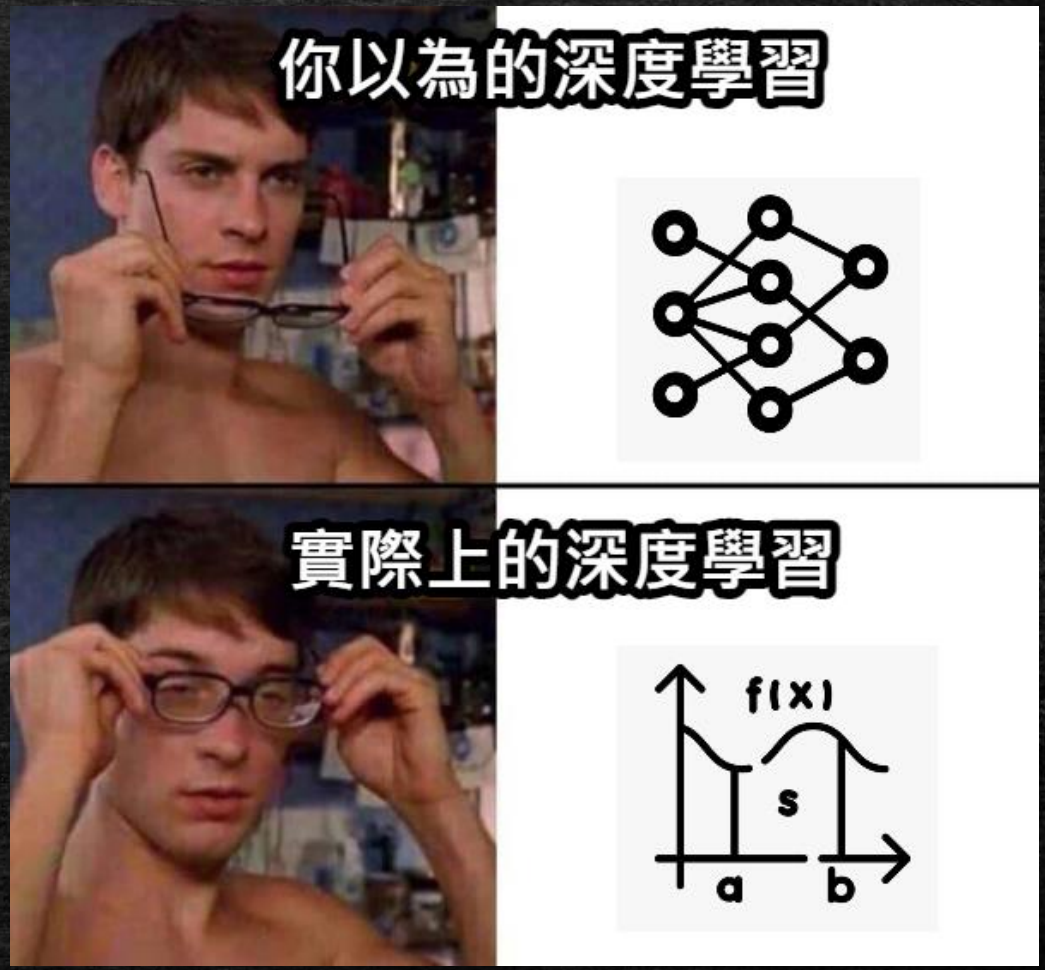
2024 鐵人賽 –我數學就爛要怎麼來學 DNN 模型安全
Day 02 – 我就爛怎麼算數學?

大綱

- 機器學習
  - 用到那些數學
  - 線性代數
  - 微積分
- 結論



機器學習 - 用到那些數學

- 類神經網路模型內部的前向傳播用到**線性代數基本的運算**，接著會用**微積分算導數**決定要逼近的方向，最後再用後向傳播更新模型

- 這其中會用到的數學包含線性代數、微積分、~~機率與統計~~

機器學習 – 矩陣及運算定義

數學上，一個$m \times n$的**矩陣**是一個有m列（row）n行（column）元素的矩形陣列。矩陣裡的元素可以是數字或符號甚至是函式。

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2j} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3j} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \dots & a_{ij} & \dots & a_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mj} & \dots & a_{mn} \end{bmatrix}$$

機器學習 – 矩陣及運算定義

運算	定義	例子
加（減）法	$m \times n$矩陣$\mathbf{A}$和$\mathbf{B}$的和（差）：$\mathbf{A} \pm \mathbf{B}$為一個$m \times n$矩陣，其中每個元素是$\mathbf{A}$和$\mathbf{B}$相應元素的和（差），$(\mathbf{A} \pm \mathbf{B})_{i,j} = \mathbf{A}_{i,j} \pm \mathbf{B}_{i,j}$，其中$1 \le i \le m, 1 \le j \le n$	$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 & 1+5 \\ 1+7 & 0+5 & 0+0 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 6 \\ 8 & 5 & 0 \end{bmatrix}$
實數積	純量c與矩陣$\mathbf{A}$的實數積：$c\mathbf{A}$的每個元素是$\mathbf{A}$的相應元素與c的乘積，$(c\mathbf{A})_{i,j} = c \cdot \mathbf{A}_{i,j}$	$2 \cdot \begin{bmatrix} 1 & 8 & -3 \\ 4 & -2 & 5 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 & 2 \cdot 8 & 2 \cdot (-3) \\ 2 \cdot 4 & 2 \cdot (-2) & 2 \cdot 5 \end{bmatrix} = \begin{bmatrix} 2 & 16 & -6 \\ 8 & -4 & 10 \end{bmatrix}$
轉置	$m \times n$矩陣$\mathbf{A}$的轉置是一個$n \times m$的矩陣，記為$\mathbf{A}^{\mathrm{T}}$（有些書中也記為$\mathbf{A}^{\mathrm{tr}}$或${}^{\mathrm{t}}\mathbf{A}$、$\mathbf{A}'$），其中的第$i$個列向量是原矩陣$\mathbf{A}$的第$i$個行向量；或者說，轉置矩陣$\mathbf{A}^{\mathrm{T}}$第$i$列第$j$行的元素是原矩陣$\mathbf{A}$第$j$列第$i$行的元素，$(\mathbf{A}^{\mathrm{T}})_{i,j} = \mathbf{A}_{j,i}$	$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & 7 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 \\ 2 & -6 \\ 3 & 7 \end{bmatrix}$

機器學習 – 矩陣及運算定義

矩陣乘法 [編輯]

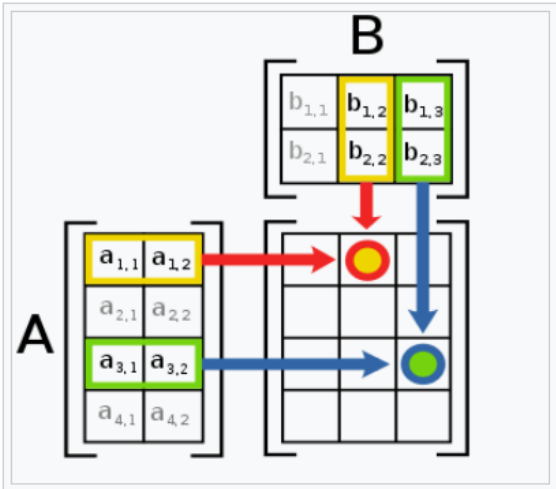
主條目：矩陣乘法

兩個矩陣的乘法唯若第一個矩陣**A**的行數(column)和另一個矩陣**B**的列數(row)相等時才能定義。如**A**是$m \times n$矩陣和**B**是$n \times p$矩陣，它們的**乘積AB**是一個$m \times p$矩陣，它的一個元素

$$[\mathbf{AB}]_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + \cdots + A_{i,n}B_{n,j} = \sum_{r=1}^{n} A_{i,r}B_{r,j}$$

其中$1 \le i \le m,\ 1 \le j \le p$[11]。

例如



矩陣**A**和**B**相乘得到**AB**的示意圖

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} (1 \times 3 + 0 \times 2 + 2 \times 1) & (1 \times 1 + 0 \times 1 + 2 \times 0) \\ (-1 \times 3 + 3 \times 2 + 1 \times 1) & (-1 \times 1 + 3 \times 1 + 1 \times 0) \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}$$

機器學習 – 矩陣及運算定義

性質 [編輯]

根據矩陣乘法的定義，單位矩陣I_n的重要性質為：

$$AI_n = A \text{且} I_n B = B$$

單位矩陣 [編輯]

條目 討論 汉 漢 臺灣正體

此條目頁的主題是主對角線元素為1、其餘元素為0的矩陣。關於所有元素皆為1的矩陣，請見「**一矩陣**」。

在線性代數中，n階**單位矩陣**，是一個$n \times n$的方形矩陣，其主對角線元素為1，其餘元素為0。單位矩陣以I_n表示；如果階數可忽略，或可由前後文確定的話，也可簡記為I[註 1]（或者E）。

$$I_1 = [1],\ I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \cdots,\ I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

一些數學書籍使用U和E（分別意為單位矩陣（*unit matrix*）和基本矩陣（*Einheitsmatrix*））， 不過I更加普遍。

機器學習 – 微積分導數

一般定義 [編輯]

直觀上 $f(x)-f(a)$代表函數值從 a 到 x 的變化量，那這樣，

$$\frac{f(x)-f(a)}{x-a}$$

代表的是從 a 到 x 的平均變化率，如果把 x 趨近於 a，似乎就可以更能貼切的描述函數值在 a 附近的變化。

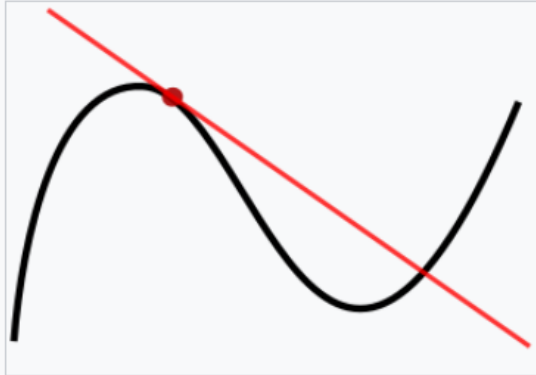
以此為動機，若實函數 f 於實數 a 有定義，且以下極限（注意這個表達式所定義的函數定義域不含 a ）

$$\lim_{x\to a}\frac{f(x)-f(a)}{x-a}$$

存在則稱 f 於 a 處**可導**，並稱這個極限為 f 於 a 處的**導數**[2]:117-118，記為$f'(a)$也可記作 $\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x=a}$ 或 $\frac{\mathrm{d}f}{\mathrm{d}x}(a)$ [1]:154。

導數（英語：derivative）是微積分學中的一個概念。函數在某一點的導數是指這個函數在這一點附近的變化率（即函數在這一點的切線斜率）。導數的本質是通過極限的概念對函數進行局部的線性逼近。當函數f的自變數在一點x_0上產生一個增量h時，函數輸出值的增量與自變數增量h的比值在h趨於0時的極限如果存在，即為f在x_0處的導數，記作$f'(x_0)$、$\frac{\mathrm{d}f}{\mathrm{d}x}(x_0)$或$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x=x_0}$。例如在運動學中，物體的位移對於時間的導數就是物體的瞬時速度[1]:153。

導數是函數的局部性質。不是所有的函數都有導數，一個函數也不一定在所有的點上都有導數。若某函數在某一點導數存在，則稱其在這一點可導(可微分)，否則稱為不可導(不可微分)。如果函數的自變數和取值都是實數的話，那麼函數在某一點的導數就是該函數所代表的曲線在這一點上的切線斜率。



一個實值函數的圖像曲線。函數在一點的導數等於它的圖像上這一點處之切線的斜率。

機器學習 – 微積分導數

基本函數的導數 [編輯]

主條目：導數列表

所謂基本函數是指一些形式簡單並且容易求出導數的函數。這些基本函數的導函數可以通過定義直接求出。

- 冪函數的導數：如果

$$f(x) = x^r,$$

其中r是任意實數，那麼

$f'(x) = rx^{r-1}$,函數f的定義域可以是整個實數域，但導函數的定義域則不一定與之相同。例如當 $r = \frac{1}{2}$時：

$$f'(x) = \frac{1}{2}x^{-\frac{1}{2}}$$ [2]:119

機器學習 – 梯度

在向量微積分中，**梯度**（英語：gradient）是一種關於多元導數的概括[1]。平常的一元（單變量）函數的導數是純量值函數，而多元函數的梯度是向量值函數。多元可微函數f在點P上的梯度，是以f在P上的偏導數為分量的向量[2]。

對向量的梯度 [編輯]

以n×1實向量**x**為變元的實純量函數f(**x**)相對於**x**的梯度為一n×1列向量**x**，定義為

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \stackrel{\text{def}}{=} \left[\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \cdots, \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right]^T = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$$

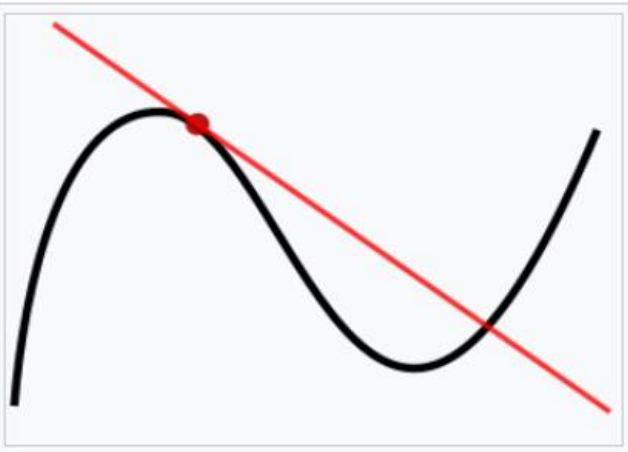
機器學習 – 梯度計算

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{b}\boldsymbol{x} + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + 1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 5 & 4 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 10 \\ 1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 1$$
$$= 5x_1^2 + 2x_2^2 + (4+3)x_1x_2 + 10x_1 + x_2 + 1$$
$$= 5x_1^2 + 2x_2^2 + 7x_1x_2 + 10x_1 + x_2 + 1$$

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_2} \end{bmatrix}$$
$$= \begin{bmatrix} \dfrac{\partial(5x_1^2 + 2x_2^2 + 7x_1x_2 + 10x_1 + x_2 + 1)}{\partial x_1} \\ \dfrac{\partial(5x_1^2 + 2x_2^2 + 7x_1x_2 + 10x_1 + x_2 + 1)}{\partial x_2} \end{bmatrix}$$
$$= \begin{bmatrix} 10x_1 + 7x_2 + 10 \\ 4x_2 + 7x_1 + 1 \end{bmatrix}$$

機器學習 – 梯度下降法

梯度下降法（英語：Gradient descent）是一個一階最佳化算法，通常也稱為**最陡下降法**，但是不該與近似積分的**最陡下降法**（英語：Method of steepest descent）混淆。 要使用梯度下降法找到一個函數的局部極小值，必須向函數上當前點對應梯度（或者是近似梯度）的*反方向*的規定步長距離點進行疊代搜索。如果相反地向梯度*正方向*疊代進行搜索，則會接近函數的局部極大值點；這個過程則被稱為**梯度上升法**。



一個實值函數的圖像曲線。函數在一點的導數等於它的圖像上這一點處之切線的斜率。

結論

- 類神經網路 DNN 都是透過數學在做運算的，所以適時的了解一些數學運算會有助於了解內部運作的方式
- 一旦了解內部運作的方式，才有機會分析模型參數，進而竄改達到攻擊者的目的。
- 明天會準備安裝 類神經網路 DNN 需要的軟體，內容會包含安裝 tensorflow、 jupyter notebook