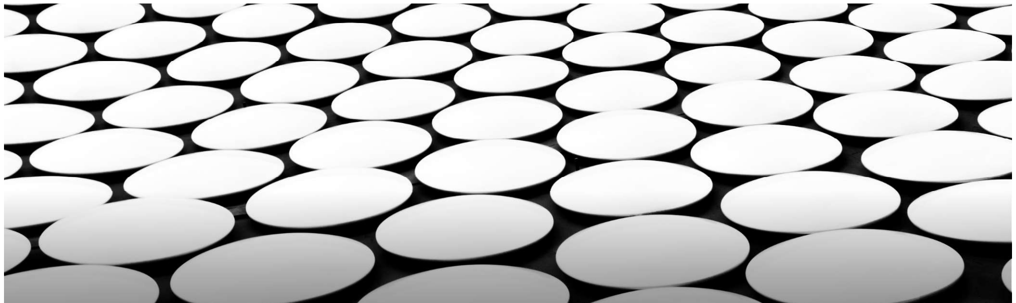


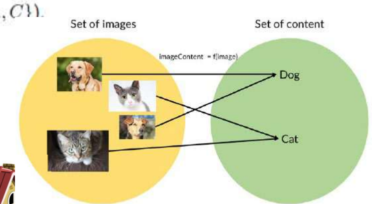
Chapter 10 ANNs
Artificial Neural Networks

Associate Professor Yachai Limpiyakorn Ph.D.

Neural Network (NN)

- Consuming time & resources during learning phase
- Black box model (not human understandable)
- Classification or Regression

Classification
Output y belongs to one of C predetermined classes (i.e. $y \in \{1, ..., C\}$).
For example, determine whether the picture shows a cat or a dog.



Regression
Output y is continuous (i.e. $y \in \mathbb{R}$).
For example, predict the price of a house given its area.

Approximating functions

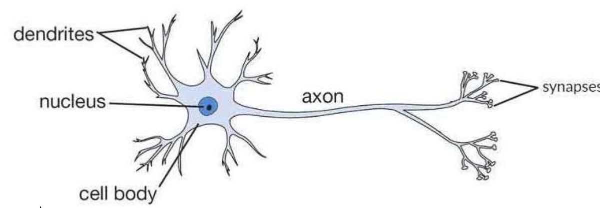
- Think about neural networks as function approximators
  y= f(x) *or* *y=f(X)* ; where input vector *X*
- Given dataset as below, the function want to approximate is

X_1	X_2	X_3	y
0	1	0	0
1	0	0	1
1	1	1	1
0	1	1	0

$$f(X) = X_1$$

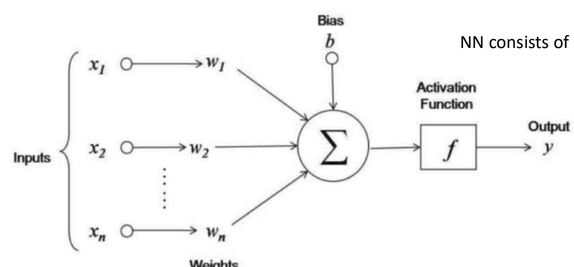
- Functions expressed by neural networks can become very complex, and it is not possible to write down the function.
  imageContent = f(image) ; where image stored as matrix of numbers
- A neural network, if it is big enough, can approximate any function.

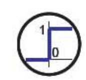
Biological Neuron



Perceptron

NN consists of atomic units called *Perceptron* ≈ human *Neuron*





Binary Activation Function

Bipolar Activation Function

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + \cdots + w_nx_n > \theta \\ -1 & \text{if } w_1x_1 + w_2x_2 + \cdots + w_nx_n < \theta \end{cases} \quad \text{(สูตรที่1)}$$

จากฟังก์ชันในสูตรที่ 1 เราจัดรูปใหม่โดยย้าย θ ไปรวมกับผลรวมเชิงเส้นแล้ว แทน $-\theta$ ด้วย w_0 ($-w_0$ คือ ค่าขีดแบ่ง θ) เราจะได้ฟังก์ชันของเอาท์พุตต่อไปนี้

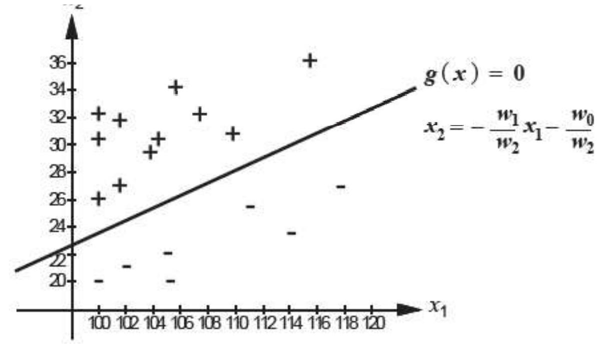
$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n < 0 \end{cases} \quad \text{(สูตรที่2)}$$

กำหนดให้ $$g(\vec{x}) = \sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x}$$

โดยที่ $\vec{x}$ แทนเวกเตอร์อินพุต เราสามารถเขียนฟังก์ชันของเอาท์พุตได้ใหม่ดังนี้

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } g(\vec{x}) > 0 \\ -1 & \text{if } g(\vec{x}) < 0 \end{cases} \quad \text{(สูตรที่3)}$$

เพอร์เซปตรอนเป็นระนาบตัดสินใจหลายมิติ (hyperplane decision surface) ในกรณีที่มีอินพุต 2 ตัว (ไม่รวม x_0) เราจะได้ $g(\vec{x}) = w_0 + w_1x_1 + w_2x_2$ ซึ่งถ้าเราให้ $g(\vec{x}) = 0$ จะได้ว่า $w_0 + w_1x_1 + w_2x_2 = 0$ ซึ่งแทนสมการเส้นตรงในระนาบสองมิติ



Perceptron

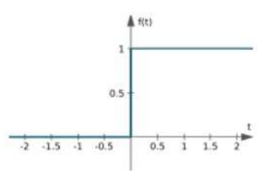
Perceptron algorithm is one of the oldest methods for binary classification.

Decision rule

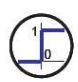
$$\hat{y} = f(\boldsymbol{w}^T \boldsymbol{x} + w_0)$$

where f is the step function defined as:

$$f(t) = \begin{cases} 1 \text{ if } t > 0, \\ 0 \text{ else.} \end{cases}$$



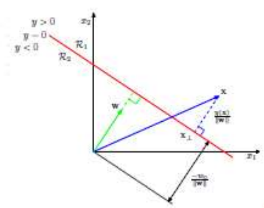
Bipolar Activation Function



Binary Activation Function

Hyperplane as a decision boundary

For a 2 class problem ($\mathcal{C} = \{0, 1\}$) we can try to separate points from the two classes by a hyperplane.



A hyperplane be defined by a normal vector $\boldsymbol{w}$ and an offset w_0.

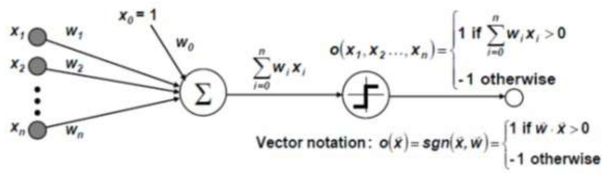
$$\boldsymbol{w}^T\boldsymbol{x} + w_0 \begin{cases} = 0 & \text{if } \boldsymbol{x} \text{ on the plane} \\ > 0 & \text{if } \boldsymbol{x} \text{ on normal's side} \\ < 0 & \text{else} \end{cases}$$

$$f_{\boldsymbol{w}}(\boldsymbol{x}_i) = w_0 + w_1x_{i1} + w_2x_{i2} + \ldots + w_Dx_{iD}$$

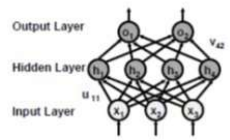
Hyperplanes are computationally very convenient: easy to evaluate.

A data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ is linearly separable if there exists a hyperplane for which all $\boldsymbol{x}_i$ with $y_i = 0$ are on one and all $\boldsymbol{x}_i$ with $y_i = 1$ on the other side.

Perceptron vs. Multi-Layer Perceptron (MLP)



- Single Neuron Model
  - Linear Threshold Unit (LTU)
    inputs to unit: defined as linear combination
  - Output of unit: threshold (activation) function on net input (threshold $\theta = w_0$)
- Neural Networks
  - Neuron is modeled using a unit connected by weighted links w_i to other units
  - Multi-Layer Perceptron (MLP): future lecture



Perceptron Training

- เริ่มจากการสุ่มค่าน้ำหนัก W_i
- สอนเพอร์เซปตรอนกับทุกตัวอย่างที่สอนทีละตัว โดย
  - คำนวณผลรวมเชิงเส้นของอินพุตทุกตัว
  - คำนวณหาค่าเอาท์พุตจากฟังก์ชันกระตุ้น
  - คำนวณหาค่าความผิดพลาดระหว่างเอาท์พุตที่ได้จากฟังก์ชันกระตุ้นและเอาท์พุตที่แท้จริง
  - แก้ไขน้ำหนักเมื่อเพอร์เซปตรอนแยกตัวอย่างผิดพลาด
- วนทำซ้ำกับตัวอย่างที่สอน จนกระทั่งเพอร์เซปตรอนแยกตัวอย่างได้ถูกต้องทั้งหมด

Weight Update

$$W_i \leftarrow W_i + \Delta W_i$$

โดยที่ $$\Delta W_i = \eta (t-o) x_i$$

เมื่อ t เป็นเอาท์พุตเป้าหมาย หรือผลลัพธ์ที่ถูกต้อง

o เป็นเอาท์พุตที่แท้จริง หรือผลลัพธ์ที่ได้จากเพอร์เซปตรอน

(t-o) คือ ค่าผิดพลาด

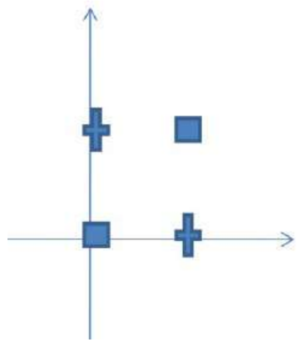
η เป็นค่าที่แสดงอัตราการเรียนรู้ (learning rate) เป็นค่าคงที่บวกจำนวนน้อยๆ

- กรณี t=1, o=-1 (ใช้ฟังก์ชันกระตุ้นสองขั้ว) หมายความว่า เพอร์เซปตรอนให้ผลรวมเชิงเส้นน้อยเกินไปและน้อยกว่า 0 น้ำหนักจึงต้องถูกปรับให้สามารถเพิ่มค่า $\sum w_i x_i$ เพื่อที่จะทำให้เพอร์เซปตรอนให้ผลลัพธ์ค่า 1 กล่าวคือ
  - W_i ของ x_i ที่เป็นค่าบวกจะถูกปรับเพิ่มขึ้น
  - W_i ของ x_i ที่เป็นค่าลบจะถูกปรับลดลง
- ในทางตรงกันข้าม เมื่อ t=-1, o=1 เพื่อให้การปรับเป็นไปในทิศทางที่ถูกต้อง
  - W_i ของ x_i ที่เป็นค่าลบจะถูกปรับเพิ่มขึ้น
  - W_i ของ x_i ที่เป็นค่าบวกจะถูกปรับลดลง

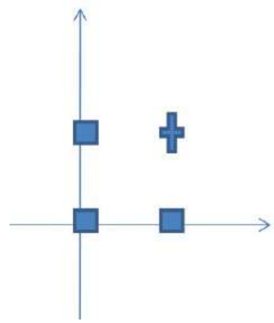
Choice of Learning Rate

Learning rate size	Advantages/disadvantages
Smaller learning rate	Converges slower but more accurate results
Larger learning rate	Less accurate, but converges faster

Linearly Nonseparable XOR Function



Linearly Separable AND Function



Perceptron Learning of Function AND Using Binary Activation Function

Bias Input x0 = +1 Learning Rate (η) = 0.7

Input X1	Input X2	1.0*W0	X1*W1	X2*W2	Net Sum Input	Target Output (t)	Actual Output (o)	η(t-o)	Weight Values ΔW0	ΔW1	ΔW2
									0.1	0.1	0.1
0	0	0.1	0	0	0.1	0	1	-0.7	-0.7	0	0
0	1	0.1	0	0.1	0.2	0	1	-0.7	-0.7	0	-0.7
1	0	0.1	0.1	0	0.2	0	1	-0.7	-0.7	-0.7	0
1	1	0.1	0.1	0.1	0.3	1	1	0	0	0	0
Update W_i									-2	-0.6	-0.6
0	0	-2	0	0	-2	0	0	0	0	0	0
0	1	-2	0	-0.6	-2.6	0	0	0	0	0	0
1	0	-2	-0.6	0	-2.6	0	0	0	0	0	0
1	1	-2	-0.6	-0.6	-3.2	1	0	0.7	0.7	0.7	0.7
Update W_i									-1.3	0.1	0.1
0	0	-1.3	0	0	-1.3	0	0	0	0	0	0
0	1	-1.3	0	0.1	-1.2	0	0	0	0	0	0
1	0	-1.3	0.1	0	-1.2	0	0	0	0	0	0
1	1	-1.3	0.1	0.1	-1.1	1	0	0.7	0.7	0.7	0.7
Update W_i									-0.6	0.8	0.8
0	0	-0.6	0	0	-0.6	0	0	0	0	0	0
0	1	-0.6	0	0.8	0.2	0	1	-0.7	-0.7	0	-0.7
1	0	-0.6	0.8	0	0.2	0	1	-0.7	-0.7	-0.7	0
1	1	-0.6	0.8	0.8	1	1	1	0	0	0	0
Update W_i									-2	0.1	0.1
0	0	-2	0	0	-2	0	0	0	0	0	0
0	1	-2	0	0.1	-1.9	0	0	0	0	0	0
1	0	-2	0.1	0	-1.9	0	0	0	0	0	0
1	1	-2	0.1	0.1	-1.8	1	0	0.7	0.7	0.7	0.7
Update W_i									-1.3	0.8	0.8
0	0	-1.3	0	0	-1.3	0	0	0	0	0	0
0	1	-1.3	0	0.8	-0.5	0	0	0	0	0	0
1	0	-1.3	0.8	0	-0.5	0	0	0	0	0	0
1	1	-1.3	0.8	0.8	0.3	1	1	0	0	0	0
Update W_i									-1.3	0.8	0.8

Perceptron Learning of Function XOR Using Binary Activation Function

Bias Input x0 = +1 Learning Rate (η) = 0.7

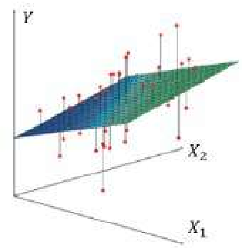
Input X1	Input X2	1.0*W0	X1*W1	X2*W2	Net Sum Input	Target Output (t)	Actual Output (o)	η(t-o)	Weight Values ΔW0	ΔW1	ΔW2
									0.1	0.1	0.1
0	0	0.1	0	0	0.1	0	1	-0.7	-0.7	0	0
0	1	0.1	0	0.1	0.2	1	1	0	0	0	0
1	0	0.1	0.1	0	0.2	1	1	0	0	0	0
1	1	0.1	0.1	0.1	0.3	0	1	-0.7	-0.7	-0.7	-0.7
Update W_i									-1.3	-0.6	-0.6
0	0	-1.3	0	0	-1.3	0	0	0	0	0	0
0	1	-1.3	0	-0.6	-1.9	1	0	0.7	0.7	0	0.7
1	0	-1.3	-0.6	0	-1.9	1	0	0.7	0.7	0.7	0
1	1	-1.3	-0.6	-0.6	-2.5	0	0	0	0	0	0
Update W_i									0.1	0.1	0.1
0	0	0.1	0	0	0.1	0	1	-0.7	-0.7	0	0
0	1	0.1	0	0.1	0.2	1	1	0	0	0	0
1	0	0.1	0.1	0	0.2	1	1	0	0	0	0
1	1	0.1	0.1	0.1	0.3	0	1	-0.7	-0.7	-0.7	-0.7
Update W_i									-1.3	-0.6	-0.6
0	0	-1.3	0	0	-1.3	0	0	0	0	0	0
0	1	-1.3	0	-0.6	-1.9	1	0	0.7	0.7	0	0.7
1	0	-1.3	-0.6	0	-1.9	1	0	0.7	0.7	0.7	0
1	1	-1.3	-0.6	-0.6	-2.5	0	0	0	0	0	0
Update W_i									0.1	0.1	0.1
0	0	0.1	0	0	0.1	0	1	-0.7	-0.7	0	0
0	1	0.1	0	0.1	0.2	1	1	0	0	0	0
1	0	0.1	0.1	0	0.2	1	1	0	0	0	0
1	1	0.1	0.1	0.1	0.3	0	1	-0.7	-0.7	-0.7	-0.7
Update W_i									-1.3	-0.6	-0.6
0	0	-1.3	0	0	-1.3	0	0	0	0	0	0
0	1	-1.3	0	-0.6	-1.9	1	0	0.7	0.7	0	0.7
1	0	-1.3	-0.6	0	-1.9	1	0	0.7	0.7	0.7	0
1	1	-1.3	-0.6	-0.6	-2.5	0	0	0	0	0	0
Update W_i									0.1	0.1	0.1

Example Network for XOR

Statistical Modeling vs. Machine Learning

- For a quick view, in statistical modeling, linear regression with two independent variables is trying to fit the best plane with the least errors, whereas when constructing a model, machine learning tries to minimize the error between the predictions and the expected outcomes (ground truth).
- That error comes from the **loss function**.
- Optimization algorithms are the heart of machine learning algorithms.
- What exactly ML algorithms optimize?
- Machine learning utilizes **optimization methods** for tuning all the parameters of various algorithms.

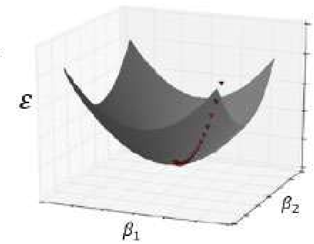
Statistical way



$$\hat{Y} = \beta_1 * X_1 + \beta_2 * X_2$$

$$\varepsilon = (Y - (\beta_1 * X_1 + \beta_2 * X_2))^2$$

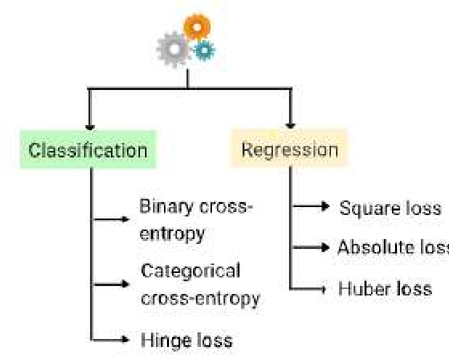
Machine learning way



Loss Function

- Output of loss function called ***Loss*** which is a measure of how well the model can predict the outcome
- A high value of ***Loss*** means the model performs very poorly, while a low value is preferrable.
- Selection of the proper loss function is critical for training an accurate model.
- Gradient descent is a famous optimizer algorithm to help finding the optimum values of parameters faster.

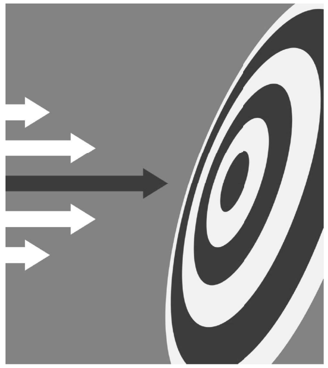
Famous Loss Function



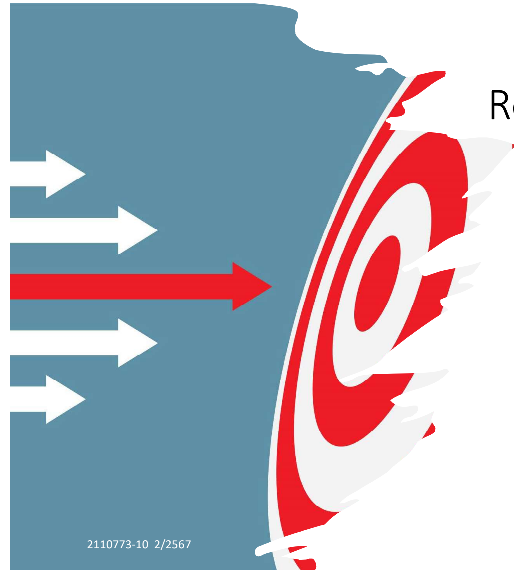
https://www.enjoyalgorithms.com/blog/loss-and-cost-functions-in-machine-learning

Loss Function vs. Cost Function

- Loss function is associated with every training example.
- Cost function is the average value of the loss function over all training samples.
- Usually try to optimize cost function rather than loss function



Regression Loss Function



- In regression tasks, we try to predict the continuous target variables. Suppose we are trying to fit the function f using machine learning on the training data X = [X1, X2, ... , Xn] so that f(x) fits Y = [Y1, Y2, ... , Yn]. But this function f can not be perfect, and there will be errors in the fitting.
- Absolute Error
- Square Error

Absolute Error / L1 Loss

L1 norm loss or **absolute loss function** is not smooth at the target, resulting in algorithms not converging well.

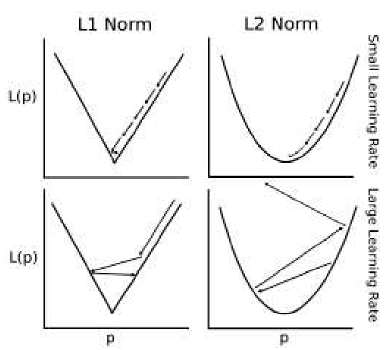
$$Loss_i = |Y_i - f(x_i)|$$

Widely used in industries, especially when training data is more prone to outliers

L1 loss is more robust to outliers than L2, or when difference is higher, L1 is more stable than L2

The corresponding cost function is Mean Absolute Error (MAE)

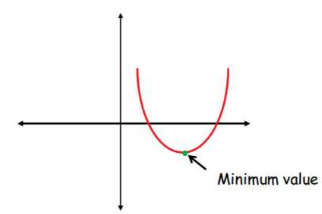
$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$



Square Error / L2 Loss

$$Loss_i = (Y_i - f(x_i))^2$$

- L2 norm loss or **Euclidean loss function** is a quadratic equation that only has a global minimum and no local minima.
- One of the most favorable loss functions as it is very curved near the target and algorithms can converge to the target closer to zero.
- L2 loss is more stable than L 1 loss, especially when the difference between prediction and actual is smaller.
- However, the squaring part magnifies the error if the model makes very bad prediction.



The corresponding cost function is Mean Squared Error (MSE) which is less robust to outlier presence.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$