Deep Neural Networks

了解如何透過在序列模型中堆疊層來建立深度神經網路。透過在隱藏層後面添加激活函數，我們賦予了網路學習數據中更複雜（非線性）關係的能力。

建立一個包含多個隱藏層的神經網絡，然後探索一些 ReLU 以外的活化函數。運行下一個單元來設定所有內容！

```
import tensorflow as tf

# Setup plotting
import matplotlib.pyplot as plt

plt.style.use('seaborn-whitegrid')
# Set Matplotlib defaults
plt.rc('figure', autolayout=True)
plt.rc('axes', labelweight='bold', labelsize='large',
       titleweight='bold', titlesize=18, titlepad=10)

# Setup feedback system
from learntools.core import binder
binder.bind(globals())
from learntools.deep_learning_intro.ex2 import *
```

在混凝土(*Concrete*)資料集中，任務是預測根據不同配方生產的混凝土的抗壓強度。

```
import pandas as pd

concrete = pd.read_csv('../input/dl-course-data/concrete.csv')
concrete.head()
```

[2]:

	Cement	BlastFurnaceSlag	FlyAsh	Water	Superplasticizer	CoarseAggregate	FineAggregate	Age	CompressiveStrength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30

1) Input Shape

此任務的目標是「抗壓強度」列('CompressiveStrength')。其餘列是我們將用作輸入的特徵。

此資料集的輸入 shape 是什麼？(如果你的每個樣本有 10 個特徵，input_shape 設定成 8)

```
input_shape = [8]

# Check your answer
q_1.check()
```

Correct

2）Define a Model with Hidden Layers(定義具有隱藏層的模型)

現在建立一個包含三個隱藏層的模型，每個隱藏層包含 512 個單元，並使用 ReLU 啟動函數。務必包含一個包含一個單元且無激活函數的輸出層，並將 input_shape 作為第一層的參數。

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(512, activation='relu', input_shape=input_shape),
    layers.Dense(512, activation='relu'),
    layers.Dense(512, activation='relu'),
    layers.Dense(1),
])
```

3）激活層

讓我們來探索一下激活函數。

將激活函數(Activation function)附加到 Dense 層通常的方法是將其作為激活參數定義的一部分。但有時，您可能會想要在 Dense 層和其激活函數之間添加其他層。（我們將在第 5 課中看到一個使用批量歸一化(*batch normalization*)的範例。）在這種情況下，我們可以在其自己的 Activation 層中定義激活函數，如下所示：

layers.Dense(units=8),

layers.Activation('relu')

這完全等同於普通的方式：

layers.Dense(units=8, activation='relu')

每個 Activation 都在其自己的 Activation layer 中。

```
# rewrite this to use activation layers
model = keras.Sequential([
    layers.Dense(32, input_shape=[8]),
    layers.Activation('relu'),
    layers.Dense(32),
    layers.Activation('relu'),
    layers.Dense(1),
])
```

ReLU 的替代方案

「ReLU」Activation 有很多變體，例如「elu」、「selu」和「swish」等等，所有這些激活函數都可以在 Keras 中使用。有時，在特定任務上，某種激活函數的效果會優於另一種，因此在開發模型時，您可以考慮嘗試不同的激活函數。ReLU 激活函數在大多數問題上都表現良好，因此是一個不錯的入門選擇。

讓我們來看看其中一些激活函數的圖表。將激活函數從 “ReLU” 更改為上面提

到的其他激活函數之一。Run cell 即可以看圖表

```
#Change 'relu' to 'elu', 'selu', 'swish'... or something else
activation_layer = layers.Activation('relu')

x = tf.linspace(-3.0, 3.0, 100)
y = activation_layer(x) # once created, a layer is callable just like a function

plt.figure(dpi=100)
plt.plot(x, y)
plt.xlim(-3, 3)
plt.xlabel("Input")
plt.ylabel("Output")
plt.show()
```