

# Example & 實作

Regression

# Regression 舉例

# Import lib

- Import
  - Numpy
  - Keras
  - matplotlib

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
import matplotlib.pyplot as plt #畫圖
import time
```

# Create Dataset

- Create X\_data and Y\_data
  - $Y = \sin(X) + \text{noise}$
  - 我們的目標是去模擬sin

```
data_set = 360 # 資料量大小
train_data_volume = 180 # 用來training的資料數量

data = (np.random.random((data_set))-0.5) * 2 * np.pi # random 0~1 取data_set數量的data 再乘上PI
output = np.sin(data )+np.random.normal(0, 0.1, (data_set, )) # 對data取sin值 並加上一個noise

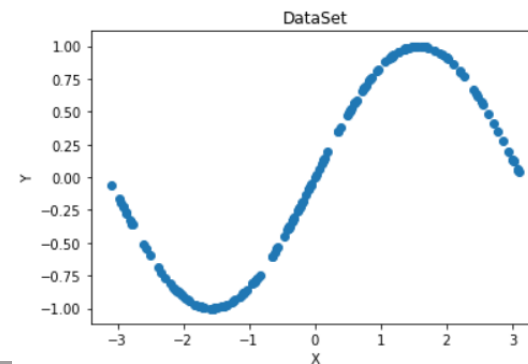
X_train, Y_train = data[:train_data_volume], output[:train_data_volume] #部分分成training
X_test, Y_test = data[train_data_volume:], output[train_data_volume:] #部分分成testing
```

# Draw Dataset

- `def draw_dataset(x,y):`
  - Function ,can return value
- `draw_dataset(x,y)`
  - Call Function
- `plt.show(block = False)`
  - If `block = True`, the program will stop until close the figure

```
def draw_dataset(x,y):  
    plt.scatter(x,y,c = "b",s=4) #畫x,y圖，設定顏色為BLUE，SIZE 為4  
    plt.title("DataSet") # 設定圖案的標題  
    plt.xlabel("X") #設定X軸的座標軸名稱  
    plt.ylabel("Y") #設定Y軸的座標軸名稱  
    plt.show(block=False)  
    plt.pause(0.3) #暫停0.3秒  
    plt.cla() #清空圖案
```

```
draw_dataset(X_train,Y_train) #畫出training資料的分布圖  
draw_dataset(X_test,Y_test) #畫出testing資料的分布圖
```



# Create Model

```
model = Sequential()  
model.add(Dense(50, input_shape=(1,)))  
model.add(Activation('relu'))  
model.add(Dense(50))  
model.add(Activation('relu'))  
model.add(Dense(1))  
  
model.compile(loss='mse', optimizer='Adam')  
model.summary()
```

# Training

- Training the model
  - Set your epoch

```
epoch = 501 #訓練次數
print('Training -----')
for step in range(epoch):
    cost = model.train_on_batch(X_train, Y_train)
    if step % 50 == 0: #每50次訓練 測試一次結果
        print('train cost: ', cost)
        test = model.predict(X_test)
```

- Result:

```
train cost: 0.6363339
train cost: 0.1652615
train cost: 0.12487392
train cost: 0.071828716
train cost: 0.029051663
train cost: 0.01684584
train cost: 0.0136627965
train cost: 0.012884193
train cost: 0.012624998
train cost: 0.012508283
train cost: 0.012435598
```

# Visualize

- Drawing the process of training

```
def draw_training(data,predict,ground,step):  
    global epoch  
    plt.scatter(data,predict,c = "r",marker = "*",label = "Predict") #畫x,y圖，設定顏色為RED，marker為*  
    plt.scatter(data,ground,c = "b",s = 4, label = "Ground Truth") #畫x,y圖，設定顏色為Blue，size為4  
    plt.xlabel("X") #設定x軸的座標軸名稱  
    plt.ylabel("Y") #設定y軸的座標軸名稱  
    plt.ion() # 不關掉圖片  
    if step != epoch-1:  
        plt.title("Training...Epoch:"+str(step)) # 設定圖案的標題  
        plt.show(block=False)  
        plt.pause(0.3) #暫停0.3秒  
        plt.cla() #清空圖案  
    else:  
        plt.title("Training Done!!!") # 設定圖案的標題  
        plt.show(block=True)
```

```
epoch = 501 #訓練次數  
print('Training -----')  
for step in range(epoch):  
    cost = model.train_on_batch(X_train, Y_train)  
    if step % 50 == 0: #每50次訓練 測試一次結果  
        print('train cost: ', cost)  
        test = model.predict(X_test)  
        draw_training(X_test,test,Y_test ,step)
```



