Deep Learning with Keras and Tensorflow

洪孝宗 Alex Hung

Ph.D. Student

臺灣師範大學 資訊工程研究所

語音暨機器智能實驗室

https://github.com/AlexHung780312/DL-TEQMS

Outline

1. 建立基礎觀念
   - Deep Learning 簡介，觀念釐清
2. 機器學習流程
   - NumPy+Tensorflow
3. 深層學習的特色
   - Keras
4. 建立舒適的工作環境
5. 結語
6. 16:00-18:00 Tutorial session

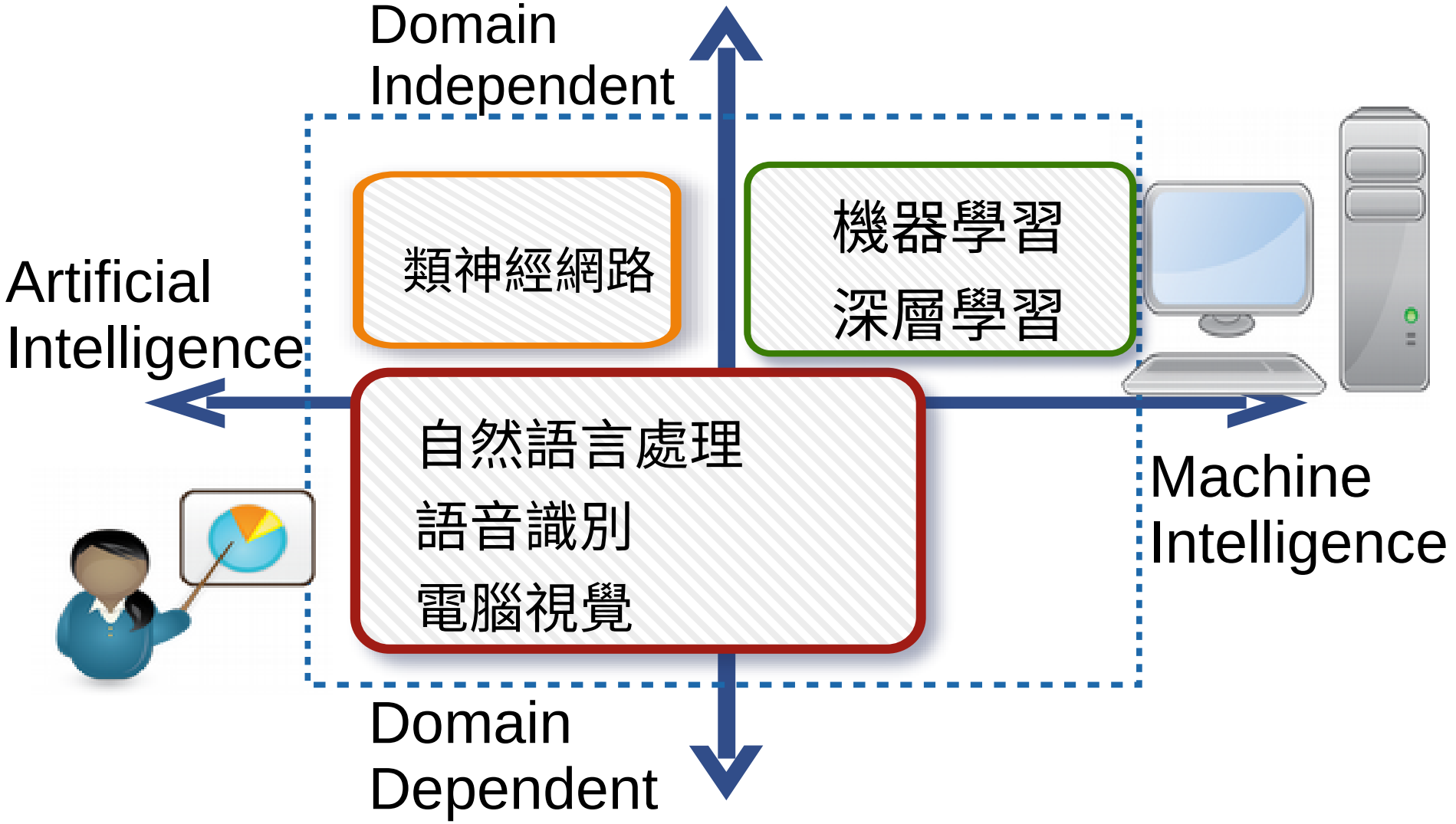
Outline

基礎理論與觀念

常見的疑問：

- 深層學習，機器學習，人工智慧的差異
- 深層類神經網路，就是多加幾層網路
- 資源問題
  - 沒有大數據，就不能參與
  - 需要昂貴的機器

領域區別


Domain
Independent
類神經網路
機器學習
深層學習
Artificial
Intelligence
自然語言處理
語音識別
電腦視覺
Machine
Intelligence
Domain
Dependent

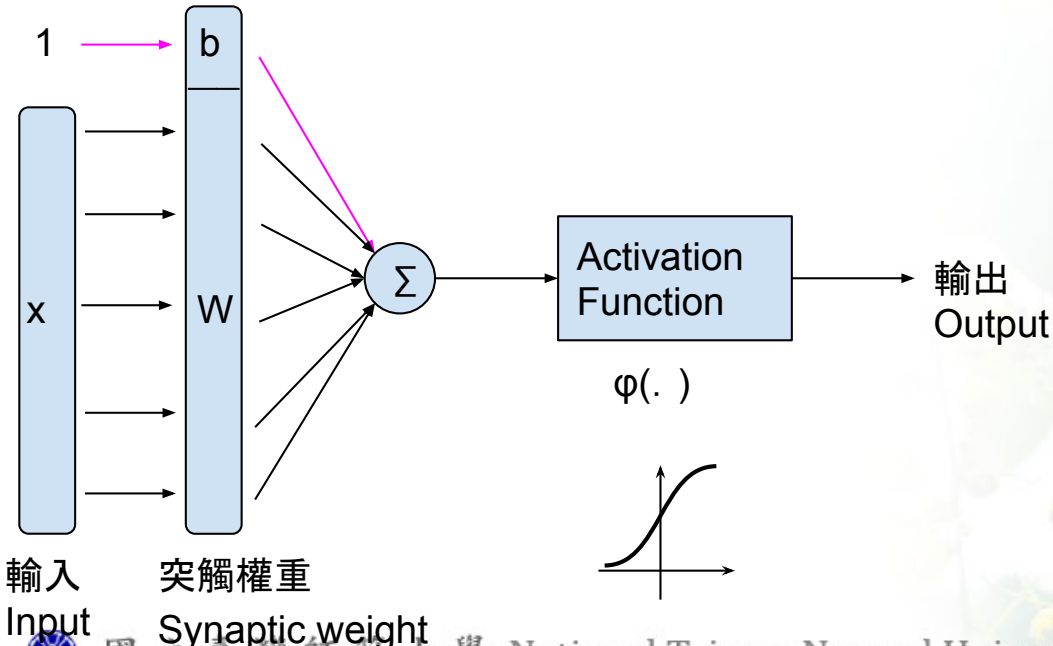
人工智能 v.s. 機器智能

- 專家介入成份：高 $\longrightarrow$ 低
- 圍棋專家教電腦下棋$\longrightarrow$專家教電腦如何學棋
- 資料依賴性：低 $\longrightarrow$ 高
- 從資料中學習

典型神經元模型

- 神經元會接受其他細胞的刺激
- 當刺激量足夠時，神經元會呈現激發狀態

$$y = \phi(\mathbf{w}^\top \mathbf{x} + b)$$



學習機器 Learning Machine

- 腦是一個”Learning Machine”
  - 大腦可以學習萬物，但腦細胞種類不多
  - 利用大量神經元，近似各種函數
- 類神經網路可以近似什麼函數？[1]
  - $\mathbf{x}$ 和 $f(\mathbf{x})$ 屬於N維 Unit Cube $[0,1]^N$ 空間
  - 存在 $G(\mathbf{x}) = \sum_j^N \alpha_j \sigma(\mathbf{w}_j^\top \mathbf{x} + \mathbf{b})$
  - 滿足 $|G(\mathbf{x}) - f(\mathbf{x})| < \epsilon$
  - 例如”機率”，Bayesian network
- $\alpha_j, \mathbf{w}_j, \mathbf{b}$ 是長期記憶，$f(x)$ 是短期記憶

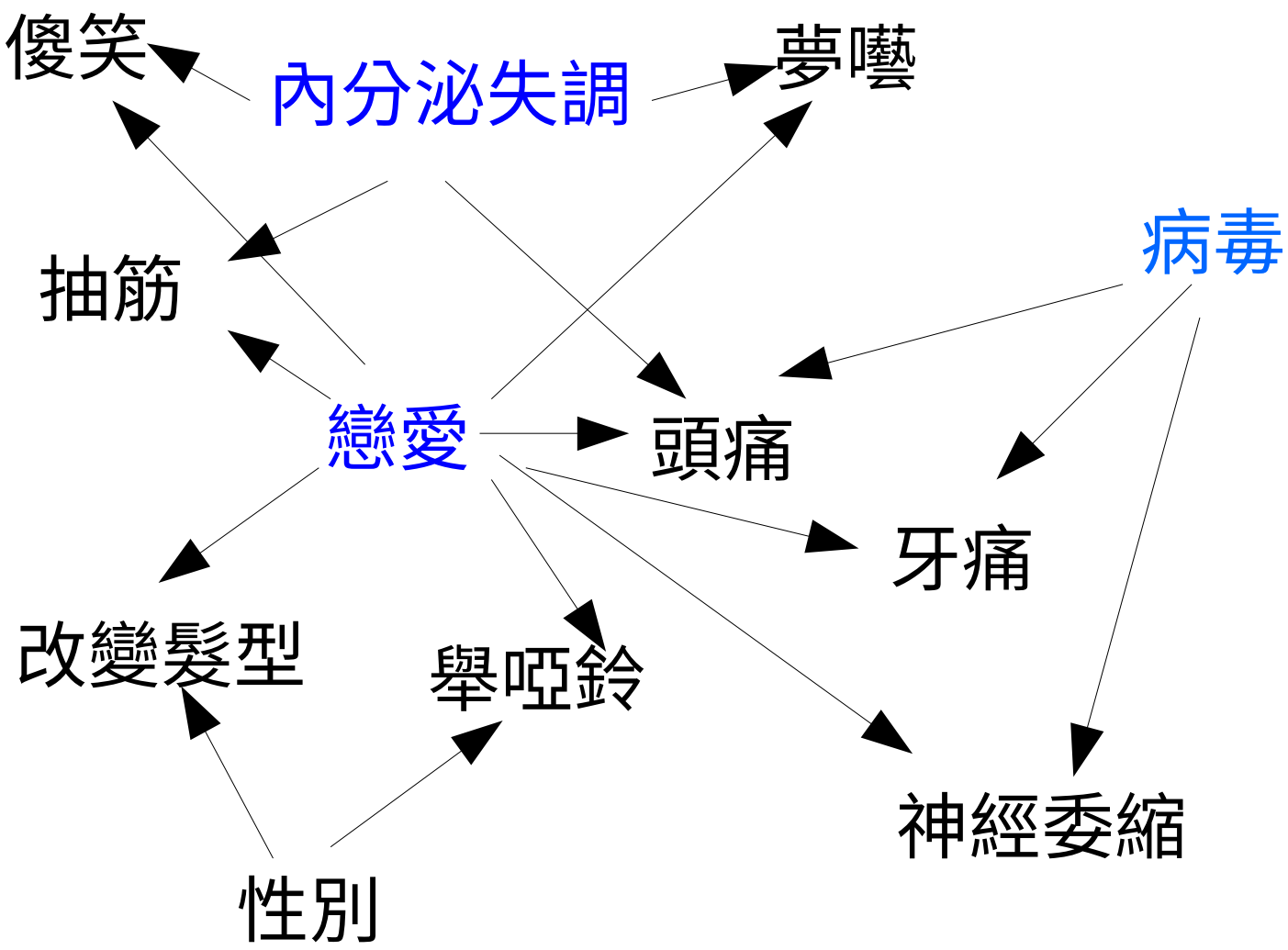
學習機器 Learning Machine

- 腦是一個”Learning Machine”
  - 大腦可以學習萬物，但腦細胞種類不多
  - 利用大量神經元，近似各種函數
- 類神經網路可以近似什麼函數？[1]
  - $\mathbf{x}$ 和 $f(\mathbf{x})$ 屬於N維 Unit Cube $[0,1]^N$ 空間
  - 存在 $G(\mathbf{x}) = \sum_j^N \alpha_j \sigma(\mathbf{w}_j^\top \mathbf{x} + \mathbf{b})$
  - 滿足 $|G(\mathbf{x}) - f(\mathbf{x})| < \epsilon$
  - 例如”機率”，Bayesian network
- $\alpha_j, \mathbf{w}_j, \mathbf{b}$ 是長期記憶，$f(x)$ 是短期記憶

戀愛症候群 – 專家知識

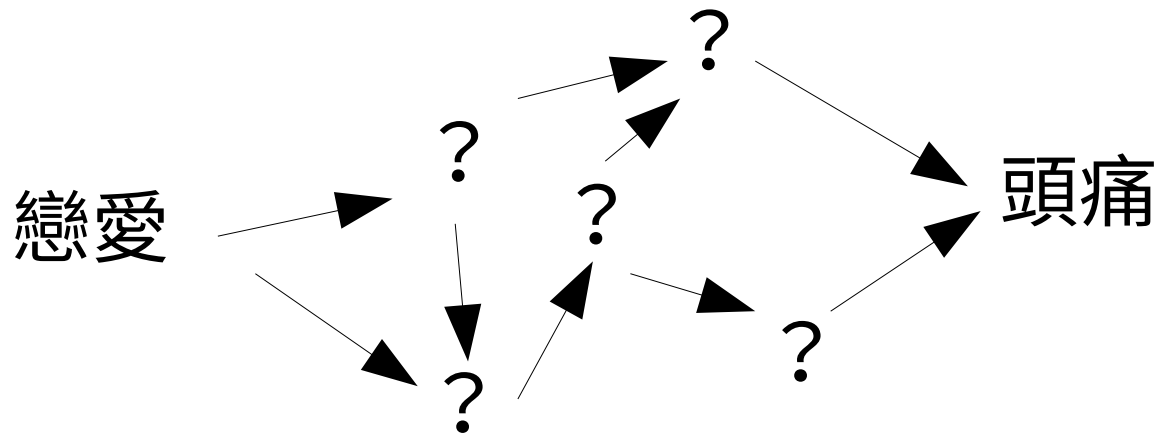
- 不管性別年齡職業體重學歷長相和血型
  沒有一個人可以免疫
- 洗澡洗得特別乾淨，刷牙得特別用力
- 女人突然改變髮型，男人開始每天練著啞鈴
- 專家學者研究後相信，戀愛是內分泌失調所引起
  卻有別人認為戀愛屬於濾過性病毒

戀愛症候群 – Baysian Network



深層學習 – 不只是網路多幾層

- 根據 專家 的知識，設計一個 學習機器
- 沒有單一模型適用全部任務
- 填補專家知識不足之處
  - 當知識可能細節描述
  - 部份資訊無法取得
  - 文字，語音，影像需要不同的設計



大公司，大數據，大者恆大

大公司的優勢：

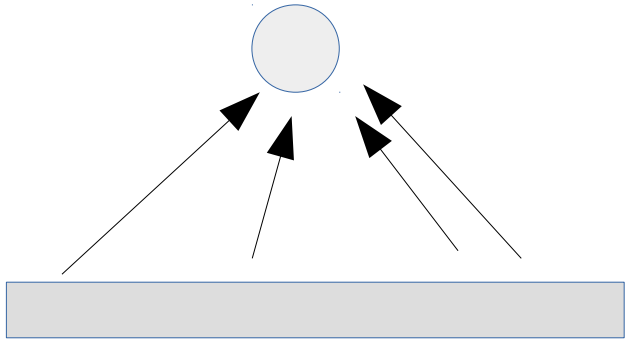
- 資料越多，未知情況越少[2]
- 學界(CS)難以跟上實驗規模，無法重製實驗
  - 加入大公司
  - 純理論研究相當困難
- 大量測資的平均表現很好

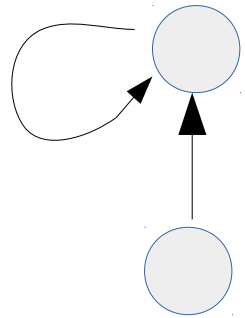
統計學者建議：”Key data” (R-Taiwan)

計算理論的反向思考

問題：輸入資料有 100 個位元，請問加總是多少？

$f([0,0,1,...,1,0]) = 15$





計算理論：[3]

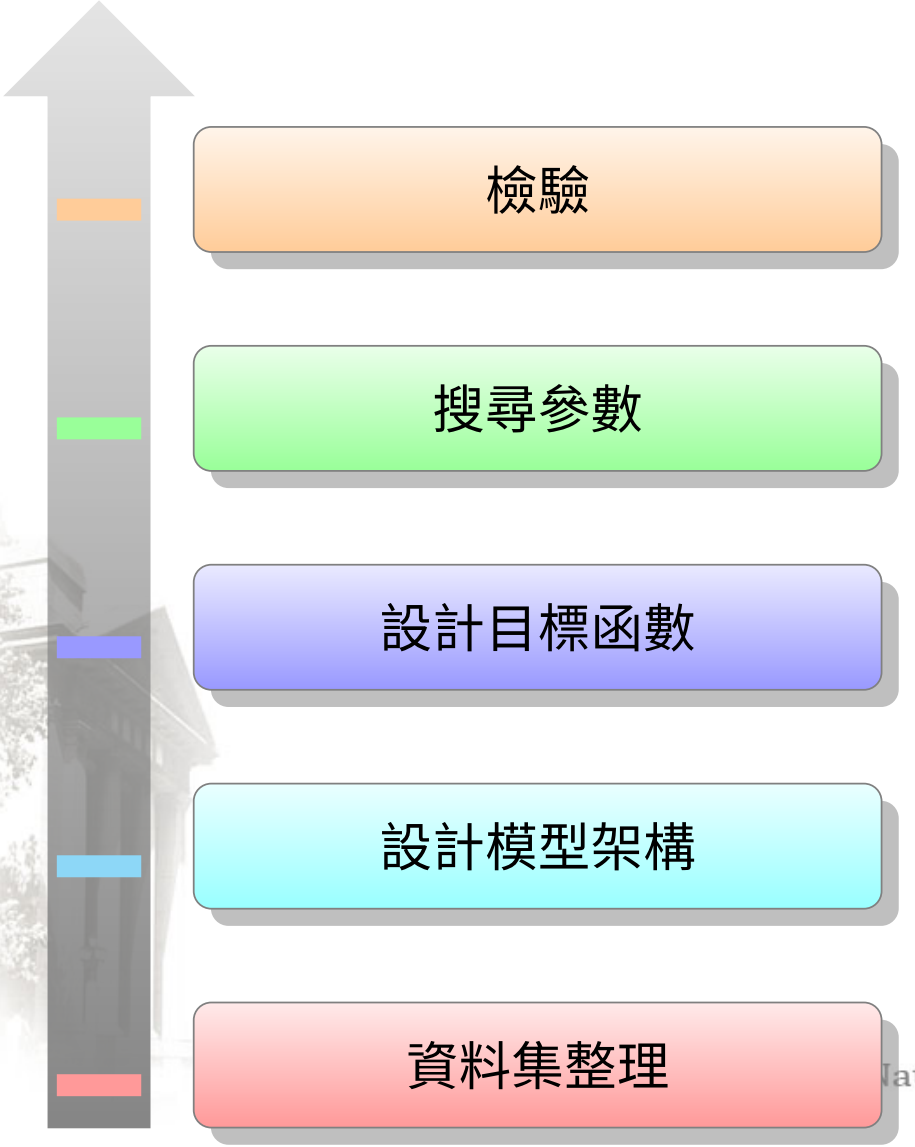
- $f : X \to Y$ 是解釋 三筆資料(x, y) 的 最簡短程式

建議：盡可能簡化搜尋範圍＋補充 Key Data

Outline

機器學習 Machine Learning

尋找合理的推論⟶尋找參數



- 資料:訓練,[驗證].測試
- 架構 = 搜尋範圍
- 目標函數 = 搜尋偏好
- 從資料學習參數

範例：單車租借量預測[1]

- 問題：預測當日單車租借次數 (day.csv)
- 輸入資料：
  - season, yr, mnth 四季，年份，月份 (整數)
  - holiday,weekday,workingday
    是否為工作天？（整數+布林值）
  - weathersit,temp,atemp,hum,windspeed
    (天氣資訊，實數)

[1] https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset

	A	B	C	D	E	F	G	H	I	J	K	
1	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum
2	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.8
3	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.6
4	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.4
5	4	2011-01-04	1	0	1	0	2	1	1	0.2	0.212122	0.5
6	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.22927	0.4
7	6	2011-01-06	1	0	1	0	4	1	1	0.204348	0.233209	0.5
8	7	2011-01-07	1	0	1	0	5	1	2	0.196522	0.208839	0.4
9	8	2011-01-08	1	0	1	0	6	0	2	0.165	0.162254	0.5
10	9	2011-01-09	1	0	1	0	0	0	1	0.138333	0.116175	0.4
11	10	2011-01-10	1	0	1	0	1	1	1	0.150833	0.150888	0.4
12	11	2011-01-11	1	0	1	0	2	1	2	0.169091	0.191464	0.6
13	12	2011-01-12	1	0	1	0	3	1	1	0.172727	0.160473	0.5
14	13	2011-01-13	1	0	1	0	4	1	1	0.165	0.150883	0.4
15	14	2011-01-14	1	0	1	0	5	1	1	0.16087	0.188413	0.5
16	15	2011-01-15	1	0	1	0	6	0	2	0.233333	0.248112	0.
17	16	2011-01-16	1	0	1	0	0	0	1	0.231667	0.234217	0.
18	17	2011-01-17	1	0	1	1	1	0	2	0.175833	0.176771	0
19	18	2011-01-18	1	0	1	0	2	1	2	0.216667	0.232333	0.8
20	19	2011-01-19	1	0	1	0	3	1	2	0.292174	0.298422	0.7
21	20	2011-01-20	1	0	1	0	4	1	2	0.261667	0.25505	0.5
22	21	2011-01-21	1	0	1	0	5	1	1	0.1775	0.157833	0.4
23	22	2011-01-22	1	0	1	0	6	0	1	0.0591304	0.0790696	
24	23	2011-01-23	1	0	1	0	0	0	1	0.0965217	0.0988391	0.4
25	24	2011-01-24	1	0	1	0	1	1	1	0.0973913	0.11793	0.4
26	25	2011-01-25	1	0	1	0	2	1	2	0.223478	0.234526	0.6
27	26	2011-01-26	1	0	1	0	3	1	3	0.2175	0.2036	0
28	27	2011-01-27	1	0	1	0	4	1	1	0.195	0.2197	0
29	28	2011-01-28	1	0	1	0	5	1	2	0.203478	0.223317	0.7

資料編碼

- 類別型資料 $\rightarrow$ 布林值
  - 登山車、公路車、折疊車 $\rightarrow [1,0,0], [0,1,0], [0,0,1]$
  - 詞典
- 有順序關係 $\rightarrow$ 整數
  - 月份，四季，時間
- 實數量測值
  - 氣溫，濕度....

編碼方式不是唯一，例如四季可以沒有順序

也可以改變順序 夏-春-冬-秋 (不好睡 - 好睡)

利用 NumPy 處理資料

讀入CSV檔案，存放為NumPy矩陣

記得資料格式為float32

```
import numpy as np
data=[]
with open('BikeSharingData/day.csv','r') as f:
  f.readline() # skip first row
  for line in f:
    columns = line.strip().split(',')
    # from season to cnt
    data.append(columns[2:])
data_mat = np.asarray(data, dtype=np.float32)
```

讀取，保存

- np.save, np.load
- 千萬不要使用 pickle 存放 NumPy資料

```
np.save('BikeSharingData/day.npy', data_mat)
data_restore = np.load('BikeSharingData/day.npy')
```

- 檢查資料

```
print(data_restore.shape) # (731, 14)
print(data_restore[0,:])  # first row
print(data_restore[:,-1]) # last column
```

資料集規劃

- 一般實驗包含三個資料集
  - 訓練集 (可視)
  - 驗證集(validation set)(可視)
  - 測試集 (不可視)
- 自己分配 8:1:1（事情況調整）
- 測試集要避免作弊的可能性
  - 奇數天，偶數天; 過去，未來

```
# only train set and test set
tr = data_restore[0:-100,:]
ts = data_restore[-100:,:]
```

資料正規化

- 調整至標準常態分布

  (Mean Variance Normalization)

$$\hat{x} = \frac{x - \bar{x}}{\sigma}$$

- 避免被數值範圍大小影響
- 注意：只能從 可視資料 中計算平均值和標準差

```
mean = np.mean(tr, axis=0)
stdv = np.std(tr, axis=0)
tr = (tr - mean) / stdv
ts = (ts - mean) / stdv
```

準備輸入特徵、輸出答案

tr 和 ts 是完整的表格，分離輸入 $\mathbf{x}$ 和輸出 y

前面 11 維是輸入特徵 $\mathbf{x}$，最後一維是單車數量 y

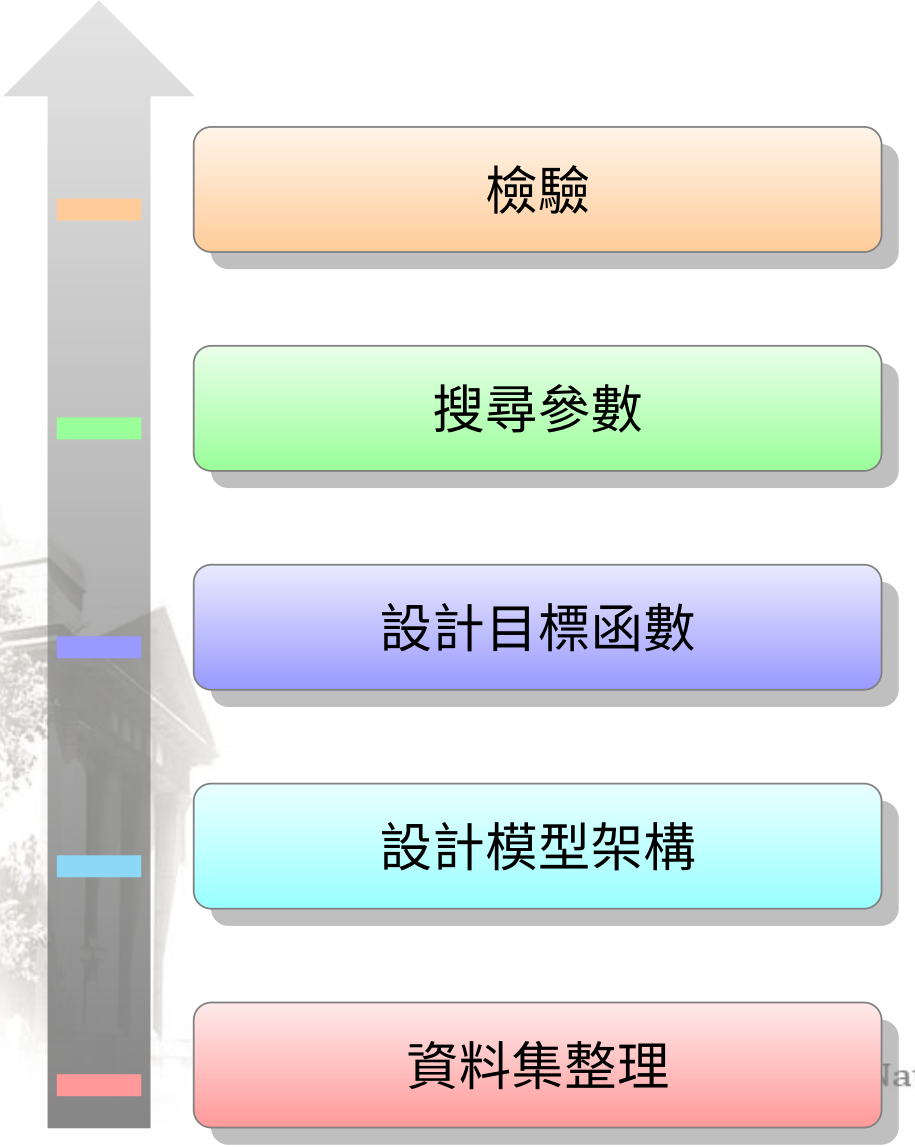
```
x_train, y_train = tr[:, 0:11], tr[:, -1:]
x_test, y_test = ts[:,0:11,], ts[:, -1:]
print(x_train.shape)
print(y_train.shape)
```

注意：y 一樣要是矩陣(Rank=2)

資料準備完成！

機器學習 Machine Learning

尋找合理的推論$\longrightarrow$尋找參數



- 資料:訓練,[驗證].測試
- 架構 = 搜尋範圍

設計模型架構

- 定義兩層隱藏層的網路

$$\mathbf{h}_1 = \sigma(W_1\mathbf{x} + \mathbf{b}_1),$$

$$\mathbf{h}_2 = \sigma(W_2\mathbf{h}_1 + \mathbf{b}_2),$$

$$y = f(\mathbf{x}) = W_o\mathbf{h}_2$$

- 在 Tensorflow 中，分為常數、變數、輸入參數
  - tf.constant, 不會改變的數值
  - tf.Variable, 變數(可學習)，例如$W_o, W_1, W_2, \mathbf{b}_1, \mathbf{b}_1$
  - tf.placeholder 輸入參數，例如$\mathbf{x}, y$

定義變數

- tf.Variable(初始化方式, name=”名稱”)
- 矩陣大小 = (輸入, 輸出)

```
import tensorflow as tf
feat_dim, hid_dim, out_dim = x_train.shape[1], 16, 1
W_1 = tf.Variable(tf.random_normal([feat_dim, hid_dim],
                                   stddev=1.0), name="W_1")
b_1 = tf.Variable(tf.zeros([hid_dim]), name="b_1")
W_2 = tf.Variable(tf.random_normal([hid_dim, hid_dim],
                                   stddev=1.0), name="W_2")
b_2 = tf.Variable(tf.zeros([hid_dim]), name="b_2")
W_o = tf.Variable(tf.random_normal([hid_dim, out_dim],
                                   stddev=1.0), name="W_o")
```

初始化方式

可以嘗試替換

```
tf.zeros(shape=(4))
tf.ones(shape=(4))
-tf.ones(shape=(4))
tf.random_uniform(shape=(5,4), minval=0,maxval=1.0)
tf.random_normal(shape=(5,4), stddev=1.0))
```

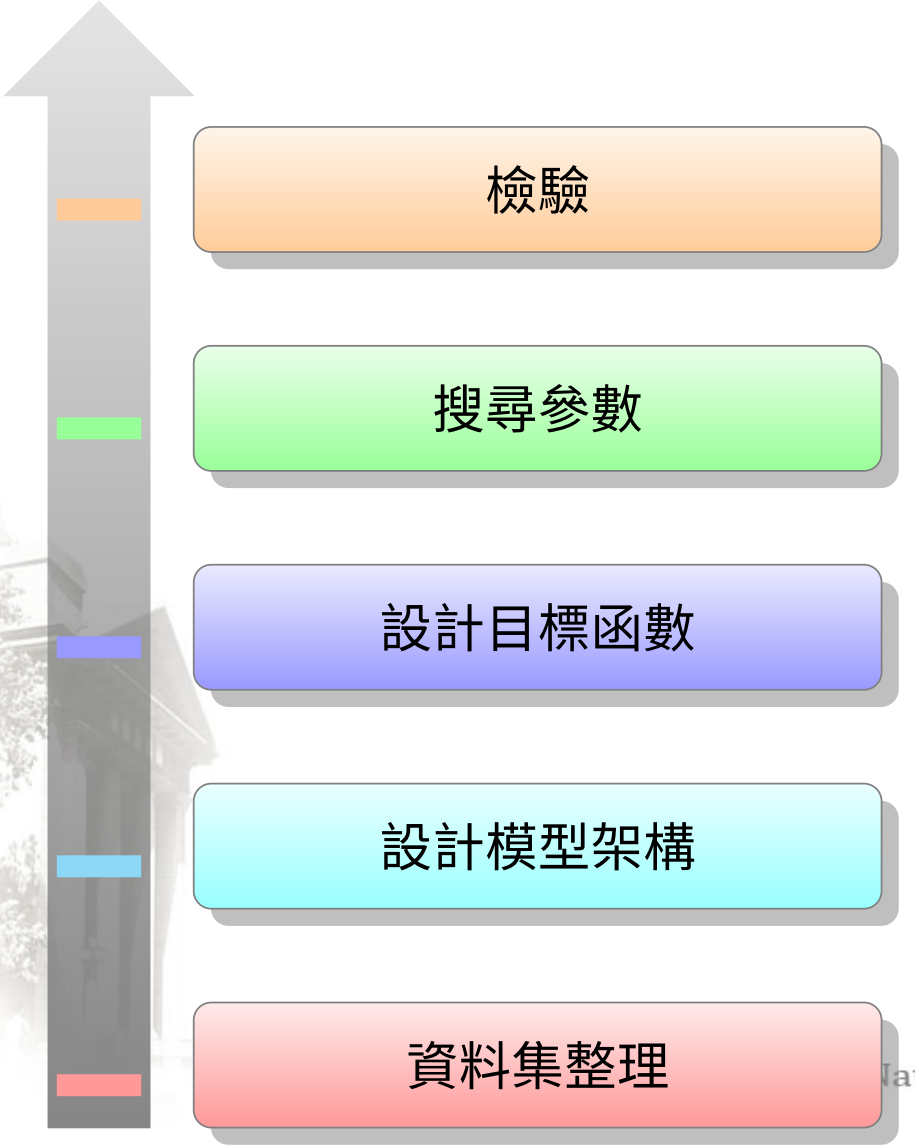
定義計算圖與輸入參數

- 允許一次計算多筆資料，所以定義 shape=(None, feat_dim)
- $W\mathbf{x}$ 修正為 XW，可以維持資料數目在第一微度
- (n, feat_dim) X (feat_dim, hid_dim) $\rightarrow$ (n, hid_dim)

```
x = tf.placeholder(dtype=tf.float32, shape=(None,
                                    feat_dim), name="x")
h1 = tf.sigmoid(tf.matmul(x, W_1)+b_1)
h2 = tf.sigmoid(tf.matmul(h1, W_2)+b_2)
y = tf.matmul(h2, W_o, name='output')
```

機器學習 Machine Learning

尋找合理的推論⟶尋找參數



- 資料:訓練,[驗證].測試
- 架構 = 搜尋範圍
- 目標函數 = 搜尋偏好

定義目標函數/減損函數

- 設計一個減損函數評估參數優劣
- Mean Squared Error

$$L = E\left[\frac{1}{2}|y_i - \hat{y}_i|^2\right]$$

- 分類問題: Cross-Entropy

$$L = -E\left[\hat{y}_i \log(y_i)\right]$$

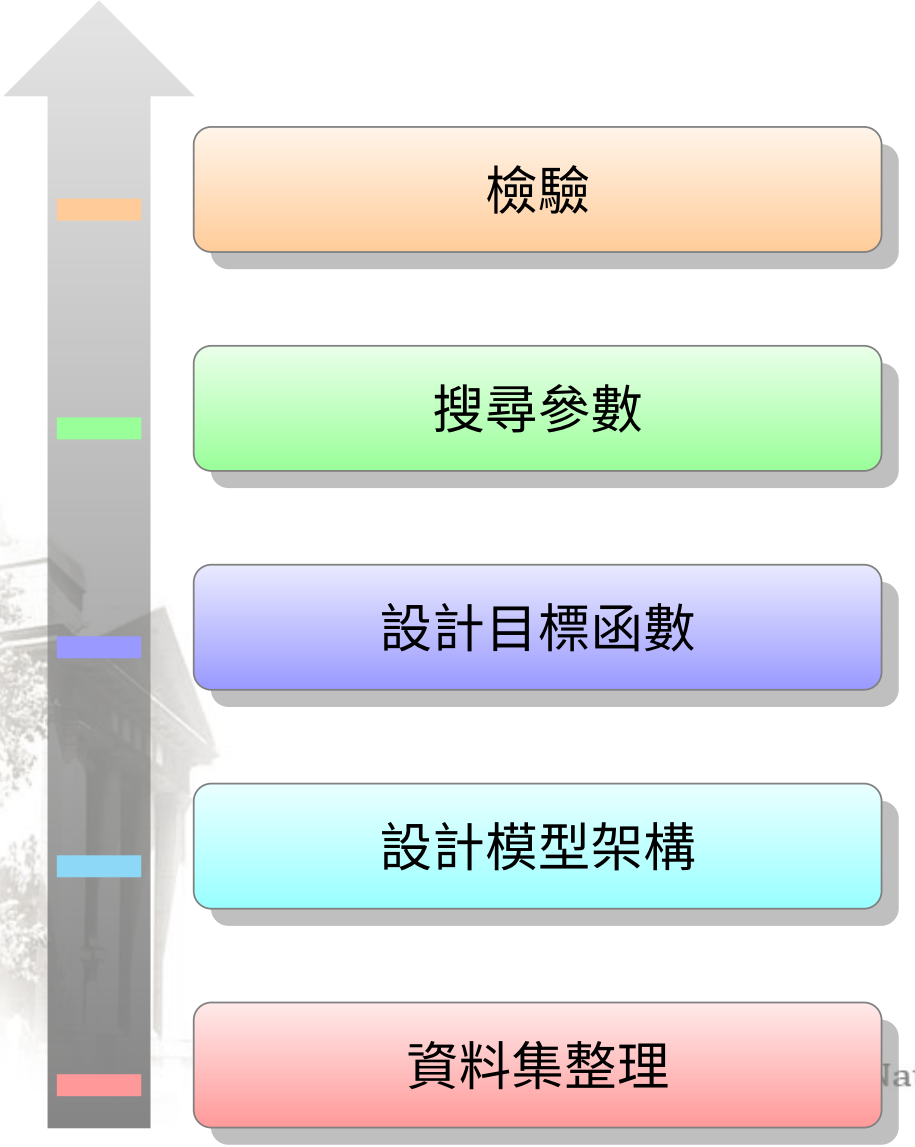
Mean Squared Error

- tf.reduce_mean 計算所有元素的平均值
- 和 np.mean 一樣，也可以指定 axis
- 檢查模型輸出 y 和正確答案 y_true 的維度一致

```
y_true = tf.placeholder(dtype=tf.float32,
                        shape=(None, 1))
loss = tf.reduce_mean((0.5)*tf.pow(y-y_true, 2))
# Check y and y_true
print (y.shape, y_true.shape)
```

機器學習 Machine Learning

尋找合理的推論⟶尋找參數



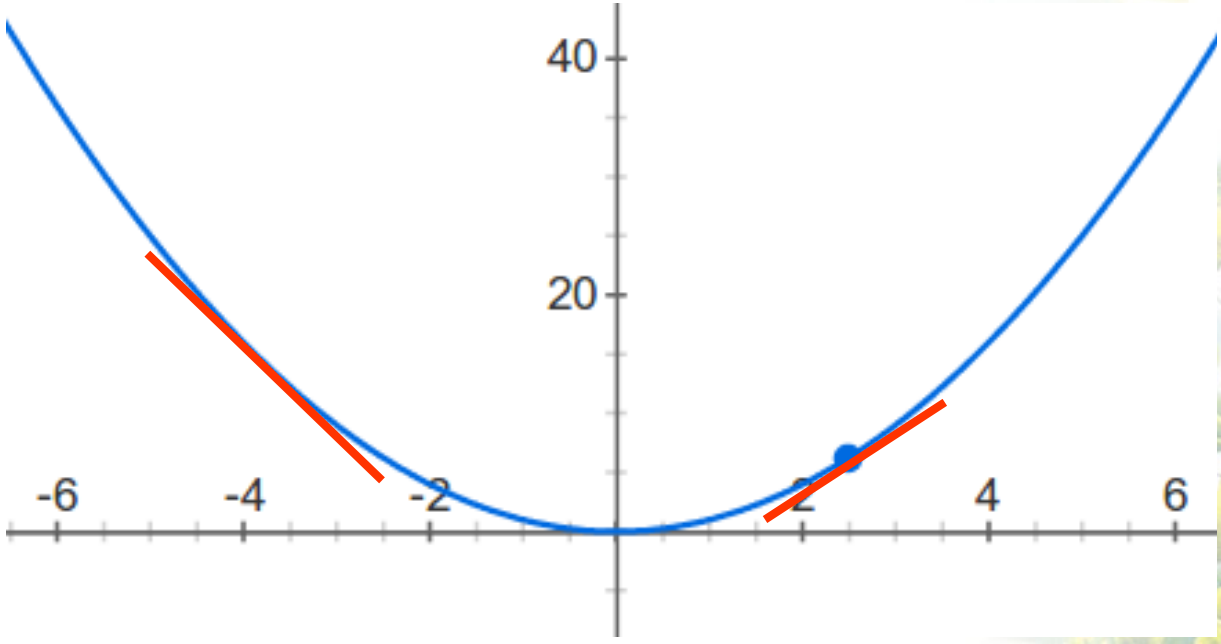
- 資料:訓練,[驗證].測試
- 架構 = 搜尋範圍
- 目標函數 = 搜尋偏好
- 從資料學習參數

調整參數

- 利用梯度下降法(Gradient Descent)調整參數
- 往切線方向移動會上升，反之下降

$$\hat{\theta} = \theta - \eta \frac{\partial L}{\partial \theta}$$

- η 控制移動步伐



學習步伐調整

- 隨著更新次數慢慢遞減

$$\text{learning\_rate} * \text{decay\_rate}^{(\text{global\_step} / \text{decay\_steps})}$$

```
batch_size, n_samples = 4,  x_train.shape[0]
decay_steps = n_samples / batch_size
global_step = tf.Variable(0, trainable=False)
learning_rate = tf.train.exponential_decay(0.02,
                                           global_step, decay_steps,
                                           decay_rate=0.5)
```

相當於每個 epoch 減半 η

加入慣性

- Momentum

$$v_t(\theta) = \gamma v_{t-1}(\theta) + \frac{\partial L}{\partial \theta}$$

$$\hat{\theta} = \theta - \eta v_t(\theta)$$

- 如果設計得當，SGD + Momentum 會是首選
- 要嘗試不同 η 設計

```
optimizer = tf.train.MomentumOptimizer(learning_rate,
                                       momentum=0.9)
train_step = optimizer.minimize(loss, global_step=
                                global_step)
```

模型儲存 Saver

- 以上程式碼包含所有運算
- 開時運算前，用 tf.train.Saver 紀錄所有設定
- 一定要先設計好所有運算再呼叫 tf.train.Saver
  - 預設記憶所有 tensorflow 變數

```
saver = tf.train.Saver()
```

Session

- 所有的運算都要先建立 Session
- Session 可以決定運算的資源和 Log 訊息紀錄等

```
sess_config = tf.ConfigProto()
with tf.Session(config=sess_config) as sess:
  # ...
```

- ”allow_soft_placement” 允許自動替換裝置
- ”log_device_placement” 留下紀錄訊息

初始化參數

- sess.run() 用來執行運算
- sess.run(取出的變數,feed_dict=輸入的參數)
- 第一步是初始化參數（重要）

```
with tf.Session(config=sess_config) as sess:
  sess.run(tf.global_variables_initializer())
  # ...
```

Mini-Batch Gradient Descent

- Mini-Batch：一次只用少量的資料
- Stochastic Gradient Descent：把資料順序打亂

```
with tf.Session(config=sess_config) as sess:
  sess.run(tf.global_variables_initializer())
  idx = np.arange(n_samples) # 0, 1, 2,... n_sample -1
  for epoch in range(1,16): # 1 ~ 15
  np.random.shuffle(idx)
    for beg in range (0, n_samples, batch_size):
      x_batch=x_train[idx[beg:beg+batch_size,:]],
      y_batch=y_train[idx[beg:beg+batch_size,:]]
      sess.run(train_step,
                feed_dict={x: x_batch,y_true: y_batch})
```

觀察 Loss 變化

- sess.run 計算 loss

```
with tf.Session(config=sess_config) as sess:
  # ...
  for epoch in range(1,16): # 1 ~ 15
    for beg in range (0, n_samples, batch):
      # ...
    tr_loss = sess.run(
      loss,feed_dict={x: x_train, y_true: y_train})
    ts_loss = sess.run(
      loss, feed_dict={x: x_test, y_true: y_test})
    print("epoch %02d: tr_loss=%f, ts_loss=%f"
      % (epoch, tr_loss, ts_loss))
```

別忘記保存參數

- 最後一個 epoch 不一定是最佳結果

```
with tf.Session(config=sess_config) as sess:
  # ...
  for epoch in range(1,16): # 1 ~ 15
    for beg in range (0, n_samples, batch):
      # ...
    # ...
    print("epoch %02d: tr_loss=%f, ts_loss=%f"
      % (epoch, tr_loss, ts_loss))
    saver.save(sess, "Model/bike.epoch_%02d" % (epoch))
```

實際觀察結果 - 讀取最佳參數

- 假如 epoch 14 是最佳結果
- 利用 saver.restore 還原
- 輸入測試資料，取出 y 的結果

```
with tf.Session() as sess:
  saver.restore(sess, "Model/bike.epoch_014")
  y_pred = sess.run(y, feed_dict={x: x_test, y_true:
                                  y_test})
```

實際觀察結果 - 還原

- 資料有正規化處理，需要還原才知道單車數量

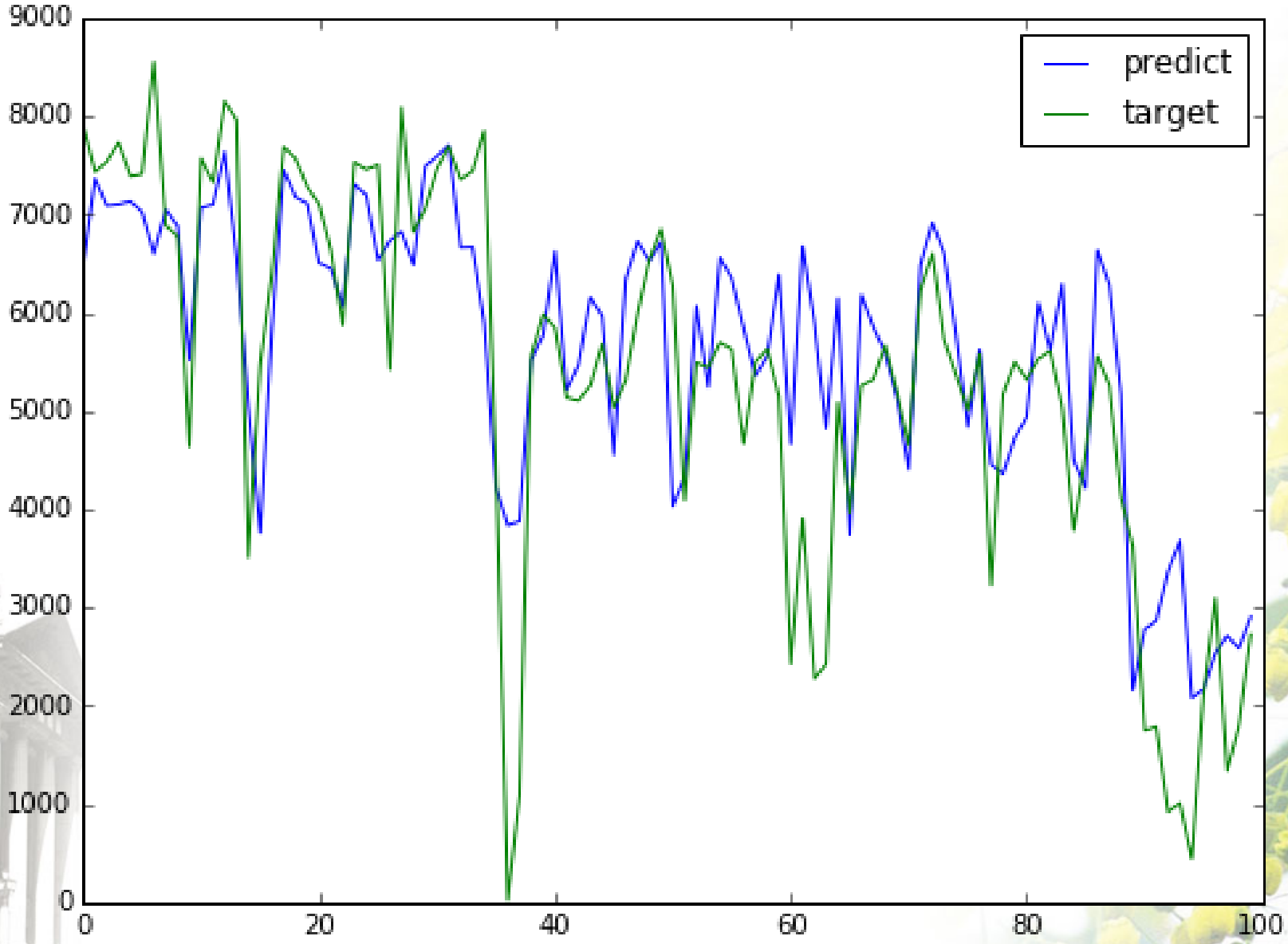
```
with tf.Session() as sess:
  saver.restore(sess, "Model/bike.epoch_014")
  y_pred = sess.run(y, feed_dict={x: x_test, y_target:
                                  y_test})

cnt_pred = y_pred * stdv[-1] + mean[-1]
cnt_true = y_test * stdv[-1] + mean[-1]
```

實際觀察結果 - 繪圖

- 以 matplotlib 為範例，可以用任何熟悉的方法

```
import matplotlib.pyplot as plt
day = range(100)
plt.plot(day, cnt_pred, label='predict')
plt.hold(True)
plt.plot(day, cnt_true, label='target')
plt.legend()
plt.show()
plt.savefig('BikeSharing.png')
```

還不錯！


predict
target
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100

Outline

深層學習的基本問題

- sigmoid 的導函數存在極值 0.25

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$$

- 許多新想法是考量 ”error flow”[4]

$$\frac{\partial e(W_L)}{\partial e(W_{L+1})}$$

- 流量, 流向

克服流量問題

- constant error flow

$$\frac{\partial e(W_L)}{\partial e(W_{L+1})} = 1$$

- Rectified Linear Unit

$$f(x) = \max(0, x)$$

- short cut:
  - 平滑化：$\mathbf{h}_t = \mathbf{h}_{t-1} + \tilde{\mathbf{h}}_t$
  - 殘差 (Residual Learning)：$x + g(\mathbf{x})$

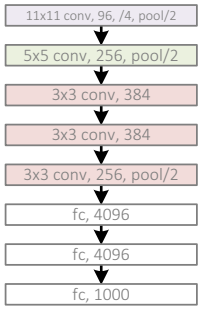
ICML 2016 tutorial: Deep ResNet

speaker: Kaiming He

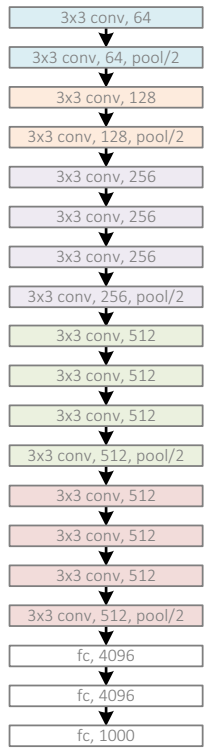
http://techtalks.tv/talks/deep-residual-networks-deep-learning-gets-way-deeper/62358/

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)

ICML 2016 tutorial: Deep ResNet

speaker: Kaiming He

http://techtalks.tv/talks/deep-residual-networks-deep-learning-gets-way-deeper/62358/

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

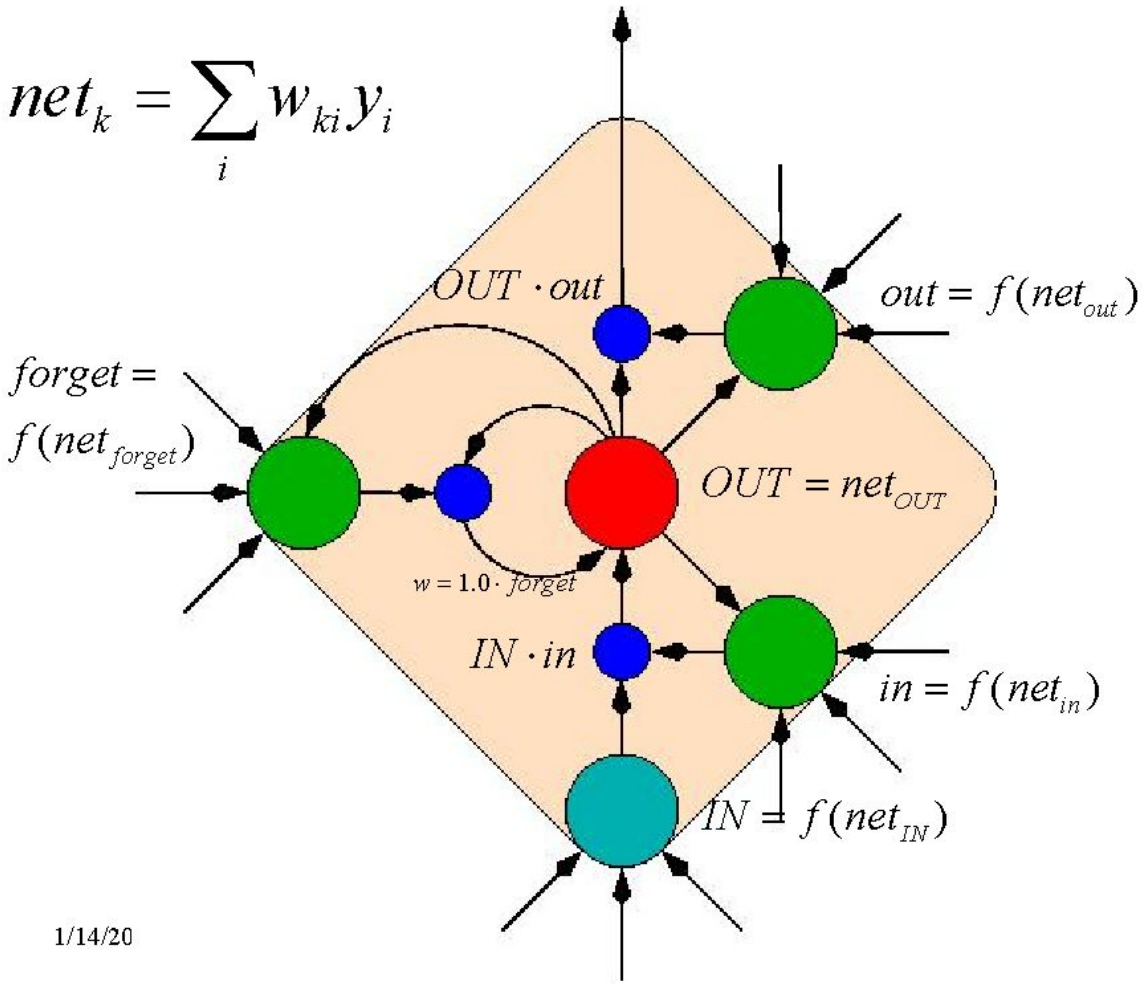
ResNet, 152 layers
(ILSVRC 2015)

流向問題

- 只要還有錯誤，所有參數都會調整
- 矛盾：穩定，可塑性
- ANN領域：Adaptive Resonance Theory
- Deep learning：資料選擇，保護沒使用的資料
- 例如：注意力機制（Attention Mechanism），閘門(Gate)

Long Short-Term Memory

當 Gate = 0，關閉部份網路

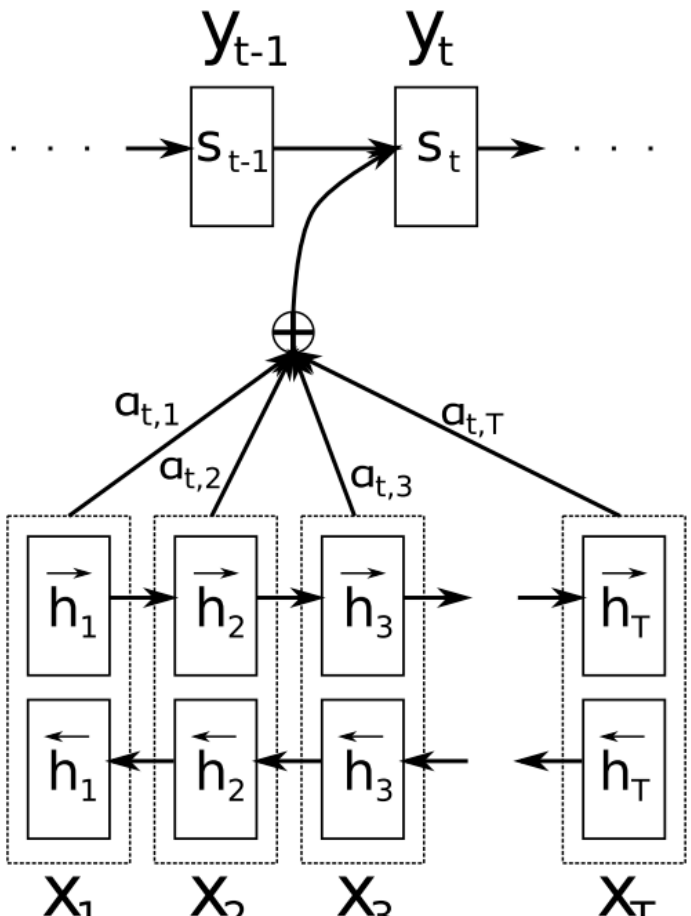


[1]http://people.idsia.ch/ juergen/lstm

Attention Mechanism

機器翻譯[5]

利用神經元學習權重 a, 將 T 個資訊結合



為何一定要認識Keras?

Tensorflow 的問題：

- Tensorflow 是從基礎運算開始建立
- 高階和低階函式混雜在一起，且不能混用
- 太多程式細節（全域，區域變數，命名範圍...）
- 缺乏統一風格（同一件事有多種寫法）
- Python 程式應保持簡潔

已經講得很累，但仍不足 <(_ _)>

Keras 簡介

http://keras.io

- 生於 Tensorflow 幼兒期
- 站在 Theano 和 TF 的肩膀上
- 手冊精簡，明確
- Google 修改TF，迎合 Keras 的程式界面
- 雖然是高階函式庫，但具備彈性

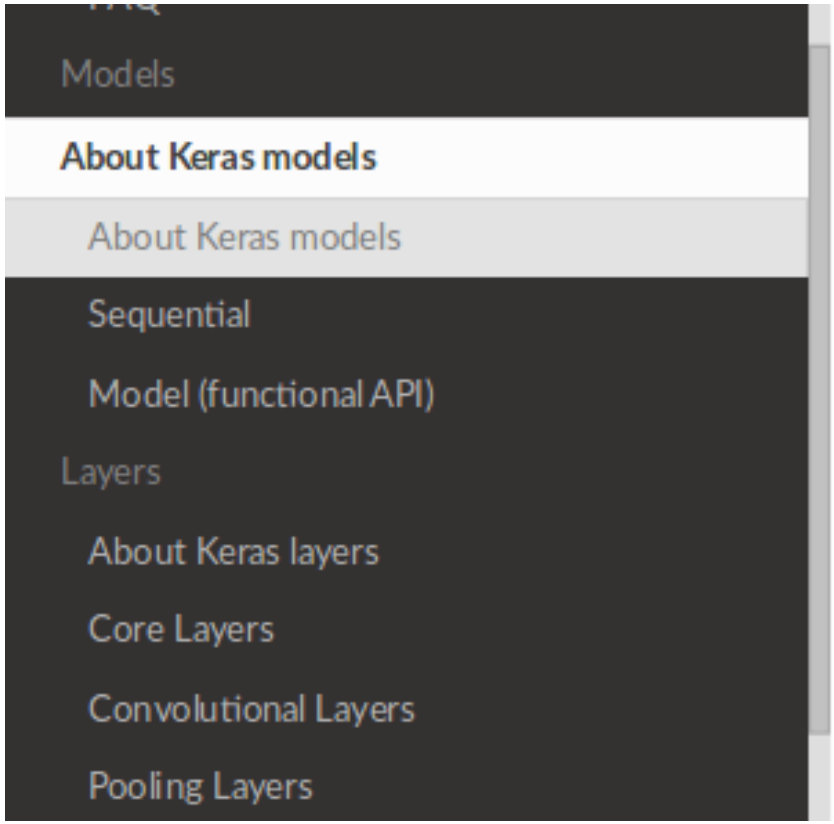
設計方法一：Sequential

- 單車租借為例：

```
from keras.models import Sequential
# ...

model = Sequential([
  Dense(16, activation='sigmoid', input_shape=(11,)),
  Dense(16, activation='sigmoid'),
  Dense(1),])
drate = 4.0/x_train.shape[0]
sgd = optimizers.SGD(lr=0.2, decay=drate, momentum=0.9)
model.compile(optimizer=sgd, loss='mean_squared_error')
model.fit(x_train, y_train, epochs=15, batch_size=4)
```

說明

- Keras元件的命名和手冊階層一致
- 可以自由替換



說明

Sequential：

- 第一層一定要指定輸入資料維度
- input_shape 不用包含Batch Size
- 如果輸入資料不多，直接存放 (資料數, 特徵維度)
- model.fit 可以自動切割 batch

Optimizer：

- SGD需要指定參數，所以特別宣告
- Adam 和 Adadelta 可以不指定參數
- 使用預設參數： optimizer=’ Adam’

說明

Loss function：

- mean_squared_error 數值預測
- categorical_crossentropy 分類問題(softmax)
- binary_crossentropy 兩類分類(sigmoid)

Model Compile

- 建立計算圖

參數訓練

- ”fit”是自動化訓練
- 過程會自動顯示訊息
- 當資料太大無法一次讀取，改用 fit_generator
- Callbacks 是訓練過程中可以執行的動作
  - 例如 ModelCheckpoint 保存每個 epoch 的參數
  - ModelCheckpoint(’epoch_{epoch:02d}.hdf5’)

```
model.fit(x_train , y_train , epochs =15 , batch_size =4, callb
```

```
Using TensorFlow backend.
```

```
Epoch 1/15
631/631 [==============================] - 0s - loss: 1.6569
Epoch 2/15
631/631 [==============================] - 0s - loss: 1.1511
Epoch 3/15
631/631 [==============================] - 0s - loss: 0.3538
Epoch 4/15
631/631 [==============================] - 0s - loss: 0.2598
Epoch 5/15
631/631 [==============================] - 0s - loss: 0.2306
Epoch 6/15
631/631 [==============================] - 0s - loss: 0.1670
Epoch 7/15
631/631 [==============================] - 0s - loss: 0.1469
```

保存，讀取

- Keras 將模型架構和參數分開存放
- save_to_json 和 save_to_yaml 可以輸出字串
- 用任何方法紀錄字串即可
- model_from_json 可以回覆架構
- 可以用model.summary() 檢查架構是否正確

model.summary()

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 16)                192
_________________________________________________________________
dense_2 (Dense)              (None, 16)                272
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 481
Trainable params: 481
Non-trainable params: 0
```

設計方法二：functional API

- 每個元件要指定輸入
- 可以合併多個輸入
- 可以重複利用(共享參數)

```
from keras.models import Model
from keras.layers import Input, Dense
i1 = Input(shape=(5,))
i2 = Input(shape=(5,))
b = Dense(32)(i1,i2)
shared_c = Dense(32)
c1 = shared_c(i1)
c2 = shared_c(i2)
model = Model(inputs=[i1,i2], outputs=[b,c1,c2])
```

Outline

建立舒適的工作環境

硬體設備

- 遊戲卡 GPU 相當便宜
- 文字，聲音，影片序列資料會需要記憶體
- 一張卡只能一個工作
- 多張卡買同規格
- 噪音，熱氣

建立舒適的工作環境

軟體環境

- 推薦用 Jupyter-Notebook 學習
  - 用網頁開啟，也可以傳檔案，開啟終端機...
- 伺服器用 Linux (學生不會偷玩 GAME)
- 個人電腦可以是任何 OS
- 進階：用 Docker 維護 Tensorflow

Outline

總結

- 深度學習是開放的舞台
  - 學術研究不限系所
  - 職場工作需要不一樣的人才
- 入行門檻不在設備、大數據
- 進行AI工作：Keras $\rightarrow$ Tensorflow
- 流程不相似，但需要高速矩陣運算：Tensorflow

工商服務時間

- 台師大暑期AI課程 https://hpc.ntnu.edu.tw
- 課程時間 8/21 9/8 (三週)
- 週一到週五下午

專題發表

分組，合作

進階深度學習

了解原則，動手設計

基礎功夫

Tensorflow, Keras

多媒體應用

語音，文字，影像

大數據處理

Spark, Pandas,...

Outline

1. 建立基礎觀念
   - Deep Learning 簡介，觀念釐清
2. 機器學習流程
   - NumPy+Tensorflow
3. 深層學習的特色
   - Keras
4. 建立舒適的工作環境
5. 結語
6. 16:00-18:00 Tutorial session

Reference I

- G. Cybenko, "Approximation by superpositions of a sigmoidal function," Mathematics of Control, Signals and Systems, vol. 2, no. 4, pp. 303–314, Dec. 1989.
- V. Vapnik, Statistical learning theory. Wiley New York, 1998, vol. 1.
- J. Schmidhuber, "Discovering neural nets with low kolmogorov complexity and high generalization capability," Neural Networks, vol. 10, no. 5, pp. 857–873, 1997.
- J. F. Kolen and S. C. Kremer, "Gradient flow in recurrent nets: The difficulty of learning longterm dependencies," in A Field Guide to Dynamical Recurrent Networks. Wiley-IEEE Press, 2001, pp. 237–243.
- D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," ArXiv preprint arXiv:1409.0473, 2014.