

Linear Regression

$$\textit{predicted_Y} = w * X + \textit{bias}$$

$$y = (-55)x + 9900 ; x \text{ คือ ปริมาณลูกค้า, } y \text{ คือ กำไร}$$

-55 = weight

9900 = bias

การปรับค่า weight และ Bias เพื่อ optimize สมการนี้ก็ทำได้ด้วยการ differentiation

Ex.

$$\textit{predicted_Y} = w1 * X1 + w2 * x2 \dots\dots\dots Wn * Xn + \textit{bias}$$

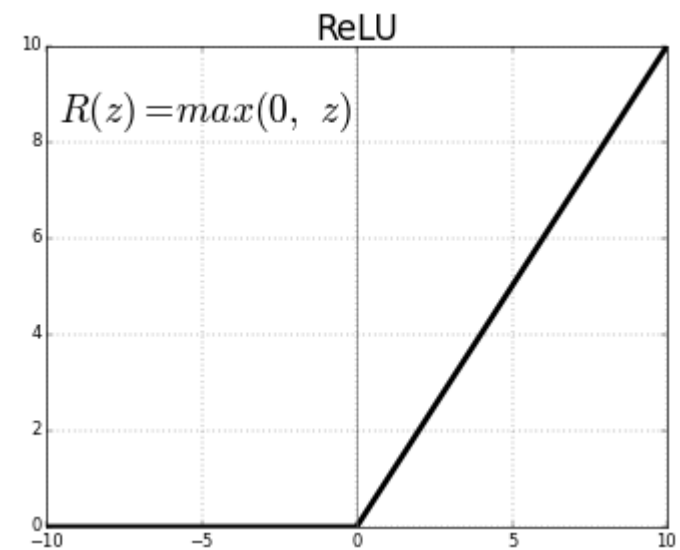
ReLU เพราะว่าได้เปรียบในเชิงแคลคูลัส ทำให้เร็วกว่าแบบอื่น

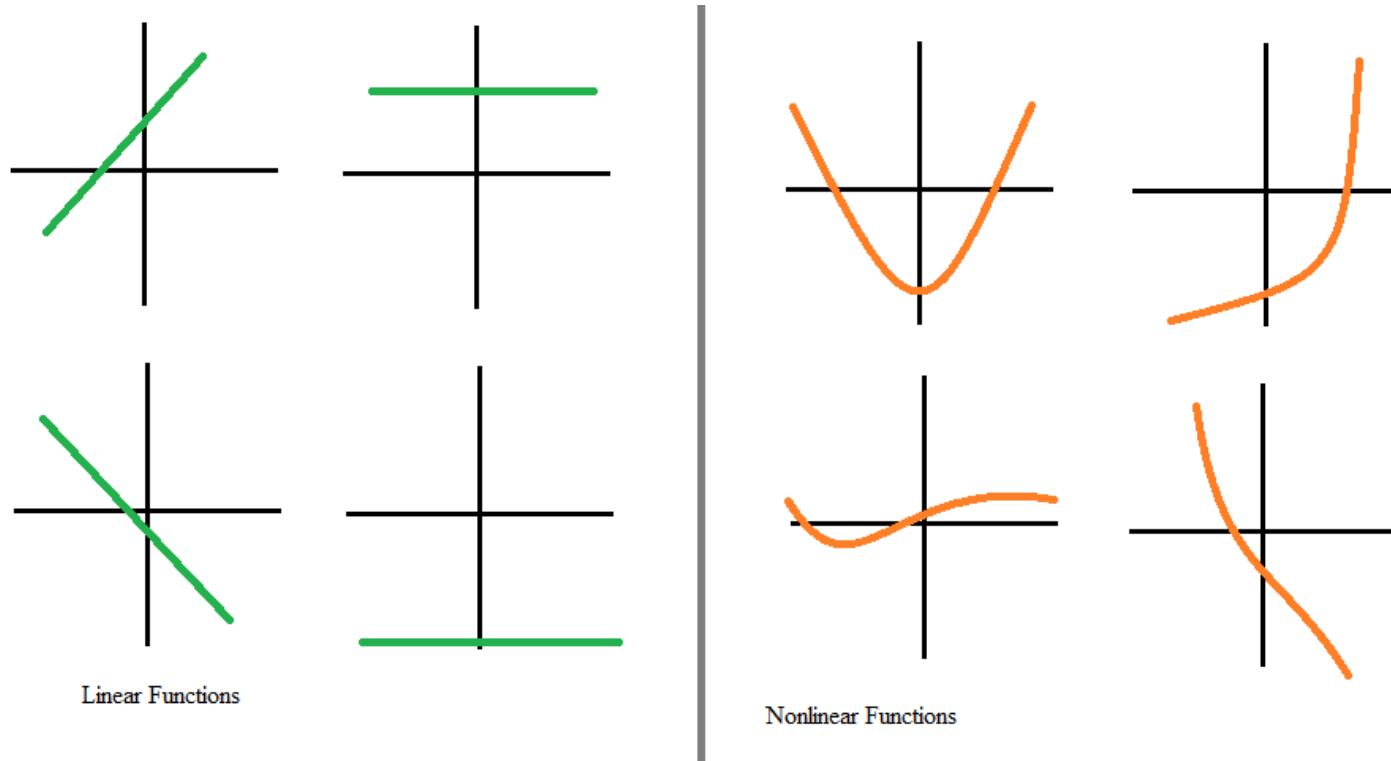
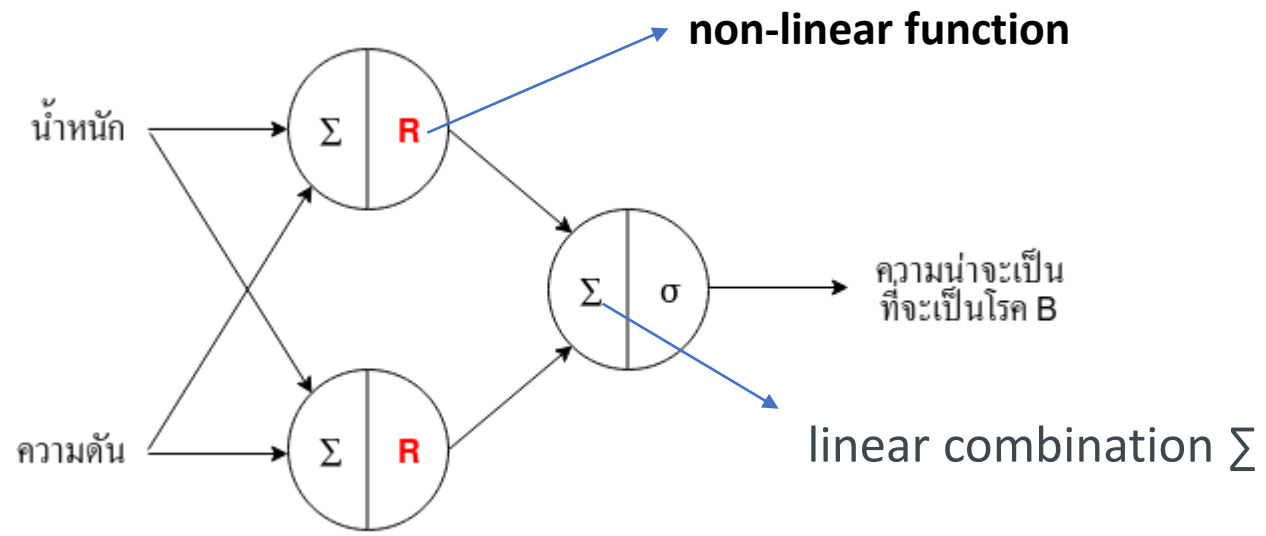
$$\text{relu}(x) = \max(0, x)$$

เมื่อเอาค่าใดๆ เข้าฟังก์ชันนี้แล้ว ถ้าค่านั้นน้อยกว่า 0 ผลลัพธ์จะออกมาเป็น 0 แต่ถ้ามีค่ามากกว่า 0 ผลลัพธ์ก็จะออกมาเป็นค่า \times นั้นๆ เลย เช่น

$$\text{relu}(10) = 10$$

$$\text{relu}(-5) = 0$$





Neural Networks

Color Guided Matrix Multiplication for a Binary Classification Task with N = 4

Input Layer

bias X1 X2

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

4 x 3

Weights w^T
(transposed)

$$\begin{bmatrix} .5 & .5 & .5 \\ .5 & .5 & .5 \\ .5 & .5 & .5 \end{bmatrix}$$

3 x 3

Go to
Hidden Nodes

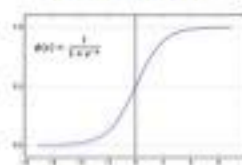
Hidden
Layer

Bias
Node 1
Node 2
Node 3

$$\begin{bmatrix} 1 & 1 & 1 \\ .5 & .5 & .5 \\ .5 & .5 & .5 \\ 1 & 1 & 1 \end{bmatrix}$$

4 x 3

Sigmoid
Function



$$\frac{1}{1 + e^{-(wx+b)}}$$

Weights

$$\begin{bmatrix} .2 & .1 \\ .4 & .1 \\ .4 & .1 \end{bmatrix}$$

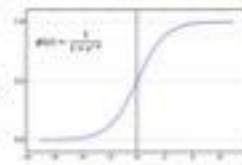
3 x 2

Output
Layer

$$\begin{bmatrix} 1 & .3 \\ .5 & .15 \\ .5 & .15 \\ 1 & .3 \end{bmatrix}$$

4 x 2

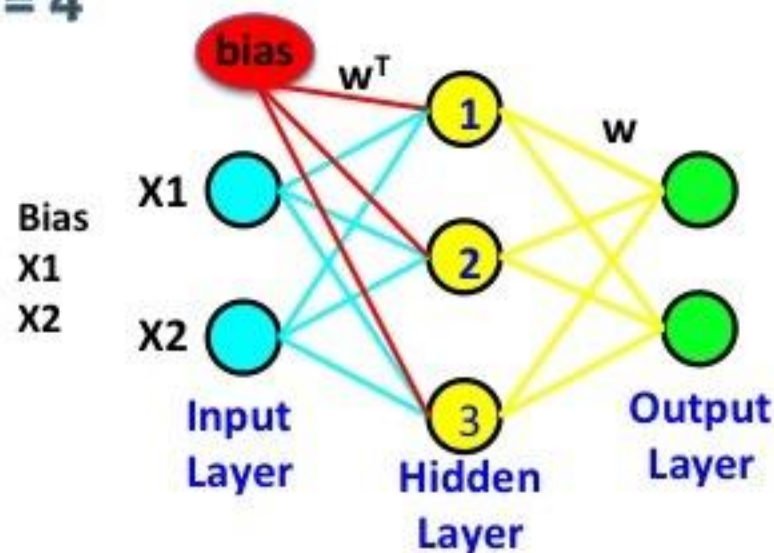
Sigmoid
Function

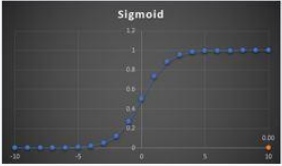
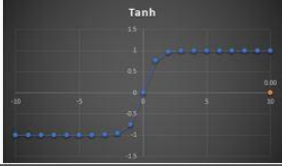
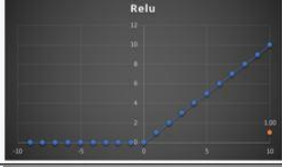
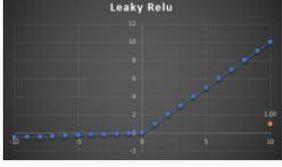



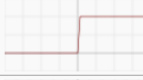
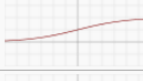
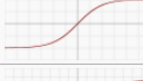

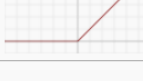



$$\frac{1}{1 + e^{-(wx+b)}}$$

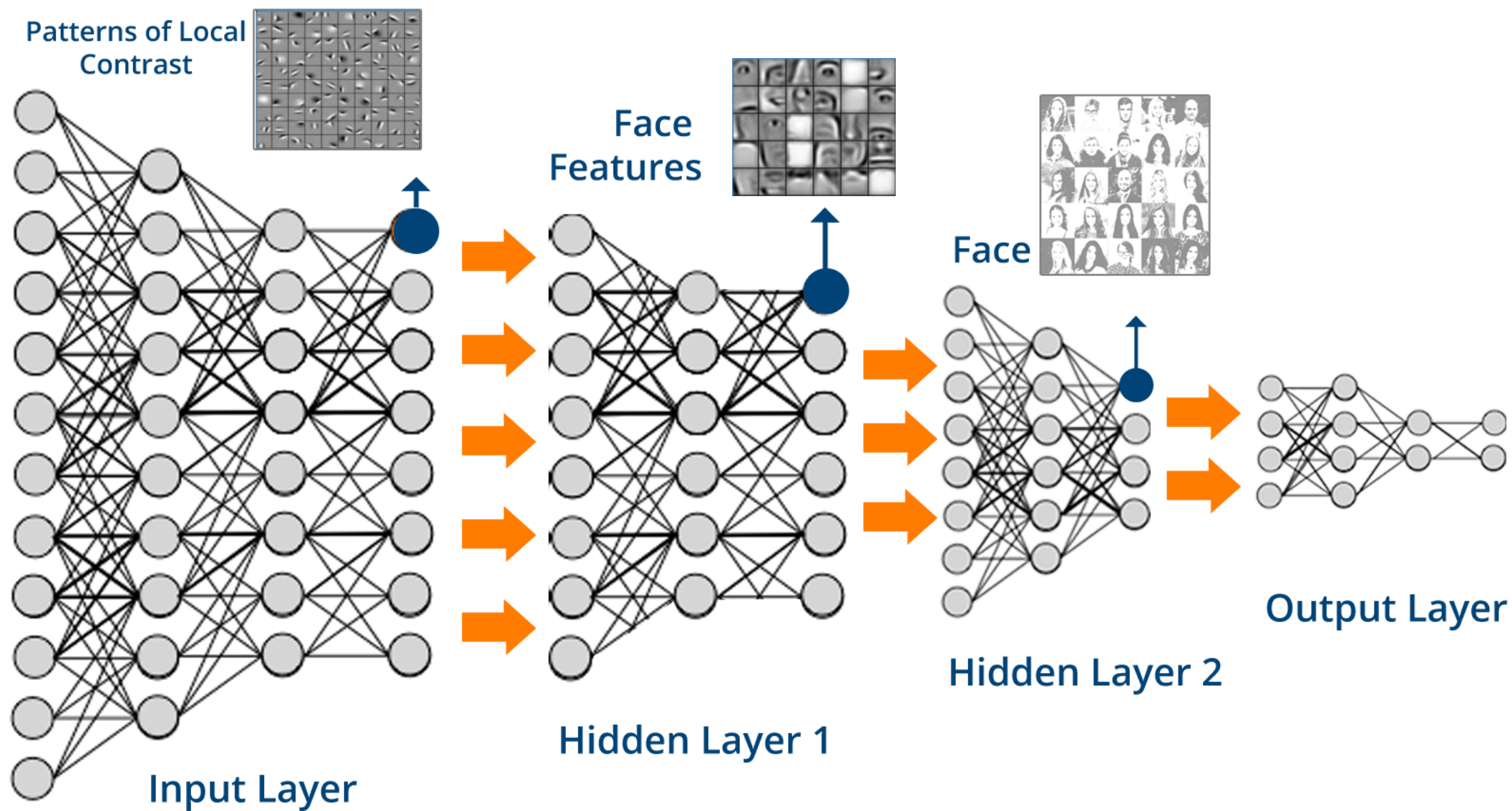
Output

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

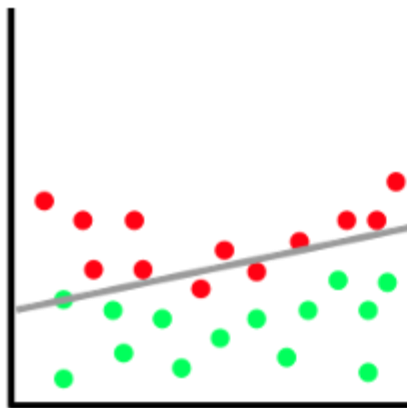


Name	Plot	Equation	Derivative
Sigmoid		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$
Rectified Linear Unit (relu)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky Rectified Linear Unit (Leaky relu)		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

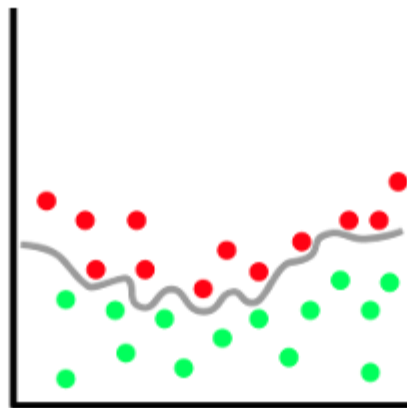
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$



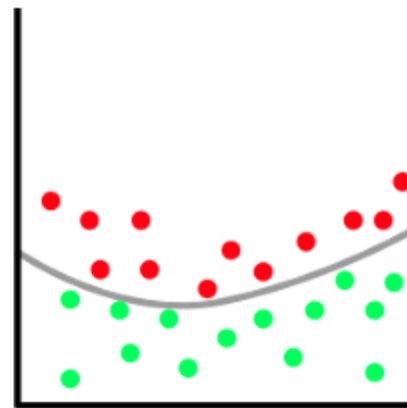
learning & regularization



Underfitting



Overfitting

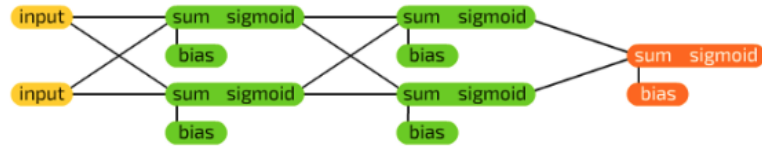


Balanced

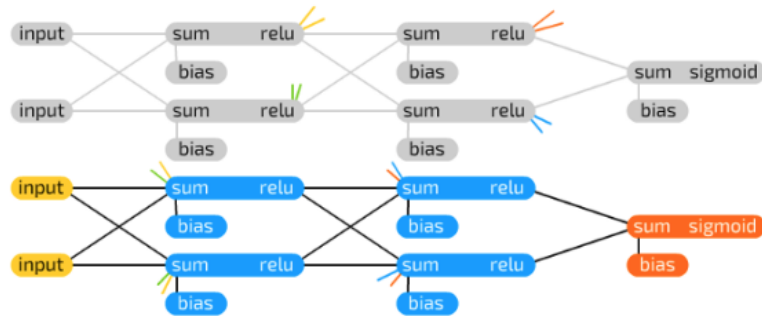
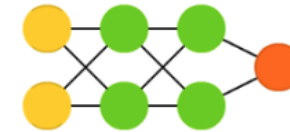
An informative chart to build

Neural Network Graphs

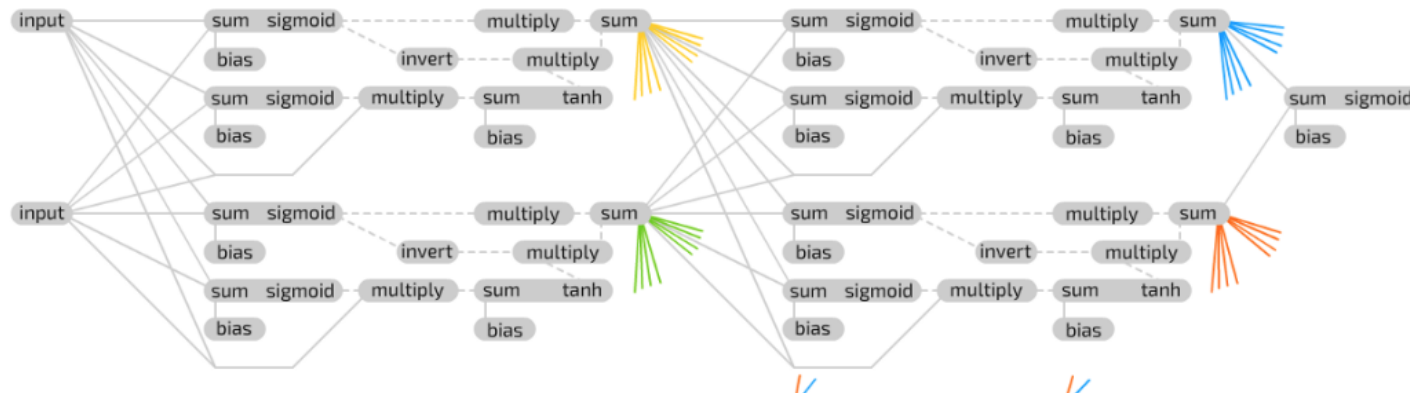
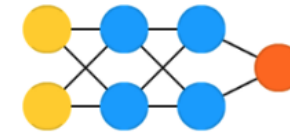
©2016 Fjodor van Veen - asimovinstitute.org



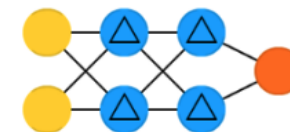
Deep Feed Forward Example



Deep Recurrent Example
(previous iteration)



Deep GRU Example
(previous iteration)



Gradient Descent

วิธีการที่เราจะใช้ปรับ weight และ bias นี้เราใช้วิธีการที่ชื่อว่า Gradient Descent

ให้ model มองชุดข้อมูลหลายๆรอบ แต่ละรอบให้ตรวจสอบว่า model ตอบคำถามของตัวอย่างไหน ผิดบ้าง และปรับปรุง model โดยดูจากตัวอย่างที่ตอบผิด โดยหวังว่ารอบถัดๆไปจะตอบผิบน้อยลงเรื่อยๆ

(Gradient descent เป็นแก่นของการแก้เพื่อหาค่าที่เหมาะสมที่สุดให้กับฟังก์ชัน)

$$y = (-55)x + 9900 ; x \text{ คือ ปริมาณลูกค้า, } y \text{ คือ กำไร}$$

openCV

open source library สำหรับทำงานแนว image processing ต่างๆ มี interface อยู่หลายภาษา ให้เราเลือกใช้ได้ (C++/Java/Python) เราสามารถนำเอา OpenCV ไปประยุกต์ใช้งานในด้าน computer vision

back propagation

Σ

Linear combination ของ input กับ parameter แสดงด้วยสัญลักษณ์ Σ