

PCA

機器學習降維

降維有三種



維持原有特徵



過多的特徵在後續的UI/UX都會造成災難，寧可犧牲一些命中率，使用重要的特徵給使用者即可，我們可以在分類器的`feature_importance`裡找到每個特徵的重要性



組合原有特徵



數千個特徵或者數萬個特徵(尤其文字&語言)是無法用傳統方式分析的，於是我們可以結合原有特徵，並且順便完成降維



畫圖：不多說！只能2D/3D

共變異數



共變異數如果你不了解的話可以先當成相關係數來思考

相關係數 $\rho = \frac{x \text{ 和 } y \text{ 的共變異數}}{x \text{ 的標準差} \times y \text{ 的標準差}}$ ↵

共變異數(covariance): $\text{cov}(x, y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu_x)(y_i - \mu_y)$ ↵

變異數(variance): $\text{var}(x) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu_x)^2$ ↵

標準差(standard deviation): $\text{std}(x) = \sqrt{\text{var}(x)}$ ↵

PCA第一步



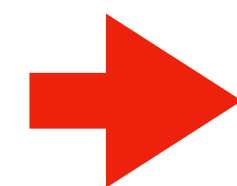
預測資料

$[x_1 \quad x_2 \quad x_3]$



共變異數矩陣

$[\text{Cov}(1, 1)]$	$\text{Cov}(2, 1)$	$\text{Cov}(3, 1)$
$[\text{Cov}(1, 2)]$	$\text{Cov}(2, 2)$	$\text{Cov}(3, 2)$
$[\text{Cov}(1, 3)]$	$\text{Cov}(2, 3)$	$\text{Cov}(3, 3)$



$$x_1 * \text{Cov}(1, 1) + x_2 * \text{Cov}(1, 2) + x_3 * \text{Cov}(1, 3)$$

物理意義：把所有對於第一個特徵的影響全部組合起來，變成新特徵

PCA第二步



上面的第一步有個問題，就是你可能會重複計算影響，因為每個轉換不是垂直的！這時候我們就可以來個**特徵值分解**！對於對稱的矩陣，我們可以把它分解成三個矩陣相乘，而且

$$A = Q \Lambda Q^T \longrightarrow \text{正交(垂直)矩陣}$$

↓
對角線矩陣

$$\begin{bmatrix} \text{常數1} & 0 & 0 \\ 0 & \text{常數2} & 0 \\ 0 & 0 & \text{常數3} \end{bmatrix}$$

PCA第三步



儘管我們可以直接把Q當作我們新的轉換了！但是還可以順便做一件事，只留常數最大的幾個數值就好！
這個留下**最大貢獻度的轉換**就是我們PCA的精髓了

$$\mathbf{A} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^T \longrightarrow \text{正交(垂直)矩陣}$$

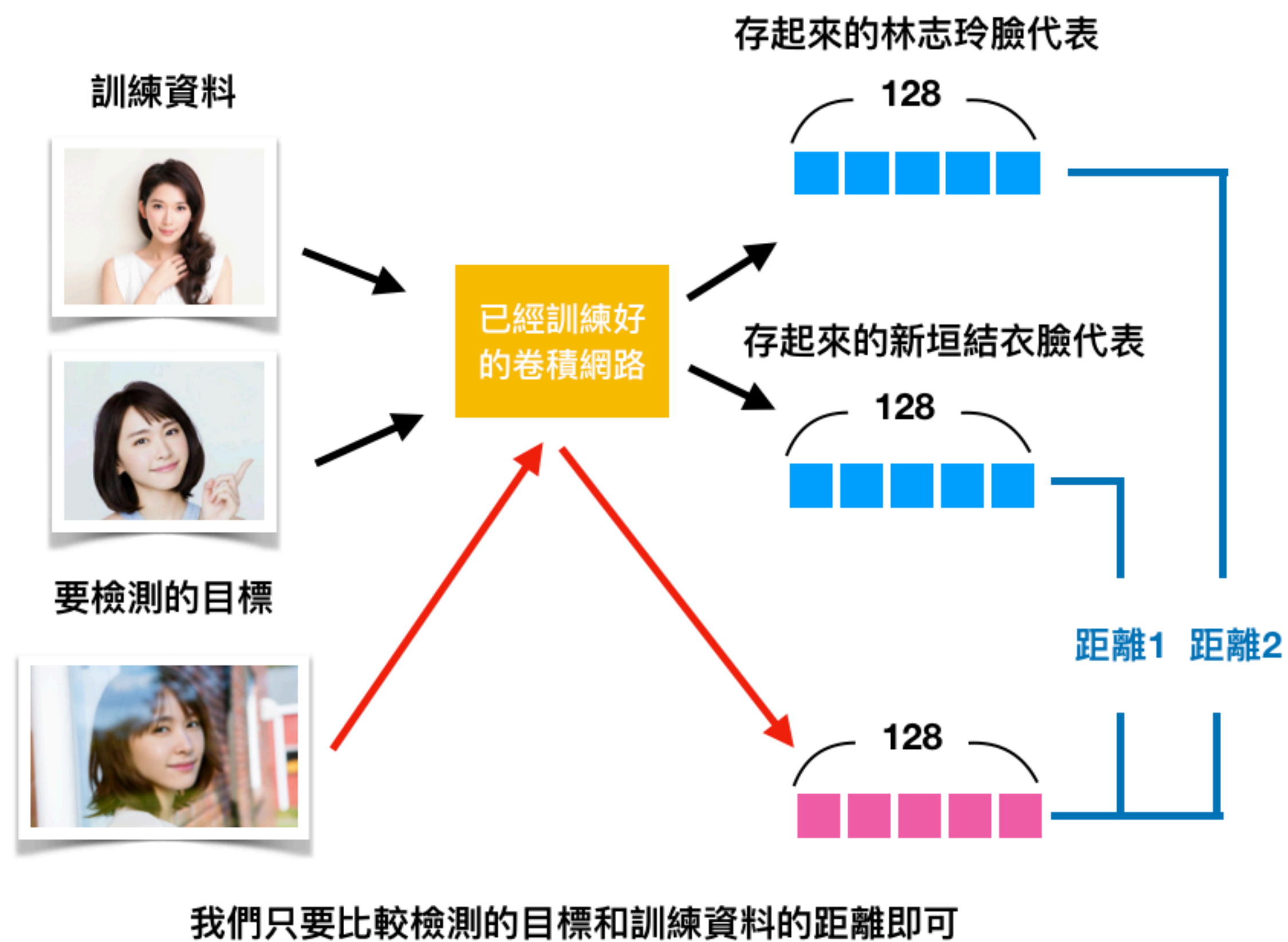
對角線矩陣

$$\begin{bmatrix} \text{常數1} & 0 & 0 \\ 0 & \text{常數2} & 0 \\ 0 & 0 & \text{常數3} \end{bmatrix}$$

人臉辨識

人臉降維

一圖流



FastText

語意分析

NLP



NLP(自然語言處理): Natural Language Processing



不管是什麼任務，中心思想就一個，讓電腦能夠瞭解人類語言



了解人類語言一定要搞定的兩件事！



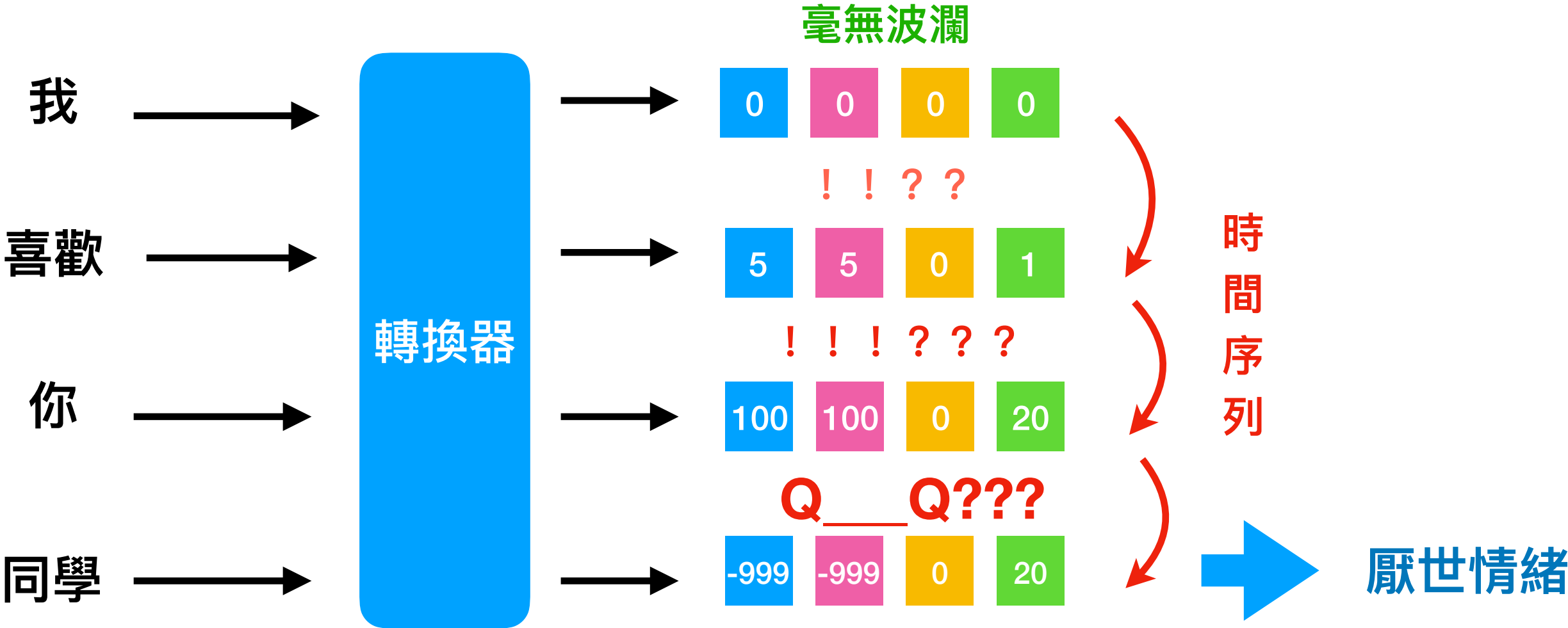
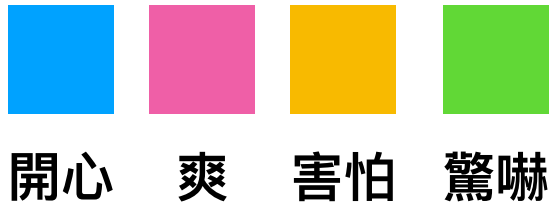
詞義(語意): 人類在聽的時候其實不是聽每個詞，而是你的大腦在聽到某個詞的時候會產生對應的情感感受



上下文(序列): 人類在聽的時候其實還會參考上下文，根據上下文的累積，我們可能會有不一樣的意思
e.g. 討厭(正常), 討厭(裝可愛), 你沒那麼讓人討厭(喜歡)

一圖流

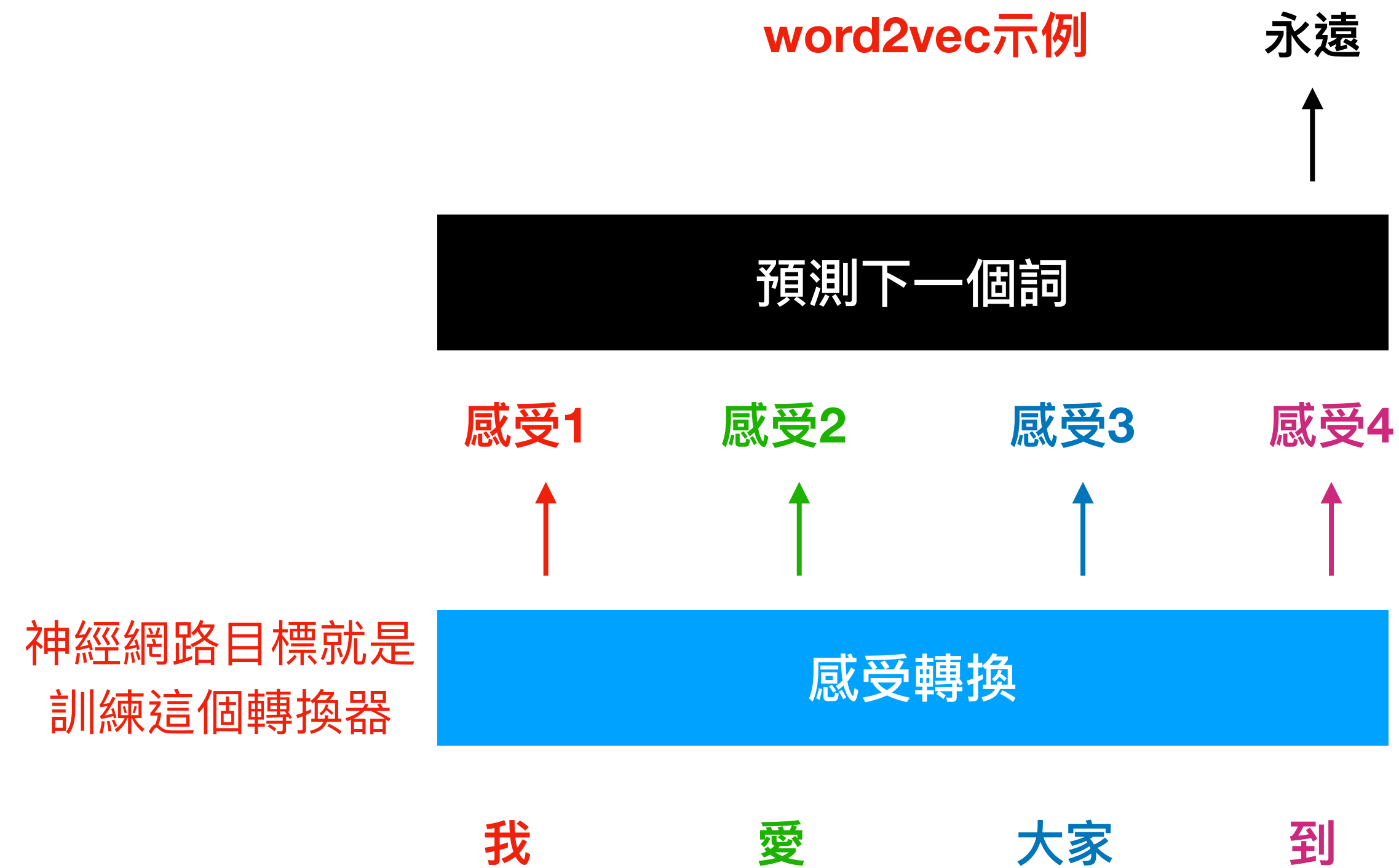
語意度量器



詞向量



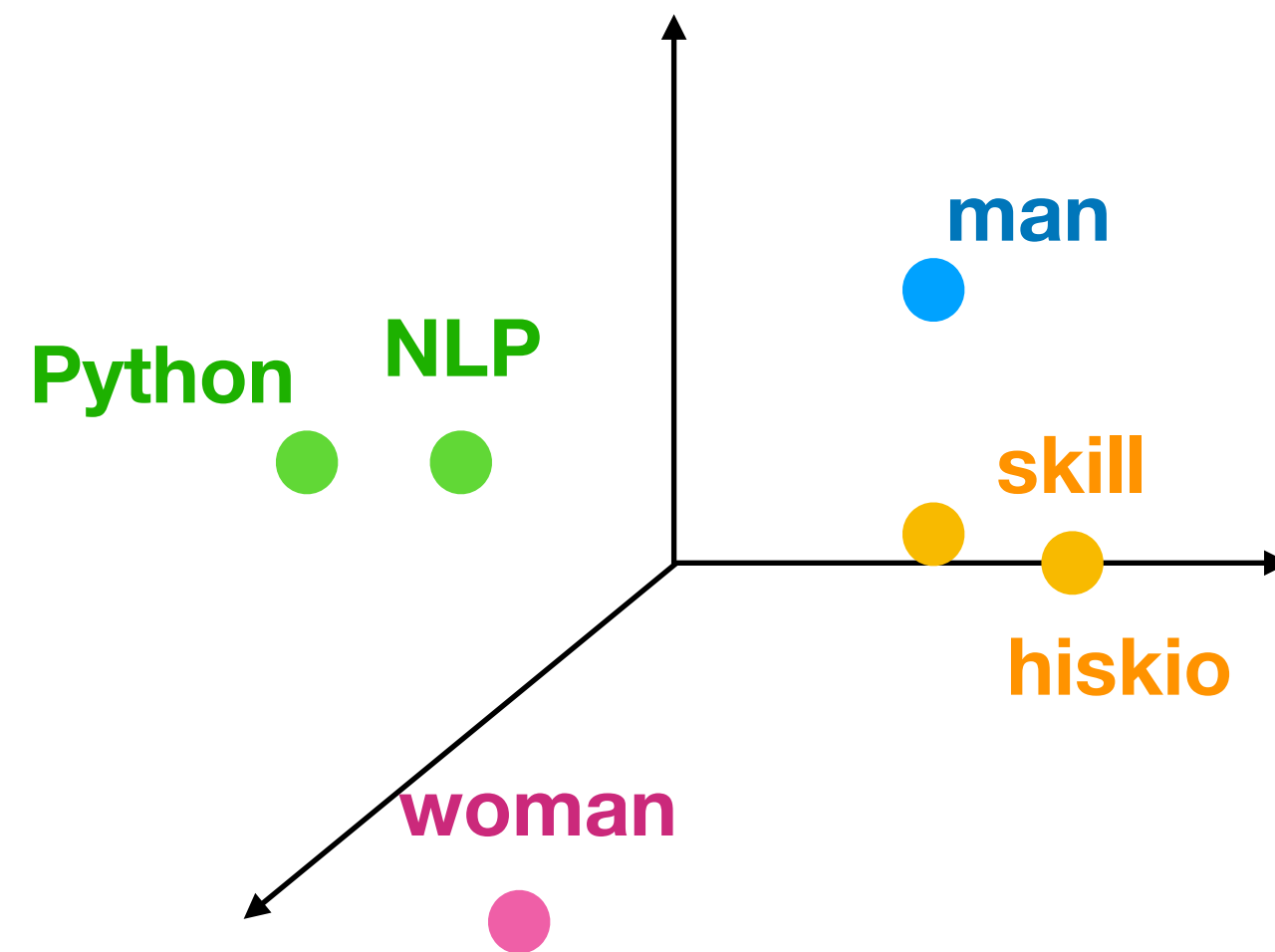
我們在NLP任務裡可能會訓練很多種不同模型，但他們在中間大部分都會先做一件事，產生語意



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)

FastText



今天我們為何介紹FastText呢



NLP一個很大的問題是資料有無辦法撐得起你的模型，何不當個『**不勞而獲的人**』利用大公司訓練好的完整模型，100%比你自已訓練出的語意還準！



FastText主要以標籤來訓練詞向量，並且加入**n-gram**的機制，字詞級別的**n-gram**，尤其字級別的n-gram對我們的英文分析如虎添翼，例如tourism，模型會同時帶入tou, our, ism...等等3-gram字，那我們就會發現他跟其他以**ism**結尾的字有點類似

[Docs](#) [Resources](#) [Blog](#) [GitHub](#)

Word vectors for 157 languages

We distribute pre-trained word vectors for 157 languages, trained on [Common Crawl](#) and [Wikipedia](#) using fastText. These models were trained using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives. We also distribute three new word analogy datasets, for French, Hindi and Polish.

號稱已經訓練好157種語言的FastText

Demo Time

Colab網址: <https://reurl.cc/xDDYG1>

#@title 比較兩個特定詞的相似度

```
text1 = 'Coldplay' #@param {type:"string"}
text2 = 'OneRepublic' #@param {type:"string"}
model.wv.similarity(text1, text2)
```

0.6214945

↑ ↓ ↺ 💬 ⚙️ 🗑️ ⋮

比較兩個特定詞的相似度

text1: " Coldplay "

text2: " OneRepublic "

進階回歸

回歸比較

回歸有三種



RandomForestRegressor



最無腦的選擇，不用多做任何特徵處理，但誤差相對大，因為你回答的是整群的平均



線性回歸(Lasso, Ridge)



效果好，但一定要將特徵處理成**常態分布**！



深度學習



將最後一層的Activation調成None，Loss調整成MSE，效果好，也不用做太多特徵處理，但需要你的訓練資料較多