

# RNN、LSTM 和GRU神經網路

吳庭育

[tyw@mail.npust.edu.tw](mailto:tyw@mail.npust.edu.tw)



# 序列資料－空間和順序關係

- 卷積神經網路主要是在處理空間關係的問題，也就是說，在 $W \times W$ 矩陣上分佈的像素（Pixels）是有意義的，我們是使用這些像素組合出圖片上的圖形，如果打亂這些像素的位置，就不會是原來的圖片



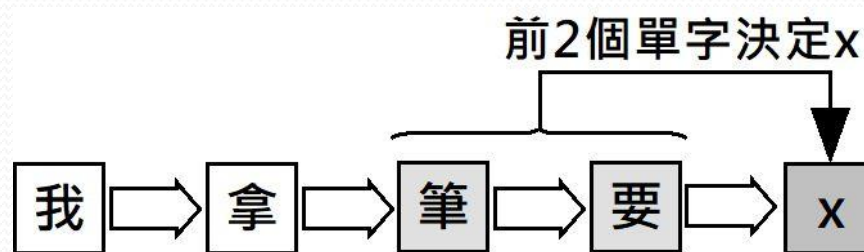
# 序列資料－空間和順序關係

- 對於傳統神經網路和卷積神經網路來說，我們重視的是每一隻螞蟻的位置，即以空間關係形成的圖形。問題是當你觀察一群行進中的螞蟻，這些螞蟻部隊是有順序性（Order）的。
- 每一隻行走螞蟻的頭是跟隨著前一隻螞蟻的尾，如果有任何一隻螞蟻轉了方向，就會影響之後所有螞蟻的行走方向，不同於空間關係，有些資料的前後關聯性是有意義的，想想看！如果你將一句話的單字調來調去，馬上就失去句子原來的意義，同理，股價如果將時間順序打亂，我們畫出的分析線型就沒有任何意義，循環神經網路就是設計用來處理這種有順序性的資料，稱為序列資料(Sequence Data)。



# 什麼是序列資料

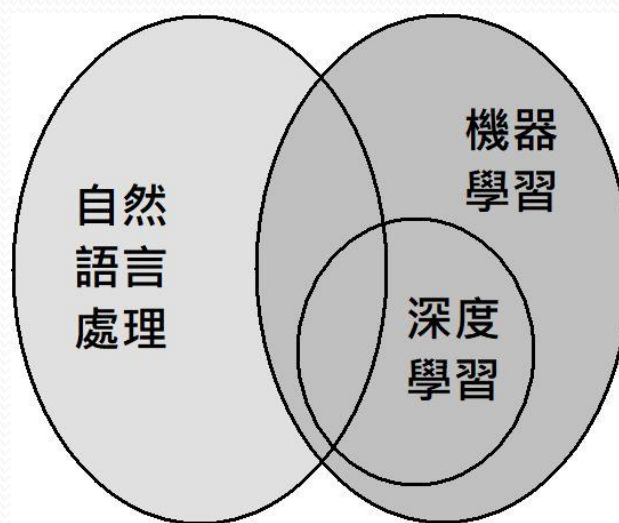
- 序列資料(Sequence Data)就是一種有順序的向量資料(不一定是時間順序)，簡單的說，序列資料是在資料前後位置之間擁有關聯性，例如：DNA序列是一種與時間序無關的序列資料(與前後位置順序有關)，自然語言的句子也是一種序列資料，其相關程度可以使用N-gram模型來判斷，分為和前一個單字有關的2-gram ( Bi-gram )和2個有關的3-gram(Tri-gram)。





# 自然語言處理

- 自然語言處理(Natural Language Processing, NLP)是計算機科學一個很大的研究領域，機器學習和深度學習事實上只涉及自然語言處理的一部分：





# 認識自然語言處理

- 自然語言處理（Natural Language Processing）就是在處理人類語言和文字的序列資料，其目的是讓電腦能夠了解語言，和使用語言來進行對話：
  - 了解自然語言(Natural Language Understanding)：系統能夠了解語言，包含口語、文章、語法、語意和直譯等。
  - 產生自然語言(Natural Language Generation)：系統能夠使用語言或文字來回應，包含產生單字、有意義的片語和句子等。



# 機器學習在自然語言處理的應用

- 機器學習/深度學習使用在自然語言處理的常用領域，如下所示：
  - 文件分類與資訊擷取。
  - 機器翻譯。
  - 語音辨識。
  - 語句和語意分析。
  - 拼字與文法檢查。
  - 問答系統－聊天機器人。

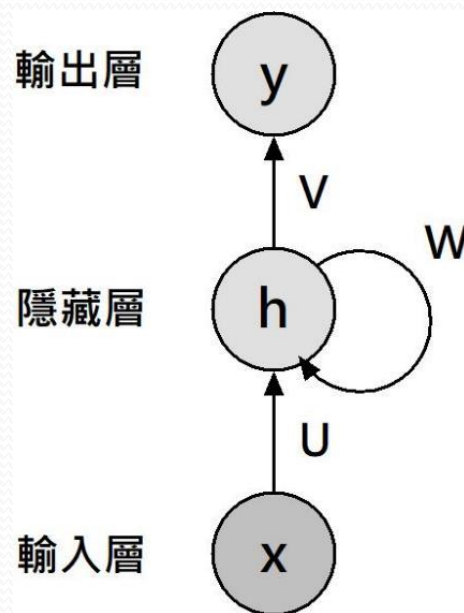


# 循環神經網路的結構

- 多層感知器的傳統神經網路和卷積神經網路都是一種前饋神經網路(Feedforward Neural Network, FNN)，在訓練過程的輸入和輸出資料是相互獨立，並不會保留任何狀態，也就是說，這種神經網路沒有記憶能力，無法處理擁有順序關係的序列資料(Sequence Data)。
- 不同於傳統神經網路，循環神經網路是一種擁有記憶能力的神經網路，能夠累積之前輸出的資料來分析目前的資料，即處理序列資料。

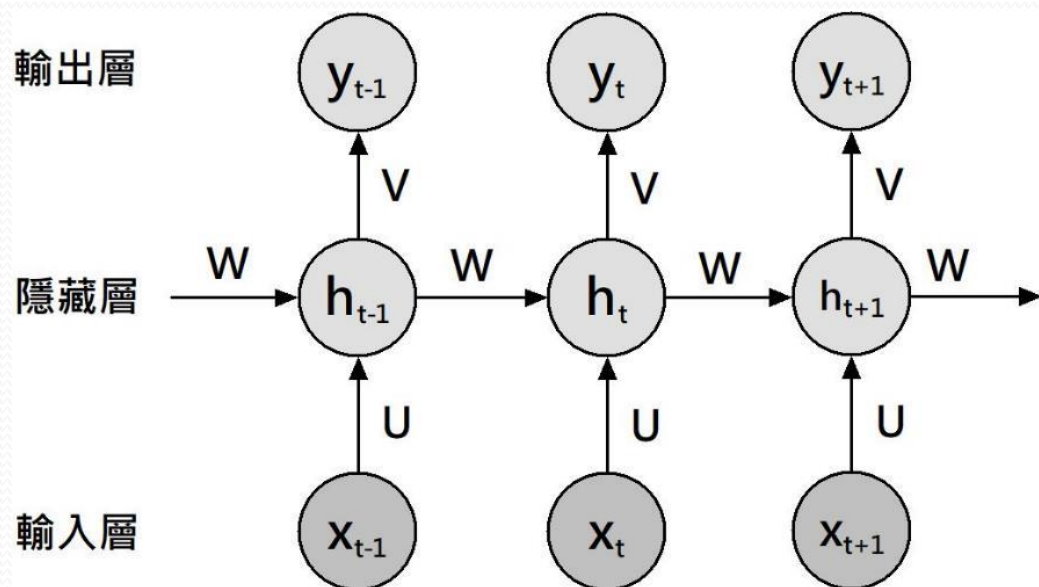
# 循環神經網路的基本結構

- 循環神經網路的結構就是一組全連接的神經網路集合，每一個神經網路的隱藏層輸出，同時也是下一個神經網路的輸入。



# 展開循環神經網路的隱藏層

- 因為隱藏層 $h$ 有一個時步的迴圈，當我們展開隱藏層時，就可以看到循環神經網路是一組三層神經網路的集合。



# 循環神經網路的結構

- 循環神經網路的前向傳播與反向傳播( $t=0$ )
  - 針對循環神經網路前向傳播，在時步 $t=0$ ，我們隨機初始 $U$ 、 $V$ 和 $W$ 的權重（為了方便說明，沒有使用偏向量）， $h_0$ 的隱藏層輸出通常初始成0，在時步 $t=1$ 時的隱藏層輸出 $h_1$ 和輸出層輸出 $y_1$ ：

$$\begin{aligned}h_1 &= f(U \bullet x_1 + W \bullet h_0) \\ y_1 &= g(V \bullet h_1)\end{aligned}$$

- 循環神經網路的前向傳播與反向傳播(時步 $t$ )
  - 循環神經網路在時步 $t$ 計算預測值 $y_t$ 的公式

$$\begin{aligned}h_t &= f(U \bullet x_t + W \bullet h_{t-1}) \\ y_t &= g(V \bullet h_t)\end{aligned}$$

# 循環神經網路的前向傳播與反向傳播(損失分數)

- 循環神經網路損失函數的損失分數計算

$$E = \sum_{i=1}^t f_e(y_i - t_i) +$$

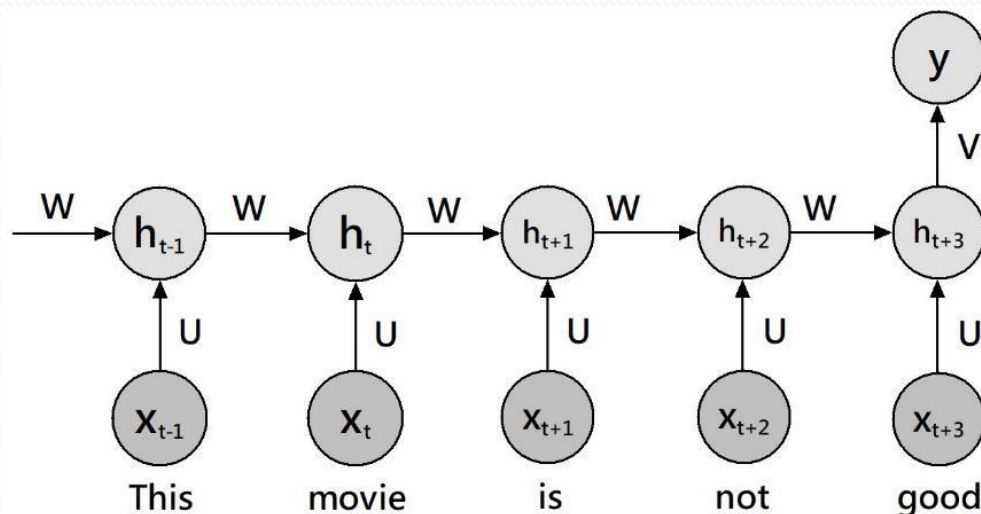


# 循環神經網路的情緒分析範例

- 循環神經網路有很多種，輸出層和輸入層的長度不見的相同，例如：情緒分析的循環神經網路可以分析英文句子的情緒是正面或負面：

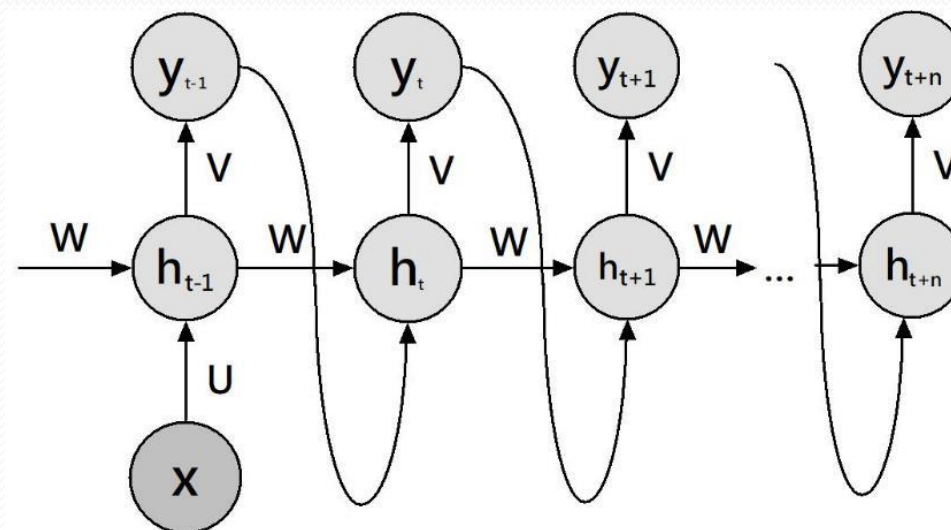
**This movie is not good.**

- 上述句子的標籤是負面情緒，用來分析上述英文句子的循環神經網路：



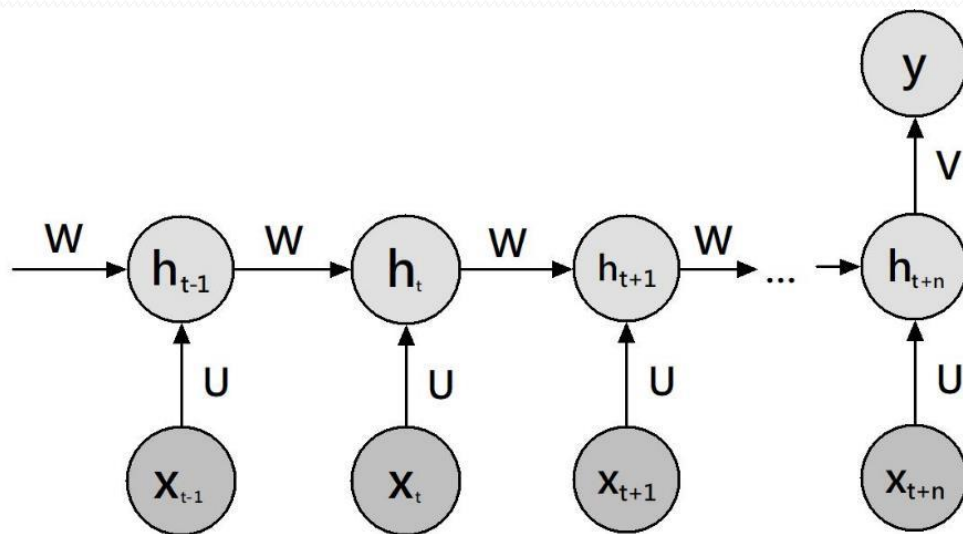
# 循環神經網路的種類 – 一對多(One to Many)

- 一對多循環神經網路有一個輸入和序列資料的輸出，這類型網路的目的是產生序列資料，例如：一張圖片的輸入可以產生圖片說明文字的序列資料，或產生音樂等。



## 循環神經網路的種類－多對一(Many to One)

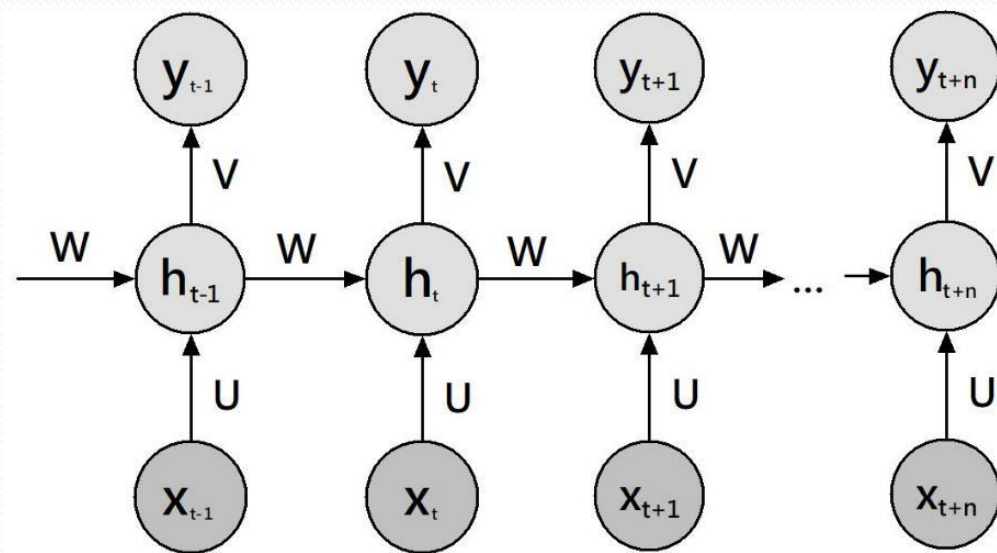
- 多對一循環神經網路是序列資料的輸入，但只產生一個輸出，這類型網路主要是使用在情緒分析，例如：輸入電影評論描述文字，可以輸出正面或負面情緒的結果。



# 循環神經網路的種類－多對多(Many to Many)

## 輸入和輸出等長

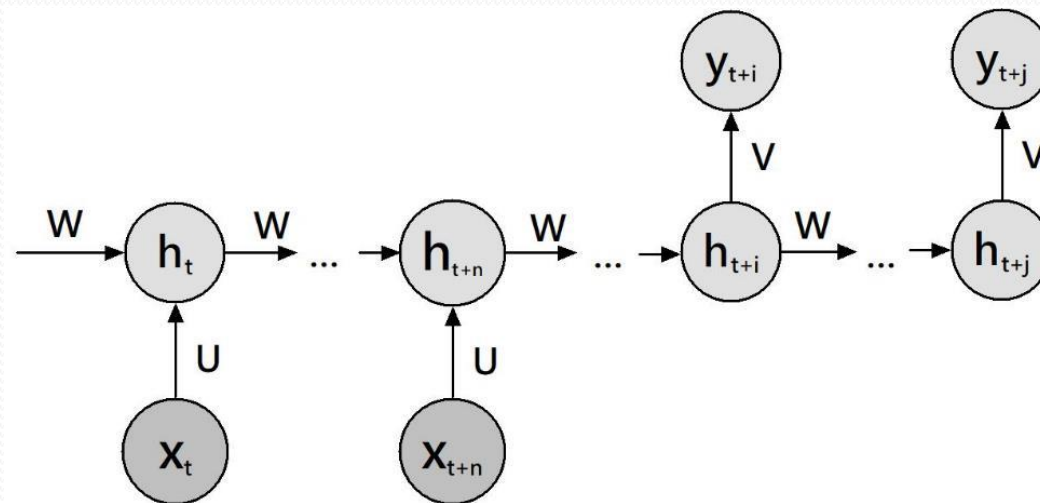
- 輸入和輸出等長：這就是每一個輸入都有對應的輸出，此時的每一個輸出如同是二元分類，例如：判斷每一個位置的單字是否是一個人名。



# 循環神經網路的種類－多對多(Many to Many)

## 輸入與輸出不等長

- 輸入與輸出不等長：因為輸入和輸出的序列資料長度不同，最常見是使用在機器翻譯，例如：將中文句子翻譯成英文，通常句子的序列資料不會是相同長度。




## 循環神經網路的梯度消失問題 (1/3)

- 循環神經網路雖然能夠保留之前累積的資訊，幫助我們解決現在的問題，例如：使用之前電影畫面的資訊，幫助我們理解目前畫面的劇情，問題是循環神經網路的表現並不好，其記憶能力只能保留很短時步的資訊，如同人類年齡大了，記性就不好，很容易健忘。

The clouds are in the \_\_\_\_.

The clouds are in the sky.



## 循環神經網路的梯度消失問題 (2/3)

- 英文句子是使用上下文來預測最後的單字，因為答案sky和相關訊息clouds間隔的時步並不長，循環神經網路足以勝任此單字的預測工作。如果英文句子上下文相關的訊息十分的長，間隔很長的時步。

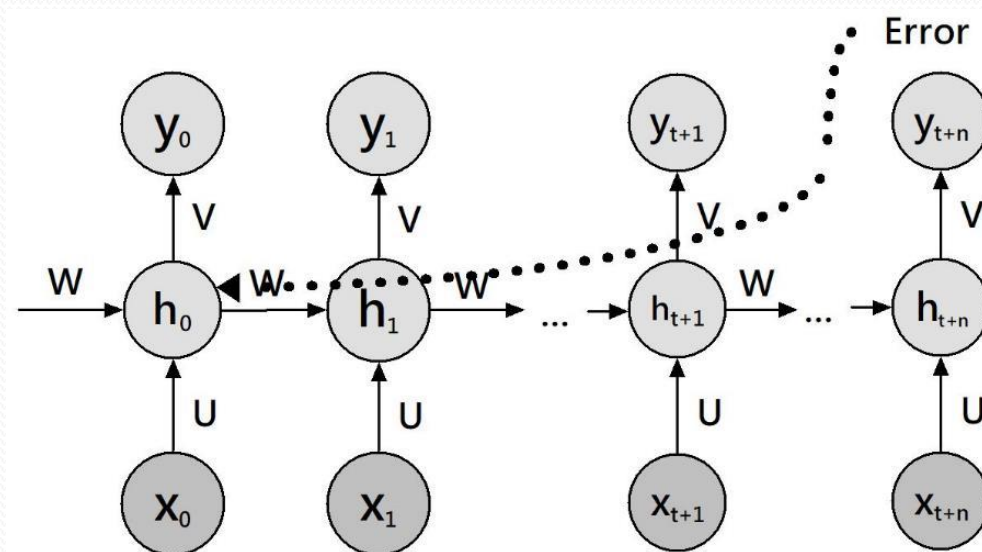
The cat which already ate, ..., was full.

The cats which already at, ..., were full.



## 循環神經網路的梯度消失問題 (3/3)

- was或were需要依據之前間隔非常長時步的cat或cats來判斷其詞性，在這種情況下，循環神經網路很難將間隔如此之遠的資訊連接起來，這也是梯度消失問題(Vanishing Gradient Problem)造成的結果。



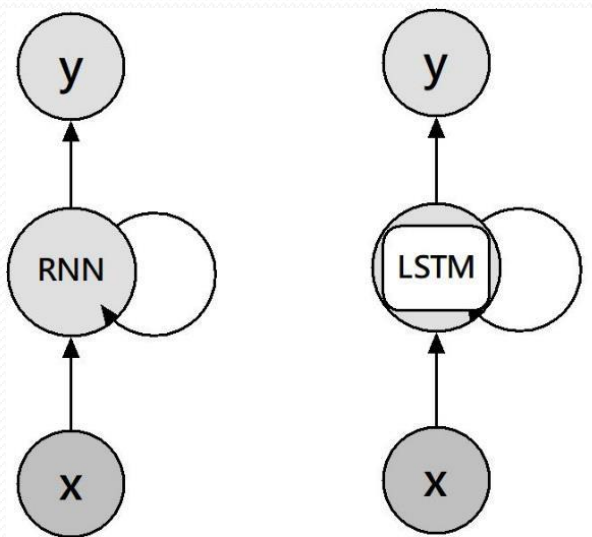
# 長短期記憶神經網路(LSTM)

# 長短期記憶神經網路的結構

- 「長短期記憶神經網路」(Long Short-term Memory, LSTM) 改良RNN，這是一種擁有長期記憶能力的神經網路，因為RNN在結構上會產生梯度消失問題，所以實務上的循環神經網路通常不是使用RNN。
- 長短期記憶神經網路(LSTM)是德國的兩位科學家Hochreiter和Schmidhuber為了解決循環神經網路的梯度消失問題，所開發出的一種循環神經網路結構。LSTM的作法是建立一條如同輸送帶的長期記憶線，然後使用多個不同閘門(Gate)來篩選處理需要長期記憶的資料，如同人腦的海馬迴負責短期記憶和長期記憶的處理。

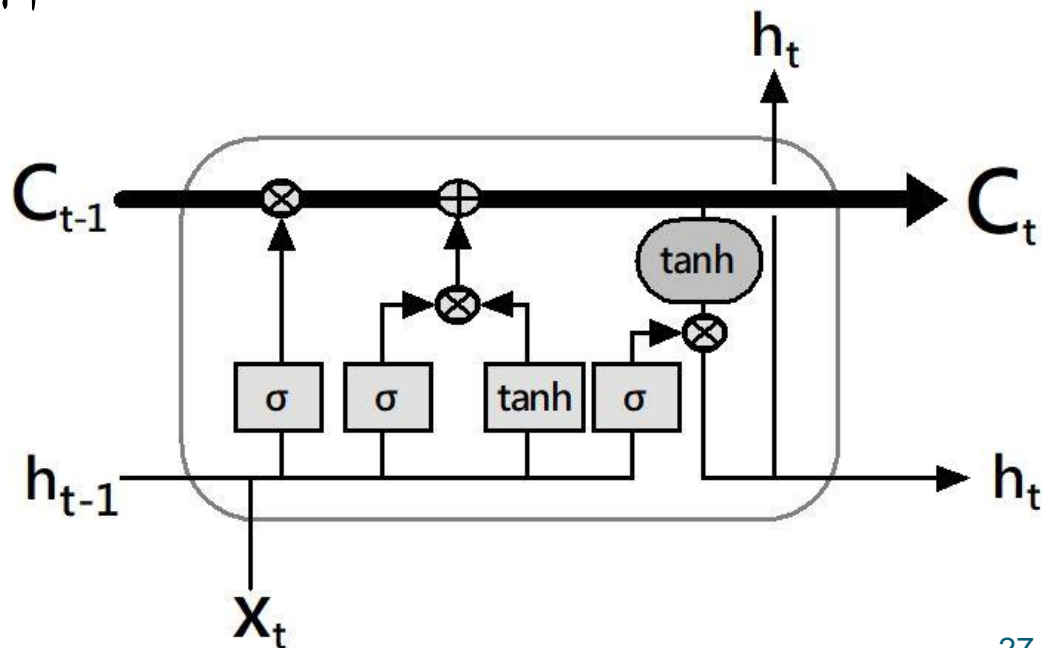
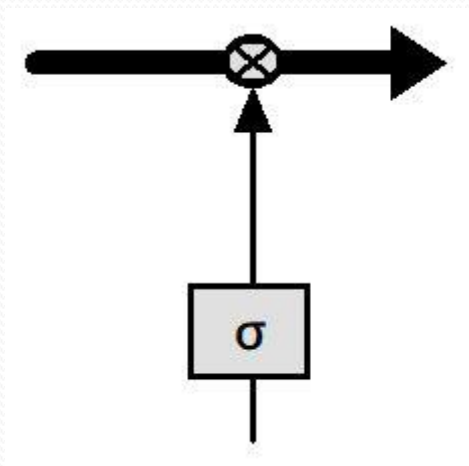
# LSTM單元

- 長短期記憶神經網路和循環神經網路的結構並沒有什麼不同，只是循環神經網路的隱藏層只有一層神經層，長短期記憶神經網路的隱藏層是一個LSTM單元(LSTM Cell)。



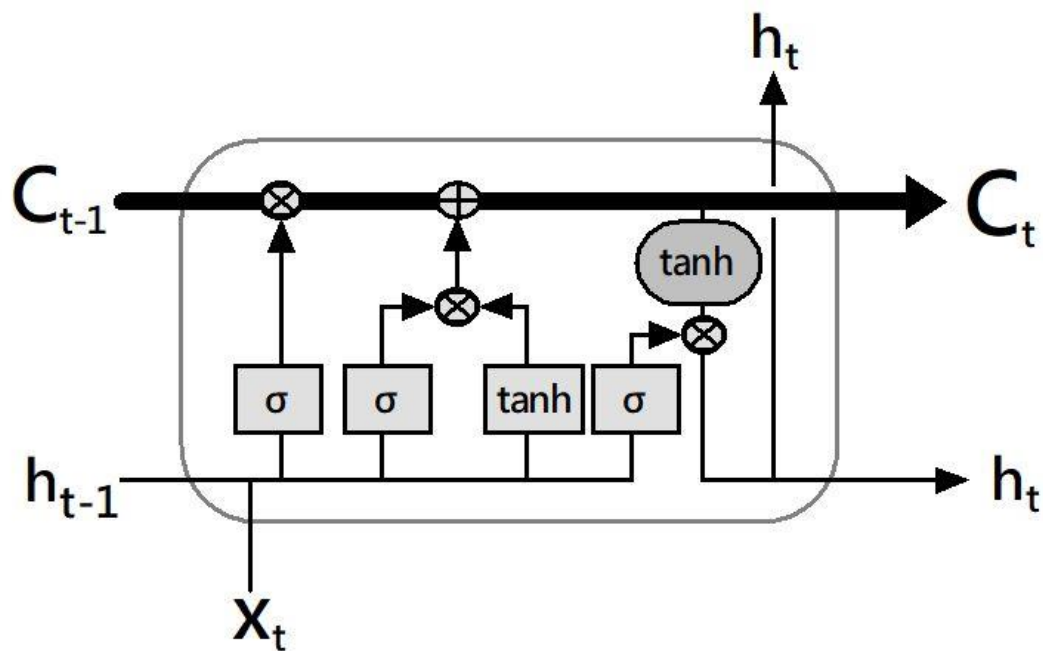
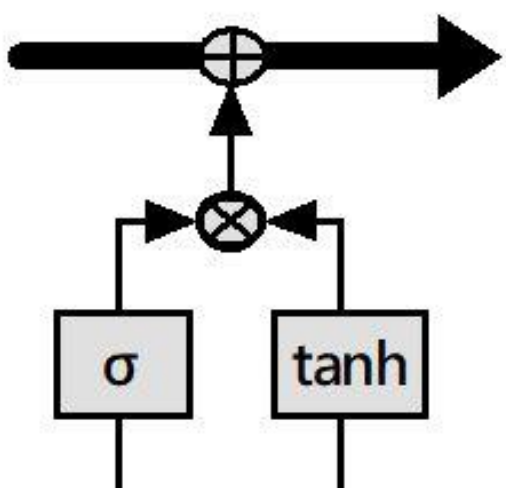
# LSTM單元

- LSTM單元不只一層神經層，而是4層神經層。
- Sigmoid神經層：3個 $\sigma$ 符號的小方框是3種閘門的神經層，使用Sigmoid函數的值0~1來控制資料通過的比例，如同開啟閘門的大小，0是關閉不讓任何資料通過；1是開啟表示全部通過，然後與資料執行逐元素相乘，例如：刪除上方輸送帶保留的記憶資料。



# LSTM單元

- Tanh神經層：Sigmoid神經層只是決定通過哪些資料，我們還需要Tanh啟動函數的神經層來取得欲通過的候選資料，才能使用逐元素相乘 建立資料後，再以逐元素相加 來更新上方輸送帶保留的記憶資料。



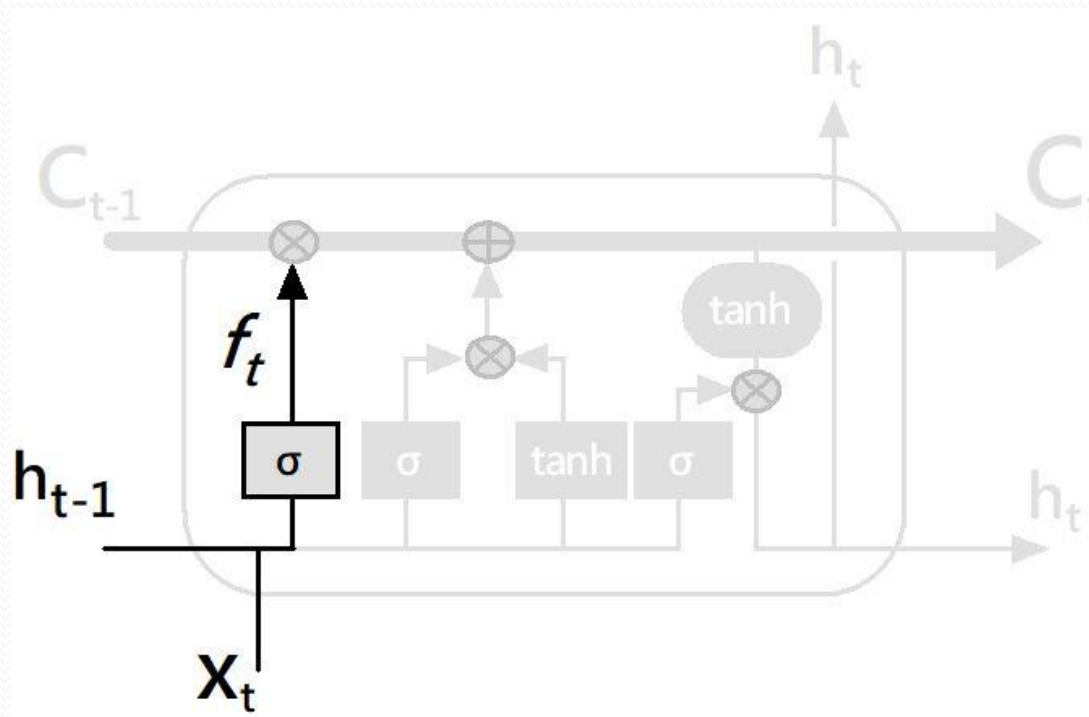
# 長短期記憶神經網路的運作機制簡介

- 在了解LSTM單元的結構後，我們就來看一看長短期記憶神經網路的運作機制，也就是輸入閘、遺忘閘和輸出閘這三種閘門是如何運作來建立擁有長期記憶力的神經網路。
- 基本上，長短期記憶神經網路的運作機制可以想像是一部很多集的電視劇，在每一集都有主角、配角和跑龍套等多種角色會登場，但是，並不是所有角色都會和下一集有關係，我們需要記得一些和之後集數有關係的角色，以便在下一集或之後集數能夠看懂劇情，這些角色就是儲存在LSTM單元上方的長期記憶線中。



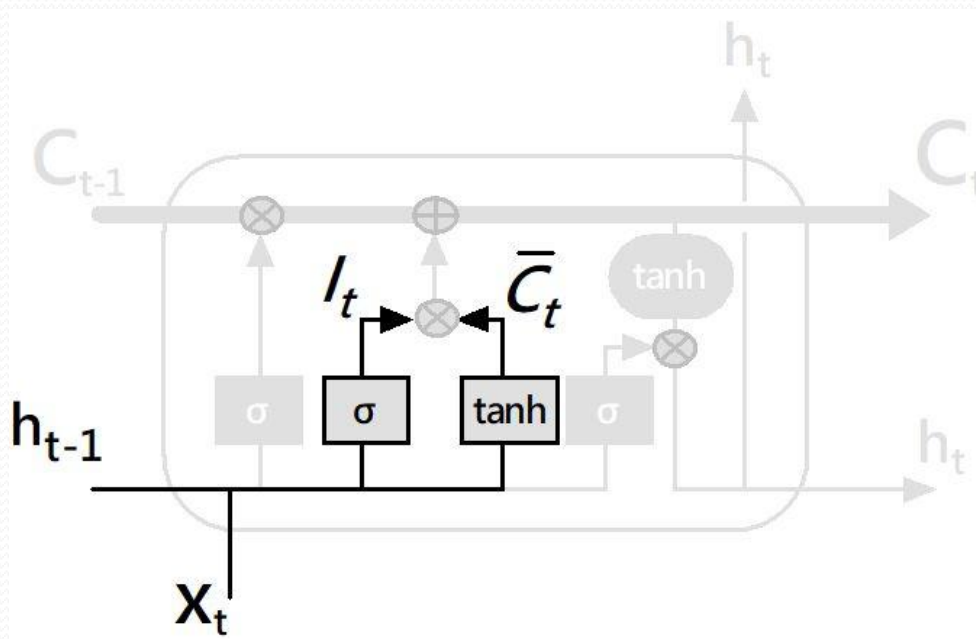
## 遺忘閘(Forget Gate)

- 遺忘閘是用來決定保留哪些資料；哪些資料可以忘掉，也就是從長期記憶線中刪除資料。遺忘閘的輸入資料是合併  $h_{t-1}$  和  $x_t$  成為  $[h_{t-1}, x_t]$  向量。



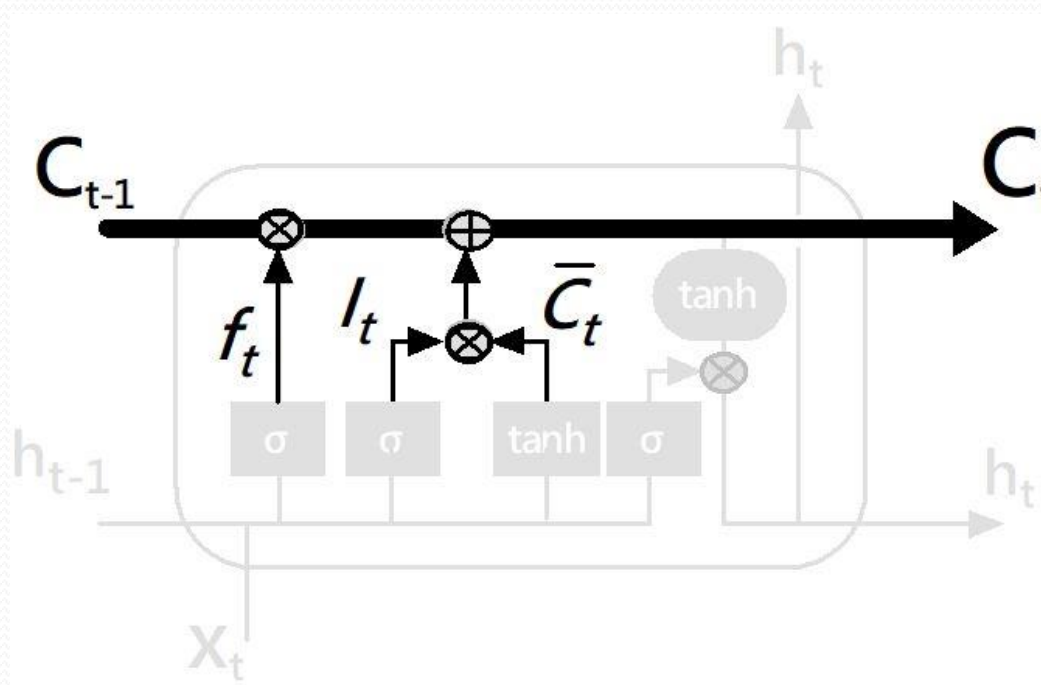
## 輸入閘(Input Gate)

- 在輸入閘決定需要更新長期記憶線中的哪些資料，包含新增資料和需替換的資料。輸入閘的輸入資料也是合併 $h_{t-1}$ 和 $x_t$ 成為 $[h_{t-1}, x_t]$ 向量。



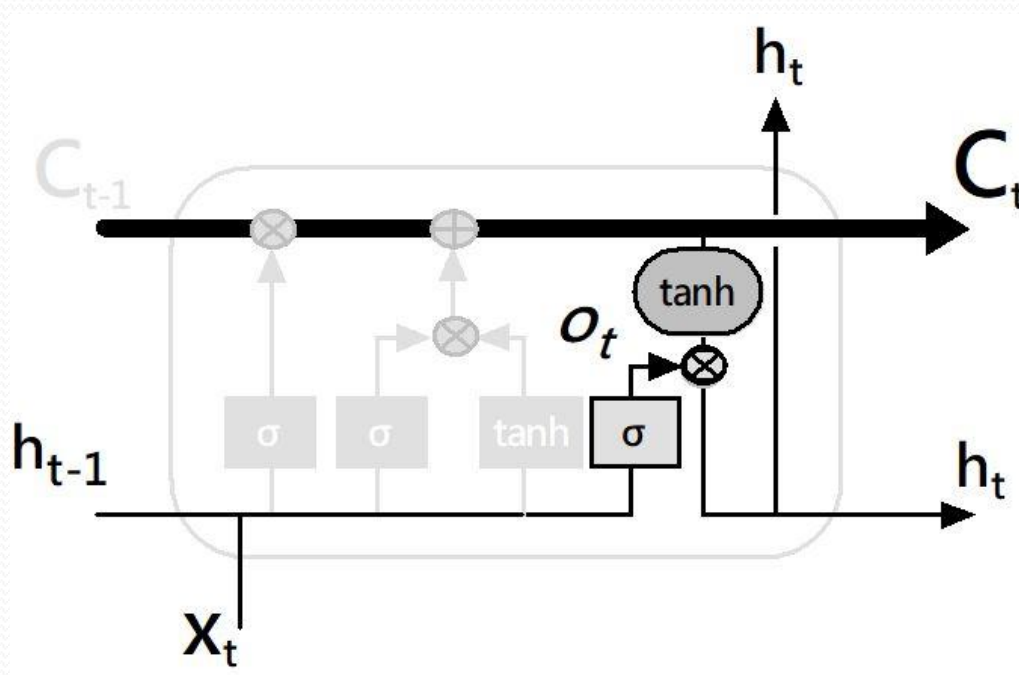
## 輸入閘(Input Gate)

- 我們已經計算出長期記憶線欲刪除和更新的資料，接著可以更新長期記憶線從前一時步 $t-1$ 的 $C_{t-1}$ 至目前時步 $t$ 的 $C_t$ ，  
。



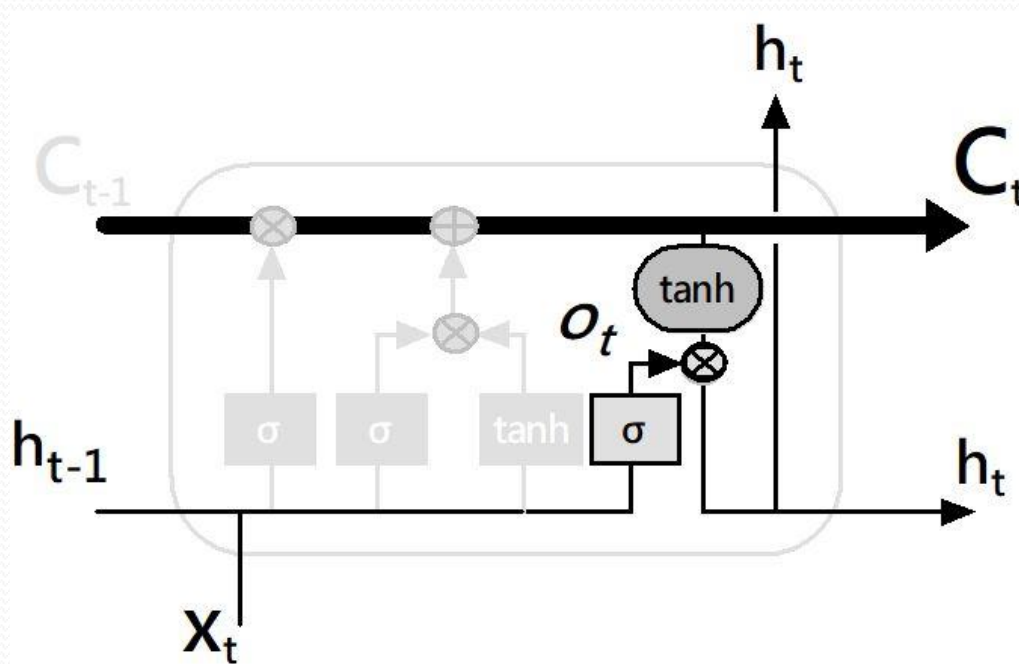
## 輸出閘(Output Gate)

- 輸出閘是用來決定從長期記憶線 $C_t$ 中有哪些資料需要輸出至下一個時步 $t+1$ ，作為下一個時步 $t+1$ 的輸入資料。



## 輸出閘(Output Gate)

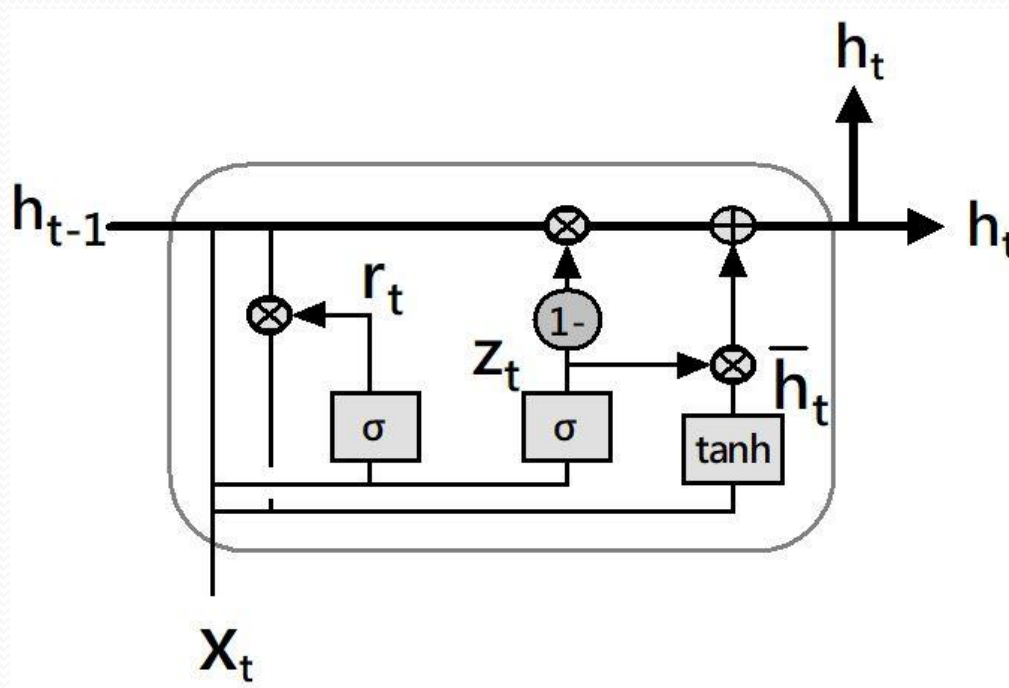
- 輸出閘是用來決定從長期記憶線 $C_t$ 中有哪些資料需要輸出至下一個時步 $t+1$ ，作為下一個時步 $t+1$ 的輸入資料。



# 閘門循環單元神經網路(GRU)

## 輸出閘(Output Gate)

- LSTM還有一個兄弟神經網路稱為「閘門循環單元神經網路」(Gated Recurrent Unit, GRU)，這是2014年Kyunghyun Cho主導團隊所提出的一種LSTM更新版，一個比LSTM結構更簡單的版本，可以提供更快的執行速度，和減少記憶體的使用。
- 如同LSTM的LSTM單元，GRU的基本結構是GRU單元(GRU Cell)。





## 重設閘(Reset Gate)

- 重設閘(Reset Gate)：GRU使用重設閘決定是否將之前記憶忘掉，換句話說， $r_t$ 就是控制保留多少之前的記憶資料，如果 $r_t$ 是0，就表示忘掉之前的所有記憶，然後重設成目前輸入資料的狀態。是Sigmoid函數，「 $\bullet$ 」是點積運算， $[h_{t-1}, x_t]$ 向量是輸入資料， $W_r$ 是重設閘權重； $b_r$ 是重設閘偏向量，其計算公式如下所示：

$$r_t = \sigma(W_r \bullet [h_{t-1}, x_t] + b_r)$$

## 更新閘(Update Gate)

- 更新閘(Update Gate)：GRU是透過更新閘來控制記憶資料的保留與更新。 $W_z$ 是更新閘權重； $b_z$ 是更新閘偏向量，其計算公式如下所示：

$$z_t = \sigma(W_z \bullet [h_{t-1}, x_t] + b_z)$$

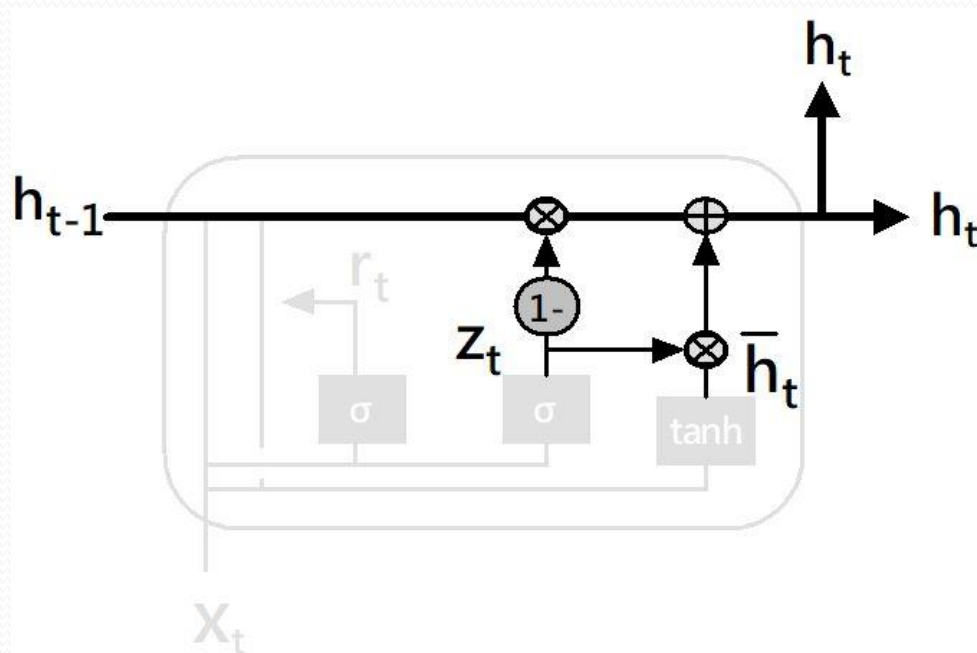
## Tanh神經層

- Tanh神經層：在Tanh神經層產生最後需輸出的候選資料，其輸入資料是將重設閘保留的之前記憶和目前的輸入資料合併成 $[r_t * h_{t-1}, x_t]$ 向量。 $W_h$ 是Tanh神經層的權重； $b_h$ 是Tanh神經層的偏向量，其計算公式如下所示：

$$\bar{h}_t = \tanh(W_h \bullet [r_t * h_{t-1}, x_t] + b_h)$$

# GRU單元的輸出資料

- 最後，我們可以計算出GRU單元的輸出資料 $h_t$ ：



$$h_t = (1 - z_t) * h_{t-1} + z_t * \bar{h}_t$$



```

1 import numpy as np
2
3 samples = ["I hated this movie",
4            "This movie is not good"]
5
6 token_index = {}
7
8 def word_tokenize(text):
9     text = text.lower()
10    return text.split()
11
12 for text in samples:
13     for word in word_tokenize(text):
14         if word not in token_index:
15             token_index[word] = len(token_index) + 1
16
17 print(token_index)
18
19 max_length = 6
20 results = np.zeros((len(samples), max_length,
21                    max(token_index.values())+1 ))
22
23 for i, text in enumerate(samples):
24     words = list(enumerate(word_tokenize(text))[:max_length])
25     for j, word in words:
26         index = token_index.get(word)
27         results[i, j, index] = 1.0
28
29 print(results[0])

```

```

{'i': 1, 'hated': 2, 'this': 3, 'movie': 4, 'is': 5, 'not': 6, 'good': 7}
[[0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]]

```

# 文字資料的One-hot編碼 -

## 英文單字的One-hot編碼

- 我們如何將英文句子以單字為單位的One-hot編碼，例如：  
現在有2個英文句子的樣本資料：

I hated this movie.

This movie is not good.

- 上述英文句子沒有區分英文大小寫（都轉成小寫）和標點符號，我們可以將2個樣本的英文句子分割成7個不重複的英文單字，稱為「分詞」（Tokenization）或斷詞：

i、hated、this、movie、is、not、good



# 英文單字的One-hot編碼

- 一樣可以使用One-hot編碼，將這些英文單字向量化（在向量的第1個元素並沒有使用），如下表所示

英文單字	One-hot編碼
i	[0. 1. 0. 0. 0. 0. 0. 0.]
hated	[0. 0. 1. 0. 0. 0. 0. 0.]
this	[0. 0. 0. 1. 0. 0. 0. 0.]
movie	[0. 0. 0. 0. 1. 0. 0. 0.]
is	[0. 0. 0. 0. 0. 1. 0. 0.]
not	[0. 0. 0. 0. 0. 0. 1. 0.]
good	[0. 0. 0. 0. 0. 0. 0. 1.]

# Python程式實作英文句子的One-hot編碼

- Python程式首先使用字串函式分割單字來建立樣本的單字索引，然後依據單字索引轉換整個英文句子成為One-hot編碼。

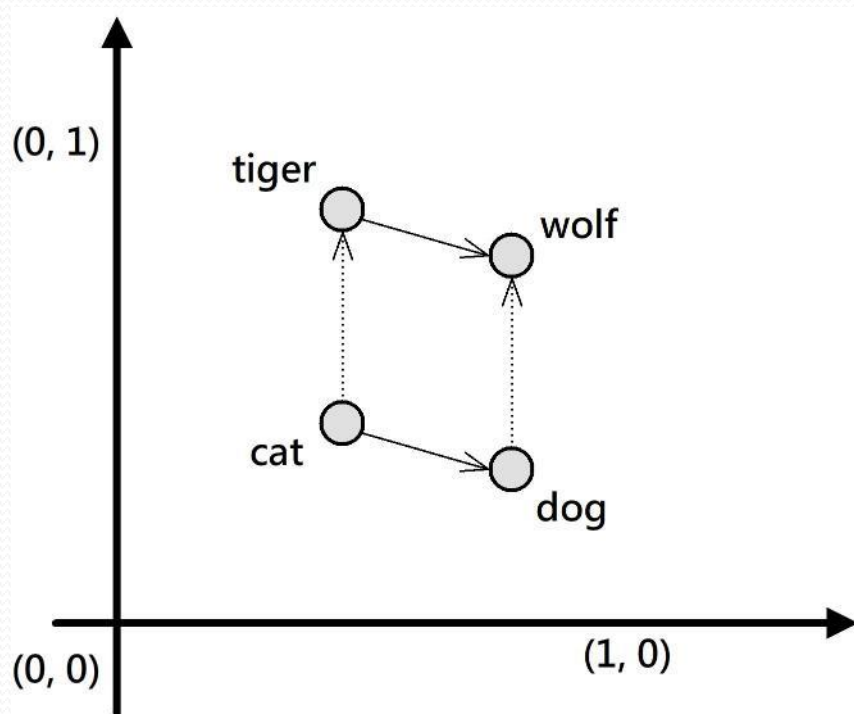


# 詞向量與詞嵌入

- 詞向量（Word Vector）或稱為詞嵌入（Word Embedding）也是一種文字資料向量化的方法，可以將單字嵌入一個浮點數的數學空間中。假設：現在有10,000個不同單字（詞庫），分別使用One-hot編碼和詞向量（使用200個神經元的隱藏層）執行文字資料向量化的差異：
  - One-hot編碼需要使用程式碼轉換單字成為向量；詞向量是建立神經網路來自行學習單字的詞向量。
  - One-hot編碼建立的是一個高維度的稀疏矩陣（每一個向量長10,000，其中只有1個1；其他都是0），以此例是10,000x10,000，即10,000個單字，每一個單字是長度10,000的向量；詞向量是低維度浮點數的緊密矩陣，因為隱藏層是200個神經元，可以壓縮成10,000x200，10,000個單字，每一個單字是長度200的向量。

# 詞向量的幾何意義

- 詞向量就是將人類的自然語言對應到幾何空間，使用空間來表示單字之間的關係，例如：英文同義字會轉換成相近的詞向量。現在，我們有4個英文單字 wolf、tiger、dog、cat，其轉換成的2D幾何空間，



# 詞向量的幾何意義

- 一個英文句子如下所示：

Mary goes crazy about deep learning.

- 在上述英文句子如果使用3個單字的窗格，即從單字goes來預測其周圍單字Mary和crazy，共有兩種作法如下：
  - CBOW模型(Continuous Bag-of-Words model)：使用周圍單字來預測中間的單字，例如：使用Mary和crazy預測中間的goes單字，goes是目標資料；其他2個單字是輸入資料。
  - Skip-gram模型(Skip-gram model)：源於N-gram模型，我們可以使用一個單字來預測周圍的單字，例如：使用goes單字預測Mary和crazy兩個周圍的單字，此時的goes單字是輸入資料；其他2個單字是目標資料(轉換成輸出機率)。

# 使用神經網路學習詞向量

- 當使用CBOW或Skip-gram模型建立神經網路所需的訓練資料後，我們可以建立神經網路來學習詞向量，例如：輸入層是單字數10,000個神經元；隱藏層是200個神經元（最後詞向量的維度）；輸出層也是單字數的10,000個神經元的三層神經網路。

