Chapter 2 Tensorflow 基本語法

討論TensorFlow 基本操作以及常見資料處理手法

In [ ]:

```
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
```

In [2]:

```
import tensorflow as tf
tf.__version__
```

Out[2]:

```
'2.0.0'
```

2-1 變數類型

TensorFlow 支持以下三种类型的张量：

1. 常量(tf.constant)：常量是其值不能改变的张量。
2. 变量(tf.Variable)：当一个量在会话中的值需要更新时，使用变量来表示。例如，在神经网络中，权重需要在训练期间更新，可以通过将权重声明为变量来实现。变量在使用前需要被显示初始化。另外需要注意的是，常量存储在计算图的定义中，每次加载图时都会加载相关变量。换句话说，它们是占用内存的。另一方面，变量又是分开存储的。它们可以存储在参数服务器上。
3. 占位符(tf.placeholder)：用于将值输入 TensorFlow 图中。它们可以和 feed_dict 一起使用来输入数据。在训练神经网络时，它们通常用于提供新的训练样本。在会话中运行计算图时，可以为占位符赋值。这样在构建一个计算图时不需要真正地输入数据。需要注意的是，占位符不包含任何数据，因此不需要初始化它们。

tf.constant

放入各式資料型態，執行基礎運算

聲明一個常量：

In [6]:

```
t_1 = tf.constant(4)
print("t_1=" , t_1)
```

```
t_1= tf.Tensor(4, shape=(), dtype=int32)
```

一個形如[1，3] 的常量向量可以用如下代碼聲明：

In [9]:

```
t_2= tf.constant([1, 3])
print("t_2=" , t_2)
```

```
t_2= tf.Tensor([1 3], shape=(2,), dtype=int32)
```

In [3]:

```
tf.constant(6)
```

Out[3]:

```
<tf.Tensor: shape=(), dtype=int32, numpy=6>
```

In [4]:

```
tf.constant(6.)
```

Out[4]:

```
<tf.Tensor: shape=(), dtype=float32, numpy=6.0>
```

In [5]:

```
tf.constant([True,False])
```

Out[5]:

```
<tf.Tensor: shape=(2,), dtype=bool, numpy=array([ True, False])>
```

In [6]:

```
tf.constant('I love tf2!!!')
```

Out[6]:

```
<tf.Tensor: shape=(), dtype=string, numpy=b'I love tf2!!!'>
```

In [7]:

```
import numpy as np
```

In [8]:

```
#Create tf.constant
a = tf.constant([1,2,3])
#Convert np arry to tf
a = np.array([1,2,3])
b = tf.convert_to_tensor(a)
```

In [9]:

```
b
```

Out[9]:

```
<tf.Tensor: shape=(3,), dtype=int64, numpy=array([1, 2, 3])>
```

tf.Variable

放置可學習變數或可求導的變數

要創建一個所有元素為零的張量，可以使用tf.zeros() 函數。這個語句可以創建一個形如[M，N] 的零元素矩陣，數據類型（dtype）可以是int32、float32

tf.zeros([M,N],tf.dtype)

例如

In [10]:

```
zero_t = tf.zeros([2,3],tf.int32)
# Results in an 2x3 array of zeros:[[0 0 0],[0 0 0]]
print("zero_t=" , zero_t)
```

```
zero_t= tf.Tensor(
[[0 0 0]
 [0 0 0]], shape=(2, 3), dtype=int32)
```

創建一個所有元素都設為1的張量。下面的語句即創建一個形如[M，N]、元素均為1的矩陣：

tf.ones([M,N],tf,dtype) 例如：

In [11]:

```
ones_t = tf.ones([2,3],tf.int32)
# Results in an 2x3 array of ones:[[1 1 1],[1 1 1]]
print("ones_t=" , ones_t)
```

```
ones_t= tf.Tensor(
[[1 1 1]
 [1 1 1]], shape=(2, 3), dtype=int32)
```

更進一步，還有以下語句： 在一定範圍內生成一個從初值到終值等差排布的序列： tf.linspace(start,stop,num)

相應的值為(stop-start)/(num-1)。例如：

In [13]:

```
range_t = tf.linspace(2.0,5.0,5)
#We get:[2. 2.75 3.5 4.25 5.]
print("range_t=" , range_t)
```

```
range_t= tf.Tensor([2.   2.75 3.5  4.25 5.  ], shape=(5,), dtype=float32)
```

從開始（默認值=0）生成一個數字序列，增量為delta（默認值=1），直到終值（但不包括終值）：

tf.range(start,limit,delta) 實例：

In [14]:

```
range_t = tf.range(10)
#Result:[0 1 2 3 4 5 6 7 8 9]
print("range_t=" , range_t)
```

```
range_t= tf.Tensor([0 1 2 3 4 5 6 7 8 9], shape=(10,), dtype=int32)
```

In [10]:

```
#Create tf.Variable
a = tf.ones(6)
b = tf.Variable(a,name='Data_point')
b
```

Out[10]:

```
<tf.Variable 'Data_point:0' shape=(6,) dtype=float32, numpy=array([1., 1.,
1., 1., 1., 1.], dtype=float32)>
```

In [11]:

```
b.trainable #check weather trainable
```

Out[11]:

```
True
```

placeholder

先給定記憶體長度或資料格式來預先佔有空間

TF2.X中，預設式轉變成eager execution，已經不使用變數預留空間

建立數據

三種方法建立初值

In [12]:

```
#Create 0 matrix
a = tf.zeros([6,6])
#or
b = tf.zeros_like(a)
#or fill -> shape and fill nums
c = tf.fill([6,6],0)
```

In [13]:

```
c
```

Out[13]:

```
<tf.Tensor: shape=(6, 6), dtype=int32, numpy=
array([[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]], dtype=int32)>
```

TensorFlow 允許創建具有不同分佈的隨機張量：

1. 使用以下語句創建一個具有一定均值（默認值=0.0）和標準差（默認值=1.0）、形狀為[M，N] 的正態分佈隨機數組：

In [18]:

```
t_random = tf.random.normal([2,3],mean=2,stddev=4,seed=12)
print("t_random=" , t_random)
```

```
t_random= tf.Tensor(
[[-4.349812   -3.0933433   8.275564  ]
 [ 7.594398    0.80044365  4.6968856 ]], shape=(2, 3), dtype=float32)
```

2. 創建一個具有一定均值（默認值=0.0）和標準差（默認值=1.0）、形狀為[M，N] 的截尾正態分佈隨機數組：

In [20]:

```
t_random = tf.random.truncated_normal([1,5],stddev=2, seed=12)
print("t_random=" , t_random)
```

```
t_random= tf.Tensor([[-0.87326276  1.689955   -0.02361972 -1.7688016  -3.877
49   ]], shape=(1, 5), dtype=float32)
```

3. 要在種子的[minval（default=0），maxval] 範圍內創建形狀為[M，N] 的給定伽馬分佈隨機數組，請執行如下語句：

In [21]:

```
t_random = tf.random.uniform([2,3],maxval=4, seed=12)
print("t_random=" , t_random)
```

```
t_random= tf.Tensor(
[[2.54461   3.6963658 2.7051091]
 [2.0085006 3.8445983 3.5426888]], shape=(2, 3), dtype=float32)
```

建立隨機矩陣

1. normal distribution:常態分配
2. truncated normal distribution:可設上下限

3. random by uniform: 隨機均勻分配

In [14]:

```
#Example random by normal distribution
a = tf.random.normal([6,6],mean=0,stddev=1)
#random by truncated normal distribution
b = tf.random.truncated_normal([6,6],mean=0,stddev=1)
#random by uniform
c = tf.random.uniform([6,6],minval=0,maxval=1)
```

In [15]:

```
a
```

Out[15]:

```
<tf.Tensor: shape=(6, 6), dtype=float32, numpy=
array([[ 0.24097888,  0.20553762,  0.8995366 ,  1.1600448 ,  0.25291708,
         0.37648833],
       [-0.42466396,  0.6204753 ,  0.18946598,  0.73095596, -0.52937424,
        -0.57800055],
       [-0.3604185 , -0.35277134,  1.3113467 , -0.88858694, -1.2658571 ,
        -2.048099  ],
       [-1.5397677 ,  2.6110299 , -0.78358126, -0.8021525 , -0.80980825,
         0.27005717],
       [ 0.6558137 ,  0.19863158,  0.19837783, -0.14081572, -0.04454594,
         1.2814765 ],
       [-0.8373519 , -0.5468744 ,  0.77884823, -0.16533092, -0.23612665,
        -0.9739968 ]], dtype=float32)>
```

TensorFlow 變量

它們通過使用變量類來創建。變量的定義還包括應該初始化的常量/隨機值。下面的代碼中創建了兩個不同的張量變量t_a 和t_b。兩者將被初始化為形狀為[50，50] 的隨機均勻分佈，最小值=0，最大值=10：

注意：變量通常在神經網絡中表示權重和偏置。

In [28]:

```
rand_t = tf.random.uniform([50,50],0,10,seed=0)
t_a= tf.Variable(rand_t)
t_b=tf.Variable(rand_t)
```

下面的代碼中定義了兩個變量的權重和偏置。權重變量使用正態分佈隨機初始化，均值為0，標準差為2，權重大小為100×100。偏置由100個元素組成，每個元素初始化為0。在這裡也使用了可選參數名以給計算圖中定義的變量命名：

In [41]:

```
weights= tf.Variable(tf.random.normal([100,100],stddev=12))
bias = tf.Variable(tf.zeros([100]), name = 'biases')
print("bias =", bias)
```

```
bias = <tf.Variable 'biases:0' shape=(100,) dtype=float32, numpy=
array([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., 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.],
      dtype=float32)>
```