GRADIENT DESCENT ALGORITHMS

Classification and Prediction

Lê Hồng Phương

<*phuonglh@gmail.com*>
Vietnam National University of Hanoi
Hanoi University of Science

February 2015

Nội dung

Giới thiệu

- Trong ước lượng thống kê và học tự động, ta cần cực tiểu hoá một hàm mục tiêu có dạng một tổng các hàm thành phần như sau:

$$L(\theta) = \sum_{i=1}^{N} L_i(\theta),$$

  trong đó θ là tham số cần ước lượng.
- Nói chung mỗi số hạng $L_i(\theta)$ được gắn với quan sát thứ i của tập dữ liệu huấn luyện.

Giới thiệu

Các bài toán cực tiểu hoá dạng này xuất hiện trong

- Phương pháp bình phương tối thiểu
- Ước lượng hợp lí cực đại khi các quan sát là độc lập
- Cực tiểu hoá hàm lỗi trên tập dữ liệu huấn luyện. Khi đó $L_i(\theta)$ là hàm lỗi ứng với quan sát thứ i.

Ví dụ: phương pháp bình phương tối thiểu trong hồi quy tuyến tính:

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{N}(h_\theta(\mathbf{x}_i) - y_i)^2$$
$$= \frac{1}{2}\sum_{i=1}^{N}(\widehat{y}_i - y_i)^2.$$

Giới thiệu

Phương pháp giảm gradient là một phương pháp tối ưu tổng quát để giải bài toán

$$L(\theta) \to \min$$

nhằm ước lượng tham số θ của mô hình, khi hàm $L(\theta)$ khả vi.

Phép lấy vi phân và đạo hàm

Giả sử $y = f(x)$. Phép lấy vi phân là một phương pháp tính tốc độ thay đổi của y ứng với tốc độ thay đổi của x.

- Nếu $x, y \in \mathbb{R}$ và ta vẽ đồ thị $f(x)$ thì đạo hàm $f'(x)$ đo *độ dốc* của đồ thị tại mỗi điểm (x, y).
- Nếu $y = ax + b$ với $a, b \in \mathbb{R}$ thì $f'(x) = a$. Độ dốc a là

$$a = \frac{\text{độ thay đổi của } y}{\text{độ thay đổi của } x} = \frac{\Delta y}{\Delta x},$$

vì

$$\begin{aligned} y + \Delta y &= f(x + \Delta x) = a(x + \Delta x) + b \\ &= ax + b + a\Delta x \\ &= y + a\Delta x. \end{aligned}$$

Phép lấy vi phân và đạo hàm

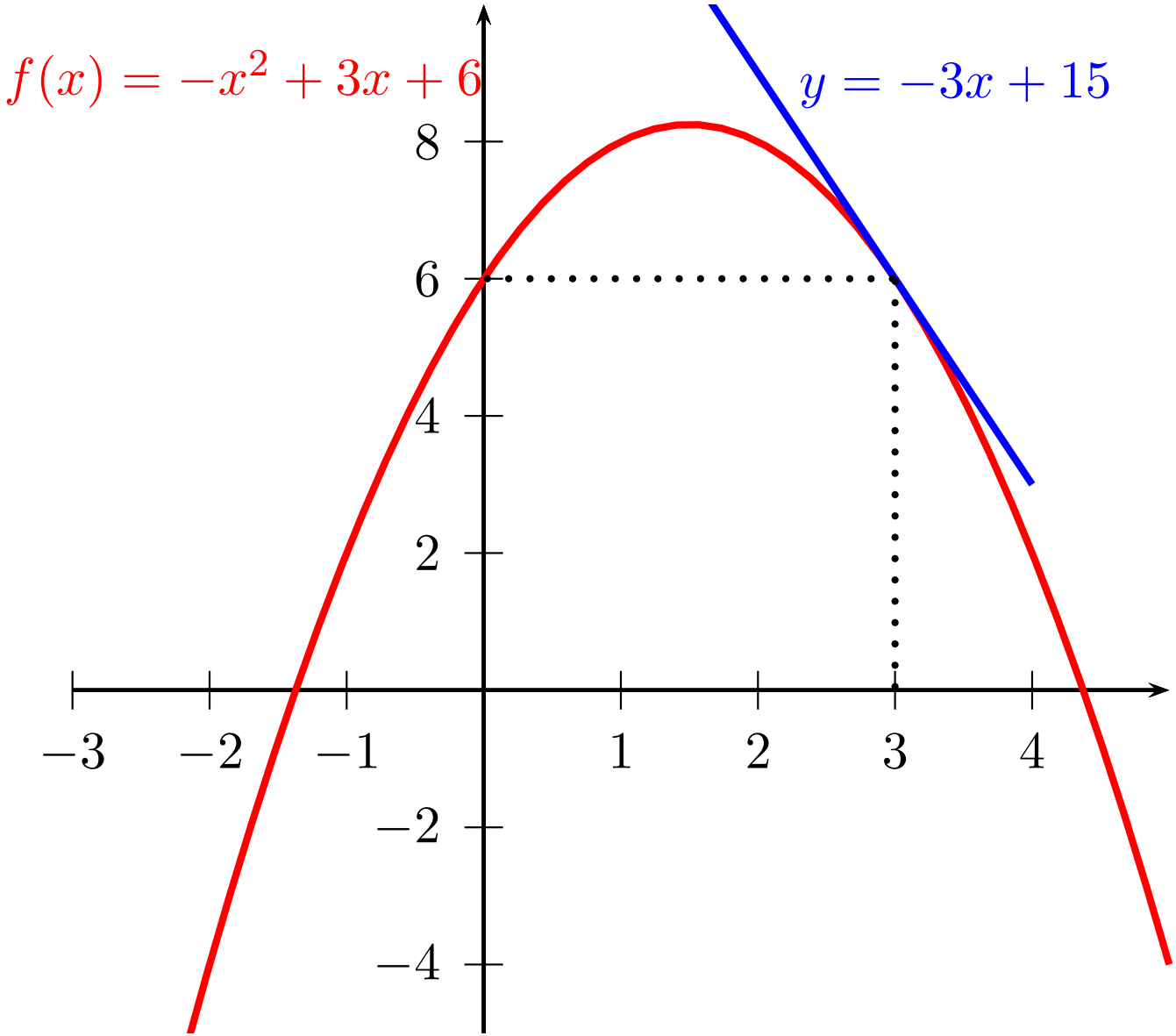
Nếu $f(x)$ không phải là hàm tuyến tính theo x thì $\Delta y/\Delta x$ không phải là hằng số:

$$\frac{\Delta y}{\Delta x} = \frac{dy}{dx} = f'(x).$$

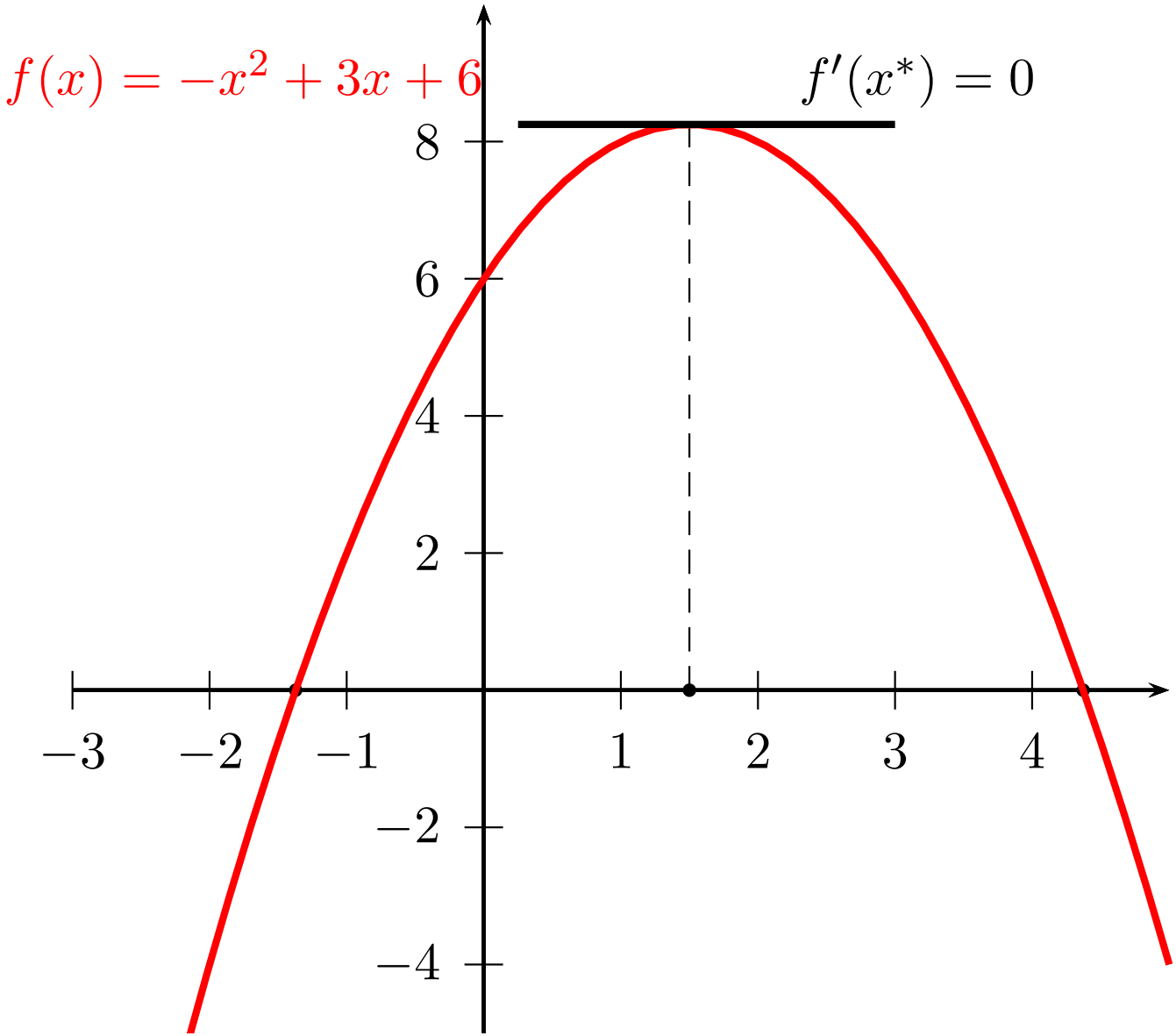
Ví dụ, tại mỗi điểm, đạo hàm của $f(x) = -x^2 + 3x + 6$ là độ dốc của đường thẳng tiếp tuyến với đồ thị $f(x)$.

$$f'(x) = -2x + 3 \Rightarrow f'(3) = -3.$$

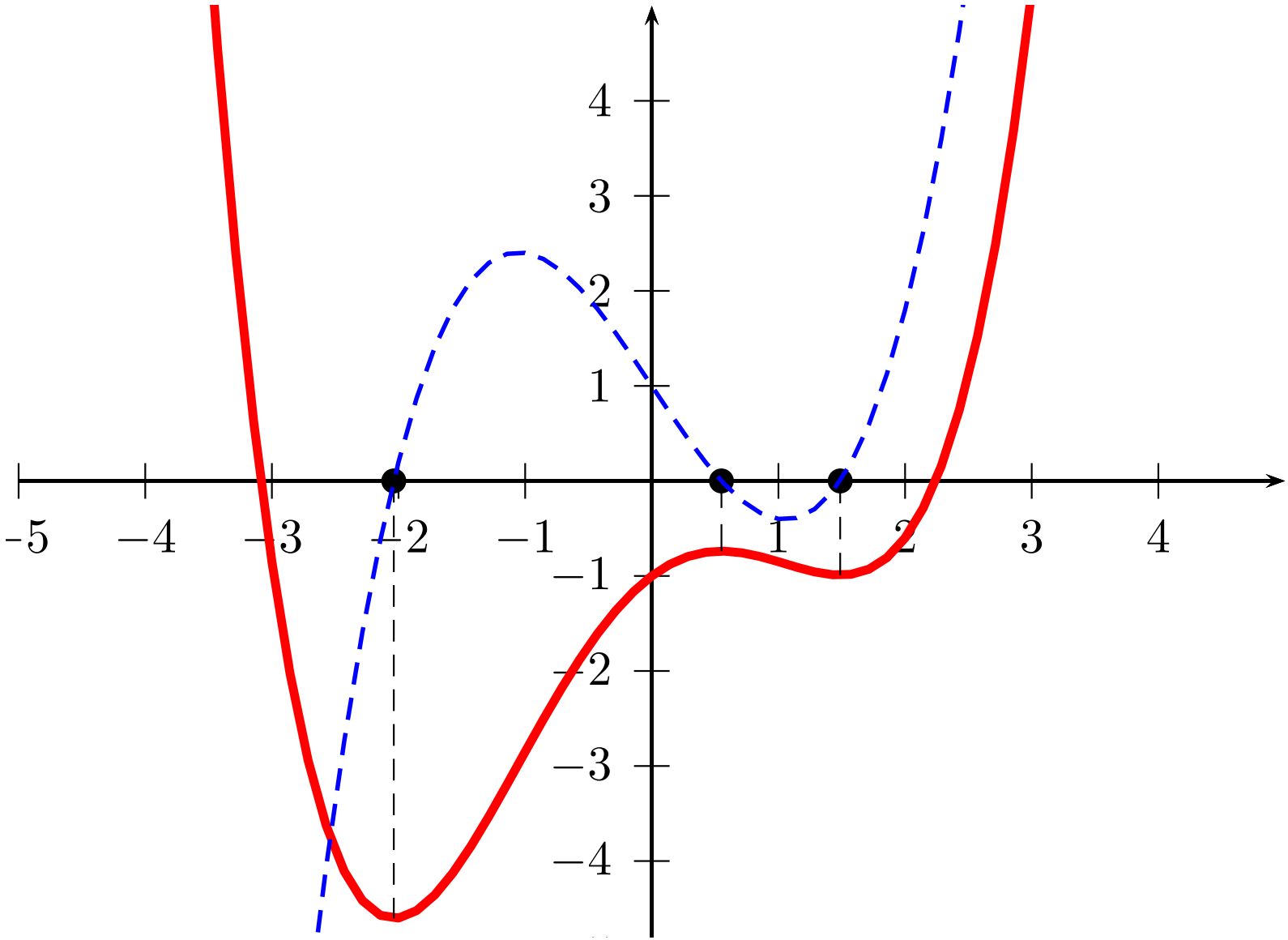
Phép lấy vi phân và đạo hàm



Phép lấy vi phân và đạo hàm



Phép lấy vi phân và đạo hàm

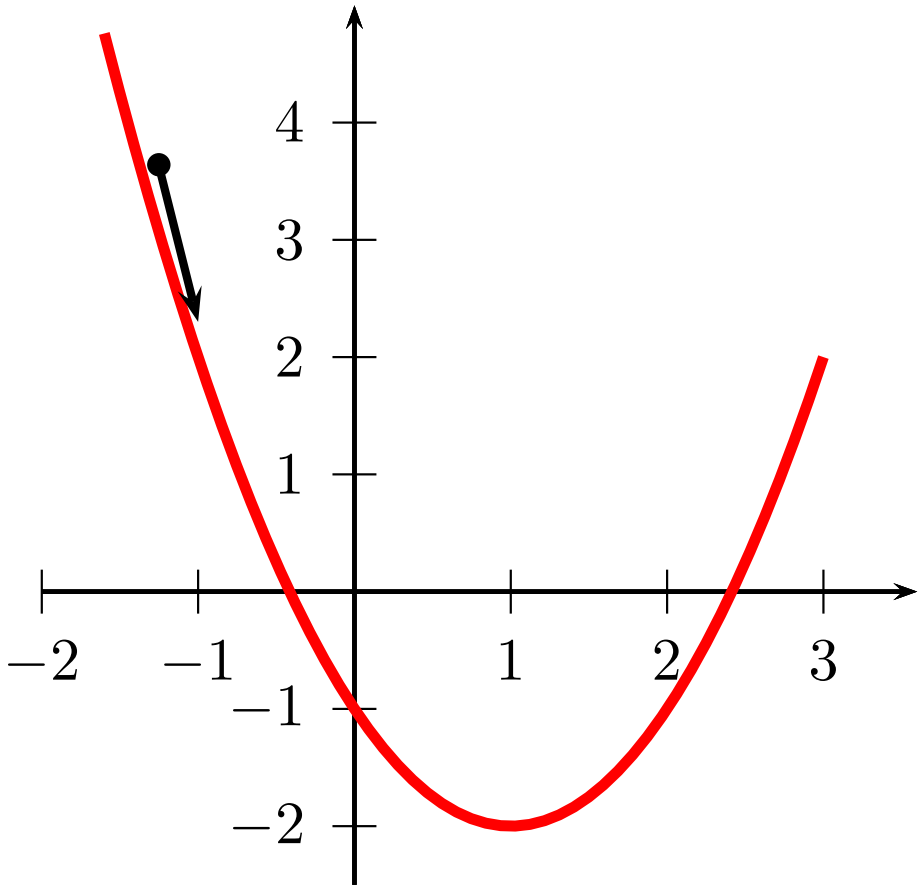


Phương pháp giảm gradient

Phương pháp giảm gradient:

- Là thuật toán tối ưu bậc 1 để tìm giá trị nhỏ nhất (cục bộ) của một hàm khả vi $f(\mathbf{x})$.
- Tại mỗi $\mathbf{x}$, $f(\mathbf{x})$ giảm nhanh nhất nếu ta đi từ $\mathbf{x}$ theo hướng âm của gradient của $f(\mathbf{x})$, tức $-\nabla f(\mathbf{x})$.
- Nếu $\mathbf{x}' = \mathbf{x} - \alpha \nabla f(\mathbf{x})$, với $\alpha > 0$ đủ bé thì $f(\mathbf{x}) \geq f(\mathbf{x}')$.

Phương pháp giảm gradient



Phương pháp giảm gradient

Từ đó, để tìm cực tiểu của hàm $f(\mathbf{x})$, ta dự đoán giá trị cực tiểu $\mathbf{x}_0$ ban đầu và xét chuỗi $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots$ sao cho

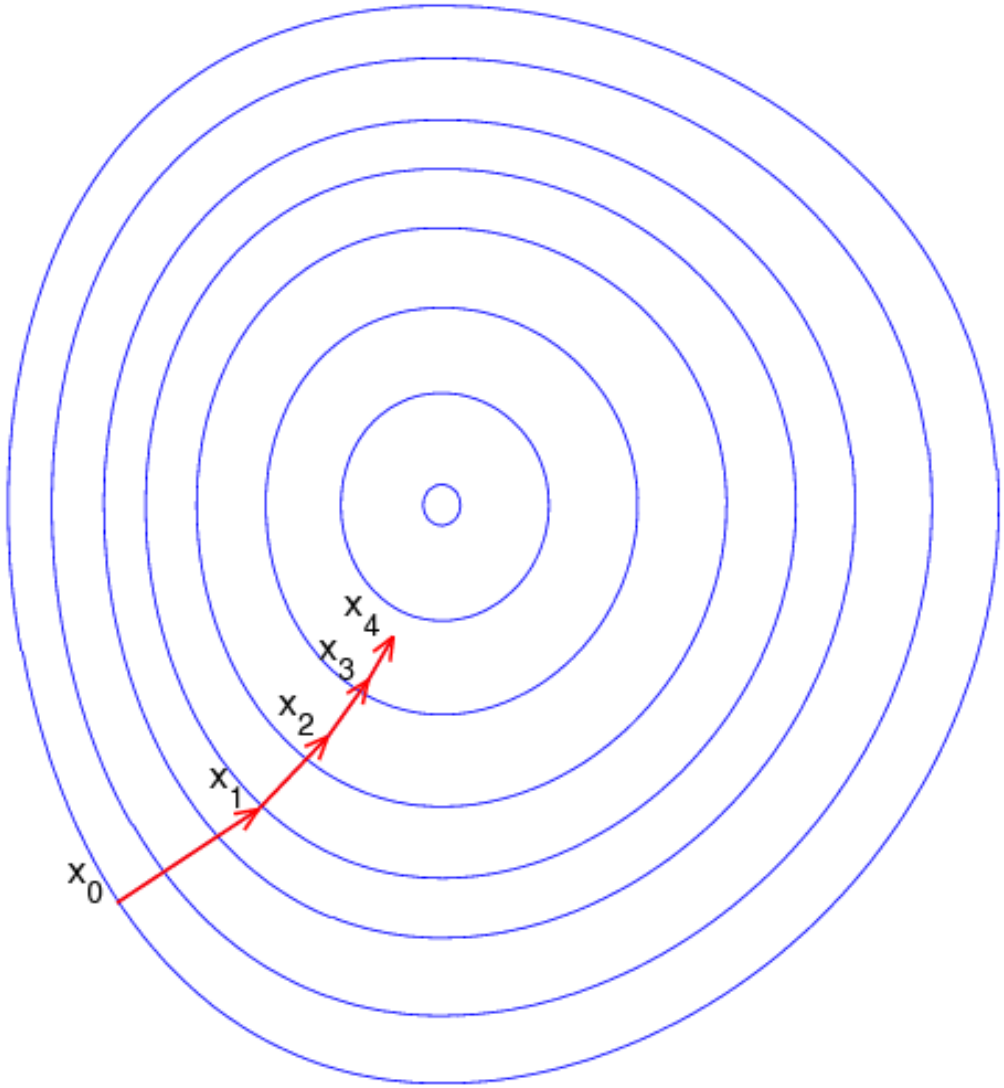
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i), \quad i \geq 0.$$

Ta có

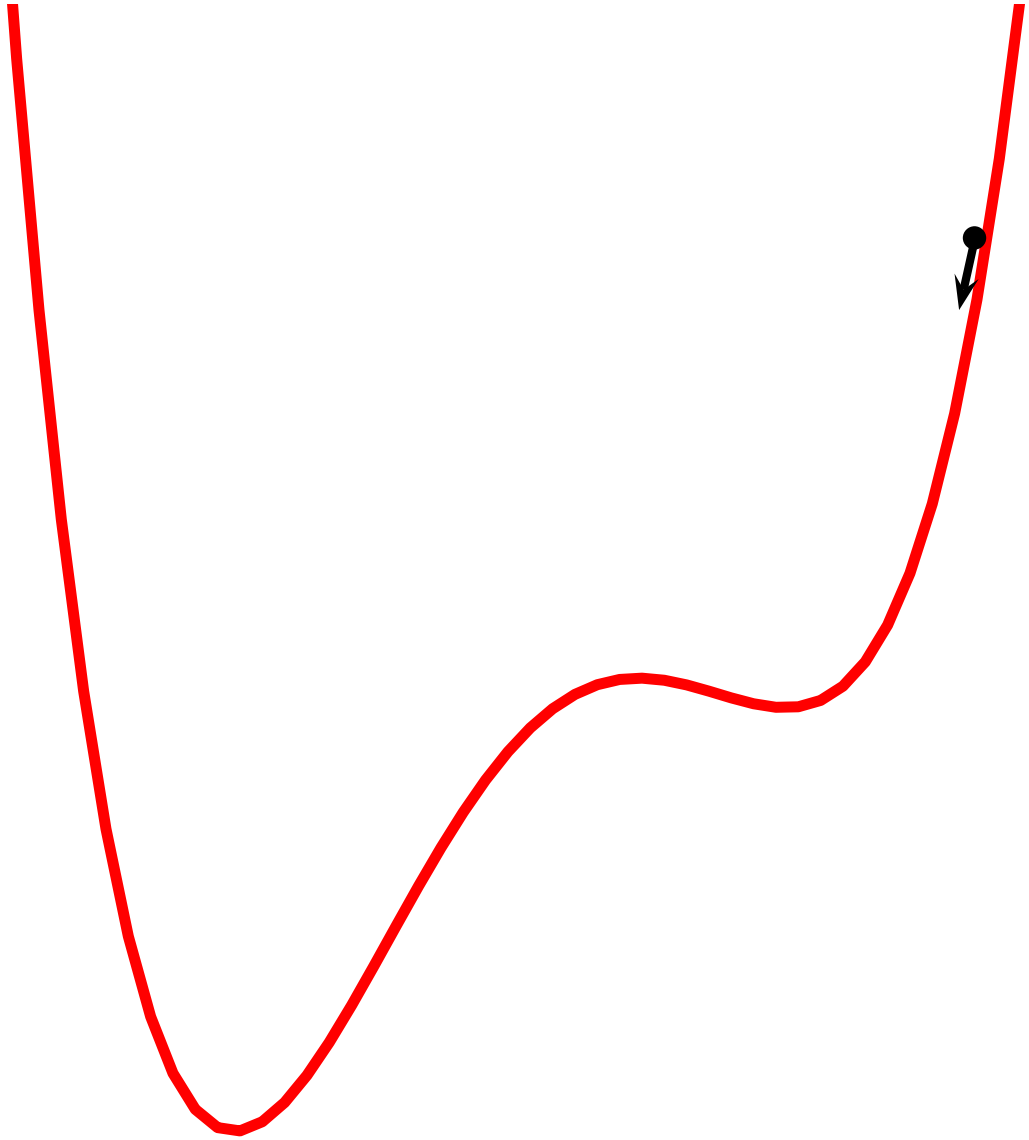
$$f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq \cdots \geq f(\mathbf{x}_i).$$

Chuỗi $\{\mathbf{x}_i\}$ sẽ hội tụ về giá trị cực tiểu.

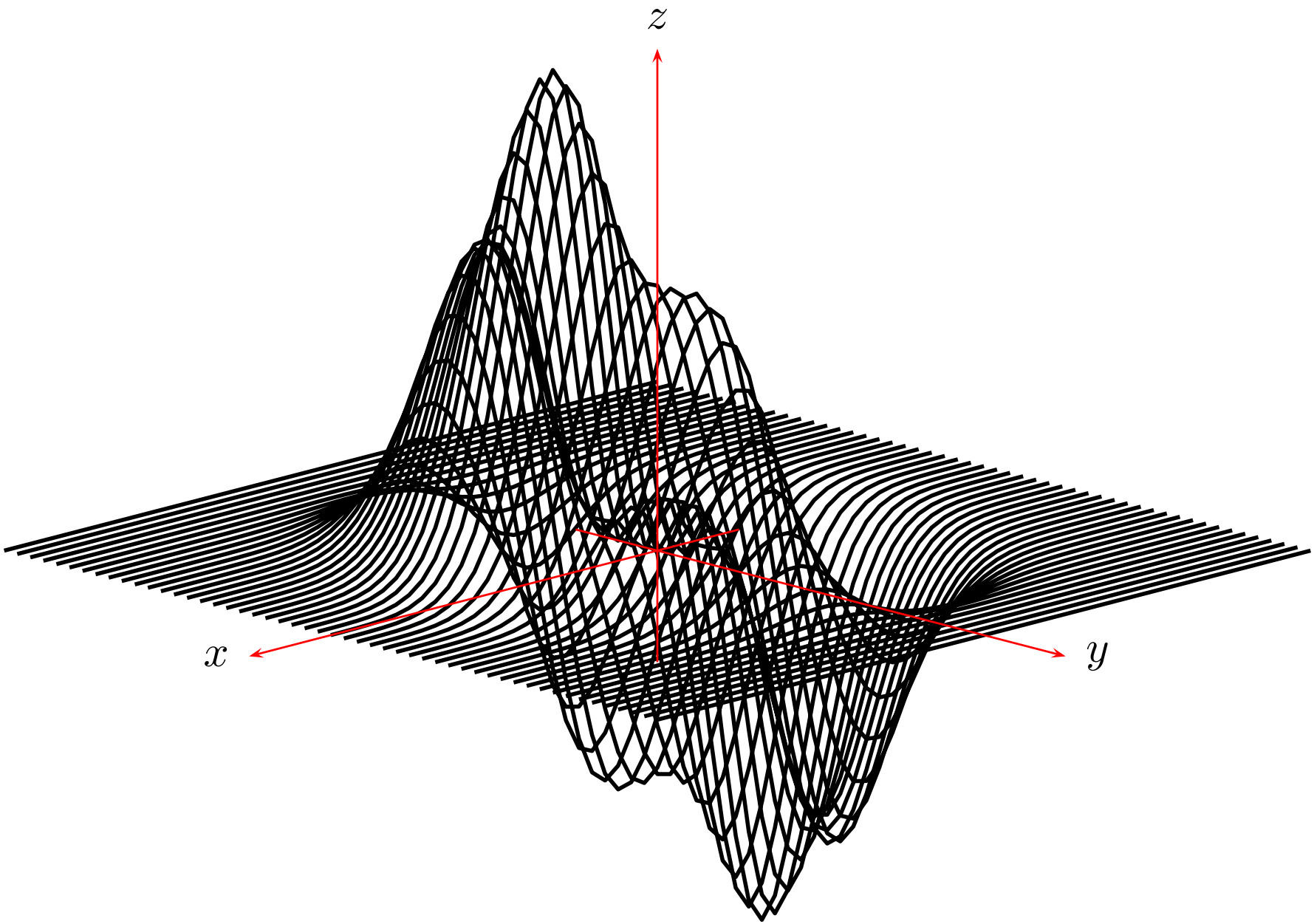
Phương pháp giảm gradient



Tối ưu cục bộ

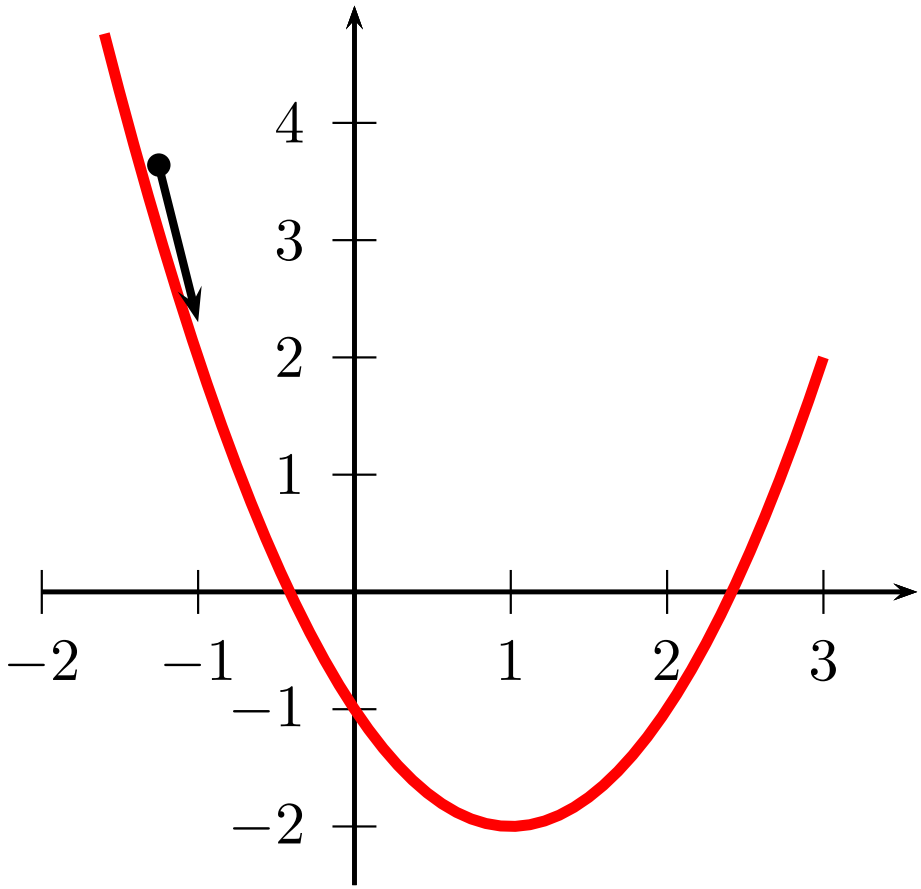


Tối ưu cục bộ

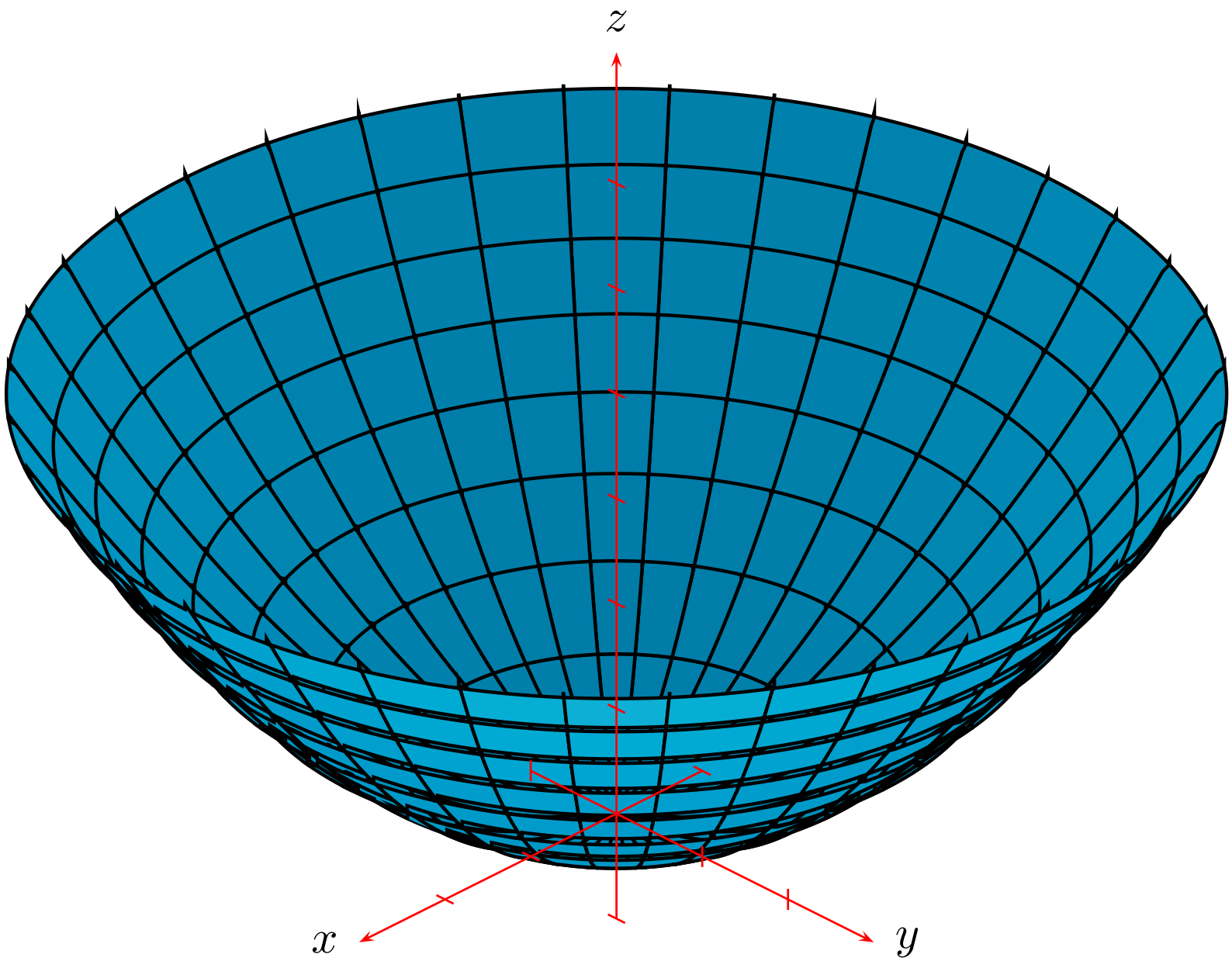


Hàm lồi

Nếu $f(\mathbf{x})$ là hàm lồi (convex) thì phương pháp giảm gradient luôn đảm bảo tìm được tối ưu toàn bộ.

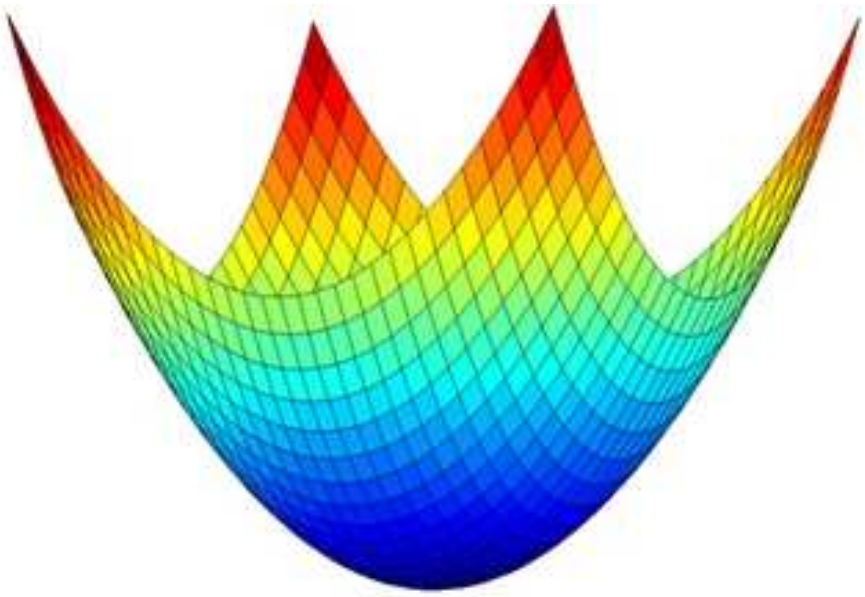


Hàm lồi



Hàm lồi

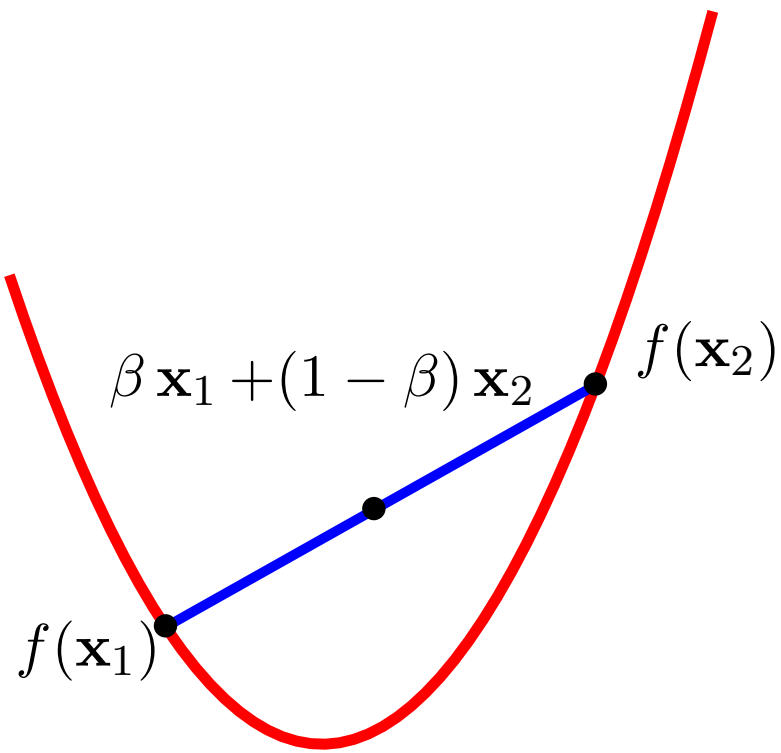
Hàm lồi có đồ thị luôn dốc về phía điểm cực tiểu:



Hàm lồi

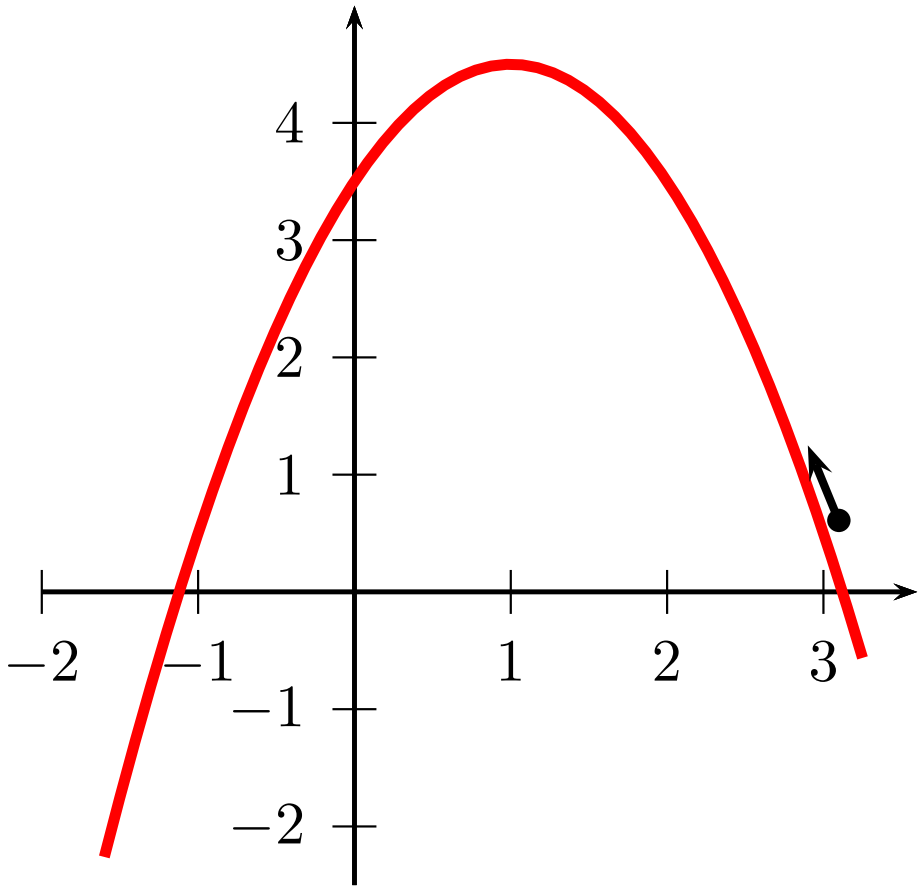
Hàm $f : \mathbb{R}^n \to \mathbb{R}$ được gọi là **hàm lồi** nếu $\forall\, \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n, \forall \beta \in [0, 1]$:

$$f(\beta\, \mathbf{x}_1 + (1 - \beta)\, \mathbf{x}_2) \leq \beta f(\mathbf{x}_1) + (1 - \beta) f(\mathbf{x}_2).$$



Hàm lõm

Hàm f được gọi là **hàm lõm** (concave) nếu $-f$ là hàm lồi. Phương pháp tăng gradient tìm giá trị cực đại:



Thuật toán giảm gradient theo loạt

Problem

$L(\theta) = \sum_{i=1}^{N} L_i(\theta) \to \min.$

Algorithm 1: Batch Gradient Descent

Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

Result: θ

$\theta \leftarrow \vec{0}$;

repeat

$\quad \theta \leftarrow \theta - \alpha \sum_{i=1}^{N} \nabla L_i(\theta)$;

until *converged*;

Thuật toán trên được gọi là thuật toán **giảm gradient theo loạt**. Hằng số $\alpha > 0$ được gọi là **tốc độ học**.

Thuật toán giảm gradient theo loạt

Một số nhận xét:

- Có thể coi mỗi hàm thành phần $L_i(\theta)$ ứng với một mẫu dữ liệu trong bài toán học có giám sát.
- Trong mỗi bước, thuật toán cần tính gradient của các hàm thành phần $L_i(\theta)$ trên toàn bộ tập dữ liệu.
- Nếu các hàm thành phần $L_i(\theta)$ là phức tạp thì việc tính các gradient $\nabla L_i(\theta)$ đòi hỏi khối lượng tính toán lớn.
- Nếu dữ liệu có kích thước N lớn thì mỗi bước lặp có khối lượng tính toán lớn, thuật toán chạy chậm.

Giảm gradient ngẫu nhiên

- Để tiết kiệm khối lượng tính toán, phương pháp giảm gradient ngẫu nhiên (Stochastic Gradient Descent – SGD)[1] xấp xỉ gradient của $L(\theta)$ bởi gradient ở **một** mẫu huấn luyện:

$$\theta \leftarrow \theta - \alpha \nabla L_i(\theta).$$

- Khi chạy qua toàn bộ tập huấn luyện ($i = 1, 2 \ldots, N$), tham số θ được cập nhật dần trên từng mẫu.
- Ta có thể chạy lại thủ tục lặp này một số lần trên tập huấn luyện cho tới khi hội tụ.

[1] Online learning

Giảm gradient ngẫu nhiên

Algorithm 2: Stochastic Gradient Descent

Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
Result: θ
$\theta \leftarrow \vec{0}$;
repeat
    Shuffle the training set randomly;
    for $i = 1$ *to* N **do**
        $\theta \leftarrow \theta - \alpha \nabla L_i(\theta)$;
until *converged*;

Tốc độ học

- Trong các thuật toán giảm gradient, tốc độ học α có thể thay đổi theo bước lặp mà không cố định.
- Nói cách khác, ta có nhiều tốc độ học, công thức giảm gradient tổng quát là

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha^{(t)} \nabla L(\theta^{(t)}),$$

trong đó (t) là chỉ số của bước cập nhật tham số.

Tốc độ học

- Về mặt lí thuyết, tốc độ học $\alpha^{(t)}$ được chọn sao cho cực đại hoá $L(\theta^{(t+1)})$.
- Việc tìm $\alpha^{(t)}$ cũng là một bài toán tối ưu (trong không gian 1 chiều), có thể giải bằng những thuật toán tối ưu hiệu quả như những thuật toán tựa Newton.
- Tuy nhiên trong thực tế, ta không cần giá trị đúng của $\alpha^{(t)}$ mà chỉ cần giá trị xấp xỉ của nó để đảm bảo tính chất hội tụ của hàm mục tiêu.

Tốc độ học

Hai điều kiện cần của các tốc độ học để thuật toán hội tụ:

1. Các tốc độ học cần phải giảm dần, tức là:

$$\sum_t (\alpha^{(t)})^2 < \infty.$$

2. Các tốc độ học không được giảm quá nhanh về hằng số, tức là:

$$\sum_t \alpha^{(t)} = \infty.$$

Nhắc lại: Mô hình hồi quy tuyến tính

Hàm dự báo là hàm tuyến tính: $h_\theta(\mathbf{x}) = \theta^T \mathbf{x}$.

$$L_i(\theta) = \frac{1}{2}[h_\theta(\mathbf{x}_i) - y_i]^2$$
$$= \frac{1}{2}(\theta^T \mathbf{x}_i - y_i)^2.$$

Sai số trên tập huấn luyện:

$$L(\theta) = \sum_{i=1}^{N} L_i(\theta) = \frac{1}{2}\sum_{i=1}^{N}(\theta^T \mathbf{x}_i - y_i)^2$$

Ta cần tìm tham số θ sao cho sai số là nhỏ nhất:

$$L(\theta) \to \min .$$

Bài trước: sử dụng phương trình chuẩn để giải đúng θ

Giảm gradient theo loạt

Véc-tơ gradient của $L(\theta)$:

$$\nabla L(\theta) = \left(\frac{\partial L(\theta)}{\partial \theta_0}, \frac{\partial L(\theta)}{\partial \theta_1}, \ldots, \frac{\partial L(\theta)}{\partial \theta_D} \right)$$

Các thành phần của gradient:

$$\frac{\partial L(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\theta^T \mathbf{x}_i - y_i) x_j, \quad \forall j = 0, 2, \ldots, D.$$

Thuật toán giảm gradient theo loạt

Algorithm 3: Batch Gradient Descent for Linear Regression

Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

Result: $\theta = (\theta_0, \theta_1, \ldots, \theta_D)$

$\theta \leftarrow \vec{0}$;

repeat

  for $j = 0$ *to* D **do**

    $\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{N} (\theta^T \mathbf{x}_i - y_i) x_j$

until *converged*;

Thuật toán giảm gradient ngẫu nhiên

Algorithm 4: Stochastic Gradient Descent for Linear Regression

Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

Result: $\theta = (\theta_0, \theta_1, \ldots, \theta_D)$

$\theta \leftarrow \vec{0}$;

repeat

  for $j = 0$ *to* D **do**

    $\theta_j \leftarrow \theta_j - \alpha(\theta^T \mathbf{x}_i - y_i)x_j$

until *converged*;

Ví dụ: Hồi quy đơn giản

- Ta cần khớp một đường thẳng $y = \theta_0 + \theta_1 x$ với một tập dữ liệu huấn luyện hai chiều $(x_1, y_1), \ldots, (x_N, y_N)$
- Hàm mục tiêu cần cực tiểu hoá là

$$L(\theta) = \sum_{i=1}^{N} L_i(\theta) = \sum_{i=1}^{N} (\theta_0 + \theta_1 x_i - y_i)^2.$$

- θ được cập nhật theo công thức:

$$\begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} \leftarrow \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} - 2\alpha(\theta_0 + \theta_1 x_i - y_i) \begin{pmatrix} 1 \\ x_i \end{pmatrix}.$$

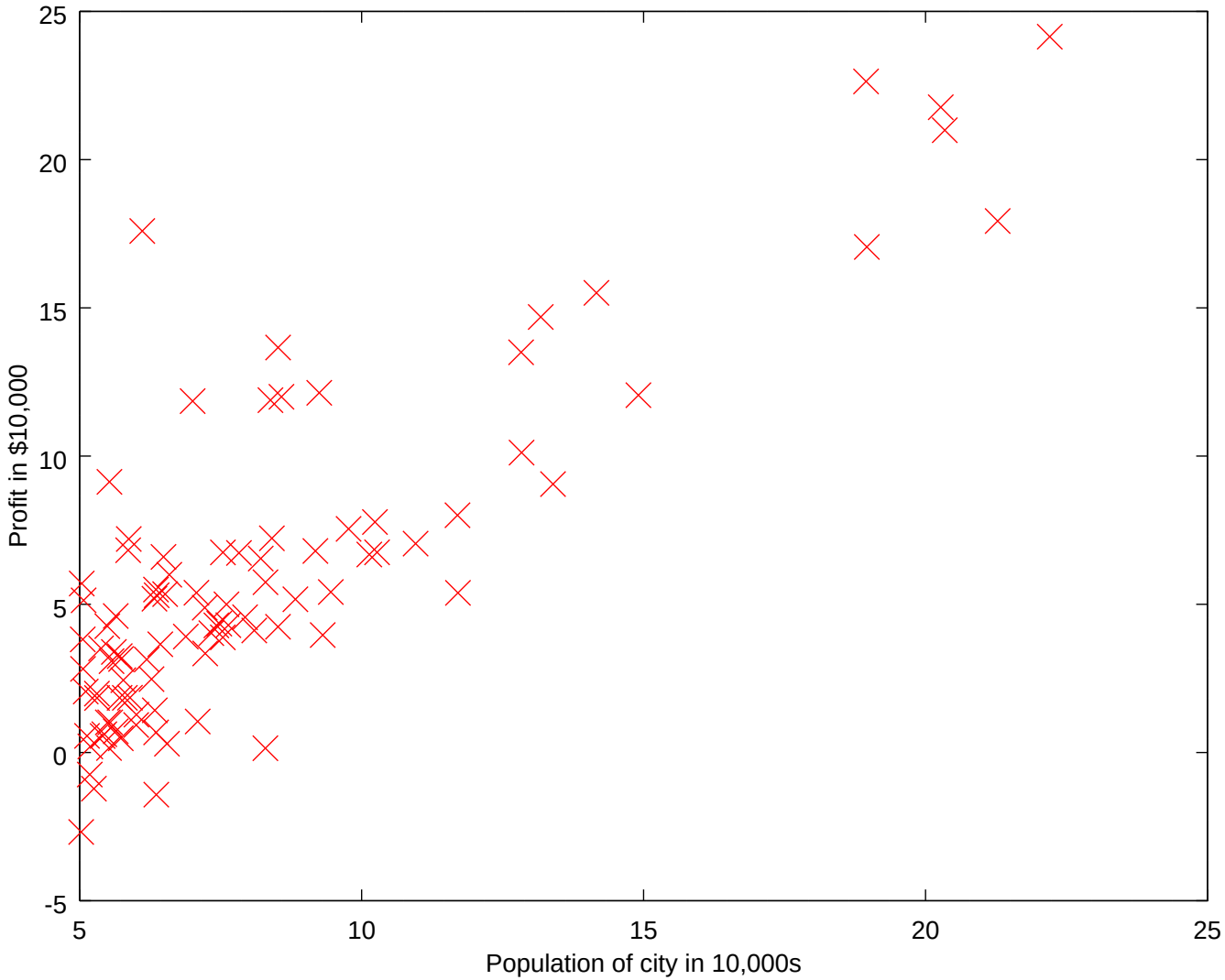
Example: Profit Prediction

- You are the CEO of a company which operates in various cities.
- You have data for profits and populations from the cities.

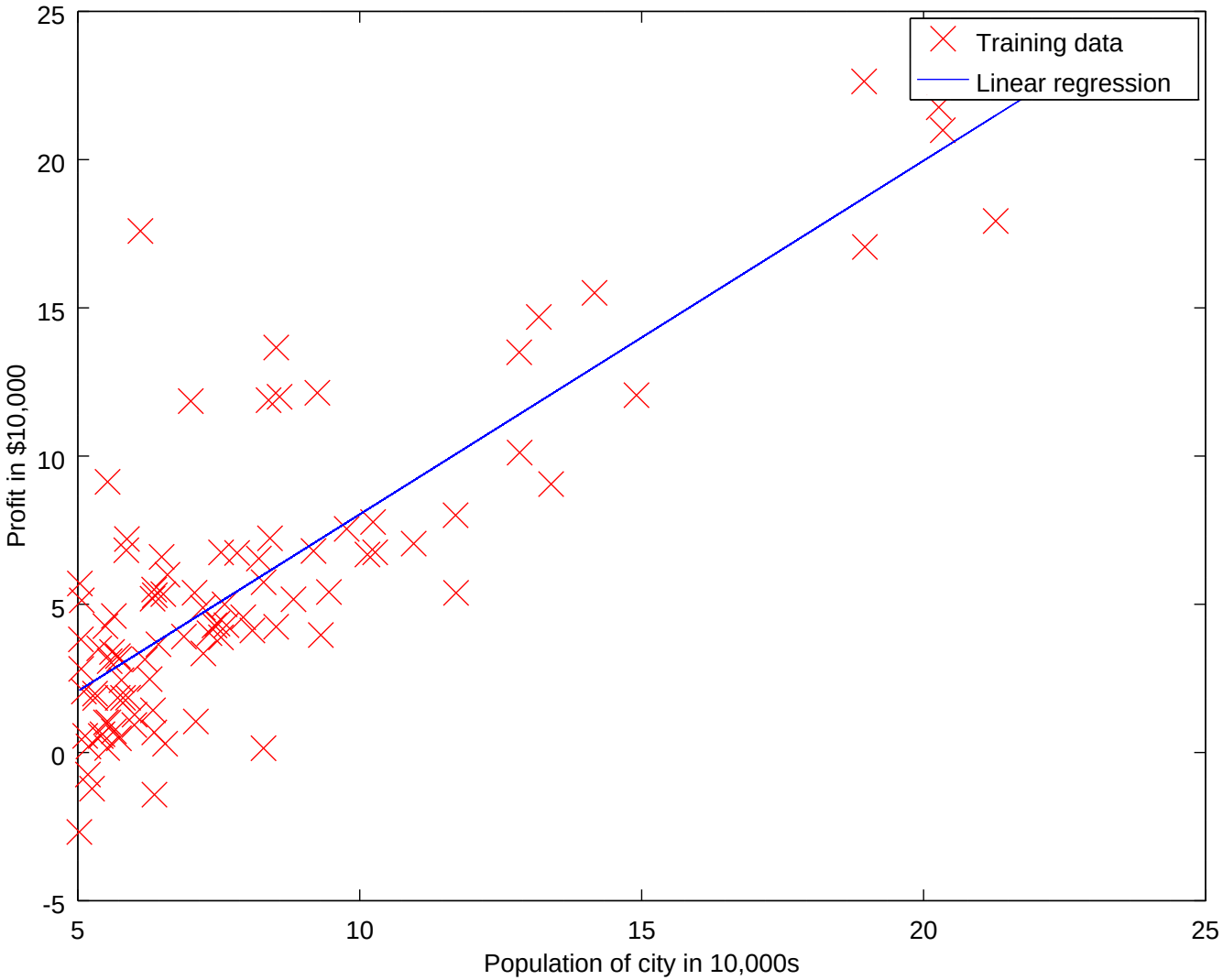
Population	**Profits**
6.1101	17.592
5.5277	9.1302
8.5186	13.662
7.0032	11.854
5.8598	6.8233
8.3829	11.886
7.4764	4.3483
8.5781	12
6.4862	6.5987
...	...

- You want to use this data to select which city to expand to next.

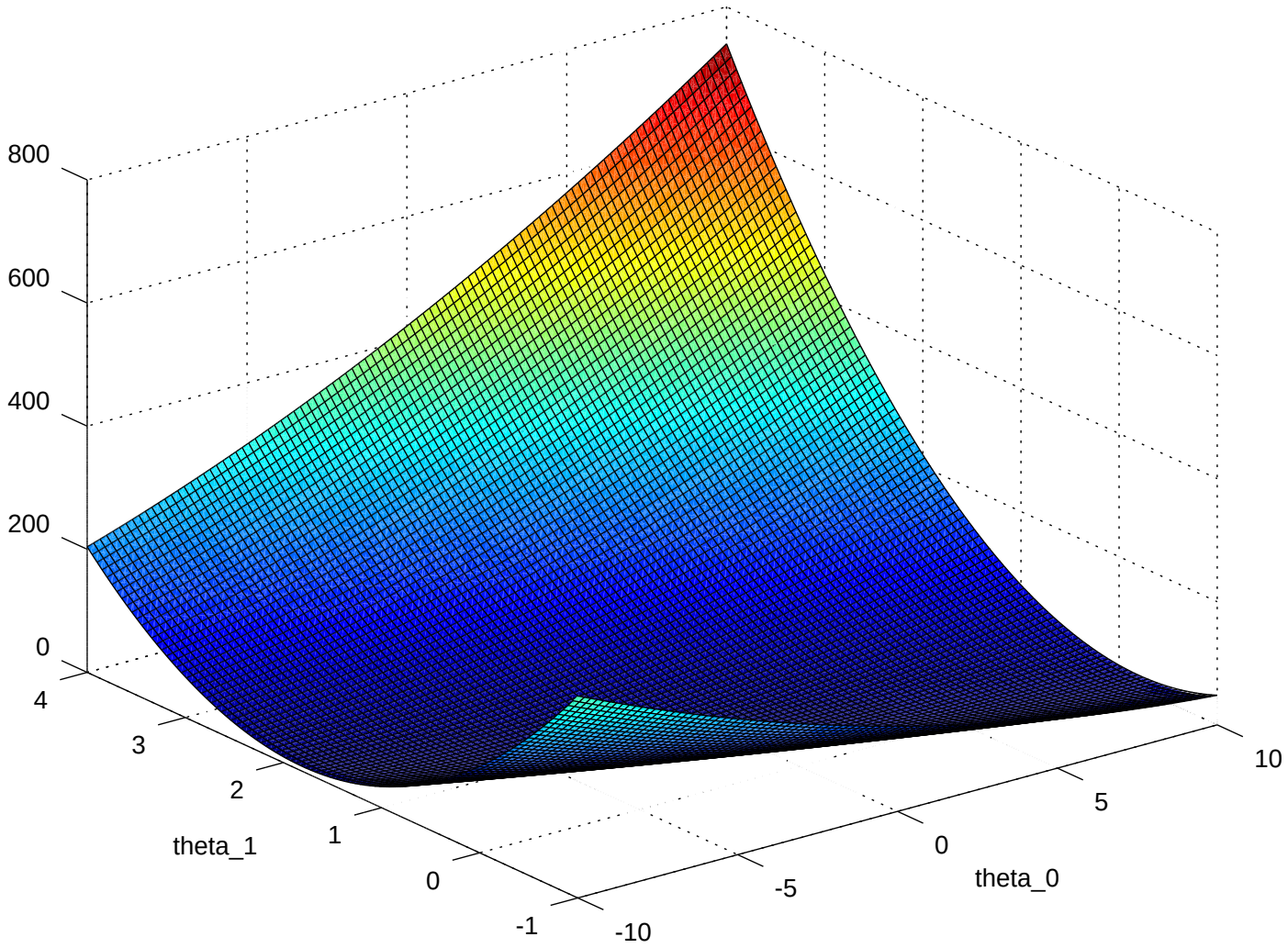
Scatter Plot



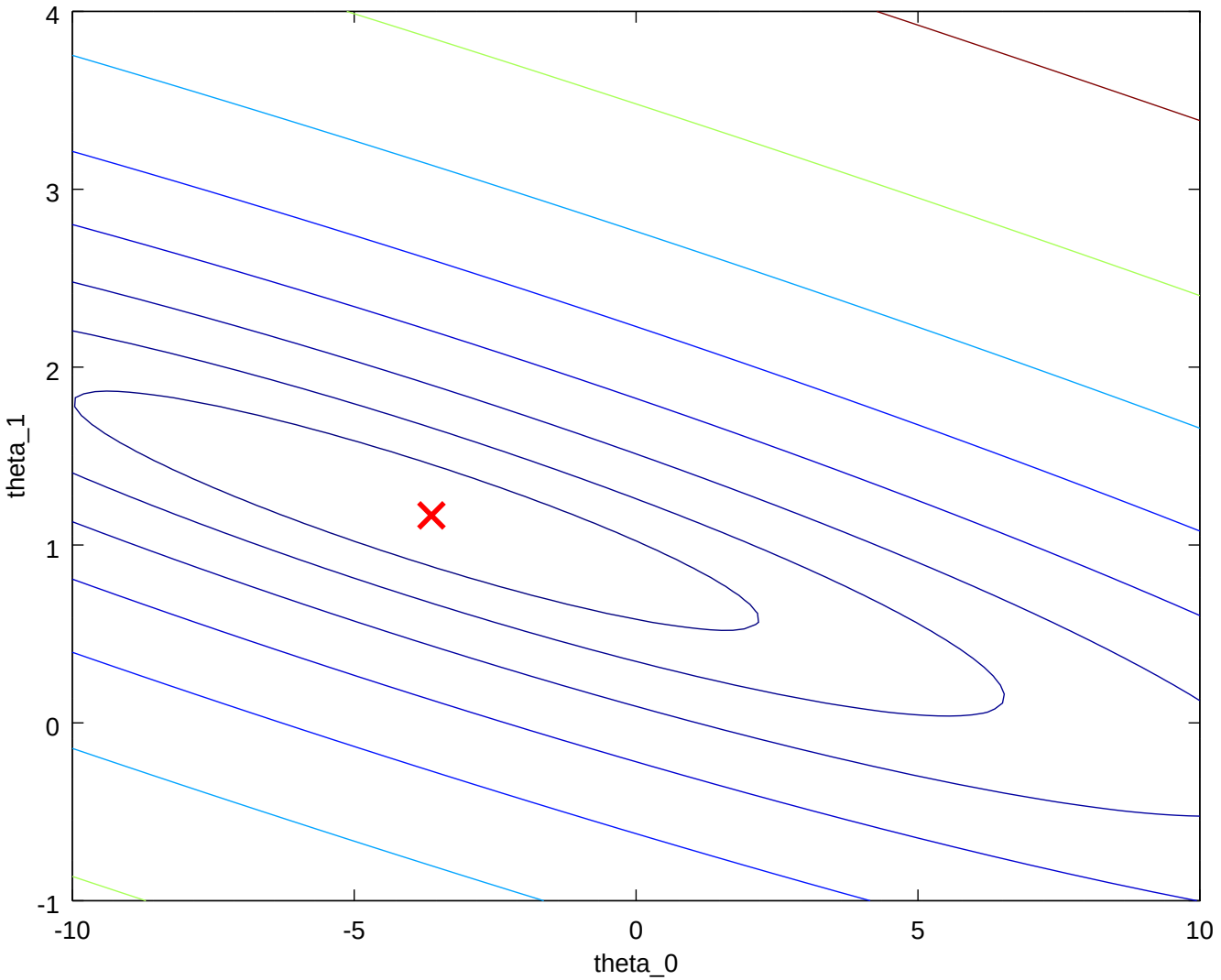
Linear Fitting



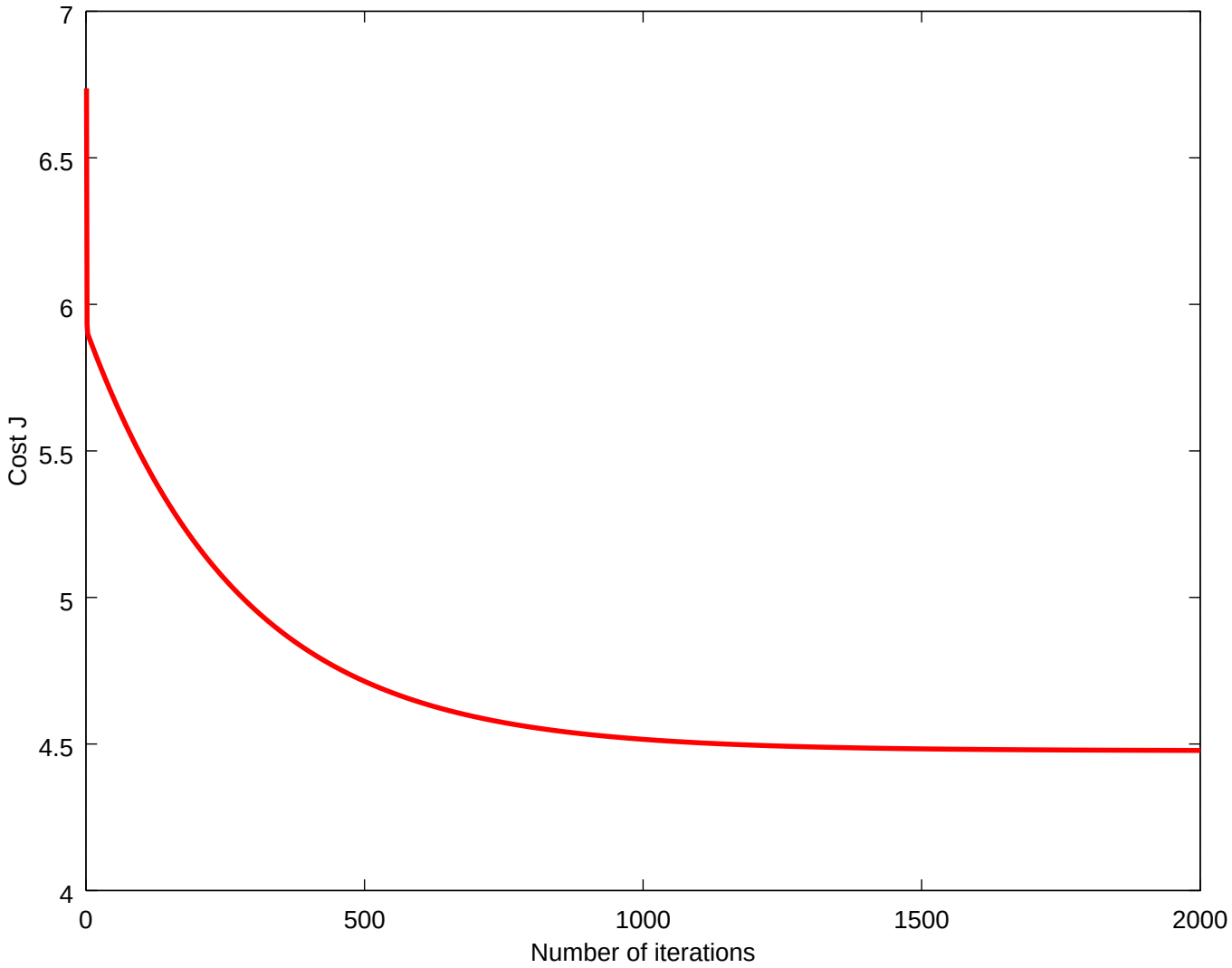
Objective Function



Gradient Descent: Contour



Gradient Descent: Convergence



Bài tập

1. Cài đặt thuật toán giảm gradient theo loạt ước lượng tham số của mô hình hồi quy tuyến tính.
2. Cài đặt thuật toán giảm gradient ngẫu nhiên ước lượng tham số của mô hình hồi quy tuyến tính.
3. So sánh kết quả với việc sử dụng phương trình chuẩn.