Introduction to Theory of Deep Learning

2024/03/05 MLSS

Masaaki Imaizumi
(The University of Tokyo/RIKEN AIP)

Today's Outline

Brief Introduction to deep learning

Approximation theory of deep learning (hands-on)

Complexity theory of deep learning

1st slot

2nd slot

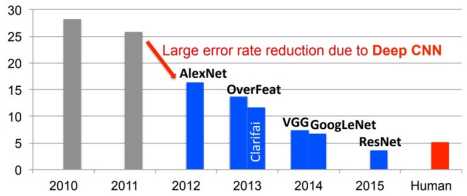
Introduction to Deep Learning

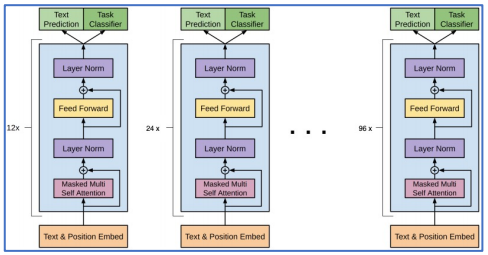
Emergence of Deep Learning (DL)

fundamental → breakthough → Transformer → ChatGPT

~2000 | 2012 | 2018 | 2022

issues





Accurate image analysis

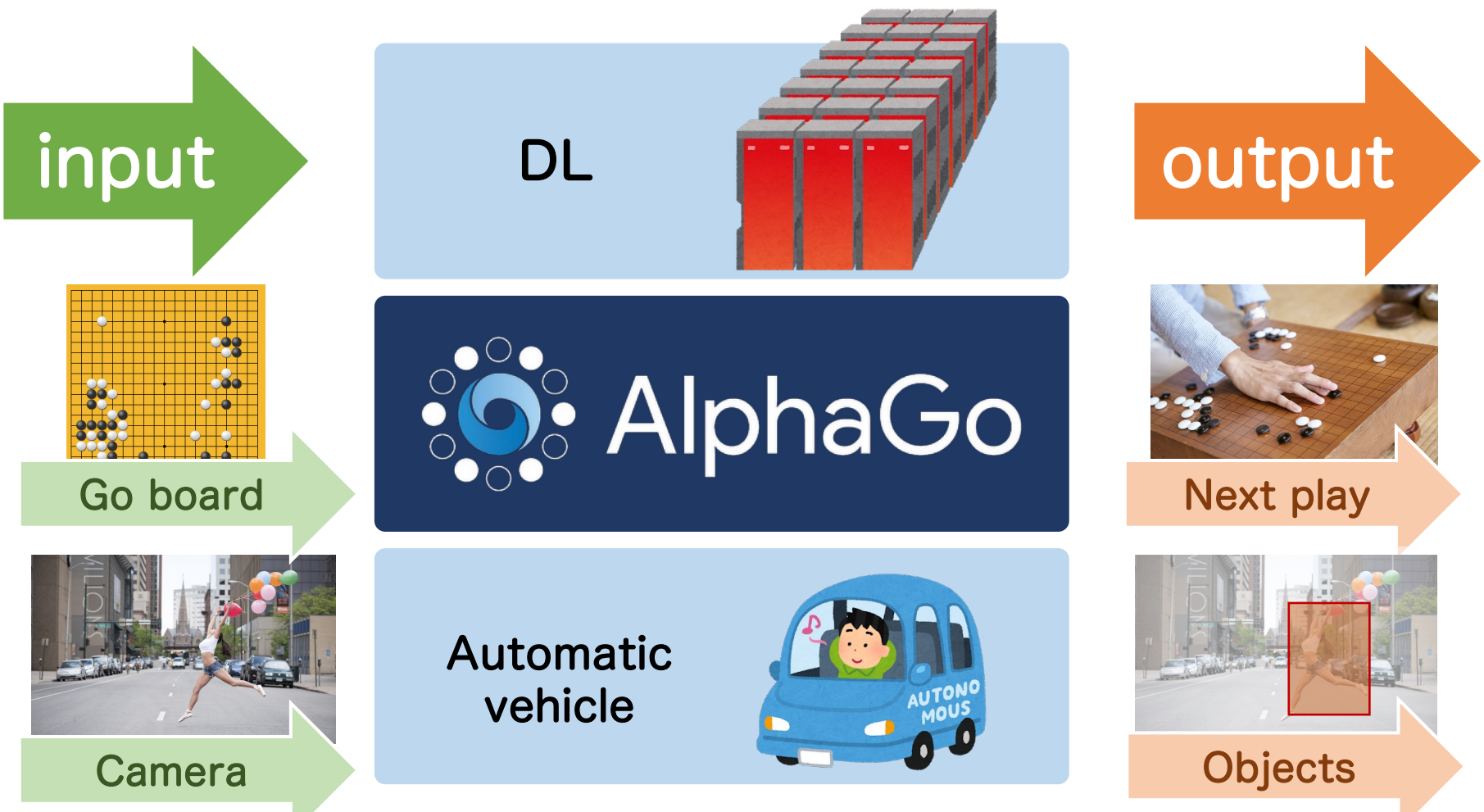
Advanced architecture

General-purpose AI model

Deep Learning (DL): accurate data-driven techology

DL makes functions

- System that produces outputs for inputs



Inside DL

Deep Neural Network (DNN)

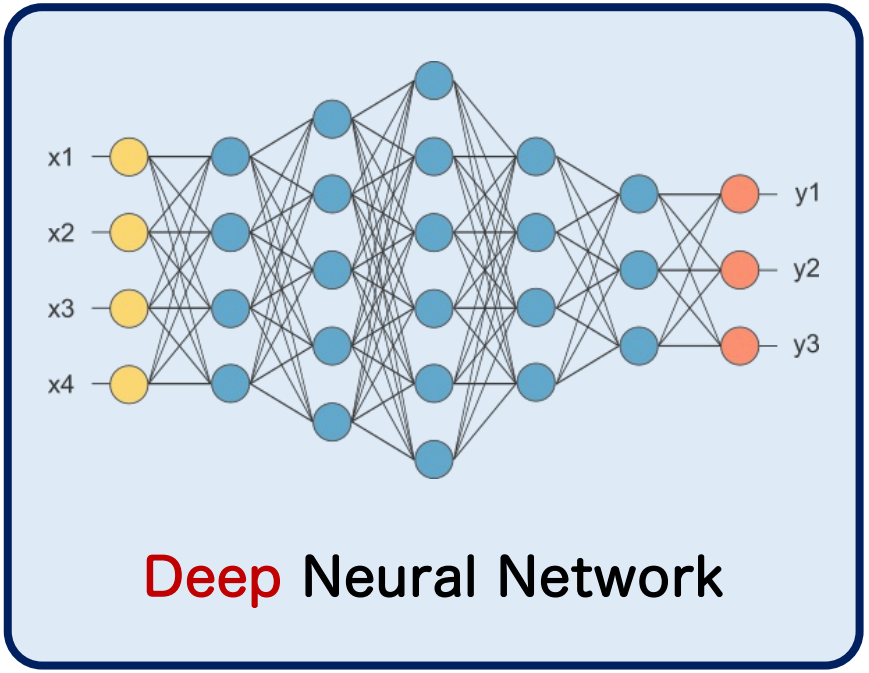
- Model of functions transforms an input vector

Input (ex. image)

convert

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

vector x



Output (ex. info.)

This is Brown Cat

convert

$$y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

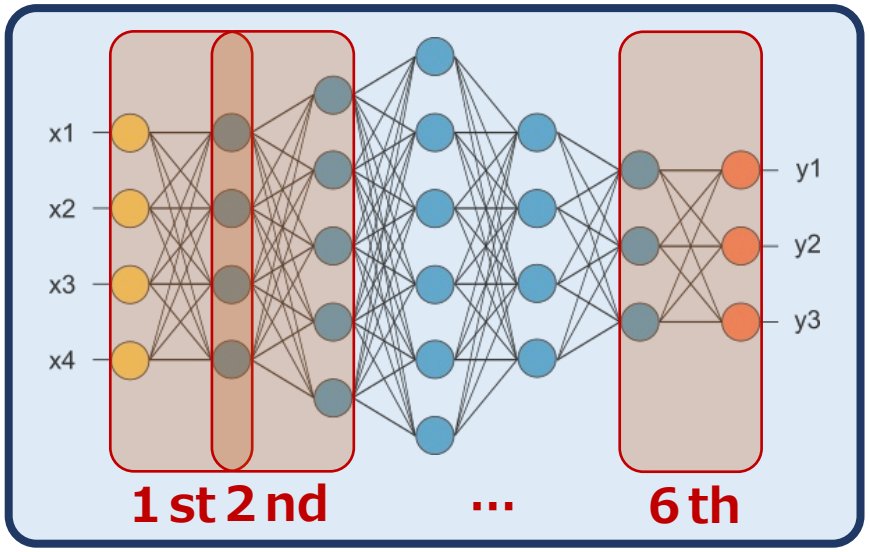
Vector y

Deep neural network (DNN)

Transform vectors for # of layers

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

Vector x



$$y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

Vector y

Transform vectors

Layer 1 $z_1 = \sigma(A_1 x + b_1)$

Layer 2 $z_2 = \sigma(A_2 z_1 + b_2)$

⋮ ⋮

Layer 6 $y = A_6 z_5 + b_6$

A: parameter matrix
b: parameter vector
σ: activation function

Model for Deep Learning

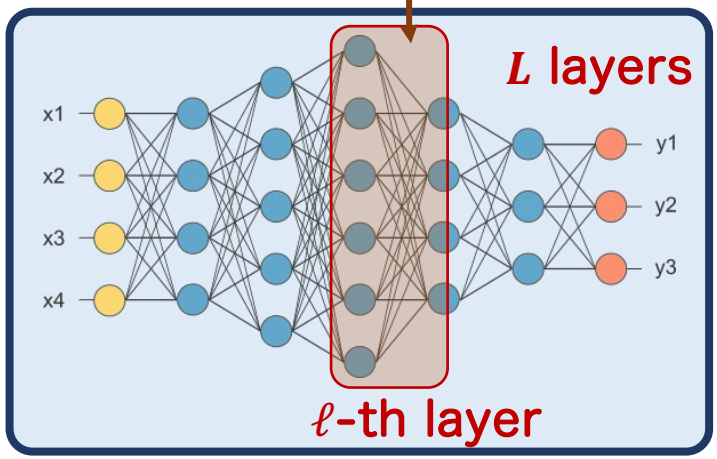
Deep Neural Network (DNN)

- Model with multiple layers for functions

At ℓ-th layer

$$f_\ell(x) := \sigma(A_\ell x + b_\ell)$$

input x



output y

$A_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}, b_\ell \in \mathbb{R}^{d_\ell}$: parameter

$\theta = \{(A_\ell, b_\ell)_{\ell=1}^{L}\}$: all parameters

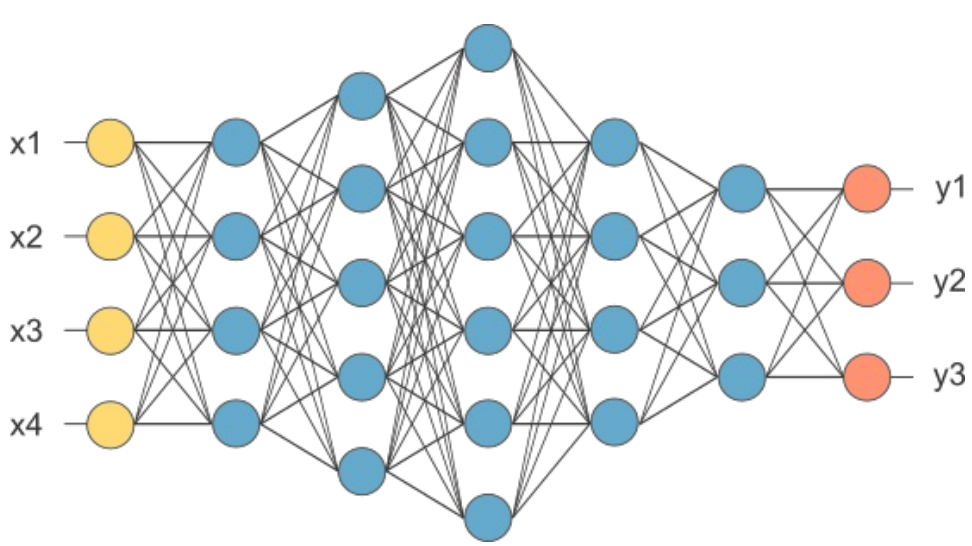
$\sigma: \mathbb{R}^d \to \mathbb{R}^d$: activation

Function by DNN

$$f_\theta = f_L \circ f_{L-1} \circ \cdots \circ f_1(x)$$

Depth and width

- Quantities characterizing the size of DNN



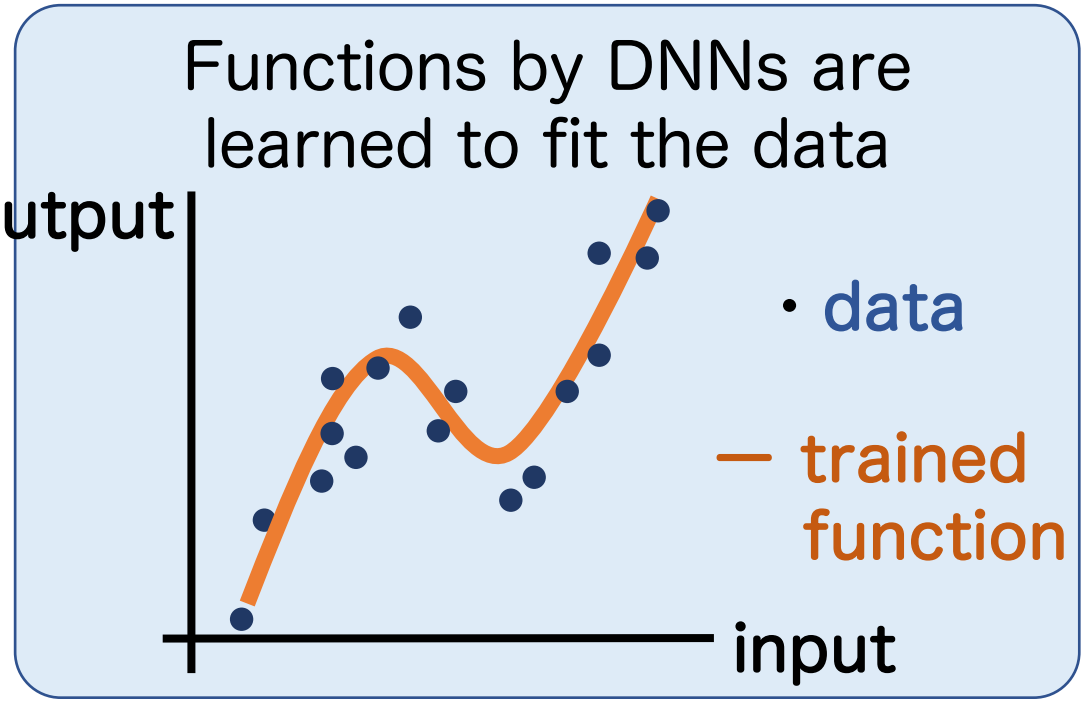
Width
(Maximum # of nodes per layer)
$\max_{\ell} d_{\ell}$

Depth L (# of layrers)

Params are learned from data

Params θ: Necessary for DNNs to work

- Learning to reproduce the structure of data



Ex. Quadratic risk

θ : parameter

(y_i, x_i) : data

$$\min_{\theta} \sum_i \underbrace{\left(y_i - f_\theta(x_i)\right)^2}_{\text{loss}}$$

= Misalignment of DNN functions and data

Parameter Training

- Minimize empirical risk,
  then study generalization error

Empirical Risk

$$R_n(\theta) = n^{-1}\Sigma_{i=1}^{n}\ell\left(y_i, f_\theta(x_i)\right)$$

Training data (size n)

$$\{(x_i, y_i)\}_{i=1}^{n}$$

ℓ: loss

Update θ to minimize $R_n(\theta)$,
then obtain $\hat{\theta}$.

Generalization Error

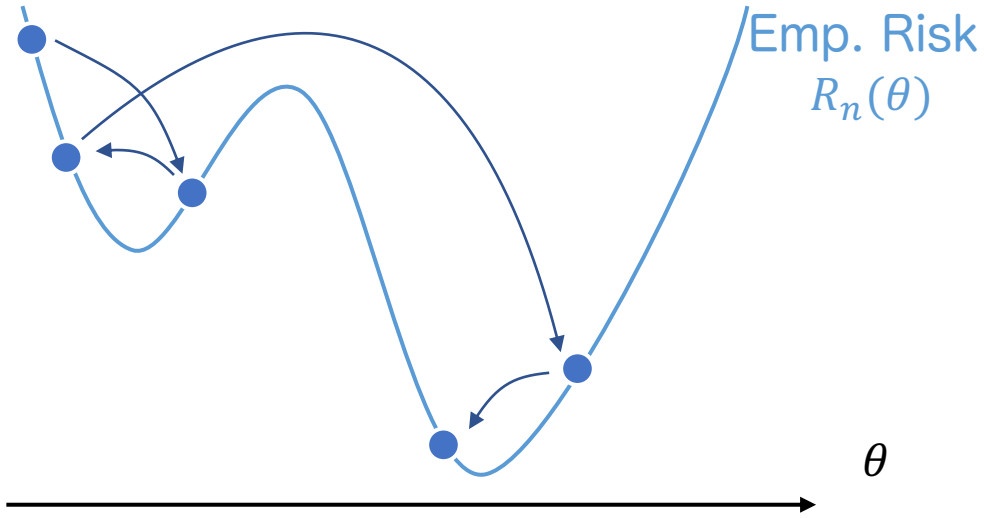
$$R(\theta) = E[\ell\left(y, f_\theta(x)\right)]$$

Expected risk
($\approx$ test error)

Algorithm for Training

Stochastic Gradient Descent (SGD)

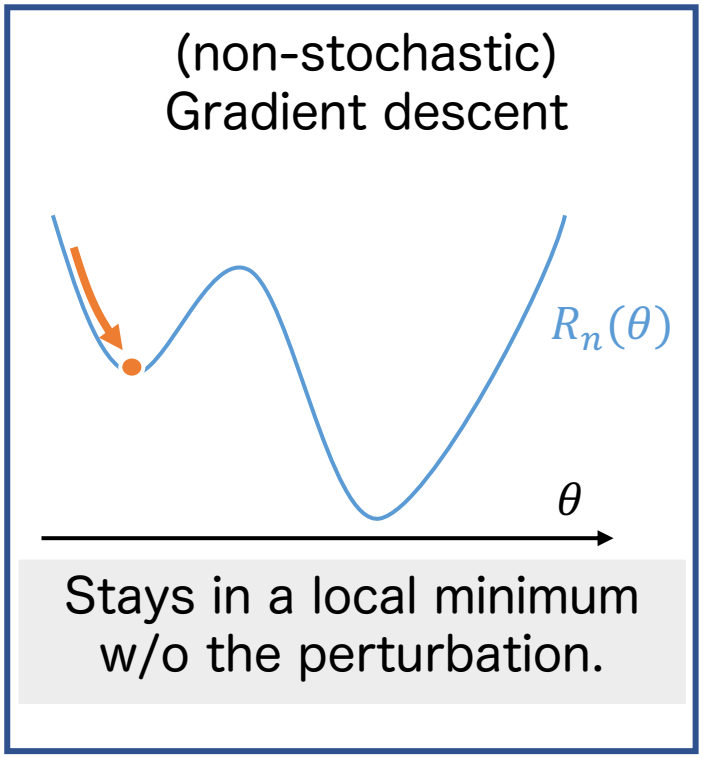
- Gradient descent + stochastic perturbation



Iteration for update ($\eta_t > 0$: learning rate)

$\hat{R}(\theta)$: mini-batch risk ($R_n(\theta)$+perturb.)

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \hat{R}(\theta)$$



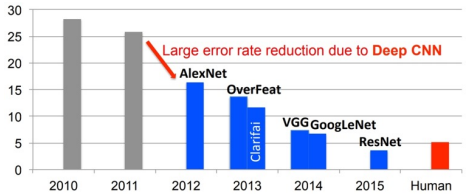
Introduction to Deep Learning Theory

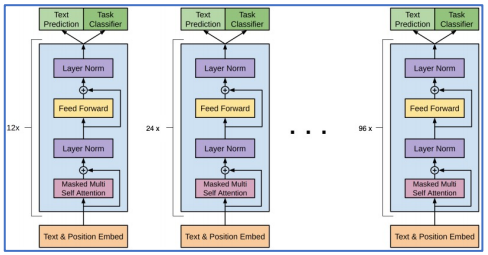
Deep Learning (DL)

fundamental → breakthough → Transformer → ChatGPT

~2000 | 2012 | 2018 | 2022

issues





Accurate image analysis

Advanced architecture

General-purpose AI model

Why DL is high-performance?

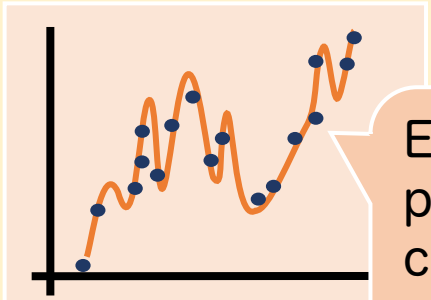
⇒ Need Theoretical Understanding

For effective usage...

Puzzle by DL

Contradiction b/w DL and (classical) theory

Classical Theory

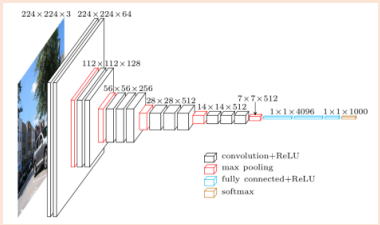


Excess parameters cause overfitting.

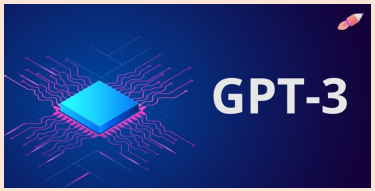
$$\text{Error} \propto \sqrt{\frac{\#\text{ of parameters}}{\#\text{ of data}}}$$

GAP

Success of DL



VGG19 Net
100 million~ parameters



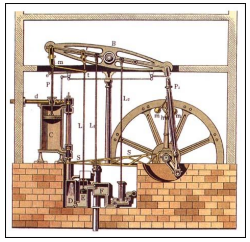
GPT-3
100 billion~ parameters

More parameter, higher accuracy

→ **Rethinking Theory**: Develop theory for DL

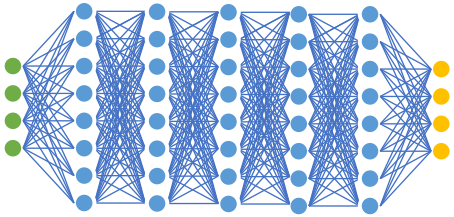
From Invention to Theory

- In history...



Steam Engine (1769) ➡ Thermodynamics

Airplane (1903) ➡ Aerodynamics



Deep Learning (2012) ➡ ?

Q : Possible to develop theory for DL?

Decomposition of Generalization Error

$\hat{\theta}$: trained parameter

Error Decomposition

Decompose gen. error into three parts

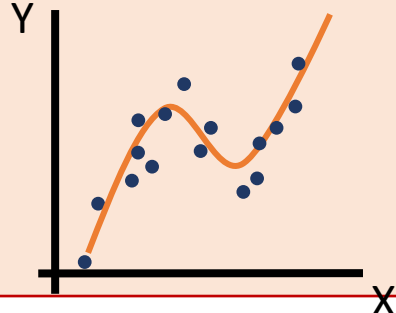
- DL contains several aspects...

$$R(\hat{\theta}) = \inf_{\theta} R_n(\theta) + R(\hat{\theta}) - R_n(\hat{\theta}) + R_n(\hat{\theta}) - \inf_{\theta} R_n(\theta)$$

Gen. Error | Approximation Error | Complexity Error | Optimization Error

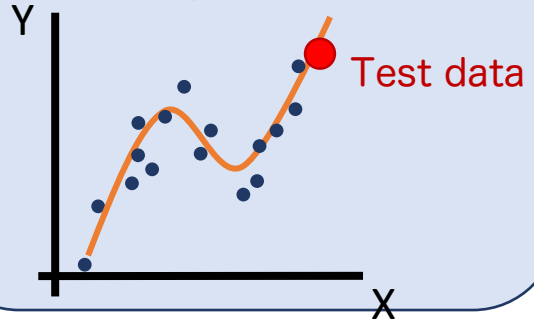
Approx. Error

How DNNs fit to data structure?



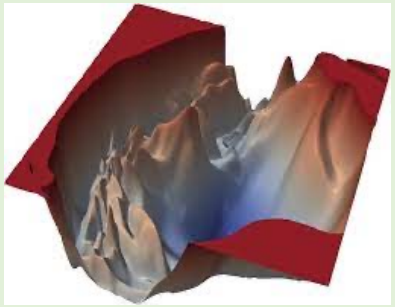
Complexity Error

How DNNs predict unknown data?



Optimization Error

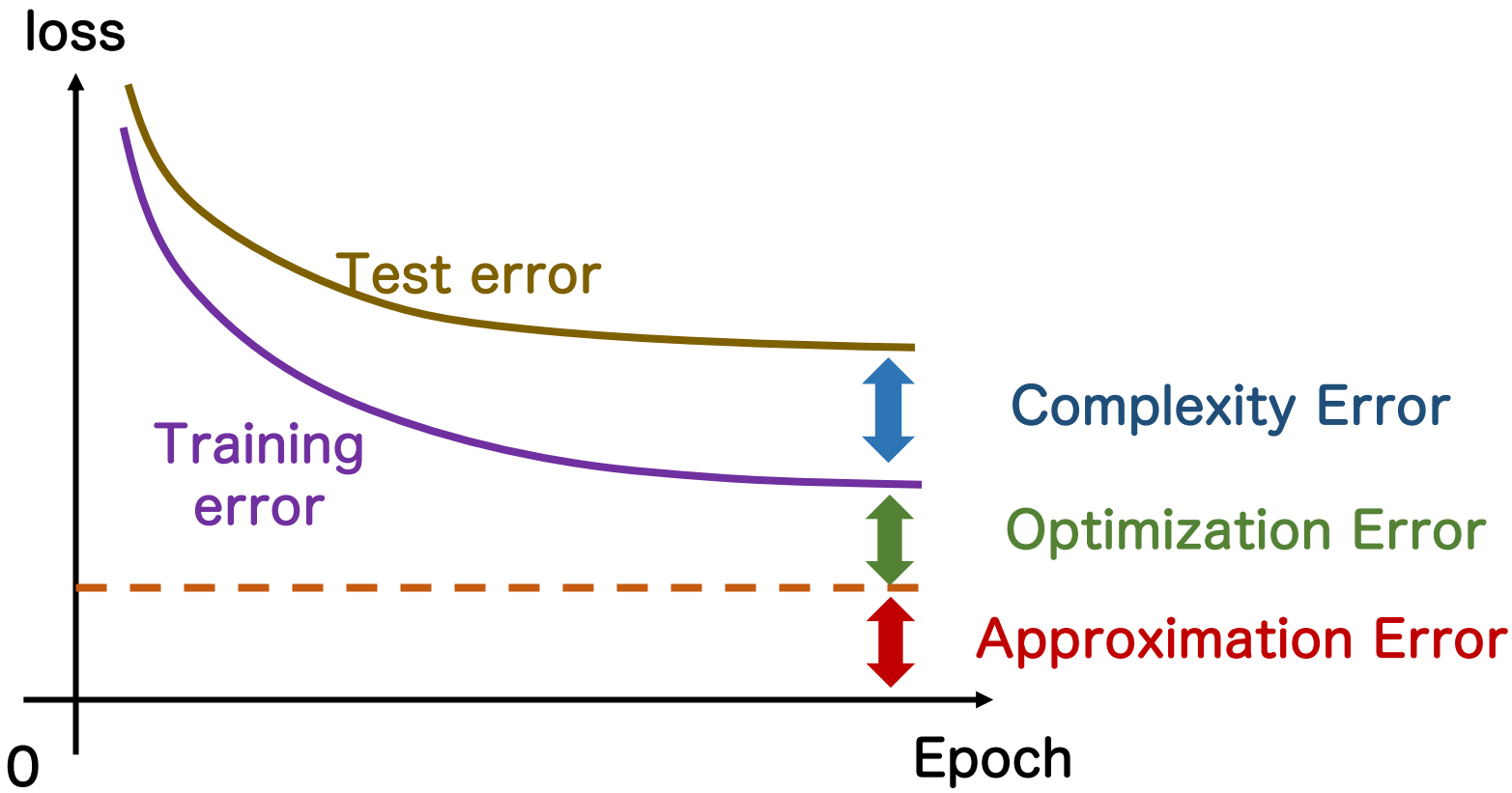
How algorithm works?



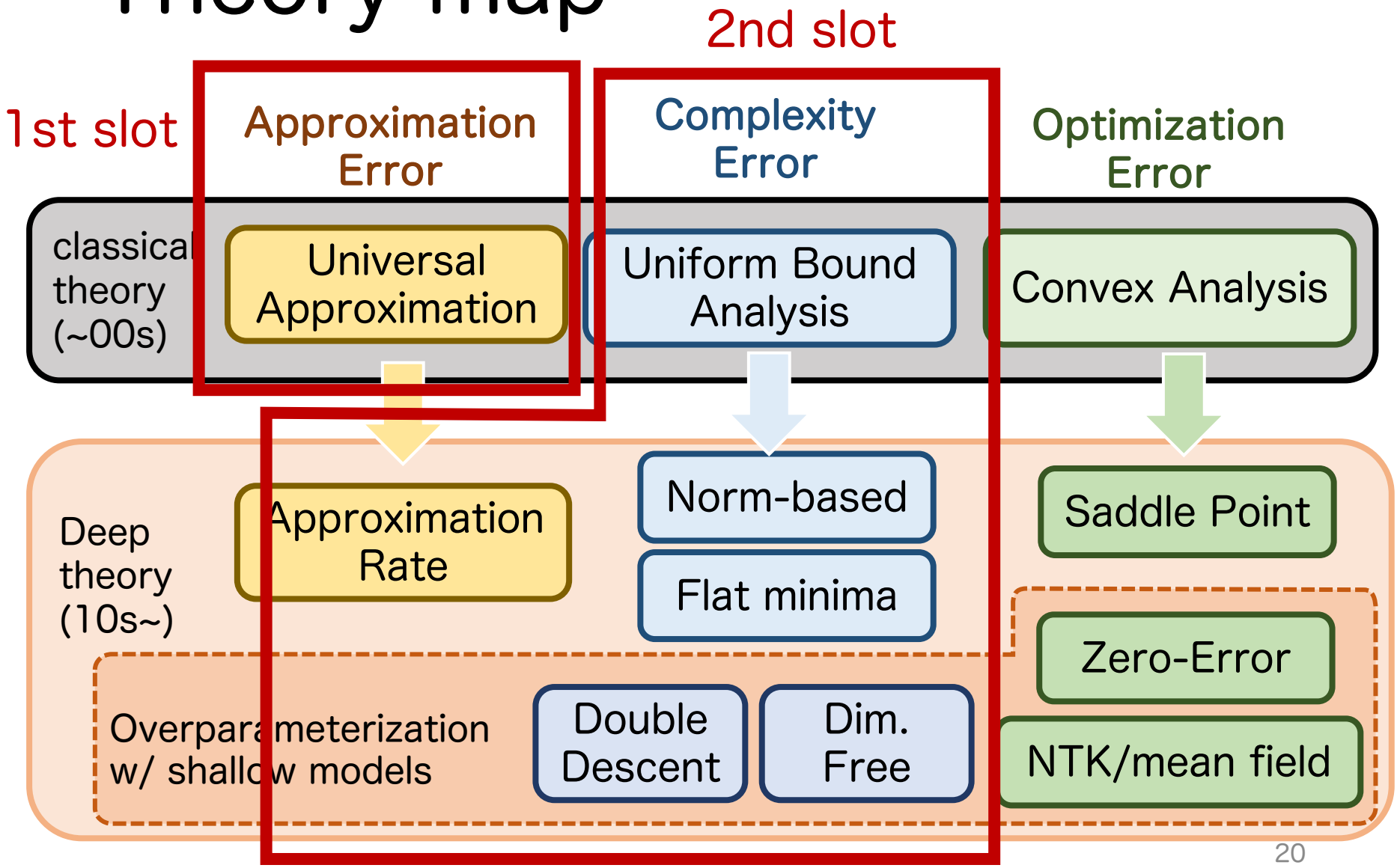
Loss surface

Error Decomposition

- Decomposition and Learning Curve

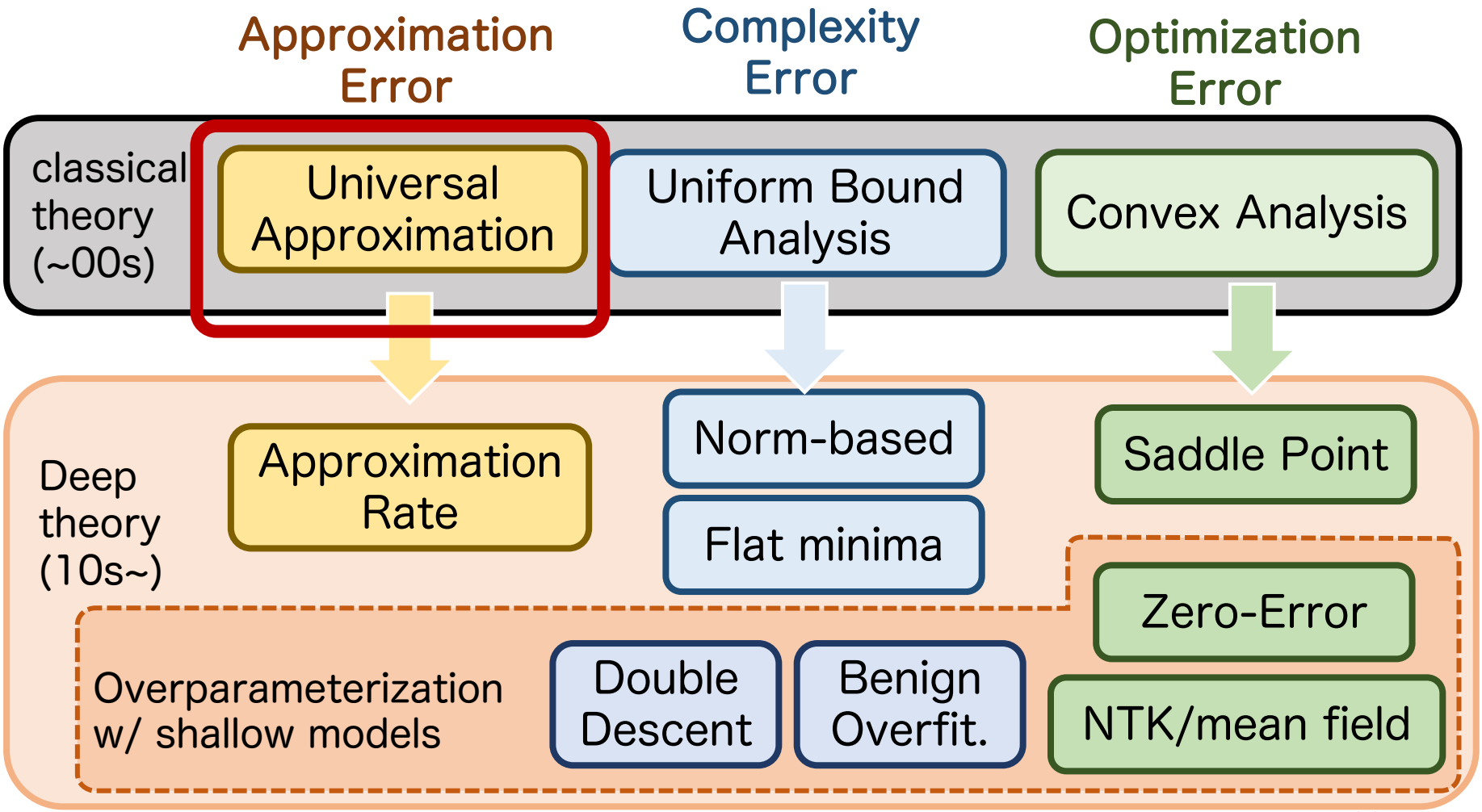


Theory map


2nd slot
1st slot
Approximation Error
Complexity Error
Optimization Error
classical theory (~00s)
Universal Approximation
Uniform Bound Analysis
Convex Analysis
Deep theory (10s~)
Approximation Rate
Norm-based
Flat minima
Saddle Point
Zero-Error
Overparameterization w/ shallow models
Double Descent
Dim. Free
NTK/mean field

Approximation Error

Theory map


Approximation Error
Complexity Error
Optimization Error
classical theory (~00s)
Universal Approximation
Uniform Bound Analysis
Convex Analysis
Deep theory (10s~)
Approximation Rate
Norm-based
Flat minima
Saddle Point
Zero-Error
Overparameterization w/ shallow models
Double Descent
Benign Overfit.
NTK/mean field

Approximation Error

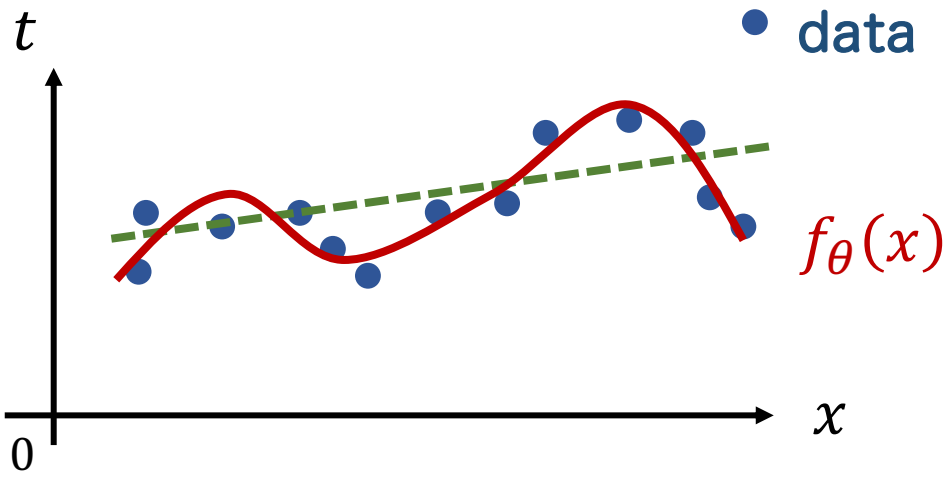
- How DNNs f_θ express data structure?

Regression case (ℓ is quadratic)

f_θ : DNN
f^*: true function

If data are generated from $Y = f^*(X) + \varepsilon$,

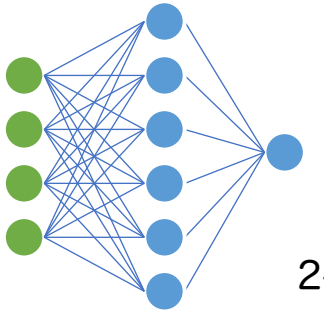
$$\inf_\theta R_n(\theta) \leq \inf_\theta \|f^* - f_\theta\|_\infty^2 + \text{Noise Terms}$$



fits the data.

does not fit the data.

Universal Approximation



2-layer NN

Claim:

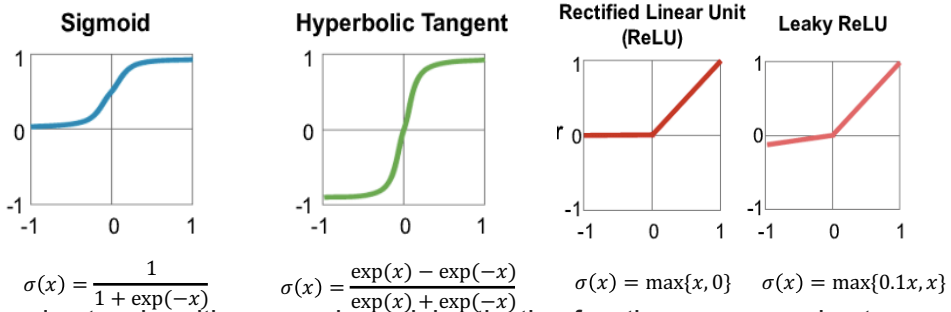
- Neural net (NN) approximates any continuous function.

Universal Approximation Theorem (UAT) (e.g. Leshno (1993))

For any continuous $f: [0,1]^d \to \mathbb{R}$ and level $\varepsilon > 0$,
there is a 2-layer NN f_θ with a non-polynomial activation σ,
such that

$$\|f - f_\theta\| \leq \varepsilon.$$

- Non-polynomial:
  $\sigma(x)$ is not a polynomial
  (like $\sum_a x^a$)

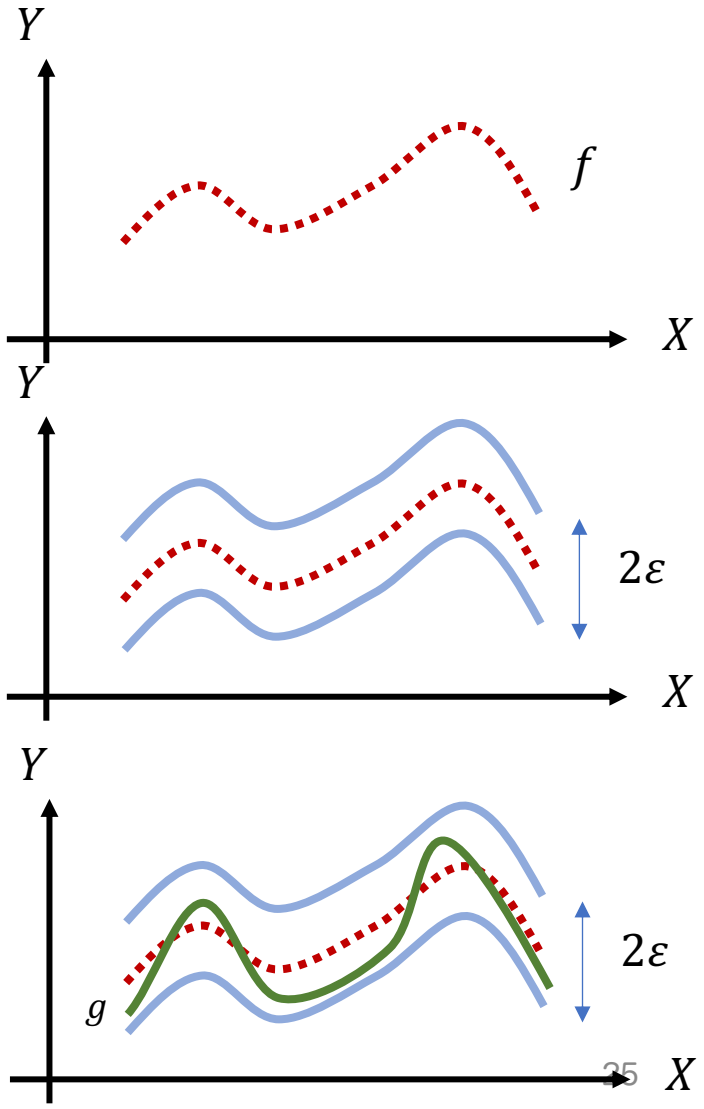


Leshno, M., Lin, V. Y., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*

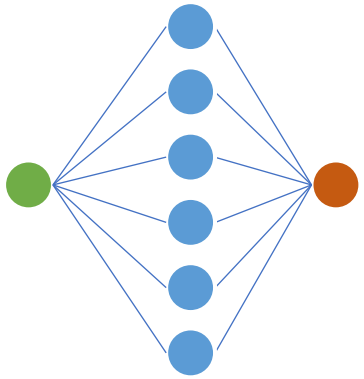
Sup-norm case:
$\|f\| = \sup_x |f(x)|$

What the theorem says?

- Need this function to fit data.
- Set an error level $\varepsilon > 0$.
  - Any positive value is OK.
- NN can make a function within this band.
  - W/ sufficiently many width.

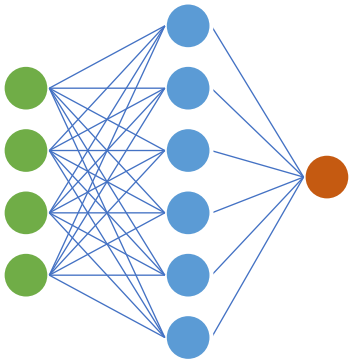


How to show it?



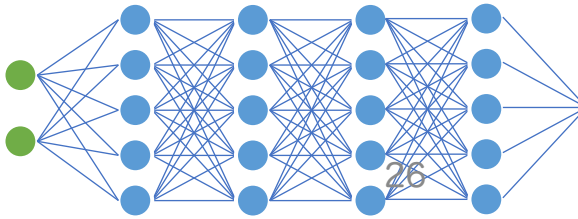
3 Steps:

1. Approximation power of DNNs with fewer layers (1-dim. input)



2. Approximation power of DNNs with fewer layers (d-dim. input)

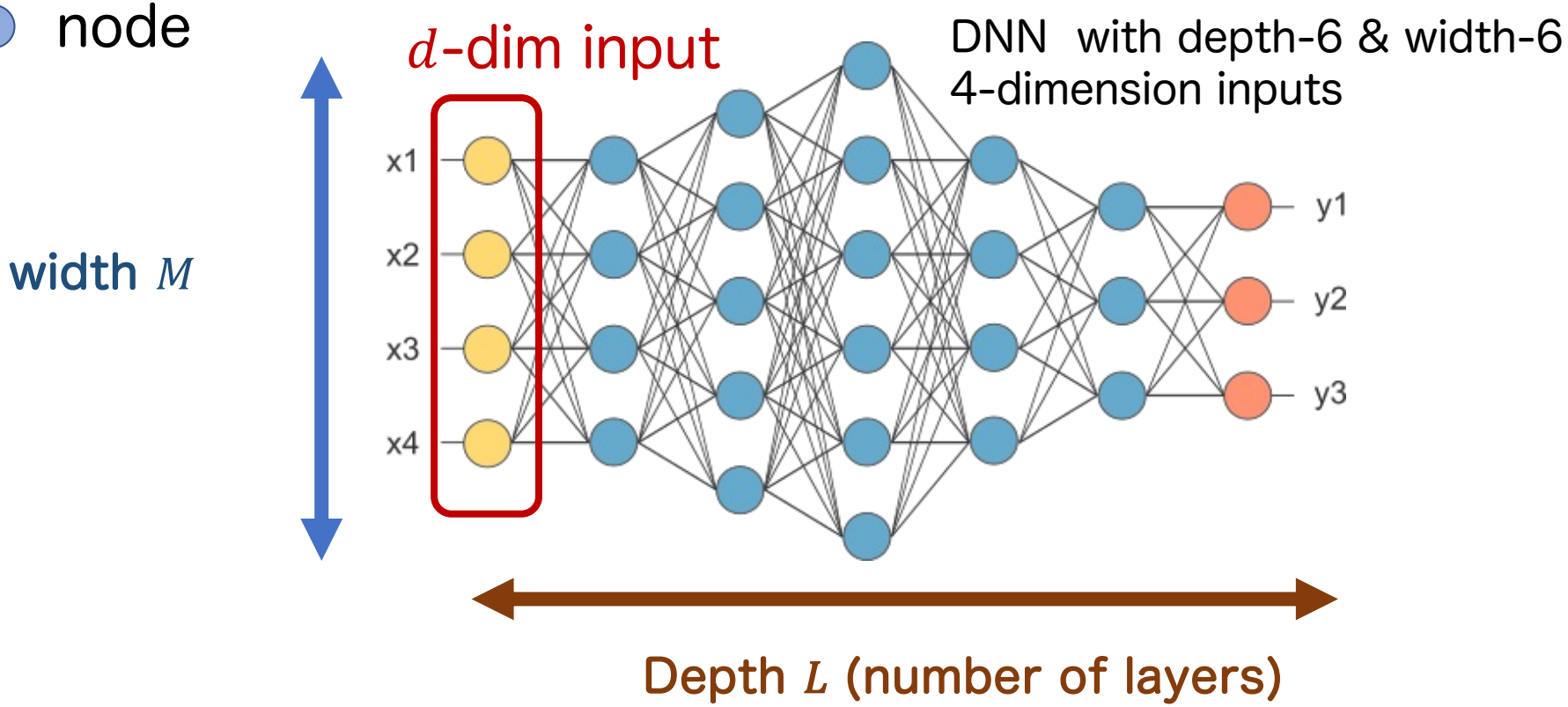
3. Approximation power of DNNs with many layers (less width)



Setup of DNN

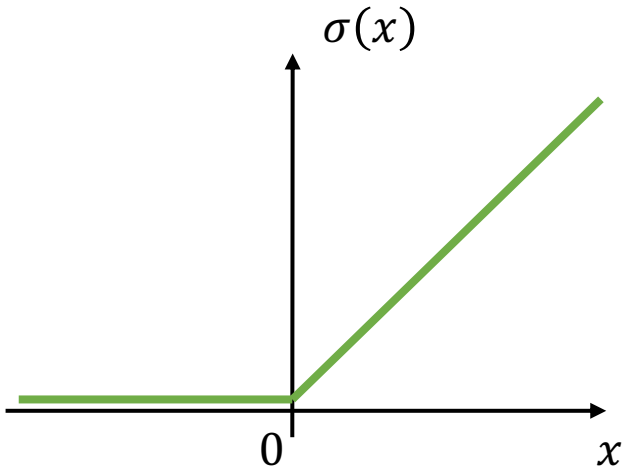
- Properties are measured by depth and width.

● node



In this talk, output is limited to 1-dim. for simplicity

Basic notation



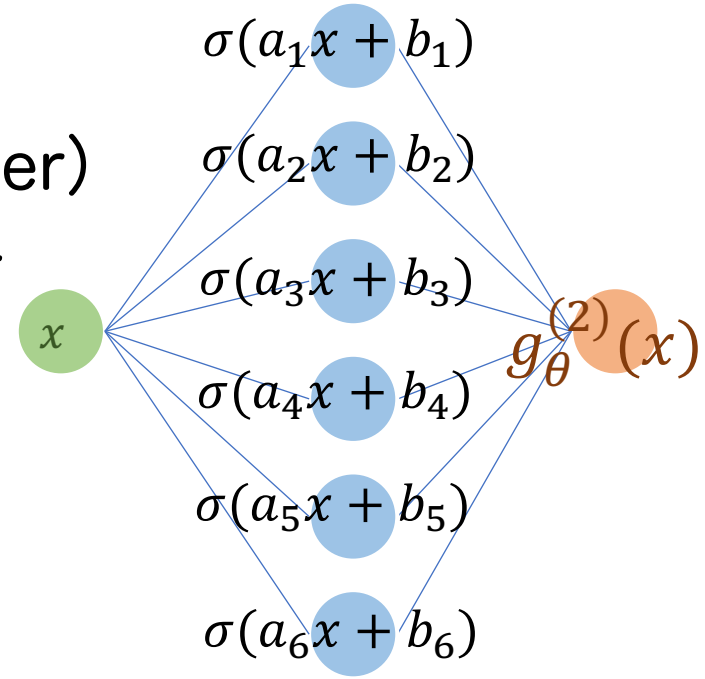
- $\sigma(x) = \max\{x, 0\}$: ReLU activation
- $1\{A\}$: indicator function
  - If A is True, $1\{A\} = 1$. If A is False, $1\{A\} = 0$.
- $f^*(x)$: function (not DNN)
  - $f(x)$: continuous function as a target of approximation
  - $\bar{f}(x)$: piecewise constant function approximating $f(x)$
- $g_\theta^*(x)$: function by DNN (θ: parameter of DNN)
  - $g_\theta^{(L)}(x)$: DNN function with depth-L
  - $g_\theta^{[M]}(x)$: DNN function with width-M

Approximation power of DNNs with fewer layers (1-dim. input)

Step 1

Setup of shallow DNN

- DNN with 2 layers
  - $x \in [0,1]$: input (1-dim)
  - M: width (# of nodes per layer)
  - $\theta = \left\{(a_j, b_j, c_j)_{j=1}^{M}\right\}$: parameter



NN with depth-2 width-6

Function by DNN with 2 layer

$$g_\theta^{(2)}(x) = \sum_{j=1}^{M} c_j \sigma(a_j x + b_j)$$

Select the parameter $\theta = \left\{(a_j, b_j, c_j)_{j=1}^{M}\right\}$
to represent the target function

Universal Approximation Theorem (UAT)

L2-norm case:

$$\|f\| = \int_{[0,1]} f(x)^2 dx$$

Claim of UAT :

- DNNs can approximate any continuous function

Universal Approximation Theorem

For any continuous $f: [0,1] \to \mathbb{R}$ and $\varepsilon > 0$, there exists $g_\theta^{(2)}(x)$ satisfying the following:

$$\|f - g_\theta^{(2)}\| \leq \varepsilon.$$

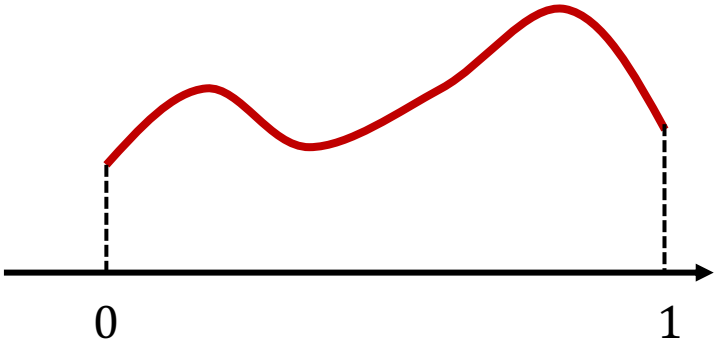
Approximated target function

DNN function

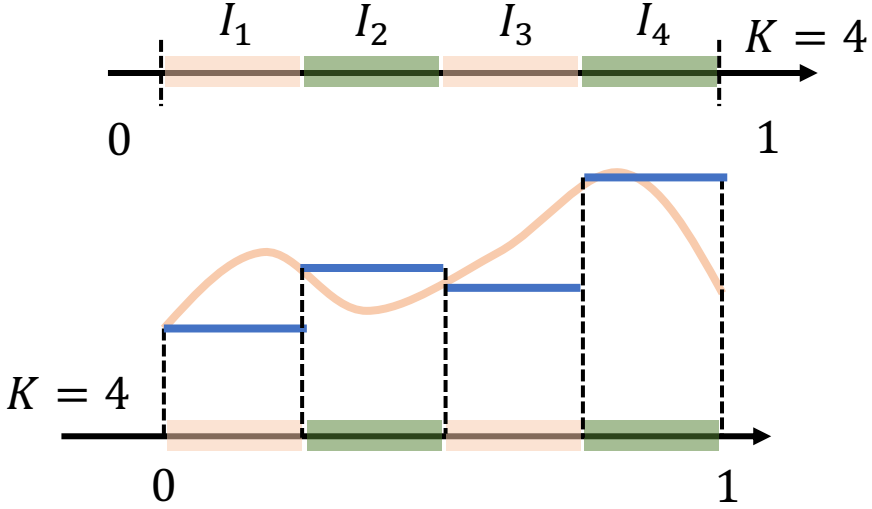
Preparation: Piecewise constant function

piece $I_k = \left[\frac{k-1}{K}, \frac{k}{K}\right)$

- Continuous functions can be approximated by piecewise constant functions
  - Kdivision of $[0,1]$



Continuous function $f(x)$



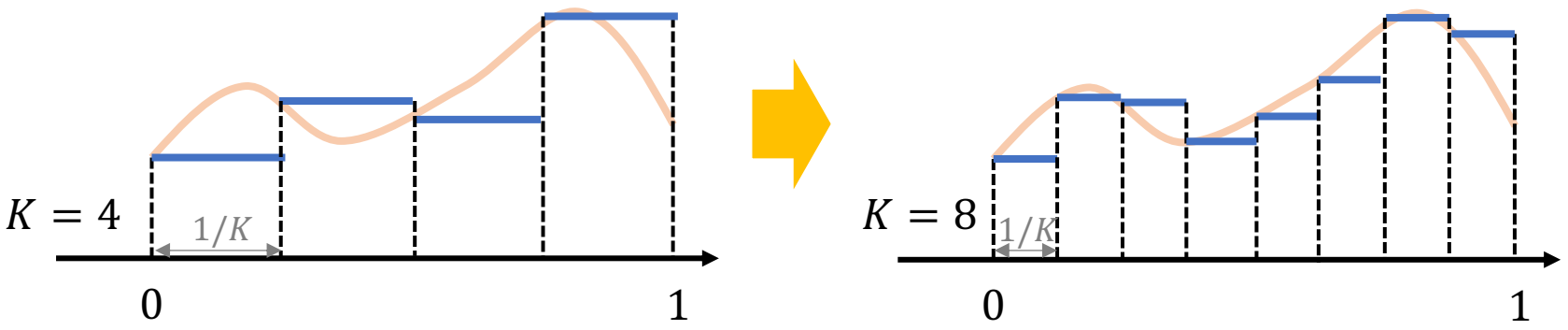
Piecewise constant function

$$\bar{f}(x) = \sum_{k=1,\ldots,K} w_k 1\{x \in I_k\}, w_k = f\left(\frac{k-1}{K}\right)$$

Piecewise constant functions are a mathematical concept (not NN).
NNs will approximate them.

Preparation: Piecewise constant function

- If the division is sufficiently fine $(K \to \infty)$, the approx. error can be arbitrarily small.



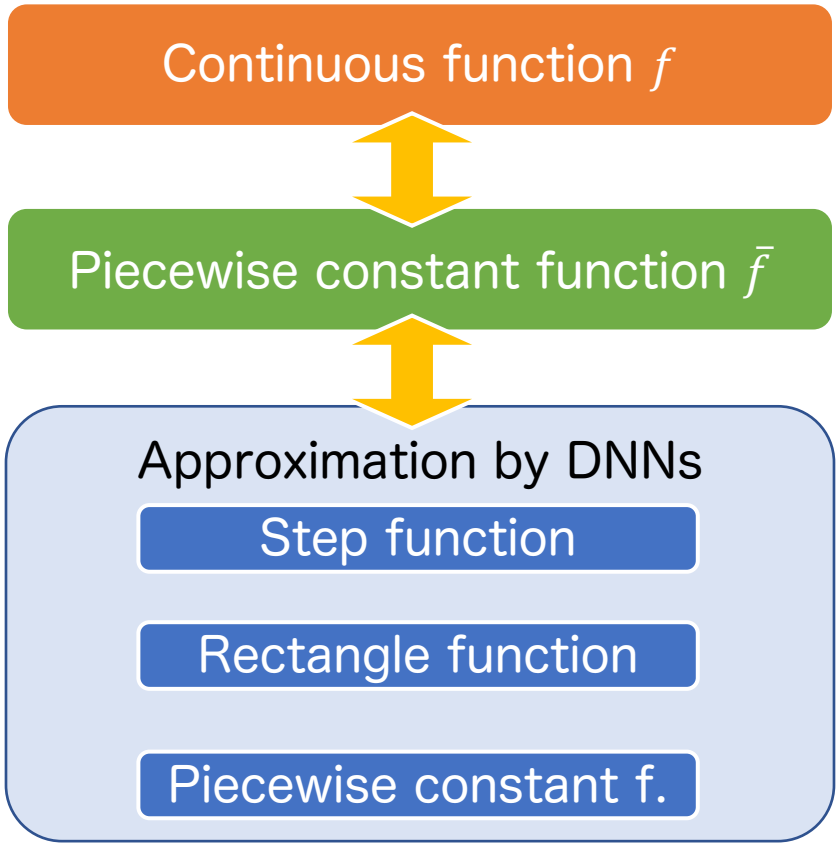
Since $f(x)$ is continuous,
for any $\varepsilon' > 0$, sufficiently large K gives $\left|f(x) - f\left(x + \frac{1}{K}\right)\right| \leq \varepsilon'$.

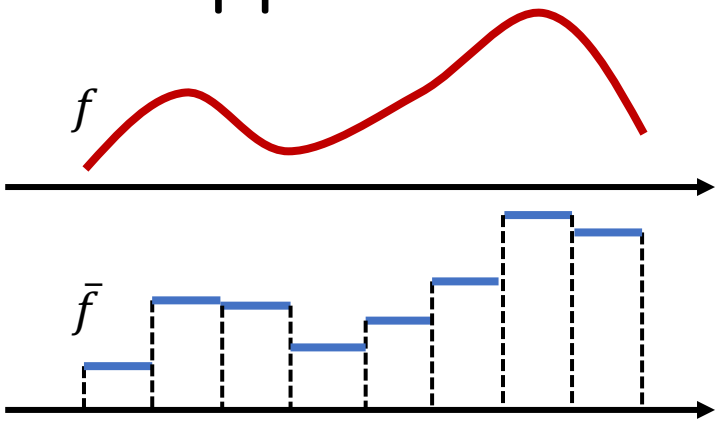
Error by piecewise constant approximation

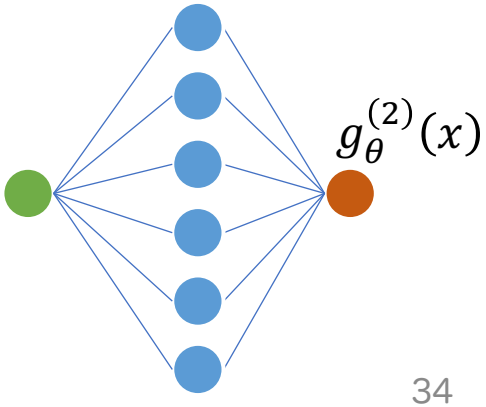
$f(x) \approx \bar{f}(x)$ as $K \to \infty$

Proof Outline of UAT

- Mathematical approx. + DNN approx.





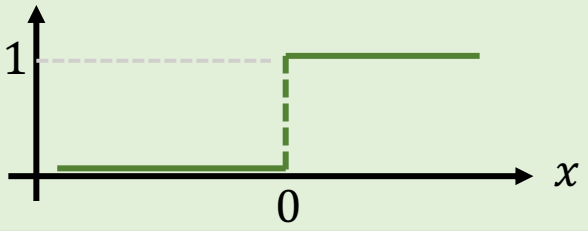


1. Step approximation

- Combination of ReLUs to make a step function

Def : Step function

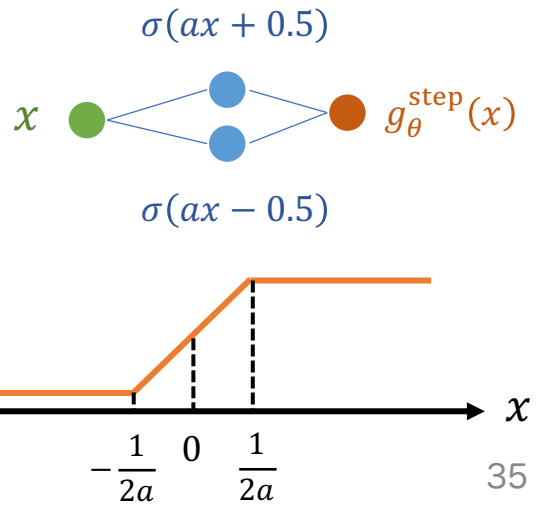
$$f^{\text{step}}(x) = 1\{x \geq 0\}$$



- Difference b/w two ReLUs
  - $a > 0$

$$g_\theta^{\text{step}}(x) \coloneqq \sigma(ax + 0.5) - \sigma(ax - 0.5)$$
$$= \max\{ax + 0.5, 0\} - \max\{ax - 0.5, 0\}$$
$$= \begin{cases} 1 & \text{if } x \geq \frac{1}{2a} \\ ax + 0.5 & \text{if } x \in \left(-\frac{1}{2a}, \frac{1}{2a}\right) \\ 0 & \text{if } x \leq -\frac{1}{2a} \end{cases}$$

$g_\theta^{\text{step}}(x)$: NN w/ depth-2 width-2

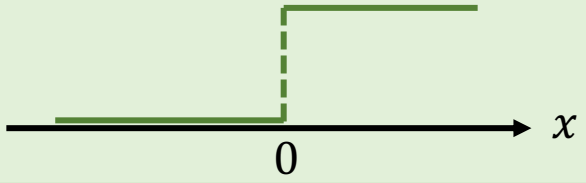


1. Step approximation

- Combination of ReLUs to make a step function

Def : Step function

$$f^{\text{step}}(x) = 1\{x \geq 0\}$$



- Increasing the slope a, then...

- Hence, as $a \to \infty$, we have

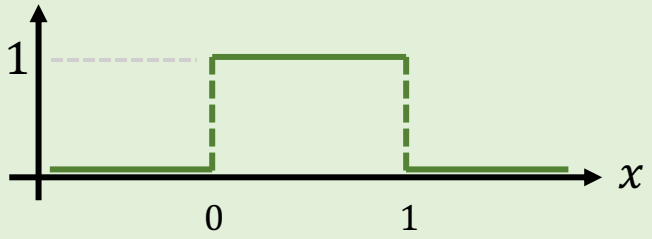
$$g_\theta^{\text{step}}(x) \approx f^{\text{step}}(x)$$

2. Rectangle approximation

- Use NN step function to make rectangular functions

Def : Rectangle function

$$f^{\mathrm{rec}}(x) = 1\{x \in [0,1]\}$$



- Method: Difference of two NN steps

$$f^{\mathrm{rec}}(x) = f^{\mathrm{step}}(x) - f^{\mathrm{step}}(x-1)$$

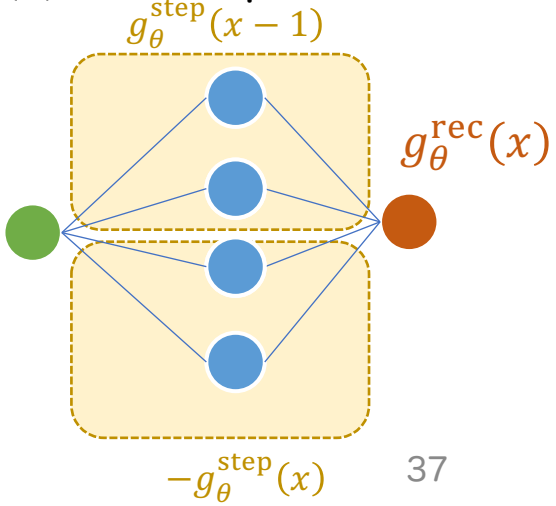
- Define NN rectangle based on this

$$g_\theta^{\mathrm{rec}}(x) := g_\theta^{\mathrm{step}}(x) - g_\theta^{\mathrm{step}}(x-1)$$

- As $a \to \infty$,

$$g_\theta^{\mathrm{rec}}(x) \approx f^{\mathrm{rec}}(x)$$

$g_\theta^{\mathrm{rec}}(x)$: NN depth-2 width-4



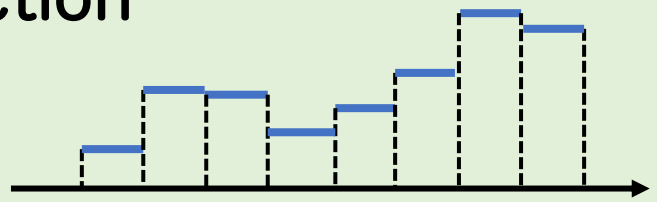
$I_k = \left[\frac{k-1}{K}, \frac{k}{K}\right)$: Division

3. Piecewise constant

- Align NN rectangular functions to form a piecewise constant function

Def : Piecewise constant function

$$\bar{f}(x) = \sum_{k=1,\dots,K} w_k 1\{x \in I_k\},$$



- Method: Align K NN rectangular functions

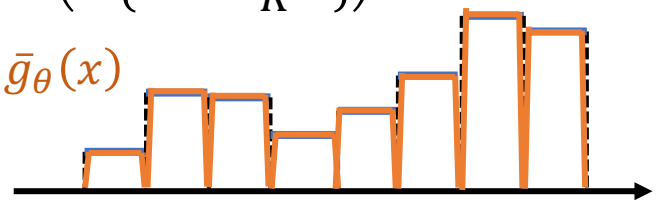
$$\bar{f}(x) = \sum_{k=1}^{K} w_k f^{\mathrm{rec}}\left(K\left\{x - \frac{k-1}{K}\right\}\right)$$

$= 1\{x \in I_k\}$

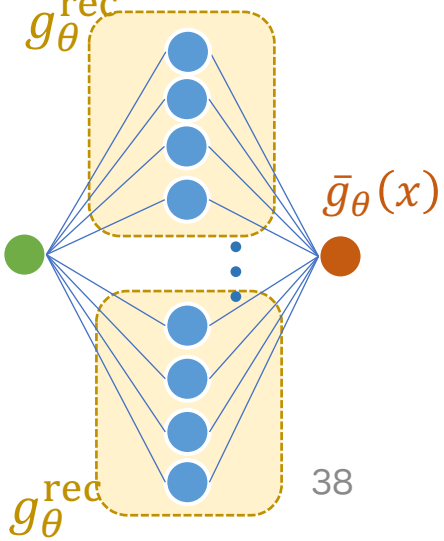
- Based on this

$$\bar{g}_\theta(x) = \sum_{k=1}^{K} w_k\, g_\theta^{\mathrm{rec}}\left(K\left\{x - \frac{k-1}{K}\right\}\right)$$

- As $a \to \infty$, $\bar{f}(x) \approx \bar{g}_\theta(x)$



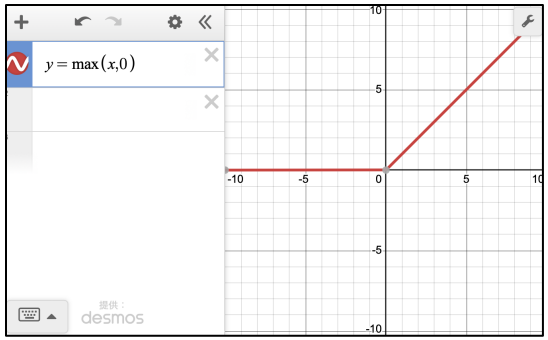
$\bar{g}_\theta(x)$: NN depth-2 width-4K



Exercise 1

Goal

- Use neural nets with ReLU to approximate step, rectangular, and piecewise constant functions.
  - Assign appropriate values to parameters



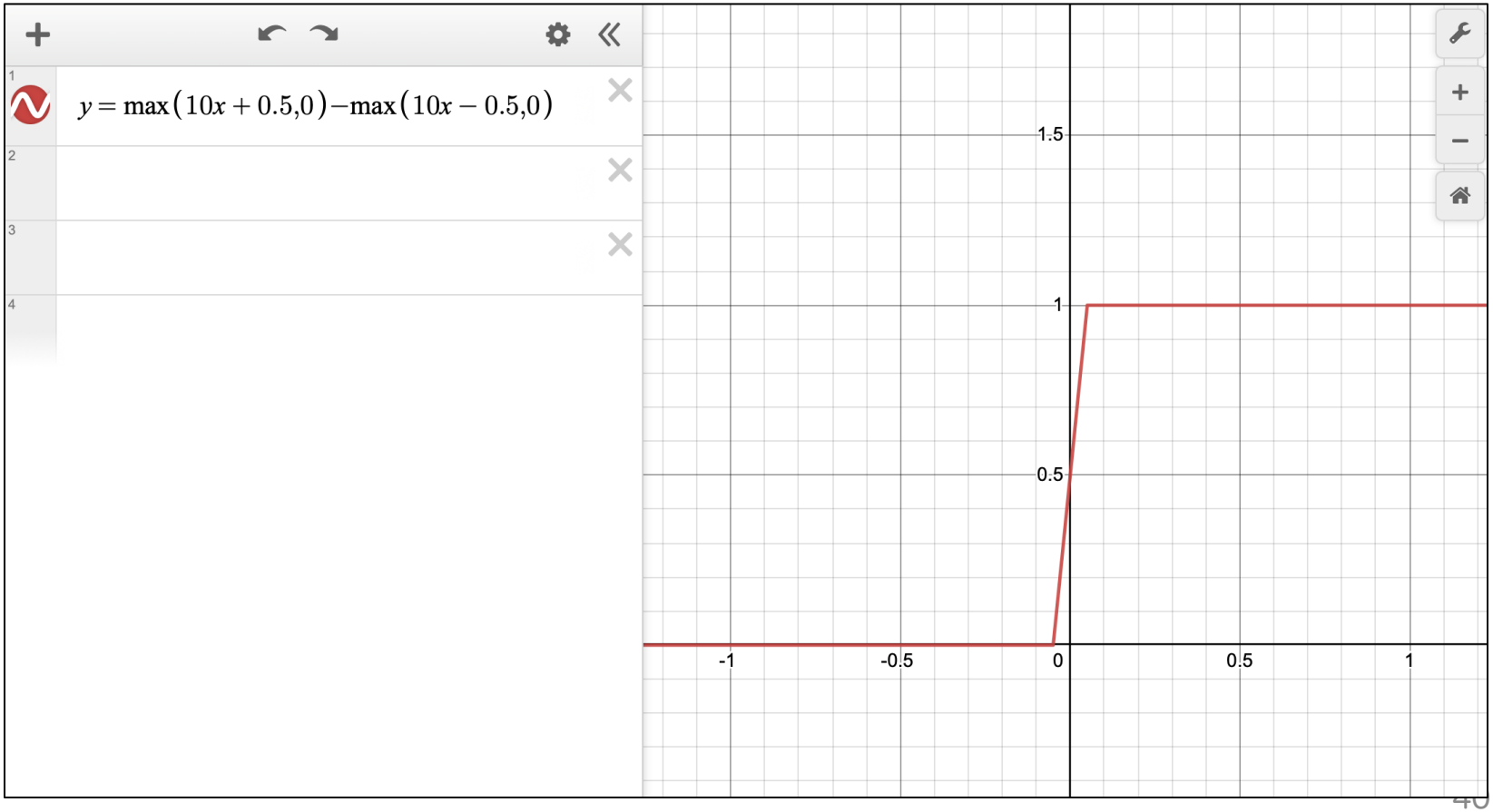
Approach

- Open Desmos and create functions using the max function
  - https://www.desmos.com/calculator
  - Put "$y = \max(x, 0)$"
  - Possible to define functions by "$f(x) = \max(x, 0)$"

- Mac users can use the preset application Grapher.

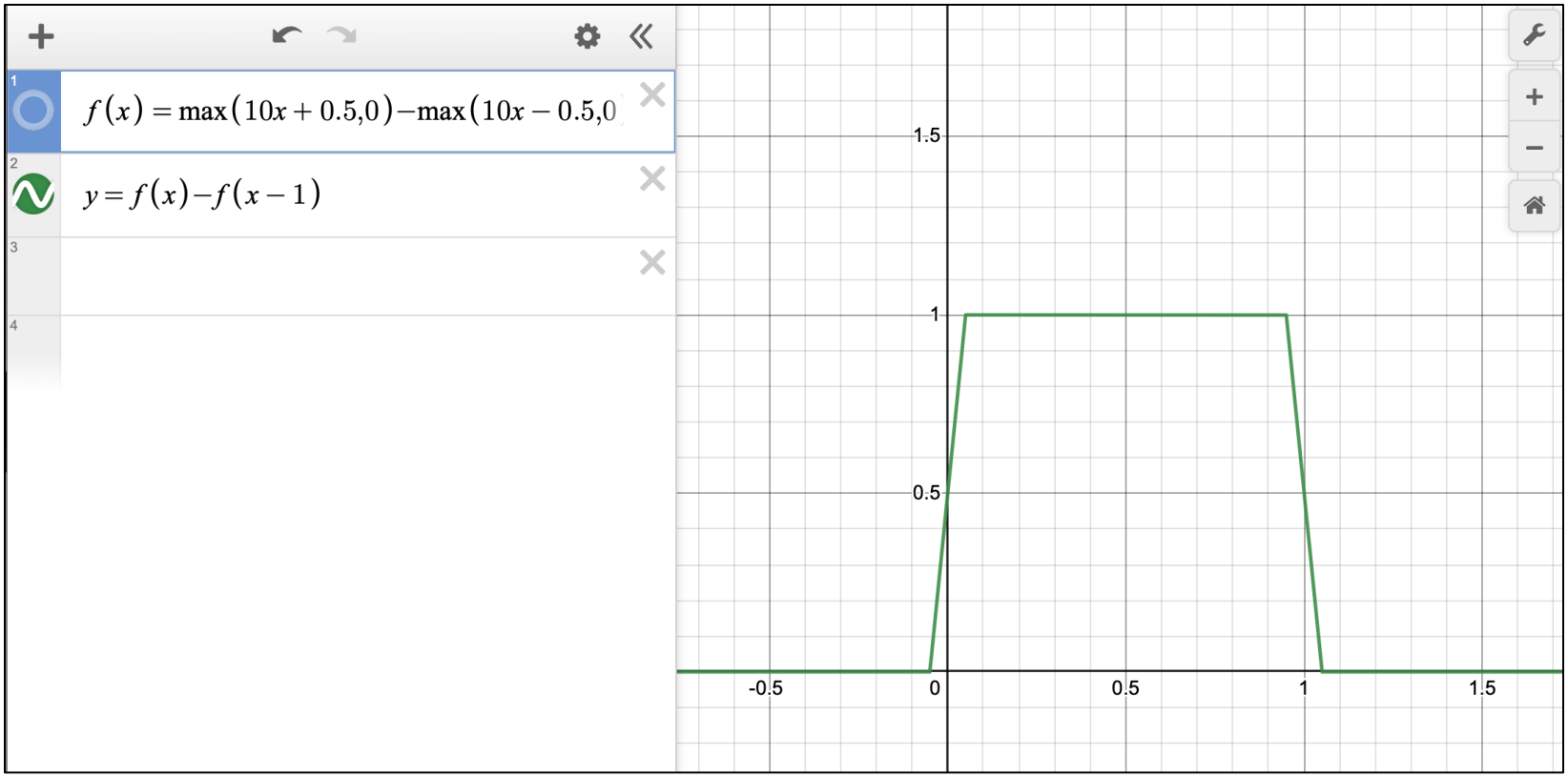
Solution 1

- Example (Step function)



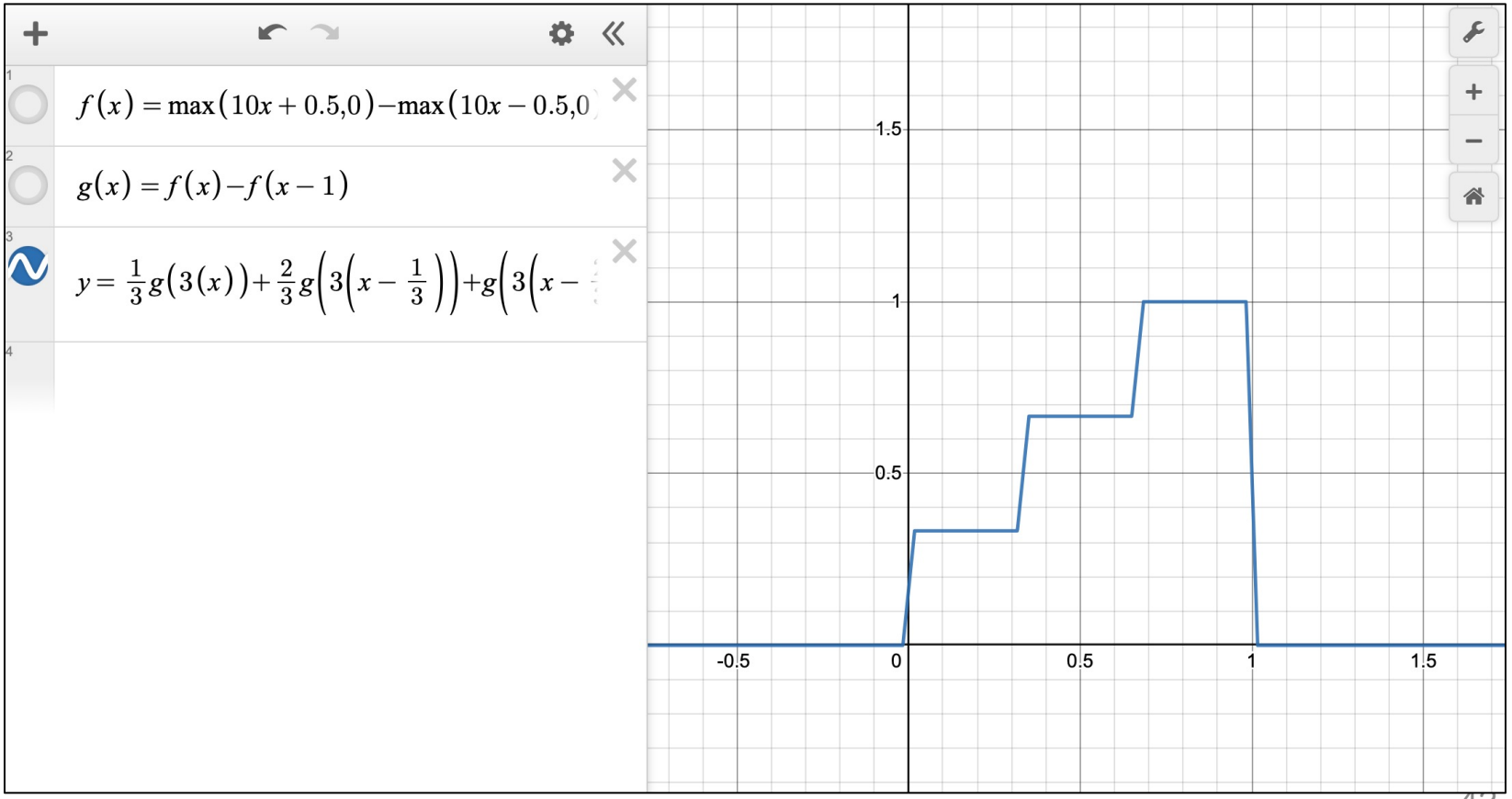
Solution 1

- Example (Rectangular function)



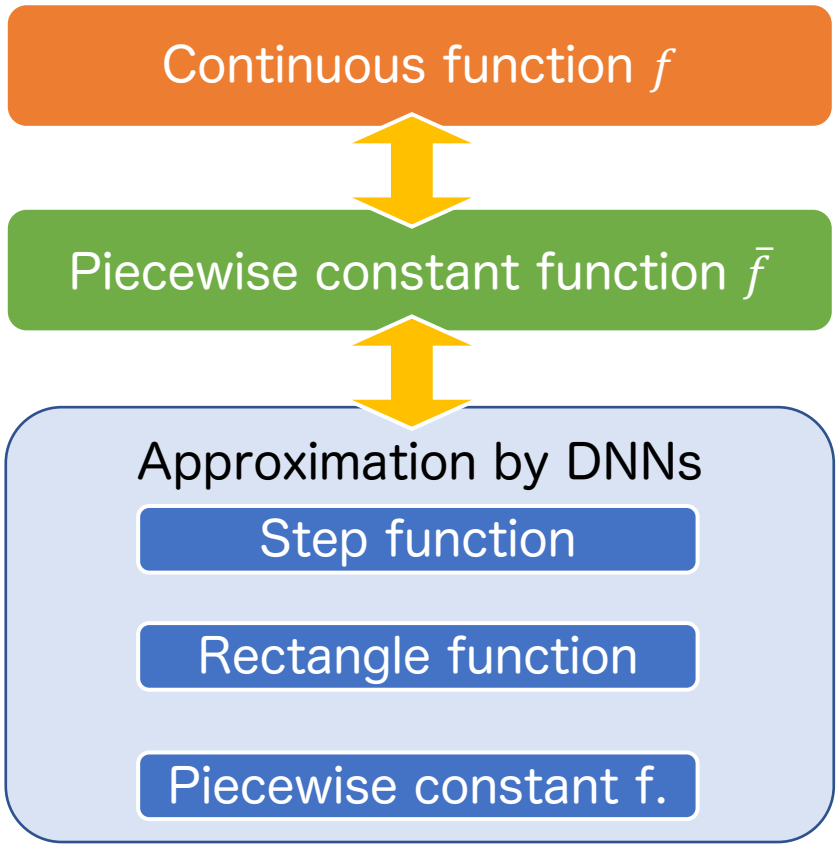
Solution 1

- Example (Piecewise Constant Functions)

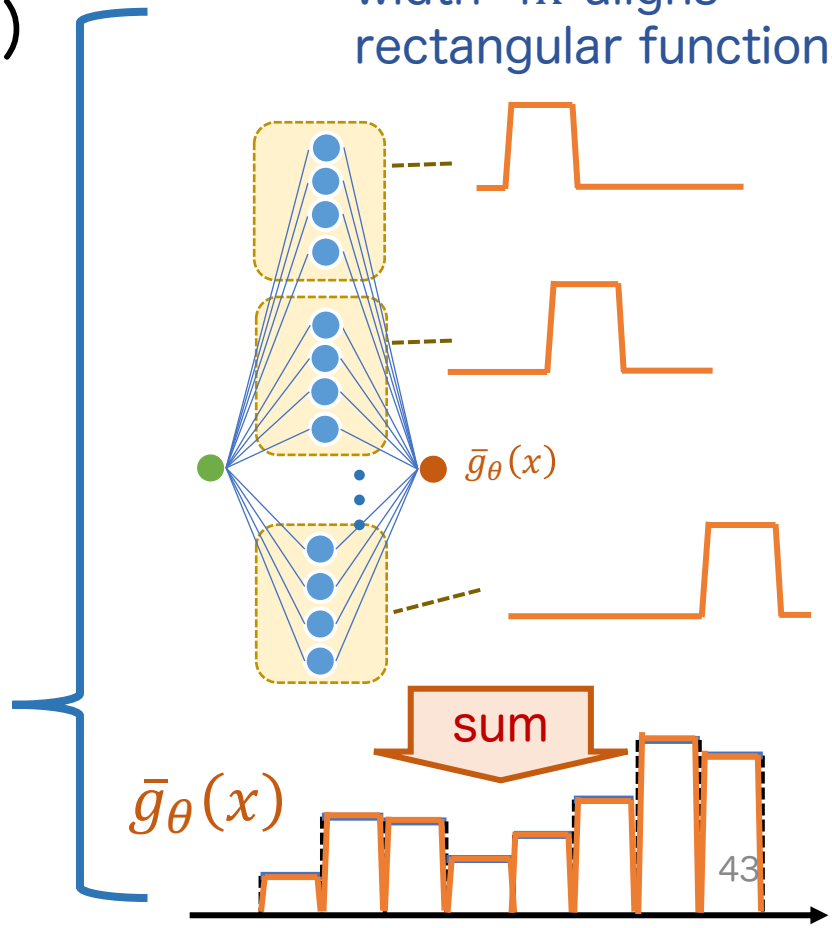


Summary of Step 1

- NN with depth-2 $g_\theta^{(2)}(x)$ can approximate continuous functions
  - Large width ($4K, K \to \infty$)



NN with depth-2 and width-$4K$ aligns rectangular functions



Easy?

Pretty easy?

Rather difficult?

Approximation power of DNNs with fewer layers (d-dim. input)

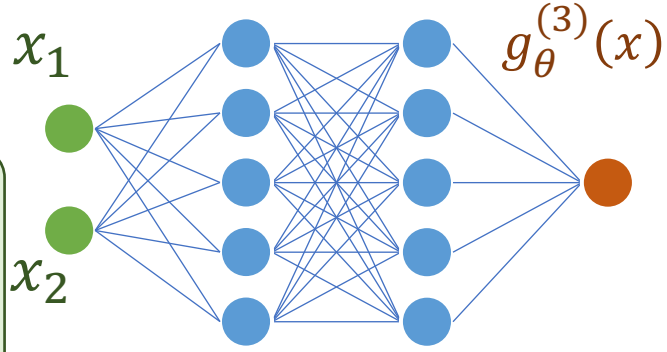
Step 2

Formulation of DNN

- 3 layers, d-dim input
  - $\boldsymbol{x} = (x_1, \ldots, x_d) \in [0,1]^d$: d-dim input
  - M: width (# of nodes per layer)
  - θ: parameter
    - $\theta = \left\{ \left(a_{ji}^{(1)}, b_j^{(1)}\right)_{j,i=1}^{M,d}, \left(a_{jj'}^{(2)}, b_{j'}^{(2)}\right)_{j,j'=1}^{M,M}, \left(c_{j'}\right)_{j'=1}^{M} \right\}$

DNN with 3 layers

$$g_\theta^{(3)}(\boldsymbol{x}) = \sum_{j'=1}^{M} c_{j'} \sigma\left(\sum_{j=1}^{M} a_{jj'}^{(2)} \sigma\left(\sum_{i=1}^{d} a_{ji}^{(1)} x_i + b_j^{(1)} \right) + b_{j'}^{(1)} \right)$$



DNN with depth-3 width-5
(2-dim input)

Select the parameter θ
to approximate the target function

Universal Approximation Theorem (UAT)

Claim of UAT :

- Similar statement holds and d-dim. Input by DNN with depth-3

Universal Approximation Theorem

For any continuous $f: [0,1]^d \to \mathbb{R}$ and $\varepsilon > 0$, there exists $g_\theta^{(3)}(x)$ satisfying the following:

$$\|f - g_\theta^{(3)}\| \leq \varepsilon.$$

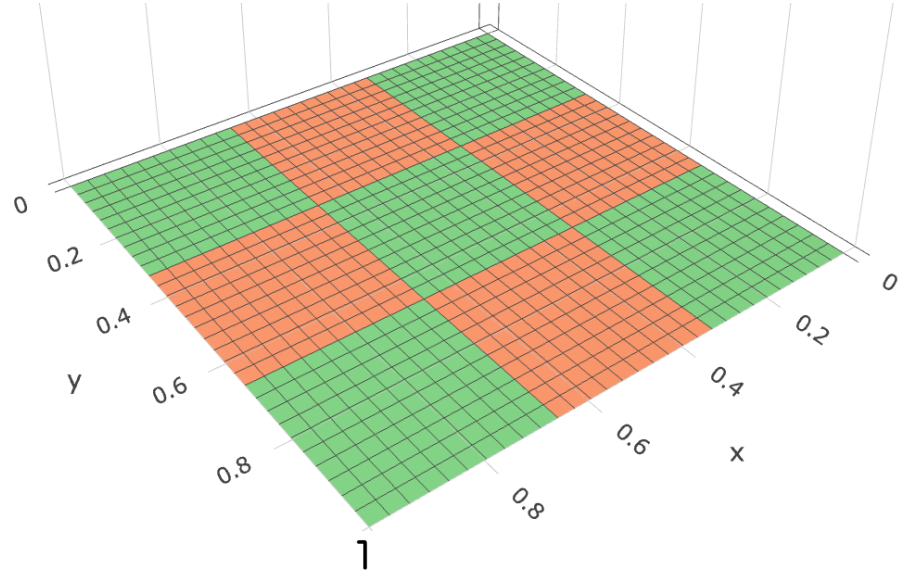
Approximated target function

DNN function

Preparation: Piecewise constant approx.

- Implement piecewise constant approx. in d-dim. space
  - Only the case $d = 2$ is presented in the lecture
- `Preparation: K^d division of $[0,1]^d$

Division with $d = 2, K = 3$



Notation for Division

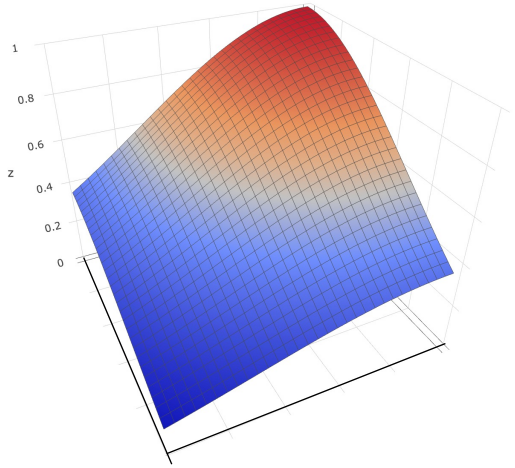
$$I_{k,k'} = \left[\frac{k-1}{K}, \frac{k}{K}\right) \times \left[\frac{k'-1}{K}, \frac{k'}{K}\right)$$

$$k, k' = 1, \ldots, K$$

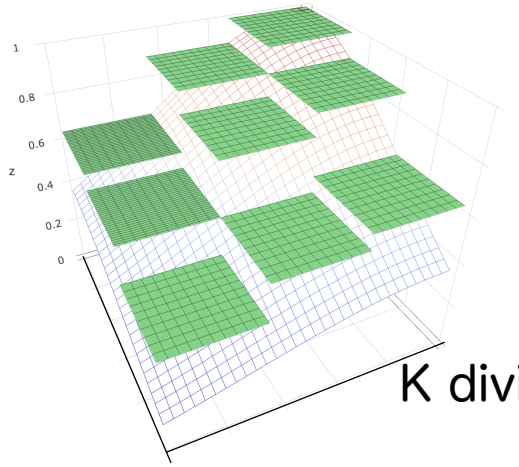
Divide the support into K^d pieces

Preparation: Piecewise constant approx.

- The approx. is also valid in d-dim.
  - W/ sufficiently fine division $(K \to \infty)$



Continuous function $f(\boldsymbol{x})$



K divisions for each axis
$(K = 3)$

Piecewise constant function for approximation

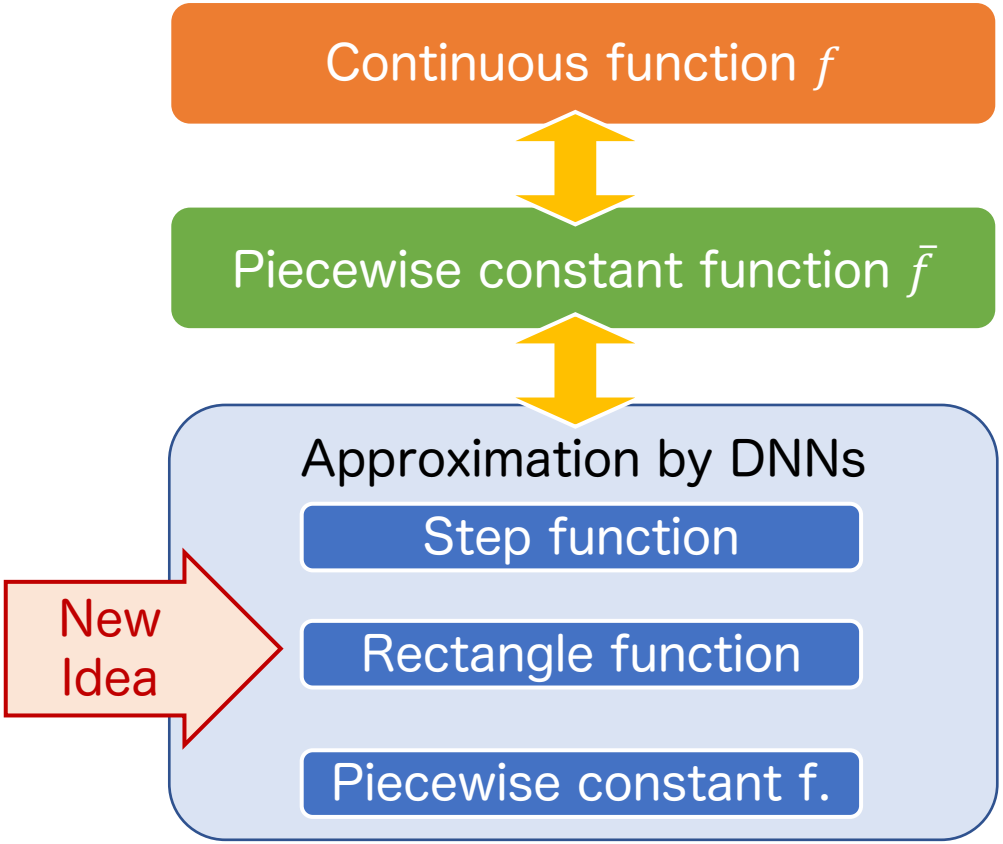
$$\bar{f}(\boldsymbol{x}) = \sum_{k,k'=1}^{K} w_{kk'} 1\{x \in I_{kk'}\}, w_{kk'} = f\left(\frac{k-1}{K}, \frac{k'-1}{K}\right)$$

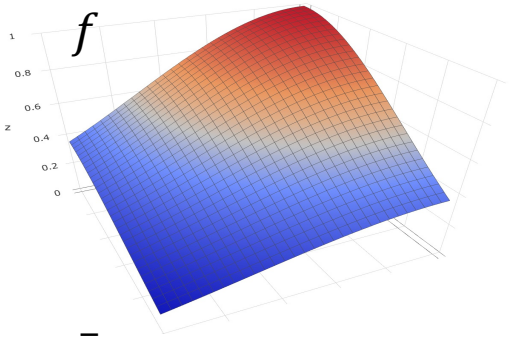
Since $f(\boldsymbol{x})$ is continuous, we have

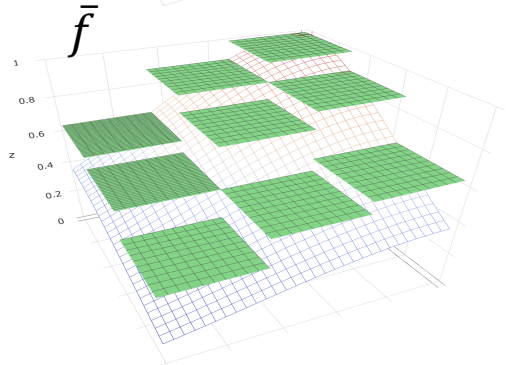
$$f(\boldsymbol{x}) \approx \bar{f}(\boldsymbol{x}) \text{ as } K \to \infty$$

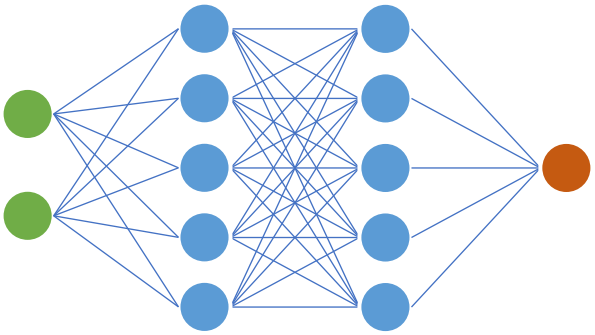
Proof Outline

- Similar to 1-dim. case







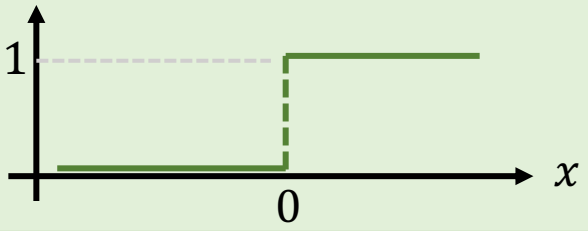


1. Step approximation (recap)

- Combination of ReLUs to make a step function

Def : Step function

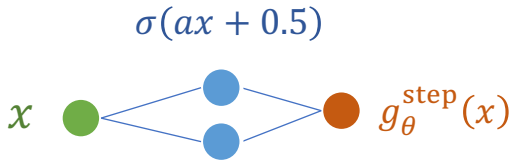
$$f^{\text{step}}(x) = 1\{x \geq 0\}$$

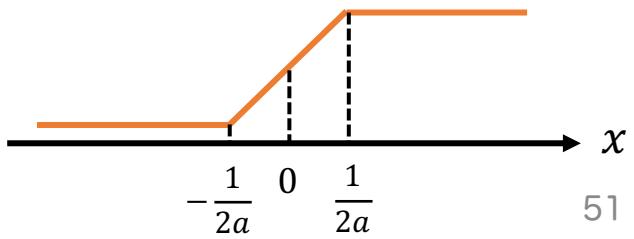


- Difference b/w two ReLUs
  - $a > 0$

$$g_\theta^{\text{step}}(x) := \sigma(ax + 0.5) - \sigma(ax - 0.5)$$
$$= \max\{ax + 0.5, 0\} - \max\{ax - 0.5, 0\}$$
$$= \begin{cases} 1 & \text{if } x \geq \frac{1}{2a} \\ ax + 0.5 & \text{if } x \in \left(-\frac{1}{2a}, \frac{1}{2a}\right) \\ 0 & \text{if } x \leq -\frac{1}{2a} \end{cases}$$

$g_\theta^{\text{step}}(x)$: NN w/ depth-2 width-2





2. d-dim rectangle approx.

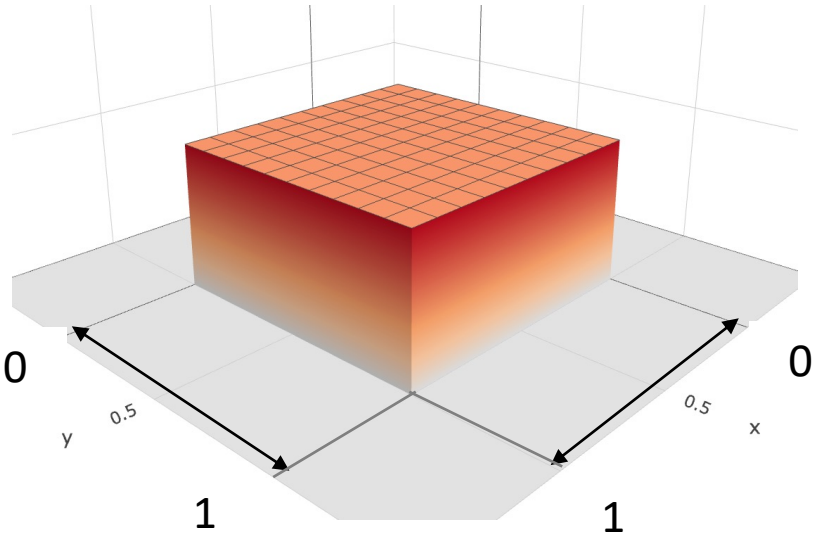
- Create a rectangular function d-dim. space

Def : Rectangle function (d-dim.)

$$f^{\mathrm{drec}}(\boldsymbol{x}) = 1\{\boldsymbol{x} \in [0,1]^d\}, \quad \boldsymbol{x} \in \mathbb{R}^d$$

- Multidimensional extension
- Stretch it to create a constant function on

$$I_{kk'} = \left[\frac{k-1}{K}, \frac{k}{K}\right) \times \left[\frac{k'-1}{K}, \frac{k'}{K}\right)$$



Exercise 2

Goal

- Set $d = 2$. Consider a DNN that can approximate an d-dim. rectangular function

Hint

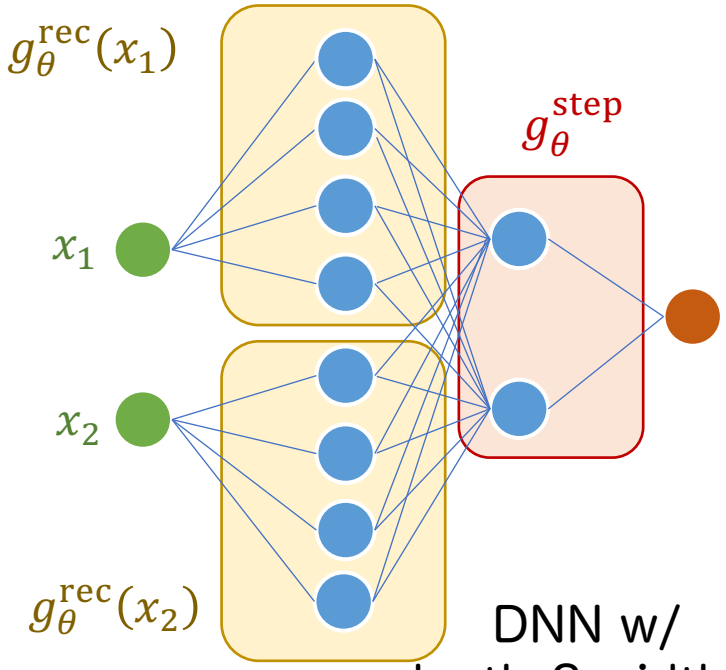
- When $d = 1$,

$$g_\theta^{\mathrm{rec}}(x) \coloneqq g_\theta^{\mathrm{step}}(x) - g_\theta^{\mathrm{step}}(x-1)$$

- When $d = 2$, we need more layers.
  - Use the step function $g_\theta^{\mathrm{step}}(x)$ and 1-dim rectangular function$g_\theta^{\mathrm{rec}}(x)$.

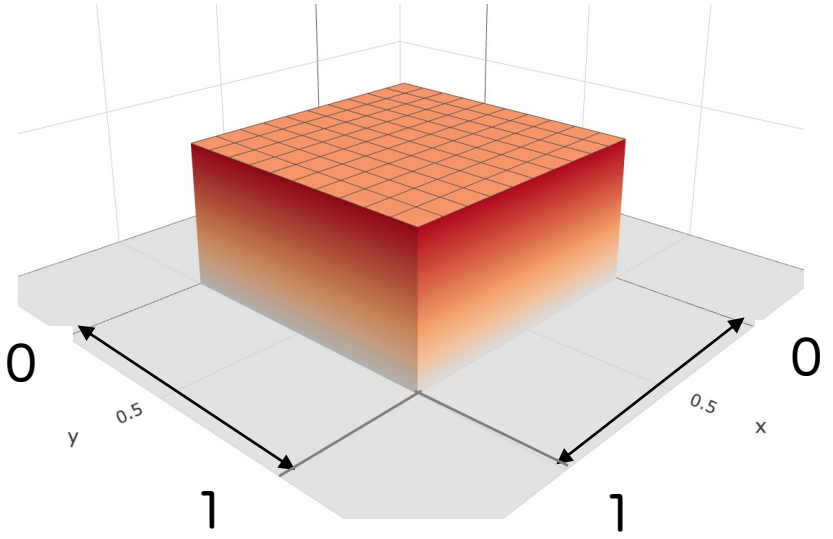
Solution 2

- With 2-dim input $\boldsymbol{x} = (x_1, x_2)$,
  $g_\theta^{\text{step}}(g_\theta^{\text{rec}}(x_1) + g_\theta^{\text{rec}}(x_2) - 1.5)$



DNN w/
depth-3 width-8

- Since $g_\theta^{\text{rec}}(x_1) \approx 1\{x_1 \in [0,1]\}$,
  - If $x_1, x_2 \in [0,1]$, then
    $g_\theta^{\text{rec}}(x_1) + g_\theta^{\text{rec}}(x_2) - 1.5 = 0.5 > 0$
  - If $x_1, \in [0,1], x_2 \notin [0,1]$, then
    $g_\theta^{\text{rec}}(x_1) + g_\theta^{\text{rec}}(x_2) - 1.5 = -0.5 < 0$
  - If $x_1, x_2 \notin [0,1]$, then
    $g_\theta^{\text{rec}}(x_1) + g_\theta^{\text{rec}}(x_2) - 1.5 = -1.5 < 0$



  - If z is positive, $g_\theta^{\text{step}}(z) = 1$.

2. d-dim rectangle approx.

- Create a rectangular function on d-dim. space

Def : Rectangle function (d-dim.)

$$f^{\mathrm{drec}}(\boldsymbol{x}) = 1\{\boldsymbol{x} \in [0,1]^d\}, \quad \boldsymbol{x} \in \mathbb{R}^d$$

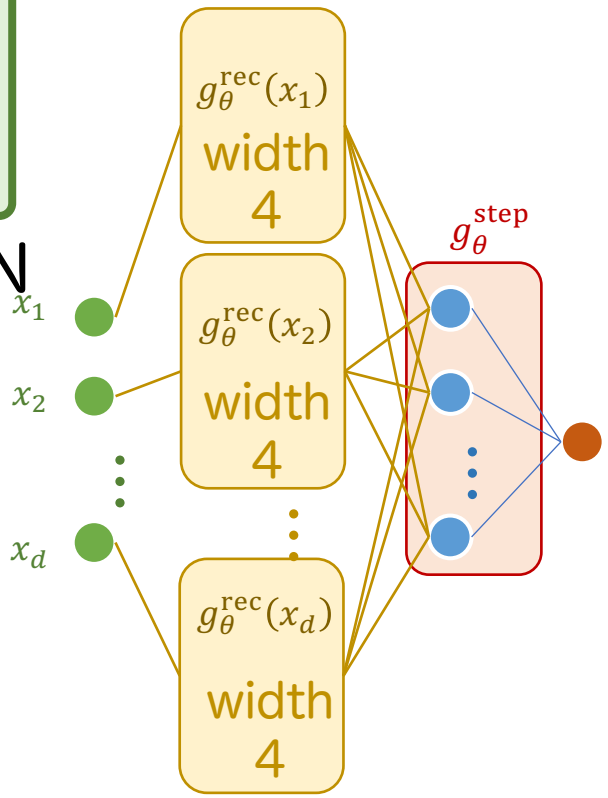
- d-dim. rectangular function by DNN

$$g_\theta^{\mathrm{drec}}(\boldsymbol{x}) \coloneqq g_\theta^{\mathrm{step}}\left(\sum_{j=1}^{d} g_\theta^{\mathrm{rec}}(x_j) - (d - 0.5)\right)$$

Rectangular function in each axis

  - Only if $x_j \in [0,1]$ for all $j(= 1, \ldots, d)$, $\sum_{j=1}^{d} g_\theta^{\mathrm{rec}}(x_j) - (d - 0.5) = 0.5 > 0$.

- Thus, $g_\theta^{\mathrm{drec}}(\boldsymbol{x}) \approx f^{\mathrm{drec}}(\boldsymbol{x})$



DNN with depth-3 width-4d

Piece in 2-dim. space

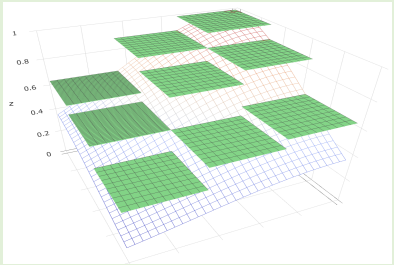
$$I_{kk'} = \left[\frac{k-1}{K}, \frac{k}{K}\right] \times \left[\frac{k'-1}{K}, \frac{k'}{K}\right]$$

3. Piecewise constant approx.

- Align rectangular functions to form a piecewise constant function

Def： partitioned constant function

$$\bar{f}(\boldsymbol{x}) = \sum_{k,k'=1}^{K} w_{kk'} 1\{x \in I_{kk'}\},$$



- Method: align K^d rectangular functions
  - d=2 case

$$\bar{g}_\theta(\boldsymbol{x}) = \sum_{k,k'=1}^{K} w_{kk'}\, g_\theta^{\mathrm{drec}}\left(K\left\{x_1 - \frac{k-1}{K}\right\}, K\left\{x_2 - \frac{k'-1}{K}\right\}\right)$$

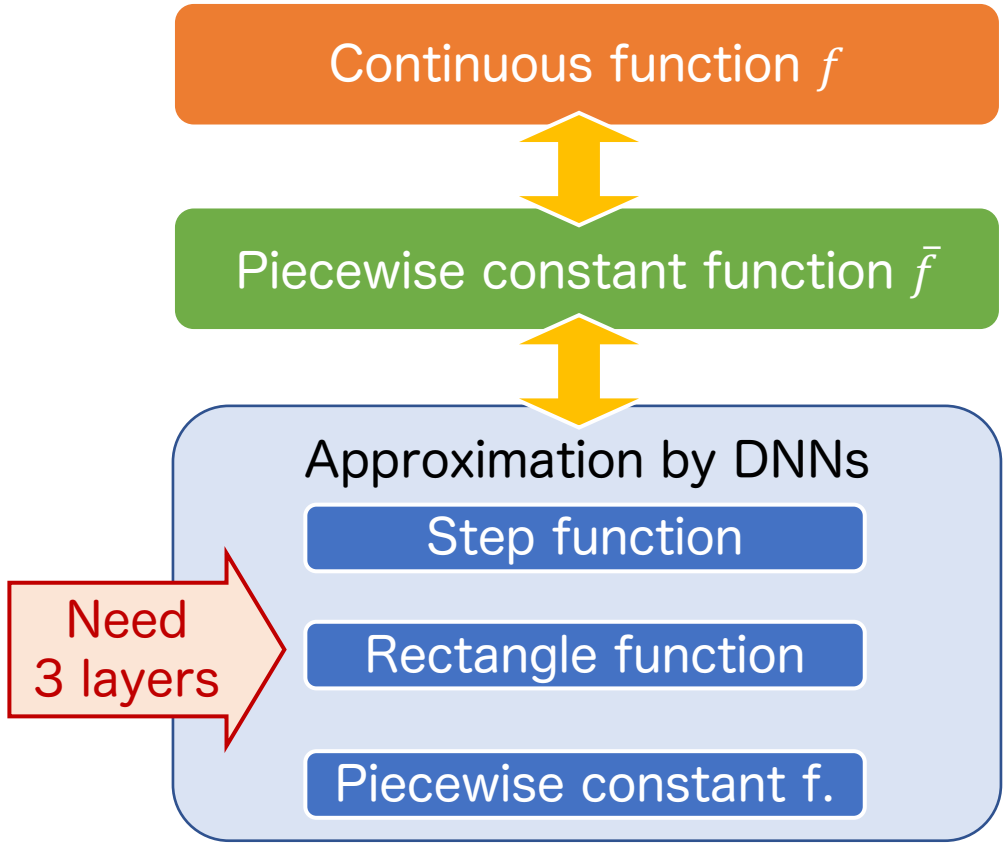
Stretch $[0,1]^2$ to represent $I_{kk'}$

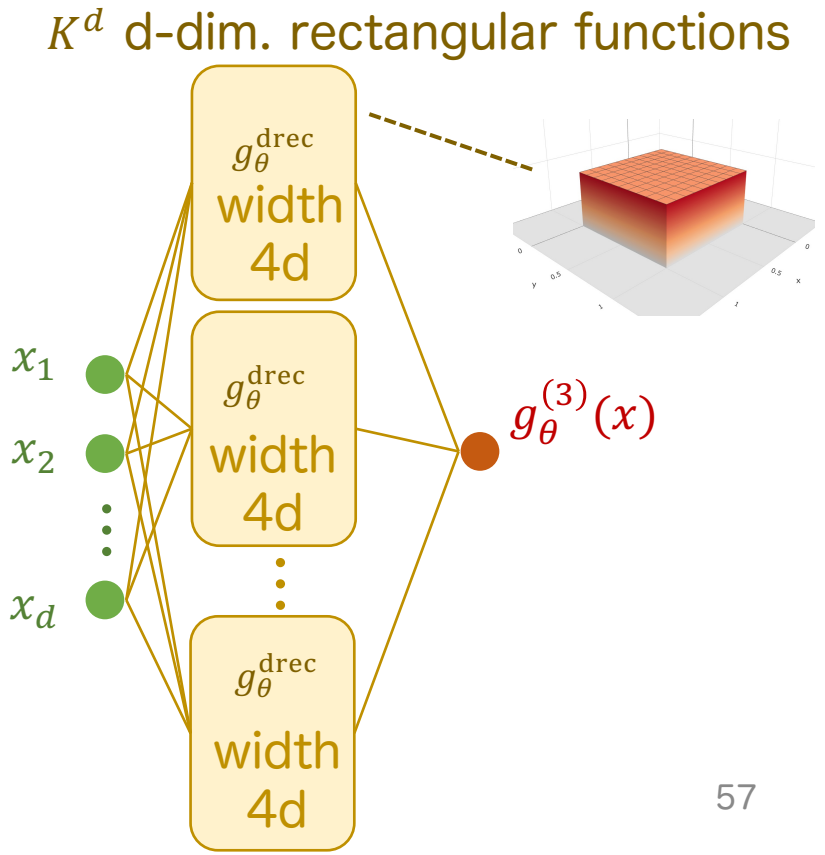
$$\rightarrow g_\theta^{\mathrm{drec}}\left(K\left\{x_1 - \frac{k-1}{K}\right\}, K\left\{x_2 - \frac{k'-1}{K}\right\}\right) \approx 1\{\boldsymbol{x} \in I_{kk'}\}$$

- Thus, $\bar{f}(\boldsymbol{x}) \approx \bar{g}_\theta(\boldsymbol{x}) =: g_\theta^{(3)}(\boldsymbol{x})$

Summary of Step 2

- DNN with depth-3 $g_\theta^{(3)}(\boldsymbol{x})$ approximates continuous functions
  - Width is too large ($4dK^d$, $K \to \infty$)





Easy?

Pretty easy?

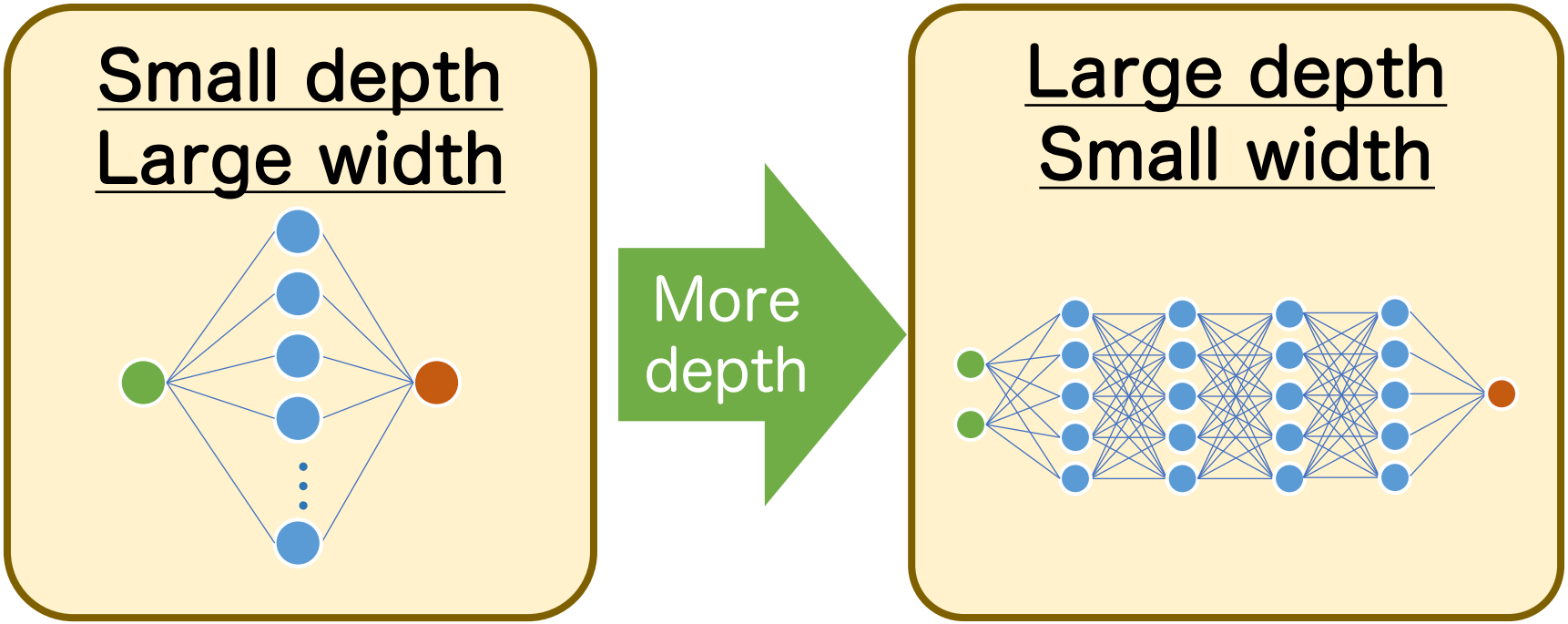
Rather difficult?

Approximation power of DNNs with many layers (less width)

Step 3

DNNs of finite width

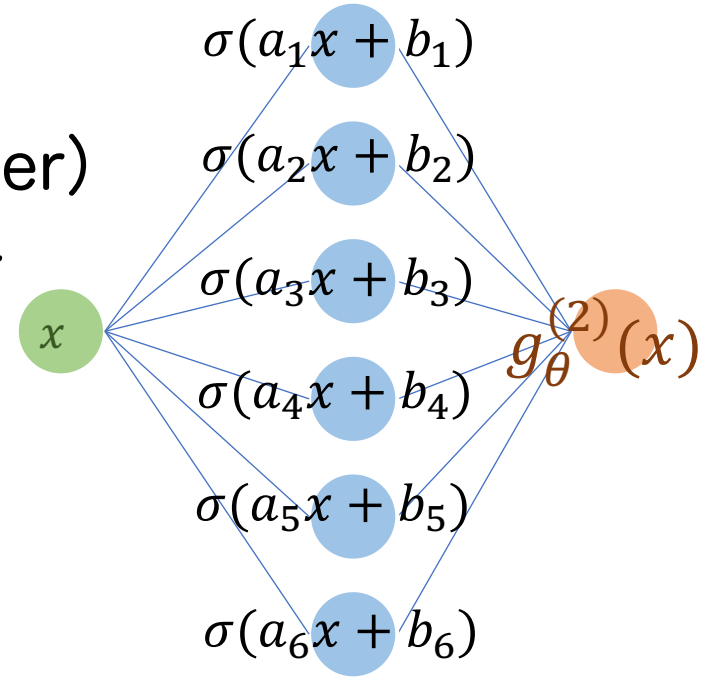
- The universal approximation theorems so far are for DNNs with large widths and few layers



Q: Does UAT holds even if width is not increased?

Setup of shallow NN

- DNN with 2 layers
  - $x \in [0,1]$: input (1-dim)
  - M: width (# of nodes per layer)
  - $\theta = \left\{(a_j, b_j, c_j)_{j=1}^{M}\right\}$: parameter



NN with depth-2 width-6

Function by DNN with 2 layer

$$g_\theta^{(2)}(x) = \sum_{j=1}^{M} c_j \sigma(a_j x + b_j)$$

Select the parameter $\theta = \left\{(a_j, b_j, c_j)_{j=1}^{M}\right\}$
to represent the target function

Setup of DNN

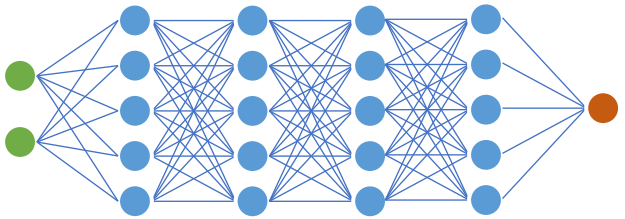
- DNN with depth-L
  - $\boldsymbol{x} = (x_1, \ldots, x_d) \in [0,1]^d$: d-dim. input
  - M: width
  - L: depth (# of layers)
  - $A_\ell \in \mathbb{R}^{M \times M}, b_\ell \in \mathbb{R}^m$: parameters

DNN with depth-L width-M

$$g_\theta^{[M]}(\boldsymbol{x}) = g_\theta^L \circ g_\theta^{L-1} \circ \cdots \circ g_\theta^1(\boldsymbol{x})$$

$\ell(= 1, \ldots, L)$-th function

$$g_\theta^\ell(\boldsymbol{z}) = \sigma(A_\ell \boldsymbol{z} + b_\ell)$$



DNN
depth-5 width-5

Universal Approximation Theorem (UAT)

Claim of UAT :

- Universal approximation is possible even with finite width

Universal Approximation Theorem

For any continuous $f: [0,1]^d \to \mathbb{R}$ and $\varepsilon > 0$, there exists $g_\theta^{[M]}(x)$ with width $M = 2d + 2$ satisfying the following:

$$\|f - g_\theta^{[M]}\| \leq \varepsilon.$$

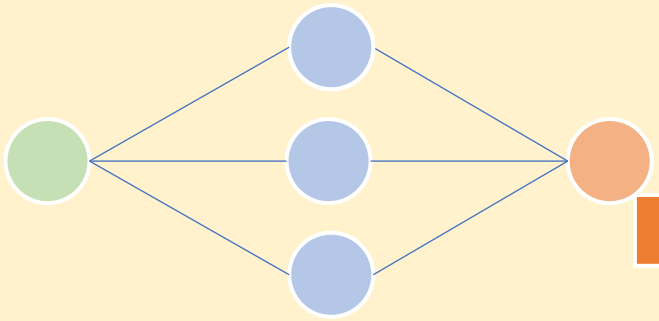
Approximated target function

DNN function

Proof Outline

- Rearranging neurons of shallow NN
  → Represent the same function by another DNN

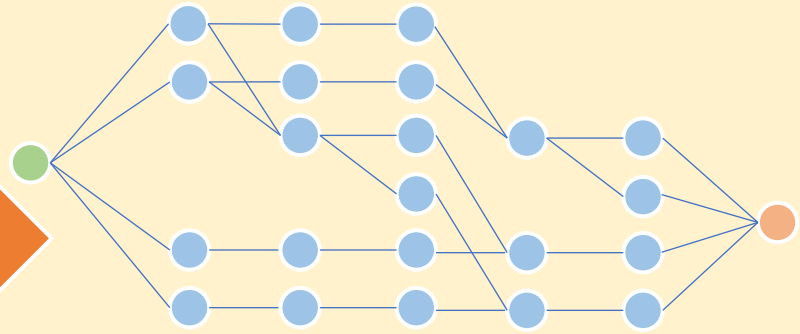
Phase 1： Create shallow NN to approximate f



NN with less layers $g_\theta^{(2)}(\boldsymbol{x})$

rearrange

Phase 2：Arrange nodes of the shallow NN to DNN



Deep (and less width) DNN $g_\theta^{[M]}(\boldsymbol{x})$

If shallow NN $g_\theta^{(2)}(\boldsymbol{x})$ satisfies UAT, and also $g_\theta^{(2)}(\boldsymbol{x}) = g_\theta^{[M]}(\boldsymbol{x})$ holds, the DNN also satisfies UAT.

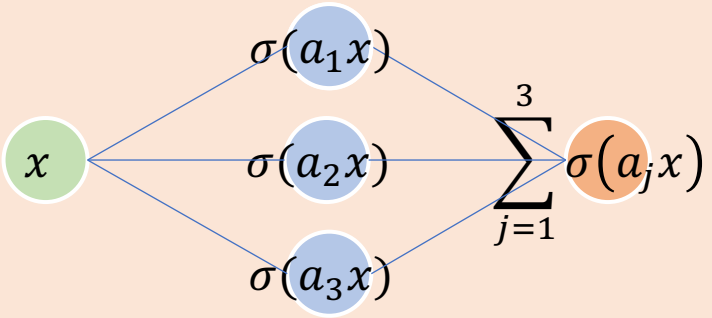
Rearrangement nodes

- Rearrange NN with depth-2 width-3 to Deep NN with more layers
  - Set $d = 1$

NN with depth-2 width-3

It has 3 nodes

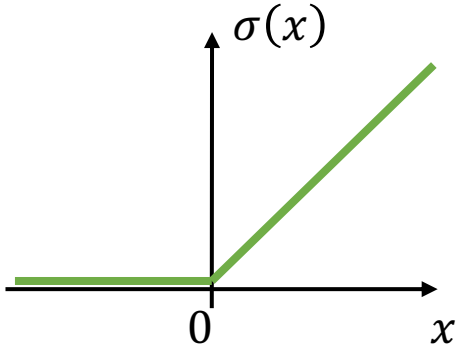
$$g_\theta^{(2)}(x) = \sum_{j=1,\dots,3} \sigma(a_j x)$$



Some parameters fixed for simplicity. ($c_j = 1, b_j = 0$)

(This simplification can be relaxed but omitted.)

Facts of ReLU



- Use the following facts of ReLU $\sigma(x) = \max\{x, 0\}$

Fact 1 : Identity function

$$-\sigma(-x) + \sigma(x) = x$$

Fact 2 : Homogeneity (1st order)

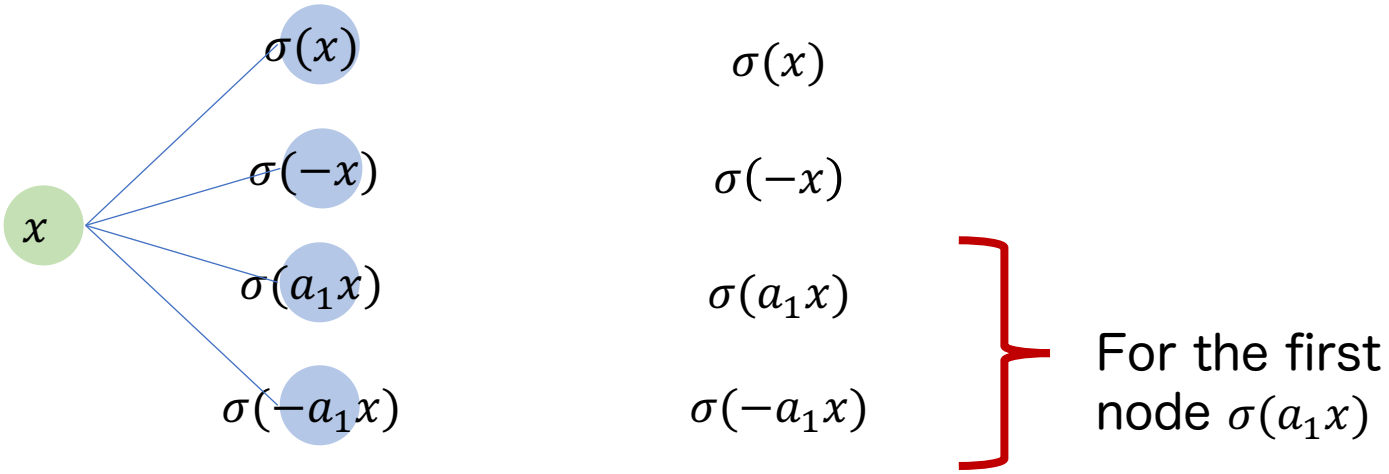
$$\sigma(ax) = a\sigma(x). \quad (a \geq 0)$$

Fact 3 : About composition

$$\sigma\big(\sigma(x)\big) = \sigma(x)\,, \sigma\big(\sigma(x) + \sigma(x')\big) = \sigma(x) + \sigma(x')$$

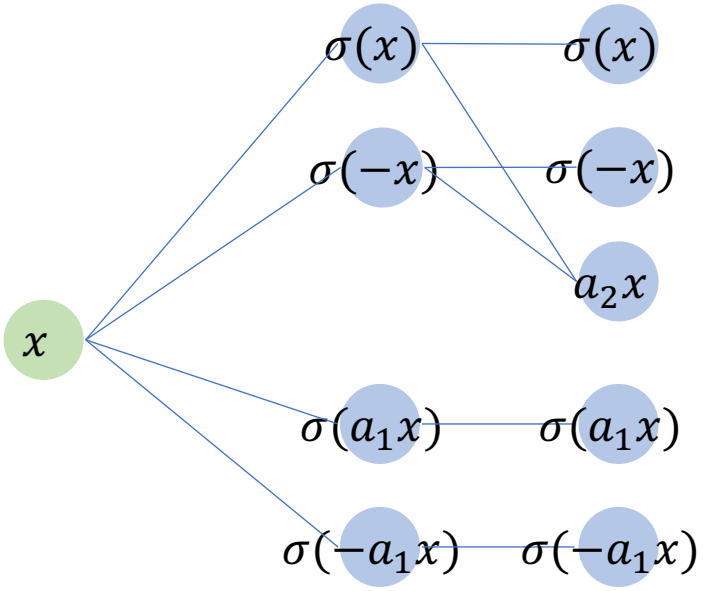
Rearranging 1

- Construct 1st layer
  - Represent the first node $\sigma(a_1 x)$ of pre-transformed shallow NN



Rearranging 2

- Construct 2nd layer
  - Represent $a_2 x$ for the second node



$$\sigma(x) = \sigma(\sigma(x))$$ Fact 3

$$\sigma(-x) = \sigma(\sigma(-x))$$ Fact 3

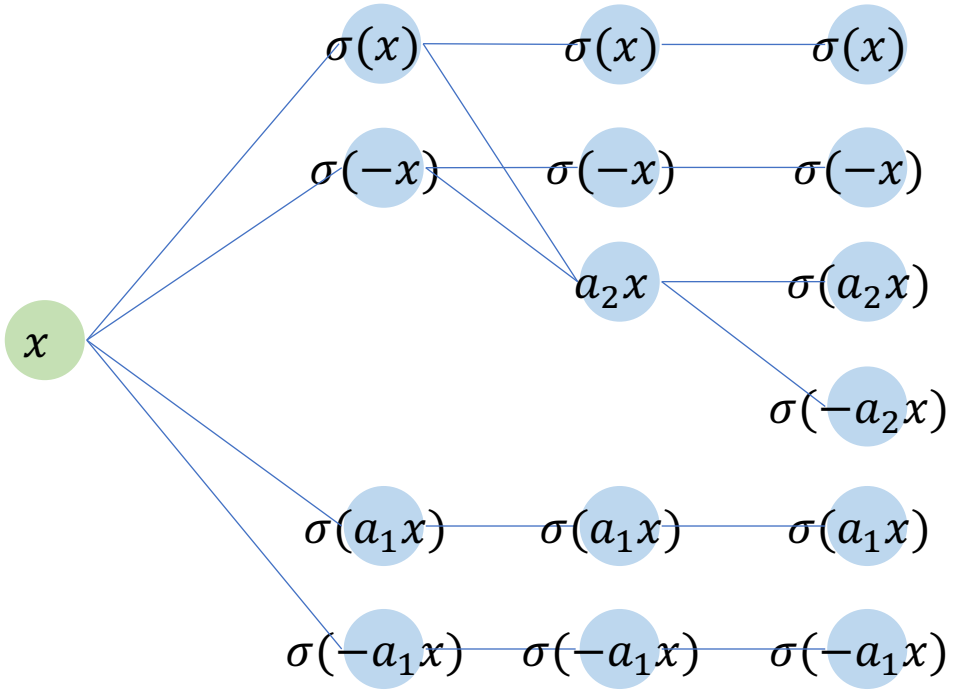
$$a_2 x = a_2\sigma(x) - a_2\sigma(-x)$$ Fact 1

$$\sigma(a_1 x) = \sigma(\sigma(a_1 x))$$ Fact 3

$$\sigma(-a_1 x) = \sigma(\sigma(-a_1 x))$$ Fact 3

Rearranging 3

- Construct 3rd layer
  - Represent $\sigma(a_2x), \sigma(-a_2x)$ for the second node



$\sigma(x) = \sigma(\sigma(x))$ Fact 3

$\sigma(-x) = \sigma(\sigma(-x))$ Fact 3

$\sigma(a_2x)$

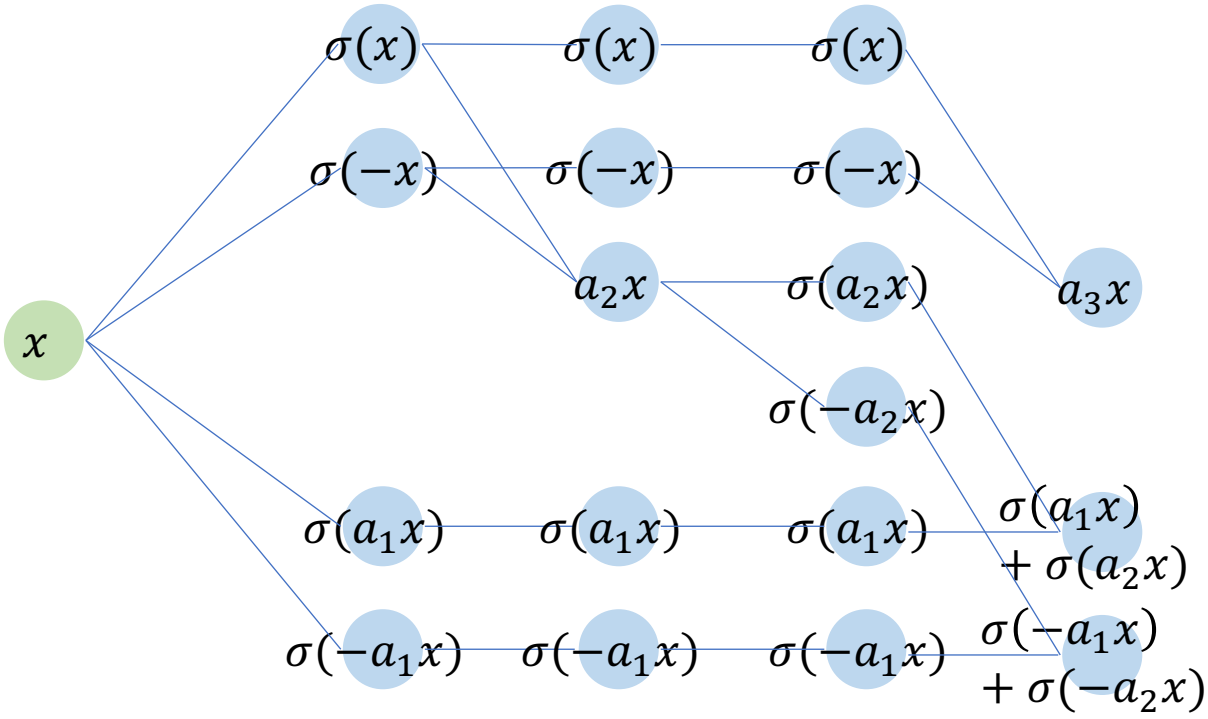
$\sigma(-a_2x)$

$\sigma(a_1x) = \sigma(\sigma(a_1x))$ Fact 3

$\sigma(-a_1x) = \sigma(\sigma(-a_1x))$ Fact 3

Rearranging 4

- Construct 4th layer
  - Represent $\sigma(a_1 x) + \sigma(a_2 x)$



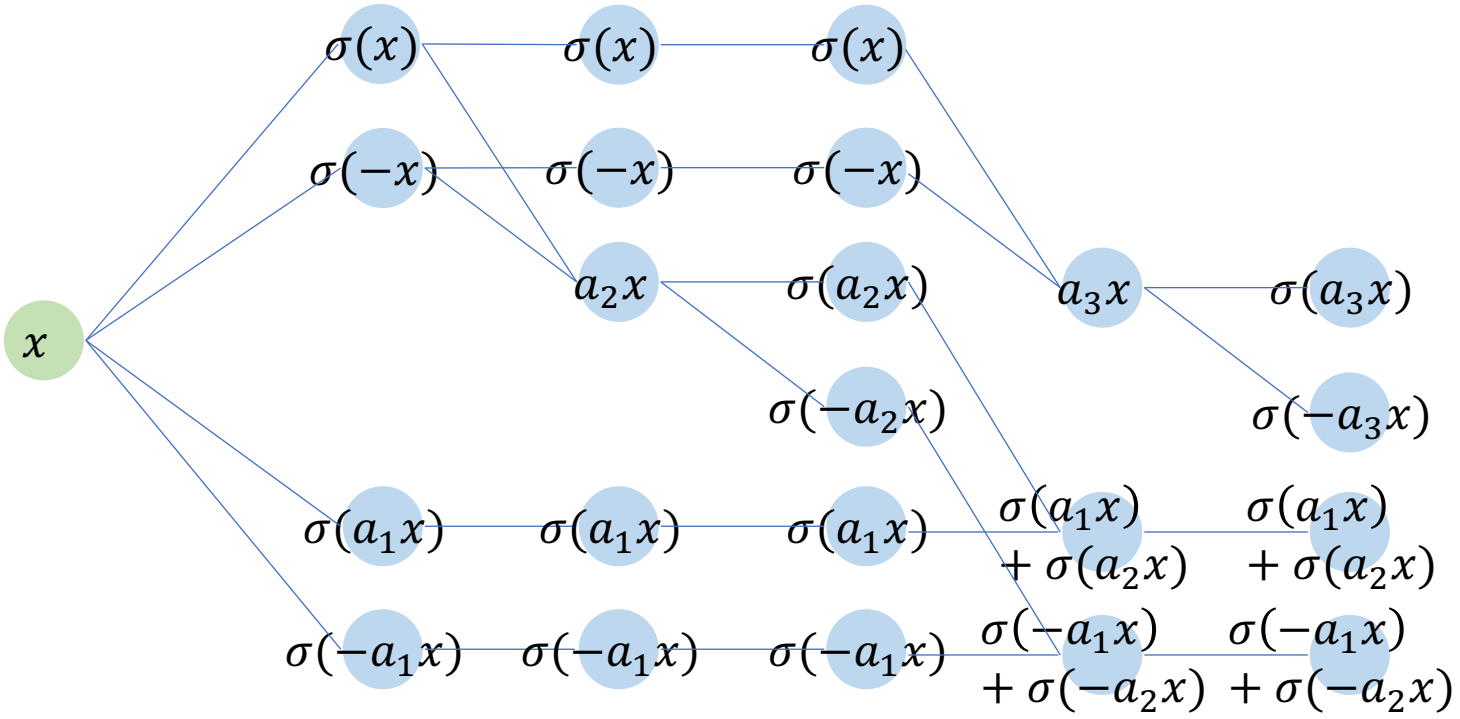
Fact 3

$$\sigma(a_1 x) + \sigma(a_2 x) = \sigma(\sigma(a_1 x) + \sigma(a_2 x))$$

$$\sigma(-a_1 x) + \sigma(-a_2 x) = \sigma(\sigma(-a_1 x) + \sigma(-a_2 x))$$

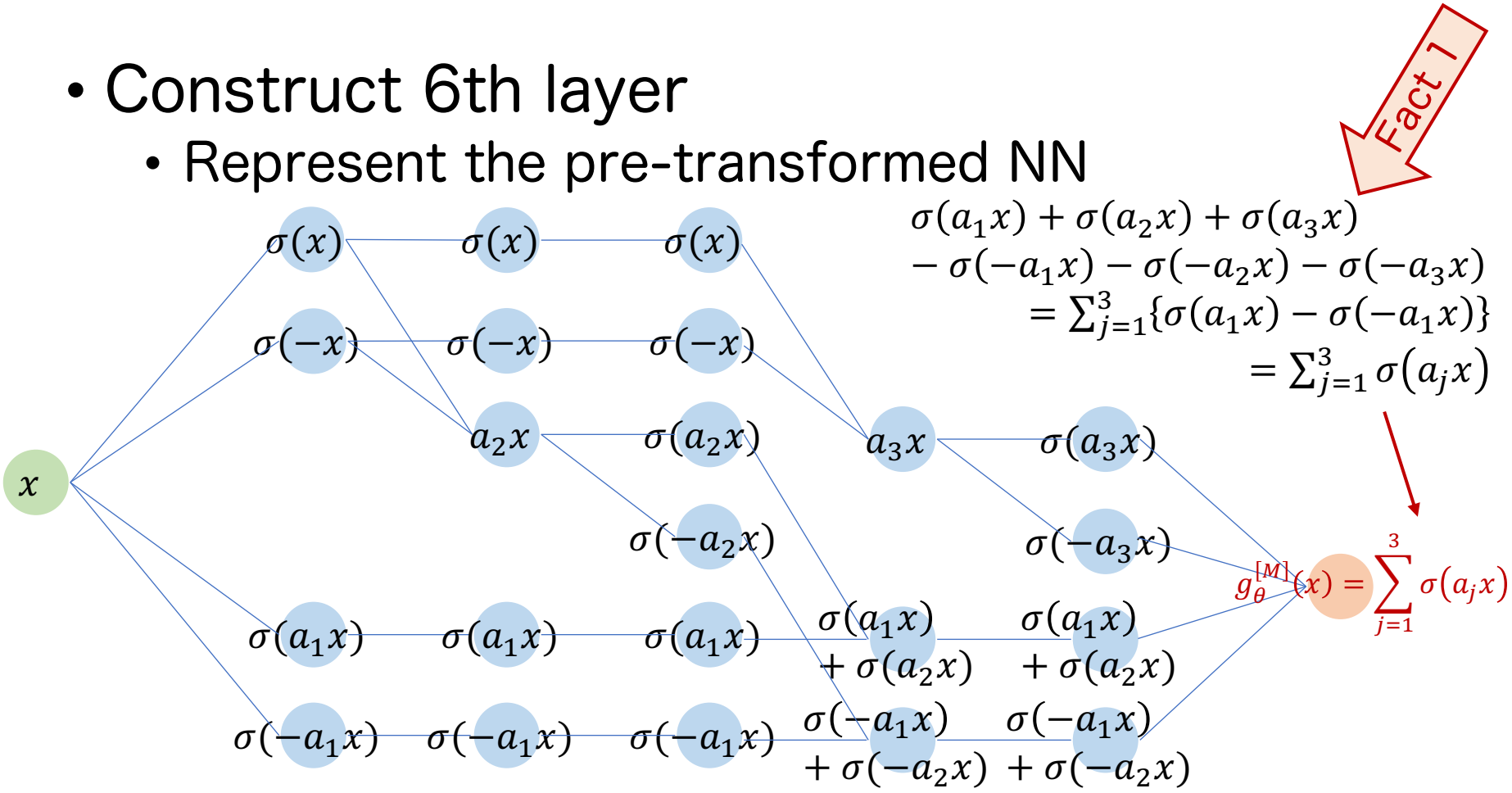
Rearranging 5

- Construct 5th layer
  - Represent $\sigma(a_3x), \sigma(-a_3x)$ for the third layer



Rearranging 6

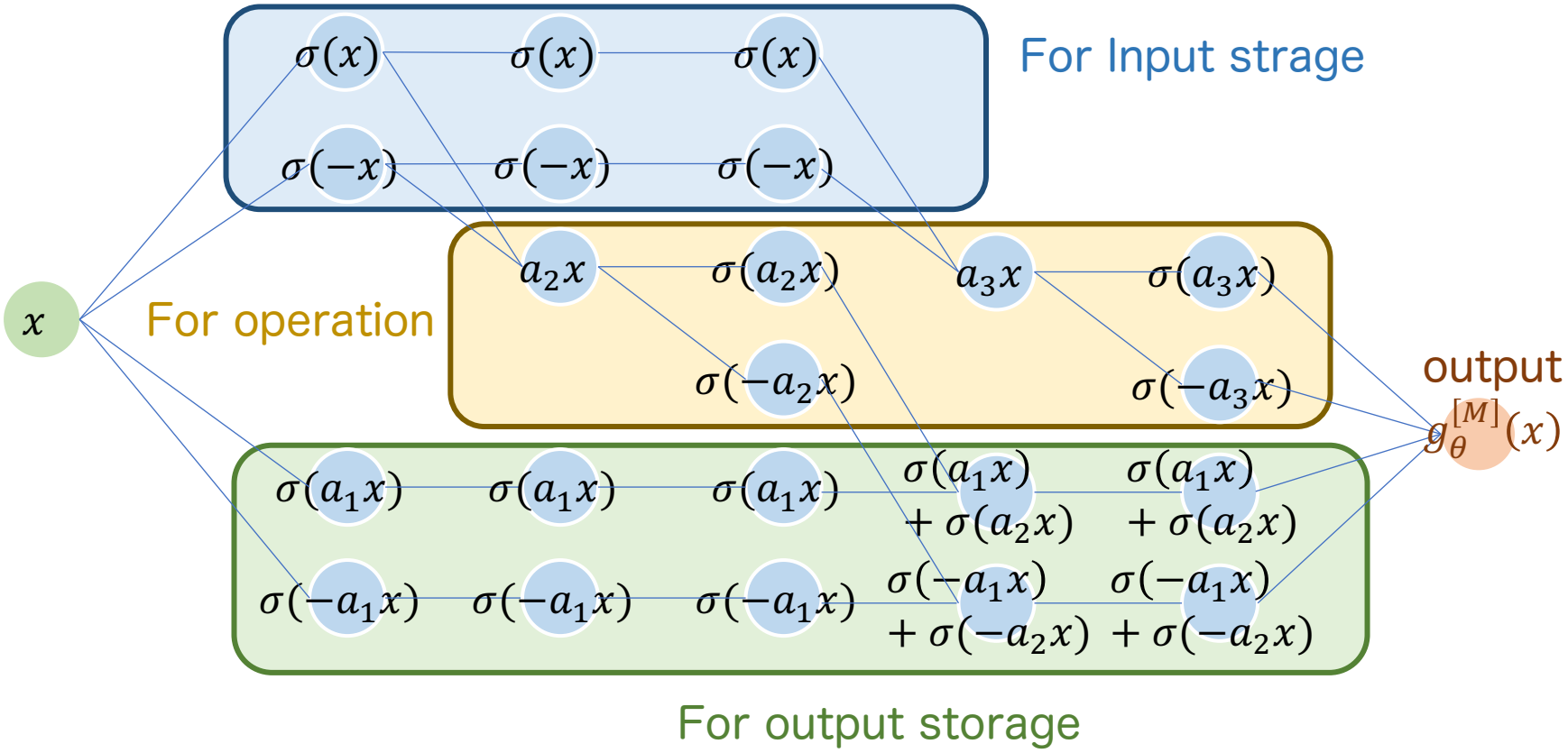
- Construct 6th layer
  - Represent the pre-transformed NN



It is the same function to $g_\theta^{(2)}(x)$.

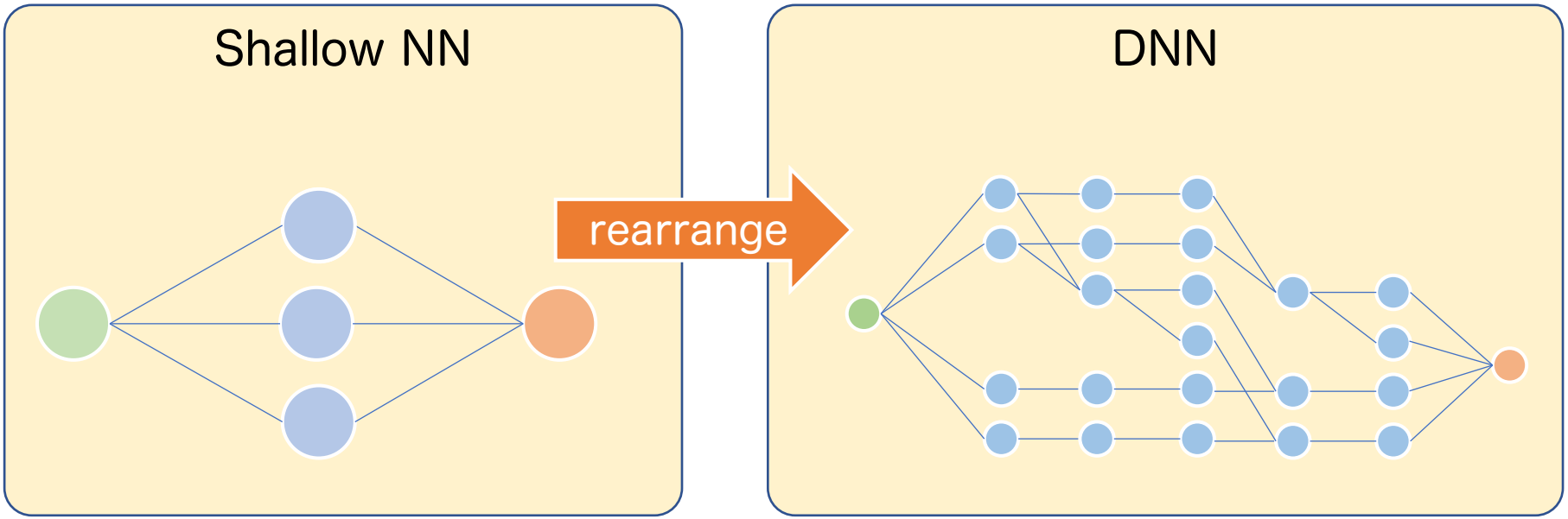
What DNN does after rearrangement?

- input storage/operations/output storage



In conclusion

- By the rearranging...



Hasn't the width increased too?
What makes you happy?

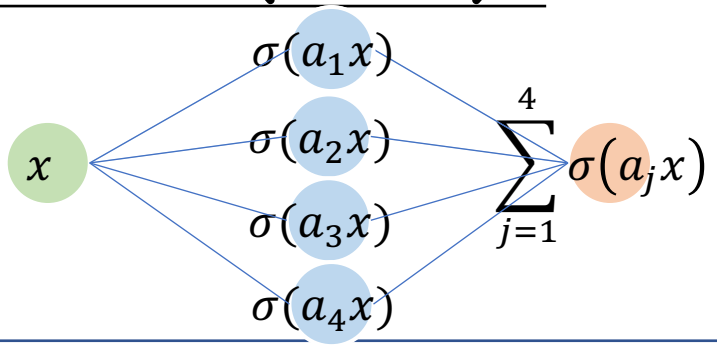
→ Consider the case of NN with larger width

Exercise 3

- What kind of DNN can you create if you rearrange the following settings?

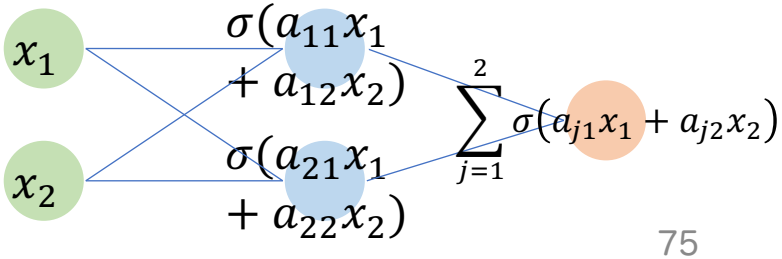
Q1 : NN with depth-2 width-4 $(d = 1)$

$$g_\theta(x) = \sum_{j=1}^{4} \sigma(a_j x)$$



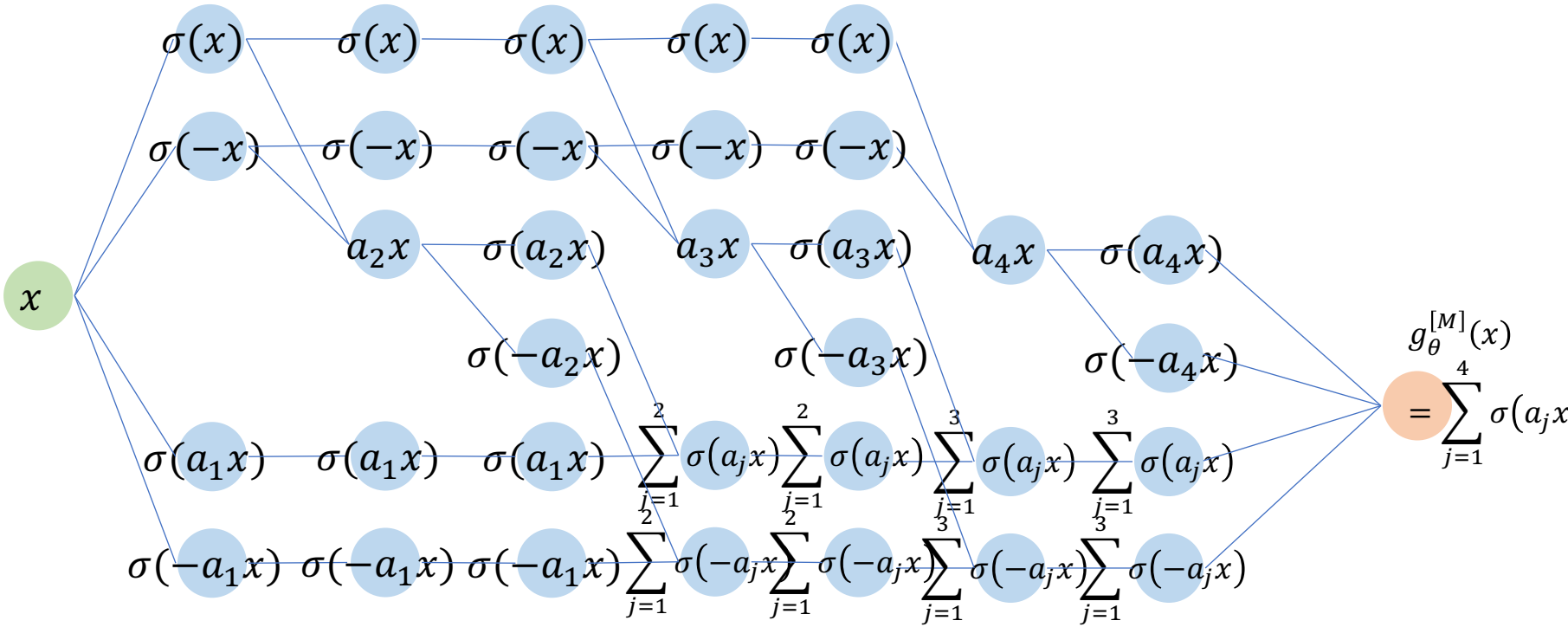
Q2 : NN with depth-2 width-2 $(d = 2)$

$$g_\theta(\boldsymbol{x}) = \sum_{j=1}^{2} \sigma(a_{j1}x_1 + a_{j2}x_2)$$



Solution 3: Q1

- DNN with depth-8 width-6
  - Simply add two more layers.



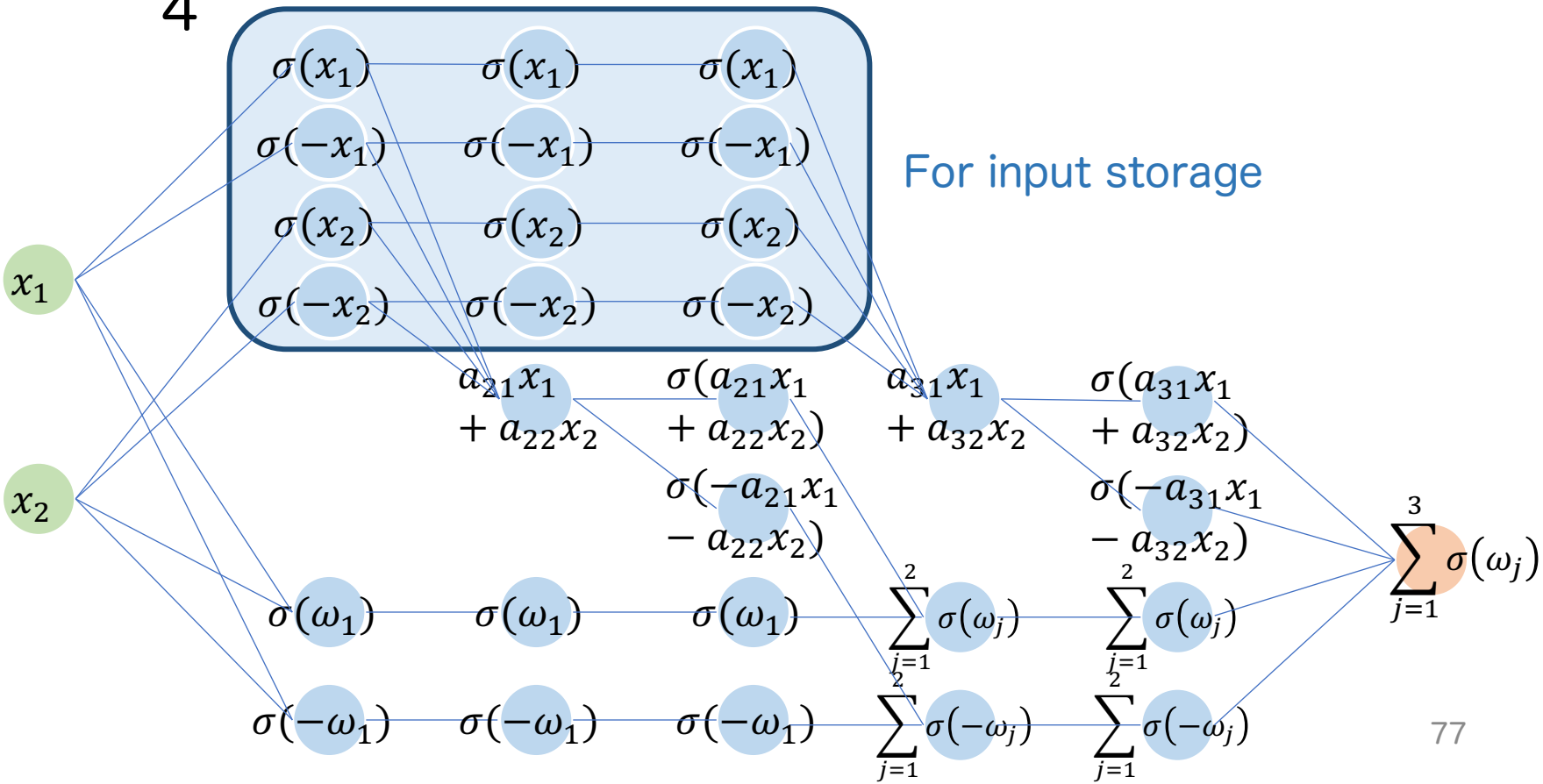
More layers, but no need to increase width

Solution 3: Q2

Notation

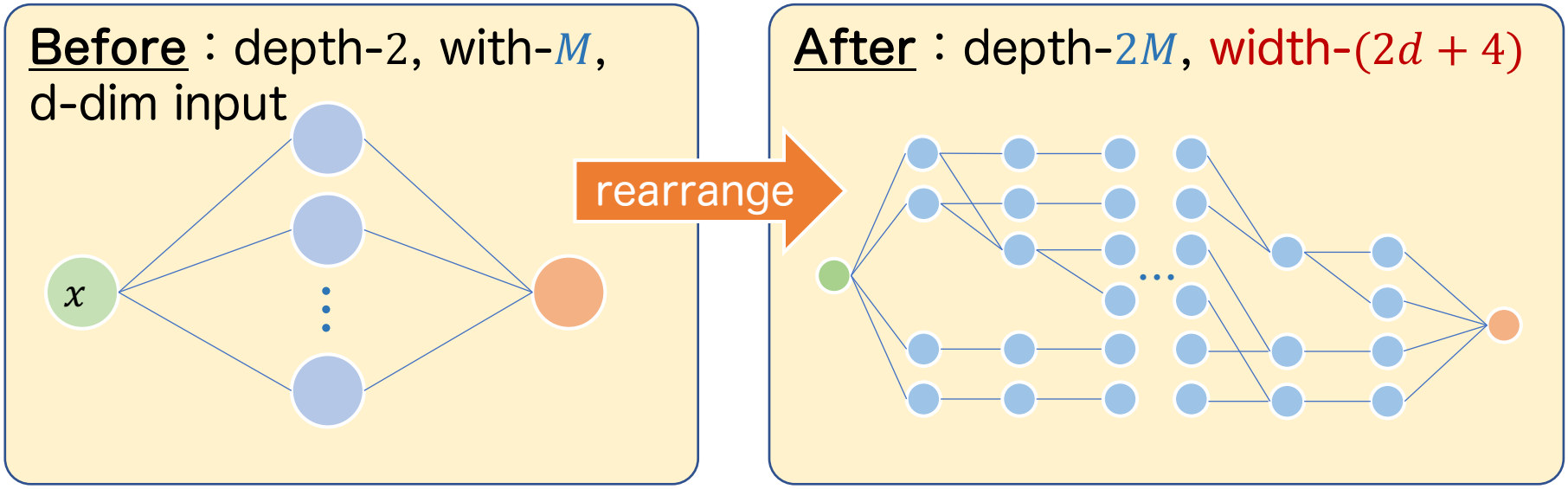
$\omega_j \coloneqq a_{j1}x_1 + a_{j2}x_2$

- DNN with depth-6 width-8
  - Increase the width of the input storage area to 4



Summary of Step 3

- This rearrangement does not increase width



Even if the width is large, $(M \to \infty)$
NN width after rearrange is always $2d$+4
→ Universal approx. even for finite width DNN!

Easy?

Pretty easy?

Rather difficult?

Summary

Summary

Topic

- Universal Approximation Theorem of DNNs
  - For various setups

Proof Outline

- DNN with less depth
  → approximation by piecewise constant
  + rectangle functions by ReLU
- DNN with more depth and less width
  → rearranging DNNs with less depth

Notes

<u>The actual UAT theorem is improved more</u>

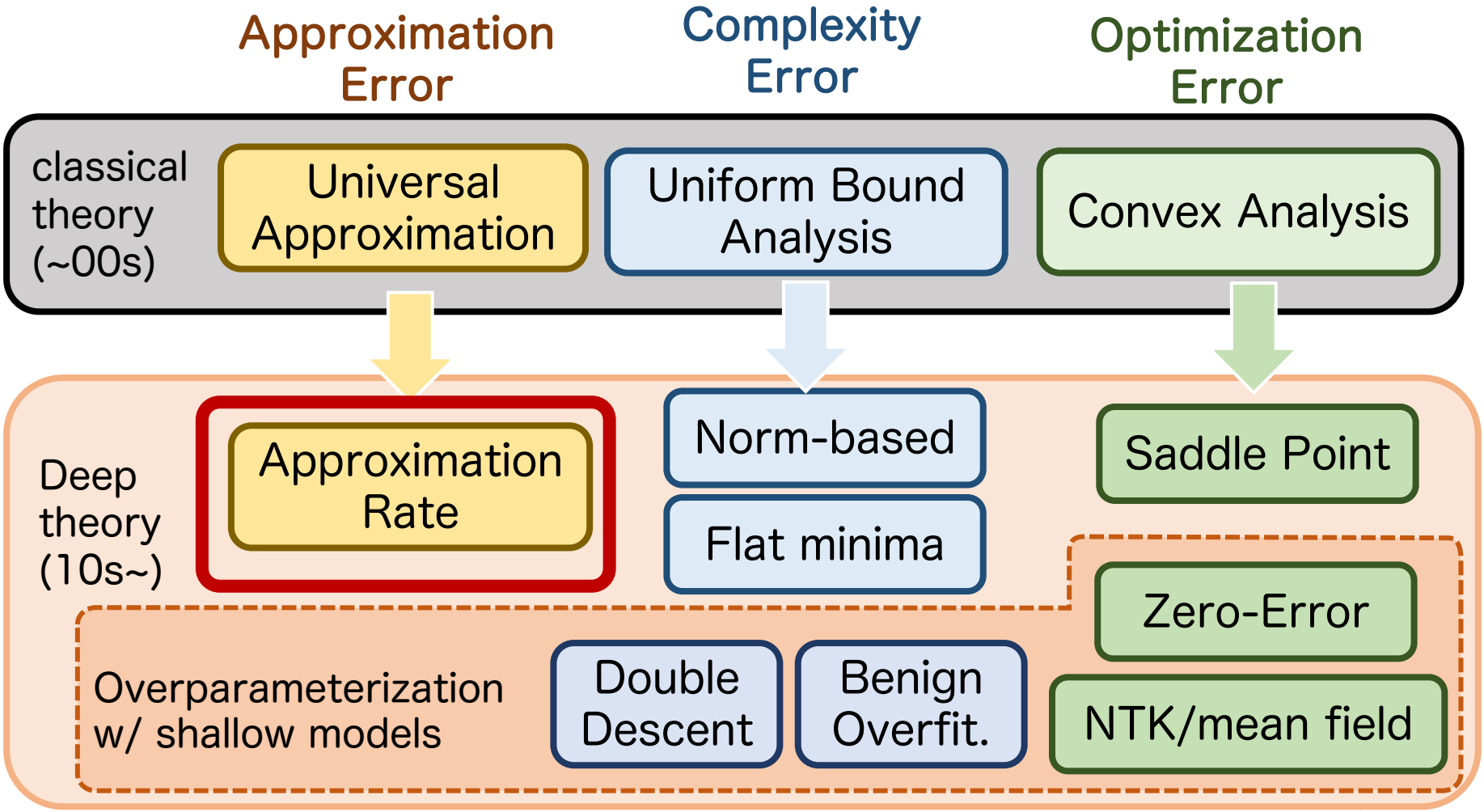
- More abstract and general proof [1,2]
- More general activation function [2]
- For d-dim. case, 2 layers are sufficient [1,2]
- The width DNNs can be reduced to $d + 2$ [3,4]

[1] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4), 303-314.
[2] Leshno, M., Lin, V. Y., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks.*
[3] Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017). The expressive power of neural networks: A view from the width. *NeurIPS*.
[4] Park, S., Yun, C., Lee, J., & Shin, J. (2021). Minimum width for universal approximation. *ICLR*.

Break

Approximation Rate Analysis

Theory map


Approximation Error
Complexity Error
Optimization Error
classical theory (~00s)
Universal Approximation
Uniform Bound Analysis
Convex Analysis
Deep theory (10s~)
Approximation Rate
Norm-based
Flat minima
Saddle Point
Zero-Error
Overparameterization w/ shallow models
Double Descent
Benign Overfit.
NTK/mean field

Rate Analysis

Landau's big o

$x_N = O(y_N), (N \to \infty)$ means

$\exists N', C > 0 \text{ s.t. } N > N' \Rightarrow |x_N| \leq C|y_N|$

Approximation Error Rate

- Decay speed of the error as parameter increases

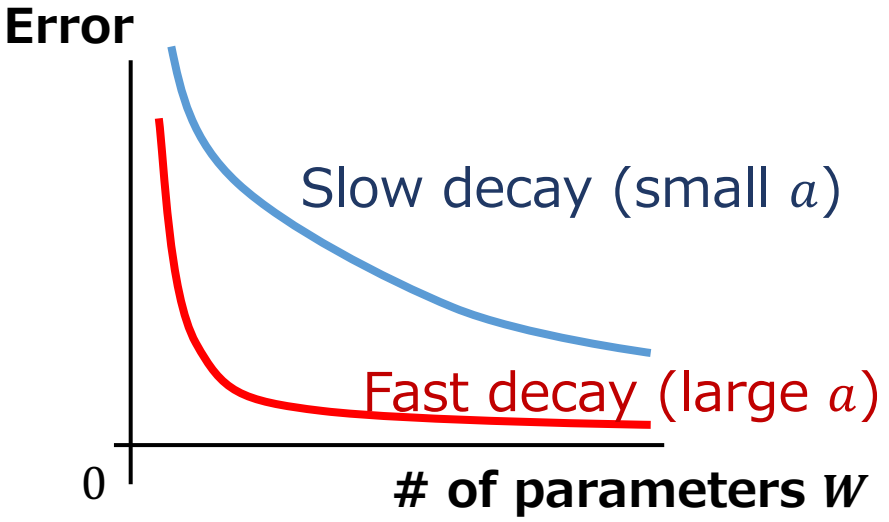
Error Rate a

f^*: Target of approximation

f_θ: DNN with W parameters

$$\inf_\theta \|f^* - f_\theta\| = O(W^{-a})$$

Approximation Error



Assumption: f^* is differentiable

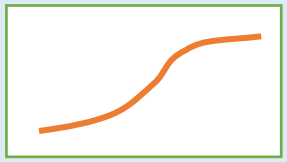
Approximation Rate

f^* : Target function (d-dim input, β-times differentiable)

f_θ : DNN (L-layer, W parameters, σ: activation)

σ is smooth (e.g. Mhaskar (1996))
DNN with $L = 2$ satisfies

$$\inf_\theta \|f^* - f_\theta\| = O(W^{-\beta/d})$$



σ is ReLU (e.g. Yarotsky (2017))
DNN with L layers satisfies

$$\inf_\theta \|f^* - f_\theta\| = O(W^{-\beta/d} + 2^{-L})$$

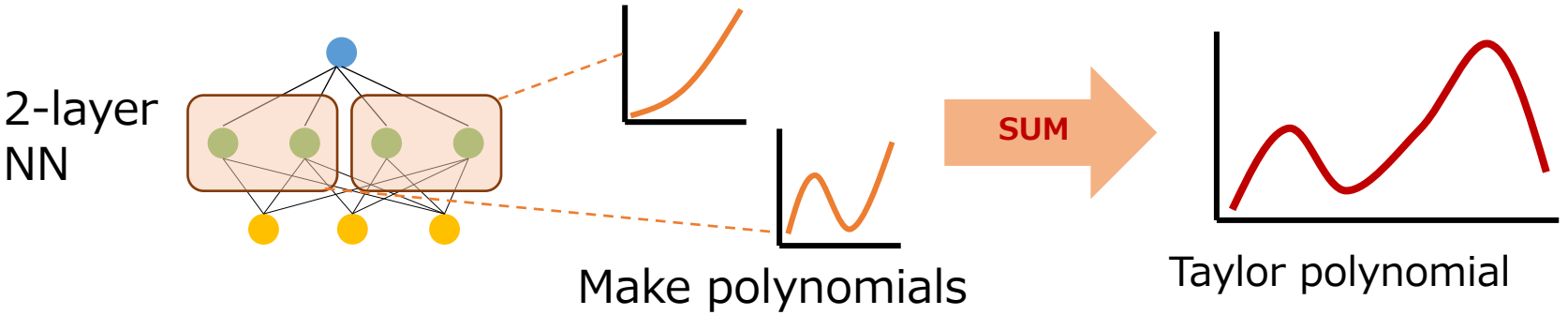
Effect from kink of ReLU

The rate β/d is described by smoothness and dimension.

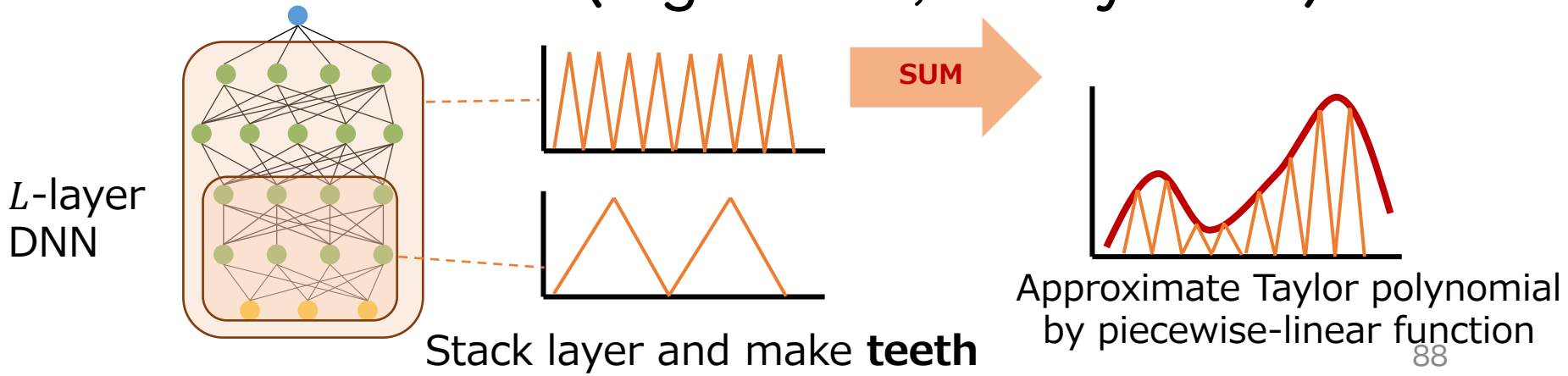
Mhaskar, H. N. (1996). Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*.
Yarotsky, D. (2017). Error bounds for approximations with deep ReLU networks. *Neural Networks*.

Difference by activation σ

- Smooth σ (e.g., sigmoid, softplus)



- Non-smooth σ (e.g. ReLU, LeakyReLU)



Rate by Depth and Width

- Derive the rate w/o W (# of parameters)

σ is ReLU case (e.g. Lu et al. (2021))
DNN with L layers and N width satisfies

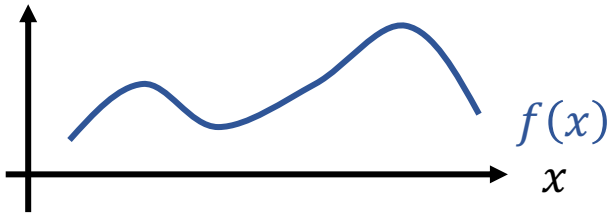
$$\inf_{\theta} \|f^* - f_\theta\| = O(N^{-2\beta/d} L^{-2\beta/d})$$

- Relation b/w W, L and N
  - $W \leq LN(N+1) = O(LN^2)$
- If L is constant, the rate is $O(W^{-\beta/d})$
  - Consistent with the previous result
- More layers, we can reduce parameters.

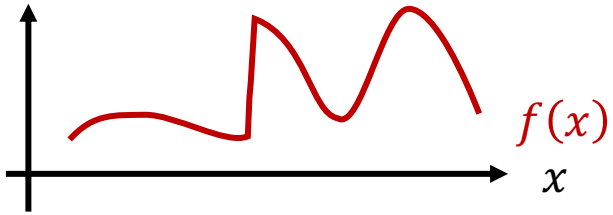
Lu, J., Shen, Z., Yang, H., & Zhang, S. (2021). Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, *53*(5), 5465-5506.

For Complicated Functions

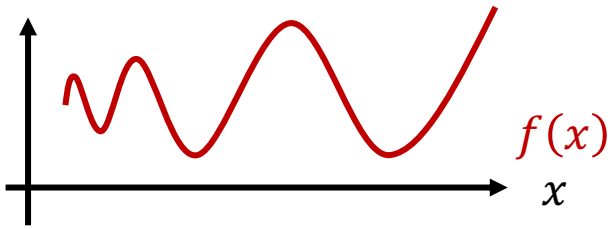
- As go beyond the simple smooth functions, more layers are effective.



Simple smooth function



Non-Smooth function



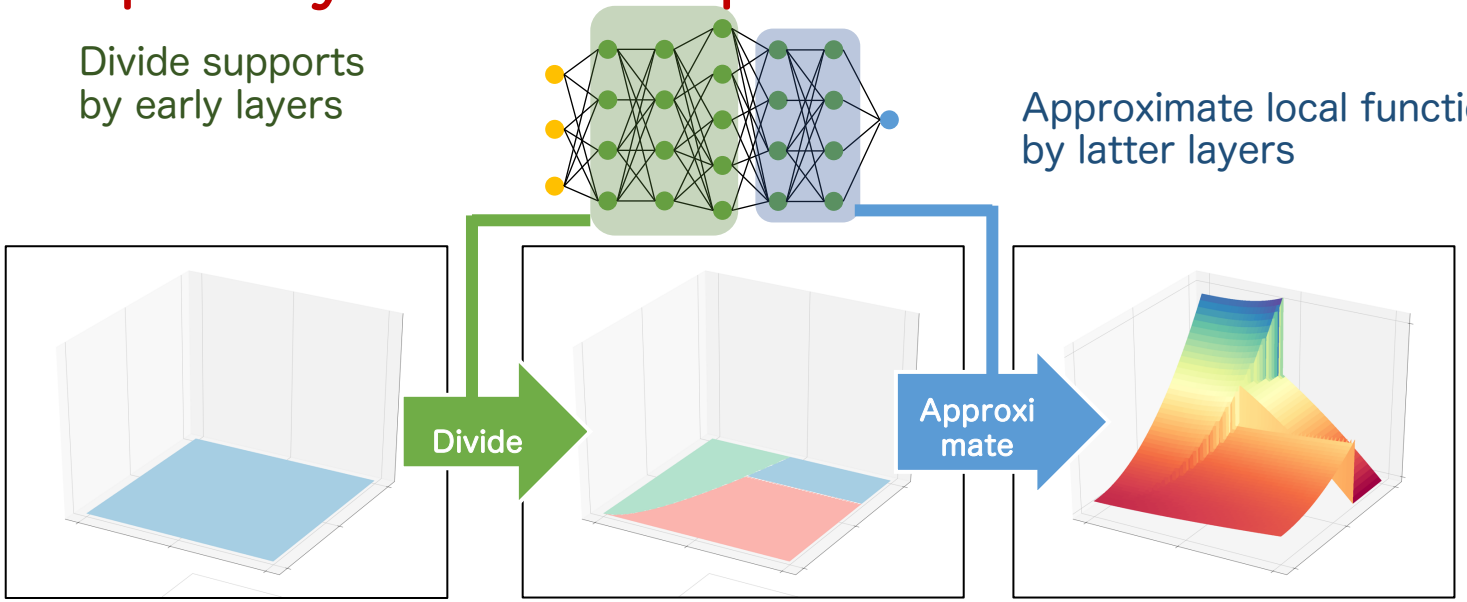
Heterogeneously smooth function

→ Non-Deep method is enough
(rate optimal)

→ DNN has an advantage

Non-Smooth Case

- Multiple layers are important



DNN has the faster rate

α: smoothness of boundary in the support

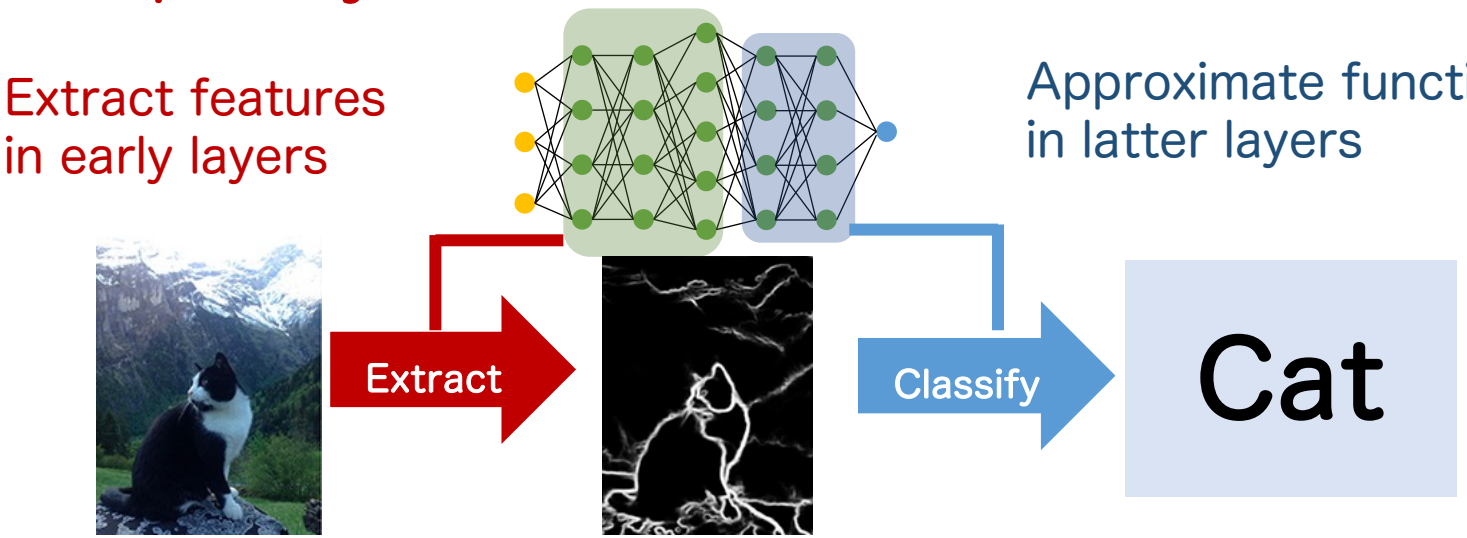
Rate by DNNs: $O(\max\{W^{-\beta/d}, W^{-\alpha/2(d-1)}\})$

Rate by non-deeps: $O(\max\{W^{-\beta/d}, W^{-\alpha/4(d-1)}\})$

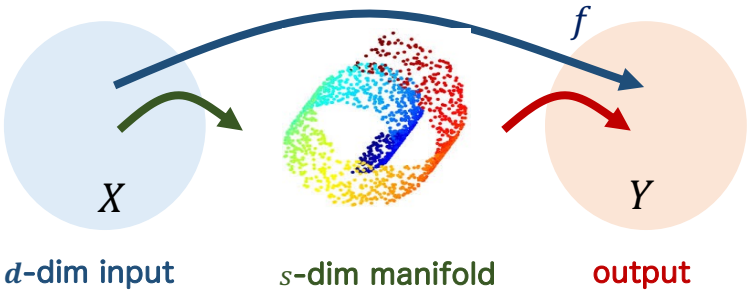
Petersen, P., & Voigtlaender, F. (2018). Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*.
Imaizumi, M., & Fukumizu, K. (2022). Advantage of Deep Neural Networks for Estimating Functions with Singularity on Hypersurfaces. *JMLR*.

Function with Features

- Multiple layers extract features



- Dim. of data-manifolds appear.



Faster Approx. Rate

Rate by DNNs	$\tilde{O}(W^{-\beta/s})$
Generate Rate	$O(W^{-\beta/d})$

Nakada, R., & Imaizumi, M. (2020). Adaptive Approximation and Generalization of Deep Neural Network with Intrinsic Dimensionality. *JMLR*.

Summary of Approximation Error

Nice result for neural networks

- Can approximate continuous functions.
- Can approximate smooth functions w/ optimality.

More results on **deep** neural networks

- More effective to approximate complicated functions
- Non-smooth activation (ReLU) is also useful.

Many nice result on the **deep** neural networks.
→Role of **depth** has been largely clarified.

Complexity Error

Decomposition

Decompose gen. error into three parts

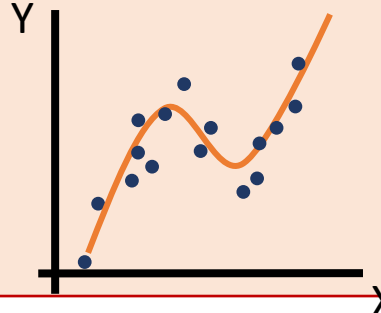
- DL contains several aspects...

$$R(\hat{\theta}) = \inf_{\theta} R_n(\theta) + R(\hat{\theta}) - R_n(\hat{\theta}) + R_n(\hat{\theta}) - \inf_{\theta} R_n(\theta)$$

Gen. Error | Approximation Error | Complexity Error | Optimization Error

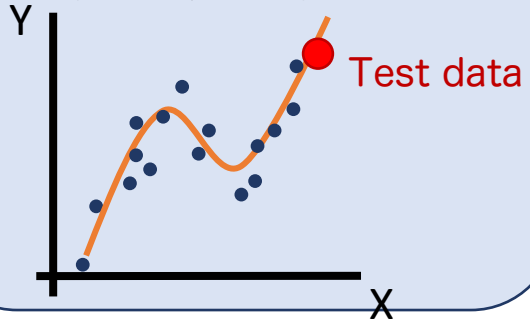
Approx. Error

How DNNs fit to data structure?



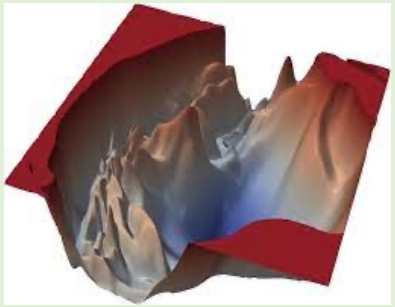
Complexity Error

How DNNs predict unknown data?



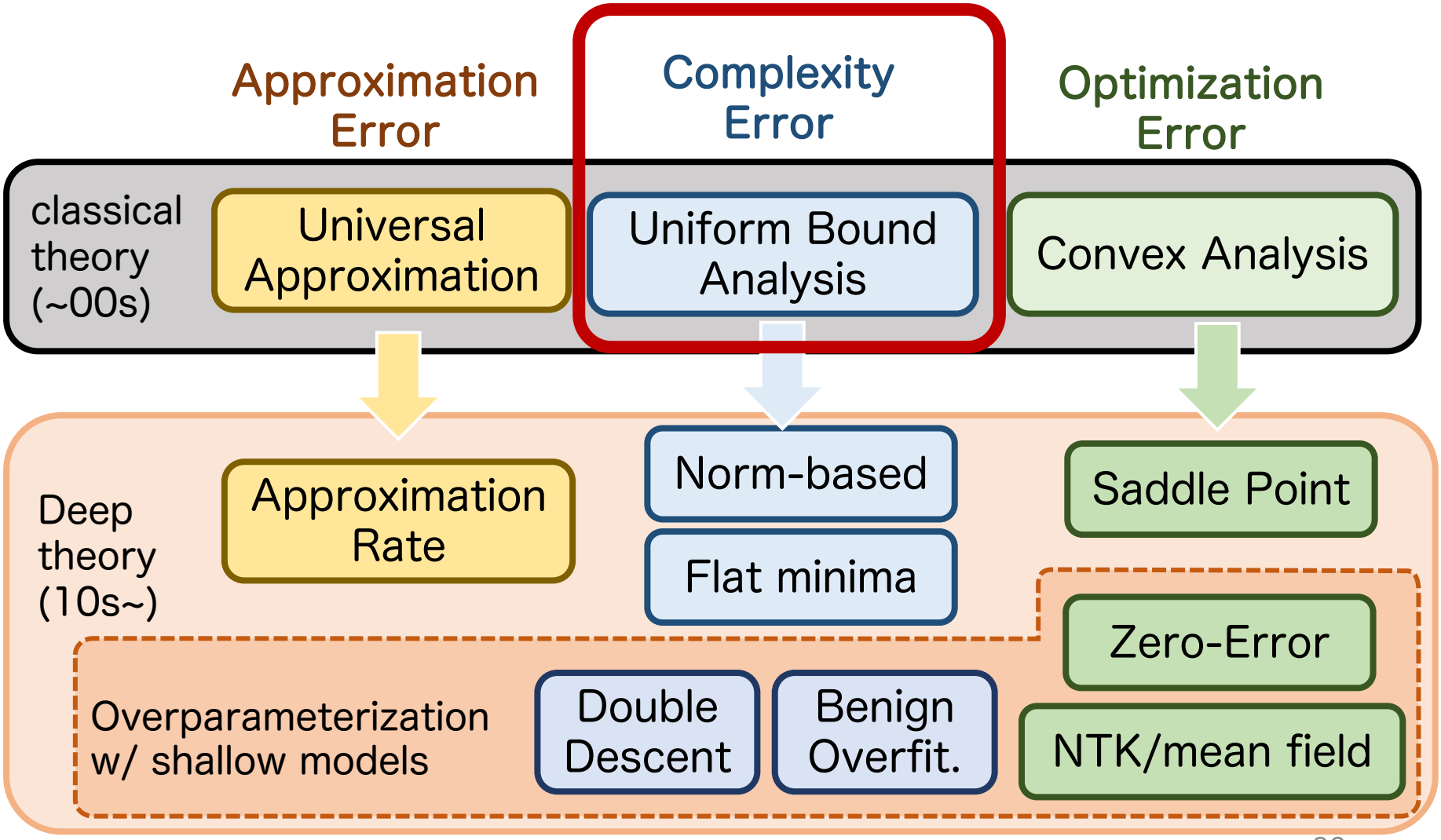
Optimization Error

How algorithm works?



Loss surface

Theory map


Approximation Error
Complexity Error
Optimization Error
classical theory (~00s)
Universal Approximation
Uniform Bound Analysis
Convex Analysis
Deep theory (10s~)
Approximation Rate
Norm-based
Flat minima
Saddle Point
Zero-Error
Overparameterization w/ shallow models
Double Descent
Benign Overfit.
NTK/mean field

Interest: Generalization gap

- Criteria for the complexity error

Empirical risk(= training loss)

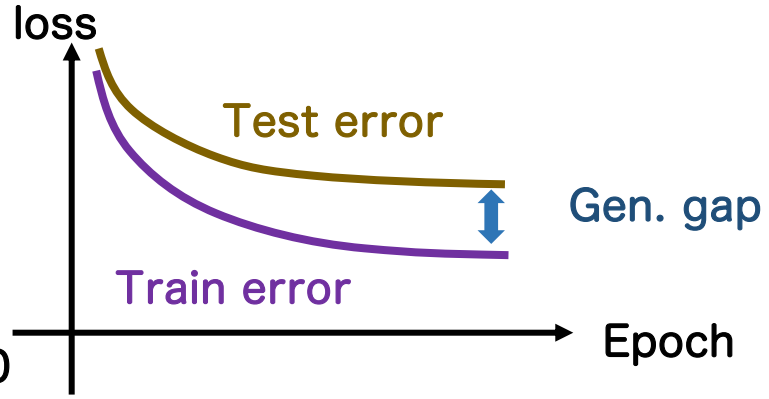
$$R_n(\theta) = n^{-1}\Sigma_{i=1}^{n}\ell\left(y_i, f_\theta(x_i)\right)$$

f_θ: Function by DNNs
$\{(x_i, y_i)\}_{i=1}^{n}$:n data
ℓ: loss

Obtain trained $\hat{\theta}$

Gen. error(≈ test error)

$$R\left(\hat{\theta}\right) = E\left[\ell\left(y, f_{\hat{\theta}}(x)\right)\right]$$



関心：Generalization gap (size of overfitting)

$$R\left(\hat{\theta}\right) - R_n\left(\hat{\theta}\right)$$

Classical Theory

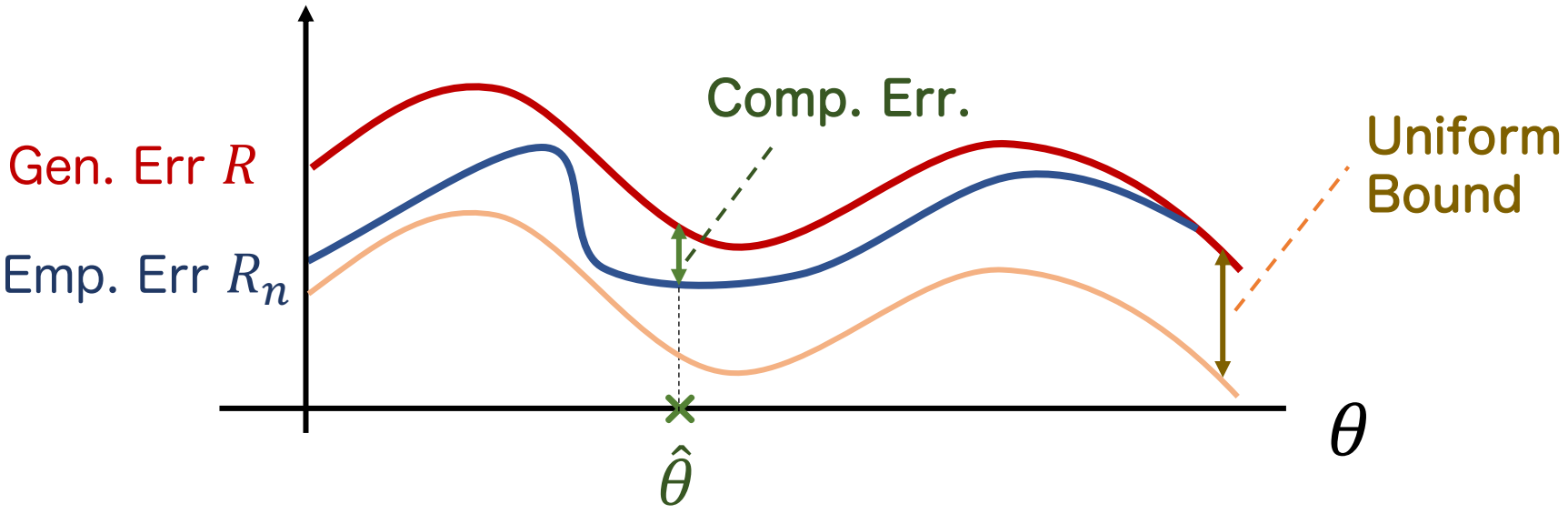
Uniform Bound Approach

Evaluate an uniform bound of the comp. err.

$$\left(R(\hat{\theta}) - R_n(\hat{\theta})\right) \leq \sup_{\theta}\left(R(\theta) - R_n(\theta)\right)$$

Complexity Error

Uniform Bound



Somewhat loose, but applicable to broad range of methods.

Result

n: # of data
W: # of parameters of DNNs
L: # of layers

Claim

- It is evaluated by model size and data size.

Theorem

$\forall \delta \in (0,1)$, with probability $\geq 1-\delta$:

$$\sup_{\theta}\left(R(\theta) - R_n(\theta)\right) \leq C\sqrt{\frac{WL\log(WL)}{n}} + C\sqrt{\frac{\log(1/\delta)}{n}}$$

$C > 0$ some constant

Key: First term in the bound
Increases in the size of DNN (WL), decrease in n.

Analysis for Uniform Bound

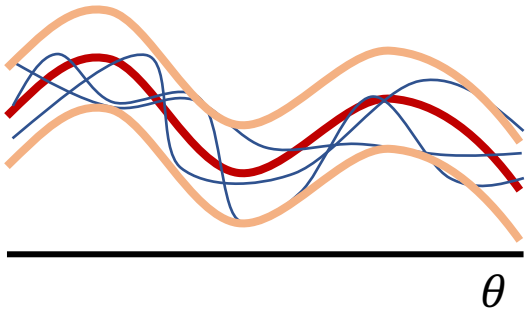
- Uniform bound is described by the size of parameter space

C: constant

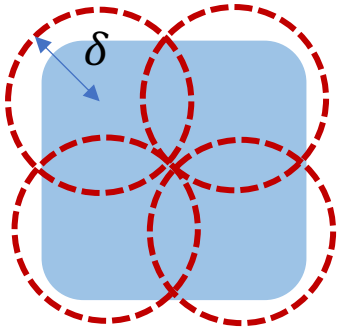
Uniform Bound

Dudley Integral

$$\sup_{\theta \in \Theta} R(\theta) - R_n(\theta) \leq \frac{C}{\sqrt{n}} \int_0^\infty \sqrt{\log N_\delta(\{f_\theta\})}\, d\delta$$



Rademacher complexity



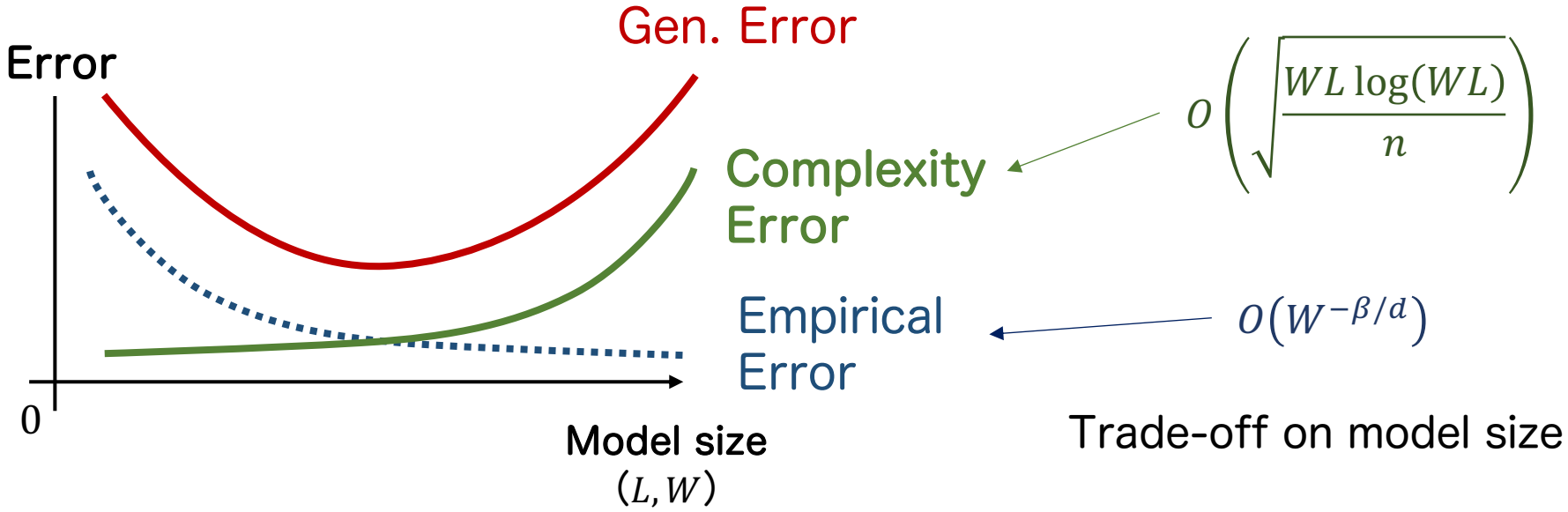
$\{f_\theta\}$

$N_\delta(\{f_\theta\})$: covering number
(# of balls to cover $\{f_\theta\}$)

DNN has many parameters
$\rightarrow \{f_\theta\}$ is a large set
$\rightarrow \log N_\delta(\{f_\theta\})$ is large value ($\propto LW$)

What the theory says

- Large models make the bound large.

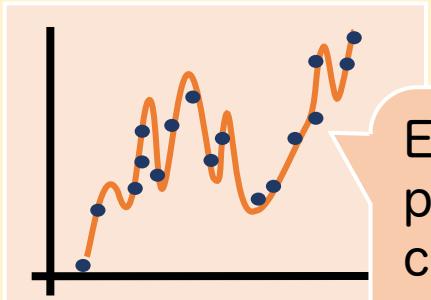


This theory cannot explain the success of DL.

Puzzle by DL

Contradiction b/w DL and (classical) theory

Classical Theory

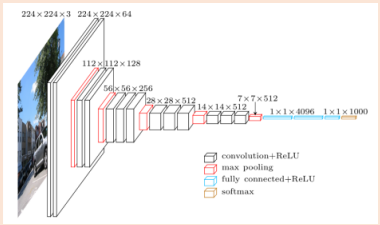


Excess parameters cause overfitting.

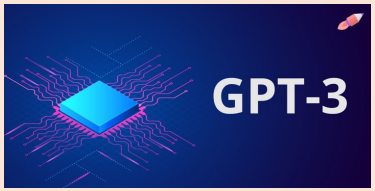
$$\text{Error} \propto \sqrt{\frac{\#\text{ of parameters}}{\#\text{ of data}}}$$

GAP

Success of DL



VGG19 Net
100 million~ parameters



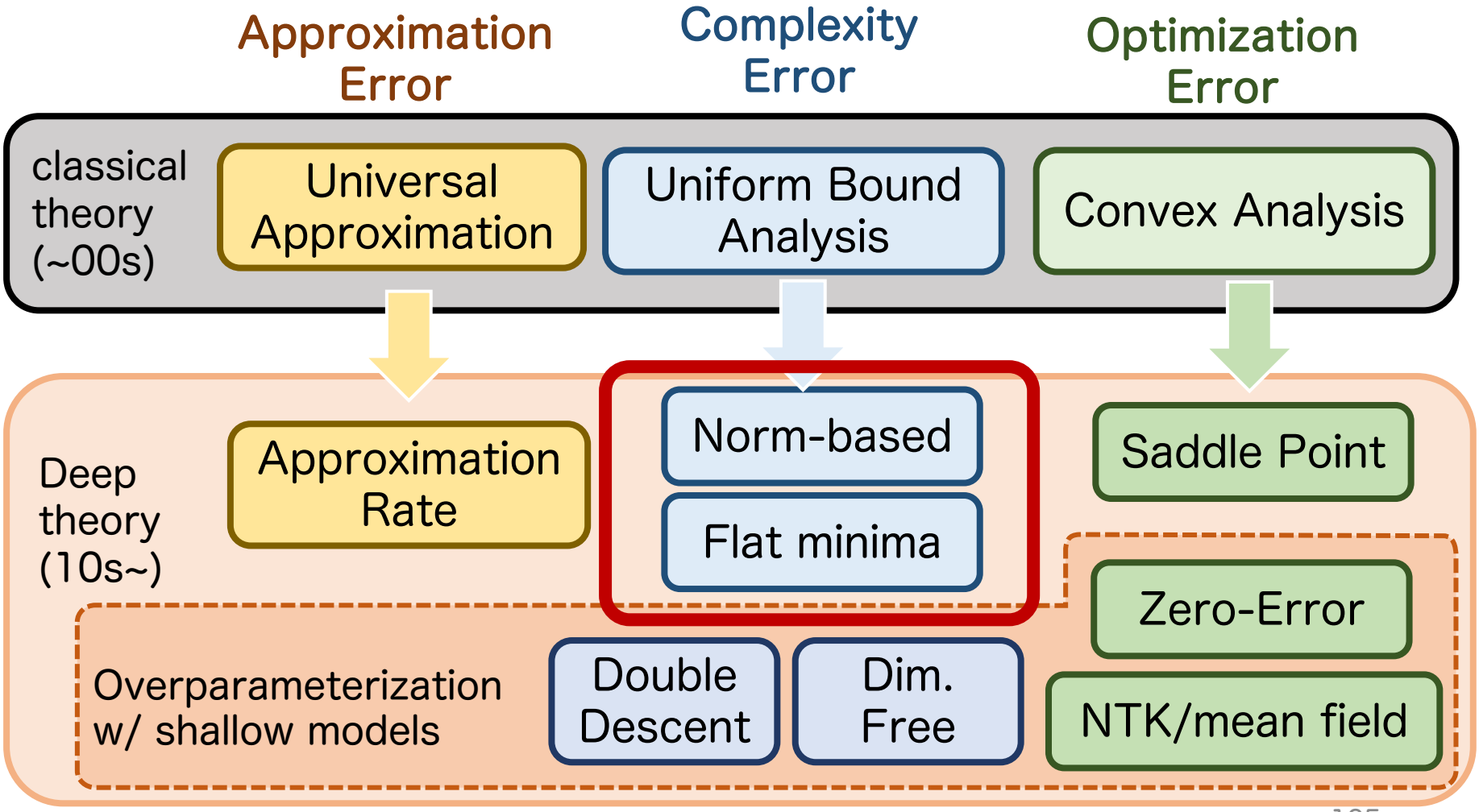
GPT-3
100 billion~ parameters

More parameter, higher accuracy

→ **Rethinking Theory**: Develop theory for DL

Recent Theory

Theory map

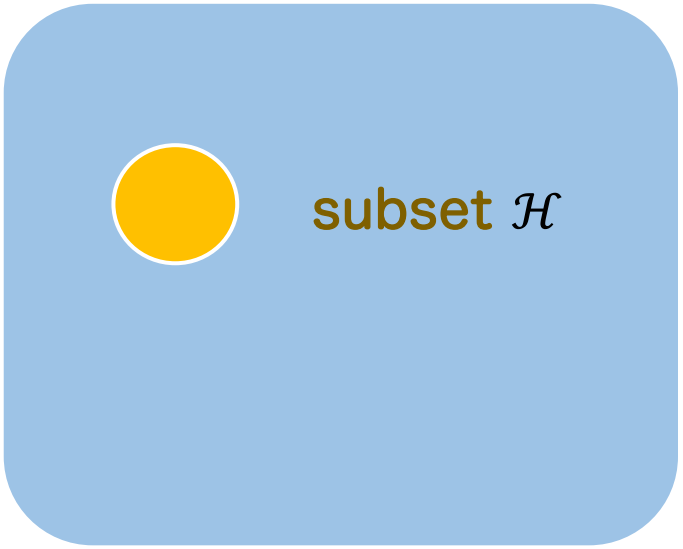

Approximation Error
Complexity Error
Optimization Error
classical theory (~00s)
Universal Approximation
Uniform Bound Analysis
Convex Analysis
Deep theory (10s~)
Approximation Rate
Norm-based
Flat minima
Saddle Point
Zero-Error
NTK/mean field
Overparameterization w/ shallow models
Double Descent
Dim. Free

Consider Small Model Class

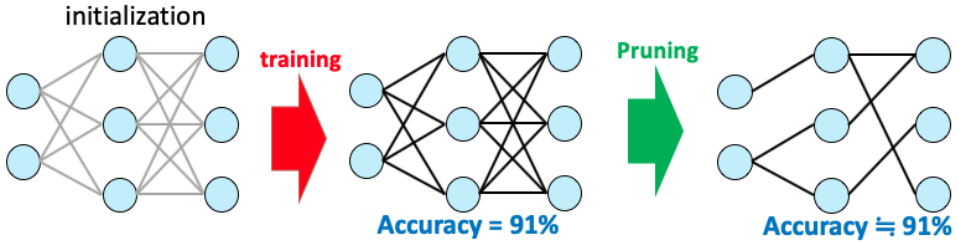
$\mathcal{F} \coloneqq \{f_\theta\}$ Set of functions by a DNN

Idea：Not necessary to consider the entire $\mathcal{F}$

- There may be an essential subset $\mathcal{H} \subset \mathcal{F}$



Set of DNN functions $\mathcal{F}$
(very large set)



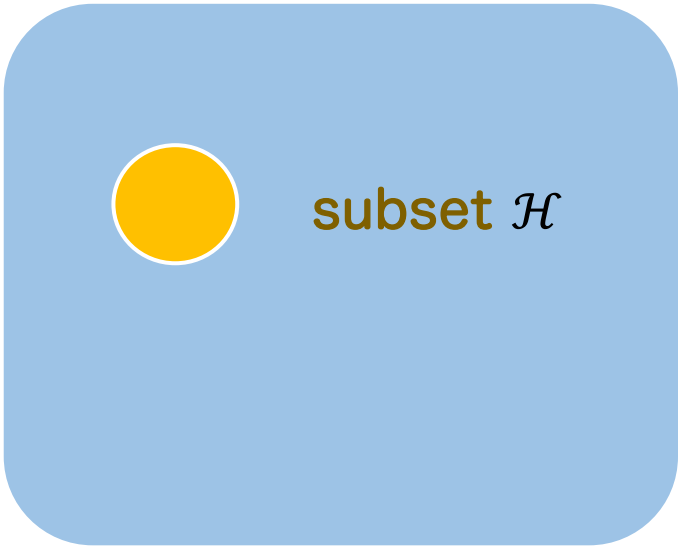
Related fact: pruning
Only a part of edges are necessary for inference.

Consider Small Model Class

$\mathcal{F} \coloneqq \{f_\theta\}$ Set of functions by a DNN

Idea：Not necessary to consider the entire $\mathcal{F}$

- There may be an essential subset $\mathcal{H} \subset \mathcal{F}$



Set of DNN functions $\mathcal{F}$
(very large set)

Classical Theory

Comp. Err. ≦ Size of the entire

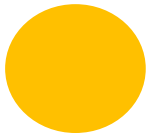
$$\sup_{\theta\in\Theta} R(\theta) - R_n(\theta) \leq \frac{C}{\sqrt{n}} \int_0^\infty \sqrt{\log N_\delta(\mathcal{F})}\, d\delta$$

New Idea

Comp. Err. ≦ Size of the subset

$$\sup_{\theta\in\Theta} R(\theta) - R_n(\theta) \leq \frac{C}{\sqrt{n}} \int_0^\infty \sqrt{\log N_\delta(\mathcal{H})}\, d\delta$$

Consider Small Model Class

Idea: Comp. Err. is described by the size of 

- Empirical Support (Zhang+ (2021))
  - Such the small subset can explain the generalization of DNNs.

What determines 

Challenges

1. Norm-based bound
2. Effect of Algorithm

etc...

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, *64*(3), 107-115.

Idea: Implicit regularization

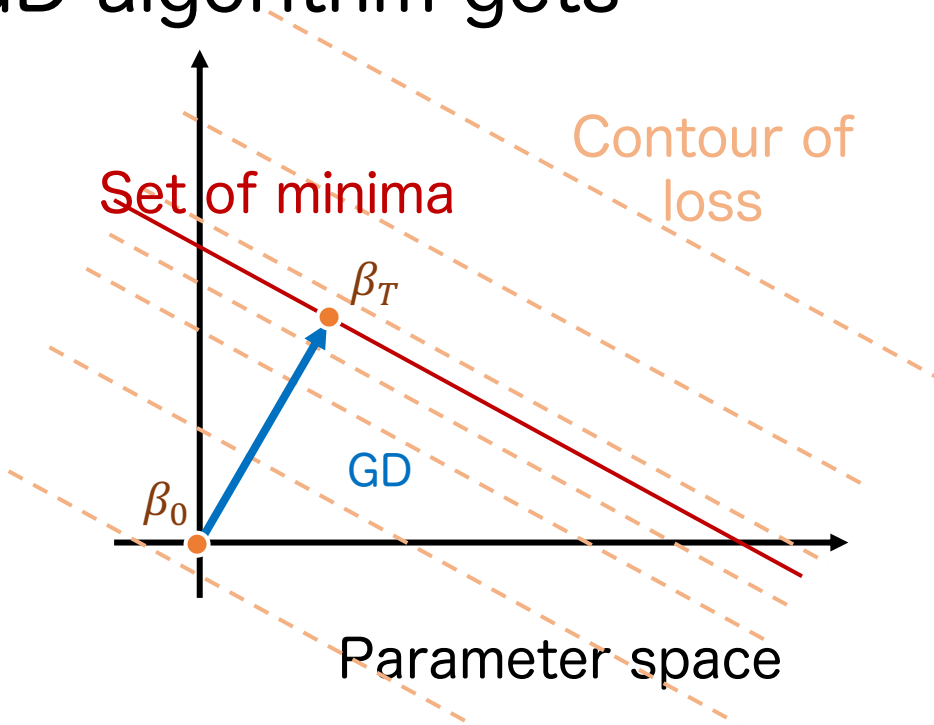
- In linear regression, GD algorithm gets close to the origin.

Linear regression

$$Y = (y_1, \dots, y_n)^\top, \in \mathbb{R}^n$$
$$X = (x_1, \dots, x_n)^\top \in \mathbb{R}^{n \times p}$$
$$\beta \in \mathbb{R}^p, \varepsilon \in \mathbb{R}^n$$
$$Y = X\beta + \varepsilon$$

Gradient Descent (GD) $(\beta_0 = 0, \eta > 0)$

$$\beta_{t+1} = \beta_t + \eta X^\top (Y - X\beta_t)$$



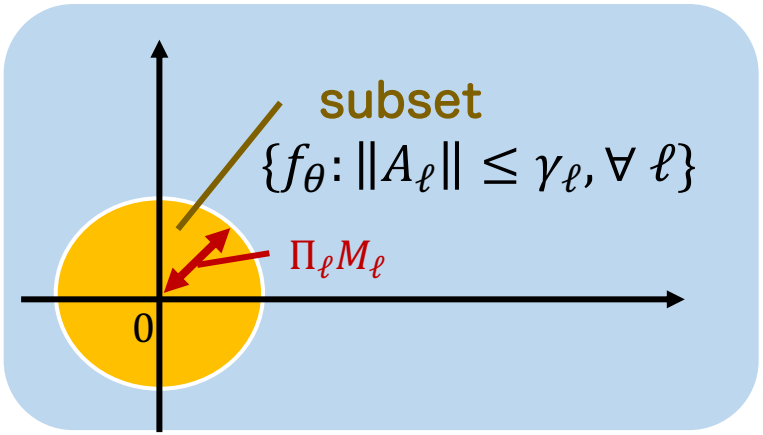
The norm of parameters are implicitly regularized.
The same thing would be happening at DNN.

At ℓ-th layer

Subset of Models

$$f_\ell(x) := \sigma(A_\ell x + b_\ell)$$

Origin neighbor (norm bound) ($\|A_\ell\|$ is bounded)

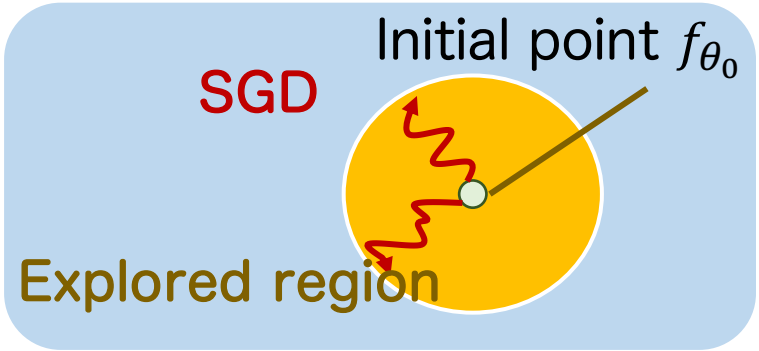


Complexity Error

$$O\left(\frac{B\sqrt{L}\prod_{\ell=1}^{L}\gamma_\ell}{\sqrt{n}}\right)$$

$B = \max_i \|x_i\|$: data scale

Initial neighbor



Complexity Error

Learning rate $\eta_t \asymp 1/t$

$$O\left(\frac{T^q}{n}\right)$$

$T \ge 1$: *# of upates,* $q \in (0,1)$:

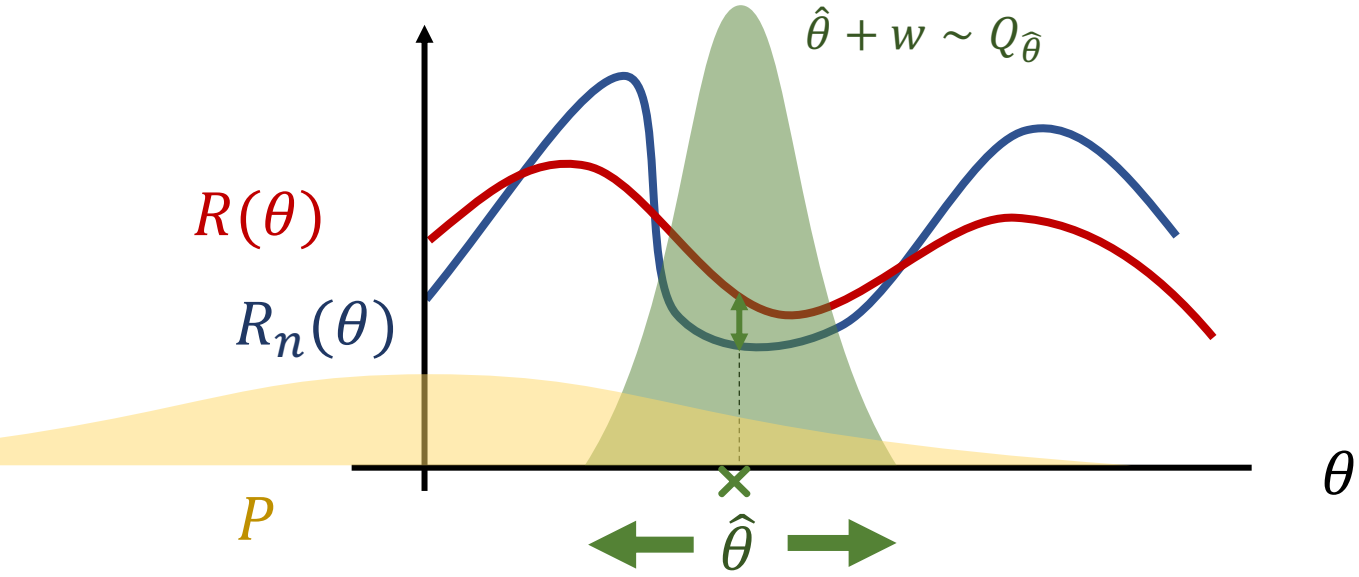
No (explicit) dependence on W (# of parameters)

Bartlett, P. L., Foster, D. J., & Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. *NIPS*.
Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural

Another: PAC-Bayes

Put perturbations on the parameter $\hat{\theta}$

→ Study it by a reference measure P



$w \sim Q$
w: perturbation
Q: distribution of w
Q_θ: distribution of θ+w

Study the complexity error by Q_θ and P.

How it works

Kullback-Leibler divergence

$$KL(Q||P) = \int \log\left(\frac{q(\theta)}{p(\theta)}\right) q(\theta)\, d\theta$$

p, q: density of distributions P, Q

Expectation of comp. err. by the perturbation

→ Explore the neighbor of θ by w

$$\left|E_{w\sim Q}\left[R(\hat{\theta}+w) - R_n(\hat{\theta}+w)\right]\right| = O_P\left(\sqrt{\frac{KL(Q_{\hat{\theta}}||P)}{n}}\right)$$

Expectation of perturbed error.

PAC-Bayes Bound by Q and P

Similar result to the norm-based bound

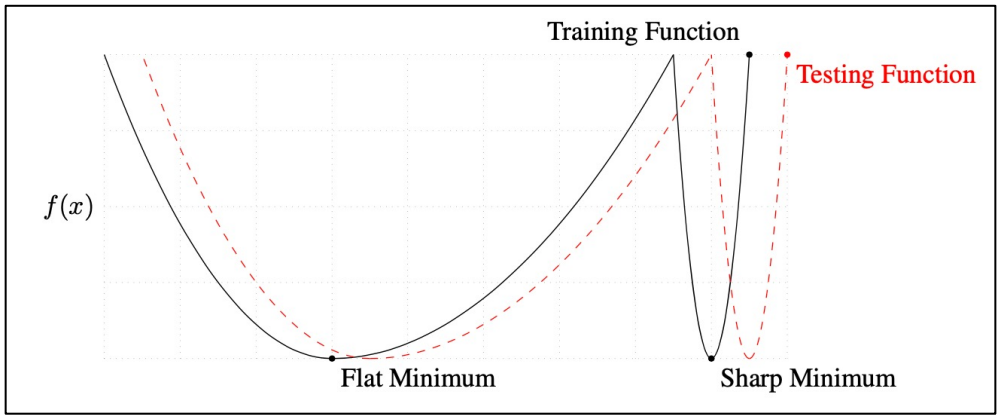
→ No (explicit) dependence on the NN size.

Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). Exploring generalization in deep learning. *arXiv preprint arXiv:1706.08947*.

Relation to Flat Minima

Flat Minima

- Empirically, a good parameter lies in a flat minima.



Flat Minima
(Keskar+ 2017)

PAC-Bayes theory explains the flat minima

$$\mathrm{E}_{\mathrm{w}\sim Q}\left[R(\hat{\theta}+w)\right]-R_n(\hat{\theta}) \leq \left\{E_{w\sim Q}\left[R_n(\hat{\theta}+w)\right]-R_n(\hat{\theta})\right\}+O_P\left(\sqrt{KL(Q_{\hat{\theta}}||P)/n}\right)$$

Approximated complexity error

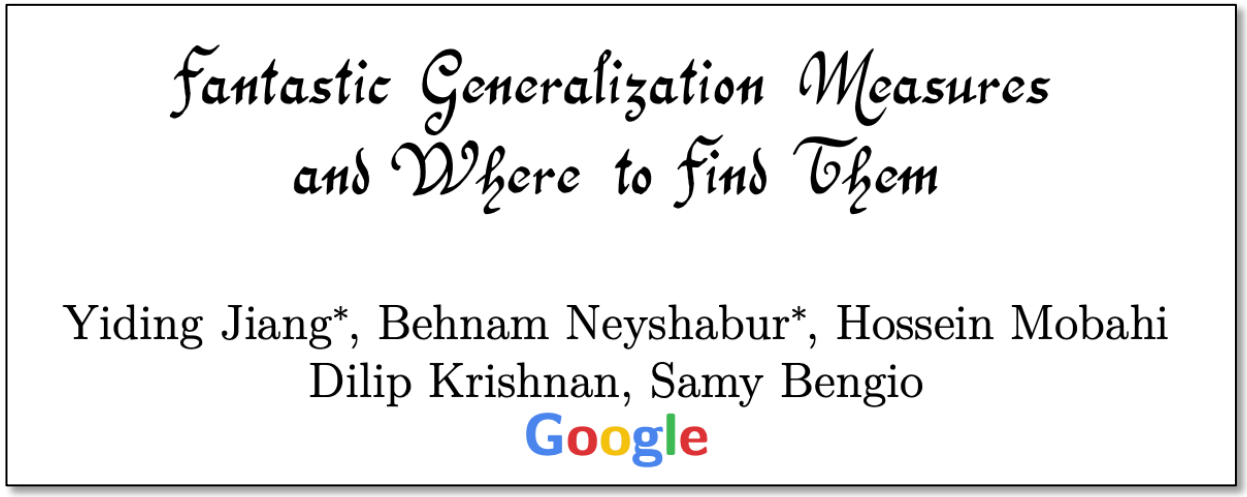
Expectation of non-flatness

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Experimental Validation

Large-scale experiment
Validate 40 theories by 2000 CNNs in experiments.

- **Result:**
  - The PAC-Bayes bound is the most promising, rather than other theories.

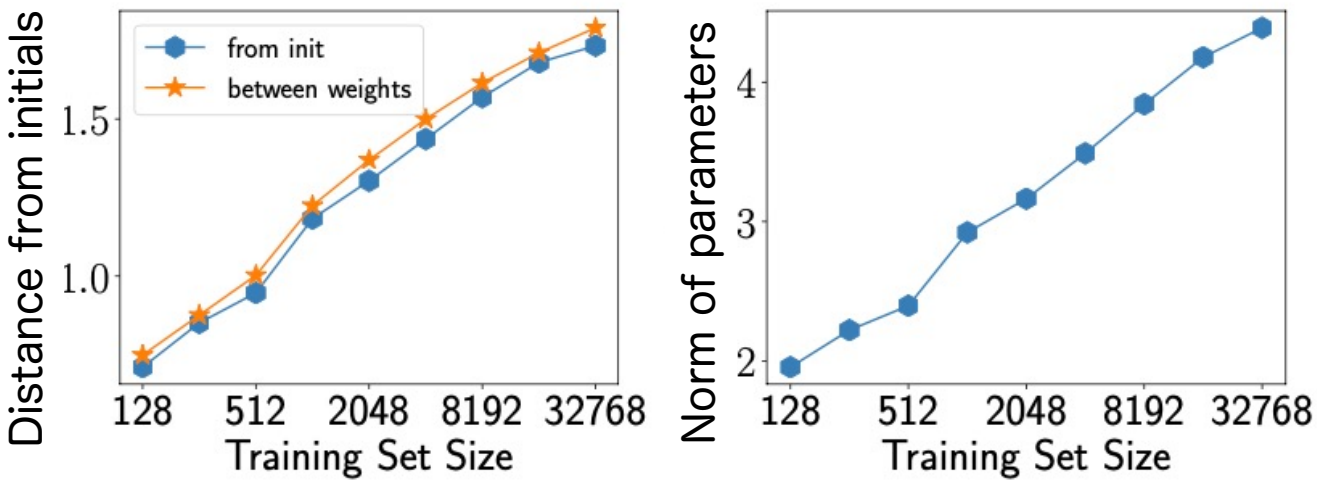


(Jiang+ 2018)

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., & Bengio, S. (2019). Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*.

Some Criticism

Experimental Observation:
Norm of parameters are not bounded.



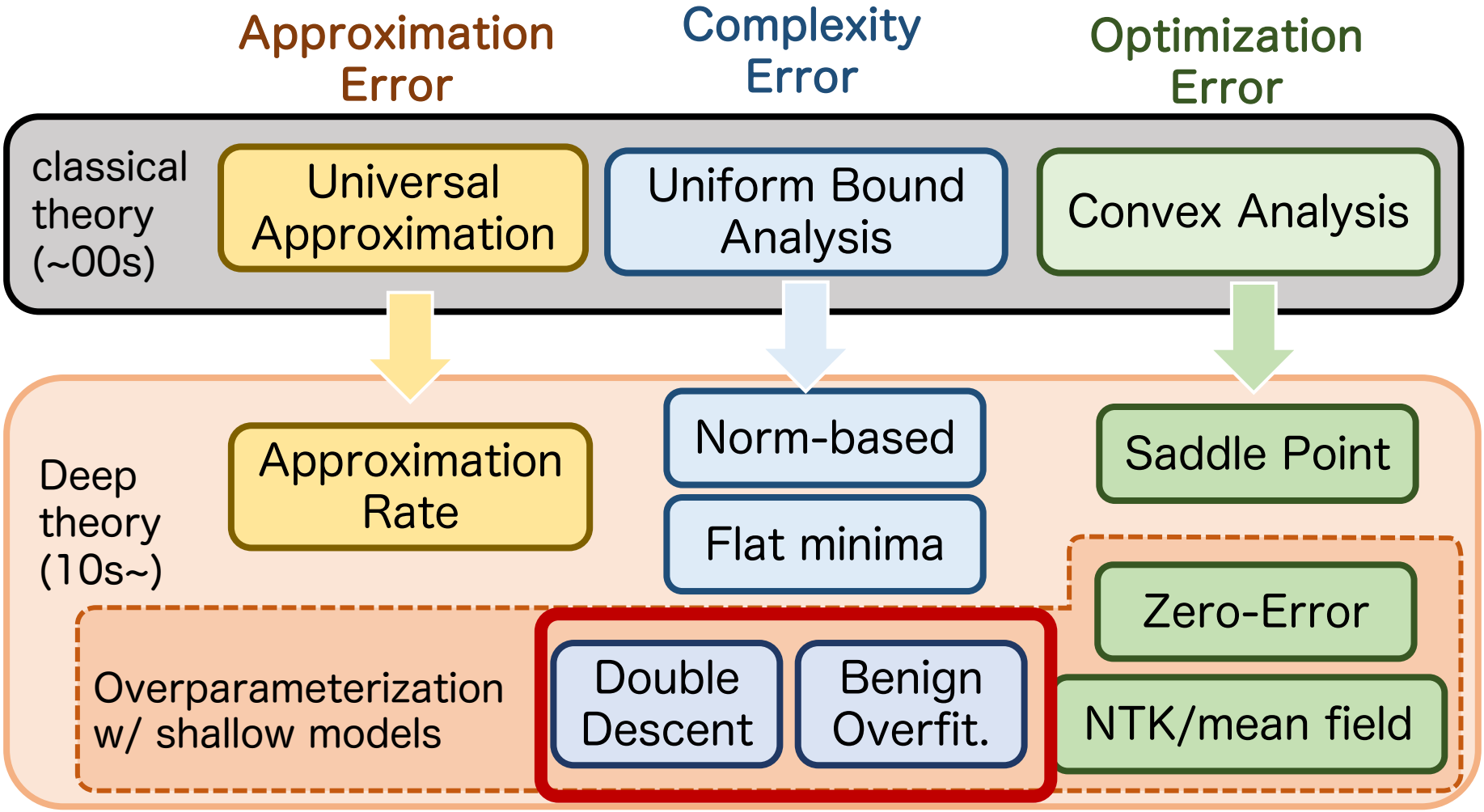
As data size increases, the norm of parameters increases.
(Nagarajan+ 2019)

→ Need to resolve this gap.

Nagarajan, V., & Kolter, J. Z. (2019). Uniform convergence may be unable to explain generalization in deep learning. *NIPS*.

Overparameterization and Shallow Models

Theory map


Approximation Error
Complexity Error
Optimization Error
classical theory (~00s)
Universal Approximation
Uniform Bound Analysis
Convex Analysis
Deep theory (10s~)
Approximation Rate
Norm-based
Flat minima
Saddle Point
Zero-Error
NTK/mean field
Overparameterization w/ shallow models
Double Descent
Benign Overfit.

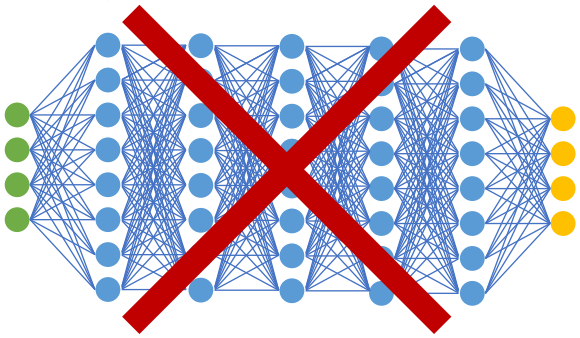
Overparameterization

Setup

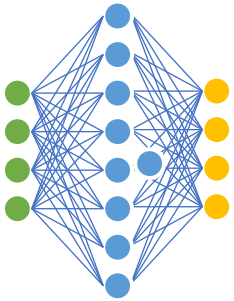
- Focus on # of prameters ≫ # of data

Limitation: Shallow model

- DNN is difficult to analyze
  →Mainly focus on 1 or 2-layer model (linear model)



$f(x) = A_L\sigma(A_{L-1}\sigma(A_{L-2}\cdots A_2\sigma(A_1x))$
Deep model -> No linearity



$f(x) = A_2\sigma(A_1x)$
Shallow model -> Linear in $\sigma(A_1x)$

Interest: Linear Model

Linear Regression

- Data $D_n = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in \mathbb{R}^p$
- Regressino model

$$y_i = \beta^{*\top} x_i + \varepsilon_i, \quad \beta^* \in \mathbb{R}^p$$

2-layer Neural net

- Data $D_n = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in \mathbb{R}^d$
- $A \in \mathbb{R}^{p \times d}$: weight matrix at 1st layer
- σ: activation

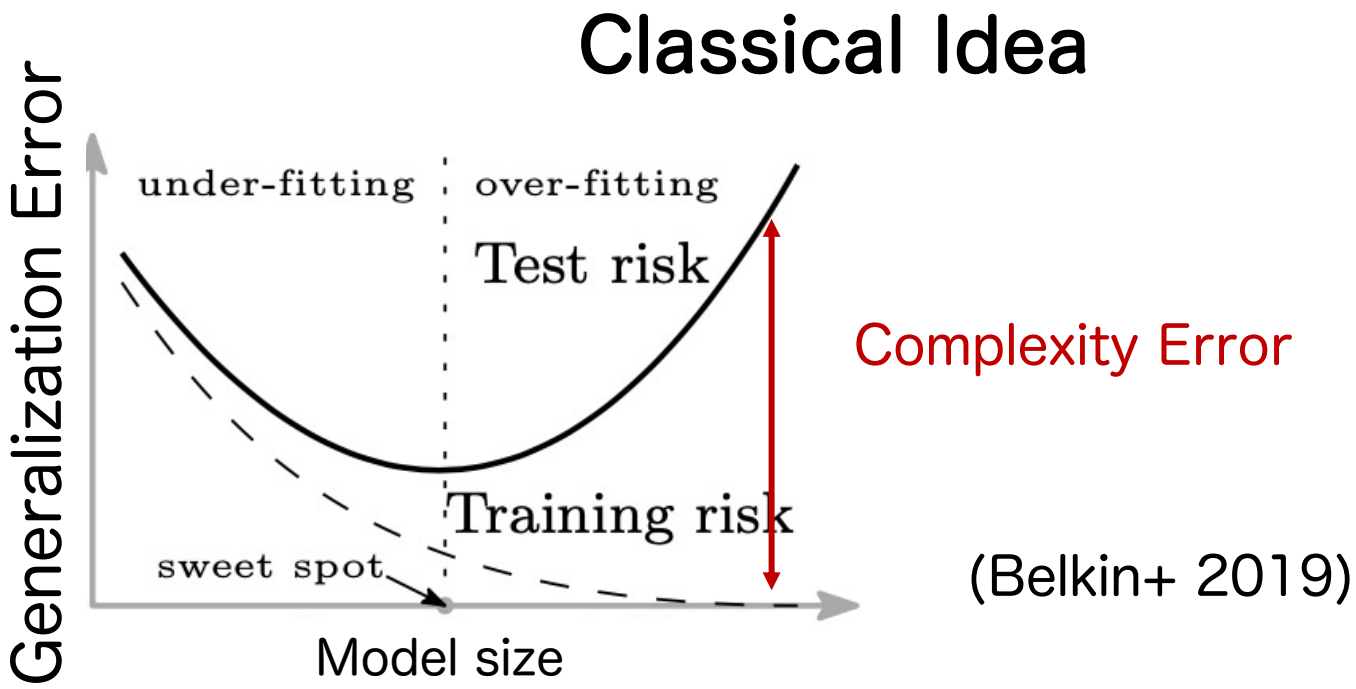
$$y_i = \beta^{*\top} \sigma(A x_i) + \varepsilon_i, \quad \beta^* \in \mathbb{R}^p$$

- With random A, it is close to the linear regression.

Double Descent

Idea of Double Descent

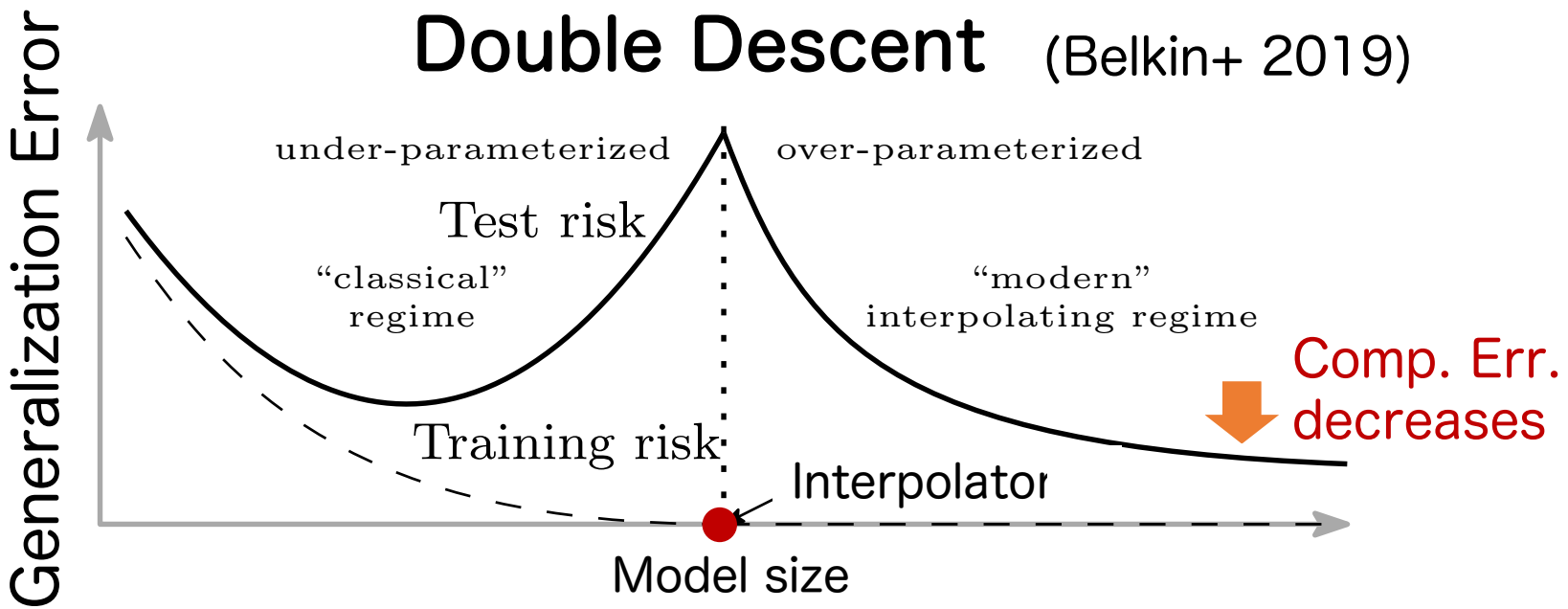
Double Descent
With an excessively large model,
the (generalization) error decreases again.



Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *PNAS*.

Idea of Double Descent

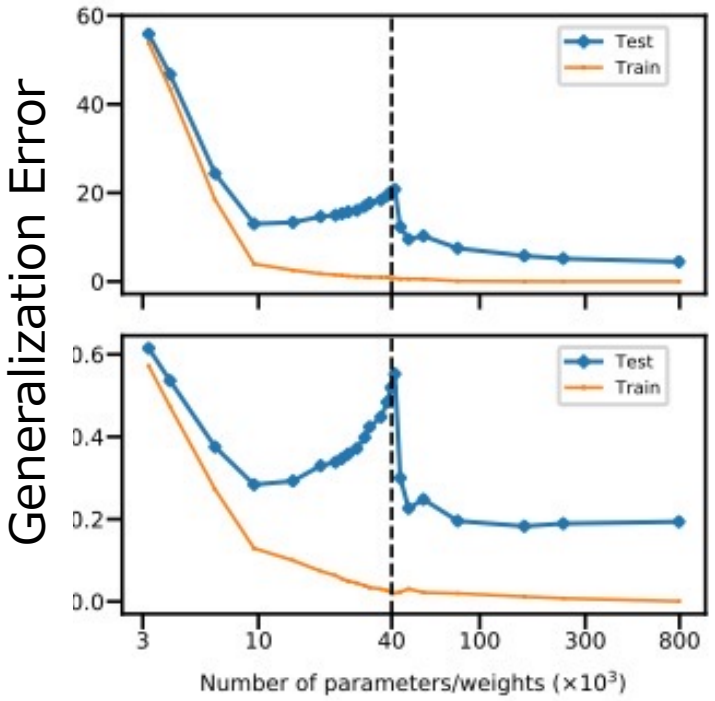
Double Descent
With an excessively large model,
the (generalization) error decreases again.

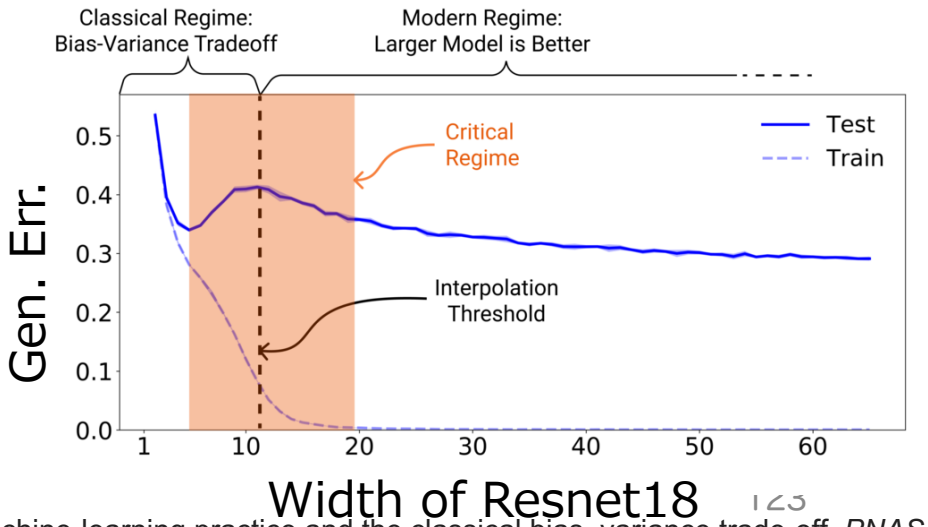


Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *PNAS*.

Experiment

- Simple model (linear regression, shallow NN)
  - As increasing parameters, the error increases then decreases. (Belkin+ 2019)
- Deep model
  - Same result appears in Resnet and CNN (Nakkiran+ 2020)





Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *PNAS*.

Theory Explains Double Descent?

Linear Regression

- Data $D_n = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in \mathbb{R}^p$

$$y_i = \beta^{*\top} x_i + \varepsilon_i, \quad \beta^* \in \mathbb{R}^p$$

Estimator (interpolator)

$$\hat{\beta} = \mathrm{argmin}\{\|\beta\|_2 : \beta \text{ minimizes } \Sigma_{i=1}^n (y_i - \beta^\top x_i)^2\}$$

Generalization Error

$\|\beta\|_\Sigma^2 = \beta^\top \Sigma \beta$

- $\Sigma = \mathrm{E}[x_i x_i^\top]$: covariance matrix

$$R(\hat{\beta}) = E_\varepsilon\left[\left\|\hat{\beta} - \beta^*\right\|_\Sigma^2\right] = \left\|E_\varepsilon[\hat{\beta}] - \beta^*\right\|_\Sigma^2 + \mathrm{tr}\left[\mathrm{Cov}_\varepsilon(\hat{\beta})\Sigma\right]$$

$= B$ (bias) (approx. err.)

$= V$ (variance) (≈comp. err.)

Hastie, T., Montanari, A., Rosset, S., & Tibshirani, R. J. (2022). Surprises in high-dimensional ridgeless least squares interpolation. *AoS*.

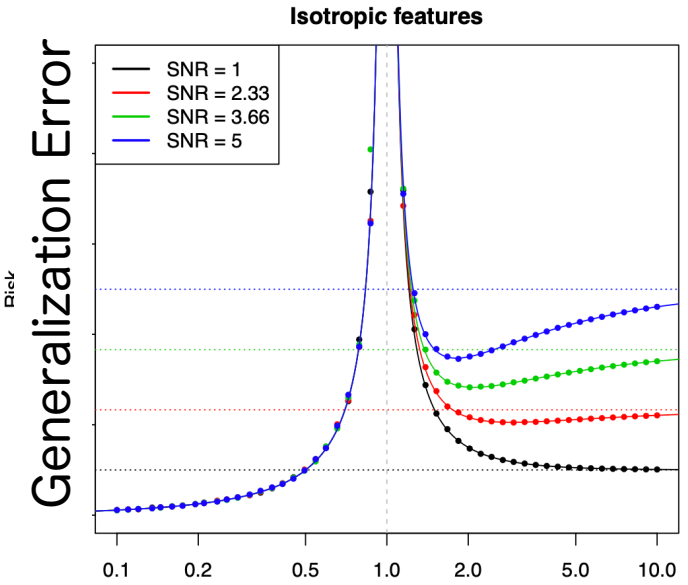
Theory Explains Double Descent

Theorem

$\gamma = \frac{p \ \ (\#\text{ of parameter})}{n \ \ (\#\text{ of data})}$, σ^2: noise variance

$$\lim_{p,n\to\infty} R(\hat{\beta}) = \begin{cases} \frac{\sigma^2\gamma}{1-\gamma}, & (\gamma < 1) \\ \|\beta^*\|_2^2(1-\gamma^{-1}) + \frac{\sigma^2}{\gamma-1}, & (\gamma > 1) \end{cases}$$

$= B$ (bias) $= V$ (variance)



As γ increases, the error increases then decreases.

The result simulates the double descent phenomena.

Hastie, T., Montanari, A., Rosset, S., & Tibshirani, R. J. (2022). Surprises in high-dimensional ridgeless least squares interpolation. *AoS*.

How it works?

Rewrite $V(\hat{\beta})$ by eigenvalues of Σ

- $X = (x_1, \ldots, x_n)^\top, Z = X\Sigma^{1/2}$ ($n \times p$ matrix)
- Empirical covariance $\hat{\Sigma} = X^\top X/n$ (random)
- $\lambda_j(A)$: j-th largest eigenvalue of A

$$V(\hat{\beta}) = \frac{\sigma^2}{n}\mathrm{tr}\left(\hat{\Sigma}^{-1}\Sigma\right) = \frac{\sigma^2}{n}\sum_{j=1}^{p}\frac{1}{\lambda_j(Z^\top Z/n)}$$

$$= \frac{\sigma^2 p}{n}\int \frac{1}{s} dF_{Z^\top Z/n}(s)$$

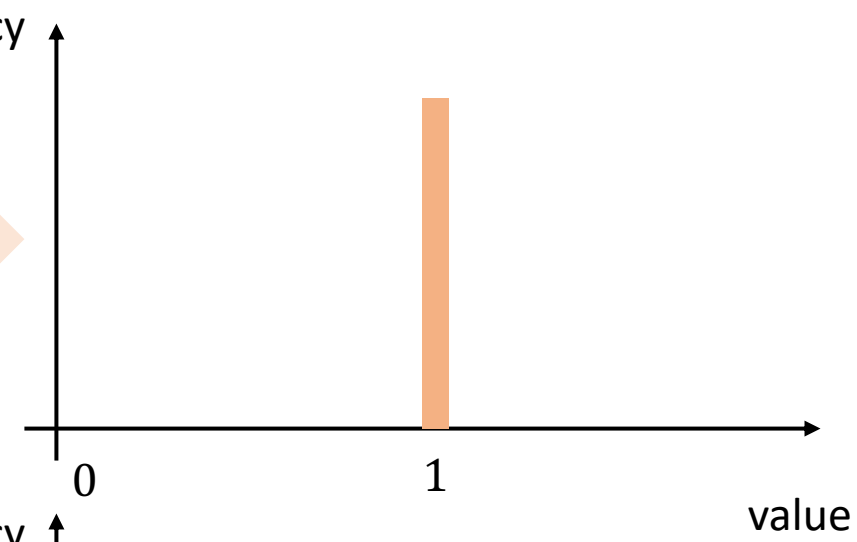
$F_{Z^\top Z/n}(s)$: eigenvalue distribution of $Z^\top Z/n$

Eigenvalue Distribution

Identity matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

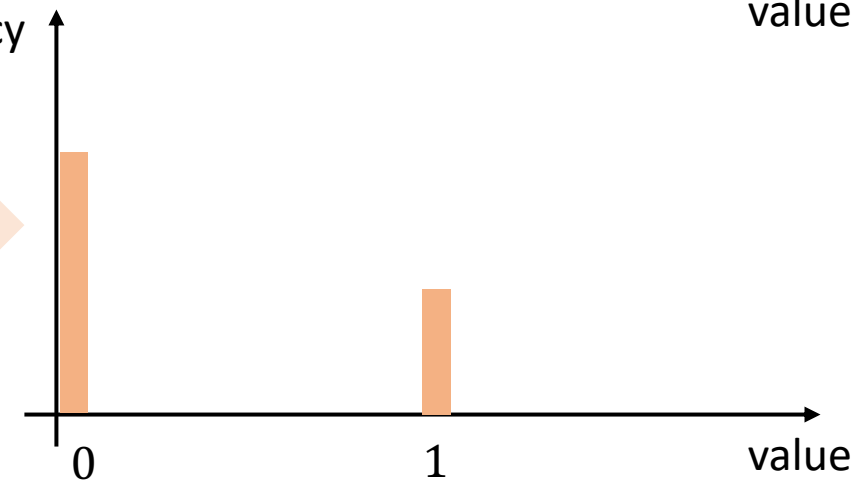
eigenvalues
1,1,1



Singular matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

eigenvalues
1,0,0

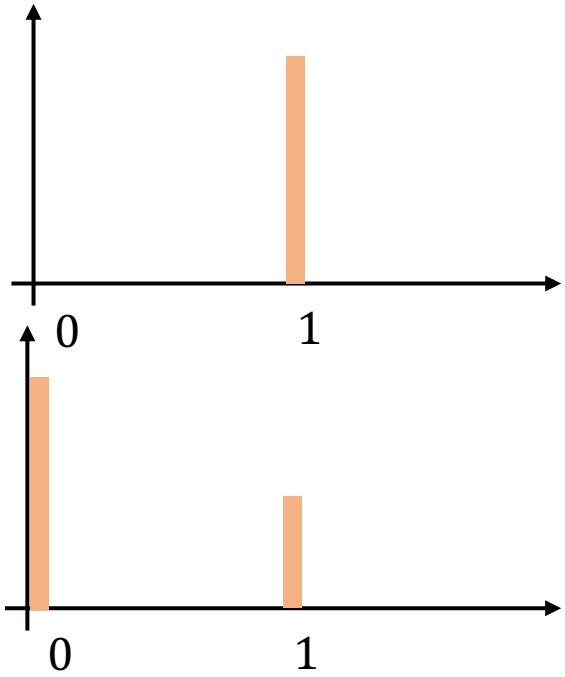


It describes the divergence of information in the matrix.

Variance by Eigenvalues

Write variance by an inverse of eigenvalues

$$V(\hat{\beta}) = \frac{\sigma^2 p}{n} \int \frac{1}{s} dF_{Z^\top Z/n}(s)$$



Positive eigenvalues

→Finite variance

Zero eigenvalues

(i.e.: $p > n$)

→Diverged variance

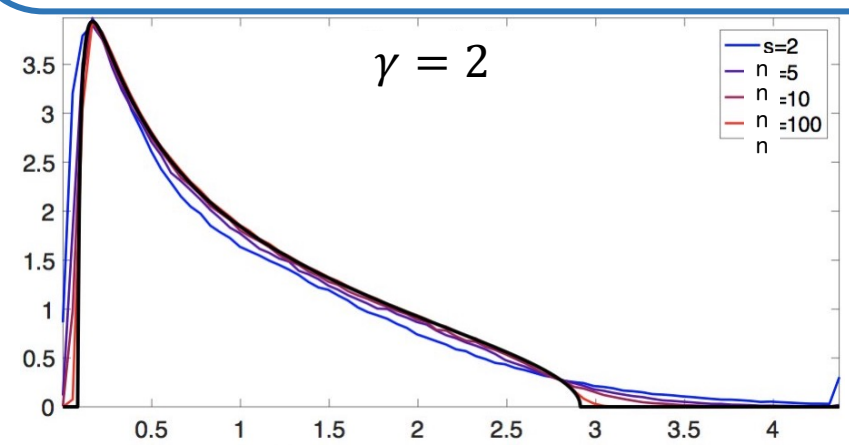
Important to see whether the distribution has mass on 0.

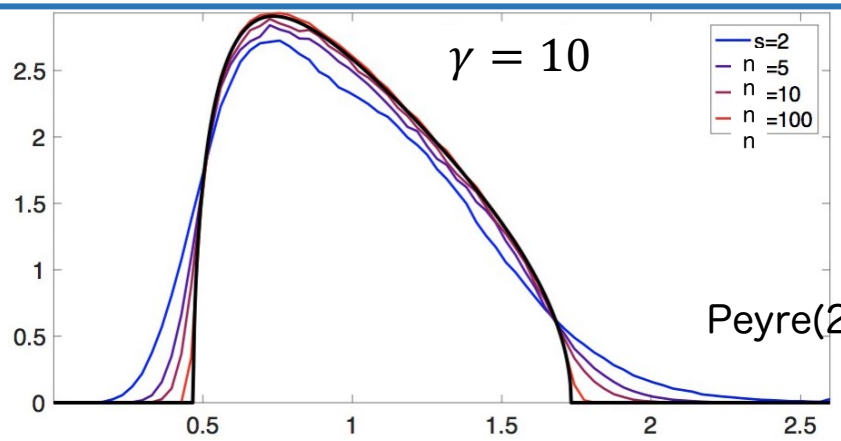
Eigenvalue distribution

Marchenko-Pastur Law

$$\lim_{n,p\to\infty,\ p/n\to\gamma} F_{Z^\top Z/n} = F_\gamma$$

$$dF_\gamma(s) = \frac{\gamma}{2\pi s}\sqrt{(s-s_-)(s_+-s)}\,1_{[s_-,s_+]}, s_\pm = \left(1\pm\sqrt{1/\gamma}\right)^2$$





Peyre(2020)

As model size (γ) increases, the distribution gets far from 0.

Hence, the variance does not diverge with large γ.

Peyre(2020): https://twitter.com/gabrielpeyre/status/928907946108571649

Extension to NNs

- For 2-layer NN, the similar result is proved.

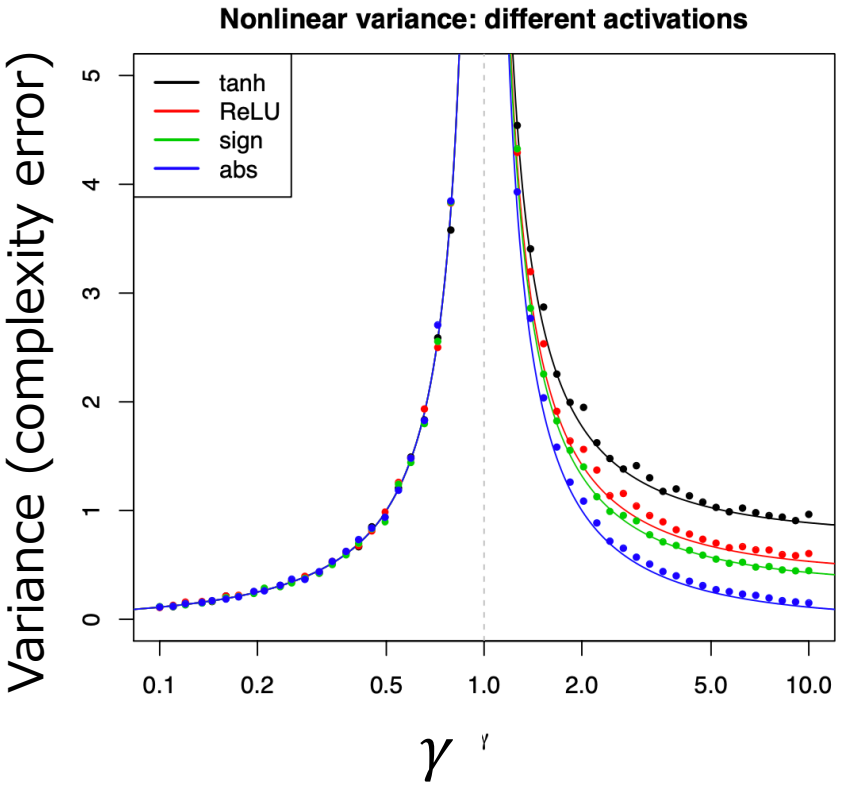
2-layer NN

$$f(x) = \sum_{j=1}^{N} w_j \sigma\left(a_j^\top x + b\right)$$

- 1st layer : random a_j, b
- 2nd layer : train w_j by GD

→ **Close to linear regression.**

To apply the similar proof, important to make the model close to linear regression.



Benign Overfitting

Setting: Linear Regression

Linear Regression

- Data $D_n = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in \mathbb{R}^p$

$$y_i = \beta^{*\top} x_i + \varepsilon_i, \quad \beta^* \in \mathbb{R}^p$$

Estimator (interpolator)

$$\hat{\beta} = \mathrm{argmin}\{\|\beta\|_2 : \beta \text{ minimizes } \Sigma_{i=1}^n (y_i - \beta^\top x_i)^2\}$$

Generalization Error

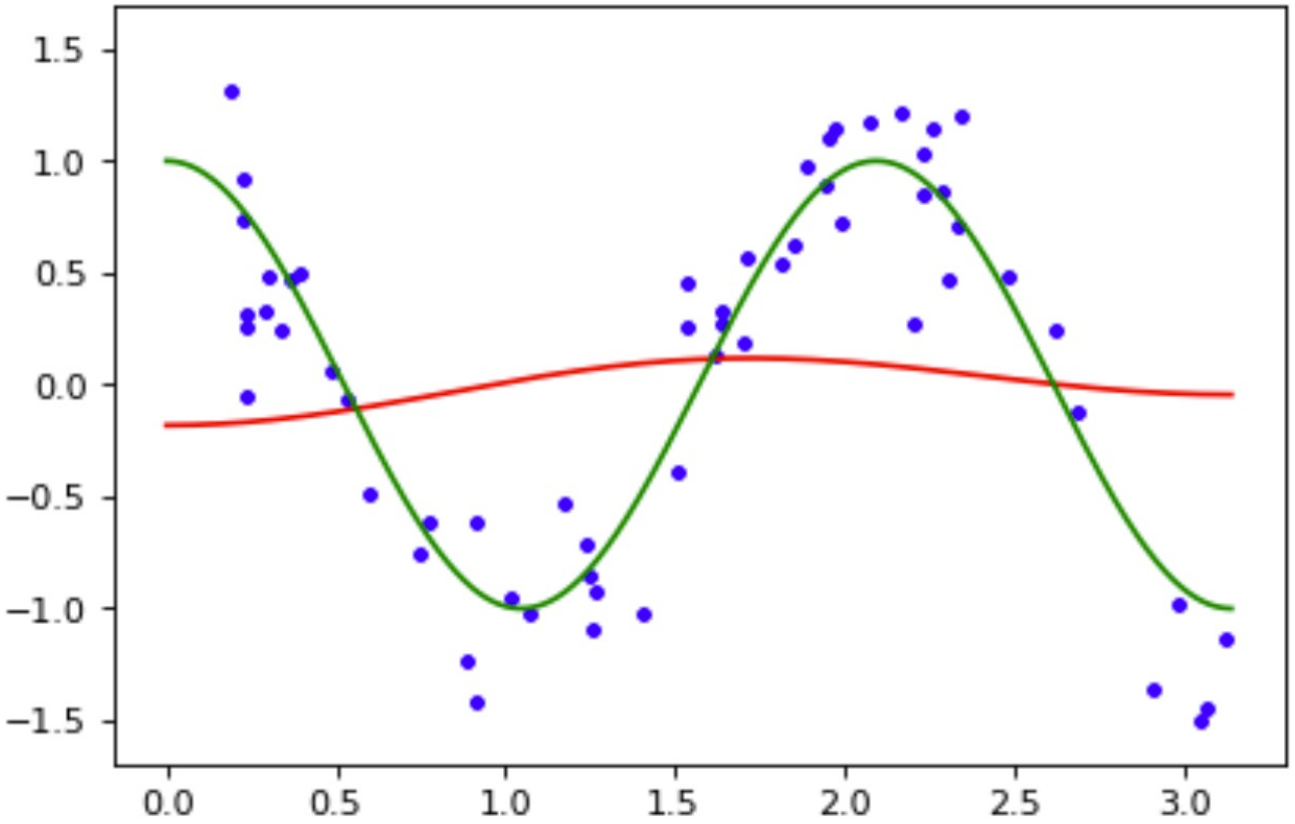
- $\Sigma = \mathrm{E}[x_i x_i^\top]$: covariance matrix

$$\|\beta\|_\Sigma^2 = \beta^\top \Sigma \beta$$

$$R(\hat{\beta}) = E_\varepsilon \left[\left\| \hat{\beta} - \beta^* \right\|_\Sigma^2 \right]$$

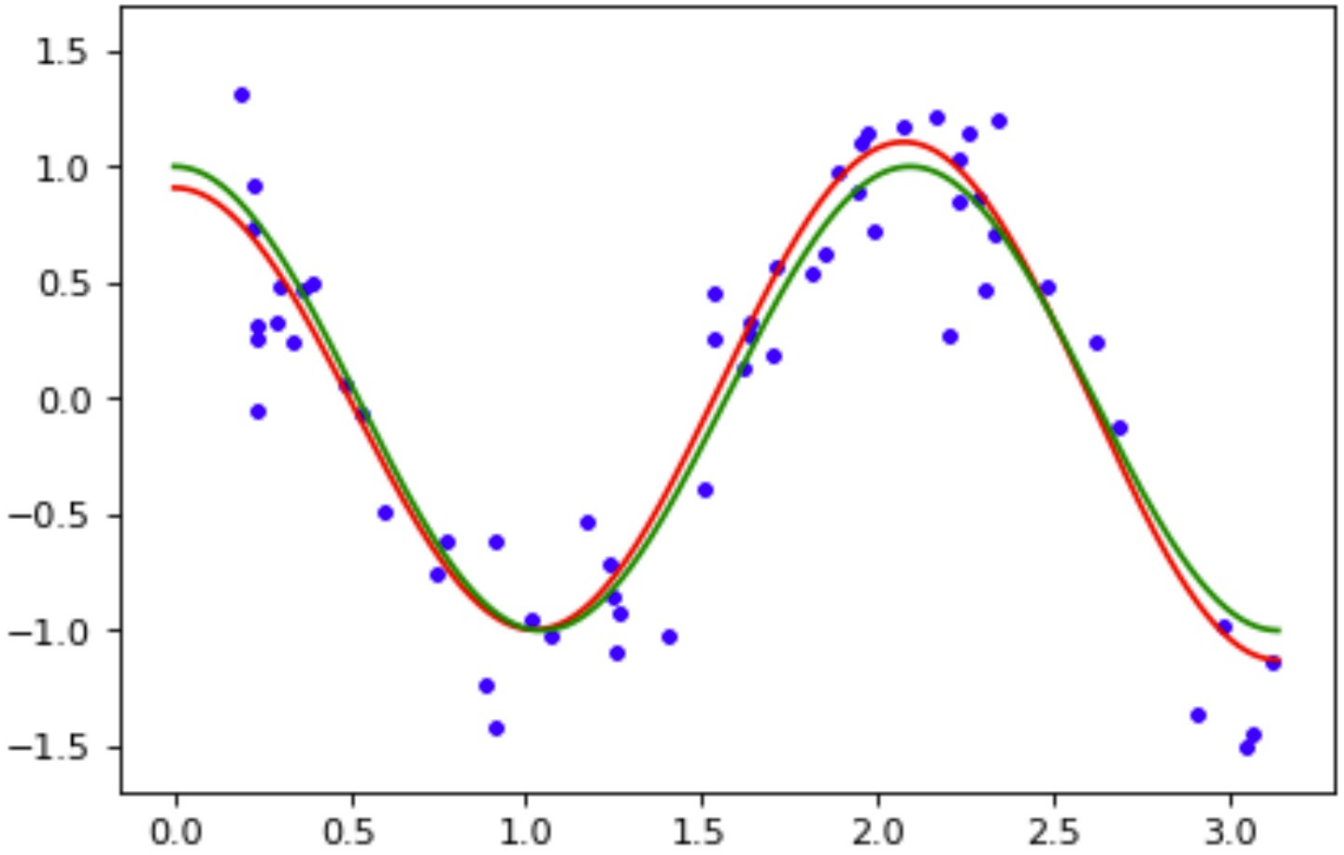
Bartlett, P. L., Long, P. M., Lugosi, G., & Tsigler, A. (2020). Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, *117*(48), 30063-30070.

Benign Overfitting



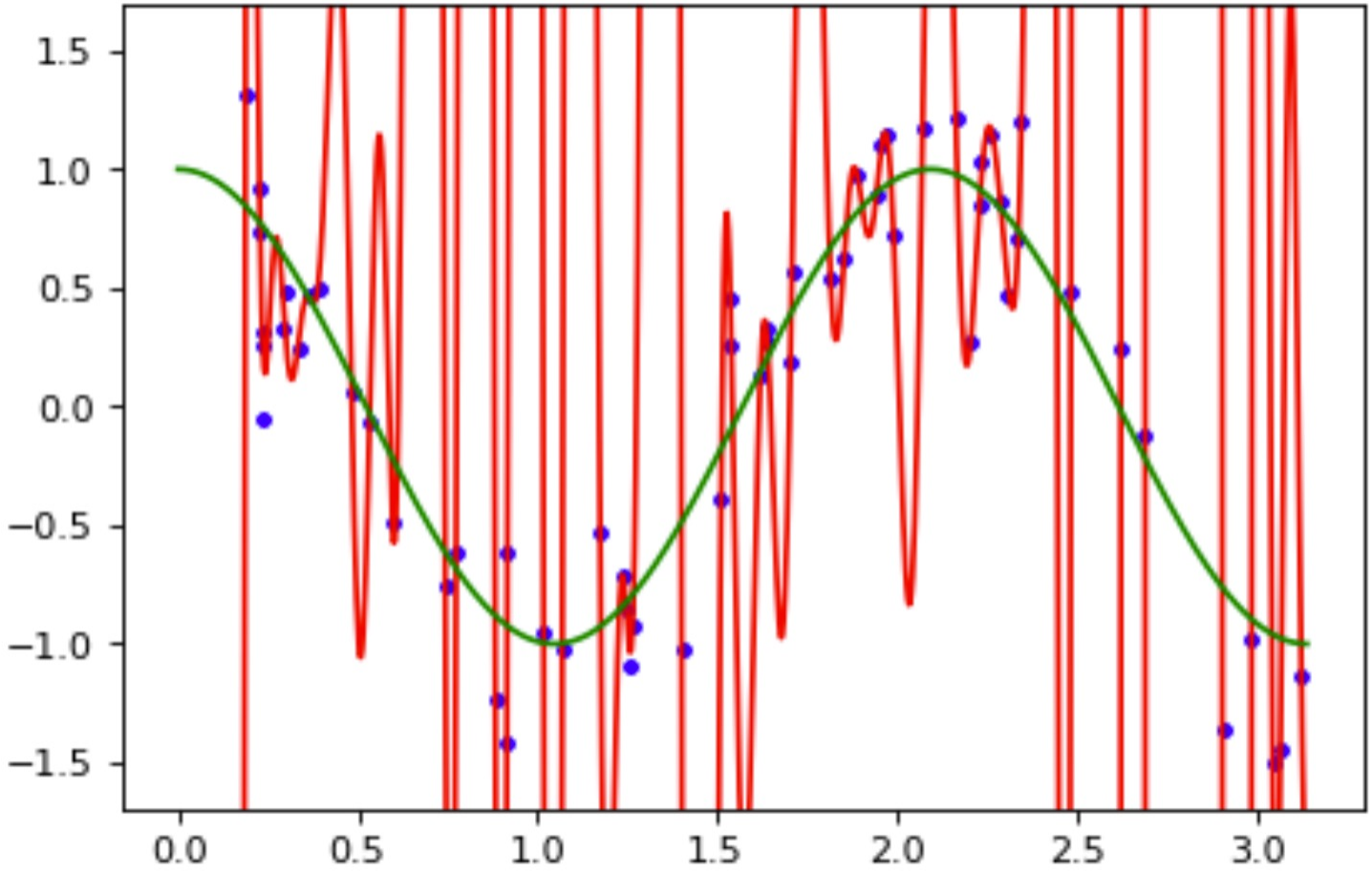
Green：True f.、Blue：Data、Red：Estimated f、**# of params: 2**
# of data: 60

Benign Overfitting



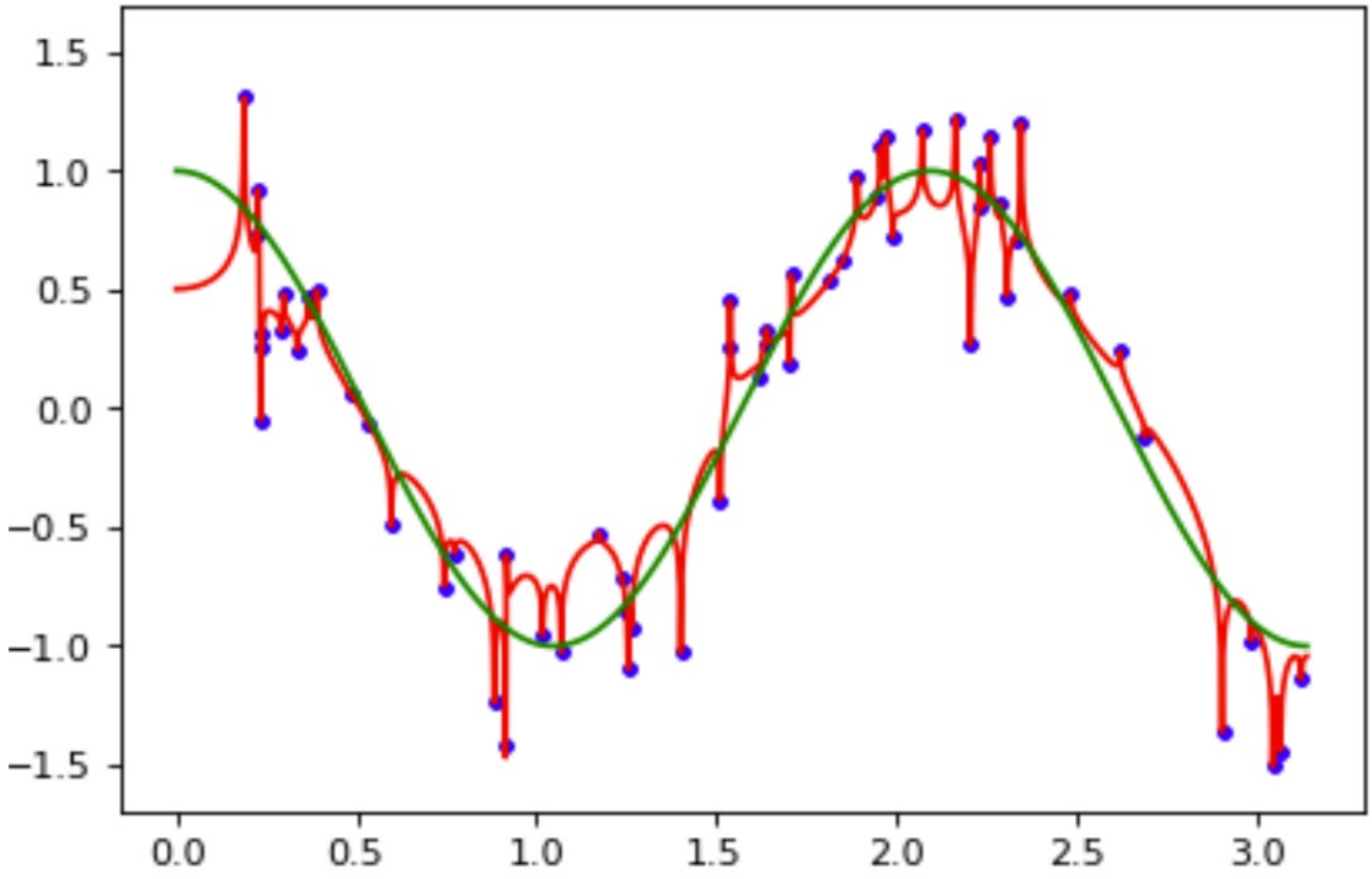
Green：True f.、Blue：Data、Red：Estimated f、**# of params: 3**
# of data: 60

Benign Overfitting



Green：True f.、Blue：Data、Red：Estimated f、# of params: 50
# of data: 60

Benign Overfitting



Green：True f.、Blue：Data、Red：Estimated f、# of params: 2000

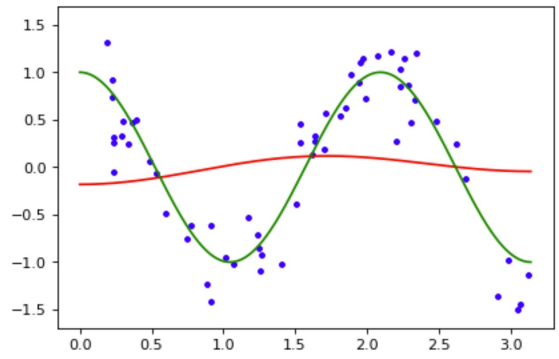
# of data: 60

Benign-Overfitting

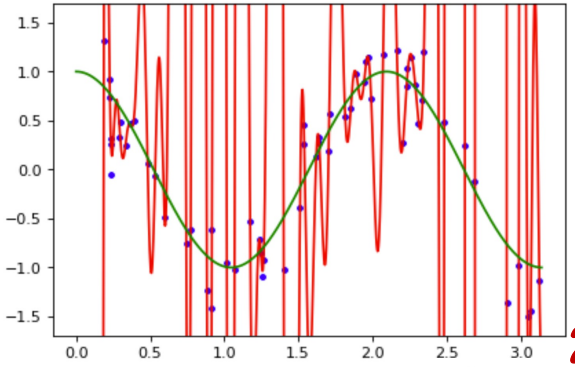
Large models combine good fit to training data with high predictive performance

Green：True f.、Blue：Data、Red：Estimated f

2 params



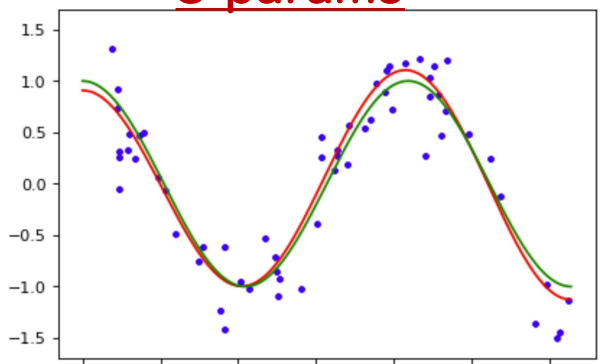
50 params



⇦ Overfitting

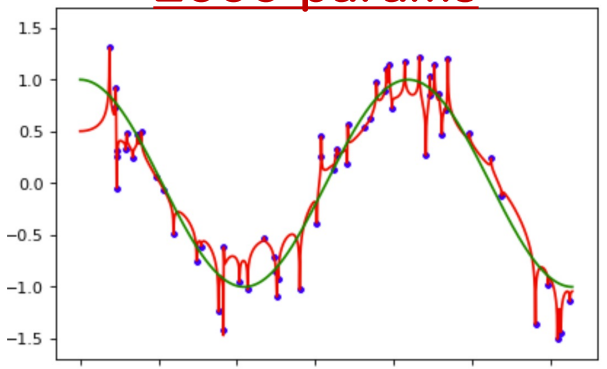
⇩ Benign Overfitting

3 params



Fitting ⇨

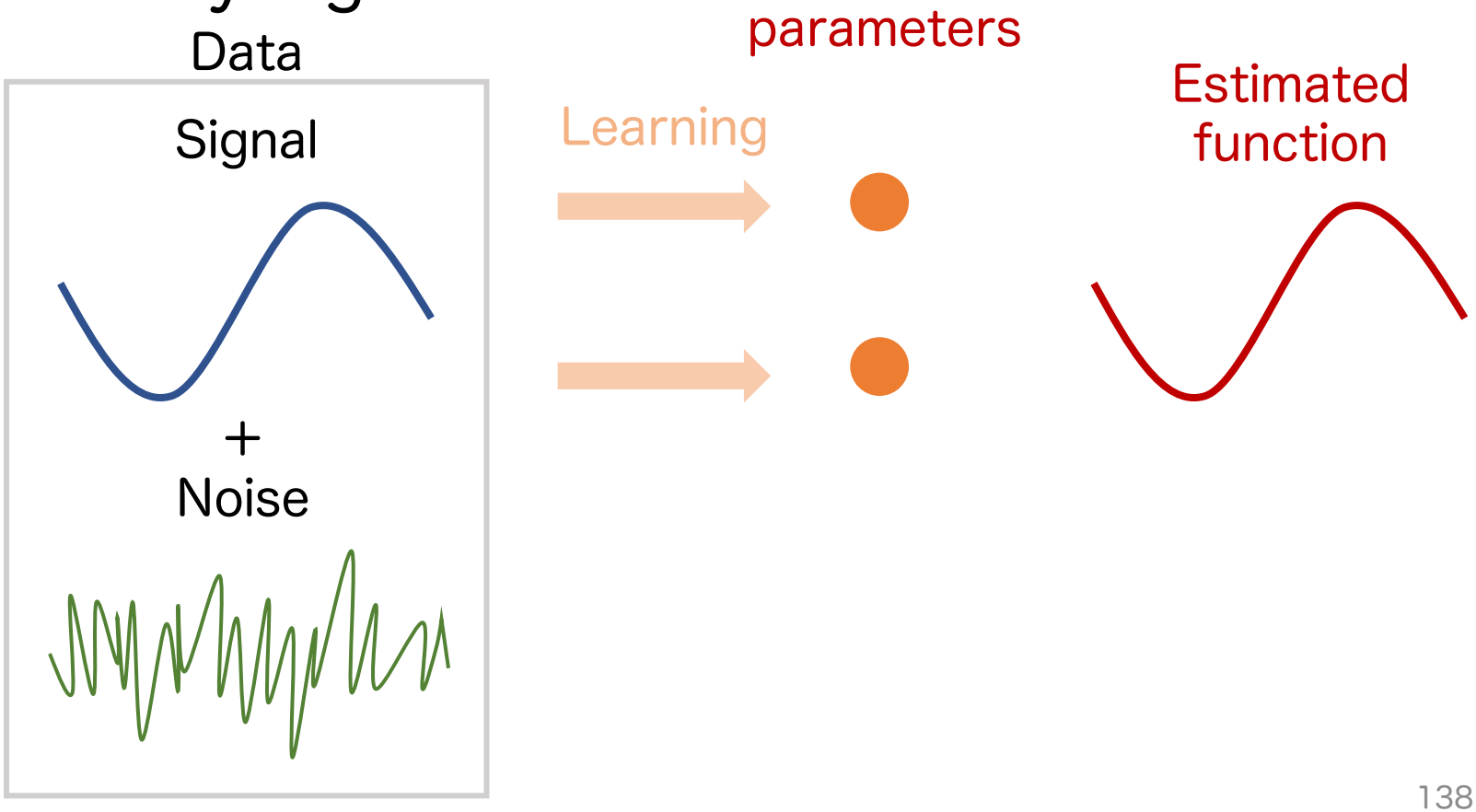
2000 params



Tsigler, A., & Bartlett, P. L. (2020). Benign overfitting in ridge regression. *arXiv preprint arXiv:2009.14286*.

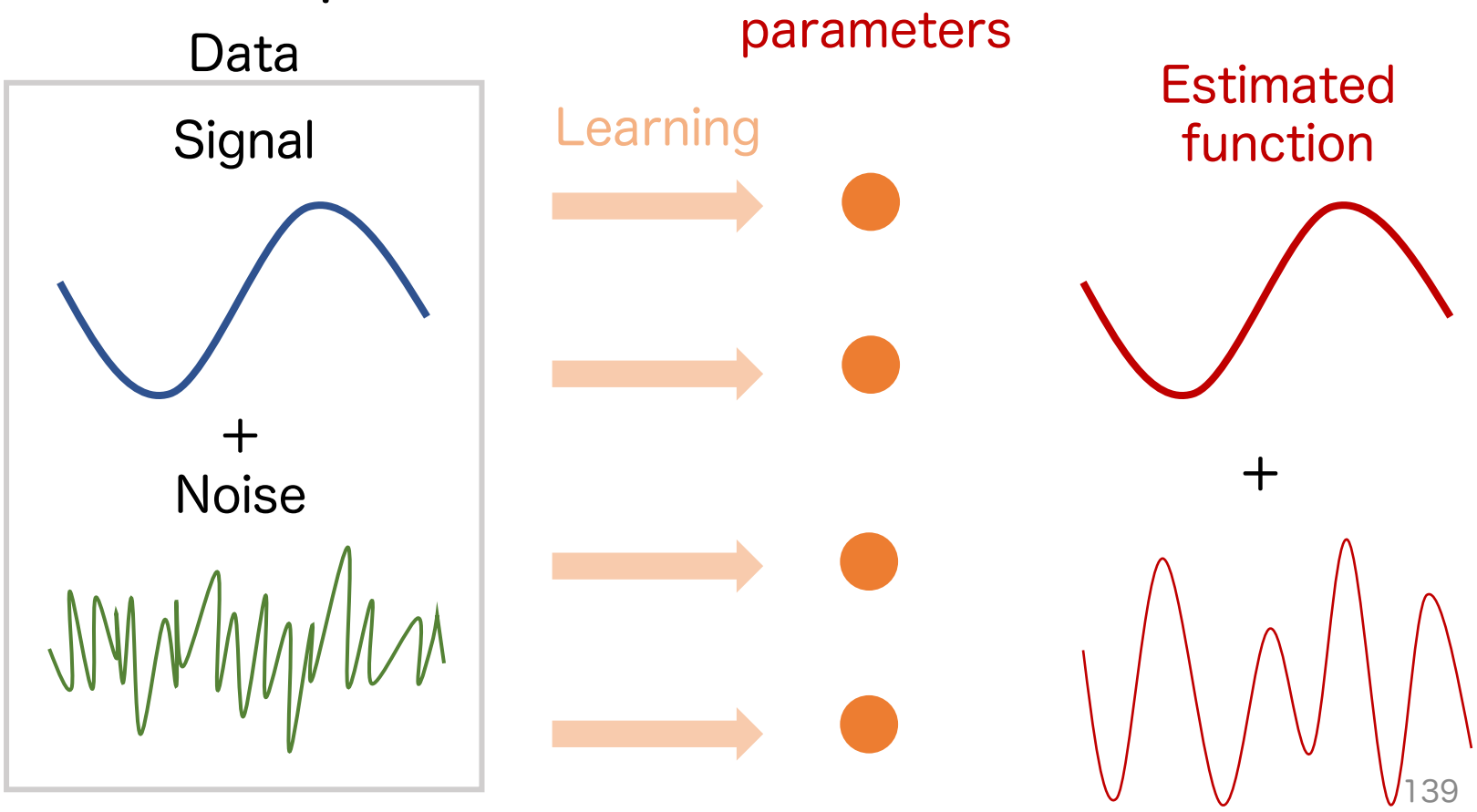
Intuition

- Less parameters
  →Only signals are learned.



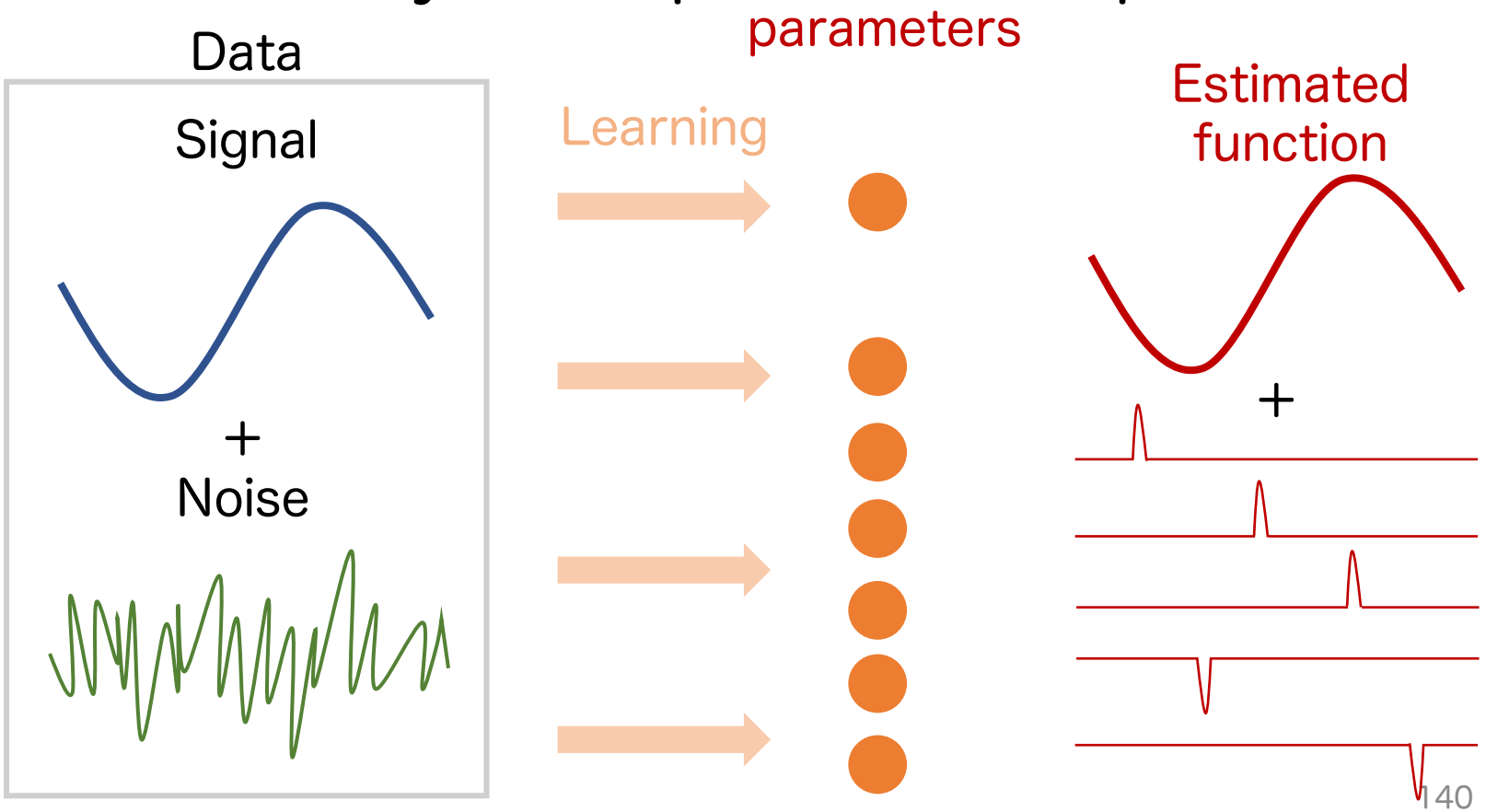
Intuition

- More parameters
  → Extra parameters learn noise as well



Intuition

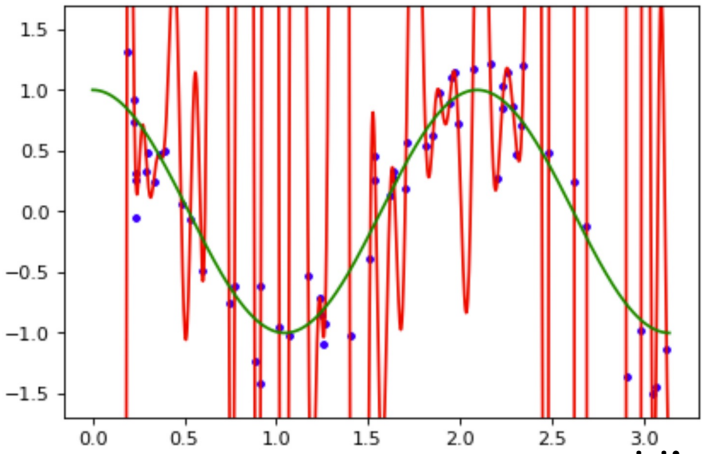
- Overparameterization
  → Too many extra parameters split the noise.



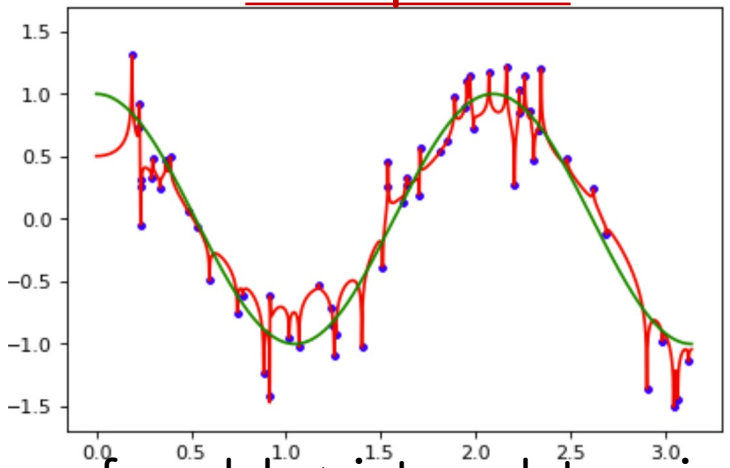
Analysis of Benign Overfitting

- With excess parameters, the model cancels out noise with high-frequency components

50 params



2000 params



High frequency of model → interpolate noise
Low frequency of model → learn the true function

Bound on generalization error

- $\mathrm{tr}(\Sigma)$: trace of Σ (sum of eigenvalues), $R_k(\Sigma)$: effective rank of Σ

$$R(\hat{\beta}) \leq c\left\{\|\theta^*\|_2^2 \sqrt{\frac{\mathrm{tr}(\Sigma)}{n}} + \left(\frac{k}{n} + \frac{n}{R_k(\Sigma)}\right)\right\}$$

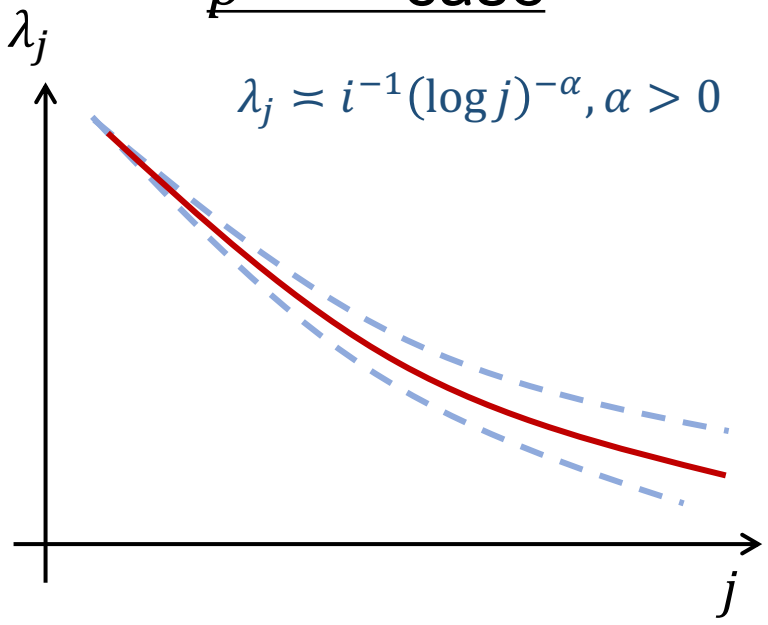
T

λ_j: j-th largest eigenvalue of Σ

Mathematical condition

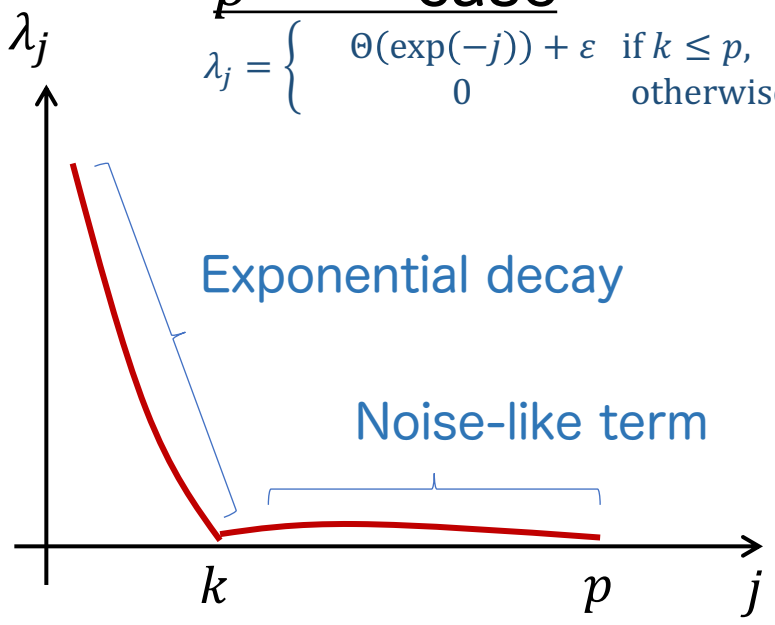
If Σ has specific eigenvalue decay,
the bound on gen. error. converges to zero.

$p = \infty$ case



$p \to \infty$ case

$$\lambda_j = \begin{cases} \Theta(\exp(-j)) + \varepsilon & \text{if } k \le p, \\ 0 & \text{otherwise.} \end{cases}$$



Bartlett, P. L., Long, P. M., Lugosi, G., & Tsigler, A. (2020). Benign overfitting in linear regression. *PNAS*.

Summary of Complexity Error

Classical Idea

- More parameters can overfit
  → Contradiction to actual DL

Theoretical challenges

- Norm-based bound, PAC-Bayes...

Overparameterization

- Study shallow models
  → Find & describe new phenomena

Summary

Deep Learning Theory

Approximation Error

- Nice and intuitive proof
- Explains the power of **deep** neural nets

Complexity Error

- Substantial gap b/w theory and practice
- Still many theoretical challenges for **deep** models
- Interesting progress for **shallow** models

Theoretical challenge

- Dynamic properties of deep models are more difficult