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

7 Topics to Mastering Basic Machine Learning with Python

1. Understanding the Python Scientific Computing Environment
2. Python Basics
3. Regression (supervised learning)
4. Classification (supervised learning)
5. Clustering (unsupervised learning)
6. More Classification (support vector machines)
7. Ensemble Methods (CatBoost, Light GBM, XGBoost)

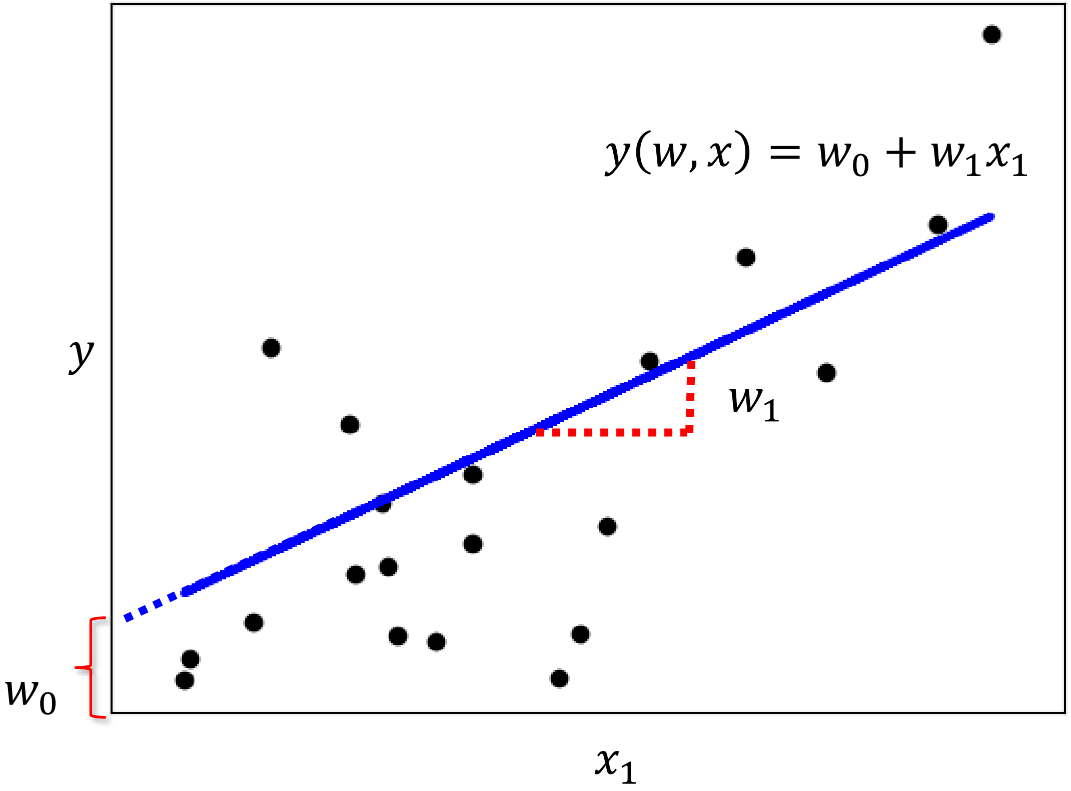
Regression algorithm(Manually)

- Regression的過程是找出輸入(independent variable, feature)和輸出(dependent variable, target)之間的關係。
- 使用線性關係(模型)描述feature與target就稱為Linear regression。
- Simple linear regression

$$y(w,x) = w_0 + w_1x_1$$

- multiple regression

$$y(w,x) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_px_p$$



w_0是截距(Intercept)
$w = (w_1, w_2,, w_p)$是斜率(Slope)

Regression algorithm(Manually)

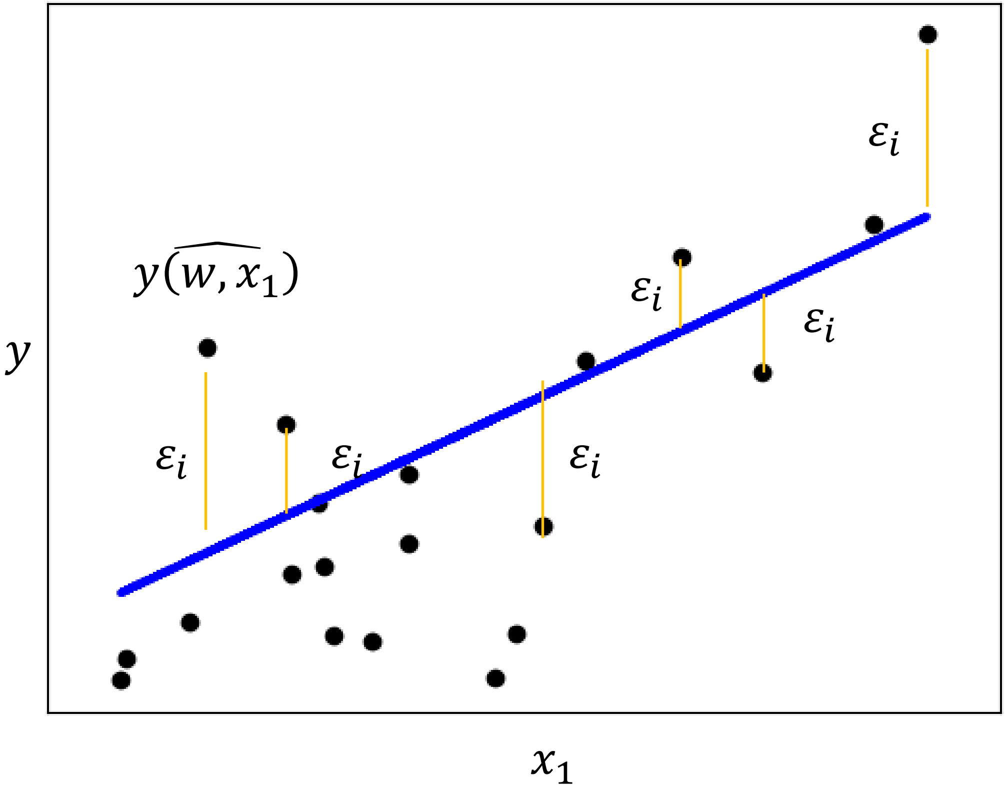
- Regression的過程是找w_0和$w = (w_1, w_2, , w_p)$。
- 收集一組資料$(x_i, y_i), i = 1,2, \ldots, n$，將每個點都帶到模型內可以得到模型的預估值

$$y\widehat{(w, x_i)} = w_0 + w_1 x_i \ \ , i = 1,2, \ldots, n$$

- 預估值和實際值的差異稱為誤差(error)或稱為殘差(Residual)

$$\varepsilon_i = y(w, x_i) - y\widehat{(w, x_i)}$$

- Regression的目標是希望找到一組參數$(\widehat{w_0}, \widehat{w_1})$使得模型的殘差越小越好，數值上有許多種方法可以找出這組參數，最小平方法是一種常用的方法。
- 找出參數的過程，本質上就是最佳化的問題。



Regression algorithm(Manually)

- 因為誤差值有正有負，取平方後皆為正值，所以我們會很希望所有訓練樣本的誤差平方和(Sum Square error, SSE)接近0。
- $\text{Loss}(\widehat{w_0}, \widehat{w_1}) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}\left(y_i - (\widehat{w_0} + \widehat{w_1}x_i)\right)^2$
- 極值會出現在微分為0的地方。對殘差分別做$\widehat{w_0}, \widehat{w_1}$的偏微分。
- $\frac{\partial \text{Loss}(\widehat{w_0},\widehat{w_1})}{\partial \widehat{w_0}} = \frac{\partial \sum_{i=1}^{n}\left(y_i-(\widehat{w_0}+\widehat{w_1}x_i)\right)^2}{\partial \widehat{w_0}} = 0$

$$\rightarrow -2\sum_{i=1}^{n}(y_i - \widehat{w_0} - \widehat{w_1}x_i) = 0 \rightarrow \widehat{w_0} = \bar{y} - \widehat{w_1}\bar{x}$$

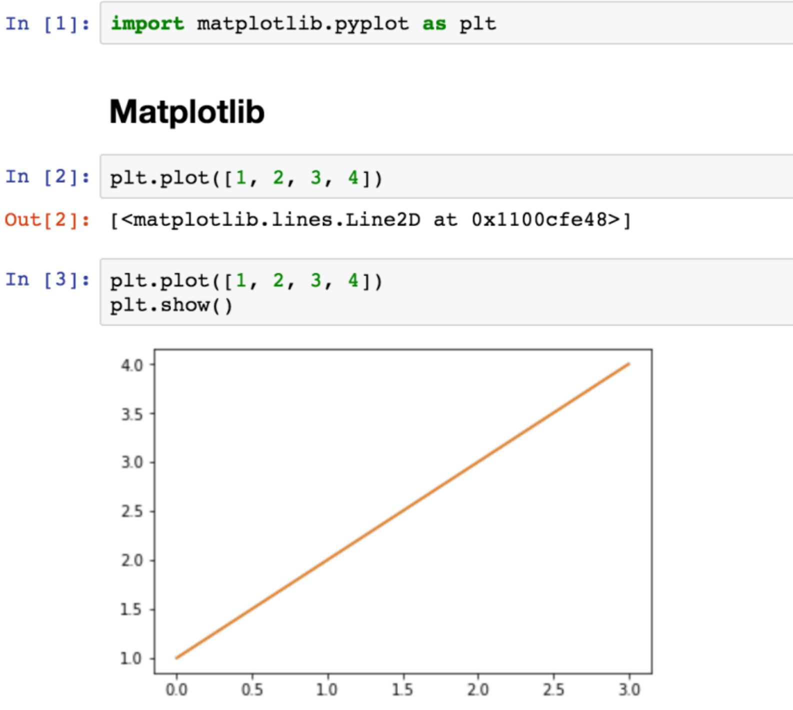
- $\frac{\partial \text{Loss}(\widehat{w_0},\widehat{w_1})}{\partial \widehat{w_1}} = \frac{\partial \sum_{i=1}^{n}\left(y_i-(\widehat{w_0}+\widehat{w_1}x_i)\right)^2}{\partial \widehat{w_1}} = 0$

$$\rightarrow -2\sum_{i=1}^{n}(y_i - \widehat{w_0} - \widehat{w_1}x_i)x_i = 0 \rightarrow \widehat{w_1} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Ipython的魔法命令

```
%matplotlib inline
```

- 以上程式碼，在python的IDE如spyder、pycharm、VSC，都會顯示是invalid syntax。
- %matplotlib inline的作用是可以在Ipython編譯器比如jupyter notebook、jupyter lab或者 jupyter qtconsole裡直接使用內嵌繪圖，並且省略掉matplotlib的plt.show()呼叫。

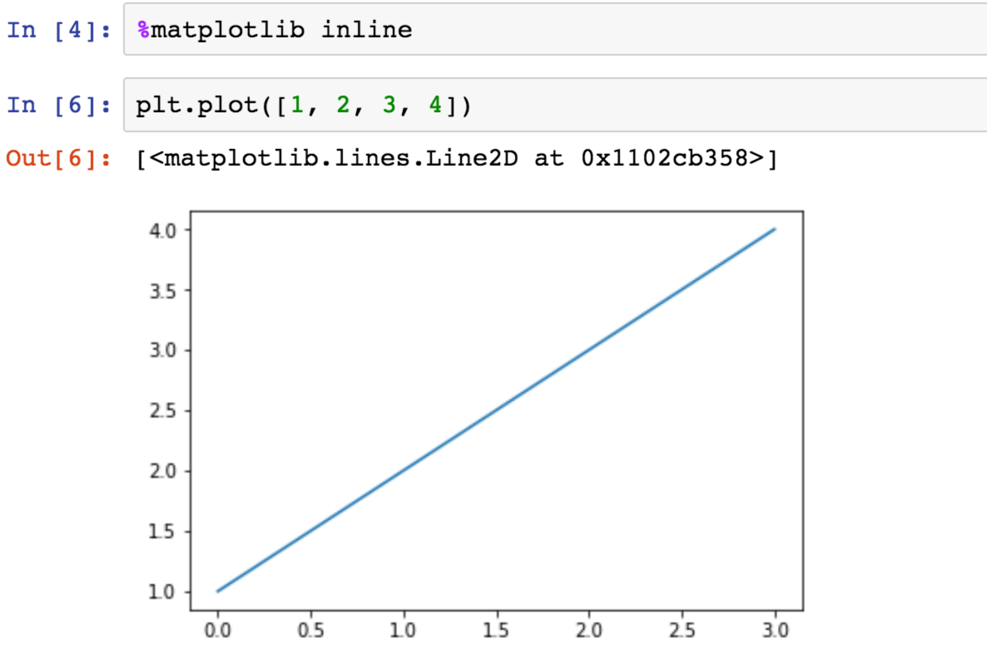


```
In [1]: import matplotlib.pyplot as plt
```

Matplotlib

```
In [2]: plt.plot([1, 2, 3, 4])
Out[2]: [<matplotlib.lines.Line2D at 0x1100cfe48>]

In [3]: plt.plot([1, 2, 3, 4])
        plt.show()
```



```
In [4]: %matplotlib inline

In [6]: plt.plot([1, 2, 3, 4])
Out[6]: [<matplotlib.lines.Line2D at 0x1102cb358>]
```

Ipython的魔法命令

- 有分成單行的魔法命令(Line magics)，還有多行的魔法命令(Cell magics)
- Line magics以一個%為首。
  - %lsmagic：列出所有magics命令
  - %conda：cell中安裝package，%conda install [pkgs]
  - %pip：在cell中使用pip指令，%pip install [pkgs]
  - %env：查看或設定環境變數，%env，%env var，%env var=value
  - %time：計時用
- Cell magics以兩個%為首。
  - %%latex：寫LATEX公式
  - %%script ：寫LATEX腳本
- 內建的魔法命令可以參考以下網址。
  https://ipython.readthedocs.io/en/stable/interactive/magics.html
- 請不要在非Ipython的環境下使用魔法命令。

載入模組

使用scipy

```
import numpy as np
from scipy.optimize import leastsq, least_squares
from scipy.optimize import least_squares
```

使用scikit-learn

```
from sklearn import linear_model
from sklearn import datasets
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.neural_network import MLPRegressor
```

資料準備

使用簡單的數據點

```
#利用list[]賦與np.array初始值
X = np.array([ 8.19,  2.72,  6.39,  8.71,  4.7 ,  2.66,  3.78])
Y = np.array([ 7.01,  2.78,  6.47,  6.71,  4.1 ,  4.23,  4.05])
```

隨機產生1000個數據點

```
# Random 1000 points by numpy
x_data = np.random.rand(1000).astype(np.float32)
Y_data = x_data * 0.1 + 0.1*np.random.rand(1000).astype(np.float32)
```

資料準備

使用diabetes dataset
UCI機器學習庫中的「Pima Indians Diabetes Database」，

```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
```

Data Set Characteristics:

Number of Instances

442

Number of Attributes

First 10 columns are numeric predictive values

Target

Column 11 is a quantitative measure of disease progression one year after baseline

Attribute Information

- age age in years
- sex
- bmi body mass index
- bp average blood pressure
- s1 tc, total serum cholesterol
- s2 ldl, low-density lipoproteins
- s3 hdl, high-density lipoproteins
- s4 tch, total cholesterol / HDL
- s5 ltg, possibly log of serum triglycerides level
- s6 glu, blood sugar level

https://www.kaggle.com/uciml/pima-indians-diabetes-database

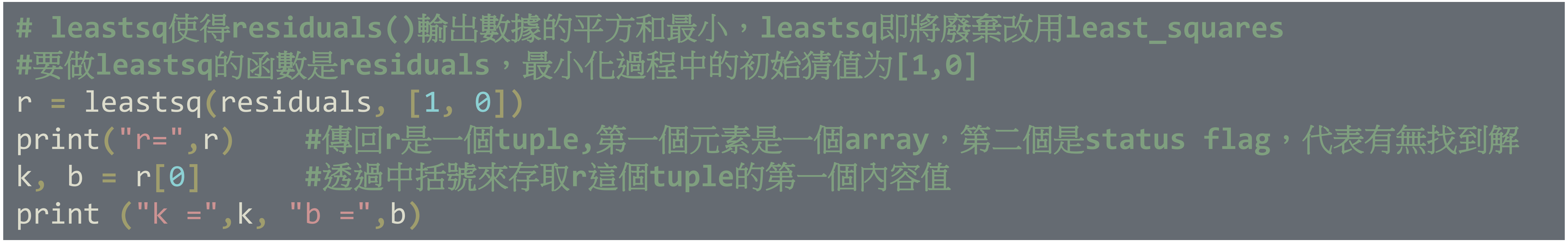
https://scikit-learn.org/stable/datasets/toy_dataset.html#diabetes-dataset

定義函數-計算殘差用

```
#計算以p為參數的直線和原始數據之間的誤差
def residuals(p):
    k, b = p    # Sequence unpacking, p is a list with two elements.
    return Y - (k*X + b)  #傳回殘差
```

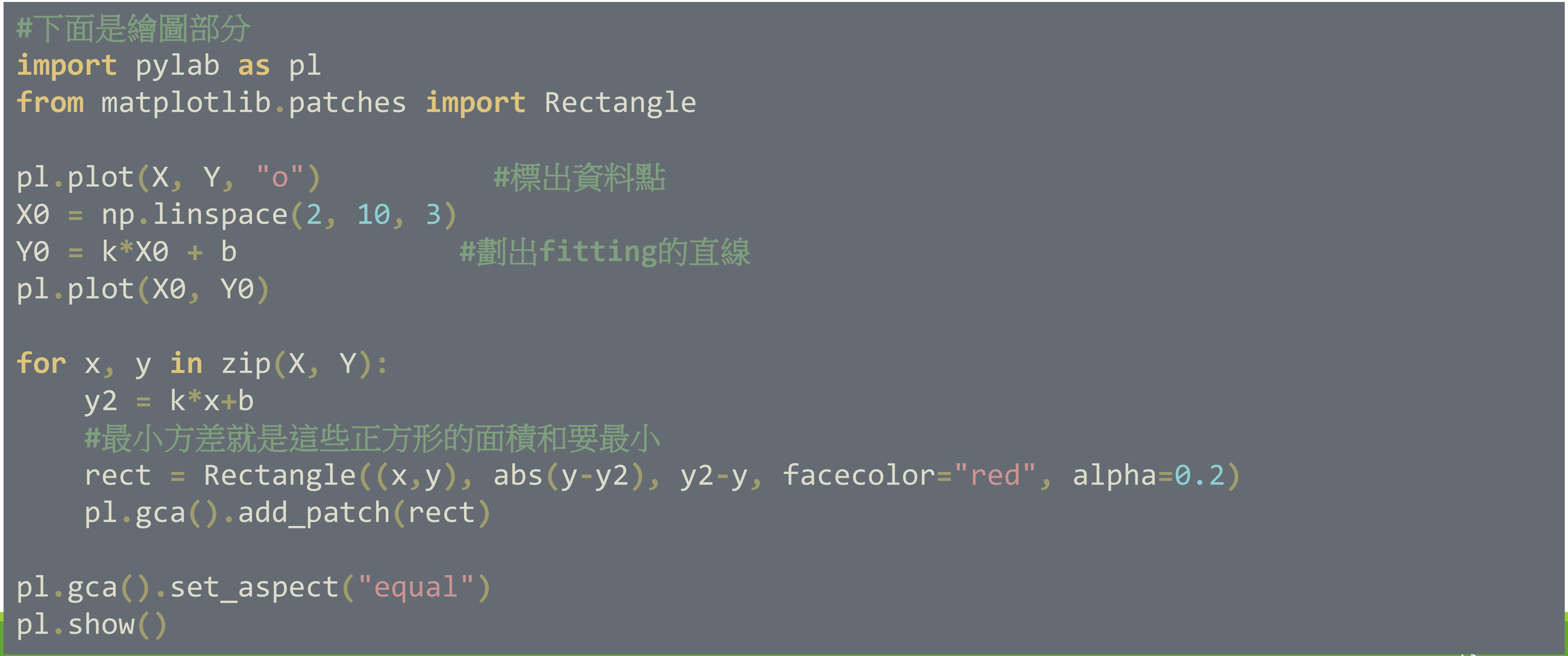
Optimization algorithm

- scipy.optimize.leastsq(即將廢棄)
- 傳入的函數不需要加上()以及arguments，只要有函數名稱就好。
- scipy.optimize.least_squares



```
# leastsq使得residuals()輸出數據的平方和最小，leastsq即將廢棄改用least_squares
#要做leastsq的函數是residuals，最小化過程中的初始猜值为[1,0]
r = leastsq(residuals, [1, 0])
print("r=",r)      #傳回r是一個tuple,第一個元素是一個array，第二個是status flag，代表有無找到解
k, b = r[0]        #透過中括號來存取r這個tuple的第一個內容值
print ("k =",k, "b =",b)
```

視覺化部分



```
#下面是繪圖部分
import pylab as pl
from matplotlib.patches import Rectangle

pl.plot(X, Y, "o")          #標出資料點
X0 = np.linspace(2, 10, 3)
Y0 = k*X0 + b             #劃出fitting的直線
pl.plot(X0, Y0)

for x, y in zip(X, Y):
    y2 = k*x+b
    #最小方差就是這些正方形的面積和要最小
    rect = Rectangle((x,y), abs(y-y2), y2-y, facecolor="red", alpha=0.2)
    pl.gca().add_patch(rect)

pl.gca().set_aspect("equal")
pl.show()
```

Regression algorithm

- sklearn.linear_model.LinearRegression

```
from sklearn import linear_model
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
```

Regression algorithm

- sklearn.sklearn.neural_network.MLPRegressor

```
from sklearn.neural_network import MLPRegressor
mlpr =  MLPRegressor(hidden_layer_sizes=(1, ), activation='identity',
solver='sgd', alpha=0.0001, batch_size='auto', learning_rate_init=0.0001,
max_iter=100000, random_state=49)

# Train the model using the training sets
mlpr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = mlpr.predict(diabetes_X_test)
```