NARLabs 財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing

TensorFlow及Keras深度學習介紹 -- 進階篇

彭鈺翔助理研究員、楊筑鈞助理研究員
E-mail:
潘怡倫 研究員
E-mail: serenapan@narlabs.org.tw
國家高速網路與計算中心

課程大綱

- Deep Learning with TensorFlow
  - 遞迴式神經網路(RNN)
  - LSTM
  - Application –Object Detection
- Labs:
  - Create a container
  - git clone https://github.com/bingo830422/AI-course.git

對象

- 以基本觀念代入深度學習
- 不介紹故事，不說明深度學習有多好
- 上完課就會寫深度學習程式
- 案例分析與實作，請接續選修國網一系列課程

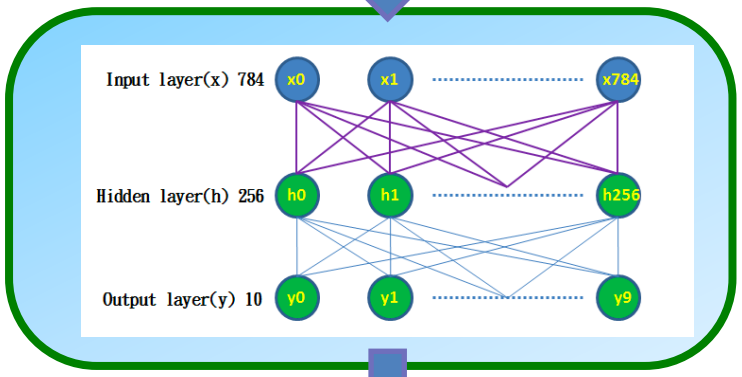
The Whole Pic. Of Learning

Dataset

Pre-Processing

Features Extraction

1. Input Features



2. Output from Model

Predictions

Labels (True Value)

3. Cross Entropy

Loss Function

Error

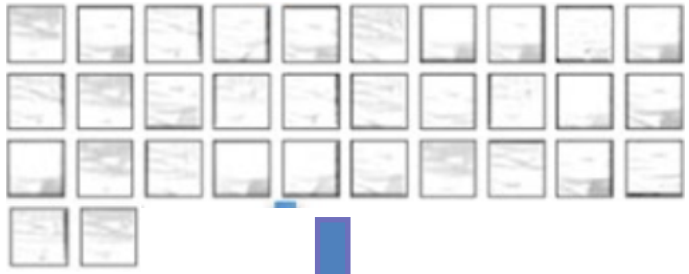
Optimizer

4. Update Weights and Bias

CNN – Logical Flow

- 32*32 with 1 image

The first convolution



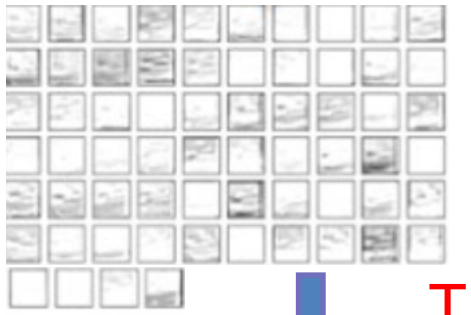
- 32*32
- 32 images

The first pooling

- 16*16
- 32 images

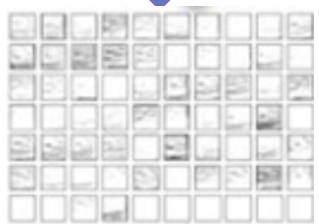
The second convolution

- 16*16
- 64 images

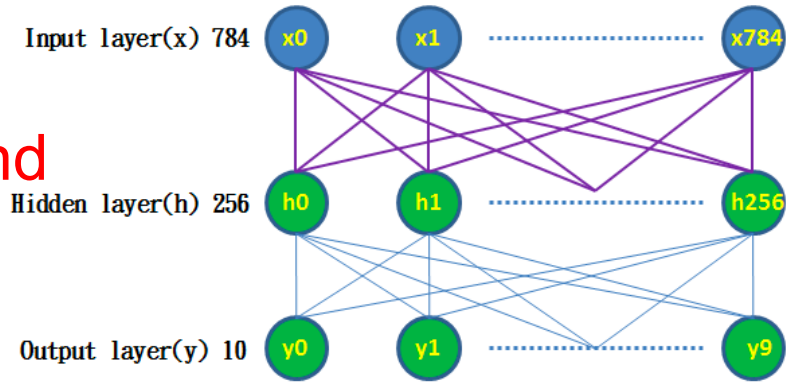


The second pooling

- 8*8
- 64 images



FC



前言 - 所使用到的工具鏈

操作整合環境（類似IDE）

jupyter

視覺化工具

matplotlib

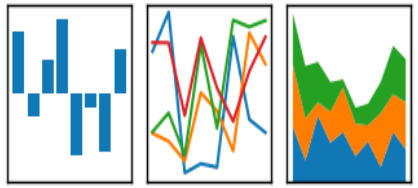
分析工具

NumPy

最基礎數值處理

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



資料處理
（偏重於資料Clean的動作，如行列調整、排列等動作）

scikit learn

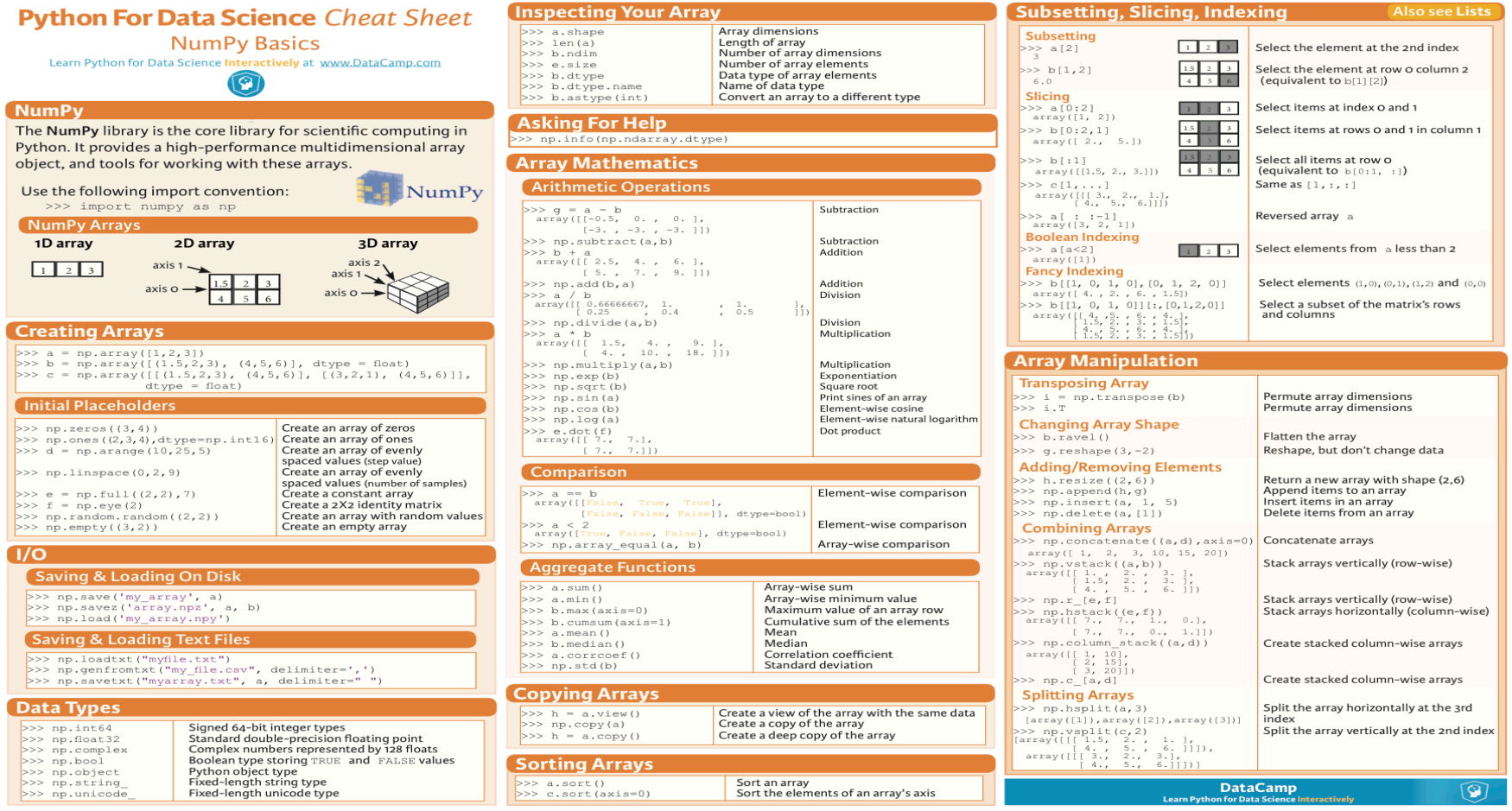
資料處理
（偏重於資料內容操作，如：符號字串轉換）

前言 - Jupyter Notebook

- Cell：
  - Code cells
  - Markdown cells
- 交互式：每運行一個cell，就會產生結果
- Magic keywords
  - 配置、調整
  - 畫圖

前言 - NumPy

- https://github.com/kailashahirwar/cheatsheets-ai



Python For Data Science *Cheat Sheet*

NumPy Basics

Learn Python for Data Science Interactively at www.DataCamp.com

NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays

1D array

2D array

3D array

Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Initial Placeholders

```
>>> np.zeros((3,4))
>>> np.ones((2,3,4),dtype=np.int16)
>>> d = np.arange(10,25,5)
>>> np.linspace(0,2,9)
>>> e = np.full((2,2),7)
>>> f = np.eye(2)
>>> np.random.random((2,2))
>>> np.empty((3,2))
```

Create an array of zeros
Create an array of ones
Create an array of evenly spaced values (step value)
Create an array of evenly spaced values (number of samples)
Create a constant array
Create a 2X2 identity matrix
Create an array with random values
Create an empty array

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

```
>>> np.int64
>>> np.float32
>>> np.complex
>>> np.bool
>>> np.object
>>> np.string_
>>> np.unicode_
```

Signed 64-bit integer types
Standard double-precision floating point
Complex numbers represented by 128 floats
Boolean type storing TRUE and FALSE values
Python object type
Fixed-length string type
Fixed-length unicode type

Inspecting Your Array

```
>>> a.shape
>>> len(a)
>>> b.ndim
>>> e.size
>>> b.dtype
>>> b.dtype.name
>>> b.astype(int)
```

Array dimensions
Length of array
Number of array dimensions
Number of array elements
Data type of array elements
Name of data type
Convert an array to a different type

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

```
>>> g = a - b
  array([[-0.5,  0. ,  0. ],
         [-3. , -3. , -3. ]])
>>> np.subtract(a,b)
>>> b + a
  array([[ 2.5,  4. ,  6. ],
         [ 5. ,  7. ,  9. ]])
>>> np.add(b,a)
>>> a / b
  array([[ 0.66666667,  1.        ,  1.        ],
         [ 0.25      ,  0.4       ,  0.5       ]])
>>> np.divide(a,b)
>>> a * b
  array([[  1.5,   4. ,   9. ],
         [  4. ,  10. ,  18. ]])
>>> np.multiply(a,b)
>>> np.exp(b)
>>> np.sqrt(b)
>>> np.sin(a)
>>> np.cos(b)
>>> np.log(a)
>>> e.dot(f)
  array([[ 7.,  7.],
         [ 7.,  7.]])
```

Subtraction
Subtraction
Addition
Addition
Division
Division
Multiplication
Multiplication
Exponentiation
Square root
Print sines of an array
Element-wise cosine
Element-wise natural logarithm
Dot product

Comparison

```
>>> a == b
  array([[False,  True,  True],
         [False, False, False]], dtype=bool)
>>> a < 2
  array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)
```

Element-wise comparison
Element-wise comparison
Array-wise comparison

Aggregate Functions

```
>>> a.sum()
>>> a.min()
>>> b.max(axis=0)
>>> b.cumsum(axis=1)
>>> a.mean()
>>> b.median()
>>> a.corrcoef()
>>> np.std(b)
```

Array-wise sum
Array-wise minimum value
Maximum value of an array row
Cumulative sum of the elements
Mean
Median
Correlation coefficient
Standard deviation

Copying Arrays

```
>>> h = a.view()
>>> np.copy(a)
>>> h = a.copy()
```

Create a view of the array with the same data
Create a copy of the array
Create a deep copy of the array

Sorting Arrays

```
>>> a.sort()
>>> c.sort(axis=0)
```

Sort an array
Sort the elements of an array's axis

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]
  3
>>> b[1,2]
  6.0
```

Select the element at the 2nd index
Select the element at row 0 column 2 (equivalent to b[1][2])

Slicing

```
>>> a[0:2]
  array([1, 2])
>>> b[0:2,1]
  array([ 2.,  5.])
>>> b[:1]
  array([[1.5, 2., 3.]])
>>> c[1,...]
  array([[[ 3.,  2.,  1.],
          [ 4.,  5.,  6.]]])
>>> a[ : :-1]
  array([3, 2, 1])
```

Select items at index 0 and 1
Select items at rows 0 and 1 in column 1
Select all items at row 0 (equivalent to b[0:1, :])
Same as [1,:,:]
Reversed array a

Boolean Indexing

```
>>> a[a<2]
  array([1])
```

Select elements from a less than 2

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]
  array([ 4. ,  2. ,  6. ,  1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]
  array([[ 4. ,  5. ,  6. ,  4. ],
         [ 1.5,  2. ,  3. ,  1.5],
         [ 4. ,  5. ,  6. ,  4. ],
         [ 1.5,  2. ,  3. ,  1.5]])
```

Select elements (1,0),(0,1),(1,2) and (0,0)
Select a subset of the matrix's rows and columns

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b)
>>> i.T
```

Permute array dimensions
Permute array dimensions

Changing Array Shape

```
>>> b.ravel()
>>> g.reshape(3,-2)
```

Flatten the array
Reshape, but don't change data

Adding/Removing Elements

```
>>> h.resize((2,6))
>>> np.append(h,g)
>>> np.insert(a, 1, 5)
>>> np.delete(a,[1])
```

Return a new array with shape (2,6)
Append items to an array
Insert items in an array
Delete items from an array

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
  array([ 1,  2,  3, 10, 15, 20])
>>> np.vstack((a,b))
  array([[ 1. ,  2. ,  3. ],
         [ 1.5,  2. ,  3. ],
         [ 4. ,  5. ,  6. ]])
>>> np.r_[e,f]
>>> np.hstack((e,f))
  array([[ 7.,  7.,  1.,  0.],
         [ 7.,  7.,  0.,  1.]])
>>> np.column_stack((a,d))
  array([[ 1, 10],
         [ 2, 15],
         [ 3, 20]])
>>> np.c_[a,d]
```

Concatenate arrays
Stack arrays vertically (row-wise)
Stack arrays vertically (row-wise)
Stack arrays horizontally (column-wise)
Create stacked column-wise arrays
Create stacked column-wise arrays

Splitting Arrays

```
>>> np.hsplit(a,3)
  [array([1]),array([2]),array([3])]
>>> np.vsplit(c,2)
[array([[[ 1.5,  2. ,  1. ],
         [ 4. ,  5. ,  6. ]]]),
 array([[[ 3. ,  2. ,  3. ],
         [ 4. ,  5. ,  6. ]]])]
```

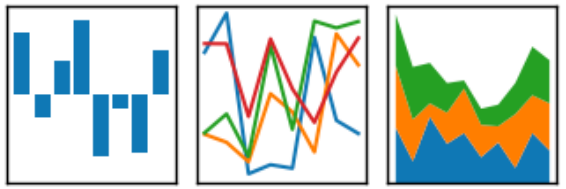
Split the array horizontally at the 3rd index
Split the array vertically at the 2nd index

DataCamp
Learn Python For Data Science Interactively

前言 - pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The **Pandas** library is built on NumPy and provides easy-to-use **data structures** and **data analysis** tools for the Python programming language.

Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

A **one-dimensional** labeled array capable of holding any data type

Index: A 3, B -5, C 7, D 4

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

Columns: Country, Capital, Population

Index: 1, 2, 3

	Country	Capital	Population
1	Belgium	Brussels	11190846
2	India	New Delhi	1303171035
3	Brazil	Brasília	207847528

A **two-dimensional** labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
 -5
```
Get one element

```
>>> df[1:]
   Country     Capital  Population
1   India   New Delhi  1303171035
2  Brazil    Brasília   207847528
```
Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[0],[0])
'Belgium'
>>> df.iat([0],[0])
'Belgium'
```
Select single value by row & column

By Label

```
>>> df.loc[0], ['Country'])
'Belgium'
>>> df.at([0], ['Country'])
'Belgium'
```
Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital    Brasília
Population  207847528
```
Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0     Brussels
1    New Delhi
2     Brasília
```
Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```
Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
>>> df[df['Population']>1200000000]
```
Series s where value is not >1
s where value is <-1 or >2
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```
Set index a of Series s to 6

Dropping

```
>>> s.drop(['a', 'c'])
>>> df.drop('Country', axis=1)
```
Drop values from rows (axis=0)
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index(by='Country')
>>> s.order()
>>> df.rank()
```
Sort by row or column index
Sort a series by its values
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
>>> df.index
>>> df.columns
>>> df.info()
>>> df.count()
```
(rows,columns)
Describe index
Describe DataFrame columns
Info on DataFrame
Number of non-NA values

Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min()/df.max()
>>> df.idmin()/df.idmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```
Sum of values
Cummulative sum of values
Minimum/maximum values
Minimum/Maximum index value
Summary statistics
Mean of values
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
>>> df.applymap(f)
```
Apply function
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
  a    10.0
  b     NaN
  c     5.0
  d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
  a    10.0
  b    -5.0
  c     5.0
  d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> pd.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

read_sql() is a convenience wrapper around read_sql_table() and read_sql_query()

```
>>> pd.to_sql('myDf', engine)
```

DataCamp
Learn Python for Data Science Interactively

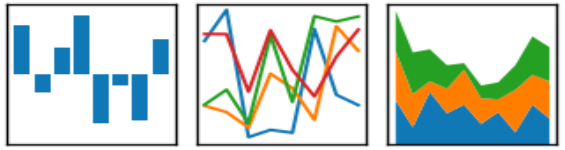
前言 - NumPy vs. Pandas

NumPy

-》升级

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



1	0	0	0	1
1	0	1	0	0
0	0	0	0	1
1	0	1	0	0
0	0	0	0	1

	Survived	Pclass	Sex	Age	SibSp	Parch
0	0	3	male	22.0	1	0
1	1	1	female	38.0	1	0
2	1	3	female	26.0	0	0
3	1	1	female	35.0	1	0
4	0	3	male	35.0	0	0

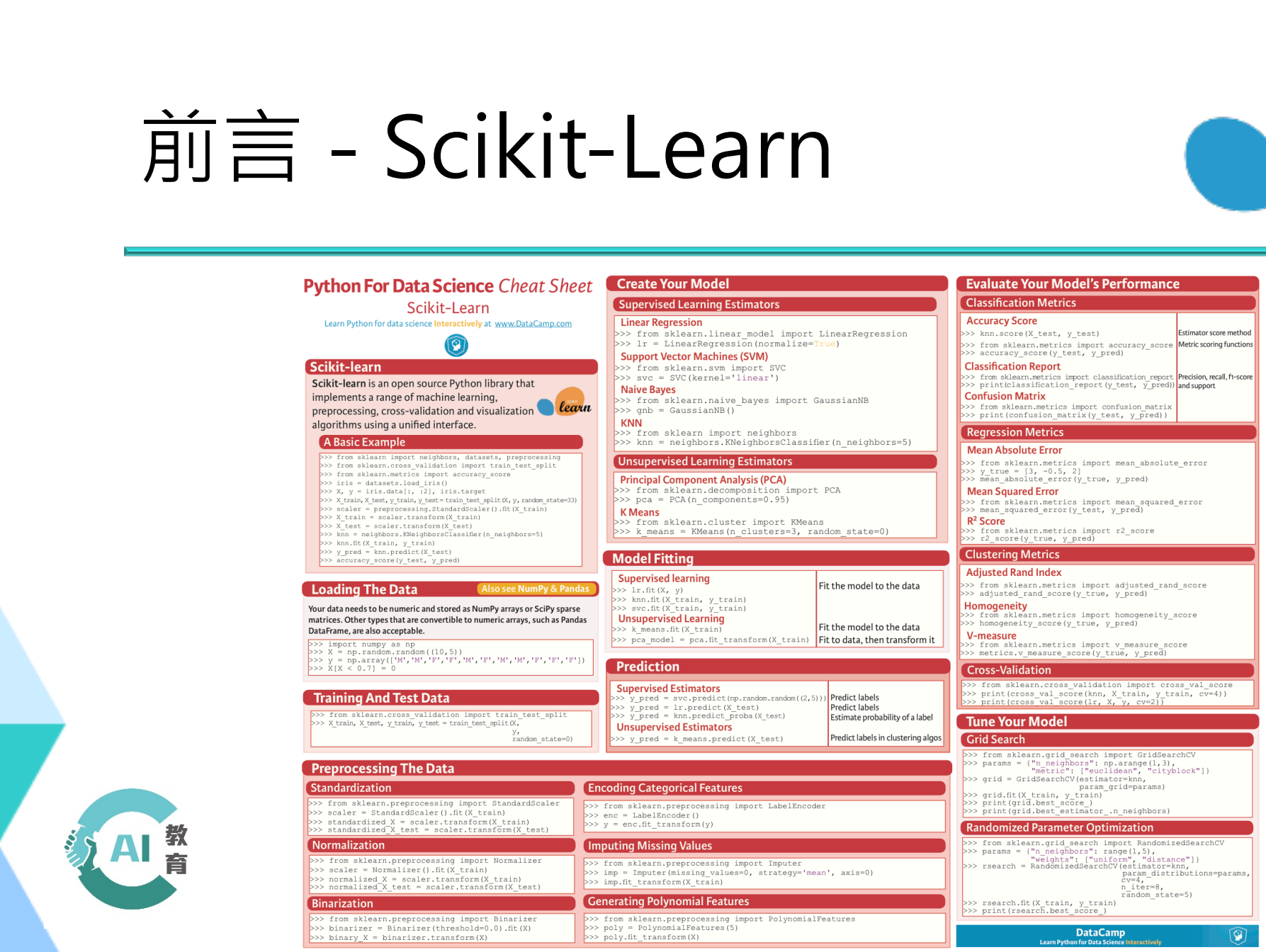
NumPy	pandas
表達資料的類型必須全部一致	只有同一列一致
基本方法	更多更處理資料的方法
直接變成pandas表示 ⟷	只有內容全是同一類型的才可降為numpy表示

前言 - Scikit-Learn

- Import 新的類別：找到你所想要的類別
- fit：輸入資料
- Transform：轉換資料


```
from sklearn.preprocessing import LabelEncoder

def sex_to_int(data):
    le = LabelEncoder()
    le.fit(["male","female"])
    data["Sex"]=le.transform(data["Sex"])
    return data
```

前言 - Scikit-Learn



Python For Data Science *Cheat Sheet*

Scikit-Learn

Learn Python for data science Interactively at www.DataCamp.com

Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test= train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

Preprocessing The Data

Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

Imputing Missing Values

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

Create Your Model

Supervised Learning Estimators

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

Model Fitting

Supervised learning	
`>>> lr.fit(X, y)`	Fit the model to the data
`>>> knn.fit(X_train, y_train)`	
`>>> svc.fit(X_train, y_train)`	
Unsupervised Learning	
`>>> k_means.fit(X_train)`	Fit the model to the data
`>>> pca_model = pca.fit_transform(X_train)`	Fit to data, then transform it

Prediction

Supervised Estimators	
`>>> y_pred = svc.predict(np.random.random((2,5)))`	Predict labels
`>>> y_pred = lr.predict(X_test)`	Predict labels
`>>> y_pred = knn.predict_proba(X_test)`	Estimate probability of a label
Unsupervised Estimators	
`>>> y_pred = k_means.predict(X_test)`	Predict labels in clustering algos

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score — Estimator score method

```
>>> knn.score(X_test, y_test)
```

Metric scoring functions

```
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Classification Report — Precision, recall, f1-score and support

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

Tune Your Model

Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
              "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
              "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
                                 param_distributions=params,
                                 cv=4,
                                 n_iter=8,
                                 random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

DataCamp
Learn Python for Data Science Interactively

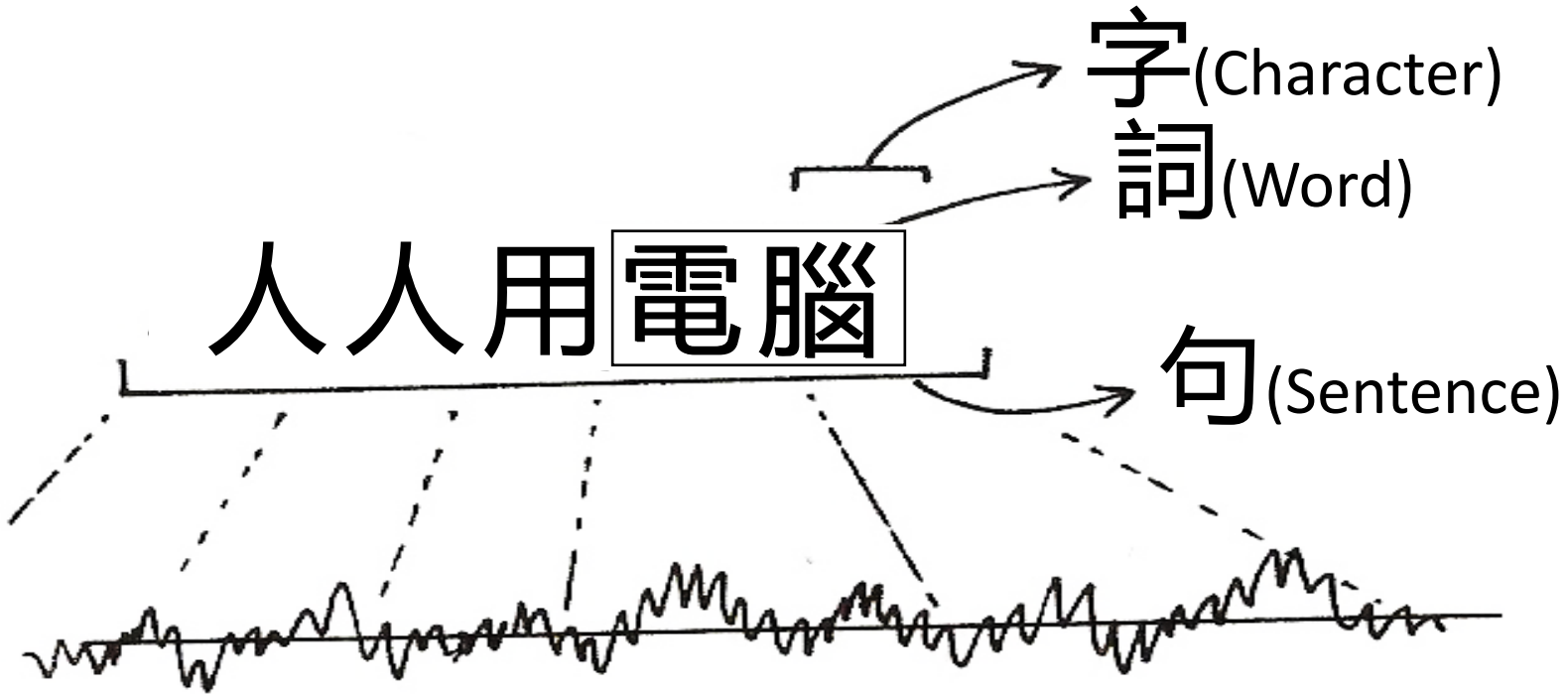
前言 - Matlabplot

matplotlib

- import matplotlib.pyplot as plt
- x=[1,2,3,4,5]
- y=[2,3,4,5,6]
- plt.plot(x, y)
- plt.show()

自然語言處理(NLP) Problem (1/2)

- 英文 - 單字開始，進而到片語(Phrase)、句子(Sentence)，加上文法(Syntax)、語意(Semantics)解析



自然語言處理(NLP) Problem (2/2)

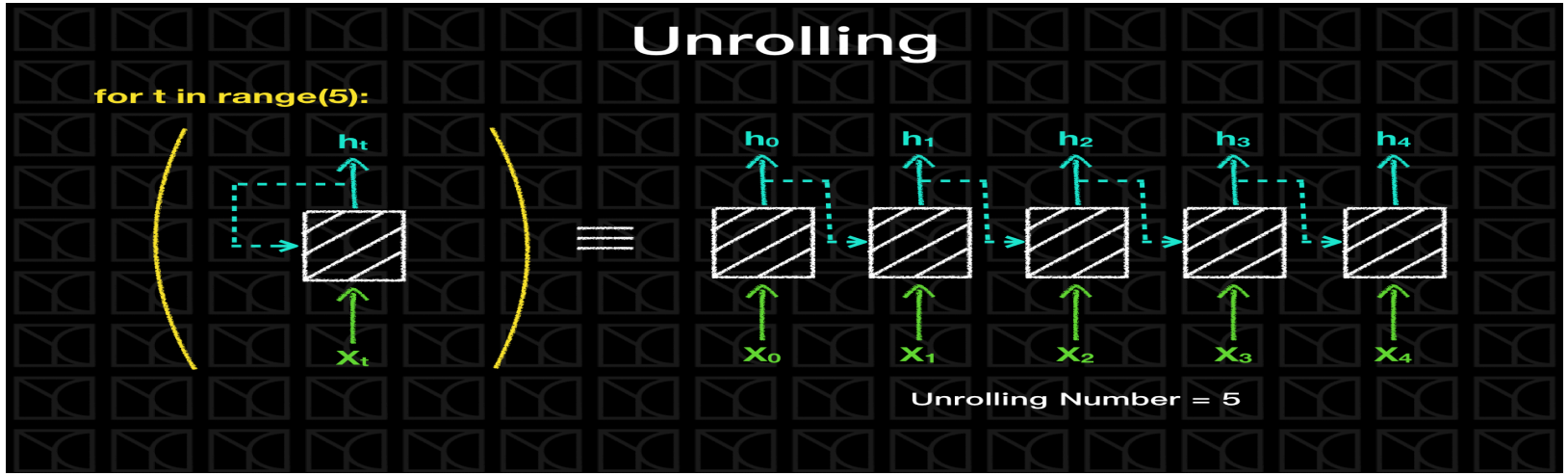
- 字句有長有短，即 input 變數數目不等，回應也是一樣的狀況，output 變數數目也不等。
- 語文會有上下文的關係，斷章取義(政治媒體的專長?)常會扭曲全文代表的意義。
- 人類擁有記憶力，對某些人、事、物會有深度的連結，講到『川普』，可能會想到『北韓』，若機器擁有記憶力機制，預測準確率就會高很多。

自然語言處理(NLP) 演算法

- 『循環神經網路』(Recurrent Neural Network, RNN)
- 長短期記憶網路(Long Short Term Memory Network, LSTM)
- GRU (Gated Recurrent Unit 閘門循環單元)

遞迴式神經網路(RNN) - 簡介

- MLP與CNN特點：假設輸入是一個獨立的**沒有**上下文聯繫的單位
- RNN – Recurrent Neural Networks：
  - Neural Network具有**時序性**
  - 不斷的輸入新資訊時，也能同時**保有歷史資訊**的影響
  - 故最簡單的作法就是將Output的結果保留，等到新資訊進來時，將新的資訊和舊的Output一起考量來訓練Neural Network -> 網路賦予了**記憶**的能力

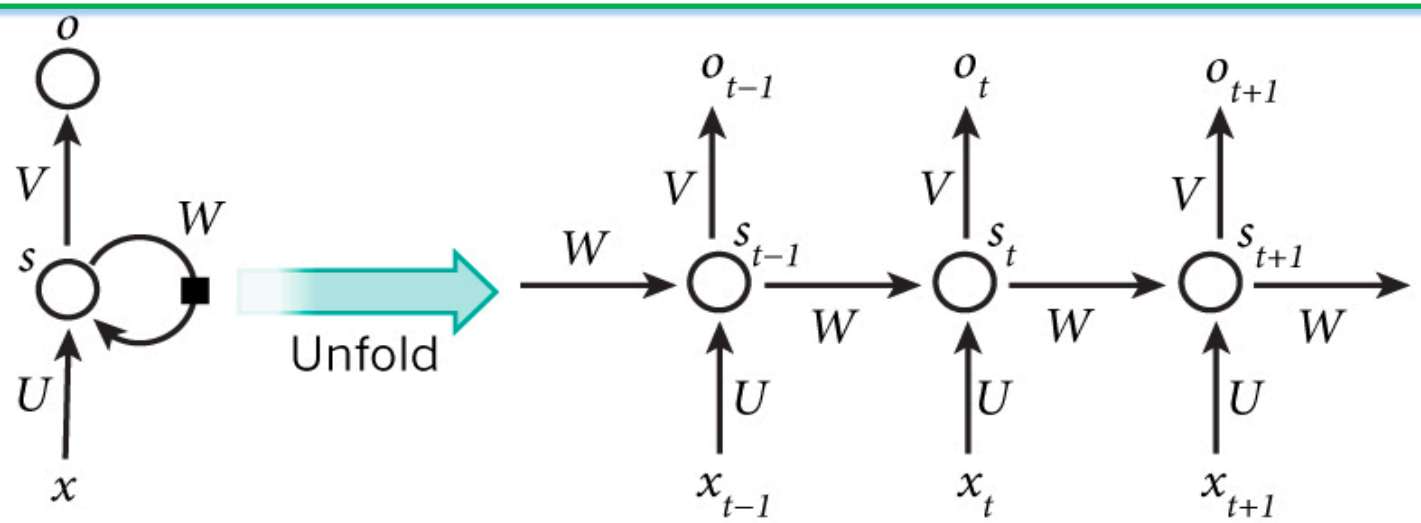


遞迴式神經網路(RNN) – 應用

- 語音辨識
- 語言模型
- 自然語言處理
- 機器翻譯
- 時間序列分析

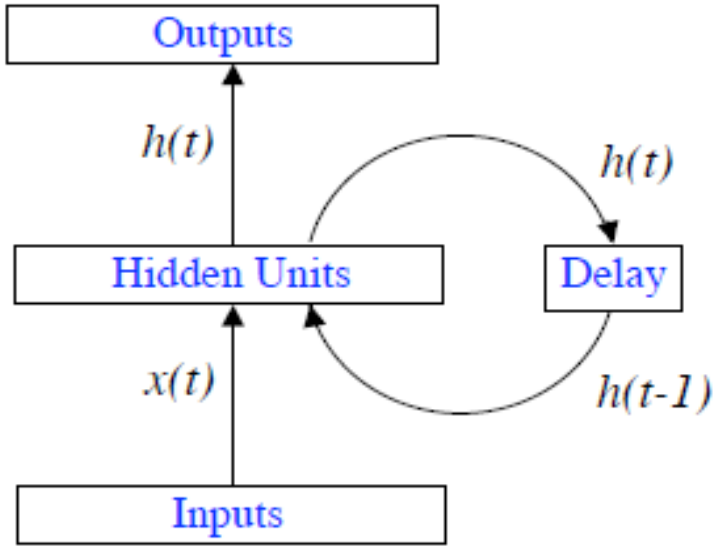
參數說明

- Xt 是 t 時間點神經網路的輸入
- Ot 是 t 時間點神經網路的輸出
- (U,V,W) 都是神經網路的參數, W 參數是 t-1 時間點的輸出, 但是作為 t 時間點的輸入.
- St 是隱藏狀態, 代表神經網路上的記憶, 是經過目前時間點的輸入 Xt, 再加上上個時間點的狀態 St-1, 再加上 U 與 W 的參數, 共同評估的結果: St = f(U*Xt + W*St-1)



遞迴式神經網路(RNN) – 原理 (1/7)

遞歸神經網絡（RNN）是連接模型，能夠選擇性地跨時序列步驟傳遞訊息，同時一次處理一個元素的順序資料。



The simplest form of ***fully recurrent neural network*** is an MLP with the previous set of hidden unit activations feeding back into the network along with the inputs

允許先前輸入的“記憶性”保持在網絡的內部狀態，從而影響網路輸出

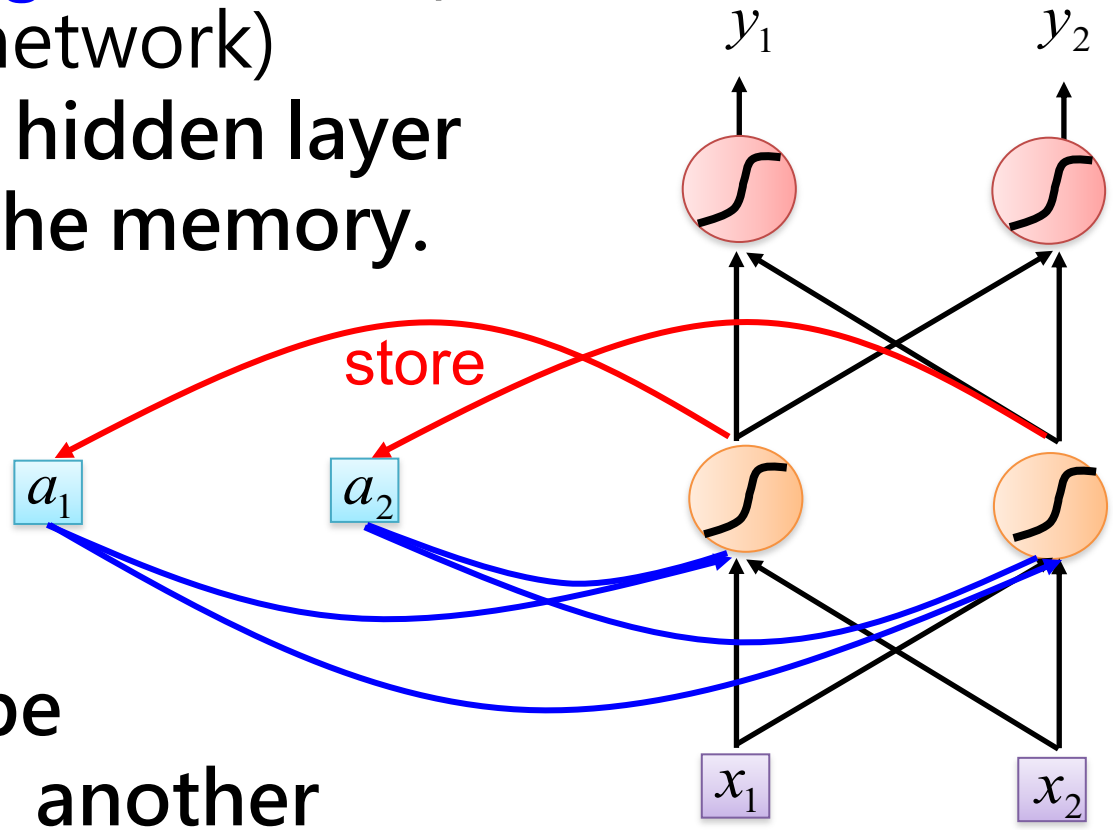
$$h(t) = f_H(W_{IH}x(t) + W_{HH}h(t-1))$$
$$y(t) = f_O(W_{HO}h(t))$$

f_H and f_O are the activation function for hidden and output unit; W_{IH}, W_{HH}, and W_{HO} are connection weight matrices which are learnt by training

遞迴式神經網路(RNN) – 原理 (2/7)

- 隨著時間的展開(***unfolding over time***)，可以將循環網路轉換成前饋網路(feed-forward network)

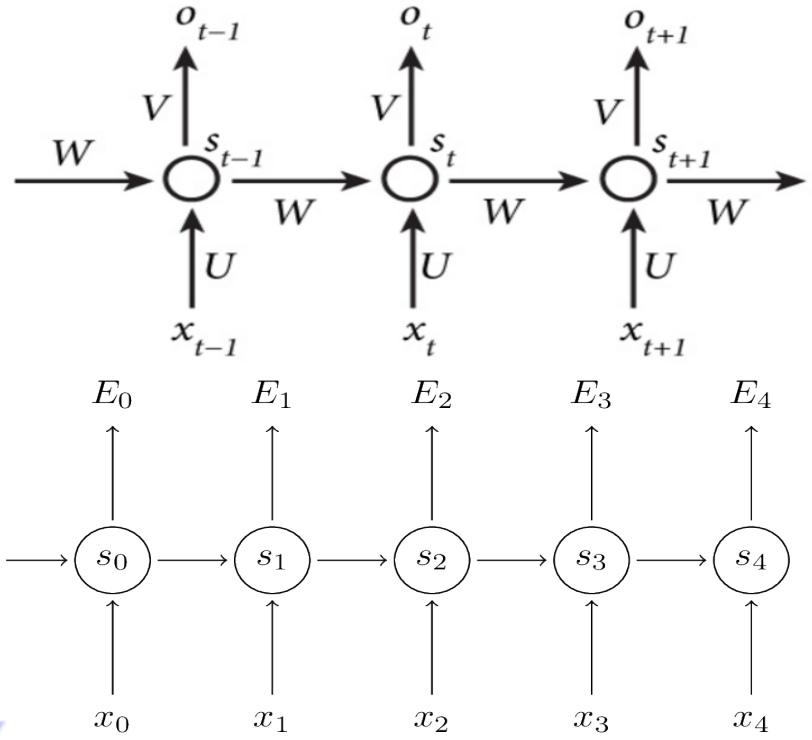
The output of hidden layer are stored in the memory.



Memory can be considered as another input.

遞迴式神經網路(RNN) – 原理 (3/7)

- Training RNNs (determine the parameters)
  - Back Propagation Through Time (BPTT) is often used to learn the RNN BPTT is an extension of the back-propagation (BP)



- The output of this RNN is $\hat{y}_t$

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$\hat{y}_t = softmax(Vs_t)$$

- The loss/error function of this network is

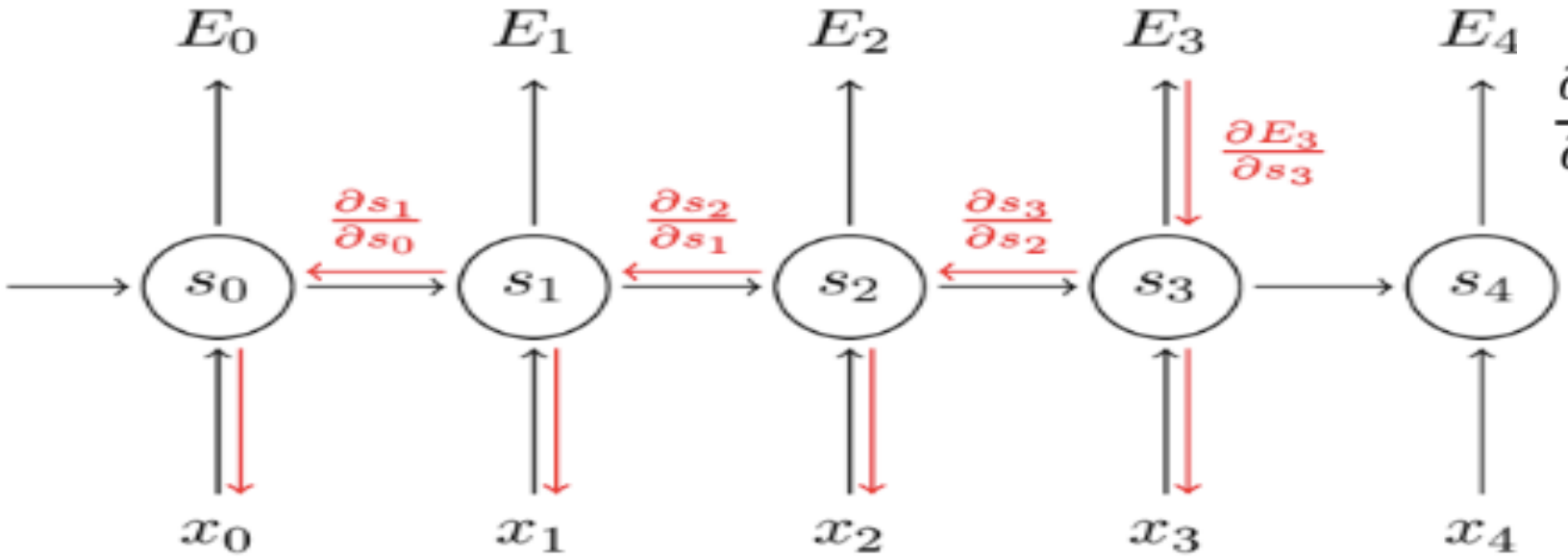
$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$ → The error at each time step

$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$ → the total loss is the sum of the errors at each time step

遞迴式神經網路(RNN) – 原理 (4/7)

- Training RNNs (determine the parameters)
  - The gradients of the error with respect to our parameters

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$



$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

遞迴式神經網路(RNN) – 原理 (5/7)

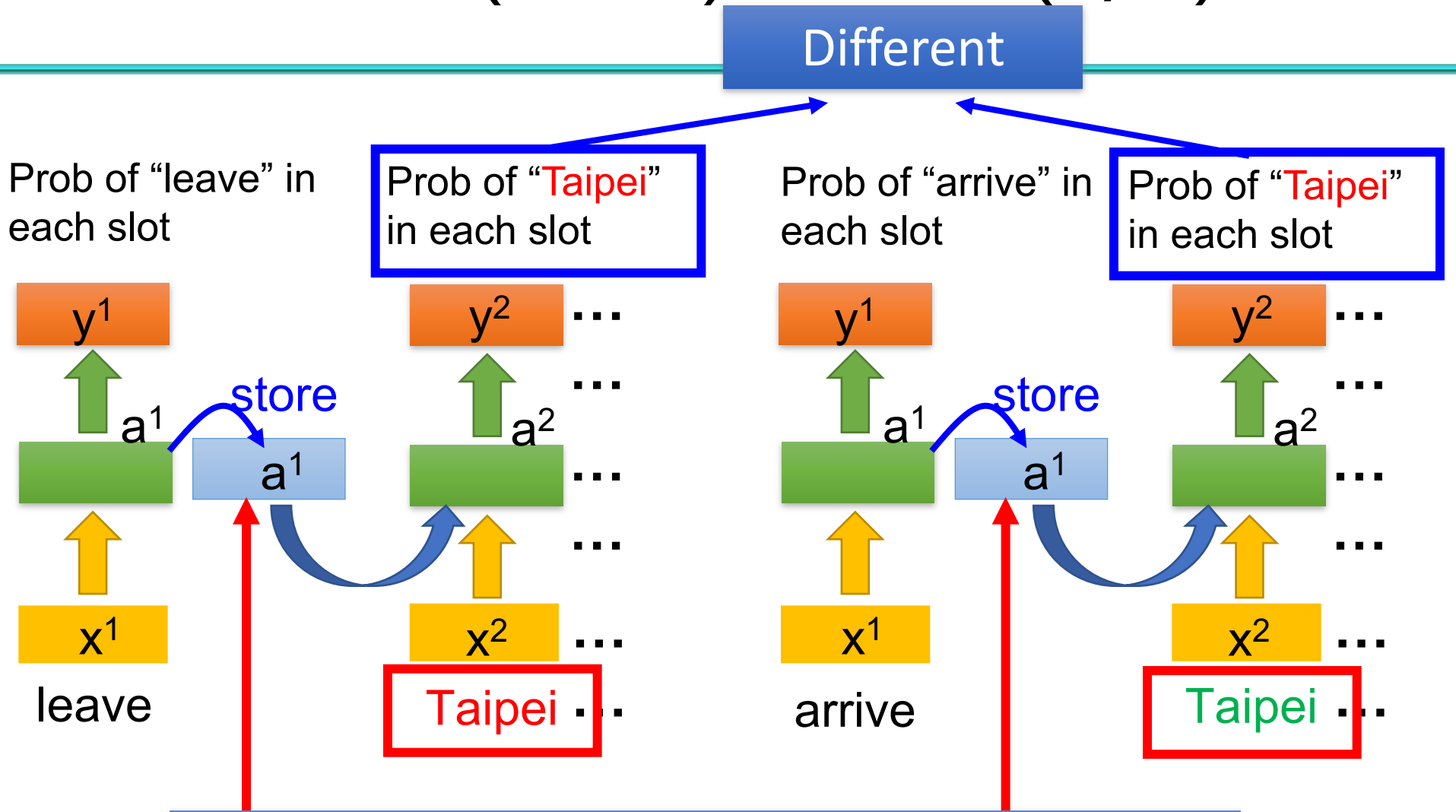
- Training RNNs (determine the parameters)
  - The gradient at each time step

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$ → Chain Rule

$$s_3 = \tanh(Ux_1 + Ws_2)$$ → s_3 depends on W and s_1, we cannot simply treat s_2 a constant

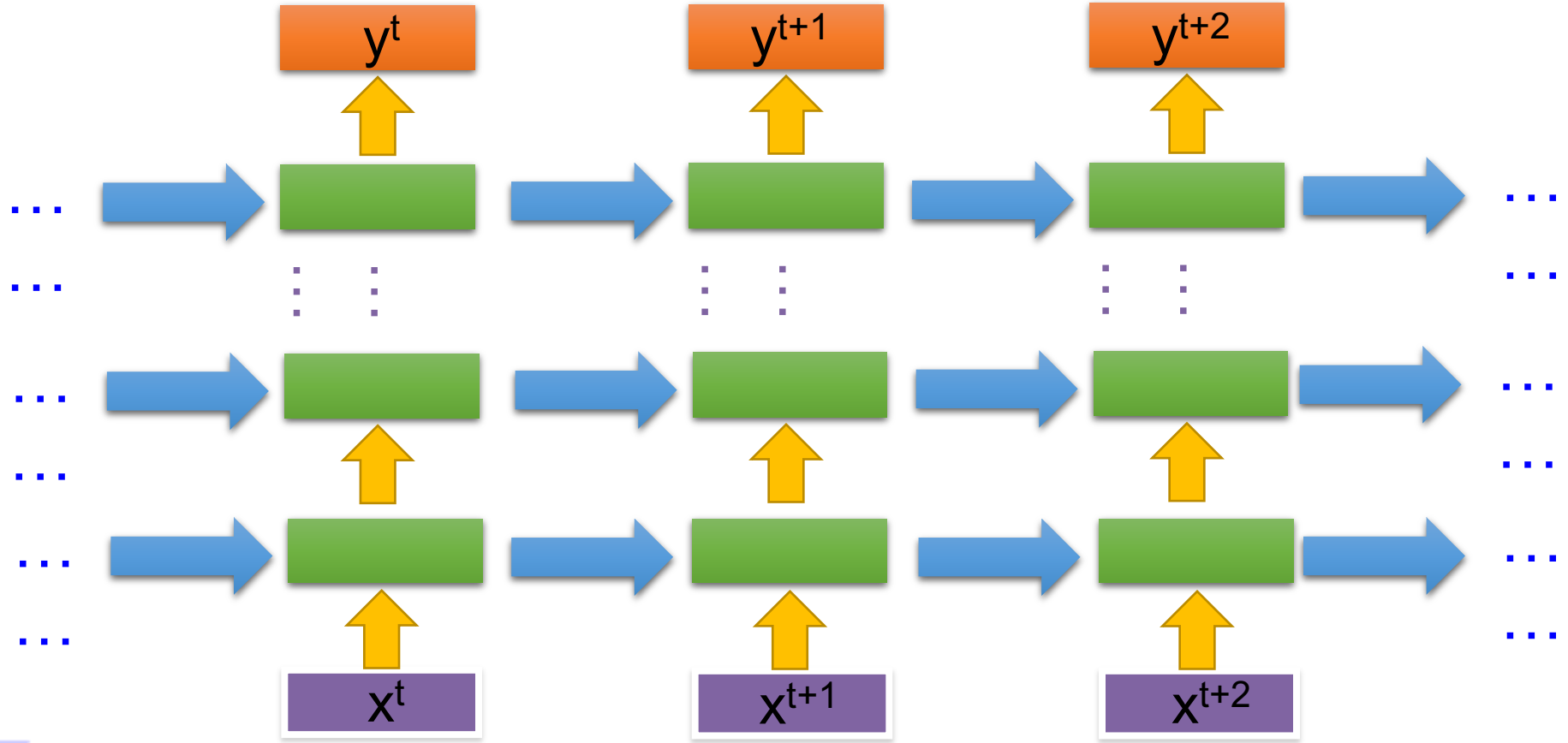
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3}\frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$ → Apply Chain Rule again on s_k

遞迴式神經網路(RNN) – 原理 (6/7)



遞迴式神經網路(RNN) – 原理 (7/7)

- RNNs當然也可以很deep



遞迴式神經網路(RNN) – 問題 (1/2)

- 梯度消失問題
  - Why?
  - Let’s take a closer look at the gradient we calculated below:

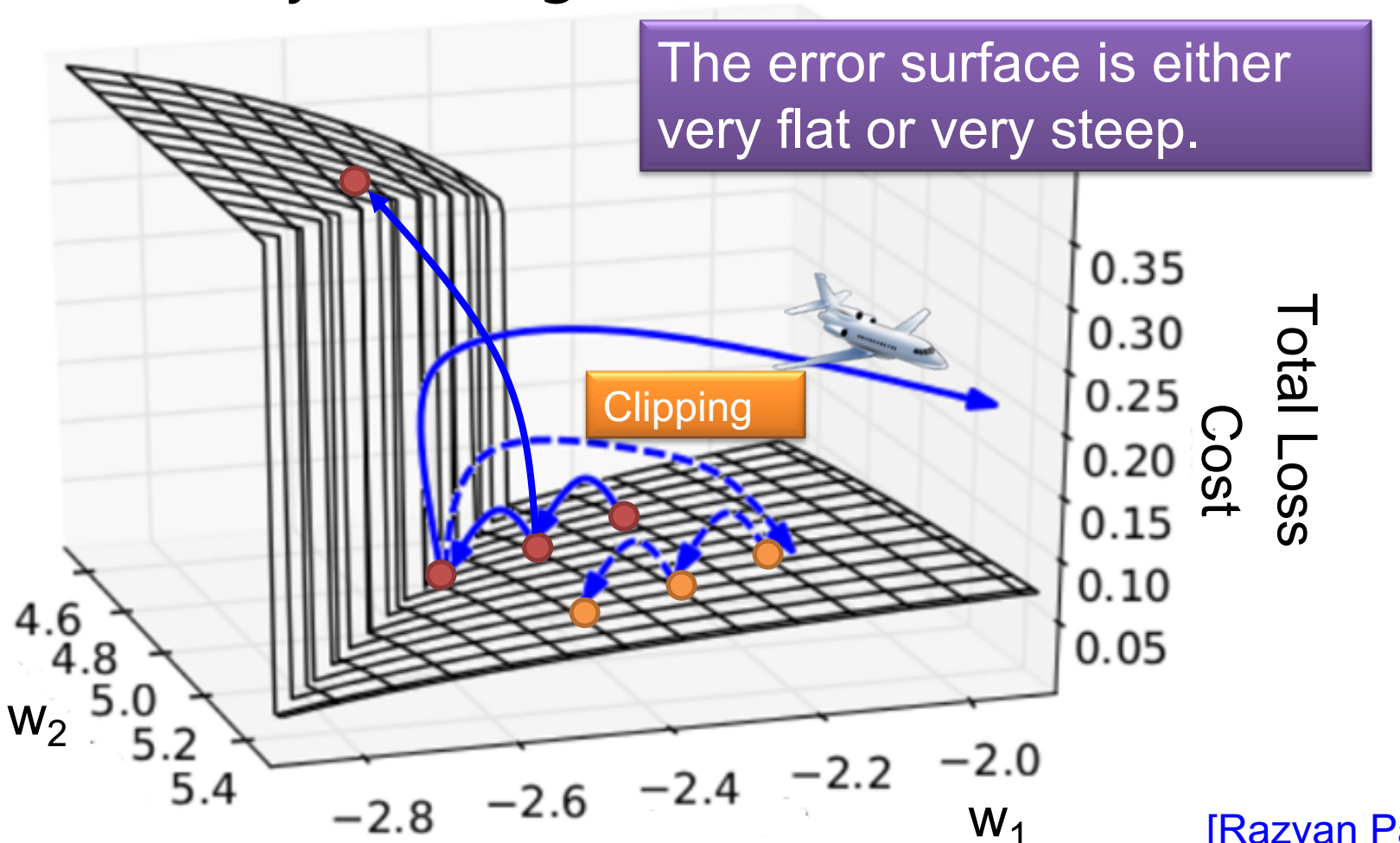
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W} \longrightarrow \frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \frac{\partial s_k}{\partial W}$$

Because the layers and time steps of deep neural networks relate to each other through multiplication, derivatives are susceptible to vanishing

Gradient contributions from “far away” steps become zero, and the state at those steps doesn’t contribute to what you are learning: You end up not learning long-range dependencies.

遞迴式神經網路(RNN) – 問題 (2/2)

- 梯度爆炸問題 - Why? - The gradient of the activation function is too lar



[Razvan Pascanu, ICML'13]

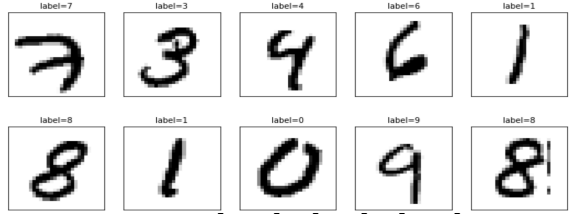
遞迴式神經網路(RNN) – Labs

- TensorFlow: Hello World!! – NN: HelloWorld_Tensorflow.ipynb
- TensorFlow 實現RNN簡單範例 – RNN.ipynb

Lab - TensorFlow: Hello World!!

HelloWorld_Tensorflow.ipynb

- 下載資料
- 查看訓練資料
- 查看多筆訓練資料images與label
- 批次讀取資料MNIST

1. mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
2. mnist.train.images[0]
3. Check 

4. batch_images_xs, batch_labels_ys = mnist.train.next_batch(batch_size=100)

Lab – RNN.ipynb (1/6)

- Step 1. 資料預處理
  - 導入模組並取得資料集

In [1]:
```
from __future__ import print_function

import tensorflow as tf
from tensorflow.contrib import rnn
```

In [2]:
```
# Import MNIST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

Lab – RNN.ipynb (2/6)

- Step 1. 資料預處理
  - 建立共用函數

In [3]:
```
# Training Parameters
learning_rate = 0.001
training_steps = 10000
batch_size = 128
display_step = 200
```

In [4]:
```
# Network Parameters
num_input = 28 # MNIST data input (img shape: 28*28)
timesteps = 28 # timesteps
num_hidden = 128 # hidden layer num of features
num_classes = 10 # MNIST total classes (0-9 digits)
```

In [5]:
```
# tf Graph input
X = tf.placeholder("float", [None, timesteps, num_input])
Y = tf.placeholder("float", [None, num_classes])

# Define weights
weights = {
    'out': tf.Variable(tf.random_normal([num_hidden, num_classes]))
}
biases = {
    'out': tf.Variable(tf.random_normal([num_classes]))
}
```

Lab – RNN.ipynb (3/6)

- Step 2. 建立模型
  - 定義input, output, loss and optimizer

```
def RNN(x, weights, biases):

    # Prepare data shape to match `rnn` function requirements
    # Current data input shape: (batch_size, timesteps, n_input)
    # Required shape: 'timesteps' tensors list of shape (batch_size, n_input)

    # Unstack to get a list of 'timesteps' tensors of shape (batch_size, n_input)
    x = tf.unstack(x, timesteps, 1)

    # Define a lstm cell with tensorflow
    lstm_cell = rnn.BasicLSTMCell(num_hidden, forget_bias=1.0)

    # Get lstm cell output
    outputs, states = rnn.static_rnn(lstm_cell, x, dtype=tf.float32)
```

```
# Define loss and optimizer
loss_op = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=logits, labels=Y))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
train_op = optimizer.minimize(loss_op)
```

Lab – RNN.ipynb (4/6)

- Step 3. 訓練模型

```
# Start training
with tf.Session() as sess:

    # Run the initializer
    sess.run(init)

    for step in range(1, training_steps+1):
        batch_x, batch_y = mnist.train.next_batch(batch_size)
        # Reshape data to get 28 seq of 28 elements
        batch_x = batch_x.reshape((batch_size, timesteps, num_input))
        # Run optimization op (backprop)
        sess.run(train_op, feed_dict={X: batch_x, Y: batch_y})
        if step % display_step == 0 or step == 1:
            # Calculate batch loss and accuracy
            loss, acc = sess.run([loss_op, accuracy], feed_dict={X: batch_x,
                                                                 Y: batch_y})
            print("Step " + str(step) + ", Minibatch Loss= " + \
                  "{:.4f}".format(loss) + ", Training Accuracy= " + \
                  "{:.3f}".format(acc))

    print("Optimization Finished!")
```

Lab – RNN.ipynb (5/6)

- Step 4. 評估模型準確率

```
# Evaluate model (with test logits, for dropout to be disabled)
correct_pred = tf.equal(tf.argmax(prediction, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
```

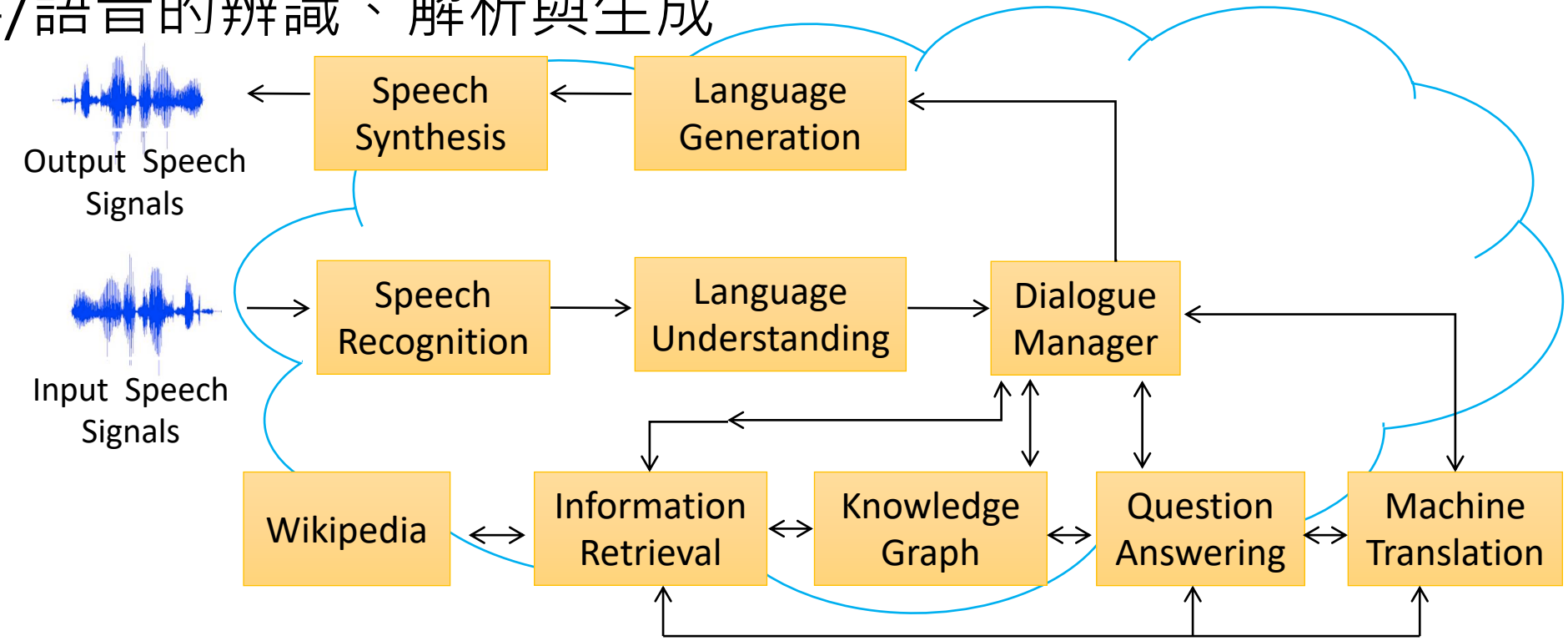
Lab – RNN.ipynb (6/6)

- Step 5. 進行預測

```
# Calculate accuracy for 128 mnist test images
test_len = 128
test_data = mnist.test.images[:test_len].reshape((-1, timesteps, num_input))
test_label = mnist.test.labels[:test_len]
print("Testing Accuracy:", \
    sess.run(accuracy, feed_dict={X: test_data, Y: test_label}))
```

RNN應用 – 自然語言處理(NLP)

- 『自然語言處理』(Natural Language Processing, NLP)，它包括文字/語音的辨識、解析與生成

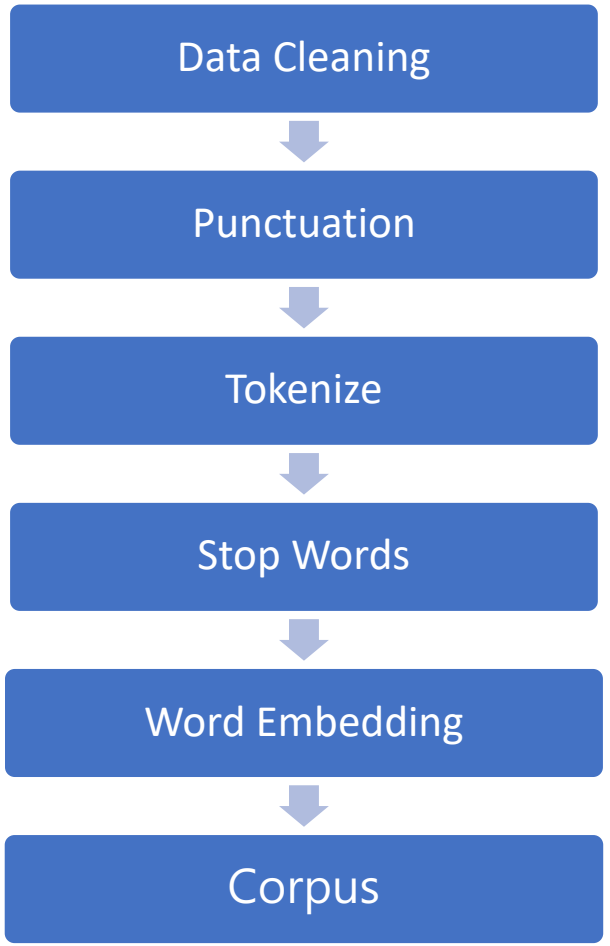


- **Examples 語音助理:**
  - Siri (Apple), Google Now (Google), Cortana (Microsoft), Alexa (Amazon)

RNN應用 – 自然語言處理(NLP) (1/3)

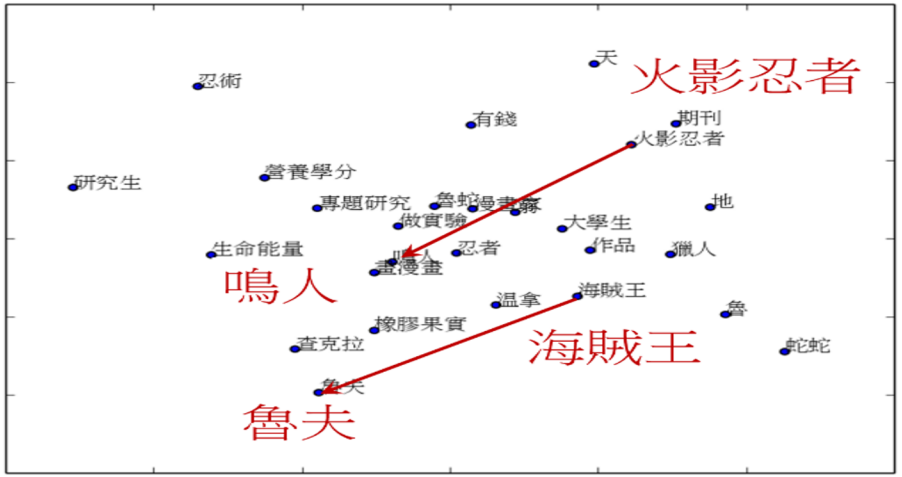
核心概念步驟(以英文為例)

1. 資料清理(Data Cleaning)：例如我們抓網頁資料，必須先清除 HTML 標籤(Tag)，取出乾淨的本文。
2. 標點符號(Punctuation)：對語意沒有影響，通常會忽略他們。
3. 分詞(Tokenize)：英文較容易，一般以空白即可，但中文就比較困難
4. Stop Words：例如英文的『the』、『as』、『to』、『from』...等介係詞或助詞，他們對語文大意的瞭解可能沒有太大的幫助，通常會忽略他們。



RNN應用 – 自然語言處理(NLP) (2/3)

5. 『詞嵌入』(Word Embedding)：
   - 要交給電腦計算，將單字轉數值會比較容易處理，『詞嵌入』技術通常會將單字轉為實數，以形成連續的向量空間，比較有名的模型包括 **Word2Vec** 及 GloVe。
   - 相似詞整理：
     - 意義相近的單字或片語，在解析字句及分類時必須能呈現出來，Google Tomas Mikolov 創造 Word2Vec 模型，以向量(Vector)空間來定義單字(Word)，就如之前介紹的『照片比對』計算Cosine，如接近1，就表示兩單字的意義相似。



RNN應用 – 自然語言處理(NLP) (3/3)

6. 『語料庫』(Corpus)：
   - 要能訓練模型，必須要有大量的標註資料，NLTK (Natural Language Toolkit)工具箱同時提供完整的函數庫及大量的語料庫

語料庫下載

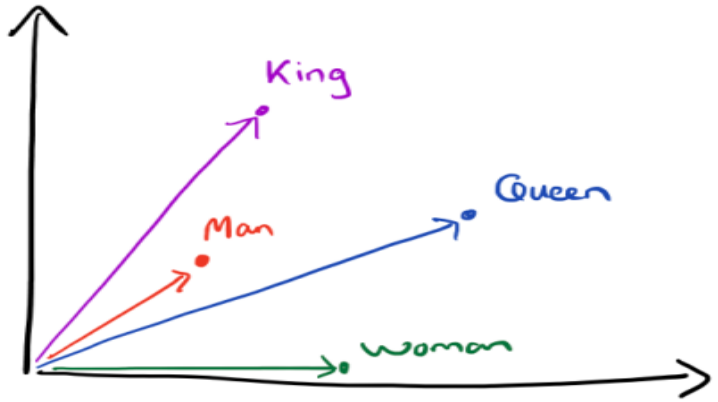
```
# pip install -U nltk
```

遞迴式神經網路(RNN) – Labs

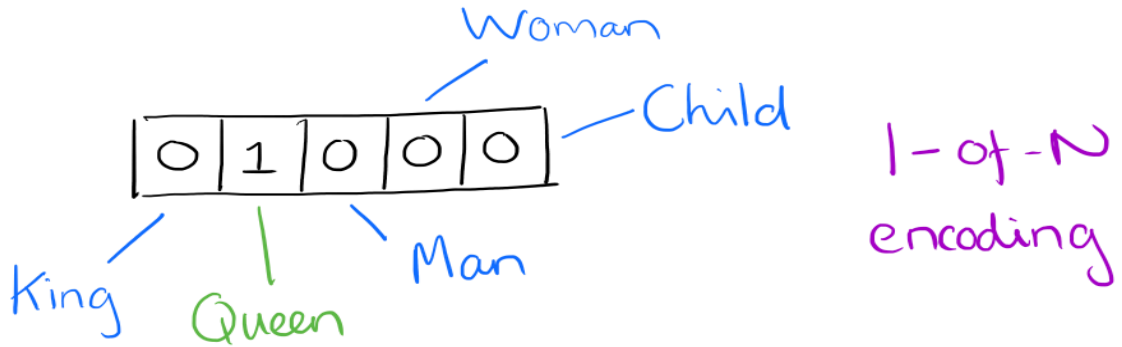
- TensorFlow 實現Word2Vec
  - 英文word2vec.ipynb

Word2Vec (1/8)

- 自然語言處理入門- Word2Vec
- Word2Vec簡介
  - 根據輸入的『詞的集合』計算出詞與詞之間的距離
  - 『字詞』轉換成『向量』形式，可以把對文本內容的處理簡化為向量空間中的向量運算，計算出向量空間上的相似度，來表示文本語義上的相似度。



Word Vectors



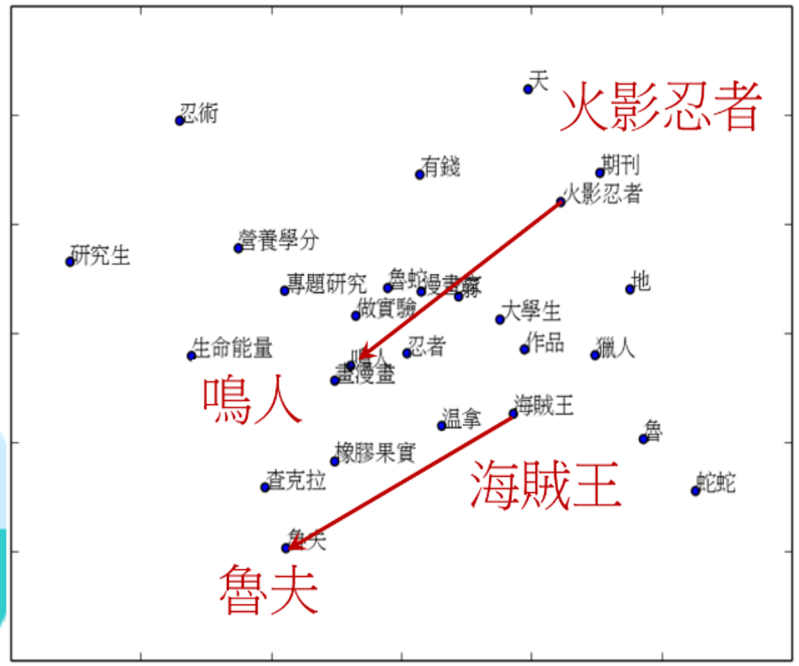
Word2Vec (2/8)

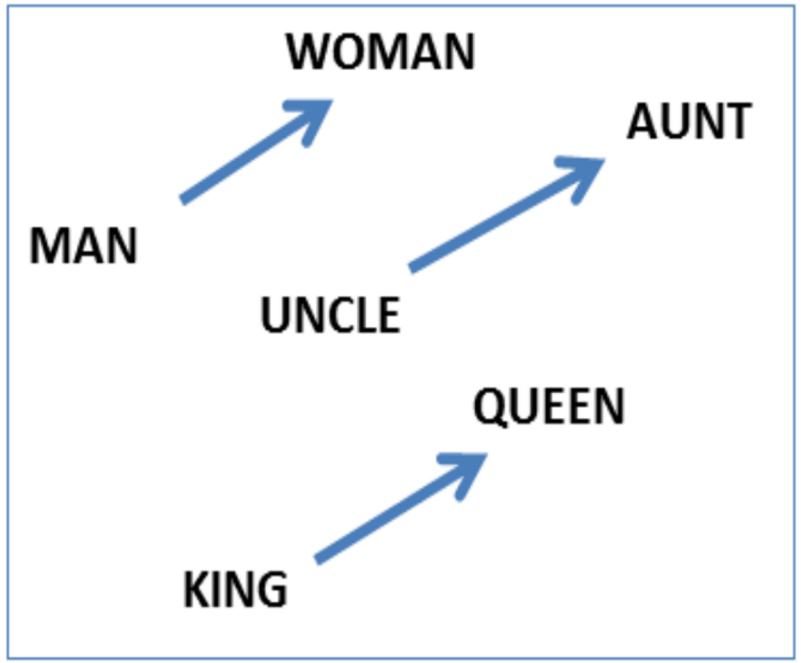
- Word2Vec簡介
  - word2vec 計算的是餘弦值 (cosine)，距離範圍為 0–1 之間，值越大代表兩個詞關聯度越高。
  - 詞向量：用 Distributed Representation 表示詞，通常也被稱為「Word Representation」或「Word Embedding」。

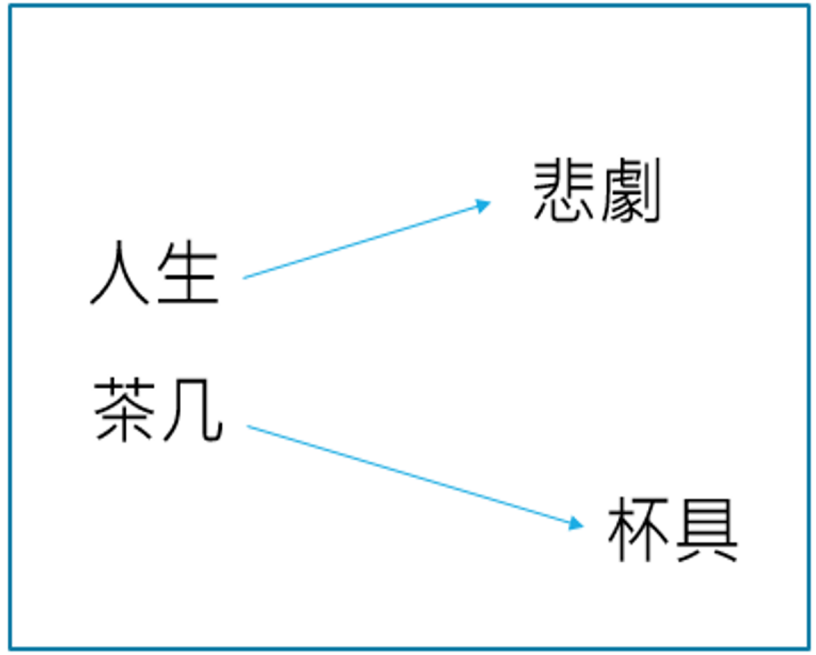
  =>詞向量表示法讓相關或者**相似的詞**，在距離上**更接近**
  - Word2Vec的形式和Autoencoder (參考附錄)有點像，一樣是從高維度的空間轉換到低維度的空間，再轉換回去原本的維度，只是這一次轉回去的東西不再是原本一模一樣的東西了
    - **Eg:** “by the way”這個用法如果多次被機器看過的話，此時"by"與"the"和"way"便會產生一個上下文的關聯性

Word2Vec (3/8)

- Word2Vec簡介







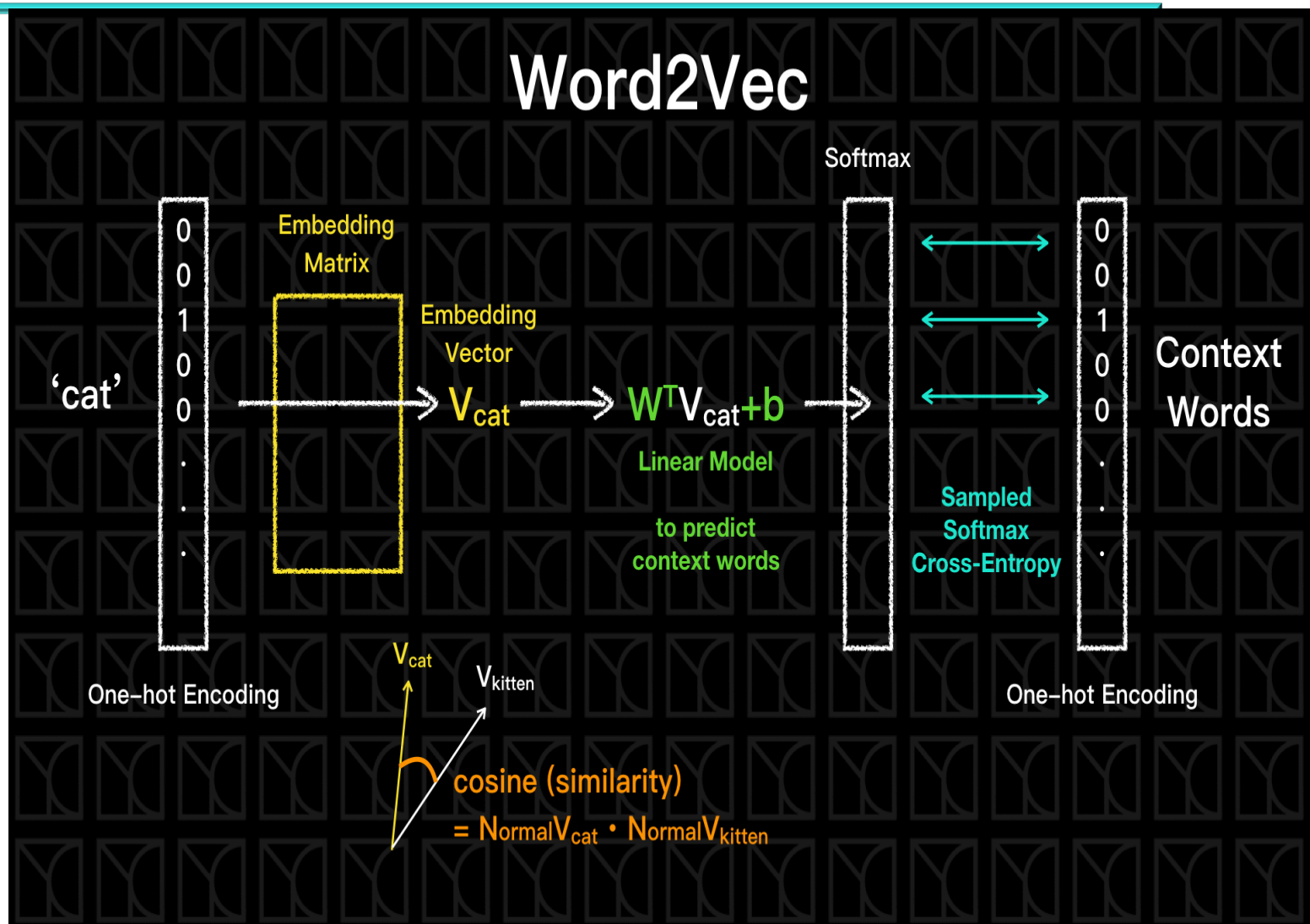
人生像茶几 擺滿了杯具 (悲劇)

vec(woman) - vec(man) ≈ vec(queen) - vec(king)

人生=[0.5 0.7 -0.8 0.3 -0.4]

Word2Vec (4/8)

- Word2Vec架構
  - Word2Vec所有的轉換都是**線性**的
  - 中間的Hidden Layer被稱為Embedding Matrix，它做了一個線性的Dimension Reduction
  - 將原本高維度的One-hot encoding降成低維度，然後再透過一個線性模型轉換回去原本的維度

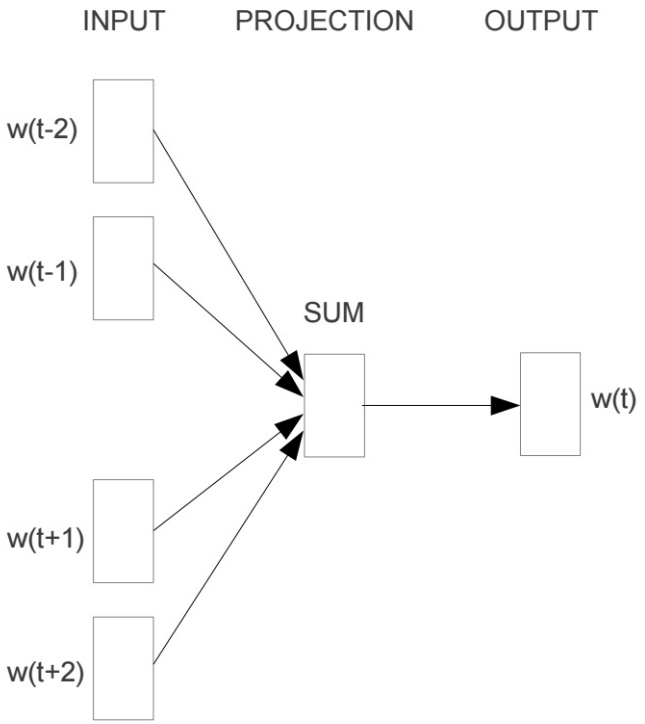


Word2Vec (5/8)

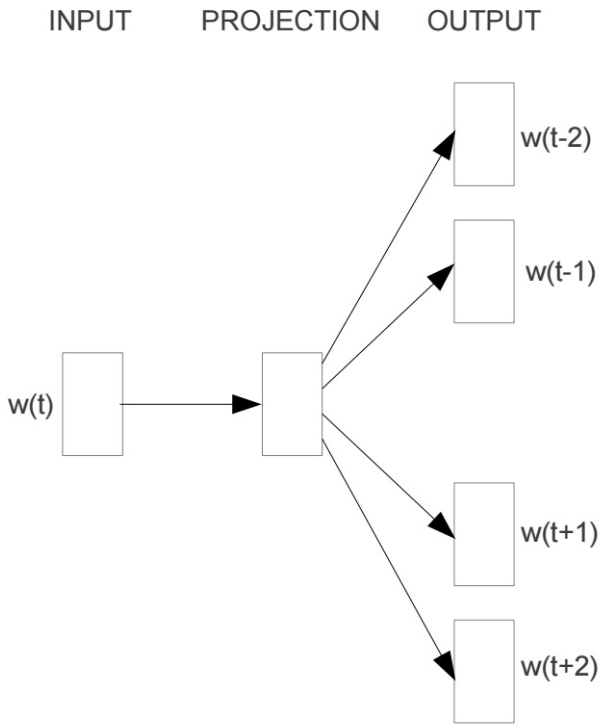
- Word2Vec架構
  - 兩個向量的相似性可以使用Cosine來評估，當兩向量的夾角越小代表它們越是相似
  - 利用Cosine來建立Similarity的大小，藉此來找到前幾個和它很靠近的詞彙。
  - 例如：羽毛球之於戴資穎，相似於排球之於劉鴻敏，這樣的比較關係也顯示在這個Embedding空間裡頭，所以在這空間裡會有以下的向量關係式：V戴資穎$-V$羽毛球$+V$排球$=V$劉鴻敏

Word2Vec (6/8)

- Word2Vec的兩種常用方法：Skip-Gram and CBOW (Continuous Bag of Words)



CBOW



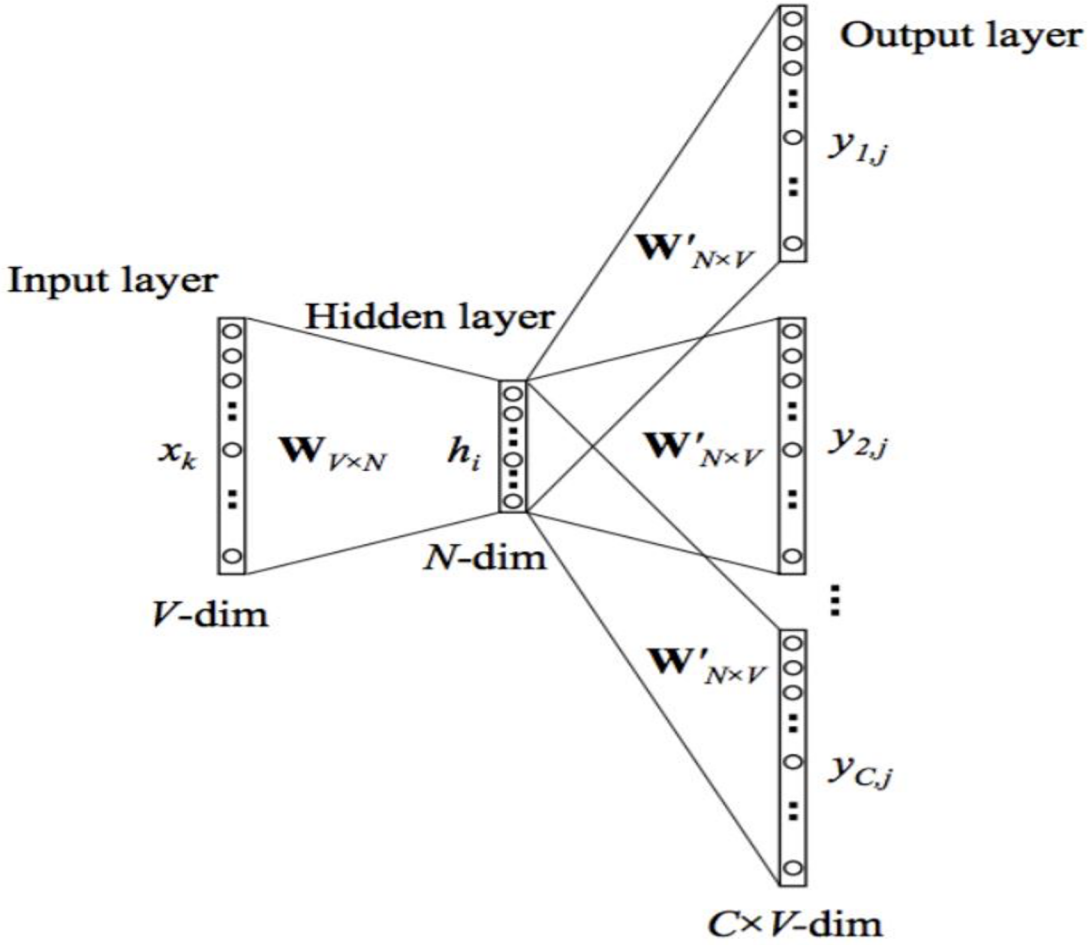
Skip-gram

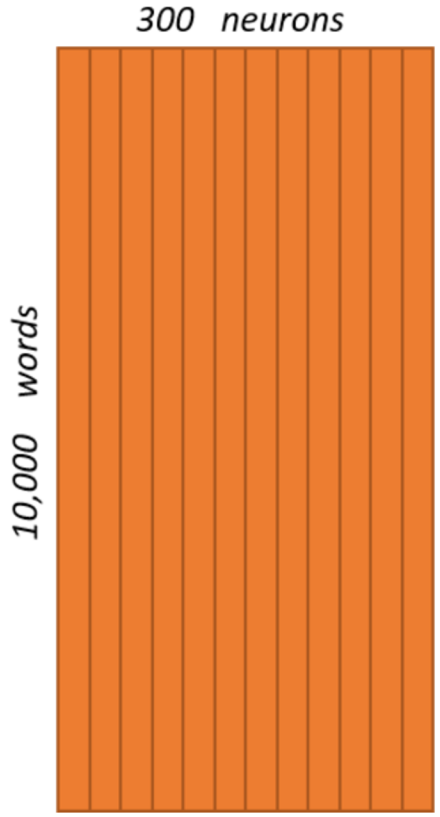
Word2Vec (7/8)

- Skip-Gram
  - **輸入一個$word(t)$時**，希望它能輸出它的前文和後文
  - 就會有兩組數據：$(w(t),w(t-1))$ 和$(w(t),w(t+1))$
- CBOW (Continuous Bag of Words)
  - **填空題**：將一排字挖掉中間一個字，然後希望由上下文的關係有辦法猜出中間那個字
  - 此時輸入層就變成會有多於1個字：轉換到Embedding空間後再相加平均，因為是線性轉換，所以直接線性累加就可以了

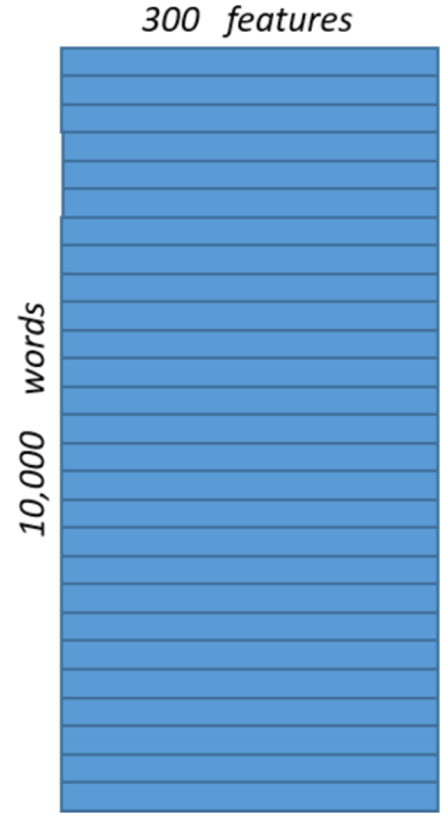
Word2Vec (8/8)

- Skip-Gram





W



W’

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

W

Labs – 1. 英文word2vec.ipynb (1/6)

- Word2Vec實作步驟
  - Step 1. 下載資料集
  - Step 2. 構建字典並用UNK Token替換罕見的單詞
  - Step 3. 使用skip-gram模型生成訓練，生成 Word2Vec 訓練樣本
  - Step 4. 構建並訓練skip-gram模型
  - Step 5. 開始訓練
  - Step 6. 視覺化Embeddings

Labs – 1. 英文word2vec.ipynb (2/6)

- Step 1. 下載資料集
  - 準備文本語料庫
  - 下載Dataset，並做一些前處理

In [5]:
```
from urllib.request import urlretrieve
def maybe_download(filename, expected_bytes):
    if not os.path.exists(filename):
        url =  "http://mattmahoney.net/dc/"
        filename, _ = urlretrieve(url + filename, filename)
    statinfo = os.stat(filename)
    if statinfo.st_size == expected_bytes:
        print('Found and verified', filename)
    else:
        print(statinfo.st_size)
        raise Exception('Failed to verify ' + filename + '. Can you get to it with a browser?')
    return filename

filename = maybe_download('text8.zip', 31344016)
```

```
Found and verified text8.zip
```

Labs – 1. 英文word2vec.ipynb (3/6)

- Step 2. 構建字典並用UNK Token替換罕見的單詞

Build the dictionary and replace rare words with UNK token

```
# 將出现频率最高的 50000 個單詞放入 count 列表中，然後放入 dictionary 中
# 前面是詞彙，最後是出现的次数，這裡的 -1 在下面會填上 UNK 出现的頻率數
# 將出现頻率最高的 50000 個詞存入count
# Encoding：如果不出现在 dictionary 中，就以 0 作為編號，否則以 dictionary 中的編號為主
# 也就是將 words 中的所有詞的編號存在 data 中，並查一下 UNK 有多少，以便替換 count 中的 -1
# 編號：詞
```

Labs – 1. 英文word2vec.ipynb (4/6)

- Step 3. 使用skip-gram模型生成訓練，生成 Word2Vec 訓練樣本

In [4]:
```
# Step 3: Function to generate a training batch for the skip-gram model.
def generate_batch(batch_size, num_skips, skip_window):
  global data_index
  assert batch_size % num_skips == 0
  assert num_skips <= 2 * skip_window
  batch = np.ndarray(shape=(batch_size), dtype=np.int32)
  labels = np.ndarray(shape=(batch_size, 1), dtype=np.int32)
  span = 2 * skip_window + 1 # [ skip_window target skip_window ]
  buffer = collections.deque(maxlen=span)
  for _ in range(span):
    buffer.append(data[data_index])
    data_index = (data_index + 1) % len(data)
  for i in range(batch_size // num_skips):
    target = skip_window  # target label at the center of the buffer
    targets_to_avoid = [ skip_window ]
    for j in range(num_skips):
      while target in targets_to_avoid:
        target = random.randint(0, span - 1)
      targets_to_avoid.append(target)
      batch[i * num_skips + j] = buffer[skip_window]
      labels[i * num_skips + j, 0] = buffer[target]
    buffer.append(data[data_index])
    data_index = (data_index + 1) % len(data)
  return batch, labels

batch, labels = generate_batch(batch_size=8, num_skips=2, skip_window=1)
for i in range(8):
  print(batch[i], reverse_dictionary[batch[i]],
      '->', labels[i, 0], reverse_dictionary[labels[i, 0]])
```

```
3081 originated -> 12 as
3081 originated -> 5234 anarchism
12 as -> 6 a
12 as -> 3081 originated
6 a -> 12 as
6 a -> 195 term
195 term -> 2 of
195 term -> 6 a
```

Labs – 1. 英文word2vec.ipynb (5/6)

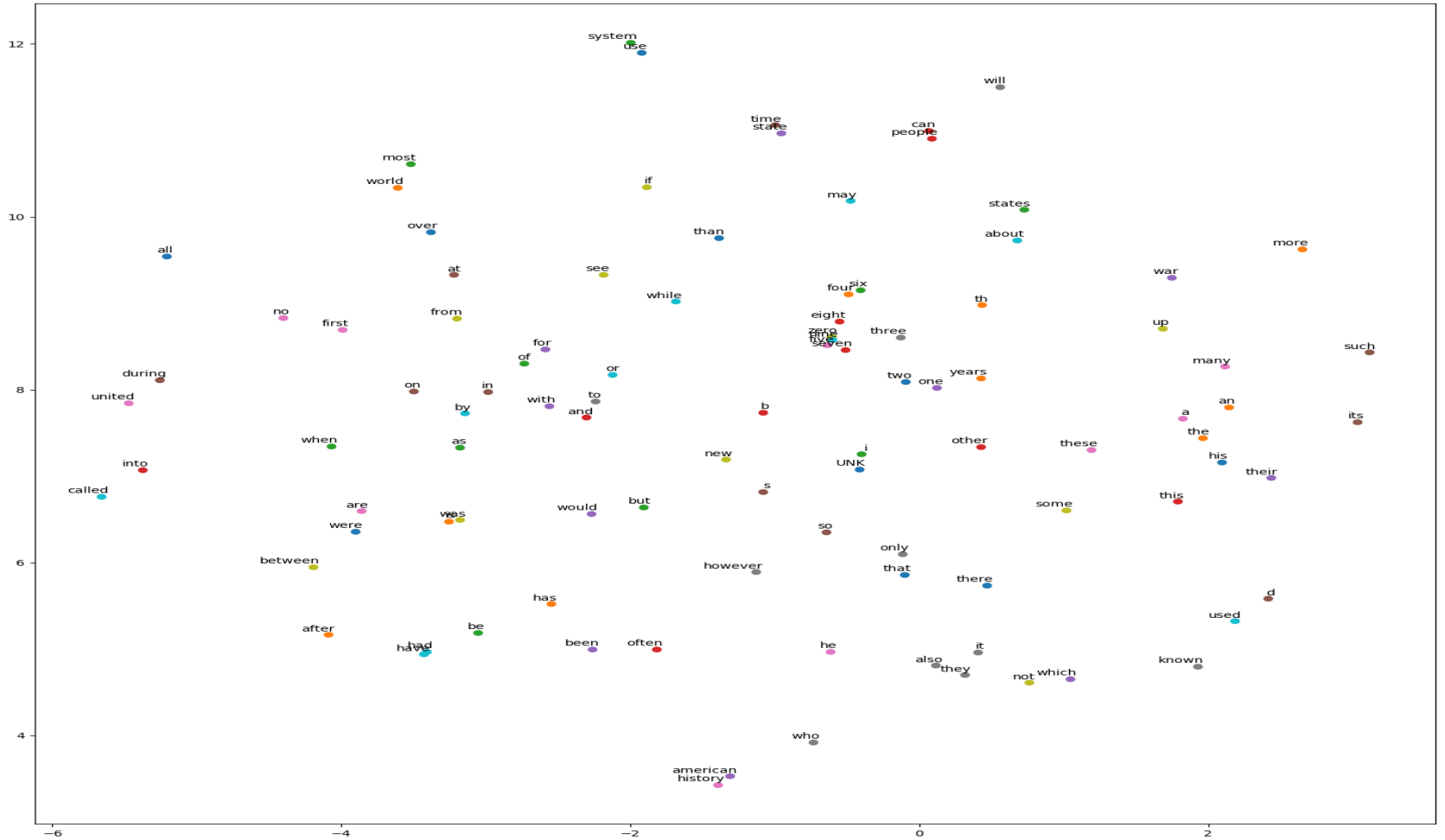
- Step 4. 構建並訓練skip-gram模型

Build and train a skip-gram model

```
# 訓練需要的參數
# 生成驗證數據，隨機抽取一些頻數最高的單詞，看向量空間上跟它們距離最近的單詞是否相關性比較高
```

Labs – 1. 英文word2vec.ipynb (6/6)

- Step 5. 開始訓練
- Step 6. 視覺化 Embeddings



Word2Vec – 中文

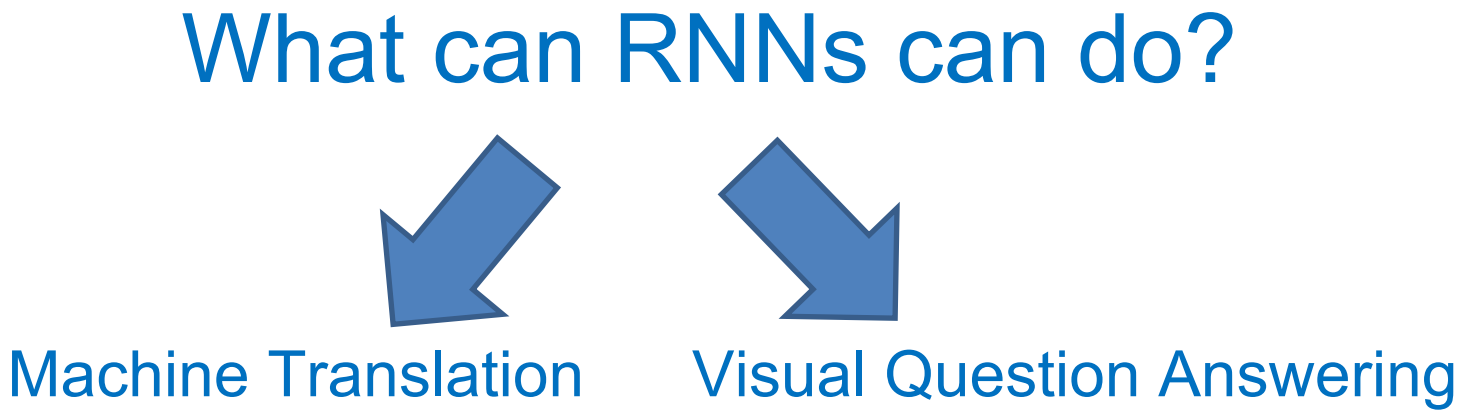
- Word2Vec實作步驟
  - Step 1. 在 terminal 進行套件安裝以及資料格式檔案轉換
  - Step 2. 在 jupyter notebook 透過 jieba 套件斷詞，再經由 word2vec 萃取文章特徵，轉換成以向量空間表示的詞向量 (**繁體中文**)
  - Step 3. 視覺化呈現「相近詞」向量之結果

Long Short-Term Memory (LSTM) - 簡介

- LSTM是現今RNN的主流
- LSTM是迄今為止這系列課程當中看過最複雜的Neural Network
- 可以解決梯度消失的問題
  - 使用ReLU代替tanh或sigmoid (參考**附錄 – 激活函數**)
  - 激活函數ReLU導數是0或1的常數，因此它不太可能遭受消失的梯度
  - LSTM有助於保留可以通過時間和層反向傳播的錯誤
  - 通過保持更加constant error，它們允許經常性的網絡繼續學習多個時間步長（超過1000）

Long Short-Term Memory (LSTM) - 應用

- 語音辨識
- 語言模型
- 自然語言處理
- 機器翻譯
- 時間序列分析



兒童界披頭四

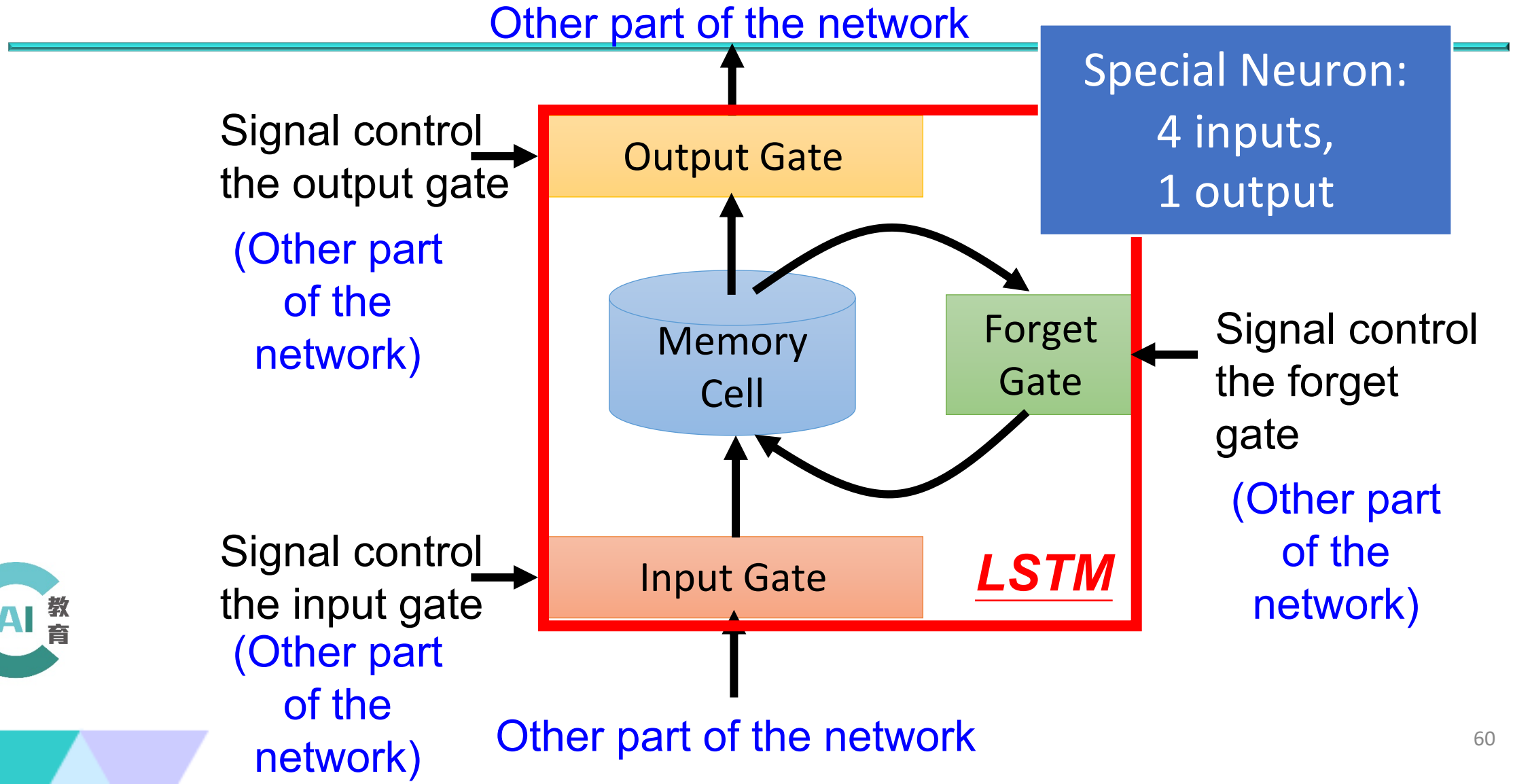
小波站在拉拉旁邊
拉拉站在迪西旁邊
迪西站在丁丁旁邊
●●●

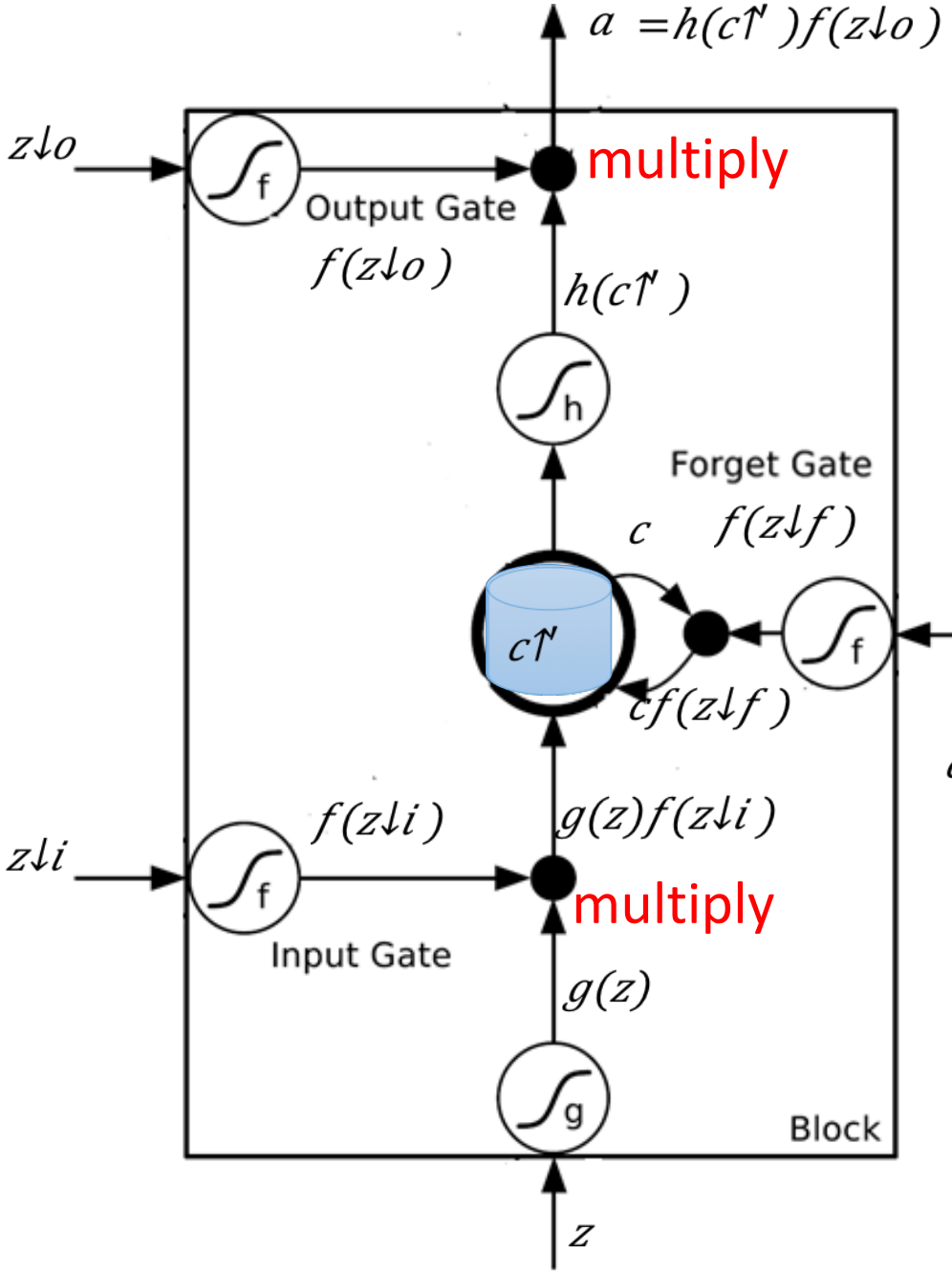
詞彙：
「小波」、「拉拉」、
「迪西」、「丁丁」、
「站在」、「旁邊」

兒童界披頭四

- RNN可能會發生的錯誤：
  - 小波站在拉拉站在迪西.....
  - 丁丁站在丁丁旁邊

Long Short-Term Memory (LSTM) - 原理 (1/6)



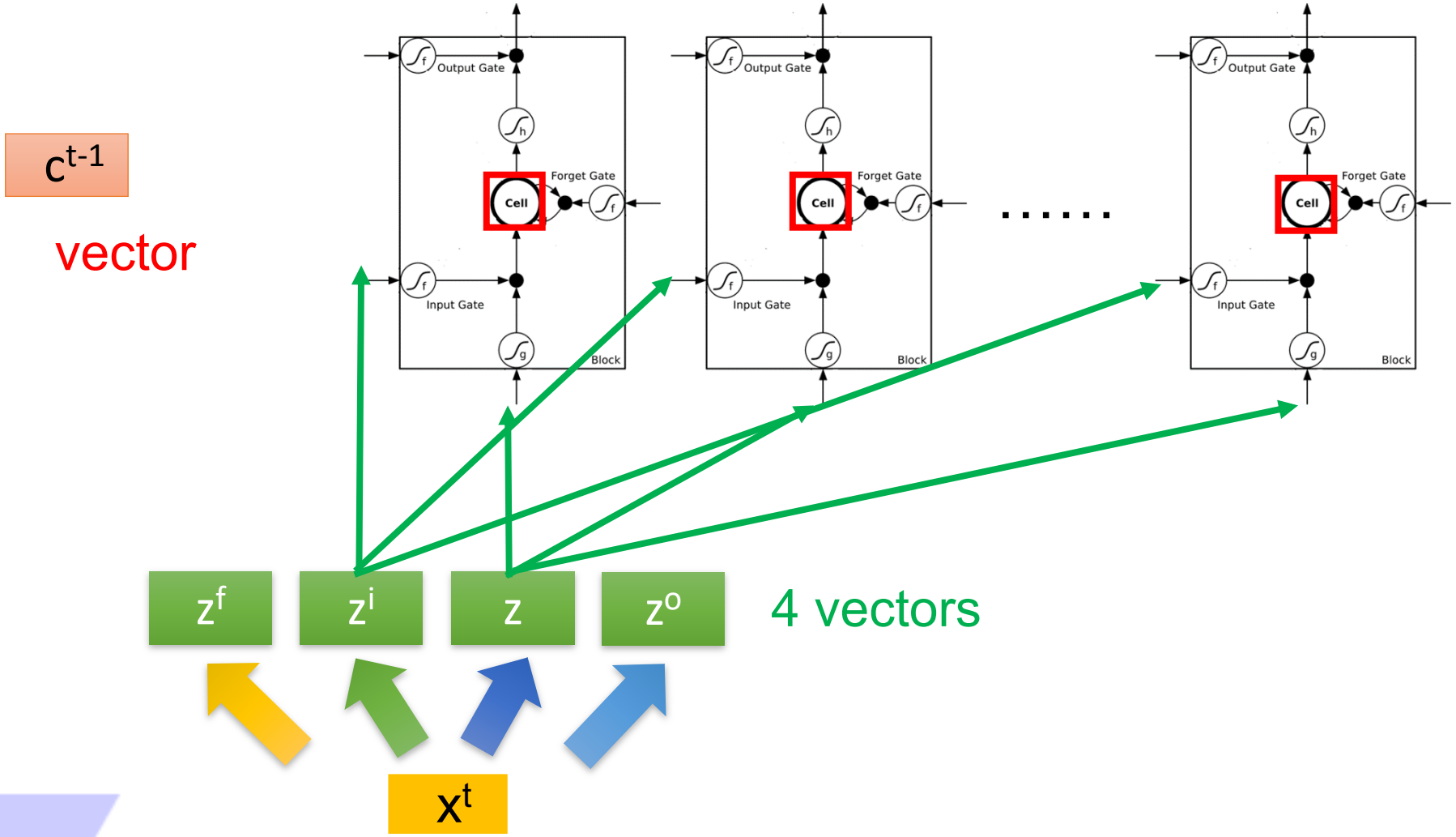


Activation function f is usually a sigmoid function

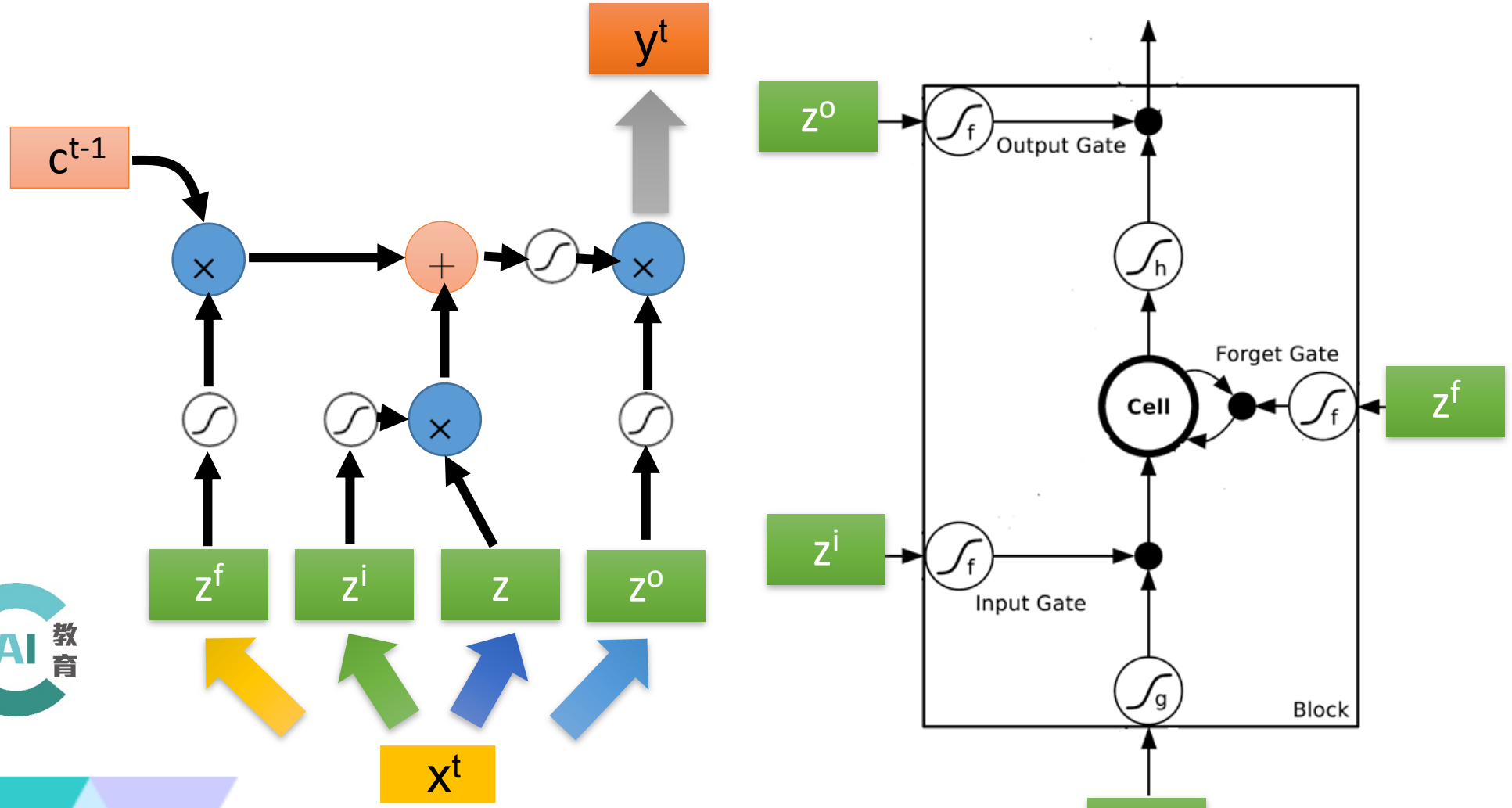
Between 0 and 1

Mimic open and close gate

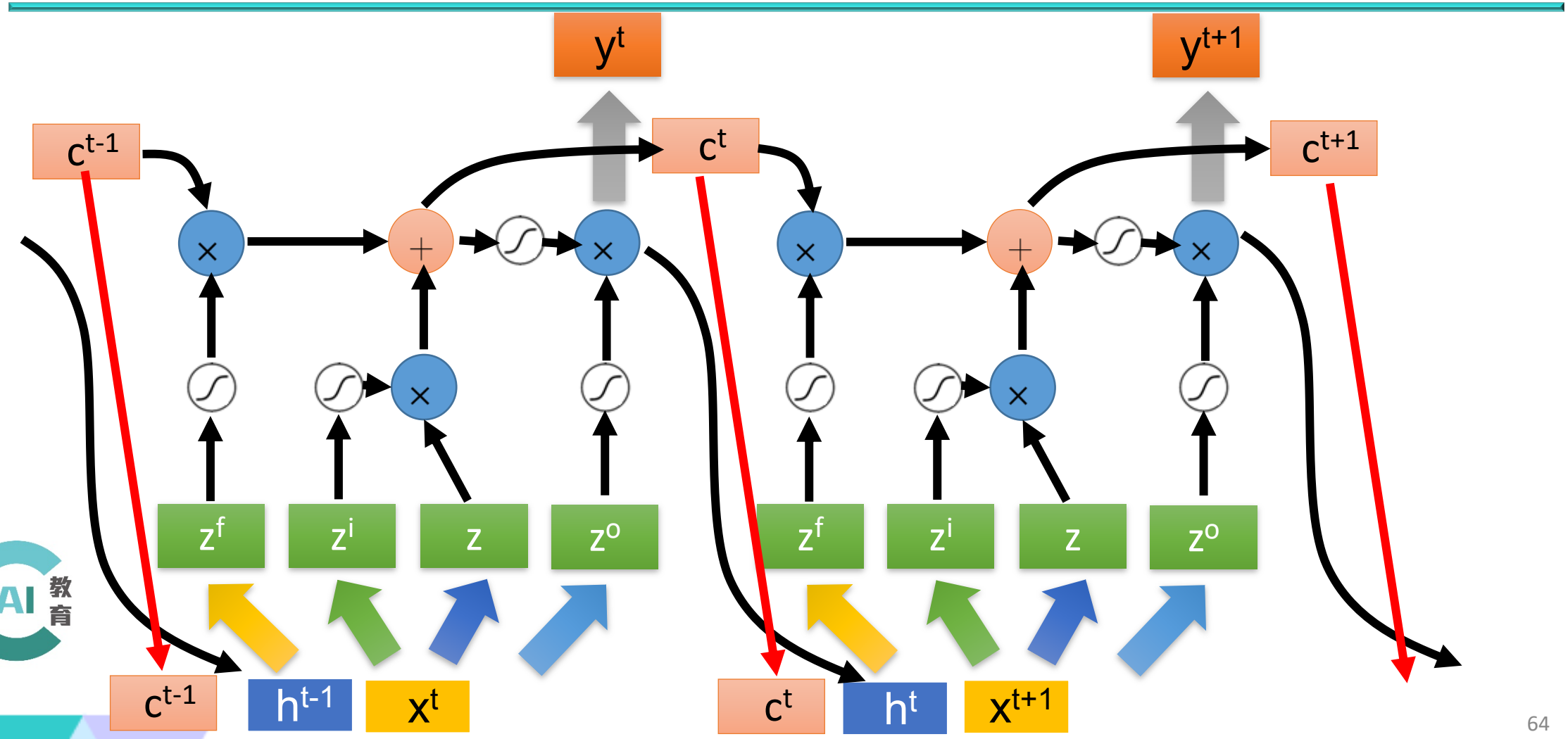
Long Short-Term Memory (LSTM) - 原理 (3/6)



Long Short-Term Memory (LSTM) - 原理 (4/6)

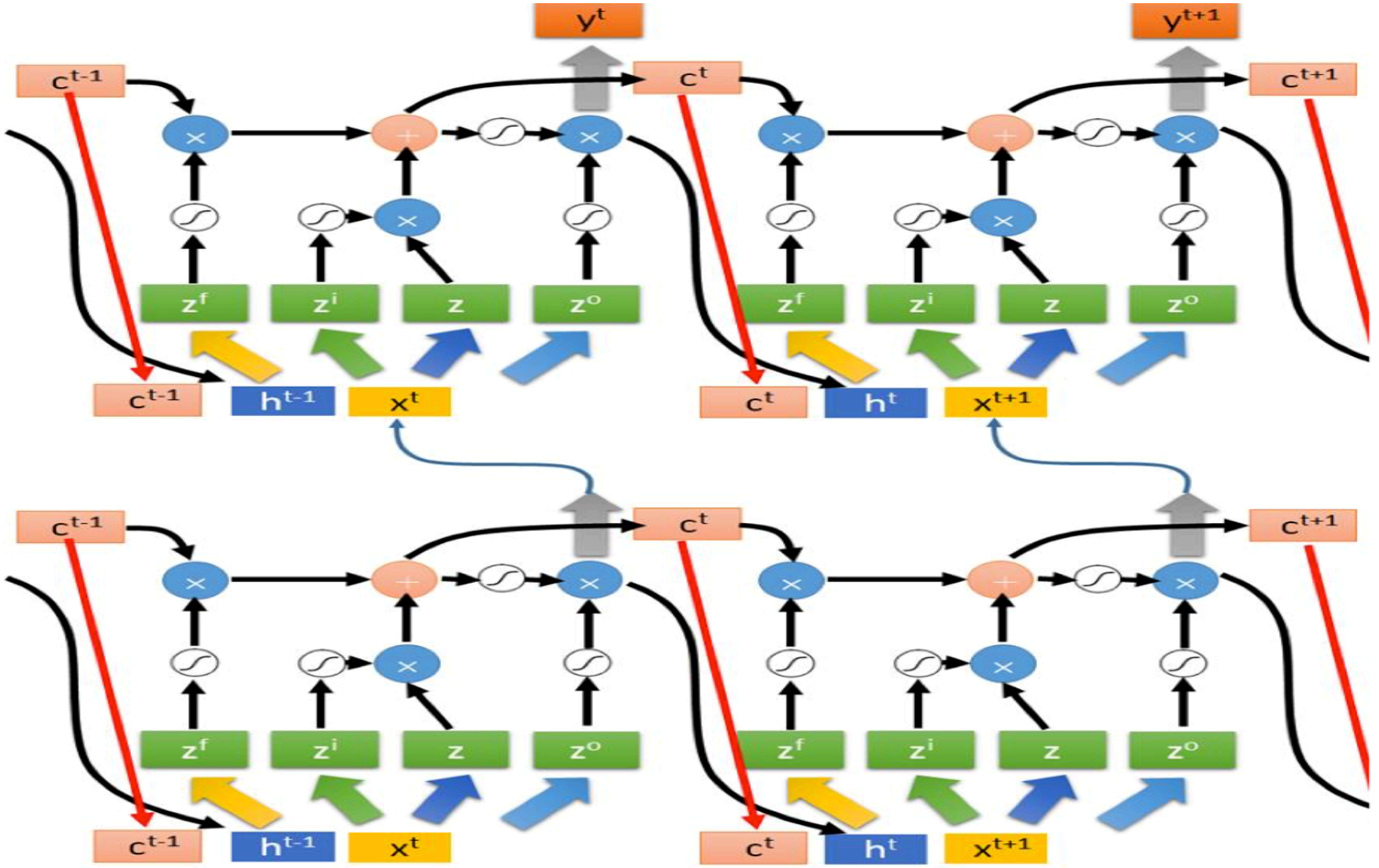


Long Short-Term Memory (LSTM) - 原理 (5/6)



Long Short-Term Memory (LSTM) - 原理 (6/6)

Multiple-layer LSTM



https://img.komicolle.org/2015-09-20/src/1442696762713l.gif

Long Short-Term Memory (LSTM) – Labs

- 實現 LSTM 進行IMDb 情緒分析
  - Sentiment_Analysis_IMDb_Introduce.ipynb
  - Sentiment_Analysis_IMDb_LSTM.ipynb

- 使用LSTM實作文章產生器
  - LSTM_Gen.ipynb

實現 LSTM 進行IMDb 情緒分析

- IMDB 資料集：
  - http://ai.stanford.edu/~amaas/data/sentiment/
  - 共有50000筆『影評文字』，其中 25000 個訓練樣本和 25000 個測試樣本
- 目標模型：
  - 經過大量『影評文字』訓練後，可用來預測影評文字為『正面評價』與『負面評價』
- 自然語言處理步驟

NLP – Logical Flow

IMDb 資料集

Step 1. 讀取IMDb資料集

25000筆 訓練資料 影評文字
The Chases were very good……

Step 2. 建立Token

Token (字典共2000字)
['the', 'and', 'full', 'involving', 'to',, ..]

Step 3. 使用token時將『影評文字』轉換為『數字list』

25000筆 『數字list』21, 187, 371, 44, 26 …

Step 4. 截長補短讓所有『數字list』長度都是100

25000筆 『數字list』(長度100)

Step 5. Embedding 層將『數字list』轉換成『向量list』

25000筆 32維度『向量list』(長度100)

Step 6. 將『向量list』送進深度學習模型，進行訓練

1. MLP, 2. RNN, 3. LSTM, 4. CNN

Lab – Sentiment_Analysis_IMDb_Introduce.ipynb (1/6)

- 前處理

In [2]:
```
import urllib.request
import os
import tarfile
```

In [4]:
```
url="http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz"
filepath="data/aclImdb_v1.tar.gz"
if not os.path.isfile(filepath):
    result=urllib.request.urlretrieve(url,filepath)
    print('downloaded:',result)
```

```
downloaded: ('data/aclImdb_v1.tar.gz', <http.client.HTTPMessage object at 0x10c1c7160>)
```

In [*]:
```
if not os.path.exists("data/aclImdb"):
    tfile = tarfile.open("data/aclImdb_v1.tar.gz", 'r:gz')
    result=tfile.extractall('data/')
```

1. Import Library ¶

In [2]:
```
from keras.datasets import imdb
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
```

```
Using TensorFlow backend.
```

Lab – Sentiment_Analysis_IMDb_Introduce.ipynb (2/6)

- Step 1. 讀取IMDb資料集
  - 分為訓練資料與測試資料

訓練資料

train_text (文字)：0至12499筆 – 正面評價影評文字
12500至24999筆 – 負面評價影評文字
y_train (標籤)：0至12499筆 – 正面評價，全部是1
12500至24999筆 – 負面評價，全部是0

測試資料

test_text (文字)：0至12499筆 – 正面評價影評文字
12500至24999筆 – 負面評價影評文字
y_test (標籤)：0至12499筆 – 正面評價，全部是1
12500至24999筆 – 負面評價，全部是0

Lab – Sentiment_Analysis_IMDb_Introduce.ipynb (3/6)

- Step 2. 建立token
  - 目的：將『影評文字』轉換『數字list』
  - 建立token筆需指定字典的字數，如2000個字的字典
  - 讀取訓練資料25000筆，依照每個英文單字出現的頻率進行排序，僅有出現頻率前2000名的英文字，列入字典

```
{'the': 1, 'and': 2, 'a': 3, 'of': 4, 'to': 5, 'is': 6, 'in': 7, 'it': 8, 'i': 9, 'this': 10, 'that': 11, 'w
as': 12, 'as': 13, 'for': 14, 'with': 15, 'movie': 16, 'but': 17, 'film': 18, 'on': 19, 'not': 20, 'you': 2
1, 'are': 22, 'his': 23, 'have': 24, 'be': 25, 'he': 26, 'one': 27, 'all': 28, 'at': 29, 'by': 30, 'an': 31,
'they': 32, 'who': 33, 'so': 34, 'from': 35, 'like': 36, 'her': 37, 'or': 38, 'just': 39, 'about': 40, "i
t's": 41, 'out': 42, 'has': 43, 'if': 44, 'some': 45, 'there': 46, 'what': 47, 'good': 48, 'more': 49, 'whe
n': 50, 'very': 51, 'up': 52, 'no': 53, 'time': 54, 'she': 55, 'even': 56, 'my': 57, 'would': 58, 'which': 5
9, 'only': 60, 'story': 61, 'really': 62, 'see': 63, 'their': 64, 'had': 65, 'can': 66, 'were': 67, 'me': 6
8, 'well': 69, 'than': 70, 'we': 71, 'much': 72, 'been': 73, 'get': 74, 'bad': 75, 'will': 76, 'also': 77,
'do': 78, 'into': 79, 'people': 80, 'other': 81, 'first': 82, 'great': 83, 'because': 84, 'how': 85, 'him':
86, 'most': 87, "don't": 88, 'made': 89, 'its': 90, 'then': 91, 'way': 92, 'make': 93, 'them': 94, 'too': 9
5, 'could': 96, 'any': 97, 'movies': 98, 'after': 99, 'think': 100, 'characters': 101, 'watch': 102, 'two':
103, 'films': 104, 'character': 105, 'seen': 106, 'many': 107, 'being': 108, 'life': 109, 'plot': 110, 'neve
r': 111, 'acting': 112, 'little': 113, 'best': 114, 'love': 115, 'over': 116, 'where': 117, 'did': 118, 'sho
```

Lab – Sentiment_Analysis_IMDb_Introduce.ipynb (4/6)

- Step 3. 使用token將『影評文字』轉換『數字list』
  - 因為建立token了，所以可以將『影評文字』轉換『數次list』

```
In [24]: print(train_text[0])
```

```
For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a mov
ie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an abs
olute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt.
```

```
In [25]: print(x_train_seq[0])
```

```
[14, 3, 16, 11, 210, 53, 1157, 46, 248, 22, 3, 172, 4, 902, 14, 10, 1524, 833, 3, 16, 117, 912, 6, 161, 158,
6, 3, 132, 1, 105, 6, 31, 1551, 102, 14, 1604, 1, 1787, 13, 3, 564]
```

Lab – Sentiment_Analysis_IMDb_Introduce.ipynb (5/6)

- Step 4.截長補短讓所有『數字list』長度都是100
  - 每一段文字都不固定，但是要轉換成『向量list』的**預先動作**

```
In [30]: print('before pad_sequences length=',len(x_train_seq[0]))
         print(x_train_seq[0])
```

```
before pad_sequences length= 41
[14, 3, 16, 11, 210, 53, 1157, 46, 248, 22, 3, 172, 4, 902, 14, 10, 1524, 833, 3, 16, 117, 912, 6, 161, 158,
6, 3, 132, 1, 105, 6, 31, 1551, 102, 14, 1604, 1, 1787, 13, 3, 564]
```

```
In [31]: print('after pad_sequences length=',len(x_train[0]))
         print(x_train[0])
```

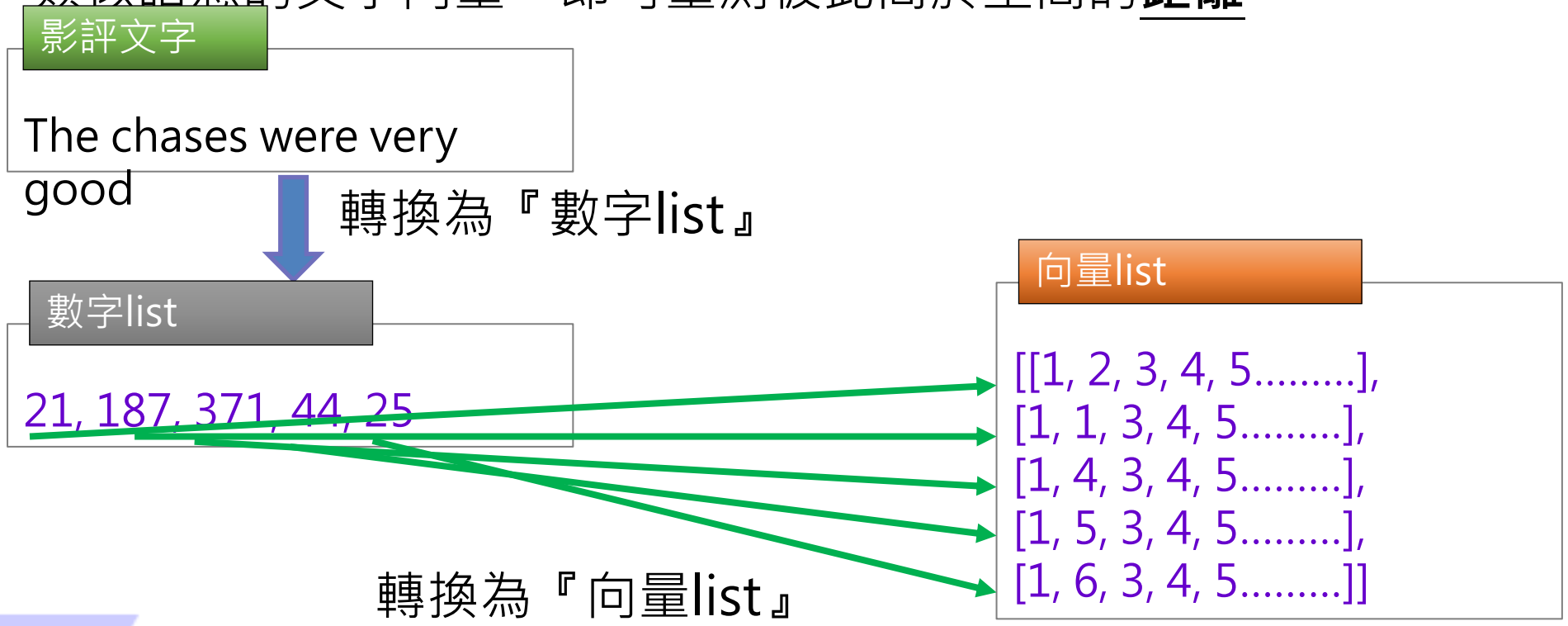
```
after pad_sequences length= 100
[   0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0   14    3   16   11  210   53 1157   46  248   22    3
  172    4  902   14   10 1524  833    3   16  117  912    6  161  158
    6    3  132    1  105    6   31 1551  102   14 1604    1 1787   13
    3  564]
```

補足長度

補上59個0
若長度是130呢？

Lab – Sentiment_Analysis_IMDb_Introduce.ipynb (6/6)

- Step 5. Embedding 層將『數字list』轉換成『向量list』
  - 使用Word Embedding，原理：將文字映射成多維幾何空間的向量
  - 類似語意的文字向量，即可量測彼此間於空間的**距離**

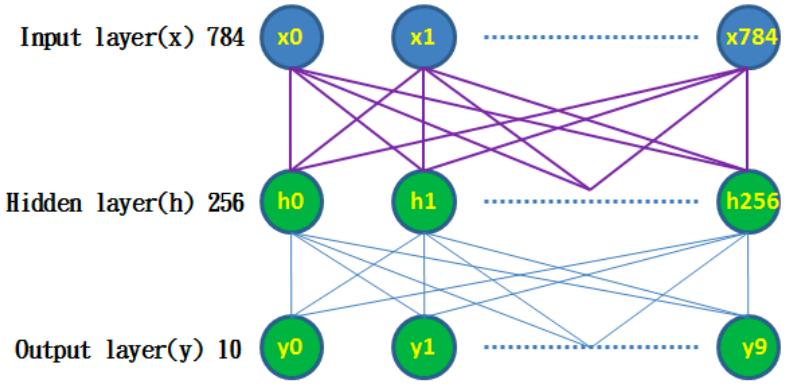


Lab – Sentiment_Analysis_IMDb_LSTM.ipynb (1/5)

IMDb 資料集

25000筆 訓練資料 影評文字預處理
Step1. 讀取IMDb資料集
Step 2. 建立Token
Step 3. 使用token時將『影評文字』轉換為『數字list』
Step 4. 截長補短讓所有『數字list』長度都是100
Step 5. Embedding 層將『數字list』轉換成『向量list』

LSTM (units = 32)



將『向量list』送進深度學習模型，進行訓練

Lab – Sentiment_Analysis_IMDb_LSTM.ipynb (2/5)

- Step 1. 建立模型

In [12]:
```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation,Flatten
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import LSTM
```

In [13]:
```
model = Sequential()
```

In [14]:
```
model.add(Embedding(output_dim=32,
                    input_dim=3800,
                    input_length=380))
model.add(Dropout(0.2))
```

In [15]:
```
model.add(LSTM(32))
```

In [16]:
```
model.add(Dense(units=256,
                activation='relu' ))
model.add(Dropout(0.2))
```

In [17]:
```
model.add(Dense(units=1,
                activation='sigmoid' ))
```

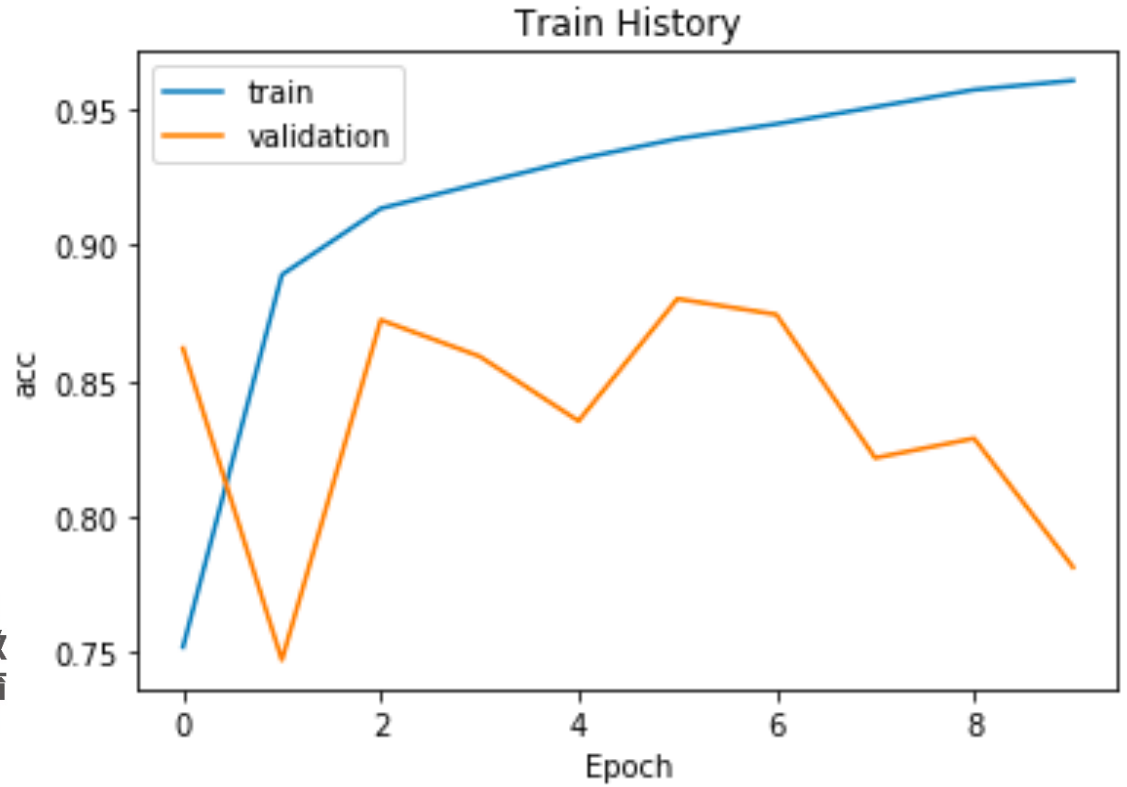
Lab – Sentiment_Analysis_IMDb_LSTM.ipynb (3/5)

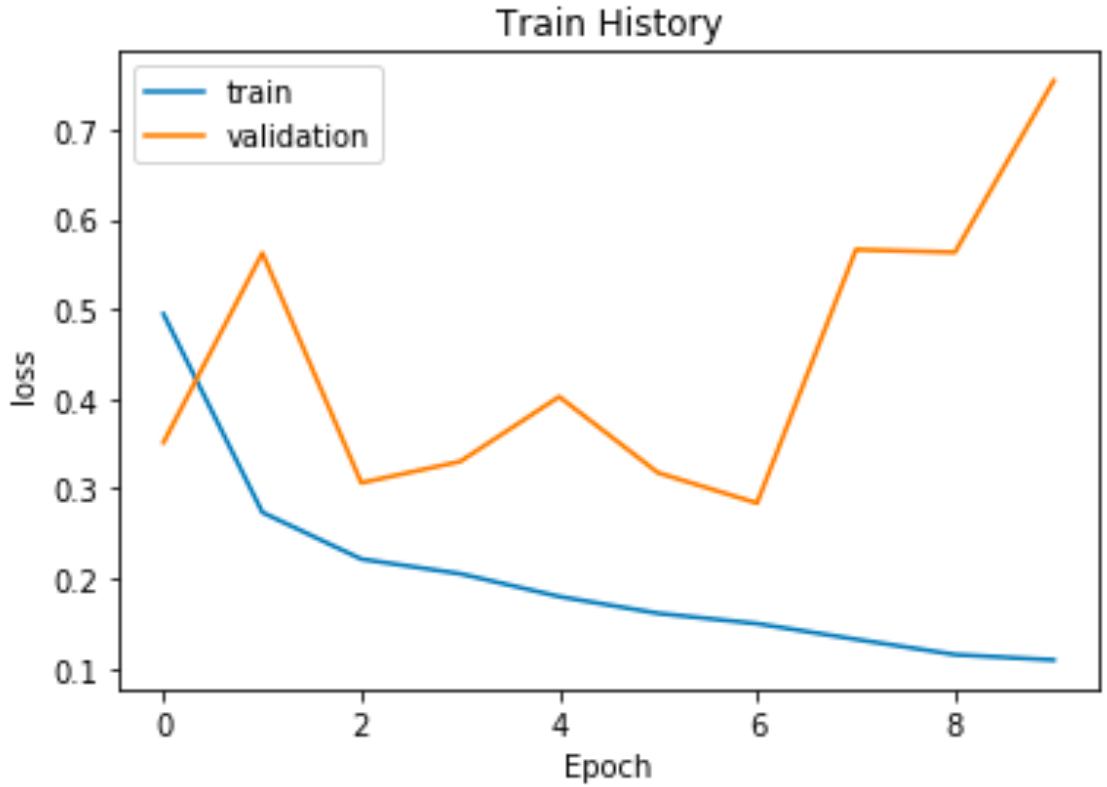
- Step 2. 查看模型摘要 – model.summary()

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 380, 32)           121600
_________________________________________________________________
dropout_1 (Dropout)          (None, 380, 32)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 32)                8320
_________________________________________________________________
dense_1 (Dense)              (None, 256)               8448
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 138,625
Trainable params: 138,625
Non-trainable params: 0
_________________________________________________________________
```

Lab – Sentiment_Analysis_IMDb_LSTM.ipynb (4/5)

- Step 2. 查看模型摘要 – acc and loss





Lab – Sentiment_Analysis_IMDb_LSTM.ipynb (5/5)

- Step 3. 評估模型準確率
- Step 4. 預測機率
- Step 5. 預測結果

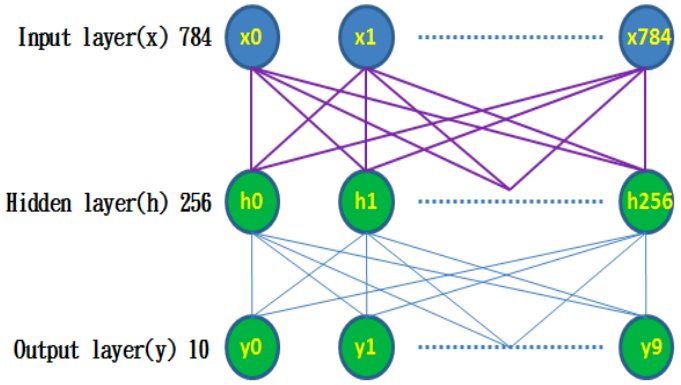
Lab –LSTM_Gen.ipynb

建制完成了字母庫

Step1. 產生Batch Data
Step 2. build training batch generator
Step 3. build validation data

設計一個LSTM Model，它的Unrolling Number為3，Batch Size為2

- 目標：做一個文章產生器，我們希望機器可以不斷的根據前文猜測下一個「字母」(Letters)應該要下什麼



Lab –LSTM_Gen.ipynb解說 (1/7)

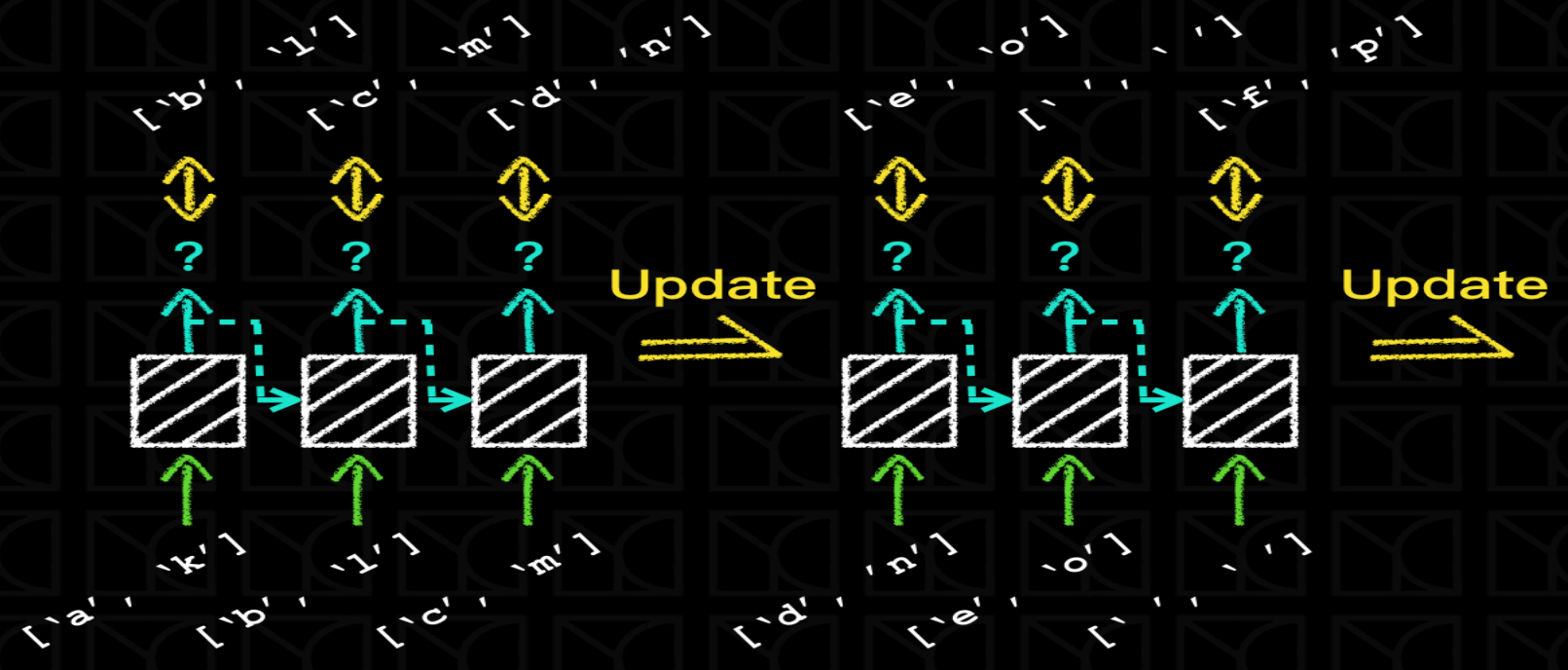
- 目的：使用LSTM實作文章產生器
  - 實作LSTM，目標是做一個文章產生器，我們希望機器可以不斷的根據前文猜測下一個「字母」(Letters)應該要下什麼，如此一來我只要給個開頭字母，LSTM就可以生成一篇文章。
- Step 1. import所需要的套件
- Step 2. 建制完成了字母庫

```
Downloading text8.zip
Found and verified ./text8.zip
=====
Data size 100000000 letters
=====
Train Dataset: size: 99999000 letters,
  first 64: ons anarchists advocate social relations based upon voluntary as
Validation Dataset: size: 1000 letters,
  first 64:  anarchism originated as a term of abuse first used against earl
```

Lab –LSTM_Gen.ipynb解說 (2/7)

- Step 3. 定義一下LSTM架構

```
Unrolling Number = 3
Batch Size = 2

_____________________________

abcde fghij klmno pqrst

Round 1:[[ 'a', 'k' ],
         [ 'b', 'l' ],
         [ 'c', 'm' ],
         [ 'd', 'n' ],]

Round 2:[[ 'd', 'n' ],
         [ 'e', 'o' ],
         [ ' ', ' ' ],
         [ 'f', 'p' ],]

......
```



Lab –LSTM_Gen.ipynb解說 (3/7)

- Step 3. 定義一下LSTM架構
  - 設計一個LSTM Model，它的Unrolling Number為3，Batch Size為2
  - 然後遇到的字串是"abcde fghij klmno pqrst"，接下來就開始產生每個Round要用的Data
  - 產生的結果，會發現產生的Data第0軸表示的是考慮unrolling需要取樣的資料，總共應該會有(Unrolling Number+1)筆，如上圖例，共有4筆，3筆當作輸入而3筆當作Labels，中間有2筆重疊使用

Lab –LSTM_Gen.ipynb解說 (4/7)

- Step 3. 定義一下LSTM架構
  - 會保留最後一筆Data當作下一個回合的第一筆，這是為了不浪費使用每一個字母前後的組合
  - 第1軸則是餵入單一LSTM需要的資料，我們一次可以餵多組不相干的字母進去，如上圖例，Batch Size=2所以餵2個字母進去，那這些不相干的字母在取樣的時候，我們會盡量讓它**平均分配**在文字庫，才能確保彼此之間不相干，以增加LSTM的訓練效率和效果

Lab –LSTM_Gen.ipynb解說 (5/7)

- Step 4. 產生Batch Data
  - 會保留最後一筆Data當作下一個回合的第一筆，這是為了不浪費使用每一個字母前後的組合
  - 定義一下會使用到的函數

```
*** train_batches:
['ons anarchi', 'when milita', 'lleria arch', ' abbeys and', 'married urr', 'hel and ric', 'y and litur', 'ay opened
f', 'tion from t', 'migration t', 'new york ot', 'he boeing s', 'e listed wi', 'eber has pr', 'o be made t', 'yer who
rec', 'ore signifi', 'a fierce cr', ' two six ei', 'aristotle s', 'ity can be ', ' and intrac', 'tion of the', 'dy to
pass ', 'f certain d', 'at it will ', 'e convince ', 'ent told hi', 'ampaign and', 'rver side s', 'ious texts ', 'o c
apitaliz', 'a duplicate', 'gh ann es d', 'ine january', 'ross zero t', 'cal theorie', 'ast instanc', ' dimensiona',
'most holy m', 't s support', 'u is still ', 'e oscillati', 'o eight sub', 'of italy la', 's the tower', 'klahoma pr
e', 'erprise lin', 'ws becomes ', 'et in a naz', 'the fabian ', 'etchy to re', ' sharman ne', 'ised empero', 'ting in
pol', 'd neo latin', 'th risky ri', 'encyclopedi', 'fense the a', 'duating fro', 'treet grid ', 'ations more', 'appea
l of d', 'si have mad']
['ists advoca', 'ary governm', 'hes nationa', 'd monasteri', 'raca prince', 'chard baer ', 'rgical lang', 'for passen
g', 'the nationa', 'took place ', 'ther well k', 'seven six s', 'ith a gloss', 'robably bee', 'to recogniz', 'ceived
the ', 'icant than ', 'ritic of th', 'ight in sig', 's uncaused ', ' lost as in', 'cellular ic', 'e size of t', ' him
a stic', 'drugs confu', ' take to co', ' the priest', 'im to name ', 'd barred at', 'standard fo', ' such as es', 'ze
on the g', 'e of the or', 'd hiver one', 'y eight mar', 'the lead ch', 'es classica', 'ce the non ', 'al analysis',
'mormons bel', 't or at lea', ' disagreed ', 'ing system ', 'btypes base', 'anguages th', 'r commissio', 'ess one ni
n', 'nux suse li', ' the first ', 'zi concentr', ' society ne', 'elatively s', 'etworks sha', 'or hirohito', 'litical
ini', 'n most of t', 'iskerdoo ri', 'ic overview', 'air compone', 'om acnm acc', ' centerline', 'e than any ', 'devot
ional ', 'de such dev']
*** valid_batches:
[' a']
['an']
```

Lab –LSTM_Gen.ipynb解說 (6/7)

- Step 5. 開始建制LSTM Model
  - build training batch generator
  - build validation data
  - build LSTM mode
  - initial model
  - online training

```
Epoch 1/30: 75s loss = 1.9014, perplexity = 5.9473
Epoch 2/30: 75s loss = 1.5807, perplexity = 6.2555
Epoch 3/30: 75s loss = 1.5002, perplexity = 5.6864
Epoch 4/30: 74s loss = 1.4585, perplexity = 5.7218
Epoch 5/30: 75s loss = 1.4371, perplexity = 6.2142

============== Validation ==============
validation perplexity = 3.6917
Generate From 'a':   ats of gioron one nine six zero one nine eight fix
Generate From 'h':   htan but the forms c chicoloveous complories of the
Generate From 'm':   m two seven receyshow there isesidon more oftencase
========================================
```

Lab –LSTM_Gen.ipynb解說 (7/7)

- Step 6. 產生一篇以"p"為開頭的1000字文章

In [8]:
```
#print(model_LSTM.generate(c='t',len_generate=1000))
print(model_LSTM.generate(c='petter',len_generate=1000))
```

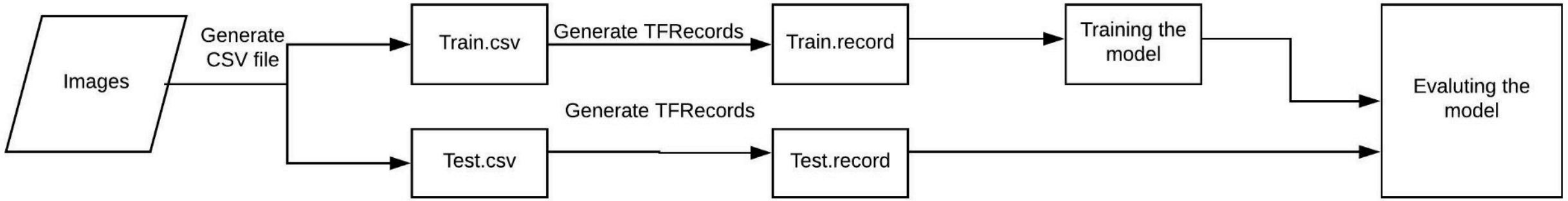
```
proted supports on tributary the question emigral coupleli three zero more the world colleges classoric a researchers
mumble or emits of and politics reals dimensional for electric and the place and that subdismer arts of forms of larg
ely number and algorithm and he weeedd of albondings company an united states f the ones the herber houses turned on
imprivation gender the monristances compared in laziel is alber s published from gashim tokana and one nine four eigh
t which john chuattel the variation had ship association phamill antonialle it for heading a setting for book new yor
k charles information retrine in two zero zero three two two b g mooporu illinoic as reption of intines defined or it
s linguy should educate the department power and loui postuna the blon bonnshm climate british discrete don shirtm th
e assoceatic clersor and however unx son in municial to absorle t c two five eight leap a defined that elements seaso
ns b of the first oreplated from ecolongments are allegan joing
```

物件偵測– Labs

- TensorFlow 實現物件偵測範例Image為例
  - 選取 models/research/object_detection/object_detection_tutorial.ipynb
  - 更換mode - object_detection_changed_model.ipynb
- 影片與網路攝影機的物件辨識
  - models/research/object_detection/stream.ipynb

Lab – object_detection_tutorial.ipynb (1/4)

- 執行邏輯



Lab – object_detection_tutorial.ipynb (2/4)

- Step 1. Import所需要的套件
- Step 2. Env setup and Object detection imports

Imports

In [18]:
```
from distutils.version import StrictVersion
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import pylab

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")
from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
  raise ImportError('Please upgrade your TensorFlow installation to v1.9.* or later!')
```

Env setup

In [19]:
```
# This is needed to display the images.
%matplotlib inline
```

Object detection imports

Here are the imports from the object detection module.

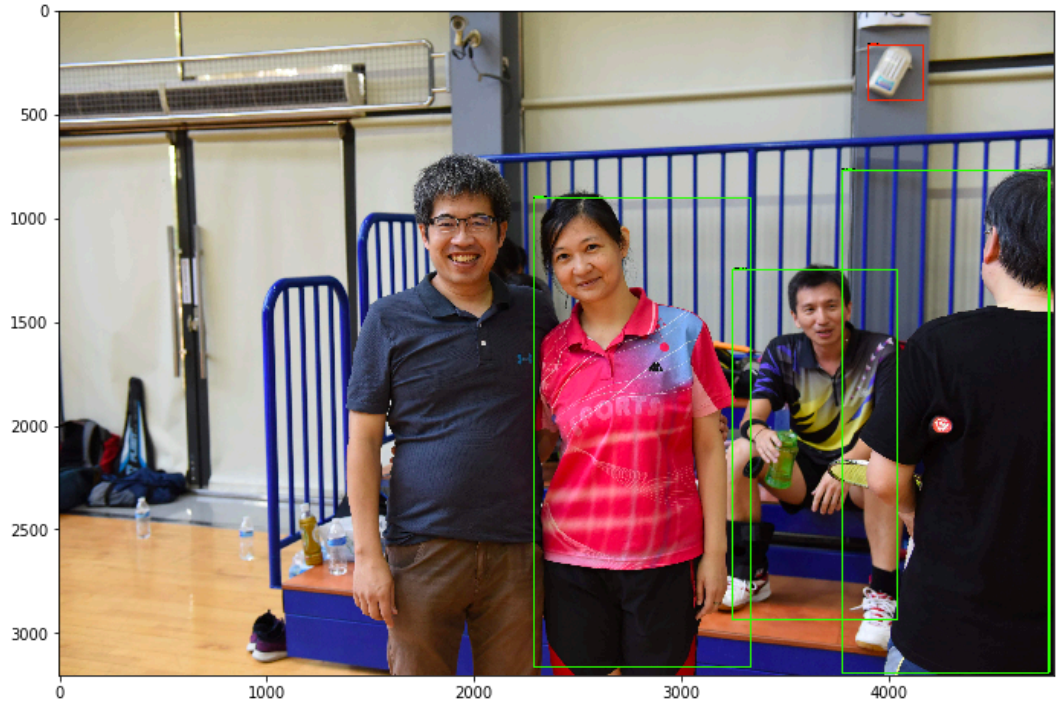
In [20]:
```
from utils import label_map_util

from utils import visualization_utils as vis_util
```

Lab – object_detection_tutorial.ipynb (3/4)

- Step 3. Model Preparation
  - 所有模型由於訓練非常龐大，故我們擷取 - https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md
  - 下載模型
- Step 4. Load a (frozen) Tensorflow model into memory
- Step 5. Load label map
  - 有label map
  - 沒有label map
- Step 6. Detection實驗

Lab – object_detection_tutorial.ipynb (4/4)

- TensorFlow 實現物件偵測範例Image為例
- 結果：
  - 這個範例程式碼所使用的模型是 SSD + Mobilenet，辨識物件的速度非常快
  - 但是精確度似乎不是非常好



Lab – object_detection_changed_model.ipynb (1/2)

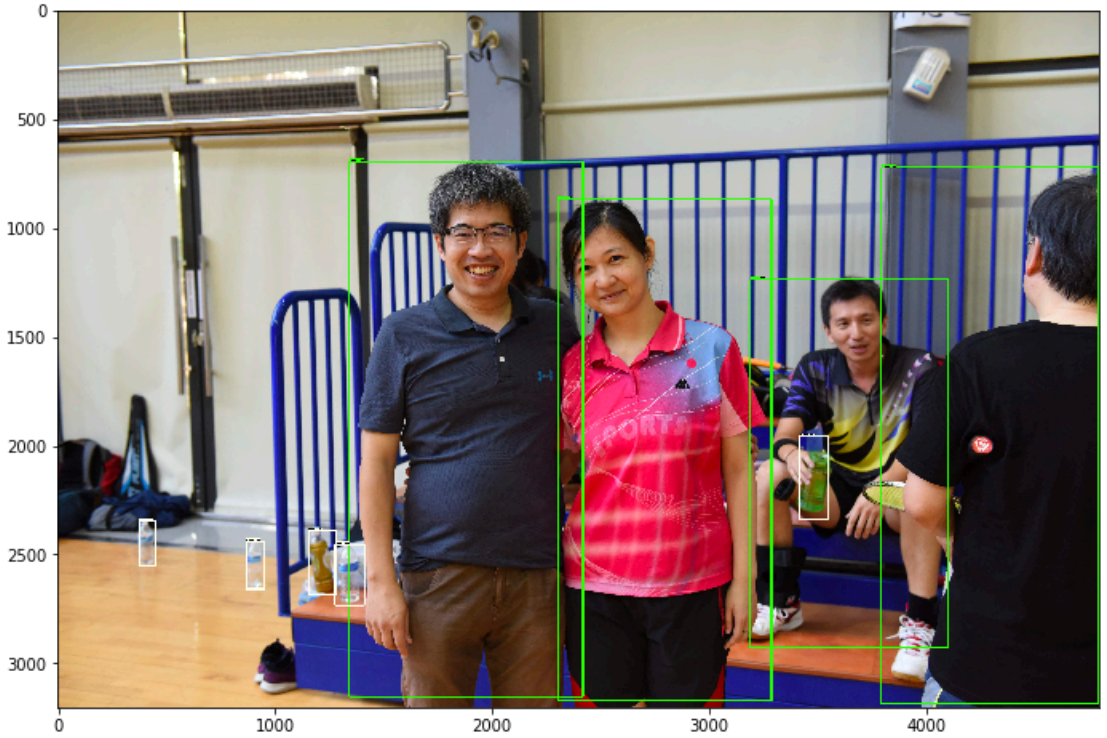
- 更換模型 - Faster RCNN + NAS 模型

object_detection_changed_model.ipynb

```
# 使用 Faster RCNN + NAS 模型

MODEL_NAME = 'faster_rcnn_nas_coco_2017_11_08' MODEL_FILE = MODEL_NAME +
'.tar.gz' DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'
```

Lab – object_detection_changed_model.ipynb (2/2)

- 結果：
  - 換成 Faster RCNN + NAS 模型之後，大部分的結果都很不錯
  - 還是會有誤判，不過感覺起來準確度是可以接受的

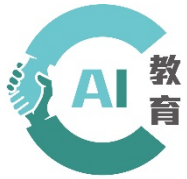


Lab –stream.ipynb

- Step 1. Import相關套件
- Step 2. 使用OpenCV模組，並讀取影像
- Step 3. 載入事先訓練好的模型
- Step 4. 處理好bounding box，並繪製出影片

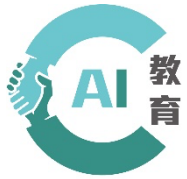
TF2_Labs

Thank You!!



附錄

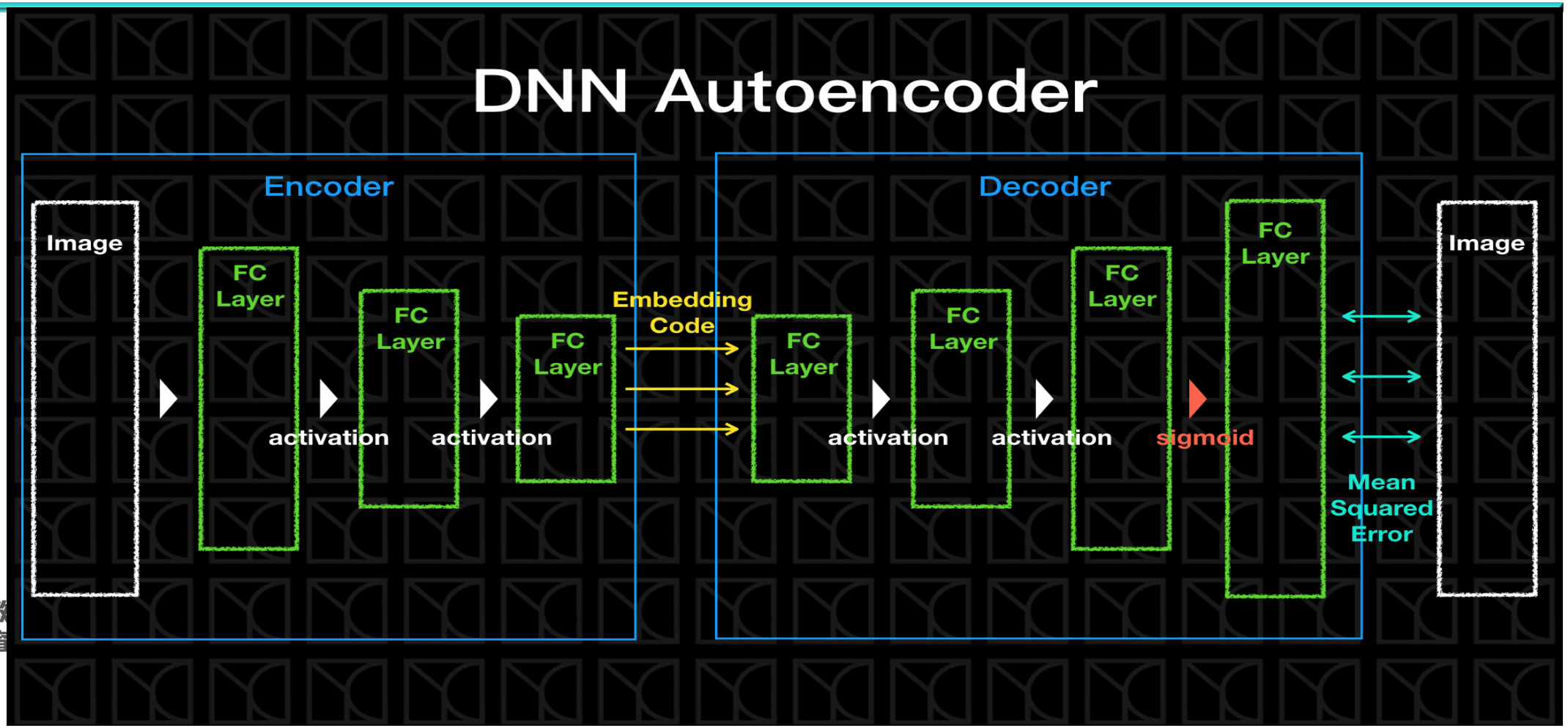
1. Autoencoder
2. 去雜訊(De-noise) Autoencoder



AutoEncoder (1/2)

- 概念：就是做資訊的壓縮
  - 如果原本是一張MNIST的圖，有28x28=784個Pixels，作壓縮之後，就要使得壓縮後的神經元可以比784個更少。
- 從高維度的空間轉換到低維度的空間，再轉換回去原本的維度
- 非線性轉換 – Activate Function
- 實現Autoencoder和之前DNN並沒有太大的差異

AutoEncoder (2/2)



Lab – Autoencoder.ipynb (1/4)

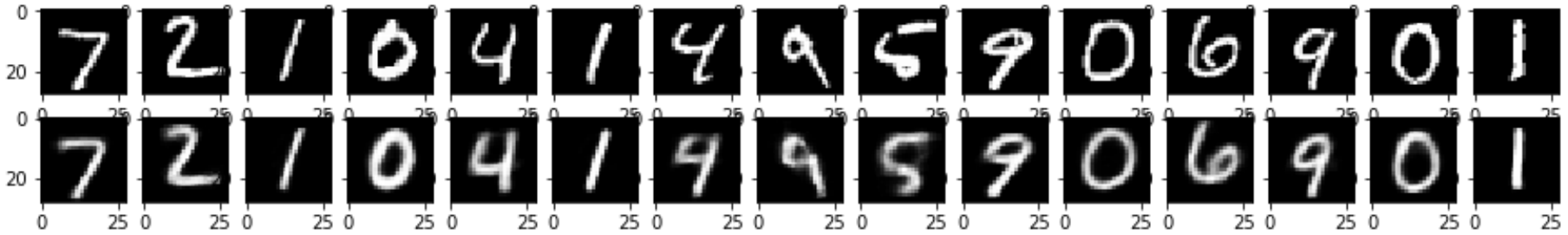
- 特別把encoder額外的在structure裡頭輸出出來
  - 增加新的函數encode，讓使用者可以使用Train好的Encoder來做Encode。
- 使用Weight-Elimination L2 Regularizer
  - 這個Regularizer的好處是會使得權重接近Sparse，也就是說權重會留下比較多的0，這有一個好處，就是每個神經元彼此之間的依賴減少了，因為內積(評估相依性)時有0的那個維度將不會有所貢獻
- Step 1. import所需要的套件
- Step 2. 實作Autoencoder類別

Lab – Autoencoder.ipynb (2/4)

- Step 3. 把原本的Input當作Output的目標答案去訓練Neural Network
- Step 4. 測試：上排是進去Autoencoder之前的圖片，下排是經過Autoencoder後的圖片

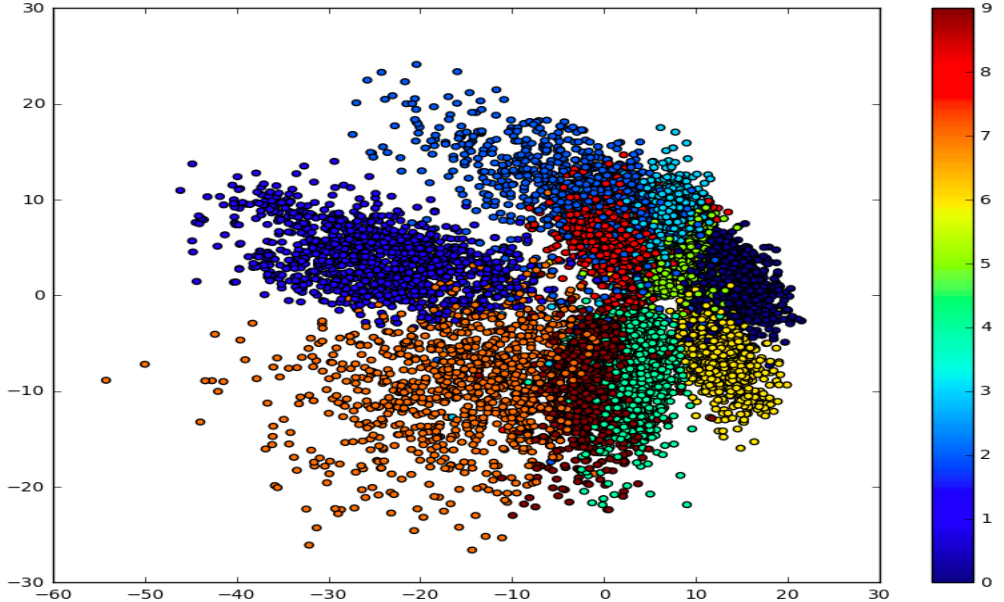
Lab – Autoencoder.ipynb (3/4)

- Step 5. 做Regularization



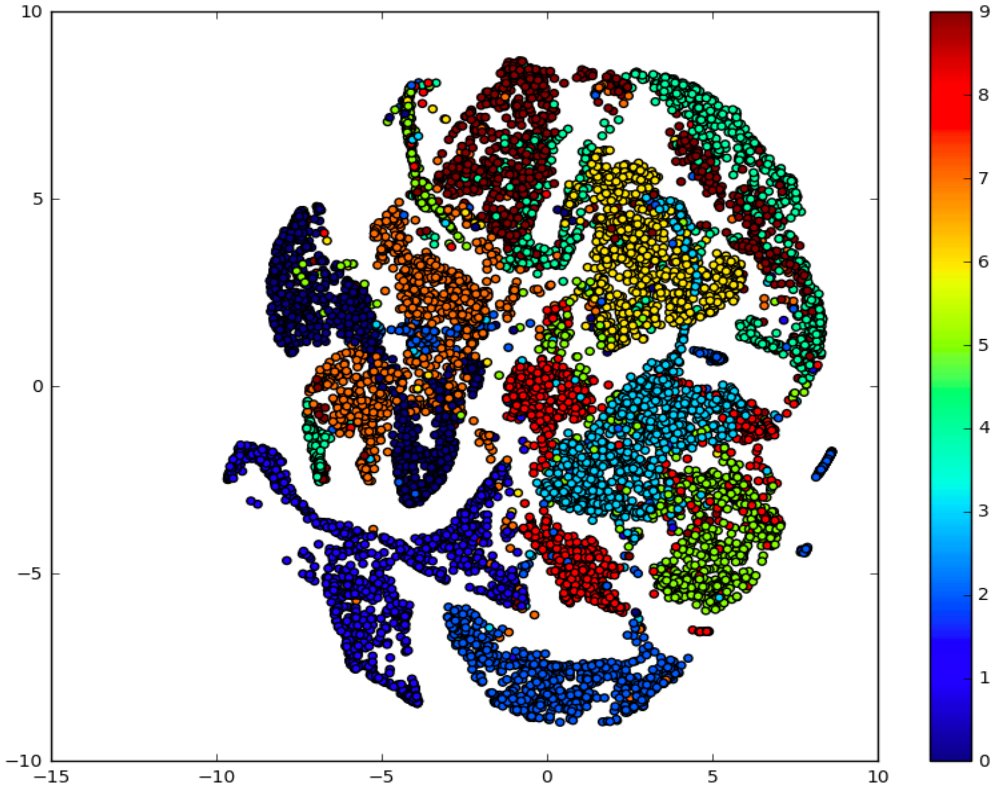
- Step 6. 壓縮碼Code與視覺化
  - 2D Visualization最流行的作法 - PCA

用線性的方式去做座標轉換，從一個橫切面去看數據，較粗略的轉換並不能在視覺化時看出資料和資料間彼此的距離，尤其是從高維度轉換過來，經常會失真



Lab – Autoencoder.ipynb (4/4)

- Step 6. 壓縮碼Code與視覺化
  - t-SNE是針對數據和數據間的距離去做轉換，最後被攤成2維時正是顯示數據點的距離關係，更能描述群聚的現象

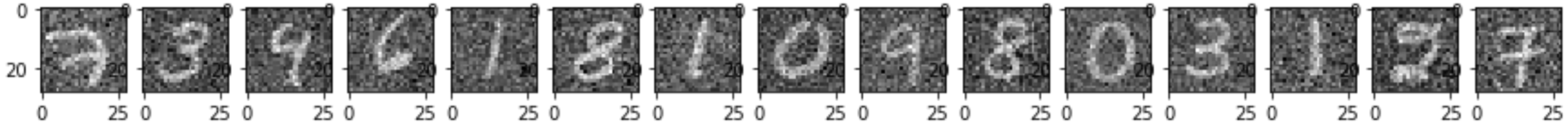


去雜訊(De-noise) Autoencoder

- 概念：將原本Autoencoder的前面加了一道人工雜訊的流程，但是最終又要讓Autoencoder試著去還原出原來沒有加入雜訊的資訊
- 目的：可以找到一個Autoencoder是可以自行消除雜訊
  - Denoising Autoencoder加到正常Neural Network的前面
  - 這個Neural Network就擁有了抑制雜訊的功用
  - 可以當作一種Regularization的方法

Lab – De-noise Autoencoder.ipynb

- Step 1. 先將圖片加上雜訊



- Step 2.髒圖片當作Input，正常圖當作Output的目標，進行訓練
- Step 3. 檢視結果