



自然語言處理與文字探勘

陳縉農 & 教研處

「版權聲明頁」

本投影片已經獲得作者授權台灣人工智慧學校得以使用於教學用途，如需取得重製權以及公開傳輸權需要透過台灣人工智慧學校取得著作人同意；如果需要修改本投影片著作，則需要取得改作權；另外，如果有需要以光碟或紙本等實體的方式傳播，則需要取得人工智慧學校散佈權。

課程內容

[講師投影片](#)

[資料與投影片](#)

[影片播放列表](#)

程式碼: ~/courses-tpe/NLP

1. Word Embedding

- Word Vector

- Glove

- gensim API

Code / Data 放在 hub 中的 courses 內

- 為維護課程資料，courses 中的檔案皆為 read-only, 如需修改請 cp 至自身的環境中
- 打開 terminal, 輸入
 - [台北班]
`cp -r courses-tpe/NLP/part2 <存放至本機的名稱>`
 - [新竹班]
`cp -r courses-hsi/NLP/part2 <存放至本機的名稱>`
 - [台中班]
`cp -r courses-txg/NLP/part2 <存放至本機的名稱>`



理論教授

Word Embedding

Word Embedding - Word2Vec

Word2Vec – Skip-Gram Model

- Goal: predict surrounding words within a window of each word
- Objective function: maximize the probability of any context word given the current center word

$$w_1, w_2, \dots, \underbrace{w_{t-m}, \dots, w_{t-1}, \underbrace{w_t}_{w_I}, w_{t+1}, \dots, w_{t+m}}_{\text{context window}}, \dots, w_{T-1}, w_T$$
$$p(w_{O,1}, w_{O,2}, \dots, w_{O,C} \mid w_I) = \prod_{c=1}^C p(w_{O,c} \mid w_I)$$

target word vector

$$C(\theta) = - \sum_{w_I} \sum_{c=1}^C \log p(w_{O,c} \mid w_I) \quad p(w_O \mid w_I) = \frac{\exp(v_{w_O}^T v_{w_I})}{\sum_j \exp(v_{w_j}^T v_{w_I})}$$

outside target word



Benefit: faster, easily incorporate a new sentence/document or add a word to vocab



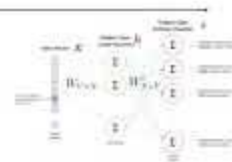
Word Embedding - Word2Vec Training

SGD Update for W

$$\frac{\partial C(\theta)}{\partial w_{ki}} = \frac{\partial C(\theta)}{\partial h_i} \frac{\partial h_i}{\partial w_{ki}}$$

$$\frac{\partial C(\theta)}{\partial h_i} = \sum_{j=1}^V \frac{\partial C(\theta)}{\partial s_j} \frac{\partial s_j}{\partial h_i} = \sum_{j=1}^V \sum_{c=1}^C (y_{jc} - t_{jc}) \cdot w'_{ij}$$

$s_j = v'_{wj}{}^T \cdot h$



Word Embedding - Negative

Negative Sampling

- Idea: only update a sample of output vectors

$$C(\theta) = -\log \sigma(v'_{w_O}{}^T v_{w_I}) + \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(v'_{w_j}{}^T v_{w_I})$$

$$v'_{w_j}{}^{(t+1)} = v'_{w_j}{}^{(t)} - \eta \cdot EI_j \cdot h$$

$$EI_j = \sigma(v'_{w_j}{}^T v_{w_I}) - t_j$$

$$v_{w_I}{}^{(t+1)} = v_{w_I}{}^{(t)} - \eta \cdot EH^T$$

$$EH = \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}} EI_j \cdot v'_{w_j}$$



Mikolov et al., "Distributed representations of words and phrases and their compositionality," in NIPS, 2013

© HARVARD

© HARVARD

© HARVARD



Word Embedding - Word2Vec Variants

Word2Vec Variants

- **Skip-gram**: predicting surrounding words given the target word (Mikolov+, 2013) better

$$p(w_{t-m}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+m} \mid w_t)$$

- **CBOW (continuous bag-of-words)**: predicting the target word given the surrounding words (Mikolov+, 2013)

$$p(w_t \mid w_{t-m}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+m})$$

- **LM (Language modeling)**: predicting the next words given the proceeding contexts (Mikolov+, 2013) first

$$p(w_{t+1} \mid w_t)$$



Mikolov et al., "Efficient estimation of word representations in vector space," in *NLP Workshop*, 2013.
Mikolov et al., "Distributed representations of words and phrases in continuous space word embeddings," in *AAAI*, 2013.



Word Embedding - GloVe

GloVe

- The relationship of w_i and w_j approximates the ratio of their co-occurrence probabilities with various w_k

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

$$F((v_{w_i} - v_{w_j})^T v_{\tilde{w}_k}) = \frac{P_{ik}}{P_{jk}} \quad F(\cdot) = \exp(\cdot)$$



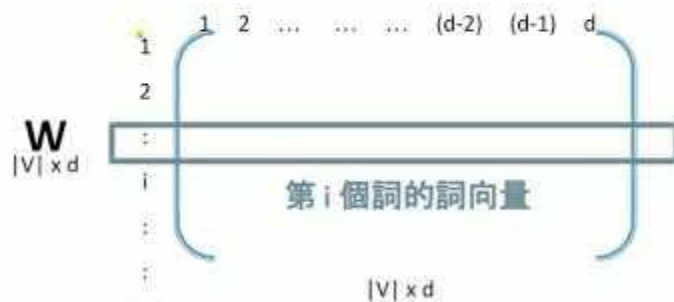
Pennington et al., "Glove: Global Vectors for Word Representation," in EMNLP, 2014



Word Embedding Conclusion

詞向量

- 訓練結束後，將 $1 \times |V|$ 字詞轉換成 d 維向量的矩陣



Why word embedding ?

- one-hot representation vs. distributed representation
- one-hot representation
 - 每個詞都是一個維度，彼此 independent
- 但每個單詞彼此 independent 明顯不符合
 - 語義：girl 和 woman
 - 複數：word 和 words
 - 時態詞：buy 和 bought
- 如何轉成 distributed representation ?
 - word embedding !!



教電腦從文章中解讀詞意

- word2vec
 - Tomas Mikolov et.al 2013 年於 Google 開發
 - Prediction-based method [reference](#)
- Glove
 - Jeffrey Pennington et al. 2015 年開發 (Stanford)
 - Count-based method [reference](#)



淺談 word2vec

- 語意相近的字較常出現在一起
 - Local window

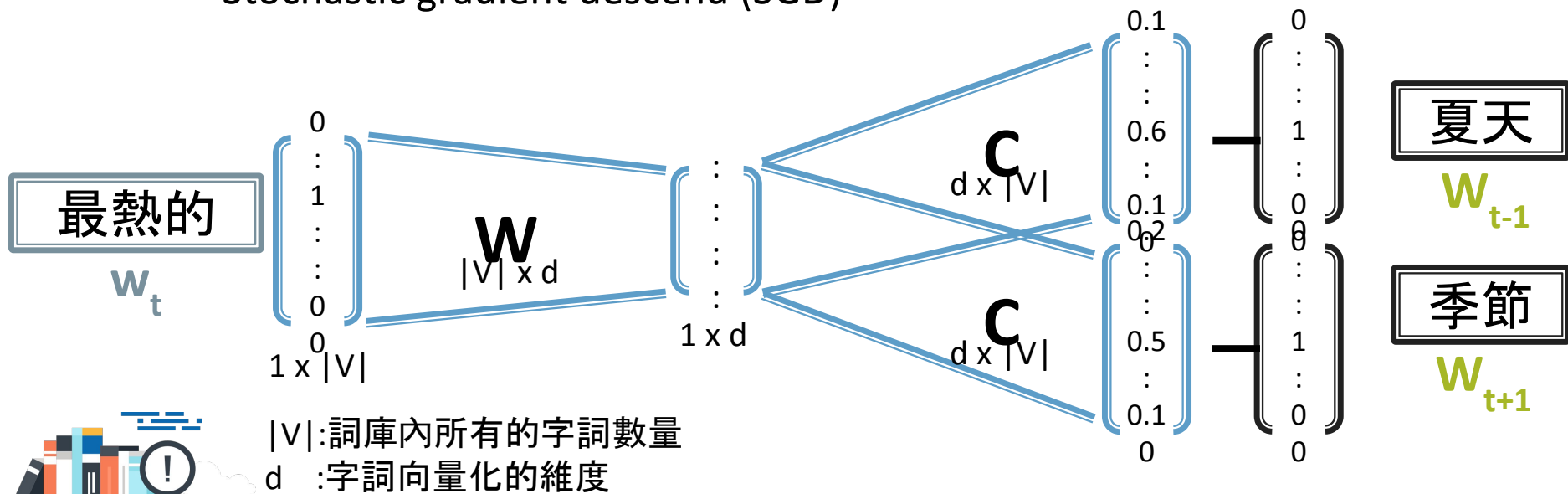


- 讓電腦玩克漏字填空學習 word embedding
 - Continuous bag of words
 - Skip-gram



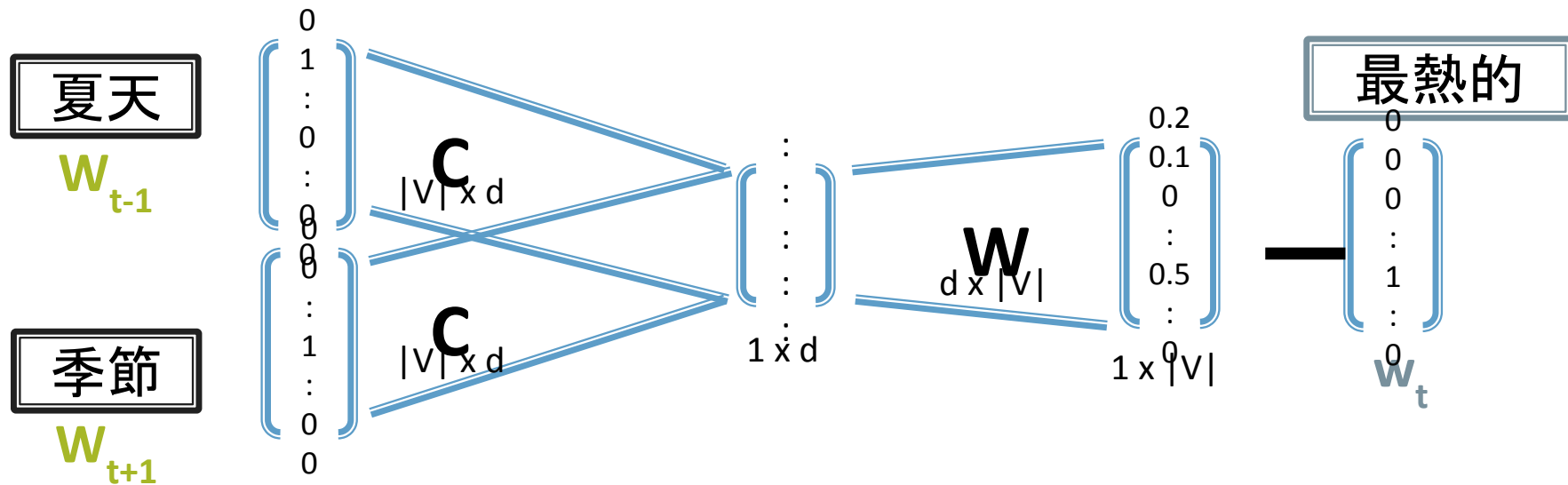
Skip-gram 模型

- 藉由 current word 推測 context words
- Neural network model
 - Stochastic gradient descend (SGD)



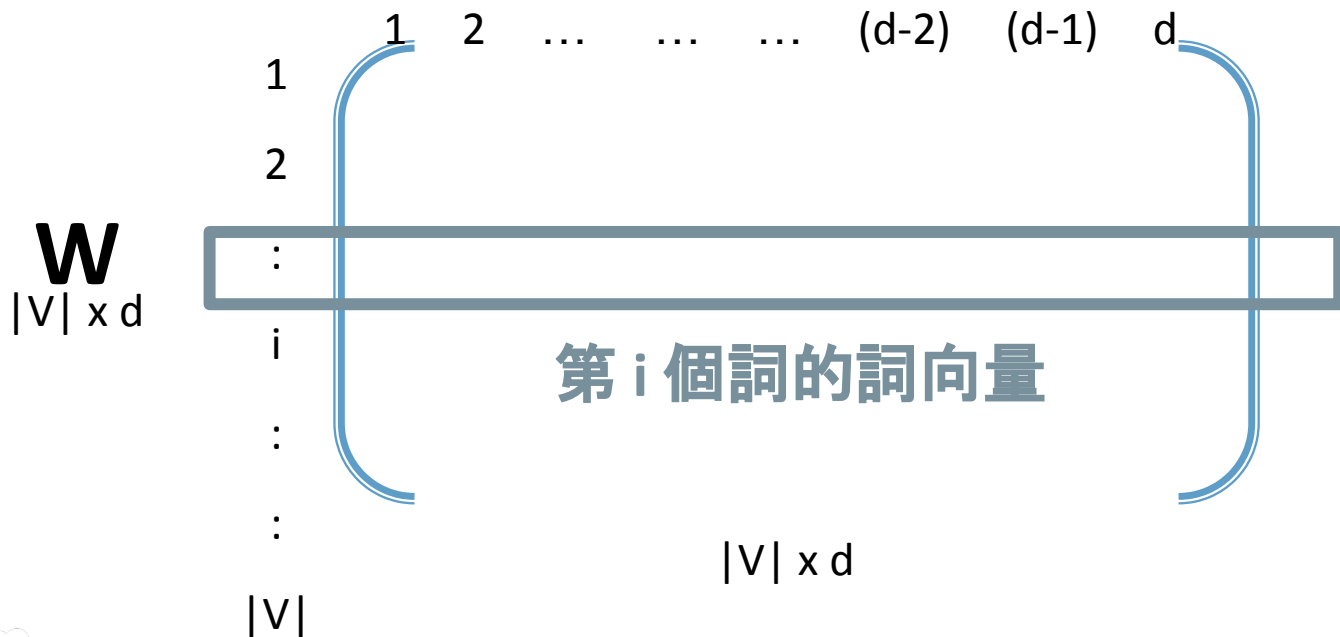
Continuous Bag of Words 模型

- 由 context words 推測 current word



詞向量

- 訓練結束後，將 $1 \times |V|$ 字詞轉換成 d 維向量的矩陣



gensim



gensim
topic modelling for humans

Get Expert Help From The Gensim Authors

- [Consulting](#) in Machine Learning & NLP
- Commercial document similarity engine: [ScaleText.ai](#)
- [Corporate trainings](#) in Python Data Science and Deep Learning

Home

Tutorials

Install

Support

API

About

Tutorials

The tutorials are organized as a series of examples that highlight various features of *gensim*. It is assumed that the reader is familiar with the [Python language](#), has [installed gensim](#) and read the [introduction](#).

The examples are divided into parts on:

[Command-Line Interface](#)



gensim 程式範例

- from gensim.models import [word2vec](#)
- sg 指定要用 CBOW (0) 或 skip-gram (1)

```
model = word2vec.Word2Vec(size=256, min_count=5, window=5, workers=10, sg=0)
```

```
: model.build_vocab(data)
```

- 訓練 model

```
for i in range(20):  
    random.shuffle(data)  
    model.train(data, total_examples=len(data), epochs=1)
```



word vector 結果範例

```
model.wv['人工智慧']
```

```
array([ 1.21915638,  0.52043217, -2.32643938,  1.93117321, -0.31237838,  
       -0.43847397, -0.76746583, -1.06985056, -0.68717992,  1.39475644,  
       -0.11933862,  0.90467405, -0.35574436, -0.03045041,  0.29790911,  
       -1.29267919, -1.00543356, -0.44228384, -0.27932352, -0.68174565,  
        0.25357905, -1.25369298,  1.50645053, -0.76435697, -1.02407789,  
        0.47269911,  2.20226884, -1.1822176 , -0.62692398,  0.53827405,  
       -1.1203531 , -0.4413209 ,  2.0929935 , -1.50965261,  0.42474991,  
       -0.23208205, -0.2567476 , -0.59872675, -0.76463807, -1.24145818,  
       -0.23847748,  1.37008536, -0.15632166, -1.79567409,  1.53807354,  
       -1.03542709, -1.40078819,  0.67201942,  1.68245482, -0.19959871,  
       -0.74226034, -0.12889996, -1.0839591 , -1.04049397,  0.2660369 ,  
        0.0702365 ,  0.0289956 , -0.04043816, -1.18576312, -0.55347919,  
       -0.67061466,  0.68260211, -0.2225527 ,  0.20782772, -0.53322947,  
       -0.44461682, -1.5774188 ,  0.82782865, -0.91934246,  0.29894483,  
       -1.03293765,  1.17304862,  1.46772027,  0.62551779, -2.06187868,  
        1.13207734, -0.80871838,  0.35943773, -1.12879944, -0.01194098,  
        0.5731349 ,  1.48613763,  0.68320924,  0.14039512,  1.14049029,  
       -0.08283772,  0.62286341,  0.73727089, -1.9068346 , -0.63660204,  
       -0.18024929, -0.95840788,  0.54120874,  0.80851376,  1.08375335,  
        0.20484021, -0.56473464, -1.69883382, -1.25609303, -0.13403395,  
        0.70849288, -0.15233018, -1.70290852, -1.39674747, -0.81129694,  
       -0.17469636,  0.66630858, -0.563559 , -0.02490964,  0.38999739,  
        0.24147406,  0.26629636, -0.87350655,  0.0266343 , -0.05054072,  
        0.60931027, -1.26577628, -0.79294878,  0.58070451,  0.92532027,  
       -0.17163102,  0.52050222, -1.31764424, -0.25275663, -0.46012122])
```



程式練習時間

- 03_word2vec_build.ipynb
 - 執行 Word2Vec 範例
 - 嘗試理解及調整參數，e.g. 改成 skip-gram



程式解說

```

jupyter 03_word2vec_build last checkpoint 12 hours ago (admin)

In [9]: # train word2vec model / shuffle data every epoch
for i in range(25):
    random.shuffle(data)
    model.train(data, total_examples=len(data), epochs=1)

2018-03-04 01:55:22,342: INFO: loading a fresh vocabulary
2018-03-04 01:55:22,950: INFO: min_count=5 retains 100734 unique words (35% of original 300118, drops 300384)
2018-03-04 01:55:22,951: INFO: min_count=5 leaves 1195104 word corpus (88% of original 1306784, drops 432477)
2018-03-04 01:55:23,183: INFO: deleting the raw events dictionary of 209119 items
2018-03-04 01:55:23,189: INFO: sample=0.001 downsampled 18 most-common words
2018-03-04 01:55:23,200: INFO: downsampled leaves estimated 1101078 word corpus (84.9% of prior 1306784)
2018-03-04 01:55:23,572: INFO: estimated required memory for 100734 words and 250 dimensions: 334006432 bytes
2018-03-04 01:55:23,902: INFO: resetting layer weights

In [9]: # train word2vec model / shuffle data every epoch
for i in range(25):
    random.shuffle(data)
    model.train(data, total_examples=len(data), epochs=1)

2018-03-04 01:55:48,024: INFO: worker thread finished; awaiting finish of 3 more threads
2018-03-04 01:55:48,030: INFO: worker thread finished; awaiting finish of 4 more threads
2018-03-04 01:55:48,042: INFO: worker thread finished; awaiting finish of 5 more threads
2018-03-04 01:55:48,046: INFO: worker thread finished; awaiting finish of 6 more threads
2018-03-04 01:55:48,048: INFO: worker thread finished; awaiting finish of 7 more threads
2018-03-04 01:55:48,050: INFO: worker thread finished; awaiting finish of 8 more threads
2018-03-04 01:55:48,051: INFO: EPOCH - 1: training on 1106784 raw words (1000000 effective words); took 8.4s, 11443
80 effective words/s
2018-03-04 01:55:48,054: INFO: training on a 1106784 raw words (1000000 effective words); took 9.5s, 110840 effecti
ve words/s
2018-03-04 01:55:48,263: WARNING: Effective 'alpha' higher than previous training epoch
2018-03-04 01:55:48,344: INFO: training model with 10 workers on 100734 vocabulary and 250 features, using sg=0 haw=0
sample=0.001 negative=5 window=5
2018-03-04 01:55:47,113: INFO: EPOCH 1 - PROGRESS: at 8.84k examples, 89168 words/s, in_queue 17, out_queue 1
2018-03-04 01:55:48,322: INFO: EPOCH 1 - PROGRESS: at 18.75k examples, 104503 words/s, in_queue 19, out_queue 0
2018-03-04 01:55:49,379: INFO: EPOCH 1 - PROGRESS: at 21.03k examples, 109824 words/s, in_queue 19, out_queue 0
2018-03-04 01:55:50,350: INFO: EPOCH 1 - PROGRESS: at 42.18k examples, 112745 words/s, in_queue 19, out_queue 0
2018-03-04 01:55:51,352: INFO: EPOCH 1 - PROGRESS: at 62.99k examples, 112922 words/s, in_queue 19, out_queue 0

```

