

ML Lecture 6: Brief Introduction of Deep Learning

臺灣大學人工智慧中心 科技部人工智慧技術暨全幅健康照護聯合研究中心 <http://ai.ntu.edu.tw>

History of Deep Learning

Ups and downs of Deep Learning

- 1958: Perceptron (linear model)
- 1969: Perceptron has limitation
- 1980s: Multi-layer perceptron
 - Do not have significant difference from DNN today
- 1986: Backpropagation
 - Usually more than 3 hidden layers is not helpful
- 1989: 1 hidden layer is “good enough”, why deep?
- 2006: RBM initialization
- 2009: GPU
- 2011: Start to be popular in speech recognition
- 2012: win ILSVRC image competition
- 2015.2: Image recognition surpassing human-level performance
- 2016.3: Alpha GO beats Lee Sedol
- 2016.10: Speech recognition system as good as humans

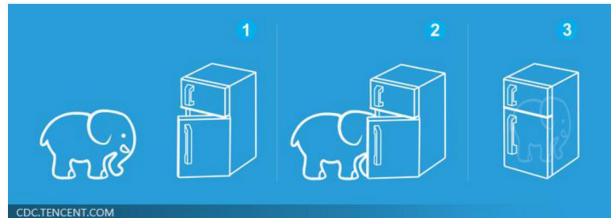
Three Steps of Deep Learning

- Deep Learning 的三個 step，和先前 Machine Learning 的三個 Step 是一樣的
- 如同將大象放進冰箱只需三步驟：「門打開、趕大象進去、門關起來」，就這麼簡單

Three Steps for Deep Learning



Deep Learning is so simple

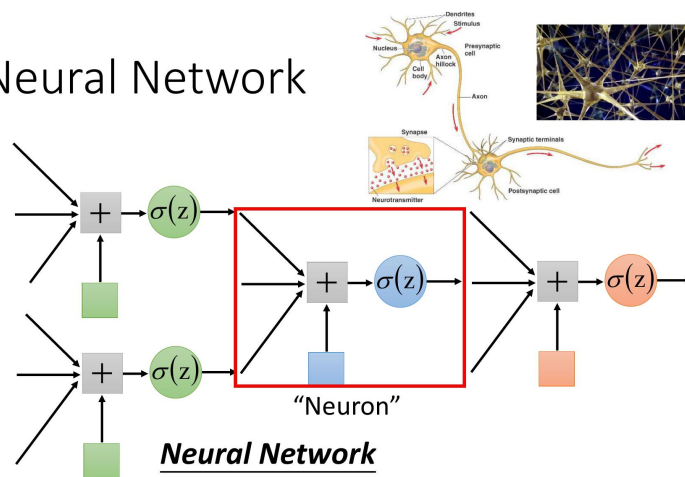


Step 1: Define a Neural Network

- **Definition**

Neuron：一個 Logistic Regression，即為一個 neuron
Neural Network：將 Logistic Regression 前後連接在一起，即為一個 neural network，以下簡稱 NN
Structure：以不同方法連接這些 NN，就形成不同的 structure
Parameter (θ)：將每個 Logistic Regression 自己的 weight 跟 bias 集合起來，就是此 NN 的 parameter

Neural Network



Neural Network

Different connection leads to different network structures

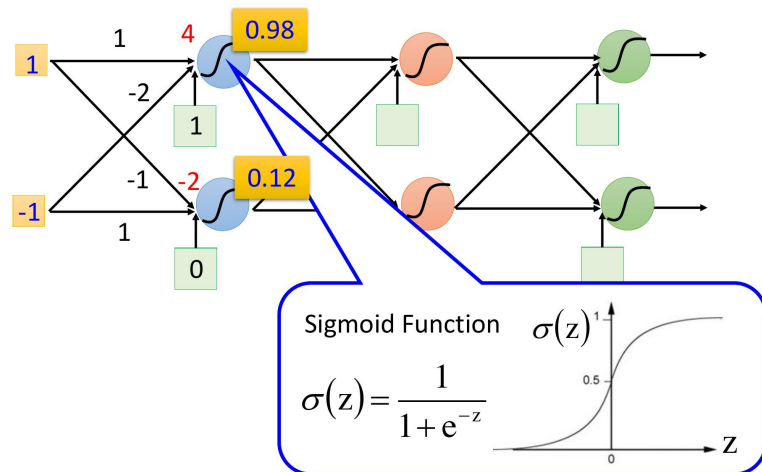
Network parameter θ : all the weights and biases in the "neurons"

- **Fully Connected Feedforward Network** (最常見的連接方式)

上面藍色 neuron 的 weight 是 (1, -2)、bias 是 -1 (綠正方形框框) 下面藍色 neuron 的 weight 是 (-1, 1)、bias 是 0 (綠正方形框框) **Input:** (1, -1) --> $1 * 1 + (-1) * (-2)$ ，再加 1(bias)，經過 sigmoid function 後，得到 0.98

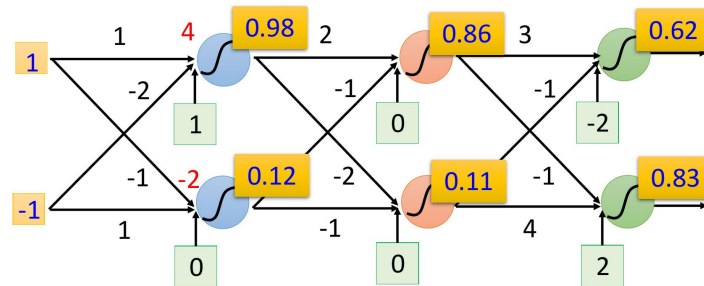
--> $1 * (-1) + (-1) * 1$ ，再加 0(bias)，經過 sigmoid function 後，得到 0.12

Fully Connect Feedforward Network



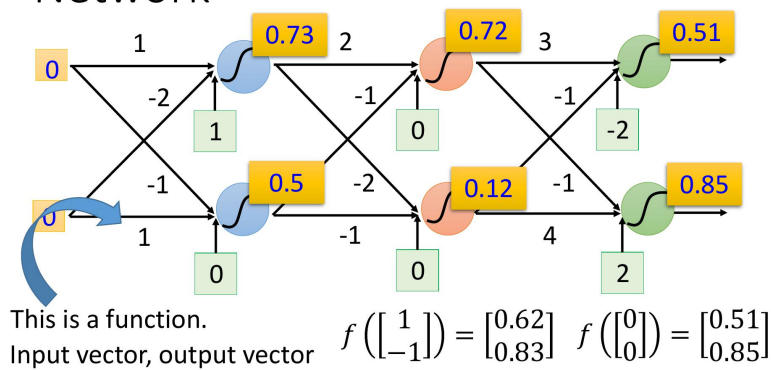
範例一（如下圖） **Input: (1, -1) --> (0.98, 0.12) --> (0.86, 0.11) --> (0.62, 0.83)**

Fully Connect Feedforward Network



範例二（如下圖） **Input: (0, 0) --> (0.73, 0.5) --> (0.72, 0.12) --> (0.51, 0.85)**

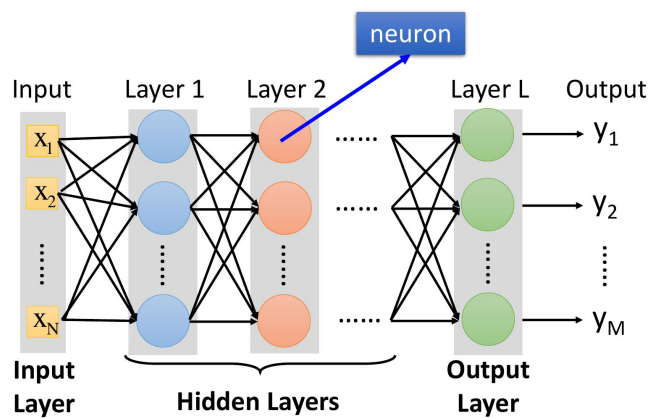
Fully Connect Feedforward Network



Given network structure, define a function set

Network 架構 (推廣) **Fully Connected** : 將 neuron 分成一排一排，每排的 neuron 都兩兩互相連接 **Feedforward Network** : 傳遞的方向為 input-->layer1-->layer2...-->Output，不斷單方向地往前傳遞 **Hidden layer** : input layer 及 output layer 以外的層皆為 hidden layer

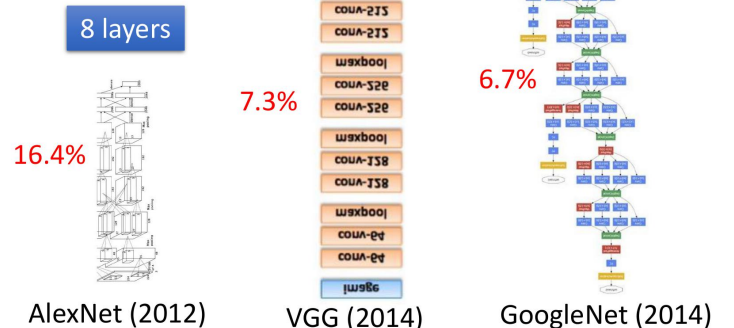
Fully Connect Feedforward Network



- 「Deep」意即 Many hidden layers

Deep = Many hidden layers

http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

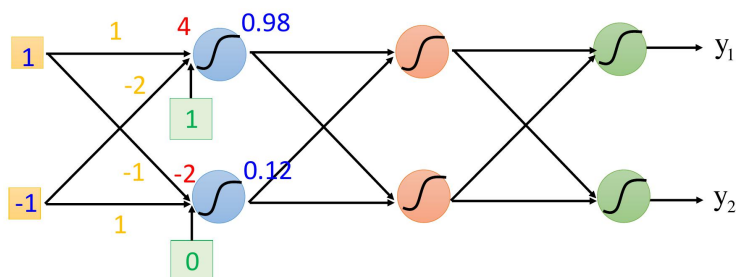


- Network 的運作

矩陣運算: $\begin{bmatrix} 1 & -1 \\ -2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \end{bmatrix}$

再通過 activation function, σ , (此處用 sigmoid function), 最後得output $\begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$

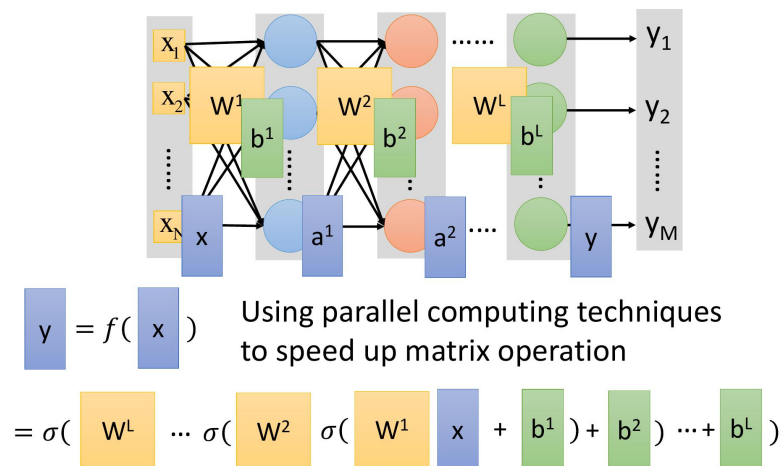
Matrix Operation



$$\sigma\left(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

⇒ Neural Network 的運作就是一連串的矩陣運算, 如下圖所示

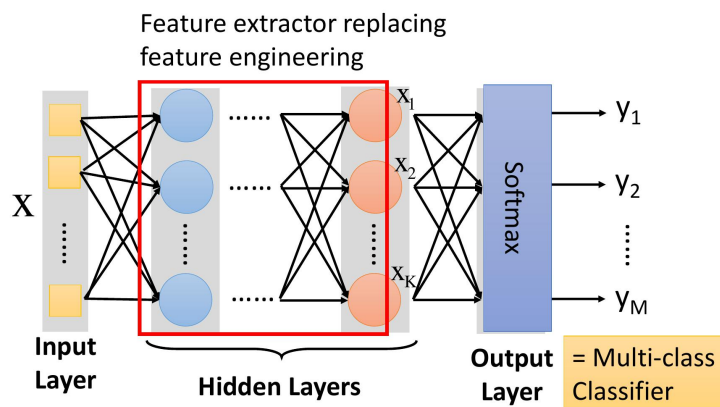
Neural Network



• Output layer 即為 Multi-Class Classifier

將 hidden layer 視為 feature extractor 將 output layer 視為 multi-class classifier，最後一個 layer 會加上 Softmax function

Output Layer as Multi-Class Classifier



• 例子：Input 一張手寫數字的 image，output 它對應到哪一個數字

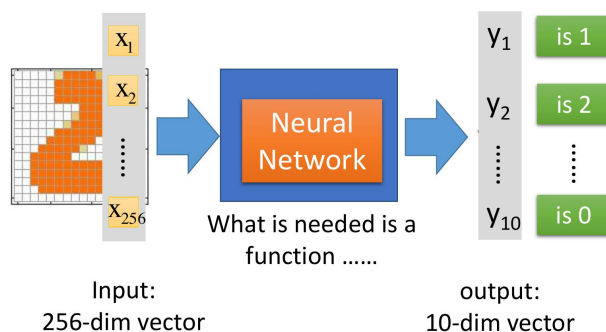
◦ 問題定義

Input：解析度 16*16 的 Image，即一個 256 維的 vector

Output：對應到 10 個數字的機率，即一個 10 維的 vector

Example Application

- Handwriting Digit Recognition



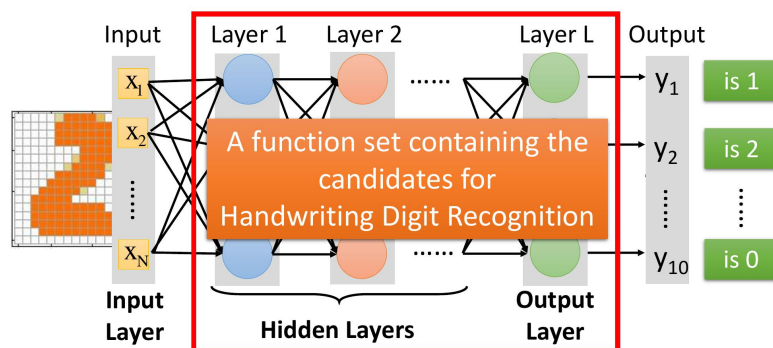
- 設計 network 架構

決定好 input, output 後，這個 network 就 define 了一個 function set 這個 function set 中，每一個 function 都可以拿來做手寫數字辨識，只有結果好壞的差別而已

⇒ 我們要設計「中間有幾層 **hidden layer**，每個 **hidden layer** 有多少的 **neuron**」

⇒ 再用 **Gradient Descent** 找一組參數，挑出最適合做手寫數字的 function

Example Application

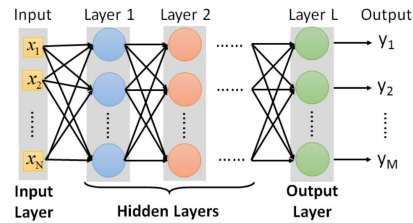


You need to decide the network structure to let a good function in your function set.

- 常見問題

- 需要多少 hidden layer？每層 hidden layer 又需要多少 neuron？⇒ 需要多方的嘗試及直覺的猜測
- 能不能夠自動學 network 的架構？⇒ 可以（細節可以請教余天立老師）
- 我們能不能自己設計 network 的架構？
⇒ 可以，一個特殊的接法就是 Convolutional Neural Network (CNN)

FAQ



- Q: How many layers? How many neurons for each layer?

Trial and Error

+

Intuition

- Q: Can the structure be automatically determined?
 - E.g. Evolutionary Artificial Neural Networks
- Q: Can we design the network structure?

Convolutional Neural Network (CNN)

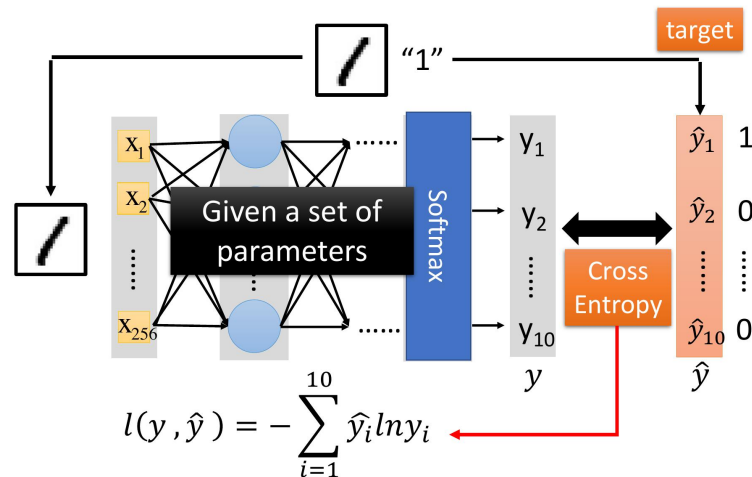
Step 2: Goodness of function

- 決定參數的好壞

計算 output (y) 跟目標 (\hat{y}) 之間的 cross entropy

⇒ 調整 network 的參數，讓 **cross entropy** 越小越好

Loss for an Example



Step 3: Pick the best function

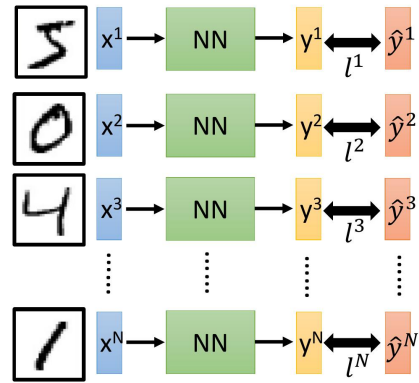
- Total loss (L)

將所有 data 的 cross entropy 全部加起來的總和，得到 total loss (L)

⇒ 在 function set 中找一個 function，或是找一組 network 的 parameter (θ^*)，讓 **total loss** 越小越好

Total Loss

For all training data ...



Total Loss:

$$L = \sum_{n=1}^N l^n$$

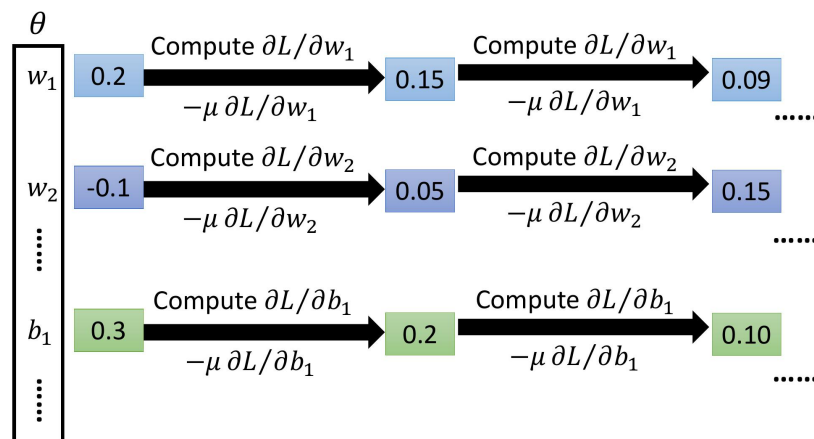
Find a function in function set that minimizes total loss L

Find the network parameters θ^* that minimize total loss L

- 用 **Gradient Descent** 找 θ^* 最小化 L

(可複習 Linear Regression 中 Gradient Descent 的做法)

Gradient Descent



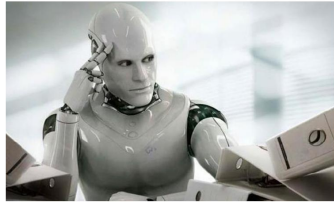
就做完 Deep Learning 了...

Gradient Descent

This is the “learning” of machines in deep learning

➡ Even alpha go using this approach.

People image



Actually



I hope you are not too disappointed :p