Bài toán hồi quy

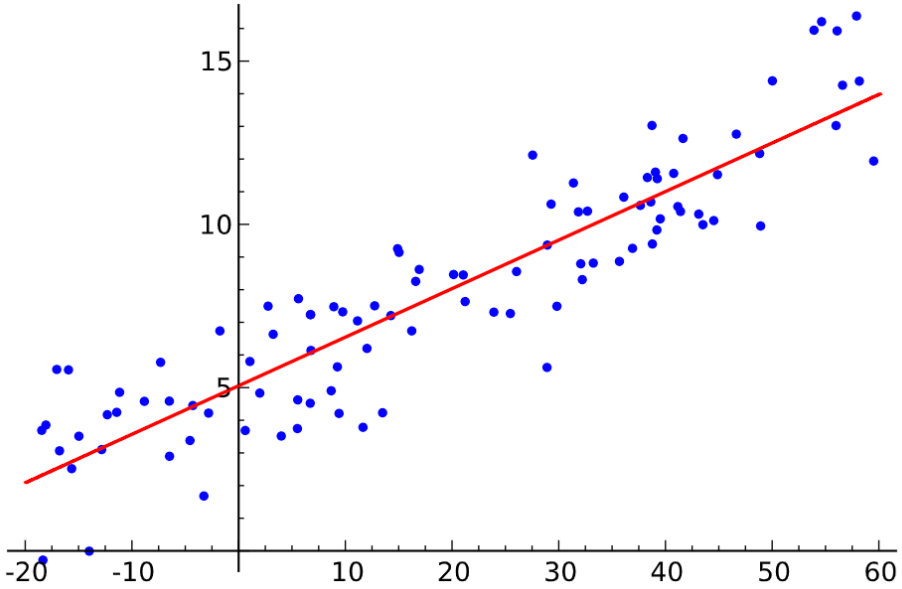
Biên soạn: **ThS. Vũ Đình Ái***(aivd@huflit.edu.vn)*

Nội dung

- Giới thiệu bài toán hồi quy
- Bài toán hồi quy 1 biến
- Thuật toán Gradien descent
- Chuẩn hoá dữ liệu
- Bài toán hồi quy đa biến

Giới thiệu bài toán hồi quy

- Linear Regression
  - là một trong những thuật toán cơ bản nhất của Machine Learning
  - là một thuật toán Supervised (học có giám sát) trong đó đầu vào và đầu ra được mô tả bởi một hàm tuyến tính.
- Bài toán Regression (hồi quy): đầu ra dự đoán có giá trị liên tục và độ dốc không đổi. Dạng toán này được sử dụng trong các bài toán có giá trị liên tục như: giá nhà, doanh số kinh doanh ...

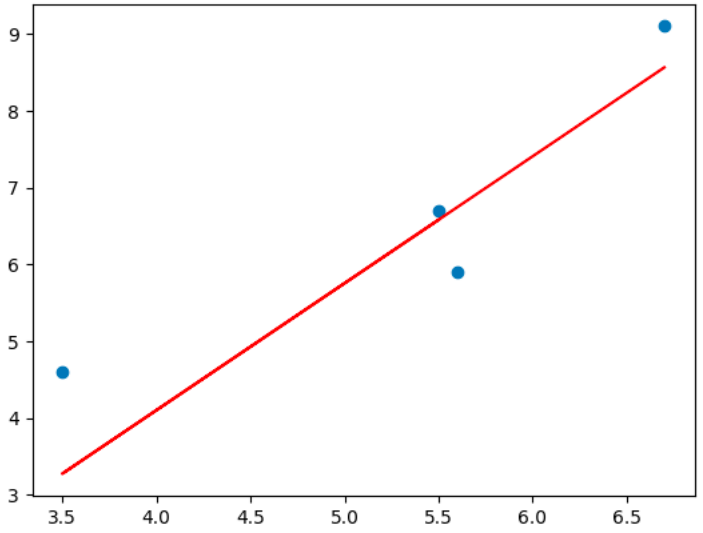


Bài toán hồi quy 1 biến

Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

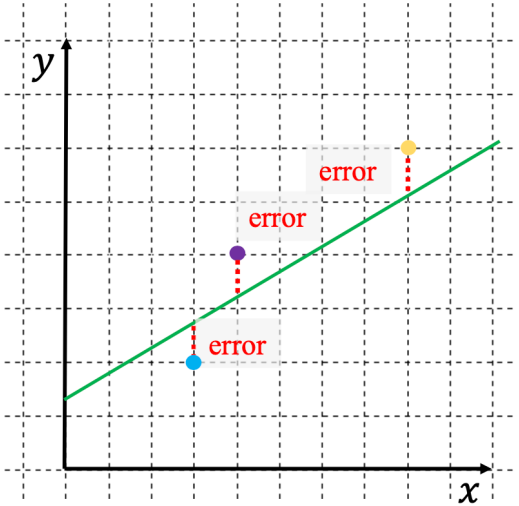
House price data

if area=6.0, price=?

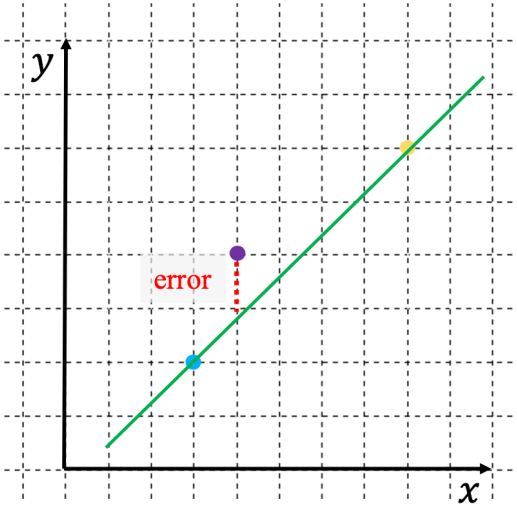


$$y = a * x + b$$

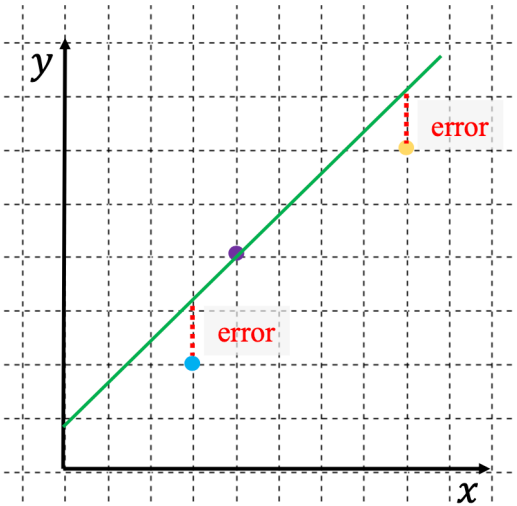
Bài toán hồi quy 1 biến



$y = a_1x + b_1$



$y = a_2x + b_2$



$y = a_3x + b_3$

Find w and b that have the smallest error. How?

Bài toán hồi quy 1 biến

- Hồi quy tuyến tính đơn biến (Linear Regression with One Variable) được sử dụng khi muốn dự đoán giá trị đầu ra `y` từ 1 giá trị đầu vào `x`. Khi đó hàm giả thuyết của có dạng:

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$$

  - Lưu ý:
    - ✓ Đây giống như phương trình đường thẳng, với 2 tham số chúng ta cần tìm là θ_0, θ_1 để dự đoán giá trị $\hat{y}$.
    - ✓ Chúng ta mong muốn giá trị $\hat{y} \approx y$. Với y là giá trị thực của đầu ra.

Bài toán hồi quy 1 biến

- Ví dụ:

Input X	Output Y
0	4
1	7
2	7
3	8

- Với giá trị ngẫu nhiên θ_0 = 2 và θ_1 = 2. Khi đó hàm giả thuyết trở thành: $h_\theta(x)$ = 2 +2x.
- Khi đó: input x = 0 ta có: $\hat{y}(0)$ = 2 + 2*0 = 2 khác xa rất nhiều so với y(0) = 4.
- ***Điều mong muốn có thể tìm được θ_0, θ_1 sao cho sự khác nhau giữa $\hat{y}$ và y là rất nhỏ.***

Bài toán hồi quy 1 biến

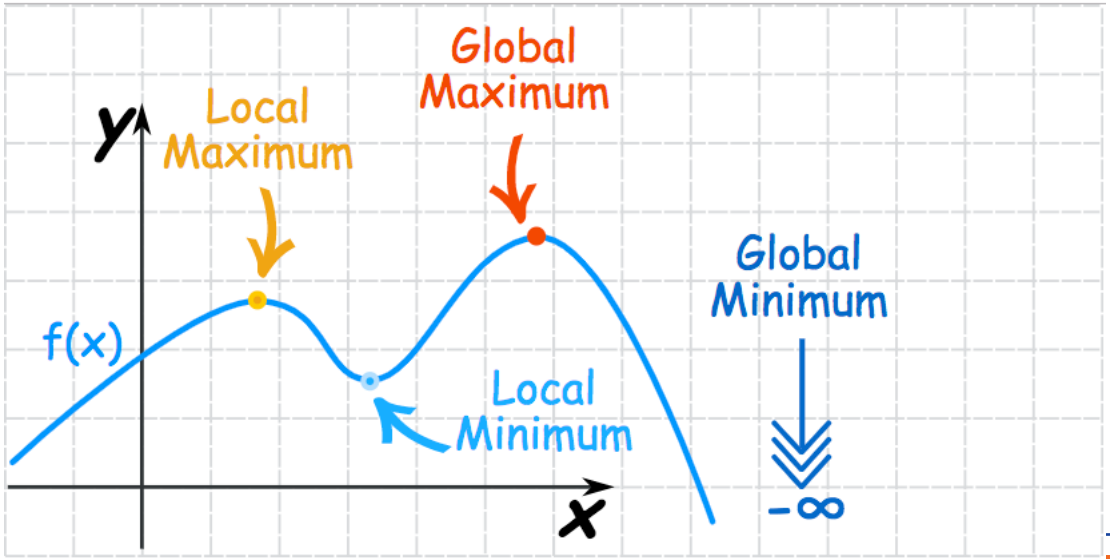
- Hàm mất mát hay còn được gọi là Cost Function hoặc Loss Function dùng để đánh giá mức độ sai lệch giữa giá trị dự đoán $\hat{y}$ và giá trị thực y .
- Mong muốn khoảng cách giữa 2 giá trị này càng nhỏ càng tốt. Để đánh giá khoảng cách này, chúng ta sử dụng sai số bình phương trung bình:

$$\mathcal{J}(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x) - y)^2$$

- Chú thích:
  - m là số quan sát (số dữ liệu)
  - $\frac{1}{m}\sum_{i=1}^{m}$ là trung bình
  - Hệ số $\frac{1}{2}$ giúp triệt tiêu 2 khi thực hiện đạo hàm $\mathcal{J}$ theo θ

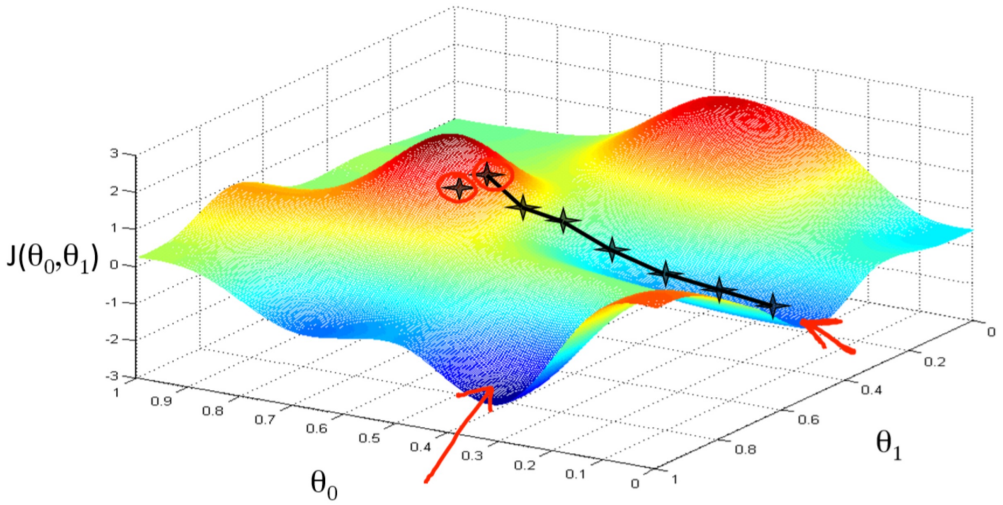
Thuật toán Gradien descent

- Gradient Descent là một thuật toán lặp tối ưu hoá được sử dụng trong các bài toán Machine Learning / Deep Learning (thường là các bài toán tối ưu lồi — Convex Optimization).
- Lưu ý:
  - Global minimum (maximum) là giá trị nhỏ nhất (lớn nhất) của hàm số, còn local minimum (maximum) là giá trị cực tiểu (cực đại) hay còn biết đến là giá trị cực tiểu (cực đại) địa phương.
  - Global minimum (maximum) là một trường hợp đặc biệt của Local minimum (maximum)



Thuật toán Gradien descent

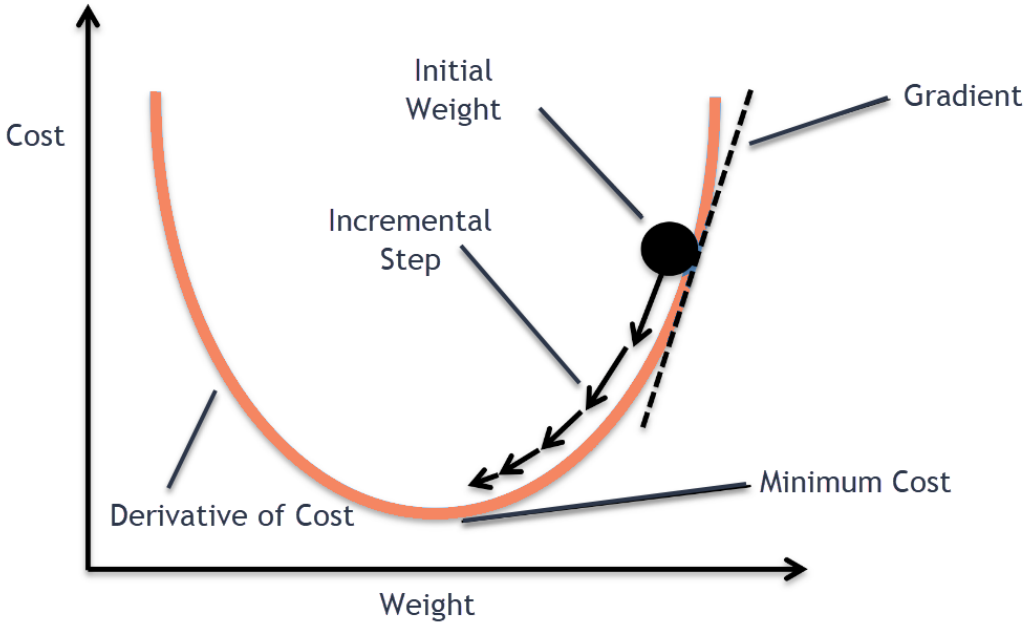
- Với các bài toán Machine Learning, thường rất phức tạp để có thể tìm được giá trị nhỏ nhất (global minimum).



- giá trị **Global Minimum** có thể là ở bên phía gần với θ_0
- Tuy nhiên, việc tìm ra các điểm cực **tiểu (local minimum)** sẽ dễ dàng hơn rất nhiều. Và ở một mức độ nào đó coi giá trị local minimum là nghiệm cần tìm của bài toán.

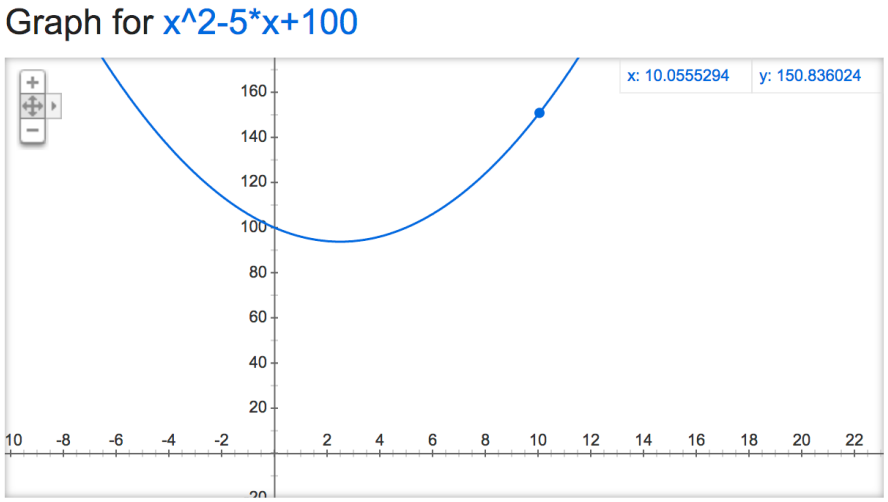
Thuật toán Gradien descent

- Cách mà thuật toán Gradient Descent thực hiện là lấy đạo hàm của Cost Function theo θ (hay còn gọi là Weight - trọng số). Chúng ta đang cố gắng đưa chấm đen trên hình càng gần về điểm Minimum càng tốt bằng cách di chuyển ***ngược dấu*** với đạo hàm.



Thuật toán Gradien descent

- Ví dụ: $y = x^2 – 5x + 100$
  - với điểm màu xanh đang ở toạ độ $x_t \approx 10$ và chúng ta mong muốn điểm này dần về vị trí nhỏ nhất x = 2.5



Thuật toán Gradien descent

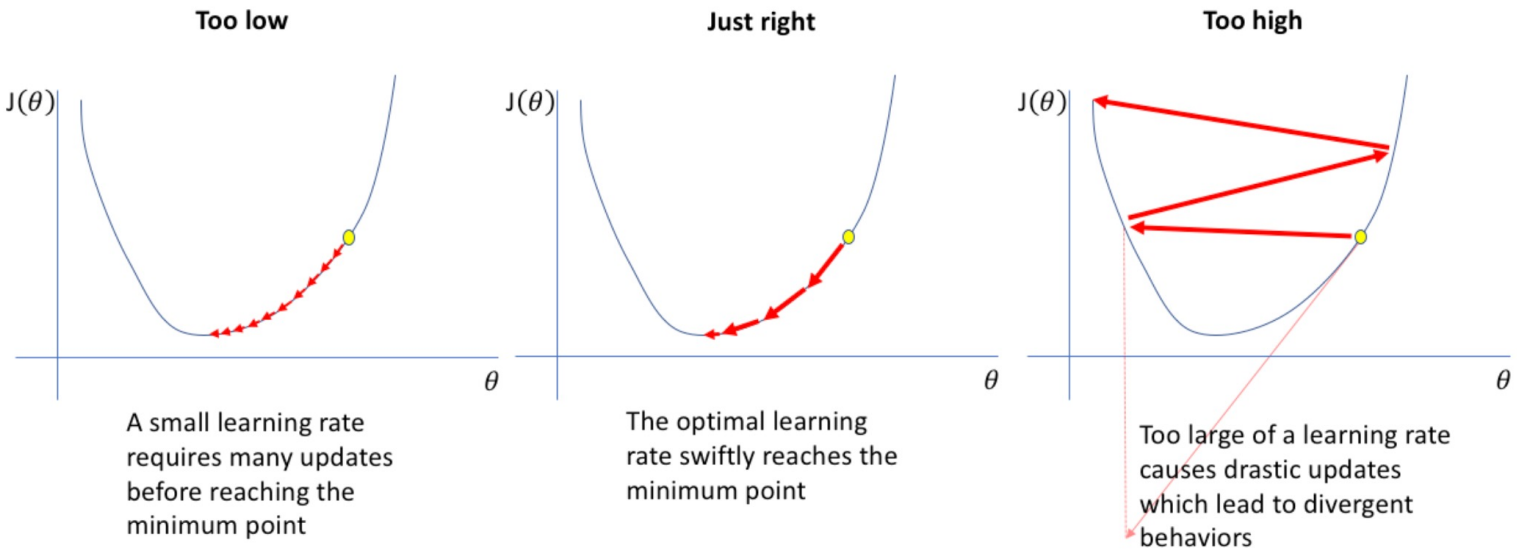
- Xét đạo hàm của hàm số $y = x^2 - 5x + 100$ ta có:
  - $y' = 2*x - 5$
  - $x_t = 10 \Rightarrow y' = 5$
- Đạo hàm trong trường hợp này mang dấu dương, khi đó ta sẽ thực hiện lặp lại việc di chuyển x ngược dấu với đạo hàm cho đến khi đạt giá trị cực tiểu:

$$x_{new} = x_t - \alpha * f'(x_t)$$

- Với α là một số dương được gọi là ***Learning Rate*** (tốc độ học tập). Dấu trừ biểu thị cho việc thực hiện đi ngược dấu với đạo hàm.

Thuật toán Gradien descent

- Learning Rate - Tốc độ học tập của thuật toán (LR), thường được ký hiệu là α.
- LR đóng vai trò quan trọng trong quá trình training của thuật toán. Việc lựa chọn LR không phù hợp có thể sẽ dẫn tới các trường hợp sau đây:



Thuật toán Gradien descent

- Trường hợp 1: Learning Rate quá nhỏ. Trong trường hợp này, với LR quá nhỏ so thì việc hội tụ sẽ rất lâu. Trong trường hợp xấu khác, thuật toán sẽ bị mắc kẹt ở local minimum không tốt (chẳng hạn như mắc kẹt ở vị trí local).
- Trường hợp 2: Learning Rate vừa đủ.Trong trường hợp này, thuật toán sẽ hội tụ đến local minimum tốt - thậm trí có thể hội tụ tới vị trí global minimum.
- Trường hợp 3: Learning Rate quá lớn. Trong trường hợp này, thuật toán sẽ không thể hội tụ mà lặp đi lặp lại xung quanh vị trí local minimum.

Thuật toán Gradien descent

- Gradient Descent cho bài toán Linear Regression : Với bài toán hồi quy tuyến tính đơn biến, chúng ta có 2 tham số cần học θ_0 là và θ_1.

$$\mathcal{J}(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x) - y)^2$$

- Với $\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$
- Như vậy, khi áp dụng Gradient Descent cho bài toán này, chúng ta sẽ thực hiện:
  - Thực hiện đạo hàm $\mathcal{J}(\theta_0, \theta_1)$ lần lượt theo các tham số θ_0 và θ_1.
  - Lặp lại cho đến khi hội tụ:

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y)$$

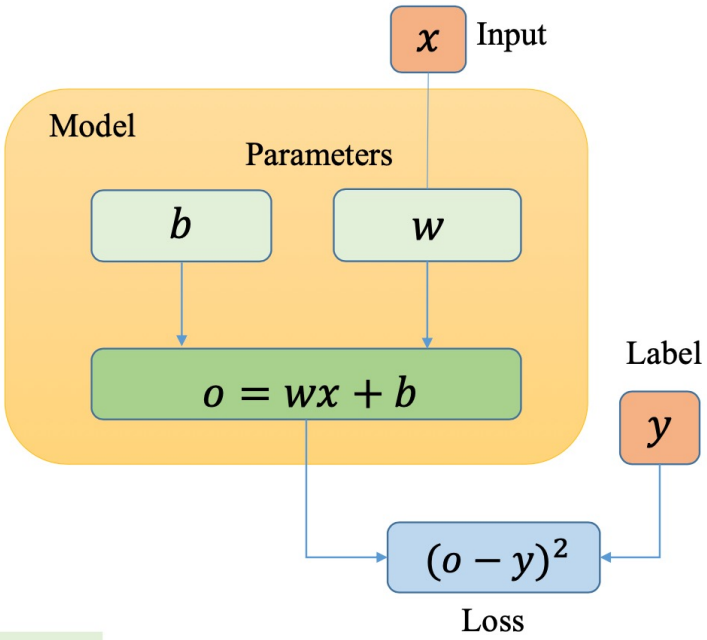
$$\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left((h_\theta(x_i) - y)\right)x_i$$

Thuật toán Gradien descent

- Điều kiện dừng của bài toán có thể là:
  - ✓ Kết thúc số lượng vòng lặp đã định sẵn.
  - ✓ Giá trị của hàm lỗi (Cost Function) đủ nhỏ.
  - ✓ Giá trị của hàm lỗi (Cost Function) không thay đổi sau `n` vòng lặp (với `n` là số nguyên người lập trình cài đặt, ví dụ: dừng sau 20 vòng lặp mà hàm lỗi không đổi).

Thuật toán Gradien descent

Diagram



Cheat sheet

Tính output o

$$o = \boldsymbol{wx} + b$$

Tính Loss

$$L = (o - y)^2$$

Tính đạo hàm

$$L'_{w_j} = 2x_j(o - y)$$

$$L'_b = 2(o - y)$$

Cập nhật tham số

$$w_j = w_j - \eta L'_{w_j}$$

$$b = b - \eta L'_b$$

Thuật toán Batch Gradien descent

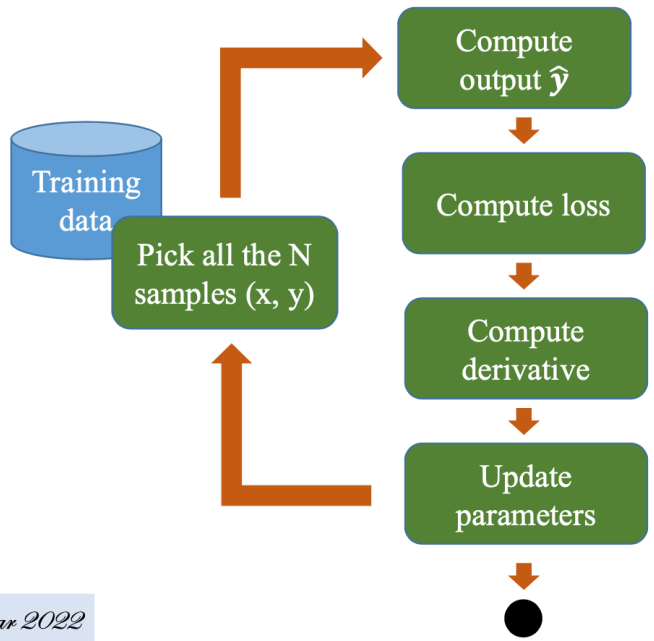
- Ý tưởng:
  - đưa toàn bộ X_train và Y_train vào để train
  - Với cách này, do đưa toàn bộ X_train và Y_train (nên gọi là batch) vào nên gradient ở mỗi vòng lặp ổn định. Cách này được khuyến khích sử dụng khi hàm cost của mình biết rõ là convex (không có nhiều hơn 1 điểm tối ưu cục bộ).
  - Nhược điểm của phương pháp này là với bộ dữ liệu lớn (hàng triệu dữ liệu) thì mỗi lần thực hiện tính toán đạo hàm tại mỗi vòng lặp sẽ rất tốn thời gian.

Thuật toán Batch Gradien descent

- input: Tập huấn luyện (X, y), learningRate, theta0, theta1, số lần huấn luyện(iters)
- output: giá trị theta0, theta1 đã được cập nhật

❖ **House price prediction**

❖ **N-sample training**



1) Pick all the N samples $(x^{(i)}, y^{(i)})$ from training data

2) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = wx^{(i)} + b \quad \text{for } 0 \le i < N$$

3) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \quad \text{for } 0 \le i < N$$

4) Tính đạo hàm

$$L_w'^{(i)} = 2x(\hat{y}^{(i)} - y^{(i)})$$
$$L_b'^{(i)} = 2(\hat{y}^{(i)} - y^{(i)}) \quad \text{for } 0 \le i < N$$

5) Cập nhật tham số

$$w = w - \eta \frac{\sum_i L_w'^{(i)}}{N}$$
$$b = b - \eta \frac{\sum_i L_b'^{(i)}}{N}$$

Learning rate η

Year 2022

Thuật toán Gradien descent (Mini-Batch)

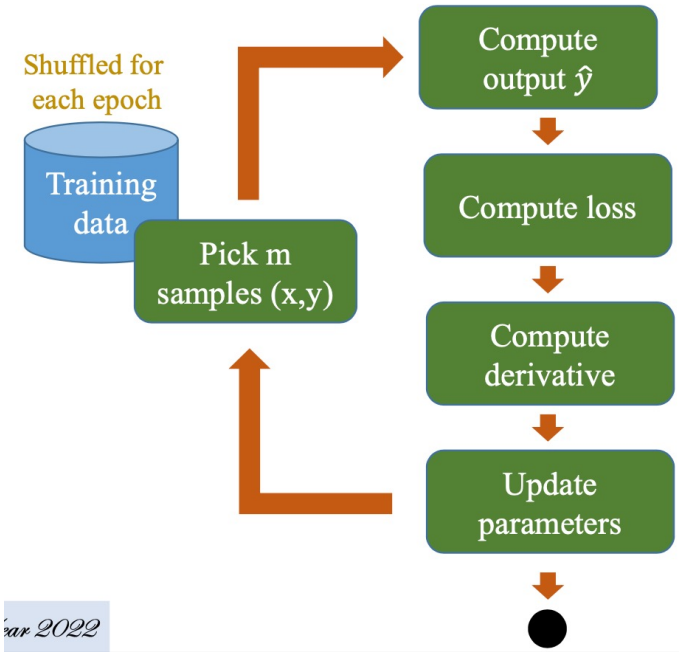
- Ý tưởng
  - Đúng theo cái tên của nó là Mini-batch tức chúng ta sẽ lấy một số lượng k dữ liệu training với $1 < k < m$. Ta sẽ chia mỗi mini-batch có k dữ liệu, có trường hợp mini-batch cuối cùng sẽ ít hơn k.nếu m không chia hết cho k.
  - Chú ý rằng trong mỗi Epoch chúng ta sẽ xáo trộn dữ liệu ngẫu nhiên rồi chia vào mỗi mini-batch.

Thuật toán Gradien descent (Mini-Batch)

- input: Tập huấn luyện (X, y), learningRate, theta0, theta1, số lần huấn luyện(iters)
- output: giá trị theta0, theta1 đã được cập nhật

❖ **House price prediction**

❖ **m-sample training (1<m<N)**



1) Pick m samples $(x^{(i)}, y^{(i)})$ from training data

2) Tính output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = wx^{(i)} + b \quad \text{for } 0 \le i < m$$

3) Tính loss

$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \quad \text{for } 0 \le i < m$$

4) Tính đạo hàm

$$L_w'^{(i)} = 2x(\hat{y}^{(i)} - y^{(i)})$$
$$L_b'^{(i)} = 2(\hat{y}^{(i)} - y^{(i)}) \quad \text{for } 0 \le i < m$$

5) Cập nhật tham số

$$w = w - \eta \frac{\sum_i L_w'^{(i)}}{m}$$
$$b = b - \eta \frac{\sum_i L_b'^{(i)}}{m}$$

Learning rate η

Thuật toán Gradien descent (Mini-Batch)

- Thực hiện đạo hàm $\mathcal{J}(\theta_0, \theta_1)$ lần lượt theo các tham số θ_0 và θ_1 với m = batch_size

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y)$$

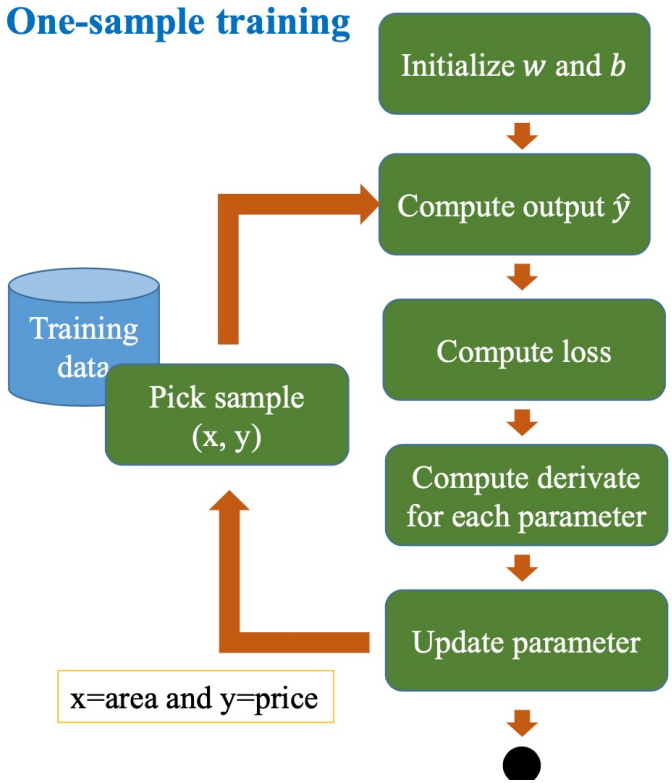
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x_i) - y)\right) x_i$$

Thuật toán Gradien descent (Stochsatic)

- Ý tưởng:
  - Trong Stochastic Gradient Descent (SGD), tại mỗi vòng lặp ta chỉ tính đạo hàm của hàm mất mát dựa trên chỉ một điểm dữ liệu x_i rồi cập nhật theta dựa trên đạo hàm này.
  - Chú ý hàm mất mát ở đây được lấy trung bình trên mỗi điểm dữ liệu nên đạo hàm tại một điểm cũng được kỳ vọng là khá gần với đạo hàm của hàm mất mát trên mọi điểm dữ liệu.

Thuật toán Gradien descent (Stochsatic)

- input: Tập huấn luyện (X, y), learningRate, theta0, theta1, số lần huấn luyện(iters)
- output: giá trị theta0, theta1 đã được cập nhật



1) Pick a sample (x, y) from training data

2) Compute the output $\hat{y}$

$$\hat{y} = wx + b$$

3) Compute loss

$$L = (\hat{y} - y)^2$$

4) Compute derivative

$$L'_w = 2x(\hat{y} - y)$$
$$L'_b = 2(\hat{y} - y)$$

5) Update parameters

$$w = w - \eta L'_w$$
$$b = b - \eta L'_b$$

η is learning rate

Thuật toán Gradien descent (Stochsatic)

- Thực hiện đạo hàm $\mathcal{J}(\theta_0, \theta_1)$ lần lượt theo các tham số θ_0 và θ_1

$$\theta_0 := \theta_0 - \alpha \sum_{i=1}^{m} (h_\theta(x_i) - y)$$

$$\theta_1 := \theta_1 - \alpha \sum_{i=1}^{m} \left((h_\theta(x_i) - y)\right) x_i$$

Chuẩn hoá dữ liệu

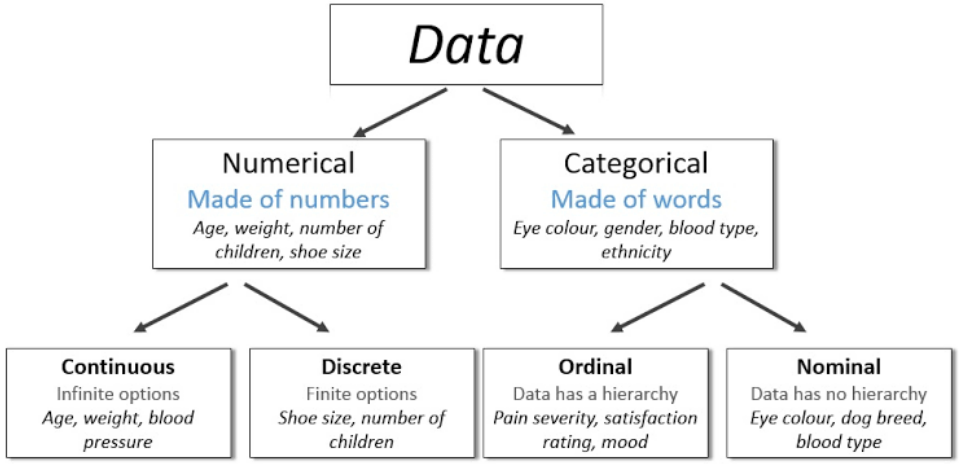
- Feature Scaling (hay còn gọi là Data Normalization) là một kỹ thuật được sử dụng để chuẩn hoá dữ liệu về cùng phạm vi (same range).
- Ví dụ: trong bài toán dự đoán giá nhà:
  - x1 = kích thước (0-2000 $feet^2$)
  - x2 = số lượng phòng (0 - 5)
- Chúng ta thấy rằng 2 Feature x_1 và x_2 có mức chênh lệnh nhau quá lớn về phạm vi: (0-2000) và (0-5). Điều này có thể khiến cho việc hội tụ tới kết quả tối ưu lâu và khó hơn.
- Chính vì vậy mà việc đưa các giữ liệu về cùng một phạm vi là một bước cần thiết trong các bài toán Machine Learning giúp việc hội tụ nhanh hơn.

Chuẩn hoá dữ liệu

- Khi các biến trong dữ liệu có độ lớn khác nhau: Việc chuẩn hóa dữ liệu giúp đưa tất cả các biến về cùng một thang đo, đảm bảo rằng tất cả các biến đều đóng góp bình đẳng vào quá trình học tập.
- Khi dữ liệu không tuân theo phân phối chuẩn: Chuẩn hóa dữ liệu có thể giúp đưa dữ liệu gần với phân phối chuẩn hơn, giúp các thuật toán học máy hoạt động tốt hơn.
- Khi sử dụng các thuật toán học máy giả định dữ liệu tuân theo phân phối chuẩn: Chuẩn hóa dữ liệu là cần thiết để đảm bảo rằng các giả định này được đáp ứng.
- Khi dữ liệu có nhiễu hoặc điểm dữ liệu ngoại lệ: Chuẩn hóa dữ liệu có thể giúp giảm thiểu ảnh hưởng của những yếu tố này đến mô hình.

Chuẩn hoá dữ liệu

- Với biến dữ liệu là biến liên lục hoặc rời rạc (**Numerical**) .
  - Các kỹ thuật chuẩn hoá có thể áp dụng là **Rescaleling** hoặc **Normalization**
- Với biến dữ liệu là biến định danh hoặc thứ bậc (**Categorical)** .
  - Các kỹ thuật chuẩn hoá có thể áp dụng là **OneHotEncoder**, **LabelEncoder**



Chuẩn hoá dữ liệu

- **Rescaling: p**hương pháp này được gọi là **min-max scaling**, là phương pháp đơn giản nhất để đưa các đặc trưng về cùng khoảng [-1;1] hay [0;1]. Nếu muốn đưa các đặc trưng về đoạn [0;1] thì công thức là:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Trong đó:
  - x: giá trị ban đầu.
  - x': giá trị đã được chuẩn hoá.
  - x_{min}: giá trị nhỏ nhất của x trong bộ dữ liệu training.
  - x_{max}: giá trị lớn nhất của x trong bộ dữ liệu training.

Chuẩn hoá dữ liệu

- **Normalization và Standardization :** Việc Normalization là đưa tất cả các điểm dữ liệu về dạng phân phối chuẩn (normal distribution).
- phân phối chuẩn còn gọi là phân phối Gauss hay (Hình chuông Gauss), là một phân phối xác suất cực kì quan trọng trong nhiều lĩnh vực.
- Nó là họ phân phối có dạng tổng quát giống nhau, chỉ khác tham số vị trí (giá trị trung bình μ) và tỉ lệ (phương sai σ^2)

Chuẩn hoá dữ liệu

- **Standardization**: (còn được gọi là **z-score normalization**) biến đổi dữ liệu sao cho phân phối của kết quả có giá trị trung bình (μ) bằng 0 và độ lệch chuẩn (σ) bằng 1. Khi đó, công thức chuẩn hoá là:

$$x' = \frac{x - x_{mean}}{\sigma}$$

- Trong đó:
  - x': giá trị sau chuẩn hoá.
  - x: giá trị ban đầu.
  - x_{mean}: giá trị mean của bộ dữ liệu.
  - σ: độ lệch chuẩn (standard deviation).

Chuẩn hoá dữ liệu

- **Normalization:** Nó chỉ khác phương pháp z-score ở mẫu số:

$$x' = \frac{x - x_{mean}}{x_{max} - x_{min}}$$

- Trong đó:
  - x': giá trị sau chuẩn hoá.
  - x: giá trị ban đầu.
  - x_{mean}: giá trị mean của bộ dữ liệu.
  - x_{max}, x_{min}: giá trị lớn nhất / nhỏ nhất trong bộ dữ liệu.
- Trong phương pháp này, giá trị lớn nhất có được là 1 và nhỏ nhất là 0. Nên dữ liệu nằm trong đoạn [0;1].

Chuẩn hoá dữ liệu

- Một số lưu ý khi chuẩn hoá dữ liệu:
  - Khi **Scaling**: thay đổi phạm vi của dữ liệu.
  - Khi **Normalization** : thay đổi hình dạng phân phối của dữ liệu.
  - Một số thuật toán học máy, chẳng hạn như hồi quy tuyến tính và mạng nơ-ron nhân tạo, giả định rằng dữ liệu tuân theo phân phối chuẩn. Do đó, chuẩn hóa dữ liệu có thể cải thiện hiệu suất của các thuật toán này.
  - Các thuật toán học máy khác, chẳng hạn như cây quyết định và SVM, không bị ảnh hưởng bởi sự khác biệt về tỷ lệ của các biến. Do đó, chuẩn hóa dữ liệu có thể không cần thiết cho các thuật toán này.

Bài toán hồi quy đa biến

- Hồi quy tuyến đa biến (Linear Regression with Multi-Variable) được sử dụng khi muốn dự đoán giá trị đầu ra `y` từ giá trị đầu vào `x`. Khi đó hàm giả thuyết của có dạng:

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

- Lưu ý:
  - $\theta_0, \theta_1, \theta_2, \ldots, \theta_n$ là các tham số cần tối ưu
  - $x_1, x_2, x_3, \ldots, x_n$ là các đặc trưng đầu vào
  - đối với bài toán hồi quy đã biến ta sử dụng thuật toán SGD để tìm giá trị tham số
  - và hàm mất mát (Loss) vẫn sư dụng MSE để tính toán

Bài toán hồi quy đa biến

- Sử dụng lại hàm mất mát ở bài toán hồi quy 1 biến

$$\mathcal{J}(\theta_0, \theta_1) = \frac{1}{2\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} (\mathrm{h}_\theta(\mathrm{x}^{\mathrm{i}}) - \mathrm{y}^{\mathrm{i}})^2$$

- Chú thích:
  - m là số quan sát (số dữ liệu)
  - x^{i}: với i = 1...n là các đặc trưng đầu vào của bài toán

Bài toán hồi quy đa biến

- Gradient descent cho bài toán hồi quy đa biến:

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)\, x_0^i$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)\, x_1^i$$

.....

$$\theta_n := \theta_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)\, x_n^i$$

Một số phương pháp đánh giá mô hình

- **Mean Squared Error**
  - Về cơ bản, MSE tìm thấy sai số bình phương trung bình giữa các giá trị được dự đoán và thực tế.
  - MSE là thước đo chất lượng của một công cụ ước tính - nó luôn không âm và các giá trị càng gần 0 càng tốt.
  - số liệu phổ biến nhất được sử dụng cho các bài toán hồi quy.
  - MSE càng thấp thì dự báo càng tốt.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

  - Trong đó:
    - ✓ n là số điểm dữ liệu,
    - ✓ y_i là giá trị quan sát
    - ✓ $\hat{y}_i$ là giá trị dự đoán.

Một số phương pháp đánh giá mô hình

- **Mean Absolute Error**
  - đo độ lớn trung bình của các lỗi trong một tập hợp các dự đoán mà không cần xem xét hướng của chúng. Đó là giá trị trung bình trên mẫu thử nghiệm về sự khác biệt tuyệt đối giữa dự đoán và quan sát thực tế, trong đó tất cả các khác biệt riêng lẻ có trọng số bằng nhau.
  - MAE được biết đến là mạnh mẽ hơn đối với các yếu tố ngoại lai so với MSE.
  - MAE được biết đến là mạnh mẽ hơn đối với các yếu tố ngoại lai so với MSE.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

  - Trong đó:
    - ✓ n là số điểm dữ liệu,
    - ✓ x_i là giá trị thực và y_i là giá trị dự đoán.

Một số phương pháp đánh giá mô hình

Hàm mất mát	Khả năng áp dụng để phân loại	Khả năng áp dụng hồi quy	Nhạy cảm với các ngoại lệ
Lỗi bình phương trung bình (MSE)	✗	✓	Cao
Sai số tuyệt đối trung bình (MAE)	✗	✓	Thấp

Một số phương pháp đánh giá mô hình

- **Root Mean Square Error**
  - độ lệch chuẩn của các phần dư (sai số dự đoán).
  - Phần dư là thước đo khoảng cách từ các điểm dữ liệu đường hồi quy; RMSE là thước đo mức độ dàn trải của những phần dư này, nói cách khác, nó cho biết mức độ tập trung của dữ liệu xung quanh đường phù hợp nhất.
  - RMSE luôn không âm và giá trị 0 (hầu như không bao giờ đạt được trong thực tế) sẽ chỉ ra sự phù hợp hoàn hảo với dữ liệu. Nói chung, RMSE thấp hơn sẽ tốt hơn RMSE cao hơn.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$$

RMSD = root-mean-square deviation

i = variable i

N = number of non-missing data points

x_i = actual observations time series

$\hat{x}_i$ = estimated time series

Q&A

- Giải quyết bài toán hồi quy tuyết tính (đơn biến và đa biến) bằng phương Gradient descent
- Cài đặt được các biến thể của phương pháp GD
- Nhận biết được khi nào cần thiết chuẩn hoá dữ liệu