Báo cáo đồ án 3

–Wine evaluation by linear regression–

Mục lục

1 Thông tin sinh viên 1
2 Problem A 1
2.1 Hàm findTheta 2
2.2 Hàm findNorm 2
2.3 Hàm trainAndTest(train_set,test_set,y_name) 3
3 Problem B 4
3.1 Hàm crossValidation 5
3.2 Hàm createNewFrame 5
4 Problem C 6
4.1 Hàm testAllCase 6
5 Tài liệu tham khảo 8

1 Thông tin sinh viên

Họ và tên: Đỗ Vương Phúc
Mã số sinh viên: 19127242
Lớp: 19CLC4

2 Problem A

Sử dụng toàn bộ 11 đặc trưng để làm model $y = \sum_{i=1}^{11} \theta_i . x_i$. Ở đây, để chúng ta có thể xử lý được dữ liệu từ file csv, ta sử dụng thư viện pandas. Ngoài ra, để xử lý ma trận thì sử dụng thêm thư viện numpy.

Đầu tiên, chúng ta cần viết hai hàm để tìm hệ số của bài toán bình phương tối tiểu và tính xem sai số bình phương tổng phần dư là bao nhiêu. Bài toán bình phương tối tiểu ở đây là tìm $x = \begin{bmatrix} \theta_1 \\ \cdots \\ \theta_n \end{bmatrix}$ sao cho

RSS là bình phương tổng phần dư (residual sum of squared) đạt nhỏ nhất:

$$RSS = || Ax - b ||^2$$

Và theo như ta đã chứng minh được (sẽ không đề cập ở đây) cho bài toán bình phương tối tiểu thì:

$$\underset{x \in R^n}{argmin}\, RSS = (A^T A)^{-1} A^T b$$

Do đó chúng ta định nghĩa hai hàm:

2.1 Hàm findTheta

- Đầu vào: Hai ma trận A và b là tham số cho bài toán bình phương tối tiểu
- Đầu ra: Tham số x để tối ưu bình phương tổng phần dư
- Cách thức thực hiện:
  - Bước 1: Tính A^T bằng hàm np.transpose
  - Bước 2: Tính $P_1 = (A^T A)^{-1}$ bằng np.linalg.inv
  - Bước 3: Tính $P_2 = A^T b$
  - Bước 4: Trả về kết quả $P_1.P_2$

2.2 Hàm findNorm

- Đầu vào: Ba ma trận A, x và b là tham số cho bài toán bình phương tối tiểu
- Đầu ra: Giá trị Frobenius norm của $Ax - b$ hay nghĩa là căn của RSS
- Cách thức thực hiện:
  - Bước 1: Tính $Ax - b$
  - Bước 2: Tính và trả kết quả bằng np.linalg.norm

Sau khi định nghĩa hai hàm trên, ta đọc dữ liệu từ file csv bằng thư viện pandas vào dataframe. Ở đây, chúng ta cần lưu ý một số cấu trúc dữ liệu của thư viện pandas. (Tên file là “wine.csv” và ngăn cách bởi dấu ‘;’):

```
df = pd.read_csv('wine.csv',sep=';')
```

Khi ta đọc một frame dữ liệu lên thì nó sẽ là kiểu dữ liệu DataFrame, còn khi trích xuất 1 trường hoặc thuộc tính (field/attribute) của một DataFrame thì sẽ là một Series.

Đối với một DataFrame, ta có thể chuyển đổi (convert) nó về kiểu danh sách của danh sách (list of list) bằng thuộc tính values:

```
df.values
```

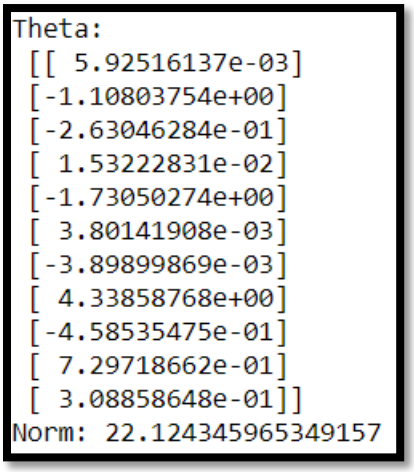
Đối với một Series, nếu ta sử dụng thuộc tính values, ta sẽ nhận được một danh sách (list) dữ liệu. Nên để chuyển thành một ma trận (list of list) thì ta có thể ép kiểu (cast) về DataFrame bằng phương thức to_frame() rồi dùng thuộc tính values.

Đối với vấn đề A, ta sẽ sử dụng bộ data gốc làm data để train thông số x cũng như là dùng để test lại xem độ sai số của nó là bao nhiêu. Ở đây, chúng ta viết một hàm trainAndTest để train và thực hiện test một bộ dữ liệu bằng mô hình hồi quy tuyến tính.

2.3 Hàm trainAndTest(train_set,test_set,y_name)

- Đầu vào: Hai bộ data (DataFrame) dùng để train và dùng để test cùng với tên của cột của biến kết buộc.
- Đầu ra: Bộ tham số x để tối ưu dữ liệu train cho bài toán hồi quy tuyến tính và độ sai số sau khi kiểm định (train) với test_set bằng phương pháp Frobineus norm.
- Cách thức thực hiện:
    - Bước 1: Trích xuất cột dữ liệu kết buộc và đưa về dạng ma trận cho bộ train_set và test_set. Đây là ma trận b của bài toán bình phương tối tiểu.
    - Bước 2: Sao chép bộ dữ liệu và xóa đi cột kết buộc để tạo thành ma trận cột các biến tự do. Đây là ma trận A của bài toán bình phương tối tiểu cho hàm hồi quy tuyến tính.
    - Bước 3: Tính tham số x là bộ theta để thu được mô hình tối ưu với bộ train_set
    - Bước 4: Tính độ sai số norm với bộ test_set
    - Bước 5: Trả về kết quả của bộ theta và độ sai số

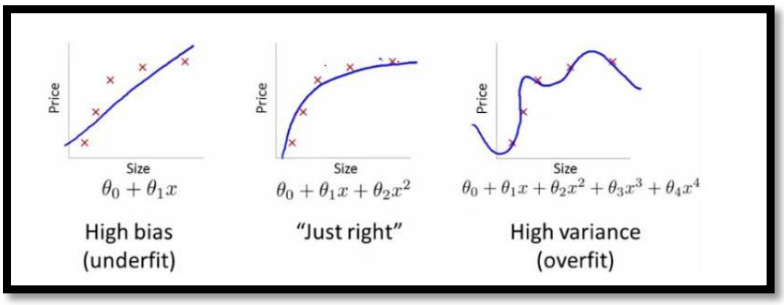
Như vậy ở vấn đề A, ta dễ dàng sử dụng hàm trainAndTest với bộ dữ liệu dùng để train cũng là bộ dữ liệu dùng để test với trường kết buộc là "quality". Do đó ta thu được kết quả như sau:



Ta thu được tham số theta cho từng trường dữ liệu như trên. Ví dụ như với trường fixed acidity sẽ là $(5.9e-3)(\ fixed\ acidity)$ và tương tự cho các trường còn lại. Với độ sai số là 22.12

3 Problem B

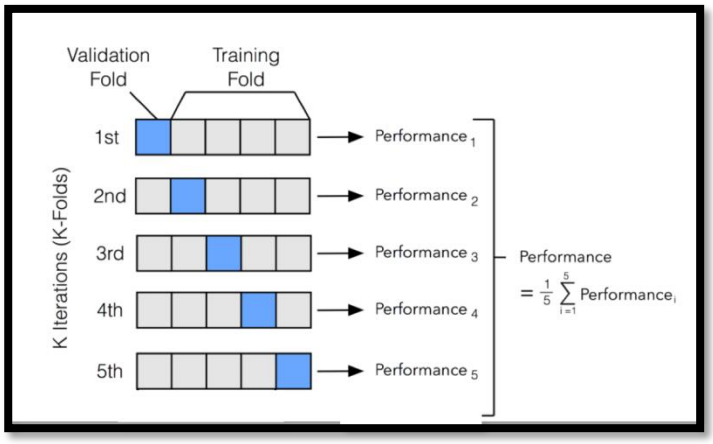
Ở vấn đề A, chúng ta có thể thấy bộ dữ liệu dùng để train hệ số x (bộ theta) cũng được dùng để test model của chúng ta luôn. Điều này dẫn đến kết quả sai số mang tính chủ quan. Trên thực tế, những dữ liệu mới khi thu thập thường sẽ không giống như dữ liệu khi train. Điều này cho thấy model của chúng ta có thể bị đánh giá sai khi chúng ta quá chủ quan (overfit):



Như ta có thể thấy ở hình trên, cùng một bộ dữ liệu, với các mô hình khác nhau ta có thể thấy trường hợp thứ 3 (overfit) cho ra độ sai số rất thấp nhưng lại không mang tính khách quan, mô hình này khó có thể sử dụng để dự đoán (predict).

Do đó, để có thể đánh giá các mô hình một cách khách quan hơn, ta sử dụng phương pháp kiểm tra chéo (cross validation). Ở phương pháp này, ta có nhiều biến thể như LOOCV, Stratified,… Tuy nhiên, trong tài liệu này chỉ sử dụng phương pháp K-Fold.

K-Fold là một phương pháp chia bộ dữ liệu của chúng ta thành K bộ dữ liệu nhỏ (fold). Sau đó ta lần lượt chọn 1 trong những fold được chia làm test set, còn những fold còn lại tạo thành train set. Như vậy ta có thể thực hiện K lần train và test khác nhau bằng cách chọn các fold:



Với mỗi một cách chọn test set (K iteration), chúng ta sẽ thu được một bộ theta (tham số x) để tối ưu train set và một sai số (Performance). Sau đó, ta có thể tính trung bình các sai số để thu được sai số cuối cùng. Đối với bộ tham số theta, ta có thể dùng toàn bộ dữ liệu để train lại model hoặc sử dụng bộ theta cho ra sai số là nhỏ nhất. Trên thực tế, phương pháp K-Fold thường sử dụng với $k = 5$ hoặc $k = 10$.

Ta có thể thấy phương pháp trên cho ta một kết quả sai số rất khách quan vì dữ liệu dùng để test không nằm trong dữ liệu dùng để train model.

Ở vấn đề B, ta sẽ kiểm định từng model với mỗi model là mô hình hồi quy tuyến tính của duy nhất một thuộc tính. Để khách quan, ta sẽ sử dụng phương pháp K-Fold Cross Validation để đánh giá từng mô hình. Đầu tiên ta sẽ viết một hàm crossValidation để thực hiện kiểm định chéo một mô hình hồi quy

3.1 Hàm crossValidation

- Đầu vào: Một dataframe thể hiện bộ dữ liệu cần thực hiện, tên của trường kết buộc và số lượng k folds muốn chia
- Đầu ra: bộ số theta được chọn (với data train có sai số nhỏ nhất) và sai số trung bình
- Cách thực hiện:
    - Bước 1: Tạo một frame là copy của dataframe ban đầu
    - Bước 2: Chia nhỏ frame thành k folds
    - Bước 3: Khởi tạo các biến dữ liệu sNorm (tổng sai số), listNormTheta (danh sách giá trị norm và bộ theta đối với từng vòng lặp K-iteration)
    - Bước 4: Với mỗi cách chọn test set ($0 \leq i < k$):
        - Bước 4.1: Tạo train set là toàn bộ các fold và xóa đi fold thứ i
        - Bước 4.2: Tạo test set là fold thứ i
        - Bước 4.3: Thực hiện ghép các fold lại thành một DataFrame bằng pandas.concat
        - Bước 4.4: Thực hiện tính bộ theta và norm bằng hàm trainAndTest bên trên
        - Bước 4.5: Thêm kết quả vừa thu được vào listNormTheta là tuple của norm và theta (Lưu ý phải đúng thứ tự)
        - Bước 4.6: Cộng giá trị sai số norm vào sNorm
    - Bước 5: Chọn tham số theta sao cho norm nhỏ nhất bằng hàm min (trả về tuple có dữ liệu đầu tiên là nhỏ nhất, ở đây là giá trị norm nhỏ nhất)
    - Bước 6: Tính norm trung bình
    - Bước 7: Trả kết quả về một bộ theta được chọn và giá trị norm trung bình

Ngoài ra, ta cần tạo được dataframe chỉ chứa trường kết buộc và trường tự do cần thiết. Do đó, ta định nghĩa thêm hàm createNewFrame.

3.2 Hàm createNewFrame

- Đầu vào: Một dataframe là dữ liệu gốc, một danh sách tên của các cột là trường tự do và tham số là tên của trường kết buộc
- Đầu ra: Một dataframe sau khi xóa các cột không cần sử dụng
- Cách thực hiện:
    - Bước 1: Tạo một bản frame là copy của dataframe nhập vào
    - Bước 2: Với mỗi cột col trong frame ta thực hiện:
        - Bước 2.1: Kiểm tra xem cột col có phải là trường kết buộc hoặc là trong danh sách cột các trường tự do cần giữ hay không. Nếu không thì xóa cột đó khỏi frame

- Bước 3: Trả về frame sau khi thực hiện xóa

Từ đây, ta có thể thực hiện vấn đề B như sau. Với mỗi một thuộc tính, ta thực hiện tạo một dataframe mới có trường tự do là thuộc tính đó và trường kết buộc là "quality" bằng hàm createNewFrame. Sau đó, ta thực hiện kiểm định Cross Validation cho frame đó rồi lưu kết quả thành một tuple. Tương tự, ta thực hiện với các cột khác. Cuối cùng, ta chọn ra tuple có norm trung bình nhỏ nhất là kết quả cần tìm:

```
= Best result =
Norm:  7.918652557961548
Theta: [[0.54298689]]
Attribute: alcohol
```

Và đây là kết quả của chương trình, ta thấy được là trường dữ liệu alcohol sẽ cho sai số trung bình nhỏ nhất khi thực hiện kiểm định 10-folds. Vậy model của chúng ta sẽ là $quality = 0.543(alcohol)$.

4 Problem C

Ở vấn đề cuối cùng, để đưa ra được hàm hồi quy tốt nhất thông thường ta sẽ vẽ các dữ liệu bằng các scatter các data lên để đánh giá được mối tương quan giữa các trường với nhau. Tuy nhiên, đối với 11 trường dữ liệu thì sẽ khó mà vẽ được. Do đó, giải pháp chúng ta có thể đưa ra là thử từng tổ hợp của các trường dữ liệu và xây dựng mô hình tuyến tính cho chúng. Cụ thể hơn, chúng ta thực hiện sinh nhị phân dãy 11 bit tương ứng với từng trường dữ liệu tự do của bài toán. Nếu như bit ở vị trí $i = 1$ nghĩa là ta sẽ chọn thuộc tính thứ 1, và ngược lại.

Ví dụ khi sinh dãy bit là 11101001000 nghĩa là ta chọn các trường dữ liệu: fixed acidity, volatile acidity, citric acid, chlorides, density. Vậy thì mô hình hồi quy tuyến tính của ta sẽ có dạng:

$$quality = \sum_{i=1}^{11} \theta_i . x_i . b_i$$

Với b_i là bit nhị phân ở vị trí thứ i. Do đó, ta sẽ thực hiện thử nghiệm $2^{11} = 2048$ trường hợp khác nhau của các tổ hợp. Để sinh được dãy nhị phân 11 bits, ta sử dụng phương pháp đệ qui quay lui (backtracking) sẽ không để cập sâu ở đây. Thông qua hàm testAllCase, ta thực hiện:

4.1 Hàm testAllCase

- Đầu vào: Một danh sách để trả kết quả, danh sác các trường dữ liệu tự do cần kiểm tra, vị trí k là trường dữ liệu đang xét trong dataframe. Mặc định danh sách các trường đang xét là rỗng và bắt đầu tại $k = 0$.
- Đầu ra: Không, kết quả sẽ trả về thông qua danh sách $result$ là các tuple (bộ) dữ liệu bao gồm giá trị norm, bộ tham số theta để tối ưu model và danh sách các trường dữ liệu đang xét
- Cách thức thực hiện:

- Bước 1: Tạo một danh sách copy của danh sách đang xét (để không ảnh hưởng để quá trình backtracking)
- Bước 2: Nếu vị trí đang xét là số thuộc tính (nghĩa là đã tạo được đủ bit) thì đến bước 5
- Bước 3: Với mỗi giá trị i là 0 hoặc 1 ($0 \le i \le 1$)
  - Bước 3.1: Nếu $i = 1$ ta thêm tên của thuộc tính k vào danh sách các trường tự do đang xét
- Bước 4: Thực hiện lại hàm testAllCase với vị trí kế tiếp $k+1$ và đi đến bước 8
- Bước 5: Tạo một frame dữ liệu với các trường tự do trong danh sách đã tạo trong quá trình backtrack
- Bước 6: Tính các giá trị theta và sai số norm trung bình tương ứng với frame dữ liệu trên bằng phương pháp K-Fold.
- Bước 7: Thêm một tuple (bộ) dữ liệu bao gồm norm trung bình, bộ theta và danh sách các trường tự do đang xét
- Bước 8: Kết thúc

Như vậy, khi ta gọi hàm testAllCase thì ta sẽ thu được một danh sách các bộ dữ liệu của tất cả tổ hợp các thuộc tính. Sau đó ta tìm ra tổ hợp nào mà có sai số trung bình nhỏ nhất:

```
= Best result =
Norm:  7.09667876204013
Theta:
[[-1.05817351e+00]
 [-2.65570255e-01]
 [-1.68571405e+00]
 [-3.43210474e-03]
 [ 4.29851787e+00]
 [-4.60521429e-01]
 [ 7.04482782e-01]
 [ 3.24614128e-01]]
Attribute
['volatile acidity', 'citric acid', 'chlorides', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol']
```

Kết quả ở đây cho thấy khi ta lập một hàm hồi quy tuyến tính theo tám thuộc tính volatitle acidity, citric acid, chlorides, total sulfur dioxide, density, pH, sulphates và alcohol thì ta thu được một mô hình có độ sai số là xấp xỉ 7.1.

Nếu ta so sánh với vấn đề A khi dùng K-fold, thì tổ hợp tám thuộc tính trên vẫn là tốt hơn cả 11 thuộc tính:

```
Theta:
 [[ 3.83862334e-03]
 [-1.05270482e+00]
 [-2.79617426e-01]
 [ 1.43406488e-02]
 [-1.65757834e+00]
 [ 2.83437290e-03]
 [-4.09579872e-03]
 [ 4.21256719e+00]
 [-4.44796104e-01]
 [ 7.02641261e-01]
 [ 3.19940378e-01]]
Norm: 7.126524340809087
```

5 Tài liệu tham khảo

1. A Gentle Introduction to k-fold Cross-Validation - Jason Brownlee - Machine Learning Mastery – 23 Tháng 5, 2018 (machinelearningmastery.com)
2. Train/Test Split and Cross Validation in Python – Adi Bronshtein – Toward datascience – 17 Tháng 5, 2017 (towardsdatascience.com)
3. Overfitting, regularization, eigenvalues and eigenvectors of machine learning – Programmer Sought (programmersought.com)
4. Model selection – Ethen – Github ethen8181