Supervised Learning & Shallow Neural Networks

CAPS

A.I Study

허상준

tkdwnsdkdlel@gmail.com

2024.09.24

1. Supervised Learning

- Supervised Learning Overview
- Linear Regression
- Loss Function
- Training
- Testing
- Summary

2. Shallow Neural Networks

- Neural Networks
- Universal approximation Theorem
- Multivariate inputs and outputs
- General Case
- Summary
- Activation Function

3. Code

- Supervised Learning Code
- Shallow Neural Network 1
- Shallow Neural Network 2

1. Supervised Learning

Supervised Learning Overview

$$\mathbf{y} = \mathbf{f}[\mathbf{x}]. \longrightarrow \mathbf{y} = \mathbf{f}[\mathbf{x}, \phi].$$

Input : x
Output : y
Function : model

Input : x
Output : y
Parameter : ϕ
Function : model

Aims to build a model that an input x and outputs a prediction y
X and Y are vectors of a predetermined and fixed size -> Stuctured or Tabular Data
Parameter ϕ determines relationship between input and output

Supervised Learning Overview

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big[L\left[\boldsymbol{\phi}\right]\Big].$$

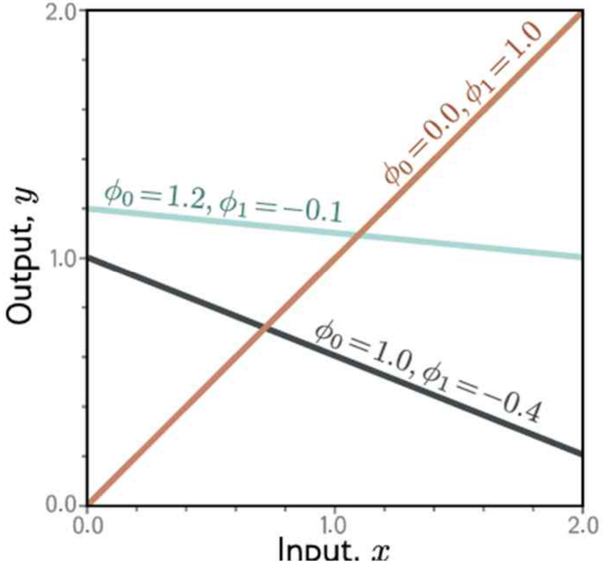
$$\begin{aligned} L[\phi] &= \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2 \\ &= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2 . \end{aligned}$$

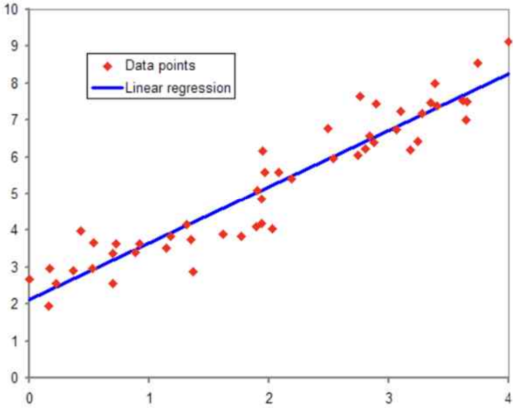
Training Model -> Attempt to find best parameters ϕ
Learn these parameters using a training dataset examples {X, Y}
Degree of mismatch = Loss Function $L[\phi]$
Minimalization of Loss $L[\phi]$ -> Best parameter ϕ

Linear Regression Model

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

$$\boldsymbol{\phi} = [\phi_0, \phi_1]^T$$





1D Linear Regression model – the relationship between input X and output Y as a straight line
Two Parameters, $\phi = [\mathrm{y - intercept, slope}]$

Loss

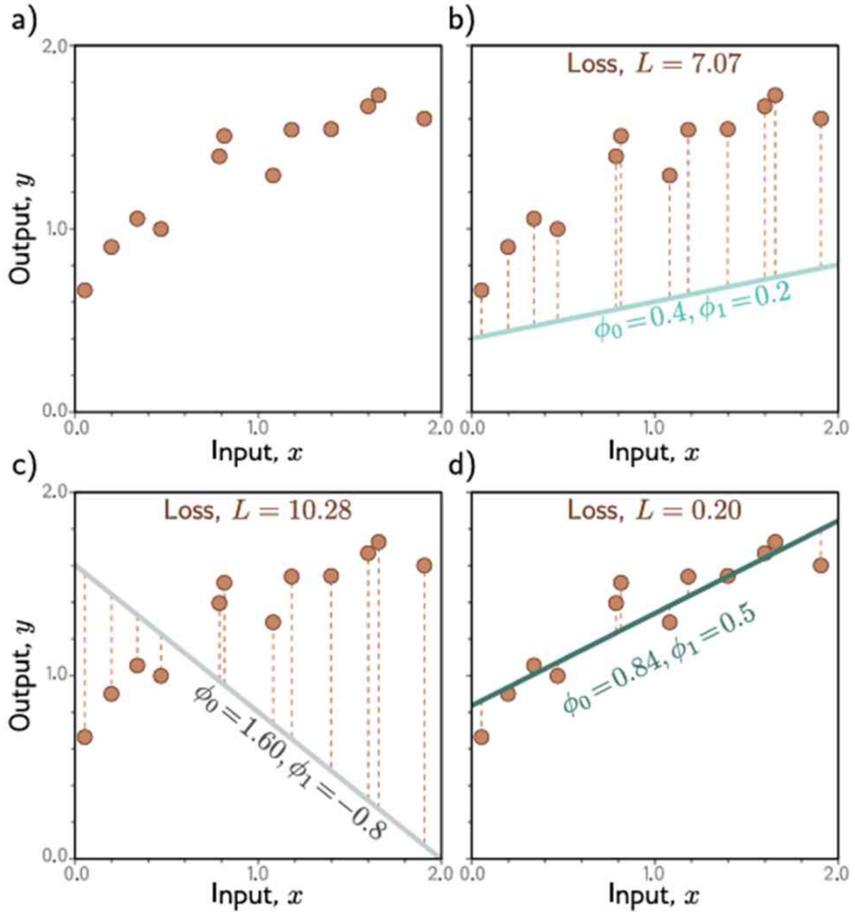
$$L[\phi] = \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2$$
$$= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2 .$$

$$least \quad - squares \quad Loss$$

Degree of mismatch = Loss function
Mismatch is the deviation between predictions and truth outputs
The best parameters ϕ minimize the Loss function

Training Model = Search for Best Parameter ϕ

Loss

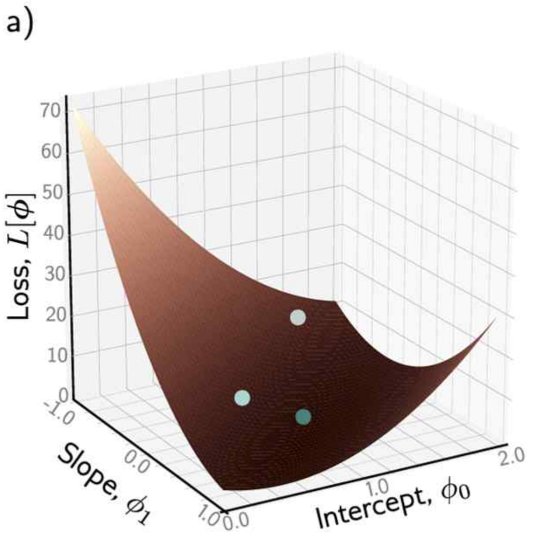


Orange Point -> Training data
Line -> model prediction
Orange Dashed Line -> loss function

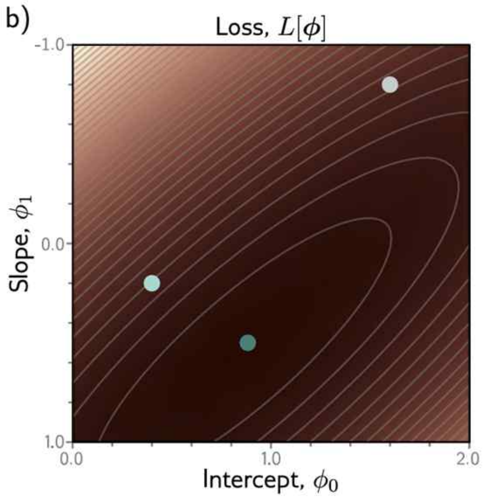
a) – Only Training data
b), c) – Large loss between prediction and truth
d) – best selection parameter

-> d) parameter are the optimal parameters

Loss



Loss function in 3D
X,Y -> parameter ϕ
Z -> loss



Loss function in 2D with contour line
X,Y -> parameter
Z -> contour line

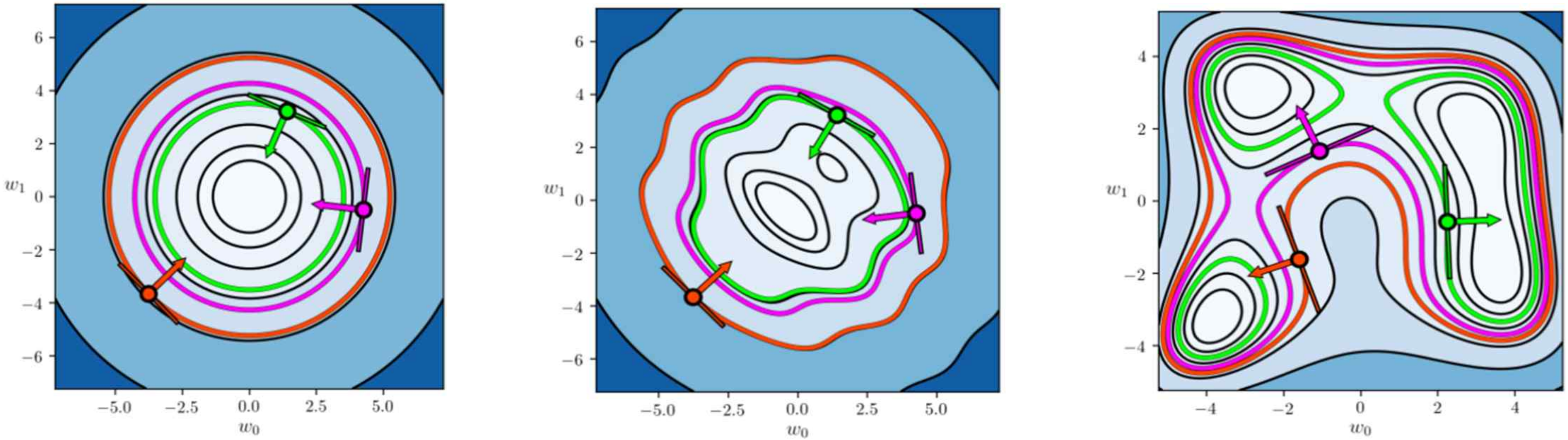
a) – Only Training data
b), c) – Large loss between prediction and truth
d) – best selection parameter

-> d) parameter are the optimal parameters

Training

Randomly choose parameters -> Calculate loss function -> Update parameters that minimize loss

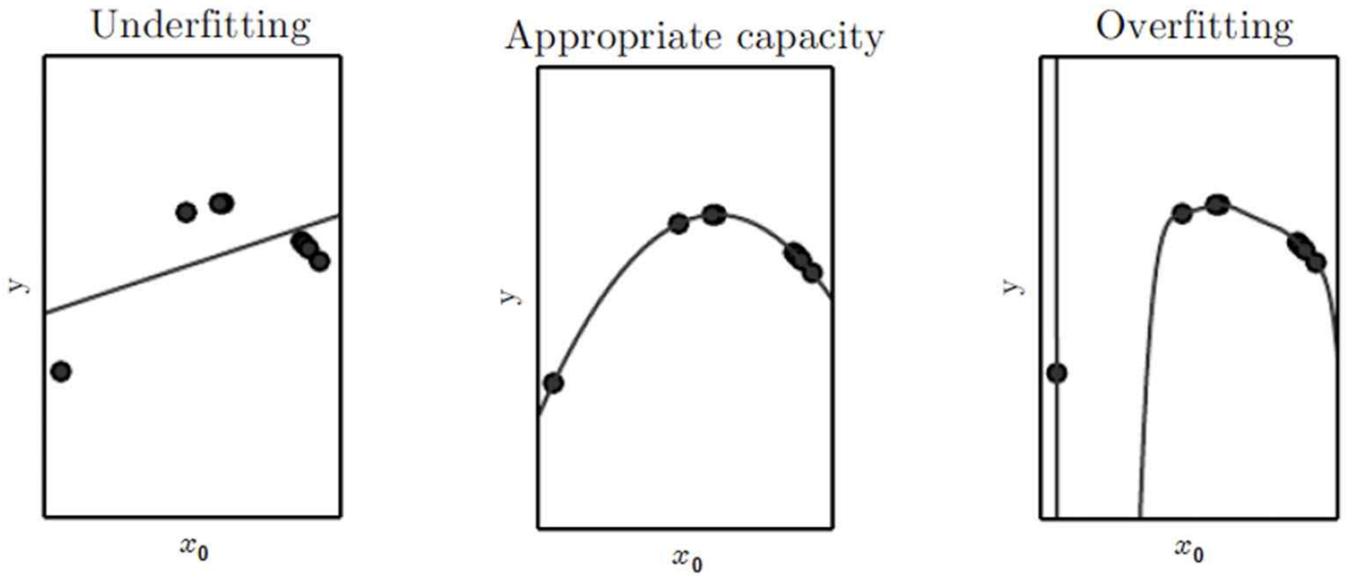
One way : measure gradient of the surface and take a step in the direction that most steeply downhill = gradient descent



X,Y -> Parameter ω , Z -> Loss

Testing

Testing with Test data(Not training data)
Underfitting = Can’t capture the true relationship
Overfitting = describe statistical peculiarities of the training data, lead to unusual predictions

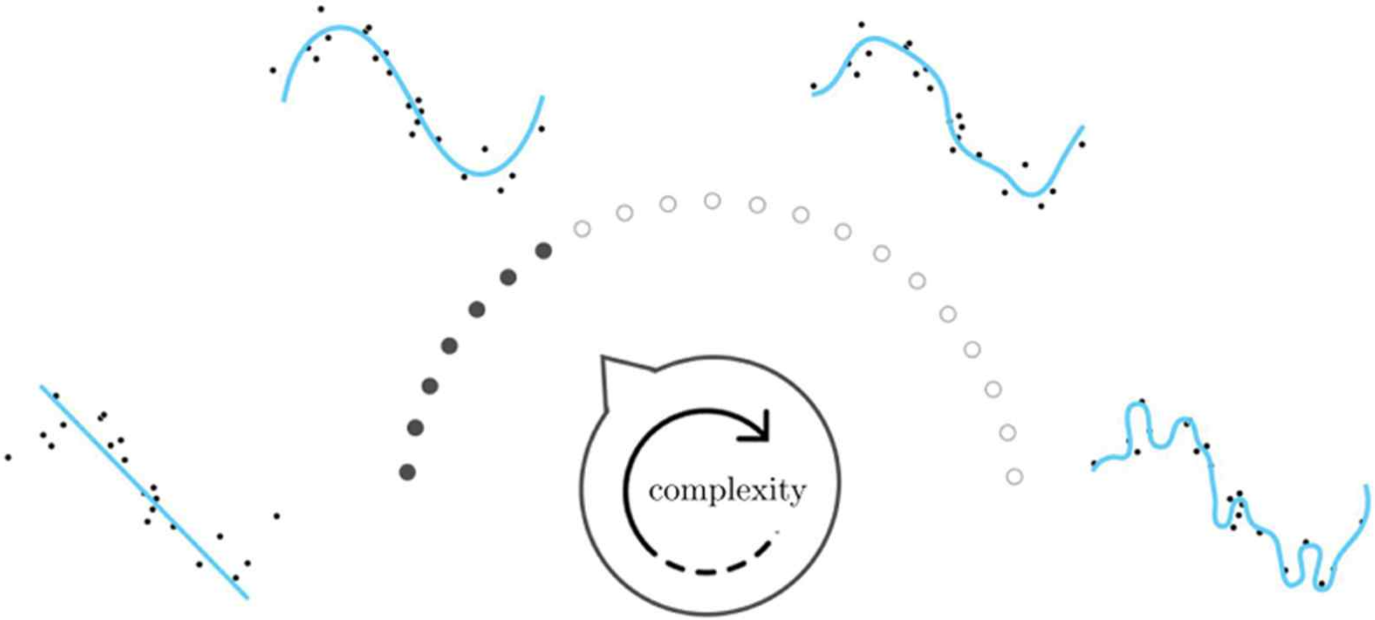


A linear function

A quadratic function

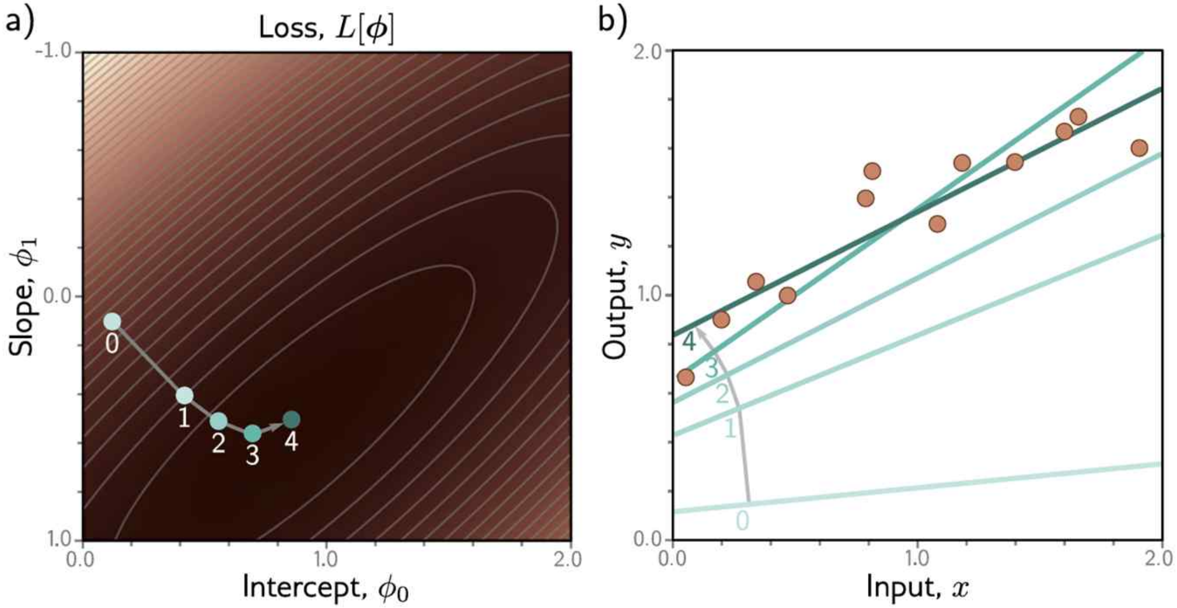
A polynomial of degree 9

Testing



Underfitting ⟷ Overfitting

Summary



Supervised learning model is a function that relates inputs X to outputs Y
Define Loss function over a training data {X, Y}
Search for the parameters that minimize the loss
Evaluate the model on a different set of test data

2. Shallow Neural Networks

Neural Networks

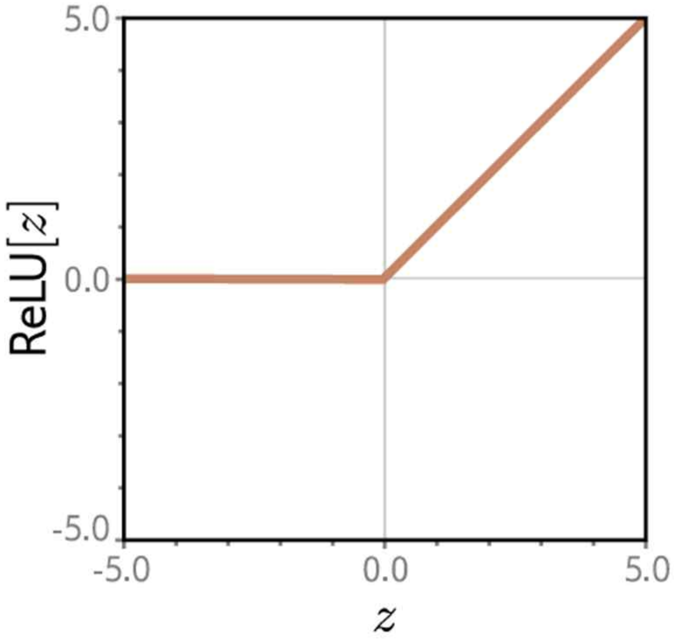
$$\begin{aligned} y &= \mathrm{f}[x, \boldsymbol{\phi}] \\ &= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x]. \end{aligned}$$

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

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

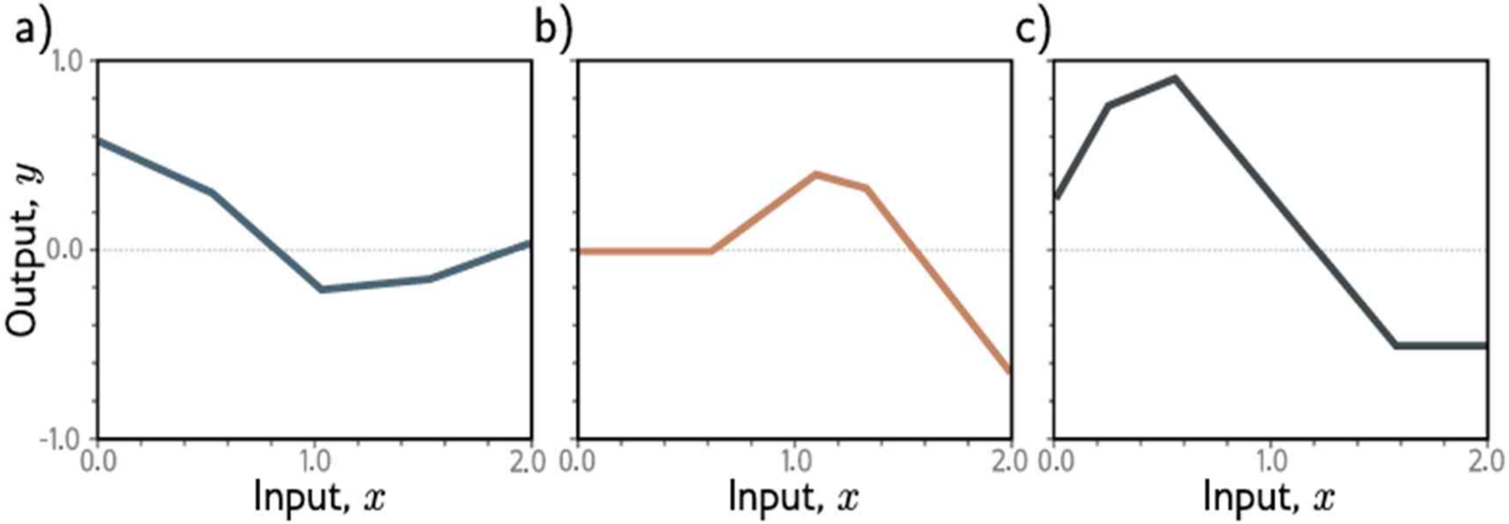
- Shallow Neural Networks = multivariate inputs X to multivariate outputs Y
- Three linear functions of the input data -> pass three results through an a activation function $a[z]$

Neural Networks



- If the input is less than zero -> return zero
  - Else -> return the input unchanged

Neural Networks



- Family of functions defined by linear regression

Neural Networks

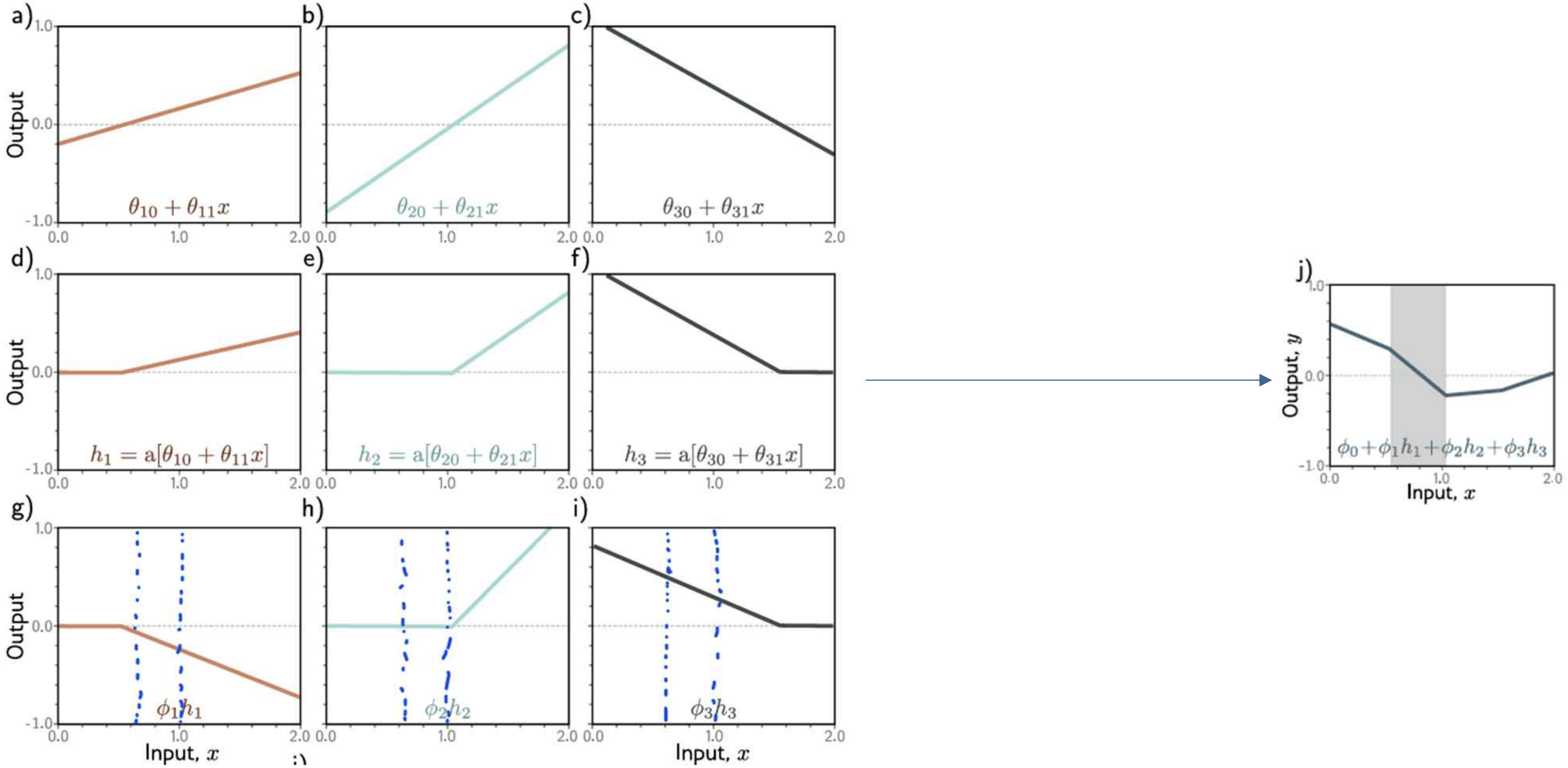
$$\theta_{10} + \theta_{11}x$$
$$\theta_{20} + \theta_{21}x$$
$$\theta_{30} + \theta_{31}x$$

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

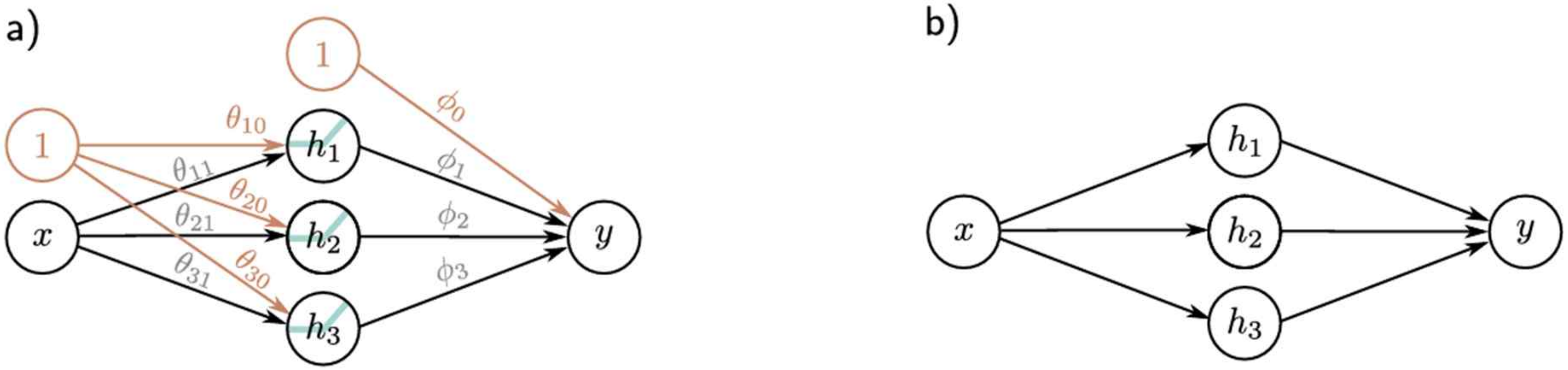
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3.$$

- h1, h2, h3 = hidden unit
- Linear regression -> activation function -> combining hidden units

Neural Networks



Neural Networks



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

- left is input data X, right is prediction Y

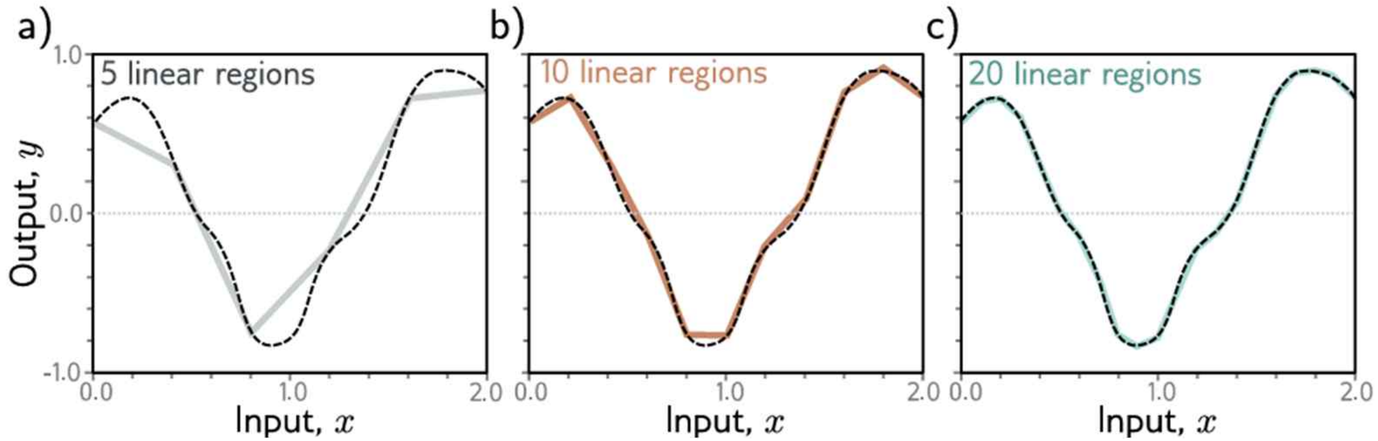
Universal approximation theorem

$$\begin{aligned} h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\ h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \longrightarrow \quad h_d = \mathrm{a}[\theta_{d0} + \theta_{d1}x], \\ h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x], \end{aligned}$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3. \longrightarrow \quad y = \phi_0 + \sum_{d=1}^{D} \phi_d h_d.$$

- Generalize and consider the case with D hidden units
- The number of hidden unit = measure of the network capacity

Universal approximation theorem



Dashed line = Truth output data y

- Add more hidden unit -> Add another linear region
- Regions become more numerous -> represent smaller sections of the function -> well approximated by a line

Multivariate inputs and outputs

$$\mathbf{x} = [x_1, x_2, \ldots, x_{D_i}]^T$$

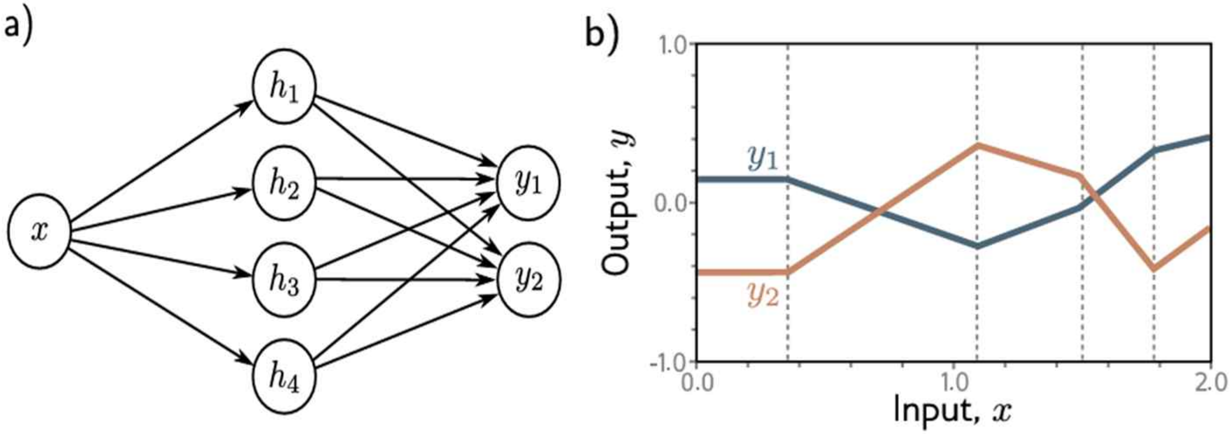
$$\mathbf{y} = [y_1, y_2, \ldots, y_{D_o}]^T$$

- More general case
- The network maps multivariate input X to multivariate input Y

Multivariate inputs and outputs

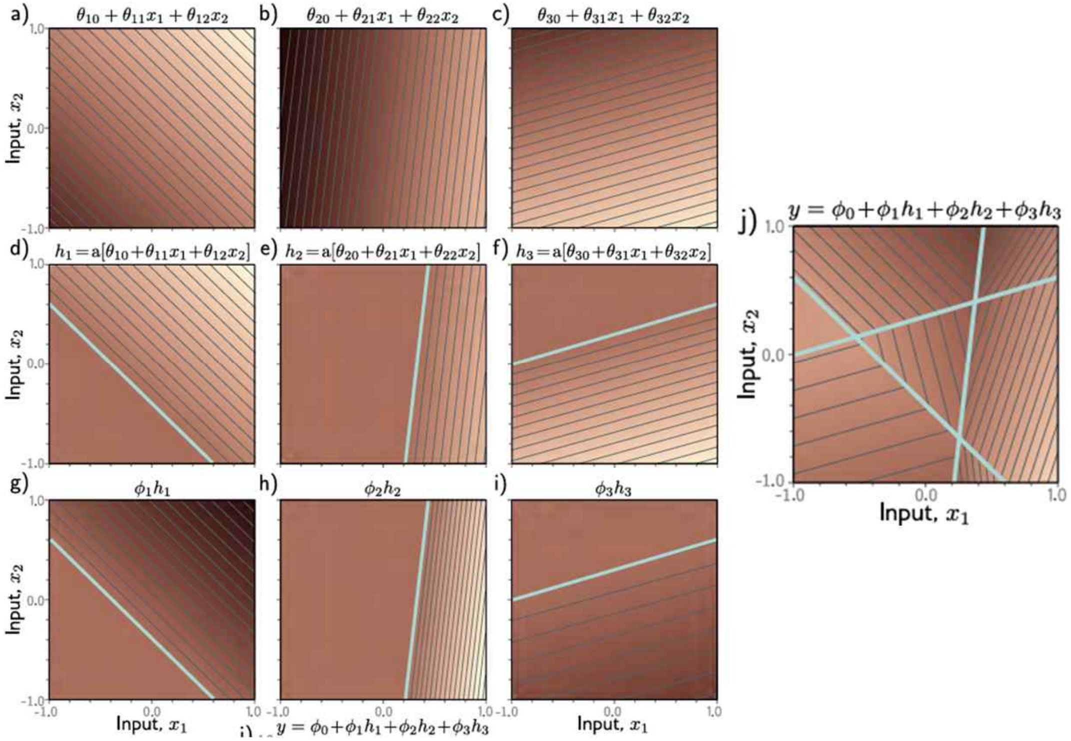
$$\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x] \\
h_4 &= \mathrm{a}[\theta_{40} + \theta_{41}x]
\end{aligned}$$

$$\begin{aligned}
y_1 &= \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4 \\
y_2 &= \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4.
\end{aligned}$$



- This network produces two piecewise linear functions
  - 4 hidden units = 4 joints

Multivariate inputs and outputs



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

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

- Two inputs, three hidden units, one outputs
  - 3 hidden units = 3 joints

General Case

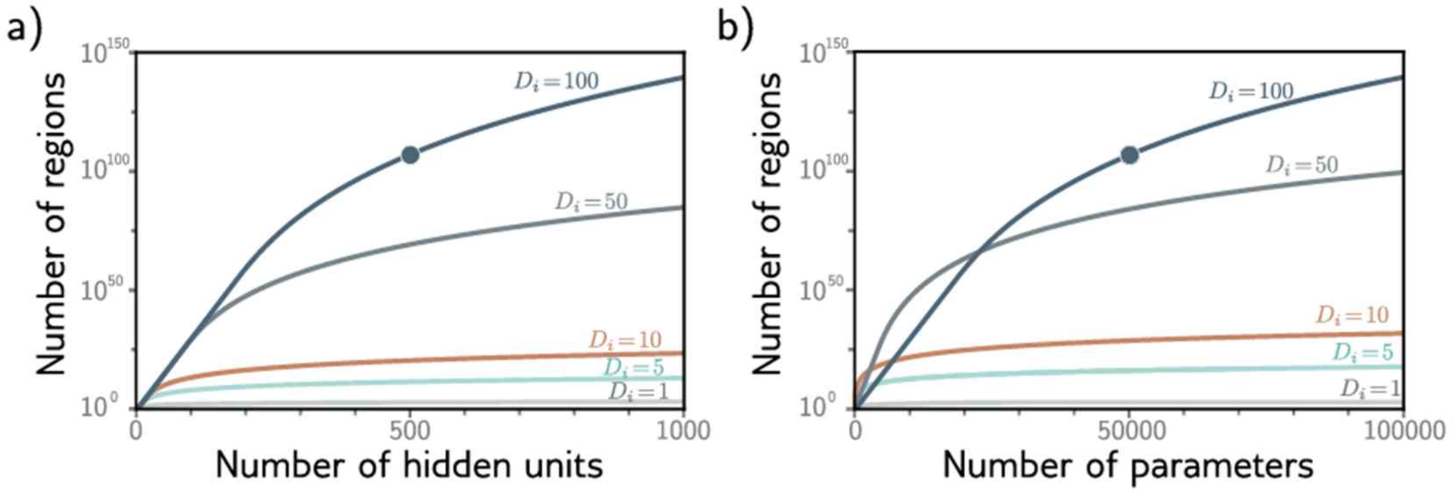
$$h_d = \mathrm{a}\left[\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i\right]$$

- Calculate hidden units
- Manipulate hidden units

$$y_j = \phi_{j0} + \sum_{d=1}^{D} \phi_{jd} h_d$$

- Calculate outputs
- Manipulate parameters

General Case

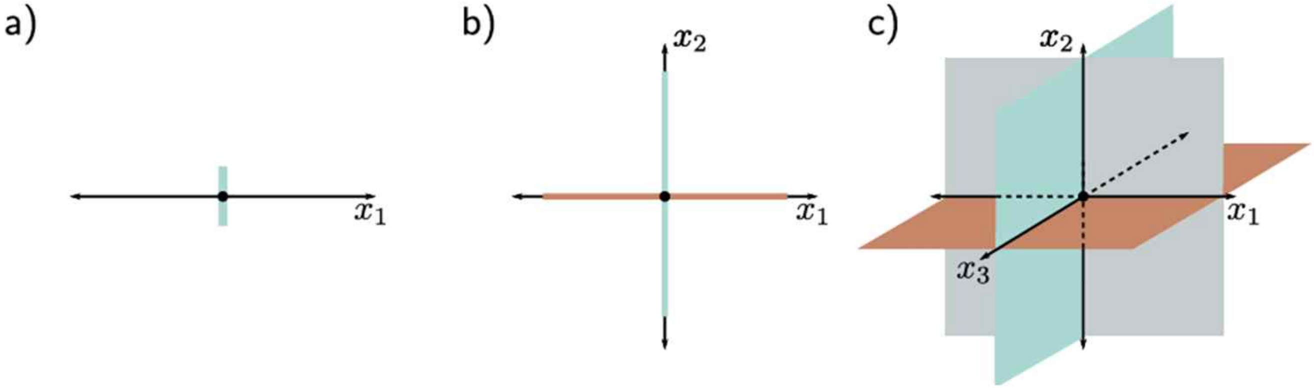


$$h_d = \mathrm{a}\left[\theta_{d0} + \sum_{i=1}^{D_i}\theta_{di}x_i\right]$$

$$y_j = \phi_{j0} + \sum_{d=1}^{D}\phi_{jd}h_d$$

- Increasing Number of Regions
- High dimensions -> rapidly increases

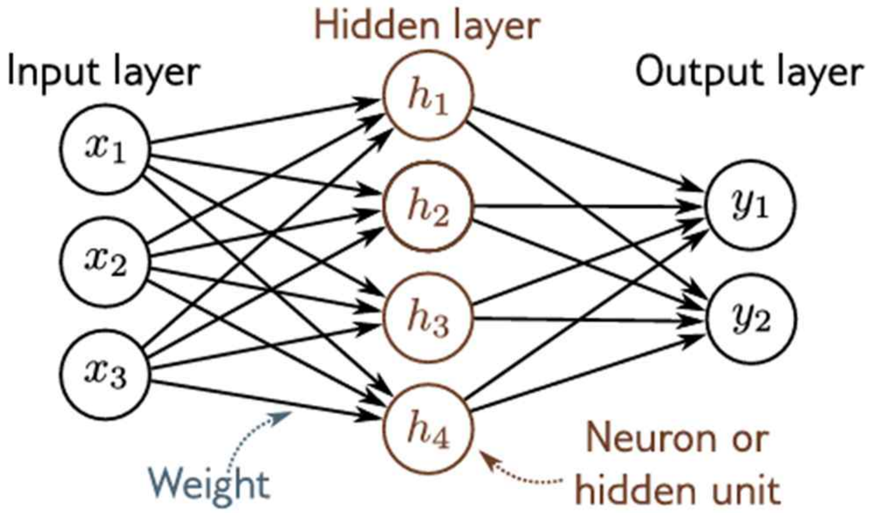
General Case



- $Input\ Dimensions = D^i, hidden\ unit = D$

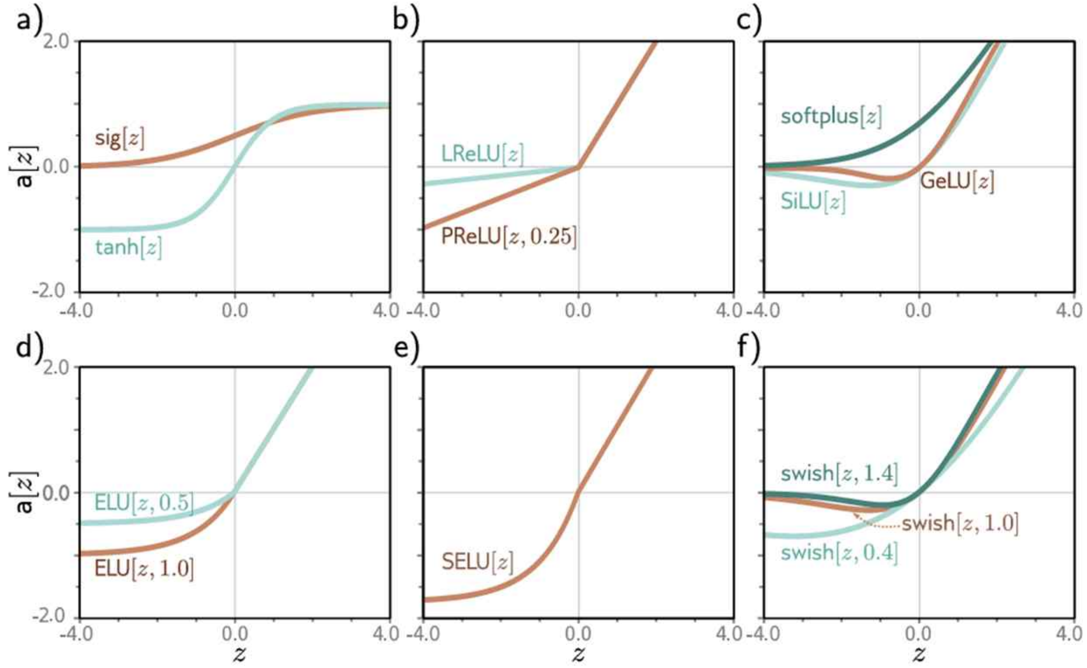
  $\rightarrow 2^{D^i} < linear\ region < 2^D$

Terminology



- Input layer = input data X
- Hidden layer = hidden units or neurons
- Output layer = output data Y

Activation Function



- Dying ReLU problems : incoming weights is locally flat so we cannot walk downhill

Contents

1. Deep neural networks

- Composing neural networks
- From composing networks to deep networks
- Deep Neural networks
- Matrix Notation
- Shallow vs. deep neural networks

2. Summary

3. Code

- Composing networks
- Clipping functions
- Deep networks

1. Deep neural networks

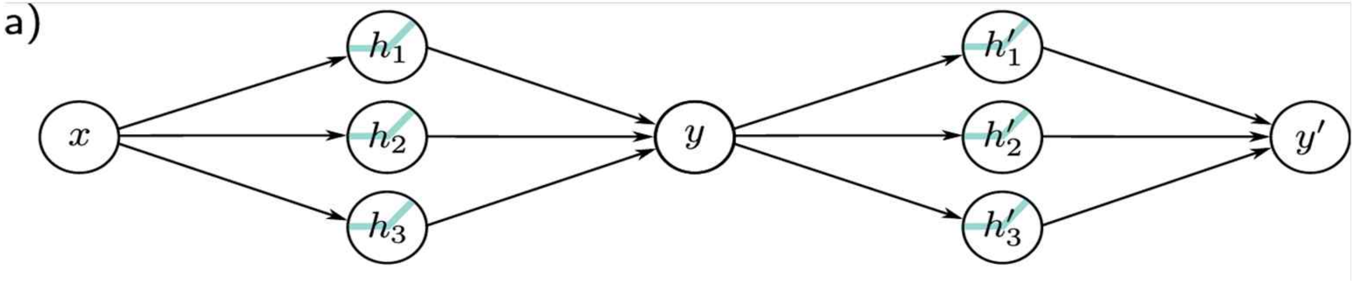
Composing neural networks

First Network

$$h_1 = a[\theta_{10} + \theta_{11}x]$$
$$h_2 = a[\theta_{20} + \theta_{21}x]$$
$$h_3 = a[\theta_{30} + \theta_{31}x]$$
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

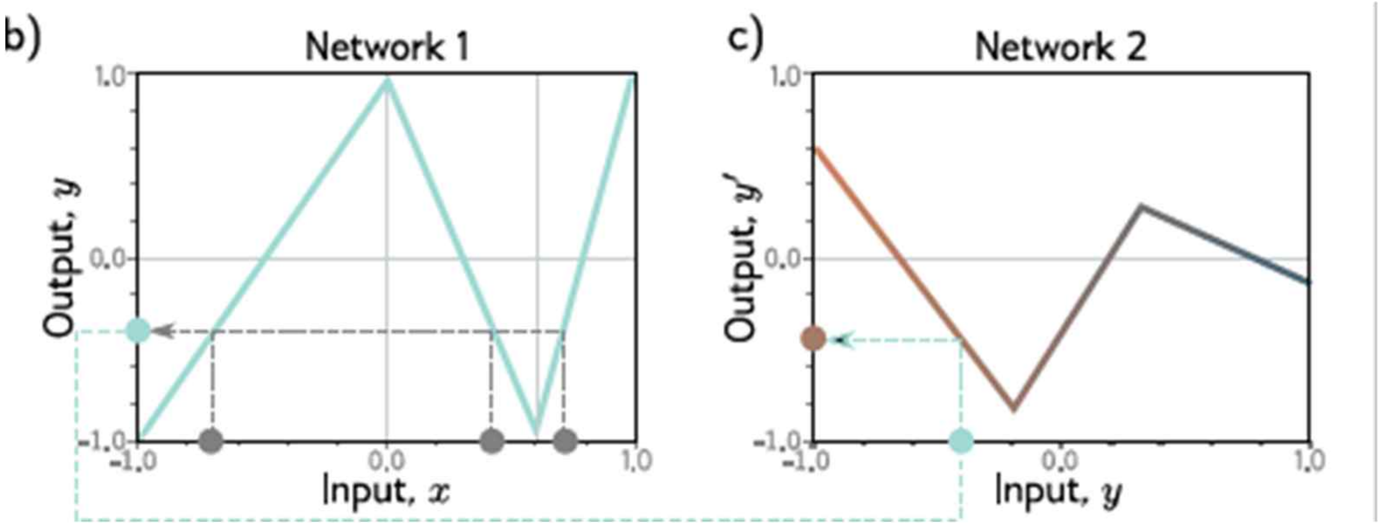
Second Network

$$h_1' = a[\theta_{10}' + \theta_{11}'y]$$
$$h_2' = a[\theta_{20}' + \theta_{21}'y]$$
$$h_3' = a[\theta_{30}' + \theta_{31}'y]$$
$$y' = \phi_0' + \phi_1' h_1' + \phi_2' h_2' + \phi_3' h_3'$$



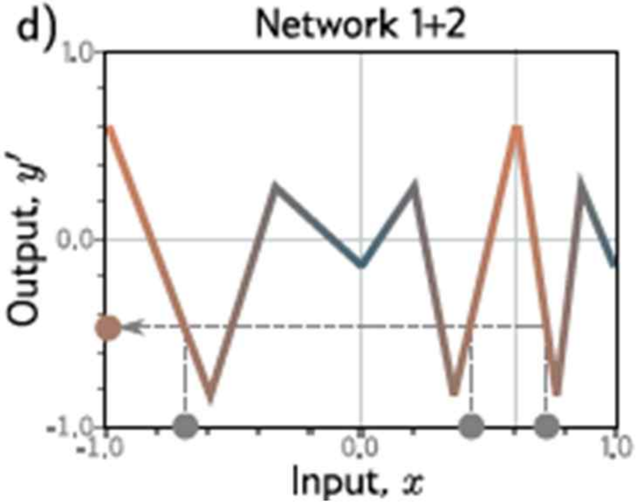
- Two shallow networks with three hidden units each
- First network -> input : x , output : y
- Second network -> input : y , output : y′

Composing neural networks



- First network maps inputs $x \in [-1, 1]$ to outputs $y \in [-1, 1]$
- Multiple inputs x can be mapped to the same output y
- Second network maps inputs $y \in [-1, 1]$ to outputs $y' \in [-1, 1]$
- Each network has three linear regions

Composing neural networks



- input, output graph of Network 1 + Network 2

- First + Second network -> input : x , output : y’
- Multiple inputs x can be mapped to the same output y’

From composing networks to deep networks

In The Second Network

$$\begin{aligned}
h'_1 &= a[\theta'_{10} + \theta'_{11}y] = a[\theta'_{10} + \theta'_{11}\phi_0 + \theta'_{11}\phi_1 h_1 + \theta'_{11}\phi_2 h_2 + \theta'_{11}\phi_3 h_3] \\
h'_2 &= a[\theta'_{20} + \theta'_{21}y] = a[\theta'_{20} + \theta'_{21}\phi_0 + \theta'_{21}\phi_1 h_1 + \theta'_{21}\phi_2 h_2 + \theta'_{21}\phi_3 h_3] \\
h'_3 &= a[\theta'_{30} + \theta'_{31}y] = a[\theta'_{30} + \theta'_{31}\phi_0 + \theta'_{31}\phi_1 h_1 + \theta'_{31}\phi_2 h_2 + \theta'_{31}\phi_3 h_3]
\end{aligned}$$

ψ : psi

$\psi_{10} = \theta'_{10} + \theta'_{11}\phi_0$, $\psi_{11} = \theta'_{11}\phi_1$

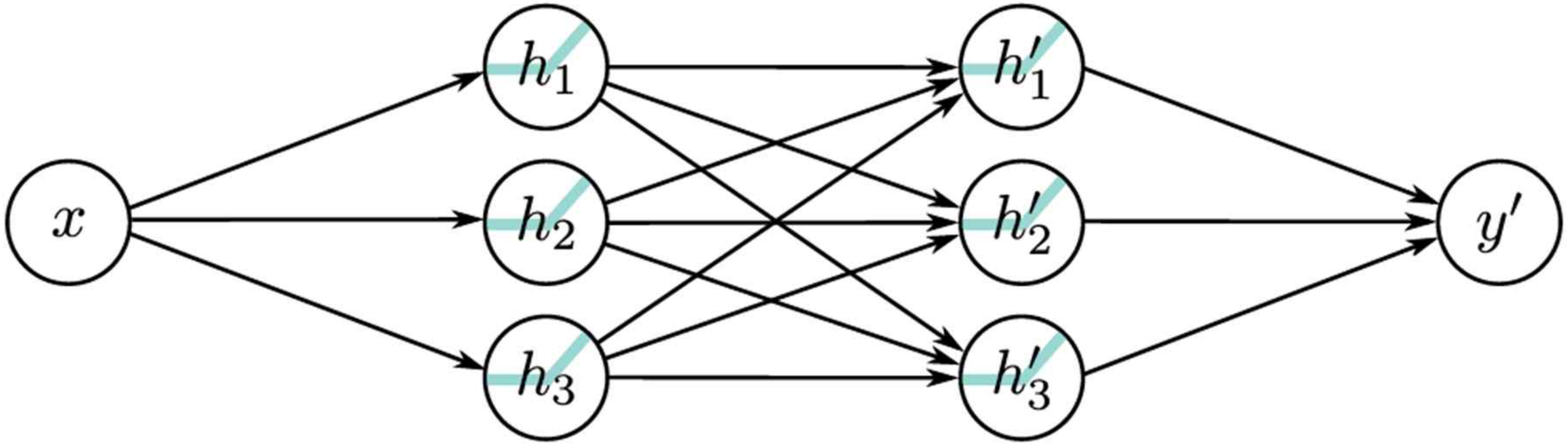
$\psi_{12} = \theta'_{11}\phi_2$, $\psi_{13} = \theta'_{11}\phi_3$ …

$$\Downarrow$$

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

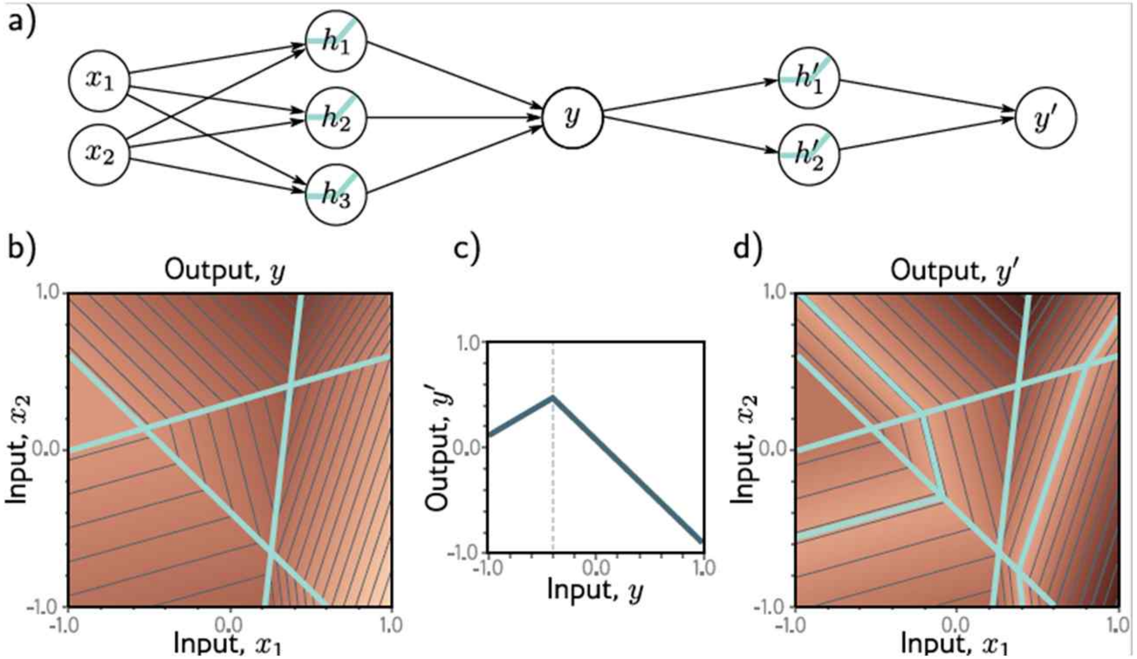
- In first equation, parameters are constrained to be products of elements from the vectors [θ'_{11}, θ'_{21}, θ'_{31}] and [ϕ_1, ϕ_2, ϕ_3]
- In second equation, ψ_{11} , ψ_{21}, … , ψ_{33} can take arbitrary value

From composing networks to deep networks



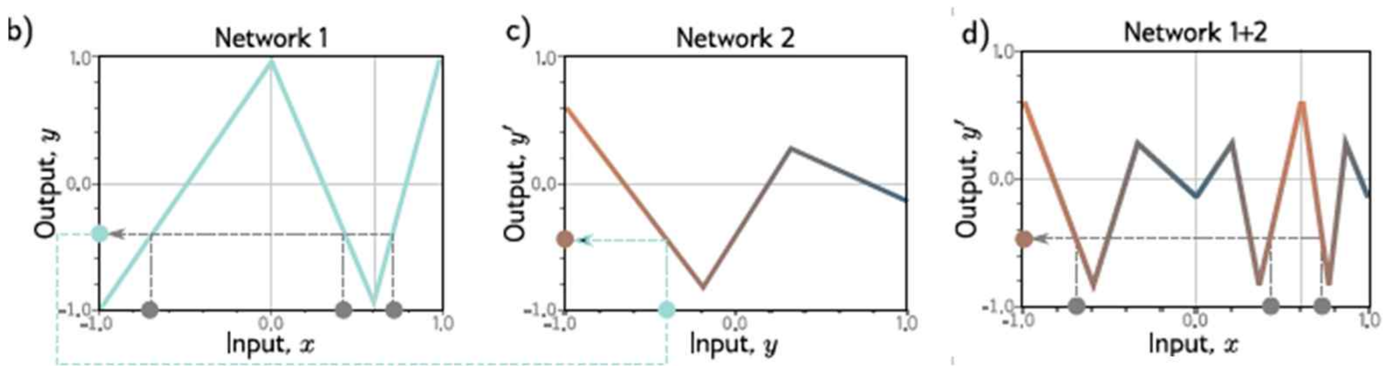
- Neural network with one input, one output, and two hidden layers
- Each layer contains three hidden units

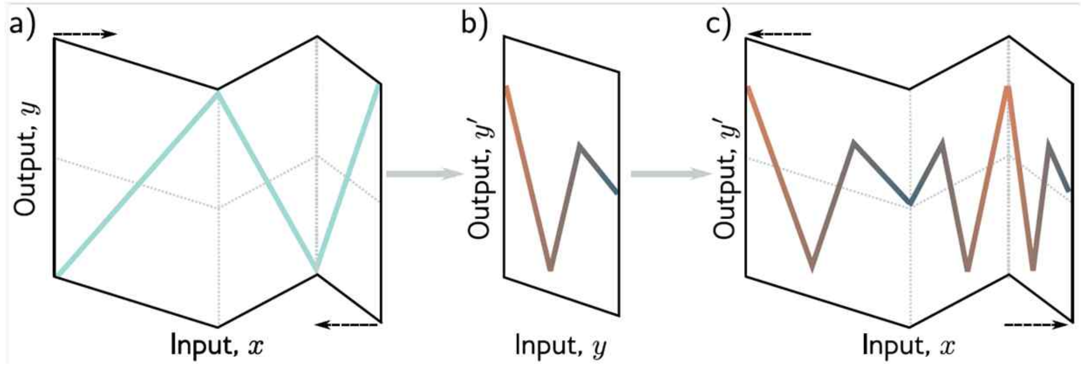
From composing networks to deep networks



- First network has two inputs, one output, three hidden units
- Second network has one input, one output, two hidden units
- b) generates 7 linear regions, c) generates 2 linear regions, d) generates 13 linear reigons

From composing networks to deep networks





Deep Neural networks

First layer

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

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

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

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

Second layer

$$h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

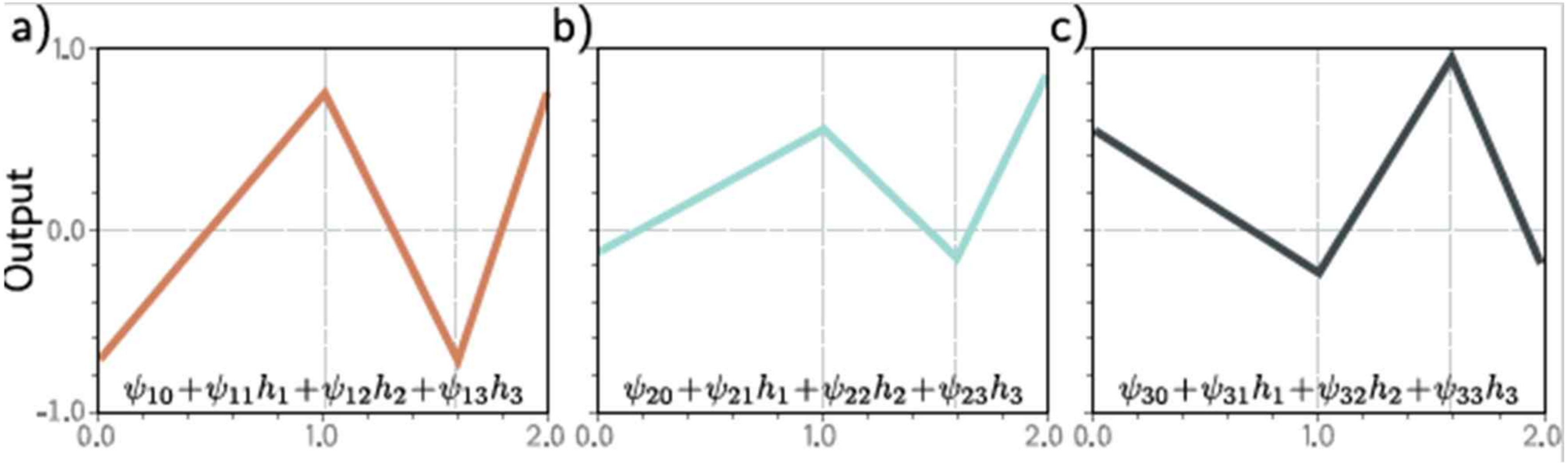
$$h'_2 = a[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = a[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

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

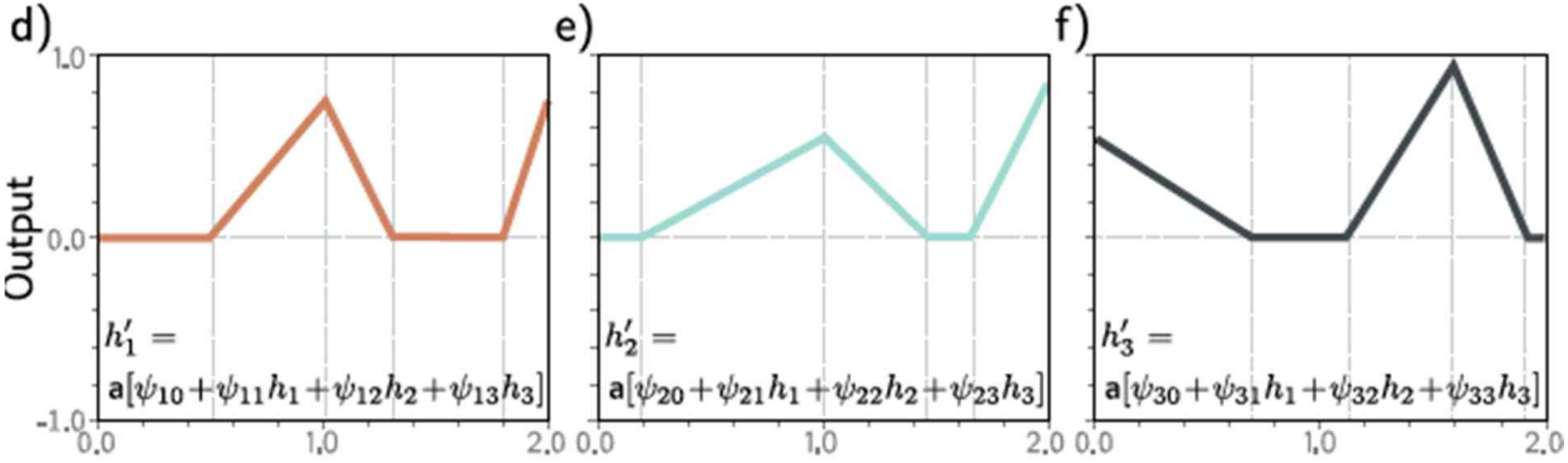
- General Deep Neural network with one input, one output, and two hidden layers
- Each layer contains three hidden units

Deep Neural networks



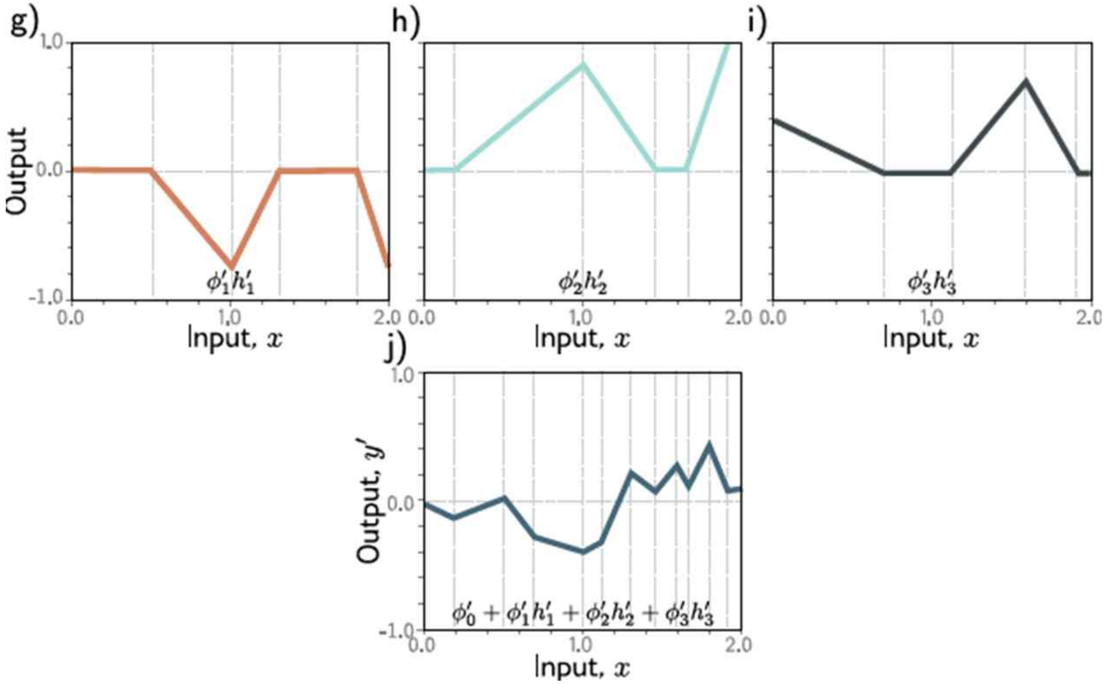
- Input to the second hidden layer
- joints between linear regions are at the same places

Deep Neural networks



- Each linear function is clipped to zero by the ReLu acvitivate function

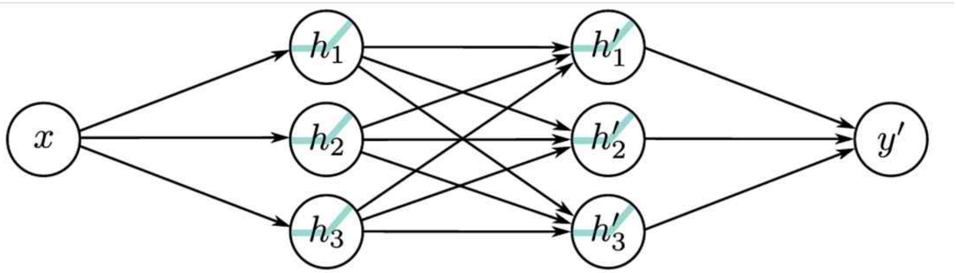
Deep Neural networks



- Clipped functions are then weighted with parameters $\phi1', \phi2', \phi3'$ respectively
- Finally, the clipped and weighted functions are summed and offset $\phi0'$ is added

Deep Neural networks

Hyperparameters

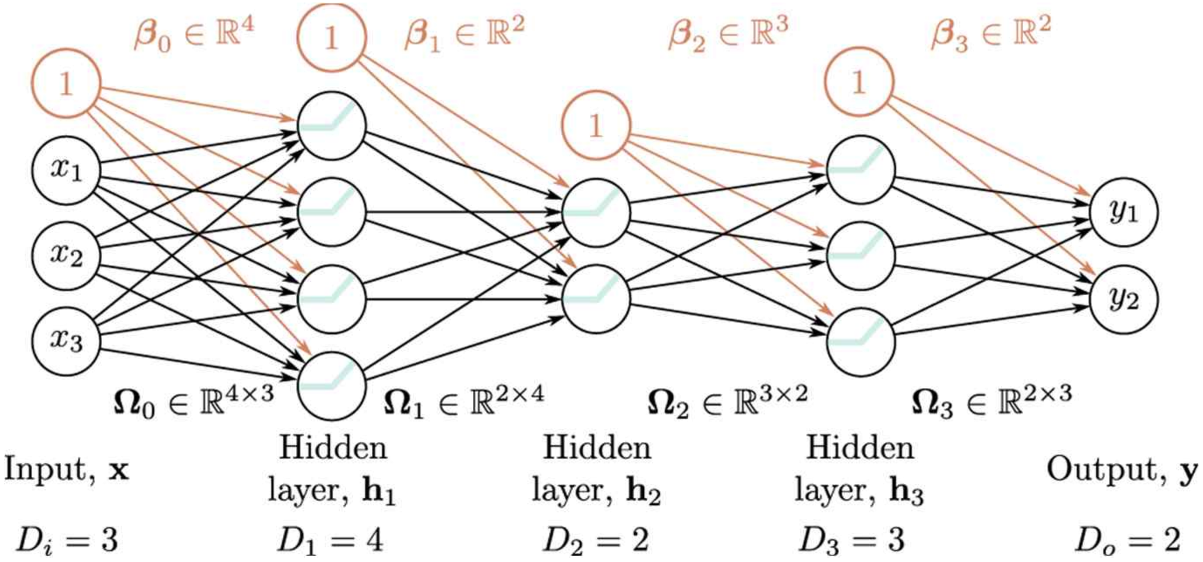


$K = 2$, $D_1 = 3$, $D_2 = 3$

- K : the number of layers (depth)
- $D_1, D_2, \ldots, D_K$: the number of hidden units at each layer (width)

Deep Neural networks

Hyperparameters



$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
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, $\mathbf{y}$
$D_i = 3$
$D_1 = 4$
$D_2 = 2$
$D_3 = 3$
$D_o = 2$

- Ω_k: weight matrix
- β_k: bias vector

Matrix notation

First layer

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

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

Second layer

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

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

⬇ ⬇

First layer

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = a\left(\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right)$$

$$output: y = \phi_0 + [\phi_1 \phi_2 \phi_3] \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

Second layer

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

$$output : y' = \phi_0' + [\phi_1' \phi_2' \phi_3'] \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix}$$

Matrix notation

First layer

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = a\left(\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right)$$

$$output: \; y = \phi_0 + [\phi_1 \phi_2 \phi_3] \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

Second layer

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

$$output \; : \; y' = \phi_0' + [\phi_1' \phi_2' \phi_3'] \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix}$$

⬇ ⬇

First layer

$$h = a[\theta_0 + \theta x]$$

$$output: \; y = \phi_0 + \phi h$$

Second layer

$$h' = a[\psi_0 + \Psi h]$$

$$output \; : y' = \phi_0' + \phi' h'$$

Matrix notation

General formulation

$$
\begin{aligned}
h_1 &= a[\beta_0 + \Omega_0 x] \\
h_2 &= a[\beta_1 + \Omega_1 h_1] \\
h_3 &= a[\beta_2 + \Omega_2 h_2] \\
&\cdots \\
h_\mathrm{k} &= a[\beta_{\mathrm{k}-1} + \Omega_{\mathrm{k}-1} h_{\mathrm{k}-1}] \\
\\
y &= \beta_\mathrm{k} + \Omega_\mathrm{k} h_\mathrm{k}
\end{aligned}
$$

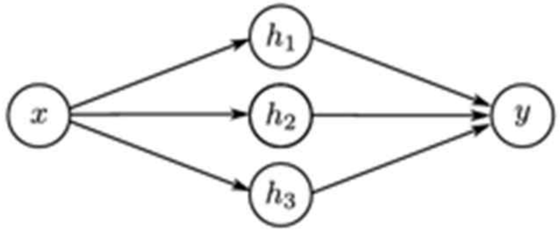
$$y = \beta_\mathrm{k} + \Omega_\mathrm{k}\, a[\beta_{\mathrm{k}-1} + \Omega_{\mathrm{k}-1}\, a[\ldots \beta_2 + \Omega_2\, a[\beta_1 + \Omega_1\, a[\beta_0 + \Omega_0\, x]] \ldots]]$$

β_k : bias vector
Ω_k : weight matrix

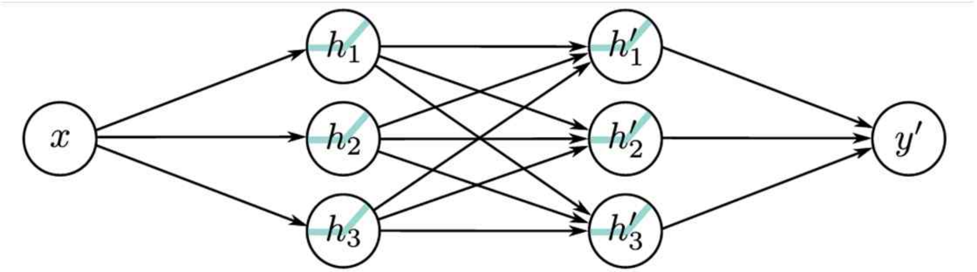
- Deep Neural Network as a single function

Shallow vs. deep neural networks

Ability to approximate different functions



VS

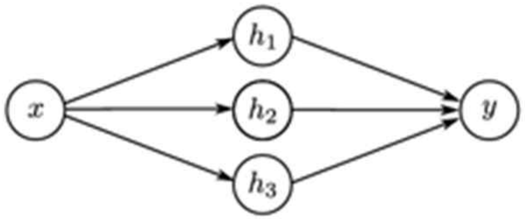


- Deep network with N hidden layers could represent the composition of N shallow networks
- If the second of these(right image) networks computes the identity function,
  this deep network replicates a single shallow network

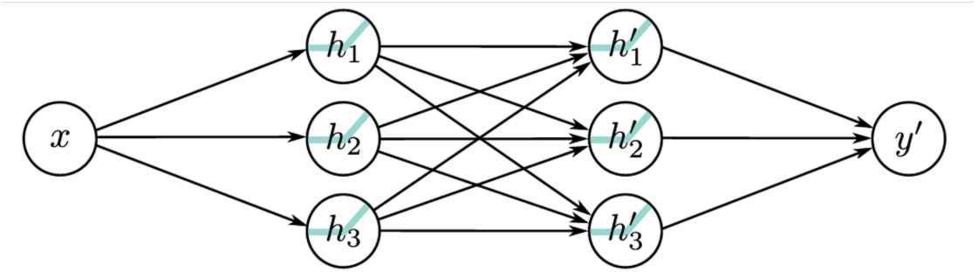
∴ Deep neural network can approximate same function as shallow neural networks do

Shallow vs. deep neural networks

Number of linear regions per parameter



VS

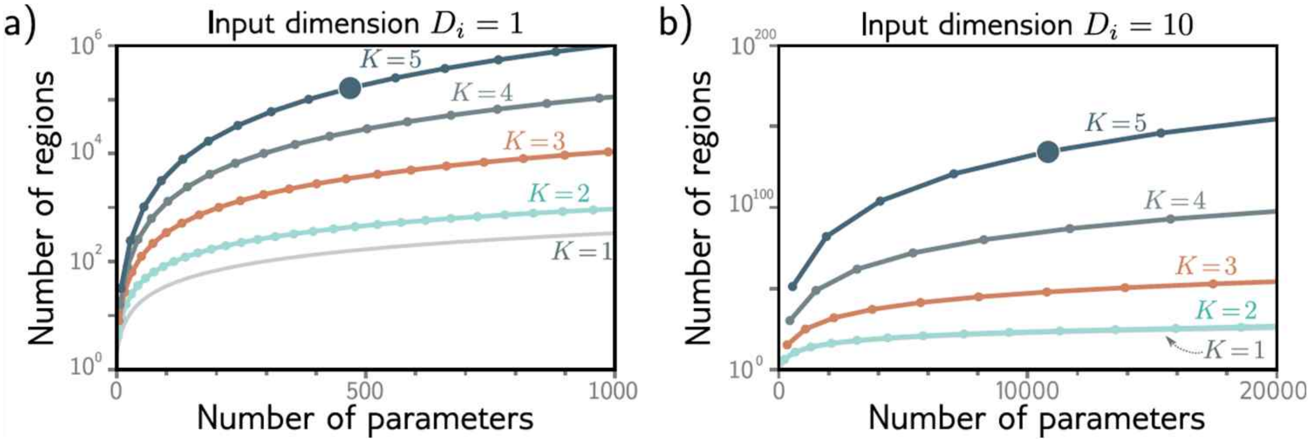


- One input , one output
- D hidden units can create up to D +1 linear regions

- One input , one output
- K layers of D hidden units can create up to $(D+1)^K$ linear regions

Shallow vs. deep neural networks

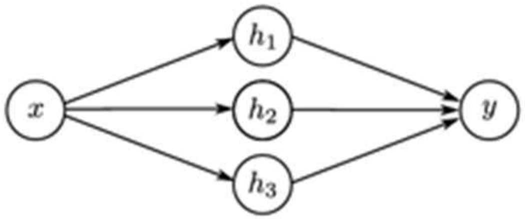
Number of linear regions per parameter



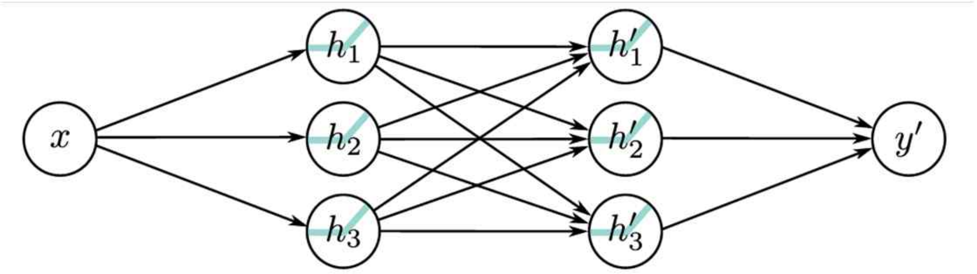
- K : number of layers, x : Number of parameters , y : Number of regions
- Each subsequent point represents 1 hidden unit in a) graph, and 10 hidden units in b) graph
- In graph a), if K = 5 , Number of hidden units per layer = 10 has about 500 parameters, and can create more than 10^5 linear regions
- In graph b), if K = 5 , Number of hidden units per layer = 50 has about10000 parameters, and can create more than 10^{100} linear regions

Shallow vs. deep neural networks

Large, structured Input , Training and generalization



VS



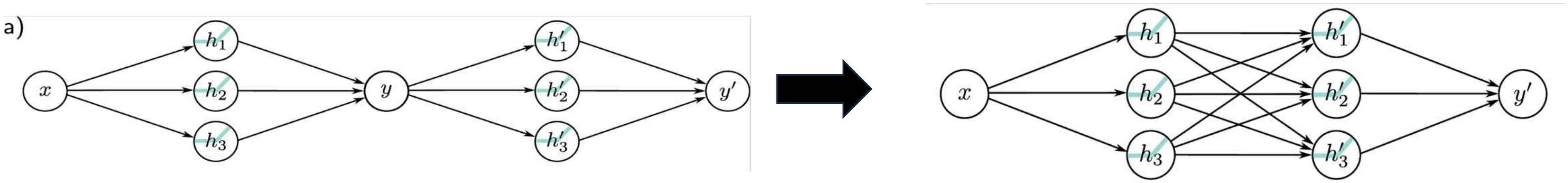
- Local-to-global processing is difficult to specify without using multiple layers (chapter 10)
- Deep network is usually easier to train moderately than to train shallow network
- More hidden layers -> More difficult to train

∴ Appropriate training techniques and hyperparameter tuning are necessary

2. Summary

Summary

A deep neural network can be constructed by combining shallow neural networks



Hyperparameter
- K : the number of layers (depth)
- $D_1, D_2, \ldots, D_K$: the number of hidden units at each layer (width)

Shallow vs. deep neural networks
- Both networks can approximately any function given enough capacity
- Deep networks produce more linear regions per parameter
- Some functions can be approximated much more efficiently by deep networks

3. Code

Code

1. Supervised learning
https://colab.research.google.com/drive/1T-8dlHoQJvDffHgiyjX1lKS62jIk19Dg#scrollTo=sfB2oX2RNvuF

2. Composing networks
https://colab.research.google.com/drive/1t7Ha60oqkDcyY0WsHBKAvEQwnpvPsXTC

3. Clipping functions
https://colab.research.google.com/drive/1Pjx2wJJ6W-L2ATySxR9J4h51w409eyjr

4. Deep neural networks
https://colab.research.google.com/drive/1XwE1mHHTSX-sovIvCMoTV1j_2QFR5A3B