

TAUTOLOGY
INNOVATION
SCHOOL

DEEP LEARNING INTERPRETATION



DEEP LEARNING INTERPRETATION

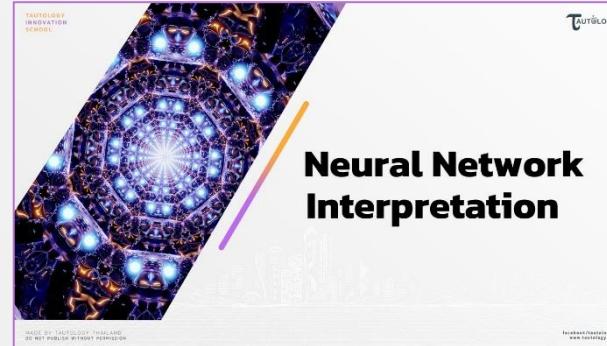
BY TAUTOLOGY

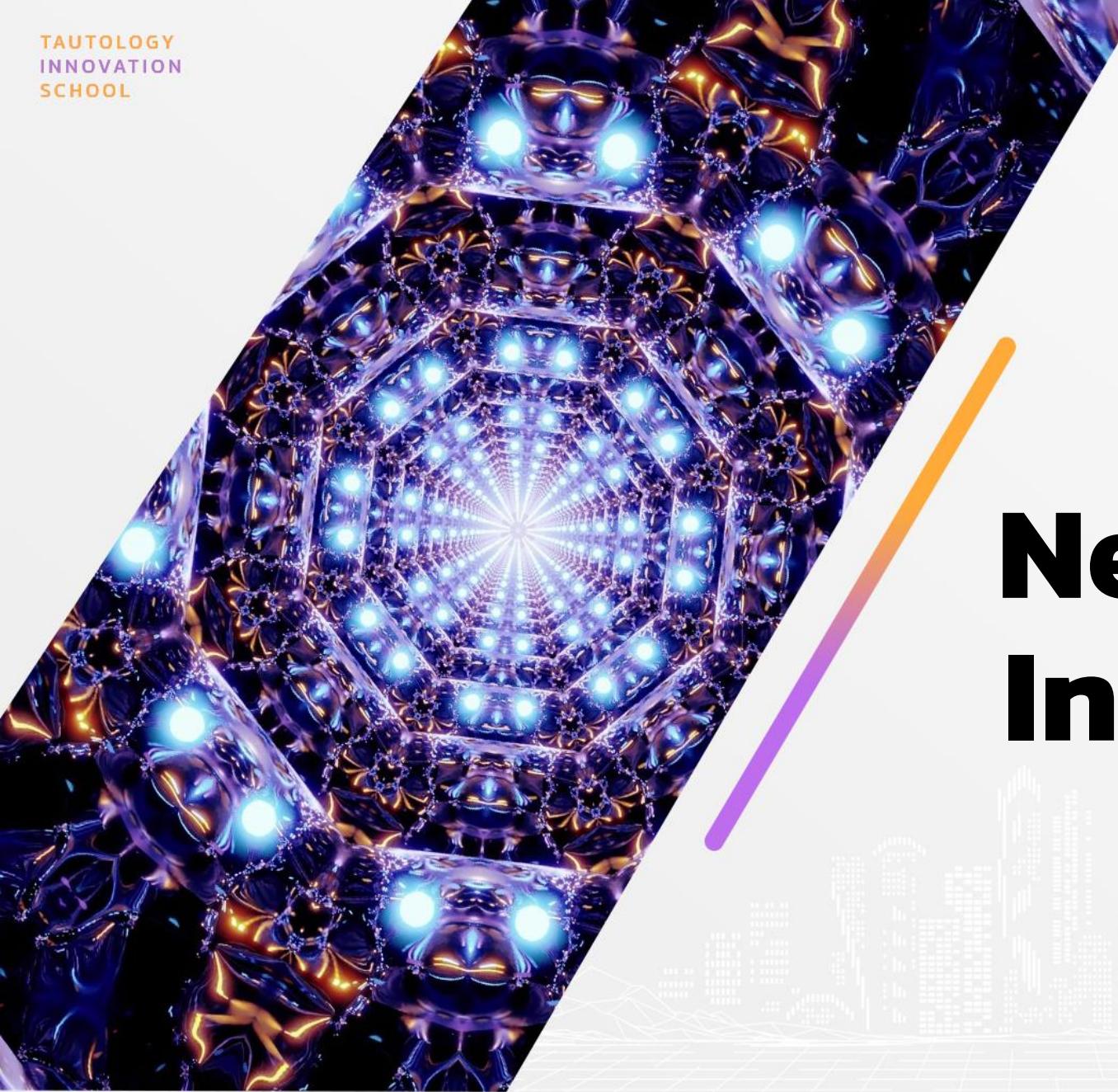
MADE BY TAUTOLOGY THAILAND
DO NOT PUBLISH WITHOUT PERMISSION

facebook/tautologyai
www.tautology.live

TAUTOLOGY

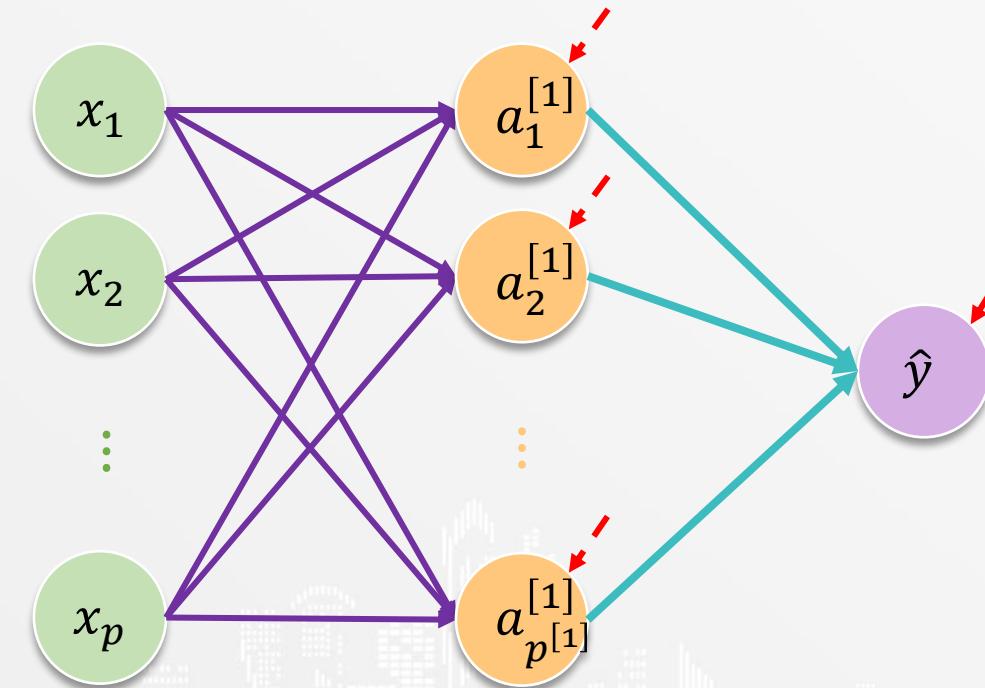
Deep Learning Interpretation





Neural Network Interpretation

Neural Network Interpretation



Neural Network Interpretation

Regression

Classification

Neural Network Interpretation

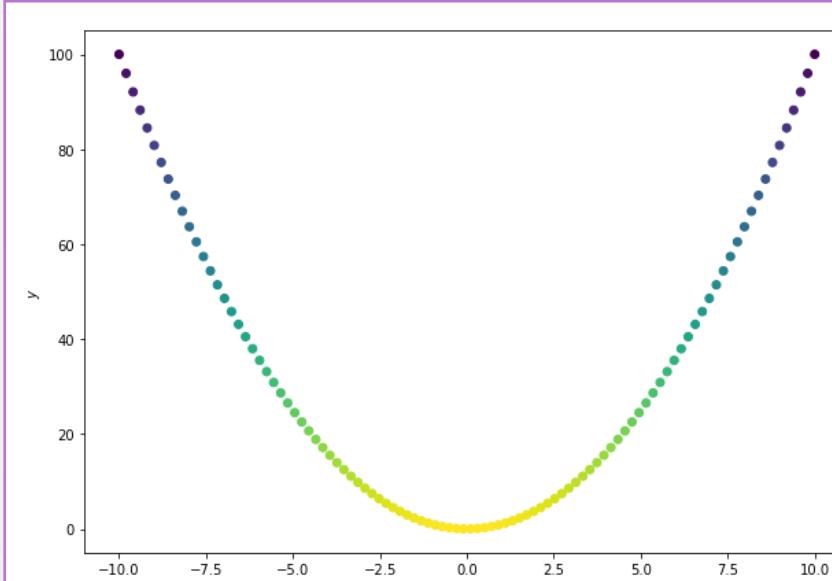
Regression

Classification

Neural Network Interpretation

Regression

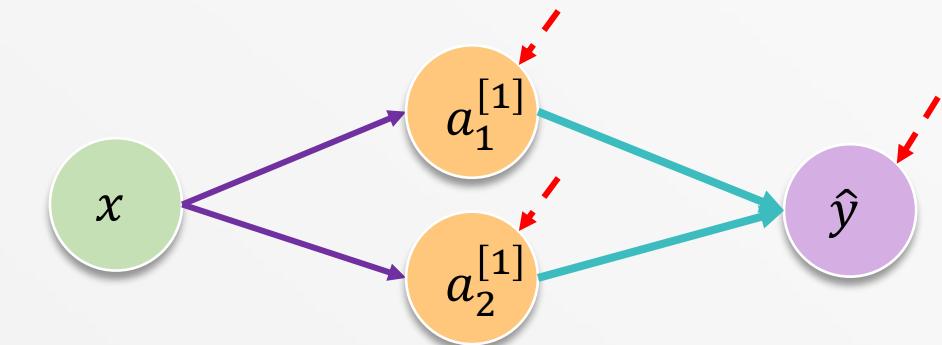
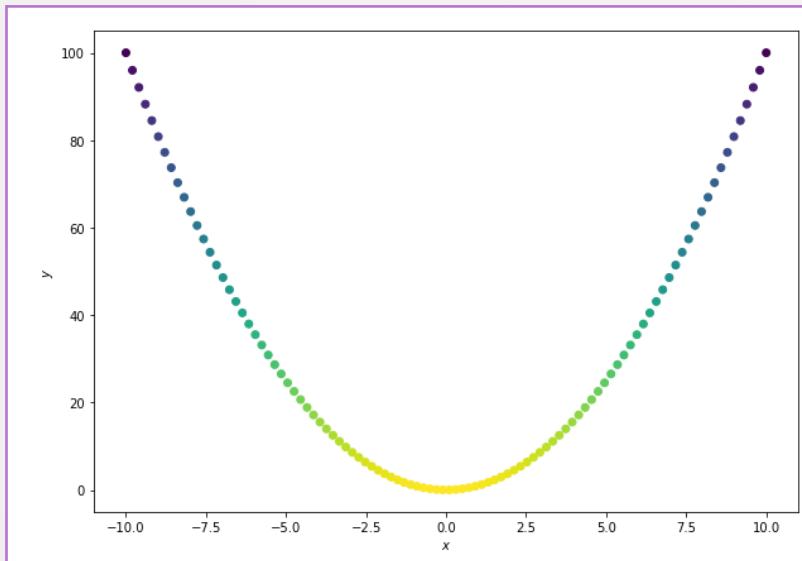
Example 1



Neural Network Interpretation

Regression

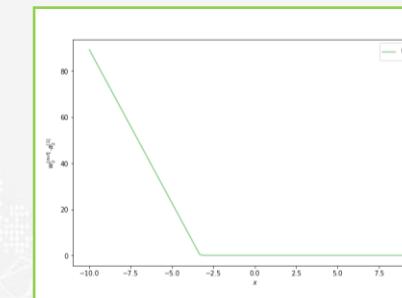
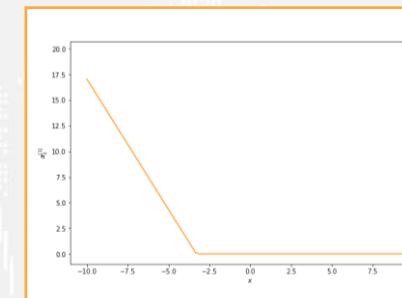
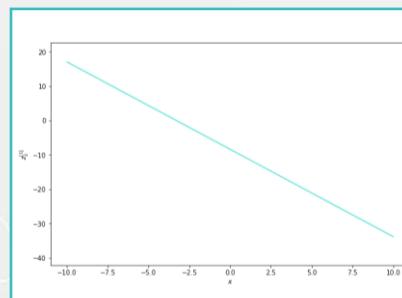
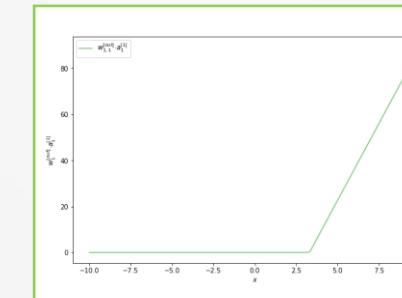
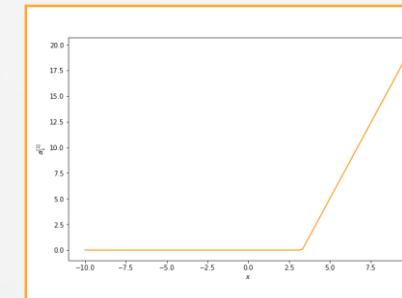
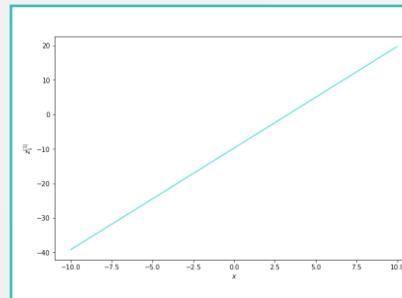
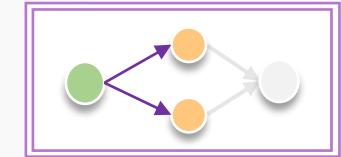
Example 1.1



Neural Network Interpretation

Regression

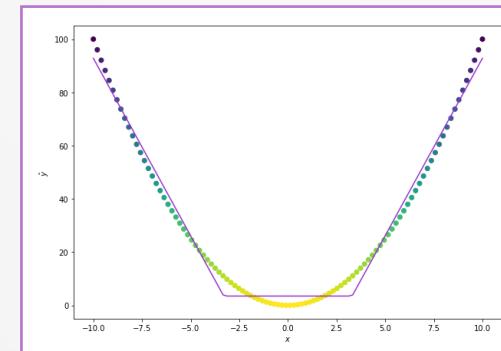
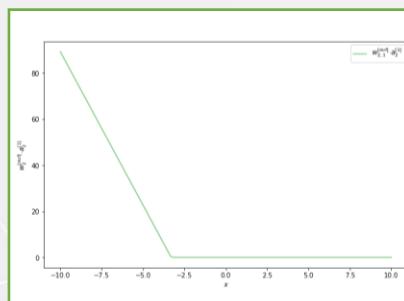
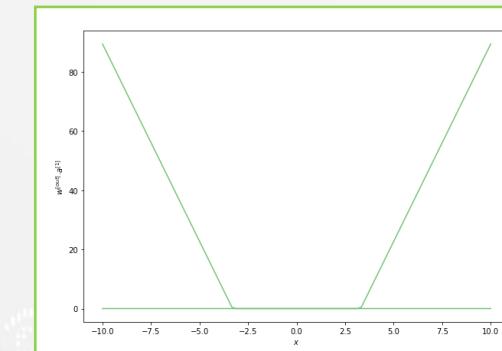
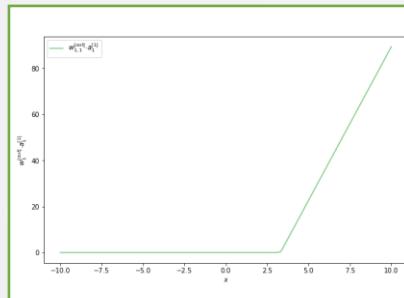
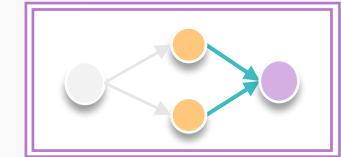
Example 1.1



Neural Network Interpretation

Regression

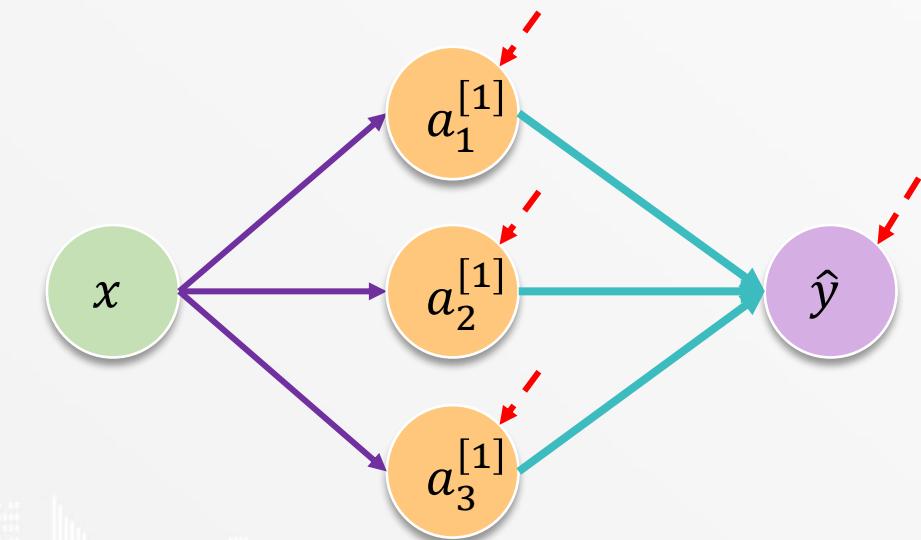
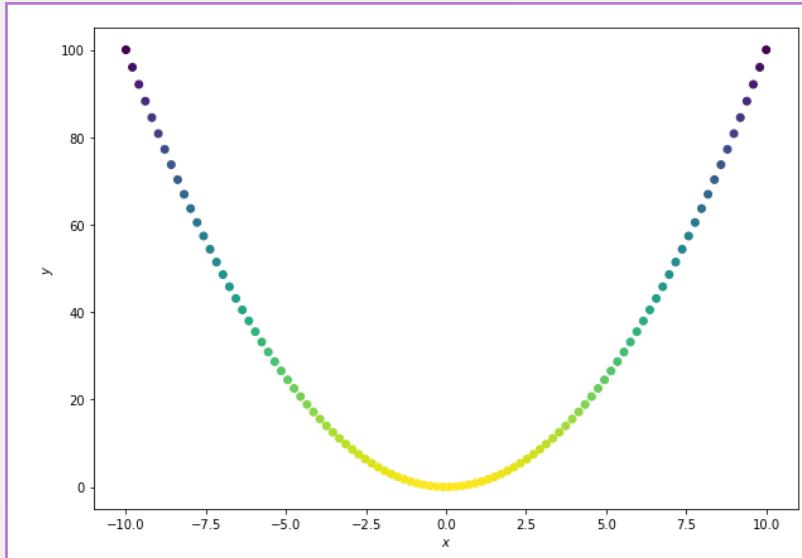
Example 1.1



Neural Network Interpretation

Regression

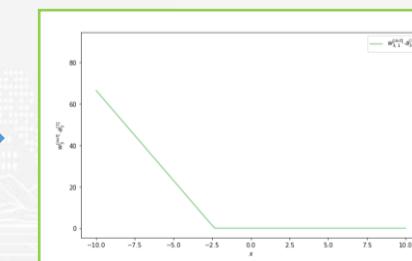
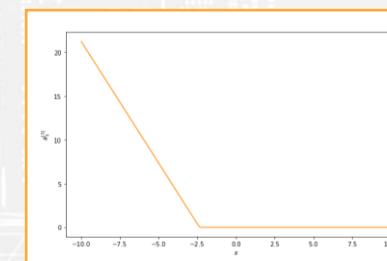
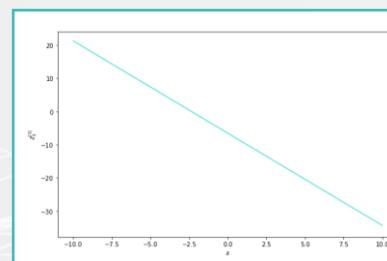
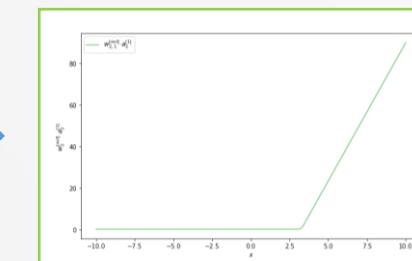
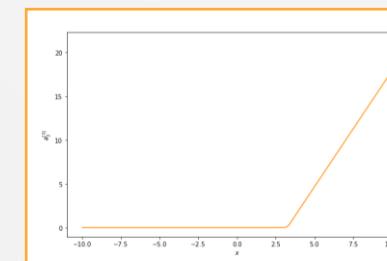
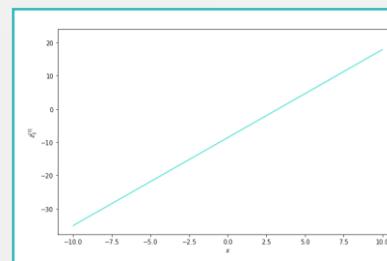
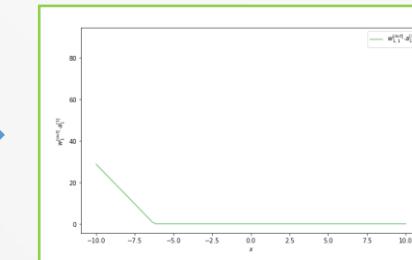
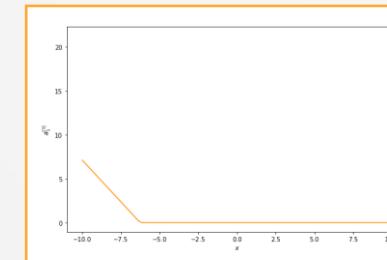
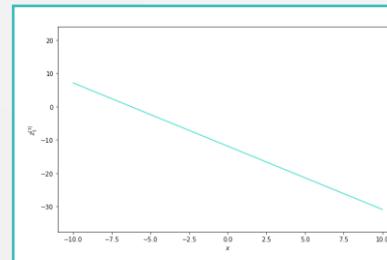
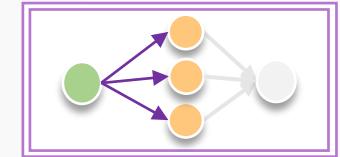
Example 1.2



Neural Network Interpretation

Regression

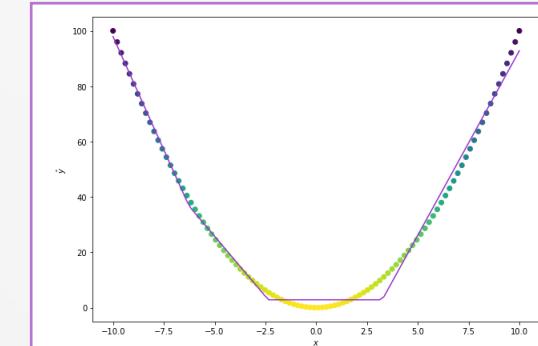
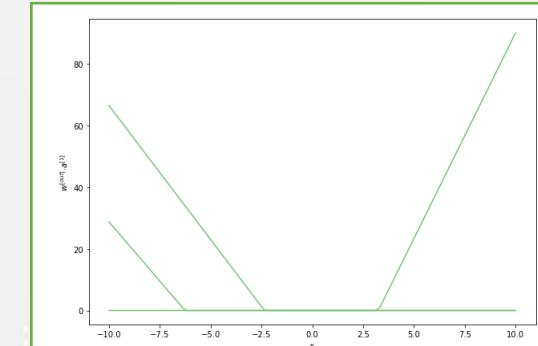
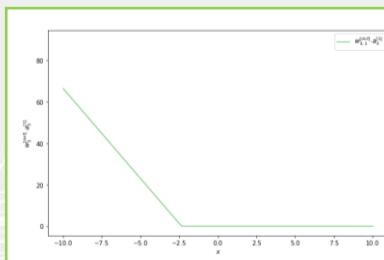
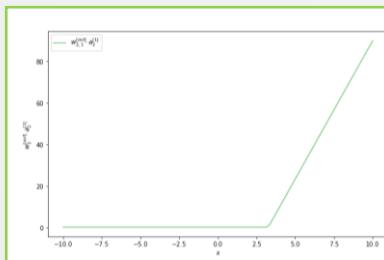
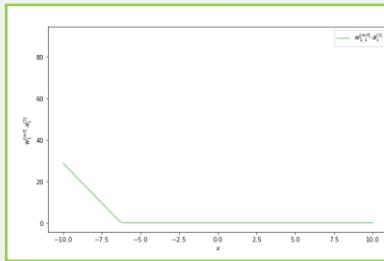
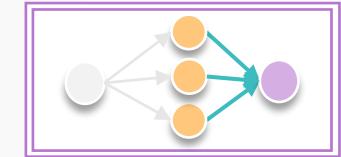
Example 1.2



Neural Network Interpretation

Regression

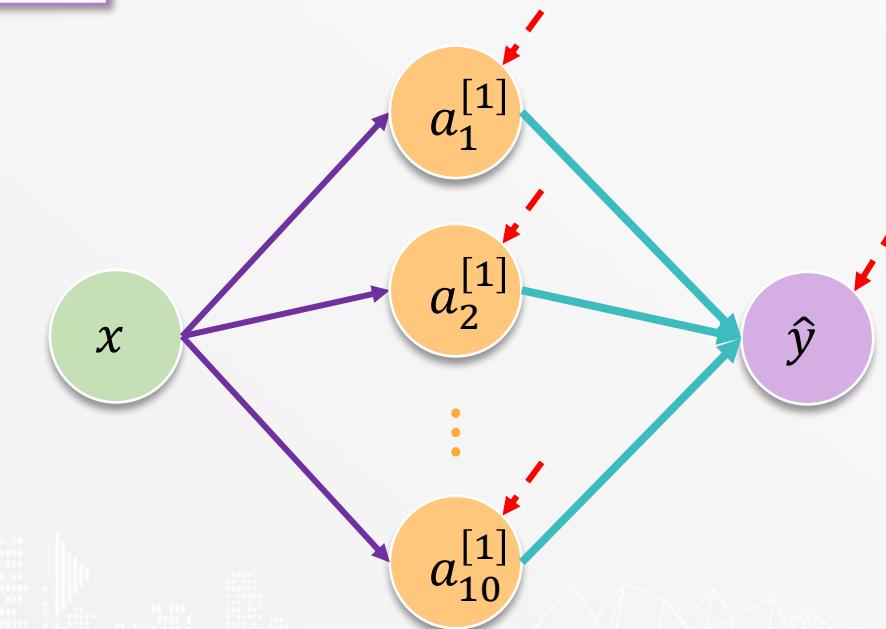
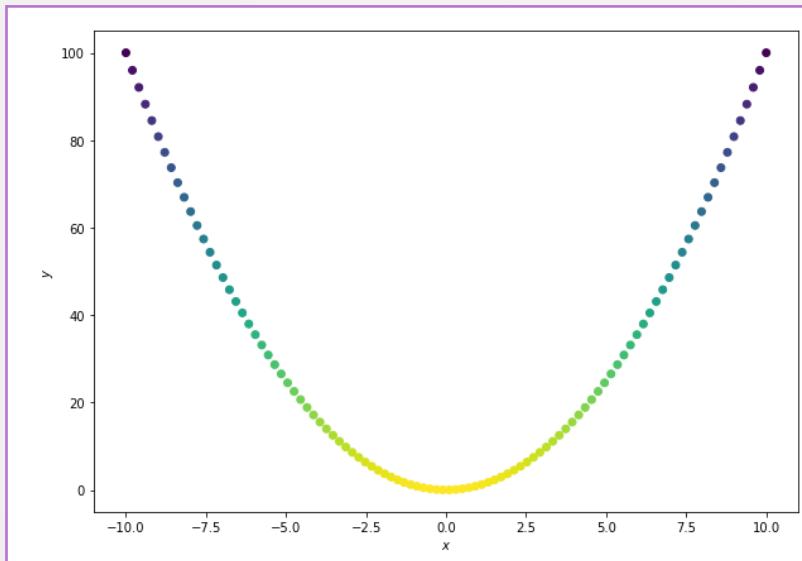
Example 1.2



Neural Network Interpretation

Regression

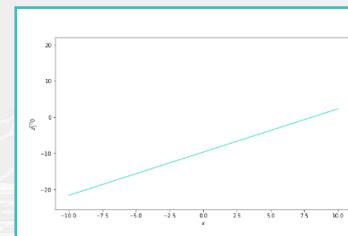
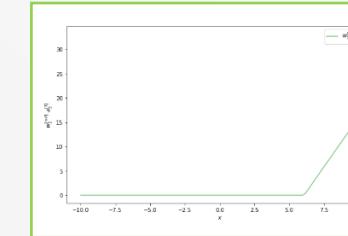
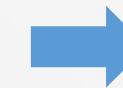
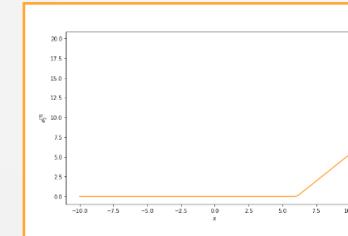
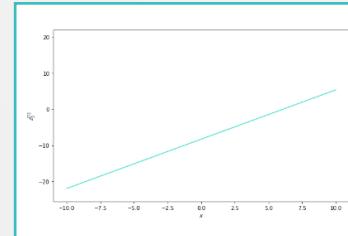
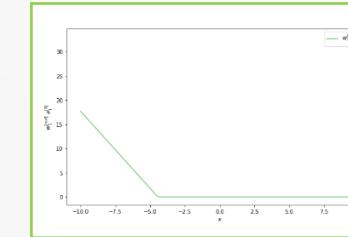
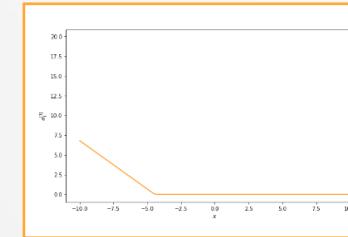
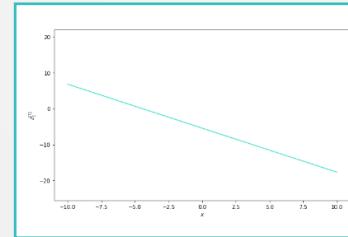
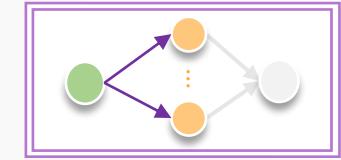
Example 1.3



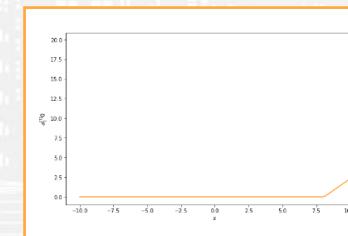
Neural Network Interpretation

Regression

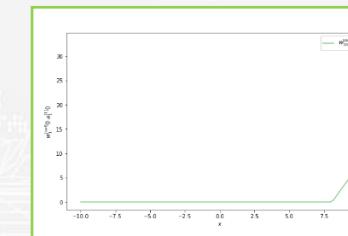
Example 1.3



10



10

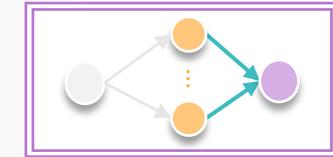
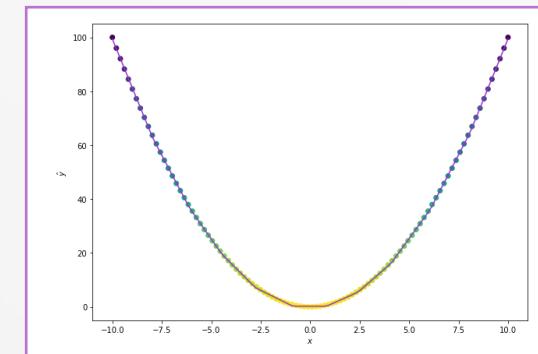
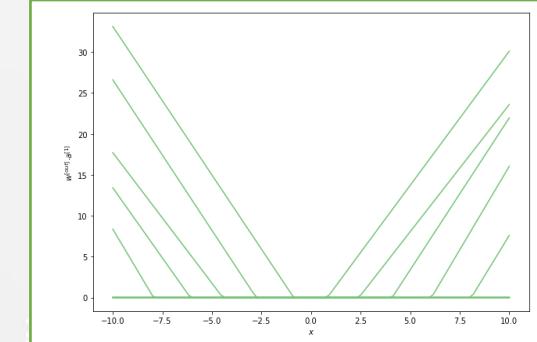
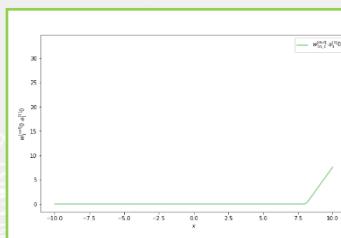
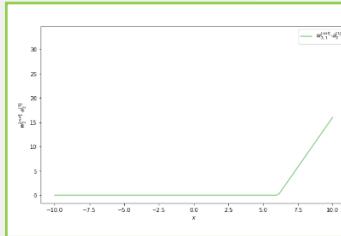
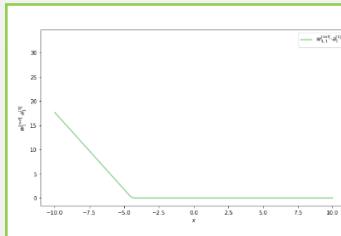


10

Neural Network Interpretation

Regression

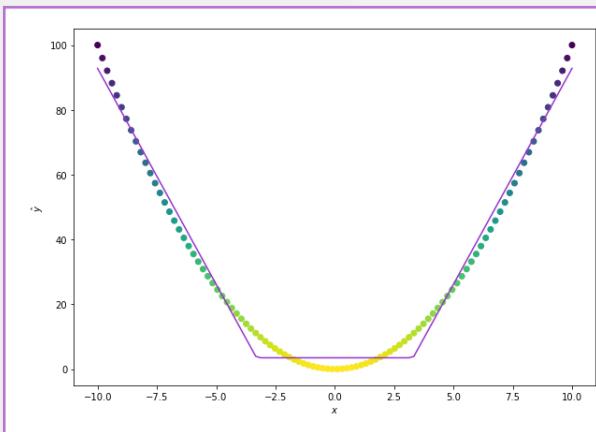
Example 1.3



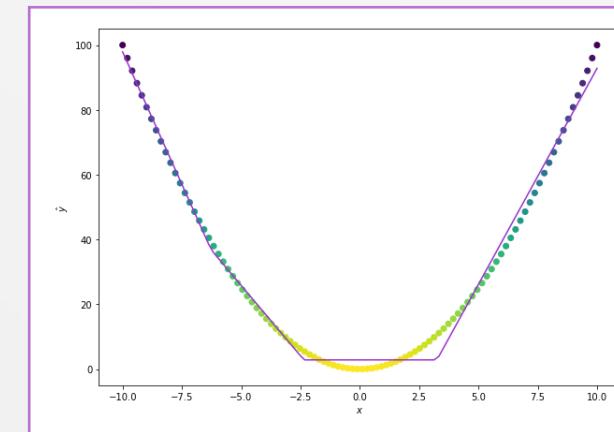
Neural Network Interpretation

Regression

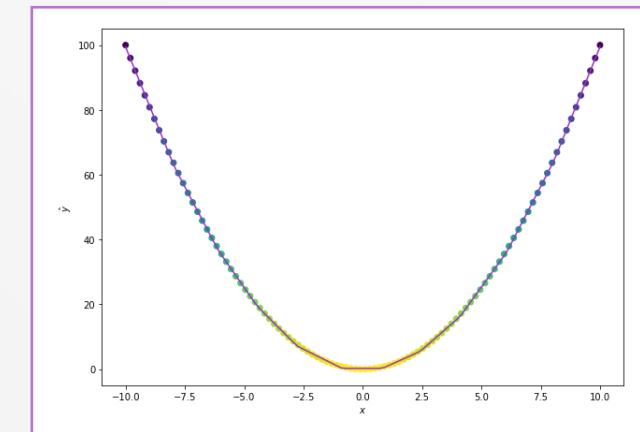
กราฟแสดงค่า predicted ของ Example 1



Example 1.1 (1,2,1)



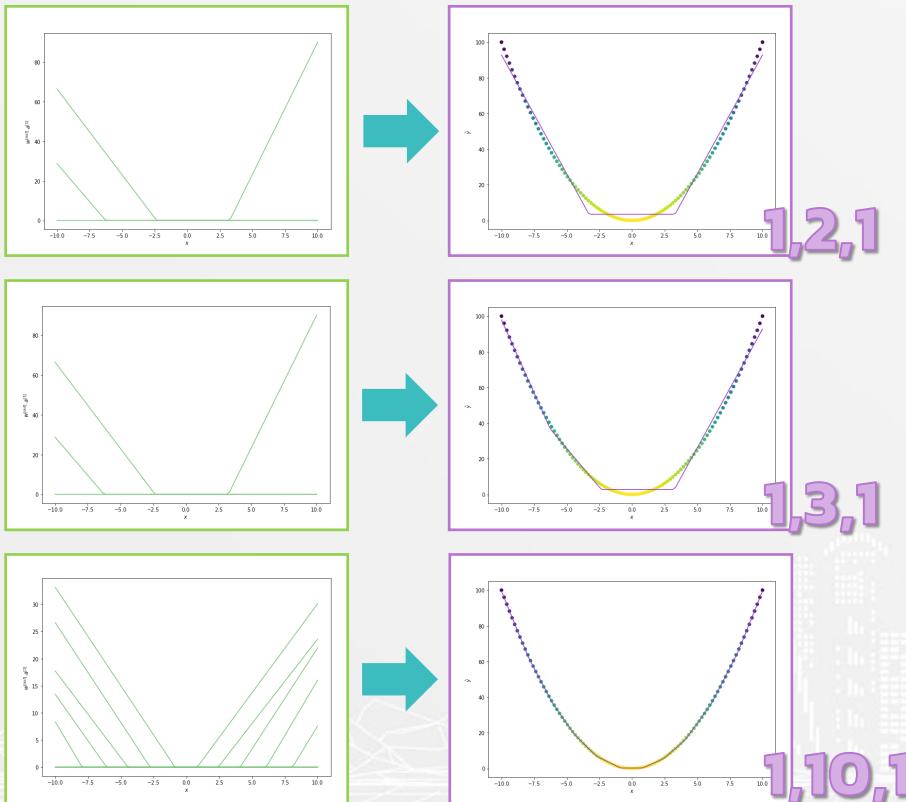
Example 1.2 (1,3,1)



Example 1.3 (1,10,1)

Neural Network Interpretation

Regression



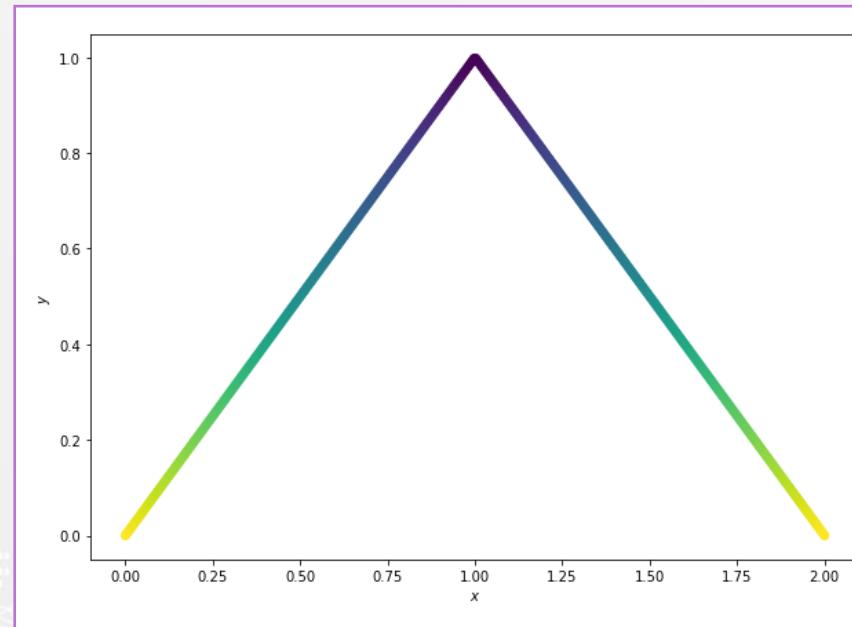
ข้อสรุปของ Example 1

- การเพิ่ม node ทำให้เกิด plane (ส่วนของเส้นตรง) ย่อ去ที่มากขึ้น
- plane (ส่วนของเส้นตรง) ย่อ去ที่มากขึ้นนั้นเกิดจากการประกอบกันของ function ย่อ去หลังผ่าน activation function

Neural Network Interpretation

Regression

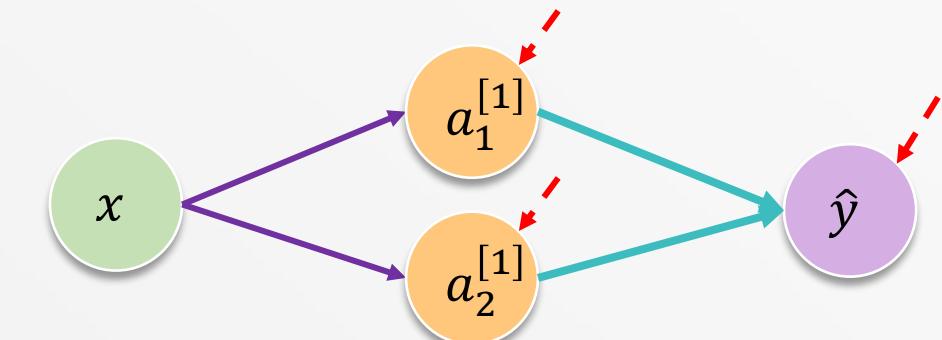
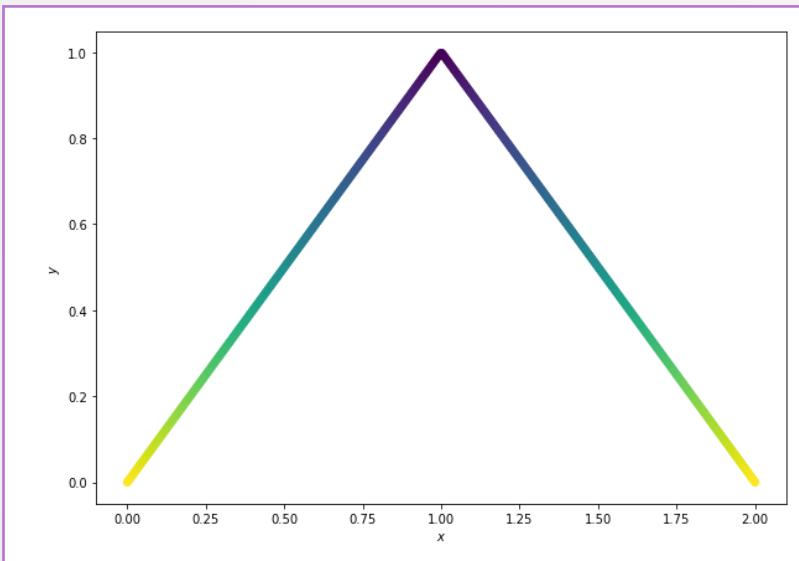
Example 2



Neural Network Interpretation

Regression

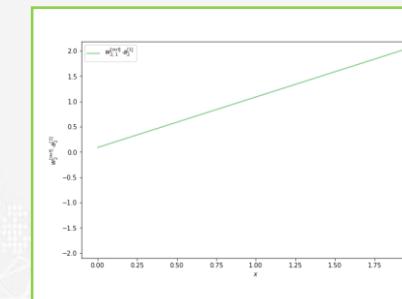
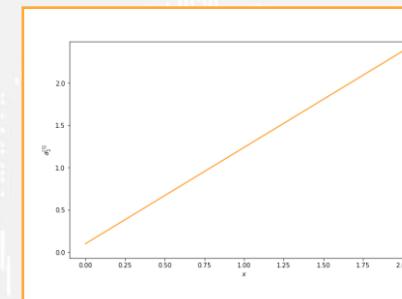
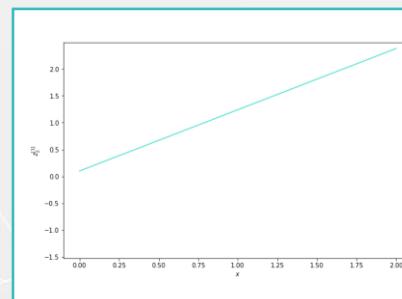
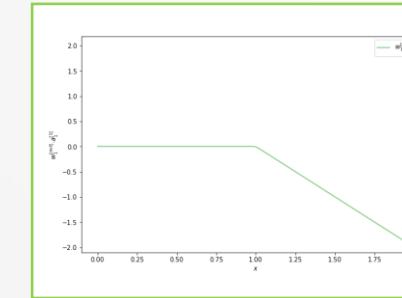
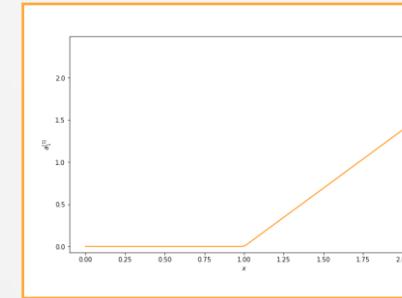
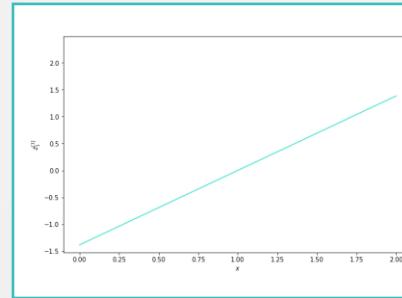
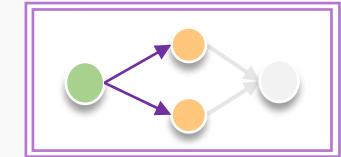
Example 2



Neural Network Interpretation

Regression

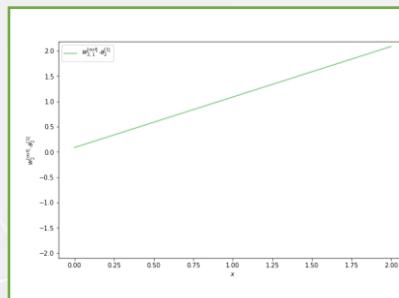
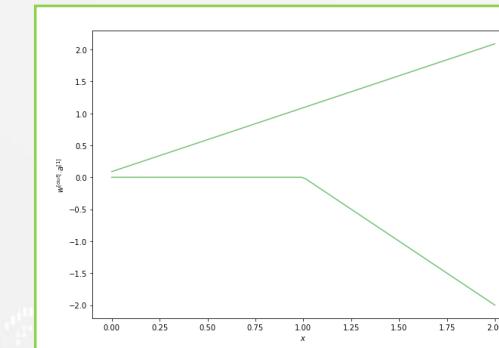
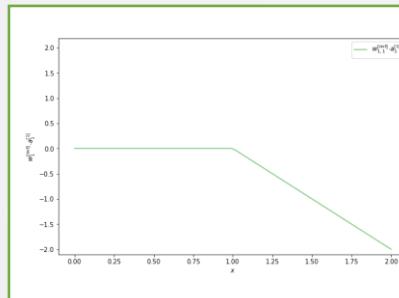
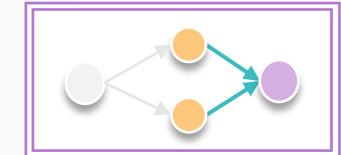
Example 2



Neural Network Interpretation

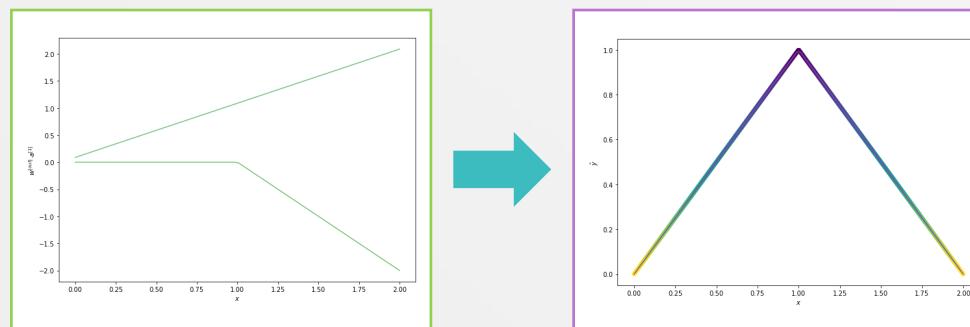
Regression

Example 2



Neural Network Interpretation

Regression



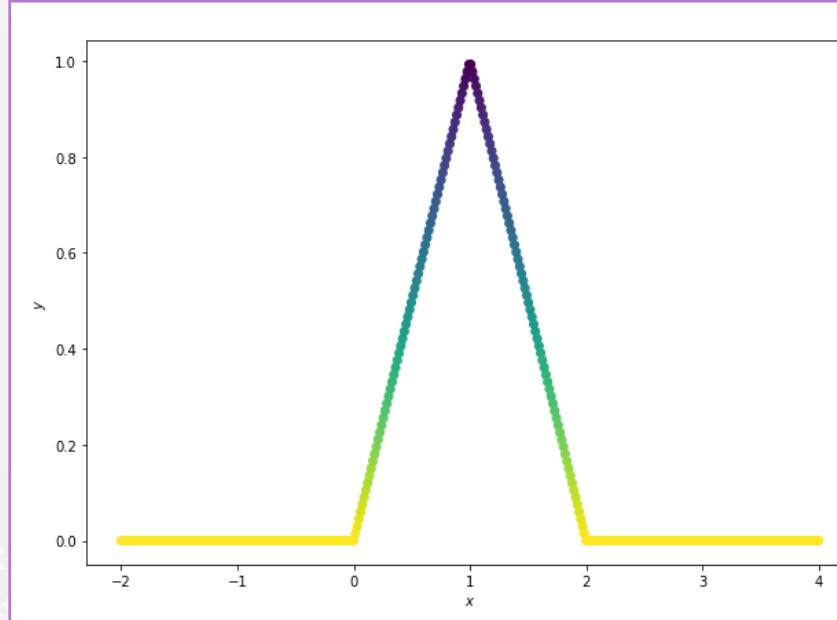
ข้อสรุปของ Example 2

- การประกอบกันของ function ย่อยหลังผ่าน activation function เกิดขึ้นก็การเสริมกัน และการหักล้างกัน

Neural Network Interpretation

Regression

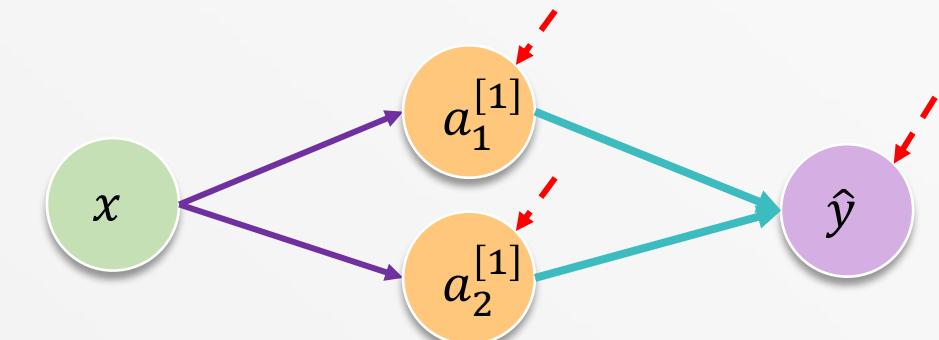
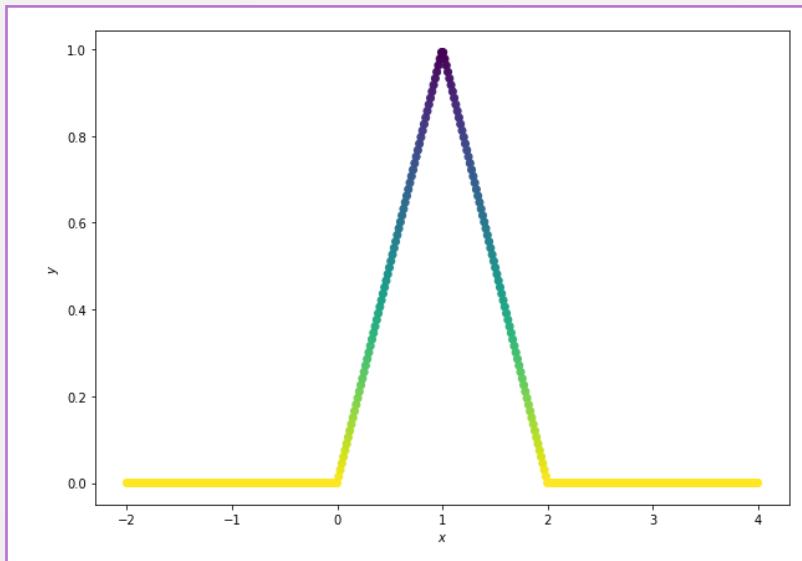
Example 3



Neural Network Interpretation

Regression

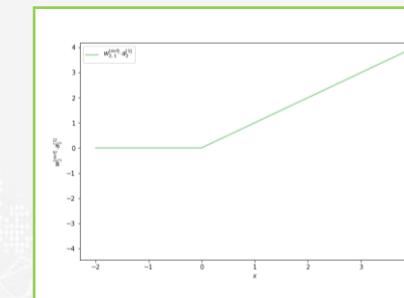
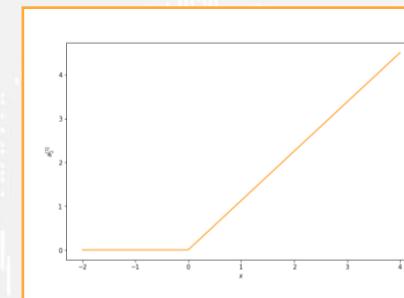
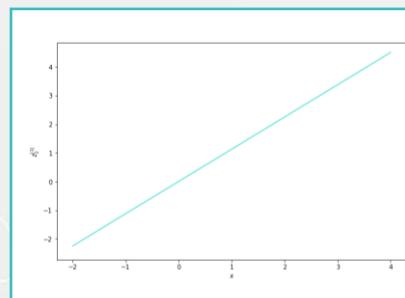
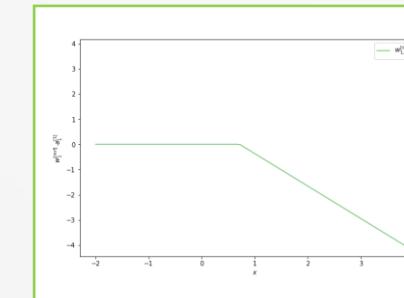
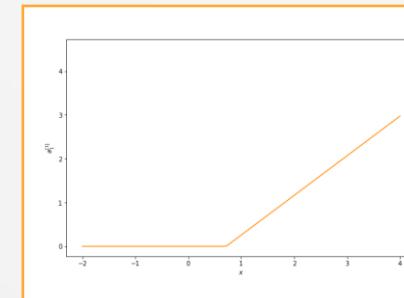
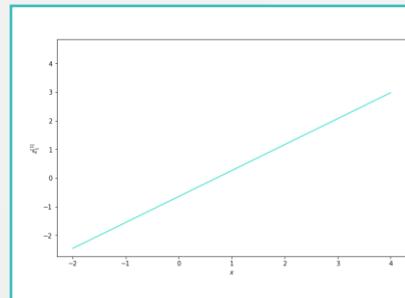
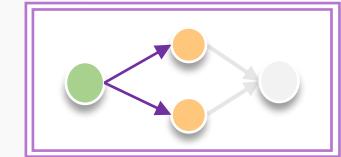
Example 3.1



Neural Network Interpretation

Regression

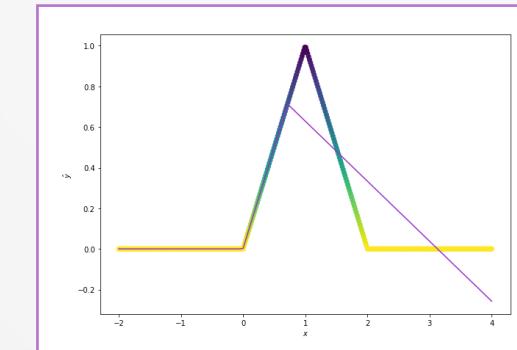
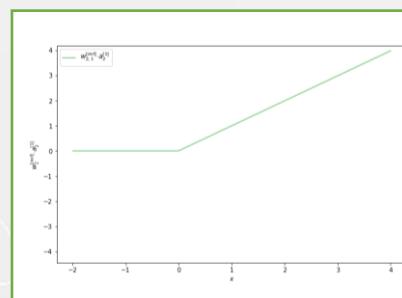
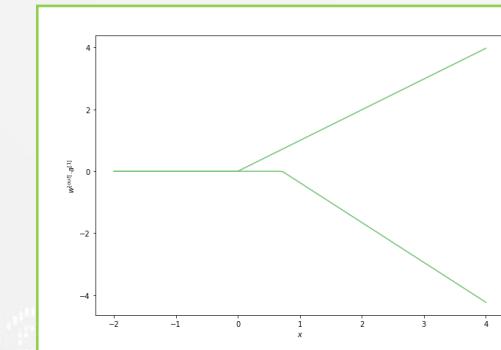
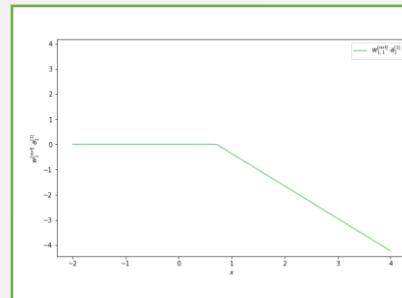
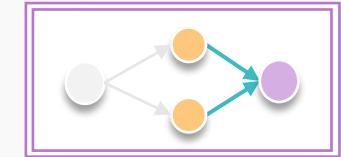
Example 3.1



Neural Network Interpretation

Regression

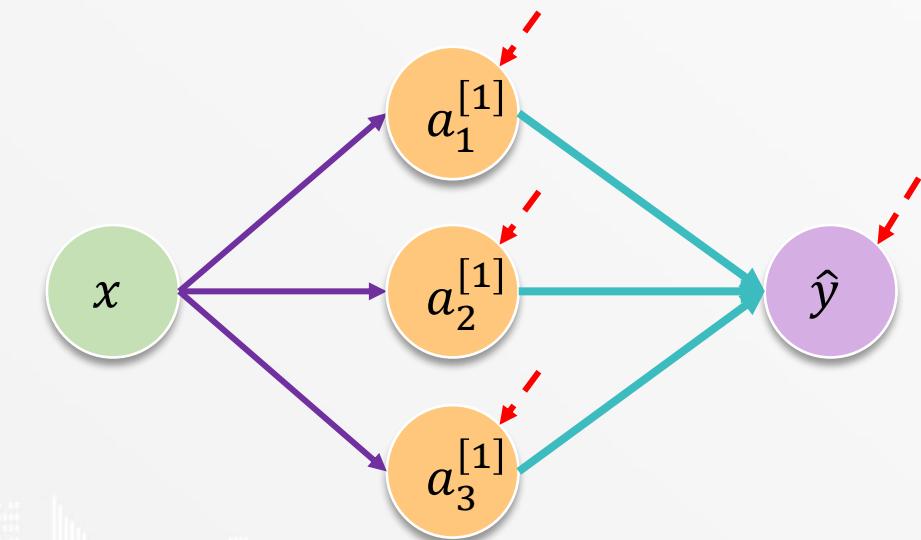
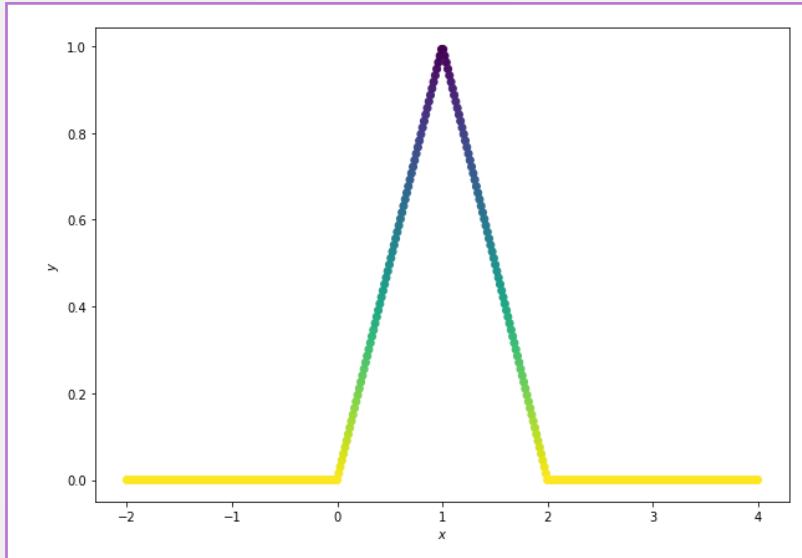
Example 3.1



Neural Network Interpretation

Regression

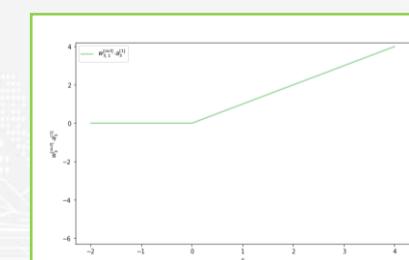
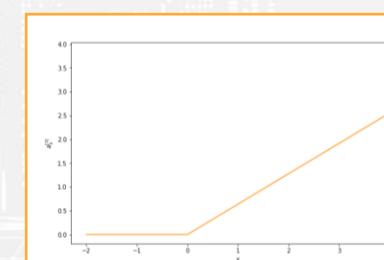
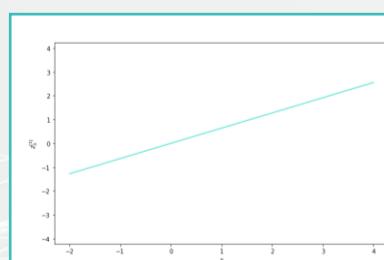
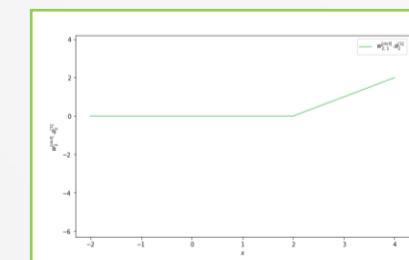
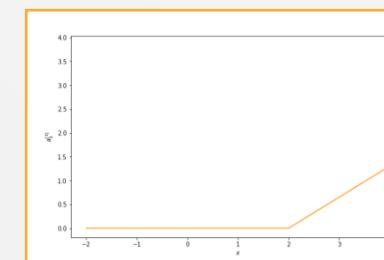
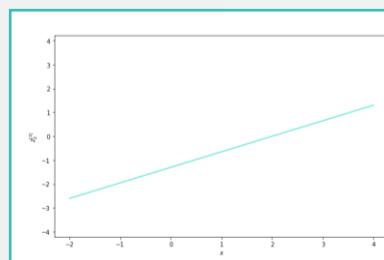
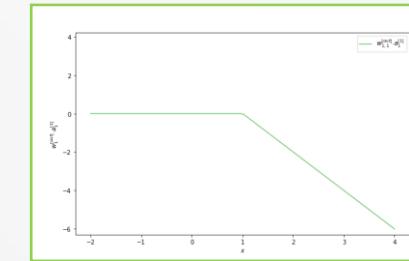
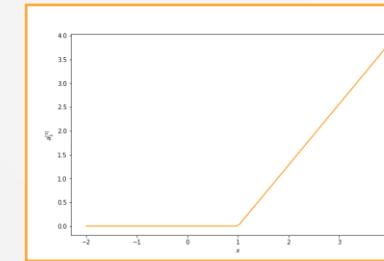
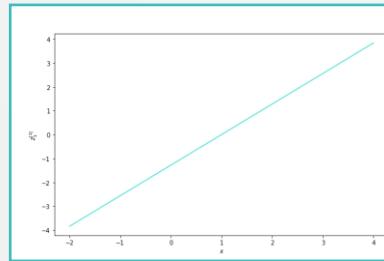
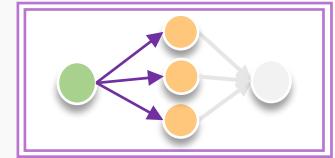
Example 3.2



Neural Network Interpretation

Regression

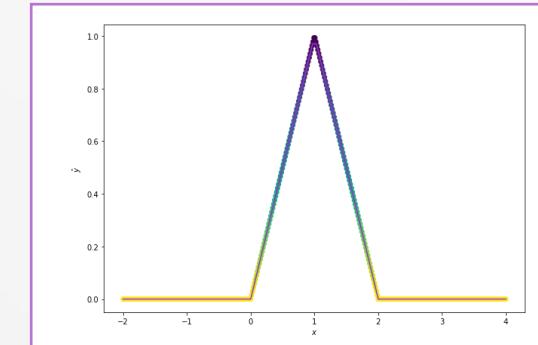
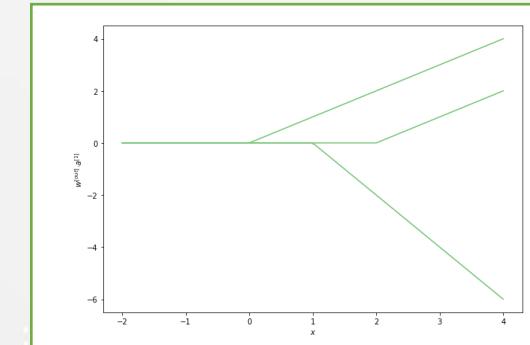
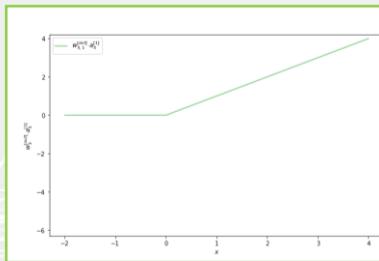
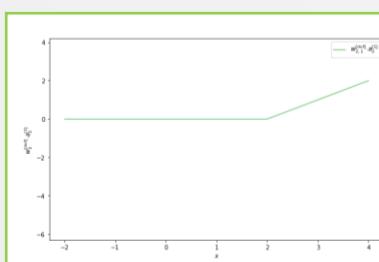
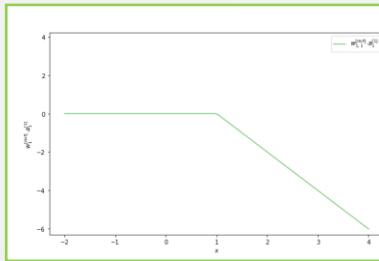
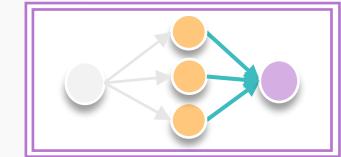
Example 3.2



Neural Network Interpretation

Regression

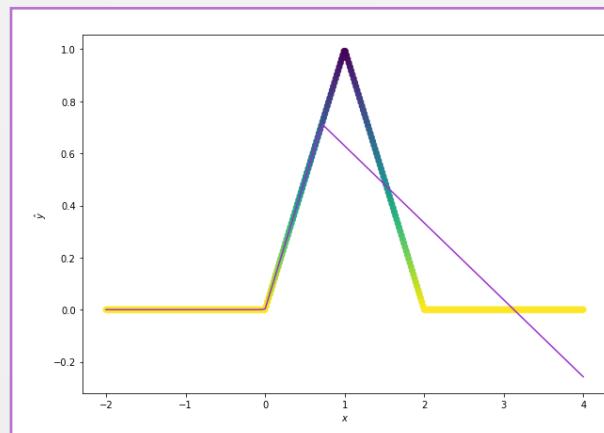
Example 3.2



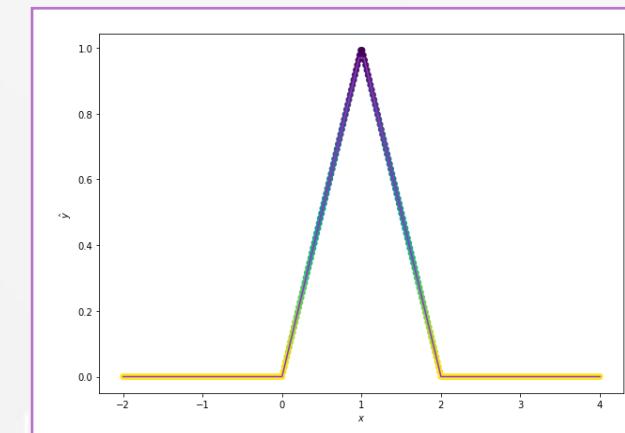
Neural Network Interpretation

Regression

กราฟแสดงค่า predicted ของ Example 3



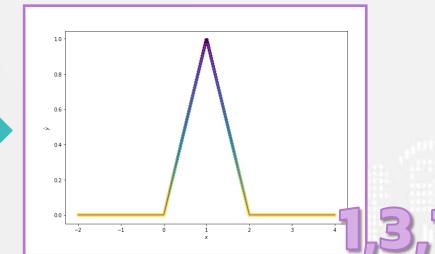
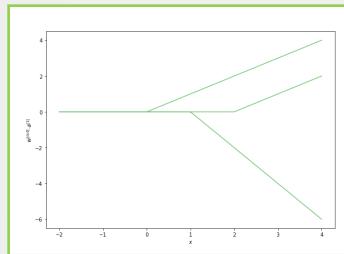
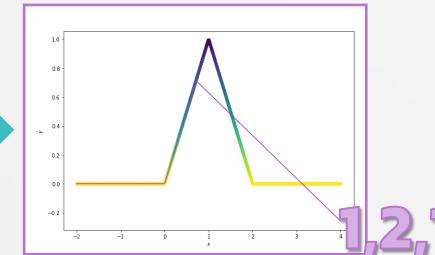
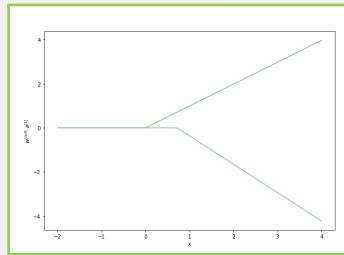
Example 3.1 (1,2,1)



Example 3.2 (1,3,1)

Neural Network Interpretation

Regression



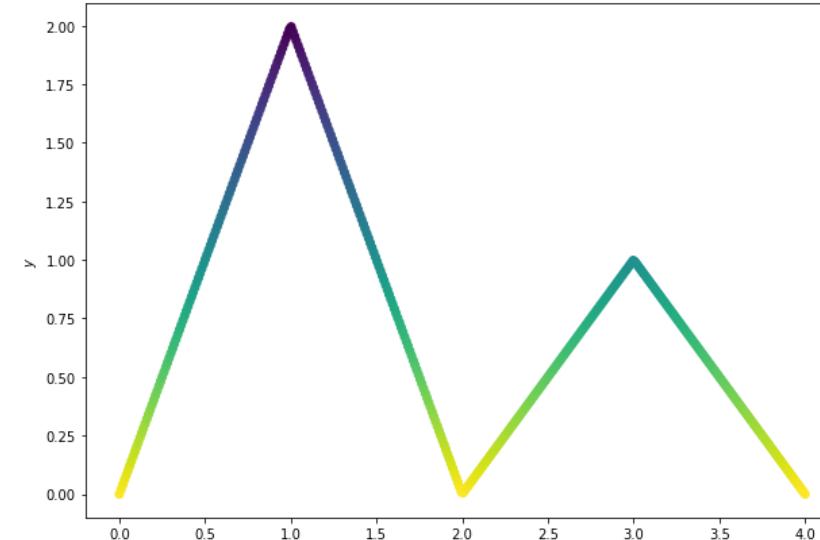
ข้อสรุปของ Example 3

- ผลของการการประกอบกันของ function ย่อย หลังผ่าน activation function นั้นจะยัง คงเหลือต่อไปในตลอดทั้งช่วงของ space

Neural Network Interpretation

Regression

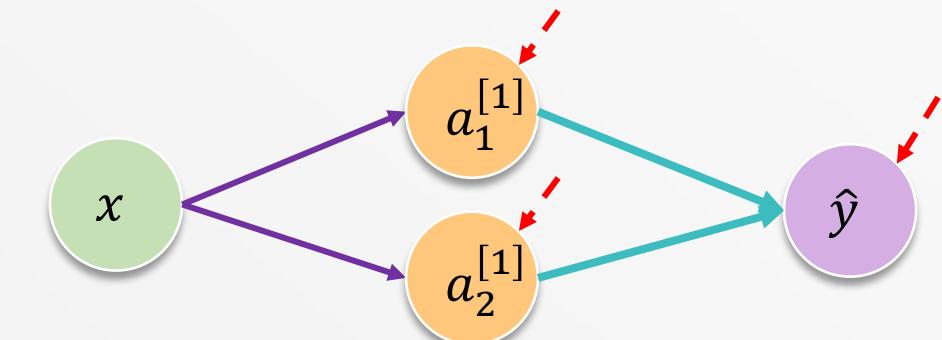
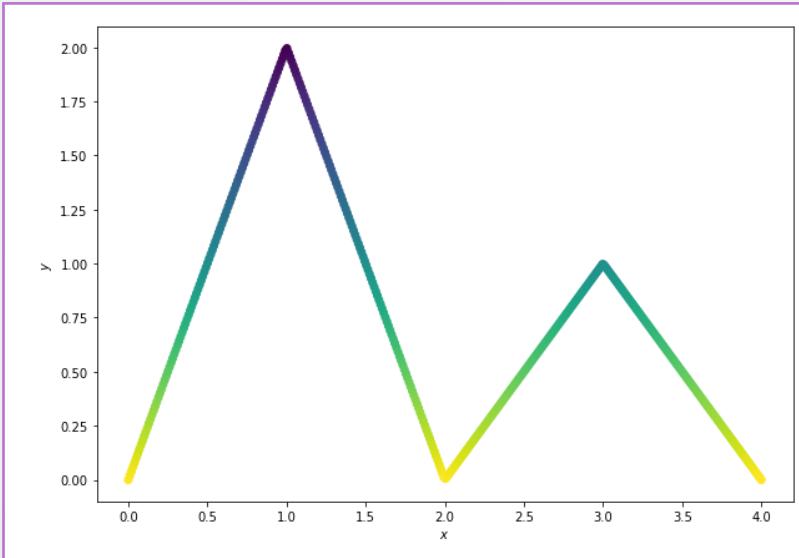
Example 4



Neural Network Interpretation

Regression

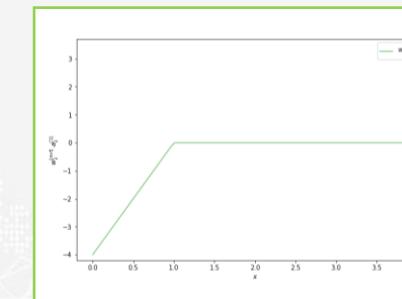
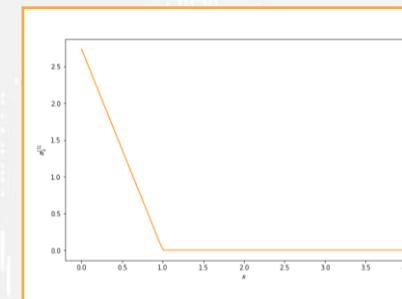
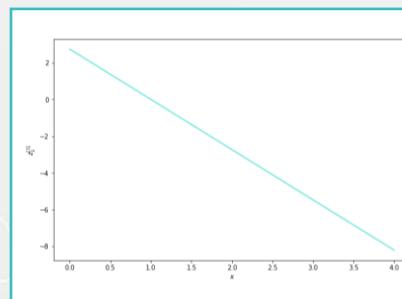
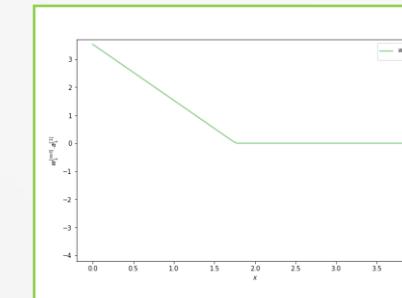
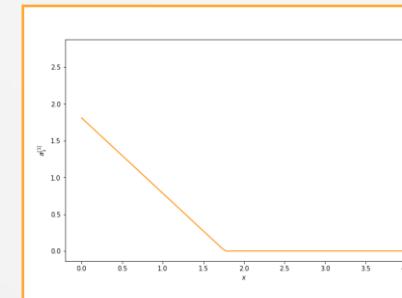
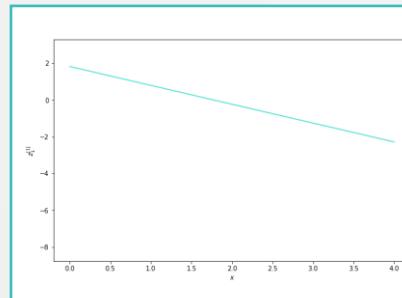
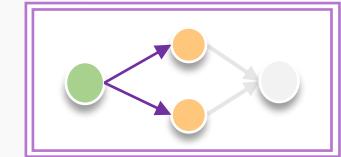
Example 4.1



Neural Network Interpretation

Regression

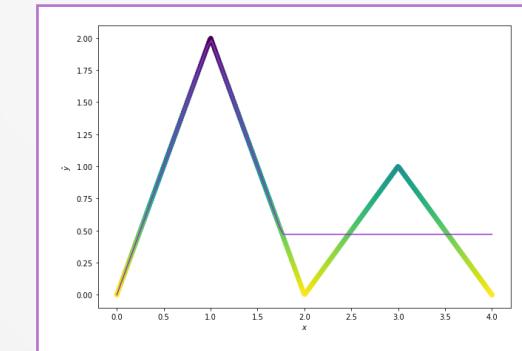
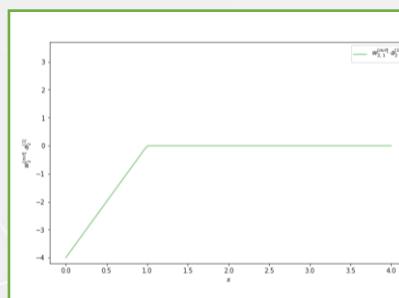
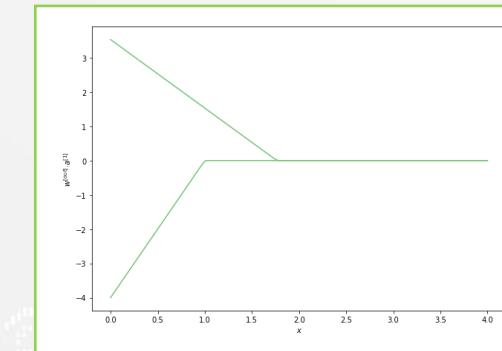
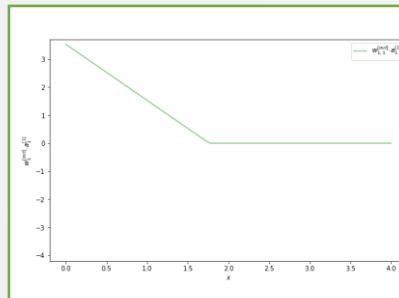
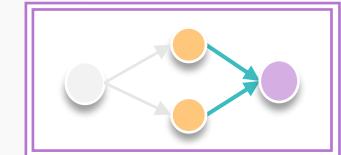
Example 4.1



Neural Network Interpretation

Regression

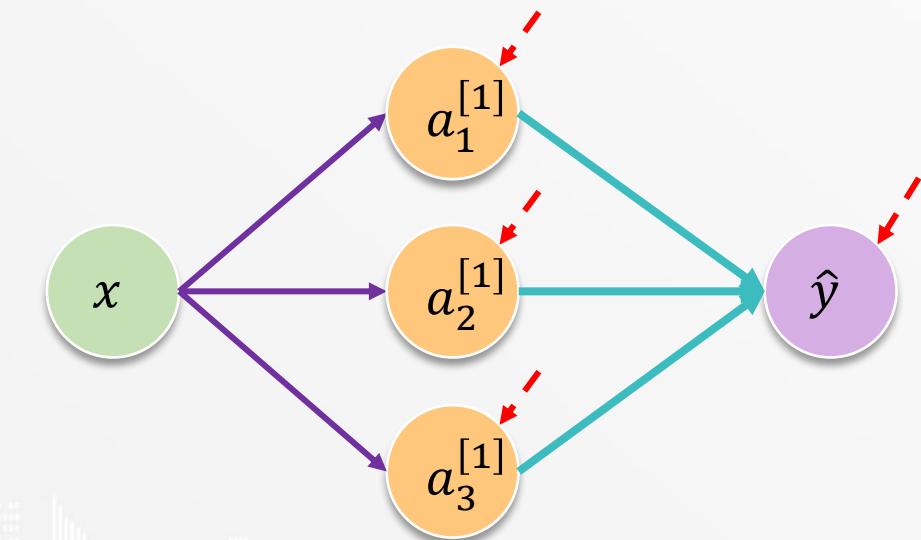
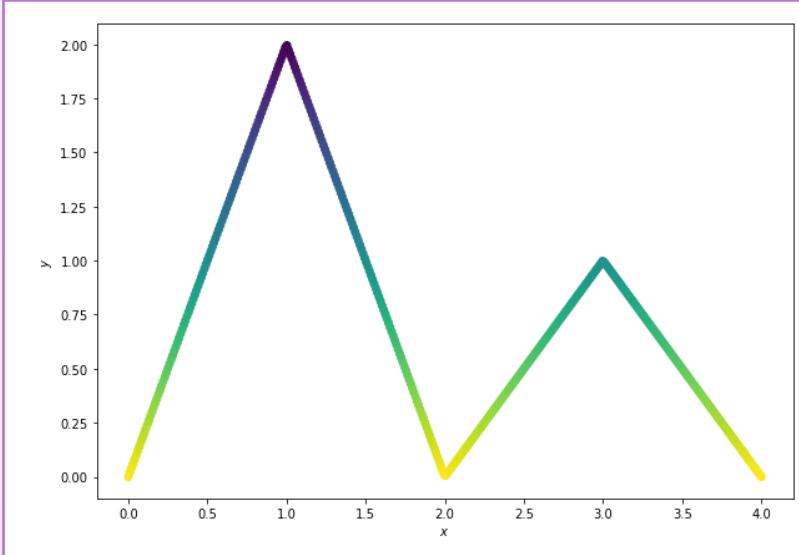
Example 4.1



Neural Network Interpretation

Regression

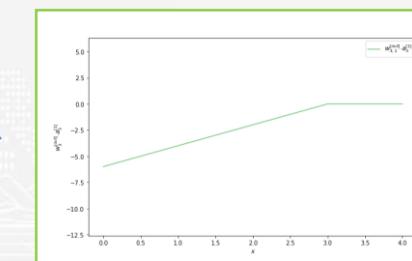
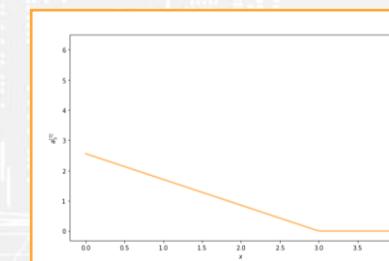
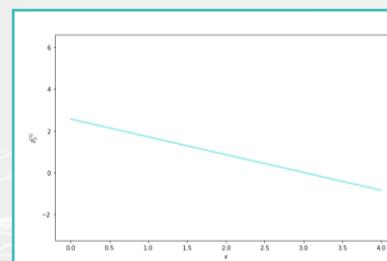
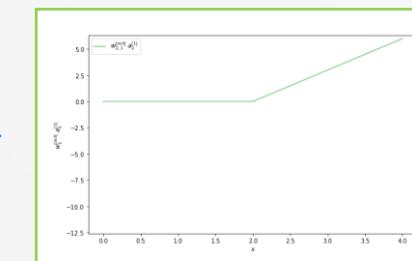
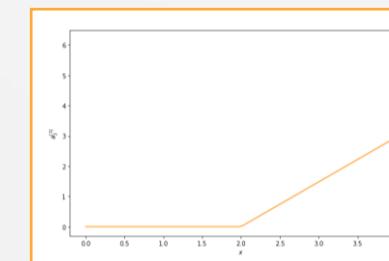
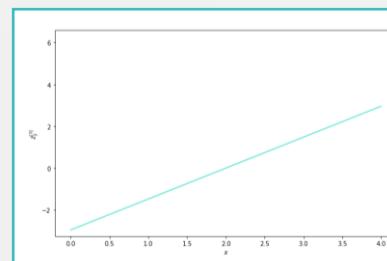
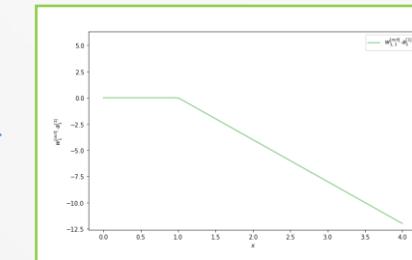
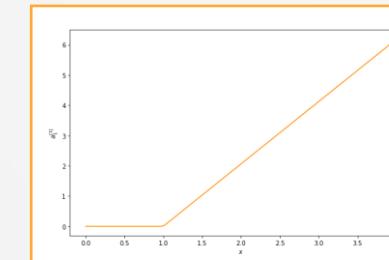
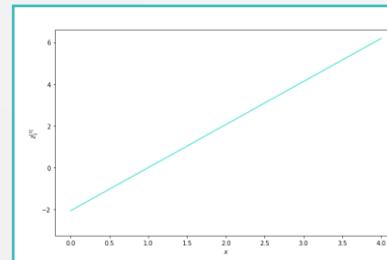
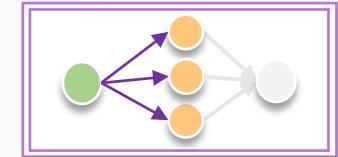
Example 4.2



Neural Network Interpretation

Regression

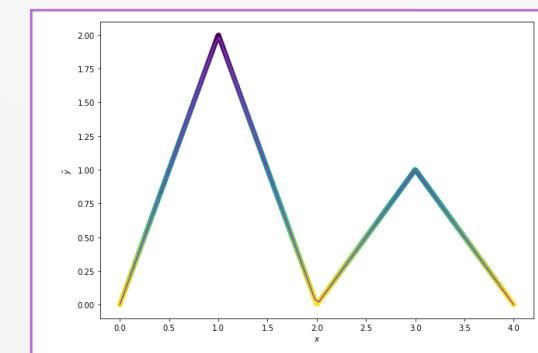
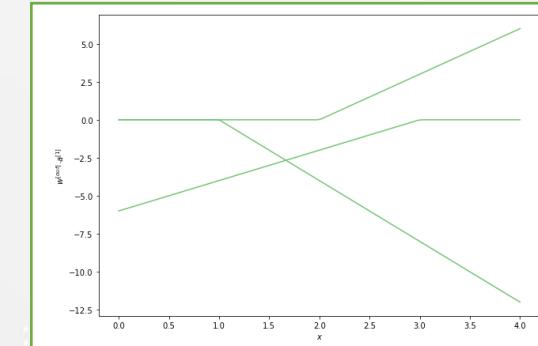
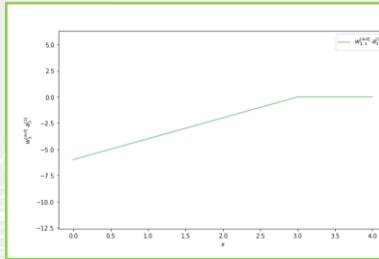
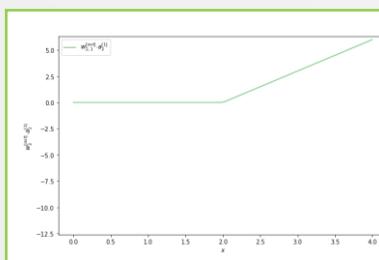
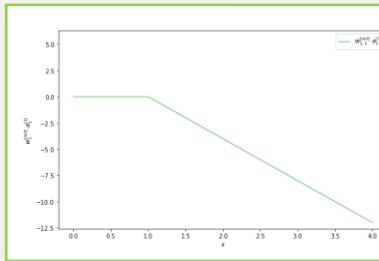
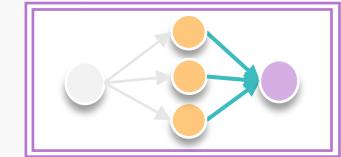
Example 4.2



Neural Network Interpretation

Regression

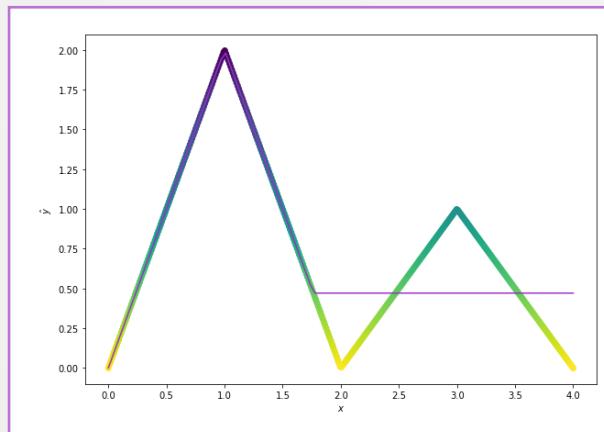
Example 4.2



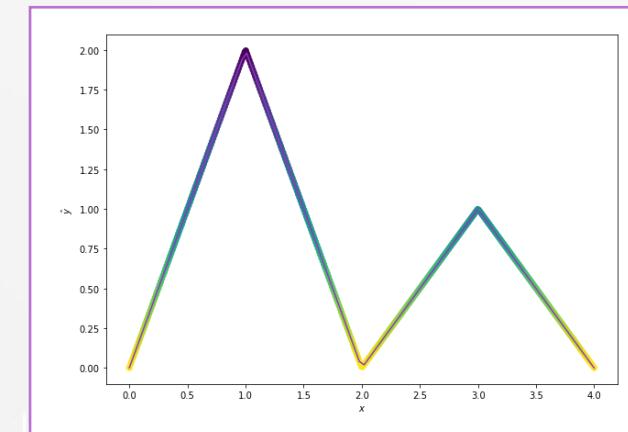
Neural Network Interpretation

Regression

กราฟแสดงค่า predicted ของ Example 4



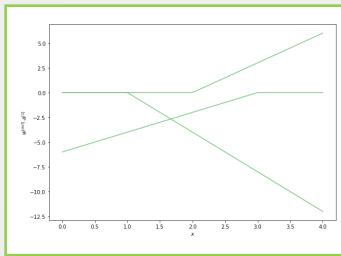
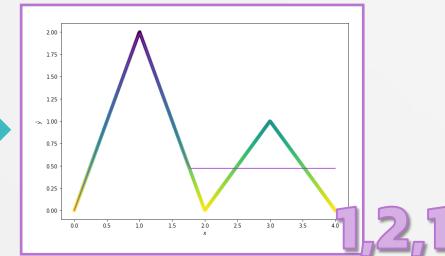
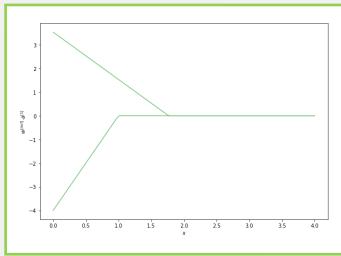
Example 4.1 (1,2,1)



Example 4.2 (1,3,1)

Neural Network Interpretation

Regression



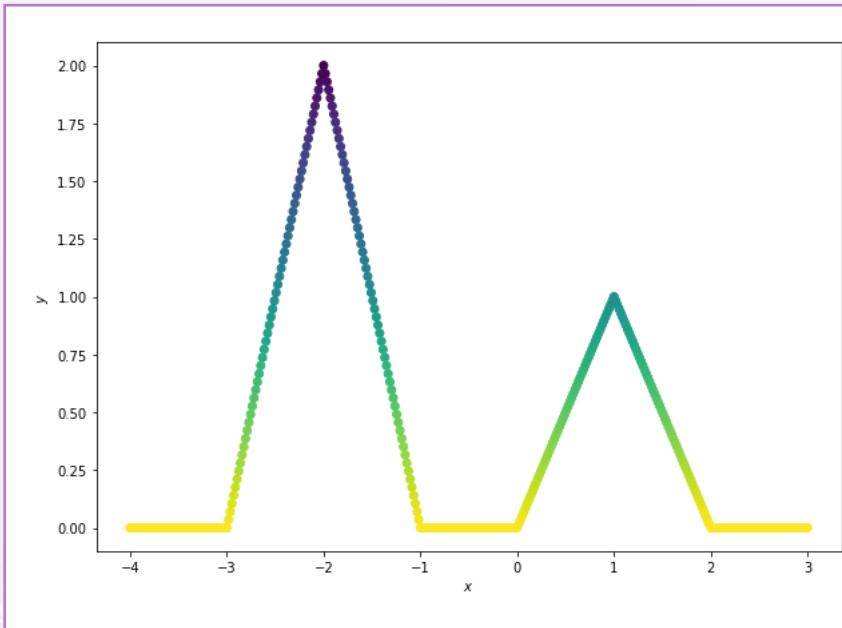
ข้อสรุปของ Example 4

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

Neural Network Interpretation

Regression

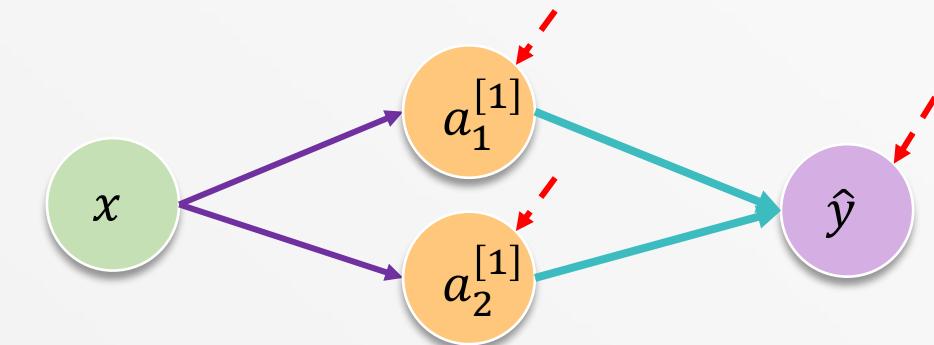
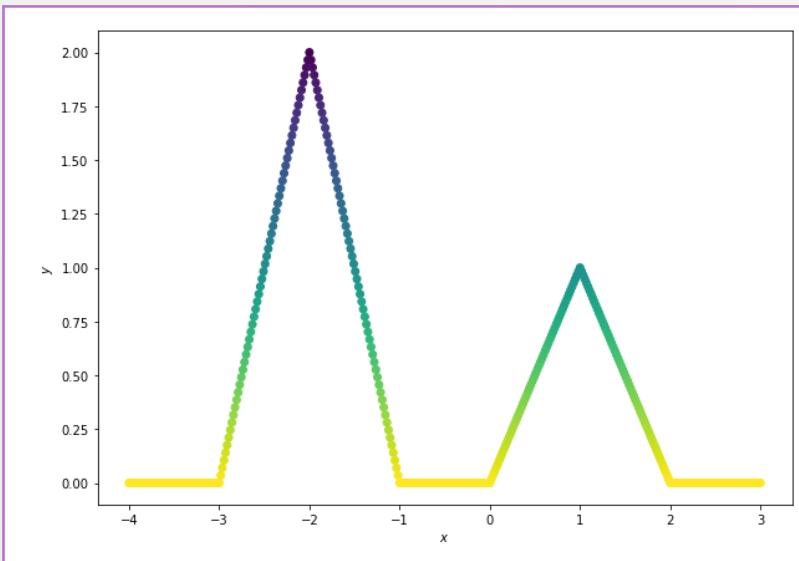
Example 5



Neural Network Interpretation

Regression

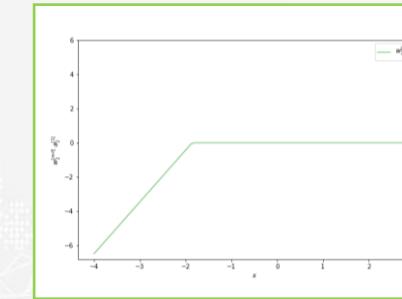
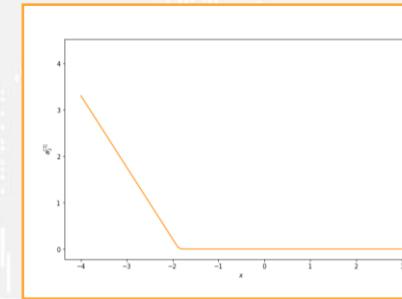
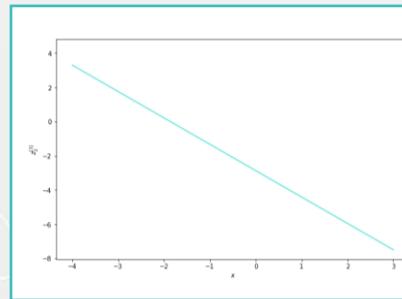
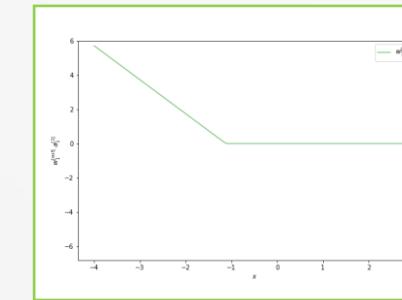
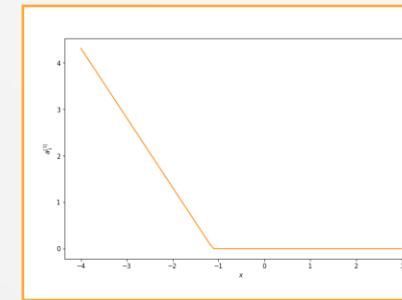
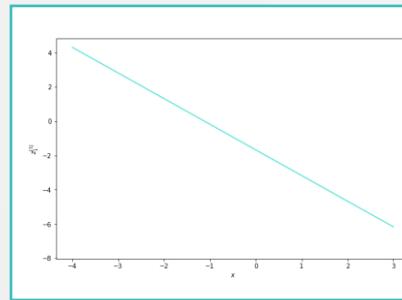
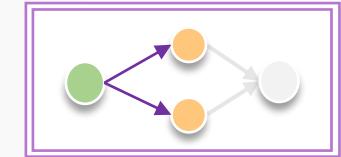
Example 5.1



Neural Network Interpretation

Regression

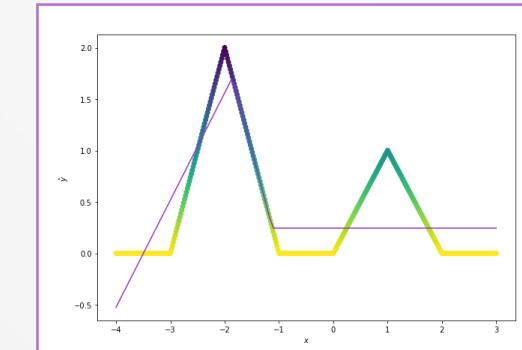
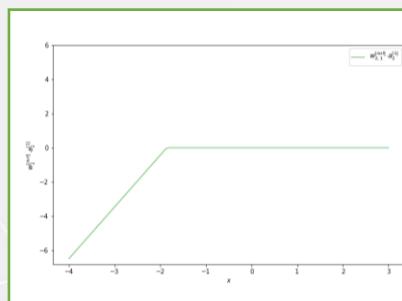
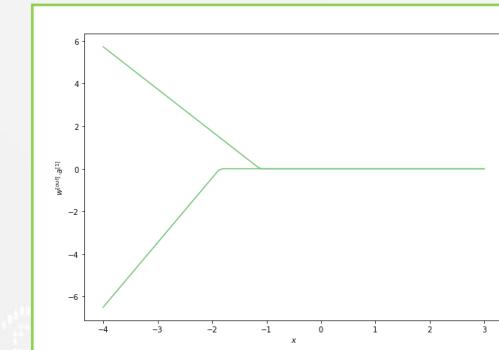
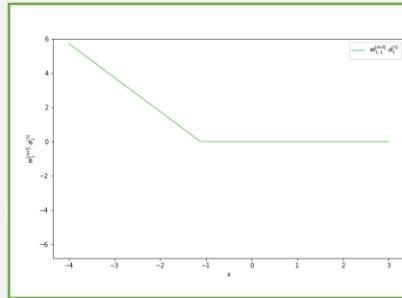
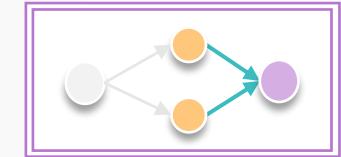
Example 5.1



Neural Network Interpretation

Regression

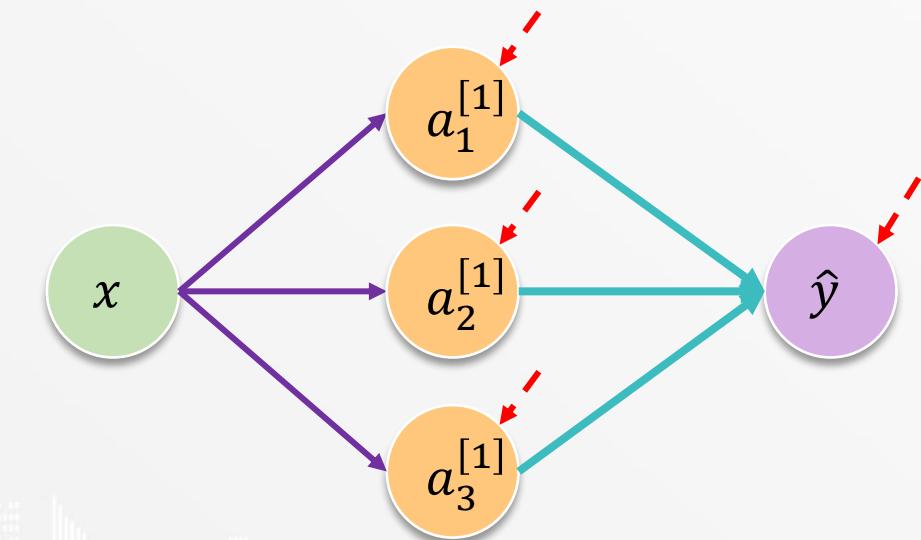
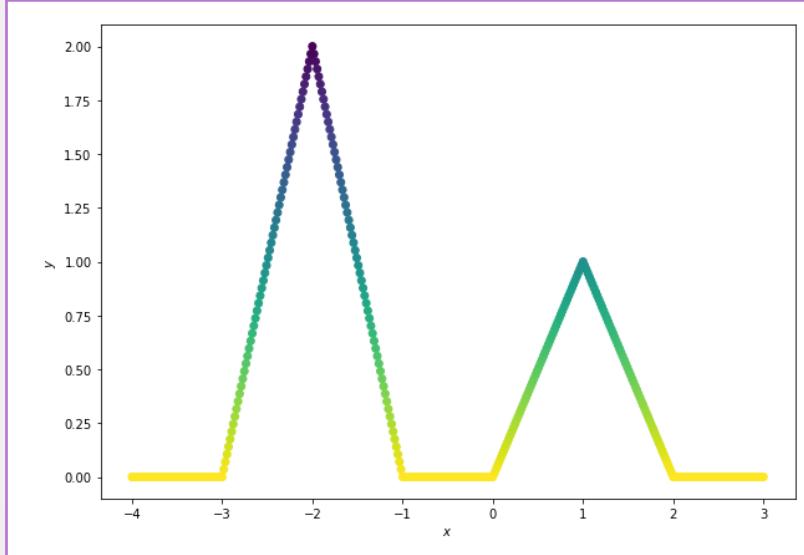
Example 5.1



Neural Network Interpretation

Regression

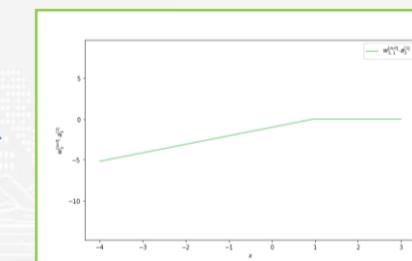
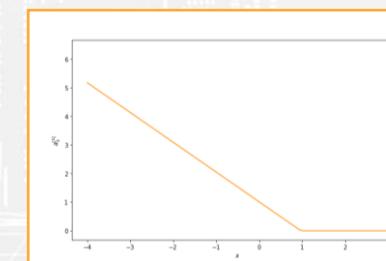
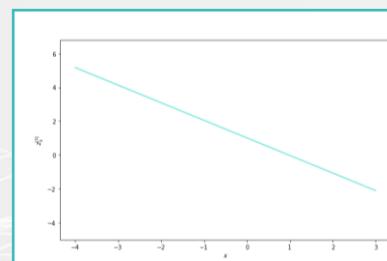
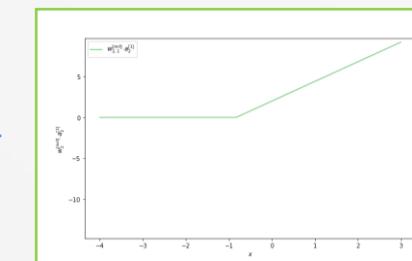
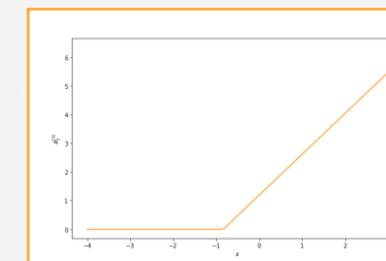
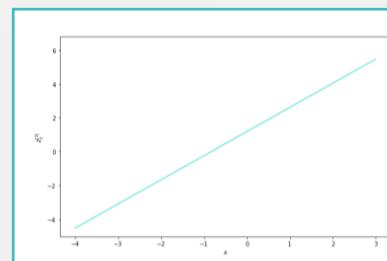
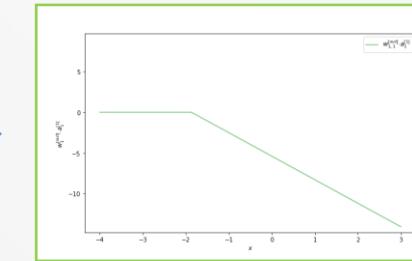
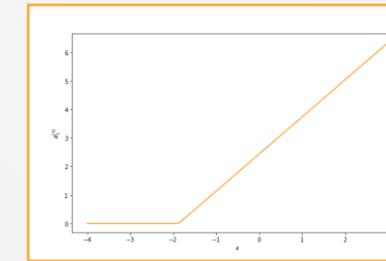
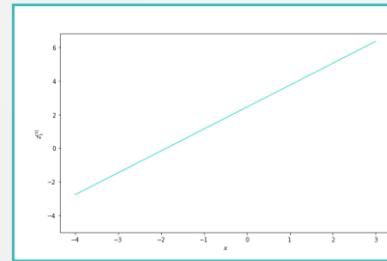
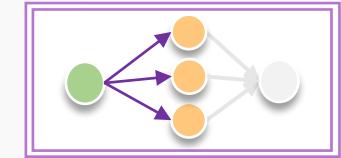
Example 5.2



Neural Network Interpretation

Regression

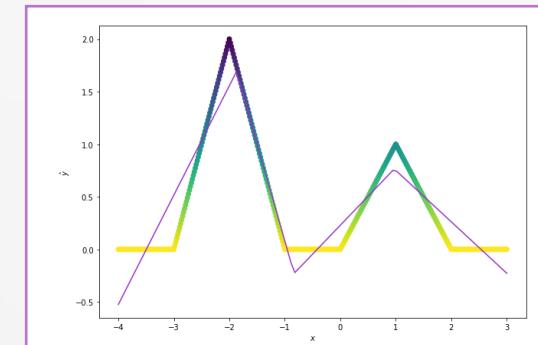
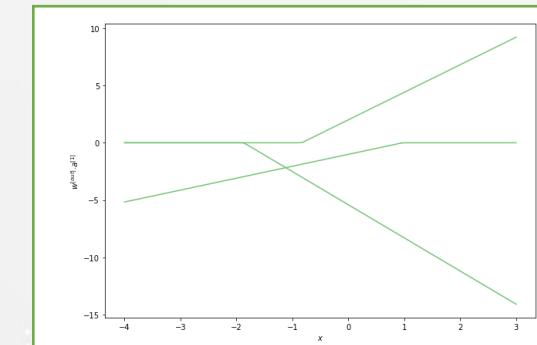
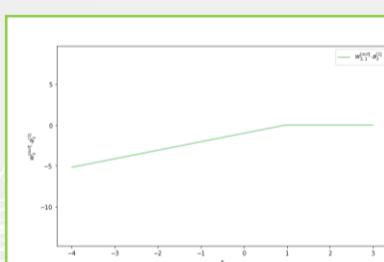
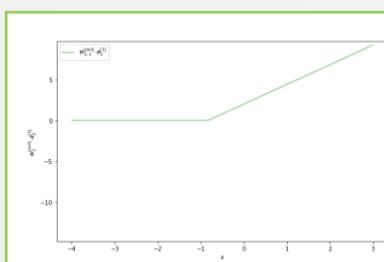
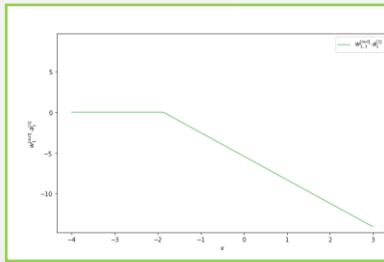
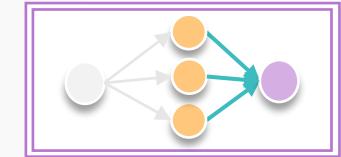
Example 5.2



Neural Network Interpretation

Regression

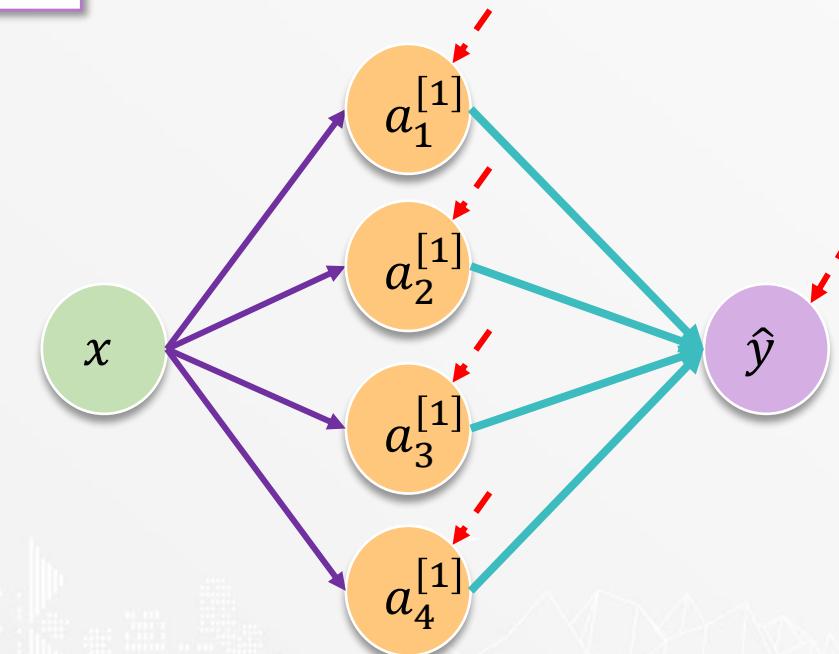
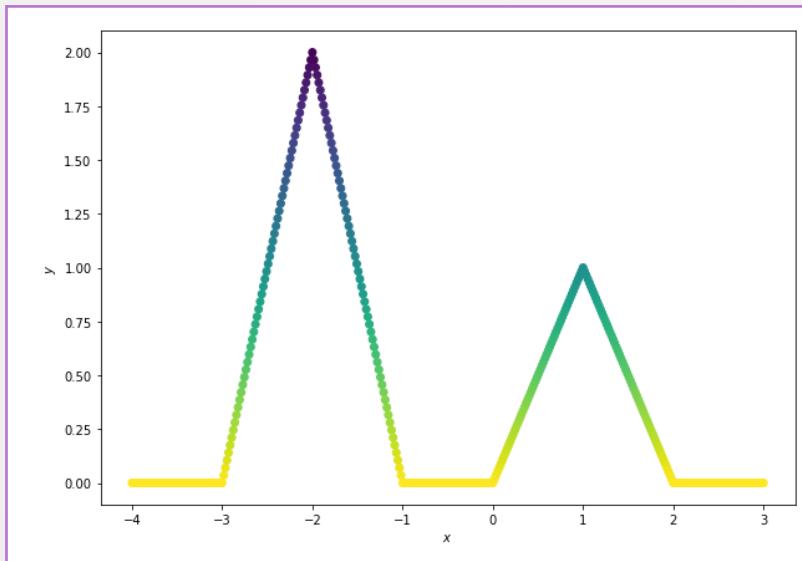
Example 5.2



Neural Network Interpretation

Regression

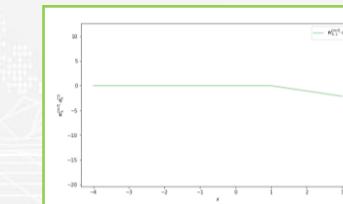
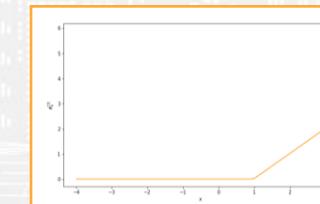
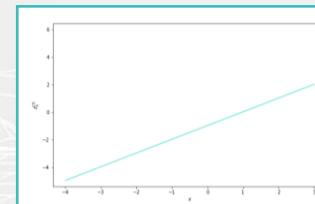
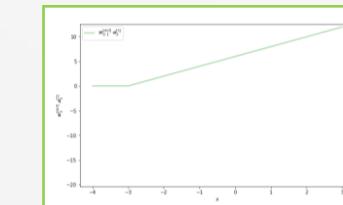
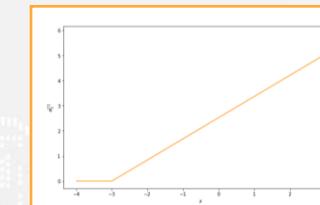
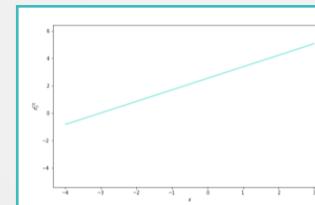
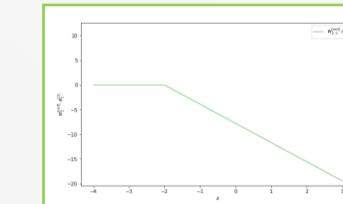
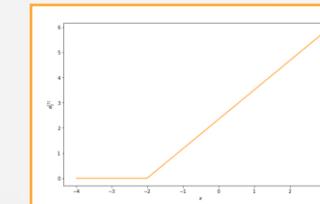
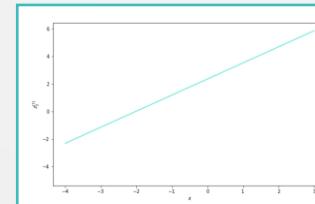
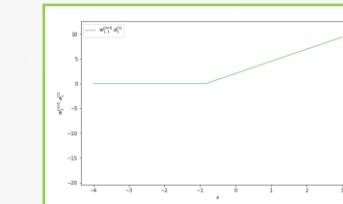
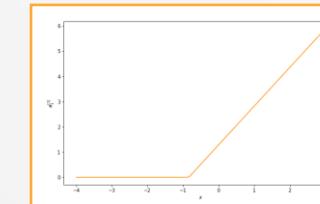
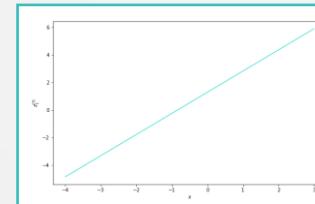
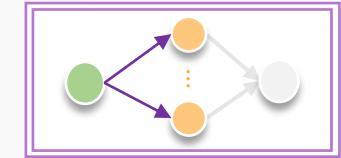
Example 5.3



Neural Network Interpretation

Regression

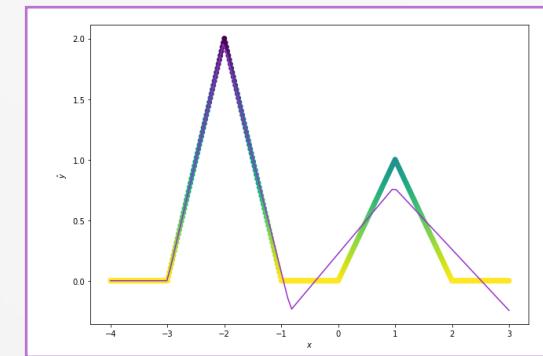
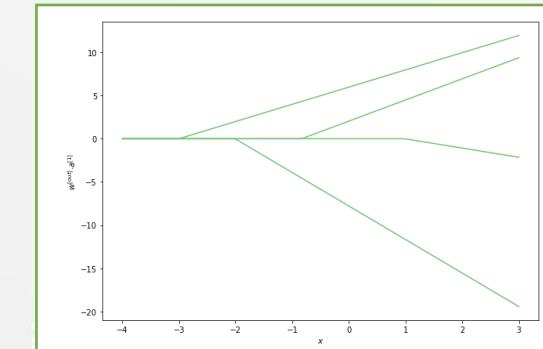
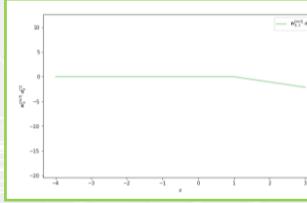
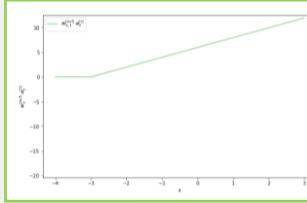
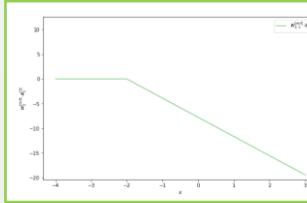
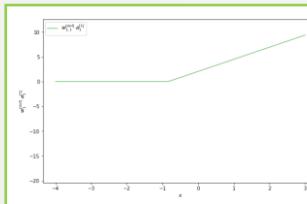
Example 5.3



Neural Network Interpretation

Regression

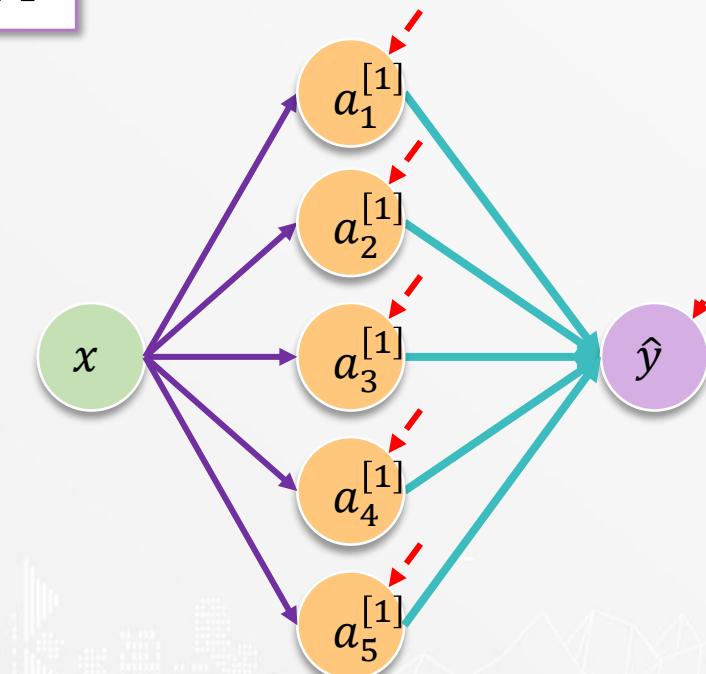
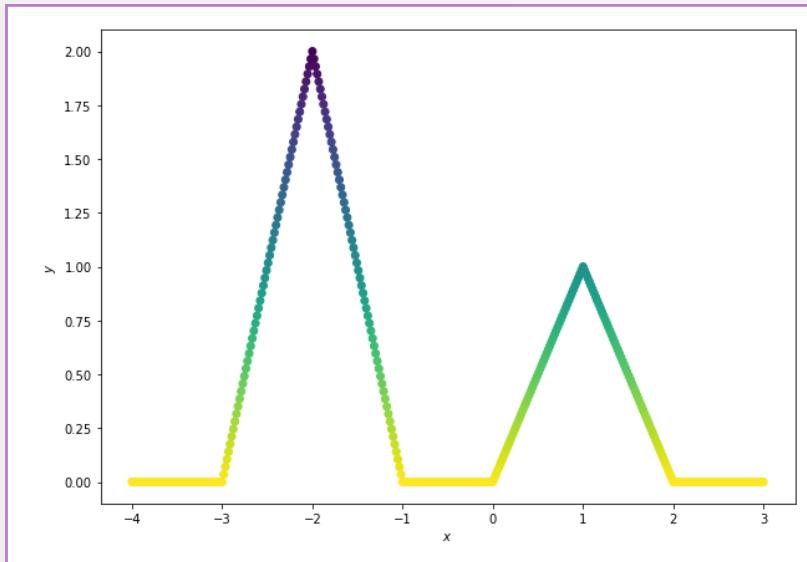
Example 5.3



Neural Network Interpretation

Regression

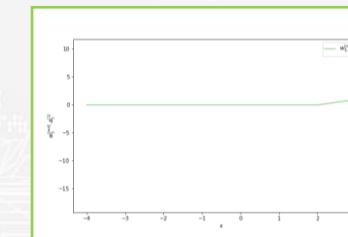
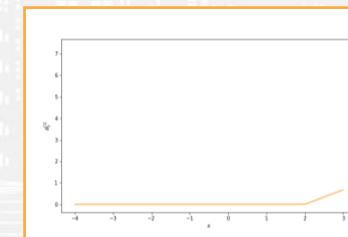
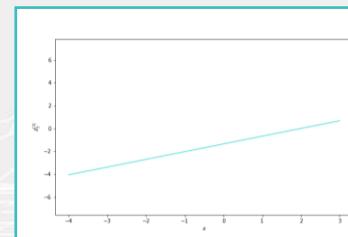
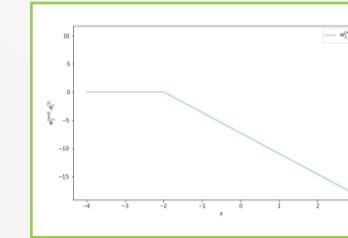
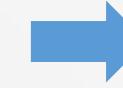
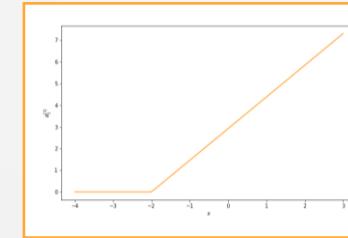
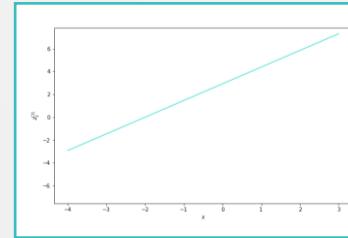
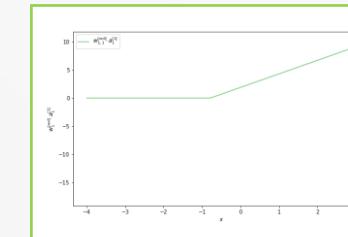
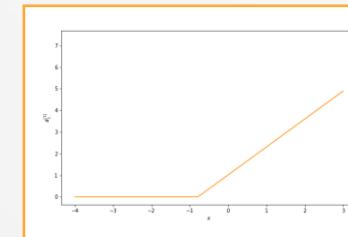
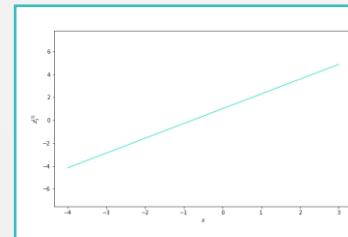
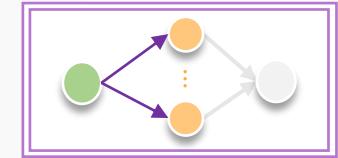
Example 5.4



Neural Network Interpretation

Regression

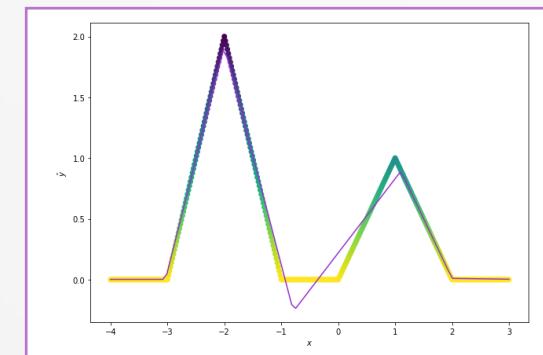
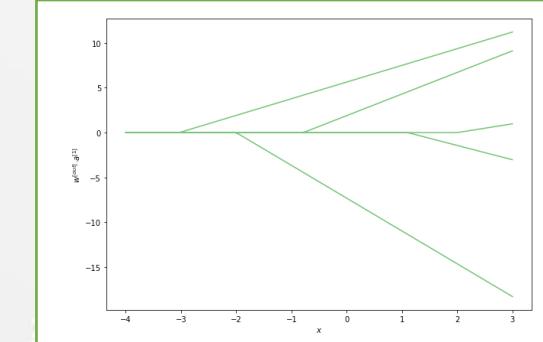
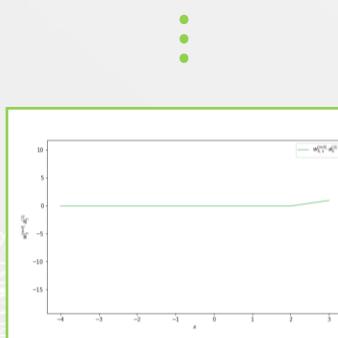
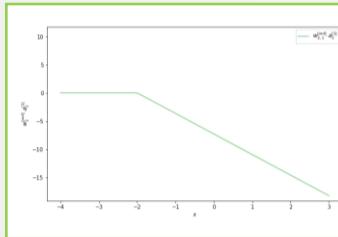
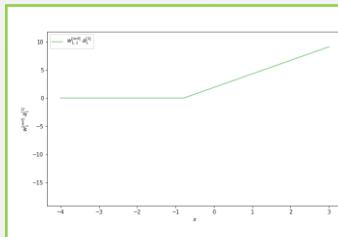
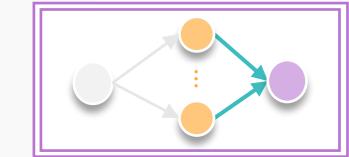
Example 5.4



Neural Network Interpretation

Regression

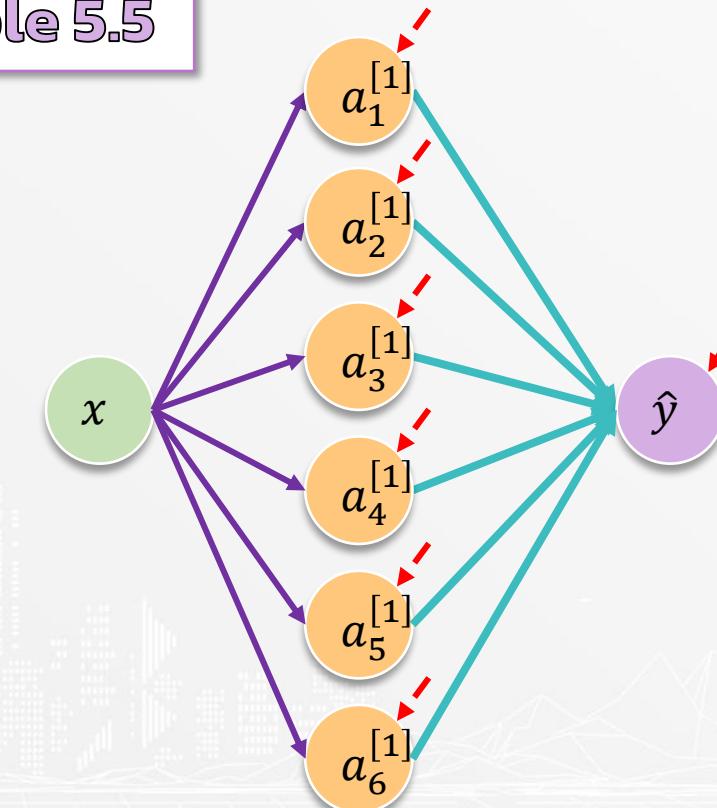
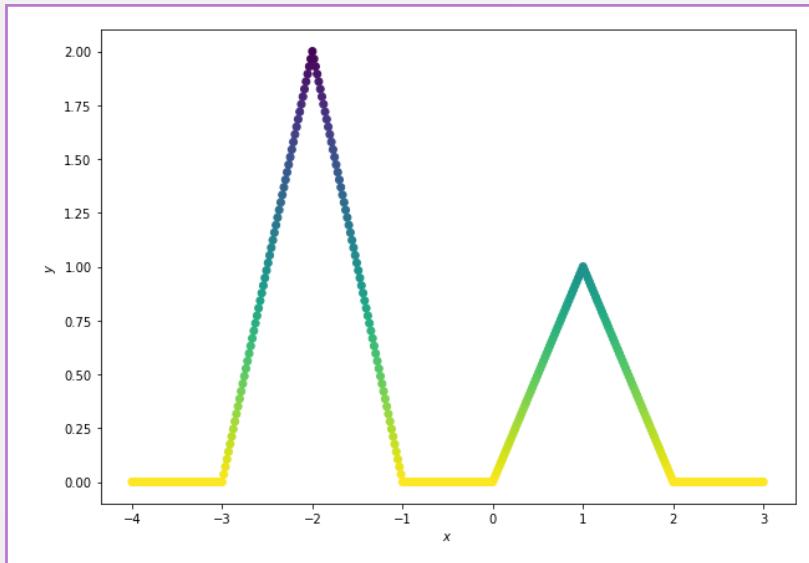
Example 5.4



Neural Network Interpretation

Regression

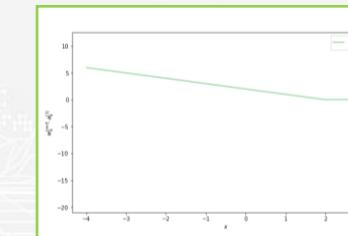
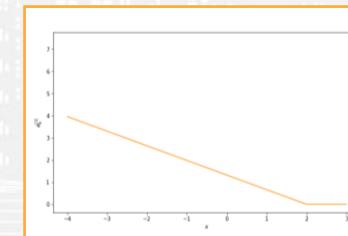
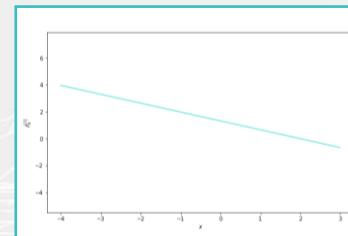
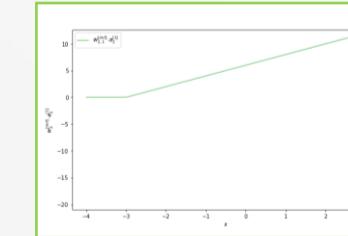
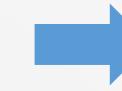
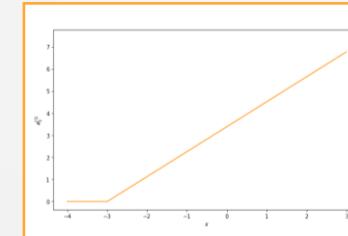
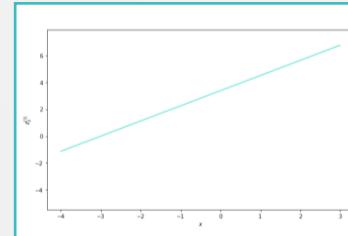
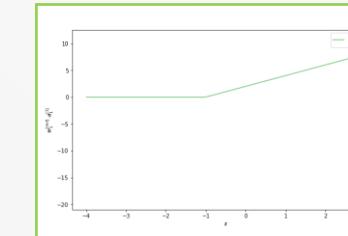
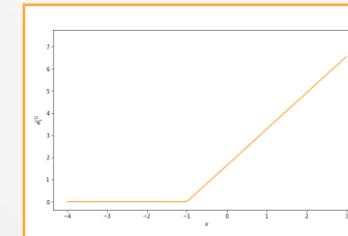
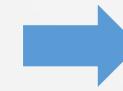
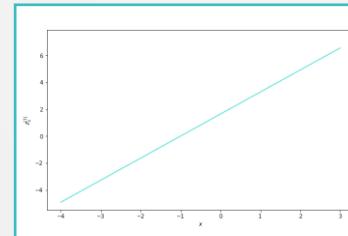
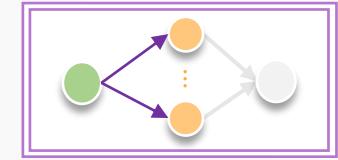
Example 5.5



Neural Network Interpretation

Regression

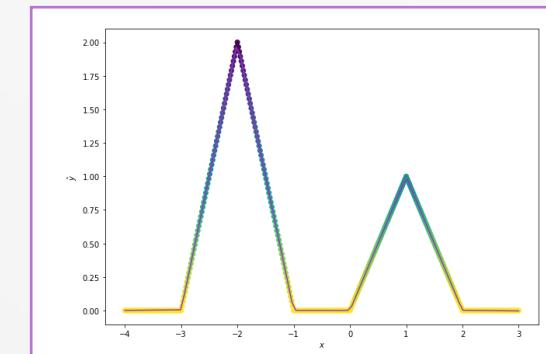
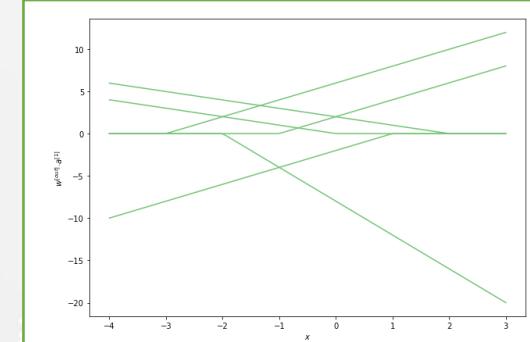
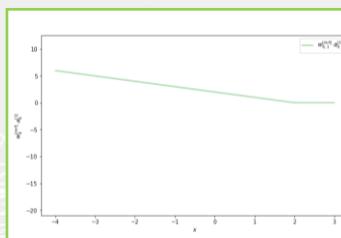
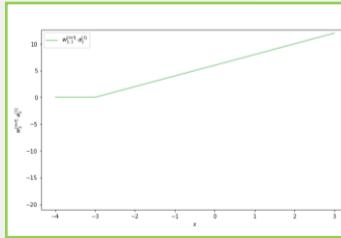
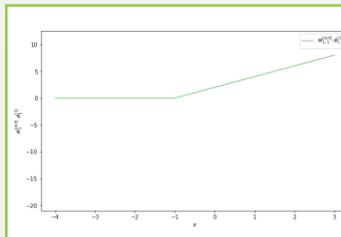
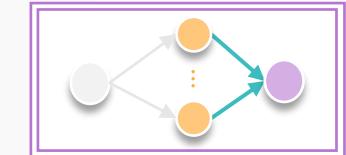
Example 5.5



Neural Network Interpretation

Regression

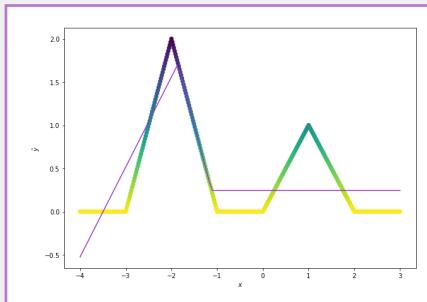
Example 5.5



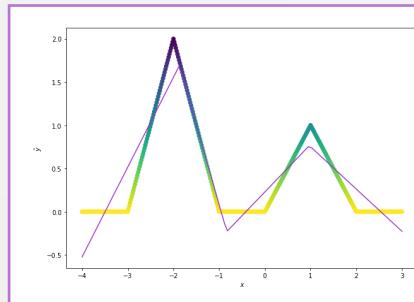
Neural Network Interpretation

Regression

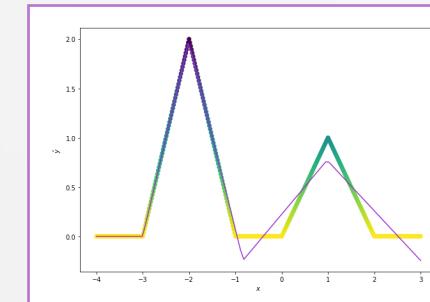
กราฟแสดงค่า predicted ของ Example 5



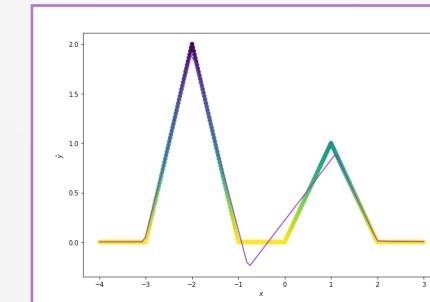
Example 5.1 (1,2,1)



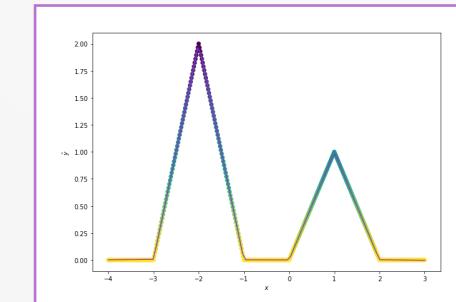
Example 5.2 (1,3,1)



Example 5.3 (1,4,1)



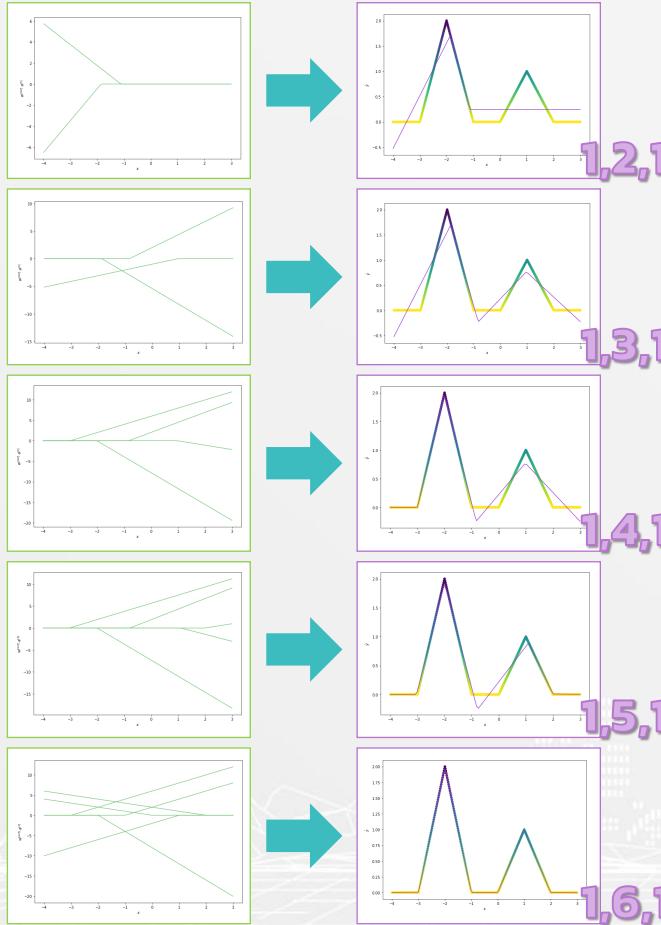
Example 5.4 (1,5,1)



Example 5.5 (1,6,1)

Neural Network Interpretation

Regression



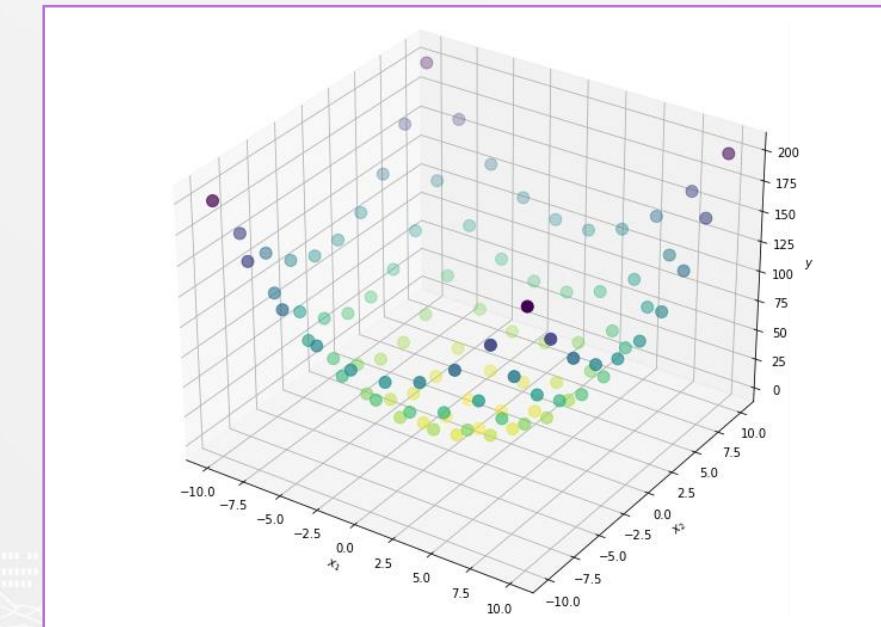
ข้อสรุปของ Example 5

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

Neural Network Interpretation

Regression

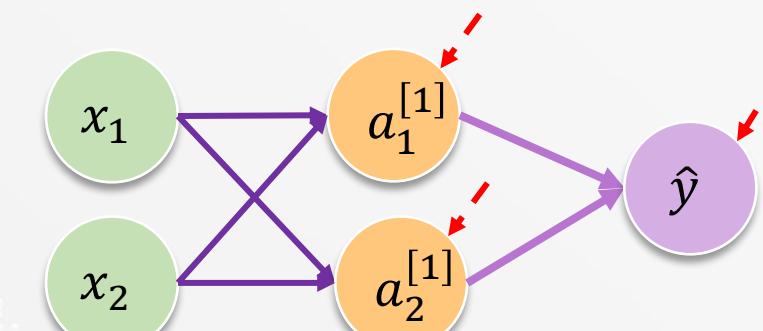
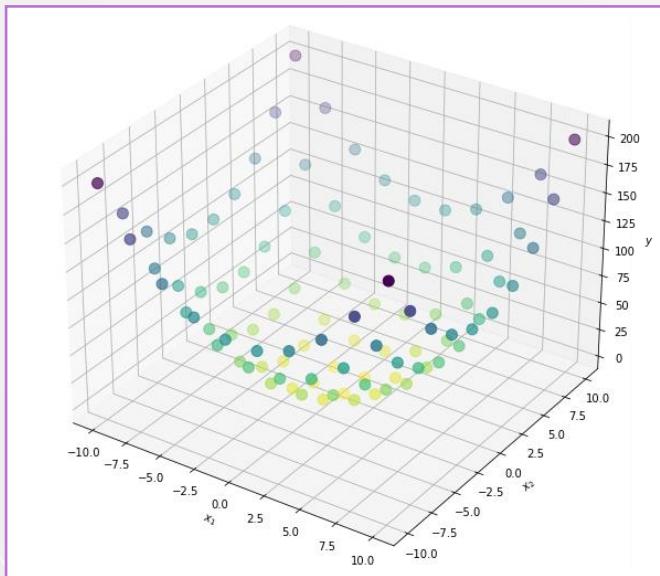
Example 6



Neural Network Interpretation

Regression

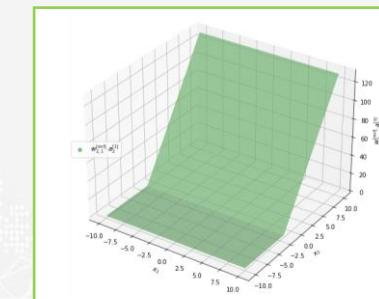
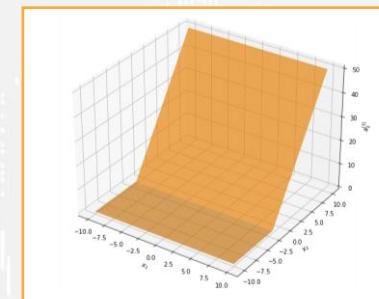
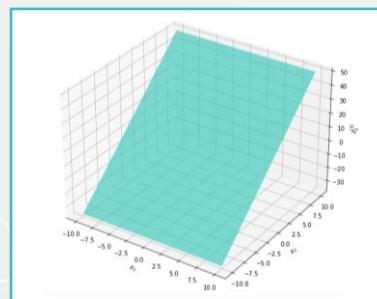
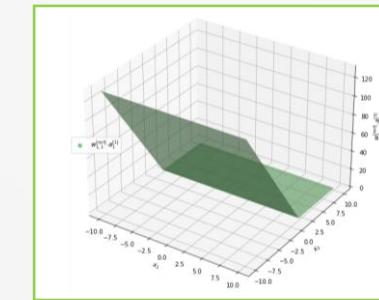
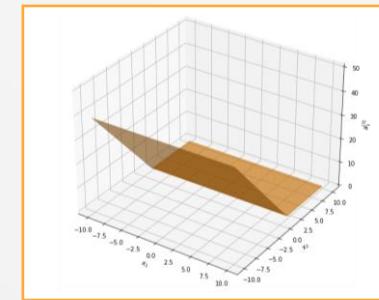
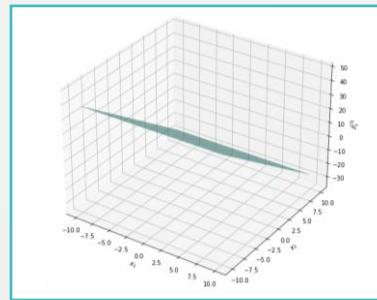
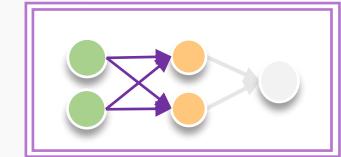
Example 6.1



Neural Network Interpretation

Regression

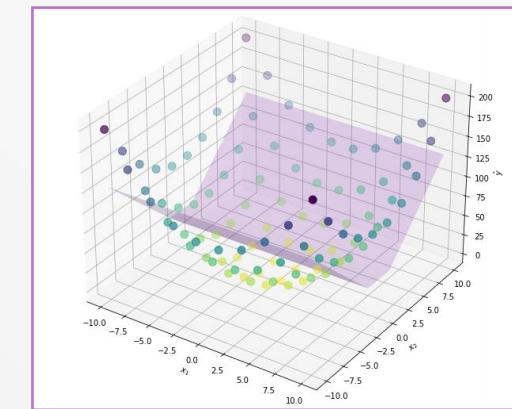
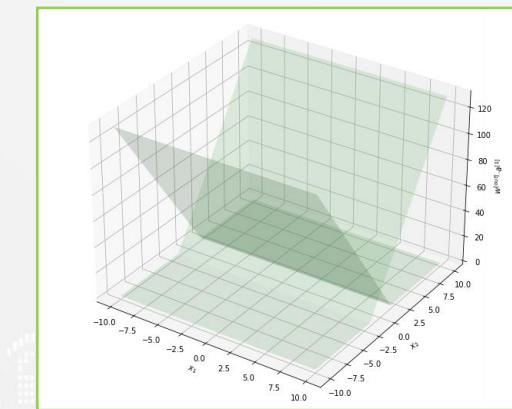
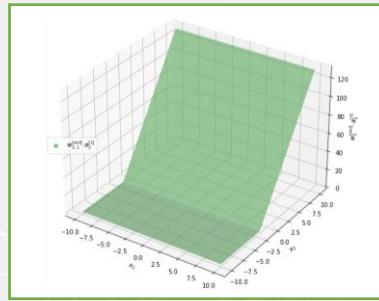
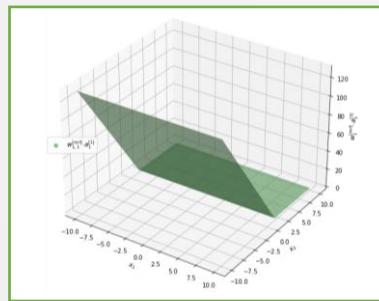
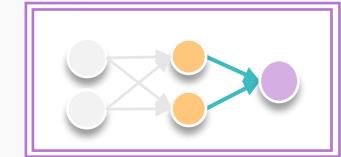
Example 6.1



Neural Network Interpretation

Regression

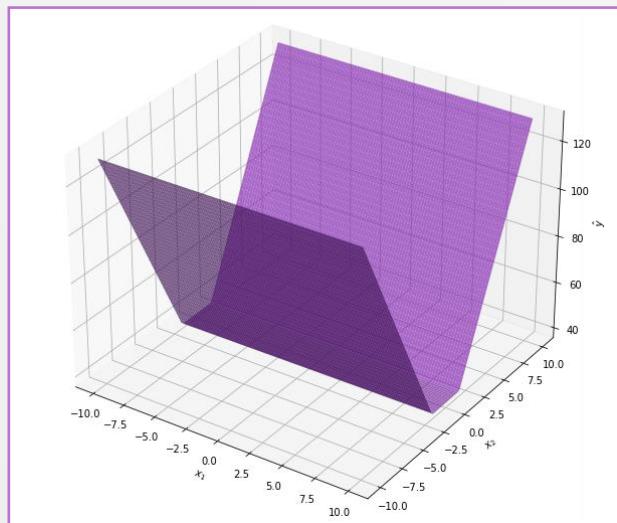
Example 6.1



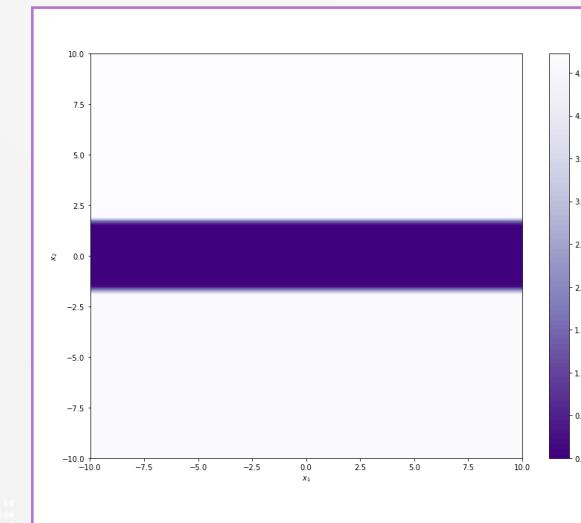
Neural Network Interpretation

Regression

Example 6.1



กราฟแสดงค่า predicted

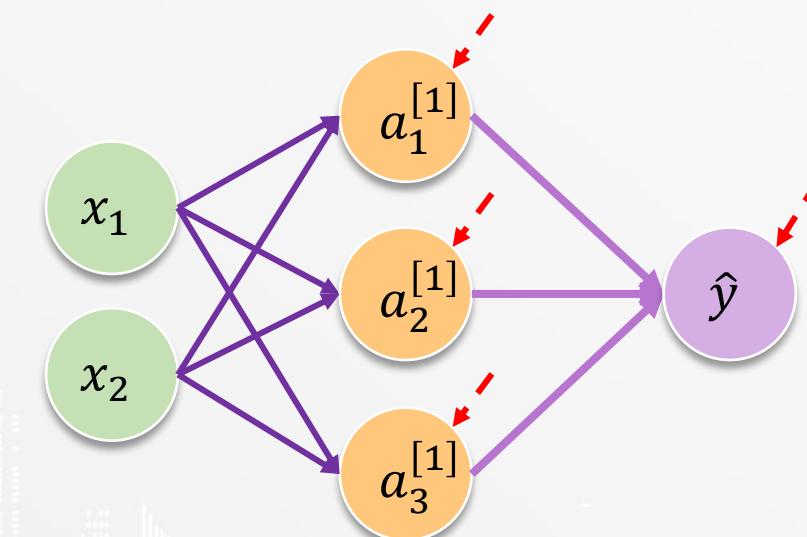
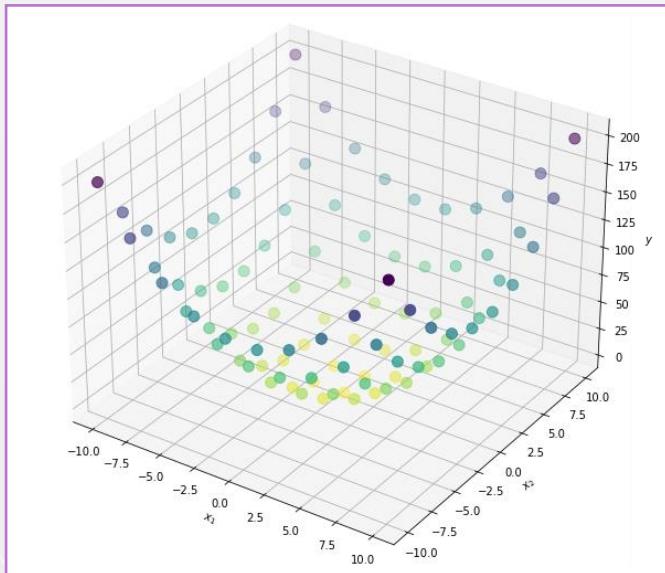


Contour plot ของ predicted

Neural Network Interpretation

Regression

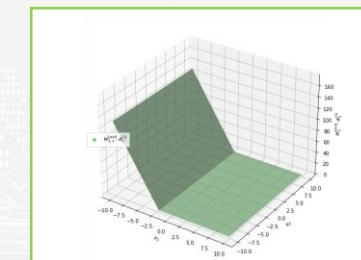
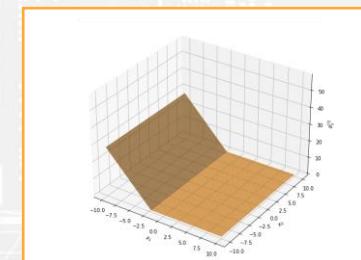
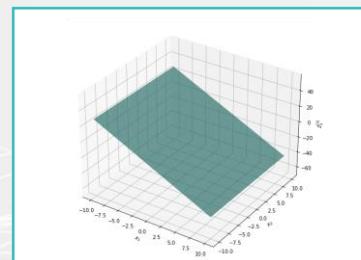
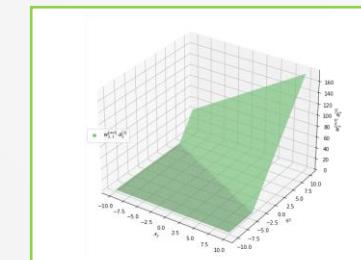
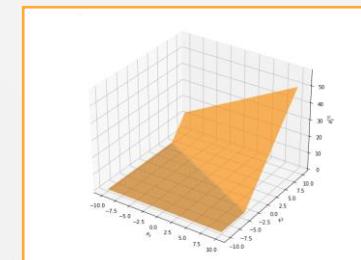
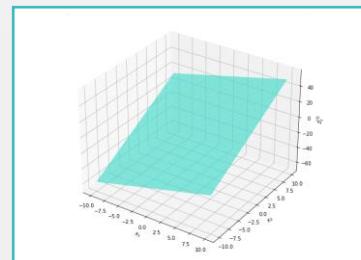
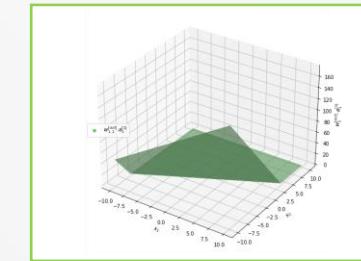
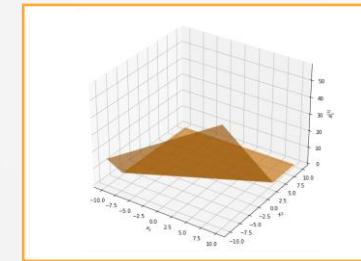
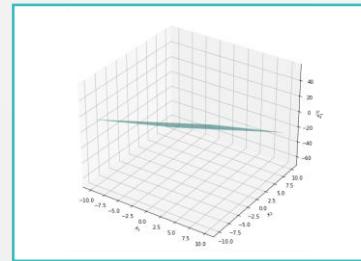
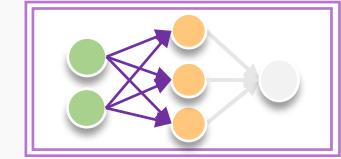
Example 6.2



Neural Network Interpretation

Regression

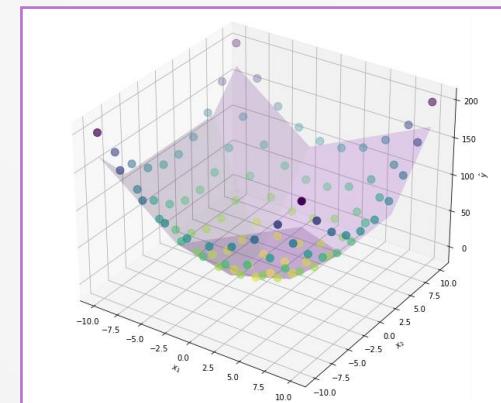
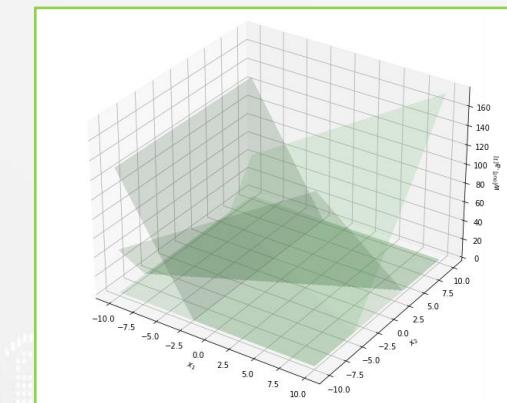
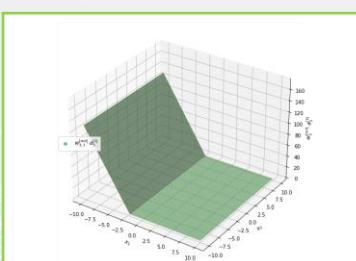
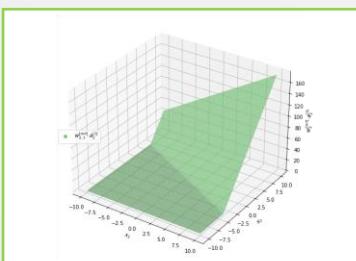
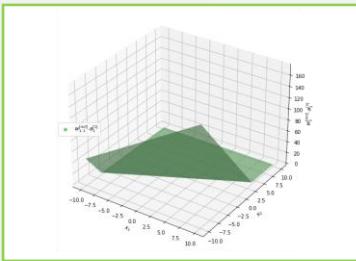
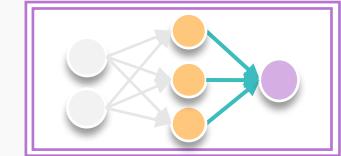
Example 6.2



Neural Network Interpretation

Regression

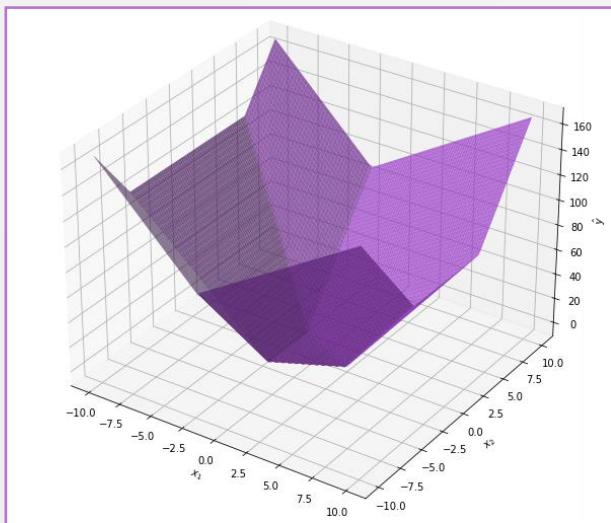
Example 6.2



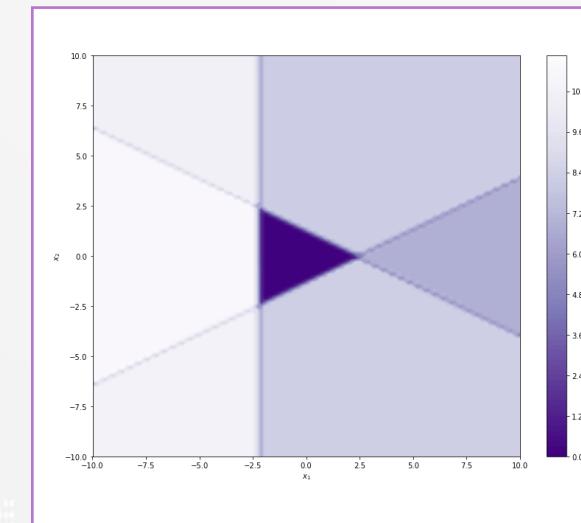
Neural Network Interpretation

Regression

Example 6.2



กราฟแสดงค่า predicted

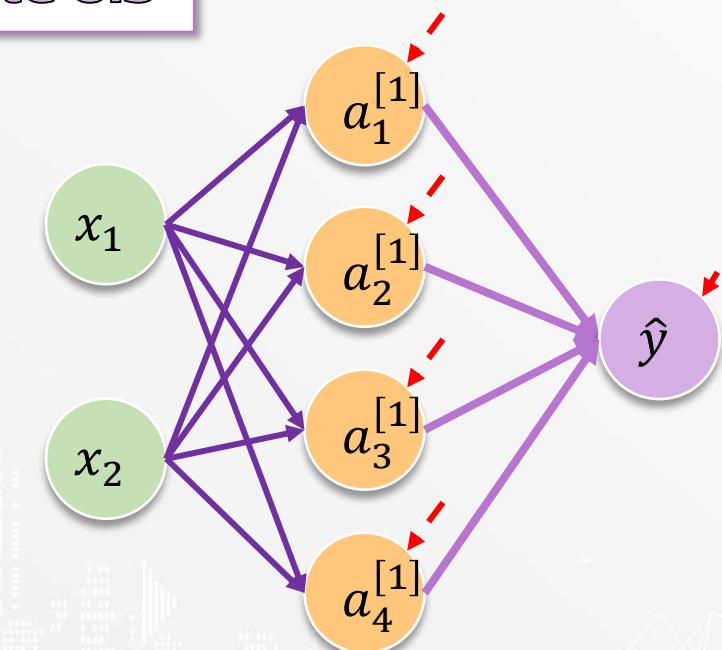
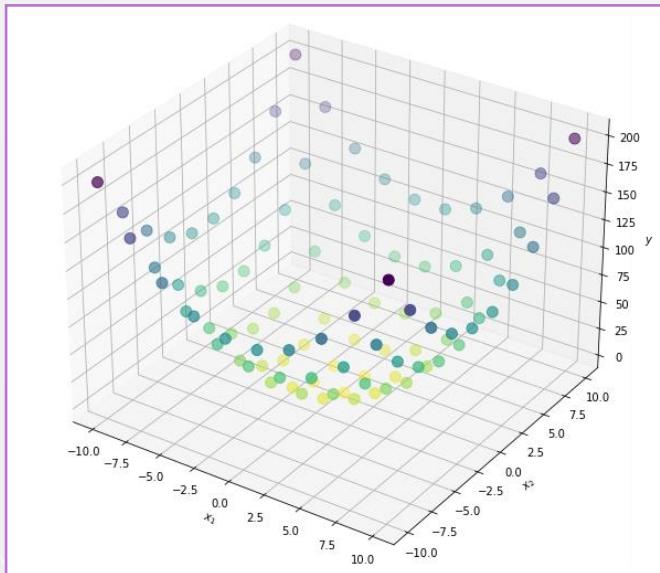


Contour plot ของ predicted

Neural Network Interpretation

Regression

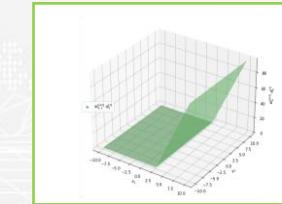
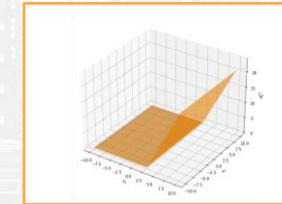
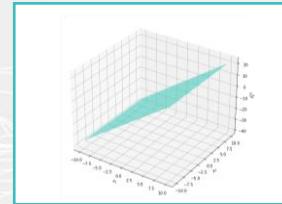
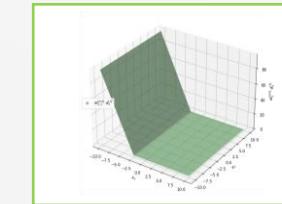
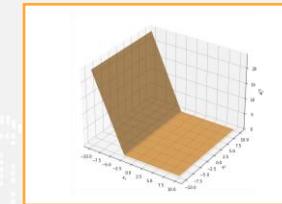
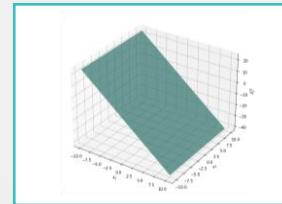
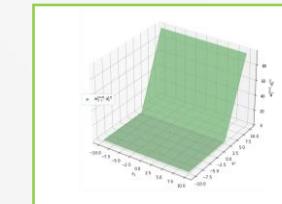
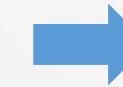
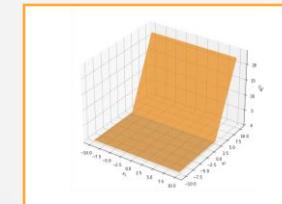
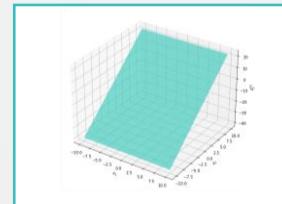
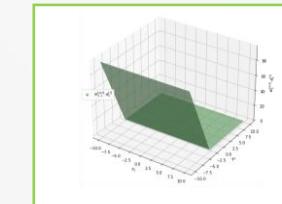
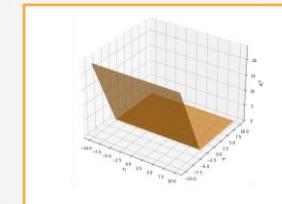
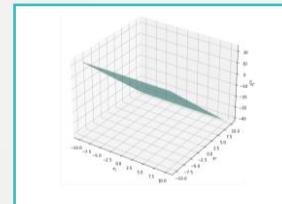
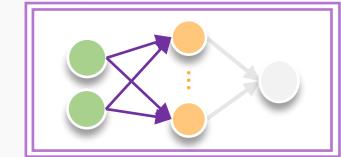
Example 6.3



Neural Network Interpretation

Regression

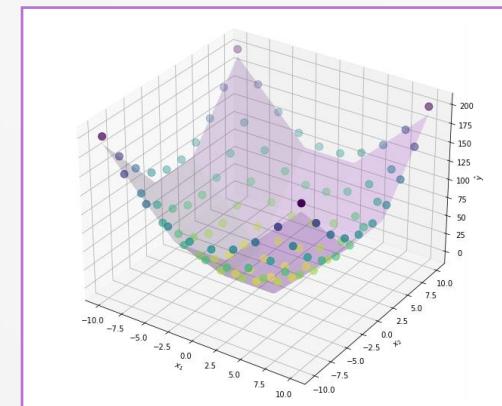
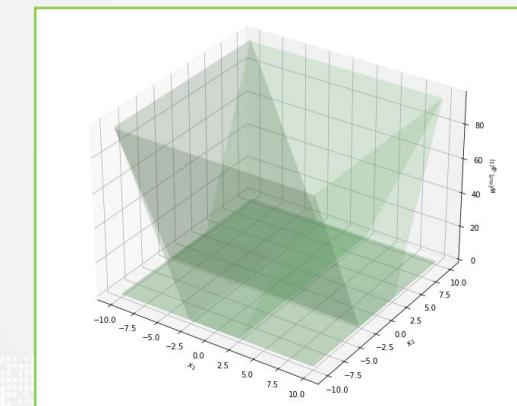
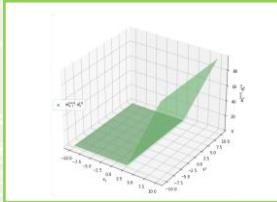
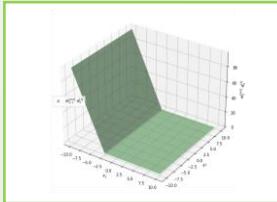
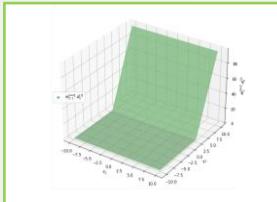
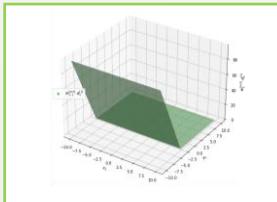
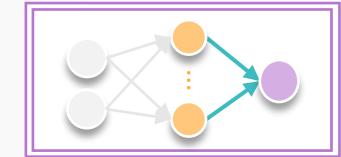
Example 6.3



Neural Network Interpretation

Regression

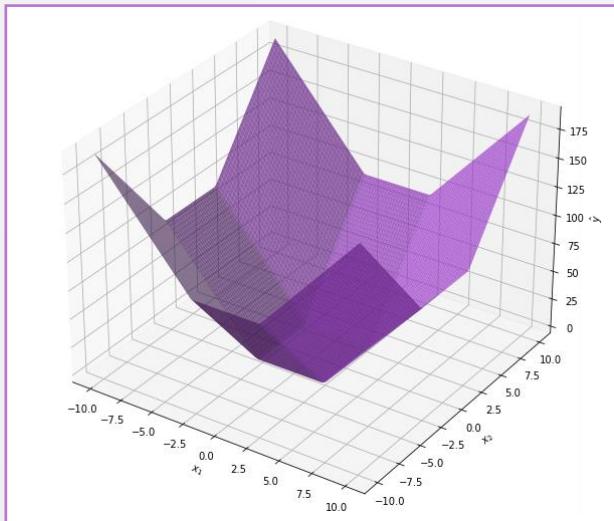
Example 6.3



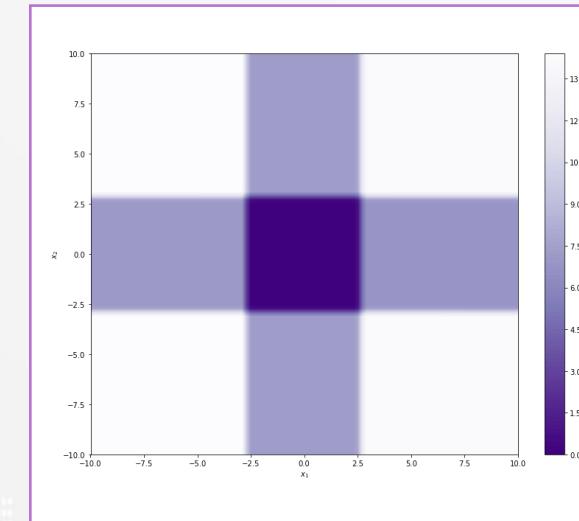
Neural Network Interpretation

Regression

Example 6.3



กราฟแสดงค่า predicted

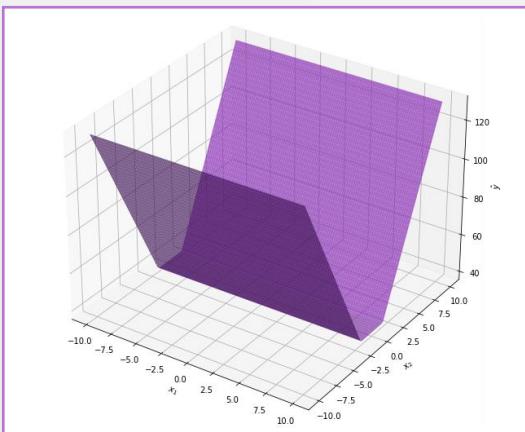


Contour plot ของ predicted

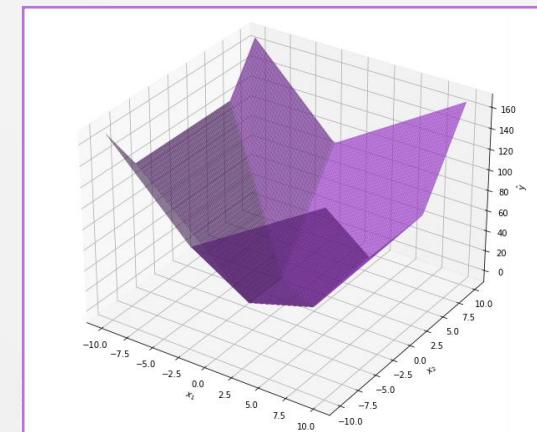
Neural Network Interpretation

Regression

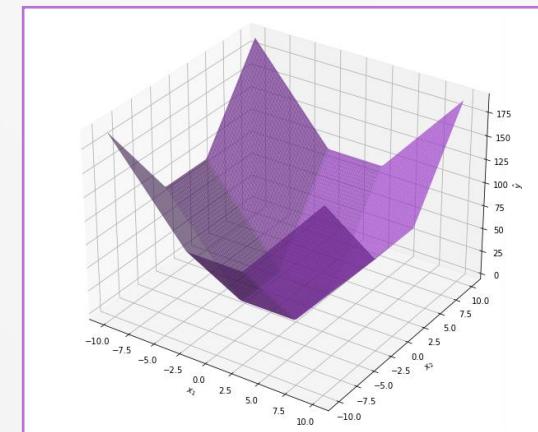
กราฟแสดงค่า predicted ของ Example 6



Example 6.1 (2,2,1)



Example 6.2 (2,3,1)

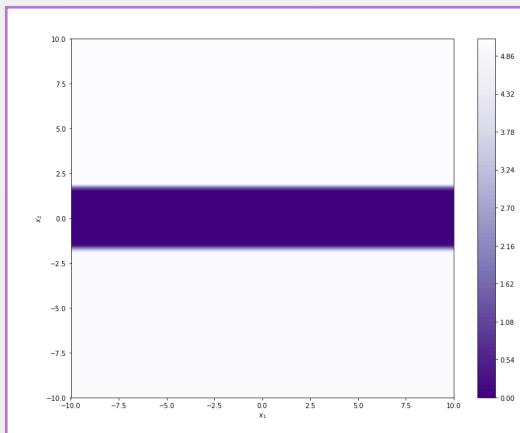


Example 6.3 (2,4,1)

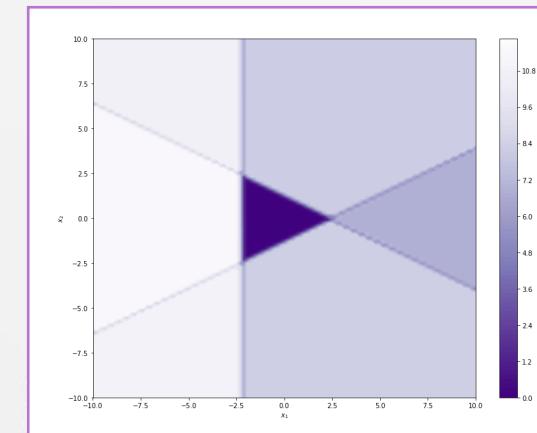
Neural Network Interpretation

Regression

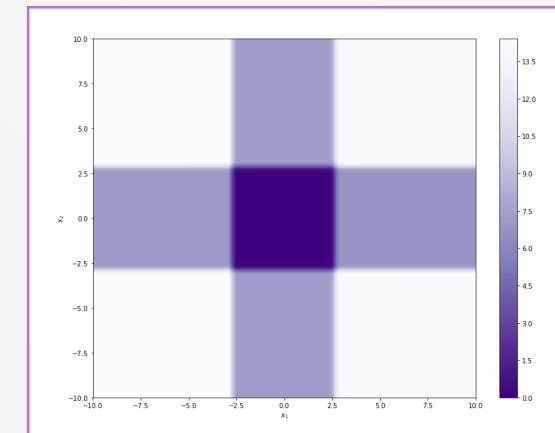
Contour plot predicted ឧប់ Example 6



Example 6.1 (2,2,1)



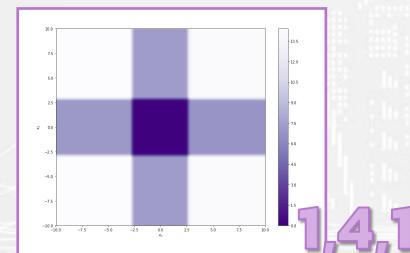
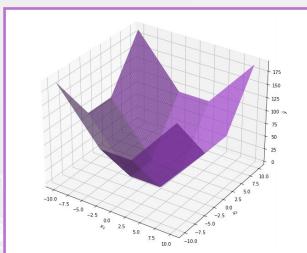
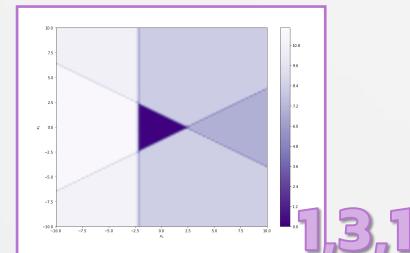
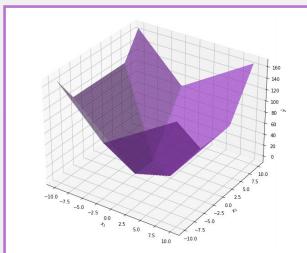
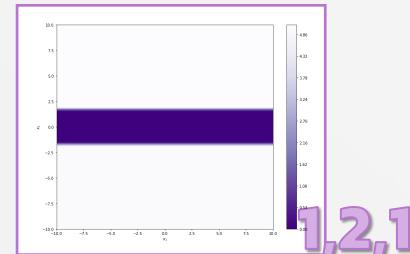
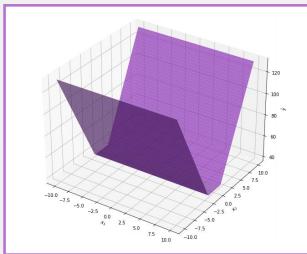
Example 6.2 (2,3,1)



Example 6.3 (2,4,1)

Neural Network Interpretation

Regression

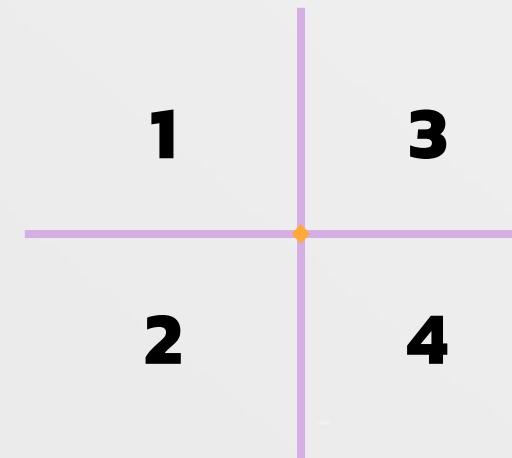
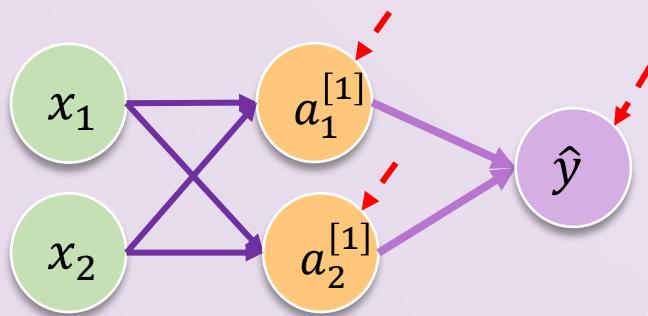


ข้อสรุปของ Example 6

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

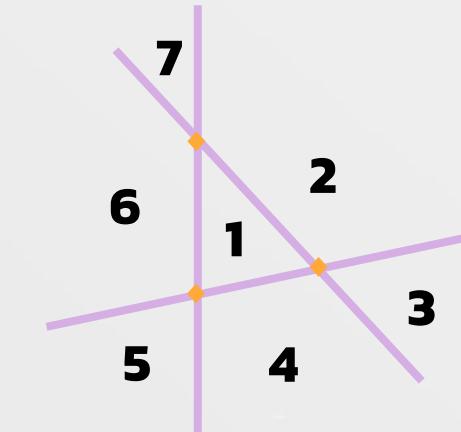
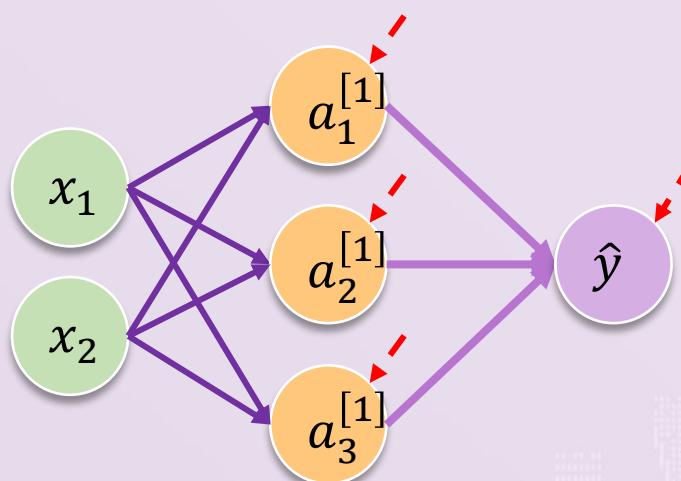
Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้



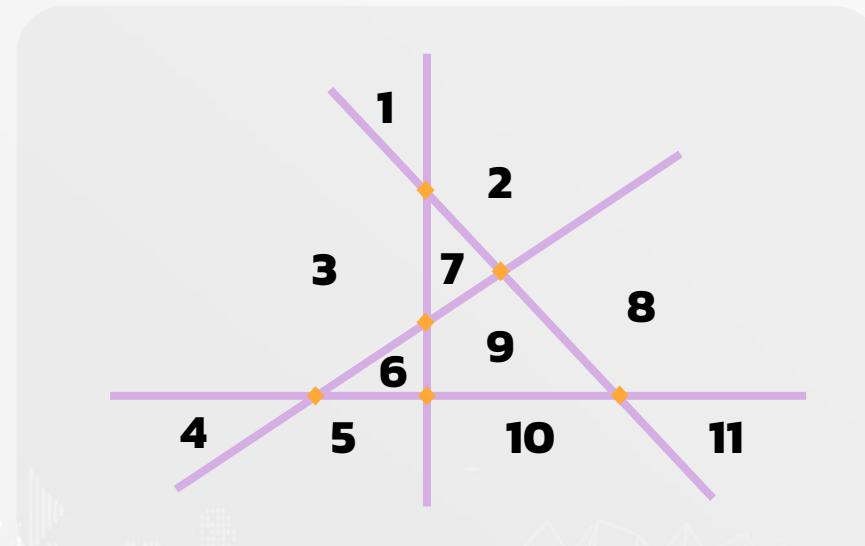
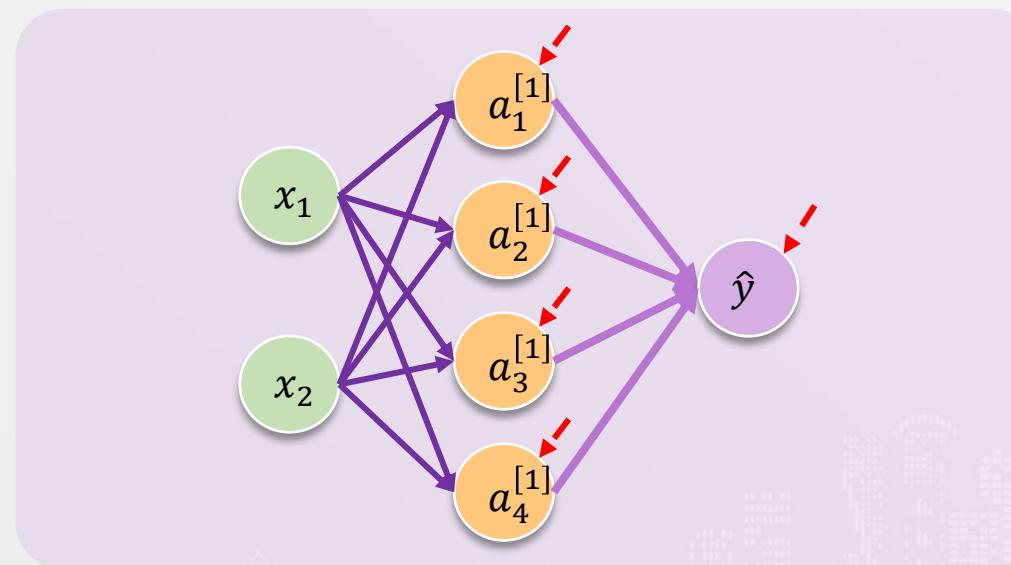
Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้



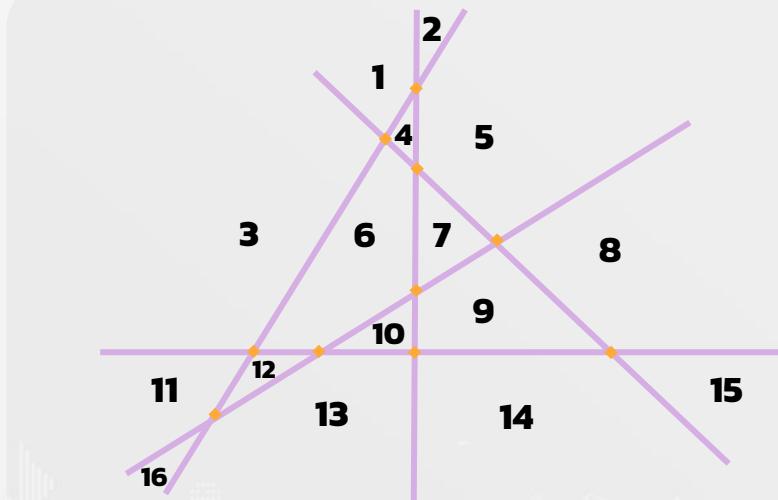
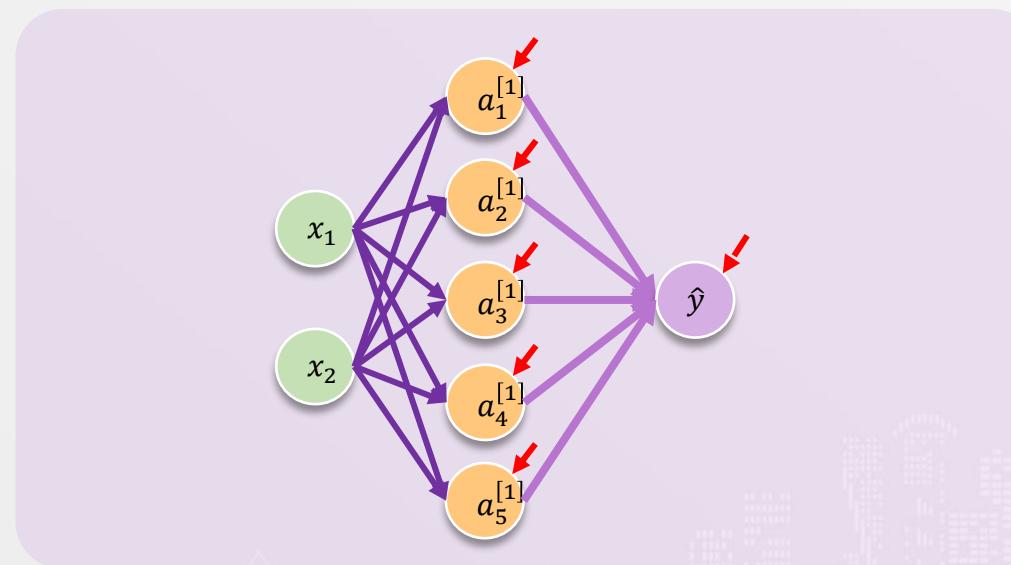
Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้



Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้

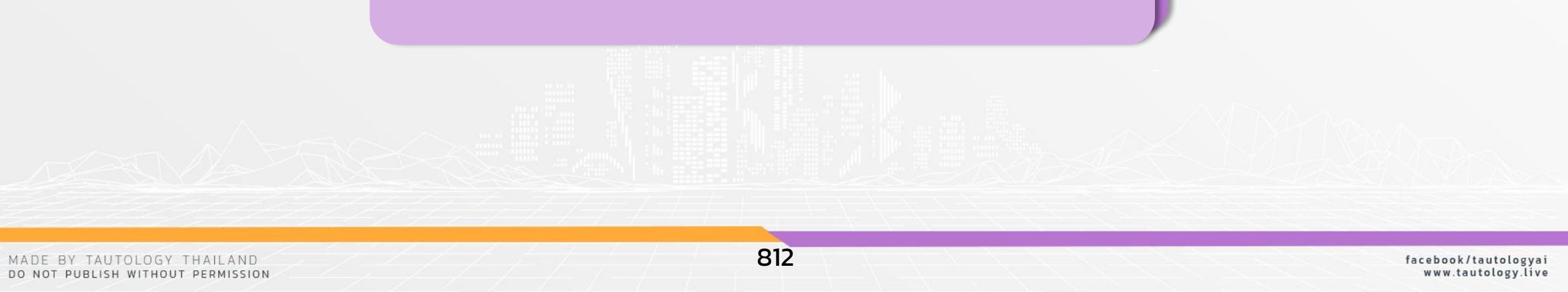


Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้

Maximum Plane ที่เกิดจาก *ReLU* จะเกิดได้ เมื่อ
รอยหัก เกิดการตัดกันกั้งหมด

(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)**

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

โดยที่ n คือ จำนวน node ใน hidden layer

Neural Network Interpretation

Maximum Plane ที่สามารถเกิดขึ้นได้

ก็จะเกิด **Maximum Plane** ได้ ก็ต่อเมื่อ
Dataset, initial parameter มีความเหมาะสม
(รอยหักเกิดการตัดกันก็จะหมด)

Neural Network Interpretation

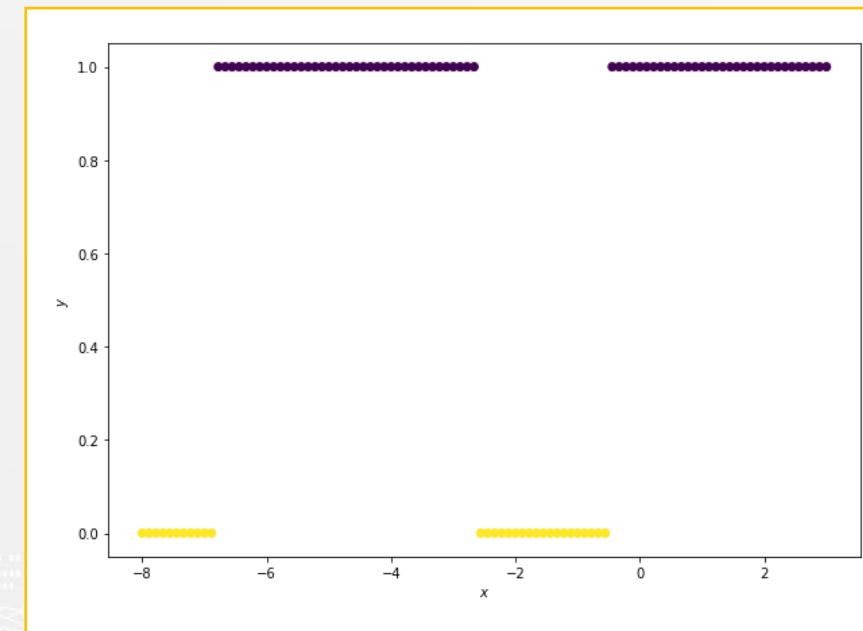
Regression

Classification

Neural Network Interpretation

Classification

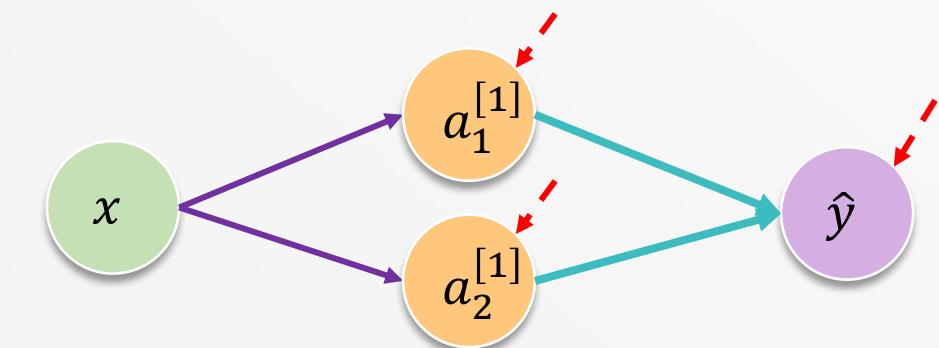
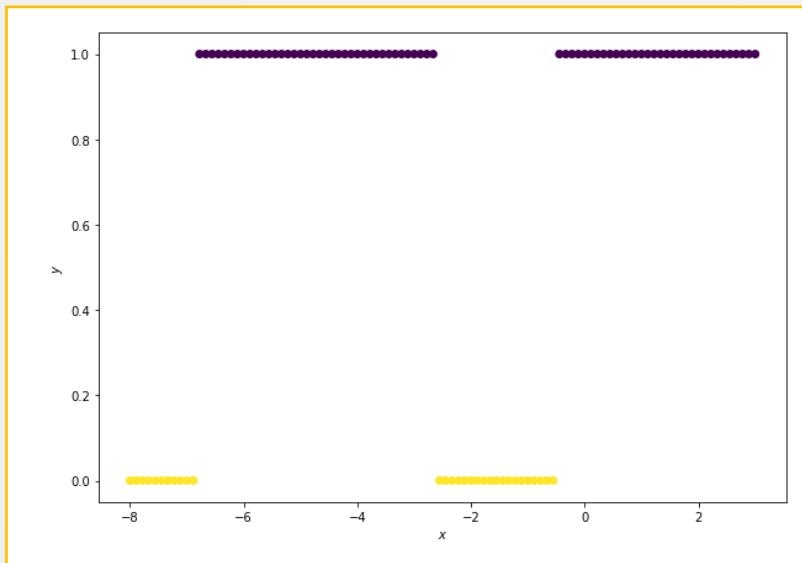
Example 1: Binary Classification



Neural Network Interpretation

Classification

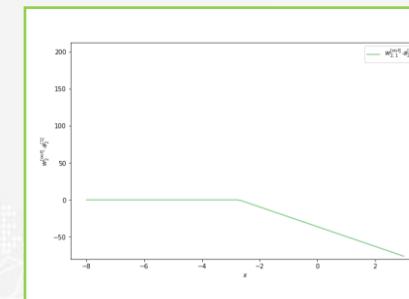
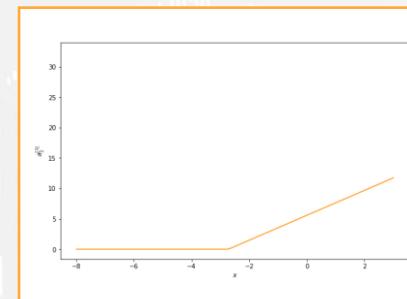
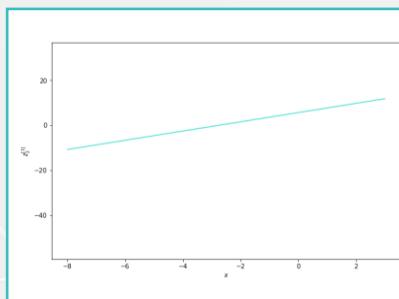
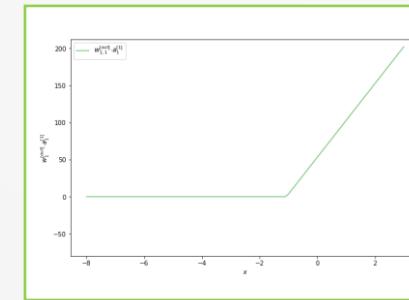
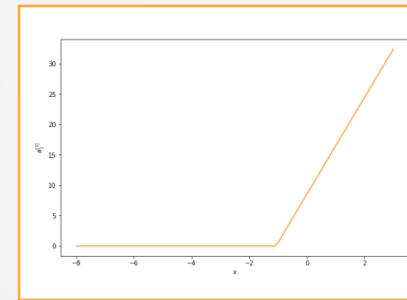
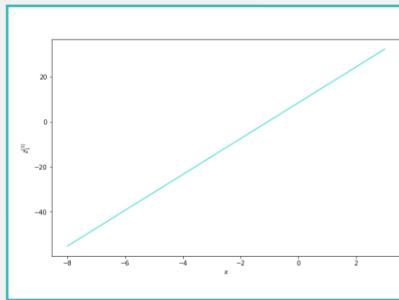
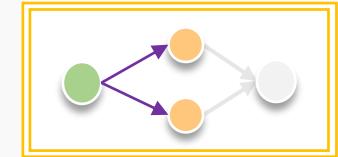
Example 1.1



Neural Network Interpretation

Classification

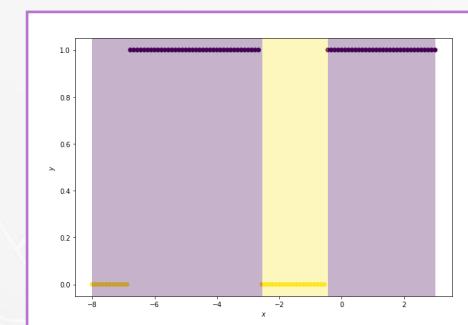
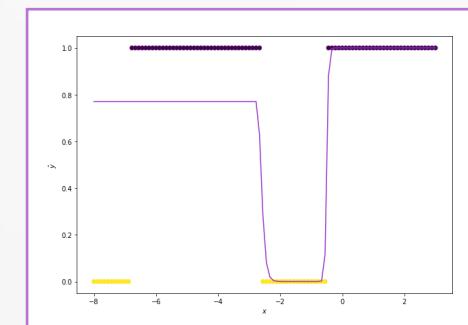
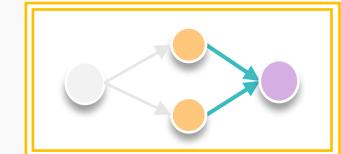
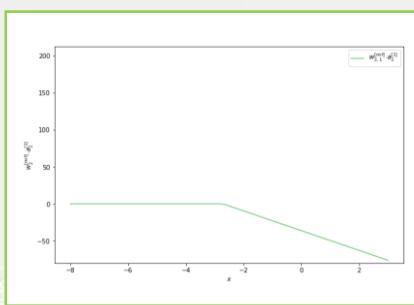
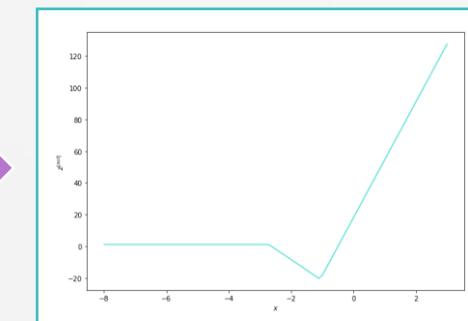
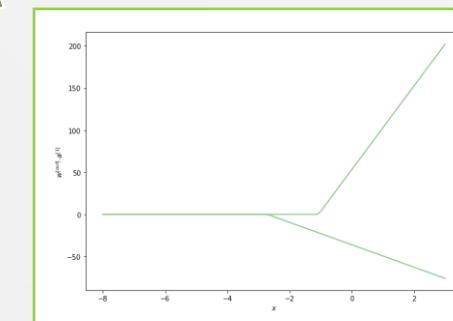
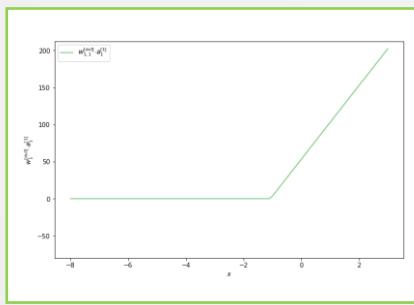
Example 1.1



Neural Network Interpretation

Classification

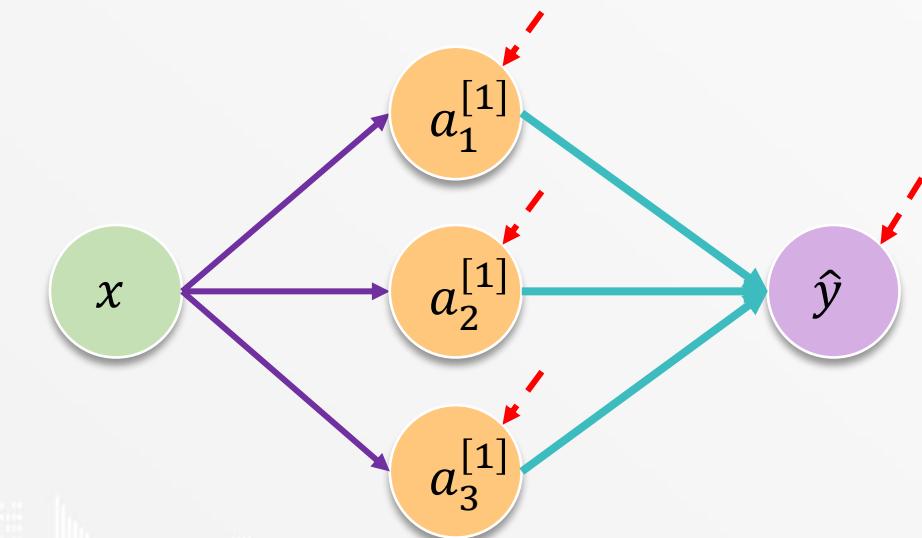
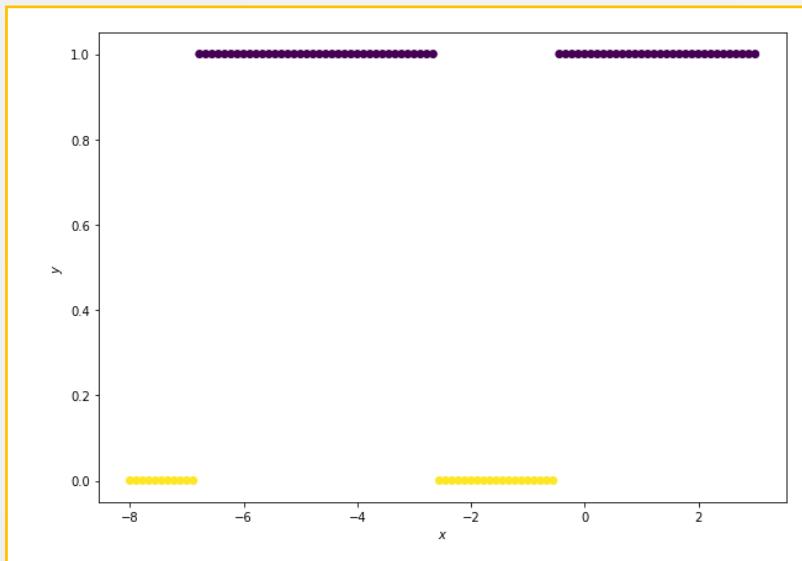
Example 1.1



Neural Network Interpretation

Classification

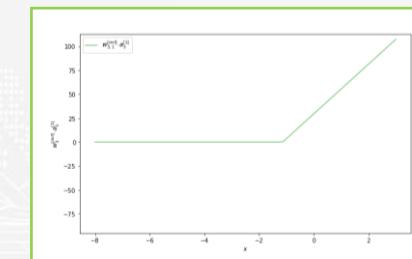
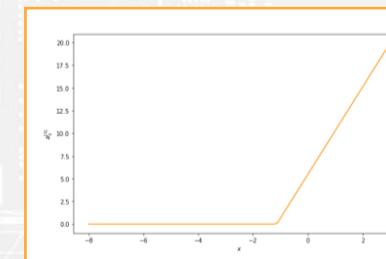
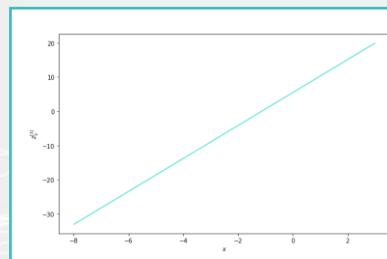
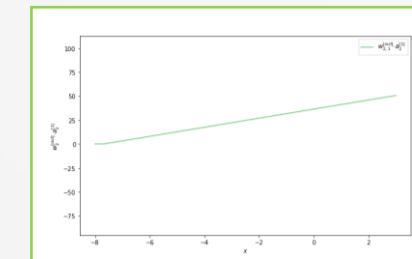
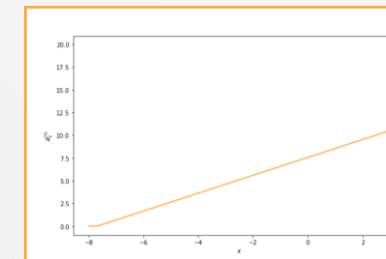
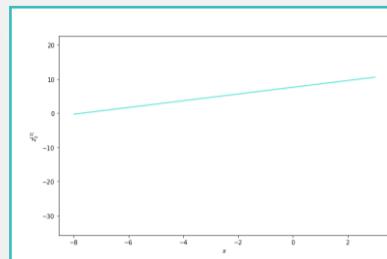
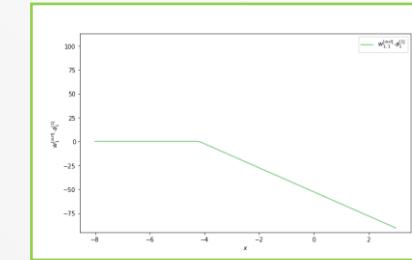
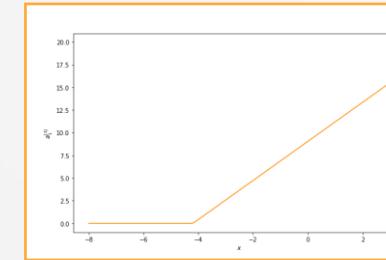
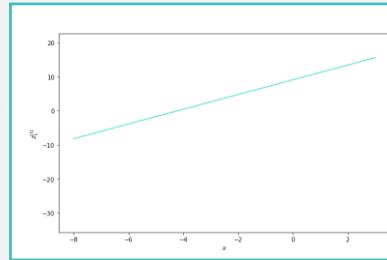
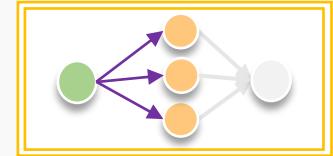
Example 1.2



Neural Network Interpretation

Classification

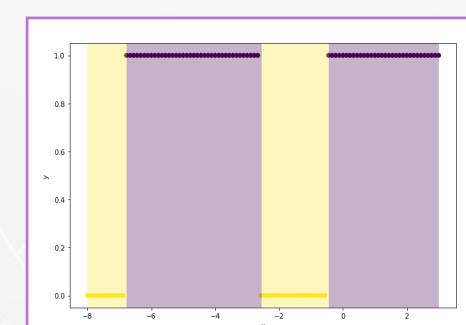
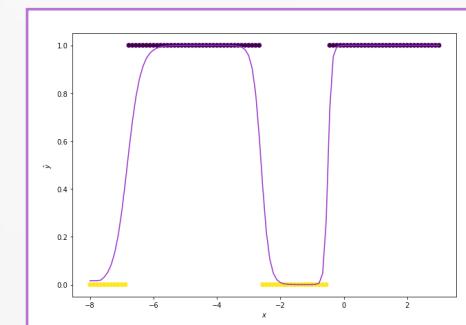
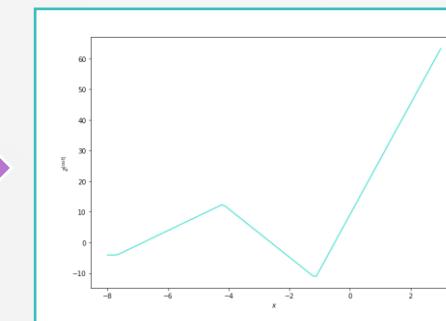
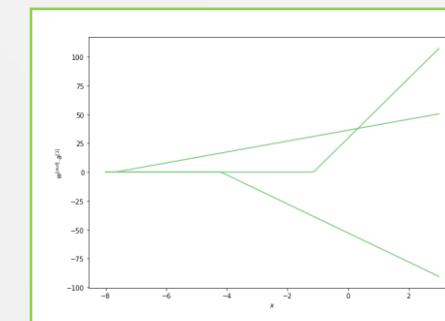
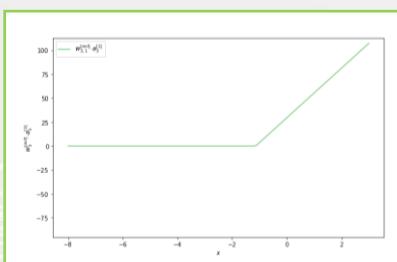
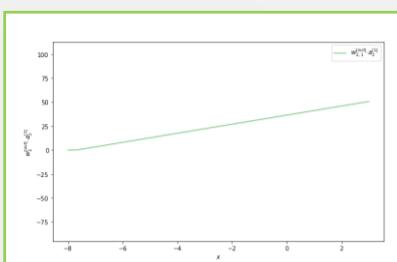
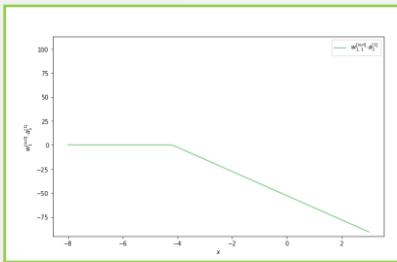
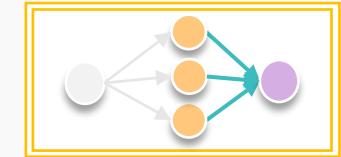
Example 1.2



Neural Network Interpretation

Classification

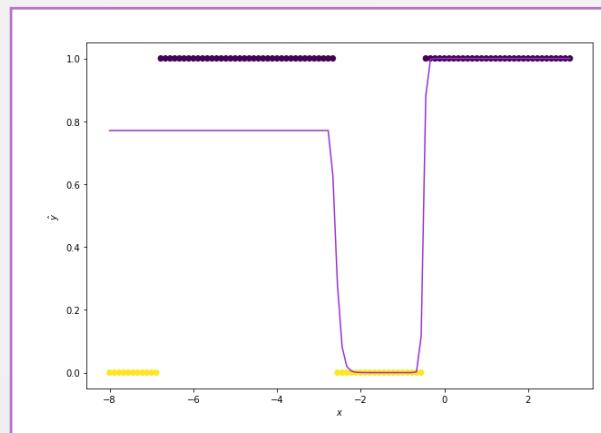
Example 1.2



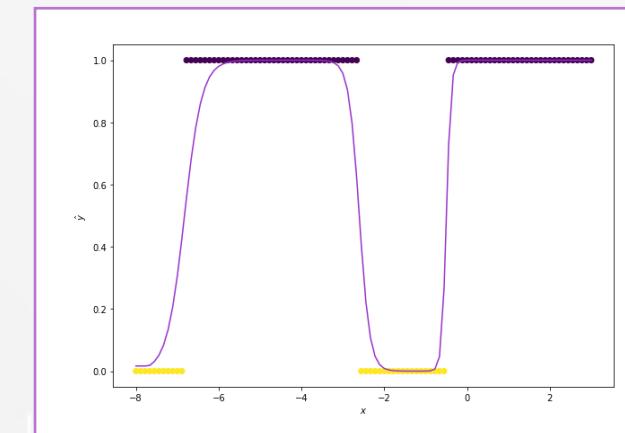
Neural Network Interpretation

Classification

กราฟแสดงค่า predicted ของ Example 1



Example 1.1 (1,2,1)

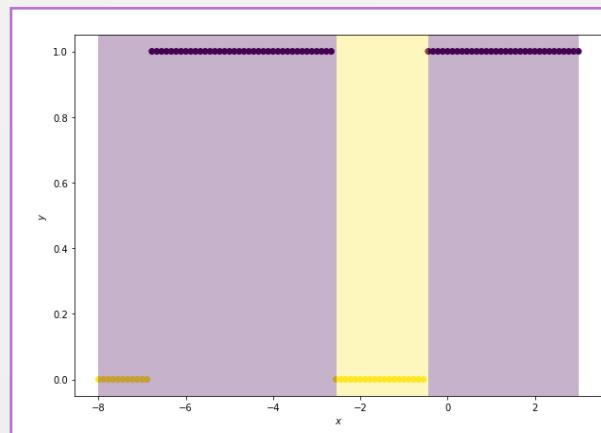


Example 1.2 (1,3,1)

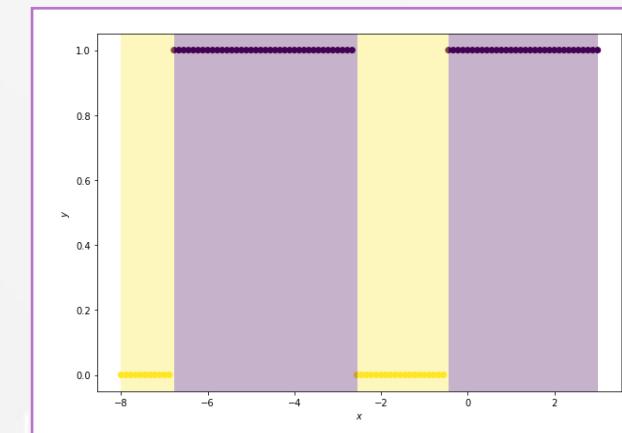
Neural Network Interpretation

Classification

กราฟแสดงพื้นที่ predicted ของ Example 1



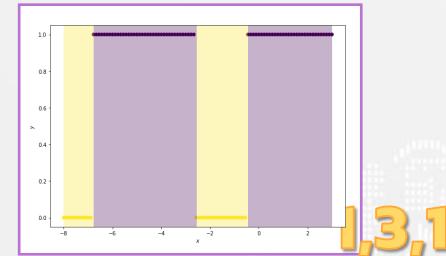
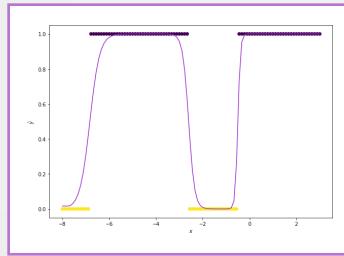
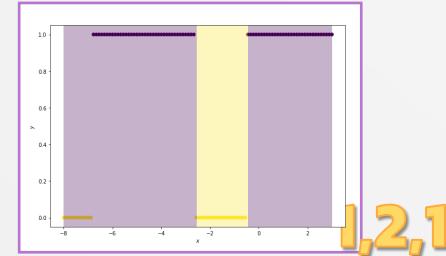
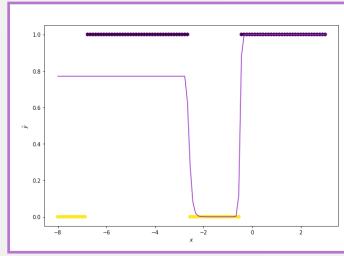
Example 1.1 (1,2,1)



Example 1.2 (1,3,1)

Neural Network Interpretation

Classification



ข้อสรุปของ Example 1

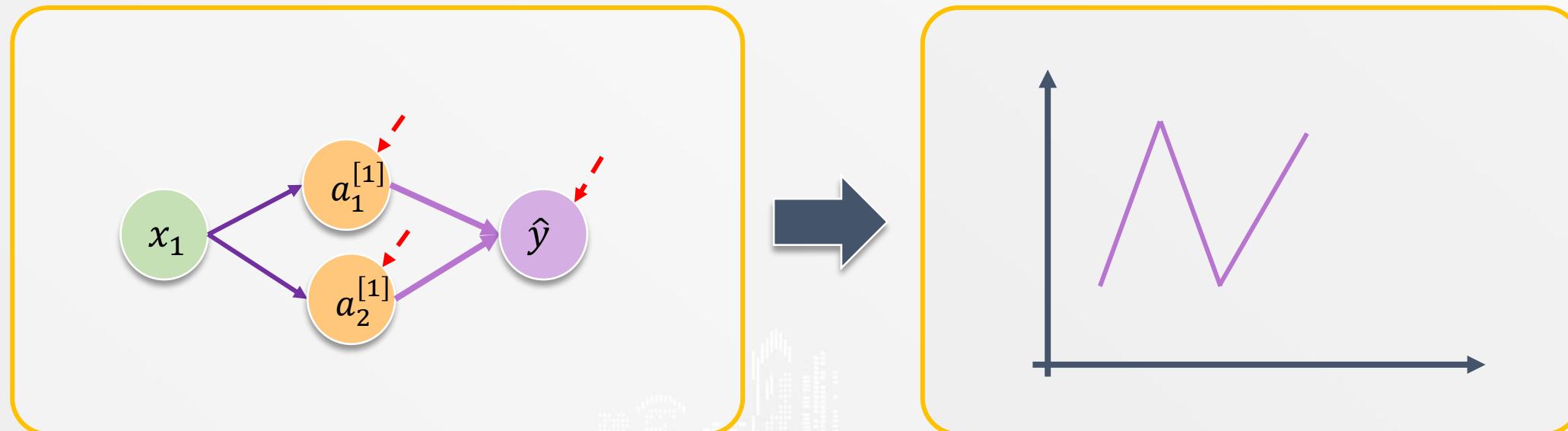
- Plane (ส่วนของเส้นตรง) ที่เพิ่มขึ้นจากจำนวน node ที่มากขึ้น เมื่อผ่าน sigmoid function และจะทำให้เกิด region ของการแบ่ง class ที่เพิ่มขึ้นตามไปด้วย

Neural Network Interpretation

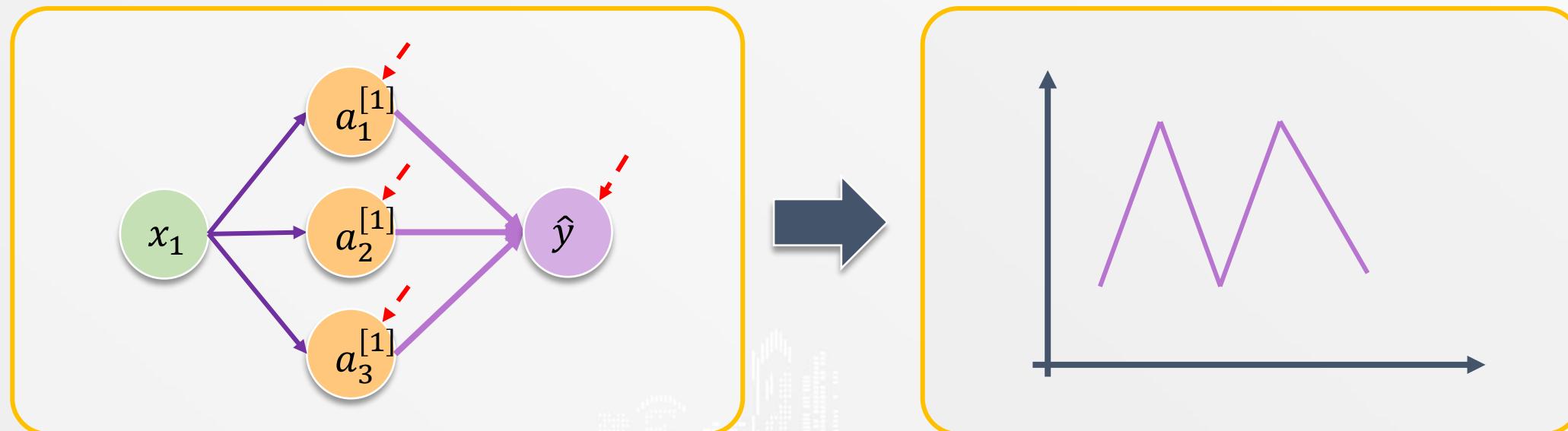
Classification

จากความรู้เดิมกี่ว่า **Neural Network Regression** ใบ 2 มิติ นั้น จำนวน plane (ส่วนของเส้นตรง) ใบชั้น output มีจำนวนมากที่สุด คือ $n + 1$

Neural Network Interpretation



Neural Network Interpretation



Neural Network Interpretation



Neural Network Interpretation

Classification

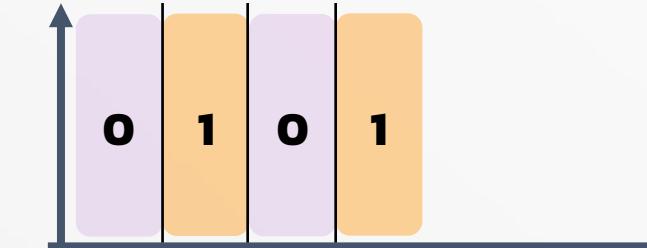
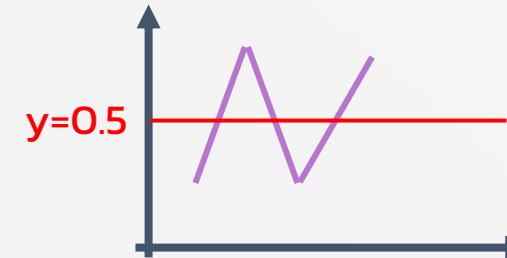
เมื่อเรานำชั้น output เหล่านั้น มาผ่าน sigmoid function เราจะได้ว่า

“ค่าของ **output** **มากกว่า 0.5** model จะพยากรณ์ผลลัพธ์ เป็น **class 1**”

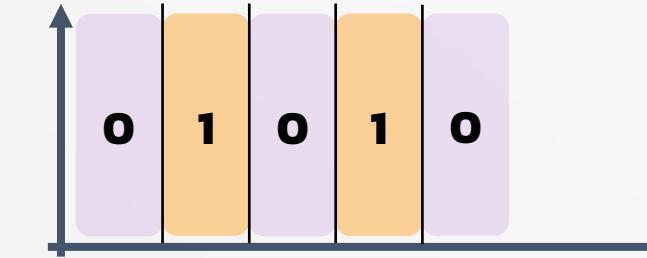
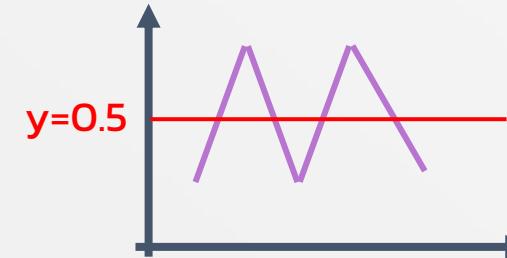
ดังนั้น เราจะใช้ $y = 0.5$ เพื่อใช้ในการแบ่ง region ของการพยากรณ์

Neural Network Interpretation

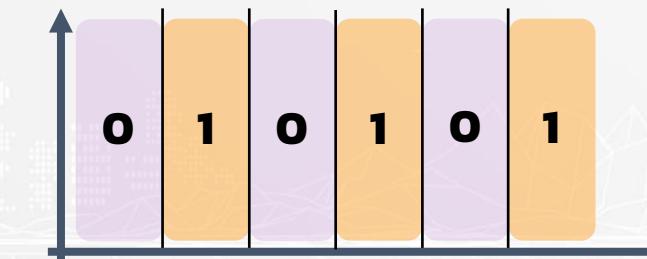
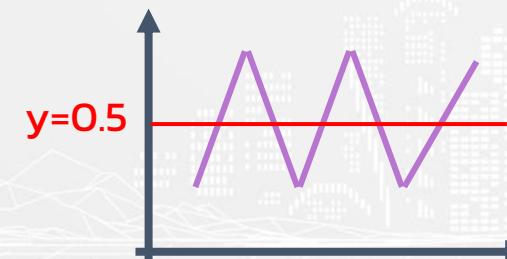
n = 2



n = 3



n = 4



Neural Network Interpretation

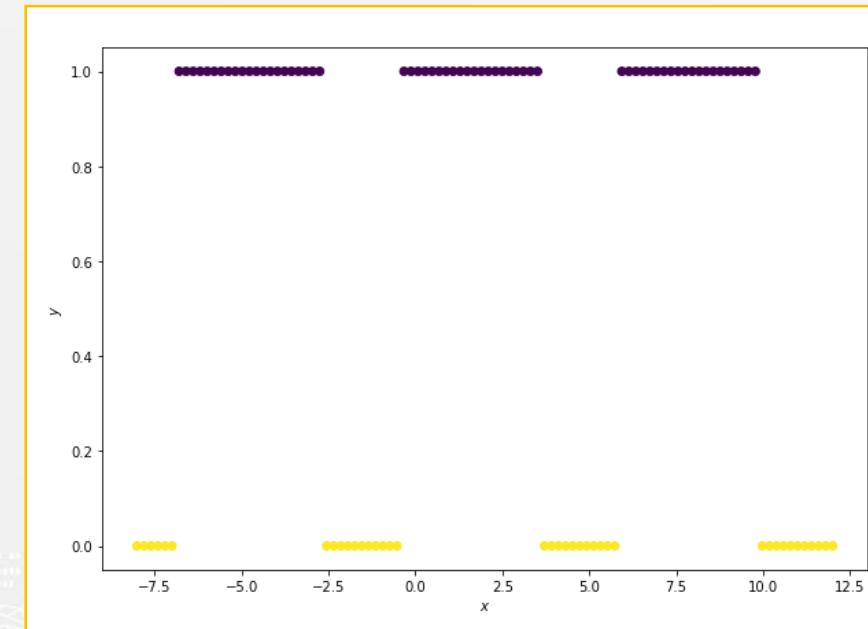
Classification

จากภาพเราจะเห็นว่า region ในการแบ่ง class ของ binary ใน 2 มิติ (ตัวแปรตัวบิ๊บ) จะมีค่าสูงสุดได้เท่ากับ $t + 2$ เมื่อ t คือ จำนวน node ใน hidden layer

Neural Network Interpretation

Classification

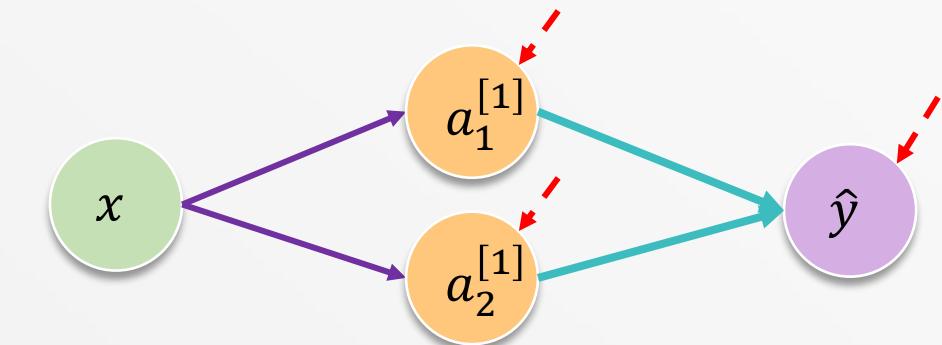
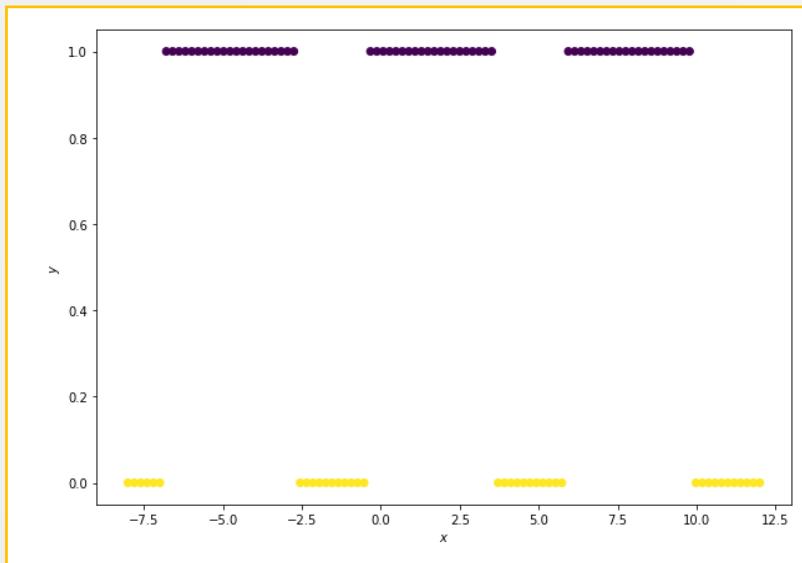
Example 2 : Binary Classification



Neural Network Interpretation

Classification

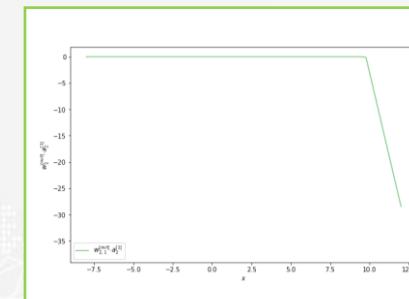
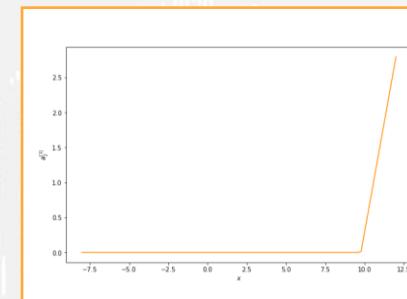
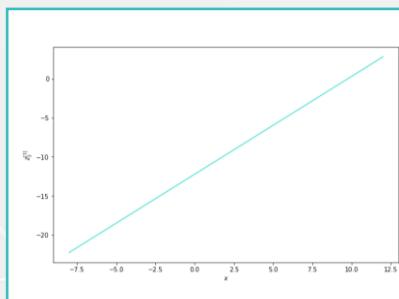
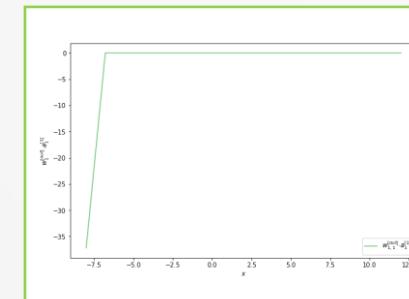
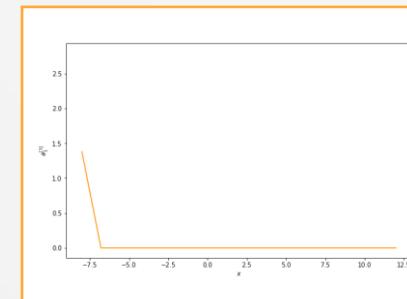
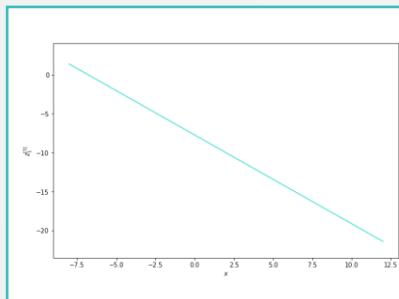
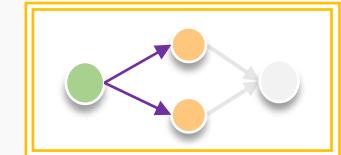
Example 2.1



Neural Network Interpretation

Classification

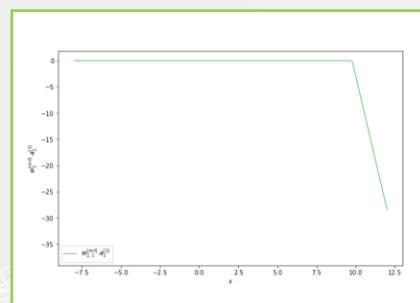
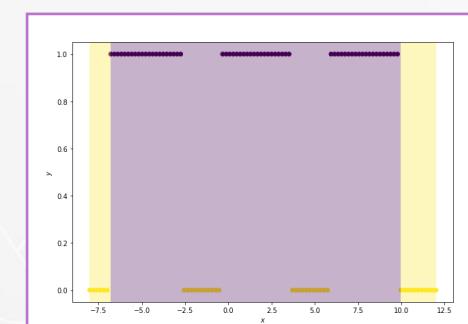
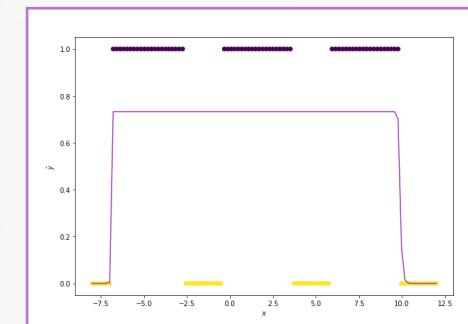
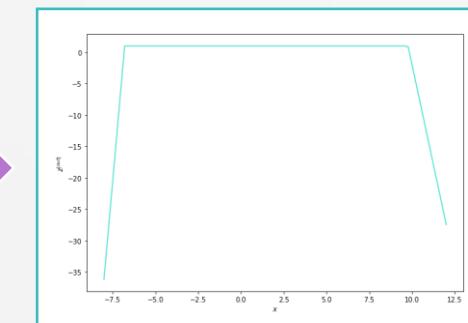
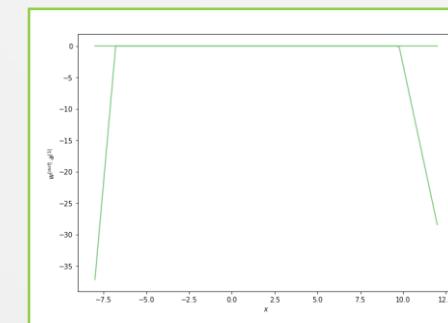
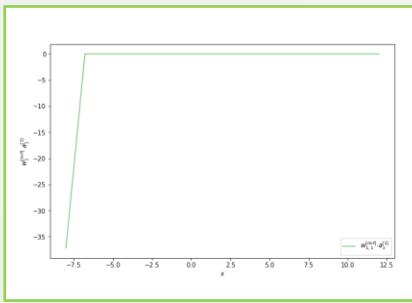
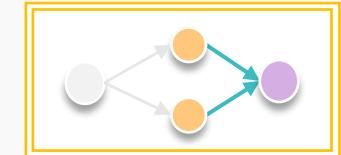
Example 2.1



Neural Network Interpretation

Classification

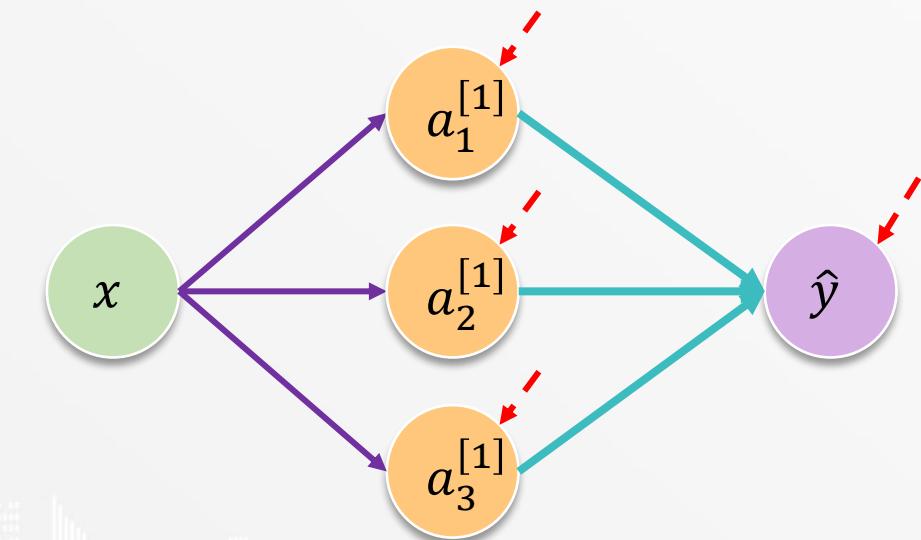
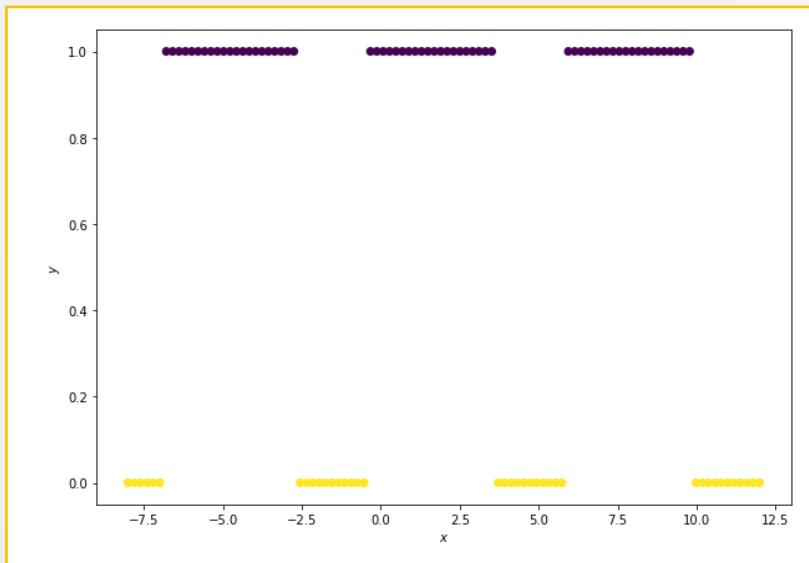
Example 2.1



Neural Network Interpretation

Classification

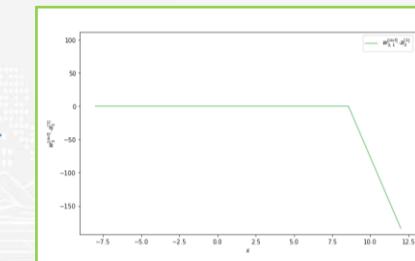
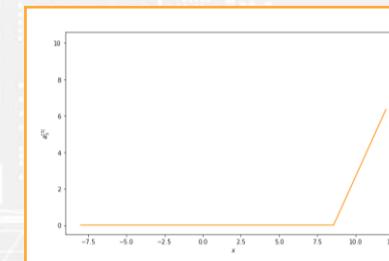
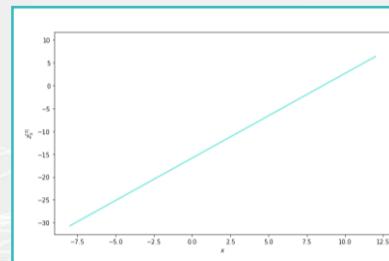
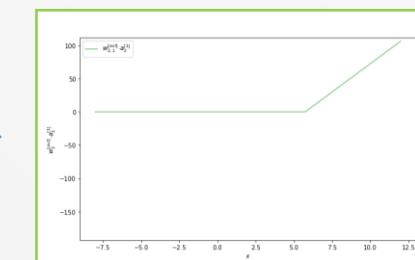
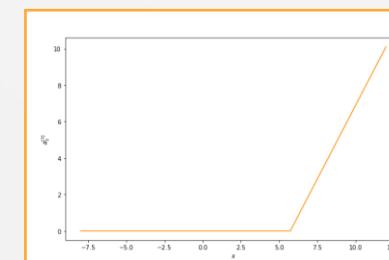
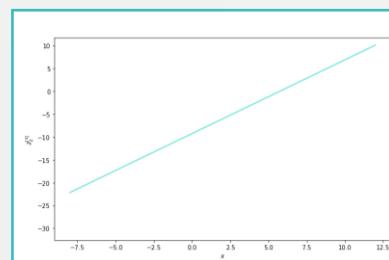
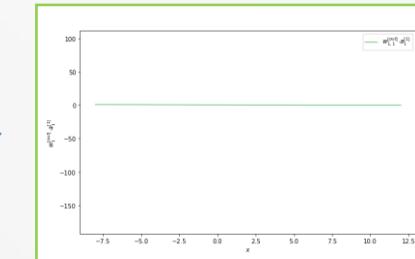
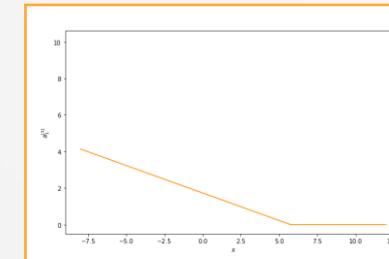
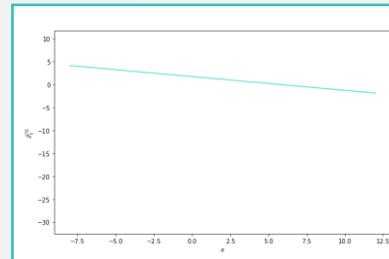
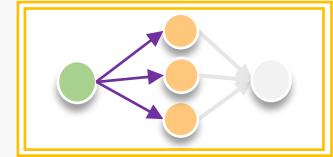
Example 2.2



Neural Network Interpretation

Classification

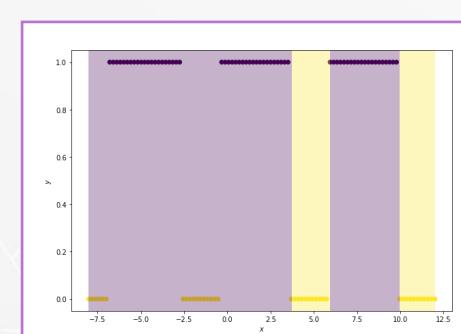
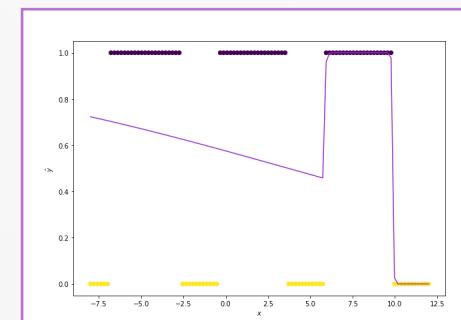
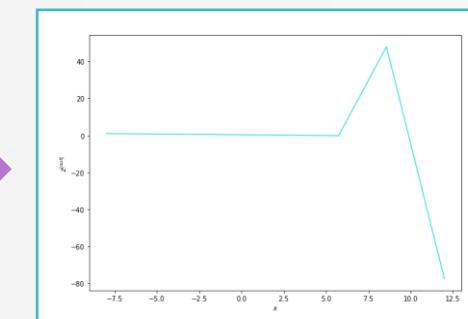
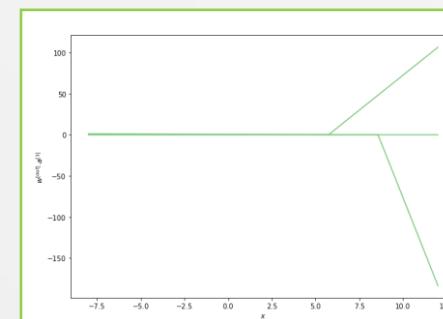
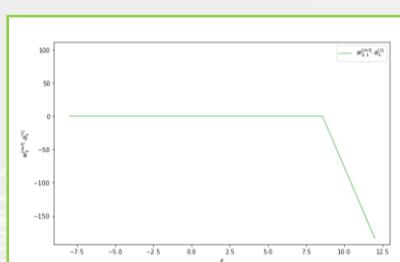
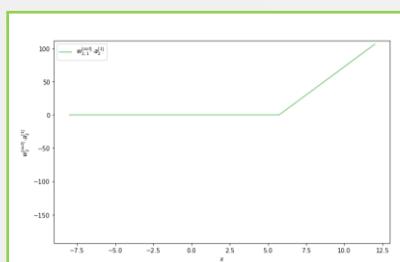
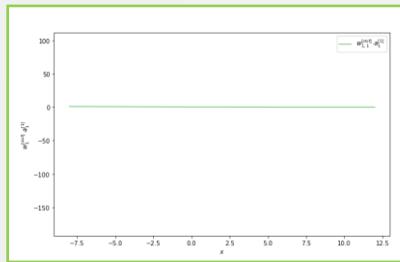
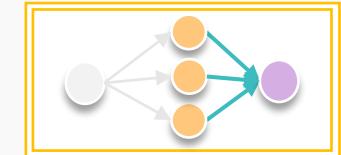
Example 2.2



Neural Network Interpretation

Classification

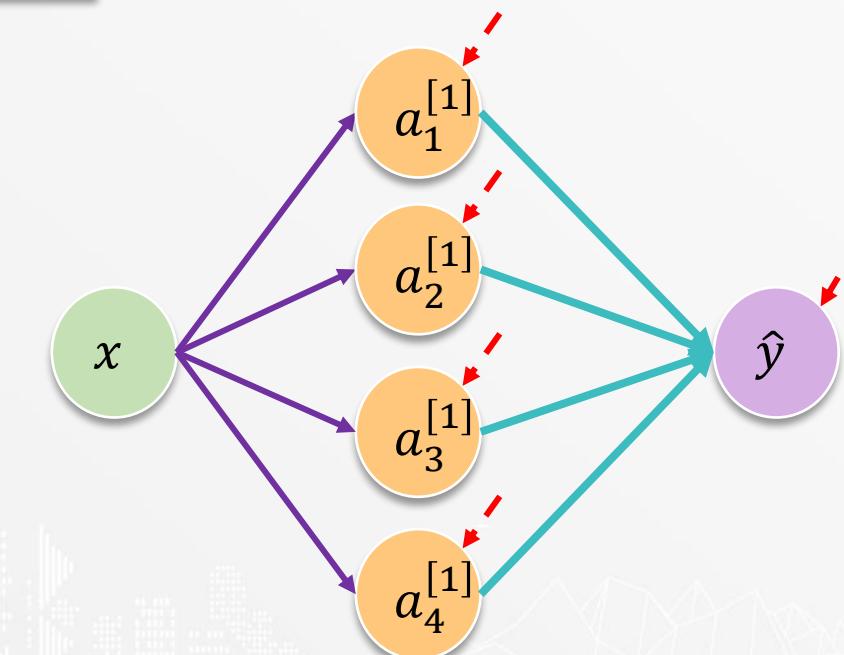
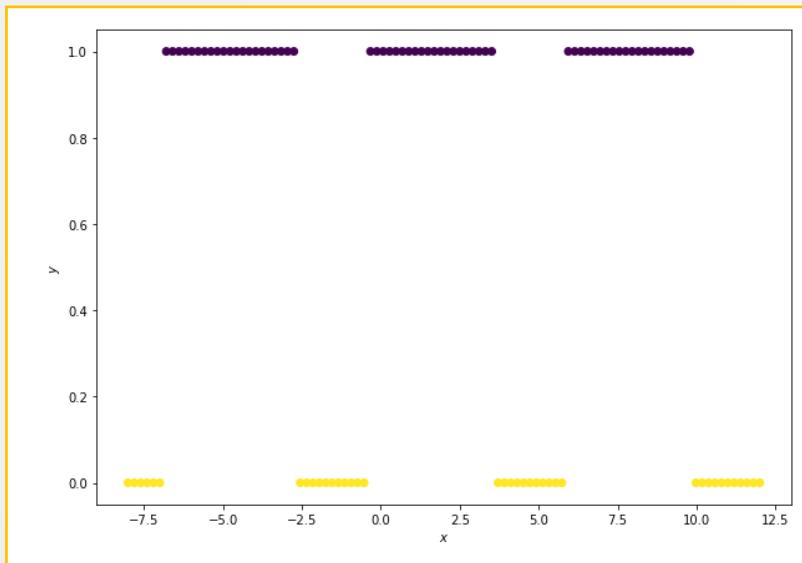
Example 2.2



Neural Network Interpretation

Classification

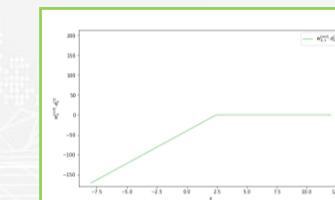
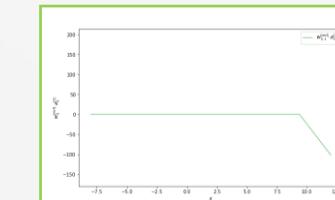
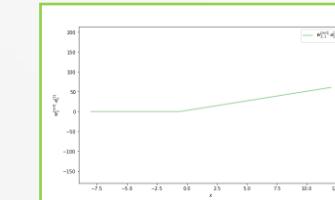
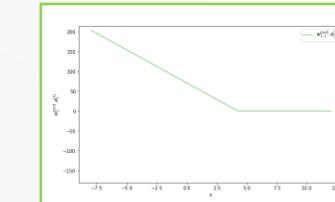
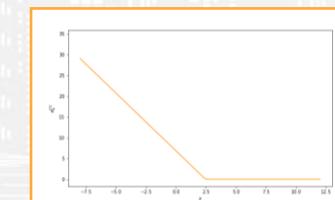
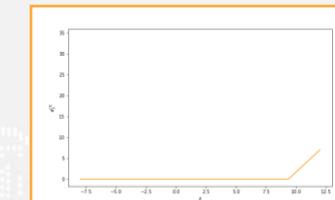
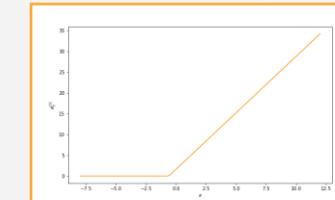
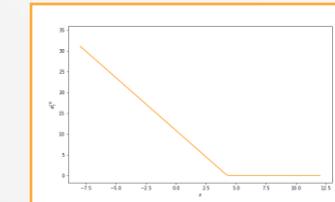
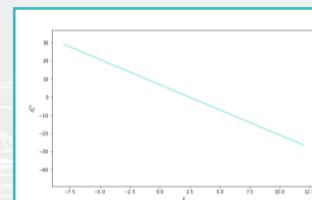
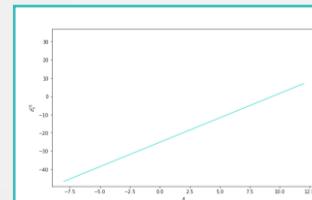
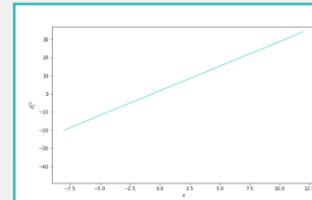
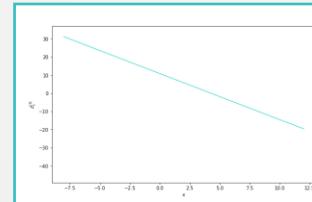
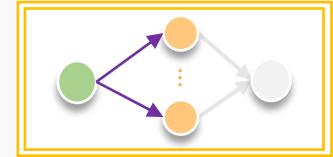
Example 2.3



Neural Network Interpretation

Classification

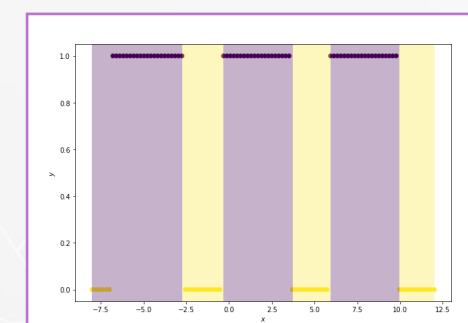
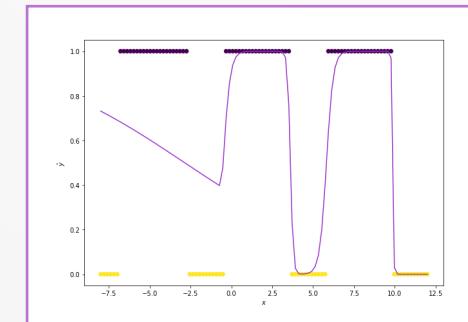
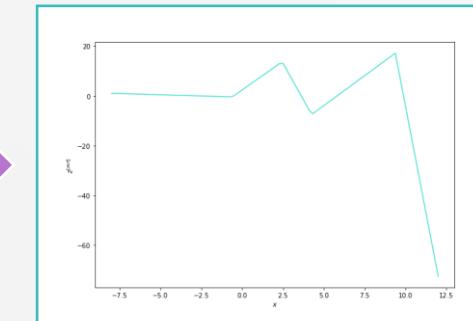
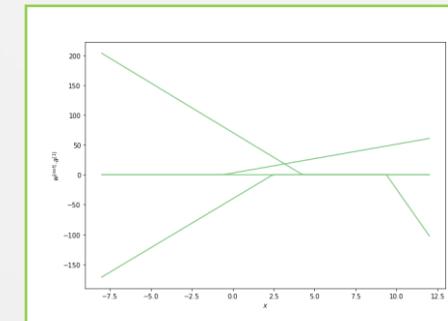
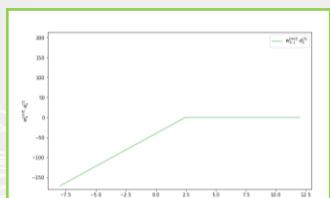
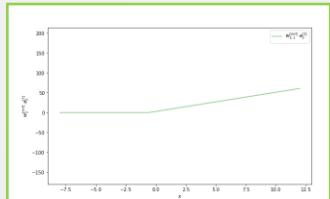
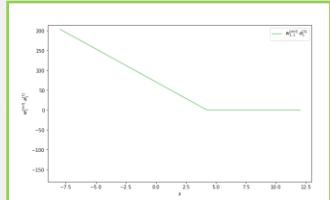
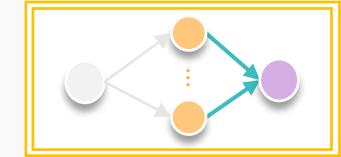
Example 2.3



Neural Network Interpretation

Classification

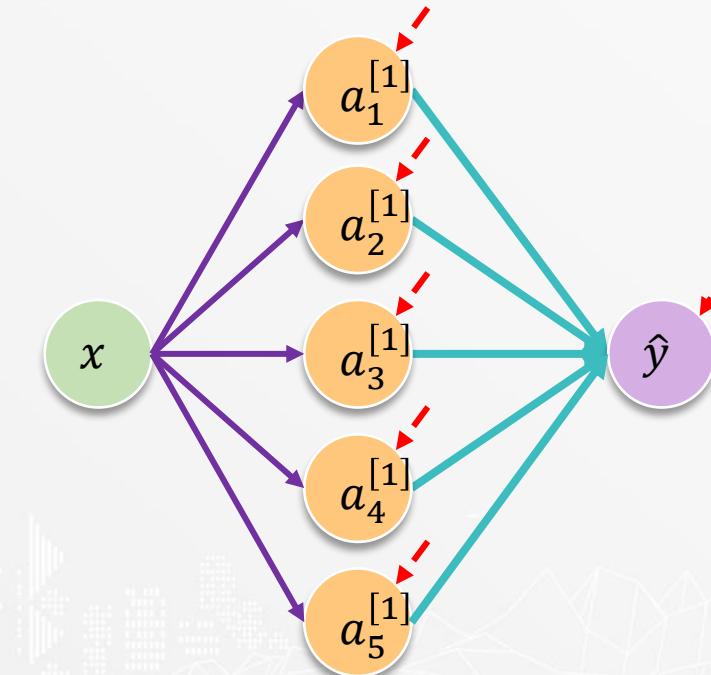
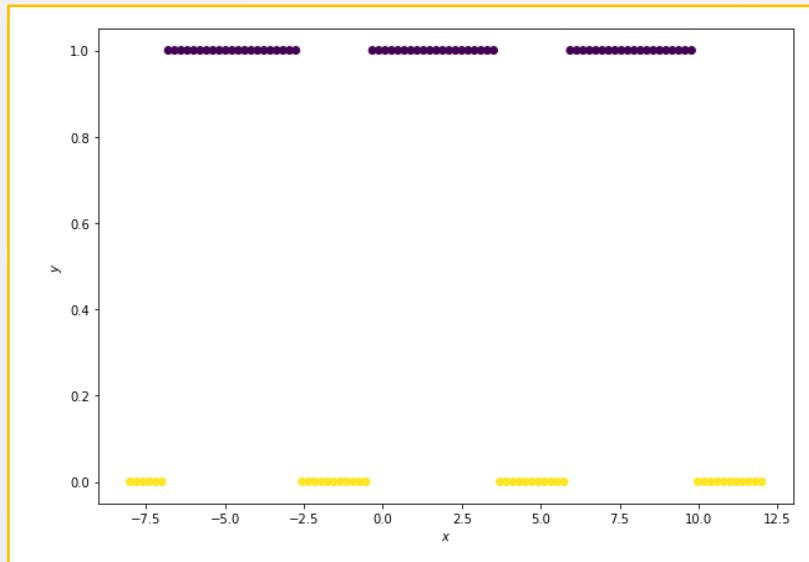
Example 2.3



Neural Network Interpretation

Classification

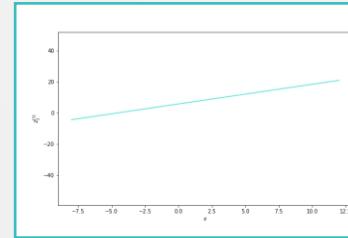
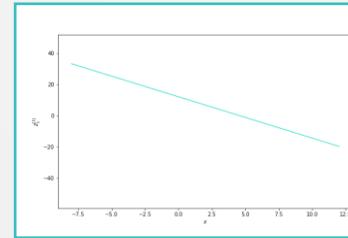
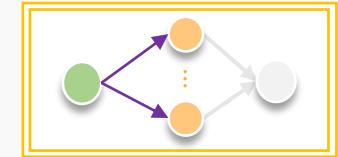
Example 2.4



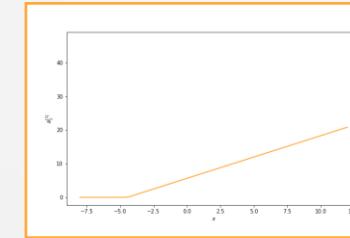
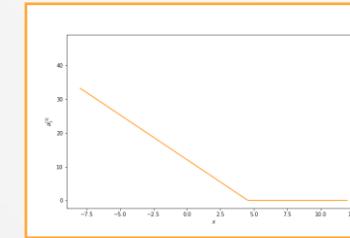
Neural Network Interpretation

Classification

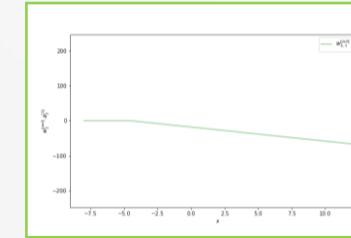
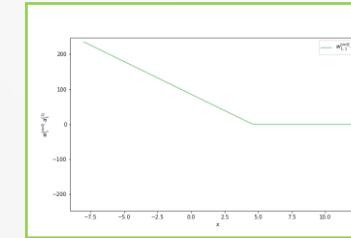
Example 2.4



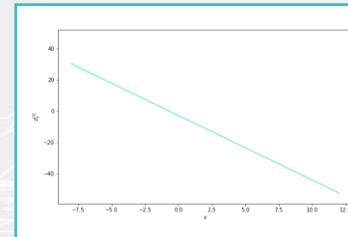
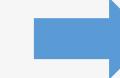
2



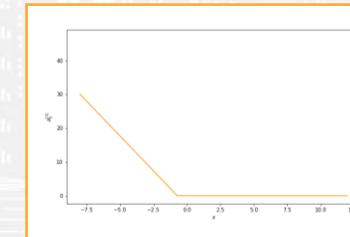
2



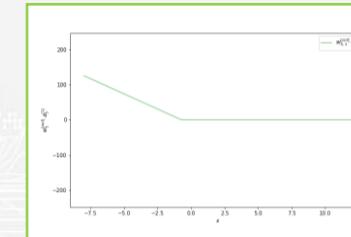
2



5



5

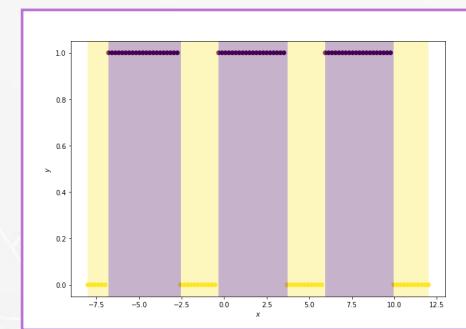
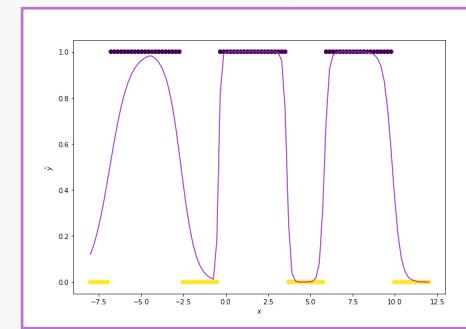
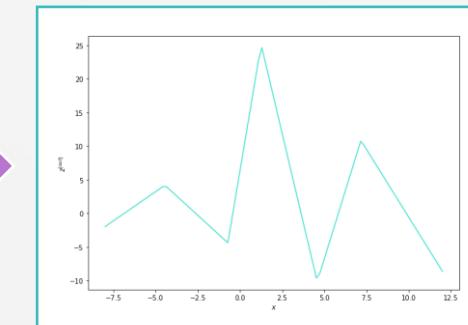
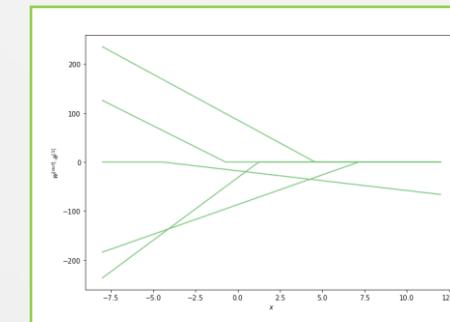
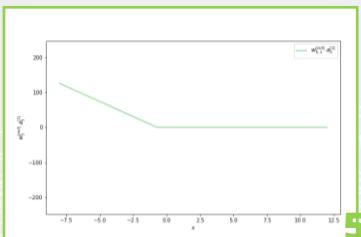
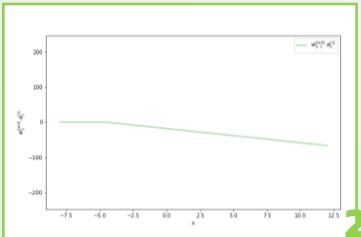
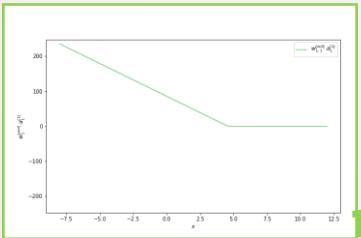
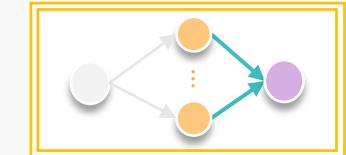


5

Neural Network Interpretation

Classification

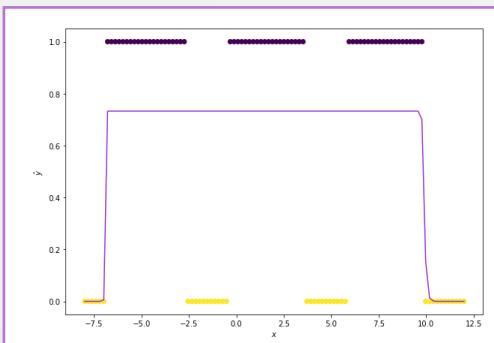
Example 2.4



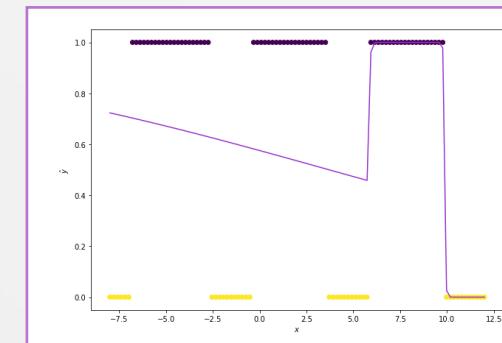
Neural Network Interpretation

Classification

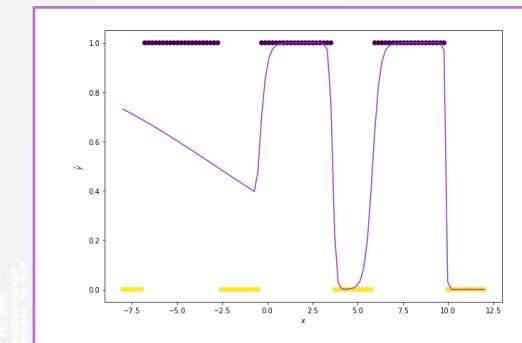
กราฟแสดงค่า predicted ของ Example 2



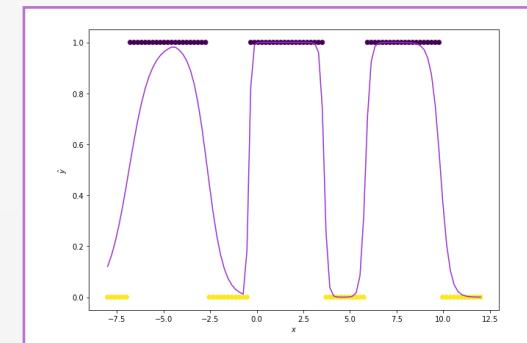
Example 2.1 (1,2,1)



Example 2.2 (1,3,1)



Example 2.3 (1,4,1)

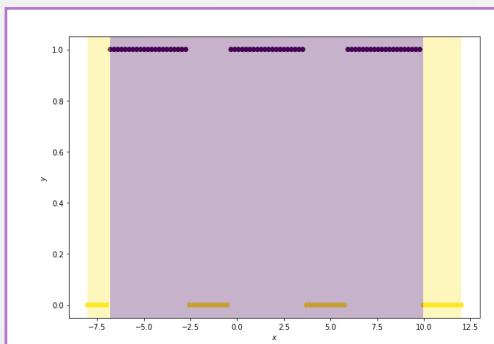


Example 2.4 (1,5,1)

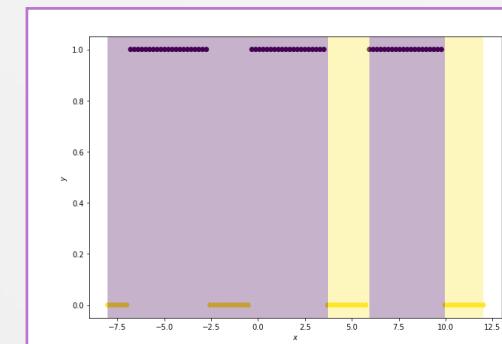
Neural Network Interpretation

Classification

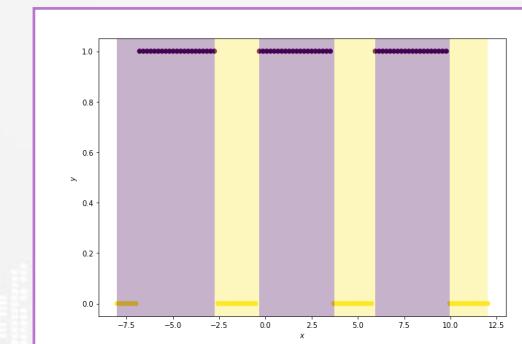
กราฟแสดงพื้นที่ predicted ของ Example 2



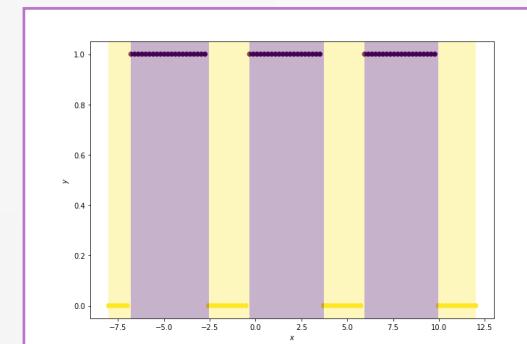
Example 2.1 (1,2,1)



Example 2.2 (1,3,1)



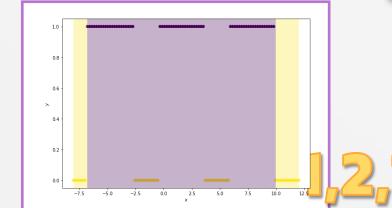
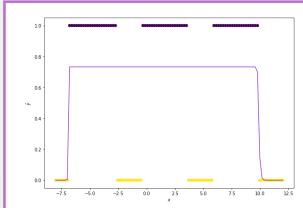
Example 2.3 (1,4,1)



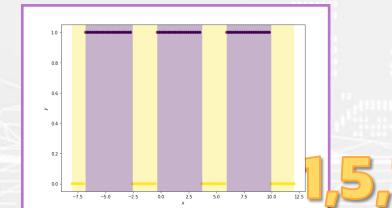
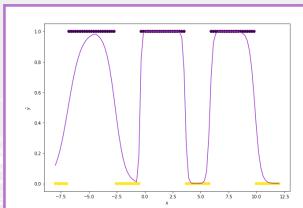
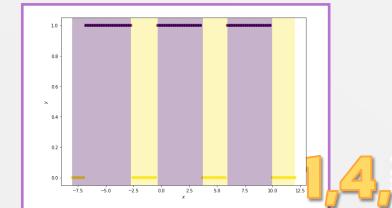
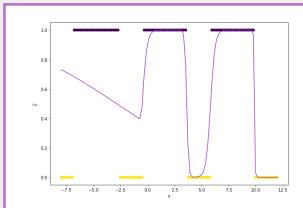
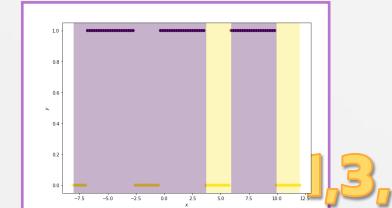
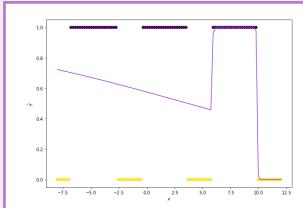
Example 2.4 (1,5,1)

Neural Network Interpretation

Classification



ข้อสรุปของ Example 2

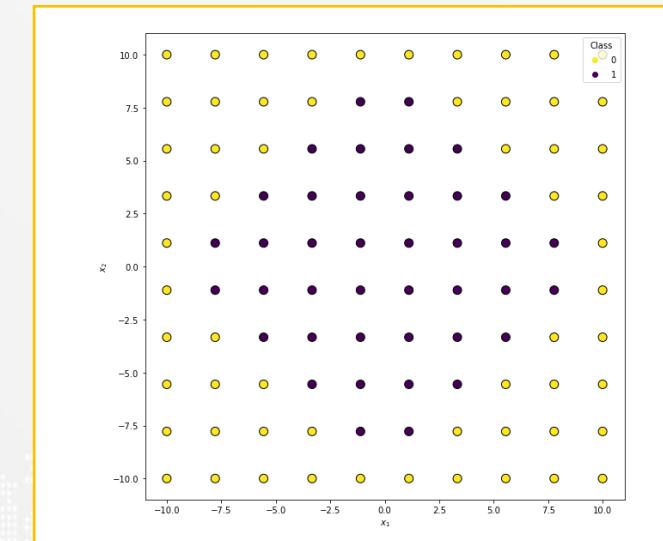
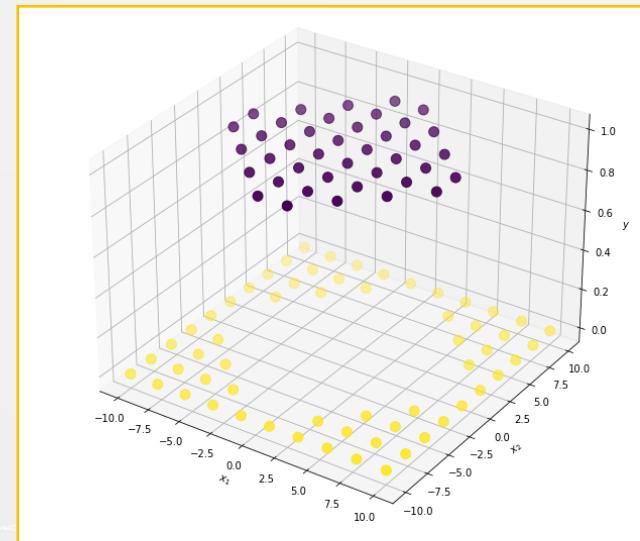


- จำนวน region ที่เกิดขึ้นใน 2 มิติ (จำนวนตัว แบ่งต้น 1 ตัว) เท่ากับ จำนวน node + 2 ทั้งนี้การ ที่จะเกิด maximum region นั้นขึ้นอยู่กับ dataset และ initial parameter ด้วย

Neural Network Interpretation

Classification

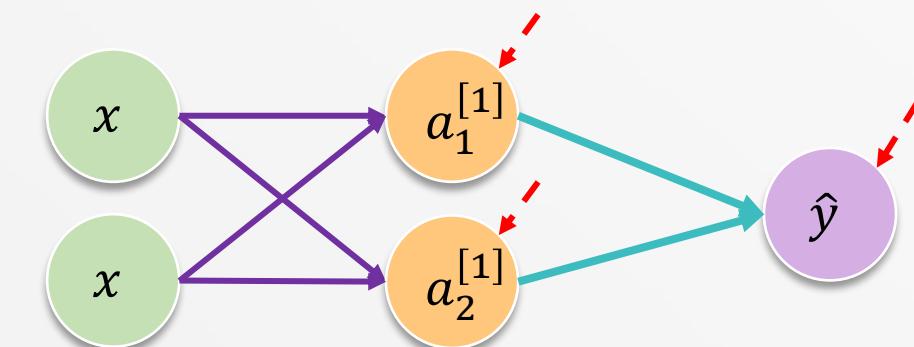
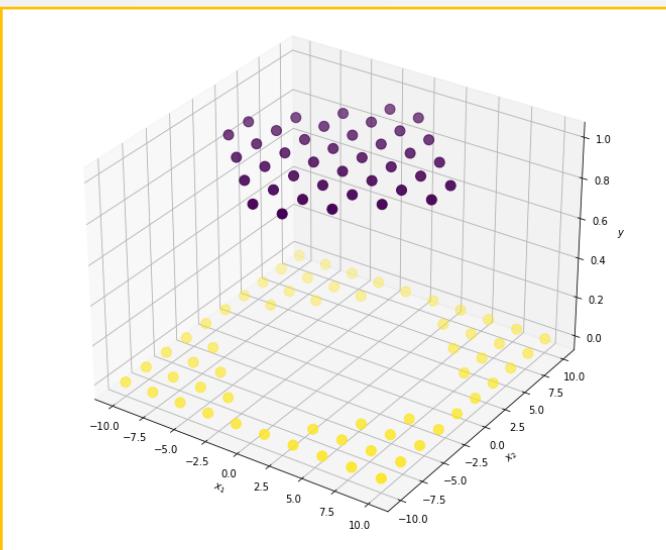
Example 3 : Binary Classification



Neural Network Interpretation

Classification

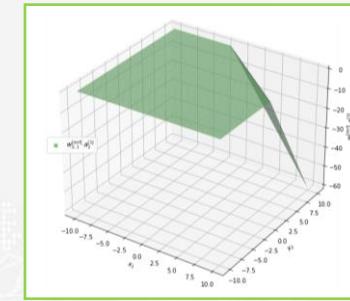
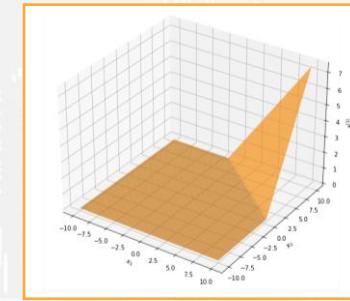
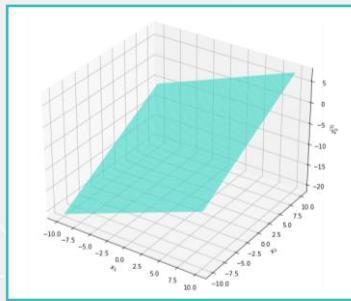
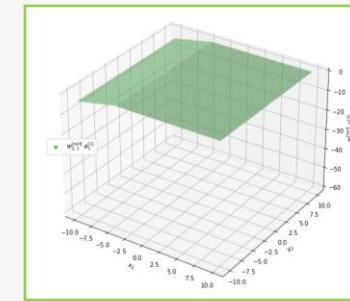
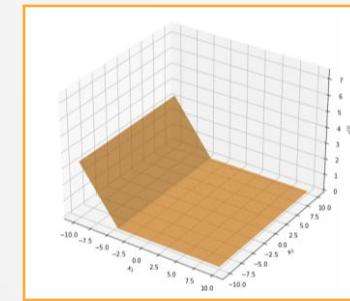
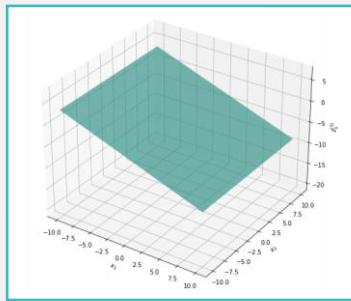
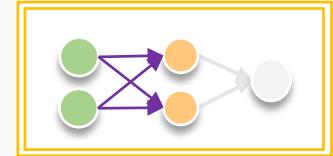
Example 3.1



Neural Network Interpretation

Classification

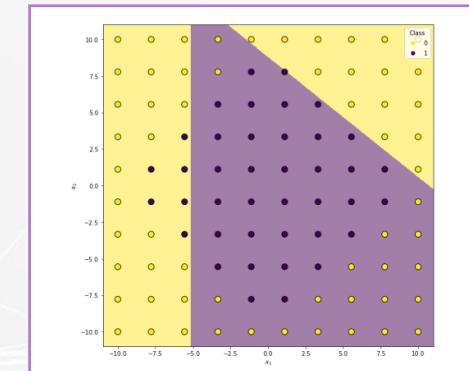
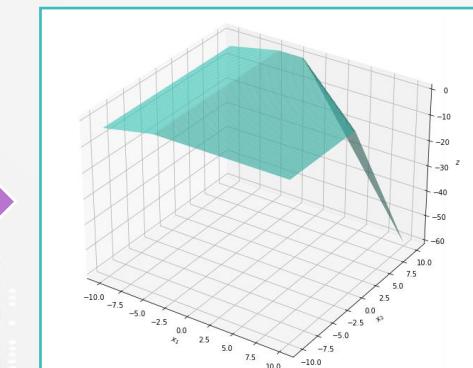
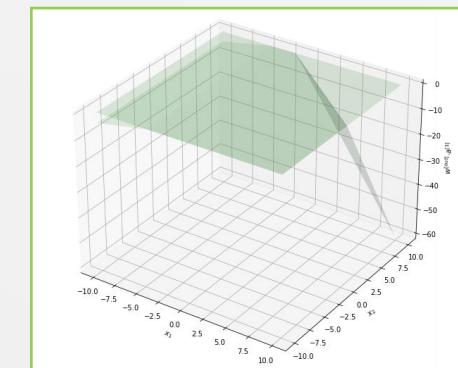
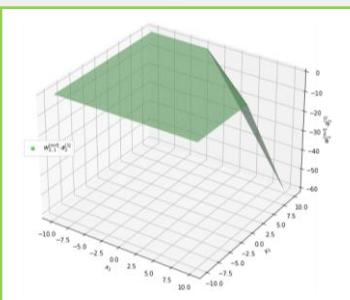
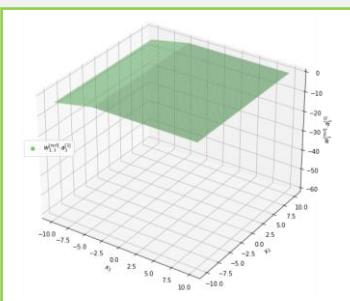
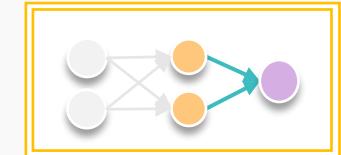
Example 3.1



Neural Network Interpretation

Classification

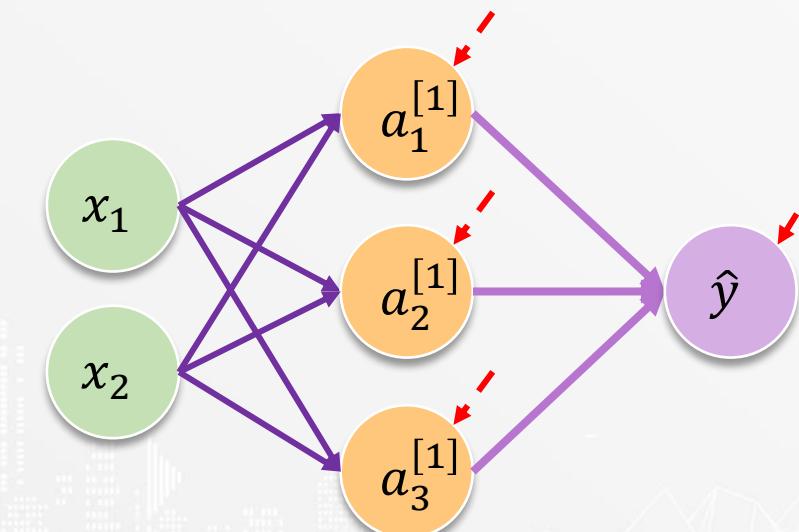
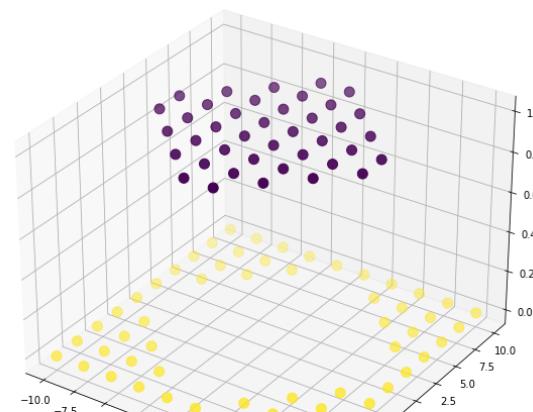
Example 3.1



Neural Network Interpretation

Classification

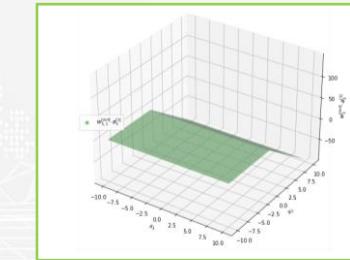
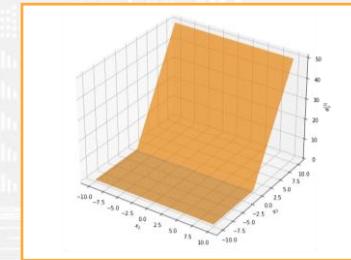
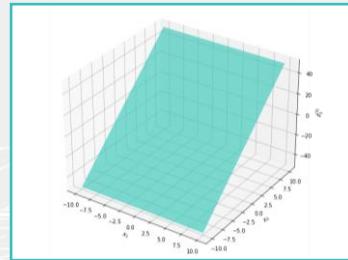
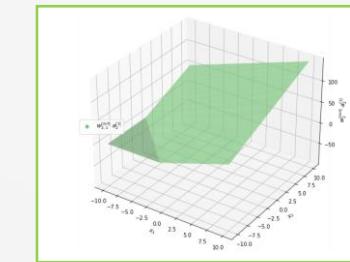
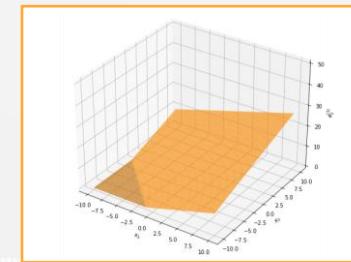
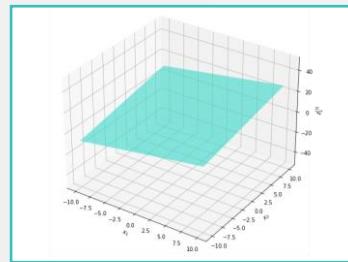
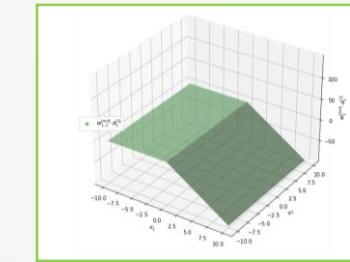
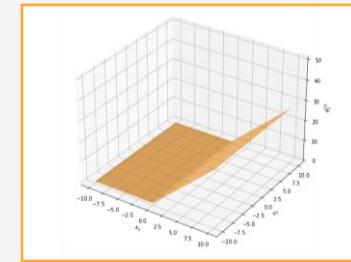
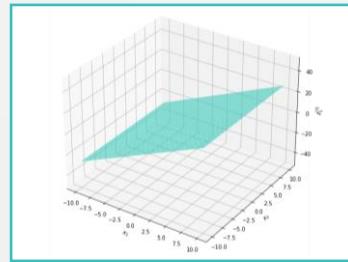
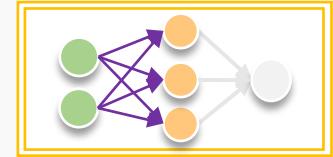
Example 3.2



Neural Network Interpretation

Classification

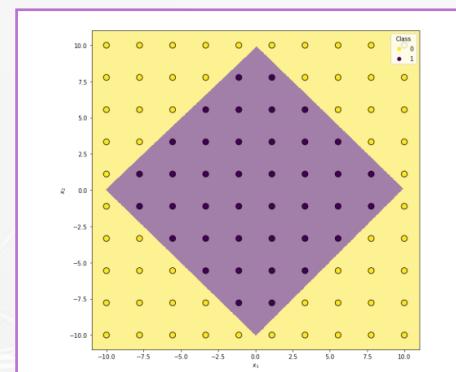
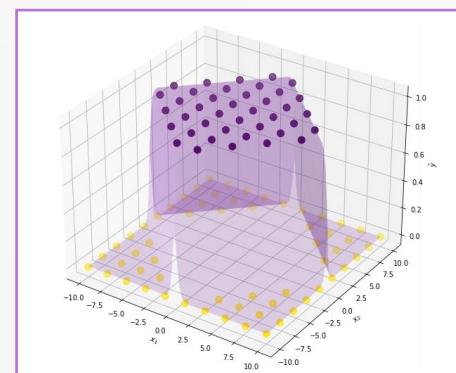
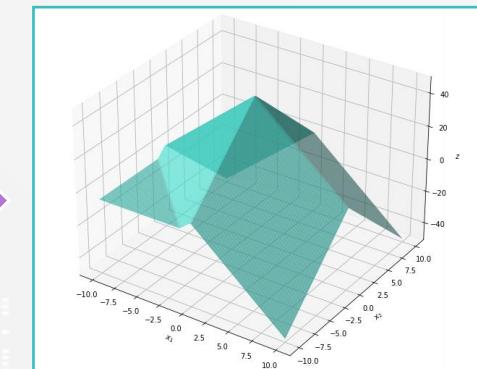
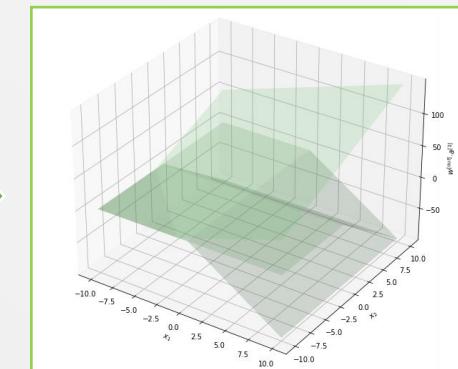
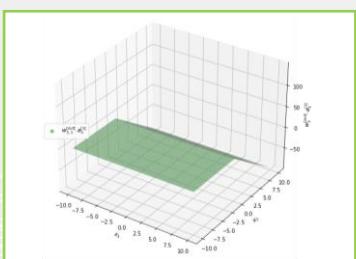
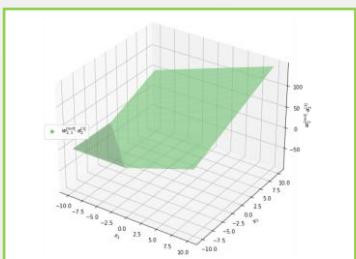
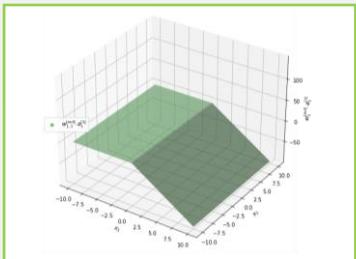
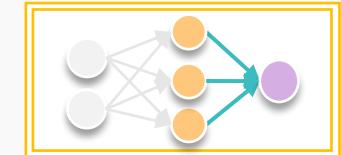
Example 3.2



Neural Network Interpretation

Classification

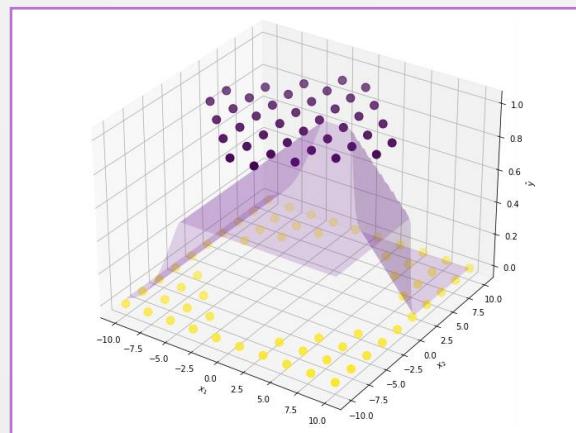
Example 3.2



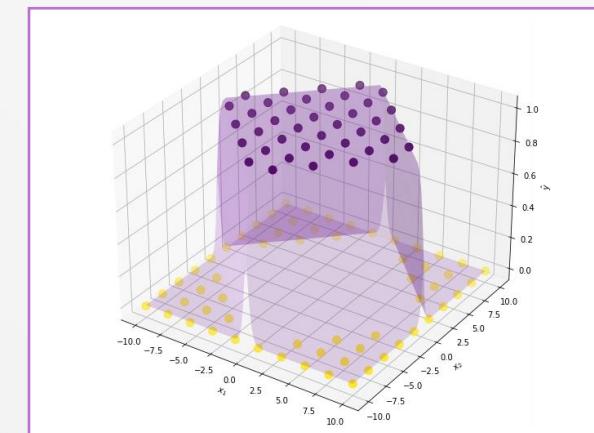
Neural Network Interpretation

Classification

กราฟแสดงค่า predicted ของ Example 3



Example 3.1 (2,2,1)

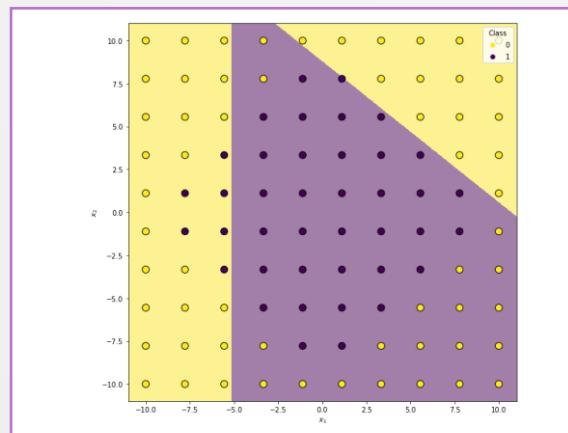


Example 3.2 (2,3,1)

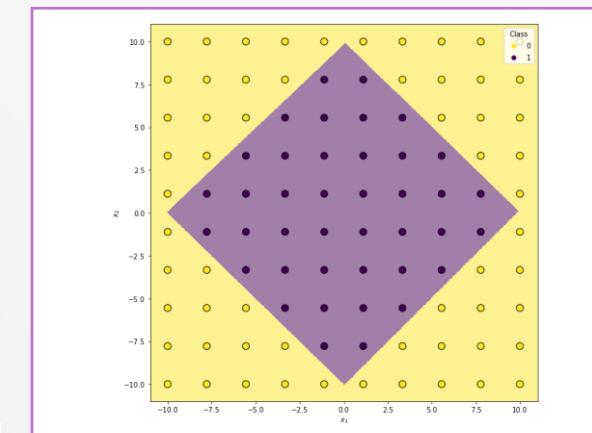
Neural Network Interpretation

Classification

กราฟแสดงพื้นที่ predicted ของ Example 3



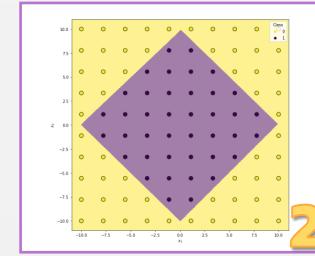
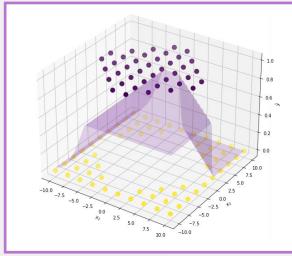
Example 3.1 (2,2,1)



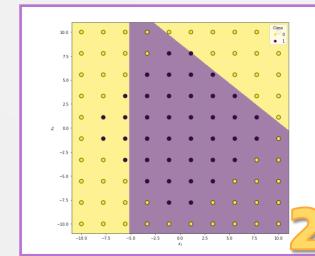
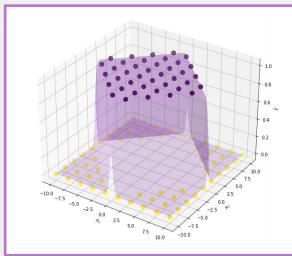
Example 3.2 (2,3,1)

Neural Network Interpretation

Classification



2,2,1



2,3,1

ข้อสรุปของ Example 3

- Plane (ส่วนของพื้นผิว) ที่เพิ่มขึ้นจากจำนวน node ที่มากขึ้น เมื่อผ่าน sigmoid function และจะทำให้เกิด 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

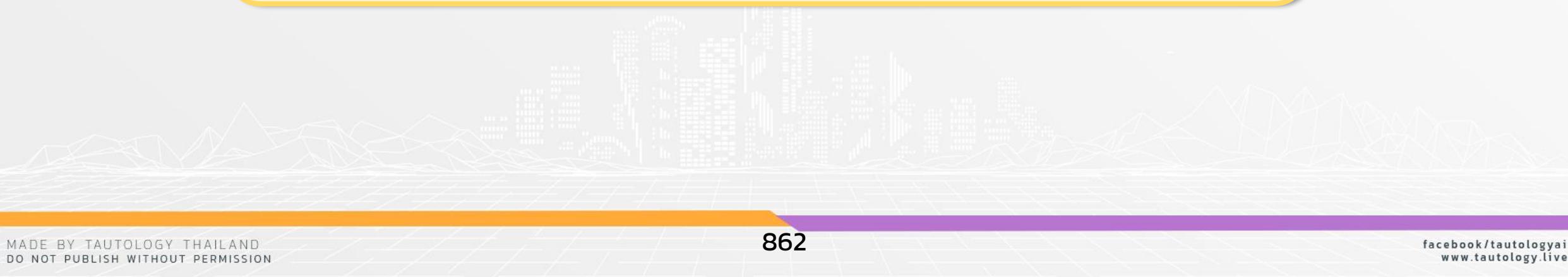
ทำให้เราได้ว่า เส้นแบ่งขอบเขต (border) ที่ใช้ในการแบ่ง region จะมีจำนวนมากที่สุดก็จะมีเท่ากับ “ $(n^2 + n + 2) / 2$ ” ด้วยเช่นกัน

Neural Network Interpretation



แต่ในการหาจำนวน maximum region ที่เกิดขึ้นใน 3 มิติ

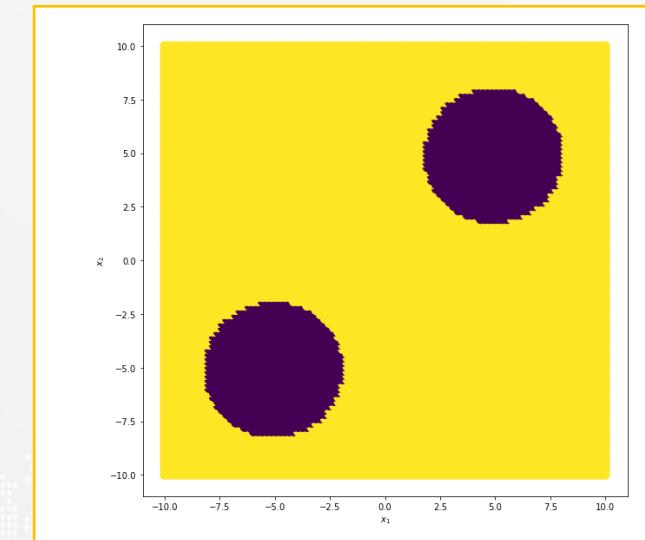
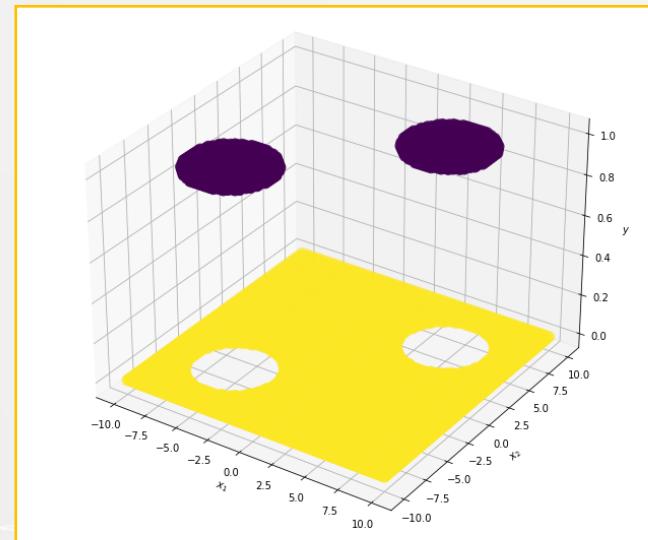
มีความซับซ้อนยิ่งการวิเคราะห์การประกอบกันของ plane
ที่ใช้ในการสร้าง region จึงขอลงไว้



Neural Network Interpretation

Classification

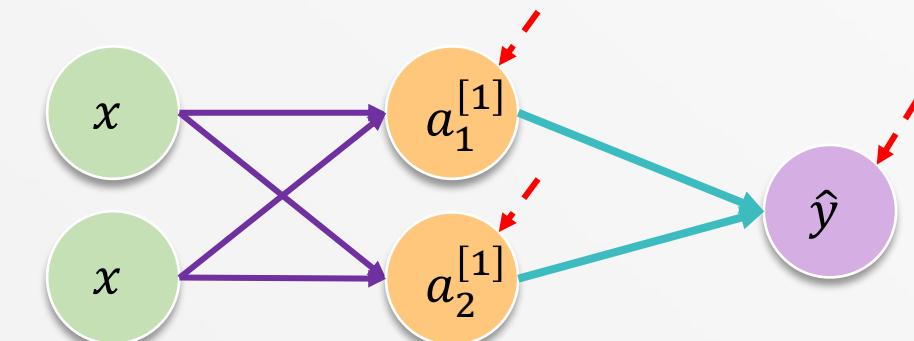
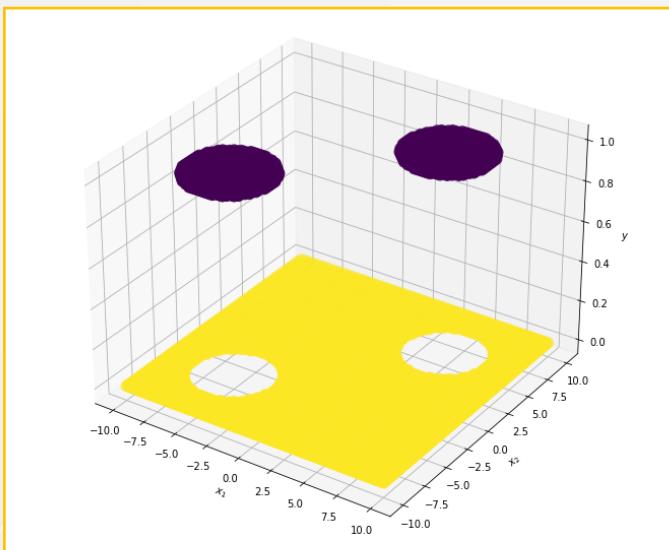
Example 4 : Binary Classification



Neural Network Interpretation

Classification

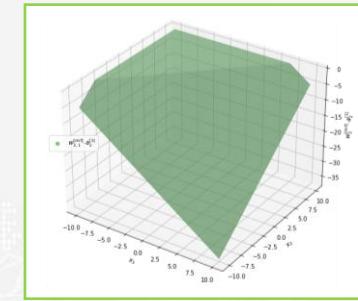
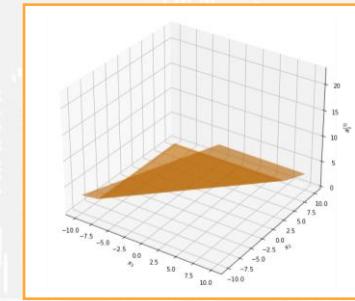
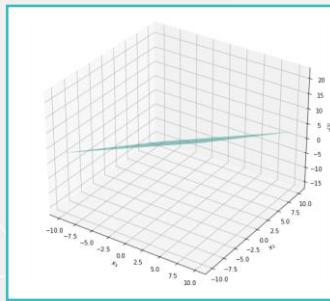
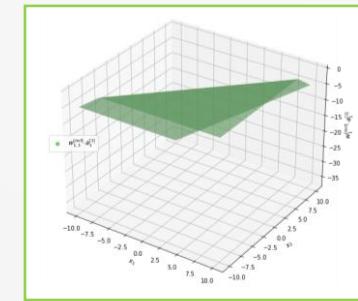
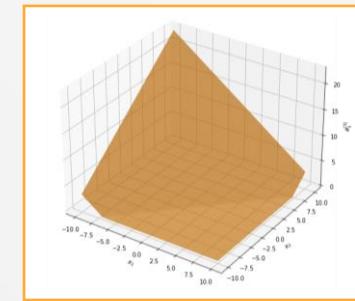
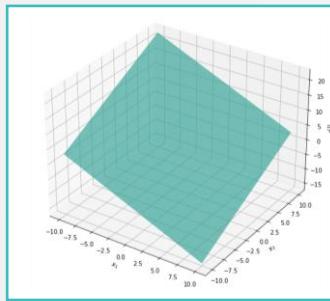
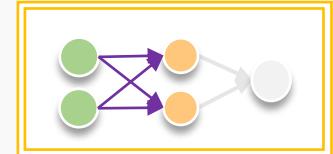
Example 4.1



Neural Network Interpretation

Classification

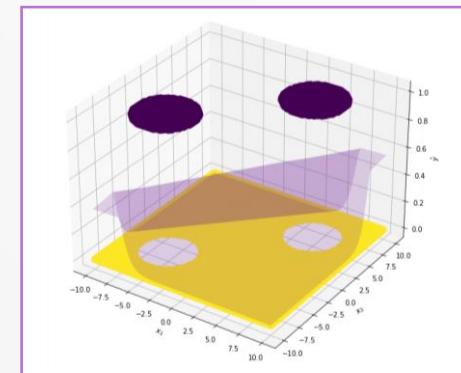
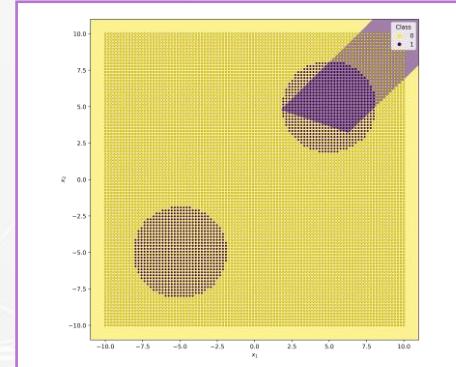
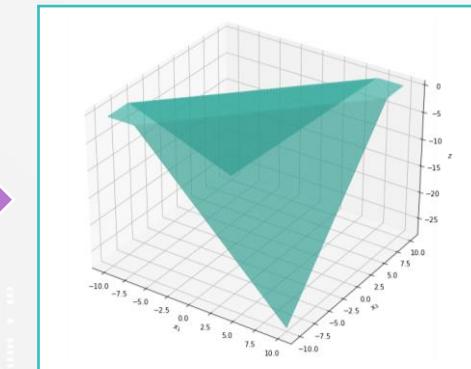
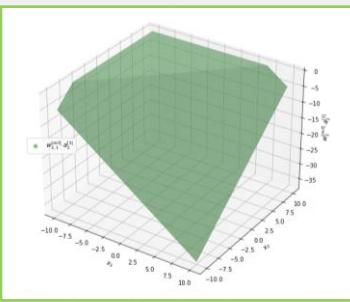
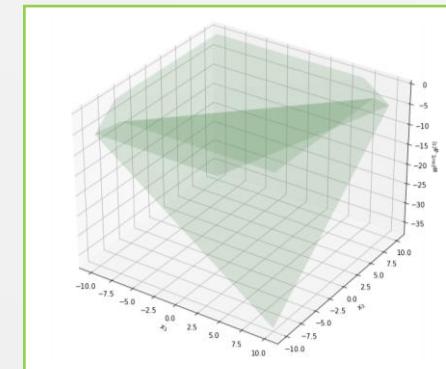
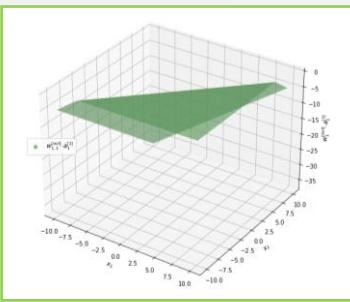
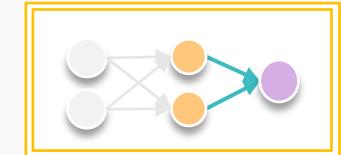
Example 4.1



Neural Network Interpretation

Classification

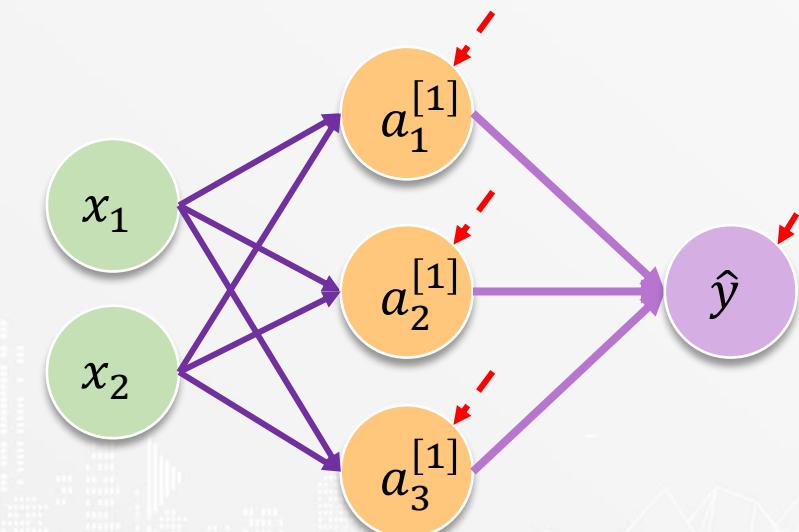
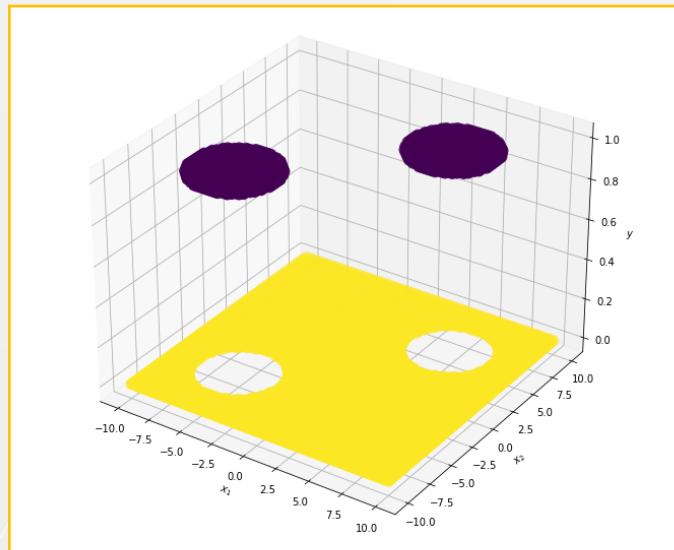
Example 4.1



Neural Network Interpretation

Classification

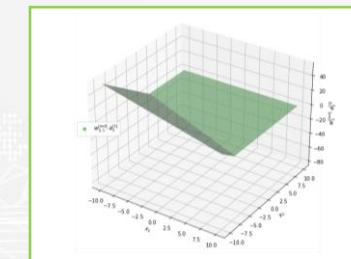
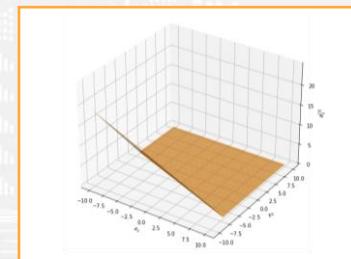
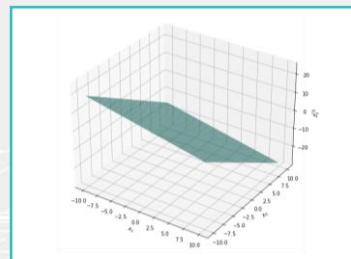
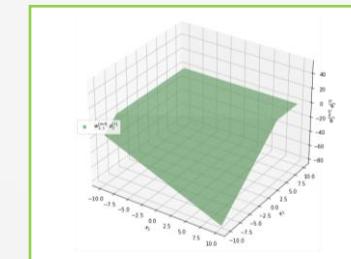
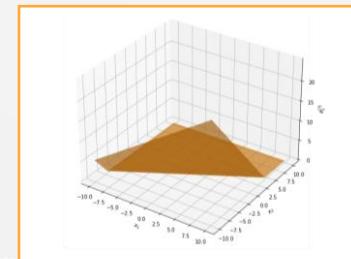
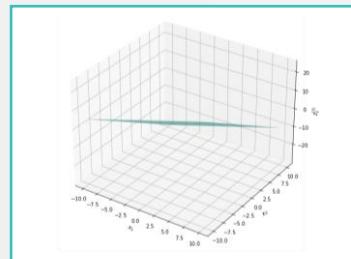
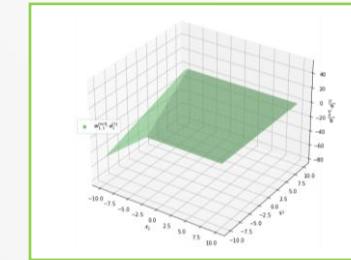
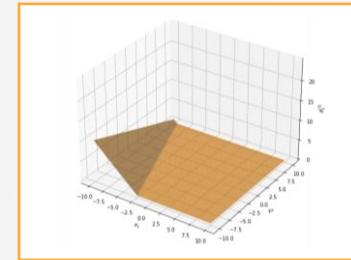
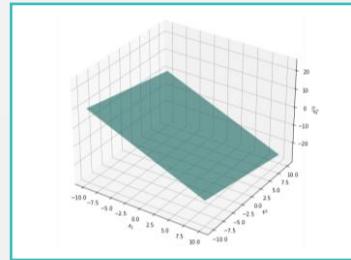
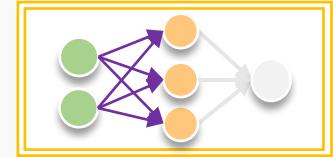
Example 4.2



Neural Network Interpretation

Classification

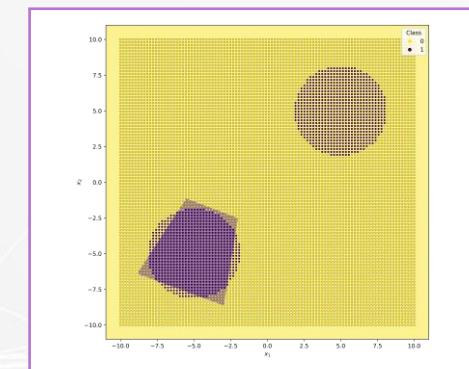
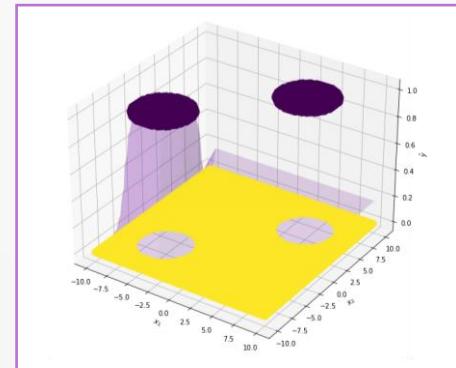
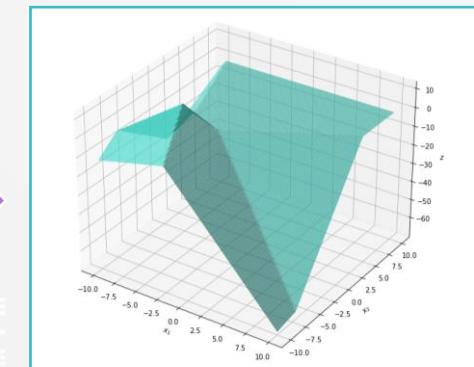
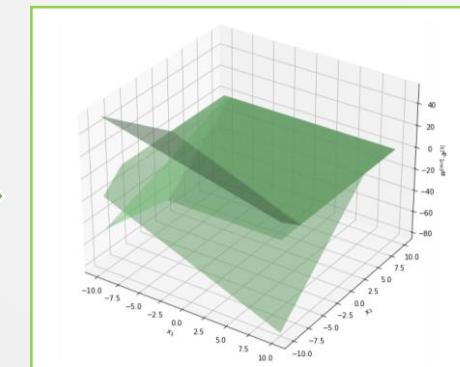
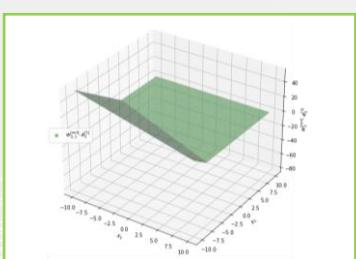
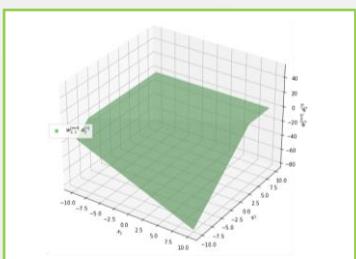
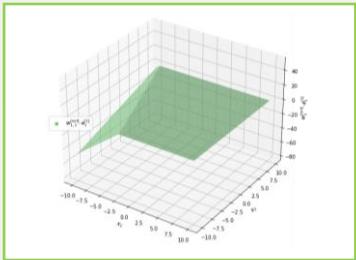
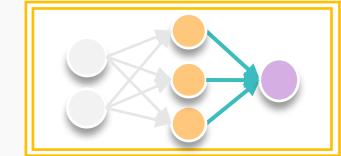
Example 4.2



Neural Network Interpretation

Classification

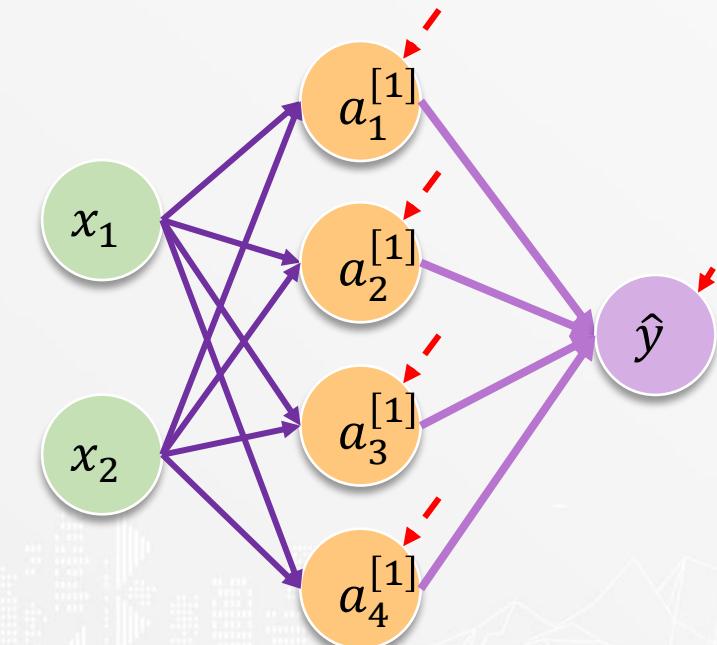
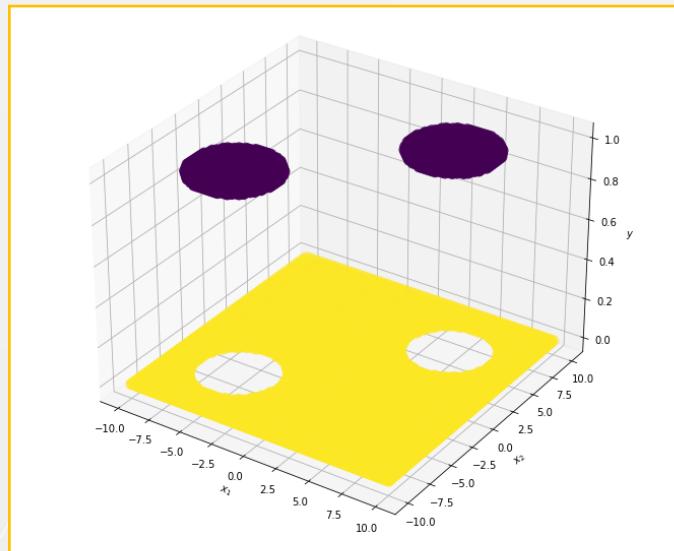
Example 4.2



Neural Network Interpretation

Classification

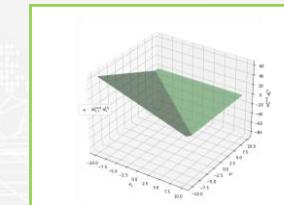
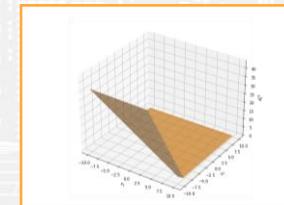
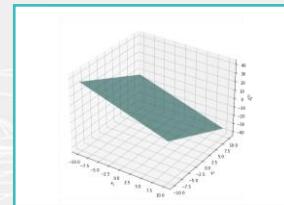
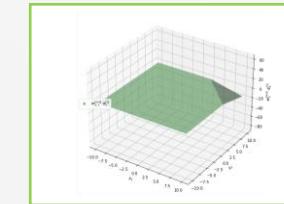
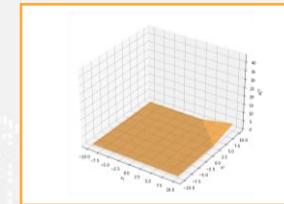
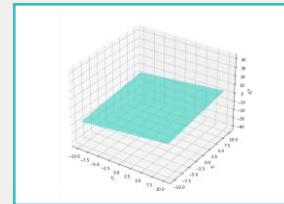
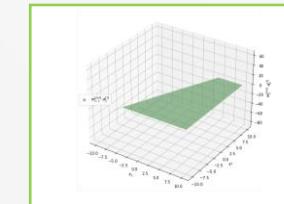
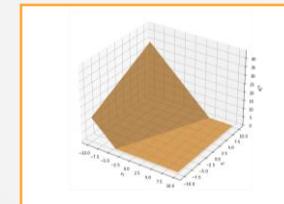
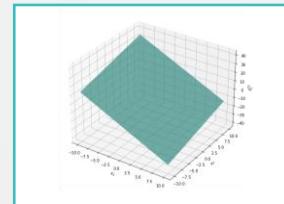
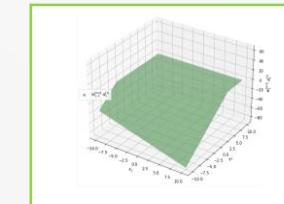
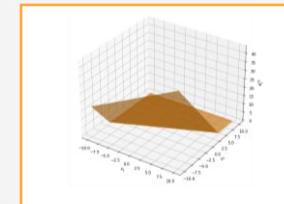
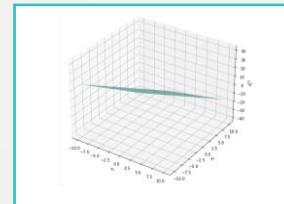
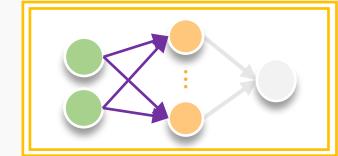
Example 4.3



Neural Network Interpretation

Classification

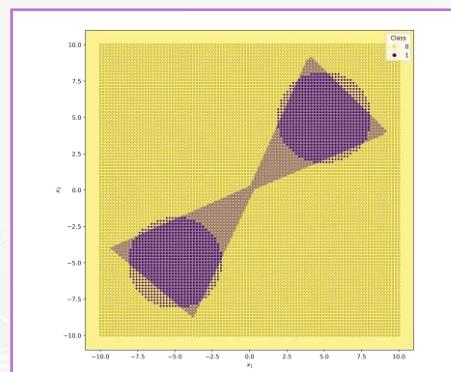
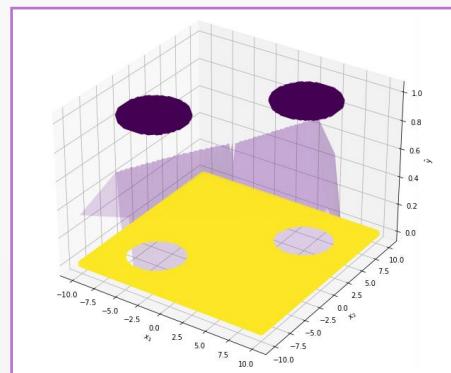
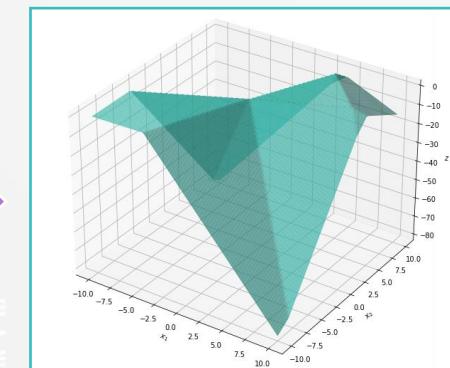
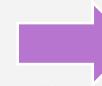
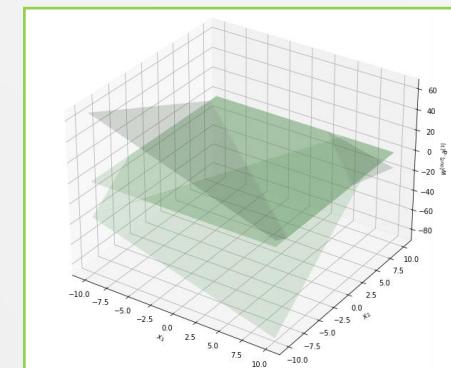
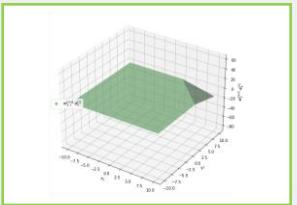
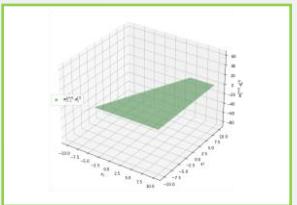
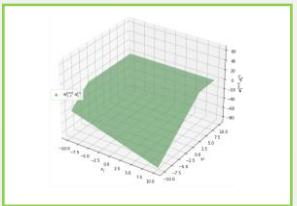
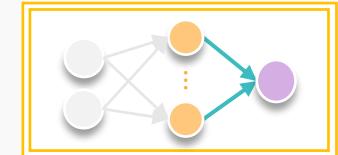
Example 4.3



Neural Network Interpretation

Classification

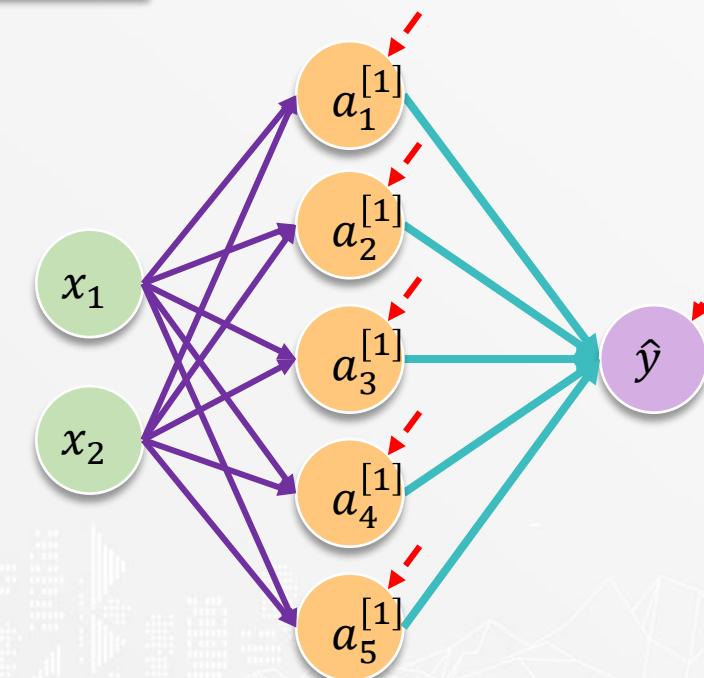
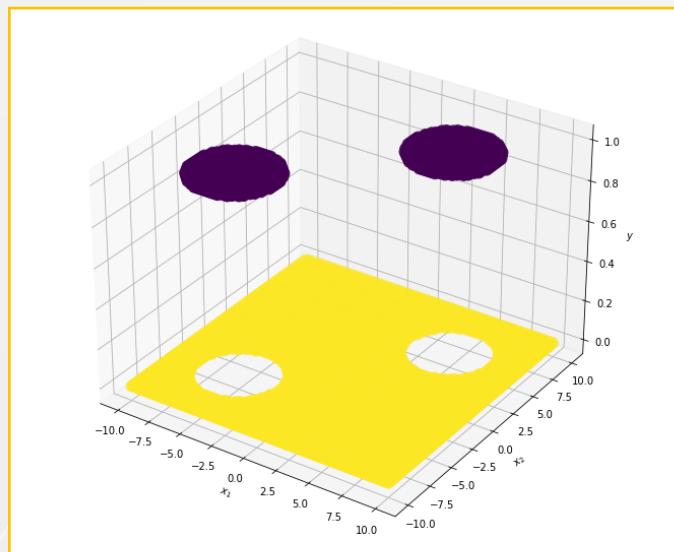
Example 4.3



Neural Network Interpretation

Classification

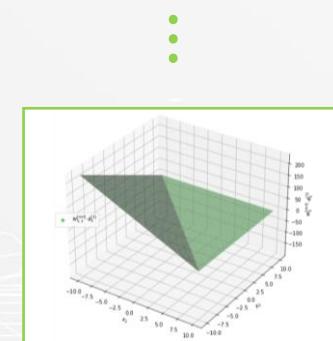
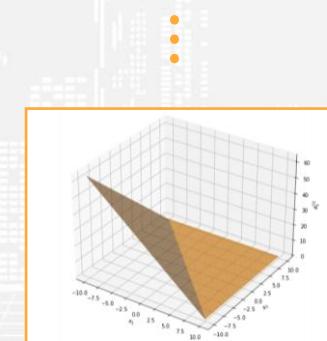
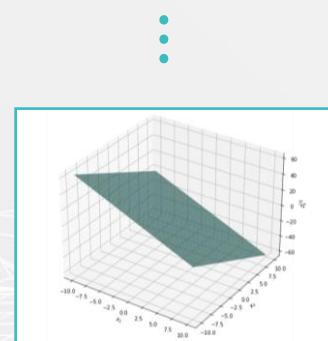
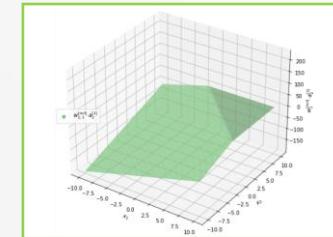
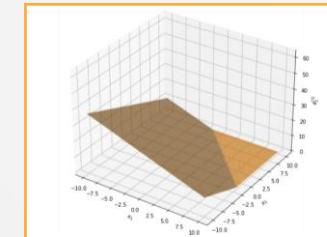
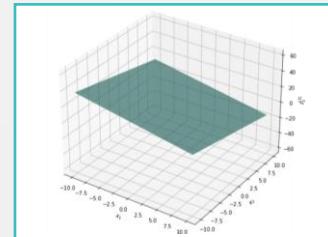
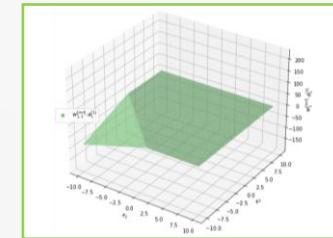
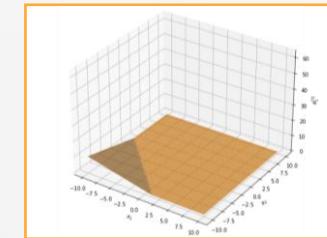
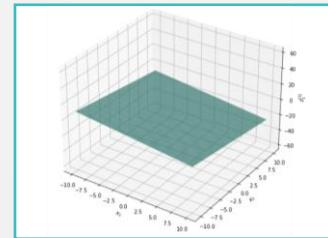
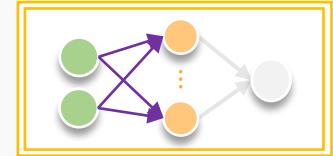
Example 4.4



Neural Network Interpretation

Classification

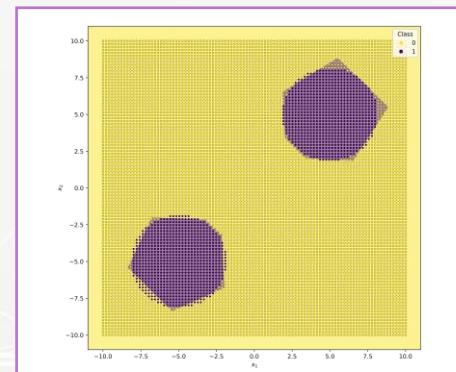
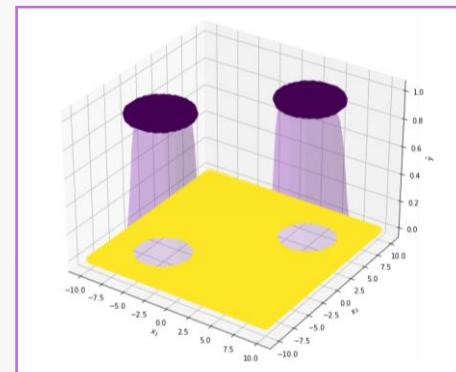
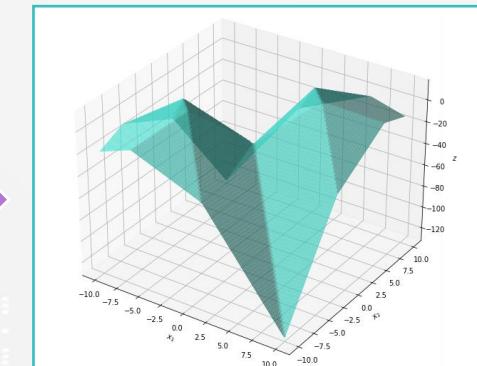
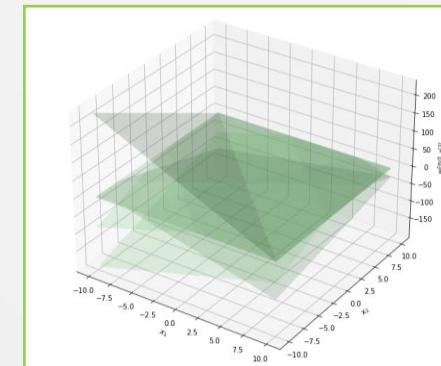
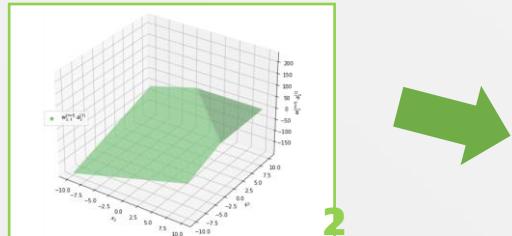
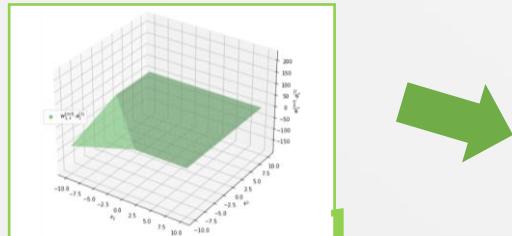
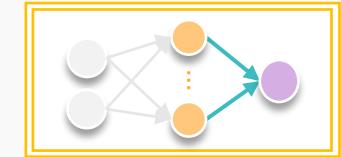
Example 4.4



Neural Network Interpretation

Classification

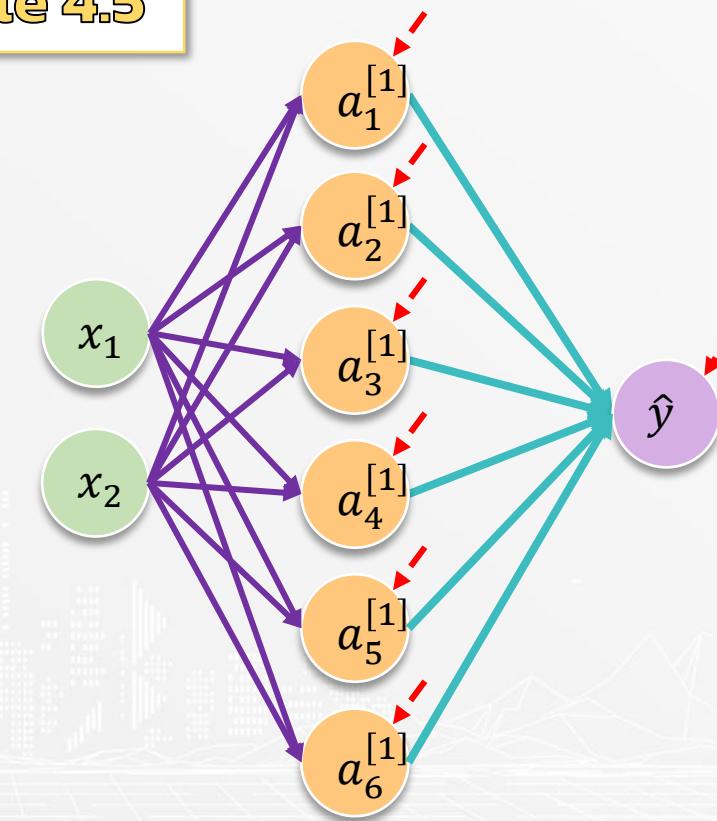
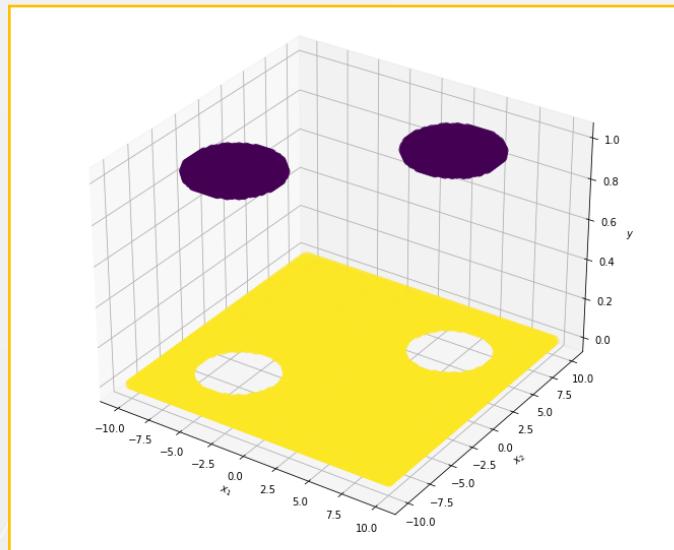
Example 4.4



Neural Network Interpretation

Classification

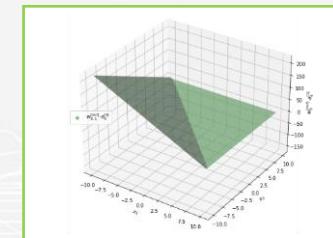
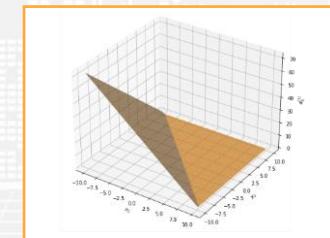
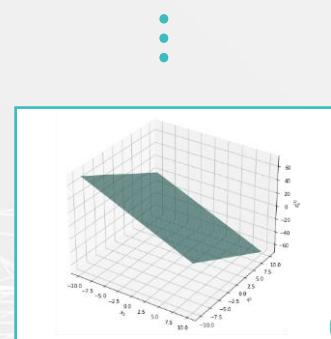
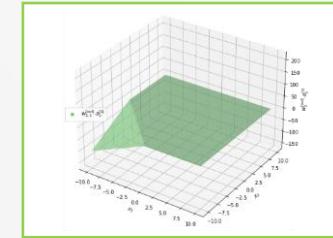
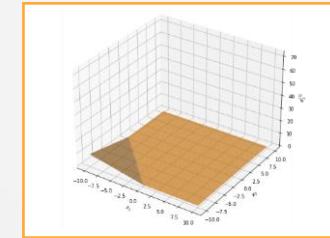
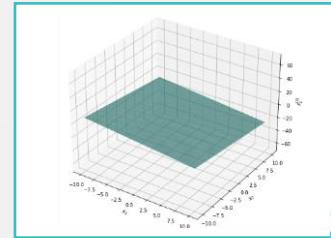
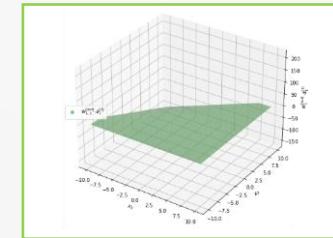
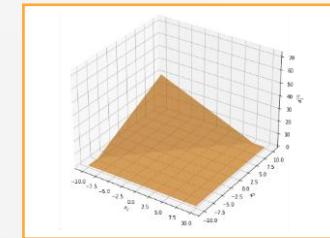
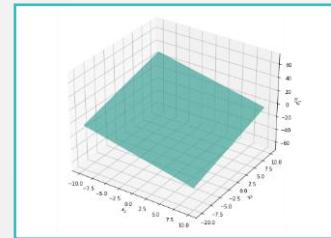
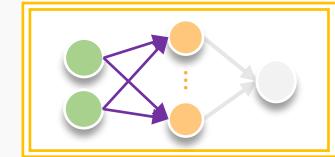
Example 4.5



Neural Network Interpretation

Classification

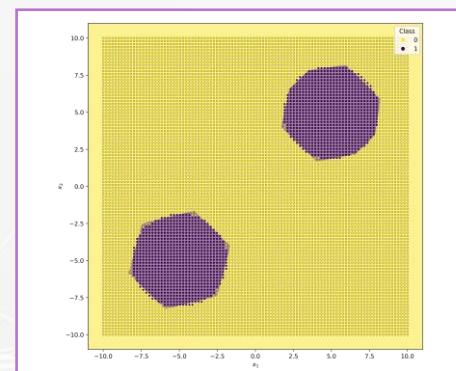
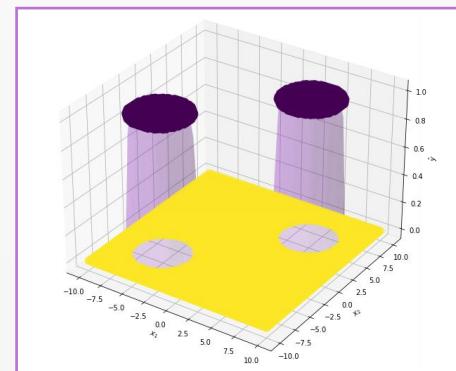
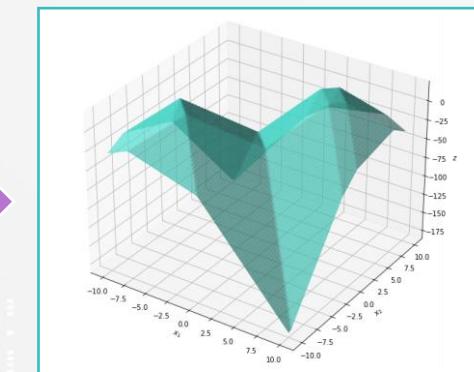
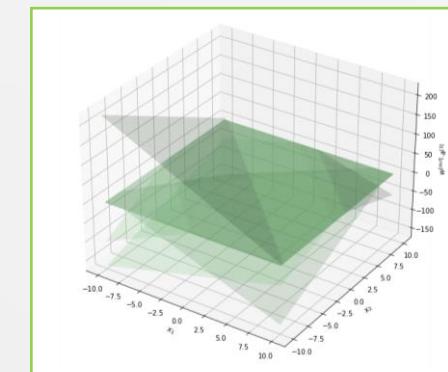
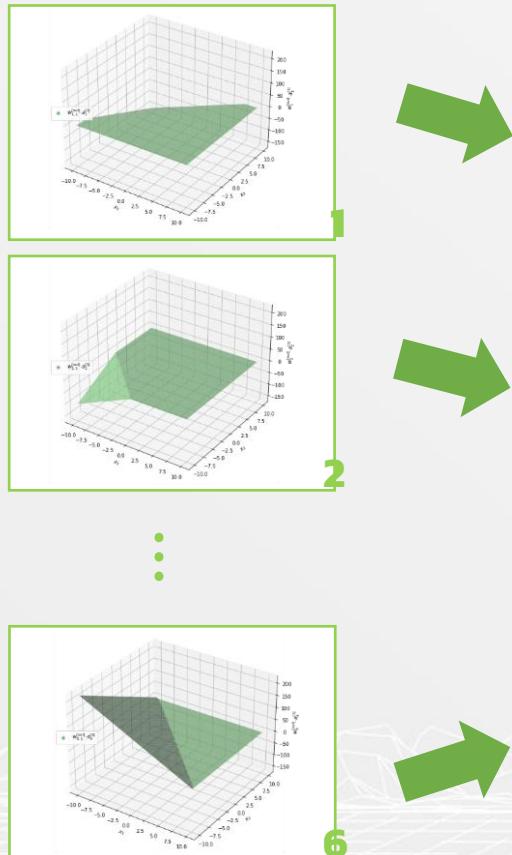
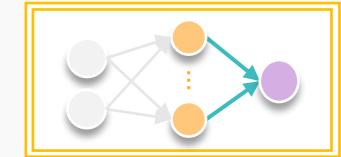
Example 4.5



Neural Network Interpretation

Classification

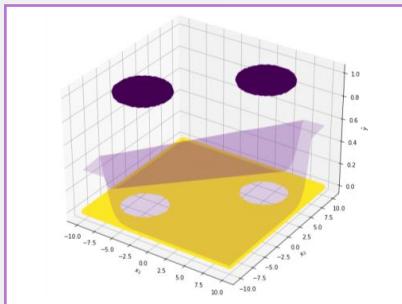
Example 4.5



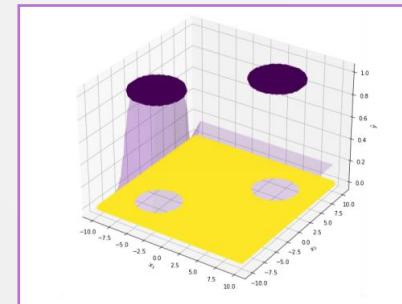
Neural Network Interpretation

Classification

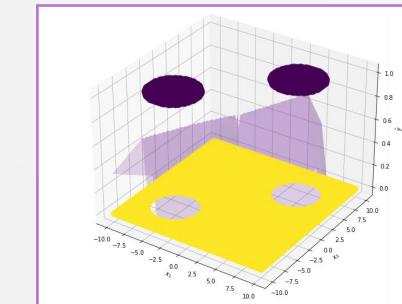
กราฟแสดงค่า predicted ของ Example 4



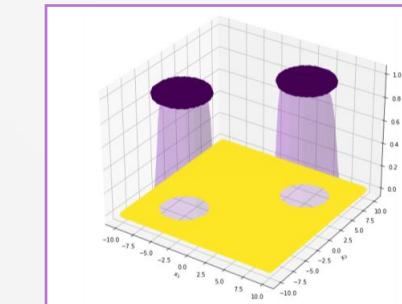
Example 5.1 (1,2,1)



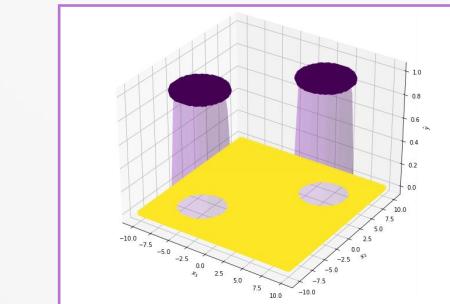
Example 5.2 (1,3,1)



Example 5.3 (1,4,1)



Example 5.4 (1,5,1)

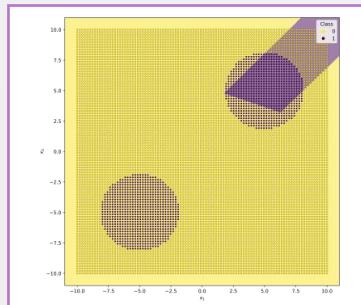


Example 5.5 (1,6,1)

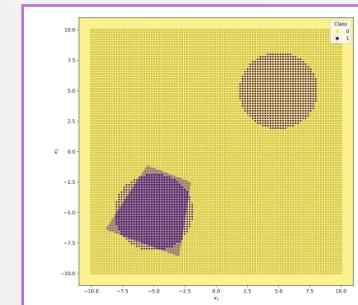
Neural Network Interpretation

Classification

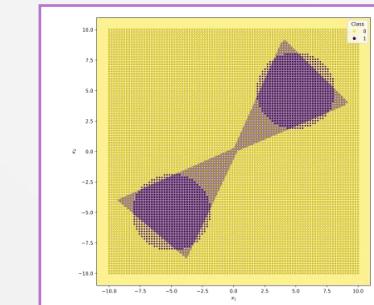
กราฟแสดงพื้นที่ predicted ของ Example 4



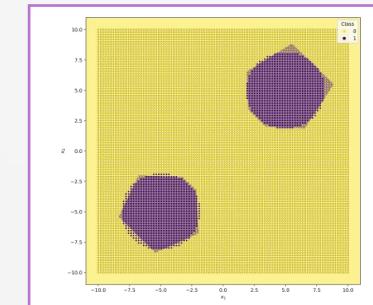
Example 4.1 (2,2,1)



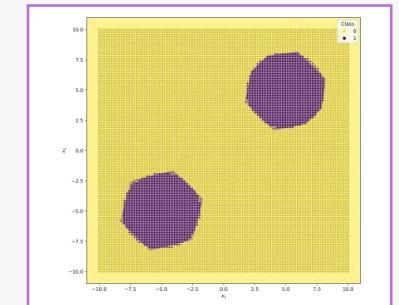
Example 4.2 (2,3,1)



Example 4.3 (2,4,1)



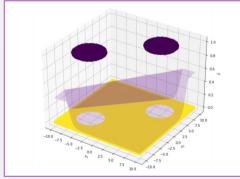
Example 4.4 (2,5,1)



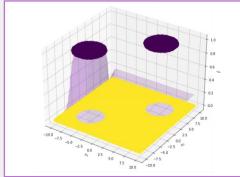
Example 4.5 (2,6,1)

Neural Network Interpretation

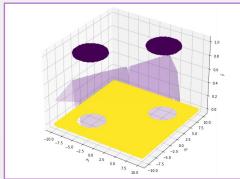
Classification



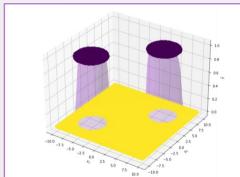
2,2,1



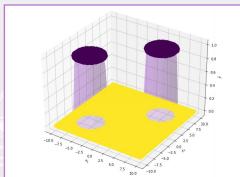
2,3,1



2,4,1



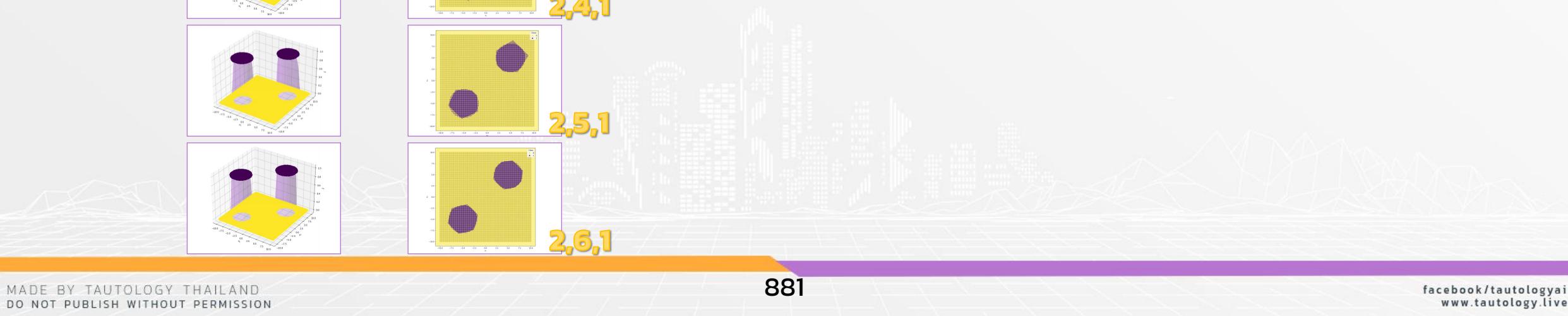
2,5,1



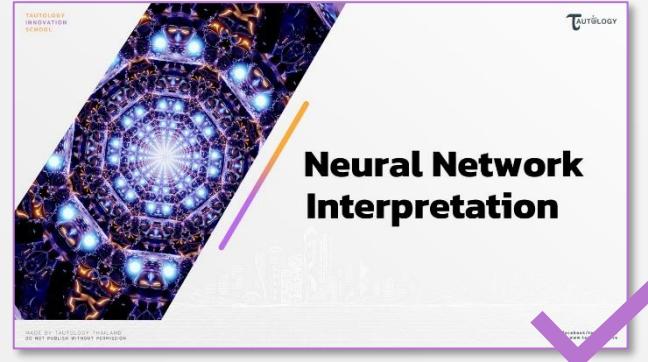
2,6,1

ข้อสรุปของ Example 4

- ทั้งนี้การที่จะเกิด maximum border นั้นขึ้นอยู่กับ dataset และ initial parameter ด้วย



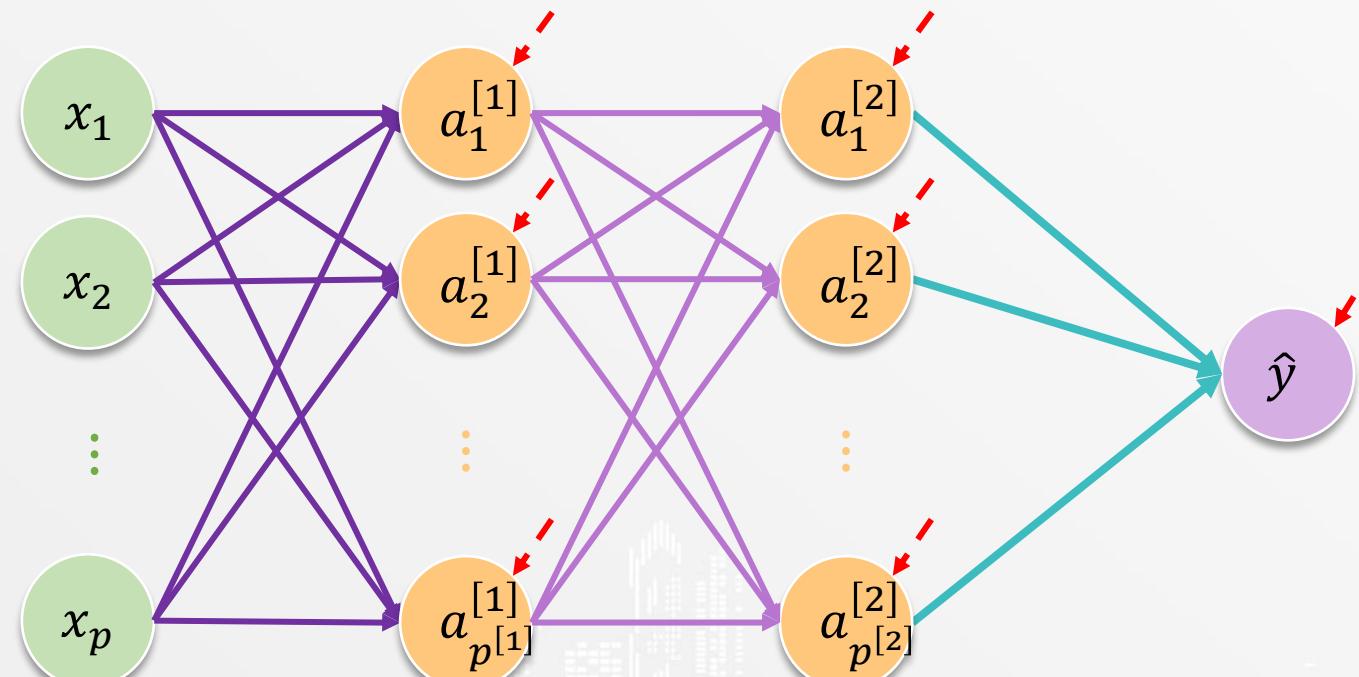
Deep Learning Interpretation





Deep Learning Interpretation

Deep Learning Interpretation

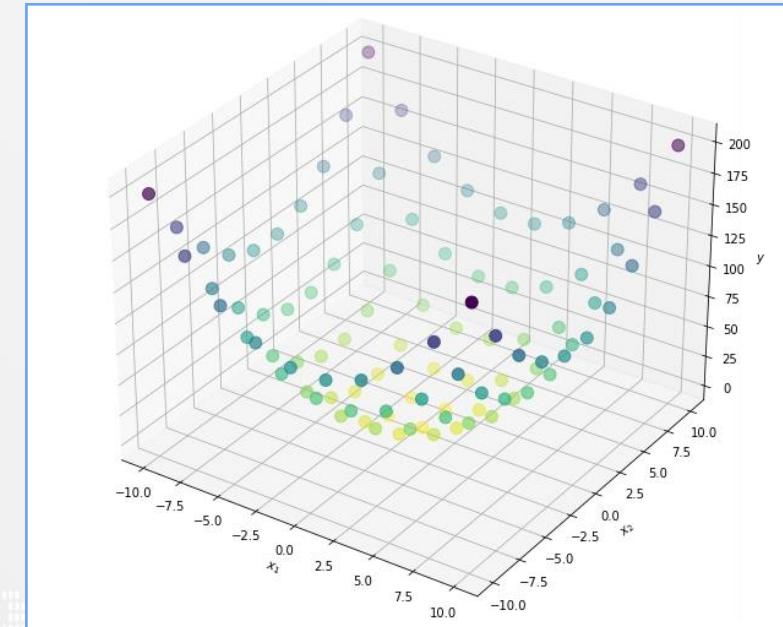


Deep Learning Interpretation

จำนวน hidden layer
ส่งผลอะไรต่อ model ?

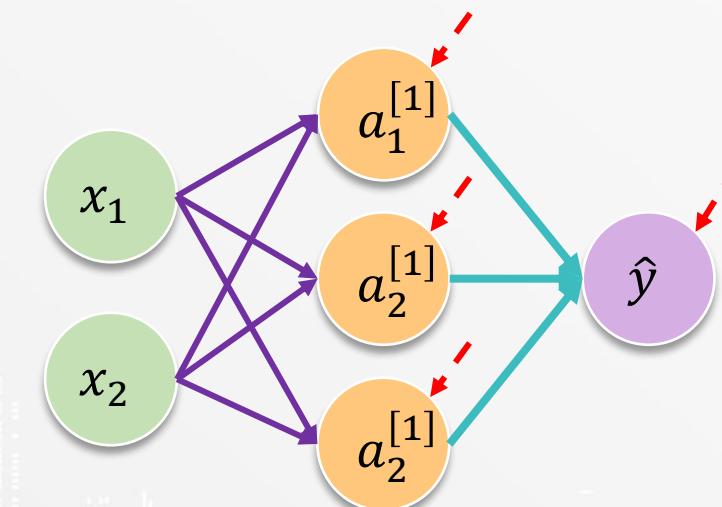
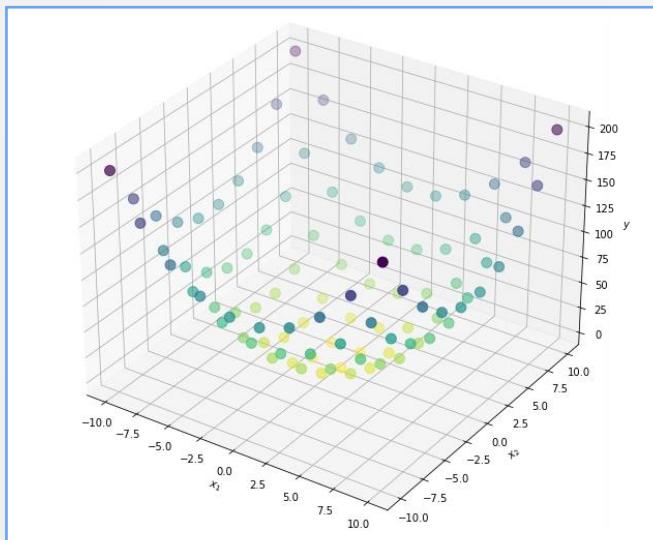
Deep Learning Interpretation

Example 1



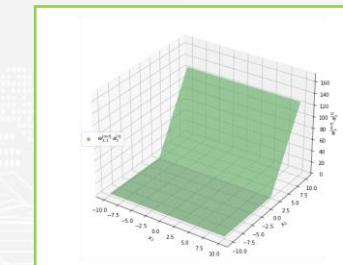
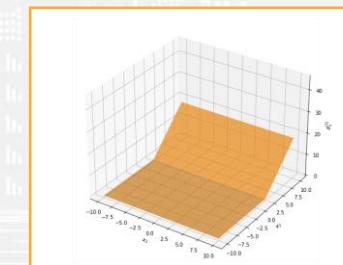
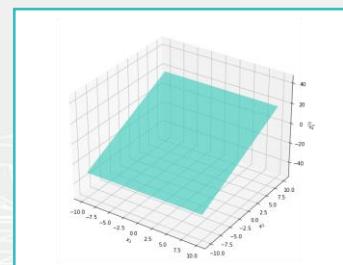
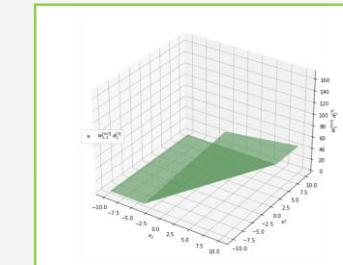
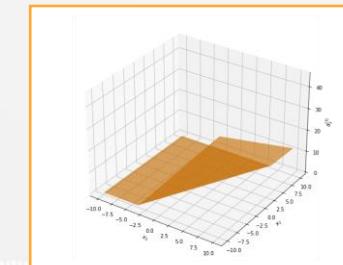
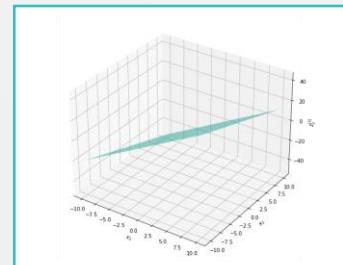
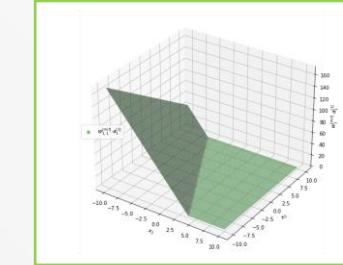
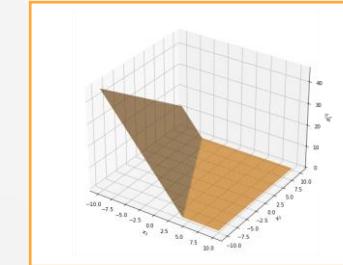
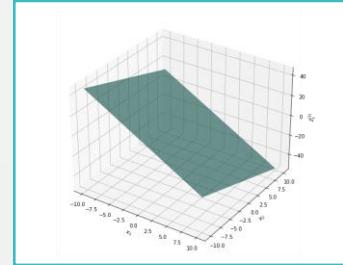
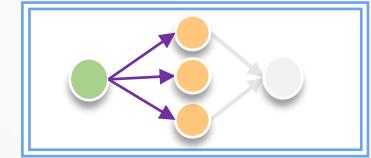
Deep Learning Interpretation

Example 1.1



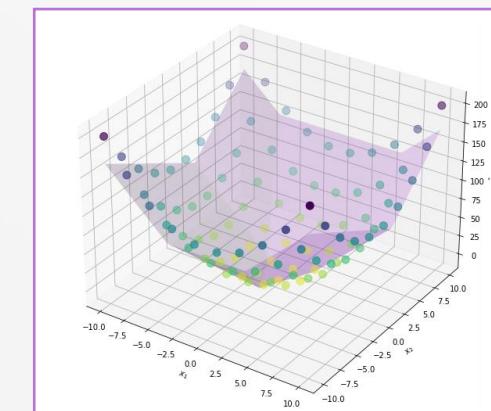
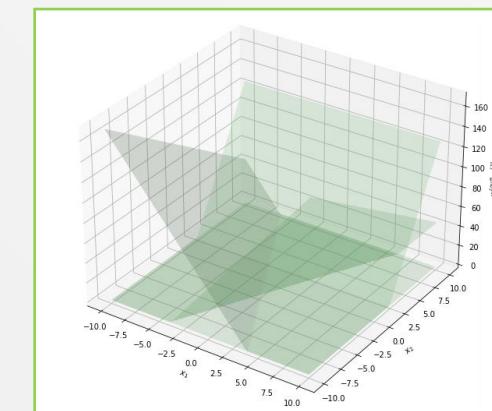
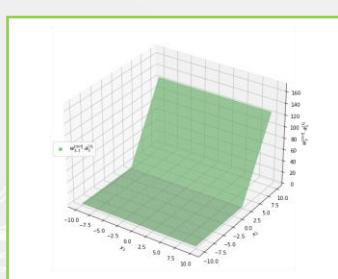
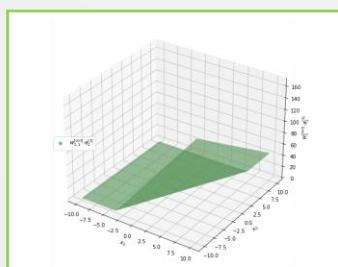
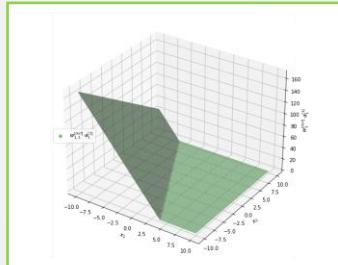
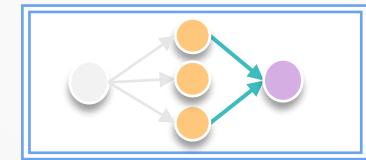
Deep Learning Interpretation

Example 1.1



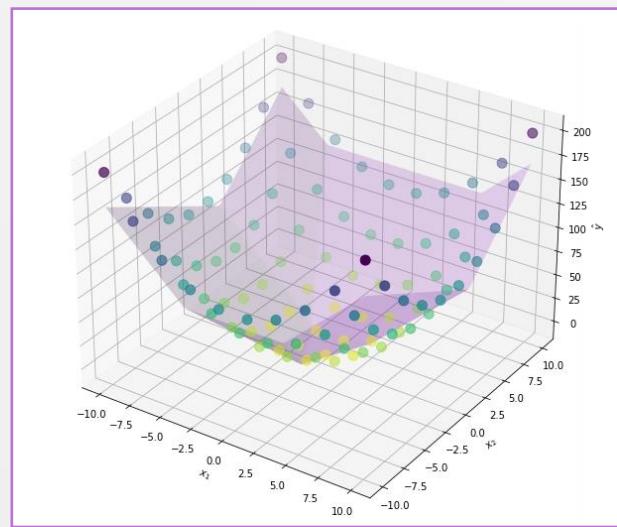
Deep Learning Interpretation

Example 1.1

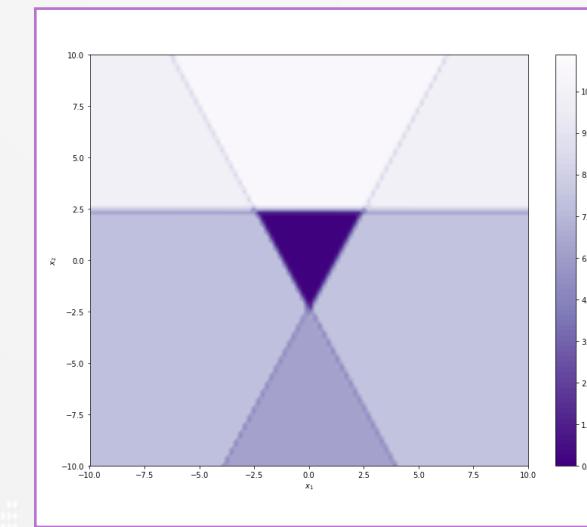


Deep Learning Interpretation

Example 1.1



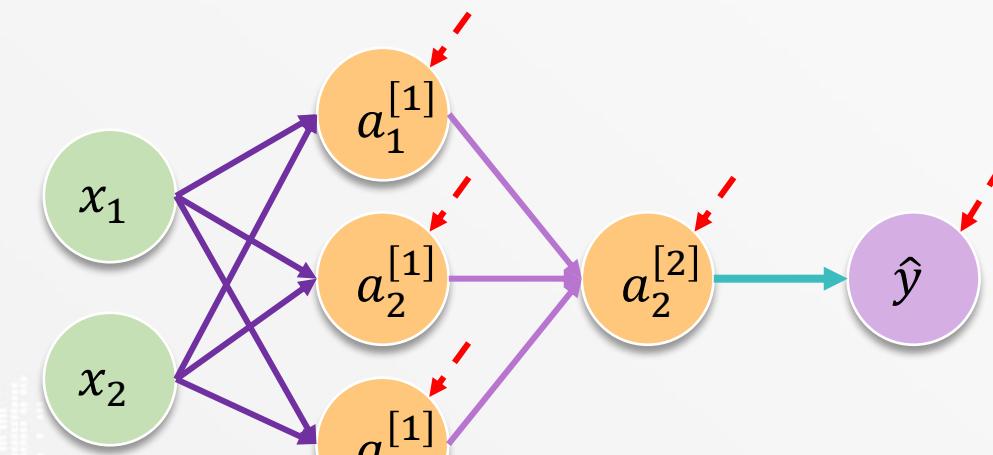
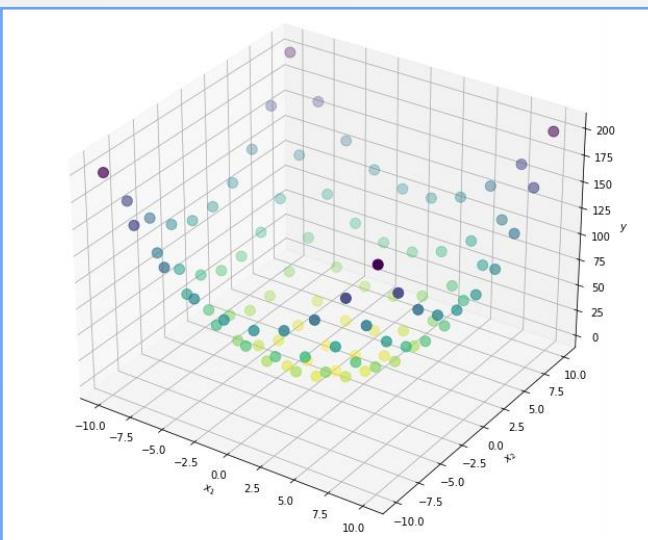
กราฟแสดงค่า predicted



Contour plot ของ predicted

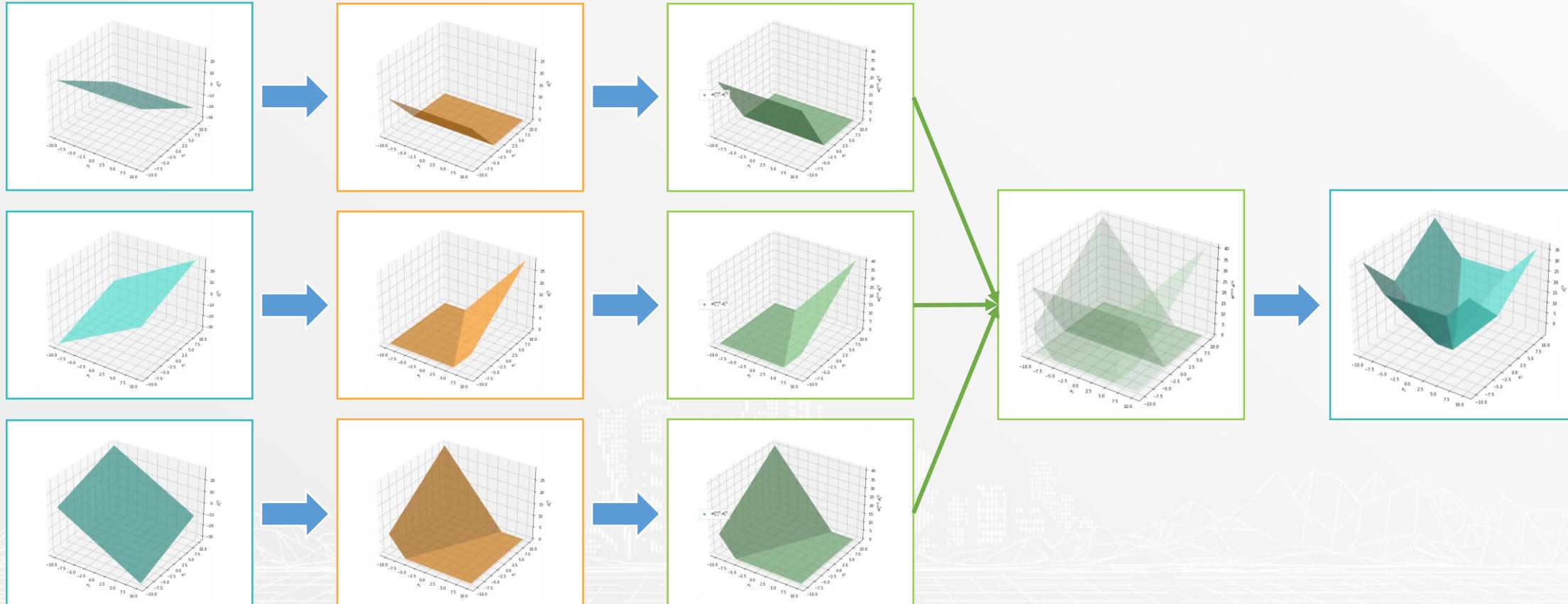
Deep Learning Interpretation

Example 1.2



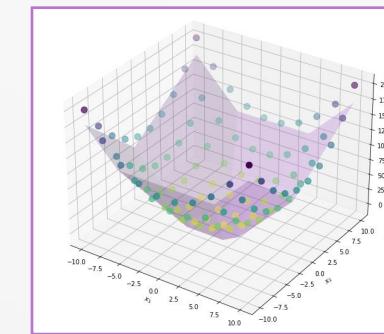
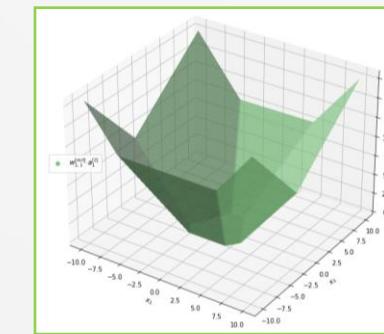
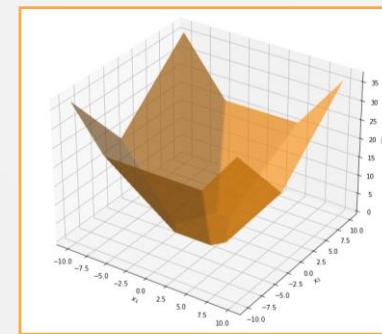
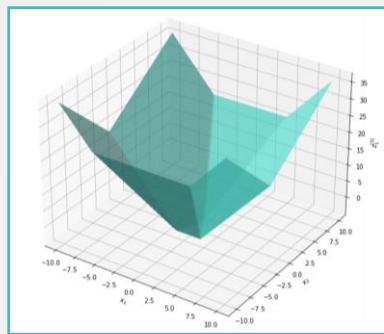
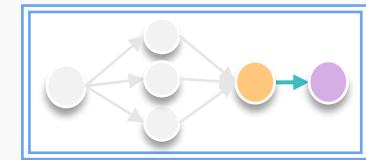
Deep Learning Interpretation

Example 1.2



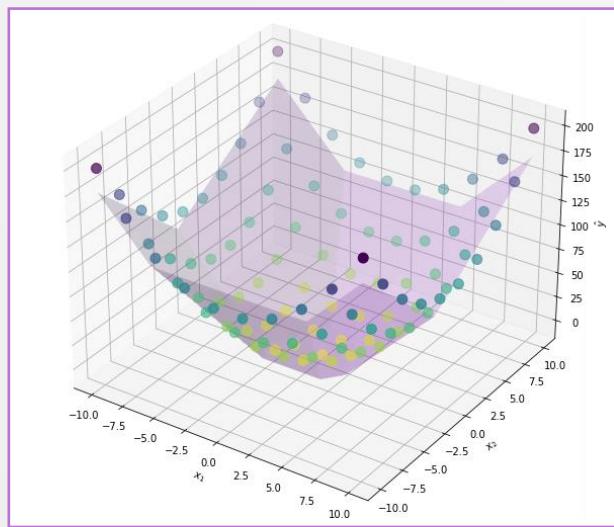
Deep Learning Interpretation

Example 1.2

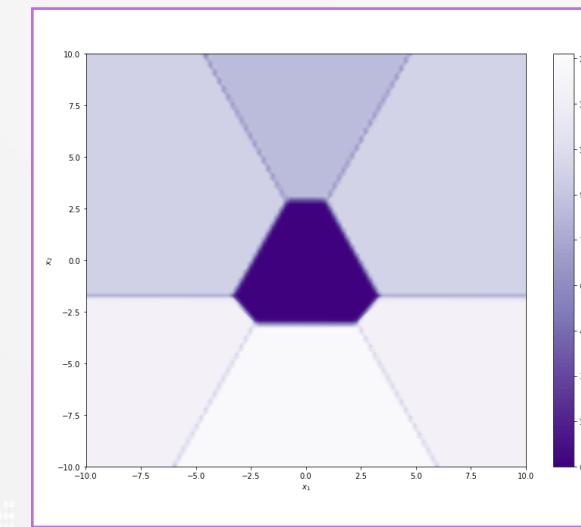


Deep Learning Interpretation

Example 1.2

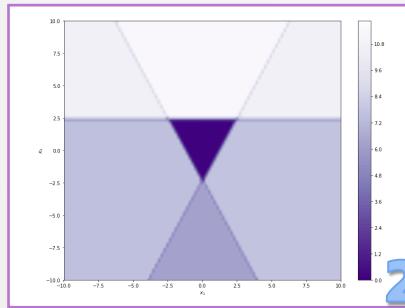
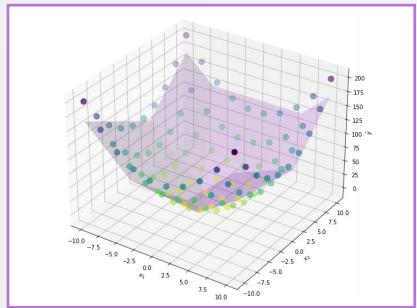


กราฟแสดงค่า predicted

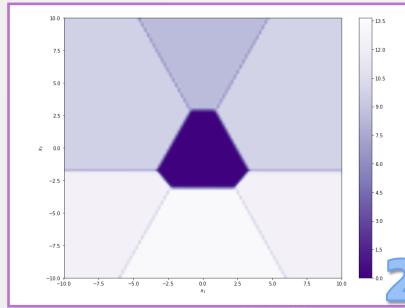
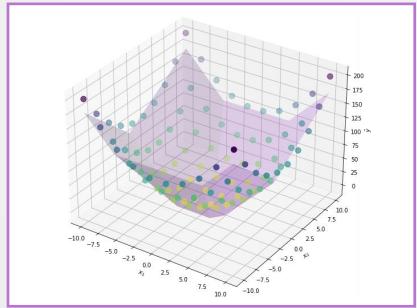


Contour plot ของ predicted

Deep Learning Interpretation



2,3,1



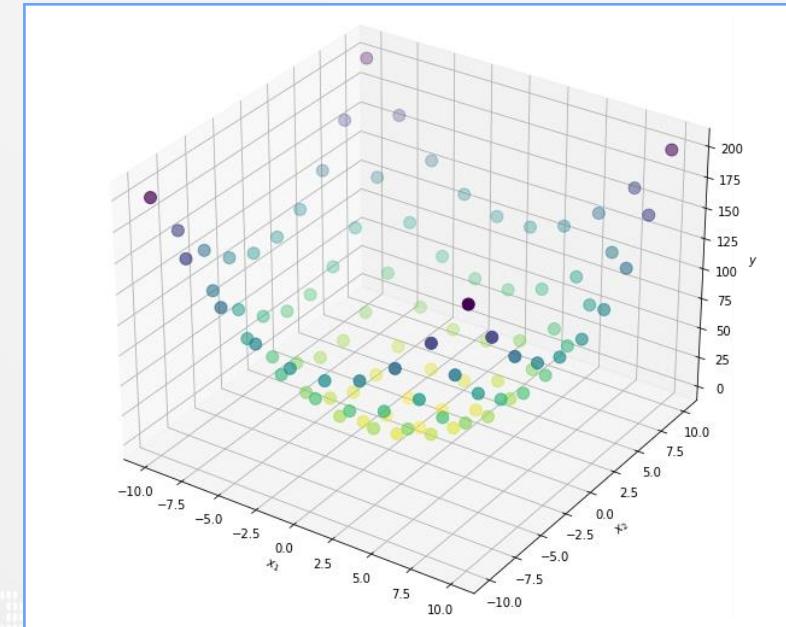
2,3,1,1

REMARK

การเพิ่ม hidden layer ทำให้เกิด plane ย่อย เพิ่มขึ้น โดยเกิดจาก การทำซ้ำของการสร้างรอยหักใน hidden layer ขึ้นต่อมา

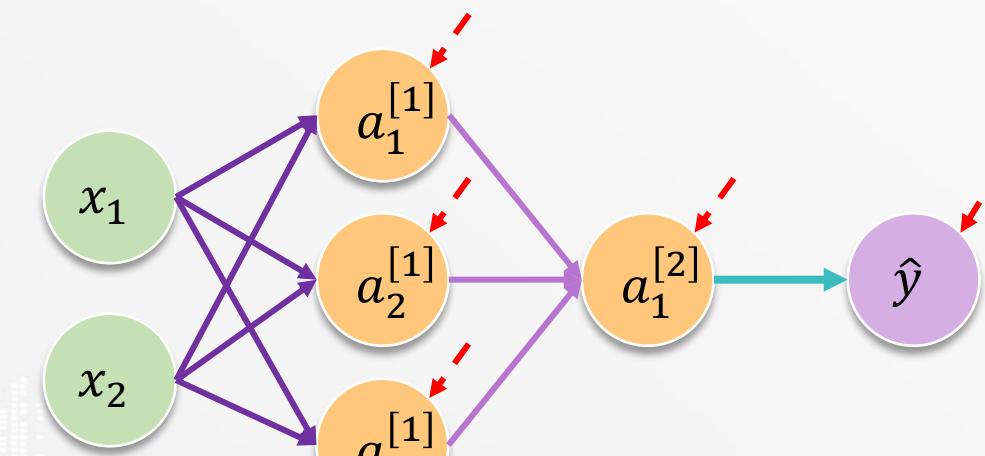
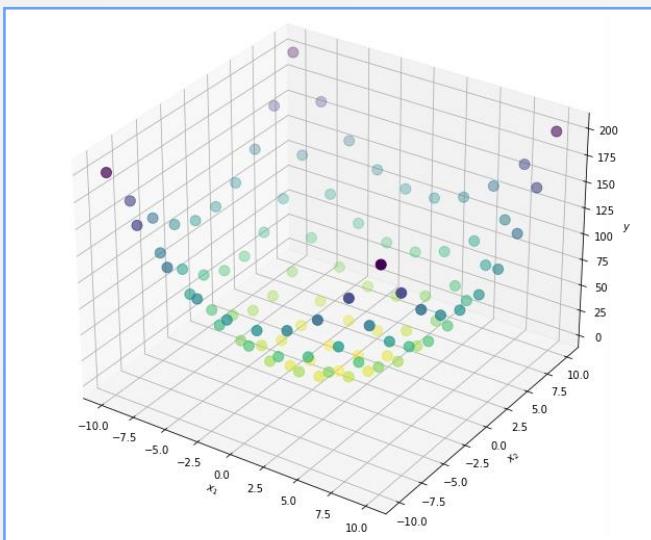
Deep Learning Interpretation

Example 2



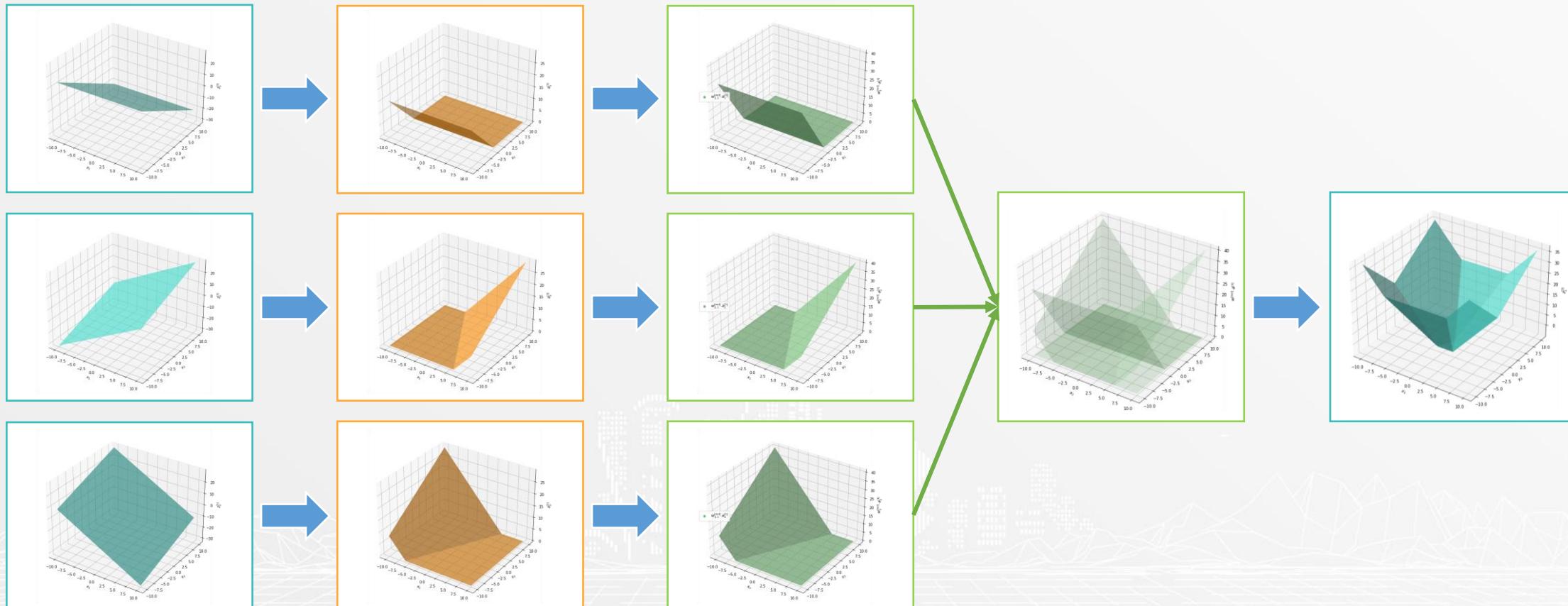
Deep Learning Interpretation

Example 2



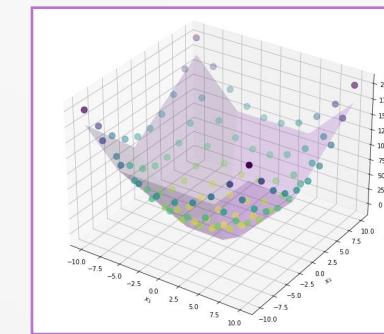
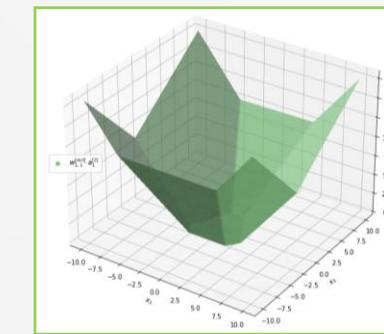
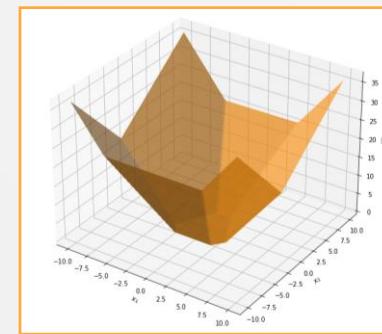
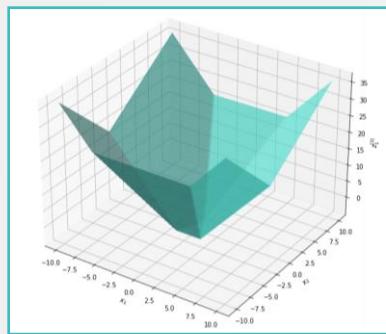
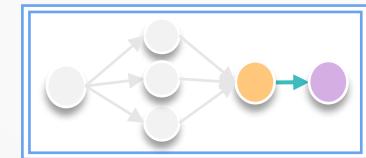
Deep Learning Interpretation

Example 2.1



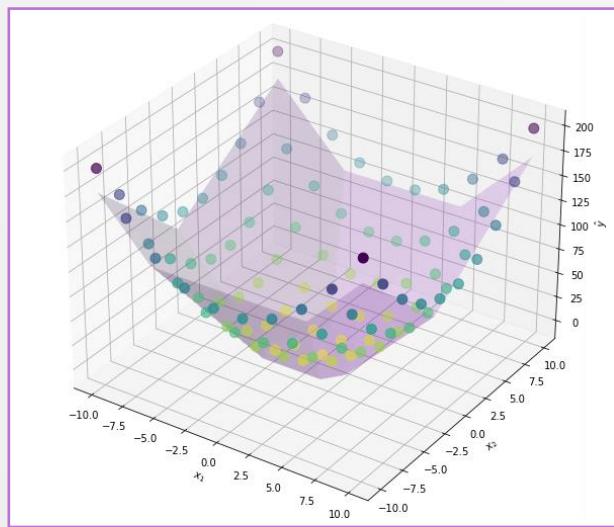
Deep Learning Interpretation

Example 2.1

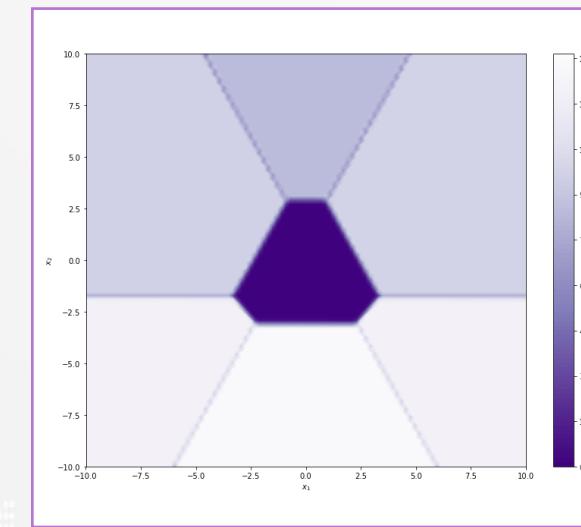


Deep Learning Interpretation

Example 2.1



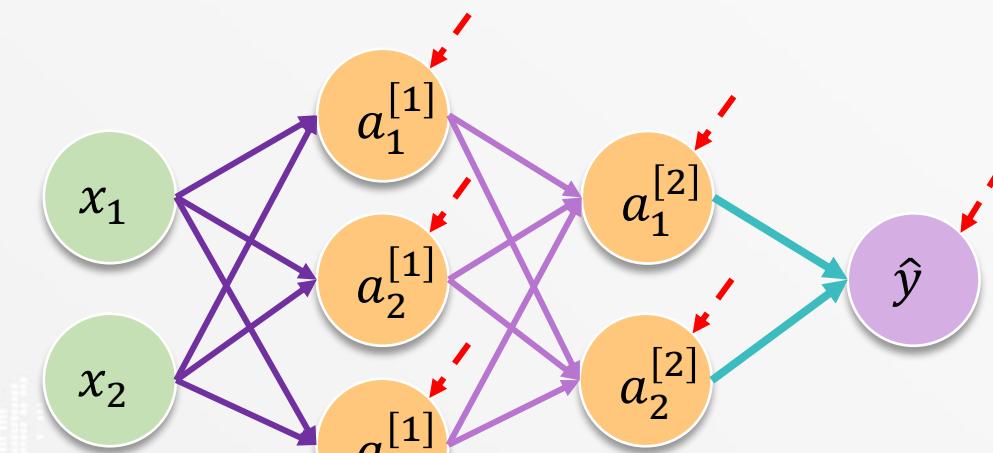
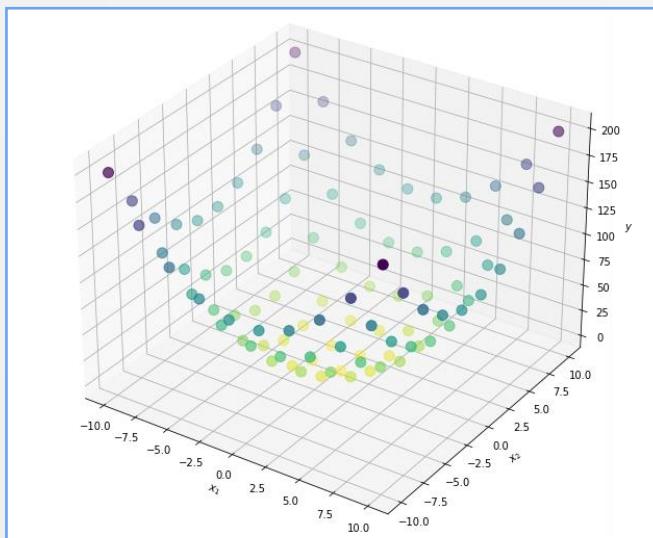
กราฟแสดงค่า predicted



Contour plot ของ predicted

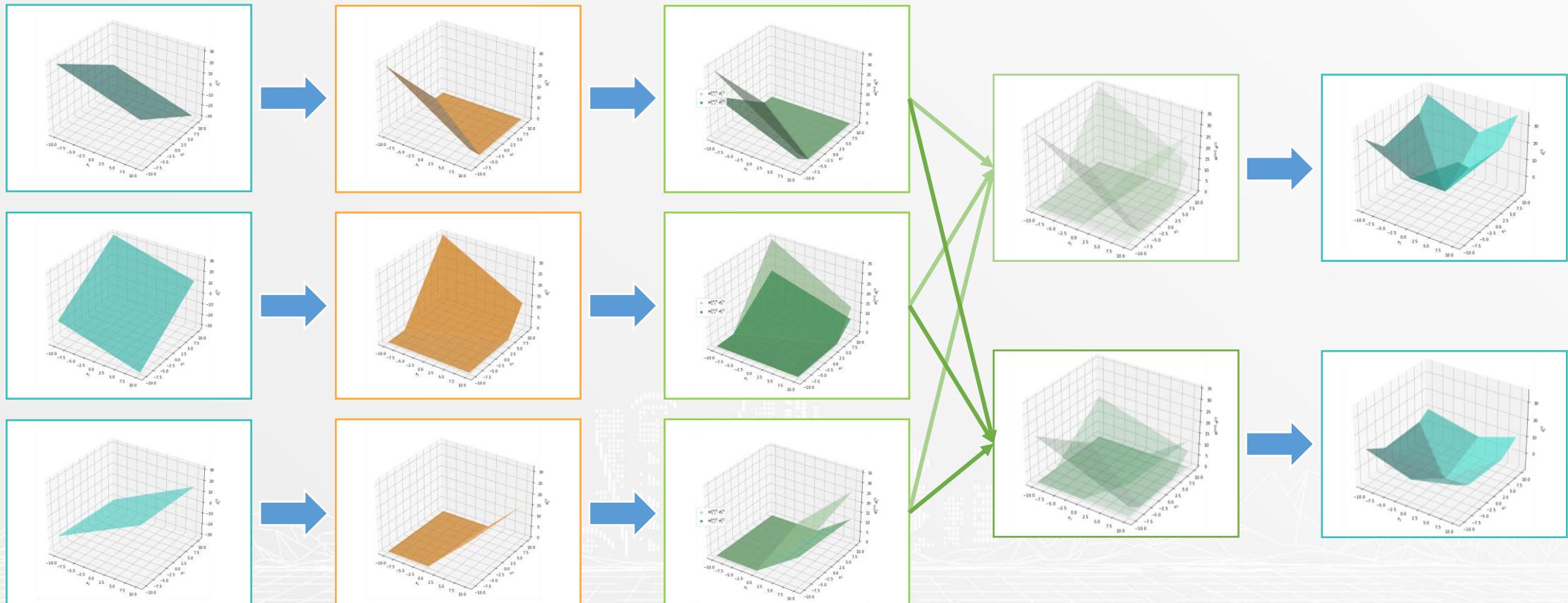
Deep Learning Interpretation

Example 2.2



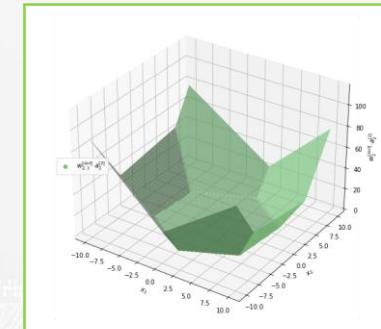
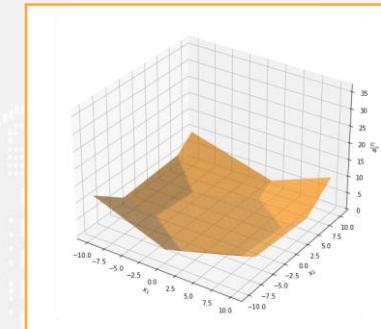
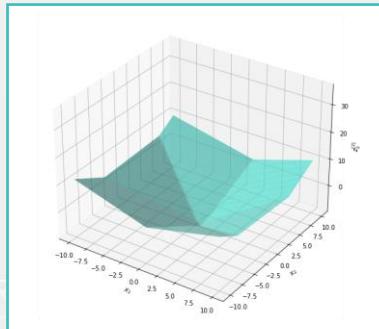
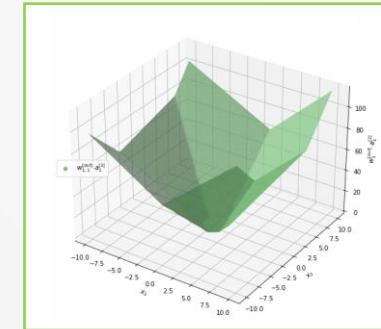
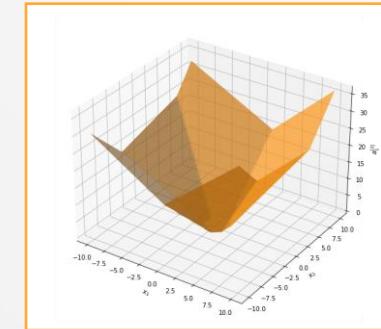
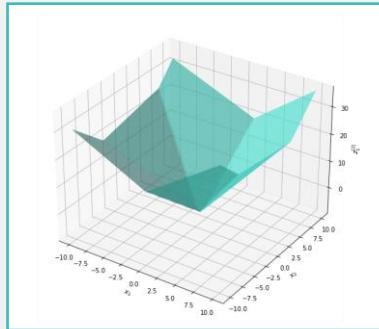
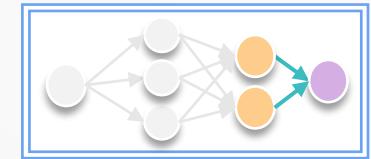
Deep Learning Interpretation

Example 2.2



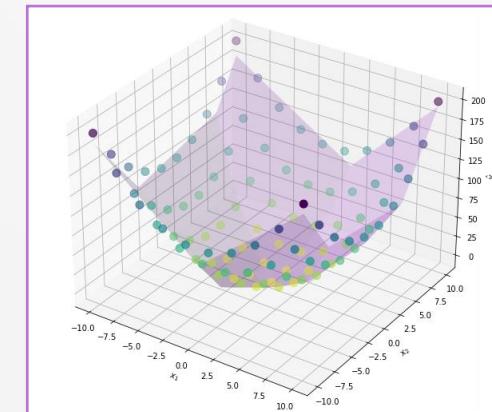
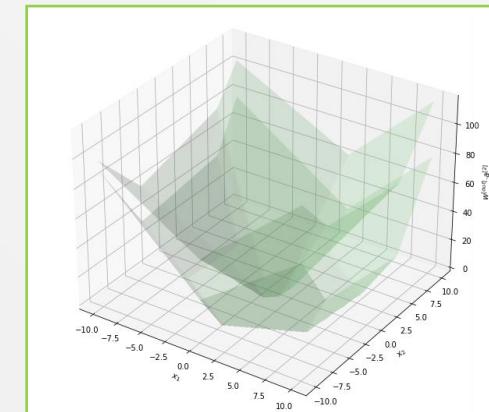
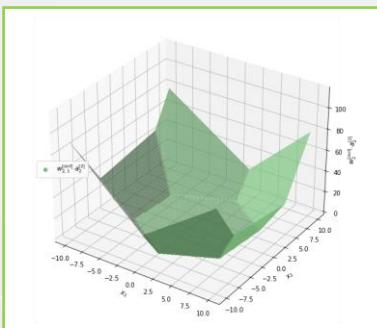
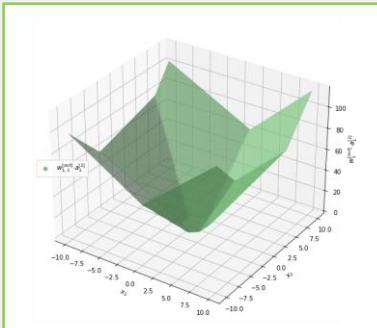
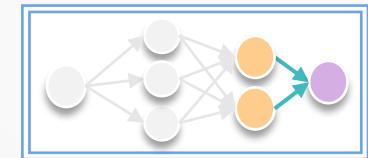
Deep Learning Interpretation

Example 2.2



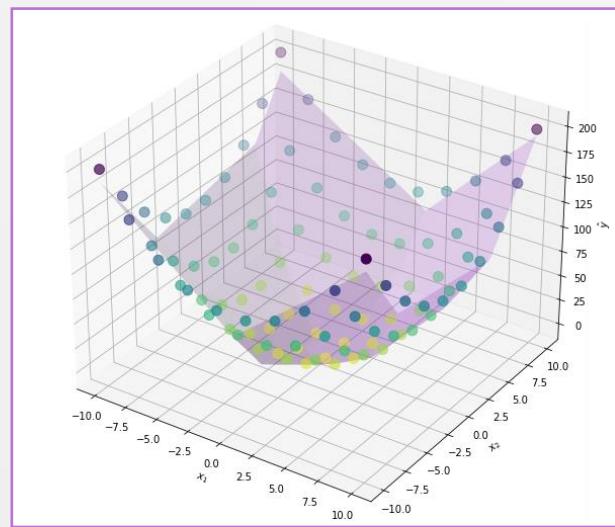
Deep Learning Interpretation

Example 2.2

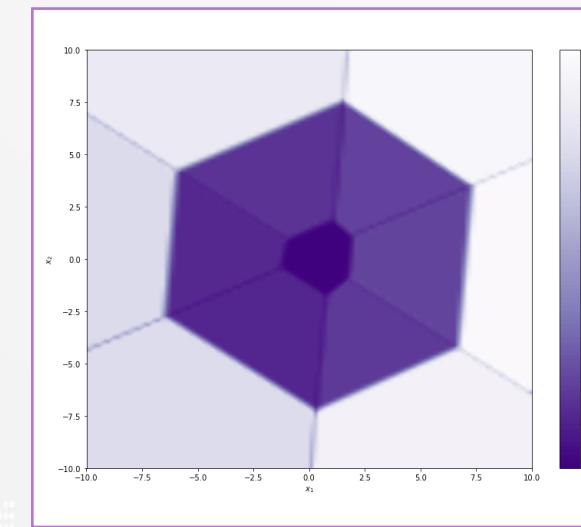


Deep Learning Interpretation

Example 2.2



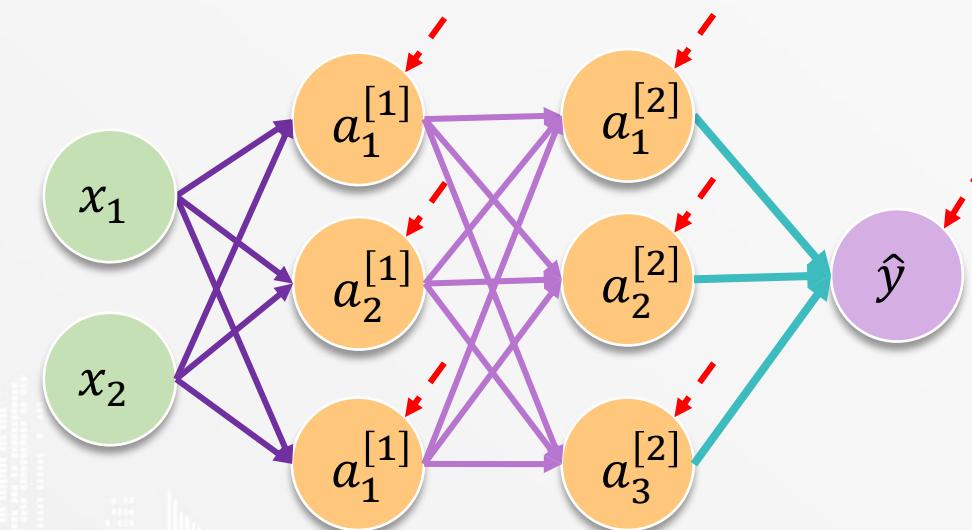
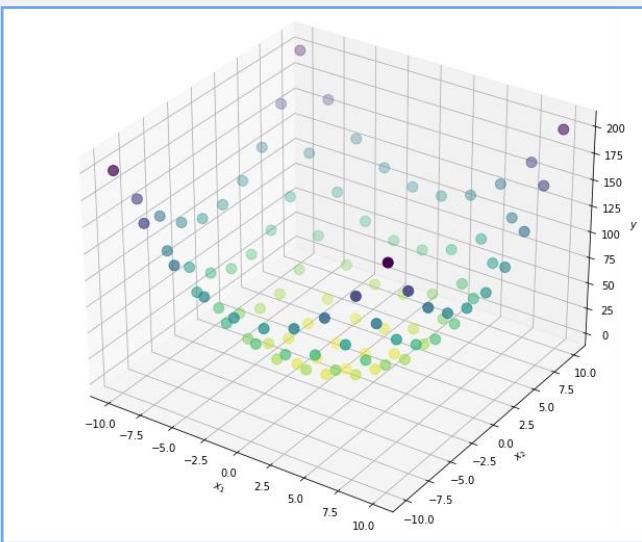
กราฟแสดงค่า predicted



Contour plot ของ predicted

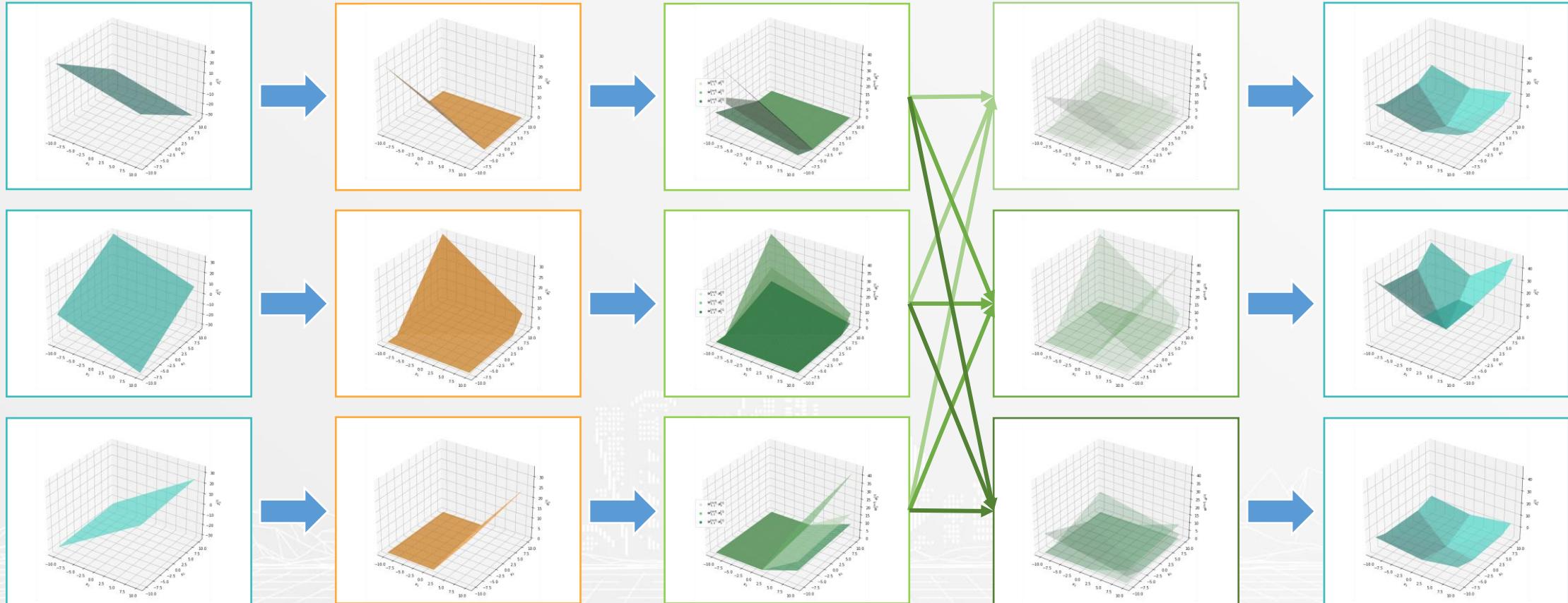
Deep Learning Interpretation

Example 2.3



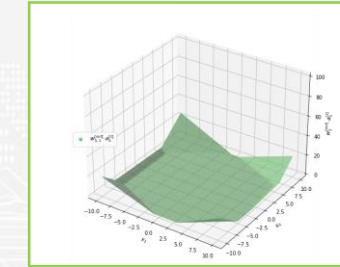
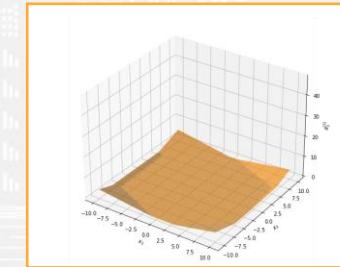
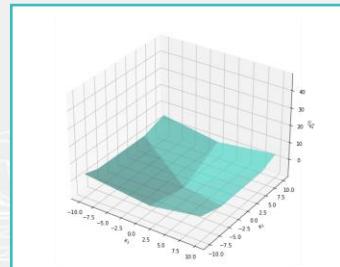
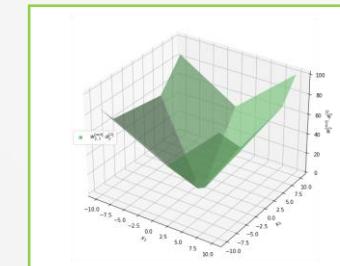
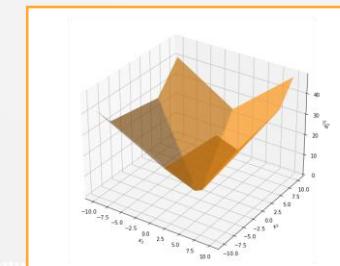
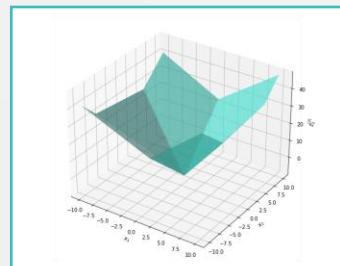
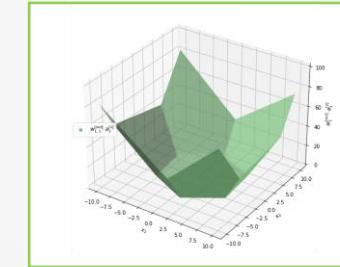
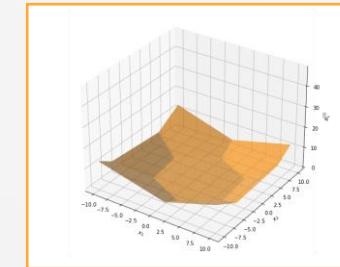
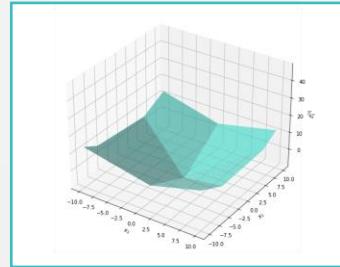
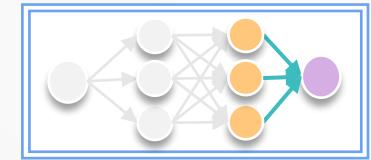
Deep Learning Interpretation

Example 2.3



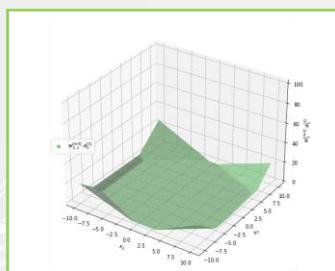
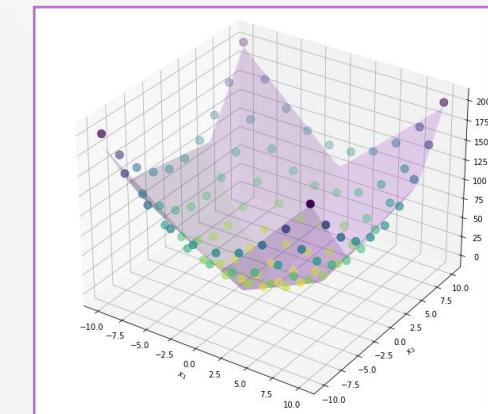
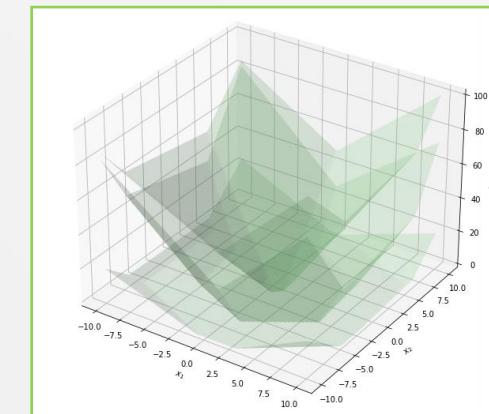
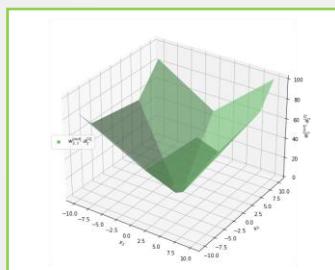
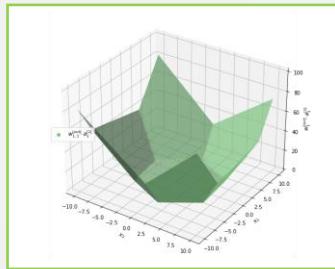
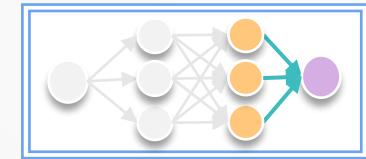
Deep Learning Interpretation

Example 2.3



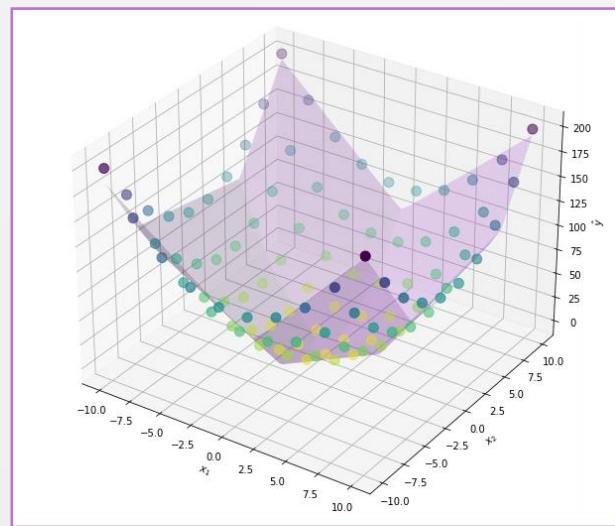
Deep Learning Interpretation

Example 2.3

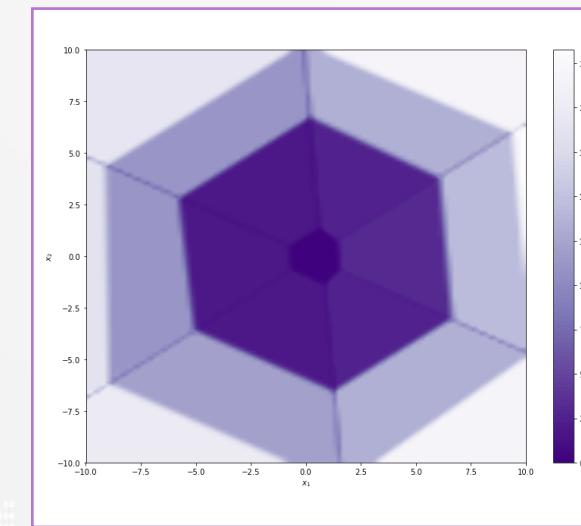


Deep Learning Interpretation

Example 2.3

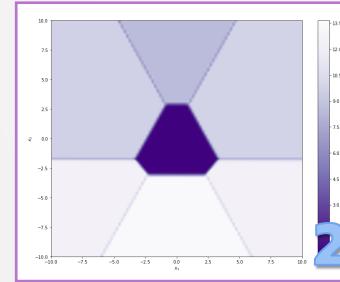
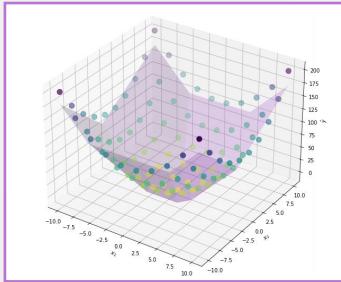


กราฟแสดงค่า predicted

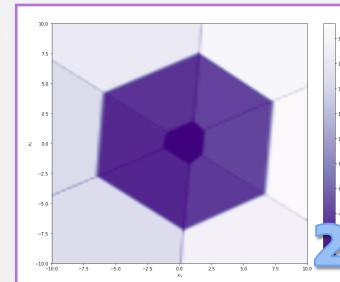
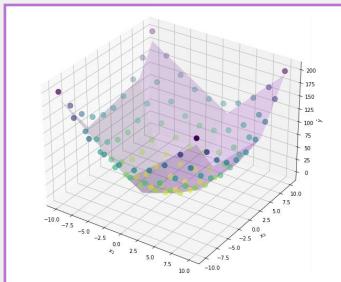


Contour plot ของ predicted

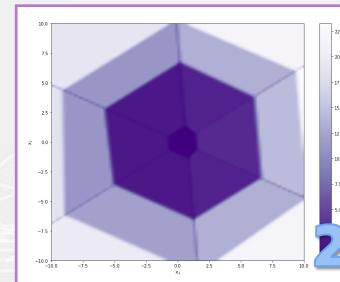
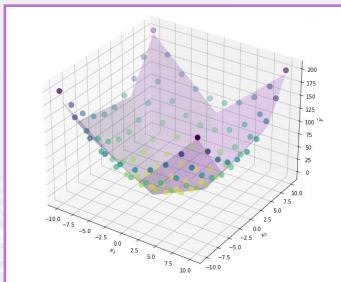
Deep Learning Interpretation



2,3,1,1



2,3,2,1

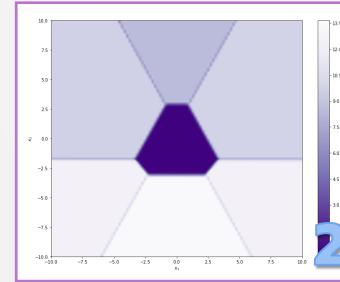
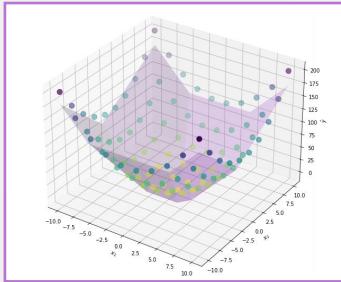


2,3,3,1

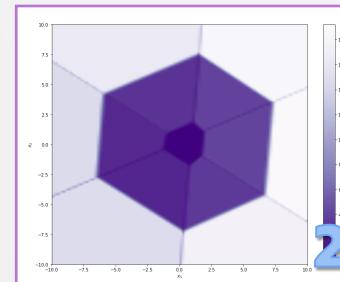
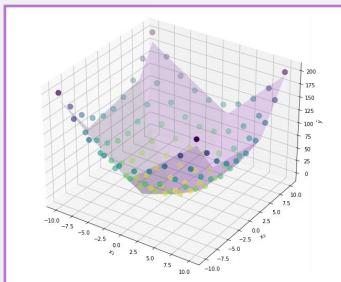
REMARK

จำนวน node ใน hidden layer
ในชั้นที่ 2 คือการสร้างความ
หลากหลายของรูปแบบของการ
หักช้ำ ด้วยการคูณด้วย weight ที่
มีค่าแตกต่างกัน

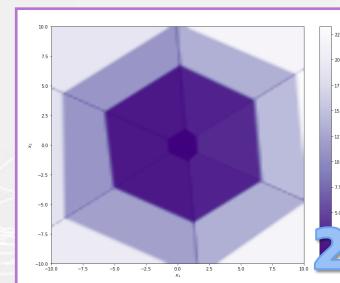
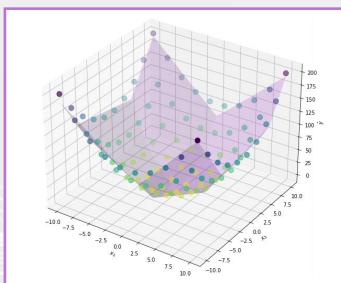
Deep Learning Interpretation



2,3,1,1



2,3,2,1

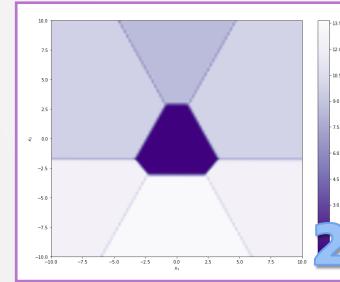
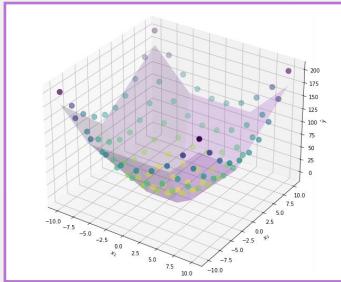


2,3,3,1

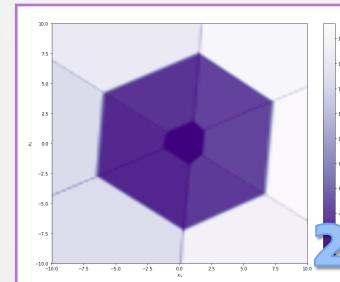
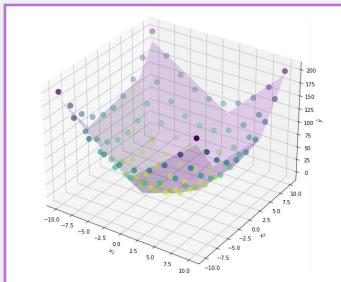
REMARK

และจะเห็นว่าเมื่อนำ function ที่ได้จากการคูณด้วย weight มาผ่าน activation function ทำให้เกิดรอยหักใหม่ๆ แตกต่างกัน

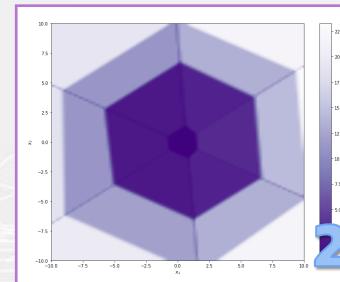
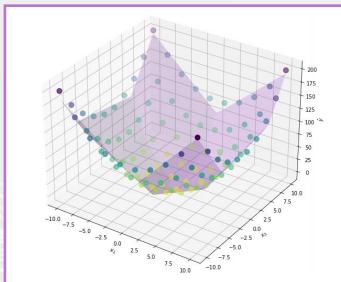
Deep Learning Interpretation



2,3,1,1



2,3,2,1



2,3,3,1

REMARK

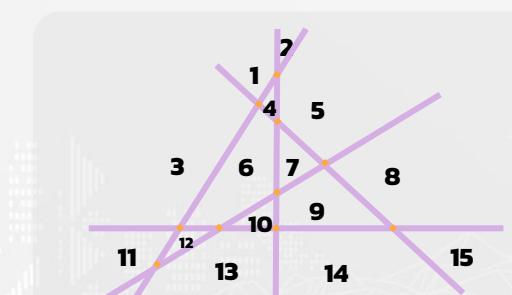
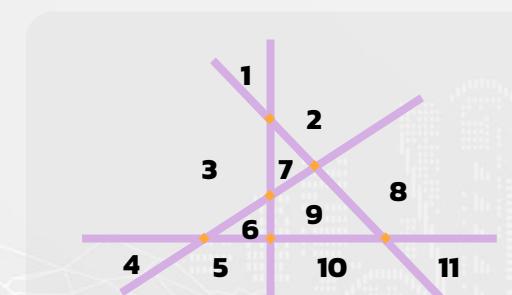
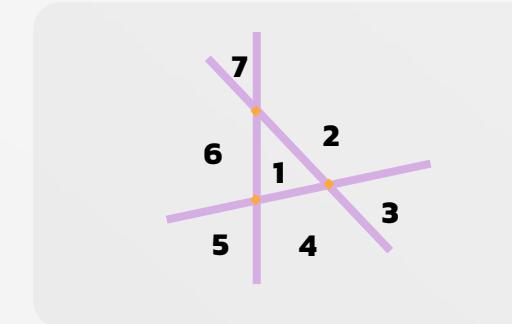
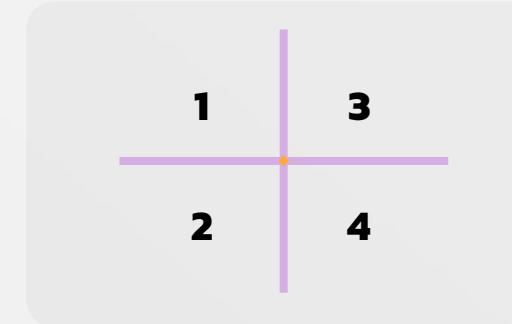
การหักใหม่ที่เกิดขึ้นนั้นเกิดจาก
การนำจุดหักเก่ามา คูณ weight
ทำหักซ้ำ ถ้ามองในมุมมอง top
view ของ 3 มิติ คือการเลื่อน
บน

Deep Learning Interpretation

การเพิ่ม **hidden layer** คือการกำ **plane** ที่เกิดจาก output ที่หากหลายที่เกิดจาก **weight** จากนั้นก่อนหน้ามาทำการประกอบ และหักช้ำผ่าน **activation function**

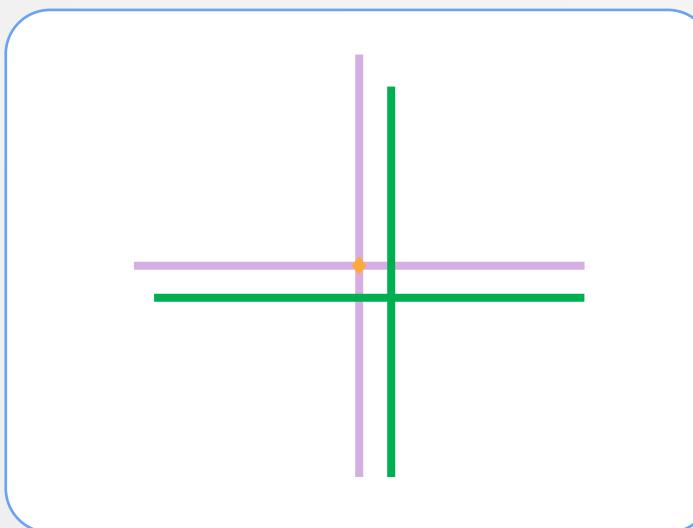
Deep Learning Interpretation

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

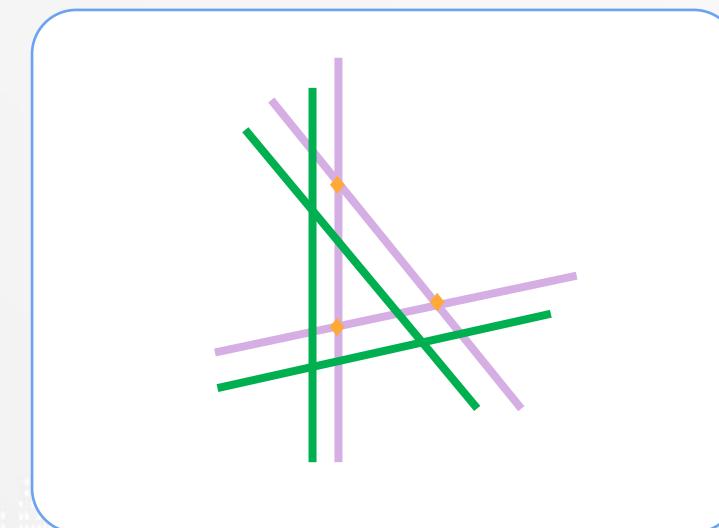


Deep Learning Interpretation

ถ้าเรานำ maximum plane มาทำการหักช้ำ โดยการเพิ่ม hidden layer ที่มี 1 node เราจะได้



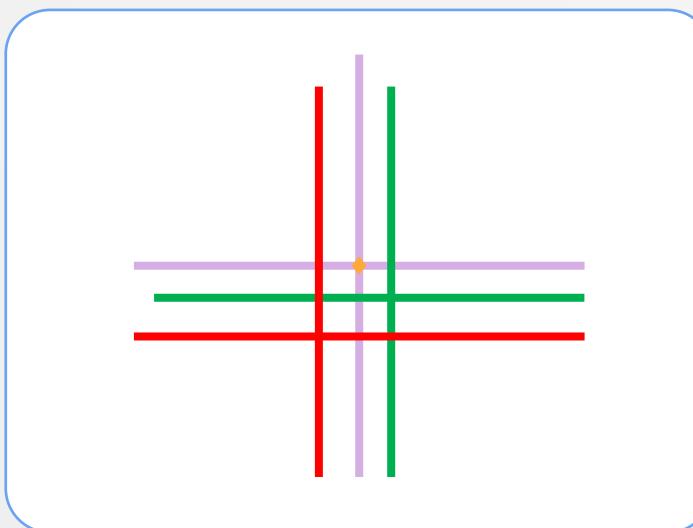
2,2,1,1



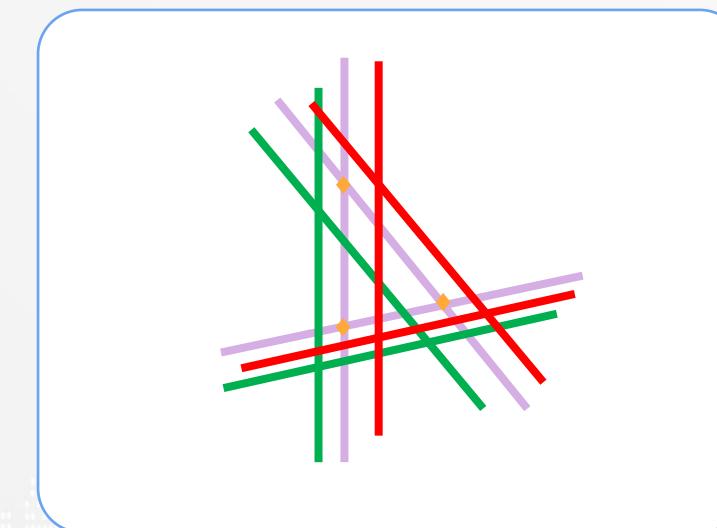
2,3,1,1

Deep Learning Interpretation

และยิ่งถ้าเราเพิ่มจำนวน node ใน hidden layer ชั้นที่ 2 เป็น 2 node เราจะได้



2,2,2,1



2,3,2,1

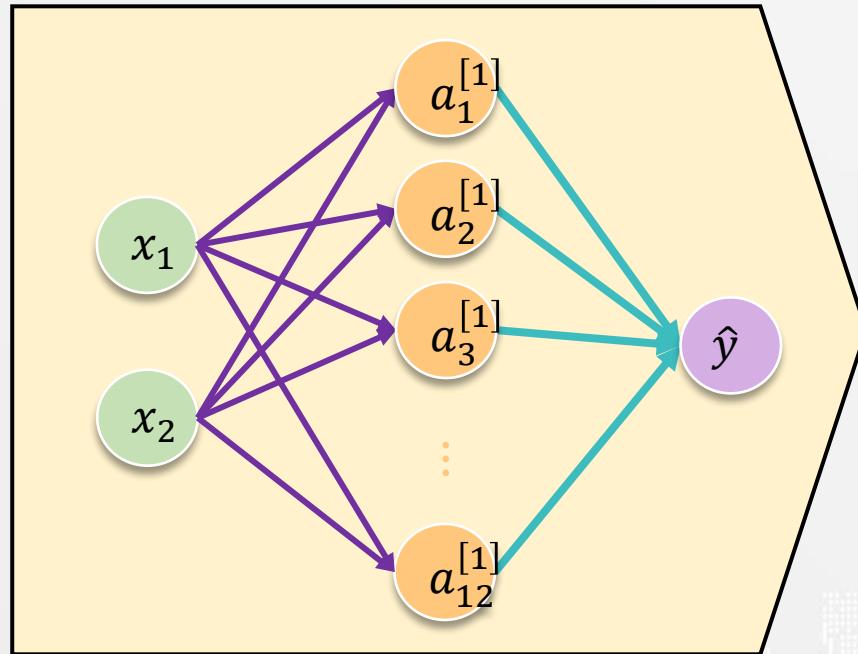
Deep Learning Interpretation

จากการเพิ่มขึ้นของ node ใน hidden layer
ทำให้ plane ใน output node มีการเพิ่มขึ้นอย่างทวีคูณ

Deep Learning Interpretation

รอยหักใน **hidden layer** ที่เกิดขึ้นมาบัน ถูกสร้างขึ้นจากการ
นำรอยหักเดิมมาคูณด้วย weight
ทำให้ถูกจำคัดเงื่อนไขของรอยหักที่เกิดขึ้นใหม่ นั้นคือการมี
ลักษณะ **ขานกับรอยหักเดิม** (ถ้ามองจาก top view)

Deep Learning Interpretation

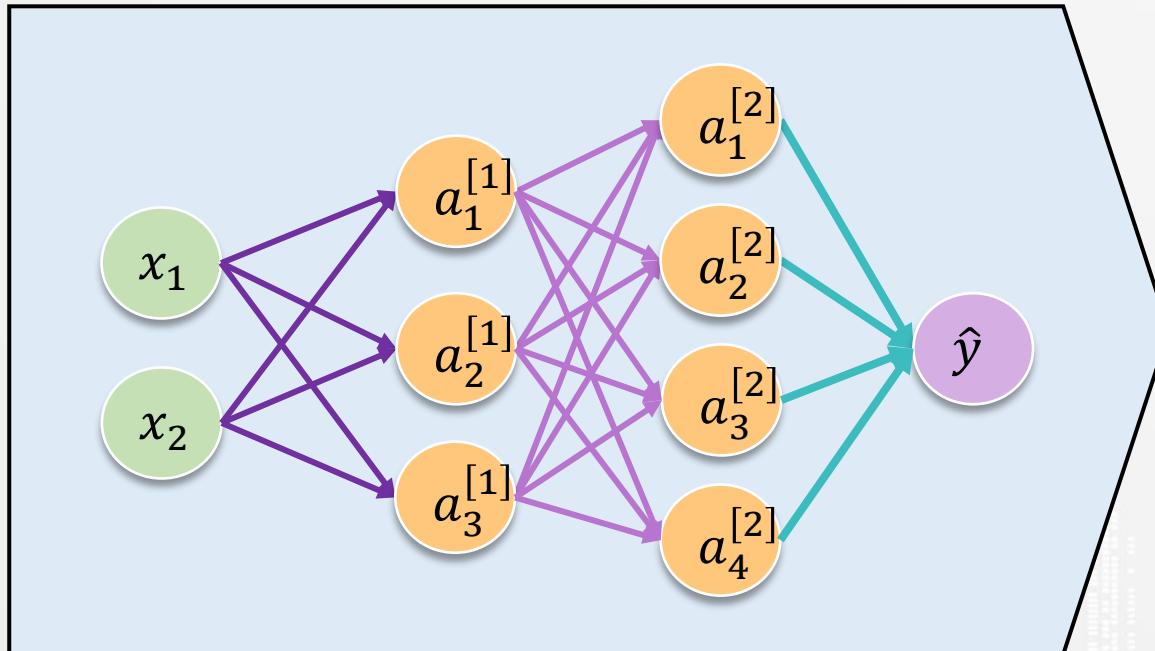


$$\text{Cost} \approx (2 \times 12) + (12 \times 1)$$

$$= 24 + 12$$

$$= 36$$

Deep Learning Interpretation



$$\text{Cost} \approx (2 \times 3) + (3 \times 4) + (4 \times 1)$$

$$= 6 + 12 + 4$$

$$= 22$$

Deep Learning Interpretation

การเพิ่ม **hidden layer** ทำให้เกิดการเพิ่มขึ้นของ **plane** อย่าง
ทวีคูณ โดยการเพิ่ม **computational cost** เพียงเล็กน้อย แต่ก็
แลกมาด้วย ข้อจำกัดขององค์ความรู้ที่เกิดขึ้นใหม่

Deep Learning Interpretation

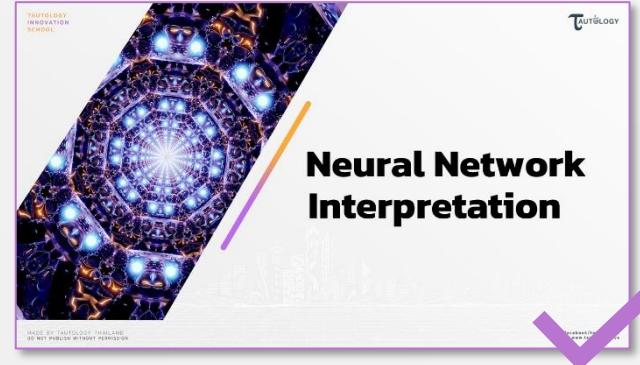
	ข้อดี	ข้อจำกัด
การเพิ่ม node	plane ที่เพิ่ม มีอิสระในการเลือกรอยหักที่เหมาะสม	จำนวน plane ที่เพิ่มน้อย เมื่อเทียบกับ cost
การเพิ่ม hidden layer	จำนวน plane เพิ่มขึ้นอย่างทวีคูณ	plane ที่เกิดขึ้นเกิดจากรอยหักเดิม

Deep Learning Interpretation

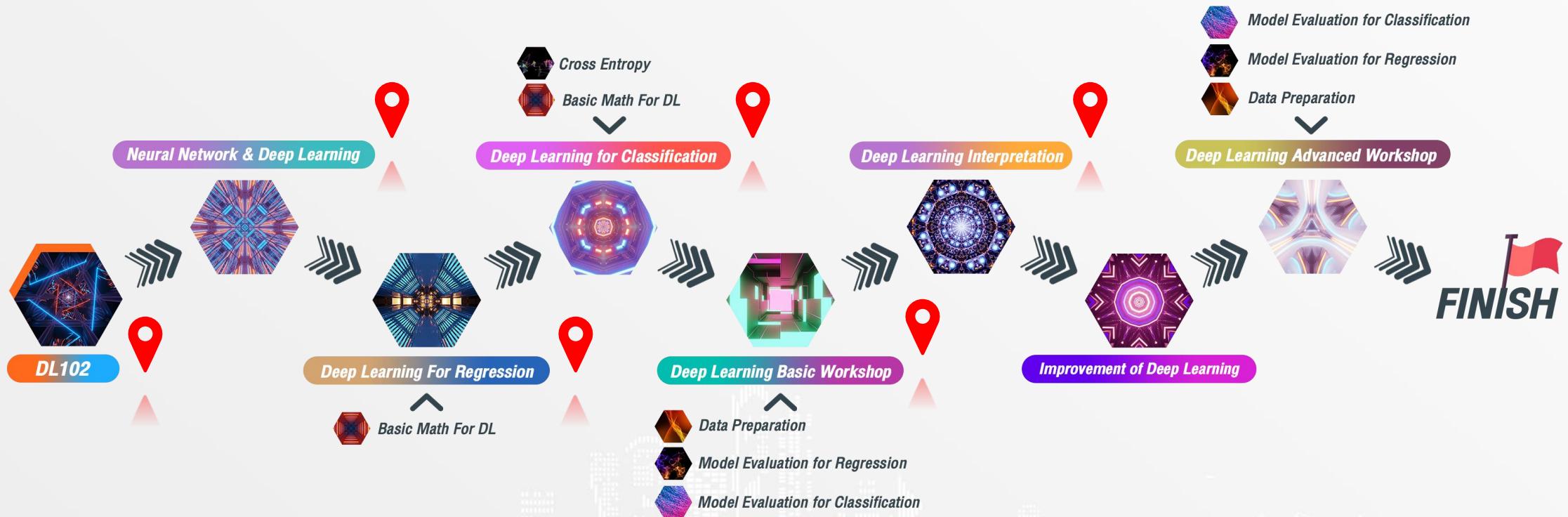
Conclusion

- การเพิ่มจำนวน node ทำให้เกิดการหักที่จุดที่แตกต่าง
- การเพิ่มจำนวน hidden layer ทำให้เกิดการนำจุดหักเก่ามา คูณ weight ทำหักซ้ำ ถ้ามองในมุมมอง top view ของ 3 มิติ คือการเลื่อนขนา
- การเพิ่มจำนวน hidden layer ทำให้รอยหักเพิ่มขึ้นเป็นกิวคูณ ในขณะที่การเพิ่มจำนวน node ทำให้รอยหักเพิ่มขึ้นทีละ 1
- รอยหักที่เกิดขึ้นจากการเพิ่มจำนวน node จะเป็นอิสระ ในขณะที่ รอยหักที่เกิดขึ้นจากการเพิ่มจำนวน hidden layer จะเกิดจากการเลื่อนขนาของหักในชั้นก่อนหน้า
- การเพิ่ม hidden layer ทำให้ได้รอยหักที่เพิ่มขึ้นแบบกิวคูณได้โดยที่ cost ใช้เพิ่มไม่เยอะ

Deep Learning Interpretation



DL103 : Deep Learning



TAUTOLOGY
INNOVATION
SCHOOL



IMPROVEMENT OF DEEP LEARNING

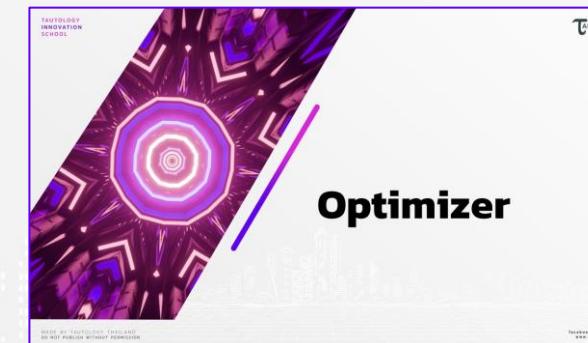
BY TAUTOLOGY

IMPROVEMENT OF DEEP LEARNING

MADE BY TAUTOLOGY THAILAND
DO NOT PUBLISH WITHOUT PERMISSION

facebook/tautologyai
www.tautology.live

Improvement of Deep Learning



Speed-Up with GPU

Speed-Up with GPU

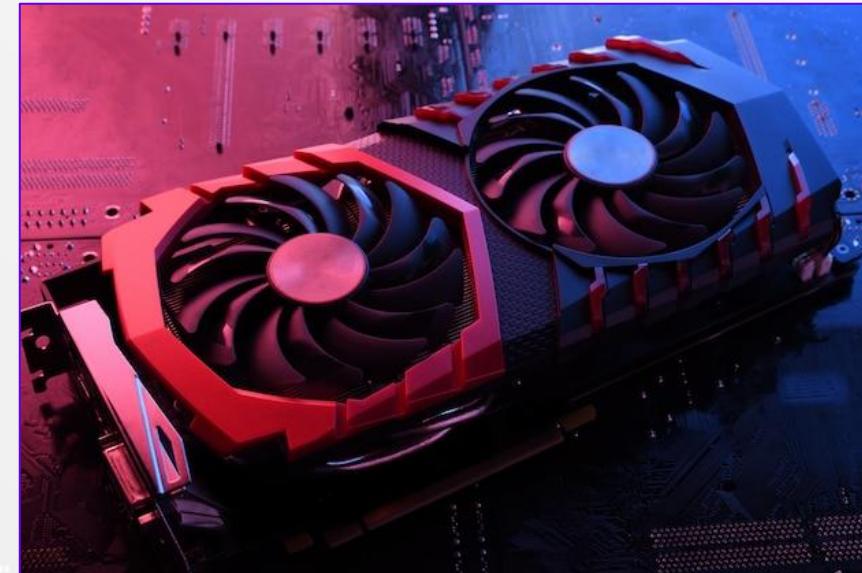
What is GPU?

How GPU Accelerate
DL?

Welcome to Colab

Train Model with
GPU on Colab

What is GPU?



Speed-Up with GPU

What is GPU?

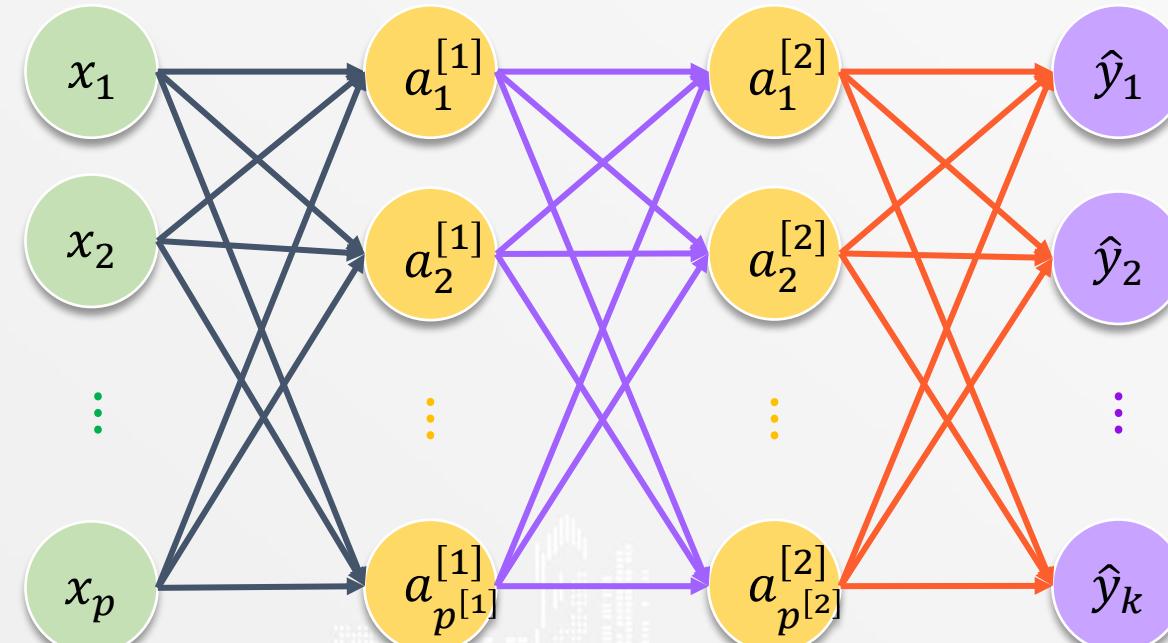


How GPU Accelerate
DL?

Welcome to Colab

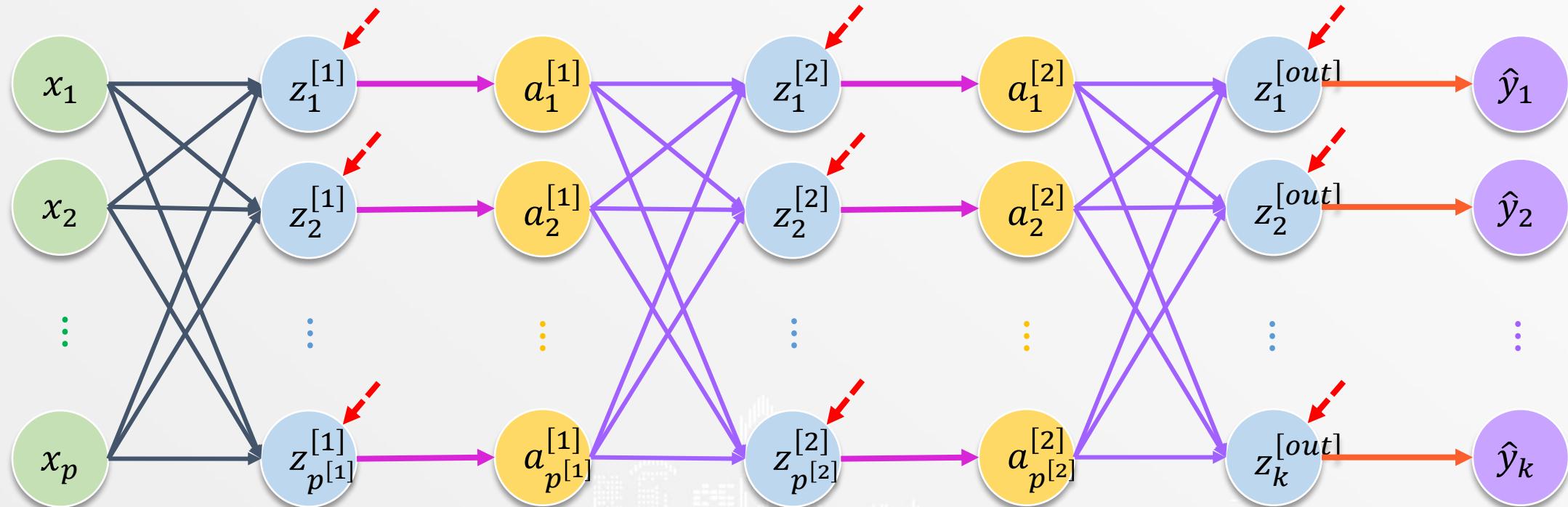
Train Model with
GPU on Colab

How GPU Accelerate DL?



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

How GPU Accelerate DL?



architecture នៃ deep learning សំអរប័ណ្ណមាតា multi-class
classification ទី២ hidden layer 2 ប៉ុណ្ណោះ

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?

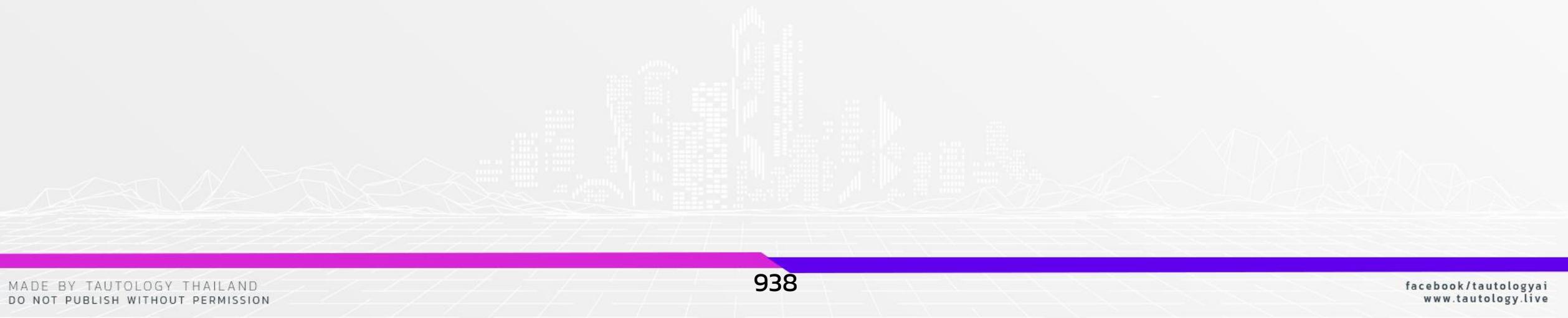
$$X \rightarrow Z^{[1]} \rightarrow A^{[1]} \rightarrow Z^{[2]} \rightarrow A^{[2]} \rightarrow Z^{[out]} \rightarrow A^{[out]}$$

How GPU Accelerate DL?

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

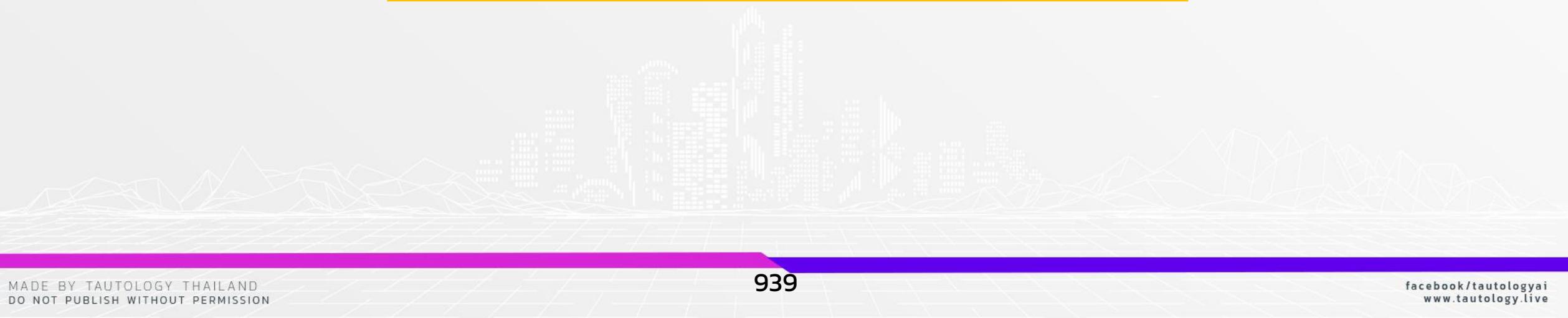
$$Z^{[2]} = A^{[1]}W^{[2]} + b^{[2]}$$

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



How GPU Accelerate DL?

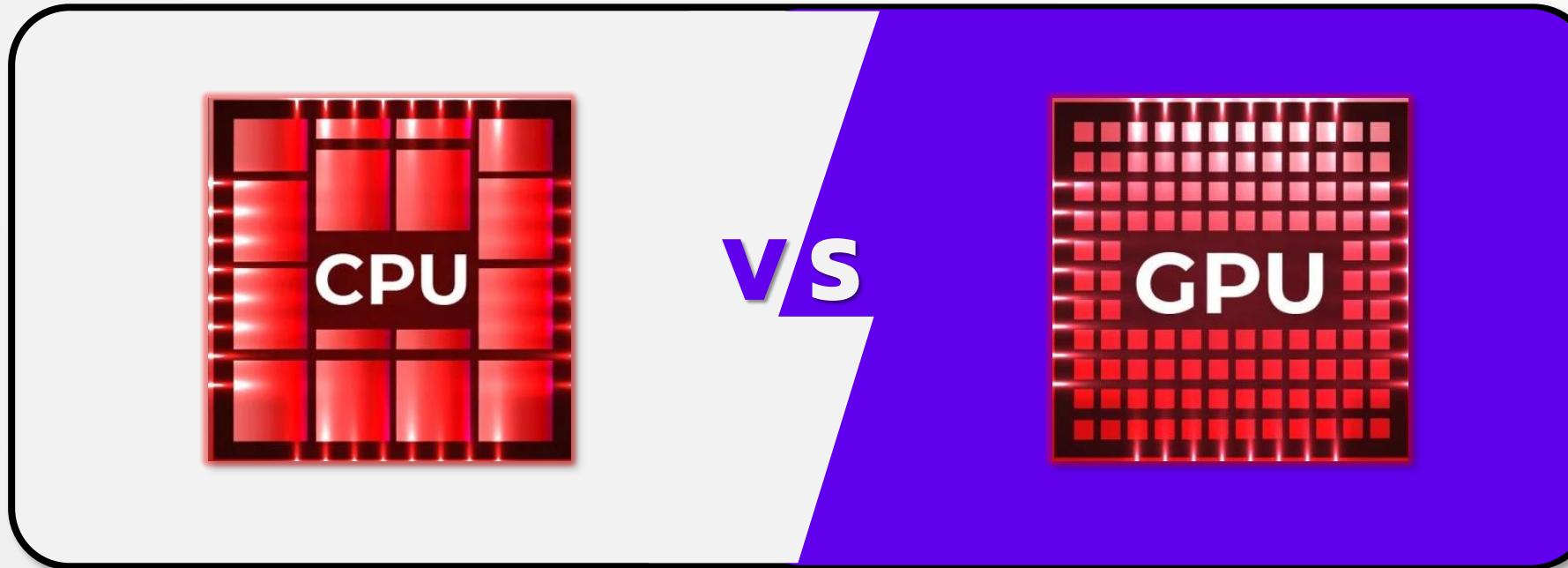
การสร้าง deep learning คือ
มหกรรมแห่งการคูณ matrix



How GPU Accelerate DL?

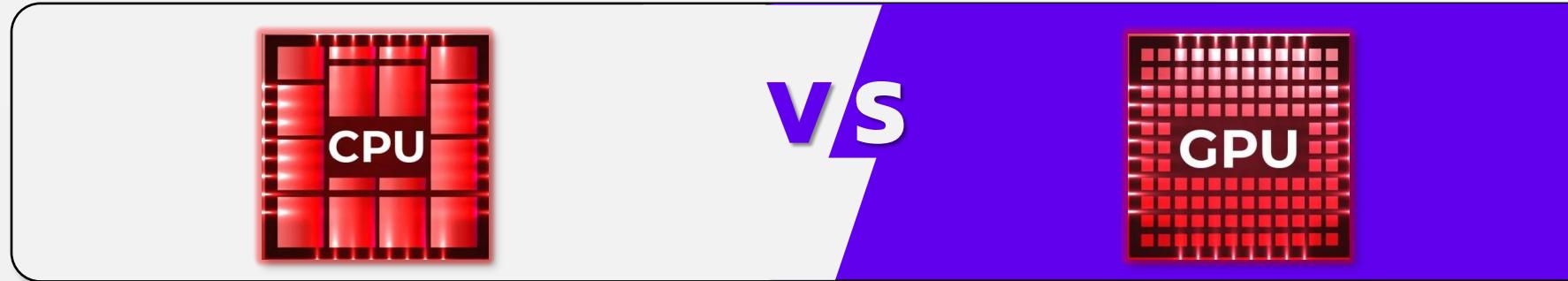
GPU คุณ matrix ได้เร็วกว่า **CPU**

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?

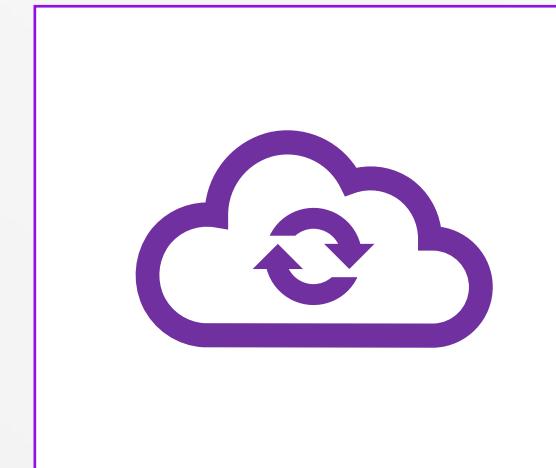


How GPU Accelerate DL?

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

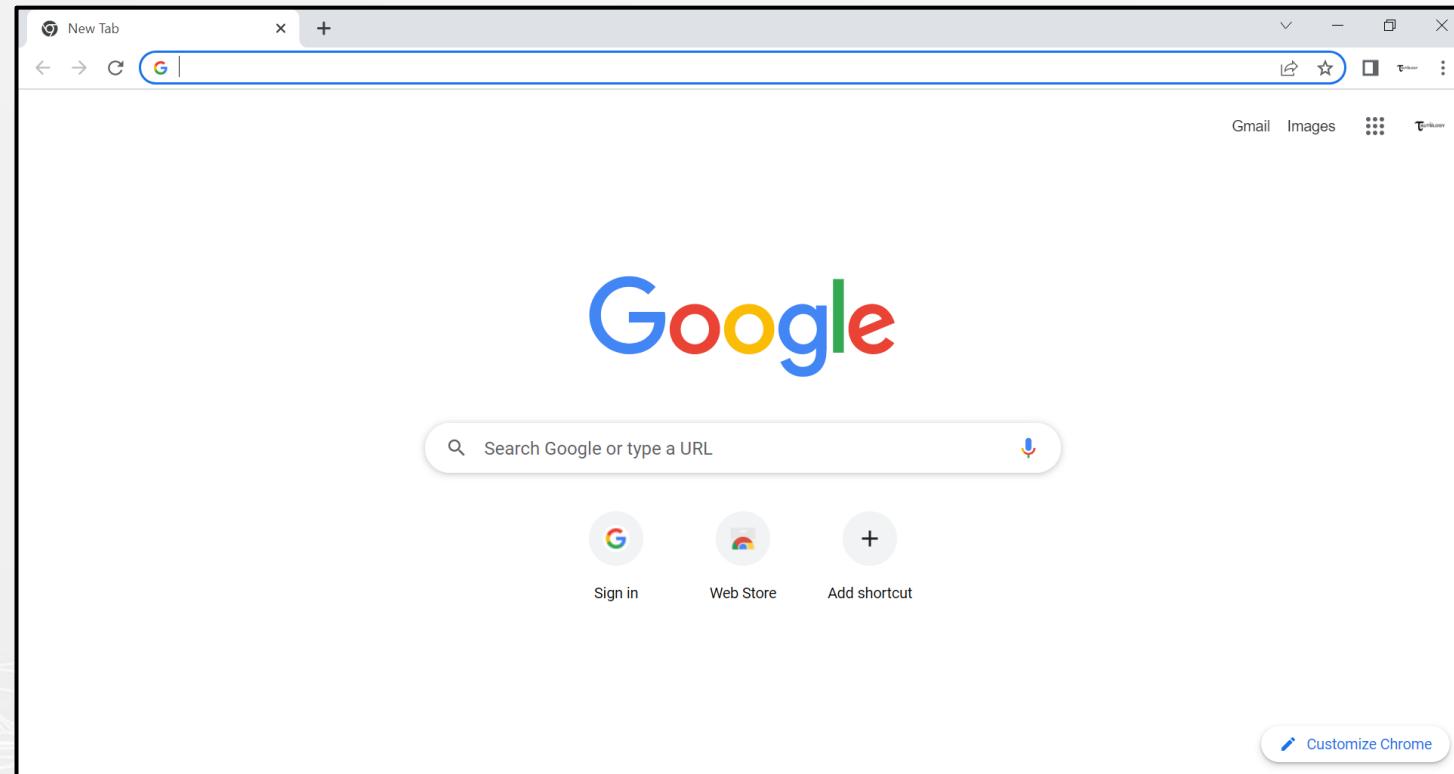


Welcome to Colab

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

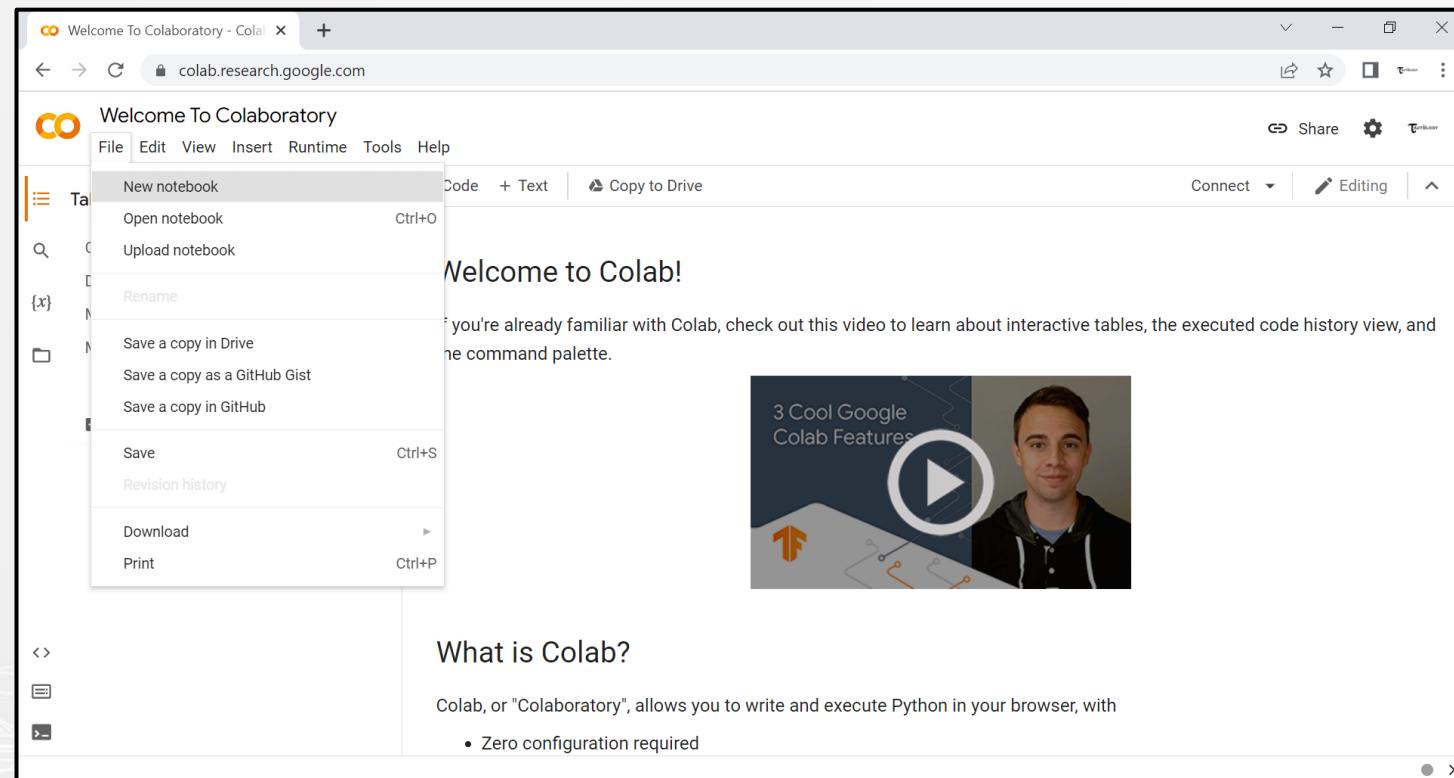
Getting Started with Colab

Step 1: Open browser



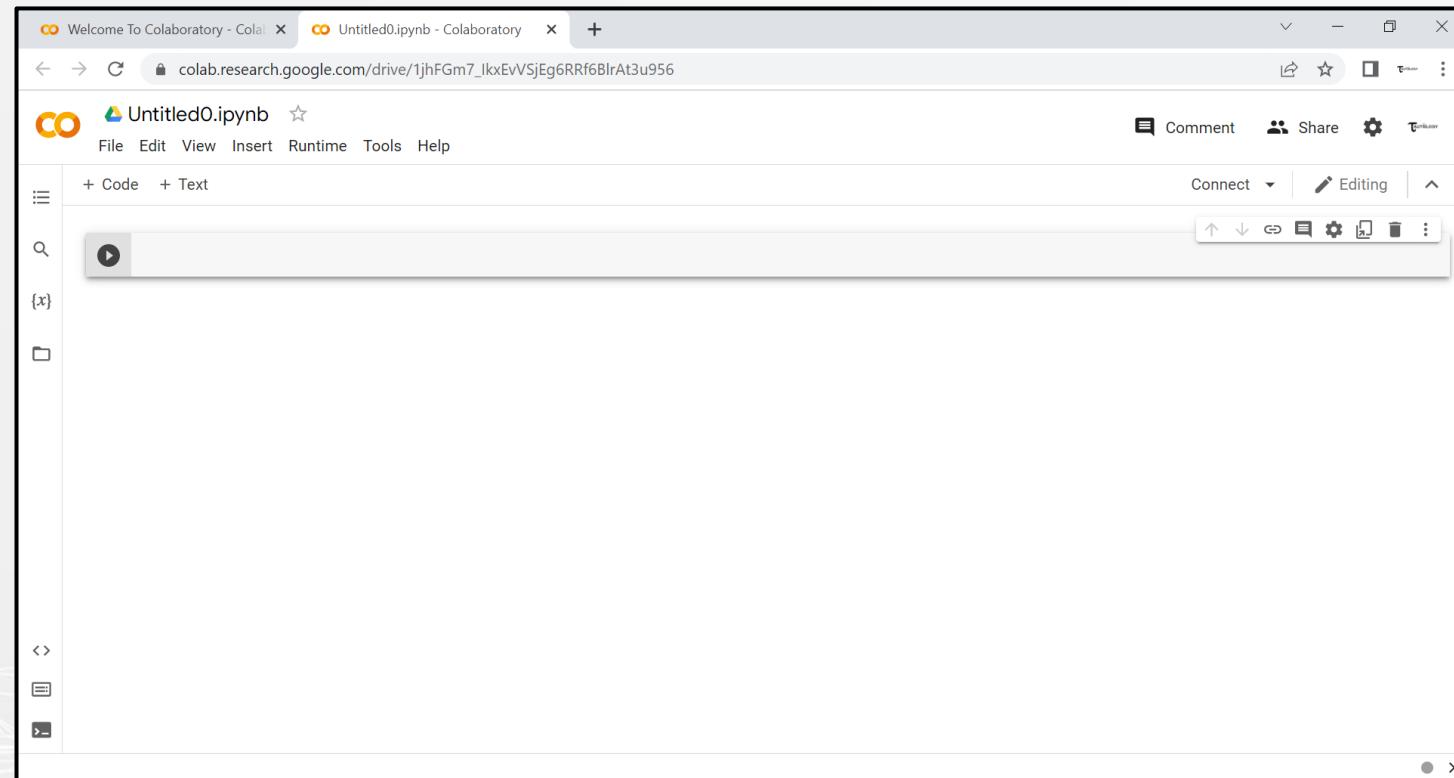
Getting Started with Colab

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



Getting Started with Colab

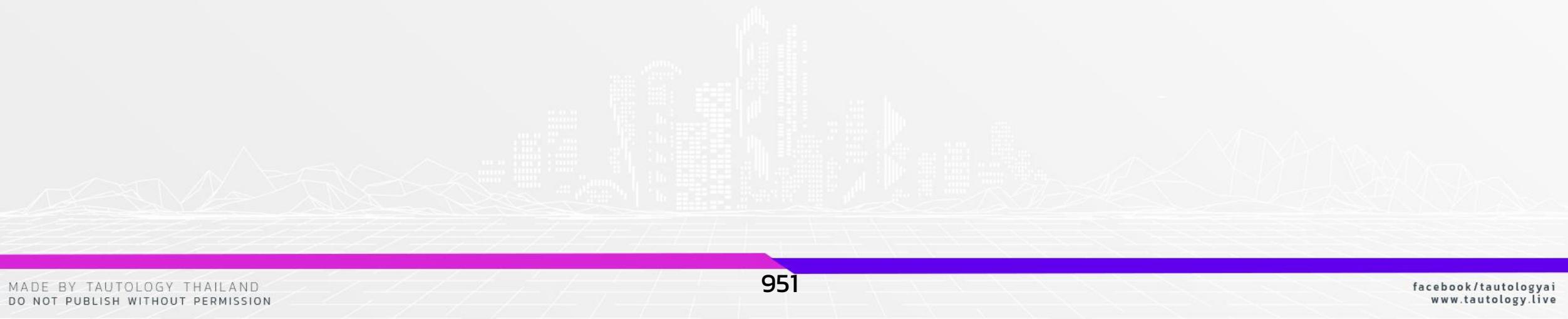
Step 3 : New notebook



Welcome to Colab

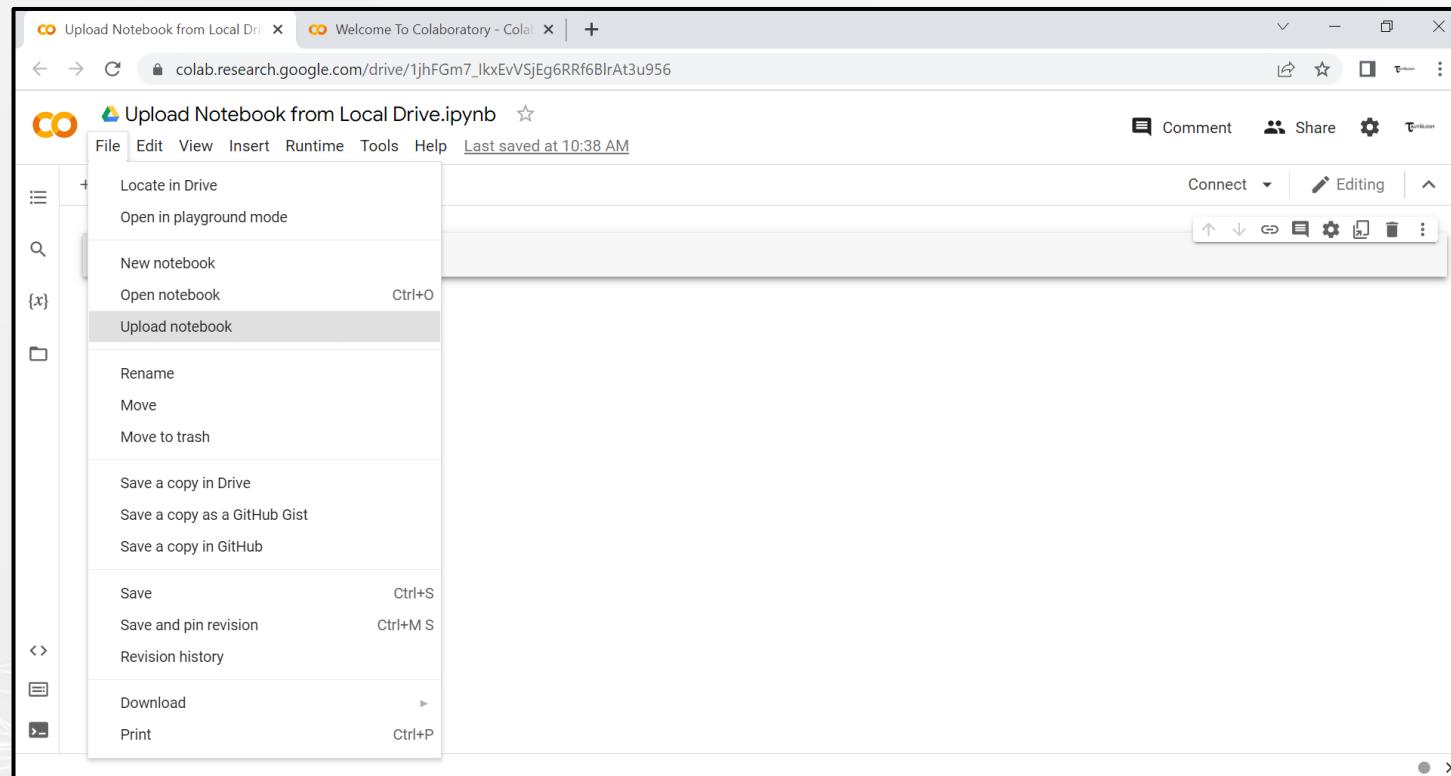
Getting Started with Colab

- Upload Notebook from Local Drive
- Import Data to Colab



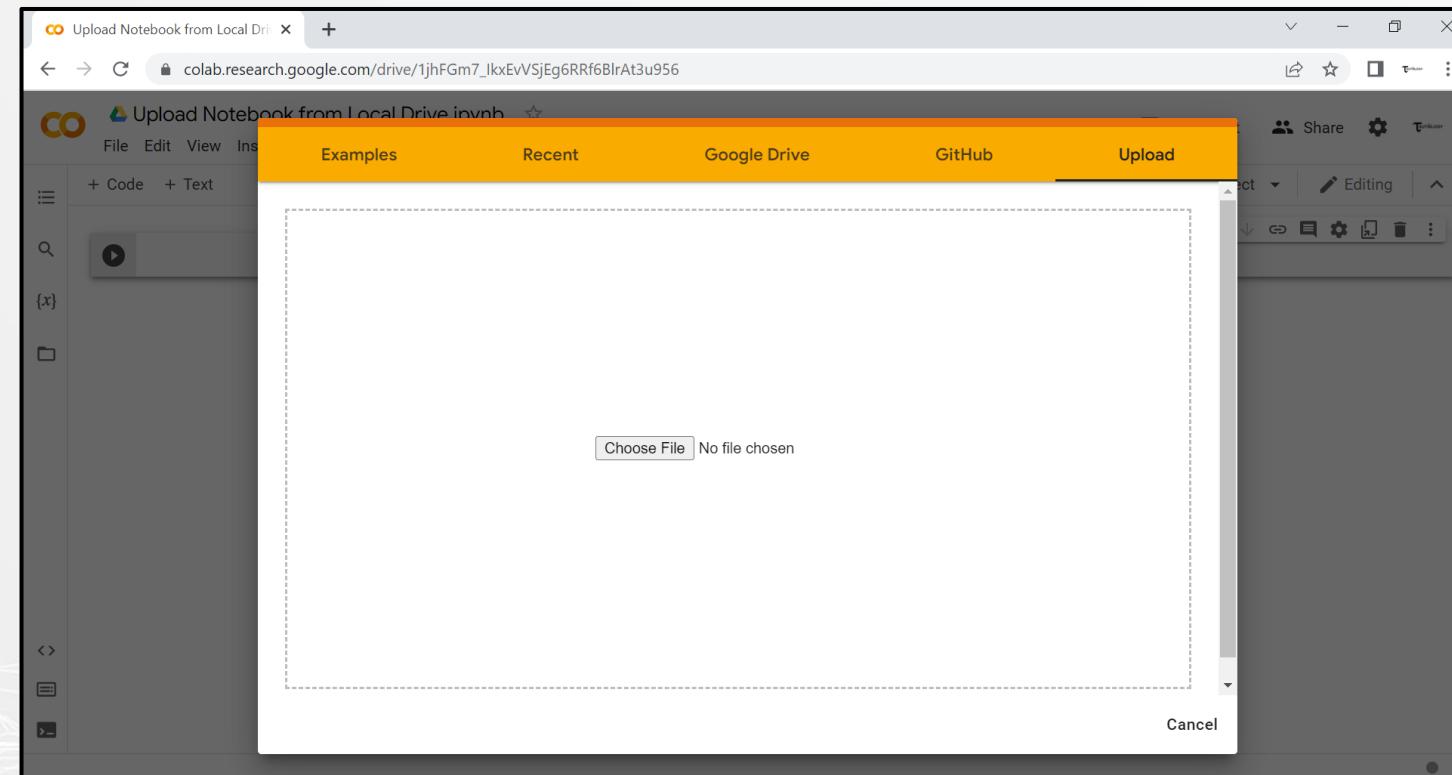
Upload Notebook from Local Drive

Step 1: Upload



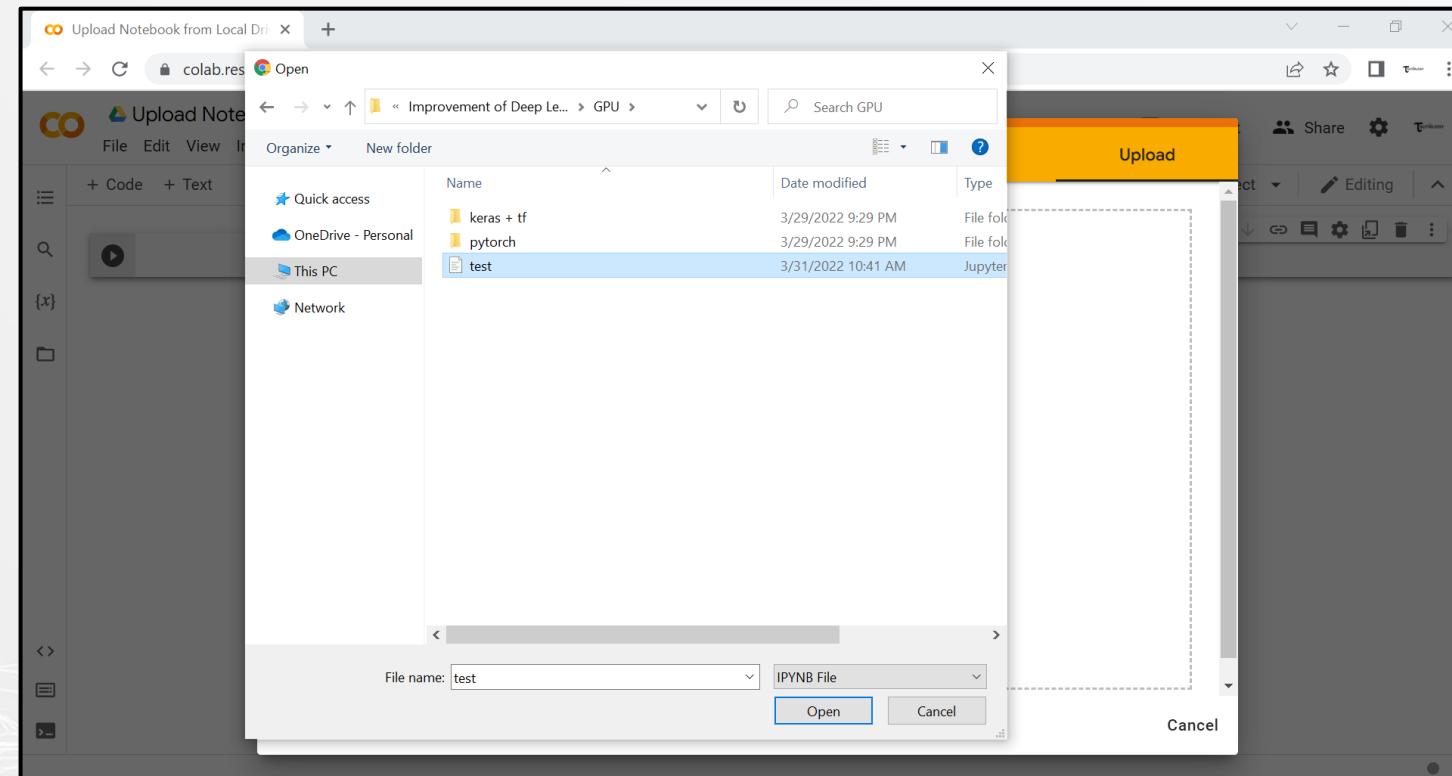
Upload Notebook from Local Drive

Step 1: Upload



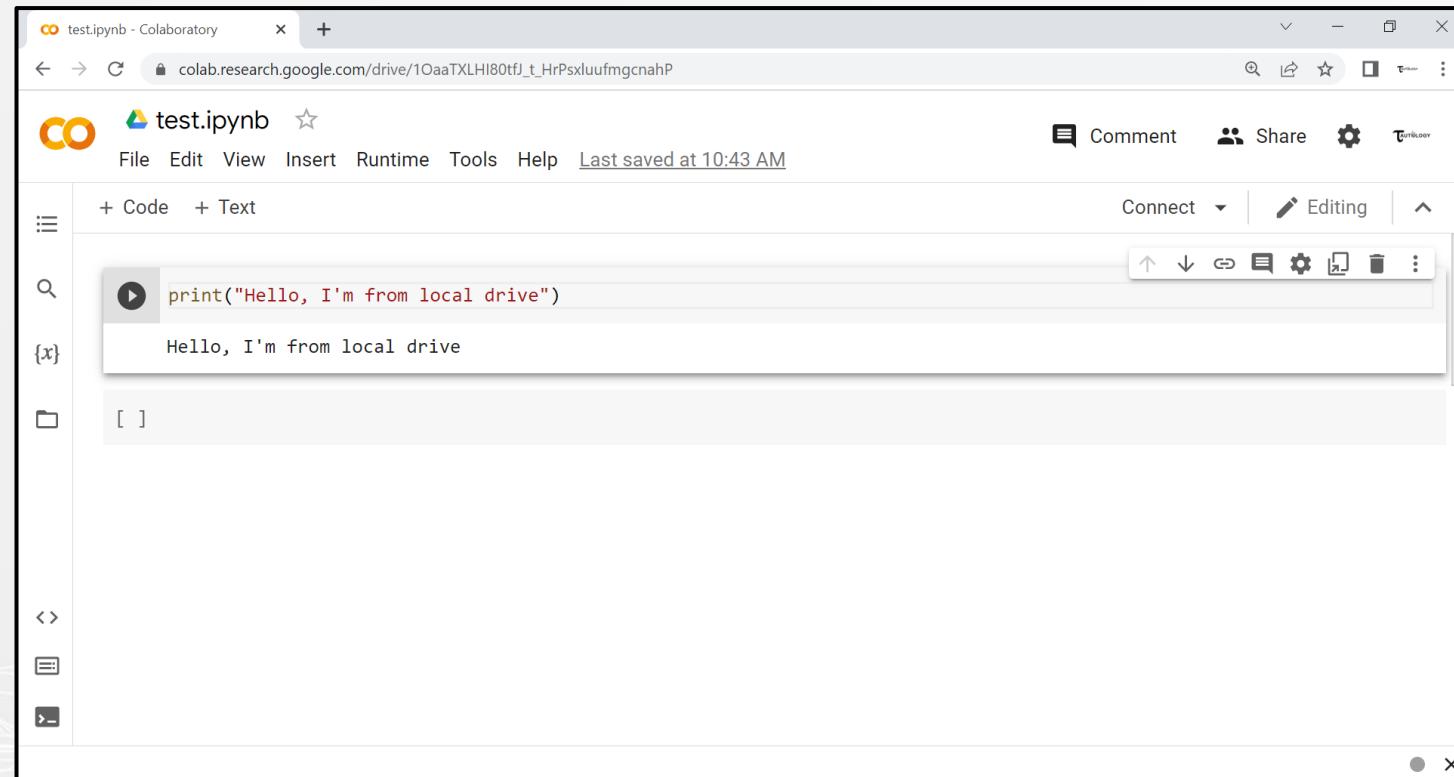
Upload Notebook from Local Drive

Step 2 : Choose File



Upload Notebook from Local Drive

Step 3 : Run



A screenshot of a Google Colab notebook titled "test.ipynb". The notebook interface includes a toolbar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The status bar shows "Last saved at 10:43 AM". The main workspace contains a code cell with the following content:

```
print("Hello, I'm from local drive")
```

The output of this cell is displayed in a box below it:

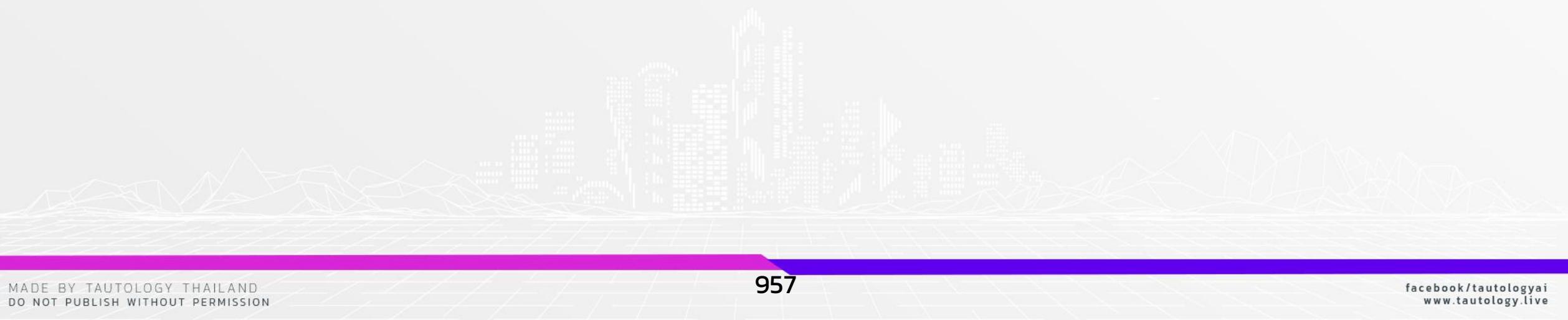
Hello, I'm from local drive

Welcome to Colab

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

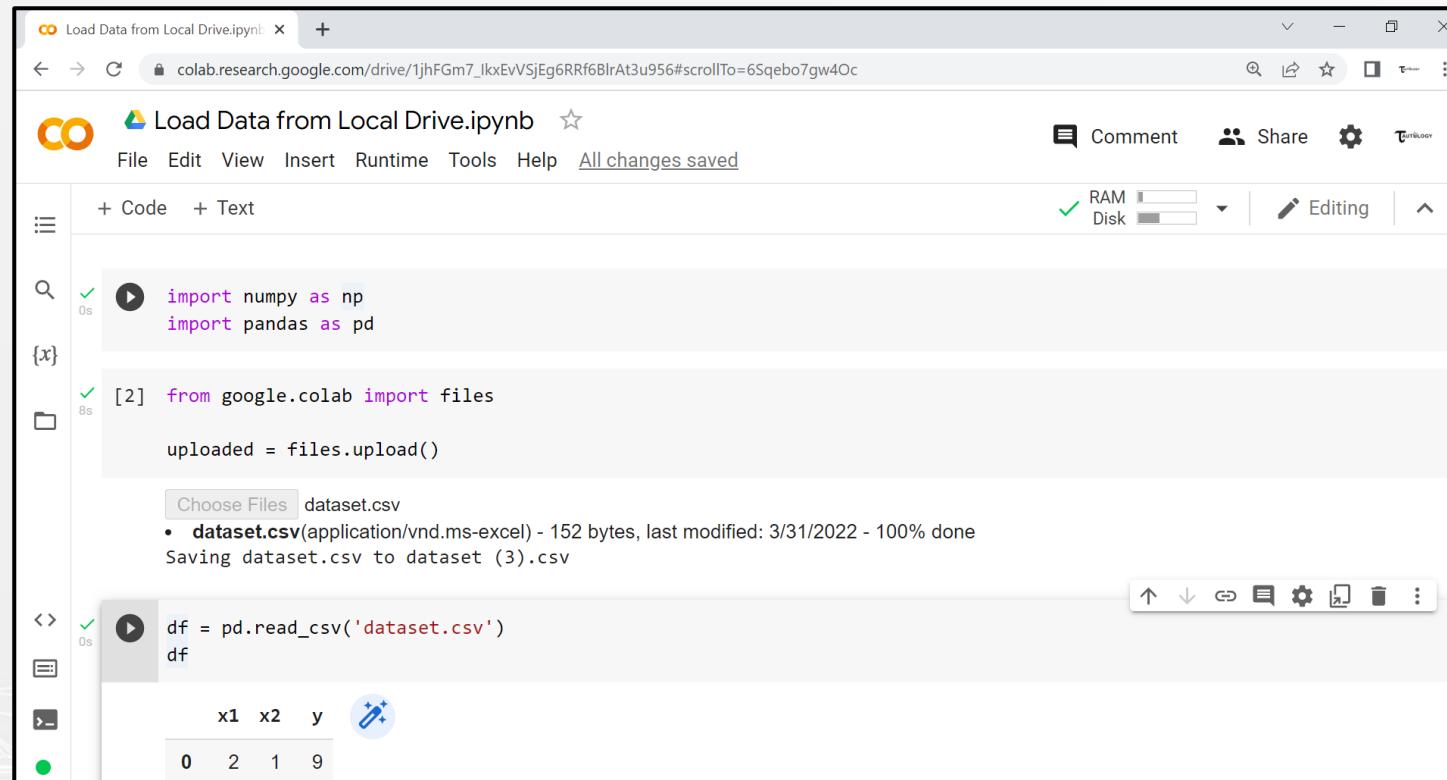
Import Data to Colab

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



Import Data to Colab

Load Data from Local Drive



```
import numpy as np
import pandas as pd

from google.colab import files

uploaded = files.upload()

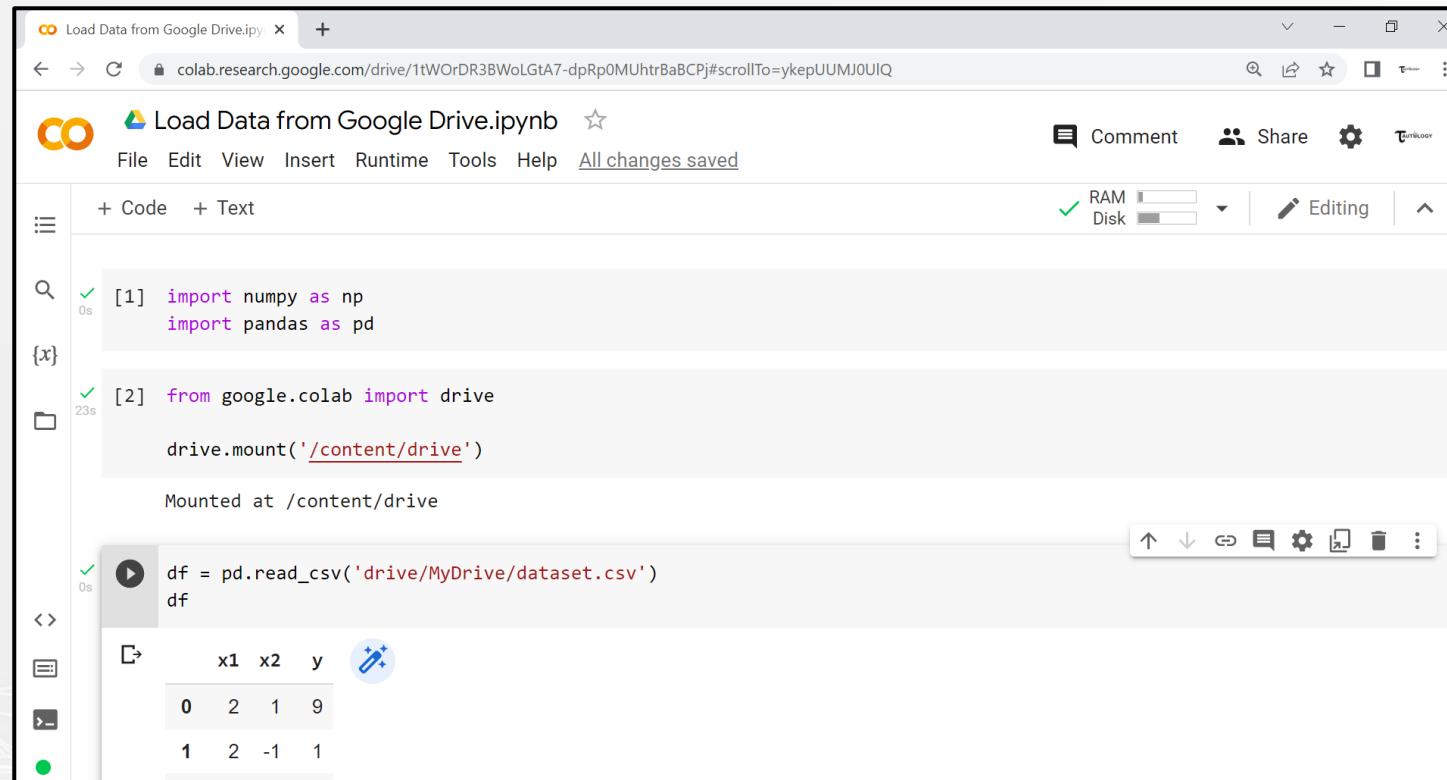
Choose Files dataset.csv
dataset.csv(application/vnd.ms-excel) - 152 bytes, last modified: 3/31/2022 - 100% done
Saving dataset.csv to dataset (3).csv

df = pd.read_csv('dataset.csv')
df
```

	x1	x2	y
0	2	1	9

Import Data to Colab

Load Data from Google Drive



```
File Edit View Insert Runtime Tools Help All changes saved
```

```
[1] import numpy as np
import pandas as pd

[2] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

df = pd.read_csv('drive/MyDrive/dataset.csv')
df
```

	x1	x2	y
0	2	1	9
1	2	-1	1

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



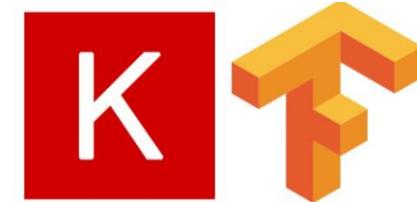
**Train Model with
GPU on Colab**



Train Model with GPU on Colab



Train Model with GPU on Colab



Keras + Tensorflow



Pytorch

Train Model with GPU on Colab



```
1 tf.config.list_physical_devices()

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

```
1 devices_type = [device.device_type for device in tf.config.list_physical_devices()]
2
3 devices_type

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

```
1 if 'GPU' in devices_type:
2   device_name = '/device:GPU:0'
3 else:
4   device_name = '/device:CPU:0'
```

Train Model with GPU on Colab



```
1 model = Sequential()  
2  
3 model.add(Dense(5, input_dim=X.shape[1], activation='relu'))  
4 model.add(Dense(4, activation='relu'))  
5 model.add(Dense(1, activation=None))  
6  
7 model.compile(loss=MeanSquaredError(), optimizer=SGD(learning_rate=0.01))
```

```
1 tqdm_callback = tfa.callbacks.TQDMProgressBar(show_epoch_progress=False)  
2  
3 with tf.device(device_name):  
4     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/104VDWLauVCY2scmmmcjROQxbXOIOzM7C?usp=sharing>

Train Model with GPU on Colab



```
1 if torch.cuda.is_available():
2     device = 'cuda:0'
3 else:
4     device = 'cpu'
```

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

Train Model with GPU on Colab



```
1 layers = []
2
3 layers.append(nn.Linear(X.shape[1], 5))
4 layers.append(nn.ReLU())
5
6 layers.append(nn.Linear(5, 4))
7 layers.append(nn.ReLU())
8
9 layers.append(nn.Linear(4, 1))
10
11 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



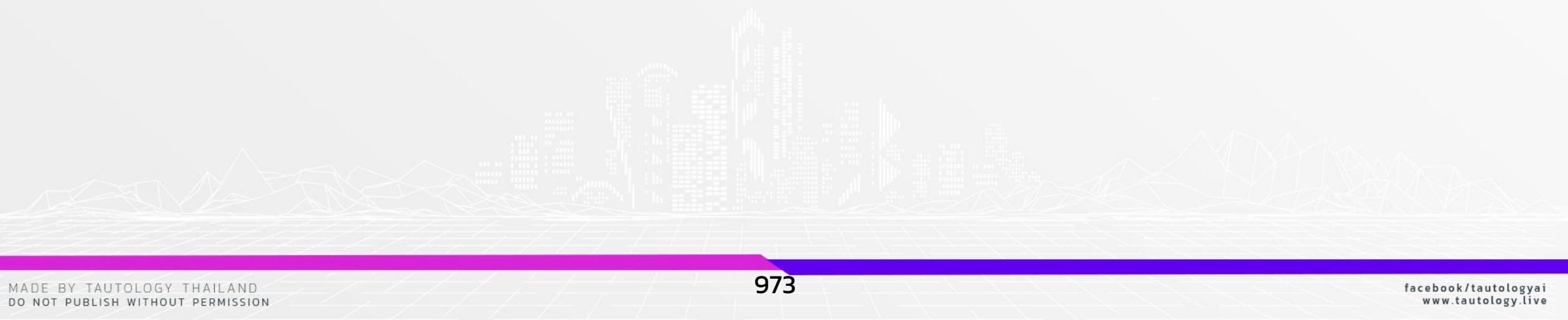
Imbalanced Class

Imbalanced Class

Problem with
Imbalanced Class

Solution

Code

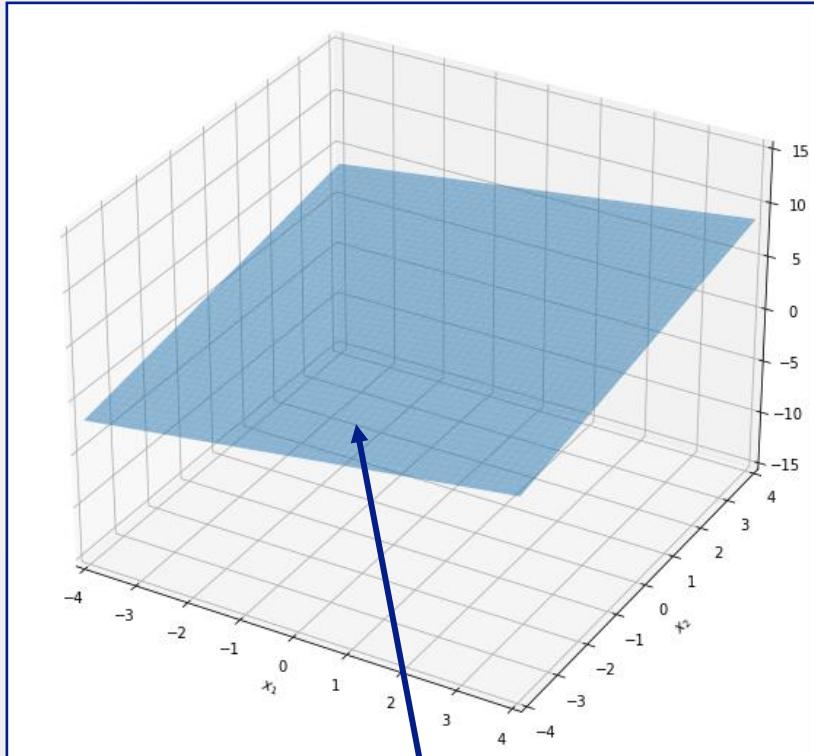


Problem with Imbalanced Class

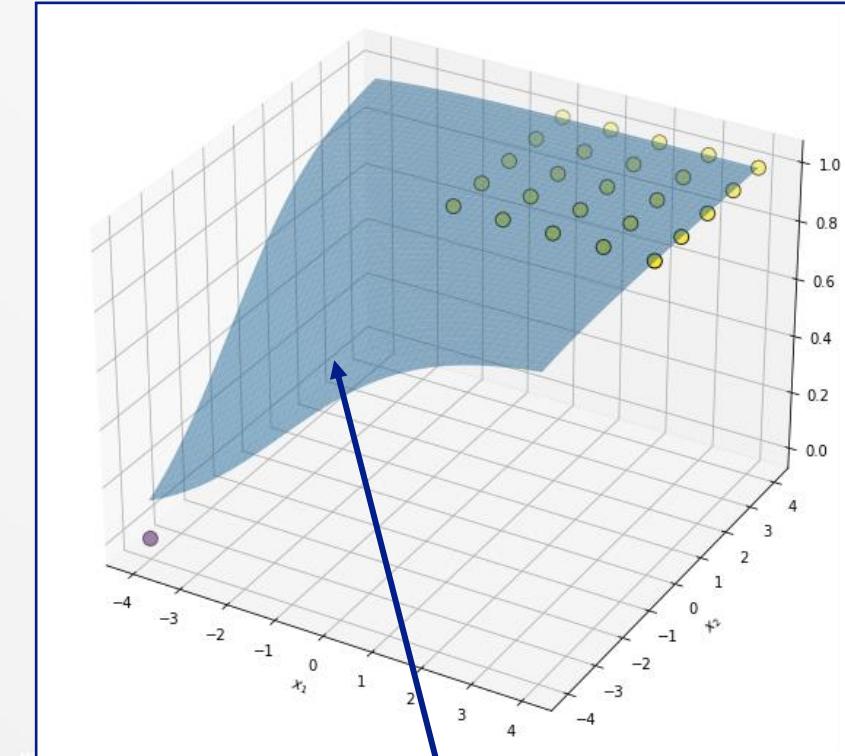
x_1	x_2	y
0	0	1
0	1	1
0	2	1
0	3	1
0	4	1
1	0	1
1	1	1
1	2	1
1	3	1
1	4	1
2	0	1
2	1	1
2	2	1

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

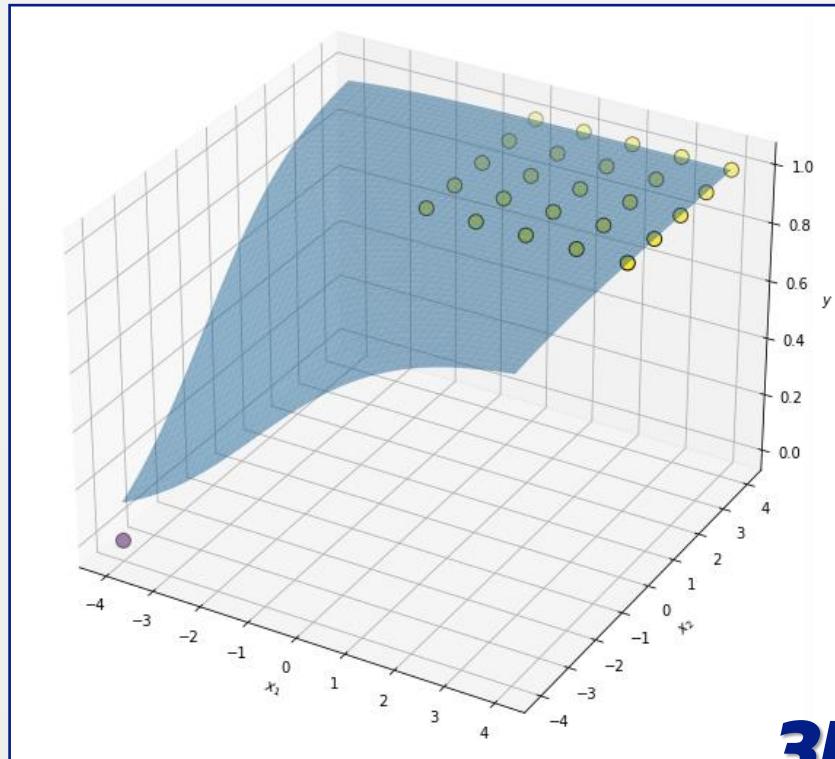


$$z = w_0 + w_1 x_1 + w_2 x_2$$

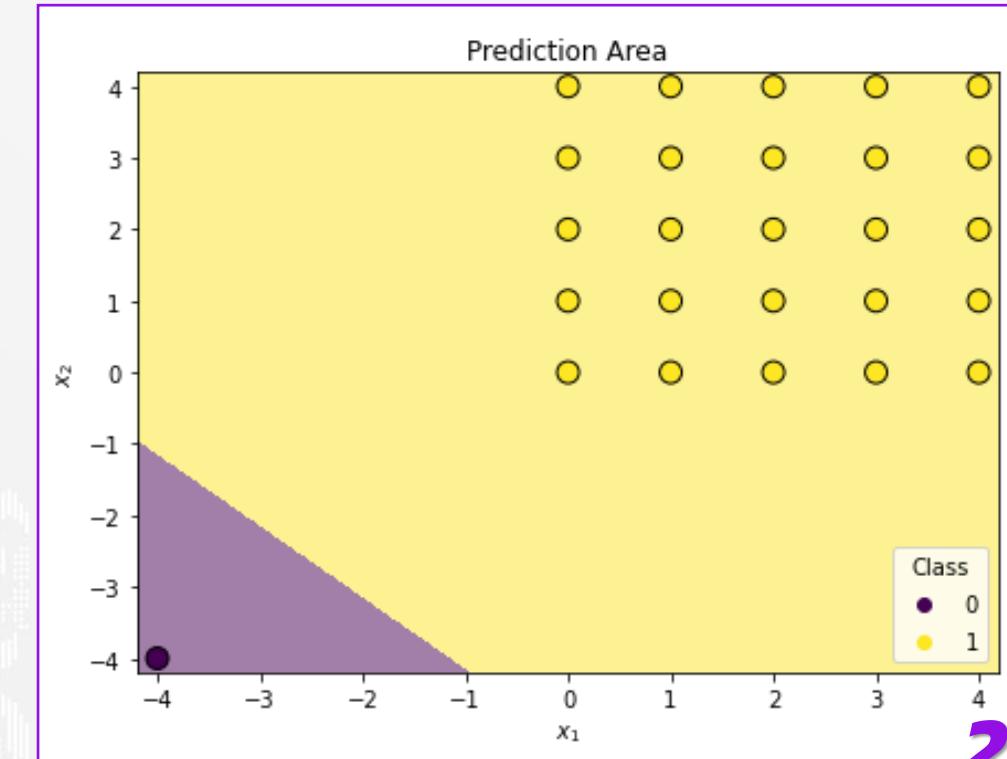


$$\hat{y} = \sigma(z)$$

Problem with Imbalanced Class

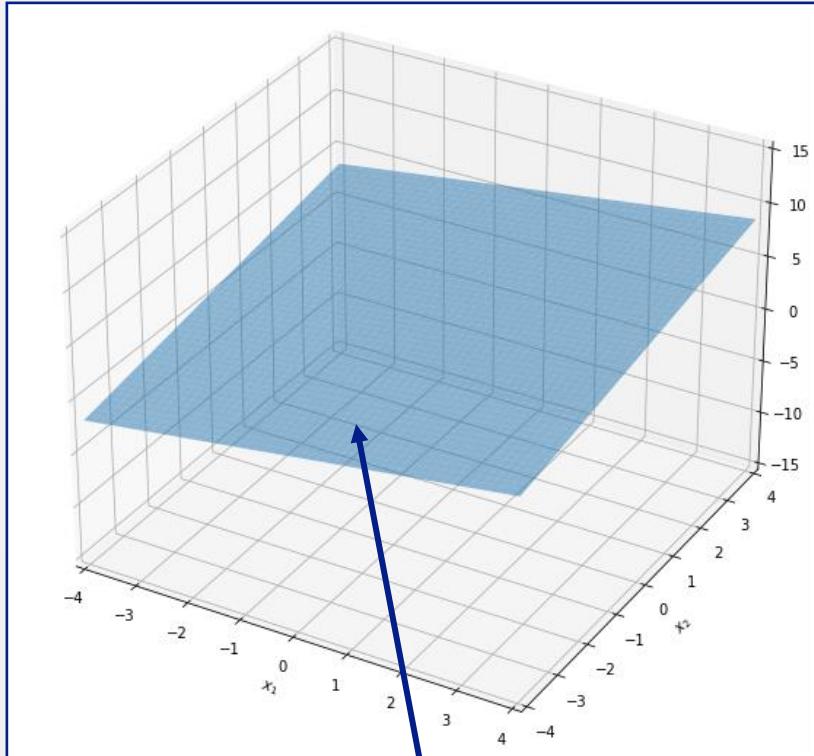


3D

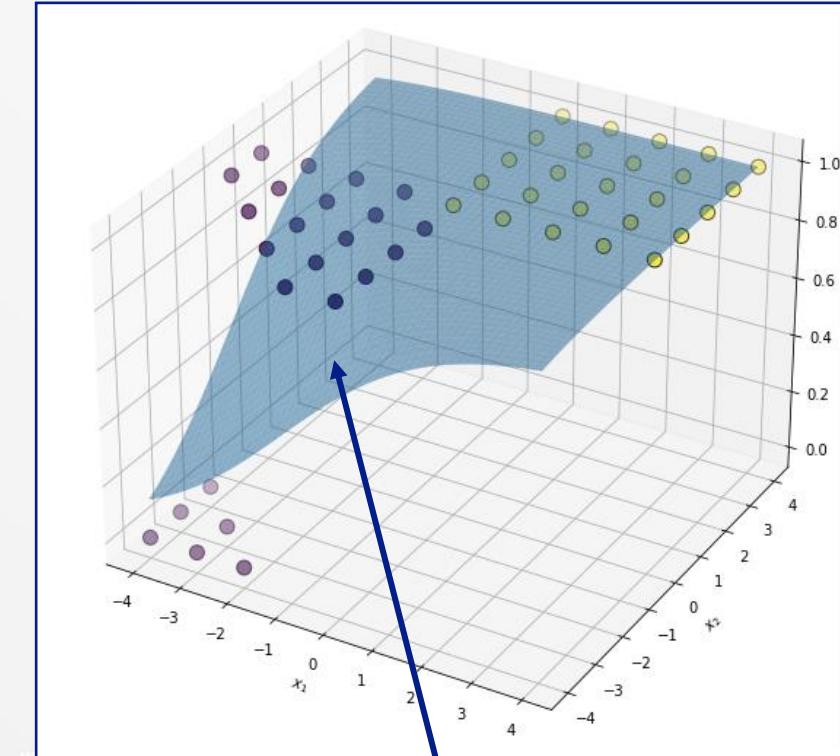


2D

Problem with Imbalanced Class

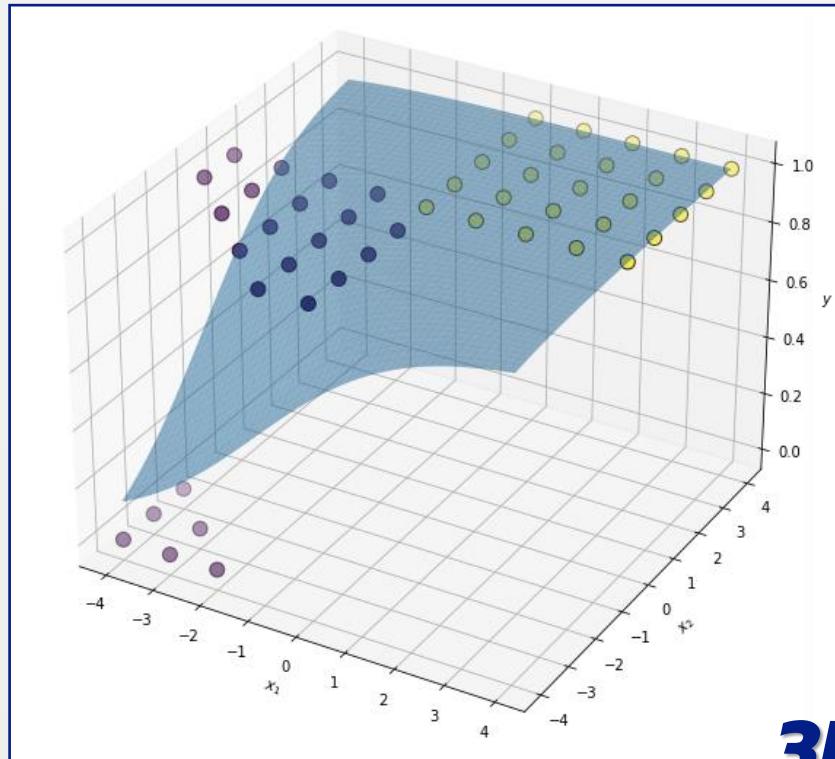


$$z = w_0 + w_1 x_1 + w_2 x_2$$

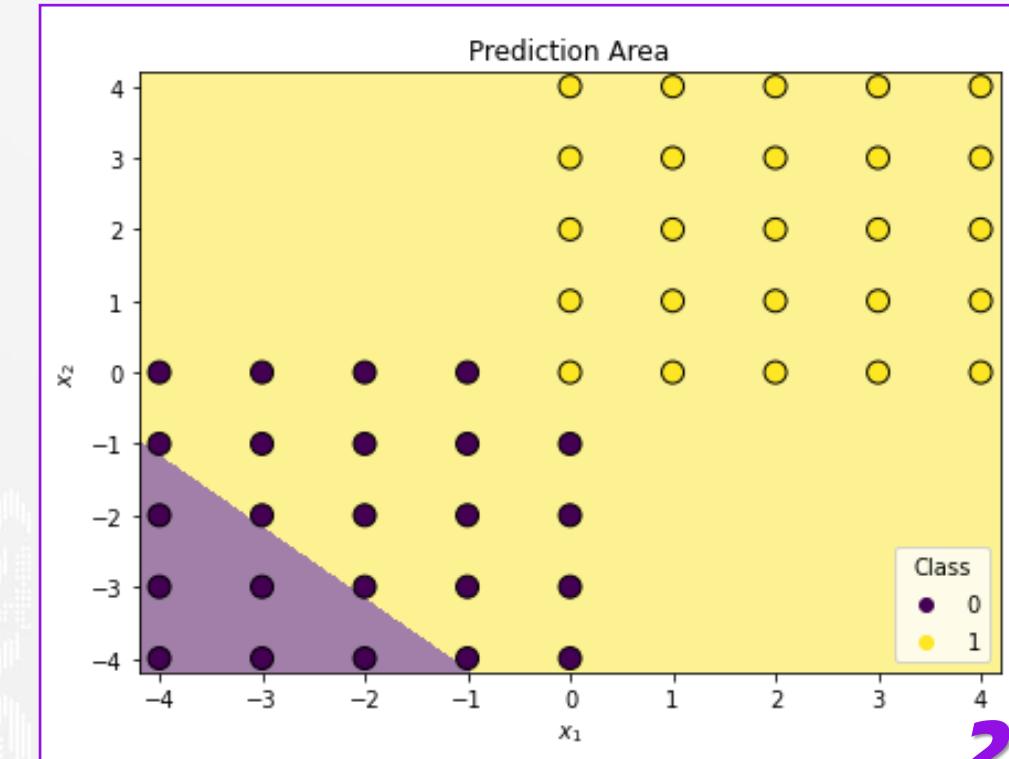


$$\hat{y} = \sigma(z)$$

Problem with Imbalanced Class

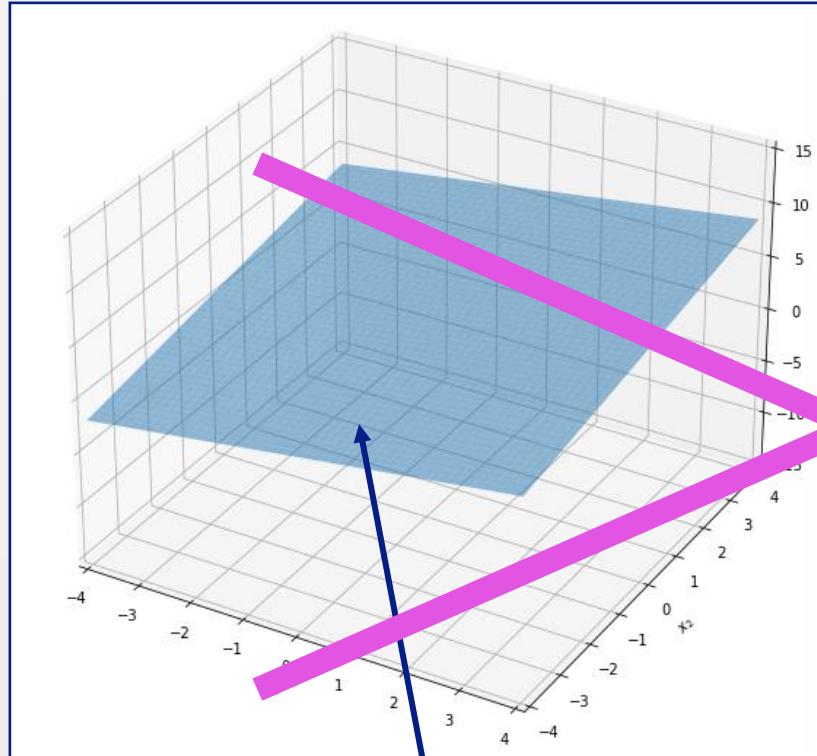


3D

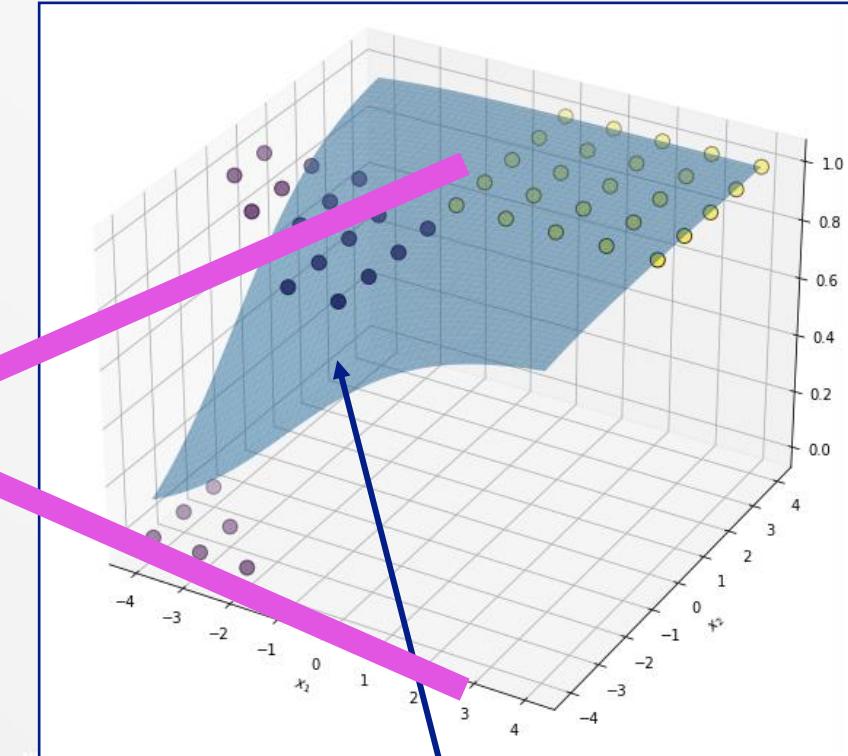


2D

Problem with Imbalanced Class

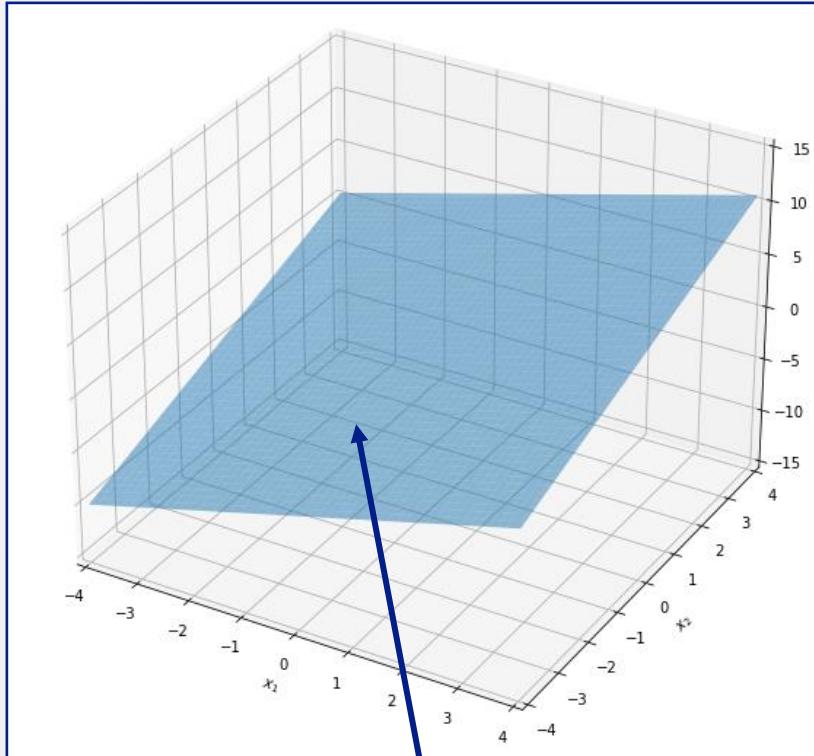


$$z = w_0 + w_1 x_1 + w_2 x_2$$

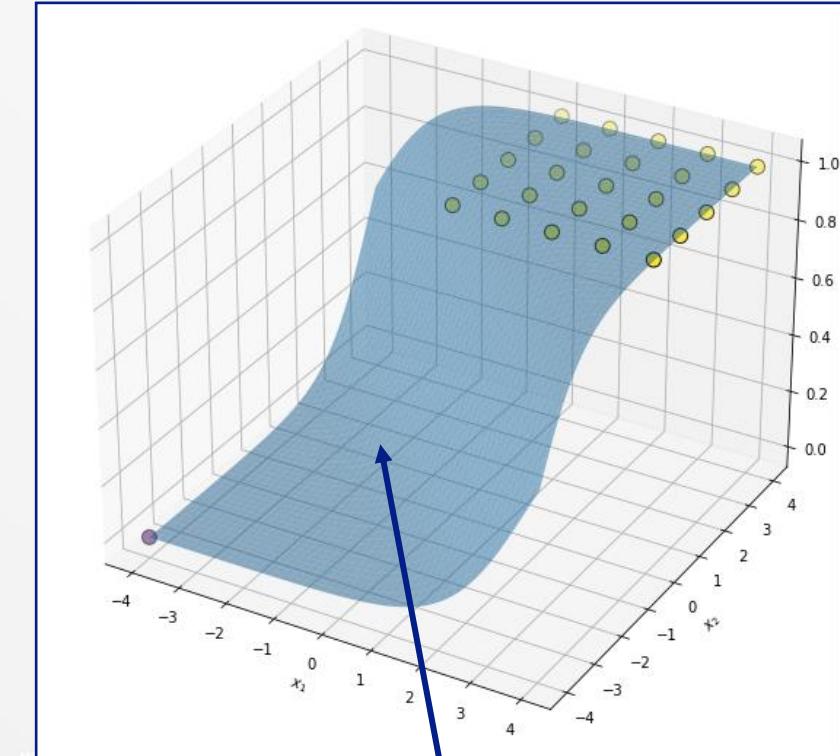


$$\hat{y} = \sigma(z)$$

Problem with Imbalanced Class

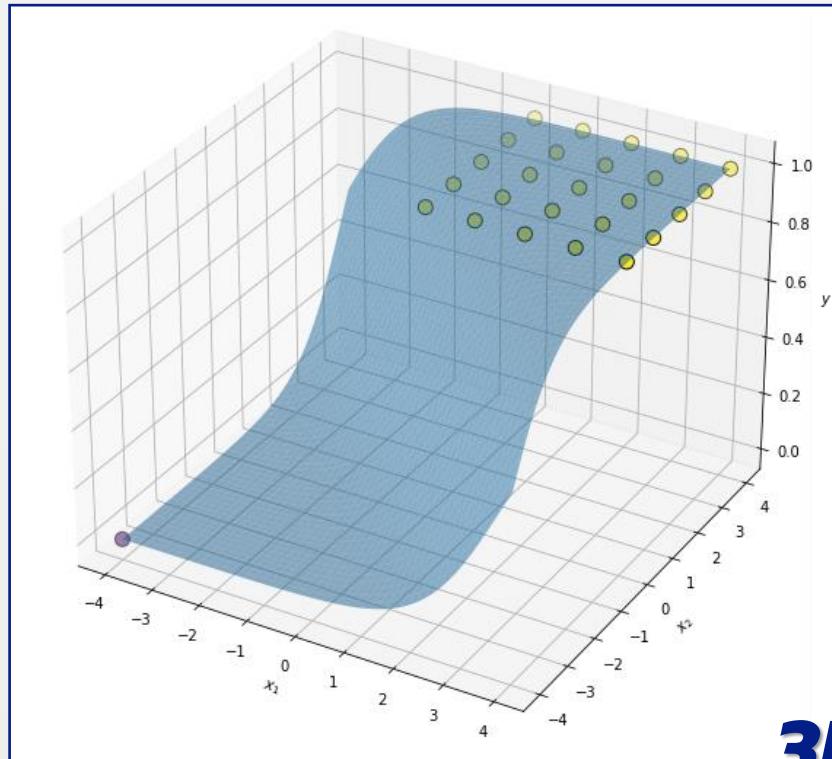


$$z = w_0 + w_1 x_1 + w_2 x_2$$

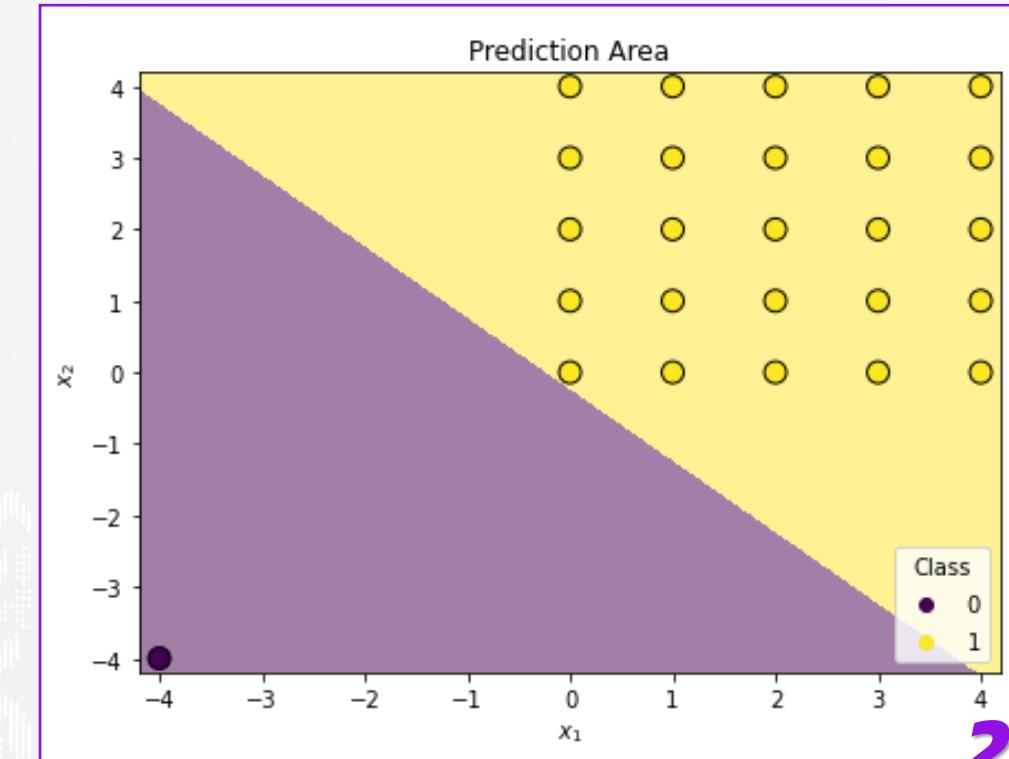


$$\hat{y} = \sigma(z)$$

Problem with Imbalanced Class

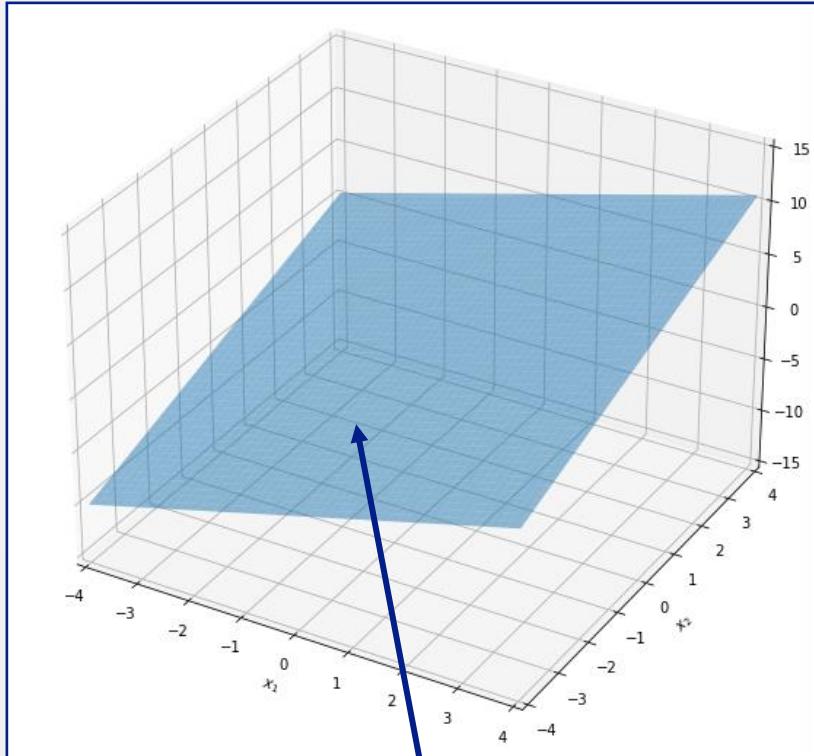


3D

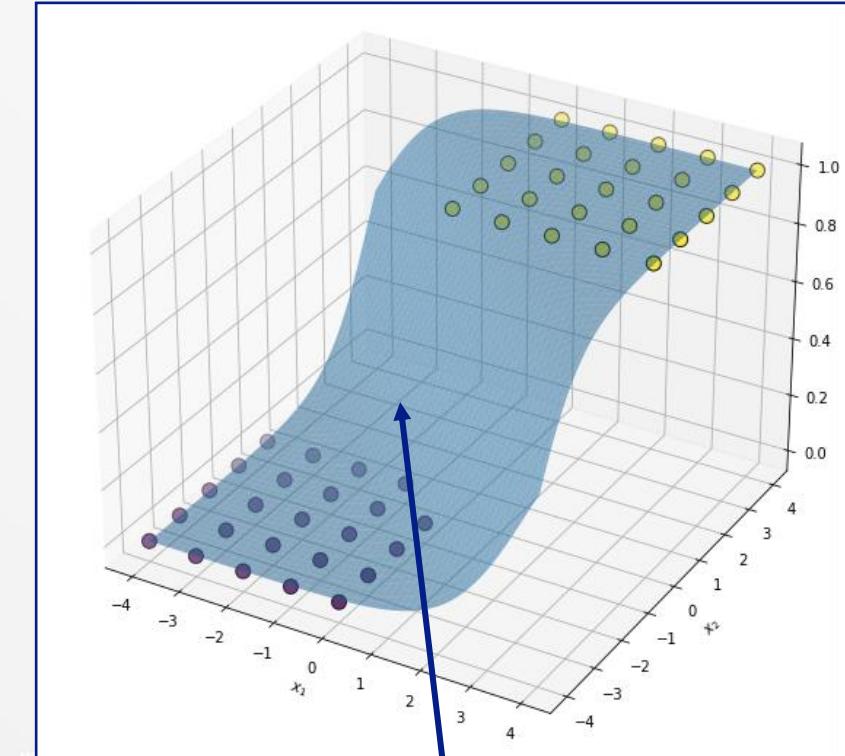


2D

Problem with Imbalanced Class

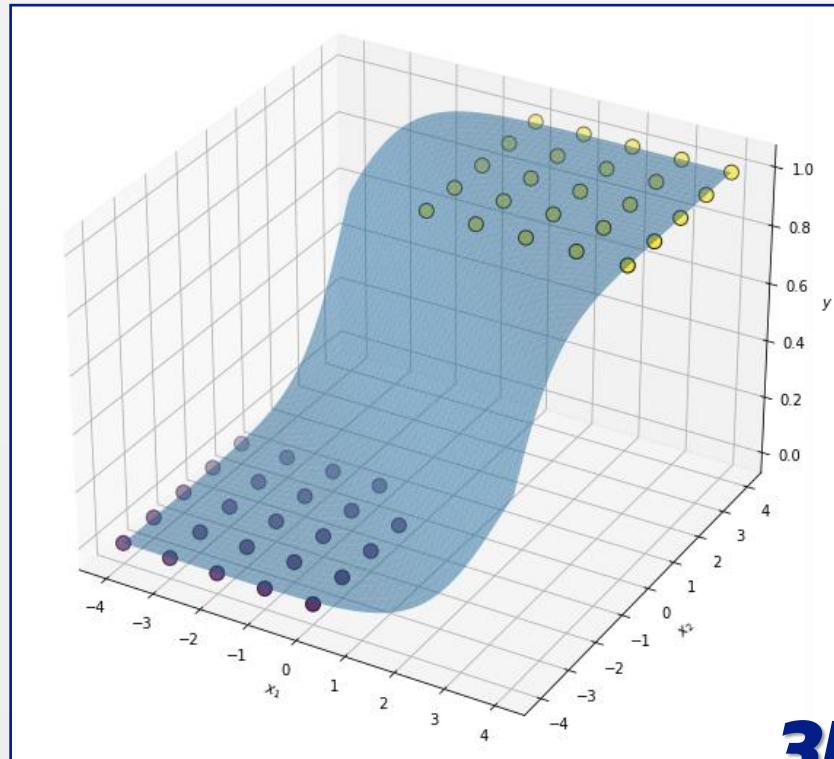


$$z = w_0 + w_1 x_1 + w_2 x_2$$

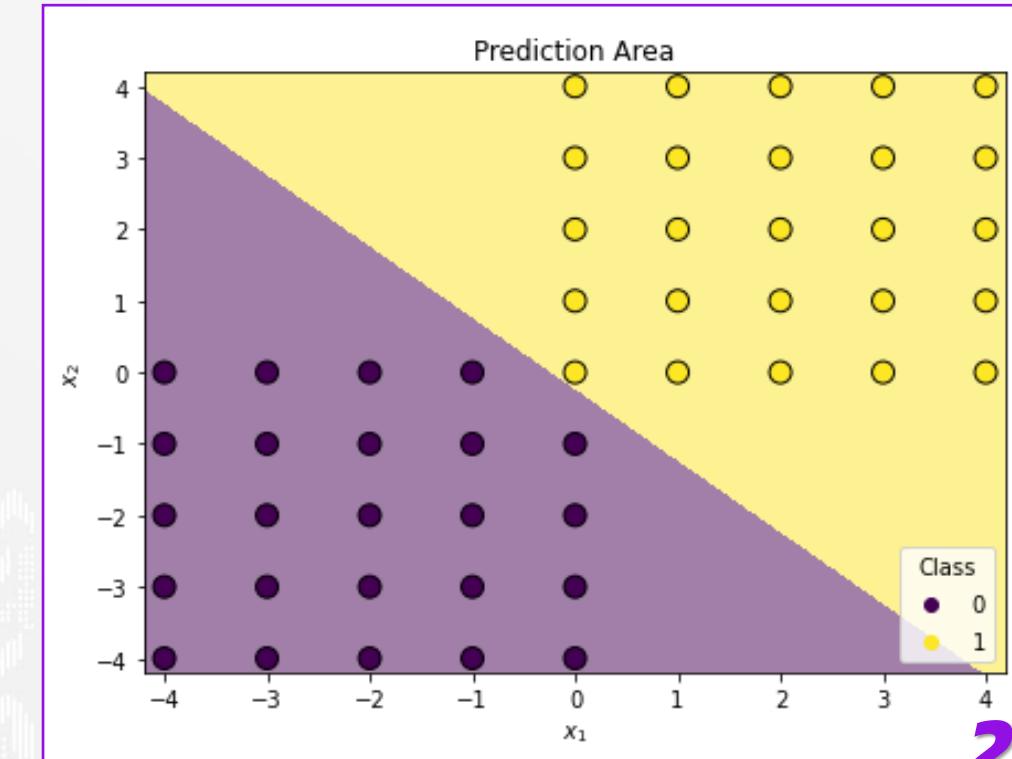


$$\hat{y} = \sigma(z)$$

Problem with Imbalanced Class



3D



2D

Imbalanced Class

**Problem with
Imbalanced Class**



Solution

Code

Solution

Bootstrapping

SMOTE

ADASYN

Balanced
Class Weight

Solution

Bootstrapping

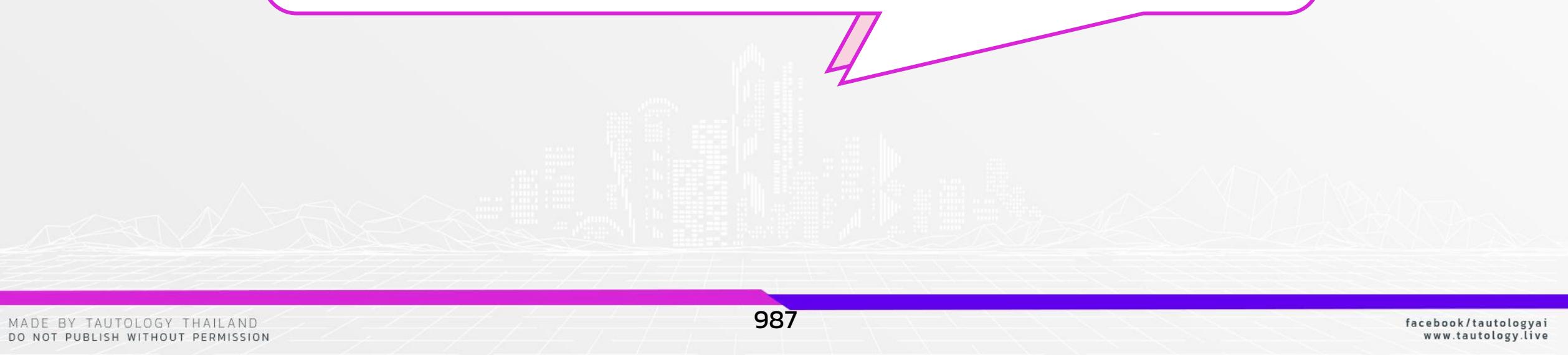
SMOTE

ADASYN

Balanced
Class Weight

Solution

หลักการทำงานของวิธี balanced class weight ก็คือ
เราจะให้ความสำคัญกับแต่ละ class อย่างเท่าเทียมกัน



Solution

x_1	x_2	y
0	0	1
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%
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**



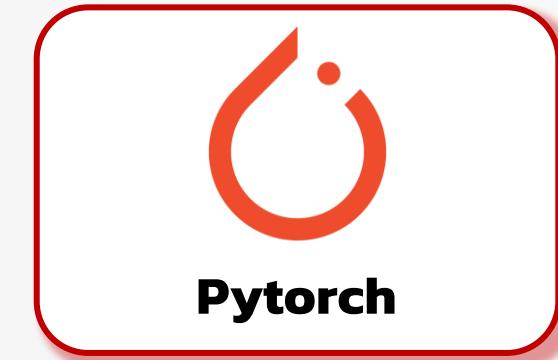
Solution



Code



Code



Code

■ Binary Classification



```
1 model = Sequential()  
2  
3 model.add(Dense(5, input_dim=X.shape[1], activation='relu'))  
4 model.add(Dense(4, activation='relu'))  
5 model.add(Dense(1, activation='sigmoid'))  
6  
7 model.compile(loss=BinaryCrossentropy(), optimizer=SGD(learning_rate=1))
```

Code

■ Binary Classification



```
1 class_weight = compute_class_weight('balanced', np.unique(y), y)
2 class_weight = dict(enumerate(class_weight.flatten()))
3
4 tqdm_callback = tfa.callbacks.TQDMProgressBar(show_epoch_progress=False)
5
6 with tf.device(device_name):
7     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



```
1 model = Sequential()  
2  
3 model.add(Dense(5, input_dim=X.shape[1], activation='relu'))  
4 model.add(Dense(4, activation='relu'))  
5 model.add(Dense(y_enc.shape[1], activation='softmax'))  
6  
7 model.compile(loss=CategoricalCrossentropy(), optimizer=SGD(learning_rate=1))
```

Code

■ Multi-Class Classification



```
1 class_weight = compute_class_weight('balanced', np.unique(y), y)
2 class_weight = dict(enumerate(class_weight.flatten()))
3
4 tqdm_callback = tfa.callbacks.TQDMProgressBar(show_epoch_progress=False)
5
6 with tf.device(device_name):
7     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



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

Code

■ Binary Classification



```
1 layers = []
2
3 layers.append(nn.Linear(X.shape[1], 5))
4 layers.append(nn.ReLU())
5
6 layers.append(nn.Linear(5, 4))
7 layers.append(nn.ReLU())
8
9 layers.append(nn.Linear(4, 1))
10
11 model = nn.Sequential(*layers).to(device)
```



Code

■ Binary Classification



```
1 optimizer = torch.optim.SGD(model.parameters(), lr=1)
2
3 pos_weight = (len(y_le) - y_le.sum())/y_le.sum()
4 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



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

Code

■ Multi-Class Classification



```
1 layers = []
2
3 layers.append(nn.Linear(X.shape[1], 5))
4 layers.append(nn.ReLU())
5
6 layers.append(nn.Linear(5, 4))
7 layers.append(nn.ReLU())
8
9 layers.append(nn.Linear(4, 3))
10
11 model = nn.Sequential(*layers).to(device)
```

Code

■ Multi-Class Classification



```
1 optimizer = torch.optim.SGD(model.parameters(), lr=1)
2
3 class_weight = compute_class_weight('balanced', np.unique(y), y)
4 class_weight = torch.from_numpy(class_weight).float()
5 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**



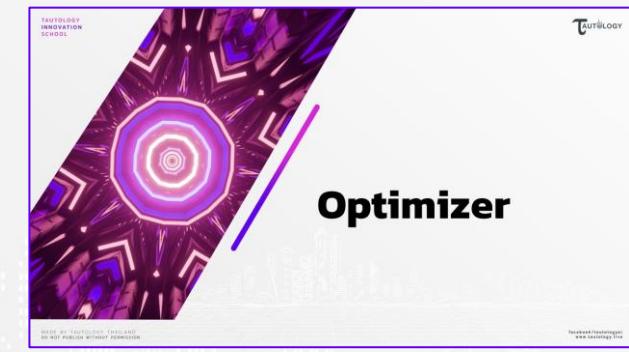
Solution



Code



Improvement of Deep Learning



Regularization

Regularization

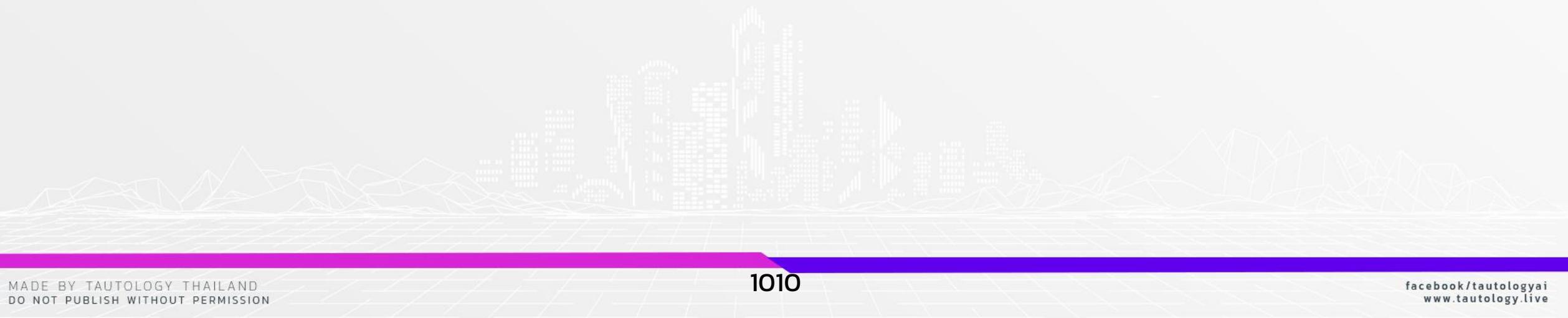
อ้างอิงจากงานวิจัยของ linear regression ในปี 1970 นักคณิตศาสตร์ค้นพบว่า

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

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

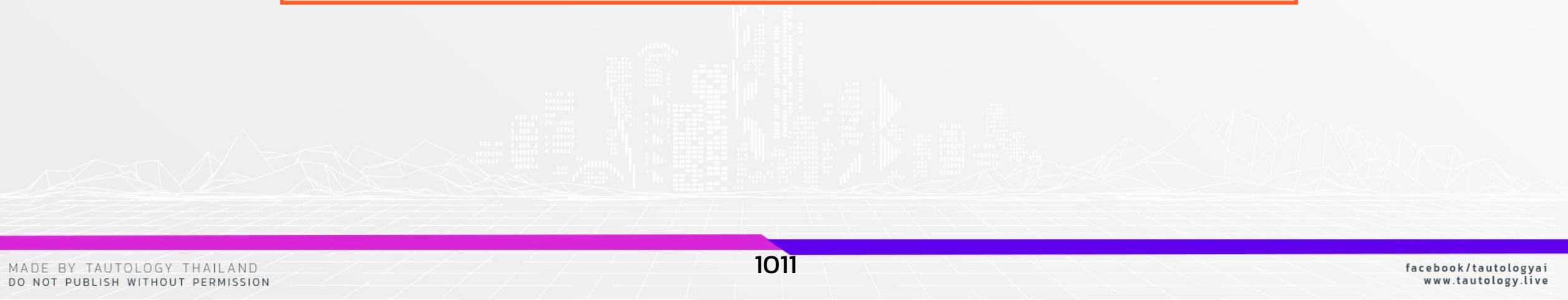
Regularization

และเบื้องจาก 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

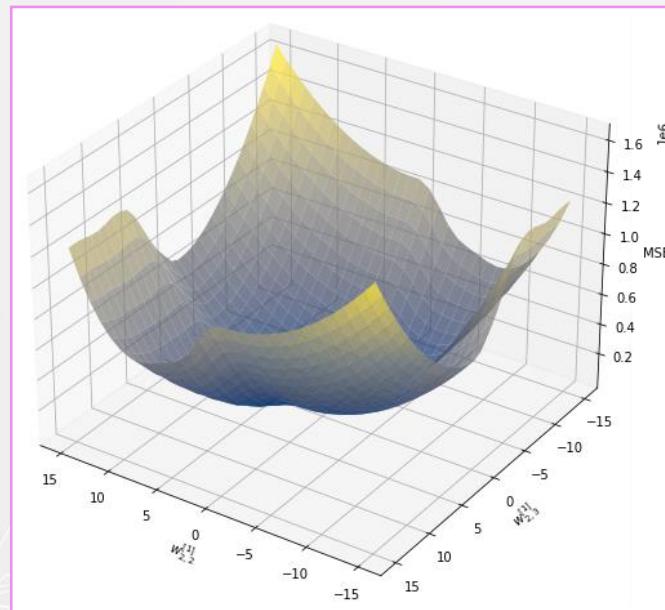
Elastic Net

Dropout
Regularization

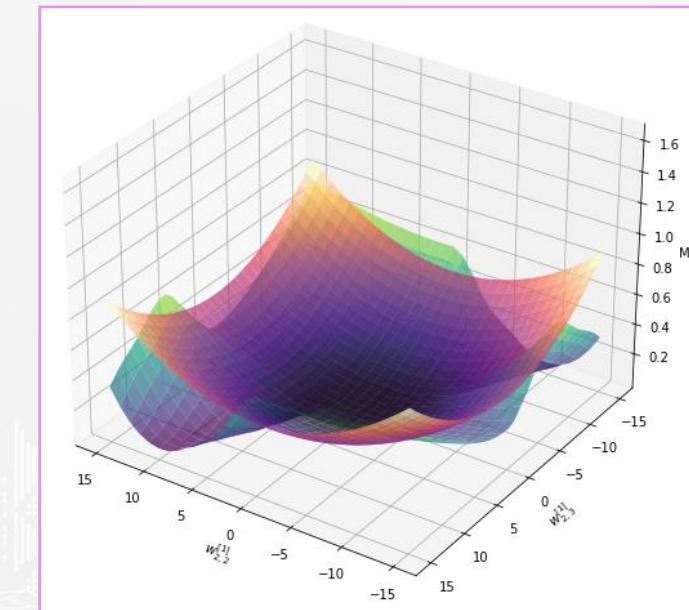
L2 Regularization

- Linearly Independent

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



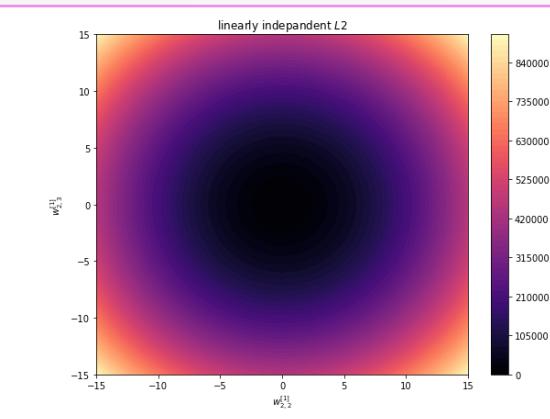
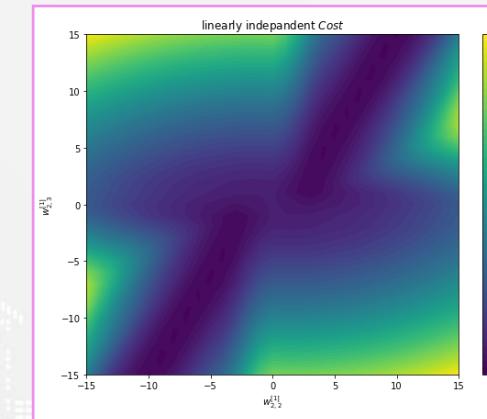
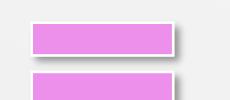
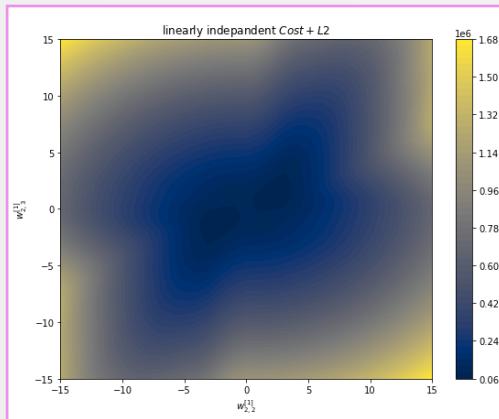
1013



L2 Regularization

■ Linearly Independent

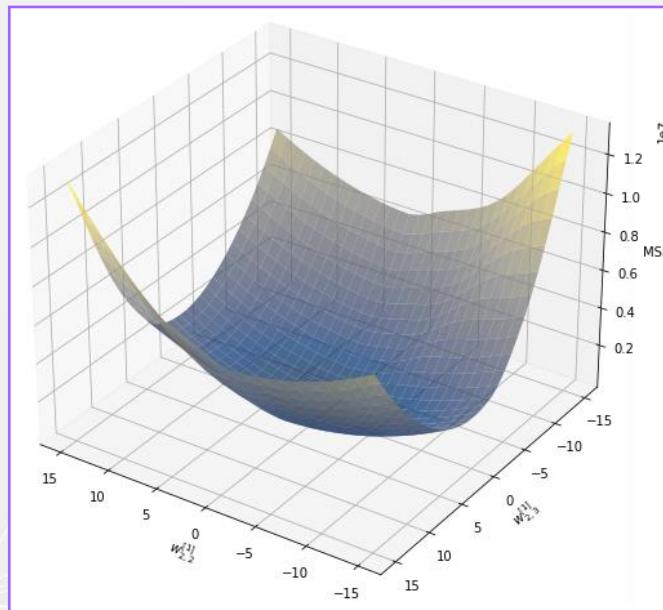
$$Cost = Cost_0 + \lambda(\sum b^2 + \sum w^2)$$



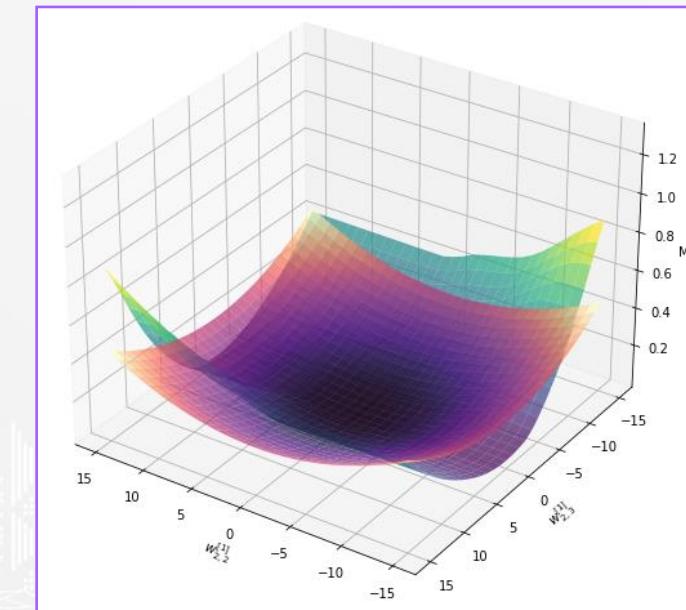
L2 Regularization

- Linearly Dependent

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



UNREGULARIZED
MSE

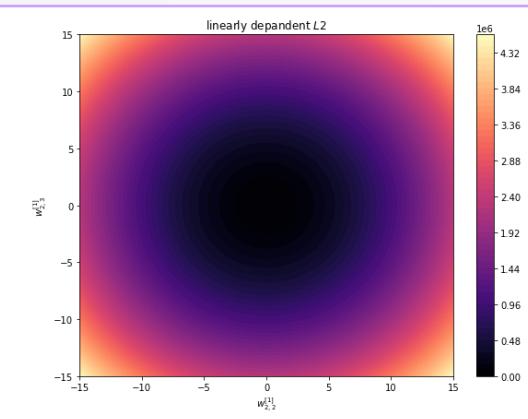
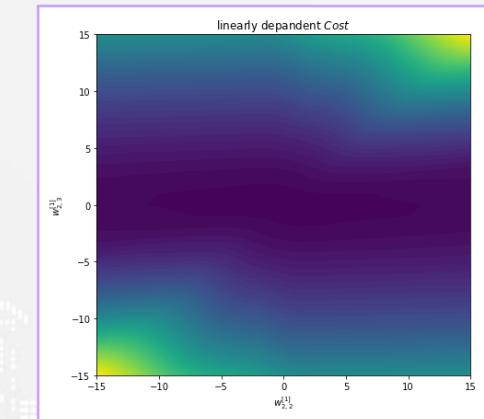
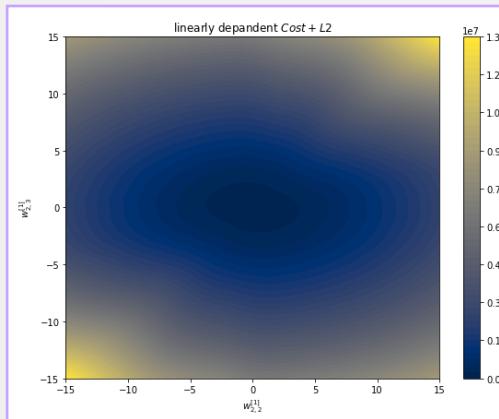


REGULARIZED
MSE

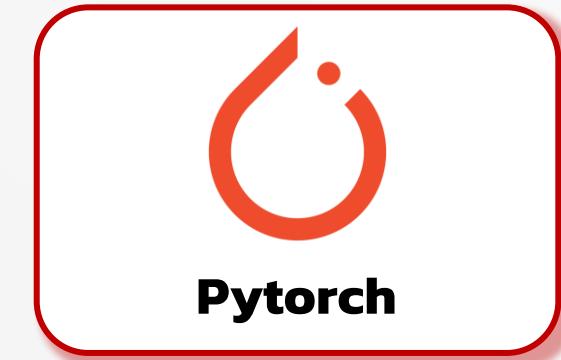
L2 Regularization

■ Linearly Dependent

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



L2 Regularization



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



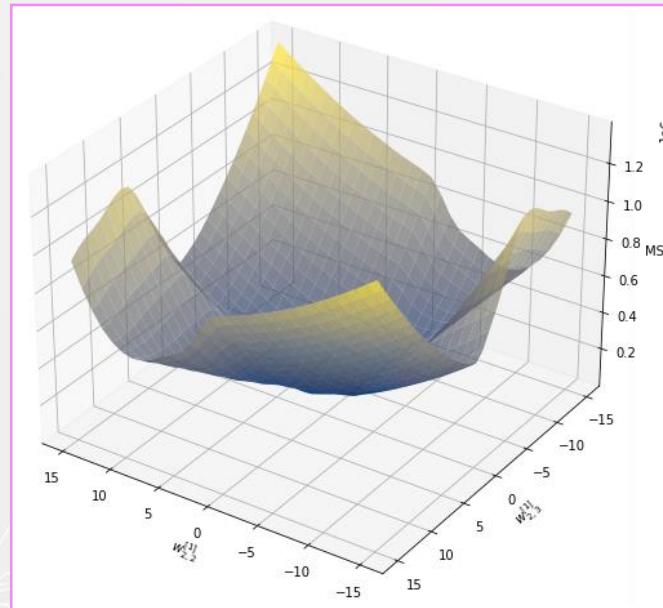
**Dropout
Regularization**



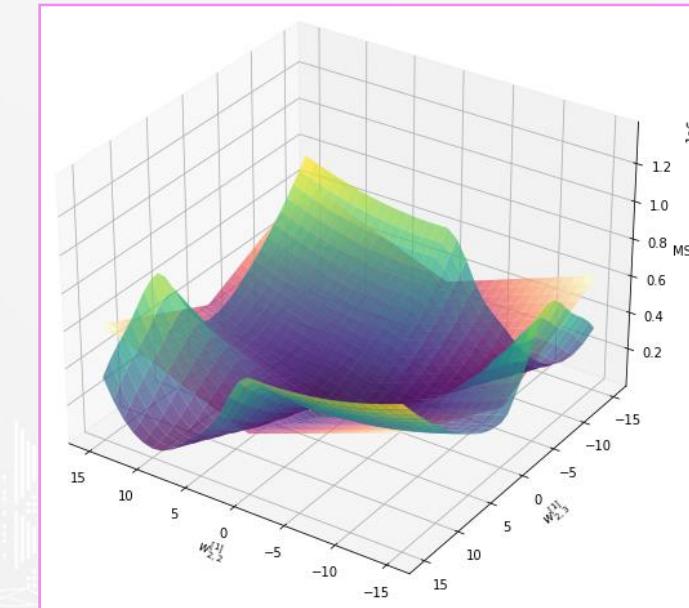
L1 Regularization

- Linearly Independent

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



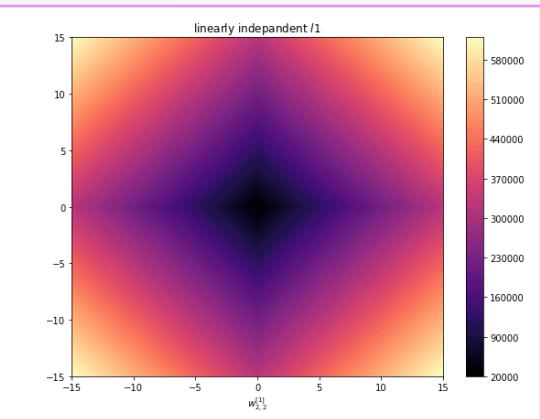
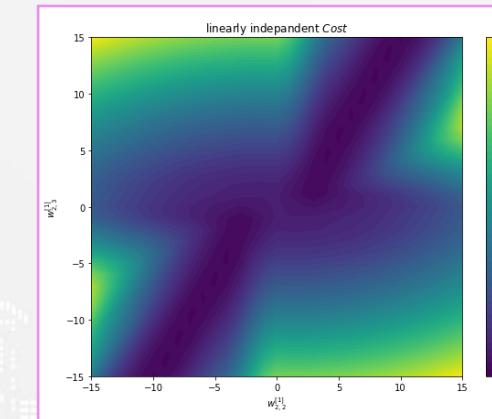
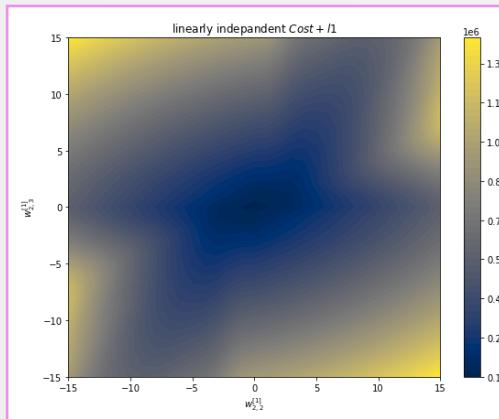
▪▪▪



L1 Regularization

■ Linearly Independent

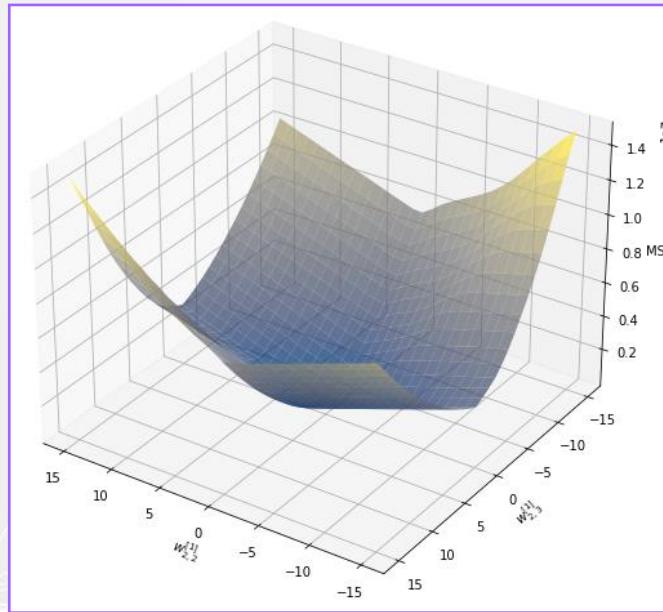
$$Cost = Cost_0 + \lambda(\sum|b| + \sum|w|)$$



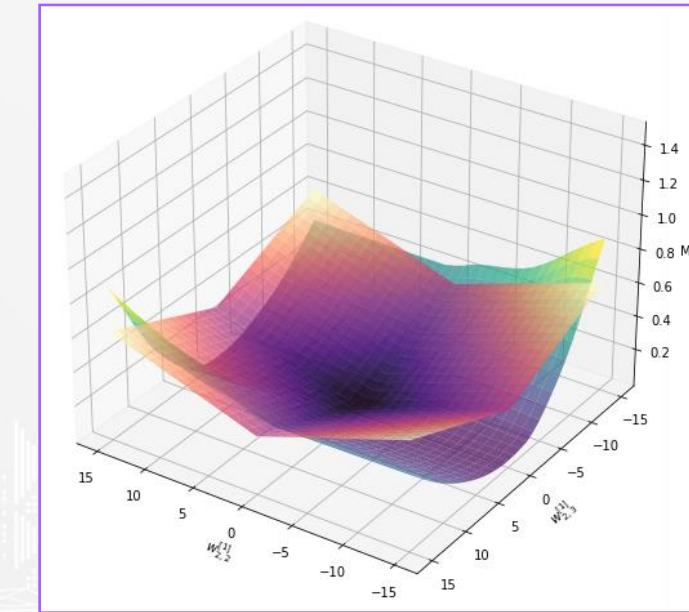
L1 Regularization

- Linearly Dependent

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



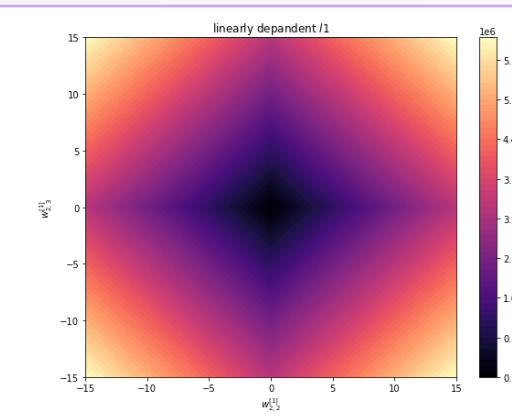
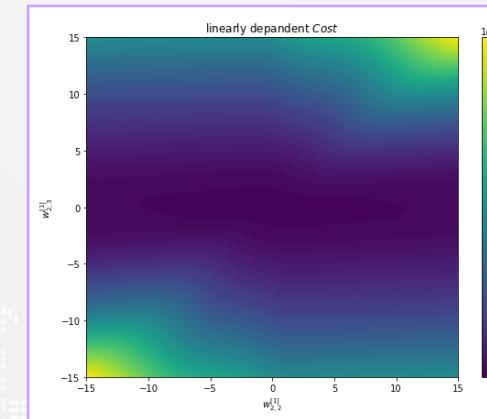
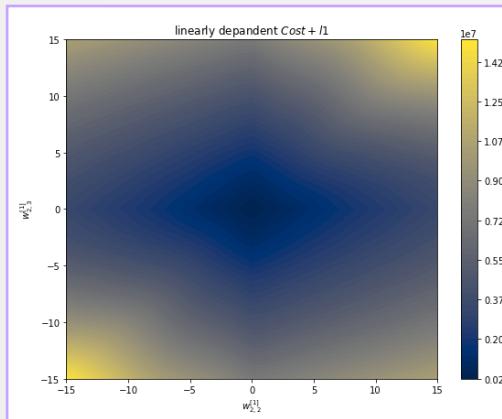
1024



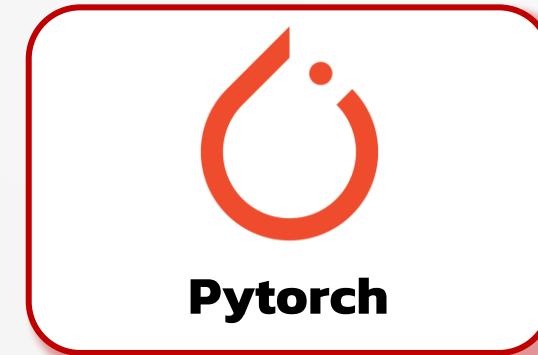
L1 Regularization

■ Linearly Dependent

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



L1 Regularization



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**



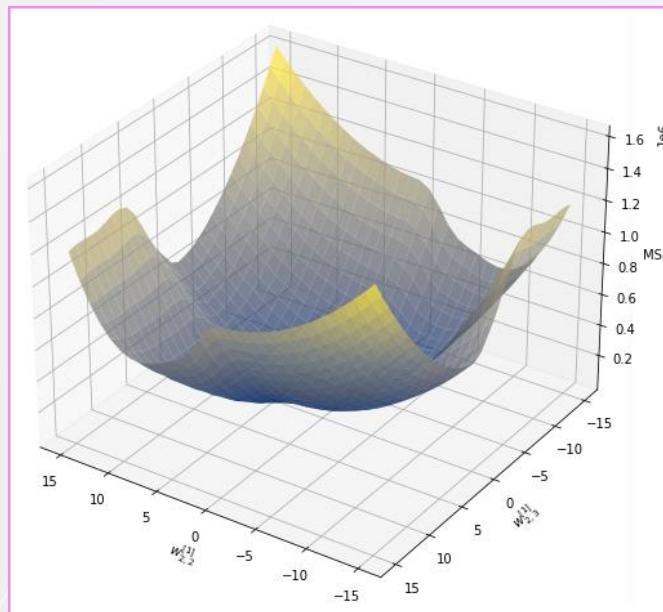
Elastic Net

Elastic net คือ การทำ regularization โดยมี cost function เป็น

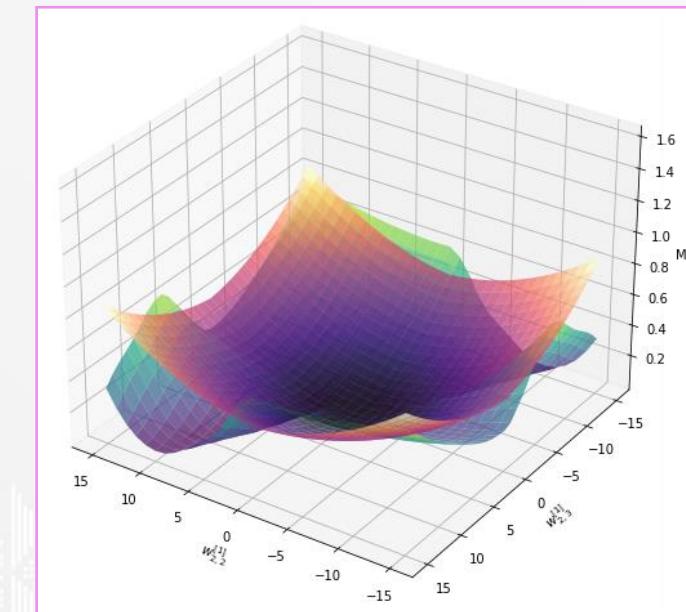
$$\begin{aligned} Cost = & Cost_0 + \lambda l1_{ratio} (\sum |b| + \sum |w|) \\ & + \lambda (1 - l1_{ratio}) (\sum b^2 + \sum w^2) \end{aligned}$$

Elastic Net

■ Linearly Independent

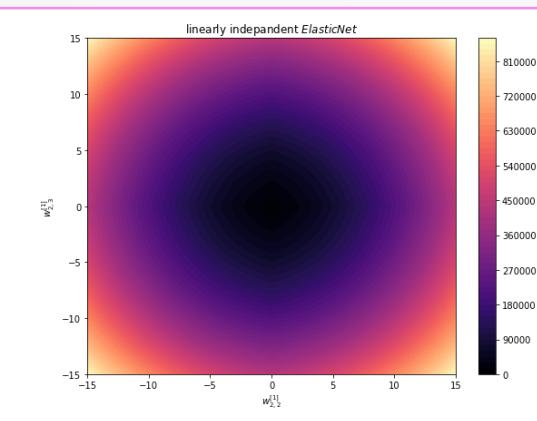
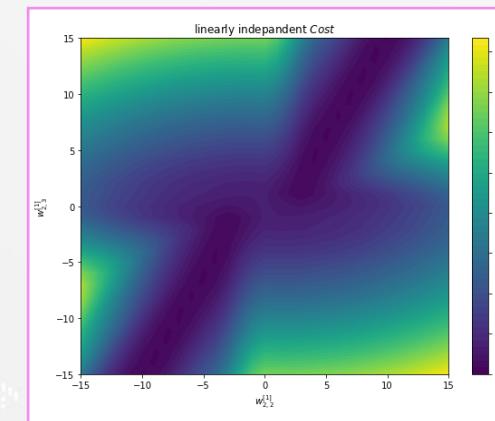
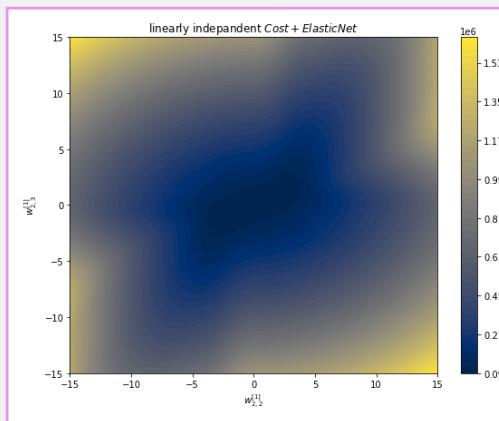


1031



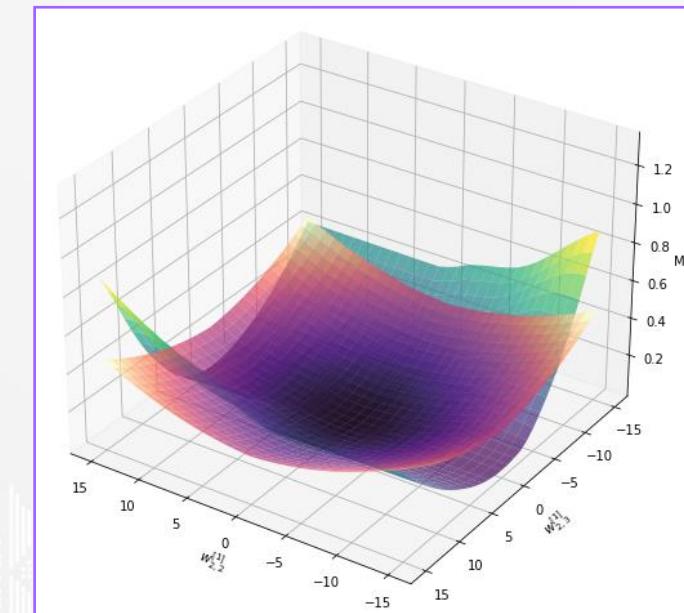
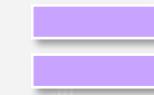
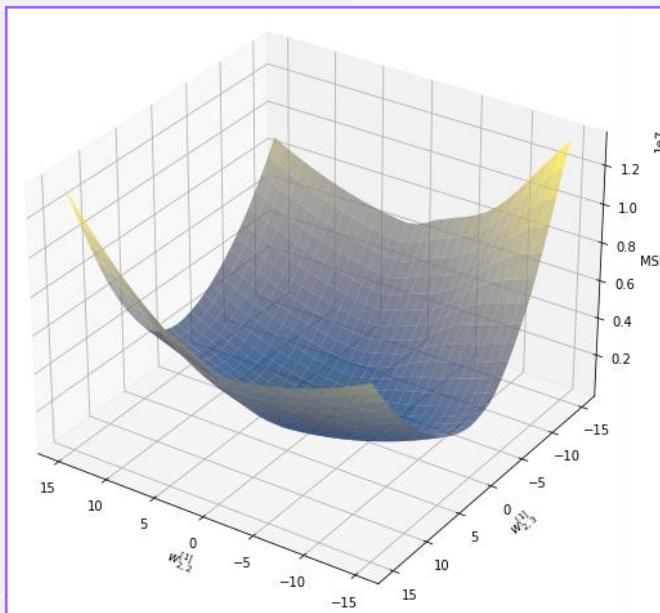
Elastic Net

■ Linearly Independent



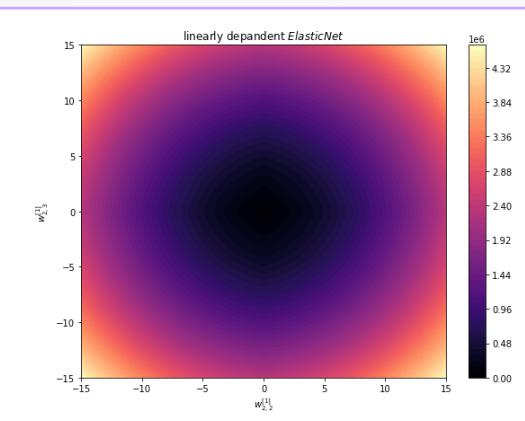
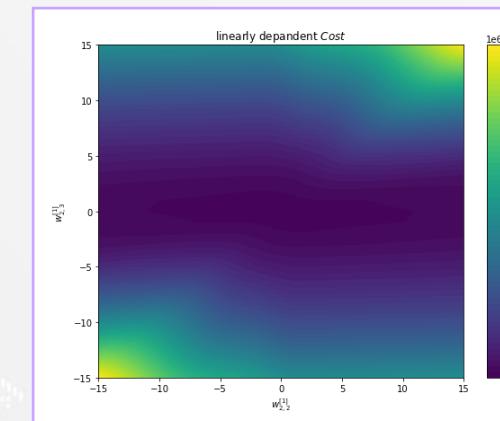
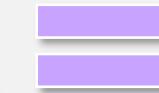
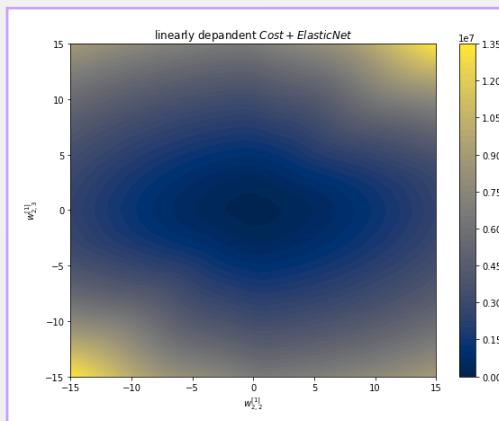
Elastic Net

- Linearly Dependent

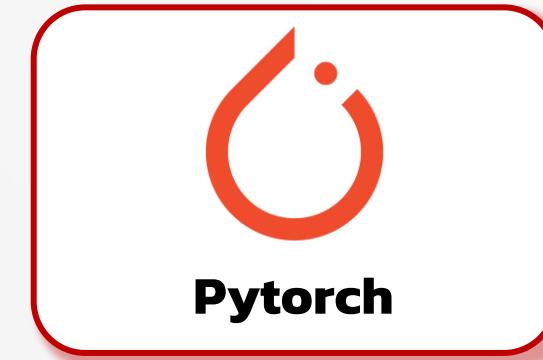


Elastic Net

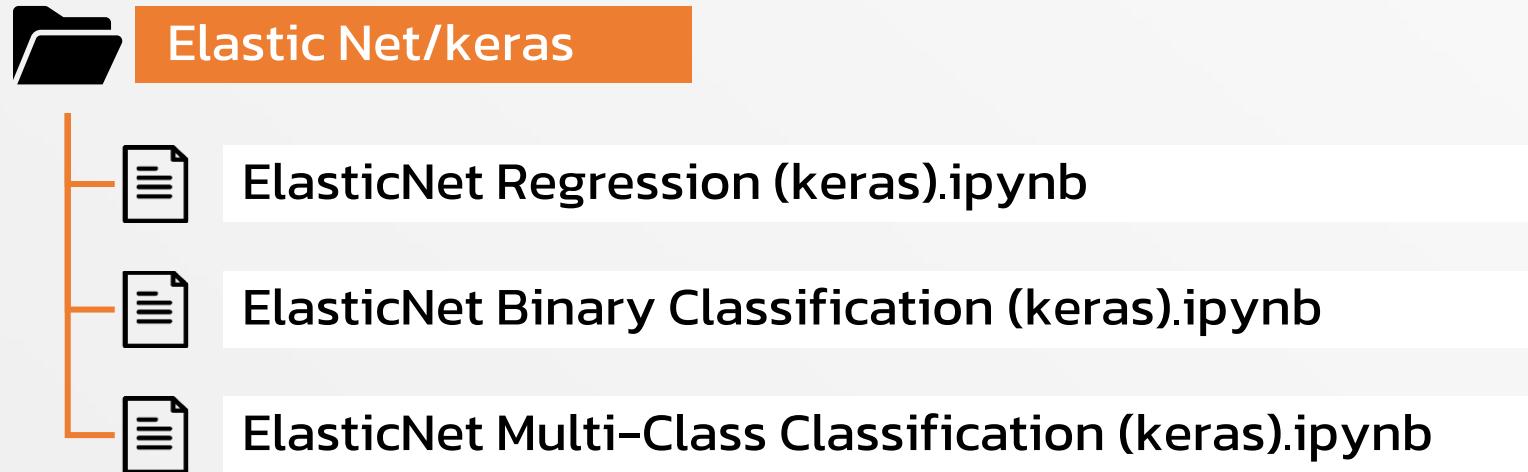
■ Linearly Dependent



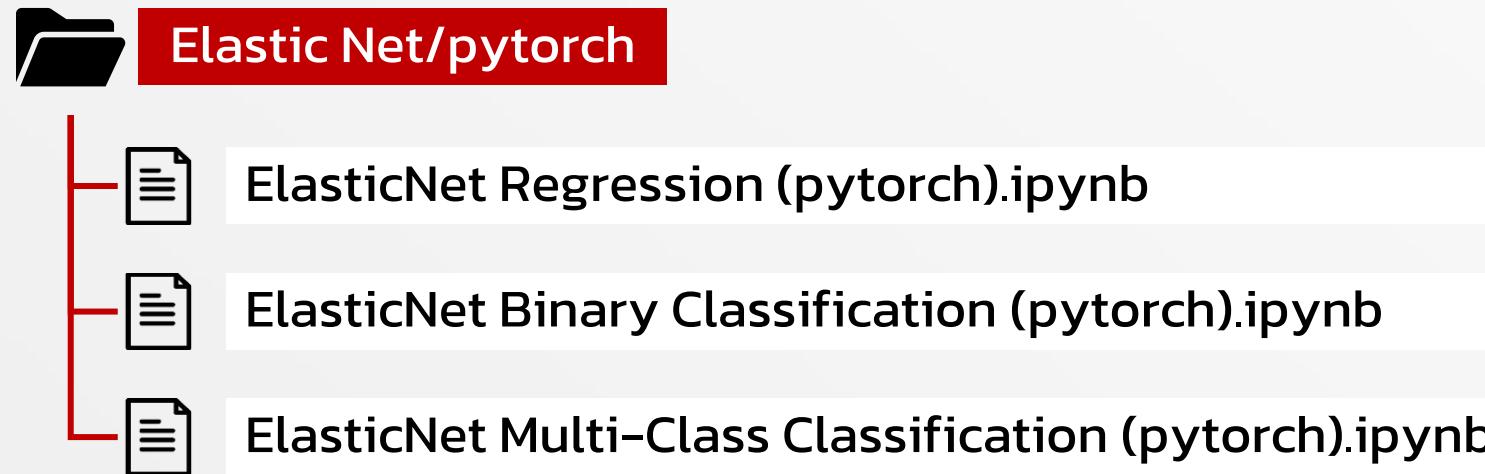
Elastic Net



Elastic Net



Elastic Net



Regularization

L2 Regularization



L1 Regularization



Elastic Net



**Dropout
Regularization**

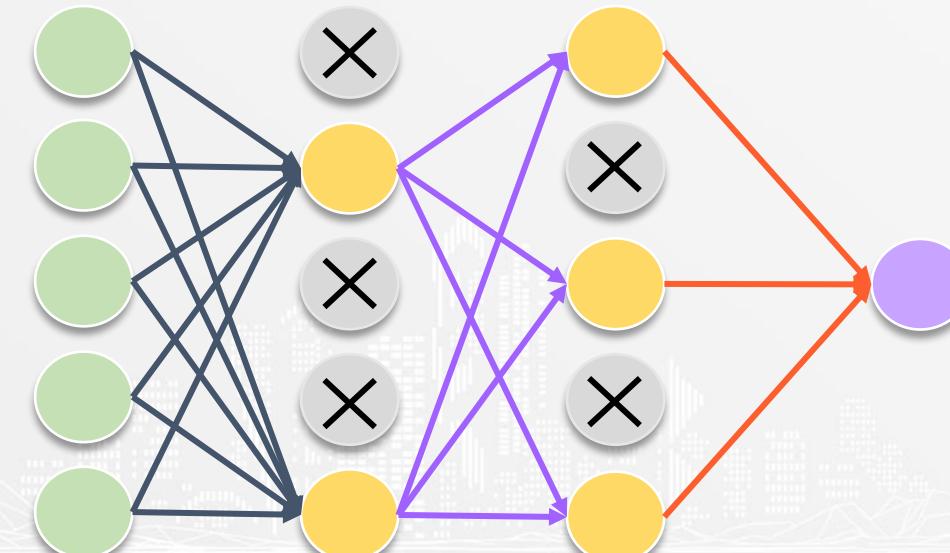


Dropout Regularization

- 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)



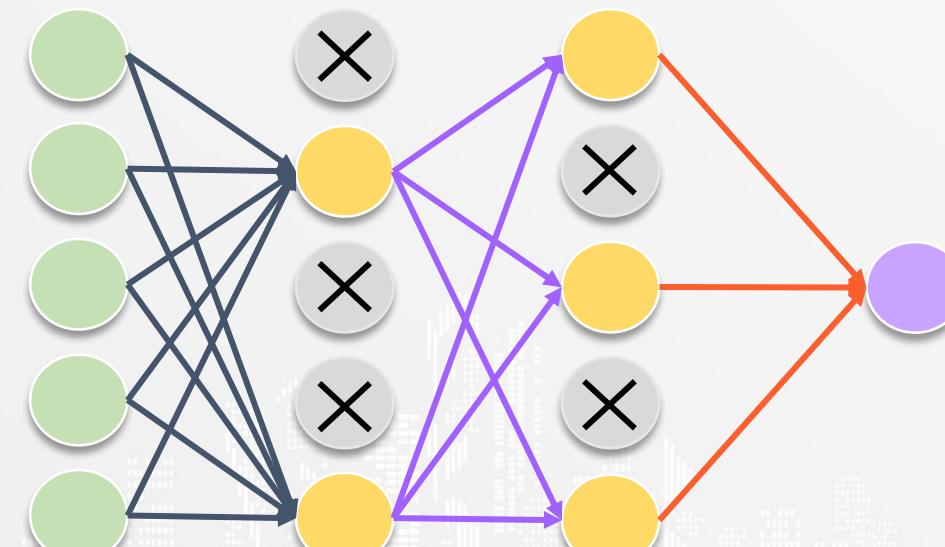
Dropout Regularization

What is Dropout Regularization?

- Effect of Dropout
- Technical of Dropout
- Code of Dropout

Effect of Dropout

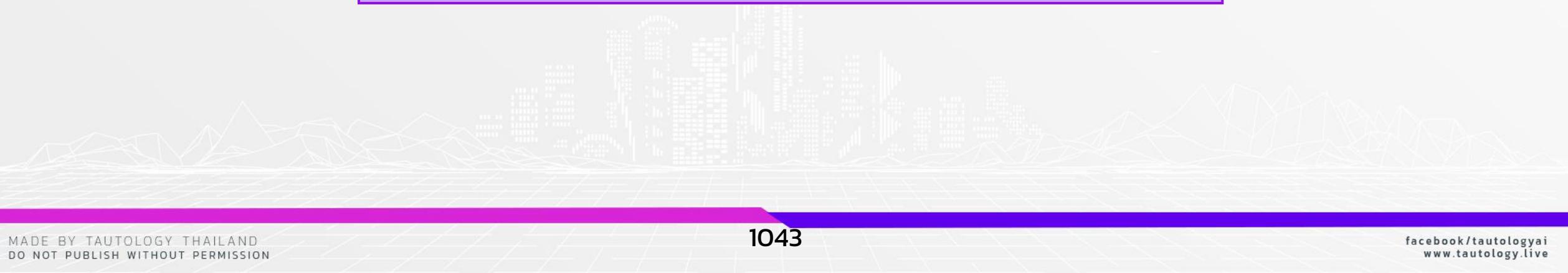
Effect of dropout คือ เมื่อบาง node ถูก drop ในระหว่างการ train model ทำให้ model ถูกบังคับให้ใช้เส้นทางอื่นที่ไม่ผ่าน node ที่ถูก drop ในการพยากรณ์ output



Effect of Dropout

เมื่อนำ model ไปใช้งานจริง model จะมีหลายเส้นทางที่สามารถใช้ในการพยากรณ์ output ได้

การใช้เส้นทางเหล่านี้ร่วมกันในการพยากรณ์ส่งผลให้ model มีความ general มาขึ้น



Dropout Regularization

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

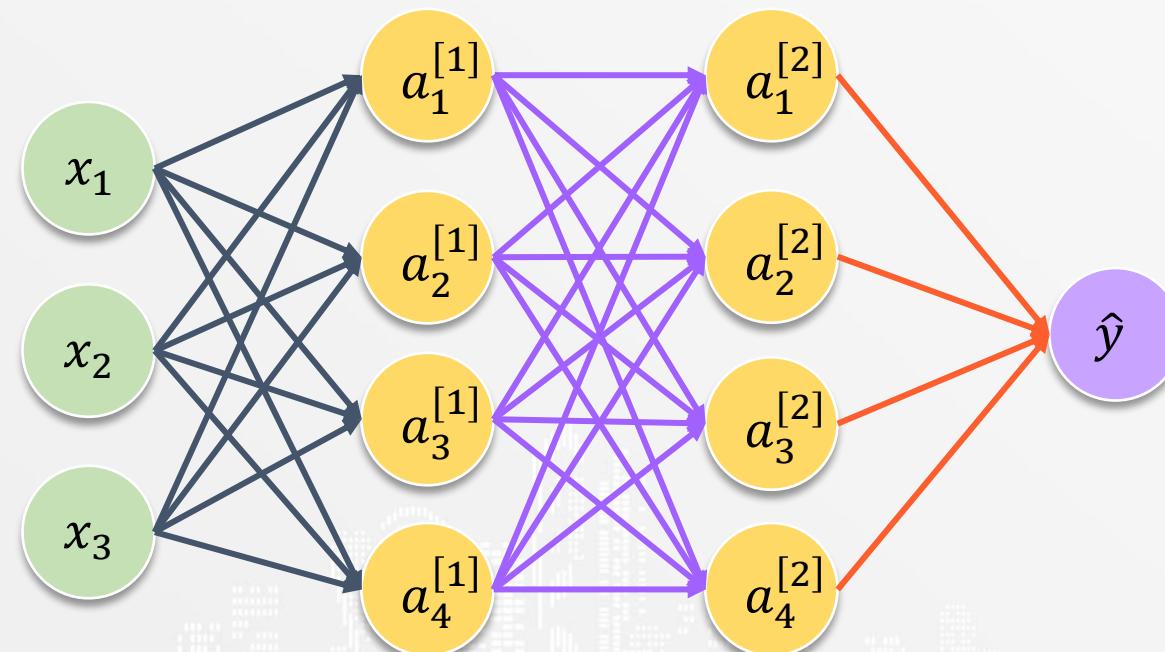
Technical of Dropout

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

ຕົວຢ່າງ



Technical of Dropout

ตัวอย่าง

- กำหนด `drop_rate` สำหรับแต่ละ `hidden layer`

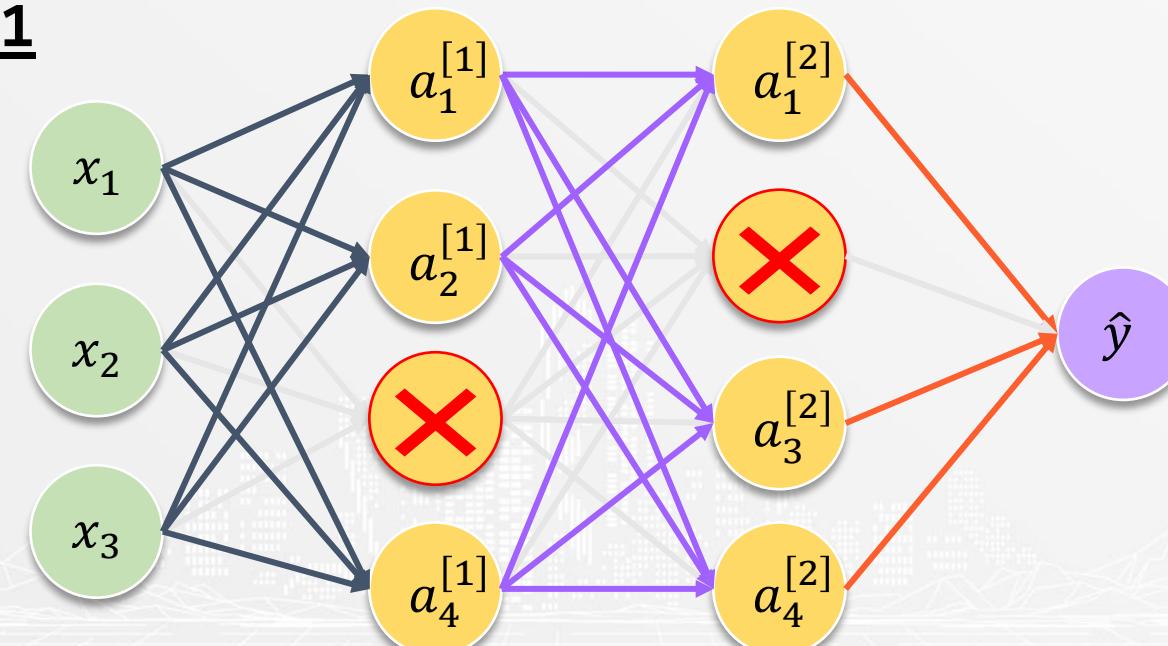
กำหนดให้ `drop_rate = 0.25`

Technical of Dropout

ຕົວຢ່າງ

2.1 Drop node แบบสุ่มໃນແຕ່ລະ hidden layer

epoch = 1



Technical of Dropout

ຕົວຢ່າງ

2.2 Feedforward

epoch = 1

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

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

$$a_3^{[1]} = \text{X}$$

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

Technical of Dropout

ຕົວຢ່າງ

2.2 Feedforward

epoch = 1

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

$$a_2^{[2]} = \text{X}$$

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

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

Technical of Dropout

ຕົວຢ່າງ

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

ຕົວຢ່າງ

2.3 Update parameter

epoch = 1

$$\begin{bmatrix} b_1^{[1]} & b_2^{[1]} & \color{red}{b_3^{[1]}} & b_4^{[1]} \\ w_{1,1}^{[1]} & w_{1,2}^{[1]} & \color{red}{w_{1,3}^{[1]}} & w_{1,4}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & \color{red}{w_{2,3}^{[1]}} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & \color{red}{w_{3,3}^{[1]}} & w_{3,4}^{[1]} \end{bmatrix}$$

Technical of Dropout

ຕົວຢ່າງ

2.3 Update parameter

epoch = 1

$$\begin{bmatrix} b_1^{[2]} & b_2^{[2]} & b_3^{[2]} & b_4^{[2]} \\ w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} & w_{1,4}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} & w_{2,4}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} & w_{3,4}^{[2]} \\ w_{4,1}^{[2]} & w_{4,2}^{[2]} & w_{4,3}^{[2]} & w_{4,4}^{[2]} \end{bmatrix}$$

Technical of Dropout

ຕົວຢ່າງ

2.3 Update parameter

epoch = 1

$$[b^{[out]}]$$

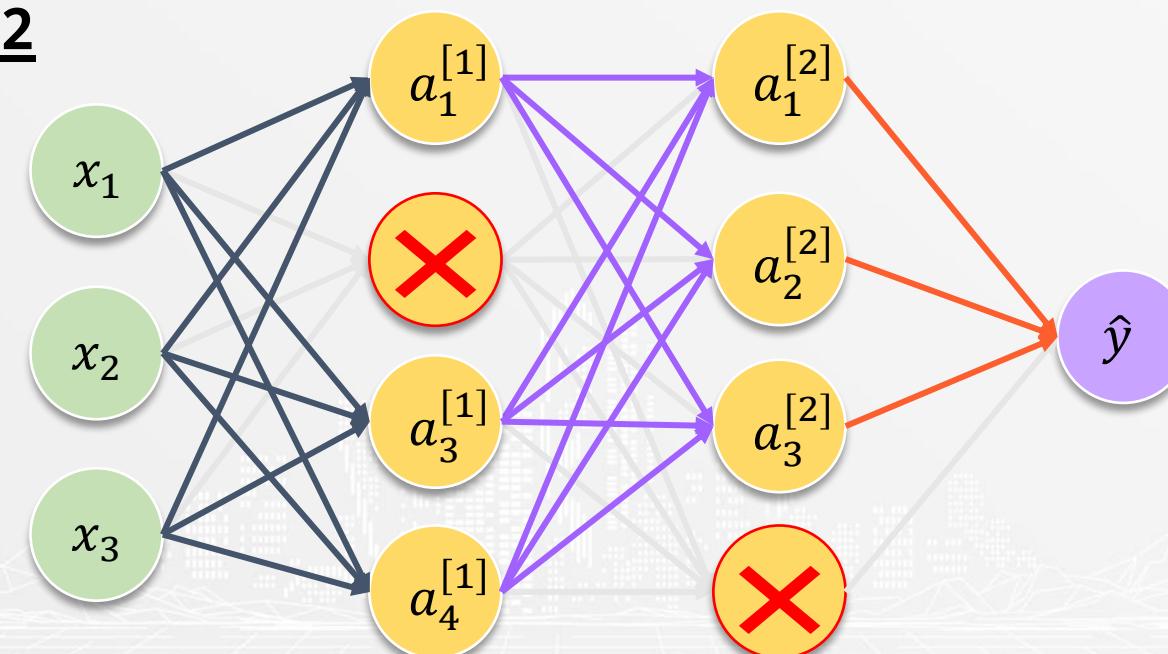
$$\begin{bmatrix} w_1^{[out]} \\ w_2^{[out]} \\ w_3^{[out]} \\ w_4^{[out]} \end{bmatrix}$$

Technical of Dropout

ຕົວຢ່າງ

2.1 Drop node แบบสุ่มໃນແຕ່ລະ hidden layer

epoch = 2



Technical of Dropout

ຕົວຢ່າງ

2.2 Feedforward

epoch = 2

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

$$a_2^{[1]} = \text{X}$$

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

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

Technical of Dropout

ຕົວຢ່າງ

2.2 Feedforward

epoch = 2

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

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

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

$$a_4^{[2]} = \text{X}$$

Technical of Dropout

ຕົວຢ່າງ

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

ຕົວຢ່າງ

2.3 Update parameter

epoch = 2

$$\begin{bmatrix} b_1^{[1]} & b_2^{[1]} & b_3^{[1]} & b_4^{[1]} \\ w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,3}^{[1]} & w_{1,4}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} & w_{3,4}^{[1]} \end{bmatrix}$$

Technical of Dropout

ຕົວຢ່າງ

2.3 Update parameter

epoch = 2

$$\begin{bmatrix} b_1^{[2]} & b_2^{[2]} & b_3^{[2]} & b_4^{[2]} \\ w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} & w_{1,4}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} & w_{2,4}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} & w_{3,4}^{[2]} \\ w_{4,1}^{[2]} & w_{4,2}^{[2]} & w_{4,3}^{[2]} & w_{4,4}^{[2]} \end{bmatrix}$$

Technical of Dropout

ຕົວຢ່າງ

2.3 Update parameter

epoch = 2

$$[b^{[out]}]$$

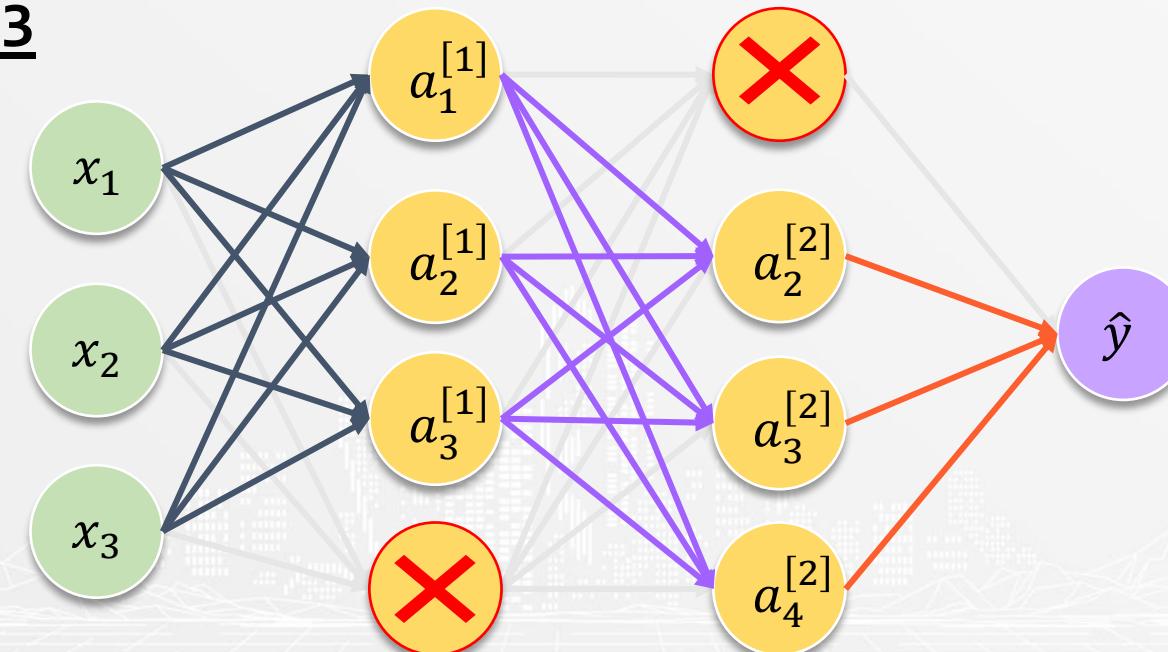
$$\begin{bmatrix} w_1^{[out]} \\ w_2^{[out]} \\ w_3^{[out]} \\ w_4^{[out]} \end{bmatrix}$$

Technical of Dropout

ຕົວຢ່າງ

2.1 Drop node แบบสุ่มໃນແຕ່ລະ hidden layer

epoch = 3



Technical of Dropout

ຕົວຢ່າງ

2.2 Feedforward

epoch = 3

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

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

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

$$a_4^{[1]} = \text{X}$$

Technical of Dropout

ຕົວຢ່າງ

2.2 Feedforward

epoch = 3

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

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

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

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

Technical of Dropout

ຕົວຢ່າງ

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

ຕົວຢ່າງ

2.3 Update parameter

epoch = 3

$$\begin{bmatrix} b_1^{[1]} & b_2^{[1]} & b_3^{[1]} & b_4^{[1]} \\ w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,3}^{[1]} & w_{1,4}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} & w_{2,4}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} & w_{3,4}^{[1]} \end{bmatrix}$$

Technical of Dropout

ຕົວຢ່າງ

2.3 Update parameter

epoch = 3

$$\begin{bmatrix} b_1^{[2]} & b_2^{[2]} & b_3^{[2]} & b_4^{[2]} \\ w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} & w_{1,4}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} & w_{2,4}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} & w_{3,4}^{[2]} \\ w_{4,1}^{[2]} & w_{4,2}^{[2]} & w_{4,3}^{[2]} & w_{4,4}^{[2]} \end{bmatrix}$$

Technical of Dropout

ຕົວຢ່າງ

2.3 Update parameter

epoch = 3

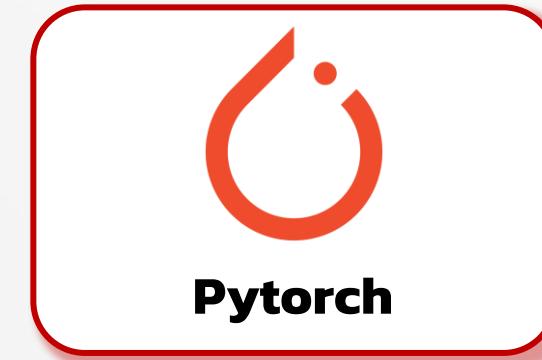
$$[b^{[out]}]$$

$$\begin{bmatrix} w_1^{[out]} \\ w_2^{[out]} \\ w_3^{[out]} \\ w_4^{[out]} \end{bmatrix}$$

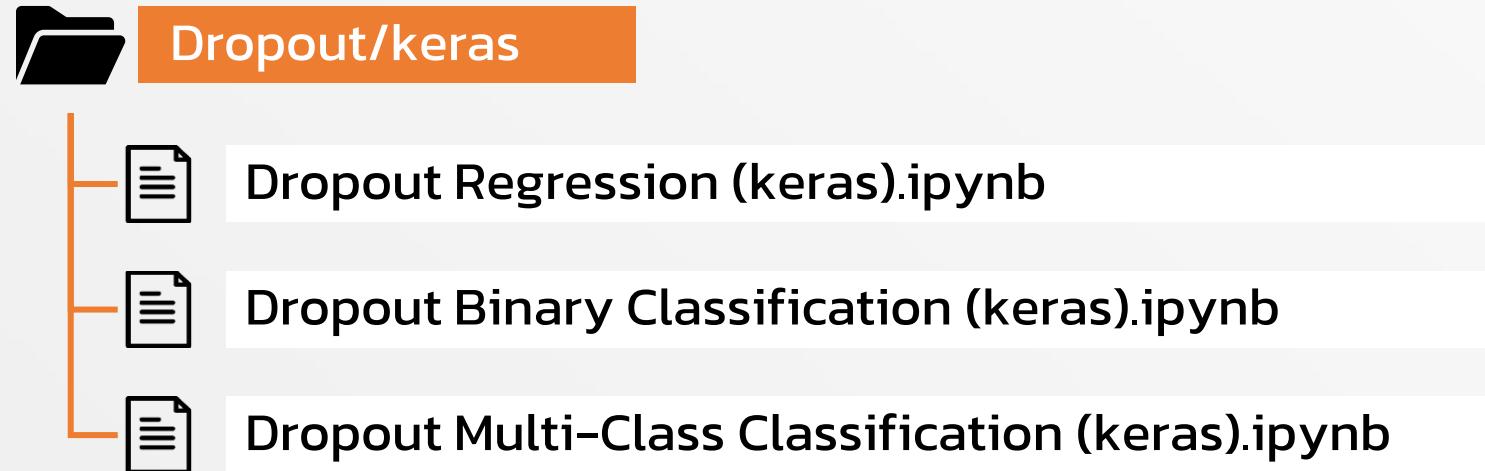
Dropout Regularization

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

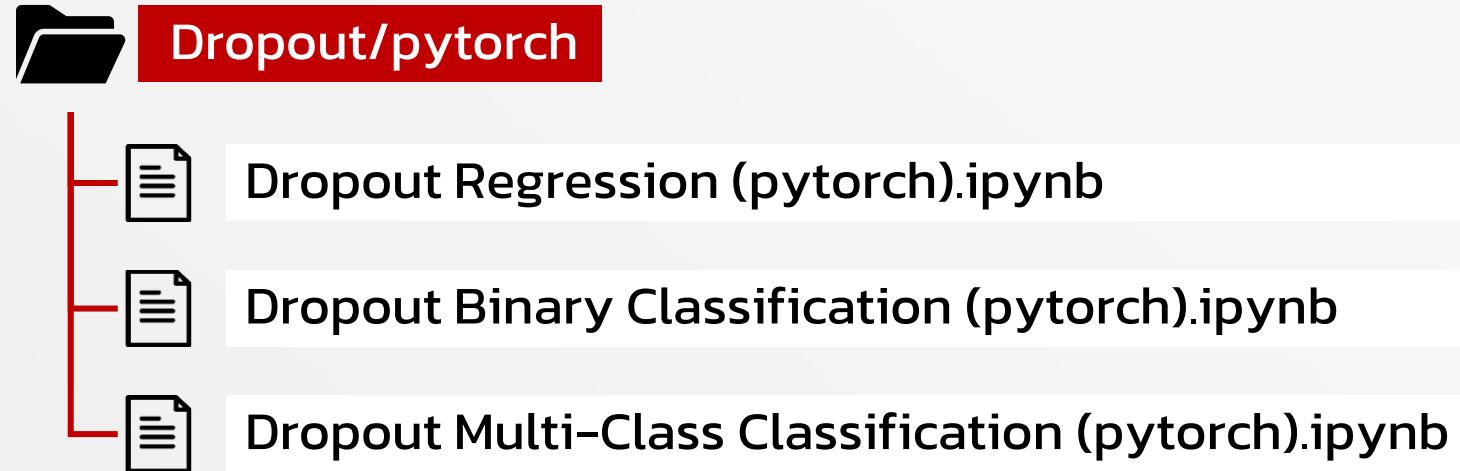
Code of Dropout



Code of Dropout



Code of Dropout



Dropout Regularization

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

Regularization

L2 Regularization



L1 Regularization



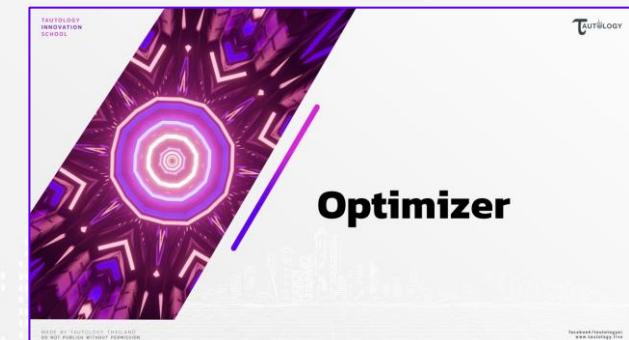
Elastic Net



**Dropout
Regularization**



Improvement of Deep Learning



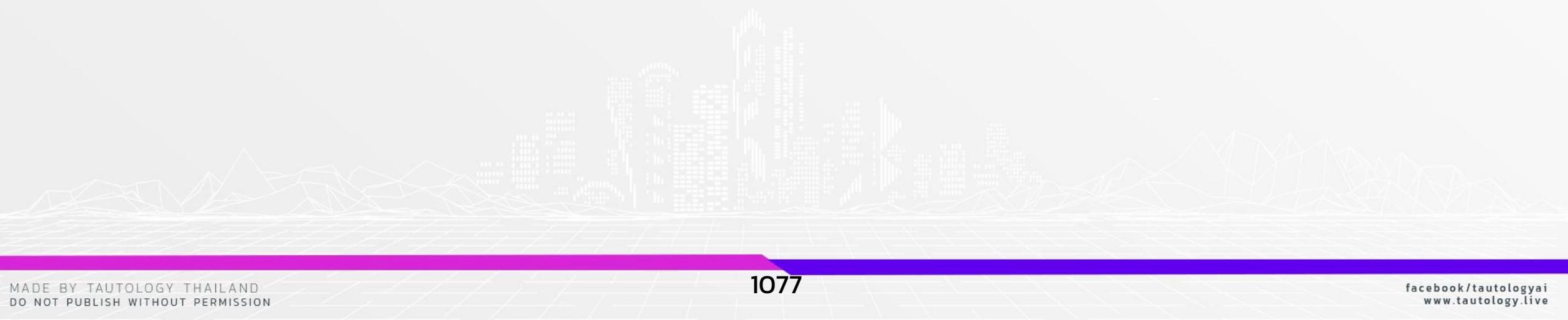
Optimizer

Optimizer

Gradient Descent
Variants

Optimization
Algorithm

Code

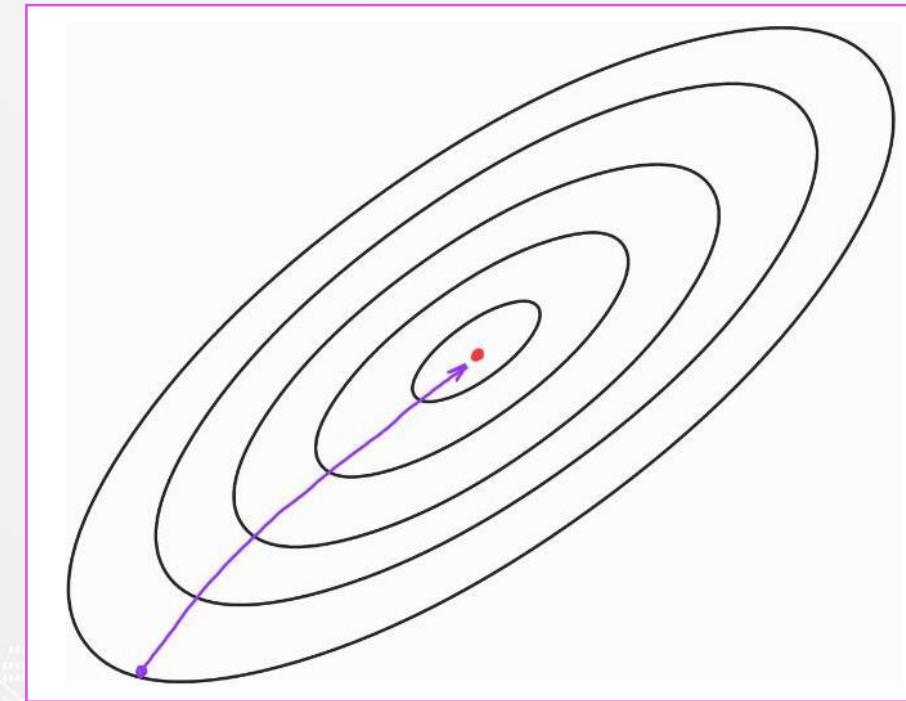


Gradient Descent Variants

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

Batch Gradient Descent

■ Batch Gradient Descent



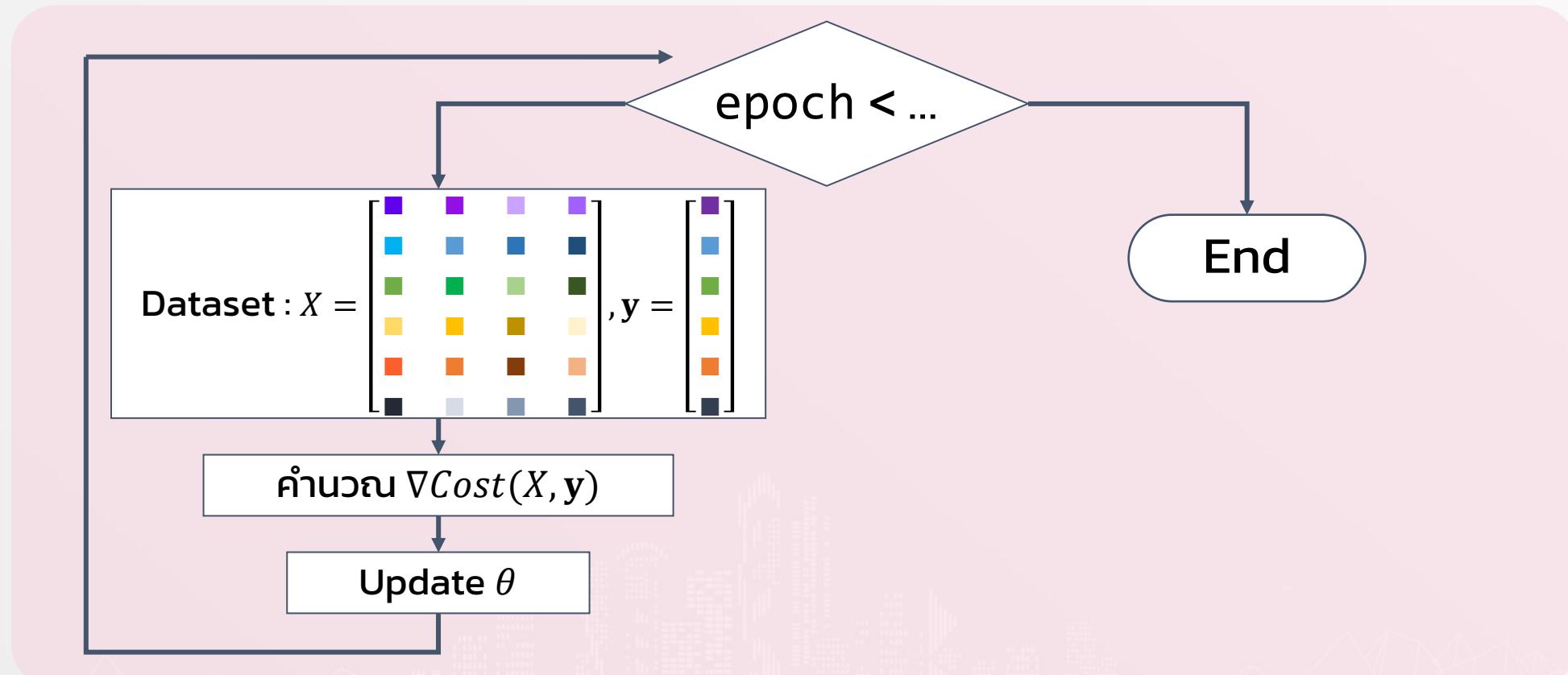
Batch Gradient Descent

■ 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

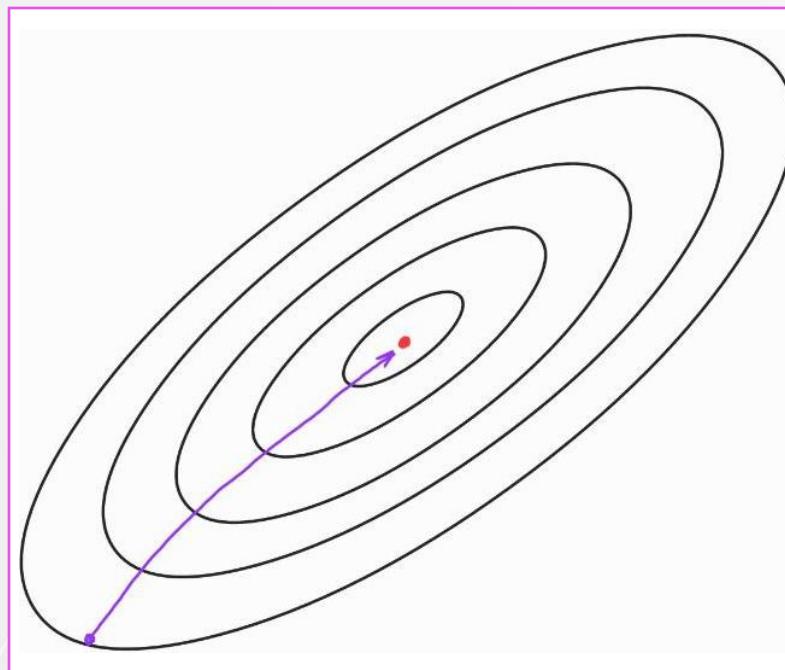
$$\theta_t = \theta_{t-1} - \frac{\alpha}{n} \nabla Cost_{t-1}$$

Batch Gradient Descent



Batch Gradient Descent

▪ Batch Gradient Descent



✓ ข้อดี ✓

> มีความ stable สูง

✗ ข้อเสีย ✗

> Computation cost สูง
> ช้า

Batch Gradient Descent

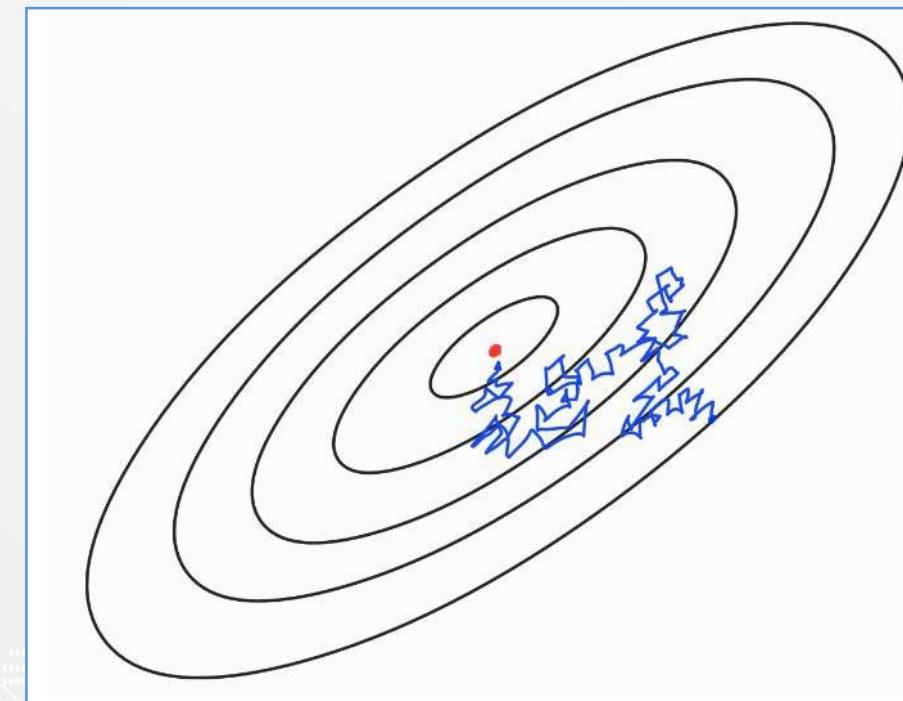
Gradient Descent Variants	ข้อดี	ข้อเสีย
Batch	<ul style="list-style-type: none">มีความ stable สูง	<ul style="list-style-type: none">Computational cost สูงช้า
Stochastic		
Minibatch		
Minibatch with learning rate		

Gradient Descent Variants

- Batch Gradient Descent**
- Stochastic Gradient Descent
- Minibatch Gradient Descent

Stochastic Gradient Descent

■ Stochastic Gradient Descent



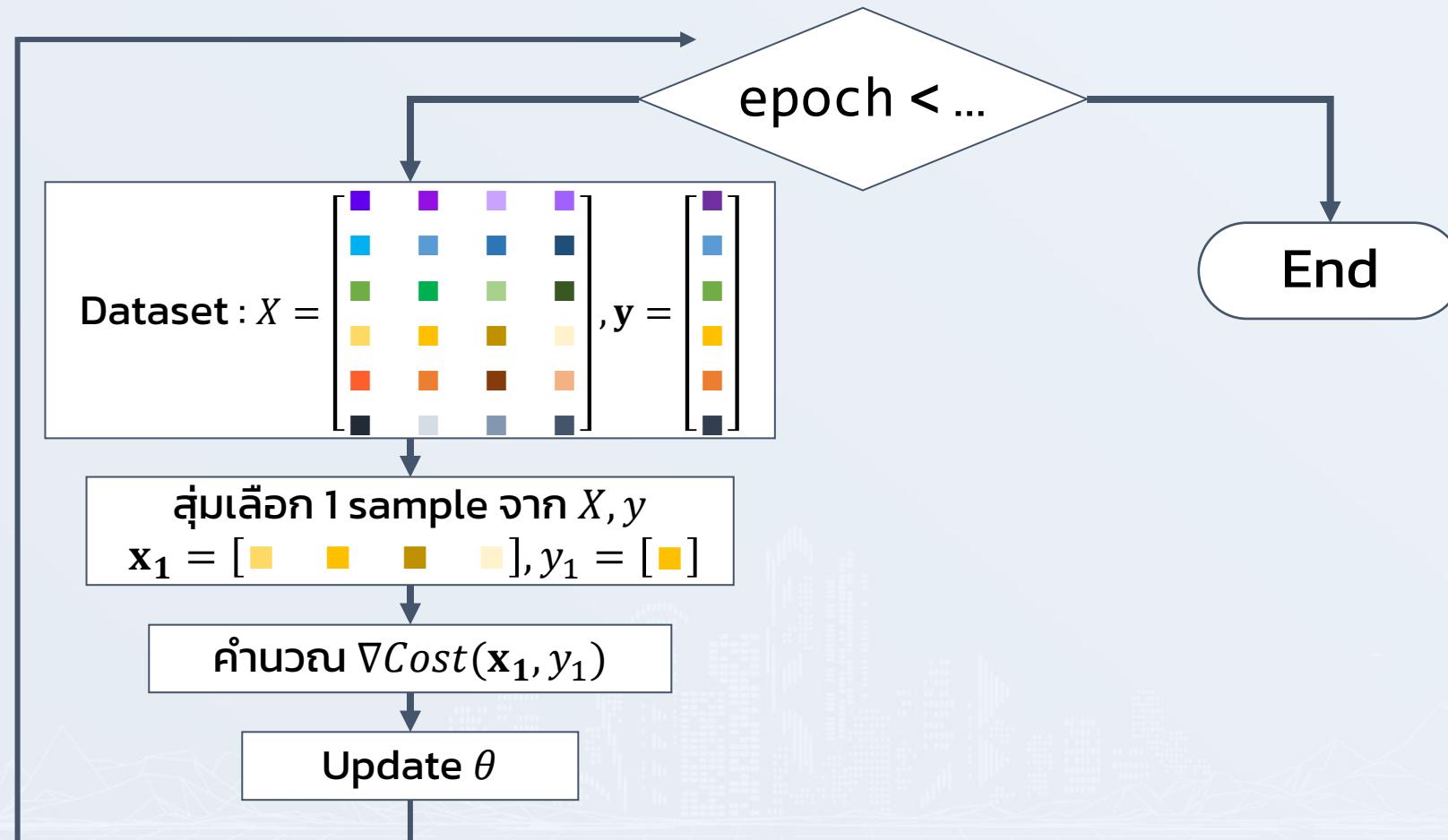
Stochastic Gradient Descent

■ 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

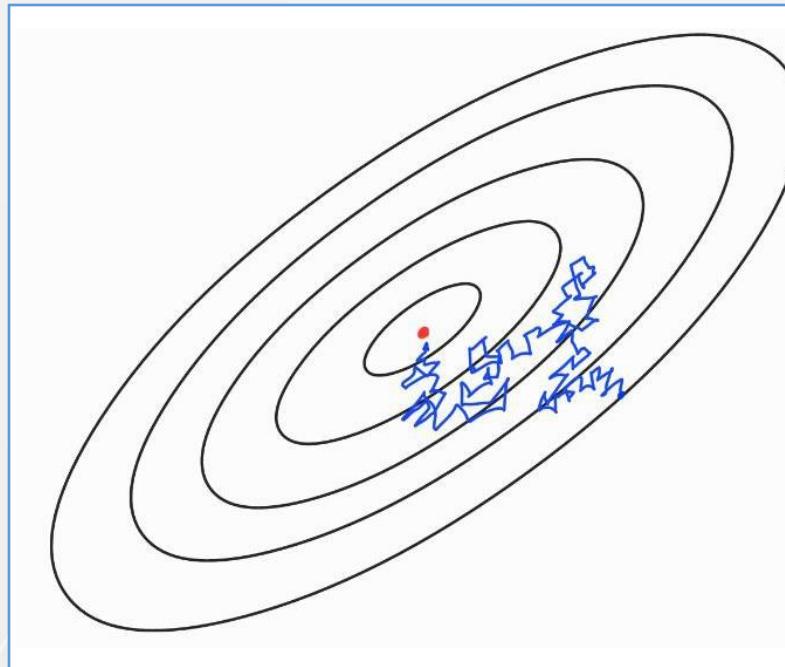
$$\theta_t = \theta_{t-1} - \alpha \nabla Cost_{t-1}$$

Stochastic Gradient Descent



Stochastic Gradient Descent

■ Stochastic Gradient Descent



✓ ข้อดี ✓

- > Computation cost ต่ำ
- > เร็ว

✗ ข้อเสีย ✗

- > มีความ stable ต่ำ

Stochastic Gradient Descent

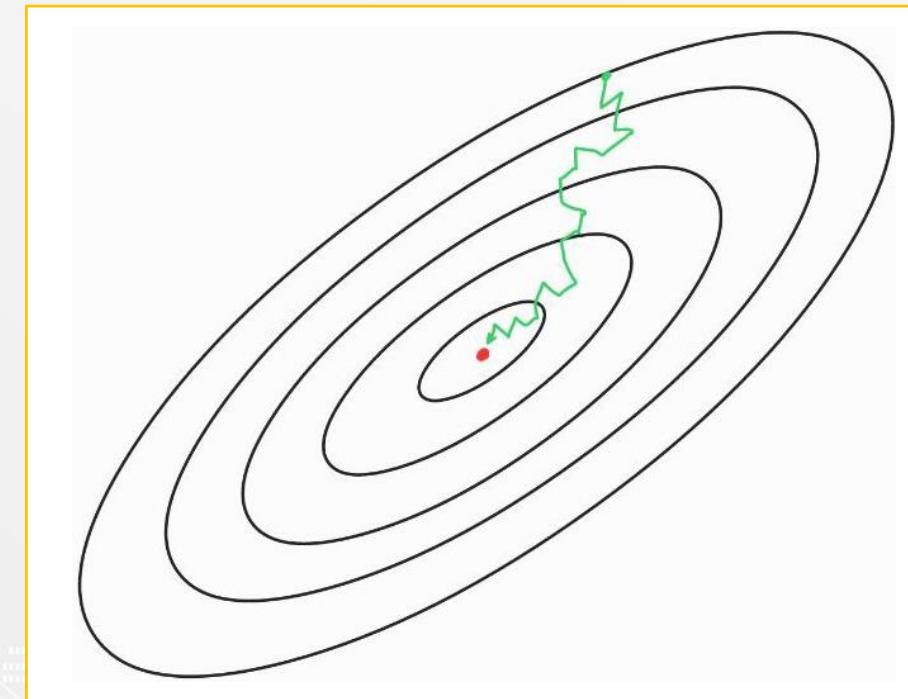
Gradient Descent Variants	ข้อดี	ข้อเสีย
Batch	<ul style="list-style-type: none">มีความ stable สูง	<ul style="list-style-type: none">Computational cost สูงช้า
Stochastic	<ul style="list-style-type: none">Computational cost ต่ำเร็ว	<ul style="list-style-type: none">มีความ stable ต่ำ
Minibatch		
Minibatch with learning rate		

Gradient Descent Variants

- Batch Gradient Descent**
- Stochastic Gradient Descent**
- Minibatch Gradient Descent

Minibatch Gradient Descent

■ Minibatch Gradient Descent



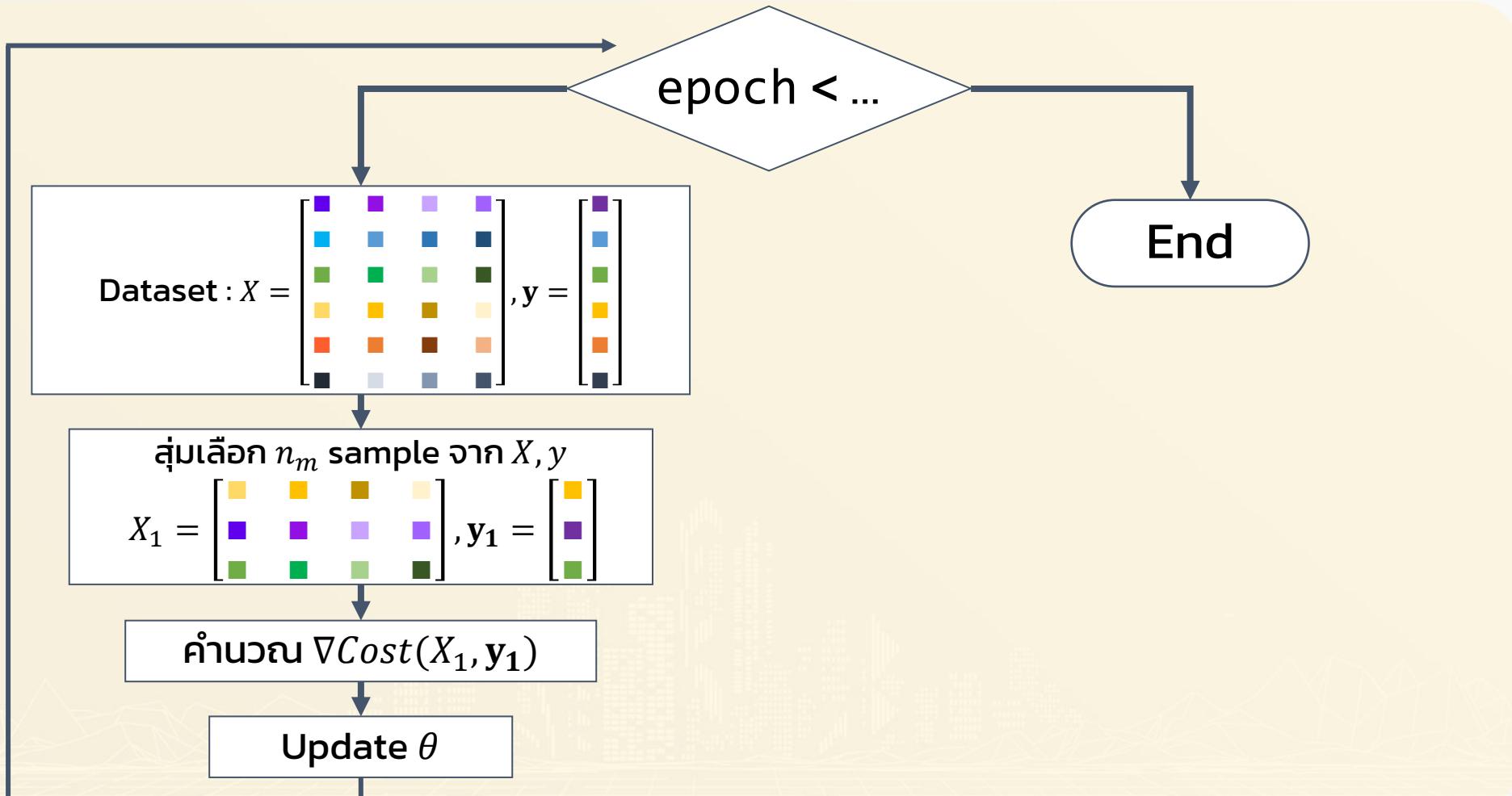
Minibatch Gradient Descent

■ 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

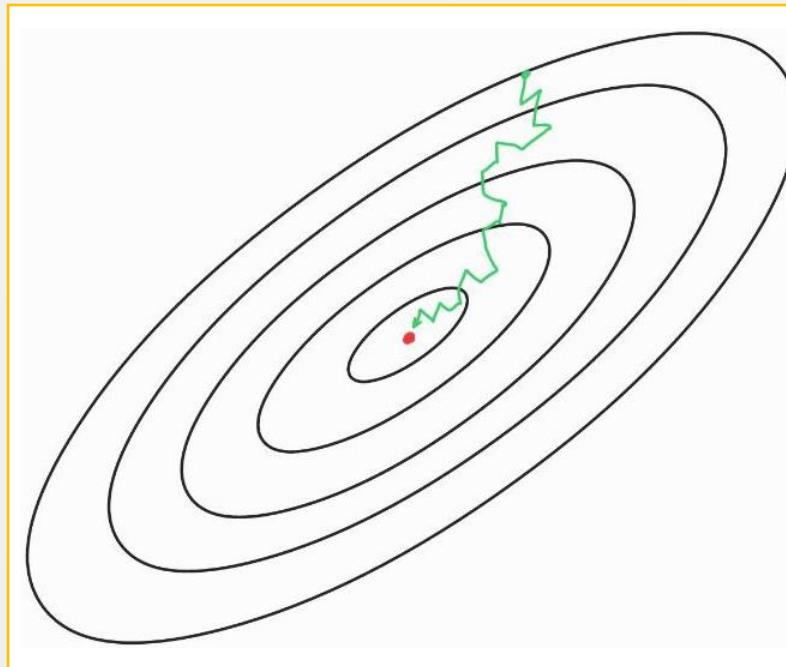


Minibatch Gradient Descent

เราต้องการให้แต่ละ sample ถูกนำมาใช้ train
model ในจำนวนครั้งที่เท่ากัน

Minibatch Gradient Descent

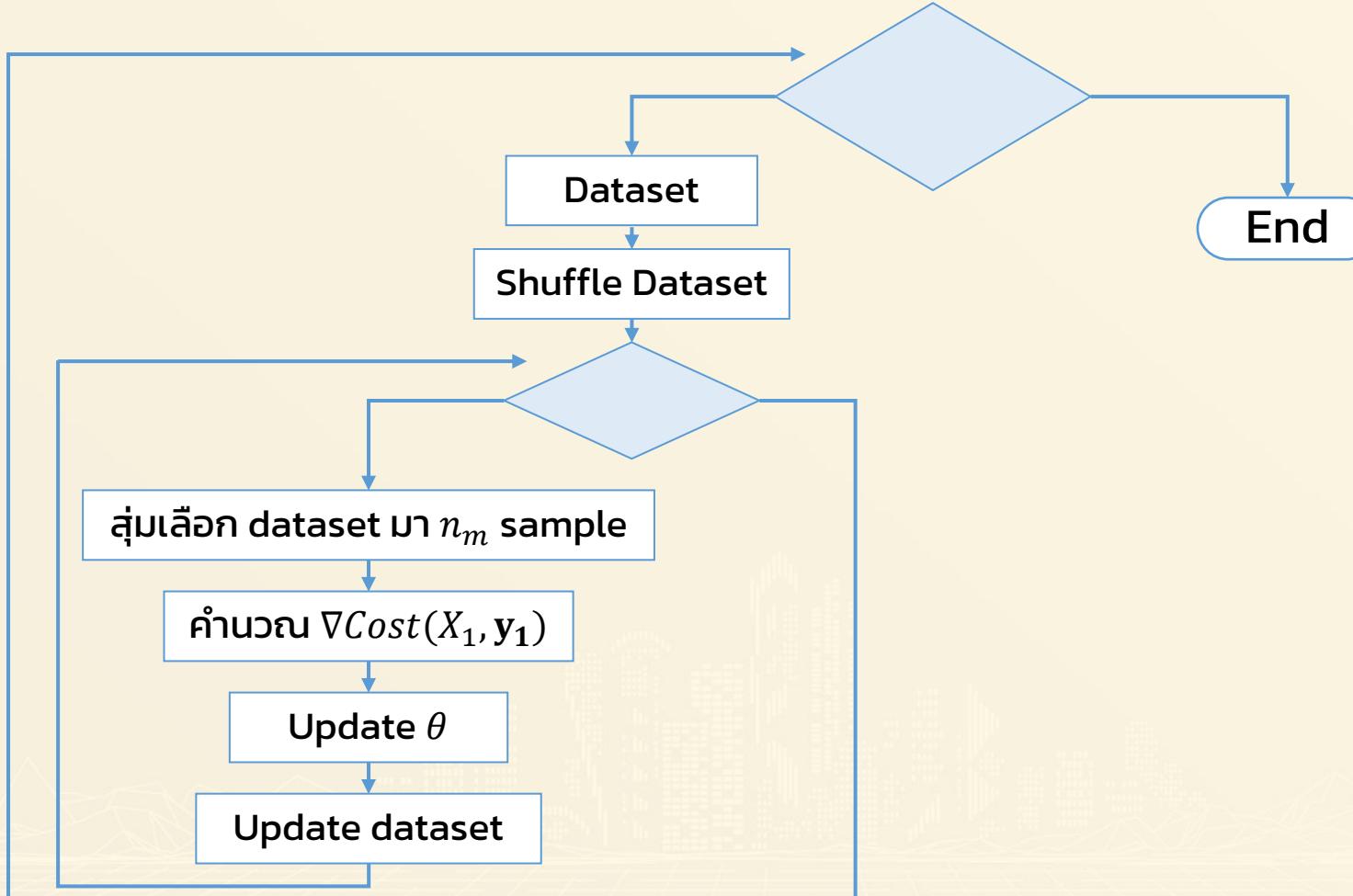
■ Minibatch Gradient Descent (Nowadays)



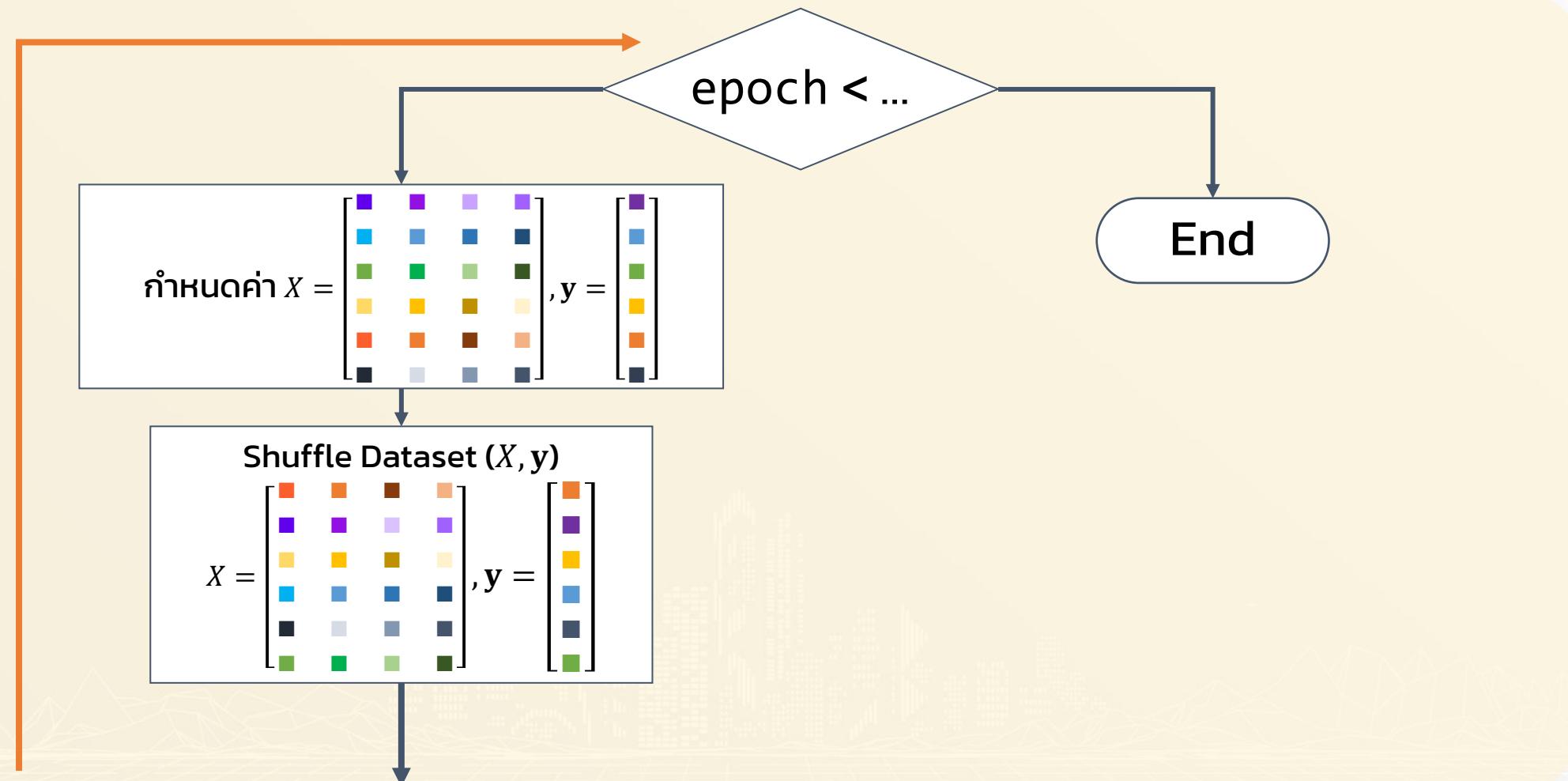
1. กำหนดค่า θ เริ่มต้น
2. กำหนดจำนวนรอบที่จะ update θ (epoch)
3. กำหนดค่า α
4. For loop เพื่อ Update θ
 - 4.1 Shuffle dataset
 - 4.2 for batch in get_batches(dataset, batch_size) :
 - คำนวณ $\nabla Cost$ ของ batch
 - Update θ

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

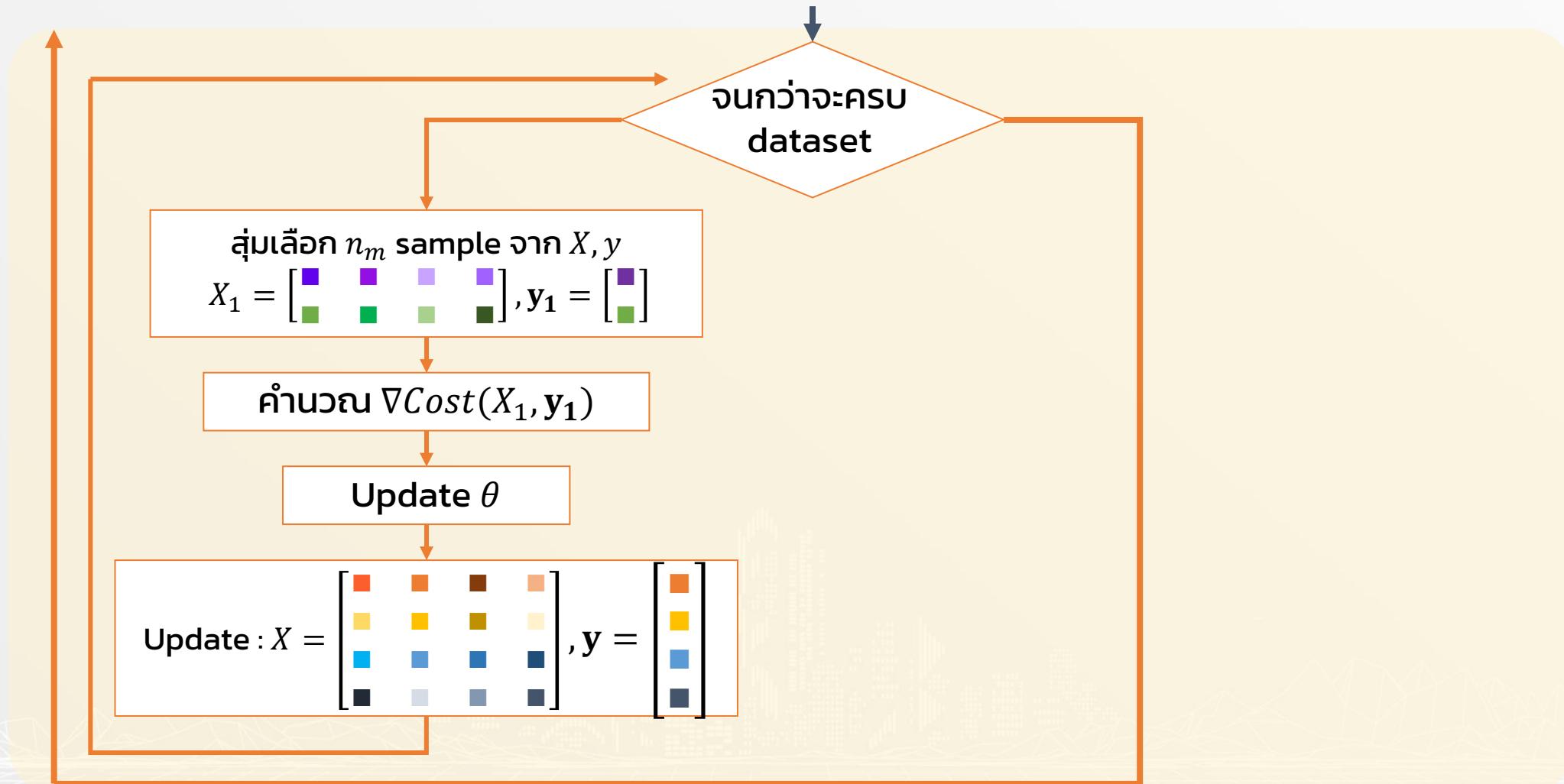
Minibatch Gradient Descent



Minibatch Gradient Descent

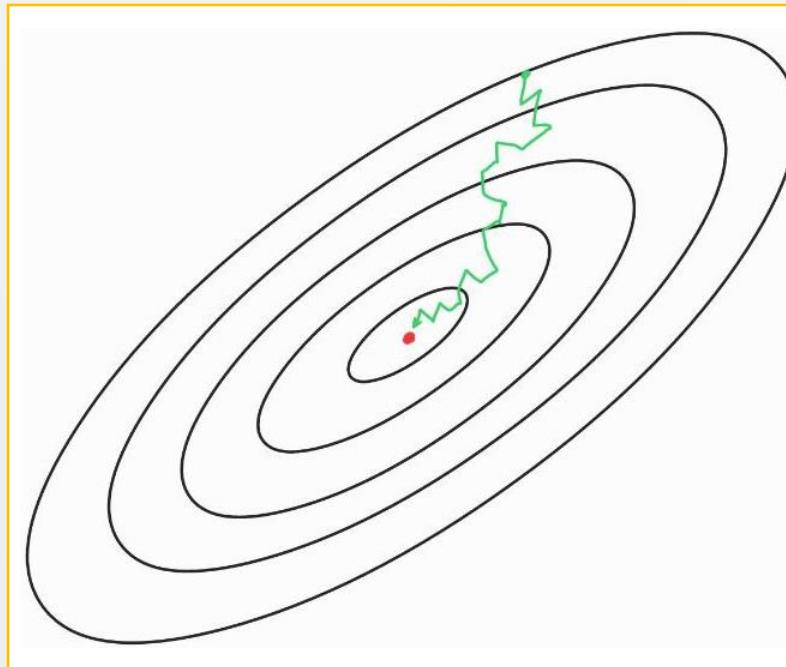


Minibatch Gradient Descent



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

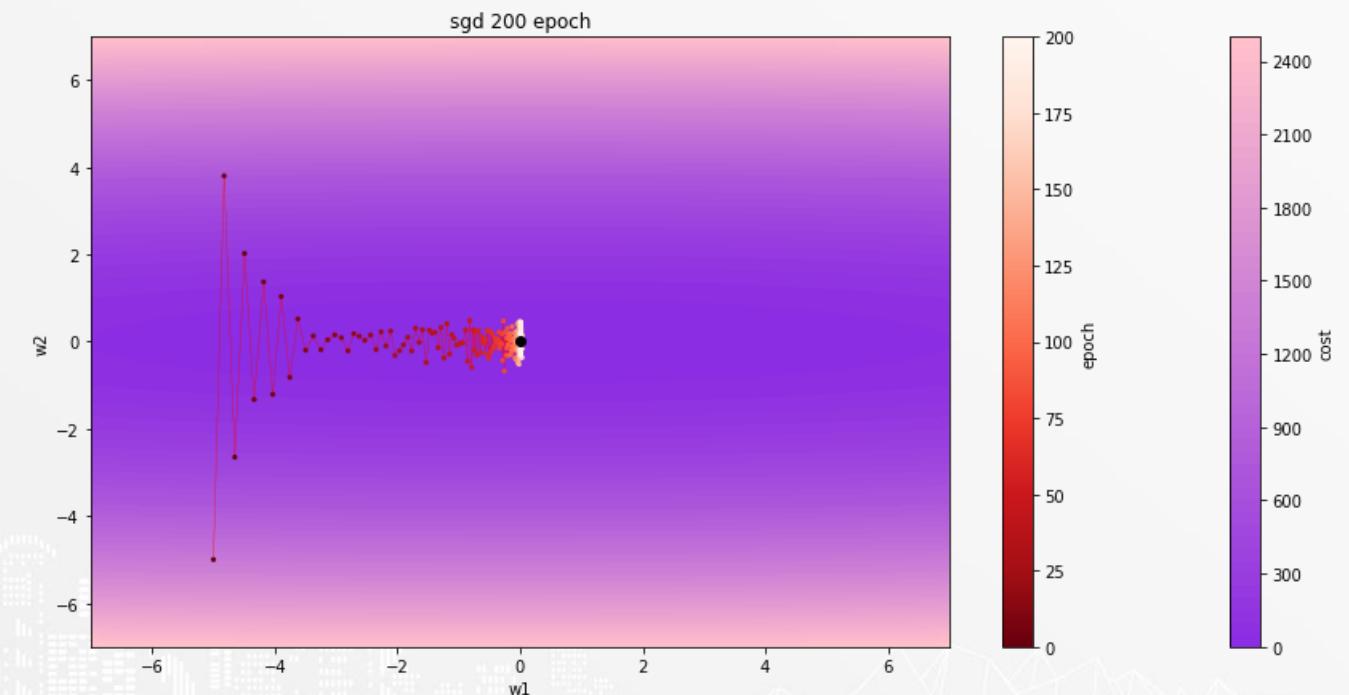
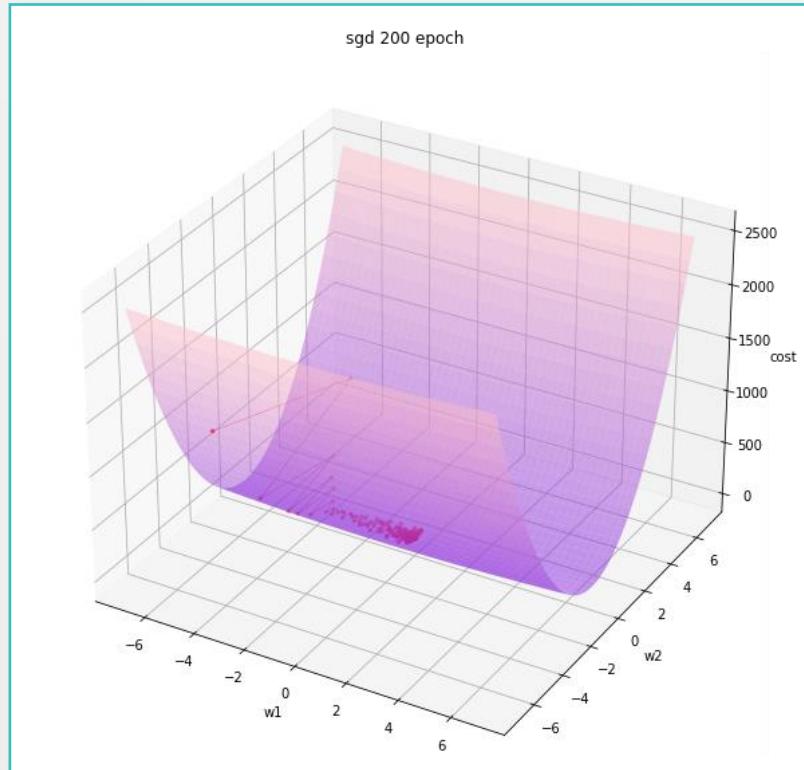
Minibatch Gradient Descent

Gradient Descent Variants	ข้อดี	ข้อเสีย
Batch	<ul style="list-style-type: none">มีความ stable สูง	<ul style="list-style-type: none">Computational cost สูงช้า
Stochastic	<ul style="list-style-type: none">Computational cost ต่ำเร็ว	<ul style="list-style-type: none">มีความ stable ต่ำ
Minibatch	<ul style="list-style-type: none">Computational cost ต่ำกว่า Batchมีความ stable สูงกว่า Stochastic	<ul style="list-style-type: none">Computational cost สูงกว่า Stochasticมีความ stable ต่ำกว่า Batch

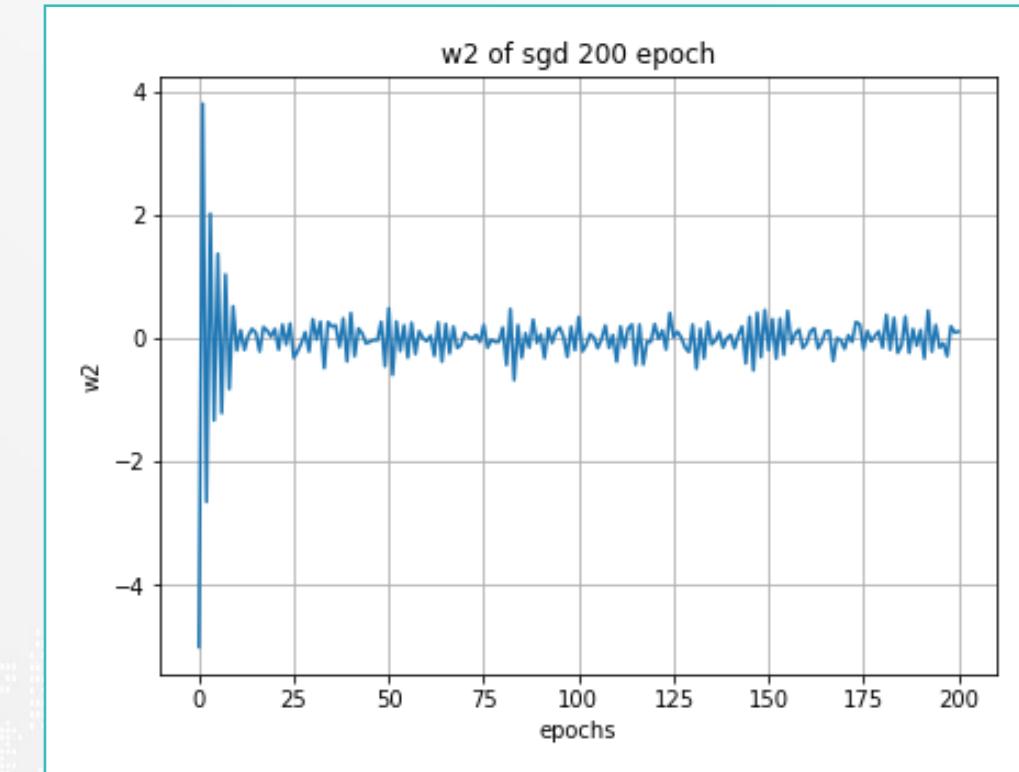
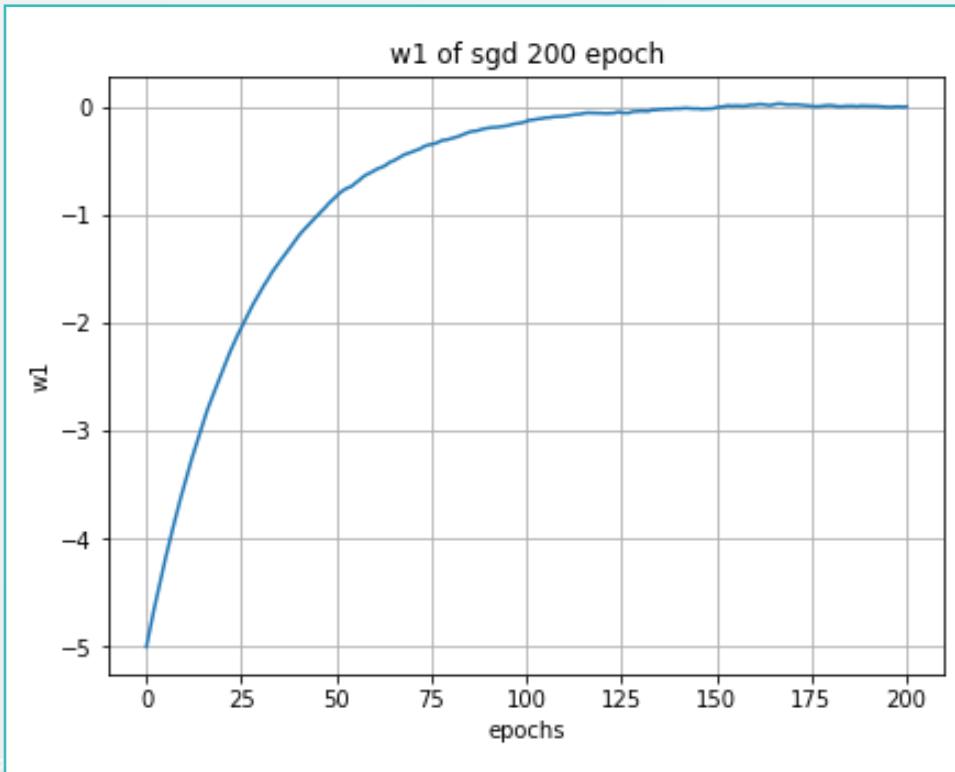
Minibatch Gradient Descent

แต่การใช้ minibatch ก็ยังคงเกิดปัญหาเรื่อง
ความ stable เมื่อเข้าใกล้จุดต่ำสุดอยู่ดี

Minibatch Gradient Descent

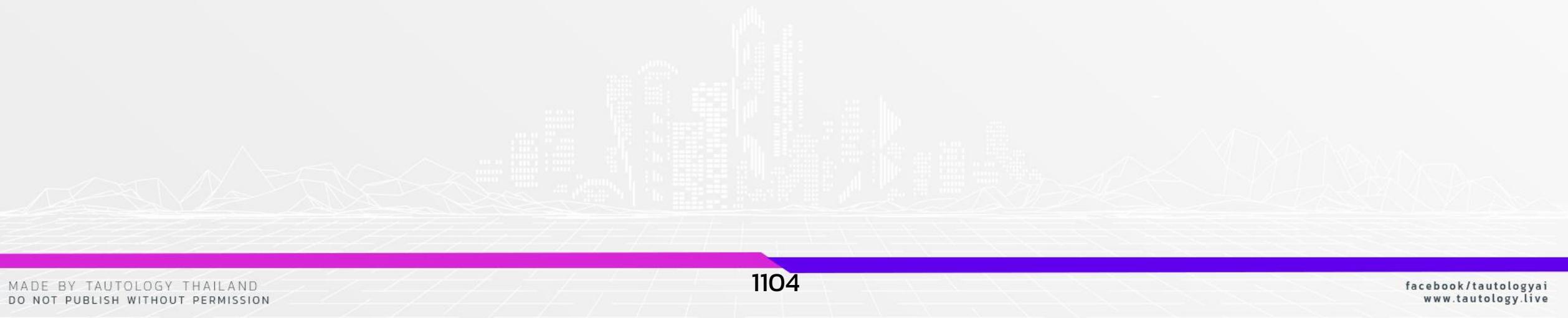


Minibatch Gradient Descent



Gradient Descent Variants

- Batch Gradient Descent**
- Stochastic Gradient Descent**
- Minibatch Gradient Descent**



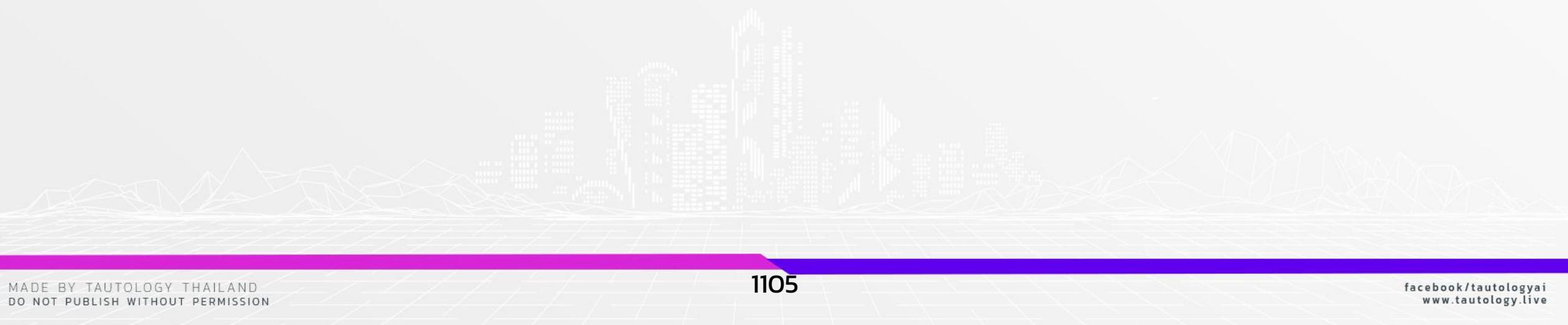
Optimizer

Gradient Descent
Variants



Optimization
Algorithm

Code

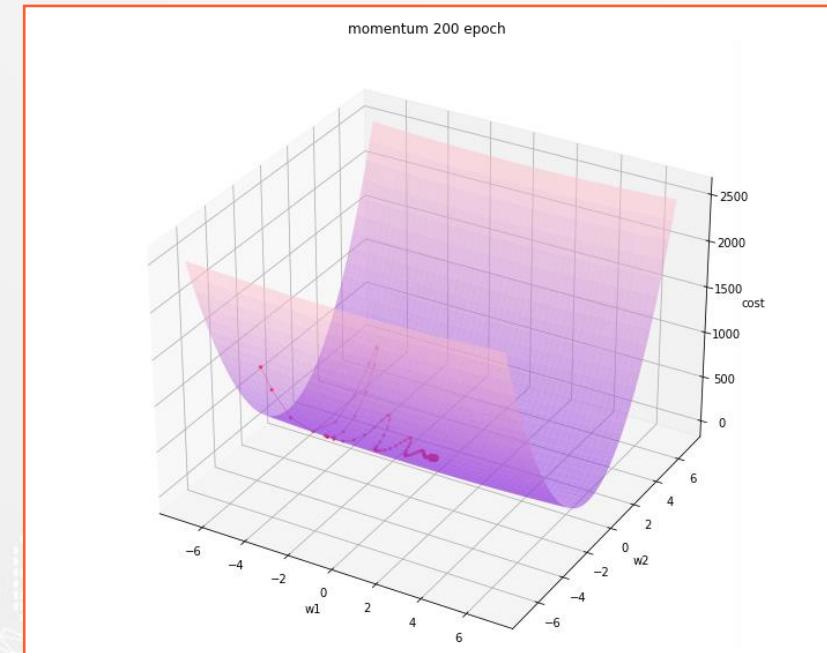


Optimization Algorithm

- 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)
 - ◆ $v_0 = 0$

Momentum Base

■ ขั้นตอนการคำนวณ

1. กำหนดค่า θ เริ่มต้น
2. กำหนดจำนวนรอบที่จะ update θ (epoch)
3. กำหนดค่า α
4. For loop เพื่อ Update θ
 - 4.1 คำนวณ g
 - 4.2 คำนวณ v
 - 4.3 คำนวณ θ

Momentum Base

1. กำหนด $\theta_0 = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$
2. กำหนด epoch = 200
3. กำหนดค่า $\alpha = 0.0015$
4. Update θ : epoch = 1

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

$$4.2 \quad v_1 = \begin{bmatrix} 0.9 \times 0 + 0.0015 \times (-10.04) \\ 0.9 \times 0 + 0.0015 \times (-495.21) \end{bmatrix} = \begin{bmatrix} -0.02 \\ -0.74 \end{bmatrix}$$

$$4.3 \quad \theta_1 = \begin{bmatrix} -5 - (-0.02) \\ -5 - (-0.74) \end{bmatrix} = \begin{bmatrix} -4.98 \\ -4.26 \end{bmatrix}$$

Momentum Base

4. Update θ : epoch = 2

$$4.1 \quad \mathbf{g}_2 = \begin{bmatrix} -10.07 \\ -431.28 \end{bmatrix}$$

$$4.2 \quad \mathbf{v}_2 = \begin{bmatrix} 0.9 \times (-0.02) + 0.0015 \times (-10.07) \\ 0.9 \times (-0.74) + 0.0015 \times (-431.28) \end{bmatrix} = \begin{bmatrix} -0.03 \\ -1.32 \end{bmatrix}$$

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

Momentum Base

4. Update θ : epoch = 3

$$4.1 \mathbf{g}_3 = \begin{bmatrix} -9.52 \\ -280.24 \end{bmatrix}$$

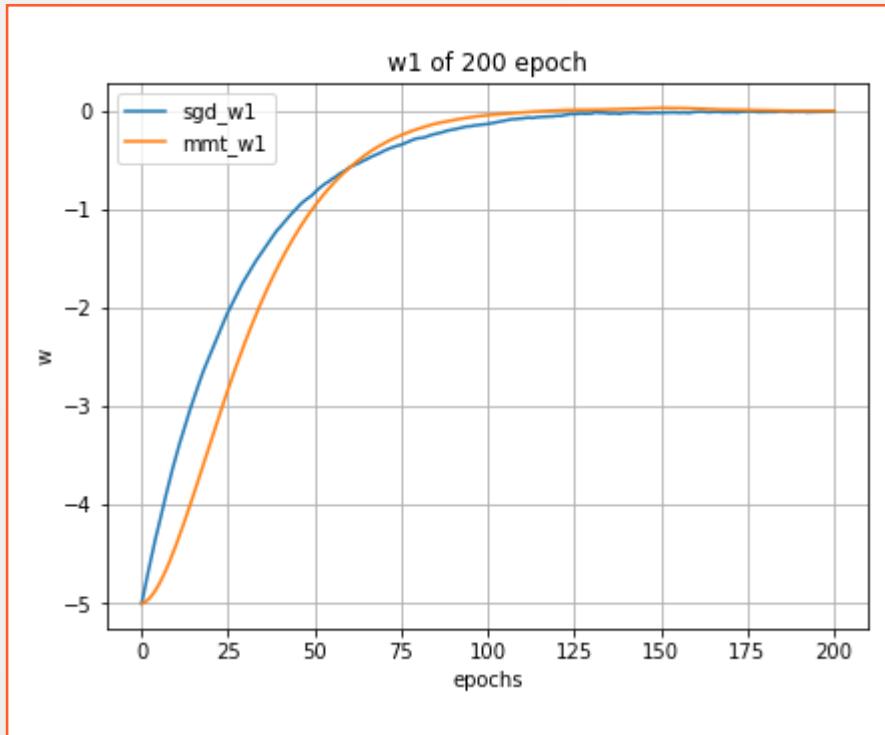
$$4.2 \mathbf{v}_3 = \begin{bmatrix} 0.9 \times (-0.03) + 0.0015 \times (-9.52) \\ 0.9 \times (-1.32) + 0.0015 \times (-280.24) \end{bmatrix} = \begin{bmatrix} -0.04 \\ -1.60 \end{bmatrix}$$

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

Momentum Base

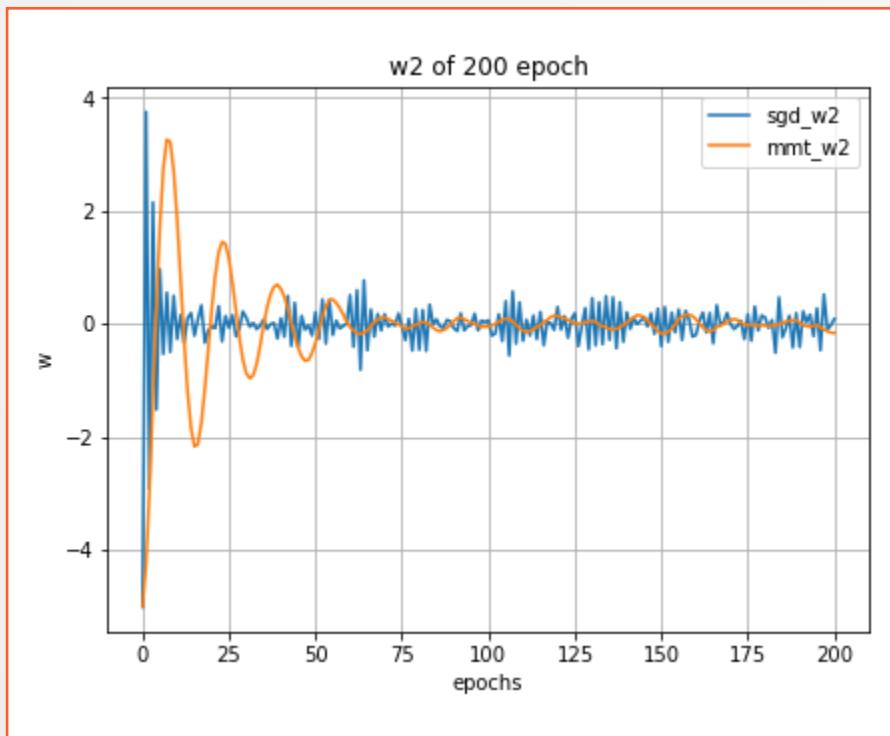
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.98	-4.26
3	-4.96	-2.94
4	-4.87	0.3
5	-4.81	1.71
6	-4.74	2.75
:	:	:
198	-0.01	-0.13
199	-0.01	-0.15
200	-0.01	-0.16

Momentum Base



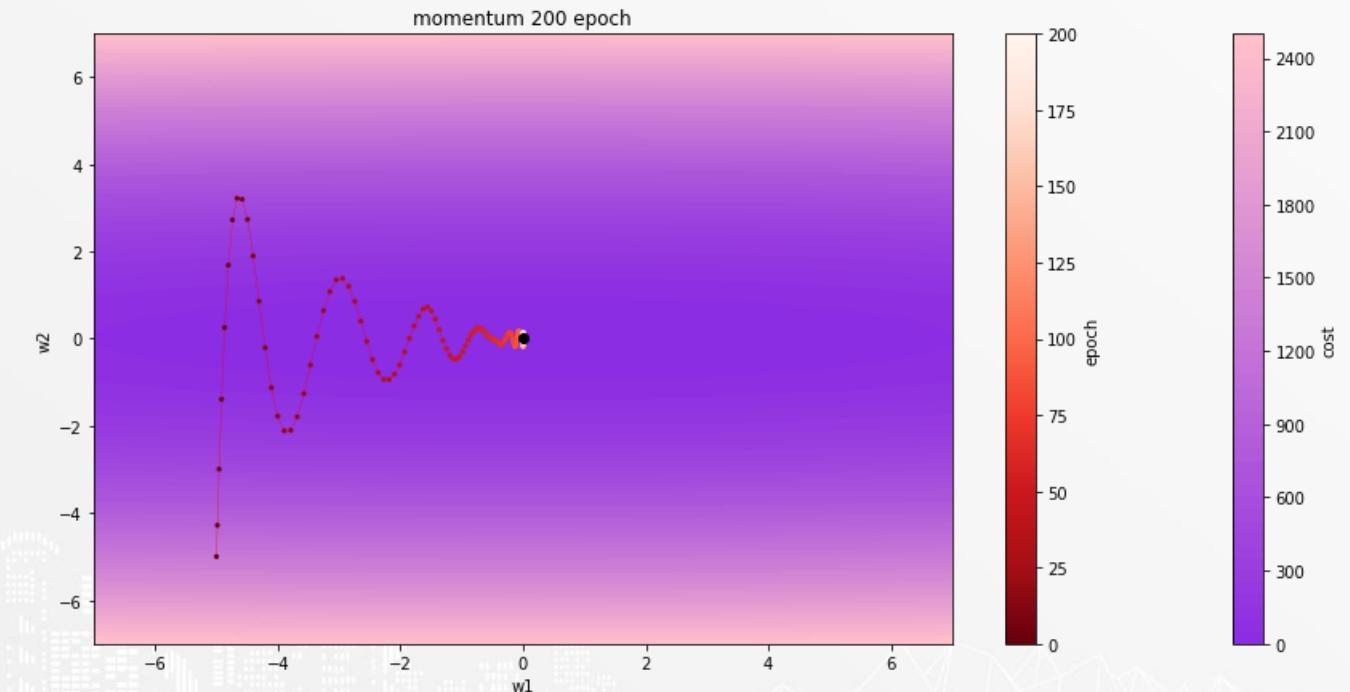
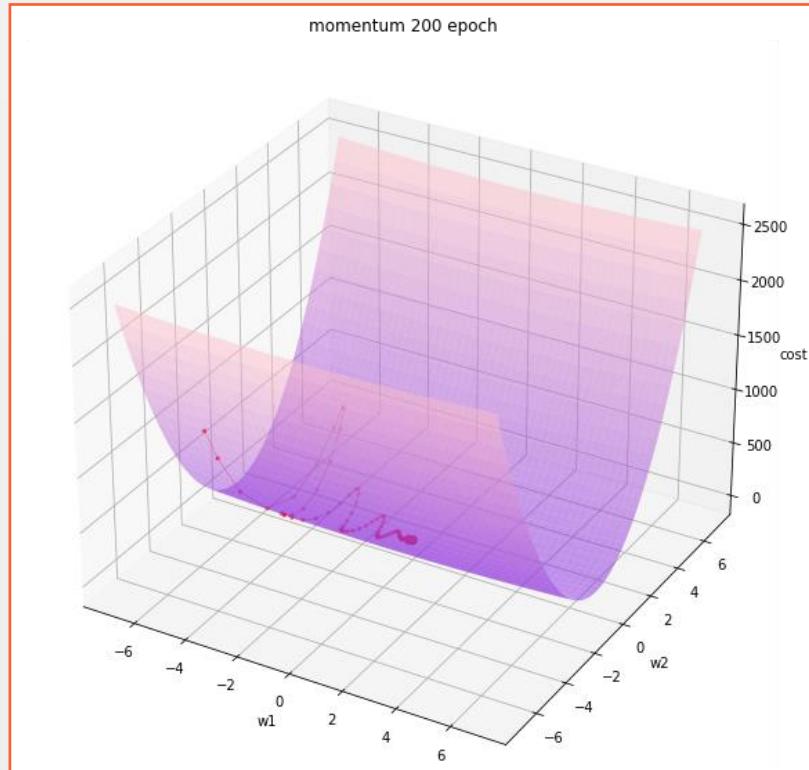
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.98	-4.26
3	-4.96	-2.94
4	-4.87	0.3
5	-4.81	1.71
6	-4.74	2.75
⋮	⋮	⋮
198	-0.01	-0.13
199	-0.01	-0.15
200	-0.01	-0.16

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



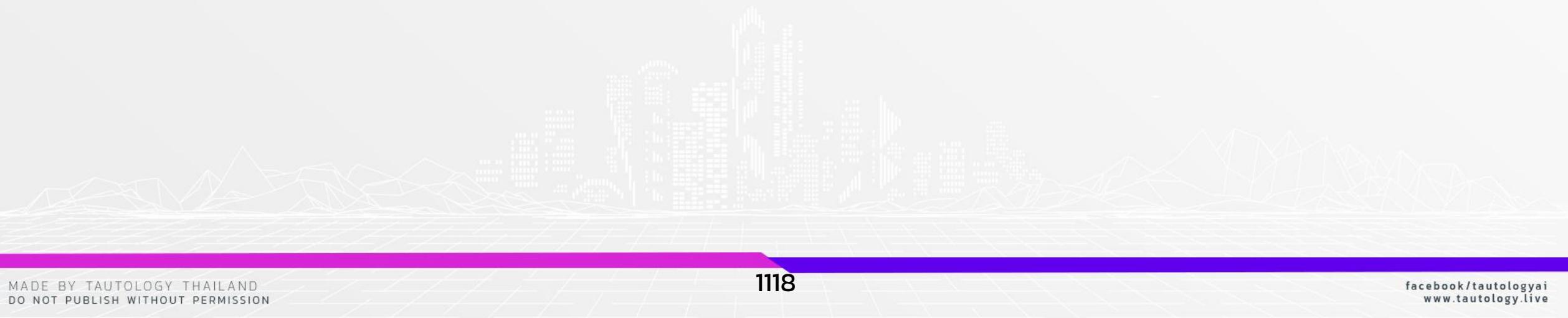
Optimization Algorithm

Momentum Base

- Adaptive Learning Rate Base Algorithm
- Adam
- Conclusion

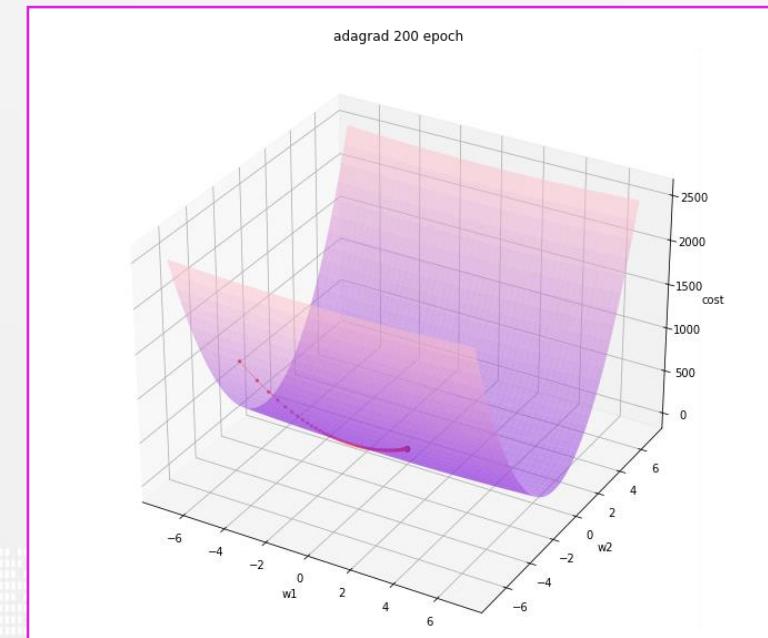
Adaptive Learning Rate Base Algorithm

- Adagrad
- RMSprop



Adagrad

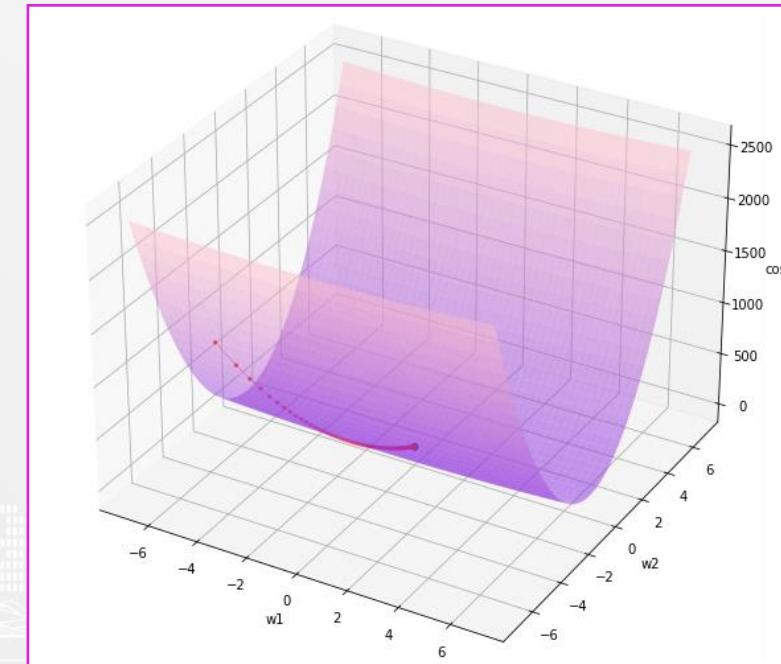
Adagrad (Adaptive gradient algorithm) คือ optimizer ที่จะถูกพิจารณาในรูปแบบของการสะสม gradient แล้วยังมีการปรับ learning rate ที่ไม่เท่ากันในแต่ละมิติ



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 g
 - 4.2 คำนวณ G
 - 4.2 คำนวณ S
 - 4.3 คำนวณ θ

Adagrad

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

Adagrad

4. Update θ : epoch = 1

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

$$4.2 \mathbf{G}_1 = \begin{bmatrix} 0 + (-10.04)^2 \\ 0 + (-495.21)^2 \end{bmatrix} = \begin{bmatrix} 100.82 \\ 245233.51 \end{bmatrix}$$

$$4.3 \mathbf{S}_1 = \begin{bmatrix} \frac{0.5}{\sqrt{100.82 + (1 \times 10^{-8})}} \\ \frac{0.5}{\sqrt{245233.51 + (1 \times 10^{-8})}} \end{bmatrix} = \begin{bmatrix} \frac{0.5}{10.04} \\ \frac{0.5}{495.21} \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.001 \end{bmatrix}$$

$$4.4 \mathbf{\theta}_1 = \begin{bmatrix} -5 - (0.05 \times -10.04) \\ -5 - (0.001 \times -495.21) \end{bmatrix} = \begin{bmatrix} -4.5 \\ -4.5 \end{bmatrix}$$

Adagrad

4. Update θ : epoch = 2

$$4.1 \mathbf{g}_2 = \begin{bmatrix} -9.1 \\ -455.56 \end{bmatrix}$$

$$4.2 \mathbf{G}_2 = \begin{bmatrix} 100.82 + (-9.1)^2 \\ 245233.51 + (-455.56)^2 \end{bmatrix} = \begin{bmatrix} 183.7 \\ 452765.96 \end{bmatrix}$$

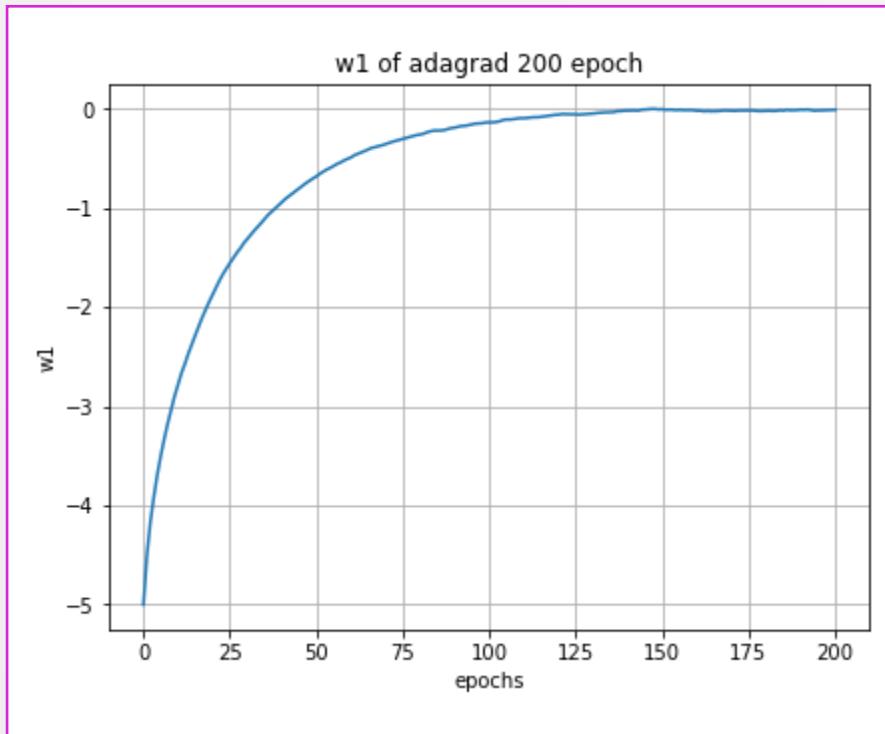
$$4.3 \mathbf{S}_2 = \begin{bmatrix} \frac{0.5}{\sqrt{183.7 + (1 \times 10^{-8})}} \\ \frac{0.5}{\sqrt{452765.96 + (1 \times 10^{-8})}} \end{bmatrix} = \begin{bmatrix} \frac{0.5}{13.55} \\ \frac{0.5}{672.88} \end{bmatrix} = \begin{bmatrix} 0.04 \\ 0.0007 \end{bmatrix}$$

$$4.4 \mathbf{\theta}_2 = \begin{bmatrix} -4.5 - (0.04 \times -9.1) \\ -4.5 - (0.0007 \times -455.56) \end{bmatrix} = \begin{bmatrix} -4.16 \\ -4.16 \end{bmatrix}$$

Adagrad

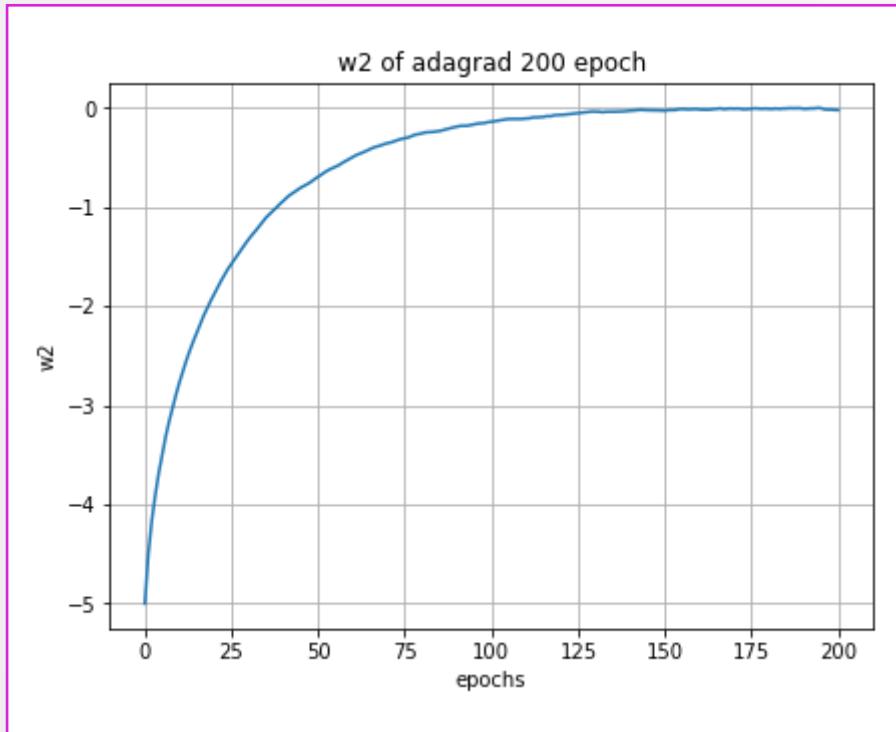
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.5	-4.5
3	-4.16	-4.16
4	-3.91	-3.9
5	-3.69	-3.68
6	-3.5	-3.49
:	:	:
198	-0.01	-0.02
199	-0.01	-0.02
200	-0.01	-0.02

Adagrad



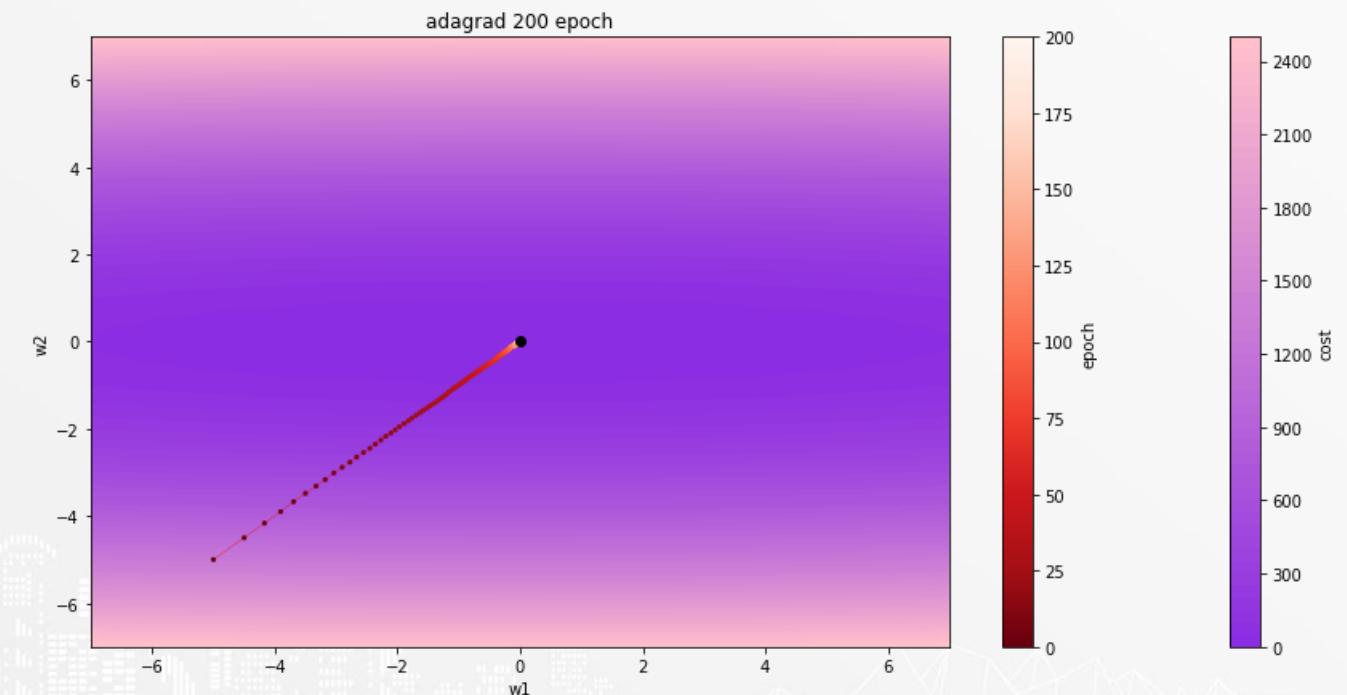
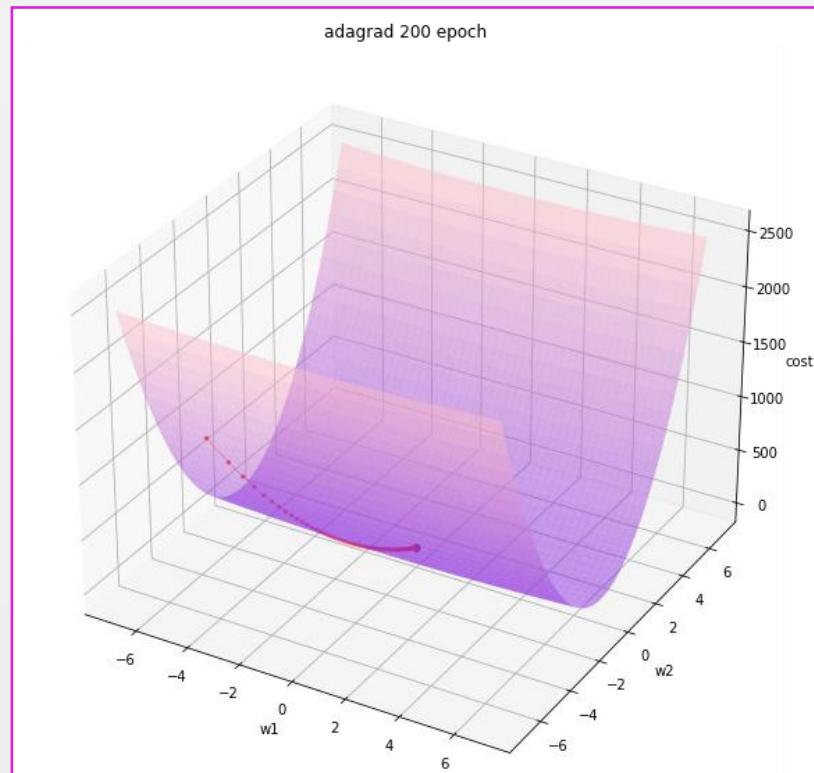
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.5	-4.5
3	-4.16	-4.16
4	-3.91	-3.9
5	-3.69	-3.68
6	-3.5	-3.49
:	:	:
198	-0.01	-0.02
199	-0.01	-0.02
200	-0.01	-0.02

Adagrad



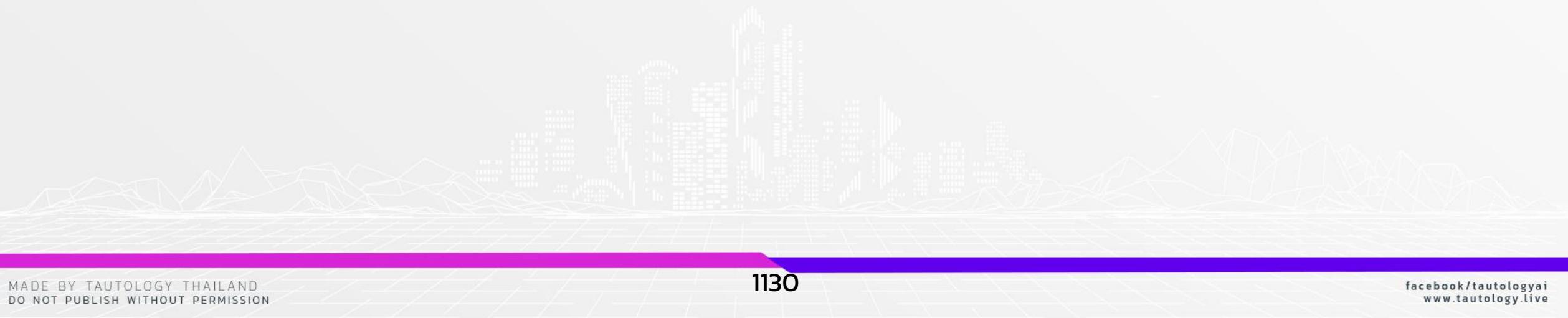
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.5	-4.5
3	-4.16	-4.16
4	-3.91	-3.9
5	-3.69	-3.68
6	-3.5	-3.49
:	:	:
198	-0.01	-0.02
199	-0.01	-0.02
209	-0.01	-0.02

Adagrad



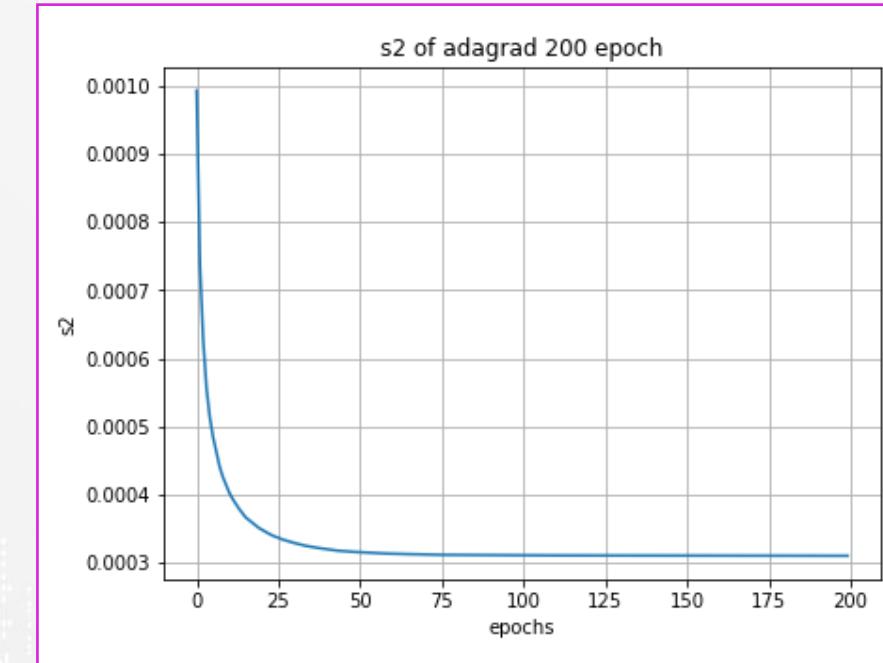
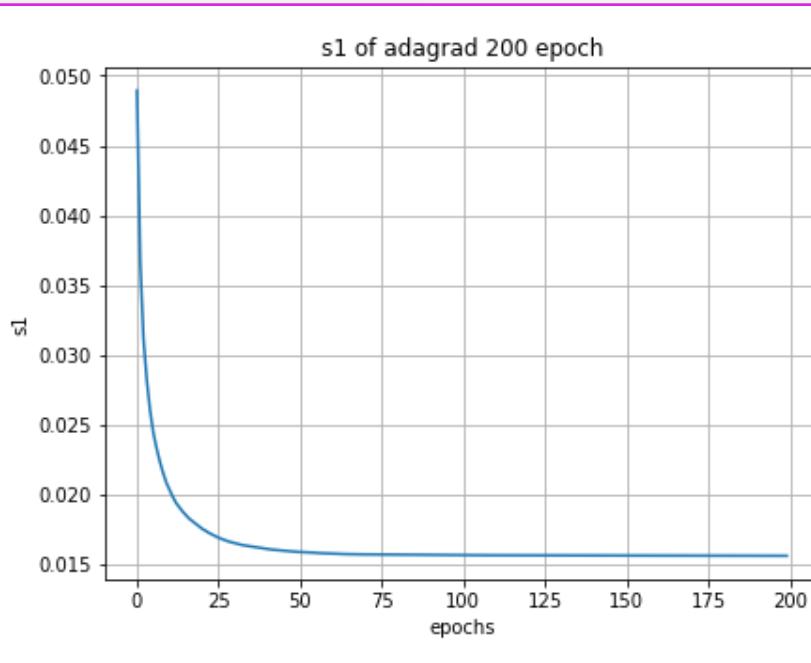
Adagrad

แต่ปัญหาของ adagrad ก็คือ เมื่อทำการ train model ไปนาน ๆ แล้ว learning rate จะลู่เข้าสู่ 0 (ถึงแม้ว่า model จะยัง train ไม่เสร็จก็ตาม)



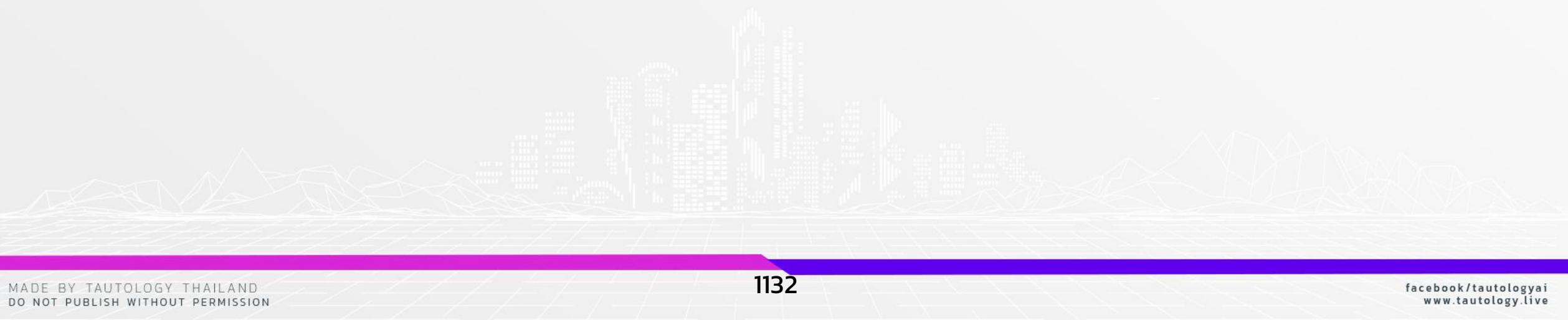
Adagrad

Adagrad problem



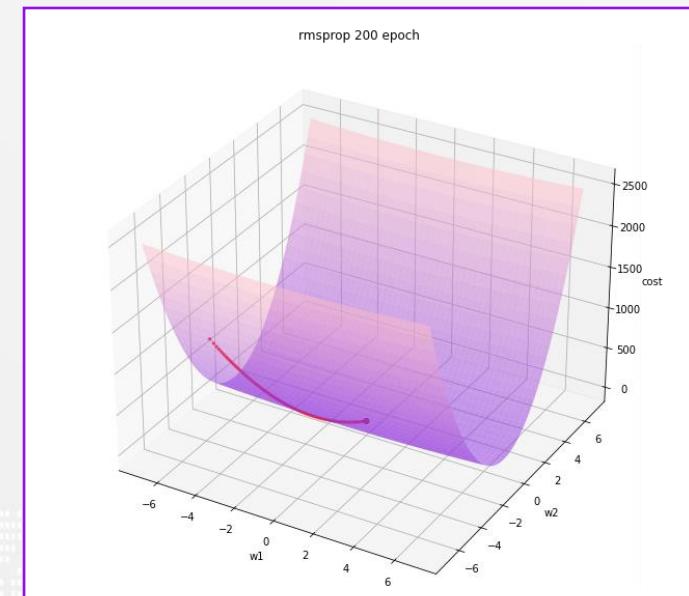
Adaptive Learning Rate Base Algorithm

- ✓ **Adagrad**
- RMSprop



RMSprop

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



RMSprop

ชั่งวิธีแก้ปัญหานั้นคือ เปลี่ยนพจน์สะสม **gradient**
กำลังสอง ให้อยู่ในรูป **exponential moving**
average

RMSprop

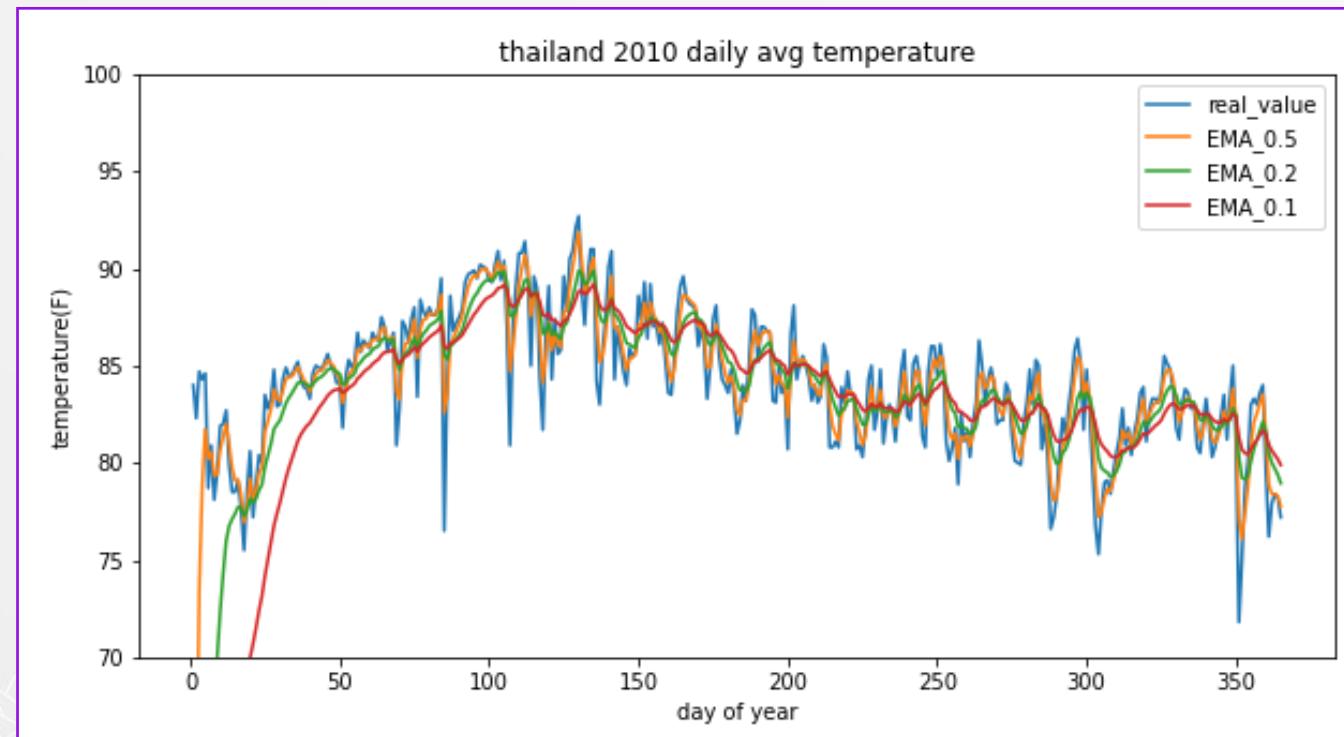
Exponential moving average คือการหาค่าเฉลี่ยถ่วงน้ำหนักแบบ exponential ของตัวแปรที่มีการเปลี่ยนแปลงตามเวลา

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

โดย ◆ $\beta \in (0,1)$ คือ ค่าคงที่ (เพื่อให้เกิดการลดลงของน้ำหนักที่เวลา ก่อนหน้า)

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 g
 - 4.2 คำนวณ G
 - 4.2 คำนวณ S
 - 4.3 คำนวณ θ

RMSprop

1. กำหนด $\theta_0 = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$
2. กำหนด epoch = 200
3. กำหนดค่า $\alpha = 0.0375, \beta = 0.9$

RMSprop

4. Update θ : epoch = 1

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

$$4.2 \mathbf{G}_1 = \begin{bmatrix} 0.9 \times 0 + 0.1 \times (-10.04)^2 \\ 0.9 \times 0 + 0.1 \times (-495.21)^2 \end{bmatrix} = \begin{bmatrix} 10.08 \\ 24523.51 \end{bmatrix}$$

$$4.3 \mathbf{S}_1 = \begin{bmatrix} \frac{0.5}{\sqrt{10.08 + (1 \times 10^{-8})}} \\ \frac{0.5}{\sqrt{24523.51 + (1 \times 10^{-8})}} \end{bmatrix} = \begin{bmatrix} \frac{0.0375}{3.18} \\ \frac{0.0375}{156.6} \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.0002 \end{bmatrix}$$

$$4.4 \mathbf{\theta}_1 = \begin{bmatrix} -5 - (0.01 \times -10.04) \\ -5 - (0.0002 \times -495.21) \end{bmatrix} = \begin{bmatrix} -4.88 \\ -4.88 \end{bmatrix}$$

RMSprop

4. Update θ : epoch = 2

$$4.1 \mathbf{g}_2 = \begin{bmatrix} -9.87 \\ -493.7 \end{bmatrix}$$

$$4.2 \mathbf{G}_2 = \begin{bmatrix} 0.9 \times 10.08 + 0.1 \times (-9.87)^2 \\ 0.9 \times 24523.51 + 0.1 \times (-493.7)^2 \end{bmatrix} = \begin{bmatrix} 18.81 \\ 46444.86 \end{bmatrix}$$

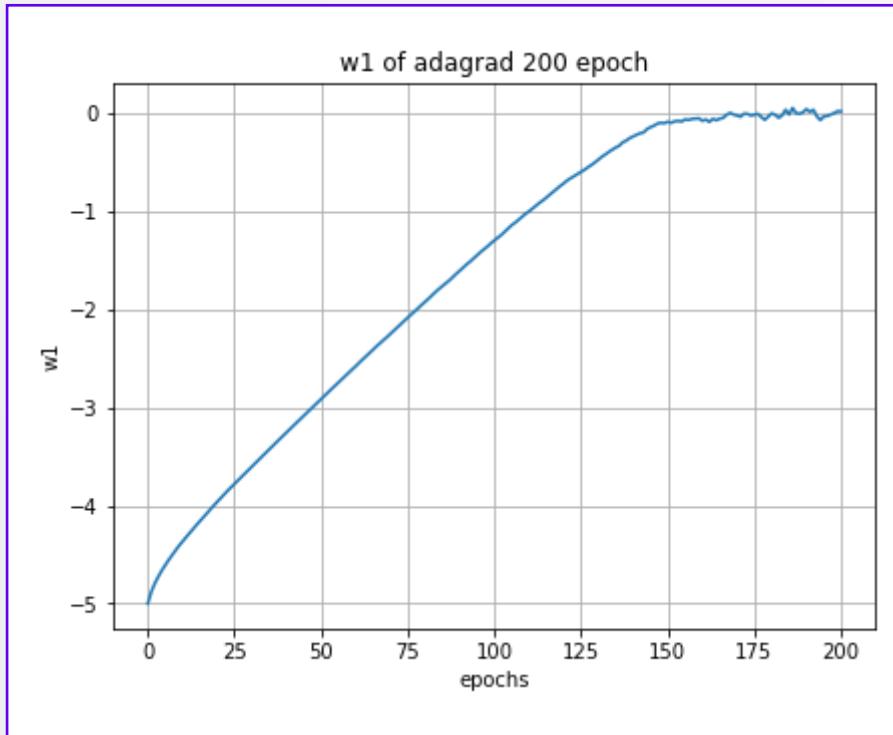
$$4.3 \mathbf{S}_2 = \begin{bmatrix} \frac{0.5}{\sqrt{18.81 + (1 \times 10^{-8})}} \\ \frac{0.5}{\sqrt{46444.86 + (1 \times 10^{-8})}} \end{bmatrix} = \begin{bmatrix} \frac{0.0375}{4.34} \\ \frac{0.0375}{215.51} \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.0002 \end{bmatrix}$$

$$4.4 \mathbf{\theta}_2 = \begin{bmatrix} -4.88 - (0.01 \times -9.87) \\ -4.88 - (0.0002 \times -493.7) \end{bmatrix} = \begin{bmatrix} -4.8 \\ -4.8 \end{bmatrix}$$

RMSprop

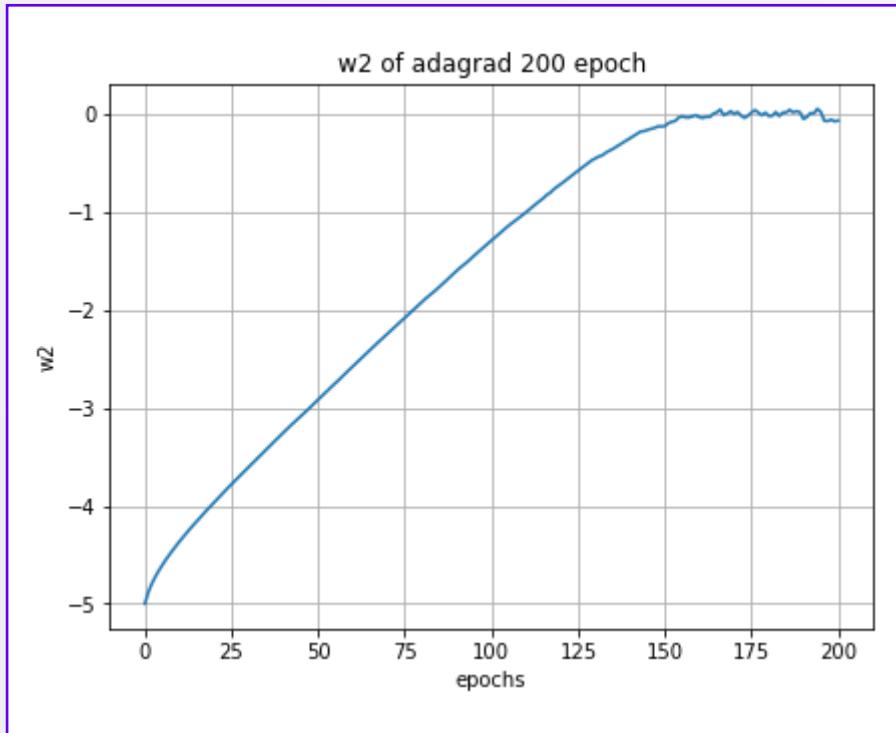
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.88	-4.88
3	-4.8	-4.8
4	-4.73	-4.73
5	-4.66	-4.66
6	-4.61	-4.61
:	:	:
198	0	-0.06
199	-0.02	-0.08
200	-0.02	-0.07

RMSprop



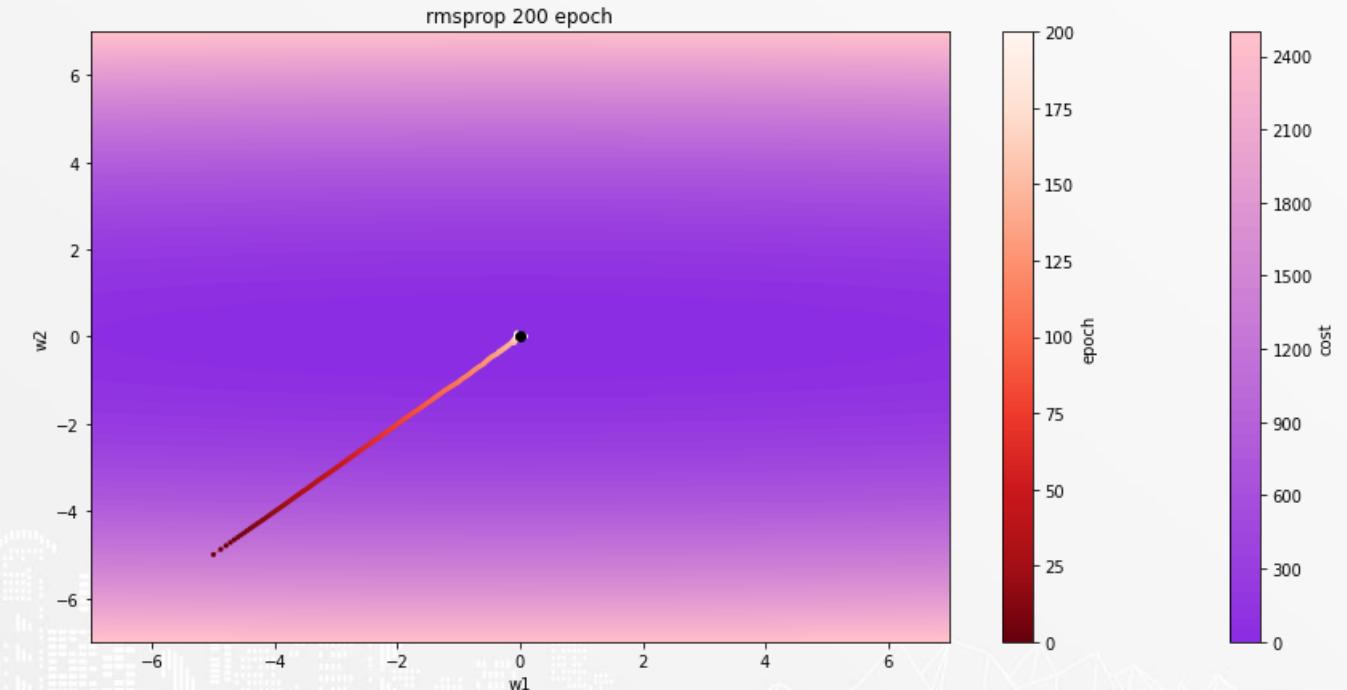
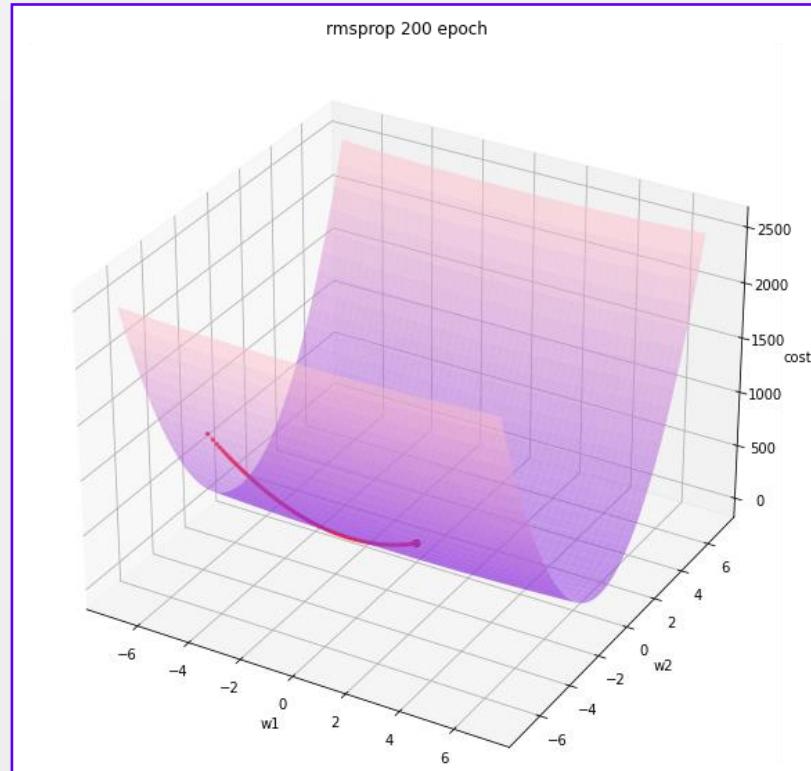
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.88	-4.88
3	-4.8	-4.8
4	-4.73	-4.73
5	-4.66	-4.66
6	-4.61	-4.61
:	:	:
198	0	-0.06
199	-0.02	-0.08
200	-0.02	-0.07

RMSprop

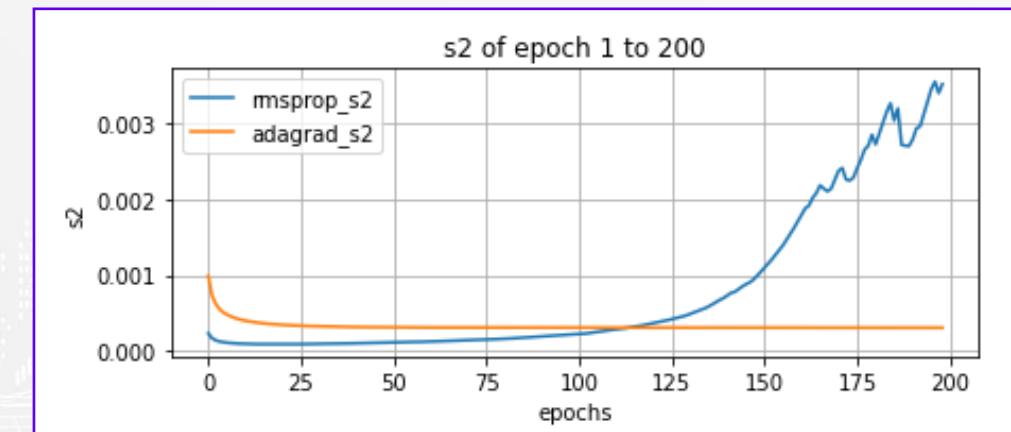
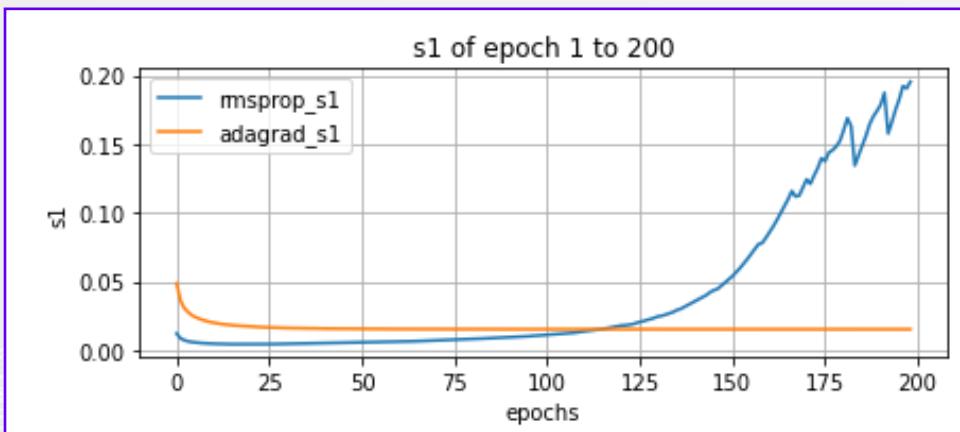
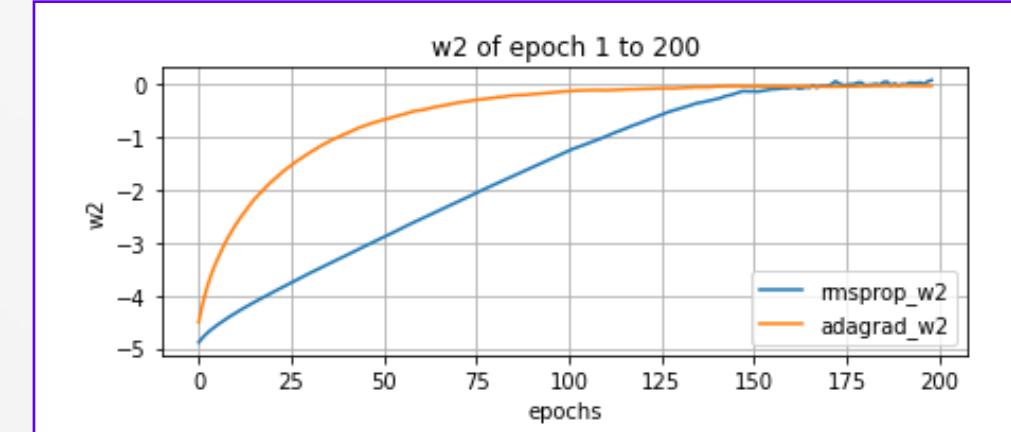
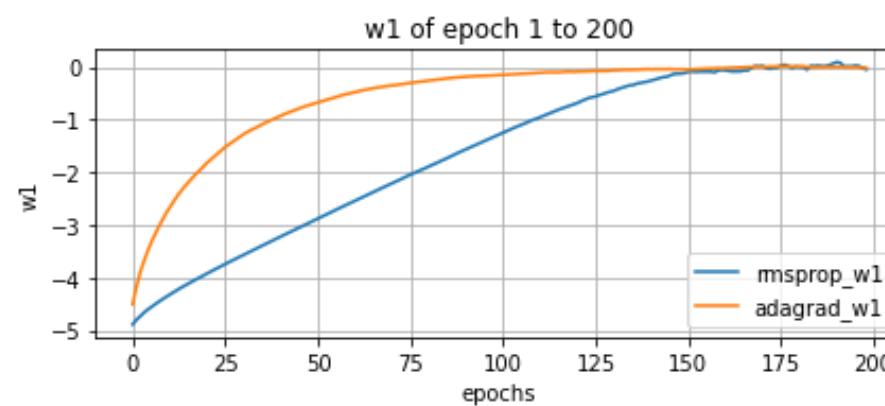


epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.88	-4.88
3	-4.8	-4.8
4	-4.73	-4.73
5	-4.66	-4.66
6	-4.61	-4.61
:	:	:
198	0	-0.06
199	-0.02	-0.08
200	-0.02	-0.07

RMSprop

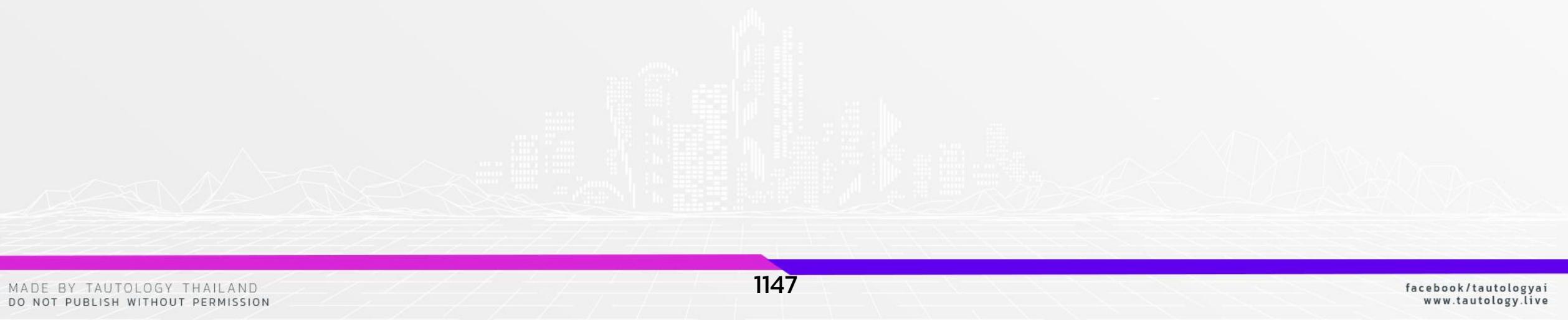


RMSprop



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

ชั้งแนวคิดที่นำมาใช้คือ

- การสะสमความเร็วในการเคลื่อนที่ในแต่ละมิติ (**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 $\diamond \varepsilon = 1 \times 10^{-8}$ (ป้องกันการหารด้วย 0)

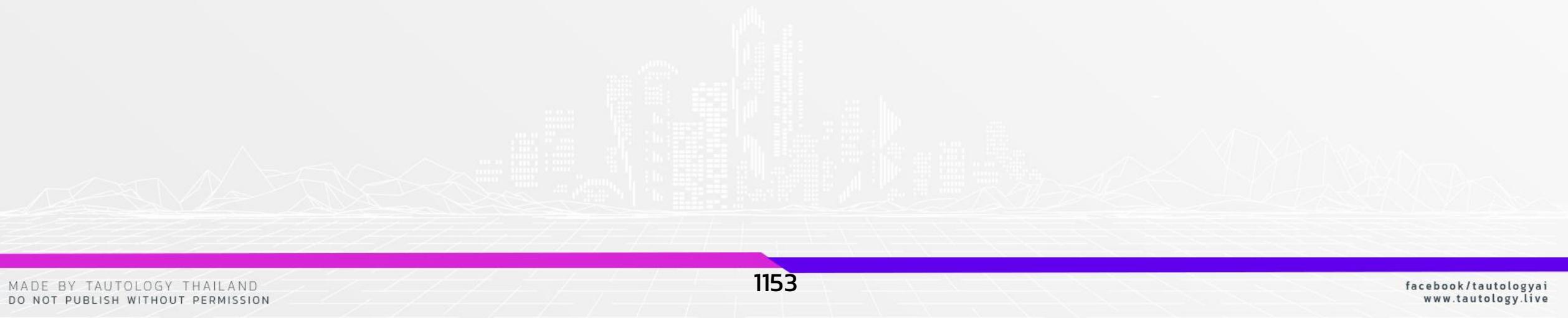
$\diamond m_0 = 0, v_0 = 0$

Adam

แต่ EMA ยังมีข้อเสียคือ...

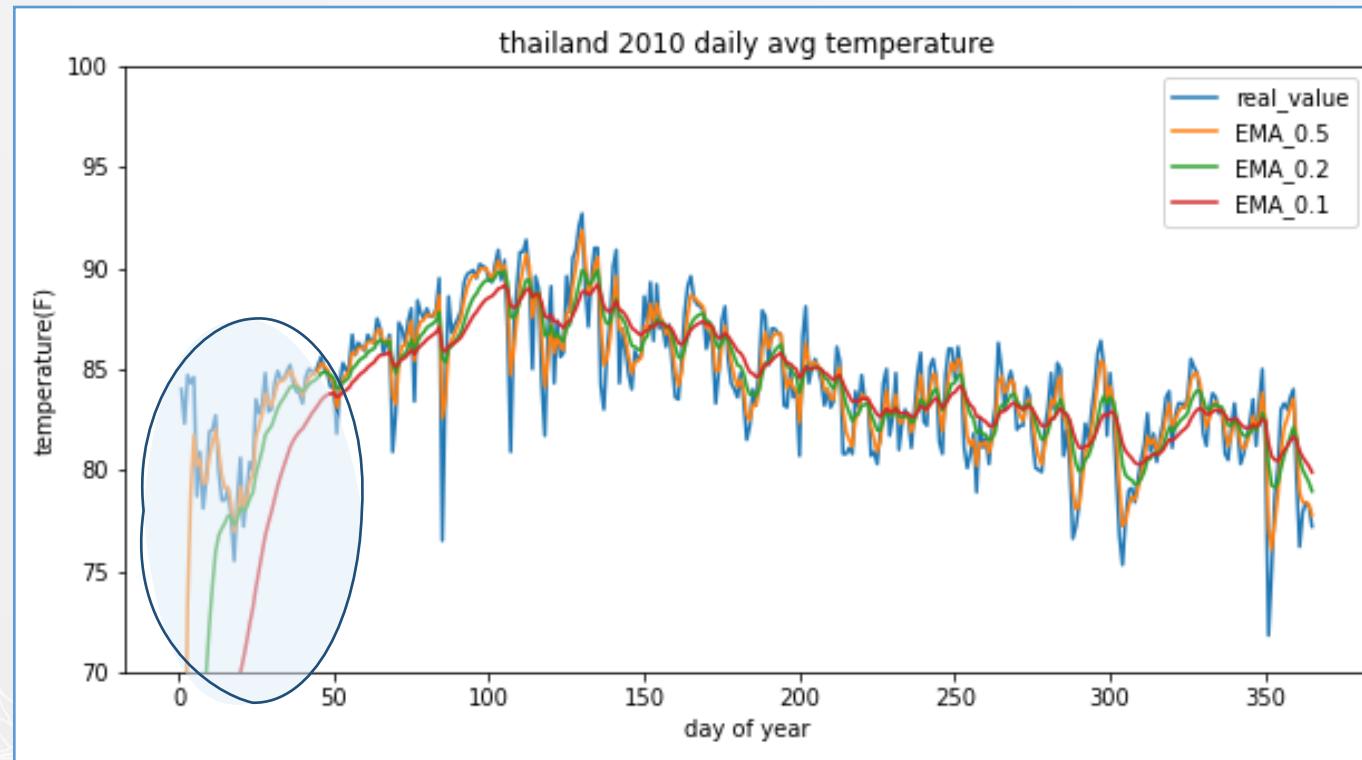
Adam

ในช่วงต้น ๆ ค่าของ EMA จะค่อนข้างแปรปรวน
เนื่องจากเรากำหนดให้ $initial = 0$ เลย



Adam

Exponential Moving Average



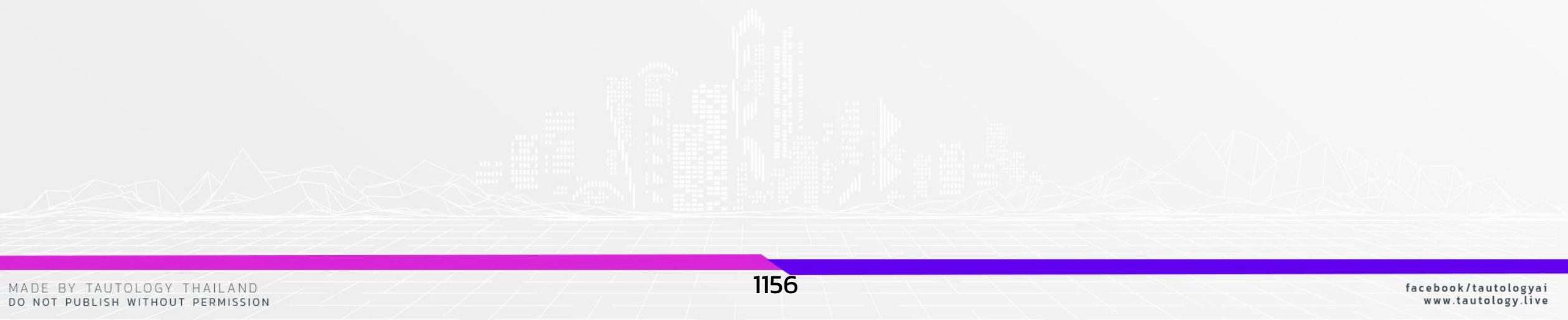
Adam

ดังนั้น เราจะแก้ปัญหาโดยใช้
“ Bias Correction ”
เพื่อให้ค่า EMA ตอนเริ่มต้นไม่เป็น 0

Adam

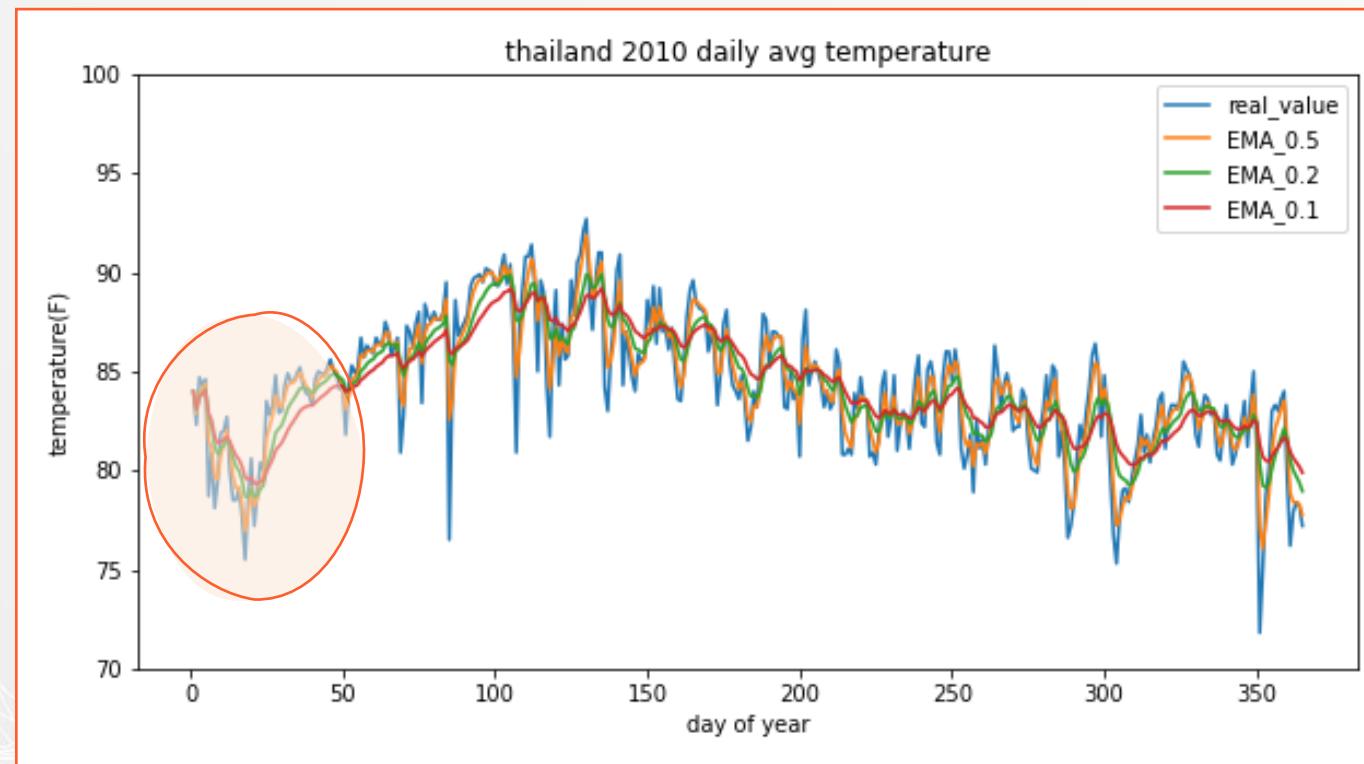
Bias Correction of Exponential Moving Average

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



Adam

Bias Correction of Exponential Moving Average



Adam

ดังนั้น Adam ที่ใช้ bias correction สามารถเขียนให้อยู่ในรูปดังต่อไปนี้

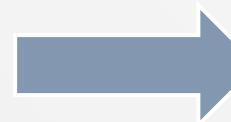
$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t + \varepsilon}} \odot m_t$$

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

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

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

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 g
 - 4.2 คำนวณ m, \hat{m}
 - 4.3 คำนวณ v, \hat{v}
 - 4.4 คำนวณ θ

Adam

1. กำหนด $\theta_0 = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$
2. กำหนด epoch = 200
3. กำหนดค่า $\alpha = 0.075, \beta_1 = 0.9, \beta_2 = 0.999$

Adam

4. Update θ : epoch = 1

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

$$4.2 \quad \mathbf{m}_1 = \begin{bmatrix} 0.9 \times 0 + 0.1 \times -10.04 \\ 0.9 \times 0 + 0.1 \times -495.21 \end{bmatrix} = \begin{bmatrix} -1 \\ -49.52 \end{bmatrix}$$

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

Adam

4. Update θ : epoch = 1

$$4.3 \mathbf{v}_1 = \begin{bmatrix} 0.999 \times 0 + 0.001 \times (-10.04)^2 \\ 0.999 \times 0 + 0.001 \times (-495.21)^2 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 245.23 \end{bmatrix}$$

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

$$4.4 \theta_1 = \begin{bmatrix} -5 - (0.01 \times -10.04) \\ -5 - (0.0002 \times -495.21) \end{bmatrix} = \begin{bmatrix} -4.93 \\ -4.93 \end{bmatrix}$$

Adam

4. Update θ : epoch = 2

$$4.1 \quad \mathbf{g}_2 = \begin{bmatrix} -9.95 \\ -498.06 \end{bmatrix}$$

$$4.2 \quad \mathbf{m}_2 = \begin{bmatrix} 0.9 \times -1.00 + 0.1 \times -9.95 \\ 0.9 \times -49.52 + 0.1 \times -498.06 \end{bmatrix} = \begin{bmatrix} -1.9 \\ -94.37 \end{bmatrix}$$

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

Adam

4. Update θ : epoch = 2

$$4.3 \mathbf{v}_2 = \begin{bmatrix} 0.999 \times 0 + 0.001 \times (-9.95)^2 \\ 0.999 \times 0 + 0.001 \times (-498.06)^2 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 493.05 \end{bmatrix}$$

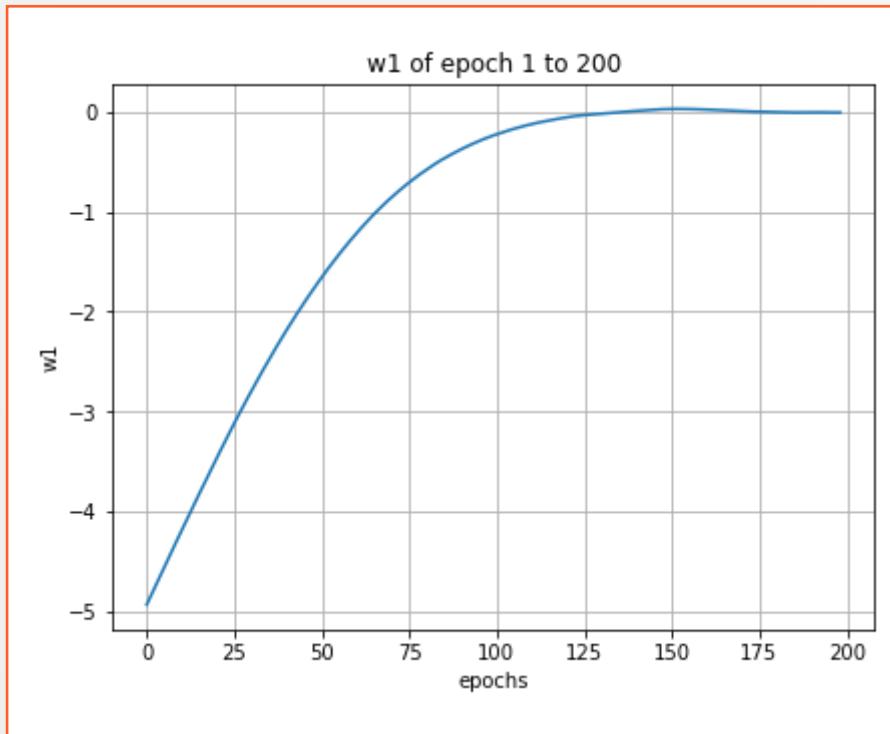
$$\hat{\mathbf{v}}_2 = \begin{bmatrix} \frac{0.2}{1 - (0.999)^1} \\ \frac{493.05}{1 - (0.999)^1} \end{bmatrix} = \begin{bmatrix} -10 \\ -496.71 \end{bmatrix}$$

$$4.4 \theta_2 = \begin{bmatrix} -4.93 - (0.01 \times -10) \\ -4.93 - (0.0002 \times -496.71) \end{bmatrix} = \begin{bmatrix} -4.85 \\ -4.85 \end{bmatrix}$$

Adam

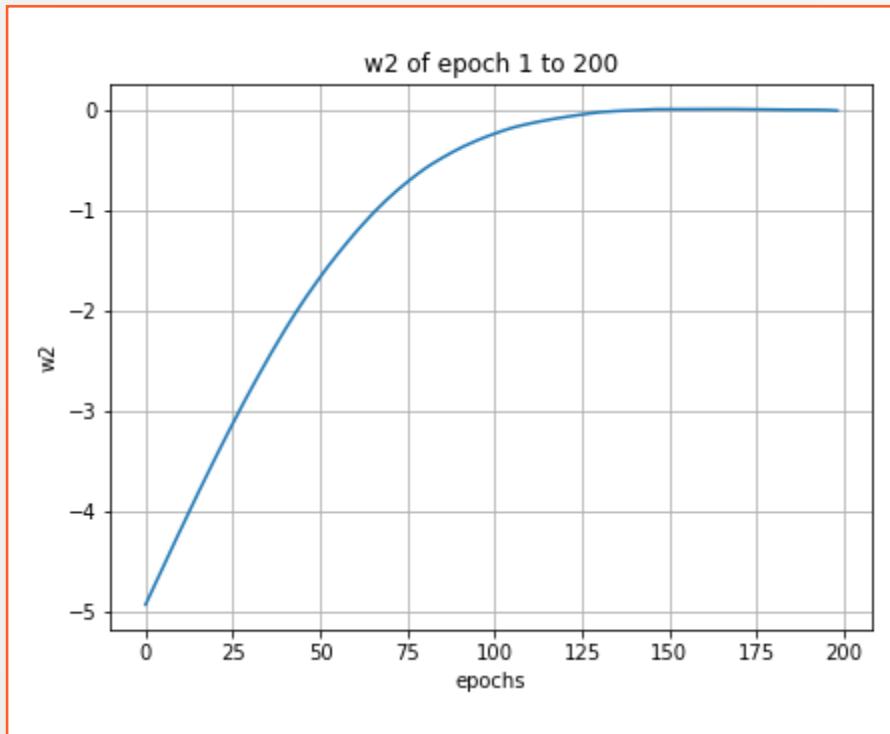
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.93	-4.93
3	-4.85	-4.85
4	-4.78	-4.78
5	-4.7	-4.7
6	-4.63	-4.63
:	:	:
198	-0.01	-0.001
199	-0.01	-0.003
200	-0.009	-0.005

Adam



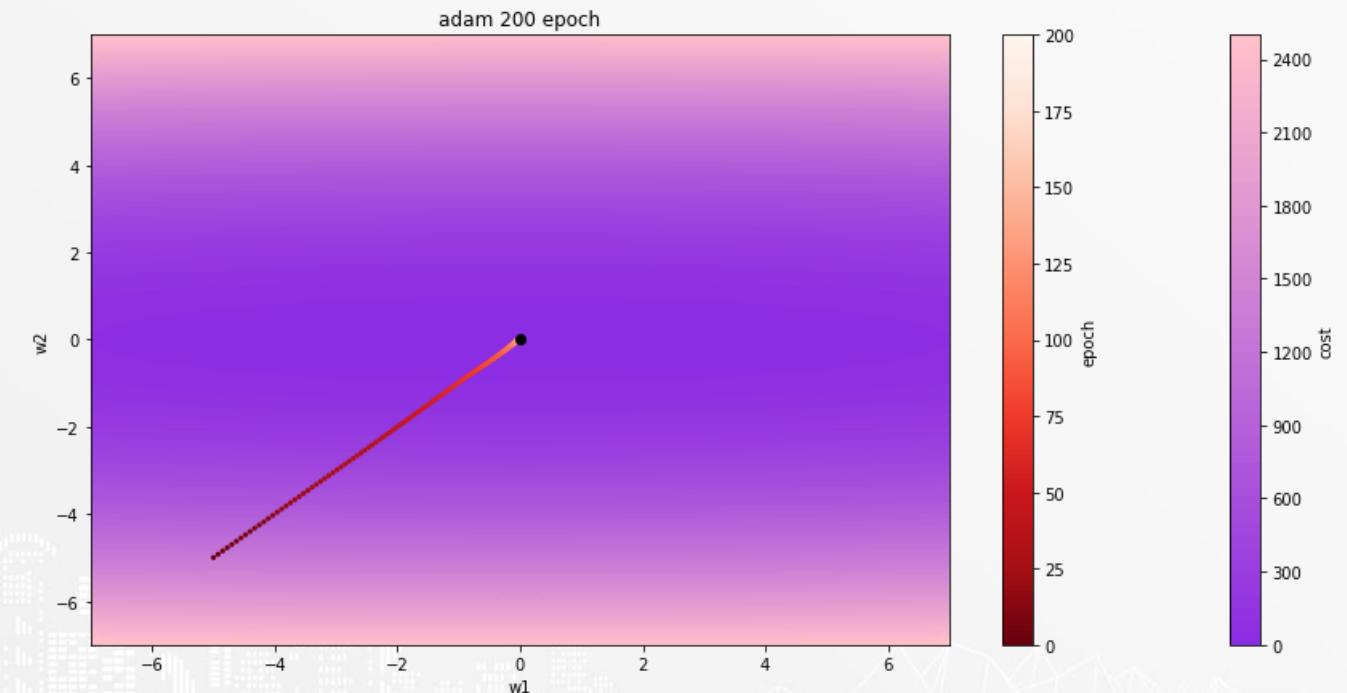
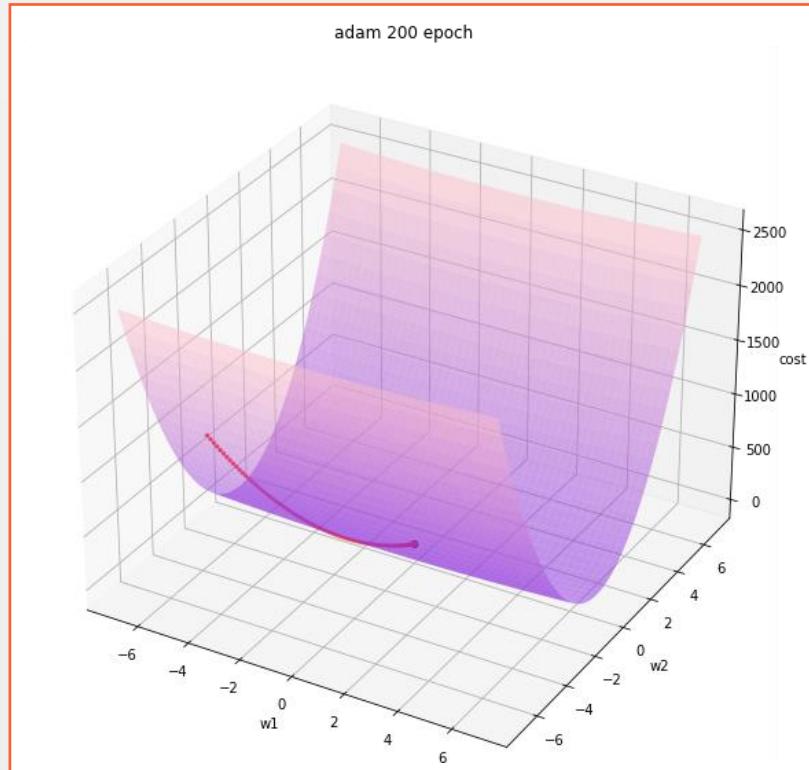
epoch	$\theta^{(1)}$	$\theta^{(2)}$
1	-5	-5
2	-4.93	-4.93
3	-4.85	-4.85
4	-4.78	-4.78
5	-4.7	-4.7
6	-4.63	-4.63
:	:	:
198	-0.01	-0.001
199	-0.01	-0.003
200	-0.009	-0.005

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

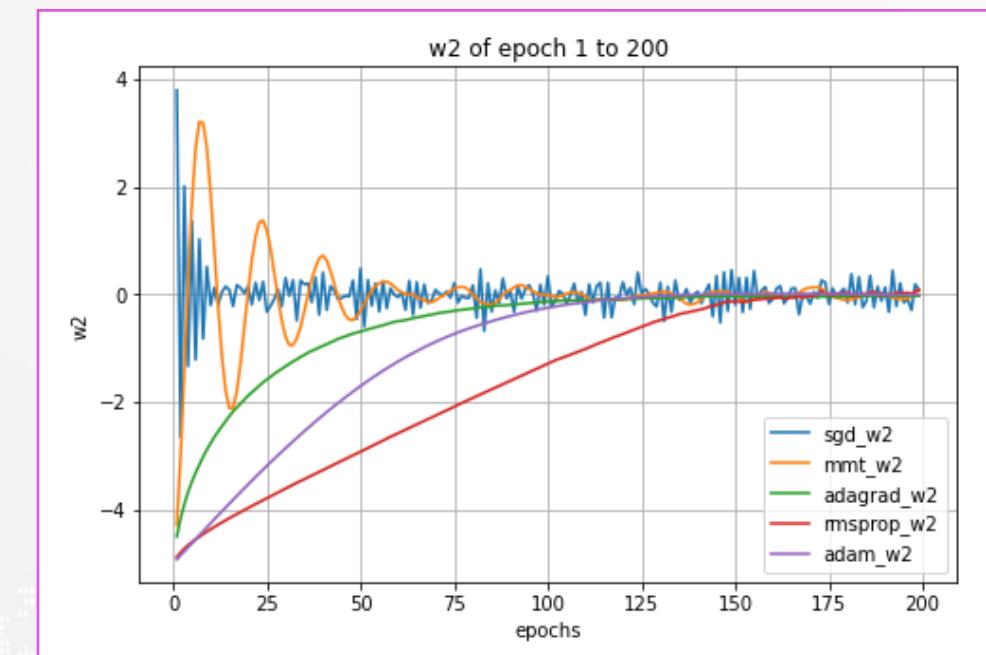
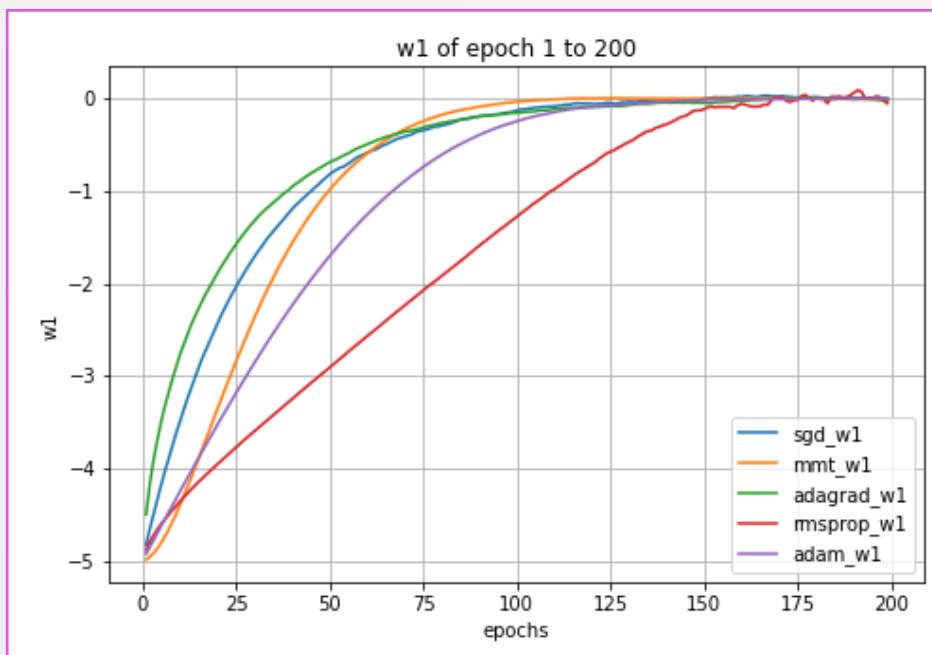


Optimization Algorithm

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

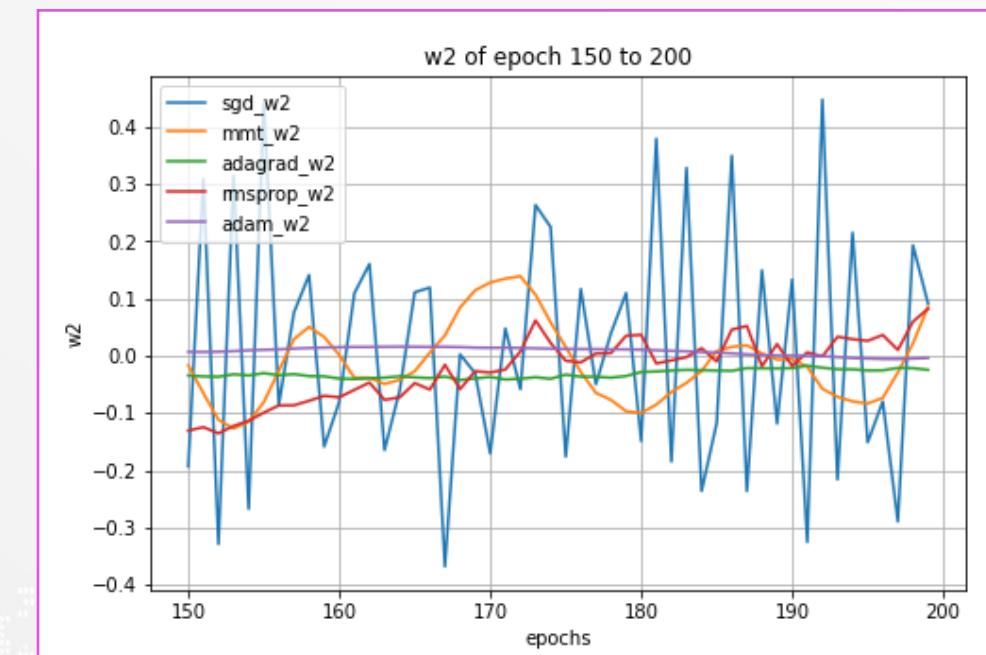
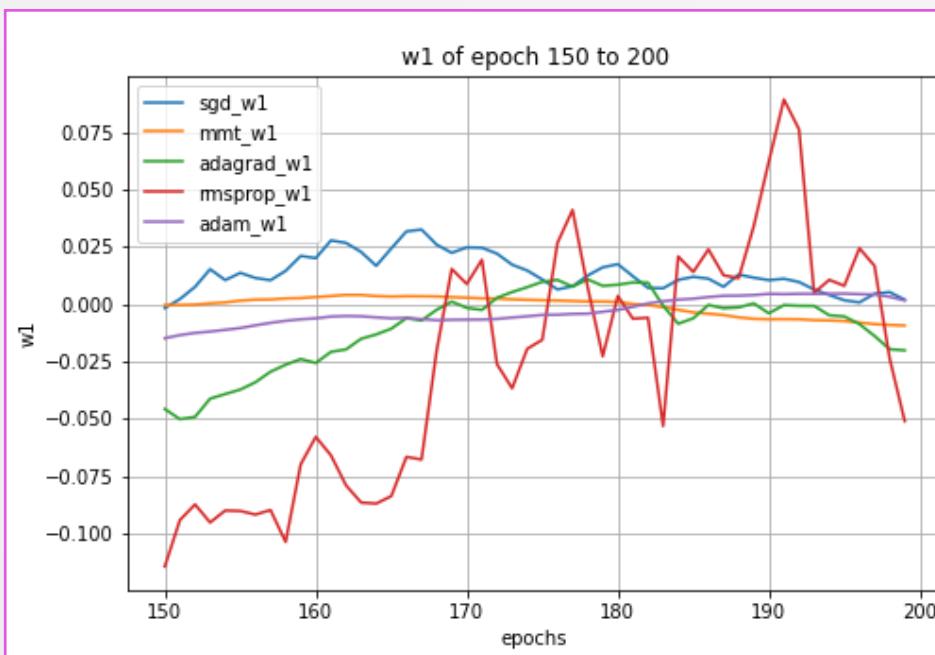
Conclusion

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

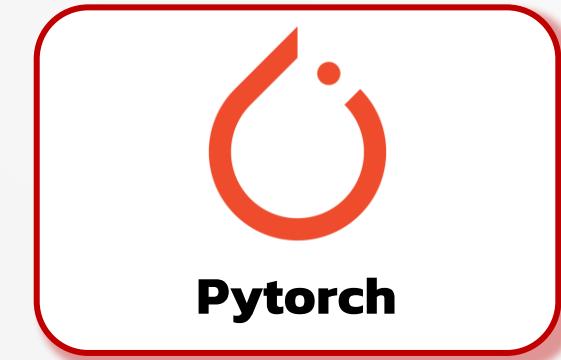
Gradient Descent
Variants

Optimization
Algorithm

Code



Code



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**



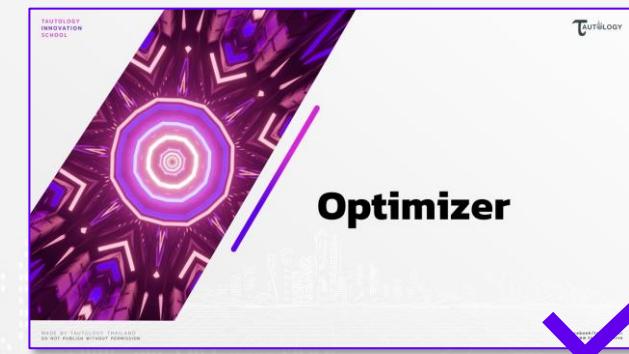
**Optimization
Algorithm**



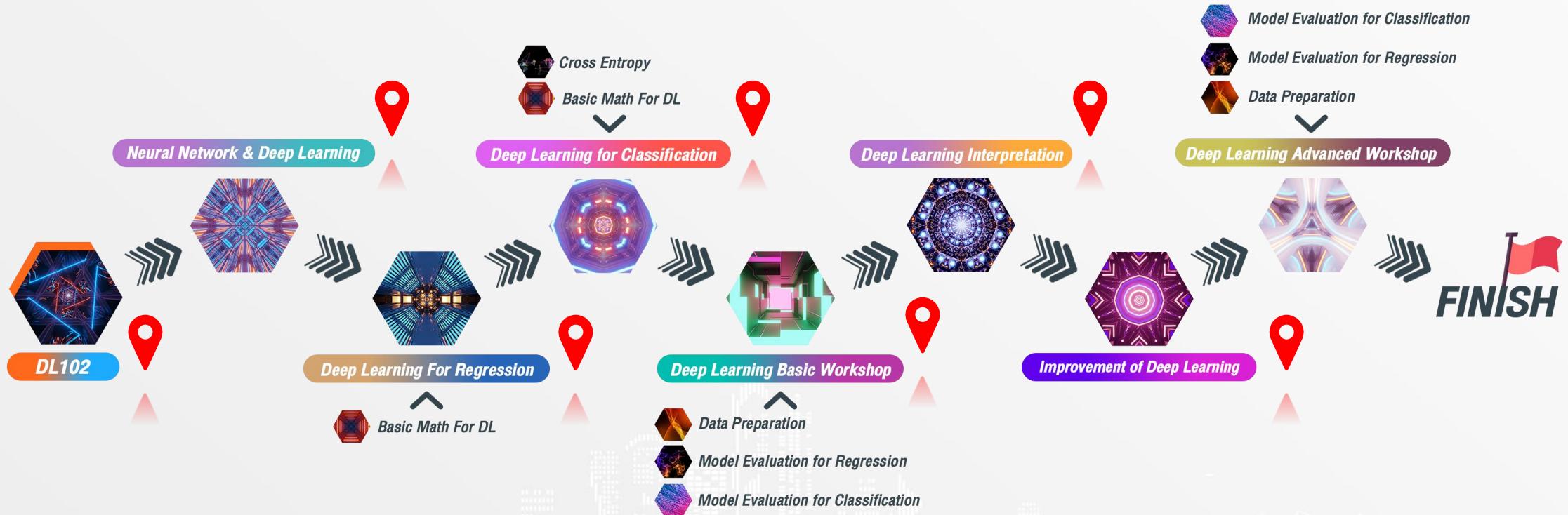
Code



Improvement of Deep Learning



DL103 : Deep Learning





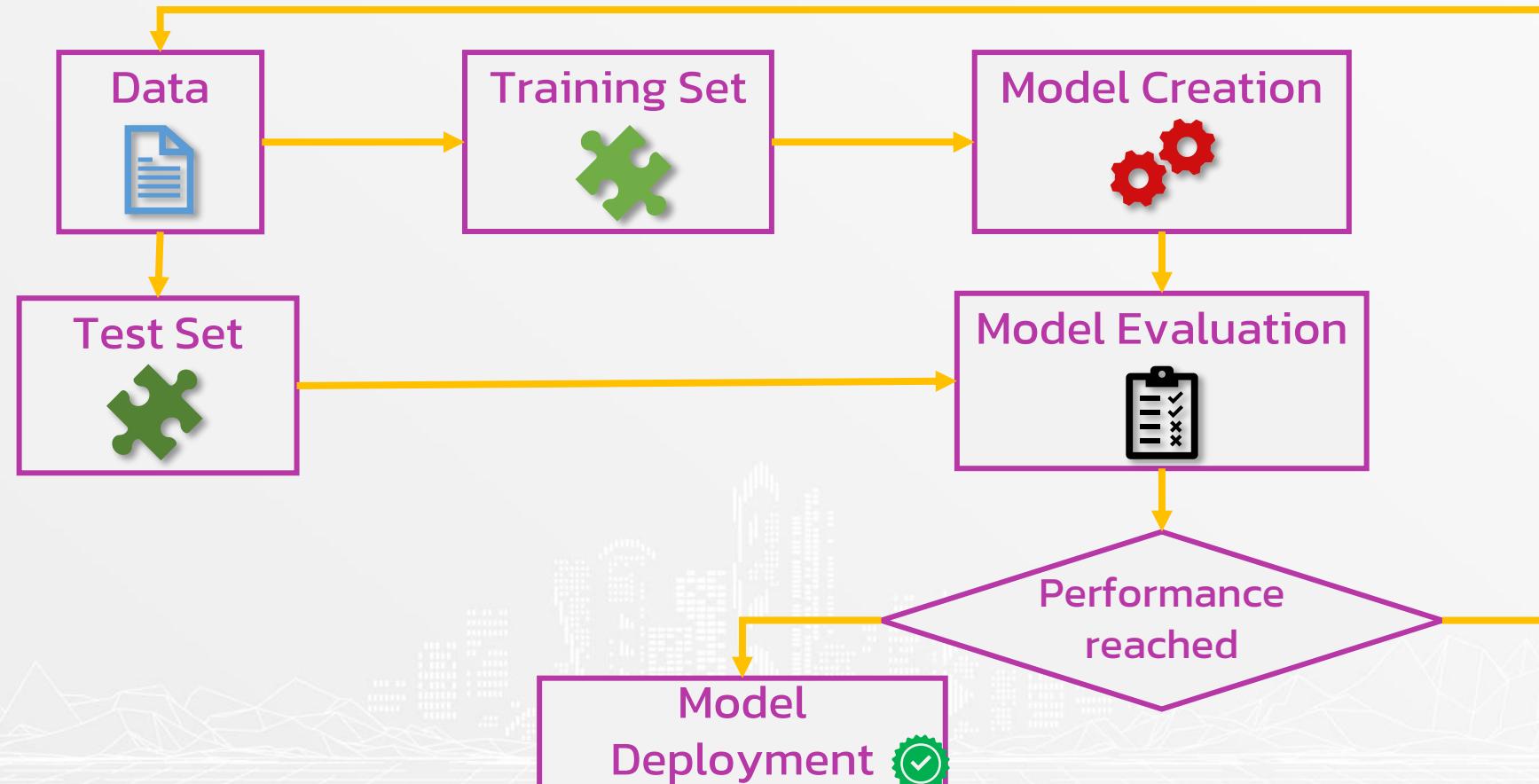
ADVANCED WORKSHOP

DEEP LEARNING

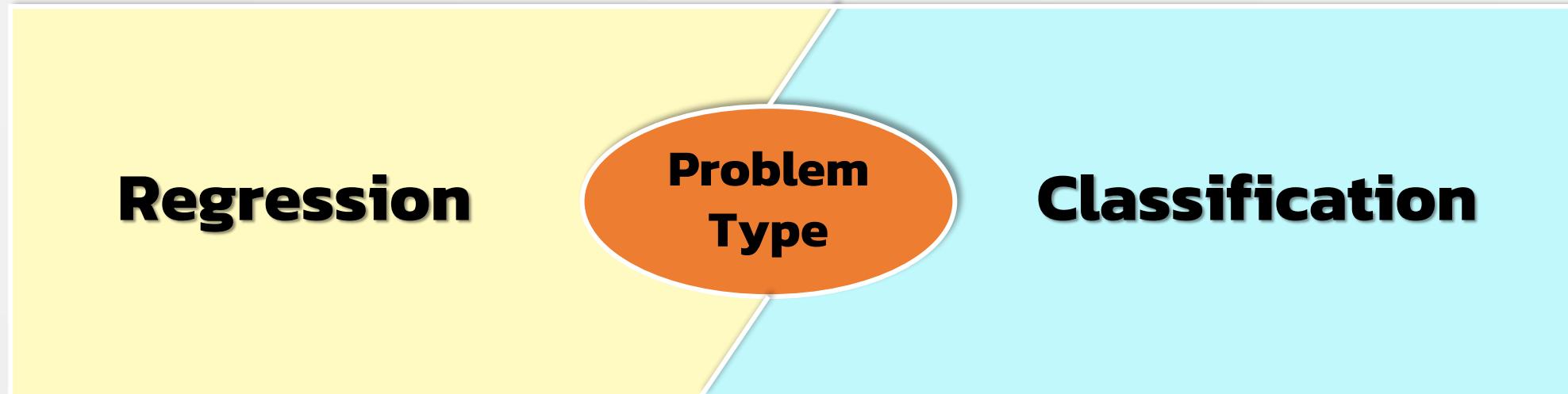
WORKSHOP

BY TAUTOLOGY

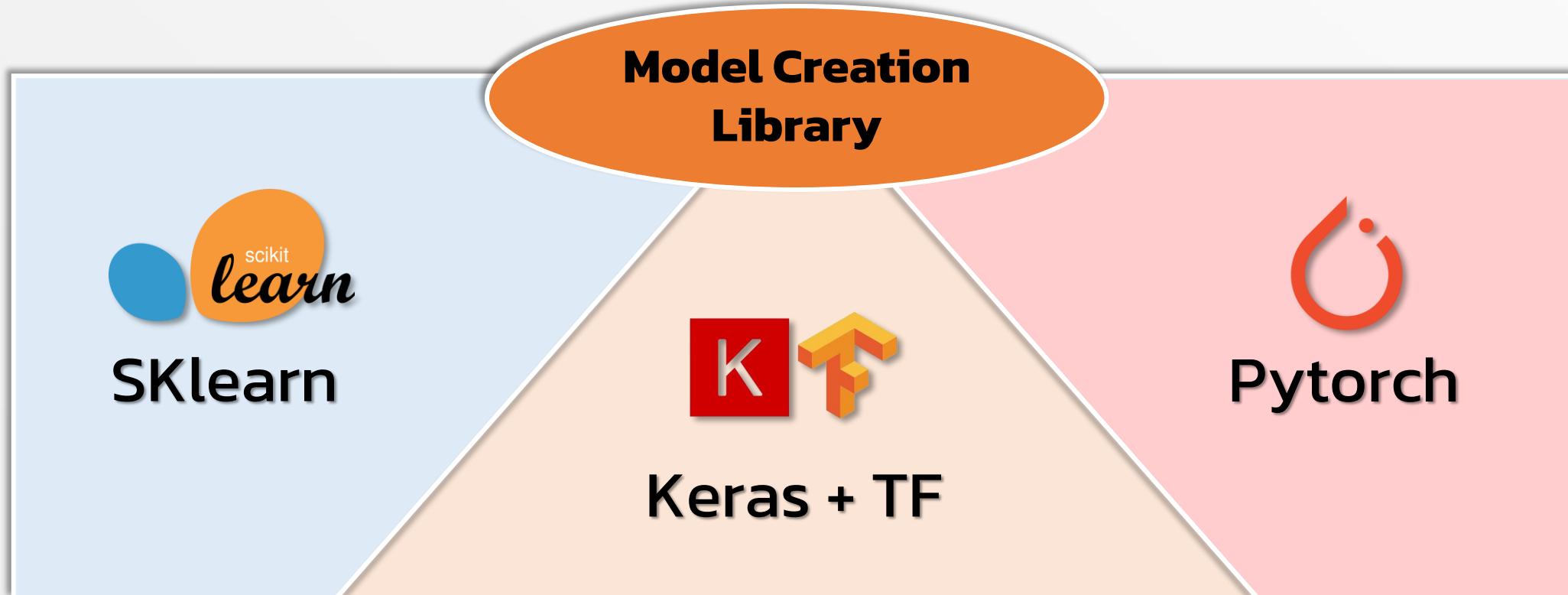
Supervised Learning Workflow



Workshop Overview



Workshop Overview



Supported Library

GPU



Supported Library

Balance Class weight



Supported Library

L2 Regularization



Supported Library

L1 Regularization



Supported Library

Dropout Regularization



Supported Library

Mini Batch



Supported Library

Adam



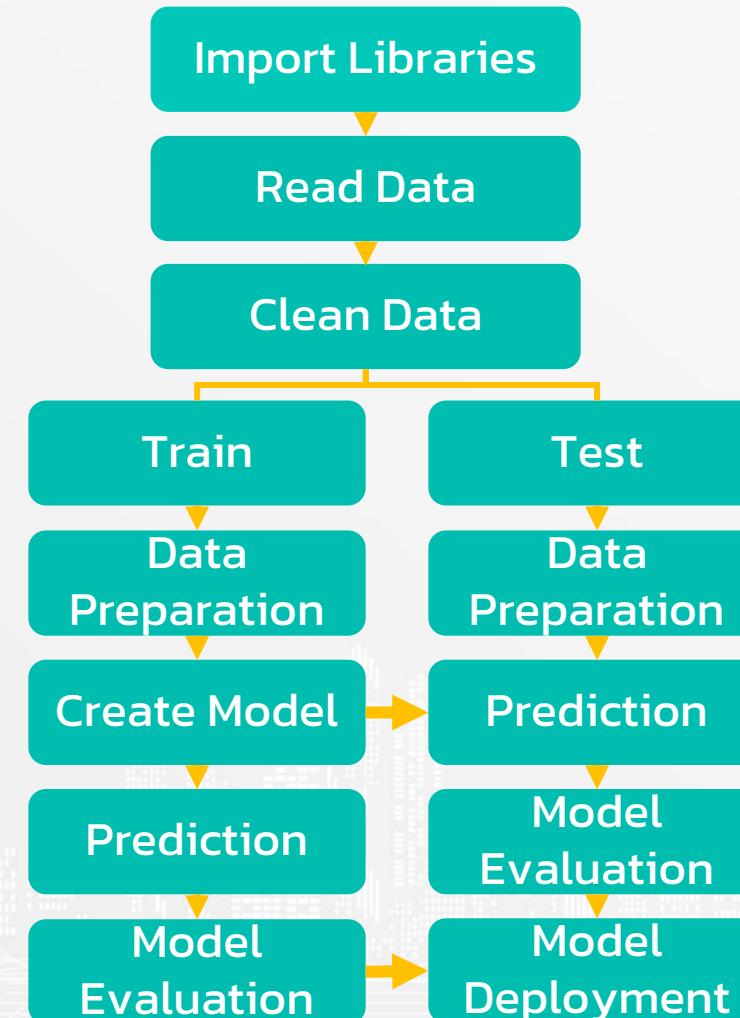
Supported Library

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

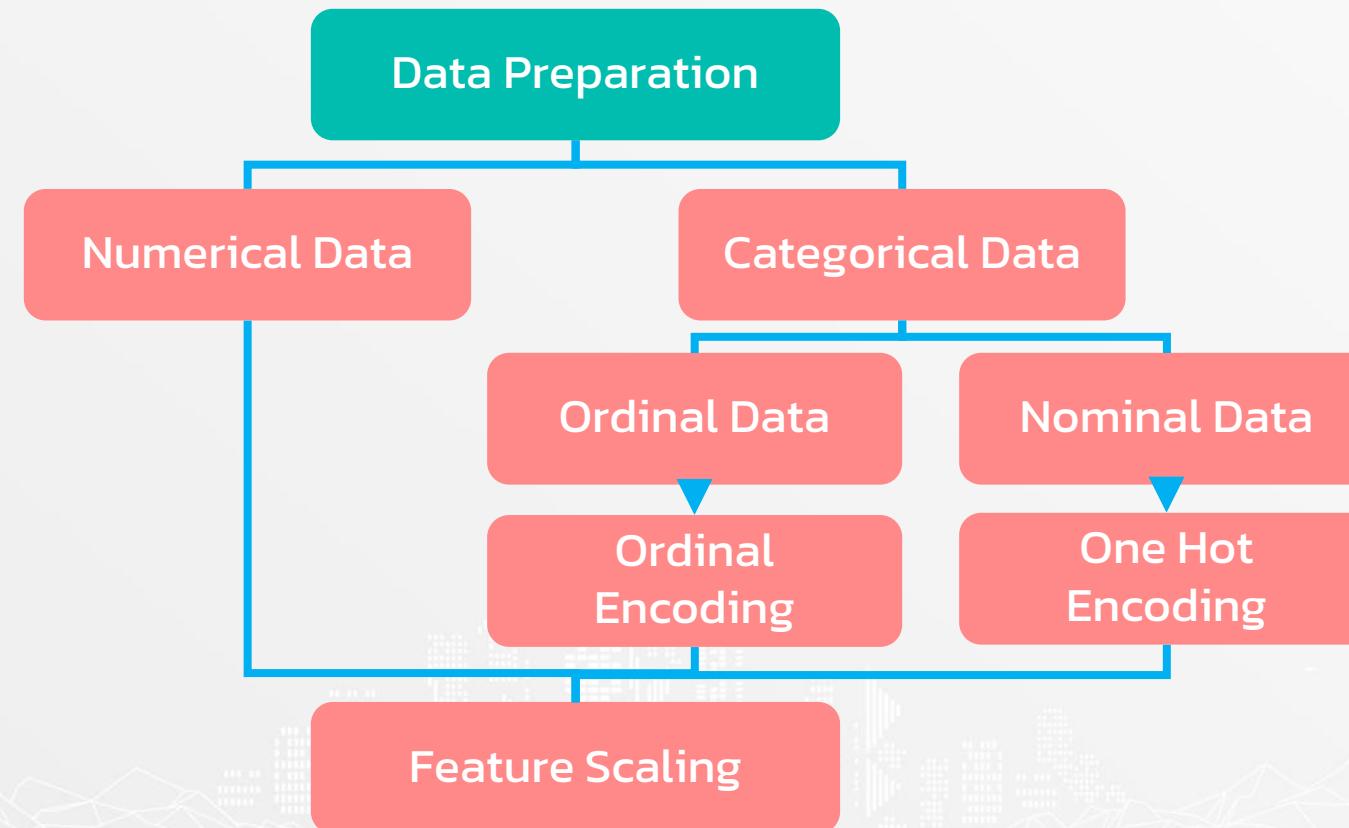
Supported Library

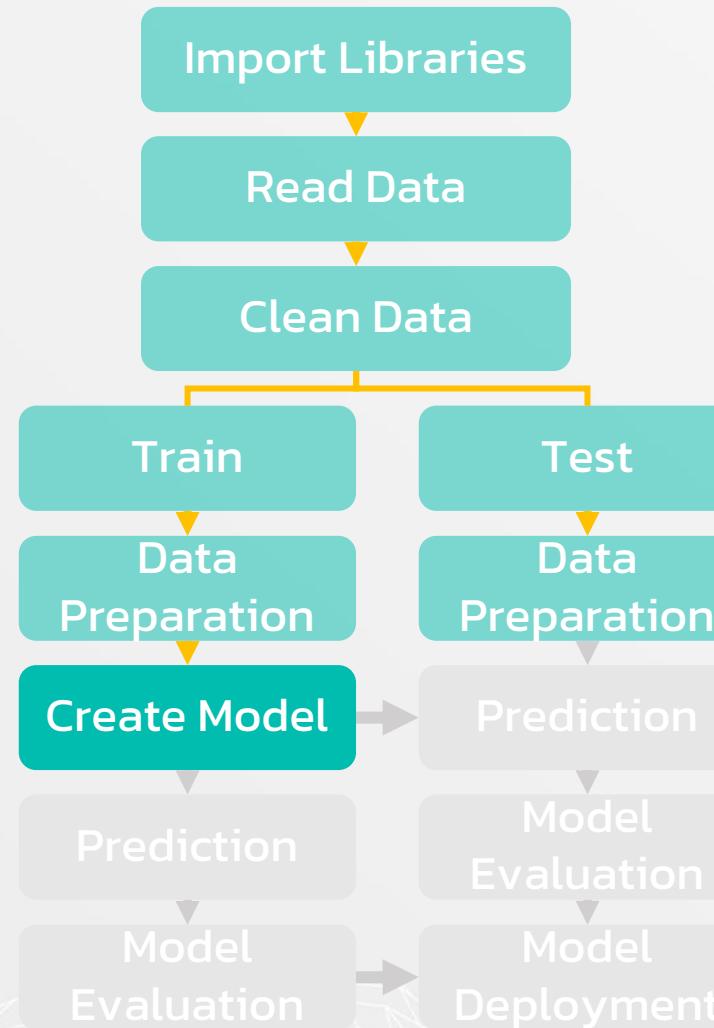
	SKlearn	Keras + Tensorflow	Pytorch
Dropout Regularization	✗	✓	✓
Mini Batch	✓	✓	✓
Adam	✓	✓	✓

Code Pipeline

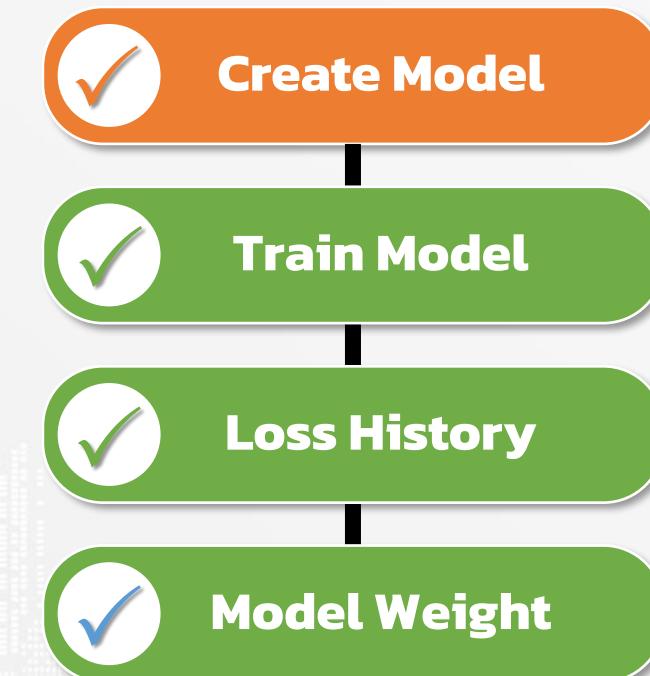


Data Preparation





Create Model



Code - Create Model

Regression

```
1 reg = MLPRegressor(  
2     hidden_layer_sizes=(10, 10),  
3     activation='relu',  
4     learning_rate_init=0.01,  
5     max_iter=1000,  
6     batch_size=64,  
7     alpha=0.1,  
8     solver='adam',  
9     beta_1=0.9,  
10    beta_2=0.999,  
11 )
```

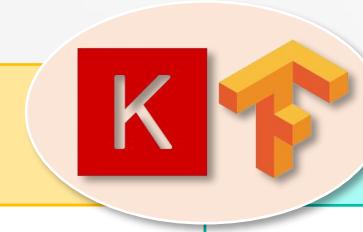
Classification

```
1 clf = MLPClassifier(  
2     hidden_layer_sizes=(10, 10),  
3     activation='relu',  
4     learning_rate_init=1,  
5     max_iter=1000,  
6     batch_size=64,  
7     alpha=0.1,  
8     solver='adam',  
9     beta_1=0.9,  
10    beta_2=0.999  
11 )
```

Code - Create Model

Regression

```
1 reg = KerasMLPRegressor(  
2     input_dim=X_train_scaled.shape[1],  
3     hidden_layer_sizes=(10, 10),  
4     activation_function='relu',  
5     learning_rate_init=0.01,  
6     epochs=1000,  
7     validation_split=0.2,  
8     use_gpu=True,  
9     l1_lambda=0.1,  
10    l2_lambda=0.1,  
11    dropout_rate=[0.25, 0.25],  
12    solver='adam',  
13    batch_size=64  
14 )
```



Classification

```
1 classes = np.unique(y_train)  
2  
1 clf = KerasMLPClassifier(  
2     input_dim=X_train_scaled.shape[1],  
3     hidden_layer_sizes=(10, 10),  
4     activation_function='relu',  
5     classes=classes,  
6     learning_rate_init=1,  
7     epochs=1000,  
8     validation_split=0.2,  
9     use_gpu=True,  
10    class_weight='balanced',  
11    l1_lambda=0.1,  
12    l2_lambda=0.1,  
13    dropout_rate=[0.25, 0.25],  
14    solver='adam',  
15    batch_size=64  
16 )
```

Code - Create Model

Regression

```
1 reg = PytorchMLPRegressor(  
2     input_dim=X_train_scaled.shape[1],  
3     hidden_layer_sizes=(10, 10),  
4     activation_function='relu',  
5     learning_rate_init=0.01,  
6     epochs=1000,  
7     validation_split=0.2,  
8     use_gpu=True,  
9     l1_lambda=0.1,  
10    l2_lambda=0.1,  
11    dropout_rate=[0.25, 0.25],  
12    solver='adam',  
13    batch_size=64  
14 )
```

Classification

```
1 classes = np.unique(y_train)  
2  
1 clf = PytorchMLPClassifier(  
2     input_dim=X_train_scaled.shape[1],  
3     hidden_layer_sizes=(10, 10),  
4     activation_function='relu',  
5     classes=classes,  
6     learning_rate_init=1,  
7     epochs=1000,  
8     validation_split=0.2,  
9     use_gpu=True,  
10    class_weight='balanced',  
11    l1_lambda=0.1,  
12    l2_lambda=0.1,  
13    dropout_rate=[0.25, 0.25],  
14    solver='adam',  
15    batch_size=64  
16 )
```

Ai in Real Estate Business

Abstract

สร้าง model เพื่อประเมินราคาของบ้าน โดย feature ที่นำมาใช้ คือ ข้อมูลทั่วไปของบ้าน และข้อมูลสภาพแวดล้อมภายนอกของบ้าน เช่น

- ขนาดพื้นที่ใช้สอยภายในบ้าน
- ขนาดของที่ดิน
- ขนาดของสระว่ายน้ำ



Why this project important?



- สามารถประเมินราคาบ้านได้อย่างสมเหตุสมผล ที่สุด
- 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

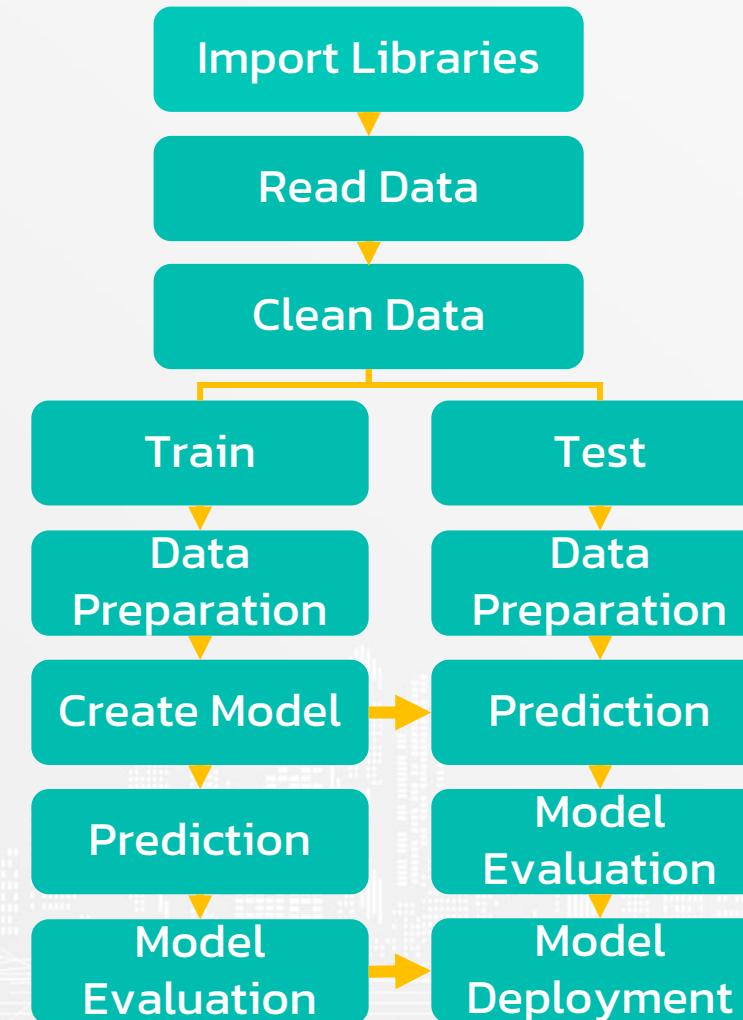
Feature

- MSSubClass – รูปแบบของสิ่งปลูกสร้าง
- MSZoning – โฉนดที่ตั้งสิ่งปลูกสร้าง
- LotFrontage – ระยะห่างจากถนน
- LotArea – ขนาดของที่ดิน
- Street – รูปแบบของการปูถนนหน้าสิ่งปลูกสร้าง (Gravel, Pave)
- :
- LotShape – รูปทรงของสิ่งปลูกสร้าง
- PoolArea – พื้นที่สระว่ายน้ำ
- SaleType – รูปแบบการขาย

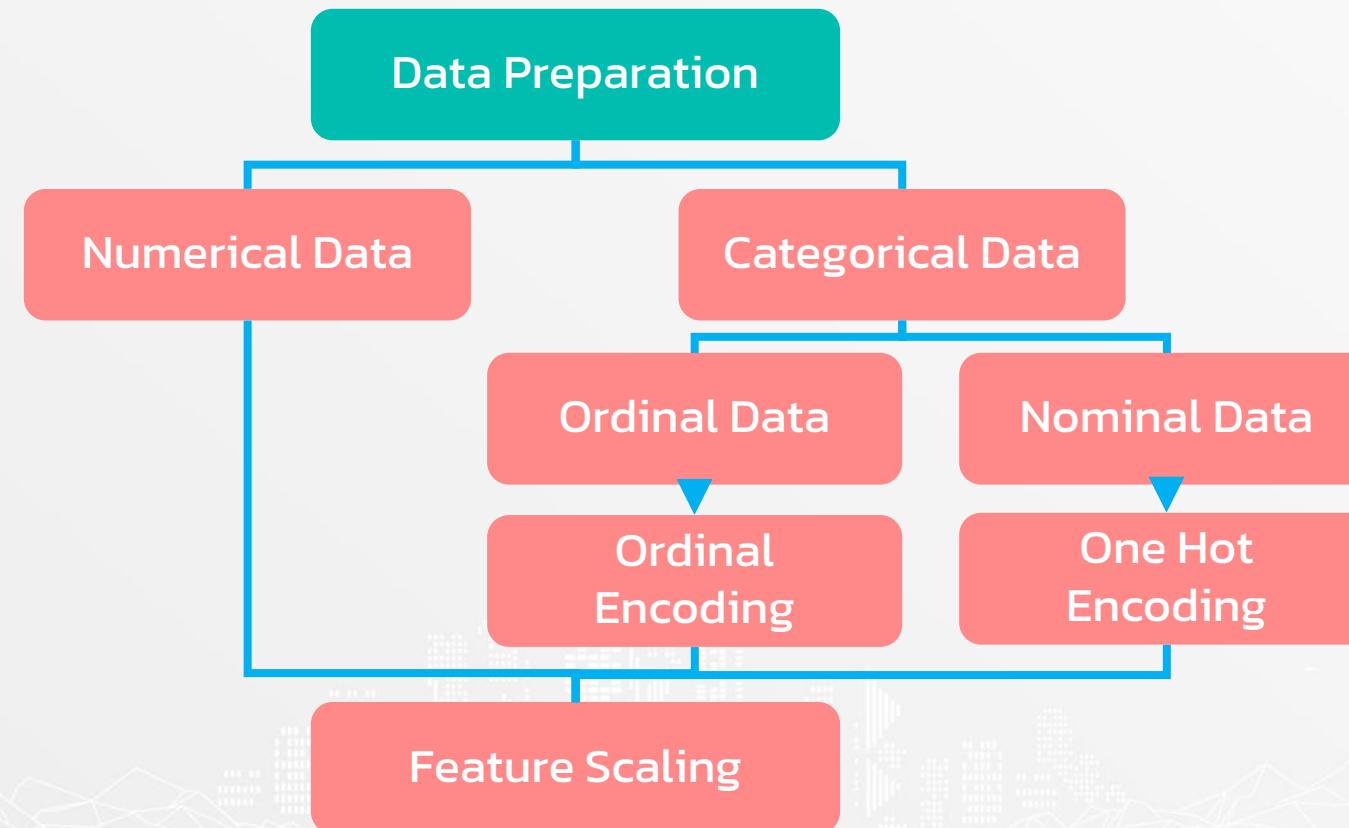
Target

- Sale Price – ราคาขาย

What we learn from this project?



Data Preparation



File



04. HOUSE PRICE



File



04. HOUSE PRICE/sklearn



 **house_price_model.pickle**

 **house_price_sklearn_mc.ipynb**

 **house_price_sklearn_md.ipynb**

File



04. HOUSE PRICE/keras



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?

- สามารถสร้างระบบสำหรับตรวจโรคอัลไซเมอร์ ที่ทำงานได้ตลอด 24 ชั่วโมง
- สามารถนำไปต่อยอดกับการวินิจฉัยโรคอื่น ๆ
- สามารถใช้เป็นพื้นฐานสำหรับการแพทย์ทางไกล
- สามารถนำไปประยุกต์ใช้กับงานที่มีลักษณะ ใกล้เคียงกันได้ เช่น การตรวจจับอาวุธในสนามบิน



Who this project is for?

- ✿ ผู้บริหารโรงพยาบาล
- ✿ บุคลากรทางการแพทย์
- ✿ นักวิเคราะห์ข้อมูล



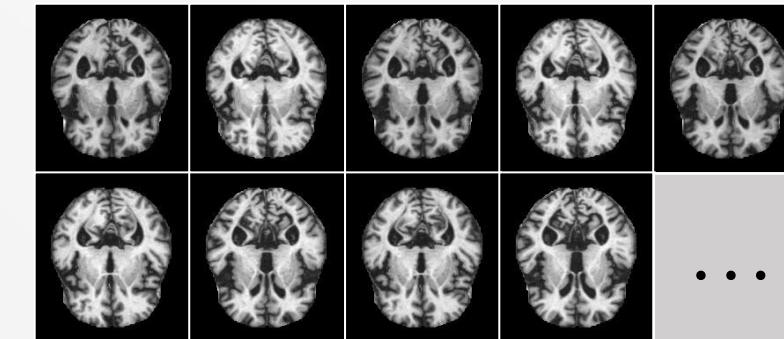
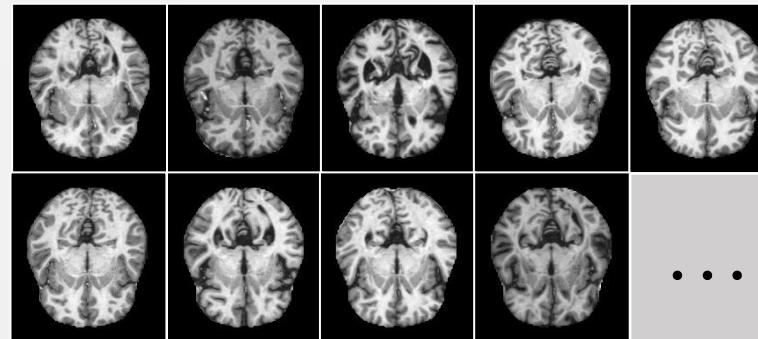
Alzheimer Dataset



<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?

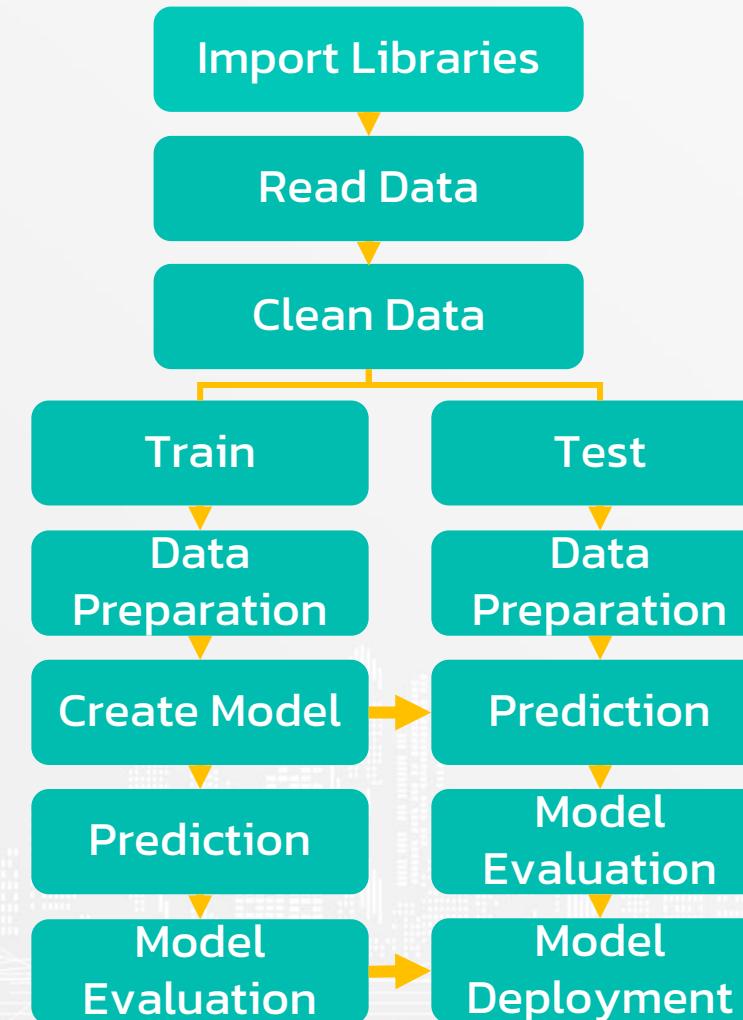


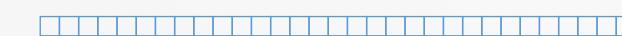
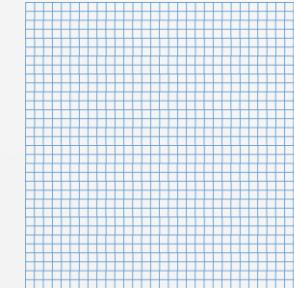
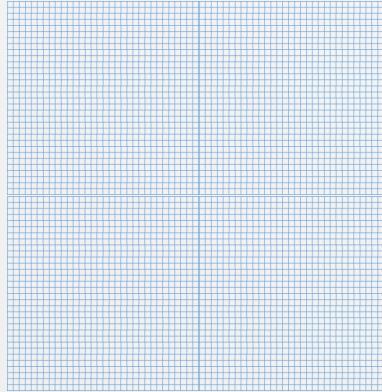
Image to CSV

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

```
1 width = 176
2 height = 208
3
4 X = np.empty([0, width*height])
5 y = np.empty([0, 1])
6
7 for _class in tqdm(classes):
8     img_path = glob('dataset/' + _class + '/*')
9     for path in tqdm(img_path):
10         img = Image.open(path)
11         img = img.resize([width, height])
12         img = np.array(img)
13         img = img.reshape(1, -1)
14         X = np.vstack([X, img])
15         if _class == 'NonDemented':
16             y = np.vstack([y, False])
17         else:
18             y = np.vstack([y, True])
```

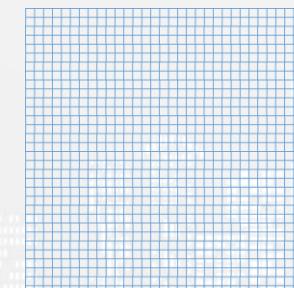
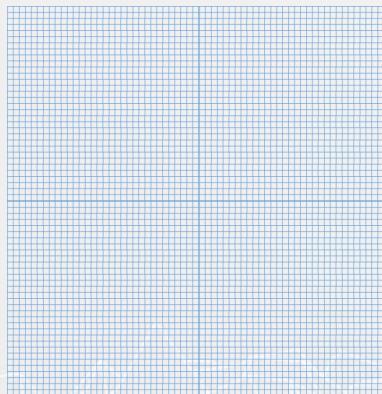


Image to CSV



ເປັນ

176x208



ໄມ່ເປັນ

176x208

1221

Image to CSV

x_1	x_2	x_3	...	x_{36608}	y
0.0	0.0	0.0	...	0.0	1.0
0.0	0.0	0.0	...	0.0	1.0
0.0	0.0	0.0	...	0.0	1.0
:	:	:	:	:	:
0.0	0.0	0.0	...	0.0	1.0

X

y

Image to CSV

```
1 columns = [f'pixel_{i}' for i in range(width*height)]  
2  
3 data = pd.DataFrame(X, columns=columns)  
4 data['label'] = y  
5  
6 data.to_csv('alzheimer_dataset.csv', index=False)
```

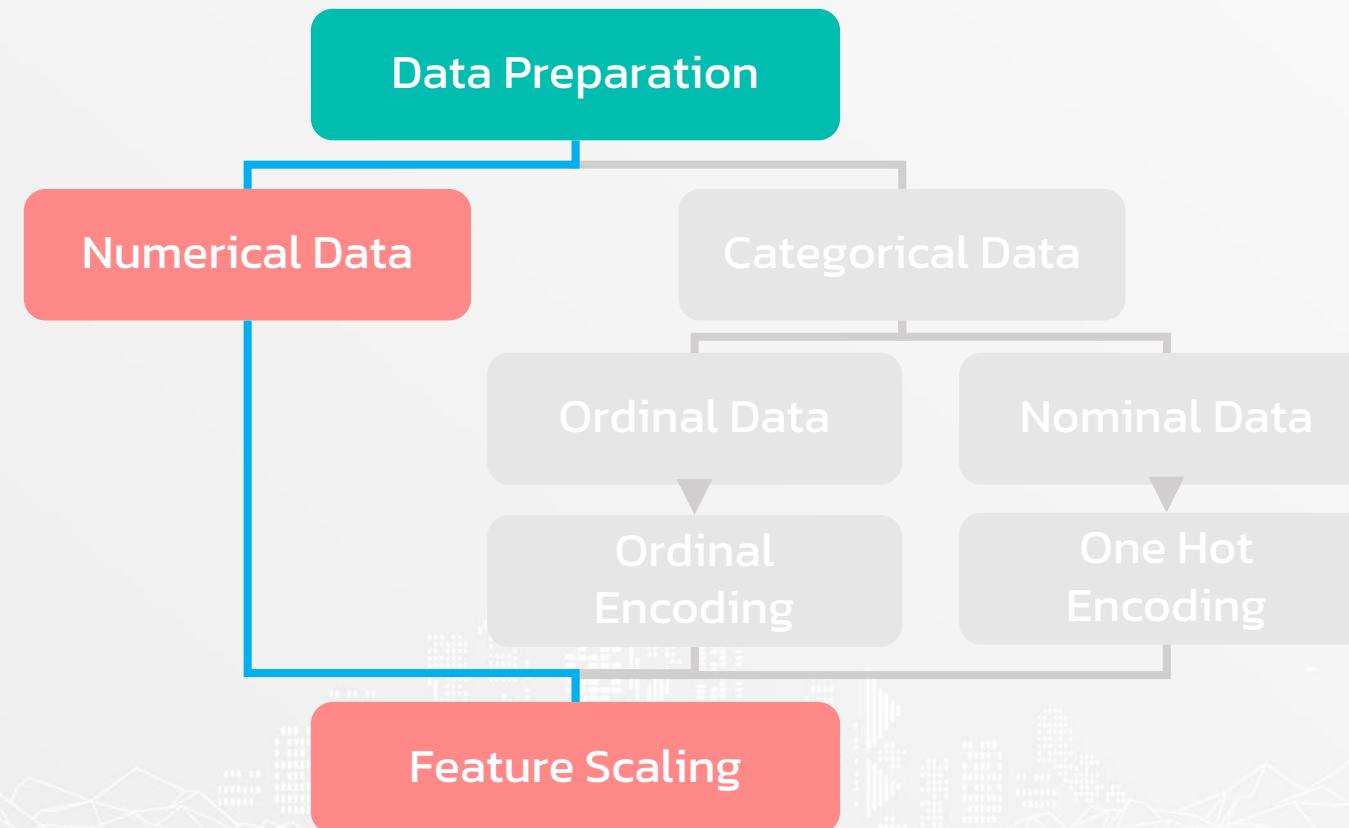
Read Data

```
1 data = pd.read_csv('../image_to_csv/alzheimer_dataset.csv')  
2  
3 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



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

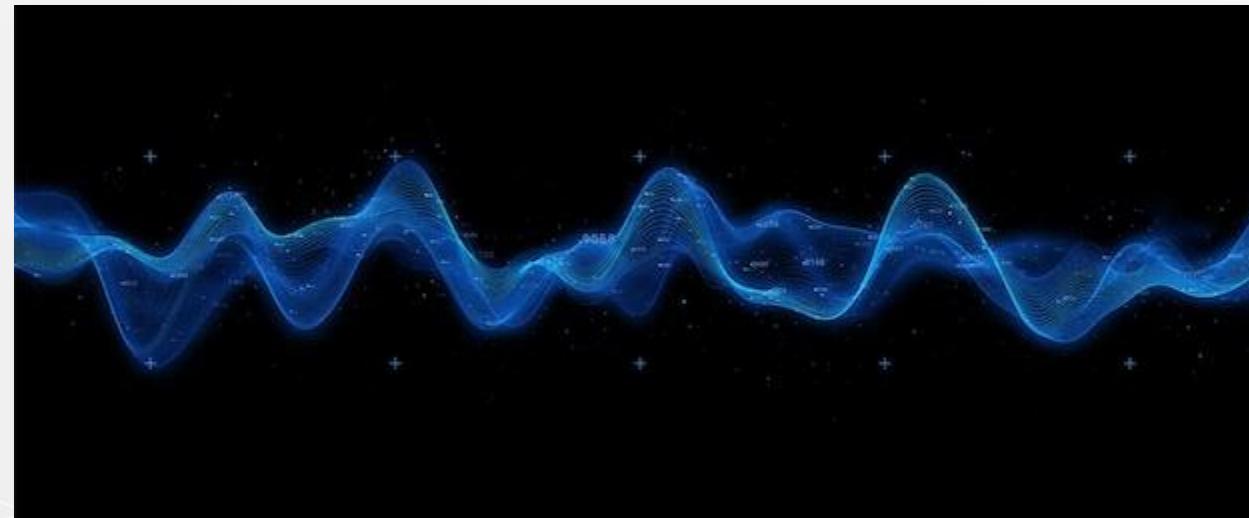


alzheimer_model

AI in Speech Recognition

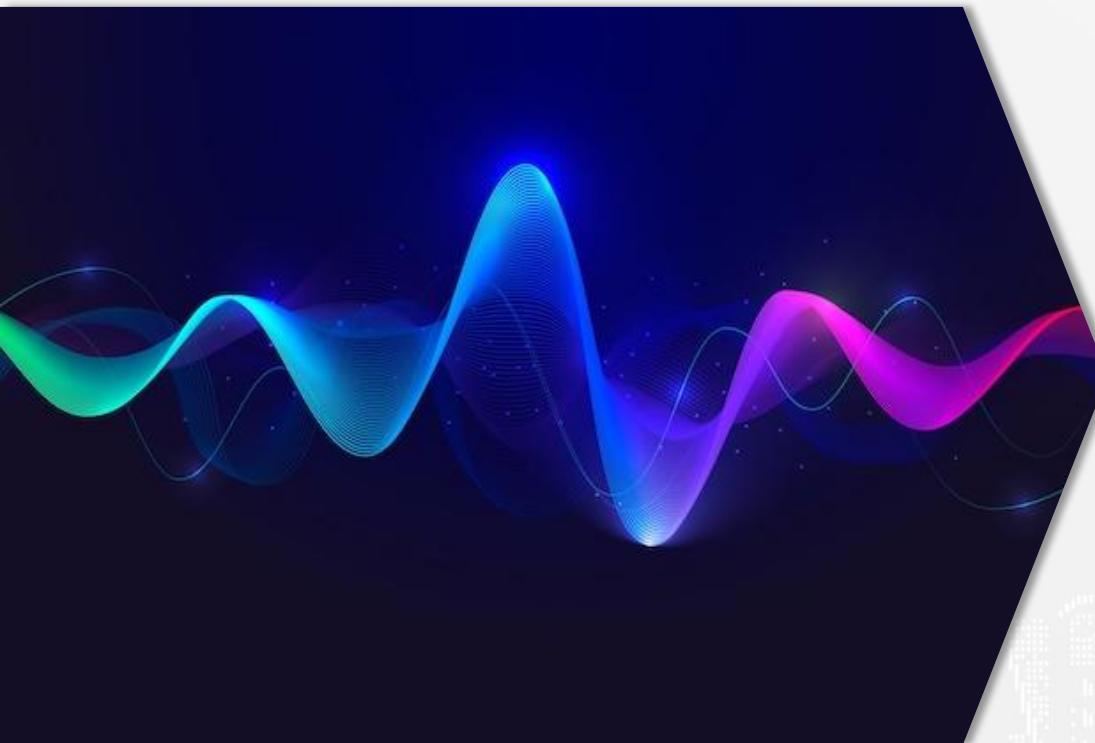
Abstract

สร้าง model เพื่อจำแนกคำจากเสียงพูด โดยพิจารณาจากภาพคลื่นเสียง



Why this project important?

- สามารถสร้างระบบที่ทำงานผ่านคำสั่งเสียง
- สามารถสร้างระบบแปลภาษาจากเสียง
- สามารถสร้างระบบคัดโน๊ตจากดบตระ



Who this project is for?

- ✿ นักพัฒนาหุ่นยนต์
- ✿ นักวิจัยด้านการรู้จำคำพูด
- ✿ นักวิเคราะห์ข้อมูล



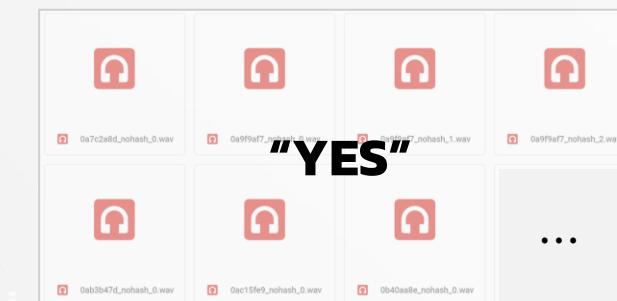
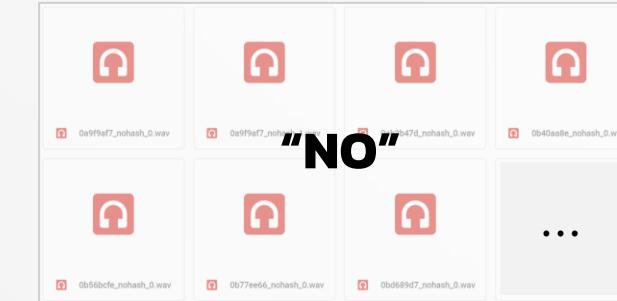
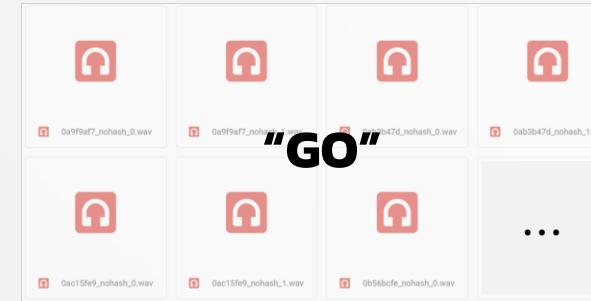
Voice Command Dataset



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

Voice Command Dataset

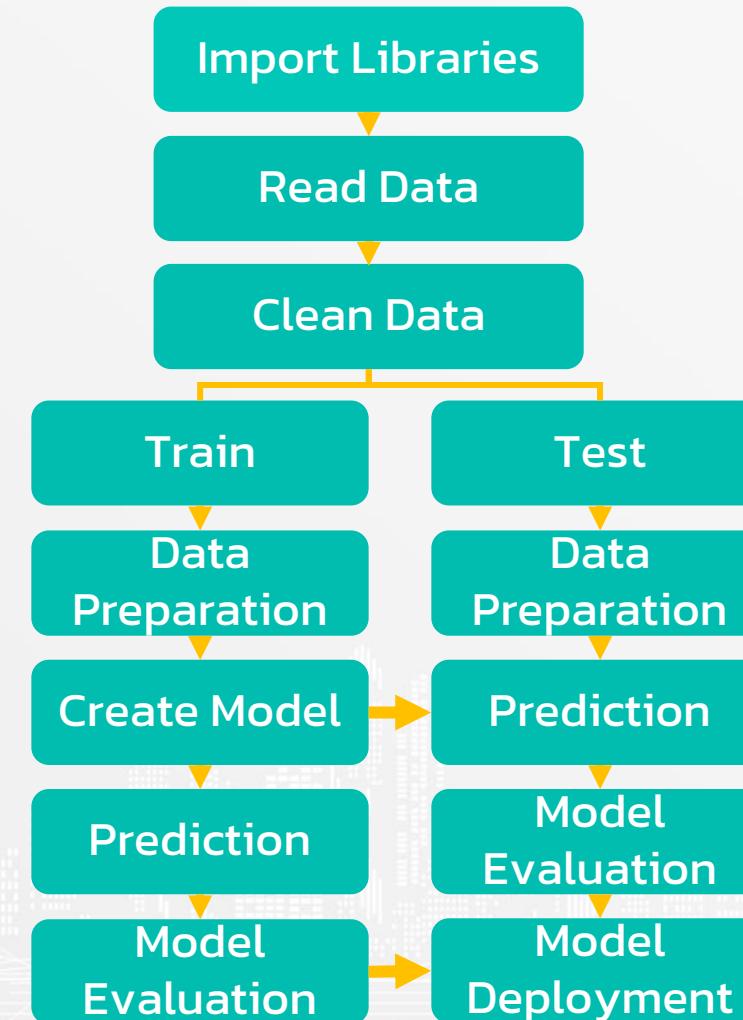
Feature



Target

- target : កំណើន (go, stop, yes, no)

What we learn from this project?



Sound to CSV

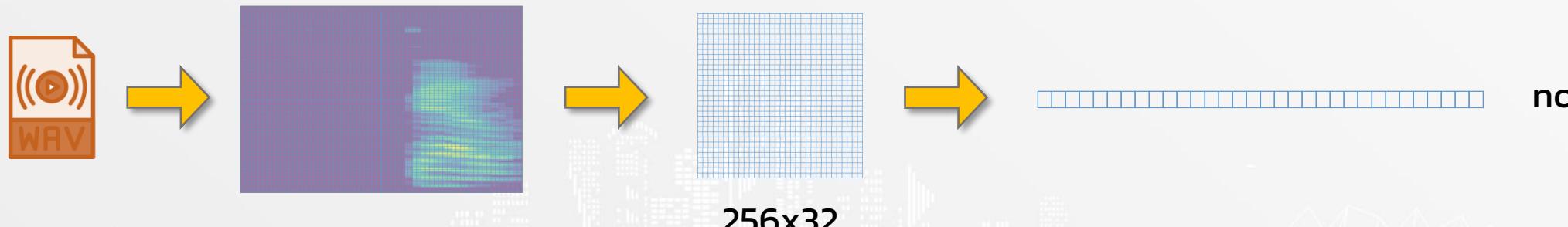
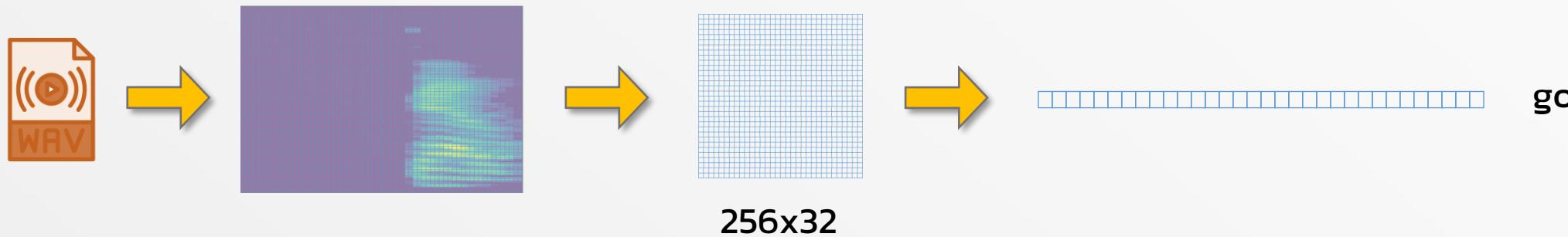
```
1 classes = ['go', 'no', 'stop', 'yes']
```

```
1 mode = 'mel'
2
3 width = 256
4 height = 32
5
6 X = np.empty([0, width*height])
7 y = np.empty([0, 1])
8
9 for _class in tqdm(classes):
10     sound_path = glob('dataset/' + _class + '/*')
11     for path in tqdm(sound_path):
12         voice_data, sampling_rate = librosa.load(path)
13         img = get_img(voice_data, sampling_rate, mode)
14         img = cv2.resize(img, dsize=(width, height))
15         img = img.reshape(1, -1)
16         X = np.vstack([X, img])
17         y = np.vstack([y, _class])
```

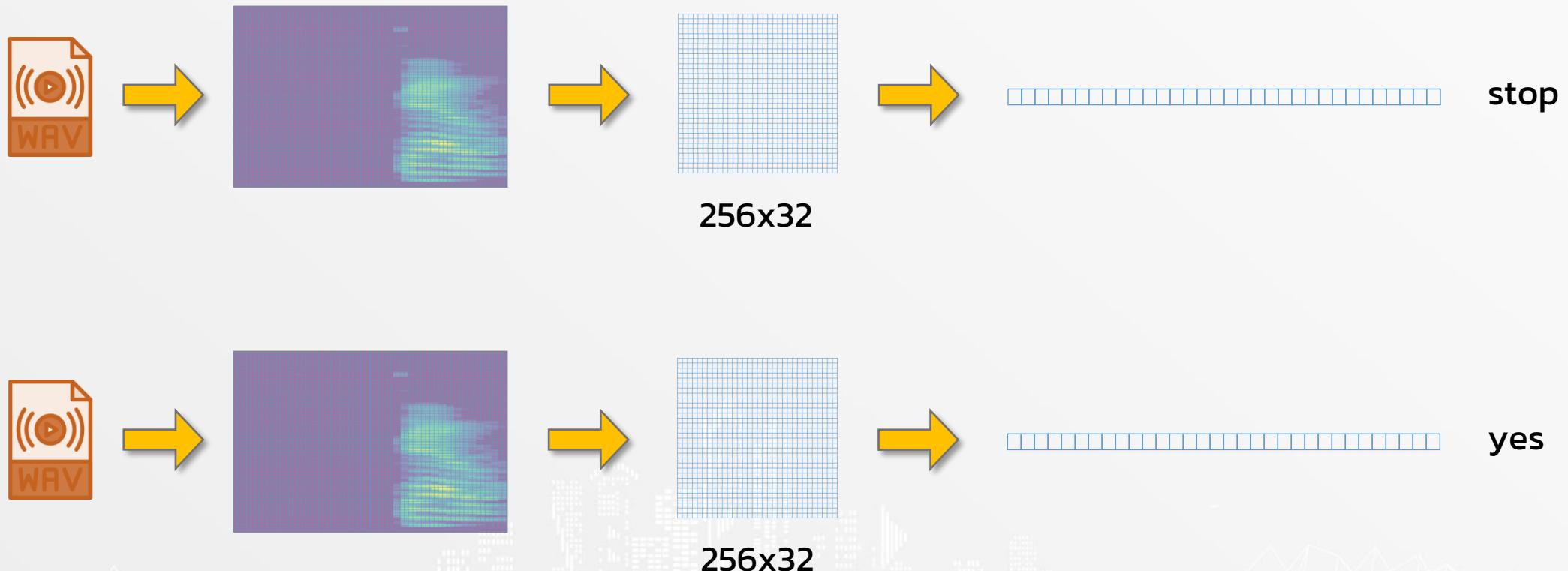
Sound to CSV

```
1 def get_img(voice_data, sampling_data, mode):  
2     if mode == 'spec':  
3         stft = np.abs(librosa.core.spectrum.stft(voice_data))  
4         return librosa.amplitude_to_db(stft, ref=np.max)  
5     elif mode == 'mel':  
6         stft = np.abs(librosa.feature.melspectrogram(voice_data))  
7         return librosa.amplitude_to_db(stft, ref=np.max)  
8     elif mode == 'chrom':  
9         stft = np.abs(librosa.core.spectrum.stft(voice_data))  
10        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	...	x_{8192}	y
-80.0	-80.0	-80.0	...	-80.0	go
-80.0	-80.0	-80.0	...	-80.0	go
-80.0	-80.0	-80.0	...	-80.0	go
:	:	:	:	:	:
-80.0	-80.0	-80.0	...	-80.0	yes

X

y

Sound to CSV

```
1 columns = [f'pixel_{i}' for i in range(width*height)]  
2  
3 data = pd.DataFrame(X, columns=columns)  
4 data['label'] = y  
5  
6 data.to_csv('voice_command_dataset.csv', index=False)
```

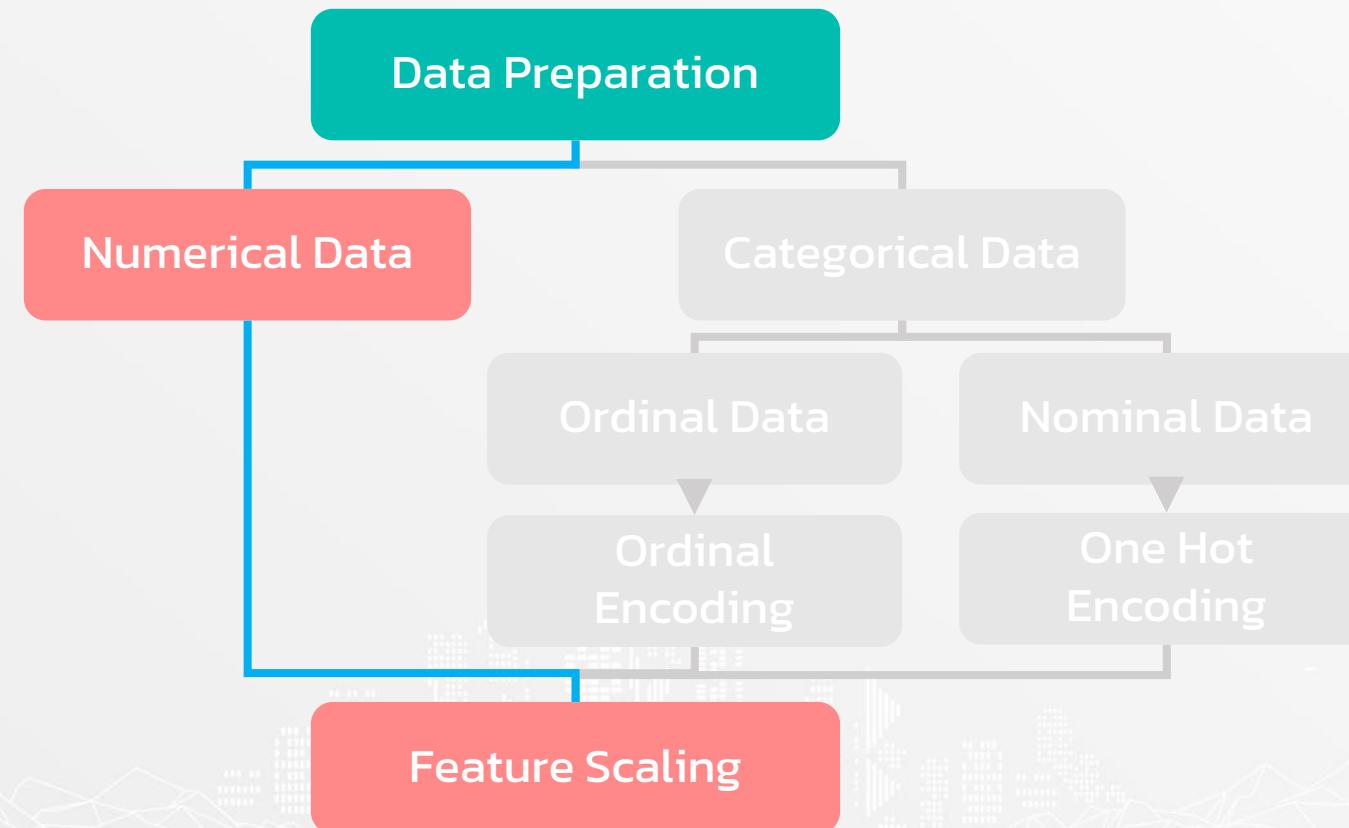
Read Data

```
1 data = pd.read_csv('../sound_to_csv/voice_command_dataset.csv')  
2  
3 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



File



06. VOICE COMMAND/sklearn

-  **voice_command_model.pickle**
-  **voice_command_sklearn_mc.ipynb**
-  **voice_command_sklearn_md.ipynb**

File



06. VOICE COMMAND/keras



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

