



國立雲林科技大學
電子工程系
Department of Electronic Engineering

教育部補助AI應用領域系列課程-
人工智慧計算晶片設計和應用人才培育

LAB2-Linear regression

國立雲林科技大學

夏世昌 特聘教授 / 電子工程系

王斯弘 助理教授 / 前瞻學位學士學程

2020, Fall Semester



安裝TensorFlow

GPU版本

需在tf2環境下安裝 **codatoolkit** 和 **cudnn**



安裝TensorFlow

GPU版本codatoolkit

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Learning

Community

Documentation

Developer Blog

Create Clone Import Remove

Search Environments

base (root)

tf

tf2

tf23

All Channels Update index... cud

Name	Description	Version
base (root)		9.2
tf	deep neural network acceleration library	7.6.5
tf2	representation of a numpy-compatible multi-dimensional array on cuda.	6.0.0

Unmark

Mark for installation

Mark for update

Mark for removal

Mark for specific version installation

11.0.221

10.2.89

10.1.243

10.1.168

10.0.130

9.2

9.0

8.0

3 packages available matching "cud"



安裝TensorFlow

GPU版本cudnn

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Learning

Community

Documentation

Developer Blog

Create Clone Import Remove

Search Environments

base (root)

tf

tf2

tf23

All Channels Update index... cud

Name	Description	Version
<input type="checkbox"/> cudatoolkit		9.2
<input checked="" type="checkbox"/> cudnn	Nvidia's cudnn deep neural network acceleration library	7.6.5
<input type="checkbox"/> Unmark	representation of a numpy-compatible multi-dimensional array on cuda.	6.0.0

Mark for installation

Mark for update

Mark for removal

Mark for specific version installation

7.6.5

7.6.4

7.6.0

7.3.1

7.1.4

3 packages available matching "cud" 1 package selected

Apply Clear



執行環境

Anaconda Navigator

File Help



Home

Environments

Learning

Community

Search Environments

base (root)

tf

tf2

tf23

All

Name

Description

Open Terminal

Open with Python

Open with IPython

Open with Jupyter Notebook

base (root)		
tf	<input type="checkbox"/>	cuda toolkit
tf2	<input type="checkbox"/>	Nvidia's cuda toolkit
tf23	<input type="checkbox"/>	Cupy is an open source NumPy-compatible array library for GPU computing



執行環境

```
C:\WINDOWS\system32\cmd.exe - python

(tf2) C:\Users\tony3>python
Python 3.7.9 (default, Aug 31 2020, 17:10:11) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



TensorFlow

<https://www.tensorflow.org/>

The screenshot shows the TensorFlow website homepage. At the top, there is a navigation bar with the TensorFlow logo, links for 'Install', 'Learn', 'API', 'Resources', 'Community', and 'Why TensorFlow', a search bar, and language selection ('English'). Below the navigation bar is a black banner with the text: 'Google is committed to advancing racial equity for Black communities. [See how.](#)'. The main content area features a large orange arrow pointing right with the text 'An end-to-end open source machine learning platform'. Below this, there are tabs for 'TensorFlow', 'For JavaScript', 'For Mobile & IoT', and 'For Production'. The 'TensorFlow' tab is selected, showing the text: 'The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.' and a button labeled 'Get started with TensorFlow'. The background of the main content area is a light gray with a stylized illustration of a computer monitor, a server rack, a car, an airplane, and various icons connected by lines, representing machine learning applications.



TensorFlow

CPU版本

在 ES302 有些電腦請 tensorflow安裝2.5.0
pip install tensorflow==2.5.0

```
C:\WINDOWS\system32\cmd.exe
```

```
(tf2) C:\Users\tony3>pip install tensorflow2.0.0
```

GPU版本

```
C:\WINDOWS\system32\cmd.exe
```

```
(tf2) C:\Users\tony3>pip install tensorflow-gpu==2.0.0
```




TensorFlow

安裝後可以在python中import tensorflow。
若沒有出現Error代表安裝成功。

```
(tf2) C:\Users\tony3>python
Python 3.7.9 (default, Aug 31 2020, 17:10:11) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.__version__
'2.0.0'
>>>
```



JUPYTER LAB

建議可安裝jupyter lab
& `pip install jupyterlab`

A screenshot of a Windows command prompt window. The title bar reads "C:\WINDOWS\system32\cmd.exe - activate". The command prompt shows the user's location as "(tf2) C:\Users\tony3" and the command being entered is "pip install jupyterlab". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\WINDOWS\system32\cmd.exe - activate
(tf2) C:\Users\tony3>pip install jupyterlab
```



執行JUPYTER LAB

& jupyter lab

A screenshot of a Windows command prompt window. The title bar reads "C:\WINDOWS\system32\cmd.exe - activate". The command prompt shows the user is in the directory "C:\Users\tony3" and has entered the command "jupyter lab". The command is highlighted in blue. The prompt is "(tf2) C:\Users\tony3>".

```
C:\WINDOWS\system32\cmd.exe - activate
(tf2) C:\Users\tony3>jupyter lab
```



執行JUPYTER LAB

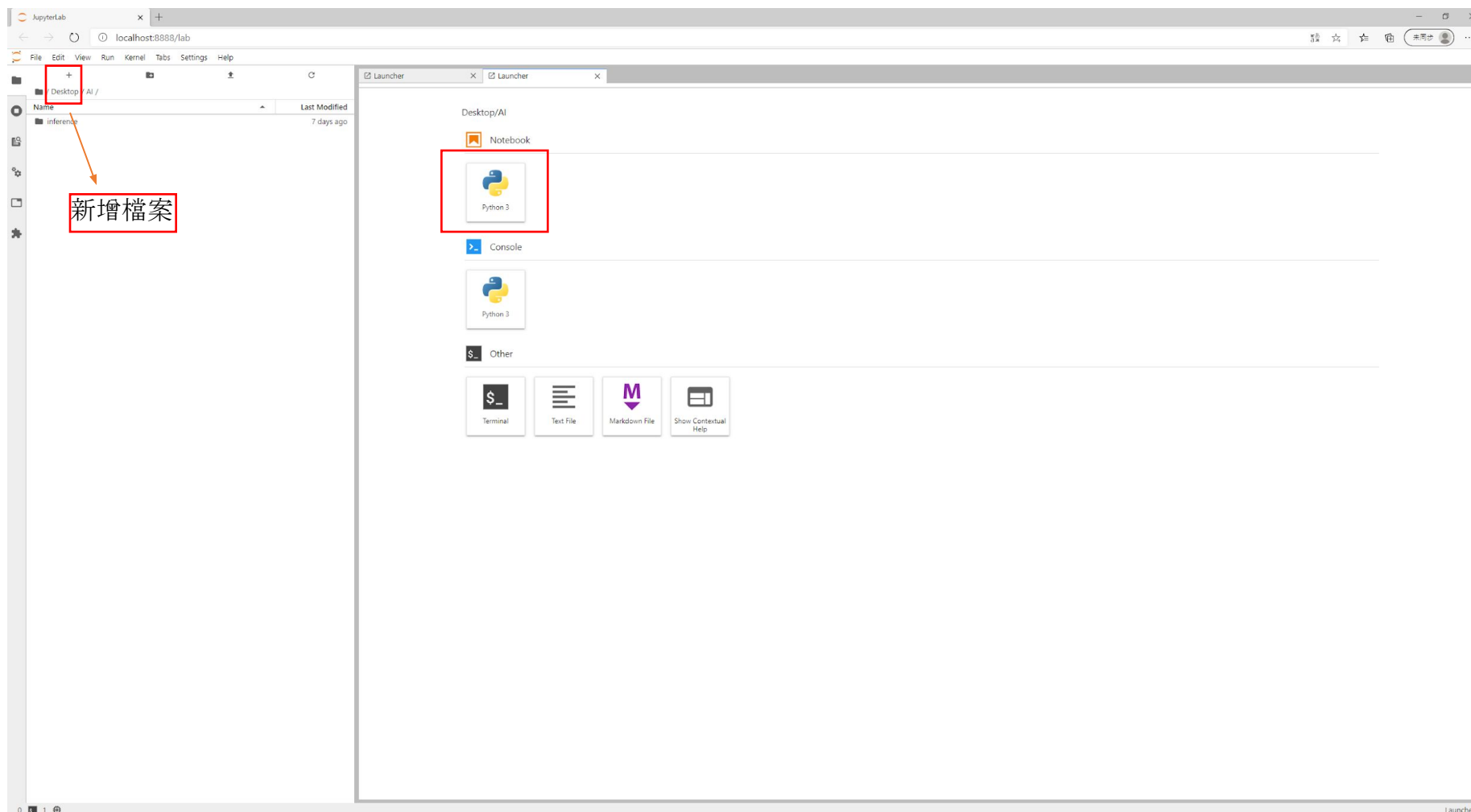
```
C:\WINDOWS\system32\cmd.exe - activate - jupyter lab

(tf2) C:\Users\tony3>jupyter lab
[I 21:39:09.273 LabApp] JupyterLab extension loaded from c:\users\tony3\anaconda3\envs\tf2\lib\site-packages\jupyterlab
[I 21:39:09.273 LabApp] JupyterLab application directory is c:\users\tony3\anaconda3\envs\tf2\share\jupyter\lab
[I 21:39:09.277 LabApp] Serving notebooks from local directory: C:\Users\tony3
[I 21:39:09.277 LabApp] Jupyter Notebook 6.1.3 is running at:
[I 21:39:09.277 LabApp] http://localhost:8888/?token=6372eb3fdb104bda2ee7dbfa58e12046ed398c026ce0b3e2
[I 21:39:09.277 LabApp] or http://127.0.0.1:8888/?token=6372eb3fdb104bda2ee7dbfa58e12046ed398c026ce0b3e2
[I 21:39:09.277 LabApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 21:39:09.333 LabApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/tony3/AppData/Roaming/jupyter/runtime/nbserver-4496-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=6372eb3fdb104bda2ee7dbfa58e12046ed398c026ce0b3e2
    or http://127.0.0.1:8888/?token=6372eb3fdb104bda2ee7dbfa58e12046ed398c026ce0b3e2
[W 21:39:12.945 LabApp] Could not determine jupyterlab build status without nodejs
[I 21:39:14.385 LabApp] Kernel started: e9f4905a-b677-4c0d-bfe3-f8fe93a6795a, name: python3
```

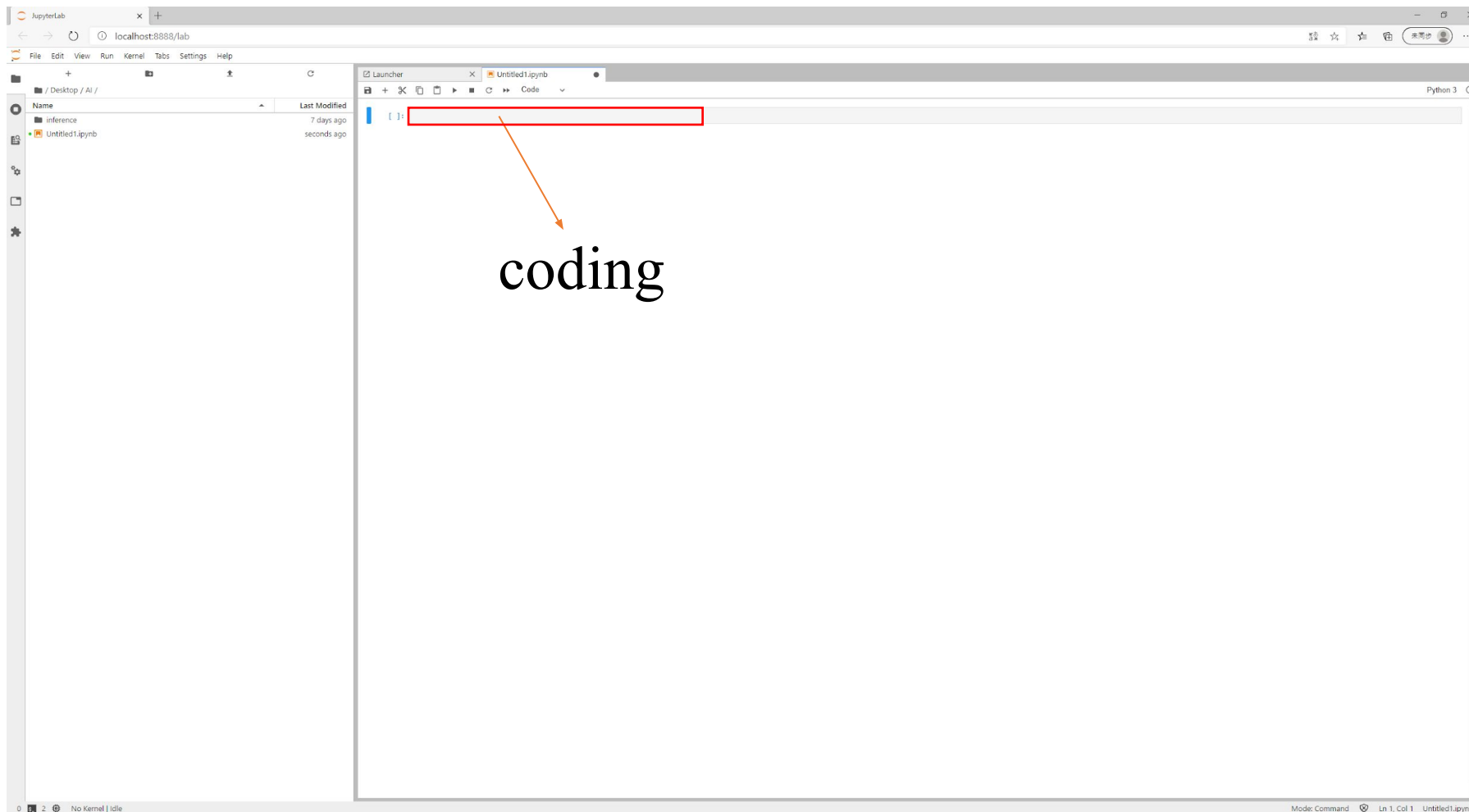


執行JUPYTER LAB





執行JUPYTER LAB





執行JUPYTER LAB

The screenshot displays the JupyterLab web interface. On the left, the file browser shows a directory structure with 'inference' and 'Untitled1.ipynb'. The main editor area shows a code cell with the following code:

```
[1]: import tensorflow as tf  
tf.__version__
```

The output of the code cell is displayed below the code:

```
[1]: '2.0.0'
```

Annotations on the image provide instructions:

- An arrow points to the run button (a triangle icon) in the code cell toolbar, with the text "Coding後 按此 或 shift + enter" (After coding, press this or shift + enter).
- Another arrow points to the output '2.0.0', with the text "顯示輸出結果" (Display output result).

The status bar at the bottom indicates "No Kernel | Idle" and "Mode: Edit | Ln 1, Col 1 | Untitled1.ipynb".



START!!!



Import library

```
import tensorflow as tf  
import numpy as np|
```

Import library

```
import tensorflow as tf
```

TensorFlow是一個開源軟體庫，用於各種感知和語言理解任務的機器學習。

https://www.tensorflow.org/api_docs/python/tf



Import library

```
import numpy as np
```

Numpy 是 Python 的一個重要模組，主要用於
資料處理上。





Import library

安裝numpy

& pip install numpy

```
(tf2.3) C:\Users\user>pip install numpy
```





dataset

```
X = np.random.rand(1000).astype(np.float32)
print('X=',X)
n = X.shape[0]
Y = X * 9 + 5
print('Y=',Y)
```

X為訓練資料，亂數產生1000筆

Y為對應X的答案



dataset

numpy.random.rand

numpy.random.rand

`numpy.random.rand` (d_0, d_1, \dots, d_n)

Random values in a given shape.

Create an array of the given shape and populate it with random samples from a uniform distribution over `[0, 1)`.

Parameters: d_0, d_1, \dots, d_n : *int, optional*

The dimensions of the returned array, should all be positive. If no argument is given a single Python float is returned.

Returns: `out` : *ndarray, shape* (d_0, d_1, \dots, d_n)

Random values.



權重初始化

```
W = tf.Variable(tf.random.normal([1]))  
print(W)  
b = tf.Variable(tf.random.normal([1]))  
print(b)
```

```
<tf.Variable 'Variable:0' shape=(1,) dtype=float32, numpy=array([0.17674959], dtype=float32)>  
<tf.Variable 'Variable:0' shape=(1,) dtype=float32, numpy=array([0.16365877], dtype=float32)>
```



權重初始化

tf.Variable()

The `Variable()` constructor requires an initial value for the variable, which can be a `Tensor` of any type and shape. This initial value defines the type and shape of the variable. After construction, the type and shape of the variable are fixed. The value can be changed using one of the assign methods.




訓練

```
for step in range(0,1001):  
    with tf.GradientTape() as g:  
        pred = W*X+b  
        loss = tf.reduce_sum(tf.pow(pred-Y,2))/n  
  
    gradient = g.gradient(loss,[W,b])  
  
    tf.optimizers.Adam(.1).apply_gradients(zip(gradient, [W, b]))  
  
    if step % 100 == 0:  
        pred = W*X+b  
        loss = tf.reduce_sum(tf.pow(pred-Y,2))/n  
        print("step: %i, loss: %f, W: %f, b: %f" % (step, loss, W.numpy(), b.numpy()))
```

預測為 $W*X+b$

Loss採用mean square

Optimizer採用Adam


$$\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$



訓練

```
loss = tf.reduce_sum(tf.pow(pred-Y,2))/n
```

$$\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$



訓練

```
step: 0, loss: 87.877869, W: 0.276753, b: 0.263662
step: 100, loss: 0.020041, W: 8.676969, b: 5.263853
step: 200, loss: 0.020030, W: 8.677159, b: 5.263757
step: 300, loss: 0.020019, W: 8.677350, b: 5.263662
step: 400, loss: 0.020008, W: 8.677541, b: 5.263566
step: 500, loss: 0.019997, W: 8.677732, b: 5.263471
step: 600, loss: 0.019986, W: 8.677922, b: 5.263376
step: 700, loss: 0.019976, W: 8.678113, b: 5.263280
step: 800, loss: 0.019965, W: 8.678304, b: 5.263185
step: 900, loss: 0.019954, W: 8.678494, b: 5.263090
step: 1000, loss: 0.019943, W: 8.678685, b: 5.262994
```



結果

```
import matplotlib.pyplot as plt
plt.plot(X, Y, 'ro', label='Original data')
plt.plot(X, np.array( W * X + b), label='pred line')
plt.legend()
plt.show()
```



結果

```
import matplotlib.pyplot as plt
```

matplotlib是Python程式語言及其數值數學擴展包 NumPy的可視化操作界面。



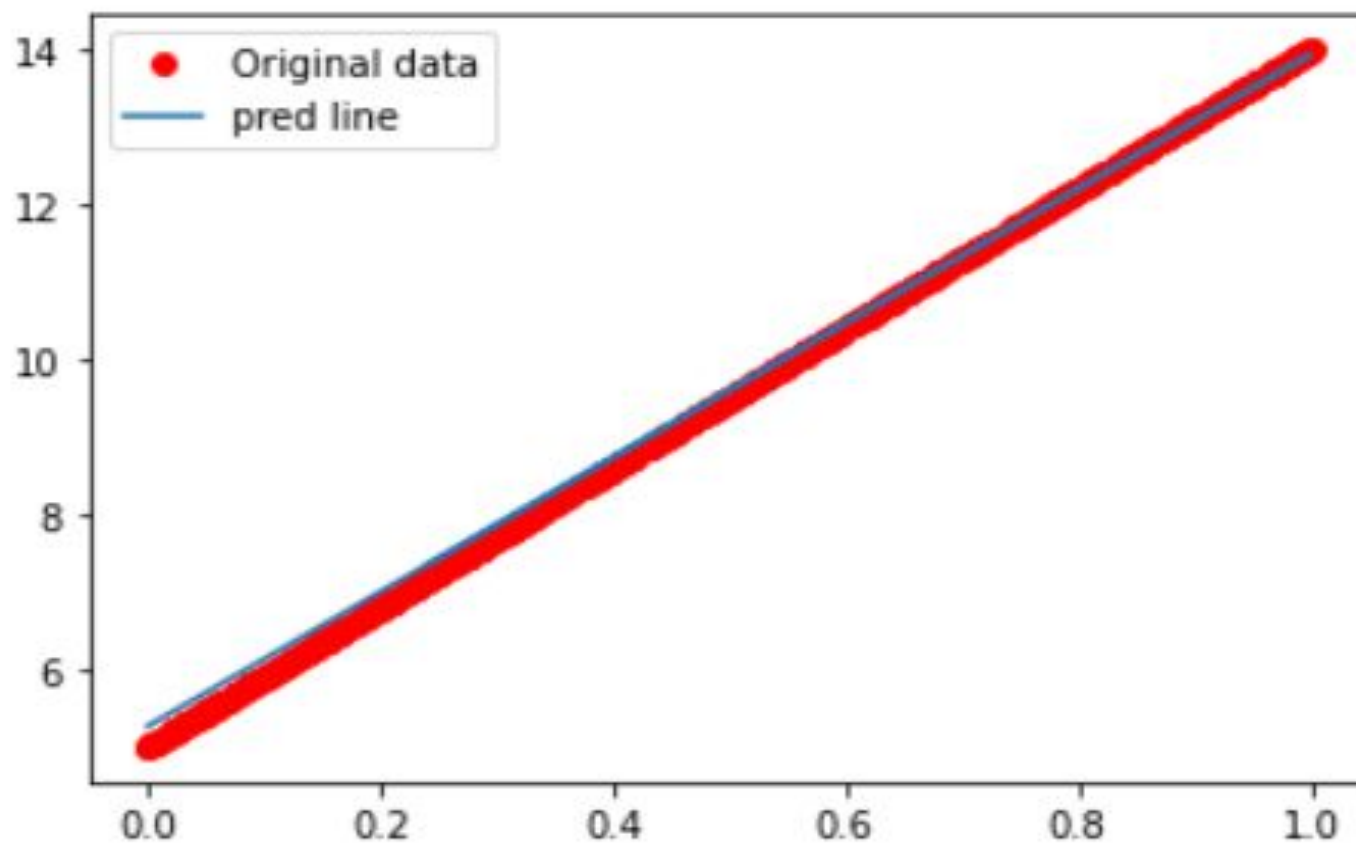
結果

安裝matplotlib

& pip install matplotlib

```
(tf2.3) C:\Users\user>pip install matplotlib
```

結果





訓練房價與面積關係

LotArea	SalePrice
8450	208500
9600	181500
11250	223500
9550	140000
14260	250000
14115	143000
10084	307000
10382	200000
6120	129900
7420	118000
11200	129500
11924	345000
12968	144000
10652	279500
10920	157000
6120	132000
11241	149000
10791	90000



Import library

```
import tensorflow as tf  
import numpy as np  
import pandas as pd
```



Import library

```
import pandas as pd
```

pandas是Python程式語言的用於數據操縱和分析的軟體庫。



dataset

```
train = pd.read_csv("train.csv")  
train = train[train['LotArea'] < 5000]  
train_X = train['LotArea'].values.reshape(-1,1)  
train_Y = train['SalePrice'].values.reshape(-1,1)  
n_sample = train_X.shape[0]
```



dataset

```
pd.read_csv("train.csv")
```

->讀取csv資料

```
train = train[train['LotArea'] < 5000]
```

->只讀取‘LotArea’中小於5000的資料



dataset

train_X = train['LotArea'].values.reshape(-1,1)

-> LotArea資料擺入train_X

train_Y = train['SalePrice'].values.reshape(-1,1)

-> SalePrice資料擺入train_Y

n_sample = train_X.shape[0]

-> n_sample 為train_X有幾筆資料



權重初始化

```
W = tf.Variable(tf.random.normal([1]))  
print(W)  
b = tf.Variable(tf.random.normal([1]))  
print(b)
```

```
<tf.Variable 'Variable:0' shape=(1,) dtype=float32, numpy=array([1.2125201], dtype=float32)>
```

```
<tf.Variable 'Variable:0' shape=(1,) dtype=float32, numpy=array([-0.7893576], dtype=float32)>
```

訓練

```
for step in range(0,10001):  
    with tf.GradientTape() as g:  
        pred = W*train_X+b  
        loss = tf.reduce_sum(tf.pow(pred-train_Y,2))/(n_sample)  
  
    gradient = g.gradient(loss,[W,b])  
  
    tf.optimizers.Adam(2).apply_gradients(zip(gradient, [W, b]))  
  
    if step % 100 == 0:  
        pred = W*train_X+b  
        loss = tf.reduce_sum(tf.pow(pred-train_Y,2))/(n_sample)  
        print("step: %i, loss: %f, W: %f, b: %f" % (step, loss, W.numpy(), b.numpy()))
```

預測為 $W \cdot \text{train_X} + b$

Loss採用mean square

Optimizer採用Adam

$$\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$



訓練

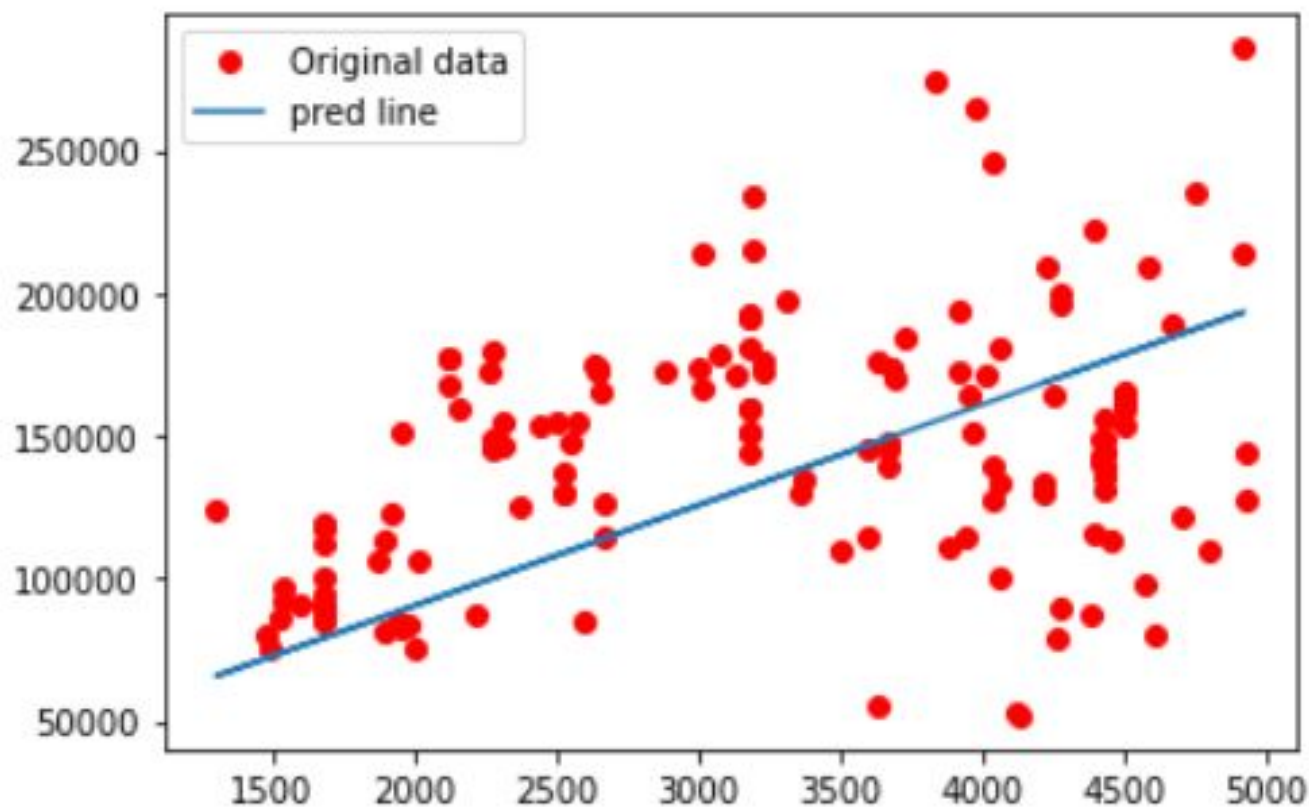
```
step: 8000, loss: 2411719936.000000, W: 39.213654, b: 16001.242188
step: 8100, loss: 2411141888.000000, W: 39.213654, b: 16201.242188
step: 8200, loss: 2409086720.000000, W: 35.213531, b: 16401.242188
step: 8300, loss: 2403479040.000000, W: 35.213531, b: 16601.242188
step: 8400, loss: 2397951488.000000, W: 35.213531, b: 16801.242188
step: 8500, loss: 2392503296.000000, W: 35.213531, b: 17001.242188
step: 8600, loss: 2387135744.000000, W: 35.213531, b: 17201.242188
step: 8700, loss: 2381848064.000000, W: 35.213531, b: 17401.242188
step: 8800, loss: 2376640256.000000, W: 35.213531, b: 17601.242188
step: 8900, loss: 2371512320.000000, W: 35.213531, b: 17801.242188
step: 9000, loss: 2366464512.000000, W: 35.213531, b: 18001.242188
step: 9100, loss: 2361496832.000000, W: 35.213531, b: 18201.242188
step: 9200, loss: 2356609024.000000, W: 35.213531, b: 18401.242188
step: 9300, loss: 2351801344.000000, W: 35.213531, b: 18601.242188
step: 9400, loss: 2347073536.000000, W: 35.213531, b: 18801.242188
step: 9500, loss: 2342425600.000000, W: 35.213531, b: 19001.242188
step: 9600, loss: 2337857792.000000, W: 35.213531, b: 19201.242188
step: 9700, loss: 2333370368.000000, W: 35.213531, b: 19401.242188
step: 9800, loss: 2328962560.000000, W: 35.213531, b: 19601.242188
step: 9900, loss: 2324634880.000000, W: 35.213531, b: 19801.242188
step: 10000, loss: 2320387072.000000, W: 35.213531, b: 20001.242188
```




結果

```
import matplotlib.pyplot as plt
plt.plot(train_X, train_Y, 'ro', label='Original data')
plt.plot(train_X, np.array( W * train_X + b), label='pred line')
plt.legend()
plt.show()
```

結果





END!!!