

# INF721

2023/2



# Aprendizado em Redes Neurais Profundas

## A4: Gradiente Descendente

# Logística

## Avisos

- ▶ Aula A3 – Regressão Logística publicada no site [slides, vídeo]
- ▶ Na próxima aula teremos o nosso primeiro Teste!
  - ▶ T1: Regressão Logística e Aprendizado de Máquina

## Última aula

- ▶ Funções de perda;
- ▶ Regressão Logística.

# Plano de Aula

- ▶ Cálculo para Otimização de RNAs
  - ▶ Derivadas
  - ▶ Derivadas parciais e vetor gradiente
  - ▶ Regra da cadeia
- ▶ Gradiente descendente
  - ▶ Regra de atualização de pesos
  - ▶ Taxa de atualização
  - ▶ Aplicação em regressão logística

# Resumo de Regressão Logística

## Entrada

Um exemplo  $x \in \mathbb{R}^d$

## Saída

A probabilidade de  $x$  ser da classe  $y = 1$

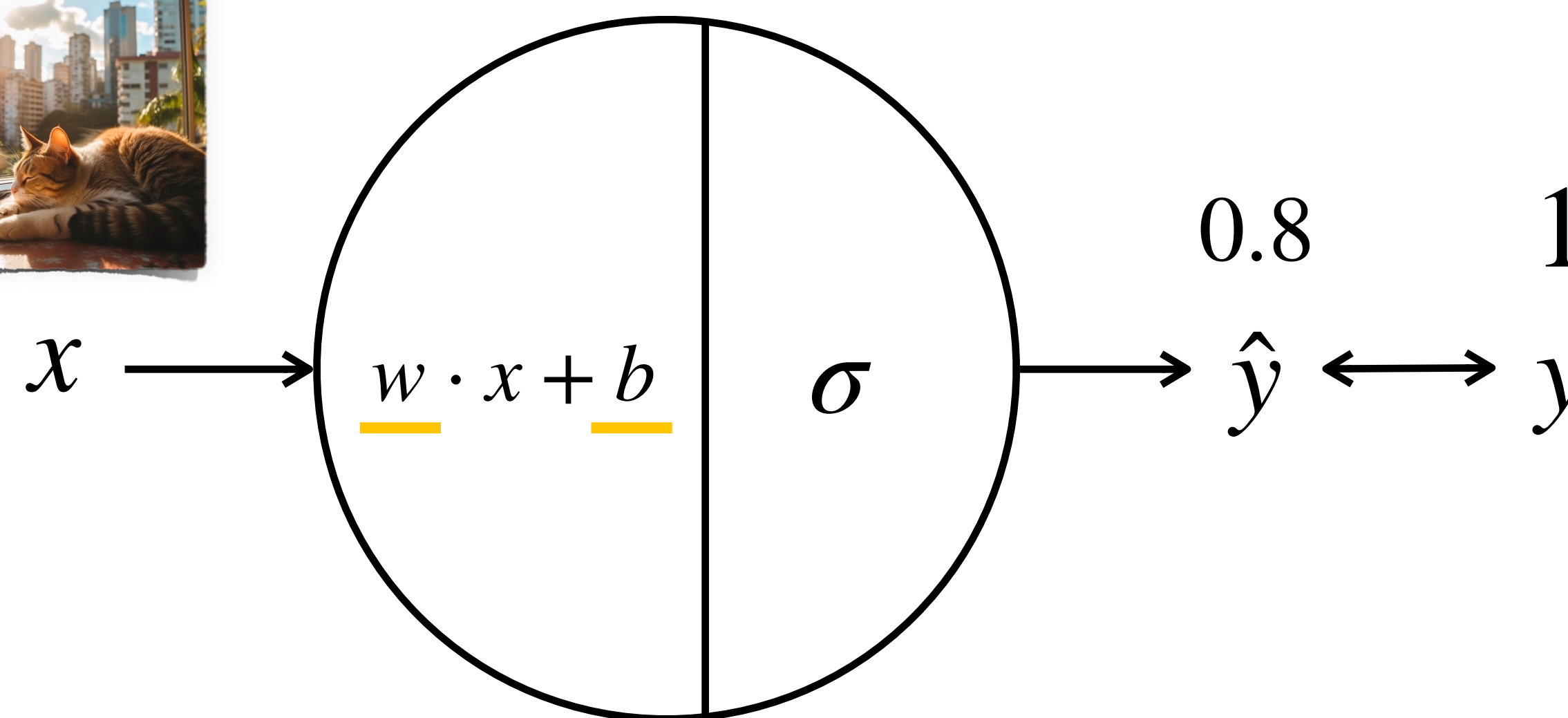
$$\hat{y} = P(y = 1 | x), 0 \leq \hat{y} \leq 1$$

## Hipótese

$$\hat{y} = h(x) = \sigma(w \cdot x + b)$$

## Função de Perda

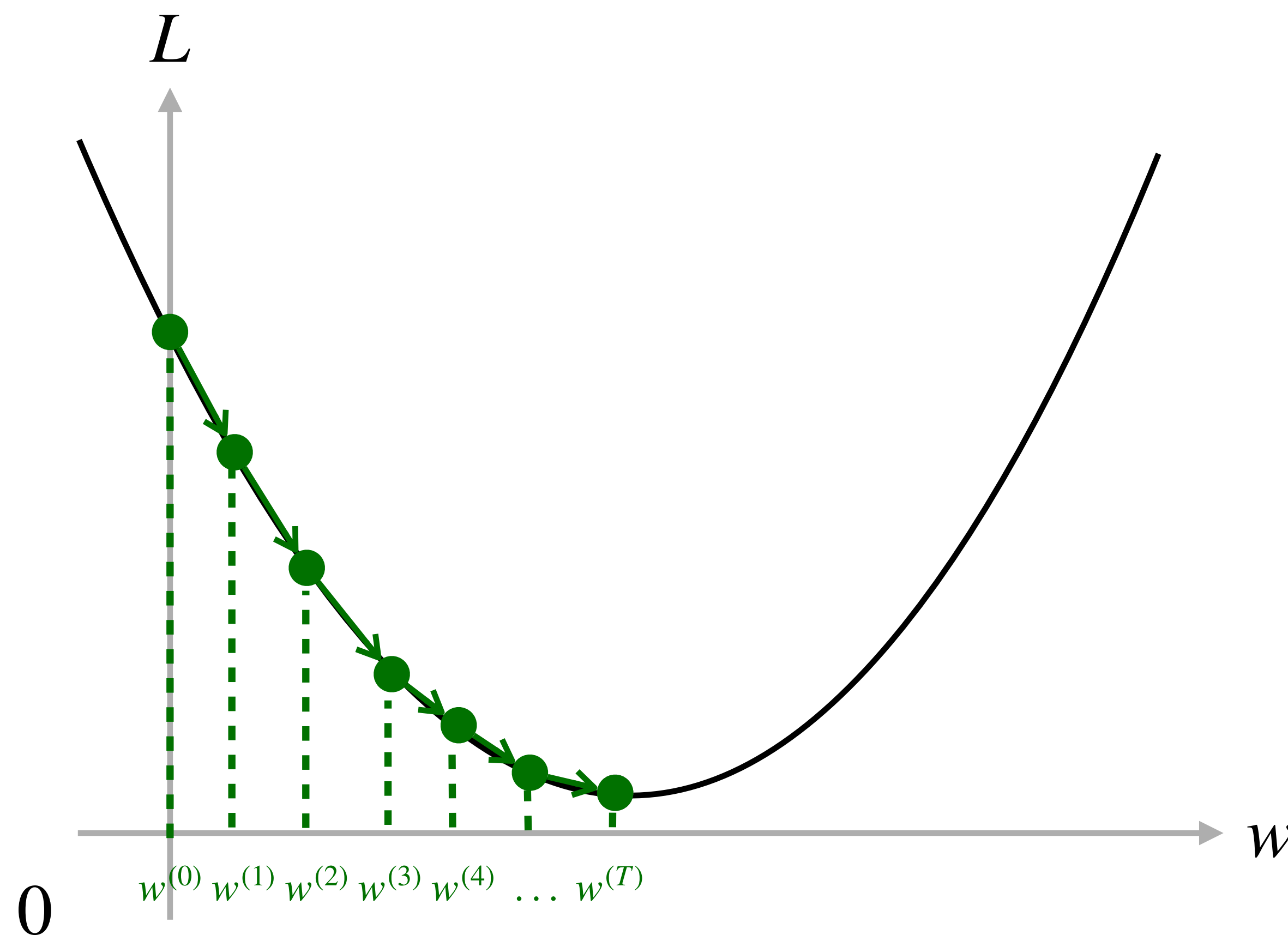
$$L(h) = -\frac{1}{n} \sum_{i=1}^n (y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i))$$



## Treinamento

Encontrar valores de  $w$  e  $b$  que minimizam  $L$ !

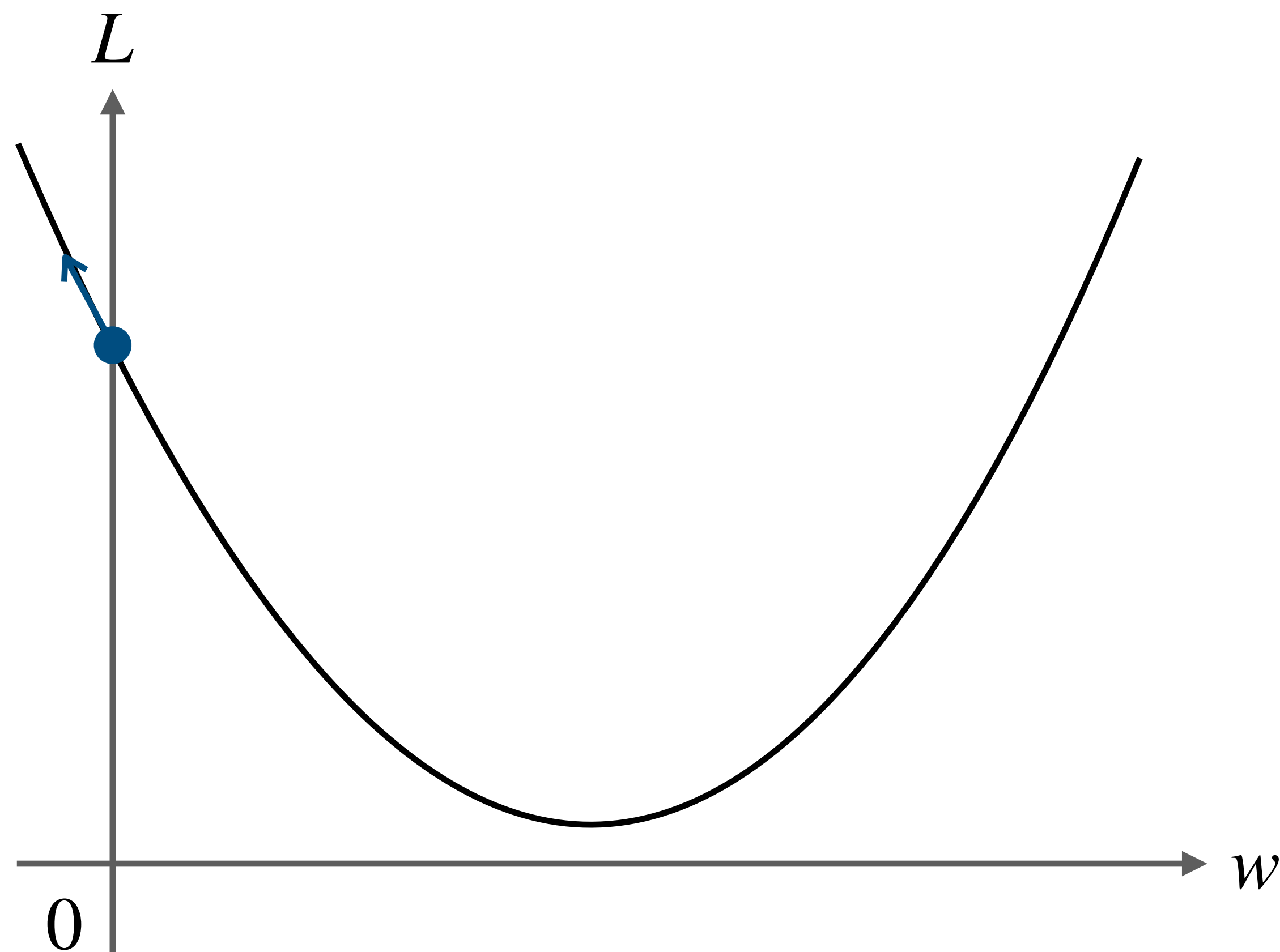
# Gradiente Descendente



Dado um valor inicial dos pesos  $w$  (e.g., 0), atualizamos iterativamente o valor de  $w$  na direção do ponto de descida mais íngreme de  $L$ .

Como calcular a direção do movimento?  
**Vetor gradiente  $\nabla L(w)$ !**

# Vetor Gradiente



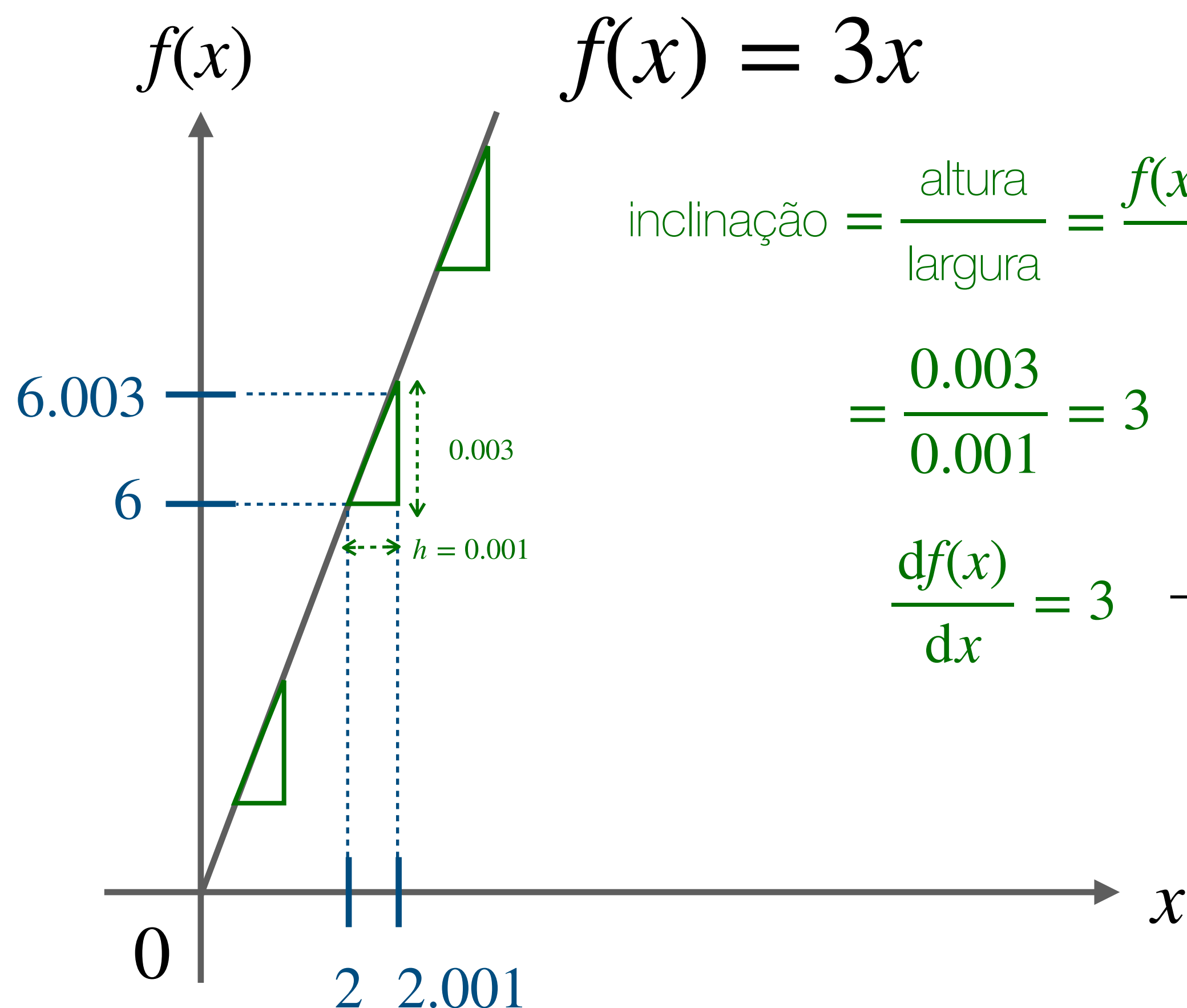
O **gradiente** de uma função multivariável  $L$  é um vetor contendo suas **derivadas parciais**:

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial w_d} \end{bmatrix}$$

O vetor  $\nabla L(w_0)$  nos diz a direção que aumenta o valor de  $L$  mais rapidamente a partir do ponto  $(w, L(w))$ .

# Derivadas

A derivada de uma função  $f(x)$  no ponto  $x = a$  representa a **inclinação** da reta tangente a essa função no ponto  $(a, f(a))$



$$h = 0.001$$

$$x = 2$$

$$f(x) = 6$$

$$x + h = 2.001$$

$$f(x + h) = 6.003$$

$$x = 5$$

$$f(x) = 15$$

$$x + h = 5.001$$

$$f(x + h) = 15.003$$

A derivada de  $f(x)$  no ponto  $x = 2$  é 3

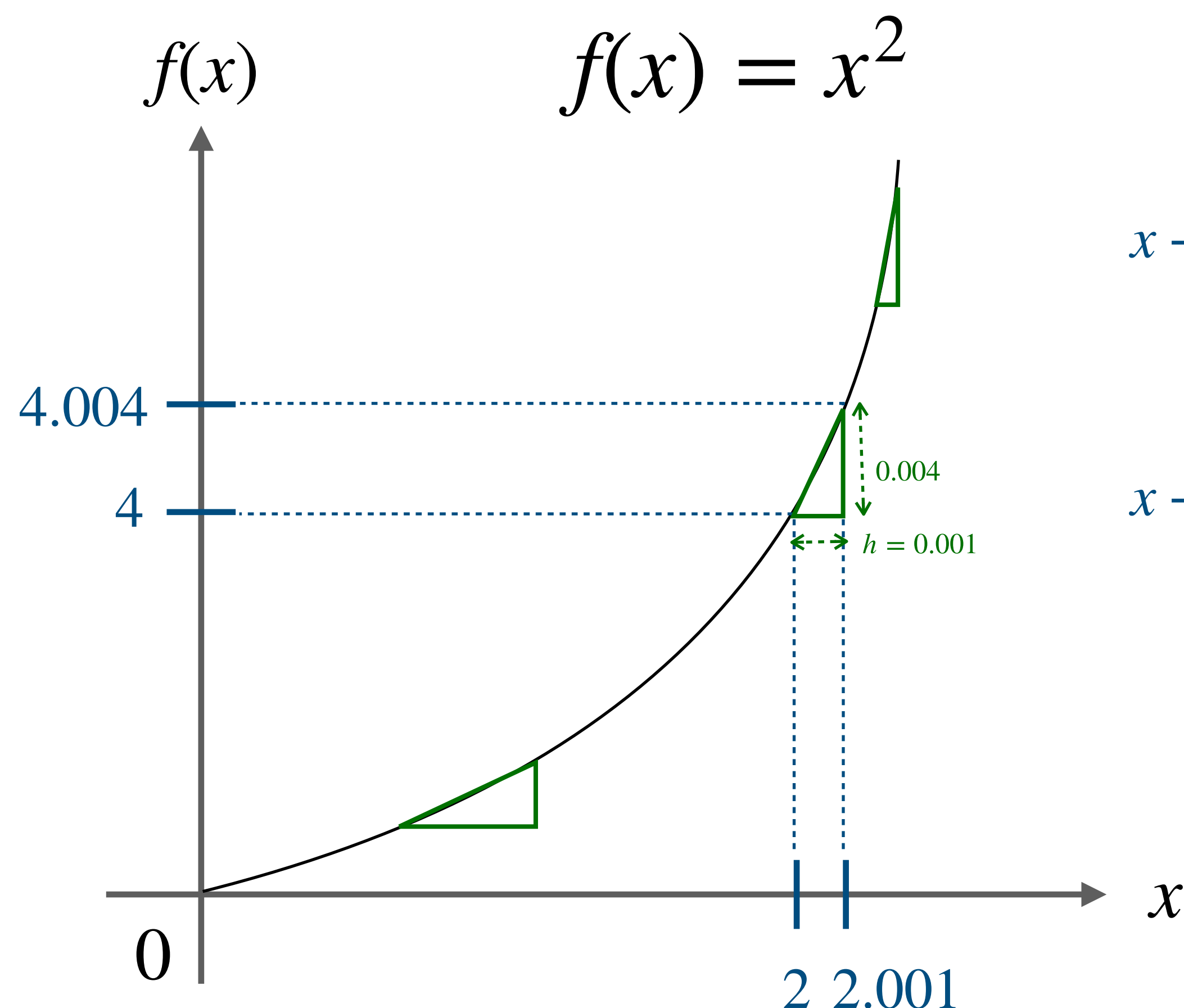
Quando aplicamos uma variação minúscula  $h$  em  $x$ , o quanto ela afeta o valor de  $f(x)$

Em cálculo, essa variação  $h$  é infinitamente pequena:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

# Derivadas

A derivada de uma função  $f(x)$  no ponto  $x = a$  representa a **inclinação** da reta tangente a essa função no ponto  $(a, f(a))$



$$h = 0.001$$

$$x = 2$$

$$x + h = 2.001$$

$$f(x) = 4$$

$$f(x + h) = 4.004$$

$$\frac{df(x)}{dx} = \frac{0.004}{0.001} = 4$$

$$x = 5$$

$$x + h = 5.001$$

$$f(x) = 25$$

$$f(x + h) = 25.010$$

$$\frac{df(x)}{dx} = \frac{0.0010}{0.001} = 10$$

$$\frac{df(x)}{dx} = \frac{dx^2}{dx} = 2x$$



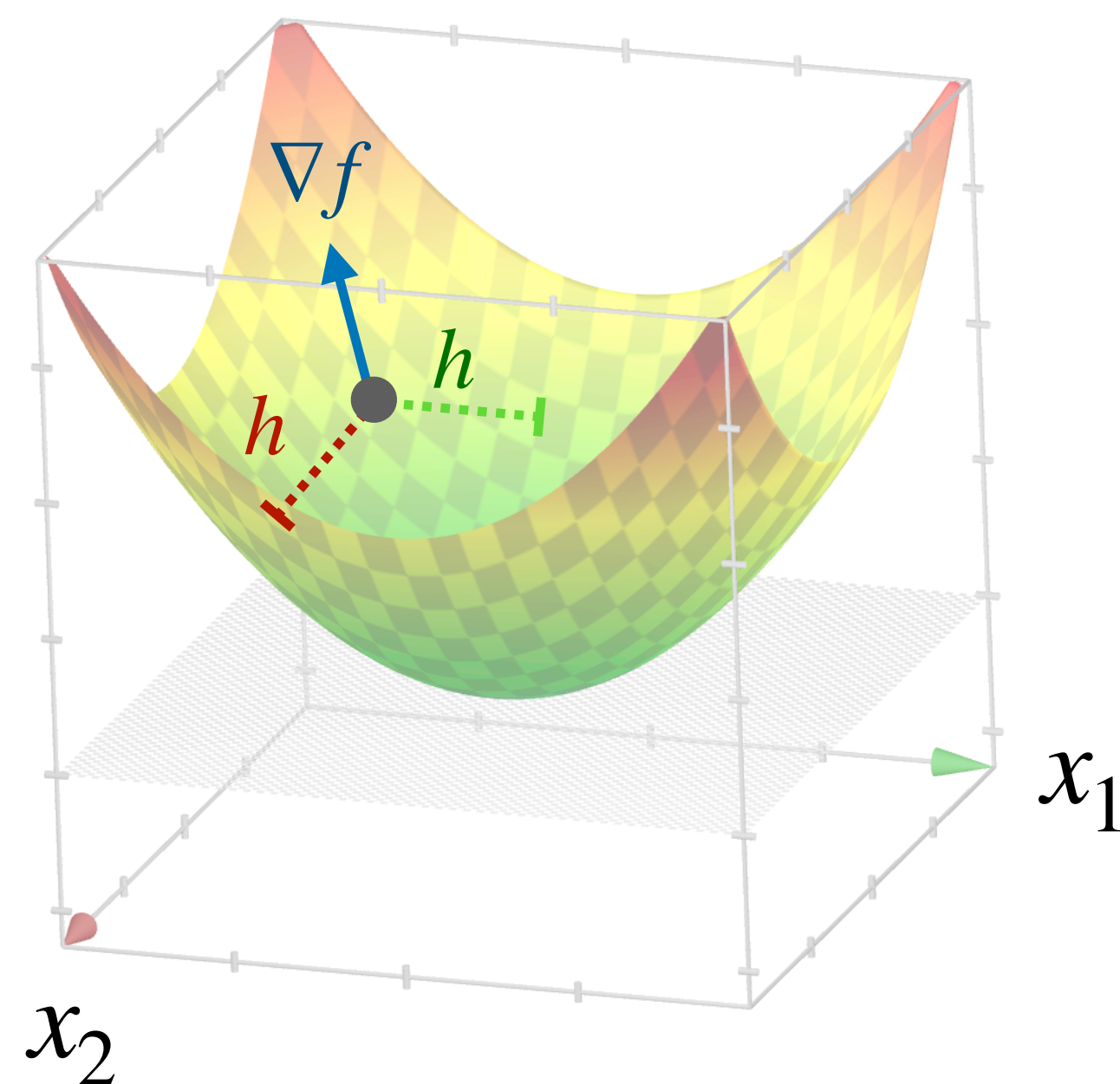
# Derivadas em Python

```
def f(x):  
    return x**2  
  
def derivative(x):  
    """Compute the exact derivative of `f` at the location `x`."""  
    return 2*x  
  
def approx_derivative(f, x, h=0.001):  
    """Compute the approximate derivative of `f` at the location `x`."""  
    return (f(x+h) - f(x))/h  
  
x = 3  
da_exact = derivative(x)  
da_approx = approx_derivative(f,x)  
print(da_exact, da_approx)
```

# Derivadas Parciais

A **derivada parcial** de uma função multivariável  $f(x_1, x_2, \dots, x_d)$ , no ponto representa como a função  $f$  muda quando aplicamos uma minúscula variação  $h$  em apenas uma das variáveis  $x_i$ .

$$f(x_1, x_2) = x_1^2 + x_2^2$$



$$(x_1, x_2) = (2, 5)$$

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = \frac{\partial x_1^2}{\partial x_1} + \frac{\partial x_2^2}{\partial x_1} = 2x_1 + 0 = 2x_1 = 2 \times 2 = 4$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = \frac{\partial x_1^2}{\partial x_2} + \frac{\partial x_2^2}{\partial x_2} = 0 + 2x_2 = 2x_2 = 2 \times 5 = 10$$

O vetor gradiente é definido pelas derivas parciais de  $\nabla f(x_1, x_2)$

$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f(x_1, x_2)}{\partial x_1} \\ \frac{\partial f(x_1, x_2)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 10 \end{bmatrix}$$

# Regra da Cadeia

$$f(x) = (x^2 + 1)^3$$

Função interna:

$$g(x) = x^2 + 1 \quad \frac{dg}{dx} = 2x$$

Função externa:

$$f(g(x)) = g(x)^3 \quad \frac{df}{dg} = 3(g(x))^2$$

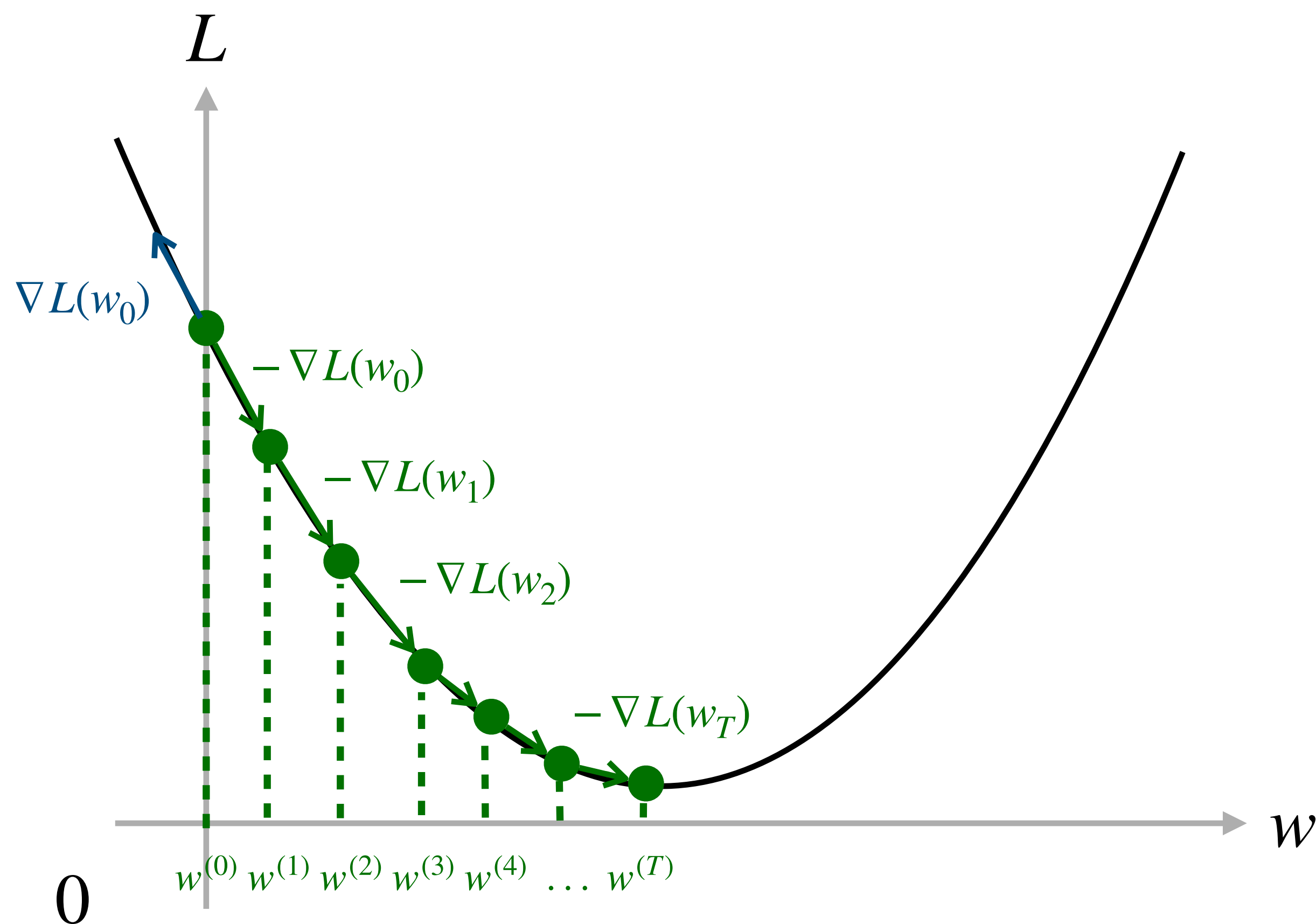
$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx} = 3(x^2 + 1)^2 \cdot (2x) = 6x(x^2 + 1)^2$$

Para calcular a derivada de uma função composta  $f(g(x))$ , temos que usar a **regra da cadeia**:

$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

A derivada da função composta  $f(g(x))$  é o produto das derivadas da função externa  $f$  em relação a

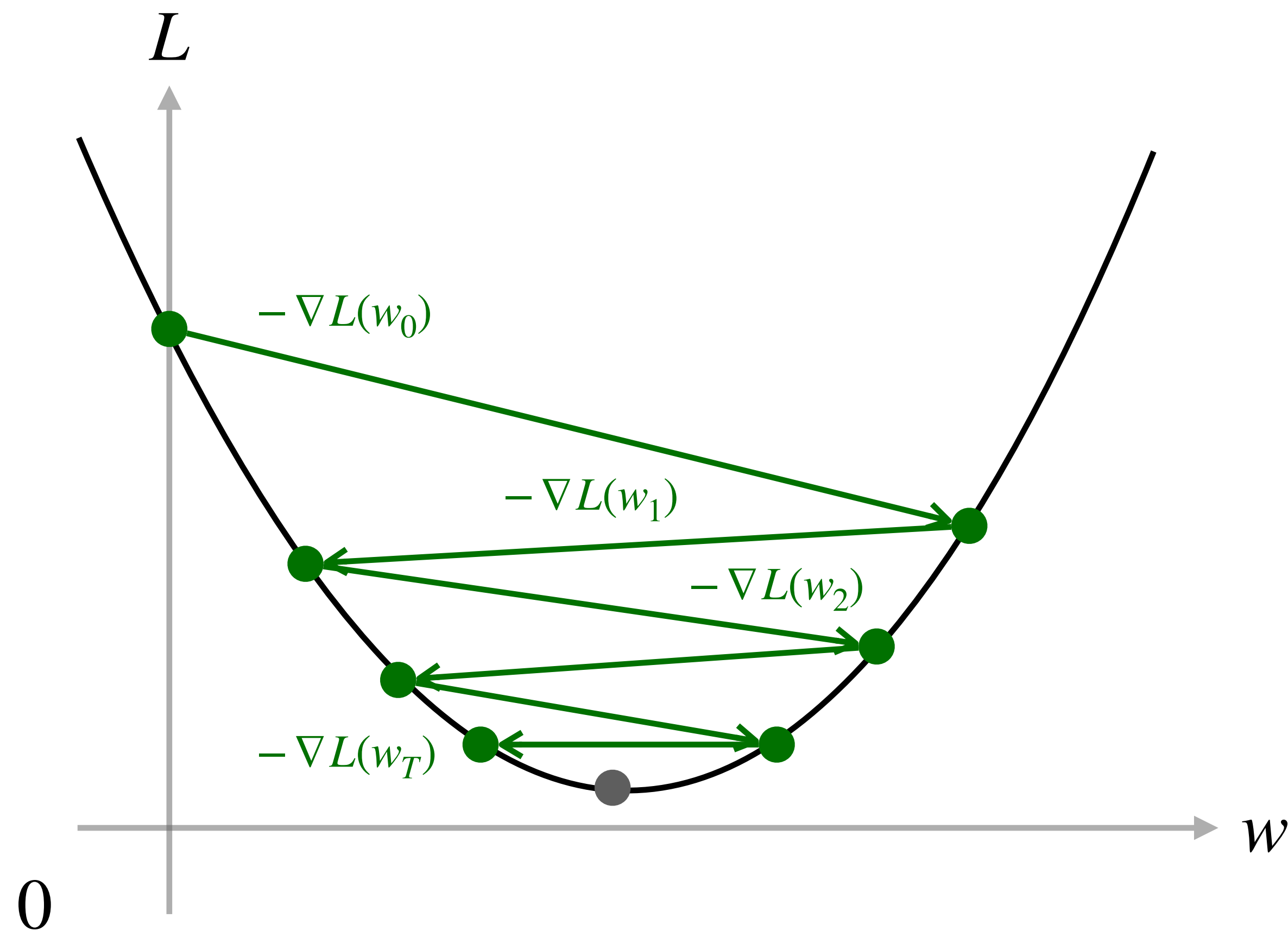
# Gradiente Descendente



Dado um valor inicial dos pesos  $w$  (e.g., 0), atualizamos iterativamente o valor de  $w$  na direção do ponto de descida mais íngreme de  $L$ .

$$w_t \leftarrow w_{t-1} - \nabla L(w_{t-1})$$

# Taxa de Aprendizado



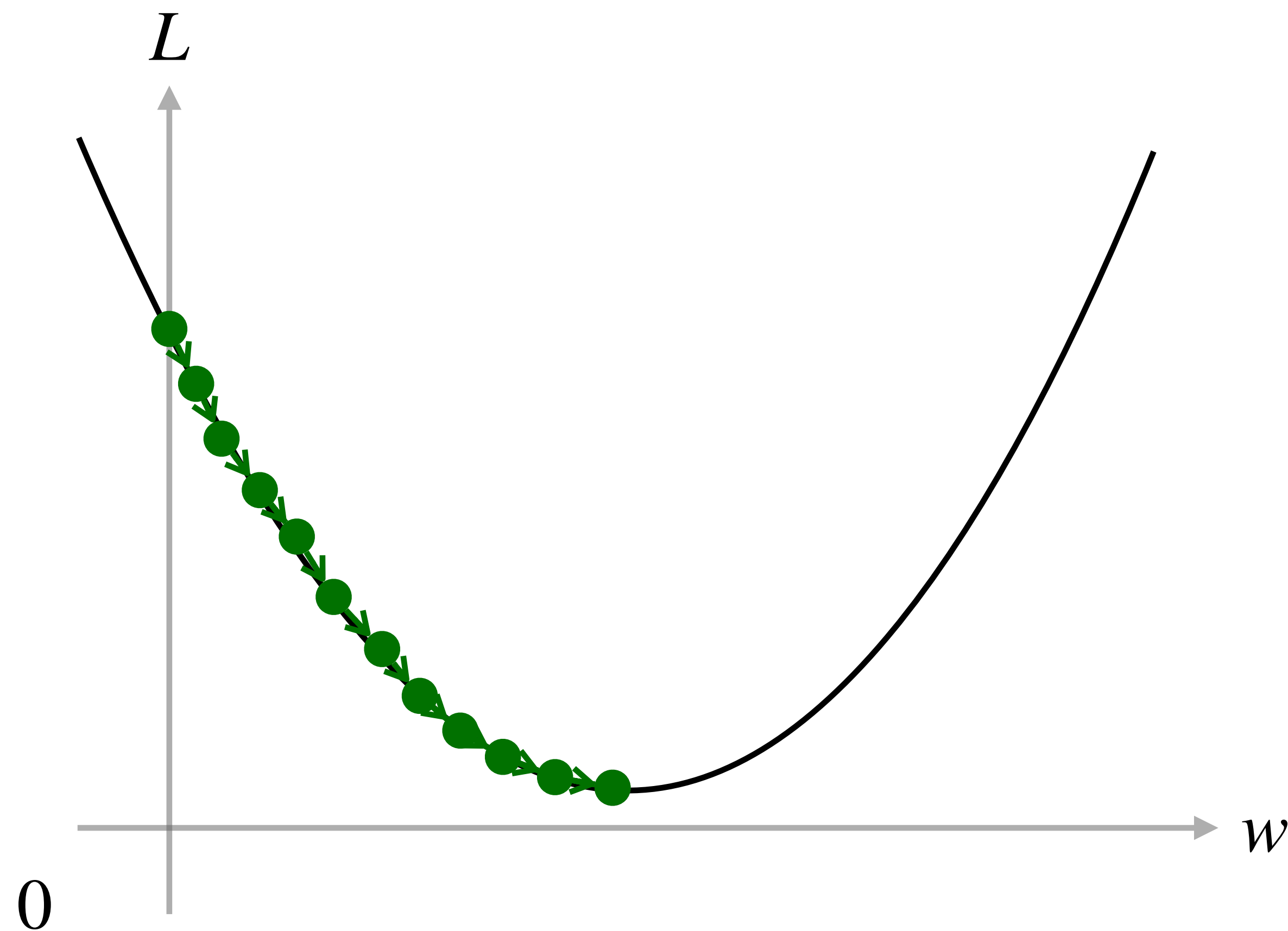
$$w_t \leftarrow w_{t-1} - \nabla L(w_{t-1})$$

Essa regra de atualização de  $w$  não tem controle sobre o comprimento do vetor gradiente  $\nabla L(w_{t-1})$ :

**Gradiente muito grande**

Convergência rápida, porém subótima!

# Taxa de Aprendizado



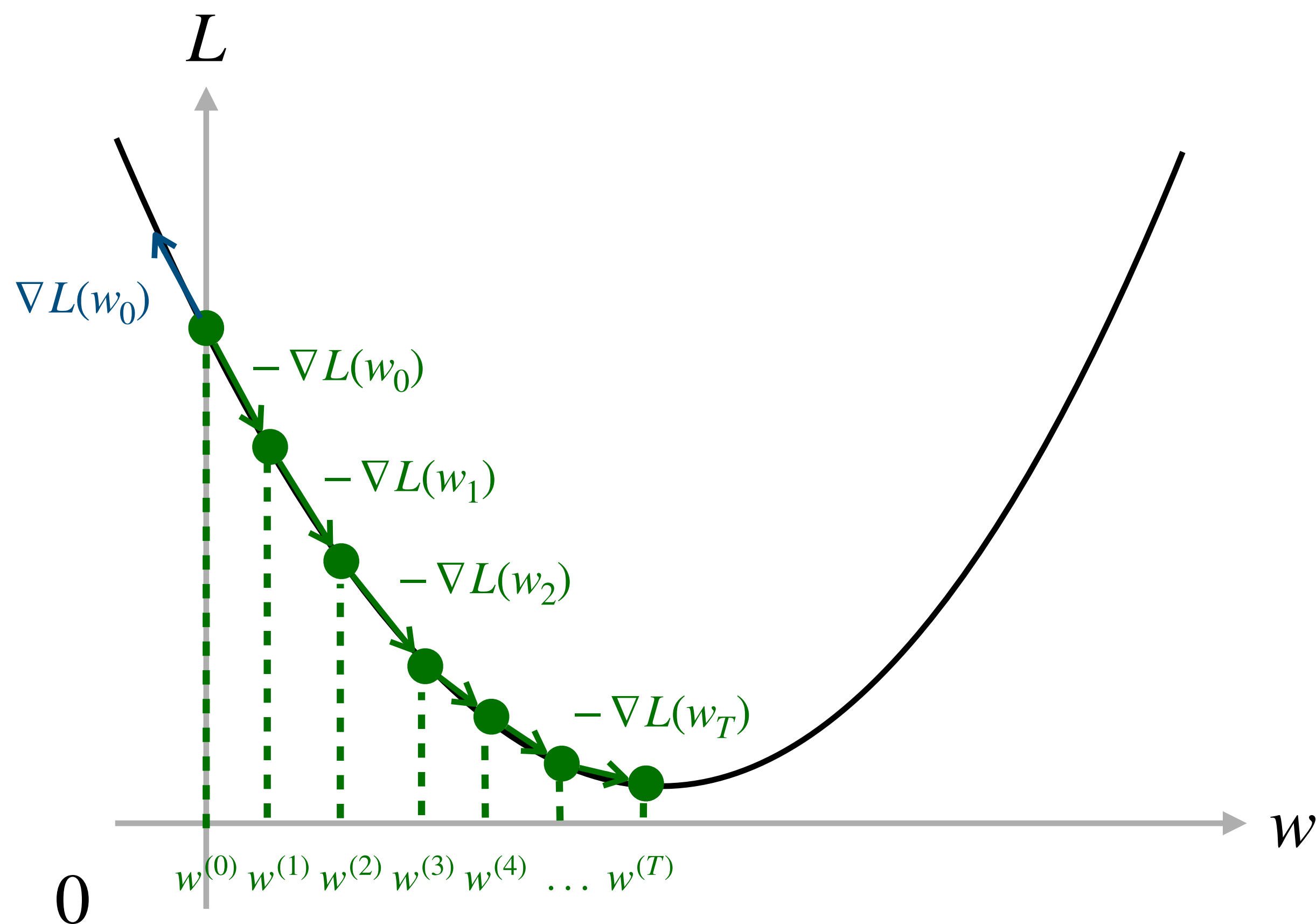
$$w_t \leftarrow w_{t-1} - \nabla L(w_{t-1})$$

Essa regra de atualização de  $w$  não tem controle sobre o comprimento do vetor gradiente  $\nabla L(w_{t-1})$ :

**Gradiente muito pequeno**

Convergência lenta e pode ficar presa em mínimos locais!

# Gradiente Descendente



Dado um valor inicial dos pesos  $w$  (e.g., 0), atualizamos iterativamente o valor de  $w$  na direção do ponto de descida mais íngreme de  $L$ .

$$w_t \leftarrow w_{t-1} - \alpha \nabla L(w_{t-1})$$

onde  $\alpha$  é um hiper-parâmetro chamado de **taxa de aprendizado** (*learning rate*), responsável por controlar o comprimento do vetor gradiente.

# Gradiente Descendente em Regressão Logística

## Hipótese

$$z = w \cdot x + b$$

$$\hat{y} = h(x) = \frac{1}{1 + e^{-z}}$$

## Função de Perda

$$L(h) = -\frac{1}{n} \sum_{i=1}^n (y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i))$$

## Gradiente

$$\nabla L(w_j) = (\hat{y} - y)x_j$$



# Próxima aula

## **A5:** Regressão Logística em Numpy

Aula prática sobre implementação de regressão logística com Numpy.