**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

-------\*\*\*-------

****

**ĐỒ ÁN 2**

**RIDGE REGRESSION**

**Giảng viên hướng dẫn: Trần Nhật Quang**

**Sinh viên thực hiện: nhóm 2**

**Nguyễn Ngọc Quý - 16110195**

**Trần Bảo Đan - 16110044**

**TP. Hồ Chí Minh - 23 tháng 5 năm 2019**

**MỤC LỤC**

[Thế nào là đạo văn? 1](#_Toc17349)

[1. REGRESSION 2](#_Toc5358)

[1.1. Linear regression 2](#_Toc16800)

[1.2. Multivariate Linear Regression 8](#_Toc15541)

[1.3. Polynomial Regression 9](#_Toc25322)

[1.4. Normal Equation (Công thức chuẩn) 1](#_Toc6436)0

[1.5. Overfitting 1](#_Toc28249)1

[2. RIDGE REGRESSION 1](#_Toc28250)3

[2.1. Giới thiệu 1](#_Toc6211)3

[2.2. Regularization](#_Toc24299) 13

[2.3. Validation](#_Toc23928) 14

[3. LẬP TRÌNH 40](#_Toc19254)

[3.1. Dataset 1](#_Toc25339)6

[3.2. Scikit-Learn 1](#_Toc24862)6

[3.3. Cài Đặt 1](#_Toc7439)7

### Thế nào là đạo văn?

#### - Định nghĩa đạo văn

Đạo văn nghĩa là:

-Copy, hình thành ý tưởng, ngôn từ hay sản phẩm mới dựa trên ý tưởng của người khác.

-Sử dụng lời văn, sản phẩm mà không chú thích nguồn hoặc không chú thích nguồn rõ ràng.

-Tạo ra sản phẩm chuyển hóa ý tưởng từ nguồn đã có sẵn.

#### - Những điều nên/không nên làm

Không nên làm :

-Không ghi nguồn hoặc ghi nguồn không chi tiết.

-Copy nguyên bản lời văn, lời phát biểu của người khác

-Sử dụng ý tưởng của người khác mà chưa có sự đồng ý

Nên làm:

-Ghi rõ nguồn gốc trích dẫn, sản phẩm

-Sử dụng ý tưởng khi có sự đồng ý của người chủ sản phẩm

#### - Lời cam kết

“Chúng tôi xin cam đoan đồ án này do chính chúng tôi thực hiện. Chúng tôi không sao chép, sử dụng bất kỳ tài liệu, mã nguồn của người khác mà không ghi rõ nguồn gốc. Chúng tôi xin chịu hoàn toàn trách nhiệm nếu vi phạm.”

## REGRESSION

### Linear regression

#### Giới thiệu

Linear Regression (Hồi Quy Tuyến Tính) thuộc nhóm Supervised learning ( Học có giám sát ). Hồi quy chính là một phương pháp thống kê để thiết lập mối quan hệ giữa một biến phụ thuộc và một nhóm tập hợp các biến độc lập.

Ví dụ : Tuổi = 3+ chiều cao\*8 + trọng lượng+10

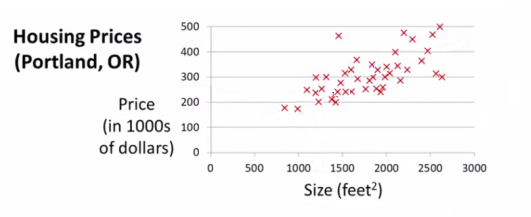
#### Học có giám sát là gì?

**Học có giám sát** là một kĩ thuật của ngành [học máy](https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_m%C3%A1y) để xây dựng một hàm (*function*) từ dữ liệu huấn luyện. [Dữ liệu huấn luyện](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_hu%E1%BA%A5n_luy%E1%BB%87n) bao gồm các cặp gồm đối tượng đầu vào (thường dạng vec-tơ), và đầu ra mong muốn. Đầu ra của một hàm có thể là một giá trị liên tục (gọi là [hồi qui](https://vi.wikipedia.org/wiki/Ph%C3%A2n_t%C3%ADch_h%E1%BB%93i_quy)), hay có thể là dự đoán một nhãn phân loại cho một đối tượng đầu vào (gọi là [phân loại](https://vi.wikipedia.org/wiki/Ph%C3%A2n_lo%E1%BA%A1i_b%E1%BA%B1ng_th%E1%BB%91ng_k%C3%AA)). Nhiệm vụ của chương trình học có giám sát là dự đoán giá trị của hàm cho một đối tượng bất kì là đầu vào hợp lệ, sau khi đã xem xét một số ví dụ huấn luyện (nghĩa là, các cặp đầu vào và đầu ra tương ứng). Để đạt được điều này, chương trình học phải tổng quát hóa từ các dữ liệu sẵn có để dự đoán được những tình huống chưa gặp phải theo một cách "hợp lý" [1]

* Ví dụ 1 :

Cách học khi nhỏ của mỗi người. Ta được thầy cô hay bố mẹ đưa cho bảng chữ cái và hướng dẫn đây là C, đây là D. Sau nhiều lần thì ta có thế nhận biết được các chữ cái. Đâu là C, đầu là D, đầu là A mỗi khi ta đọc một sách gì đó mà ở trong có các chữ cái đó.

* Ví dụ 2 : Dự đoán giá nhà

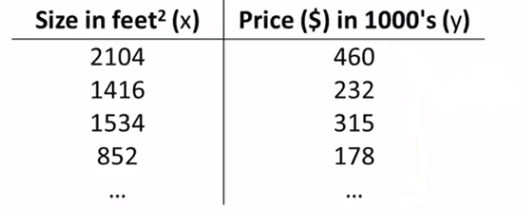


*Hình 1: Biểu dồ dự đoán giá nhà [4]*

Các ngôi nhà kích thước khác nhau sẽ được bán với gián khác nhau. Giả dụ bạn đang muốn bán một canh nhà và bạn thấy nếu canh nhà đó có kích thước 1250 feet vuông thì khi nhìn vào biểu đồ này, ta có thể bán với giá khoảng 220.000$ .

Đây là ví dụ về học có giám sát vì bạn đã đưa ra câu trả lời chính xác cho mỗi dữ liệu khác nhau. Cụ thể ở đây là bạn đã đưa ra giá nhà thực tế của căn nhà.

#### Các kí hiệu

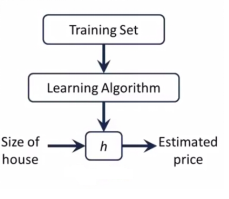
**

*Hình 2: Dữ liệu giá nhà theo kích thước [4]*

* Chúng ta có bộ dữ liệu như ở trên và gọi là : Tập huấn luyện (training set)
* Ký hiệu số lượng cho mỗi huấn luyện : m
* Các biến đầu vào : x (feature)
* Các biến đầu ra : y ( target variable)
* Một mẫu huấn luyện : (x,y)
*  : mẫu huấn luyện thứ i

#### Hàm hypothesis (h)

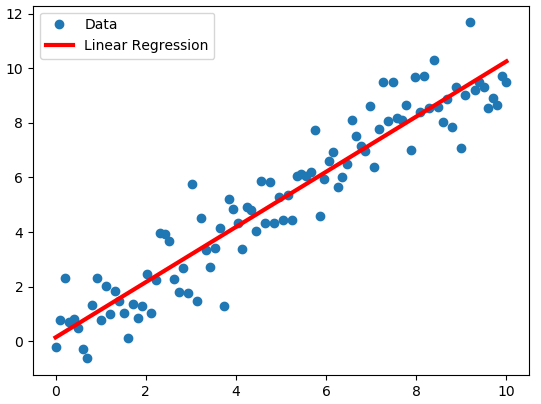
Ta có training set giá nhà và đưa nó vào thuật toán học. Công việc của thuật toán học là cho ra 1 hàm - ký hiệu là “h”. Lấy giá trị đầu vào là kích thước ngôi nhà có thể là kích thước mới. Nó lấy giá trị x và thử ra 1 giá trị ước tính y cho ngôi nhà tương ứng. Ta có thể nói hàm “h” là 1 hàm ánh xạ.



*Hình 3: Hàm h [4]*

Hàm h : hoặc 

#### Linear regression



Hình 4: Đồ thị Linear regression

Theo đồ hình 4, ta có thể dự đoán y theo một hàm h nào đó cụ thể ở đây là h của x với  cộng với . Đây là mô hình hồi quy tuyến tính. Nó hồi quy tuyến tính với một biến là x. Dự đoán tất cả giá nhà với một biến x.

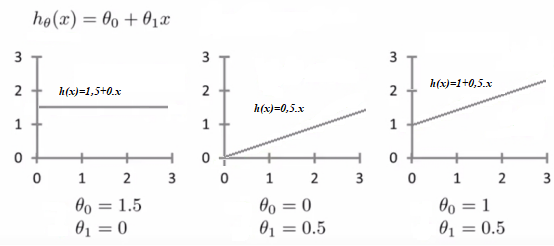
#### Cost funtion

Ta có :

: parameter

 và  không đổi: tham số của mô hình.

Cách chọn theta.



*Hình 5 : Đồ thị hàm h theo các theta khác nhau*

Ý tưởng : chọn  và  để h(x) là giá trị dự đoán đầu vào x gần nhất với các giá trị y cho những trainning set. Hay nói cách khác là làm thế nào giá trị dự đoán gần với giá trị thực tế nhất hoặc có thể nói là mục đích tìm  và  là tìm đường thẳng phù hợp với dữ liệu. Như vậy, trong trainning set kích thước ngôi nhà, ta đưa vào một số mẫu mà chúng ta biết x là kích thước căn nhà và ta biết giá trị thực sự nó đã bán là bao nhiêu. Vậy trong hồi quy tuyến tính, việc ta làm là giải bài toán tìm số nhỏ nhất.

Kí hiệu cho bài toàn số nhỏ nhất:và cái ta muốn là sự khác nhau giữa h(x) và y là nhỏ nhất rồi cực tiểu hóa bình phương sự khác nhau đó.

: sự khác biệt giữa giá dự định và thực tế

Vậy bài toán của ta là tìm các giá trị của  và  sao cho trung bình,  và giá trị thật trên tập huấn luyện là nhỏ nhất. Đây là mục tiêu của “Hồi quy tuyến tính” và hàm ở trên chính là Cost Funtion hay còn gọi là hàm J : giúp ta tìm ra cách để tìm đường tuyến tính tốt nhất cho dữ liệu.

Kí hiệu:



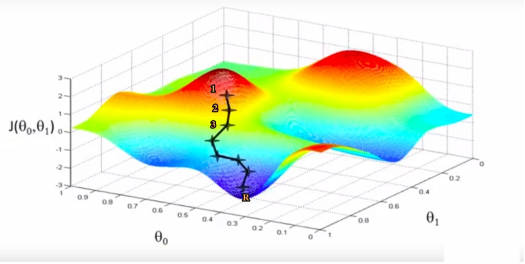
#### Gradient Descent

Gradient Descent là một thuật toán tối ưu hóa lặp đầu tiên để tìm mức tối thiểu của hàm [2]. Cụ thể ở đây là cực tiểu hóa hàm J.

Ý tưởng:

* Đầu tiên ta khởi đầu với một vài giá trị nào đó cho  và . Thông thường sẽ cho cả 2 tham số bằng 0.
* Thay đổi  và  một chút để giảm giá trị của J đến một lúc nào đó chúng có thể đạt cực tiểu.

Để dễ hiểu hơn, xem ví dụ bên dưới



*Hình 6: Ví dụ về Gradient Descent [4]*

Ở đây ta có 2 trục ,  và độ cao của bề mặt là hàm J. Mục đích là cực tiểu hỏa “tấm thảm” này. Đầu tiên bắt đầu với một vị trí (,) nào đó. Cụ thể trong hình là vị trí 1 với giá trị là (0,0). Bây giờ tưởng tượng rằng tấm thảm này là một cái hố với 2 ngọn đồi màu đỏ và ta đang đứng tại vị trí nào đó trong ngọn đồi màu đỏ, cụ thể ở đây là vị trí 1. Trong gradient descent, ta sẽ quay 360 độ xung quanh ở một hướng ngẫu nhiên. Nếu muốn đi xuống ngọn đồi nhanh nhất có thể thì hướng nào chúng ta nên bước? Dễ thấy rằng, nếu bạn đang đứng tại vị trí 1 trên ngọn đồi, thì hướng đi hiệu quả nhất là vị trí 2. Khi bước tới vị trí số 2, ta lại đang đứng ở 1 vị trí mới trên ngọn đồi và tiếp tục tìm hướng đi tiếp theo để xuống ngọn đồi nhanh nhất. Cụ thể vị trí tiếp theo là vị trí 3, và cứ tiếp tục lặp lại như vậy cho đến khi đến vị trí R là nơi chúng ta muốn xuống. Tuy nhiên, nếu bạn lặp lại quá trình trên với vị trí khác thì Gradient descent sẽ chỉ bạn đường đi tối ưu khác.

Như vậy , nếu bạn bắt đầu bằng điểm thứ nhất, bạn sẽ đạt vùng tối ưu thứ nhất. Nhưng nếu bắt đầu với vị trí khác hơn một chút thì nó sẽ đạt vùng tối ưu khác. Đây là tính chất của Gradient Desent. Đó là về mặt hình ảnh còn về mặt toán học ta có công thức Gradient Descent:

 for (j=0 và j=1)

: tỷ lệ học (learning rate). Điều khiển độ lớn của một bước mà chúng ta đi xuống đồi (ví dụ hình 6) với gradient desent. Nếu  lớn thì tương đương với quá trinh gradient descent với các bước rất lớn và ngược lại.

 : Đạo hàm

Cách triển khai:  và  cập nhật đồng thời

temp0: = 

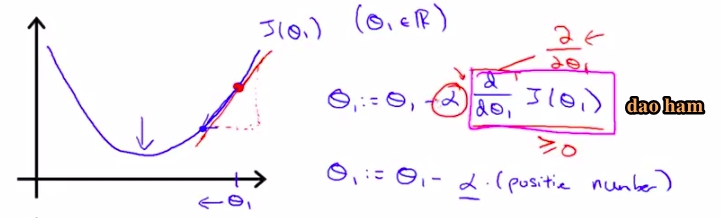
temp1:= 

:= temp0

:= temp1

##### Đạo hàm trong Gradient Desent.

Chúng ta cùng xét ví dụ sau:



*Hình 7: Ý nghĩa của đạo hảm trong Gradient Descent [4]*

Đường màu đỏ trong hình là dường đạo hàm và nó có độ dốc dương khi đạo hàm là số dương. Như vậy, cập nhật  sẽ là:  vì  luôn dương nên khi cập nhật lại , theo công thức thì sẽ bị nhỏ đi và nó có hướng về phía bên trái theo đồ thị trên nghĩa là nó hướng về vùng cực tiểu.

##### 15Learning rate trong Gradient Descent

Trước tiên ta xét khi  nhỏ như đồ thị đầu tiên, khi quá nhỏ thì các bước đi của Gradient Descent sẽ rất nhỏ, ta có thể thấy các dấu chấm rất sát nhau và dẫn đến để đi đến vùng cực tiểu sẽ mất rất nhiều thời gian.

Ngược lại khi  quá lớn. Bước đi quá lớn nên đôi khi nó có thể vượt qua vùng cực tiểu, thậm chí đối hướng và nó có thể càng lúc càng xa như trong đồ thị thứ 2 trong hình 8.

*Hình 8: ví dụ và* [4]

1. Gradient Descent cho Linear Regression

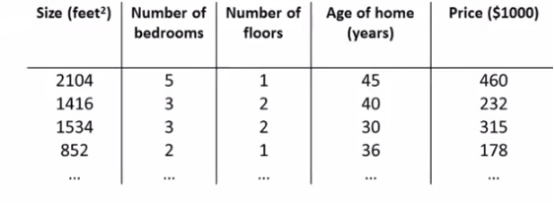
Công thức:





### Multivariate Linear Regression

Giả sử chúng ta có nhiều feature hơn như trong hình sau:



*Hình 10 : Ví dụ về Data trong Multivariate Linear Regression [8]*

Ngoài feature kích thước ngôi nhà, ta còn có số phòng ngủ, số tầng và độ tuổi ngôi nhà. Nhiều thông tin hơn để dự đoán giá trị ngôi nhà.

Các kí hiệu:

: Kích thước ngôi nhà

: Số phòng

: Số tầng

: Độ tuổi ngôi nhà

n : Số lượng feature

: feature đầu vào. Ví dụ :  là một vecto với các giá trị {1416,3,2,40} .

: Kí hiệu cho giá trị thứ j . Ví dụ = 2.

* Hàm h:



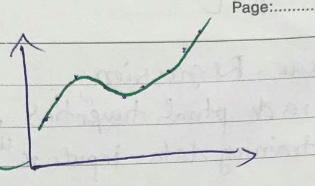
* Cost Function:



* Gradient Descent



### Polynomial Regression

Trong trường hợp này ta thấy đồ thị theo hình cong lên cong xuống chứ không phải một đường thẳng. Chính vì thế nên khi sử dụng hàm h(x) này, có thể kết quả dự đoán sẽ không tốt.

*Hình 11: Ví dụ về đường Polynomial Regression*

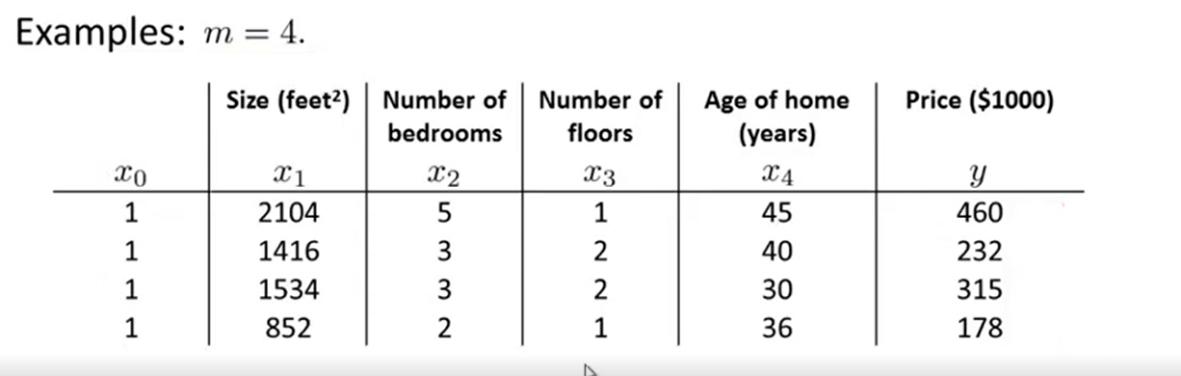
Hình dạng cong hơn thì sẽ nhiều bậc hơn. Vì vậy công thức Polynomial Regression là công thức Linear Regression được cải tiến thêm nhiều bậc hơn để phù hợp với dữ liệu. Một ví dụ về công thức của Polynomial Regression:

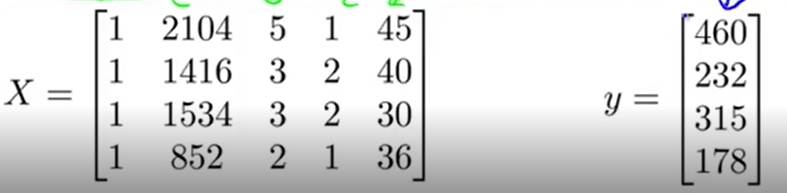


### Normal Equation (Công thức chuẩn)

Cũng giống như Gradient Descent, Normal Equation cũng là một thuật toán giúp ta tìm giá trị tốt nhất của . Đối với Gradient Descent chúng ta phải chạy thuật toán nhiều lần để nó hội tụ lại tại vùng cực tiểu. Còn Normal Equation nó sẽ phân tích . Thay vì phải chạy nhiều lần như Gradient Descent thì Normal Equation sẽ giải quyết việc tìm  tối ưu nhất chỉ trong một lần chạy.

Chúng ta cùng xem xét training set sau:





*Hình 12. Tập huyến luyện về Normal Equation [8]*

Theo hình 12 thì ma trận X là ma trận chứa tất cả các feature từ training set. Ma trận X là ma trận [m x (n+1)] với m là số mẫu huấn luyện và n là số feature, (n+1) vì ở đây có feature  và nó luôn luôn bằng 1.

Ta có công thức Normal Equation sau:

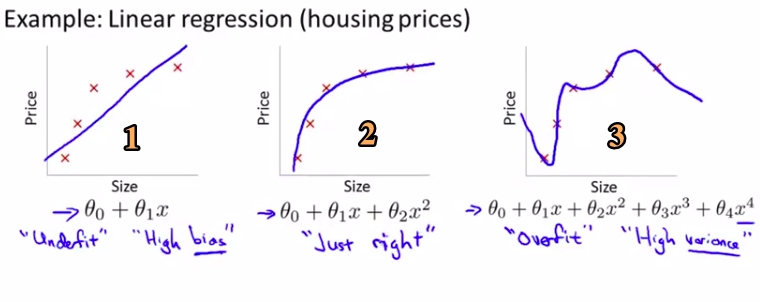


Giả sử bạn có m trainning set và n features ta có bảng so sánh sau:

|  |  |
| --- | --- |
| Gradient Descent | Normal Equation |
| - Phải chọn  - Cần nhiều vòng lặp  - Hiệu quả khi số lượng features lớn. | - Không phải chọn  - Không cần lặp  - Cần tính : .  - Tính rất chậm khí features lớn |

*Bảng 2: So sánh Gradient Descent và Normal Equation*

### Overfitting



*Hình 13. Ví dụ về Overfitring và Underfirting [8]*

Ví dụ về dự đoán giá nhà với thuật toán Linear regression.

* Biểu đồ 1:
* Mô hình dự đoán quá khác so với mô hình thực tế. Chúng ta gọi đây là “Underfitting” hay là thuật toán này có “high bias”
* “High bias” : nghĩa là nếu áp dụng một đường tuyến tính cho dữ liệu thì thuật toán sẽ rất lệch. Ý là giá nhà sẽ tăng lên tuyến tính theo kích thước của chúng mặc dù sự thật dữ liệu thì ngược lại. Dù dữ liệu ngược lại nhưng thuật toán vẫn cho ra một đường tuyến tính và điều này dẫn đến việc không phù hợp dữ liệu.
* Biểu đồ 3:
* Hypothesis trong biểu đồ này đi qua tất cả các điểm dữ liệu huấn luyện. Nói cách khác, nó khớp (fit) dữ liệu huấn luyện rất tốt. Tuy nhiên, vì có số bậc (degree) lớn, nó biến đổi quá nhiều nên không mô tả được khuynh hướng chính của dữ liệu. Vì vậy, nhiều khả năng nó sẽ dự đoán dữ liệu mới (giá nhà với các features chưa từng được huấn luyện) không tốt. Vấn đề này gọi là “Overfitting” hay thuật toán này có “high variance”
* “High variance”: Nếu bạn áp dụng một hàm đa thức bậc cao thì giả thuyết hầu như phù hợp với hầu hết các hàm tuy nhiên nó dẫn tới việc giả thuyết này quá lớn, quá bất định và chúng ta không có đủ dữ liệu để giới hạn nó để có được một giả thuyết tốt.
* Vấn đề Overfitting xảy ra khi chúng ta có quá nhiều feature. Lúc dó giả thuyết có thể phù hợp rất tốt với tập huấn luyện. Do đó hàm J có thể rất gần 0. Nhưng điều này lại dẫn đến một đường cong và nhìn rất phức tạp.
* Biểu đồ 2:
* Biểu đồ gần phù hợp với dữ liệu. Đây ta gọi là “Just right”

Vậy làm thế nào khi gặp Overffiting. Chúng ta có 2 cách:

* Giảm số lượng feature: Chúng ta sẽ xét những feature quan trọng để giữ lại và những feature không quan trọng chúng ta có thể bỏ. Tuy nhiên khi chúng ta giảm số lượng feature như vậy thì nó cũng có thể bỏ đi một vài thông tin của bài toán.
* Regularization : Giữ lại tất cả các feature nhưng giảm độ lớn hoặc giảm giá trị .

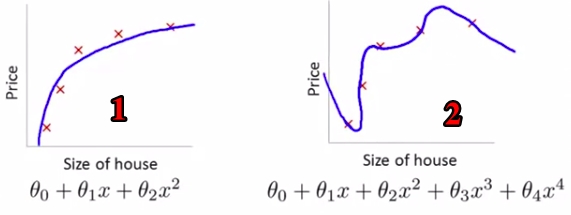
## RIDGE REGRESSION

### Giới thiệu

Hồi quy Ridge (Ridge Regression) bắt đầu từ một phương pháp có tên [Tikhonov Regularization](https://en.wikipedia.org/wiki/Tikhonov_regularization) - đặt theo tên của nhà toán học người Nga [Andrey Nikolayevich Tikhonov](https://en.wikipedia.org/wiki/Andrey_Nikolayevich_Tikhonov) cho các ứng dụng của phương trình vi phân. Nhưng người sử dụng cách tiếp cận và diễn giải nó theo ngôn ngữ của thống kê là Arthur E. Hoerl và từ đó phương pháp này được gọi là hồi quy Ridge. Hồi quy ridge là biến thể của hồi quy tuyến tính nhầm khác phục tình trạng overfitting. [5]

### Regularization

Regularization, một cách cơ bản, là thay đổi mô hình một chút để tránh overfitting trong khi vẫn giữ được tính tổng quát của nó (tính tổng quát là tính mô tả được nhiều dữ liệu, trong cả tập training và test). Một cách cụ thể hơn, ta sẽ tìm cách di chuyển nghiệm của bài toán tối ưu hàm mất mát tới một điểm gần nó. Hướng di chuyển sẽ là hướng làm cho mô hình ít phức tạp hơn mặc dù giá trị của hàm mất mát có tăng lên một chút.



*Hình 14: Ví dụ về regularization*

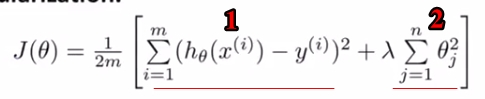
Cùng xét ví dụ sau , giả sử chúng ta muốn cực tiểu hàm này:



Cách duy nhất để cực tiểu hàm J này thì theta3 và theta4 phải rất nhỏ.

Vì vậy khi cực tiểu hóa hàm J này thì chúng ta sẽ tìm cách để theta3 và theta4 gần với 0. Và khi theta3 và theta4 gần đến 0 thì biểu thức ở đồ thị 2 sẽ không có theta3 và theta4. Nghĩa là nó sẽ trở lại hàm ở biểu đồ 1. Chúng ta sẽ có giả thuyết tốt hơn.

Vậy khi hàm J có hàng trăm hay hàng ngàn theta , vậy làm sao chúng có thể cực tiểu hóa hàm J được? Chúng ta sẽ cộng vào hàm J một cụm regularization ở phía sau để giảm từng tham số. Cụ thể là hàm J của Linear Regression



*Hình 15. Giải thích về tham số *

: tham số regularization. thường là một số dương để cân bằng giữa hai đại lượng ở vế phải.

Cụm 1 là cụm mục tiêu đó là những gì chúng ta muốn áp dụng tập huấn luyện. Và cụm 2 là chúng ta muốn giữa các tham số nhỏ để giả thuyết đơn giản hơn. .Và việc của lamda là kiểm soát sự khác biệt giữa 2 cụm này.

Việc lựa chọn tham số vô cùng quan trọng, được lựa chọn bằng phương pháp **cross-validation**. Vậy có thể hiểu Ridge regression là Linear Regression có sử dụng thêm Regularization.

### Validation

1. **Validation**

Việc chia tập dữ liệu ra thành hai tập nhỏ: training data và test data và ta không được sử dụng test data. Vậy làm cách nào để biết được chất lượng của mô hình với unseen data (dữ liệu không xác định)?

Đơn giản nhất là lấy từ tập training data ra một tập con nhỏ và thực hiện việc đánh giá mô hình trên tập con nhỏ này và tập này được gọi là validation set. Lúc này, **training set là phần còn lại của training set ban đầu**. Train error được tính trên training set mới này.

**Train error:** Hàm mất mát áp dụng lên training data. Cần có một thừa số để tính giá trị trung bình, tức mất mát trung bình trên mỗi điểm dữ liệu. [3]

**Test error:** Tương tự như trên nhưng áp dụng mô hình tìm được vào **test data**. Chú ý rằng, khi xây dựng mô hình, ta không được sử dụng thông tin trong tập dữ liệu test. Dữ liệu test chỉ được dùng để đánh giá mô hình [3]

1. **Cross-validation.**

Nhiều trường hợp, chúng ta cần hạn chế số lượng dữ liệu để xây dựng mô hình. Nếu lấy quá nhiều dữ liệu trong tập training ra làm dữ liệu validation, phần dữ liệu còn lại của tập training là không đủ để xây dựng mô hình. Lúc này, tập validation phải thật nhỏ để giữ được lượng dữ liệu cho training đủ lớn. Tuy nhiên, một vấn đề khác nảy sinh. Khi tập validation quá nhỏ, hiện tượng overfitting lại có thể xảy ra với tập training còn lại. [3]

Cross validation là một cải tiến của validation với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác nhau. Một cách thường đường sử dụng là chia tập training ra kk tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử , được gọi là run, một trong số kk tập con được lấy ra làm validata set. Mô hình sẽ được xây dựng dựa vào hợp của k−1k−1 tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi là **k-fold cross validation**. [3]

Khi k bằng với số lượng phần tử trong tập training ban đầu, tức mỗi tập con có đúng 1 phần tử, ta gọi kỹ thuật này là **leave-one-out**. [3]

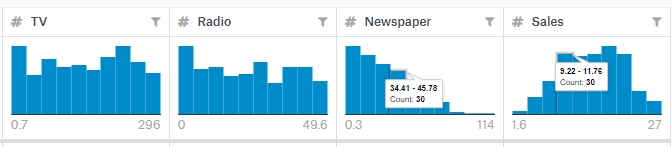
## LẬP TRÌNH

### Dataset

Bộ dữ liệu quảng cáo được cung cấp trong ISLR và phân tích mối quan hệ giữa 'quảng cáo truyền hình' và 'doanh số'.

Gồm 4 cột : TV, radio, newpapers va sales. “sales” là cột chúng ta cần dự đoán.

Dữ liệu được lấy từ [6]



Hình 16: Biểu đồ về dữ liệu của Dataset [6]

### Scikit-learn

cikit-learn (viết tắt là sklearn) là một thư viện mã nguồn mở dành cho học máy - một ngành trong trí tuệ nhân tạo, rất mạnh mẽ và thông dụng với cộng đồng Python, được thiết kế trên nền NumPy và SciPy. Scikit-learn chứa hầu hết các thuật toán machine learning hiện đại nhất, đi kèm với documentations, luôn được cập nhật.

Tại sao nên dùng scikit-learn?

- Hỗ trợ hầu hết các thuật toán của machine learning một cách đơn giản, hiệu quả mà chúng ta không cần phải mất công ngồi cài đặt lại.

- Có tài liệu hướng dẫn sử dụng

- Độ tin cậy cao do scikit-learn được xây dựng bởi các chuyên gia hàng đầu

- Có nguồn dữ liệu phong phú: iris, digit, …

Sử dụng scikit-learn như thế nào?

Trước tiên máy bạn phải được cài đặt Python rồi. Sau đó bạn có thể vào trực tiếp trang chủ của scikit-learn để xem cách cài đặt nó. Đơn giản nhất là sử dụng Anaconda

Sklearn được xây dựng trên NumPy và SciPy nên để sử dụng sklearn, chúng ta bắt buộc phải có 2 packages này. Tất nhiên ta nên cài đặt matplotlib, một package không thể thiếu trong scientific plotting. Cuối cùng, chúng ta cần cài thêm package pandas, phục vụ cho data wrangling và analysis.

Nếu đã có Python thì có thể cài đặt các thư viện trên sử dụng pip:

*pip install numpy scipy matplotlib scikit-learn pandas*

### Cài đặt

Thư viện được sử dụng:

Sklearn, matplotlib, numpy, pandas.

- Pandas: Pandas là 1 open source, được cộng đồng đánh giá là high-performance, việc xử lý dữ liệu, tính toán sẽ dễ dàng hơn rất nhiều cách truyền thống.

- Numpy: Numpy cung cấp các đối tượng và phương thức để làm việc với mảng nhiều chiều và các phép toán đại số tuyến tính. Trong numpy, chiều của mảng gọi là axes; trong khi số chiều gọi là rank.

- matplotlib : Vẽ đồ thị

- Sklearn : sử dụng thuật toán Machine learning

- Mean squared error - MSE

MSE của một phép ước lượng là trung bình của bình phương của sai số, tức là sự khác biệt giữa các giá trị được mô hình dự đoán và gía trị thực.

- GridSearchCV: giả dụ giá trị của 2 parameter lần lượt từ 0-9. Grid Search sẽ lần lượt ghép từng giá trị của param 1 với param 2 để tính toán độ chính xác của model. Đảm bảo không bỏ sót cặp parameter nào.

* Code đầy đủ:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import GridSearchCV

from sklearn.linear\_model import Ridge

data = pd.read\_csv("Advertising1.csv")

print(data.head())

data.drop(['Unnamed: 0'], axis=1, inplace=True)

print(data.head())

def scatter\_plot(feature, target):

plt.figure(figsize=(16, 8))

plt.scatter(

data[feature],

data[target],

c='black'

)

plt.xlabel("Money spent on {} ads ".format(feature))

plt.ylabel("Sale ")

plt.show()

scatter\_plot('TV', 'sales')

scatter\_plot('radio', 'sales')

scatter\_plot('newspaper','sales')

Xs = data.drop(['sales'], axis=1)

y = data['sales'].values.reshape(-1,1)

lin\_reg = LinearRegression()

MSEs = cross\_val\_score(lin\_reg, Xs, y, scoring='neg\_mean\_squared\_error', cv=5)

mean\_MSEs = np.mean(MSEs)

print(mean\_MSEs)

ridge = Ridge()

parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10,20]}

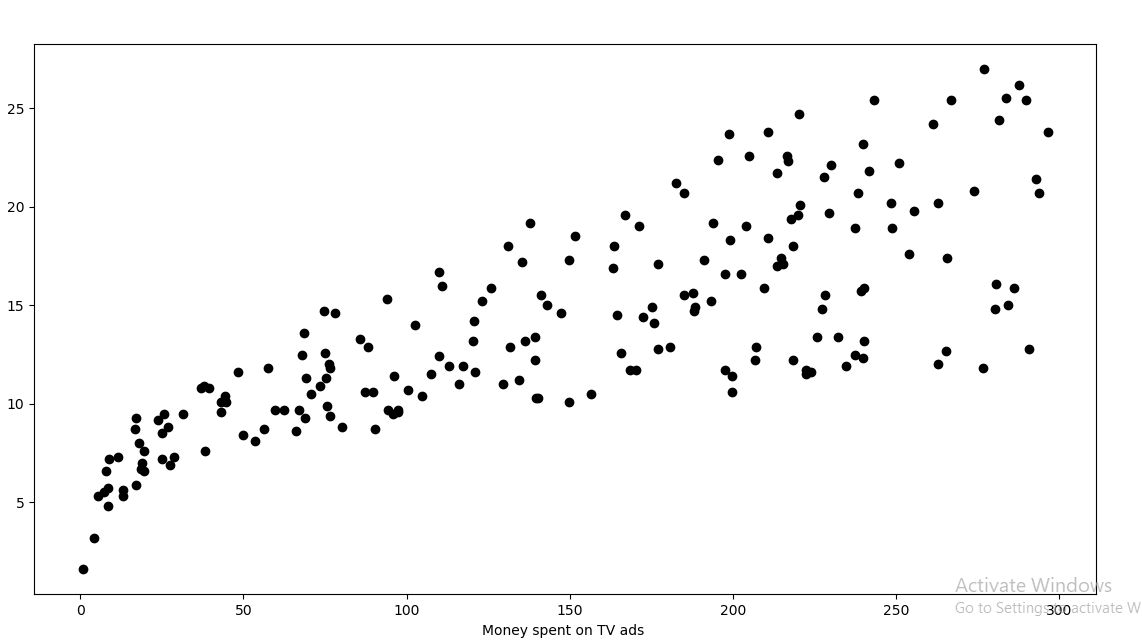
ridge\_regressor = GridSearchCV(ridge, parameters, scoring='neg\_mean\_squared\_error', cv=5)

ridge\_regressor.fit(Xs, y)

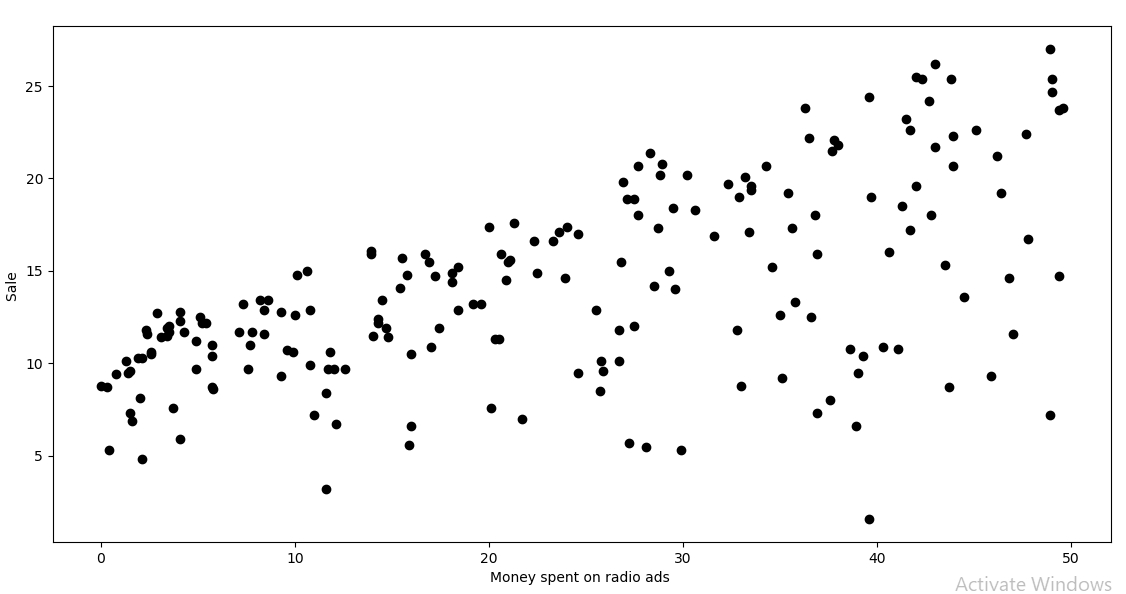
print(ridge\_regressor.best\_params\_)

print(ridge\_regressor.best\_score\_)

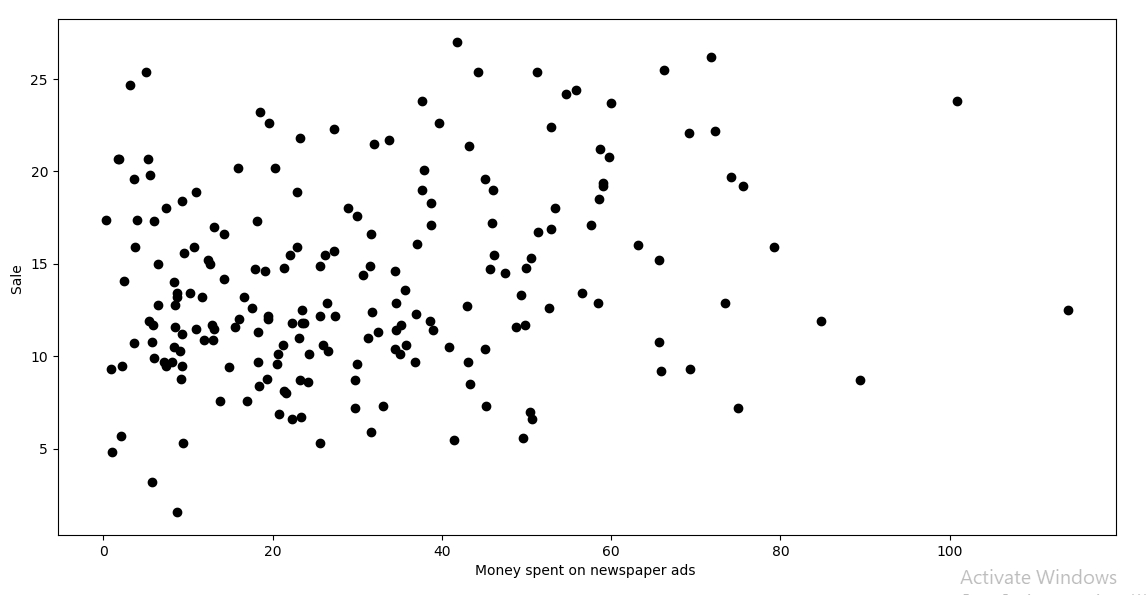
* Kết quả:



*Hình17 : Sales liên quan đến tiền chi cho quảng cáo truyền hình*



*Hình 18: Sales liên quan đến tiền chi cho quảng cáo radios*



*Hình 19 : Sales liên quan đến tiền chi cho quảng cáo trên báo*

Kết quả của MSEs khi sử dụng mô hình Linear Regression : -3.0729465971002092

Kết quả của MSEs khi sử dụng mô hình Ridge Regression : -3.0726713383411424 với alpha là 20. Bạn có thấy nó tốt hơn 1 chút so với Linear Regression.

**BẢNG PHÂN CÔNG**

|  |  |  |
| --- | --- | --- |
| Họ tên và MSSV | Công việc | Đánh giá |
| Nguyễn Ngọc Quý - 16110195 | Làm báo cáo, code, Linear Regression, Ridge Regression | 50% |
| Trần Bảo Đan - 16110044 | Tim hiểu đạo văn, Làm báo cáo , Linear Regression | 50% |

**TÀI LIỆU THAM KHẢO**

[1] Học có giám sát – Wikipedia tiếng Việt. (2005, December 21). Retrieved May 16, 2019, from https://vi.wikipedia.org/wiki/H%E1%BB%8Dc\_c%C3%B3\_gi%C3%A1m\_s%C3%A1t

[2] Gradient descent – Wikipedia. (2018 September 20). Retrieved May 16, 2019, from https://en.wikipedia.org/wiki/Gradient\_descent

[3] Overfitting – Machine Learning Cơ Bản (2017 March 4). Retrieved May 16, 2019, from https://machinelearningcoban.com/2017/03/04/overfitting/

1. Machine Learning - Artificial Intelligence - All in One (2016 November 07). Retrieved May 16, 2019 from <https://www.youtube.com/channel/UC5zx8Owijmv-bbhAK6Z9apg>
2. Ridge Regression - Machine Learning (Section 1): Linear Regression and Its Cousins - Ridge Regression . Retrived May 17, 2019 from <https://rstudio-pubs-static.s3.amazonaws.com/305616_794b978e70d14fd8bfd423df3cd3860e.html>

[6] Ridge Regression with python - How to Perform Lasso and Ridge Regression in Python (2019 january 13). Retrieved May 16,2918 from

<https://towardsdatascience.com/how-to-perform-lasso-and-ridge-regression-in-python-3b3b75541ad8>