ACTIVATION FUNCTION

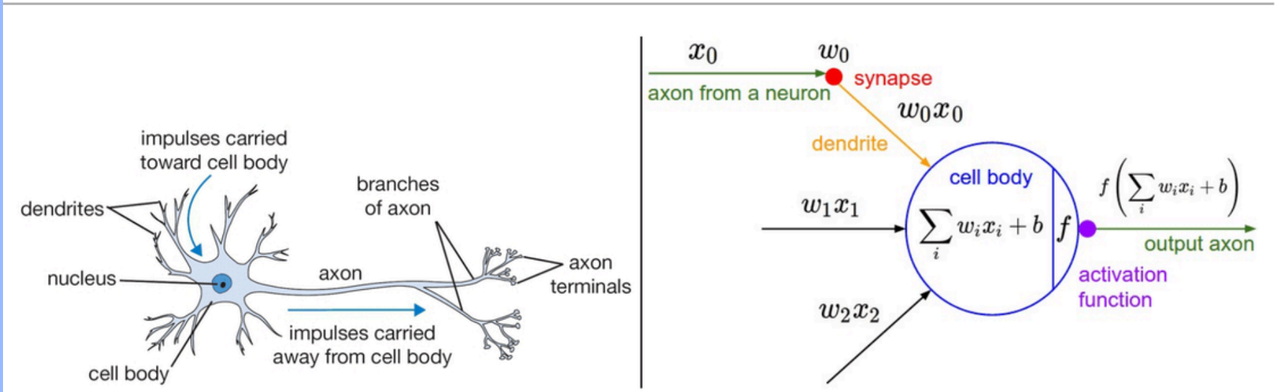
ว30263

การเขียนโปรแกรมปัญญาประดิษฐ์

ACTIVATION FUNCTION

จะประกอบด้วยหน่วยเล็ก ๆ เรียกว่า นิวรอน (Neuron) จำนวนประมาณ 8 หมื่น 6 พันล้านนิวรอน ดังรูปด้านบน และแต่ละนิวรอนก็จะเชื่อมต่อโยงใยกันด้วยเส้นประสาทเรียกว่า ไซแนปส์ (Synapse) รวมแล้วประมาณ 1 พันล้านล้านไซแนปส์ ซึ่งนักวิทยาศาสตร์คอมพิวเตอร์ได้นำมาเป็นแนวคิดในการออกแบบ Artificial Neural Network

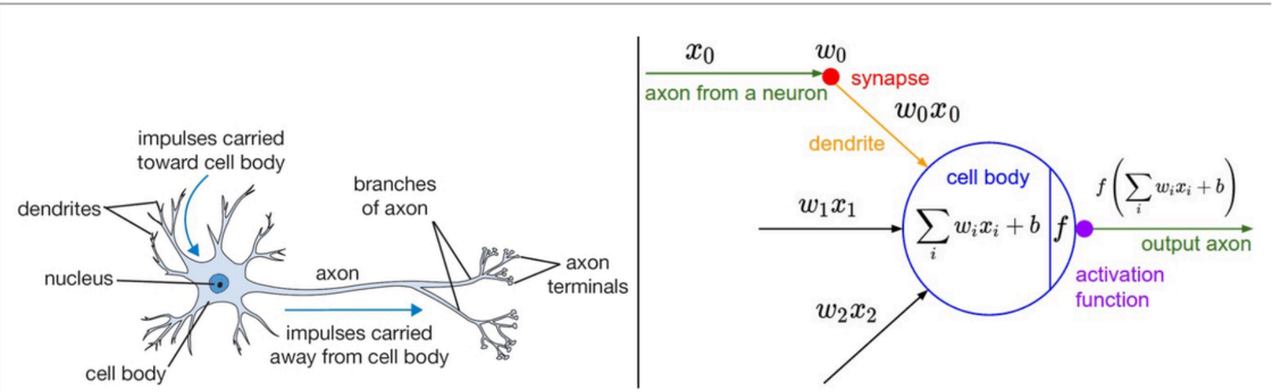
แต่ละนิวรอน จะได้รับ Input หลาย ๆ อัน จากหลาย ๆ กิ่งก้านสาขาของ Dendrite แล้วนำมาประมวลผล ออกมาเป็น Output 1 อัน ออกไปที่ Axon เพื่อส่งต่อไปให้ Dendrite ของนิวรอนอื่น ๆ ใช้เป็น Input ต่อไป เราสามารถเขียนเป็นโมเดล



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

ACTIVATION FUNCTION

Activation Function คือ ฟังก์ชันที่รับผลรวมการประมวลผลทั้งหมด จากทุก Input (ทุก Dendrite) ภายใน 1 นิวรอน แล้วพิจารณาว่าจะส่งต่อเป็น Output เท่าไร (เปรียบเทียบกับความถี่ของสัญญาณประสาท Output ใน Axon) เรื่องระบบการทำงานภายใน Artificial Neural Network จะอธิบายต่อไป Activation Function ที่เป็นที่นิยมในอดีต คือ Sigmoid Function ที่รับข้อมูลอะไรก็ตามเข้าไป เปลี่ยนเป็นค่าระหว่าง 0-1

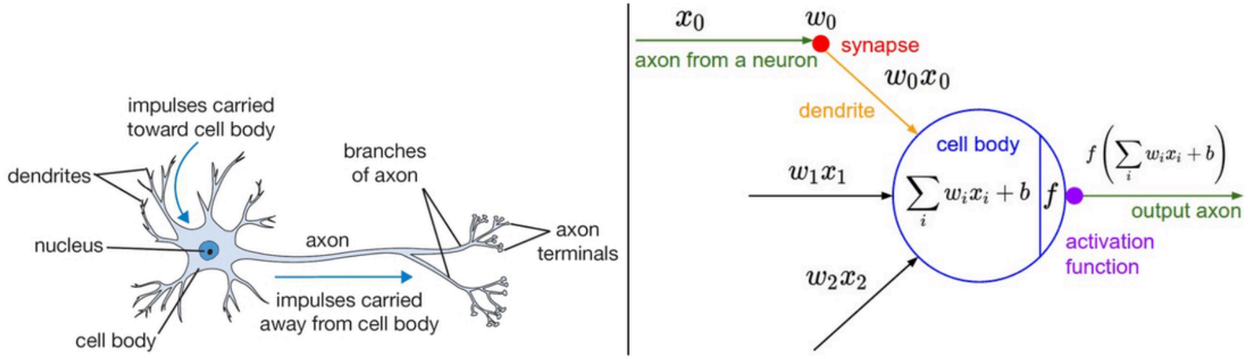


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

ACTIVATION FUNCTION

ประเภทของ Activation Function

1. **Linear Activation Function:**
   - **ส่งผ่านข้อมูลโดยตรง (ค่า y = x)**
   - **ข้อเสีย: ไม่สามารถจัดการกับข้อมูลที่ไม่เป็นเส้นตรงได้**



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

ACTIVATION FUNCTION

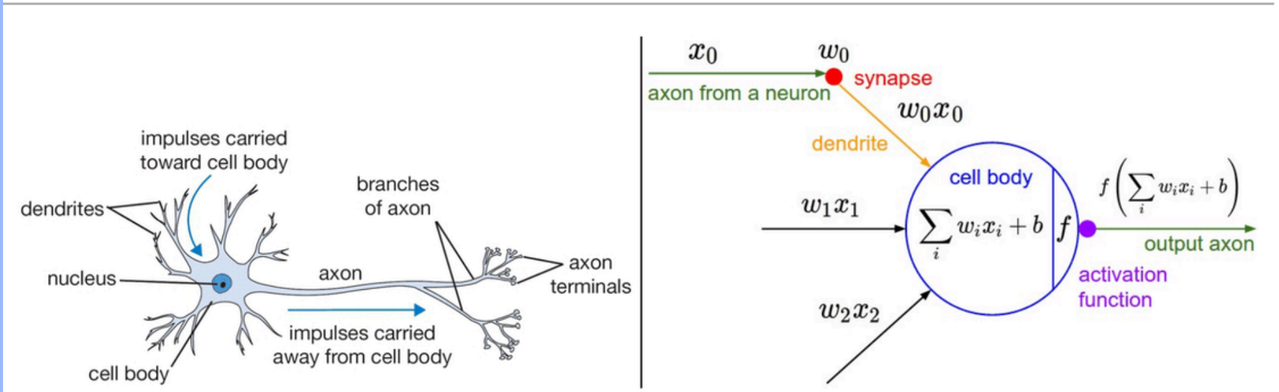
ประเภทของ Activation Function

2.**Non-linear Activation Function:**
ช่วยให้โครงข่ายเรียนรู้ได้ซับซ้อนมากขึ้นและจัดการกับข้อมูลที่ไม่เป็นเส้นตรงได้ดี
ตัวอย่าง:

Sigmoid: มีค่าอยู่ในช่วง 0 ถึง 1 มักใช้ในชั้นสุดท้ายของการจำแนกประเภท (classification)

Tanh: มีค่าอยู่ในช่วง -1 ถึง 1 มีข้อดีคือสามารถจัดการกับค่าลบได้ ทำให้เหมาะกับการใช้งานที่ต้องการค่าในช่วงที่กว้างขึ้น

ReLU (Rectified Linear Unit): มีค่าเป็น 0 เมื่อ x น้อยกว่า 0 และเท่ากับ x เมื่อ x มากกว่า 0 นิยมใช้มากในโครงข่ายที่ลึก (deep network) เนื่องจากช่วยลดปัญหา gradient vanishing



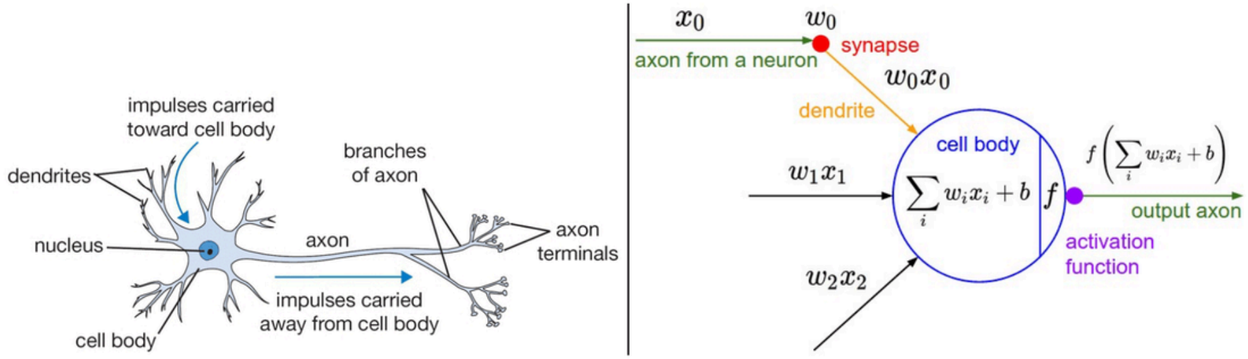
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

ACTIVATION FUNCTION

ประเภทของ Activation Function

3.Advanced Activation Functions:

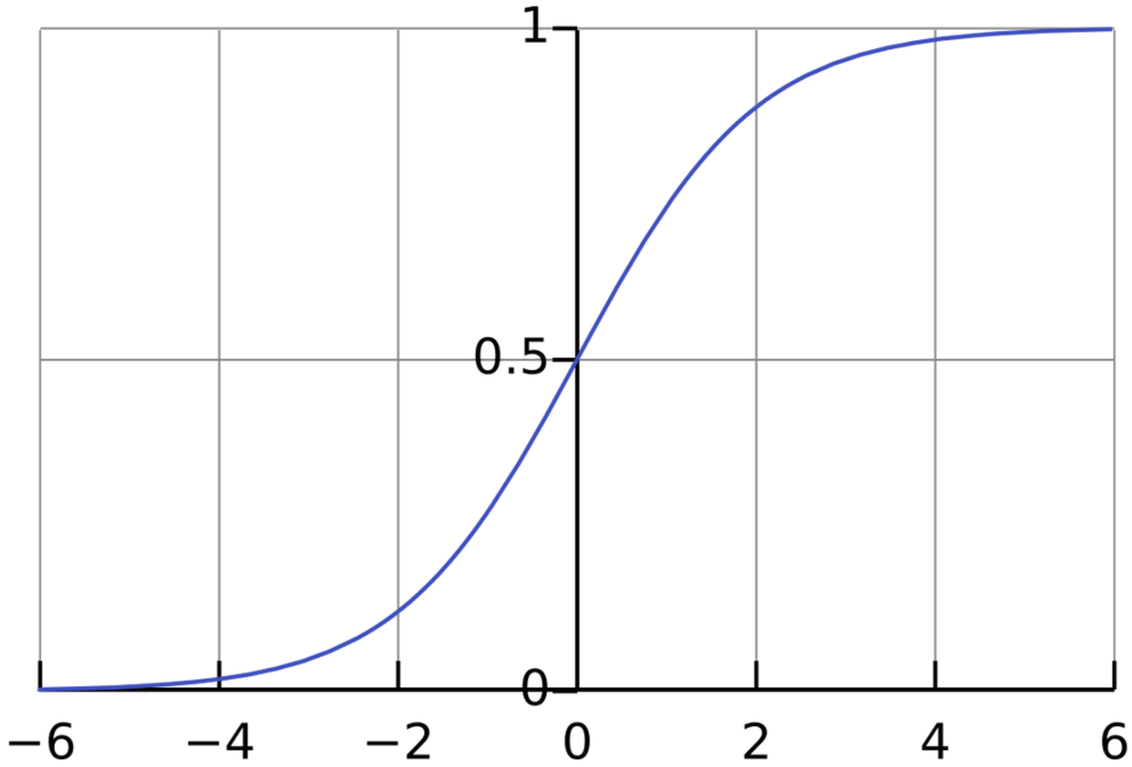
- เช่น **Leaky ReLU**, **Parametric ReLU (PReLU)**, และ **Softmax** ซึ่งใช้ในงานเฉพาะ เช่น การจัดกลุ่มข้อมูล หรือการประมวลผลภาพ



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

SIGMOID

- เหมาะสำหรับ: งานที่ต้องการได้ค่า output ในช่วง 0 ถึง 1 เช่น Binary Classification
- ข้อดี: เหมาะสำหรับปัญหาที่ต้องการค่าความน่าจะเป็น (Probability)
- ข้อเสีย: เมื่อค่า x มากหรือน้อยเกินไป ฟังก์ชันจะค่อยๆ แบนราบ (saturate) ทำให้เกิดปัญหา Vanishing Gradient ซึ่งอาจทำให้การฝึกโครงข่ายใช้เวลานาน

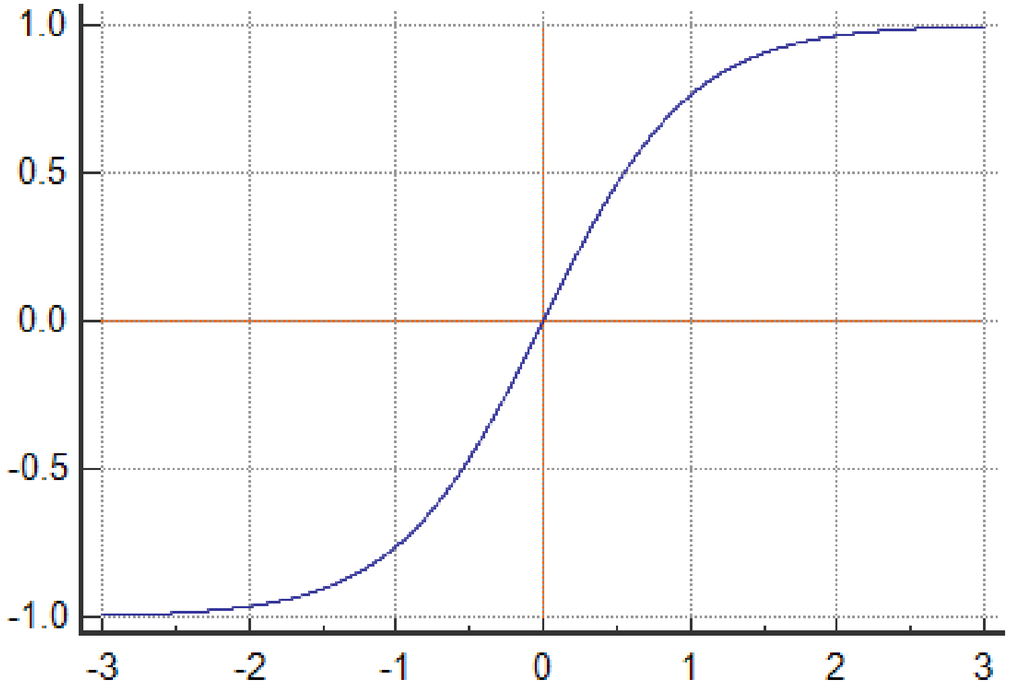



```python
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    s = sigmoid(x)
    return s * (1 - s)
```

TANH

- เหมาะสำหรับ: งานที่ต้องการ output ในช่วง -1 ถึง 1 โดยเฉพาะงานที่ต้องการให้ค่ามีการกระจายตัวในช่วงที่กว้างขึ้นกว่าค่าบวกหรือลบเพียงอย่างเดียว
- ข้อดี: มีค่าอยู่ระหว่าง -1 ถึง 1 ทำให้สามารถจัดการกับข้อมูลที่มีค่าเป็นบวกและลบได้ดี
- ข้อเสีย: มีปัญหา Vanishing Gradient เช่นเดียวกับ Sigmoid แต่มีแนวโน้มที่จะเกิดน้อยกว่า

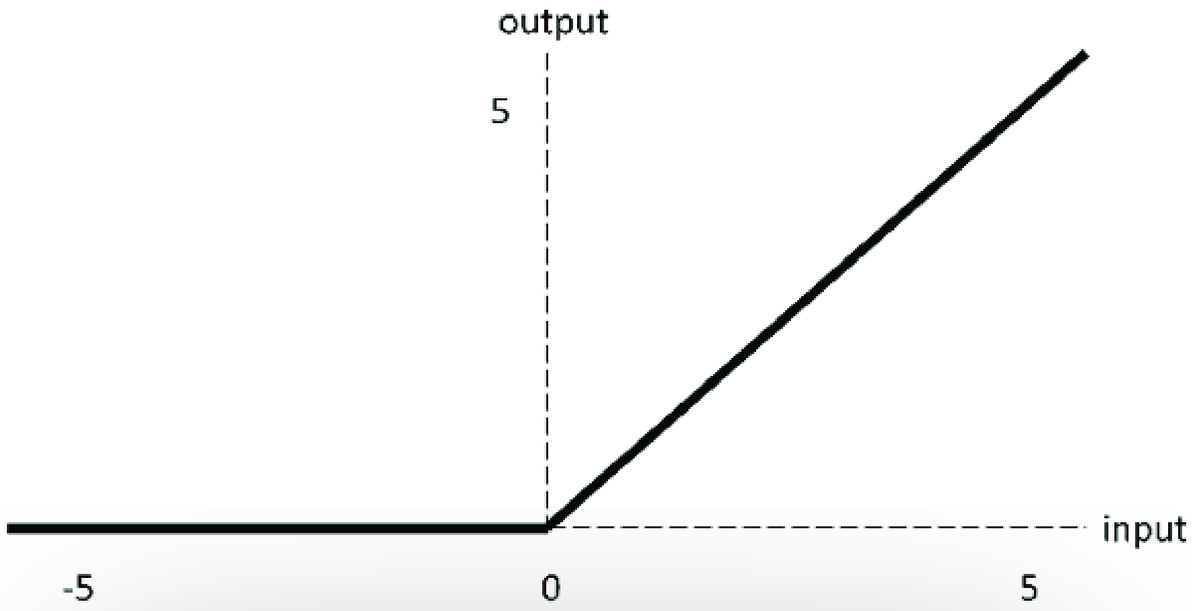



```python
def tanh(x):
    return np.tanh(x)

def tanh_derivative(x):
    return 1 - np.tanh(x)**2
```

RELU (RECTIFIED LINEAR UNIT)

- เหมาะสำหรับ: โครงข่ายประสาทที่มีหลายชั้น (Deep Neural Network) และงานที่ต้องการการคำนวณที่รวดเร็ว เช่น Image Recognition และ Natural Language Processing
- ข้อดี: คำนวณง่าย และช่วยลดปัญหา Vanishing Gradient เนื่องจากมีค่าไม่อิ่มตัว เหมือน Sigmoid และ Tanh
- ข้อเสีย: ค่า gradient เป็น 0 เมื่อติดลบ (Negative Side) ทำให้บางนิวรอนหยุดการเรียนรู้ไปเลย (Dead Neuron)

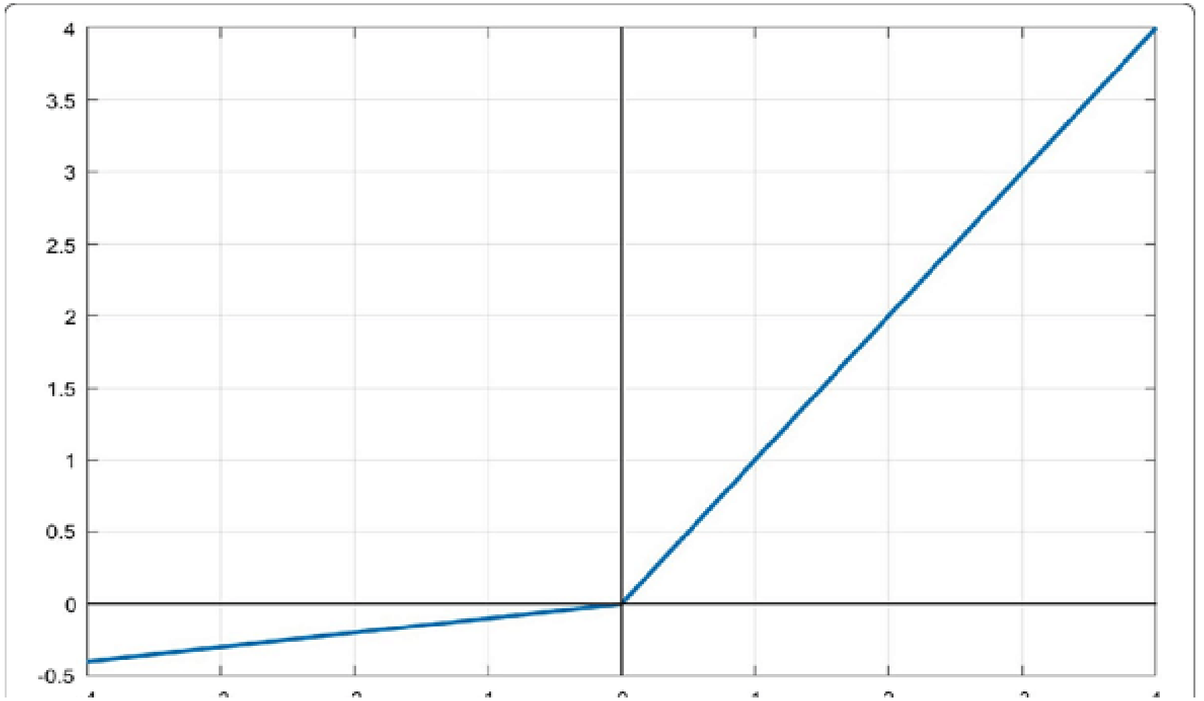



```python
def relu(x):
    return np.maximum(0, x)

def relu_derivative(x):
    return np.where(x > 0, 1, 0)
```

LEAKY RELU

- เหมาะสำหรับ: ใช้ในงานที่ต้องการหลีกเลี่ยงปัญหา Dead Neuron ใน ReLU
- ข้อดี: ให้ gradient ที่ค่าติดลบเล็กน้อย ช่วยแก้ปัญหา Dead Neuron ได้
- ข้อเสีย: ค่า gradient ที่ติดลบอาจทำให้ผลลัพธ์มีการแปรผันมากขึ้นในบางกรณี

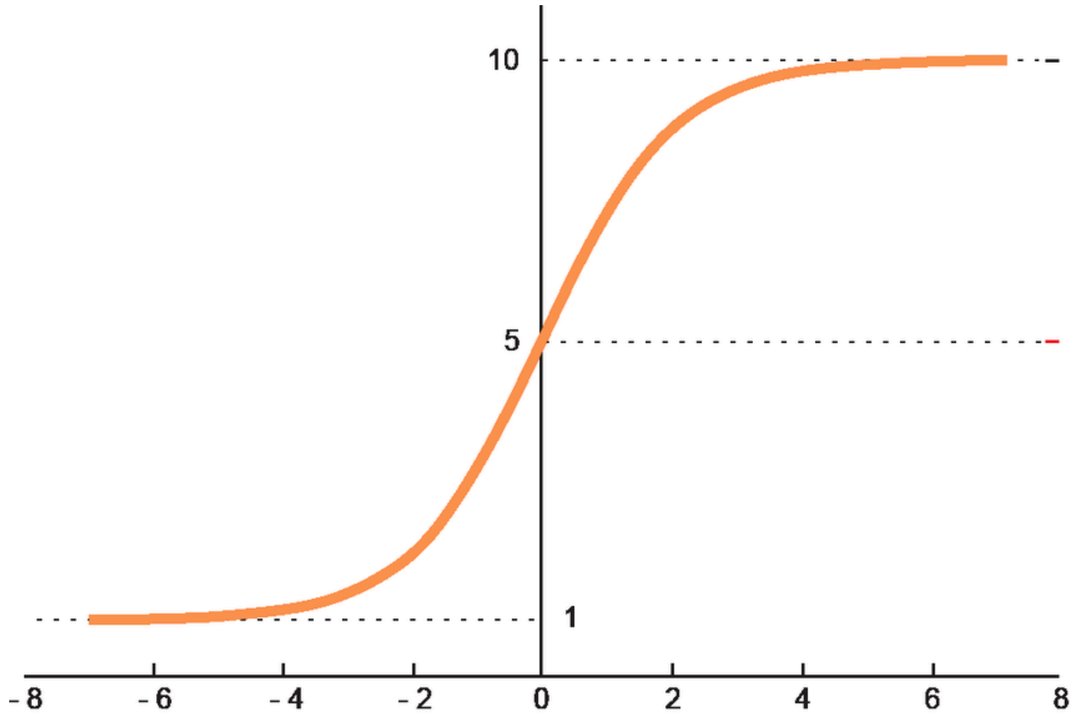



```python
def leaky_relu(x, alpha=0.01):
    return np.where(x > 0, x, alpha * x)

def leaky_relu_derivative(x, alpha=0.01):
    return np.where(x > 0, 1, alpha)
```

SOFTMAX

- เหมาะสำหรับ: ใช้ในชั้นสุดท้ายของโครงข่ายที่มีหลายคลาส (Multi-Class Classification) เช่น Image Classification ที่มีหลายประเภท
- ข้อดี: เปลี่ยนค่า output ให้เป็นความน่าจะเป็นโดยรวมได้ 1 เหมาะสำหรับการเลือกคำตอบที่มีความน่าจะเป็นสูงสุด
- ข้อเสีย: ไม่เหมาะสำหรับชั้นซ่อน (Hidden Layer) เนื่องจากไม่ช่วยแก้ปัญหา gradient แบบฟังก์ชัน Non-linear อื่น ๆ




```python
def softmax(x):
    exp_x = np.exp(x - np.max(x))
    return exp_x / exp_x.sum(axis=0, keepdims=True)

def softmax_derivative(x):
    s = softmax(x).reshape(-1, 1)
    return np.diagflat(s) - np.dot(s, s.T)
```

งาน

- ให้เปรียบเทียบการทำงานของแต่ละ **Activation Function** ใน **hidden layer** โดยเปรียบเทียบ
  - **Accuracy**
  - ประสิทธิภาพ
    - จำนวน **epoch** ดูว่า **Activation Function** ใดต้องการ **epochs** น้อยกว่า ให้ถึงระดับ **accuracy** ที่กำหนด(**Val Accuracy ≥ 85%**)
    - ความเร็วใน ดูว่า **Activation Function** ไหนสามารถทำงานได้เร็วกว่า (**Train:** วินาที/**epoch**)