機器學習於材料資訊的應用
Machine Learning on Material Informatics

陳南佑(NAN-YOW CHEN)

nanyow@narlabs.org.tw

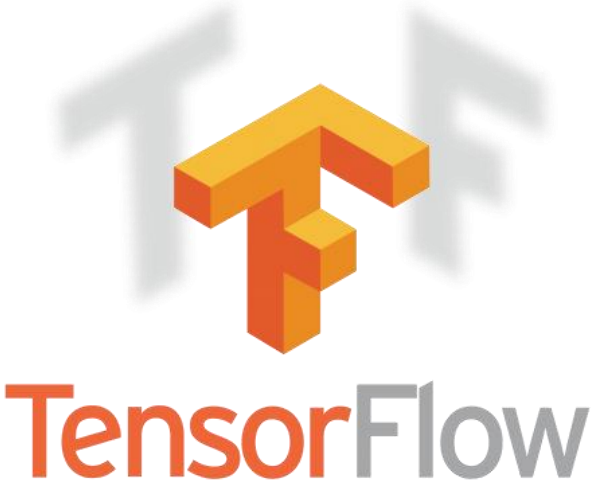
楊安正(AN-CHENG YANG)

acyang@narlabs.org.tw

Framework For Machine Learning

提供
Machine Learning演算法
資料處理相關函式
特徵工程相關函式

提供高階的API幫助用戶實現許多常用神經網路構建模塊，例如layers, objectives, activation functions, optimizers。
支援的後端包含了TensorFlow、CNTK、R、Theano or PlaidML



TensorFlow 是一個廣為流行的機器學習開源軟體庫(框架)，透過Computational Graph，來進行行數值演算。

Installation of tensorflow

1. conda activate mpp
2. conda install tensorflow tensorboard keras

(replace tensorflow with tensorflow-gpu, if you have gpu HW)

What is tensor?

- TensorFlow 的基本運算單位是張量（Tensor），零維張量等於是純量(Scalar)，一維張量是向量(Vector)，二維張量是矩陣(Matrix)等等。
- 和Numpy相對比，ndarray（n 維陣列）觀念與 Tensor（n 維張量）觀念類似以外，TensorFlow 函數的命名、參數與設計概念都很接近 NumPy。

What is tensor?

NUMPY

```
one_d_arr = np.arange(24)

two_d_arr = np.arange(24).reshape(6, 4)

three_d_arr = np.arange(24).reshape(2, 3, 4)
```

TENSORFLOW

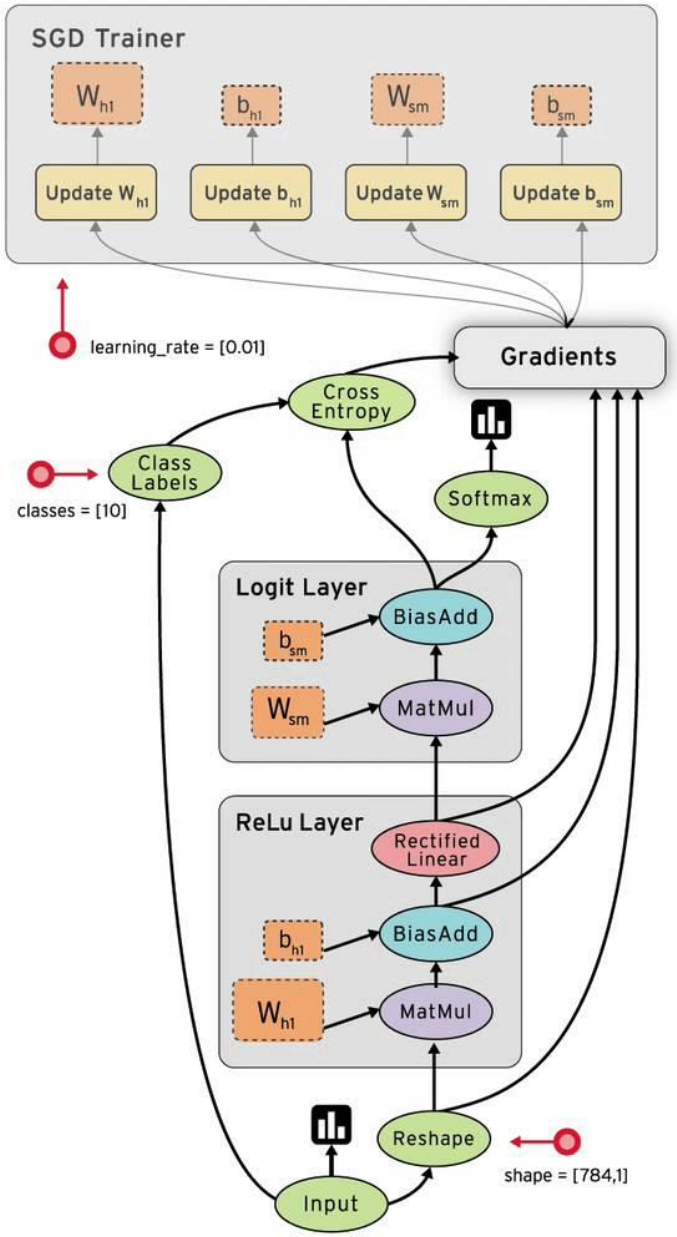
```
one_d_tensor = tf.constant(np.arange(24))

two_d_tensor = tf.constant(np.arange(24), shape=(6, 4))

three_d_tensor = tf.constant(np.arange(24), shape=(2, 3, 4))
```

What is Flow?

- Tensorflow主要的運作流程分為以下兩個部分
  - 建立模型(Build Model)
  - 執行運算(Run)
- Tensorflow設計的核心就是Tensor的流動，建立Graph的過程其實只是定義好Tensor如何流動並運算的過程，但真正的資料其實並沒有被運算，真正的計算需要用session來執行。



Hello World in Tensorflow

```
import tensorflow as tf

hw = tf.constant("Hello World")
with tf.Session() as sess:
    print(sess.run(hw))
```

matrix multiplication in tensorflow

```
import tensorflow as tf

# Build a dataflow graph.
c = tf.constant([[1.0, 2.0], [3.0, 4.0]])
d = tf.constant([[1.0, 1.0], [0.0, 1.0]])
e = tf.matmul(c, d)

# Construct a `Session` to execute the graph.
with tf.Session() as sess:
  # Execute the graph and store the value that `e` represents in `result`.
  result = sess.run(e)

print(result)
```

Tensor的類型

- Tensor可以被宣告為
  - 常數（Constant）
  - 變數（Variable）
  - 佔位符(Placeholder)

常數（Constant）

PYTHON

```
print("Python:")
A = 19
B = 3
```

TENSORFLOW

```
print("TensorFlow:")


x = tf.constant(19)
y = tf.constant(3)
```

數值運算

- Tensor數值運算這些函數都必須在 TensorFlow 的 Session 中執行才會有運算結果的輸出，否則只是顯示張量物件的資訊而已。
  - 加 + : tf.add()
  - 減 - : tf.sub()
  - 乘 * : tf.multiply()
  - 除 / : tf.divide()
  - 次方 ** : tf.pow()
  - 求餘數 % : tf.mod()
  - 求商數 // : tf.div()

數值運算

```
x = tf.constant(19)
y = tf.constant(3)

print("TensorFlow:")
print( tf.add(x, y) )
print( tf.subtract(x, y) )
print( tf.multiply(x, y) )
print( tf.divide(x, y) )
print( tf.pow(x, y) )
print( tf.mod(x, y) )
print( tf.div(x, y) )
```

```
TensorFlow:
Tensor("Add:0", shape=(), dtype=int32)
Tensor("Sub:0", shape=(), dtype=int32)
Tensor("Mul:0", shape=(), dtype=int32)
Tensor("truediv:0", shape=(), dtype=float64)
Tensor("Pow:0", shape=(), dtype=int32)
Tensor("FloorMod:0", shape=(), dtype=int32)
WARNING:tensorflow:From <ipython-input-5-47e0c147644b>:11: div (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Deprecated in favor of operator or tf.math.divide.
Tensor("div:0", shape=(), dtype=int32)
```

Tensor 運算

- 一個 Tensor的運算需要三個步驟：
  - 宣告張量
  - 使用張量的運算公式
  - 在 Session 裡執行運算

Tensor 運算

```
x = tf.constant(19)
y = tf.constant(3)

with tf.Session() as sess:
    print( sess.run(tf.add(x, y)) )
    print( sess.run(tf.subtract(x, y)) )
    print( sess.run(tf.multiply(x, y)) )
    print( sess.run(tf.divide(x, y)) )
    print( sess.run(tf.pow(x, y)) )
    print( sess.run(tf.mod(x, y)) )
    print( sess.run(tf.div(x, y)) )
```

```
22
16
57
6.333333333333333
6859
1
6
```

常數張量建構函數

- tf.zeros() ：建構內容數值皆為 0 的常數張量
- tf.ones() ：建構內容數值皆為 1 的常數張量
- tf.fill() ：建構內容數值皆為特定值的常數張量
- tf.range() ：建構內容數值為 (start, limit, delta) 數列的常數張量
- tf.random_normal() ：建構內容數值為符合常態分佈數列的常數張量
- tf.random_uniform() ：建構內容數值為符合均勻分佈數列的常數張量

矩陣運算函數

- tf.reshape() ：調整矩陣外觀
- tf.eye() ：建構單位矩陣
- tf.diag() ：建構對角矩陣
- tf.matrix_transpose() ：轉置矩陣
- tf.matmul() ：矩陣相乘

變數（Variable）

- 程式設計中為了保持彈性，必須將值賦與給變數（Variables），讓使用者能夠動態地進行相同的計算來得到不同的結果，這在 TensorFlow 中是以 tf.Variable() 來完成。
- 但宣告變數張量並不如 Python 或者先前宣告常數張量那麼單純，它需要兩個步驟：
  - 宣告變數張量的初始值、類型與外觀
  - 初始化變數張量

初始化變數

未初始化

```
# TensorFlow:
FailedPreconditionError
var_tf = tf.Variable(47)
print("TensorFlow:")
with tf.Session() as sess:
    print(sess.run(var_tf))
```

初始化

```
var_tf = tf.Variable(47)
print("TensorFlow:")
with tf.Session() as sess:
    sess.run(var_tf.initializer)
    print(sess.run(var_tf))
```

重新賦值

- 初始化成功後的變數張量，可以透過 .assign() 方法重新賦予不同值。
- 重新賦值這件事對 TensorFlow 來說也是一個運算，必須在宣告之後放入 Session 中執行，否則重新賦值並不會有作用。
- 重新賦值時必須要注意類型，賦予不同類型的值會得到 TypeError。
- 不僅是值的類型，外觀也必須跟當初所宣告的相同，賦予不同外觀的值會得到 ValueError。

重新賦值

```
var_tf = tf.Variable(47)
assign_op = var_tf.assign(24)

print("TensorFlow:")
with tf.Session() as sess:
    sess.run(var_tf.initializer)
    print('Before assign',sess.run(var_tf))
    #sess.run(assign_op)
    print('After assign',sess.run(var_tf))
```

佔位符(Placeholder)

- 在Tensorflow中我們都是先建好Graph再決定資料的input與output，在上面的例子由於我們已經先定義好constant Tensor所以並不需要給Graph任何Input我們就可以得到各種張量運算的output，但如果我們操作的不是常數張量呢?

- 這時候我們就需要Placeholder來幫助我們在還沒有資料的時候先佔個位子，這是一種常見將資料輸入 TensorFlow 計算圖形（Graph）的方法。

佔位符(Placeholder)

- Placeholder 張量和變數張量一樣，必須預先定義好之後欲輸入的資料類型與外觀。使用 tf.placeholder() 可以建出 Placeholder 張量。

- 宣告完 Placeholder 以後，TensorFlow 資料輸入 placeholder的術語稱作是 Feed dictionaries，顧名思義就是將資料以 Python dictionaries 餵進（Feed）Placeholder 張量之中。

佔位符(Placeholder)

```
p1 = tf.placeholder(tf.float32, shape=(4,))
p2 = tf.placeholder(tf.float32, shape=(4,))


print(p1,p2)
add_op = p1 + p2
print(add_op)
with tf.Session() as sess:
    adder=sess.run(add_op, feed_dict={p1: [7, 14, 21, 28], p2: [5, 10,
15, 20]})
    print(adder)
```

Python 的 with 使用方法

```
# 開啟檔案
f = open(filename)


# ...


# 關閉檔案
f.close()
```

使用檔案的過程中發生了一些例外狀況，造成程式提早跳開時，這個開啟的檔案就會沒有被關閉

```
# 開啟檔案
f = open(filename)


try:
    # ...

finally:
    # 關閉檔案
    f.close()
```

正規寫法雖然不會有問題，但是缺點就是必須手動加入關閉檔案的程式碼，不是很方便，也很容易忘記。

Python 的 with 使用方法

```
# 以 with 開檔並寫入檔案
with open('file.txt', 'w') as f:
  f.write('Hello, world!')
```

使用 with 的話，檔案使用完之後就會自動關閉

with 開啟檔案時，會將開啟的檔案一樣放在 f 變數中，但是這個 f 只有在這個 with 的範圍內可以使用，而離開這個範圍時 f 就會自動被關閉，回收相關的資源。

with 這個獨特的語法，可以來幫助使用者管理使用的資源，像是開啟的檔案、網路 socket，包含tensorflow的 session。