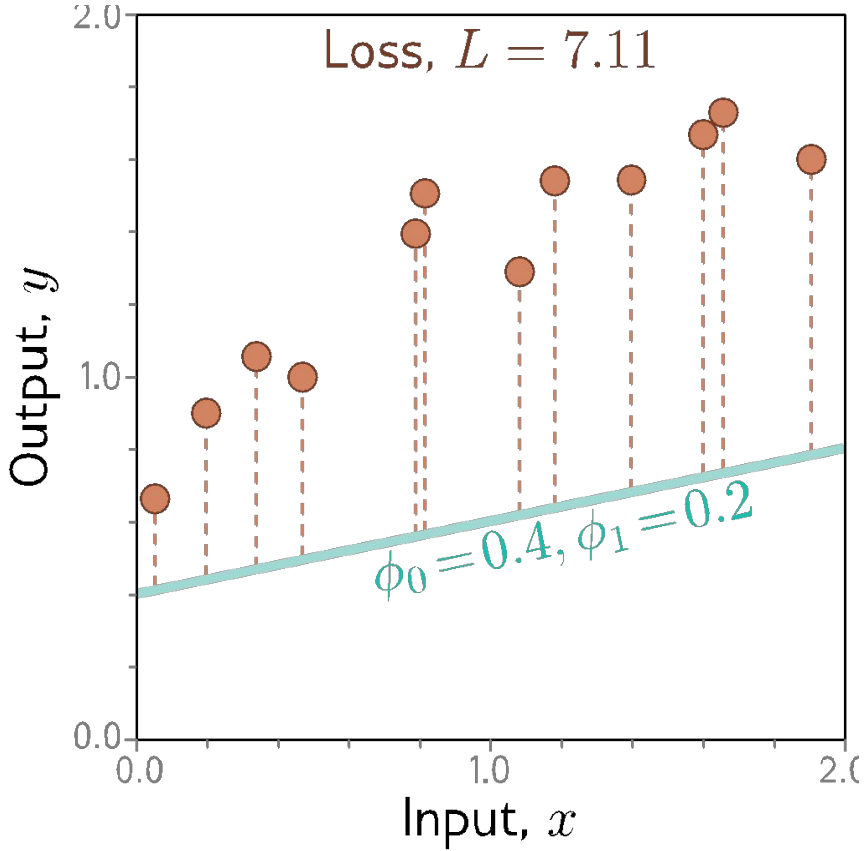

A GOOD EXAMPLE OF
OVERFITTING

Regression



Univariate regression problem (one output, real value)

1D *linear* regression



Loss function:

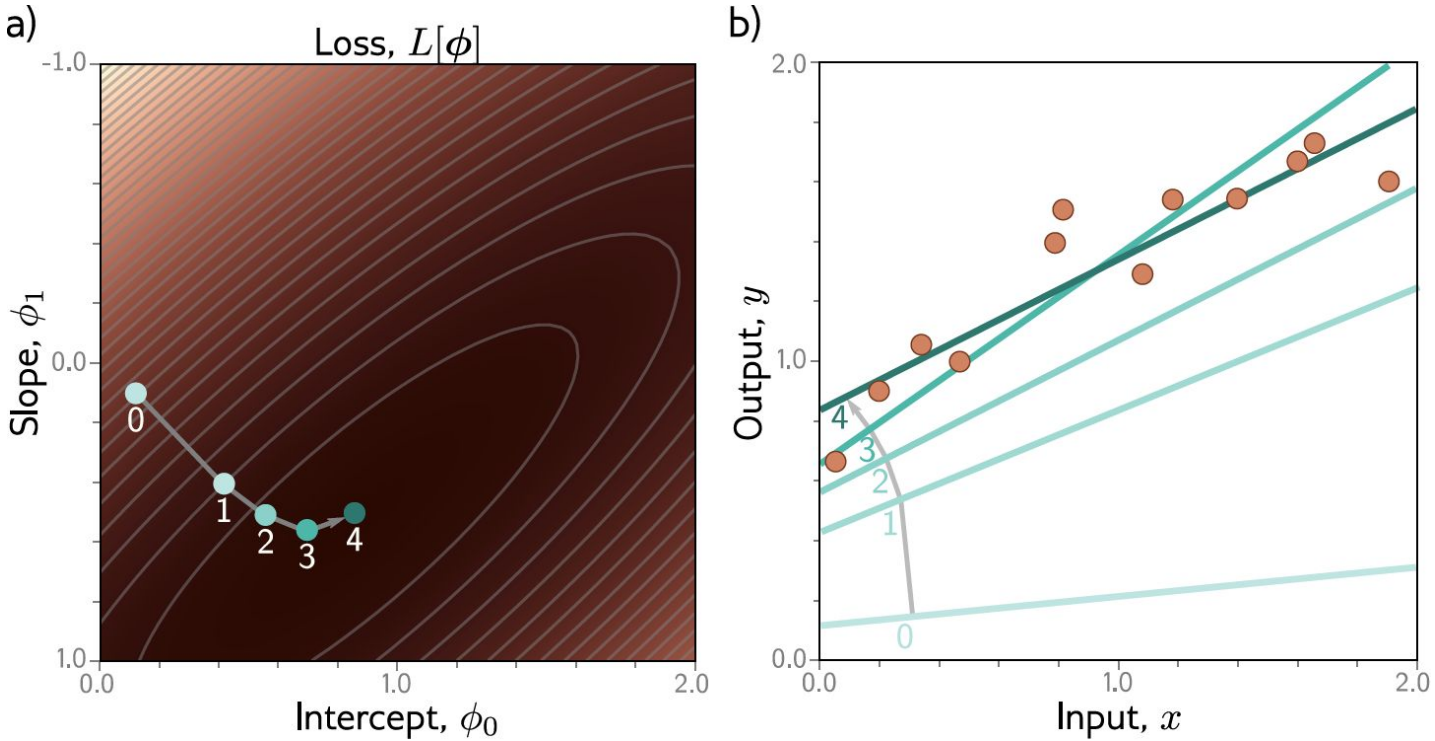
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

? ?

B

1D *linear* regression



Gradient descent

Shallow neural networks

Why 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

- 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

Consider a *shallow* neural network

Let's go from a two-parameter linear model:

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

To something just a bit more complicated:

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

Shallow neural network

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

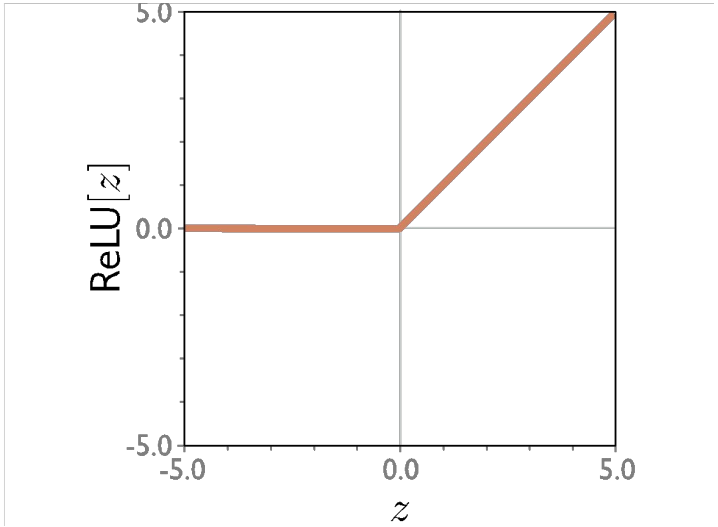
Shallow neural network

activation function

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

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

Rectified Linear Unit
(particular kind of activation function)



Shallow neural network

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

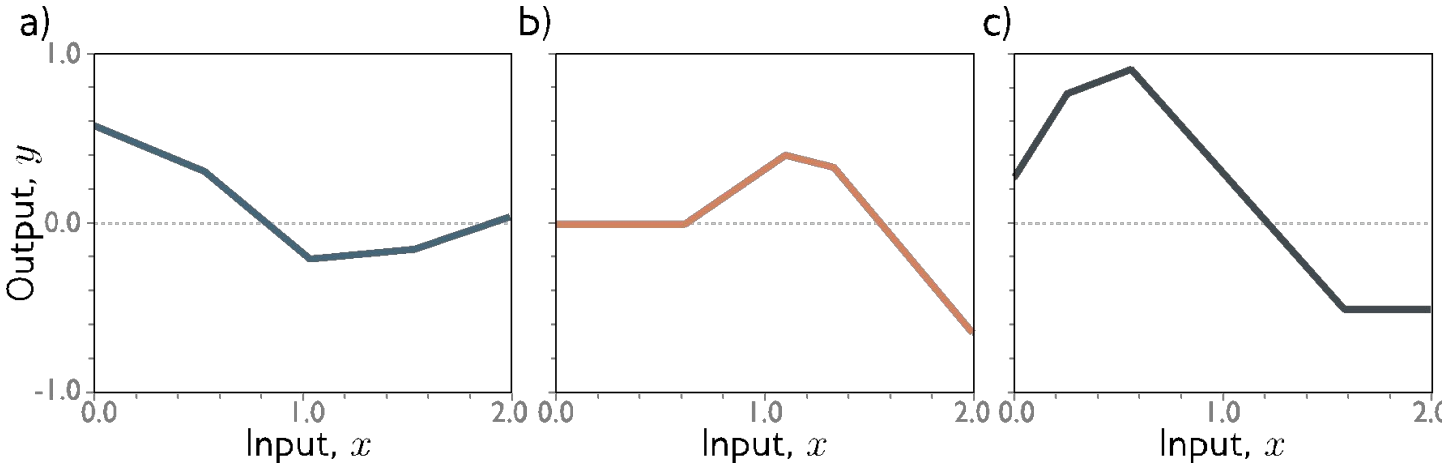
This model has 10 parameters:

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

- Represents a family of functions
- Parameters determine particular function
- Given parameters, it can perform inference (run equation)
- Given training dataset
  - Define loss function (eg.: least squares)
  - Change parameters to minimize loss function

Shallow neural network

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



Piecewise linear functions with three joints

Shallow neural network

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

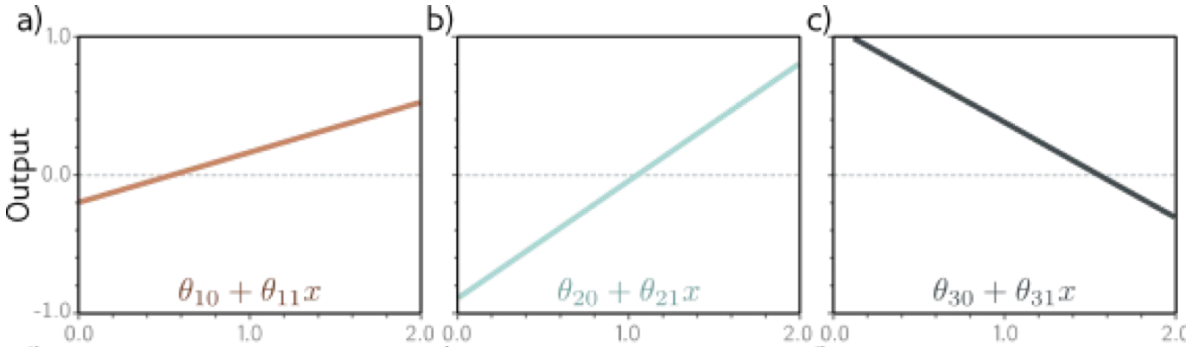
Break it down into two parts:

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

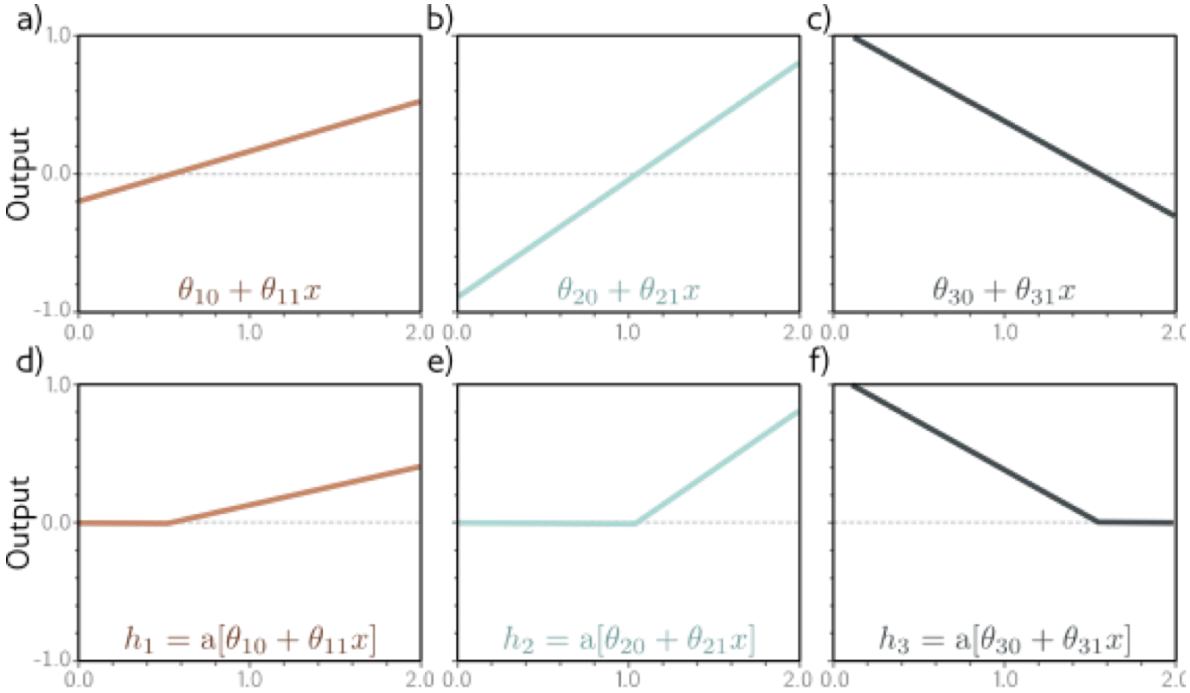
where:

hidden units

$$
\begin{cases}
h_1 = \mathrm{a}[\theta_{10} + \theta_{11} x] \\
h_2 = \mathrm{a}[\theta_{20} + \theta_{21} x] \\
h_3 = \mathrm{a}[\theta_{30} + \theta_{31} x]
\end{cases}
$$

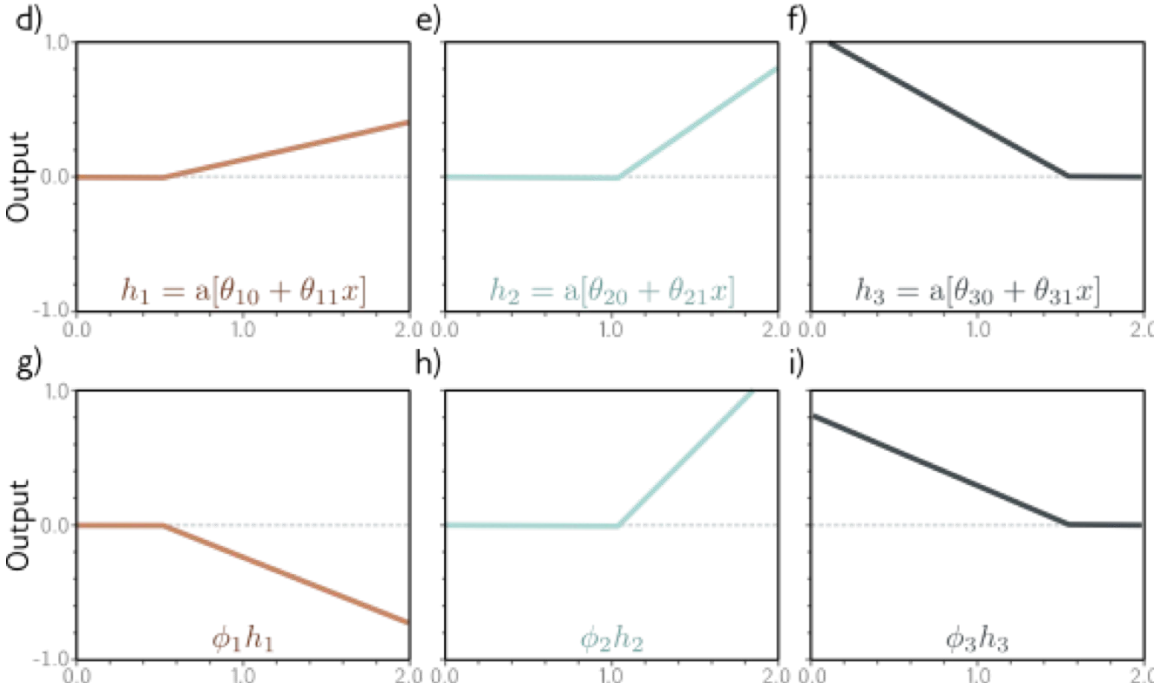


1. Compute three linear functions

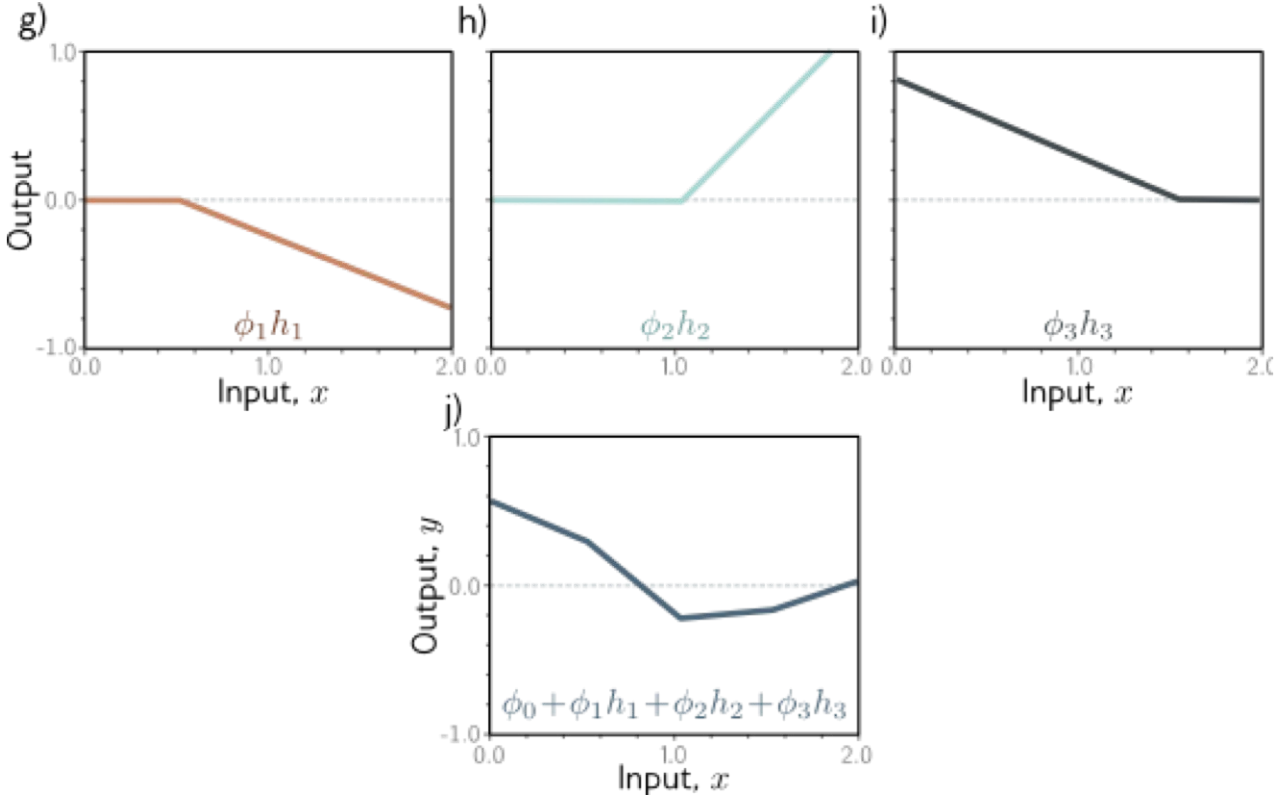


2. Pass through ReLU activations (hidden units)

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x],$$



3. Weight the hidden units

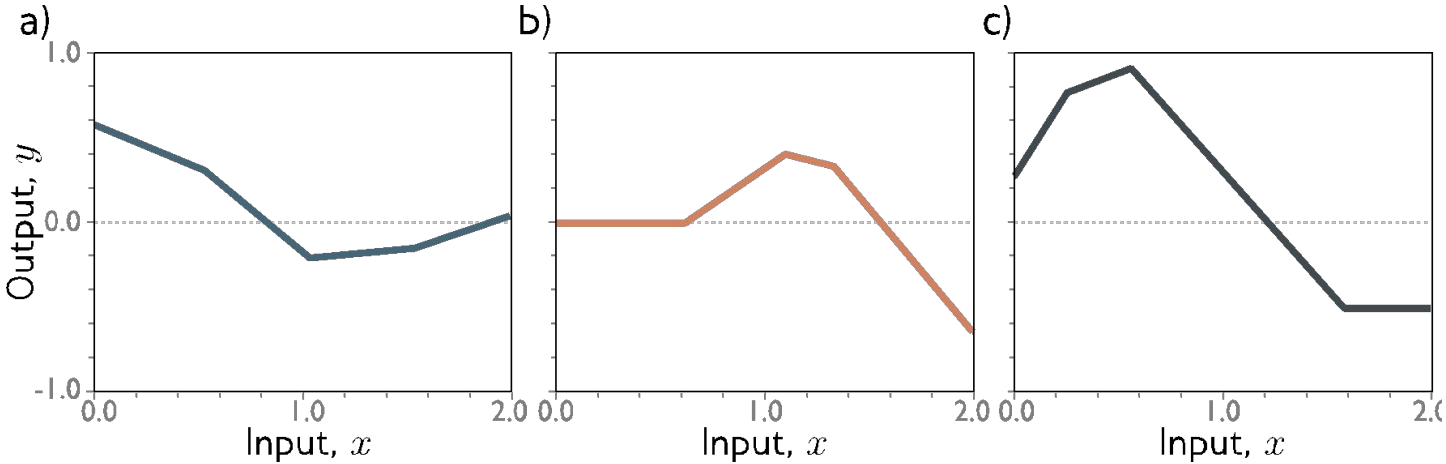


4. Sum the weighted activations

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

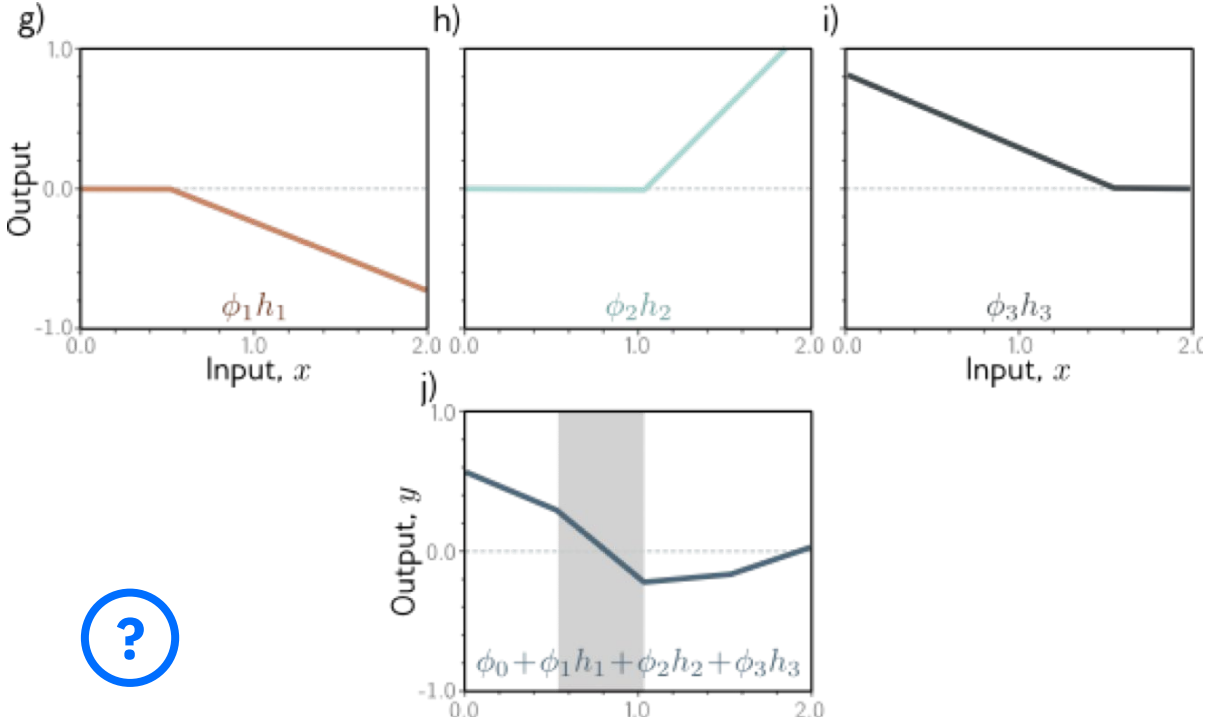
Shallow neural network

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



Piecewise linear functions

Activation pattern: which hidden units are activated

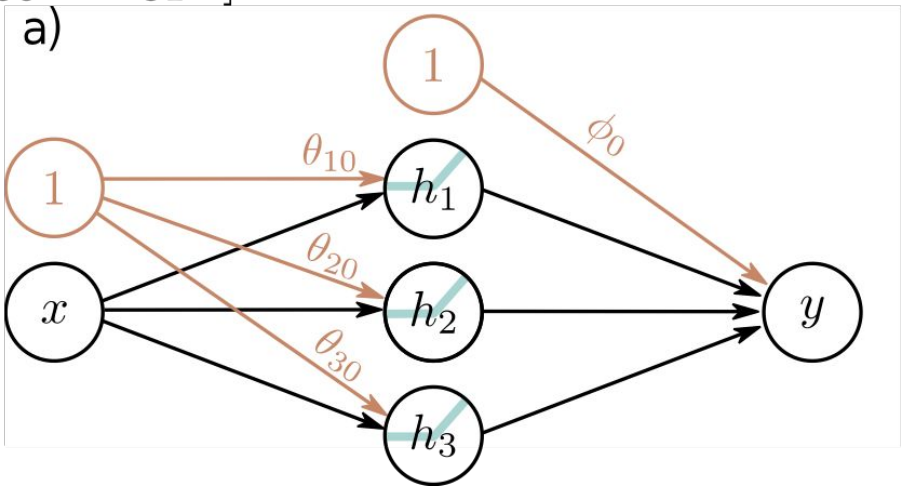


Shaded region: unit 1 active, unit 2 inactive, unit 3 active

Depicting neural networks

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

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



Each parameter multiplies its sources and adds to its target

Universal approximation theorem

Arbitrary number of hidden units

From 3 hidden units:

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

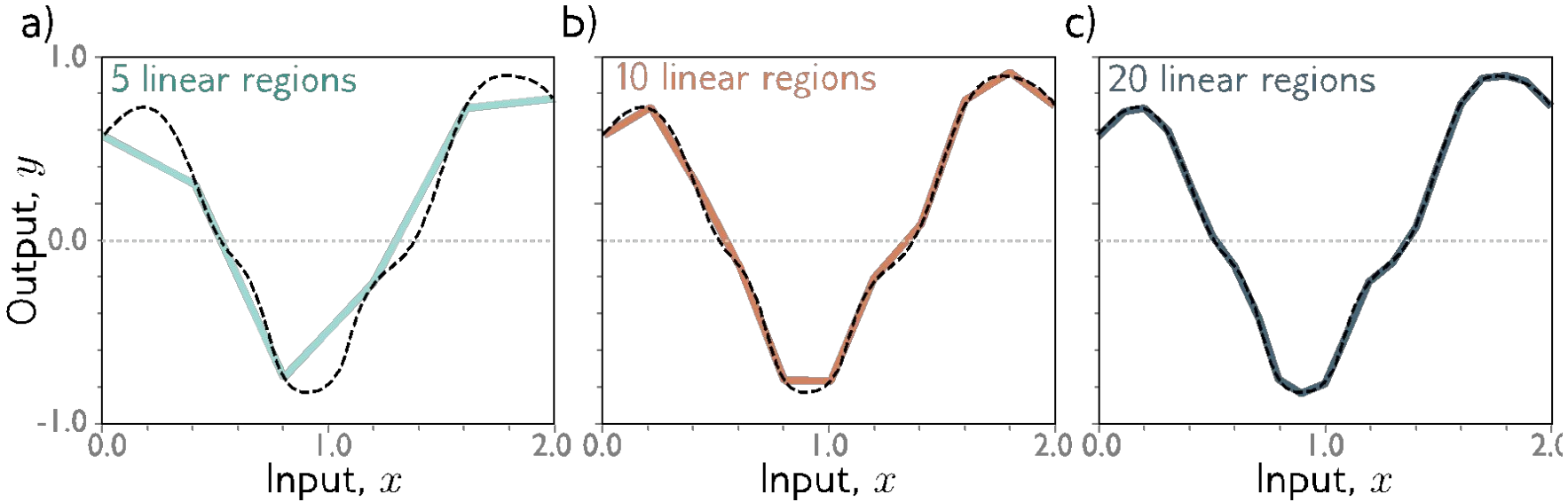
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

To D hidden units:

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

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

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 on a compact subset of $\mathbb{R}^D$ to arbitrary precision"

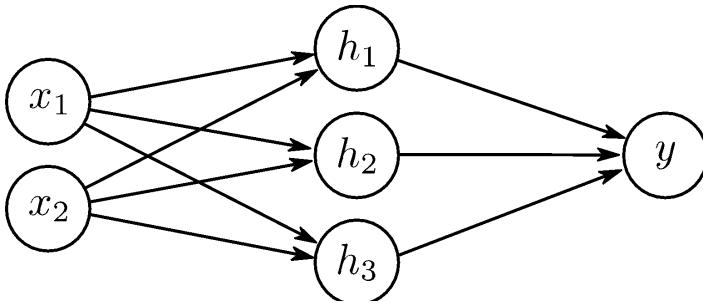
Hornik, 1990

* without guaranteeing a construction though!

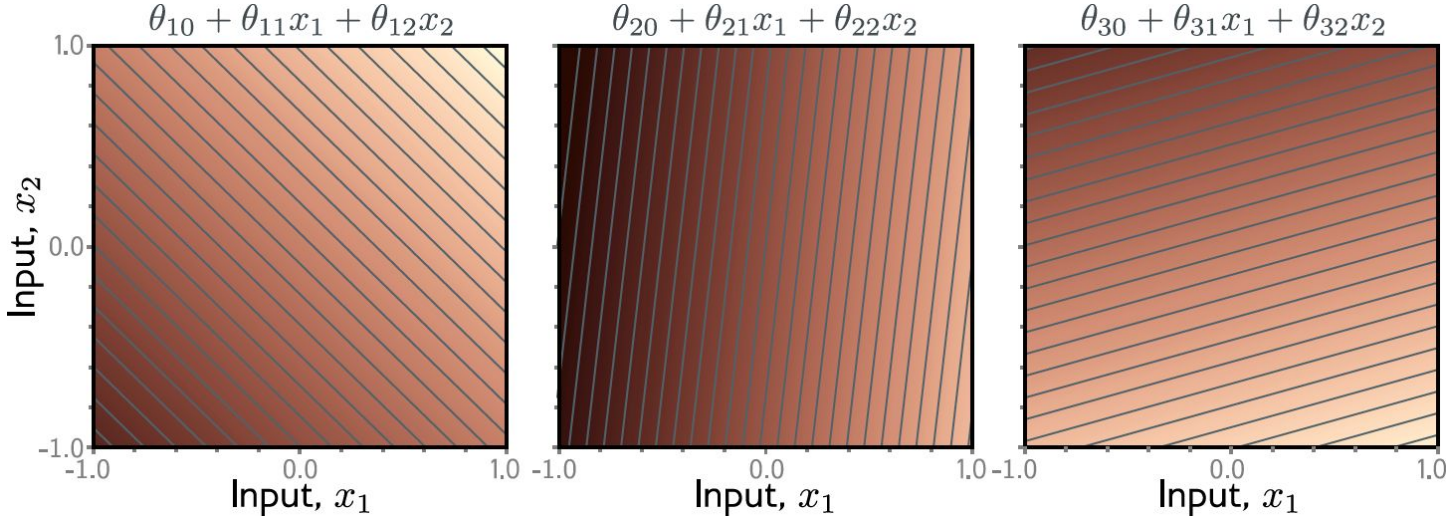
Multivariate *inputs*

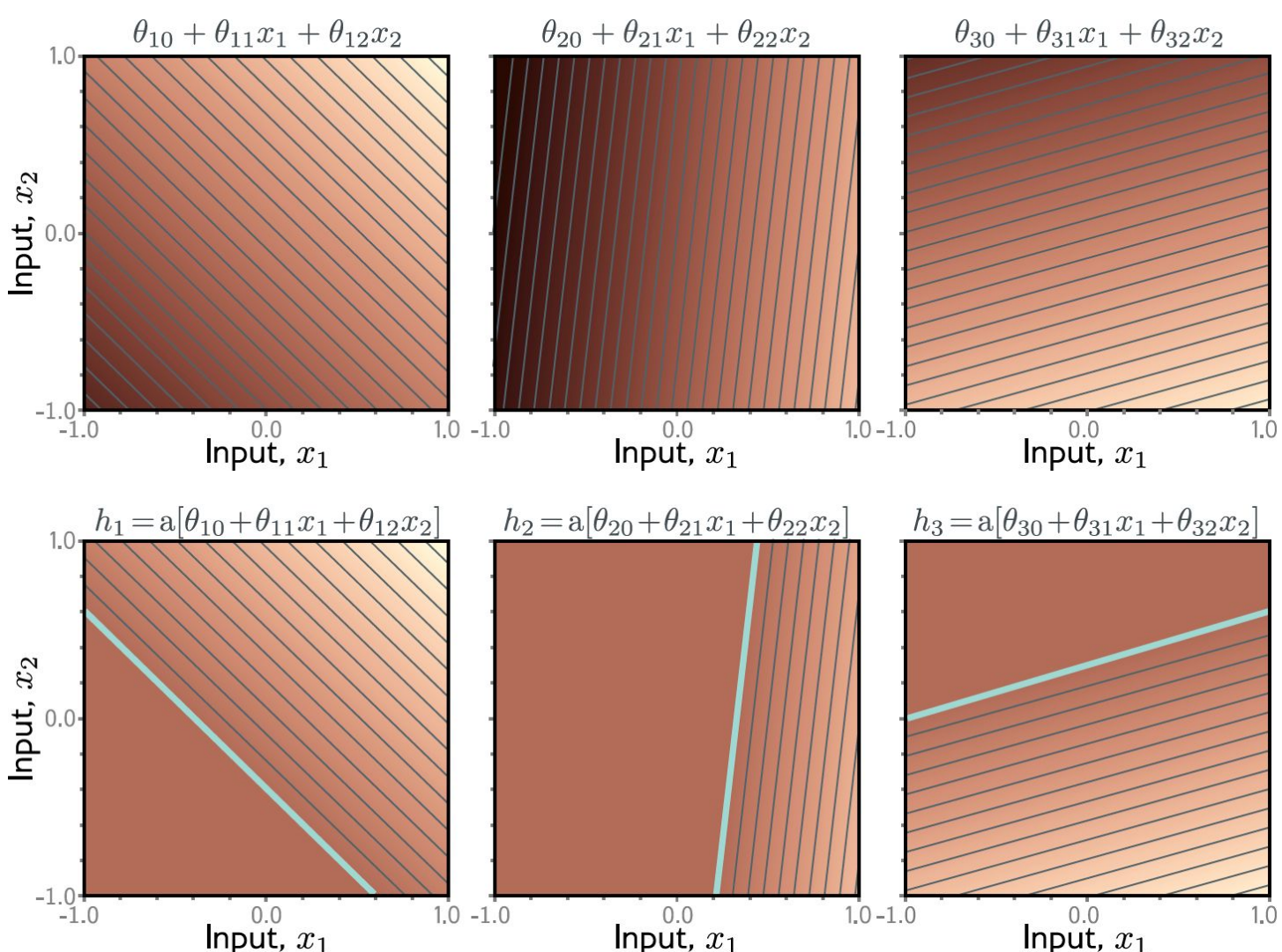
$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$$

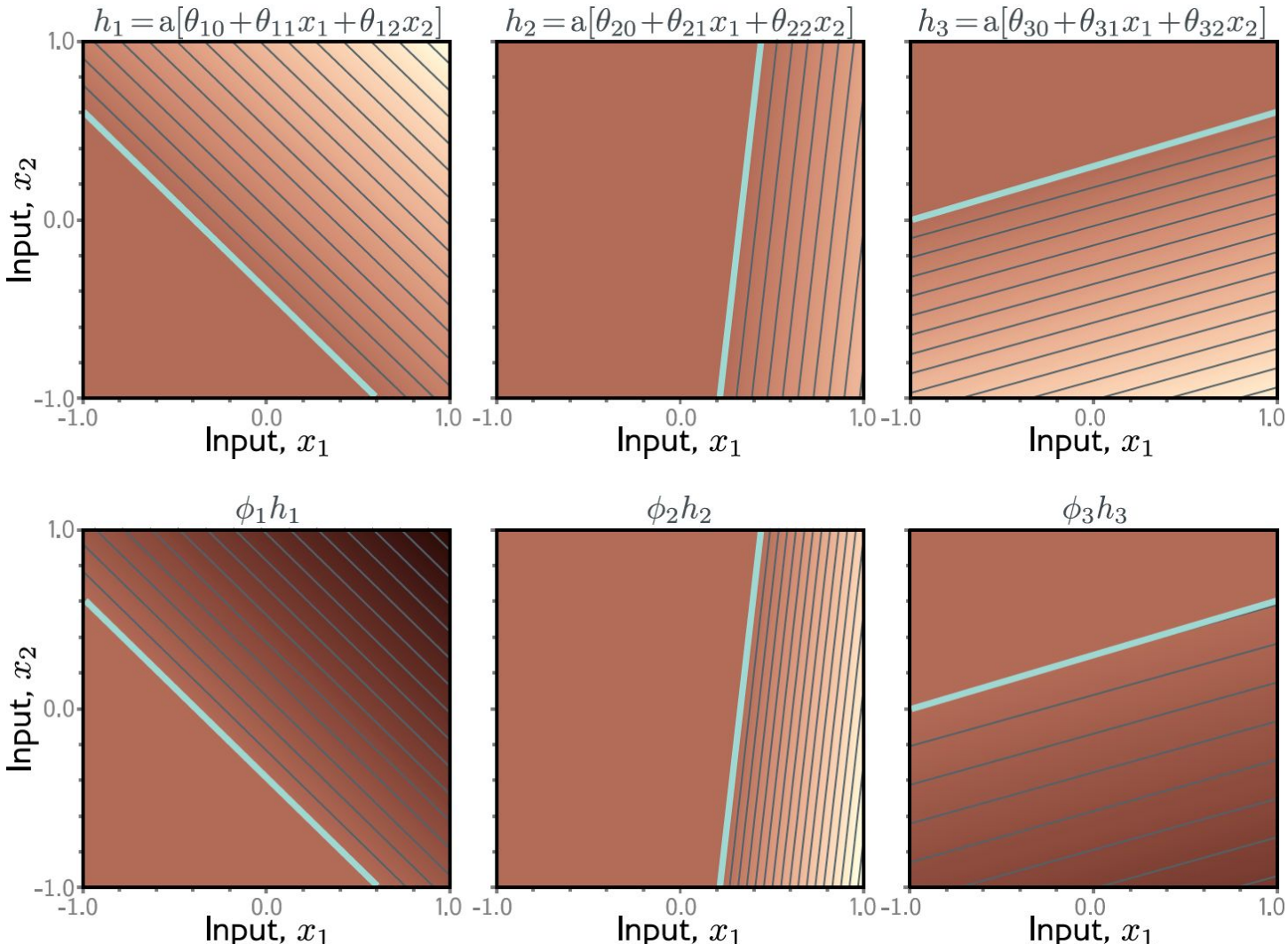
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

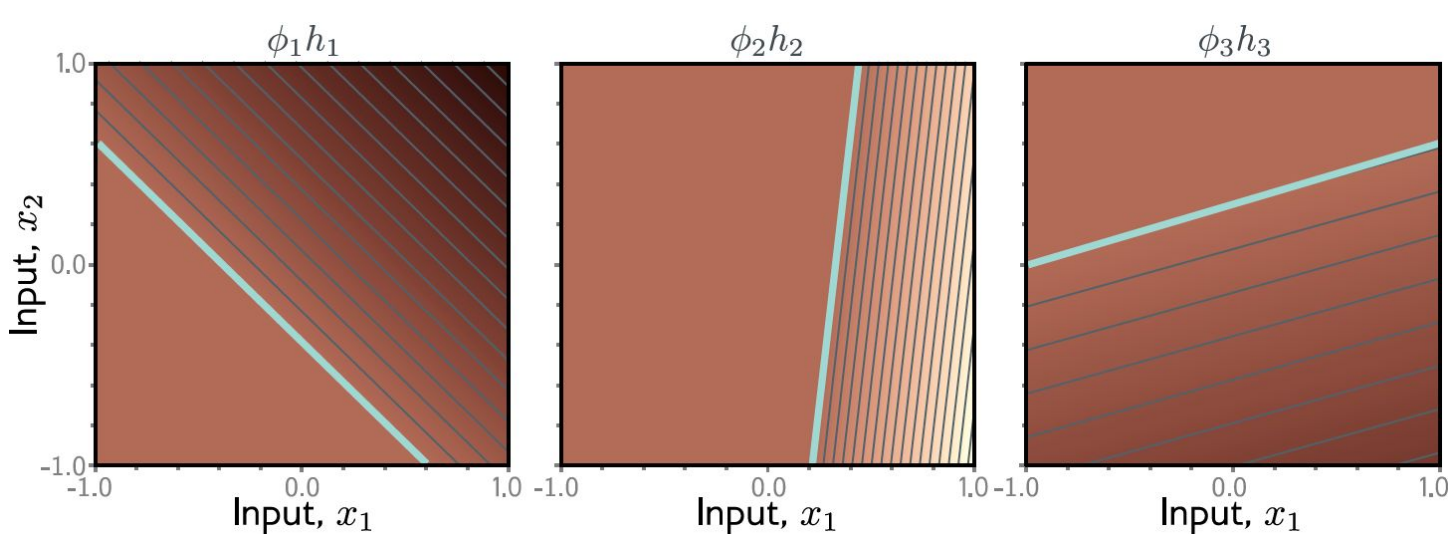


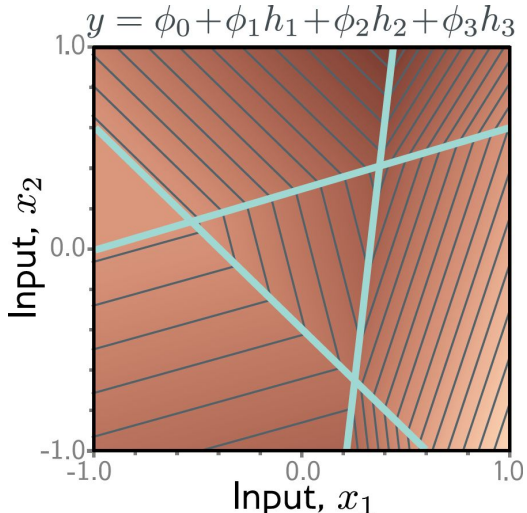
2 inputs, 3 hidden units, 1 output

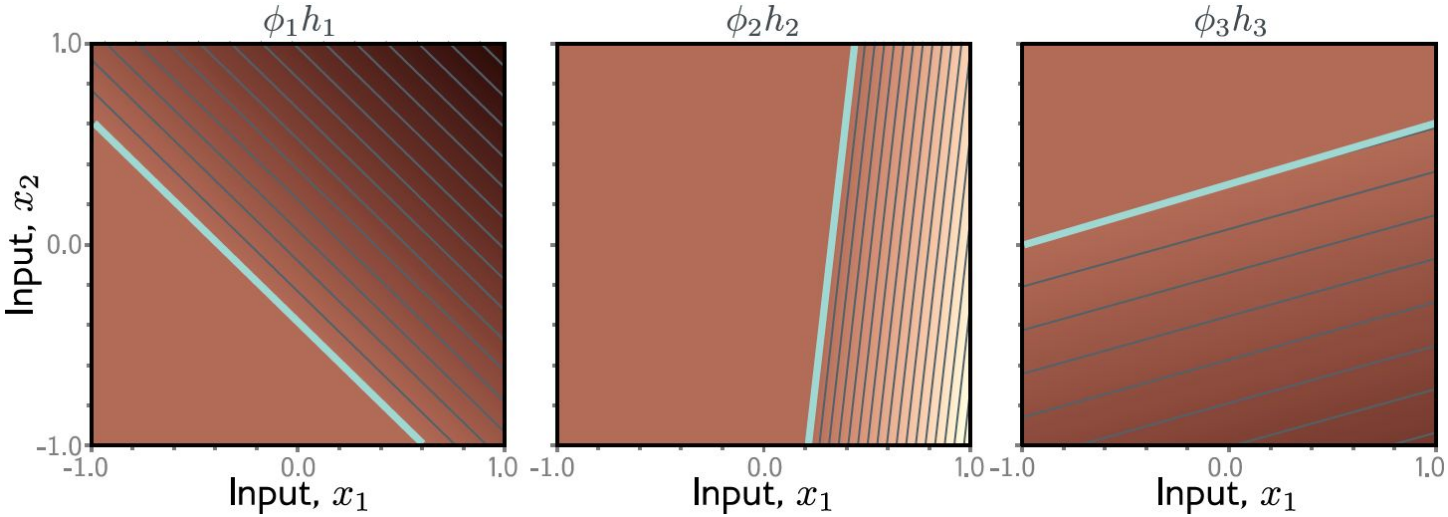

$\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2$
$\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2$
$\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2$
Input, x_2
1.0
0.0
-1.0
-1.0
0.0
1.0
Input, x_1


$\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2$
$\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2$
$\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2$
$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$
$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$
$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$
Input, x_1
Input, x_2
1.0
0.0
-1.0

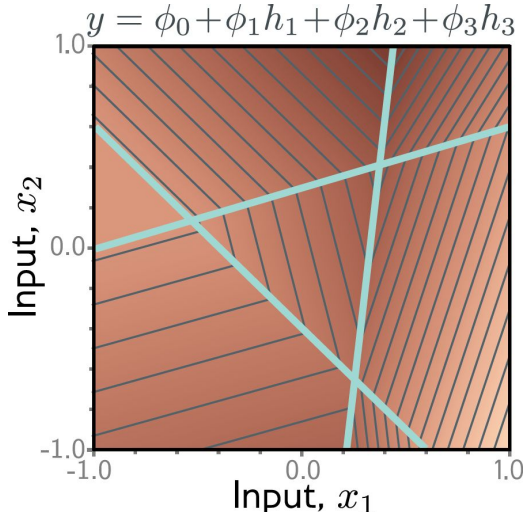

$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$
$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$
$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$
$\phi_1 h_1$
$\phi_2 h_2$
$\phi_3 h_3$
Input, x_1
Input, x_2
1.0
0.0
-1.0


$\phi_1 h_1$
$\phi_2 h_2$
$\phi_3 h_3$
1.0
0.0
-1.0
Input, x_2
Input, x_1


$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$
1.0
0.0
-1.0
Input, x_2
Input, x_1

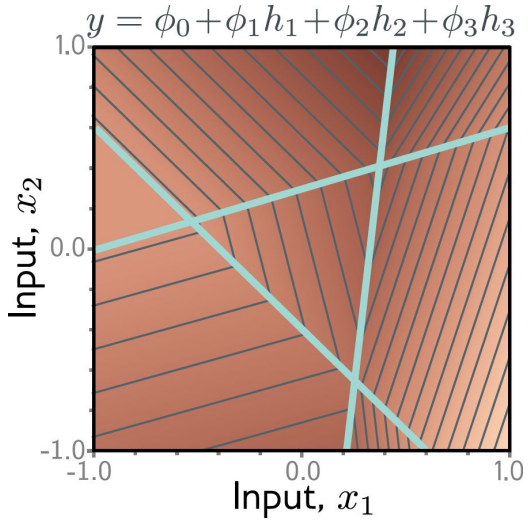


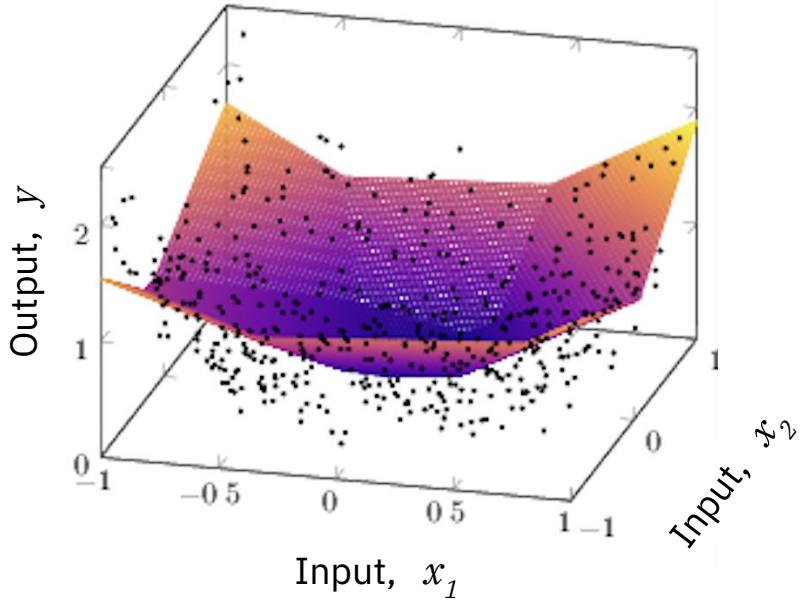
?



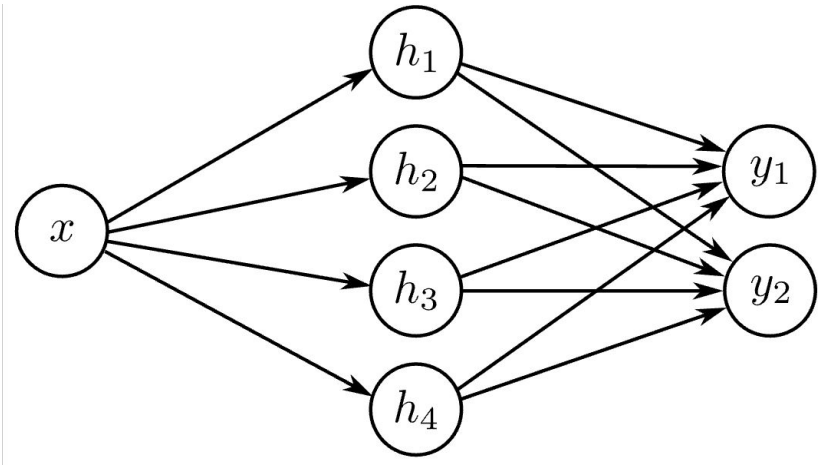
Convex polygons

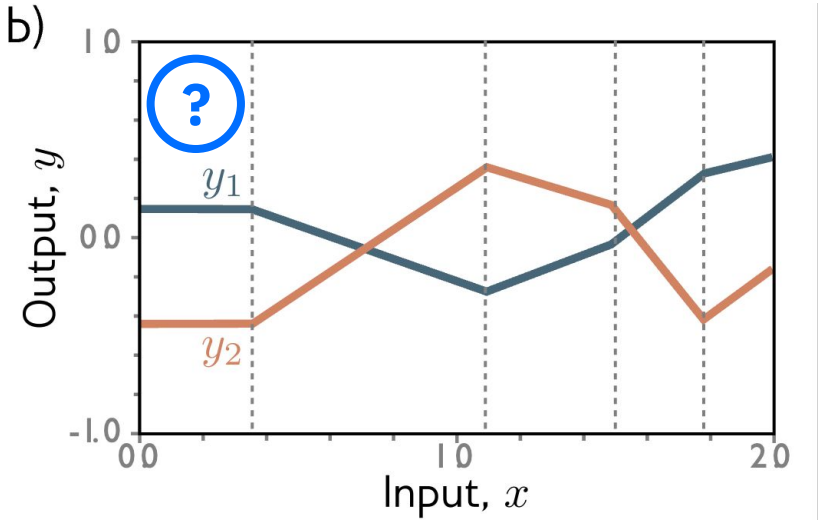
Fitting


$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$
1.0
0.0
-1.0
-1.0
0.0
1.0
Input, x_2
Input, x_1


Output, y
2
1
0
−1
−0 5
0
0 5
1
−1
0
1
Input, x_2
Input, x_1

Multivariate *outputs*





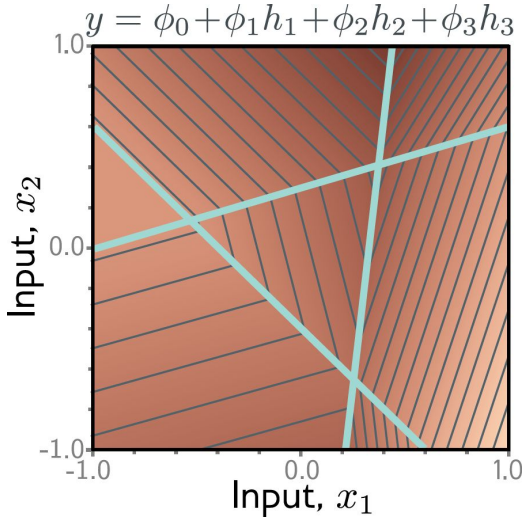
1 input, 4 hidden units, 1 output

B

Question:

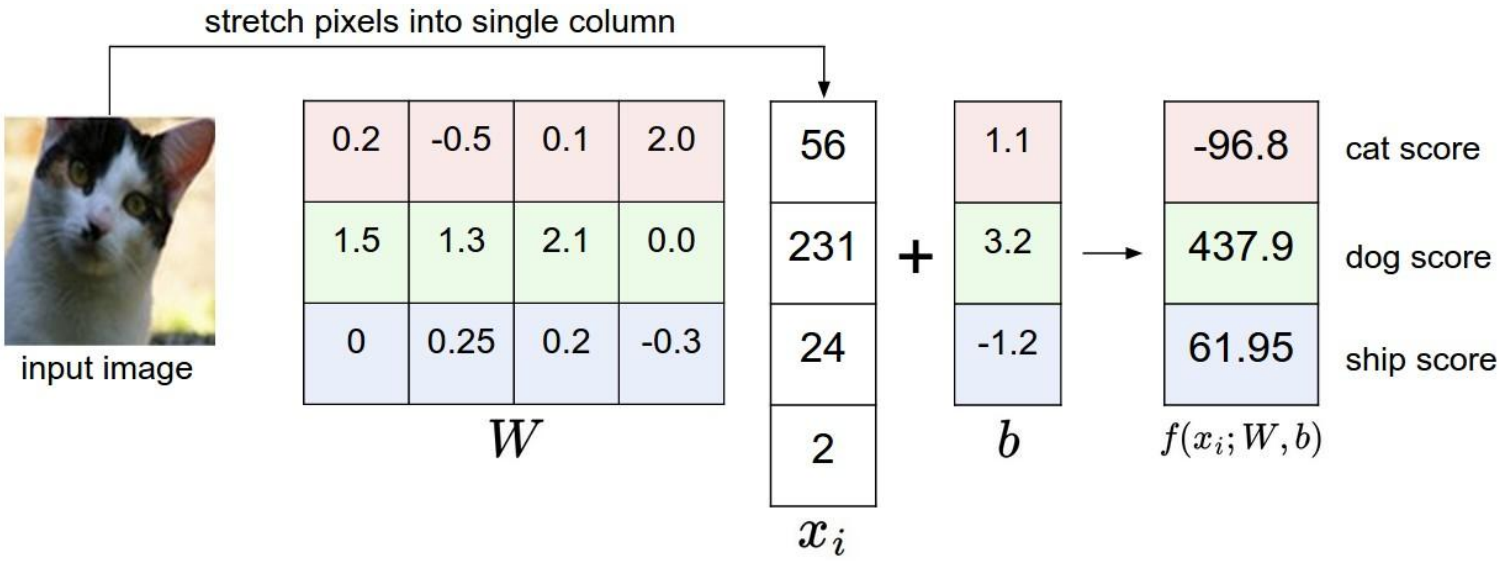
For the 2D case, what if there were two outputs?

If this is one of the outputs, what would the other one look like?

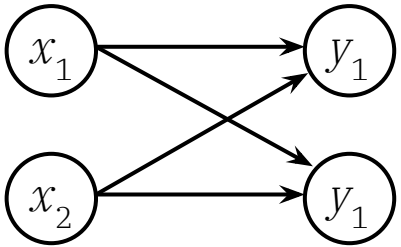


Functional *vs.* Representational

Linear classification



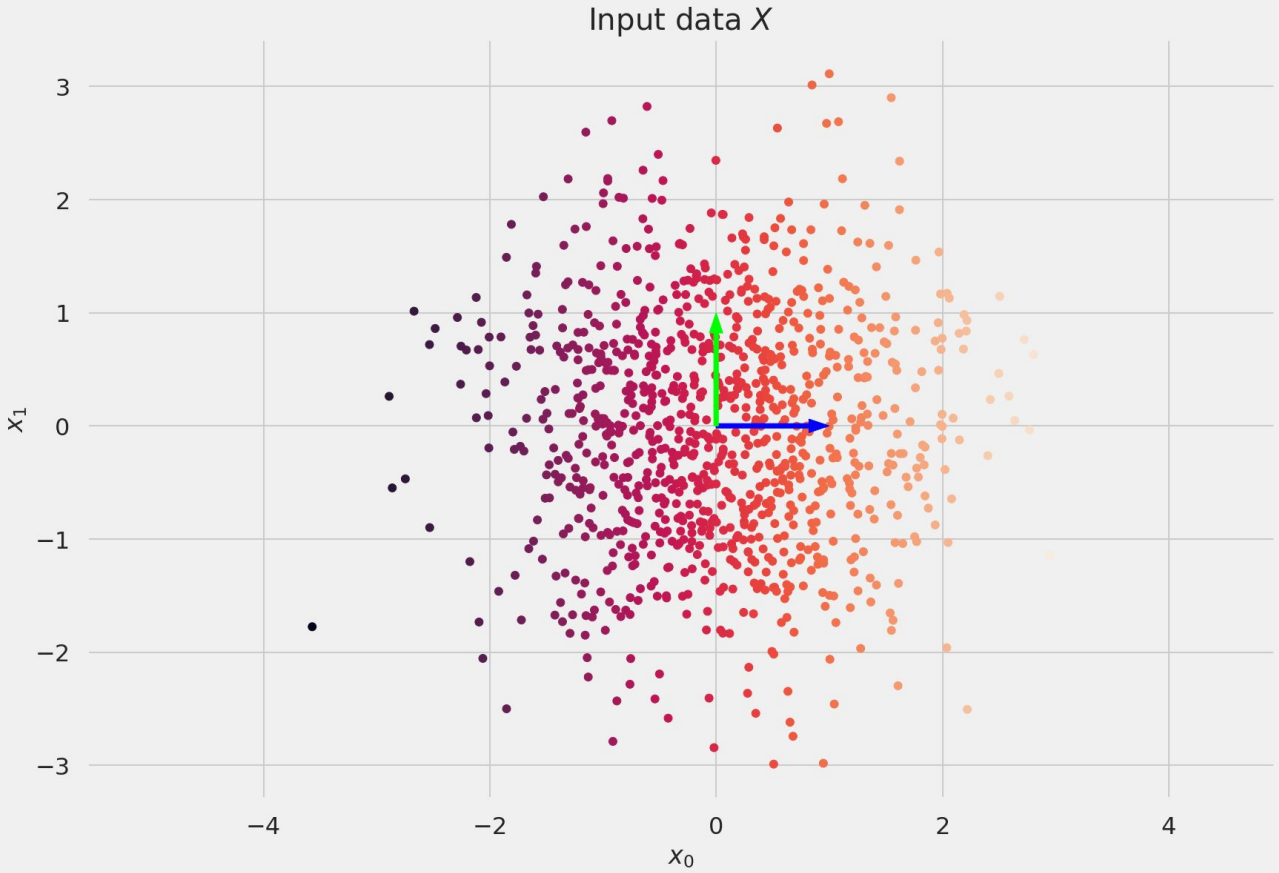
The *representation* view. Linear model



$\mathbf{y} = \boldsymbol{\Theta}\mathbf{x}$, that is $y_1 = \theta_{11}x_1 + \theta_{12}x_2$, ...

A linear model with four weights.
Also can be seen as two stacked ”neurons” without activation functions.

The *representation* view. Linear model



The *representation* view. Linear model

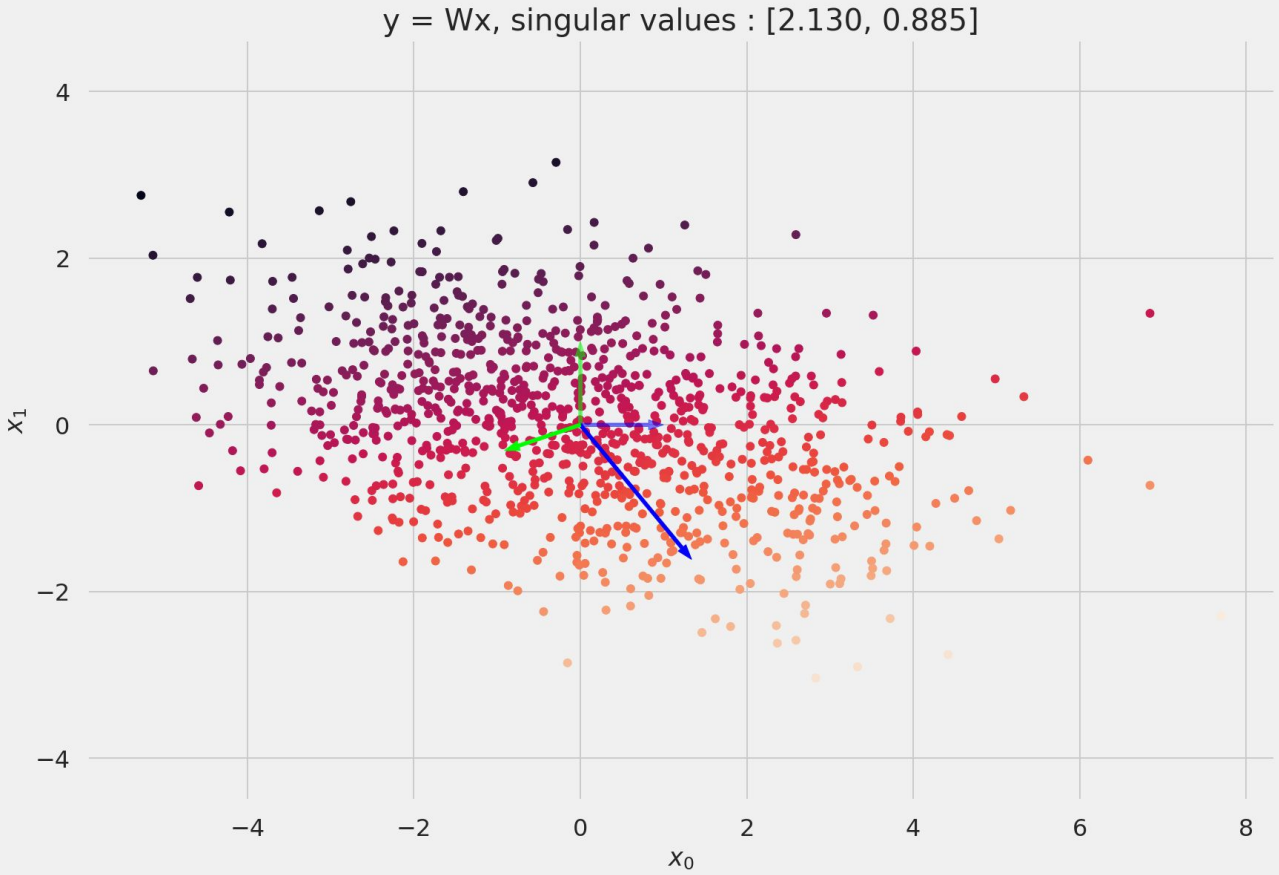
- Our model is: $\mathbf{y} = \boldsymbol{\Theta}\mathbf{x}$
- What is $\boldsymbol{\Theta}$ doing to $\mathbf{x}$?
- Let's perform *singular value decomposition** for some intuition:

$$\boldsymbol{\Theta} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\top}$$

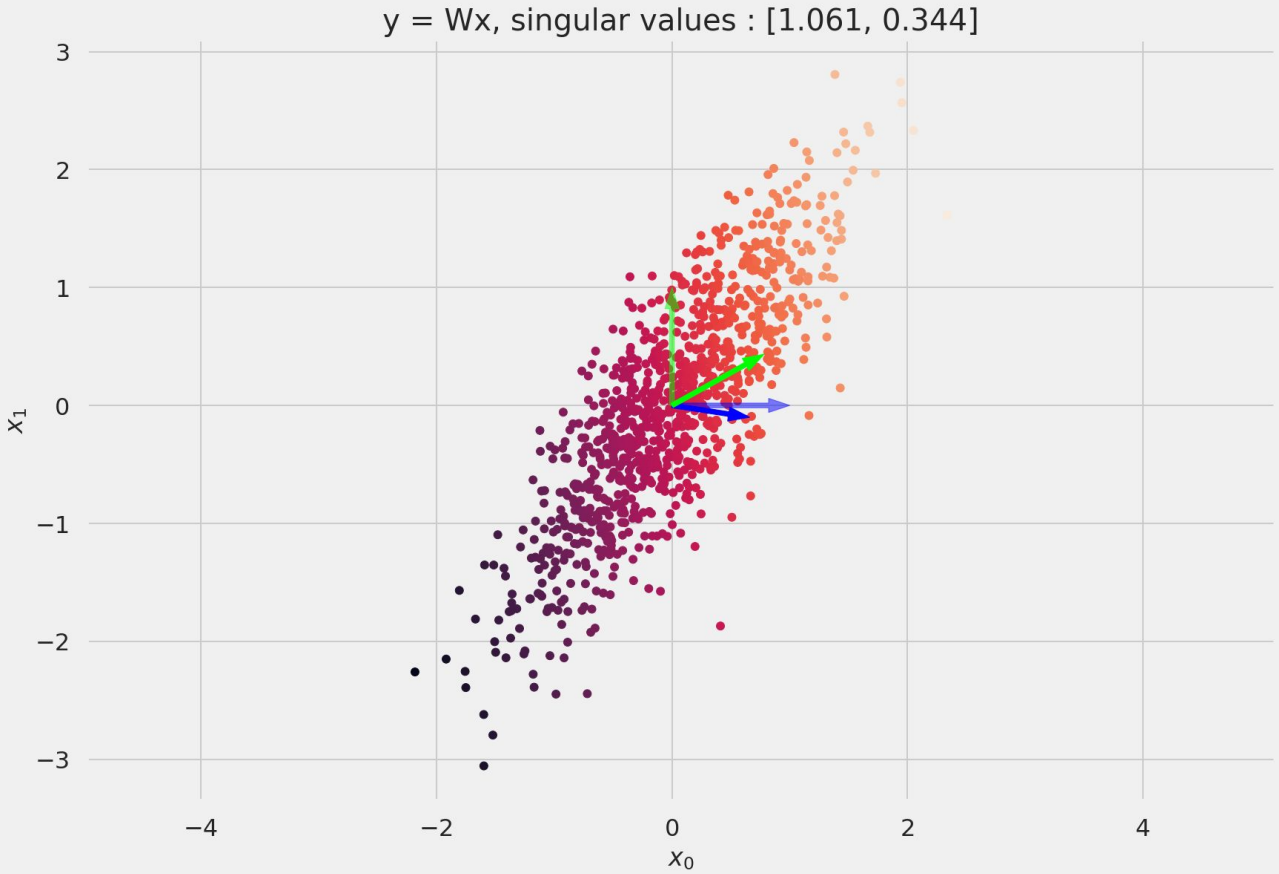
$$\begin{bmatrix} 0.19 & 1.84 \\ -0.97 & -0.93 \end{bmatrix} = \underbrace{\begin{bmatrix} 0.83 & -0.55 \\ -0.55 & -0.83 \end{bmatrix}}_{\text{rotation}} \times \underbrace{\begin{bmatrix} 2.17 & 0.00 \\ 0.00 & 0.74 \end{bmatrix}}_{\text{scale}} \times \underbrace{\begin{bmatrix} 0.32 & 0.94 \\ 0.94 & -0.32 \end{bmatrix}}_{\text{reflection}}$$

* Deisenroth, *Mathematics for Deep Learning*, ch. 4.

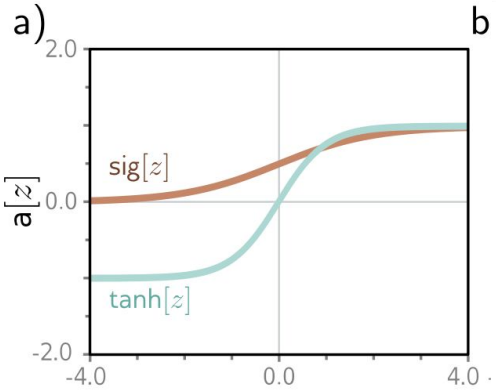
The *representation* view. Linear model



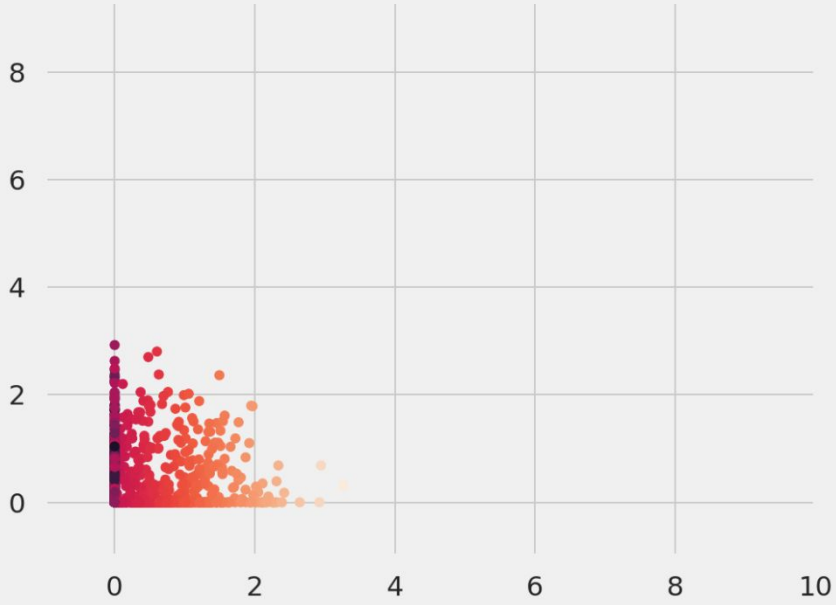
The *representation* view. Linear model

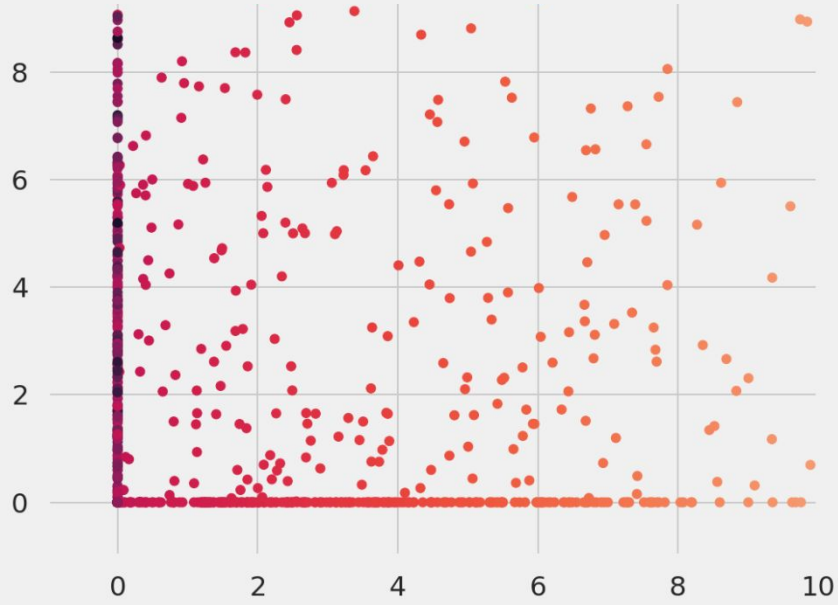


The *representation* view. Activation functions



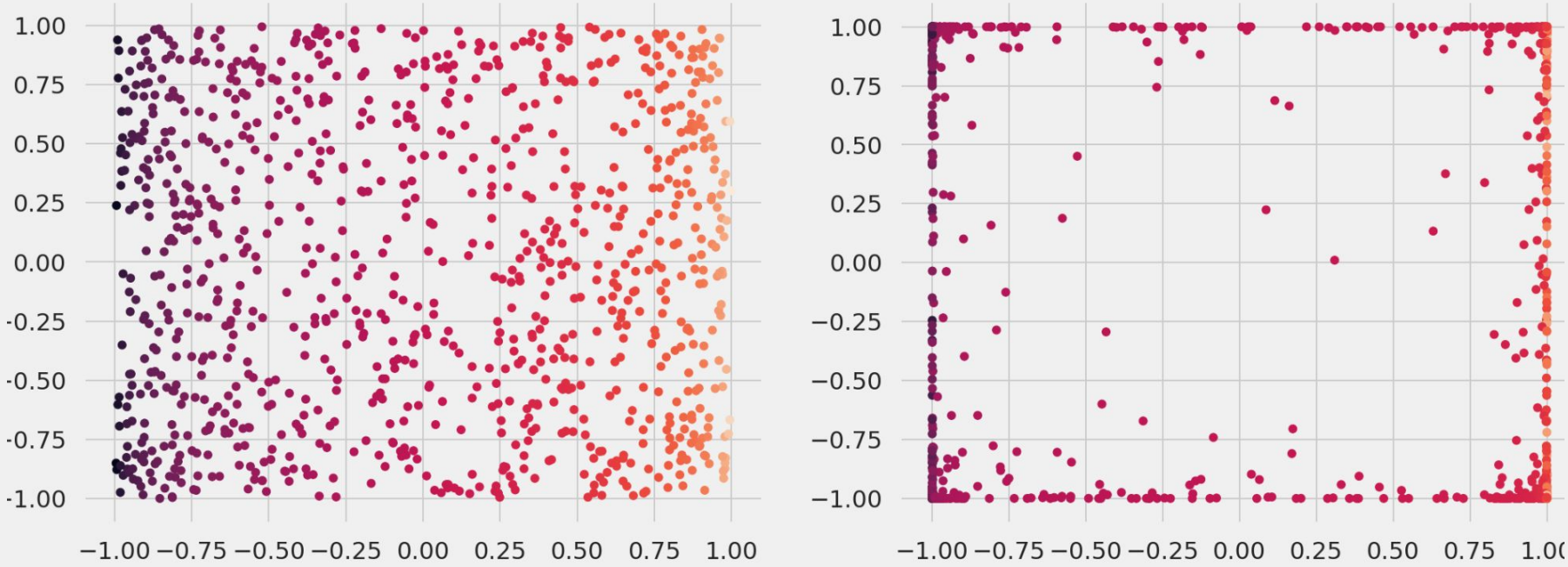
The representation view. *Non*-linear model | ReLU





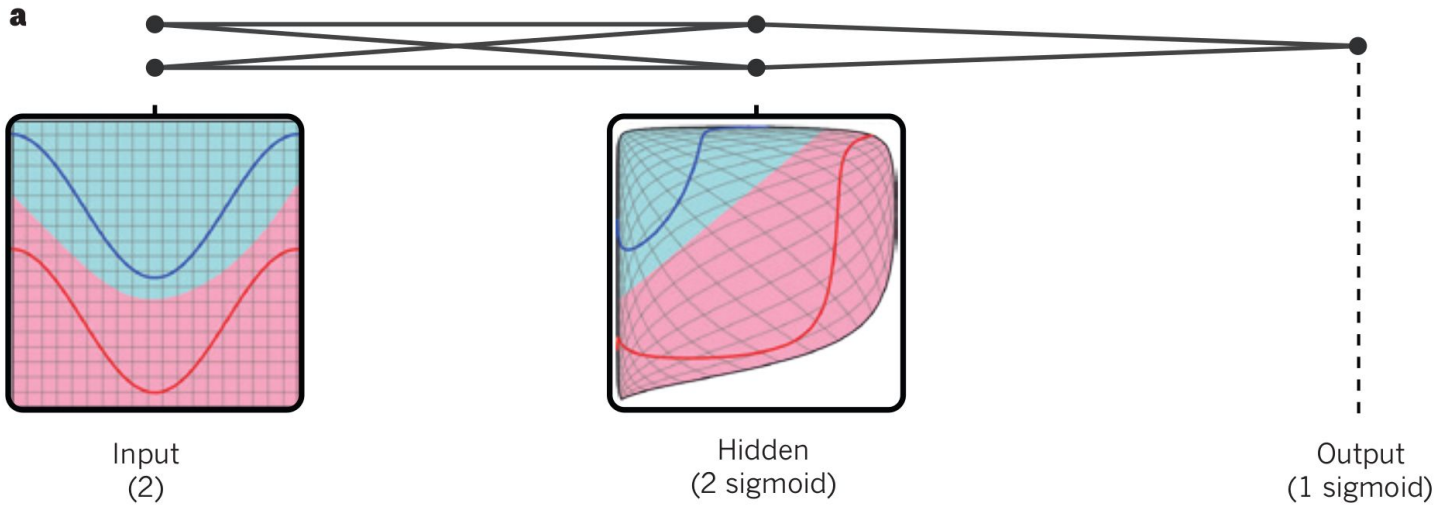
ReLU(**Sx**) activation function with two different scale factors S

The representation view. *Non*-linear model | tanh

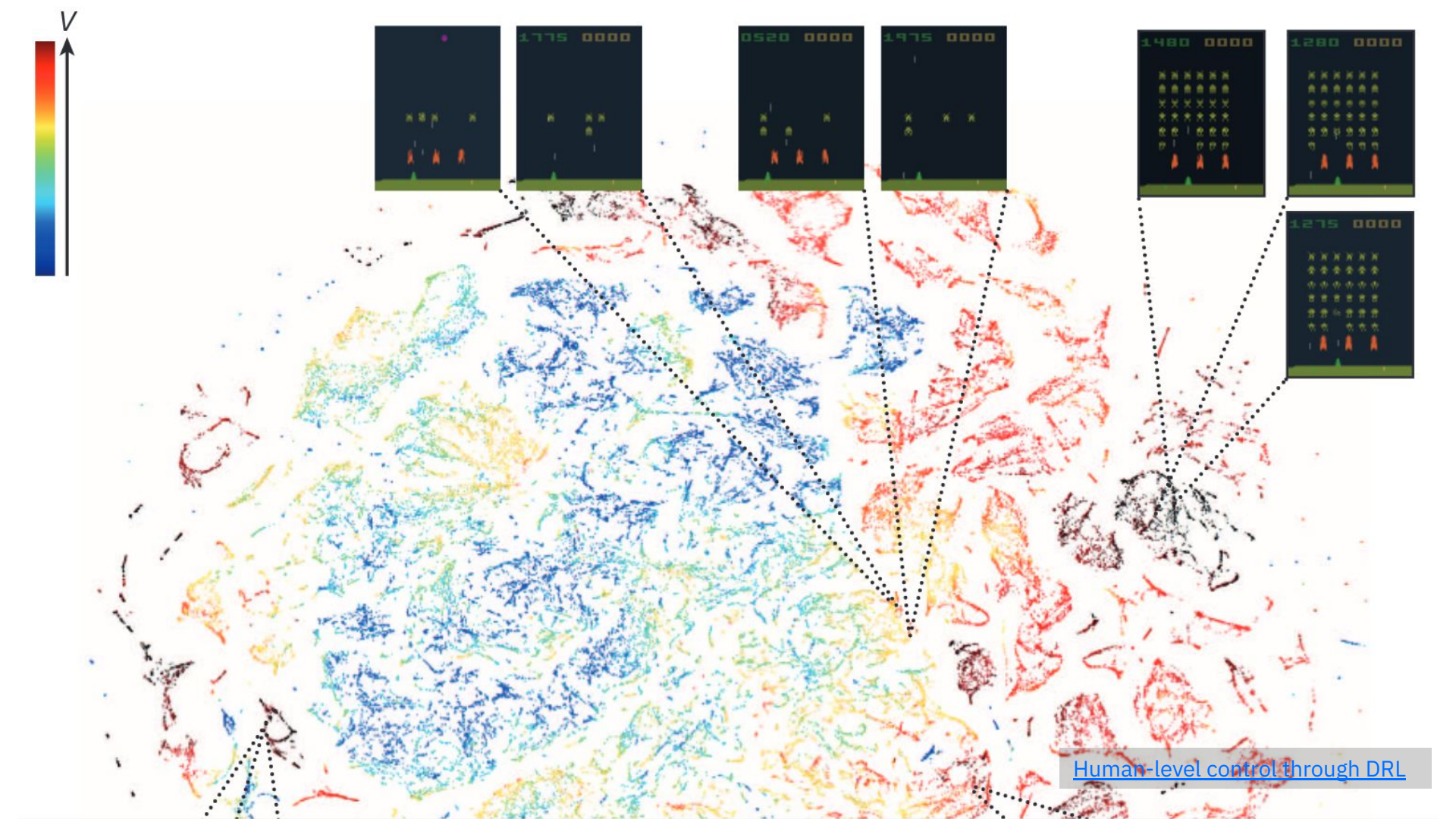


tanh(**Sx**) activation function with two different scale factors S

The representation view. *Warping* space



A neural network with two inputs, two hidden units and one output “warps” space so that data can become linearly separable.



Human-level control through DRL

Functional view

- Usually well defined (eg. “linear regions”)
- Useful for the theory of neural networks
- Better at explaining the preference for *deep* neural networks

Representation view

- Not well defined (in general)
- Many methods in the literature describe themselves as “learning representations”
- Drove the research in *transfer* and *self-supervised* learning for some time
- Shows up in natural language processing also, see “thought vectors”
- Neural network depth associated with learning *hierarchical representations*

Deep Neural Networks

Reading suggestion

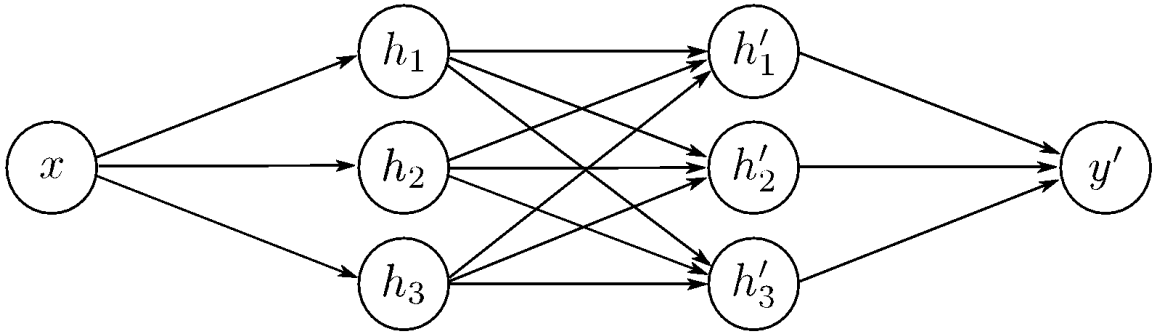
Check 4.1 and 4.2 in your textbook for a discussion on a special case of deep neural networks not covered in these slides.

Two-layer networks

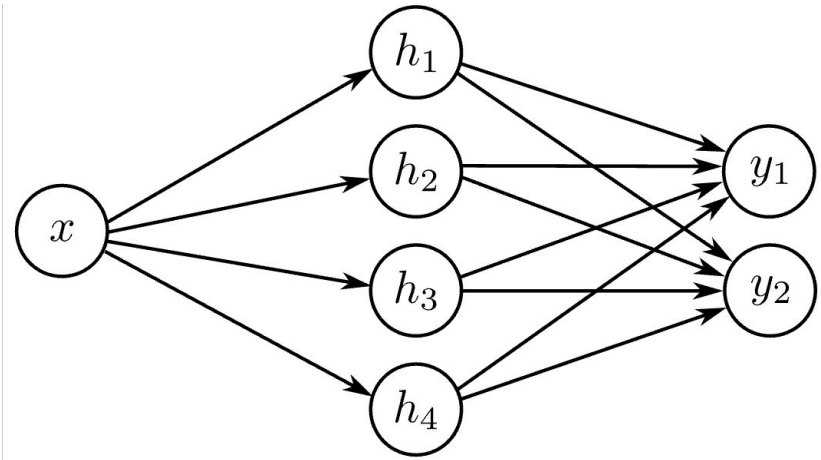
$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

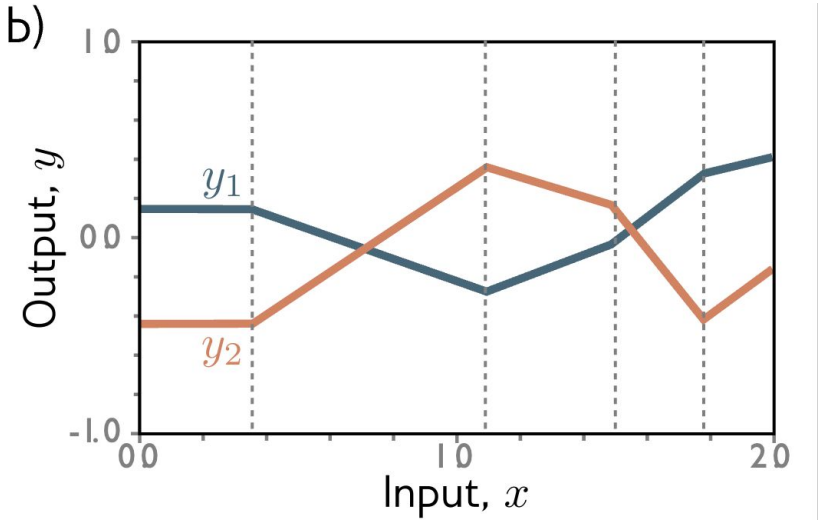
$$h'_1 = \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$
$$h'_2 = \mathrm{a}[\psi_{20} + \psi_{21}h_2 + \psi_{22}h_2 + \psi_{23}h_3]$$
$$h'_3 = \mathrm{a}[\psi_{30} + \psi_{31}h_2 + \psi_{32}h_2 + \psi_{33}h_3]$$

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$



Remember *shallow* network with two outputs





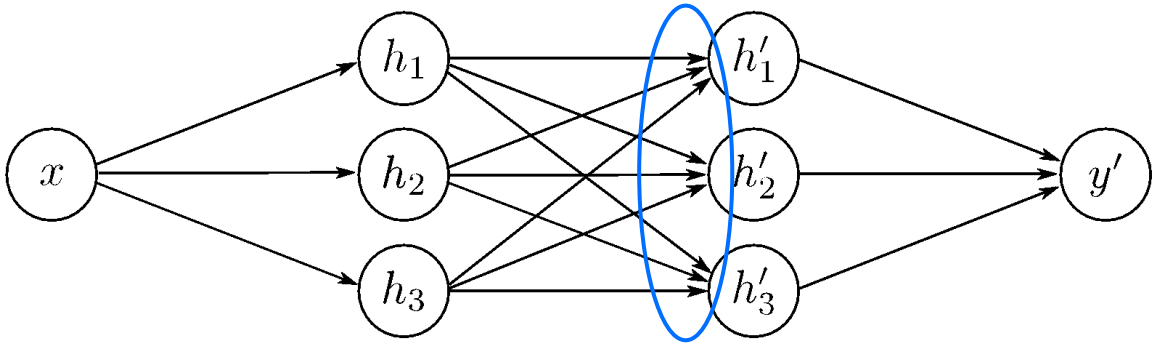
1 input, 4 hidden units, 1 output

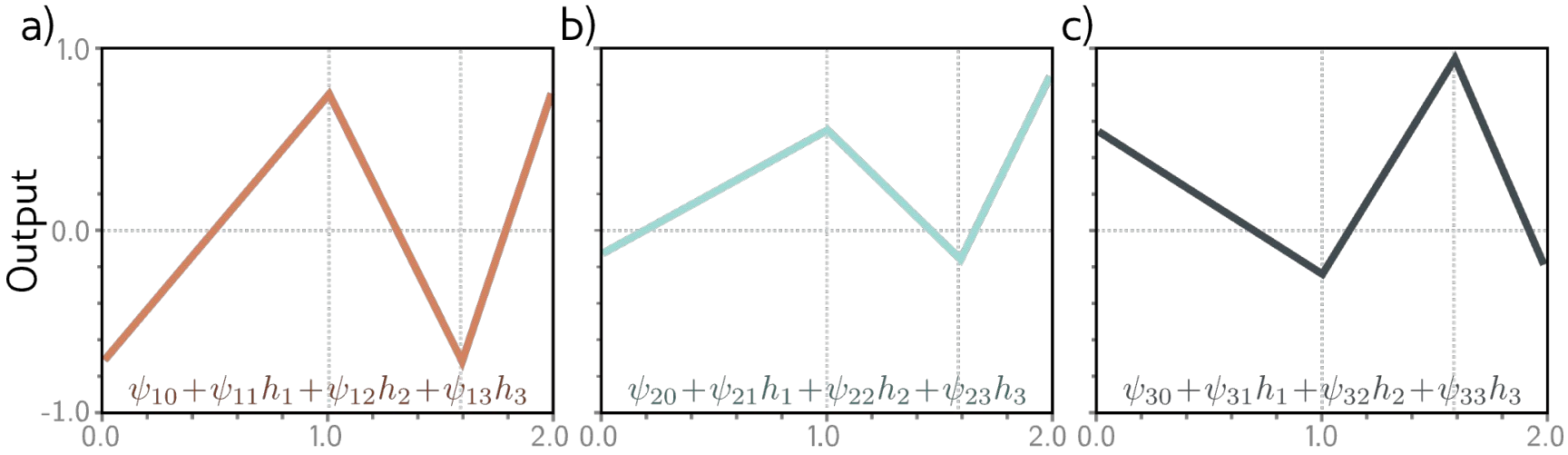
Deep networks are just function composition!

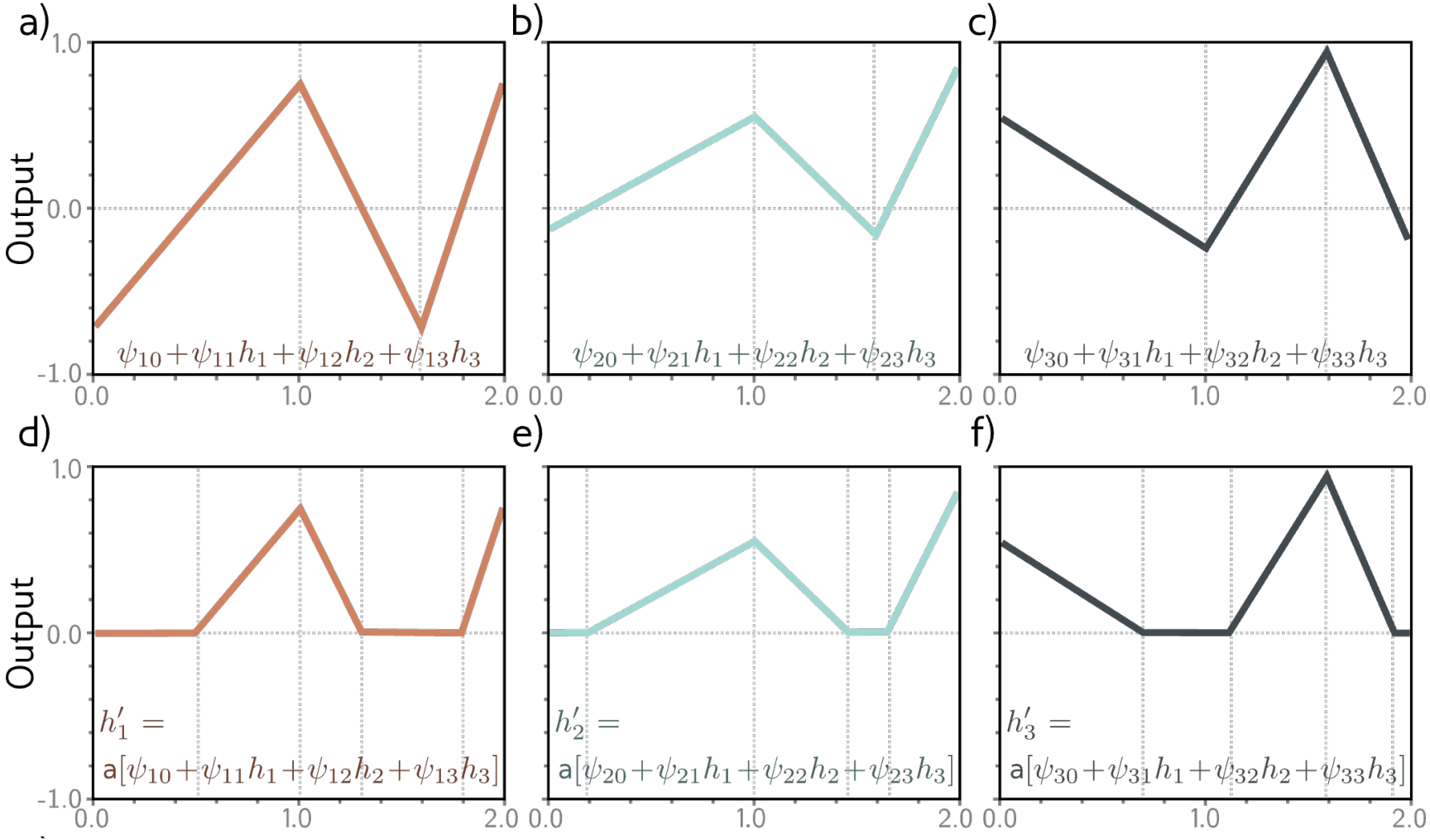
$$h_1 = a[\theta_{10} + \theta_{11}x]$$
$$h_2 = a[\theta_{20} + \theta_{21}x]$$
$$h_3 = a[\theta_{30} + \theta_{31}x]$$

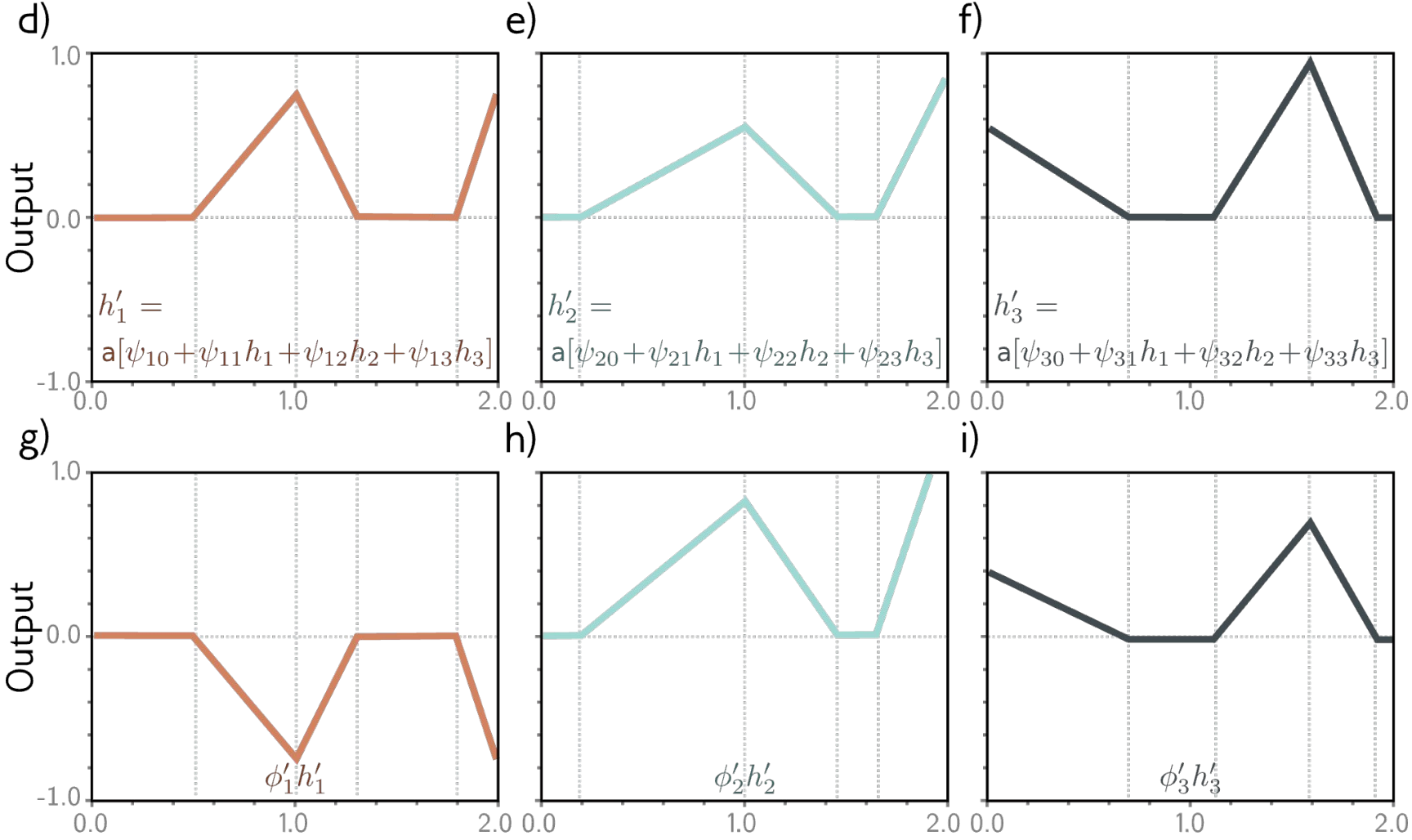
$$h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$
$$h'_2 = a[\psi_{20} + \psi_{21}h_2 + \psi_{22}h_2 + \psi_{23}h_3]$$
$$h'_3 = a[\psi_{30} + \psi_{31}h_2 + \psi_{32}h_2 + \psi_{33}h_3]$$

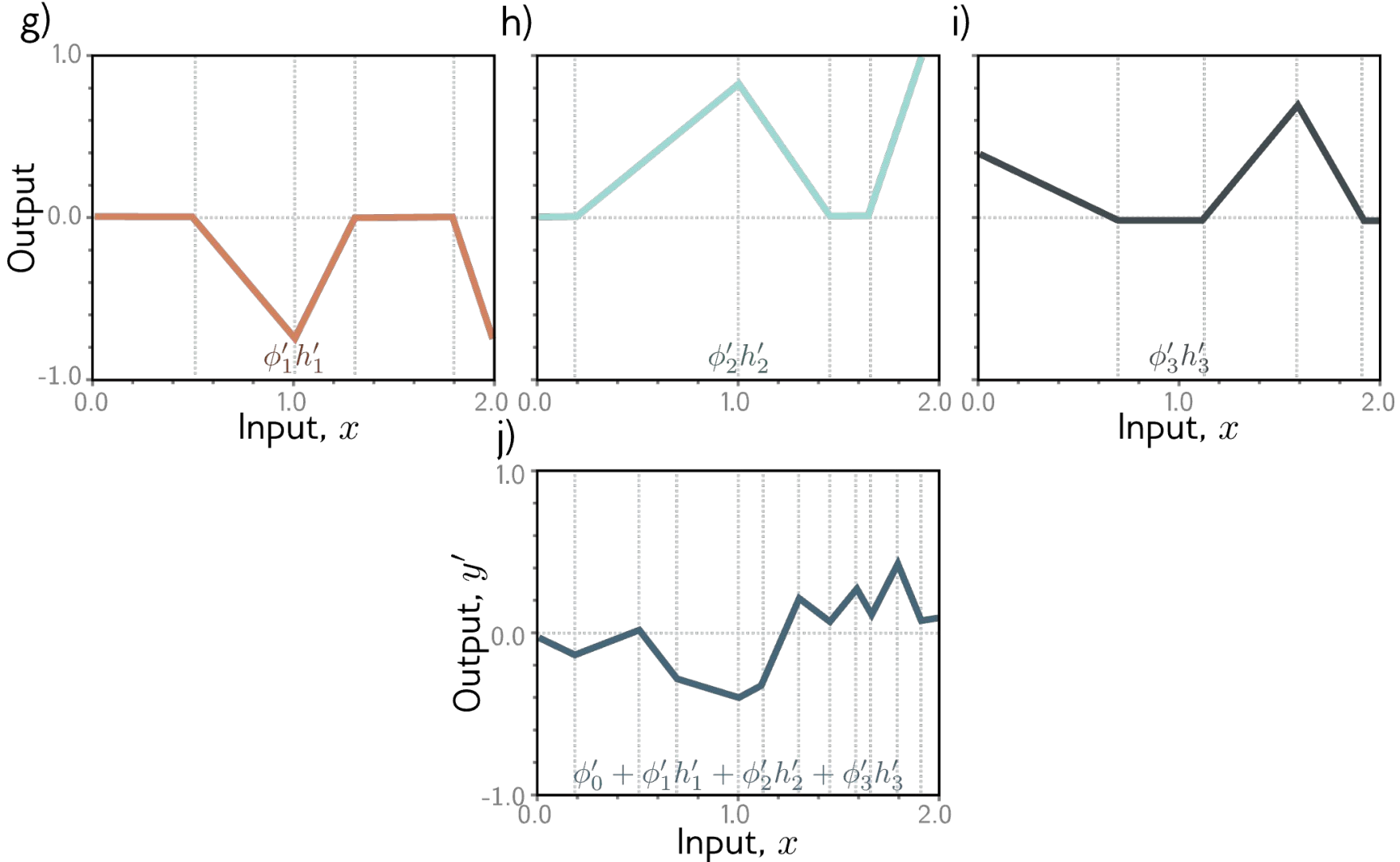
Consider the pre-activations at the second hidden units.
At this point, it's a one-layer network with three outputs.




a)
b)
c)
Output
1.0
0.0
-1.0
0.0
1.0
2.0
$\psi_{10}+\psi_{11}h_1+\psi_{12}h_2+\psi_{13}h_3$
$\psi_{20}+\psi_{21}h_1+\psi_{22}h_2+\psi_{23}h_3$
$\psi_{30}+\psi_{31}h_1+\psi_{32}h_2+\psi_{33}h_3$


a)
b)
c)
d)
e)
f)
Output
Output
1.0
0.0
-1.0
0.0
1.0
2.0
$\psi_{10}+\psi_{11}h_1+\psi_{12}h_2+\psi_{13}h_3$
$\psi_{20}+\psi_{21}h_1+\psi_{22}h_2+\psi_{23}h_3$
$\psi_{30}+\psi_{31}h_1+\psi_{32}h_2+\psi_{33}h_3$
$h_1' =$
$\mathsf{a}[\psi_{10}+\psi_{11}h_1+\psi_{12}h_2+\psi_{13}h_3]$
$h_2' =$
$\mathsf{a}[\psi_{20}+\psi_{21}h_1+\psi_{22}h_2+\psi_{23}h_3]$
$h_3' =$
$\mathsf{a}[\psi_{30}+\psi_{31}h_1+\psi_{32}h_2+\psi_{33}h_3]$


d)
e)
f)
g)
h)
i)
Output
Output
1.0
0.0
-1.0
0.0
1.0
2.0
$h_1' = \mathrm{a}[\psi_{10}+\psi_{11}h_1+\psi_{12}h_2+\psi_{13}h_3]$
$h_2' = \mathrm{a}[\psi_{20}+\psi_{21}h_1+\psi_{22}h_2+\psi_{23}h_3]$
$h_3' = \mathrm{a}[\psi_{30}+\psi_{31}h_1+\psi_{32}h_2+\psi_{33}h_3]$
$\phi_1' h_1'$
$\phi_2' h_2'$
$\phi_3' h_3'$


g)
h)
i)
j)
Output
Input, x
Output, y'
1.0
0.0
-1.0
0.0
1.0
2.0
$\phi_1' h_1'$
$\phi_2' h_2'$
$\phi_3' h_3'$
$\phi_0' + \phi_1' h_1' + \phi_2' h_2' + \phi_3' h_3'$

Hyperparameters

- K layers: network depth
- D_k hidden units / layer: network width

- We chose these hyperparameters before training.
- And we usually search for the best values by retraining with different hyperparameters.

Notation change #1

$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x]
\end{aligned}
\quad \longrightarrow \quad
\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right]
$$

$$
\begin{aligned}
h'_1 &= \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3] \\
h'_2 &= \mathrm{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3] \\
h'_3 &= \mathrm{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]
\end{aligned}
$$

$$
y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3
$$

Notation change #1

$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x]
\end{aligned}
\quad \longrightarrow \quad
\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right]
$$

$$
\begin{aligned}
h'_1 &= \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3] \\
h'_2 &= \mathrm{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3] \\
h'_3 &= \mathrm{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]
\end{aligned}
\quad \longrightarrow \quad
\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right]
$$

Notation change #1

$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x]
\end{aligned}
\longrightarrow
\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right]
$$

$$
\begin{aligned}
h'_1 &= \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3] \\
h'_2 &= \mathrm{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3] \\
h'_3 &= \mathrm{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]
\end{aligned}
\longrightarrow
\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{32} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right]
$$

$$
y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3
\longrightarrow
y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix}
$$

Notation change #2

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right] \longrightarrow \mathbf{h} = \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta} x\right]$$

$$\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{32} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right] \longrightarrow \mathbf{h}' = \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}\right]$$

$$y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} \longrightarrow y = \boldsymbol{\phi}'_0 + \boldsymbol{\phi}'\mathbf{h}'$$

Notation change #3 (multivariate inputs)

$$\mathbf{h} = \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta} x\right] \longrightarrow \mathbf{h}_1 = \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0 \mathbf{x}]$$

$$\mathbf{h}' = \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}\right] \longrightarrow \mathbf{h}_2 = \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1 \mathbf{h}_1]$$

$$y = \phi_0' + \boldsymbol{\phi}'\mathbf{h}' \longrightarrow \mathbf{y} = \boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2$$

Notation change #3

Bias vector

Weight matrix

$$\mathbf{h} = \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta} x\right] \longrightarrow \mathbf{h}_1 = \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0 \mathbf{x}]$$

$$\mathbf{h}' = \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}\right] \longrightarrow \mathbf{h}_2 = \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1 \mathbf{h}_1]$$

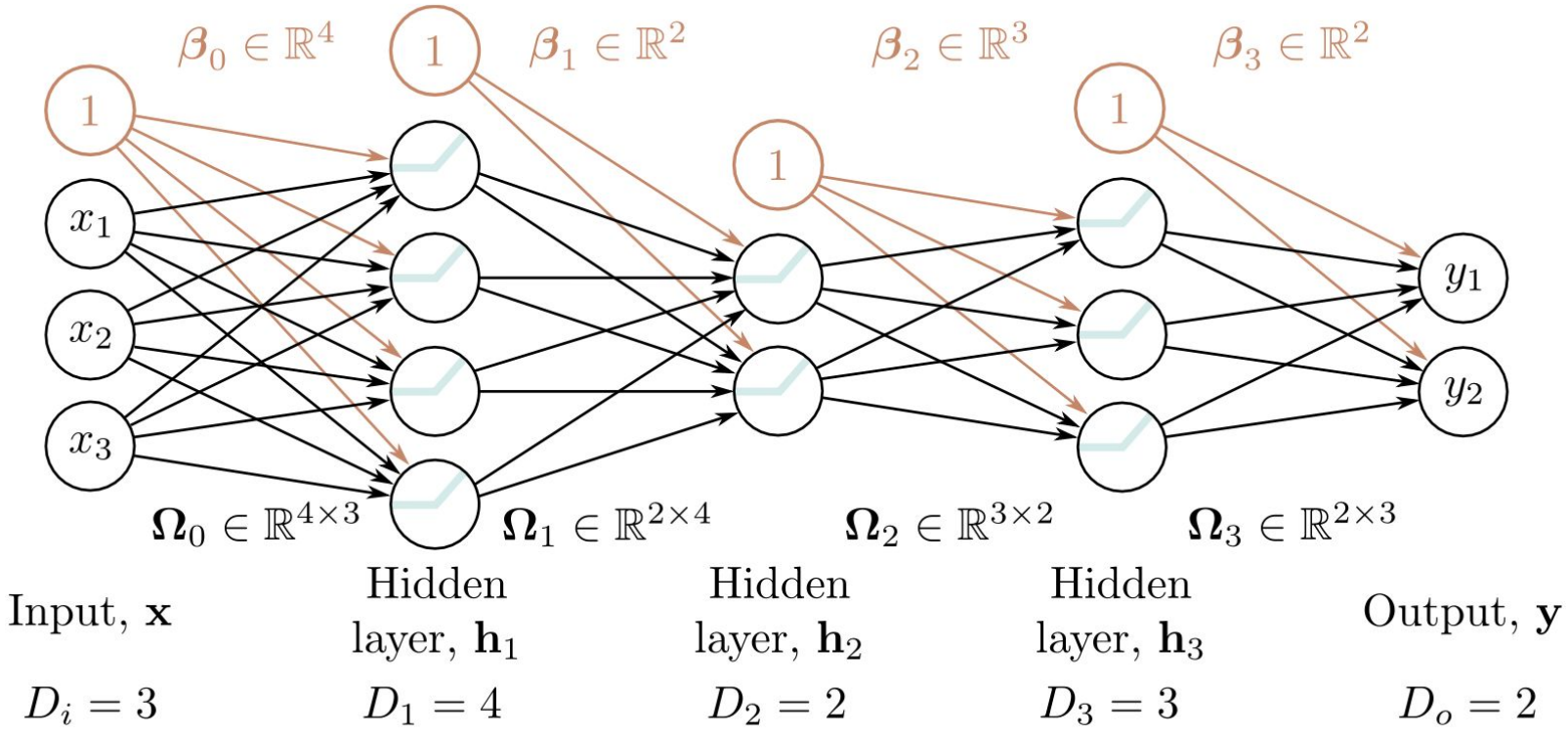
$$y = \phi_0' + \boldsymbol{\phi}'\mathbf{h}' \longrightarrow \mathbf{y} = \boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2$$

General equation for deep networks

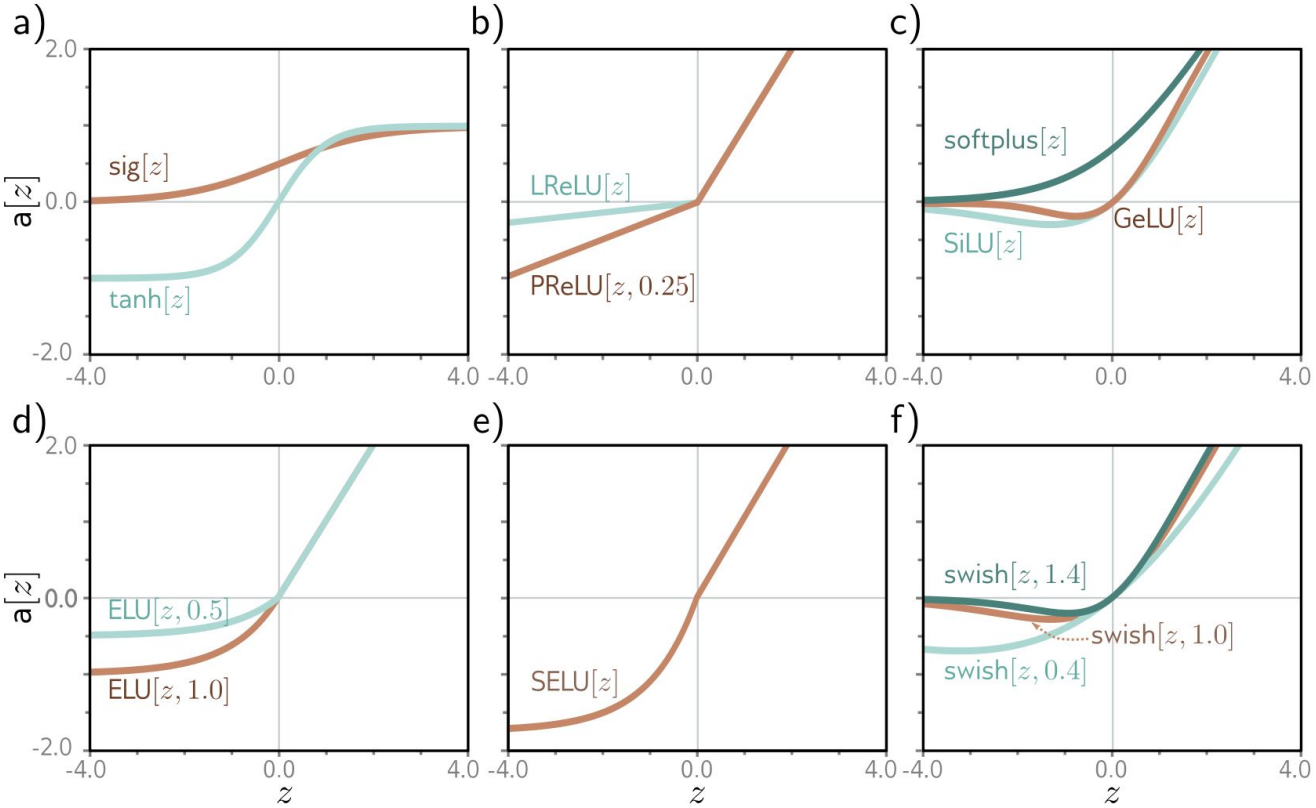
$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}] \\
\mathbf{h}_2 &= \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{h}_1] \\
\mathbf{h}_3 &= \mathbf{a}[\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{h}_2] \\
&\vdots \\
\mathbf{h}_K &= \mathbf{a}[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1}\mathbf{h}_{K-1}] \\
\mathbf{y} &= \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K\mathbf{h}_K,
\end{aligned}
$$

$$
\mathbf{y} = \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K\mathbf{a}\left[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1}\mathbf{a}\left[\ldots\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{a}\left[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{a}\left[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}\right]\right]\ldots\right]\right]
$$

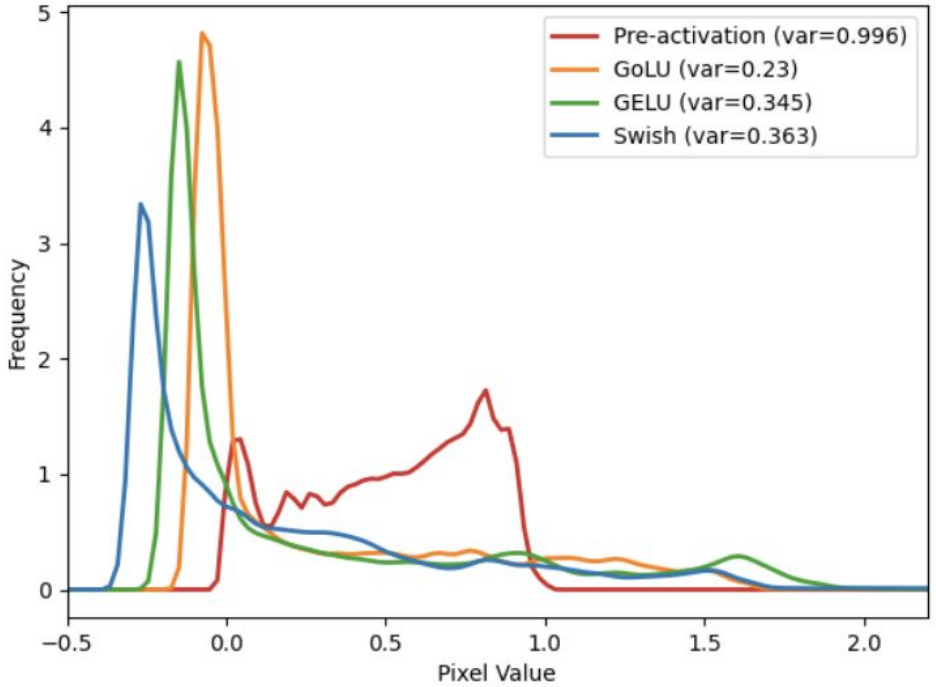
...also known as an MLP (multi-layer perceptron)



Activation functions



Activation functions

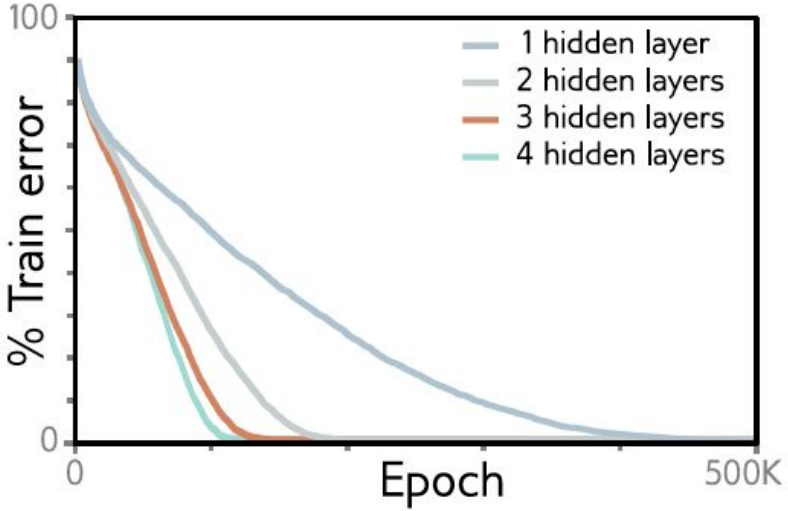


https://arxiv.org/abs/2502.03654

Depth efficiency of neural networks

Shallow vs. deep networks

Figure 20.2 MNIST-1D training. Four fully connected networks were fit to 4000 MNIST-1D examples with random labels using full batch gradient descent, He initialization, no momentum or regularization, and learning rate 0.0025. Models with 1,2,3,4 layers had 298, 100, 75, and 63 hidden units per layer and 15208, 15210, 15235, and 15139 parameters, respectively. All models train successfully, but deeper models require fewer epochs.



Shallow vs. deep networks

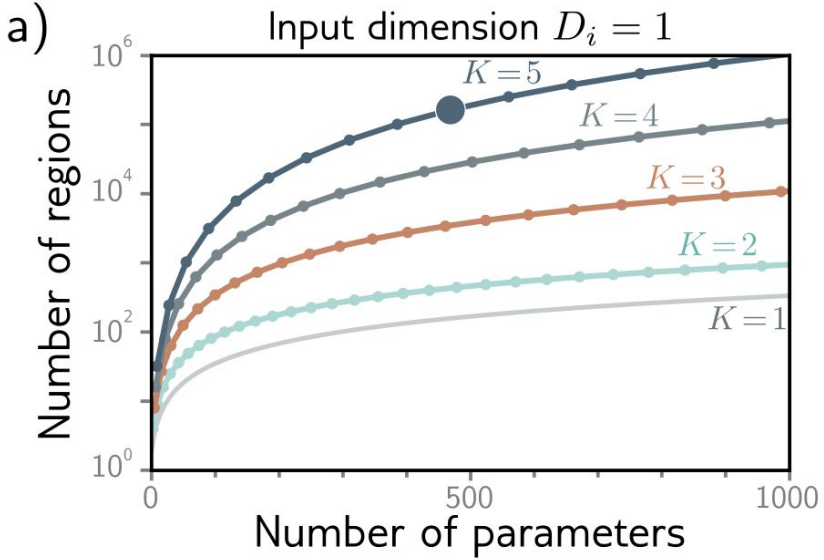
The best results are obtained by deep networks with many layers:

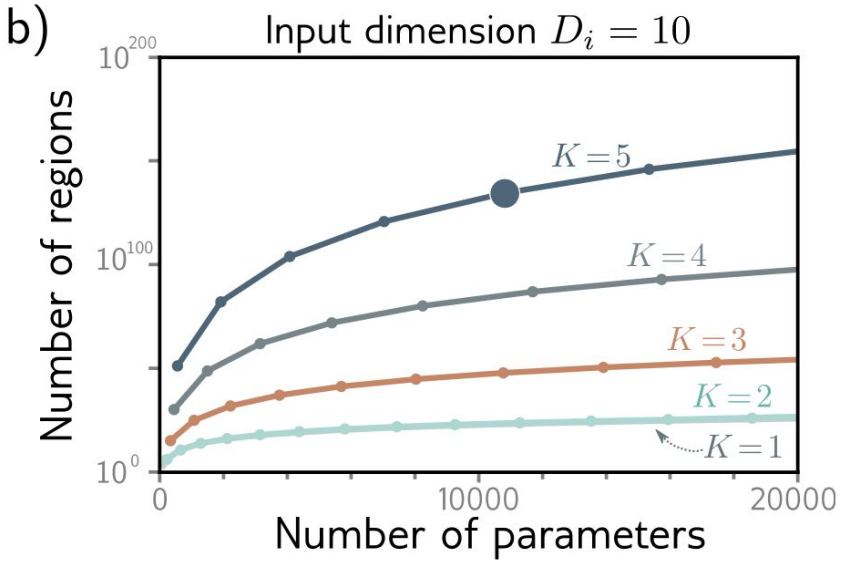
- 50-1000 layers for most applications
- Over ~10-15 layers additional tricks are required (normalisation, residual connections)
- Best results in:
  - Computer vision
  - Natural language processing
  - Graph neural networks
  - Generative models
  - Reinforcement learning

1. Function approximation capacity

- They both obey the universal approximation theorem.
- Argument: one layer is enough!

2. Number of linear regions per parameter





5 layers (up to)
10 hidden units per layer
471 parameters
161,501 linear regions

5 layers (up to)
50 hidden units per layer
10,801 parameters
10^{134} linear regions

What’s next?

- We have defined families of very flexible networks that map multiple inputs to multiple outputs
- Now we need to train them
  - How to choose loss functions
  - How to find minima of the loss function
  - How to do this in particular for deep networks
- Then we need to test them

How to reach me:

florin.gogianu@gmail.com

Please send unstructured feedback, since this is a new version of the lecture!