TAUTOLOGY
INNOVATION
SCHOOL

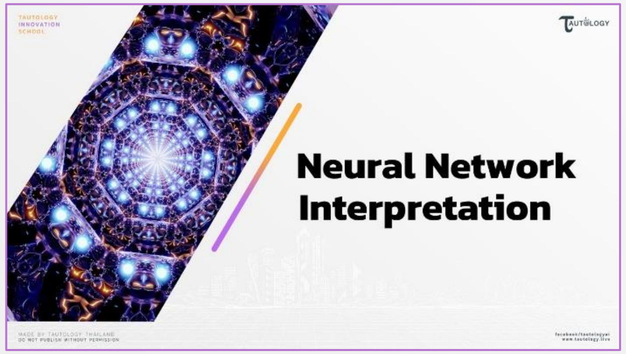
TAUTOLOGY

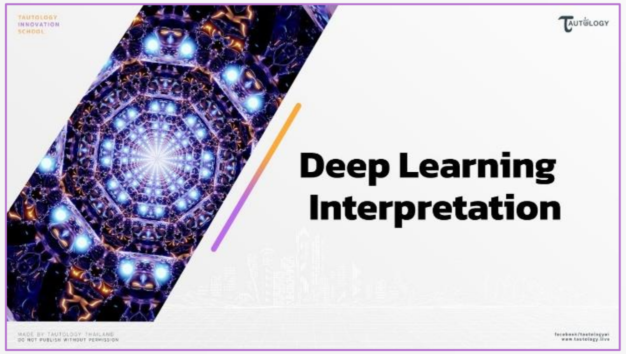
DEEP LEARNING INTERPRETATION

DEEP LEARNING INTERPRETATION

BY TAUTOLOGY

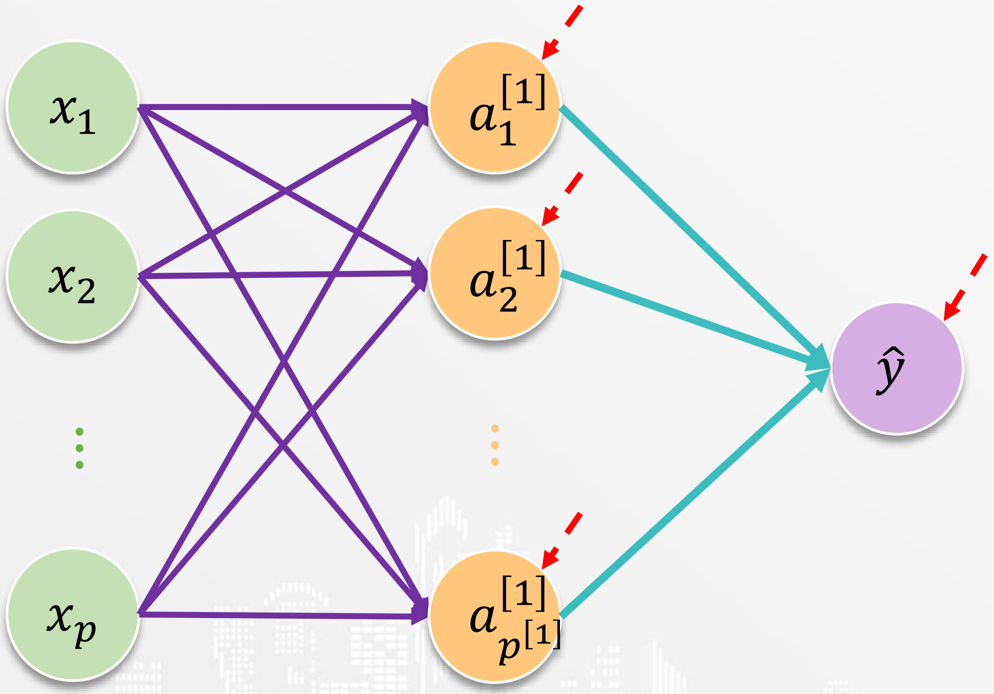
Deep Learning Interpretation





Neural Network Interpretation

Neural Network Interpretation



Neural Network Interpretation

Regression

Classification

Neural Network Interpretation

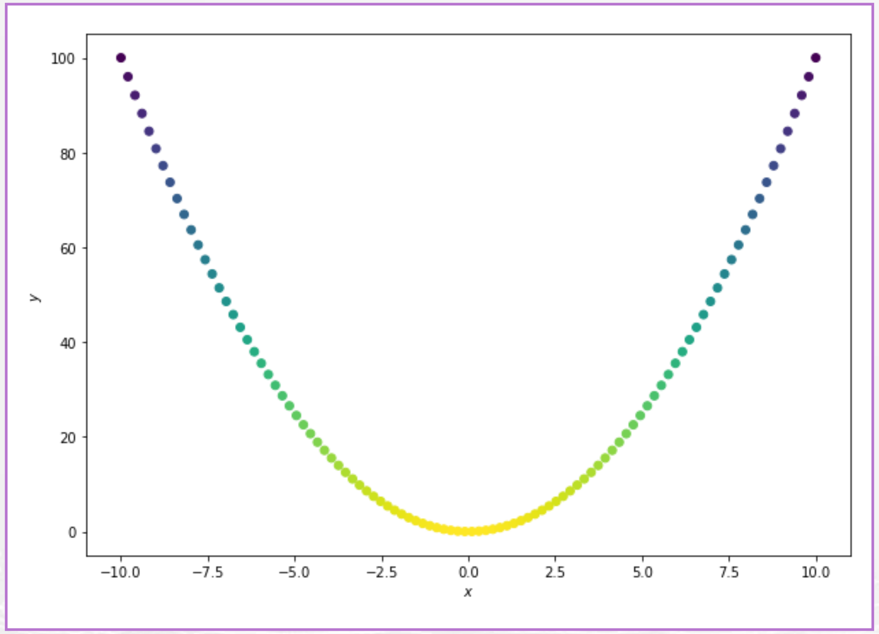
Regression

Classification

Neural Network Interpretation

Regression

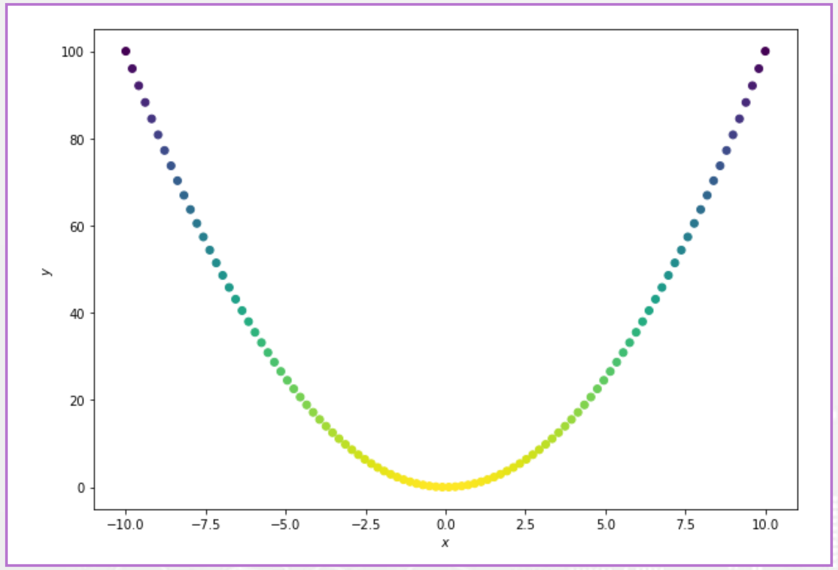
Example 1

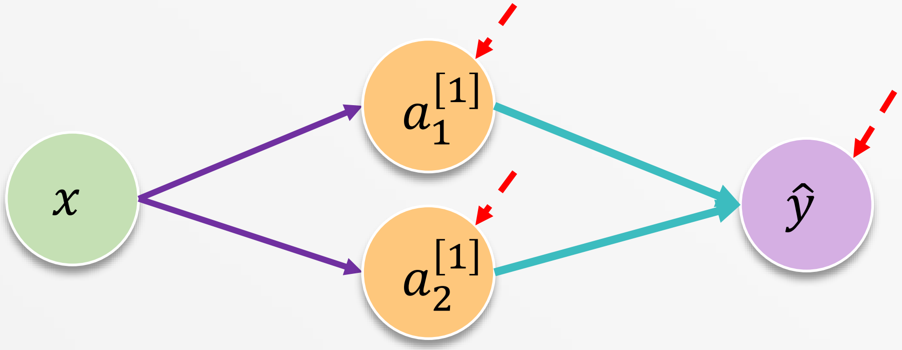


Neural Network Interpretation

Regression

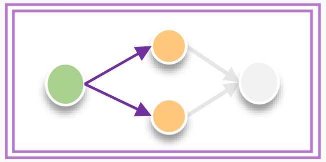
Example 1.1

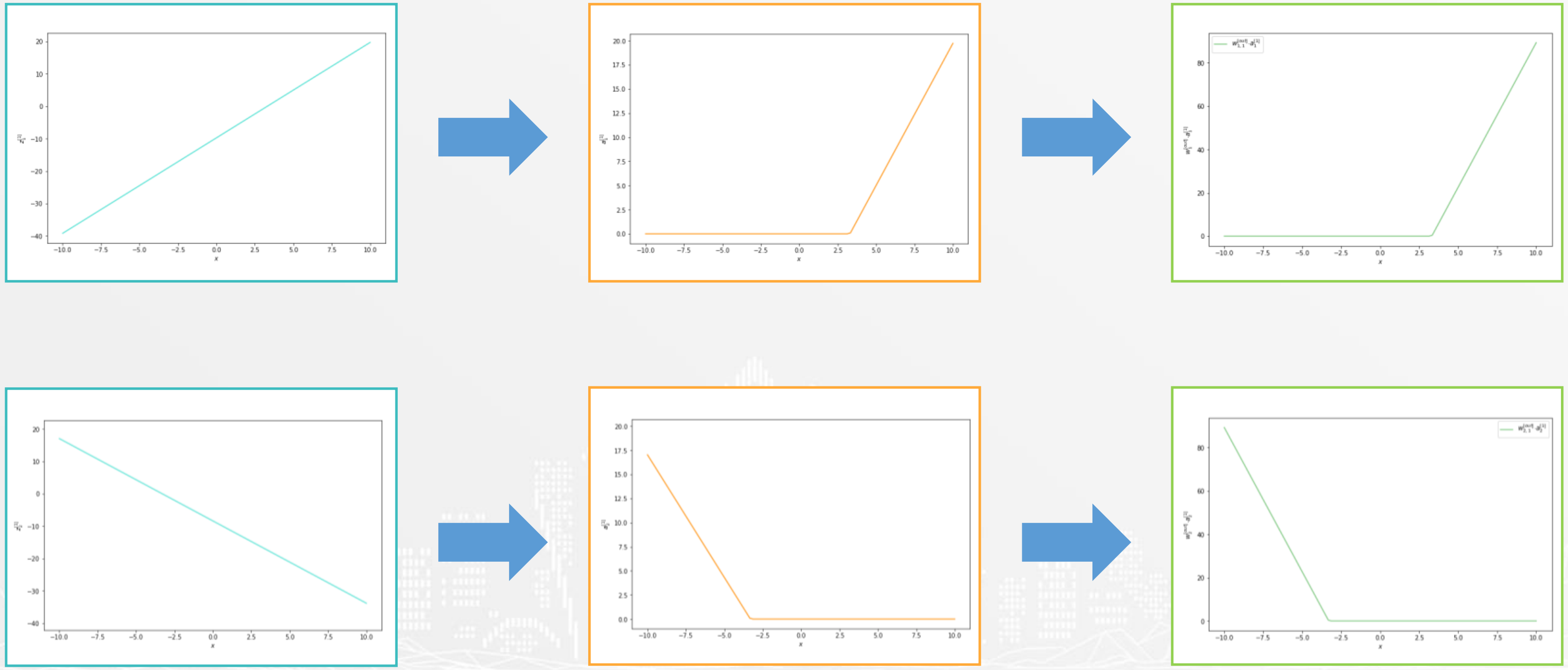




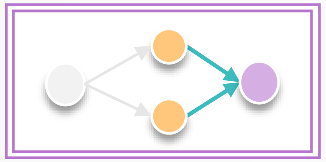
Neural Network Interpretation

Regression **Example 1.1**

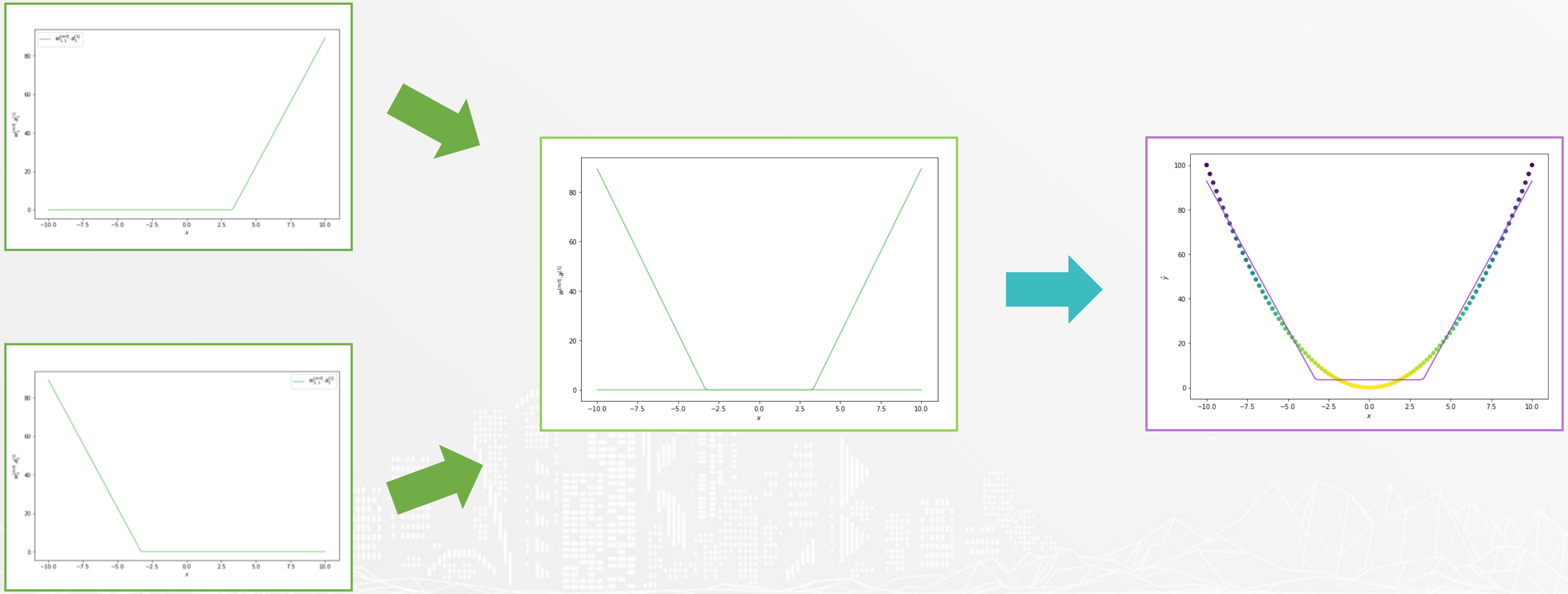




Neural Network Interpretation



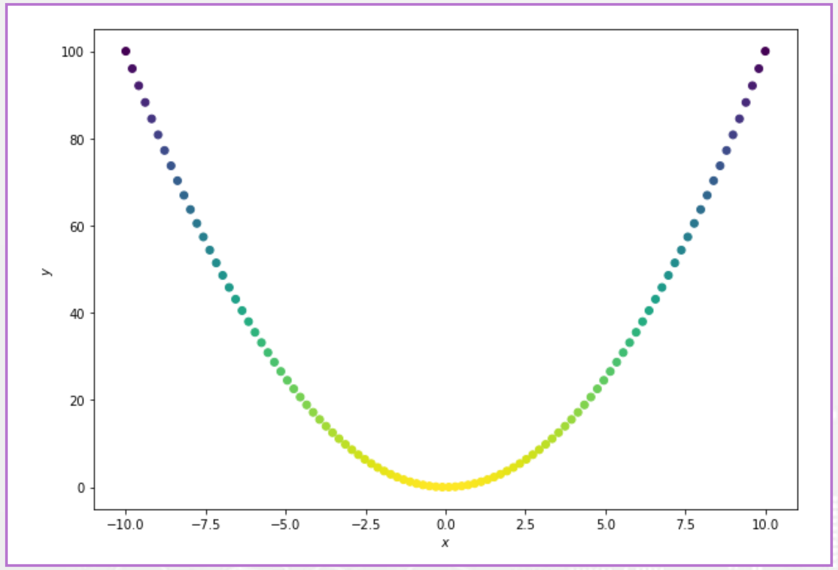
Regression | **Example 1.1**

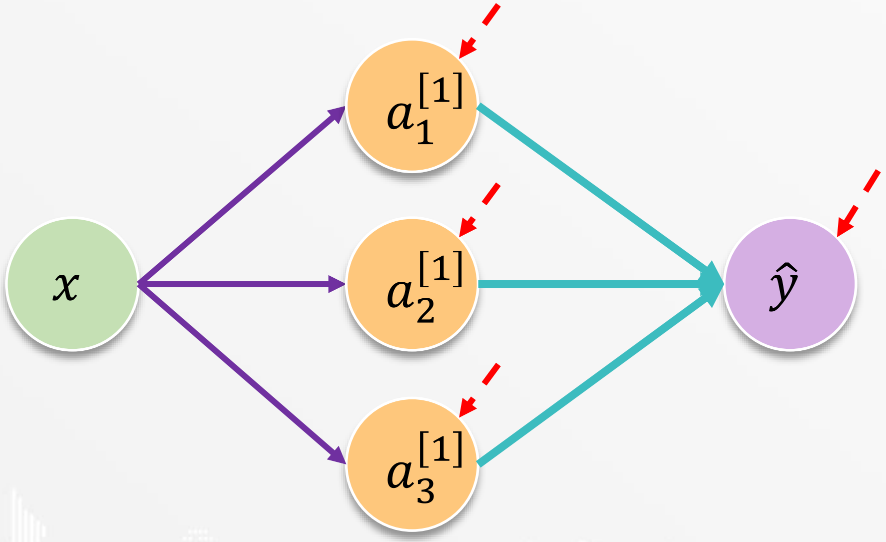


Neural Network Interpretation

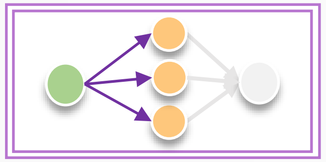
Regression

Example 1.2

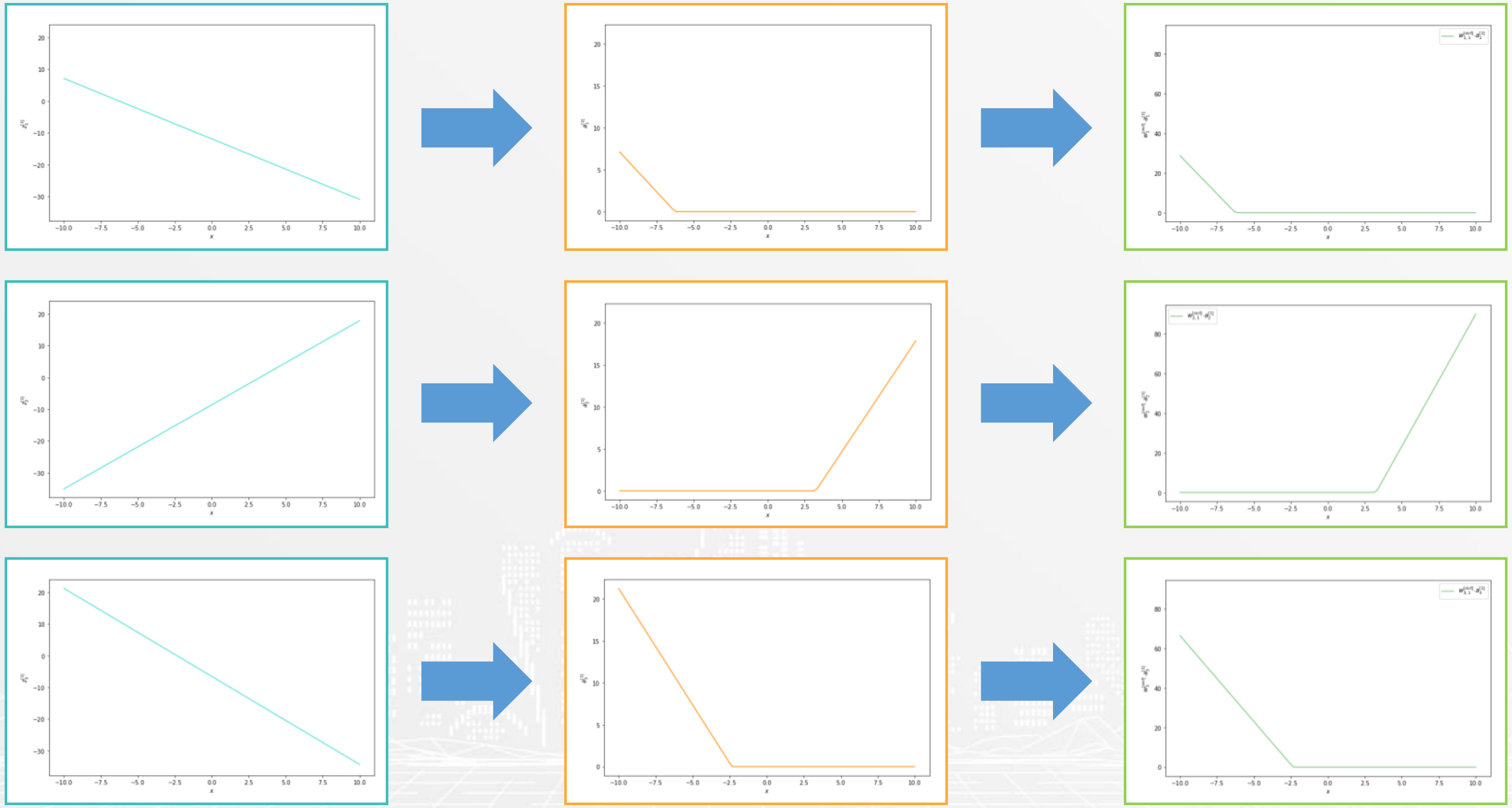




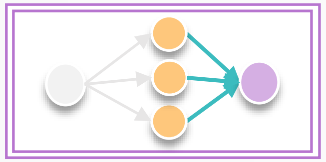
Neural Network Interpretation



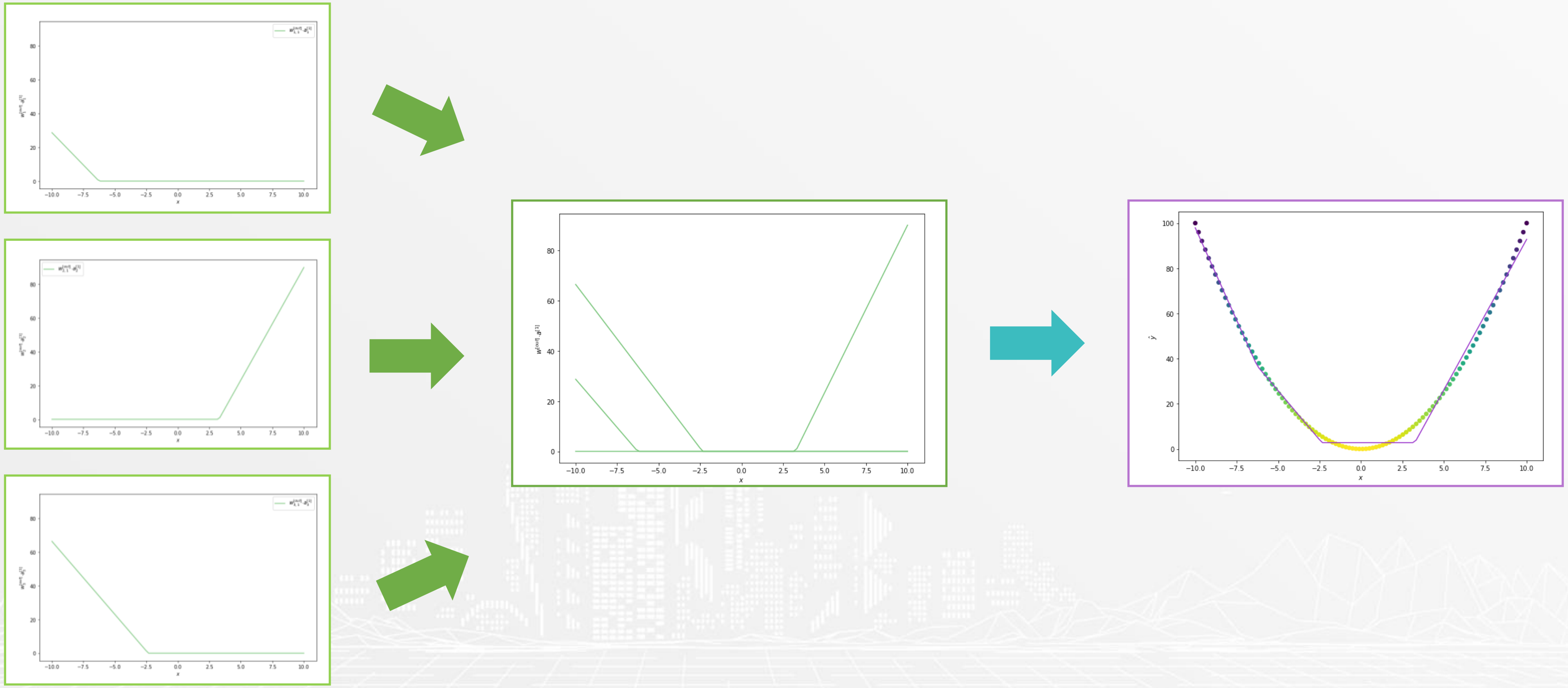
Regression | **Example 1.2**



Neural Network Interpretation



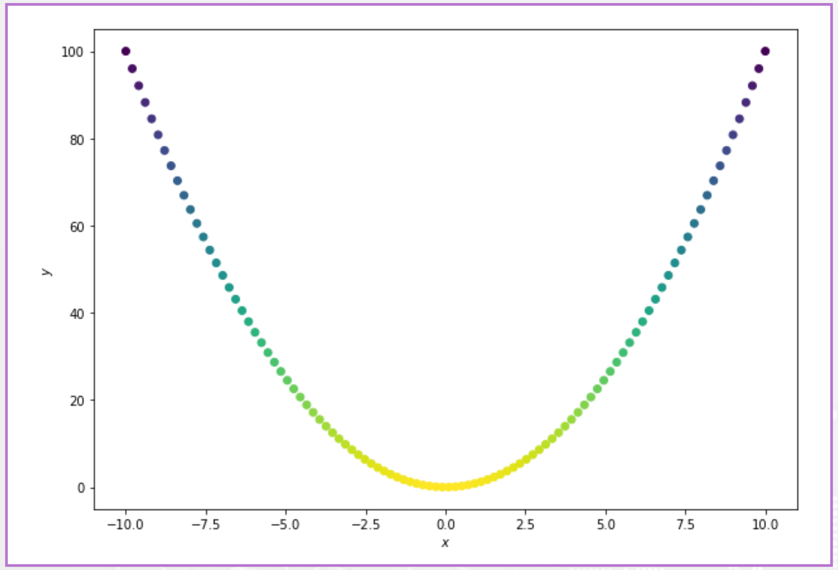
Regression | **Example 1.2**

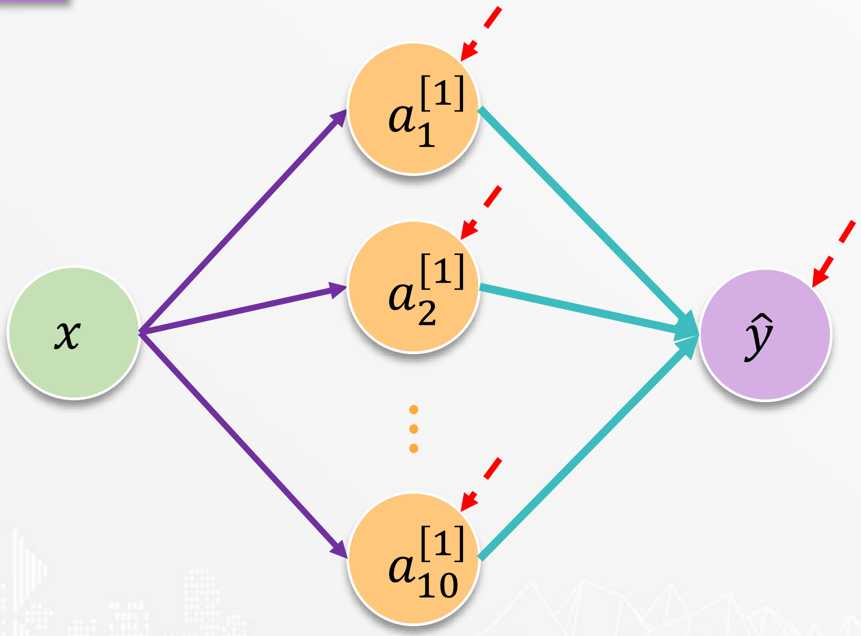


Neural Network Interpretation

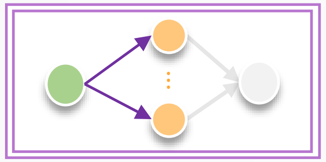
Regression

Example 1.3

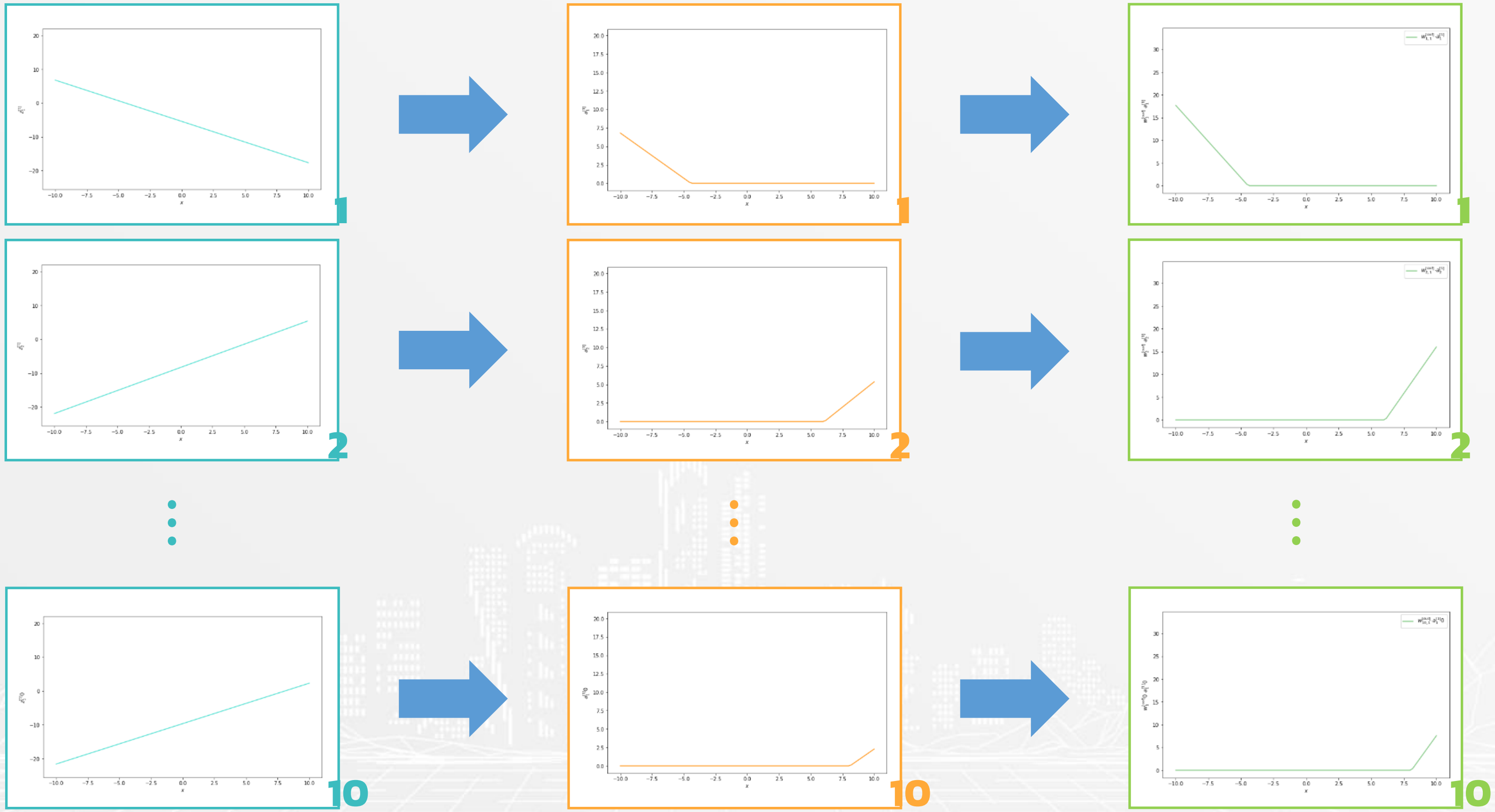




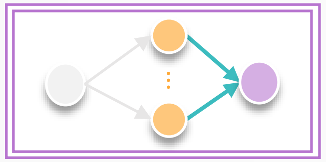
Neural Network Interpretation



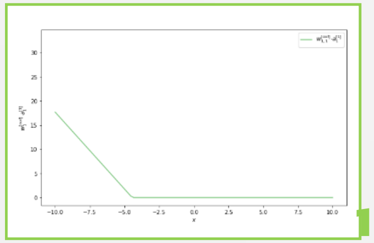
Regression | **Example 1.3**



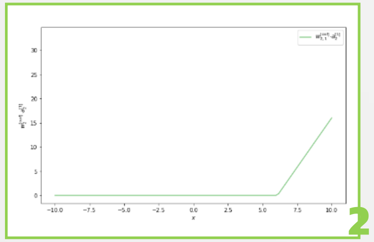
Neural Network Interpretation



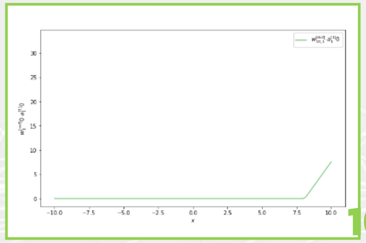
Regression | **Example 1.3**



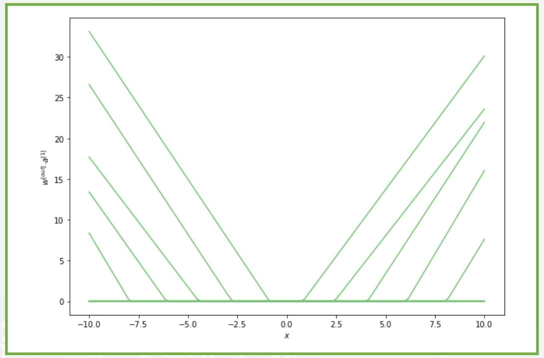
1

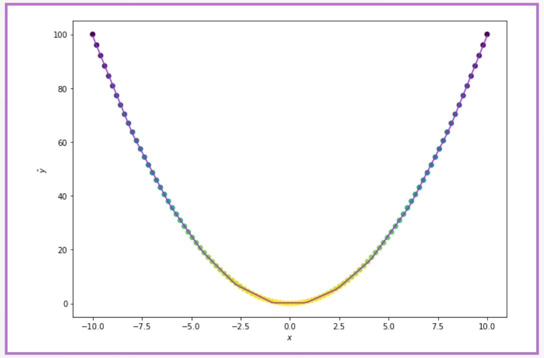


2



10

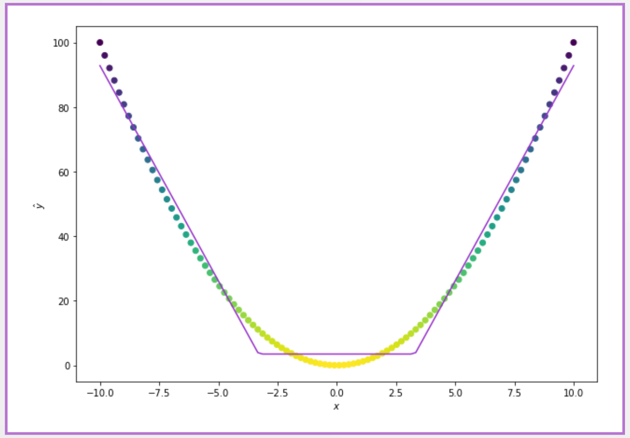




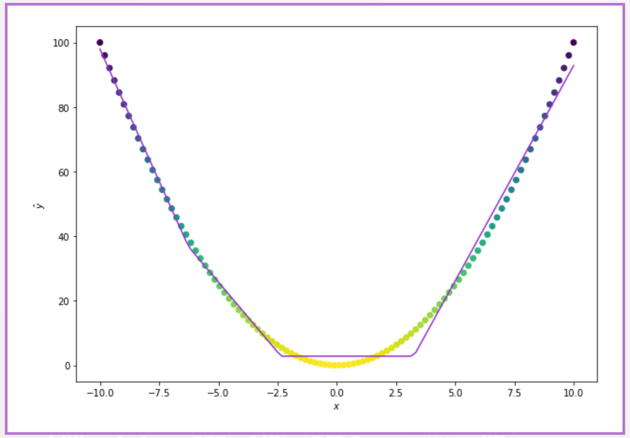
Neural Network Interpretation

Regression

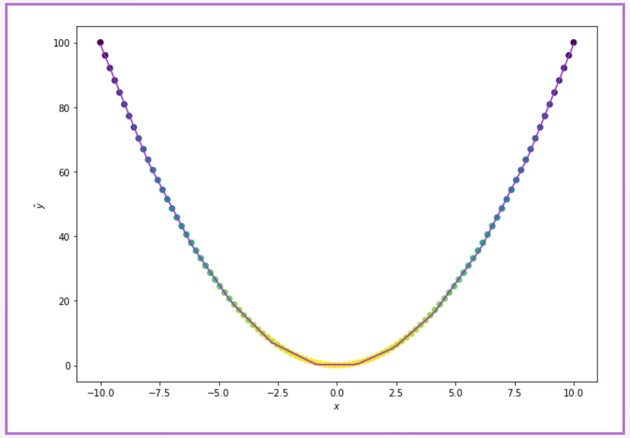
กราฟแสดงค่า predicted ของ Example 1



Example 1.1 (1,2,1)



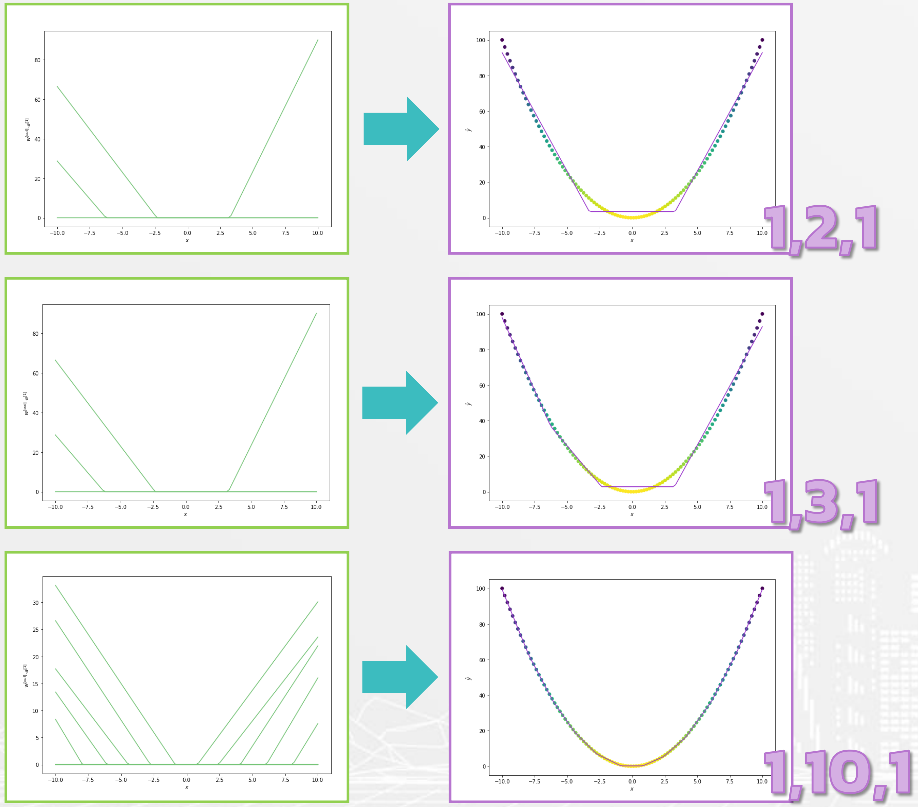
Example 1.2 (1,3,1)



Example 1.3 (1,10,1)

Neural Network Interpretation

Regression



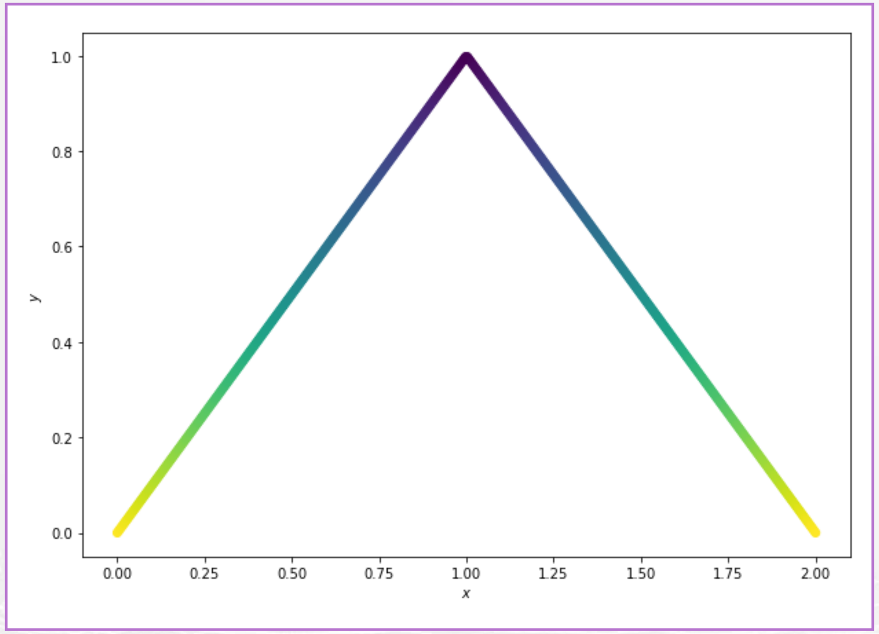
ข้อสรุปของ Example 1

- การเพิ่ม node ทำให้เกิด plane (ส่วนของเส้นตรง) ย่อยที่มากขึ้น
- plane (ส่วนของเส้นตรง) ย่อยที่มากขึ้นนั้นเกิดจากการประกอบกันของ function ย่อยหลังผ่าน activation function

Neural Network Interpretation

Regression

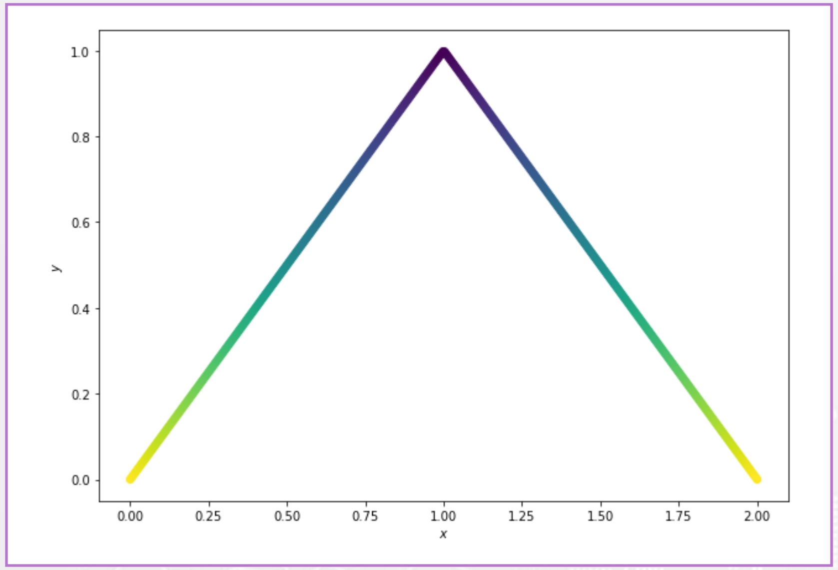
Example 2

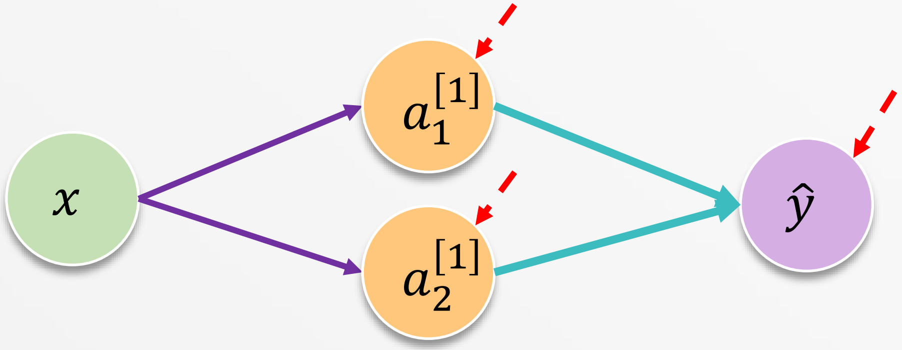


Neural Network Interpretation

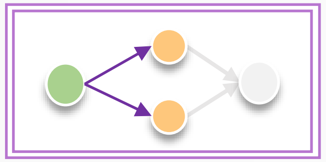
Regression

Example 2



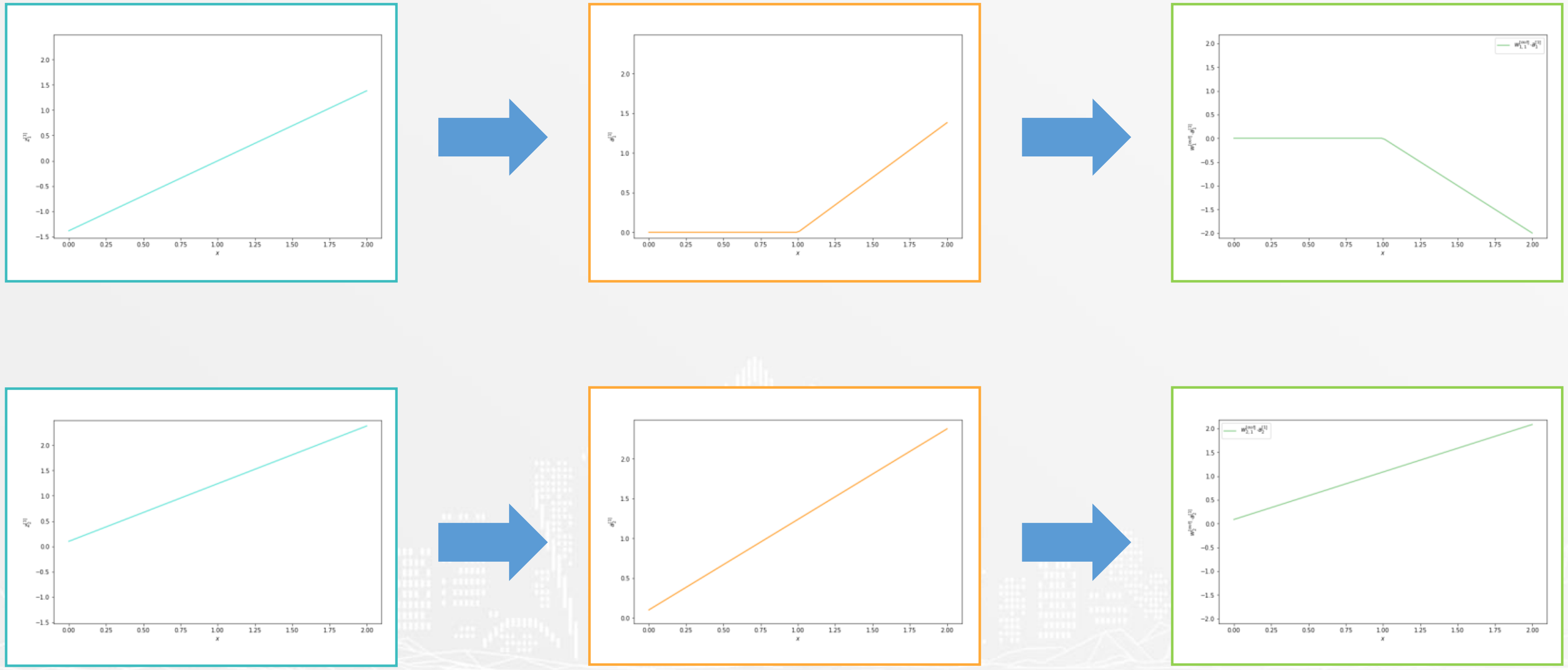


Neural Network Interpretation



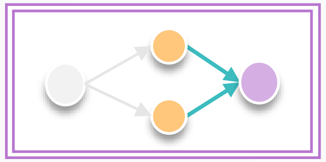
Regression

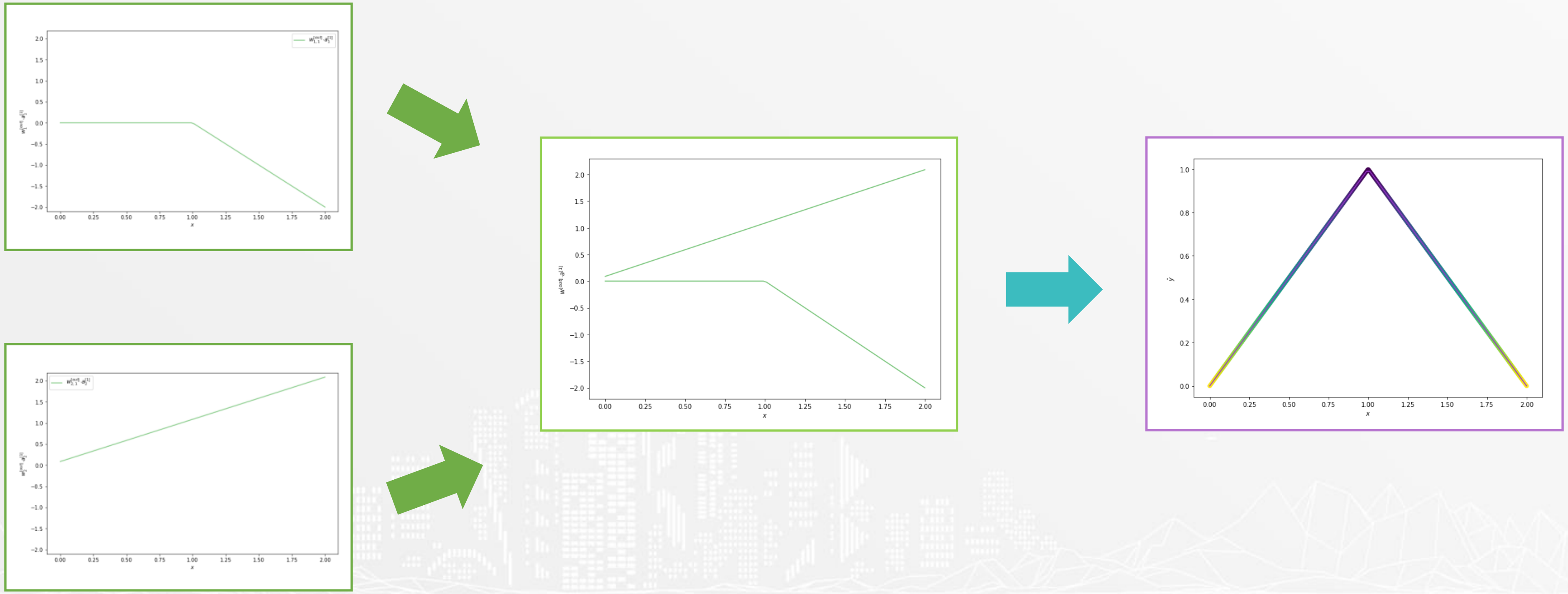
Example 2



Neural Network Interpretation

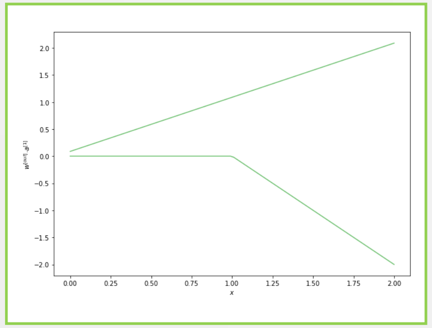
Regression | **Example 2**

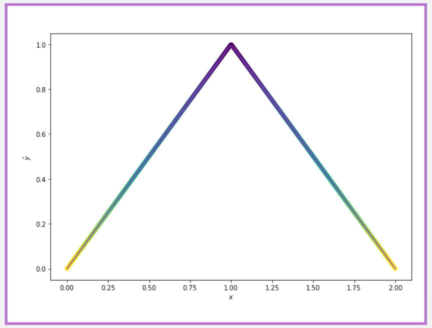




Neural Network Interpretation

Regression





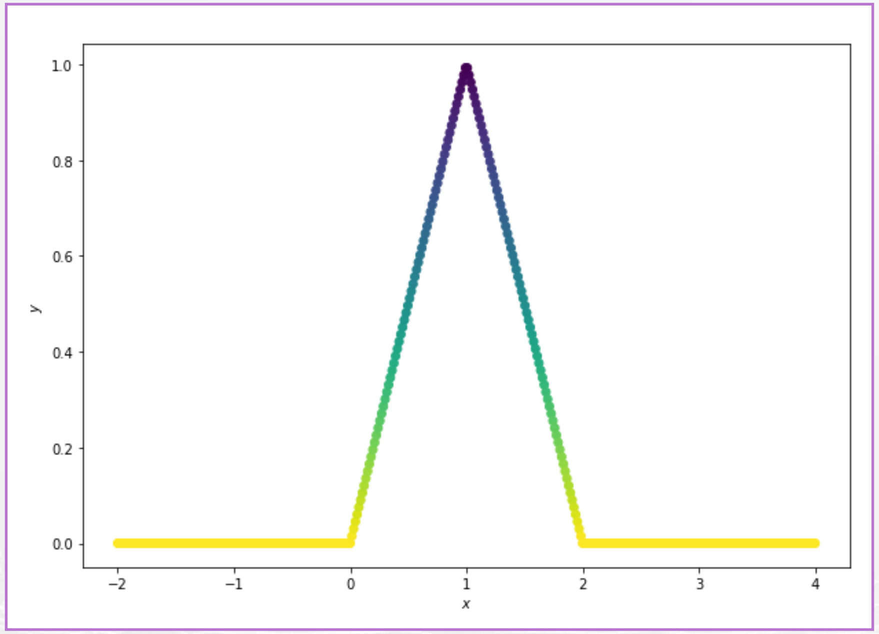
ข้อสรุปของ Example 2

- การประกอบกันของ function ย่อยหลังผ่าน activation function เกิดขึ้นทั้งการเสริมกัน และการหักล้างกัน

Neural Network Interpretation

Regression

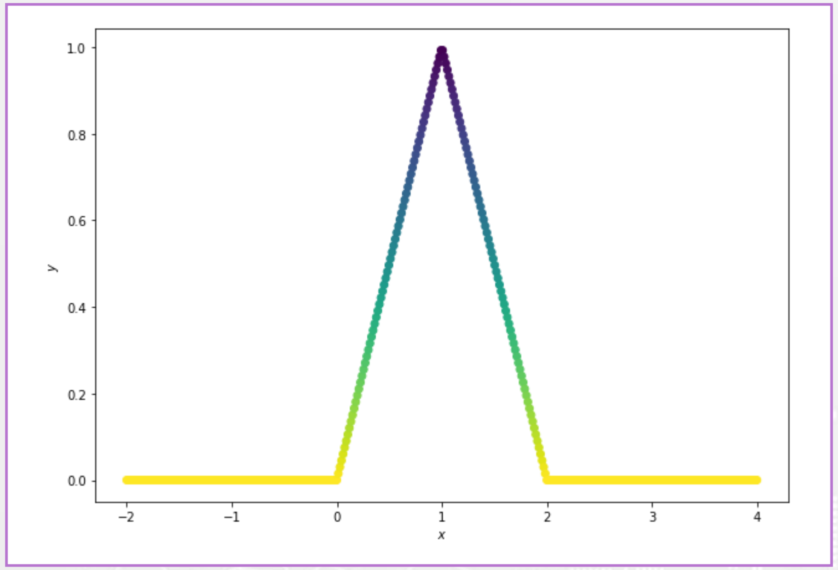
Example 3

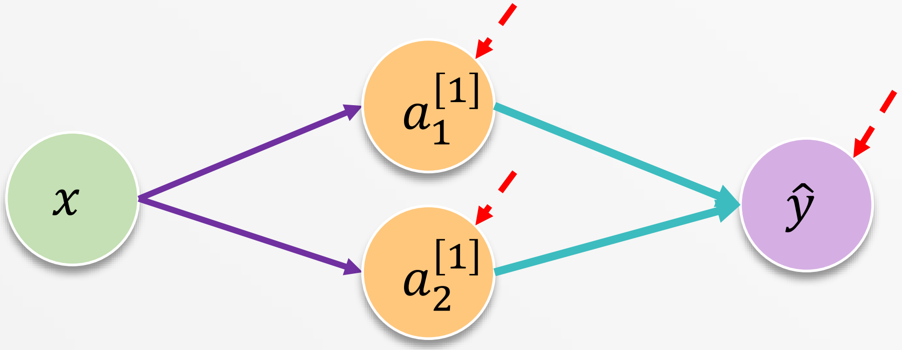


Neural Network Interpretation

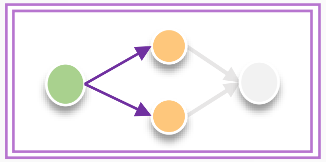
Regression

Example 3.1

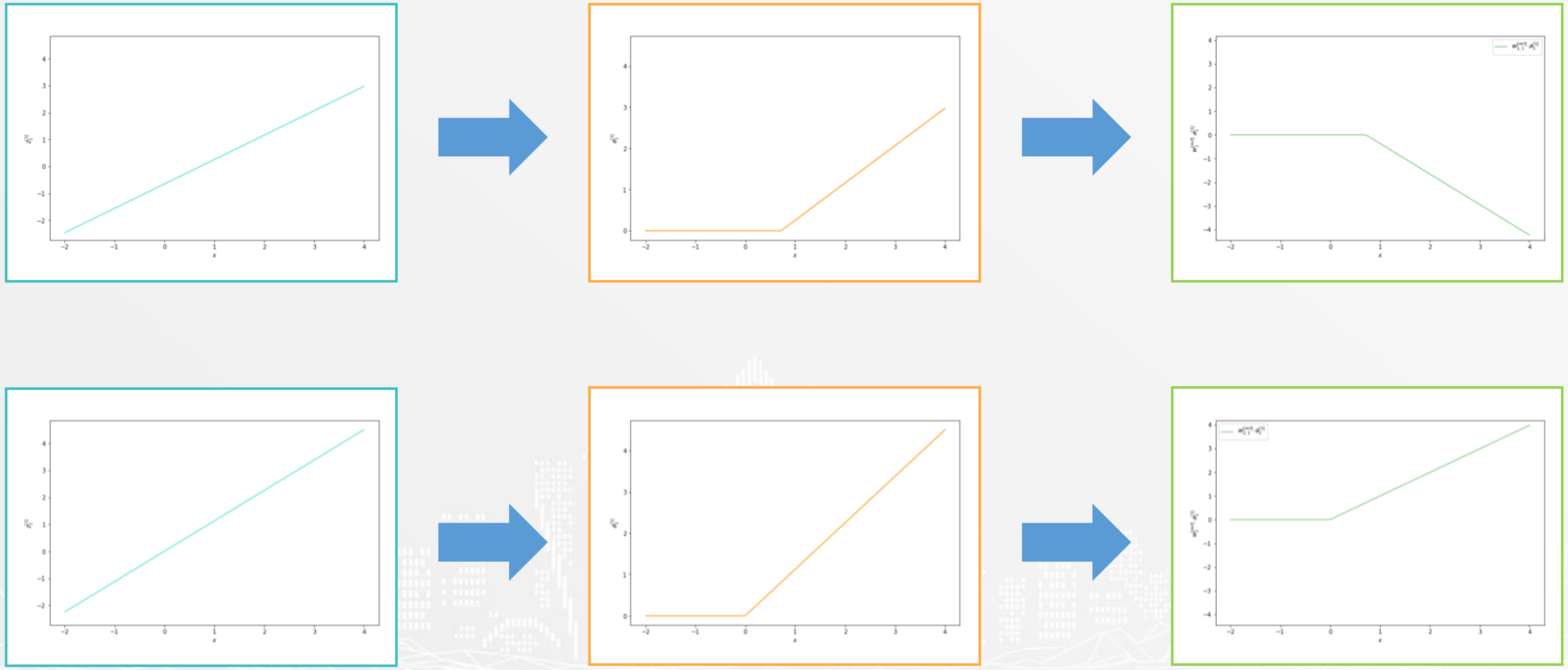




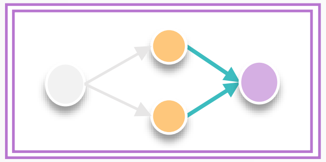
Neural Network Interpretation



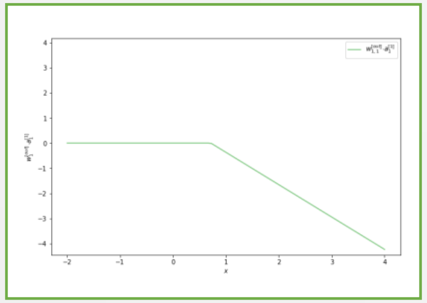
Regression | **Example 3.1**

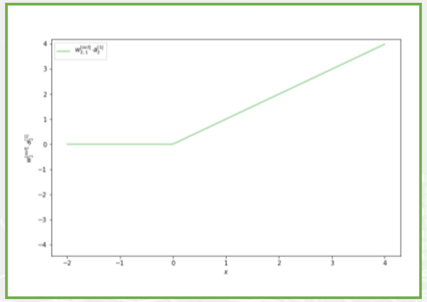


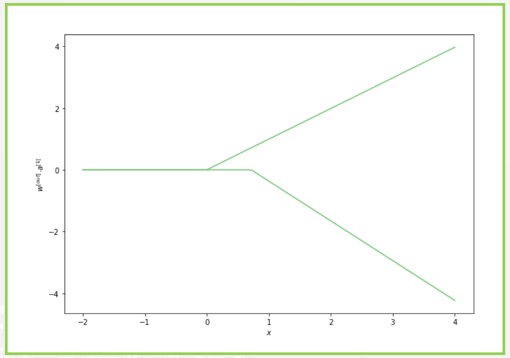
Neural Network Interpretation

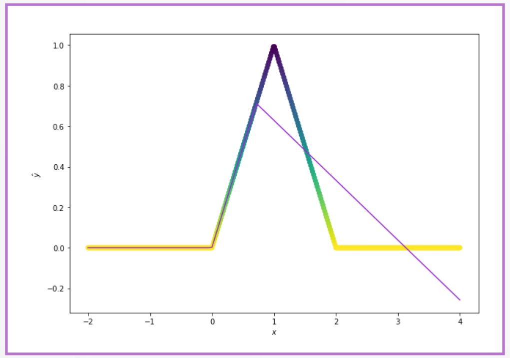


Regression | **Example 3.1**





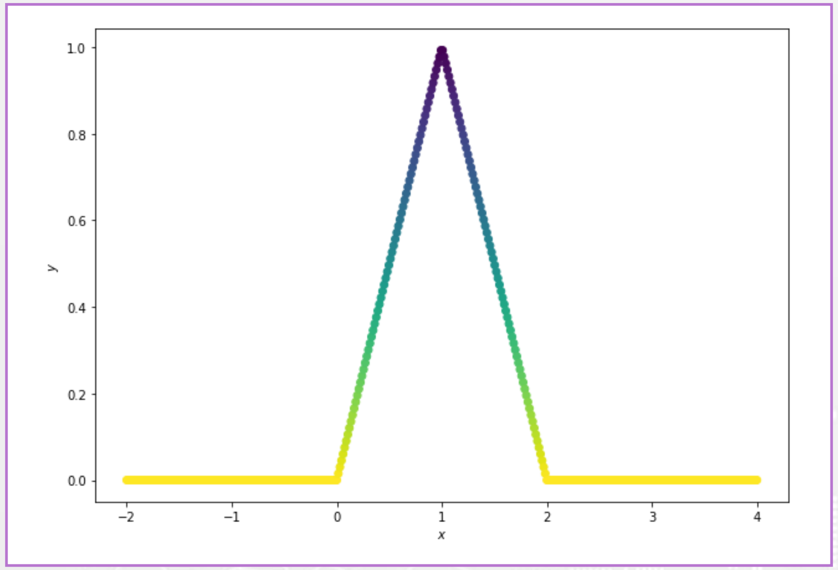


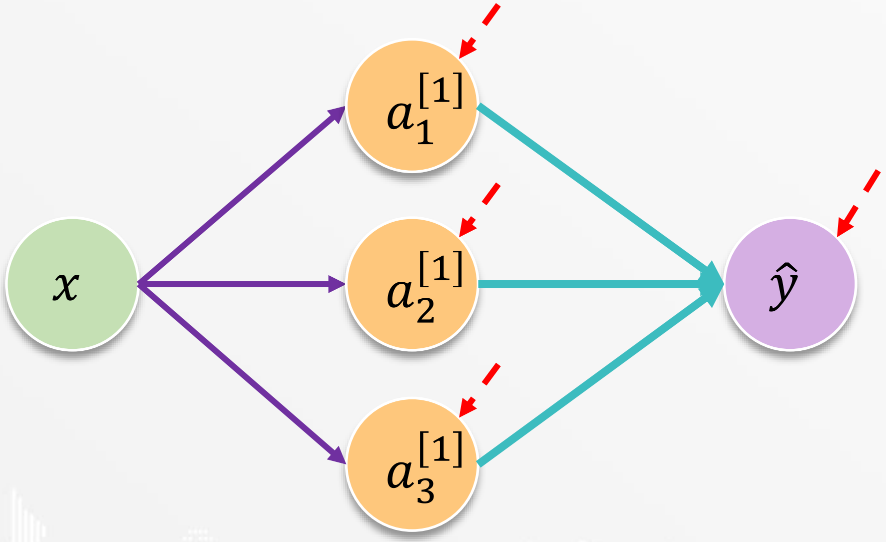


Neural Network Interpretation

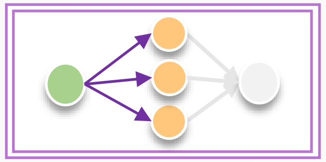
Regression

Example 3.2

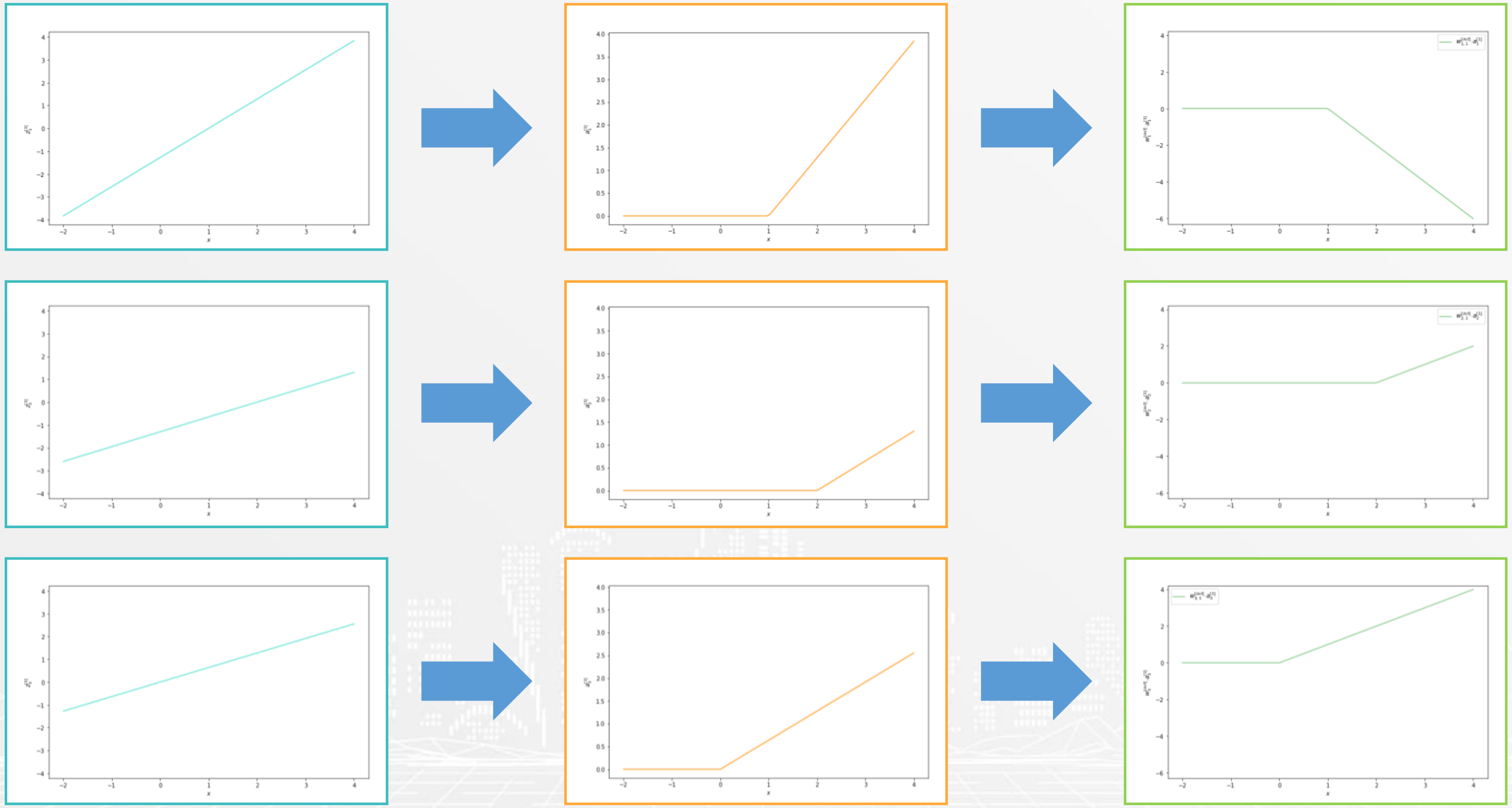




Neural Network Interpretation



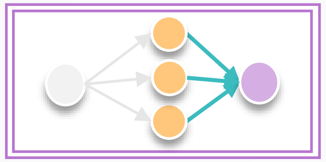
Regression | **Example 3.2**

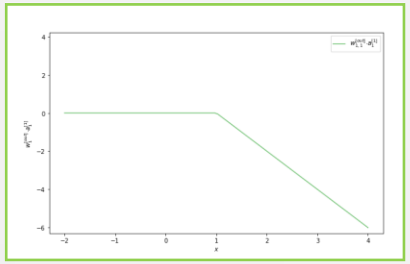


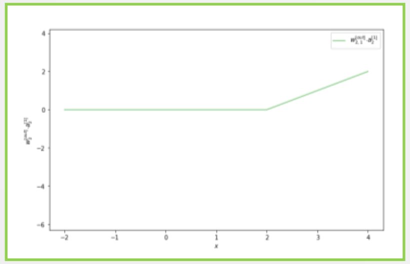
Neural Network Interpretation

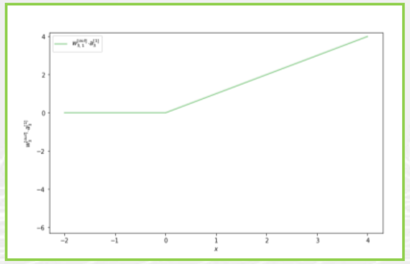
Regression

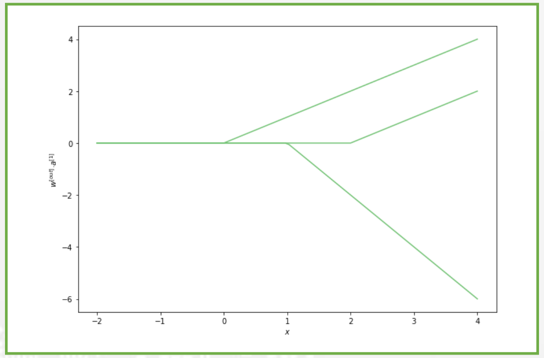
Example 3.2

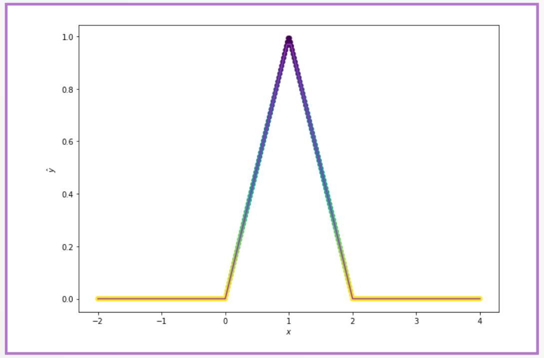








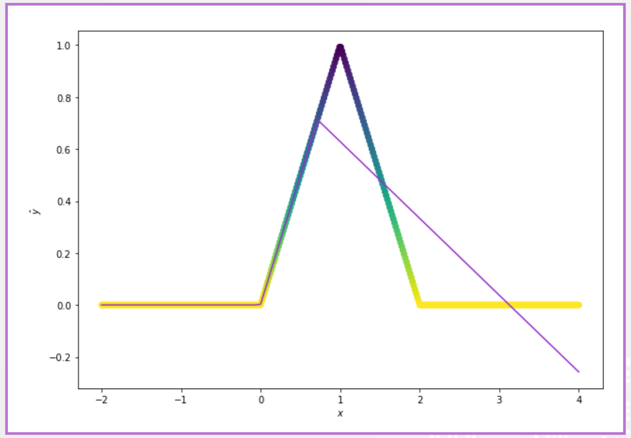




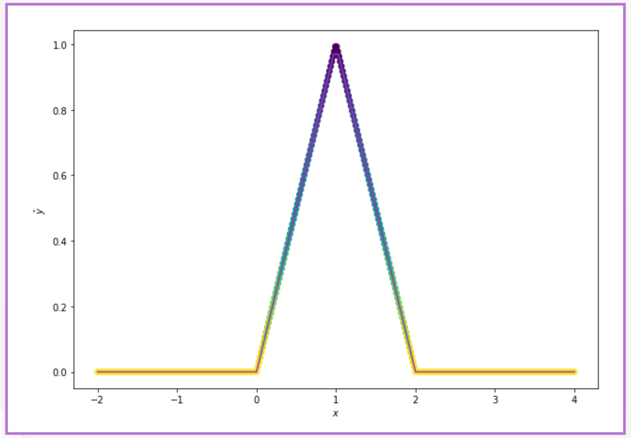
Neural Network Interpretation

Regression

กราฟแสดงค่า predicted ของ Example 3



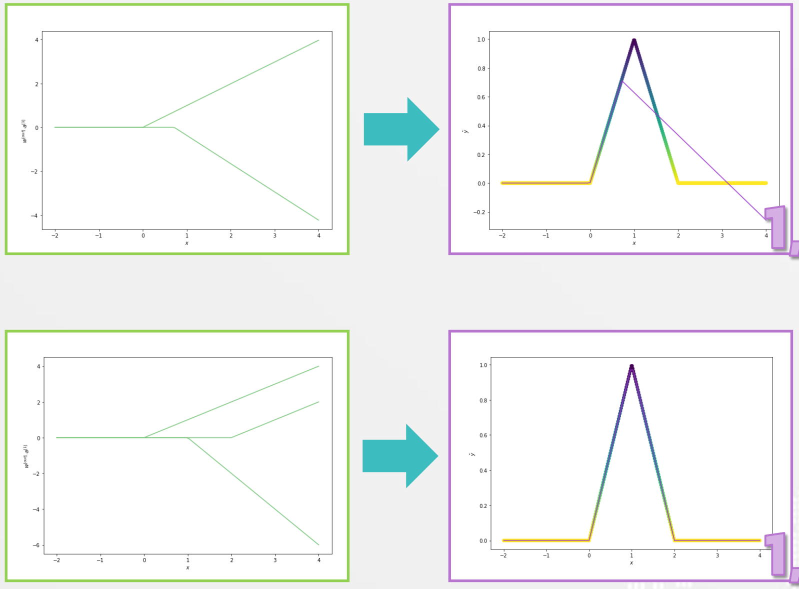
Example 3.1 (1,2,1)



Example 3.2 (1,3,1)

Neural Network Interpretation

Regression



1,2,1

1,3,1

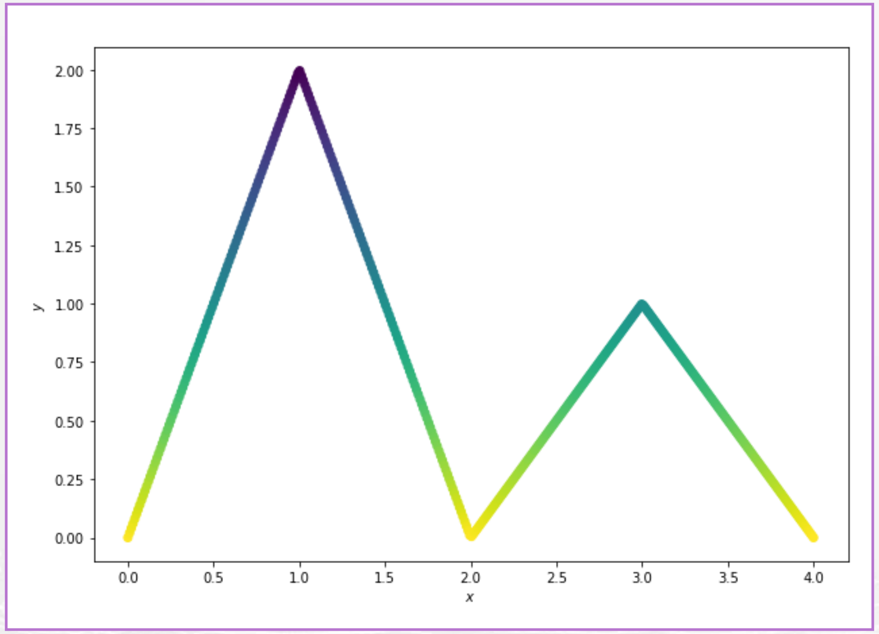
ข้อสรุปของ Example 3

- ผลของการการประกอบกันของ function ย่อยหลังผ่าน activation function นั้นจะยังหลงเหลือต่อไปในตลอดทั้งช่วงของ space

Neural Network Interpretation

Regression

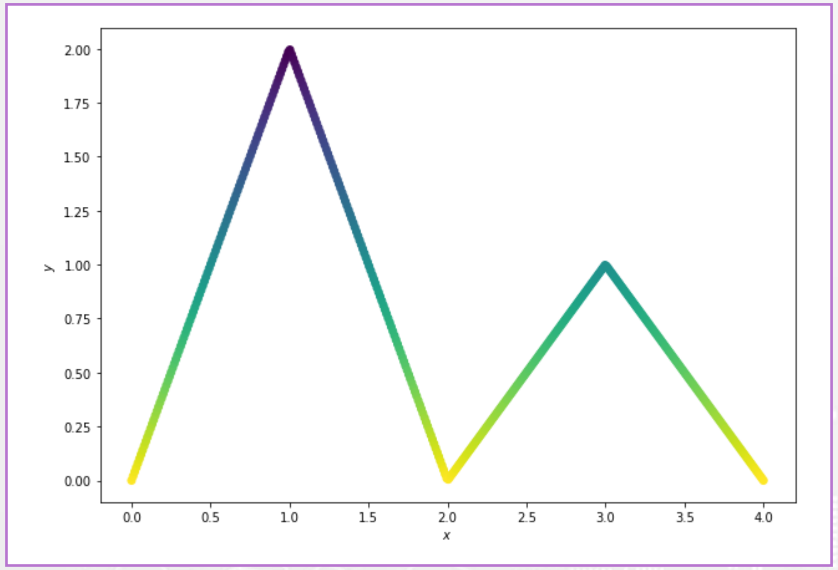
Example 4

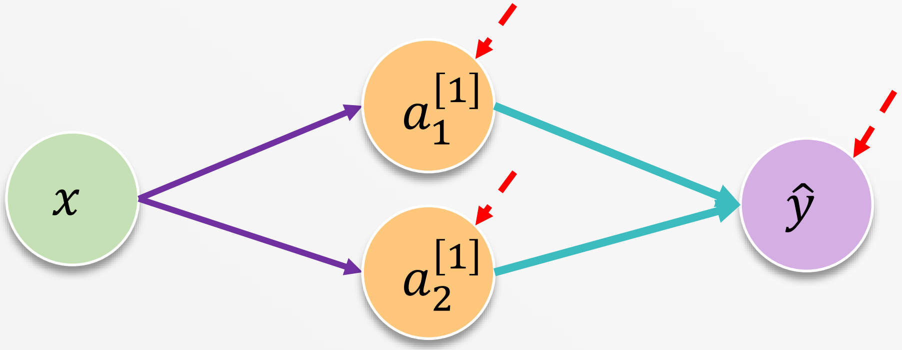


Neural Network Interpretation

Regression

Example 4.1

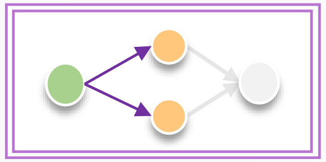


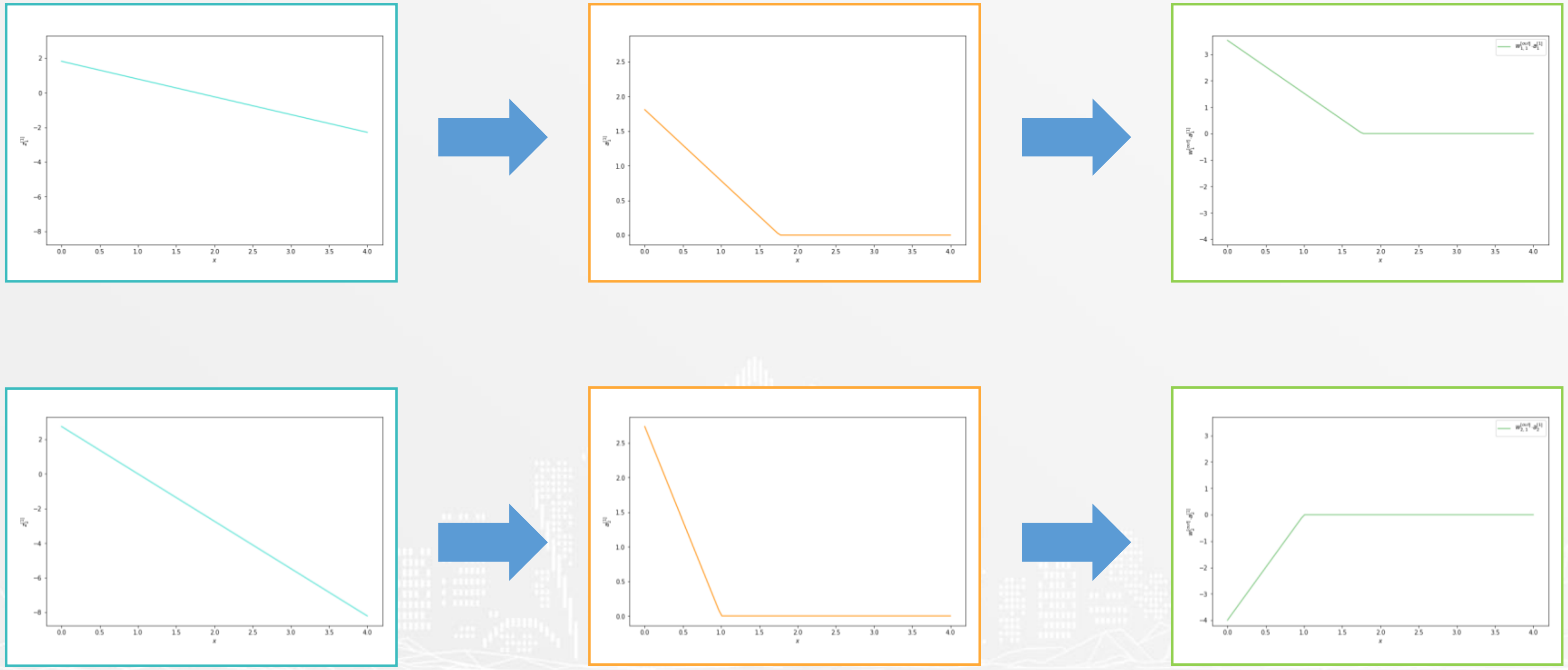


Neural Network Interpretation

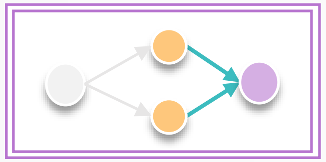
Regression

Example 4.1

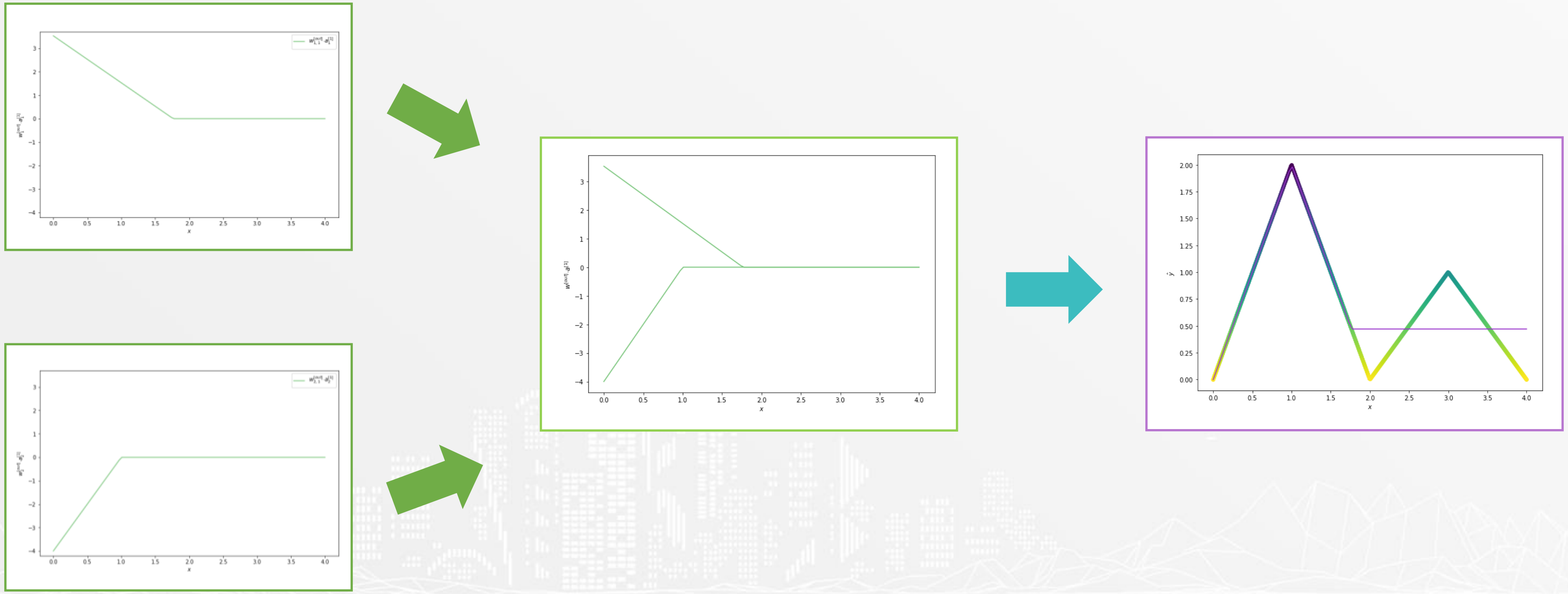




Neural Network Interpretation



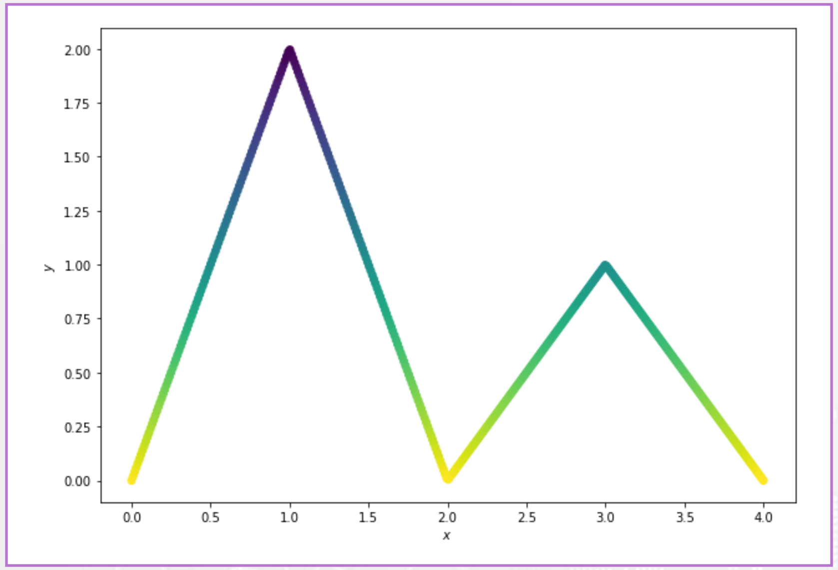
Regression | **Example 4.1**

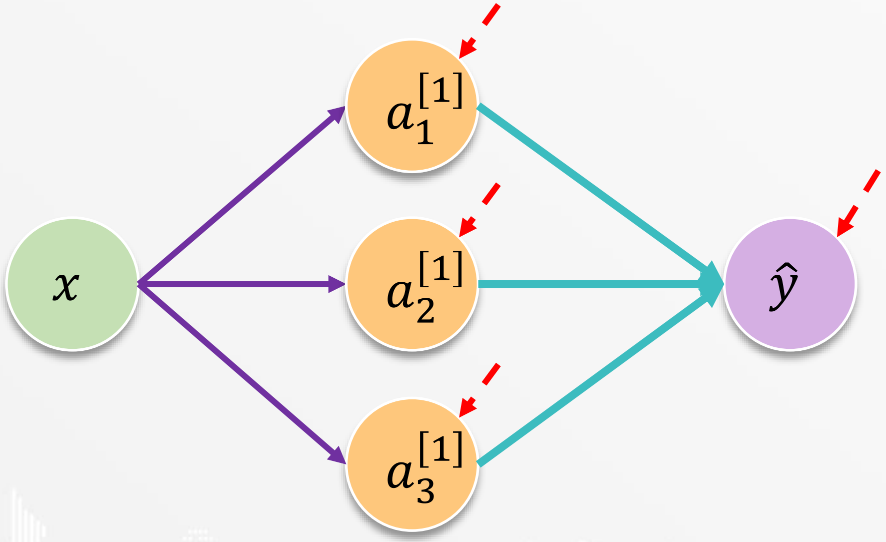


Neural Network Interpretation

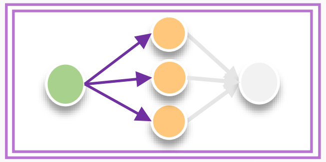
Regression

Example 4.2

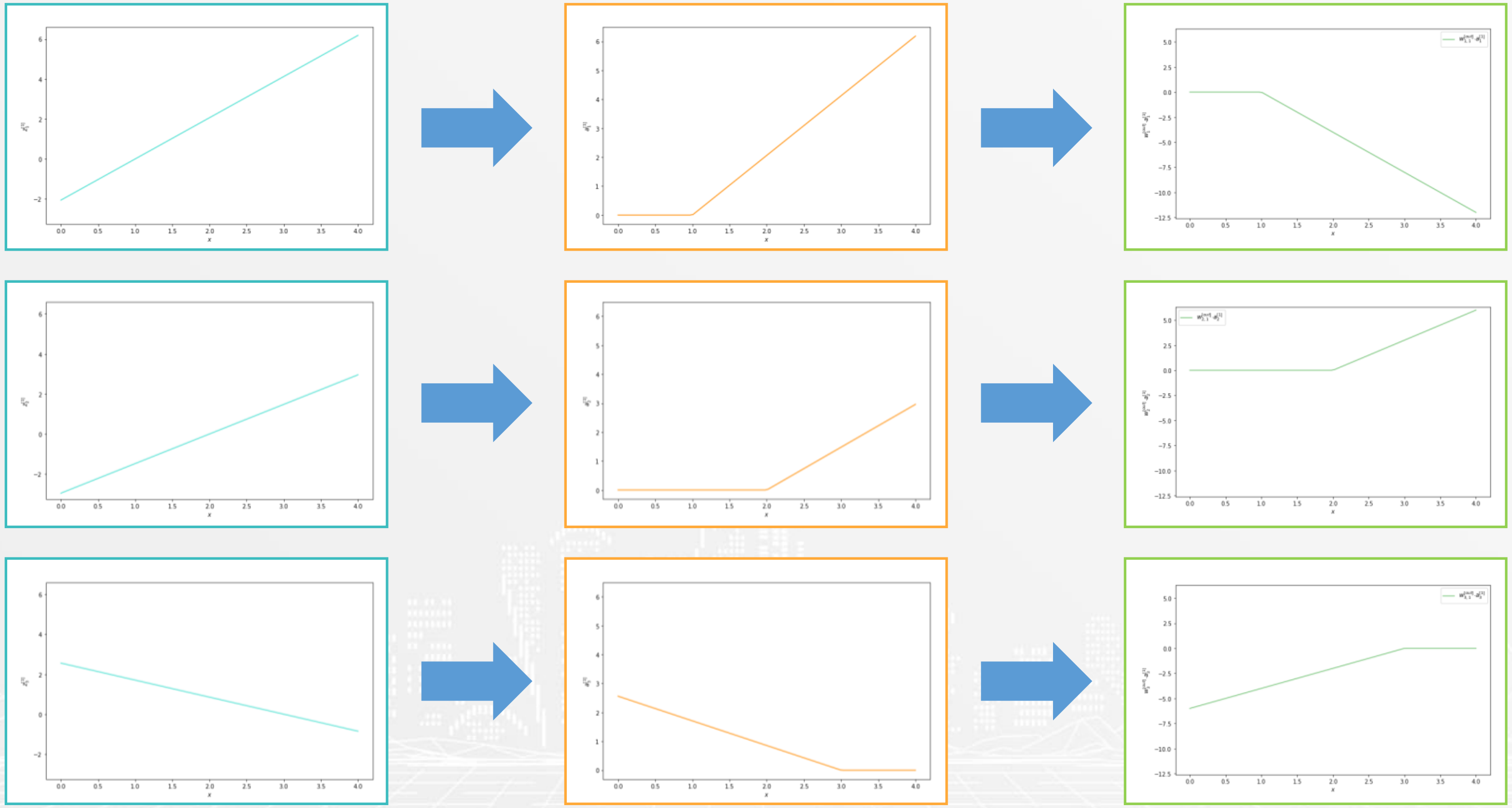




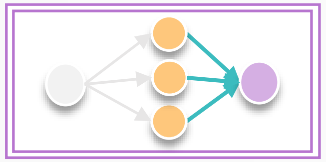
Neural Network Interpretation



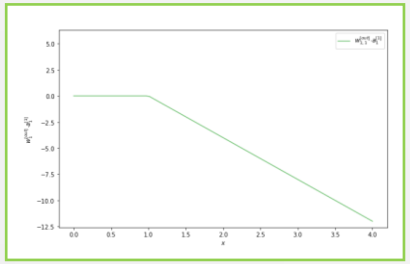
Regression | **Example 4.2**

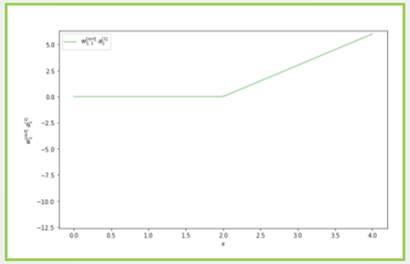


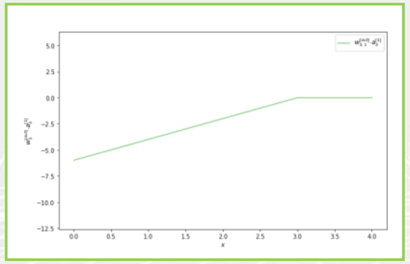
Neural Network Interpretation

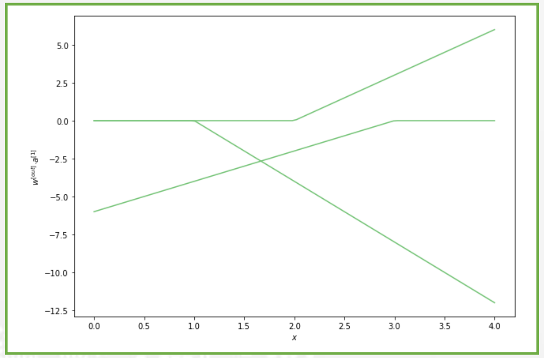


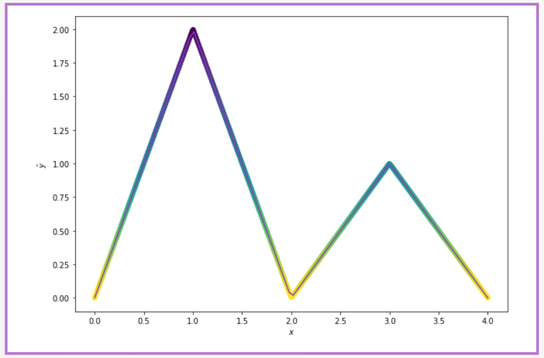
Regression | **Example 4.2**







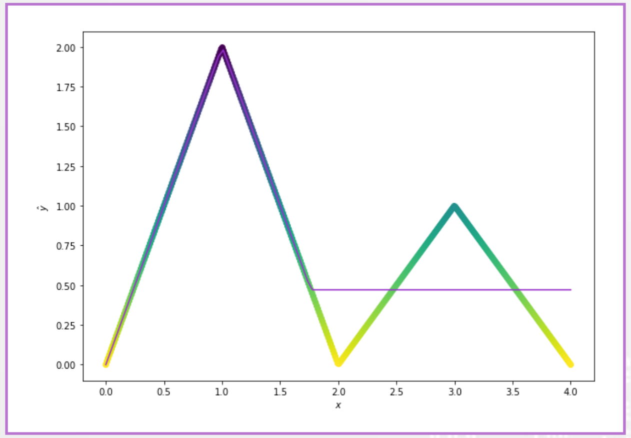




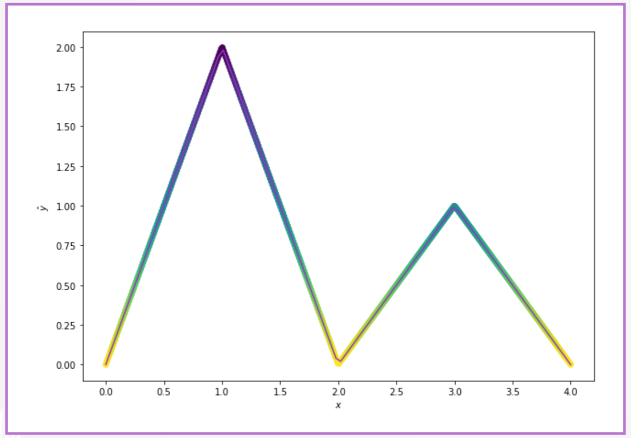
Neural Network Interpretation

Regression

กราฟแสดงค่า predicted ของ Example 4



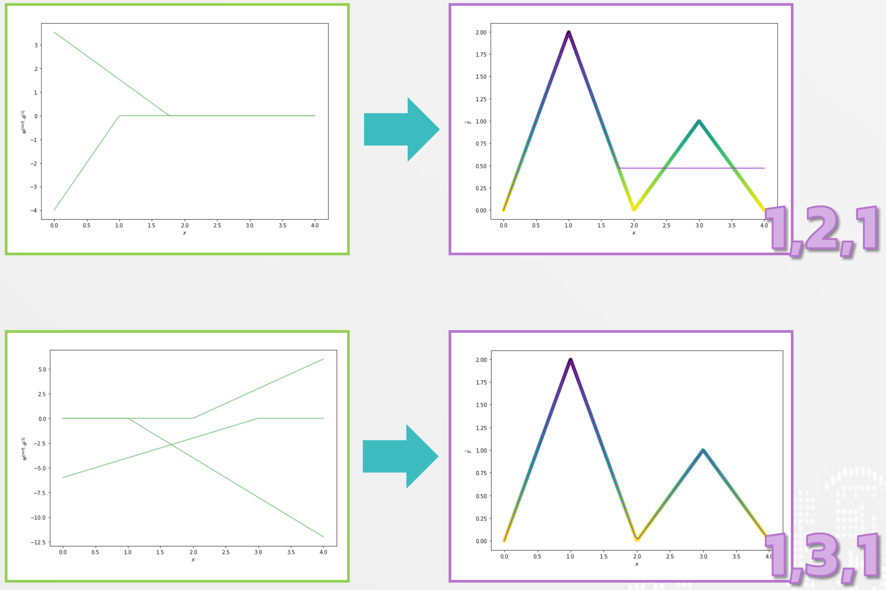
Example 4.1 (1,2,1)



Example 4.2 (1,3,1)

Neural Network Interpretation

Regression



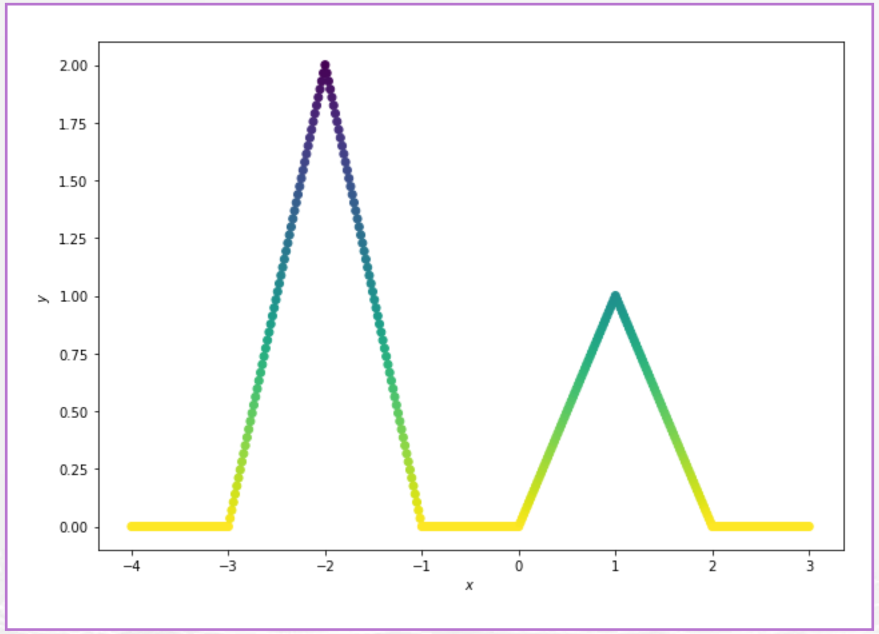
ข้อสรุปของ Example 4

- จำนวน plane (ส่วนของเส้นตรง) สูงสุดที่เกิดขึ้นในชั้น output ใน 2 มิติ (จำนวนตัวแปรต้น 1 ตัว) นั้นมีจำนวนเท่ากับ จำนวน node ใน hidden layer + 1

Neural Network Interpretation

Regression

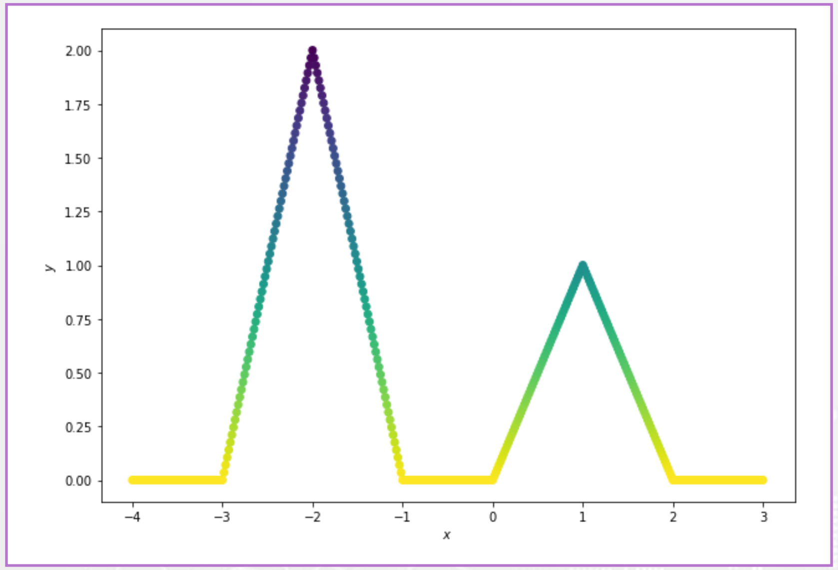
Example 5

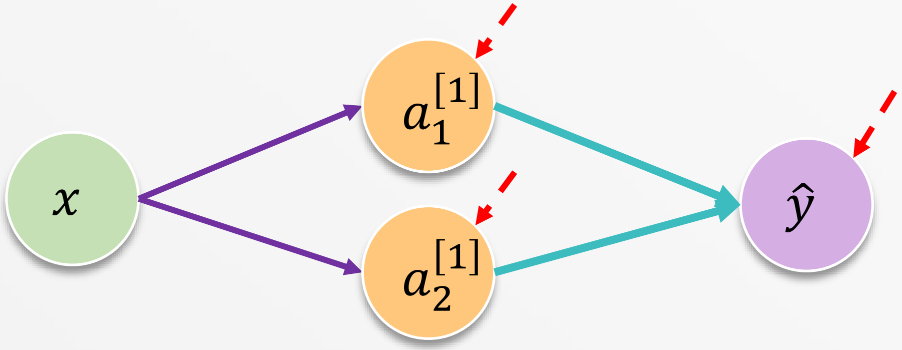


Neural Network Interpretation

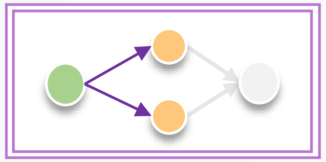
Regression

Example 5.1

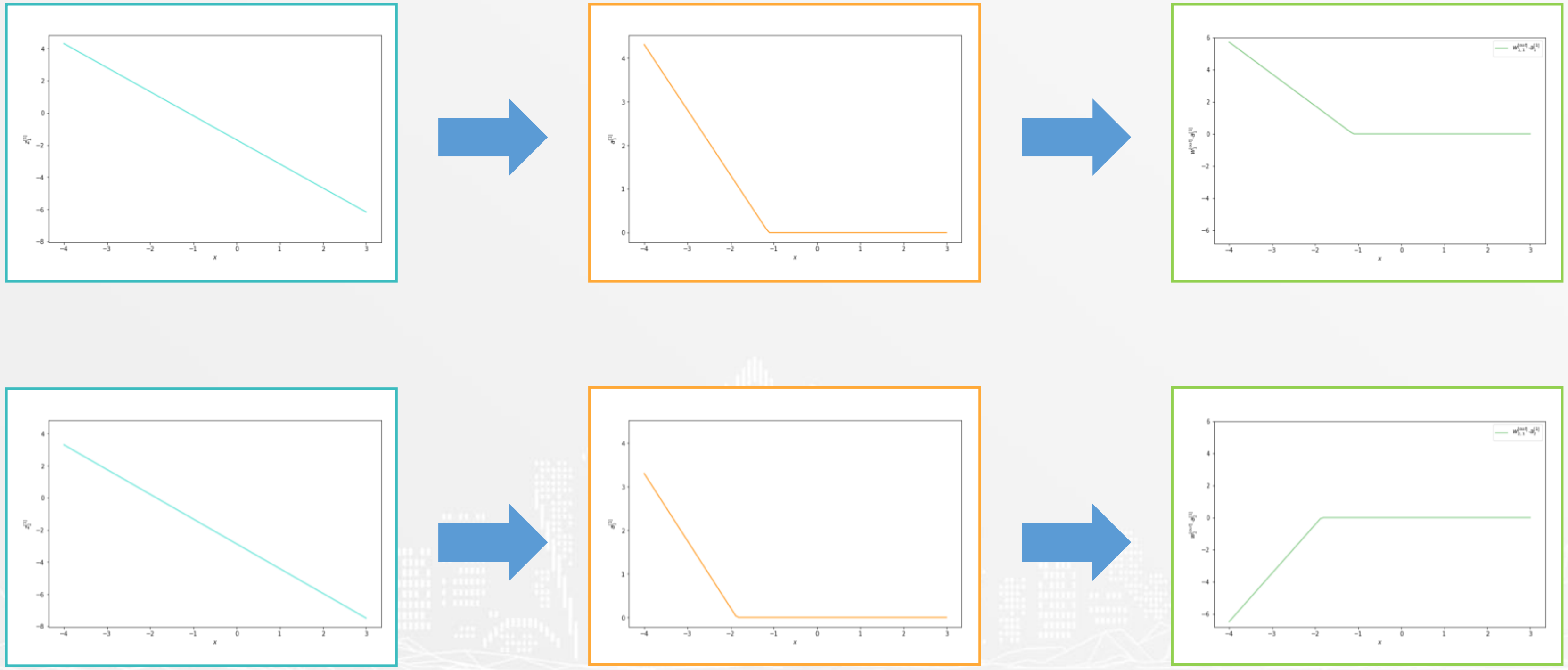




Neural Network Interpretation

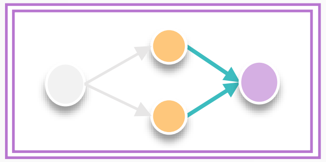


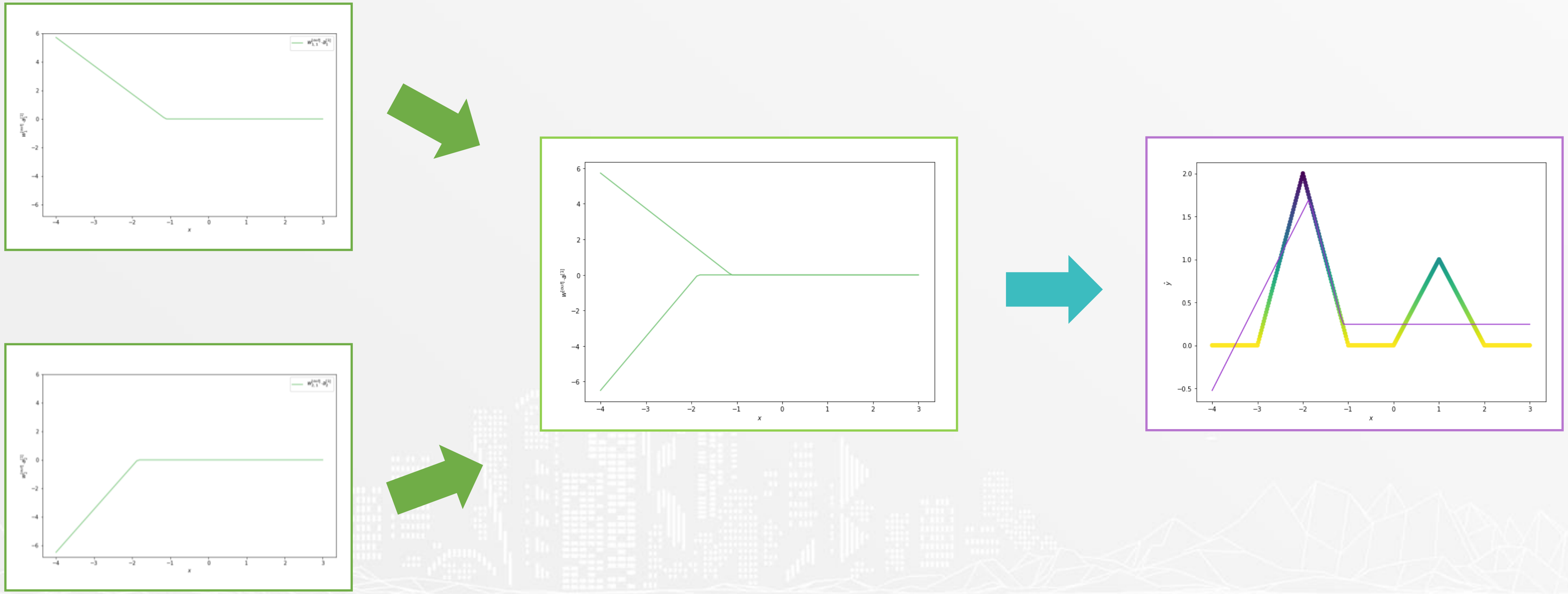
Regression | **Example 5.1**



Neural Network Interpretation

Regression | **Example 5.1**

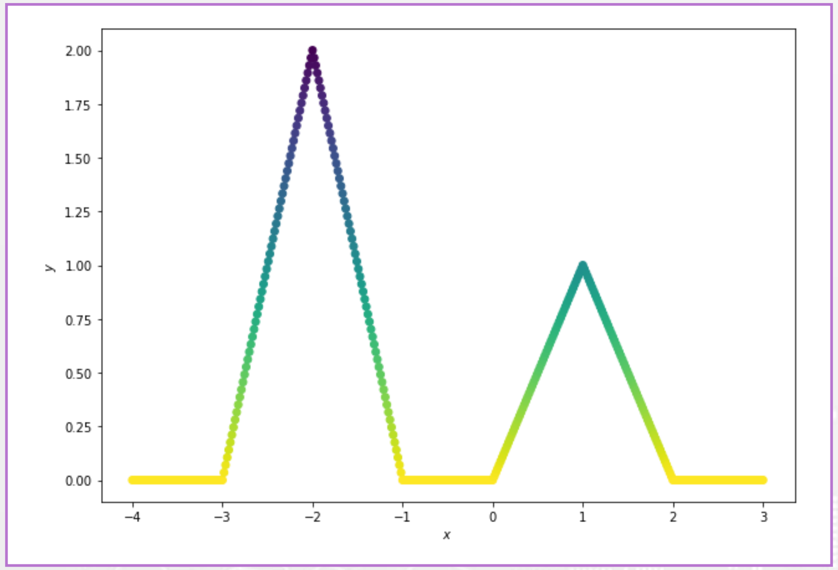


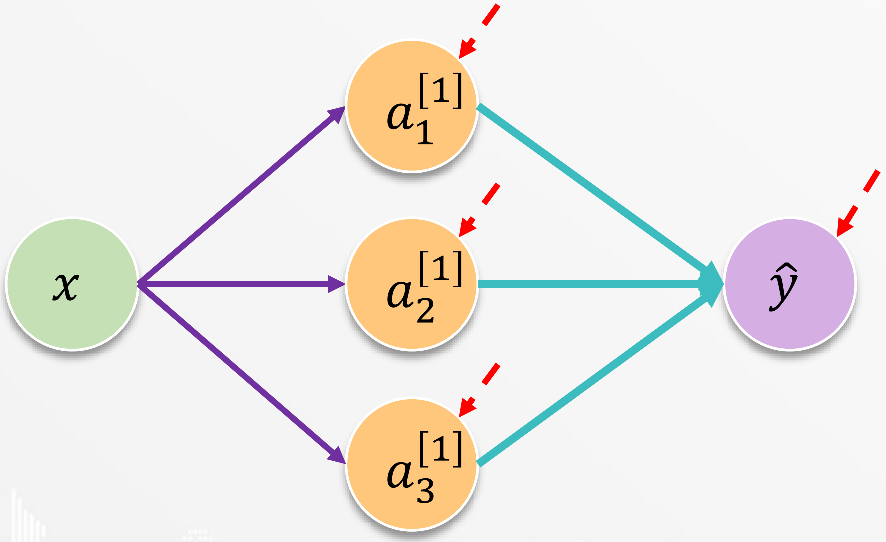


Neural Network Interpretation

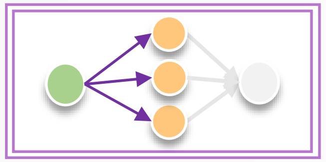
Regression

Example 5.2

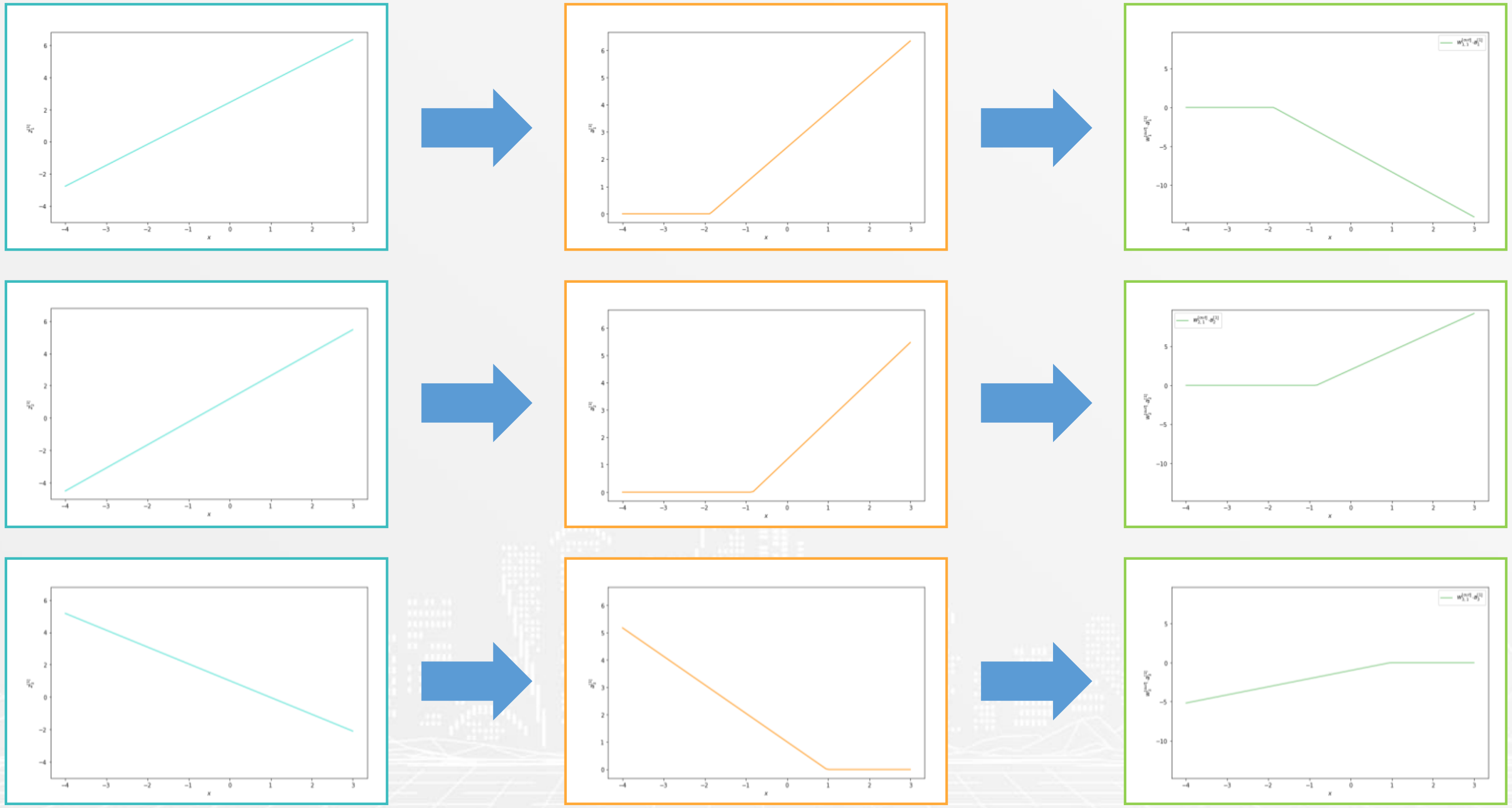




Neural Network Interpretation

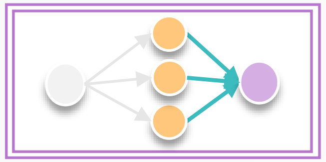


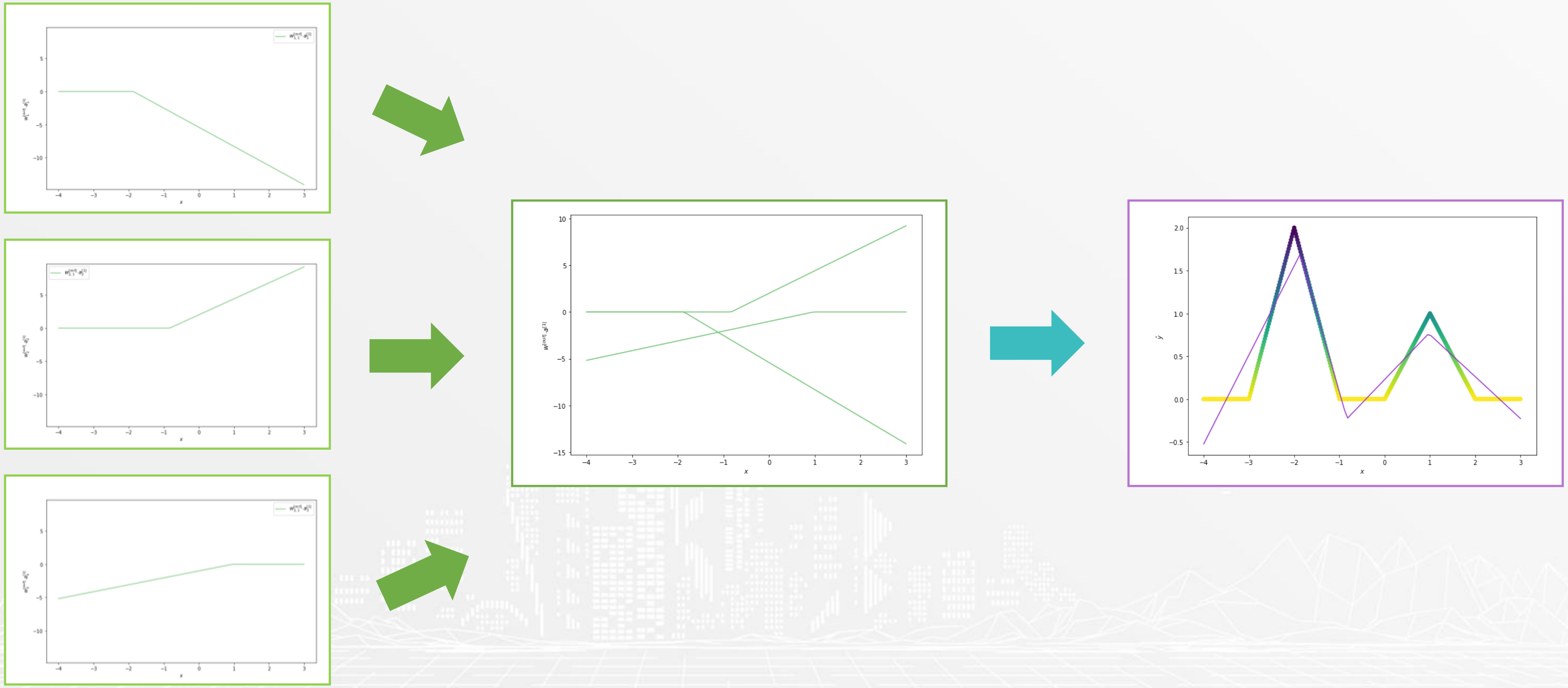
Regression | **Example 5.2**



Neural Network Interpretation

Regression **Example 5.2**

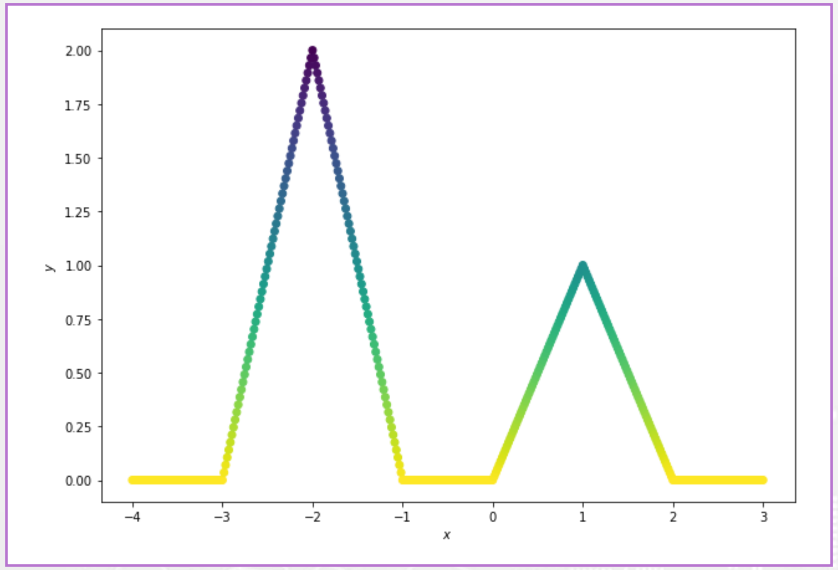


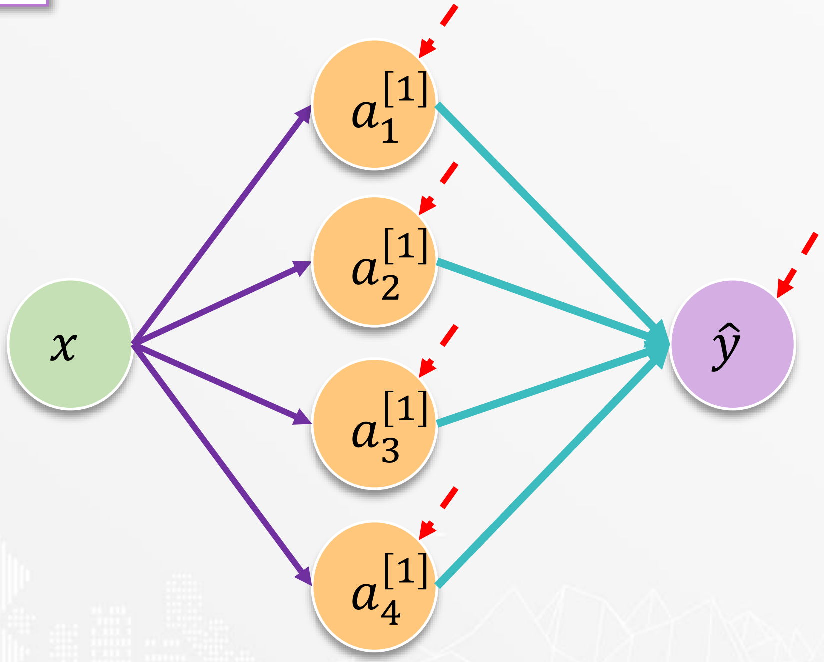


Neural Network Interpretation

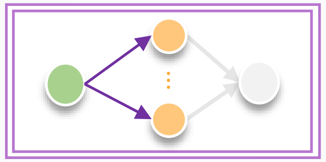
Regression

Example 5.3

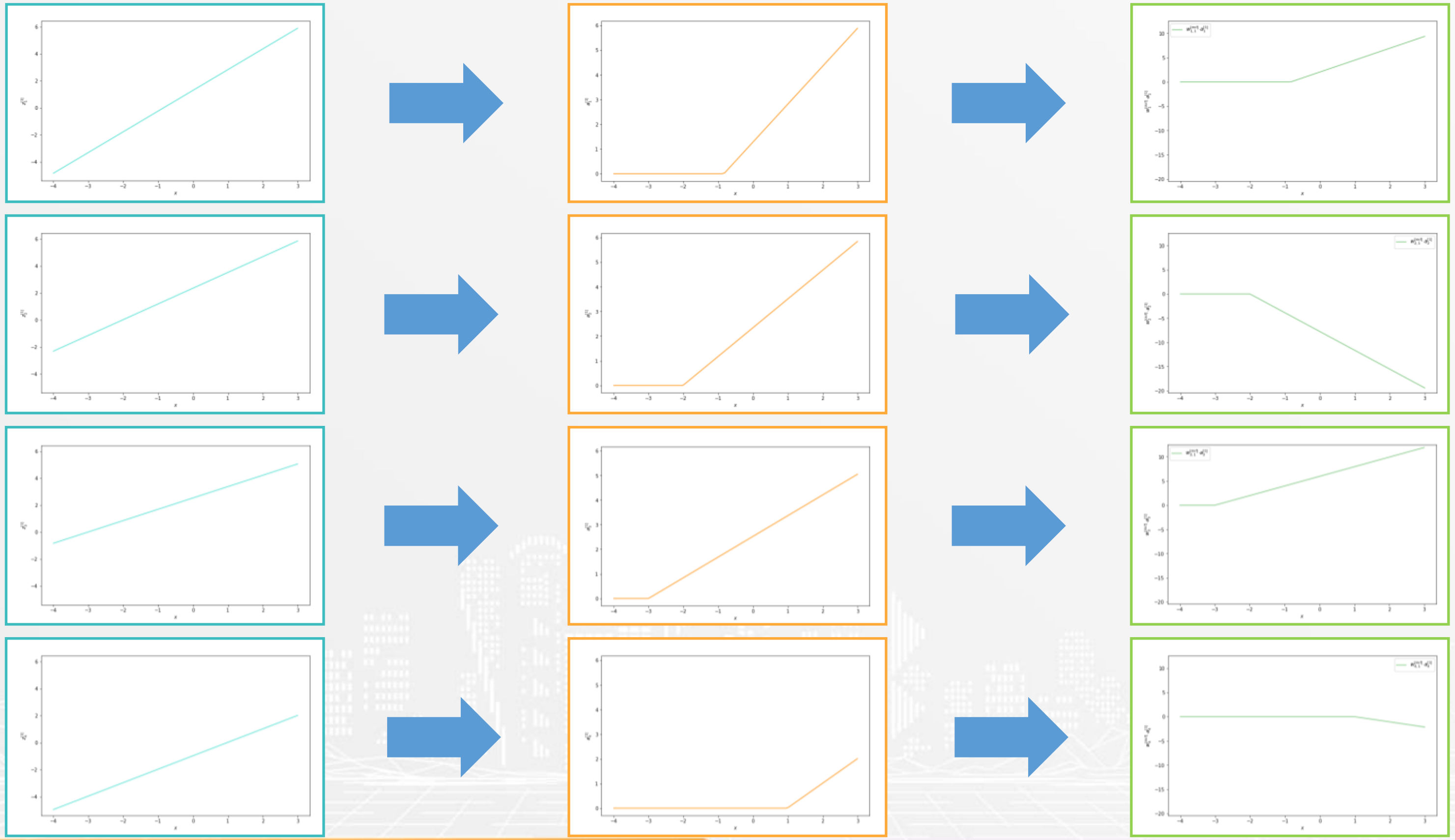




Neural Network Interpretation

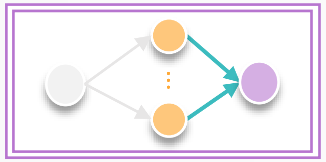


Regression | **Example 5.3**

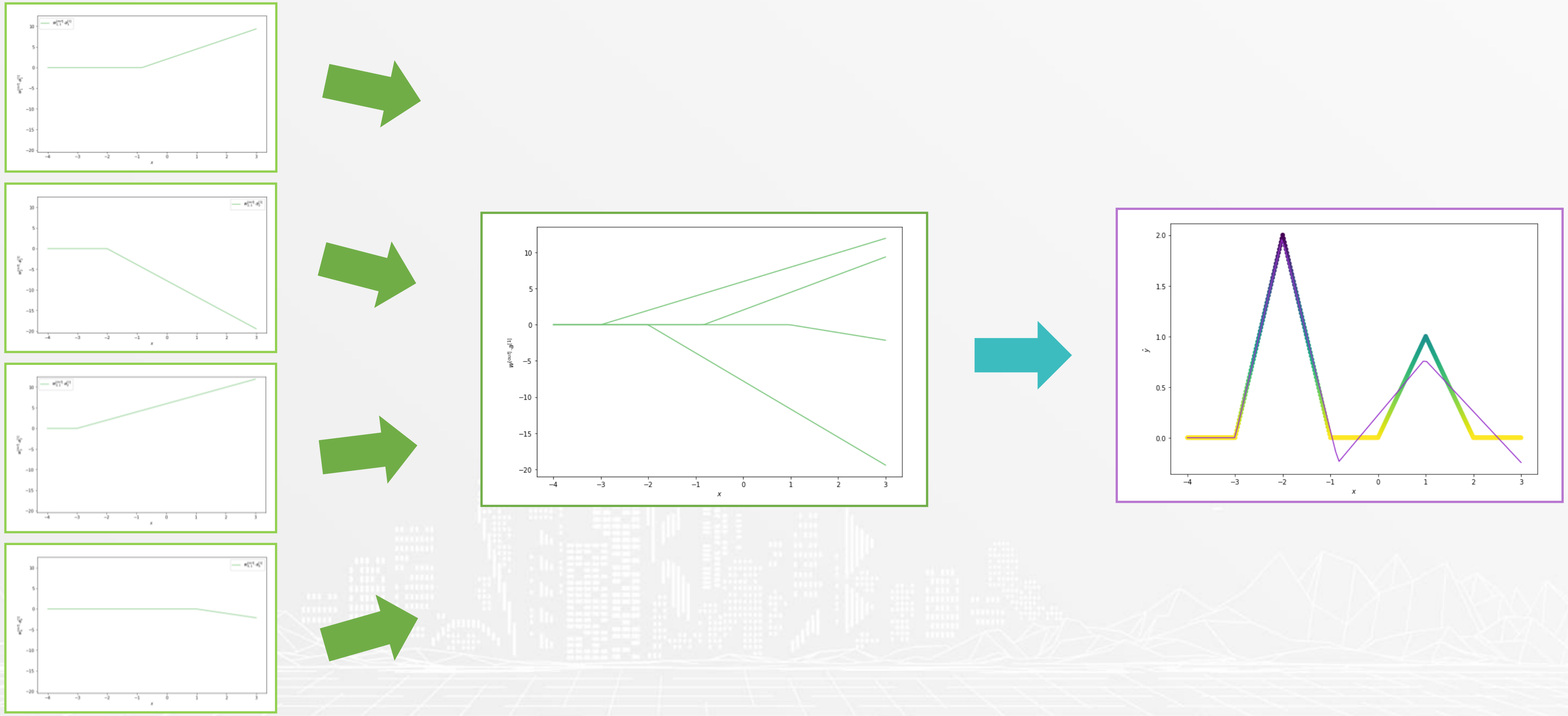


TAUTOLOGY

Neural Network Interpretation



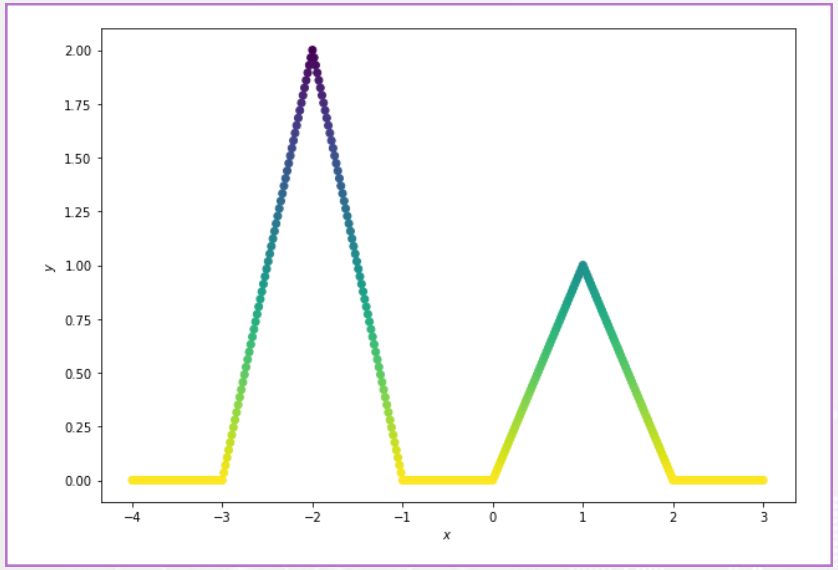
Regression | **Example 5.3**

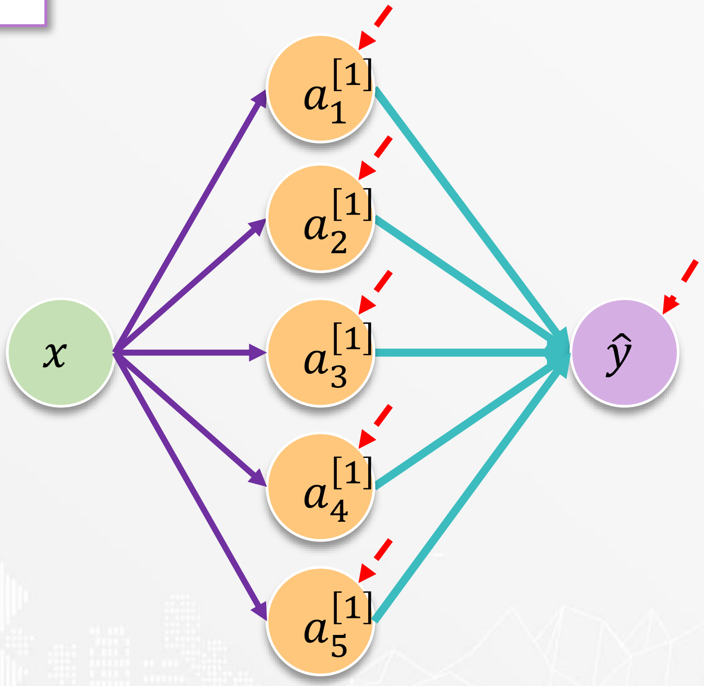


Neural Network Interpretation

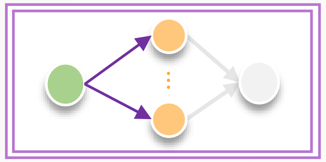
Regression

Example 5.4

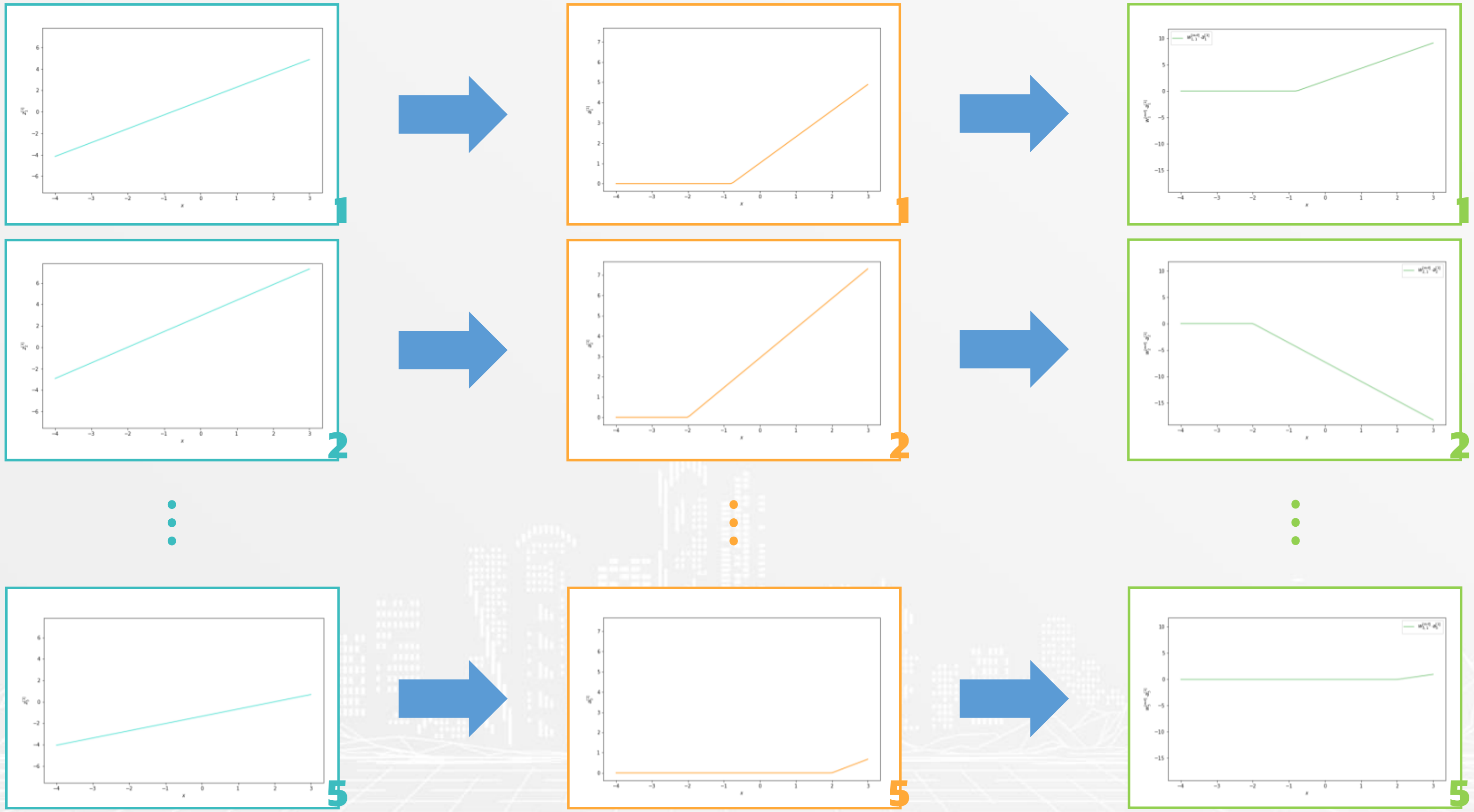




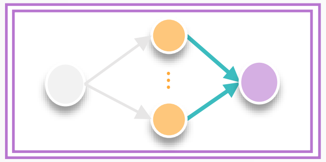
Neural Network Interpretation



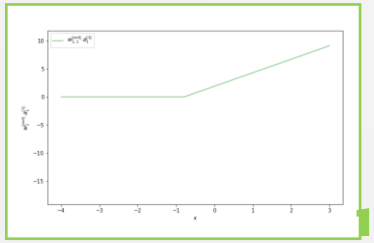
Regression | **Example 5.4**

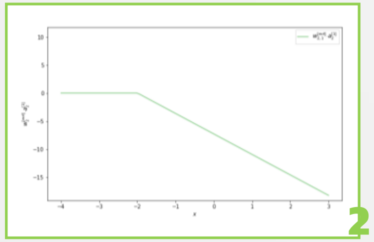


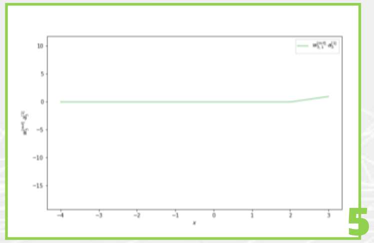
Neural Network Interpretation

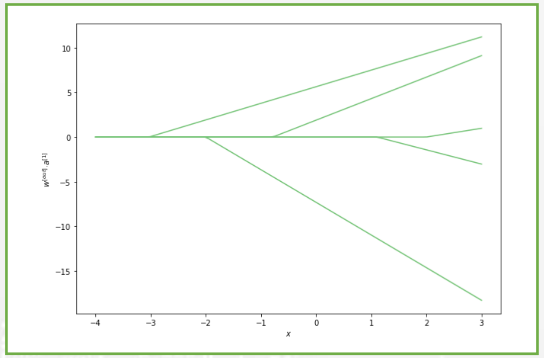


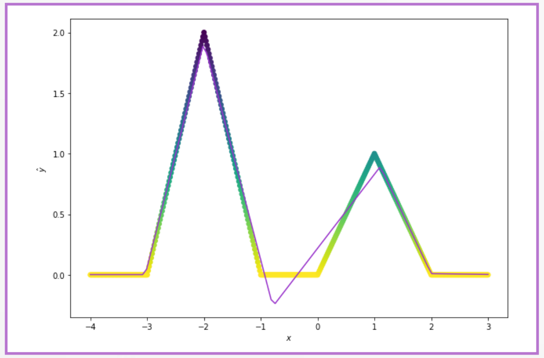
Regression | **Example 5.4**







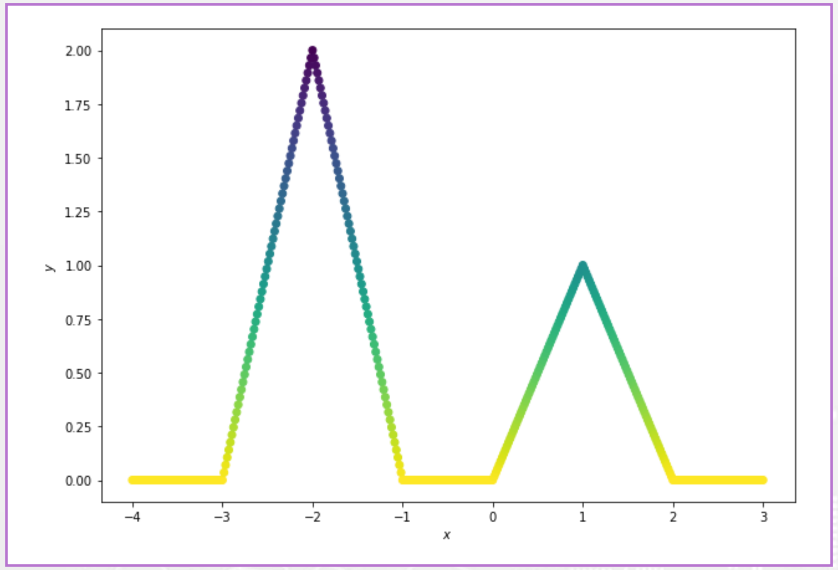


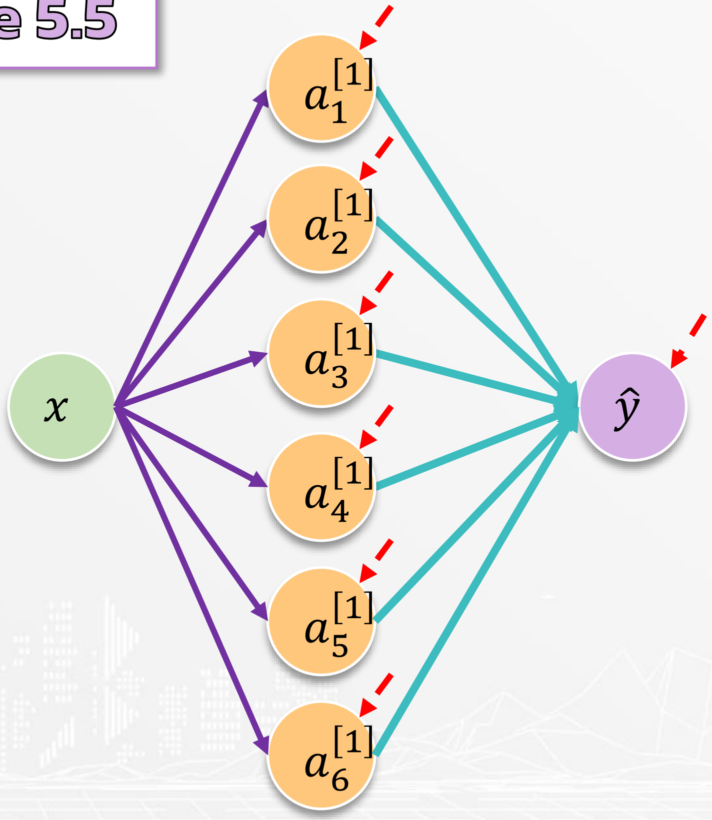


Neural Network Interpretation

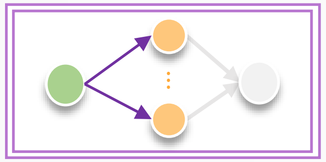
Regression

Example 5.5

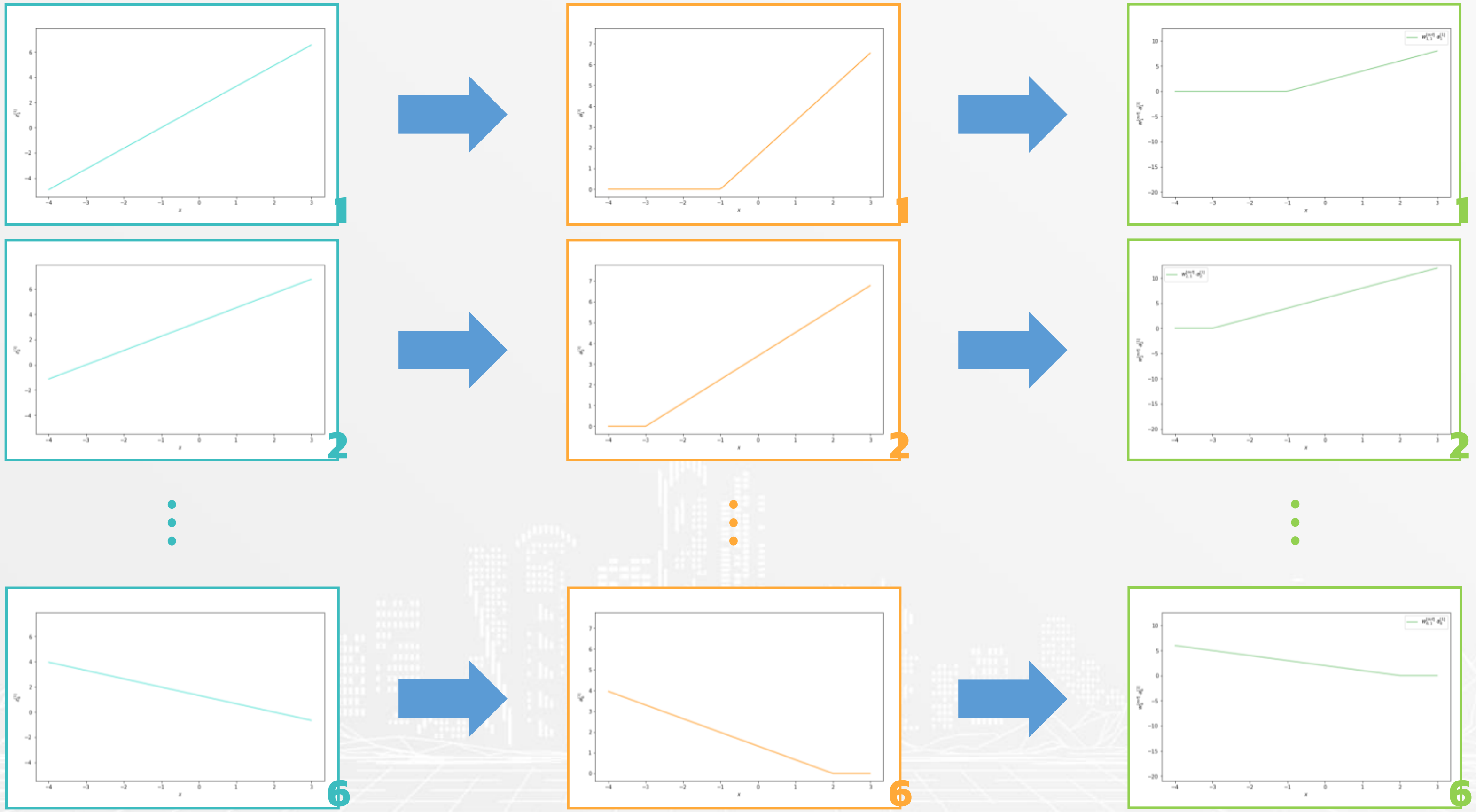




Neural Network Interpretation

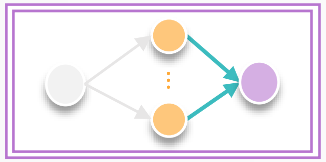


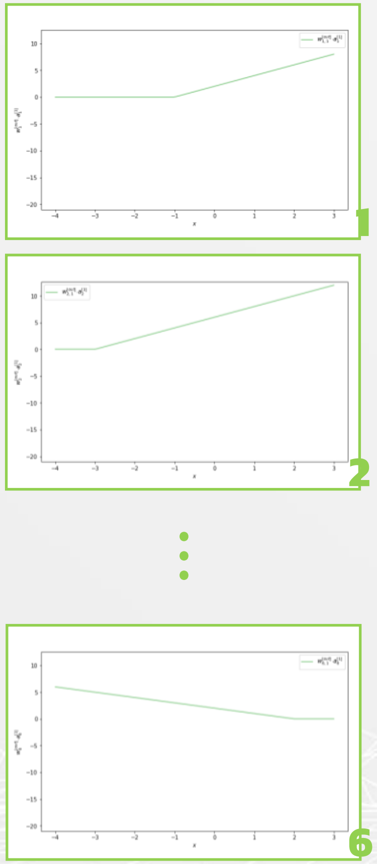
Regression | **Example 5.5**

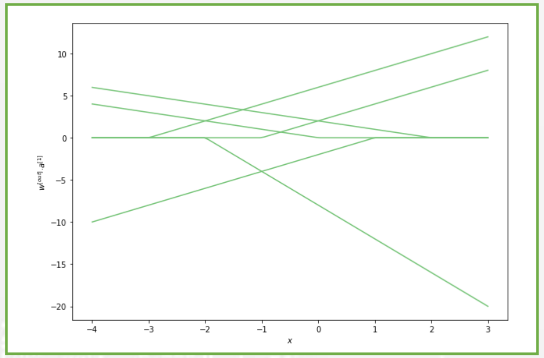


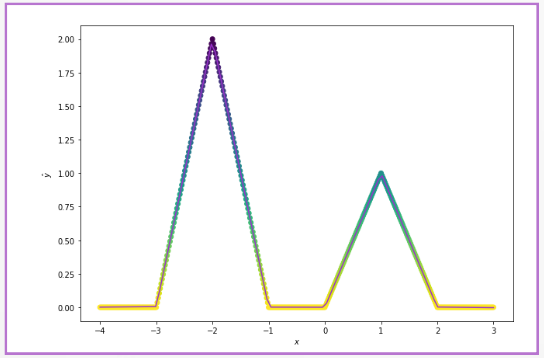
Neural Network Interpretation

Regression | **Example 5.5**





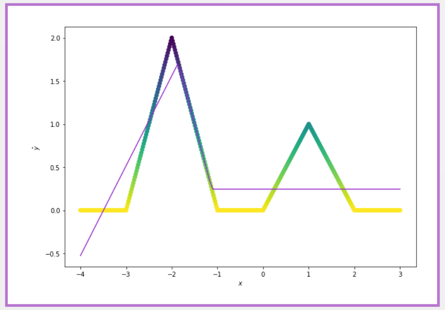




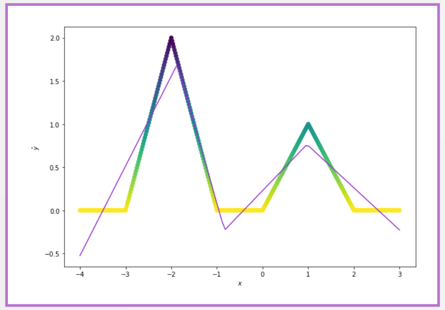
Neural Network Interpretation

Regression

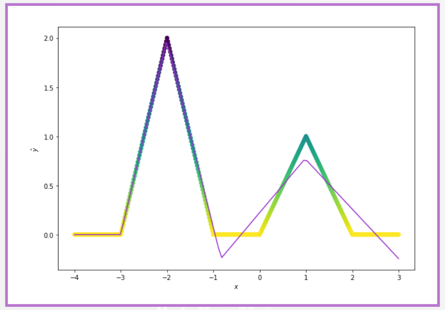
กราฟแสดงค่า predicted ของ Example 5



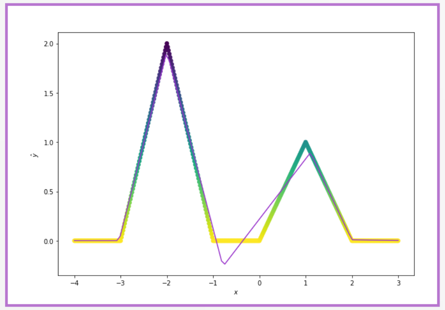
Example 5.1 (1,2,1)



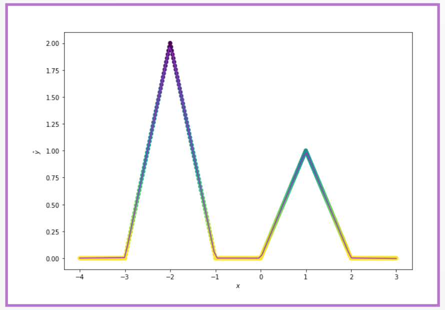
Example 5.2 (1,3,1)



Example 5.3 (1,4,1)



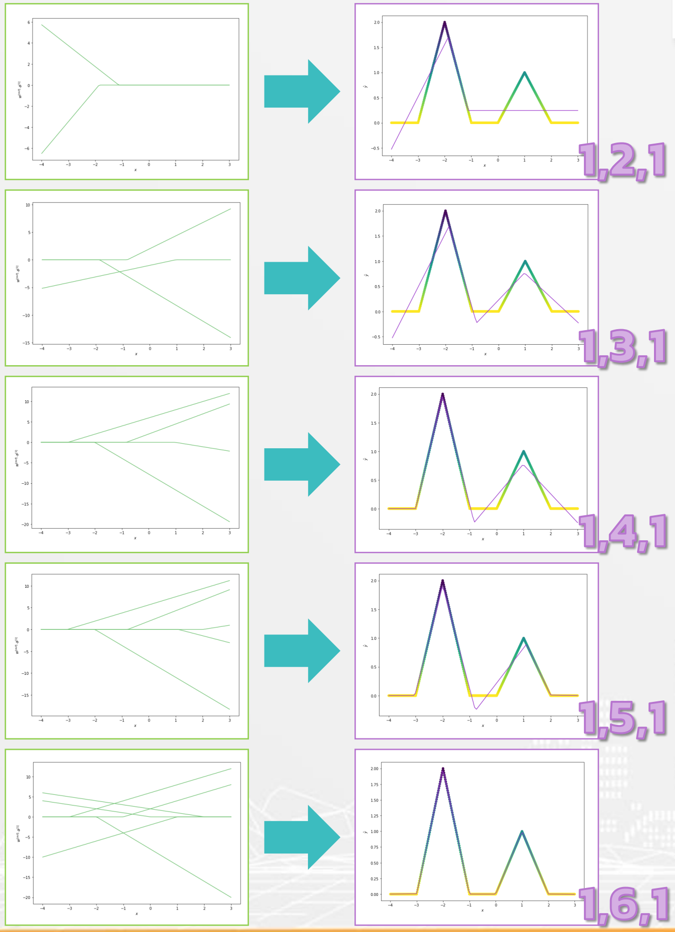
Example 5.4 (1,5,1)



Example 5.5 (1,6,1)

Neural Network Interpretation

Regression



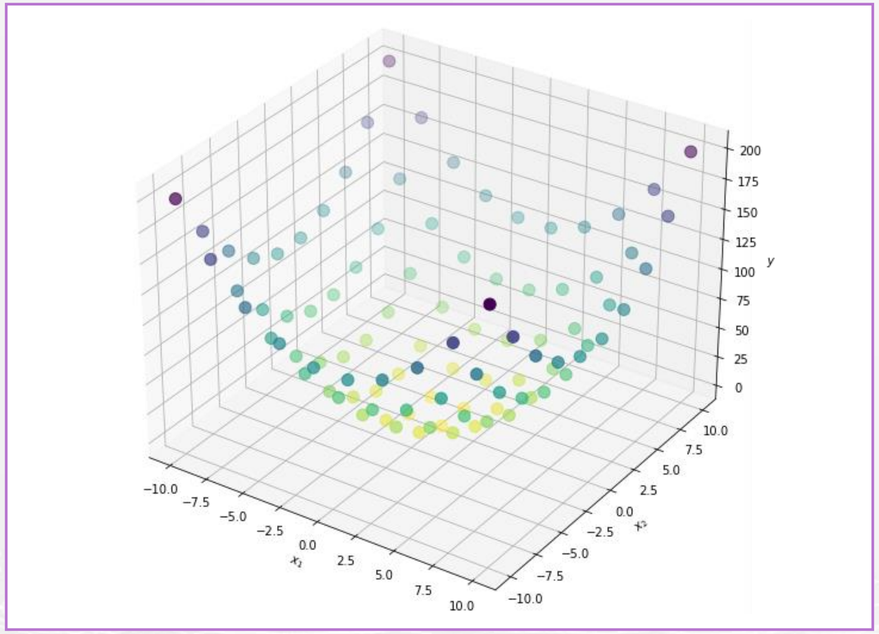
ข้อสรุปของ Example 5

- จำนวน plane (ส่วนของเส้นตรง) สูงสุดที่เกิดขึ้นในชั้น output ใน 2 มิติ(จำนวนตัวแปรต้น 1 ตัว) ที่มีจำนวนเท่ากับ จำนวน node ใน hidden layer + 1 นั้นใช้ได้เพียงแค่ใน 2 มิติเท่านั้น

Neural Network Interpretation

Regression

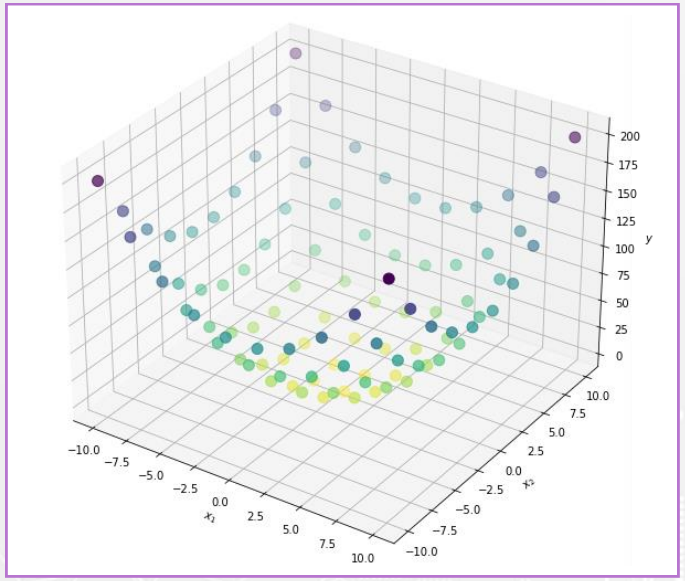
Example 6

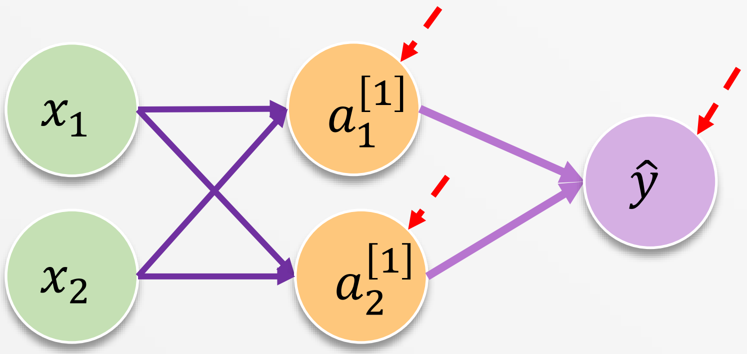


Neural Network Interpretation

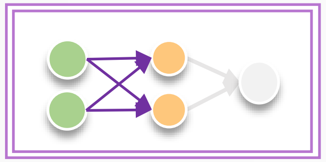
Regression

Example 6.1

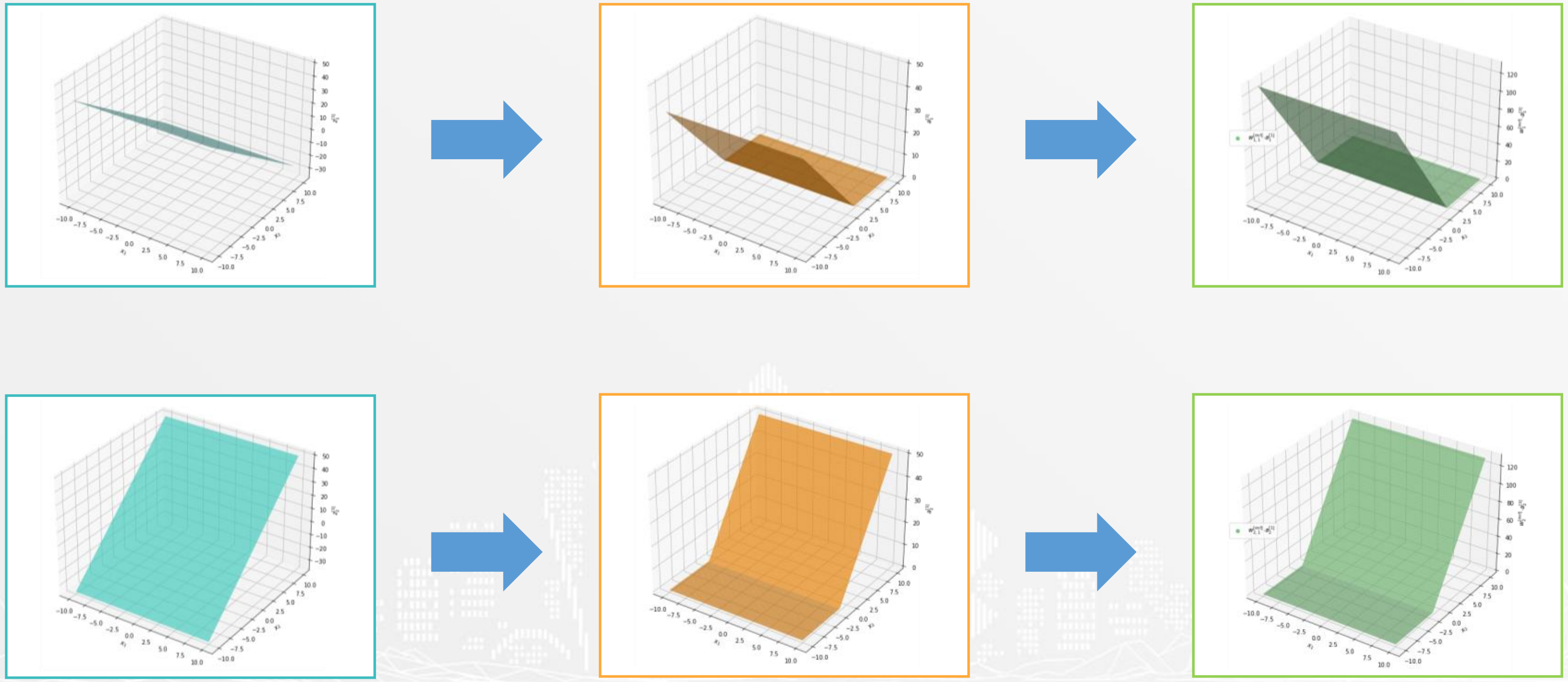




Neural Network Interpretation

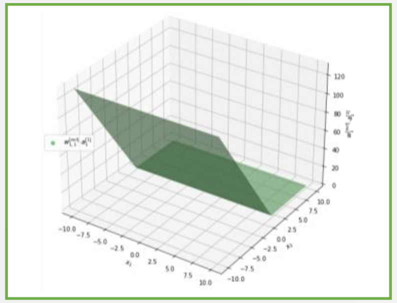


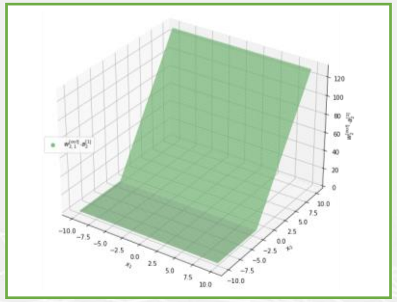
Regression | **Example 6.1**

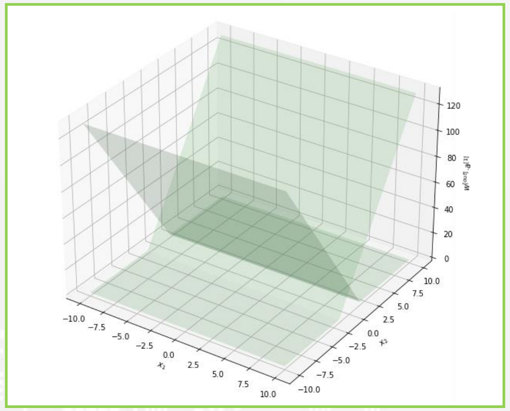


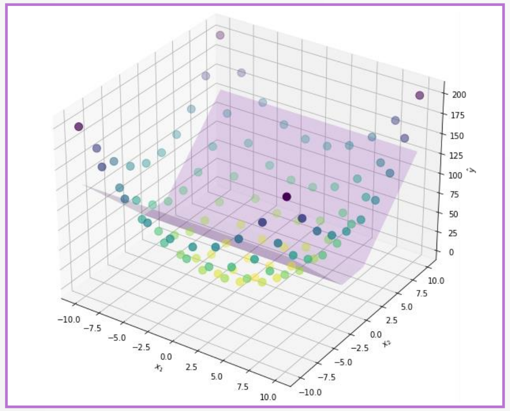
Neural Network Interpretation

Regression **Example 6.1**



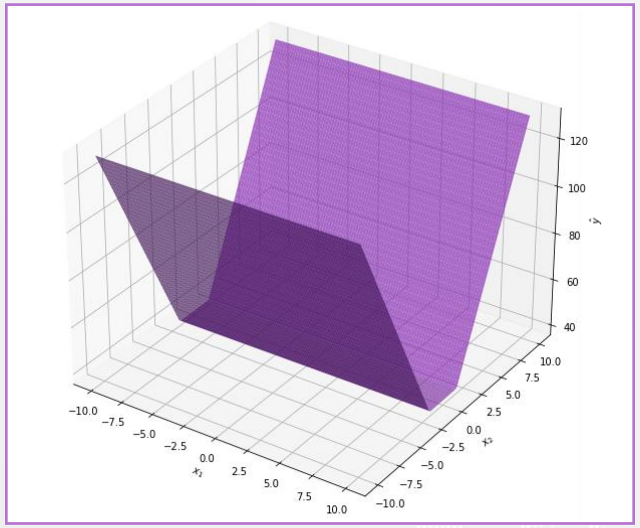




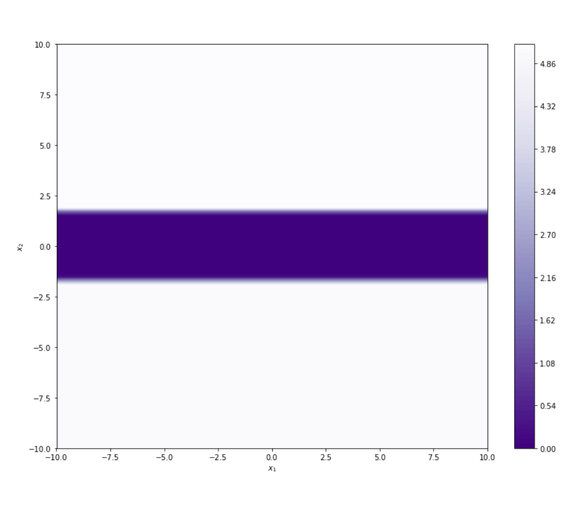


Neural Network Interpretation

Regression | Example 6.1



กราฟแสดงค่า predicted

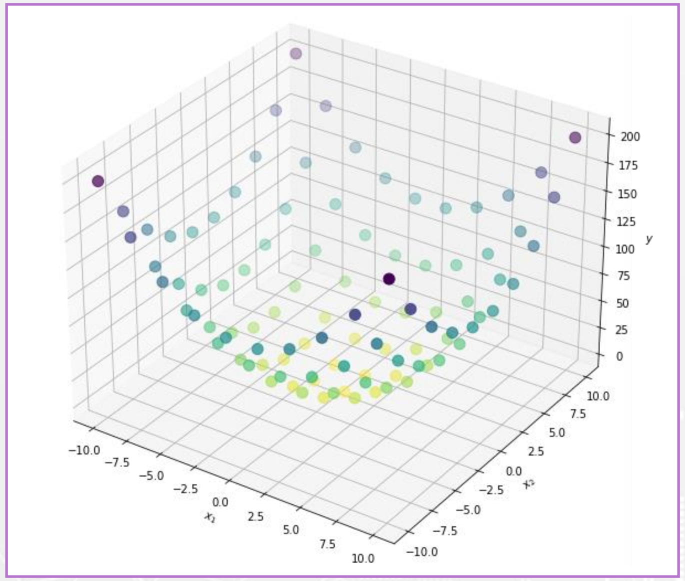


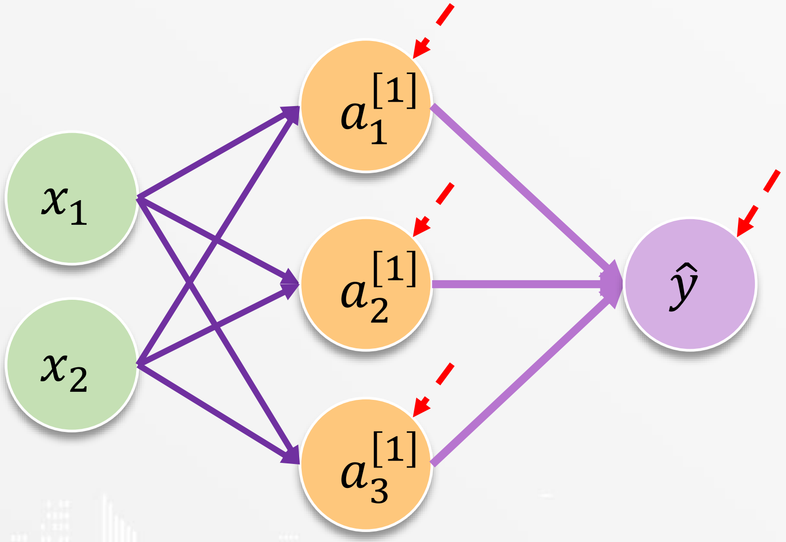
Contour plot ของ predicted

Neural Network Interpretation

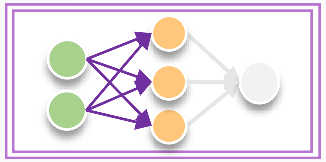
Regression

Example 6.2

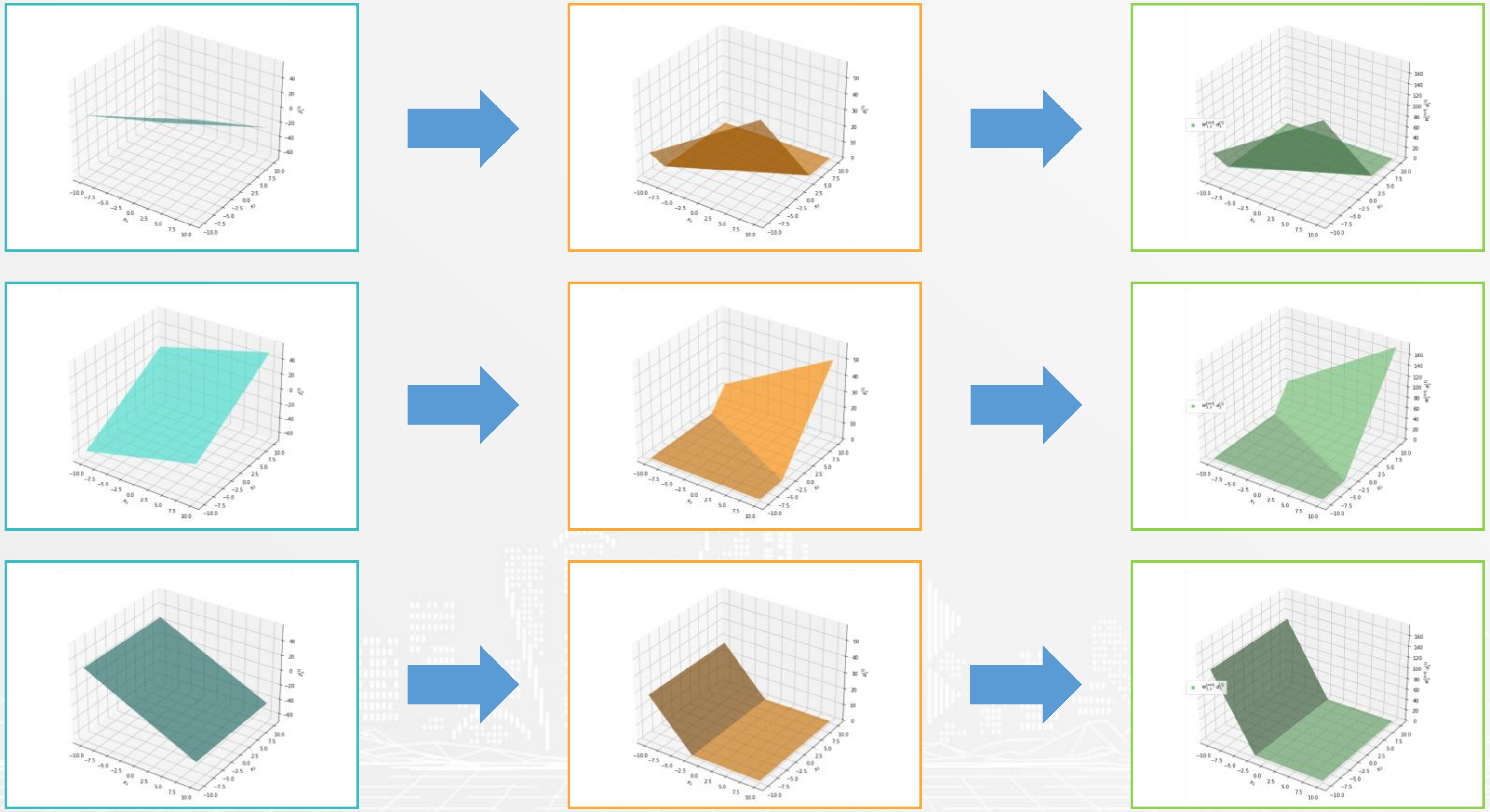




Neural Network Interpretation



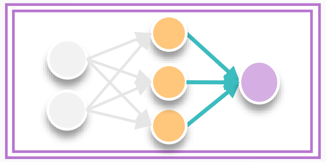
Regression | **Example 6.2**

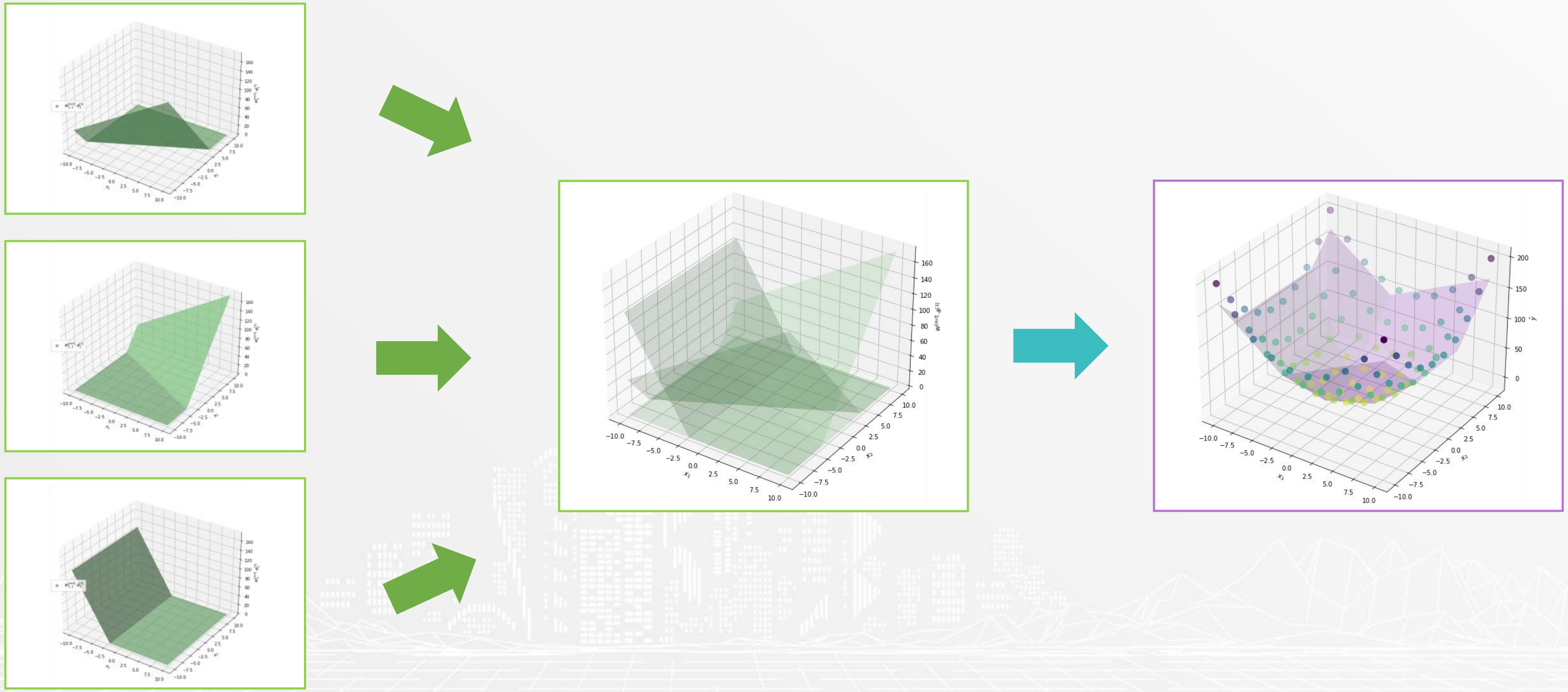


TAUTOLOGY

Neural Network Interpretation

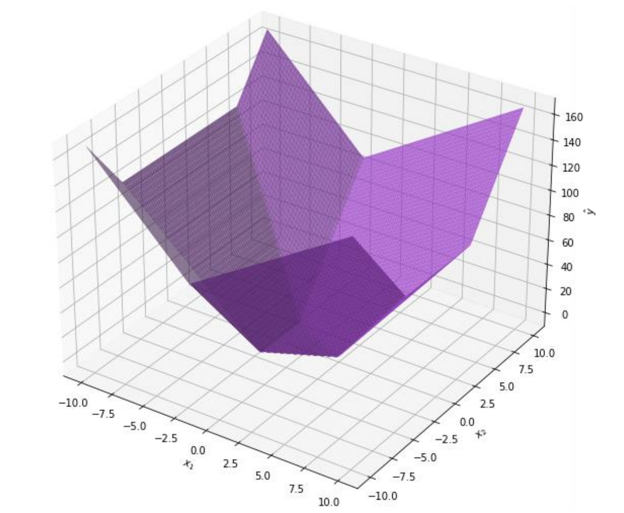
Regression | Example 6.2



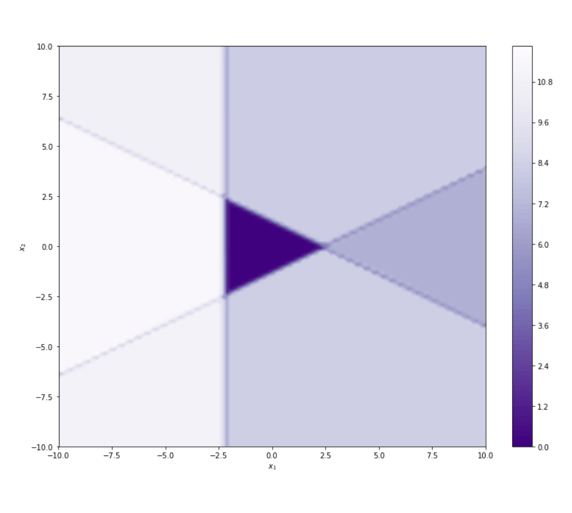


Neural Network Interpretation

Regression **Example 6.2**



กราฟแสดงค่า predicted

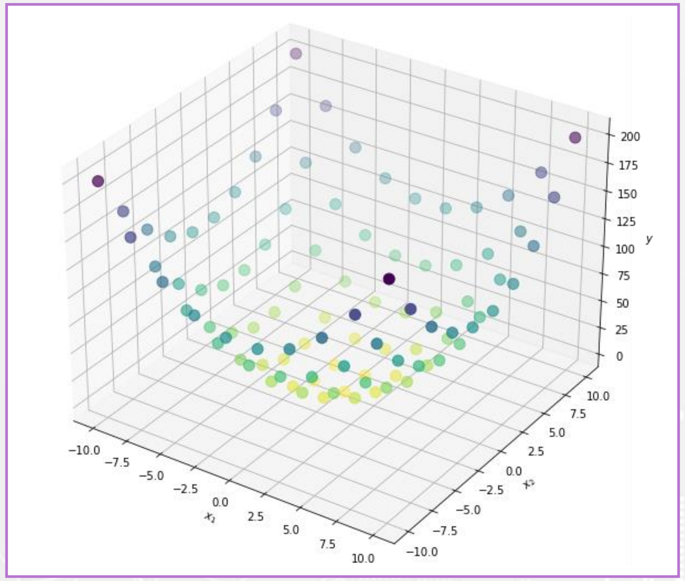


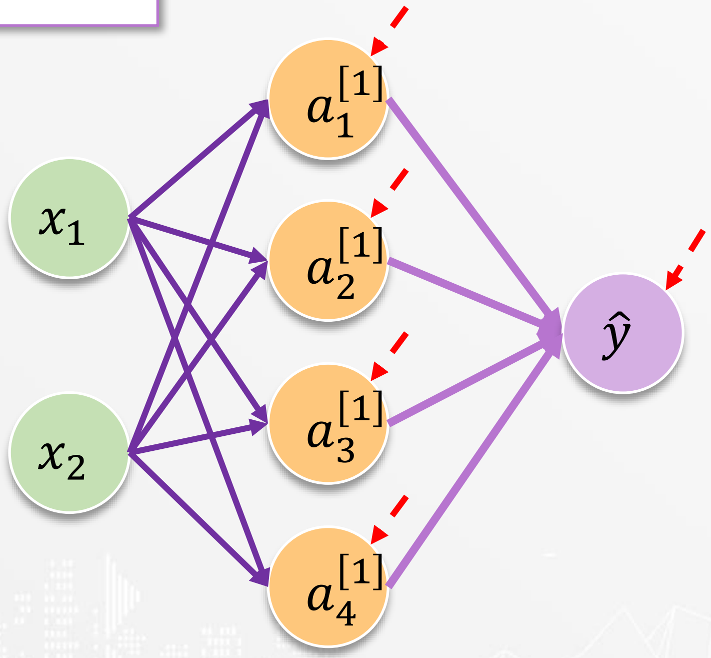
Contour plot ของ predicted

Neural Network Interpretation

Regression

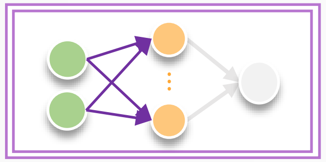
Example 6.3



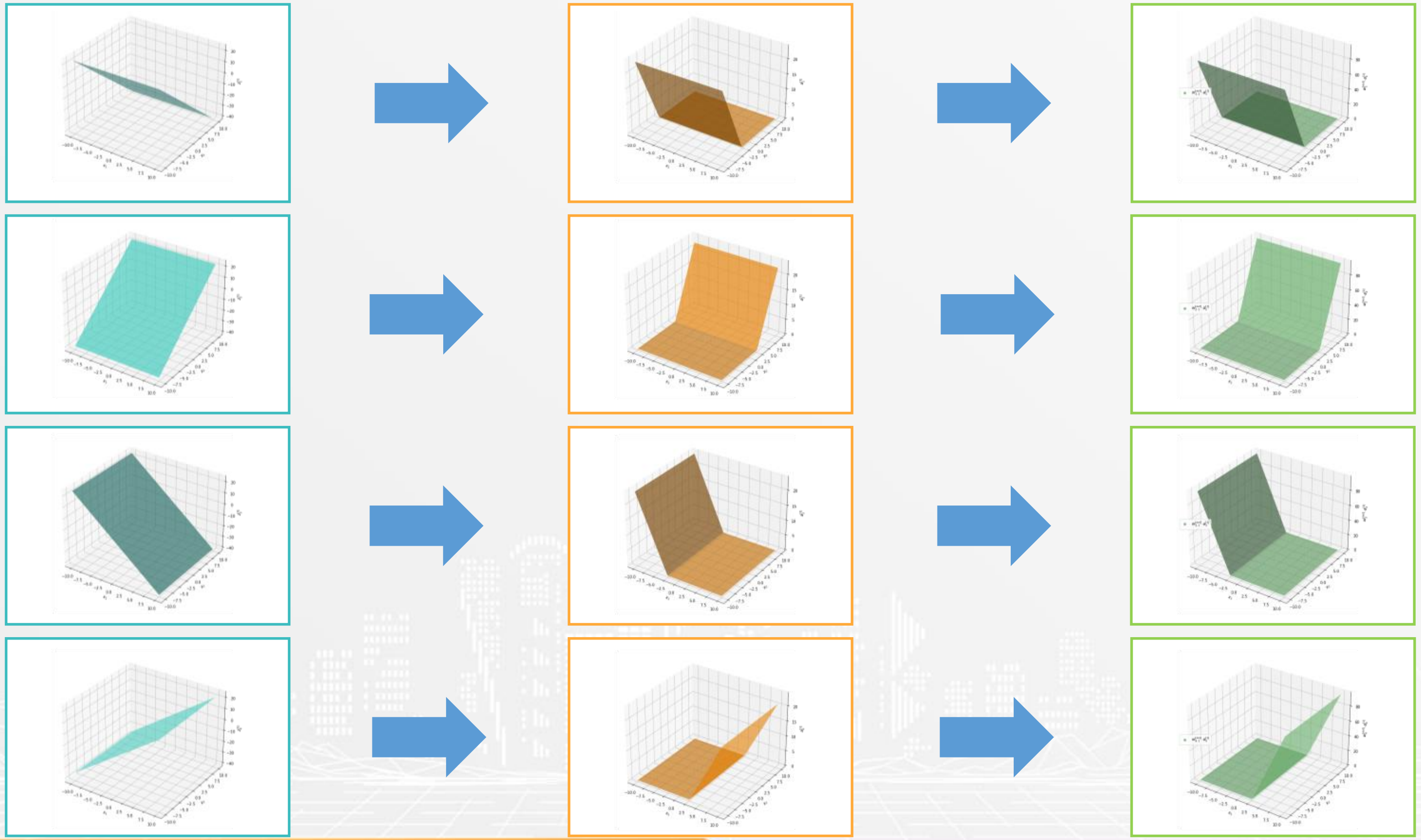


TAUTOLOGY

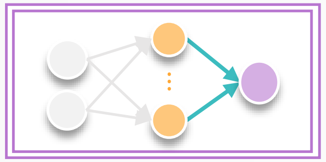
Neural Network Interpretation



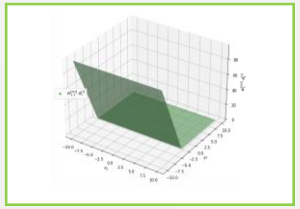
Regression | Example 6.3

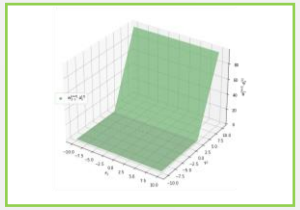


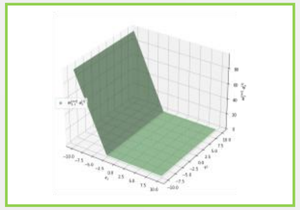
Neural Network Interpretation

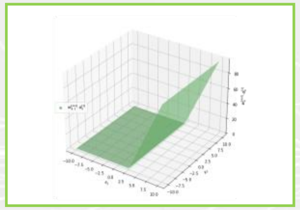


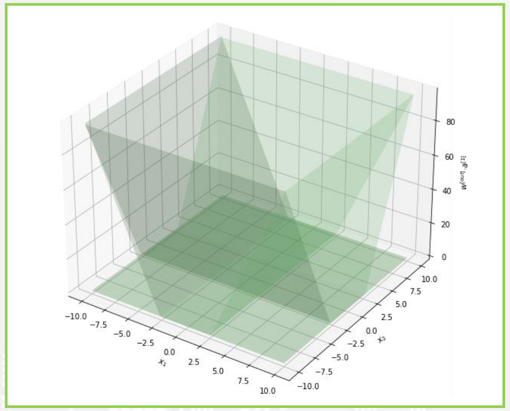
Regression | **Example 6.3**

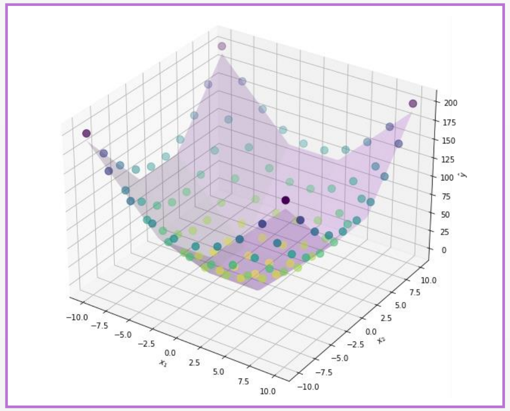








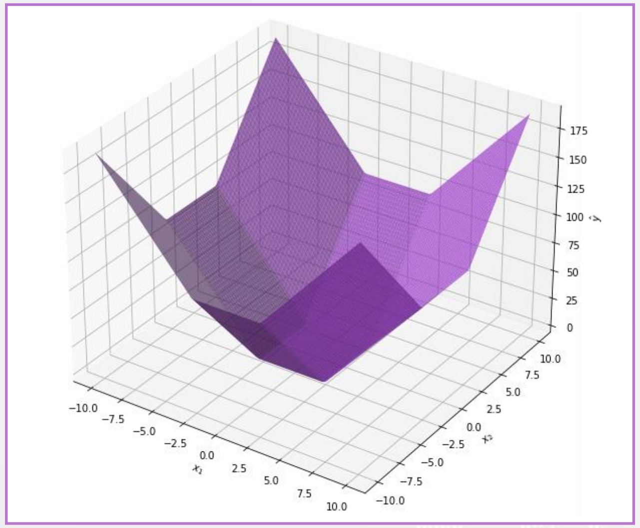




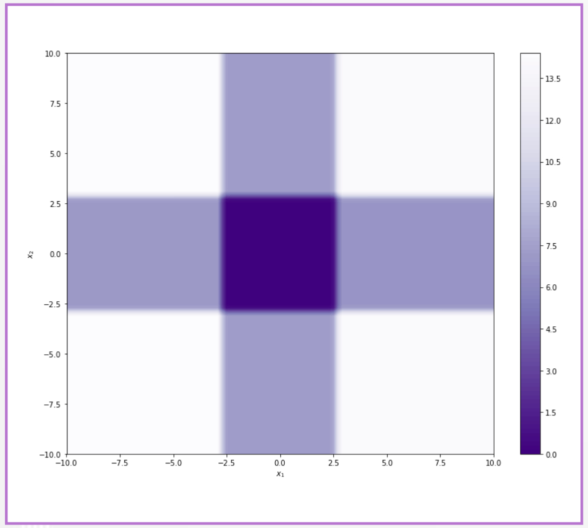
TAUTOLOGY

Neural Network Interpretation

Regression | Example 6.3



กราฟแสดงค่า predicted

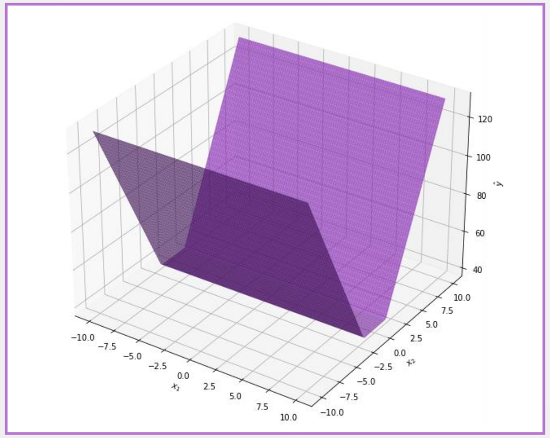


Contour plot ของ predicted

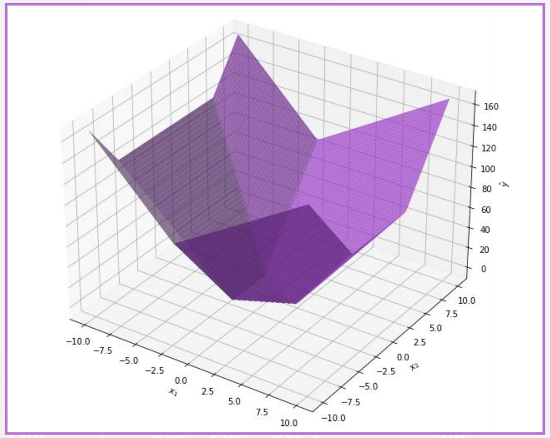
Neural Network Interpretation

Regression

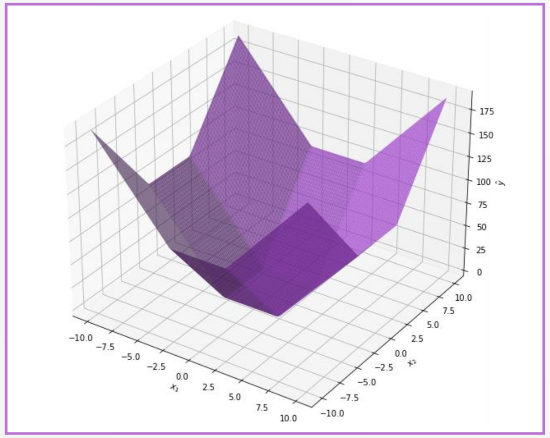
กราฟแสดงค่า predicted ของ Example 6



Example 6.1 (2,2,1)



Example 6.2 (2,3,1)

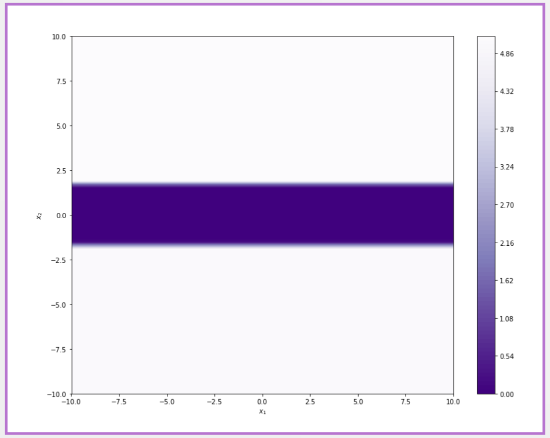


Example 6.3 (2,4,1)

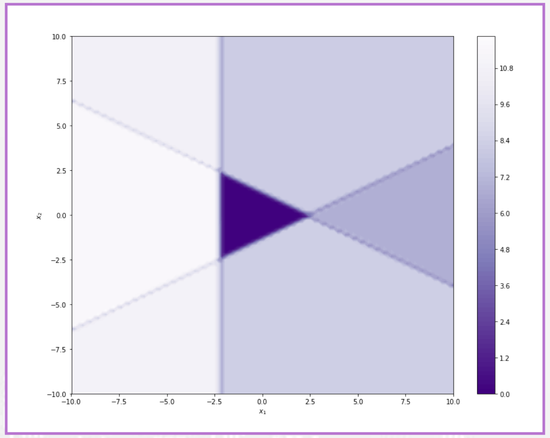
Neural Network Interpretation

Regression

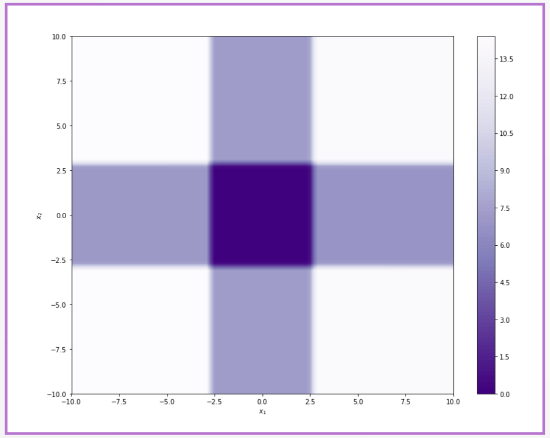
Contour plot predicted ของ Example 6



Example 6.1 (2,2,1)



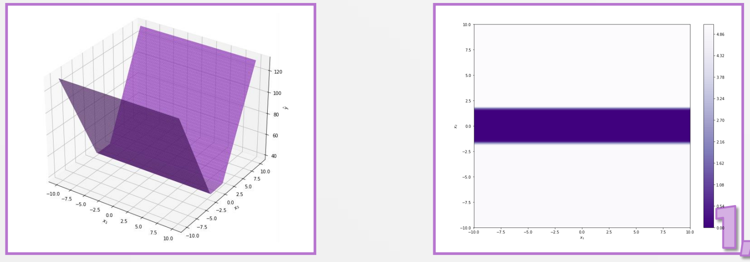
Example 6.2 (2,3,1)



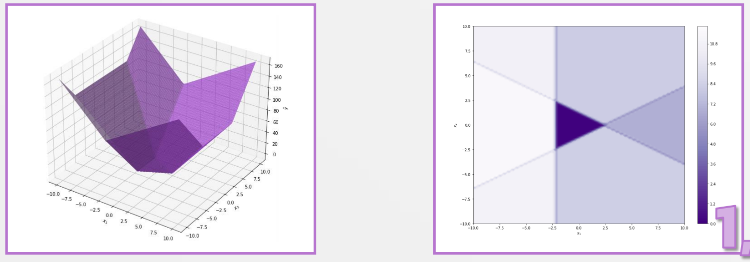
Example 6.3 (2,4,1)

Neural Network Interpretation

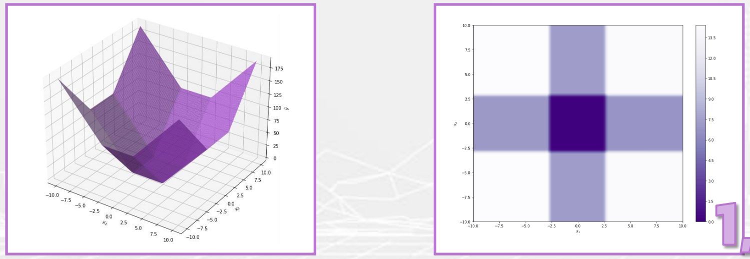
Regression



1,2,1



1,3,1



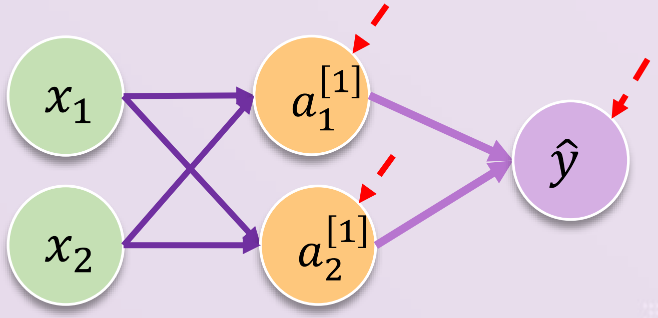
1,4,1

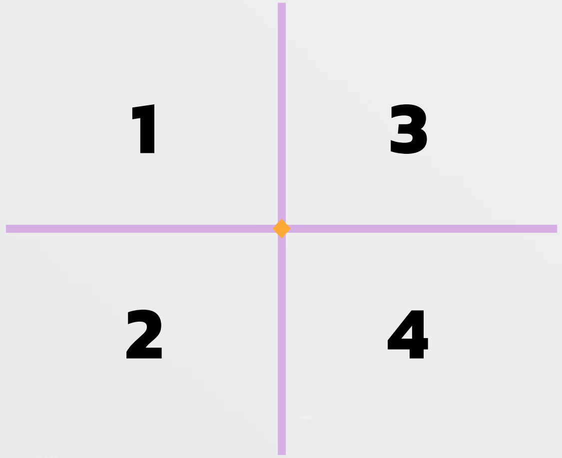
ข้อสรุปของ Example 6

- จำนวน plane (ส่วนของพื้นผิว) สูงสุดที่เกิดขึ้นในชั้น output ใน 3 มิติ (จำนวนตัวแปรต้น 2 ตัว) มีความแตกต่างจาก จำนวน plane (ส่วนของเส้นตรง) สูงสุดที่เกิดขึ้นในชั้น output ใน 2 มิติ (จำนวนตัวแปรต้น 1 ตัว)

Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้

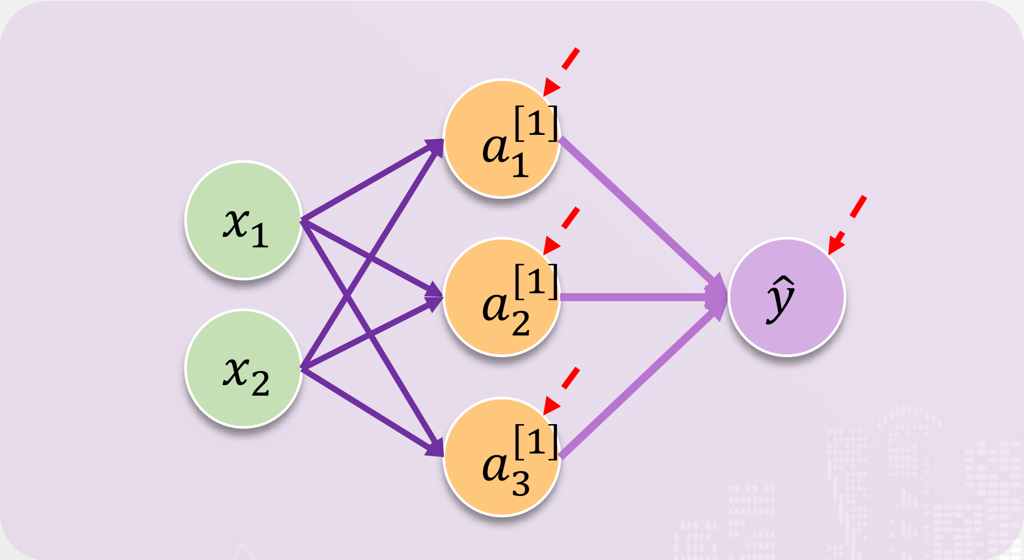


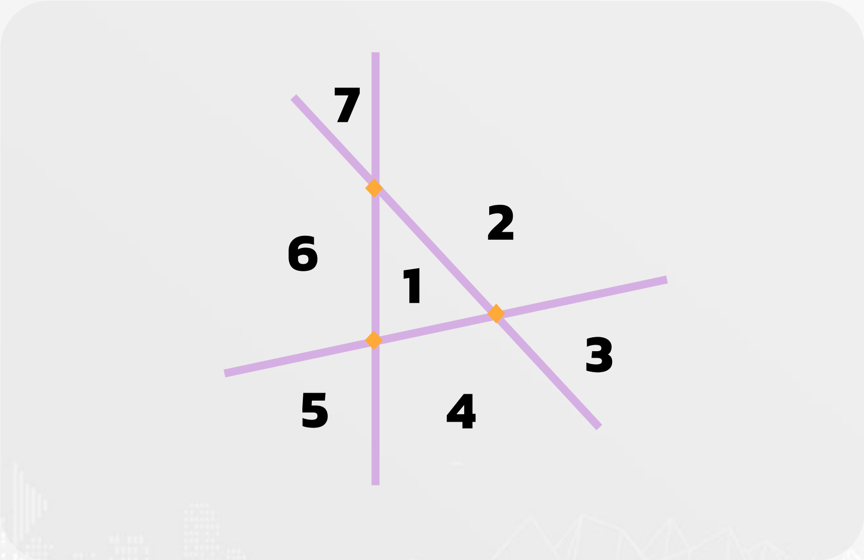


TAUTOLOGY

Neural Network Interpretation

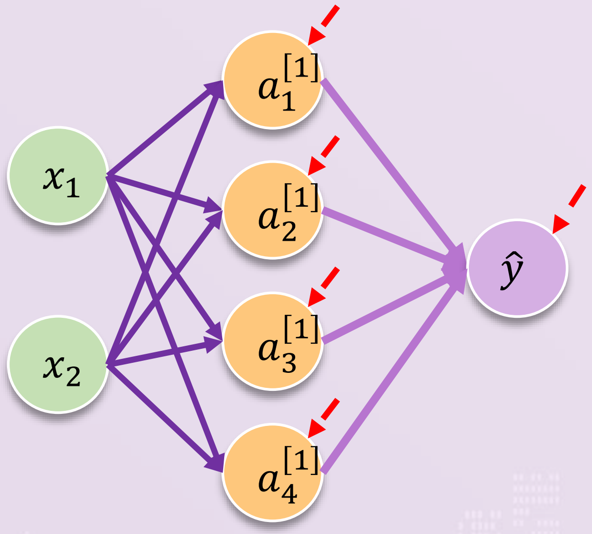
Maximum Plane ที่สามารถเกิดขึ้นได้

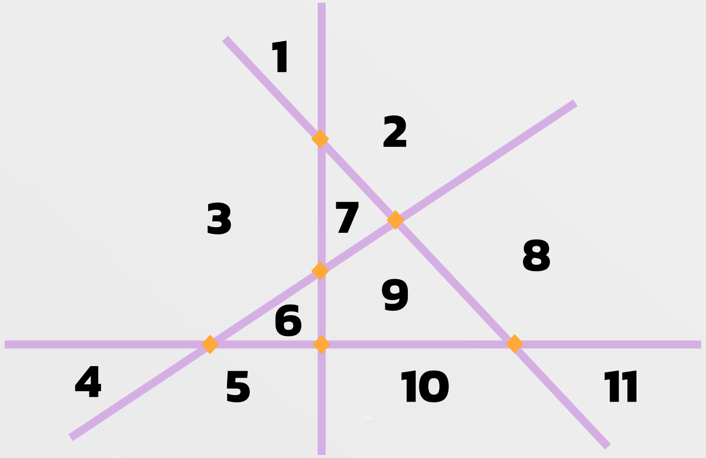




Neural Network Interpretation

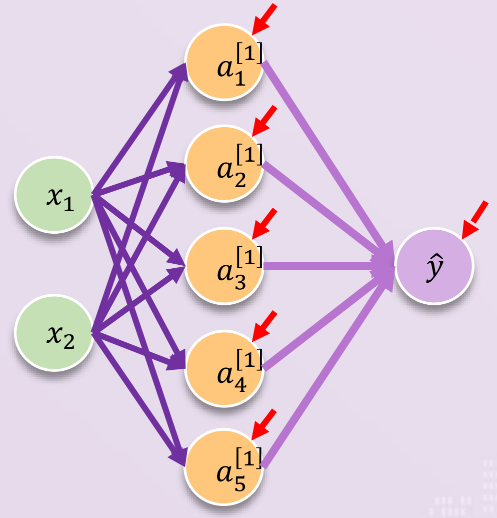
Maximum Plane ที่สามารถเกิดขึ้นได้

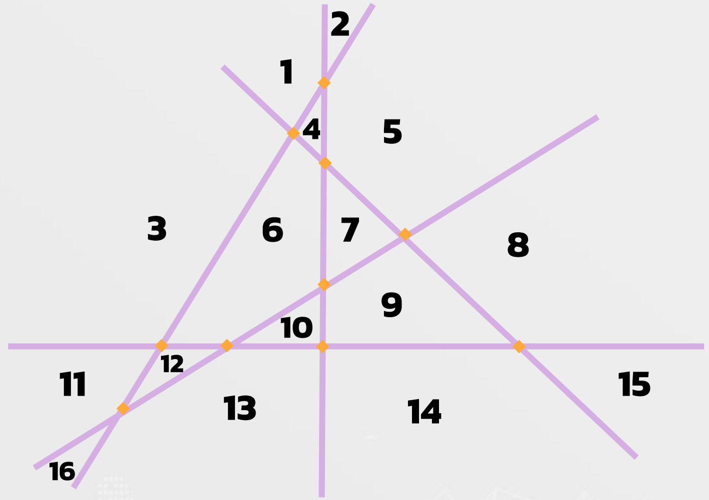




Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้





Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้

Maximum Plane ที่เกิดจาก $ReLU$ จะเกิดได้ เมื่อรอยหัก เกิดการตัดกันทั้งหมด

(ในแต่ละรอยหัก จะเกิดจุดตัด ตามจำนวน **"node-1"**)

Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้

จำนวน node ใน hidden layer	จำนวน combine plane ที่เกิดขึ้นได้มากที่สุด
2	4
3	7
4	11
5	16
⋮	⋮

Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้

หากคำนวณ เราจะได้ว่า maximum plane ที่เกิดขึ้น จะเท่ากับ (3 มิติ)

$$\boxed{\frac{1}{2}(n^2 + n + 2)}$$

โดยที่ n คือ จำนวน node ใน hidden layer

Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้

ทั้งนี้จะเกิด Maximum Plane ได้ ก็ต่อเมื่อ **Dataset**, **initial parameter** มีความเหมาะสม (รอยหักเกิดการตัดกันทั้งหมด)

Neural Network Interpretation

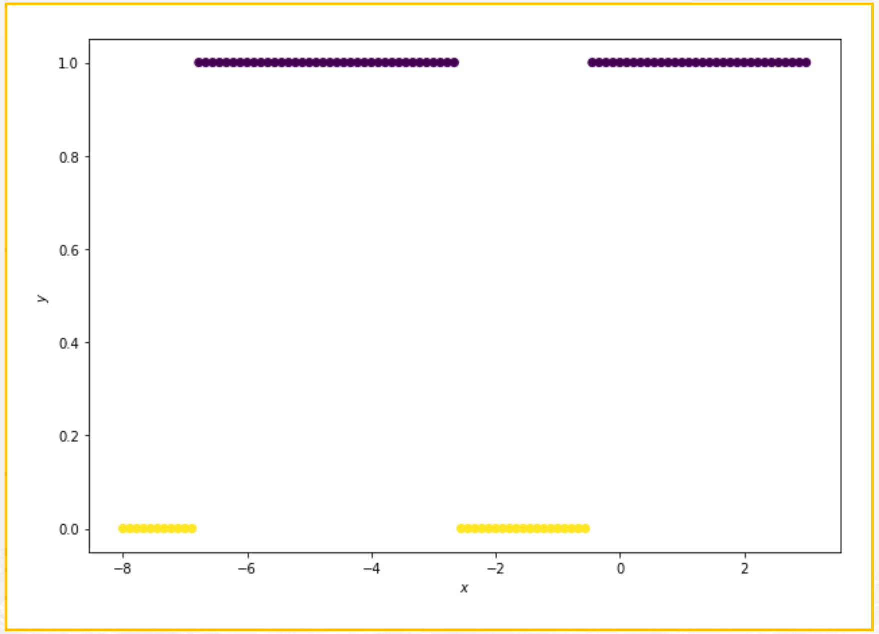
Regression

Classification

Neural Network Interpretation

Classification

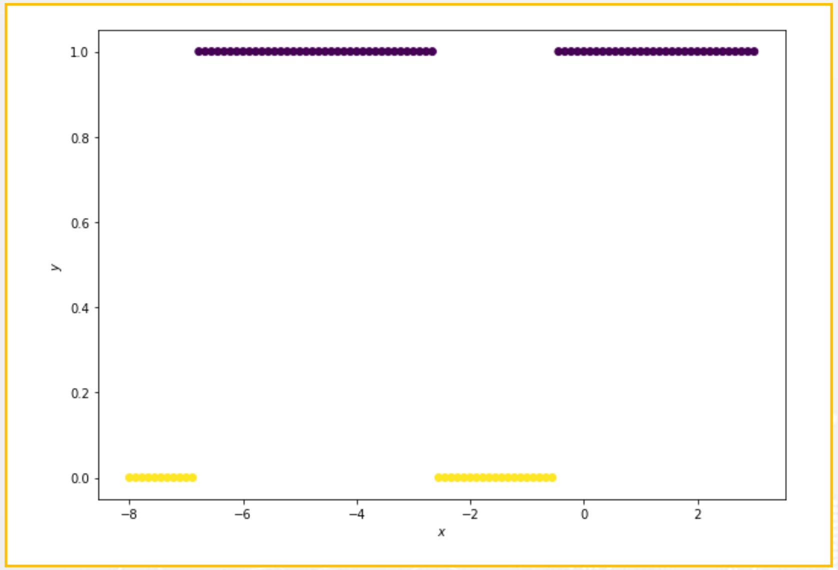
Example 1 : Binary Classification

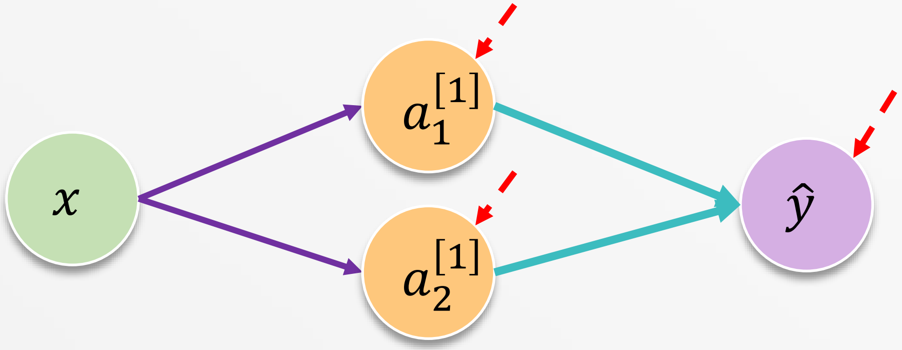


Neural Network Interpretation

Classification

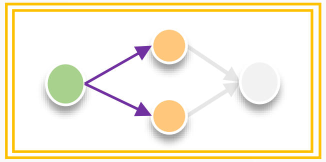
Example 1.1

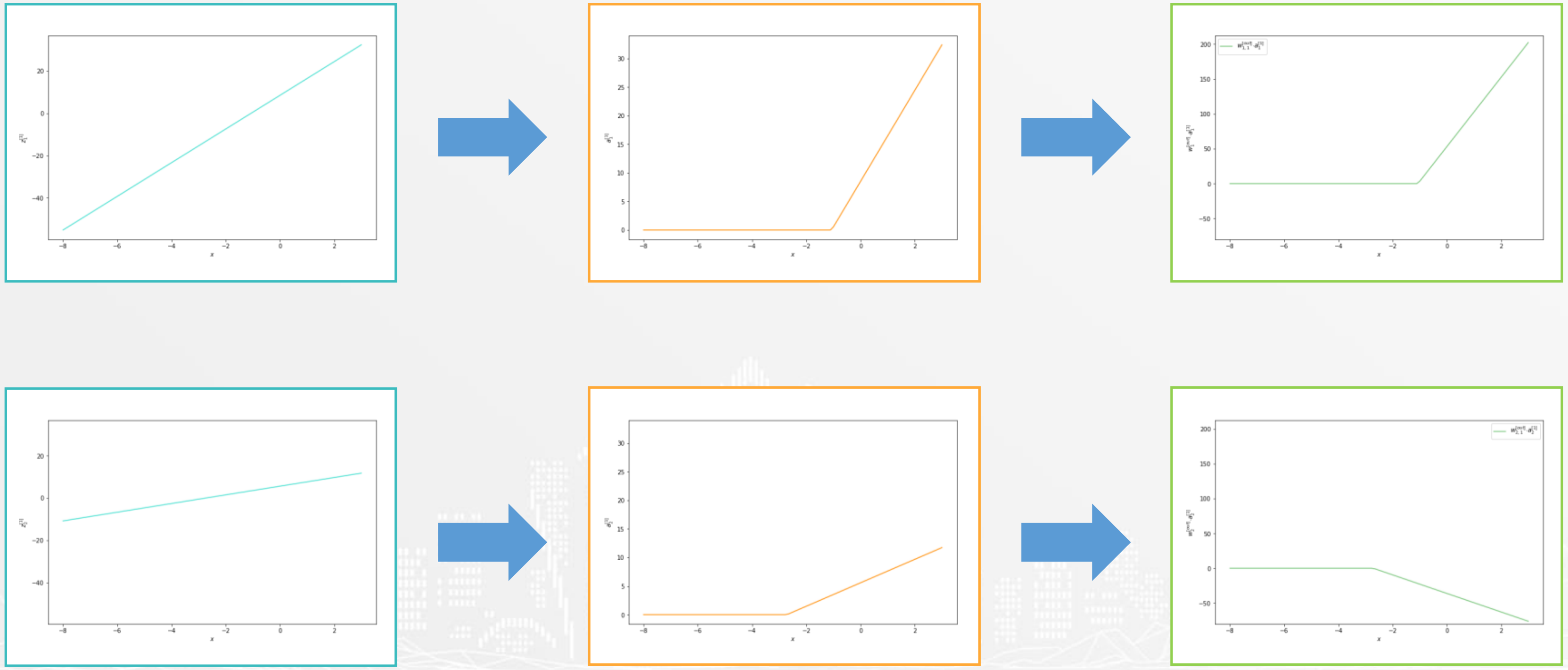




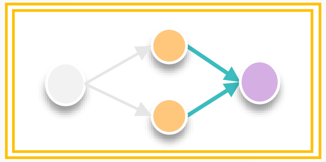
Neural Network Interpretation

Classification **Example 1.1**

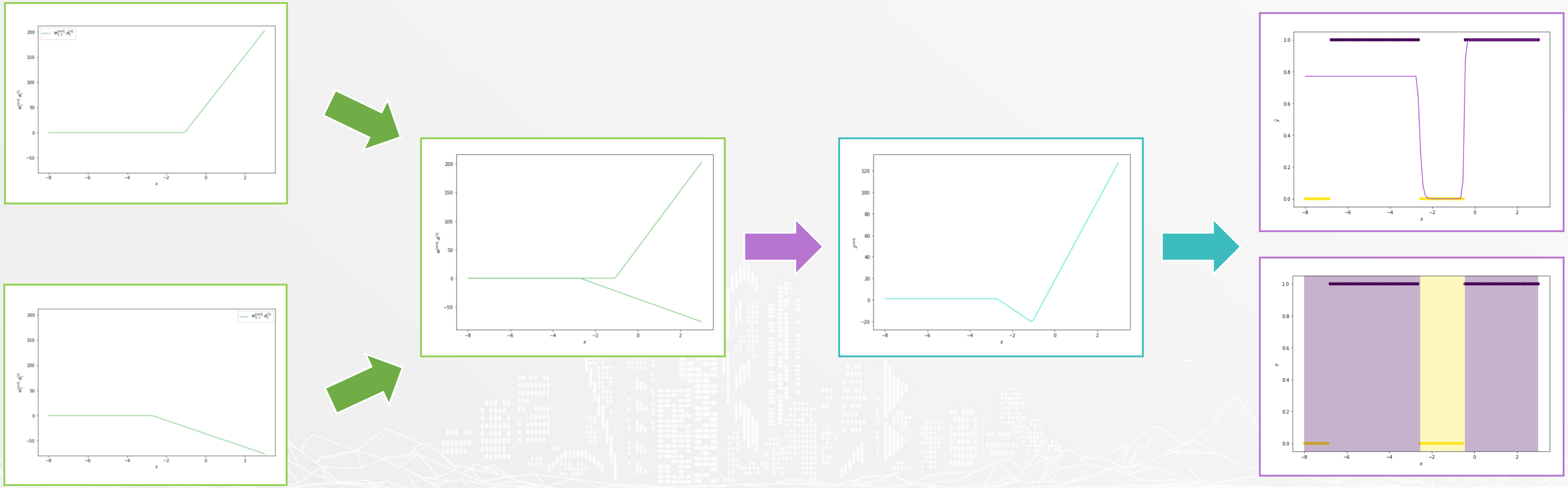




Neural Network Interpretation



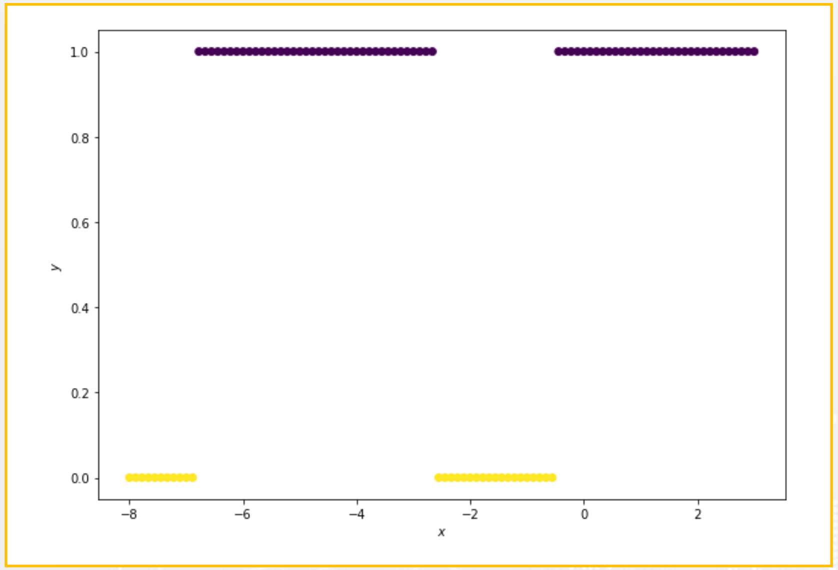
Classification **Example 1.1**

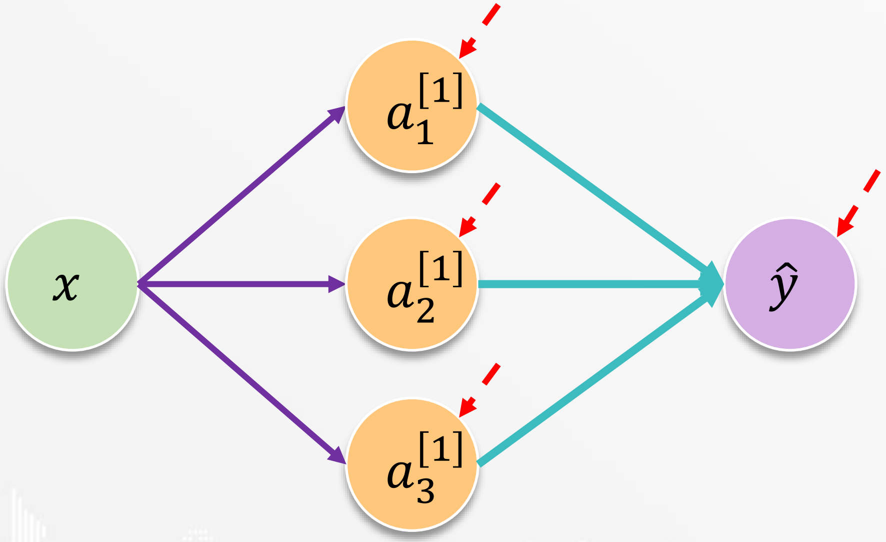


Neural Network Interpretation

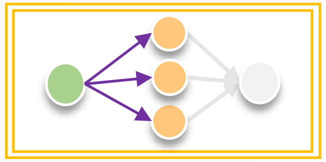
Classification

Example 1.2

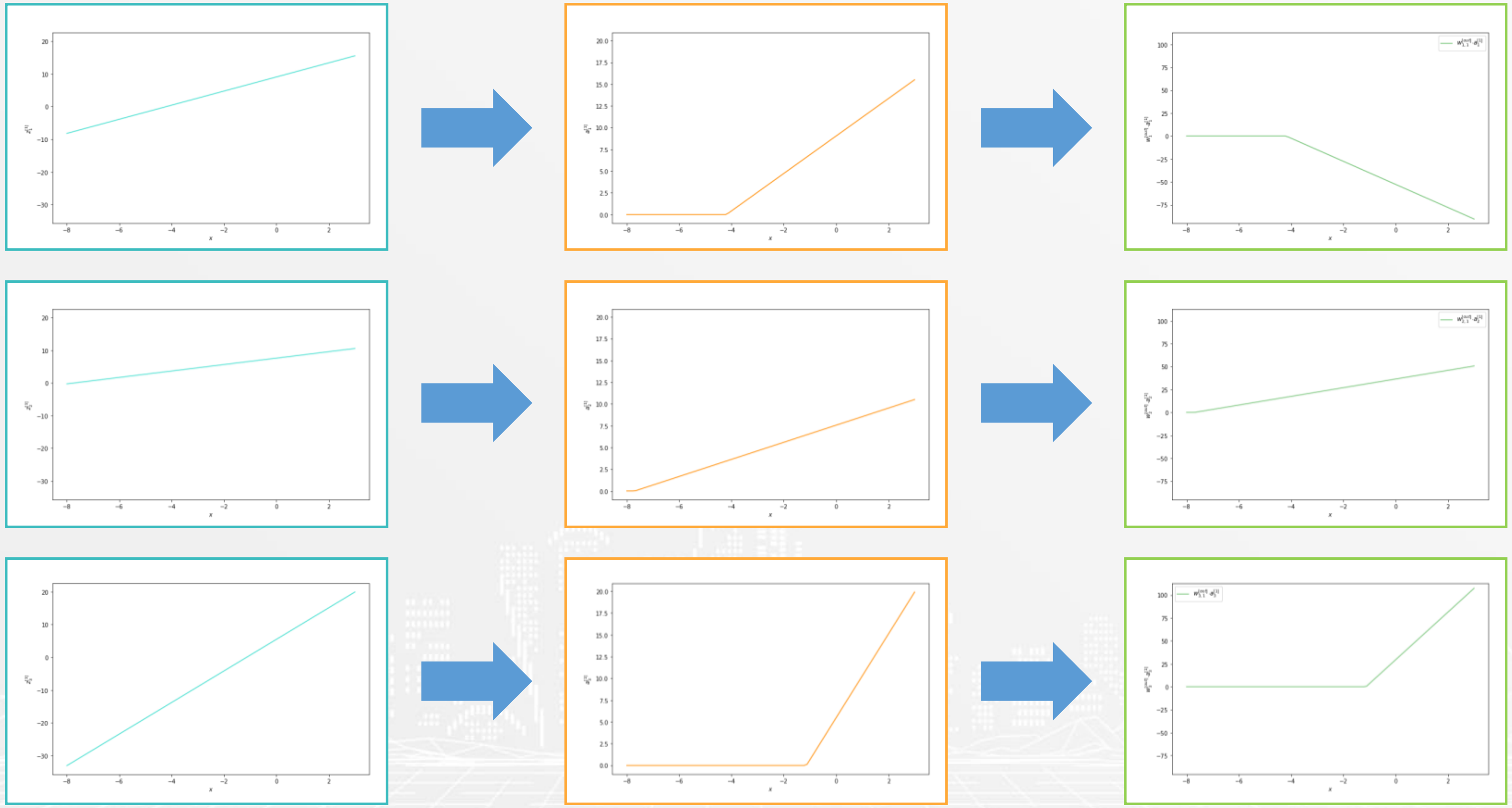




Neural Network Interpretation



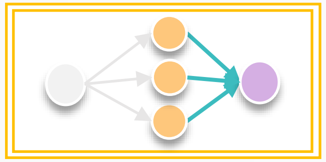
Classification | **Example 1.2**

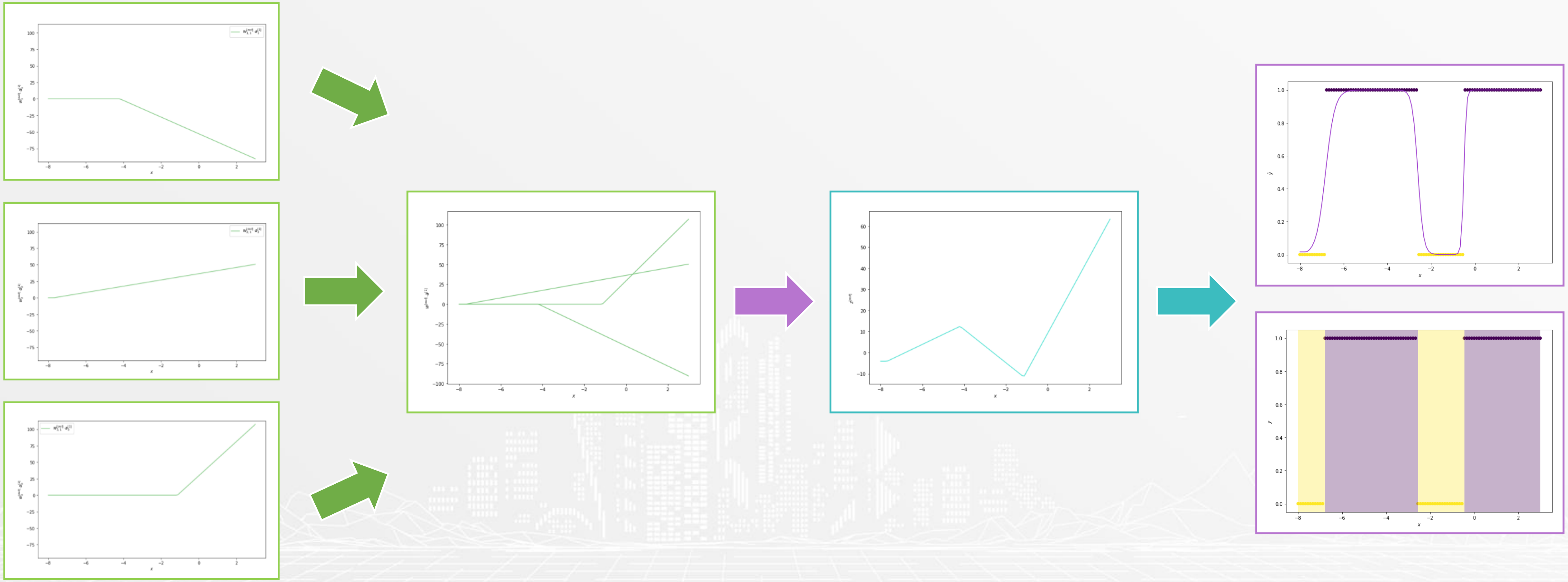


Neural Network Interpretation

Classification

Example 1.2

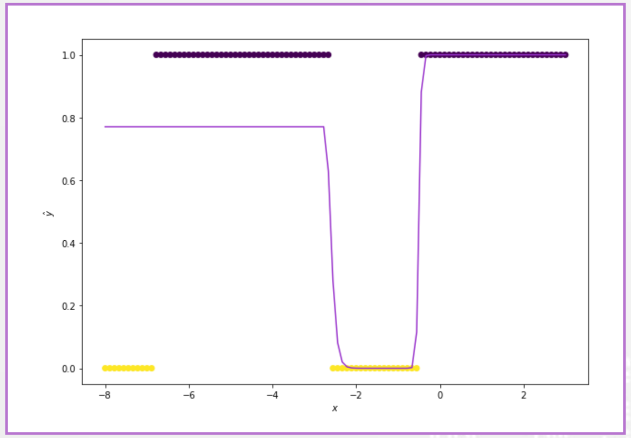




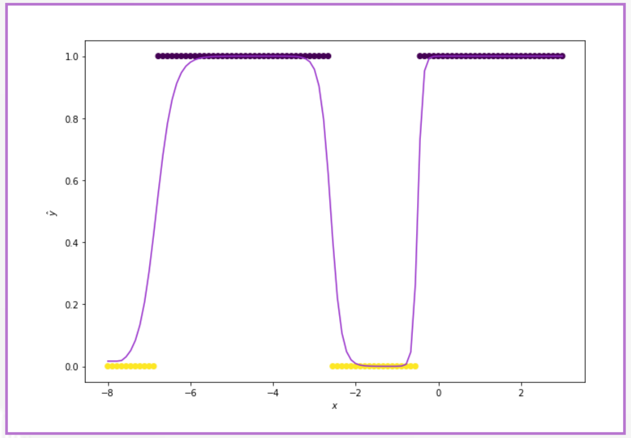
Neural Network Interpretation

Classification

กราฟแสดงค่า predicted ของ Example 1



Example 1.1 (1,2,1)

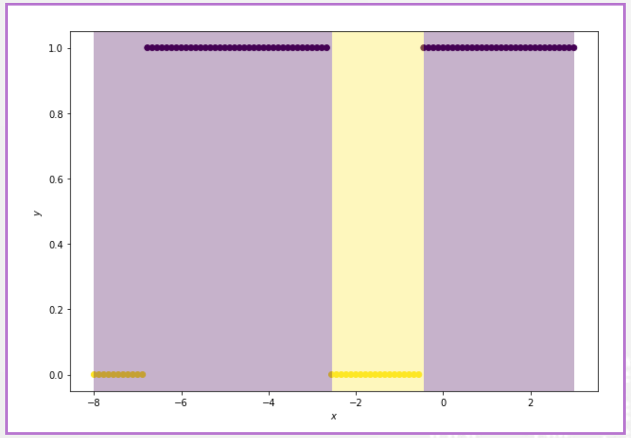


Example 1.2 (1,3,1)

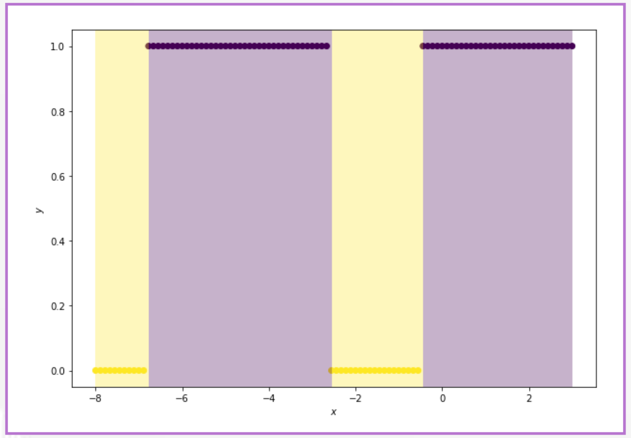
Neural Network Interpretation

Classification

กราฟแสดงพื้นที่ predicted ของ Example 1



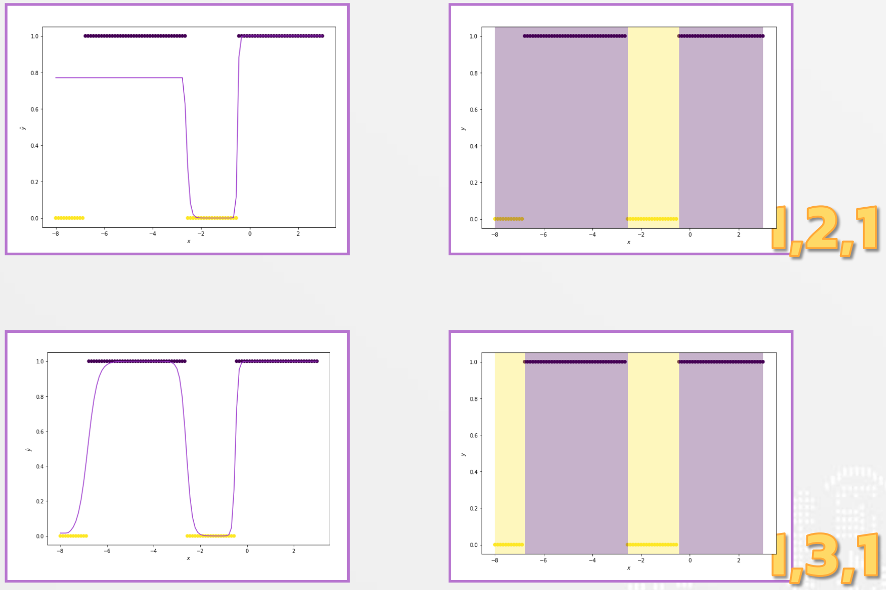
Example 1.1 (1,2,1)



Example 1.2 (1,3,1)

Neural Network Interpretation

Classification



ข้อสรุปของ Example 1

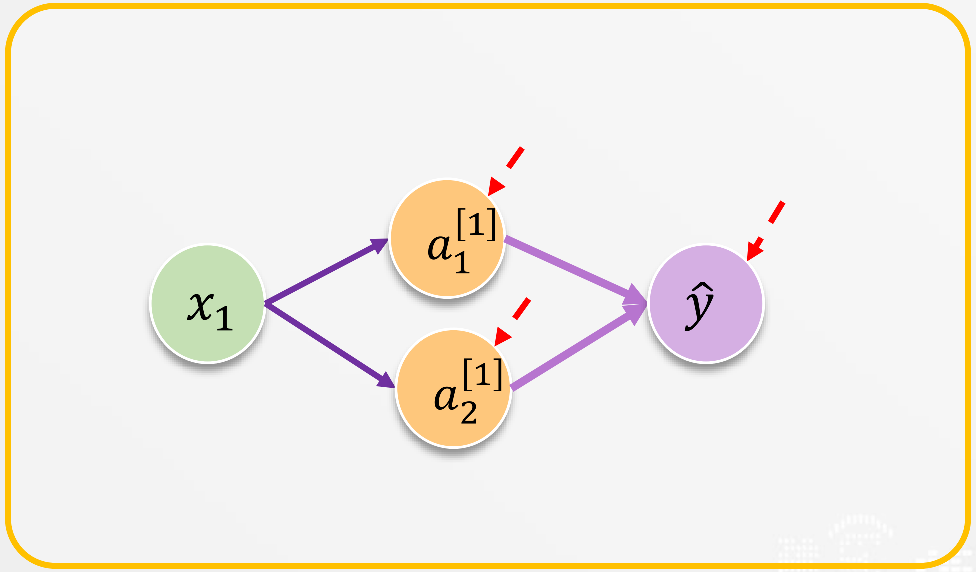
- Plane (ส่วนของเส้นตรง) ที่เพิ่มขึ้นจากจำนวน node ที่มากขึ้น เมื่อผ่าน sigmoid function แล้วจะทำให้เกิด region ของการแบ่ง class ที่เพิ่มขึ้นตามไปด้วย

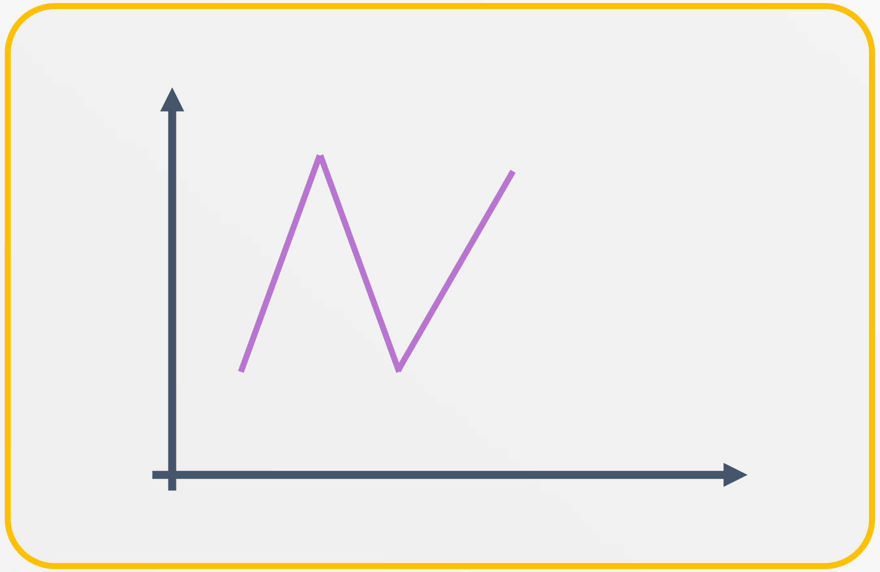
Neural Network Interpretation

Classification

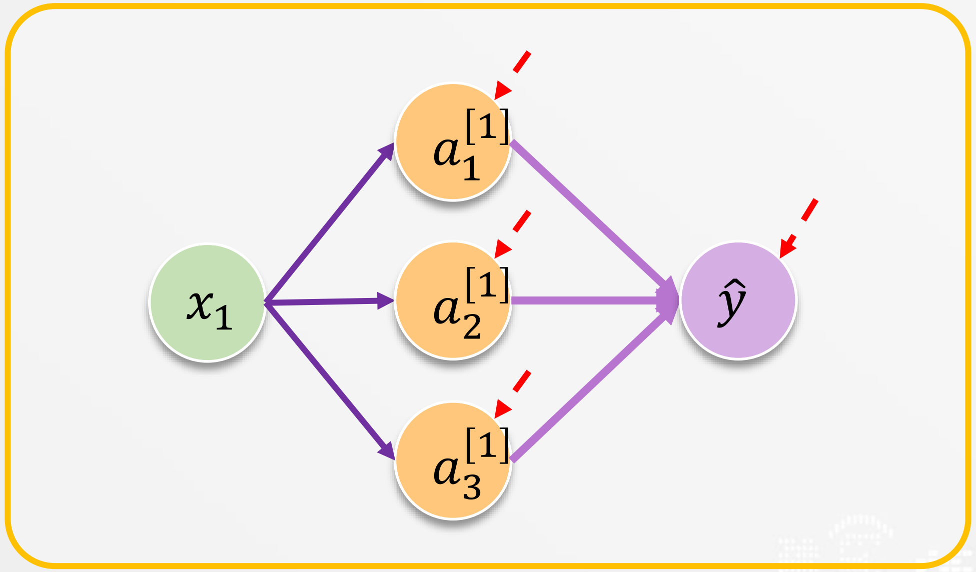
จากความรู้เดิมที่ว่า **Neural Network Regression** ใน 2 มิติ นั้น จำนวน plane (ส่วนของเส้นตรง) ในชั้น output มีจำนวนมากที่สุด คือ **n + 1**

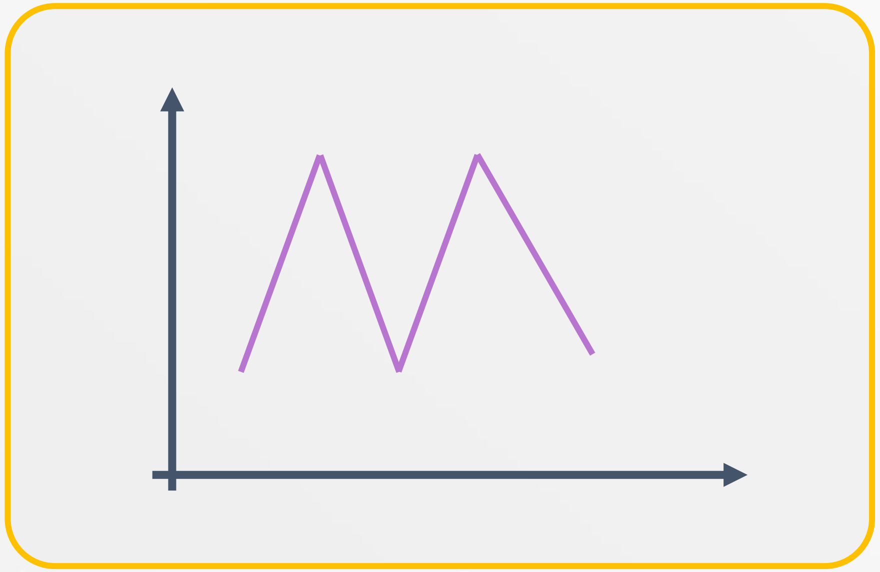
Neural Network Interpretation





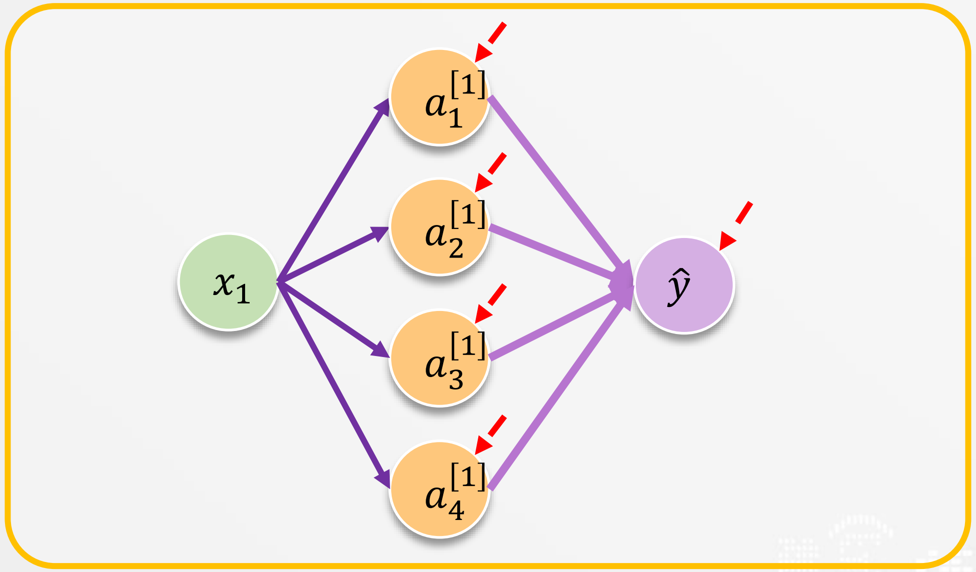
Neural Network Interpretation

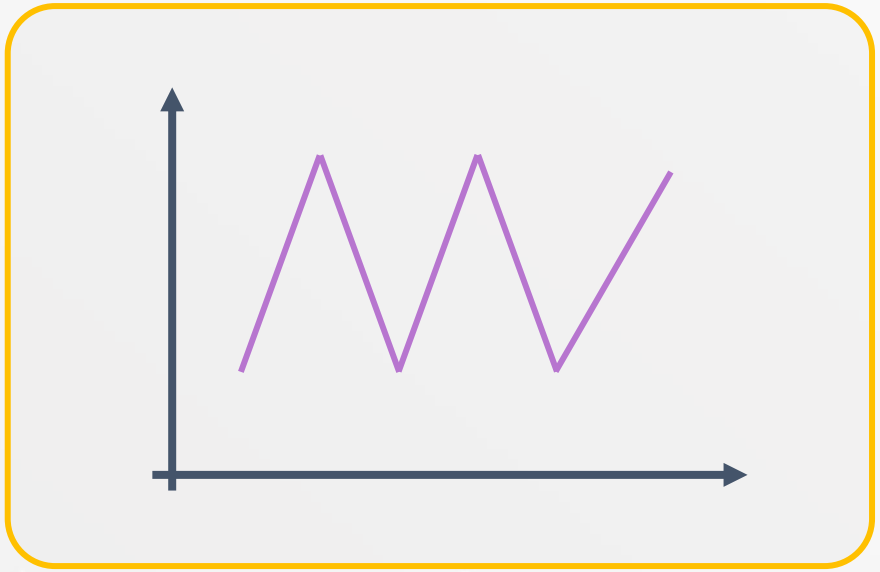




TAUTOLOGY

Neural Network Interpretation





Neural Network Interpretation

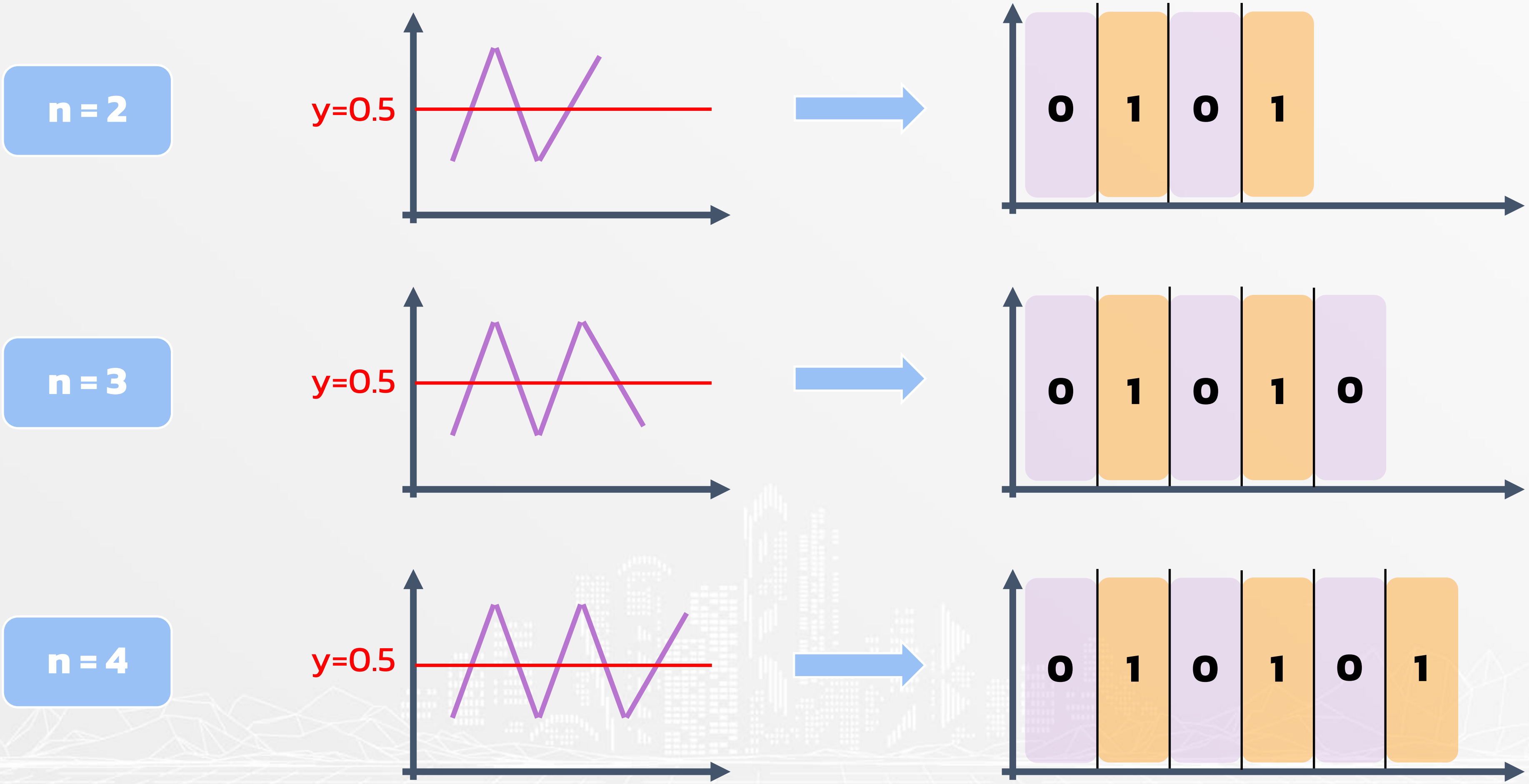
Classification

เมื่อเรานำชั้น output เหล่านั้น มาผ่าน sigmoid function เราจะได้ว่า

" ค่าของ **output มากกว่า 0.5** model จะพยากรณ์ผลลัพท์ เป็น **class 1** "

ดังนั้น เราจะใช้ y = 0.5 เพื่อใช้ในการแบ่ง region ของการพยากรณ์

Neural Network Interpretation



Neural Network Interpretation

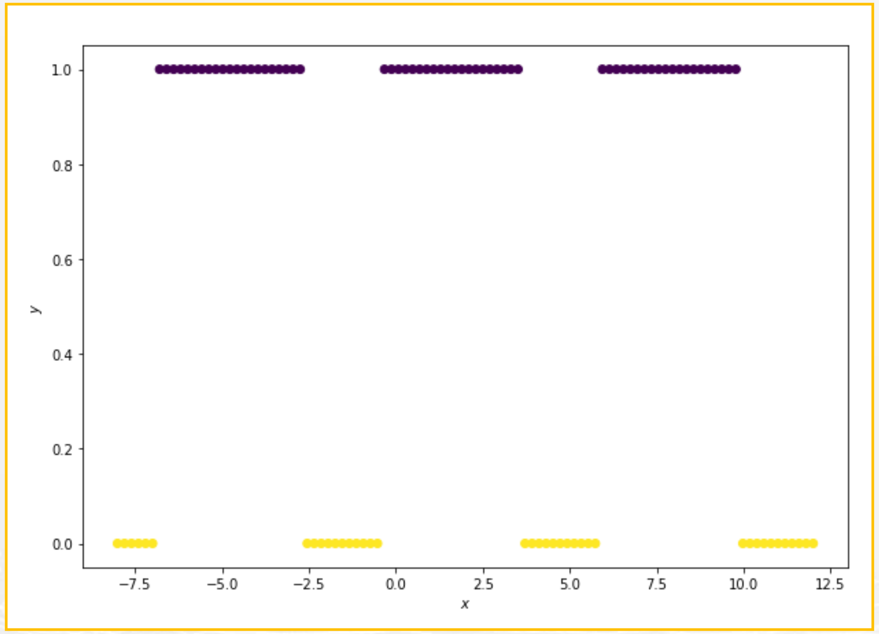
Classification

จากภาพเราจะเห็นว่า region ในการแบ่ง class ของ binary ใน 2 มิติ (ตัวแปรต้น) จะมีค่าสูงสุดได้เท่ากับ n + 2 เมื่อ n คือ จำนวน node ใน hidden layer

Neural Network Interpretation

Classification

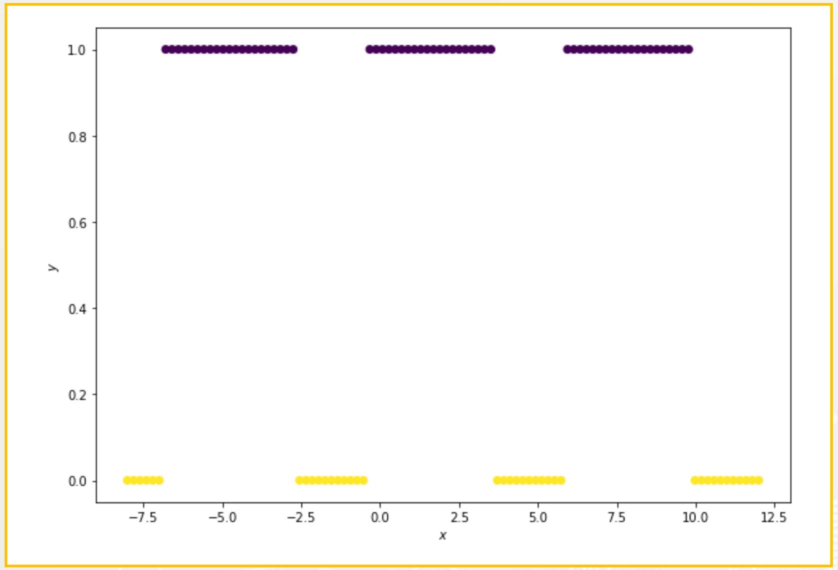
Example 2 : Binary Classification

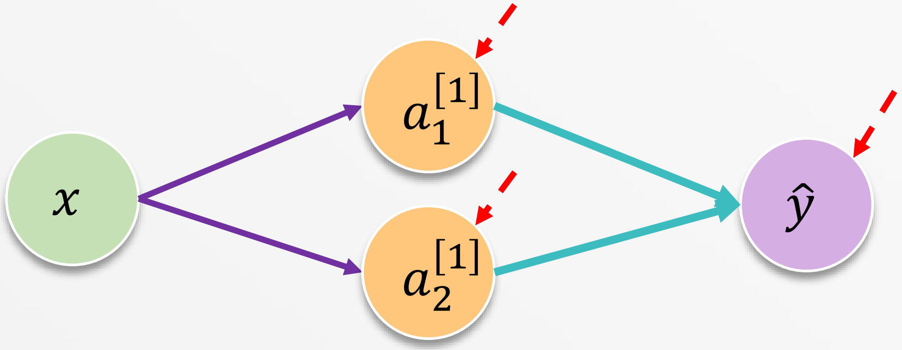


Neural Network Interpretation

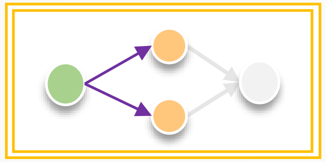
Classification

Example 2.1



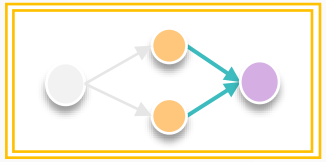


Neural Network Interpretation

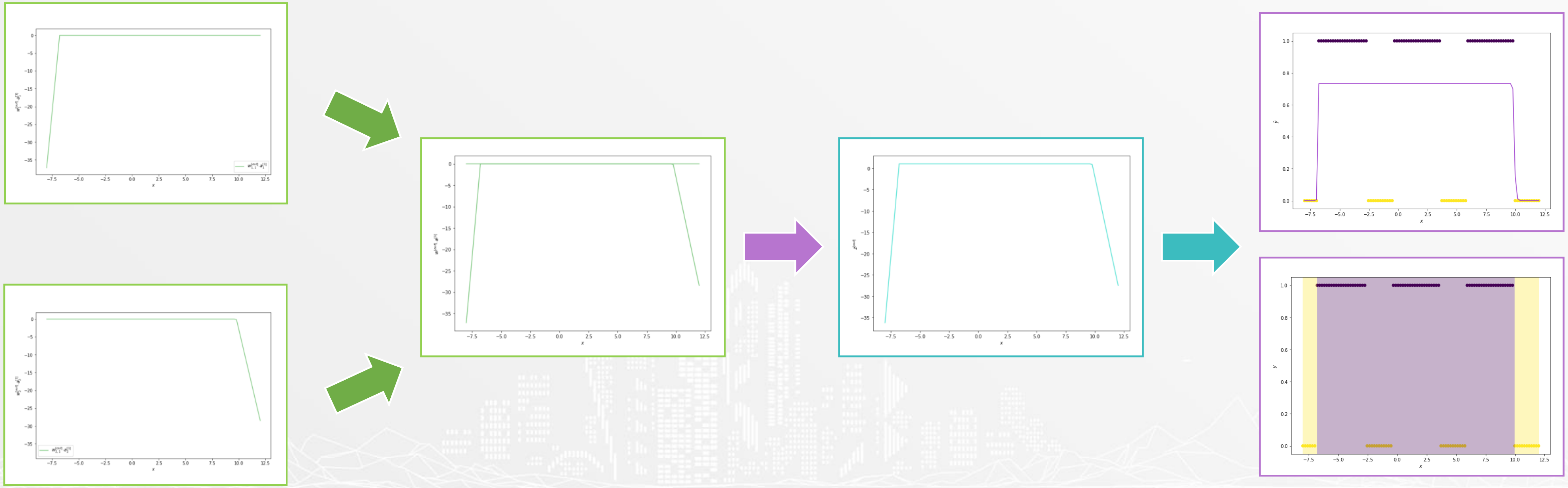


Classification | **Example 2.1**

Neural Network Interpretation



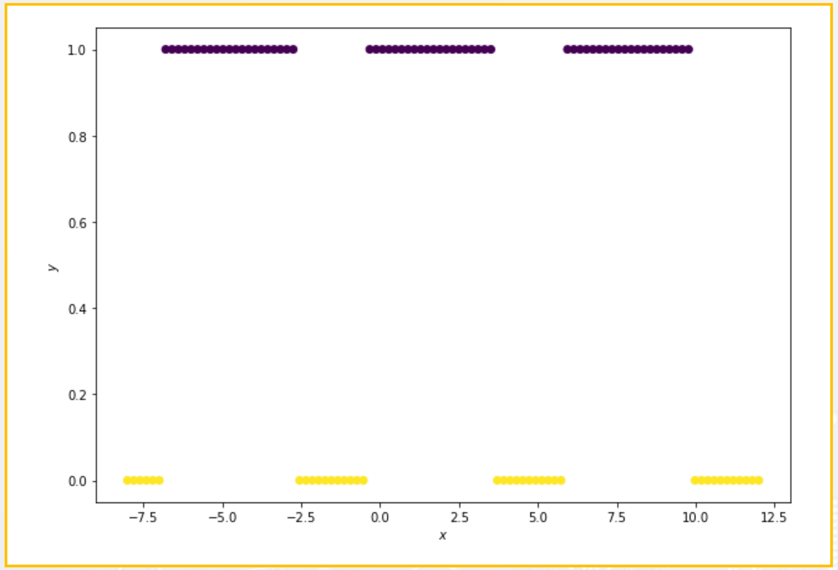
Classification **Example 2.1**

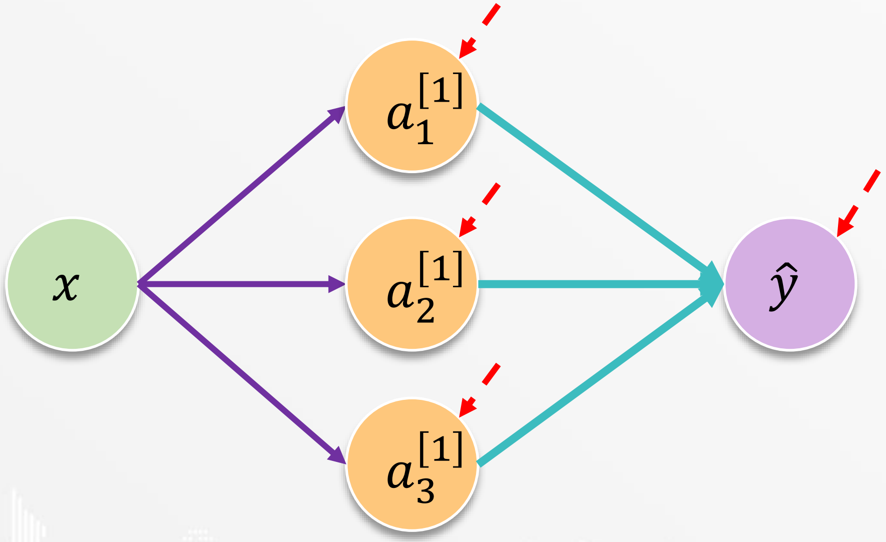


Neural Network Interpretation

Classification

Example 2.2

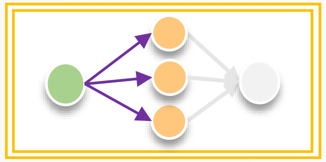


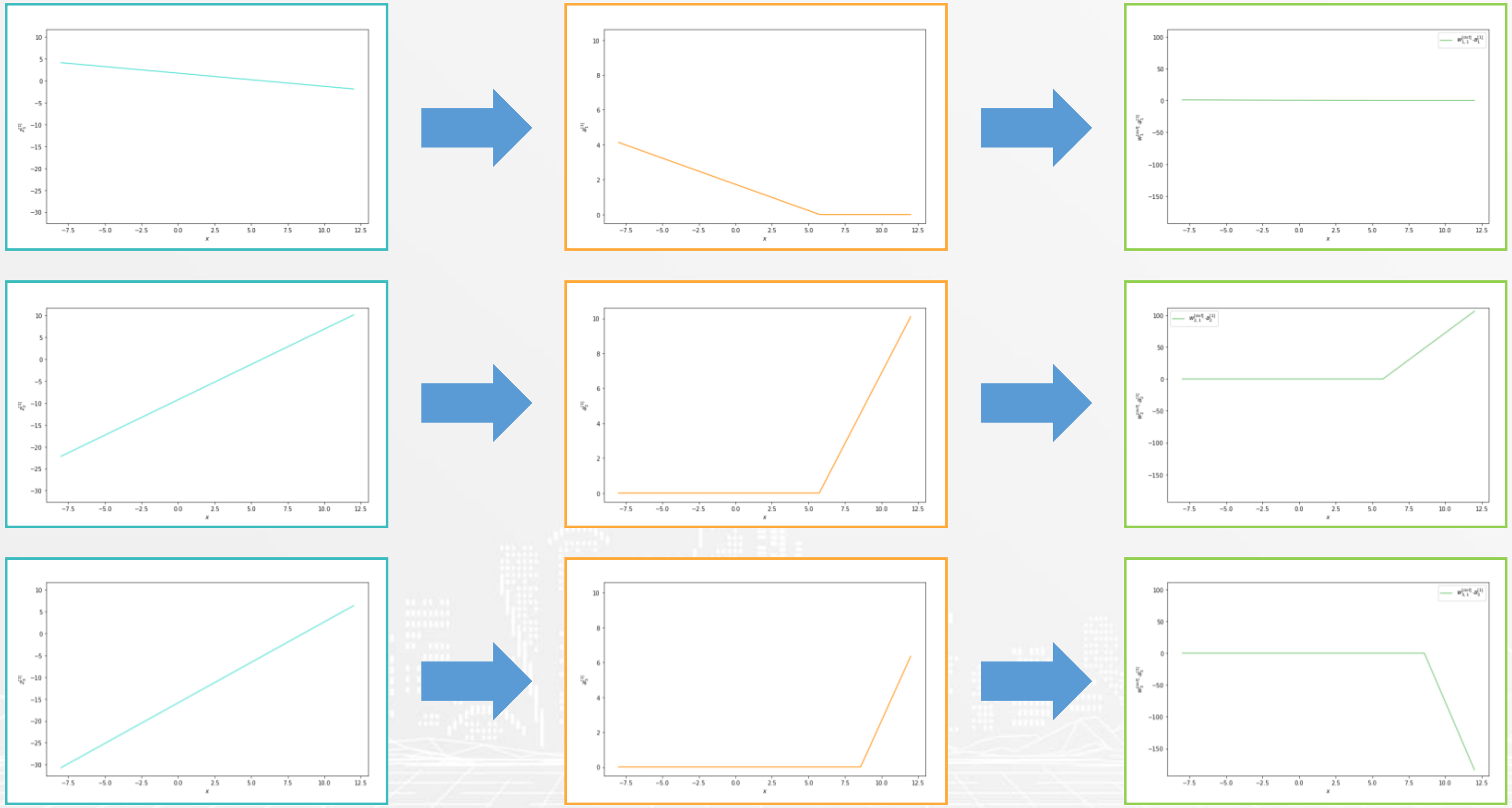


Neural Network Interpretation

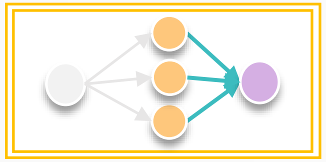
Classification

Example 2.2

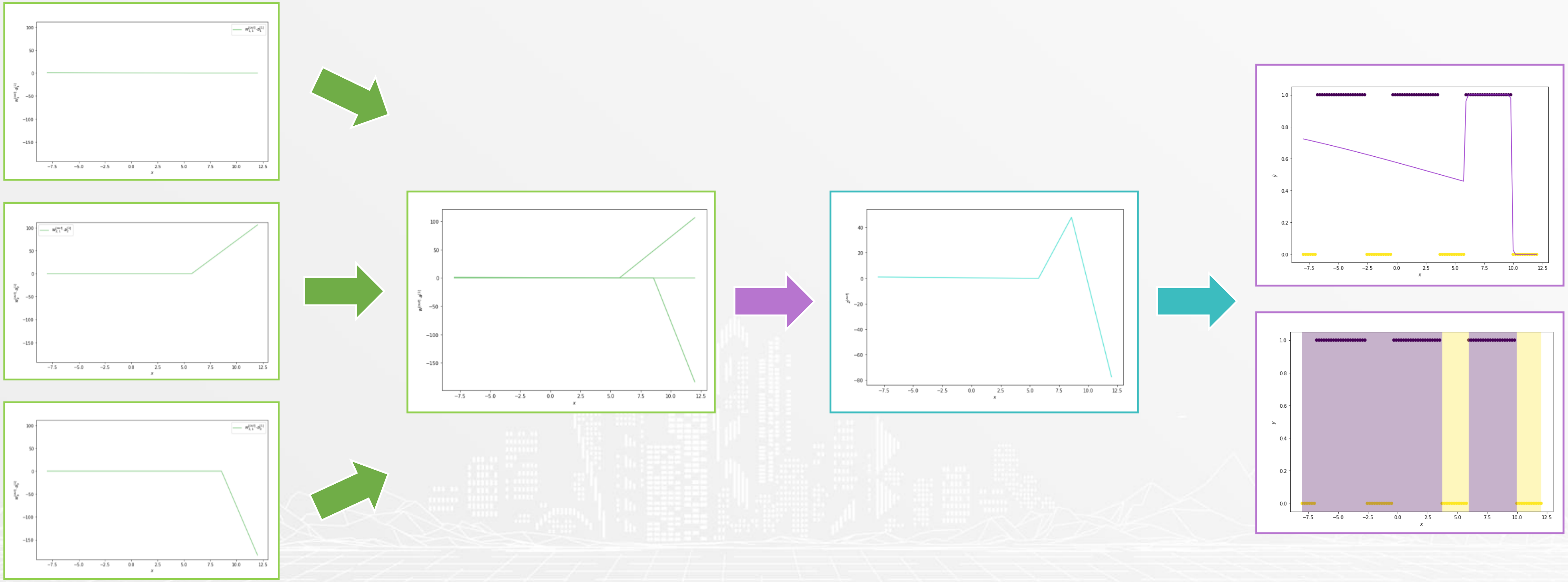




Neural Network Interpretation



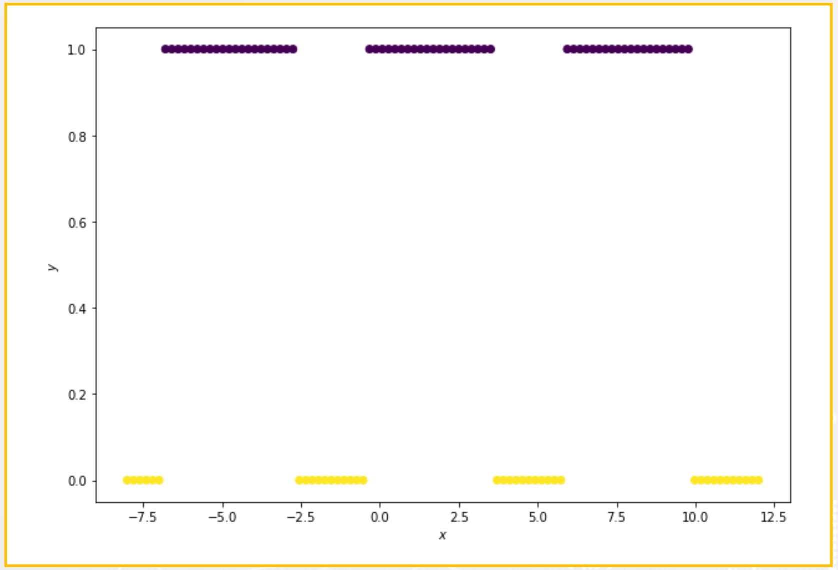
Classification | **Example 2.2**

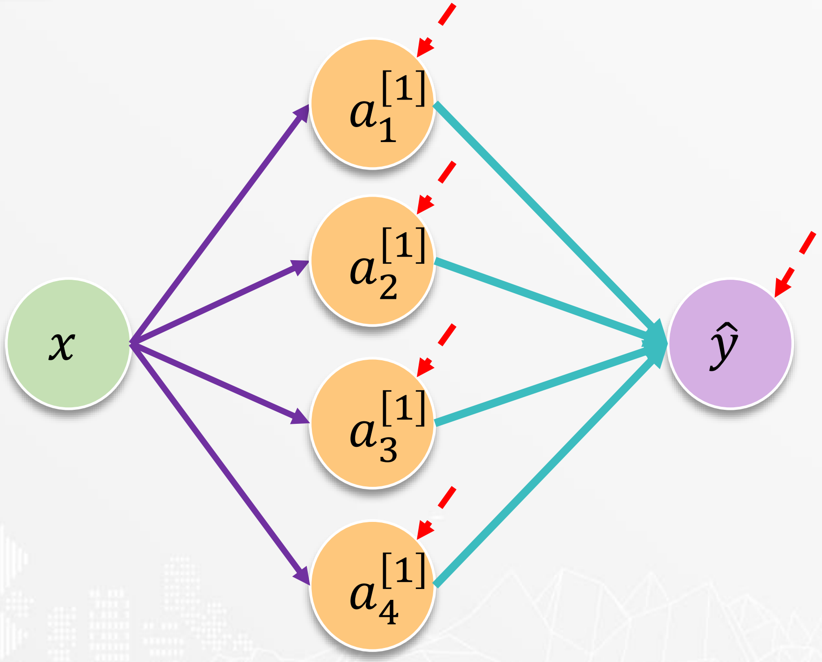


Neural Network Interpretation

Classification

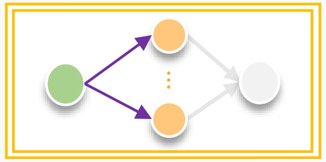
Example 2.3



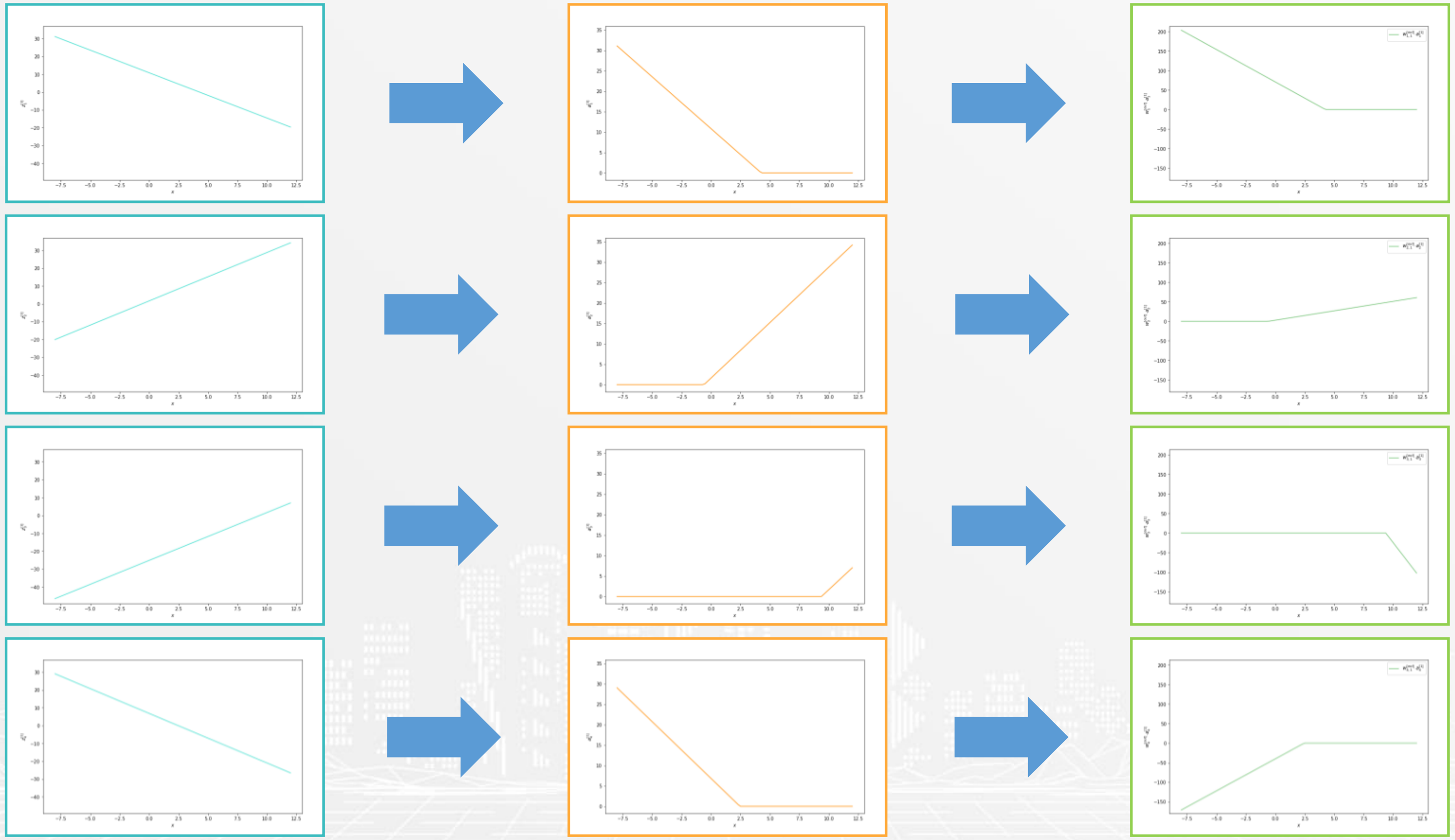


TAUTOLOGY

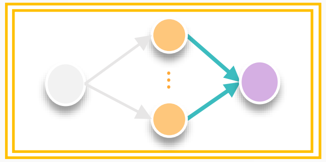
Neural Network Interpretation



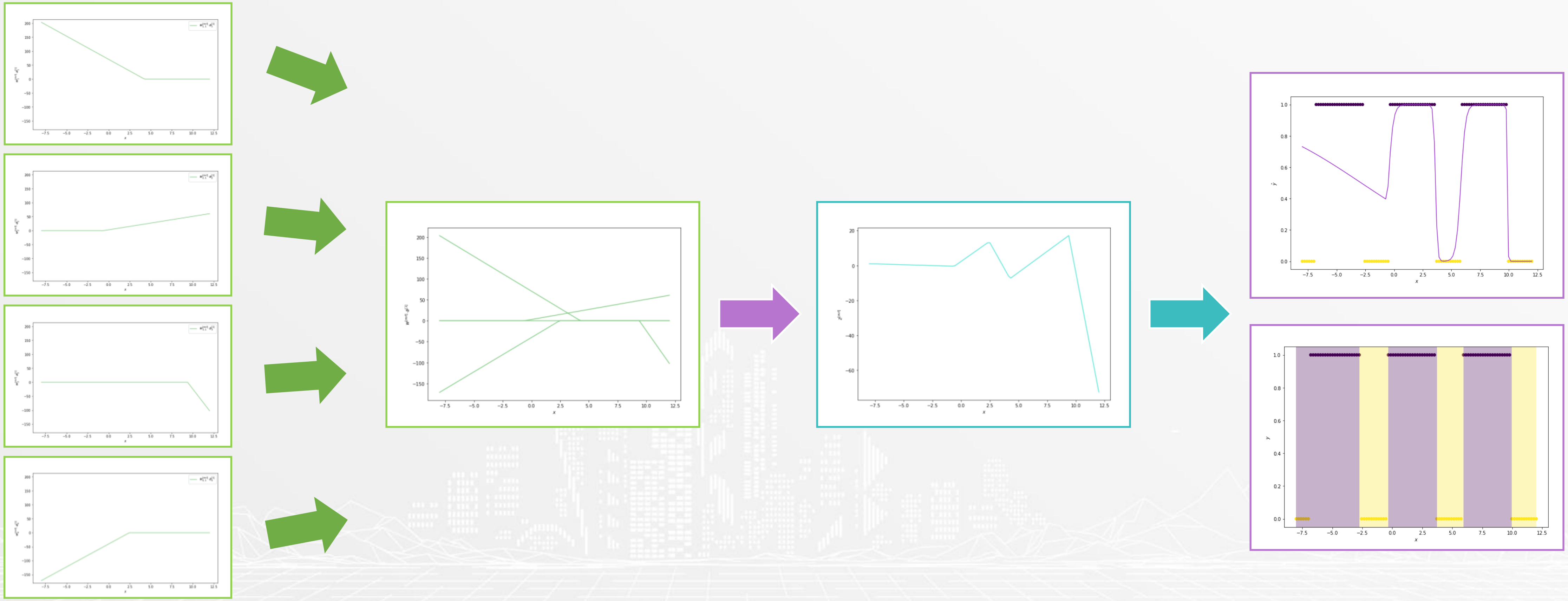
Classification | **Example 2.3**



Neural Network Interpretation



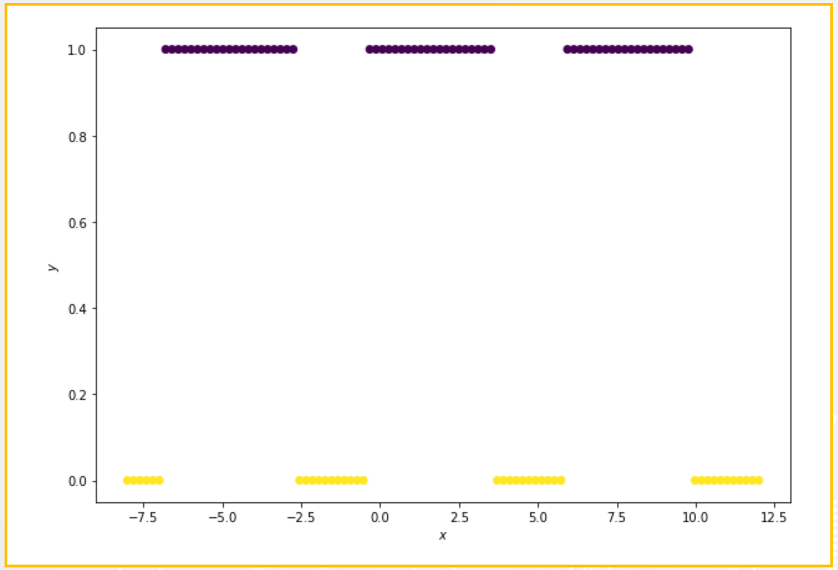
Classification **Example 2.3**

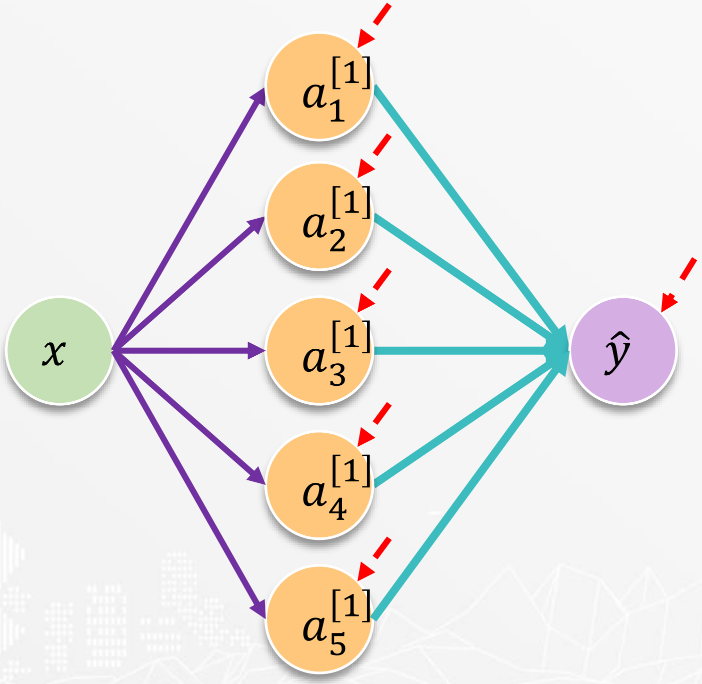


Neural Network Interpretation

Classification

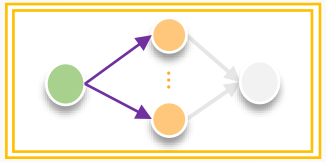
Example 2.4





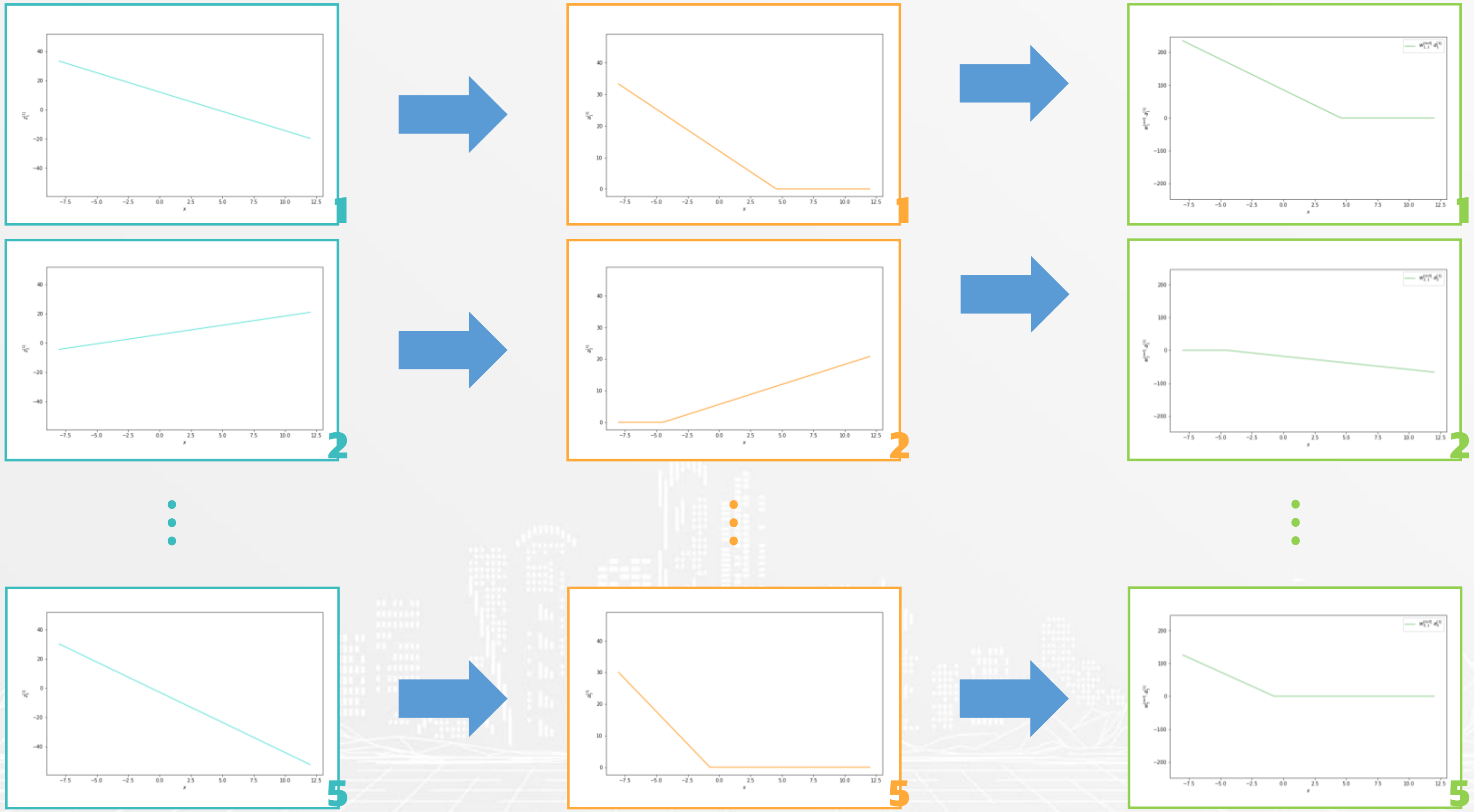
TAUTOLOGY

Neural Network Interpretation



Classification

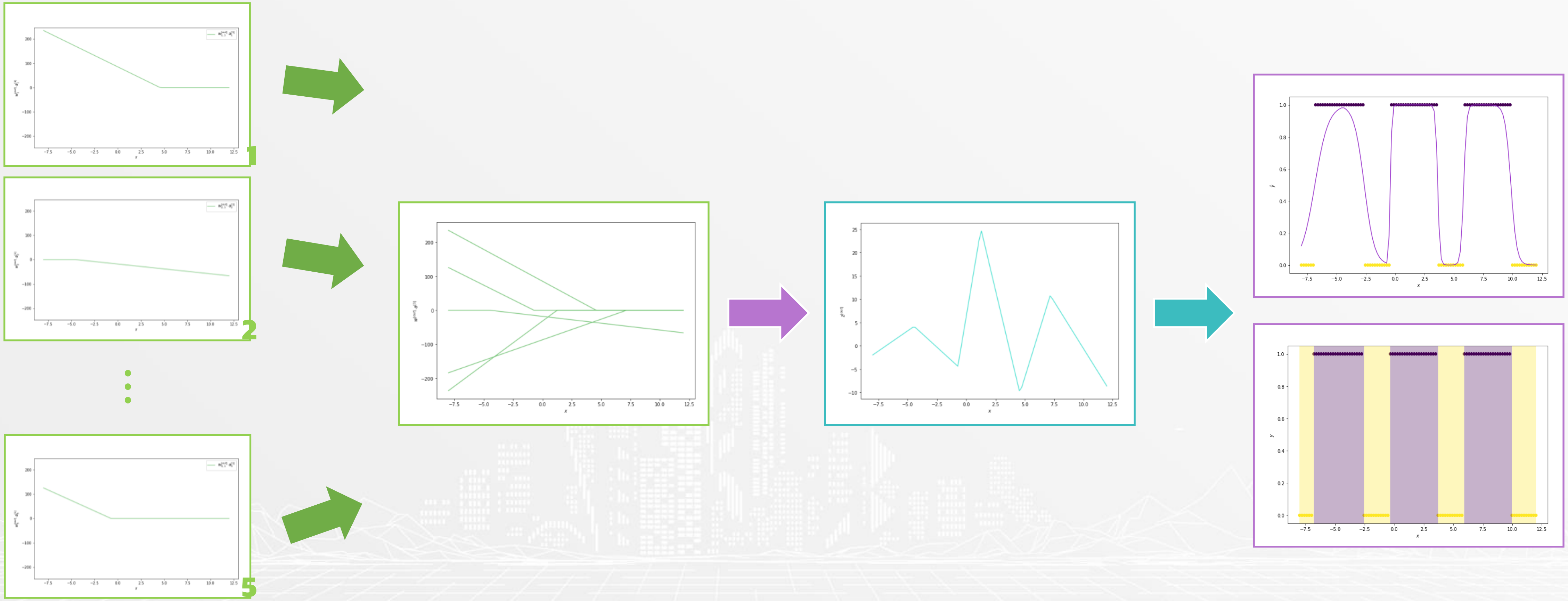
Example 2.4



Neural Network Interpretation

Classification

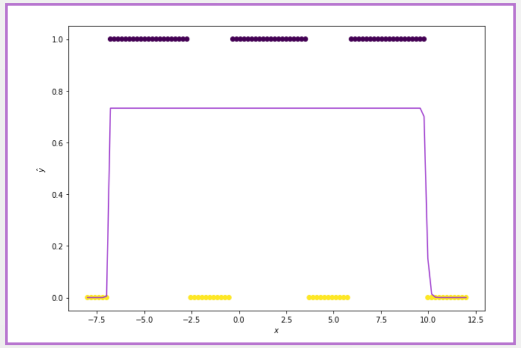
Example 2.4



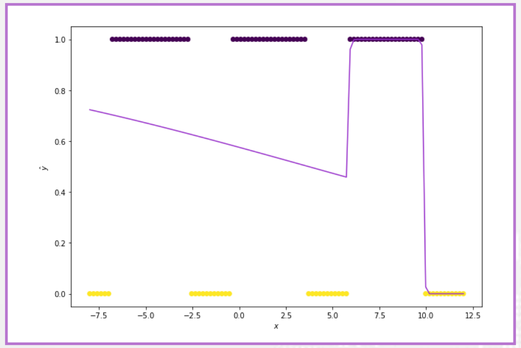
Neural Network Interpretation

Classification

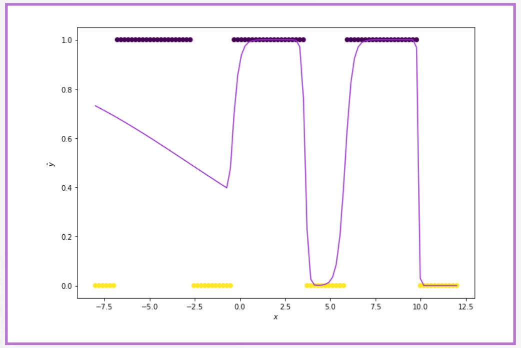
กราฟแสดงค่า predicted ของ Example 2



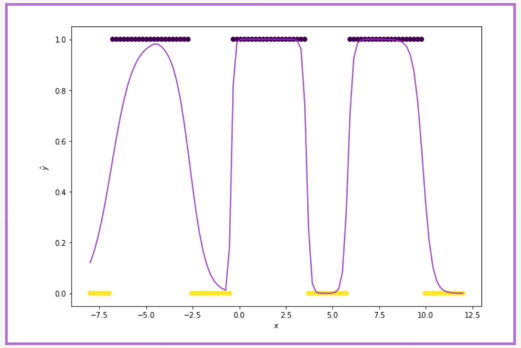
Example 2.1 (1,2,1)



Example 2.2 (1,3,1)



Example 2.3 (1,4,1)

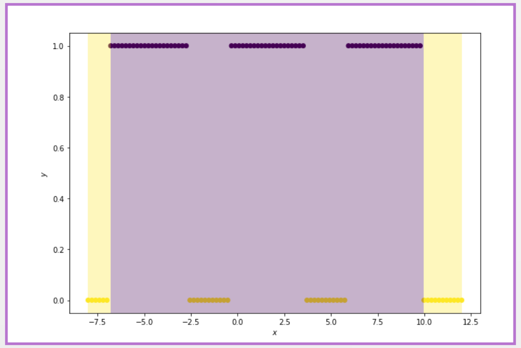


Example 2.4 (1,5,1)

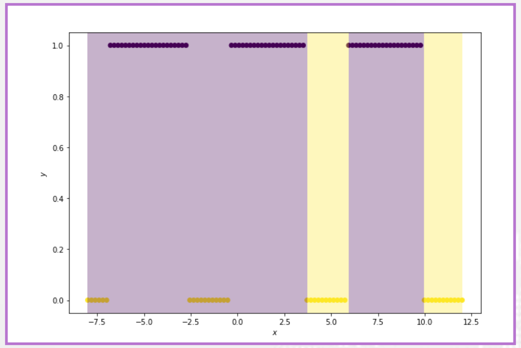
Neural Network Interpretation

Classification

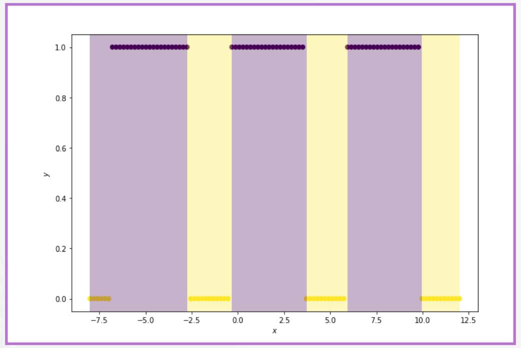
กราฟแสดงพื้นที่ predicted ของ Example 2



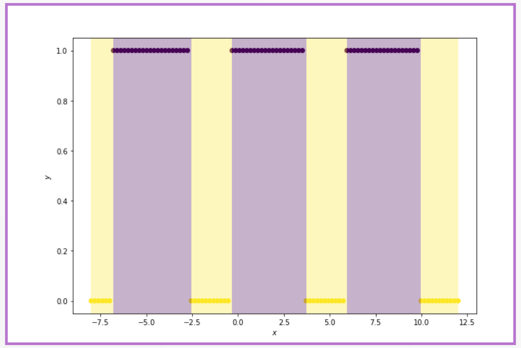
Example 2.1 (1,2,1)



Example 2.2 (1,3,1)



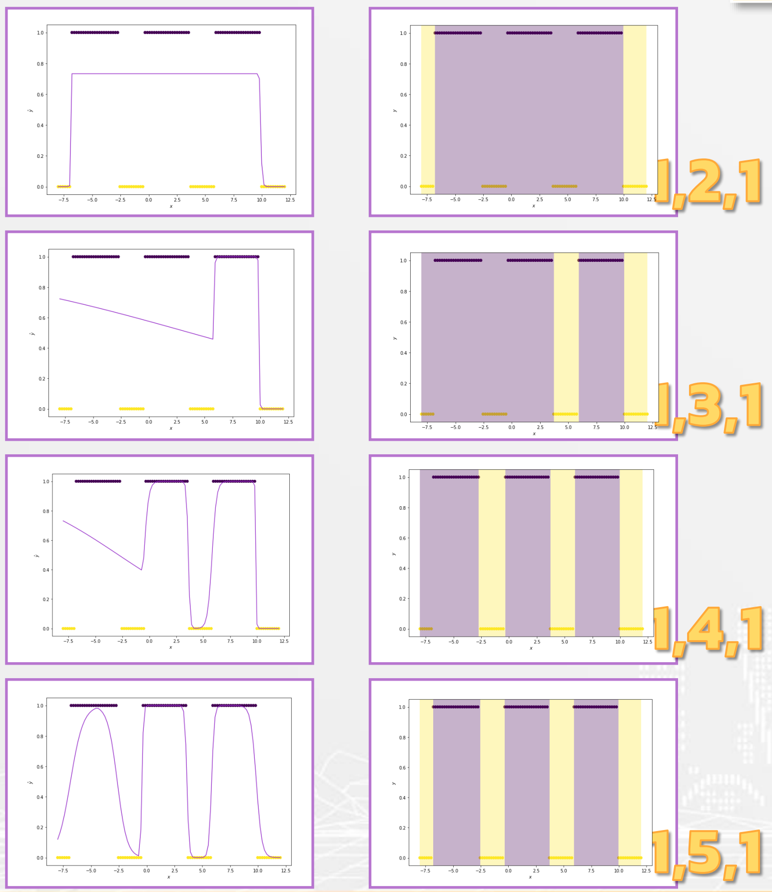
Example 2.3 (1,4,1)



Example 2.4 (1,5,1)

Neural Network Interpretation

Classification



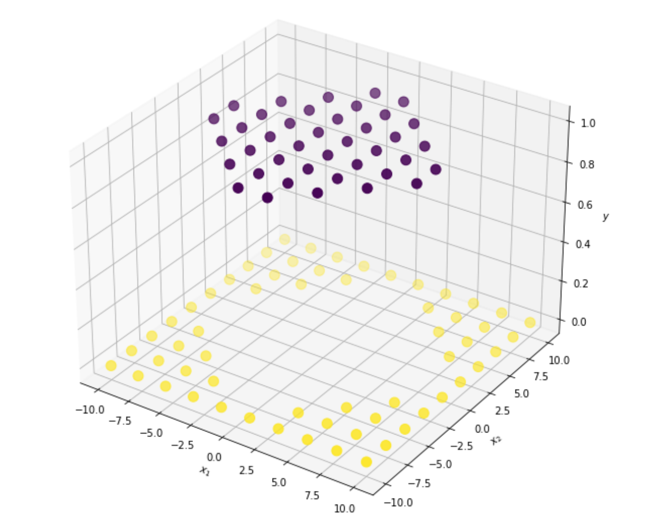
ข้อสรุปของ Example 2

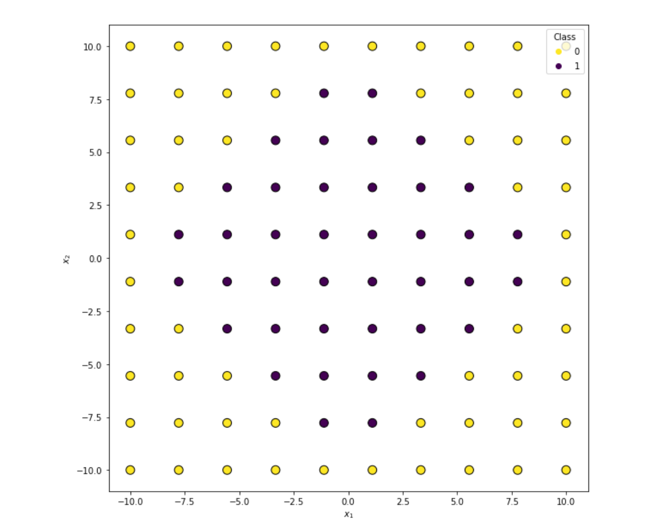
- จำนวน region ที่เกิดขึ้นใน 2 มิติ (จำนวนตัวแปรต้น 1 ตัว) เท่ากับ จำนวน node + 2 ทั้งนี้การที่จะเกิด maximum region นั้นขึ้นอยู่กับ dataset และ initial parameter ด้วย

Neural Network Interpretation

Classification

Example 3 : Binary Classification

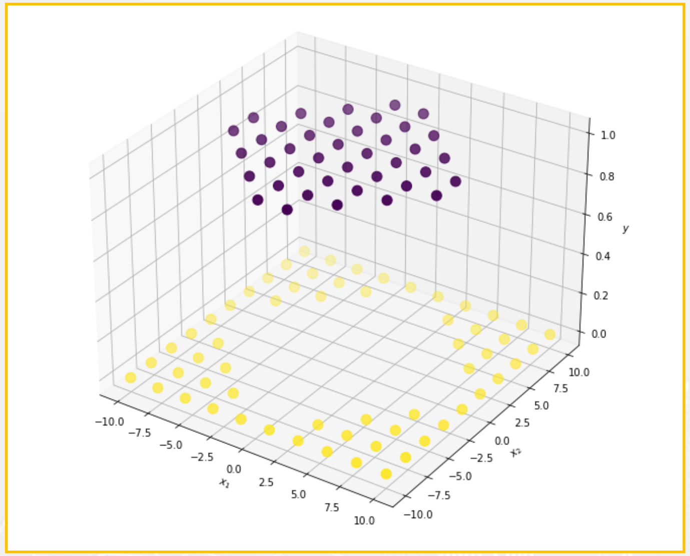


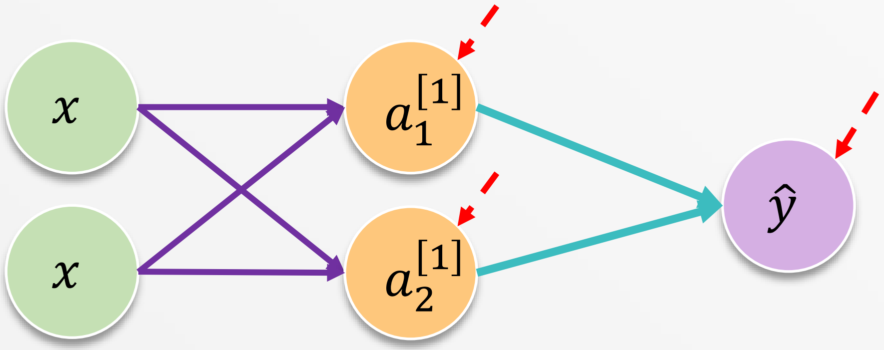


Neural Network Interpretation

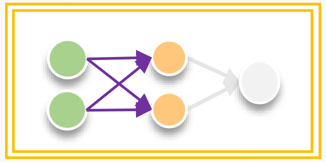
Classification

Example 3.1

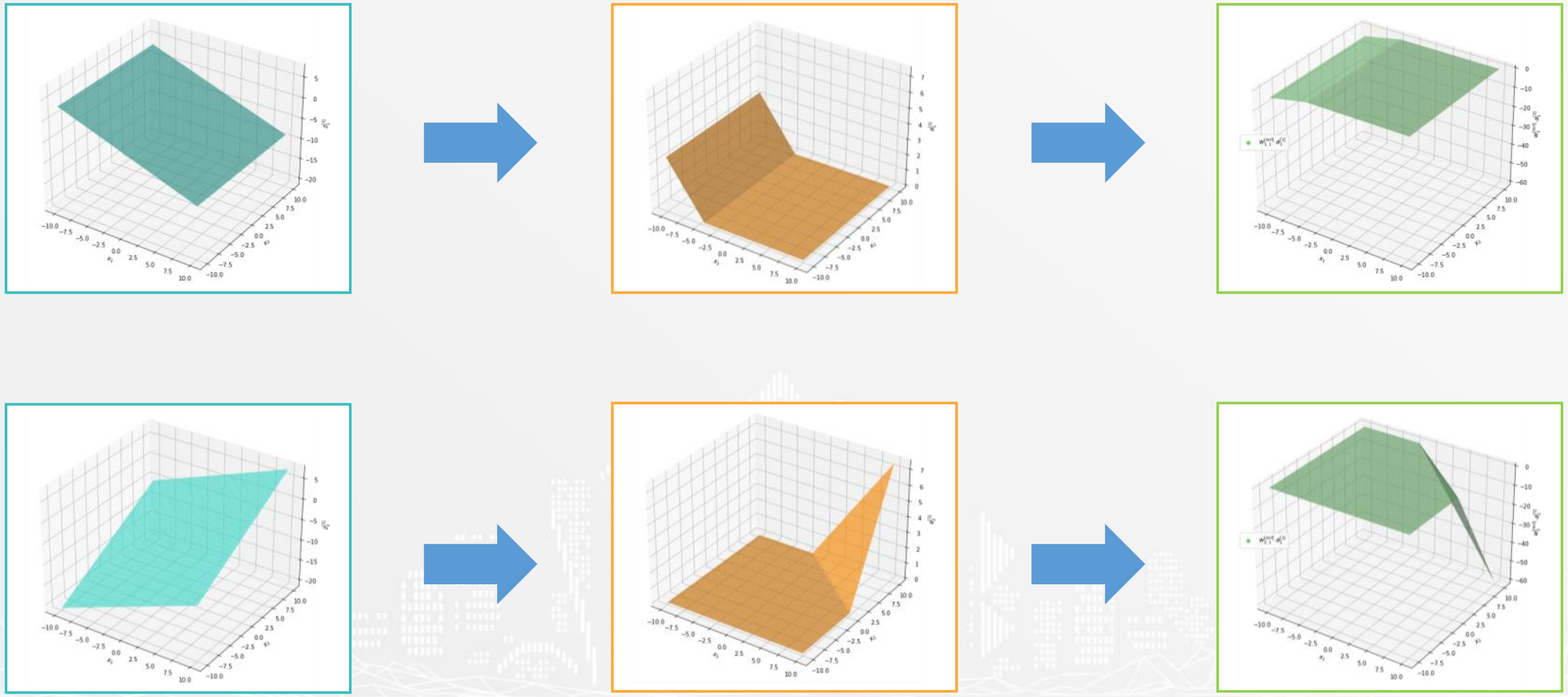




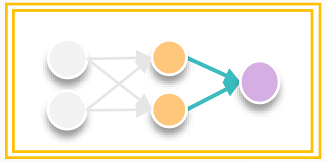
Neural Network Interpretation



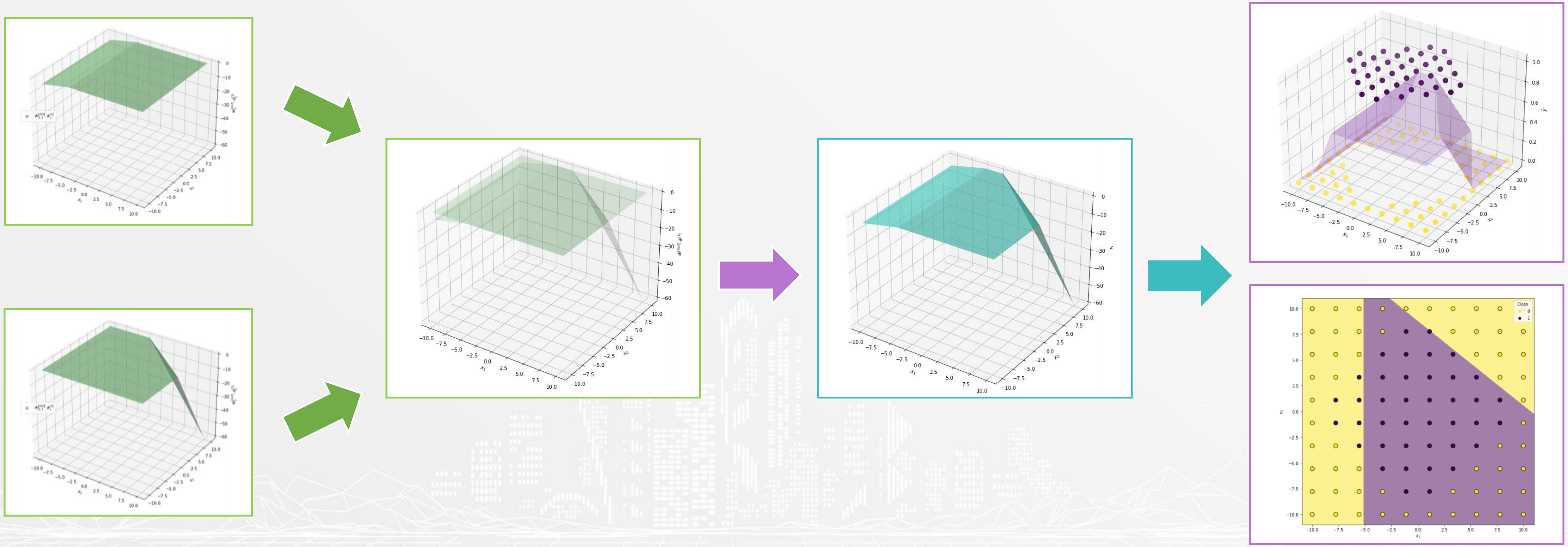
Classification | **Example 3.1**



Neural Network Interpretation



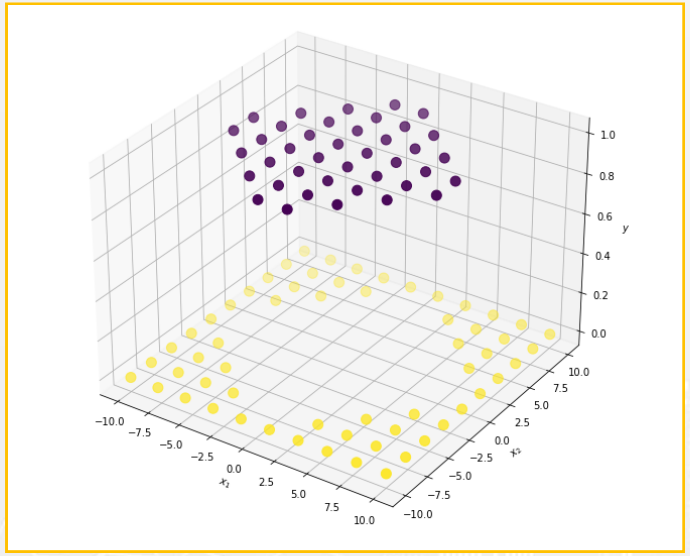
Classification | **Example 3.1**

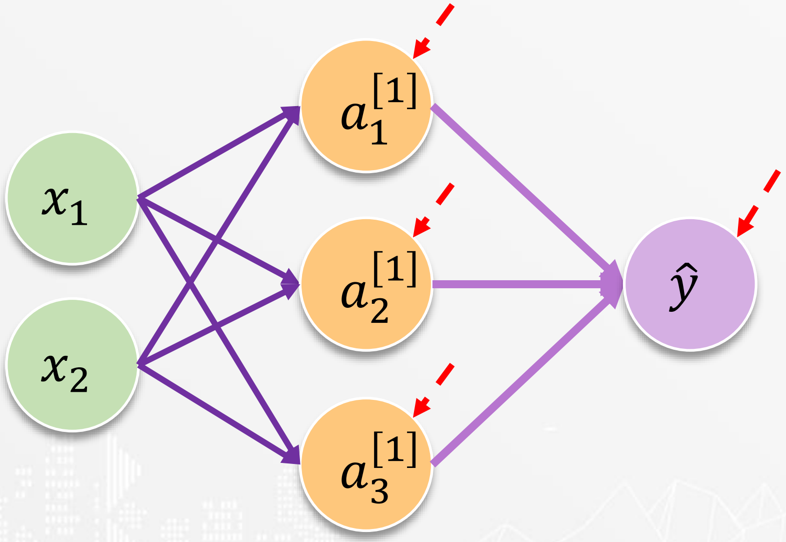


Neural Network Interpretation

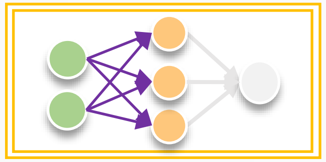
Classification

Example 3.2

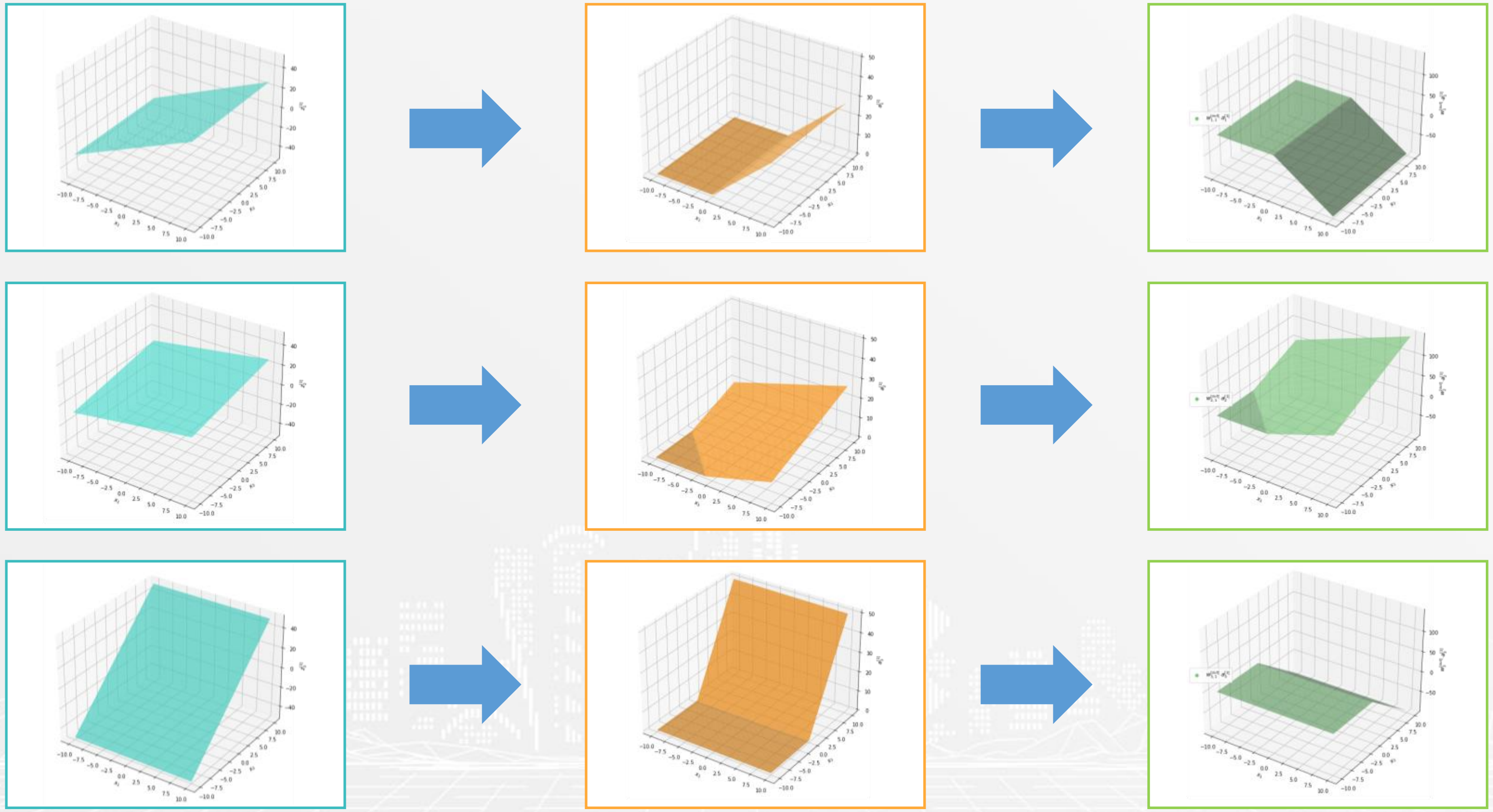




Neural Network Interpretation

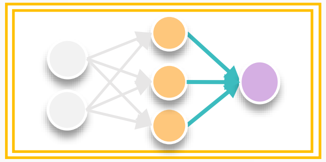


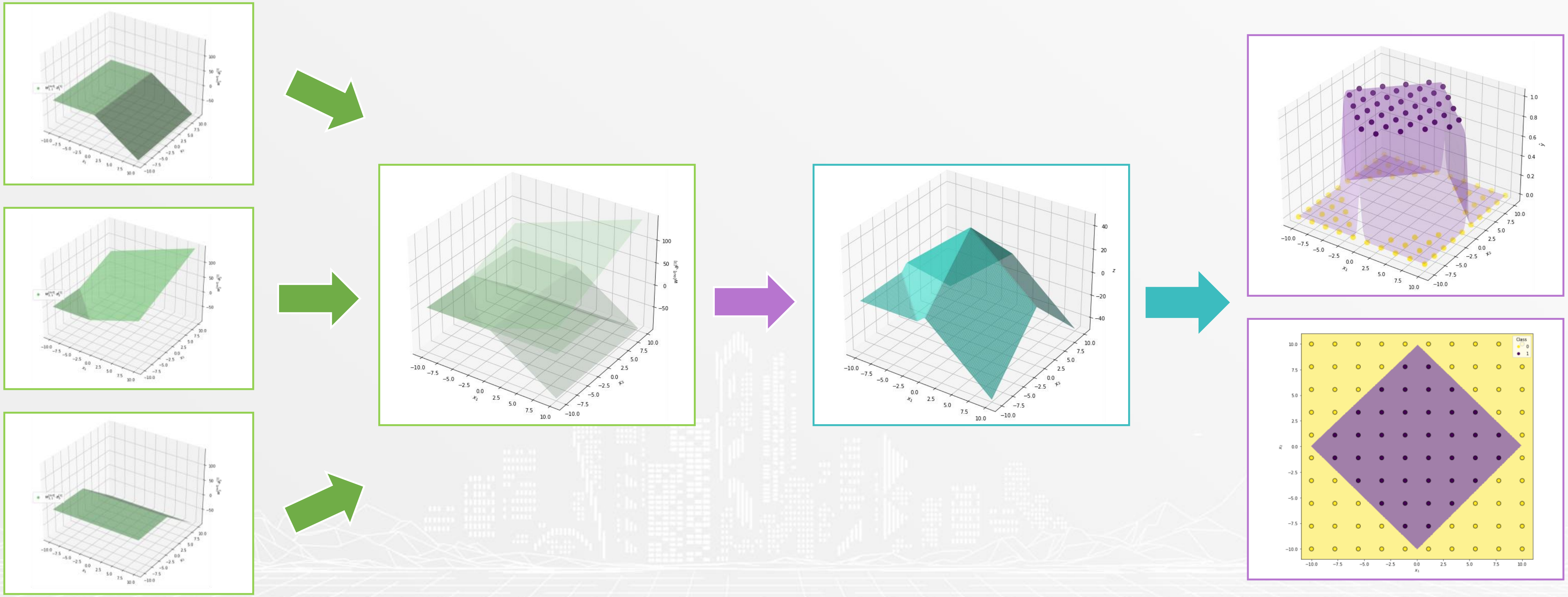
Classification **Example 3.2**



Neural Network Interpretation

Classification **Example 3.2**

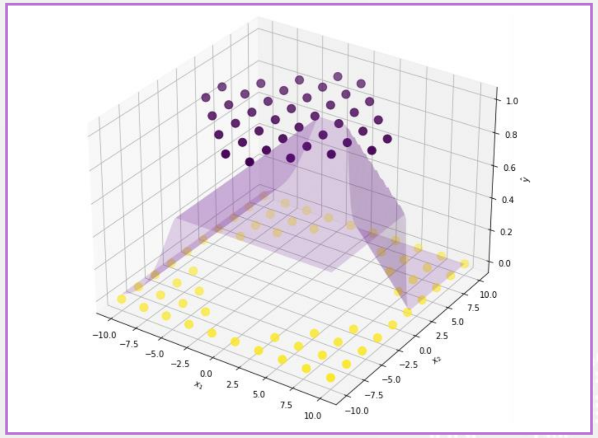




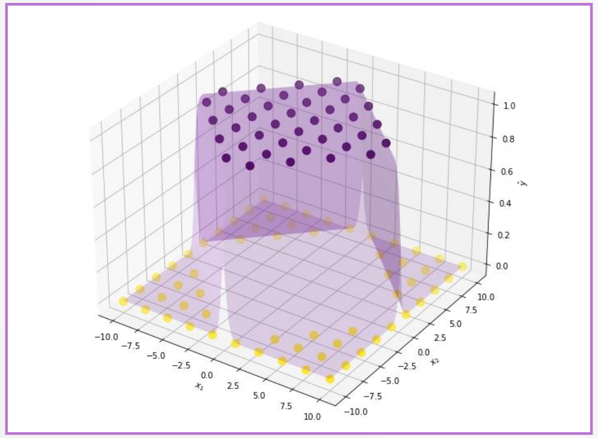
Neural Network Interpretation

Classification

กราฟแสดงค่า predicted ของ Example 3



Example 3.1 (2,2,1)

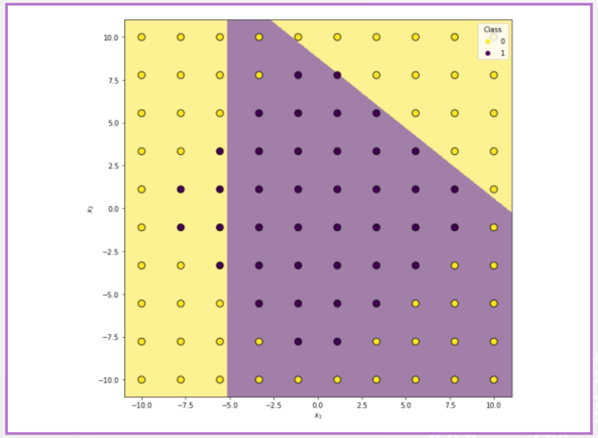


Example 3.2 (2,3,1)

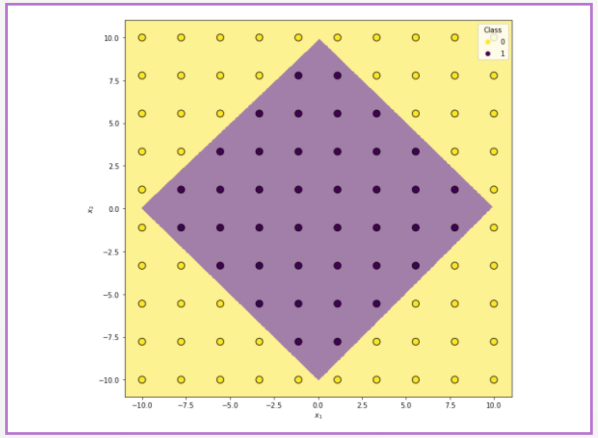
Neural Network Interpretation

Classification

กราฟแสดงพื้นที่ predicted ของ Example 3



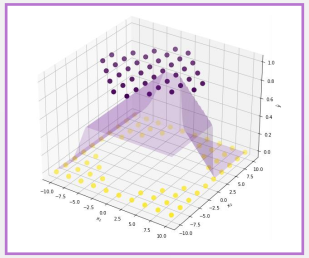
Example 3.1 (2,2,1)

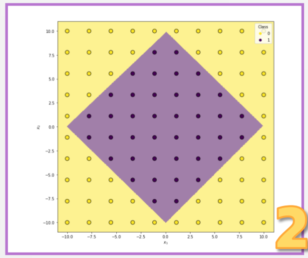


Example 3.2 (2,3,1)

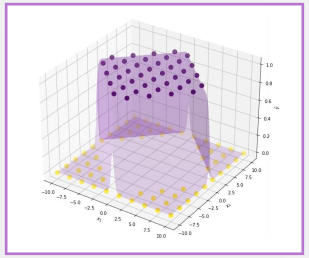
Neural Network Interpretation

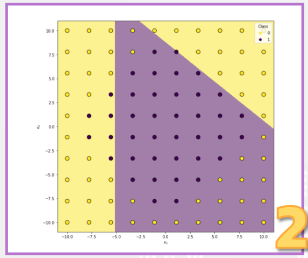
Classification





2,2,1





2,3,1

ข้อสรุปของ Example 3

- Plane (ส่วนของพื้นผิว) ที่เพิ่มขึ้นจากจำนวน node ที่มากขึ้น เมื่อผ่าน sigmoid function แล้วจะทำให้เกิด region ของการแบ่ง class ที่เพิ่มขึ้นตามไปด้วย

Neural Network Interpretation

จากความรู้เดิมใน regression 3 มิติ เราได้ว่า จำนวน plane (ส่วนของพื้นผิว) ที่มากที่สุดที่เกิดขึ้นได้ในชั้น output คือ

“ $(n^2 + n + 2) / 2$ ”

Neural Network Interpretation

ทำให้เราได้ว่า เส้นแบ่งขอบเขต (border) ที่ใช้ในการแบ่ง region จะมีจำนวนมากที่สุดก็จะมีเท่ากับ **“ $(n^2 + n + 2) / 2$ ”** ด้วยเช่นกัน

Neural Network Interpretation

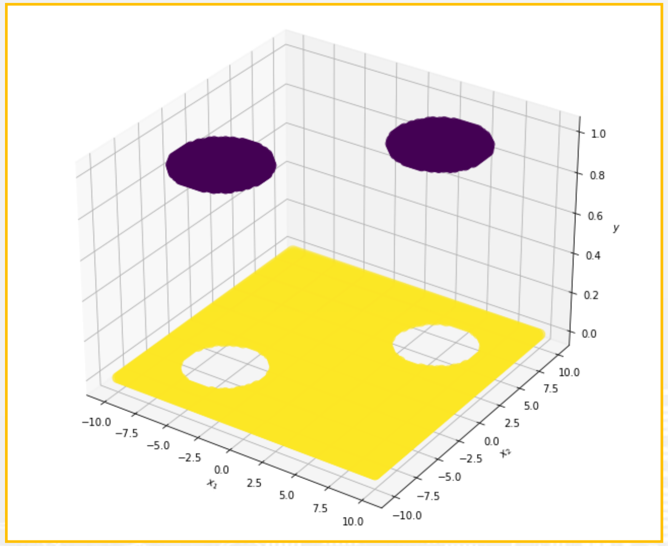
แต่ในการหาจำนวน maximum region ที่เกิดขึ้นใน 3 มิติ

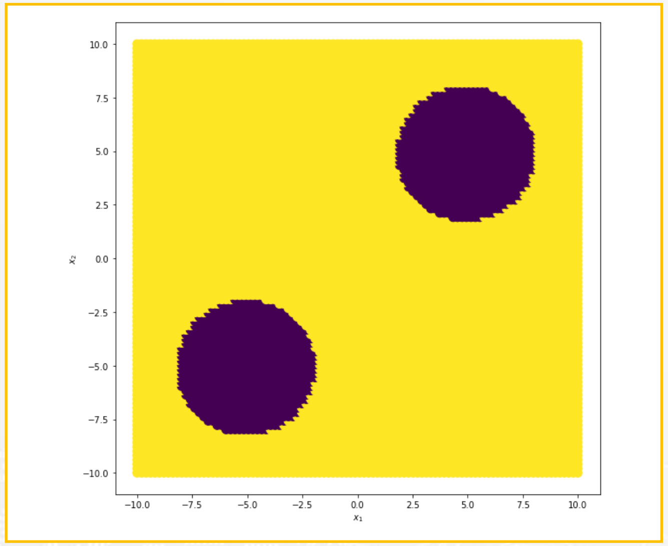
มีความซับซ้อนในการวิเคราะห์การประกอบกันของ plane
ที่ใช้ในการสร้าง region จึงขอละไว้

Neural Network Interpretation

Classification

Example 4 : Binary Classification

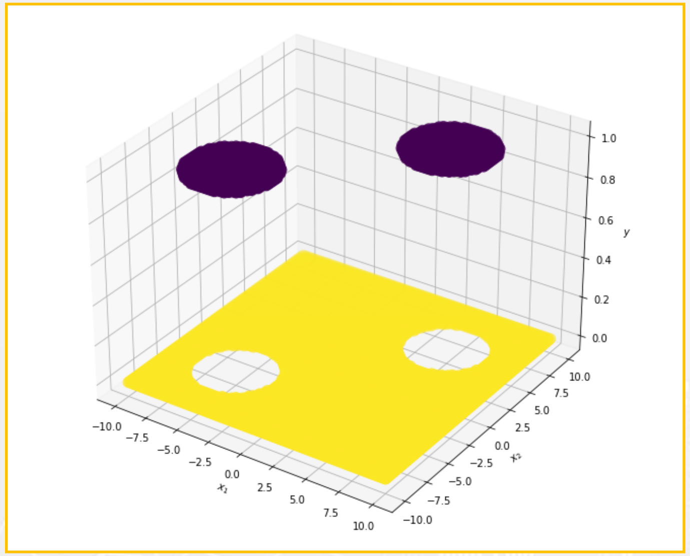


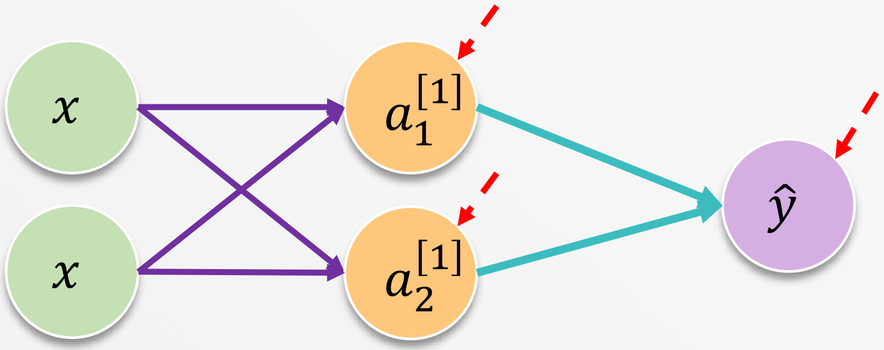


Neural Network Interpretation

Classification

Example 4.1

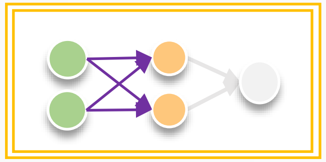


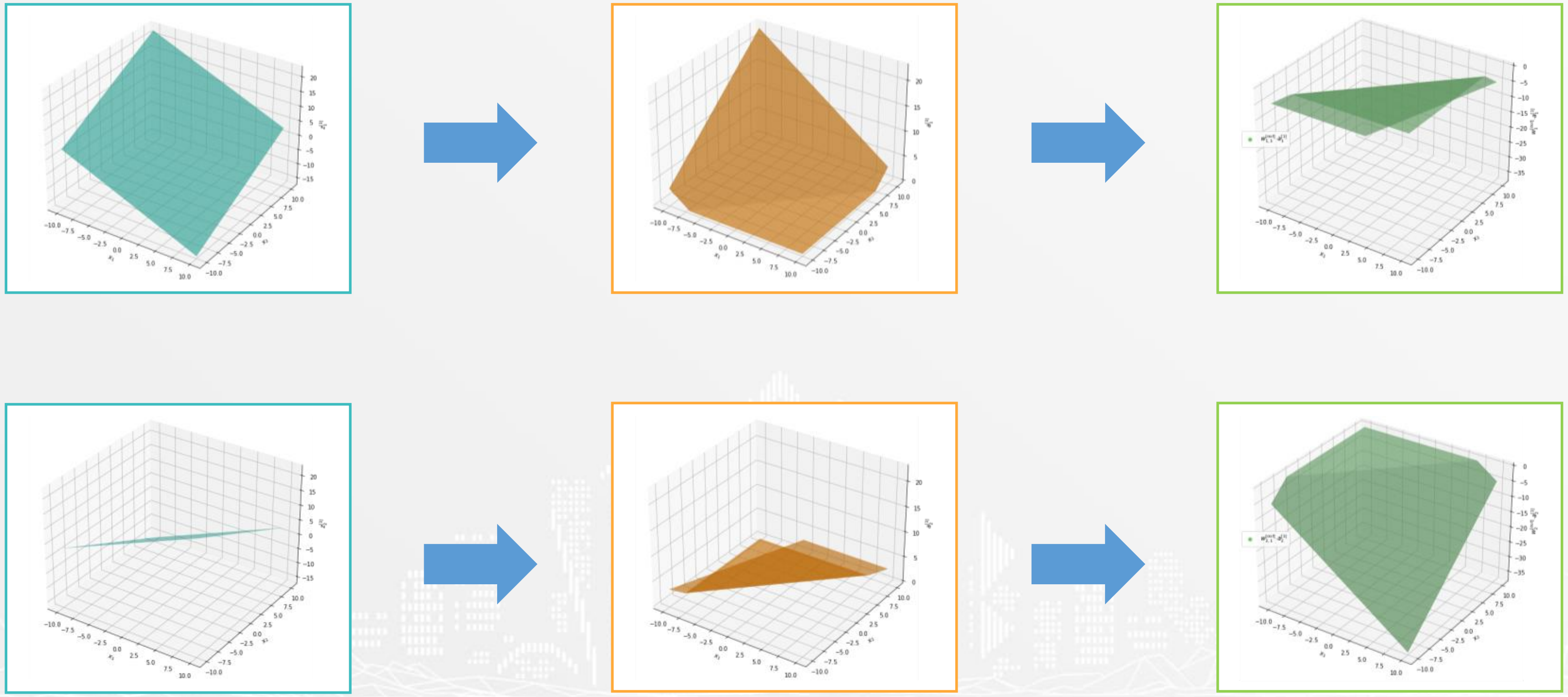


Neural Network Interpretation

Classification

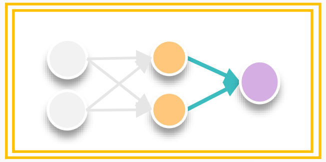
Example 4.1

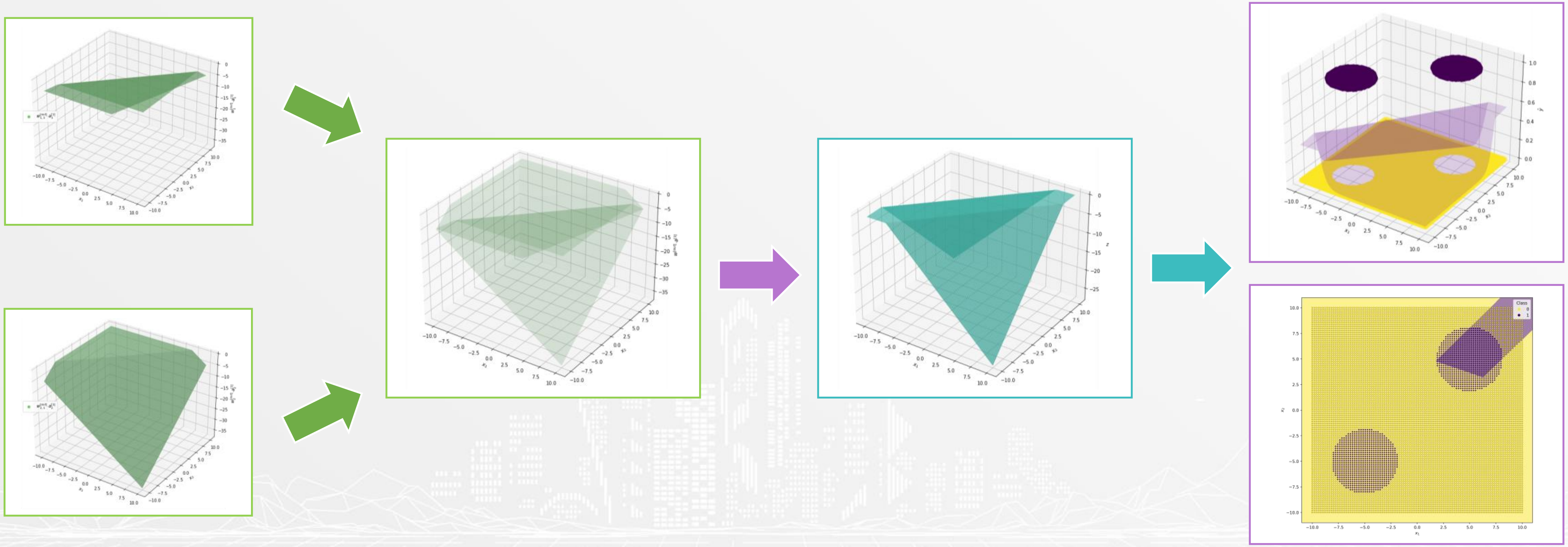




Neural Network Interpretation

Classification **Example 4.1**

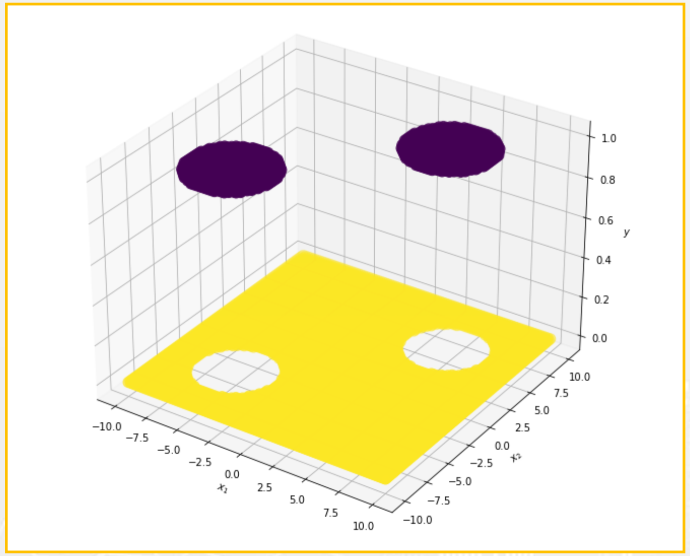


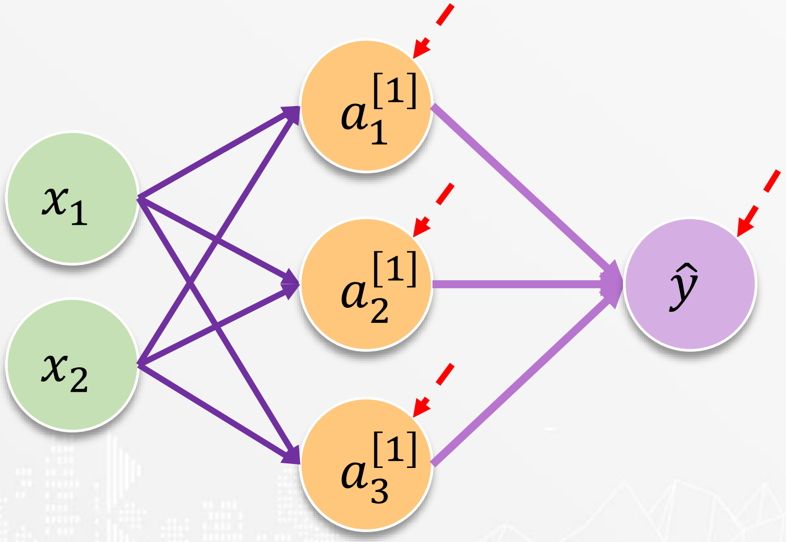


Neural Network Interpretation

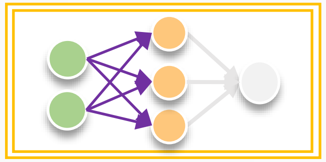
Classification

Example 4.2

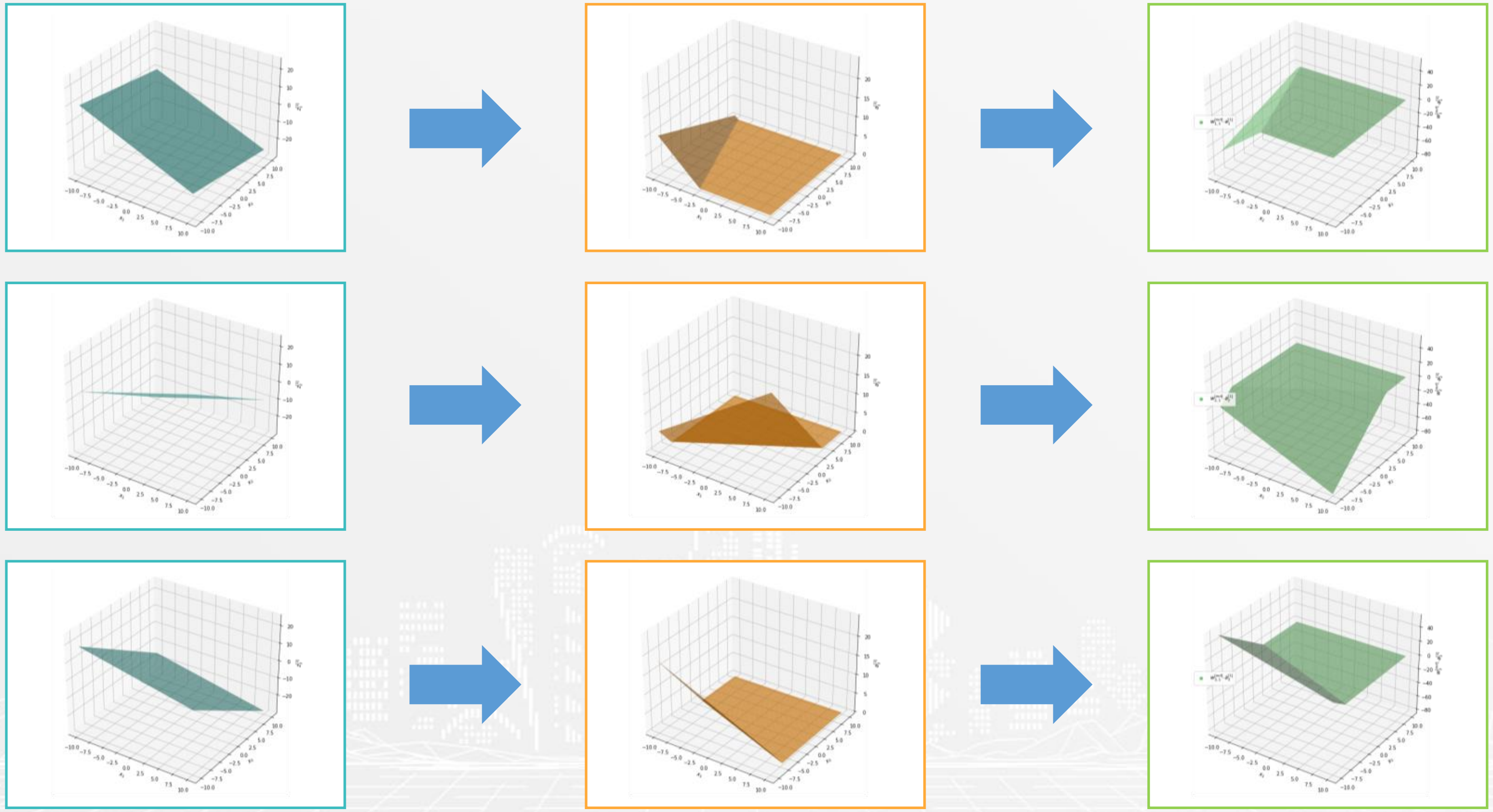




Neural Network Interpretation

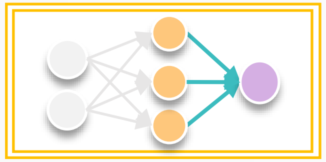


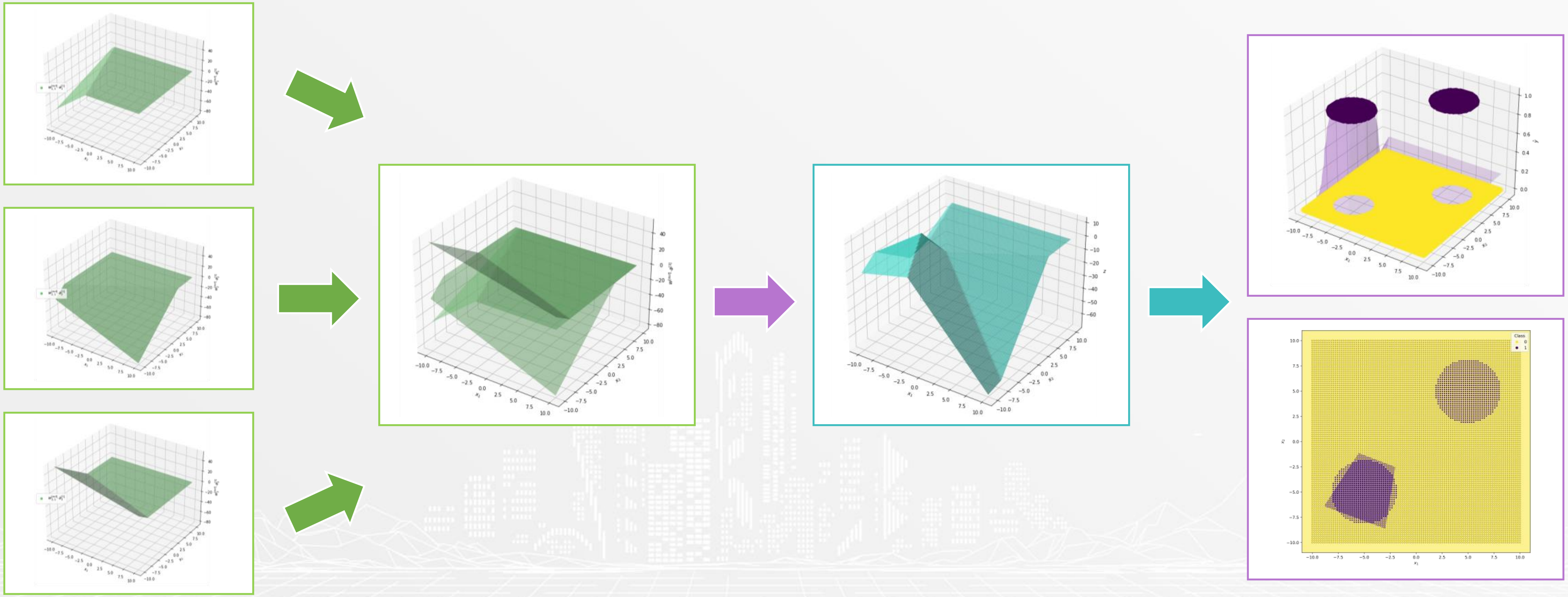
Classification **Example 4.2**



Neural Network Interpretation

Classification **Example 4.2**

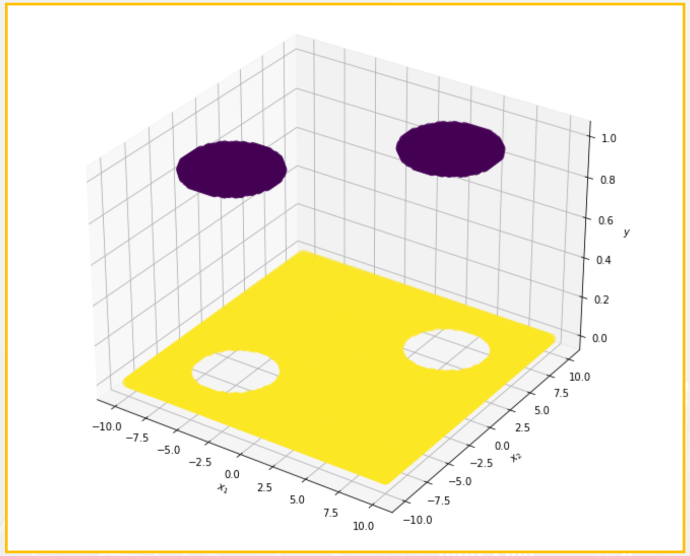


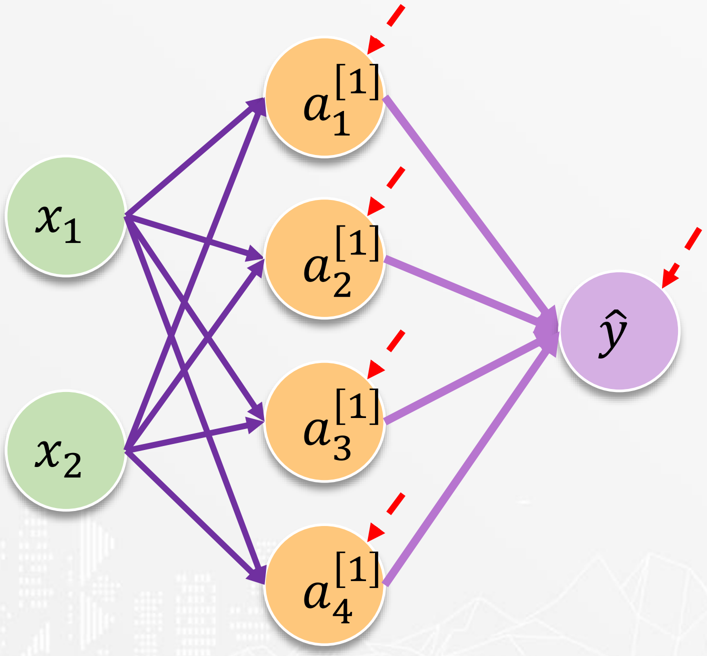


Neural Network Interpretation

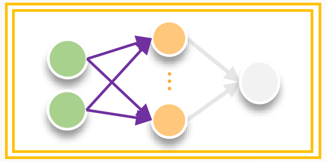
Classification

Example 4.3

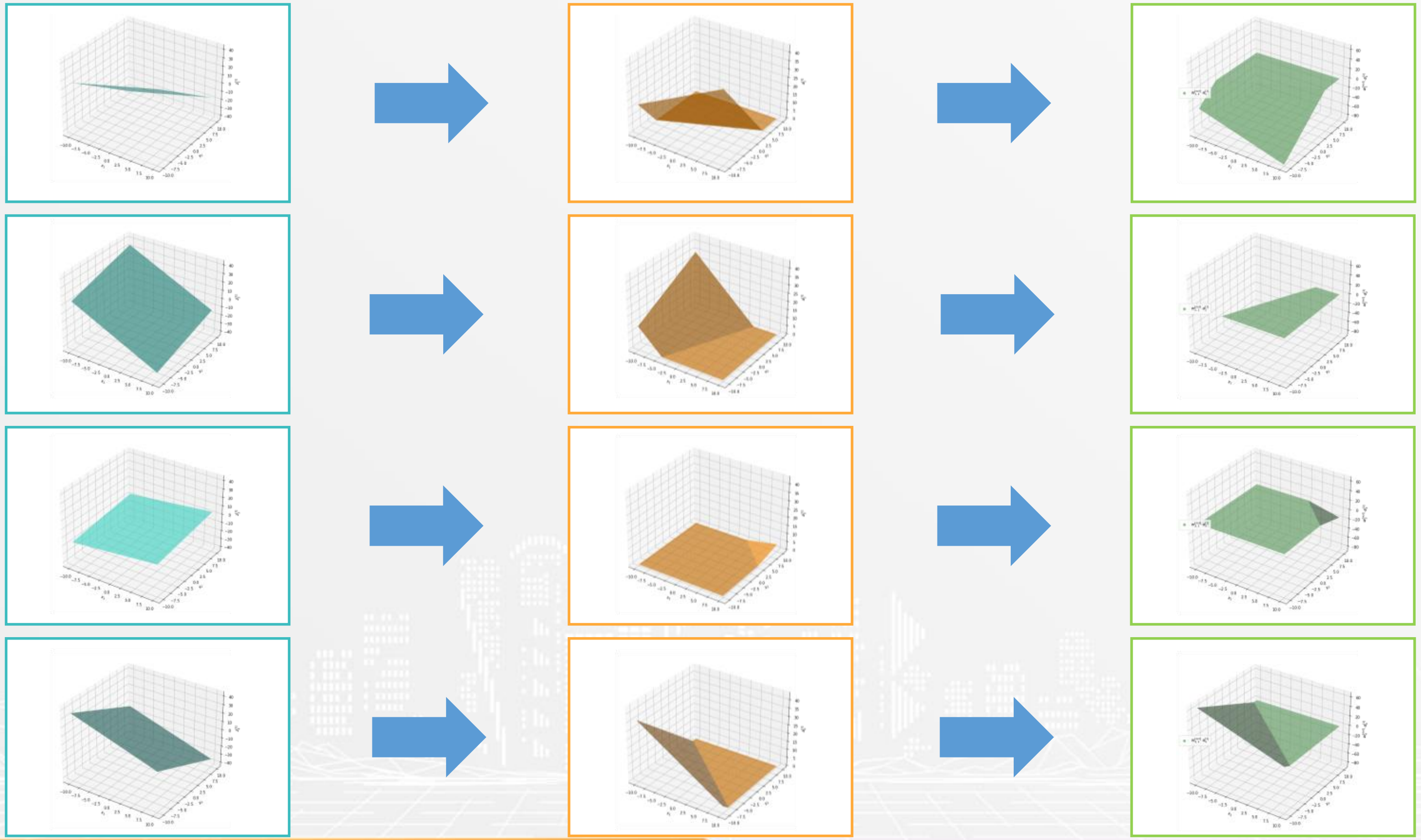




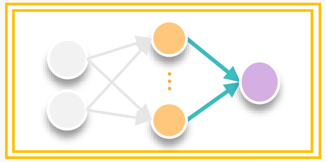
Neural Network Interpretation



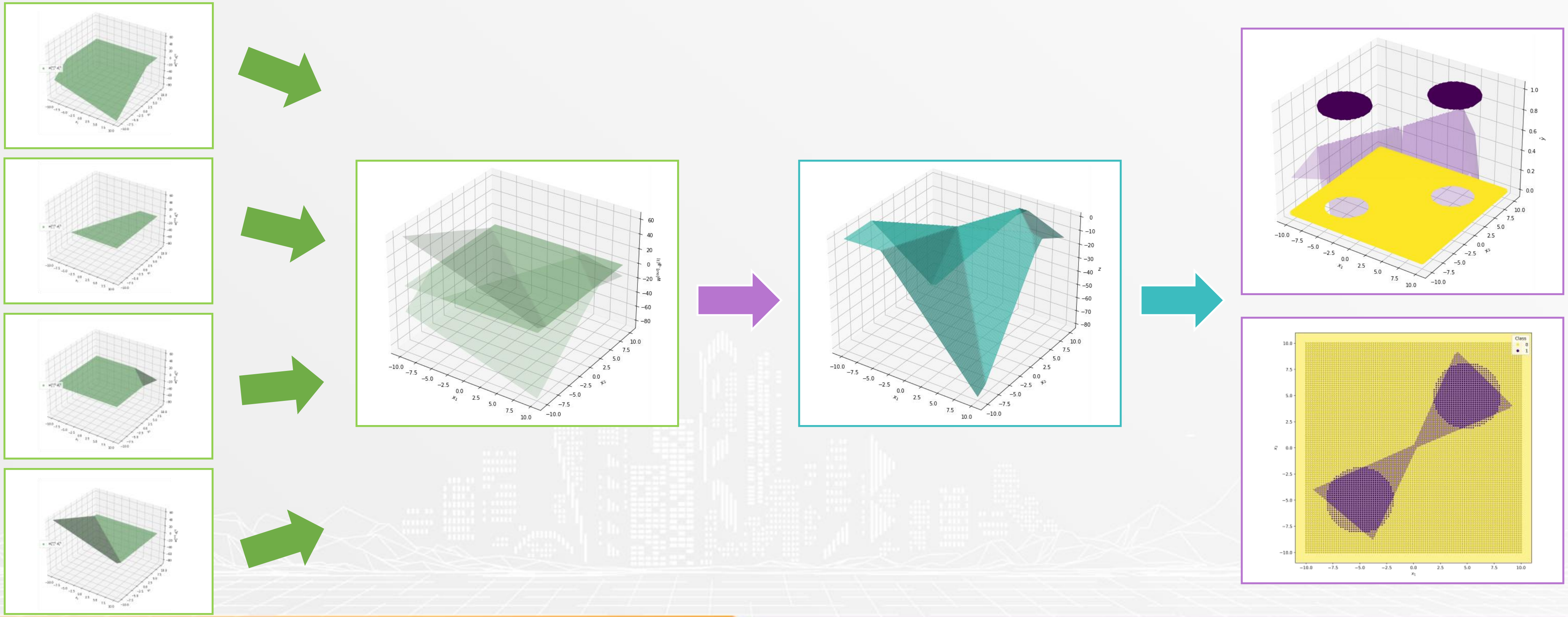
Classification | **Example 4.3**



Neural Network Interpretation



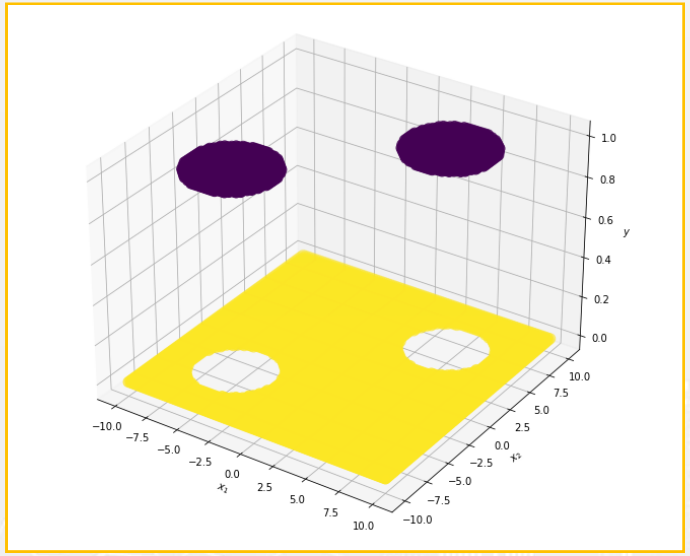
Classification | **Example 4.3**

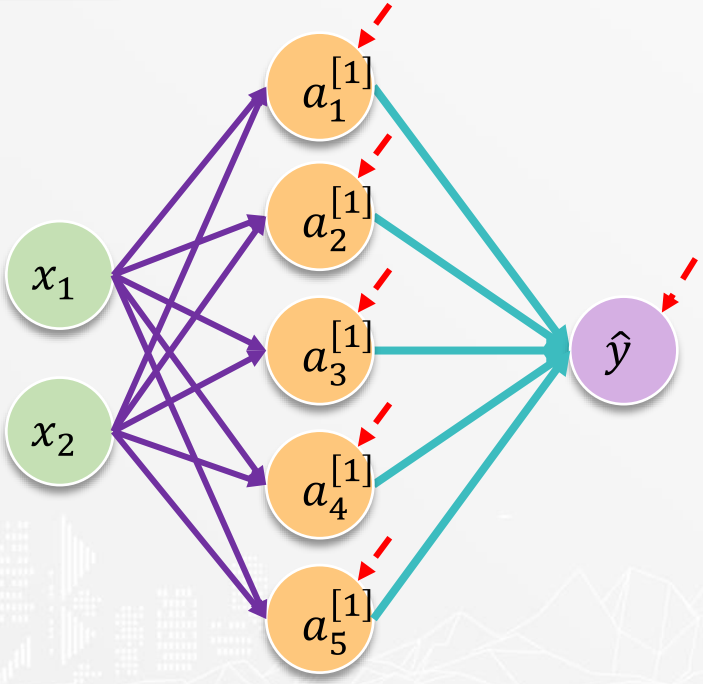


Neural Network Interpretation

Classification

Example 4.4

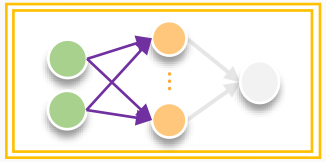


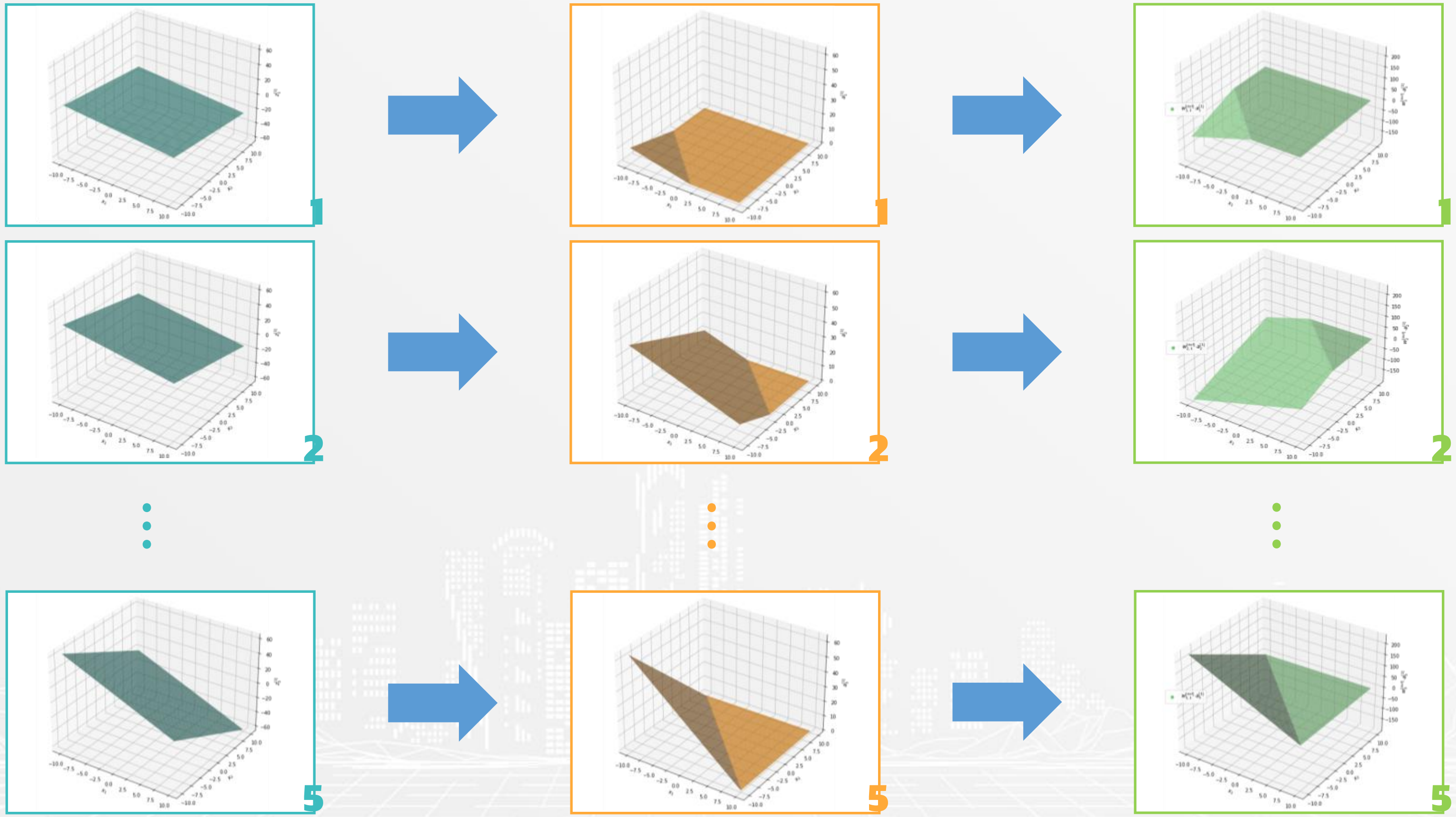


Neural Network Interpretation

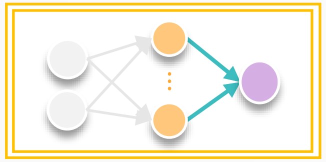
Classification

Example 4.4

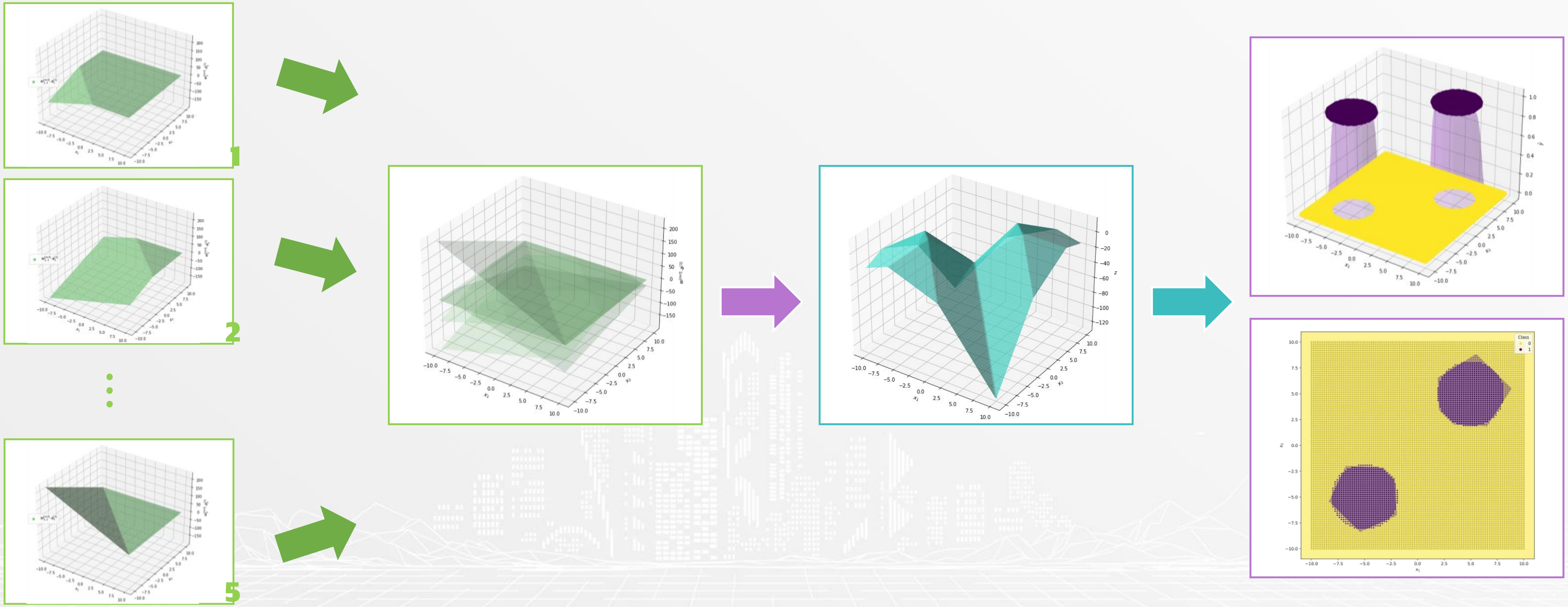




Neural Network Interpretation



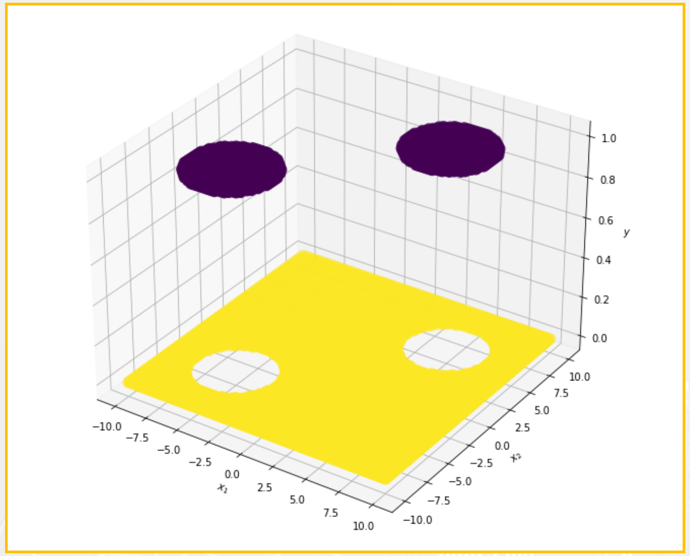
Classification **Example 4.4**

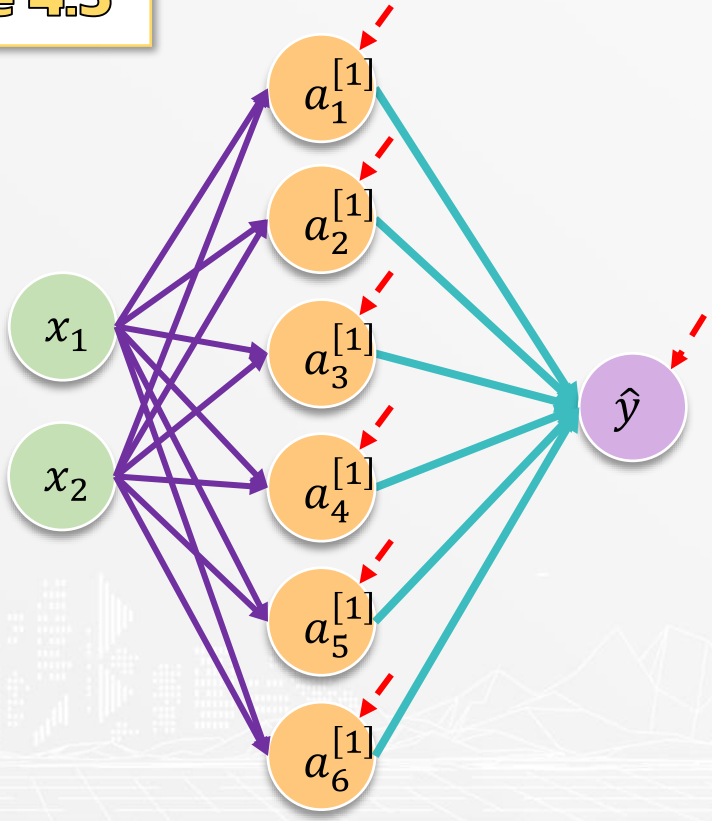


Neural Network Interpretation

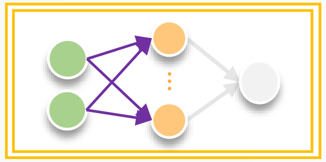
Classification

Example 4.5



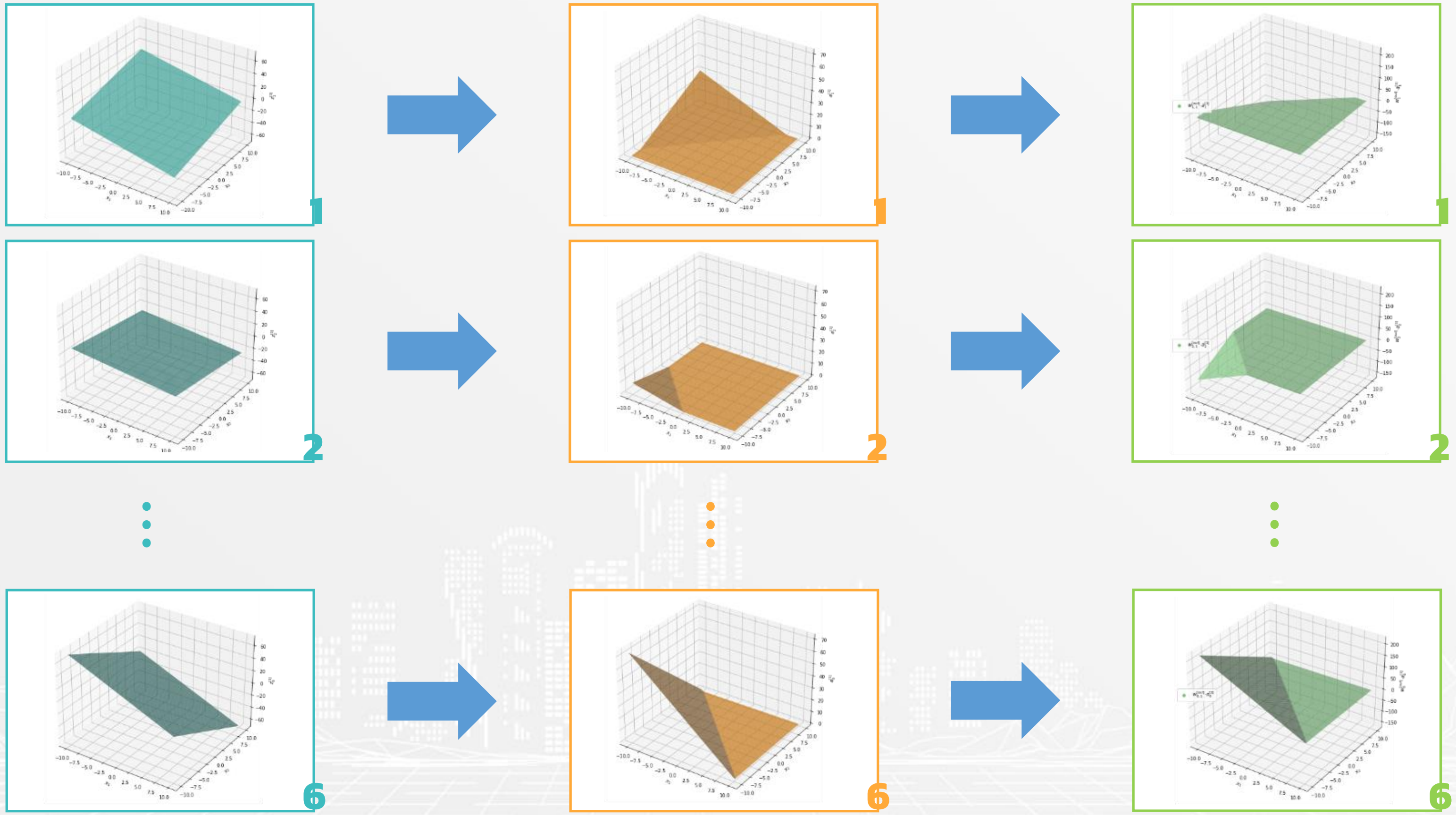


Neural Network Interpretation



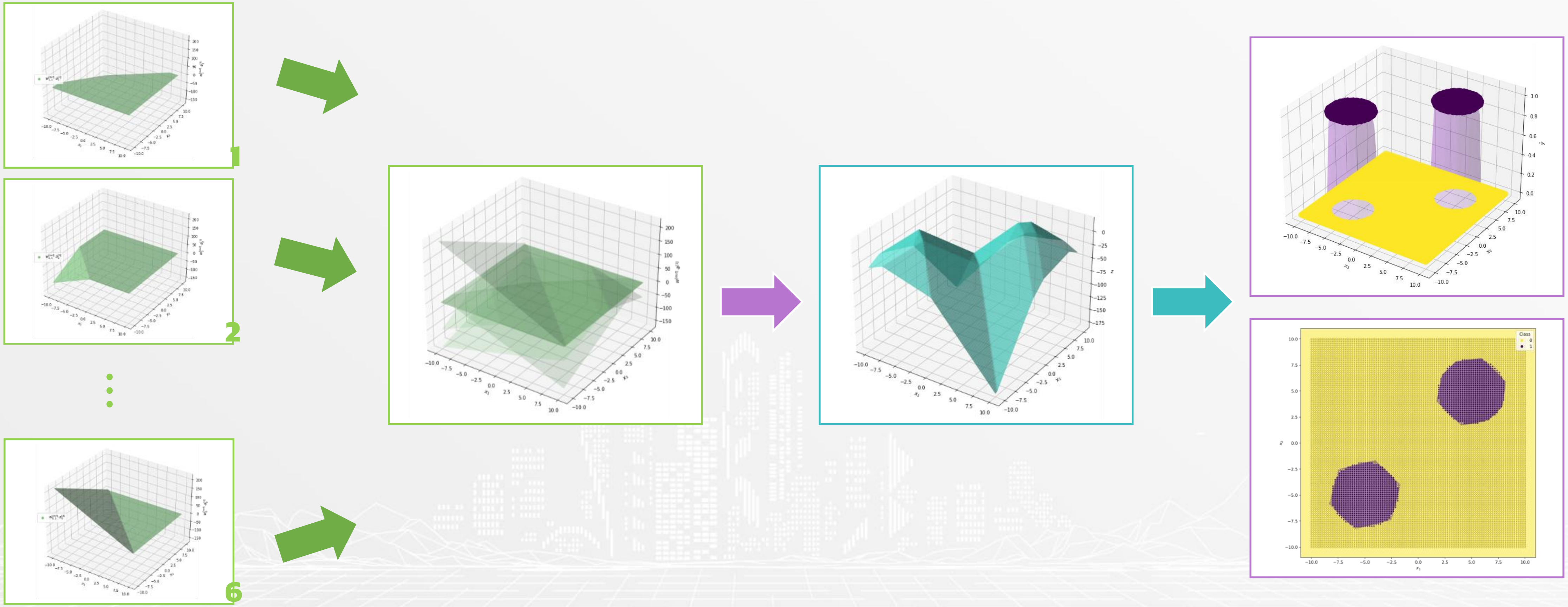
Classification

Example 4.5



Neural Network Interpretation

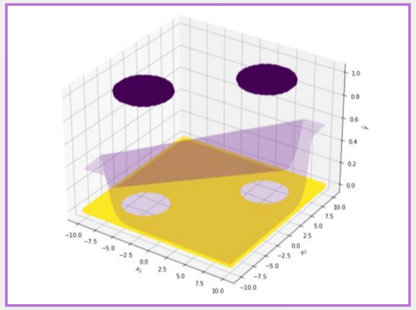
Classification **Example 4.5**



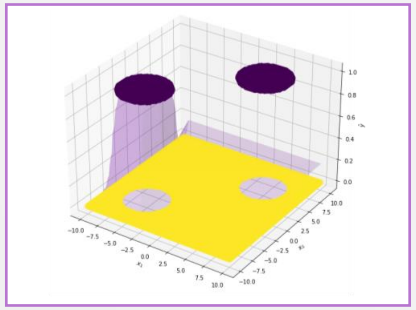
Neural Network Interpretation

Classification

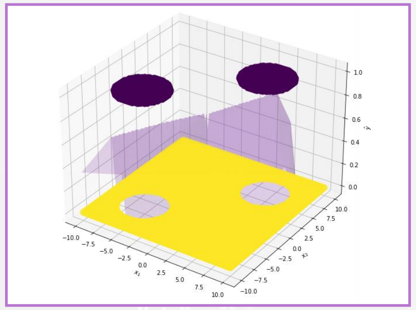
กราฟแสดงค่า predicted ของ Example 4



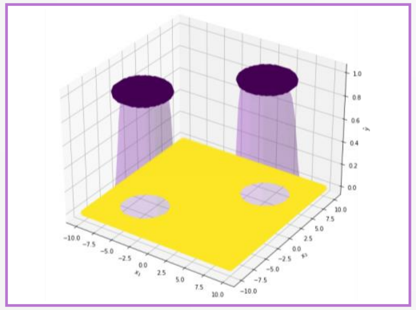
Example 5.1 (1,2,1)



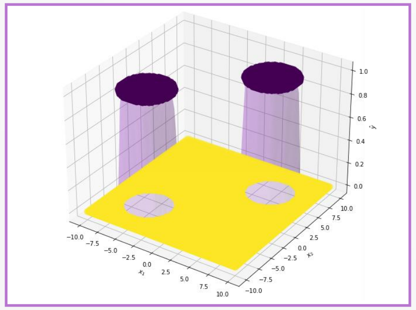
Example 5.2 (1,3,1)



Example 5.3 (1,4,1)



Example 5.4 (1,5,1)

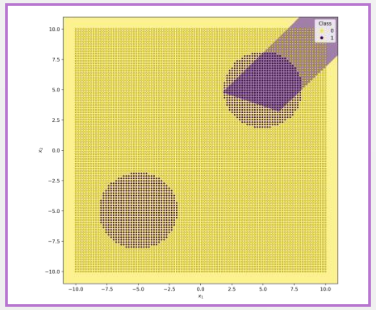


Example 5.5 (1,6,1)

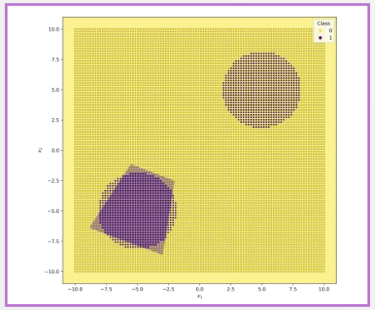
Neural Network Interpretation

Classification

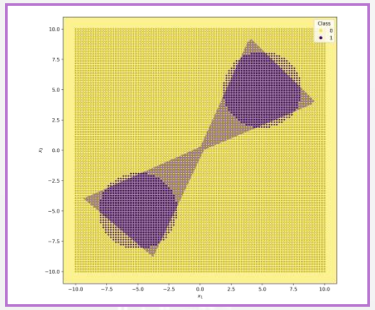
กราฟแสดงพื้นที่ predicted ของ Example 4



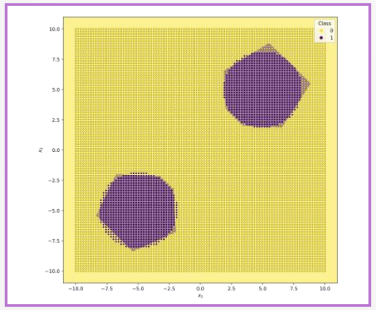
Example 4.1 (2,2,1)



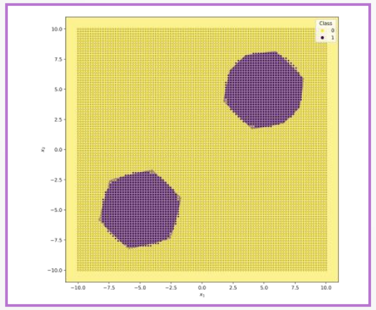
Example 4.2 (2,3,1)



Example 4.3 (2,4,1)



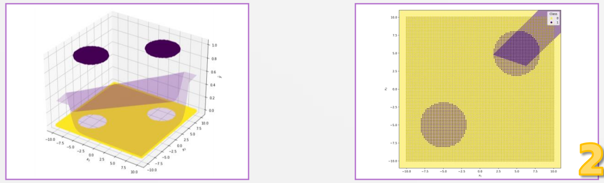
Example 4.4 (2,5,1)



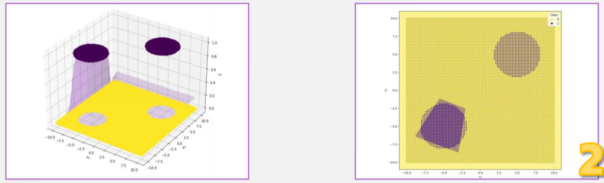
Example 4.5 (2,6,1)

Neural Network Interpretation

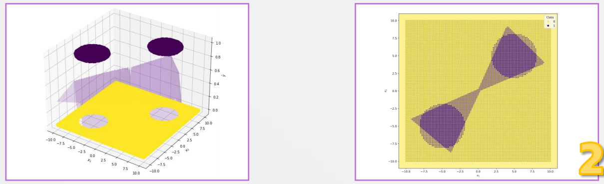
Classification



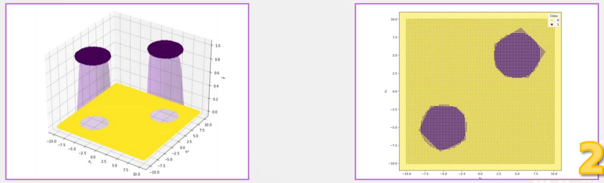
2,2,1



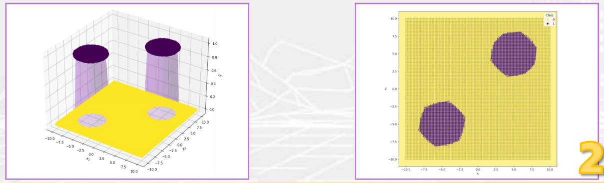
2,3,1



2,4,1



2,5,1

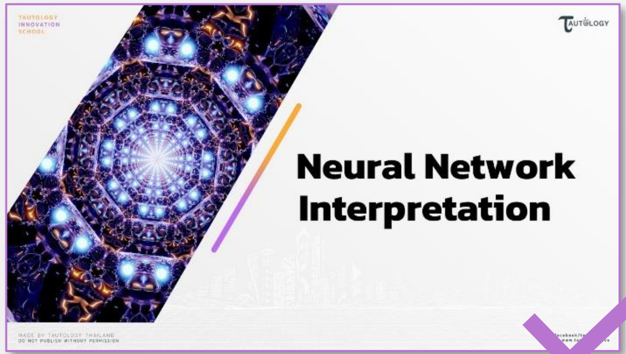


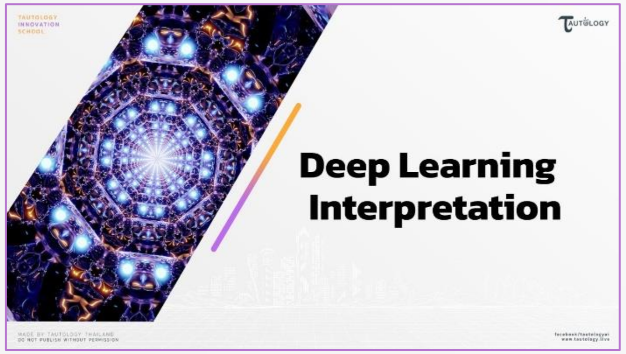
2,6,1

ข้อสรุปของ Example 4

- ทั้งนี้การที่จะเกิด maximum border นั้นขึ้นอยู่กับ dataset และ initial parameter ด้วย

Deep Learning Interpretation

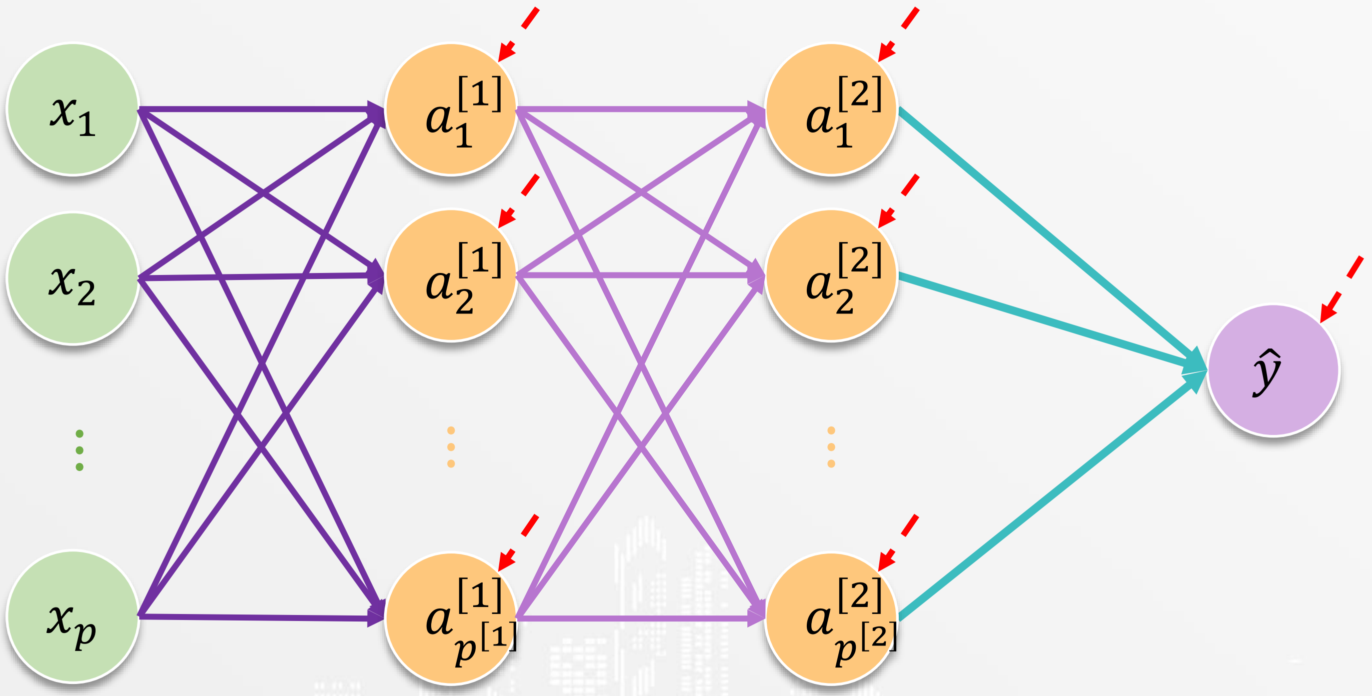




TAUTOLOGY

Deep Learning Interpretation

Deep Learning Interpretation

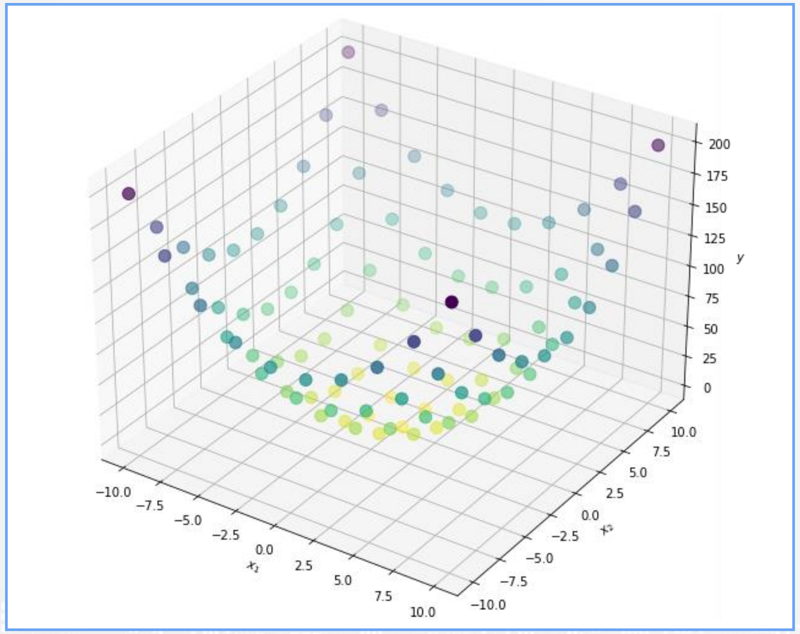


Deep Learning Interpretation

จำนวน hidden layer
ส่งผลอะไรต่อ model ?

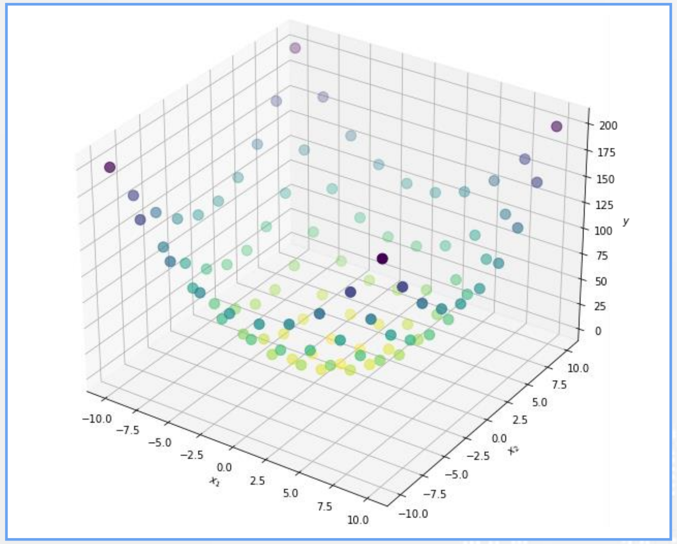
Deep Learning Interpretation

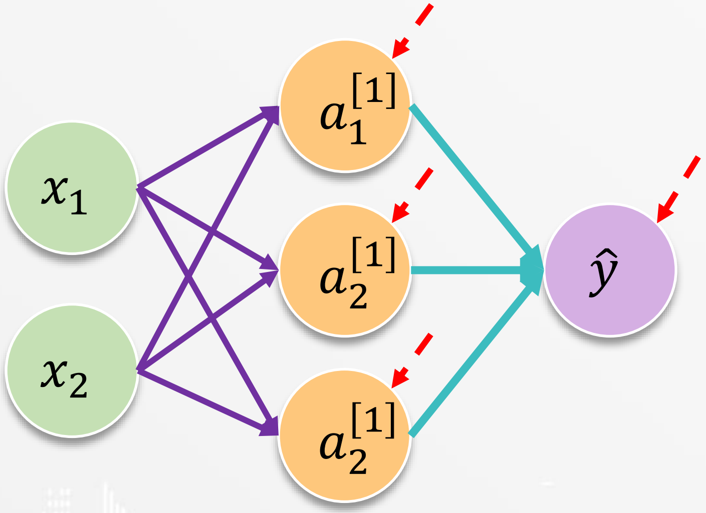
Example 1



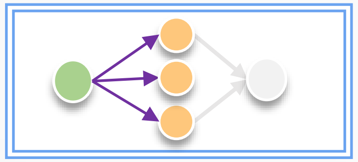
Deep Learning Interpretation

Example 1.1

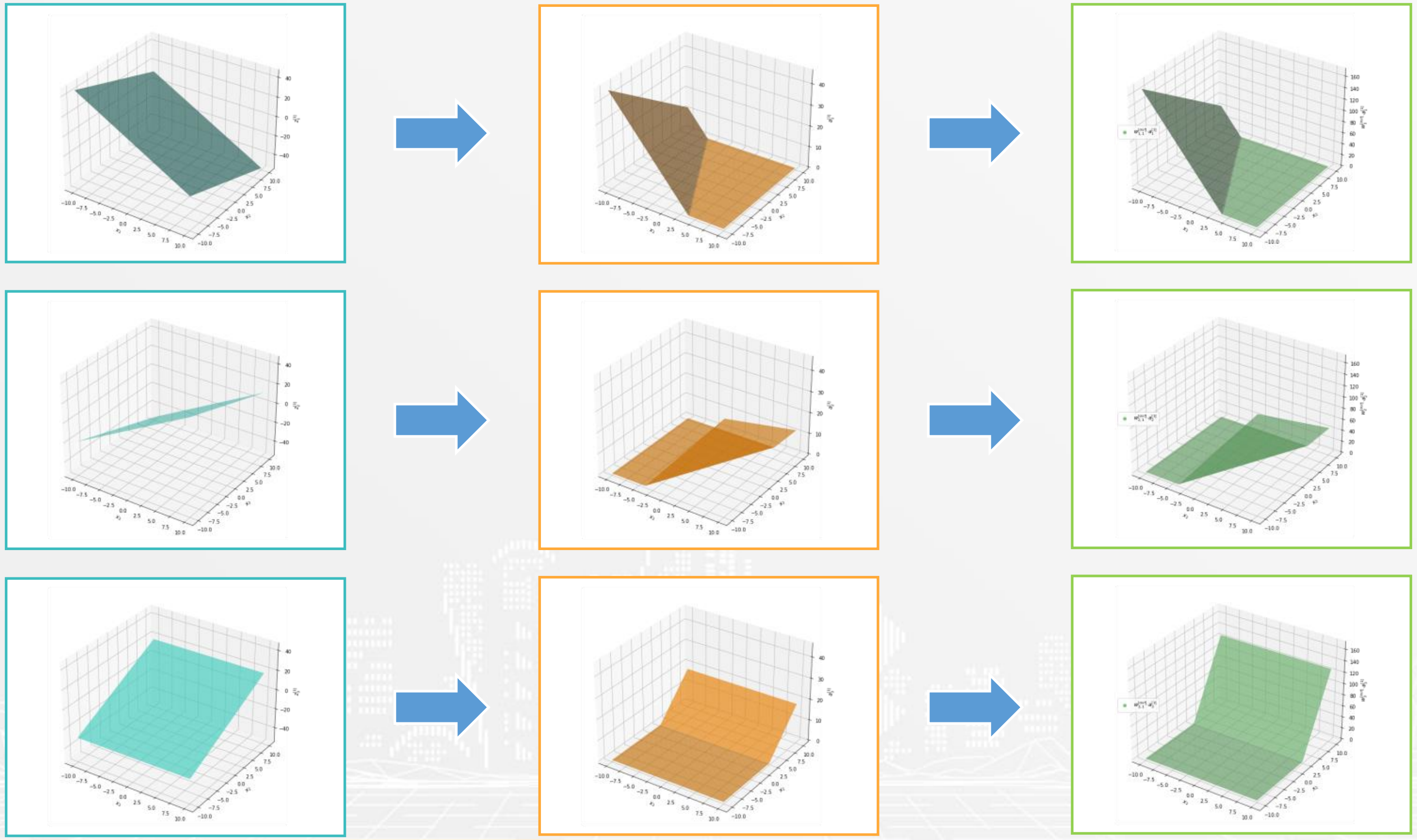




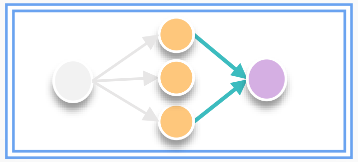
Deep Learning Interpretation



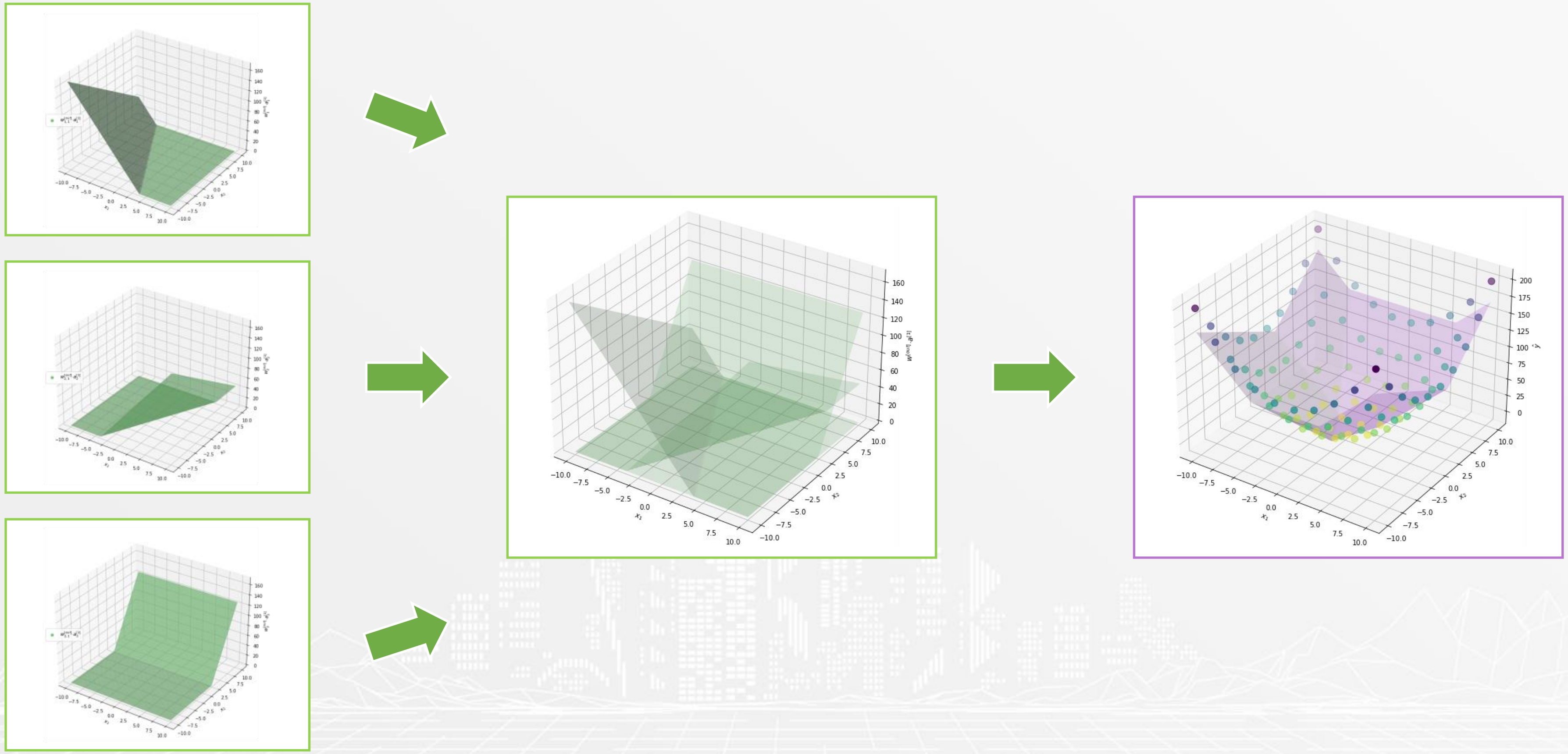
Example 1.1



Deep Learning Interpretation

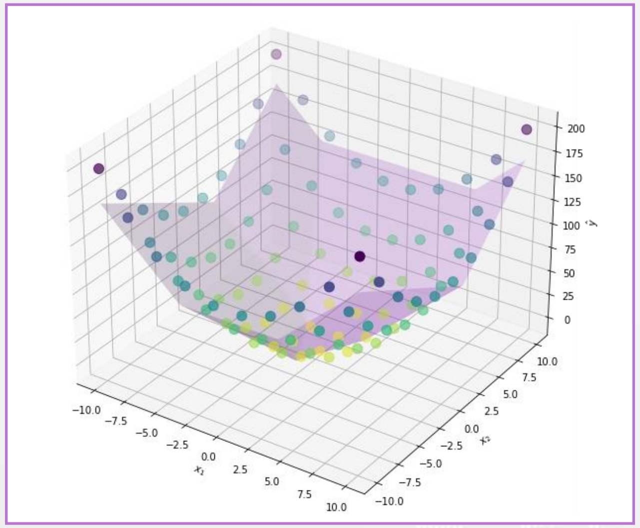


Example 1.1

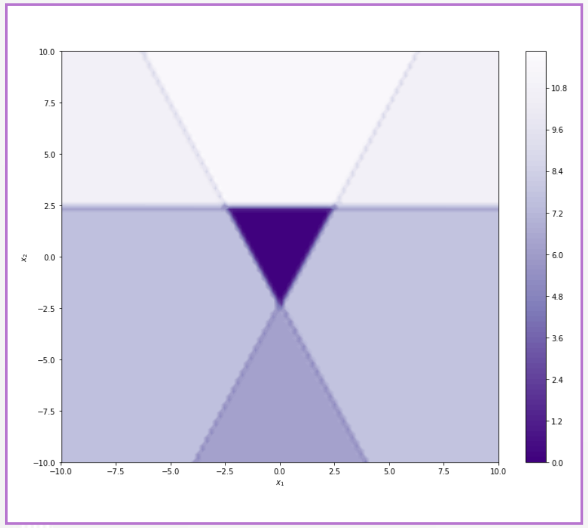


Deep Learning Interpretation

Example 1.1



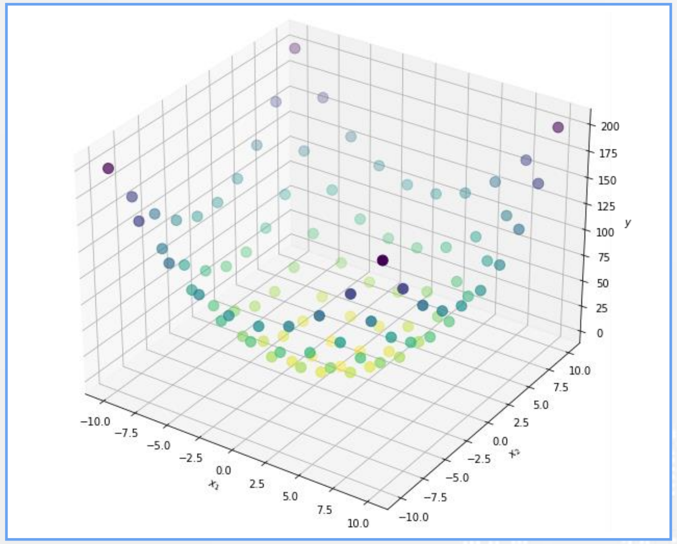
กราฟแสดงค่า predicted

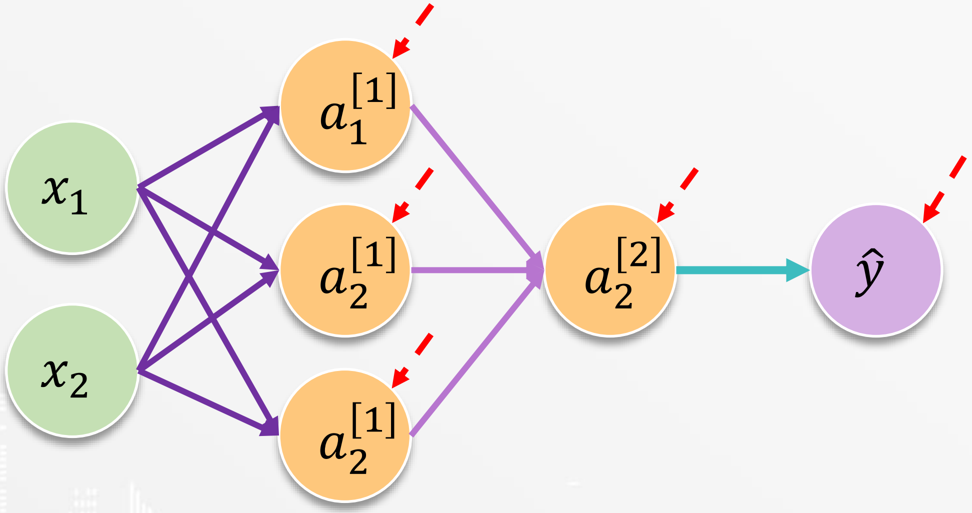


Contour plot ของ predicted

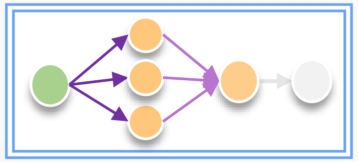
Deep Learning Interpretation

Example 1.2

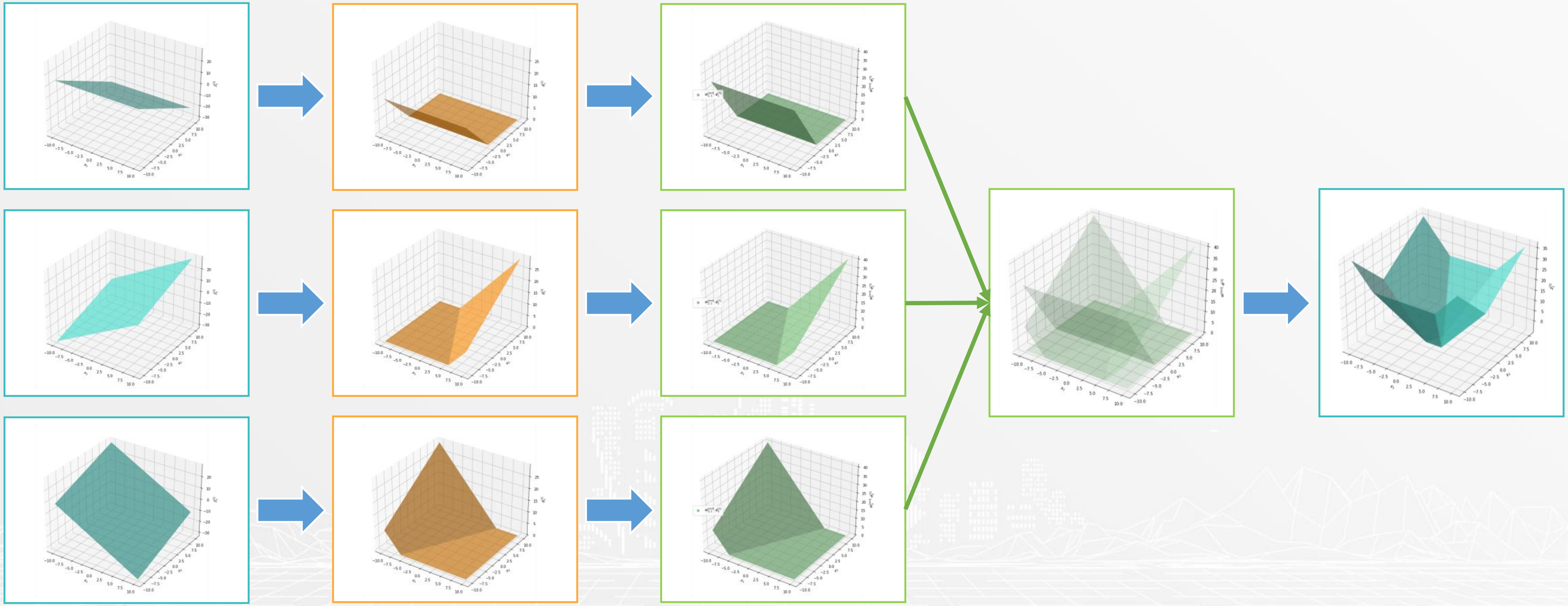




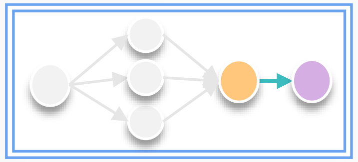
Deep Learning Interpretation



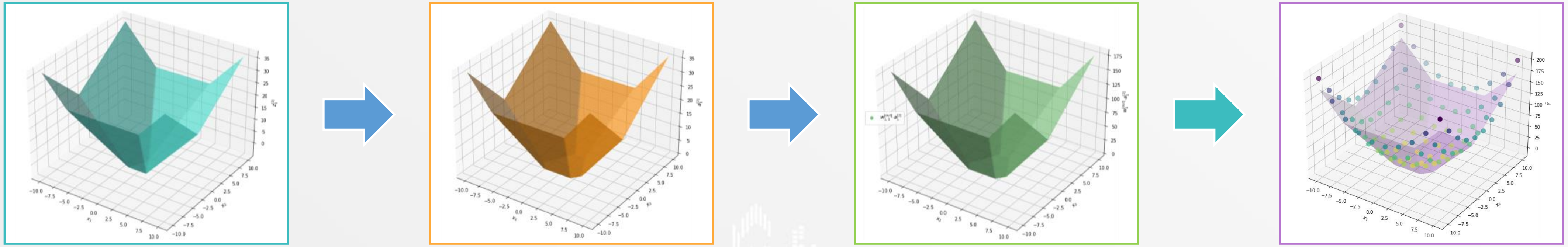
Example 1.2



Deep Learning Interpretation

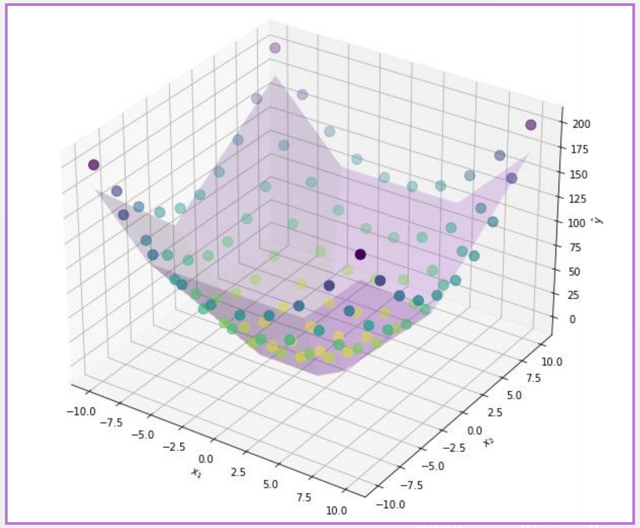


Example 1.2

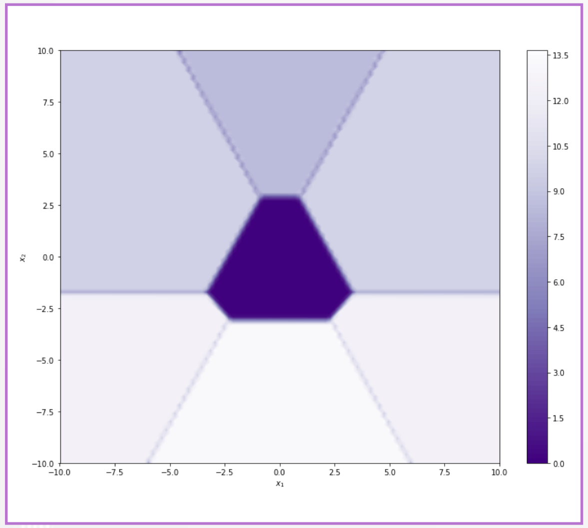


Deep Learning Interpretation

Example 1.2

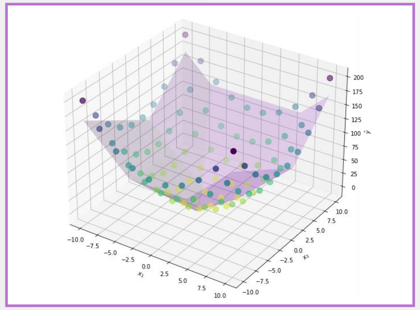


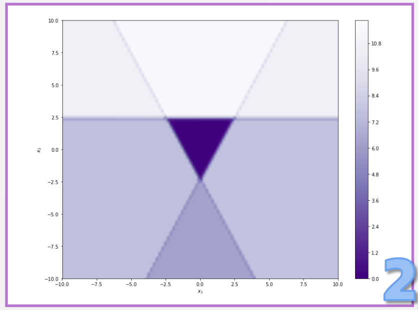
กราฟแสดงค่า predicted



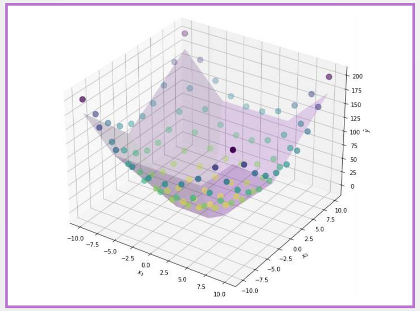
Contour plot ของ predicted

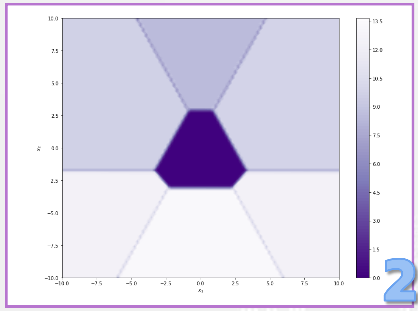
Deep Learning Interpretation





2,3,1





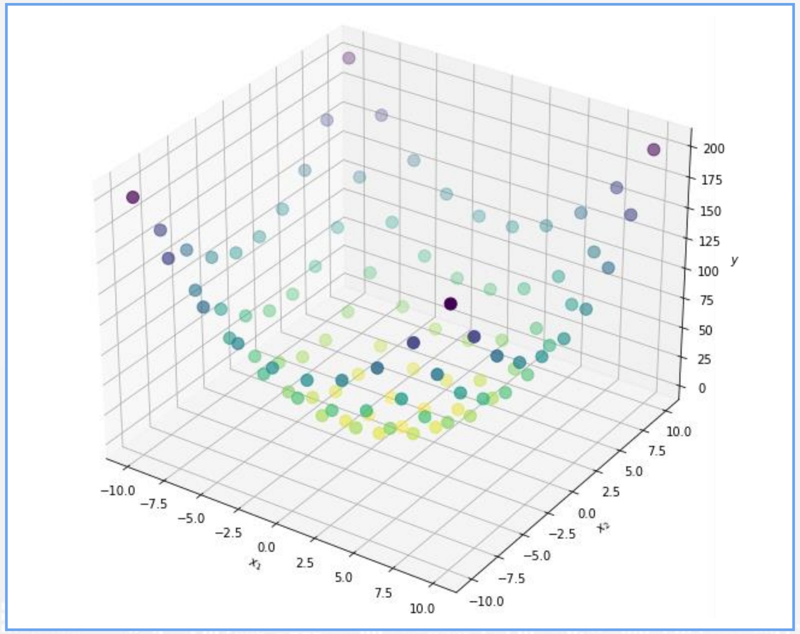
2,3,1,1

REMARK

การเพิ่ม hidden layer ทำให้เกิด plane ย่อย เพิ่มขึ้น โดยเกิดจากการทำซ้ำของการสร้างรอยหักใน hidden layer ขั้นต่อมา

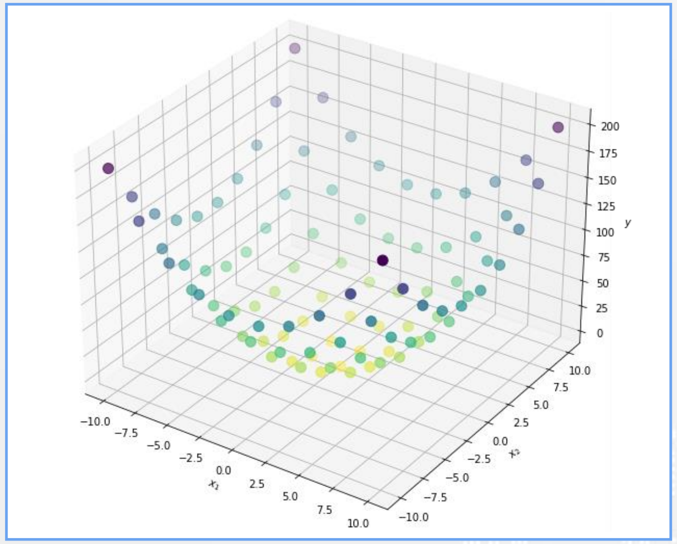
Deep Learning Interpretation

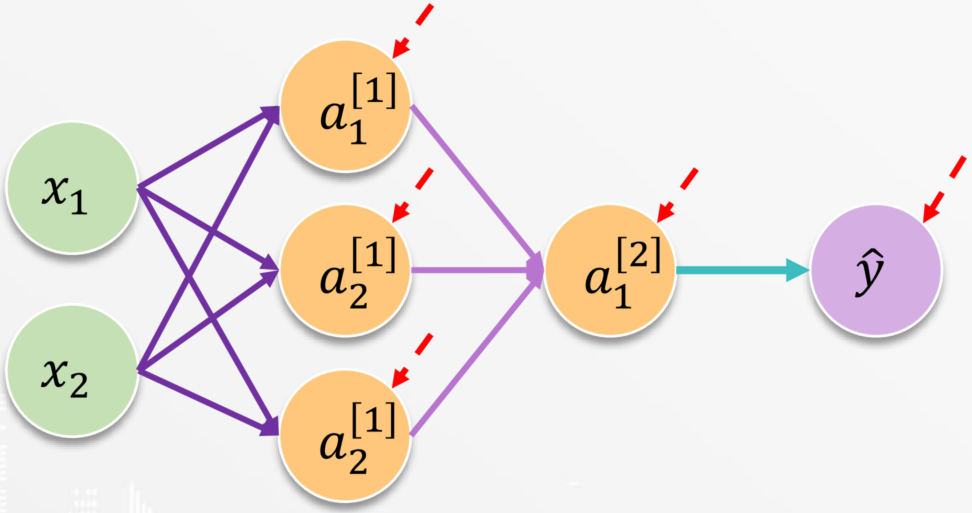
Example 2



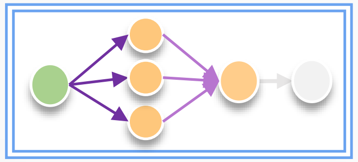
Deep Learning Interpretation

Example 2

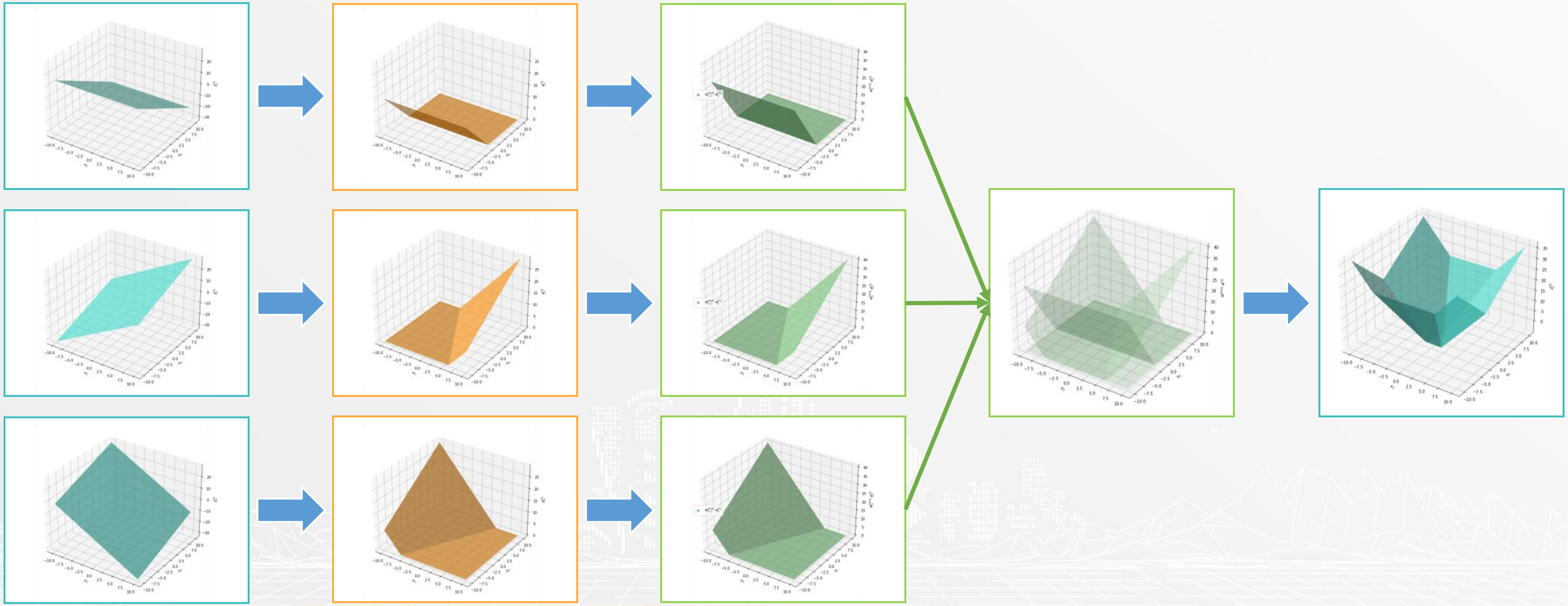




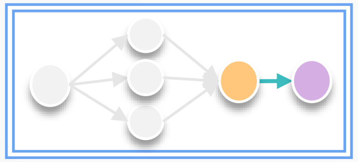
Deep Learning Interpretation



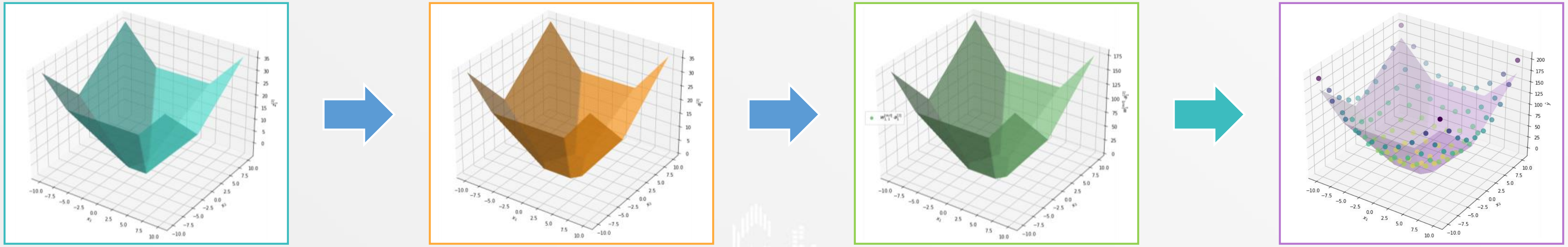
Example 2.1



Deep Learning Interpretation

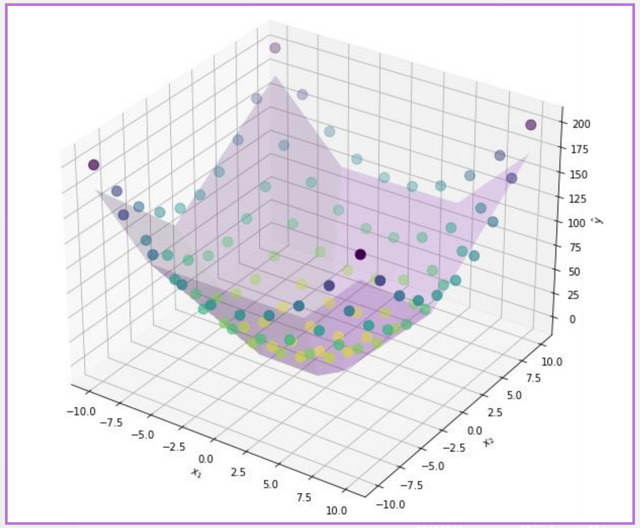


Example 2.1

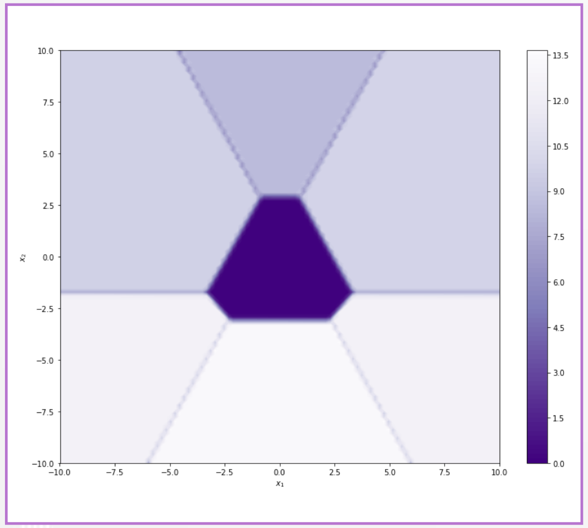


Deep Learning Interpretation

Example 2.1



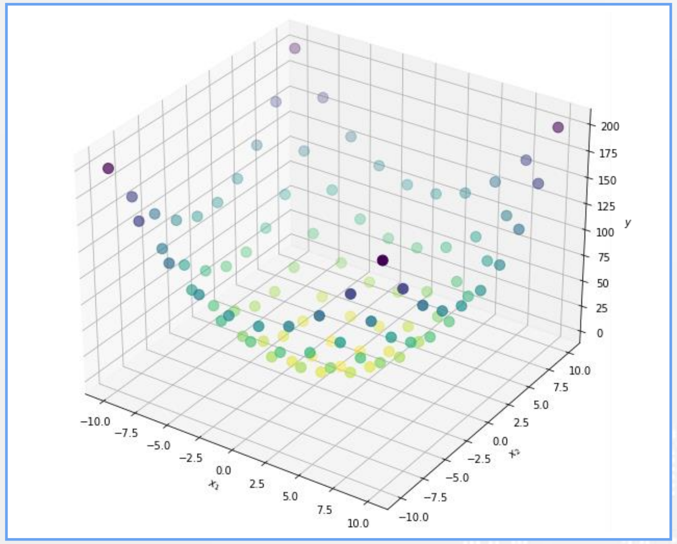
กราฟแสดงค่า predicted

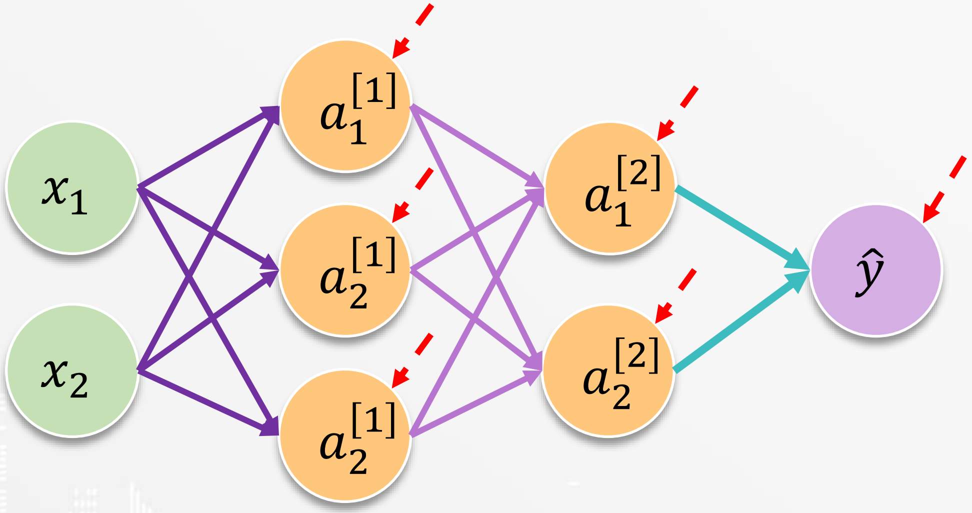


Contour plot ของ predicted

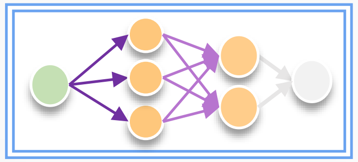
Deep Learning Interpretation

Example 2.2

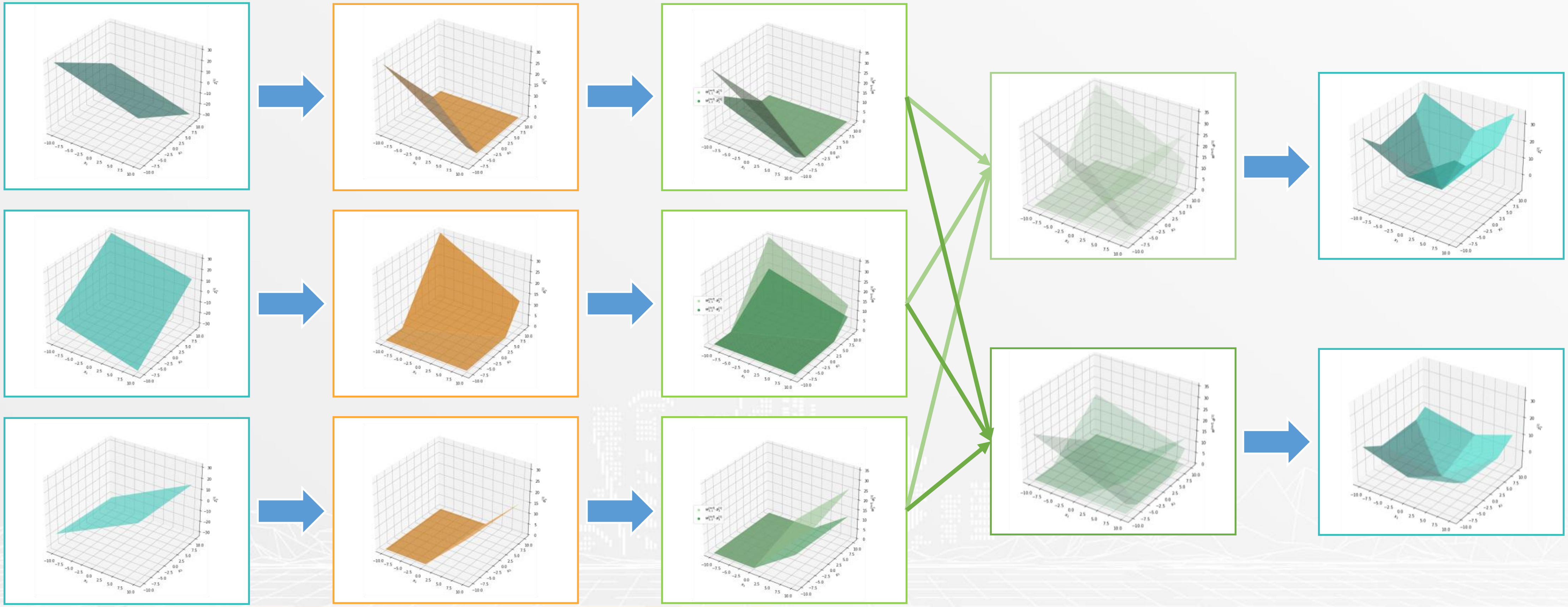




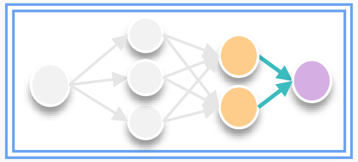
Deep Learning Interpretation



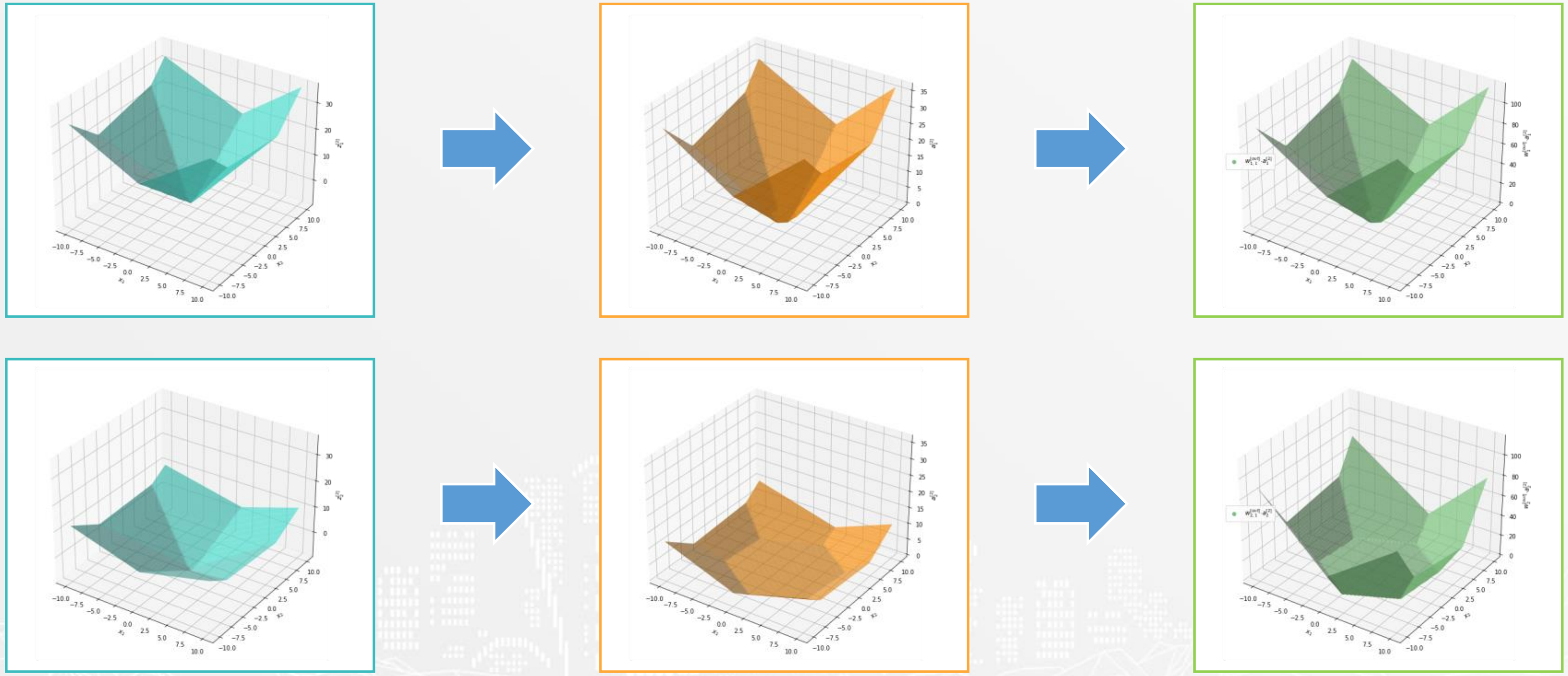
Example 2.2



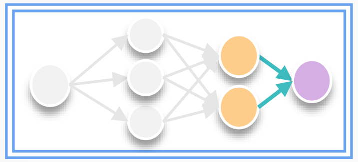
Deep Learning Interpretation



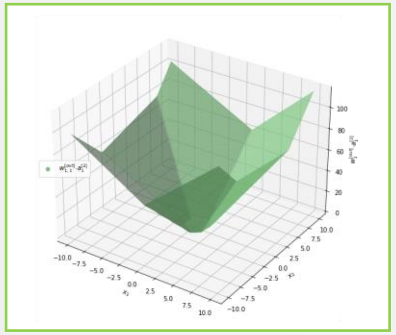
Example 2.2

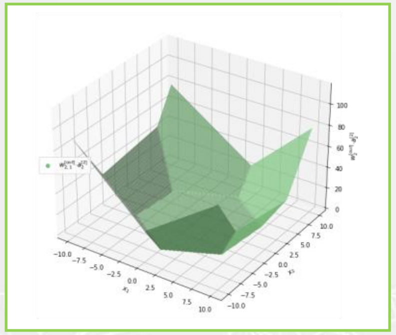


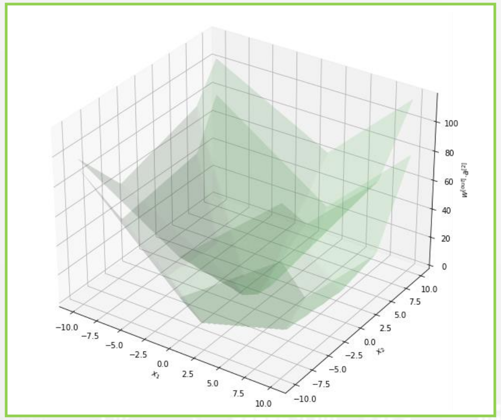
Deep Learning Interpretation

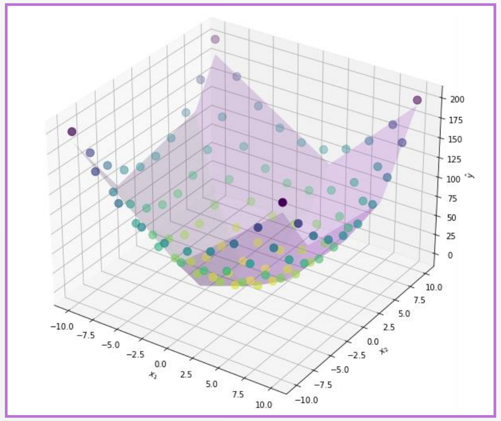


Example 2.2



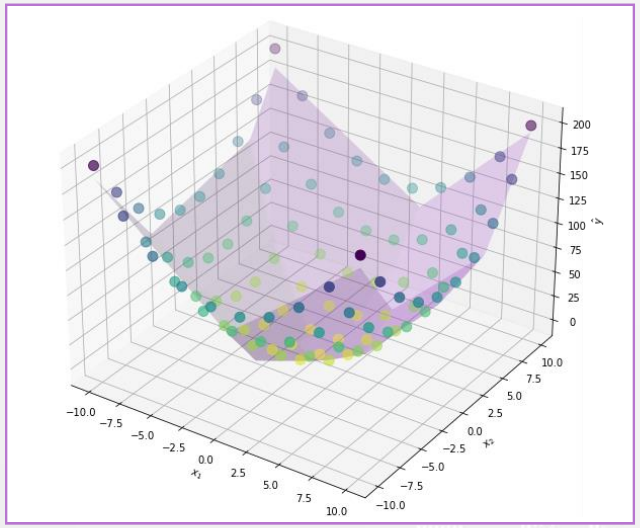




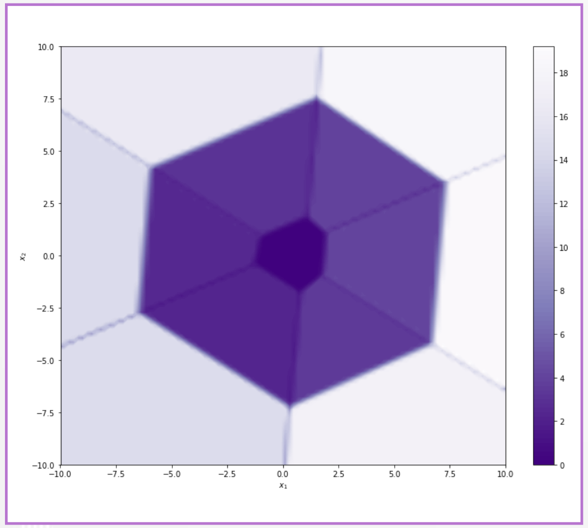


Deep Learning Interpretation

Example 2.2



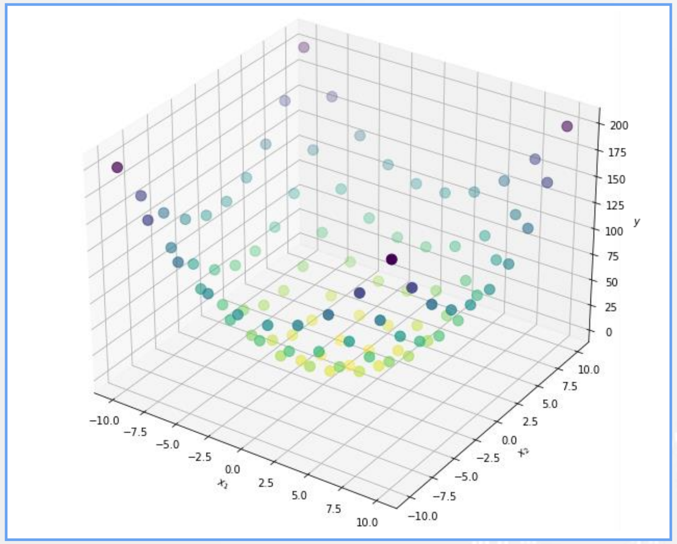
กราฟแสดงค่า predicted

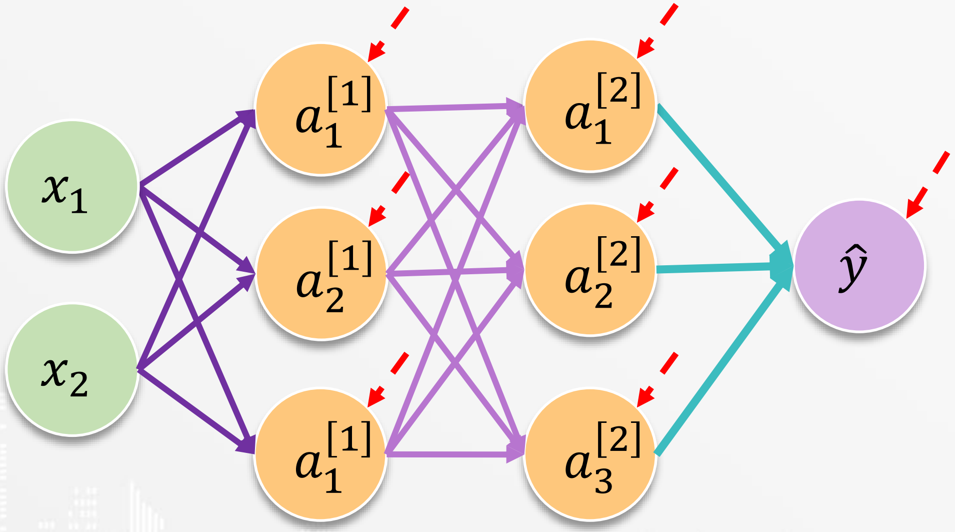


Contour plot ของ predicted

Deep Learning Interpretation

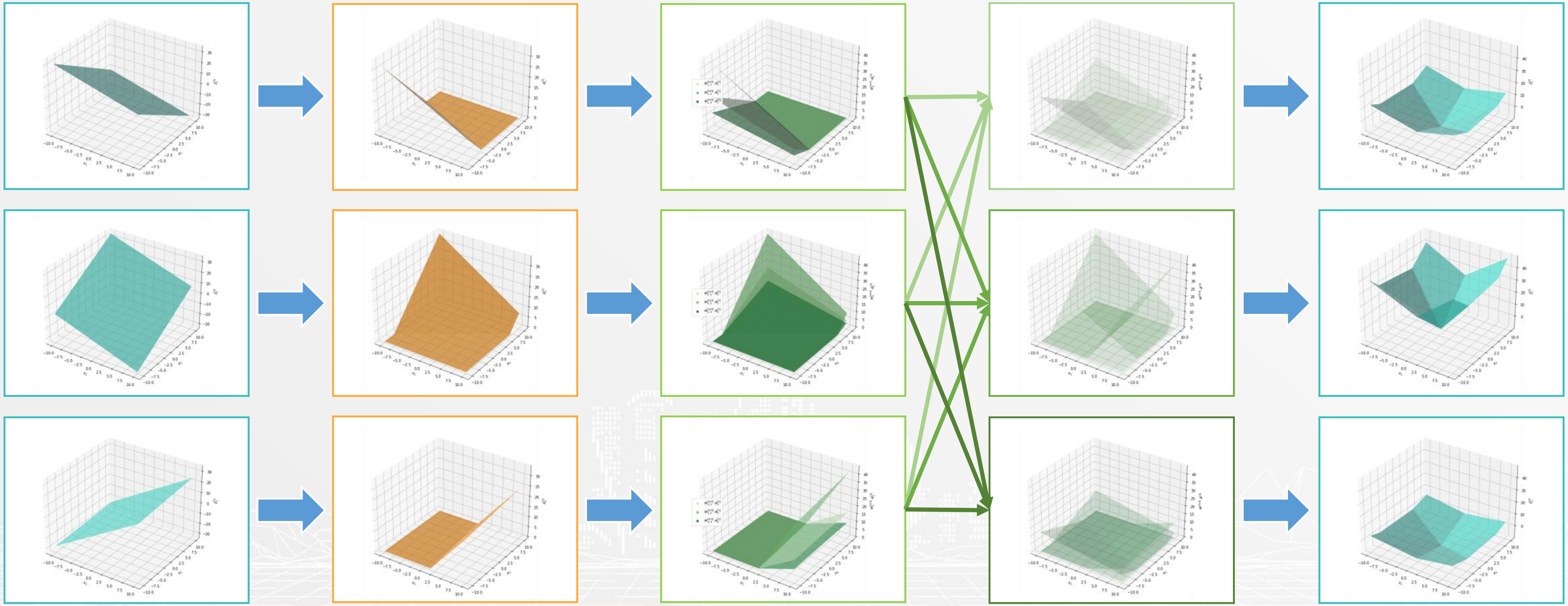
Example 2.3



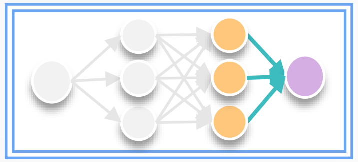


Deep Learning Interpretation

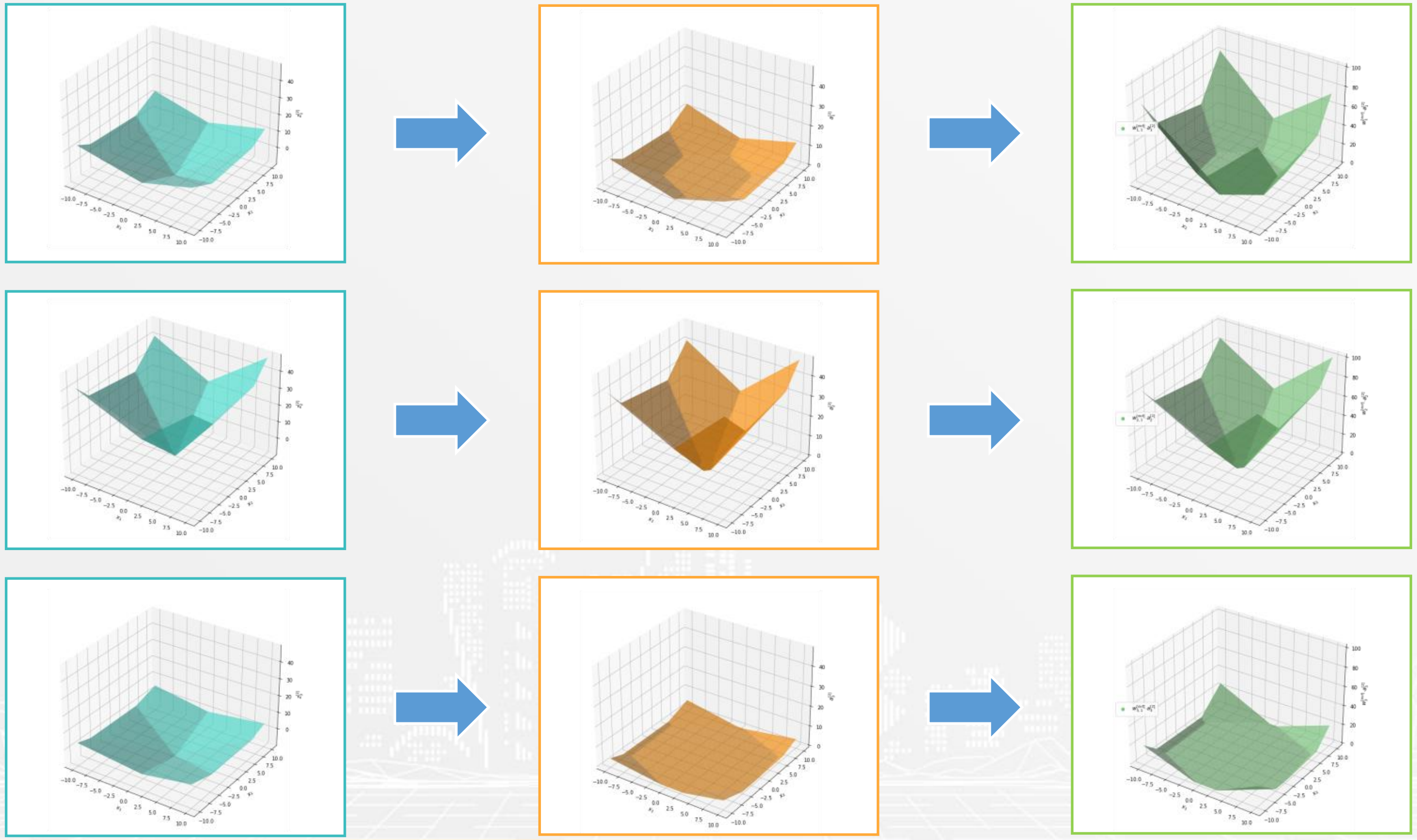
Example 2.3



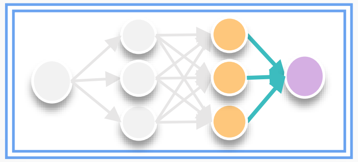
Deep Learning Interpretation



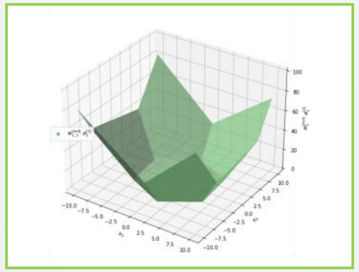
Example 2.3

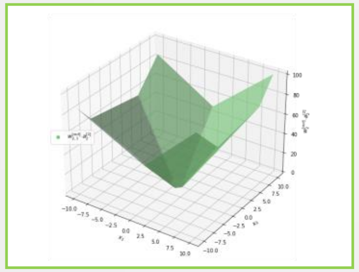


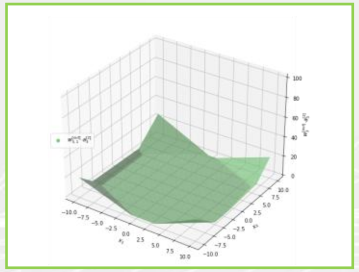
Deep Learning Interpretation

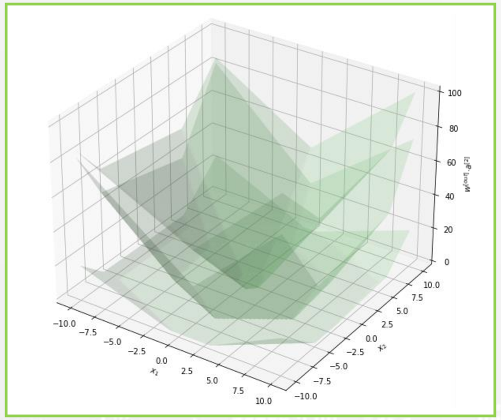


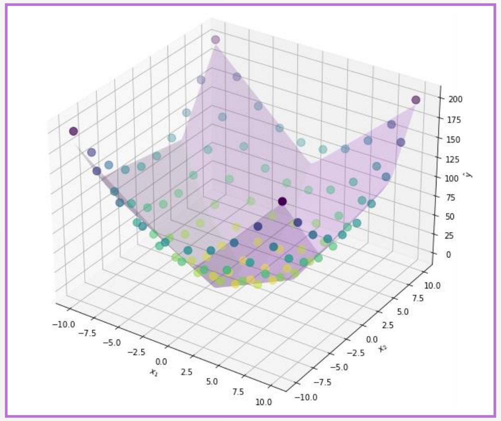
Example 2.3





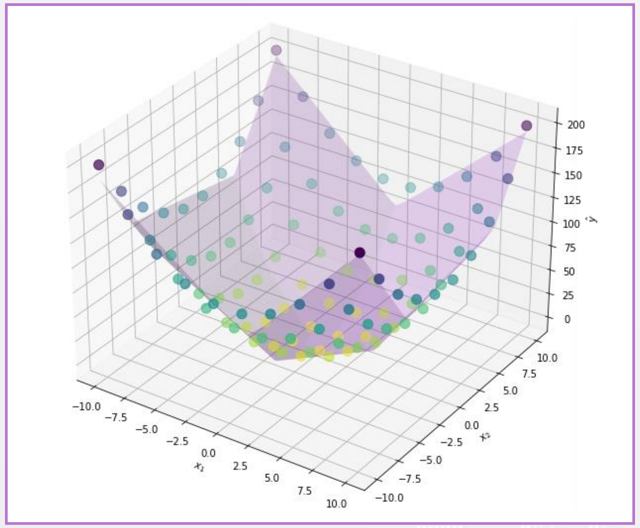




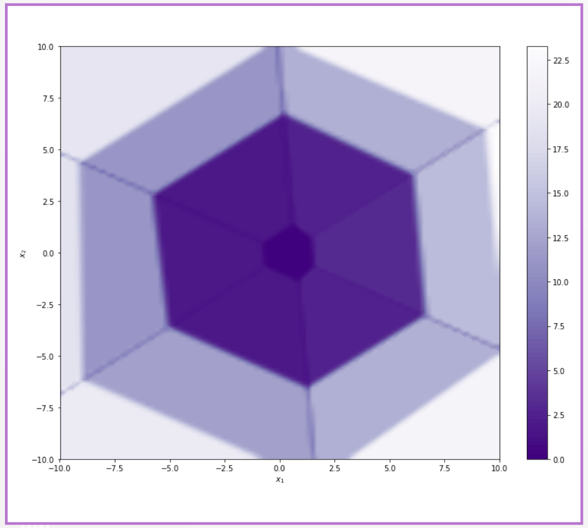


Deep Learning Interpretation

Example 2.3

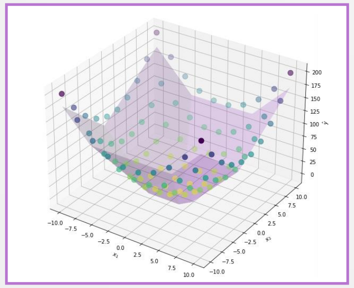


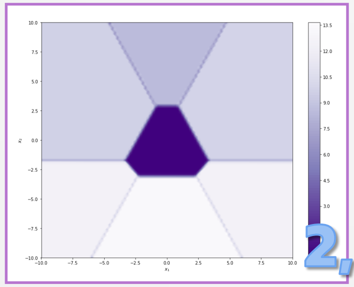
กราฟแสดงค่า predicted



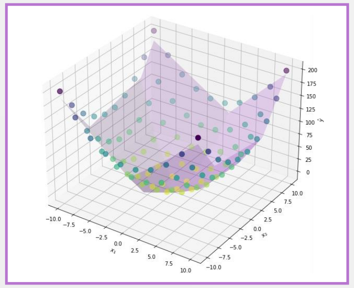
Contour plot ของ predicted

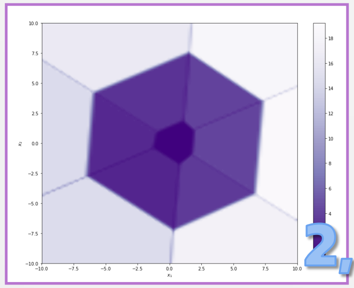
Deep Learning Interpretation



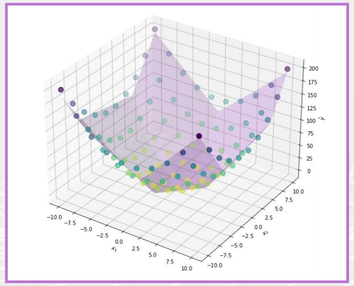


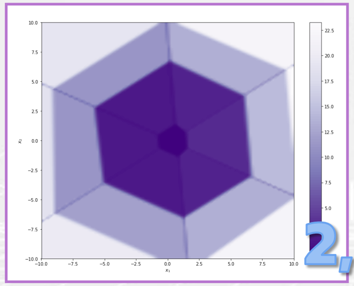
2,3,1,1





2,3,2,1



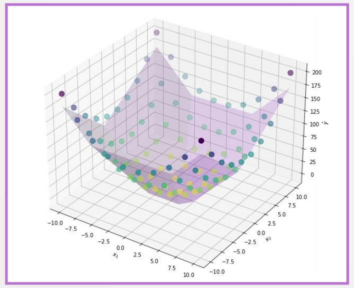


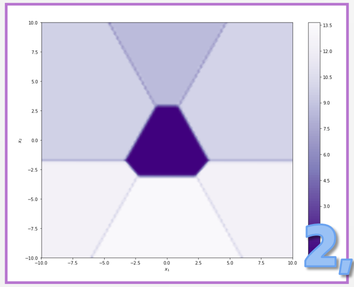
2,3,3,1

REMARK

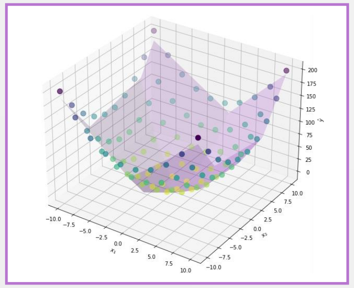
จำนวน node ใน hidden layer ในชั้นที่ 2 คือการสร้างความหลากหลายของรูปแบบของการหักซ้ำ ด้วยการคูณด้วย weight ที่มีค่าแตกต่างกัน

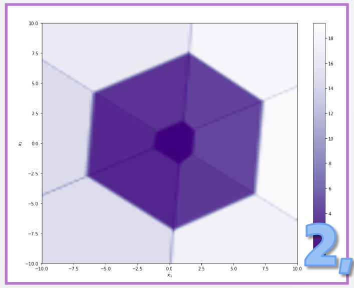
Deep Learning Interpretation



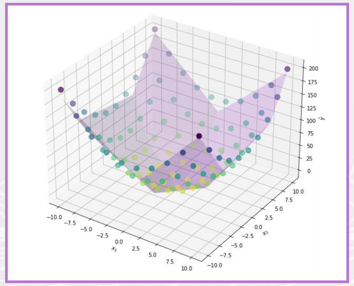


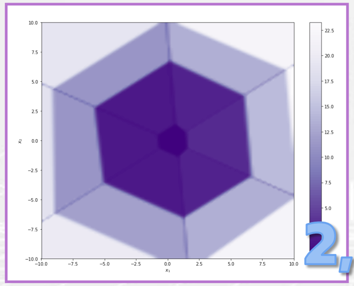
2,3,1,1





2,3,2,1



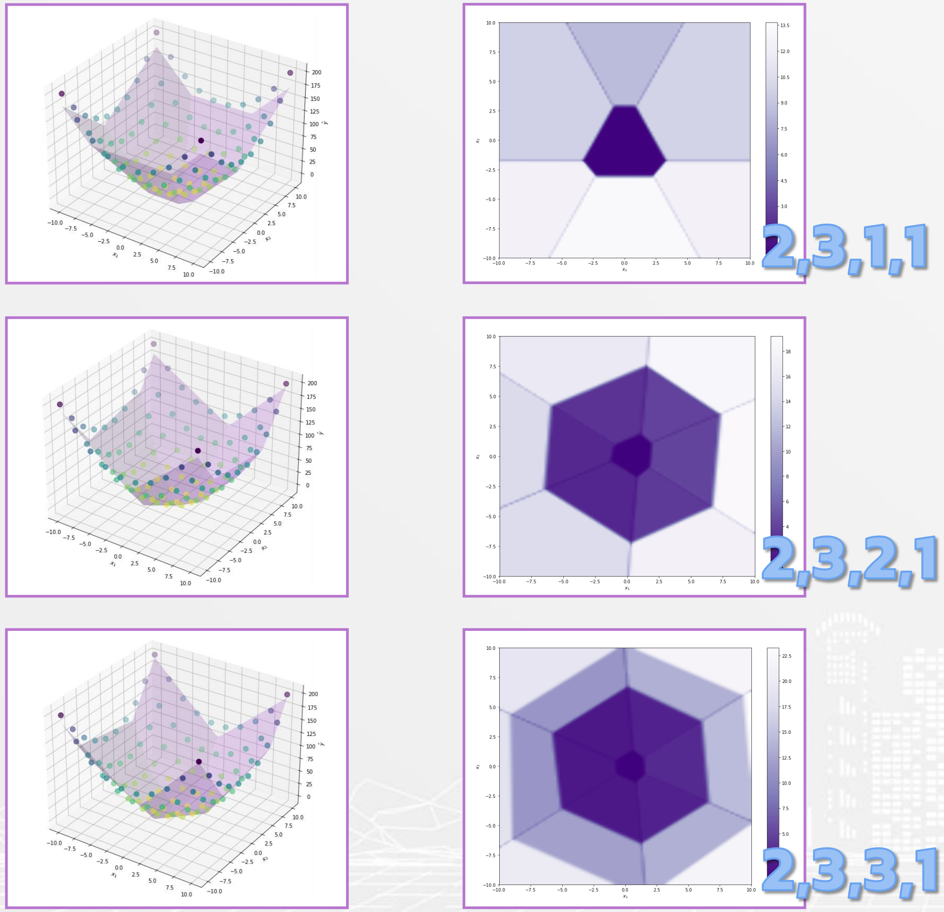


2,3,3,1

REMARK

และจะเห็นว่าเมื่อนำ function ที่ได้จากการ คูณด้วย weight มาผ่าน activation function ทำให้เกิดรอยหักใหม่ที่แตกต่างกัน

Deep Learning Interpretation



REMARK

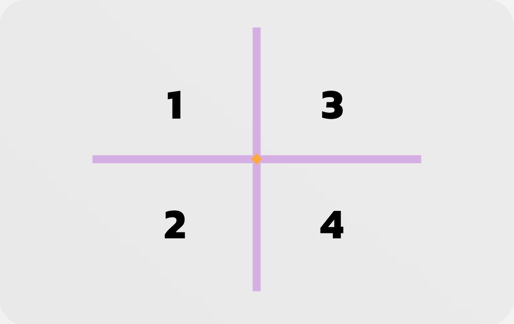
การหักใหม่ที่เกิดขึ้นนั้นเกิดจากการนำจุดหักเก่ามา คูณ weight ทำหักซ้ำ ถ้ามองในมุมมอง top view ของ 3 มิติ คือการเลื่อนขนาน

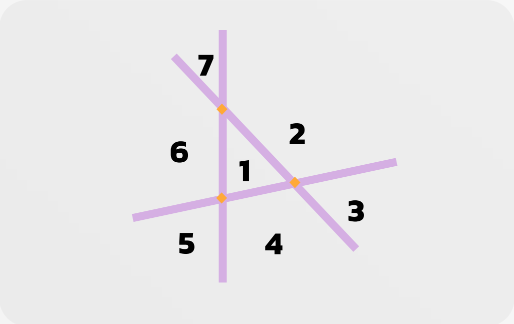
Deep Learning Interpretation

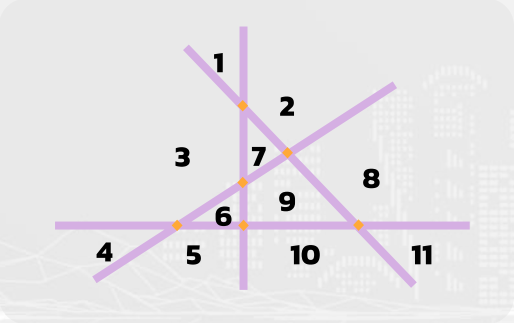
การเพิ่ม hidden layer คือการทำ plane ที่เกิดจาก output ที่หลากหลายที่เกิดจาก weight จากนั้นก่อนหน้ามาทำการประกอบ และหักซ้ำผ่าน activation function

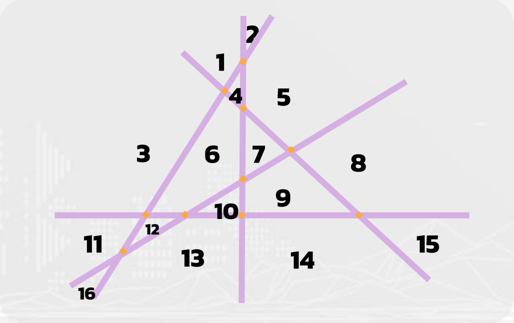
Deep Learning Interpretation

จากความรู้เดิม ลักษณะของ maximum plane ใน **Neural Network Interpretation**





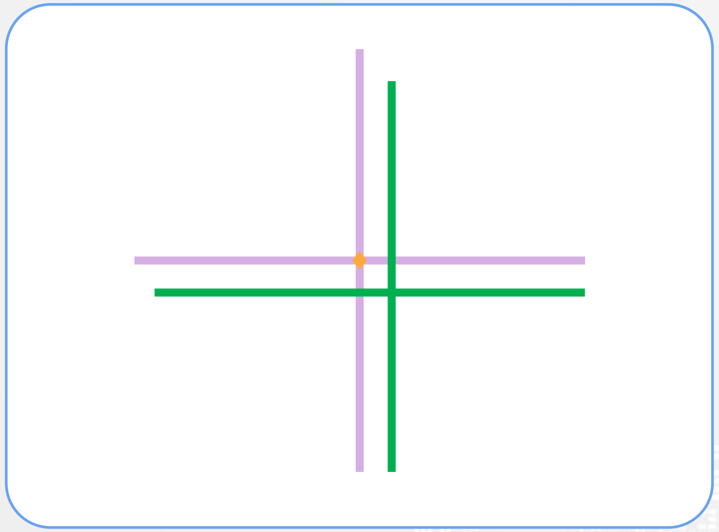




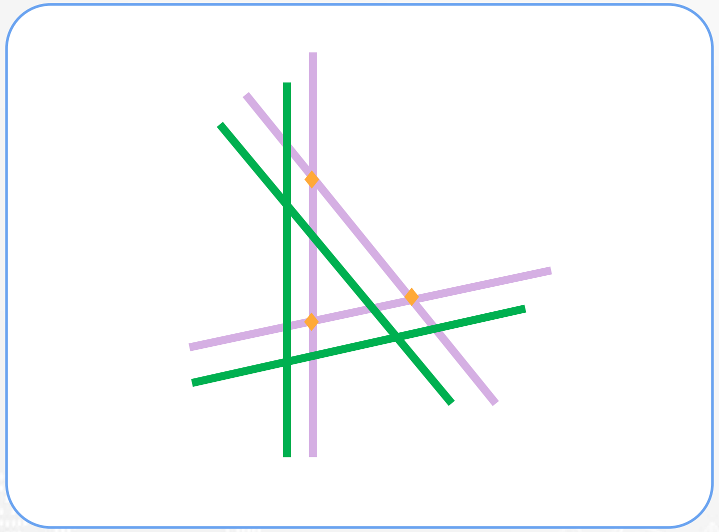
TAUTOLOGY

Deep Learning Interpretation

ถ้าเรานำ maximum plane มาทำการหักซ้ำ โดยการเพิ่ม hidden layer ที่มี 1 node เราจะได้



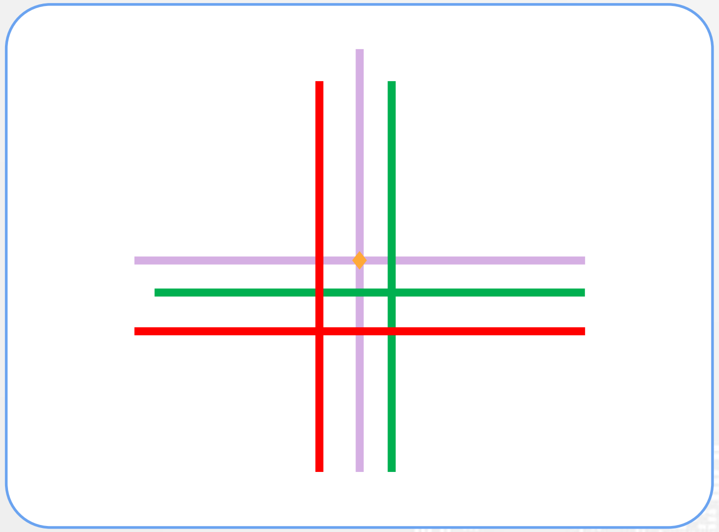
2,2,1,1



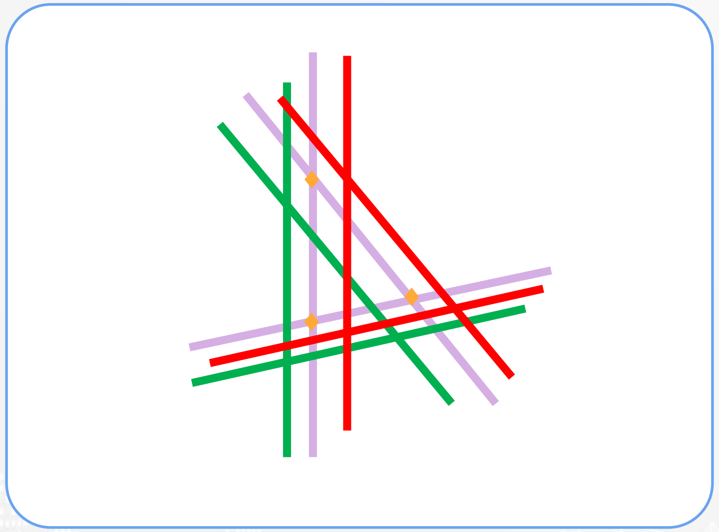
2,3,1,1

Deep Learning Interpretation

และยิ่งถ้าเราเพิ่มจำนวน node ใน hidden layer ชั้นที่ 2 เป็น 2 node เราจะได้



2,2,2,1



2,3,2,1

Deep Learning Interpretation

จากภาพเราจะเห็นว่า **การเพิ่มขึ้นของ node** ใน hidden layer

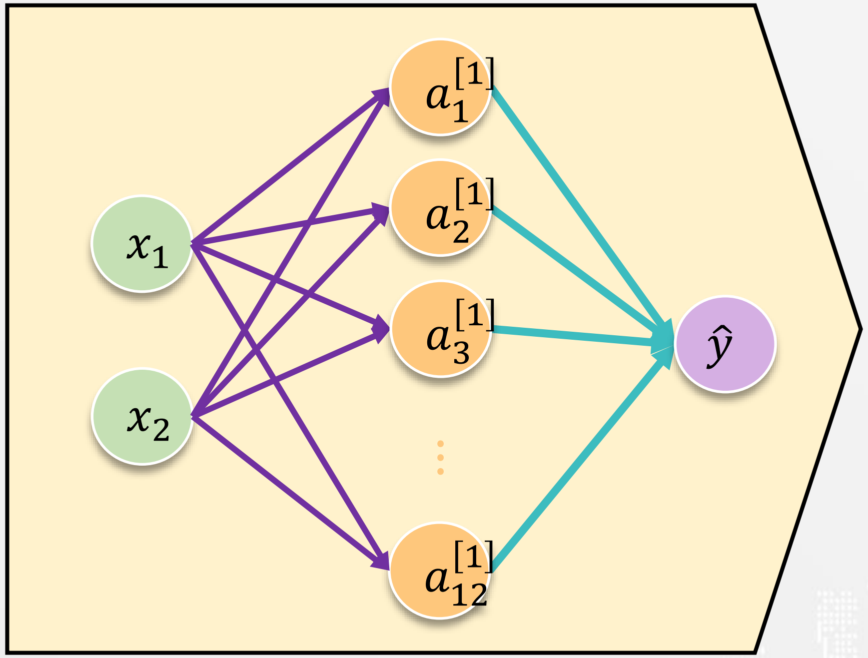
ทำให้ plane ใน output node มีการ**เพิ่มขึ้นอย่างทวีคูณ**

Deep Learning Interpretation

รอยหักใน hidden layer ที่เกิดขึ้นมานั้น ถูกสร้างขึ้นจากการนำรอยหักเดิมมาคูณด้วย weight

ทำให้ถูกจำกัดเงื่อนไขของรอยหักที่เกิดขึ้นใหม่ นั้นคือการมีลักษณะ**ขนานกับรอยหักเดิม** (ถ้ามองจาก top view)

Deep Learning Interpretation

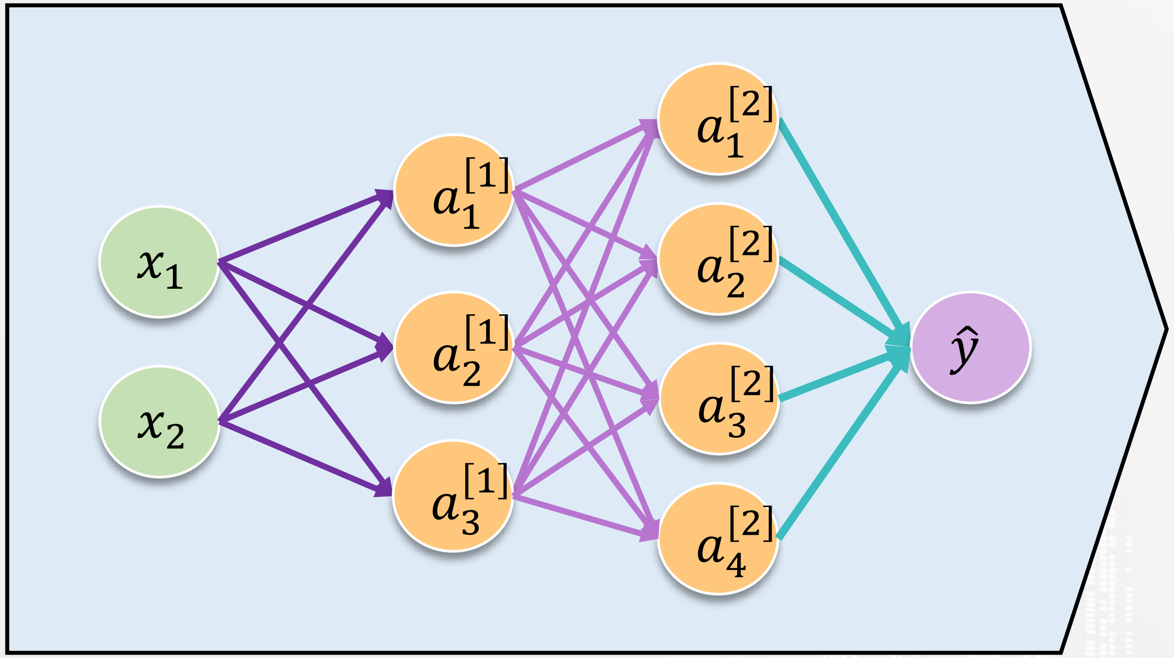


Cost ≈ (2 x 12) + (12 x 1)

= 24 +12

= 36

Deep Learning Interpretation



Cost ≈ (2 x 3) + (3 x 4) + (4 x 1)

= 6 + 12 + 4

= 22

Deep Learning Interpretation

การเพิ่ม hidden layer ทำให้เกิดการ**เพิ่มขึ้นของ plane อย่างทวีคูณ** โดยการเพิ่ม **computational cost เพียงเล็กน้อย** แต่ก็แลกมาด้วย ข้อจำกัดของององศาของรอยหักที่เกิดขึ้นใหม่

Deep Learning Interpretation

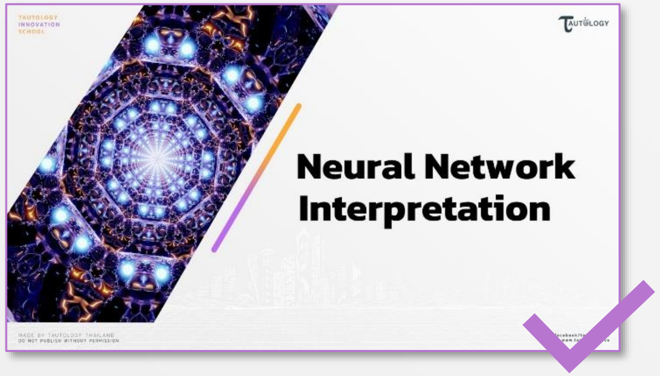
	ข้อดี	ข้อจำกัด
การเพิ่ม node	plane ที่เพิ่ม มีอิสระในการเลือกรอยหักที่เหมาะสม	จำนวน plane ที่เพิ่มน้อยเมื่อเทียบกับ cost
การเพิ่ม hidden layer	จำนวน plane เพิ่มขึ้นอย่างทวีคูณ	plane ที่เกิดขึ้นเกิดจากรอยหักเดิม

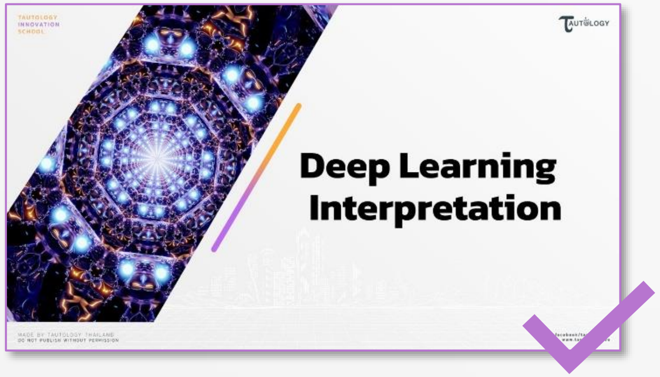
Deep Learning Interpretation

Conclusion

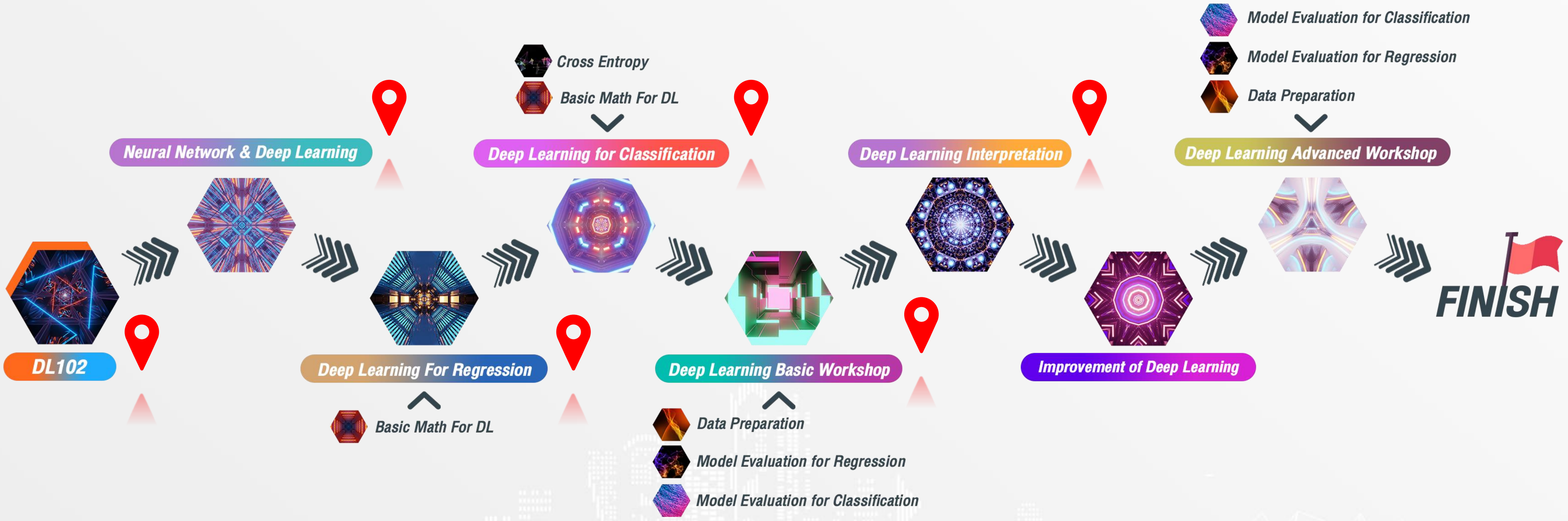
- การเพิ่มจำนวน node ทำให้เกิดการหักที่จุดที่แตกต่าง
- การเพิ่มจำนวน hidden layer ทำให้เกิดการนำจุดหักเก่ามา คูณ weight ทำหักซ้ำ ถ้ามองในมุมมอง top view ของ 3 มิติ คือการเลื่อนขนาน
- การเพิ่มจำนวน hidden layer ทำให้รอยหักเพิ่มขึ้นเป็นทวีคูณ ในขณะที่การเพิ่มจำนวน node ทำให้รอยหักเพิ่มขึ้นทีละ 1
- รอยหักที่เกิดขึ้นจากการเพิ่มจำนวน node จะเป็นอิสระ ในขณะที่ รอยหักที่เกิดขึ้นจากการเพิ่มจำนวน hidden layer จะเกิดจากการเลื่อนขนานรอยหักในชั้นก่อนหน้า
- การเพิ่ม hidden layer ทำให้ได้รอยหักที่เพิ่มขึ้นแบบทวีคูณได้โดยที่ cost ใช้เพิ่มไม่เยอะ

Deep Learning Interpretation





DL103 : Deep Learning



TAUTOLOGY
INNOVATION
SCHOOL

TAUTOLOGY

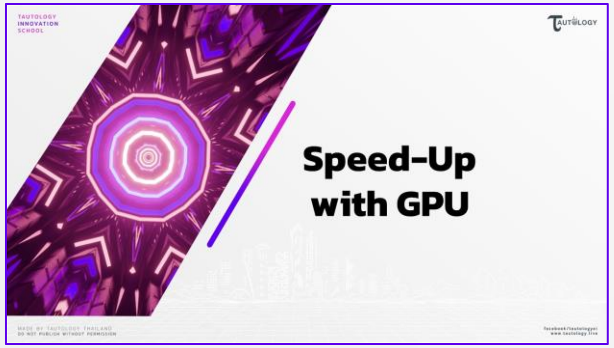
IMPROVEMENT OF DEEP LEARNING

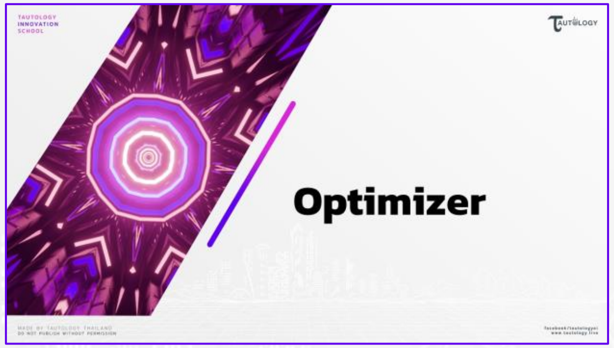
IMPROVEMENT OF DEEP LEARNING

BY TAUTOLOGY

facebook/tautologyai
www.tautology.live

Improvement of Deep Learning





Speed-Up with GPU

Speed-Up with GPU

What is GPU?

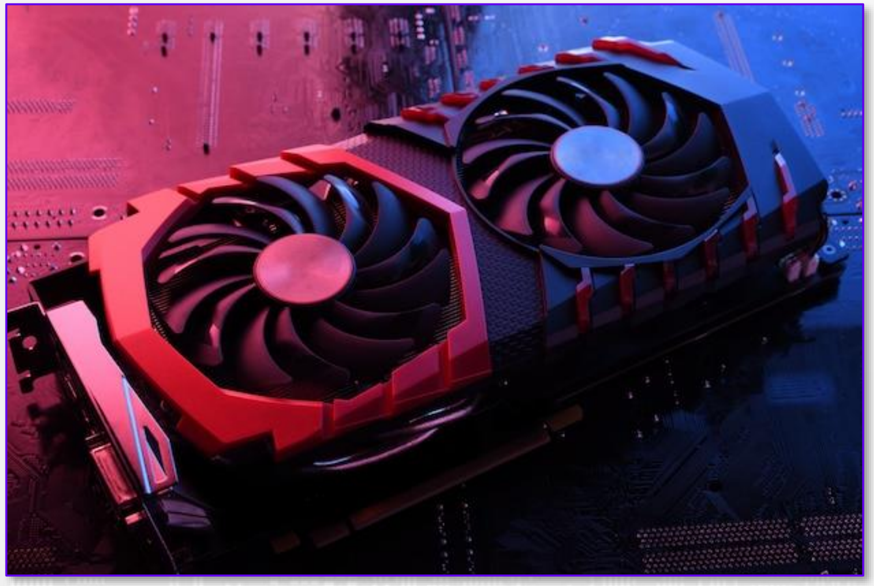
How GPU Accelerate DL?

Welcome to Colab

Train Model with GPU on Colab

What is GPU?

GPU (Graphic Processing Unit) คือ หน่วยประมวลผลกราฟฟิค ซึ่งทำหน้าที่ประมูลผล ภาพ/วิดีโอ/โมเดล 3 มิติ



Speed-Up with GPU

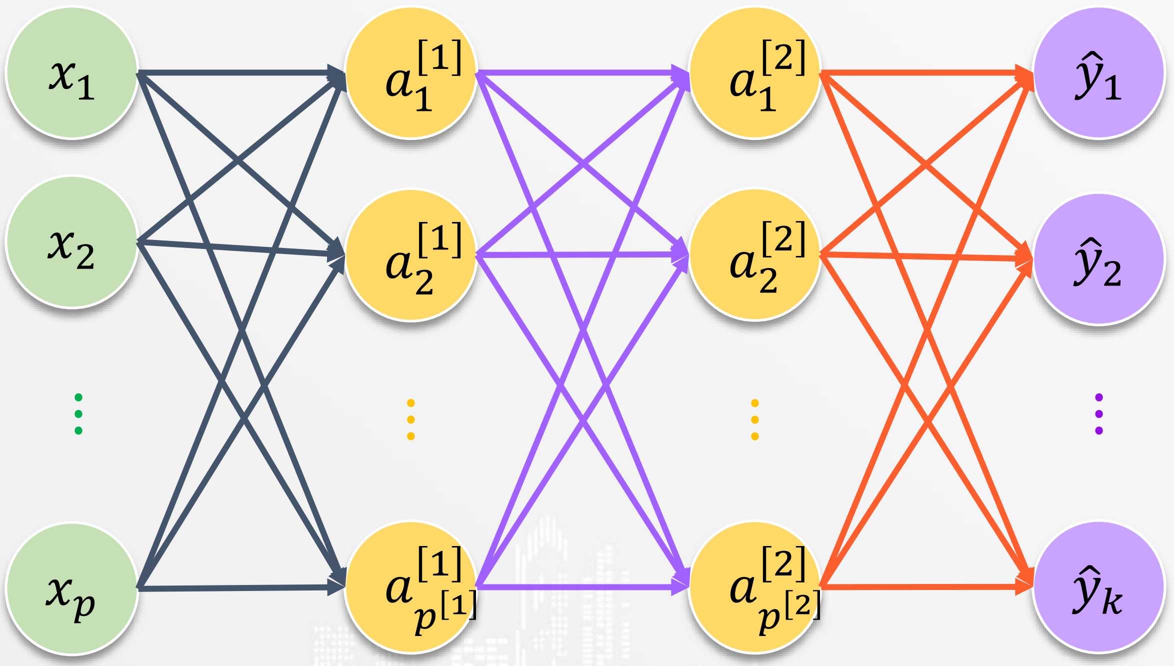
What is GPU?

How GPU Accelerate DL?

Welcome to Colab

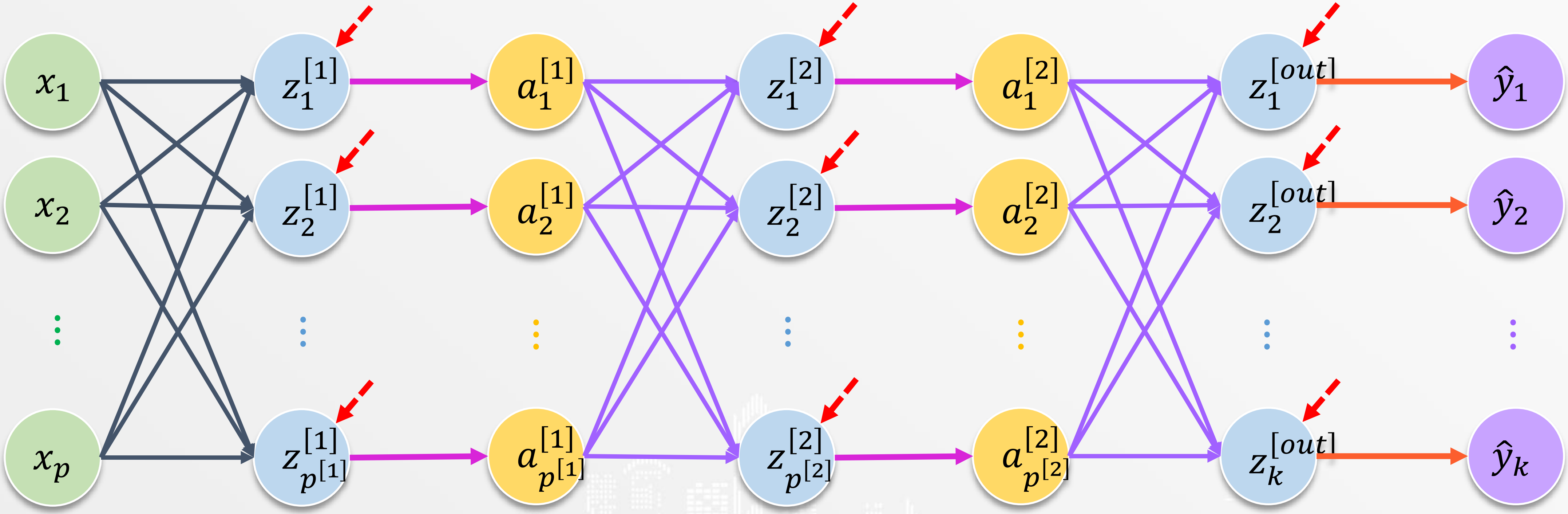
Train Model with GPU on Colab

How GPU Accelerate DL?



architecture ของ deep learning สำหรับปัญหา multi-class classification ที่มี hidden layer 2 ชั้น

How GPU Accelerate DL?



architecture ของ deep learning สำหรับปัญหา multi-class classification ที่มี hidden layer 2 ชั้น

How GPU Accelerate DL?

$$X \rightarrow Z^{[1]} \rightarrow A^{[1]} \rightarrow Z^{[2]} \rightarrow A^{[2]} \rightarrow Z^{[out]} \rightarrow \hat{Y}$$

How GPU Accelerate DL?

$$X \rightarrow Z^{[1]} \rightarrow A^{[1]} \rightarrow Z^{[2]} \rightarrow A^{[2]} \rightarrow Z^{[out]} \rightarrow A^{[out]}$$

How GPU Accelerate DL?

$$Z^{[1]} = XW^{[1]} + b^{[1]}$$

$$Z^{[2]} = A^{[1]}W^{[2]} + b^{[2]}$$

$$Z^{[out]} = A^{[2]}W^{[out]} + b^{[out]}$$

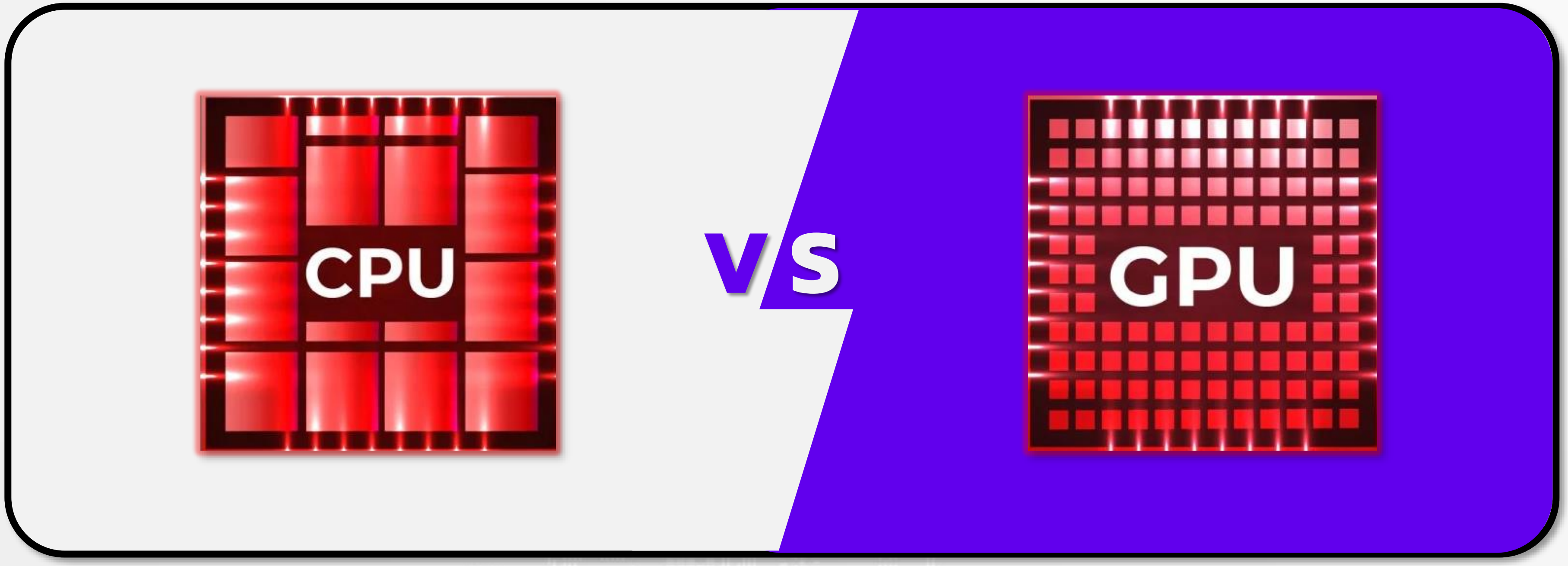
How GPU Accelerate DL?

การสร้าง deep learning คือ
มหกรรมแห่งการคูณ matrix

How GPU Accelerate DL?

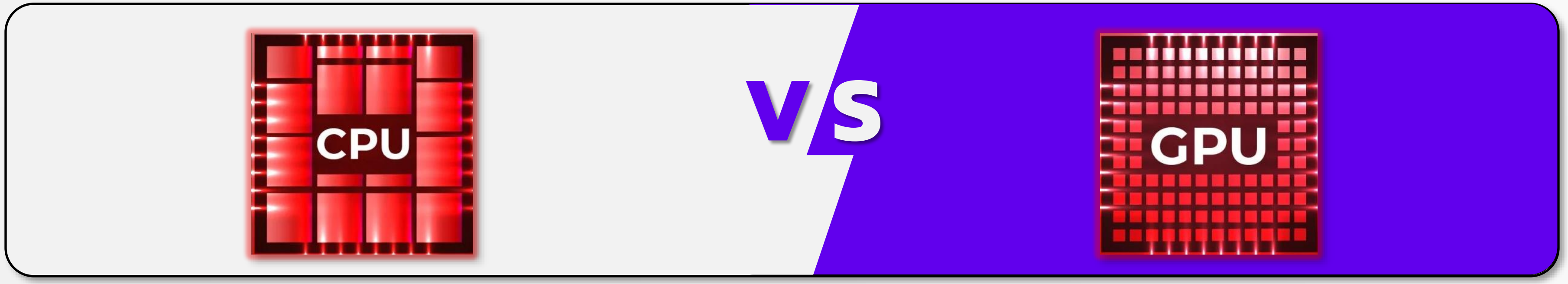
GPU คูณ matrix ได้เร็วกว่า **CPU**

How GPU Accelerate DL?



ref : cgdirector.com

How GPU Accelerate DL?



- CPUs have few strong cores
- Suited for serial workloads
- Designed for general purpose calculation

- GPUs have thousands of weaker cores
- Suited for parallel workloads
- Specialize in graphics processing

How GPU Accelerate DL?

"GPU"

How GPU Accelerate DL?

Way to use GPU

On your Notebook

On Cloud

How GPU Accelerate DL?

Way to use GPU

Google Colaboratory

Speed-Up with GPU

What is GPU?

How GPU Accelerate DL?

Welcome to Colab

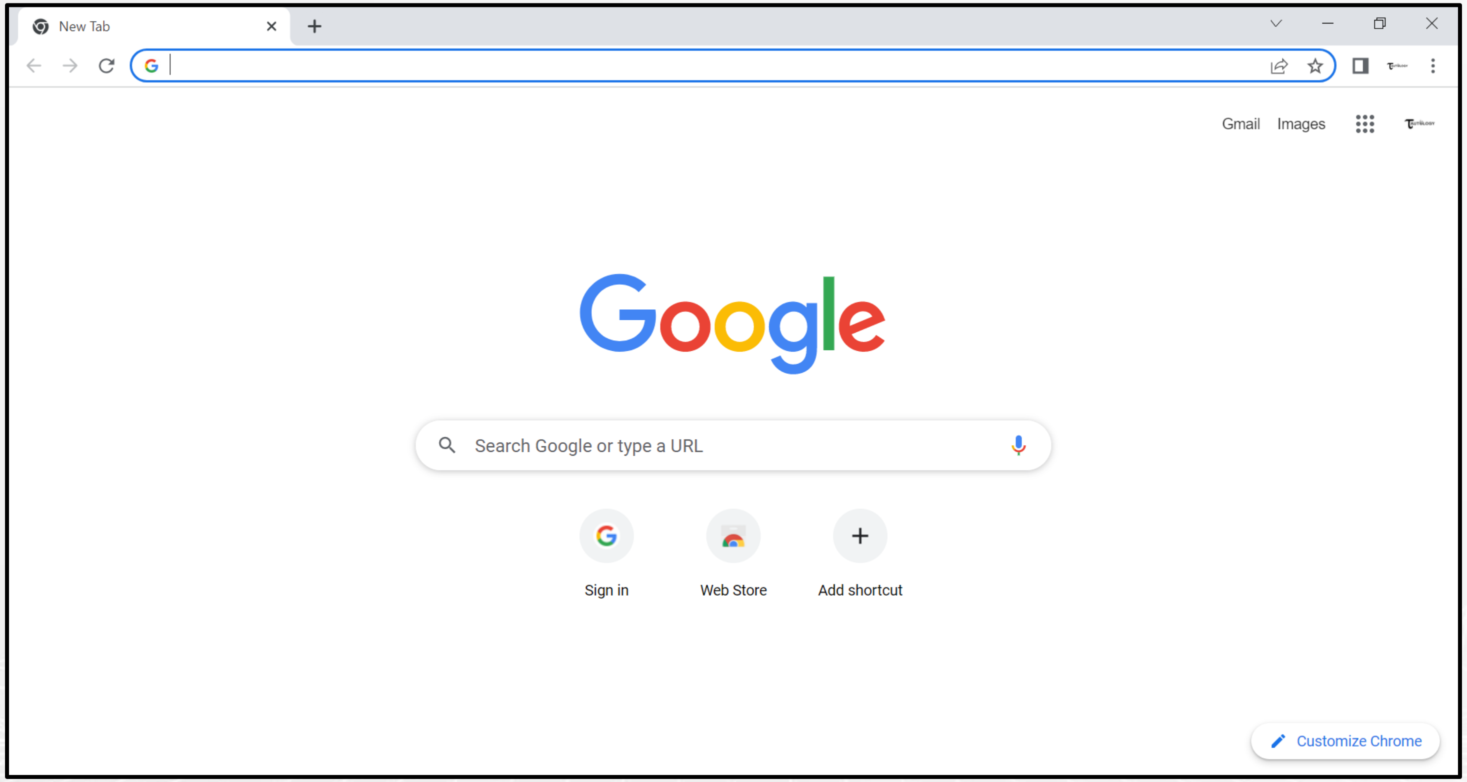
Train Model with GPU on Colab

Welcome to Colab

- ☐ Getting Started with Colab
- ☐ Upload Notebook from Local Drive
- ☐ Import Data to Colab

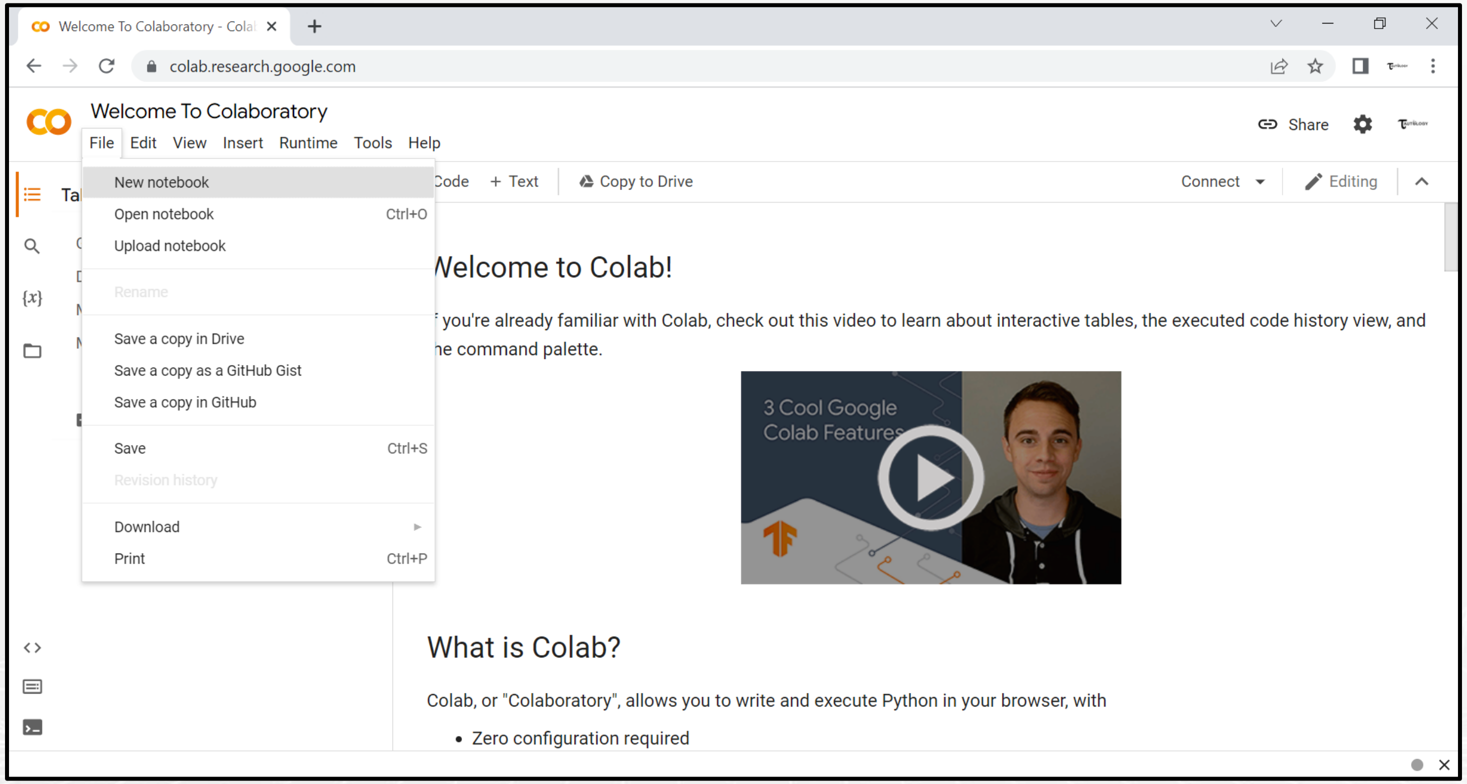
Getting Started with Colab

Step 1 : Open browser



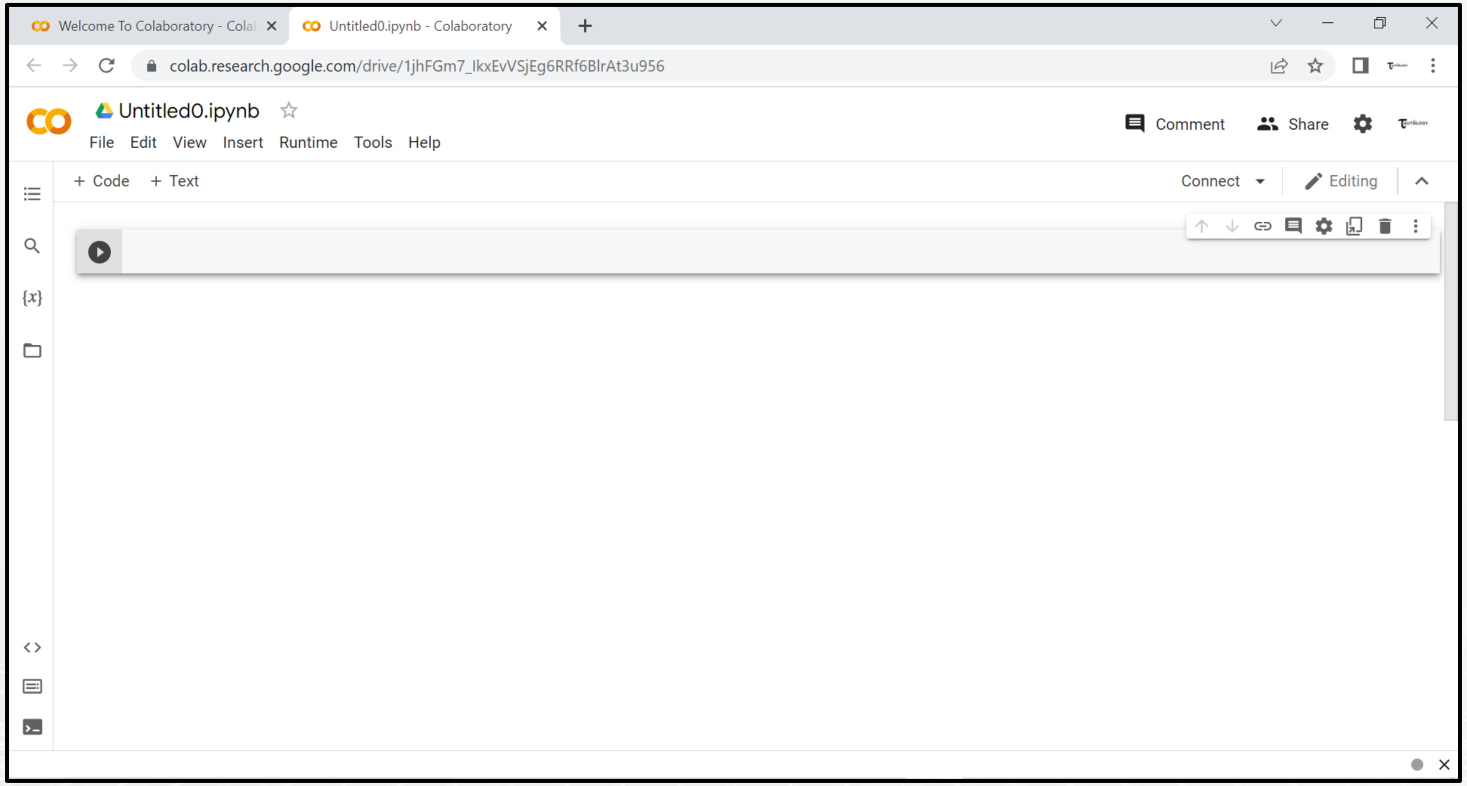
Getting Started with Colab

Step 2 : https://colab.research.google.com



Getting Started with Colab

Step 3 : New notebook

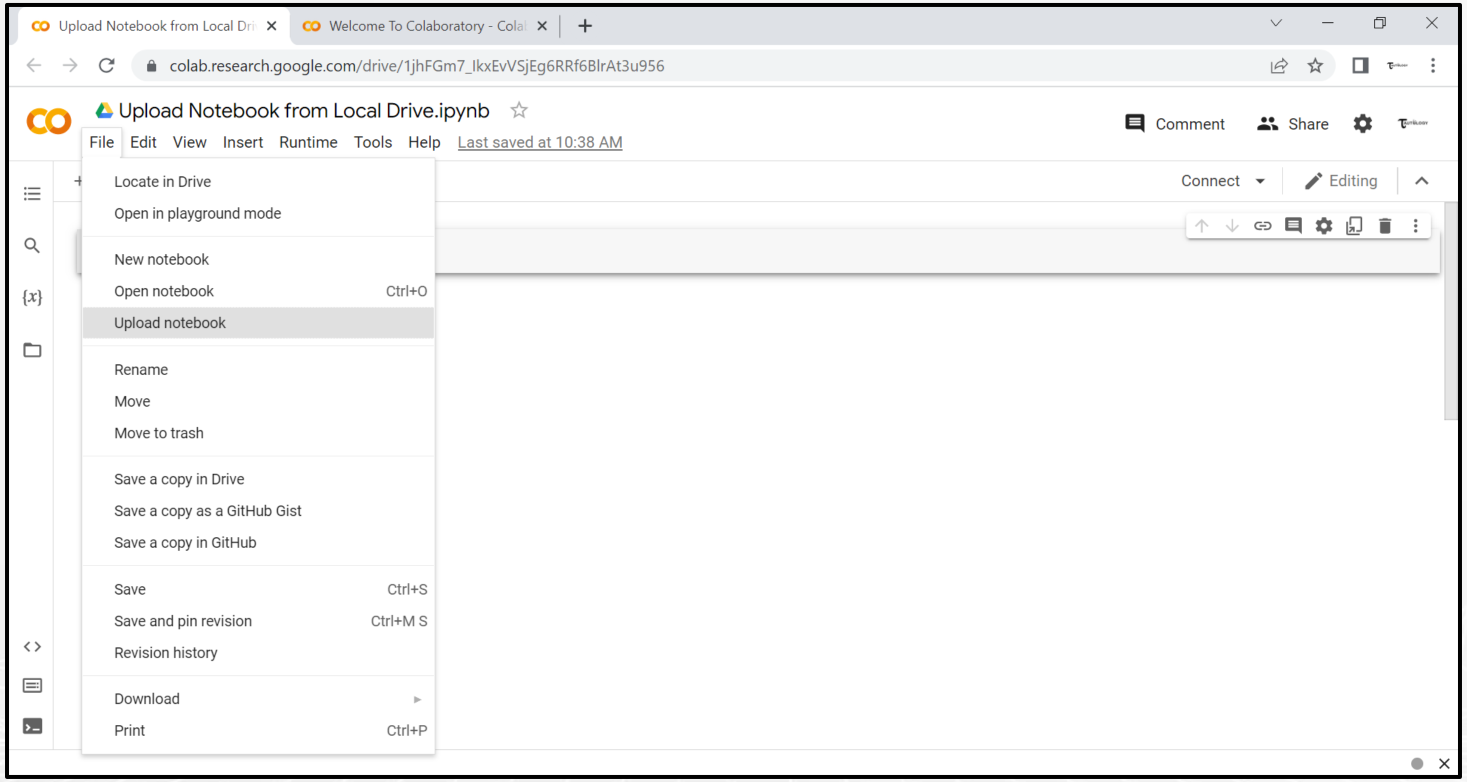


Welcome to Colab

- ☑ **Getting Started with Colab**
- ☐ Upload Notebook from Local Drive
- ☐ Import Data to Colab

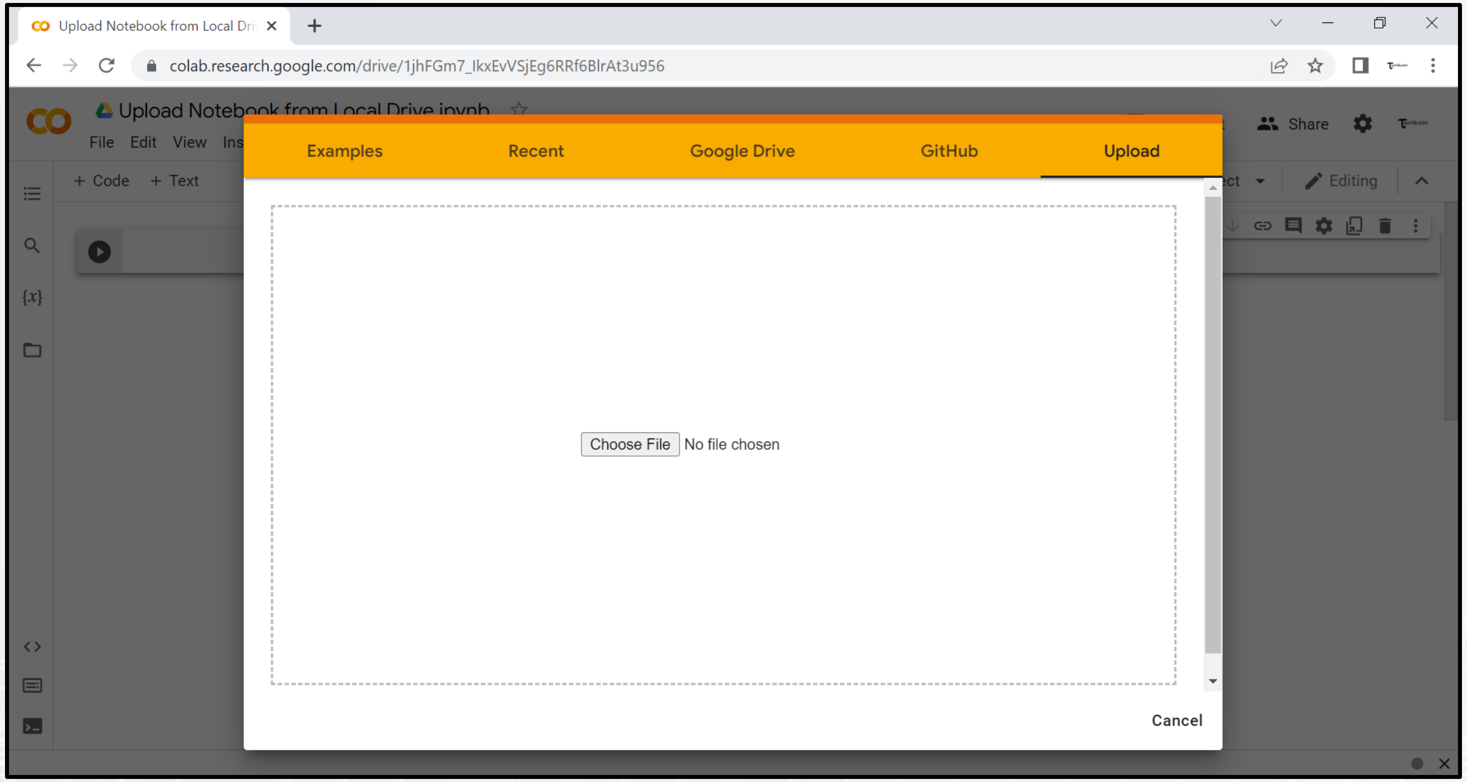
Upload Notebook from Local Drive

Step 1 : Upload



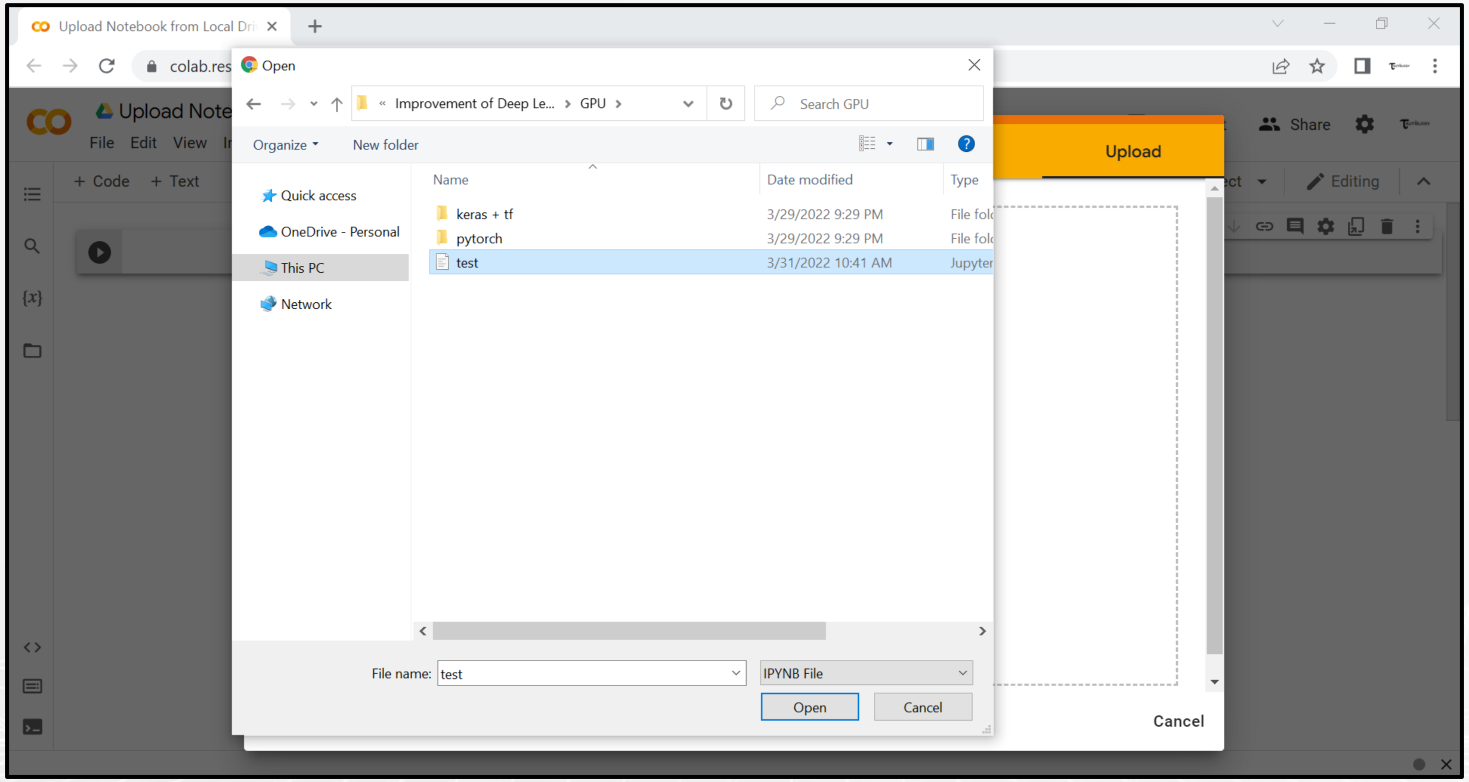
Upload Notebook from Local Drive

Step 1 : Upload



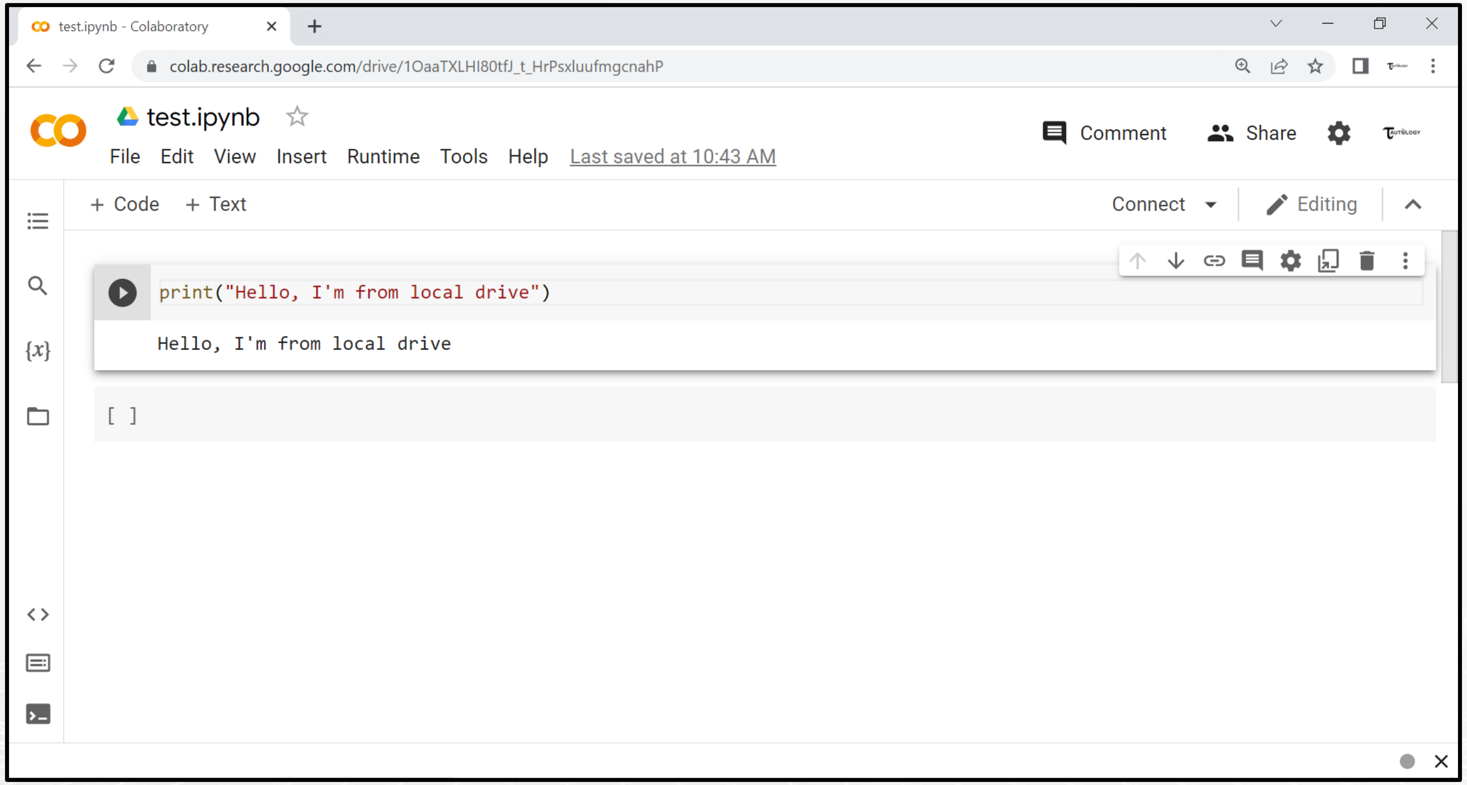
Upload Notebook from Local Drive

Step 2 : Choose File



Upload Notebook from Local Drive

Step 3 : Run



Welcome to Colab

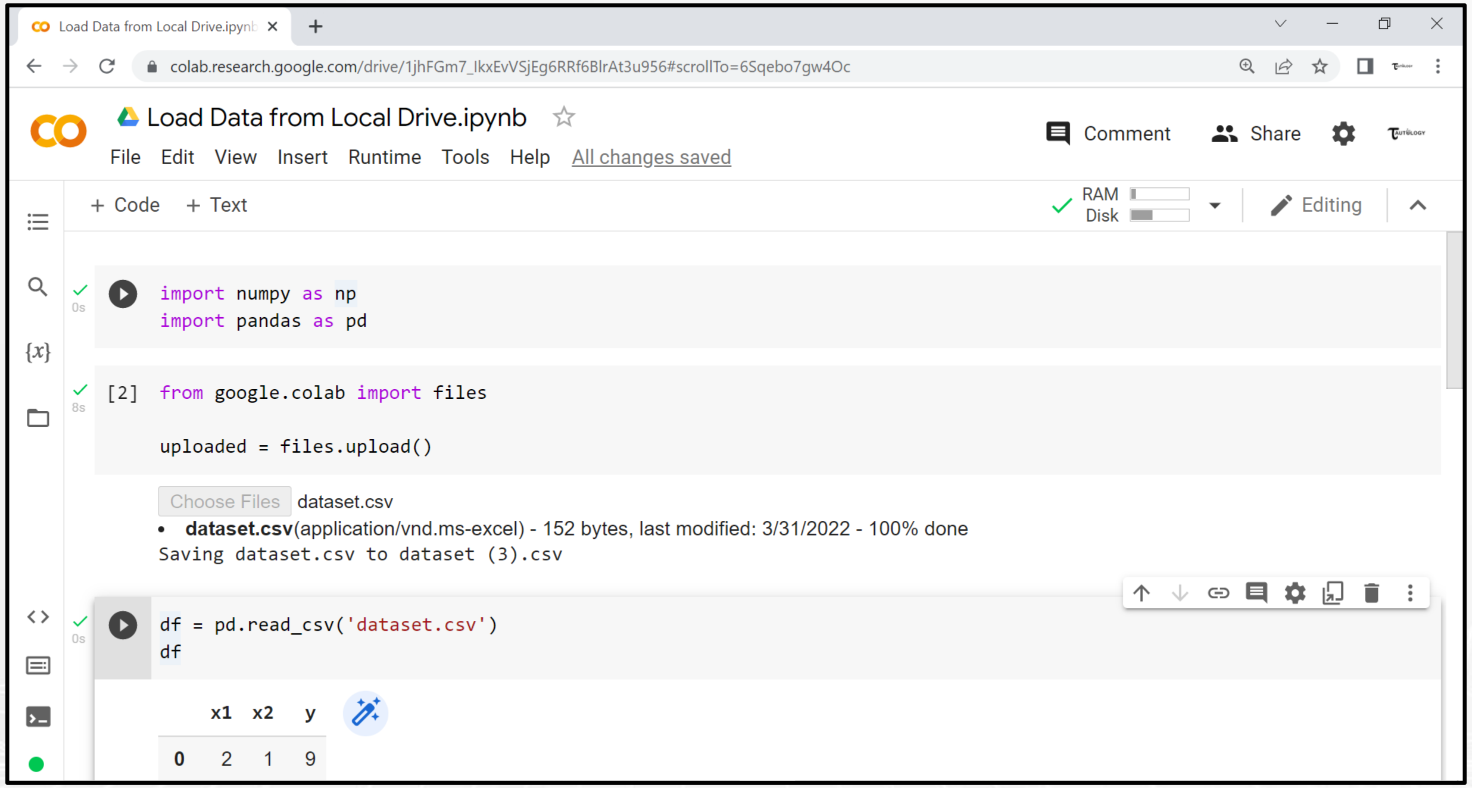
- ☑ **Getting Started with Colab**
- ☑ **Upload Notebook from Local Drive**
- ☐ Import Data to Colab

Import Data to Colab

- Load Data from Local Drive
- Load Data from Google Drive

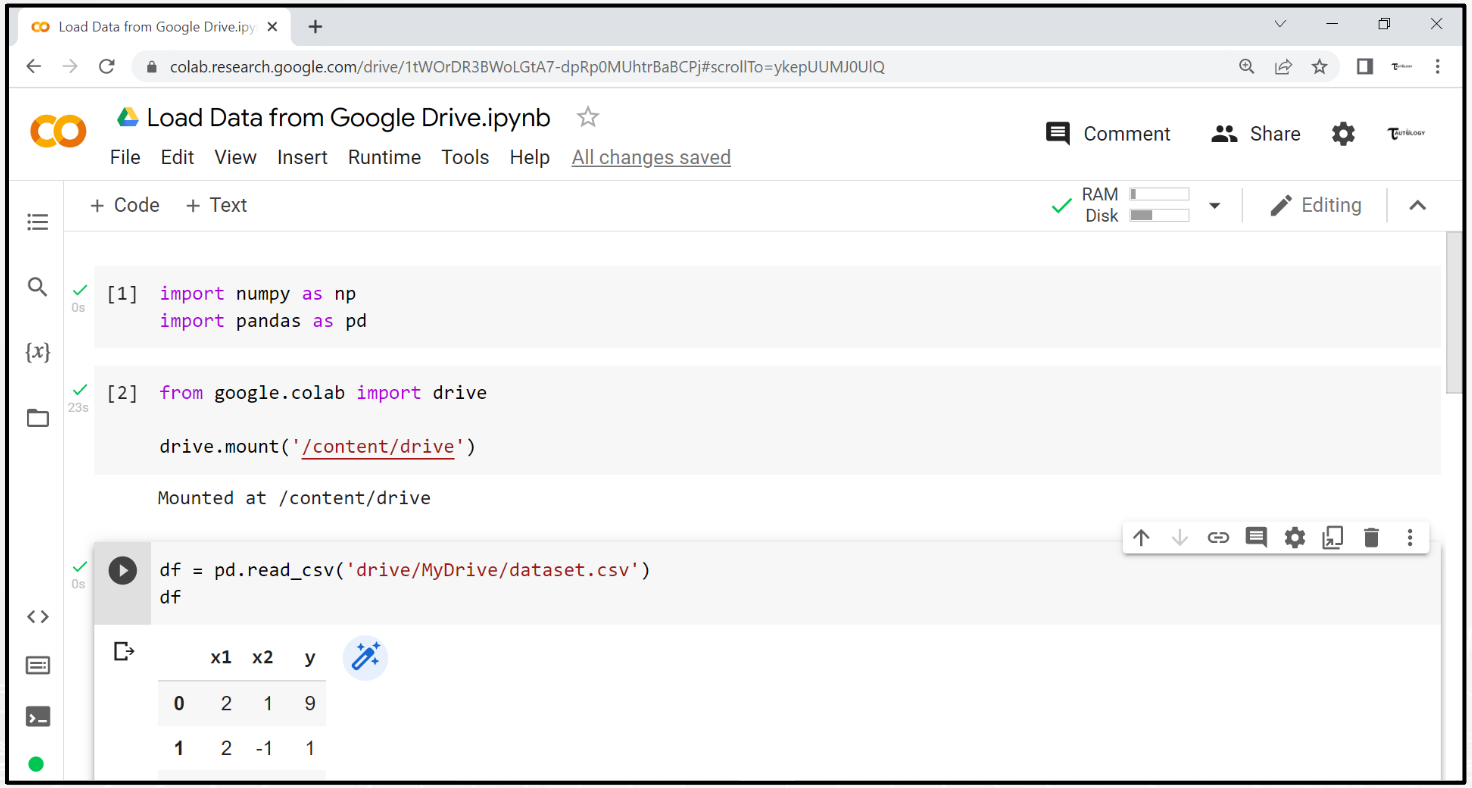
Import Data to Colab

▪ Load Data from Local Drive



Import Data to Colab

▪ Load Data from Google Drive



Welcome to Colab

- ☑ Getting Started with Colab
- ☑ Upload Notebook from Local Drive
- ☑ Import Data to Colab

Speed-Up with GPU

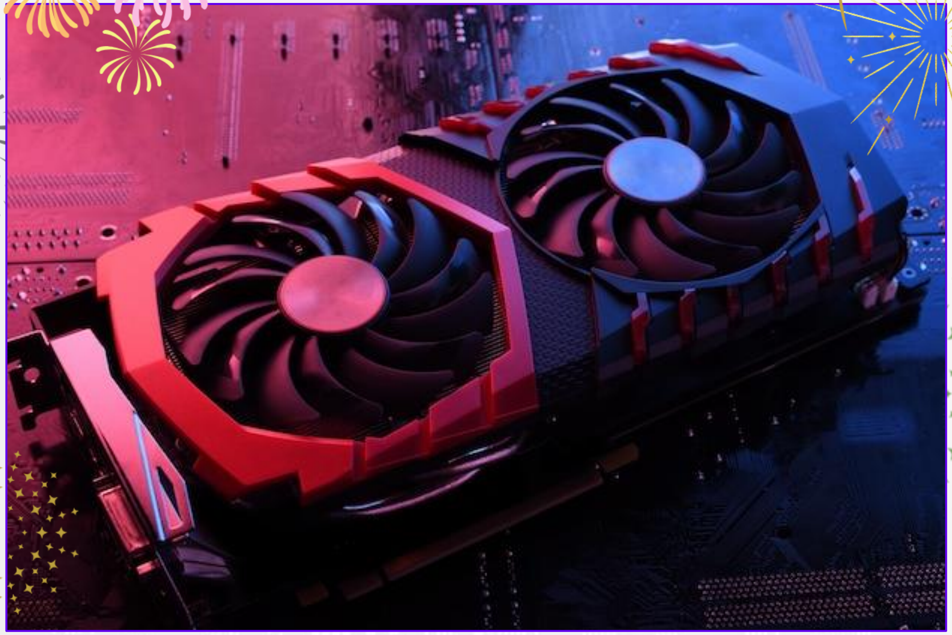
What is GPU?

How GPU Accelerate DL?

Welcome to Colab

Train Model with GPU on Colab

Train Model with GPU on Colab



Train Model with GPU on Colab

Keras + Tensorflow

Pytorch

Train Model with GPU on Colab

```
tf.config.list_physical_devices()
```

```
[PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU'),
 PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
devices_type = [device.device_type for device in tf.config.list_physical_devices()]

devices_type
```

```
['CPU', 'GPU']
```

```
if 'GPU' in devices_type:
  device_name = '/device:GPU:0'
else:
  device_name = '/device:CPU:0'
```

Train Model with GPU on Colab

```
model = Sequential()

model.add(Dense(5, input_dim=X.shape[1], activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation=None))

model.compile(loss=MeanSquaredError(), optimizer=SGD(learning_rate=0.01))
```

```
tqdm_callback = tfa.callbacks.TQDMProgressBar(show_epoch_progress=False)

with tf.device(device_name):
  history = model.fit(X, y, epochs=500, callbacks=[tqdm_callback])
```

Train Model with GPU on Colab

Code for this section

Open Link

https://colab.research.google.com/drive/104VDWLauVCY2scmmmcjR0QxbX0I0zM7C?usp=sharing

Train Model with GPU on Colab

```python
if torch.cuda.is_available():
  device = 'cuda:0'
else:
  device = 'cpu'
```

```python
X = torch.from_numpy(X).float().to(device)
y = torch.from_numpy(y).float().to(device)
```

Train Model with GPU on Colab

```
layers = []

layers.append(nn.Linear(X.shape[1], 5))
layers.append(nn.ReLU())

layers.append(nn.Linear(5, 4))
layers.append(nn.ReLU())

layers.append(nn.Linear(4, 1))

model = nn.Sequential(*layers).to(device)
```

Train Model with GPU on Colab

Code for this section

Open Link

https://colab.research.google.com/drive/1ijjtDlknBljvsf8n1-r6l3wYAPYoEV9c?usp=sharing

Speed-Up with GPU

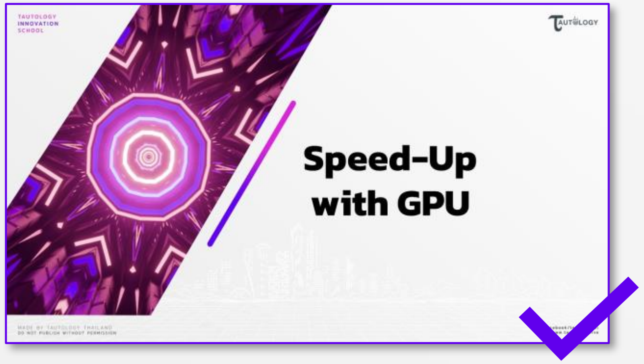
What is GPU?

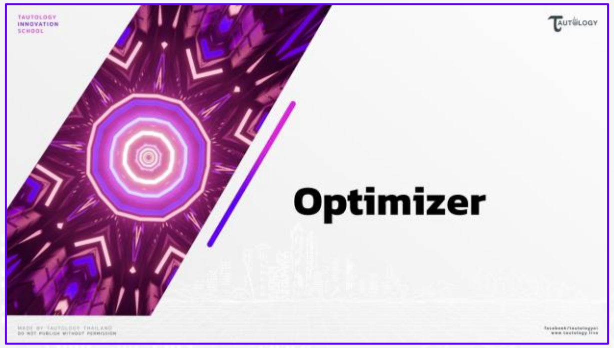
How GPU Accelerate DL?

Welcome to Colab

Train Model with GPU on Colab

Improvement of Deep Learning





Imbalanced Class

Imbalanced Class

Problem with Imbalanced Class

Solution

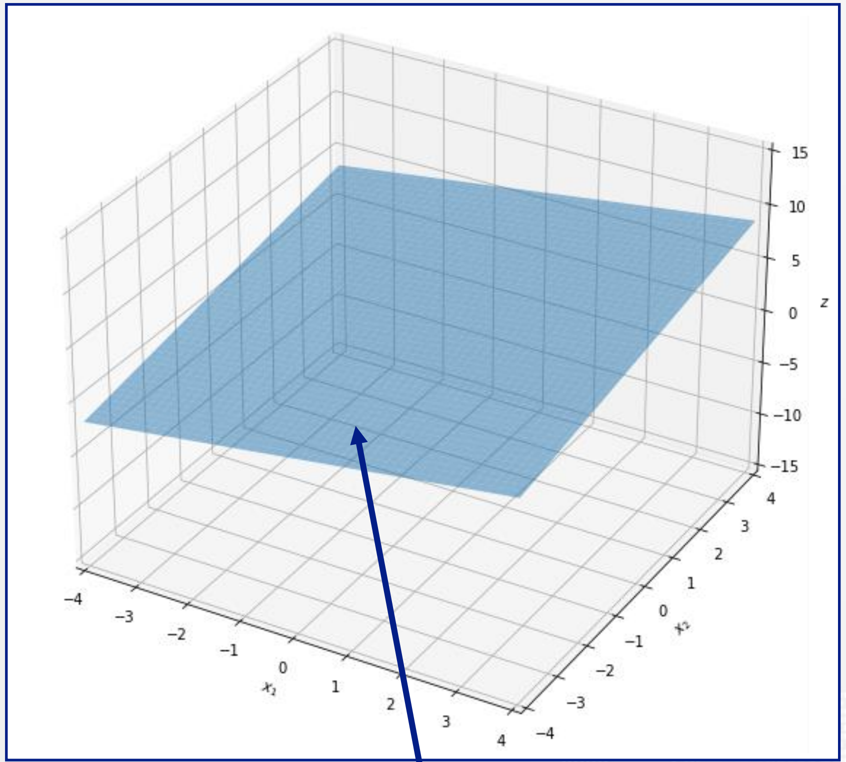
Code

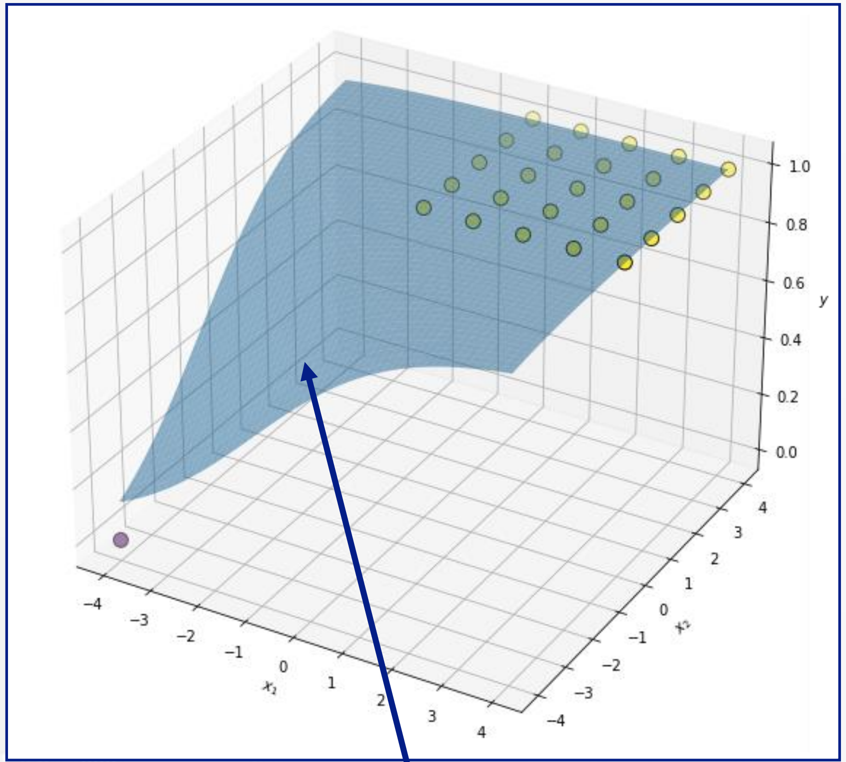
Problem with Imbalanced Class

x_1	x_2	y
0	0	1
0	1	1
0	2	1
0	3	1
0	4	1
1	0	1
1	1	1
1	2	1
1	3	1
1	4	1
2	0	1
2	1	1
2	2	1

x_1	x_2	y
2	3	1
2	4	1
3	0	1
3	1	1
3	2	1
3	3	1
3	4	1
4	0	1
4	1	1
4	2	1
4	3	1
4	4	1
-4	-4	0

Problem with Imbalanced Class

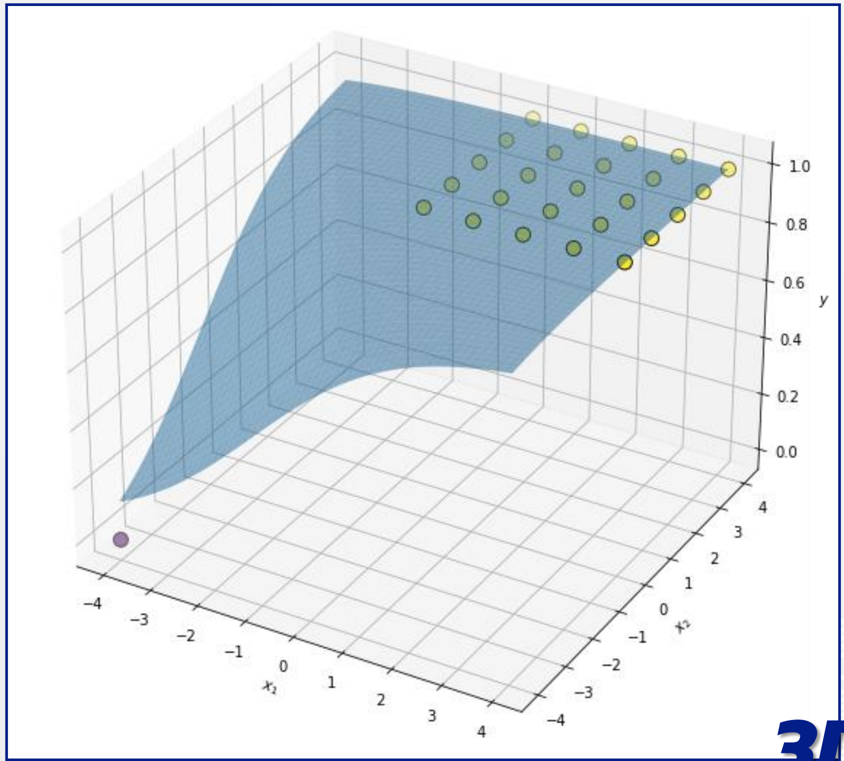




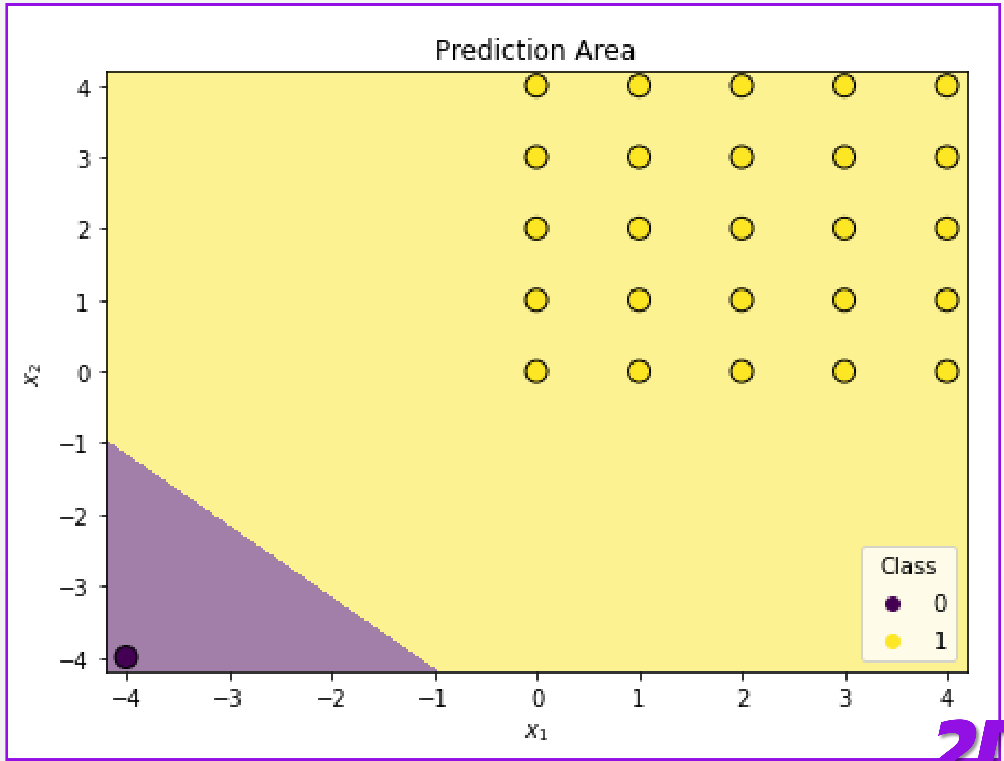
$$z = w_0 + w_1x_1 + w_2x_2$$

$$\hat{y} = \sigma(z)$$

Problem with Imbalanced Class

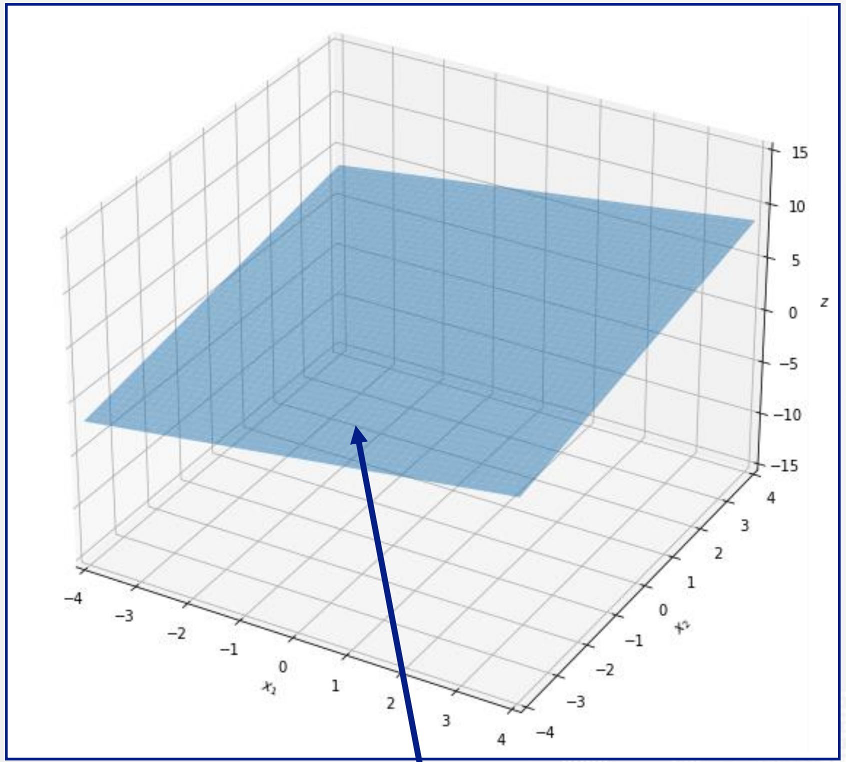


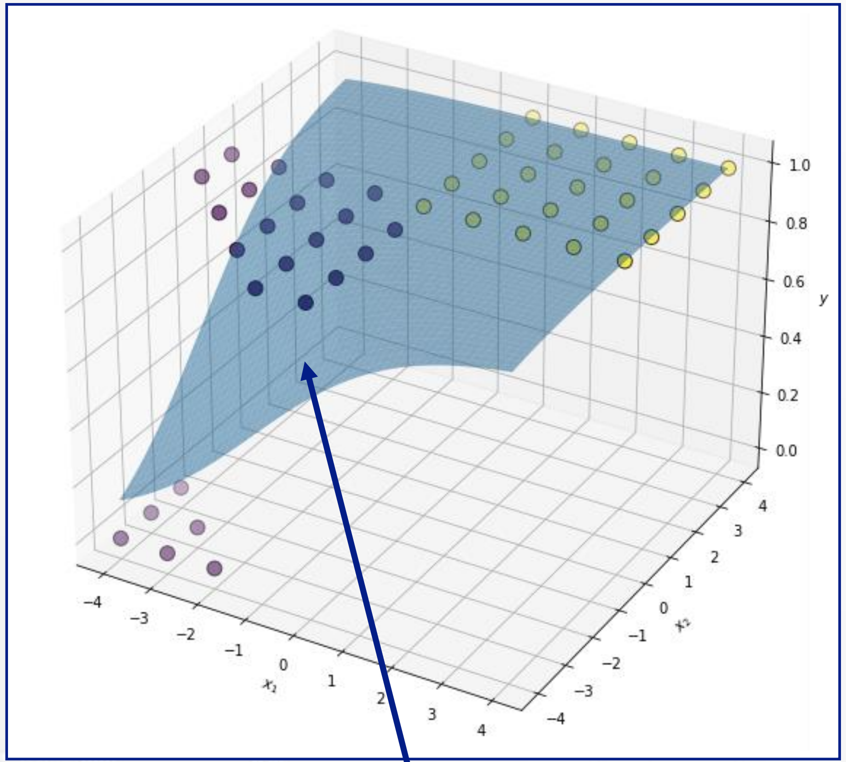
3D



2D

Problem with Imbalanced Class

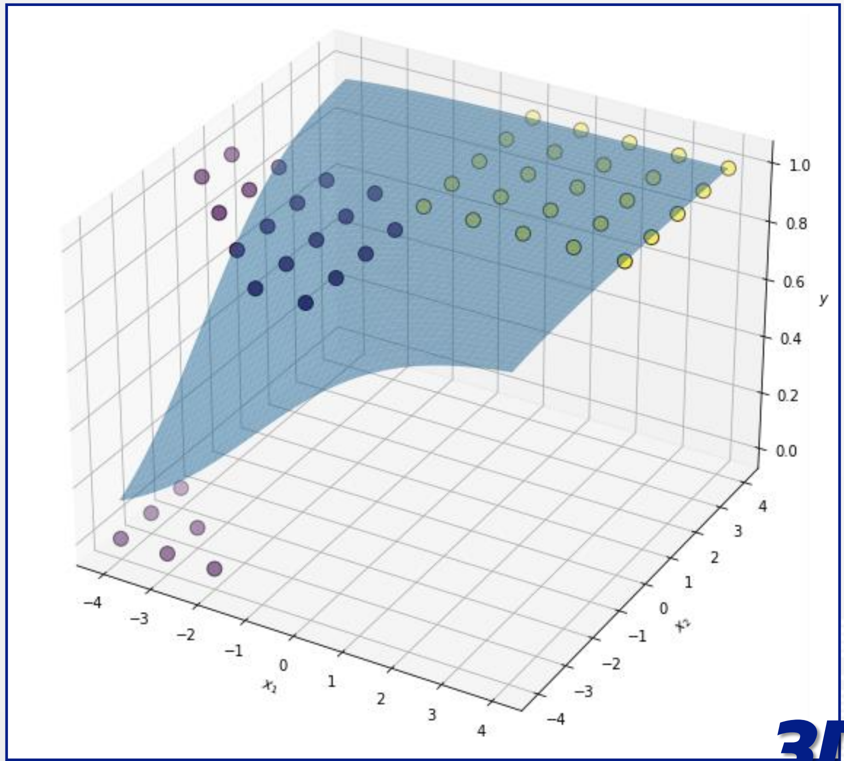




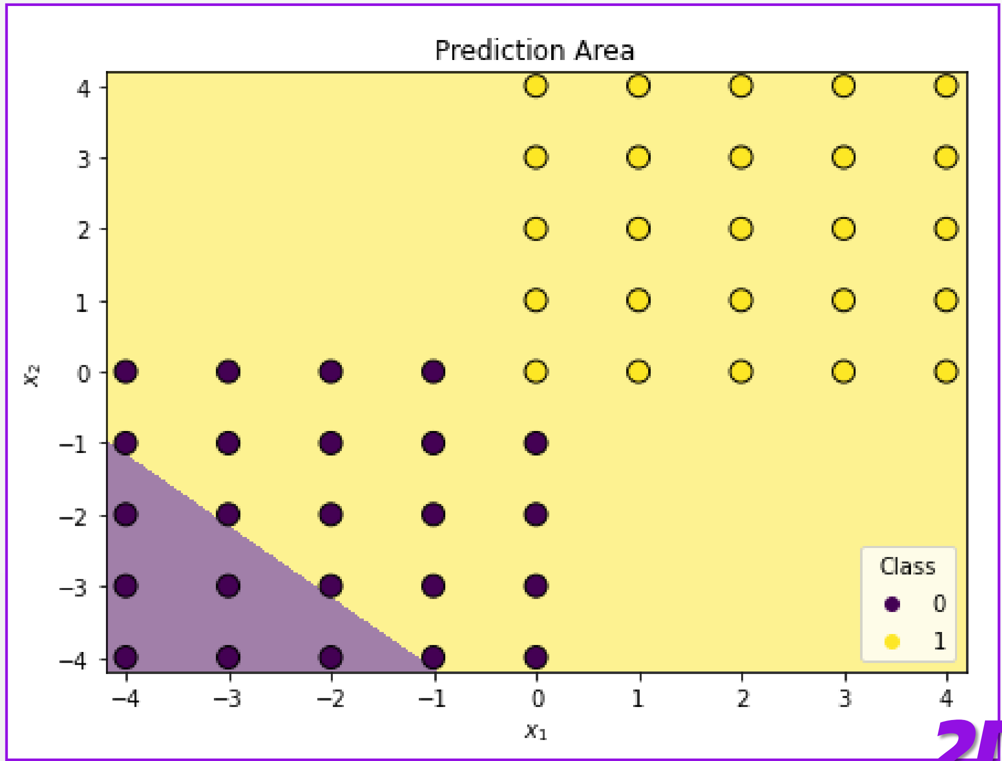
$z = w_0 + w_1 x_1 + w_2 x_2$

$\hat{y} = \sigma(z)$

Problem with Imbalanced Class

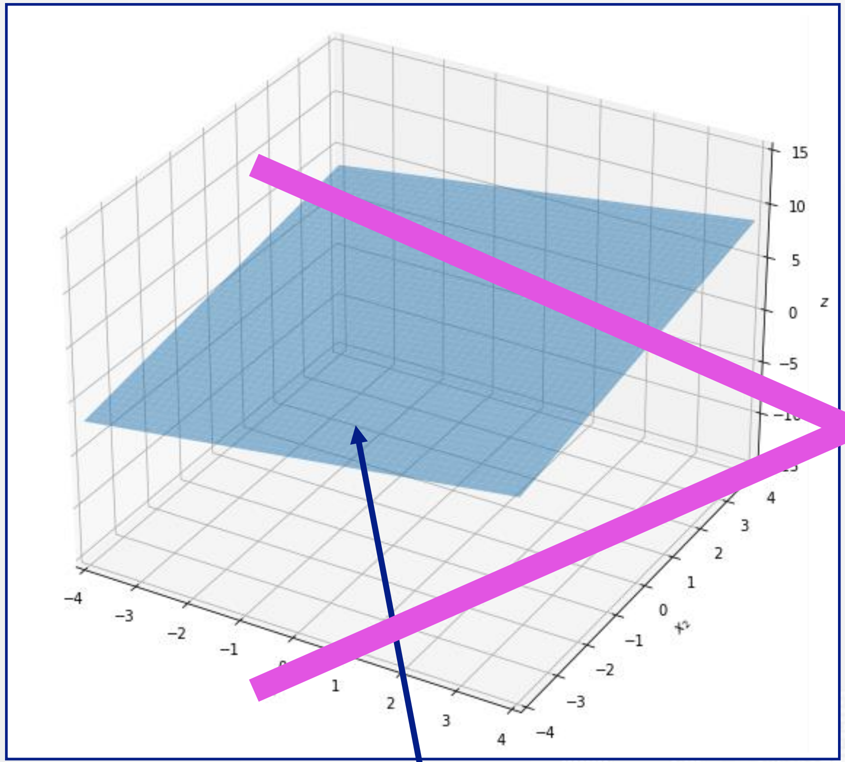


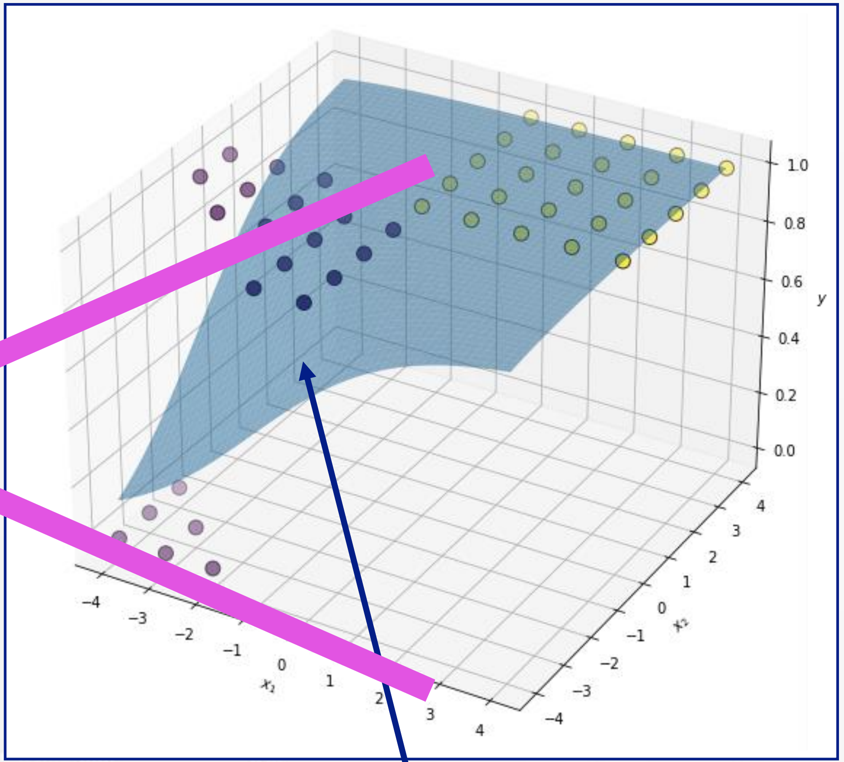
3D



2D

Problem with Imbalanced Class

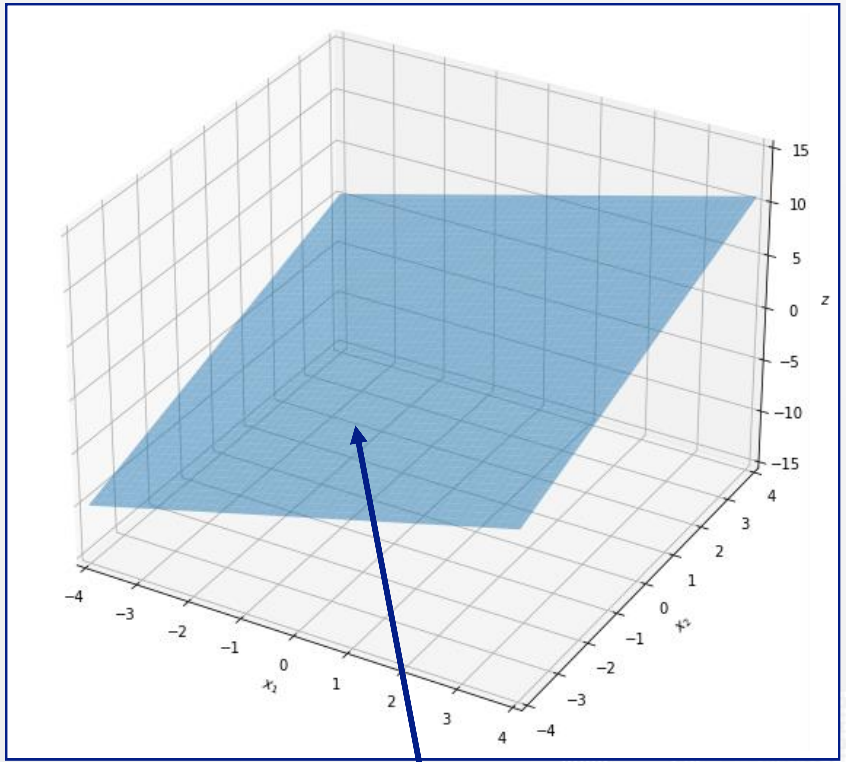




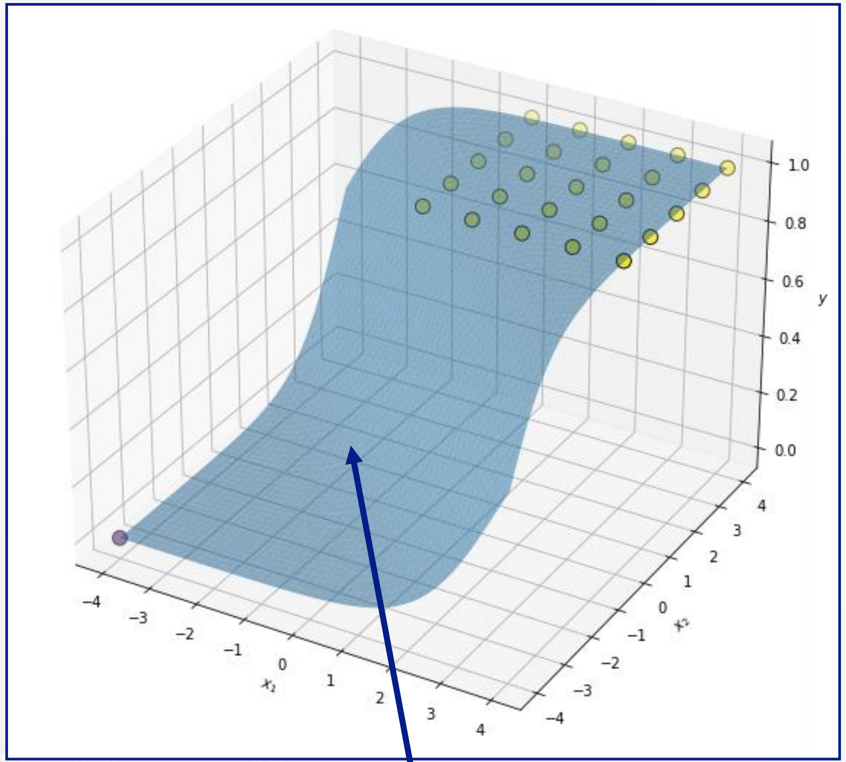
$z = w_0 + w_1x_1 + w_2x_2$

$\hat{y} = \sigma(z)$

Problem with Imbalanced Class

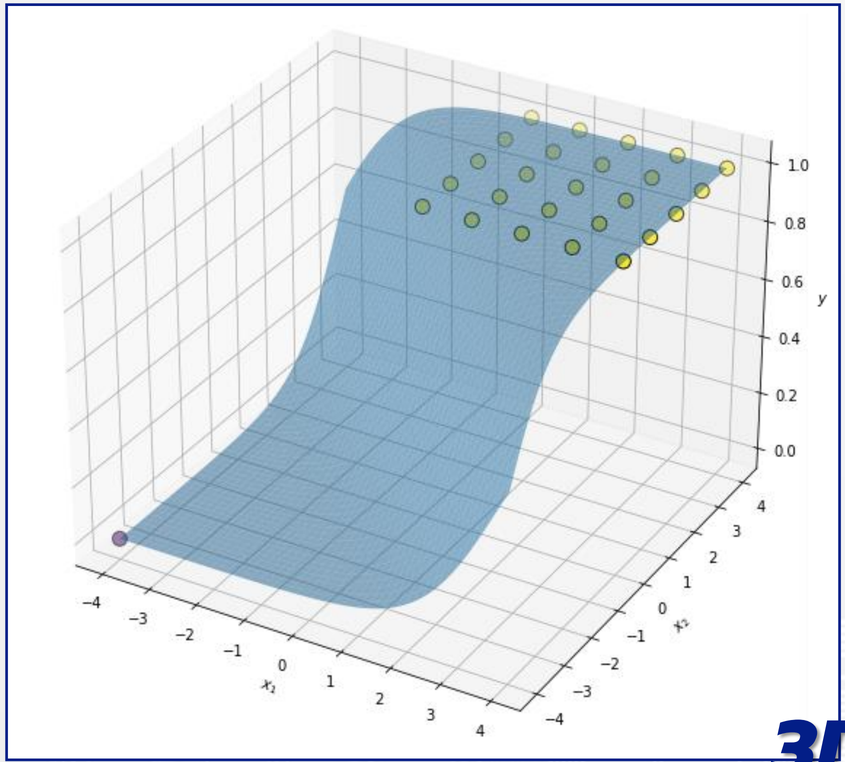


$z = w_0 + w_1 x_1 + w_2 x_2$

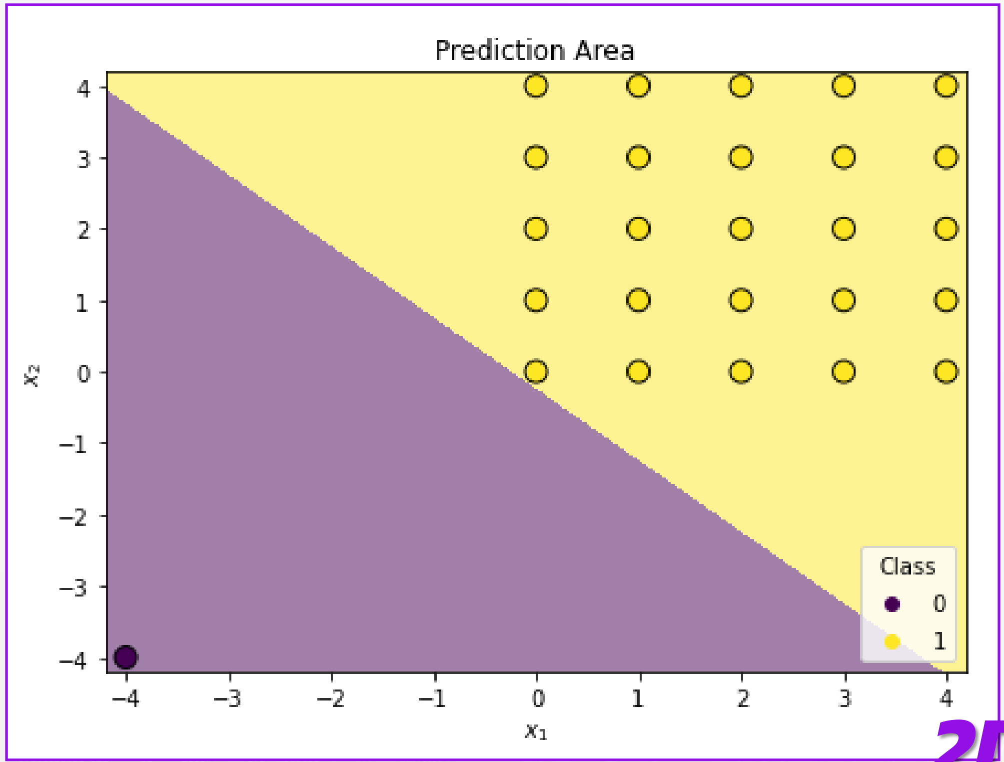


$\hat{y} = \sigma(z)$

Problem with Imbalanced Class

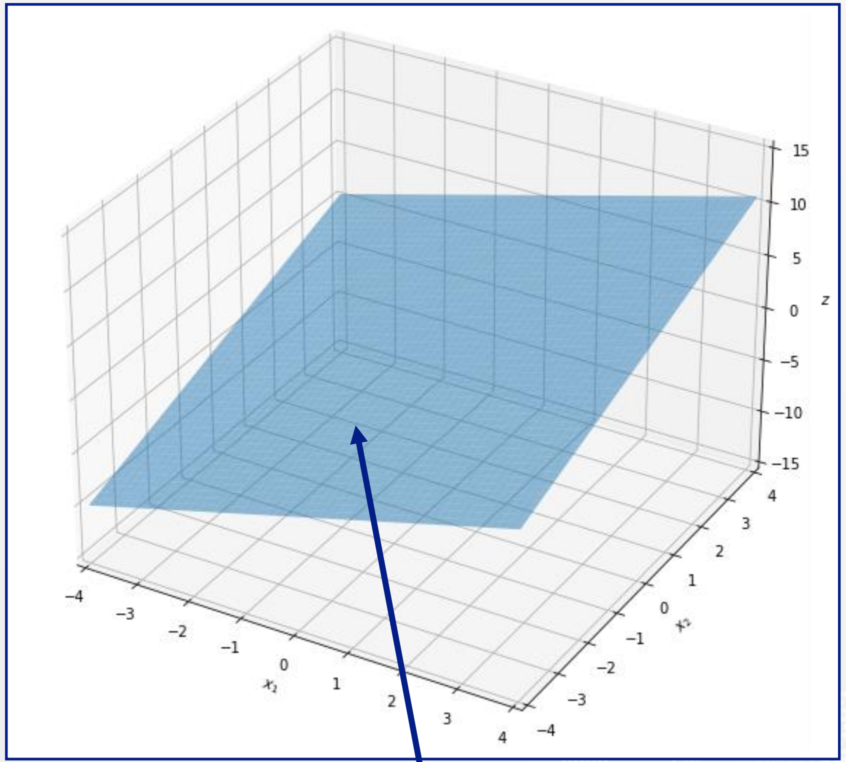


3D

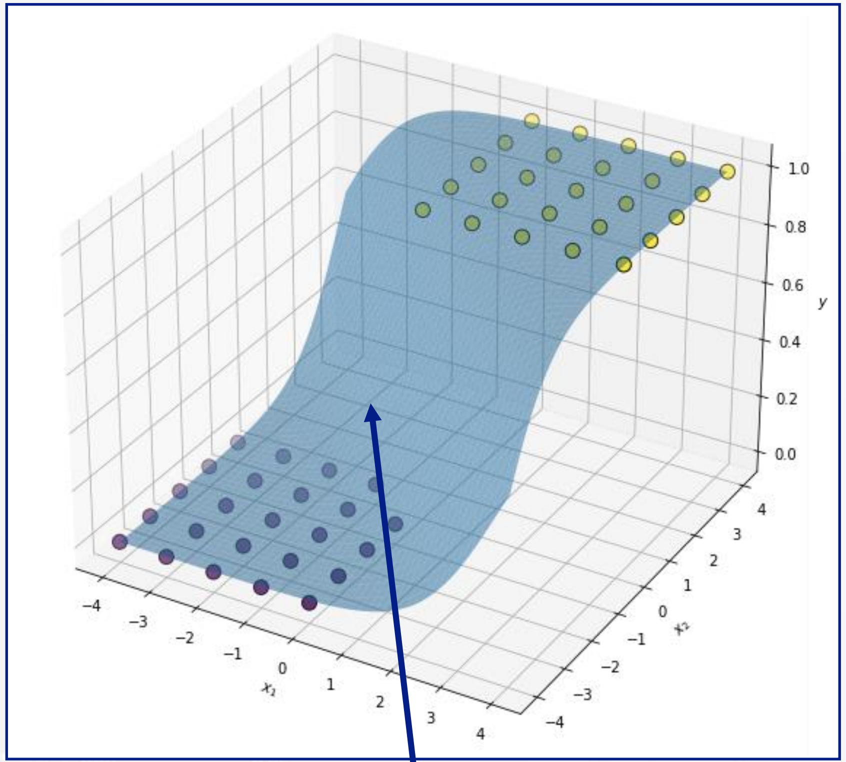


2D

Problem with Imbalanced Class

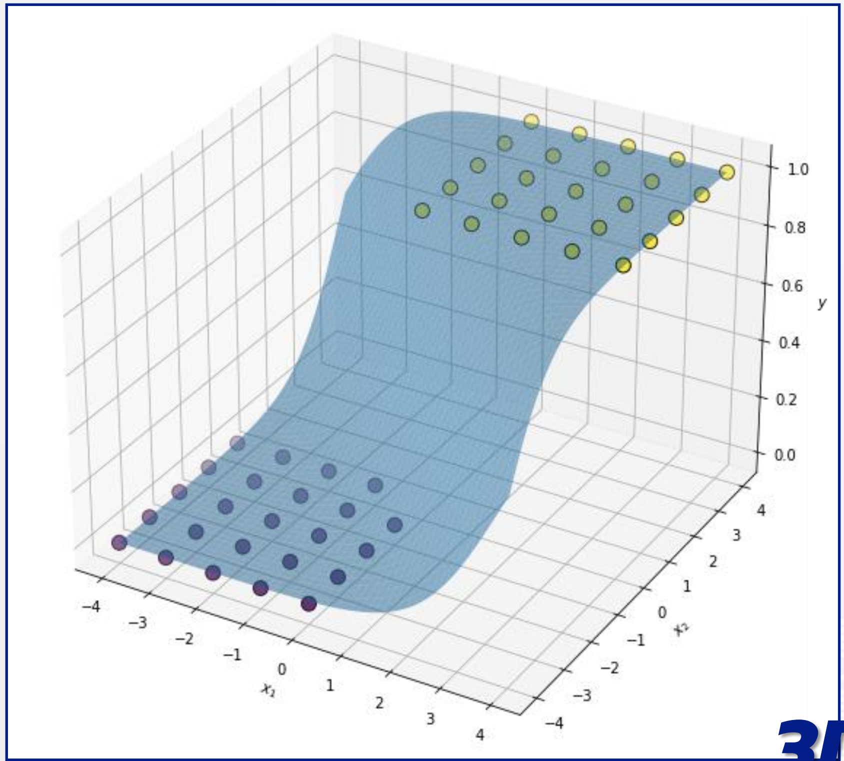


$z = w_0 + w_1 x_1 + w_2 x_2$

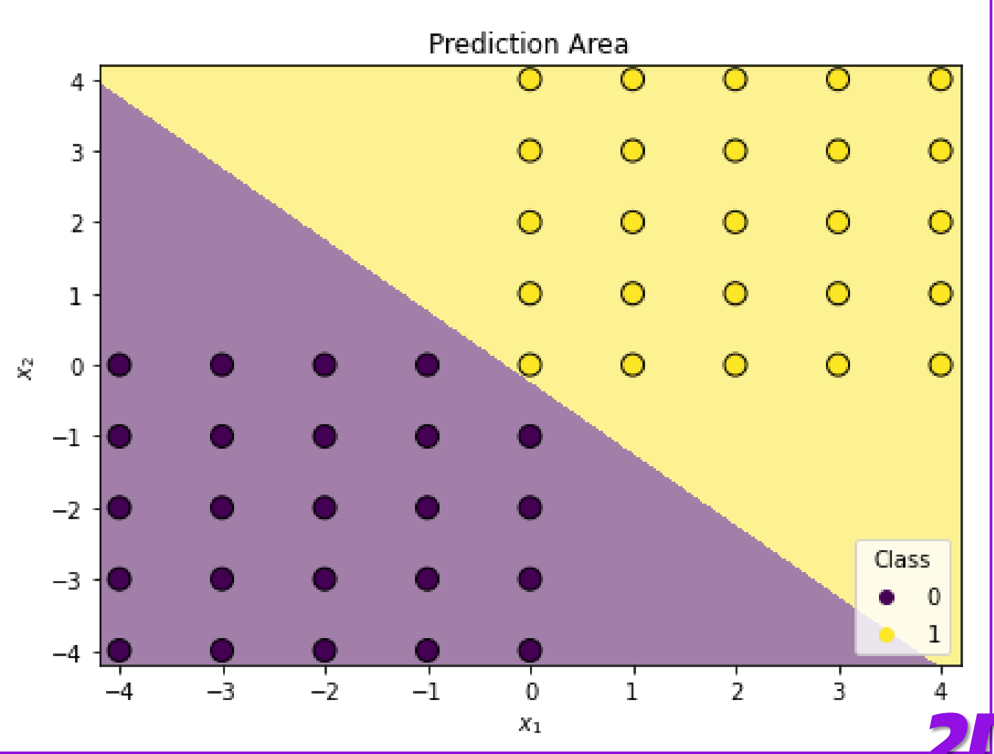


$\hat{y} = \sigma(z)$

Problem with Imbalanced Class



3D



2D

Imbalanced Class

Problem with Imbalanced Class

Solution

Code

Solution

Bootstrapping

SMOTE

ADASYN

Balanced
Class Weight

Solution

Bootstrapping

SMOTE

ADASYN

Balanced
Class Weight

Solution

หลักการทำงานของวิธี balanced class weight ก็คือ
เราจะให้ความสำคัญกับแต่ละ class อย่างเท่าเทียมกัน

Solution

x_1	x_2	y	
0	0	1	2%
0	1	1	2%
0	2	1	2%
0	3	1	2%
0	4	1	2%
1	0	1	2%
1	1	1	2%
1	2	1	2%
1	3	1	2%
1	4	1	2%
2	0	1	2%
2	1	1	2%
2	2	1	2%

x_1	x_2	y	
2	3	1	2%
2	4	1	2%
3	0	1	2%
3	1	1	2%
3	2	1	2%
3	3	1	2%
3	4	1	2%
4	0	1	2%
4	1	1	2%
4	2	1	2%
4	3	1	2%
4	4	1	2%
−4	−4	0	50%

Solution

$$W = W - \alpha \nabla Cost \begin{bmatrix} 0.02 \\ 0.02 \\ \vdots \\ 0.02 \\ 0.5 \end{bmatrix}$$

Imbalanced Class

- Problem with Imbalanced Class ✓
- Solution ✓
- Code

Code

SKlearn

Keras + Tensorflow

Pytorch

Code

- Binary Classification

```
model = Sequential()

model.add(Dense(5, input_dim=X.shape[1], activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss=BinaryCrossentropy(), optimizer=SGD(learning_rate=1))
```

Code

- Binary Classification

```python
class_weight = compute_class_weight('balanced', np.unique(y), y)
class_weight = dict(enumerate(class_weight.flatten()))

tqdm_callback = tfa.callbacks.TQDMProgressBar(show_epoch_progress=False)

with tf.device(device_name):
    history = model.fit(X, y_le, epochs=500, class_weight=class_weight, callbacks=[tqdm_callback])
```

Code

- Binary Classification

Code for this section

Open File

Imbalanced Class/keras/ Imbalanced Binary Classification (keras).ipynb

Code

- Multi-Class Classification

```
model = Sequential()

model.add(Dense(5, input_dim=X.shape[1], activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(y_enc.shape[1], activation='softmax'))

model.compile(loss=CategoricalCrossentropy(), optimizer=SGD(learning_rate=1))
```

Code

- Multi-Class Classification

```
class_weight = compute_class_weight('balanced', np.unique(y), y)
class_weight = dict(enumerate(class_weight.flatten()))

tqdm_callback = tfa.callbacks.TQDMProgressBar(show_epoch_progress=False)

with tf.device(device_name):
    history = model.fit(X, y_enc, epochs=500, class_weight=class_weight, callbacks=[tqdm_callback])
```

Code

- Multi-Class Classification

Code for this section

Open File

Imbalanced Class/keras/ Imbalanced Multi-Class Classification (keras).ipynb

Code

- Binary Classification

```
X = torch.from_numpy(X).float().to(device)
y_le = torch.from_numpy(y_le).float().to(device)
```

Code

- Binary Classification

```
layers = []

layers.append(nn.Linear(X.shape[1], 5))
layers.append(nn.ReLU())

layers.append(nn.Linear(5, 4))
layers.append(nn.ReLU())

layers.append(nn.Linear(4, 1))

model = nn.Sequential(*layers).to(device)
```

Code

- Binary Classification

```
optimizer = torch.optim.SGD(model.parameters(), lr=1)

pos_weight = (len(y_le) - y_le.sum())/y_le.sum()
loss_function = nn.BCEWithLogitsLoss(pos_weight=pos_weight)
```

Code

- Binary Classification

Code for this section

Open File

Imbalanced Class/pytorch/Imbalanced Binary Classification (pytorch).ipynb

Code

- Multi-Class Classification

```
X = torch.from_numpy(X).float().to(device)
y_le = torch.from_numpy(y_le).long().to(device)
```

Code

- Multi-Class Classification

```
layers = []

layers.append(nn.Linear(X.shape[1], 5))
layers.append(nn.ReLU())

layers.append(nn.Linear(5, 4))
layers.append(nn.ReLU())

layers.append(nn.Linear(4, 3))

model = nn.Sequential(*layers).to(device)
```

Code

- Multi-Class Classification

```
optimizer = torch.optim.SGD(model.parameters(), lr=1)

class_weight = compute_class_weight('balanced', np.unique(y), y)
class_weight = torch.from_numpy(class_weight).float()
loss_function = nn.CrossEntropyLoss(weight=class_weight)
```

Code

- Multi-Class Classification

Code for this section

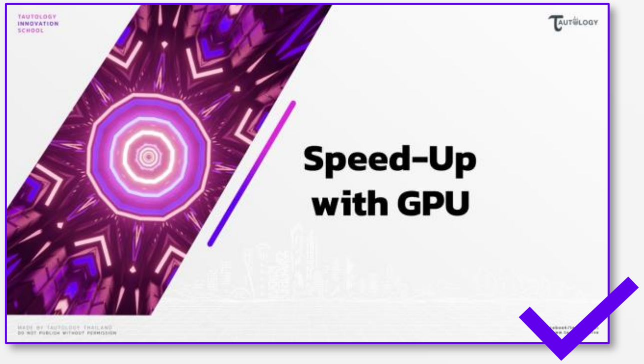
Open File

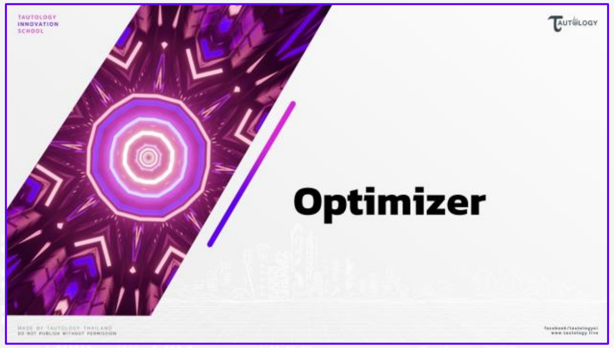
Imbalanced Class/pytorch/Imbalanced Multi-Class Classification (pytorch).ipynb

Imbalanced Class

- Problem with Imbalanced Class
- Solution
- Code

Improvement of Deep Learning





Regularization

Regularization

อ้างอิงจากงานวิจัยของ linear regression ในปี 1970 นักคณิตศาสตร์ค้นพบว่า

“ การใส่ $\lambda > 0$ จะทำให้ error ของ model ลดลงกว่าการไม่ใส่ λ $(\lambda = 0)$ ”

**ยกเว้น $y = \hat{y}$ (model ไม่มี error เมื่อ $\lambda = 0$)

Regularization

และเนื่องจาก deep learning ถูกสร้างขึ้นจาก linear regression

Regularization

สำหรับ deep learning **“ การใส่ $\lambda > 0$ จะทำให้ error ของ model ลดลงกว่าการไม่ใส่ λ $(\lambda = 0)$ ”**

Regularization

L2 Regularization

L1 Regularization

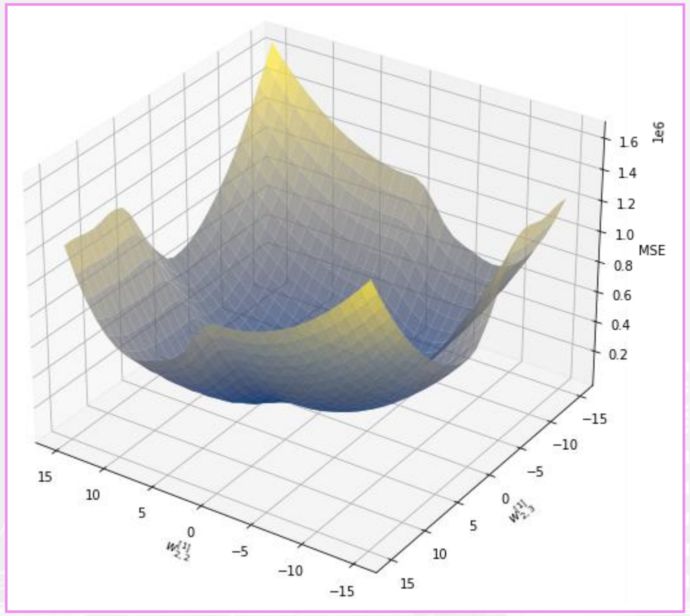
Elastic Net

Dropout Regularization

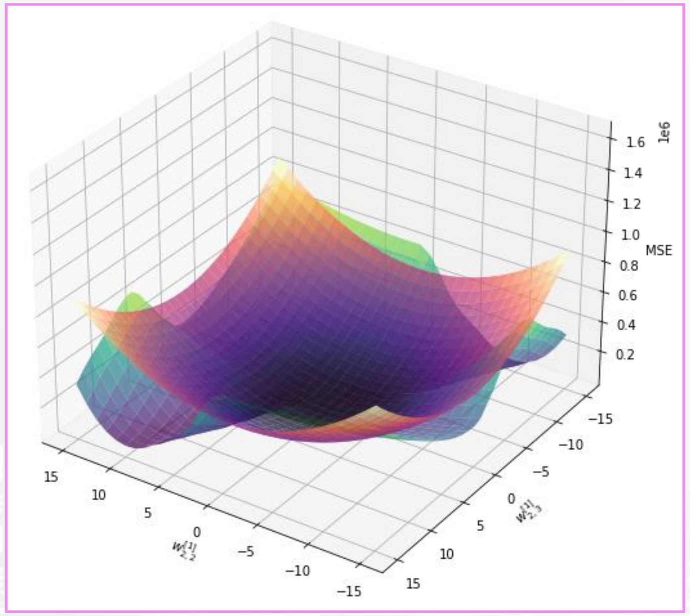
L2 Regularization

- Linearly Independent

$$Cost = Cost_0 + \lambda(\sum b^2 + \sum w^2)$$



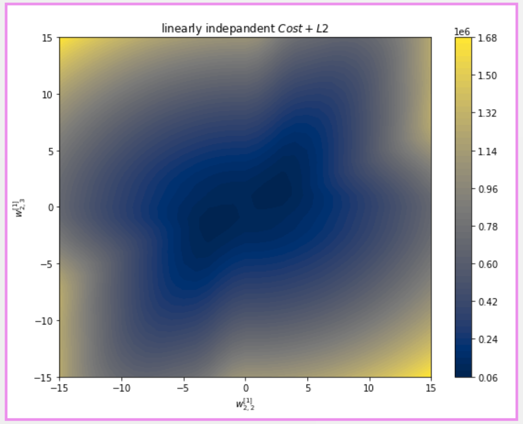
=



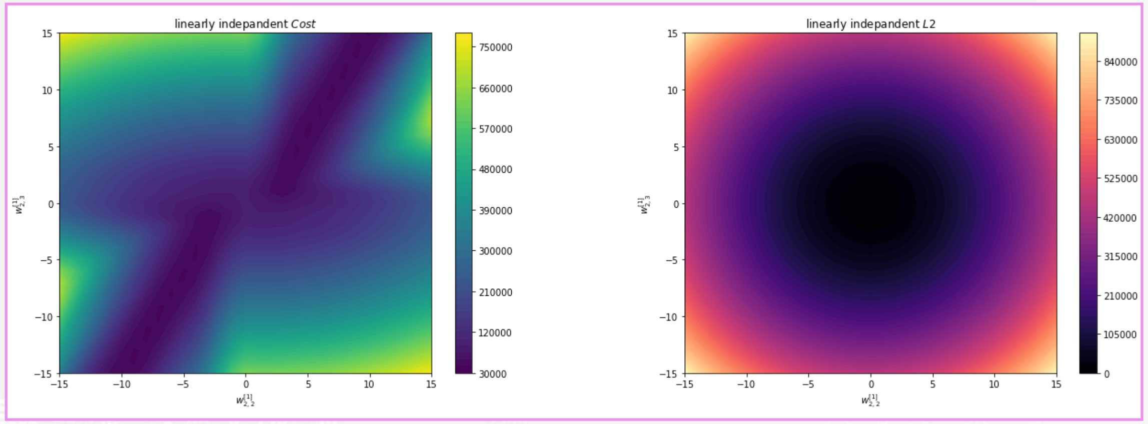
L2 Regularization

- Linearly Independent

$$Cost = Cost_0 + \lambda(\sum b^2 + \sum w^2)$$



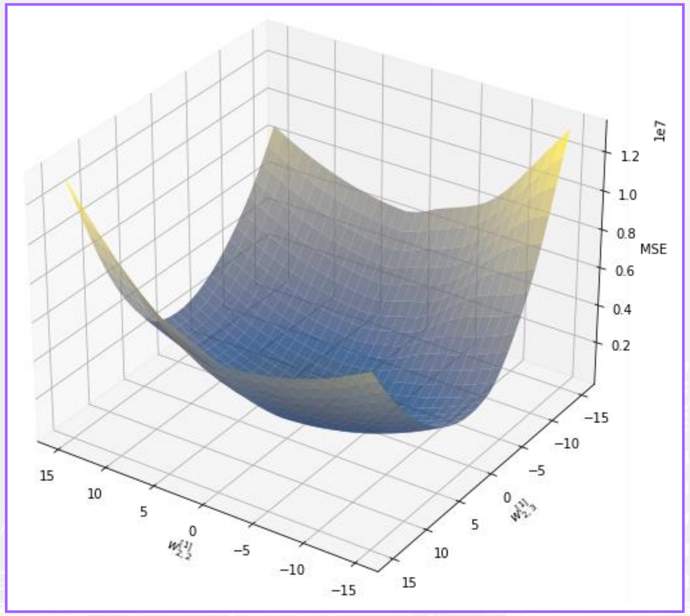
=



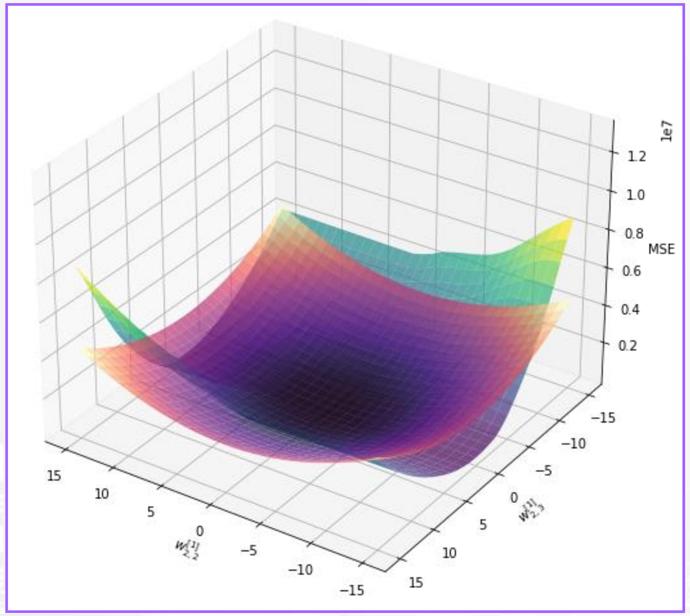
L2 Regularization

- Linearly Dependent

$$Cost = Cost_0 + \lambda(\sum b^2 + \sum w^2)$$



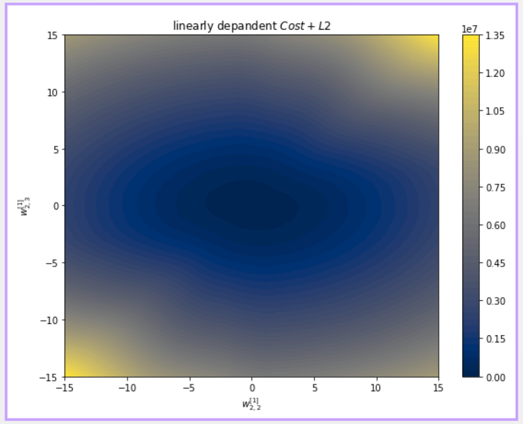
=



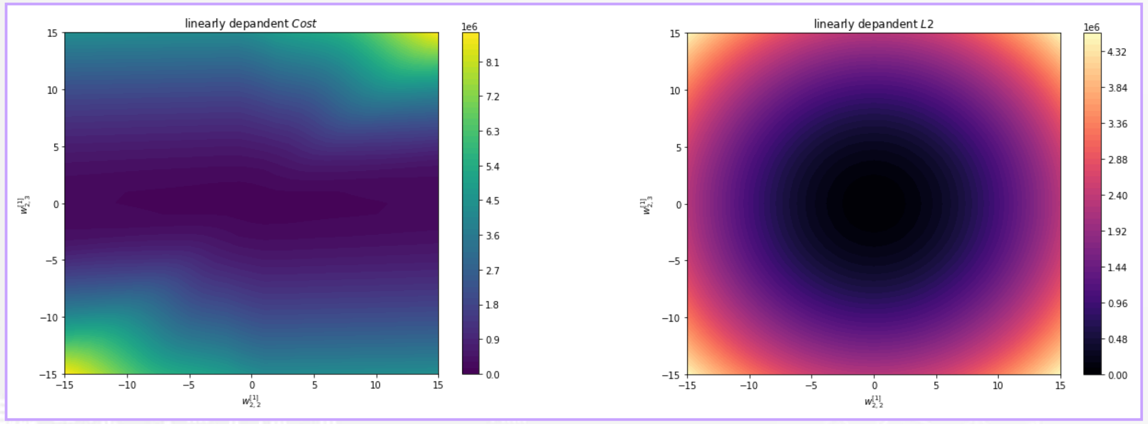
L2 Regularization

- Linearly Dependent

$$Cost = Cost_0 + \lambda(\sum b^2 + \sum w^2)$$



=



L2 Regularization

SKlearn

Keras + Tensorflow

Pytorch

L2 Regularization

- L2 Regularization/sklearn
  - L2 Regression (sklearn).ipynb
  - L2 Binary Classification (sklearn).ipynb
  - L2 Multi-Class Classification (sklearn).ipynb

L2 Regularization

- L2 Regularization/keras
  - L2 Regression (keras).ipynb
  - L2 Binary Classification (keras).ipynb
  - L2 Multi-Class Classification (keras).ipynb

L2 Regularization

- L2 Regularization/pytorch
  - L2 Regression (pytorch).ipynb
  - L2 Binary Classification (pytorch).ipynb
  - L2 Multi-Class Classification (pytorch).ipynb

Regularization

L2 Regularization

L1 Regularization

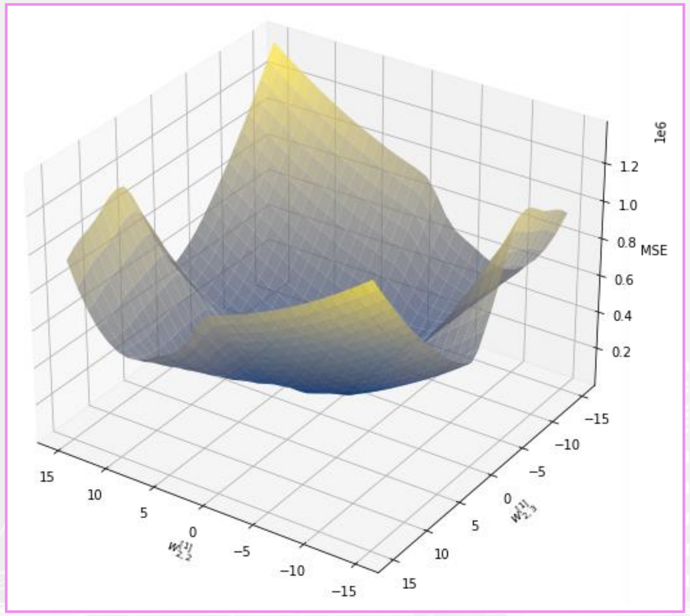
Elastic Net

Dropout Regularization

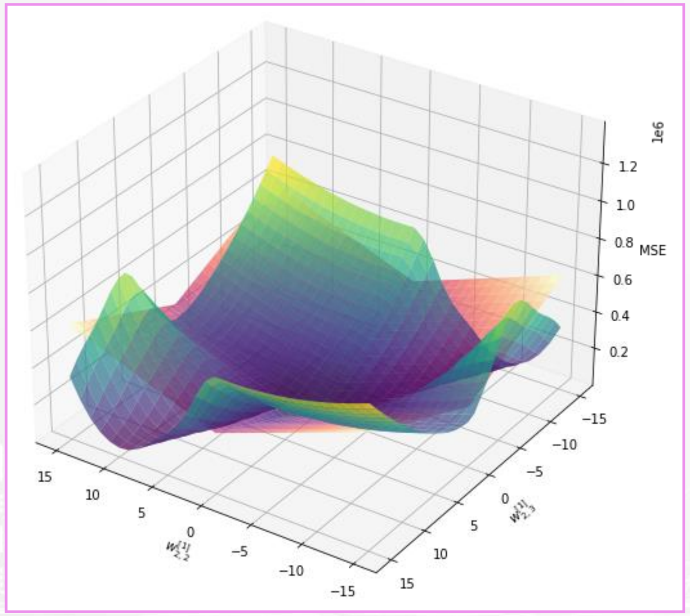
L1 Regularization

- Linearly Independent

$$Cost = Cost_0 + \lambda(\sum|b| + \sum|w|)$$



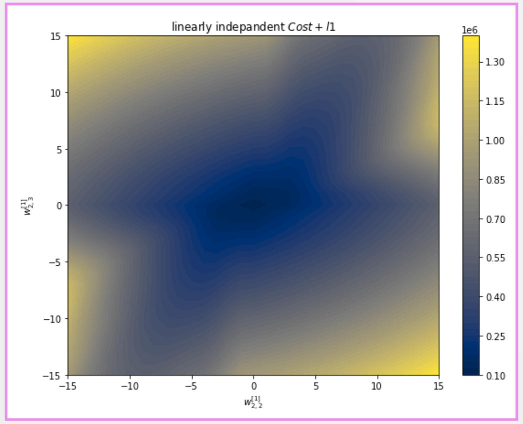
=

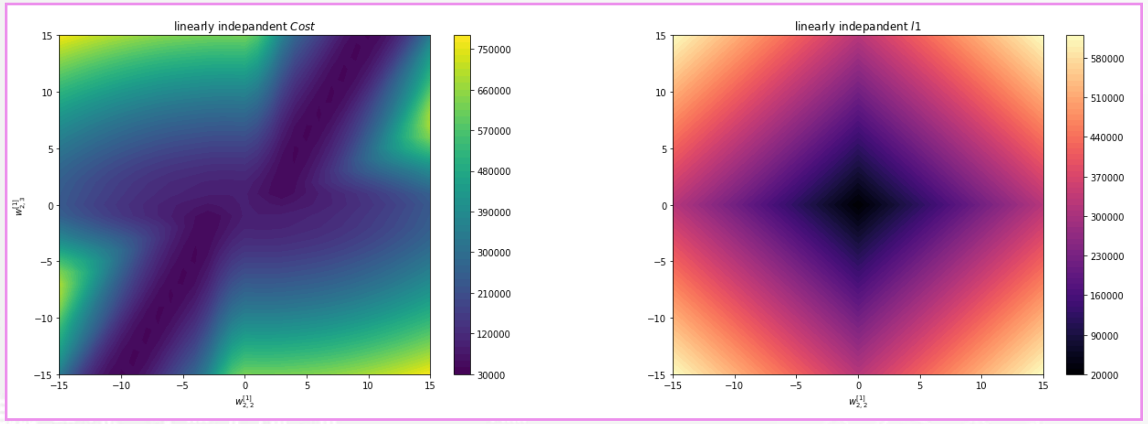


L1 Regularization

- Linearly Independent

$$Cost = Cost_0 + \lambda\left(\sum|b| + \sum|w|\right)$$

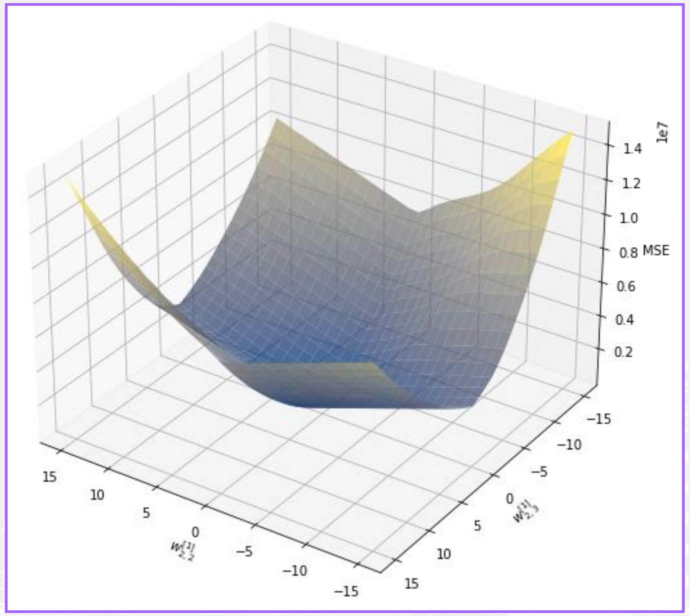




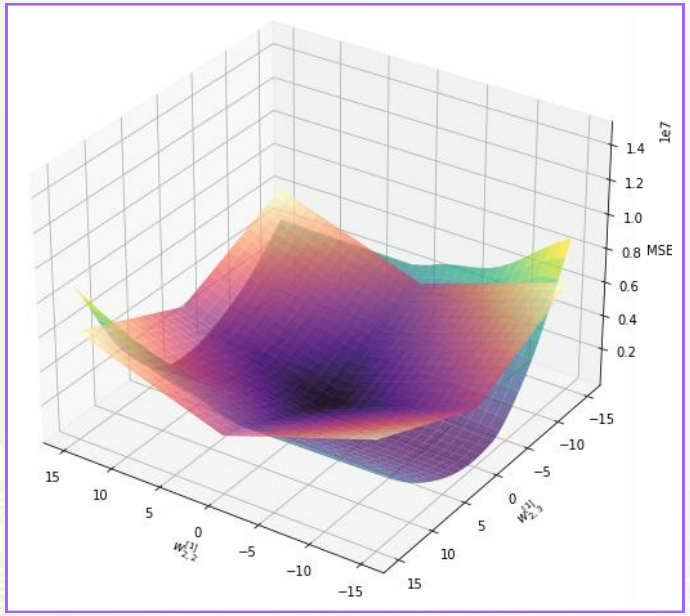
L1 Regularization

- Linearly Dependent

$$Cost = Cost_0 + \lambda(\sum|b| + \sum|w|)$$



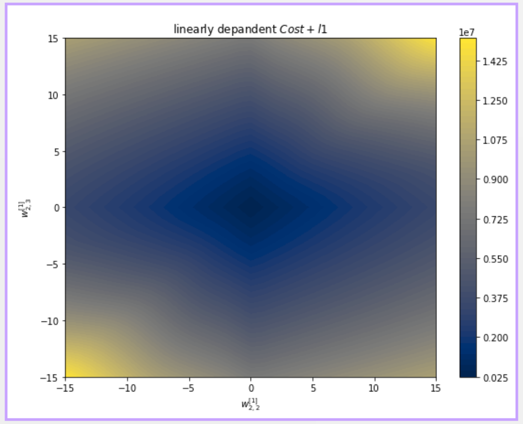
=



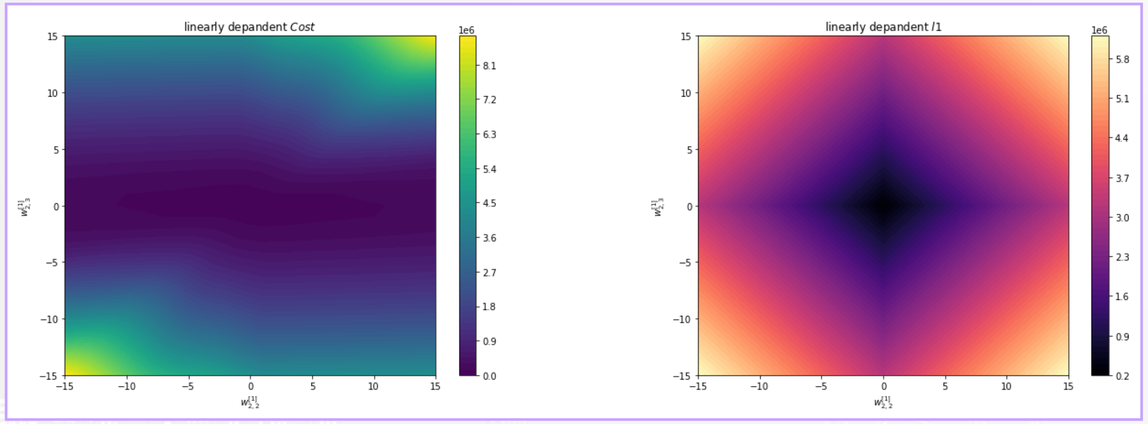
L1 Regularization

- Linearly Dependent

$$Cost = Cost_0 + \lambda(\sum|b| + \sum|w|)$$



=



L1 Regularization

SKlearn

Keras + Tensorflow

Pytorch

L1 Regularization

- L1 Regularization/keras
  - L1 Regression (keras).ipynb
  - L1 Binary Classification (keras).ipynb
  - L1 Multi-Class Classification (keras).ipynb

L1 Regularization

- L1 Regularization/pytorch
  - L1 Regression (pytorch).ipynb
  - L1 Binary Classification (pytorch).ipynb
  - L1 Multi-Class Classification (pytorch).ipynb

Regularization

L2 Regularization

L1 Regularization

Elastic Net

Dropout Regularization

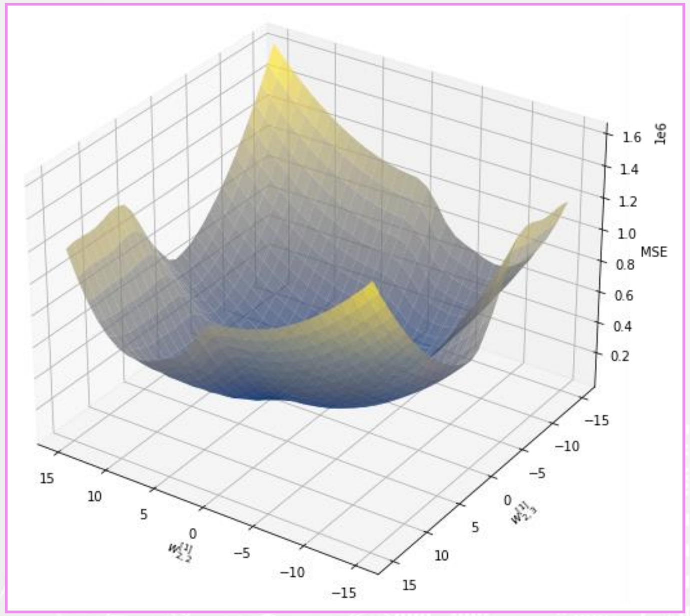
Elastic Net

Elastic net คือ การทำ regularization โดยมี cost function เป็น

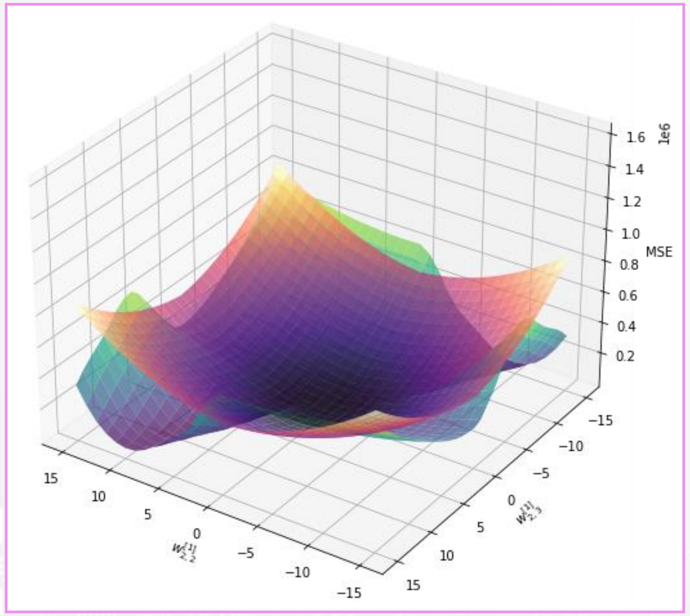
$$Cost = Cost_0 + \lambda l1_{ratio}(\sum|b| + \sum|w|)$$
$$+\lambda(1 - l1_{ratio})(\sum b^2 + \sum w^2)$$

Elastic Net

- Linearly Independent

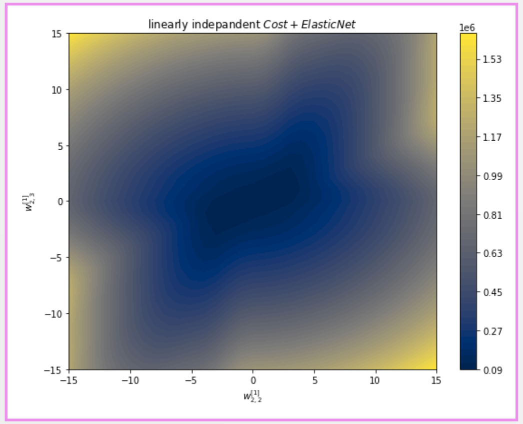


=

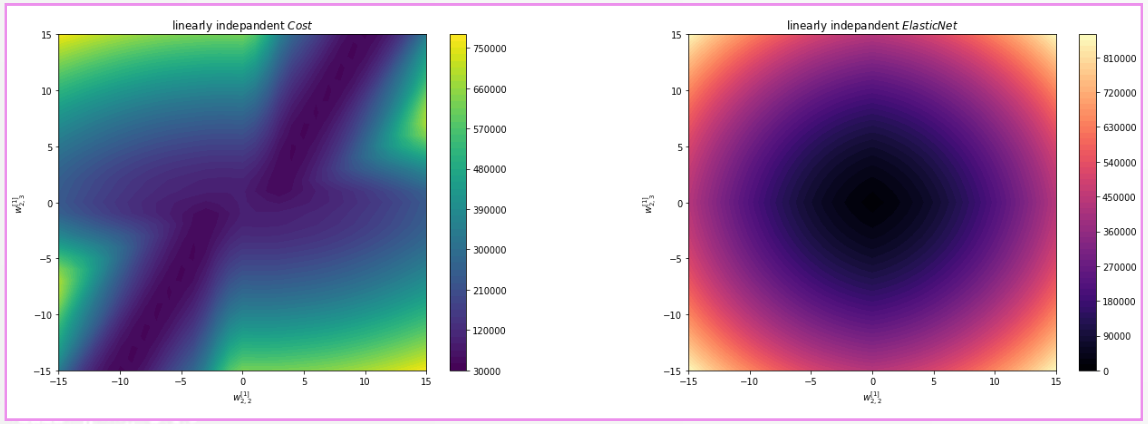


Elastic Net

- Linearly Independent

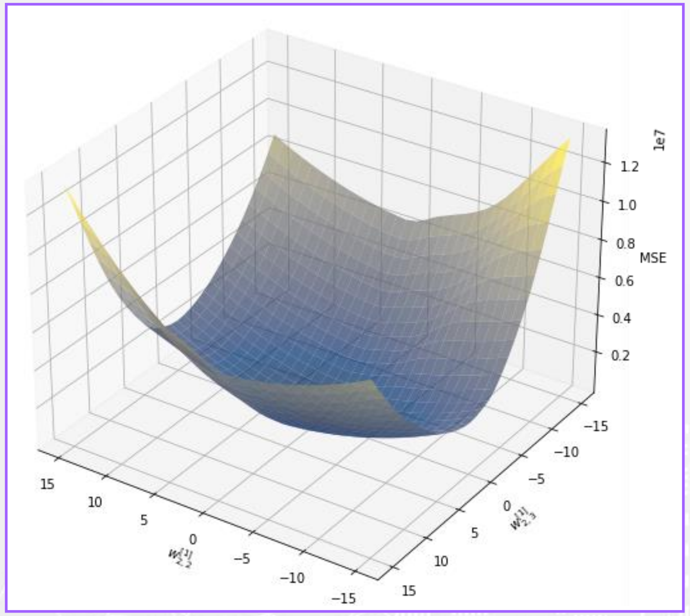


=

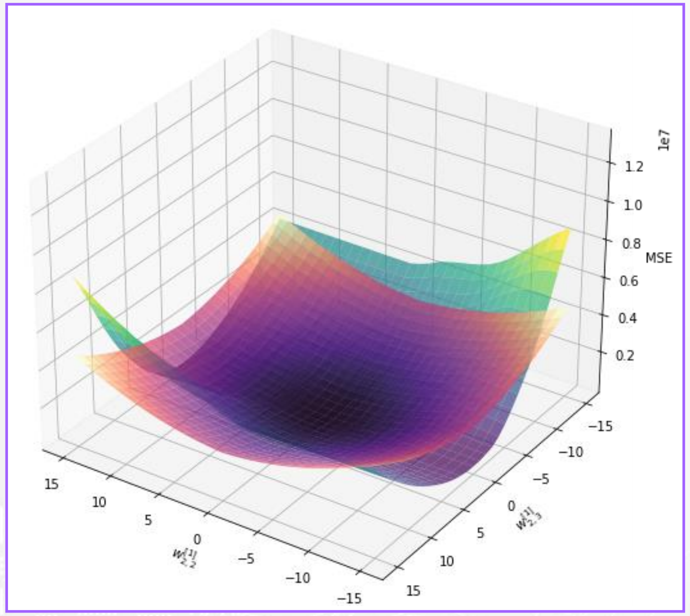


Elastic Net

- Linearly Dependent

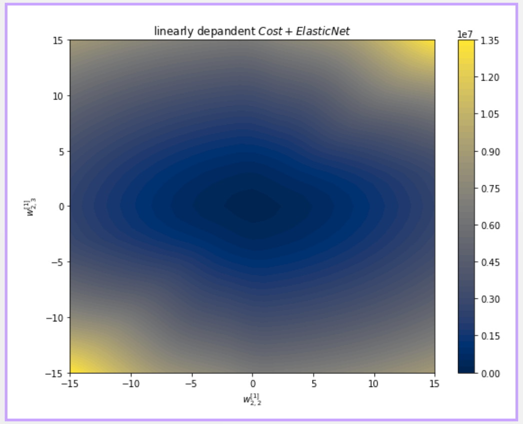


=

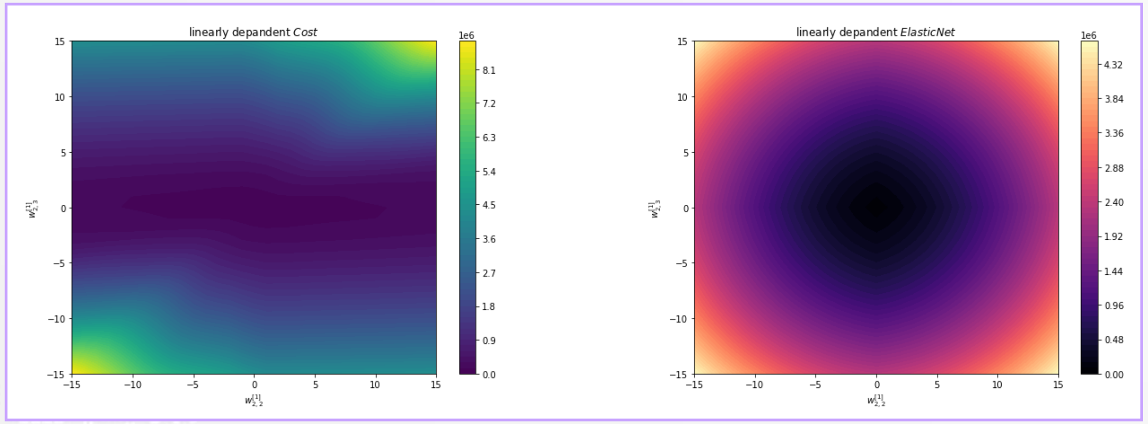


Elastic Net

- Linearly Dependent



=



Elastic Net

scikit learn

SKlearn

Keras + Tensorflow

Pytorch

Elastic Net

Elastic Net/keras

- ElasticNet Regression (keras).ipynb
- ElasticNet Binary Classification (keras).ipynb
- ElasticNet Multi-Class Classification (keras).ipynb

Elastic Net

Elastic Net/pytorch

- ElasticNet Regression (pytorch).ipynb
- ElasticNet Binary Classification (pytorch).ipynb
- ElasticNet Multi-Class Classification (pytorch).ipynb

Regularization

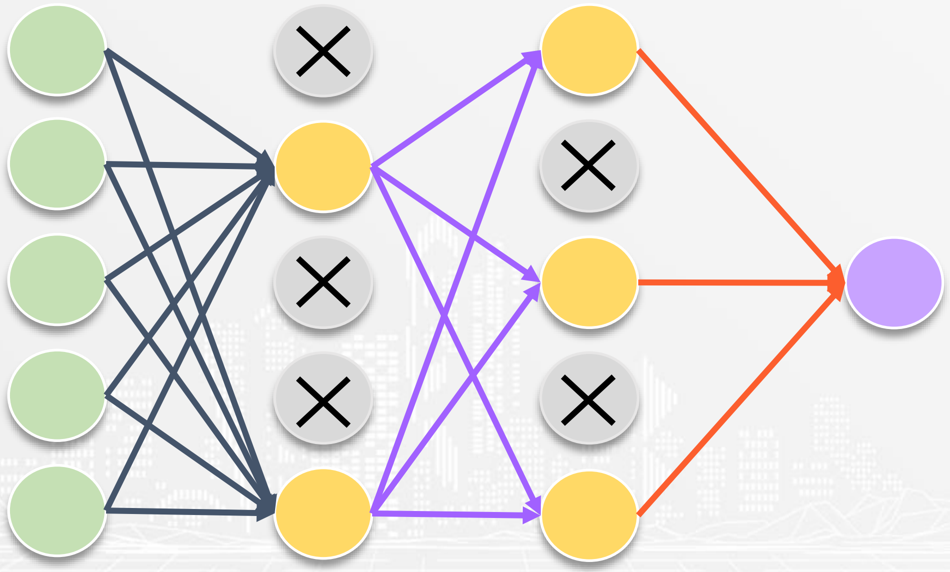
- L2 Regularization ✓
- L1 Regularization ✓
- Elastic Net ✓
- Dropout Regularization

Dropout Regularization

- ☐ What is Dropout Regularization?
- ☐ Effect of Dropout
- ☐ Technical of Dropout
- ☐ Code of Dropout

What is Dropout Regularization?

Dropout regularization คือ หนึ่งในวิธีการ regularization โดยแนวคิดของวิธีการนี้ คือ ในแต่ละ epoch ให้ทำการ drop node แบบสุ่มใน network ไม่ให้ส่งข้อมูลไปยัง node อื่น ในชั้นถัดไปในระหว่างการ train model (เปิดทุก node เมื่อนำ model ไปใช้งานได้จริง)

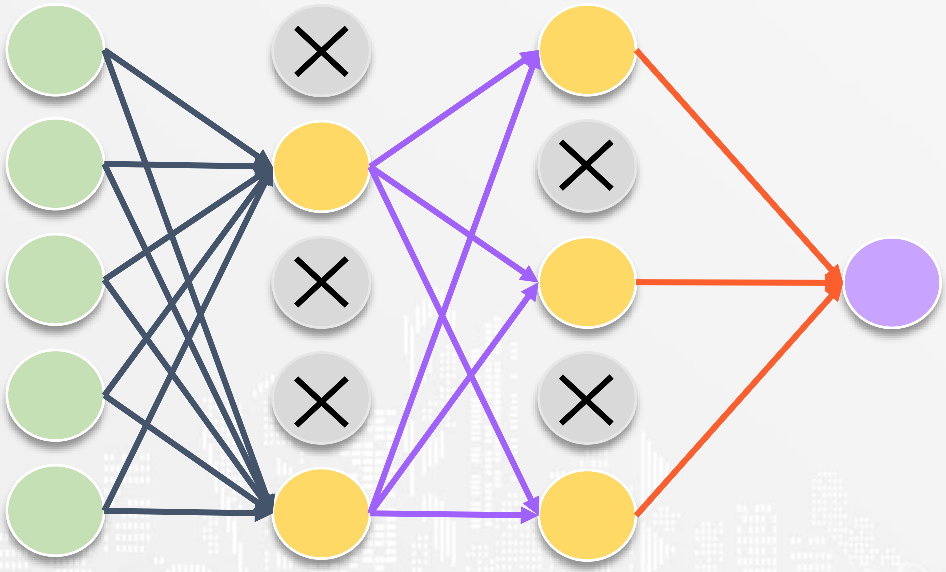


Dropout Regularization

- ☑ **What is Dropout Regularization?**
- ☐ Effect of Dropout
- ☐ Technical of Dropout
- ☐ Code of Dropout

Effect of Dropout

Effect of dropout คือ เมื่อบาง node ถูก drop ในระหว่างการ train model ทำให้ model ถูกบังคับให้ใช้เส้นทางอื่นที่ไม่ผ่าน node ที่ถูก drop ในการพยากรณ์ output



Effect of Dropout

เมื่อนำ model ไปใช้งานจริง model จะมีหลายเส้นทางที่สามารถใช้ในการพยากรณ์ output ได้

การใช้เส้นทางเหล่านั้นร่วมกันในการพยากรณ์ส่งผลให้ model มีความ general มากขึ้น

Dropout Regularization

- ☑ **What is Dropout Regularization?**
- ☑ **Effect of Dropout**
- ☐ Technical of Dropout
- ☐ Code of Dropout

Technical of Dropout

ขั้นตอนในการสร้าง dropout regularization

1. กำหนดความน่าจะเป็นในการ drop node ในแต่ละ hidden layer (`drop_rate`)
2. ในแต่ละ epoch ให้ทำการ

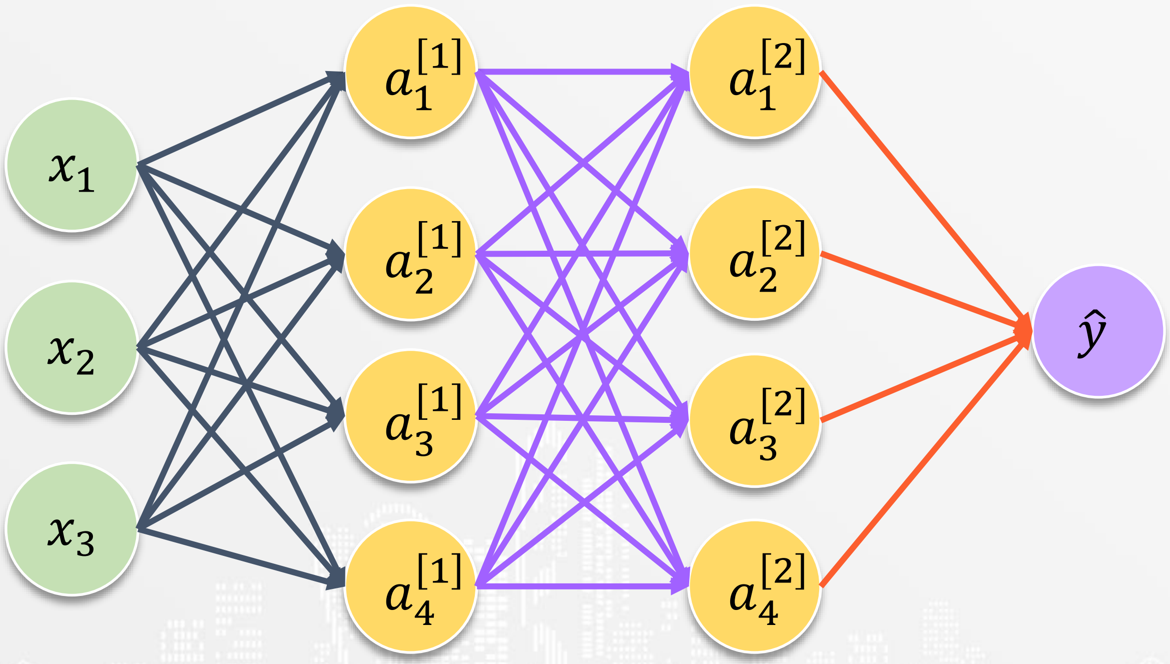
   2.1 Drop node แบบสุ่มในแต่ละ hidden layer ตาม `drop_rate` ที่กำหนด

   2.2 Feedforward

   2.3 Update parameters

Technical of Dropout

ตัวอย่าง



Technical of Dropout

ตัวอย่าง

1. กำหนด `drop_rate` สำหรับแต่ละ hidden layer

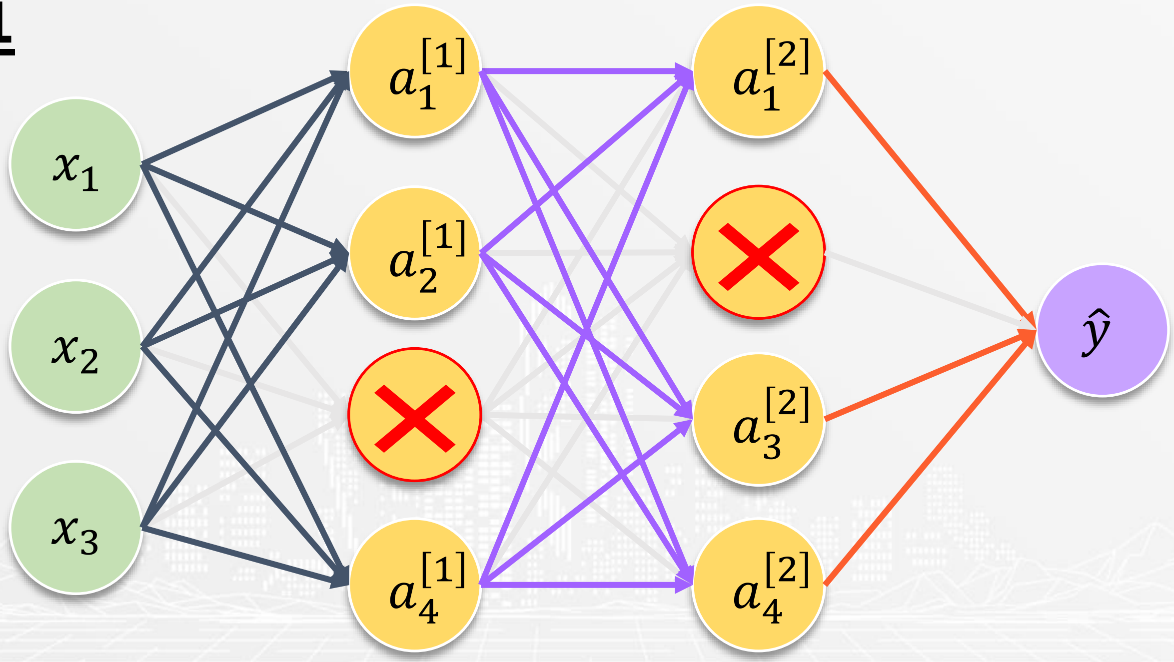
กำหนดให้ `drop_rate = 0.25`

Technical of Dropout

ตัวอย่าง

2.1 Drop node แบบสุ่มในแต่ละ hidden layer

epoch = 1



Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 1

$$a_1^{[1]} = ReLU\left(b_1^{[1]} + w_{1,1}^{[1]}x_1 + w_{2,1}^{[1]}x_2 + w_{3,1}^{[1]}x_3\right)$$

$$a_2^{[1]} = ReLU\left(b_2^{[1]} + w_{1,2}^{[1]}x_1 + w_{2,2}^{[1]}x_2 + w_{3,2}^{[1]}x_3\right)$$

$$a_3^{[1]} = \times$$

$$a_4^{[1]} = ReLU\left(b_4^{[1]} + w_{1,4}^{[1]}x_1 + w_{2,4}^{[1]}x_2 + w_{3,4}^{[1]}x_3\right)$$

Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 1

$$a_1^{[2]} = ReLU\left(b_1^{[2]} + w_{1,1}^{[2]}a_1^{[1]} + w_{2,1}^{[2]}a_2^{[1]} + w_{3,1}^{[2]}a_3^{[1]} + w_{4,1}^{[2]}a_4^{[1]}\right)$$

$$a_2^{[2]} = \times$$

$$a_3^{[2]} = ReLU\left(b_3^{[2]} + w_{1,3}^{[2]}a_1^{[1]} + w_{2,3}^{[2]}a_2^{[1]} + w_{3,3}^{[2]}a_3^{[1]} + w_{4,3}^{[2]}a_4^{[1]}\right)$$

$$a_4^{[2]} = ReLU\left(b_4^{[2]} + w_{1,4}^{[2]}a_1^{[1]} + w_{2,4}^{[2]}a_2^{[1]} + w_{3,4}^{[2]}a_3^{[1]} + w_{4,4}^{[2]}a_4^{[1]}\right)$$

Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 1

$$\hat{y} = b^{[out]} + w_1^{[out]} a_1^{[2]} + w_2^{[out]} a_2^{[2]} + w_3^{[out]} a_3^{[2]} + w_4^{[out]} a_4^{[2]}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 1

$$\begin{bmatrix} b_1^{[1]} & b_2^{[1]} & \color{red}{b_3^{[1]}} & b_4^{[1]} \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} & \color{red}{w_{1,3}^{[1]}} & w_{1,4}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & \color{red}{w_{2,3}^{[1]}} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & \color{red}{w_{3,3}^{[1]}} & w_{3,4}^{[1]} \end{bmatrix}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 1

$$\begin{bmatrix} b_1^{[2]} & b_2^{[2]} & b_3^{[2]} & b_4^{[2]} \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} & w_{1,4}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} & w_{2,4}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} & w_{3,4}^{[2]} \\ w_{4,1}^{[2]} & w_{4,2}^{[2]} & w_{4,3}^{[2]} & w_{4,4}^{[2]} \end{bmatrix}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 1

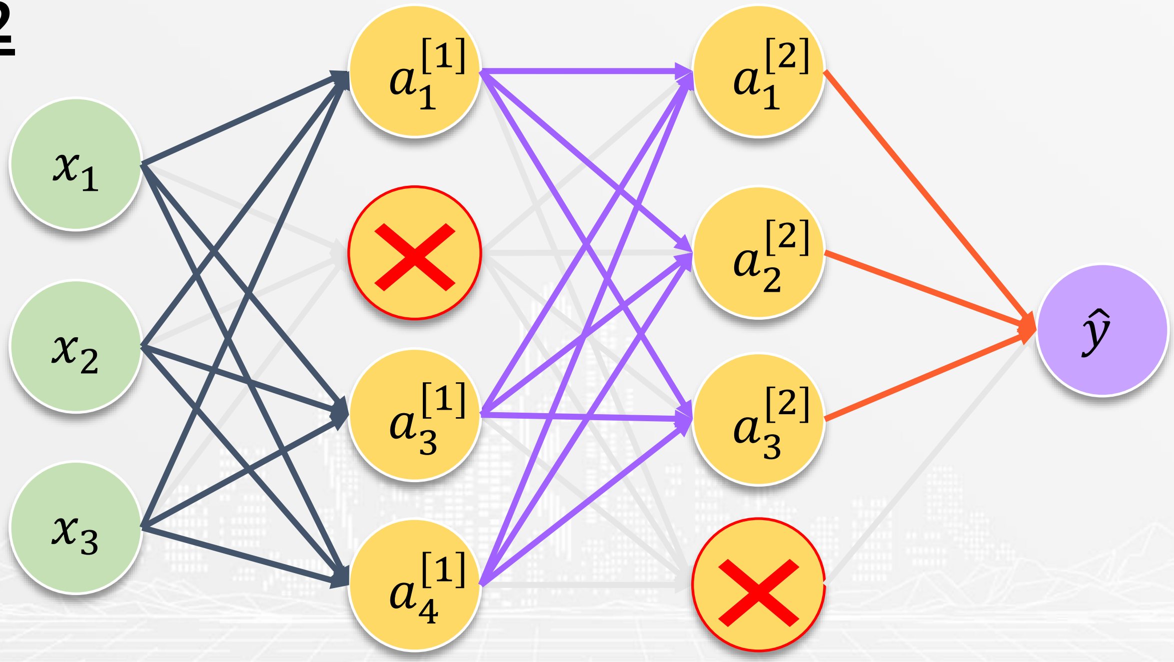
$$\left[b^{[out]}\right] \qquad \begin{bmatrix} w_1^{[out]} \\ w_2^{[out]} \\ w_3^{[out]} \\ w_4^{[out]} \end{bmatrix}$$

Technical of Dropout

ตัวอย่าง

2.1 Drop node แบบสุ่มในแต่ละ hidden layer

epoch = 2



Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 2

$$a_1^{[1]} = ReLU\left(b_1^{[1]} + w_{1,1}^{[1]}x_1 + w_{2,1}^{[1]}x_2 + w_{3,1}^{[1]}x_3\right)$$

$$a_2^{[1]} = \times$$

$$a_3^{[1]} = ReLU\left(b_3^{[1]} + w_{1,3}^{[1]}x_1 + w_{2,3}^{[1]}x_2 + w_{3,3}^{[1]}x_3\right)$$

$$a_4^{[1]} = ReLU\left(b_4^{[1]} + w_{1,4}^{[1]}x_1 + w_{2,4}^{[1]}x_2 + w_{3,4}^{[1]}x_3\right)$$

Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 2

$$a_1^{[2]} = ReLU\left(b_1^{[2]} + w_{1,1}^{[2]} a_1^{[1]} + w_{2,1}^{[2]} a_2^{[1]} + w_{3,1}^{[2]} a_3^{[1]} + w_{4,1}^{[2]} a_4^{[1]}\right)$$

$$a_2^{[2]} = ReLU\left(b_1^{[2]} + w_{1,2}^{[2]} a_1^{[1]} + w_{2,2}^{[2]} a_2^{[1]} + w_{3,2}^{[2]} a_3^{[1]} + w_{4,2}^{[2]} a_4^{[1]}\right)$$

$$a_3^{[2]} = ReLU\left(b_3^{[2]} + w_{1,3}^{[2]} a_1^{[1]} + w_{2,3}^{[2]} a_2^{[1]} + w_{3,3}^{[2]} a_3^{[1]} + w_{4,3}^{[2]} a_4^{[1]}\right)$$

$$a_4^{[2]} = \times$$

Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 2

$$\hat{y} = b^{[out]} + w_1^{[out]} a_1^{[2]} + w_2^{[out]} a_2^{[2]} + w_3^{[out]} a_3^{[2]} + w_4^{[out]} a_4^{[2]}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 2

$$\begin{bmatrix} b_1^{[1]} & b_2^{[1]} & b_3^{[1]} & b_4^{[1]} \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,3}^{[1]} & w_{1,4}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} & w_{3,4}^{[1]} \end{bmatrix}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 2

$$\begin{bmatrix} b_1^{[2]} & b_2^{[2]} & b_3^{[2]} & {\color{red}b_4^{[2]}} \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} & {\color{red}w_{1,4}^{[2]}} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} & {\color{red}w_{2,4}^{[2]}} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} & {\color{red}w_{3,4}^{[2]}} \\ w_{4,1}^{[2]} & w_{4,2}^{[2]} & w_{4,3}^{[2]} & {\color{red}w_{4,4}^{[2]}} \end{bmatrix}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 2

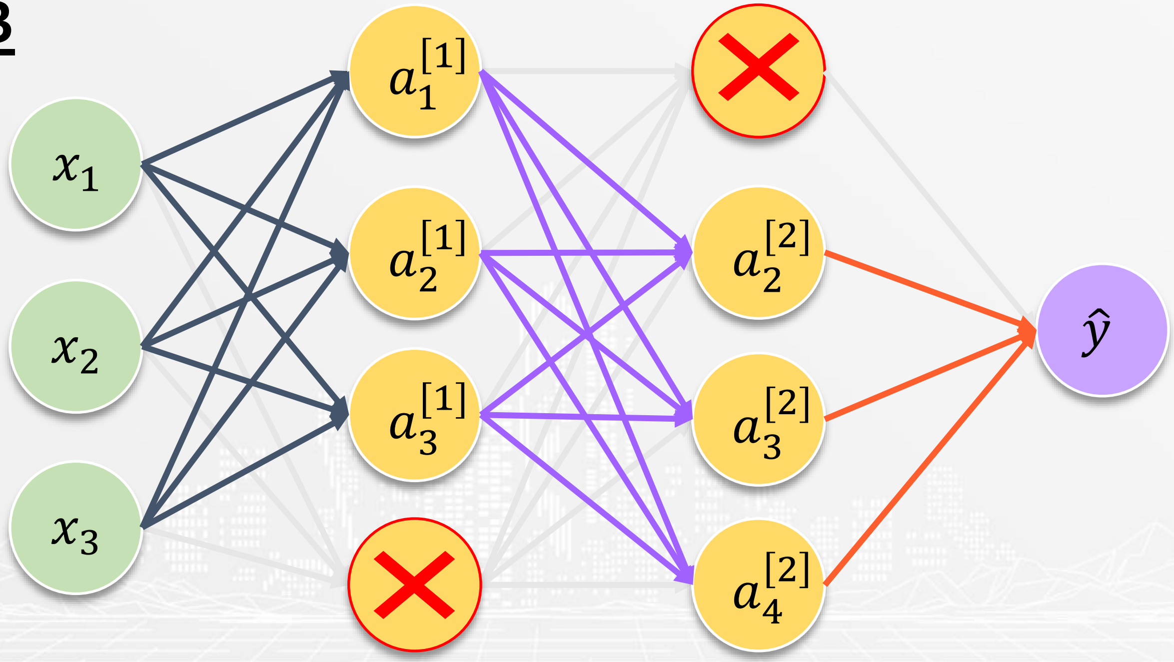
$$\left[b^{[out]}\right] \qquad \begin{bmatrix} w_1^{[out]} \\ w_2^{[out]} \\ w_3^{[out]} \\ \color{red}{w_4^{[out]}} \end{bmatrix}$$

Technical of Dropout

ตัวอย่าง

2.1 Drop node แบบสุ่มในแต่ละ hidden layer

<u>epoch = 3</u>



Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 3

$$a_1^{[1]} = ReLU\left(b_1^{[1]} + w_{1,1}^{[1]}x_1 + w_{2,1}^{[1]}x_2 + w_{3,1}^{[1]}x_3\right)$$

$$a_2^{[1]} = ReLU\left(b_2^{[1]} + w_{1,2}^{[1]}x_1 + w_{2,2}^{[1]}x_2 + w_{3,2}^{[1]}x_3\right)$$

$$a_3^{[1]} = ReLU\left(b_3^{[1]} + w_{1,3}^{[1]}x_1 + w_{2,3}^{[1]}x_2 + w_{3,3}^{[1]}x_3\right)$$

$$a_4^{[1]} = \times$$

Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 3

$$a_1^{[2]} = \times$$

$$a_2^{[2]} = ReLU\left(b_2^{[2]} + w_{1,2}^{[2]}a_1^{[1]} + w_{2,2}^{[2]}a_2^{[1]} + w_{3,2}^{[2]}a_3^{[1]} + w_{4,2}^{[2]}a_4^{[1]}\right)$$

$$a_3^{[2]} = ReLU\left(b_3^{[2]} + w_{1,3}^{[2]}a_1^{[1]} + w_{2,3}^{[2]}a_2^{[1]} + w_{3,3}^{[2]}a_3^{[1]} + w_{4,3}^{[2]}a_4^{[1]}\right)$$

$$a_4^{[2]} = ReLU\left(b_4^{[2]} + w_{1,4}^{[2]}a_1^{[1]} + w_{2,4}^{[2]}a_2^{[1]} + w_{3,4}^{[2]}a_3^{[1]} + w_{4,4}^{[2]}a_4^{[1]}\right)$$

Technical of Dropout

ตัวอย่าง

2.2 Feedforward

epoch = 3

$$\hat{y} = b^{[out]} + w_1 a_1^{[2]} + w_2^{[out]} a_2^{[2]} + w_3^{[out]} a_3^{[2]} + w_4^{[out]} a_4^{[2]}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 3

$$\begin{bmatrix} b_1^{[1]} & b_2^{[1]} & b_3^{[1]} & \color{red}{b_4^{[1]}} \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,3}^{[1]} & \color{red}{w_{1,4}^{[1]}} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} & \color{red}{w_{2,4}^{[1]}} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} & \color{red}{w_{3,4}^{[1]}} \end{bmatrix}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 3

$$\begin{bmatrix} b_1^{[2]} & b_2^{[2]} & b_3^{[2]} & b_4^{[2]} \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} & w_{1,4}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} & w_{2,4}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} & w_{3,4}^{[2]} \\ w_{4,1}^{[2]} & w_{4,2}^{[2]} & w_{4,3}^{[2]} & w_{4,4}^{[2]} \end{bmatrix}$$

Technical of Dropout

ตัวอย่าง

2.3 Update parameter

epoch = 3

$$\left[b^{[out]}\right] \qquad \begin{bmatrix} w_1^{[out]} \\ w_2^{[out]} \\ w_3^{[out]} \\ w_4^{[out]} \end{bmatrix}$$

Dropout Regularization

- ☑ **What is Dropout Regularization?**
- ☑ **Effect of Dropout**
- ☑ **Technical of Dropout**
- ☐ Code of Dropout

Code of Dropout

Keras + Tensorflow

Pytorch

Code of Dropout

- Dropout/keras
  - Dropout Regression (keras).ipynb
  - Dropout Binary Classification (keras).ipynb
  - Dropout Multi-Class Classification (keras).ipynb

Code of Dropout

- Dropout/pytorch
  - Dropout Regression (pytorch).ipynb
  - Dropout Binary Classification (pytorch).ipynb
  - Dropout Multi-Class Classification (pytorch).ipynb

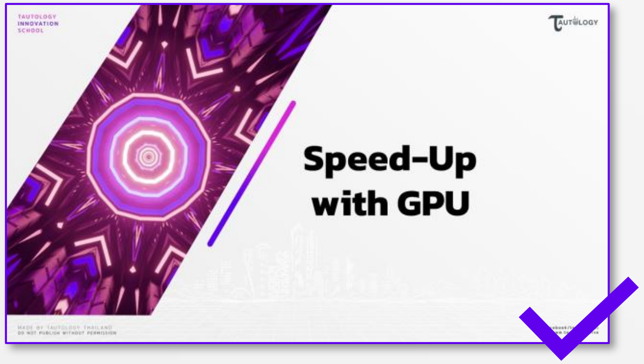
Dropout Regularization

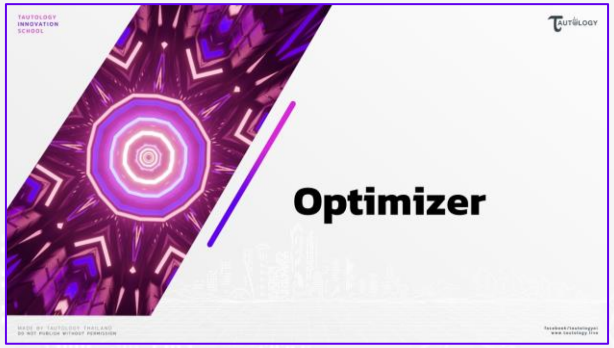
- ☑ What is Dropout Regularization?
- ☑ Effect of Dropout
- ☑ Technical of Dropout
- ☑ Code of Dropout

Regularization

- L2 Regularization
- L1 Regularization
- Elastic Net
- Dropout Regularization

Improvement of Deep Learning





Optimizer

Optimizer

Gradient Descent Variants

Optimization Algorithm

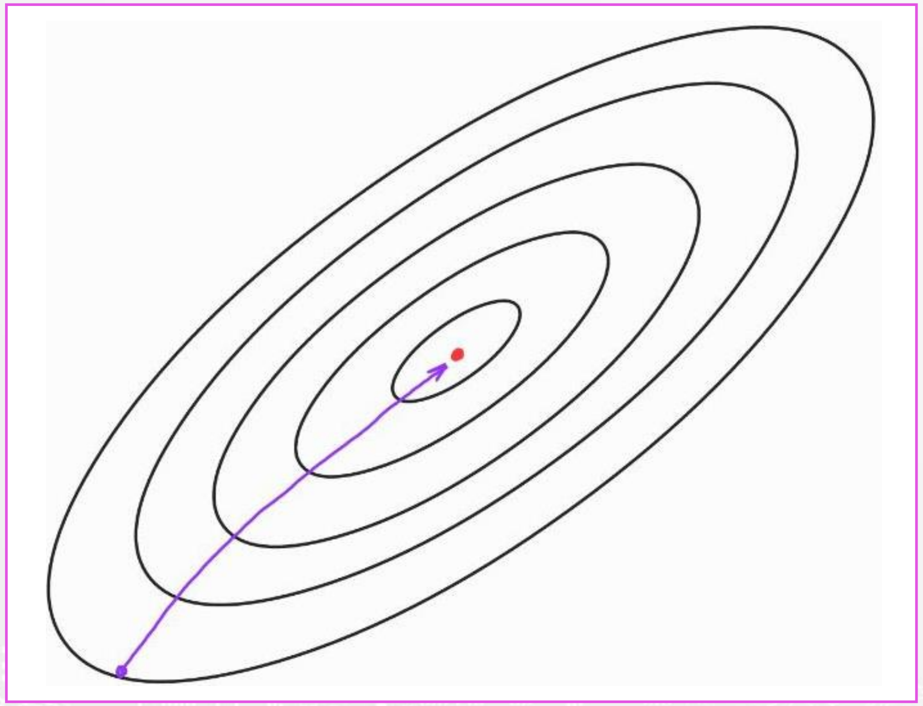
Code

Gradient Descent Variants

- ☐ Batch Gradient Descent
- ☐ Stochastic Gradient Descent
- ☐ Minibatch Gradient Descent

Batch Gradient Descent

- **Batch Gradient Descent**



Batch Gradient Descent

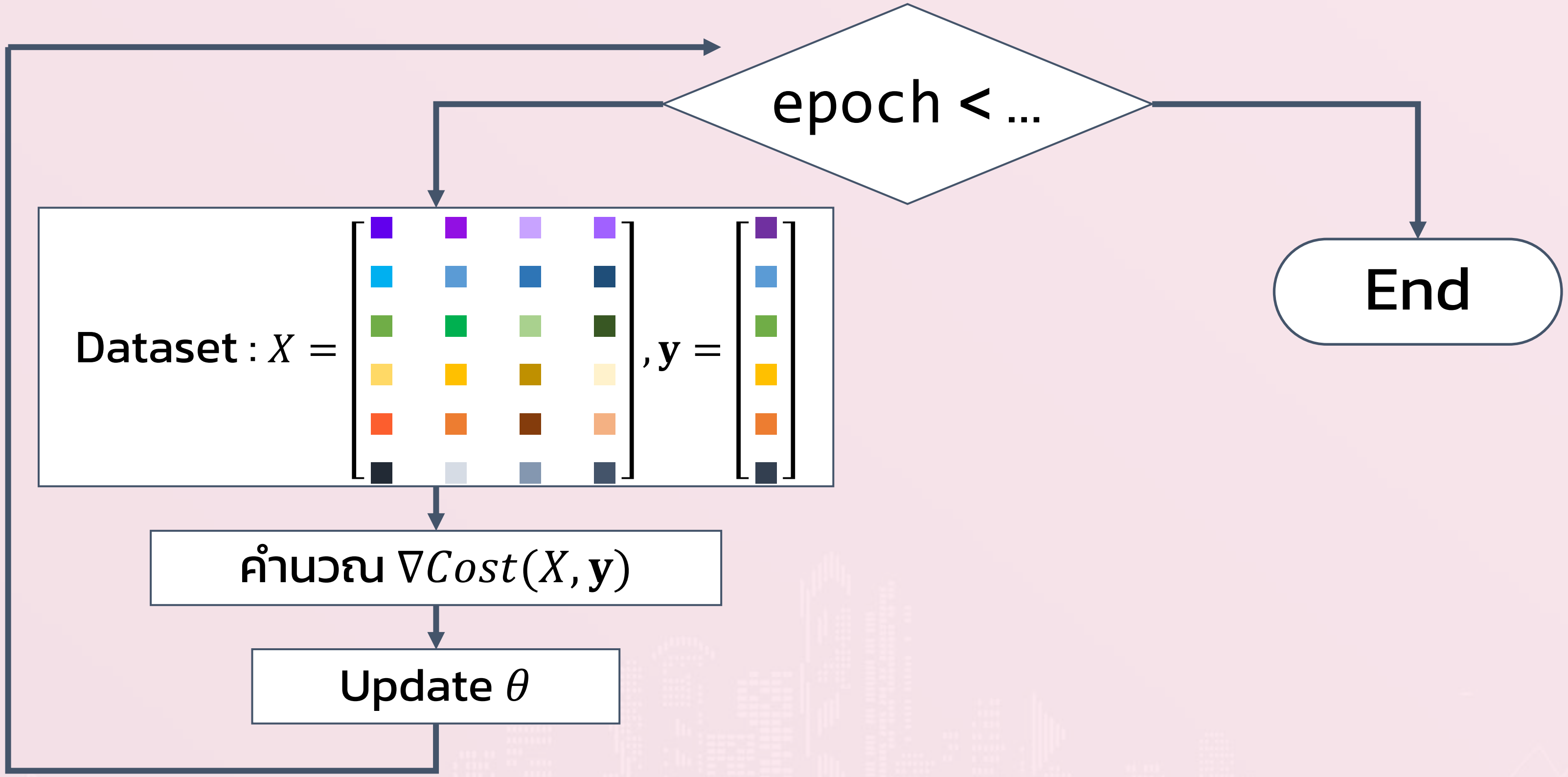
- **Batch Gradient Descent**
  1. กำหนดค่า θ เริ่มต้น
  2. กำหนดจำนวนรอบที่จะ update θ (epoch)
  3. กำหนดค่า α
  4. For loop เพื่อ Update θ

     2.1 คำนวณ $\nabla Cost$ ของทุก sample

     2.2 Update θ

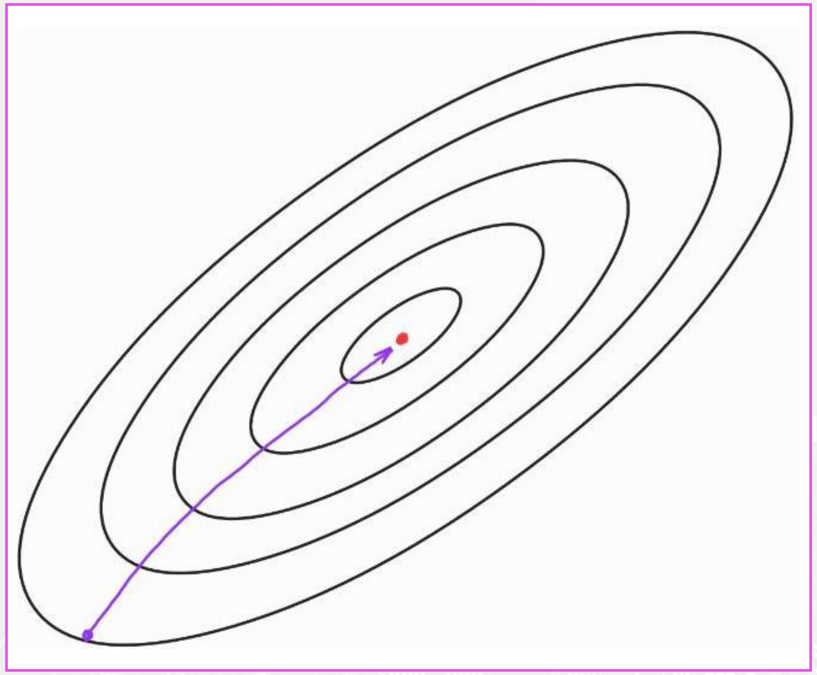
$$\theta_t = \theta_{t-1} - \frac{\alpha}{n} \nabla Cost_{t-1}$$

Batch Gradient Descent



Batch Gradient Descent

- **Batch Gradient Descent**



✓ข้อดี✓

> มีความ stable สูง

×ข้อเสีย×

> Computation cost สูง
> ช้า

Batch Gradient Descent

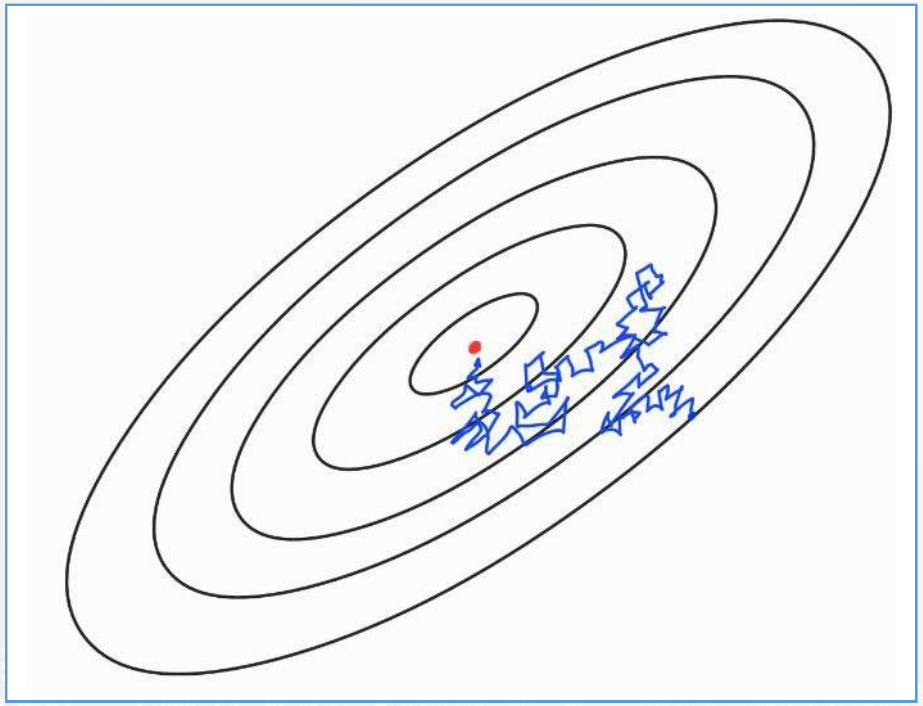
Gradient Descent Variants	ข้อดี	ข้อเสีย
Batch	• มีความ stable สูง	• Computational cost สูง • ช้า
Stochastic		
Minibatch		
Minibatch with learning rate		

Gradient Descent Variants

- ☑ **Batch Gradient Descent**
- ☐ Stochastic Gradient Descent
- ☐ Minibatch Gradient Descent

Stochastic Gradient Descent

- **Stochastic Gradient Descent**

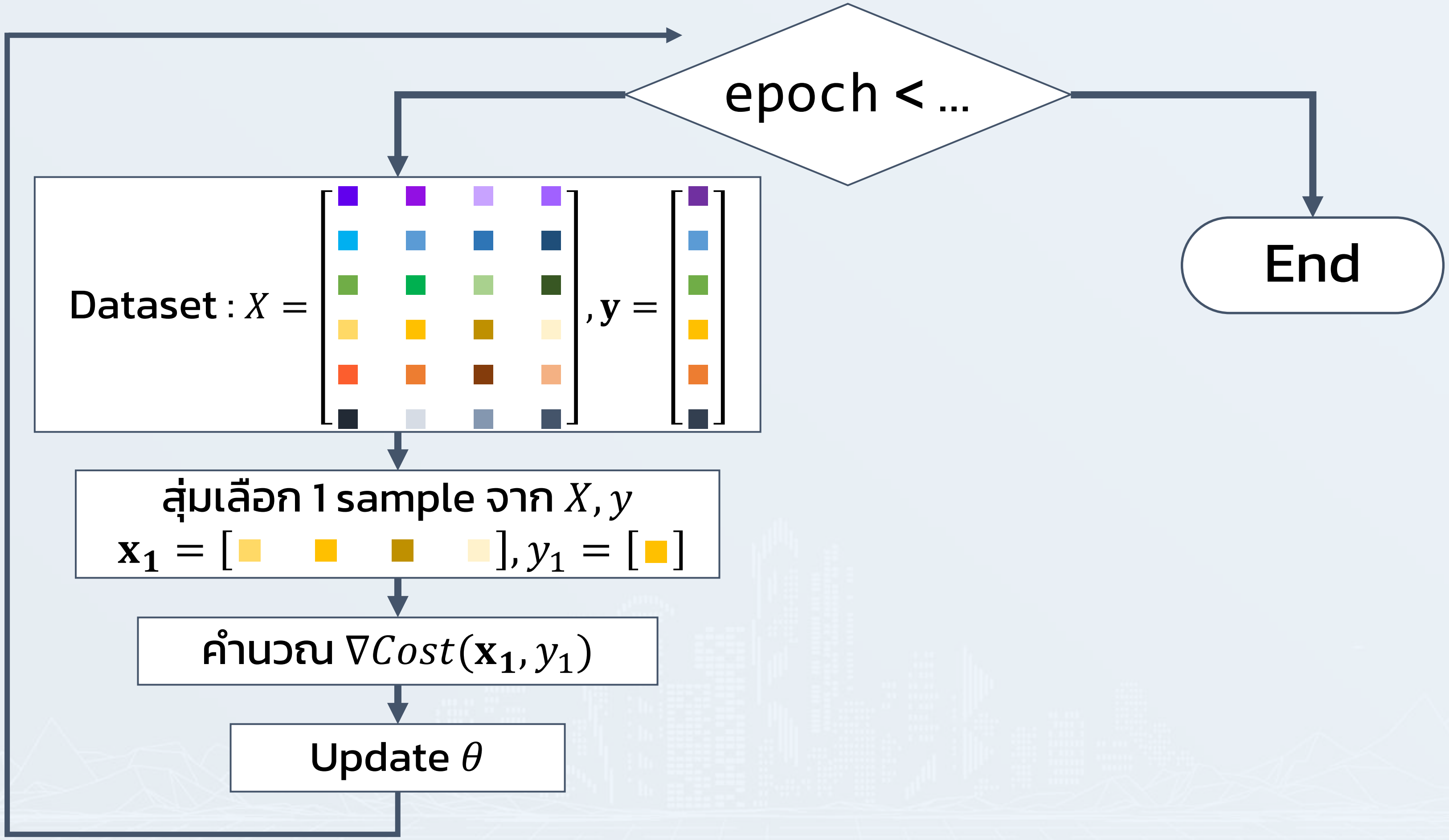


Stochastic Gradient Descent

- **Stochastic Gradient Descent**
  1. กำหนดค่า θ เริ่มต้น
  2. กำหนดจำนวนรอบที่จะ update θ (epoch)
  3. กำหนดค่า α
  4. For loop เพื่อ Update θ
     - 4.1 สุ่มเลือก 1 sample
     - 4.2 คำนวณ $\nabla Cost$ ของ sample นั้น
     - 4.3 Update θ

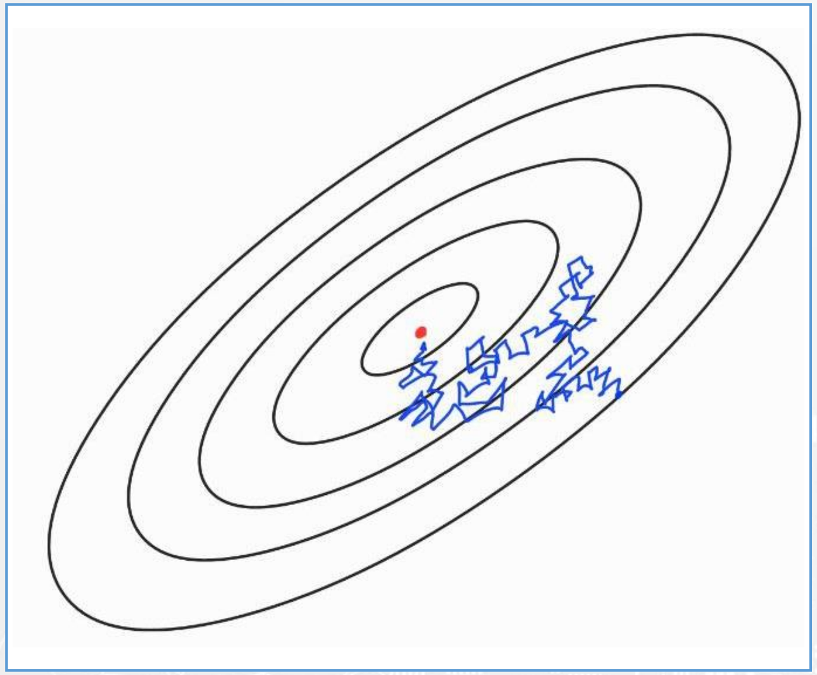
$$\boxed{\theta_t = \theta_{t-1} - \alpha \nabla Cost_{t-1}}$$

Stochastic Gradient Descent



Stochastic Gradient Descent

- **Stochastic Gradient Descent**



✓ข้อดี✓

> Computation cost ต่ำ
> เร็ว

✕ข้อเสีย✕

> มีความ stable ต่ำ

Stochastic Gradient Descent

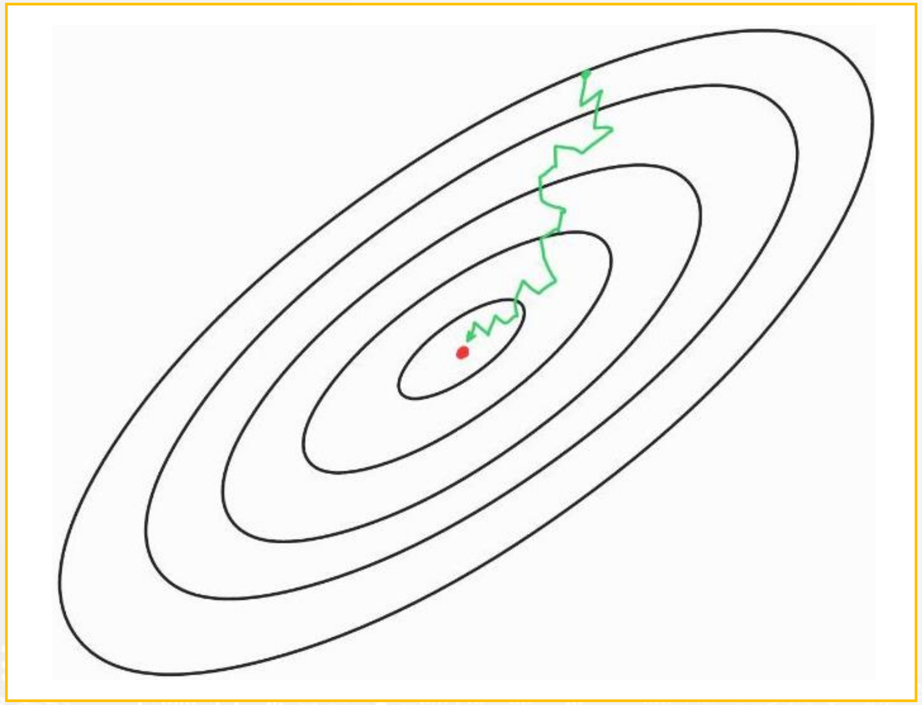
Gradient Descent Variants	ข้อดี	ข้อเสีย
Batch	• มีความ stable สูง	• Computational cost สูง • ช้า
Stochastic	• Computational cost ต่ำ • เร็ว	• มีความ stable ต่ำ
Minibatch		
Minibatch with learning rate		

Gradient Descent Variants

- ☑ **Batch Gradient Descent**
- ☑ **Stochastic Gradient Descent**
- ☐ Minibatch Gradient Descent

Minibatch Gradient Descent

- **Minibatch Gradient Descent**

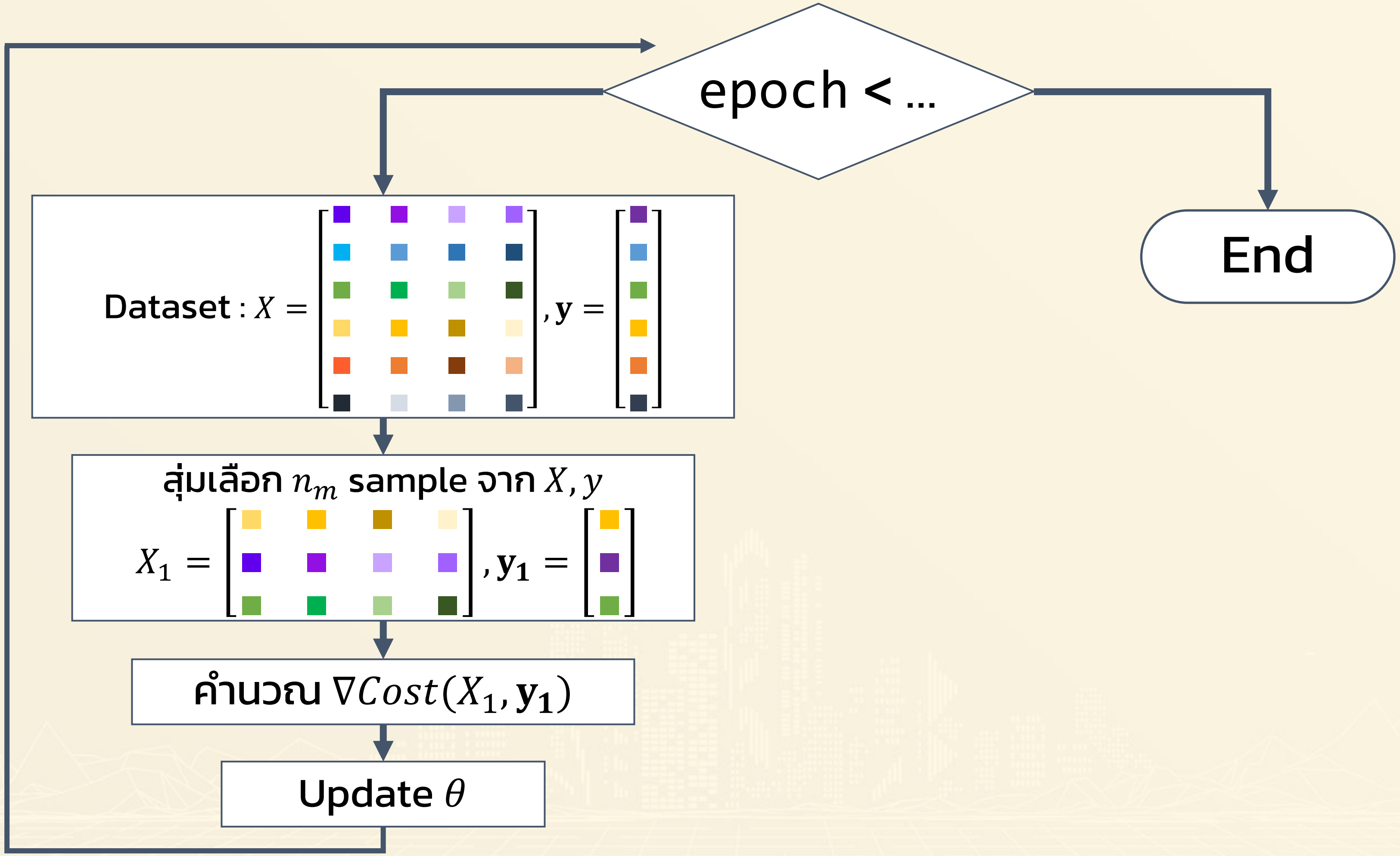


Minibatch Gradient Descent

- **Minibatch Gradient Descent**
  1. กำหนดค่า θ เริ่มต้น
  2. กำหนดจำนวนรอบที่จะ update θ (epoch)
  3. กำหนดค่า α
  4. For loop เพื่อ Update θ
     - 4.1 สุ่มเลือก n_m sample
     - 4.2 คำนวณ $\nabla Cost$ ของทั้ง n_m sample นั้น
     - 4.3 Update θ

$$\theta_t = \theta_{t-1} - \frac{\alpha}{n_m} \nabla Cost_{t-1}$$

Minibatch Gradient Descent

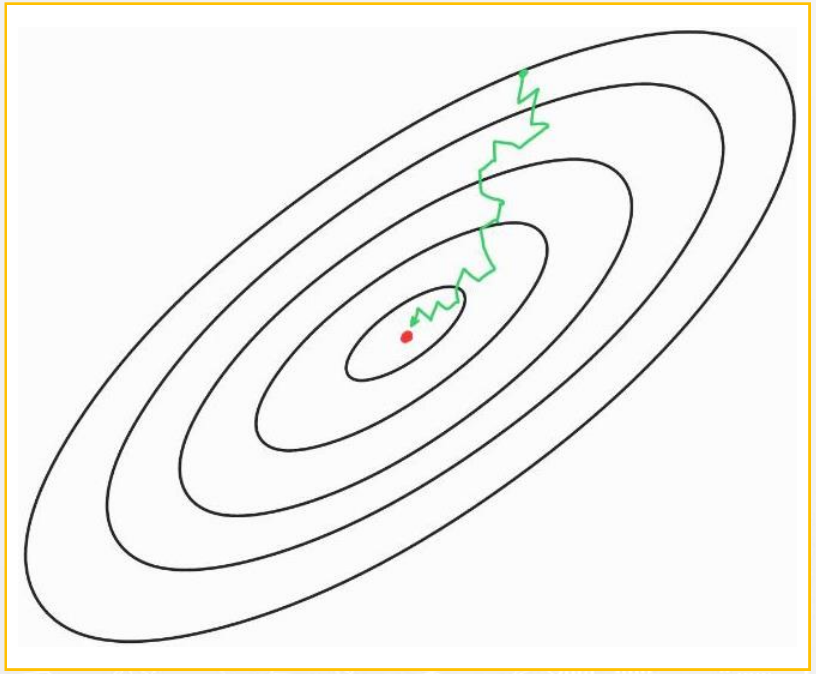


Minibatch Gradient Descent

เราต้องการให้แต่ละ sample ถูกนำมาใช้ train model ในจำนวนครั้งที่เท่ากัน

Minibatch Gradient Descent

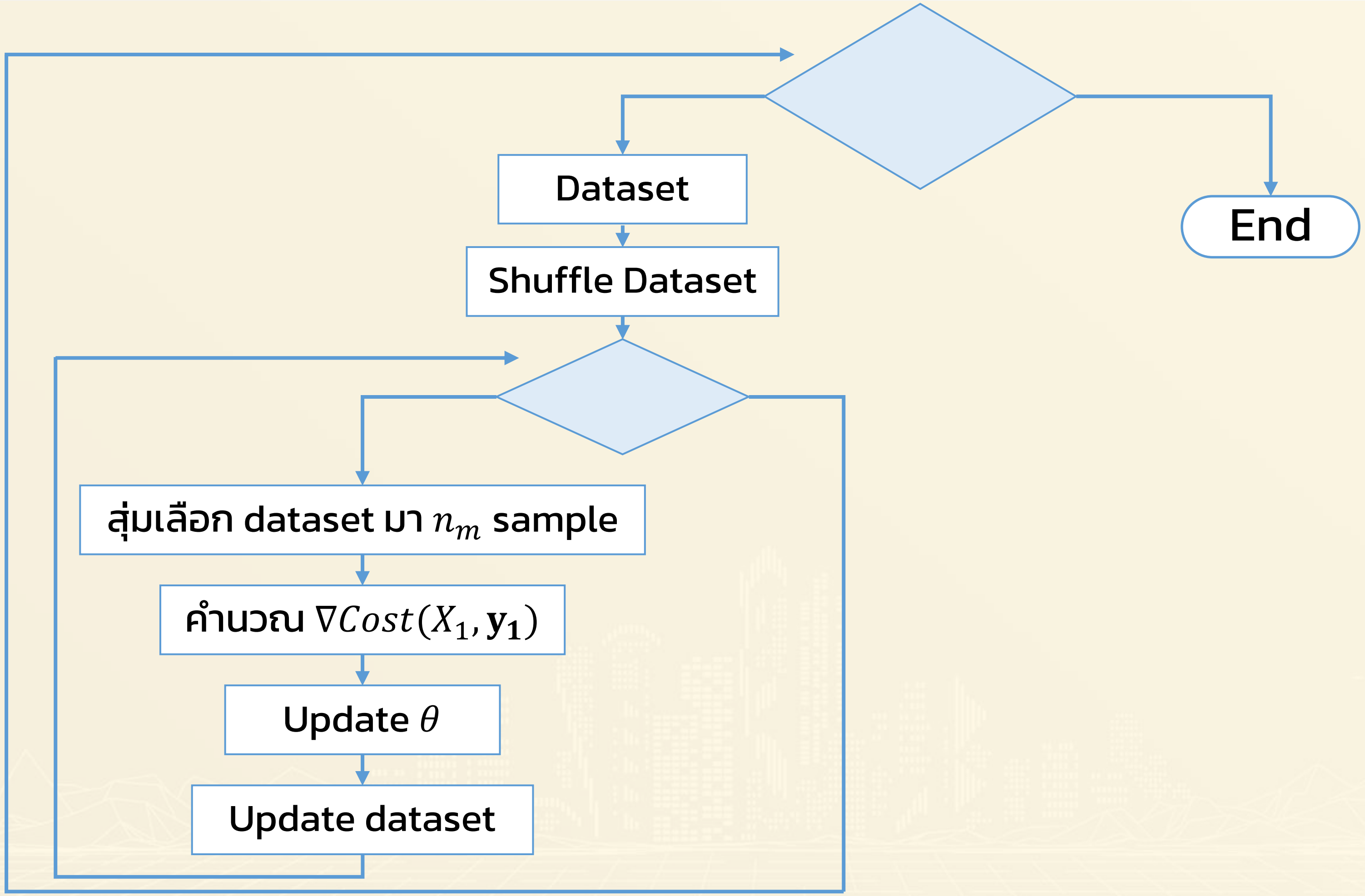
▪ Minibatch Gradient Descent (Nowadays)



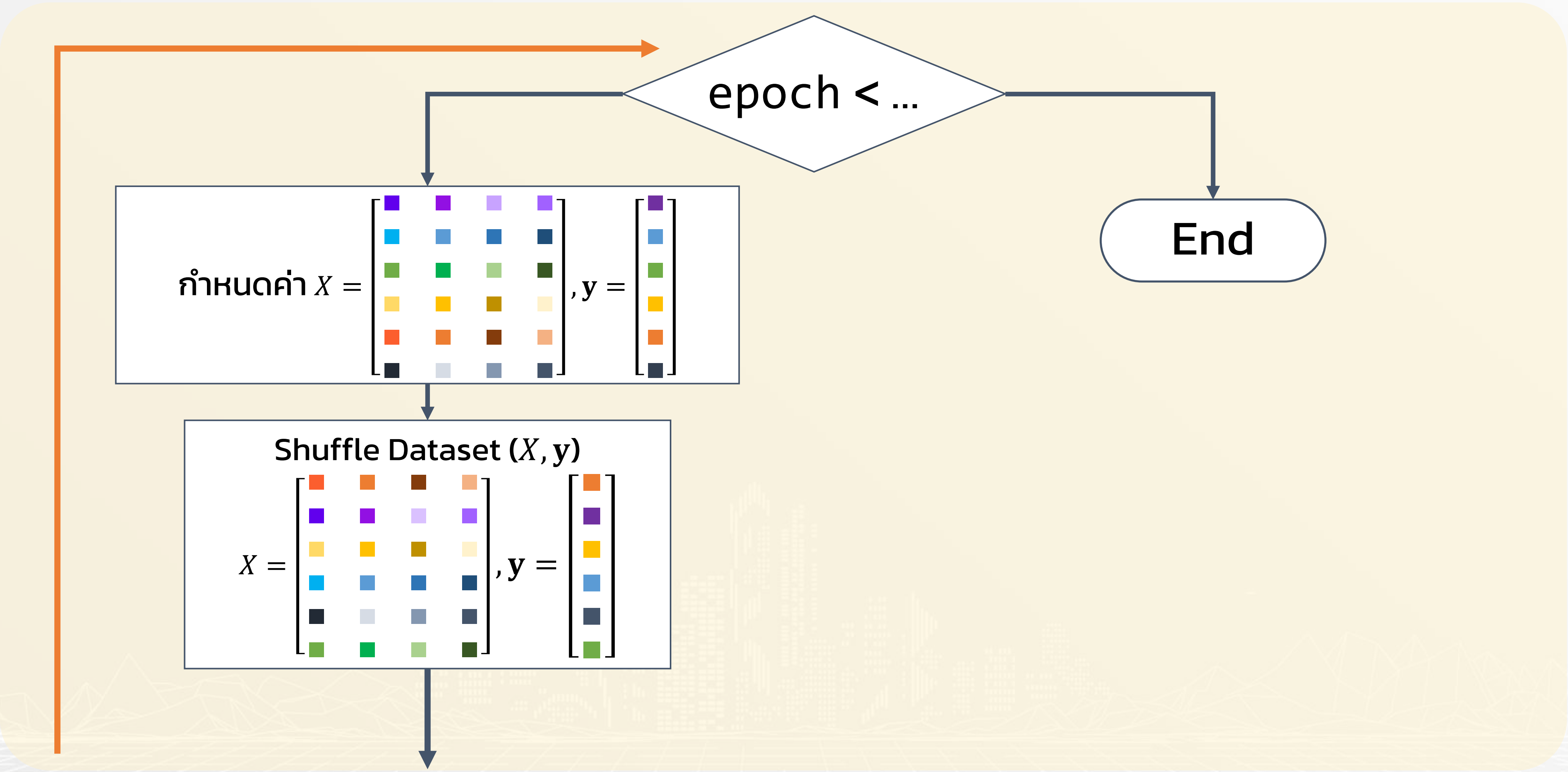
1. กำหนดค่า θ เริ่มต้น
2. กำหนดจำนวนรอบที่จะ update θ (epoch)
3. กำหนดค่า α
4. For loop เพื่อ Update θ
   4.1 Shuffle dataset
   4.2 for batch in get_batches(dataset, batch_size) :
   - คำนวณ $\nabla Cost$ ของ batch
   - Update θ

$$\theta_t = \theta_{t-1} - \frac{\alpha}{n_m} \nabla Cost_{t-1}$$

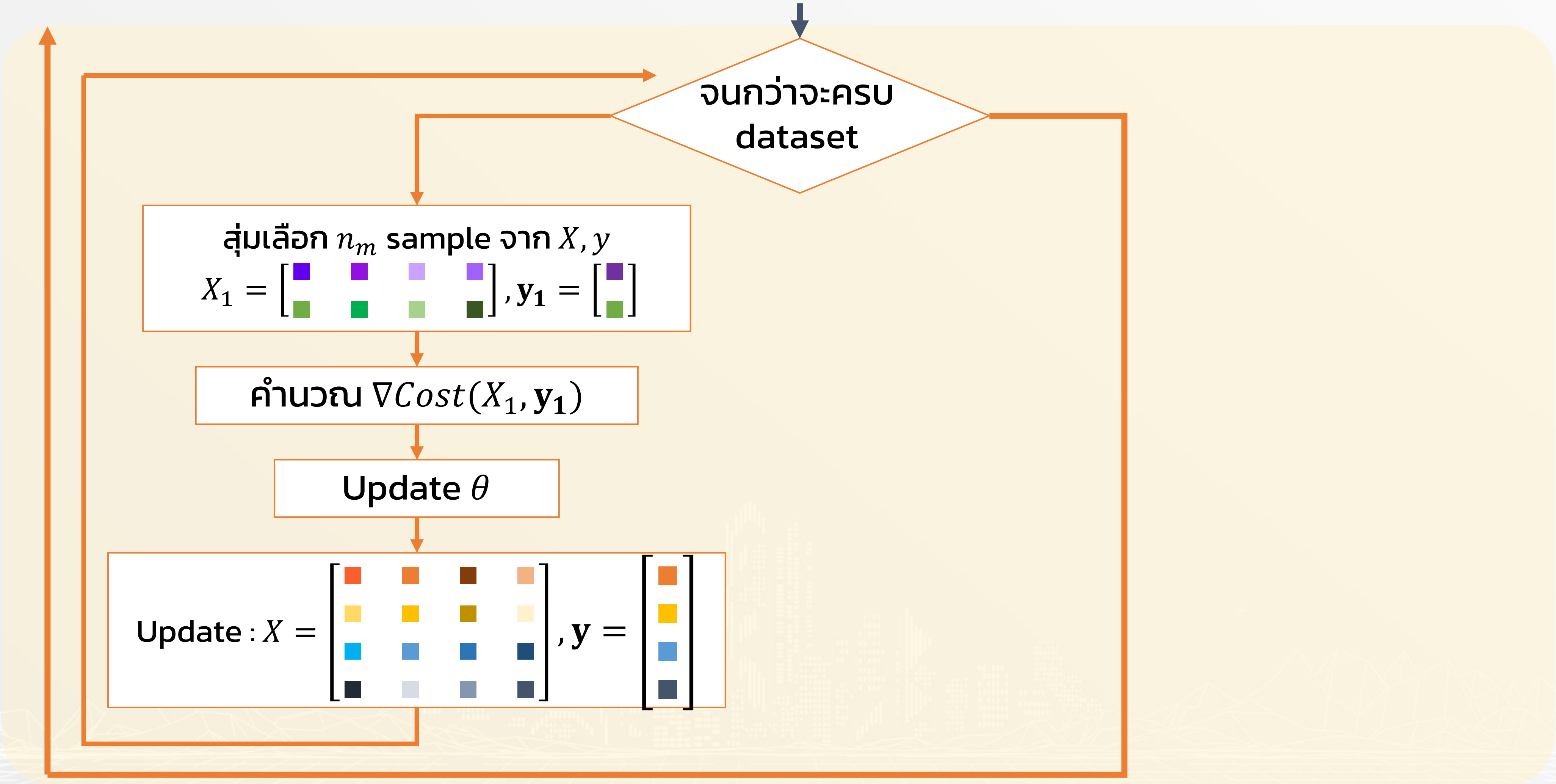
Minibatch Gradient Descent



Minibatch Gradient Descent

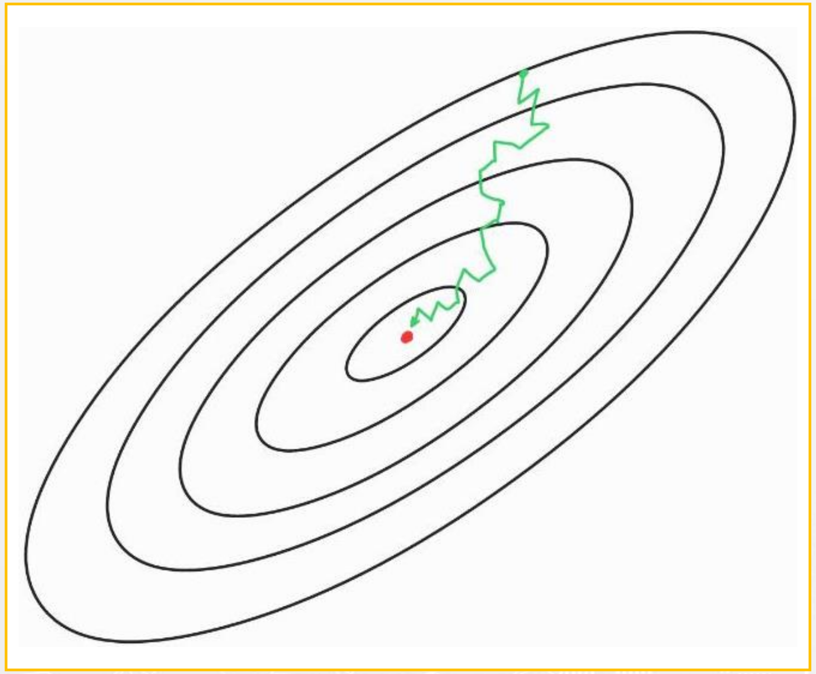


Minibatch Gradient Descent



Minibatch Gradient Descent

- **Minibatch Gradient Descent (Nowadays)**



✓ข้อดี✓

> Computational cost ต่ำกว่า Batch
> มีความ stable สูงกว่า Stochastic

✕ข้อเสีย✕

> Computational cost สูงกว่า Stochastic
> มีความ stable ต่ำกว่า Batch

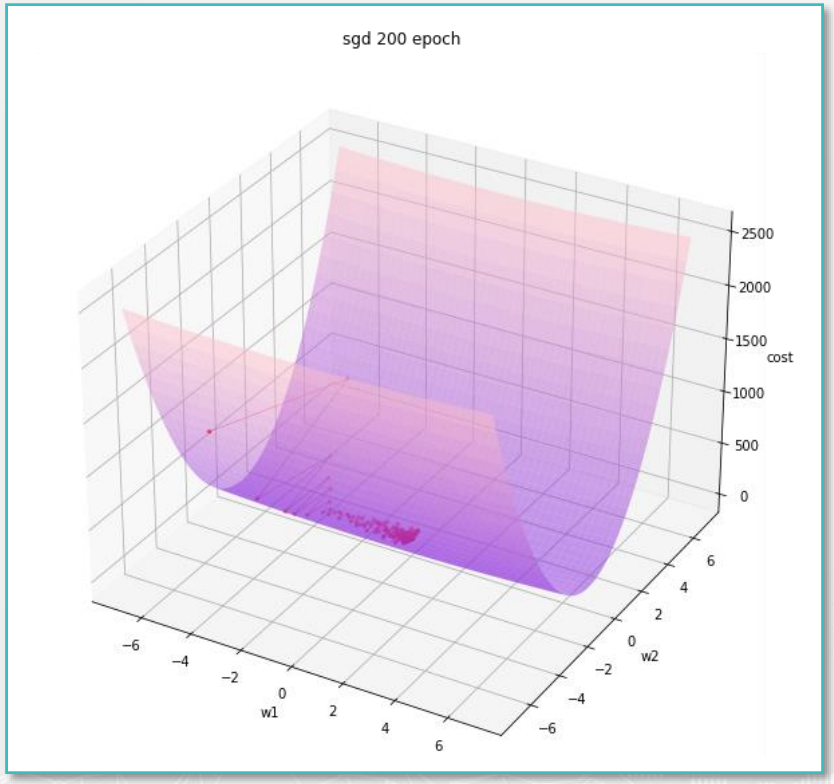
Minibatch Gradient Descent

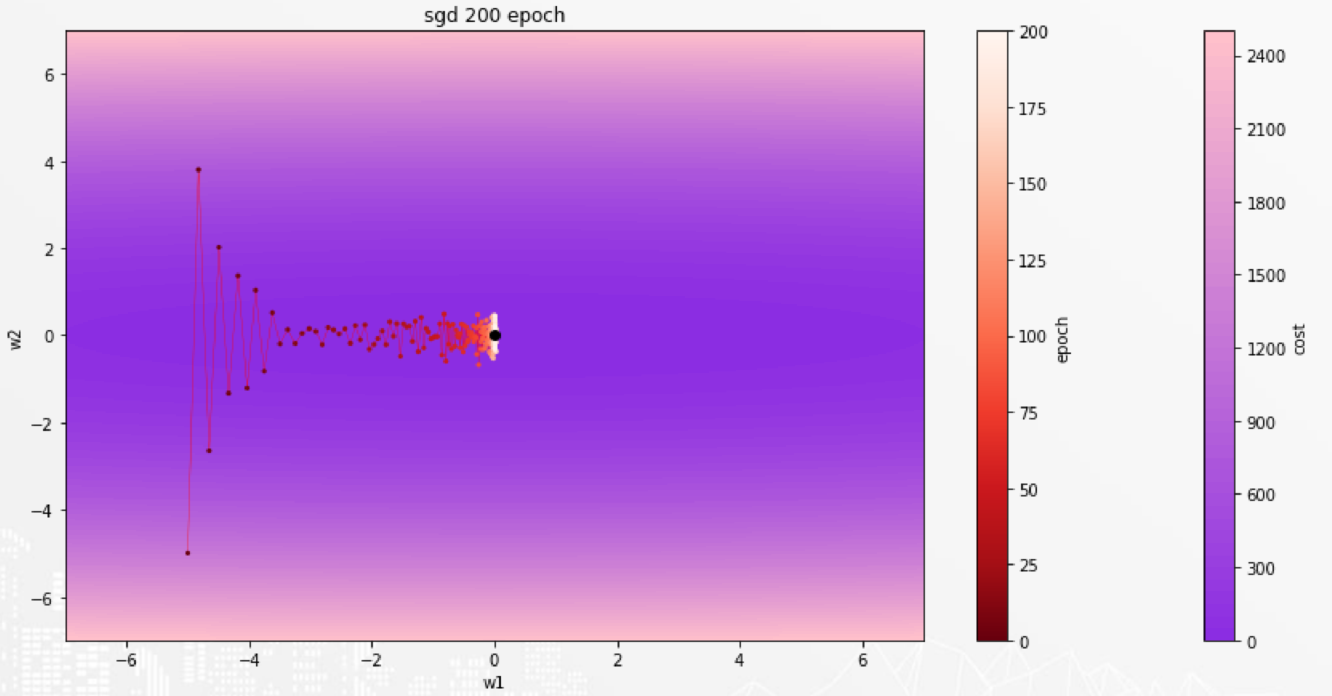
Gradient Descent Variants	ข้อดี	ข้อเสีย
Batch	• มีความ stable สูง	• Computational cost สูง • ช้า
Stochastic	• Computational cost ต่ำ • เร็ว	• มีความ stable ต่ำ
Minibatch	• Computational cost ต่ำกว่า Batch • มีความ stable สูงกว่า Stochastic	• Computational cost สูงกว่า Stochastic • มีความ stable ต่ำกว่า Batch

Minibatch Gradient Descent

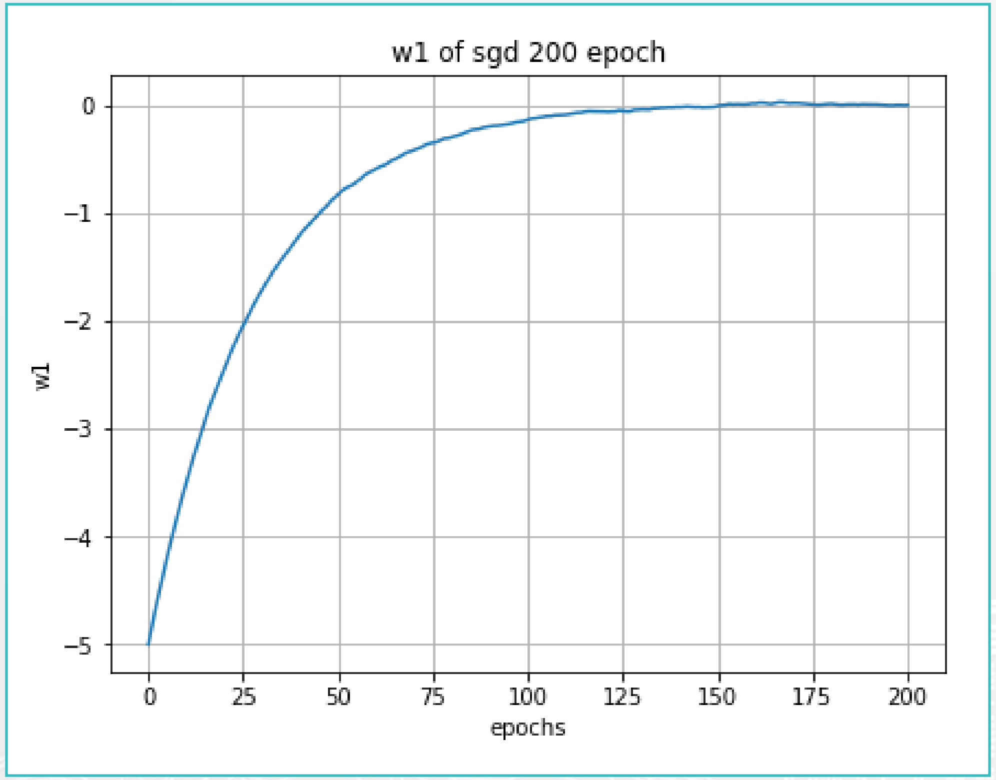
แต่การใช้ minibatch ก็ยังคงเกิดปัญหาเรื่อง
ความ stable เมื่อเข้าใกล้จุดต่ำสุดอยู่ดี

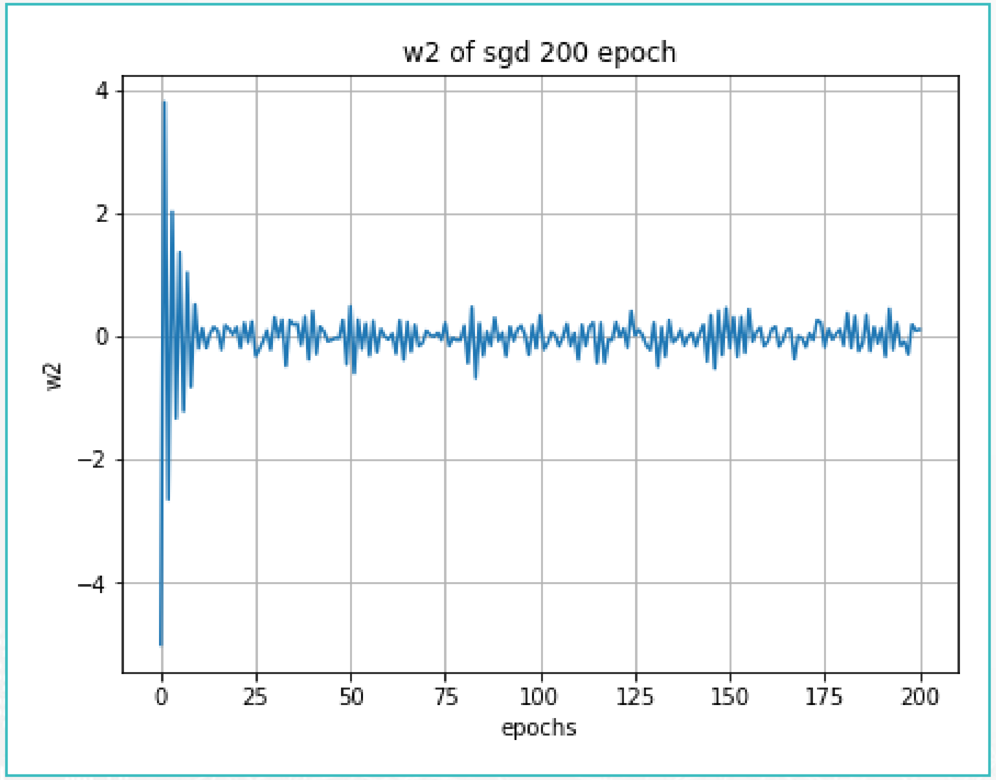
Minibatch Gradient Descent





Minibatch Gradient Descent





Gradient Descent Variants

- ☑ Batch Gradient Descent
- ☑ Stochastic Gradient Descent
- ☑ Minibatch Gradient Descent

Optimizer

Gradient Descent Variants

Optimization Algorithm

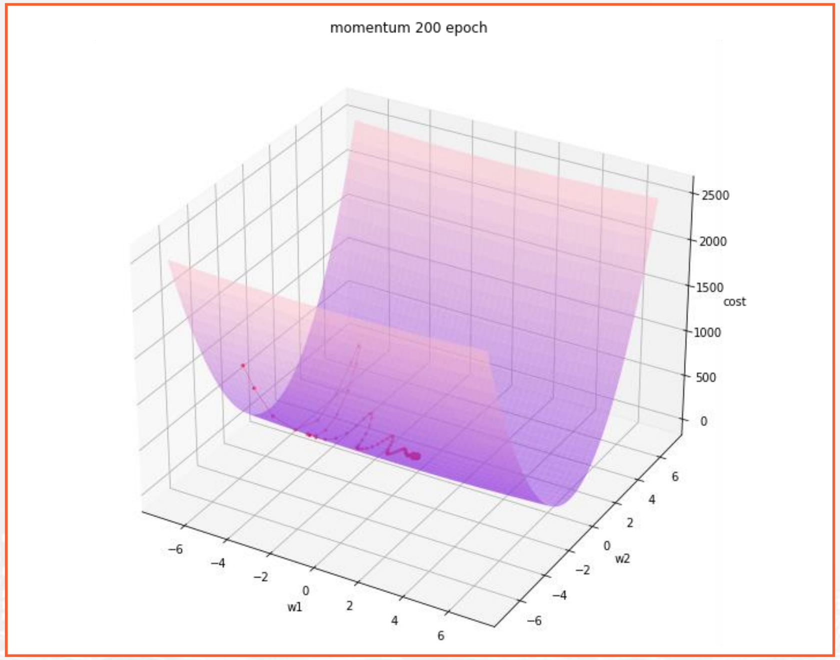
Code

Optimization Algorithm

- ☐ Momentum Base
- ☐ Adaptive Learning Rate Base Algorithm
- ☐ Adam
- ☐ Conclusion

Momentum Base

Momentum base คือ optimizer ที่พจน์ gradient จะถูกพิจารณาในรูปแบบของการสะสม gradient ของ epoch ก่อนหน้า รวมกับ gradient ของ epoch ปัจจุบัน



Momentum Base

Momentum base สามารถเขียนให้อยู่ในรูปดังนี้

$$\theta_t = \theta_{t-1} - v_t$$

$$v_t = \gamma v_{t-1} + \alpha g_t$$

$$g_t = \nabla Cost_{t-1}$$

โดย

- $\gamma \in (0,1)$ คือ ค่าคงที่ (ใช้เพื่อถ่วงน้ำหนักของ gradient ในรอบก่อนหน้า) (โดยปกติเท่ากับ 0.9)
- $v_o = 0$

Momentum Base

- **ขั้นตอนการคำนวณ**
  1. กำหนดค่า θ เริ่มต้น
  2. กำหนดจำนวนรอบที่จะ update θ (epoch)
  3. กำหนดค่า α
  4. For loop เพื่อ Update θ
     - 4.1 คำนวณ $\mathbf{g}$
     - 4.2 คำนวณ $\mathbf{v}$
     - 4.3 คำนวณ $\boldsymbol{\theta}$

Momentum Base

1. กำหนด $\boldsymbol{\theta}_0 = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$
2. กำหนด `epoch = 200`
3. กำหนดค่า $\alpha = 0.0015$
4. Update θ : `epoch = 1`

   4.1 $\mathbf{g}_1 = \begin{bmatrix} -10.04 \\ -495.21 \end{bmatrix}$

   4.2 $\mathbf{v}_1 = \begin{bmatrix} 0.9 \times 0 + 0.0015 \times (-10.04) \\ 0.9 \times 0 + 0.0015 \times (-495.21) \end{bmatrix} = \begin{bmatrix} -0.02 \\ -0.74 \end{bmatrix}$

   4.3 $\boldsymbol{\theta}_1 = \begin{bmatrix} -5 - (-0.02) \\ -5 - (-0.74) \end{bmatrix} = \begin{bmatrix} -4.98 \\ -4.26 \end{bmatrix}$

Momentum Base

4. Update θ : epoch = 2

4.1 $\mathbf{g_2} = \begin{bmatrix} -10.07 \\ -431.28 \end{bmatrix}$

4.2 $\mathbf{v_2} = \begin{bmatrix} 0.9 \times (-0.02) + 0.0015 \times (-10.07) \\ 0.9 \times (-0.74) + 0.0015 \times (-431.28) \end{bmatrix} = \begin{bmatrix} -0.03 \\ -1.32 \end{bmatrix}$

4.3 $\boldsymbol{\theta_2} = \begin{bmatrix} -4.98 - (-0.03) \\ -4.26 - (-1.32) \end{bmatrix} = \begin{bmatrix} -4.96 \\ -2.94 \end{bmatrix}$

Momentum Base

4. Update θ : epoch = 3

4.1 $\mathbf{g_3} = \begin{bmatrix} -9.52 \\ -280.24 \end{bmatrix}$

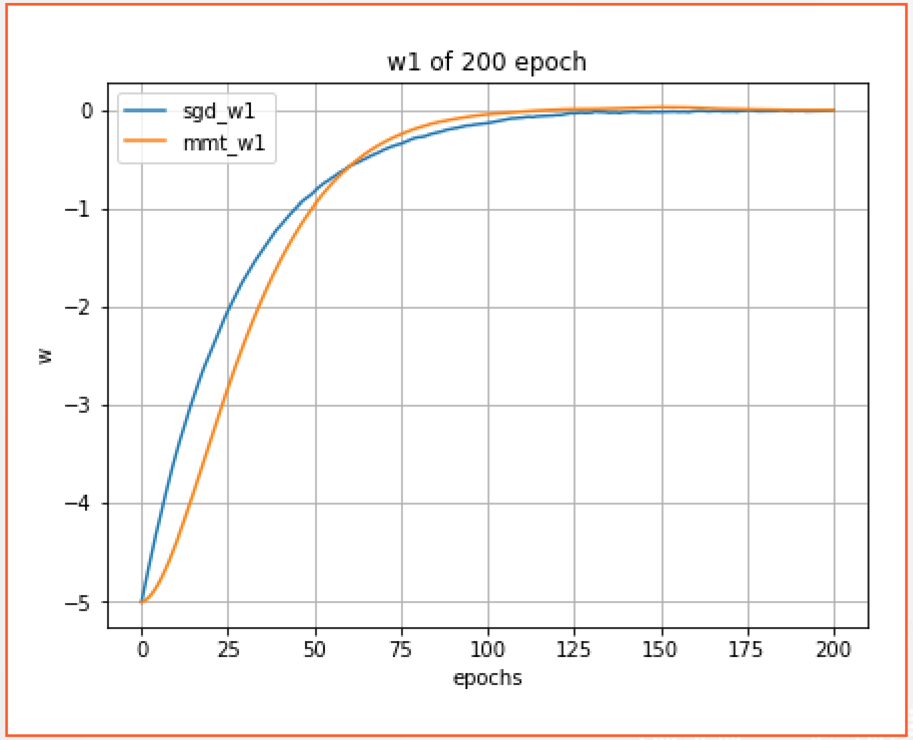
4.2 $\mathbf{v_3} = \begin{bmatrix} 0.9 \times (-0.03) + 0.0015 \times (-9.52) \\ 0.9 \times (-1.32) + 0.0015 \times (-280.24) \end{bmatrix} = \begin{bmatrix} -0.04 \\ -1.60 \end{bmatrix}$

4.3 $\boldsymbol{\theta_3} = \begin{bmatrix} -4.96 - (-0.04) \\ -2.94 - (-1.60) \end{bmatrix} = \begin{bmatrix} -4.92 \\ -1.34 \end{bmatrix}$

Momentum Base

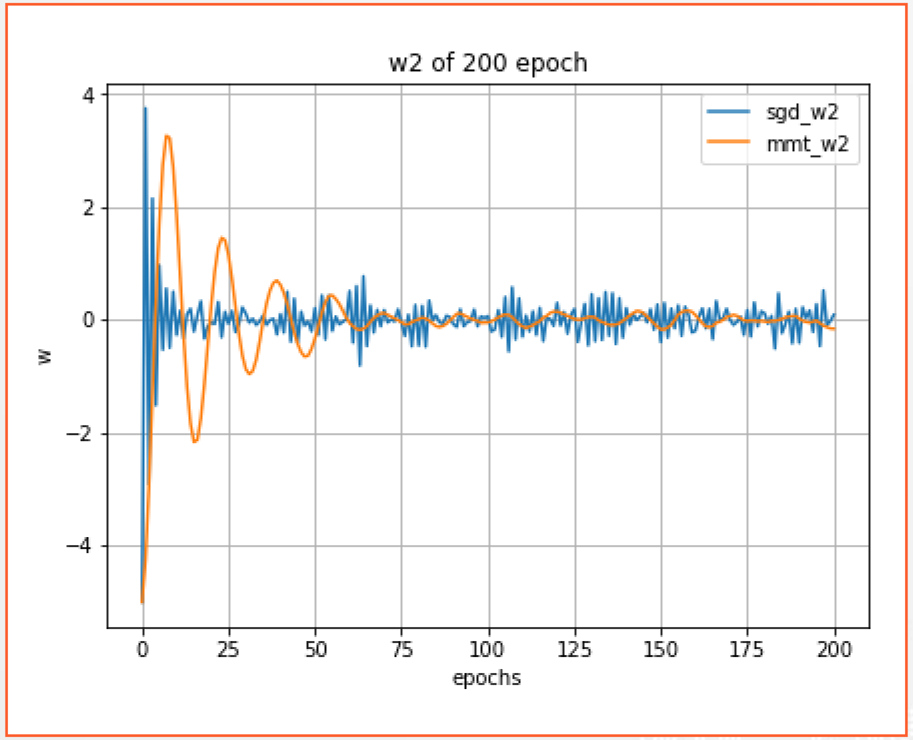
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.98	-4.26
3	-4.96	-2.94
4	-4.87	0.3
5	-4.81	1.71
6	-4.74	2.75
⋮	⋮	⋮
198	-0.01	-0.13
199	-0.01	-0.15
200	-0.01	-0.16

Momentum Base



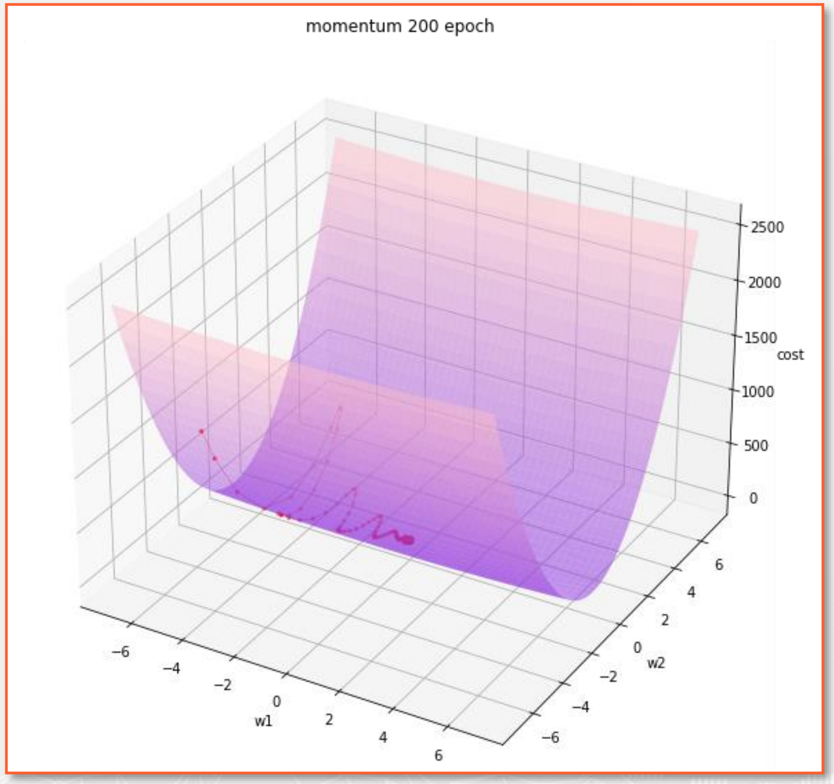
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.98	-4.26
3	-4.96	-2.94
4	-4.87	0.3
5	-4.81	1.71
6	-4.74	2.75
⋮	⋮	⋮
198	-0.01	-0.13
199	-0.01	-0.15
200	-0.01	-0.16

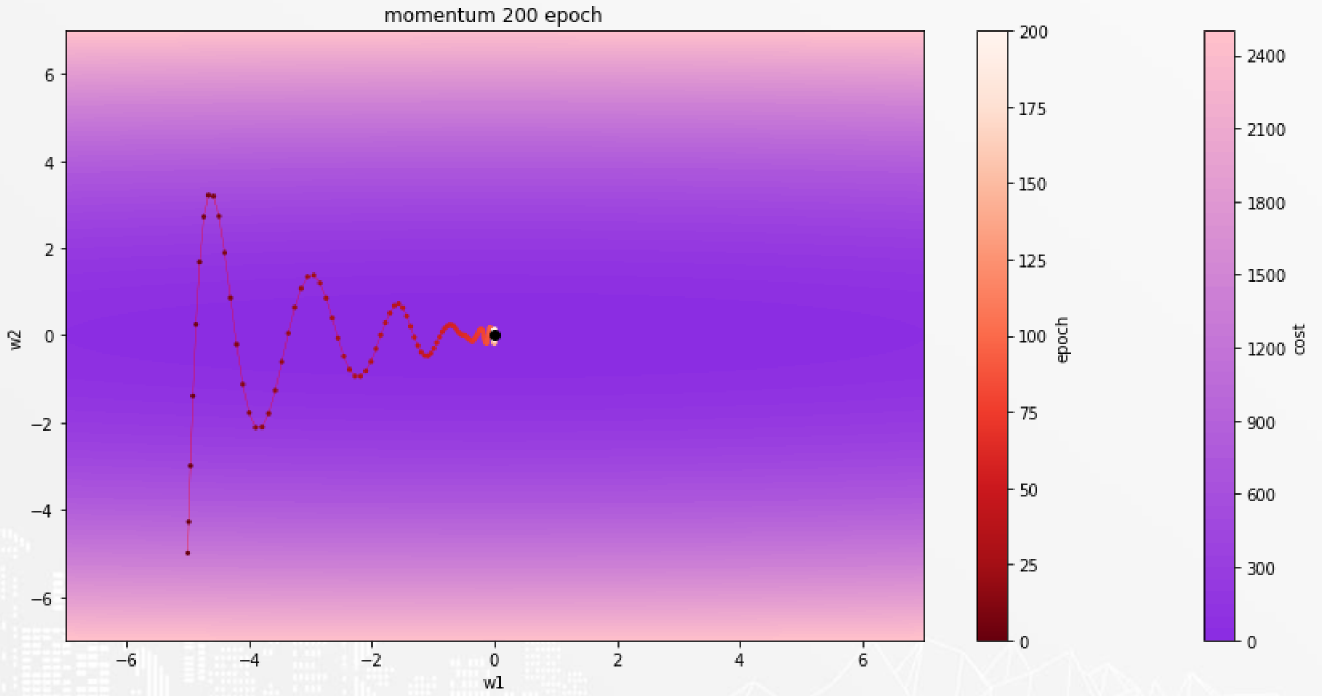
Momentum Base



epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.98	-4.26
3	-4.96	-2.94
4	-4.87	0.3
5	-4.81	1.71
6	-4.74	2.75
⋮	⋮	⋮
198	-0.01	-0.13
199	-0.01	-0.15
200	-0.01	-0.16

Momentum Base





Optimization Algorithm

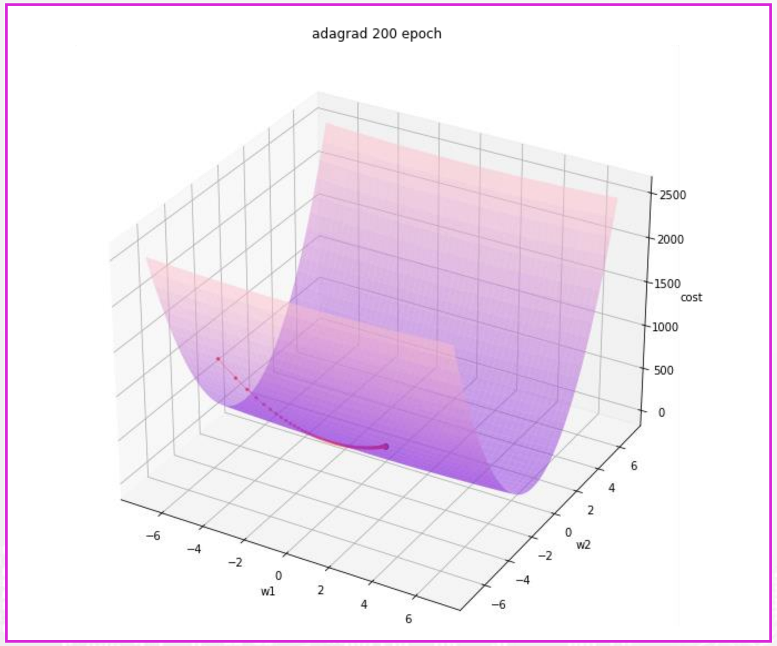
- ☑ **Momentum Base**
- ☐ Adaptive Learning Rate Base Algorithm
- ☐ Adam
- ☐ Conclusion

Adaptive Learning Rate Base Algorithm

- Adagrad
- RMSprop

Adagrad

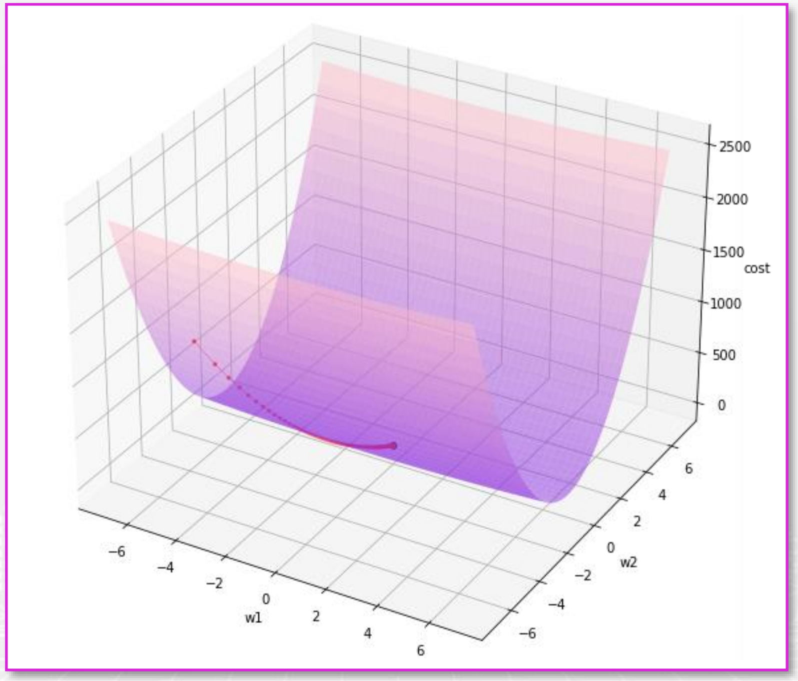
Adagrad (Adaptive gradient algorithm) คือ optimizer ที่จะถูกพิจารณาในรูปแบบของการสะสม gradient แล้วยังมีการปรับ learning rate ที่ไม่เท่ากันในแต่ละมิติ



Adagrad

โดยวิธีการปรับ learning rate จะหลักการคือ

" ปรับ(ลด) learning rate ตาม gradient ที่มีการเคลื่อนที่ผ่านมาแล้ว "



Adagrad

Adagrad สามารถเขียนให้อยู่ในรูปดังนี้

$$\theta_t = \theta_{t-1} - S_t \odot g_t$$
$$S_t = \frac{\alpha}{\sqrt{G_t + \varepsilon}}$$
$$G_t = G_{t-1} + (g_t \odot g_t)$$
$$g_t = \nabla Cost_{t-1}$$

โดย

- G_t คือ พจน์ของการสะสมขนาดของ gradient ในแต่ละมิติ และ $G_0 = 0$
- $\varepsilon = 1 \times 10^{-8}$ (ป้องกันการหารด้วย 0)

Adagrad

- **ขั้นตอนการคำนวณ**
  1. กำหนดค่า θ เริ่มต้น
  2. กำหนดจำนวนรอบที่จะ update θ (epoch)
  3. กำหนดค่า α
  4. For loop เพื่อ Update θ

     4.1 คำนวณ $\mathbf{g}$

     4.2 คำนวณ $\mathbf{G}$

     4.2 คำนวณ $\mathbf{S}$

     4.3 คำนวณ $\boldsymbol{\theta}$

Adagrad

1. กำหนด $\boldsymbol{\theta_0} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$
2. กำหนด `epoch = 200`
3. กำหนดค่า $\alpha = 0.0015$

Adagrad

4. Update θ : epoch = 1

4.1 $\mathbf{g_1} = \begin{bmatrix} -10.04 \\ -495.21 \end{bmatrix}$

4.2 $\mathbf{G_1} = \begin{bmatrix} 0 + (-10.04)^2 \\ 0 + (-495.21)^2 \end{bmatrix} = \begin{bmatrix} 100.82 \\ 245233.51 \end{bmatrix}$

4.3 $\mathbf{S_1} = \begin{bmatrix} \frac{0.5}{\sqrt{100.82+(1\times10^{-8})}} \\ \frac{0.5}{\sqrt{245233.51+(1\times10^{-8})}} \end{bmatrix} = \begin{bmatrix} \frac{0.5}{10.04} \\ \frac{0.5}{495.21} \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.001 \end{bmatrix}$

4.4 $\mathbf{\theta_1} = \begin{bmatrix} -5 - (0.05 \times -10.04) \\ -5 - (0.001 \times -495.21) \end{bmatrix} = \begin{bmatrix} -4.5 \\ -4.5 \end{bmatrix}$

Adagrad

4. Update θ : $\boxed{\text{epoch} = 2}$

4.1 $\mathbf{g_2} = \begin{bmatrix} -9.1 \\ -455.56 \end{bmatrix}$

4.2 $\mathbf{G_2} = \begin{bmatrix} 100.82 + (-9.1)^2 \\ 245233.51 + (-455.56)^2 \end{bmatrix} = \begin{bmatrix} 183.7 \\ 452765.96 \end{bmatrix}$

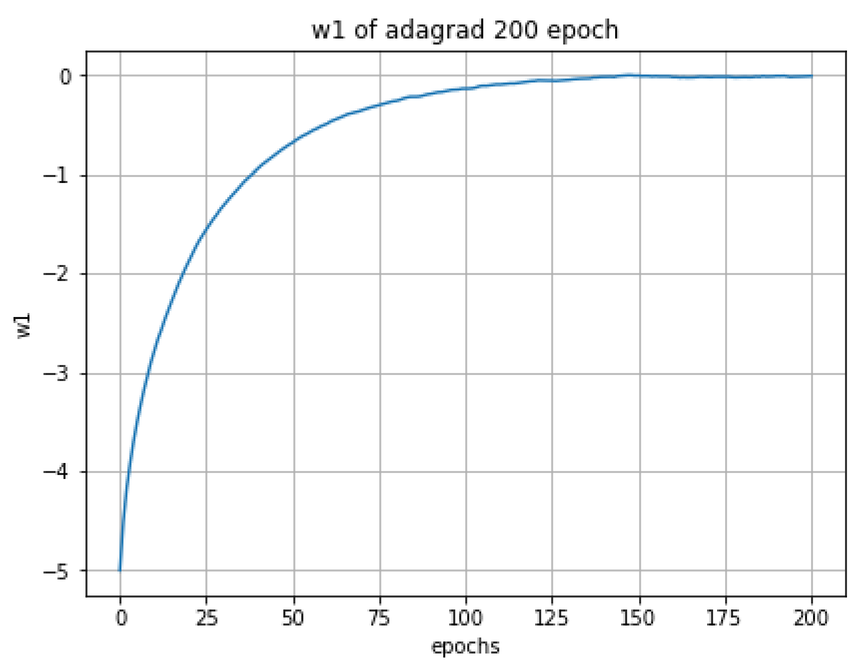
4.3 $\mathbf{S_2} = \begin{bmatrix} \frac{0.5}{\sqrt{183.7+(1\times10^{-8})}} \\ \frac{0.5}{\sqrt{452765.96+(1\times10^{-8})}} \end{bmatrix} = \begin{bmatrix} \frac{0.5}{13.55} \\ \frac{0.5}{672.88} \end{bmatrix} = \begin{bmatrix} 0.04 \\ 0.0007 \end{bmatrix}$

4.4 $\boldsymbol{\theta_2} = \begin{bmatrix} -4.5 - (0.04 \times -9.1) \\ -4.5 - (0.0007 \times -455.56) \end{bmatrix} = \begin{bmatrix} -4.16 \\ -4.16 \end{bmatrix}$

Adagrad

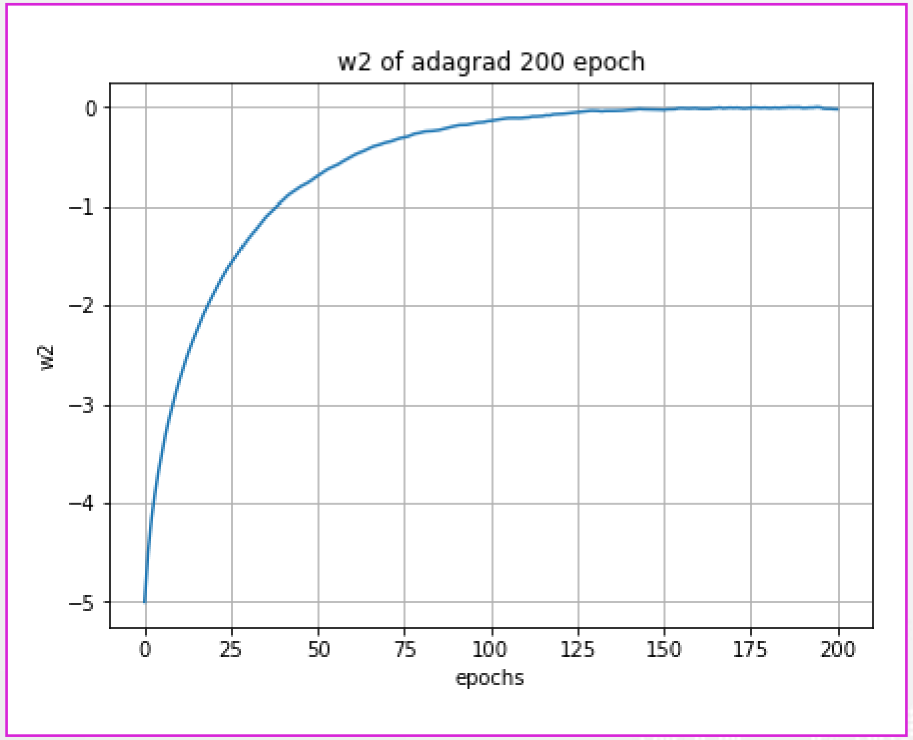
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.5	-4.5
3	-4.16	-4.16
4	-3.91	-3.9
5	-3.69	-3.68
6	-3.5	-3.49
⋮	⋮	⋮
198	-0.01	-0.02
199	-0.01	-0.02
200	-0.01	-0.02

Adagrad



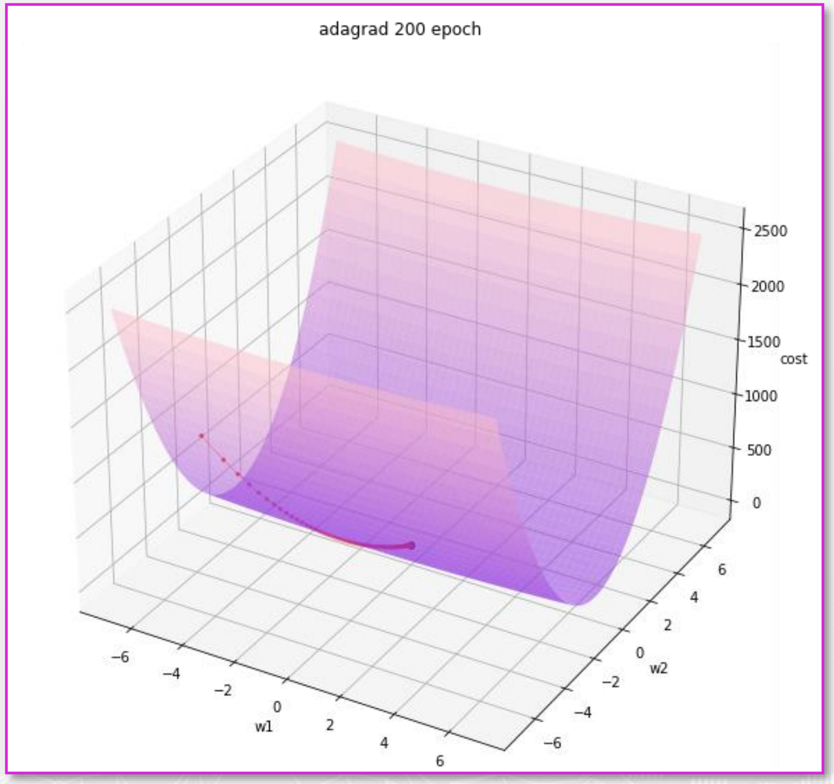
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.5	-4.5
3	-4.16	-4.16
4	-3.91	-3.9
5	-3.69	-3.68
6	-3.5	-3.49
$\vdots$	$\vdots$	$\vdots$
198	-0.01	-0.02
199	-0.01	-0.02
200	-0.01	-0.02

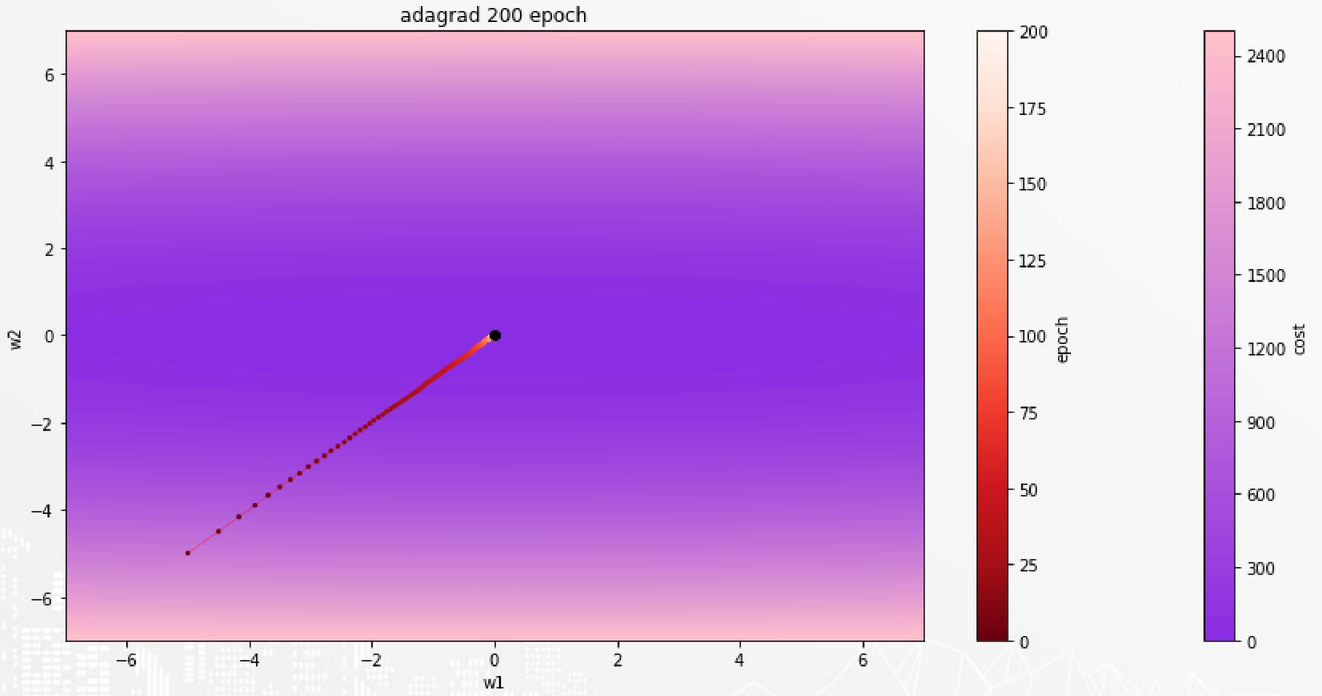
Adagrad



epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.5	-4.5
3	-4.16	-4.16
4	-3.91	-3.9
5	-3.69	-3.68
6	-3.5	-3.49
⋮	⋮	⋮
198	-0.01	-0.02
199	-0.01	-0.02
200	-0.01	-0.02

Adagrad


adagrad 200 epoch
2500
2000
1500
1000
500
0
cost
w1
w2


adagrad 200 epoch
w1
w2
epoch
cost

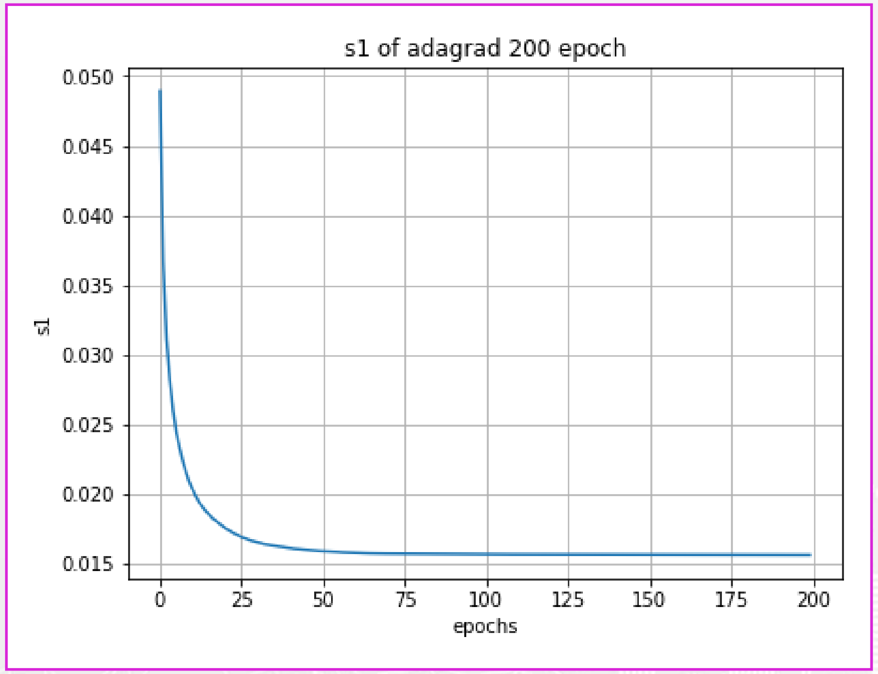
Adagrad

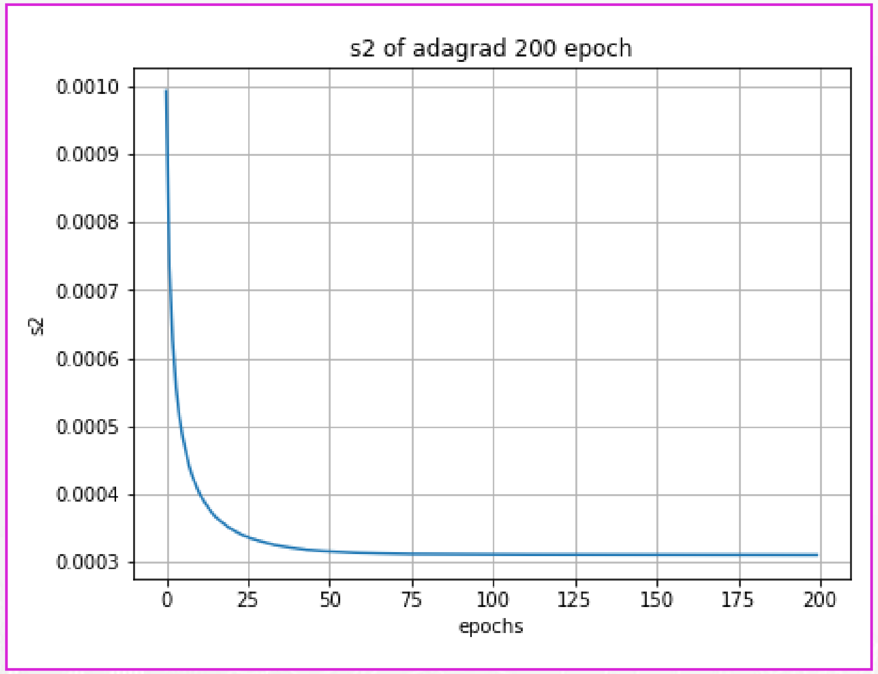
แต่ปัญหาของ adagrad ก็คือ เมื่อทำการ train model ไปนาน ๆ แล้ว learning rate จะลู่เข้าสู่ 0

(ถึงแม้ว่า model จะยัง train ไม่เสร็จก็ตาม)

Adagrad

Adagrad problem



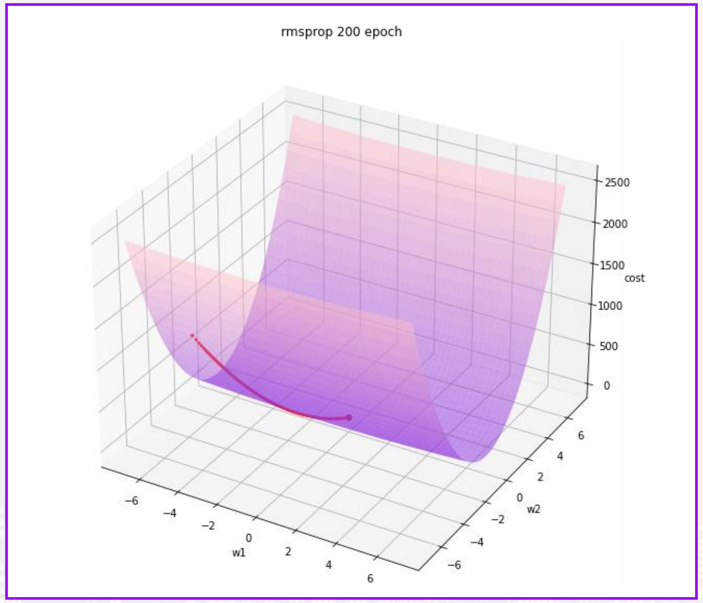


Adaptive Learning Rate Base Algorithm

- ✓ **Adagrad**
- ○ RMSprop

RMSprop

RMSprop (Root Mean Square propagation) คือ algorithm ที่ใช้ในการแก้ไขปัญหาการหยุดนิ่งของการเคลื่อนที่ของ θ ใน epoch ท้าย ๆ ของ adagrad



RMSprop

ซึ่งวิธีแก้ปัญหานั้นคือ เปลี่ยน**พจน์สะสม gradient กำลังสอง** ให้อยู่ในรูป **exponential moving average**

RMSprop

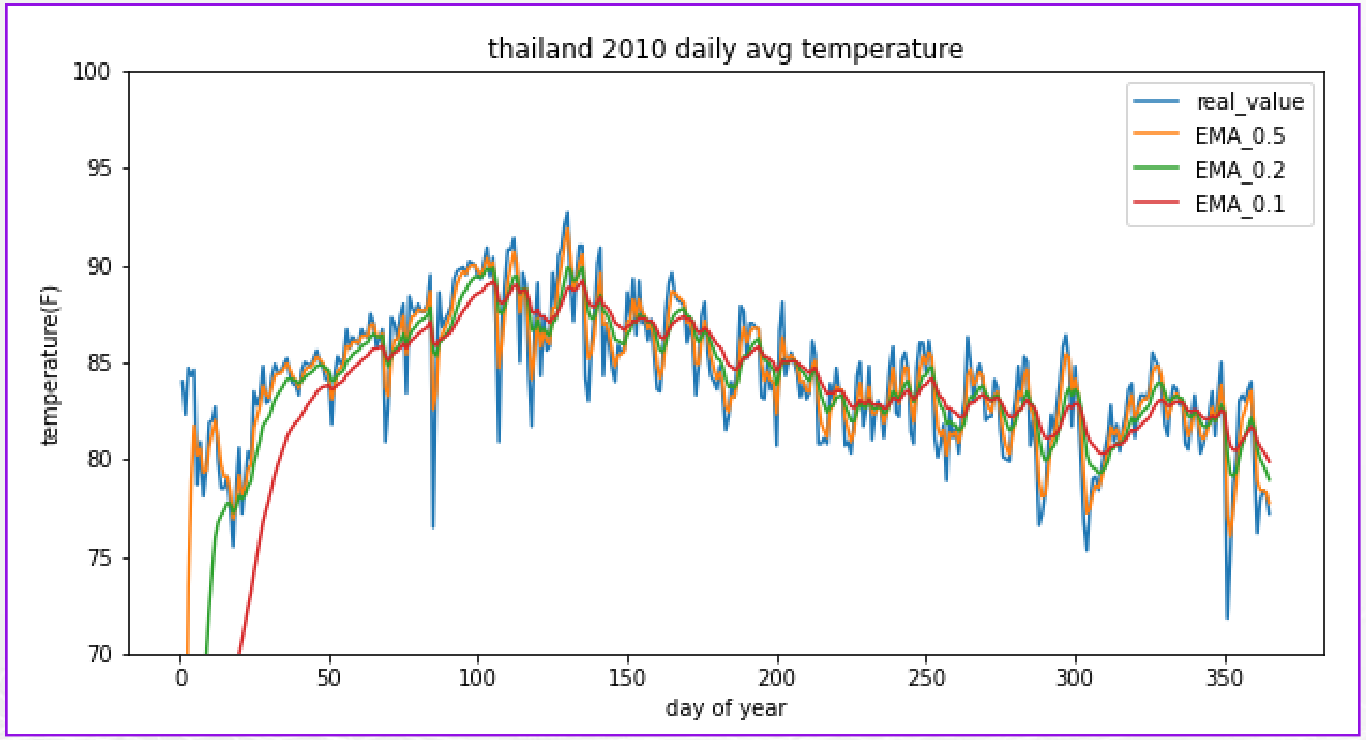
Exponential moving average คือการหาค่าเฉลี่ยถ่วงน้ำหนักแบบ exponential ของตัวแปรที่มีการเปลี่ยนแปลงตามเวลา

$$EMA_t = (1-\beta)EMA_{t-1} + \beta x_t$$

โดย
- $\beta \in (0,1)$ คือ ค่าคงที่ (เพื่อให้เกิดการลดลงของน้ำหนักที่เวลาก่อนหน้า)

RMSprop

Exponential moving average



RMSprop

RMSprop สามารถเขียนให้อยู่ในรูปดังนี้

$$\theta_t = \theta_{t-1} - S_t \odot g_t$$

$$S_t = \frac{\alpha}{\sqrt{G_t + \varepsilon}}$$

$$G_t = \beta G_{t-1} + (1 - \beta)(g_t \odot g_t)$$

$$g_t = \nabla Cost_{t-1}$$

โดย
- G_t คือ พจน์ของการสะสมขนาดของ gradient ในแต่ละมิติ และ $G_0 = 0$
- $\varepsilon = 1 \times 10^{-8}$ (ป้องกันการหารด้วย 0)

RMSprop

- **ขั้นตอนการคำนวณ**
  1. กำหนดค่า θ เริ่มต้น
  2. กำหนดจำนวนรอบที่จะ update θ (epoch)
  3. กำหนดค่า α, β
  4. For loop เพื่อ Update θ

     4.1 คำนวณ $\mathbf{g}$

     4.2 คำนวณ $\mathbf{G}$

     4.2 คำนวณ $\mathbf{S}$

     4.3 คำนวณ $\boldsymbol{\theta}$

RMSprop

1. กำหนด $\boldsymbol{\theta}_{\mathbf{0}} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$
2. กำหนด `epoch = 200`
3. กำหนดค่า $\alpha = 0.0375,\ \beta = 0.9$

RMSprop

4. Update θ : epoch = 1

4.1 $\mathbf{g_1} = \begin{bmatrix} -10.04 \\ -495.21 \end{bmatrix}$

4.2 $\mathbf{G_1} = \begin{bmatrix} 0.9 \times 0 + 0.1 \times (-10.04)^2 \\ 0.9 \times 0 + 0.1 \times (-495.21)^2 \end{bmatrix} = \begin{bmatrix} 10.08 \\ 24523.51 \end{bmatrix}$

4.3 $\mathbf{S_1} = \begin{bmatrix} \frac{0.5}{\sqrt{10.08+(1\times10^{-8})}} \\ \frac{0.5}{\sqrt{24523.51+(1\times10^{-8})}} \end{bmatrix} = \begin{bmatrix} \frac{0.0375}{3.18} \\ \frac{0.0375}{156.6} \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.0002 \end{bmatrix}$

4.4 $\mathbf{\theta_1} = \begin{bmatrix} -5 - (0.01 \times -10.04) \\ -5 - (0.0002 \times -495.21) \end{bmatrix} = \begin{bmatrix} -4.88 \\ -4.88 \end{bmatrix}$

RMSprop

4. Update θ : $\boxed{\text{epoch} = 2}$

4.1 $\mathbf{g_2} = \begin{bmatrix} -9.87 \\ -493.7 \end{bmatrix}$

4.2 $\mathbf{G_2} = \begin{bmatrix} 0.9 \times 10.08 + 0.1 \times (-9.87)^2 \\ 0.9 \times 24523.51 + 0.1 \times (-493.7)^2 \end{bmatrix} = \begin{bmatrix} 18.81 \\ 46444.86 \end{bmatrix}$

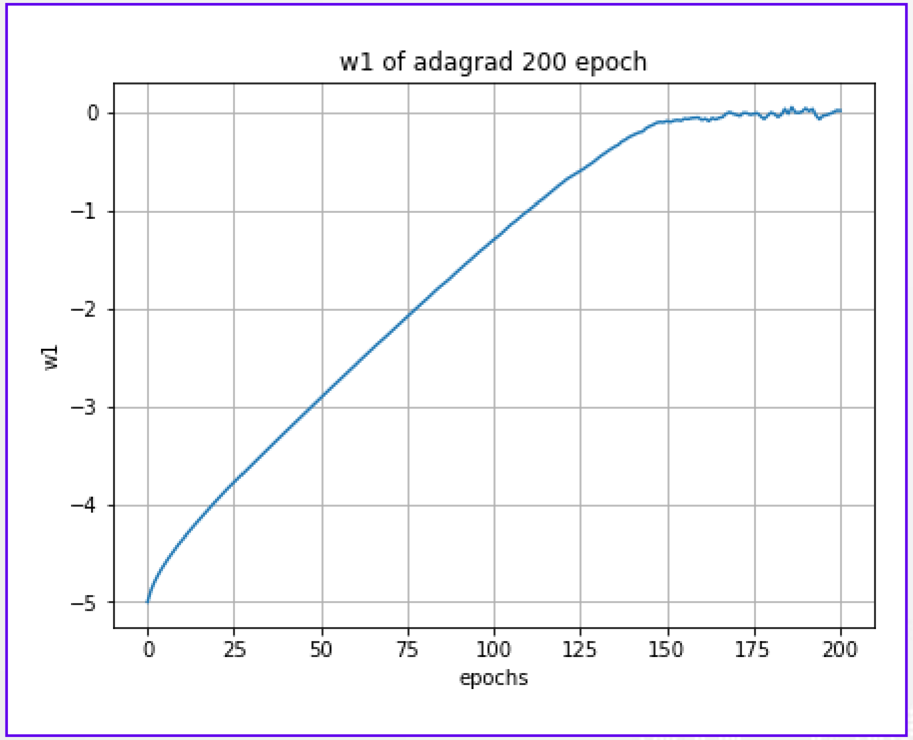
4.3 $\mathbf{S_2} = \begin{bmatrix} \frac{0.5}{\sqrt{18.81+(1\times10^{-8})}} \\ \frac{0.5}{\sqrt{46444.86+(1\times10^{-8})}} \end{bmatrix} = \begin{bmatrix} \frac{0.0375}{4.34} \\ \frac{0.0375}{215.51} \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.0002 \end{bmatrix}$

4.4 $\mathbf{\theta_2} = \begin{bmatrix} -4.88 - (0.01 \times -9.87) \\ -4.88 - (0.0002 \times -493.7) \end{bmatrix} = \begin{bmatrix} -4.8 \\ -4.8 \end{bmatrix}$

RMSprop

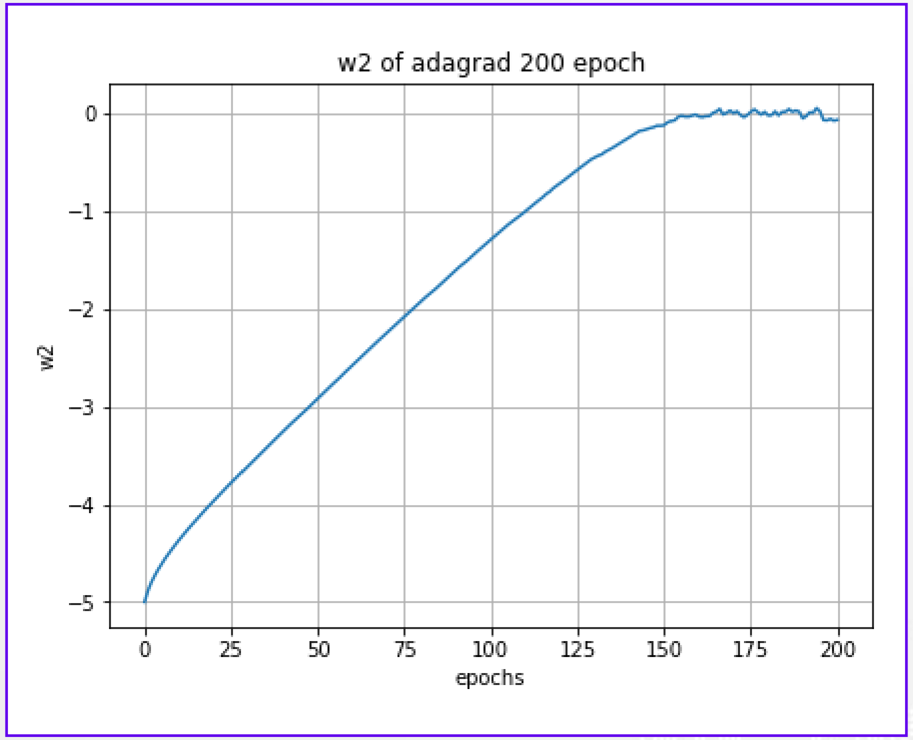
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.88	-4.88
3	-4.8	-4.8
4	-4.73	-4.73
5	-4.66	-4.66
6	-4.61	-4.61
⋮	⋮	⋮
198	0	-0.06
199	-0.02	-0.08
200	-0.02	-0.07

RMSprop



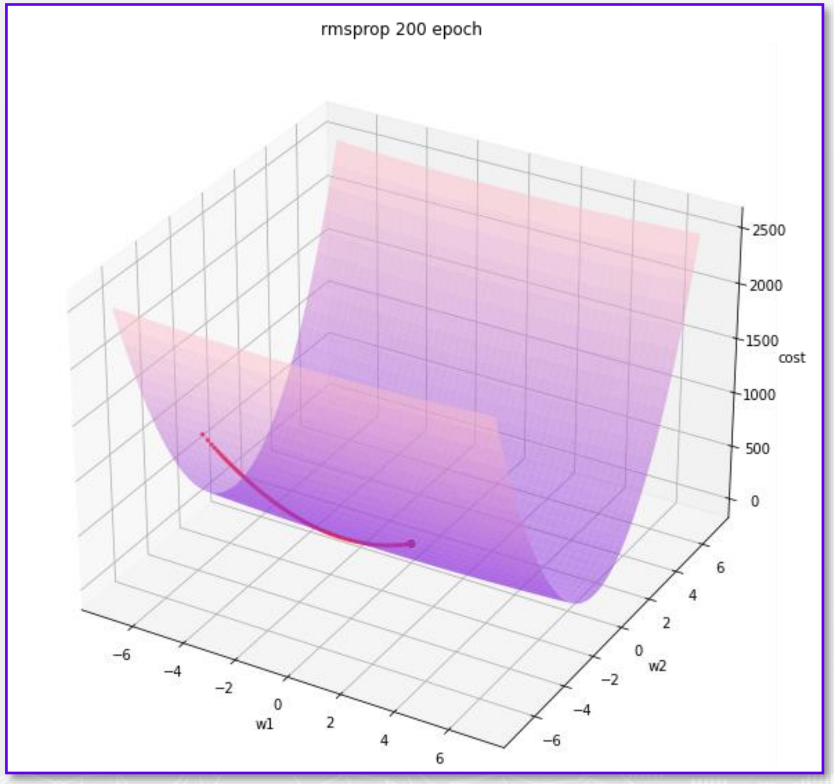
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.88	-4.88
3	-4.8	-4.8
4	-4.73	-4.73
5	-4.66	-4.66
6	-4.61	-4.61
⋮	⋮	⋮
198	0	-0.06
199	-0.02	-0.08
200	-0.02	-0.07

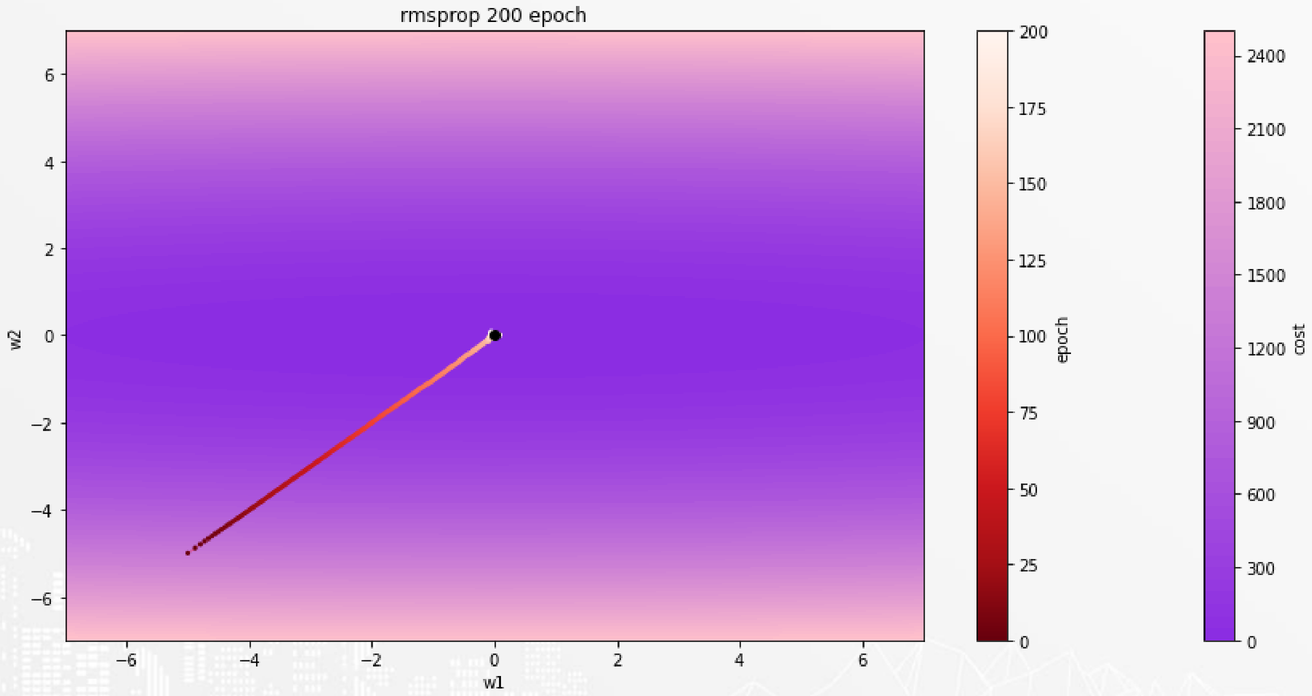
RMSprop



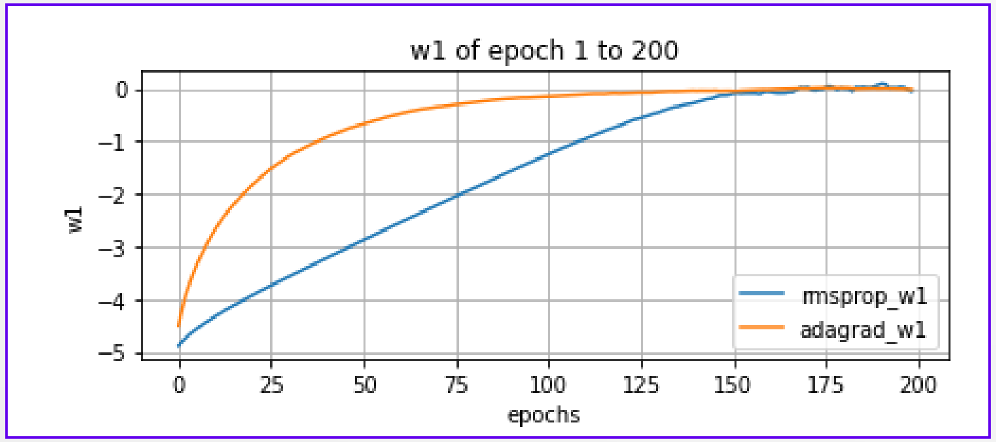
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.88	-4.88
3	-4.8	-4.8
4	-4.73	-4.73
5	-4.66	-4.66
6	-4.61	-4.61
⋮	⋮	⋮
198	0	-0.06
199	-0.02	-0.08
200	-0.02	-0.07

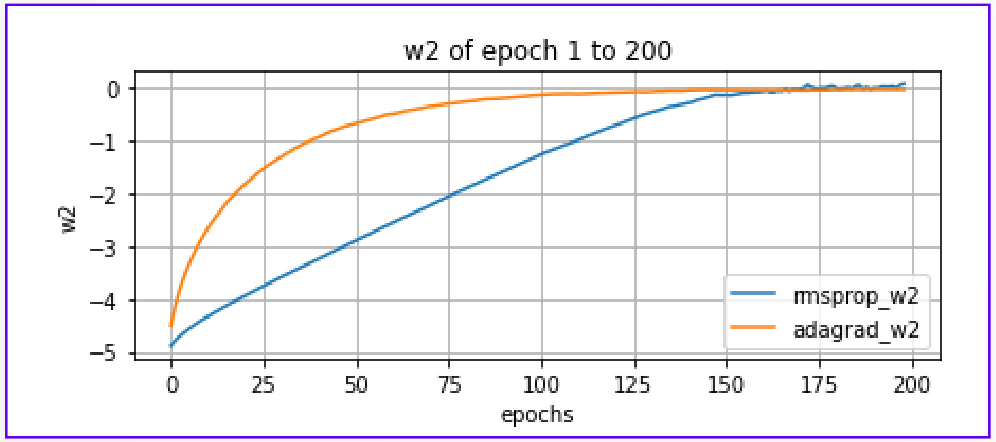
RMSprop

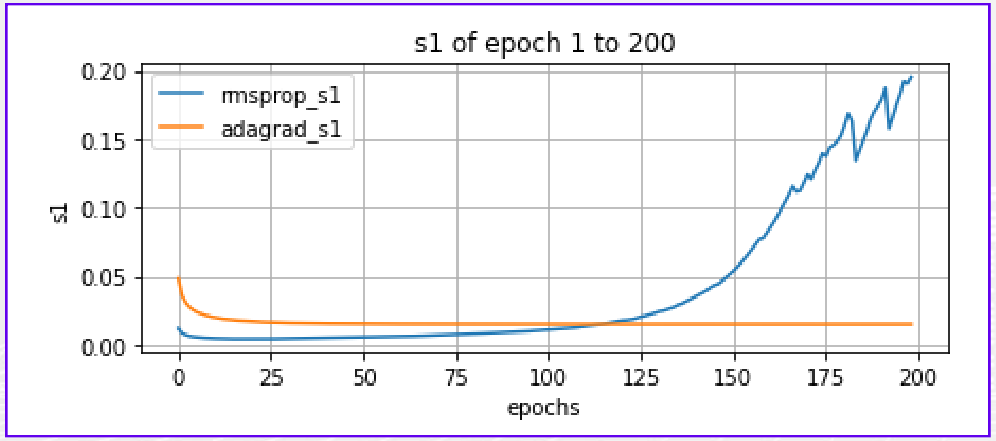

rmsprop 200 epoch
cost
w1
w2

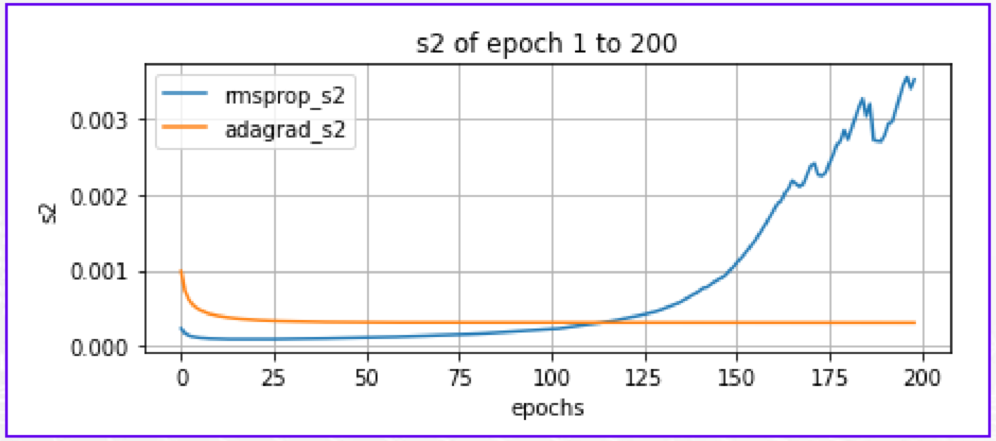

rmsprop 200 epoch
w1
w2
epoch
cost

RMSprop


w1 of epoch 1 to 200
w1
epochs
rmsprop_w1
adagrad_w1


w2 of epoch 1 to 200
w2
epochs
rmsprop_w2
adagrad_w2


s1 of epoch 1 to 200
s1
epochs
rmsprop_s1
adagrad_s1


s2 of epoch 1 to 200
s2
epochs
rmsprop_s2
adagrad_s2

Adaptive Learning Rate Base Algorithm

- ✓ Adagrad
- ✓ RMSprop

Optimization Algorithm

- ☑ **Momentum Base**
- ☑ **Adaptive Learning Rate Base Algorithm**
- ☐ Adam
- ☐ Conclusion

Adam

Adam (Adaptive moment estimation) คือ algorithm ที่ใช้แนวคิดของ RMSprop และ momentum base มาใช้ร่วมกัน

Adam

ซึ่งแนวคิดที่นำมาใช้คือ

- การสะสมความเร็วในการเคลื่อนที่ในแต่ละมิติ **(momentum base)**
- การลดขนาด learning rate ในแต่ละมิติด้วย การสะสม gradient โดยใช้ EMA **(RMSprop)**

Adam

Adam สามารถเขียนให้อยู่ในรูปดังนี้

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t + \varepsilon}} \odot m_t$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t \odot g_t)$$

$$g_t = \nabla Cost_{t-1}$$

โดย

- $\varepsilon = 1 \times 10^{-8}$ (ป้องกันการหารด้วย 0)
- $m_0 = 0, v_0 = 0$

Adam

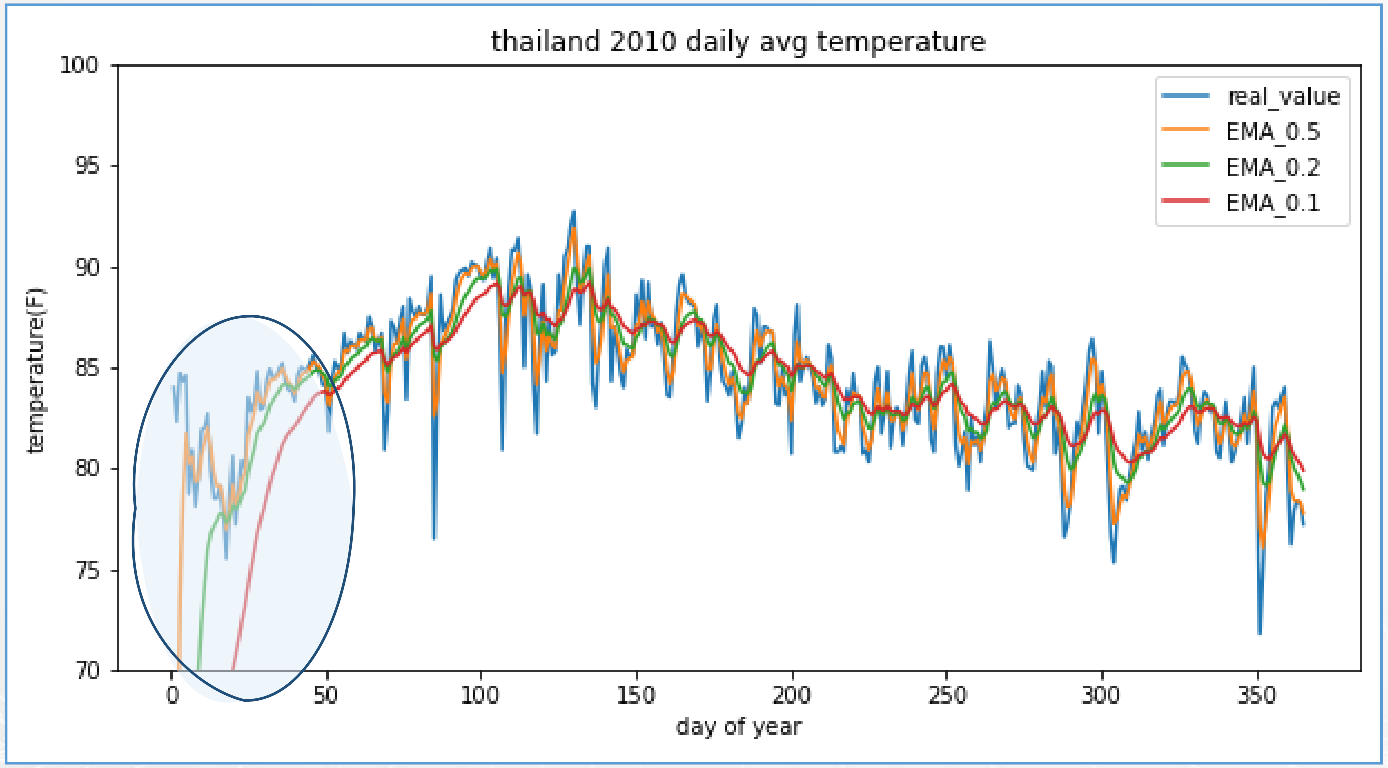
แต่ EMA ยังมีข้อเสียคือ...

Adam

ในช่วงต้น ๆ ค่าของ EMA จะค่อนข้างแปรปรวน
เนื่องจากเรากำหนดให้ initial = 0 เลย

Adam

Exponential Moving Average



Adam

ดังนั้น เราจะแก้ปัญหาโดยใช้

“ Bias Correction ”

เพื่อให้ค่า EMA ตอนเริ่มต้นไม่เป็น 0

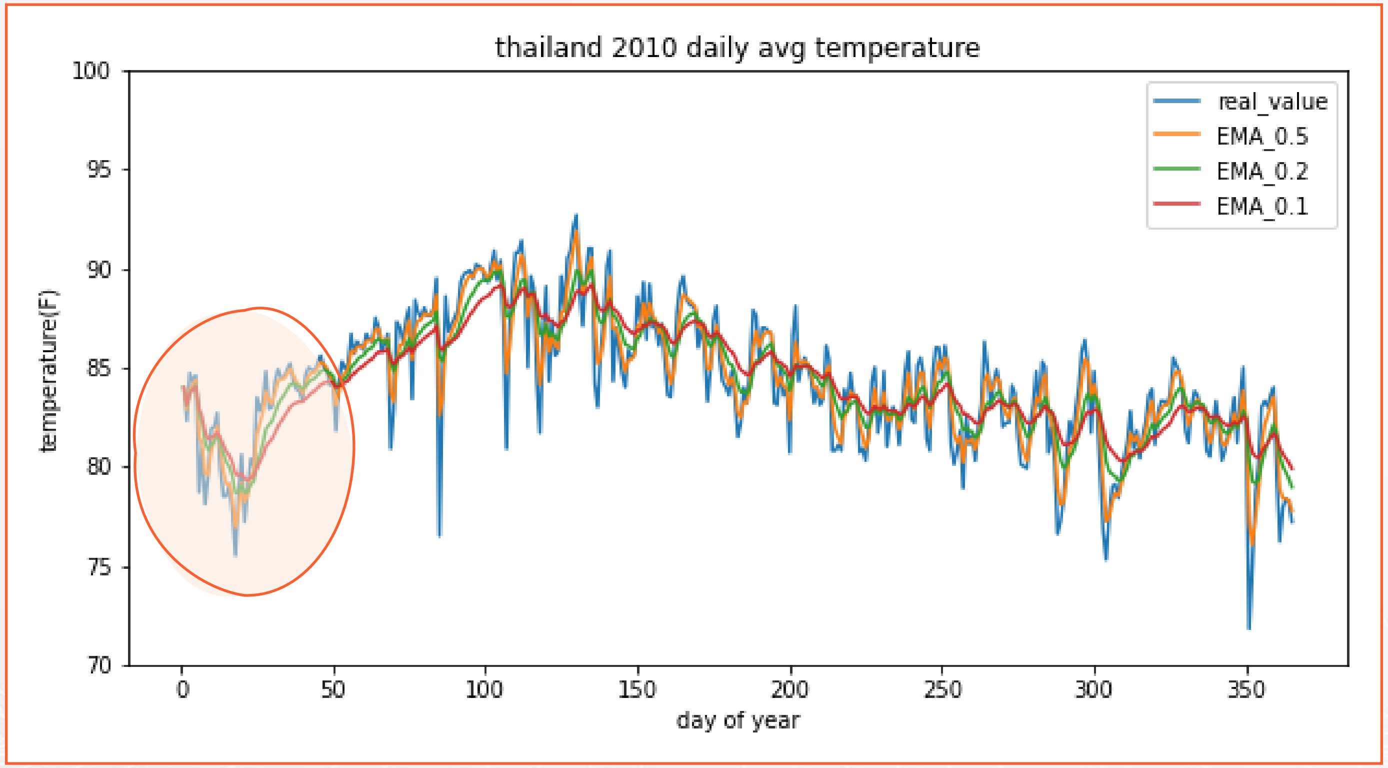
Adam

Bias Correction of Exponential Moving Average

$$EMA_t^{corr} = \frac{EMA_t}{1-(1-\beta)^t}$$

Adam

Bias Correction of Exponential Moving Average



Adam

ดังนั้น Adam ที่ใช้ bias correction สามารถเขียนให้อยู่ในรูปดังต่อไปนี้

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t + \varepsilon}} \odot m_t$$
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t \odot g_t)$$
$$g_t = \nabla Cost_{t-1}$$

Adam with EMA

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t + \varepsilon}} \odot \hat{m}_t$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t \odot g_t)$$
$$g_t = \nabla Cost_{t-1}$$

Adam with Bias correction

Adam

- **ขั้นตอนการคำนวณ**
  1. กำหนดค่า θ เริ่มต้น
  2. กำหนดจำนวนรอบที่จะ update θ (epoch)
  3. กำหนดค่า α, β_1, β_2
  4. For loop เพื่อ Update θ
     - 4.1 คำนวณ $\mathbf{g}$
     - 4.2 คำนวณ $\mathbf{m}, \widehat{\mathbf{m}}$
     - 4.3 คำนวณ $\mathbf{v}, \hat{\mathbf{v}}$
     - 4.4 คำนวณ $\boldsymbol{\theta}$

Adam

1. กำหนด $\boldsymbol{\theta}_{\mathbf{0}} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$

2. กำหนด epoch = 200

3. กำหนดค่า $\alpha = 0.075,\ \beta_1 = 0.9,\ \beta_2 = 0.999$

Adam

4. Update θ : epoch = 1

4.1 $\mathbf{g_1} = \begin{bmatrix} -10.04 \\ -495.21 \end{bmatrix}$

4.2 $\mathbf{m_1} = \begin{bmatrix} 0.9 \times 0 + 0.1 \times -10.04 \\ 0.9 \times 0 + 0.1 \times -495.21 \end{bmatrix} = \begin{bmatrix} -1 \\ -49.52 \end{bmatrix}$

$$\hat{\mathbf{m}}_1 = \begin{bmatrix} \frac{-1}{1-(0.9)^1} \\ \frac{-49.52}{1-(0.9)^1} \end{bmatrix} = \begin{bmatrix} -10.04 \\ -495.21 \end{bmatrix}$$

Adam

4. Update θ : epoch = 1

$$4.3\ \mathbf{v_1} = \begin{bmatrix} 0.999 \times 0 + 0.001 \times (-10.04)^2 \\ 0.999 \times 0 + 0.001 \times (-495.21)^2 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 245.23 \end{bmatrix}$$

$$\hat{\mathbf{v}}_1 = \begin{bmatrix} \frac{0.1}{1-(0.999)^1} \\ \frac{245.23}{1-(0.999)^1} \end{bmatrix} = \begin{bmatrix} -10.04 \\ -495.21 \end{bmatrix}$$

$$4.4\ \boldsymbol{\theta}_1 = \begin{bmatrix} -5 - (0.01 \times -10.04) \\ -5 - (0.0002 \times -495.21) \end{bmatrix} = \begin{bmatrix} -4.93 \\ -4.93 \end{bmatrix}$$

Adam

4. Update θ : epoch = 2

4.1 $\mathbf{g_2} = \begin{bmatrix} -9.95 \\ -498.06 \end{bmatrix}$

4.2 $\mathbf{m_2} = \begin{bmatrix} 0.9 \times -1.00 + 0.1 \times -9.95 \\ 0.9 \times -49.52 + 0.1 \times -498.06 \end{bmatrix} = \begin{bmatrix} -1.9 \\ -94.37 \end{bmatrix}$

$\hat{\mathbf{m}}_2 = \begin{bmatrix} \frac{-1.9}{1-(0.9)^2} \\ \frac{-94.37}{1-(0.9)^2} \end{bmatrix} = \begin{bmatrix} -10 \\ -496.71 \end{bmatrix}$

Adam

4. Update θ : $\boxed{\text{epoch} = 2}$

$$4.3\ \mathbf{v_2} = \begin{bmatrix} 0.999 \times 0 + 0.001 \times (-9.95)^2 \\ 0.999 \times 0 + 0.001 \times (-498.06)^2 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 493.05 \end{bmatrix}$$

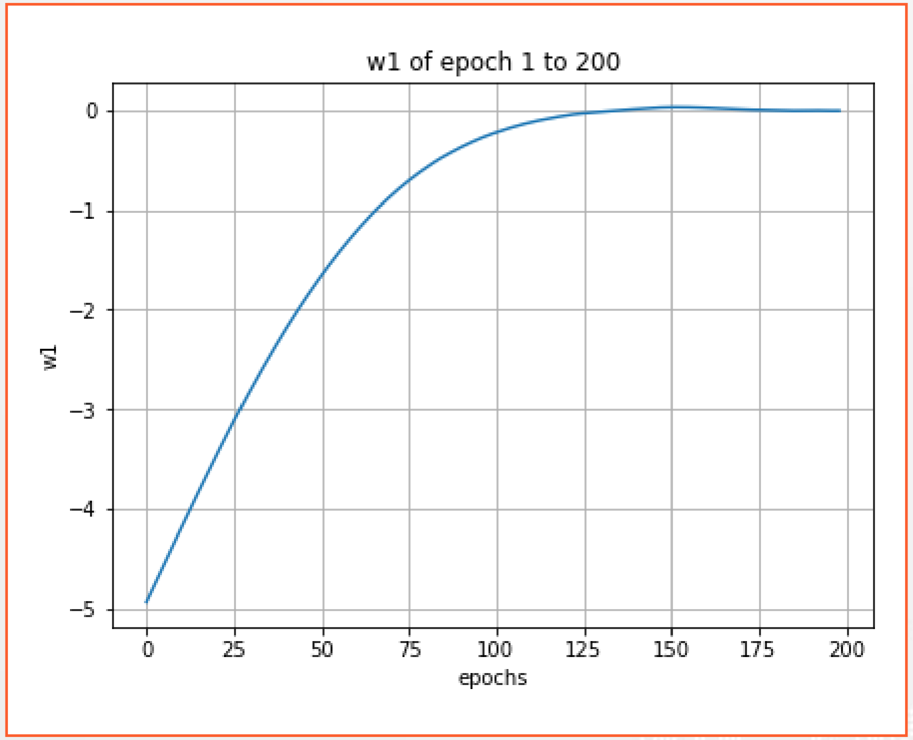
$$\hat{\mathbf{v}}_2 = \begin{bmatrix} \frac{0.2}{1-(0.999)^1} \\ \frac{493.05}{1-(0.999)^1} \end{bmatrix} = \begin{bmatrix} -10 \\ -496.71 \end{bmatrix}$$

$$4.4\ \boldsymbol{\theta}_2 = \begin{bmatrix} -4.93 - (0.01 \times -10) \\ -4.93 - (0.0002 \times -496.71) \end{bmatrix} = \begin{bmatrix} -4.85 \\ -4.85 \end{bmatrix}$$

Adam

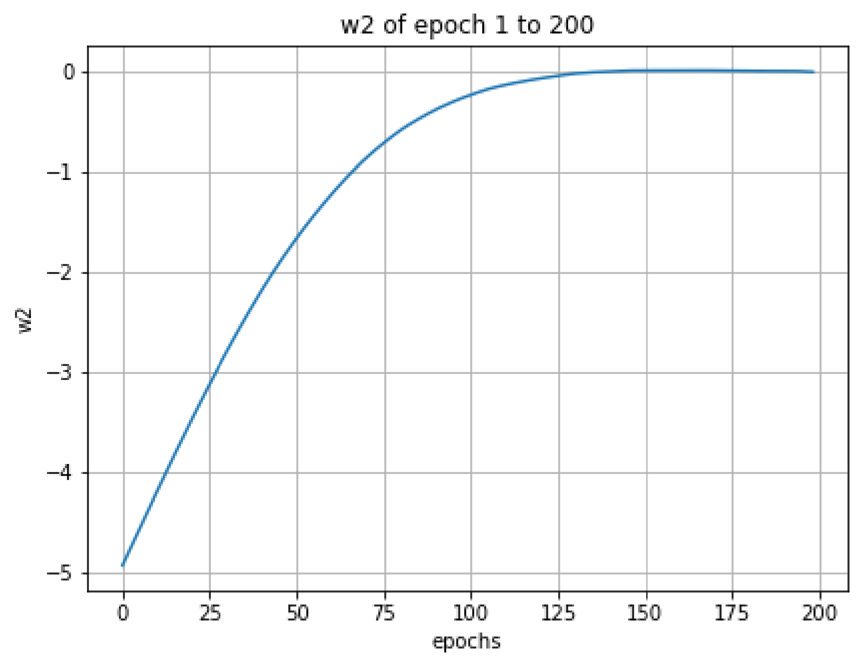
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.93	-4.93
3	-4.85	-4.85
4	-4.78	-4.78
5	-4.7	-4.7
6	-4.63	-4.63
$\vdots$	$\vdots$	$\vdots$
198	-0.01	-0.001
199	-0.01	-0.003
200	-0.009	-0.005

Adam



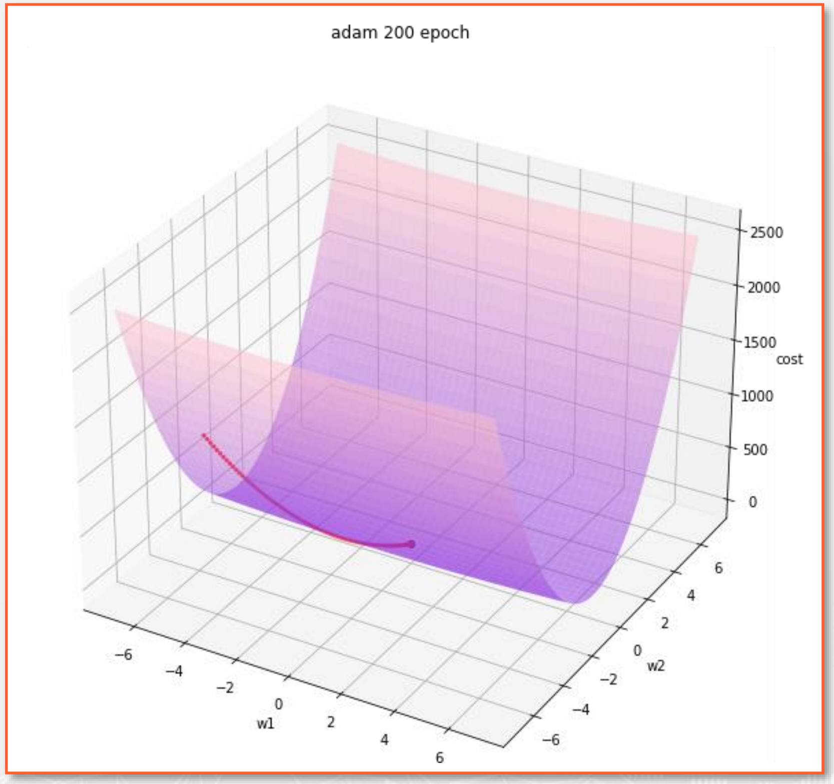
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.93	-4.93
3	-4.85	-4.85
4	-4.78	-4.78
5	-4.7	-4.7
6	-4.63	-4.63
⋮	⋮	⋮
198	-0.01	-0.001
199	-0.01	-0.003
200	-0.009	-0.005

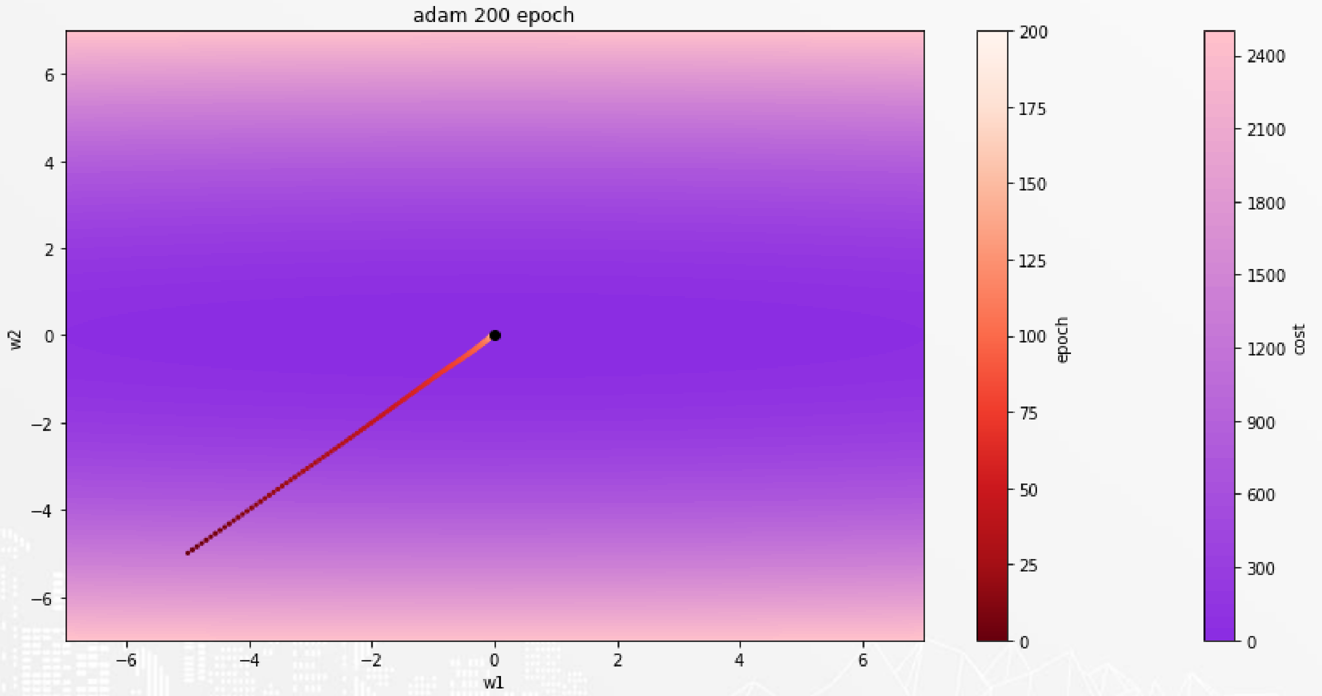
Adam



epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.93	-4.93
3	-4.85	-4.85
4	-4.78	-4.78
5	-4.7	-4.7
6	-4.63	-4.63
⋮	⋮	⋮
198	-0.01	-0.001
199	-0.01	-0.003
200	-0.009	-0.005

Adam


adam 200 epoch
2500
2000
1500
1000
500
0
cost
−6
−4
−2
0
2
4
6
w1
6
4
2
0
−2
−4
−6
w2

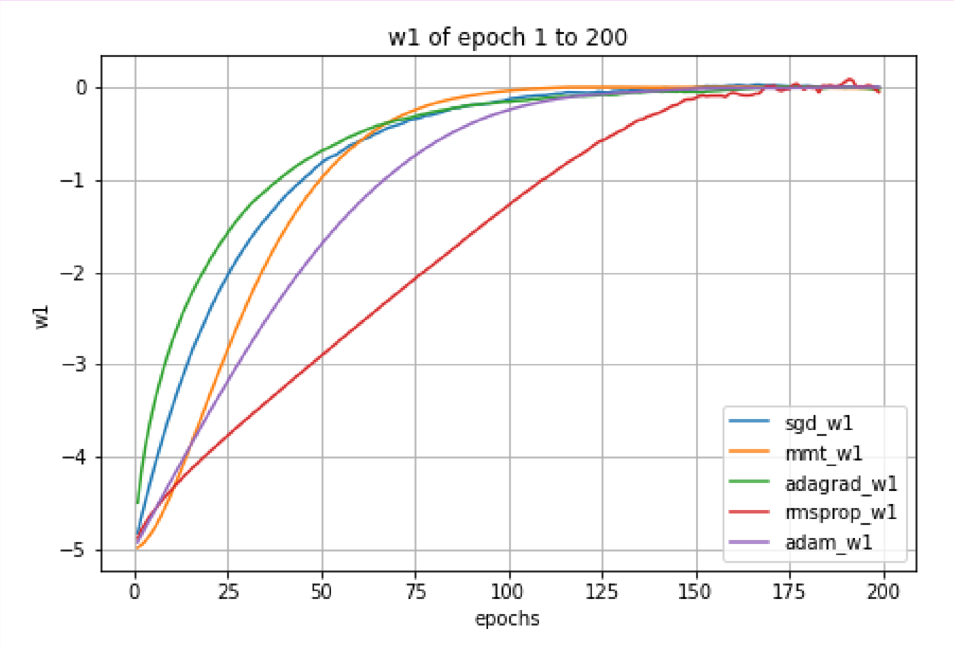

adam 200 epoch
6
4
2
0
−2
−4
−6
w2
−6
−4
−2
0
2
4
6
w1
200
175
150
125
100
75
50
25
0
epoch
2400
2100
1800
1500
1200
900
600
300
0
cost

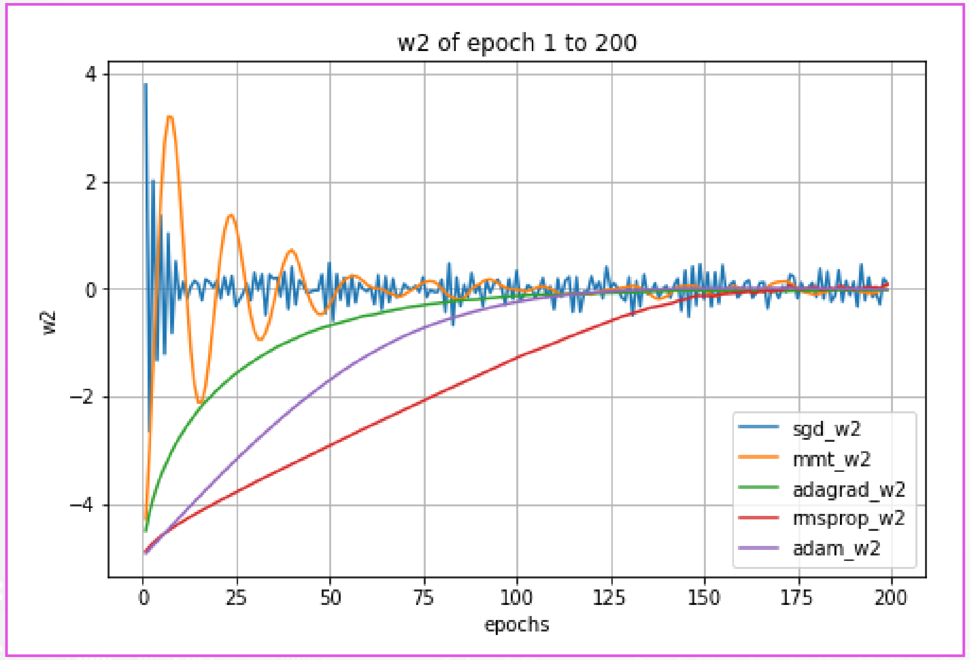
Optimization Algorithm

- ☑ Momentum Base
- ☑ Adaptive Learning Rate Base Algorithm
- ☑ Adam
- ☐ Conclusion

Conclusion

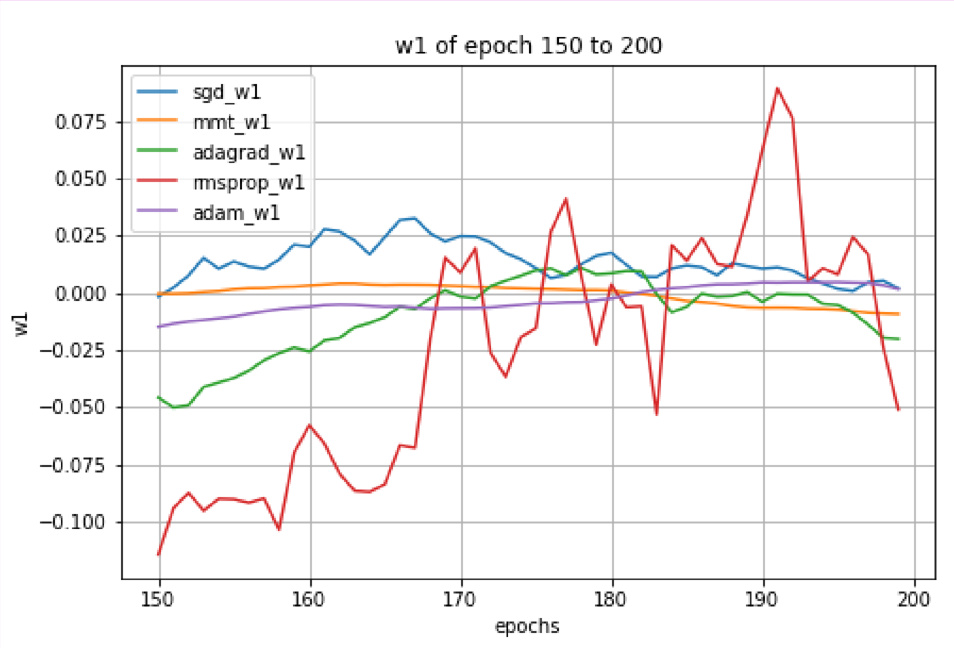
visualization of algorithms

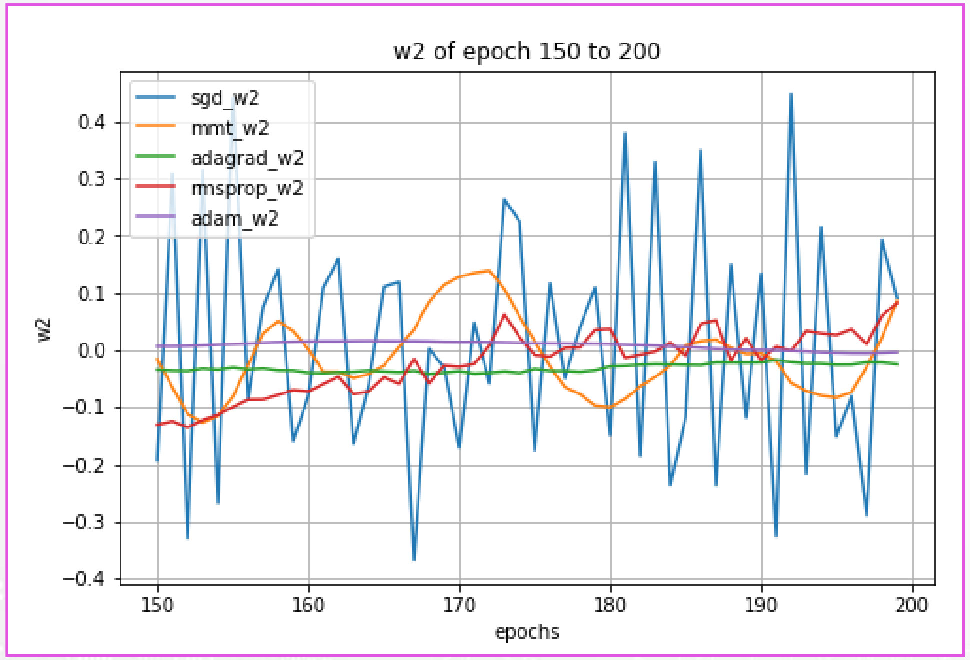




Conclusion

visualization of algorithms





Conclusion

ควรใช้ optimizer ตัวไหนดี?

Adam เป็น algorithm ที่ดีที่สุด เนื่องจากการทำ bias correction ที่ปรับค่า exponential moving average ให้มีความถูกต้องมากยิ่งขึ้น

(Kingma, D. P., & Ba, J. L. (2015)*)

* Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, 1–13

Optimization Algorithm

- ☑ Momentum Base
- ☑ Adaptive Learning Rate Base Algorithm
- ☑ Adam
- ☑ Conclusion

Optimizer

Gradient Descent Variants

Optimization Algorithm

Code

Code

scikit learn

SKlearn

K

Keras + Tensorflow

Pytorch

Code

Optimizer/sklearn

- Adam Regression (sklearn).ipynb
- Adam Binary Classification (sklearn).ipynb
- Adam Multi-Class Classification (sklearn).ipynb

Code

Optimizer/keras

- Adam Regression (keras).ipynb
- Adam Binary Classification (keras).ipynb
- Adam Multi-Class Classification (keras).ipynb

Code

- Optimizer/pytorch
  - Adam Regression (pytorch).ipynb
  - Adam Binary Classification (pytorch).ipynb
  - Adam Multi-Class Classification (pytorch).ipynb

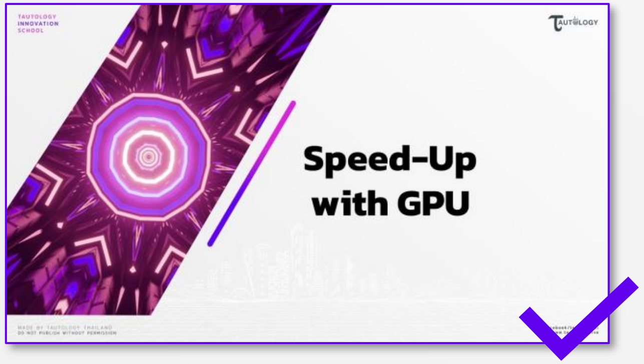
Optimizer

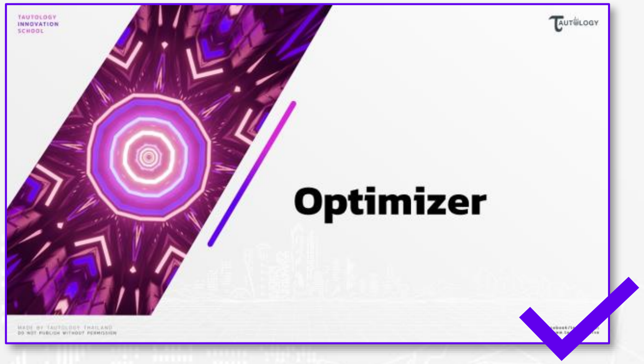
Gradient Descent Variants

Optimization Algorithm

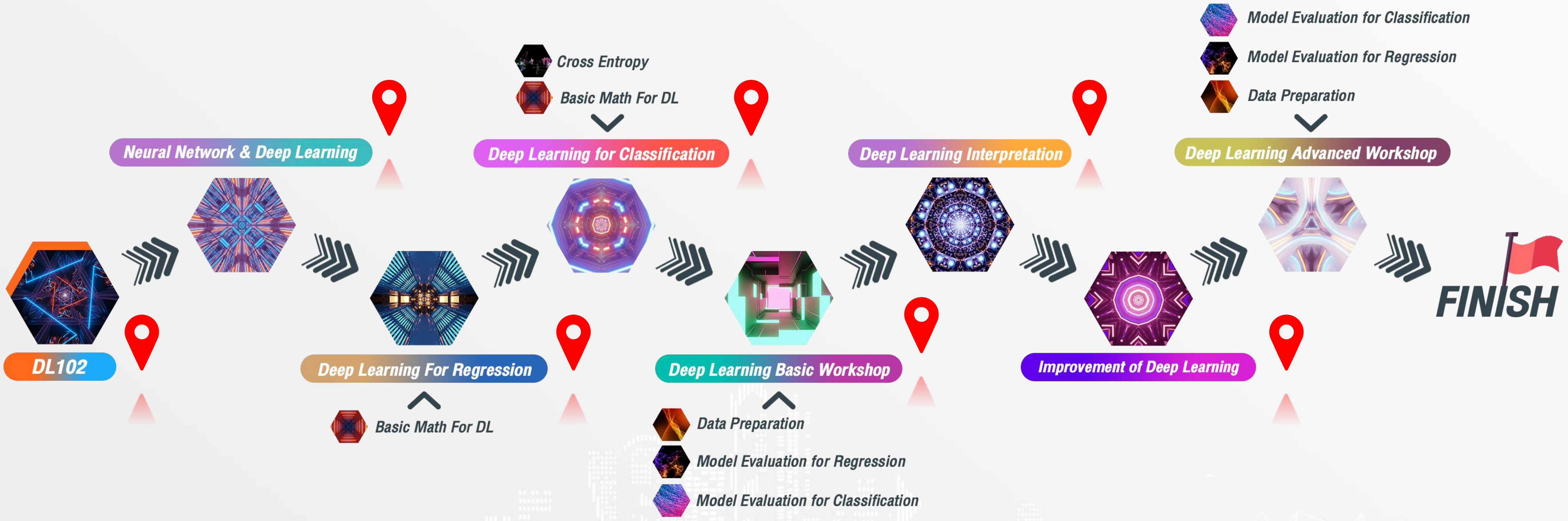
Code

Improvement of Deep Learning





DL103 : Deep Learning



TAUTOLOGY
INNOVATION
SCHOOL

TAUTOLOGY

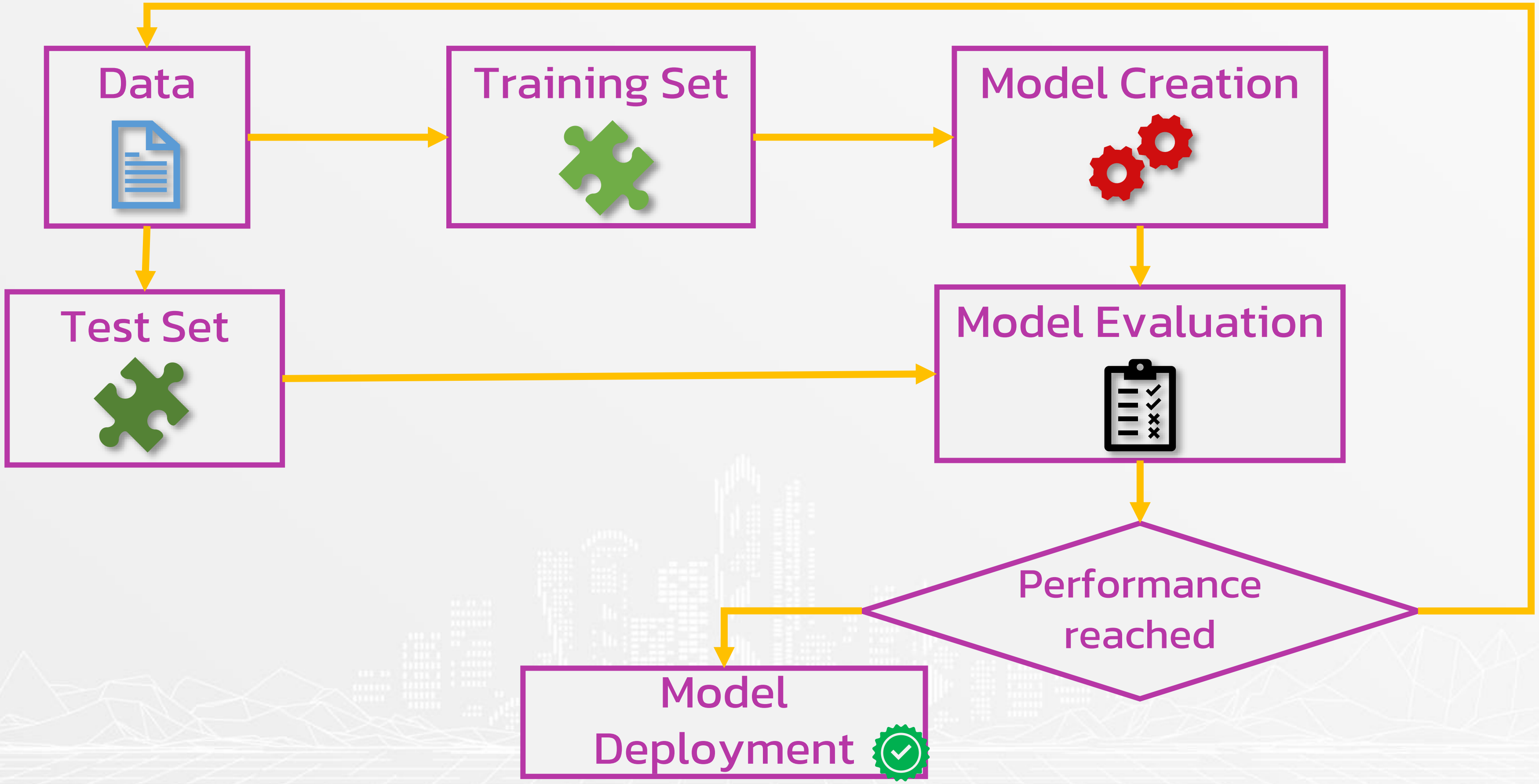
DEEP LEARNING ADVANCED WORKSHOP

DEEP LEARNING
ADVANCED WORKSHOP

BY TAUTOLOGY

facebook/tautologyai
www.tautology.live

Supervised Learning Workflow



Workshop Overview

Regression

Problem Type

Classification

Workshop Overview

Model Creation Library

SKlearn

Keras + TF

Pytorch

Supported Library

GPU

SKlearn

Keras + Tensorflow

Pytorch

Supported Library

Balance Class weight

SKlearn

Keras + Tensorflow

Pytorch

Supported Library

L2 Regularization

SKlearn

Keras + Tensorflow

Pytorch

Supported Library

L1 Regularization

SKlearn

Keras + Tensorflow

Pytorch

Supported Library

Dropout Regularization

SKlearn

Keras + Tensorflow

Pytorch

Supported Library

Mini Batch

SKlearn

Keras + Tensorflow

Pytorch

Supported Library

Adam

SKlearn

Keras + Tensorflow

Pytorch

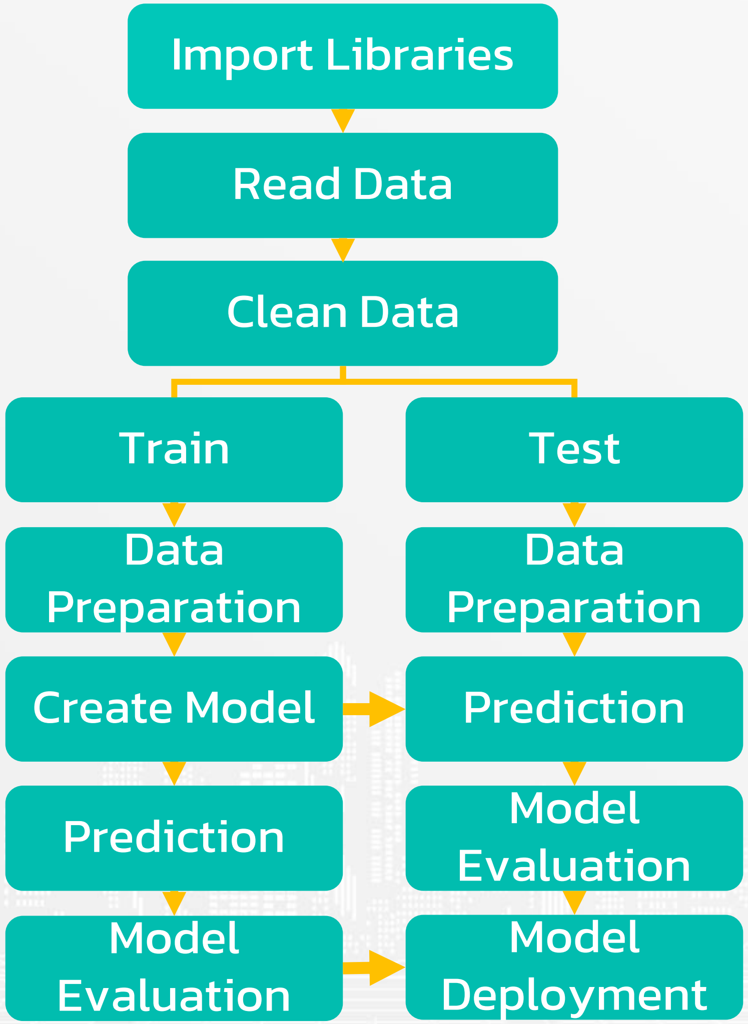
Supported Library

	SKlearn	Keras + Tensorflow	Pytorch
GPU	✗	✓	✓
Balanced Class weight	✓	✓	✓
L2 Regularization	✓	✓	✓
L1 Regularization	✗	✓	✓

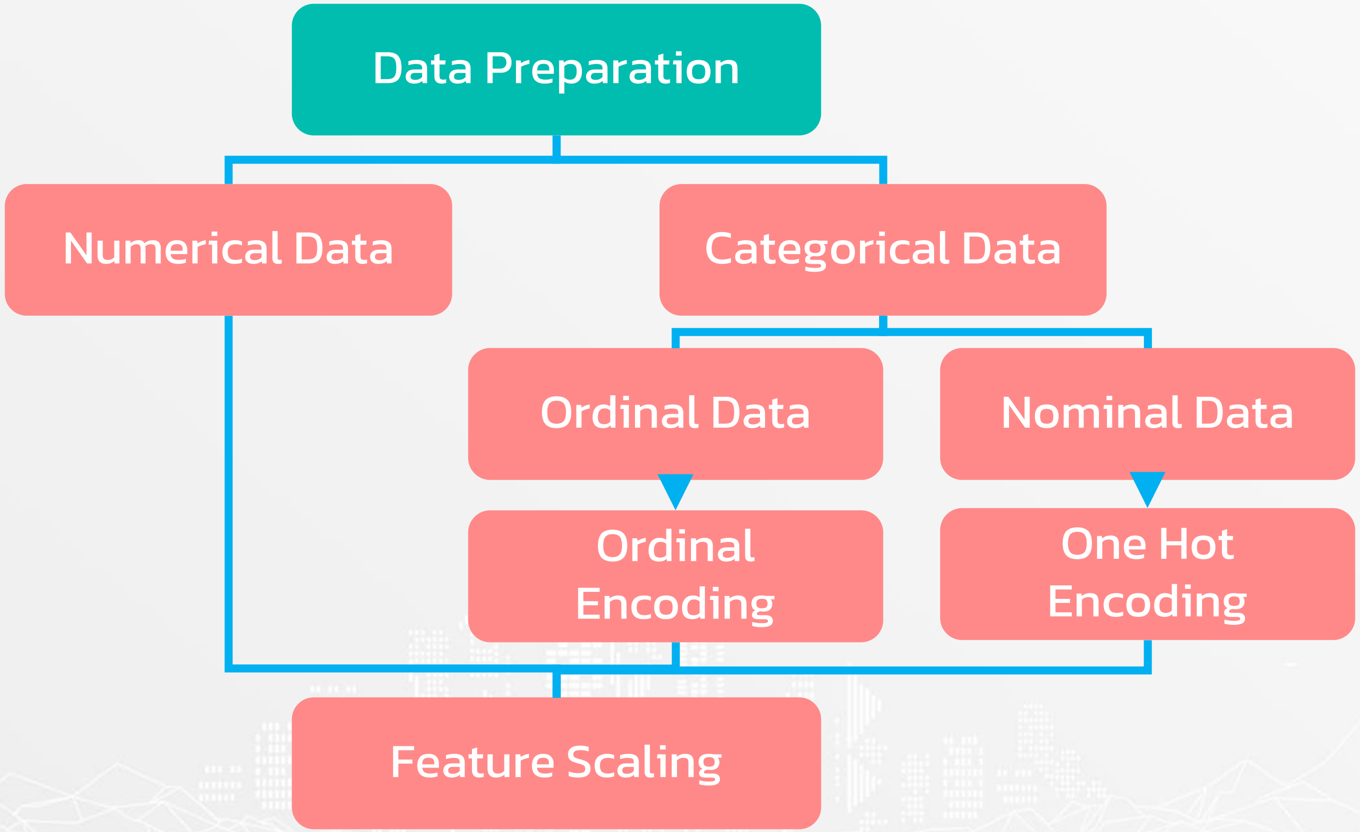
Supported Library

	SKlearn	Keras + Tensorflow	Pytorch
Dropout Regularization	✗	✓	✓
Mini Batch	✓	✓	✓
Adam	✓	✓	✓

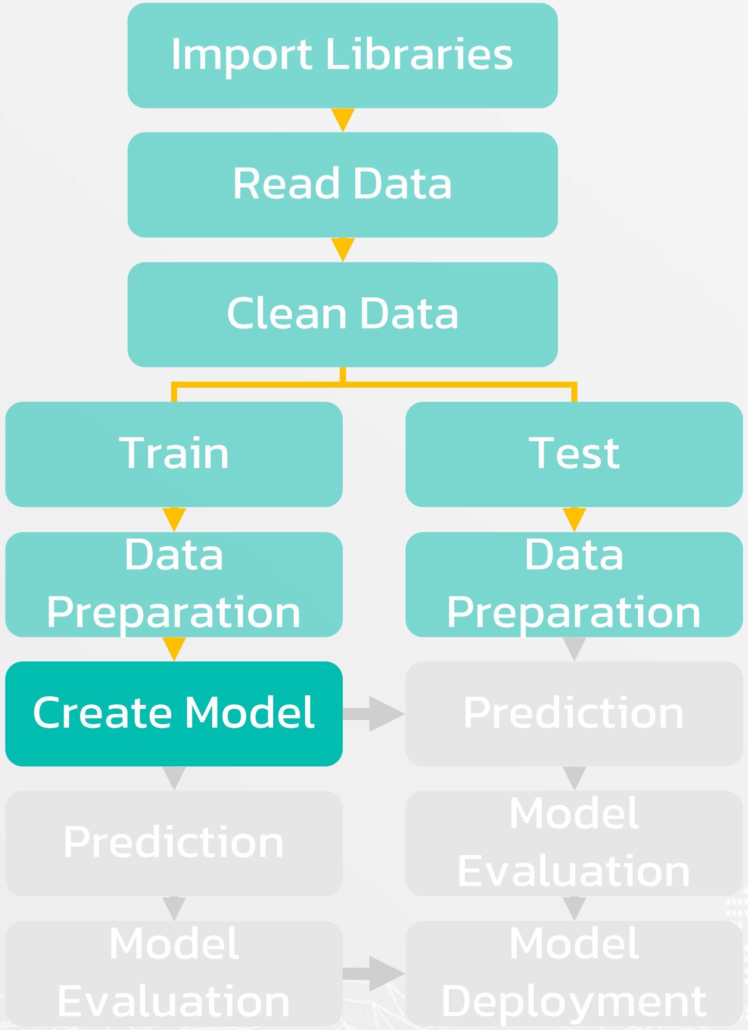
Code Pipeline



Data Preparation



Create Model



Code – Create Model

Regression

```python
reg = MLPRegressor(
    hidden_layer_sizes=(10, 10),
    activation='relu',
    learning_rate_init=0.01,
    max_iter=1000,
    batch_size=64,
    alpha=0.1,
    solver='adam',
    beta_1=0.9,
    beta_2=0.999,
)
```

Classification

```python
clf = MLPClassifier(
    hidden_layer_sizes=(10, 10),
    activation='relu',
    learning_rate_init=1,
    max_iter=1000,
    batch_size=64,
    alpha=0.1,
    solver='adam',
    beta_1=0.9,
    beta_2=0.999
)
```

Code – Create Model

Regression

```
reg = KerasMLPRegressor(
    input_dim=X_train_scaled.shape[1],
    hidden_layer_sizes=(10, 10),
    activation_function='relu',
    learning_rate_init=0.01,
    epochs=1000,
    validation_split=0.2,
    use_gpu=True,
    l1_lambda=0.1,
    l2_lambda=0.1,
    dropout_rate=[0.25, 0.25],
    solver='adam',
    batch_size=64
)
```

Classification

```
classes = np.unique(y_train)
```

```
clf = KerasMLPClassifier(
    input_dim=X_train_scaled.shape[1],
    hidden_layer_sizes=(10, 10),
    activation_function='relu',
    classes=classes,
    learning_rate_init=1,
    epochs=1000,
    validation_split=0.2,
    use_gpu=True,
    class_weight='balanced',
    l1_lambda=0.1,
    l2_lambda=0.1,
    dropout_rate=[0.25, 0.25],
    solver='adam',
    batch_size=64
)
```

Code – Create Model

Regression

```python
reg = PytorchMLPRegressor(
    input_dim=X_train_scaled.shape[1],
    hidden_layer_sizes=(10, 10),
    activation_function='relu',
    learning_rate_init=0.01,
    epochs=1000,
    validation_split=0.2,
    use_gpu=True,
    l1_lambda=0.1,
    l2_lambda=0.1,
    dropout_rate=[0.25, 0.25],
    solver='adam',
    batch_size=64
)
```

Classification

```python
classes = np.unique(y_train)
```

```python
clf = PytorchMLPClassifier(
    input_dim=X_train_scaled.shape[1],
    hidden_layer_sizes=(10, 10),
    activation_function='relu',
    classes=classes,
    learning_rate_init=1,
    epochs=1000,
    validation_split=0.2,
    use_gpu=True,
    class_weight='balanced',
    l1_lambda=0.1,
    l2_lambda=0.1,
    dropout_rate=[0.25, 0.25],
    solver='adam',
    batch_size=64
)
```

Ai in Real Estate Business

Abstract

สร้าง model เพื่อประเมินราคาของบ้าน โดย feature ที่นำมาใช้ คือ ข้อมูลทั่วไปของบ้าน และข้อมูลสภาพแวดล้อมภายนอกของบ้าน เช่น

- ขนาดพื้นที่ใช้สอยภายในบ้าน
- ขนาดของที่ดิน
- ขนาดของสระว่ายน้ำ

Why this project important?

- สามารถประเมินราคาบ้านได้อย่างสมเหตุสมผลที่สุด
- สามารถนำความรู้ที่ได้จากการสร้าง model ไปประยุกต์ใช้กับธุรกิจประเภทอื่น ๆ ที่มีลักษณะคล้ายกัน

Who this project is for?

- นักธุรกิจอสังหาริมทรัพย์
- นักประเมินราคาสินทรัพย์
- นักวิเคราะห์ข้อมูล

House Price Dataset

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data

House Price Dataset

Feature

- MSSubClass – รูปแบบของสิ่งปลูกสร้าง
- MSZoning – โซนที่ตั้งสิ่งปลูกสร้าง
- LotFrontage – ระยะห่างจากถนน
- LotArea – ขนาดของที่ดิน
- Street – รูปแบบของการปูถนนหน้าสิ่งปลูกสร้าง (Gravel, Pave)

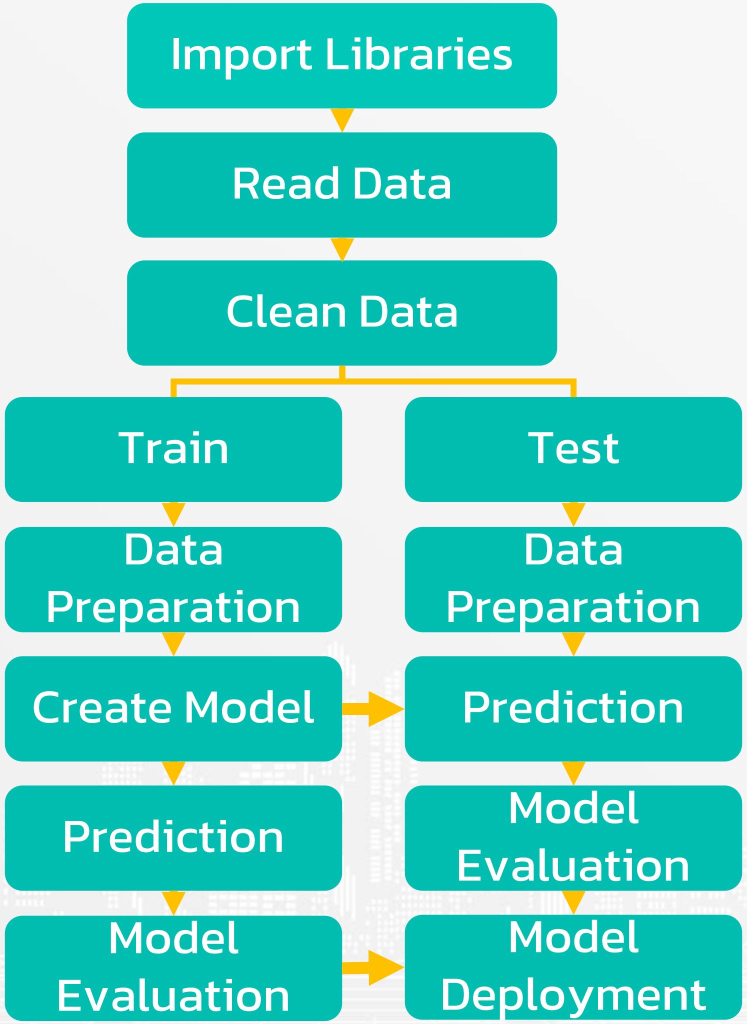
⋮

- LotShape – รูปทรงของสิ่งปลูกสร้าง
- PoolArea – พื้นที่สระว่ายน้ำ
- SaleType – รูปแบบการขาย

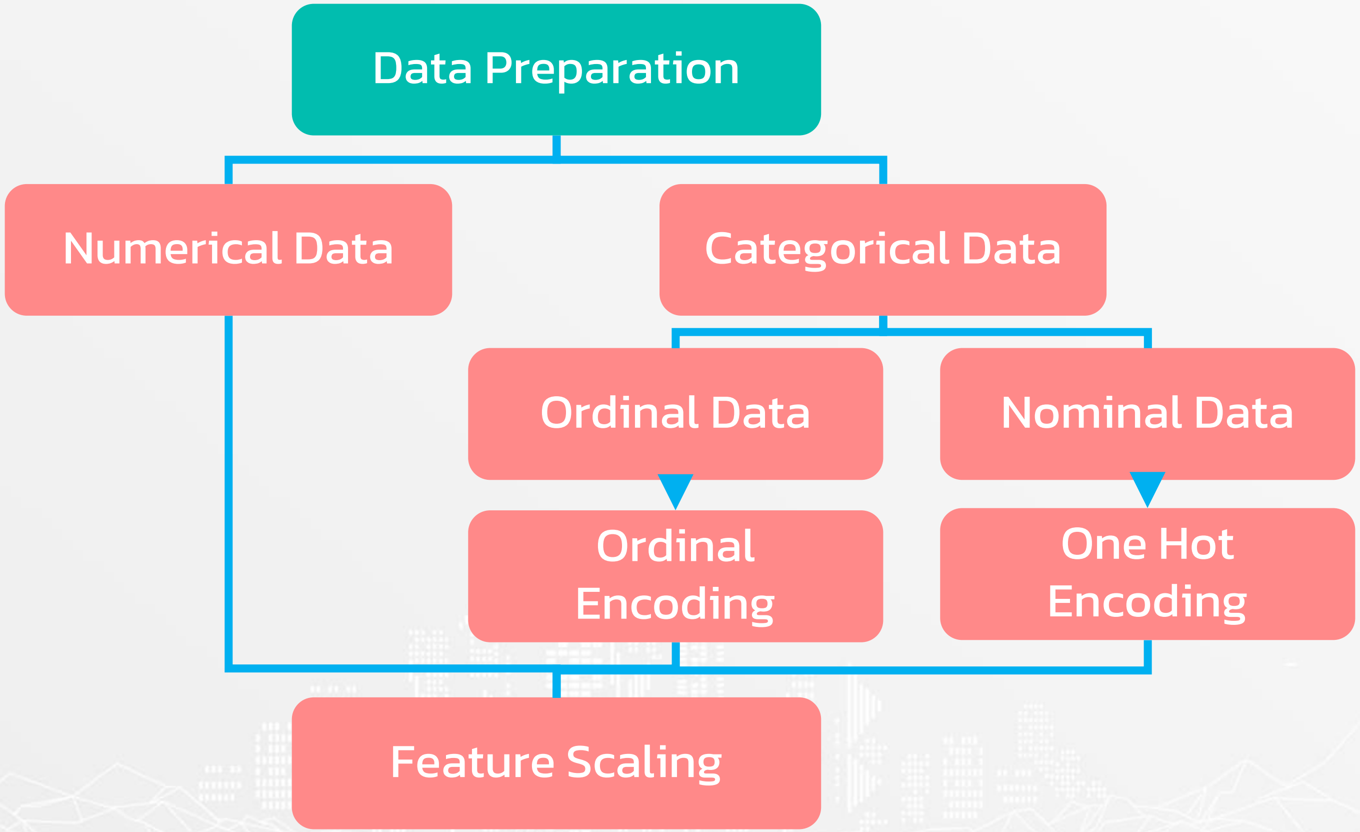
Target

- Sale Price – ราคาขาย

What we learn from this project?



Data Preparation



File

04. HOUSE PRICE

- **sklearn**
- **keras**
- **pytorch**
- **house_price_dataset.csv**

File

04. HOUSE PRICE/sklearn

- **house_price_model.pickle**
- **house_price_sklearn_mc.ipynb**
- **house_price_sklearn _md.ipynb**

File

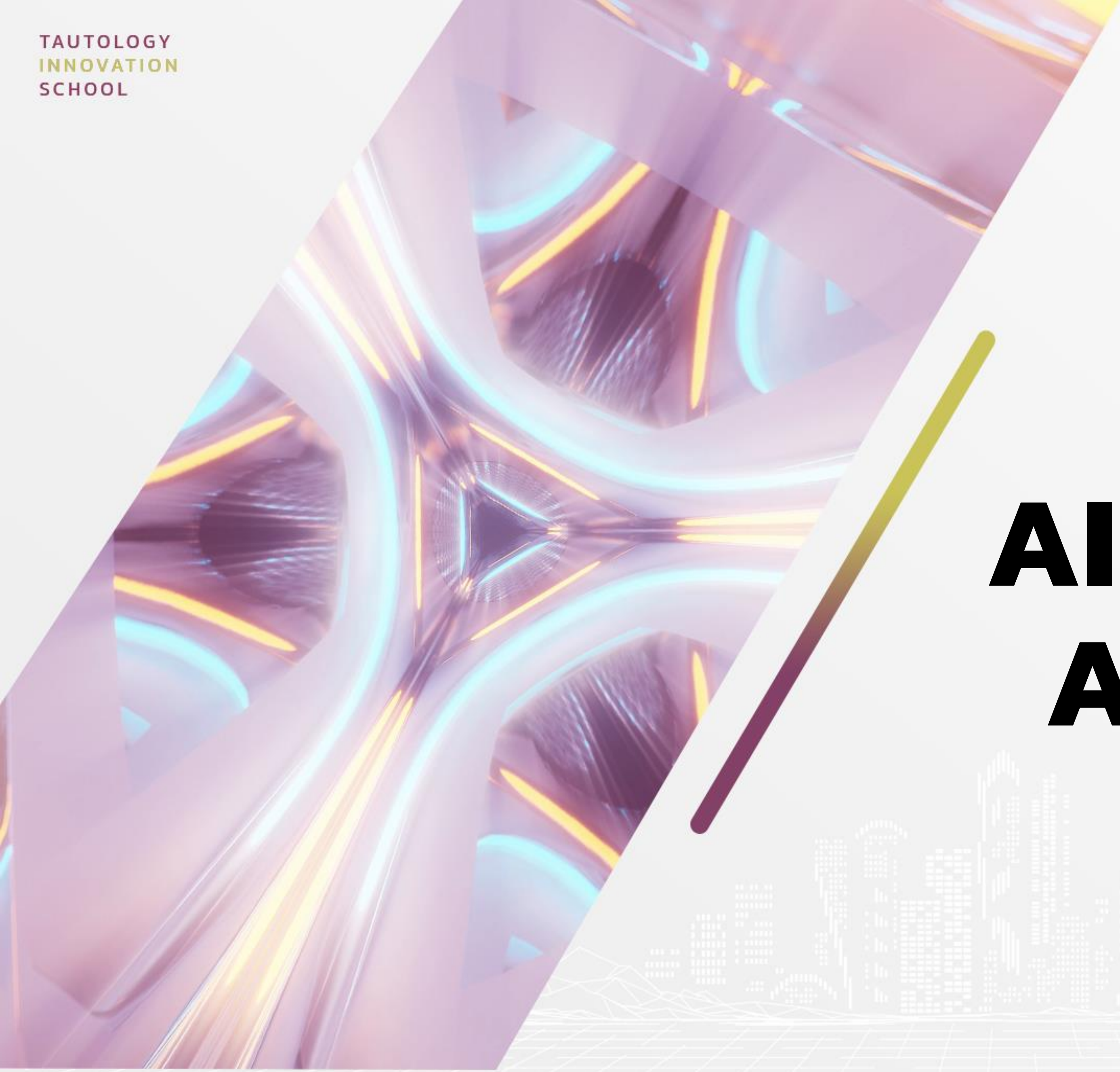
04. HOUSE PRICE/keras

- house_price_env.pickle
- house_price_keras_mc.ipynb
- house_price_keras _md.ipynb
- house_price_model

File

04. HOUSE PRICE/pytorch

- **house_price_env.pickle**
- **house_price_pytorch_mc.ipynb**
- **house_price_pytorch _md.ipynb**
- **house_price_model**



AI in Diagnosing Alzheimer's

Abstract

สร้าง model เพื่อวินิจฉัยโรคอัลไซเมอร์ โดยพิจารณาจากภาพ MRI ของสมอง

Why this project important?

- สามารถสร้างระบบสำหรับตรวจโรคอัลไซเมอร์ที่ทำงานได้ตลอด 24 ชั่วโมง
- สามารถนำไปต่อยอดกับการวินิจฉัยโรคอื่น ๆ
- สามารถใช้เป็นพื้นฐานสำหรับการแพทย์ทางไกล
- สามารถนำไปประยุกต์ใช้กับงานที่มีลักษณะใกล้เคียงกันได้ เช่น การตรวจจับอาวุธในสนามบิน

Who this project is for?

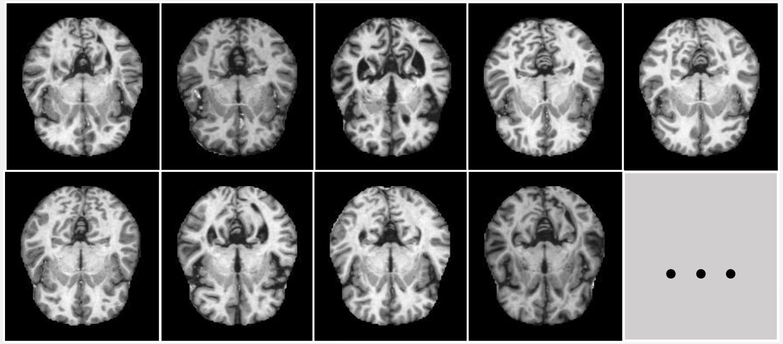
- ผู้บริหารโรงพยาบาล
- บุคลากรทางการแพทย์
- นักวิเคราะห์ข้อมูล

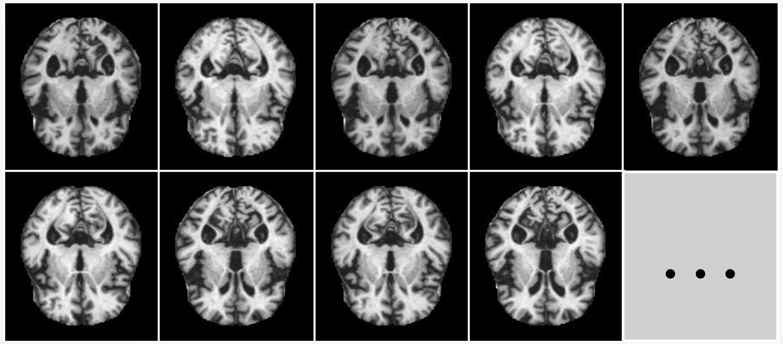
Alzheimer Dataset

https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images

Alzheimer Dataset

Feature





Target

- การเป็นโรคอัลไซเมอร์ (1 = เป็น, 0 = ไม่เป็น)

What we learn from this project?

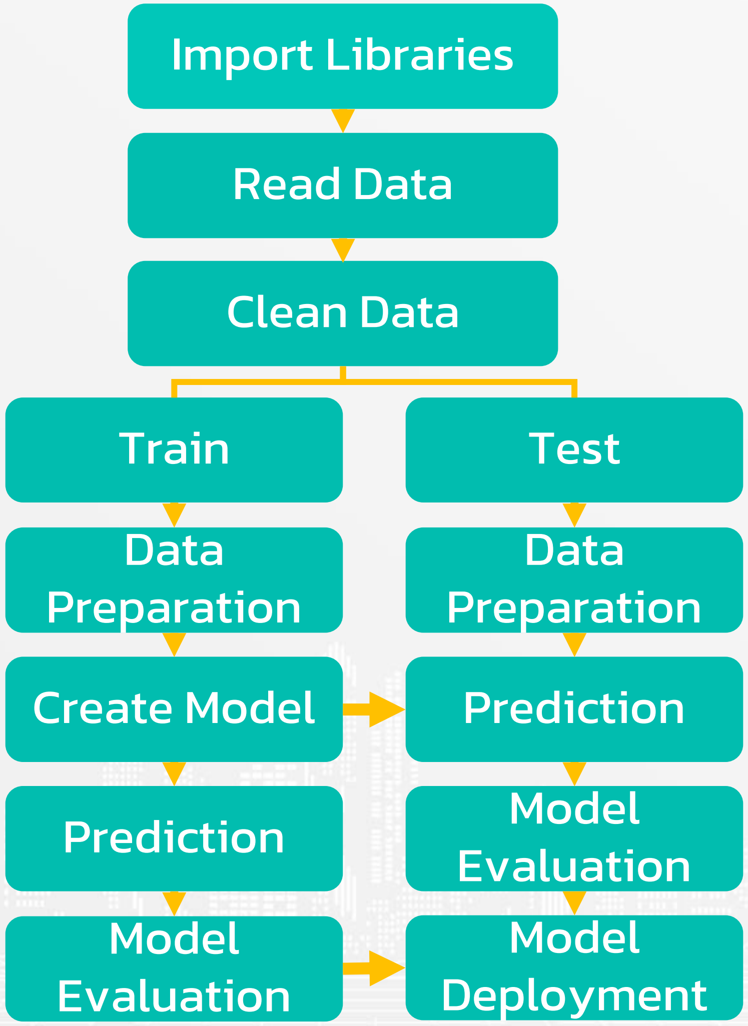


Image to CSV

```
classes = ['MildDemented', 'ModerateDemented', 'NonDemented', 'VeryMildDemented']
```

```
width = 176
height = 208

X = np.empty([0, width*height])
y = np.empty([0, 1])

for _class in tqdm(classes):
    img_path = glob('dataset/' + _class + '/*')
    for path in tqdm(img_path):
        img = Image.open(path)
        img = img.resize([width, height])
        img = np.array(img)
        img = img.reshape(1, -1)
        X = np.vstack([X, img])
        if _class == 'NonDemented':
            y = np.vstack([y, False])
        else:
            y = np.vstack([y, True])
```

Image to CSV

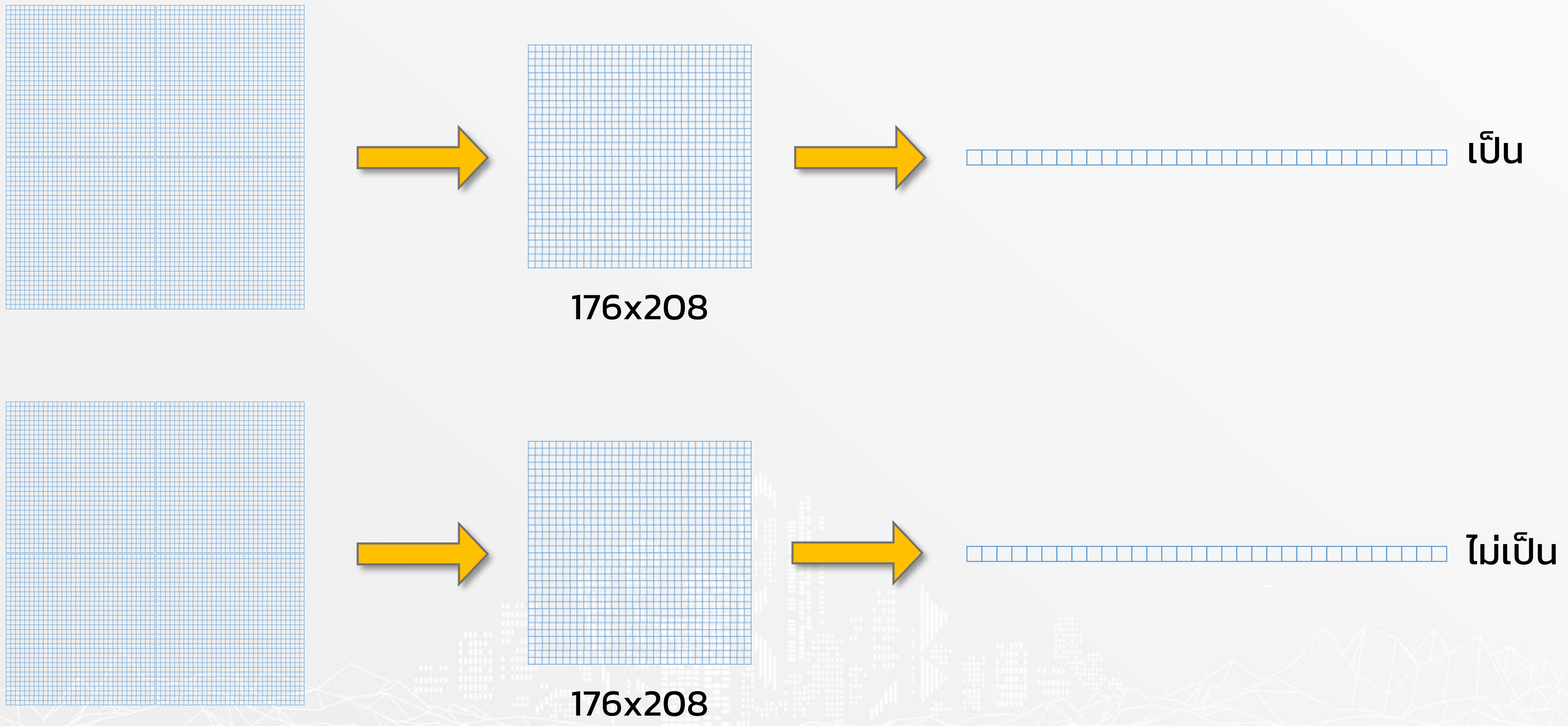


Image to CSV

x_1	x_2	x_3	$\cdots$	x_{36608}
0.0	0.0	0.0	$\cdots$	0.0
0.0	0.0	0.0	$\cdots$	0.0
0.0	0.0	0.0	$\cdots$	0.0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
0.0	0.0	0.0	$\cdots$	0.0

X

y
1.0
1.0
1.0
$\vdots$
1.0

$\mathbf{y}$

Image to CSV

```python
columns = [f'pixel_{i}' for i in range(width*height)]

data = pd.DataFrame(X, columns=columns)
data['label'] = y

data.to_csv('alzheimer_dataset.csv', index=False)
```

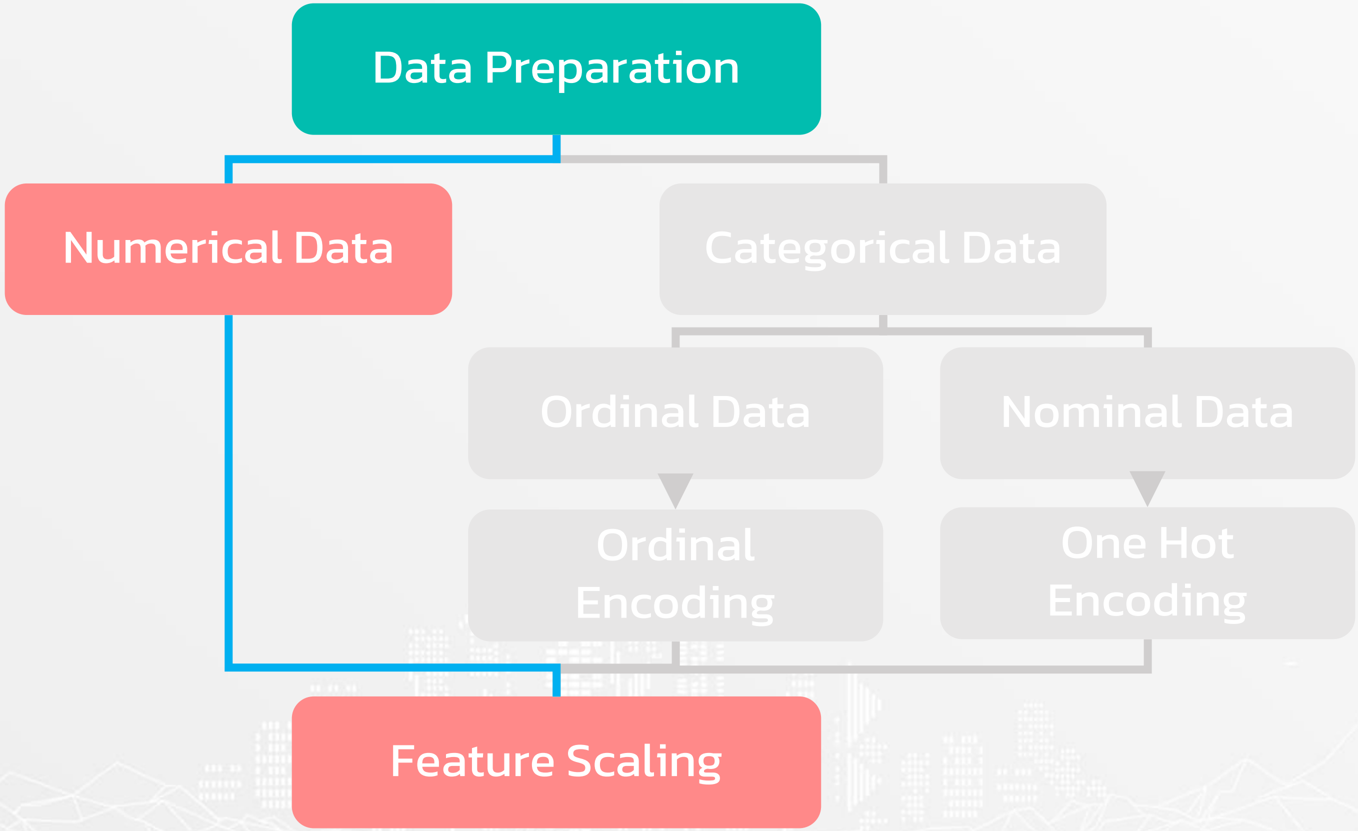
Read Data

```
data = pd.read_csv('../image_to_csv/alzheimer_dataset.csv')

data
```

	pixel_0	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	...	pixel_36606	pixel_36607	label
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5116	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
5117	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
5118	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
5119	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0
5120	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0

5121 rows × 36609 columns

Data Preparation



File

- **05. ALZHEIMER**
  - **sklearn**
  - **keras**
  - **pytorch**
  - **image_to_csv**

File

05. ALZHEIMER/sklearn

- **alzheimer_model.pickle**
- **alzheimer_sklearn_mc.ipynb**
- **alzheimer_sklearn _md.ipynb**

File

05. ALZHEIMER/keras

- **alzheimer_env.pickle**
- **alzheimer_keras_mc.ipynb**
- **alzheimer_keras _md.ipynb**
- **alzheimer_model**

File

05. ALZHEIMER/pytorch

- **alzheimer_env.pickle**
- **alzheimer_pytorch_mc.ipynb**
- **alzheimer_pytorch _md.ipynb**
- **alzheimer_model**

AI in Speech Recognition

Abstract

สร้าง model เพื่อจำแนกคำจากเสียงพูด โดยพิจารณาจากภาพคลื่นเสียง

Why this project important?

- สามารถสร้างระบบที่ทำงานผ่านคำสั่งเสียง
- สามารถสร้างระบบแปลภาษาจากเสียง
- สามารถสร้างระบบถอดโน๊ตจากดนตรี

Who this project is for?

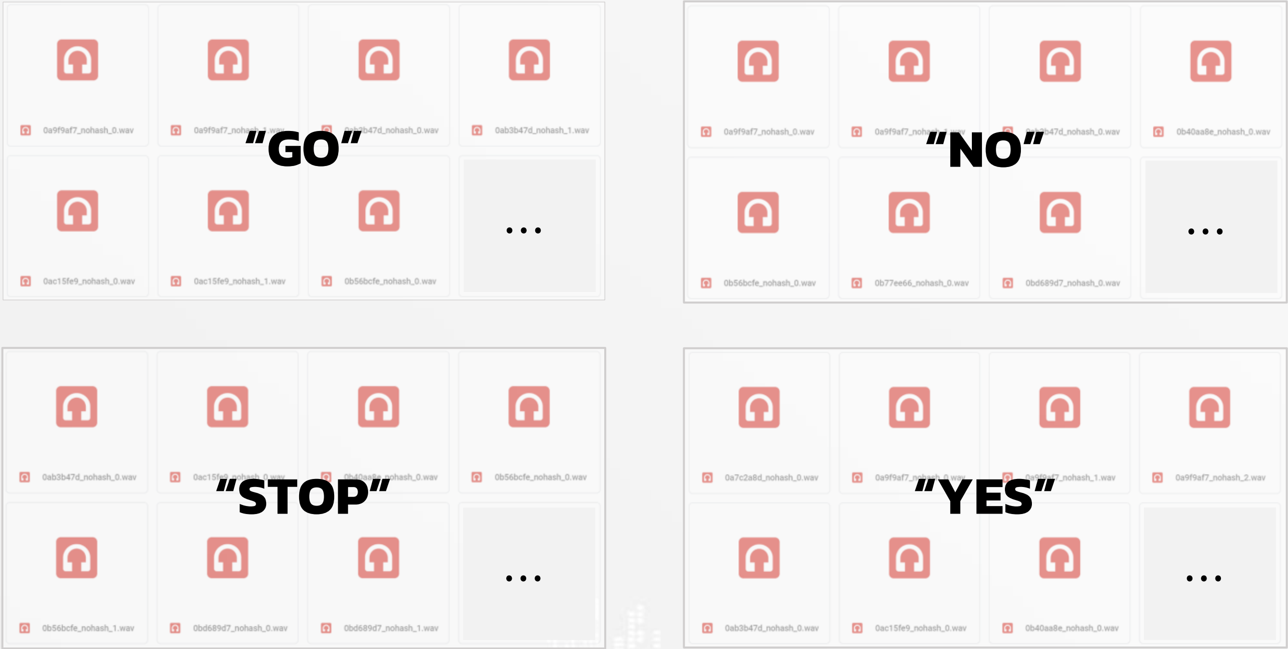
- นักพัฒนาหุ่นยนต์
- นักวิจัยด้านการรู้จำคำพูด
- นักวิเคราะห์ข้อมูล

Voice Command Dataset

https://www.kaggle.com/competitions/tensorflow-speech-recognition-challenge/data

Voice Command Dataset

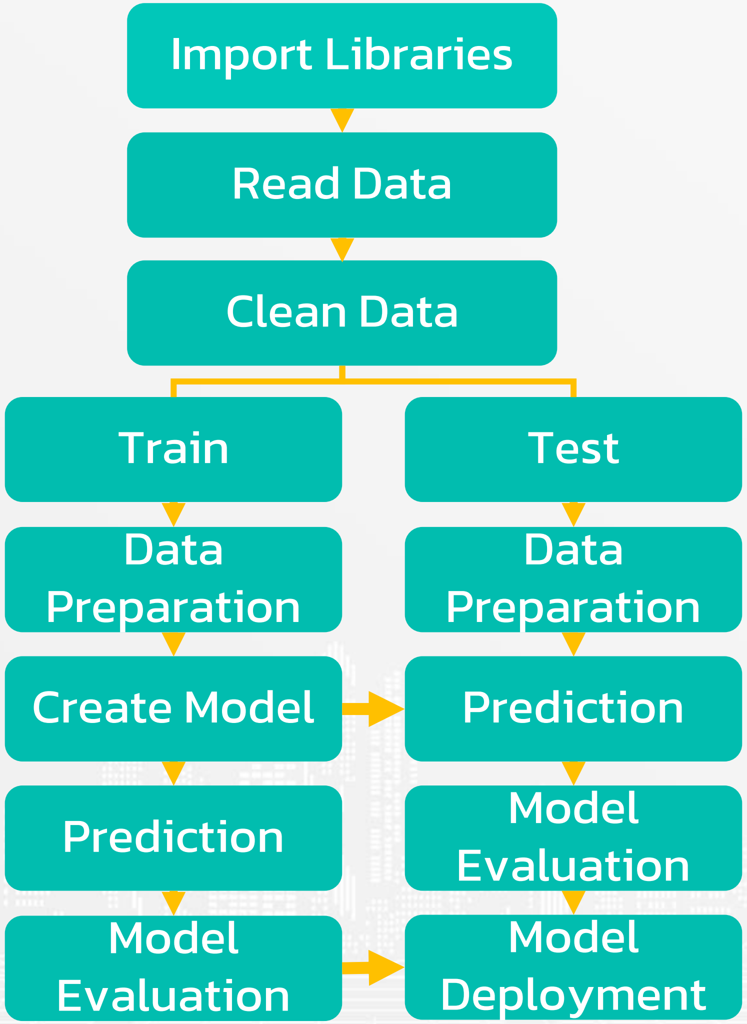
Feature



Target

- target : คำพูด (go, stop, yes, no)

What we learn from this project?



Sound to CSV

```
classes = ['go', 'no', 'stop', 'yes']
```

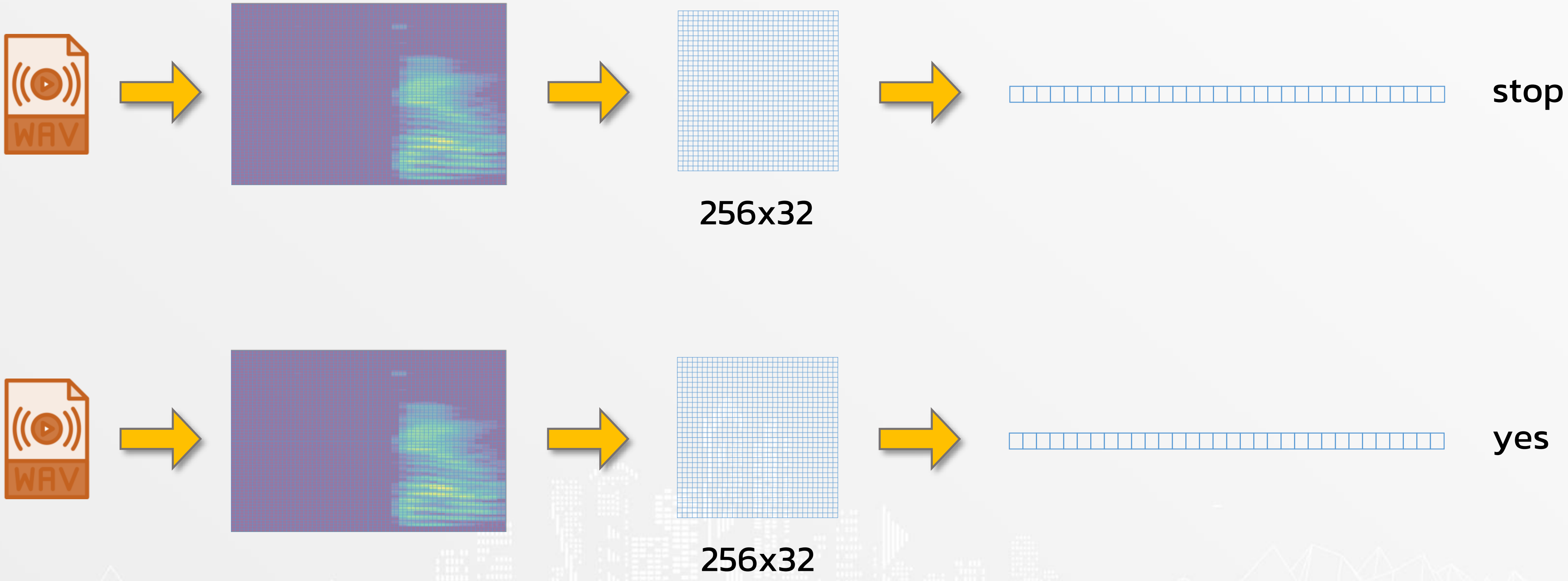
```
mode = 'mel'

width = 256
height = 32

X = np.empty([0, width*height])
y = np.empty([0, 1])

for _class in tqdm(classes):
    sound_path = glob('dataset/' + _class + '/*')
    for path in tqdm(sound_path):
        voice_data, sampling_rate = librosa.load(path)
        img = get_img(voice_data, sampling_rate, mode)
        img = cv2.resize(img, dsize=(width, height))
        img = img.reshape(1, -1)
        X = np.vstack([X, img])
        y = np.vstack([y, _class])
```

Sound to CSV

```python
def get_img(voice_data, sampling_data, mode):
    if mode == 'spec':
        stft = np.abs(librosa.core.spectrum.stft(voice_data))
        return librosa.amplitude_to_db(stft, ref=np.max)
    elif mode == 'mel':
        stft = np.abs(librosa.feature.melspectrogram(voice_data))
        return librosa.amplitude_to_db(stft, ref=np.max)
    elif mode == 'chrom':
        stft = np.abs(librosa.core.spectrum.stft(voice_data))
        return librosa.feature.chroma_stft(S=stft, sr=sampling_rate)
```

Sound to CSV

Sound to CSV



Sound to CSV

x_1	x_2	x_3	$\cdots$	x_{8192}
-80.0	-80.0	-80.0	$\cdots$	-80.0
-80.0	-80.0	-80.0	$\cdots$	-80.0
-80.0	-80.0	-80.0	$\cdots$	-80.0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
-80.0	-80.0	-80.0	$\cdots$	-80.0

y
go
go
go
$\vdots$
yes

X $\qquad$ y

Sound to CSV

```
columns = [f'pixel_{i}' for i in range(width*height)]

data = pd.DataFrame(X, columns=columns)
data['label'] = y

data.to_csv('voice_command_dataset.csv', index=False)
```

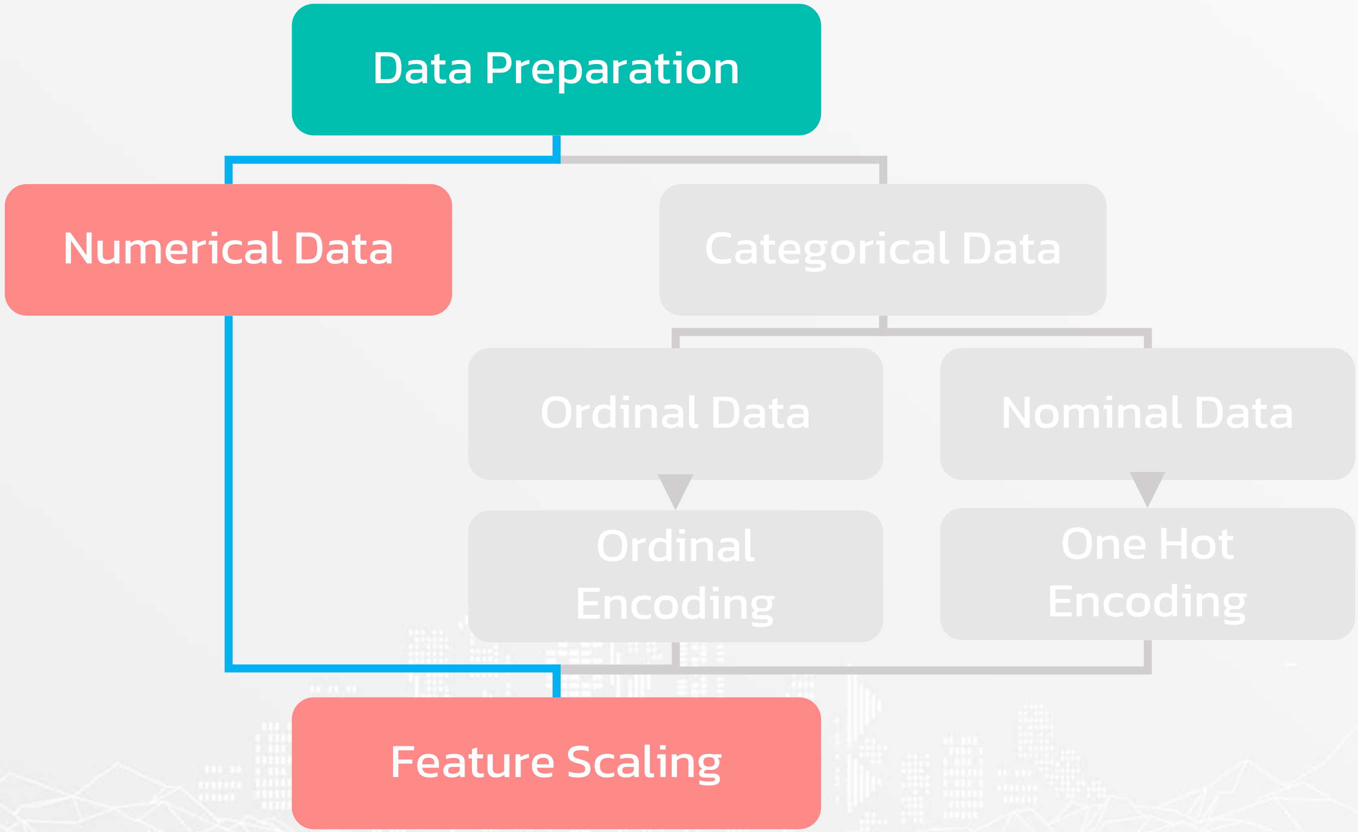
Read Data

```python
data = pd.read_csv('../sound_to_csv/voice_command_dataset.csv')

data
```

	pixel_0	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	...	pixel_8190	pixel_8191	label
0	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	...	-80.0	-80.0	go
1	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	...	-80.0	-80.0	go
2	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	...	-80.0	-80.0	go
3	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	...	-80.0	-80.0	go
4	-62.529934	-62.529934	-62.529934	-62.152111	-61.512722	-60.873333	-60.233940	-59.594551	-58.955162	-58.632915	...	-80.0	-80.0	go
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9499	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	...	-80.0	-80.0	yes
9500	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	...	-80.0	-80.0	yes
9501	-71.944351	-71.944351	-71.944351	-71.608337	-71.039703	-70.471054	-69.902420	-69.333778	-68.765137	-68.614075	...	-80.0	-80.0	yes
9502	-71.259109	-71.259109	-71.259109	-70.686661	-69.717888	-68.749115	-67.780342	-66.811569	-65.842796	-66.136719	...	-80.0	-80.0	yes
9503	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	-80.000000	...	-80.0	-80.0	yes

9504 rows × 8193 columns

Data Preparation



File

06. VOICE COMMAND

- **sklearn**
- **keras**
- **pytorch**
- **sound_to_csv**

TAUTOLOGY

File

06. VOICE COMMAND/sklearn

- **voice_command_model.pickle**
- **voice_command_sklearn_mc.ipynb**
- **voice_command_sklearn _md.ipynb**

File

06. VOICE COMMAND/keras

- **voice_command_env.pickle**
- **voice_command_keras_mc.ipynb**
- **voice_command_keras _md.ipynb**
- **voice_command_model**

File

06. VOICE COMMAND/pytorch

- **voice_command_env.pickle**
- **voice_command_pytorch_mc.ipynb**
- **voice_command_pytorch _md.ipynb**
- **voice_command_model**

DL103 : Deep Learning