從這篇開始我們會介紹一些在深度學習領域會使用到的一些數學知識，我們將從線性代數 (Linear Algebra) 開始。文中會適時地補上特定名詞的英文，以便未來在觀看英文文獻時更容易閱讀。

1 名詞認識 (純量、向量、矩陣)

在開始之前我們先認識幾個基本的名詞

1. Scalars 純量: 甚麼是純量呢，簡單來說就是一個單純的數字。

2. Vectors 向量: 一般來說，向量是指一個值但她不能僅用一個單純的數字大小 (scalar) 來表示同時還帶有方向性。但是在這邊我們的向量也可以是指一序列的數字組合稱為向量。通常向量會以粗斜體字代表或是在符號上機上一頂帽子如 $\hat{x}$ ，物理學上可能更多會使用箭頭的方式如 $\vec{x}$，在這裡我們就都統一用 $\hat{x}$ 來表示。那向量到底是甚麼概念呢，就是一串有序數列，表示如下

$$\hat{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

   這代表有一個向量 $\hat{x}$ 內容從 $x_1 \sim x_n$，沒錯，通常以 x_i 表示向量中的第 i 個值

3. Matrices 矩陣: 順帶一提，Matrix 正是知名電影駭客任務的原電影名稱。矩陣可以理解為 2-D 的向量，通常以斜體粗大寫表示 $\boldsymbol{A}$。假如我們有個實數 (Real-valued) 大小為 $m \times n$ 的矩陣，我們可以表示為 $A \in \mathbb{R}^{m \times n}$，而實際標示會是

$$A = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

2 線性? 代數?

其實線性代數 (Linear Algebra) 這個名稱的由來有點玄幻，簡單來說它就是為了處理數學上的線性關係 (linear relationships) 與結構 (structures)。在數學上，一個線性關係代表著一條直線 (straight-line) 關係或是一個特定關係可以直線形式表示都可以稱為線性關係。那何為線性呢？線性 (Linear) 在高等數學中有著兩個特殊定義 - Additivity 相加性: 簡單來說就是 $f(x+y) = f(x) + f(y)$，這邊的 f 可以是函數。- Homogeneity 均勻性: 簡單來說就是 $f(mx) = mf(x)$

原則上只要符合這兩個條件就符合線性的概念。那這些又跟向量、矩陣有甚麼關係呢，讓我娓娓道來。我們可以想像一個線性方程組

$$\begin{cases} 2x + y = 20 \\ 4y + 3z = 20 \\ 2x + z = 10 \end{cases}$$

如果我們將係數 (coefficient) 分離並整理成向量與矩陣，就會變成以下

$$A = \begin{bmatrix} 2 & 1 \\ 4 & 3 \\ 2 & 1 \end{bmatrix} \hat{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \hat{b} = \begin{bmatrix} 20 \\ 20 \\ 10 \end{bmatrix}$$

有了這三個東西之後，我們就可以把我們的方程組簡化成

$$Ax = b$$

是不是像極了我們常見的一般線性方程，這就是線性代數在做的事，因此向量與 (尤其是) 矩陣的運算在線性代數中是非常重要的角色，所以把線性代數當成一門研究矩陣與向量計算的數學分之也是一個很正常的事。

“線性代數是一門研究向量與線性函數的學科 “

3 Vector 向量

從這開始我們會詳細的介紹各個關鍵的線性代數觀念和計算方法，從向量開始！

3.1 Vector Addition (向量加法)

向量的加法非常好理解，用一個舉例就可以了

1. Number Addition: $1 + 2 = 3$
2. Vector Addition:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$$

舉一個簡單的實例，假如我們要把兩個多項式函數相加

$$f_1(x) = 2x^3 + 3x^2 + x + 7$$

$$f_2(x) = 4x^3 + 2x^2 + 4x + 1$$

可以係數分離向量表示為

$$\hat{f_1} = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 7 \end{bmatrix}, \hat{f_2} = \begin{bmatrix} 4 \\ 2 \\ 4 \\ 1 \end{bmatrix}$$

那麼 $f_1(x) + f_2(x)$ 可以用 $\hat{f_1} + \hat{f_2}$ 來取得

$$\begin{bmatrix} 2 \\ 3 \\ 1 \\ 7 \end{bmatrix} + \begin{bmatrix} 4 \\ 2 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \\ 5 \\ 8 \end{bmatrix}$$

因此

$$f_{1+2}(x) = f_1(x) + f_2(x) = 6x^3 + 5x^2 + 5x + 8$$

這只是個簡單的例子，還有非常多其他的用法。

3.2 Vector Multiplication 向量乘法

首先我們看到係數積 (scalar multiplication)，簡單來說就是

$$2\begin{bmatrix}2\\3\\1\\7\end{bmatrix} = \begin{bmatrix}4\\6\\2\\14\end{bmatrix}$$

了解係數積之後就可以更好的理解一個方程式，如下

$$\begin{cases}2x + 10y = 20\\3x + 2y = 7\end{cases} \Rightarrow x\begin{bmatrix}2\\3\end{bmatrix} + y\begin{bmatrix}10\\2\end{bmatrix} = \begin{bmatrix}20\\7\end{bmatrix}$$

就是這樣子的概念。接著我們看到**向量內積 (dot production)** 兩個向量是可以做乘法的，其中一種稱為內積，以下說明。首先我們要有兩個向量

$$\hat{a} = \begin{bmatrix}1\\2\\3\end{bmatrix}, \hat{b} = \begin{bmatrix}3\\4\\5\end{bmatrix}$$

則

$$ab = a_1b_1 + a_2b_2 + \cdots + a_nb_n = \sum_{i=1}^{n} a_ib_i$$

以上面的例子來看 ab 就是 24。

4 Matrix 矩陣

4.1 單位矩陣 Identity Matrix (Unit Matrix)

矩陣乘法有非常多不同的種類，如阿達瑪乘積等，但是我們要學的是一般的矩陣乘法。首先我們要知道矩陣乘法有個前提就是被乘矩陣的行數 (column) 要跟乘矩陣的列數 (row) 一樣，簡單來說就是 $A_{m\times n} \times B_{n\times l}$，則 $AB = C_{m\times l}$。接下來就可以來做運算了

$$(AB)_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

舉個例子

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}$$

從這邊看，可以發現矩陣乘法是沒辦法符合交換律的，因為交換的話，結果會不一樣，即便如此，仍然符合結合律和分配律，即

$$(AB)C = (AB)C$$

$$(A + B)C = AC + BC$$

$$C(A + B) = CA + CB$$

4.2 單位矩陣 Identity Matrix (Unit Matrix)

單位矩陣是一種特殊的方陣，最明顯的特徵就是他從左上角到右下角都是 1，除此之外的元素都是 0，舉個例子

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

*from wikipedia

那他有甚麼用呢？正如他的名字，單位，他就是類似一個單位的矩陣，甚麼意思？就是任一矩陣與他的乘積將會是原本的矩陣。(通常以 I_n 表示 $n \times n$ 的 Identity Matrix)，舉例，假設有一矩陣 $A_{m\times n}$

$$I_m A = A I_n = A.$$

4.3 行列式 Determinant

行列式是一種能夠從**方陣**計算出來的一個純量，可視為此矩陣的純亮表示。假設有一矩陣 A 我們通常以 $\det(A)$，或是兩條豎線表示 $|A|$，如下

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix},\ \det(A) = |A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}$$

那麼該如何計算行列式呢，一般來說會使用萊布尼茨公式來求，對於一個 n 階方陣 A，其行列式如下

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

其中的 $\prod$ 符號是大寫的 π，為求積符號，類似於乘法版的 Σ。其中的 $\sigma \in S_n$ 的 S_n 是一個自然數集合，其內容為 $\{1, 2, ..., n\}$，$\sum_{\sigma \in S_n}$ 表示將集合內元素遍歷求和。接下來的 sgn() 是符號差，想要理解這個概念我們就必須知道甚麼是奇、偶排列。集合 $\{1, 2, ..., n\}$ 的有序排列 (由小到大)，我們稱為 $1, 2, ..., n$ 的排列 (permutation)，而這些數字可以被重新排列出 $n!$ 種排列方式，而一個排列只要出現大數排在小數前方如 $(2, 1, 3)$，這種情況稱為逆序，繼續觀察，會發現對第 1 個元素 2 後方來說有一個數字出現了逆序也就是 1，所以該元素的逆序數量就是 1 個。舉個例子，求 $(4, 3, 5, 2, 1)$ 之逆序，我們可以一個一個元素來看，第一元素 4 的後方有三個逆序 $(3, 2, 1)$，第二元素 3 後面有兩個逆序 $(2, 1)$，依此類推，第三元素 5 有兩個 $(2, 1)$，第四元素 2 有一個 (1)，最後一個元素沒有逆序，因此 $(4, 3, 5, 2, 1)$ 之逆序有 $3 + 2 + 2 + 1 = 8$ 共 8 個，8 是偶數，所以 $(4, 3, 5, 2, 1)$ 是偶排列，反之如果算出來是奇數個那就是奇排列。然而 sgn 函數在遇到偶排列時會成為 1，奇排列為 -1。說了這麼多可能有點難理解，因此我們舉個方陣來求他的行列式

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

大概就是這樣，可以套進去剛剛的公式做驗算會比較好理解。然而我們實際在計算行列式的時候是非常難計算的，因此我們可以使用 numpy 來處理，程式碼如下

```
import numpy as np

matrix = np.array([[2, 3],
                   [4, 5]])
print(np.linalg.det(matrix)) # 2*5 - 3*4
```

使用 ‘det()‘ 函數就可以求得該矩陣的行列式。

4.4 逆矩陣 Inverse Matrix

一般的數學計算我們都會用到倒數的概念，如 2 的倒數正是 $\frac{1}{2}$，然而在矩陣當中也有倒數的概念，不過稱為逆矩陣 (Inverse Matrix)，因為矩陣不能當除數，所以會表示成 A^{-1}，其中 A 為一個矩陣。就如同一般的數字倒數一樣，倒數與原數相乘會是 1，如

$$2 \times \frac{1}{2} = 1$$

逆矩陣也一樣，相乘會變成前面提到的 ** 單位矩陣 (Unit Matrix)**，如下

$$A \times A^{-1} = I$$

看起來是個是個非常簡單的概念，但是在計算上是相當複雜的，以下舉個尋找 2×2 Matrix 的 Inverse Matrix。

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

或是使用稍早提到的行列式來表達

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

舉個例子

$$A = \begin{bmatrix} 5 & 1 \\ 3 & 1 \end{bmatrix} \quad A^{-1} = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -3 & 5 \end{bmatrix} = \begin{bmatrix} 0.5 & -0.5 \\ -1.5 & 2.5 \end{bmatrix}$$

從這邊我們就可以成功找到 A 的逆矩陣，而他們兩個相乘會變成

$$AA^{-1} = \begin{bmatrix} 5 & 1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & -0.5 \\ -1.5 & 2.5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

也就是 I_2。除此之外，也可以發現 $ad - bc$ 若小於 0 (即行列式為 0)，是無法找到逆矩陣的。我們也可以使用 numpy 來求得 inverse matrix，如下

```
import numpy as np
matrix = np.array([[5, 1], [3, 1]])
print(np.linalg.inv(matrix))
```

就可以求得了

5 結語

這次我們淺略的介紹一些線性代數的基本概念，在以後的 AI 中可能會用到，可以看到再回來這邊好好詳讀。

6 Reference

1. https://web.math.sinica.edu.tw/math_media/d52/5201.pdf
2. http://www.wunan.com.tw/www2/download/preview/2C78.PDF
3. https://www.scribd.com/document/467089621/linear-guest-pdf
4. https://www.deeplearningbook.org/