

Word2Vec

2017.2.18

陳冠穎

真的太**高深**惹...

歡迎打臉

Vector Space of Semantics

	Vector Space Model	Latent Semantic Analysis
Definitions	<ul style="list-style-type: none"> • VSM represents each document as a vector of terms. • As a result, each document has been projected to a vector space. 	<ul style="list-style-type: none"> • LSA is based on singular value decomposition (SVD), a mathematical matrix decomposition technique that is applicable to text corpora. • Because of SVD, the new dimensions are assumed to be orthogonal. • LSA solve problems about sparseness and synonym via <u>dimension reduction</u>.
Similarity Measure	<ul style="list-style-type: none"> • Cosine similarity 	<ul style="list-style-type: none"> • Cosine similarity
Limitations	<ul style="list-style-type: none"> • Sparseness: VSM generate a very high dimensional semantic space, but many terms in a single document are inexistent. • Synonym: Different terms may represent the same meaning but VSM only takes account of their frequency. 	<ul style="list-style-type: none"> • Problem of polysemy (one term with multiple meanings) is not addressed. • No probabilistic model of term occurrences.

	PLSA	LDA
Definitions	<ul style="list-style-type: none"> • PLSA is a way to locate document to latent semantic space, and it is a probabilistic model evolved from LSA. • PLSA is a topic model for general co-occurrence data associated each observation with a latent variable, stands for topic. 	<ul style="list-style-type: none"> • LDA extends PLSA and overcomes problems of PLSA by further considering dirichlet priors on the parameters. • LDA is a generative probabilistic topic model of a collection of documents. • Documents are represented as mixtures of latent topics, where each topic is characterized by a distribution over words.
Similarity Measure	<ul style="list-style-type: none"> • Cosine similarity • KL divergence 	<ul style="list-style-type: none"> • Cosine similarity • KL divergence
Limitations	<ul style="list-style-type: none"> • The number of parameters grows linearly with the number of documents, so that PLSA tends to over-fit the training data. • Although PLSA is a generative model of the training documents on which it is estimated, it is not a generative model of new documents. 	<ul style="list-style-type: none"> • Similar to PLSA, but with dirichlet priors for the document-topic and topic-word distributions. This prevents over-fitting. • LDA is time-consuming.

我們很習慣...

一篇文章 = 一個向量

一個詞彙 = 文章向量中的一個維度

利用 **cosine similarity** 計算兩篇文章的語意相似度

term-document 矩陣

TF, TF-IDF...

The dog run.
The cat run.
The dog sleep.
The cat sleep.
The dog bark.
The cat meows.
The bird fly.
The bird sleep.

文章 \ 字彙	The	dog	cat	bird	run	sleep	bark	meows	fly
Doc 1	1	1			1				
Doc 2	1		1		1				
Doc 3	1	1				1			
Doc 4	1		1			1			
Doc 5	1	1					1		
Doc 6	1		1					1	
Doc 7	1			1					1
Doc 8	1			1		1			

term-document 矩陣

TF, TF-IDF...

The dog run.

The cat run.

The dog sleep.

The cat sleep.

The dog bark.

The cat meows.

The bird fly.

The bird sleep.

文章 \ 字彙	The	dog	cat	bird	run	sleep	bark	meows	fly
Doc 1	1	1			1				
Doc 2	1		1		1				
Doc 3	1	1				1			
Doc 4	1		1			1			
Doc 5	1	1					1		
Doc 6	1		1					1	
Doc 7	1			1					1
Doc 8	1			1		1			

向量維度過大，造成資料過度稀疏、佔用記憶體空間

那麼...

將詞彙 向量化 呢？

如果可以的話...

一個詞彙 = 一個向量

利用 **cosine similarity** 計算兩個詞彙的關聯性

如何將詞彙
向量化呢？

.....

Context

1. 日本 青森 的 蘋果 又大又甜
2. **ipad, iphone** 是 蘋果 公司的 電子產品

相同 word + 不同的前後文 = 不同的意涵

Context

如果用「蘋果」搜尋這三份文件 ...

1. 蘋果 **iphone7** 推出 新色 曜石黑
2. **JetBlack iphone7** 是 蘋果 公司 的 新品
3. 曜石黑 **JetBlack** 出現 落漆 災情

Context

只能搜尋到 1, 2 號文件...

可是 3 號文件其實也是在講蘋果公司...

1. 蘋果 iphone7 推出 新色 曜石黑
2. JetBlack iphone7 是 蘋果 公司 的 新品
3. 曜石黑 JetBlack 出現 落漆 災情

Context

善用 context 能改善搜尋結果

1. 蘋果 **iphone7** 推出 新色 曜石黑
2. **JetBlack** **iphone7** 是 蘋果 公司 的 新品
3. 曜石黑 **JetBlack** 出現 落漆 災情

感覺到了嗎...？

詞彙的 **向量化**

和 **前後文** 息息相關

word-context 矩陣

context: n=1

The dog run.
The cat run.
The dog sleep.
The cat sleep.
The dog bark.
The cat meows.
The bird fly.
The bird sleep.

	The	dog	cat	bird	run	sleep	bark	meows	fly
The	0	3	3	2	0	0	0	0	0
dog	3	0	0	0	1	1	1	0	0
cat	3	0	0	0	1	1	0	1	0
bird	2	0	0	0	0	1	0	0	1
run	0	1	1	0	0	0	0	0	0
sleep	0	1	1	1	0	0	0	0	0
bark	0	1	0	0	0	0	0	0	0
meows	0	0	1	0	0	0	0	0	0
fly	0	0	0	1	0	0	0	0	0

co-occurrence

word-context 矩陣 co-occurrence

context: n=1

The dog run.
The cat run.
The dog sleep.
The cat sleep.
The dog bark.
The cat meows.
The bird fly.
The bird sleep.

	The	dog	cat	bird	run	sleep	bark	meows	fly
The	0	3	3	2	0	0	0	0	0
dog	3	0	0	0	1	1	1	0	0
cat	3	0	0	0	1	1	0	1	0
bird	2	0	0	0	0	1	0	0	1
run	0	1	1	0	0	0	0	0	0
sleep	0	0	0	0	0	0	0	0	0
bark	0	0	0	0	0	0	0	0	0
meows	0	0	0	0	0	0	0	0	0
fly	0	0	0	0	0	0	0	0	0

由於 **dog** 和 **cat** 這兩個字出現在類似的上下文情境中，因此可以判斷出 **dog** 和 **cat** 語意相近。

➡ cosine similarity

The dog run.
The cat run.
The dog sleep.
The cat sleep.
The dog bark.
The cat meows.
The bird fly.
The bird sleep.

	The	dog	cat	bird	run	sleep	bark	meows	fly
The	0	3	3	2	0	0	0	0	0
dog	3	0	0	0	1	1	1	0	0
cat	3	0	0	0	1	1	0	1	0
bird	2	0	0	0	0	1	0	0	1
run	0	1	1	0	0	0	0	0	0
sleep	0	1	1	1	0	0	0	0	0
bark	0	1	0	0	0	0	0	0	0
meows	0	0	1	0	0	0	0	0	0
fly	0	0	0	1	0	0	0	0	0

向量維度等於總詞彙量，造成資料過度稀疏

Word2Vec 原理

將詞彙做 **one-hot encoding**，再透過 **word2vec** 類神經網路計算，求出降維後的詞向量（語意向量）

太複雜的

我不懂

太簡單的

不用教

Word Embedding

詞彙的詞向量、語意向量

Corpus

1. 日本 青森 的 蘋果 又大 又 甜
2. 蘋果 **iphone7** 推出 新色 曜石黑
3. 曜石黑 **JetBlack** 出現 落漆 災情

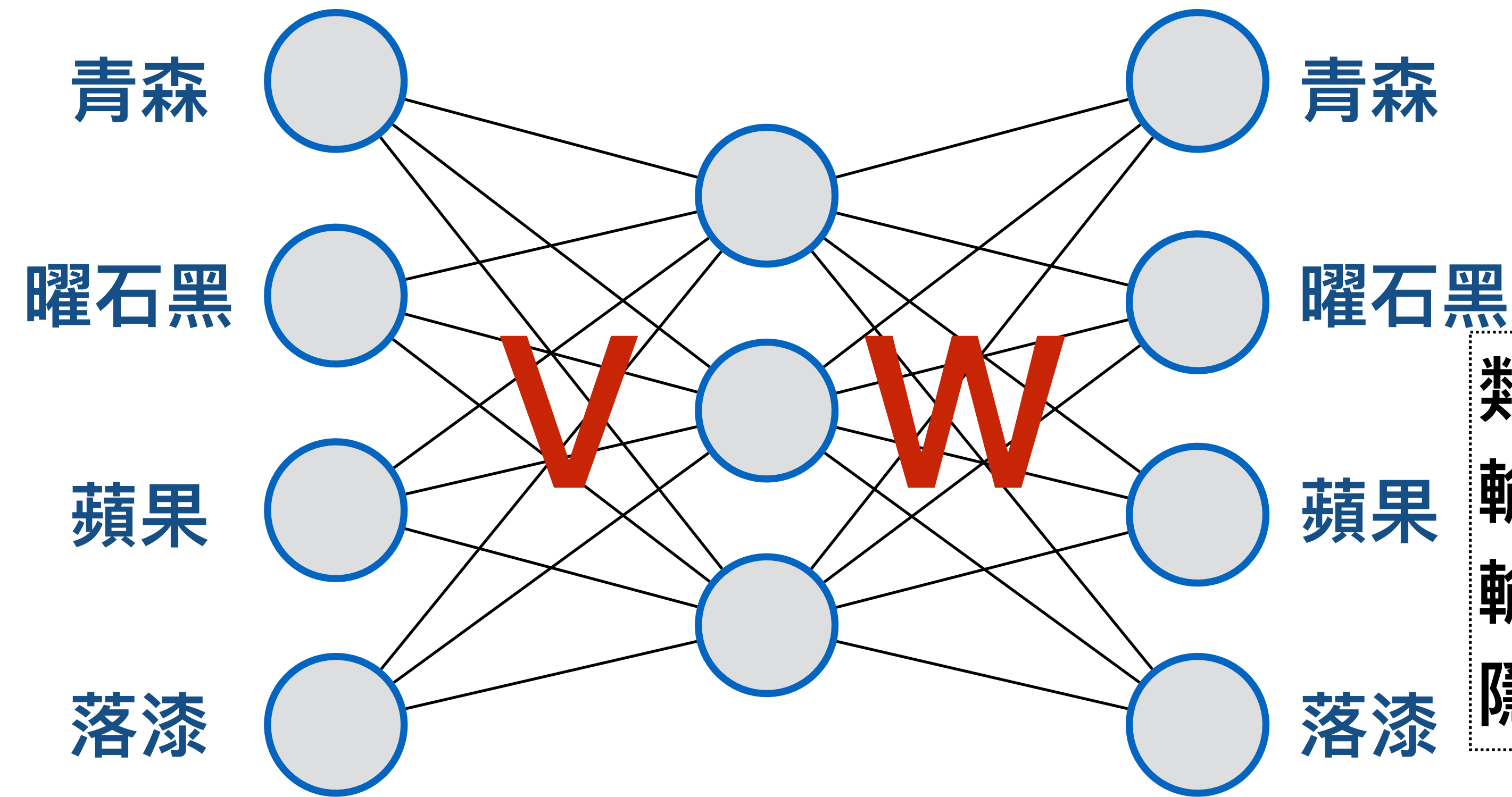
One-Hot Encoding

青森	曜石黑	蘋果	落漆
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

每個詞彙都有一個代表它的維度，任意兩個 **one-hot encoding** 的詞彙向量內積結果皆為 **0**，表示兩向量是垂直的（即：假設每個詞彙的語意是不相干的）。

假設總詞彙量只有 4 個

Word2Vec 類神經網路架構



類神經網路共三層：

輸入層，神經元數 = 總詞彙量

輸出層，神經元數 = 總詞彙量

隱藏層，神經元數 = 降維後的維度個數

隱藏層，沒有 **activation function**

輸出層的 **activation function** 為 **sigmoid**

Word2Vec 類神經網路架構

權重：

$$V_{4 \times 3}$$

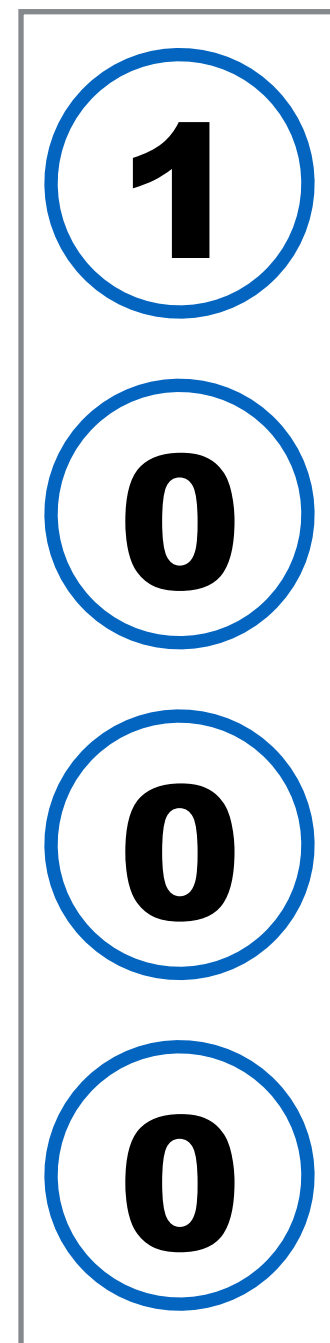
$$W_{3 \times 4}$$

$$V = \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \\ v_{41} & v_{42} & v_{43} \end{bmatrix}$$

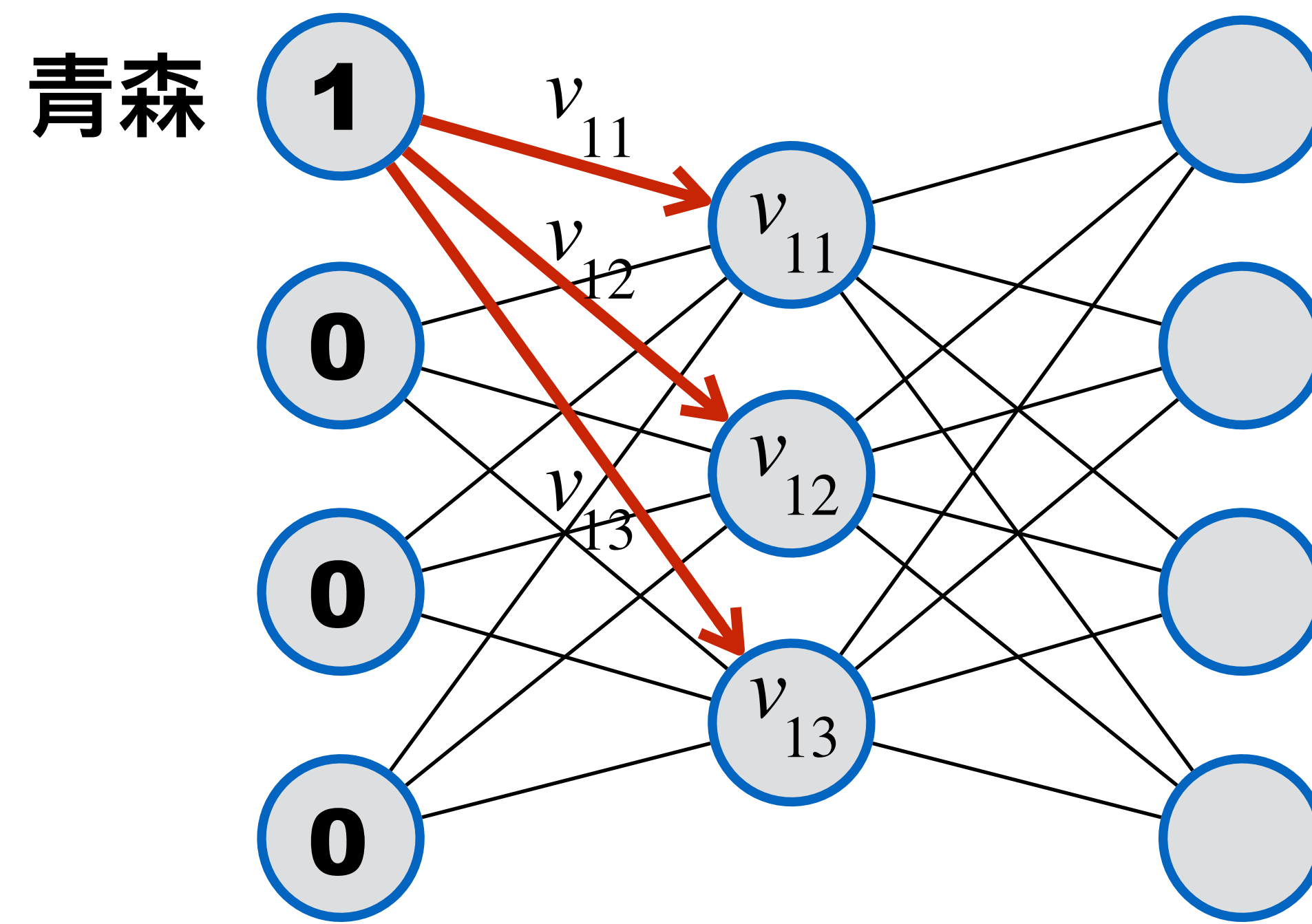
$$W^T = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}$$

Word2Vec 類神經網路架構

青森

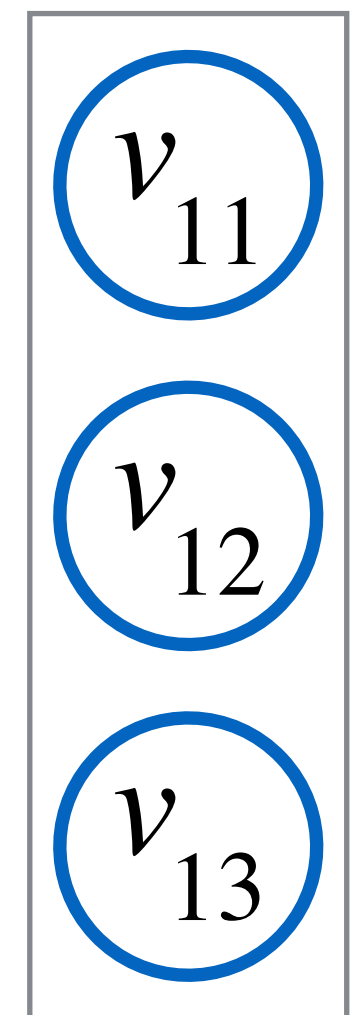


高維度



青森

降維後的語意向量



低維度

Word2Vec 類神經網路架構

因 **input** 是 **one-hot encoding** 的向量，且隱藏層沒有 **activation function**，
➡ 故輸入資料進入類神經網路後，在隱藏層所取得的值，即 **V** 矩陣中某橫排的值。

例如，輸入的是「青森」的 **one-hot encoding**，與權重矩陣相乘後，在隱藏層取得的值是權重矩陣中的第一個橫排：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \\ v_{41} & v_{42} & v_{43} \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & v_{13} \end{bmatrix}$$

Word2Vec 類神經網路架構

因此，

V 中的某個橫排，就是某個詞彙（降維後）的語意詞向量。

從反方向來看，

由於輸出層也是對應到詞彙的 **one-hot encoding**，

因此，**W** 轉置矩陣中的某個橫排，也是某個詞彙的語意向量。

Word2Vec 類神經網路架構

所以...

一個詞彙分別在 V 和 W 中各有一個語意詞向量。

但通常會選擇 權重 V 中的語意向量，

作為 **Word2Vec** 的輸出結果。

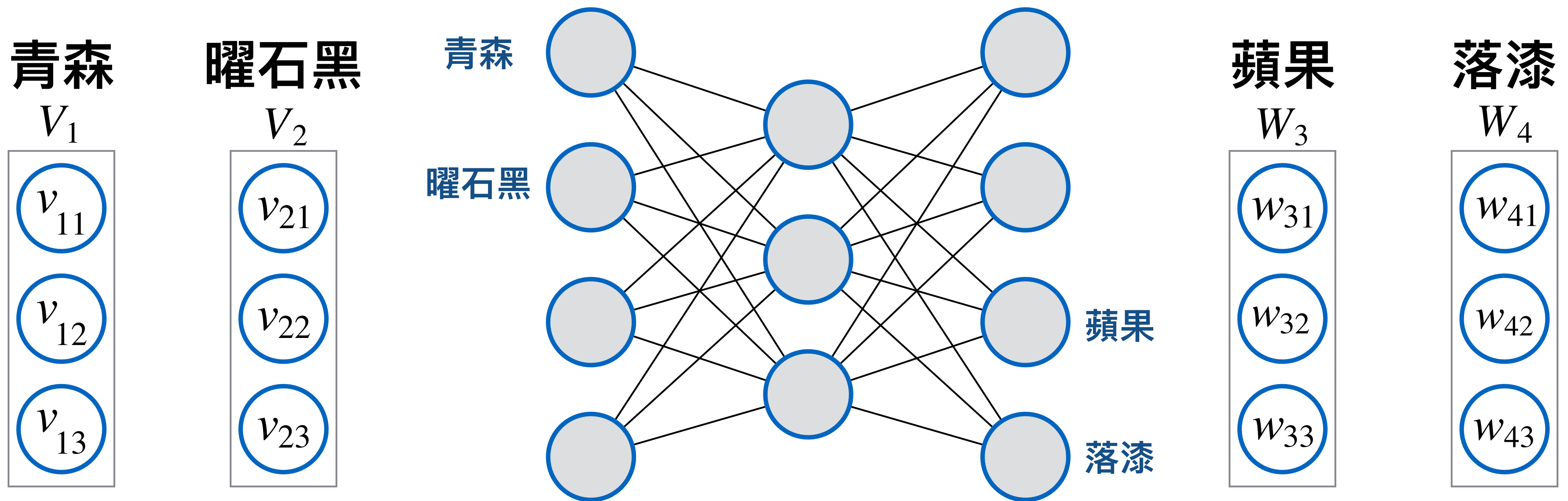
權重矩陣 V 的維度：總詞彙數目 \times 詞向量降維後的維度個數

初始化 Word2Vec

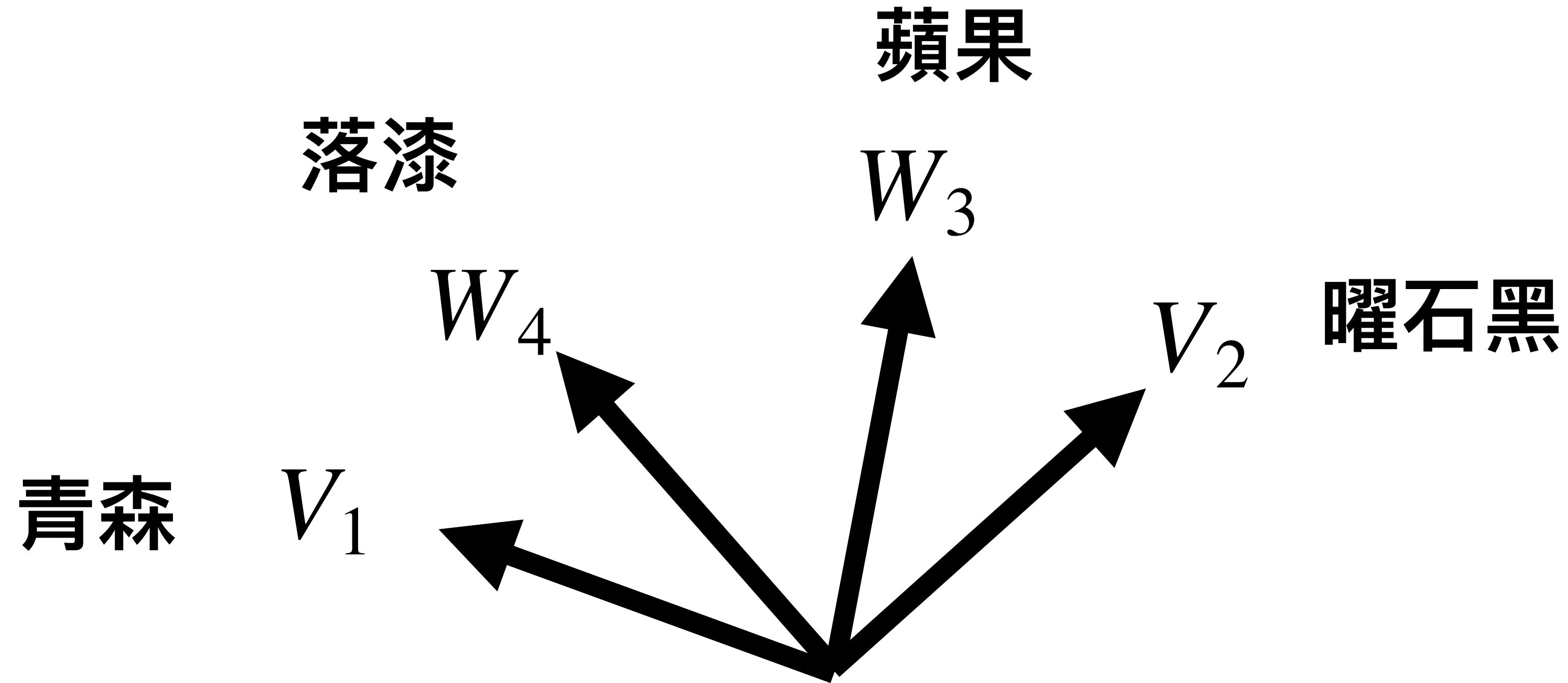
類神經網路

隨機初始化每個詞彙的詞向量

(隨機給每個 **weight** 不同的數值)



這些向量的方向都是隨機的，跟語意無關



訓練 Word2Vec

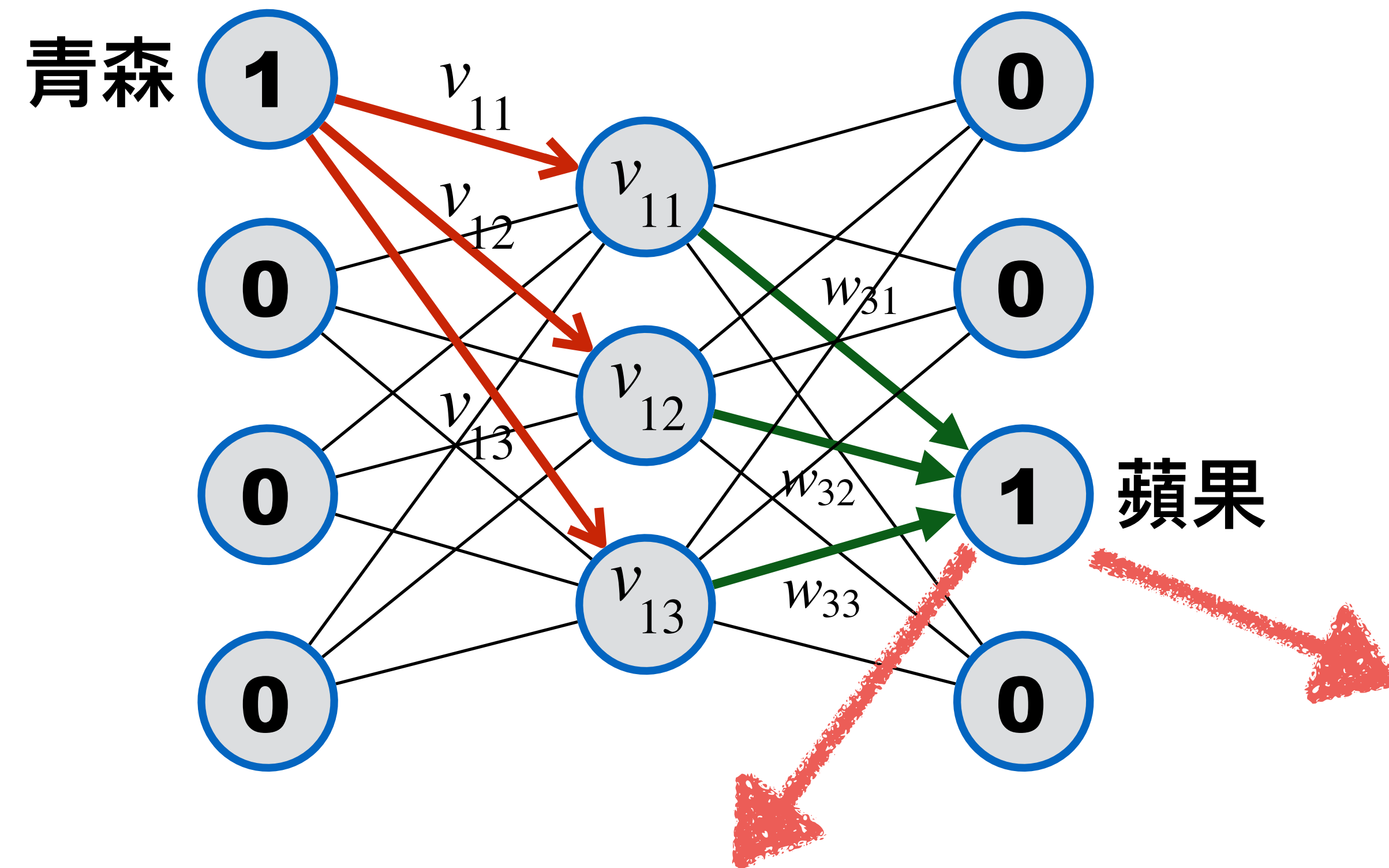
類神經網路

訓練目的：希望讓語意向量真的和語意有關

利用某詞彙 上下文 的字

來做訓練，讓語意向量能抓到文字的語意

若 青森 的上下文有 蘋果，
令 青森 為 **Word2Vec** 的 **input**，蘋果 為 **output**

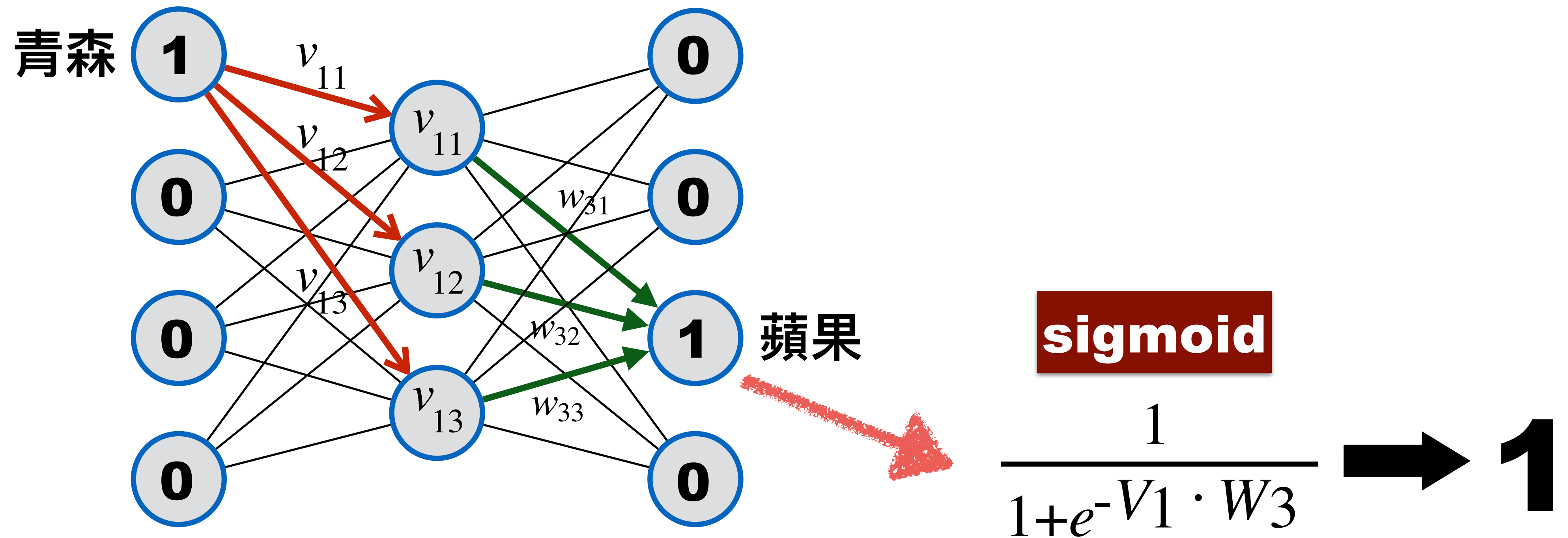


sigmoid

$$\frac{1}{1+e^{-V_1 \cdot W_3}}$$

內積 $V_1 \cdot W_3 = v_{11}w_{31} + v_{12}w_{32} + v_{13}w_{33}$

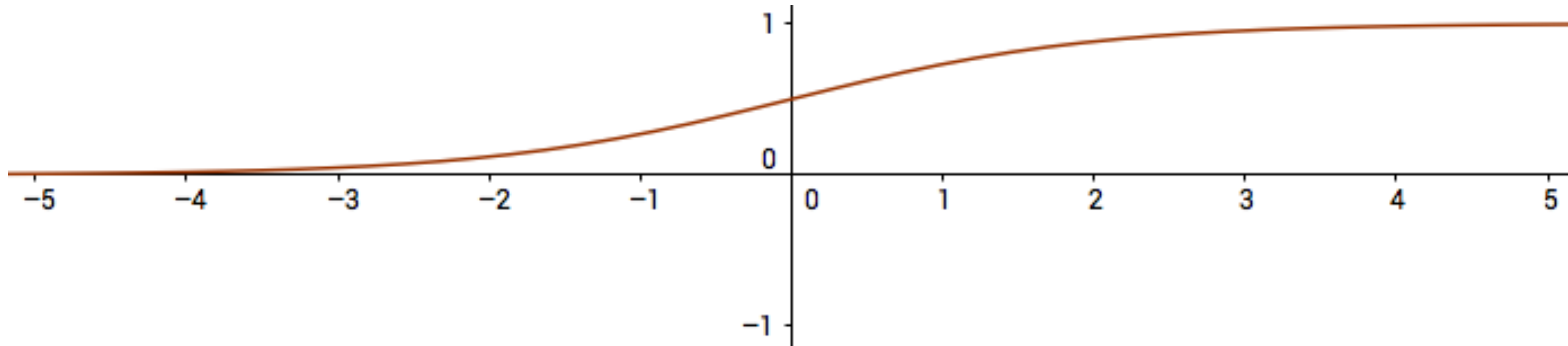
由於 蘋果 出現在 青森 的上下文中，所以要
訓練類神經網路，在 蘋果 位置可以輸出 1



sigmoid

最常見的非線性作用函數，連續函數（可微分），值域介於 **0~1** 之間

$$f(x) = \frac{1}{1 + e^{-x}}$$



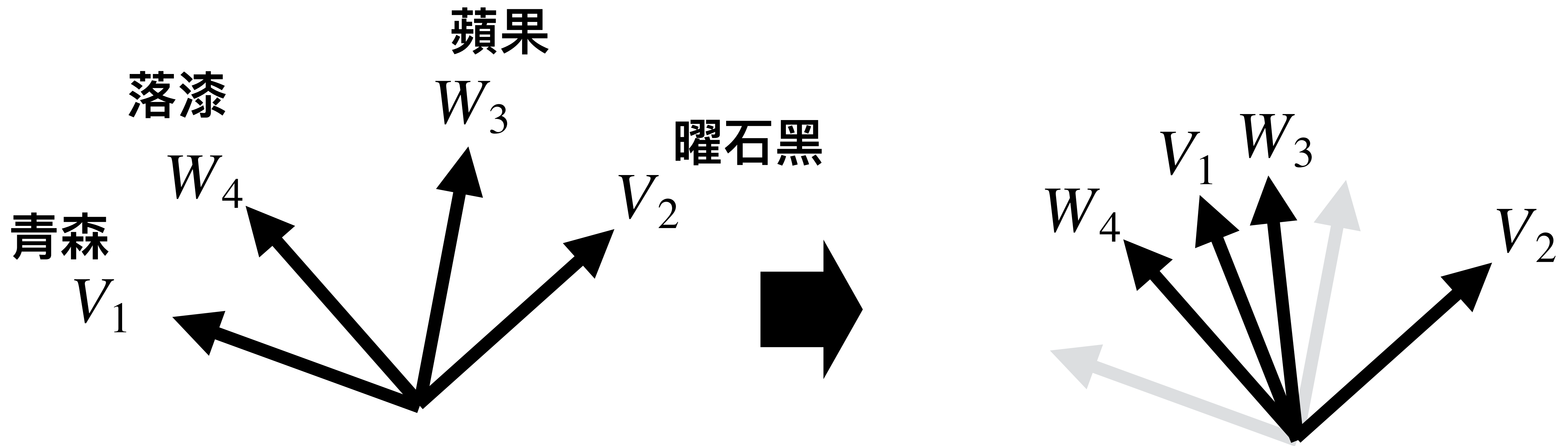
如果要讓 蘋果 的位置輸出為 1 ，

則 **V1 和 W3 的內積要越大越好。**

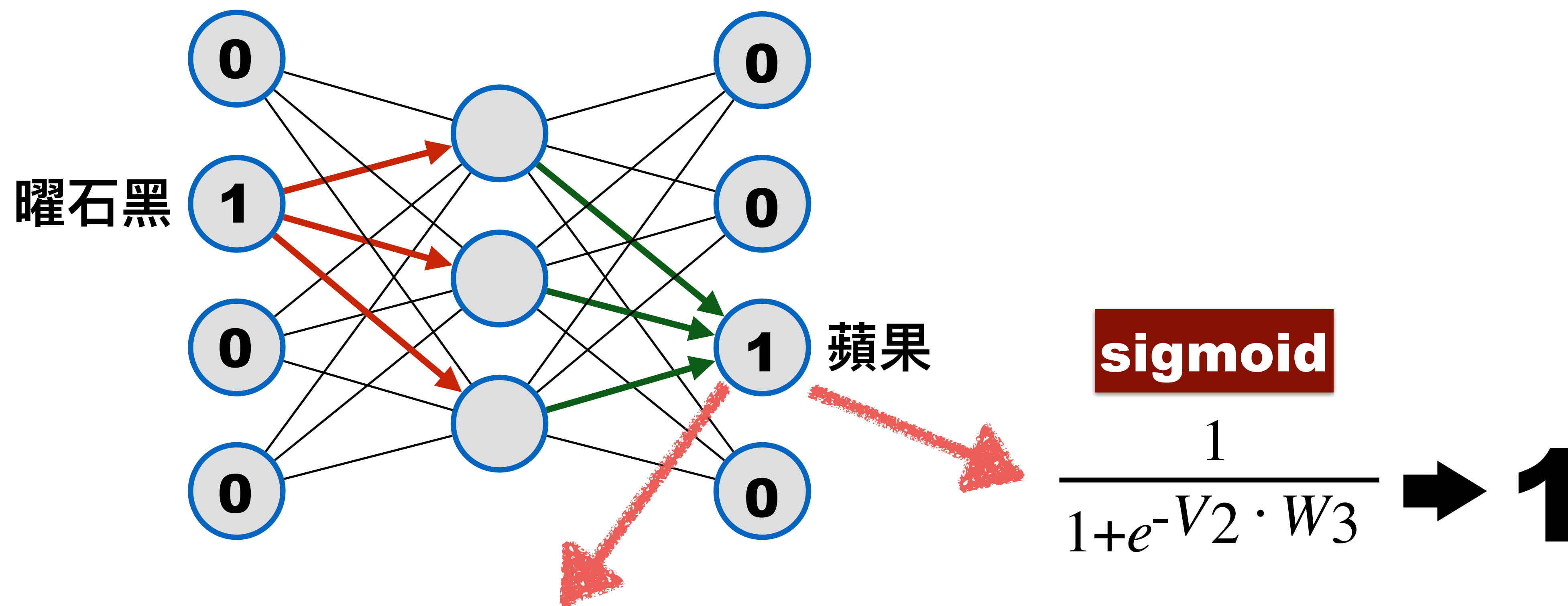
內積要大，就是**向量角度要越小**，

訓練過程中，會修正這兩個向量的角度。

修正完後， V_1 和 W_3 的角度變小

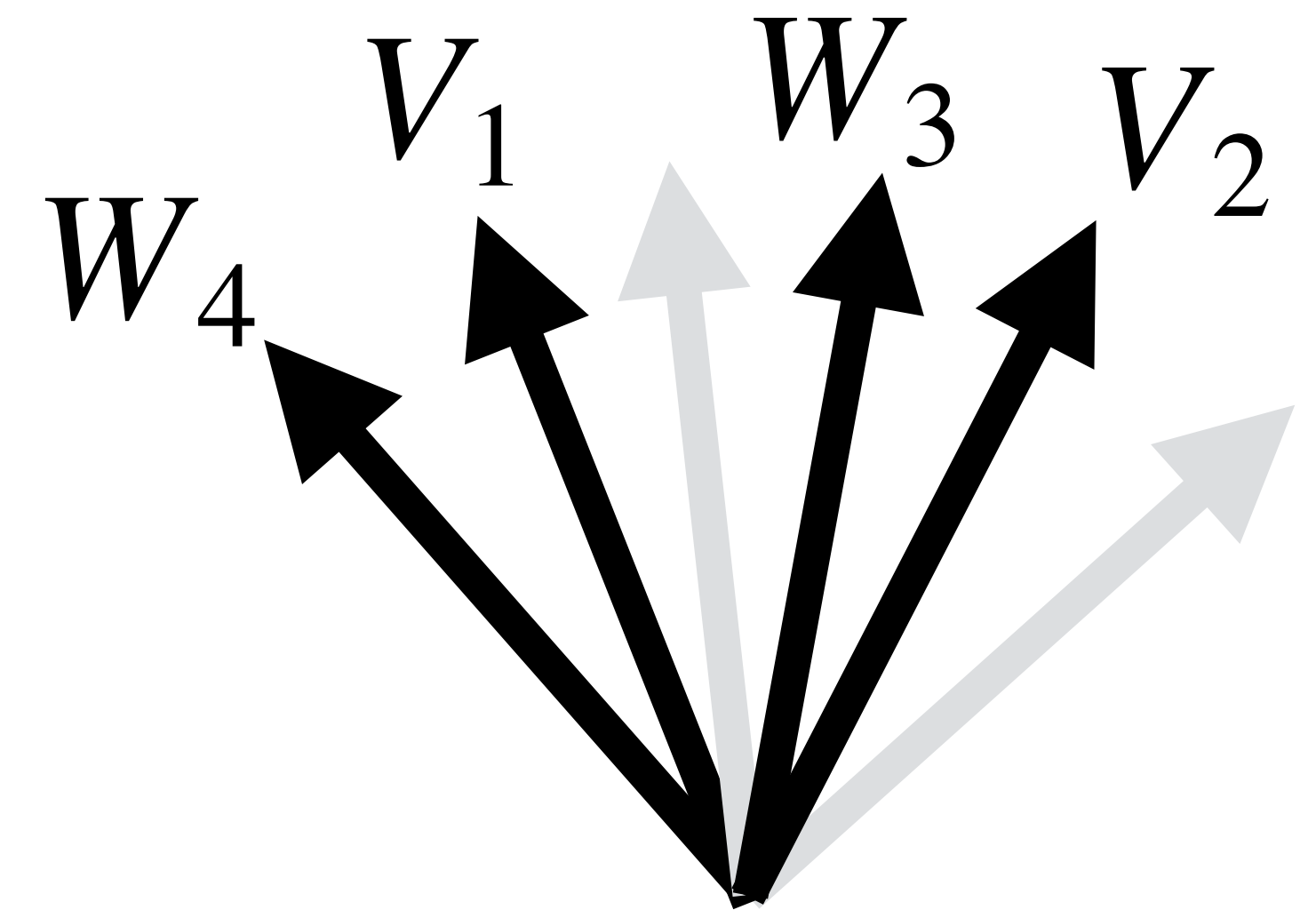
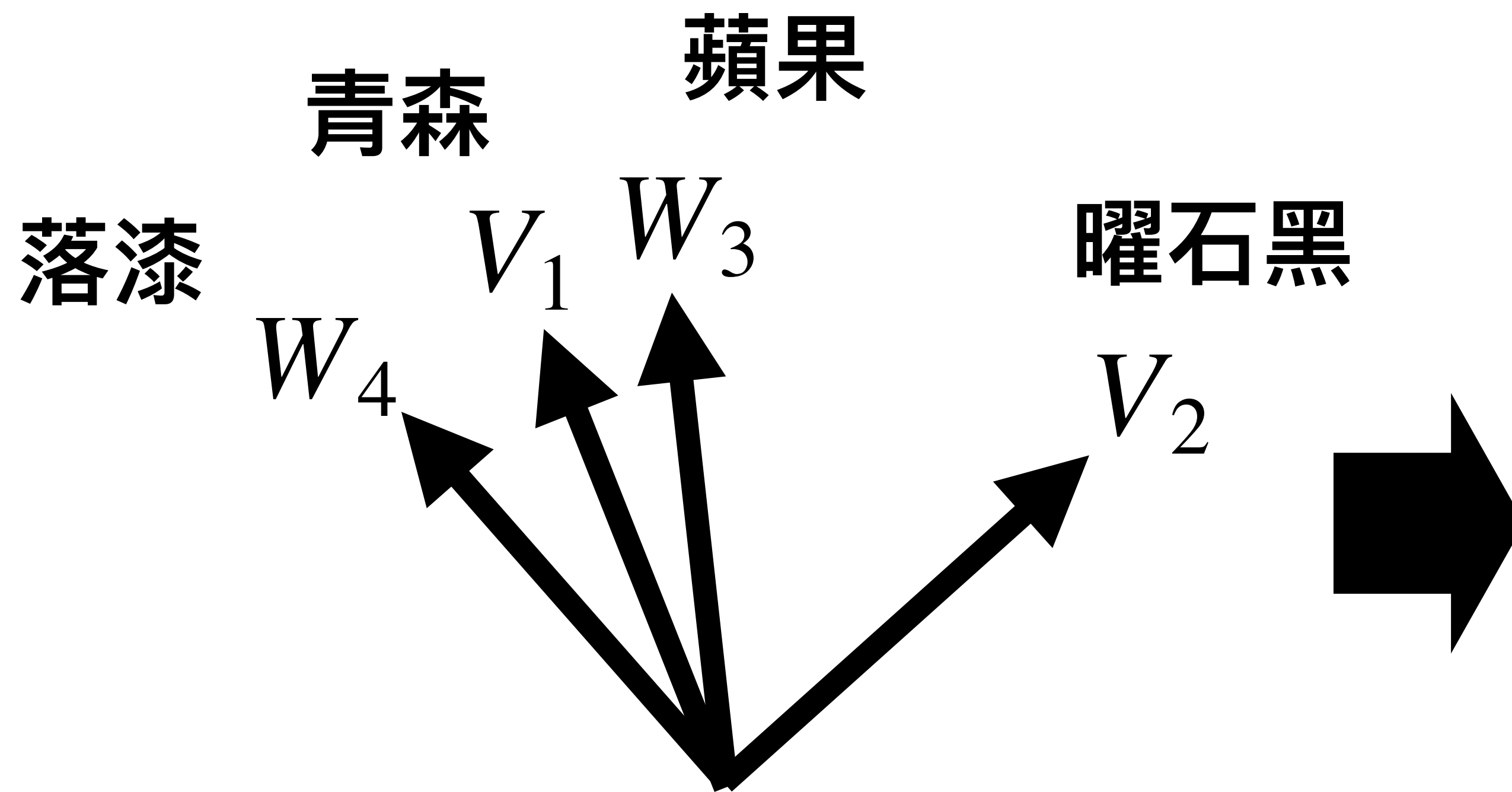


同理，若 曜石黑 的上下文有 蘋果，
令 曜石黑 為 **Word2Vec** 的 **input**，蘋果 為 **output**



內積 $V_2 \cdot W_3 = v_{21}w_{31} + v_{22}w_{32} + v_{23}w_{33}$

修正完後， V_2 和 W_3 的角度變小



不過...

事情沒有笨蛋想的那麼簡單...

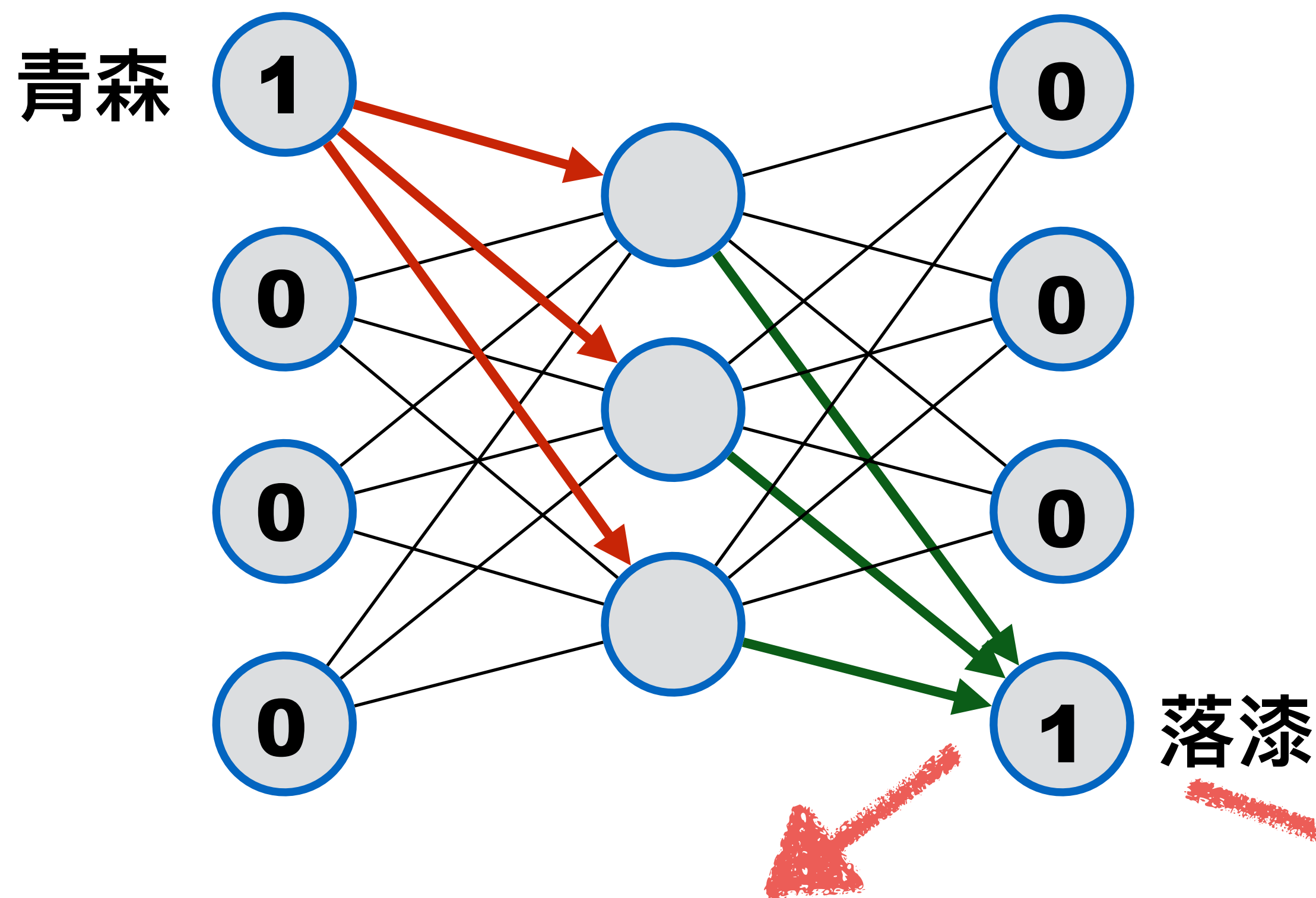
以上的訓練方法有個小問題...

訓練完後，所有的向量都會位於
同一條直線上，而無法

分辨出每個詞彙的語意差異

若要让 **Word2Vec** 學會分辨語意的差異，就需要加入**反例**，也就是沒有出現在上下文的詞彙

若 青森 的上下文不會出現 落漆



sigmoid

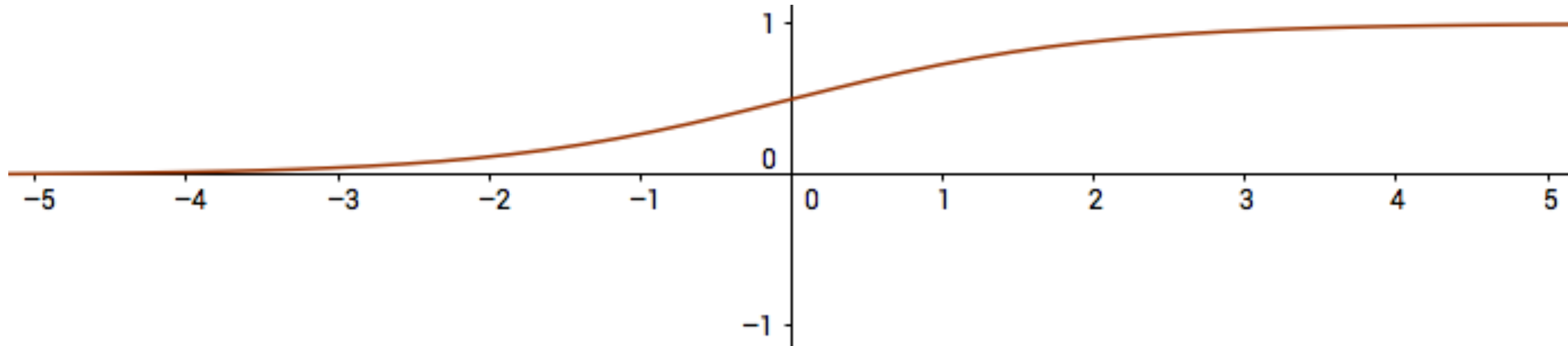
$$\frac{1}{1+e^{-V_1 \cdot W_4}} \Rightarrow 0$$

內積 $V_1 \cdot W_4 = v_{11}w_{41} + v_{12}w_{42} + v_{13}w_{43}$

sigmoid

最常見的非線性作用函數，連續函數（可微分），值域介於 **0~1** 之間

$$f(x) = \frac{1}{1 + e^{-x}}$$



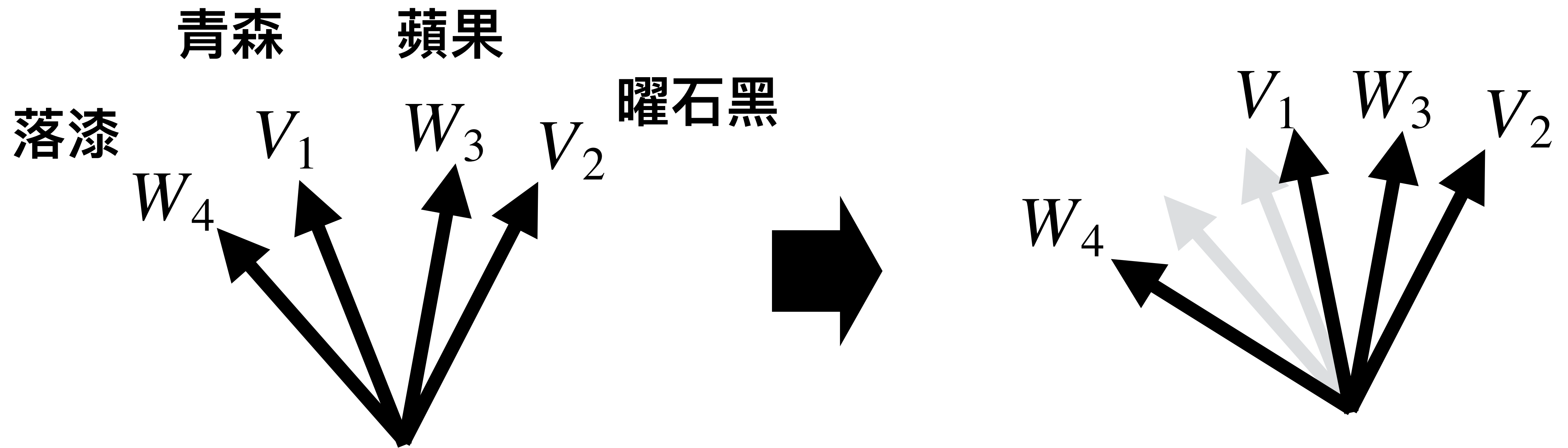
如果要讓 落漆 的位置輸出為 0 ，

則 **V1 和 W4 的內積要越小越好。**

內積要小，就是 **向量角度要越大，**

訓練過程中，會修正這兩個向量的角度。

修正完後， V_1 和 W_4 的角度變大



訓練完成後，得出的語意向量：

語意相近的，夾角越小；

語意相差越遠的，夾角越大！

訓練 Word2Vec

類神經網路

目標函數 **cost function: cross-entropy**

由於 曜石黑 的上下文有 蘋果，所以 蘋果 為
positive example；

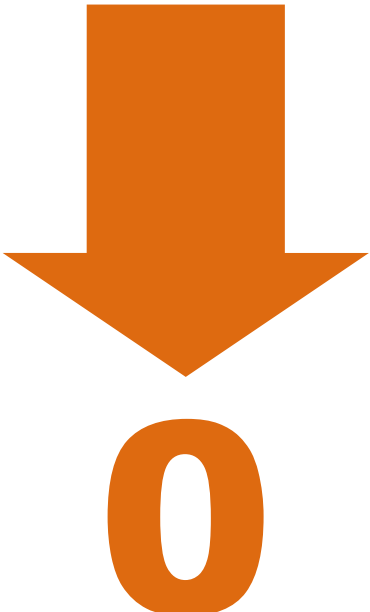
而 曜石黑 的上下文沒有 青森，所以 青森 為
negative example

希望 **positive example** 輸出為 1，
negative example 輸出為 0

➡ **最小化** 目標函數 J

Min. cost function: cross-entropy

$$J = -\log\left(\frac{1}{1+e^{-V_I^T \cdot W_{pos}}}\right) - \sum_{neg} \log\left(1 - \frac{1}{1+e^{-V_I^T \cdot W_{neg}}}\right)$$

 0

希望輸出 1

希望輸出 0

其中 V_I 為輸入端的詞向量，而 W_{pos} 和 W_{neg} 為輸出端的詞向量。

通常，對於每筆 V_I 而言，會找一個 **positive example** 和多個 **negative example**，因此用 \sum 將這些 **negative example** 算出來的結果加起來。

gradient descent

利用 **Backward Propagation**

調整**權重矩陣 V 和 W** 的值



最小化目標函數

對不起

我偷懶...

實際上...

**Word2Vec 的輸入層和輸出層
各有兩種架構**

輸入層有 **CBOW** 和 **skip-gram** 兩種，

輸出層有 **hierarchical softmax** 和
negative sampling 兩種

以上只有說明 **skip-gram** 搭配
negative sampling 的架構

下回待續...?

Hands On

gensim 套件

```
class gensim.models.word2vec.Word2Vec(sentences=None,  
size=100, alpha=0.025, window=5, min_count=5,  
max_vocab_size=None, sample=0.001, seed=1, workers=3,  
min_alpha=0.0001, sg=0, hs=0, negative=5, cbow_mean=1,  
hashfxn=<built-in function hash>, iter=5, null_word=0,  
trim_rule=None, sorted_vocab=1, batch_words=10000)
```

**The training algorithm:
sg=0 , CBOW
sg=1 , skip-gram**

**hs=0 , negative sampling
hs=1 , hierarchical softmax**

if negative > 0, negative sampling will be used

**if cbow_mean = 0, use the sum
of the context word vectors.
If 1 (default), use the mean.
Only applies when cbow is used.**

Window Size

window is the maximum distance between the current and predicted word within a sentence.

1. 日本 青森 蘋果 產量 占 全國 產量 一半 以上
2. 川普 當選 美國 總統 蘋果 執行長 庫克 致電 祝賀
3. 美國 製造業 回流 搞得 產業界 風聲鶴唳 原以為 蘋果 供應鏈
會是 風暴中心 發現 相關 企業 集體 冷處理
4. 蘋果 希望 印度 聯邦政府 能給予 10 年 進口關稅 優惠
5. 蘋果 將在 印度 生產 製造 iPhone 開設 蘋果 零售店

該來的還是躲不掉...

CBOW

依序輸入 **context word** ➡ 訓練 **target word**

skip-gram

同時輸入 **context word** ➡ 訓練 **target word**

Initialize the model from an iterable of sentences.

Each sentence is a list of
words (unicode strings) that
will be used for training.

Gensim 的 Word2Vec 的輸入資料是句子的序列，每個句子由單詞的列表所組成

```
class gensim.models.word2vec.LineSentence(source,  
max_sentence_length=10000, limit=None)
```

**Simple format: one sentence = one line;
words already preprocessed and separated by whitespace.**

**Source can be either a string or a file object.
Clip the file to the first limit lines
(or no clipped if limit is None, the default).**

save(*args, **kwargs)

Save the object to file.

gensim 中預設的 Word2Vec model 格式

save_word2vec_format(fname, fvocab=None, binary=False)

Store the input-hidden weight matrix in the same format used by the original C word2vec-tool, for compatibility.

原始 C 語言版本 Word2Vec 的 向量格式之模型

Word2Vec 的 瓶頸

斷詞也是很重要的...

Word2Vec 的訓練結果

和輸入的語料庫息息相關

語料庫越大越完備

詞向量越好

Word2Vec 相關應用

機器翻譯

Word2Vec 模型主要用於詞粒度的機器翻譯

RNN 主要用於 **sentence-level** 的機器翻譯

推薦系統

任何基於 **co-occurrence 矩陣** 的算法
模型，都可以套用 **Word2Vec** 的思路加以改進。

推薦系統的協同過濾演算法

user-item 的 co-occurrence 矩陣

The End.