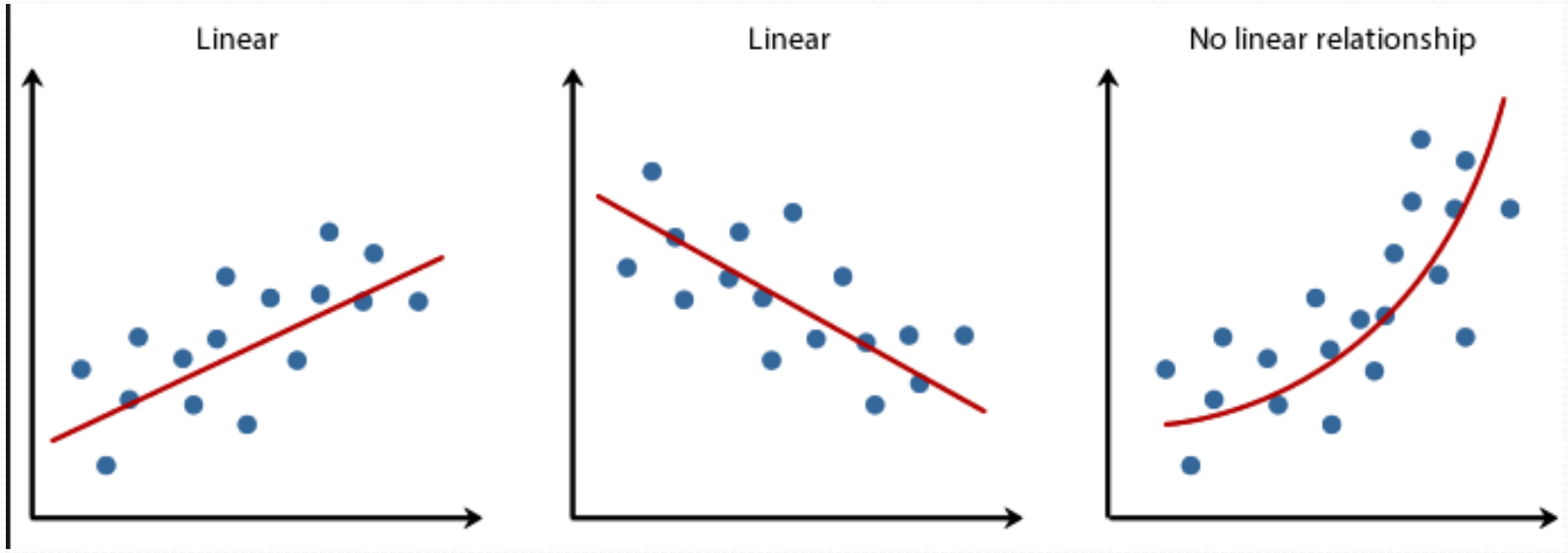
Linear Regression

Nội dung

- Khái niệm hồi qui tuyến tính (Linear Regression)
- Hồi qui tuyến tính đơn biến
- Hồi qui tuyến tính đa biến
  - Phương pháp ước lượng tham số
- Các mở rộng
- Linear Regression dùng Gradient Descent
- Câu hỏi và bài tập

Linear Regression

- Hồi quy tuyến tính:
  - Là phương pháp học máy có giám sát đơn giản, được sử dụng để dự đoán (predict) giá trị đầu ra (liên tục, dạng số).
  - Là phương pháp dựa trên thống kê để thiết lập mối quan hệ giữa một biến phụ thuộc và một nhóm tập hợp các biến độc lập.



Linear Regression

- Ví dụ:

Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

Notation:

m = Number of training examples

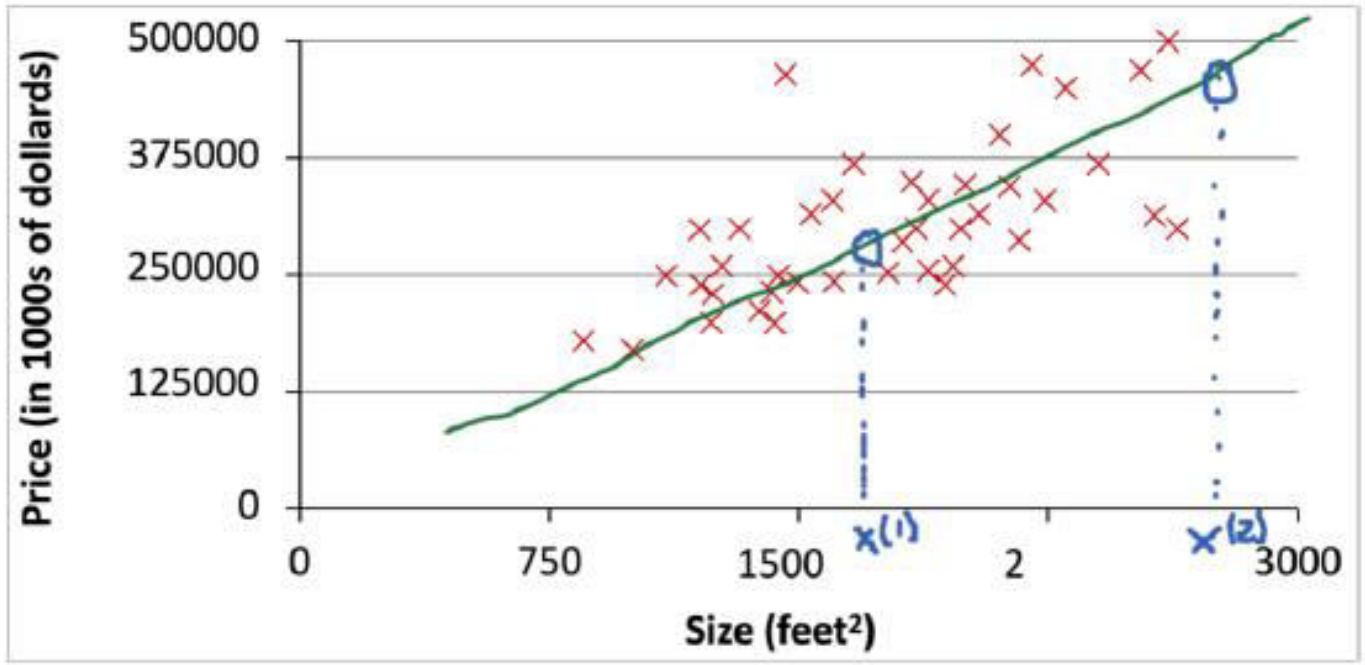
x's = "input" variable / features

y's = "output" variable / "target" variable

Linear Regression

- Ví dụ:

Housing Prices (Portland, OR)



Linear Regression

- Ví dụ:

Training Set

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

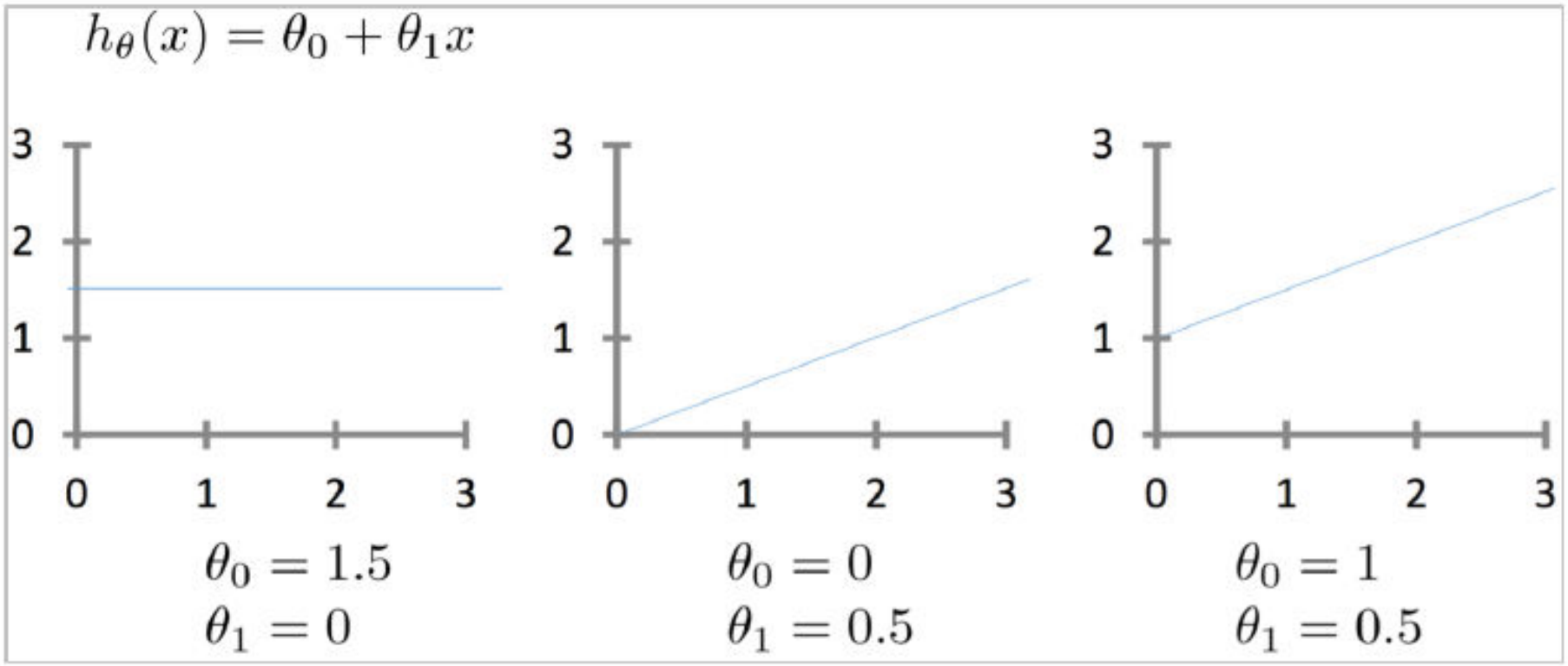
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i 's: Parameters

How to choose θ_i 's ?

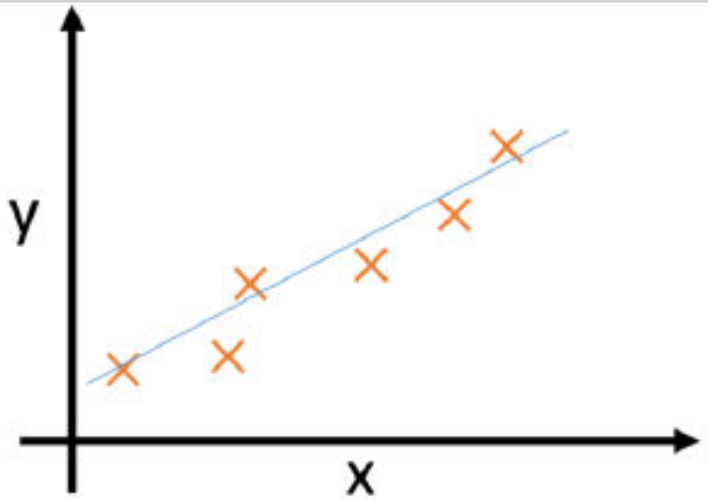
Linear Regression

- Ví dụ: Quan sát



Linear Regression

- Ví dụ:

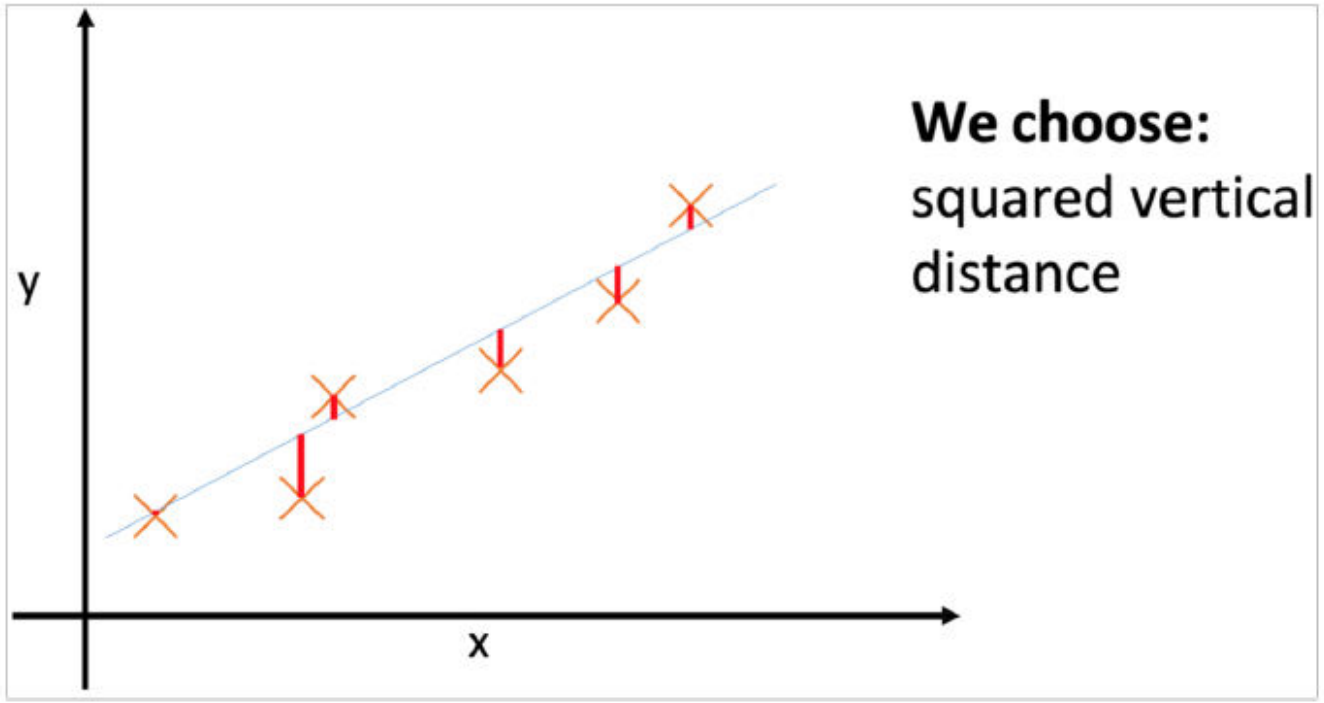


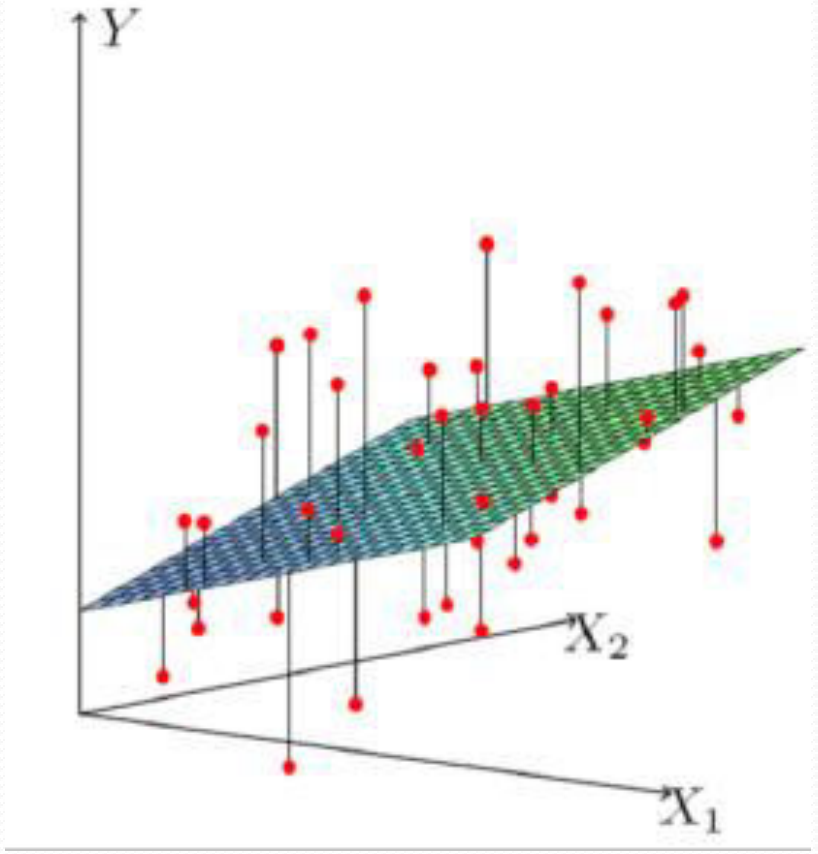
But what does "close" mean?

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

Linear Regression

- Ví dụ:





Simple Linear Regression

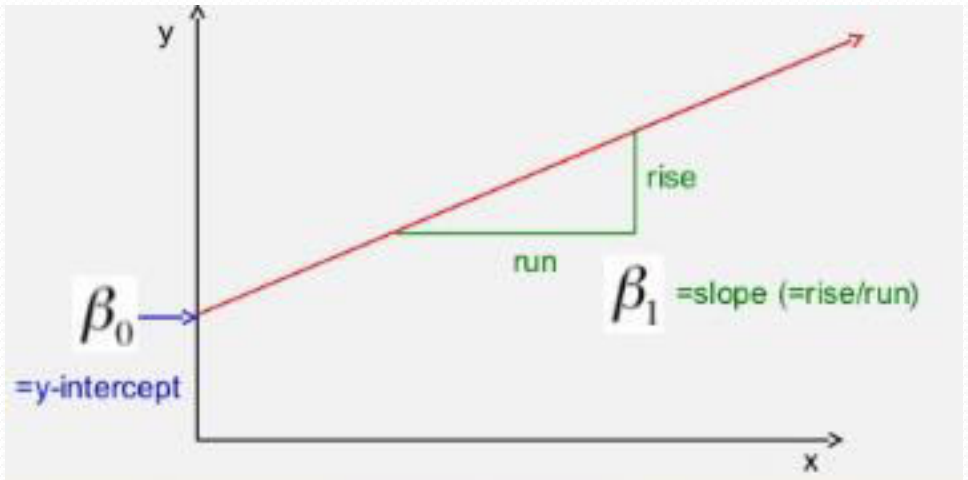
- Giả thuyết: Output Y và input X có mối quan hệ tuyến tính như sau

$$Y = \beta_0 + \beta_1 X$$

trong đó

β_0 intercept hệ số chặn

β_1 slope độ dốc



Simple Linear Regression

- Ta cần ước lượng giá trị β_o và β_1 .

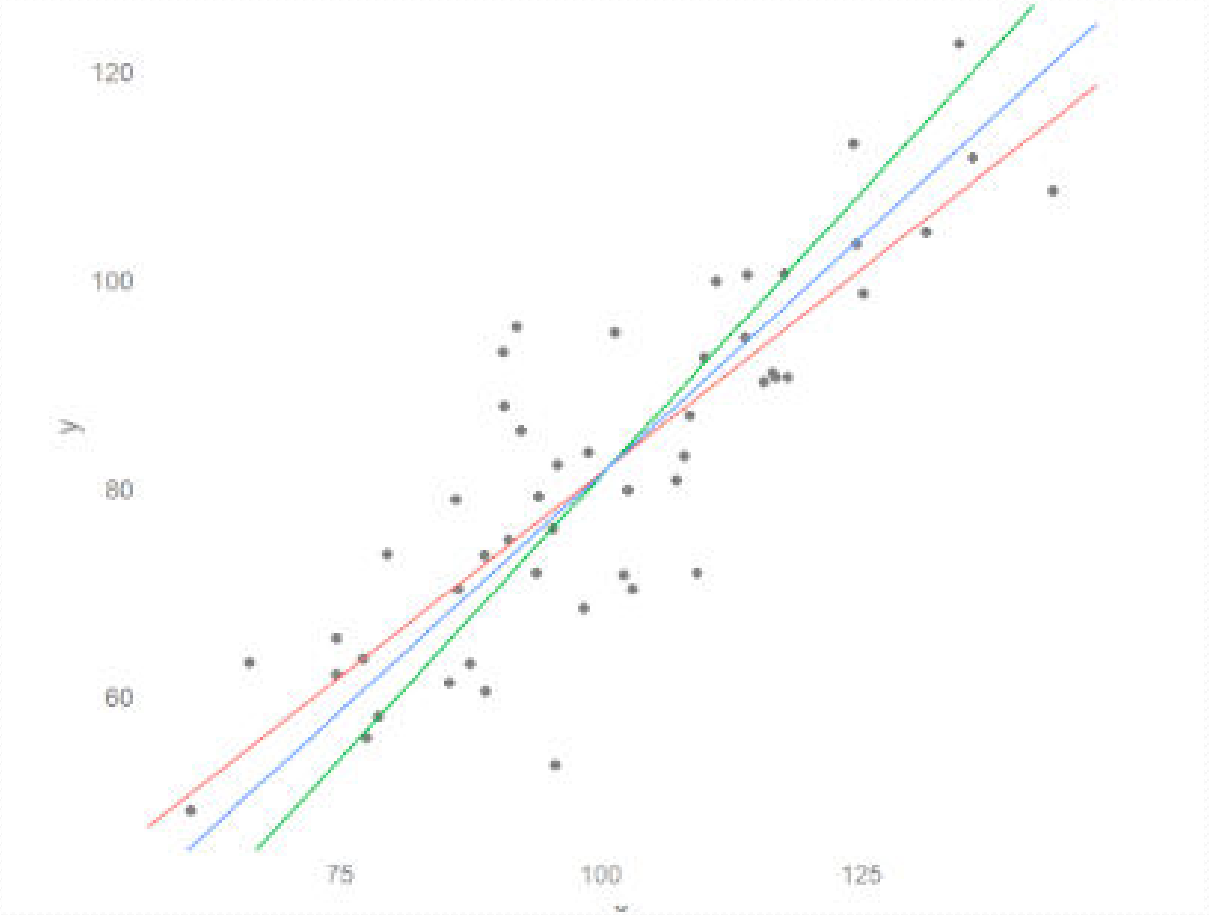
$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

- Chọn $\hat{\beta}_0$ và $\hat{\beta}_1$ sao cho mô hình khớp tốt nhất (good fit) đối với tập huấn luyện

$$Y^{(i)} \approx \hat{\beta}_0 + \hat{\beta}_1 X^{(i)}, \quad i = 1, \ldots, n$$

Simple Linear Regression

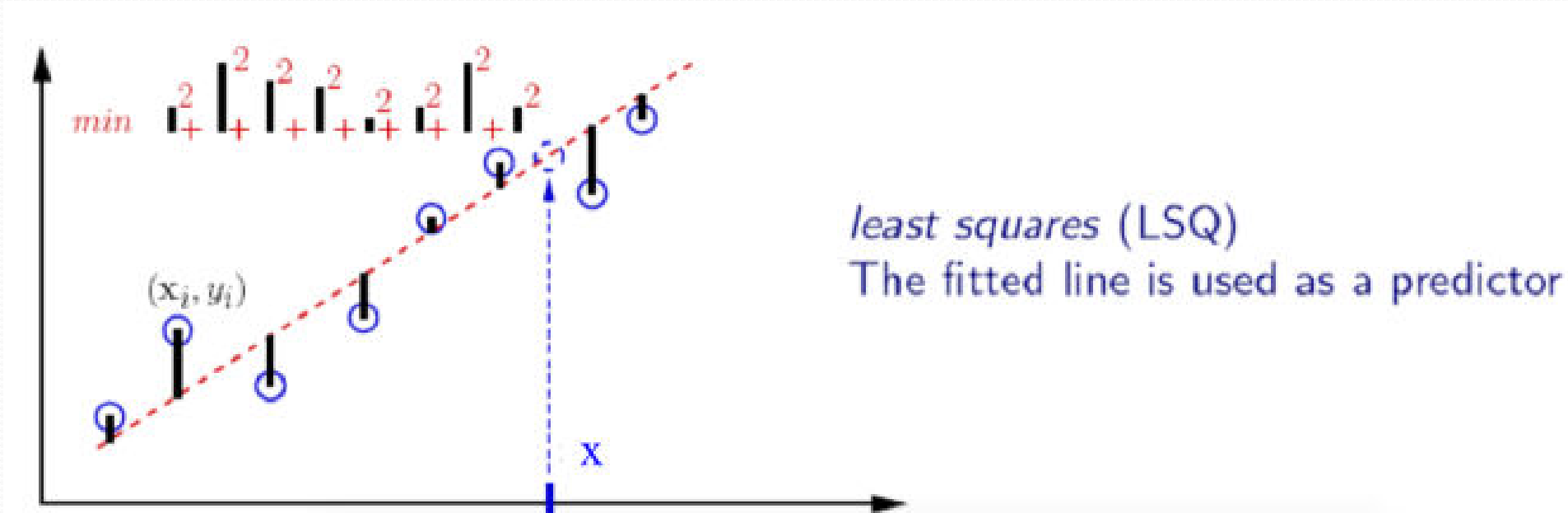
- Ví dụ: đường sấp xỉ nào tốt nhất



Simple Linear Regression

- **Bình phương tối thiểu**
- Lỗi bình phương trung bình (Mean squared error):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y^{(i)} - \hat{Y}^{(i)}\right)^2$$



Simple Linear Regression

- **Phương pháp ước lượng tham số :**

$$\min_{(\hat{\beta}_0,\hat{\beta}_1)} \left[\frac{1}{n} \sum_{i=1}^{n} \left(Y^{(i)} - \left(\hat{\beta}_0 + \hat{\beta}_1 X^{(i)} \right) \right)^2 \right]$$

- Solution:
  - Hệ số dốc của đường thẳng

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_x}$$

$$SS_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \quad \text{và} \quad SS_x = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Simple Linear Regression

- Hệ số chặn của đường thẳng

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

trong đó

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \qquad \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Simple Linear Regression

- Ví dụ:

X	Y
kilograms	cost $
17	132
21	150
35	160
39	162
50	149
65	170

$\bar{x} = 37.83$

$\bar{y} = 153.83$

$SS_{xy} = 891.83$

$SS_x = 1612.83$

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_x} = \frac{891.83}{1612.83} = 0.533$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = 153.83 - 0.553 \times 37.83 = 132.91$$

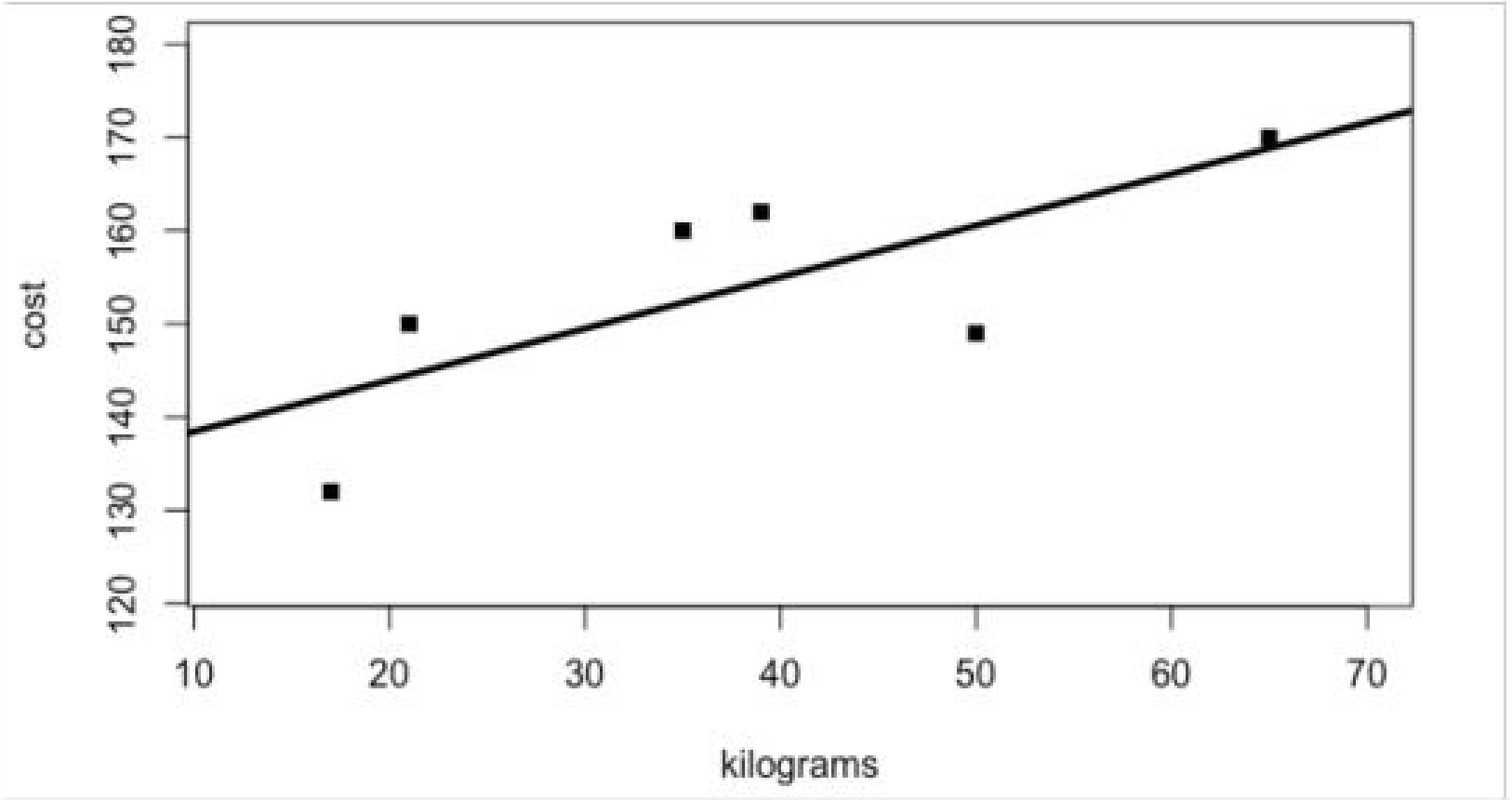
phương trình tìm được là

$Y = 132.91 + 0.553 * X$

khi thay đổi 1 kg của X, giá của Y thay đổi 0.553$

Simple Linear Regression

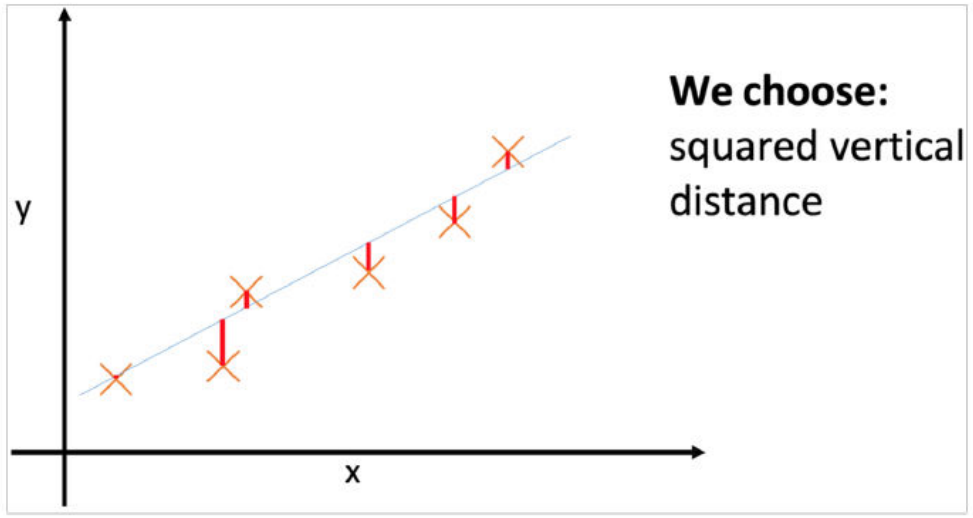
- Ví dụ:

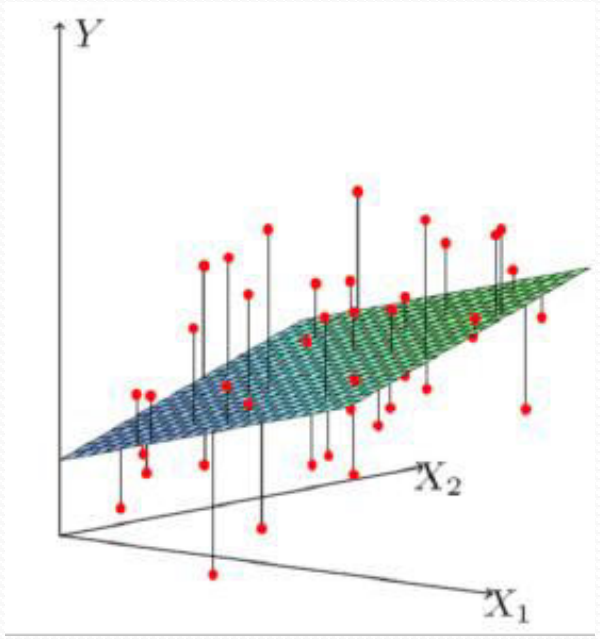


Linear Regression

- Hồi quy tuyến tính đa biến

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_d X_d$$





Linear Regression

- Phương pháp ước lượng tham số:

$$\hat{\beta} = \arg\min_{\beta} \|Y - X^T\beta\|^2$$

$$X = \begin{bmatrix} 1 & X^{(1)^T} \\ & \dots \\ 1 & X^{(n)^T} \end{bmatrix} \qquad \hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \dots \\ \hat{\beta}_a \end{bmatrix} \qquad Y = \begin{bmatrix} Y^{(1)} \\ \dots \\ Y^{(n)} \end{bmatrix}$$

Linear Regression

- Solution:

$$X^T X \hat{\beta} = X^T Y$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Linear Regression

- Ví dụ: Cho

$$\mathbf{y} = \begin{bmatrix} 6 \\ 9 \\ 12 \\ 5 \\ 13 \\ 2 \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} 1 & 3 & 9 & 16 \\ 1 & 6 & 13 & 13 \\ 1 & 4 & 3 & 17 \\ 1 & 8 & 2 & 10 \\ 1 & 3 & 4 & 9 \\ 1 & 2 & 4 & 7 \end{bmatrix} \qquad \hat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{bmatrix}$$

$$\boldsymbol{X}^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 3 & 6 & 4 & 8 & 3 & 2 \\ 9 & 13 & 3 & 2 & 4 & 4 \\ 16 & 13 & 17 & 10 & 9 & 7 \end{bmatrix} \qquad \boldsymbol{X}^T\boldsymbol{X} = \begin{bmatrix} 6 & 26 & 35 & 72 \\ 26 & 138 & 153 & 315 \\ 35 & 153 & 295 & 448 \\ 72 & 315 & 448 & 944 \end{bmatrix} \qquad \boldsymbol{X}^T\boldsymbol{y} = \begin{bmatrix} 47 \\ 203 \\ 277 \\ 598 \end{bmatrix}$$

Linear Regression

- Ví dụ

$$\hat{\beta} = (X^T X)^{-1} X^T y = \begin{bmatrix} 2.59578 & -0.15375 & -0.01962 & -0.13737 \\ -0.15375 & 0.03965 & -0.00014 & -0.00144 \\ -0.01962 & -0.00014 & 0.01234 & -0.00431 \\ -0.13737 & -0.00144 & -0.00431 & 0.01406 \end{bmatrix} \begin{bmatrix} 47 \\ 203 \\ 277 \\ 598 \end{bmatrix}$$

$$= \begin{bmatrix} 3.20975 \\ -0.07573 \\ -0.11162 \\ 0.46691 \end{bmatrix}$$

$$\hat{\beta}_0 = 3.20975 \quad \hat{\beta}_1 = -0.07573 \quad \hat{\beta}_2 = -0.11162 \quad \hat{\beta}_3 = 0.46691$$

$$\hat{y} = 3.20975 - 0.07573x_1 - 0.11162x_2 + 0.46691x_3$$

Đánh giá

- Các phương pháp đánh giá

root mean square error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}$$

mean Absolute Error

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

CASE 1: Evenly distributed errors

ID	Error	|Error|	Error^2
1	2	2	4
2	2	2	4
3	2	2	4
4	2	2	4
5	2	2	4
6	2	2	4
7	2	2	4
8	2	2	4
9	2	2	4
10	2	2	4

MAE	RMSE
2.000	2.000

CASE 2: Small variance in errors

ID	Error	|Error|	Error^2
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	3	3	9
7	3	3	9
8	3	3	9
9	3	3	9
10	3	3	9

MAE	RMSE
2.000	2.236

CASE 3: Large error outlier

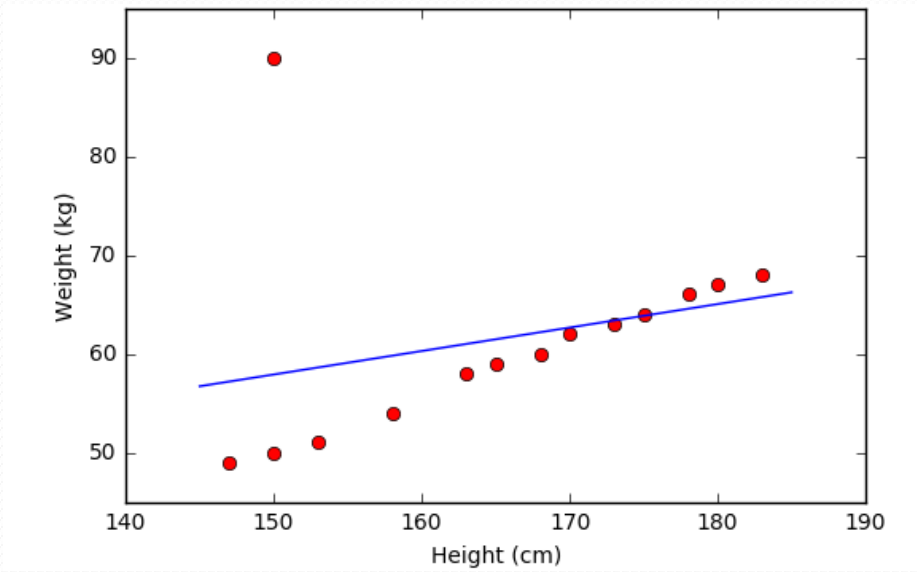
ID	Error	|Error|	Error^2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	20	20	400

MAE	RMSE
2.000	6.325

MAE and RMSE for cases of increasing error variance

Linear Regression

- Pros
  - Mô hình đơn giản, dễ hiểu
  - Dễ tìm nghiệm
  - Kết quả tốt khi dữ liệu quan sát nhỏ và tuyến tính
  - Có thể mở rộng
- Cons:
  - Nhạy cảm với dữ liệu ngoại lai (outliers)



Mở rộng

- Hàm số $Y \approx X^T \beta = \hat{Y}$ là hàm tuyến tính theo cả X và β . Tuy nhiên, Linear Regression có thể áp dụng cho các mô hình chỉ cần tuyến tính theo β

$$Y = \sum_{i=0}^{d} \beta_i \phi_i(X) = \beta^T \phi(X) \qquad \phi_0(X) = 1$$

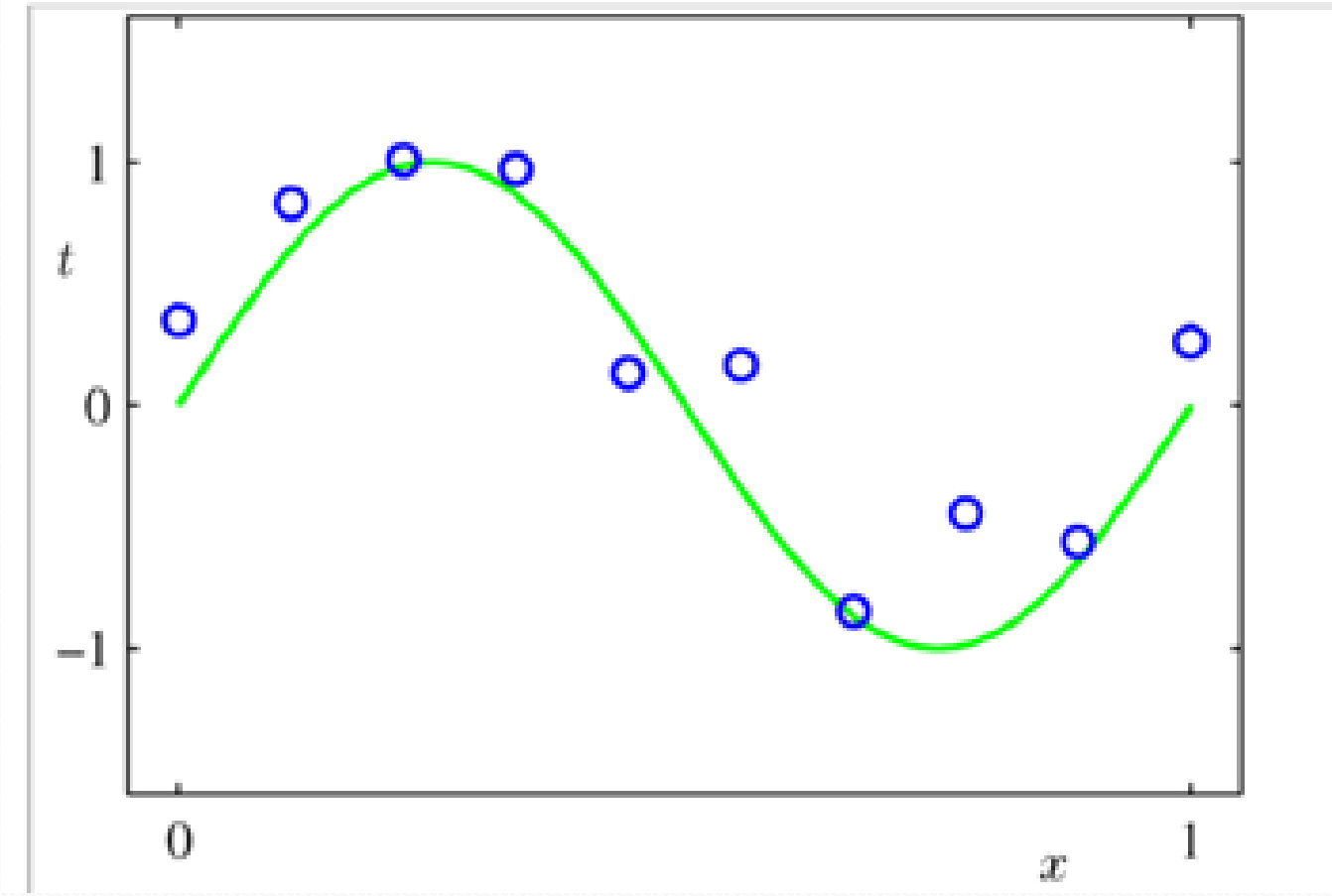
- Cho phép dùng linear regression để "fit" non-linear dataset

Mở rộng

- Ví dụ:
- Đơn giản $\phi(\mathbf{x}) = \mathbf{x}$
- Đa thức hoá $\phi_i(\mathbf{x}) = \mathbf{x}^i$

Mở rộng

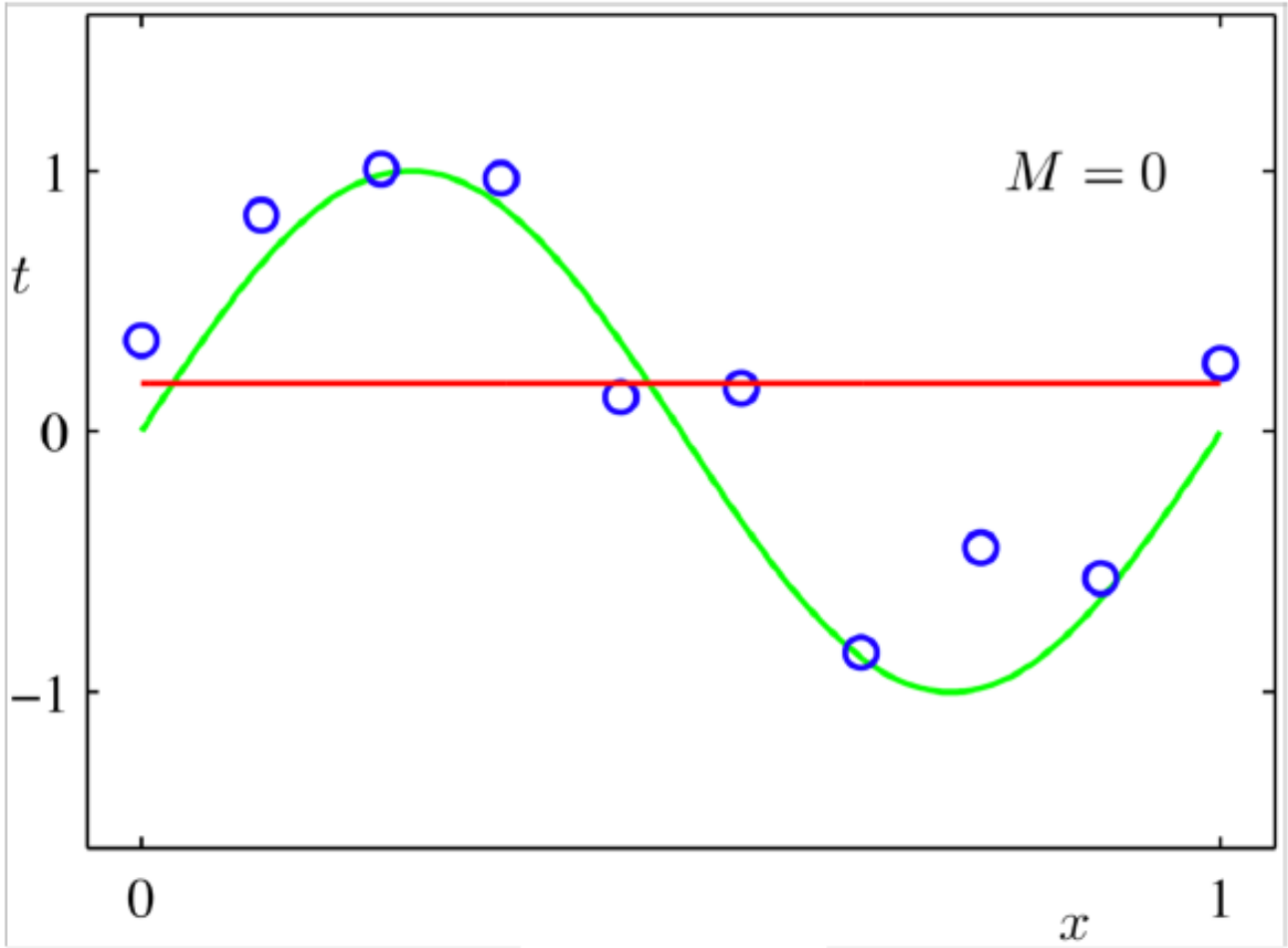
- Ví dụ:



$$Y = \beta_0 + \beta_1 X \ + \ldots + \beta_M X^M = \sum_{i=0}^{M} \beta_i X^i$$

Mở rộng

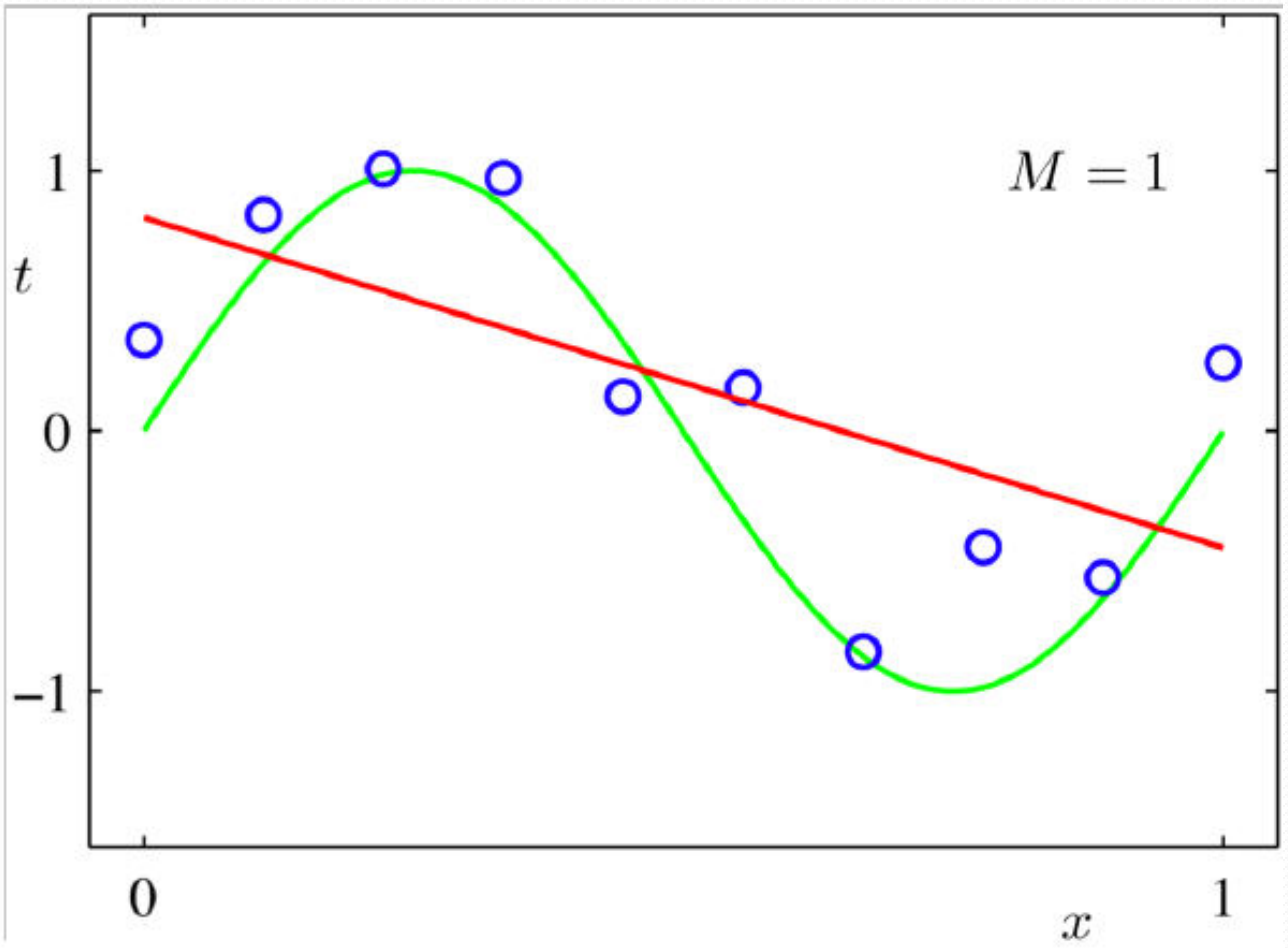
- Ví dụ:



$$Y = \beta_0$$

Mở rộng

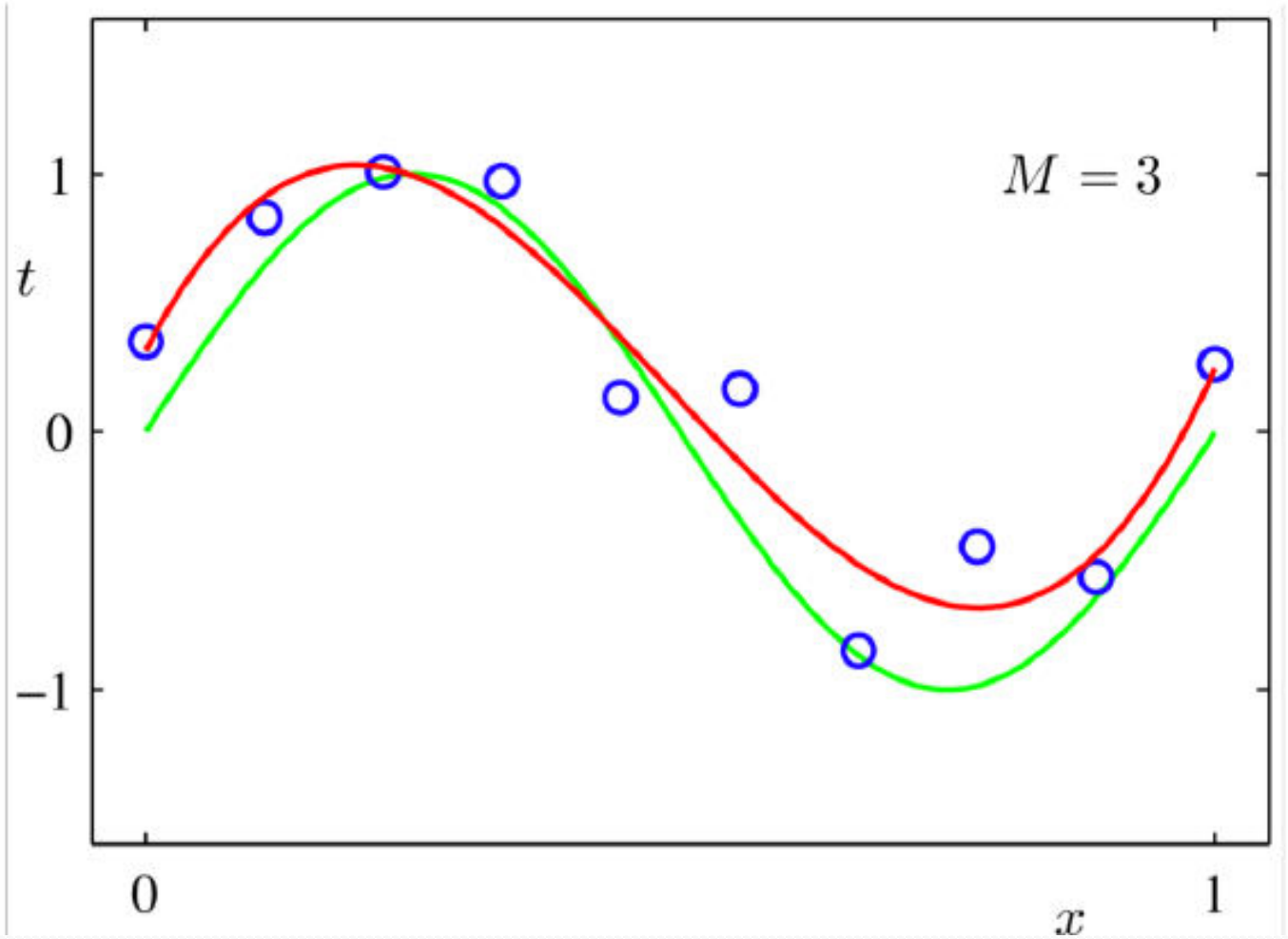
- Ví dụ:



$$Y = \beta_0 + \beta_1 X$$

Mở rộng

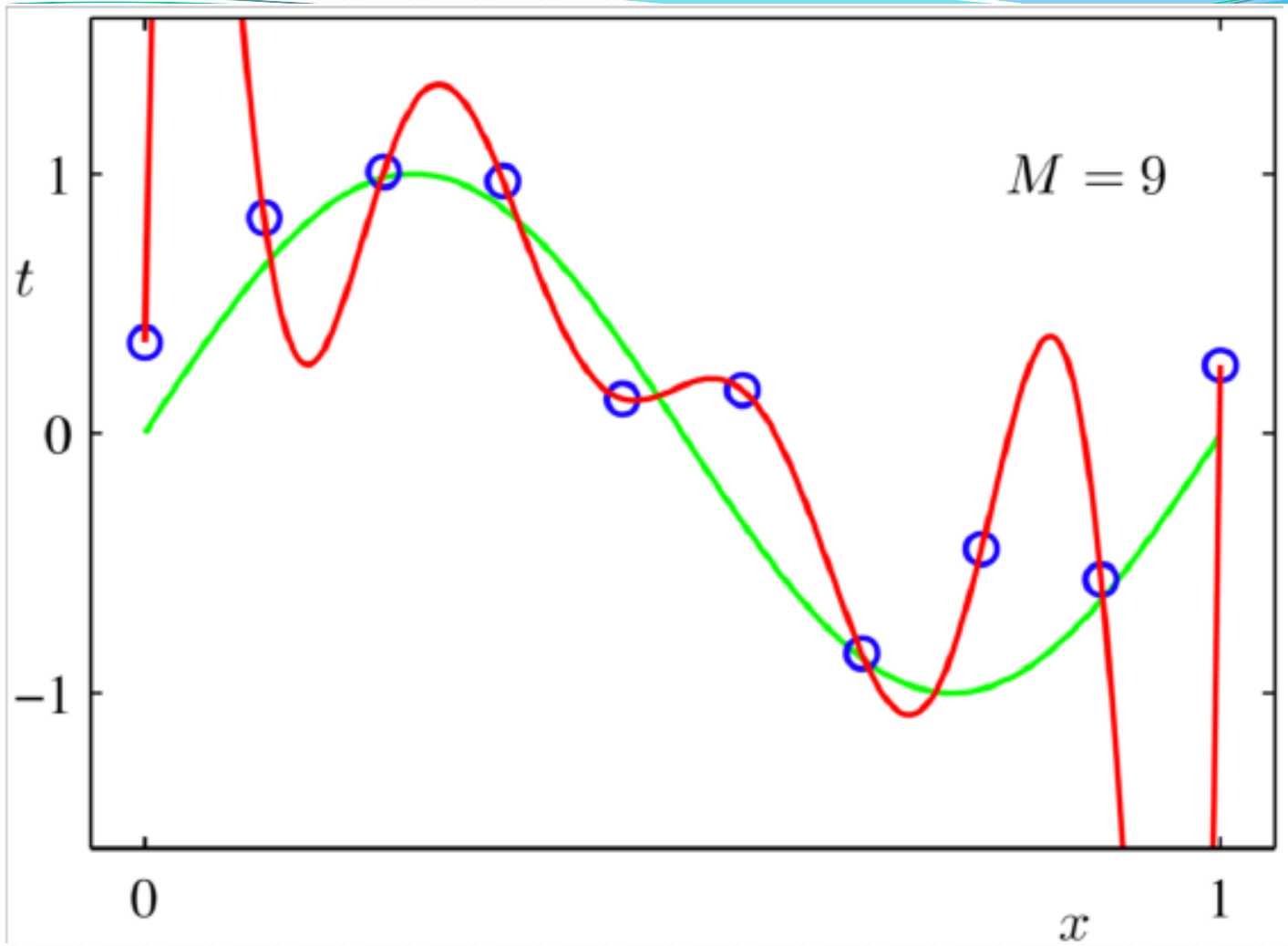
- Ví dụ:



$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$$

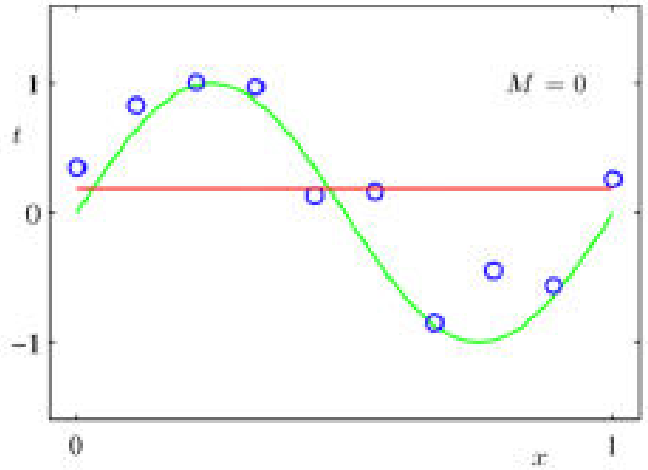
Mở rộng

- Ví dụ:

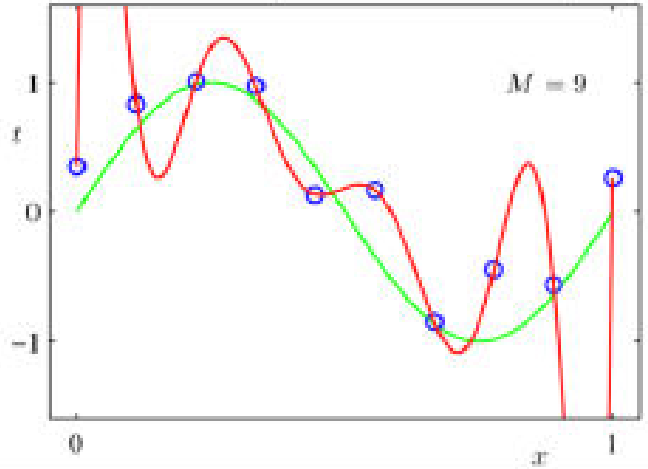


$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \ldots + \beta_9 X^9$$

Underfitting : The model is too simple - does not fit the data.



Overfitting : The model is too complex - fits perfectly, does not generalize.

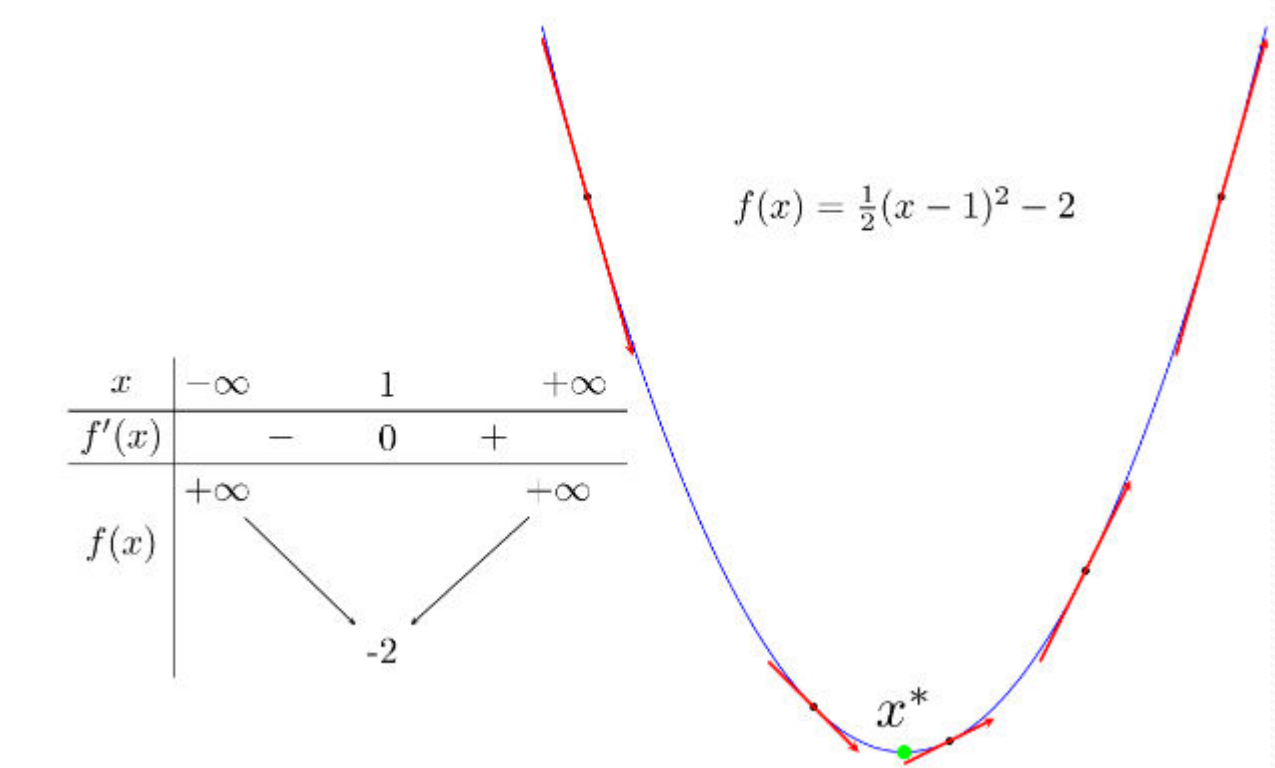


Tìm hiểu thêm

- Linear Regression using Gradient Descent

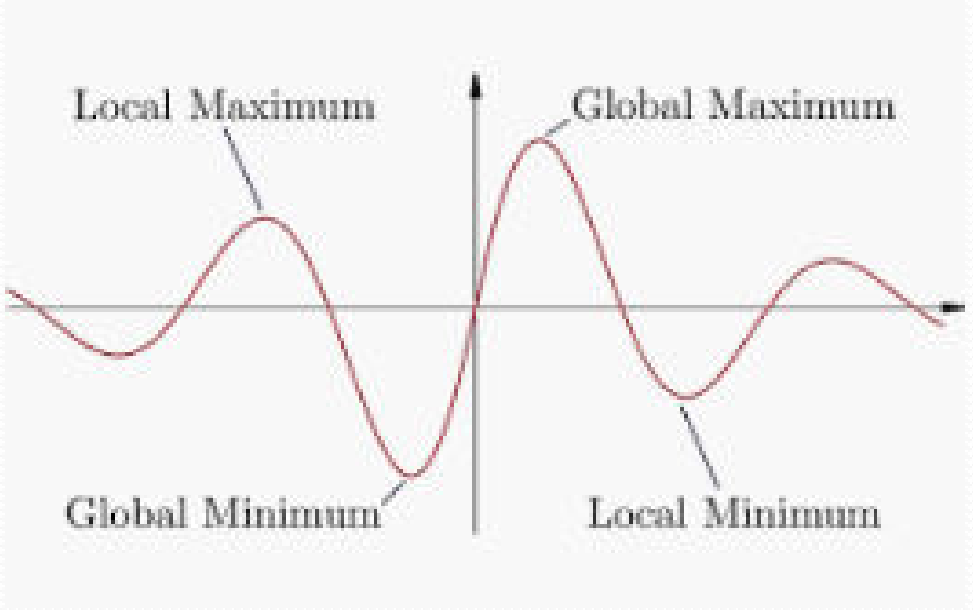
Gradient Descent

- Ví dụ:



x	$-\infty$		1		$+\infty$
$f'(x)$		$-$	0	$+$	
$f(x)$	$+\infty$		-2		$+\infty$

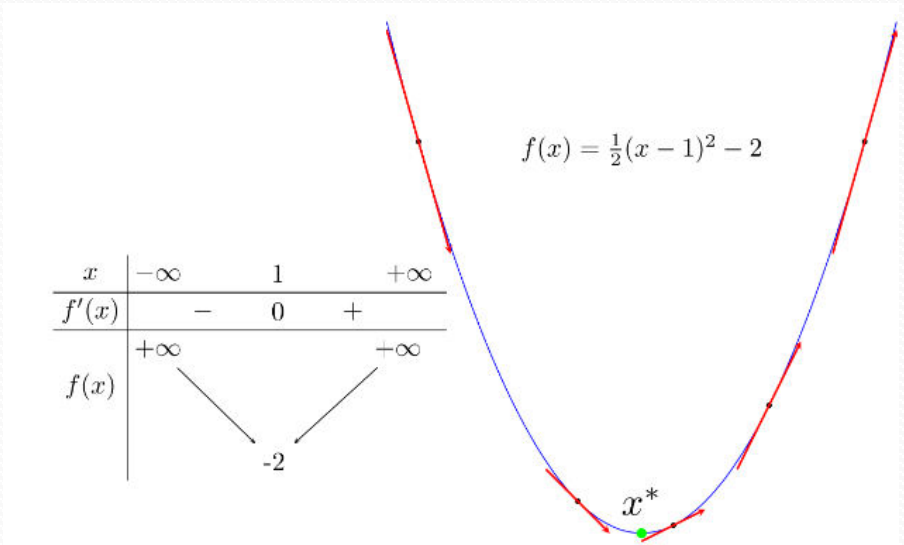
local minimum vs. global minimum



Gradient Descent

Ví dụ:

- local minimum x^* của hàm số là điểm có đạo hàm $f'(x^*)$ bằng 0
- đạo hàm của các điểm phía bên trái $\mathbf{x}^*$ là không dương
- đạo hàm của các điểm phía bên phải $\mathbf{x}^*$ là không âm

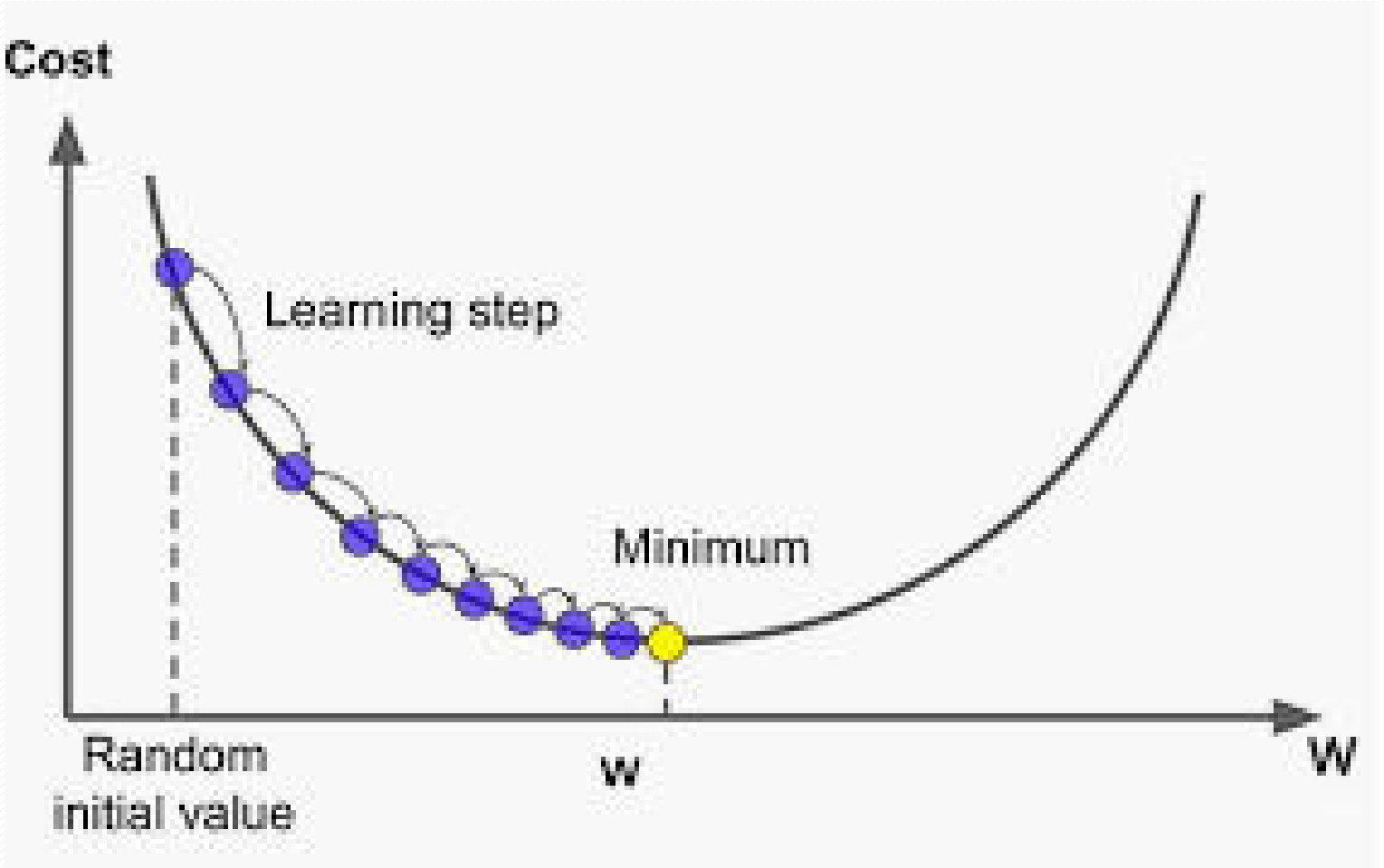


Gradient Descent

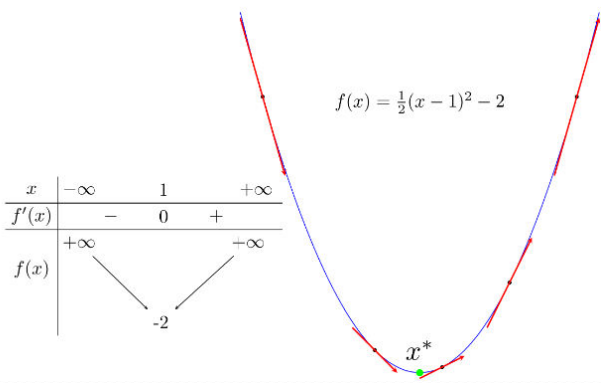
- Trong Machine Learning nói riêng và Toán Tối Ưu nói chung, chúng ta thường xuyên phải tìm giá trị nhỏ nhất (hoặc đôi khi là lớn nhất) của một hàm số nào đó.
  - global minimum rất phức tạp, thậm chí là bất khả thi.
  - local minimum (nghiệm của phương trình đạo hàm bằng 0) -> giải phương trình đạo hàm bằng 0 là bất khả thi (sự phức tạp của dạng của đạo hàm, từ việc các điểm dữ liệu có số chiều lớn, hoặc từ việc có quá nhiều điểm dữ liệu)
- Cần một phương pháp sắp xỉ?
  - xuất phát từ một điểm mà chúng ta coi là *gần* với nghiệm của bài toán, sau đó dùng một phép toán lặp để *tiến dần* đến điểm cần tìm, tức đến khi đạo hàm gần với 0

Gradient Descent

- Ví dụ:



Gradient Descent cho hàm 1 biến



- Giả sử $\mathbf{x_t}$ là điểm ta tìm được sau vòng lặp thứ tt. Ta cần tìm một thuật toán để đưa $\mathbf{x_t}$ về càng gần $\mathbf{x^*}$ càng tốt.
- $f'(x_t) > 0$ thì $\mathbf{x_t}$ nằm về bên phải so với $\mathbf{x^*}$
  - Để điểm tiếp theo $\mathbf{x_{t+1}}$ gần với $\mathbf{x^*}$ hơn, chúng ta cần di chuyển $\mathbf{x_t}$ về phía bên trái, tức về phía *âm*
- $f'(x_t) < 0$ thì $\mathbf{x_t}$ nằm về bên trái so với $\mathbf{x^*}$
  - Để điểm tiếp theo $\mathbf{x_{t+1}}$ gần với $\mathbf{x^*}$ hơn, chúng ta cần di chuyển $\mathbf{x_t}$ về phía bên phải, tức về phía *dương*.
- **Chúng ta cần di chuyển ngược dấu với đạo hàm**:

$$x_{t+1} = x_t + \Delta$$

Trong đó Δ là một đại lượng ngược dấu với đạo hàm $f'(x_t)$

Gradient Descent cho hàm 1 biến

- x_t càng xa x^* về phía bên phải thì $f'(x_t)$ càng lớn hơn 0 (và ngược lại).
- Vậy, lượng di chuyển Δ là tỉ lệ thuận với $-f'(x_t)$
- *Do đó:*

$$x_{t+1} = x_t - \eta f'(x_t)$$

η là một số dương được gọi là *learning rate*

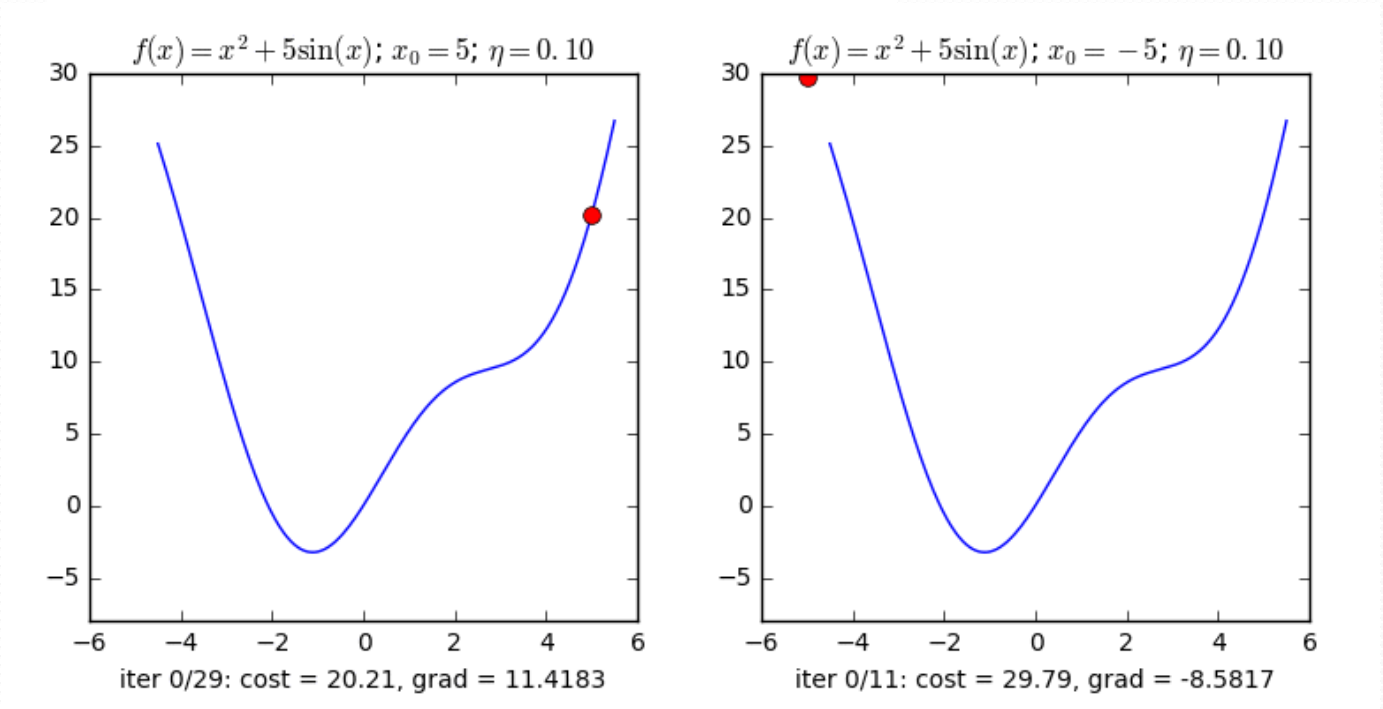
- Thuật toán gradient descent hoạt động dựa trên nhận xét: Nếu hàm số $f(x)$ xác định và khả vi tại điểm $\boldsymbol{x_t}$, khi đó giá trị của f sẽ **giảm nhanh nhất** khi đi theo **hướng ngược với gradient** của f
- Nếu ta chọn điểm xuất phát $\boldsymbol{x_0}$, và sau đó đi theo công thức trên thì ta sẽ đi dần đến điểm cực tiểu (local minimum) của hàm f (cực tiểu ở đây không chắc là giá trị nhỏ nhất của hàm số).

Gradient Descent cho hàm 1 biến

- Ví dụ: $f(x) = x^2 + 5\sin(x)$ với đạo hàm $f'(x) = 2x + 5\cos(x)$
- Giả sử bắt đầu từ một điểm x_0 nào đó, tại vòng lặp thứ t, chúng ta sẽ cập nhật như sau:

$$x_{t+1} = x_t - \eta(2x_t + 5\cos(x_t))$$

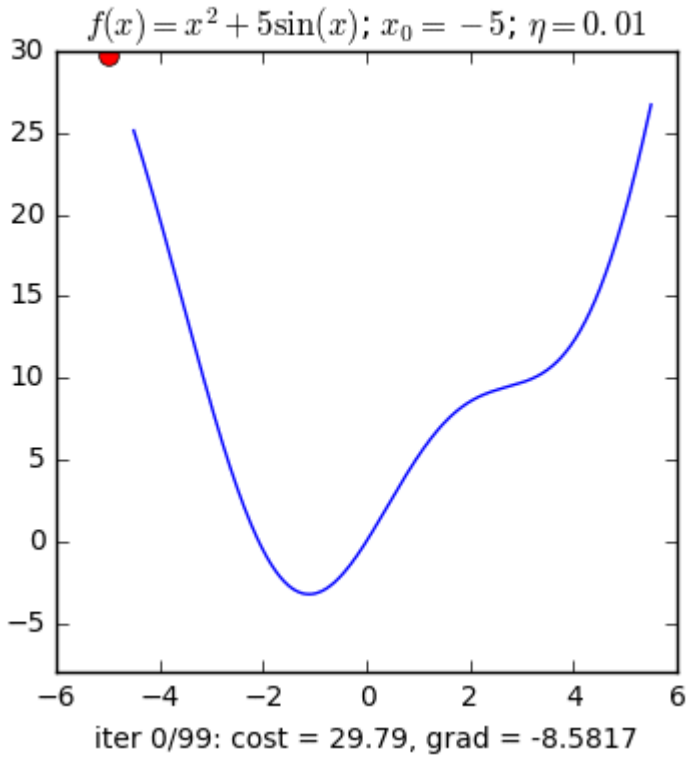
Giá trị khởi tạo khác nhau

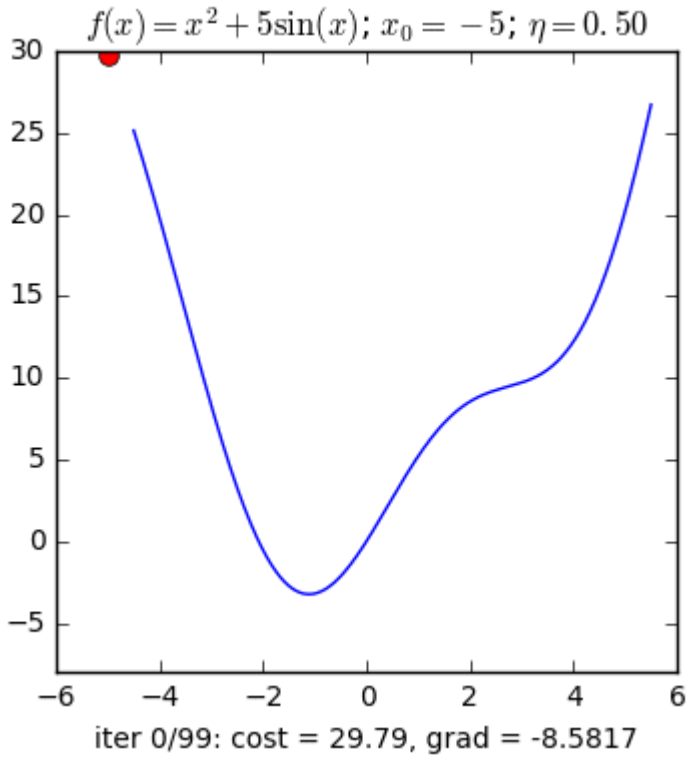


Gradient Descent cho hàm 1 biến

- Ví du:

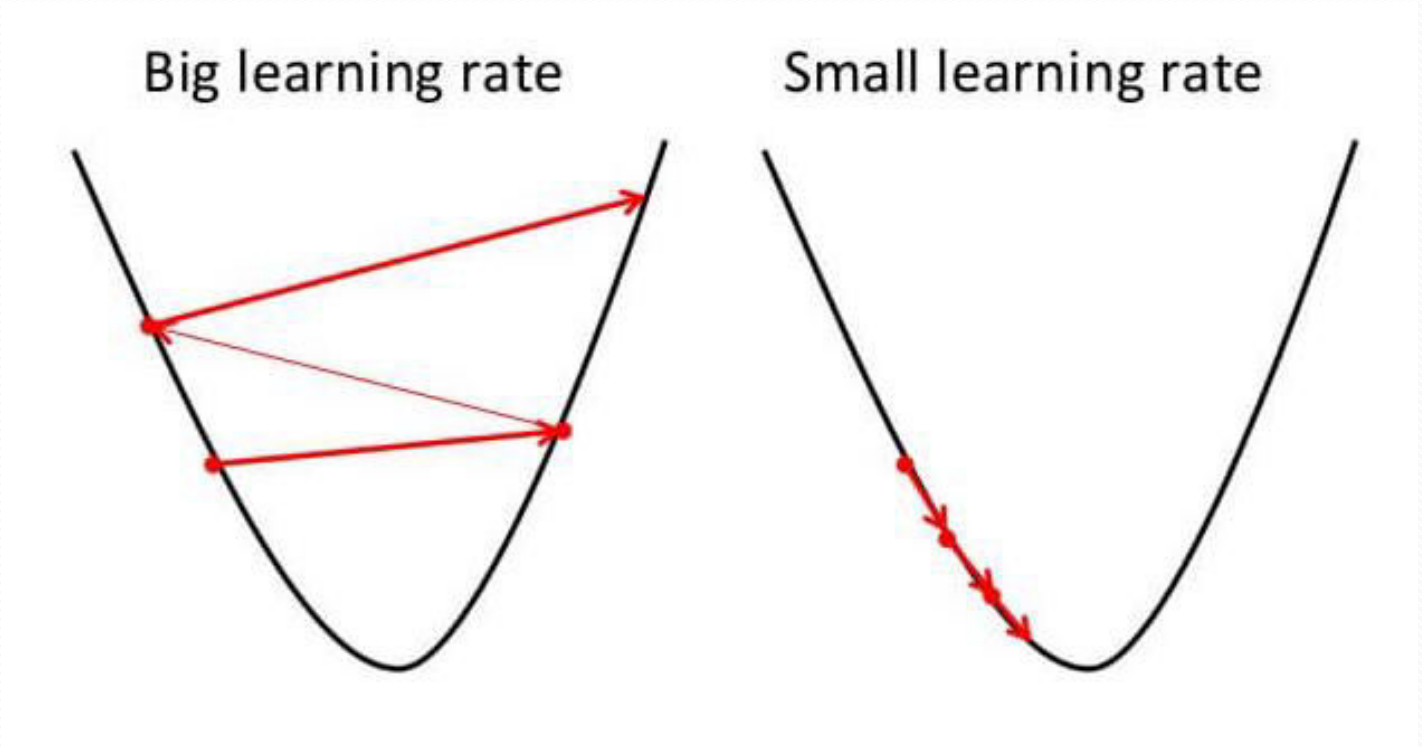
Learning rate khác nhau





Gradient Descent

- Learning rate khác nhau



Gradient Descent

- Tổng quá cho hàm nhiều biến: tìm global minimum cho hàm $f(\theta)$ trong đó θ là một vector.
- Thuật toán Gradient Descent

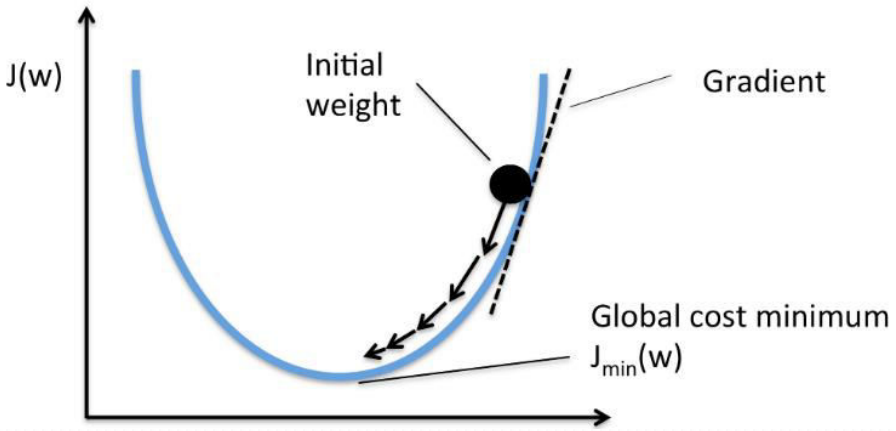
Khởi tạo ngẫu nhiên θ_0.

Lặp (cho đến khi hội tụ hoặc số lượng vòng lặp vượt quá một ngưỡng)

{

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta f(\theta_t)$$

}



Gradient Descent

- Gradient Descent cho bài toán Linear Regression.
- Loss function (hàm mất mát) của Linear Regression là:

$$J(\mathbf{w}) = \frac{1}{2N}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2$$

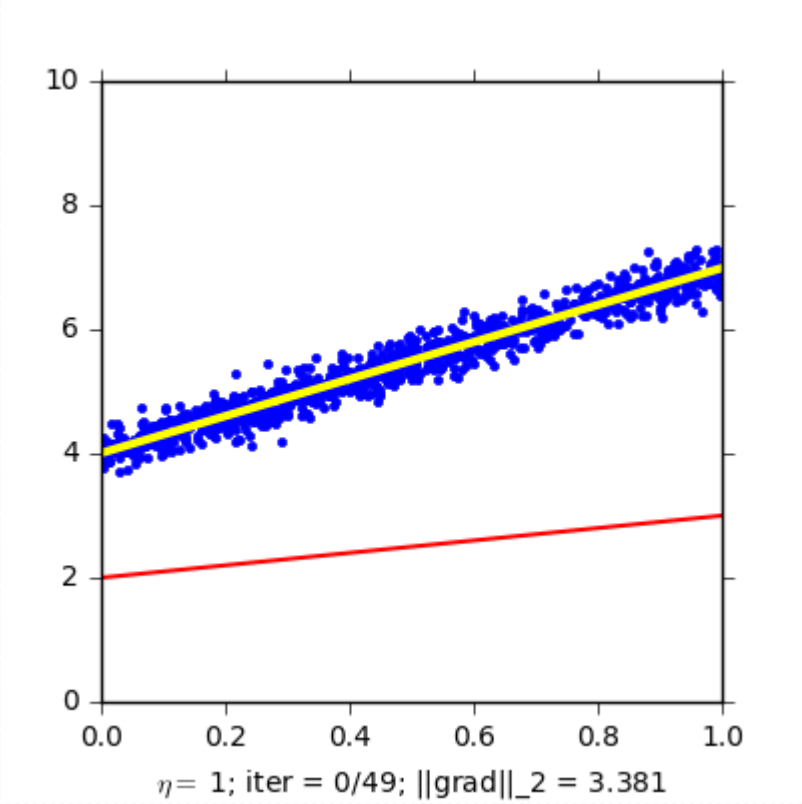
$$= \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{x}_i\mathbf{w} - y_i)^2$$

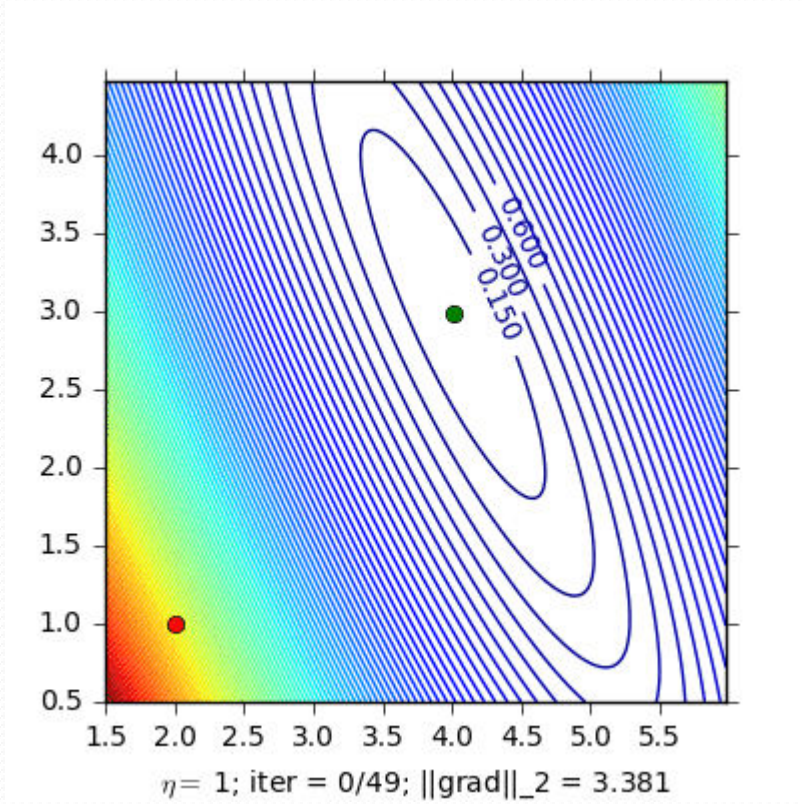
- Đạo hàm của hàm mất mát là:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i^T(\mathbf{x}_i\mathbf{w} - y_i)$$

Gradient Descent

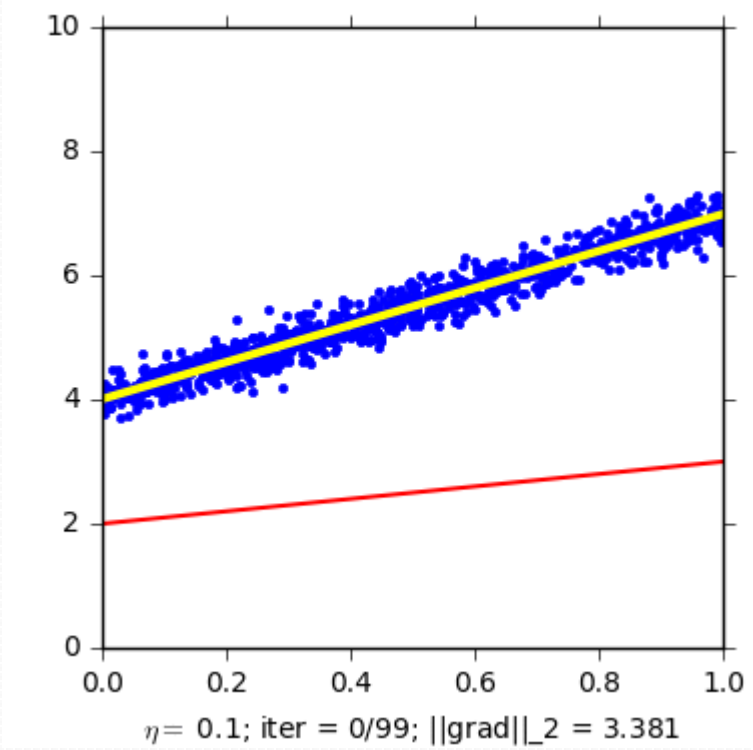
- Gradient Descent cho bài toán Linear Regression.
- Ví dụ:

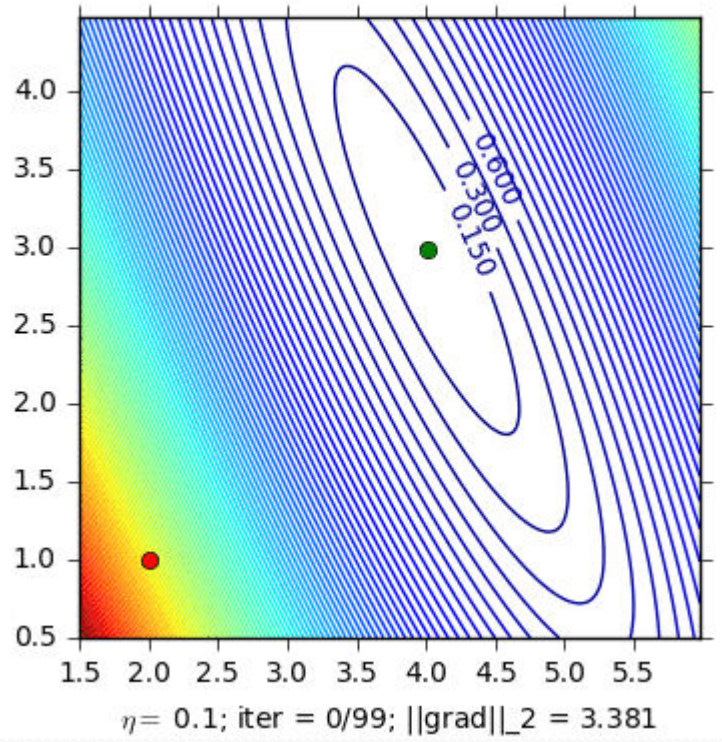




Gradient Descent

- Gradient Descent cho bài toán Linear Regression.
- Ví dụ: learning rate nhỏ



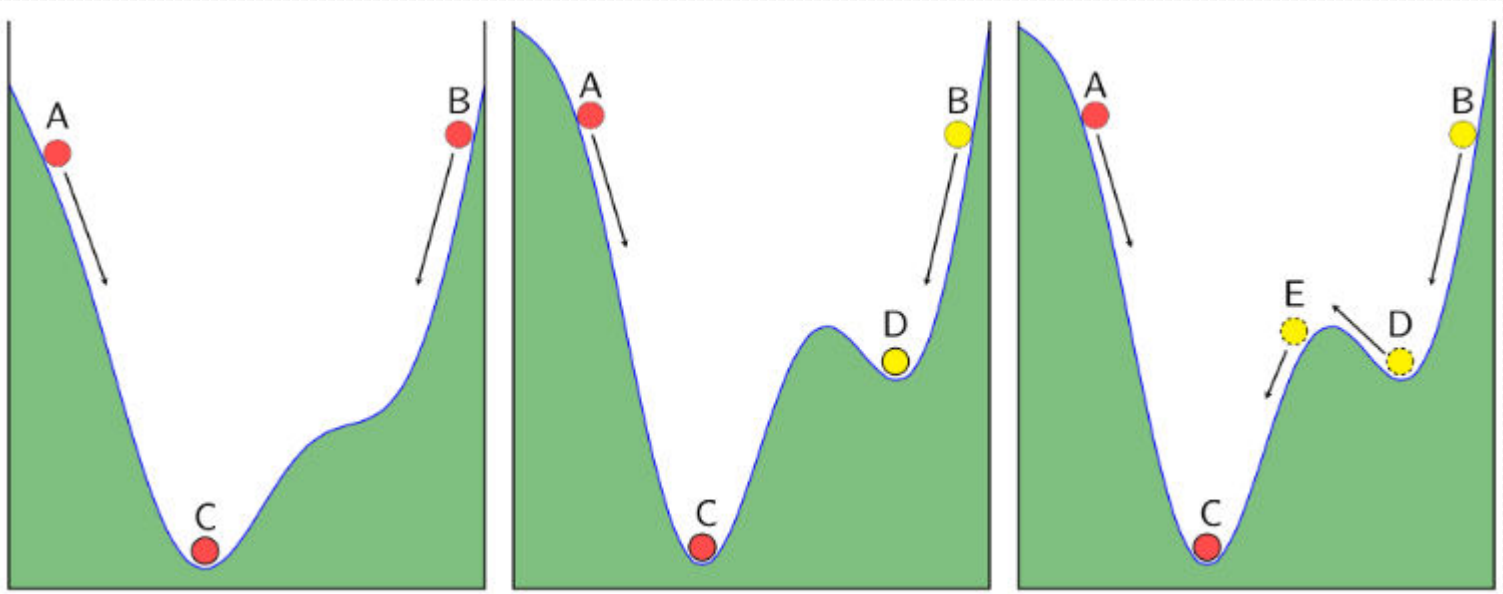


Các cải tiến của Gradient Descent

- Các cải tiến của Gradient Descent
  - Momentum
  - Batch Gradient Descent
  - Stochastic Gradient Descent.
  - Mini-batch Gradient Descent

Các cải tiến của Gradient Descent

- Momentum



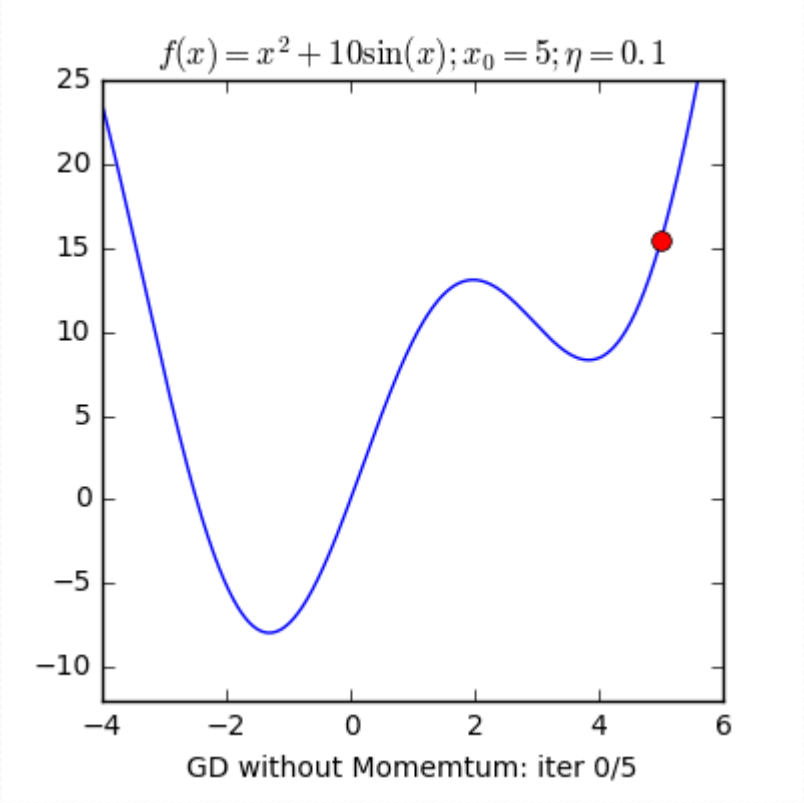
vận tốc $\mathbf{v_t}$

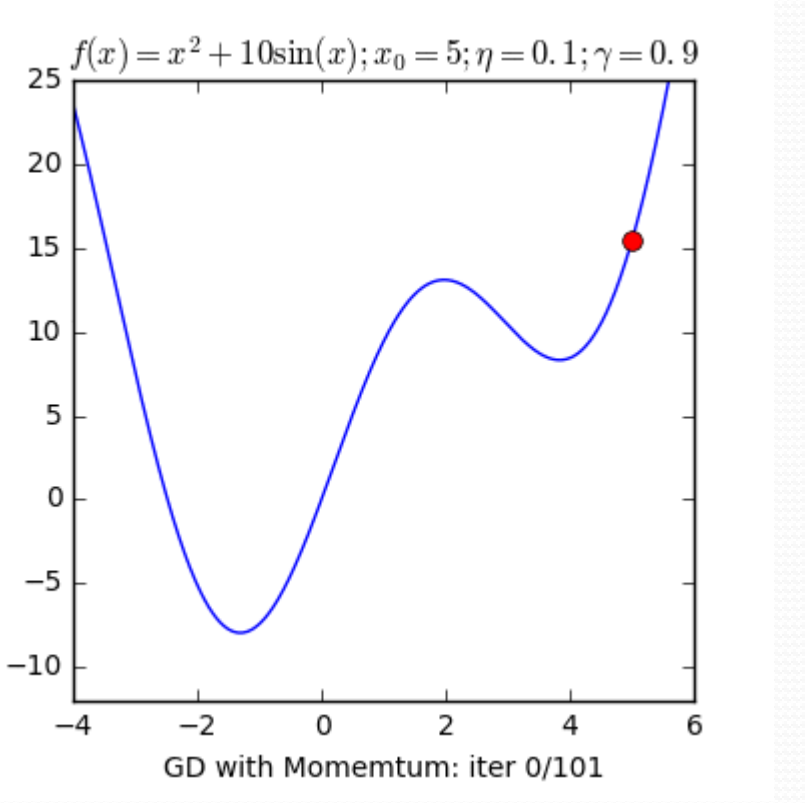
$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$$\theta_{t+1} = \theta_t - v_t$$

Các cải tiến của Gradient Descent

- Momentum





Các cải tiến của Gradient Descent

- Batch Gradient Descent:
  - Sử dụng tất cả các điểm dữ liệu huấn luyện (x_i) để cập nhật và tính đạo hàm
  - Ví dụ: tính đạo hàm của Linear Regression

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i^T (\mathbf{x}_i \mathbf{w} - y_i)$$

  - Hạn chế khi dữ liệu lớn.
  - Không thể dùng trong online learning

Các cải tiến của Gradient Descent

- Stochastic Gradient Descent
- Tại 1 thời điểm, ta chỉ tính đạo hàm của loss function dựa trên *chỉ một* điểm dữ liệu $\mathbf{x}_i$ rồi cập nhật tham số $\boldsymbol{\theta}$ ***(hoặc w trong linear regression)*** dựa trên đạo hàm này.

$$\theta = \theta - \eta \nabla_\theta J(\theta; \mathbf{x}_i; \mathbf{y}_i)$$

- Epoch vs. Iteration
  - Iteration: có N mẫu thì có N lần lặp để cập nhật $\boldsymbol{\theta}$.
  - *Epoch:* một epoch ứng với N lần cập nhật $\boldsymbol{\theta}$.

- Phù hợp cho online learning (số epoch ko quá nhiều)
- Các mẫu nên được lựa chọn ngẫu nhiên khi cập nhật $\boldsymbol{\theta}$ trong mỗi epoch

Các cải tiến của Gradient Descent

- Stochastic Gradient Descent cho Linear Regression

$$\nabla_{\mathbf{w}} J(\mathbf{w}; \mathbf{x}_i; y_i) = \mathbf{x}_i^T (\mathbf{x}_i \mathbf{w} - y_i)$$

- Batch Gradient Descent cho Linear Regression

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i^T (\mathbf{x}_i \mathbf{w} - y_i)$$

Các cải tiến của Gradient Descent

- Mini-batch Gradient Descent
  - mini-batch sử dụng một số lượng n lớn hơn 1 (nhưng vẫn nhỏ hơn tổng số dữ liệu N rất nhiều)

$$\theta = \theta - \eta \nabla_\theta J(\theta; \mathbf{x}_{i:i+n}; \mathbf{y}_{i:i+n})$$

Các cải tiến của Gradient Descent

- Stochastic Gradient Descent

$$\theta = \theta - \eta \nabla_\theta J(\theta; \mathbf{x}_i; \mathbf{y}_i)$$

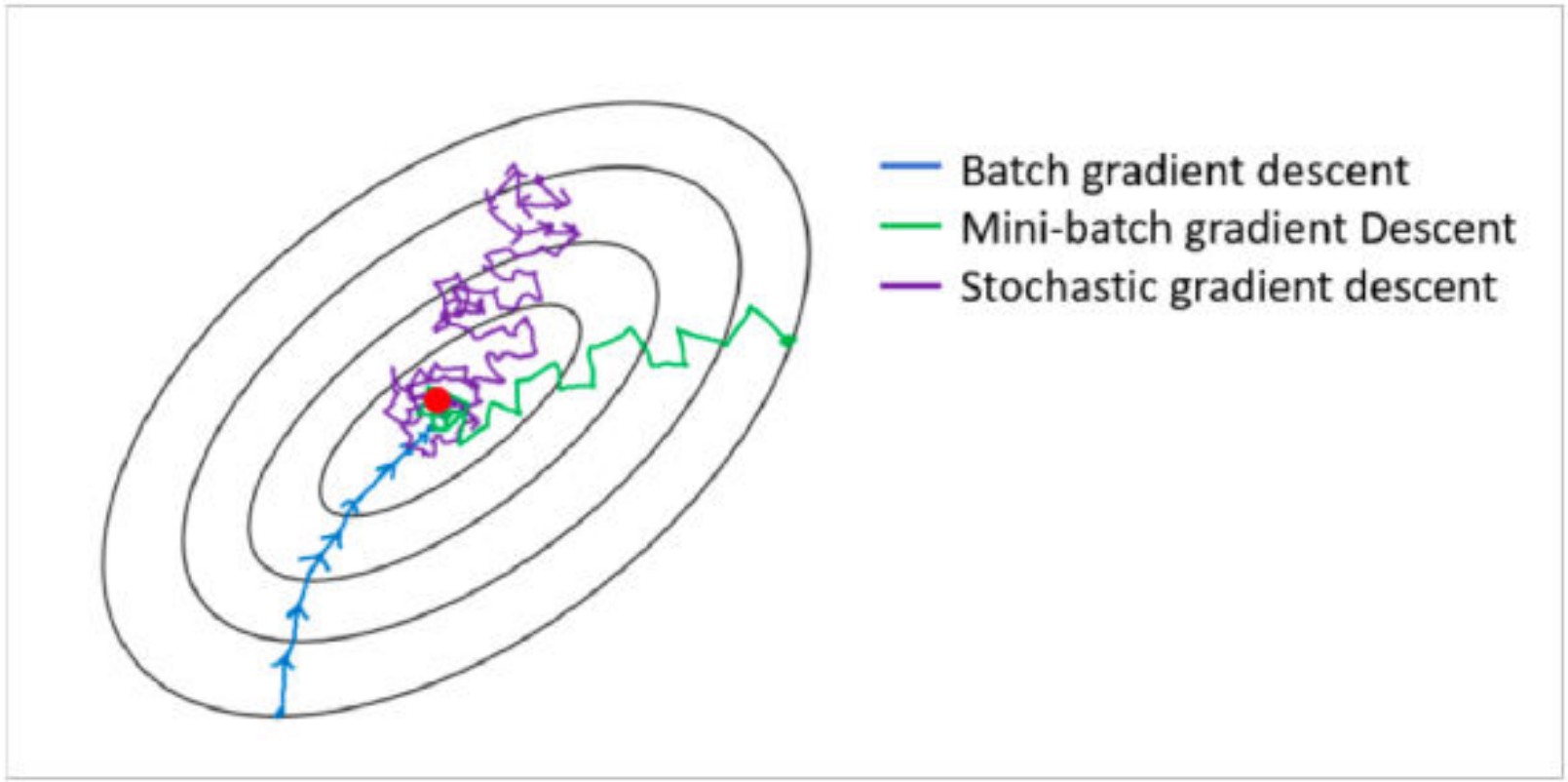
- Batch Gradient Descent

$$\theta = \theta - \eta \nabla_\theta J(\theta; \mathbf{x}; \mathbf{y})$$

- Mini-batch Gradient Descent

$$\theta = \theta - \eta \nabla_\theta J(\theta; \mathbf{x}_{i:i+n}; \mathbf{y}_{i:i+n})$$

Các cải tiến của Gradient Descent



Gradient descent variants' trajectory towards minimum

Trade-off

Depending on the amount of data, they make a trade-off:

- The **accuracy** of the parameter update
- The **time** it takes to perform an update.
- The **memory** it takes to perform an update.
- Capability of **online learning**.

Method	Accuracy	Time	Memory Usage	Online Learning
Batch gradient descent	○	Slow	High	×
Stochastic gradient descent	△	High	Low	○
Mini-batch gradient descent	○	Medium	Medium	○

Tìm hiểu thêm

- Newton's method tìm nghiệm tối ưu
  - second-order method
  - Hessian matrix **H**

Áp dụng phương pháp này cho việc giải phương trình $f'(x) = 0$ ta có:

$$x_{t+1} = x_t - (f''(x_t))^{-1} f'(x_t)$$

Và trong không gian nhiều chiều với θ là biến:

$$\theta = \theta - \mathbf{H}(J(\theta))^{-1} \nabla_\theta J(\theta)$$

Bài Tập

1) Toy example: bảng dữ liệu về chiều cao và cân nặng của 15 người như dưới đây. Có thể dự đoán cân nặng của một người dựa vào chiều cao của họ không?

- Cài đặt chương trình demo bằng python (có thể dùng thư viên scikit-learn)

Chiều cao (cm)	Cân nặng (kg)	Chiều cao (cm)	Cân nặng (kg)
147	49	168	60
150	50	170	72
153	51	173	63
155	52	175	64
158	54	178	66
160	56	180	67
163	58	183	68
165	59		

Linear Regression

- Toy example

$$y \approx f(\mathbf{x}) = \hat{y}$$

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0$$

$$\mathbf{w} = [w_0, w_1, w_2, w_3]^T$$

Matrix form

$$\bar{\mathbf{x}} = [1, x_1, x_2, x_3]$$

$$y \approx \bar{\mathbf{x}}\mathbf{w} = \hat{y}$$

Điểm tối ưu của bài toán Linear Regression có dạng $\mathbf{w} = \mathbf{A}^{\dagger}\mathbf{b} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{\dagger}\bar{\mathbf{X}}^T\mathbf{y}$

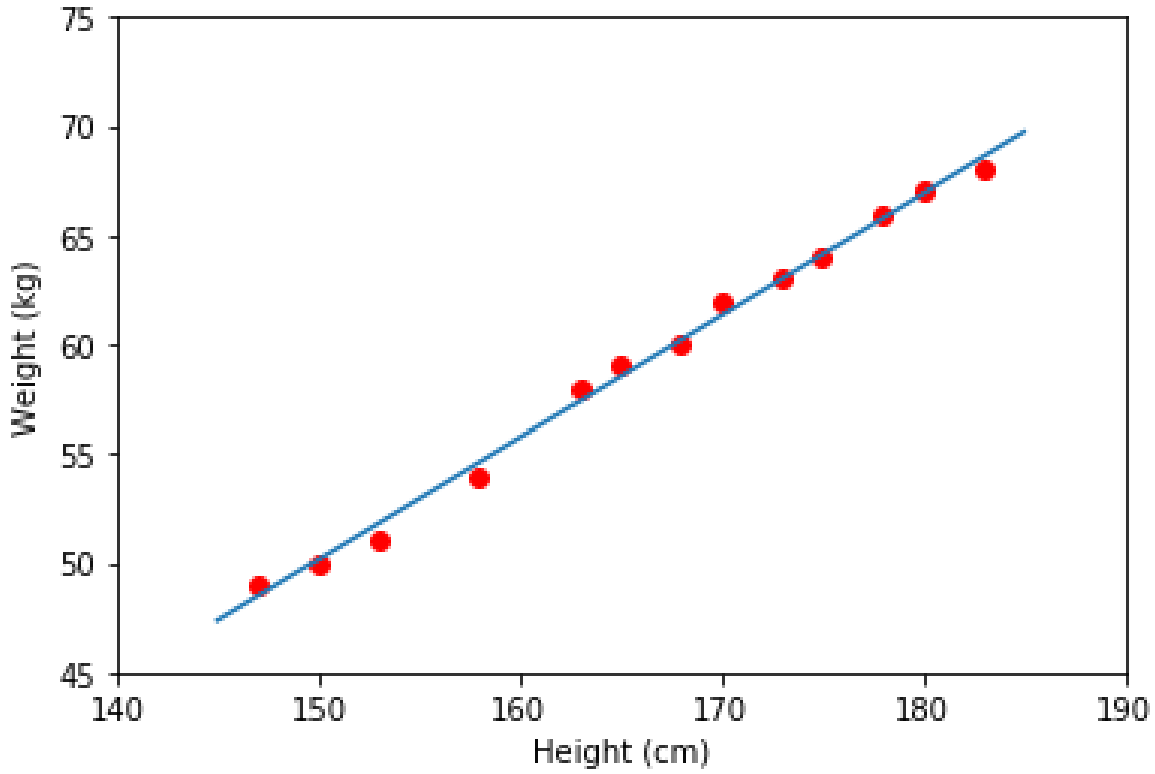
Linear Regression

- Toy example

mô hình Linear Regression
(cân nặng) = w_1*(chiều cao) + w_0)

w_0 = -33.7354
w_1 = 0.5592

```
w =  [[-33.73541021]
 [   0.55920496]]
```



Bài Tập

2) Dự đoán giá bất động sản : **Boston Housing Dataset**

- This data was originally a part of UCI Machine Learning Repository and has been removed now.
- This data also ships with the **scikit-learn library**. There are **506 samples** and **13 feature variables in this data-set**.
- The objective is to **predict the value of prices of the house** using the given features.

- Tham khảo:

https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155

- Dự đoán giá bất động sản : **Boston Housing Dataset**
- **Information**
  - *data*: contains the information for various houses
  - *target*: prices of the house
  - *feature_names*: names of the features
  - *DESCR*: describes the dataset

```
from sklearn.datasets import load_boston

boston_dataset = load_boston()
```

Boston Housing Dataset

CRIM: Per capita crime rate by town
ZN: Proportion of residential land zoned for lots over 25,000 sq. ft
INDUS: Proportion of non-retail business acres per town
CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX: Nitric oxide concentration (parts per 10 million)
RM: Average number of rooms per dwelling
AGE: Proportion of owner-occupied units built prior to 1940
DIS: Weighted distances to five Boston employment centers
RAD: Index of accessibility to radial highways
TAX: Full-value property tax rate per $10,000
PTRATIO: Pupil-teacher ratio by town
B: $1000(Bk - 0.63)^2$, where Bk is the proportion of [people of African American descent] by town
LSTAT: Percentage of lower status of the population
MEDV: Median value of owner-occupied homes in $1000s

Boston Housing Dataset

- The prices of the house indicated by the variable MEDV is our ***target variable*** and the remaining are the ***feature variables*** based on which we will predict the value of a house.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33