TensorFlow Tutorial

Ying-Ting Wang

**Introduction**

TensorFlow 是 google 開發神經網路的 python 結構包

用 TensorFlow 可快速入門 神經網路

**Session**

we can use sess.run() to point to a part in the picture and only run that part.

Session 在程式中是執行命令的控制，Session為一個object，所以在使用時S必須大寫 : tf.Session()；且在執行run()時，Tensorflow才會執行呼叫的結構，這是Tensorflow的思考模式。

“Session” example code:

>>> matrix1 = tf.constant([[3,3]])

>>> matrix2 = tf.constant([[2],

                                [2]])

>>> product = tf.matmul (matrix1, matrix2)    # matrix multiply

>>> #method 1

>>> # 執行 tf.Session() 並可以使用 sess 作為呼叫的縮寫

>>> sess = tf.Session( )

>>> result = sess.run(product)

>>> print(result)

>>> sess.close( )

>>> #method 2

>>> # 使用“with”語句會在包含在with範圍內執行完後，自動做 sess.close()

>>> with tf.Session( ) as sess:

>>>     result2 = sess.run(product)

>>>     print(result2)

**Variable**

宣告 variable 語法

>>> state = tf.Variable(0, name='counter')

上面的式子，指定 state = 0，且給他名字 "counter"

如果有宣告過 variable，必須要 "初始化" 以及執行 "session run" 才能真正產生該變數

>>> # init 針對程式中所有變數初始化

>>> init = tf.initialize\_all\_variables( )

>>> with tf.Session() as sess:

>>> sess.run(init)

>>> print(sess.run(state))

要印出值時不能直接印 (print(state))，仍必須透過 sess.run 執行印出該值

**placeholder**

placeholder 傳入值 (在 sess.run() 時以 "feed\_dict" 的形式傳入值)

“placeholder” example code:

import tensorflow as tf

input1 = tf.placeholder(tf.float32)

input2 = tf.placeholder(tf.float32)

output = tf.mul(input1, input2) # 執行乘法 (multiply) input1 x input2

with tf.Session( ) as sess:

     print(sess.run(output, feed\_dict={input1: [7.], input2: [2.]} ))

在 sess.run() 時，以 "feed\_dict" 的形式傳入值

**activation function 激勵函數**

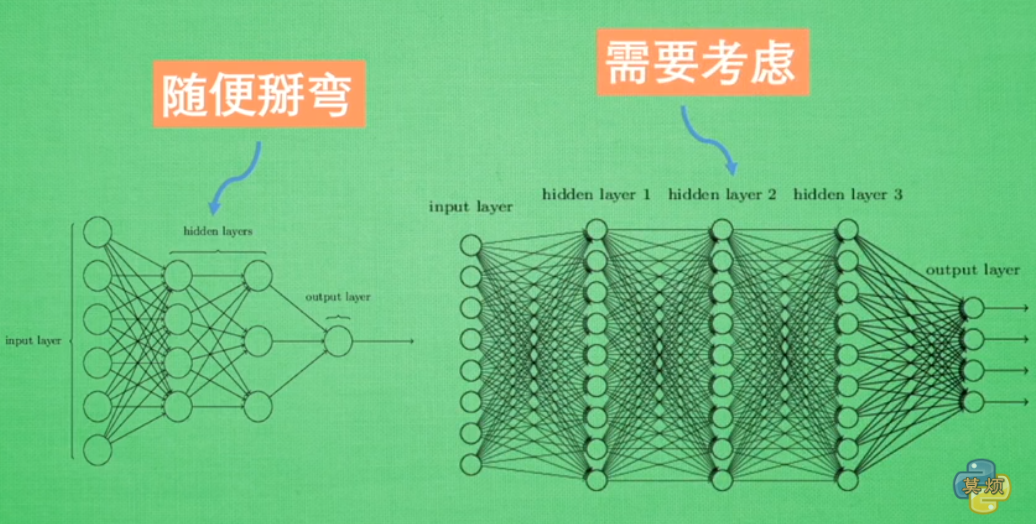
\*\*\* 解決線性方程"不能"解決的問題 \*\*\*

\*\*\* activation function 為 Non-linear function \*\*\*

activation function必須是可微分的，因為在反向誤差傳遞時，只有可微分的activation function才能將誤差傳遞回去

<https://www.youtube.com/watch?v=tI9AbaBfnPc&index=10&list=PLXO45tsB95cKI5AIlf5TxxFPzb-0zeVZ8>

使用 activation function 時機 :



當神經網路複雜時，使用 activation function 可能會有梯度爆炸、梯度消失的問題

activation function 意義 :

激活某一部分的神經元，經過訓練不斷調整神經元，使得神經元對真正重要的訊息敏感，使之能正確判斷 (inference)

tensorflow官網 activation function頁面

<https://www.tensorflow.org/versions/r0.11/api_docs/python/nn/>

**自行定義添加層 (def add\_layer)**

# 以下為添加層的 sample code

import tensorflow as tf

import numpy as np

# 定義此function是否有activation function傳入

def add\_layer(inputs, in\_size, out\_size, activation\_function=None):

     Weights = tf.Variable(tf.random\_normal([insize, out\_size]))

     biases = tf.Variable(tf.zeros[1, out\_size]) + 0.1)

     Wx\_plus\_b = tf.matmul(inputs, Weights) + biases

     if(activation\_function is None):

          outputs = Wx\_plus\_b

     else:

          outputs = activation\_function(Wx\_plus\_b)

     return outputs

x\_data = np.linspace(-1, 1, 300)[:, np.newaxis]

noise = np.random.normal(0, 0.05, x\_data.shape)

y\_data = np.square(x\_data) - 0.5 + noise

# xs, ys 裡 None 的意思是要餵幾個參數都可以

xs = tf.placeholder(tf.float32, [None, 1])

ys = tf.placeholder(tf.float32, [None, 1])

l1 = add\_layer(xs, 1, 10, activation\_function=tf.nn.relu)

prediction = add\_layer(l1, 10, 1, activation\_function=None)

loss = tf.reduce\_mean(tf.reduce\_sum(tf.square(ys - prediction), reduction\_indices=[1]))

train\_step = tf.train.GradientDescentOptimizer(0.1).minimize(loss)

init = tf.initialize\_all\_variables()

sess = tf.Session()

sess.run(init)

for i in range(1000):

     sess.run(train\_step, feed\_dict={xs: x\_data, ys: y\_data})

     if (i % 50 == 0):

          # 所有通過 placeholder 做計算的，都必須使用 feed\_dict 帶參數

          print(sess.run(loss, feed\_dict={xs: x\_data, ys:y\_data}))