



# 8/9(Sun) 智慧醫療影像分類 (暖身2)

## AI for Medical Image Classification

---

Presenter : 森元俊成 (中研院統計所)

# 目標

- 在1、2小時內快速地抓到基本概念

# 預備知識

- 高中程度的數學
- 基本的數理統計



# 神經網路模型簡介

目標： 理解神經網路計算輸出的過程

# 合成函數與神經網路 part1

- 神經網路是當收到輸入 $x$ 時，輸出 $y$ 的系統
- 神經網路可以視為「合成函數」的模型



把 $x$ 轉換成 $y$ 的公式可以用合成函數來表達

# 合成函數與神經網路 part2

讓我們來複習一下高中數學吧！

令 (Let)

$$\begin{cases} f_1(x) &= x^2 \\ f_2(x) &= \sin(x) \\ f_3(x) &= e^{-x} \end{cases}$$

試求 (Find)

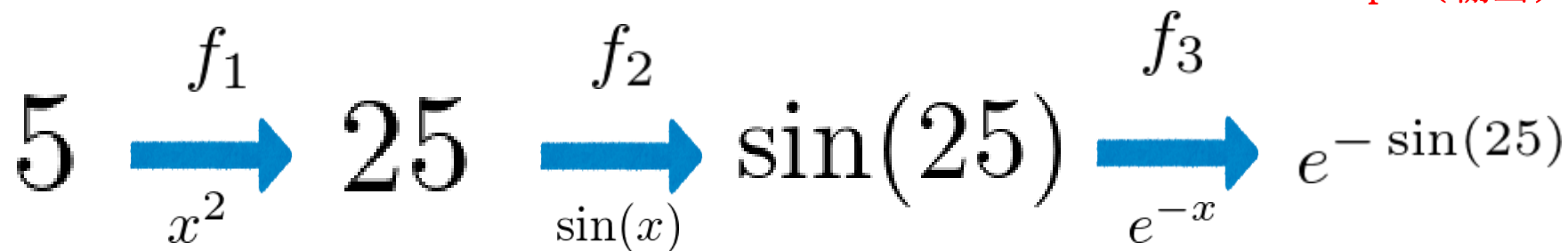
$$f_3 \circ f_2 \circ f_1(5)$$



# 合成函數與神經網路 part3

答案:

Input (輸入)



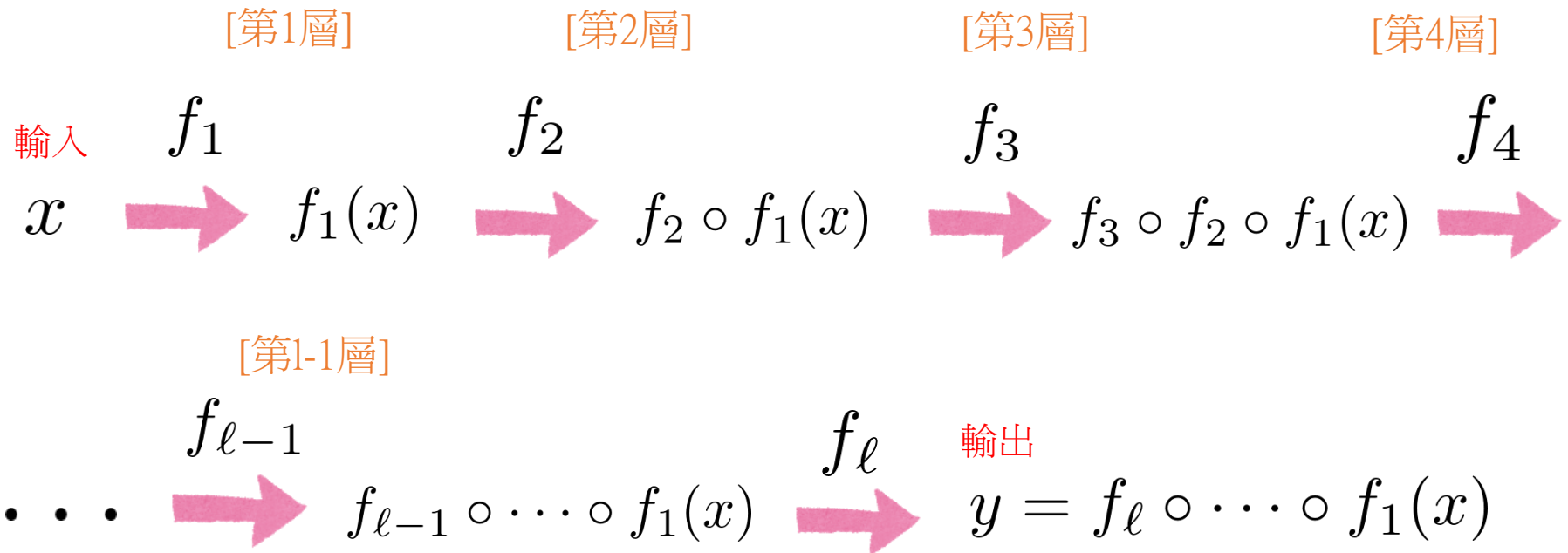
Output (輸出)

# 合成函數與神經網路 part4

神經網路把輸入 $x$ 轉換成輸出 $y$ 的過程  
也可以用合成函數來描述。

# 合成函數與神經網路 part5

神經網路計算輸出的過程：





## 疑問：

- 實際的神經網路接收到怎樣的輸入 $x$ ？
- 實際的神經網路輸出怎樣的 $y$ ？
- 如何把神經網路應用在醫療影像分類的方法上？

# 合成函數與神經網路 part6

實際的輸入資料：

輸入

$x$  不一定是一個數字，可能是向量、矩陣或張量！

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$$

向量

$$x = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}$$

矩陣

[例子]

$$x = \text{[X-ray image]}$$

黑白影像是「矩陣」，在課程中，我們會把影像當224x224的矩陣使用

# 合成函數與神經網路 part7

實際的輸出資料（作為「兩類分類器」的神經網路）：

輸出

$y$   $x$ 屬於group1的機率（ $0 \leq y \leq 1$ ）

在考慮兩類分類的問題時， $x$ 屬於group1(病人)或group0（健康的人）的其中一個

輸入的例子

$x =$



某一個人的肺部影像

$$f_{\ell} \circ f_{\ell-1} \circ \cdots \circ f_1(x)$$



神經網路把 $x$ 轉換為 $y$

輸出的例子

$y = 0.145\dots$

左方肺部影像的人罹患肺炎的機率

# 小總結：

- 神經網路是當接收到輸入 $x$ 時，輸出 $y$ 的系統。
  - 神經網路把輸出 $x$ 轉換成 $y$ 的過程可以用合成函數來描述。
  - 在本課程上，神經網路接收到的輸入 $x$ 是影像資料(矩陣)，神經網路的輸出 $y$ 是機率( $x$ 是病人的肺部影像的機率)。
- 
- 函數 $f_1, f_2, \dots, f_l(x)$ 是怎樣的函數？
  - 如何把神經網路設計成能夠精準地預測病人機率的系統？

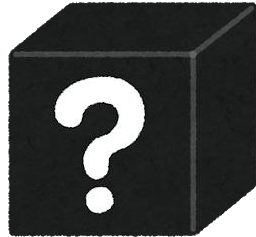
# 隱藏層與參數

目標：預習一下神經網路中出現的隱藏層(Layer)

預計在8月11日的課堂上也會講解，  
但今天先稍微預習一下

# 隱藏層為何？

- 先前已經提到神經網路是好幾個函數的合成函數。
- 但是我們還沒說明具體的 $f_1, f_2, \dots, f_l$ 是怎樣的函數。

$$f_1(x), f_2(x) \cdots f_\ell(x) =$$


我們以後把這些函數稱為隱藏層

# 與向量相關的隱藏層

雖然我們主要感興趣的對象是影像資料（矩陣或張量），  
但我們先介紹一下與向量相關的隱藏層。



# 全連結層 part1

- 全連結層 (Fully Connected Layer or Dense Layer)

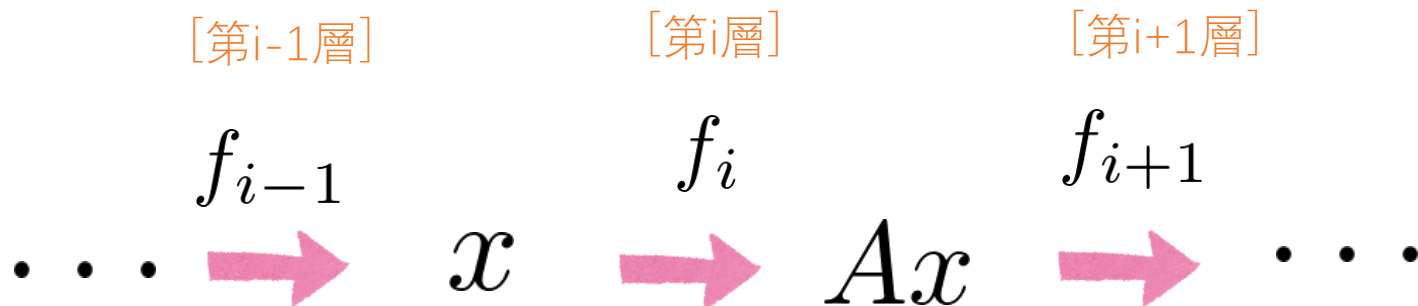
現在考慮資料是「向量」的情形

$$f_i(x) = Ax$$

矩陣與向量的乘法

- $x$  為  $p$  維的向量
- $A$  為  $m \times p$  的矩陣

$$\text{or } Ax + b$$



$x$  = 第i-1層的輸出 = 第i層的輸入

# 全連結層 part2

- 全連結層 (Fully Connected Layer or Dense Layer)

$$f_i(x) = Ax$$



這個東西A被稱為**參數**(parameter)

Q: **參數**的數值是該怎麼決定的？

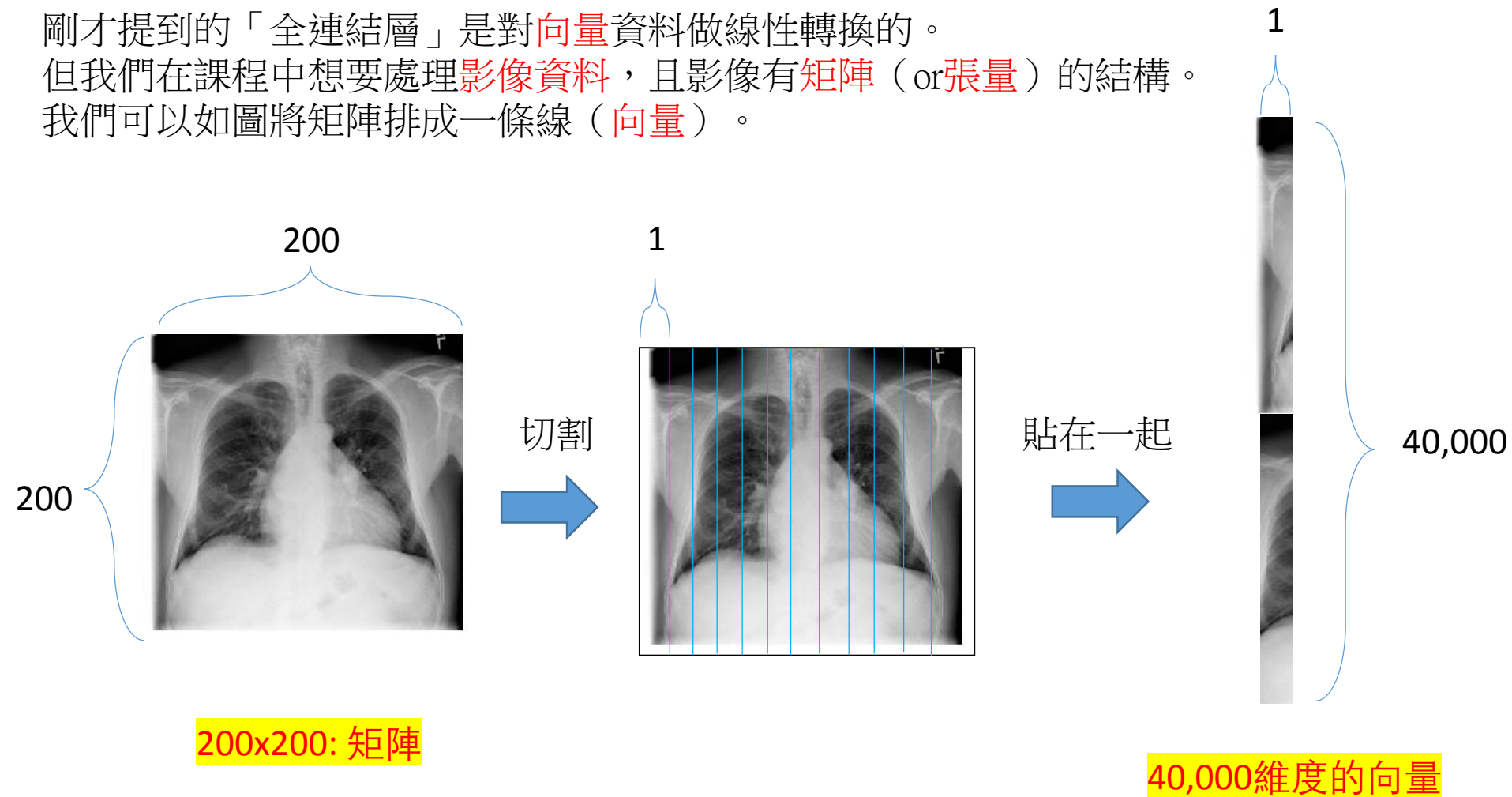
A: 我們透過**訓練(training)**這個動作找到合適的數值。在一開始時，我們把A設定為隨機給的數值，之後再利用「訓練資料」來慢慢地做調整，以使神經網路輸出更合理的數值（= y: 機率）。

# 與矩陣（or張量）相關的隱藏層

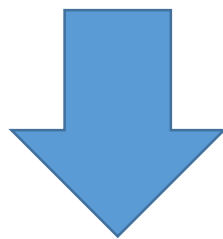
雖然我們主要感興趣的對象是影像資料（矩陣或張量），  
但我們先介紹一下與向量相關的隱藏層。

# 如何處理矩陣資料？

剛才提到的「全連結層」是對**向量**資料做線性轉換的。  
但我們在課程中想要處理**影像資料**，且影像有**矩陣**（or**張量**）的結構。  
我們可以如圖將矩陣排成一條線（**向量**）。



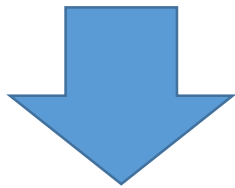
但這樣子的動作破壞影像本身的平面結構，  
因而會失去影像具有的重要訊息！



所以……

我們想要保留影像的平面結構！

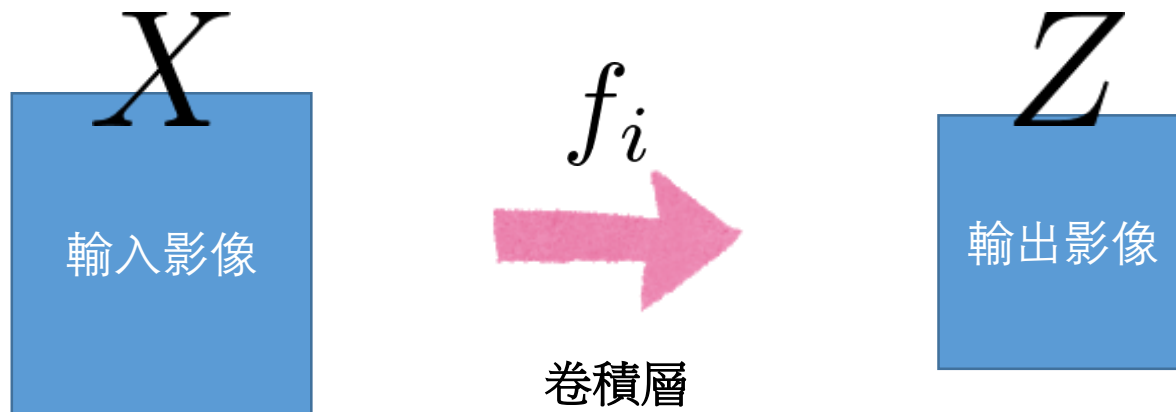
不要把它排成一條線（向量）



卷積層：Convolutional Layer

# 卷積層 part1

- 卷積層 (Convolutional Layer)



通常變成比原本的 $X$ 小一點的影像

卷積層的輸入 $X$ 是影像，輸出 $Z$ 也是影像

但是卷積層的輸入和輸出不一定是一張影像

例如：輸入可能是彩色影像。彩色影像是R、G、B三張影像疊在一起的東西



為了簡單起見，今天只考慮輸入輸出都是一張影像的情形

在介紹卷積層之前，

讓我們先複習一下國小的數學!!

請計算：

$$\begin{pmatrix} 5 \times 2 & 1 \times 0 & 0 \times (-2) \\ 1 \times 0 & 6 \times 2 & 1 \times 0 \\ 4 \times (-2) & 1 \times 0 & 6 \times 2 \end{pmatrix}$$

# 答案

$$\begin{pmatrix} 10 & 0 & 0 \\ 0 & 12 & 0 \\ -8 & 0 & 12 \end{pmatrix}$$



這九個數字的相加為多少？

$$10 + 0 - 8 + 0 + 12 + 0 + 0 + 0 + 12$$

$$= 26$$

其實卷積層算一樣的東西

## 卷積層計算的例子

$$\begin{pmatrix} 5 & 1 & 0 & 1 & 0 \\ 1 & 6 & 1 & 7 & 0 \\ 4 & 1 & 6 & 3 & 5 \\ 1 & 8 & 0 & 0 & 1 \\ 2 & 2 & 8 & 0 & \end{pmatrix} \begin{array}{c} \textcircled{*} \\ \text{卷積} \end{array} \begin{pmatrix} 2 & 0 & -2 \\ 0 & 2 & 0 \\ -2 & 0 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 26 & 6 & -6 \\ 0 & -4 & 8 \\ 24 & -14 & -4 \end{pmatrix}$$

卷積層中的參數W

卷積層的輸入X

 $X$ 

卷積層的輸出Z

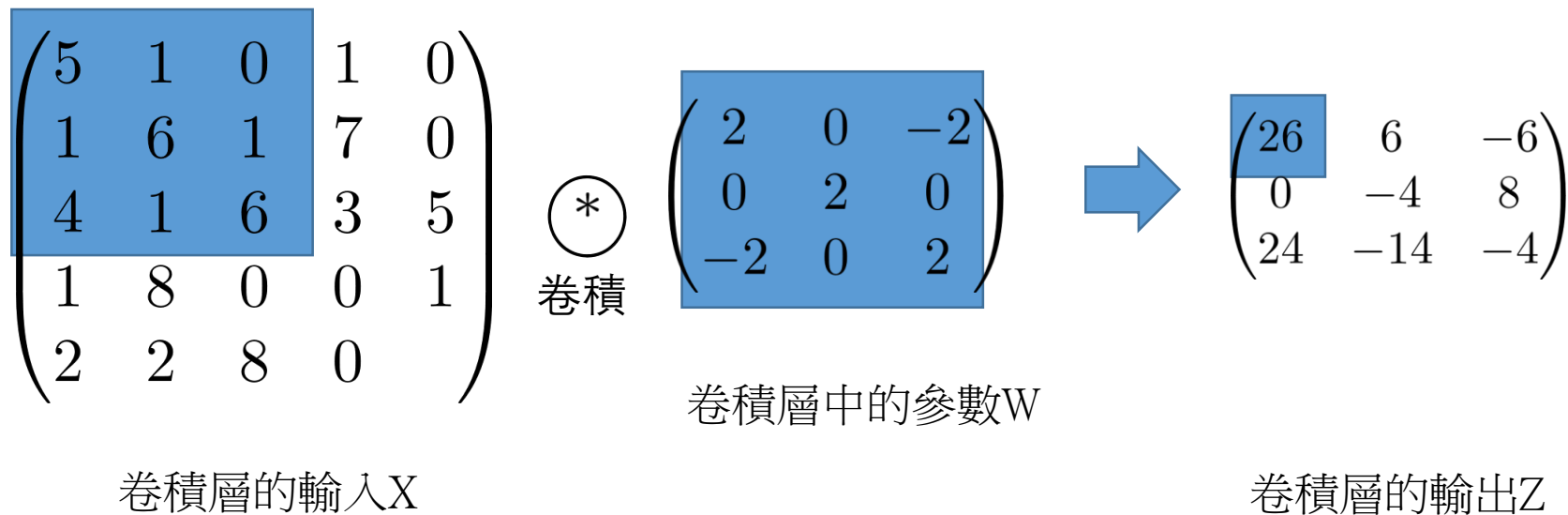
 $Z$

Why?



## 卷積層的例子

# 卷積層 part3



請看一下藍色的地方  
(對這些數字有印象嗎?)

# Again:

$$\begin{pmatrix} 5 \times 2 & 1 \times 0 & 0 \times (-2) \\ 1 \times 0 & 6 \times 2 & 1 \times 0 \\ 4 \times (-2) & 1 \times 0 & 6 \times 2 \end{pmatrix}$$



$$\begin{pmatrix} 10 & 0 & 0 \\ 0 & 12 & 0 \\ -8 & 0 & 12 \end{pmatrix}$$



相加

# 26

## 卷積層的例子

# 卷積層 part4

$$\begin{pmatrix} 5 & 1 & 0 & 1 & 0 \\ 1 & 6 & 1 & 7 & 0 \\ 4 & 1 & 6 & 3 & 5 \\ 1 & 8 & 0 & 0 & 1 \\ 2 & 2 & 8 & 0 & \end{pmatrix}$$

卷積層的輸入X



卷積

$$\begin{pmatrix} 2 & 0 & -2 \\ 0 & 2 & 0 \\ -2 & 0 & 2 \end{pmatrix}$$

卷積層中的參數W



$$\begin{pmatrix} 26 & 6 & -6 \\ 0 & -4 & 8 \\ 24 & -14 & -4 \end{pmatrix}$$

卷積層的輸出Z

把藍色的範圍往右方移動，

## 卷積層的例子

# 卷積層 part5

$$\begin{pmatrix} 5 & 1 & 0 & 1 & 0 \\ 1 & 6 & 1 & 7 & 0 \\ 4 & 1 & 6 & 3 & 5 \\ 1 & 8 & 0 & 0 & 1 \\ 2 & 2 & 8 & 0 & \end{pmatrix}$$

卷積層的輸入X



卷積

$$\begin{pmatrix} 2 & 0 & -2 \\ 0 & 2 & 0 \\ -2 & 0 & 2 \end{pmatrix}$$

卷積層中的參數W



$$\begin{pmatrix} 26 & 6 & -6 \\ 0 & -4 & 8 \\ 24 & -14 & -4 \end{pmatrix}$$

卷積層的輸出Z



把藍色的範圍往右方移動

## 卷積層的例子

# 卷積層 part6

$$\begin{pmatrix} 5 & 1 & 0 & 1 & 0 \\ 1 & 6 & 1 & 7 & 0 \\ 4 & 1 & 6 & 3 & 5 \\ 1 & 8 & 0 & 0 & 1 \\ 2 & 2 & 8 & 0 & \end{pmatrix}$$

卷積層的輸入X



卷積

$$\begin{pmatrix} 2 & 0 & -2 \\ 0 & 2 & 0 \\ -2 & 0 & 2 \end{pmatrix}$$

卷積層中的參數W



$$\begin{pmatrix} 26 & 6 & -6 \\ 0 & -4 & 8 \\ 24 & -14 & -4 \end{pmatrix}$$

卷積層的輸出Z



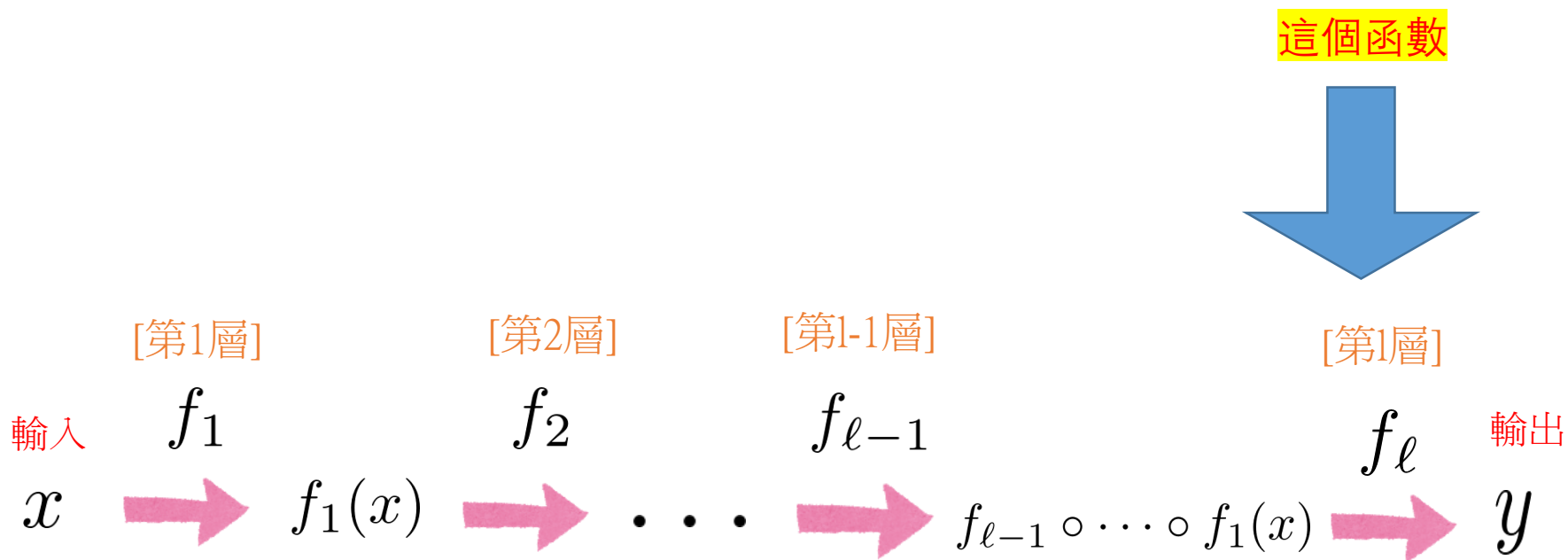
走到最右邊的時候，先回到最左邊，再往下移動

# 小總結：

- 我們介紹了神經網路中常見的隱藏層
  - 全連接層（對輸入的向量做線性轉換）
  - 卷積層（對影像資料=矩陣使用的隱藏層）
- 隱藏層會包含一些參數
  - 我們透過「訓練」這個動作調整參數數值

等一下！！

- 但我沒還沒講到神經網路的最後一層的輸出的怎麼算的？
- 怎樣才能使最後一層輸出為0到1的數值（機率）呢？



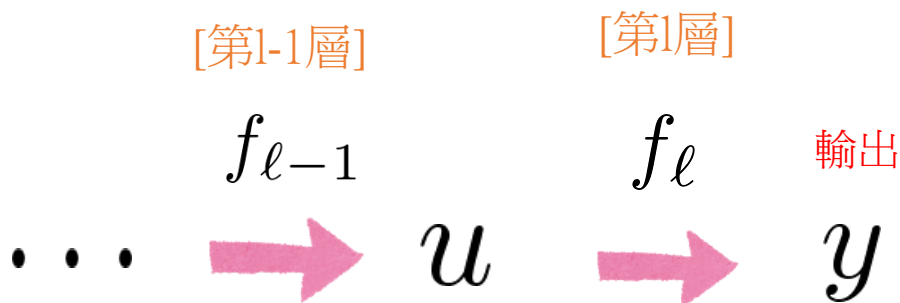
$$y = f_{\ell} \circ \dots \circ f_1(x)$$



# 輸出層的設計

- 倒數第二層(第 $l-1$ 層)的輸出 $u$ 是單變數
- 最後一層(第 $l$ 層)透過sigmoid函數把 $u$ 轉換成0到1的數值
- 我們把最後一層的輸出 $y$ 想成病人的機率

$$f_l(u) = \frac{1}{1+e^{-u}}$$



# 神經網路的參數訓練

先回顧一下前面講到的內容

# 回顧一下...

- 我們可以將神經網路視為合成函數的模型。
- 神經網路接收到 $x$ (影像, 矩陣)時, 輸出 $x$ 為病人的機率( $=y$ )
- 函數 $f_1, f_2 \dots f_l$ 會包含一些「參數」(如:  $A, W, b \dots$  etc)
- 我們一開始把「參數」設定成隨機產生的數值, 之後利用「訓練資料」來調整參數數值。

矩陣 (影像)

$x =$



神經網路把 $x$ 轉換成 $y$

有病的機率



$y = 0.145\dots$

$$f_l \circ f_{l-1} \circ \dots \circ f_1(x)$$



其中包含一些參數

利用「訓練資料」來調整參數數值

$A, W, b \dots$



訓練資料為何？

假設我們拿到一群人的肺部影像以及這些人是病人還是健康的人的資料



我們把這筆資料稱為「訓練資料」。  
我們利用這些人的資料來建立一個“聰明的神經網路”

## 例子

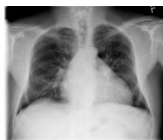
# 訓練資料 part1

影像(矩陣)

病人or健康的人

第一個人

$x_1$



$$t_1 = 1$$

病人

第二個人

$x_2$



$$t_2 = 0$$

健康

第三個人

$x_3$



$$t_3 = 0$$

病人

⋮

⋮

⋮

第n個人

$x_n$



$$t_n = 1$$

病人

還沒訓練神經網路的時候...





# 訓練資料 part2

## 神經網路輸出的機率

$$y_1 = 0.03 (= f_\ell \circ \dots \circ f_1(x_1))$$

$$y_2 = 0.98 (= f_\ell \circ \dots \circ f_1(x_2))$$

( $y_1$ 、 $y_2$ 分別為第一個人、第二個人是病人的機率)



因為一開始給的參數是隨機產生的，  
所以神經網路預測的結果可能不符合實際狀況。



我們需要調整參數，以使神經網路輸出合理的數值

第一個人  $x_1$



$$t_1 = 1 \quad \text{病人}$$

第二個人  $x_2$



$$t_2 = 0 \quad \text{健康}$$

OK...我們已經知道訓練神經網路的必要性了

但是我們該如何調整那些參數呢？

我們先從數理統計的參數估計問題出發，  
再探討神經網路訓練參數的方法

# 統計：最大概似估計 part1

伯努利分配的參數估計問題 (1)

$$T_1, T_2, \dots, T_n \stackrel{\text{iid}}{\sim} \text{Bernoulli}(p)$$

Find the maximum likelihood estimator of  $p$

先複習一下機率統計...

# 伯努利分配 (1)

- 我們丟一個銅板，出現正面的機率是 $p$ （不一定是 $1/2$ ）
- 我們約定如果銅板出現正面，令 $T=1$ ，否則令 $T=0$
- 我們說 $T$ 服從參數 $p$ 的伯努利分配。



# 伯努利分配 (2)

$$\begin{cases} P(T = 1) = p \\ P(T = 0) = 1 - p \end{cases}$$



這兩件事情其實可以寫成：

$$P(T = t) = p^t \cdot (1 - p)^{(1-t)}$$

$$t = 0, 1$$

# 伯努利分配 (3)

如果我們丟了n次銅板的話...

$$P(T_1 = t_1, T_2 = t_2, \dots, T_n = t_n) = \prod_{i=1}^n p^{t_i} \cdot (1 - p)^{1-t_i}$$



# 讓我們回到伯努利分配的最大概 似估計的問題...



假設今天我們不知道 $p$ 是多少，我們只知道 $t_1, t_2, \dots, t_n$ 是多少。  
我們想從觀測值 $t_1, t_2, \dots, t_n$ 猜 $p$ 是多少

# 統計：最大概似估計 part2

## 伯努利分配的參數估計問題 (1)

STEP 1: 將機率質量函數寫下來：

$$P(T_1 = t_1, \dots, T_n = t_n \mid p) = \prod_{i=1}^n p^{t_i} (1 - p)^{1-t_i}$$

STEP 2: 為了方便起見，取兩邊的自然對數：

$$\ln P = \sum_{i=1}^n \{t_i \ln p + (1 - t_i) \ln(1 - p)\}$$

STEP 3: 我們把上述式子視為關於p的函數：

$$\ell(p) = \sum_{i=1}^n \{t_i \ln p + (1 - t_i) \ln(1 - p)\}$$

STEP 4: 求p使得 $\ell(p)$ 取最大值

$$\hat{p} \leftarrow \operatorname{argmax}_p \ell(p)$$

最大概似估計法：

我們在考慮怎樣的 $p$ 最有可能產生我們現在所觀測到的  $t_1, \dots, t_n$

這個數理統計的題目跟神經網路的參數訓練到底有什麼關聯？



我們先討論一個類似的問題再說...

我們剛才考慮的狀況是每次出現正面的機率是一樣的 $p$



我們接下來考慮每次出現正面的機率是不同的狀況（的最大概似估計）

# 統計：最大概似估計 part3

## 伯努利的參數估計問題 (2)

我們剛剛考慮的狀況是每個 $T_i = 1$ 的機率均為共同的 $p$ ，但接下來考慮較複雜的情況。假設我們有 $n$ 筆如下兩兩一組的樣本。 $(T_i)$ 是隨機變數， $x_i$ 是已知的共變數

$$(T_1, x_1), (T_2, x_2), \dots, (T_n, x_n) \quad x_1, \dots, x_n : \text{共變數 (已知)}$$

每個 $T_i$ 皆為獨立且服從Bernoulli分配。但每個 $T_i$ 為1的機率均不同且該機率由 $x_i$ 來決定。

$$T_i \sim \text{Bernoulli}(y_i), \quad y_i = g_\theta(x_i)$$

$g_\theta(x)$ 是一個包含參數 $\theta$ 的函數



Q. 試求 $\theta$ 的最大概似估計量(MLE)

題目的情況稍微不同，  
但解題步驟是差不多的

# 統計：最大概似估計 part4

## 伯努利分配的參數估計問題 (2)

STEP 1: 將機率質量函數寫下來：

$$P(T_1 = t_1, \dots, T_n = t_n \mid \theta) = \prod_{i=1}^n y_i^{t_i} (1 - y_i)^{1-t_i}$$

STEP 2: 為了方便起見，取兩邊的自然對數：

$$\ln P = \sum_{i=1}^n \{t_i \ln(y_i) + (1 - t_i) \ln(1 - y_i)\}$$

STEP 3: 我們把上述式子視為關於  $\theta$  的函數：

$$\ell(\theta) = \sum_{i=1}^n \{t_i \ln(y_i) + (1 - t_i) \ln(1 - y_i)\}$$

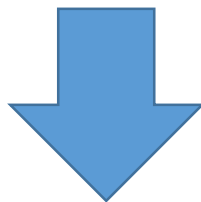
STEP 4: 求  $\theta$  使得  $\ell(\theta)$  取最大值

注  $y_i = g_\theta(x_i)$

$$\hat{\theta} \leftarrow \operatorname{argmax}_{\theta} \ell(\theta)$$



所以這個問題跟神經網路的參數訓練有什麼關聯？



- 請把剛剛的 $x_i$ 想成肺部影像的矩陣。
- 請把剛剛的 $y_i$ 想成神經網路輸出的機率。
- 請把 $t_i=1$ ( $t_i=0$ )想成第 $i$ 個人是病人（健康的人）這件事情



這樣我們已經知道該怎麼找參數了

明天8/10的課程會提到的Logistic Regression也是類似的概念

(若輸入 $x$ 是向量，且接一層全連結層+輸出層為sigmoid函數的簡單神經網路 = Logistic Regression的模型)

# 小總結：

訓練神經網路的參數

= 找使得損失函數  $l(\theta)$  最小的參數  $\theta$  .



Q. 但是找到使損失函數最小的  $\theta$  這件事情簡單嗎？

# 神經網路中的最佳化問題

我們已經知道訓練神經網路時需要最小化的損失函數長怎麼樣，但是把它最小化也是一件不容易的事情！

# 最小化的問題不一定簡單

## 最佳化的問題

Q. 求 $x$ 使得以下式子為最小:

$$x^2 - 2x + 4$$



很簡單！

Q. 求 $\theta$ 使得以下式子為最小:

$$\frac{1}{\theta^4 + 1} + 10\theta^{\sqrt{2}} + |\cos \theta|$$



有點困難！

# 神經網路損失函數的最小化也困難！

其實，神經網路的損失函數寫起來相當困難

$$-\ell(\theta) = \sum_{i=1}^n \{-t_i \ln(y_i) - (1 - t_i) \ln(1 - y_i)\}$$

因為：

- $\theta$  不只是一個單變數，而包含許多向量、矩陣、張量
- $\theta$  的訊息被包含在 $y_i$ 裡面。但 $y_i$ 寫起來也很複雜（合成函數）





那些困難的最小化的問題可以利用「梯度下降法」來解決。

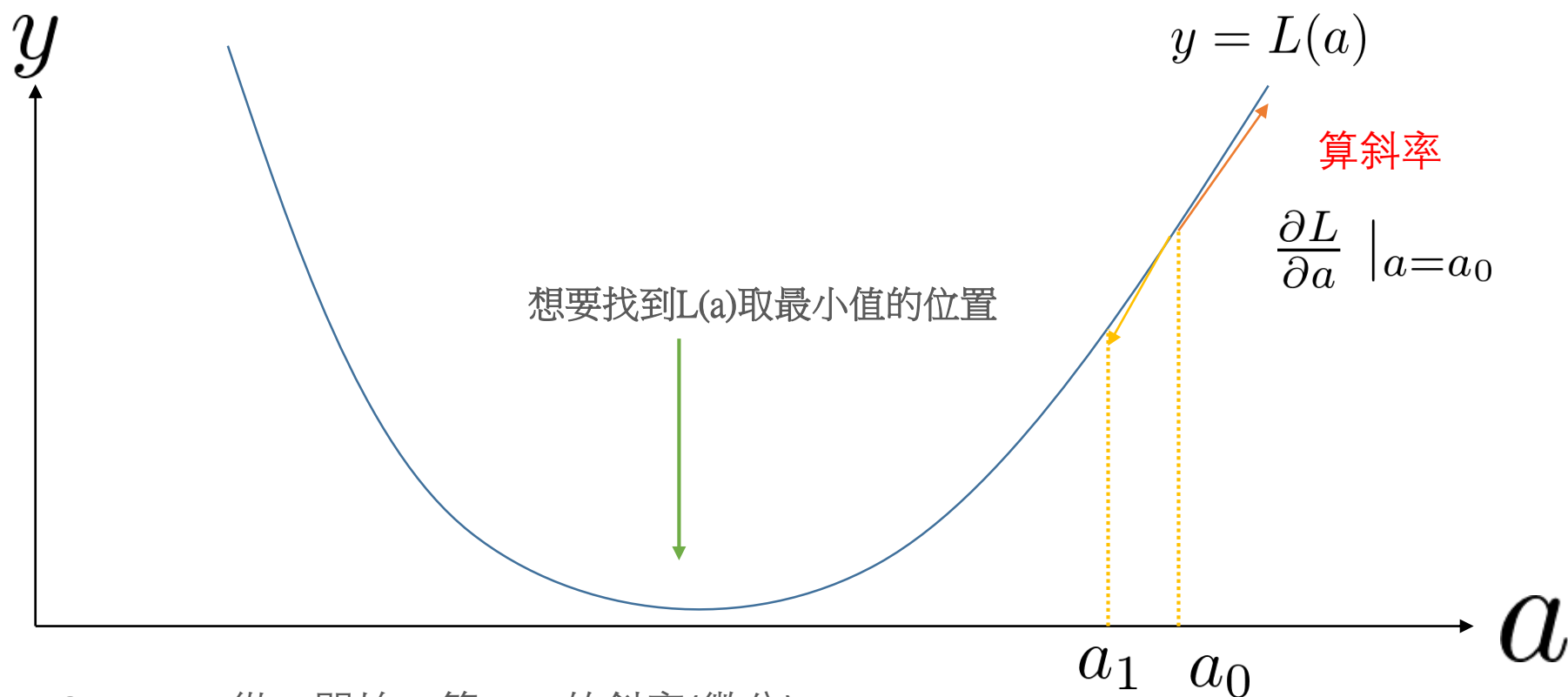
（如果損失函數是可微的話…）

# 梯度下降法是怎樣的方法？



利用曲線、曲面的斜率（梯度）來慢慢地接近取最小值的那個點

# 梯度下降法



- STEP1: 從 $a_0$ 開始，算 $a=a_0$ 的斜率(微分)
- STEP2: 根據以下公式更新 $a_0$ 為 $a_1$
- STEP3: 反復一樣的動作

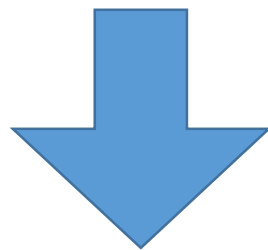
$$a_1 \leftarrow a_0 - \gamma \frac{\partial L}{\partial a} \Big|_{a=a_0}$$



為了簡單起見，剛才只考慮一個變數的情形，但多變數也可以用同樣的方法來解決！

P.S. 梯度下降法有很多改良版（momentum、AdaGrad、Adam）

OK ... 所以，我們現在基本上只要計算損失函數對參數的微分就好了？ 對吧？



YES ... 但是算微分會有點麻煩...

Why?

A. 因為神經網路的輸出 $y$ 是一堆函數的合成函數




# 損失函數的微分算起來很麻煩？

我們現在要算損失函數 $-\ell(\theta)$ 的微分：

$$-\ell(\theta) = \sum_{i=1}^n \{-t_i \ln(y_i) - (1 - t_i) \ln(1 - y_i)\}$$

$\theta = (\theta_1, \theta_2, \dots)$  有好幾個變數

$\theta$  的訊息在  $y_i$  裡面


$$y_i = \underbrace{f_\ell \circ \dots \circ f_1}_{\theta} (x_i)$$

$\theta = (\theta_1, \theta_2, \dots)$  在函數  $f_1, f_2, \dots, f_\ell$  中



# 反向傳播算法可以有效率地算微分

反向傳播演算法(Backpropagation)聽起來是一個很了不起的概念，  
但它只不過是利用微分的連鎖律來把一個微分分解成幾個微分的相乘而已…

# 反向傳播演算法 part1

為了簡單起見，我們考慮以下L對 $\theta_1, \dots, \theta_5$ 的微分

$$L(\theta_1, \dots, \theta_5) = \sum_{i=1}^n f_5 \circ f_4 \circ \dots \circ f_1(x_i)$$

$$\begin{cases} f_1(x) = f_1(x \mid \theta_1), \\ f_2(x) = f_2(x \mid \theta_2), \\ \dots \\ f_5(x) = f_5(x \mid \theta_5) \end{cases}$$

為了避免符號變得太過複雜，我們把第一層到第五層的輸出命名為 $t_i, u_i, v_i, w_i, z_i$

$$\left\{ \begin{array}{l} t_i = f_1(x_i \mid \theta_1) \\ u_i = f_2(t_i \mid \theta_2) \\ v_i = f_3(u_i \mid \theta_3) \\ w_i = f_4(v_i \mid \theta_4) \\ z_i = f_5(w_i \mid \theta_5) \end{array} \right.$$

$$L = \sum_{i=1}^n z_i$$

# 反向傳播演算法 part 2

**STEP1:** 我們先考慮L對  $\theta_5$  的微分 (因為比較簡單)  $L = \sum_{i=1}^n z_i$

$$\frac{\partial L}{\partial \theta_5} = \sum_{i=1}^n \frac{\partial z_i}{\partial \theta_5} = \sum_{i=1}^n \frac{\partial f_5(w_i | \theta_5)}{\partial \theta_5}$$

# 反向傳播演算法 part 3

**STEP2:** 接著考慮L對  $\theta_4$  的微分 (因為比較簡單)

$$\frac{\partial L}{\partial \theta_4} = \sum_{i=1}^n \frac{\partial f_5(w_i)}{\partial w_i} \frac{\partial f_4(v_i \mid \theta_4)}{\partial \theta_4}$$

利用微分的連鎖律(chain rule)可以寫成兩個微分的相乘



但還看不出來什麼叫做反向傳播演算法...

不好意思，請繼續看下一頁...

# 反向傳播演算法 part4

**STEP3:** 接著考慮L對  $\theta_3$  的微分

同樣地，我們可以利用連鎖律把它寫成三個微分的相乘

$$\frac{\partial L}{\partial \theta_3} = \sum_{i=1}^n \frac{\partial f_5(w_i)}{\partial w_i} \frac{\partial f_4(v_i)}{\partial v_i} \frac{\partial f_3(u_i | \theta_3)}{\partial \theta_3}$$



很開心的是，我們先前已經算過這個微分  
(不用再算一次)



# 反向傳播演算法 part5

STEP4: 接著考慮L對  $\theta_2$  的微分

$$\frac{\partial L}{\partial \theta_2} = \sum_{i=1}^n \frac{\partial f_5(w_i)}{\partial w_i} \frac{\partial f_4(v_i)}{\partial v_i} \frac{\partial f_3(u_i)}{\partial u_i} \frac{\partial f_2(t_i | \theta_2)}{\partial \theta_2}$$

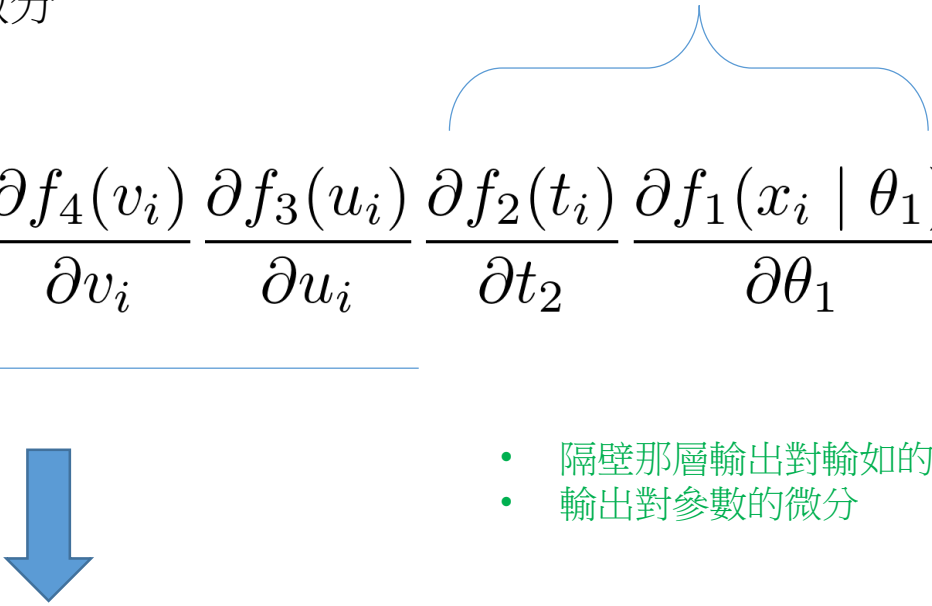


哇！這兩個微分我們已經算過！！不用再算一次！

# 反向傳播演算法 part6

我們每層的微分只要算這兩個東西即可！

**STEP5:** 最後考慮L對  $\theta_1$  的微分

$$\frac{\partial L}{\partial \theta_1} = \sum_{i=1}^n \frac{\partial f_5(w_i)}{\partial w_i} \frac{\partial f_4(v_i)}{\partial v_i} \frac{\partial f_3(u_i)}{\partial u_i} \frac{\partial f_2(t_i)}{\partial t_2} \frac{\partial f_1(x_i | \theta_1)}{\partial \theta_1}$$


- 隔壁那層輸出對輸如的微分
- 輸出對參數的微分

耶！！這三個微分我們在前面已經算過！！！不用再算一次！

因為從後面往前面計算微分，  
所以這個演算法名稱才會有「反向」這個字

# 總結

- 神經網路輸出的計算方法
- 神經網路中的參數的訓練方法（損失函數）
- 損失函數最小化的方法（梯度下降法）
- 介紹反向傳播算法



接下來的課程會講解更多更富的內容，敬請期待！

**Thank you for your attention!**

[juncheng@stat.sinica.edu.tw](mailto:juncheng@stat.sinica.edu.tw)