語言處理

語意

語意，白話文的說就是『感受』，人在聽到一段話的時候，你的大腦做出判斷的時候其實並不是看詞語的表象，而是這個詞語帶給你大腦什麼感受(開心/困惑...等等)

文本翻譯
文本摘要
情緒分析
相似比較
模糊搜索
推薦系統

序列

『詞彙』，在很多情況出現的『順序』不一樣會帶來不同的結果，e.g. 我/喜歡/你 和 你/喜歡/我 就是不同的意思

語意的重要

我有四種詞，於是我定義了四個輸入

我　　表示成 [1, 0, 0, 0]
很　　表示成 [0, 1, 0, 0]
喜歡　表示成 [0, 0, 1, 0]
你　　表示成 [0, 0, 0, 1]

?

問題1: 我想要比較每個詞的相似度

無法！因為每一個詞距離都一模一樣

?

問題2: 我想要當成其他機器學習的輸入

無法！因為維度太高了，你根本
只有一個維度有值，其他都是0

語意

語意

用另外的方式來說，其實就是把『抽象的感受』化成是『具象的數字』

以前的作法

人訂立的對照表

喜歡

5	0	2	0
正面	負面	開心	難過

問題在於：同一個詞不同情境，不同任務可能有不同的語意，而且人類也無法定義到底有幾種感受

深度學習

我/喜歡/你 → 降維演算法 → 5 2 1 3

正面/負面 → 目標演算法 → 2 1 0 2

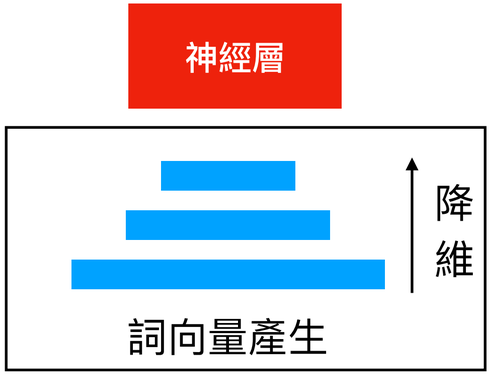
8 2 1 2

深度學習的重點就是讓網路根據『參數』和『正確距離』來自己調整參數，因此訓練出的語意就跟你的輸出是有關係的，而且也可以網路找出何謂是『正確感受』

深度學習作法

基本長相

目標輸出

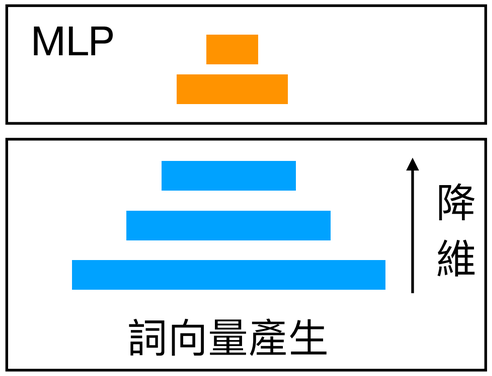


一群的詞

1. 利用『梯度下降』調整參數訓練前面的詞向量產生器
2. 最經典的是：後面的輸出層只是一個『工具人』為了幫我們訓練前面的產生器而已，訓練完我就可以丟掉它

監督式學習

分類問題

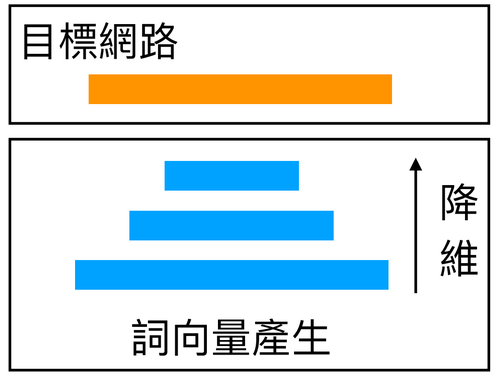


一篇文章的詞

1. 輸出可以是 正/負 或者是 新聞分類，利用分類來訓練前面的詞向量產生
2. 所以『語意』是基於你的『輸出』主題的，所以你的相似性會是『基於這主題的相似性』

非監督式學習

詞3



詞1 / 詞2 / 詞4 / 詞5

1. 通常『輸入 - 輸出』就會是『上下文 - 目標』或是『目標 - 上下文』
2. 所以『語意』是基於你的『上下文』的，所以你的相似性會是『基於上下文的相似性』

Word2Vec

Word2Vec是由網路訓練『語意提取』最有名的始祖，至今仍然是非常好用的一套方法

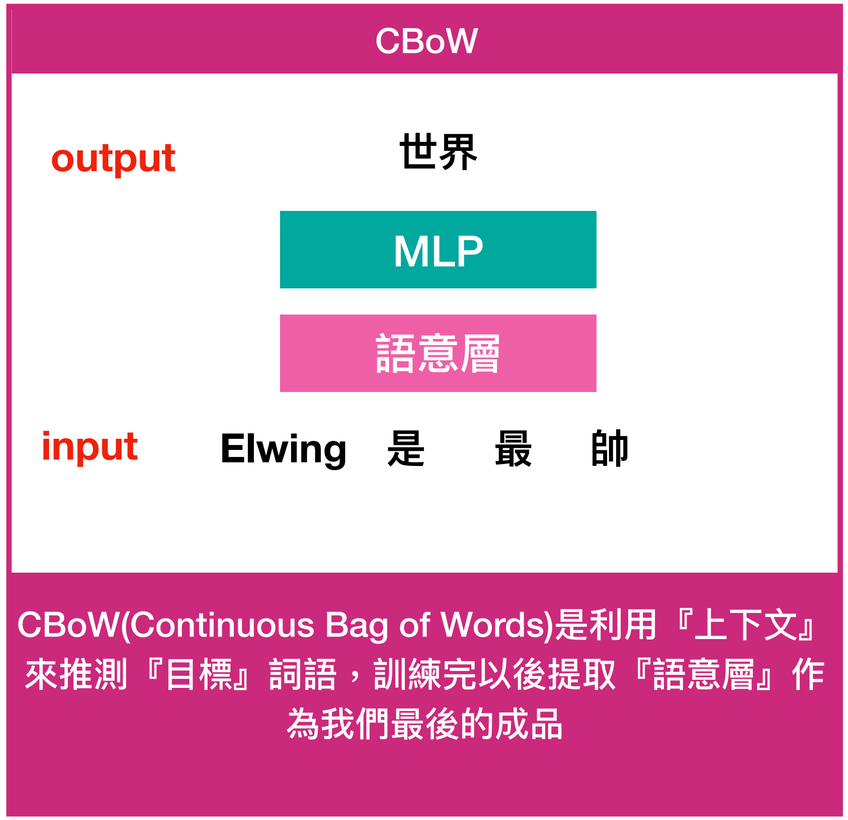
不過 Word2Vec 雖然是使用『神經網路』的方式，但她卻比較算是『淺層網路』只有一層的隱藏層，不過我們驚訝的發現，其實他的效果真的是不錯！

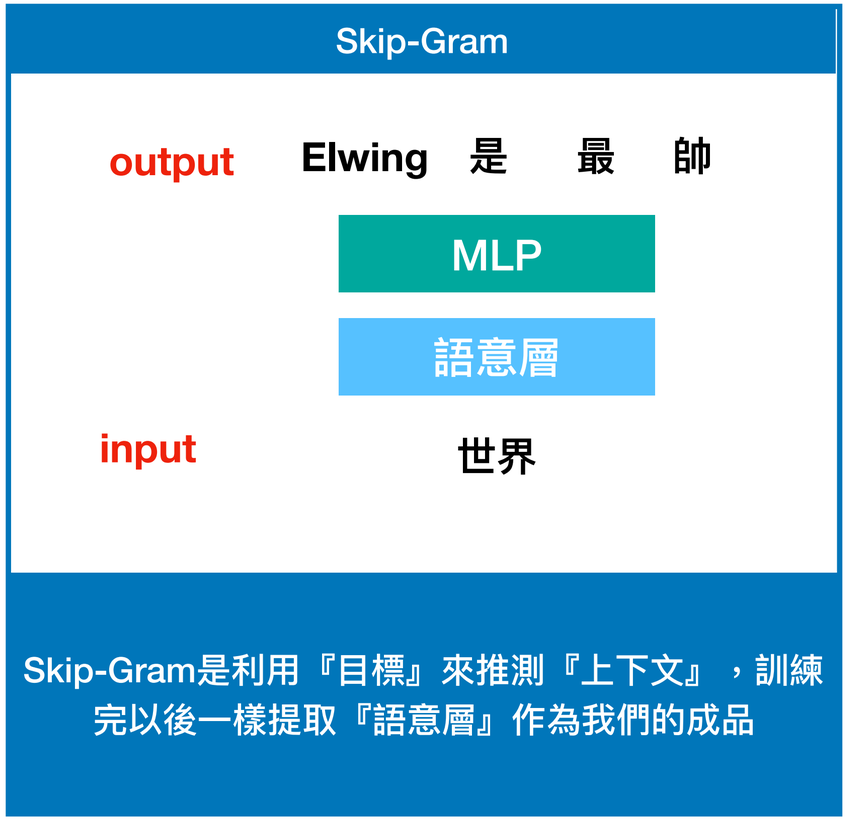
Word2Vec 採用的方式是『非監督式』的方式，利用『上下文』和『對象』的關係來構築『對象』的語意

Word2Vec種類

Word2Vec當時提出兩種不一樣的『上下文模型』，
一種叫做『CBoW』，一種叫做『Skip-Gram』

例句：Elwing / 是 / 世界 / 最 / 帥 / 的 / 男人





Word2Vec語意

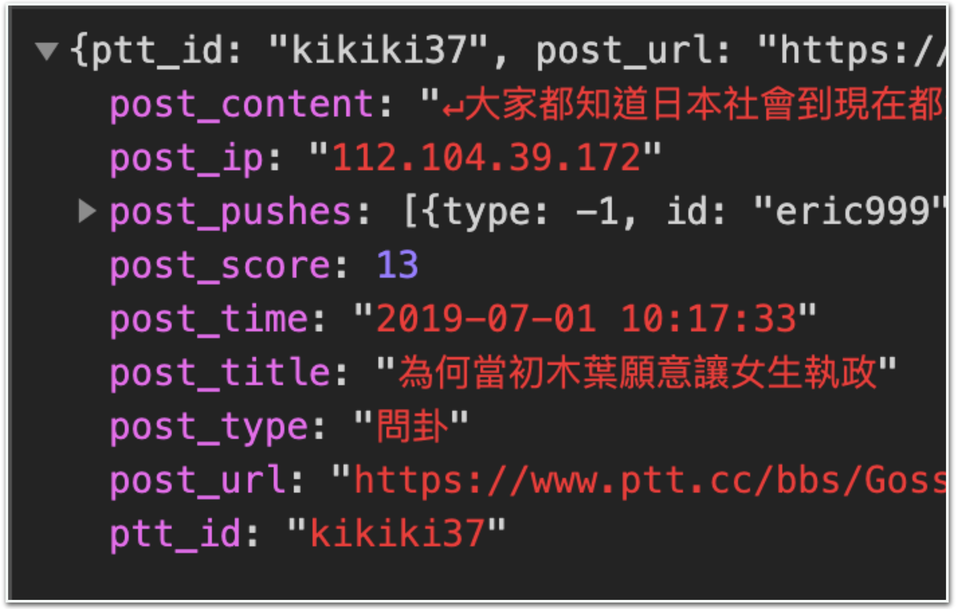
所以Word2Vec的語意會跟你想像的有點不一樣，他的語意會比較像是跟上下文的『共現性』，像『喜歡』和『討厭』就不是完全相反的語意，因為他們出現的時機點其實很像的

兔子 是 一種 __ 的 生物

在空格裡可以常常填入的詞可能是『傲嬌』或者是『兇狠』，因此這兩個詞轉換成語意就會有一定程度上的相似

來實作吧

我們使用的是我自己抓取的PTT文章，包含使用者IP，主題，發文日期，內容...等等無所不包



利用瀏覽器觀察JSON格式

Gensim

我們可以使用 NLP 裡最常用的函式庫 Gensim 來讀取 word2vec 的向量格式，不過你要把你的向量儲存成這樣！

xx.txt

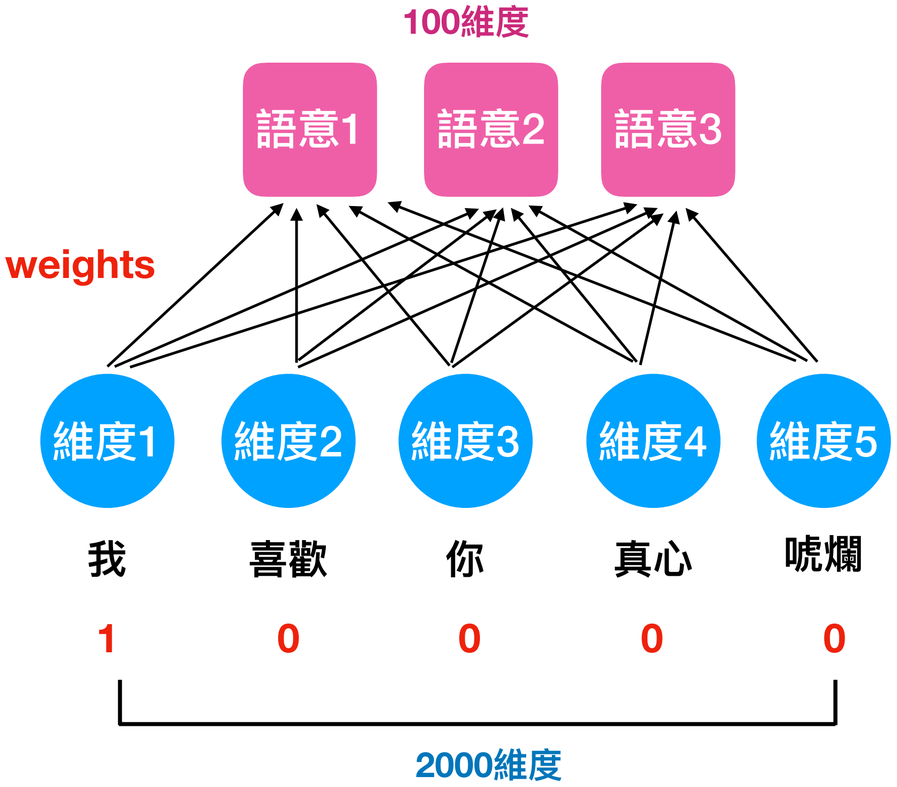
第一行 詞數 向量維度

後續行 我 1 2 0.5 ...

你 2 1 0.5 ...

得到詞向量

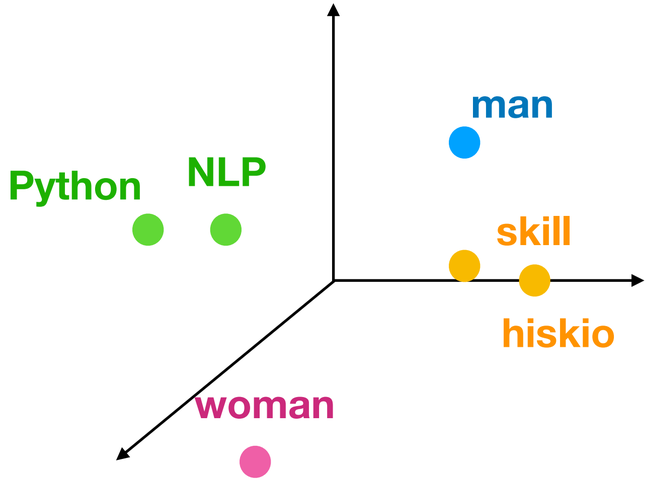
其實weights就是你的詞向量



一個詞『我』

相似度

有了感受我們就可以把感受排列在空間中，比較兩個感受的相似度或者計算兩個感受間的差距



有了詞向量
我們就可以把詞表示在空間中!

cos距離
只計算方向，不計算大小
-1(180度,最不相似)
1(0度,最相似)

Subsampling

許多敏銳的同學一定會問，有些『出現頻率高』的詞，不就總是出現在『上文』和『下文』，這樣我們的每一個詞得到的『詞向量』不就會被這些『較無貢獻』的詞影響嗎？

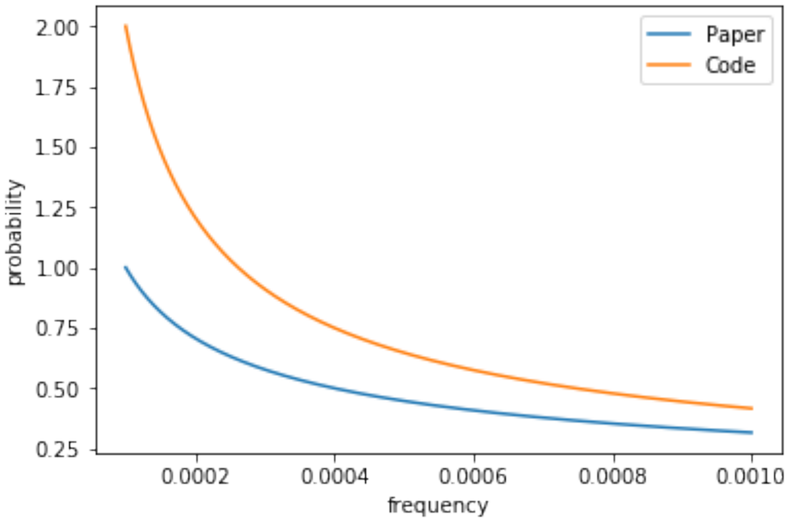
事實上，Word2Vec已經幫你考慮過這個問題了，對於每一個詞，他都會設置一個 『取樣機率』 (留下來的機率) 這個機率基本上會跟『出現頻率』成反比

(論文) 保留機率P ＝ $\sqrt{\frac{t}{f(w_i)}}$

t = 常數(論文給出經驗數值10^-5)

(實作) 保留機率P ＝ $(\sqrt{\frac{sample}{freq(w_i)}} + \frac{sample}{freq(w_i)})$

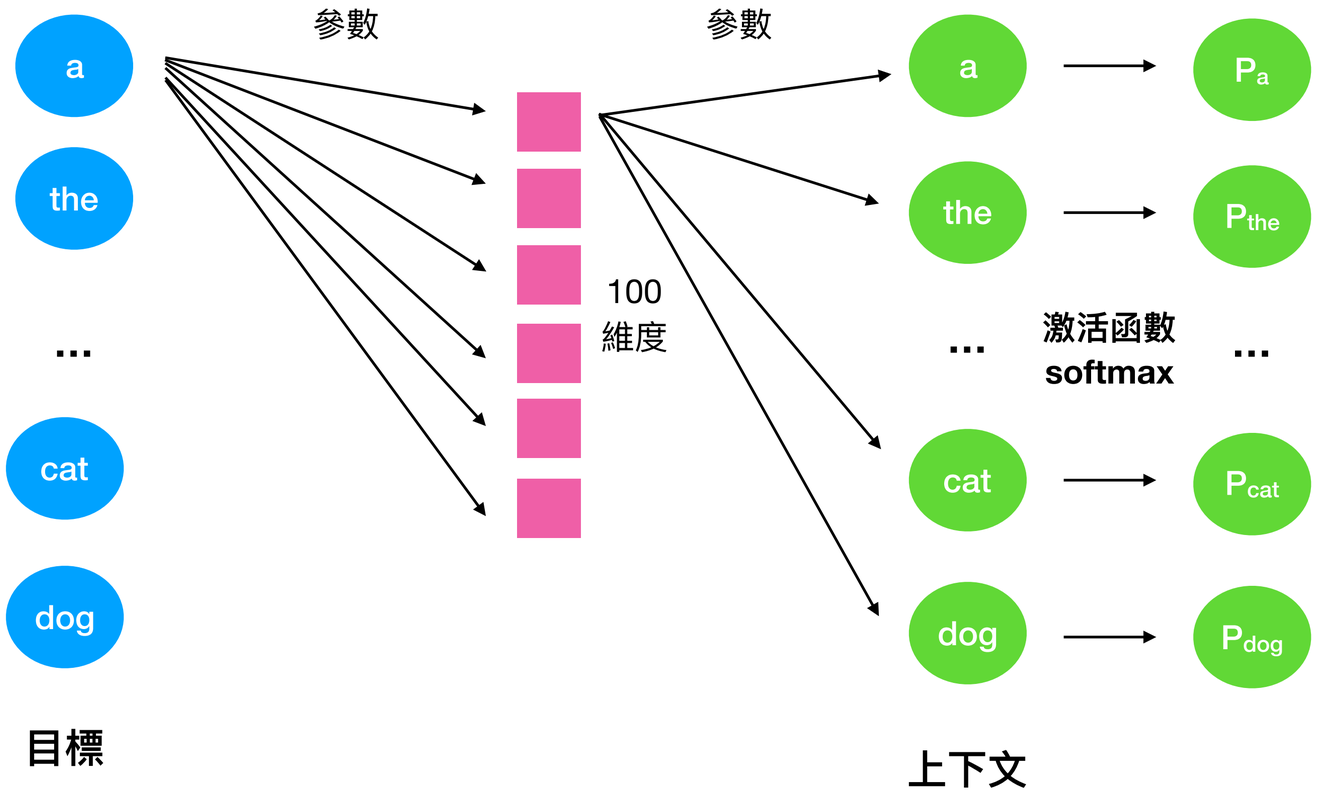
sample = 常數(經驗數值10^-4)



常數都在10^-4時的比較，可以看到程式碼對於低頻的保留率較高

訓練時間問題

我們發現，我們自製的 Word2Vec 訓練的非常非常慢，原因是為什麼呢？



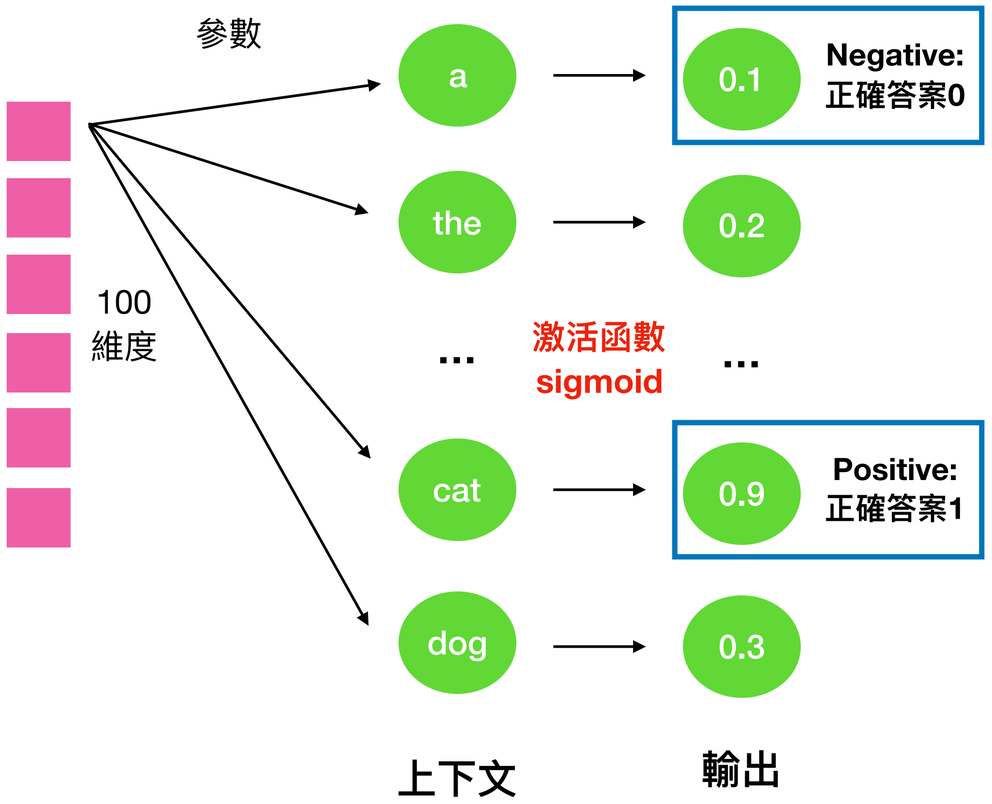
softmax的問題

$$softmax = \frac{e^x}{\sum e^{x_i}}$$

softmax為了讓加起來等於1，所以必須把前面所有的分數都考慮進去，也就意味著每一次的計算你都要計算最後一層的機率以及更新所有的參數，這計算量是超級超級大

Negative Sampling

第一種改進的方法我們叫做 Negative Sampling，也是我們最常使用的一種方式



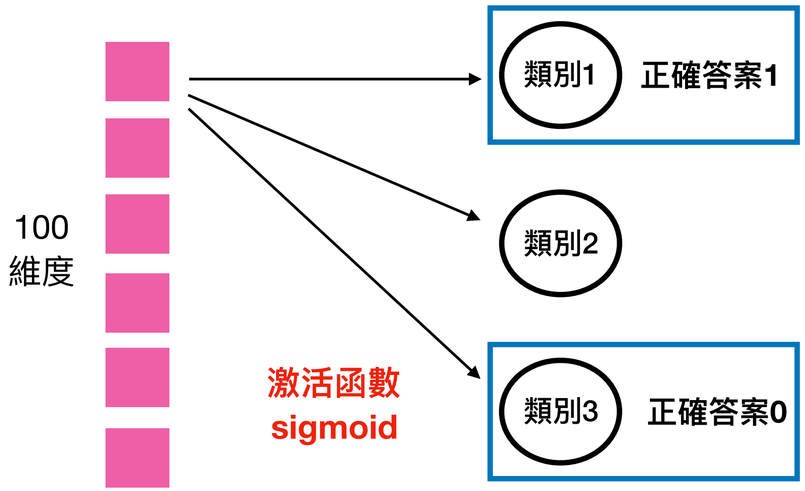
Negative Sampling

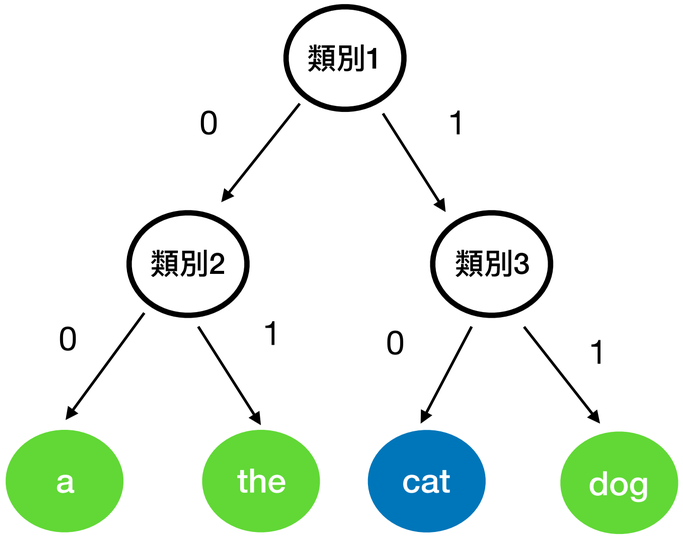
$$sigmoid = \frac{1}{1+e^{-x}}$$

使用 sigmoid 就相當於我們有很多個『二元分類器』，1就是『是』，0就是『否』，而我們除了更新那個該是 1 的輸出以外，還會隨機選擇一些 0 做更新，這些隨機選的 0 我們就叫做 Negative Samples

Hierarchical Softmax

Hierarchical Softmax的想法比較特別，他還是讓完整的機率等於1，但他使用了一個樹形的結構 Huffman Tree (類似決策樹) 來儲存每個詞





n個輸出會有n-1個中間類別

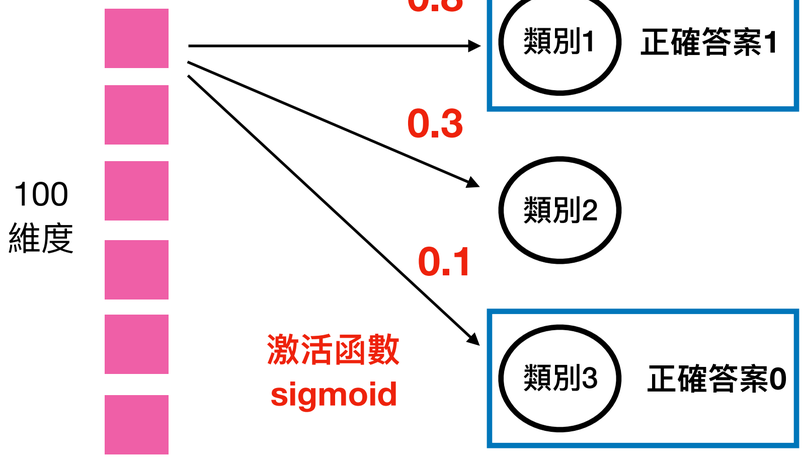
Hierarchical Softmax

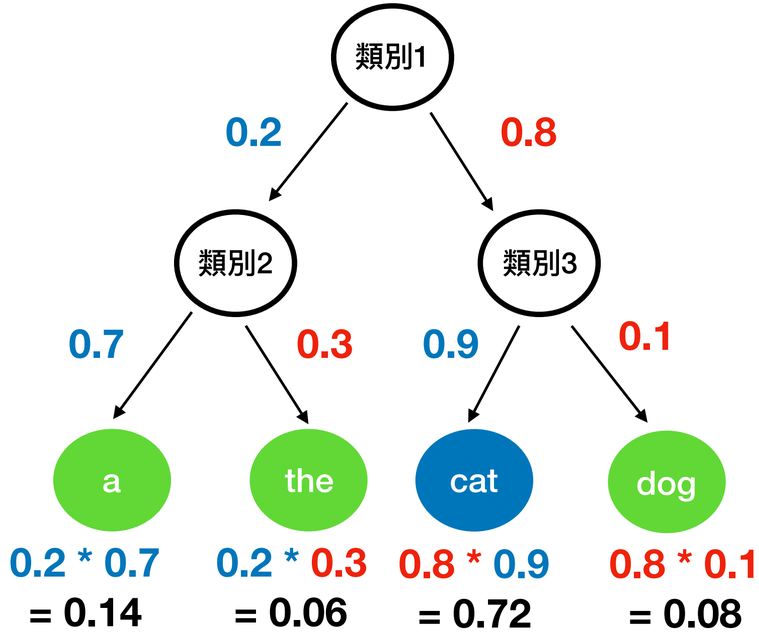
$$P_{cat} = P(\text{類別}1 = 1) \times P(\text{類別}3 = 0)$$

這個做法比較難懂一點，你可以把每一個sigmoid的輸出當作 往右的機率，那 (1 - 輸出) 自然就是往左的機率，所以我們在調整 cat 的時候『類別1』應該往 1 調整，而『類別3』應該往 0 去做調整，之所以還是叫做softmax是因為你會發現每一個葉子加起來的機率還是等於1，等我們只要調整『深度』這麼多個輸出就好

Hierarchical Softmax

我們來看看為何他還是叫做Softmax





機率加起來一樣 = 1