	VIETTEL AI RACE	TD035
	HỒI QUY TUYẾN TÍNH	Lần ban hành: 1

Thuật toán linear regression giải quyết các bài toán có đầu ra là giá trị thực, ví dụ: dự đoán giá nhà, dự đoán giá cổ phiếu, dự đoán tuổi,...

1. Bài toán

Bạn làm ở công ty bất động sản, bạn có dữ liệu về diện tích và giá nhà, giờ có một ngôi nhà mới bạn muốn ước tính xem giá ngôi nhà đó khoảng bao nhiêu. Trên thực tế thì giá nhà phụ thuộc rất nhiều yếu tố: diện tích, số phòng, gần trung tâm thương mại,.. nhưng để cho bài toán đơn giản giả sử giá nhà chỉ phụ thuộc vào diện tích căn nhà. Bạn có dữ liệu về diện tích và giá bán của 30 căn nhà như sau:

Diện tích(m2)	Giá bán (triệu VNĐ)
30	448.524
32.4138	509.248
34.8276	535.104
37.2414	551.432
39.6552	623.418
...	...

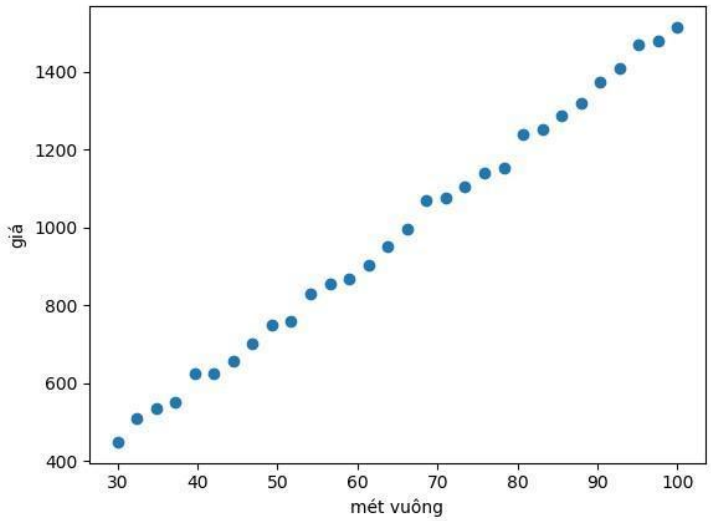
Khi có dữ liệu mình sẽ visualize dữ liệu lên hình 3.1

Nếu giờ yêu cầu bạn ước lượng nhà 50 mét vuông khoảng bao nhiêu tiền thì bạn sẽ làm thế nào? Vẽ một đường thẳng gần với các điểm trên nhất và tính giá nhà ở điểm 50 như ở hình 3.2

Về mặt lập trình cũng cần làm 2 việc như vậy:

. Training: Tìm đường thẳng (model) gần các điểm trên nhất. Mọi người có thể vẽ ngay được đường thẳng mô tả dữ liệu từ hình 1, nhưng máy tính thì không, nó phải đi tìm bằng thuật toán Gradient descent ở phía dưới. (Từ model và đường thẳng được dùng thay thế lẫn nhau trong phần còn lại của bài này).

. Prediction: Dự đoán xem giá của ngôi nhà 50 m^2 có giá bao nhiêu dựa trên đường tìm được ở phần trên.



Hình 3.1: Đồ thị quan hệ giữa diện tích và giá nhà.

2. Thiết lập công thức

2.1 Model

Phương trình đường thẳng có dạng $y = ax+b$ ví dụ hình 3.3. Thay vì dùng kí hiệu a, b cho phương trình đường thẳng; để tiện cho biểu diễn ma trận phần sau ta sẽ thay $w_1 = a, w_0 = b$

Nên phương trình được viết lại thành: $y = w_1 * x + w_0$ => Việc tìm đường thẳng giờ thành việc tìm w_0, w_1.

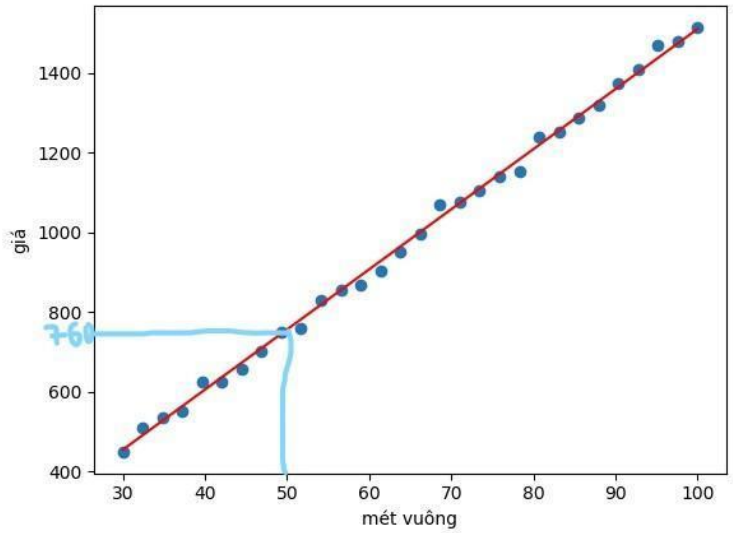
Để tiện cho việc thiết lập công thức, ta sẽ đặt ký hiệu cho dữ liệu ở bảng dữ liệu: (x_1, y_1) = (30, 448.524), (x_2, y_2) = (32.4138, 509.248),..

Tức là nhà diện tích x_i thực sự có giá y_i . Còn giá trị mà model hiện tại đang dự đoán kí hiệu là $\hat{y}_i = w_1 * x_i + w_0$

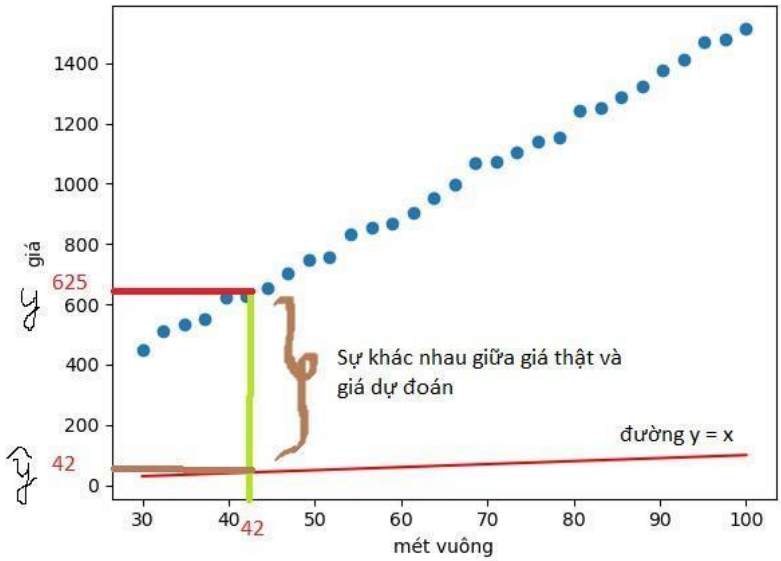
2.2 Loss function

Việc tìm w_0, w_1 có thể đơn giản nếu làm bằng mắt nhưng máy tính không biết điều đấy, nên ban đầu giá trị được chọn ngẫu nhiên ví dụ $w_0 = 0, w_1 = 1$ sau đấy được chỉnh dần.

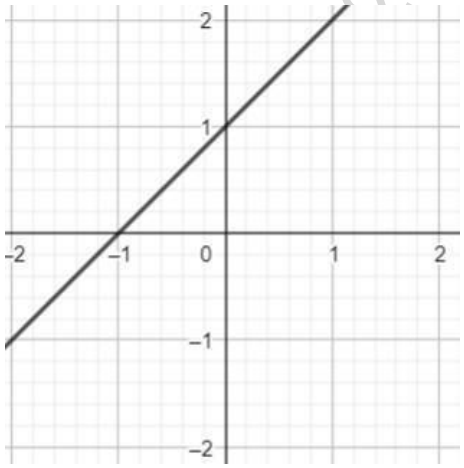
Rõ ràng có thể thấy đường y = x như ở hình 3.4 không hề gần các điểm hay không phải là đường mà ta cần tìm. Ví dụ tại điểm x = 42 (nhà 42 m^2) giá thật là 625 triệu nhưng giá mà model dự đoán chỉ là 42 triệu.



Hình 3.2: Ước tính giá căn nhà 50 m^2



Hình 3.4: Sự khác nhau tại điểm x = 42 của model đường thẳng y = x và giá trị thực tế ở bảng 1



Hình 3.3: Ví dụ đường thẳng y = x + 1 (a = 1 và b = 1)

Nên giờ cần 1 hàm để đánh giá là đường thẳng với bộ tham số $(w_0,w_1) = (0,1)$ có tốt hay không. Với mỗi điểm dữ liệu (x_i,y_i) độ chênh lệch giữa giá thật và giá dự đoán được tính bằng: $\frac{1}{2}(\hat{y}_i - y_i)^2$.

Và độ chênh lệch trên toàn bộ dữ liệu tính bằng tổng chênh lệch của từng điểm:

$J = \frac{1}{2} \cdot \frac{1}{N} \cdot \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$ (N là số điểm dữ liệu). Nhận xét:

- J không âm
- J càng nhỏ thì đường thẳng càng gần điểm dữ liệu. Nếu J = 0 thì đường thẳng đi qua tất các điểm dữ liệu.

J được gọi là loss function, hàm để đánh giá xem bộ tham số hiện tại có tốt với dữ liệu không.

=> Bài toán tìm đường thẳng gần các điểm dữ liệu nhất trở thành tìm w_0,w_1 sao cho hàm J đạt giá trị nhỏ nhất.

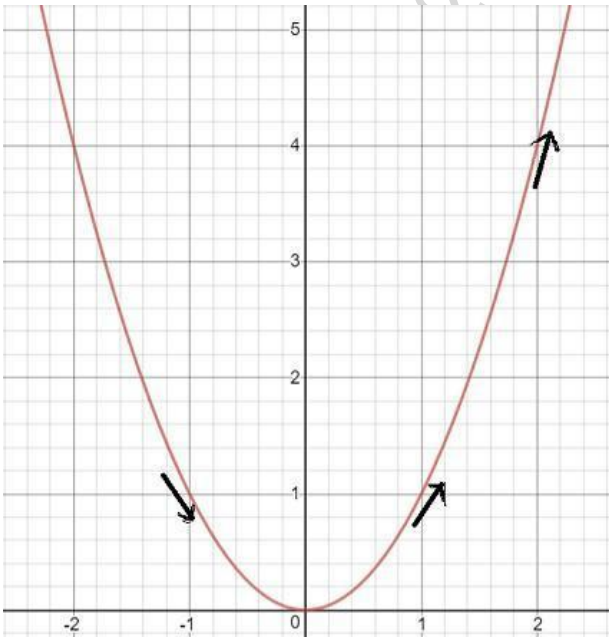
Tóm tắt: Cần tìm đường thẳng (model) fit nhất với dữ liệu, tương ứng với việc tìm tham số w_0,w_1 để cực tiểu hóa hàm J.

Giờ cần một thuật toán để tìm giá trị nhỏ nhất của hàm $J(w_0, w_1)$. Đó chính là thuật toán gradient descent.

3. Thuật toán gradient descent

3.1 Đạo hàm là gì?

Có nhiều người có thể tính được đạo hàm của hàm $f(x) = x^2$ hay $f(x) = sin(cos(x))$ nhưng vẫn không biết thực sự đạo hàm là gì. Theo tiếng hán đạo là con đường, hàm là hàm số nên đạo hàm chỉ sự biến đổi của hàm số hay có tên thân thương hơn là độ dốc của đồ thị.



Hình 3.5: Đồ thị $y = x^2$

Như mọi người đã học đạo hàm $f(x) = x^2$ là $f'(x) = 2*x$ (hoàn toàn có thể chứng minh từ định nghĩa nhưng cấp 3 mọi người đã học quá nhiều về công thức nên tôi không đề cập lại). Nhận xét:

- f'(1) = 2 * 1 < f'(2) = 2 * 2 nên mọi người có thể thấy trên hình là đồ thị tại điểm x = 2 dốc hơn đồ thị tại điểm x = 1, tuy nhiên f'(-2) = -4 < f'(-1) = -2 nhưng đồ thị tại x = -2 dốc hơn đồ thị tại x = -1 => trị tuyệt đối của đạo hàm tại một điểm càng lớn thì đồ thị tại điểm đấy càng dốc.

- f'(-1) = 2 * (-1) = -2 < 0 => đồ thị đang giảm hay khi tăng x thì y sẽ giảm; ngược lại đạo hàm tại điểm nào đó mà dương thì đồ thị tại điểm đấy đang tăng.

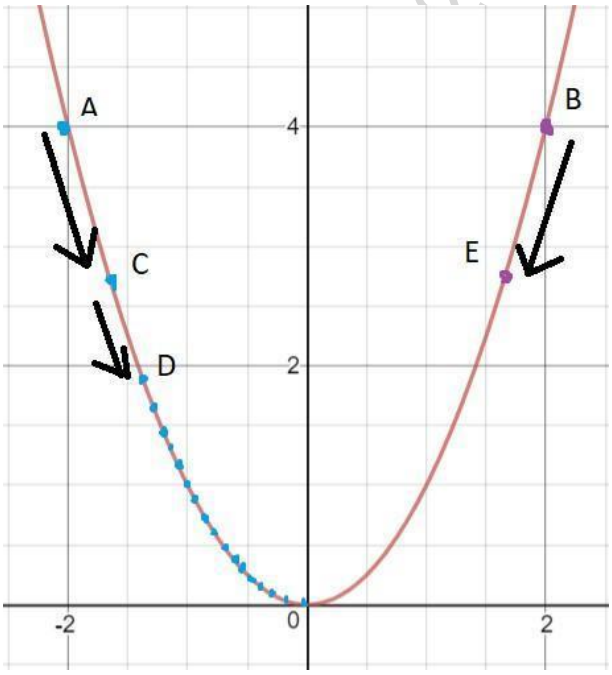
3.2 Gradient descent

Gradient descent là thuật toán tìm giá trị nhỏ nhất của hàm số f(x) dựa trên đạo hàm. Thuật toán:

1. Khởi tạo giá trị $x = x_0$ tùy ý

2. Gán x = x - learning_rate * f'(x) (learning_rate là hằng số dương ví dụ learning_rate = 0.001) 3. Tính lại f(x): Nếu f(x) đủ nhỏ thì dừng lại, ngược lại tiếp tục bước 2

Thuật toán sẽ lặp lại bước 2 một số lần đủ lớn (100 hoặc 1000 lần tùy vào bài toán và hệ số learning_rate) cho đến khi f(x) đạt giá trị đủ nhỏ. Ví dụ cần tìm giá trị nhỏ nhất hàm $y = x^2$, hàm này ai cũng biết là giá giá trị nhỏ nhất là 0 tại x = 0 nhưng để cho mọi người dễ hình dung hơn về thuật toán Gradient descent nên tôi lấy ví dụ đơn giản.



Hình 3.6: Ví dụ về thuật toán gradient descent

1. Bước 1: Khởi tạo giá trị ngẫu nhiên x = -2 (điểm A).

2. Bước 2: Do ở A đồ thị giảm nên f'(x=-2) = 2*(-2) = -4 < 0 => Khi gán x = x - learning_rate * f'(x) nên x tăng nên đồ thị bước tiếp theo ở điểm C. Tiếp tục thực hiện bước 2, gán x = x learning_rate * f'(x) thì đồ thị ở điểm D,... => hàm số giảm dần dần tiến tới giá trị nhỏ nhất.

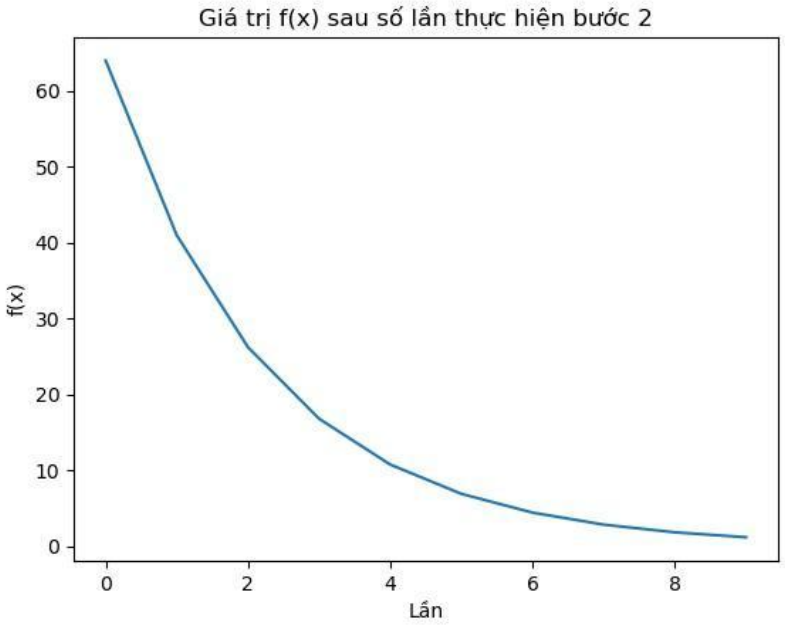
Mọi người có để ý là trị tuyệt đối của đạo hàm tại A lớn hơn tại C và tại C lớn hơn tại D không? Đến khi đến gần điểm đạt giá trị nhỏ nhất x = 0, thì đạo hàm xấp xỉ 0 đến khi hàm đạt giá trị nhỏ nhất tại x = 0, thì đạo hàm bằng 0, nên tại điểm gần giá trị nhỏ nhất thì bước 2 gán x = x - learning_rate * f'(x) là không đáng kể và gần như là giữ nguyên giá trị của x.

Tương tự nếu giá trị khởi tạo tại x = 2 (tại B) thì đạo hàm tại B dương nên do x = x - learning_rate * f'(x) giảm -> đồ thị ở điểm E -> rồi tiếp tục gán x=x -learning_rate * f'(x) thì hàm f(x) cũng sẽ giảm dần dần đến giá trị nhỏ nhất.

Ví dụ: chọn x = 10, learning_rate = 0.1, bước 2 sẽ cập nhất x = x - learning_rate*f'(x) = x learning_rate *2*x, giá trị f(x) = x^2 sẽ thay đổi qua các lần thực hiện bước 2 như sau:

Lần	x	f(x)
1	8.00	64.00
2	6.40	40.96

3	5.12	26.21
4	4.10	16.78
5	3.28	10.74
6	2.62	6.87
7	2.10	4.40
8	1.68	2.81
9	1.34	1.80
10	1.07	1.15



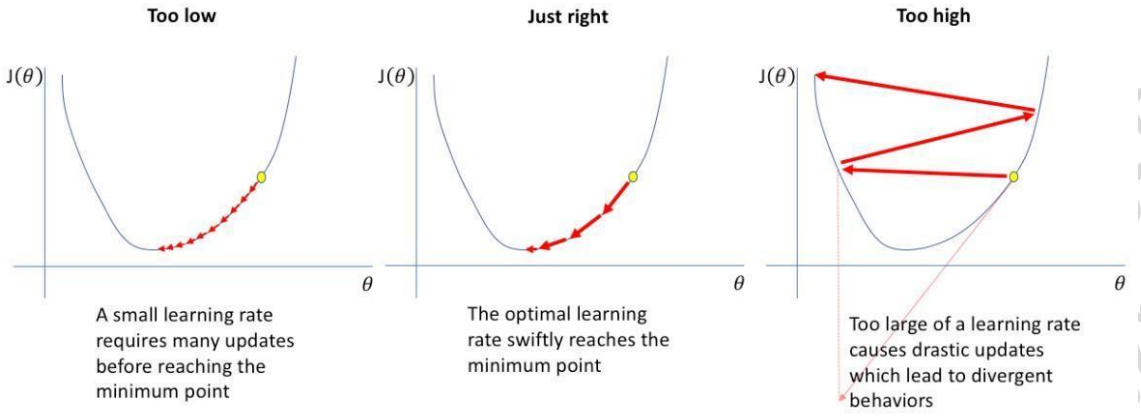
Hình 3.7: Ví dụ về thuật toán gradient descent

Nhận xét:

- Thuật toán hoạt động rất tốt trong trường hợp không thể tìm giá trị nhỏ nhất bằng đại số tuyến tính.
- Việc quan trọng nhất của thuật toán là tính đạo hàm của hàm số theo từng biến sau đó lặp lại bước 2.

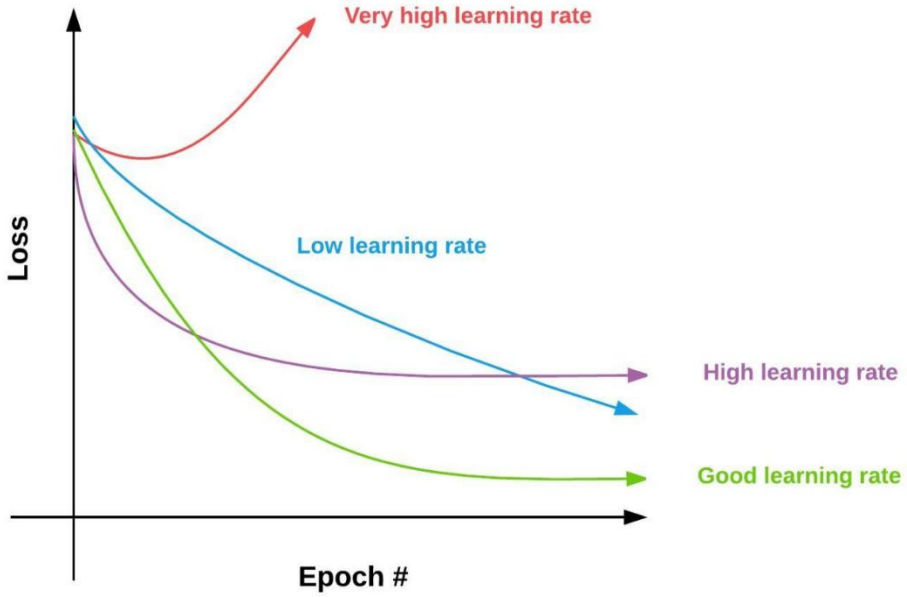
Việc chọn hệ số learning_rate cực kì quan trọng, có 3 trường hợp:

- Nếu learning_rate nhỏ: mỗi lần hàm số giảm rất ít nên cần rất nhiều lần thực hiện bước 2 để hàm số đạt giá trị nhỏ nhất.
- Nếu learning_rate hợp lý: sau một số lần lặp bước 2 vừa phải thì hàm sẽ đạt giá trị đủ nhỏ.
- Nếu learning_rate quá lớn: sẽ gây hiện tượng overshoot (như trong hình 3.8) và không bao giờ đạt được giá trị nhỏ nhất của hàm.



Hình 3.8: Các giá trị learning_rate khác nhau [13]

Cách tốt nhất để kiểm tra learning_rate hợp lý hay không là kiểm tra giá trị hàm f(x) sau mỗi lần thực hiện bước 2 bằng cách vẽ đồ thị với trục x là số lần thực hiện bước 2, trục y là giá trị loss function tương ứng ở bước đấy.



Hình 3.9: Loss là giá trị của hàm cần tìm giá trị nhỏ nhất, Epoch ở đây là số cần thực hiện bước 2 [13]

4. So sánh các loss function

4.1 Mean Absolute Error, L1 Loss

Mean Absolute Error (MAE) hay còn được gọi là L1 Loss là một loss function được sử dụng cho các mô hình hồi quy, đặc biệt cho các mô hình hồi quy tuyến tính. MAE được tính bằng tổng các trị tuyệt đối của hiệu giữa giá trị thực (y_i: target) và giá trị mà mô hình của chúng ra dự đoán ($\hat{y}_i$: predicted).

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i|$$

4.2 Mean Square Error, L2 Loss

Mean Square Error (MSE) hay còn được gọi là L2 Loss là một loss function cũng được sử dụng cho các mô hình hồi quy, đặc biệt là các mô hình hồi quy tuyến tính. MSE được tính bằng tổng các bình phương của hiệu giữa giá trị thực (y_i: target) và giá trị mà mô hình của chúng ra dự đoán ($\hat{y}_i$: predicted).

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$