Bài 1: Linear Regression

Bài toán: dự đoán cân nặng y dựa trên chiều cao x đã có.

1. Các bước làm:

- B1: Khai báo thư viện

```
from __future__ import division, print_function, unicode_literals
import numpy as np
import matplotlib.pyplot as plt
```

- B2: Load dữ liệu X, y. Visualize data
- B3: Chuẩn hóa dữ liệu -> X bar
- B4: Tính trọng số w theo công thức xác định

$$\mathbf{w} = \mathbf{A}^{\dagger}\mathbf{b} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{\dagger}\bar{\mathbf{X}}^T\mathbf{y}$$

- B5: Vẽ model (đường thẳng) sau khi xác định các trọng số của nó (Ý tưởng: Tìm 2 điểm trên đường thẳng đó. Tạo 2 list X_line = [x1, x2] cho tương ứng Y_line = [y1, y2], sau đó plt.plot(X_line, Y_line))
- B6: Dự đoán theo model đã tìm với trọng số đã xác định.

2. Kỹ thuật Python:

- Dữ liệu input dạng matrix, đọc dữ liệu vào dạng mảng 2 chiều, kiểu X = ([[1, 1, 1, …], [x1, x2, x3]]). Khi đó đây là matrix, và có thể dùng X.T (X transpose)
- Tương tự, dữ liệu output cũng ở dạng matrix Nx1, chứ không phải dạng mảng.
- Hiển thị đồ thị khi biết cặp các điểm tương ứng theo chỉ số vector, dưới dạng vector X, vector y:

  `plt.plot(X, y, 'ro')` ⇔ `plt.scatter(X, y)`

  `plt.axis([140, 190, 45, 75])`: định khoảng đồ thị 2 trên 2 trục

- Tạo matrix biết size_matrix và giá trị all elements:

```
np.ones((size_row, size_column), dtype=’int/float32/float64’)
np.zeros((size_row, size_column), dtype=’int/float32/float64’)
np.full((size_row, size_column), value,
dtype=’int/float32/float64’)
```

- Nối matrix:

  np.concatenate((name_matrix_1, name_matrix_2), axis = 0/1): axis = 0 tức nối dọc, axis = 1 tức nối ngang

- Nhân 2 ma trận:

  np.dot(matrix_1, matrix_2): matrix là dạng mảng 2 chiều.

  np.dot(scalar_1, scalar_2): nhân 2 số int/float

  np.dot(complex_1, comple_2): nhân 2 số phức

- Định thức ma trận vuông A:

  scalar_real_number = np.linalg.det(A)

- Ma trận nghịch đảo của matrix A (nếu A khả nghịch):

  np.linalg.pinv(A): trả về *matrix nghịch đảo* khi A khả nghịch và *matrix giả nghịch đảo* khi A không khả nghịch (ngoài ra có thể dùng **np.linalg.inv(A)** tuy nhiên nó chỉ trả về nghịch đảo khi A khả nghịch)

  np.eye(sizeOfMatrix): ma trận đơn vị

- Ma trận chéo
  - Tạo ma trận vuông _chéo: x = np.diag([x1, x2, …]): ma trận đường chéo, truyền vào 1 *list* phần tử
  - Lấy ma trận chéo từ ma trận vuông đã biết: list = np.diag(nameOfSquareMatrix), trả về list các phần tử trên đường chéo.
- Tạo mảng 1-D cách đều nhau, biết giá trị đầu/cuối và số lượng phần tử => trả về 1 list []:

```
np.linspace(start_value, end_value, NumOfElements, dtype='int/float32/float64')
```

- Áp dụng 1 hàm cho tất cả các phần tử của mảng:

```
x_predict = np.array([[170, 158]])

y_predict = w[0][0] + w[1][0] * x_predict
```