**BÁO CÁO**

case study 2

Linear Regression - Boston Housing Data

ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

TP. HCM, ngày 20 tháng 04 năm 2019



Lê Quốc Thịnh

17521087

Hà Quốc Tiến

17521122

Môn: Máy học

Lớp: KHTN2017

Mã lớp: CS114.J21.KHTN

GVHD: Huỳnh Thị Thanh Thương

Mục lục

[1. PHÁT BIỂU BÀI TOÁN 4](#_Toc6901494)

[1.1. Mục đích 4](#_Toc6901495)

[1.2. Đầu vào 4](#_Toc6901496)

[1.3. Đầu ra 4](#_Toc6901497)

[1.4. Cách đánh giá mô hình 4](#_Toc6901498)

[2. XÂY DỰNG MÔ HÌNH TỪ TẬP HUẤN LUYỆN 4](#_Toc6901499)

[2.1. Tiền xử lí 4](#_Toc6901500)

[2.2. Xây dựng mô hình Linear Regression 4](#_Toc6901501)

[2.2.1. Gradient descent 4](#_Toc6901502)

[2.2.2. Normal equations 5](#_Toc6901503)

[3. THỰC NGHIỆM ĐÁNH GIÁ 5](#_Toc6901504)

[3.1. Mean squared error (MSE) 5](#_Toc6901505)

[3.2. Kết quả đánh giá 5](#_Toc6901506)

[3.2.1. Gradient Descent 5](#_Toc6901507)

[3.2.2. Normal Equations 6](#_Toc6901508)

[3.3. Nhận xét 7](#_Toc6901509)

[4. LẬP TRÌNH CÀI ĐẶT 7](#_Toc6901510)

[4.1. Tool và thư viện đã dùng 7](#_Toc6901511)

[4.1.1. Anaconda 7](#_Toc6901512)

[4.1.2. Sklearn 7](#_Toc6901513)

[4.1.3. NumPy 7](#_Toc6901514)

[4.1.4. Matplotlib 7](#_Toc6901515)

[4.1.5. Numba 7](#_Toc6901516)

[4.1.6. Pandas 7](#_Toc6901517)

[4.2. Cách cài đặt tool và chạy chương trình 7](#_Toc6901518)

[4.2.1. Cài đặt tool 7](#_Toc6901519)

[4.2.2. Chạy chương trình 8](#_Toc6901520)

[4.3. Mã nguồn 8](#_Toc6901521)

[4.3.1. Tiền xử lí 8](#_Toc6901522)

[4.3.2. Gradient descent 9](#_Toc6901523)

[4.3.3. Normal equations 11](#_Toc6901524)

[4.4. Kết quả chạy 13](#_Toc6901525)

[4.4.1. Gradient descent 13](#_Toc6901526)

[4.4.2. Normal equation 13](#_Toc6901527)

[5. TÀI LIỆU THAM KHẢO 13](#_Toc6901528)

# PHÁT BIỂU BÀI TOÁN

## Mục đích

Xây dựng các mô hình dự đoán giá nhà ngoại ô Boston dựa trên phương pháp Hồi quy Tuyến tính (Linear Regression) và đánh giá chúng.

## Đầu vào

Dataset gồm thông tin của 506 căn nhà ngoại ô Boston với 13 đặc trưng nhận giá trị liên tục (đã bao gồm cả giá nhà) và 1 đặc trưng mang giá trị nhị phân.

## Đầu ra

Mô hình phỏng đoán về giá cả của các căn nhà ngoại ô Boston.

## Cách đánh giá mô hình

Đánh giá dựa trên Mean Squared Error (MSE).

# XÂY DỰNG MÔ HÌNH TỪ TẬP HUẤN LUYỆN

## Tiền xử lí

* Thực hiện việc chuẩn hóa dữ liệu theo công thức:

*Trong đó:*

*u là giá trị trung bình*

*s là độ lệch chuẩn*

* Chia Dataset thành 2 phần examples (x) và labels (y), với examples là các điểm dữ liệu còn lables là các nhãn tương ứng.
* Hai tập này được xáo trộn và tiếp tục chia thành các tập nhỏ hơn: xTrain, xTest, yTrain, yTest. Trong đó, xTrain và yTrain dùng để huấn luyện mô hình và 2 tập còn lại dùng để đánh giá. Tỉ lệ training set : test set là 3:1.

## Xây dựng mô hình Linear Regression

* Như đã biết, mô hình Linear regression có dạng: ,  
  hay còn được viết theo dưới ma trận là: , với:
* Việc xây dựng mô hình đồng nghĩa với việc tìm .

### Gradient descent

* Tìm bộ trọng số qua việc tối thiểu hàm mất mát , đồng nghĩa với việc tìm cực tiểu local cho hàm này. Phương pháp gradient descent giúp ta đạt được điều này qua điều chỉnh W từ từ, ngược với chiều tăng của *gradient của E* để đi đến tới điểm cực tiểu của E.
* Gradient của E chính là đạo hàm từng phần của E, do đạo hàm này đôi khi rất phức tạp nên việc tìm nghiệm đạo hàm bằng 0 để giải tìm cực tiểu là không khả thi nên ta dùng phương pháp gradient descent, với:
* Cập nhật W bằng cách đi từ từ ngược chiều của :

với được gọi là *learning rate*, là số thường nhỏ, qui định độ dài bước đi, tùy thuộc vào đặc điểm dữ liệu mà sẽ có giá trị khác nhau phù hợp nhất.

### Normal equations

* Để tối thiểu độ lỗi cho mô hình, cần tìm W sao cho: ,ta dễ dàng tìm ra được
* Sau khi thực hiện biến đổi (reshape) phần examples thành dạng , ta tìm dựa vào công thức trên.

# THỰC NGHIỆM ĐÁNH GIÁ

## Mean squared error (MSE)

* MSE là một hàm rủi ro được xác định bằng giá trị trung bình của bình phương các lỗi sai lệch (độ chênh lệch giữa giá trị dự đoán được và giá trị thực tế).

*Trong đó:*

*n: số lượng điểm dữ liệu*

*: giá trị thực tế*

*: giá trị dự đoán*

* Dựa vào công thức tính, ta thấy được MSE luôn mang giá trị không âm. Mô hình có MSE càng nhỏ, càng gần về 0 thì càng tốt (cả MSE\_train và MSE\_test), nhưng thực tế thì MSE không bao giờ đạt được giá trị 0 vì không có mô hình ứng dụng nào dự đoán đúng được 100% cả.

## Kết quả đánh giá

Dùng mean squared error để đánh giá và trực quan hóa tính chính xác của mô hình trong việc dự đoán giá nhà (so sánh giá nhà thực tế và giá dự đoán) qua biểu đồ phân tán, các điểm càng gần đường thằng màu xanh lục nghĩa các examples đó có giá càng được dự đoán gần với giá thực tế.

Thêm plot và độ lỗi ở đây

## Nhận xét

* Việc sử dụng phương pháp Normal equations cho bộ dữ liệu này không gây ra tình trạng overfitting với MSE\_train và MSE\_test xấp xỉ nhau.

# LẬP TRÌNH CÀI ĐẶT

## Tool và thư viện đã dùng

### Anaconda

Anaconda là một bản phân phối miễn phí và mã nguồn mở của Python. Nó bao gồm nhiều thư viện phổ biến của Python và một số công cụ như Spyder IDE, Jupyter notebook,... Anaconda hướng đến việc quản lí các package một cách đơn giản, phù hợp với mọi người. Hệ thống quản lí package của Anaconda là Conda. Bản phân phối Anaconda tính đến nay được sử dụng hơn 6 triệu người, và đính kèm hơn 250 gói package khoa học dữ liệu phù hợp cho Windows, Linux và MacOS.

### Sklearn

Scikit-learn là thư viện mã nguồn mở mang tính đơn giản và hiệu quả cho việc khai thác dữ liệu và phân tích dữ liệu, đặc biệt thư viện cung cấp rất nhiều công cụ cho máy học. Nó được xây dựng dựa trên numpy, scipy và matplotlib nên rất tương thích với các thư viện này.

### NumPy

Numpy là thư viện quan trọng cho việc tính toán khoa học trong Python. Nó cung cấp mảng đa chiều hiệu năng cao và những công cụ để làm việc với mảng.

### Matplotlib

Matplotlib là một thư viện dùng để vẽ graph 2D trong Python, cho ra những graph chất lượng cao và có thể được dùng trong các Python shell, IDE khác nhau. Matplotlib có thể tạo plots, histograms, bar charts, scatterplots,... với vài dòng code.

### Numba

Numba dịch code hàm Python thành mã máy ở runtime, dùng thư viện compiler LLVM. Thuật toán số học viết trong Python được compile bởi Numba có thể tiệm cận tốc độ của C hay FORTRAN. Numba được thiết kế để làm việc với mảng NumPy một cách hiệu quả.

### Pandas

Pandas là một thư viện mã nguồn mở cung cấp cấu trúc dữ liệu hiệu năng cao, dễ sử dụng và các công cụ phân tích dữ liệu cho ngôn ngữ Python.

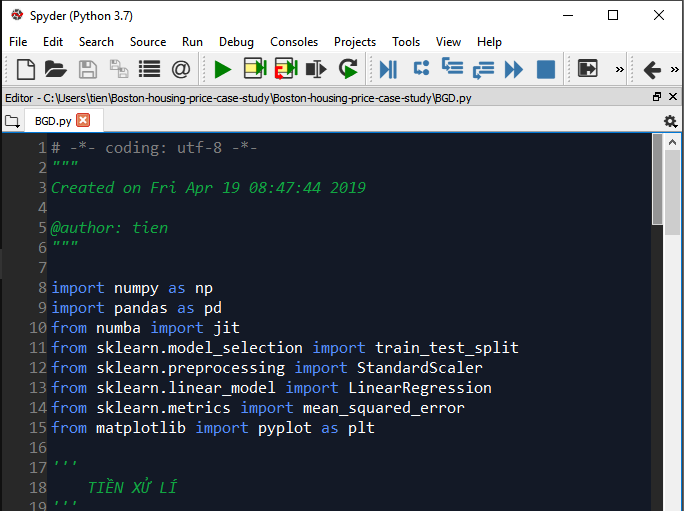
## Cách cài đặt tool và chạy chương trình

### Cài đặt tool

* Cài đặt Anaconda: <https://docs.anaconda.com/anaconda/install/>
* Thư viện Sklearn, NumPy, Matplotlib, Numba và Pandas đều đã được bao gồm trong Anaconda.

### Chạy chương trình

Soạn thảo code trên IDE Spyder như trên hình:



Và sau đó chọn nút tam giác màu xanh lá cây  để chạy.

## Mã nguồn

### Gradient Descent

#### Batch

import numpy as np

from numba import jit

'''

BGD

'''

# hàm tính deltaE

@jit

def dE(yHat, y, x):

m = x.shape[0]

deltaE = np.full((x.shape[1]), 0.0, dtype='float')

for i in range(x.shape[1]):

xi = x[:,i]

deltaE[i] = 2/m\*np.sum((yHat - y)\*xi)

return deltaE

# hàm tính hàm giả thiết H

@jit

def h(x, w):

return np.sum(x\*w, axis = 1)

# hàm tính độ lỗi

@jit

def Er(x, y, w):

m = x.shape[0]

yHat = h(x, w)

return (1/m)\*np.sum((y-yHat)\*\*2)

# hàm thêm cột 1 vào ma trận examples

@jit

def add1Col(x):

m = x.shape[0]

return np.hstack((np.full((m, 1), 1), x))

# hàm chạy batch gradient descent

@jit

def BGD(x, y, learningRate, w = None, epsilon= 0.001):

if w == None:

w = np.zeros((x.shape[1]+1))

x = add1Col(x)

y = np.array(y).flatten()

E = float('Inf')

Enext = 0

wnext = np.array([])

while True:

yHat = h(x, w)

deltaE = dE(yHat, y, x)

wnext = w - learningRate\*deltaE

Enext = Er(x, y, wnext)

if abs(E - Enext) < epsilon:

E = Enext

w = wnext

break

E = Enext

w = wnext

return w

#### Stochastic

import numpy as np

from numba import jit

'''

SGD

'''

# hàm tính deltaE

@jit

def dE(yHat, y, x):

deltaE = np.full((x.shape[0]), 0.0, dtype='float')

for i in range(x.shape[0]):

xi = x[i]

deltaE[i] = 2\*(yHat-y)\*xi

return deltaE

# hàm thêm cột 1 vào ma trận examples

@jit

def add1Col(x):

m = x.shape[0]

return np.hstack((np.full((m, 1), 1), x))

# hàm chạy batch gradient descent

@jit

def SGD(X, Y, learningRate = 0.0005, w = None, epsilon= 1e-5, numberOfIterations = 100):

if w == None:

w = np.zeros((X.shape[1]+1))

X = add1Col(X)

Y = np.array(Y).flatten()

E = float('Inf')

Enext = 0

wnext = np.array([])

done = False

i = 0

while not done and i < numberOfIterations:

for Xi, Yi in zip(X, Y):

yHat = np.sum(Xi\*w)

deltaE = dE(yHat, Yi, Xi)

wnext = w - learningRate\*deltaE

Enext = (Yi-yHat)\*\*2

if abs(E - Enext) < epsilon:

E = Enext

w = wnext

done = True

break

E = Enext

w = wnext

i += 1

return w

#### Minibatch

import numpy as np

from numba import jit

'''

MBGD

'''

# hàm tính deltaE

@jit

def dE(yHat, y, x):

m = x.shape[0]

deltaE = np.full((x.shape[1]), 0.0, dtype='float')

for i in range(x.shape[1]):

xi = x[:,i]

deltaE[i] = 2/m\*np.sum((yHat - y)\*xi)

return deltaE

# hàm tính hàm giả thiết H

@jit

def h(x, w):

return np.sum(x\*w, axis = 1)

# hàm tính độ lỗi

@jit

def Er(x, y, w):

m = x.shape[0]

yHat = h(x, w)

return (1/m)\*np.sum((y-yHat)\*\*2)

# hàm thêm cột 1 vào ma trận examples

@jit

def add1Col(x):

m = x.shape[0]

return np.hstack((np.full((m, 1), 1), x))

# hàm chạy batch gradient descent

@jit

def MiniBatch(x, y, learningRate, w):

yHat = h(x, w)

deltaE = dE(yHat, y, x)

wnext = w - learningRate\*deltaE

Enext = Er(x, y, w)

return wnext, Enext

def MBGD(x, y, learningRate, w = None, epsilon= 0.001, size = 30):

if w == None:

w = np.zeros((x.shape[1]+1))

x = add1Col(x)

y = np.array(y).flatten()

E = float('Inf')

Enext = 0

wnext = w

cond = True

a = 0

while cond:

for i in range(0,y.shape[0],size):

wnext, Enext = MiniBatch(x[i:i+size], y[i:i+size], learningRate, wnext)

if abs(Enext - E) < epsilon:

E = Enext

w = wnext

cond = False

break

E = Enext

w = wnext

a = a + 1

return w

### Normal Equations

from BGD import add1Col

import numpy as np

'''

NE

'''

def NormalEquation(x, y):

x = add1Col(x)

y = y.flatten()

w = np.linalg.inv((x.transpose().dot(x))).dot(x.transpose().dot(y))

return w

## Kết quả chạy

### Batch gradient descent

### Minibatch gradient descent

### Stochastic gradient descent

### Normal equation

# TÀI LIỆU THAM KHẢO

* [Preprocessing with sklearn: a complete and comprehensive guide](https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9)
* [Pandas Documentation](https://pandas.pydata.org/pandas-docs/stable/reference/frame.html)
* [SkLearn Documentation](https://scikit-learn.org/stable/index.html)
* [NumPy Documentation](https://docs.scipy.org/doc/numpy/index.html)
* [Matplotlib Documentation](https://matplotlib.org/)
* [Machine learning Cơ bản](https://machinelearningcoban.com/)