Neural Networks

Florin Gogianu, Lucian Bușoniu

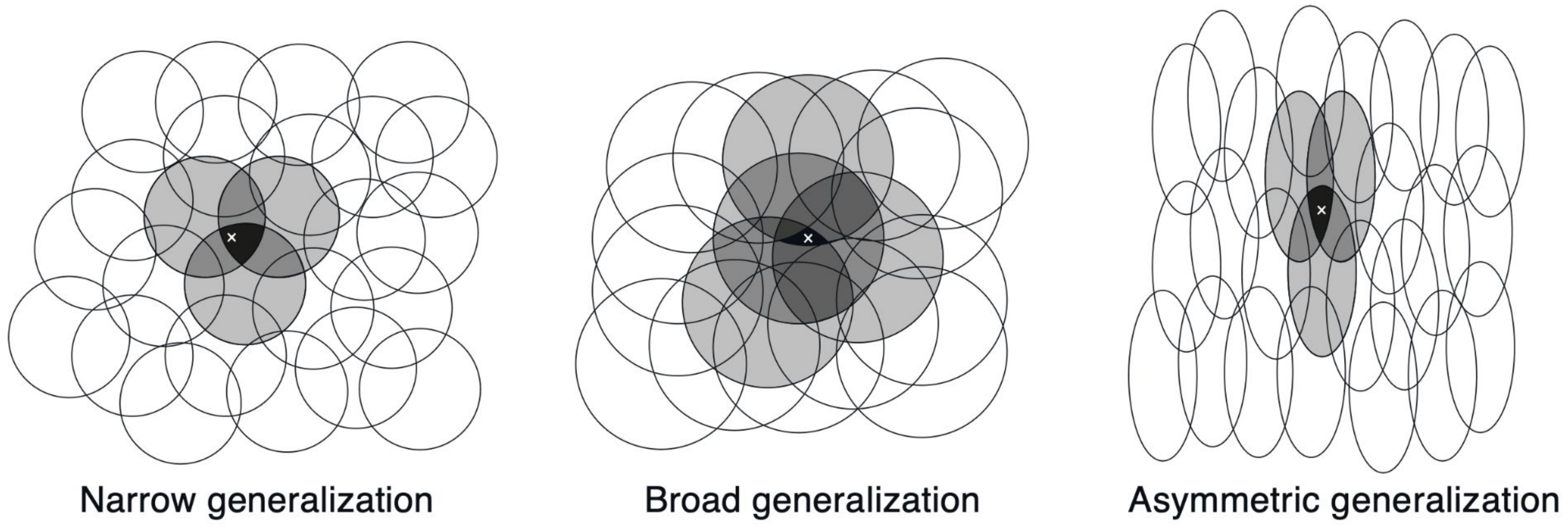
Technical University of Cluj-Napoca
Bitdefender

* slides, illustrations and ideas adapted from: prof. Simon Prince UDLB, Stanford CS231n, LeCun, Canziani, NYU Deep Learning
** this lecture can be considered a complementary view of the Neural Networks course taught by prof. Vlad Miclea.

Recap. Function approximation

1. What is function approximation (in control / RL)?
2. Why bother with function approximation?
3. What types of function approximation are out there?

Generalization with linear function approximation



Feature design for function approximation affects the generalization of the estimator (Sutton, 2019).

High-level goal

Represent and learn from data V(x), Q(x, u) and even h(x) using neural networks:

- $\tilde{V}(x; \boldsymbol{\theta})$
- $\tilde{Q}(x, \mathrm{u}; \boldsymbol{\theta})$
- $\tilde{h}(x; \boldsymbol{\theta})$

Be able to compute the terms required to update the weights $\boldsymbol{\theta}$:

$$\theta_{k+1} = \theta_k + \alpha_k \frac{\partial}{\partial \theta} \widehat{Q}(x_k, u_k; \theta_k) \cdot \left[r_{k+1} + \gamma \max_{u'} \widehat{Q}(x_{k+1}, u'; \theta_k) - \widehat{Q}(x_k, u_k; \theta_k) \right]$$

* Notation will be slightly different for this course, eg. tilde instead of hat.

Shallow neural networks

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

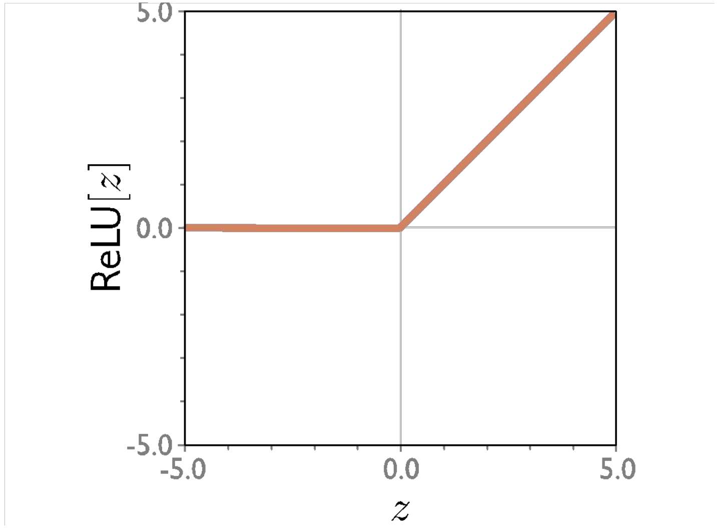
activation function

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

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

Rectified Linear Unit

(particular kind of activation function)



Shallow neural network

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

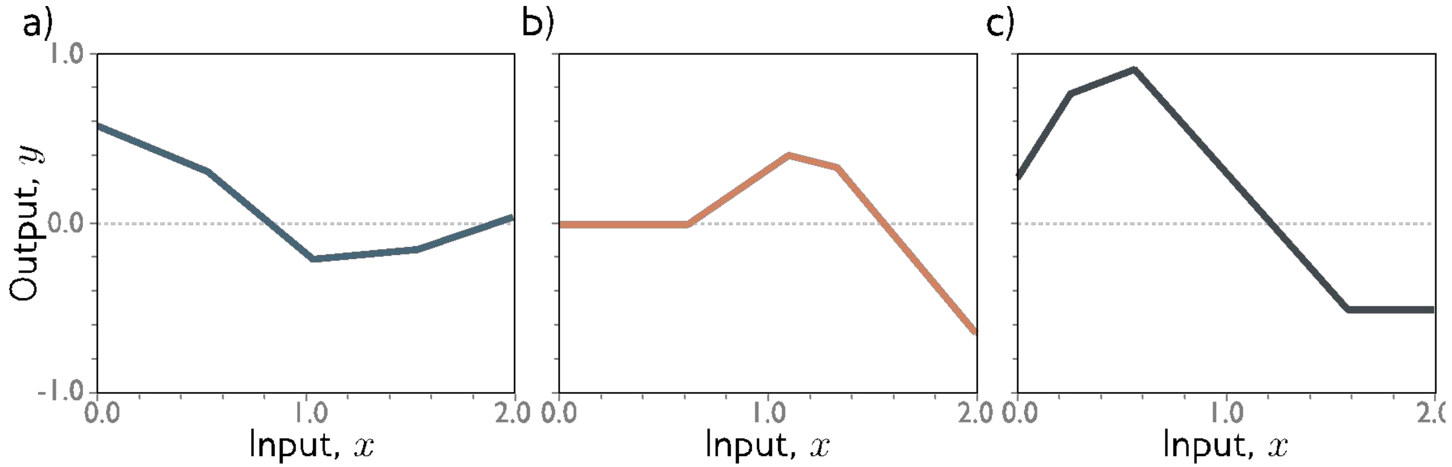
This model has 10 parameters:

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

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

Shallow neural network

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



Piecewise linear functions with three joints

Shallow neural network

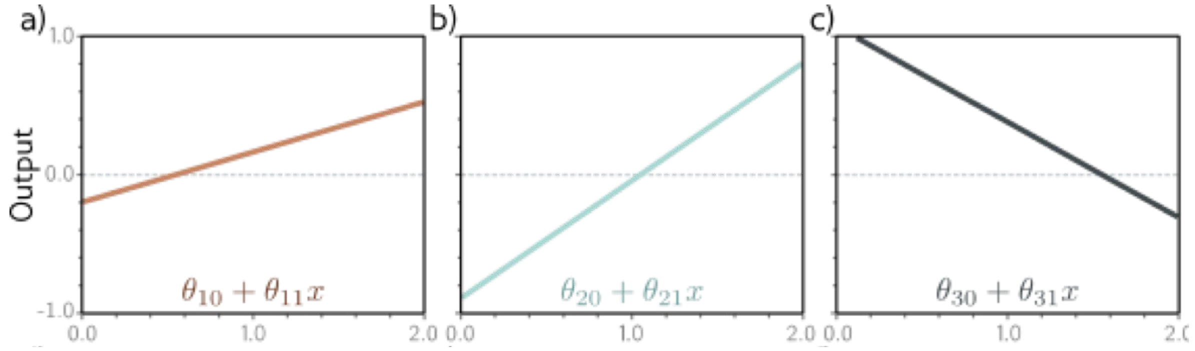
$$
\begin{aligned}
y &= \mathrm{f}[x, \boldsymbol{\phi}] \\
&= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x]
\end{aligned}
$$

Break it down into two parts:

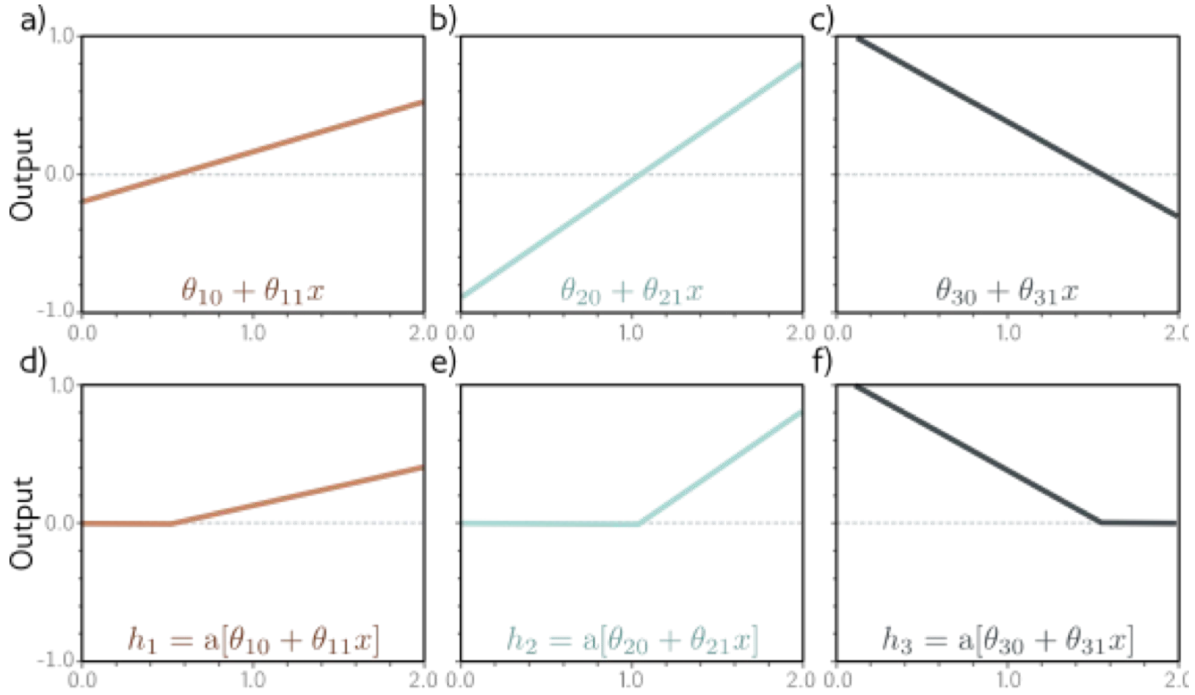
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

where:

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

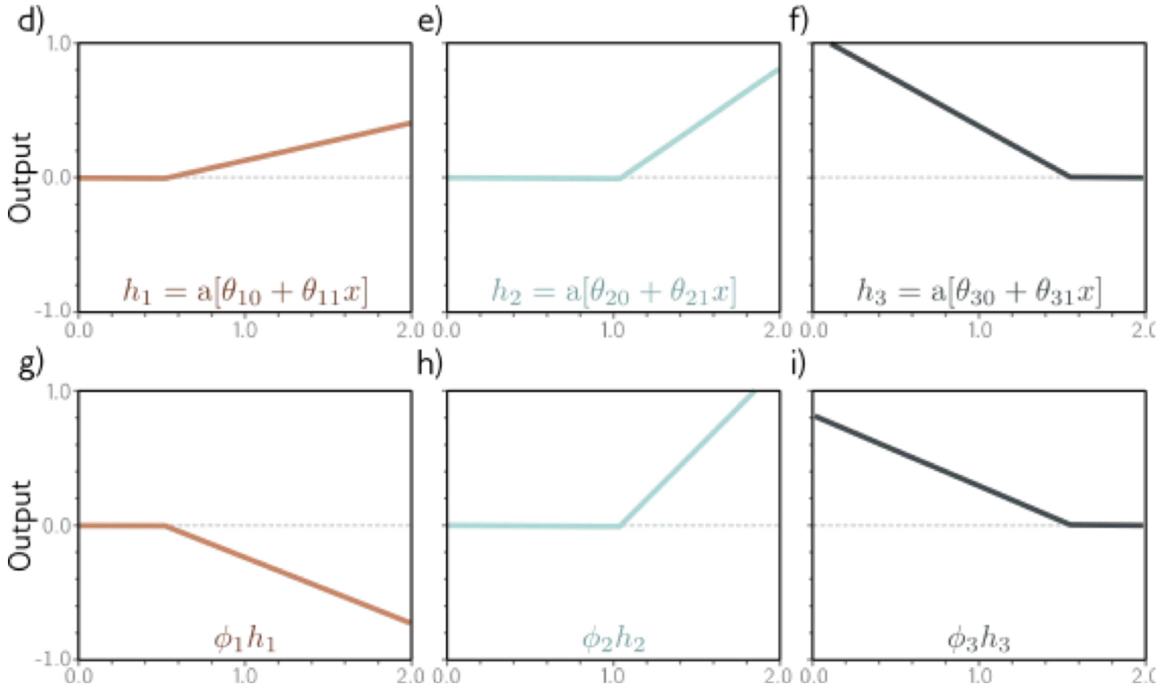


1. Compute three linear functions

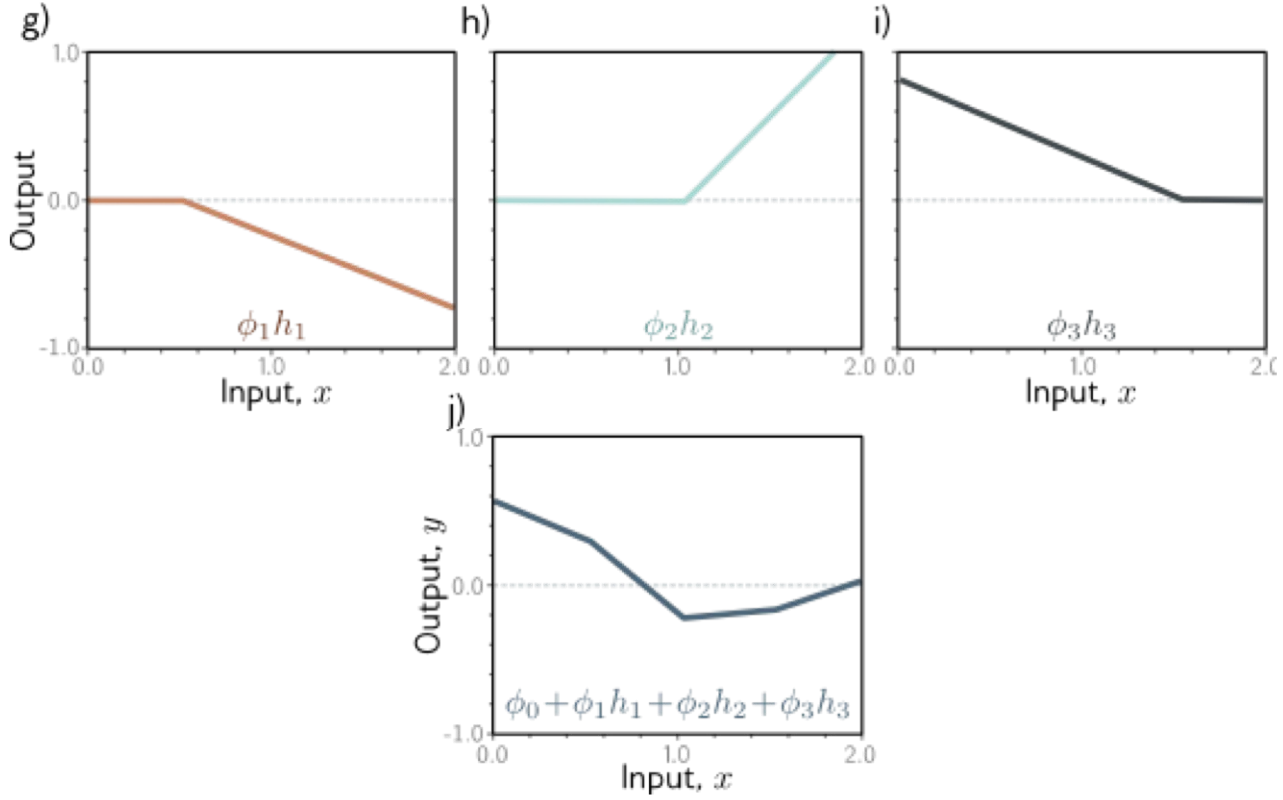


$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x],$$

2. Pass through ReLU activations (hidden units)



3. Weigh the hidden units

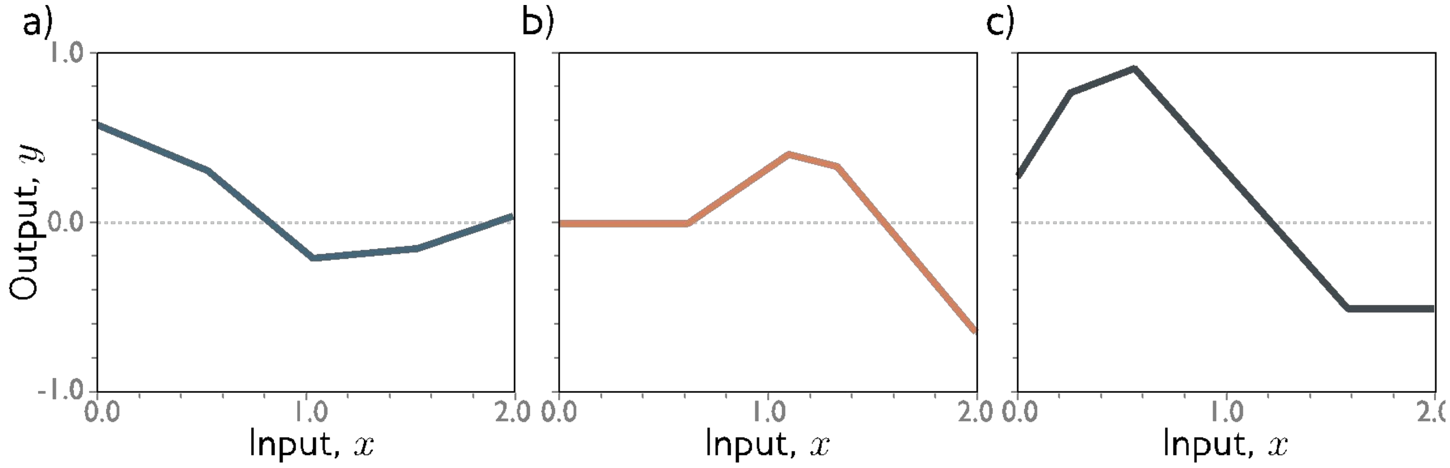


4. Sum the weighted activations

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

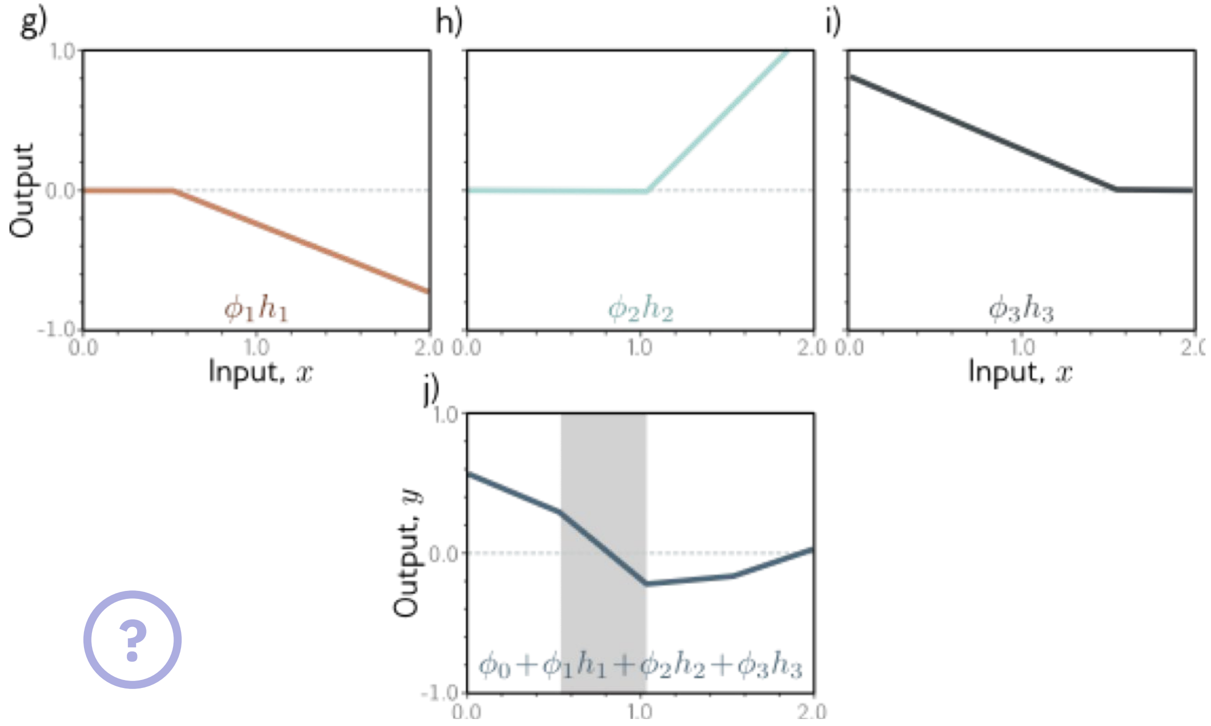
Shallow neural network

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



Piecewise linear functions

Activation pattern: which hidden units are activated



?

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

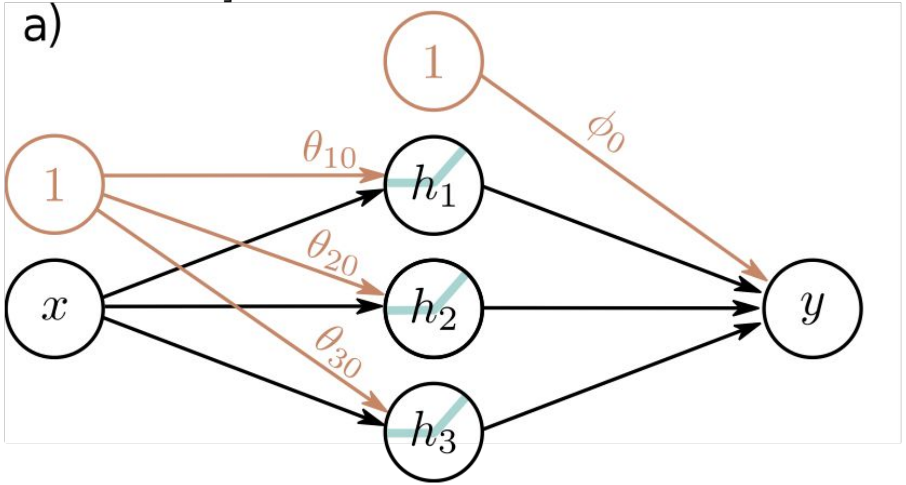
Depicting neural networks

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

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

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

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



Each parameter multiplies its sources and adds to its target

Universal approximation theorem

Arbitrary number of hidden units

From 3 hidden units:

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

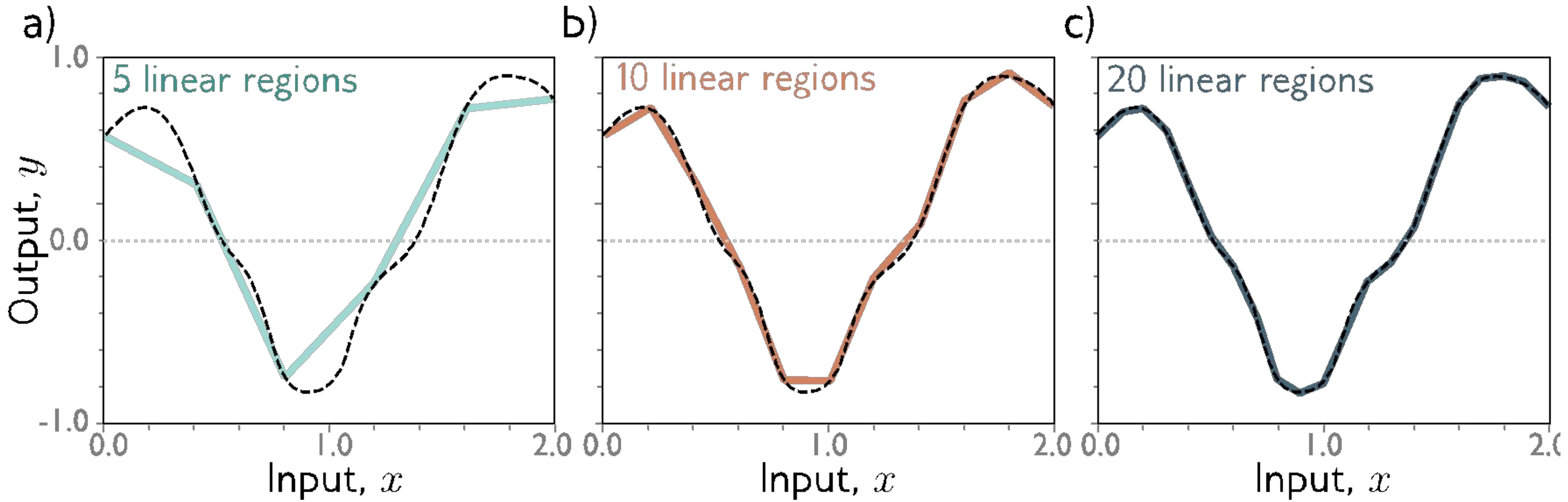
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

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”

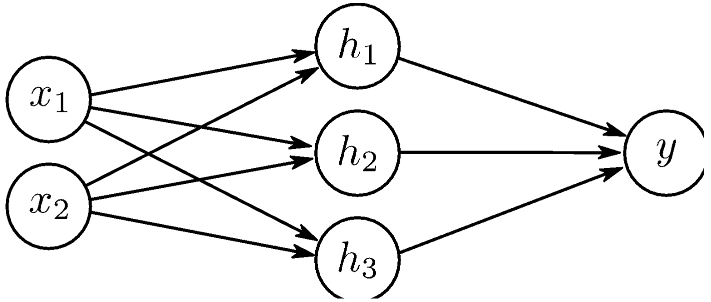
Hornik, 1990

* without guaranteeing a construction though!

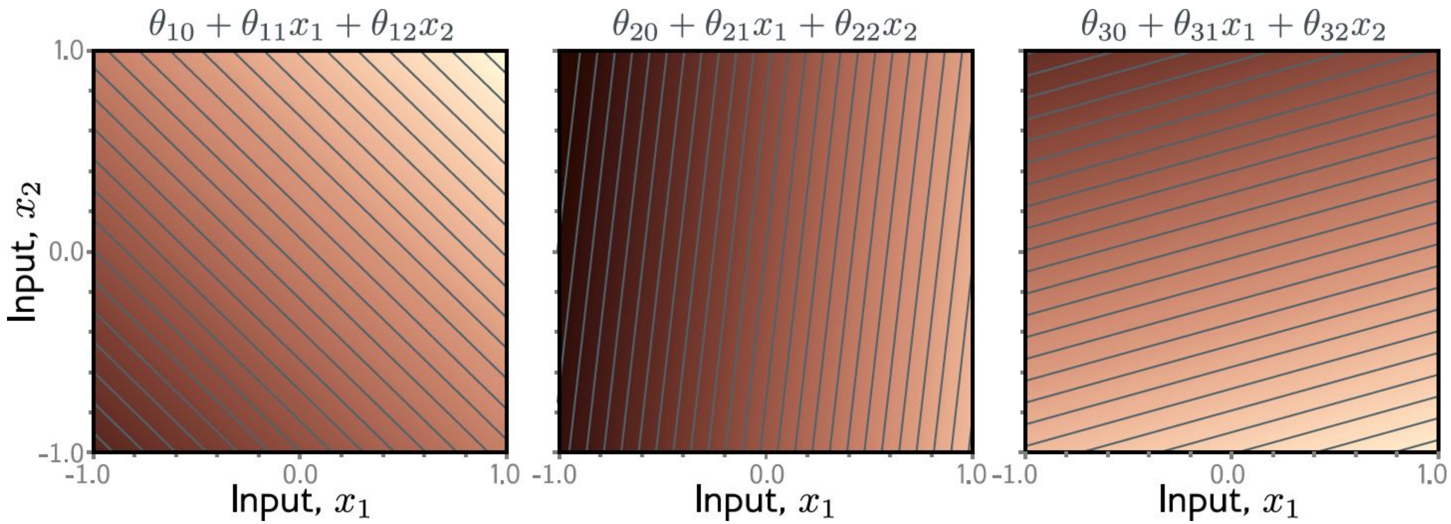
Multivariate *inputs*

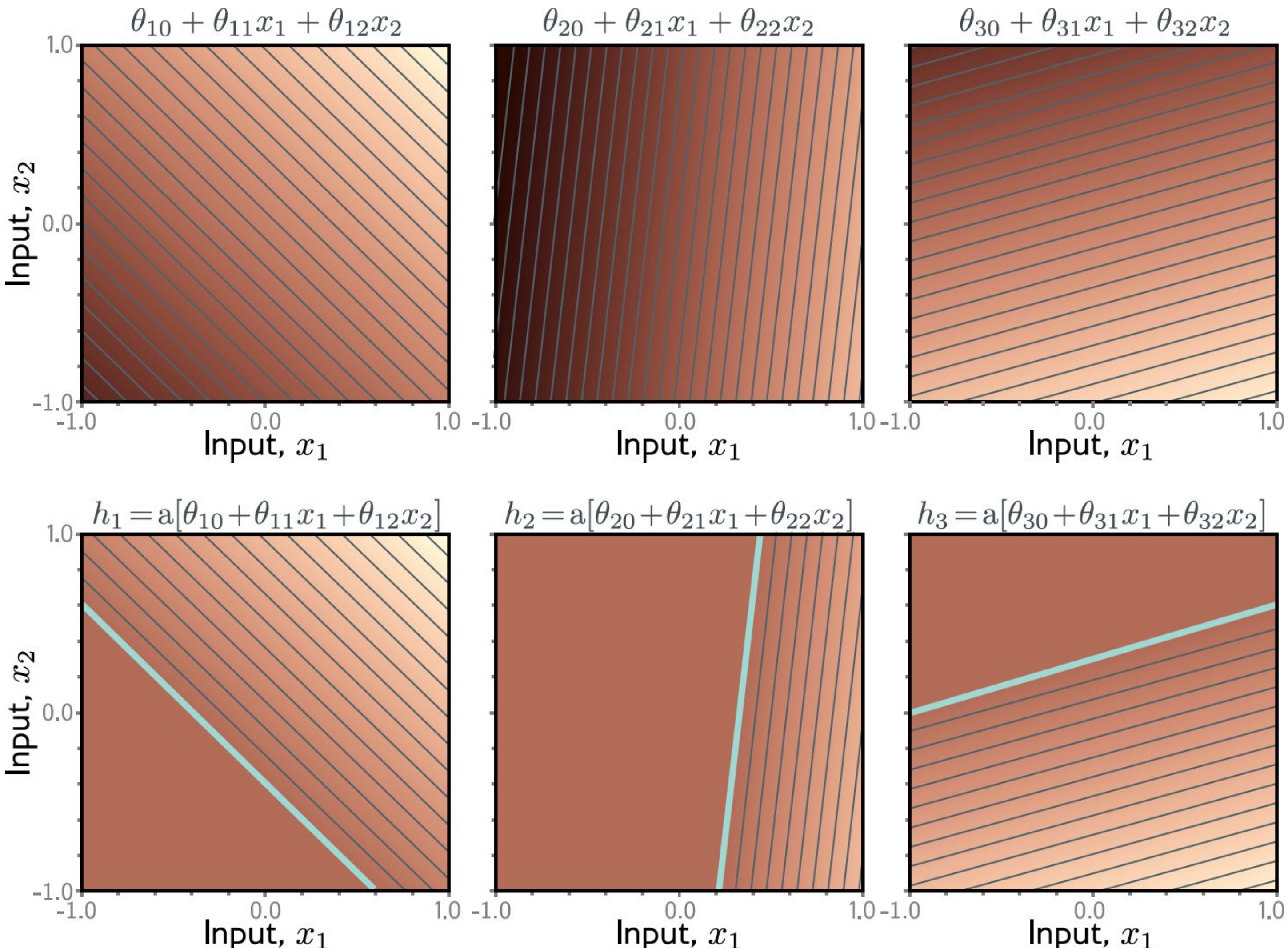
$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$$

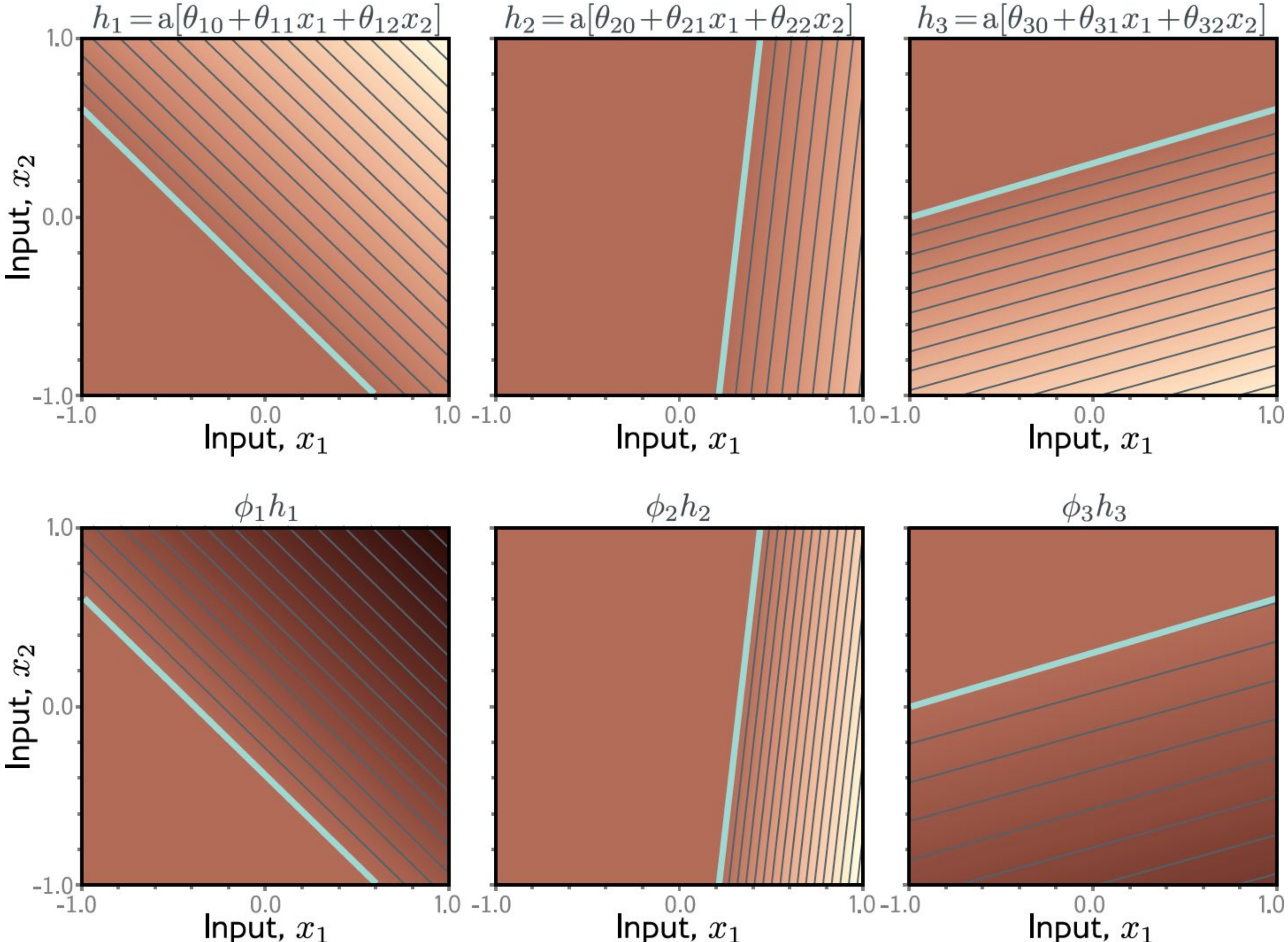
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$



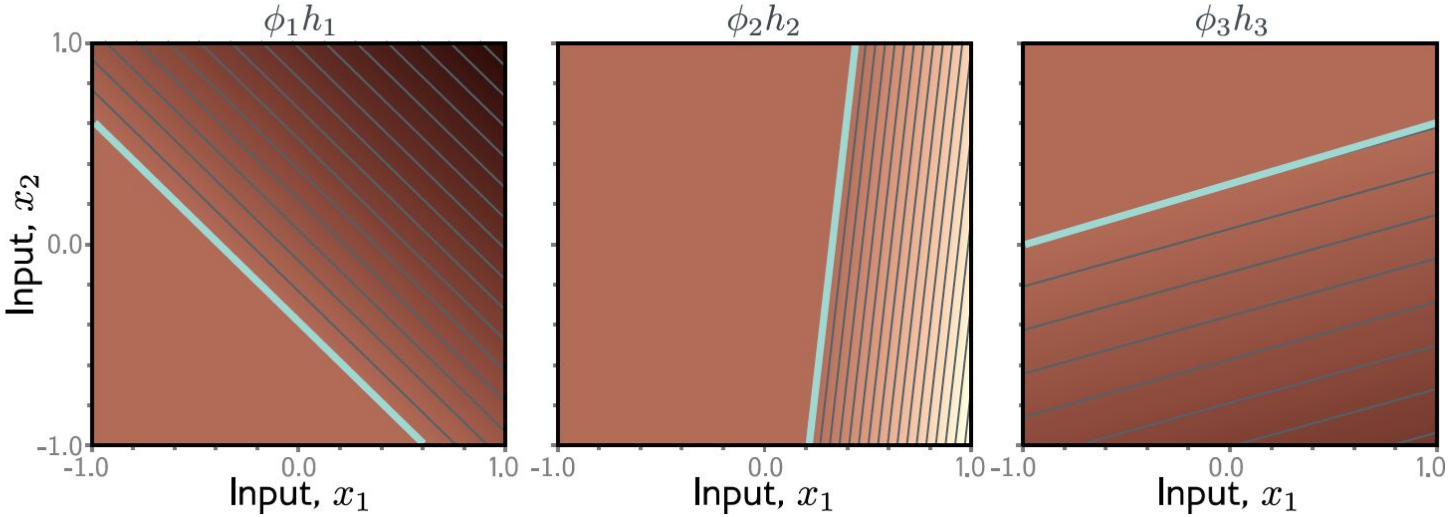
2 inputs, 3 hidden units, 1 output


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

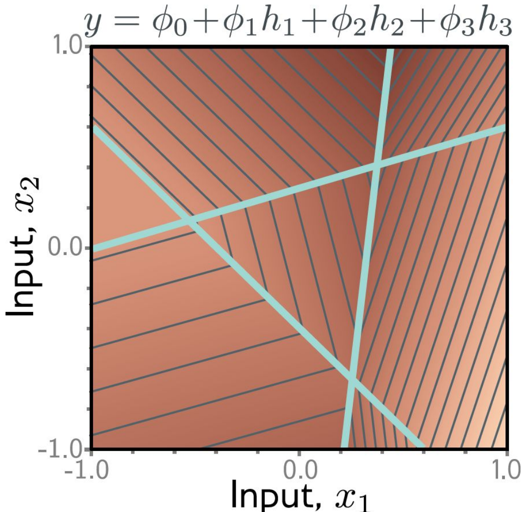

$\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2$
$\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2$
$\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2$
$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$
$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$
$h_3 = \mathrm{a}[\theta_{30}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



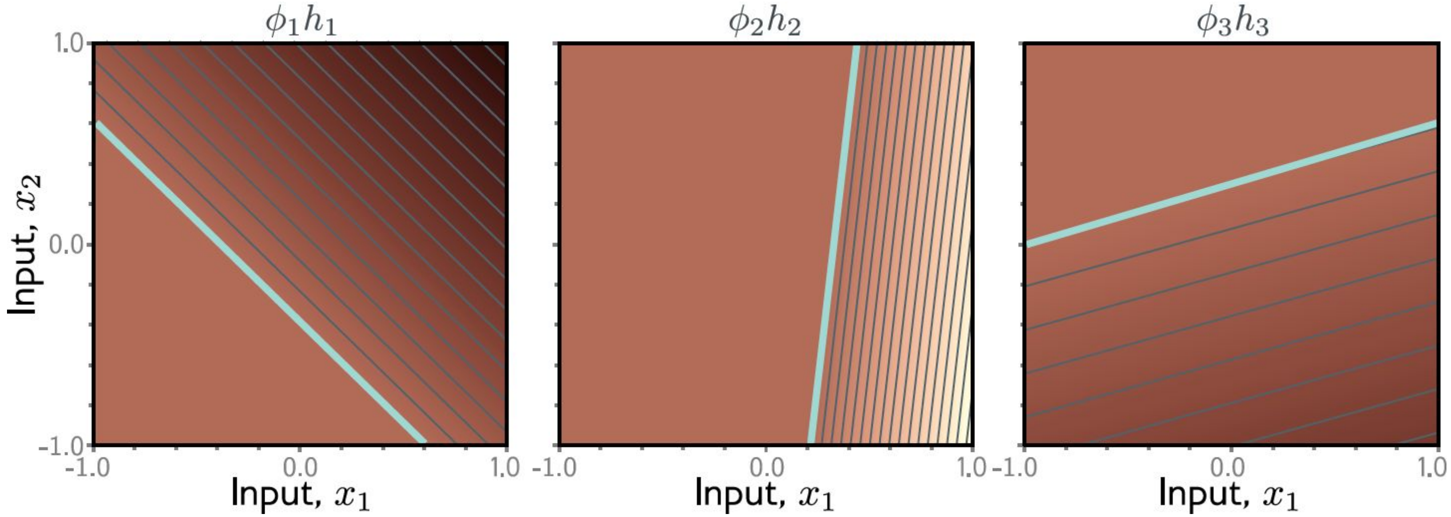
$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$
$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$
$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$
$\phi_1 h_1$
$\phi_2 h_2$
$\phi_3 h_3$
Input, x_1
Input, x_2
1.0
0.0
-1.0

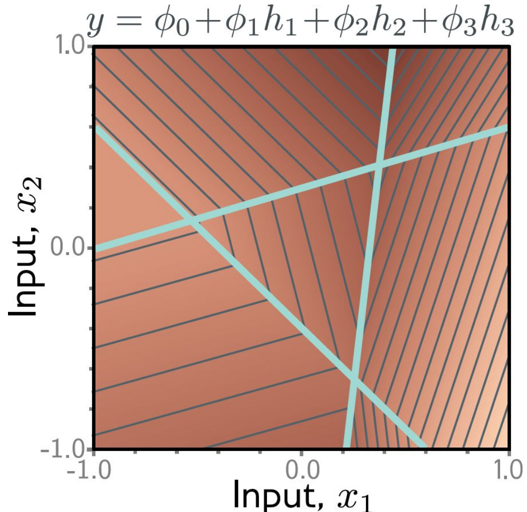


$\phi_1 h_1$
$\phi_2 h_2$
$\phi_3 h_3$
1.0
0.0
-1.0
-1.0
0.0
1.0
Input, x_1
Input, x_2



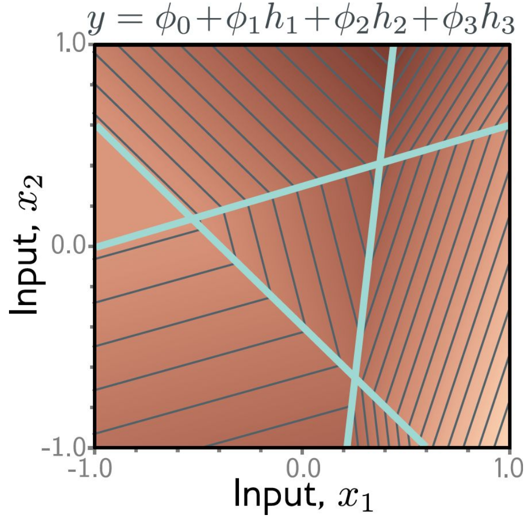
$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$
1.0
0.0
-1.0
-1.0
0.0
1.0
Input, x_1
Input, x_2

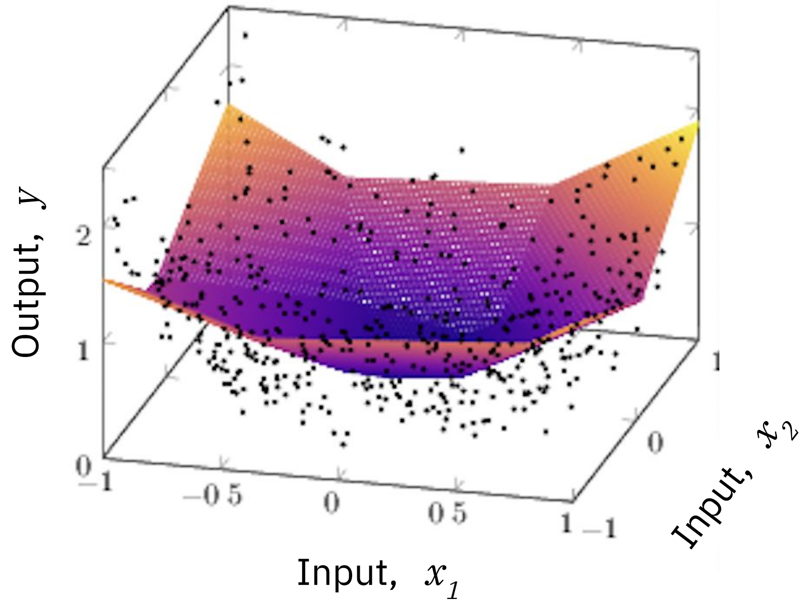




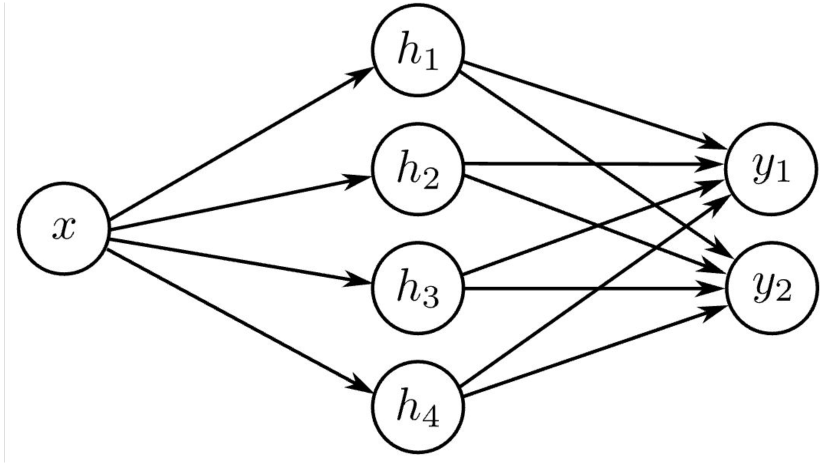
Convex polygons

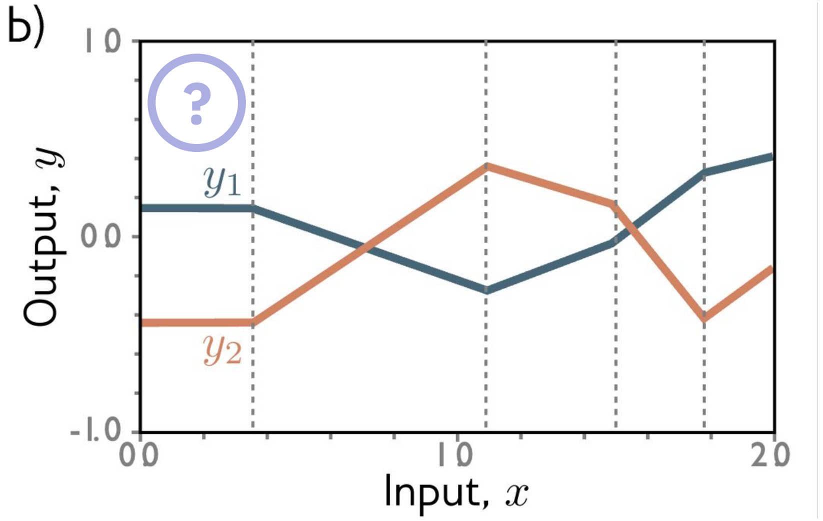
Function space / Data space





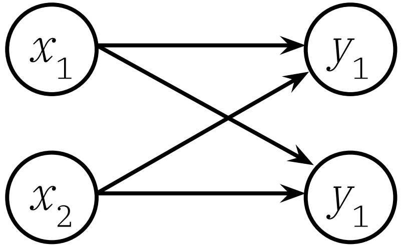
Multivariate *outputs*





Functional *vs.* Representational

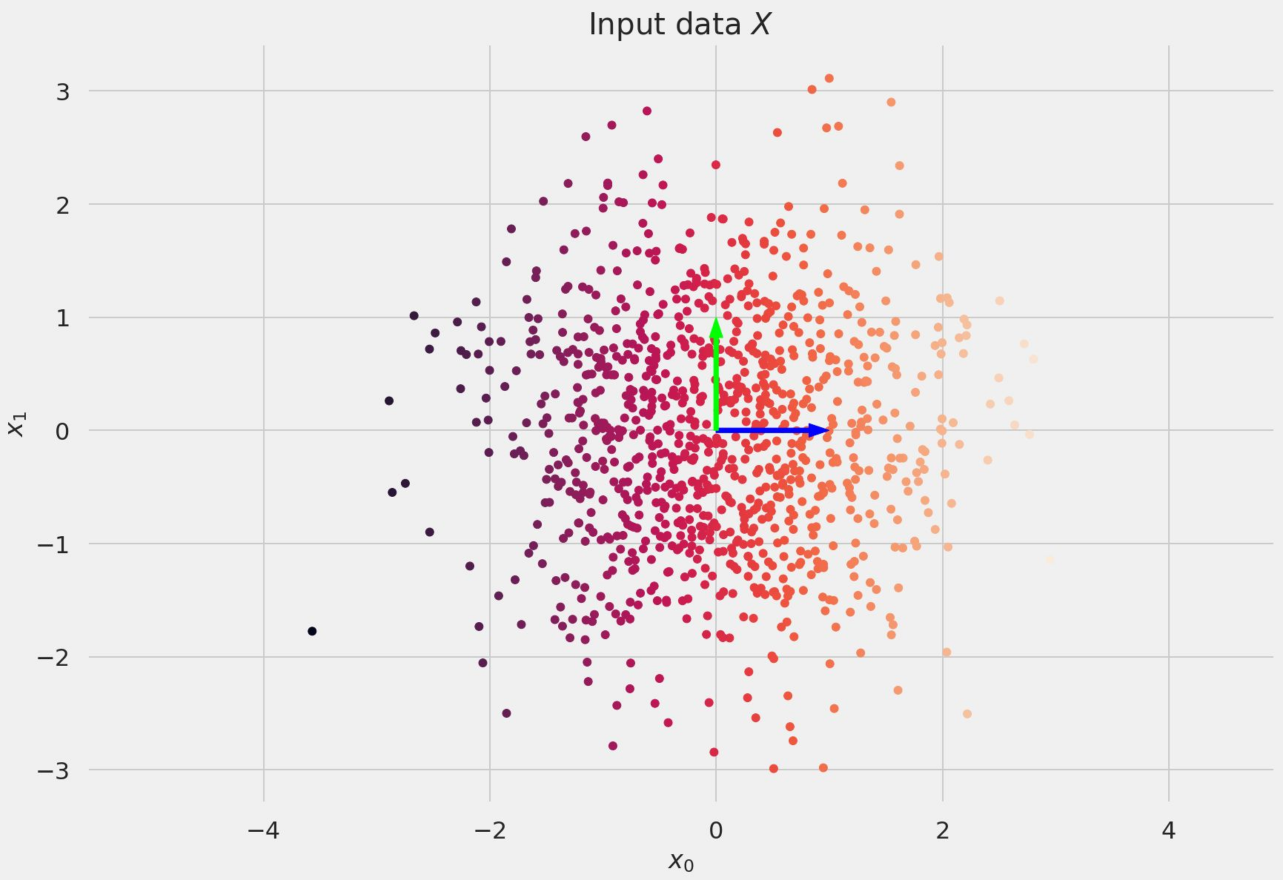
The *representation* view. Linear model



$\mathbf{y} = \mathbf{\Theta x}$, 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

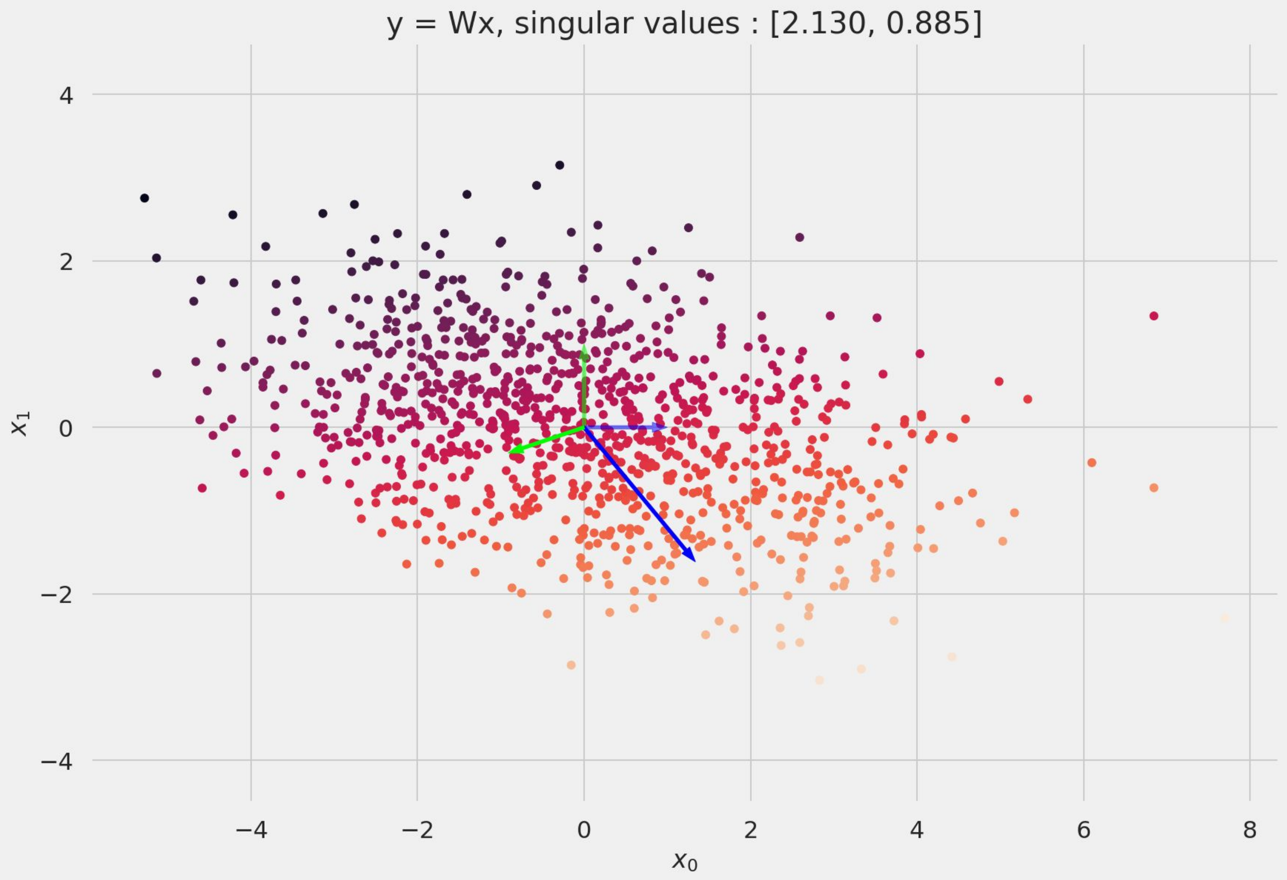
- Our model is: $\mathbf{y} = \boldsymbol{\Theta}\mathbf{x}$
- What is $\boldsymbol{\Theta}$ doing to $\mathbf{x}$?
- Let's perform *singular value decomposition** for some intuition:

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

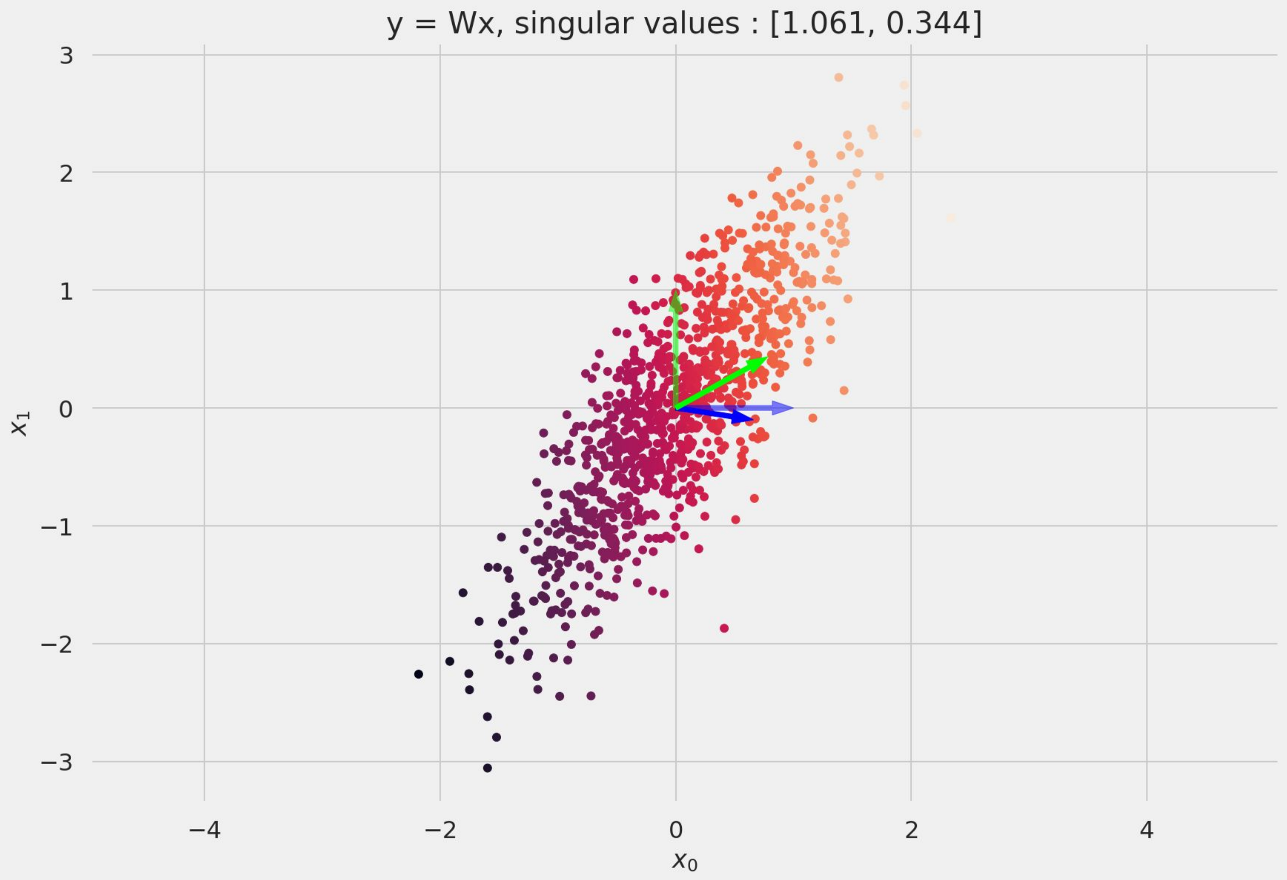
$$\begin{bmatrix} 0.19 & 1.84 \\ -0.97 & -0.93 \end{bmatrix} = \underbrace{\begin{bmatrix} 0.83 & -0.55 \\ -0.55 & -0.83 \end{bmatrix}}_{\text{rotation}} \times \underbrace{\begin{bmatrix} 2.17 & 0.00 \\ 0.00 & 0.74 \end{bmatrix}}_{\text{scale}} \times \underbrace{\begin{bmatrix} 0.32 & 0.94 \\ 0.94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

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

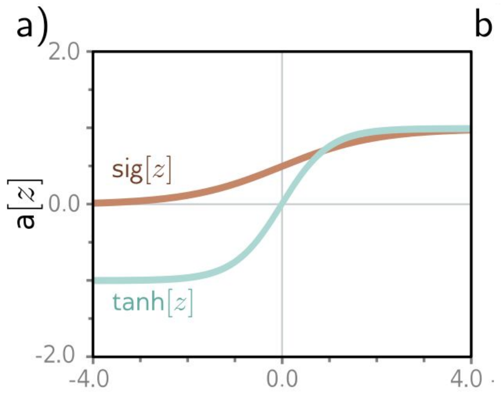
The *representation* view. Linear model



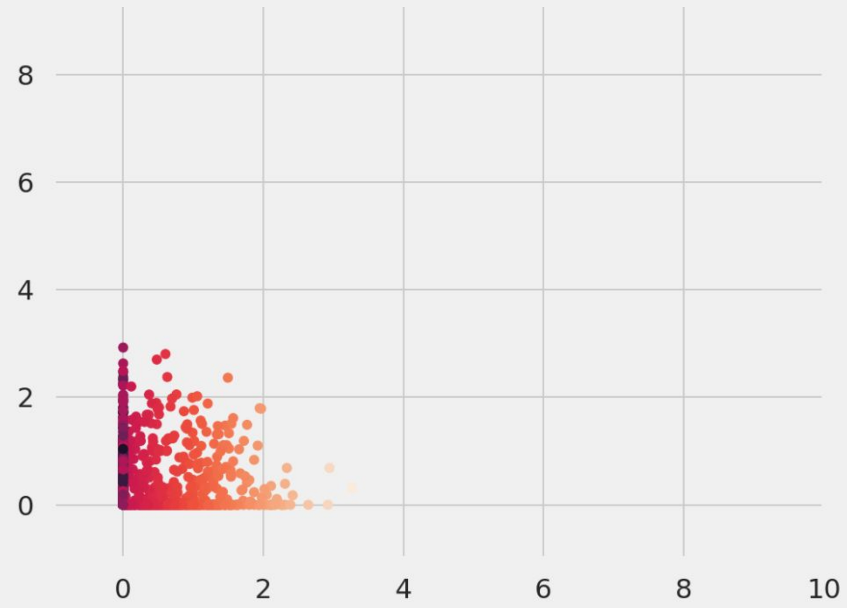
The *representation* view. Linear model

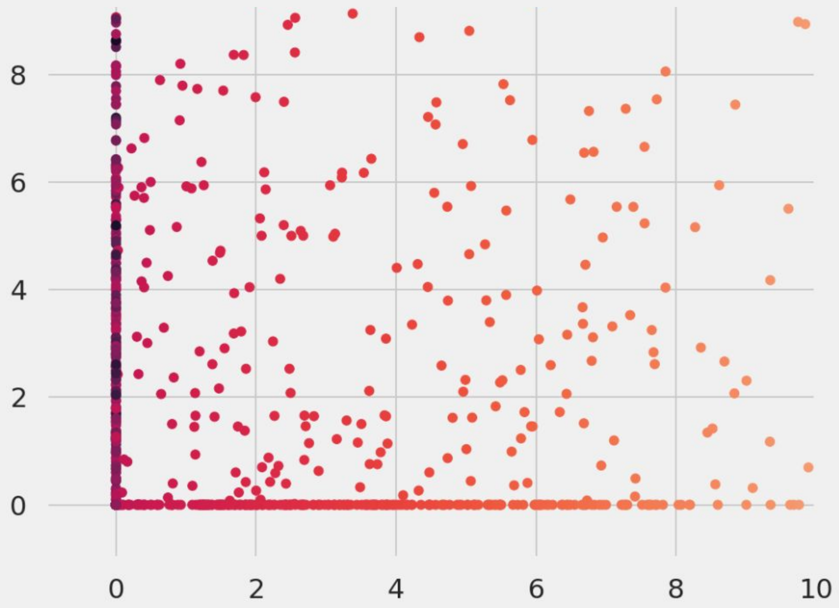


The *representation* view. Activation functions



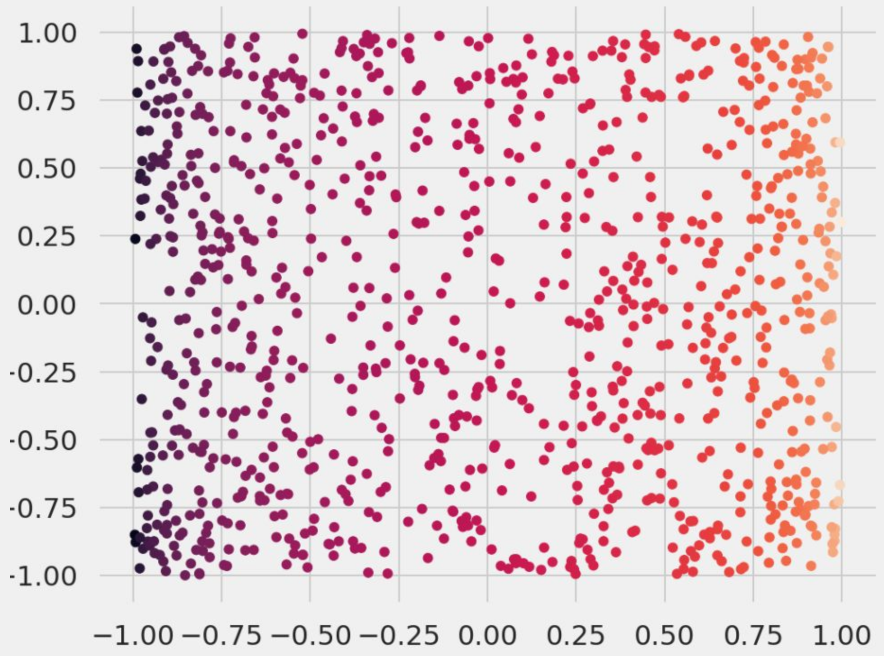
The representation view. *Non*-linear model | ReLU

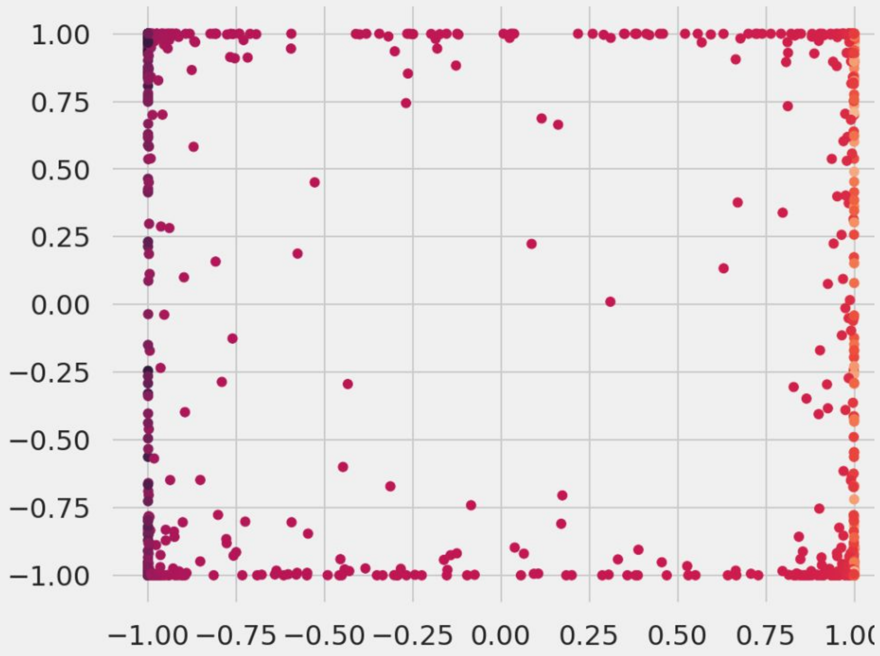




ReLU($\mathbf{Sx}$) activation function with two different scale factors S

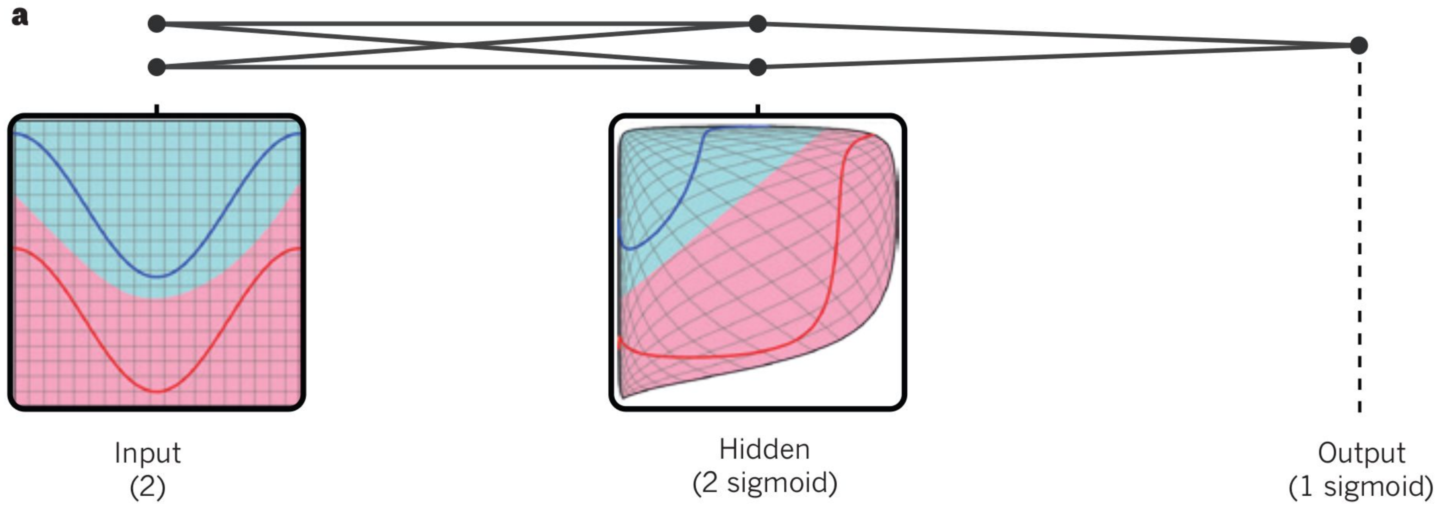
The representation view. *Non*-linear model | tanh



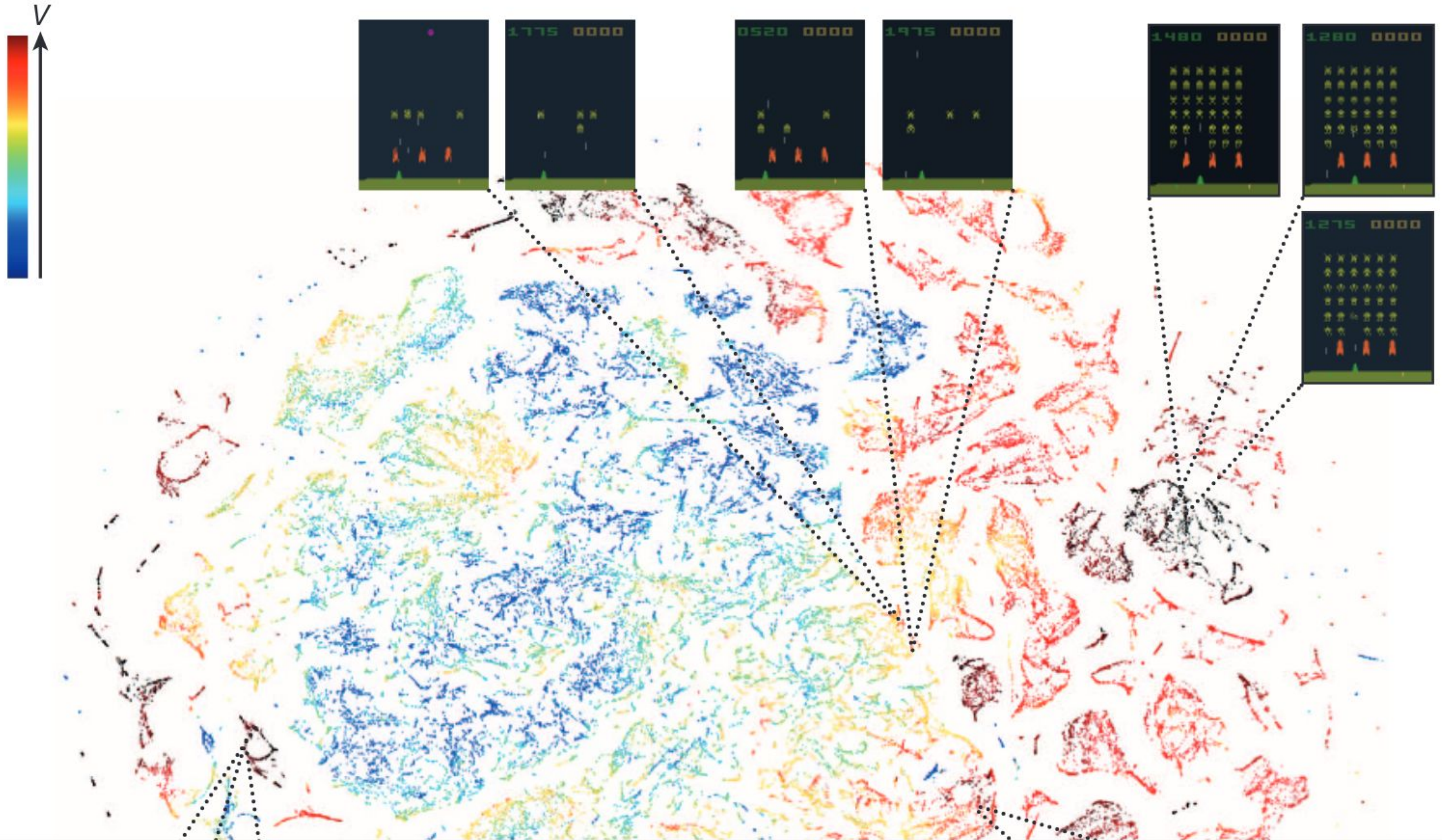


tanh($\mathbf{Sx}$) activation function with two different scale factors S

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.


V
1775 0000
0520 0000
1975 0000
1480 0000
1280 0000
1275 0000

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.

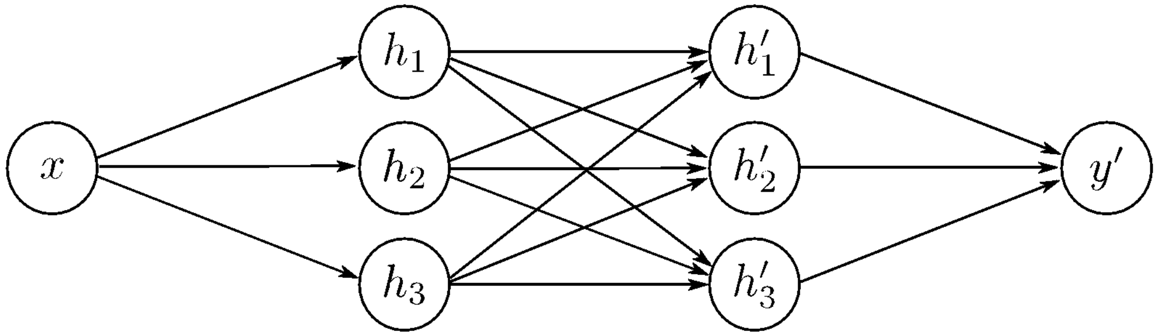
* Simon J.D. Prince, *Understanding Deep Learning*.

Two-layer networks

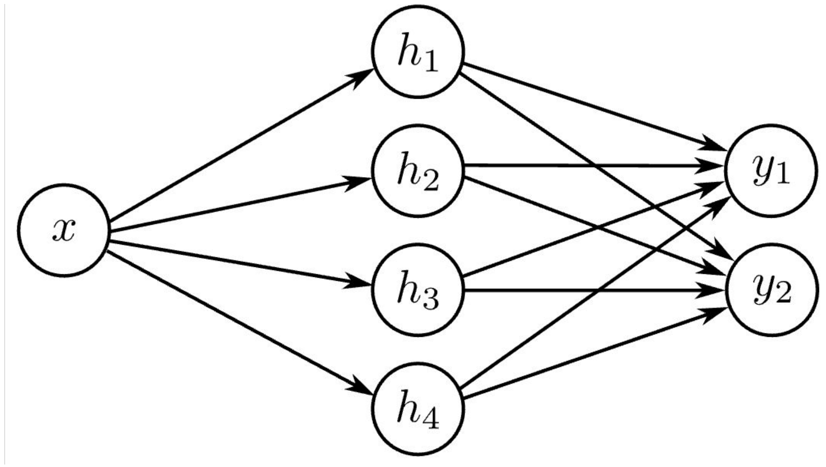
$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x]
\end{aligned}
$$

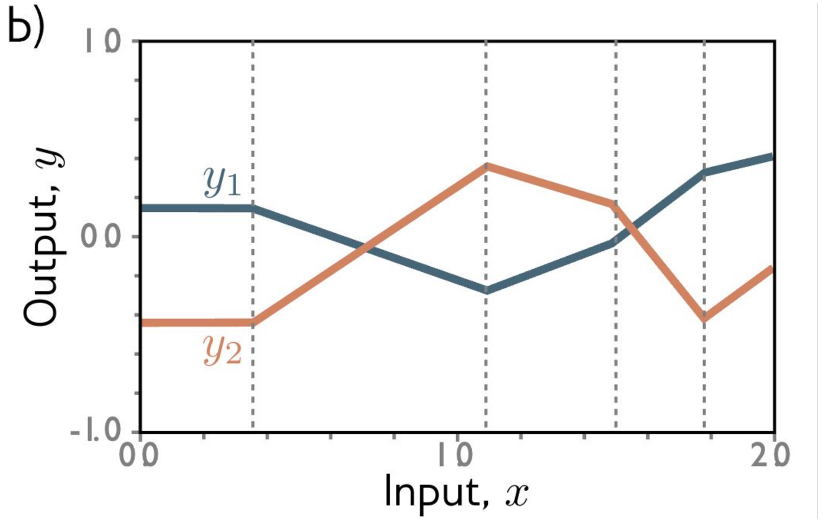
$$
\begin{aligned}
h'_1 &= \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3] \\
h'_2 &= \mathrm{a}[\psi_{20} + \psi_{21}h_2 + \psi_{22}h_2 + \psi_{23}h_3] \\
h'_3 &= \mathrm{a}[\psi_{30} + \psi_{31}h_2 + \psi_{32}h_2 + \psi_{33}h_3]
\end{aligned}
$$

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



Remember *shallow* network with two outputs

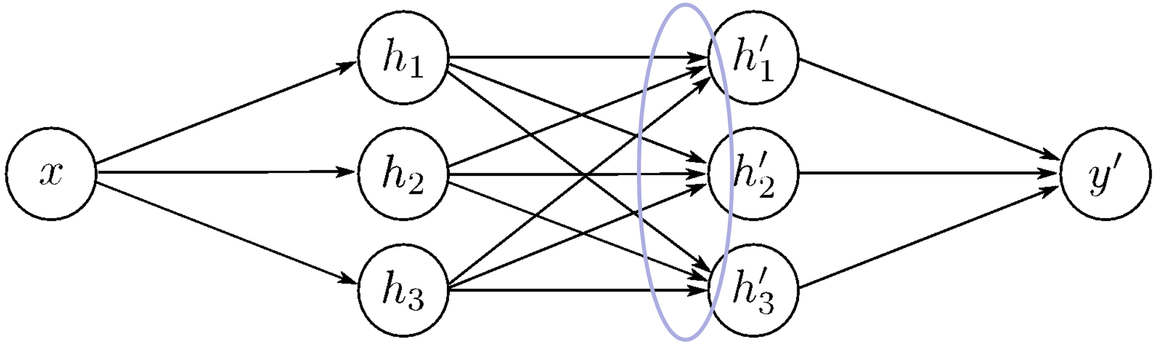


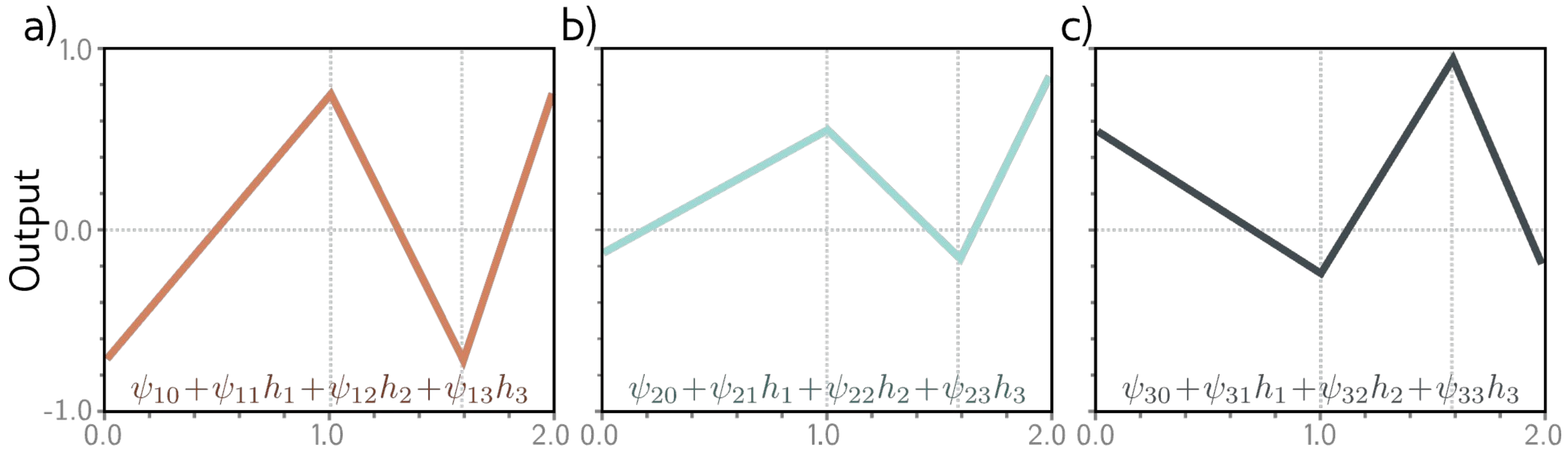


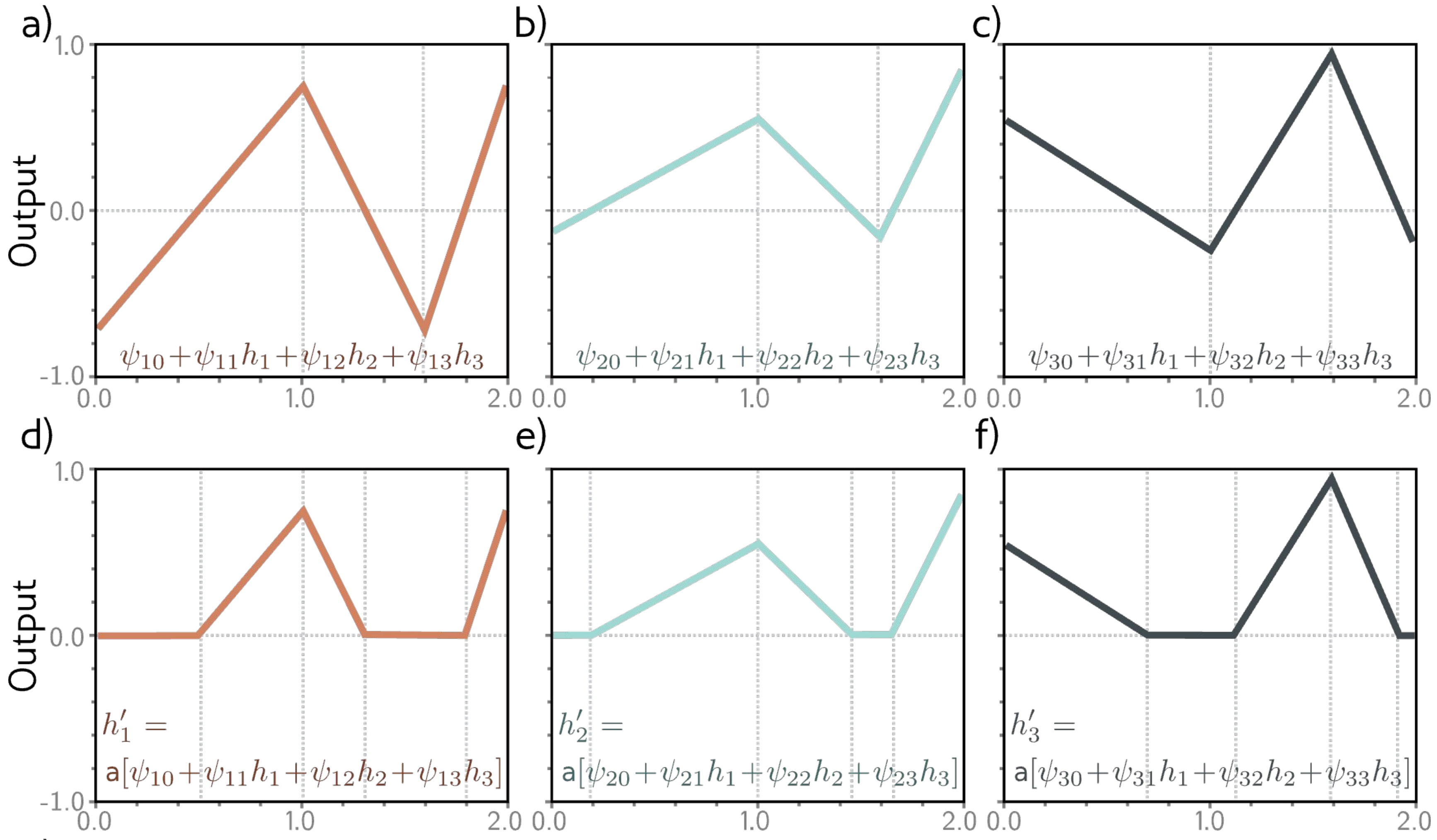
Deep networks are just function composition!

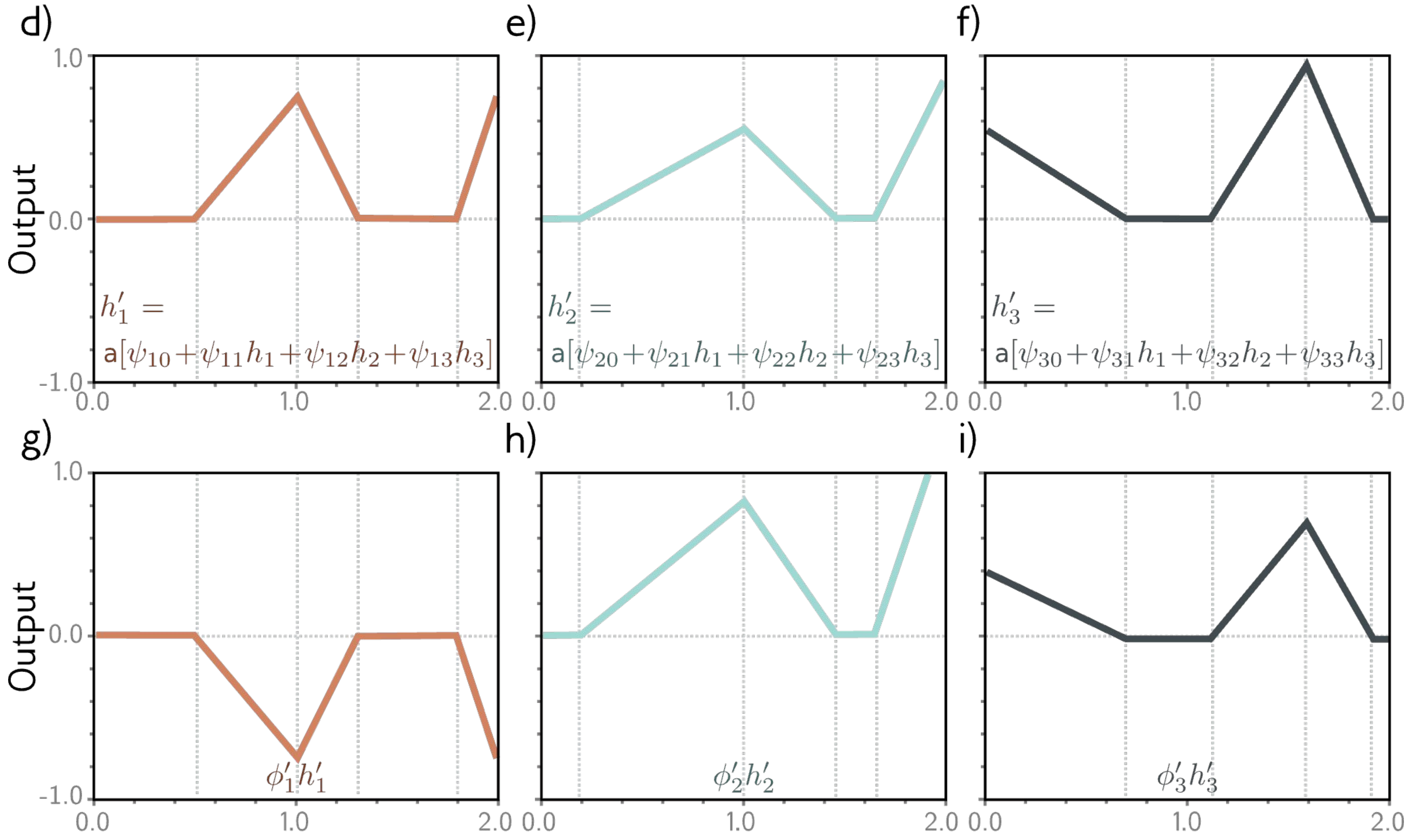
$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x]
\end{aligned}
\qquad
\begin{aligned}
h'_1 &= \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3] \\
h'_2 &= \mathrm{a}[\psi_{20}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

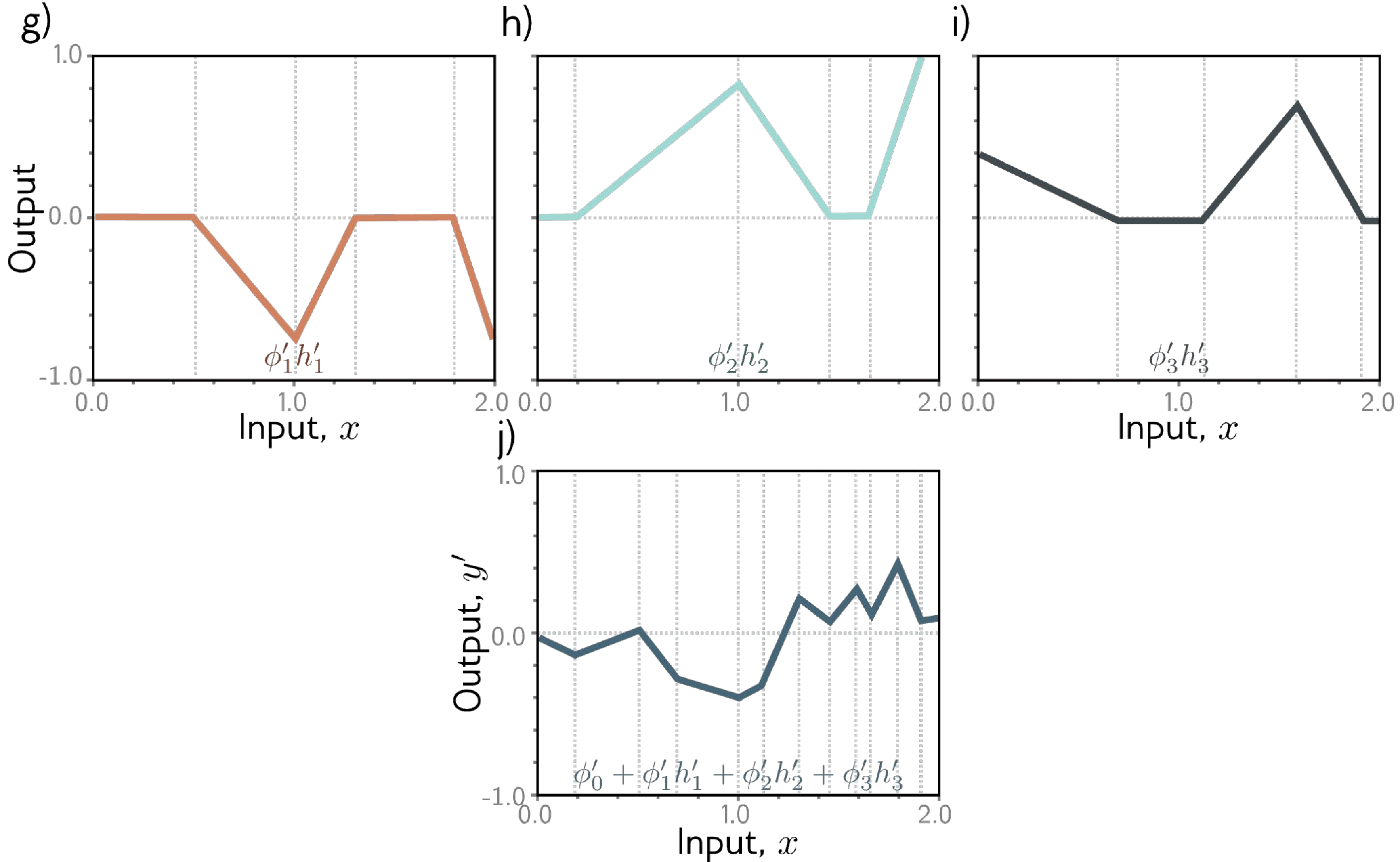
Consider the pre-activations at the second hidden units.
At this point, it's a one-layer network with three outputs.




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


d)
e)
f)
g)
h)
i)
Output
1.0
0.0
-1.0
2.0
$h'_1 =$
$\mathsf{a}[\psi_{10}+\psi_{11}h_1+\psi_{12}h_2+\psi_{13}h_3]$
$h'_2 =$
$\mathsf{a}[\psi_{20}+\psi_{21}h_1+\psi_{22}h_2+\psi_{23}h_3]$
$h'_3 =$
$\mathsf{a}[\psi_{30}+\psi_{31}h_1+\psi_{32}h_2+\psi_{33}h_3]$
$\phi'_1 h'_1$
$\phi'_2 h'_2$
$\phi'_3 h'_3$


g)
h)
i)
j)
1.0
0.0
-1.0
Output
Output, y'
0.0
1.0
2.0
Input, x
$\phi'_1 h'_1$
$\phi'_2 h'_2$
$\phi'_3 h'_3$
$\phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$

Hyperparameters

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

- We choose 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

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

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

Notation change #1

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

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

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

Notation change #2

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

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

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

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

$$\mathbf{h} = \mathbf{a}[\theta_0 + \boldsymbol{\theta} x] \longrightarrow \mathbf{h}_1 = \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0 \mathbf{x}]$$

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

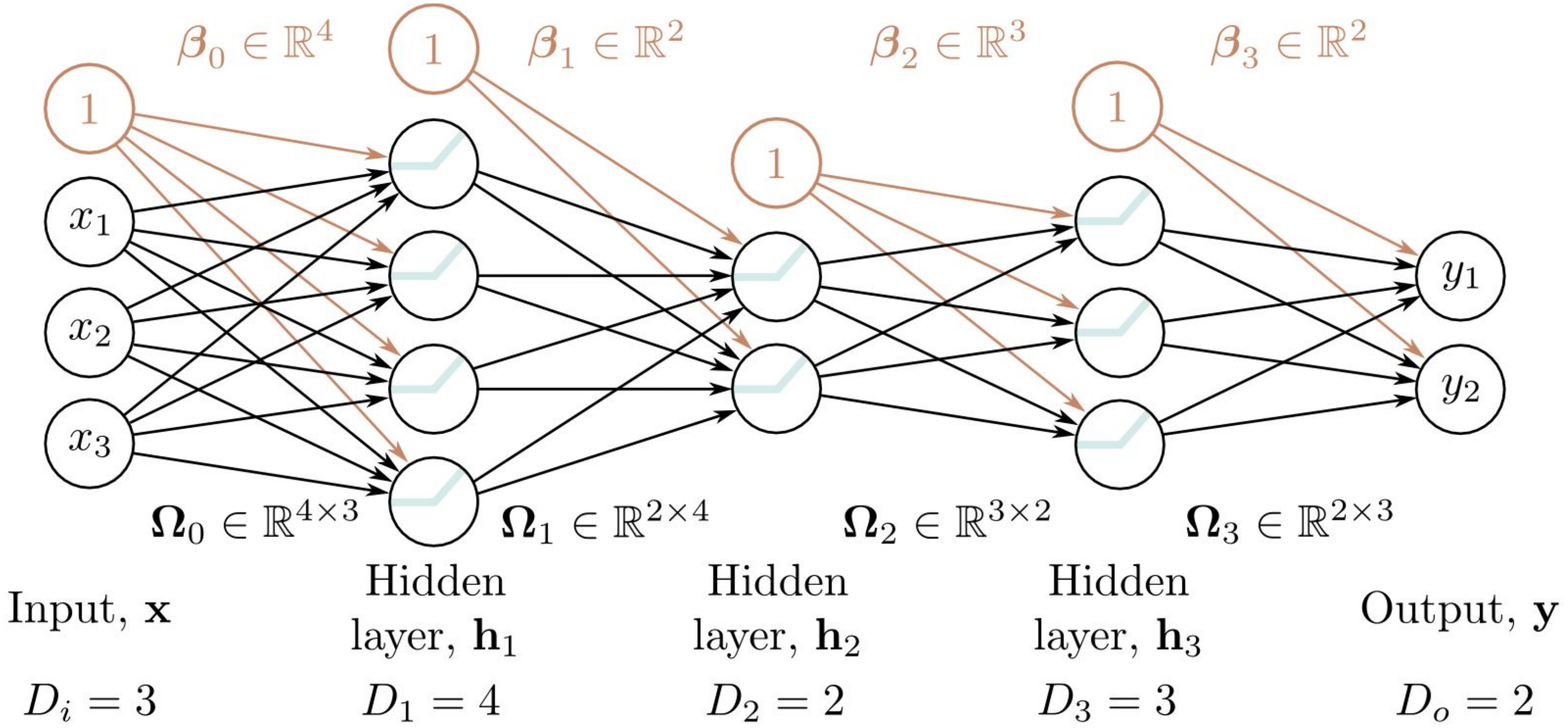
$$y = \phi'_0 + \boldsymbol{\phi}'\mathbf{h}' \longrightarrow \mathbf{y} = \boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2$$

General equation for deep networks

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

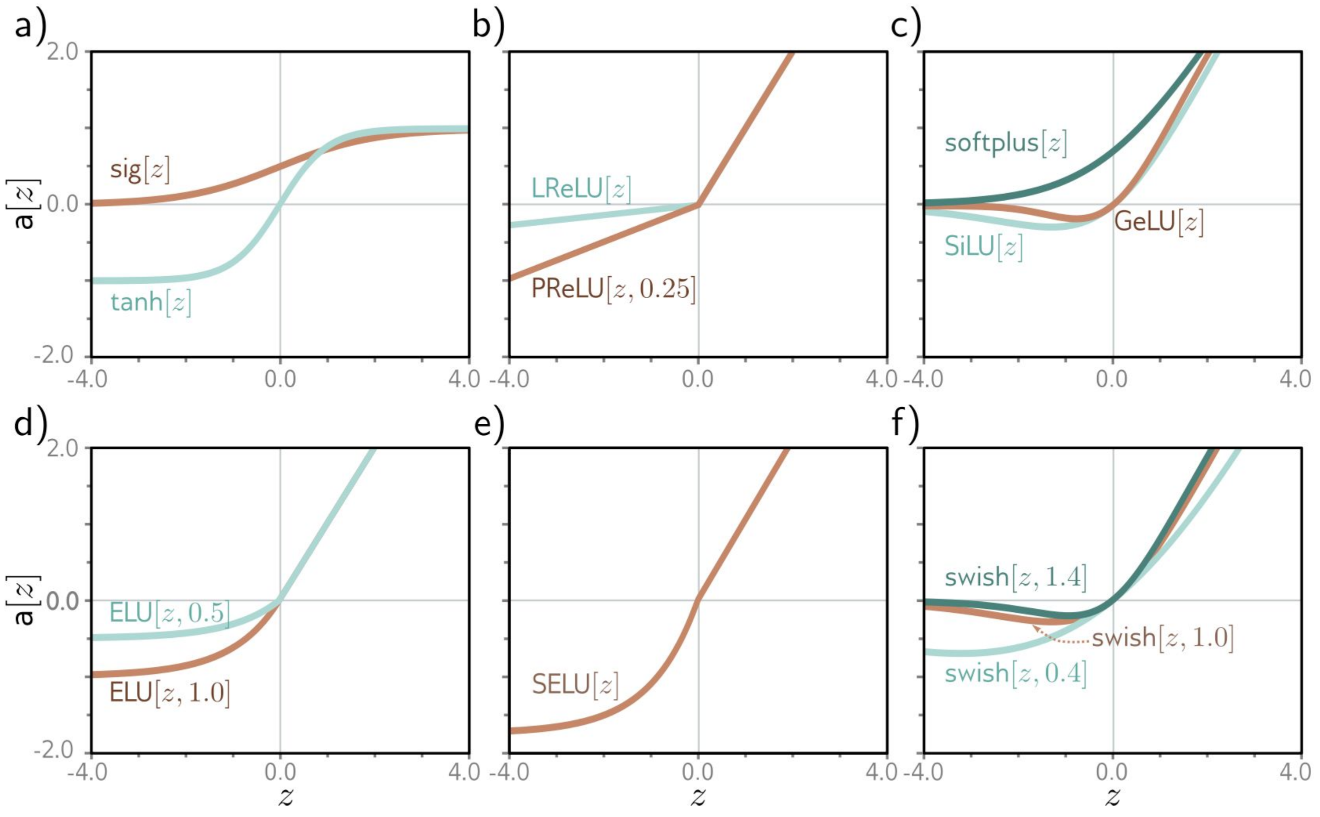
$$
\mathbf{y} = \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K\mathbf{a}\left[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1}\mathbf{a}\left[\ldots\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{a}\left[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{a}\left[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}\right]\right]\ldots\right]\right]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

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



$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
1
1
1
x_1
x_2
x_3
y_1
y_2
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{h}_1$
Hidden layer, $\mathbf{h}_2$
Hidden layer, $\mathbf{h}_3$
Output, $\mathbf{y}$
$D_i = 3$
$D_1 = 4$
$D_2 = 2$
$D_3 = 3$
$D_o = 2$

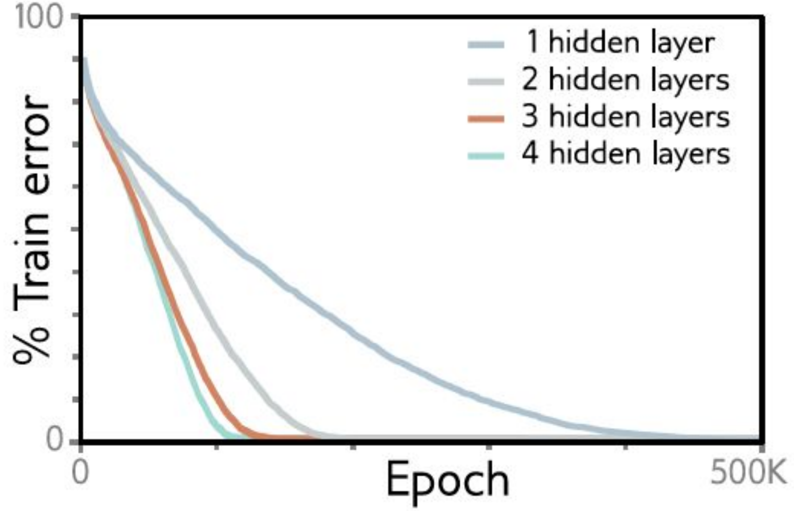
Activation functions



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

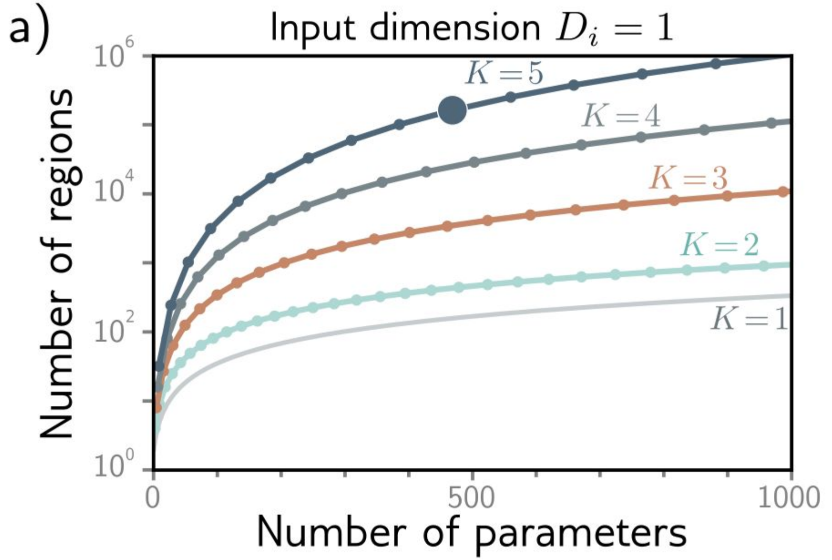
The best results are obtained by deep networks with many layers:

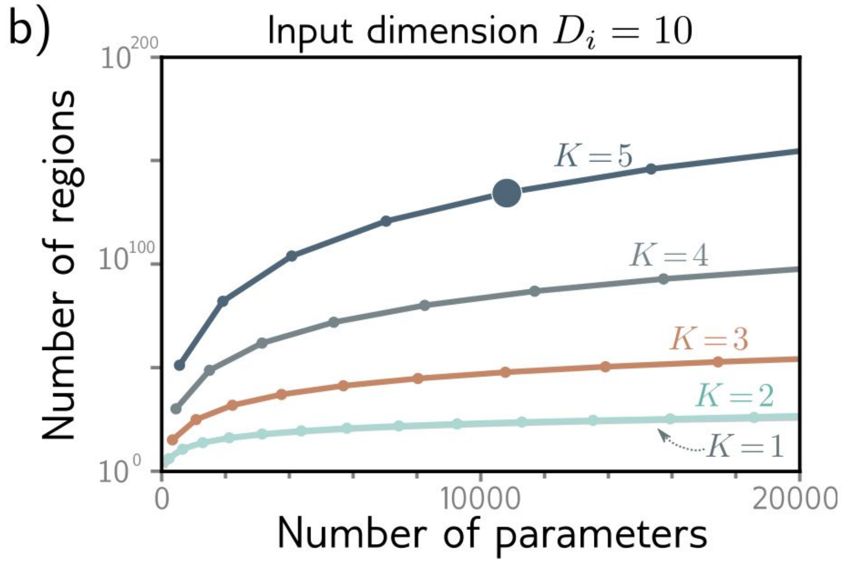
- 50-1000 layers for most applications
- Over ~10-15 layers additional tricks are required (normalisation, residual connections)
- Best results in:
  - Computer vision
  - Natural language processing
  - Graph neural networks
  - Generative models
  - Reinforcement learning (ongoing research)

1. Function approximation capacity

- Both shallow and deep neural networks obey the universal approximation theorem.
- Does it mean that one layer is enough?!

2. Number of linear regions per parameter



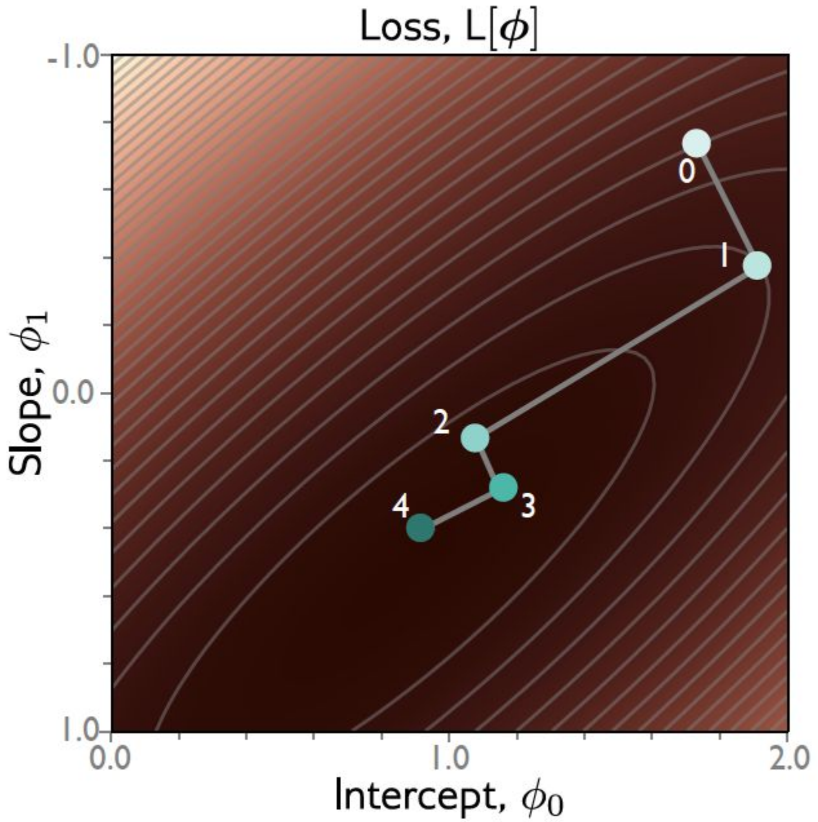


5 layers (up to)
marker: 10 hidden units per layer
471 parameters
161,501 linear regions

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

Loss functions

Training a simple model



1. Define a loss function
2. Compute the change in parameters required to make the loss smaller
3. Apply the change and get new parameters
4. Repeat from (2)

Loss function

- Training dataset of I pairs of input/output examples:

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$$

- Loss function or cost function measures how bad the model is:

$$L\Big[\underbrace{\phi, \mathrm{f}[\mathbf{x}, \phi]}_{\text{model}}, \underbrace{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}}_{\text{train data}}\Big]$$

Loss function

- Training dataset of I pairs of input/output examples:

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$$

- Loss function or cost function measures how bad model is:

$$L[\phi]$$

Returns a scalar that is smaller when model maps inputs to outputs better

Loss function as an optimization objective

$$L\left[\underbrace{\phi, \mathrm{f}[\mathbf{x}, \phi]}_{\text{model}}, \underbrace{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}}_{\text{train data}}\right]$$

Find the parameters that minimize the loss:

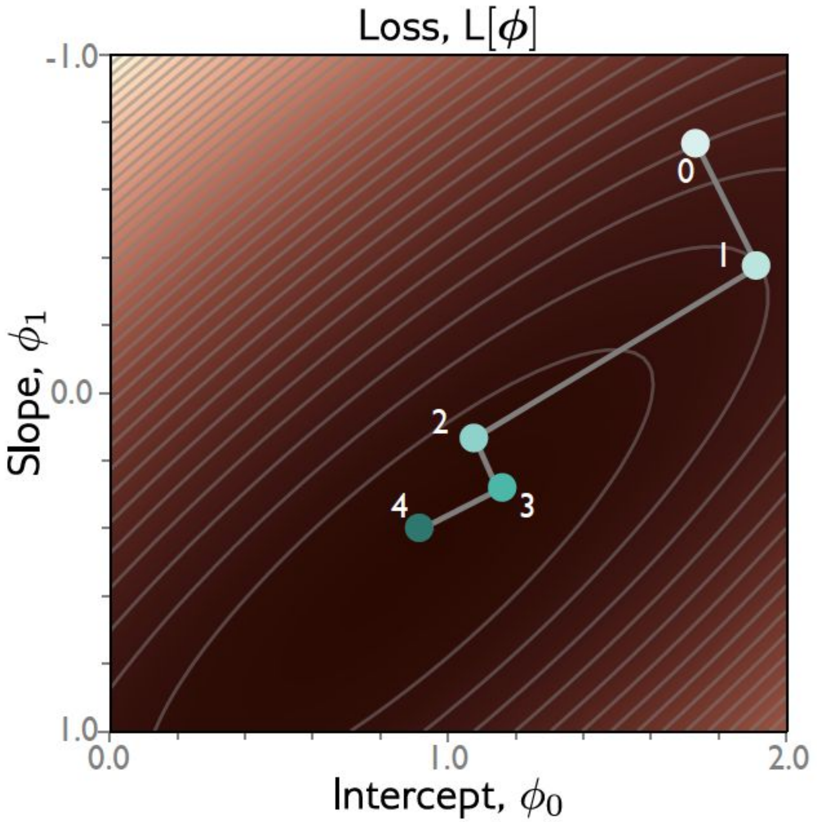
$$\phi = \operatorname{argmin}_{\phi} \left[\, L(\phi) \,\right]$$

Example:

$$L(\phi) = \mathbb{E}[\underbrace{r + \gamma \max_u Q(x', u'; \phi)}_{\text{target}} - \underbrace{Q(x, u; \phi)}_{\text{current estimate}}]^2$$

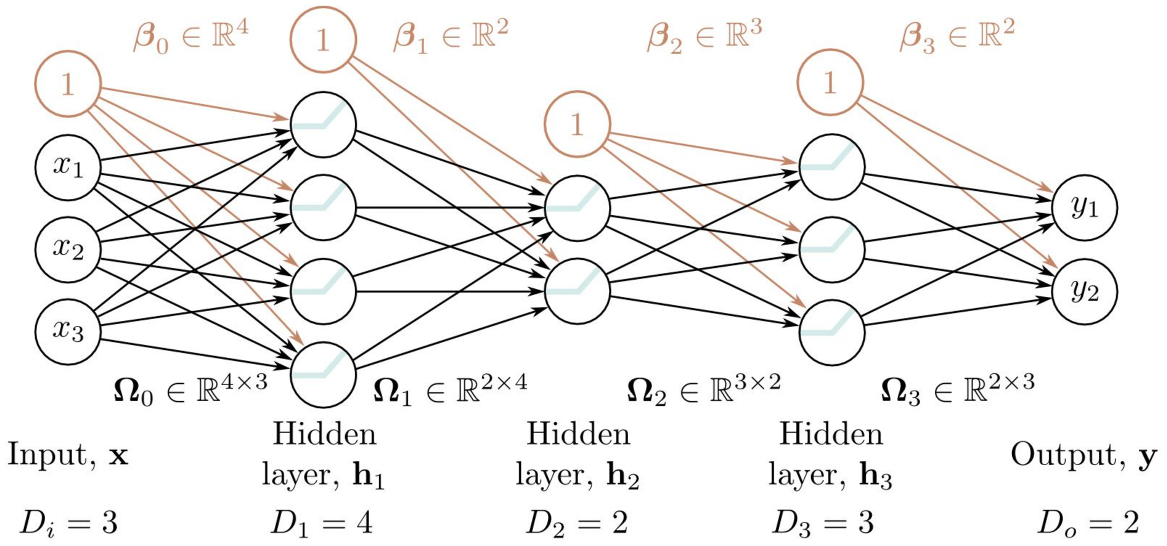
Computing gradients

Training a simple model



1. Define a loss function
2. Compute the change in parameters required to make the loss smaller
3. Apply the change and get new parameters
4. Repeat from (2)

Neural network


$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
1
1
1
x_1
x_2
x_3
y_1
y_2
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times 3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times 4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times 2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times 3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{h}_1$
Hidden layer, $\mathbf{h}_2$
Hidden layer, $\mathbf{h}_3$
Output, $\mathbf{y}$
$D_i = 3$
$D_1 = 4$
$D_2 = 2$
$D_3 = 3$
$D_o = 2$

$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}] \\
\mathbf{h}_2 &= \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{h}_1] \\
\mathbf{h}_3 &= \mathbf{a}[\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{h}_2] \\
\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}] &= \boldsymbol{\beta}_3 + \boldsymbol{\Omega}_3\mathbf{h}_3
\end{aligned}
$$

Setup

Loss, sum of individual terms:

$$L[\phi] = \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \mathrm{l}[\mathrm{f}[\mathbf{x}_i, \phi], y_i$$

SGD algorithm

$$\phi_{t+1} \longleftarrow \phi_t - \alpha \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t]}{\partial \phi}$$

Parameters

$$\phi = \{\boldsymbol{\beta}_0, \boldsymbol{\Omega}_0, \boldsymbol{\beta}_1, \boldsymbol{\Omega}_1, \boldsymbol{\beta}_2, \boldsymbol{\Omega}_2, \boldsymbol{\beta}_3, \boldsymbol{\Omega}_3\}$$

How to compute gradients?

$$\frac{\partial \ell_i}{\partial \boldsymbol{\beta}_k} \quad \text{and} \quad \frac{\partial \ell_i}{\partial \boldsymbol{\Omega}_k}$$

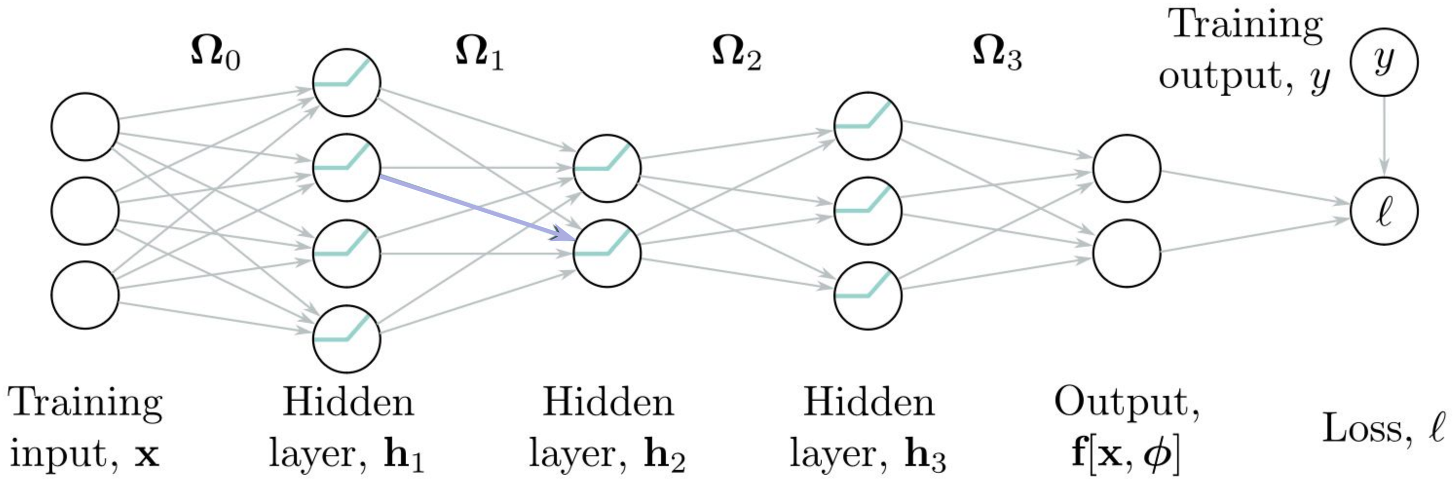
Big deal?

$$\begin{aligned} y' = \phi'_0 + \phi'_1 \mathrm{a}\left[\psi_{10} + \psi_{11}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{12}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{13}\mathrm{a}[\theta_{30} + \theta_{31}x]\right] \\ + \phi'_2\mathrm{a}[\psi_{20} + \psi_{21}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{22}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{23}\mathrm{a}[\theta_{30} + \theta_{31}x]] \\ + \phi'_3\mathrm{a}[\psi_{30} + \psi_{31}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{32}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{33}\mathrm{a}[\theta_{30} + \theta_{31}x]] \end{aligned}$$

Huge equation, and we need to compute derivatives:

- for every parameter
- for every point in the batch
- for every iteration of SGD

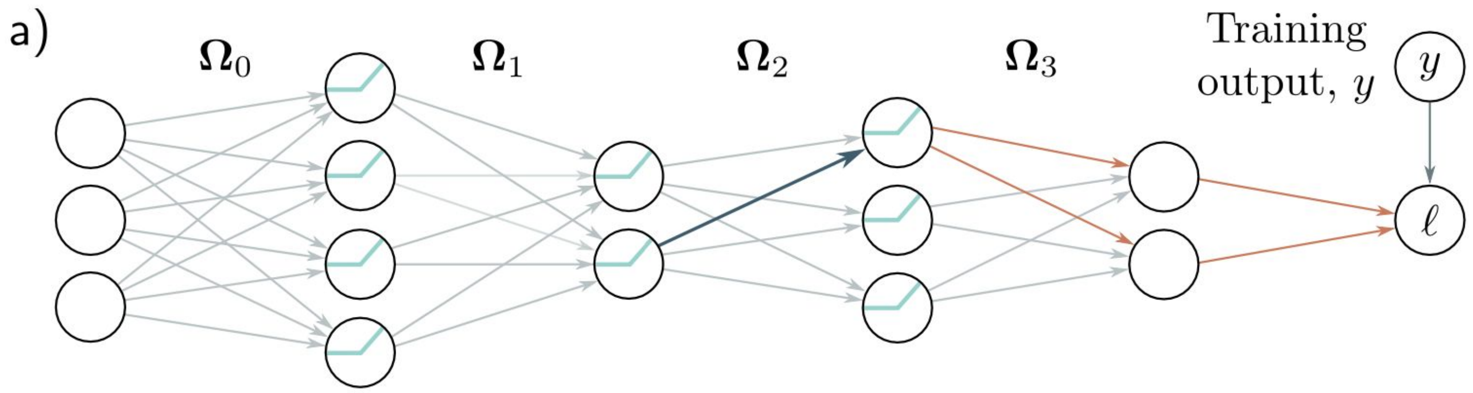
Backpropagation algorithm. Forward pass



$\mathbf{\Omega}_0$
$\mathbf{\Omega}_1$
$\mathbf{\Omega}_2$
$\mathbf{\Omega}_3$
Training output, y
y
ℓ
Training input, $\mathbf{x}$
Hidden layer, $\mathbf{h}_1$
Hidden layer, $\mathbf{h}_2$
Hidden layer, $\mathbf{h}_3$
Output, $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$
Loss, ℓ

- Blue weight multiplies activation (ReLU output) in previous layer
- We want to know how change in blue weight affects loss
- If we double activation in previous layer, weight will have twice the effect
- Conclusion: we need to know the activations at each layer.

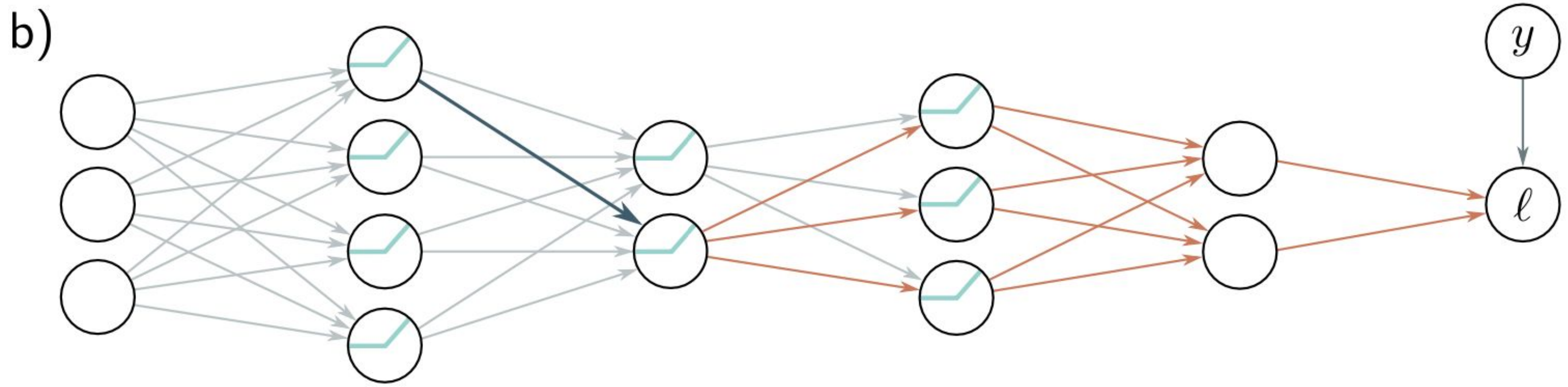
Backpropagation algorithm. Backward pass



To compute how a small change in a weight feeding into $\mathbf{h}_3$ modifies the loss, we need:

- How $\mathbf{h}_3$ changes the model output
- How the output changes the loss

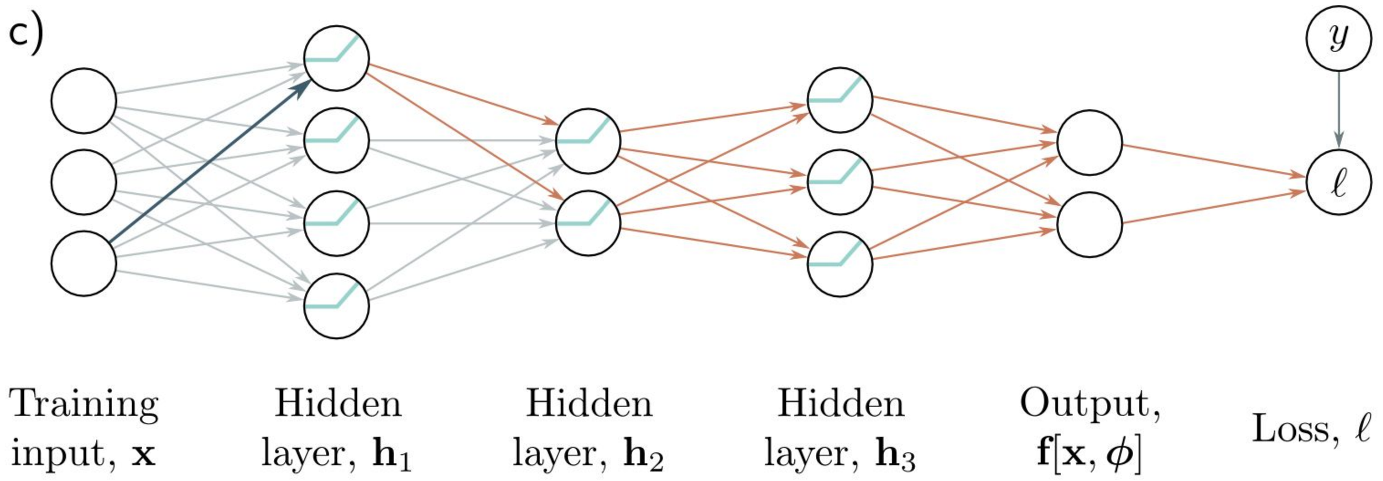
Backpropagation algorithm. Backward pass



To compute how a small change in a weight feeding into $\mathbf{h}_2$ modifies the loss, we need:

- How a change in layer $\mathbf{h}_2$ affects $\mathbf{h}_3$
- How $\mathbf{h}_3$ changes the model output
- How the output changes the loss

Backpropagation algorithm. Backward pass


c)
y
ℓ
Training input, $\mathbf{x}$
Hidden layer, $\mathbf{h}_1$
Hidden layer, $\mathbf{h}_2$
Hidden layer, $\mathbf{h}_3$
Output, $\mathbf{f}[\mathbf{x}, \phi]$
Loss, ℓ

To compute how a small change in a weight feeding into $\mathbf{h}_2$ modifies the loss, we need:

- How a change in layer $\mathbf{h}_1$ affects $\mathbf{h}_2$
- How a change in layer $\mathbf{h}_2$ affects $\mathbf{h}_3$
- How $\mathbf{h}_3$ changes the model output
- How the output changes the loss

Toy example

$$\mathrm{f}[x, \boldsymbol{\phi}] = \beta_3 + \omega_3 \cdot \cos\Big[\beta_2 + \omega_2 \cdot \exp\big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x]\big]\Big]$$

$$\ell_i = (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

- A series of functions composed with each other
- Unlike in neural networks it consists of scalars and not vectors and matrices
- The “activation functions” are just $\sin$, $\exp$, $\cos$

Toy example

$$\mathrm{f}[x, \boldsymbol{\phi}] = \beta_3 + \omega_3 \cdot \cos\Big[\beta_2 + \omega_2 \cdot \exp\big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x]\big]\Big]$$

$$\ell_i = (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

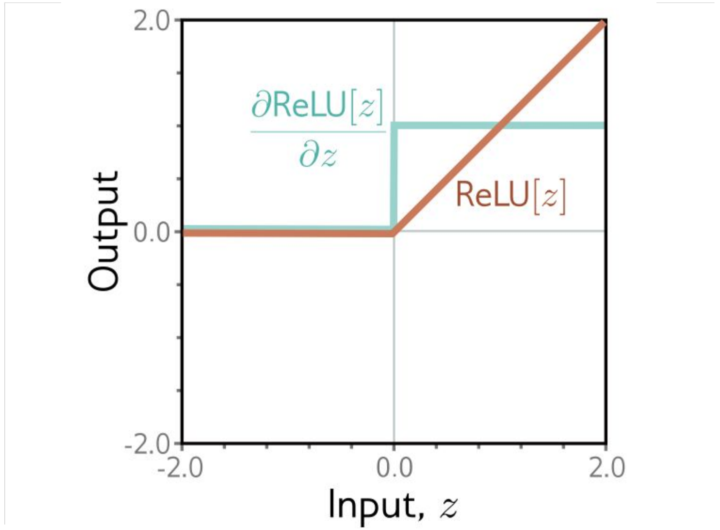
Derivatives of the activation functions

$$\frac{\partial \cos[z]}{\partial z} = -\sin[z] \qquad \frac{\partial \exp[z]}{\partial z} = \exp[z] \qquad \frac{\partial \sin[z]}{\partial z} = -\cos[z$$

Warmup! Derivative of ReLU

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

Rectified Linear Unit



$$\mathbb{I}[z > 0]$$

Toy example

$$\mathrm{f}[x, \boldsymbol{\phi}] = \beta_3 + \omega_3 \cdot \cos\Big[\beta_2 + \omega_2 \cdot \exp\big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x]\big]\Big]$$

$$\ell_i = (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

How does a small change in β_2 change the loss ℓ_i for the i'th example?

We want to compute:

$$\frac{\partial \ell_i}{\partial \beta_0}, \quad \frac{\partial \ell_i}{\partial \omega_0}, \quad \frac{\partial \ell_i}{\partial \beta_1}, \quad \frac{\partial \ell_i}{\partial \omega_1}, \quad \frac{\partial \ell_i}{\partial \beta_2}, \quad \frac{\partial \ell_i}{\partial \omega_2}, \quad \frac{\partial \ell_i}{\partial \beta_3}, \quad \text{and} \quad \frac{\partial \ell_i}{\partial \omega_3}$$

Gradients of composite functions

$$\mathrm{f}[x, \boldsymbol{\phi}] = \beta_3 + \omega_3 \cdot \cos\Big[\beta_2 + \omega_2 \cdot \exp\big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x]\big]\Big]$$

$$\ell_i = (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

Calculating expressions by hand:

- Some expressions are very complicated
- There are some redundancies

$$\begin{aligned}\frac{\partial \ell_i}{\partial \omega_0} = \quad & -2\left(\beta_3 + \omega_3 \cdot \cos\Big[\beta_2 + \omega_2 \cdot \exp\big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x_i]\big]\Big] - y_i\right) \\ & \cdot \omega_1 \omega_2 \omega_3 \cdot x_i \cdot \cos[\beta_0 + \omega_0 \cdot x_i] \cdot \exp\Big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x_i]\Big] \\ & \cdot \sin\Big[\beta_2 + \omega_2 \cdot \exp\Big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x_i]\Big]\Big]\end{aligned}$$

Forward pass

$$\mathrm{f}[x, \boldsymbol{\phi}] = \beta_3 + \omega_3 \cdot \cos\Big[\beta_2 + \omega_2 \cdot \exp\big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x]\big]\Big]$$

$$\ell_i = (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

1. Write this as series of intermediate calculations

2. Compute these intermediate quantities

$$\begin{aligned} f_0 &= \beta_0 + \omega_0 \cdot x_i \\ h_1 &= \sin[f_0] \\ f_1 &= \beta_1 + \omega_1 \cdot h_1 \\ h_2 &= \exp[f_1] \end{aligned}$$

$$\begin{aligned} f_2 &= \beta_2 + \omega_2 \cdot h_2 \\ h_3 &= \cos[f_2] \\ f_3 &= \beta_3 + \omega_3 \cdot h_3 \\ \ell_i &= (f_3 - y_i)^2. \end{aligned}$$

$$\begin{tikzcd} x_i \arrow[r] & f_0 \arrow[r] & h_1 \arrow[r] & f_1 \arrow[r] & h_2 \arrow[r] & f_2 \arrow[r] & h_3 \arrow[r] & f_3 \arrow[r] & \ell_i \end{tikzcd}$$

Backward pass

$$\mathrm{f}[x, \boldsymbol{\phi}] = \beta_3 + \omega_3 \cdot \cos\Big[\beta_2 + \omega_2 \cdot \exp\big[\beta_1 + \omega_1 \cdot \sin[\beta_0 + \omega_0 \cdot x]\big]\Big]$$

$$\ell_i = (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$\frac{\partial \ell_i}{\partial f_3}, \quad \frac{\partial \ell_i}{\partial h_3}, \quad \frac{\partial \ell_i}{\partial f_2}, \quad \frac{\partial \ell_i}{\partial h_2}, \quad \frac{\partial \ell_i}{\partial f_1}, \quad \frac{\partial \ell_i}{\partial h_1}, \quad \text{and} \quad \frac{\partial \ell_i}{\partial f_0}$$

$$\begin{tikzcd}
x_i & \frac{\partial \ell_i}{\partial f_0} & \frac{\partial \ell_i}{\partial h_1} \arrow[l, "\frac{\partial h_1}{\partial f_0}"'] & \frac{\partial \ell_i}{\partial f_1} \arrow[l, "\frac{\partial f_1}{\partial h_1}"'] & \frac{\partial \ell_i}{\partial h_2} \arrow[l, "\frac{\partial h_2}{\partial f_1}"'] & \frac{\partial \ell_i}{\partial f_2} \arrow[l, "\frac{\partial f_2}{\partial h_2}"'] & \frac{\partial \ell_i}{\partial h_3} \arrow[l, "\frac{\partial h_3}{\partial f_2}"'] & \frac{\partial \ell_i}{\partial f_3} \arrow[l, "\frac{\partial f_3}{\partial h_3}"'] & \ell_i \arrow[l]
\end{tikzcd}$$

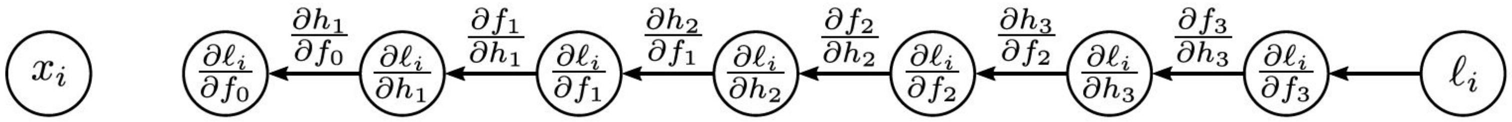
Backward pass

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$
\begin{aligned}
f_0 &= \beta_0 + \omega_0 \cdot x_i \\
h_1 &= \sin[f_0] \\
f_1 &= \beta_1 + \omega_1 \cdot h_1 \\
h_2 &= \exp[f_1]
\end{aligned}
\qquad
\begin{aligned}
f_2 &= \beta_2 + \omega_2 \cdot h_2 \\
h_3 &= \cos[f_2] \\
f_3 &= \beta_3 + \omega_3 \cdot h_3 \\
\ell_i &= (f_3 - y_i)^2.
\end{aligned}
$$

The first are easy:

$$\frac{\partial \ell_i}{\partial f_3} = 2(f_3 - y_i)$$



$$\begin{tikzcd}
x_i & \frac{\partial \ell_i}{\partial f_0} & \frac{\partial \ell_i}{\partial h_1} \arrow[l, "\frac{\partial h_1}{\partial f_0}"'] & \frac{\partial \ell_i}{\partial f_1} \arrow[l, "\frac{\partial f_1}{\partial h_1}"'] & \frac{\partial \ell_i}{\partial h_2} \arrow[l, "\frac{\partial h_2}{\partial f_1}"'] & \frac{\partial \ell_i}{\partial f_2} \arrow[l, "\frac{\partial f_2}{\partial h_2}"'] & \frac{\partial \ell_i}{\partial h_3} \arrow[l, "\frac{\partial h_3}{\partial f_2}"'] & \frac{\partial \ell_i}{\partial f_3} \arrow[l, "\frac{\partial f_3}{\partial h_3}"'] & \ell_i \arrow[l]
\end{tikzcd}$$

Backward pass

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$
\begin{aligned}
f_0 &= \beta_0 + \omega_0 \cdot x_i \\
h_1 &= \sin[f_0] \\
f_1 &= \beta_1 + \omega_1 \cdot h_1 \\
h_2 &= \exp[f_1]
\end{aligned}
$$

$$
\begin{aligned}
f_2 &= \beta_2 + \omega_2 \cdot h_2 \\
h_3 &= \cos[f_2] \\
f_3 &= \beta_3 + \omega_3 \cdot h_3 \\
\ell_i &= (f_3 - y_i)^2.
\end{aligned}
$$

The rest are computed using the *chain rule*.

$$
\frac{\partial \ell_i}{\partial h_3} = \frac{\partial f_3}{\partial h_3}\frac{\partial \ell_i}{\partial f_3}
$$

How does a small change in $\mathbf{h}_3$ change ℓ_i?

How does a small change in $\mathbf{h}_3$ change $\mathbf{f}_3$?

How does a small change in $\mathbf{f}_3$ change ℓ_i?

Backward pass

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$\begin{aligned} f_0 &= \beta_0 + \omega_0 \cdot x_i \\ h_1 &= \sin[f_0] \\ f_1 &= \beta_1 + \omega_1 \cdot h_1 \\ h_2 &= \exp[f_1] \end{aligned}$$

$$\begin{aligned} f_2 &= \beta_2 + \omega_2 \cdot h_2 \\ h_3 &= \cos[f_2] \\ f_3 &= \beta_3 + \omega_3 \cdot h_3 \\ \ell_i &= (f_3 - y_i)^2. \end{aligned}$$

The rest are computed using the *chain rule*.

$$\frac{\partial \ell_i}{\partial h_3} = \frac{\partial f_3}{\partial h_3}\frac{\partial \ell_i}{\partial f_3}$$

?

How does a small change in $\mathbf{h}_3$ change ℓ_i?

w_3

Already computed!

Backward pass

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$\begin{aligned} f_0 &= \beta_0 + \omega_0 \cdot x_i \\ h_1 &= \sin[f_0] \\ f_1 &= \beta_1 + \omega_1 \cdot h_1 \\ h_2 &= \exp[f_1] \end{aligned}$$

$$\begin{aligned} f_2 &= \beta_2 + \omega_2 \cdot h_2 \\ h_3 &= \cos[f_2] \\ f_3 &= \beta_3 + \omega_3 \cdot h_3 \\ \ell_i &= (f_3 - y_i)^2. \end{aligned}$$

The rest are computed using the *chain rule*.

$$\frac{\partial \ell_i}{\partial f_2} = \underbrace{\left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\right)}_{\text{Already computed!}} \underbrace{\frac{\partial h_3}{\partial f_2}}_{-\sin(\mathbf{f}_2)}$$

?

Backward pass

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$
\begin{aligned}
f_0 &= \beta_0 + \omega_0 \cdot x_i \\
h_1 &= \sin[f_0] \\
f_1 &= \beta_1 + \omega_1 \cdot h_1 \\
h_2 &= \exp[f_1]
\end{aligned}
$$

$$
\begin{aligned}
f_2 &= \beta_2 + \omega_2 \cdot h_2 \\
h_3 &= \cos[f_2] \\
f_3 &= \beta_3 + \omega_3 \cdot h_3 \\
\ell_i &= (f_3 - y_i)^2.
\end{aligned}
$$

$$\frac{\partial \ell_i}{\partial f_2} = \left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\right)\frac{\partial h_3}{\partial f_2}$$

The rest are computed using the *chain rule*.

Backward pass

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$
\begin{aligned}
f_0 &= \beta_0 + \omega_0 \cdot x_i \\
h_1 &= \sin[f_0] \\
f_1 &= \beta_1 + \omega_1 \cdot h_1 \\
h_2 &= \exp[f_1]
\end{aligned}
$$

$$
\begin{aligned}
f_2 &= \beta_2 + \omega_2 \cdot h_2 \\
h_3 &= \cos[f_2] \\
f_3 &= \beta_3 + \omega_3 \cdot h_3 \\
\ell_i &= (f_3 - y_i)^2.
\end{aligned}
$$

The rest are computed using the *chain rule*.

$$
\begin{aligned}
\frac{\partial \ell_i}{\partial f_2} &= \left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\right)\frac{\partial h_3}{\partial f_2} \\
\frac{\partial \ell_i}{\partial h_2} &= \left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\frac{\partial h_3}{\partial f_2}\right)\frac{\partial f_2}{\partial h_2}
\end{aligned}
$$

Backward pass

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$\begin{aligned} f_0 &= \beta_0 + \omega_0 \cdot x_i \\ h_1 &= \sin[f_0] \\ f_1 &= \beta_1 + \omega_1 \cdot h_1 \\ h_2 &= \exp[f_1] \end{aligned}$$

$$\begin{aligned} f_2 &= \beta_2 + \omega_2 \cdot h_2 \\ h_3 &= \cos[f_2] \\ f_3 &= \beta_3 + \omega_3 \cdot h_3 \\ \ell_i &= (f_3 - y_i)^2. \end{aligned}$$

The rest are computed using the *chain rule*.

$$\begin{aligned}
\frac{\partial \ell_i}{\partial f_2} &= \left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\right)\frac{\partial h_3}{\partial f_2} \\
\frac{\partial \ell_i}{\partial h_2} &= \left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\frac{\partial h_3}{\partial f_2}\right)\frac{\partial f_2}{\partial h_2} \\
\frac{\partial \ell_i}{\partial f_1} &= \left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\frac{\partial h_3}{\partial f_2}\frac{\partial f_2}{\partial h_2}\right)\frac{\partial h_2}{\partial f_1} \\
\frac{\partial \ell_i}{\partial h_1} &= \left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\frac{\partial h_3}{\partial f_2}\frac{\partial f_2}{\partial h_2}\frac{\partial h_2}{\partial f_1}\right)\frac{\partial f_1}{\partial h_1} \\
\frac{\partial \ell_i}{\partial f_0} &= \left(\frac{\partial \ell_i}{\partial f_3}\frac{\partial f_3}{\partial h_3}\frac{\partial h_3}{\partial f_2}\frac{\partial f_2}{\partial h_2}\frac{\partial h_2}{\partial f_1}\frac{\partial f_1}{\partial h_1}\right)\frac{\partial h_1}{\partial f_0}.
\end{aligned}$$

Backward pass

1. Compute the derivatives of the loss with respect to these intermediate quantities, in *reverse order*.

$$
\begin{aligned}
f_0 &= \beta_0 + \omega_0 \cdot x_i \\
h_1 &= \sin[f_0] \\
f_1 &= \beta_1 + \omega_1 \cdot h_1 \\
h_2 &= \exp[f_1]
\end{aligned}
\qquad
\begin{aligned}
f_2 &= \beta_2 + \omega_2 \cdot h_2 \\
h_3 &= \cos[f_2] \\
f_3 &= \beta_3 + \omega_3 \cdot h_3 \\
\ell_i &= (f_3 - y_i)^2.
\end{aligned}
$$

Chain rule all the way down!

$$\begin{tikzcd}
x_i & \frac{\partial \ell_i}{\partial f_0} & \frac{\partial \ell_i}{\partial h_1} \arrow[l, "\frac{\partial h_1}{\partial f_0}"'] & \frac{\partial \ell_i}{\partial f_1} \arrow[l, "\frac{\partial f_1}{\partial h_1}"'] & \frac{\partial \ell_i}{\partial h_2} \arrow[l, "\frac{\partial h_2}{\partial f_1}"'] & \frac{\partial \ell_i}{\partial f_2} \arrow[l, "\frac{\partial f_2}{\partial h_2}"'] & \frac{\partial \ell_i}{\partial h_3} \arrow[l, "\frac{\partial h_3}{\partial f_2}"'] & \frac{\partial \ell_i}{\partial f_3} \arrow[l, "\frac{\partial f_3}{\partial h_3}"'] & \ell_i \arrow[l]
\end{tikzcd}$$

Backward pass

2. Find how the loss changes as a function of the parameters β and ω.

$$\begin{aligned} f_0 &= \beta_0 + \omega_0 \cdot x_i \\ h_1 &= \sin[f_0] \\ f_1 &= \beta_1 + \omega_1 \cdot h_1 \\ h_2 &= \exp[f_1] \end{aligned}$$

$$\begin{aligned} f_2 &= \beta_2 + \omega_2 \cdot h_2 \\ h_3 &= \cos[f_2] \\ f_3 &= \beta_3 + \omega_3 \cdot h_3 \\ \ell_i &= (f_3 - y_i)^2. \end{aligned}$$

Chain rule all the way down! Same recipe for weight and bias terms too!

$$\begin{tikzcd}
x_i & & \frac{\partial \ell_i}{\partial f_0} \arrow[dl, "\frac{\partial f_0}{\partial \beta_0}"'] \arrow[dr, "\frac{\partial f_0}{\partial \omega_0}"] & & \frac{\partial \ell_i}{\partial h_1} \arrow[ll] & & \frac{\partial \ell_i}{\partial f_1} \arrow[ll] \arrow[dl, "\frac{\partial f_1}{\partial \beta_1}"'] \arrow[dr, "\frac{\partial f_1}{\partial \omega_1}"] & & \frac{\partial \ell_i}{\partial h_2} \arrow[ll] & & \frac{\partial \ell_i}{\partial f_2} \arrow[ll] \arrow[dl, "\frac{\partial f_2}{\partial \beta_2}"'] \arrow[dr, "\frac{\partial f_2}{\partial \omega_2}"] & & \frac{\partial \ell_i}{\partial h_3} \arrow[ll] & & \frac{\partial \ell_i}{\partial f_3} \arrow[ll] \arrow[dl, "\frac{\partial f_3}{\partial \beta_3}"'] \arrow[dr, "\frac{\partial f_3}{\partial \omega_3}"] & & \ell_i \arrow[ll] \\
 & \frac{\partial \ell_i}{\partial \beta_0} & & \frac{\partial \ell_i}{\partial \omega_0} & & \frac{\partial \ell_i}{\partial \beta_1} & & \frac{\partial \ell_i}{\partial \omega_1} & & \frac{\partial \ell_i}{\partial \beta_2} & & \frac{\partial \ell_i}{\partial \omega_2} & & \frac{\partial \ell_i}{\partial \beta_3} & & \frac{\partial \ell_i}{\partial \omega_3} &
\end{tikzcd}$$

Backward pass

2. Find how the loss changes as a function of the parameters β and ω.

$$\begin{aligned} f_0 &= \beta_0 + \omega_0 \cdot x_i \\ h_1 &= \sin[f_0] \\ f_1 &= \beta_1 + \omega_1 \cdot h_1 \\ h_2 &= \exp[f_1] \end{aligned}$$

$$\begin{aligned} f_2 &= \beta_2 + \omega_2 \cdot h_2 \\ h_3 &= \cos[f_2] \\ f_3 &= \beta_3 + \omega_3 \cdot h_3 \\ \ell_i &= (f_3 - y_i)^2. \end{aligned}$$

Chain rule all the way down! Same recipe for weight and bias terms too!

$$\frac{\partial \ell_i}{\partial \omega_k} = \frac{\partial f_k}{\partial \omega_k} \frac{\partial \ell_i}{\partial f_k}$$

Reading suggestion

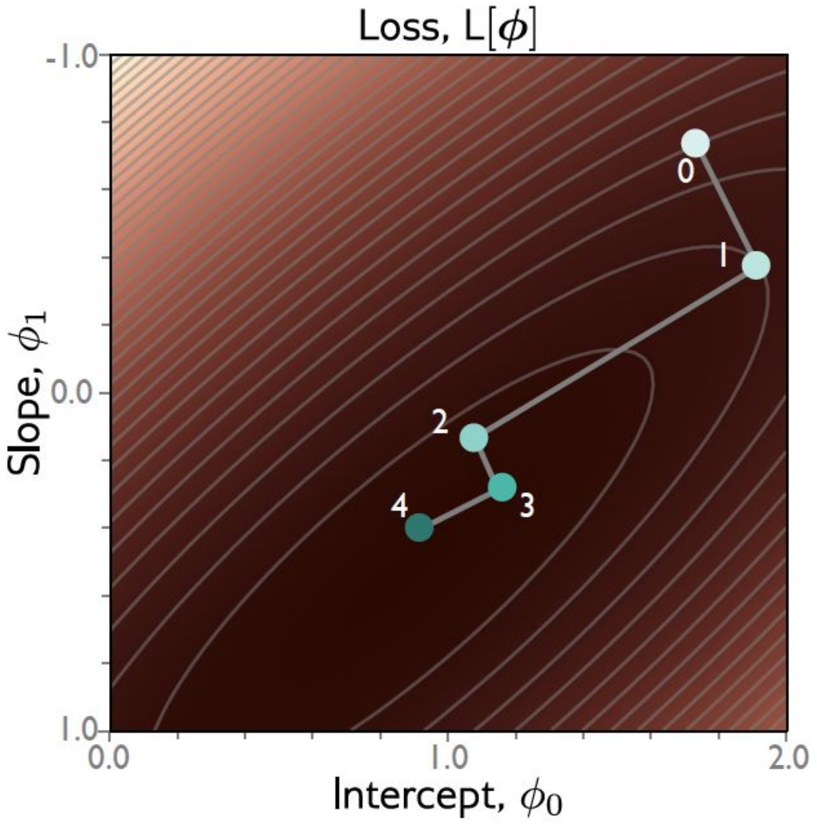
Check 7.4 in your textbook for a discussion on the extension to matrix calculus.

Automatic (or algorithmic) differentiation

- Modern deep learning frameworks compute derivatives automatically
- You just have to specify the model and the loss
- How?
    - Each component knows how to compute its own derivative
        - ReLU knows how to compute deriv of output w.r.t. input
        - Linear function knows how to compute deriv of output w.r.t. input
        - Linear function knows how to compute deriv of output w.r.t. parameter
    - You specify the order of the components
    - It can compute the chain of derivatives
- Works with branches as long as it's still an acyclic graph
- In a nutshell: AD takes a program which computes a value and automatically constructs a procedure for computing derivatives of that value.

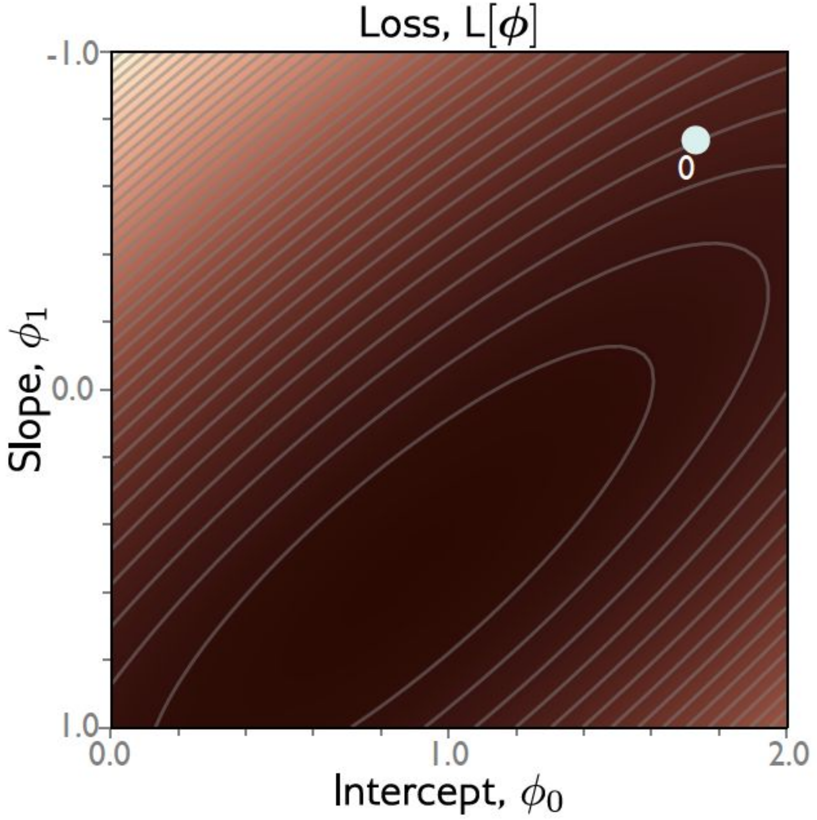
Optimisation

Training a simple model



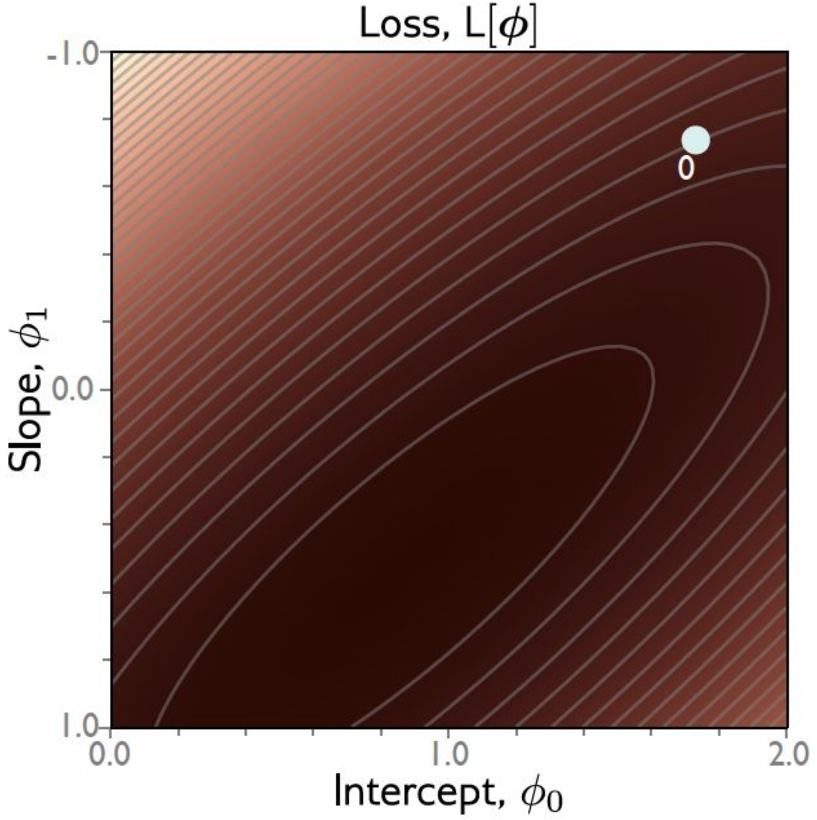
1. Define a loss function
2. Compute the change in parameters required to make the loss smaller
3. Apply the change and get new parameters
4. Repeat from (2)

Gradient descent



$$
\begin{aligned}
L[\phi] &= \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2 \\
&= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2
\end{aligned}
$$

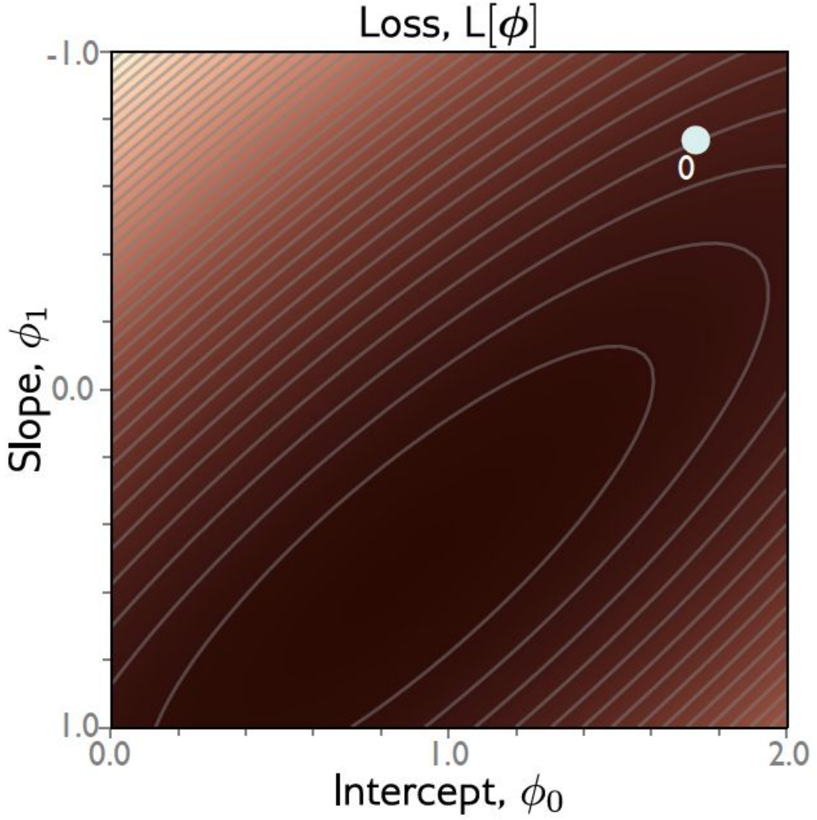
Gradient descent



$$\begin{aligned} L[\boldsymbol{\phi}] &= \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2 \\ &= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2 \end{aligned}$$

$$\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}$$

Gradient descent

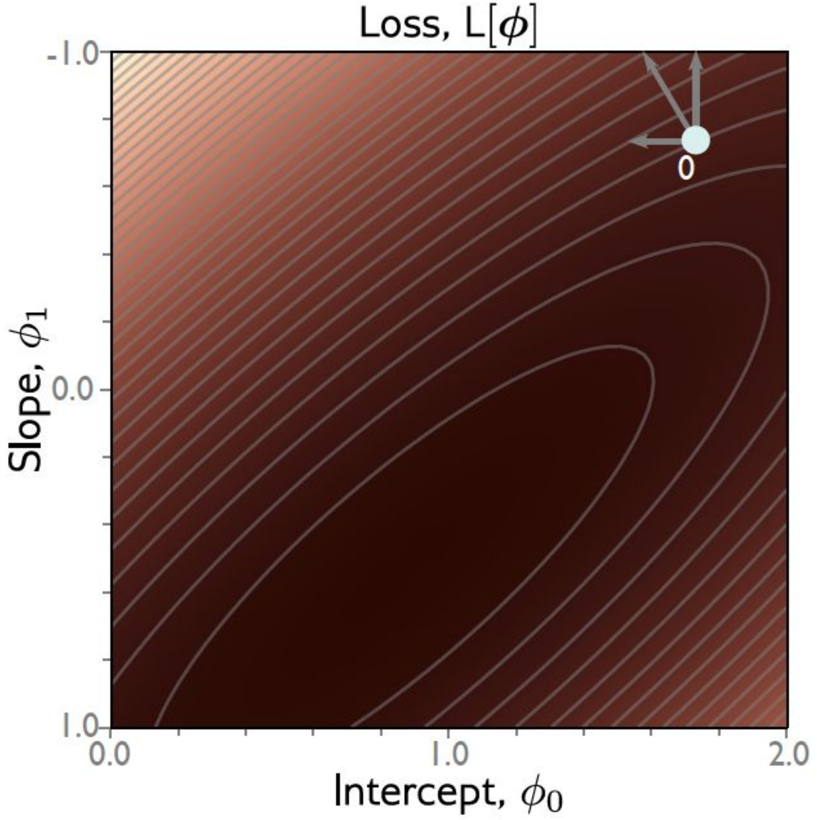


$$
\begin{aligned}
L[\boldsymbol{\phi}] &= \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2 \\
&= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2
\end{aligned}
$$

$$
\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}
$$

$$
\frac{\partial \ell_i}{\partial \boldsymbol{\phi}} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}
$$

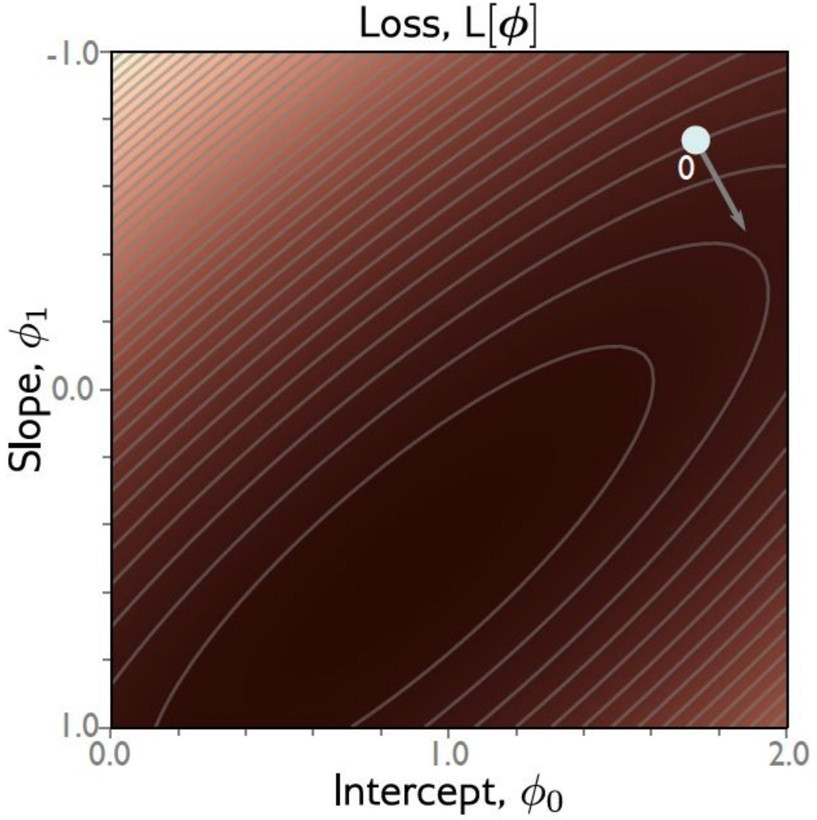
Gradient descent



$$\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}$$

$$\frac{\partial \ell_i}{\partial \boldsymbol{\phi}} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Gradient descent



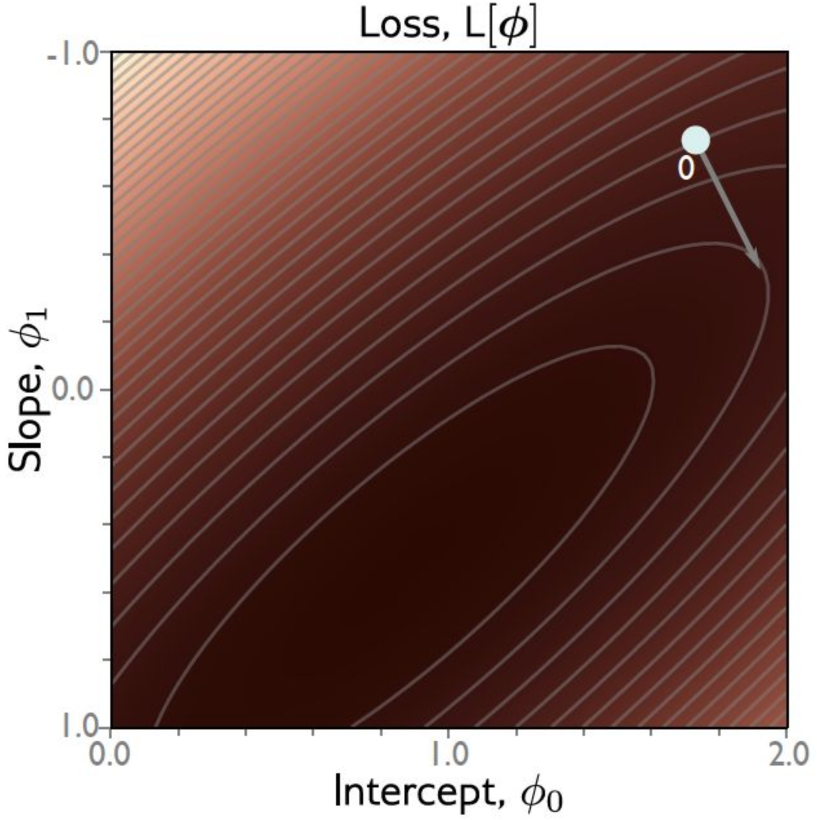
$$\frac{\partial L}{\partial \phi} = \frac{\partial}{\partial \phi}\sum_{i=1}^{I}\ell_i = \sum_{i=1}^{I}\frac{\partial \ell_i}{\partial \phi}$$

$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

$$\phi \longleftarrow \phi - \alpha \frac{\partial L}{\partial \phi}$$

α = step size or learning rate if fixed

Gradient descent



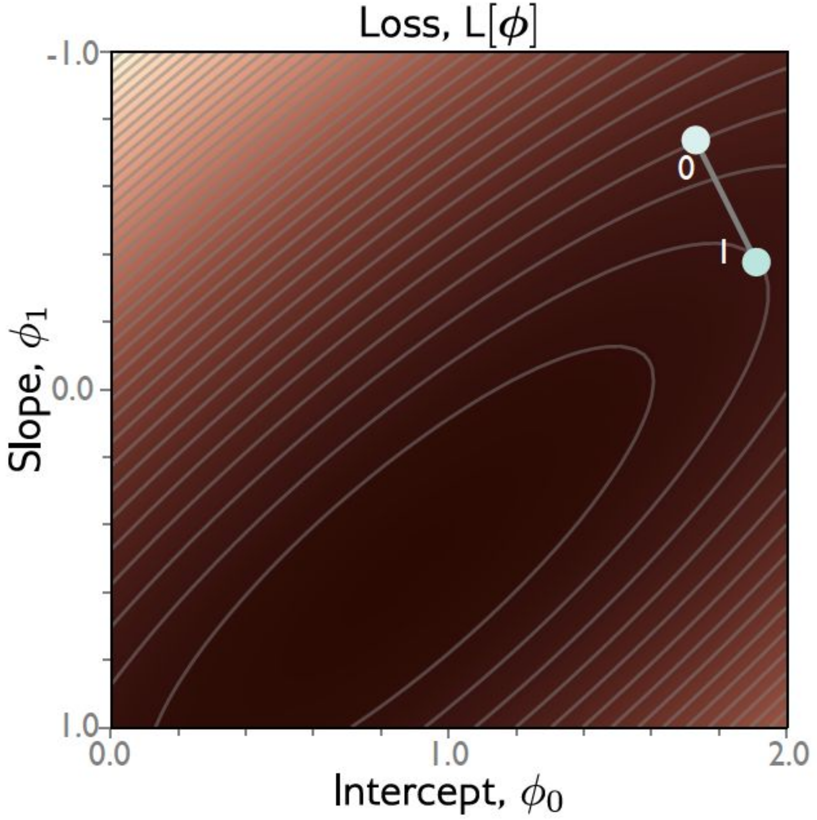
$$\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}$$

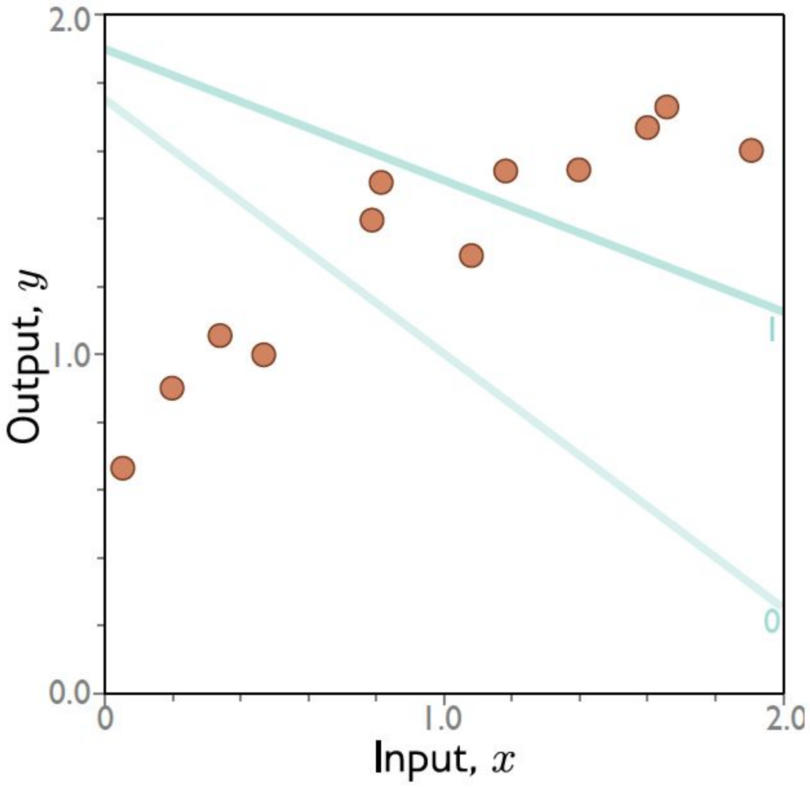
$$\frac{\partial \ell_i}{\partial \boldsymbol{\phi}} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

$$\boldsymbol{\phi} \longleftarrow \boldsymbol{\phi} - \alpha \frac{\partial L}{\partial \boldsymbol{\phi}}$$

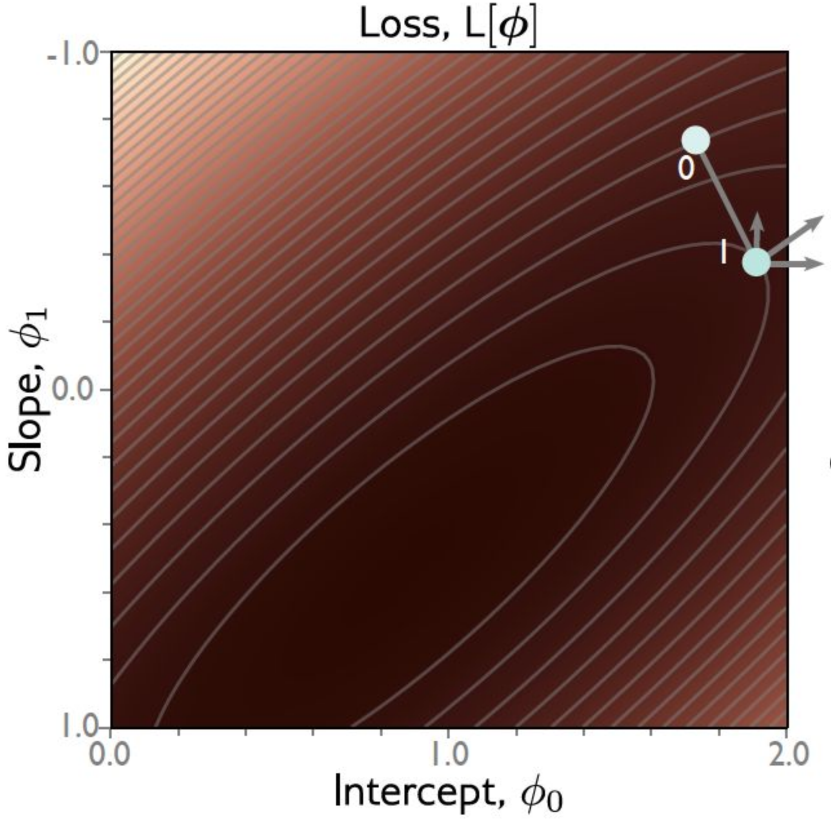
α = step size

Gradient descent





Gradient descent



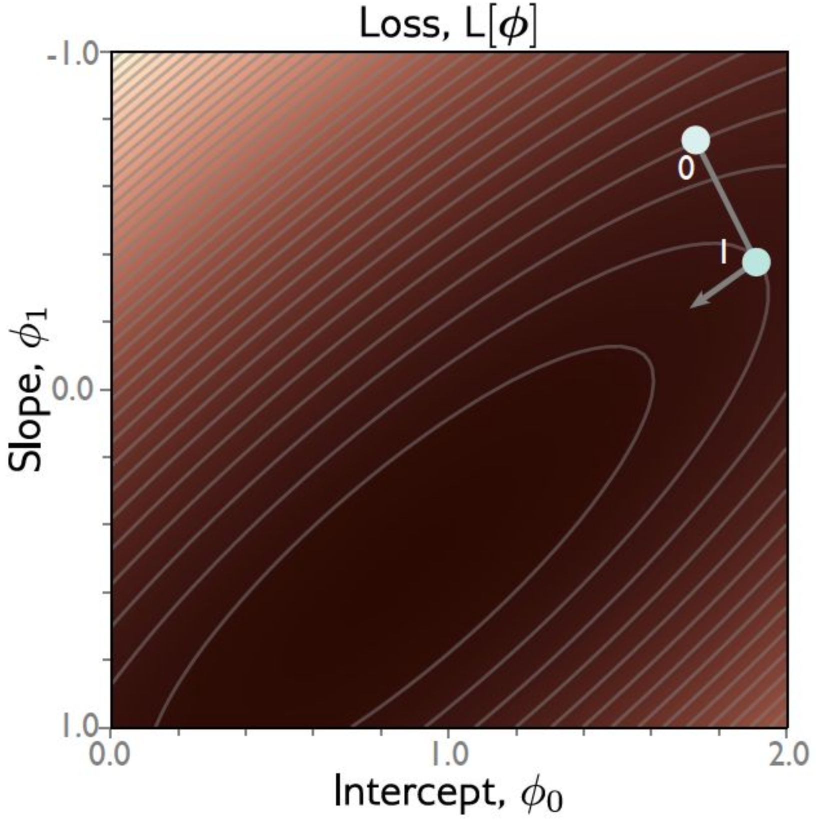
$$\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}$$

$$\frac{\partial \ell_i}{\partial \boldsymbol{\phi}} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

$$\boldsymbol{\phi} \longleftarrow \boldsymbol{\phi} - \alpha \frac{\partial L}{\partial \boldsymbol{\phi}}$$

α = step size

Gradient descent



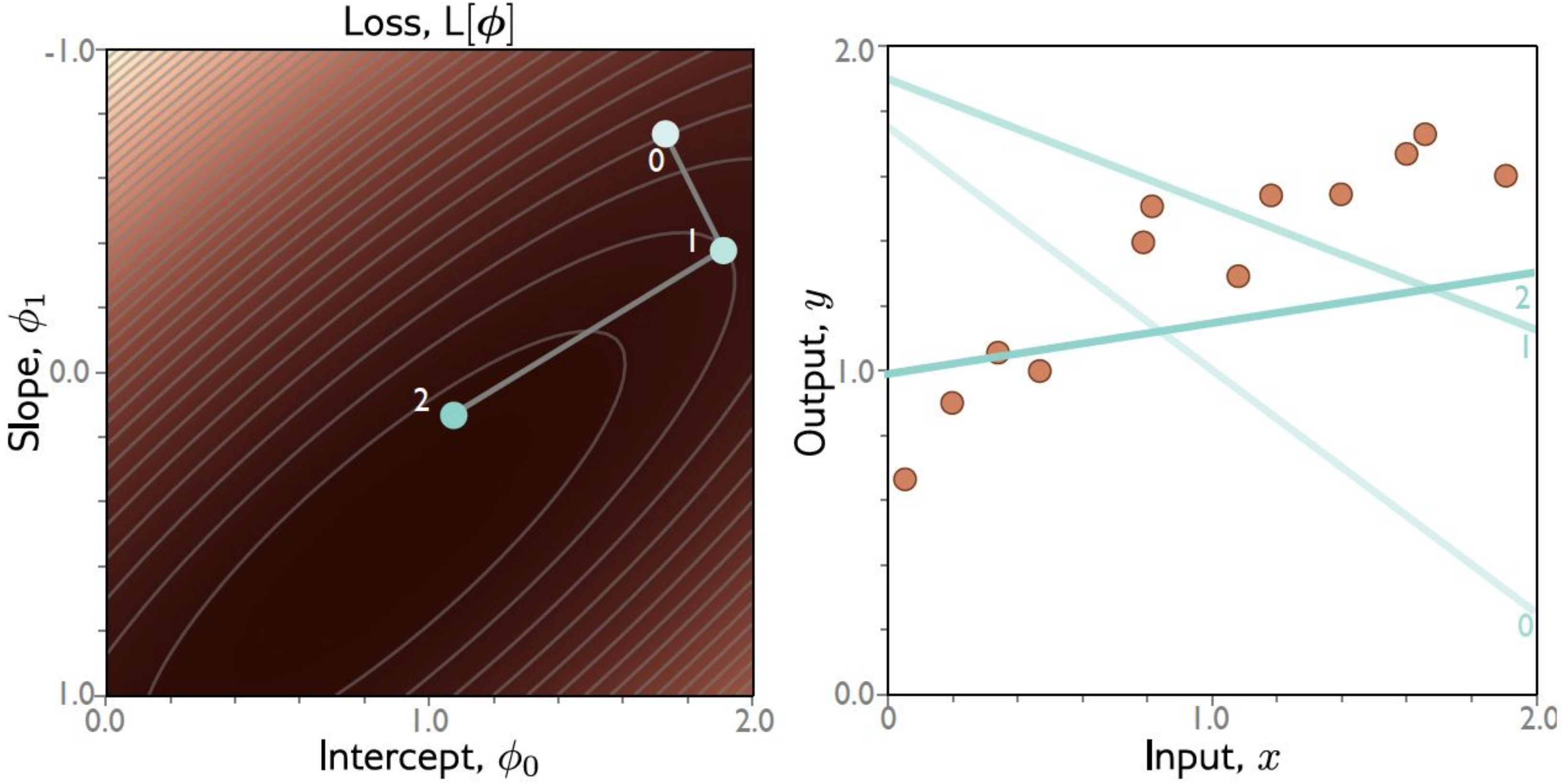
$$\frac{\partial L}{\partial \phi} = \frac{\partial}{\partial \phi}\sum_{i=1}^{I}\ell_i = \sum_{i=1}^{I}\frac{\partial \ell_i}{\partial \phi}$$

$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix}\frac{\partial \ell_i}{\partial \phi_0}\\ \frac{\partial \ell_i}{\partial \phi_1}\end{bmatrix} = \begin{bmatrix}2(\phi_0+\phi_1 x_i - y_i)\\ 2x_i(\phi_0+\phi_1 x_i - y_i)\end{bmatrix}$$

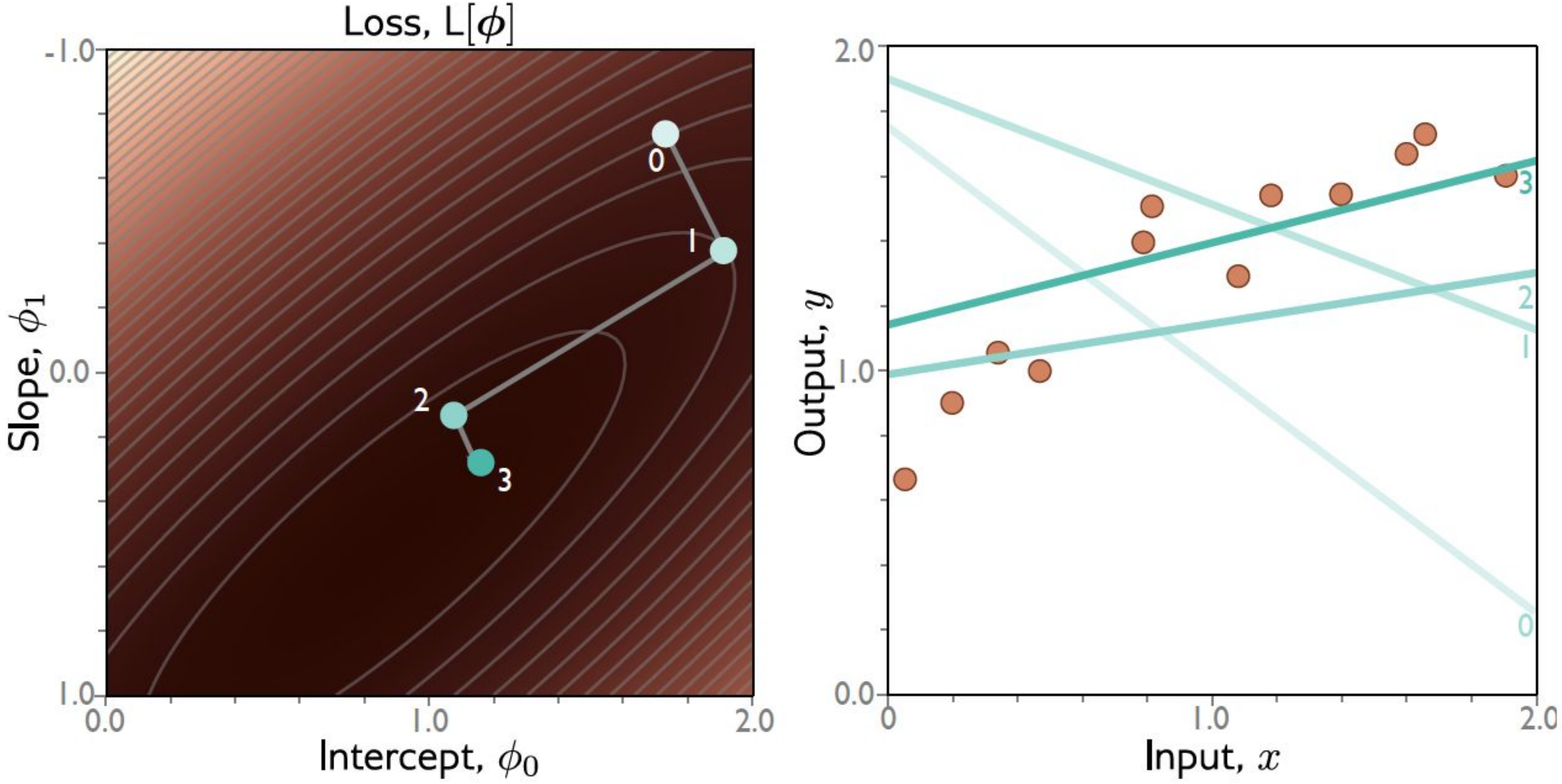
$$\phi \longleftarrow \phi - \alpha\frac{\partial L}{\partial \phi}$$

α = step size

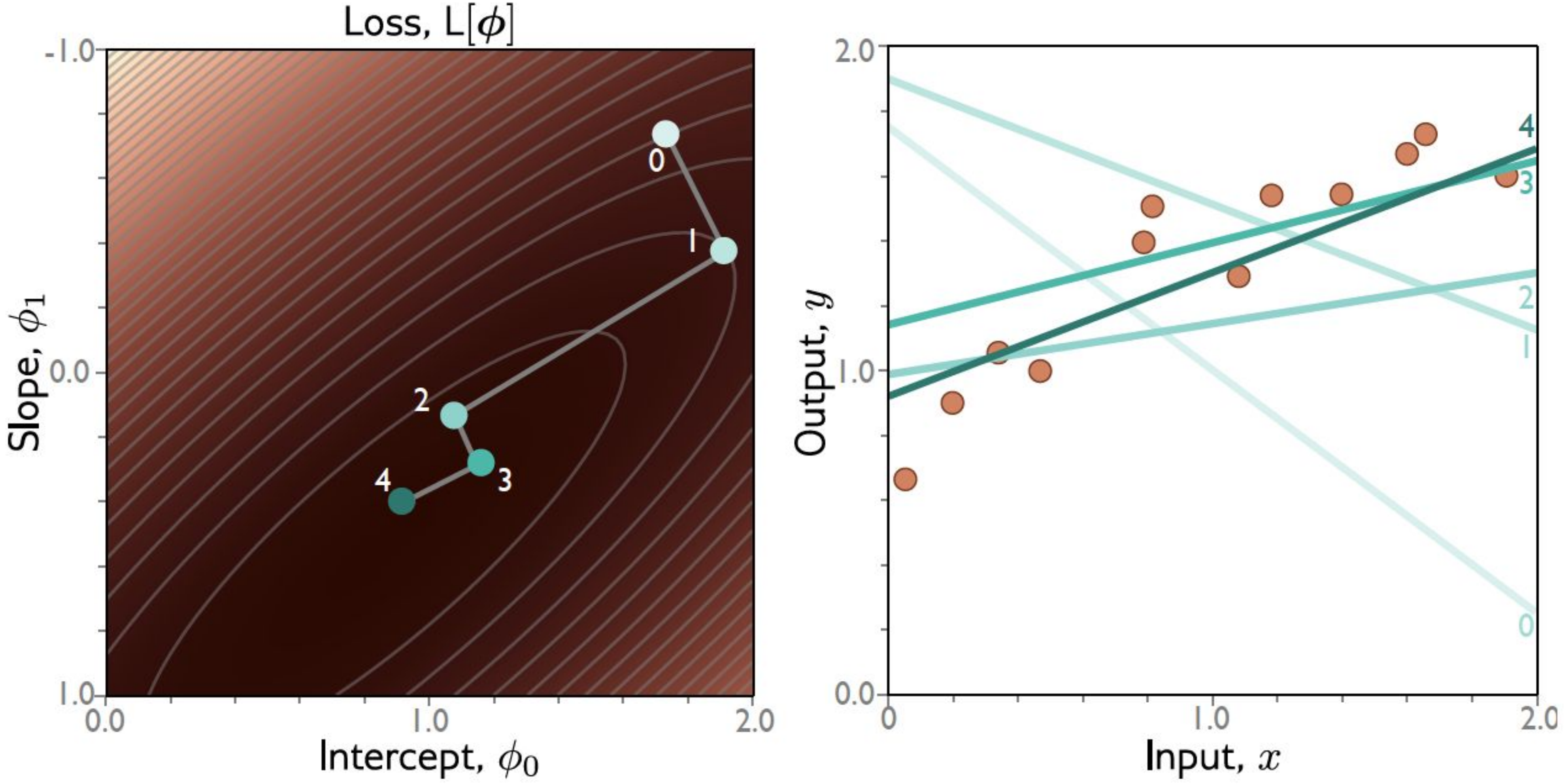
Gradient descent



Gradient descent

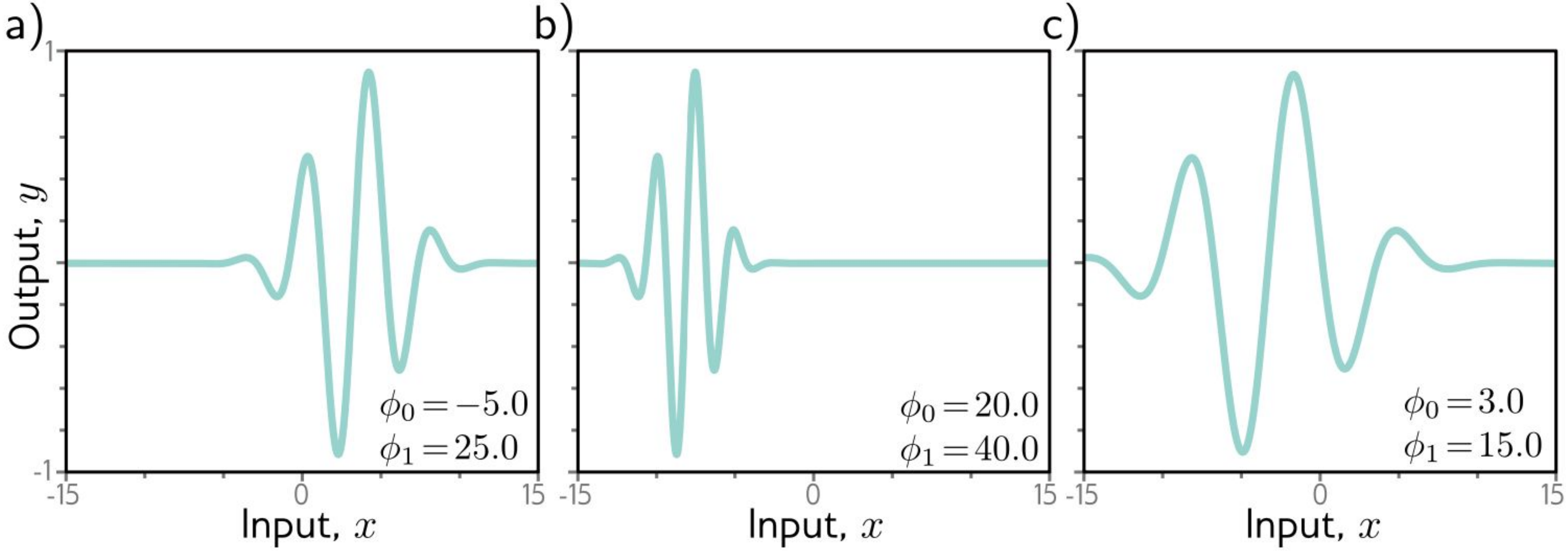


Gradient descent



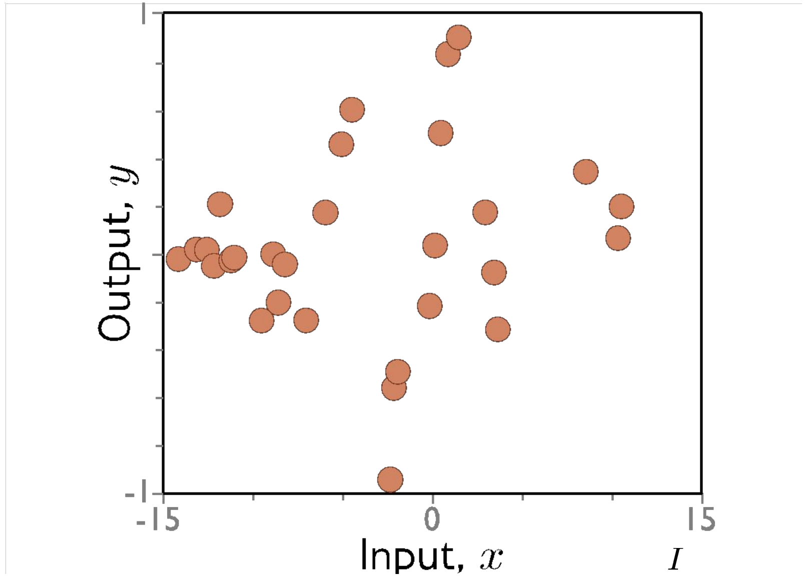
Non-convex case. Gabor model

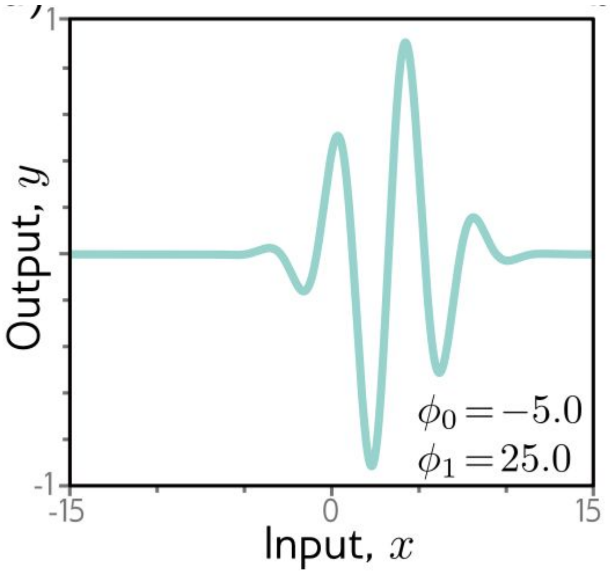
$$\mathrm{f}[x, \boldsymbol{\phi}] = \sin[\phi_0 + 0.06 \cdot \phi_1 x] \cdot \exp\left(-\frac{(\phi_0 + 0.06 \cdot \phi_1 x)^2}{8.0}\right)$$



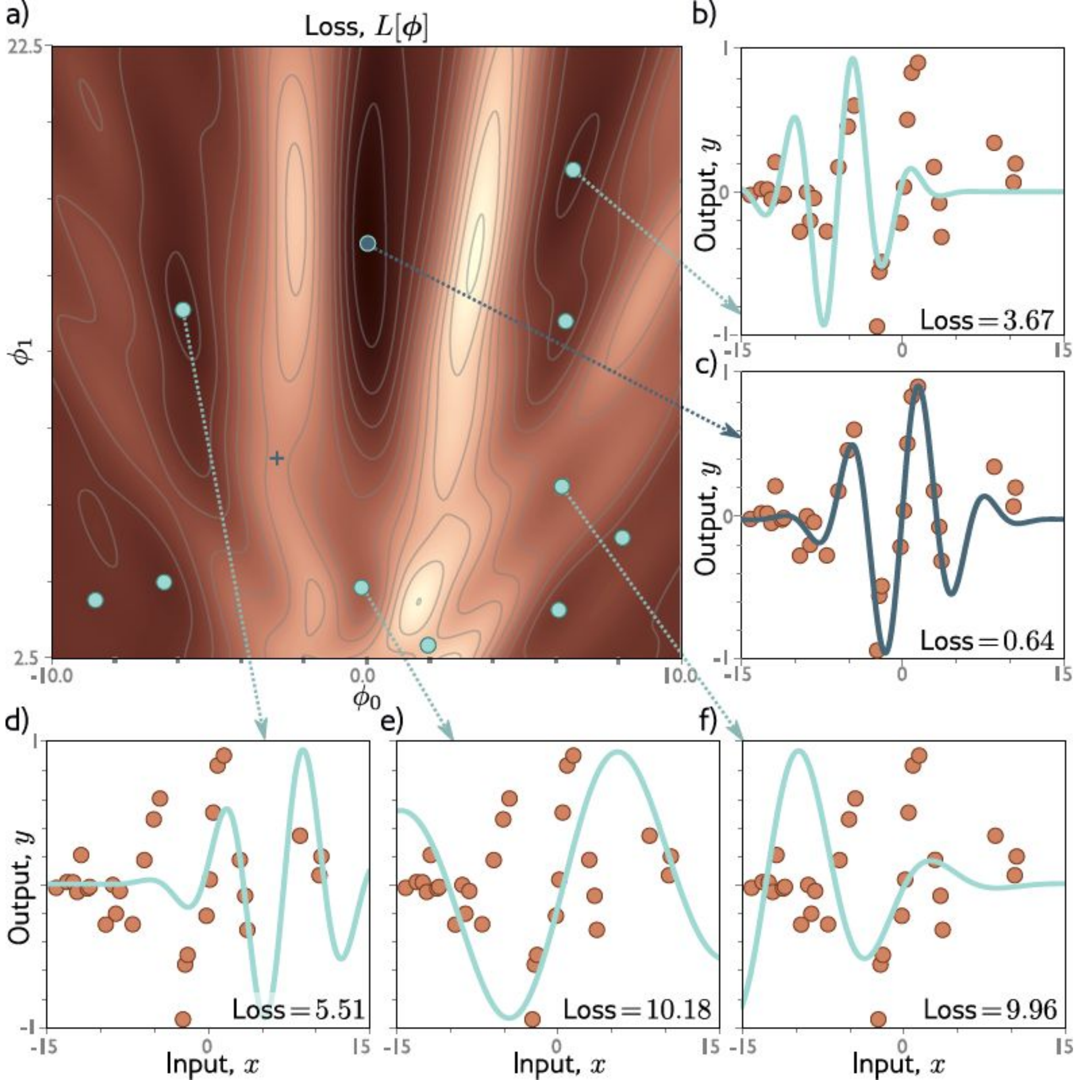
Gabor model

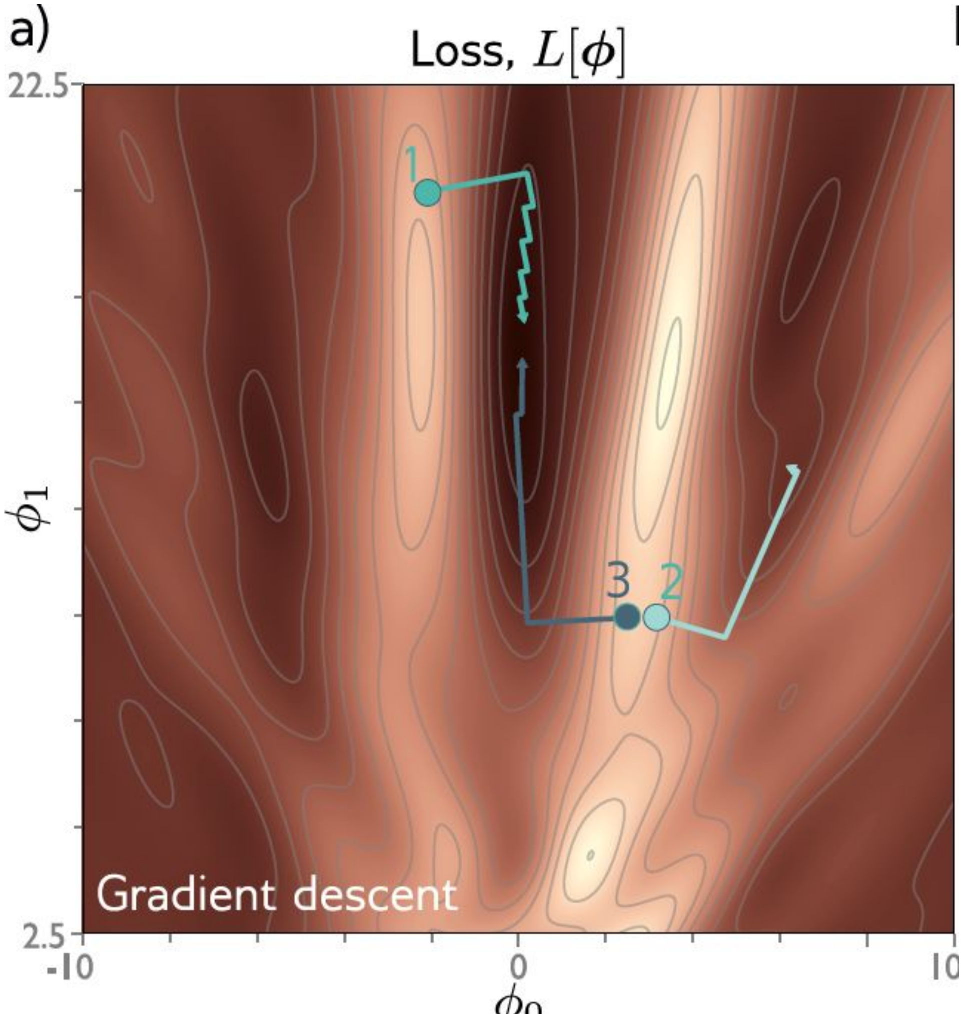
$$\mathrm{f}[x,\boldsymbol{\phi}] = \sin[\phi_0 + 0.06\cdot\phi_1 x]\cdot\exp\left(-\frac{(\phi_0+0.06\cdot\phi_1 x)^2}{8.0}\right)$$



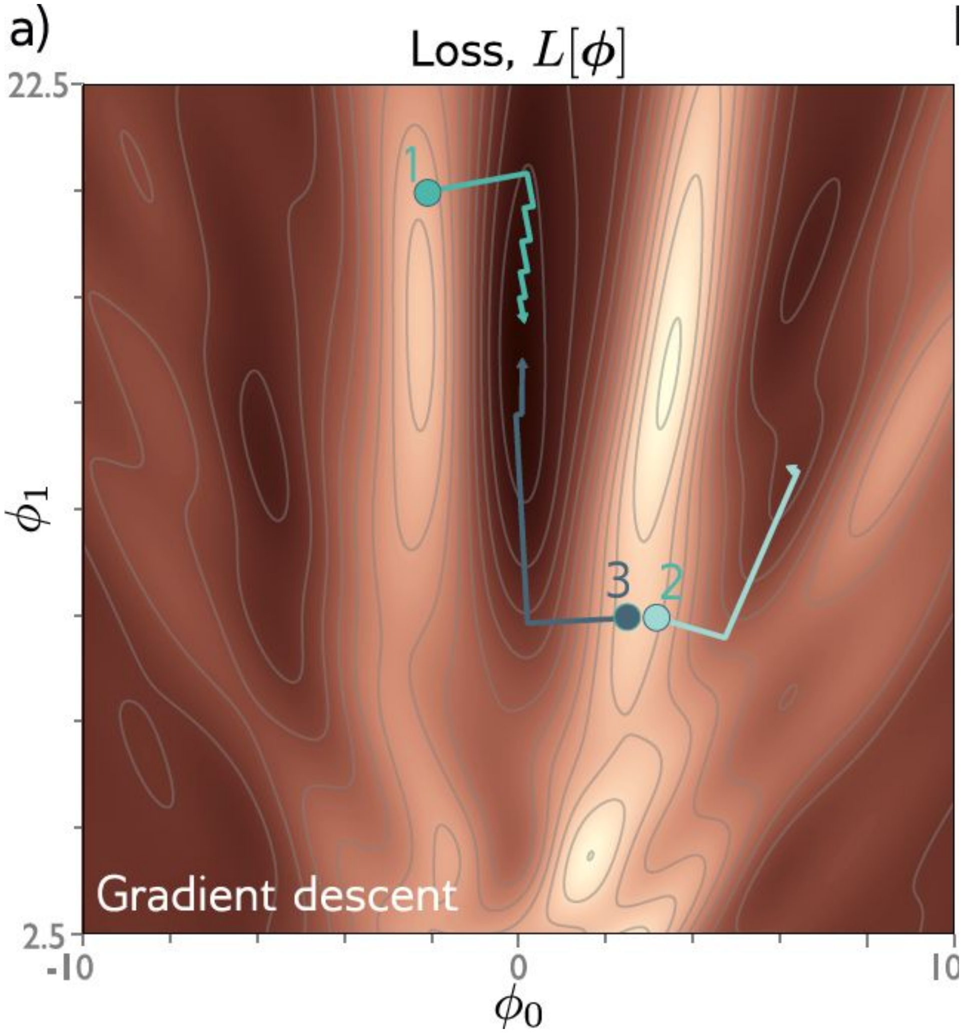


$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I}\left(\mathrm{f}[x_i,\boldsymbol{\phi}] - y_i\right)^2$$


a)
Loss, $L[\phi]$
22.5
2.5
ϕ_1
-10.0
0.0
10.0
ϕ_0
b)
c)
d)
e)
f)
Output, y
Input, x
1
0
-1
-15
0
15
Loss$=3.67$
Loss$=0.64$
Loss$=5.51$
Loss$=10.18$
Loss$=9.96$

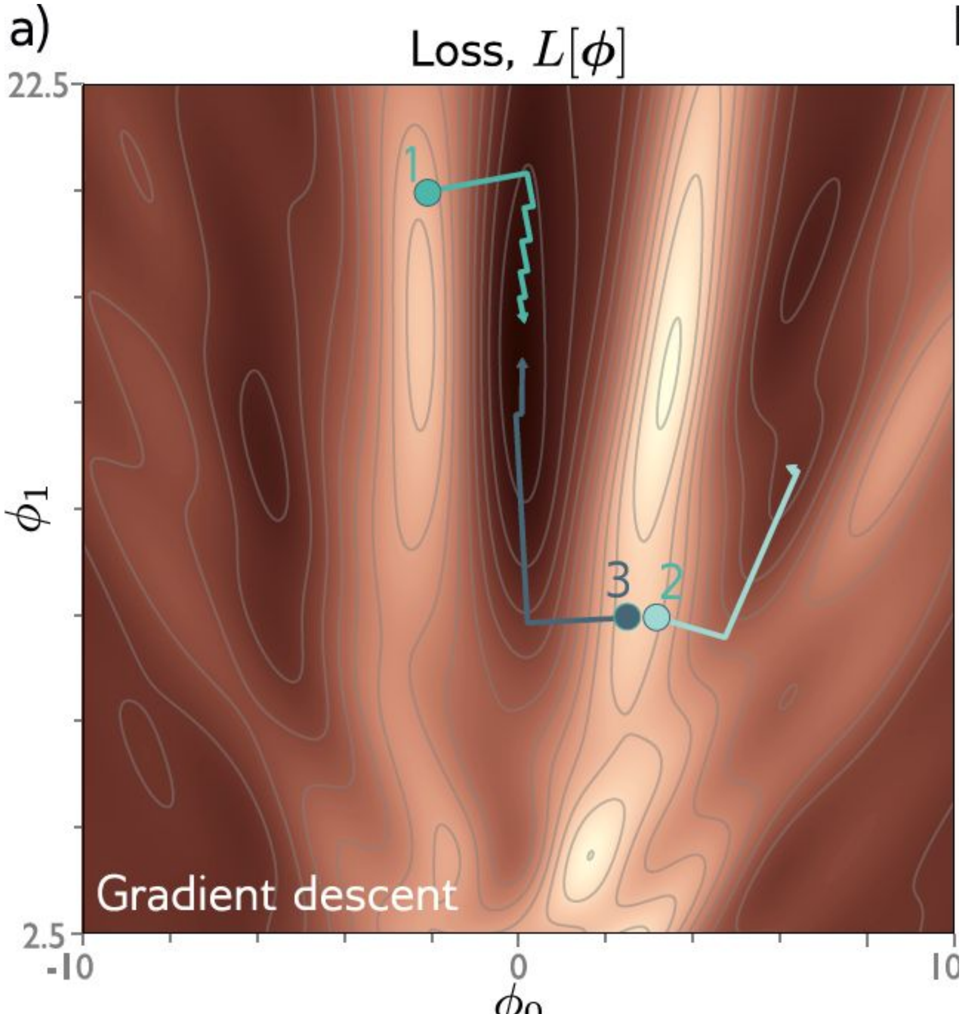


- Gradient descent gets to the global minimum if we start in the right “valley”
- Otherwise, descent to a local minimum
- Or get stuck near a saddle point



Solution: add noise!

- Stochastic gradient descent
- Compute gradient based on only a subset of points – a mini-batch
- Work through dataset sampling without replacement
- One pass though the data is called an epoch



Stochastic gradient descent

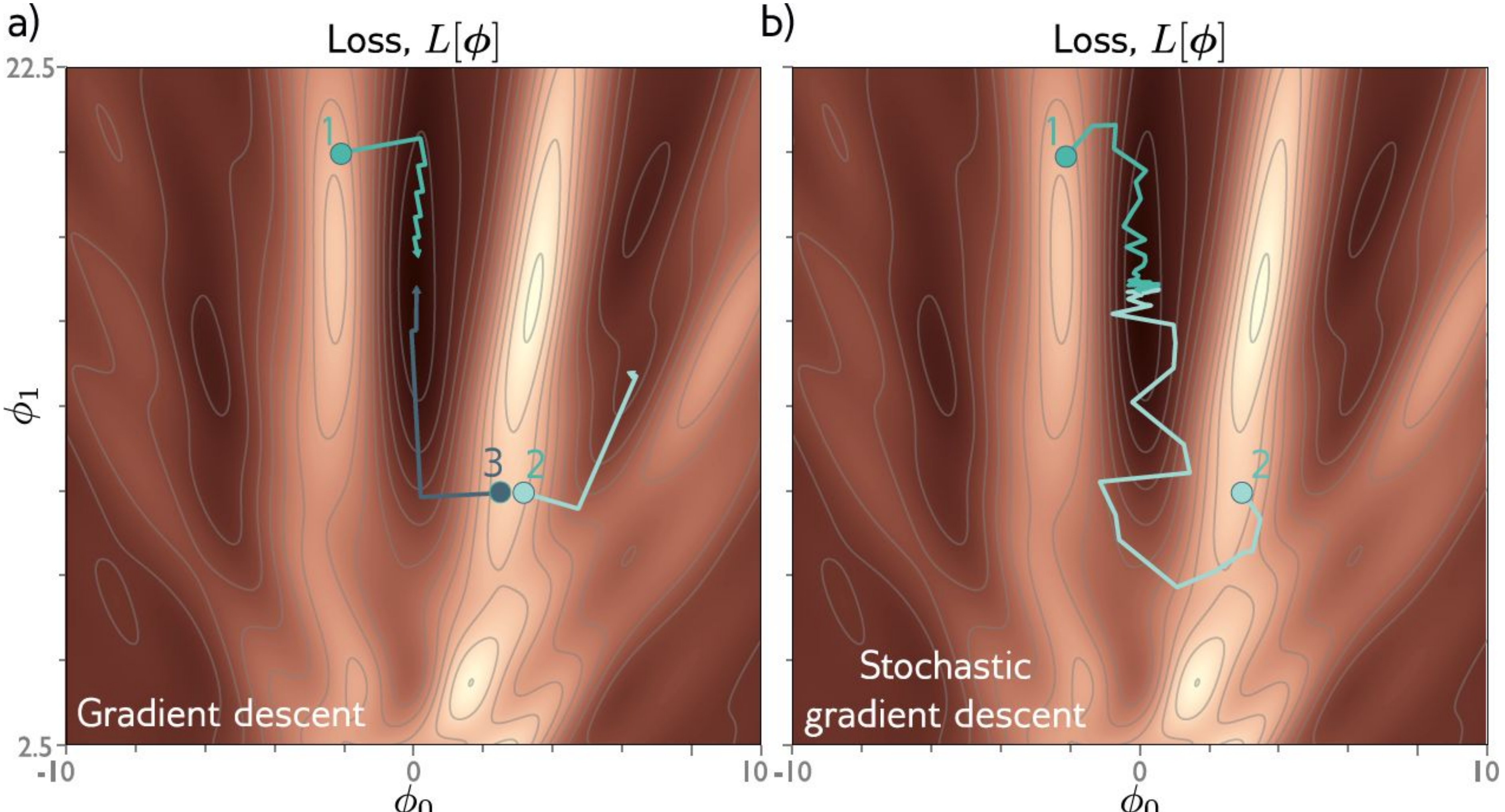
Before (full batch descent)

$$\phi_{t+1} \longleftarrow \phi_t - \alpha \sum_{i=1}^{I} \frac{\partial \ell_i[\phi_t]}{\partial \phi};$$

After (SGD)

$$\phi_{t+1} \longleftarrow \phi_t - \alpha \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t]}{\partial \phi};$$

Fixed learning rate α

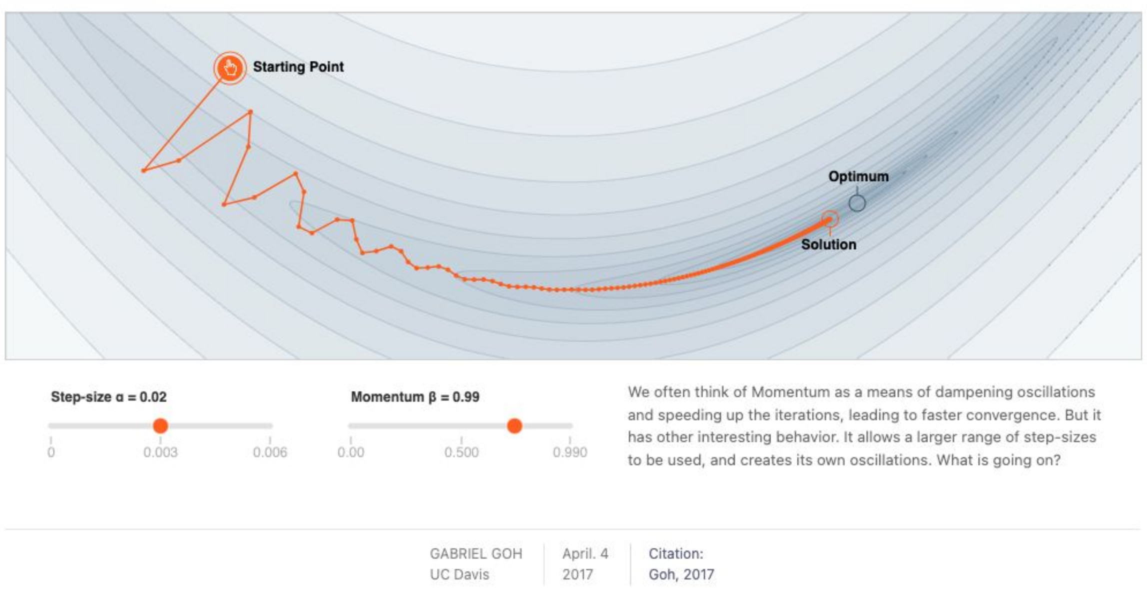

a)
Loss, $L[\phi]$
22.5
2.5
ϕ_1
1
2
3
-10
0
10
ϕ_0
Gradient descent
b)
Loss, $L[\phi]$
1
2
-10
0
10
ϕ_0
Stochastic gradient descent

Properties of SGD

- Can escape from local minima
- Adds noise, but still sensible updates as based on part of data
- Uses all data equally
- Less computationally expensive
- Seems to find better solutions

- Doesn’t converge in traditional sense
- Learning rate schedule – decrease learning rate over time

Momentum



We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

GABRIEL GOH
UC Davis

April. 4
2017

Citation:
Goh, 2017

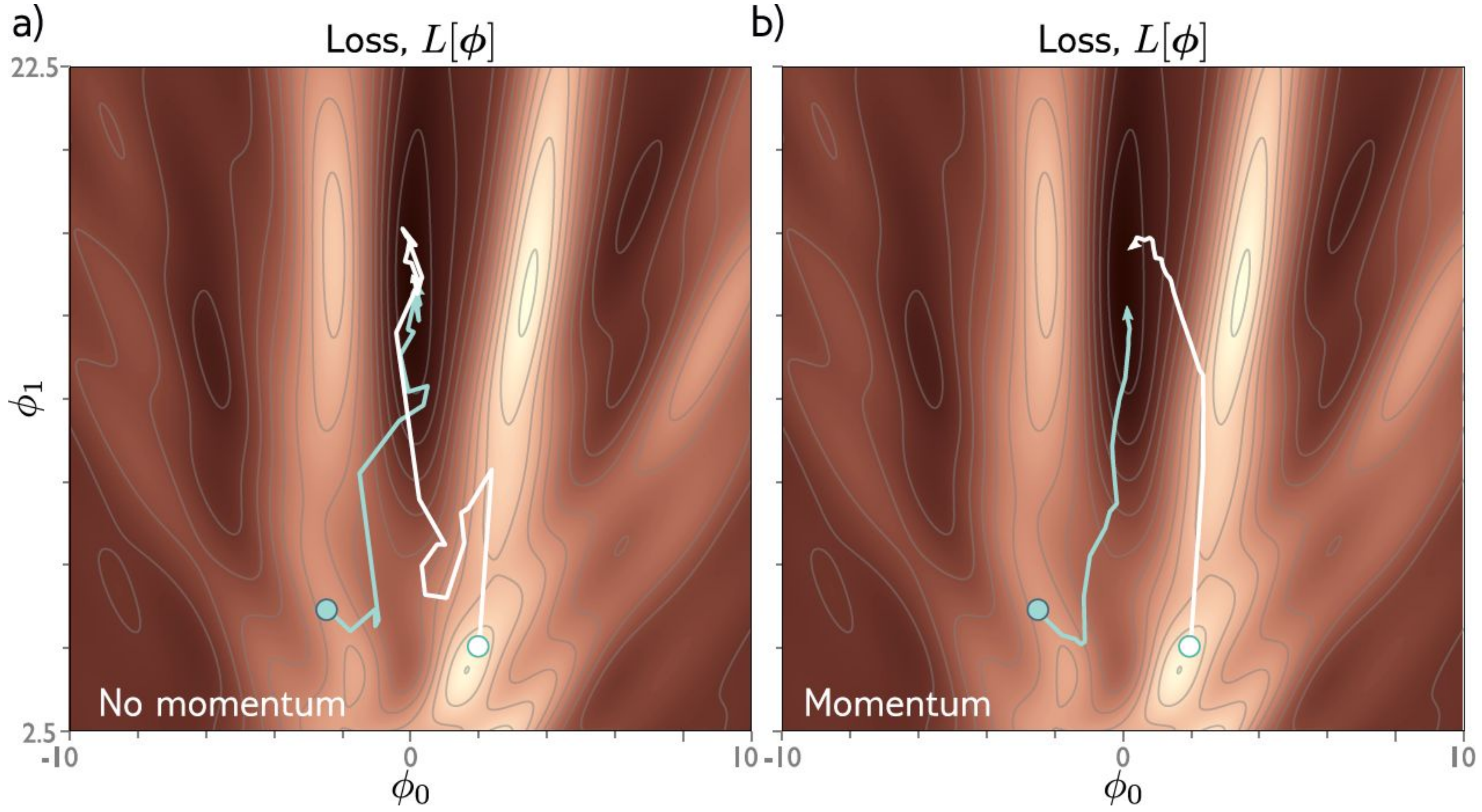
Goh, *Why momentum really works*

Momentum

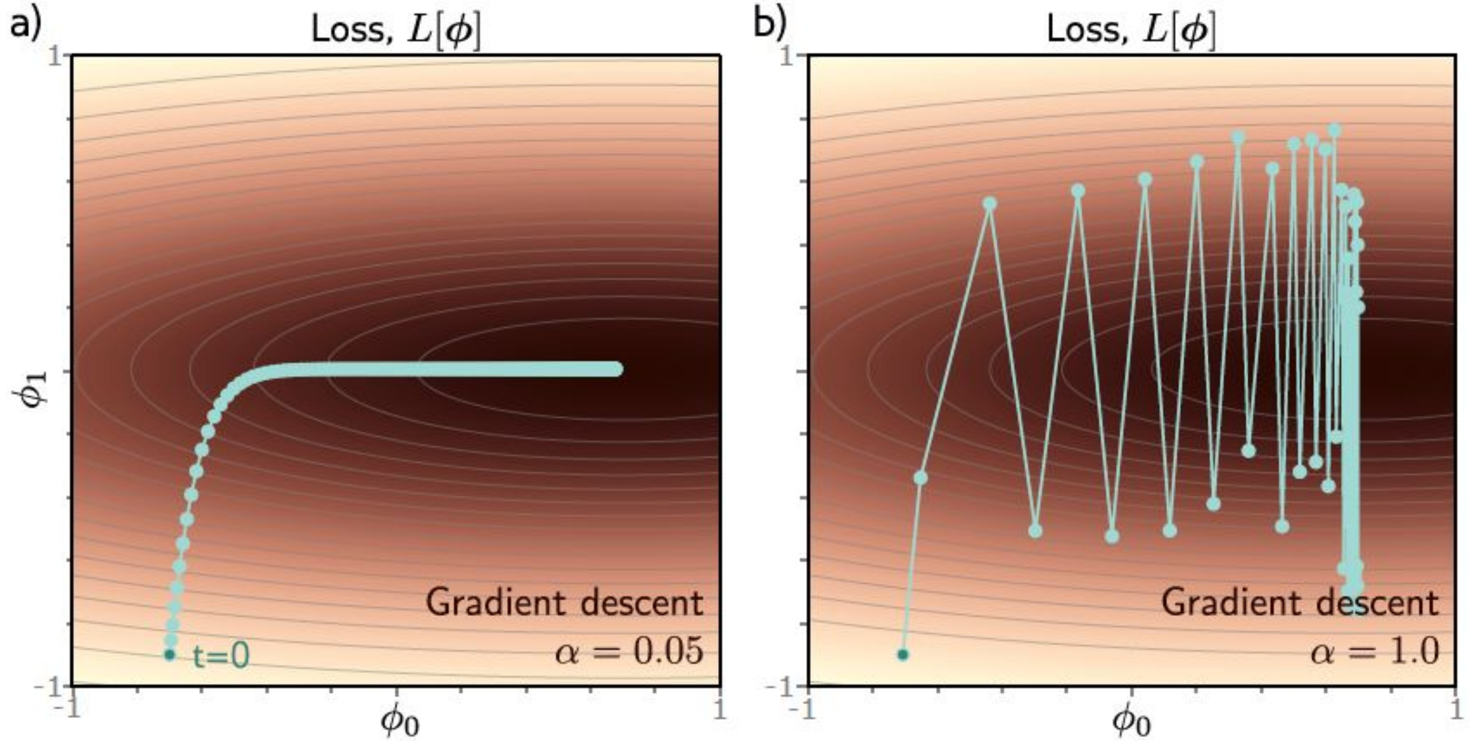
Weighted sum of this gradient and previous gradient

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1-\beta) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \mathbf{m}_{t+1}$$


a)
Loss, $L[\phi]$
b)
Loss, $L[\phi]$
22.5
2.5
ϕ_1
-10
0
10
-10
0
10
ϕ_0
ϕ_0
No momentum
Momentum

Adaptive moment estimation. Adam



Normalized gradients

Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}^2$$

Normalize:

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$

Normalized gradients

Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}^2$$

$$\mathbf{m}_{t+1} = \begin{bmatrix} 3.0 \\ -2.0 \\ 5.0 \end{bmatrix}$$

$$\mathbf{v}_{t+1} = \begin{bmatrix} 9.0 \\ 4.0 \\ 25.0 \end{bmatrix}$$

Normalize:

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$

$$\frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon} = \begin{bmatrix} 1.0 \\ -1.0 \\ 1.0 \end{bmatrix}$$

Normalized gradients

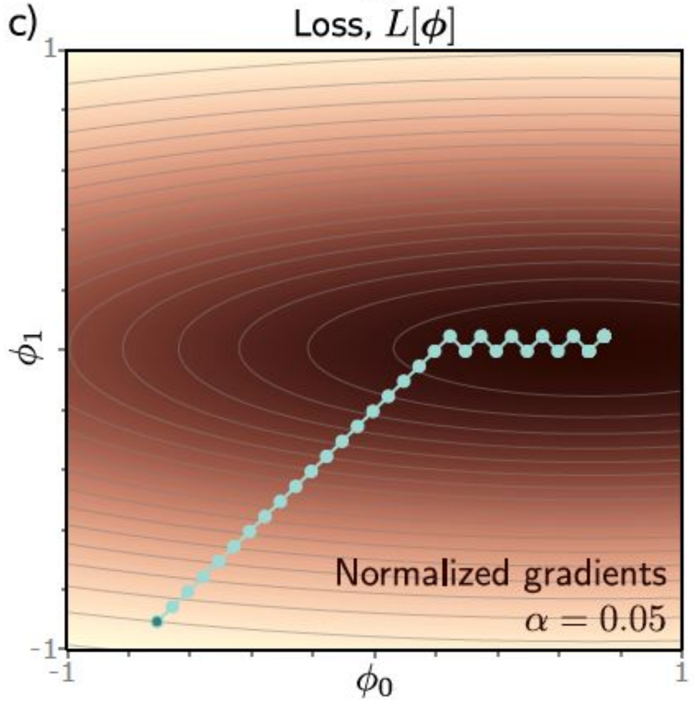
Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}^2$$

Normalize:

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$



Normalized gradients

Compute mean and pointwise squared gradients with momentum

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1-\beta)\frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \gamma \cdot \mathbf{v}_t + (1-\gamma)\left(\frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}\right)^2$$

Moderate near start of the sequence

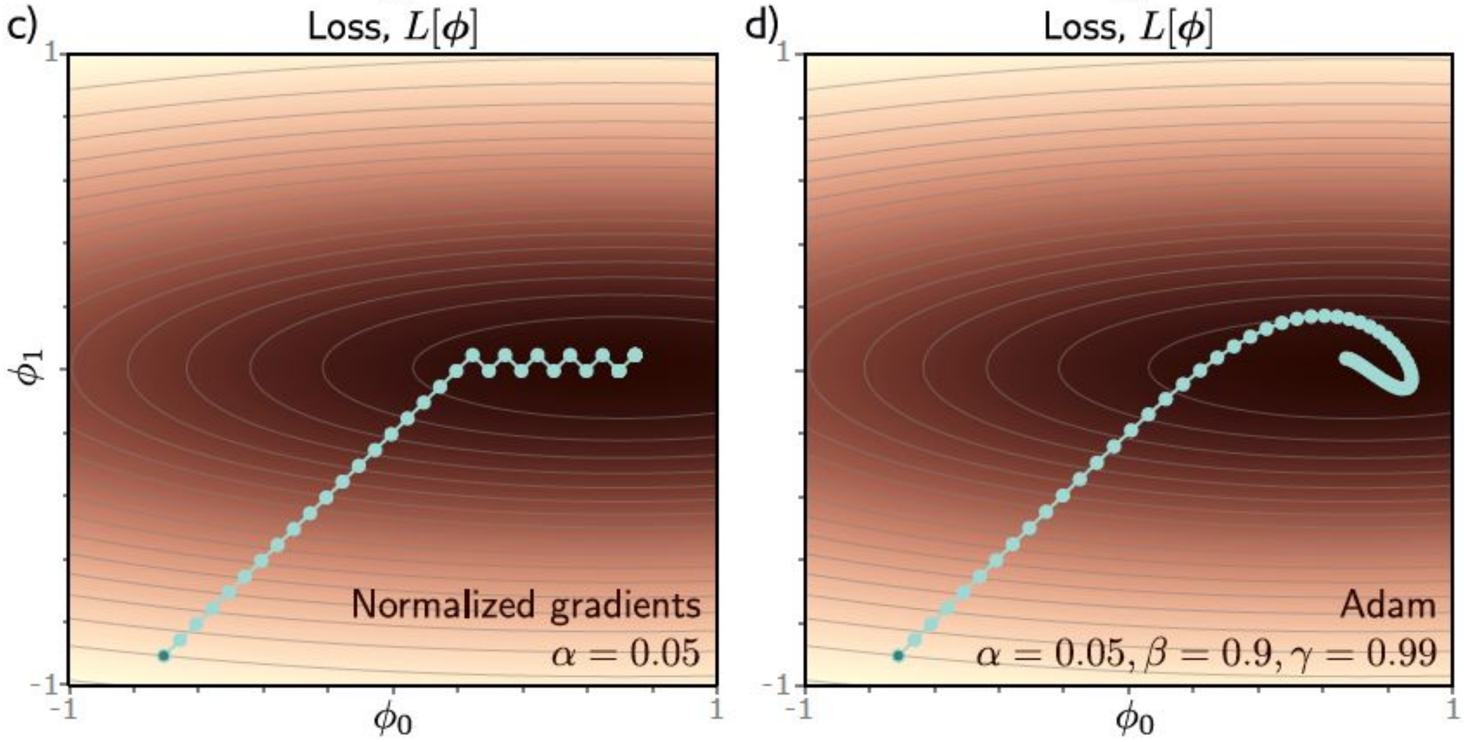
$$\tilde{\mathbf{m}}_{t+1} \leftarrow \frac{\mathbf{m}_{t+1}}{1-\beta^{t+1}}$$

$$\tilde{\mathbf{v}}_{t+1} \leftarrow \frac{\mathbf{v}_{t+1}}{1-\gamma^{t+1}}$$

Update the parameters

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\tilde{\mathbf{m}}_{t+1}}{\sqrt{\tilde{\mathbf{v}}_{t+1}} + \epsilon}$$

Adaptive moment estimation. Adam



RMSprop (precursor to Adam)

Compute ~~mean and~~ pointwise squared gradients with momentum

$$\cancel{\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1-\beta)\frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}}$$

$$\mathbf{v}_{t+1} \leftarrow \gamma \cdot \mathbf{v}_t + (1-\gamma)\left(\frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}\right)^2$$

Update the parameters

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\cancel{\tilde{\mathbf{m}}_{t+1}}}{\sqrt{\tilde{\mathbf{v}}_{t+1}} + \epsilon} \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

How to reach me:

florin.gogianu@gmail.com

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