COMP 414/514: Optimization – Algorithms, Complexity and Approximations

Lecture 2

Overview

$$\min_{x} \quad f(x)$$

$$\text{s.t.} \quad x \in \mathcal{C}$$

- Different objective classes
- Different strategies within each problem
- Different approaches based on computational capabilities
- Different approaches based on constraints

And, always having in mind applications in machine learning, AI and signal processing

Overview

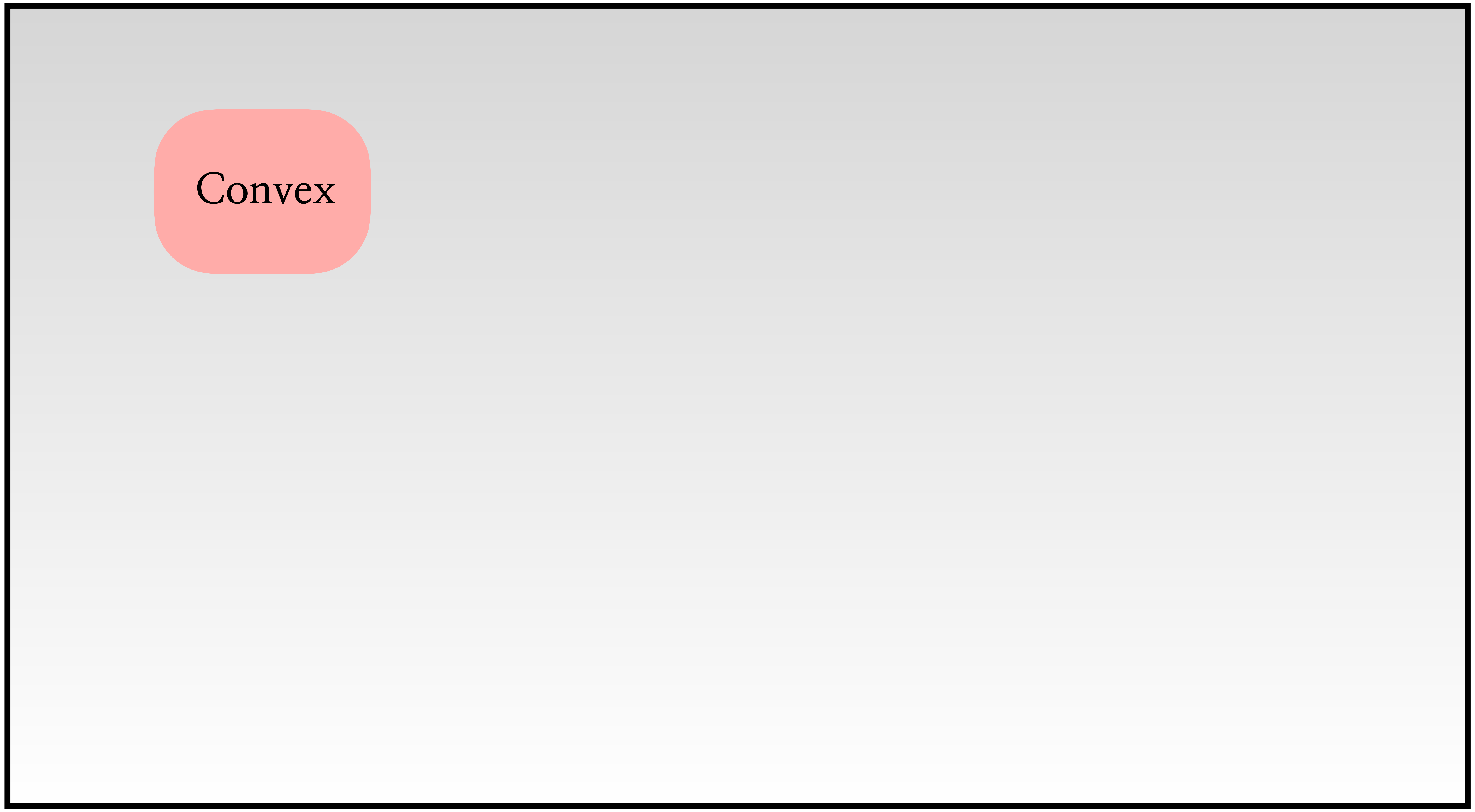
- In the last lecture, we:
    - Introduced some very basic ideas from linear algebra

- In this lecture, we will:
    - Discuss briefly **smooth continuous optimization**
    - Introduce derivatives, Taylor approximation, Lipschitz conditions
    - Discuss about gradient descent, and provide the first **convergence rate**

Convex vs. non−convex optimization

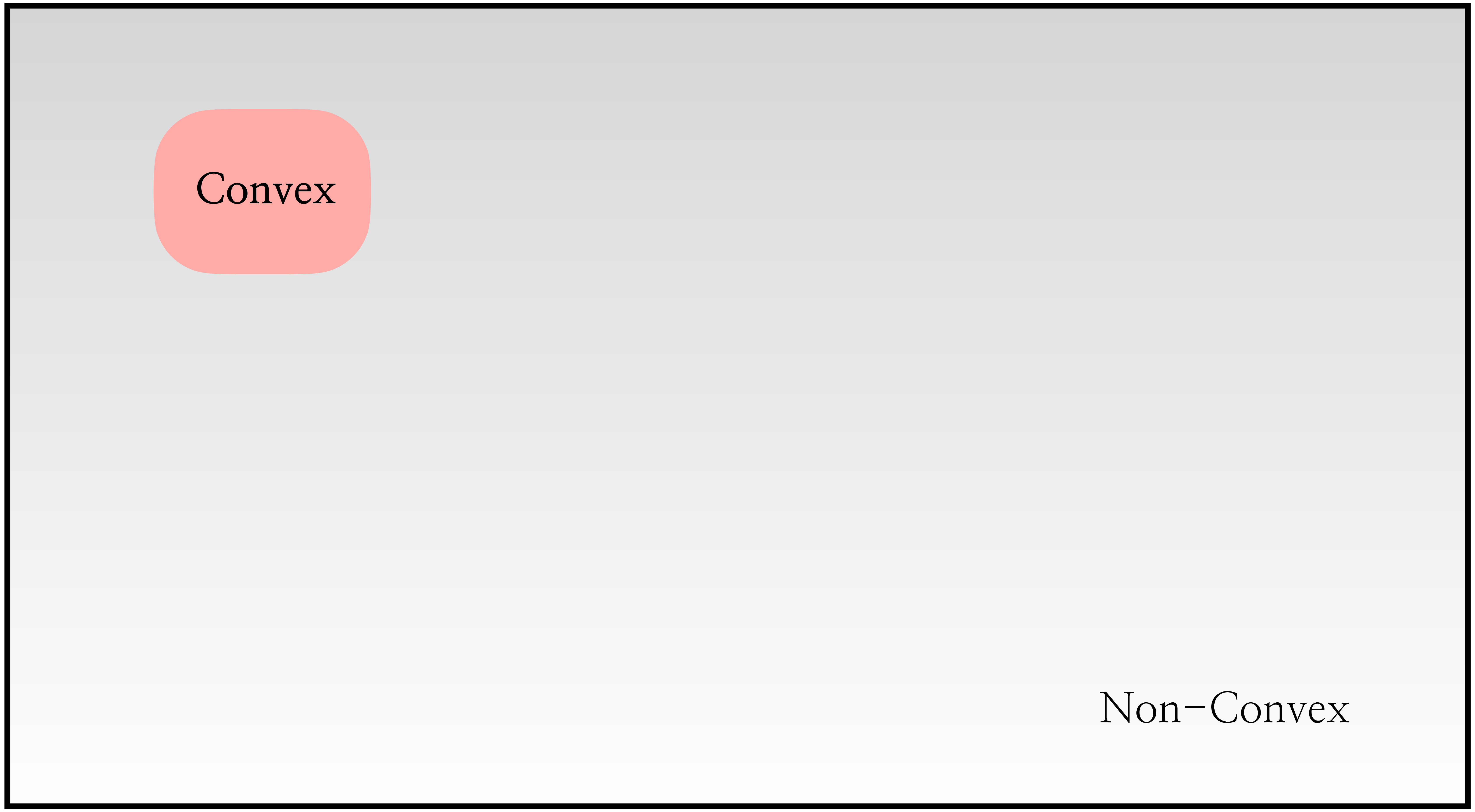
(Naive interpretation of) Space of optimization problems

Convex vs. non−convex optimization



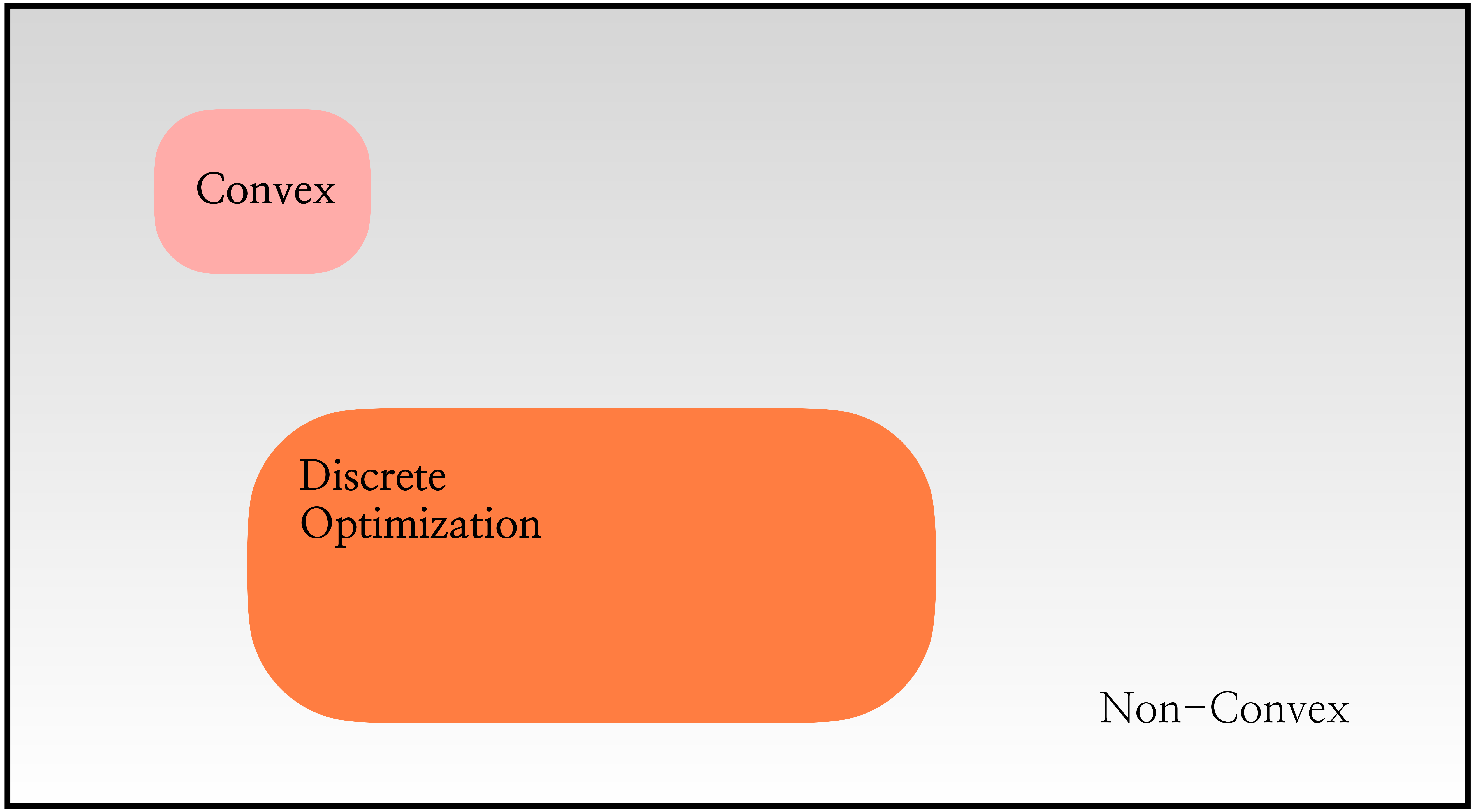
(Naive interpretation of) Space of optimization problems

Convex vs. non−convex optimization



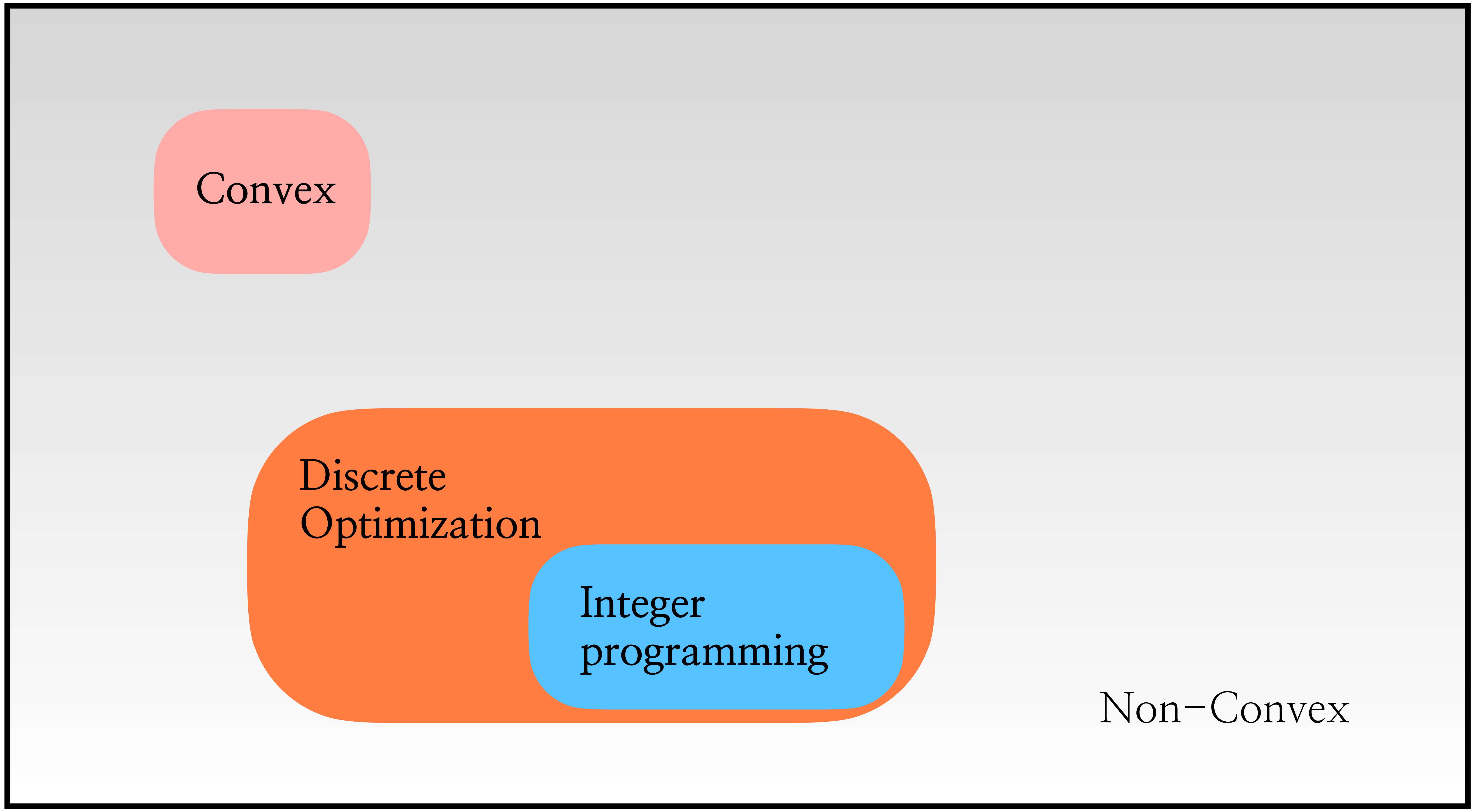
(Naive interpretation of) Space of optimization problems

Convex vs. non−convex optimization



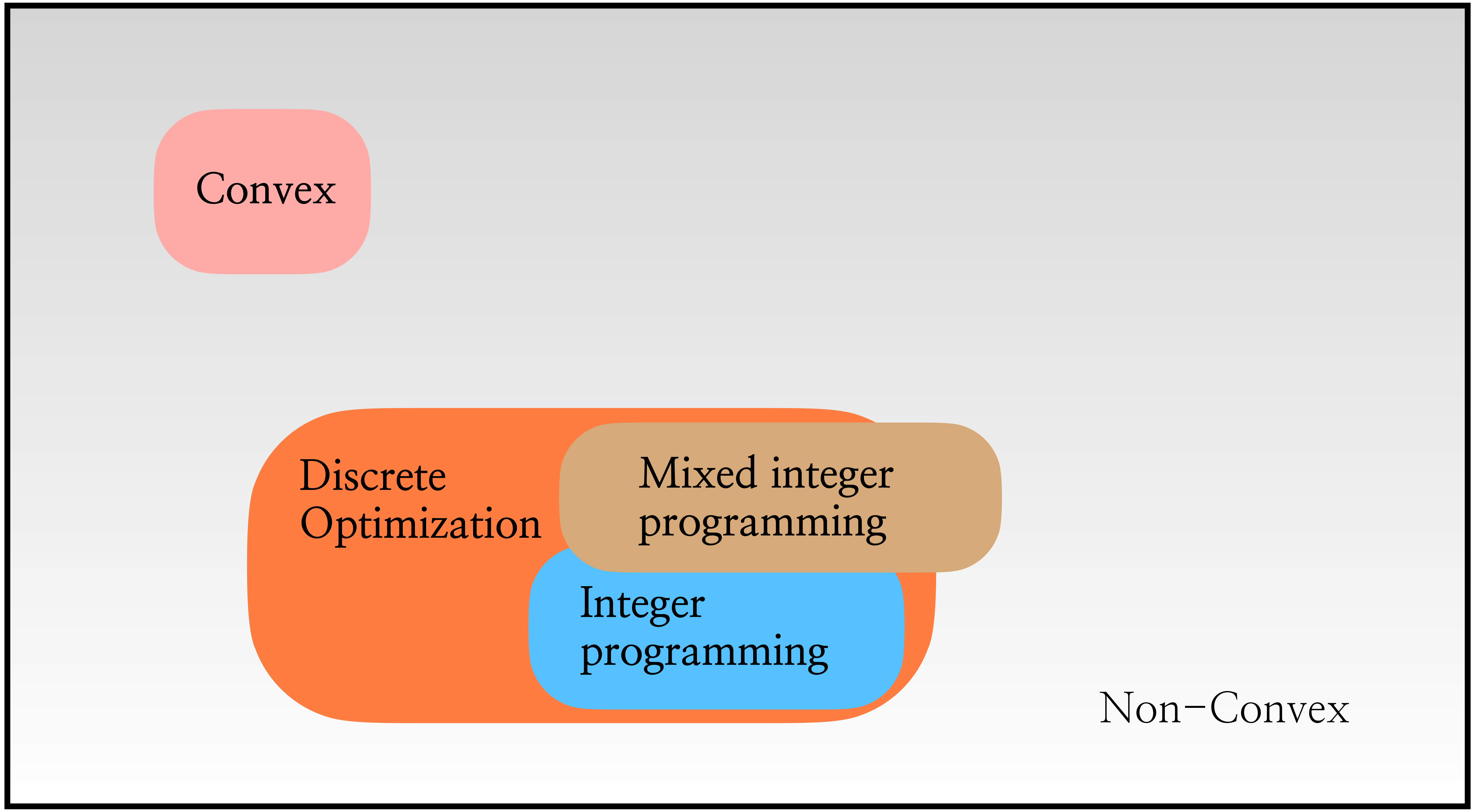
(Naive interpretation of) Space of optimization problems

Convex vs. non−convex optimization



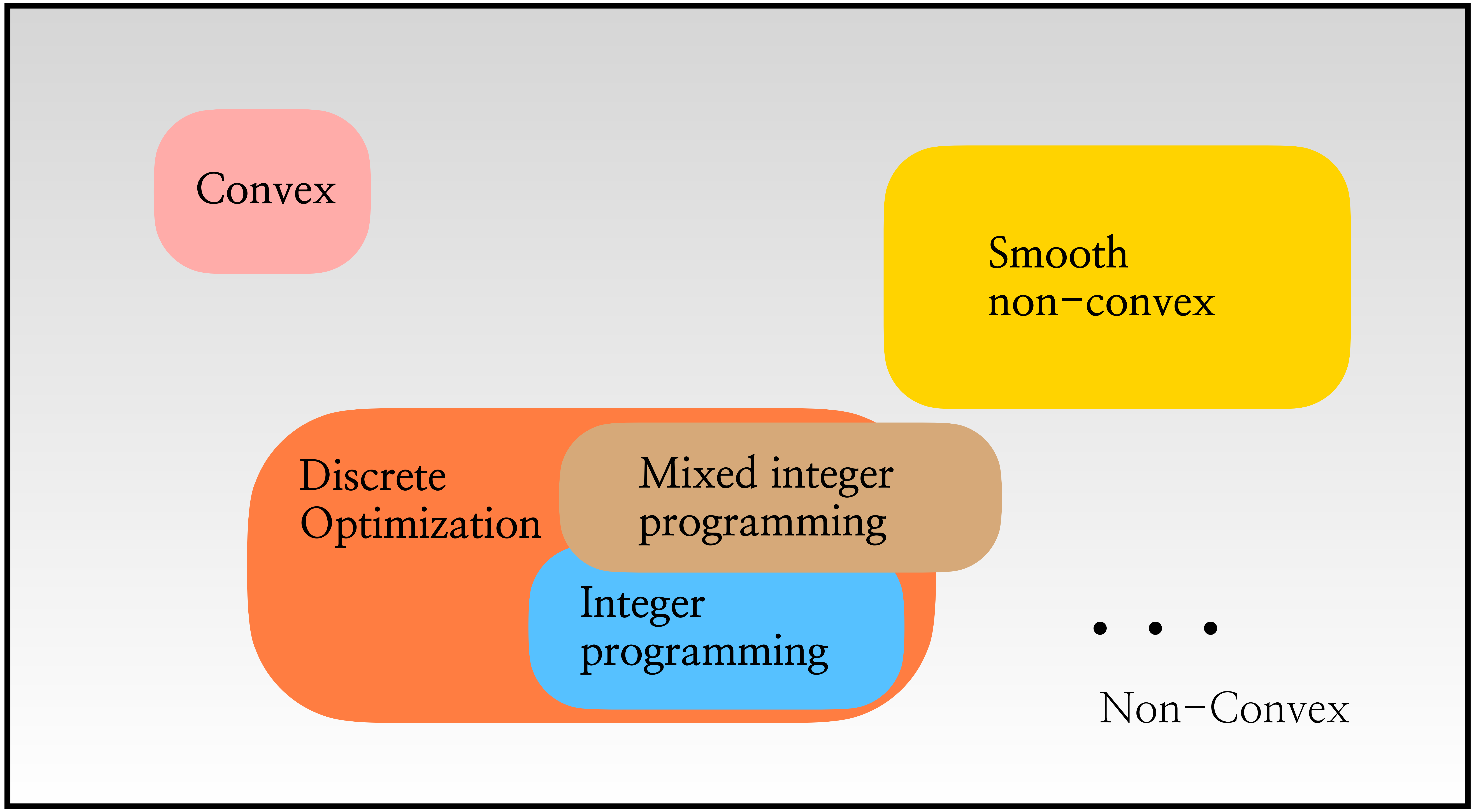
(Naive interpretation of) Space of optimization problems

Convex vs. non−convex optimization



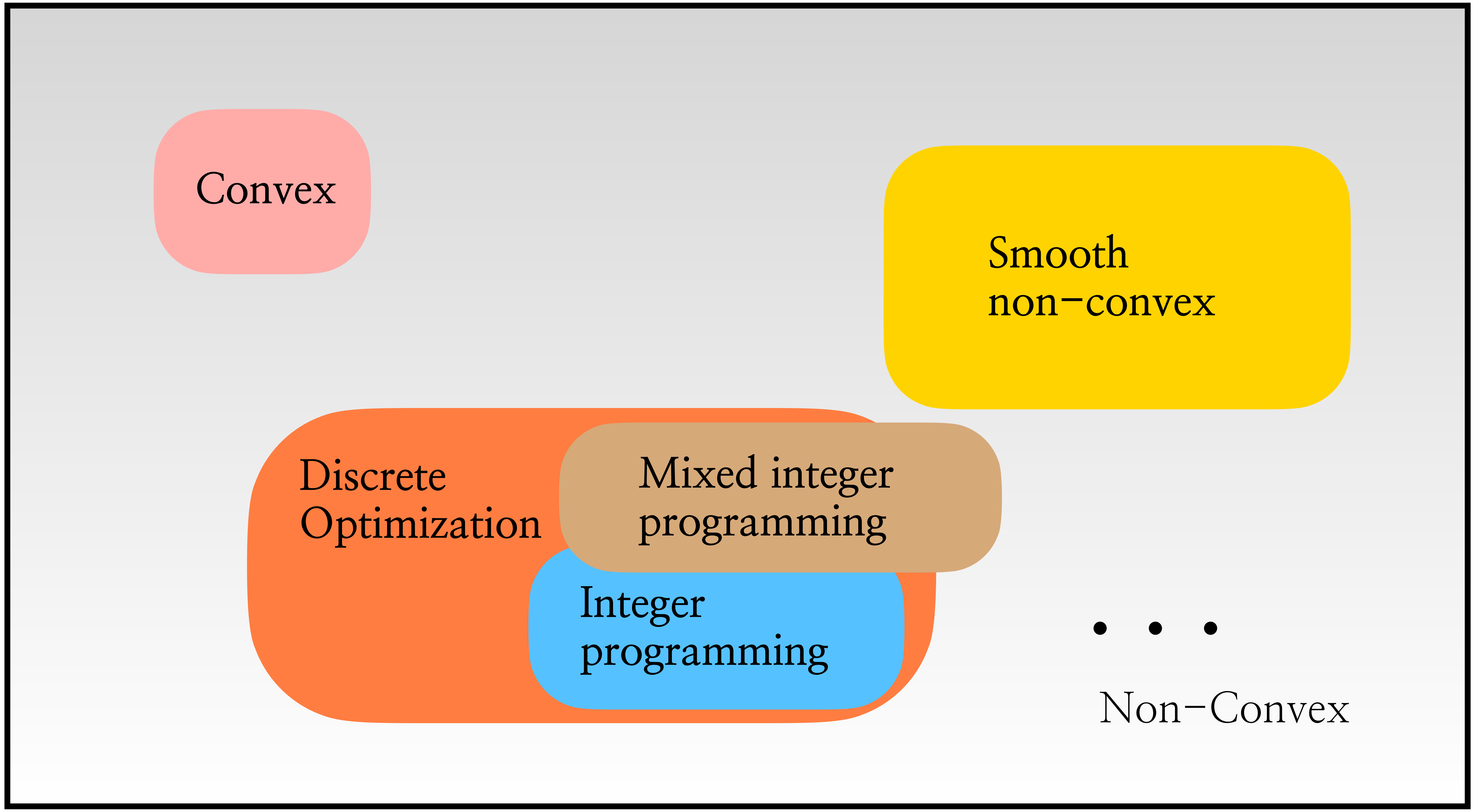
(Naive interpretation of) Space of optimization problems

Convex vs. non−convex optimization



(Naive interpretation of) Space of optimization problems

Convex vs. non−convex optimization



(Naive interpretation of) Space of optimization problems

Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \to \mathbb{R} \quad \Bigg| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

Derivatives and gradients

- Definition of a **derivative**

$$f : \mathbb{R} \rightarrow \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

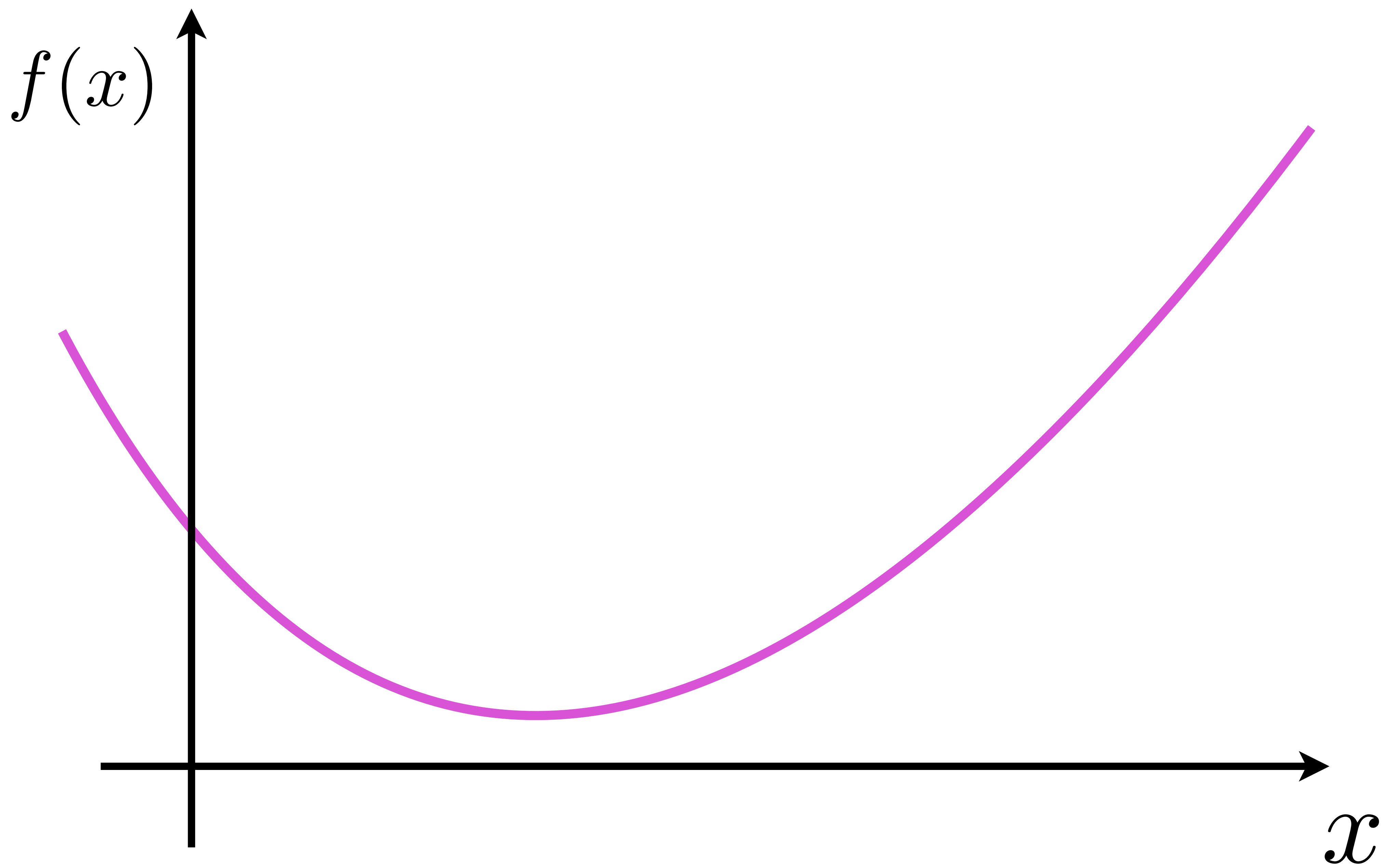
- Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \rightarrow 0$

Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

– Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \to 0$

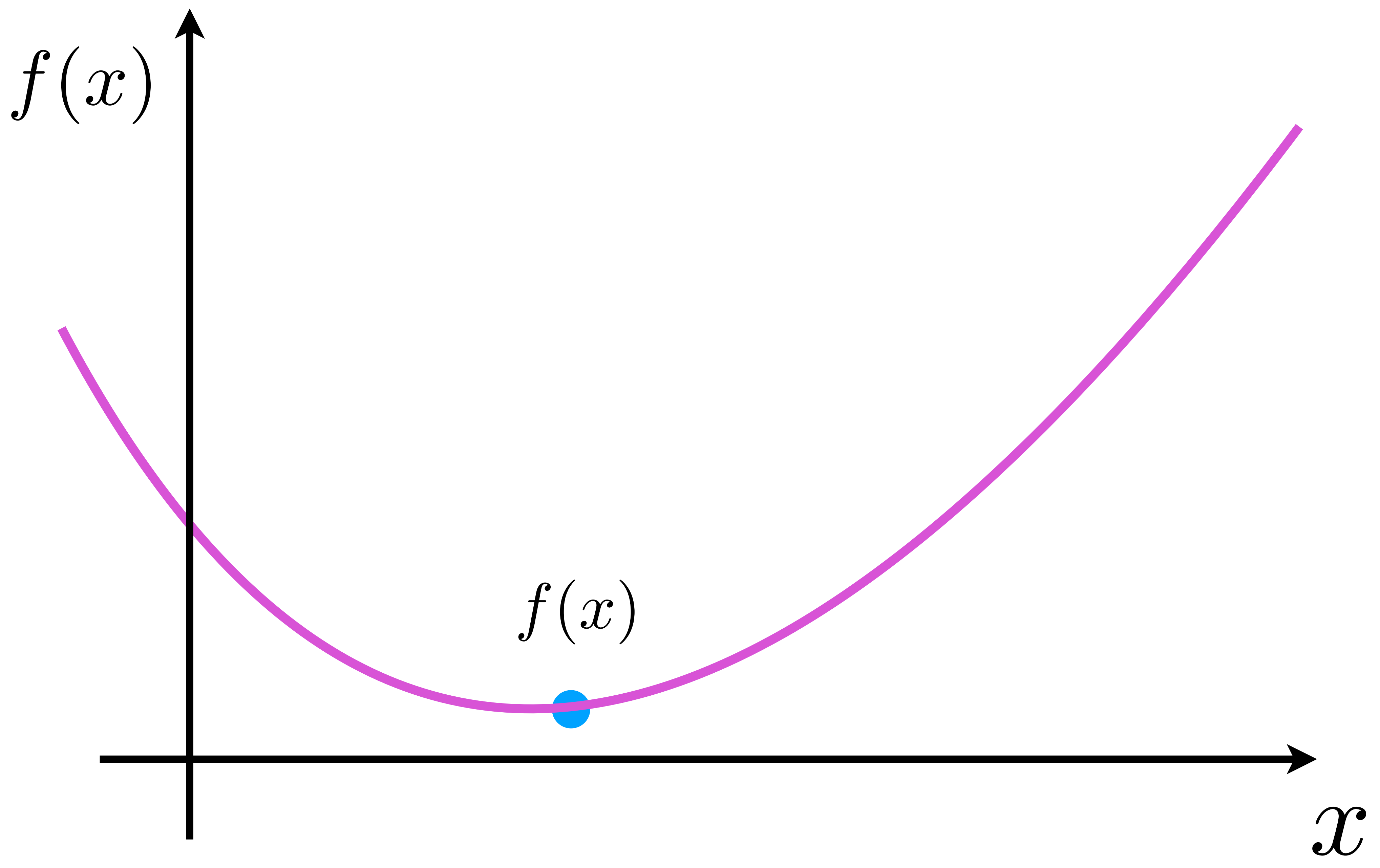


Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

– Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \to 0$

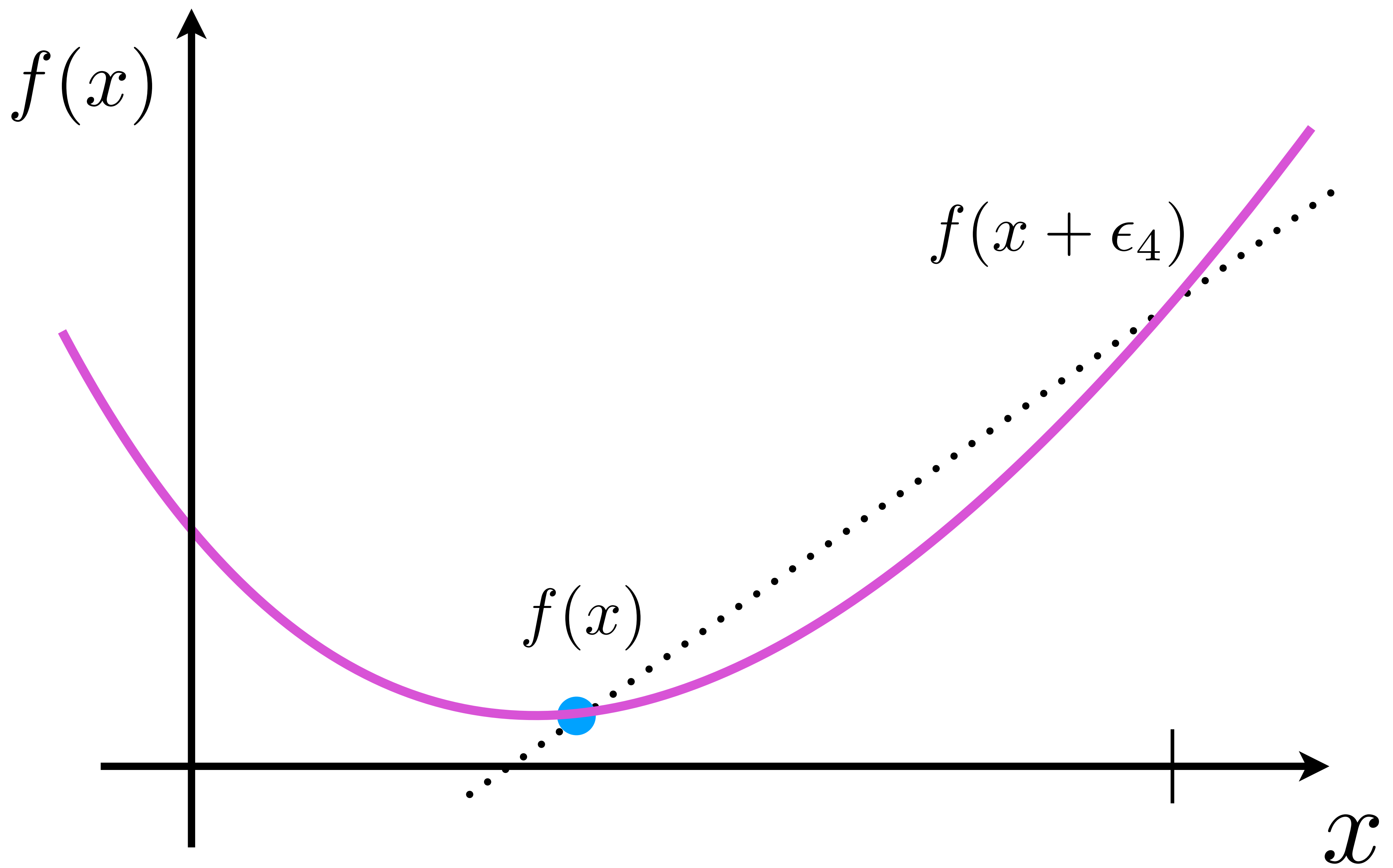


Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

– Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \to 0$

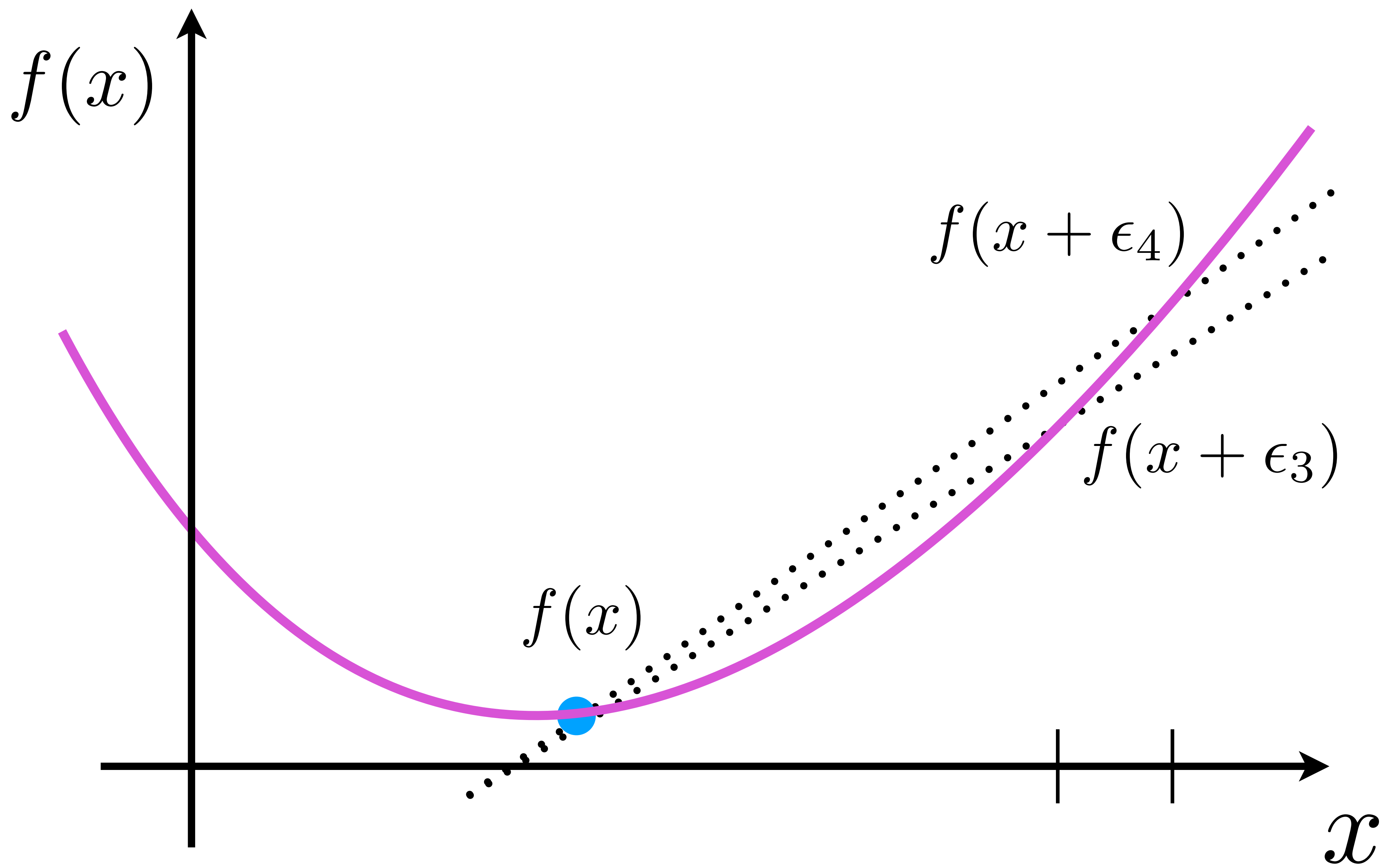


Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

– Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \to 0$

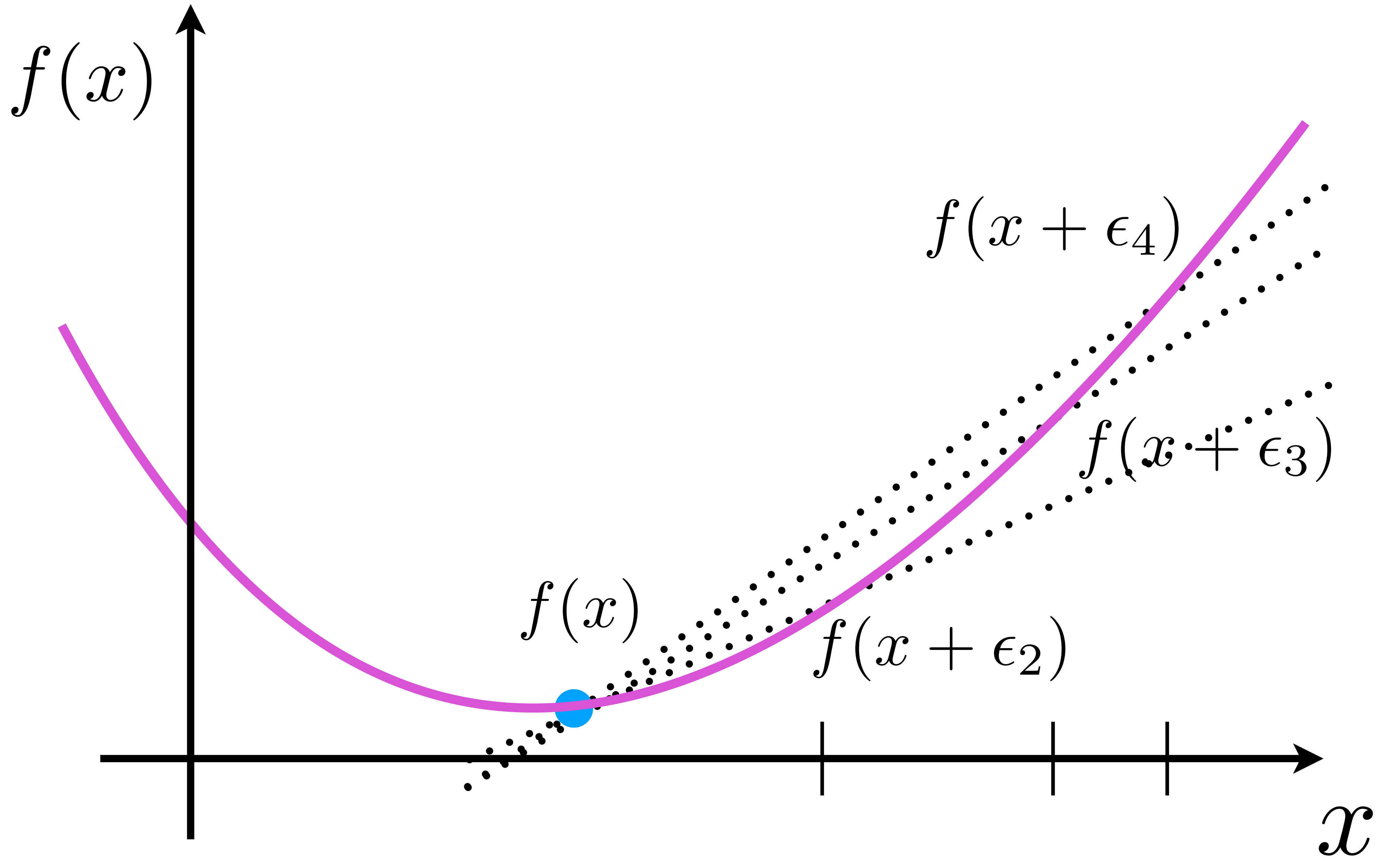


Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

– Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \to 0$

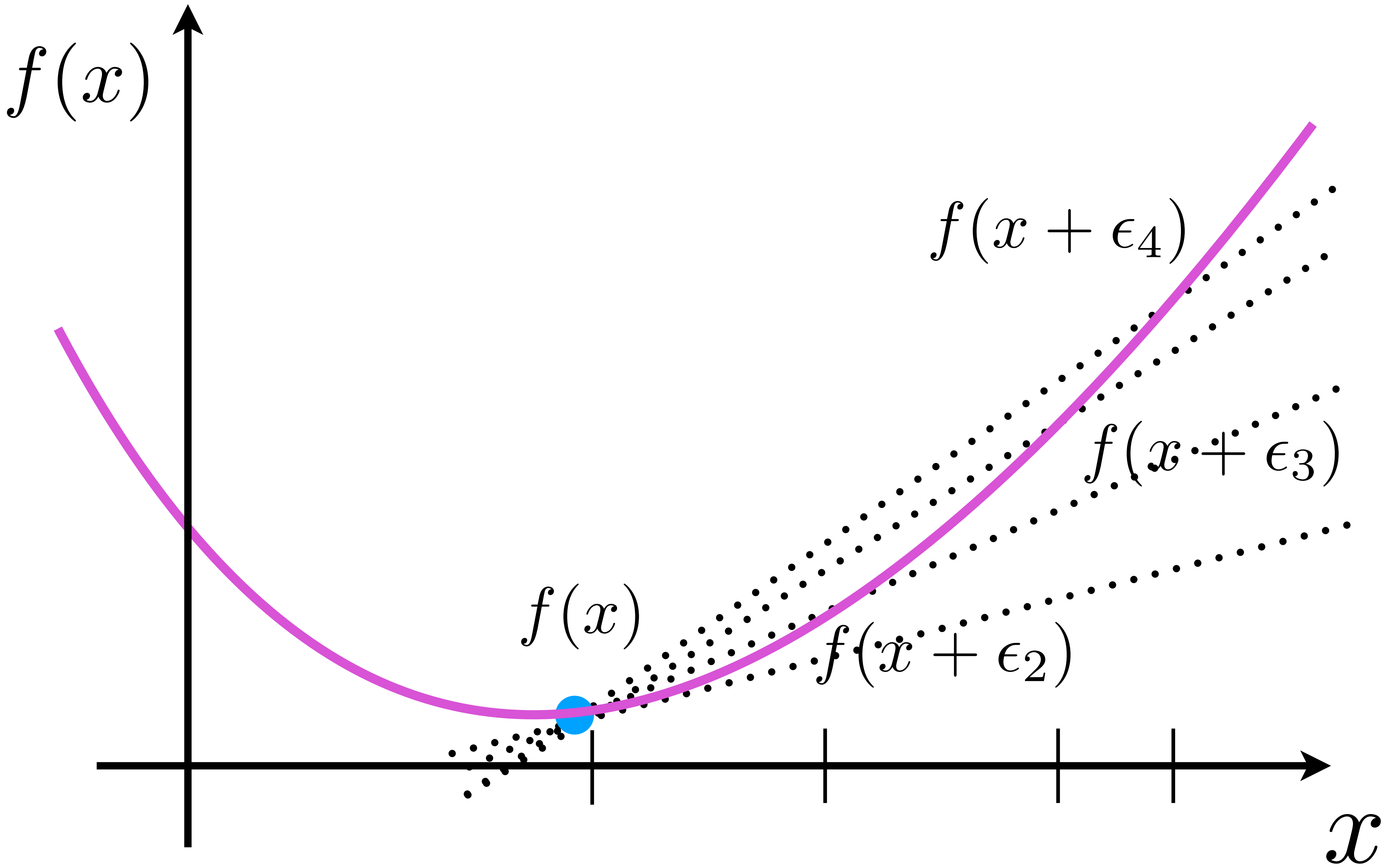


Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \rightarrow \mathbb{R} \quad \Bigg| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \rightarrow 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

– Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \rightarrow 0$

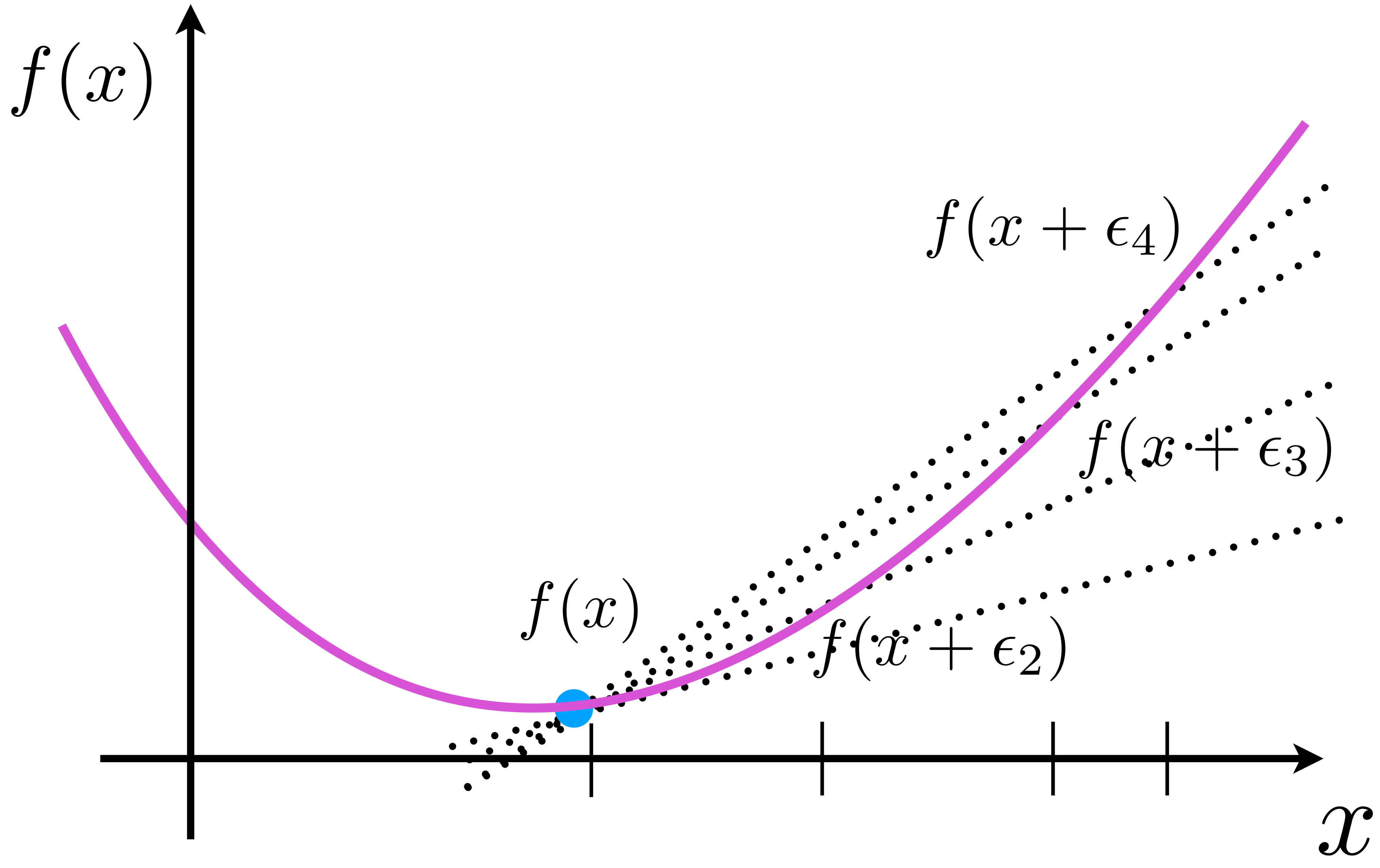


Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \rightarrow \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \rightarrow 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

– Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \rightarrow 0$



Represents the rate that function changes

Derivatives and gradients

– Definition of a **derivative**

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

– Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \to 0$

– Definition of **second-order** derivative

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial^2 f}{\partial x^2} = f''(x) = \lim_{\epsilon \to 0} \frac{f'(x+\epsilon) - f'(x)}{\epsilon}$$

Derivatives and gradients

- Definition of a **derivative**

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial f}{\partial x} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

- Intuition: generate a sequence of points $\{f(x+\epsilon_i), \epsilon_i\}$, and compute the limit as $\epsilon_i \to 0$

- Definition of **second-order** derivative

$$f : \mathbb{R} \to \mathbb{R} \quad \Big| \quad \frac{\partial^2 f}{\partial x^2} = f''(x) = \lim_{\epsilon \to 0} \frac{f'(x+\epsilon) - f'(x)}{\epsilon}$$

Represents the local curvature:
How the slope of the function changes

Derivatives and gradients

– Generalization to multiple components: **gradient**

$$f : \mathbb{R}^p \to \mathbb{R} \qquad \Big| \qquad \nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_p} \end{bmatrix} \in \mathbb{R}^p$$

where

$$\frac{\partial f}{\partial x_i} = \lim_{\epsilon \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + \epsilon, x_{i+1}, \ldots, x_p) - f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_p)}{\epsilon} = \frac{f(x + \epsilon e_i) - f(x)}{\epsilon}$$

Derivatives and gradients

– **Jacobian** matrix (relates to neural networks)

$$f : \mathbb{R}^p \to \mathbb{R}^m \quad \Big| \quad Df(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_p} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_p} \end{bmatrix} \in \mathbb{R}^{m \times p}$$

– Generalizes the notion of gradient to multiple–output functions

Derivatives and gradients

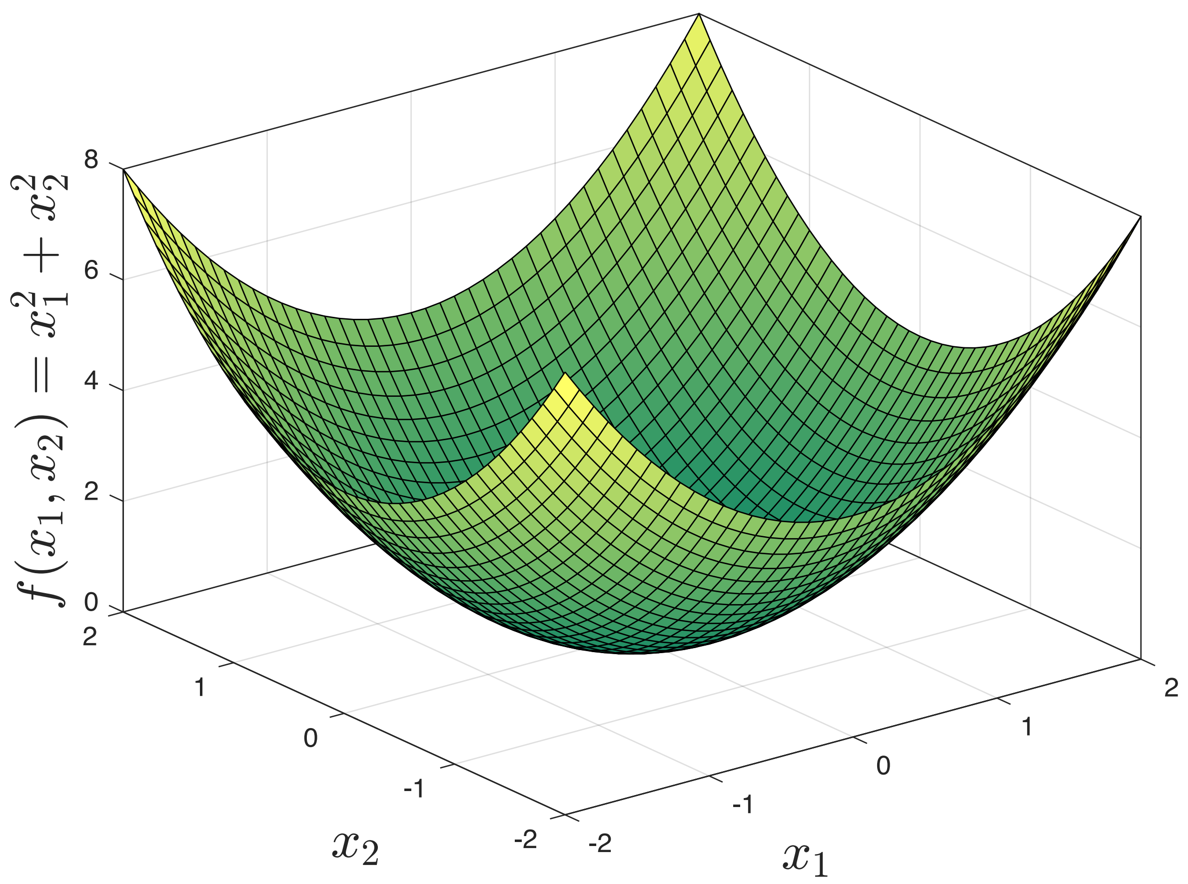
– **Hessian** matrix

$$f : \mathbb{R}^p \to \mathbb{R} \qquad \Bigg| \qquad \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_p} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_p} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_p x_1} & \frac{\partial^2 f}{\partial x_p \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_p^2} \end{bmatrix} \in \mathbb{R}^{p \times p}$$

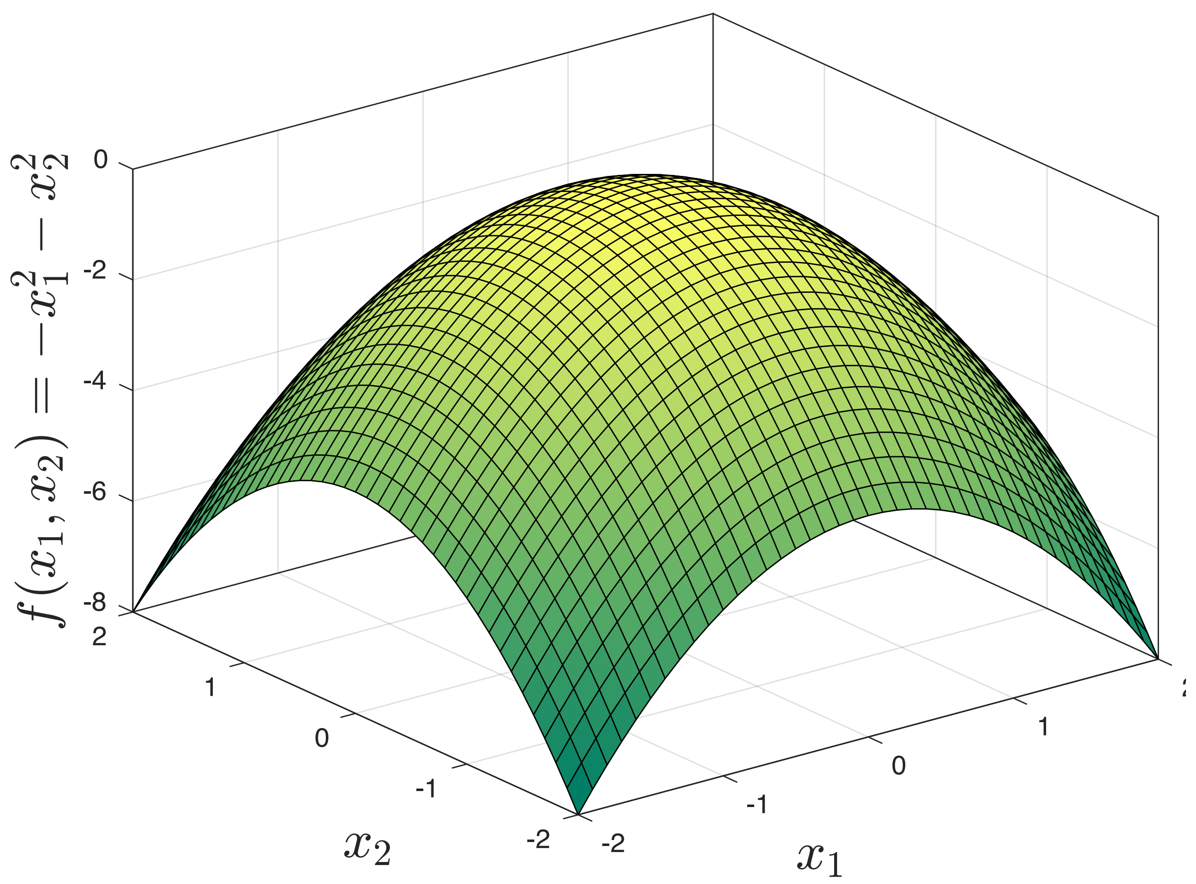
Derivatives and gradients

- **Hessian** matrix

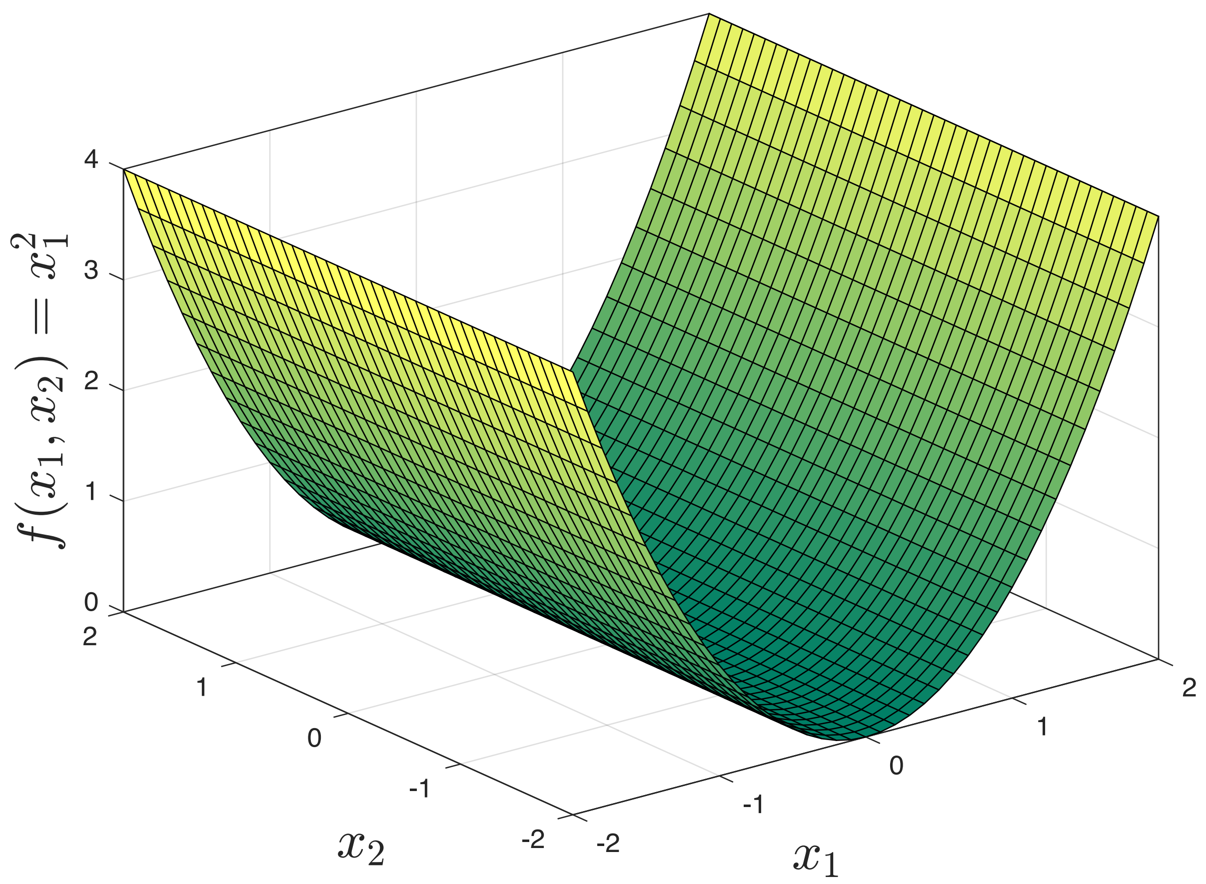
$$f : \mathbb{R}^p \to \mathbb{R} \quad \Bigg| \quad \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_p} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_p} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_p x_1} & \frac{\partial^2 f}{\partial x_p \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_p^2} \end{bmatrix} \in \mathbb{R}^{p \times p}$$



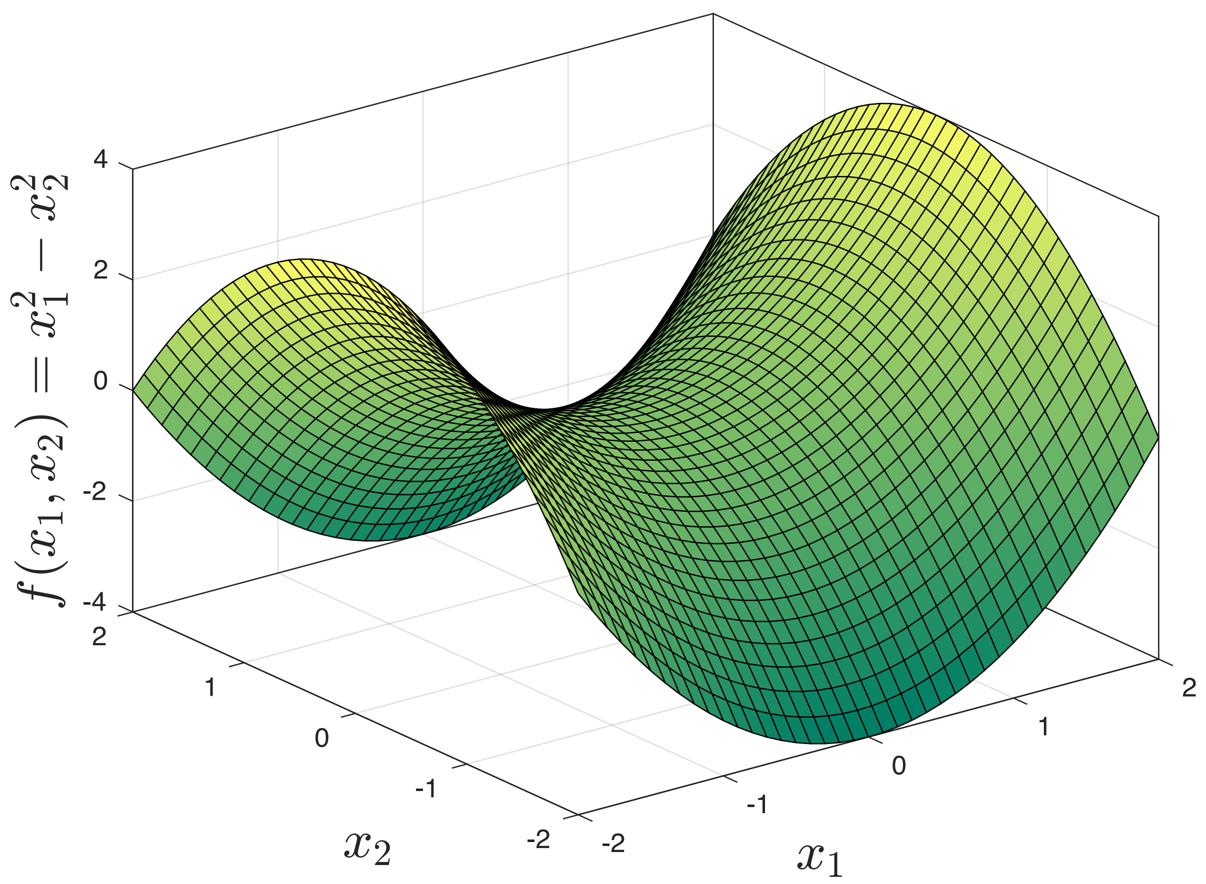
$\nabla^2 f(\cdot) \succ 0$



$\nabla^2 f(\cdot) \prec 0$



$\nabla^2 f(\cdot) \succeq 0$

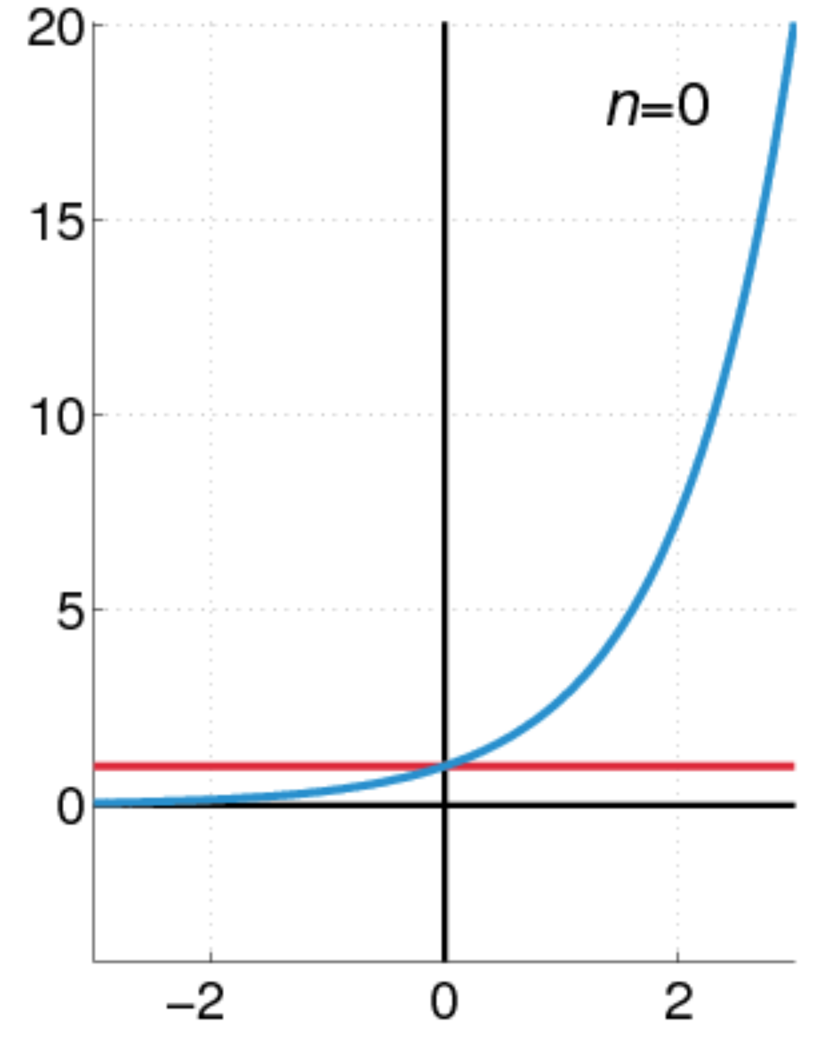


Indefinite

Taylor's expansion

– **Taylor's** expansion: used for (locally) approximating a function

$$f(x)\Big|_{x=\alpha} = f(\alpha) + f'(\alpha)(x-\alpha) + \frac{f''(a)}{2!}(x-\alpha)^2 + \cdots + \frac{f^{(n)}(\alpha)}{n!}(x-\alpha)^n + R_n$$

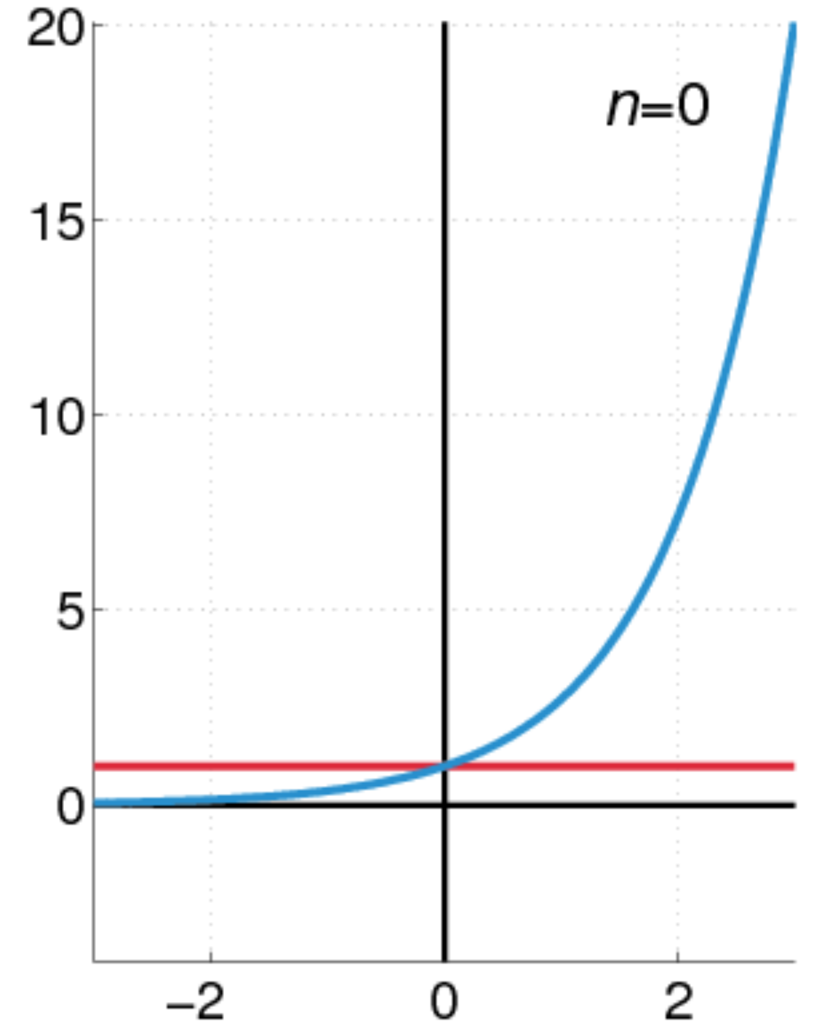


Example: exponential function

Taylor's expansion

– **Taylor's** expansion: used for (locally) approximating a function

$$f(x)\Big|_{x=\alpha} = f(\alpha) + f'(\alpha)(x-\alpha) + \frac{f''(a)}{2!}(x-\alpha)^2 + \cdots + \frac{f^{(n)}(\alpha)}{n!}(x-\alpha)^n + R_n$$

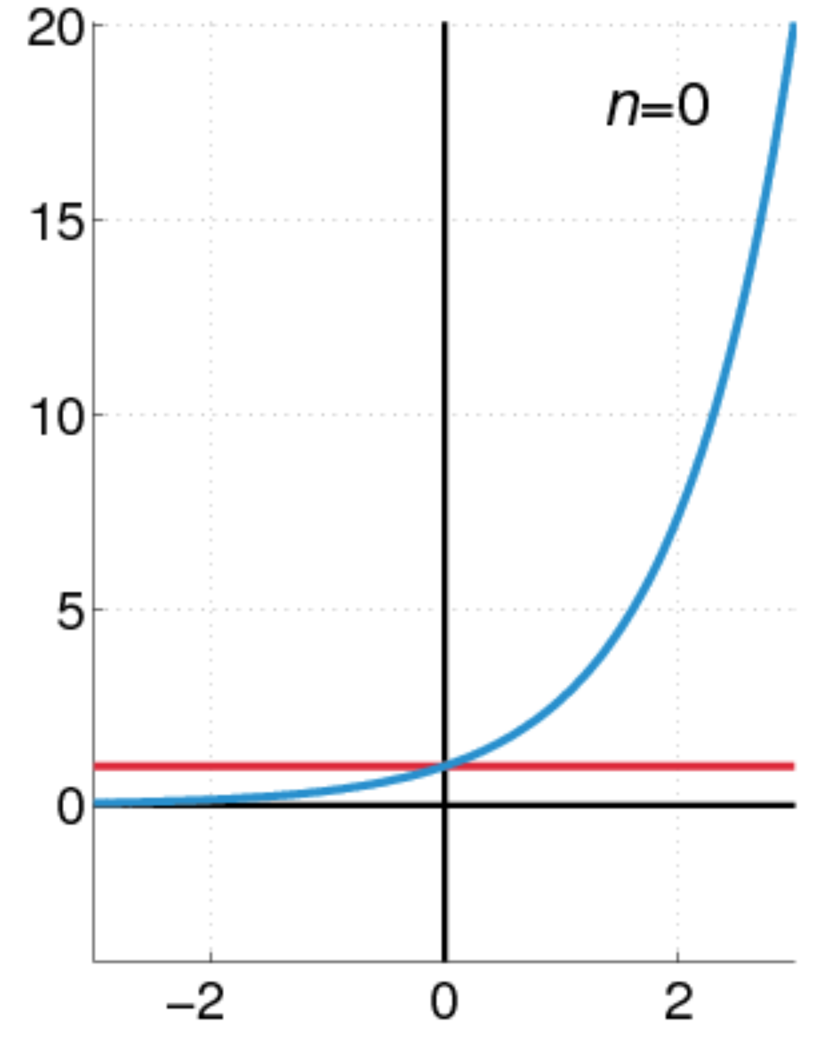


Example:
exponential function

Taylor's expansion

- **Taylor's** expansion: used for (locally) approximating a function

$$f(x)\Big|_{x=\alpha} = f(\alpha) + f'(\alpha)(x-\alpha) + \frac{f''(\alpha)}{2!}(x-\alpha)^2 + \cdots + \frac{f^{(n)}(\alpha)}{n!}(x-\alpha)^n + R_n$$



Example: exponential function

- Key properties/assumptions:
  - Function f is differentiable as many times we'd like
  - Provides **(locally)** a good approximation of the function

Taylor's expansion

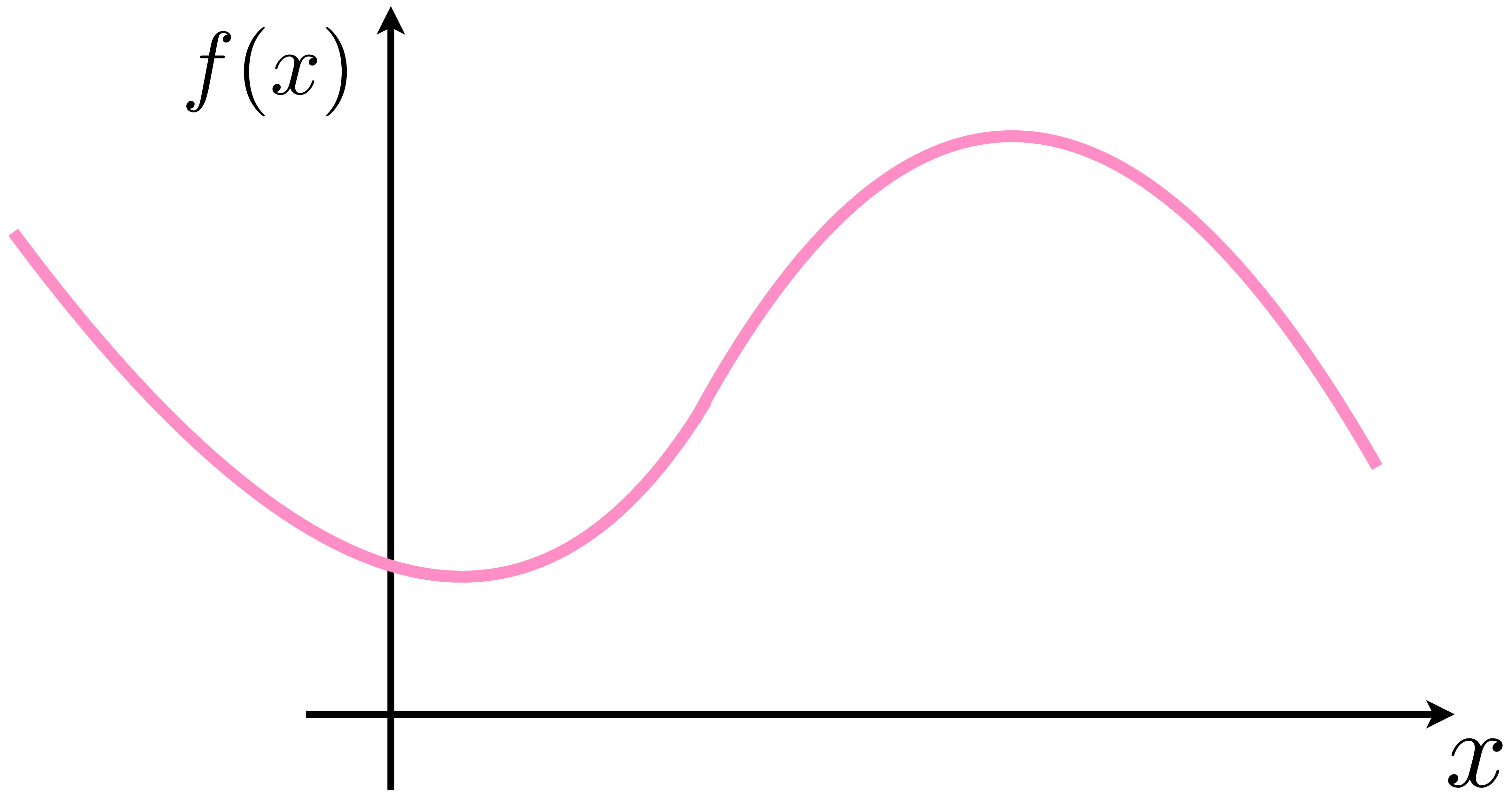
- First–order Taylor's approximation

$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle, \alpha \in \mathbb{R}^p$$

Taylor's expansion

- First–order Taylor's approximation

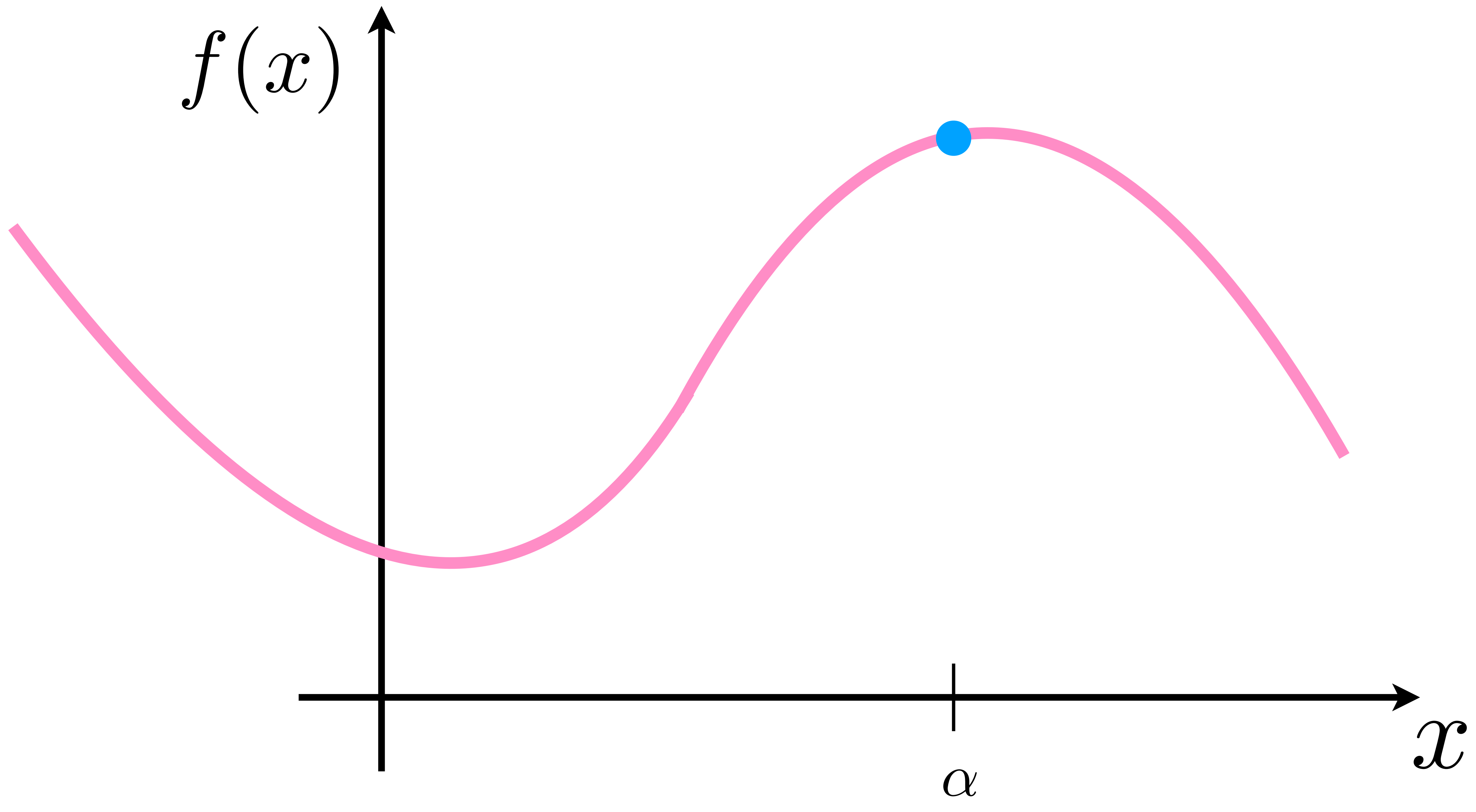
$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle , \alpha \in \mathbb{R}^p$$



Taylor's expansion

- First-order Taylor's approximation

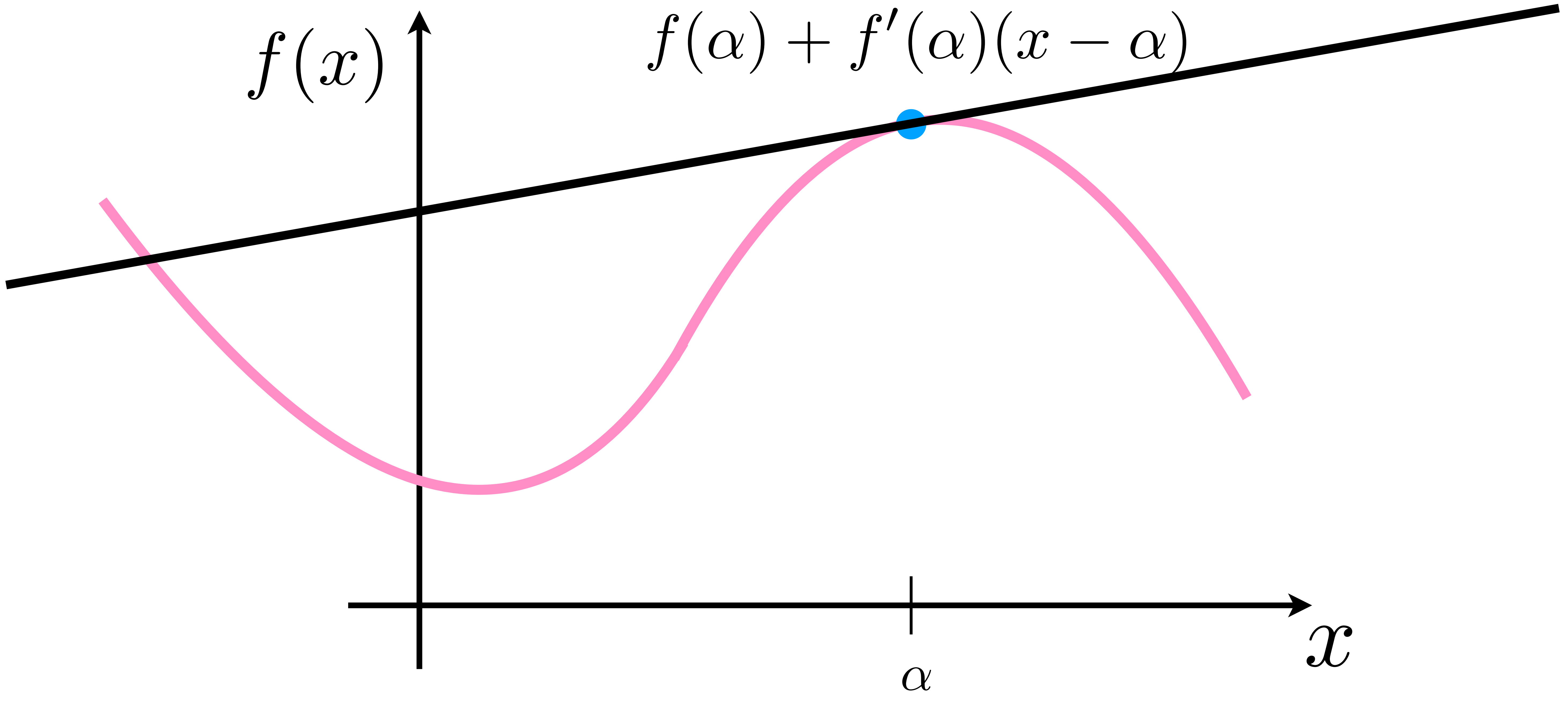
$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle \,, \alpha \in \mathbb{R}^p$$



Taylor's expansion

- First-order Taylor's approximation

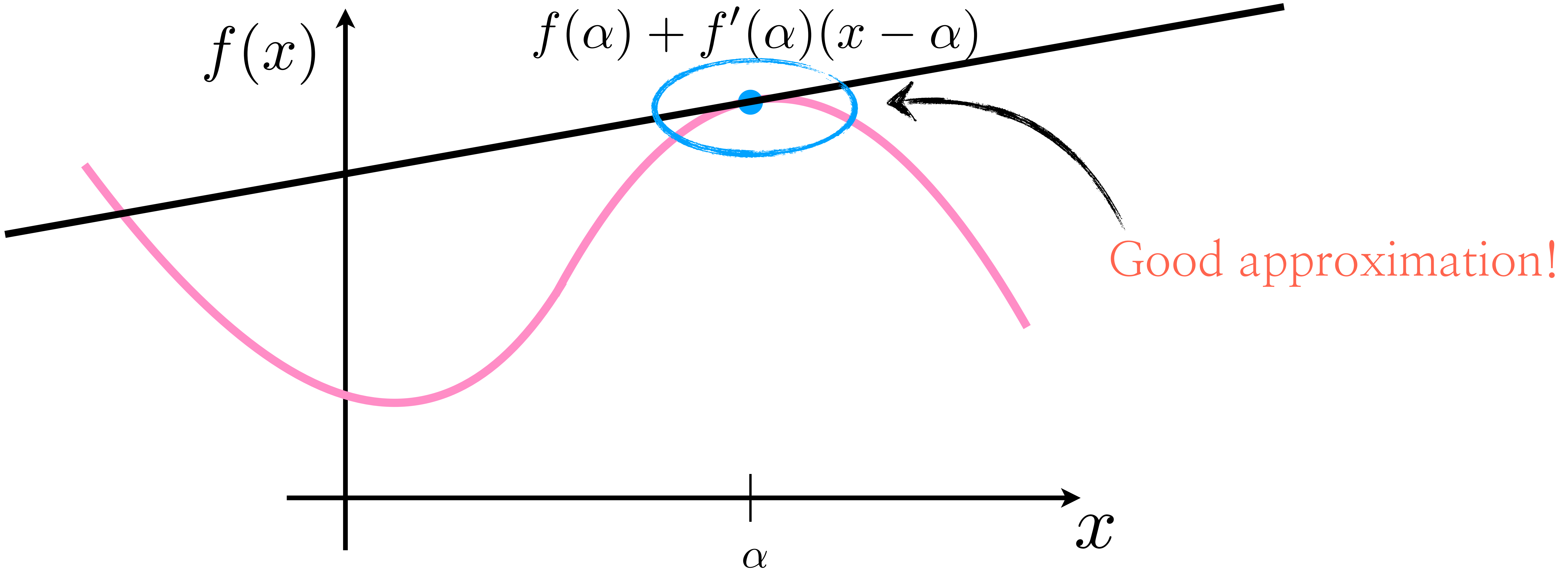
$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle, \alpha \in \mathbb{R}^p$$



Taylor's expansion

– First−order Taylor's approximation

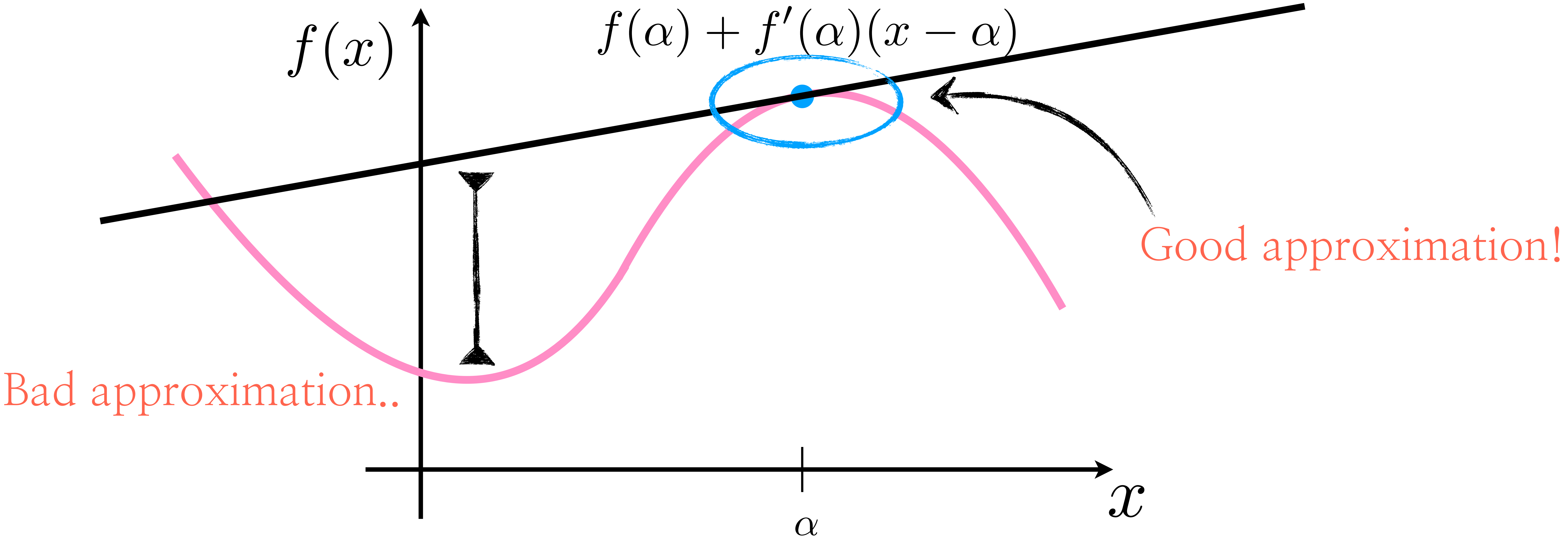
$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle \,, \alpha \in \mathbb{R}^p$$



Taylor's expansion

- First−order Taylor's approximation

$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle \,, \alpha \in \mathbb{R}^p$$



Taylor's expansion

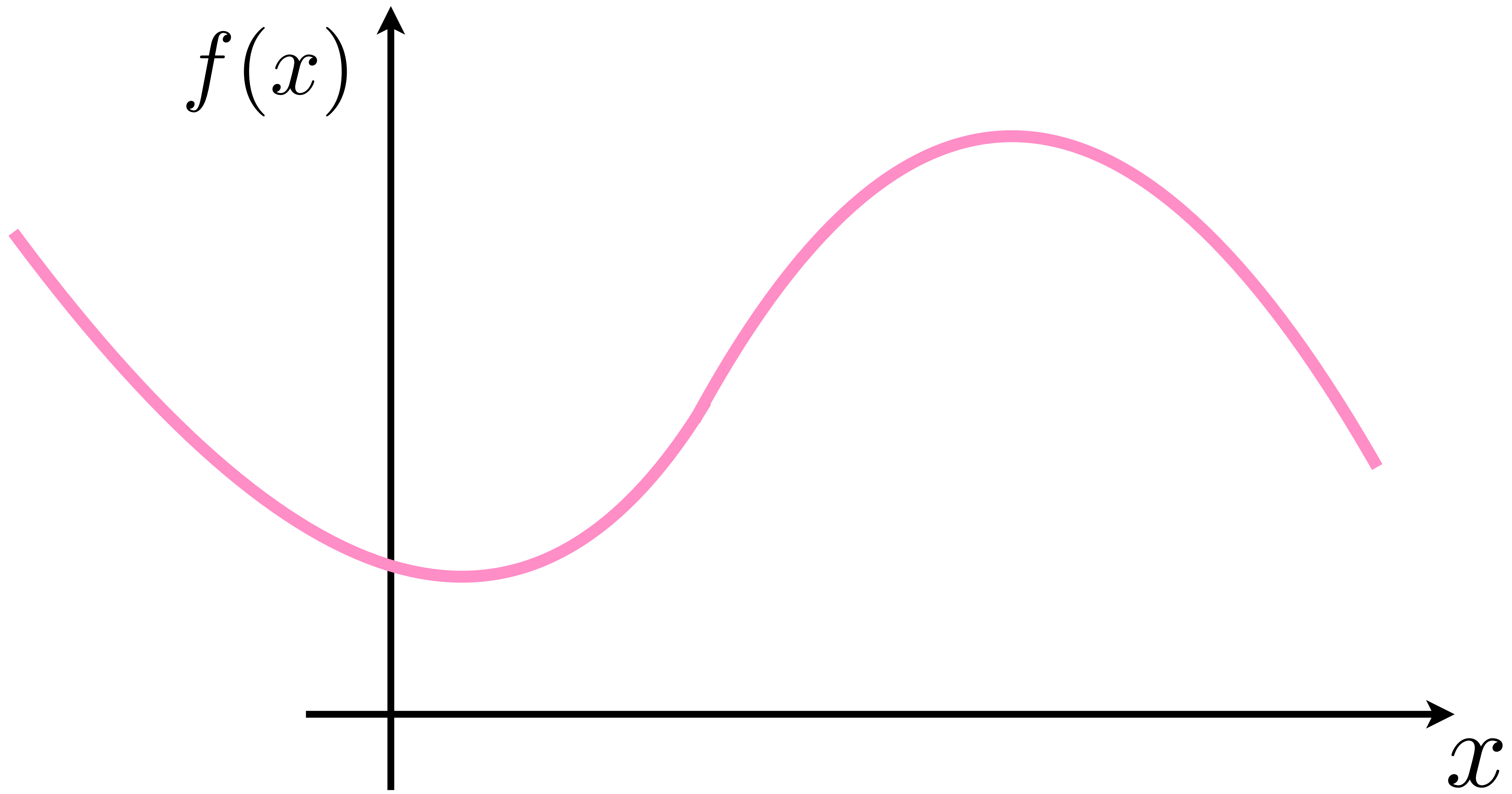
- Second-order Taylor's approximation

$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle + \frac{1}{2} \left\langle \nabla^2 f(\alpha)(x - \alpha), x - \alpha \right\rangle, \alpha \in \mathbb{R}^p$$

Taylor's expansion

– Second–order Taylor's approximation

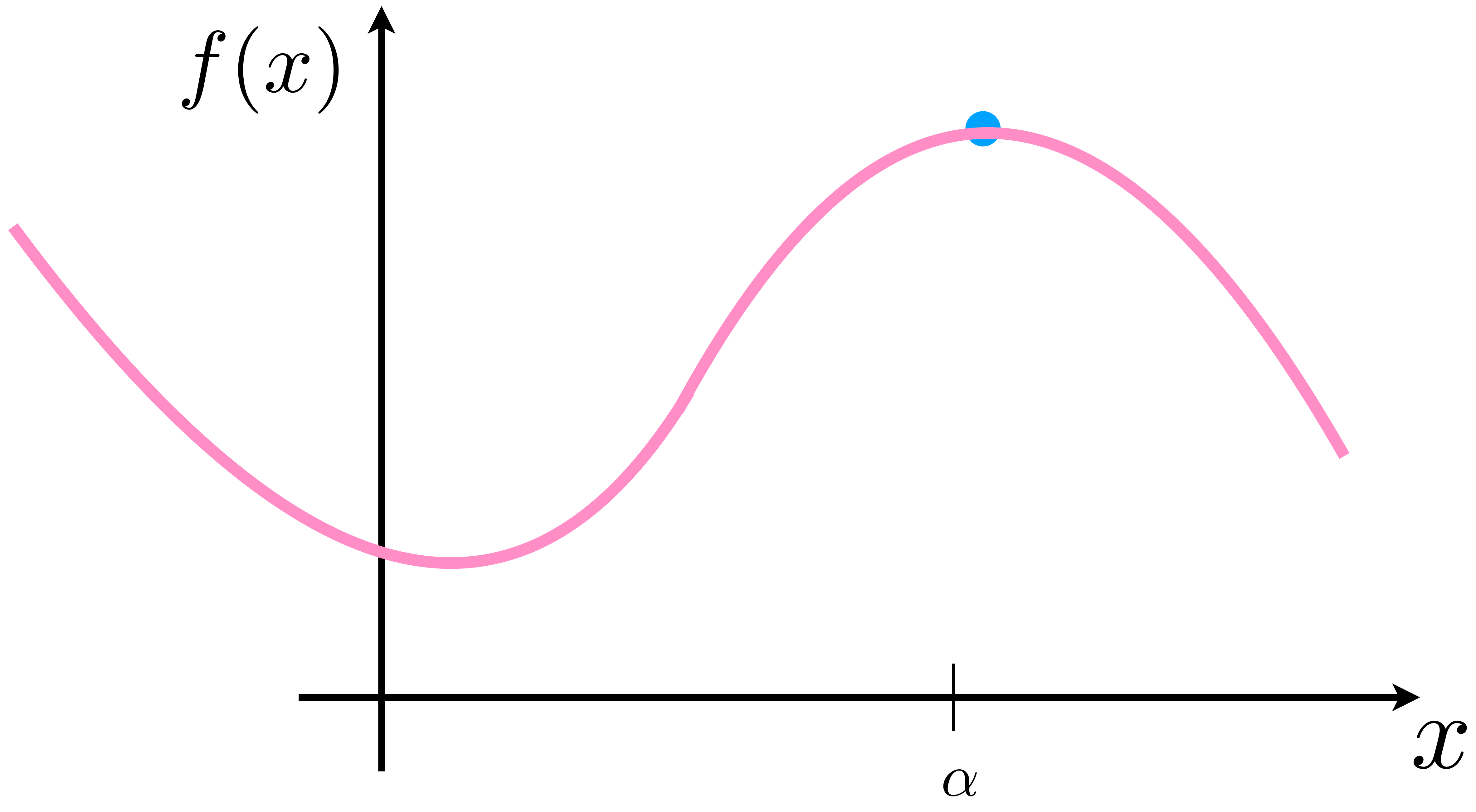
$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle + \frac{1}{2} \langle \nabla^2 f(\alpha)(x - \alpha), x - \alpha \rangle, \alpha \in \mathbb{R}^p$$



Taylor's expansion

- Second-order Taylor's approximation

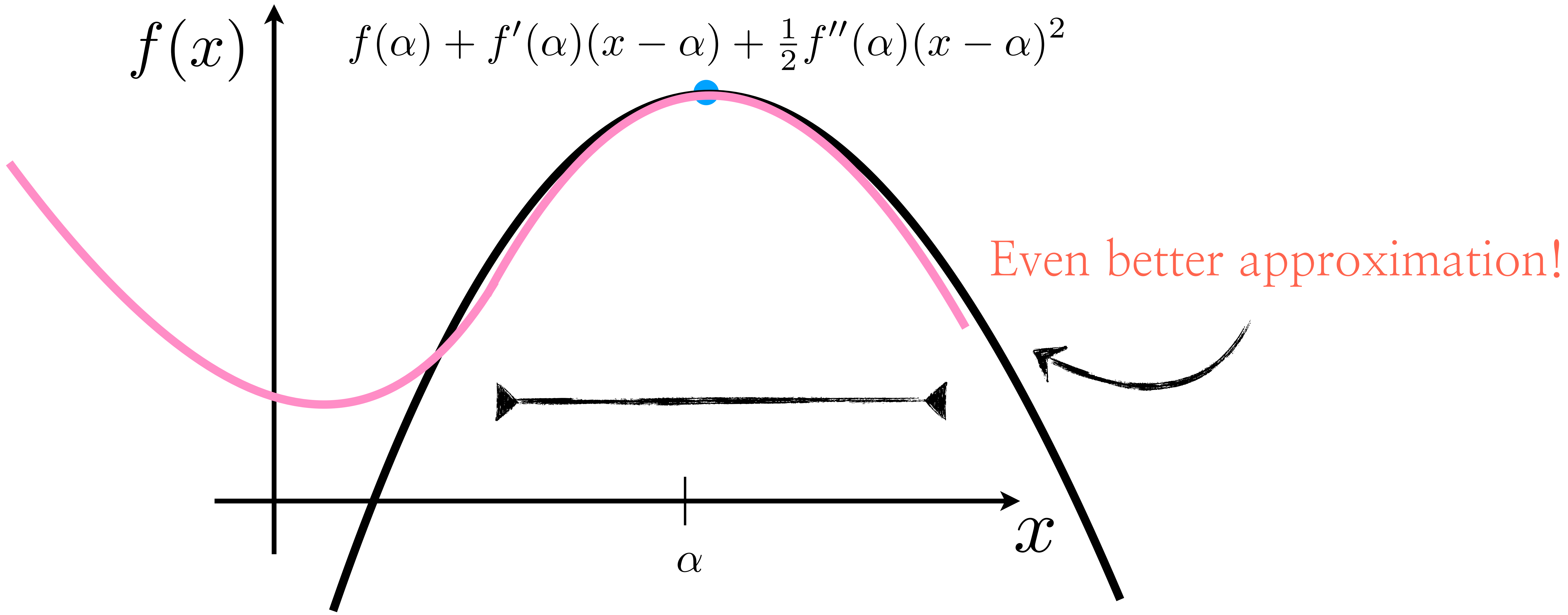
$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle + \frac{1}{2} \langle \nabla^2 f(\alpha)(x - \alpha), x - \alpha \rangle, \alpha \in \mathbb{R}^p$$



Taylor's expansion

- Second–order Taylor's approximation

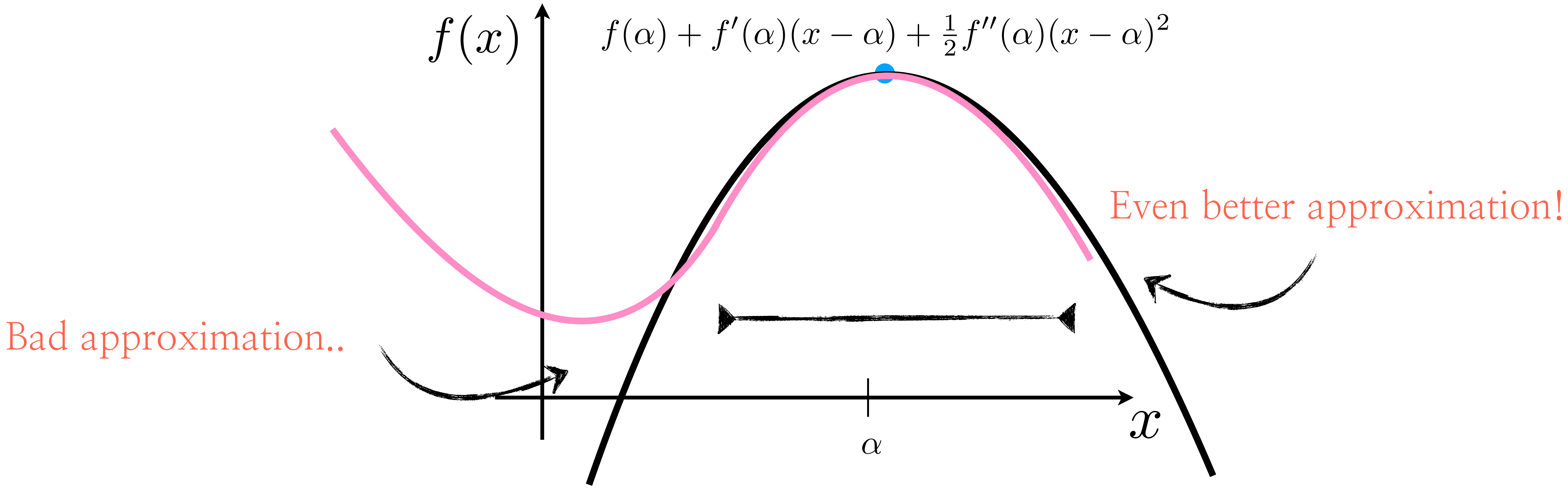
$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle + \frac{1}{2} \langle \nabla^2 f(\alpha)(x-\alpha), x - \alpha \rangle, \alpha \in \mathbb{R}^p$$



Taylor's expansion

– Second–order Taylor's approximation

$$f : \mathbb{R}^p \to \mathbb{R} \quad \Big| \quad f(x) \approx f(\alpha) + \langle \nabla f(\alpha), x - \alpha \rangle + \tfrac{1}{2} \langle \nabla^2 f(\alpha)(x - \alpha), x - \alpha \rangle, \alpha \in \mathbb{R}^p$$



Taylor's expansion

- Spoiler alert: "Why are all these useful?"
  - Often, we optimize a function through its local approximations
  - E.g., second order approximations are.. quadratic functions!

Taylor's expansion

- Spoiler alert: "Why are all these useful?"
  - Often, we optimize a function through its local approximations
  - E.g., second order approximations are.. quadratic functions!

$$\min_x f(x)$$

Taylor's expansion

- Spoiler alert: "Why are all these useful?"
  - Often, we optimize a function through its local approximations
  - E.g., second order approximations are.. quadratic functions!

Weirdest function ever
(but differentiable)

$$\min_x f(x)$$

Taylor's expansion

- Spoiler alert: "Why are all these useful?"
  - Often, we optimize a function through its local approximations
  - E.g., second order approximations are.. quadratic functions!

Weirdest function ever
(but differentiable)

$$\min_x f(x) \longrightarrow \min_x \left\{ f(x_0) + \nabla f(x_0)^\top (x - x_0) + \frac{1}{2}(x - x_0)^\top \nabla^2 f(x_0)(x - x_0) \right\}$$

Taylor's expansion

- Spoiler alert: "Why are all these useful?"
  - Often, we optimize a function through its local approximations
  - E.g., second order approximations are.. quadratic functions!

Weirdest function ever
(but differentiable)

$$\min_x f(x) \longrightarrow \min_x \{p^\top x + \frac{1}{2}x^\top H x\}$$

Taylor's expansion

- Spoiler alert: "Why are all these useful?"
  - Often, we optimize a function through its local approximations
  - E.g., second order approximations are.. quadratic functions!

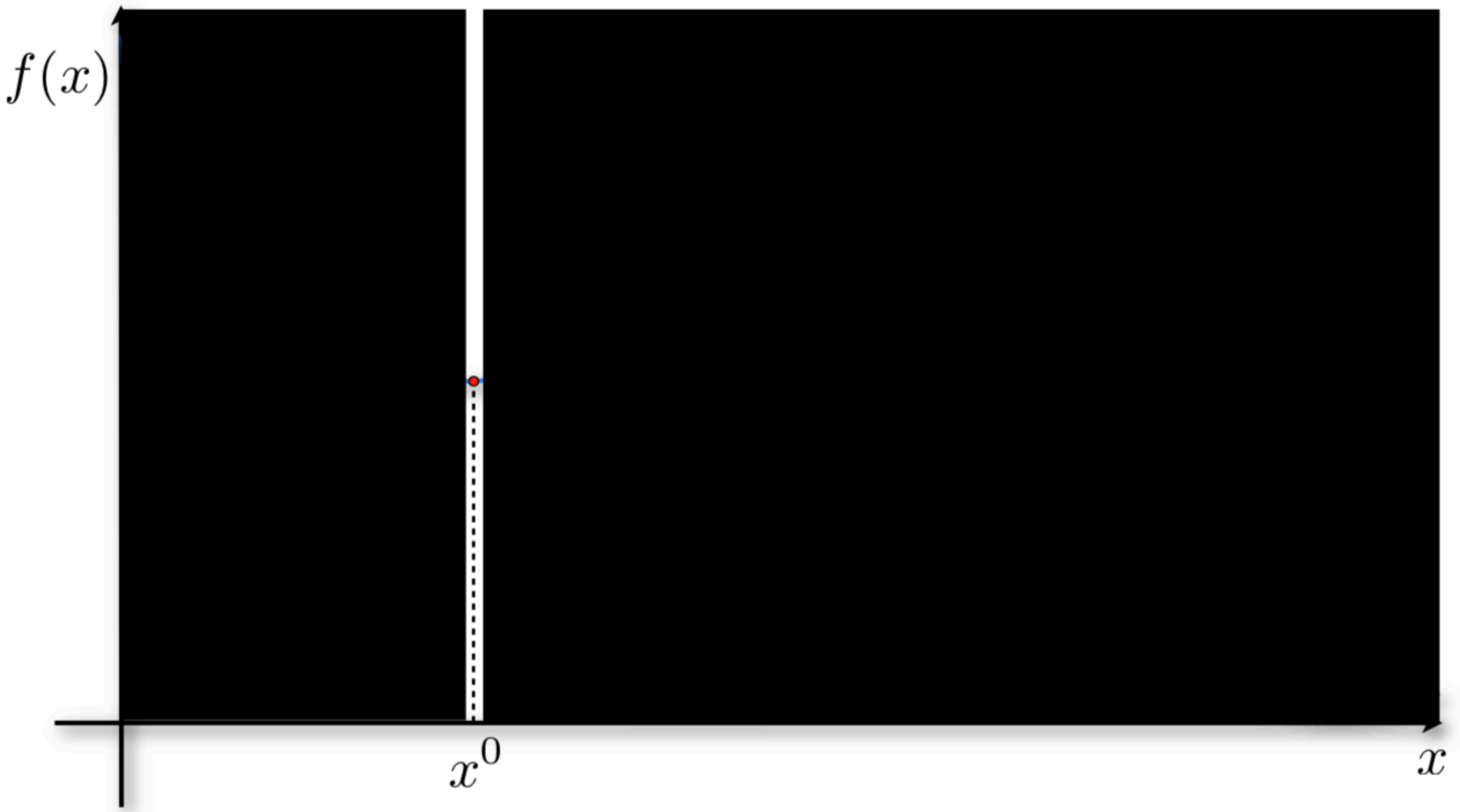
Weirdest function ever
(but differentiable)

$$\min_x f(x) \longrightarrow \min_x \{p^\top x + \tfrac{1}{2}x^\top Hx\} \longrightarrow \dots$$

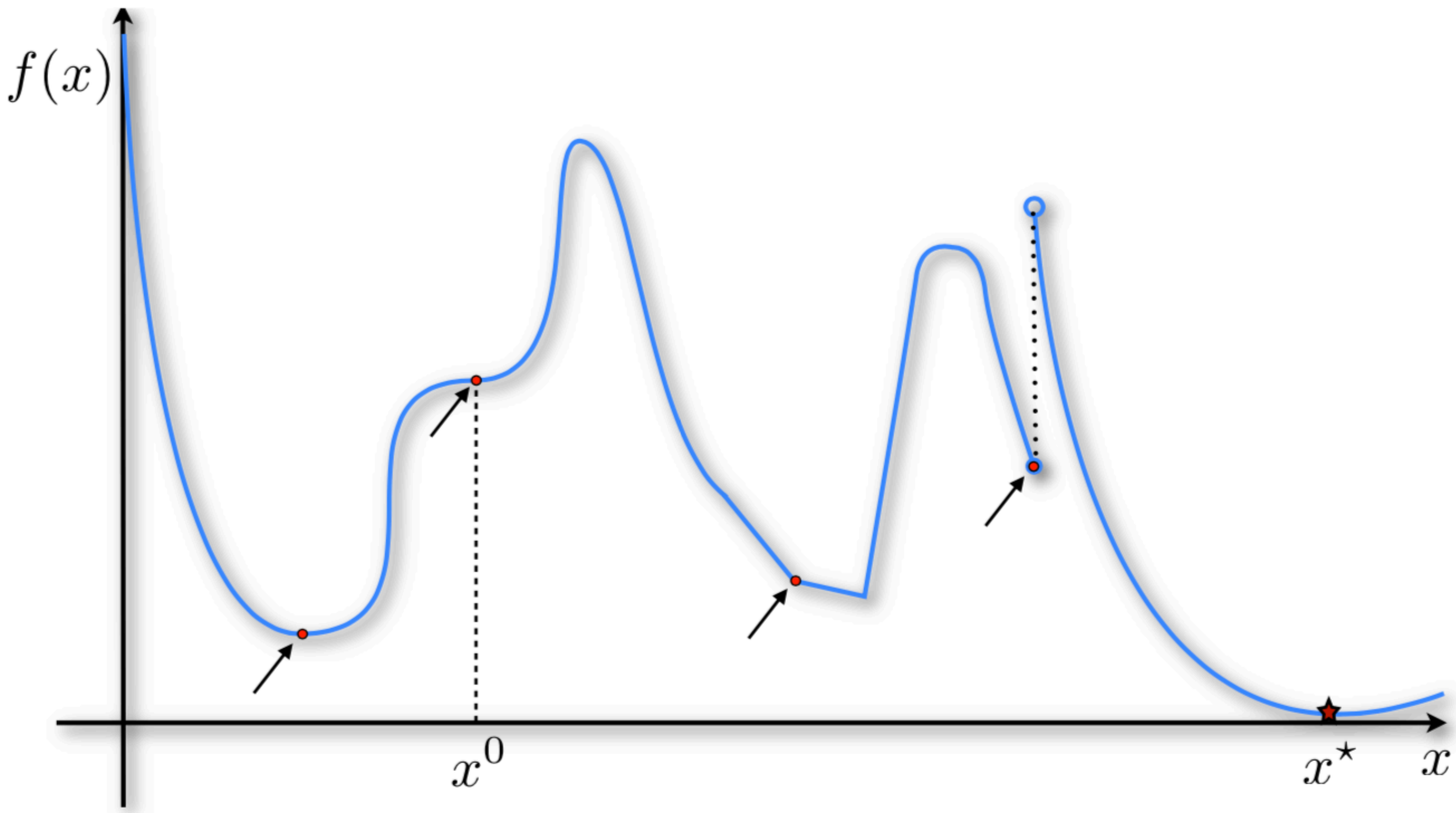
Agnostic optimization

Demo

Agnostic optimization



Agnostic optimization



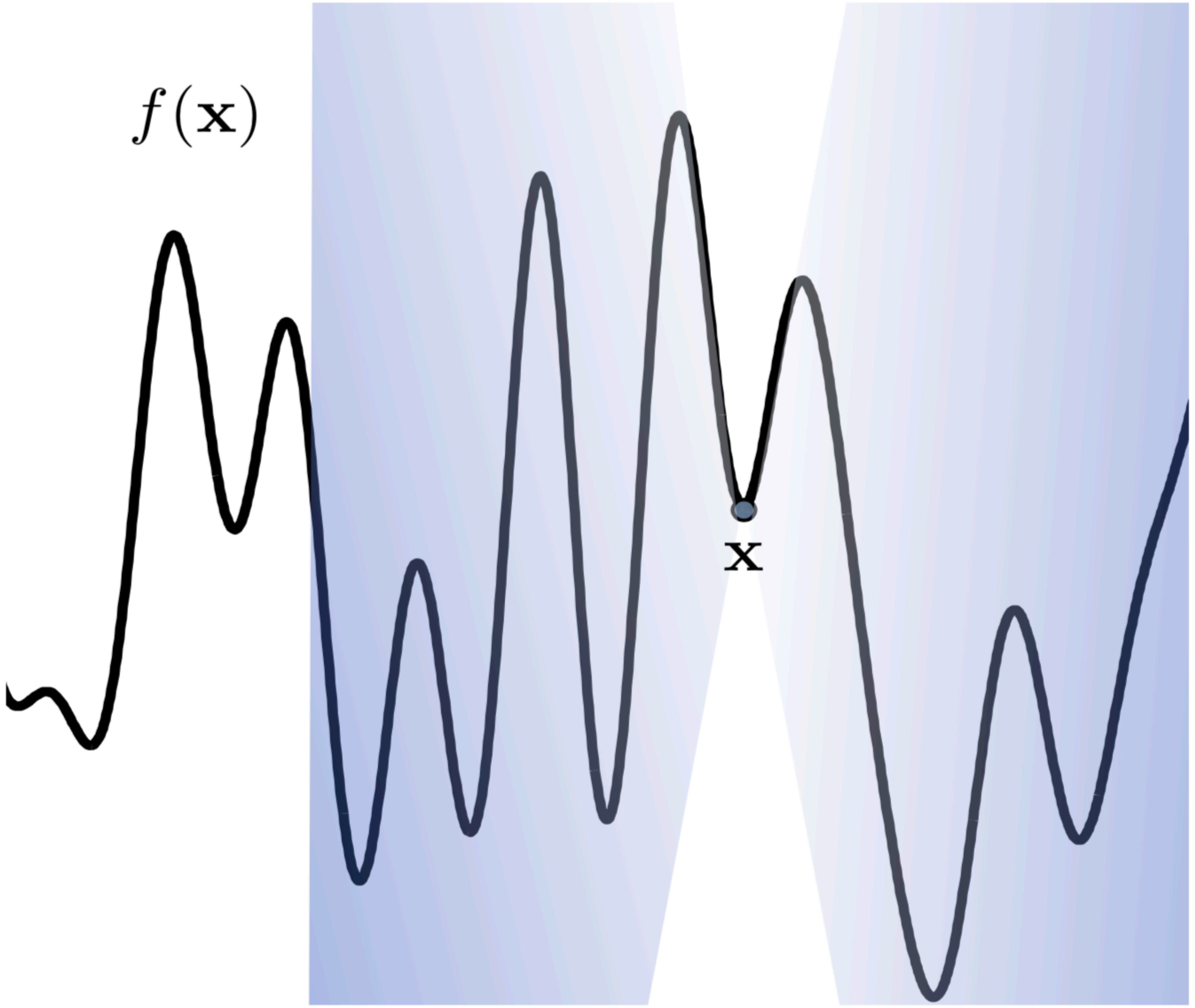
Lipschitz conditions

- **Lipschitz continuity**: $|f(x) - f(y)| \leq M\|x - y\|_2, \quad \forall x, y$

Lipschitz conditions

- **Lipschitz continuity**: $|f(x) - f(y)| \leq M\|x - y\|_2, \quad \forall x, y$

Interpretation via double cone



Function examples:
1. Absolute value
2. Trigonometric functions
3. Quadratics (..)

Lipschitz conditions

- **Lipschitz gradient continuity**： $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

Lipschitz conditions

- **Lipschitz gradient continuity**: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$
  - Intuition + comparison with Lipschitz continuity:

    "Lipschitz continuity implies that f should not be too steep"

    "Lipschitz gradient continuity implies that changes in the slope of f should not happen suddenly"

Lipschitz conditions

- **Lipschitz gradient continuity**: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

  - Intuition + comparison with Lipschitz continuity:

    "Lipschitz continuity implies that f should not be too steep"

    "Lipschitz gradient continuity implies that changes in the slope of f should not happen suddenly"

  - Example: Quadratics are not **globally** Lipschitz continuous

    (the function becomes arbitrarily steep as we approach infinity)

    but: $\|\nabla f(x) - \nabla f(y)\|_2 \leq \|A^\top A\|_2 \cdot \|x - y\|_2$

    for: $f(x) = \frac{1}{2}\|Ax - b\|_2^2$

Lipschitz conditions

- **Lipschitz gradient continuity**: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$
  - Intuition + comparison with Lipschitz continuity:

    "Lipschitz continuity implies that f should not be too steep"

    "Lipschitz gradient continuity implies that changes in the slope of f should not happen suddenly"

  - Example: Quadratics are not **globally** Lipschitz continuous

    (the function becomes arbitrarily steep as we approach infinity)

    but: $\|\nabla f(x) - \nabla f(y)\|_2 \leq \|A^\top A\|_2 \cdot \|x - y\|_2$

    for: $f(x) = \frac{1}{2}\|Ax - b\|_2^2$

    Largest singular value

Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

(some require convexity – to be defined later)

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|_2^2$$

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \leq L\|x - y\|_2^2$$

$$\vdots \qquad \vdots$$

Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

(some require convexity – to be defined later)

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|_2^2$$

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \leq L\|x - y\|_2^2$$

$$\vdots \qquad \vdots$$

– Another important one:

$$\nabla^2 f(x) \preceq LI$$

Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

(some require convexity – to be defined later)

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|_2^2$$

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \leq L\|x - y\|_2^2$$

$$\vdots \qquad \vdots$$

– Another important one:

$$\nabla^2 f(x) \preceq LI$$

Interpretation?

Lipschitz conditions

- Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

(some require convexity – to be defined later)

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|_2^2$$

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \leq L\|x - y\|_2^2$$

$$\vdots \quad \vdots$$

- Another important one:

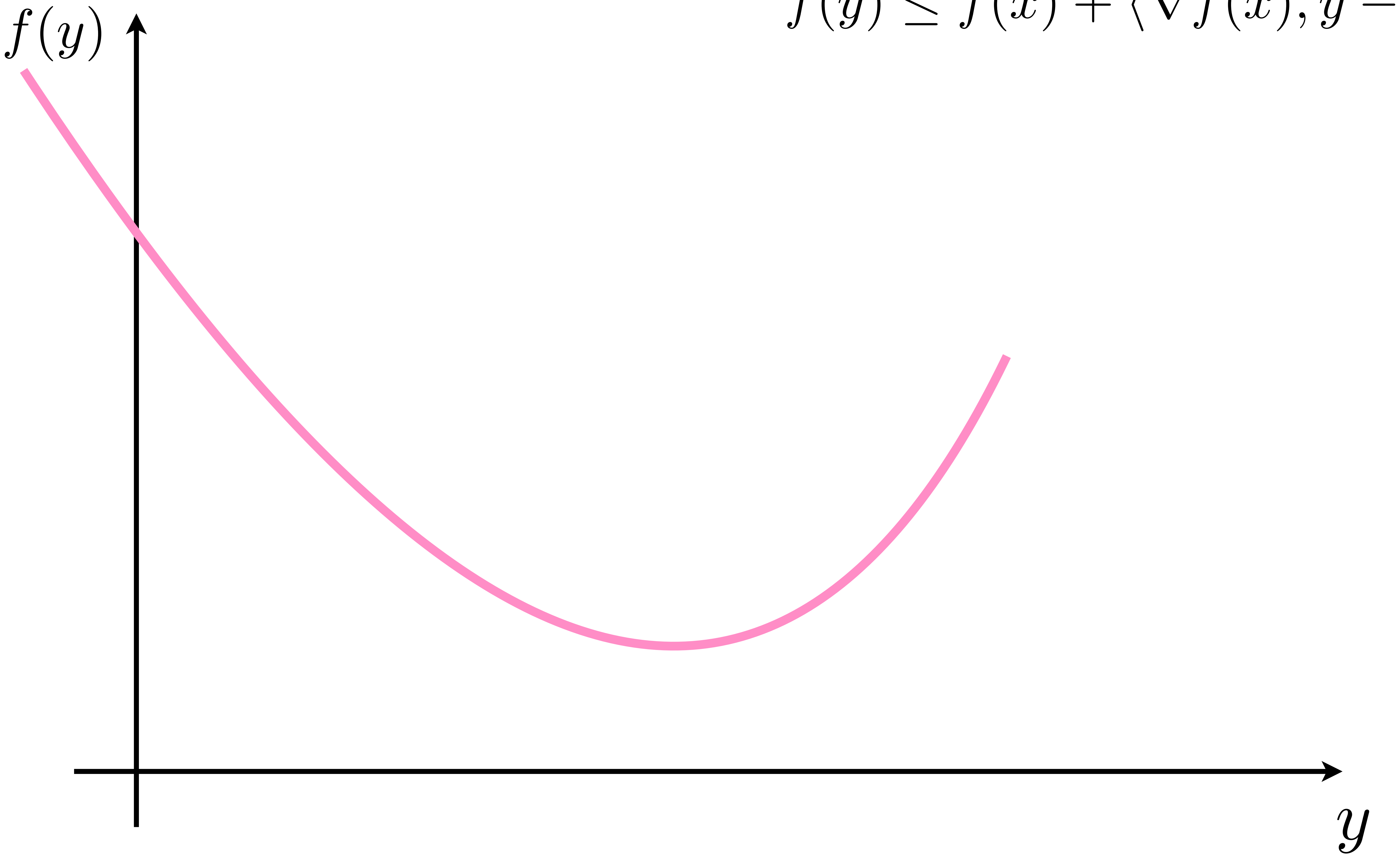
$$\nabla^2 f(x) \preceq LI$$

Interpretation?

Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

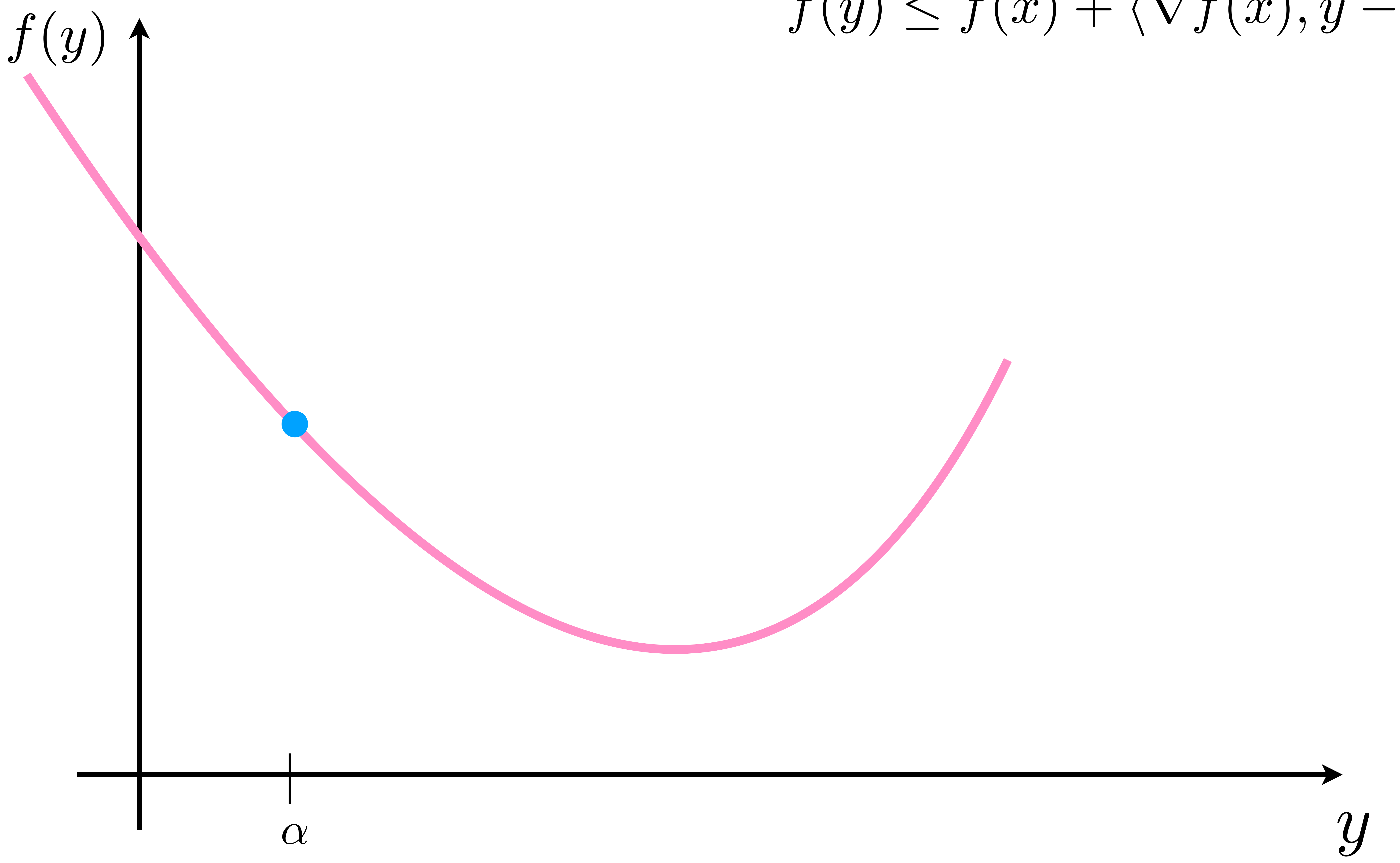
$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$



Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

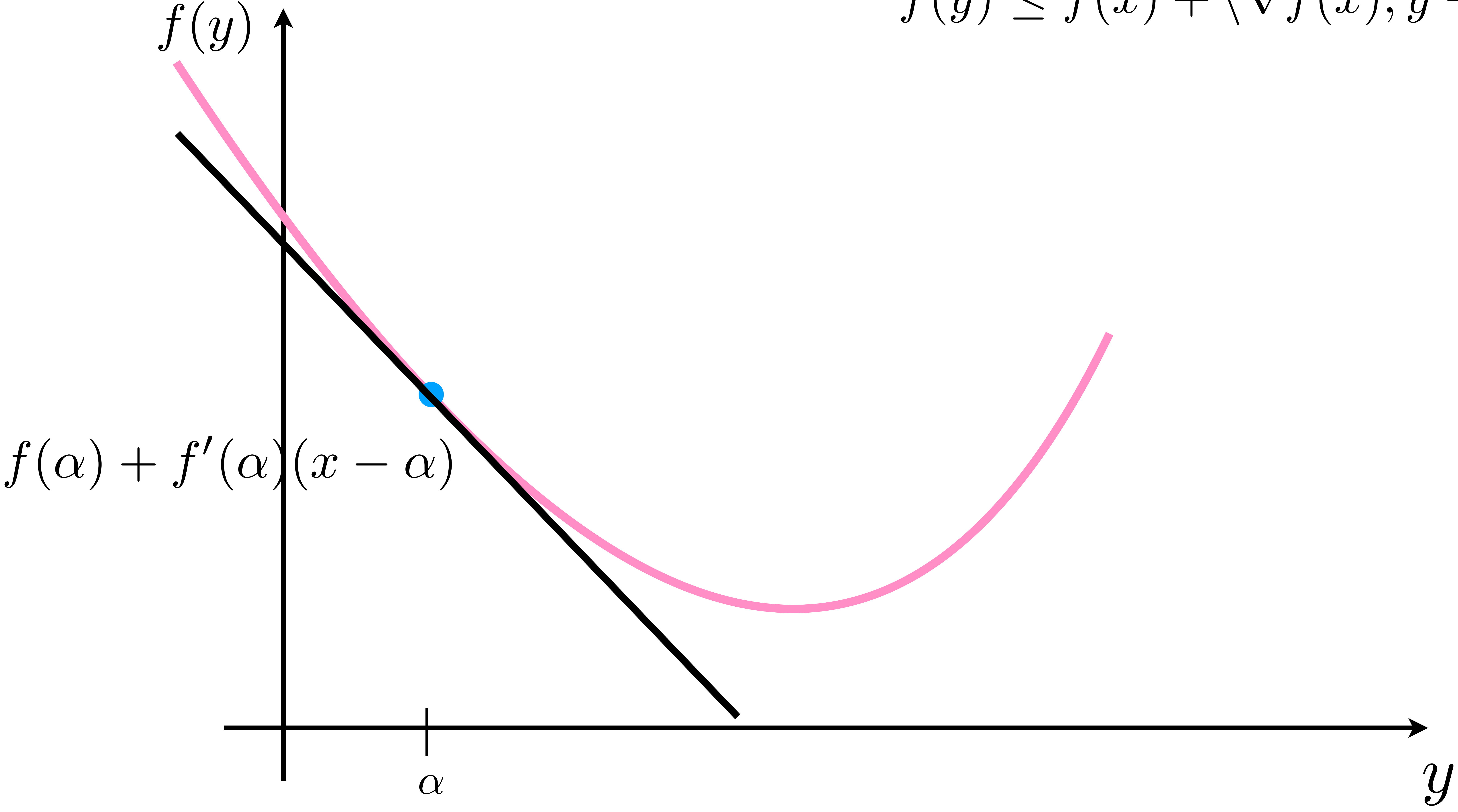
$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$



Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

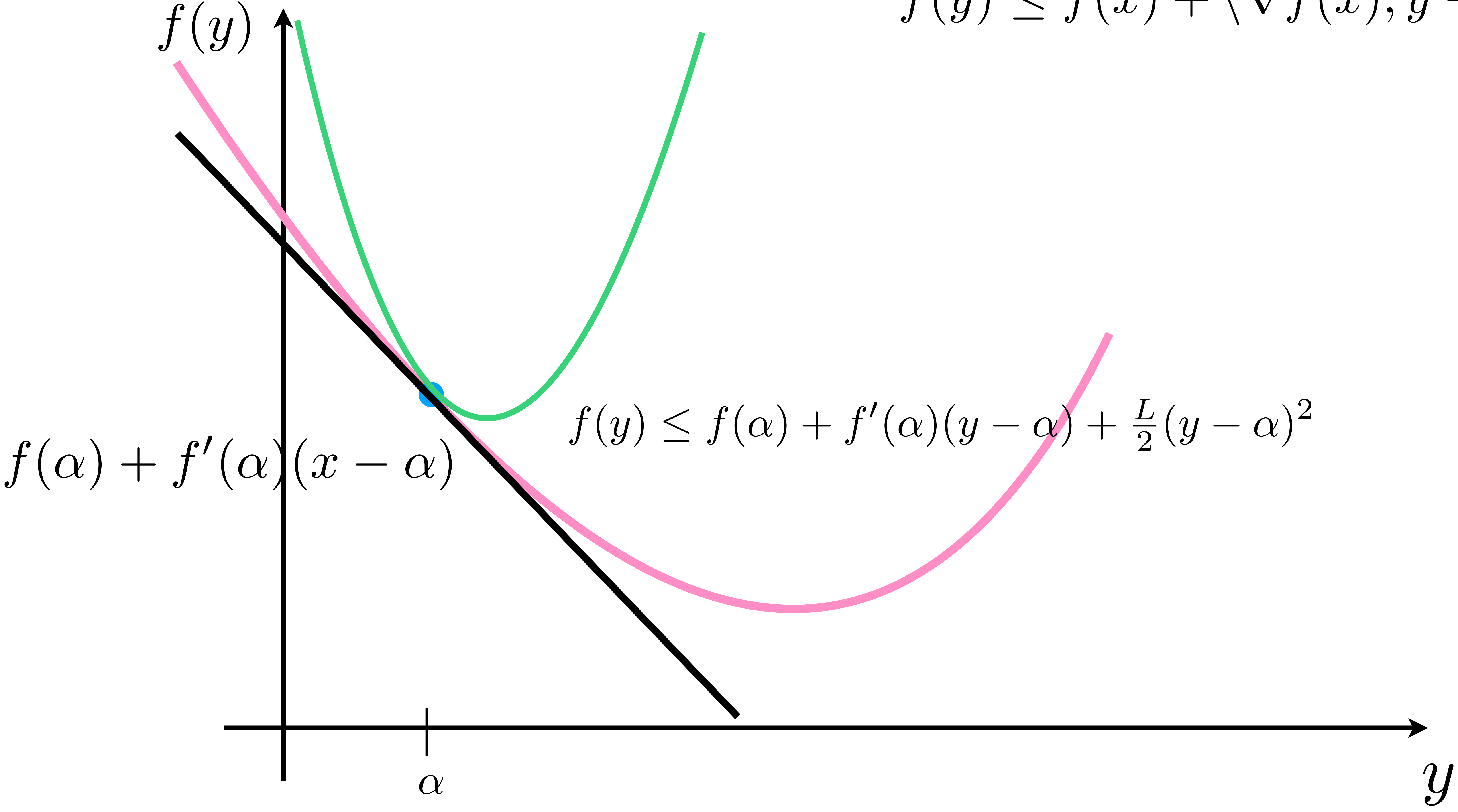
$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$



Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

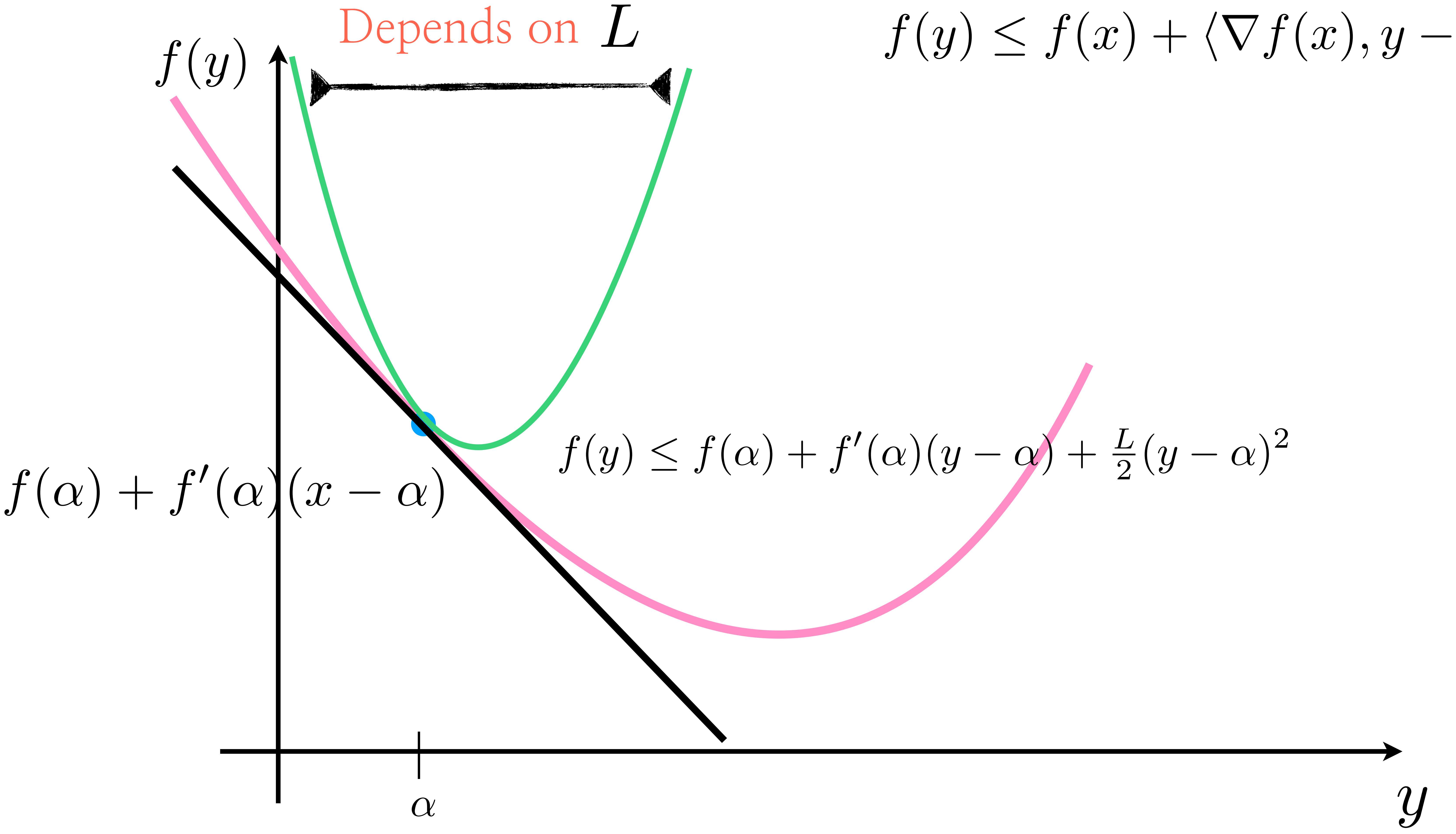
$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$



Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

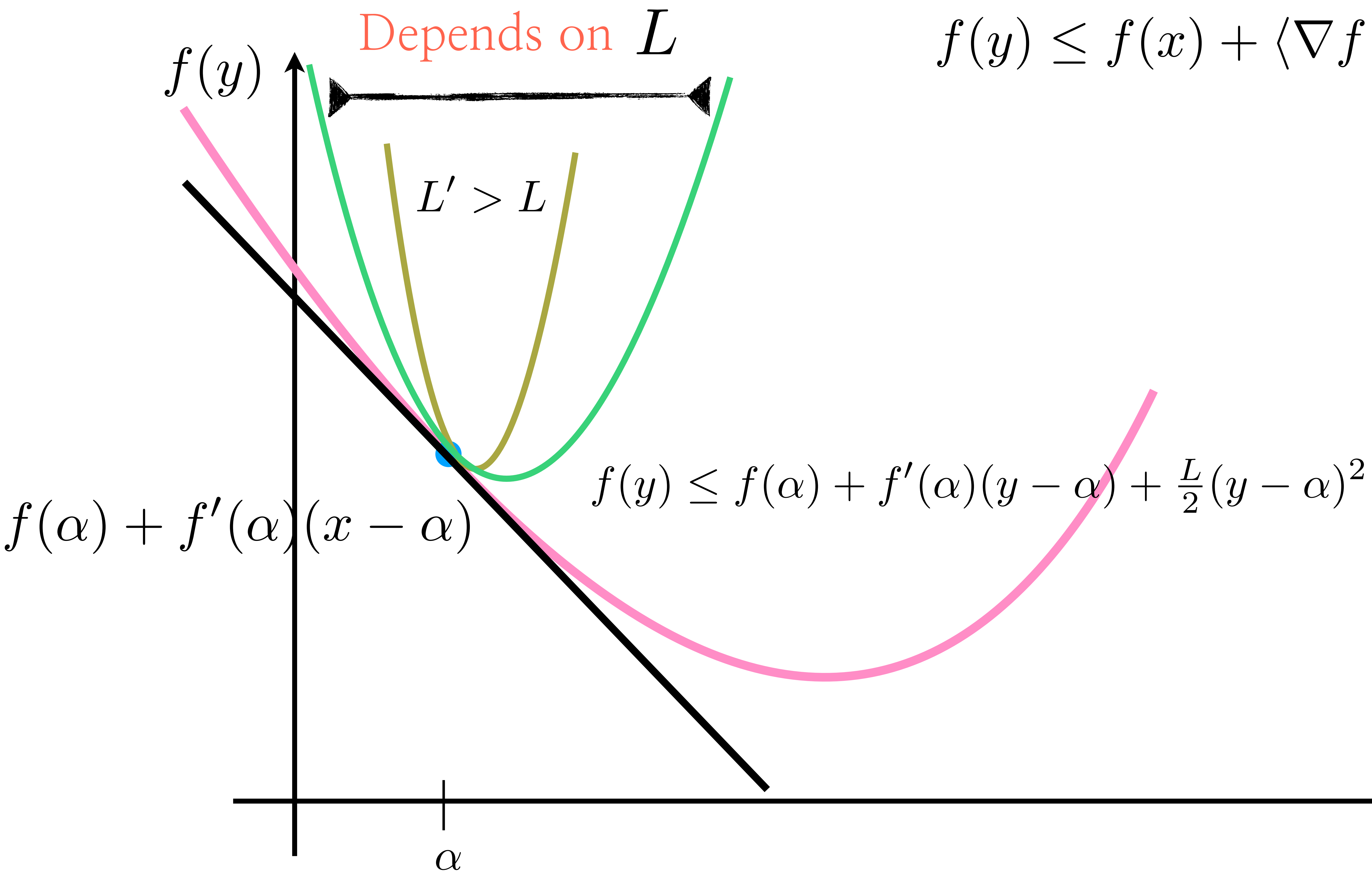
$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$



Lipschitz conditions

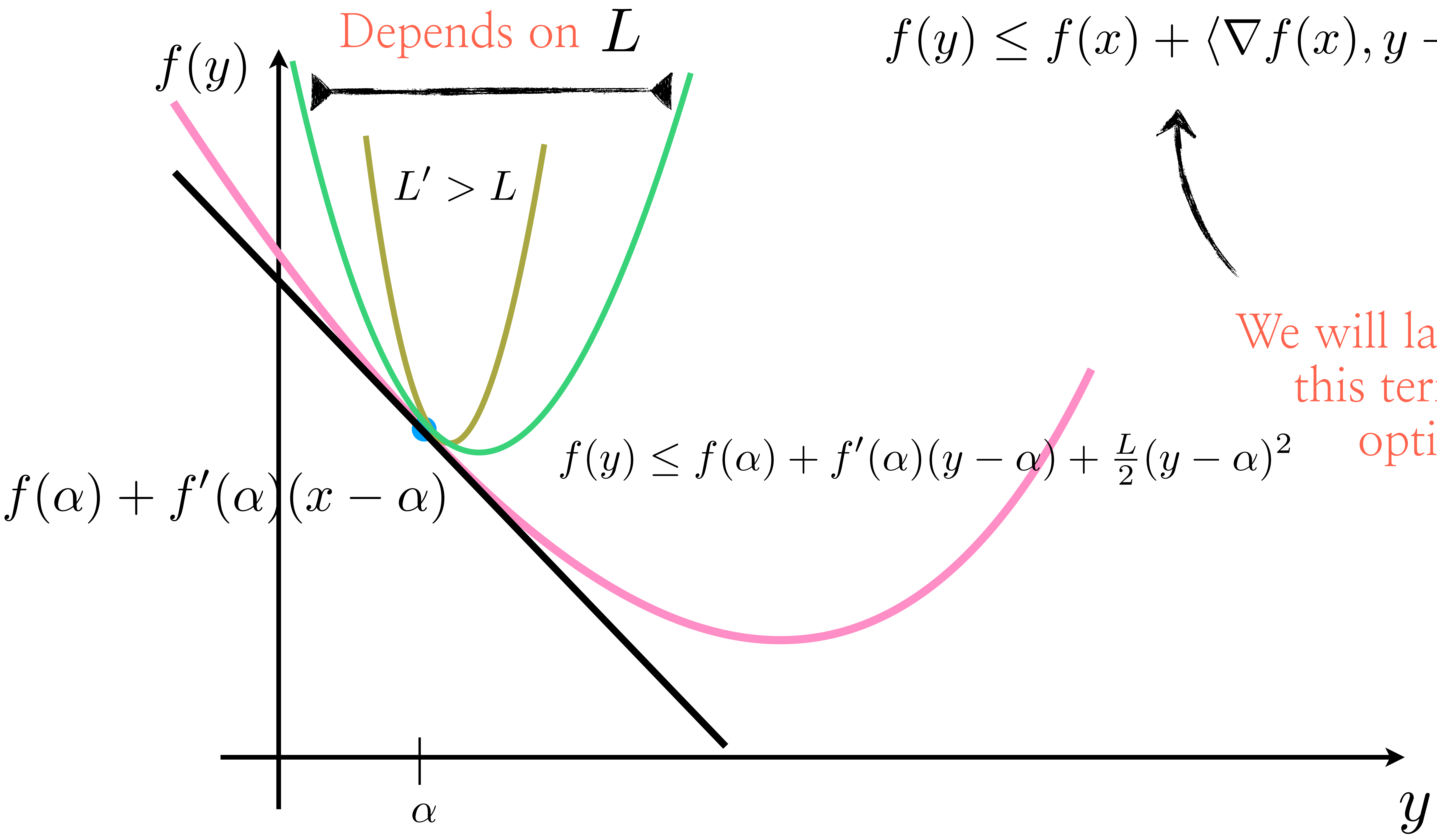
- Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$



Lipschitz conditions

– Equivalent characterizations: $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$



$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|_2^2$$

We will later on see how this term appear in optimization

Lipschitz conditions

- How does this relate to Taylor's expansion?

Lipschitz conditions

- How does this relate to Taylor's expansion?

$$f(y) = f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \left\langle \nabla^2 f(z)(y - x), y - x \right\rangle$$

Lipschitz conditions

– How does this relate to Taylor's expansion?

$$f(y) = f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \left\langle \nabla^2 f(z)(y - x), y - x \right\rangle$$

– From $\nabla^2 f(x) \preceq LI$, we have:

$$\nabla^2 f(x) \preceq LI \Rightarrow \|\nabla^2 f(x)\|_2 \leq \|LI\|_2 \Rightarrow \|\nabla^2 f(x)\|_2 \leq L$$

Lipschitz conditions

- How does this relate to Taylor's expansion?

$$f(y) = f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \left\langle \nabla^2 f(z)(y - x), y - x \right\rangle$$

- From $\nabla^2 f(x) \preceq LI$, we have:

$$\nabla^2 f(x) \preceq LI \Rightarrow \|\nabla^2 f(x)\|_2 \leq \|LI\|_2 \Rightarrow \|\nabla^2 f(x)\|_2 \leq L$$

- Then:

$$\frac{1}{2} \left\langle \nabla^2 f(x)(y - x), y - x \right\rangle \leq \frac{1}{2} \|\nabla^2 f(x)(y - x)\|_2 \cdot \|y - x\|_2 \leq \|\nabla^2 f(x)\|_2 \|y - x\|_2^2$$

Lipschitz conditions

– How does this relate to Taylor's expansion?

$$f(y) = f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \left\langle \nabla^2 f(z)(y - x), y - x \right\rangle$$

– From $\nabla^2 f(x) \preceq LI$, we have:

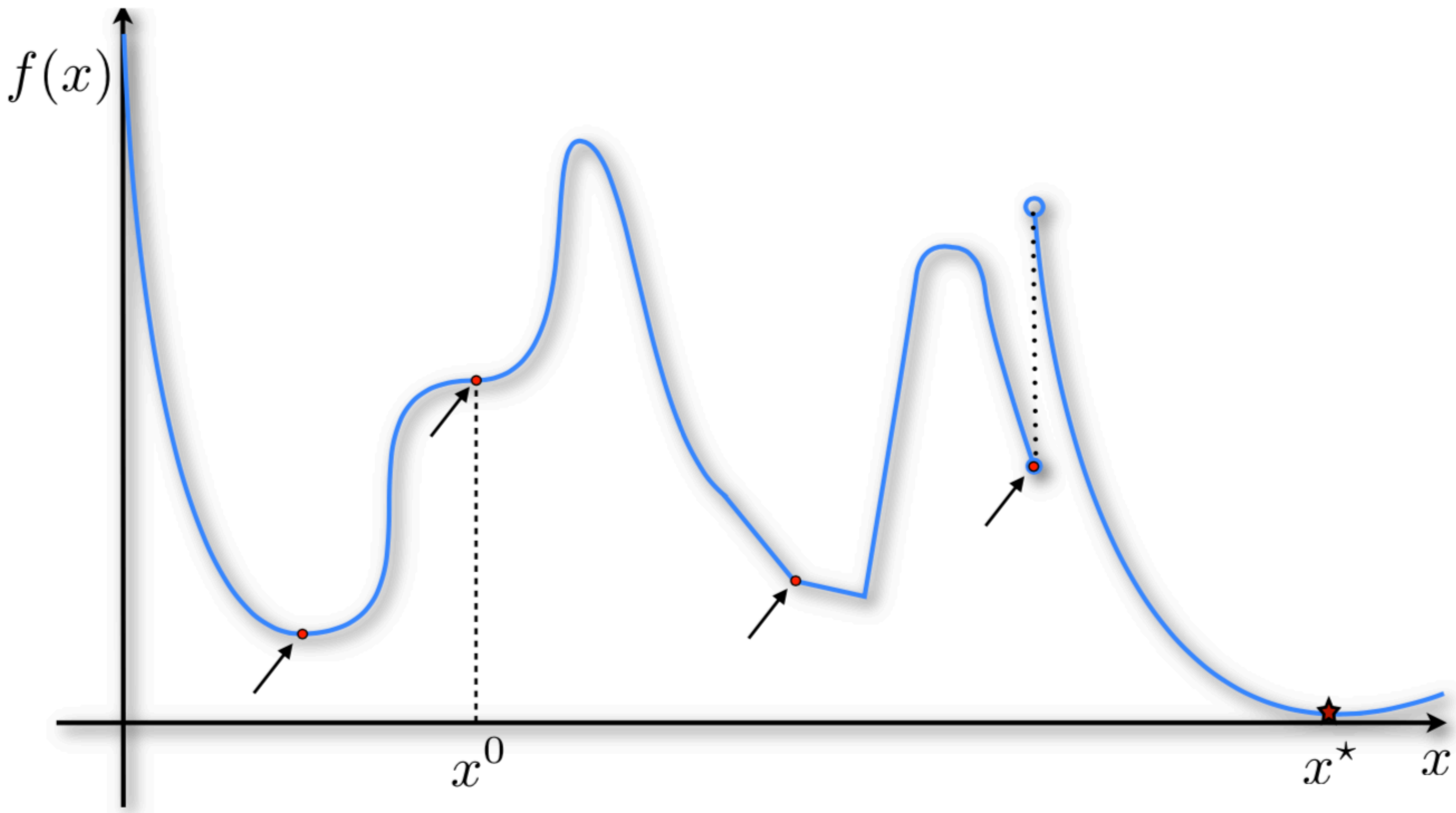
$$\nabla^2 f(x) \preceq LI \Rightarrow \|\nabla^2 f(x)\|_2 \leq \|LI\|_2 \Rightarrow \|\nabla^2 f(x)\|_2 \leq L$$

– Then:

$$\frac{1}{2} \left\langle \nabla^2 f(x)(y - x), y - x \right\rangle \leq \frac{1}{2} \|\nabla^2 f(x)(y - x)\|_2 \cdot \|y - x\|_2 \leq \|\nabla^2 f(x)\|_2 \|y - x\|_2^2$$

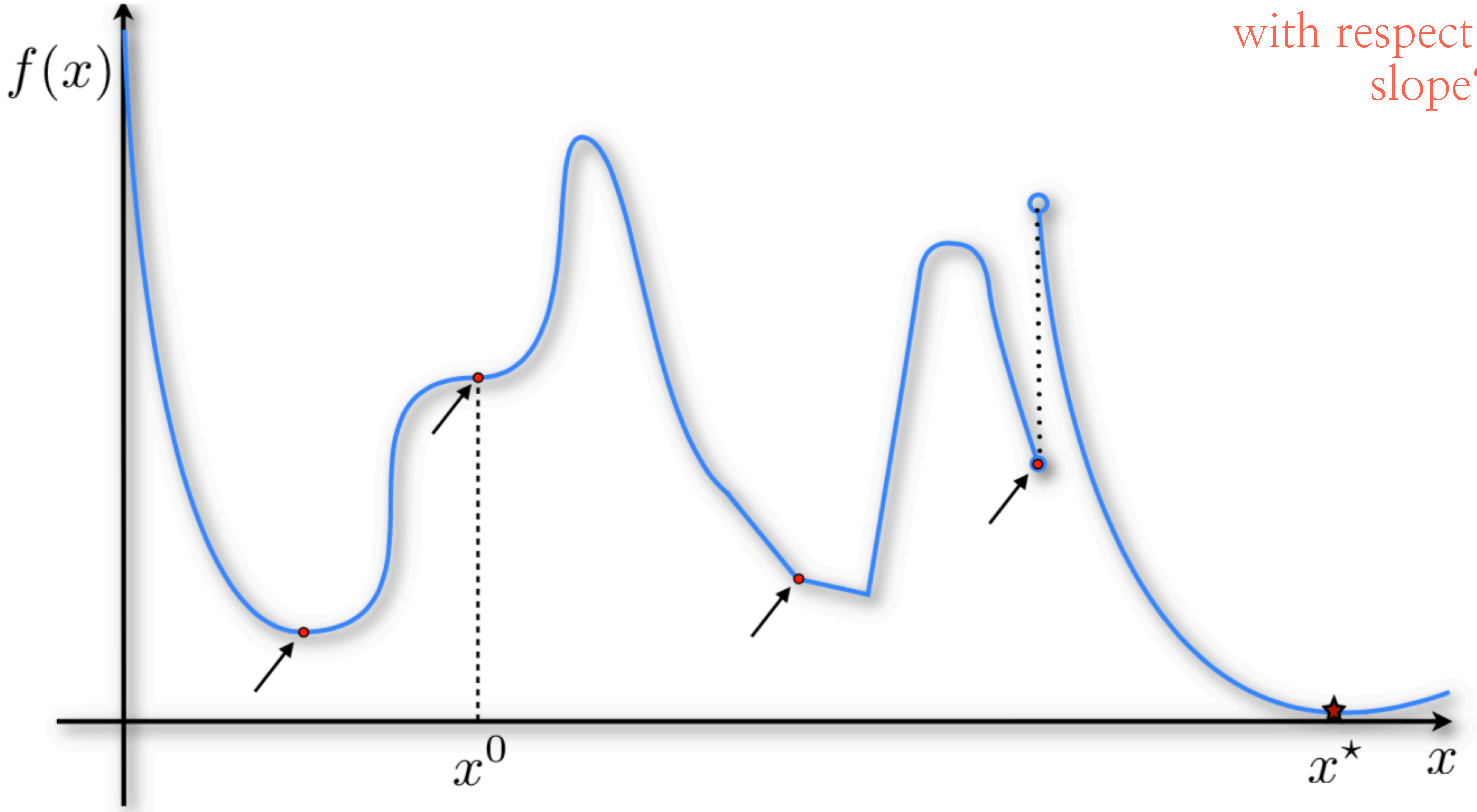
$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|x - y\|_2^2$$

Agnostic optimization



Agnostic optimization

What do you observe at local minima/maxima with respect to their slope?



Types of solutions

- Global minimizer $x^\star$:

Types of solutions

- Global minimizer $x^\star$: $f(x^\star) \leq f(x), \ \forall x$

(If we could somehow check that, we are golden!)

Types of solutions

- Global minimizer $x^\star$: $f(x^\star) \leq f(x), \ \forall x$

(If we could somehow check that, we are golden!)

- Local minimizer $\widehat{x}$:

Types of solutions

- Global minimizer $x^\star$: $f(x^\star) \leq f(x), \ \forall x$

(If we could somehow check that, we are golden!)

- Local minimizer $\widehat{x}$: $f(\widehat{x}) \leq f(x), \ \forall x \in \mathcal{N}_{\widehat{x}}$

Types of solutions

- Global minimizer $x^\star$: $f(x^\star) \leq f(x), \ \forall x$

(If we could somehow check that, we are golden!)

- Local minimizer $\widehat{x}$: $f(\widehat{x}) \leq f(x), \ \forall x \in \mathcal{N}_{\widehat{x}}$

- What is the meaning of strict inequality vs. inequality?

Types of solutions

- Global minimizer $x^\star$: $f(x^\star) \leq f(x), \; \forall x$

(If we could somehow check that, we are golden!)

- Local minimizer $\widehat{x}$: $f(\widehat{x}) \leq f(x), \; \forall x \in \mathcal{N}_{\widehat{x}}$

- What is the meaning of strict inequality vs. inequality?

- How do we recognize that a solution we have is a local (global) solution?

Types of solutions

- Global minimizer $x^\star$: $f(x^\star) \leq f(x), \; \forall x$

  (If we could somehow check that, we are golden!)

- Local minimizer $\widehat{x}$: $f(\widehat{x}) \leq f(x), \; \forall x \in \mathcal{N}_{\widehat{x}}$

- What is the meaning of strict inequality vs. inequality?

- How do we recognize that a solution we have is a local (global) solution?

  - 1^{st}–order optimality condition: $\nabla f(\widehat{x}) = 0$

Types of solutions

- Global minimizer $x^\star$: $f(x^\star) \leq f(x), \; \forall x$

(If we could somehow check that, we are golden!)

- Local minimizer $\widehat{x}$: $f(\widehat{x}) \leq f(x), \; \forall x \in \mathcal{N}_{\widehat{x}}$

- What is the meaning of strict inequality vs. inequality?

- How do we recognize that a solution we have is a local (global) solution?

  - 1^{st}–order optimality condition: $\nabla f(\widehat{x}) = 0$

  - 2^{nd}–order optimality condition: $\nabla f(\widehat{x}) = 0$ and $\nabla^2 f(\widehat{x}) \succeq 0$

Types of solutions

- Global minimizer $x^\star$: $f(x^\star) \leq f(x), \; \forall x$

(If we could somehow check that, we are golden!)

- Local minimizer $\widehat{x}$: $f(\widehat{x}) \leq f(x), \; \forall x \in \mathcal{N}_{\widehat{x}}$

- What is the meaning of strict inequality vs. inequality?

- How do we recognize that a solution we have is a local (global) solution?

Necessary

- 1^{st}–order optimality condition: $\nabla f(\widehat{x}) = 0$

- 2^{nd}–order optimality condition: $\nabla f(\widehat{x}) = 0$ and $\nabla^2 f(\widehat{x}) \succeq 0$

Gradient descent

Gradient as local information (short story)

Gradient as local information (short story)

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

Gradient as local information (long story)

- Problem to solve: $\min_{x \in \mathbb{R}^p} f(x)$

Gradient as local information (long story)

- Problem to solve: $\min_{x \in \mathbb{R}^p} f(x)$
- Assume f is differentiable, a first order Taylor expansion around x :

$$f(x+\delta) = f(x) + \langle \nabla f(x), \delta \rangle + o\left(\|\delta\|\right)$$

Gradient as local information (long story)

- Problem to solve: $\min_{x \in \mathbb{R}^p} f(x)$
- Assume f is differentiable, a first order Taylor expansion around x :

$$f(x + \delta) = f(x) + \langle \nabla f(x), \delta \rangle + o(\|\delta\|)$$

- Minimizing f locally, the steepest descent direction is δ such that:

$$\langle \delta, \nabla f(x) \rangle$$

Gradient as local information (long story)

- Problem to solve: $\min_{x \in \mathbb{R}^p} f(x)$
- Assume f is differentiable, a first order Taylor expansion around x :

$$f(x+\delta) = f(x) + \langle \nabla f(x), \delta \rangle + o\left(\|\delta\|\right)$$

- Minimizing f locally, the steepest descent direction is δ such that:

$$\langle \delta, \nabla f(x) \rangle$$

- Focusing on the direction to move to, and assuming Euclidean space, we get:

$$\delta = -\frac{\nabla f(x)}{\|\nabla f(x)\|_2}$$

(normalized direction)

Gradient as local information (long story)

- Problem to solve: $\min_{x \in \mathbb{R}^p} f(x)$
- Assume f is differentiable, a first order Taylor expansion around x :

$$f(x+\delta) = f(x) + \langle \nabla f(x), \delta \rangle + o\left(\|\delta\|\right)$$

- Minimizing f locally, the steepest descent direction is δ such that:

$$\langle \delta, \nabla f(x) \rangle$$

- Focusing on the direction to move to, and assuming Euclidean space, we get:

$$\delta = -\frac{\nabla f(x)}{\|\nabla f(x)\|_2}$$

(normalized direction)

- **Gradient descent:** $x_{t+1} = x_t + \delta_t = x_t - \eta \nabla f(x_t)$

Gradient as local information (long story)

- Problem to solve: $\min_{x \in \mathbb{R}^p} f(x)$
- Assume f is differentiable, a first order Taylor expansion around x :

$$f(x+\delta) = f(x) + \langle \nabla f(x), \delta \rangle + o\left(\|\delta\|\right)$$

- Minimizing f locally, the steepest descent direction is δ such that:

$$\langle \delta, \nabla f(x) \rangle$$

- Focusing on the direction to move to, and assuming Euclidean space, we get:

$$\delta = -\frac{\nabla f(x)}{\|\nabla f(x)\|_2}$$

(normalized direction)

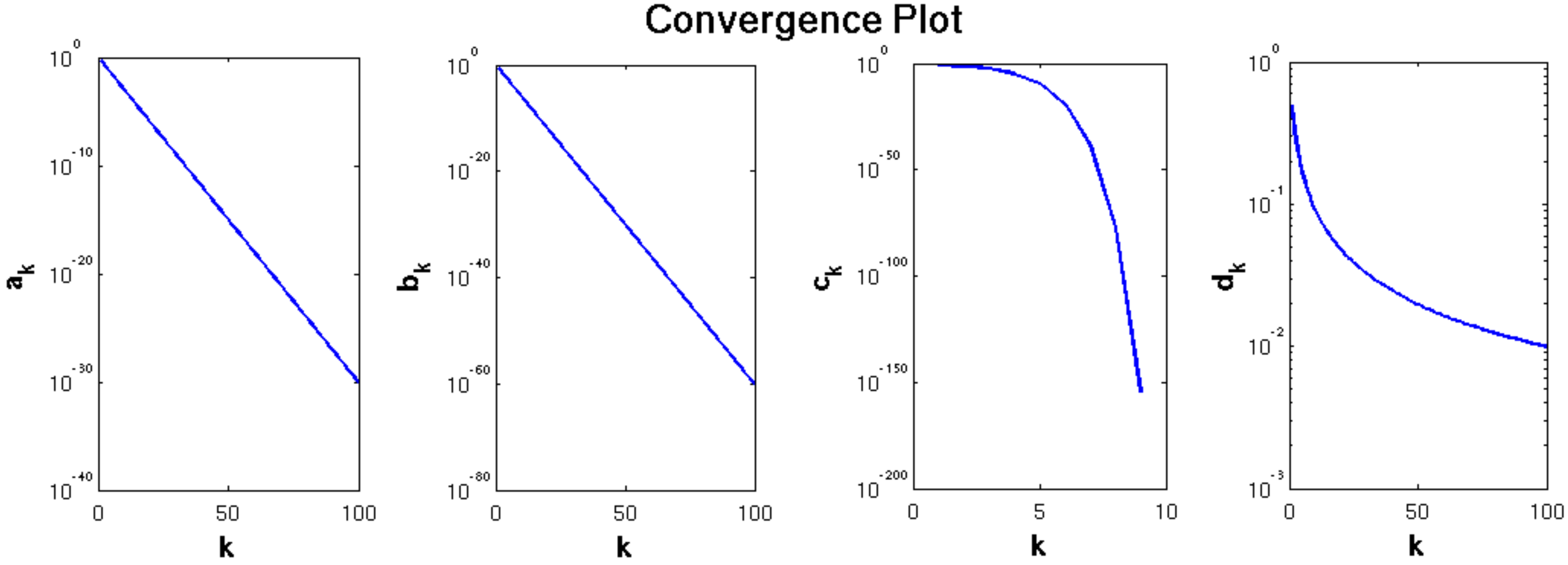
- **Gradient descent:** $x_{t+1} = x_t + \delta_t = \boxed{x_t - \eta \nabla f(x_t)}$

First convergence result

Whiteboard

Convergence rates 101

(Source: Wikipedia)



$O(\log 1/\varepsilon)$

$q^k, \ q \in (0,1)$

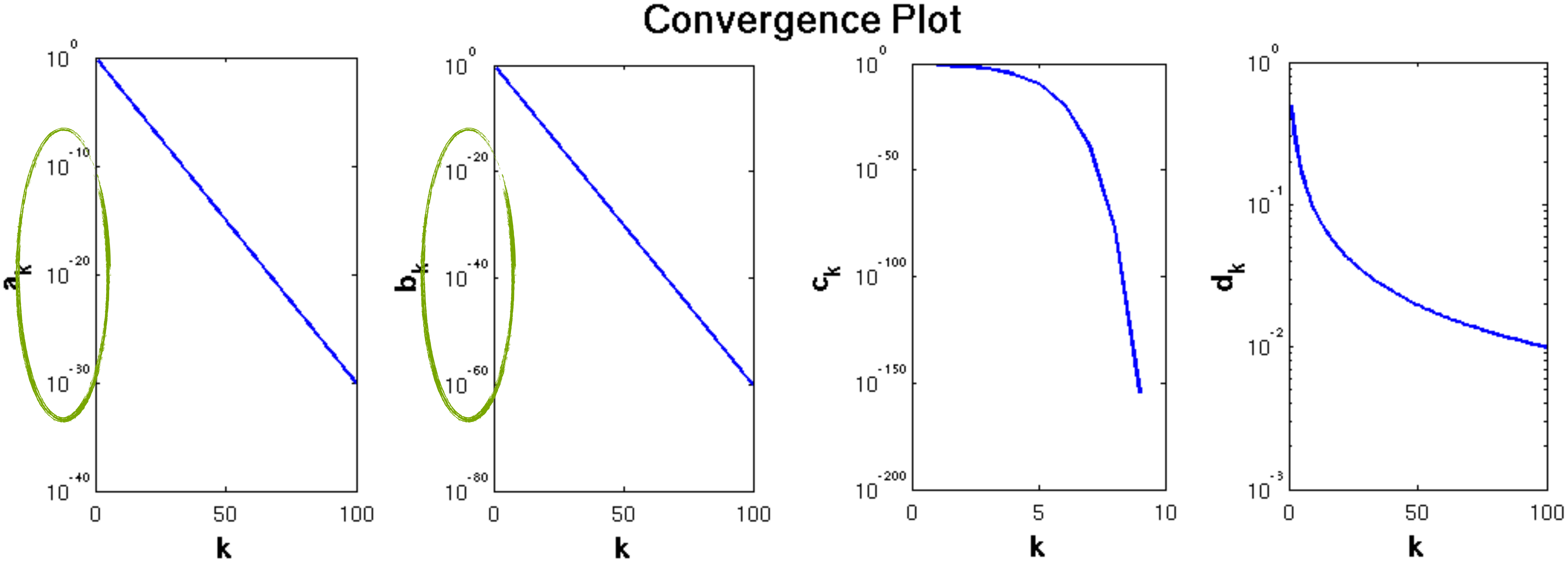
$O\left(\log\log(1/\varepsilon)\right)$

$O(1/\varepsilon^2), \ O(1/\varepsilon), \ O(1/\sqrt{\varepsilon})$

$O(1/k^2), O(1/k), O(\sqrt(k))$

Convergence rates 101

(Source: Wikipedia)



$O(\log 1/\varepsilon)$

$q^k,\ q \in (0,1)$

$O\left(\log\log(1/\varepsilon)\right)$

$O(1/\varepsilon^2),\ O(1/\varepsilon),\ O(1/\sqrt{\varepsilon})$

$O(1/k^2), O(1/k), O(\sqrt(k))$

First convergence result

$$\min_{x \in \mathbb{R}^p} f(x)$$

“Assume the objective is has Lipschitz continuous gradients. Then, gradient descent:

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

with step size

$$\eta = \frac{1}{L}$$

converges sublinearly to a stationary point; i.e.,

$$\min_t \|\nabla f(x_t)\|_2 \leq \sqrt{\frac{2L}{T+1}} \cdot \left(f(x_0) - f(x^\star)\right)^{1/2} = O\left(\frac{1}{\sqrt{T}}\right)$$ “

First convergence result

- "But, which functions satisfy Lipschitz gradient continuity?"

First convergence result

- "But, which functions satisfy Lipschitz gradient continuity?"
  - Least-squares objectives: $f(x) = \frac{1}{2}\|Ax - b\|_2^2$

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq \|A^\top A\|_2 \cdot \|x - y\|_2$$

First convergence result

- "But, which functions satisfy Lipschitz gradient continuity?"
  - Least−squares objectives: $f(x) = \frac{1}{2}\|Ax - b\|_2^2$

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq \|A^\top A\|_2 \cdot \|x - y\|_2$$

  - Logistic regression objectives: $f(x) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp(-y_i \alpha_i^\top x)\right)$

Whiteboard

First convergence result

- "But, which functions satisfy Lipschitz gradient continuity?"
  - Least−squares objectives: $f(x) = \frac{1}{2}\|Ax - b\|_2^2$

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq \|A^\top A\|_2 \cdot \|x - y\|_2$$

  - Logistic regression objectives: $f(x) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp(-y_i \alpha_i^\top x)\right)$

Whiteboard

"But these are actually convex!"

First convergence result

– "But, which functions satisfy Lipschitz gradient continuity?"

– Least−squares objectives: $f(x) = \frac{1}{2}\|Ax - b\|_2^2$

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq \|A^\top A\|_2 \cdot \|x - y\|_2$$

– Logistic regression objectives: $f(x) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp(-y_i \alpha_i^\top x)\right)$

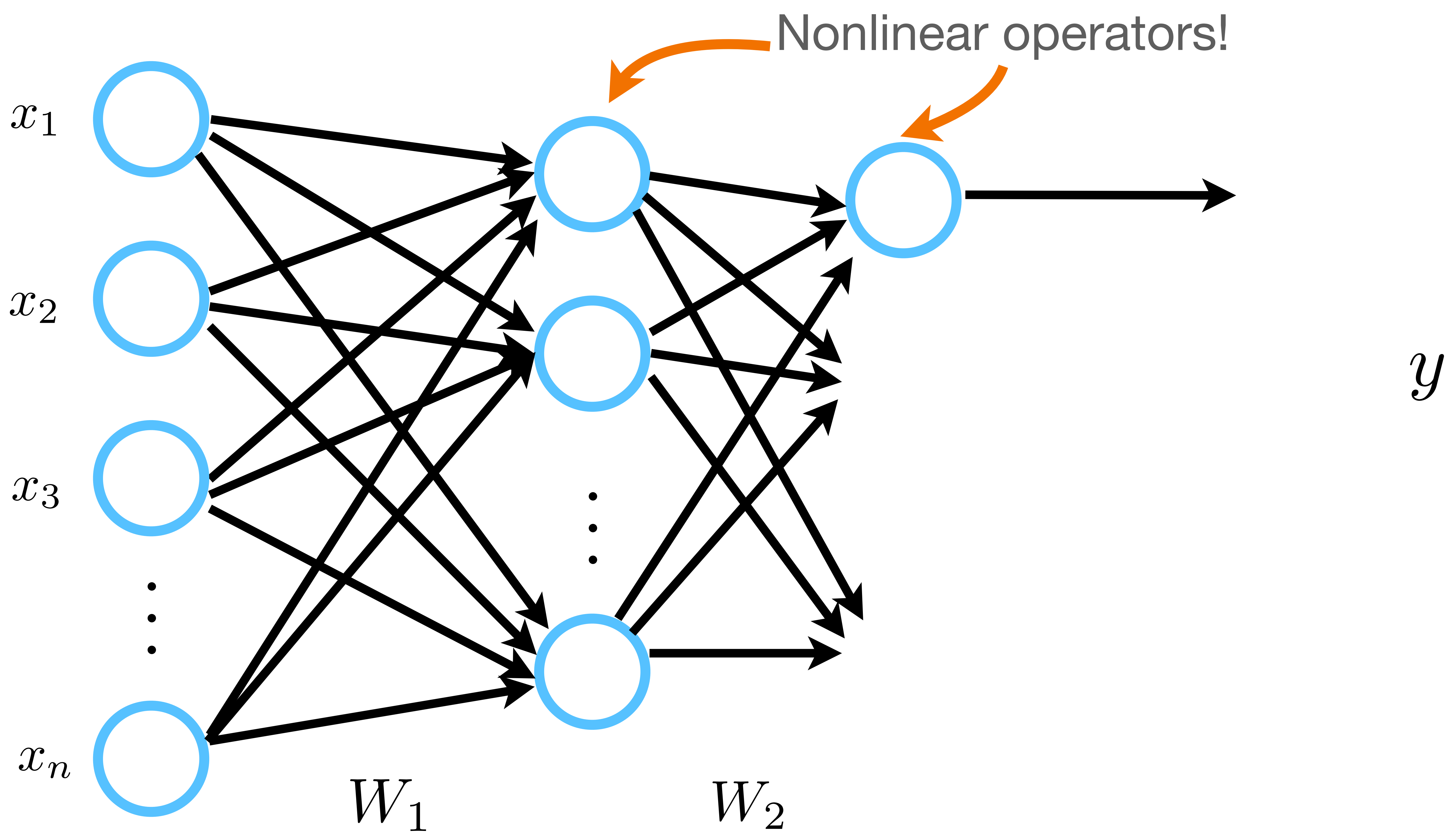
Whiteboard

"But these are actually convex!"

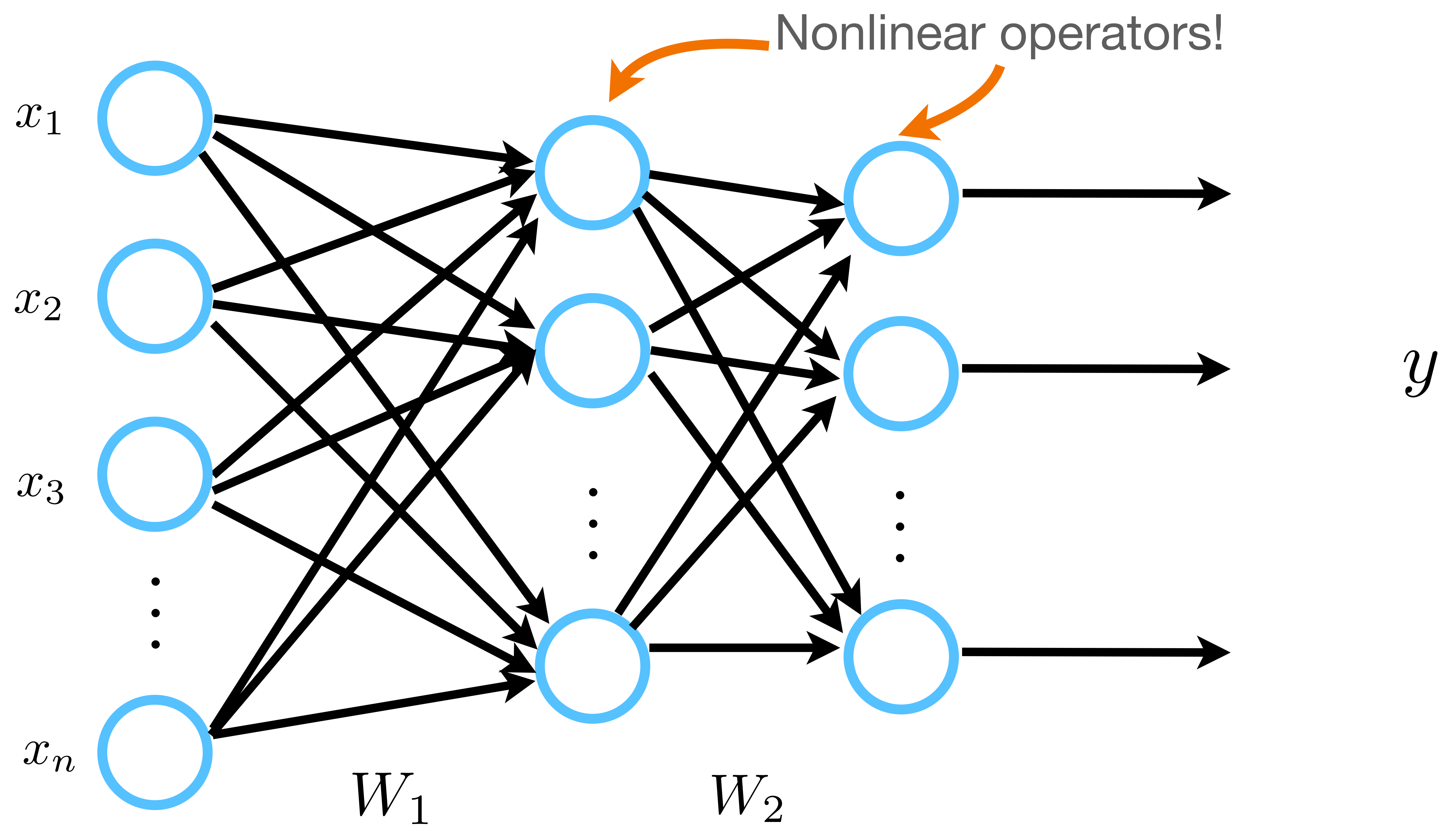
– Non−convex objective: $f(x) = x^2 + 3\sin^2(x)$

Demo

Combining all these things together: MLP

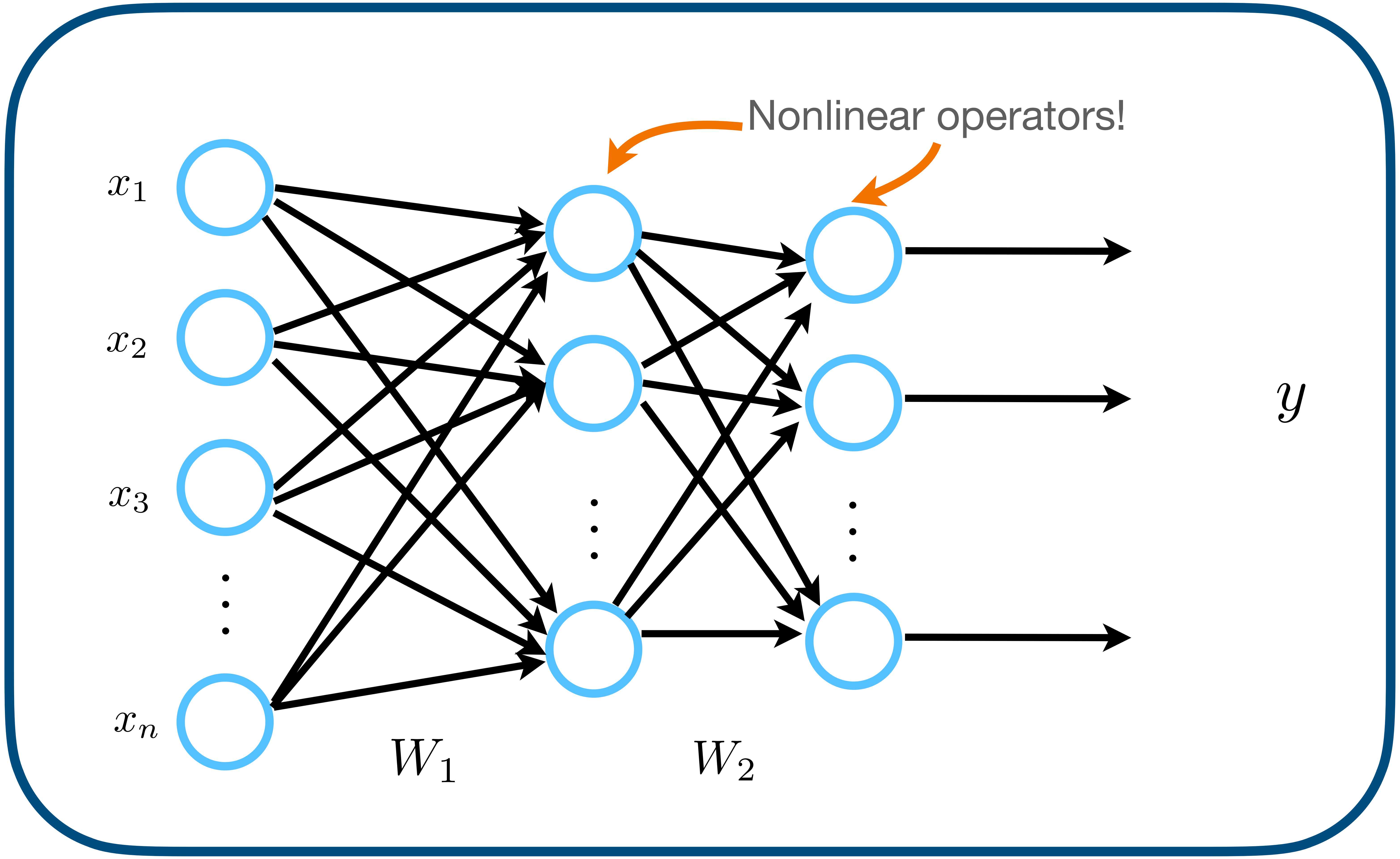


Combining all these things together: MLP



Combining all these things together: MLP

Feedforward/fully connected neural network



A bit of math in the mix: Neural Networks

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

  The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

  The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

  Through layers, the neural network learns a hierarchical representation of data (tries to encode domain information).

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

  The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

  Through layers, the neural network learns a hierarchical representation of data (tries to encode domain information).

  Neural networks are trained with local search optimization algorithms such as stochastic gradient descent, Adam, Adagrad, etc.

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

  The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

  Through layers, the neural network learns a hierarchical representation of data (tries to encode domain information).

  Neural networks are trained with local search optimization algorithms such as stochastic gradient descent, Adam, Adagrad, etc.

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

$$\widehat{y_i} = f\left(W, x_i\right)$$

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

Input

$$\widehat{y}_i = f\left(W, x_i\right)$$

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

Model

Input

$$\widehat{y}_i = f\left(W, x_i\right)$$

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

Model

Input

$$\widehat{y}_i = f\left(W, x_i\right)$$

Output

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

A bit of math in the mix: Neural Networks

– Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non–linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

Variables

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

Variables

$$W = \{W_1, W_2, \ldots, W_L\}$$

$$W_i \in \mathbb{R}^{p_{\text{in}} \times p_{\text{out}}}$$

A bit of math in the mix: Neural Networks

– Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

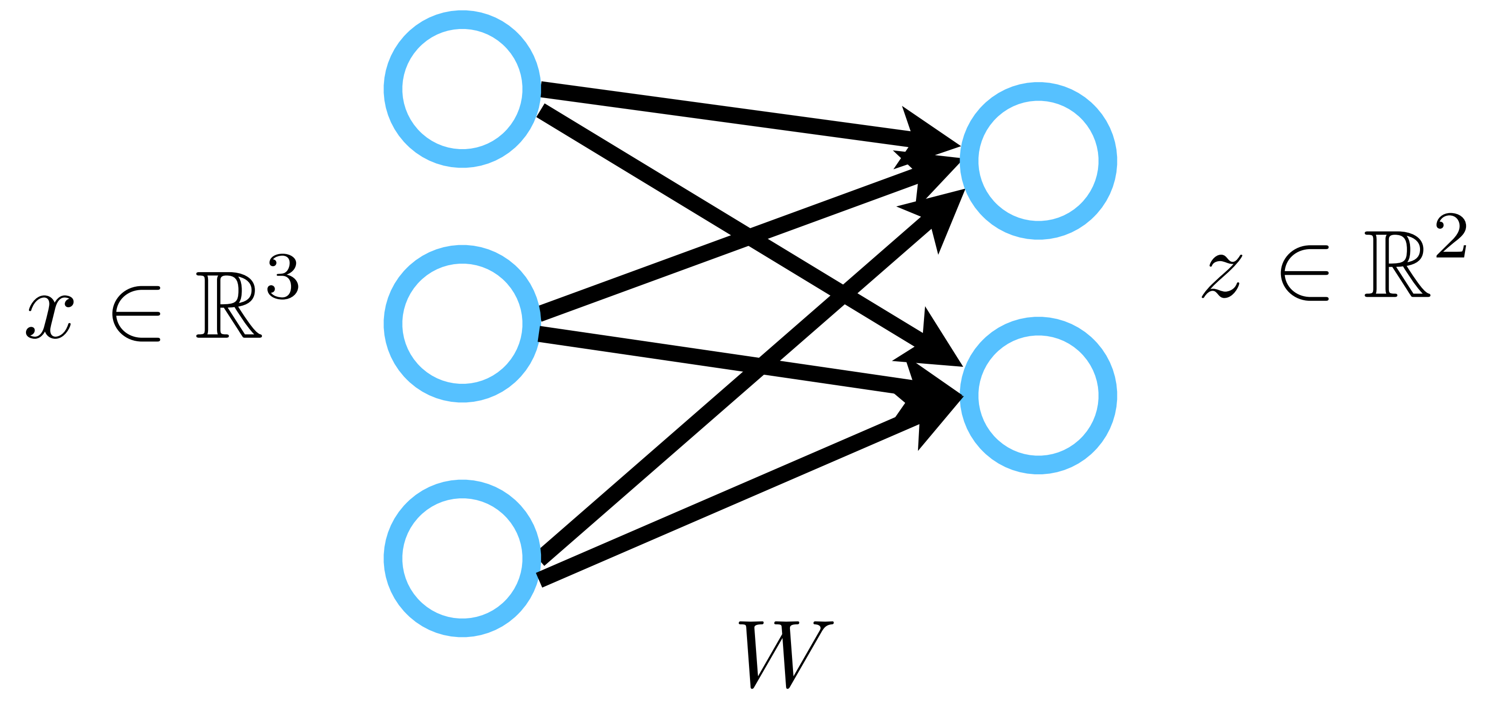
$$\hat{y}_i = f(W, x_i)$$

Variables

$$W = \{W_1, W_2, \dots, W_L\}$$

$$W_i \in \mathbb{R}^{p_{\text{in}} \times p_{\text{out}}}$$

Example:



A bit of math in the mix: Neural Networks

– Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

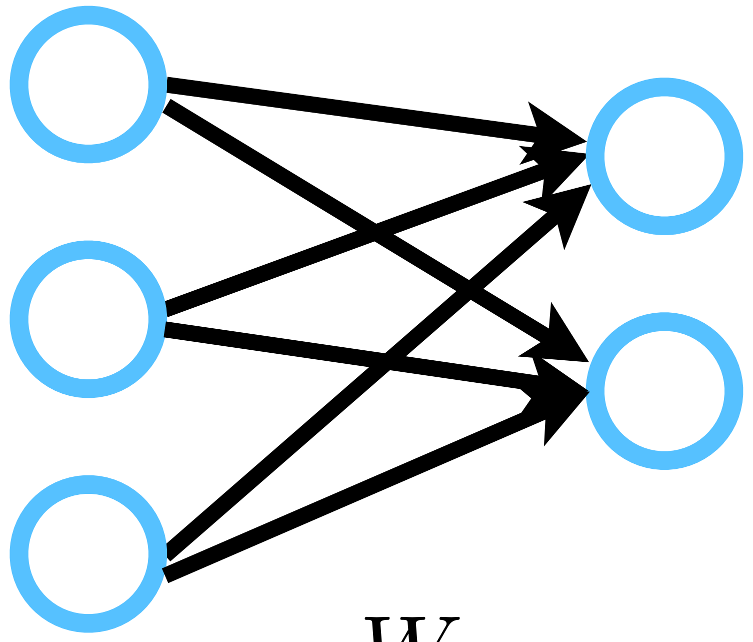
Variables

$$W = \{W_1, W_2, \ldots, W_L\}$$

$$W_i \in \mathbb{R}^{p_{\text{in}} \times p_{\text{out}}}$$

Example:

$x \in \mathbb{R}^3$



$z \in \mathbb{R}^2$

W

$$z = Wx = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

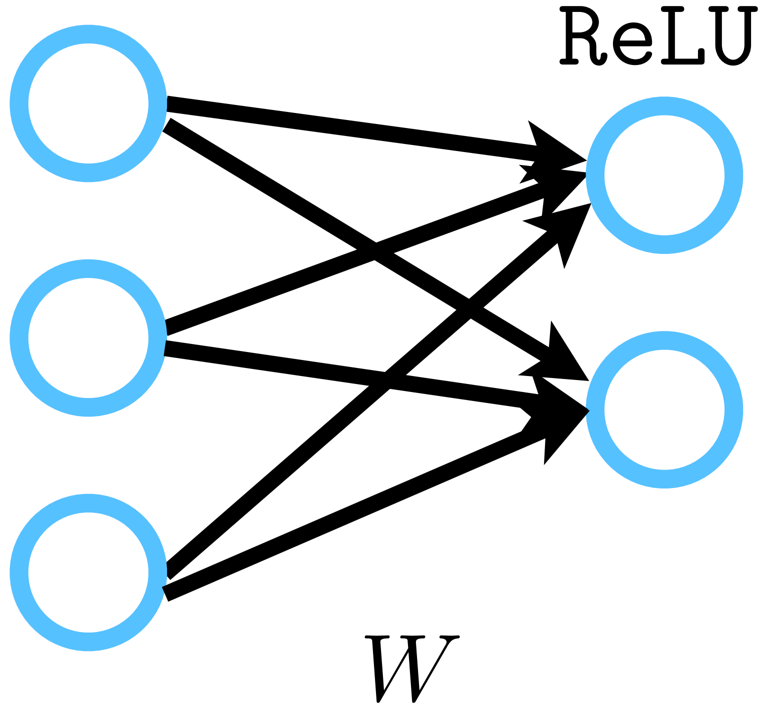
Variables

$$W = \{W_1, W_2, \ldots, W_L\}$$

$$W_i \in \mathbb{R}^{p_{\text{in}} \times p_{\text{out}}}$$

Example:

$x \in \mathbb{R}^3$



$z \in \mathbb{R}^2$

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \texttt{ReLU}(W_{1,:}x) \\ \texttt{ReLU}(W_{2,:}x) \end{bmatrix}$$

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

  The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

A bit of math in the mix: Neural Networks

– Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non–linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

Estimated output

The output of the model is compared to some ground truth in order to minimize a loss function (user–defined).

A bit of math in the mix: Neural Networks

– Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

Estimated output

How does it compare with real labels/output?

A bit of math in the mix: Neural Networks

– Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non–linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

The output of the model is compared to some ground truth in order to minimize a loss function (user–defined).

Estimated output

How does it compare with real labels/output?

$$\ell(y_i, \widehat{y}_i)$$

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

Estimated output

How does it compare with real labels/output?

$\ell(y_i, \widehat{y}_i)$ Examples:

$$\ell(y_i, \widehat{y}_i) = \frac{1}{2}\|y_i - \widehat{y}_i\|_2^2$$

$$\ell(y_i, \widehat{y}_i) = \texttt{cross-entropy}(y_i, \widehat{y}_i)$$

A bit of math in the mix: Neural Networks

– Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non–linear transformations.

$$\widehat{y}_i = f\left(W, x_i\right)$$

The output of the model is compared to some ground truth in order to minimize a loss function (user–defined).

Estimated output

How does it compare with real labels/output?

$\ell(y_i, \widehat{y}_i)$ Examples: $\ell(y_i, \widehat{y}_i) = \frac{1}{2}\|y_i - \widehat{y}_i\|_2^2$

$\ell(y_i, \widehat{y}_i) = \texttt{cross-entropy}(y_i, \widehat{y}_i)$

Goal: make loss as small as possible over the whole dataset ($\{x_i, y_i\}_{i=1}^n$)

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

  The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

  Through layers, the neural network learns a hierarchical representation of data (tries to encode domain information).

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

  The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

  Through layers, the neural network learns a hierarchical representation of data (tries to encode domain information).

Learns from data: input could be pixels or words - usually no other information provided.
(This highlights the difference with the so far procedure: hand-crafted representation learning)

A bit of math in the mix: Neural Networks

- Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

  The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non-linear transformations.

  The output of the model is compared to some ground truth in order to minimize a loss function (user-defined).

  Through layers, the neural network learns a hierarchical representation of data (tries to encode domain information).

  Neural networks are trained with local search optimization algorithms such as stochastic gradient descent, Adam, Adagrad, etc.

A bit of math in the mix: Neural Networks

– Verbose description: A neural network is a **model/black box** that takes input and and provides some answer.

$$W^\star \in \arg\min_W \sum_{\{x_i,y_i\}} \ell(y_i, \widehat{y}_i)$$

$$\arg\min_W \sum_{\{x_i,y_i\}} \ell(y_i, f(W, x_i))$$

(Quite abstract for now)

The model is **parameterized by a number of variables** and includes various operations (matrix/matrix multiplications, convolutions, etc) as well as non–linear transformations.

The output of the model is compared to some ground truth in order to minimize a loss function (user–defined).

Through layers, the neural network learns a hierarchical representation of data (tries to encode domain information).

Neural networks are trained with local search optimization algorithms such as stochastic gradient descent, Adam, Adagrad, etc.

What differentiates one neural network from another

What differentiates one neural network from another

- Operations involved: Feedforward layers only

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)
  Residual steps
  Recurrent steps

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)
  Residual steps
  Recurrent steps
- Non-linear functions used: ReLU, leaky ReLUs, tanh, sigmoid, etc.

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)
  Residual steps
  Recurrent steps
- Non-linear functions used: ReLU, leaky ReLUs, tanh, sigmoid, etc.
- Deep vs. shallow, wide vs. narrow: Although shallow wide NNs work well in theory, deep nets are more efficient, and generalize better.

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)
  Residual steps
  Recurrent steps
- Non-linear functions used: ReLU, leaky ReLUs, tanh, sigmoid, etc.
- Deep vs. shallow, wide vs. narrow: Although shallow wide NNs work well in theory, deep nets are more efficient, and generalize better.
- Objective functions (or type of learning): euclidean norm (regression), cross entropy (classification)
  only input data (autoencoders), min-max (GANs), etc.

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  - Convolution layers
  - Pooling operations
  - Regularization techniques (dropout, batch normalization)
  - Residual steps
  - Recurrent steps

  Modules

- Non-linear functions used: ReLU, leaky ReLUs, tanh, sigmoid, etc.
- Deep vs. shallow, wide vs. narrow: Although shallow wide NNs work well in theory, deep nets are more efficient, and generalize better.
- Objective functions: euclidean norm (regression), cross entropy (classification) only input data (autoencoders), min-max (GANs), etc.

  (or type of learning)

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  - Convolution layers
  - Pooling operations
  - Regularization techniques (dropout, batch normalization)
  - Residual steps
  - Recurrent steps

  Modules

- Non-linear functions used: ReLU, leaky ReLUs, tanh, sigmoid, etc.
- Deep vs. shallow, wide vs. narrow: Although shallow wide NNs work well in theory, deep nets are more efficient, and generalize better.

Modules

- Objective functions (or type of learning): euclidean norm (regression), cross entropy (classification), only input data (autoencoders), min-max (GANs), etc.

Overview

- Introduction to neural networks
- Feedforward / Fully−connected neural networks or MLPs
- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- Optimization in neural network training

Overview

- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- Optimization in neural network training

Neural networks represented as boxes

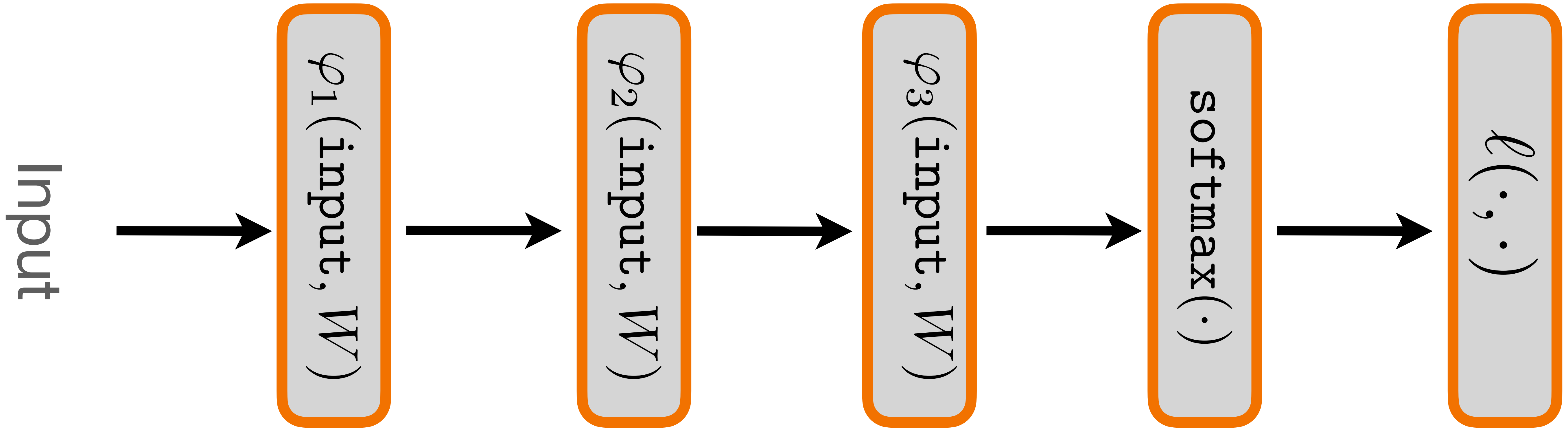
- Most used representation of neural networks (but math will get us the details)

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot, \cdot)$

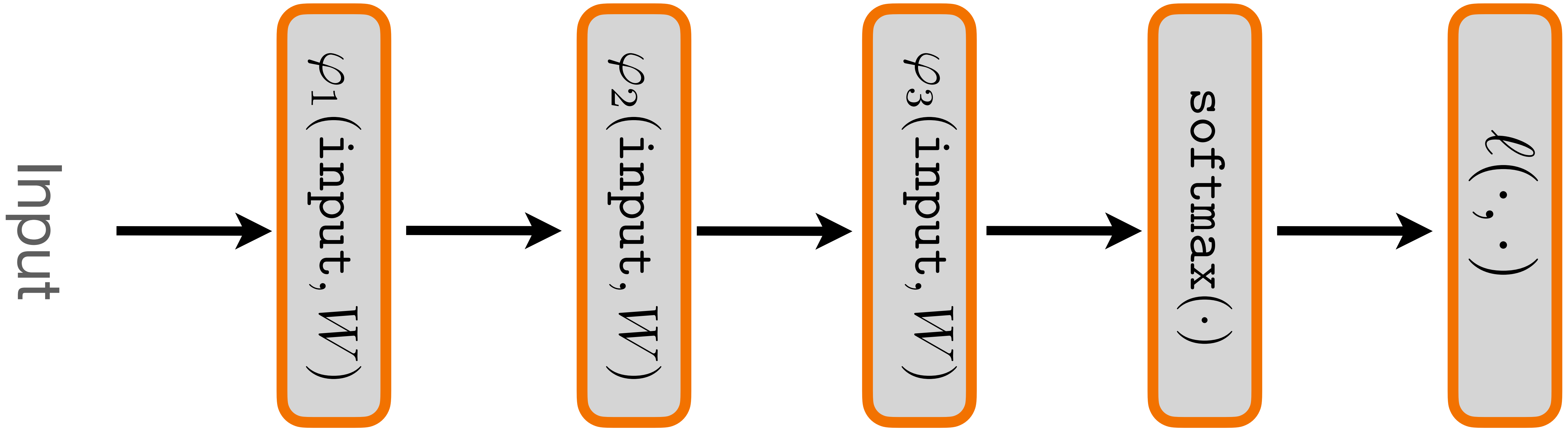
Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot,\cdot)$



Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot, \cdot)$

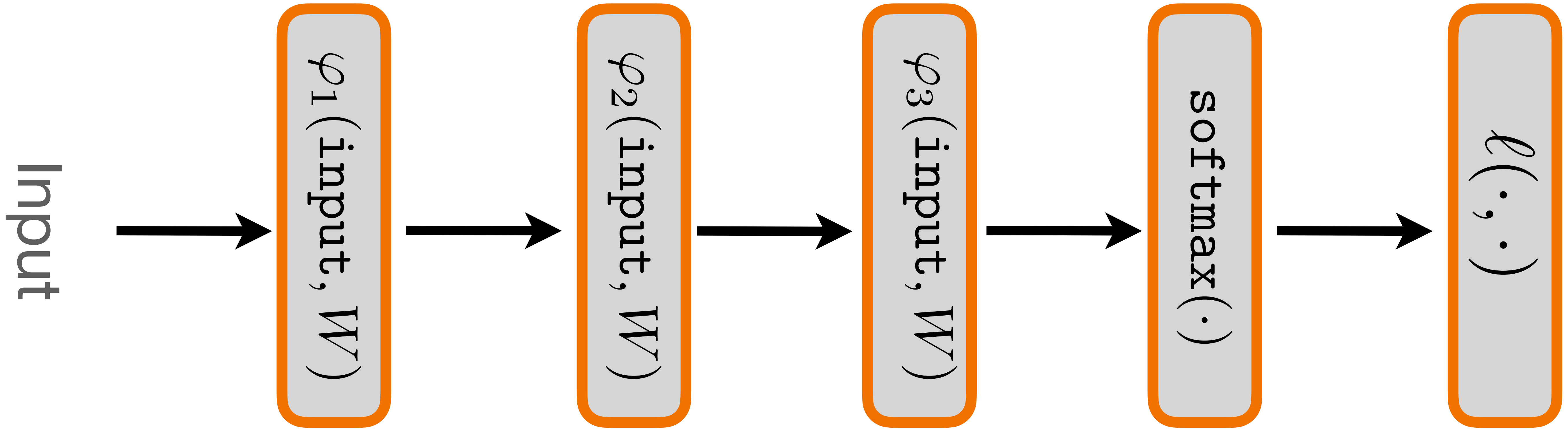


$$\ell\left(\texttt{softmax}\left(\varphi_3(\varphi_2(\varphi_1(x_i, W), W), W\right), \quad y_i\right)$$

Softmax: function that normalizes and outputs a vector of probabilities on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot,\cdot)$

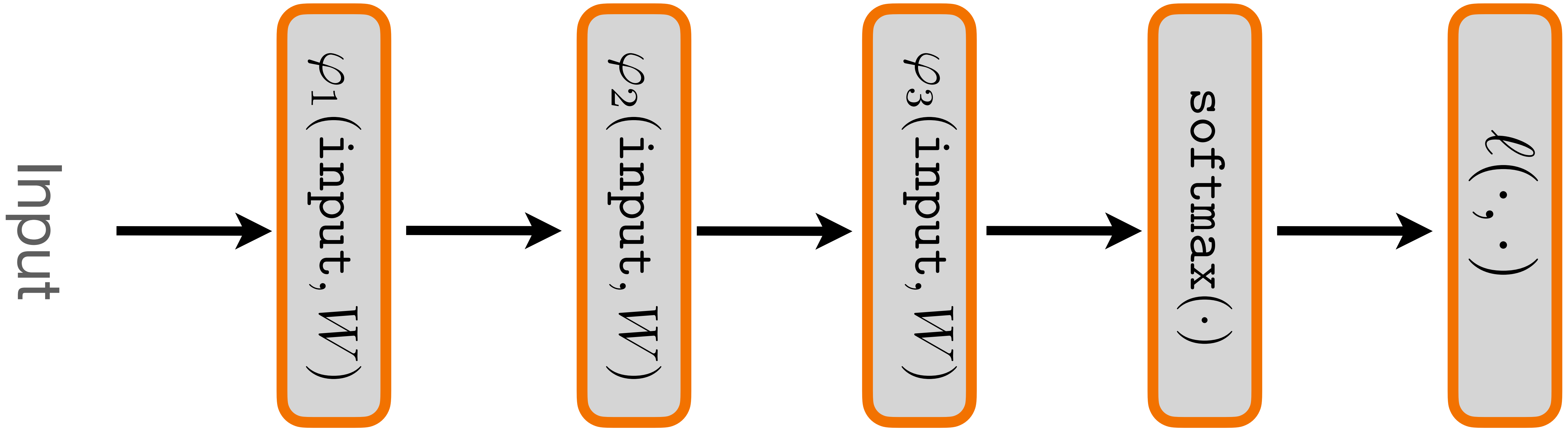


$$\ell\left(\texttt{softmax}\left(\varphi_3(\varphi_2(\varphi_1(x_i, W), W), W\right), \; y_i\right)$$

Softmax: function that normalizes and outputs a vector of probabilities on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot, \cdot)$

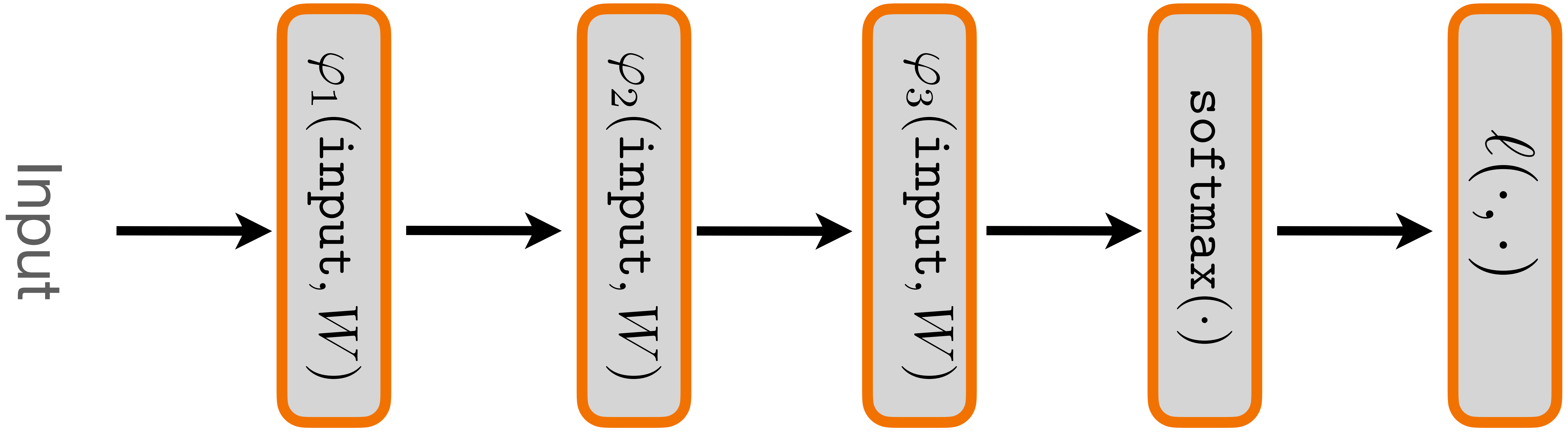


$$\ell\left(\texttt{softmax}\left(\varphi_3\left(\varphi_2\left(\varphi_1(x_i, W), W\right), W\right), \quad y_i\right)$$

Softmax: function that normalizes
and outputs a vector of probabilities
on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot,\cdot)$

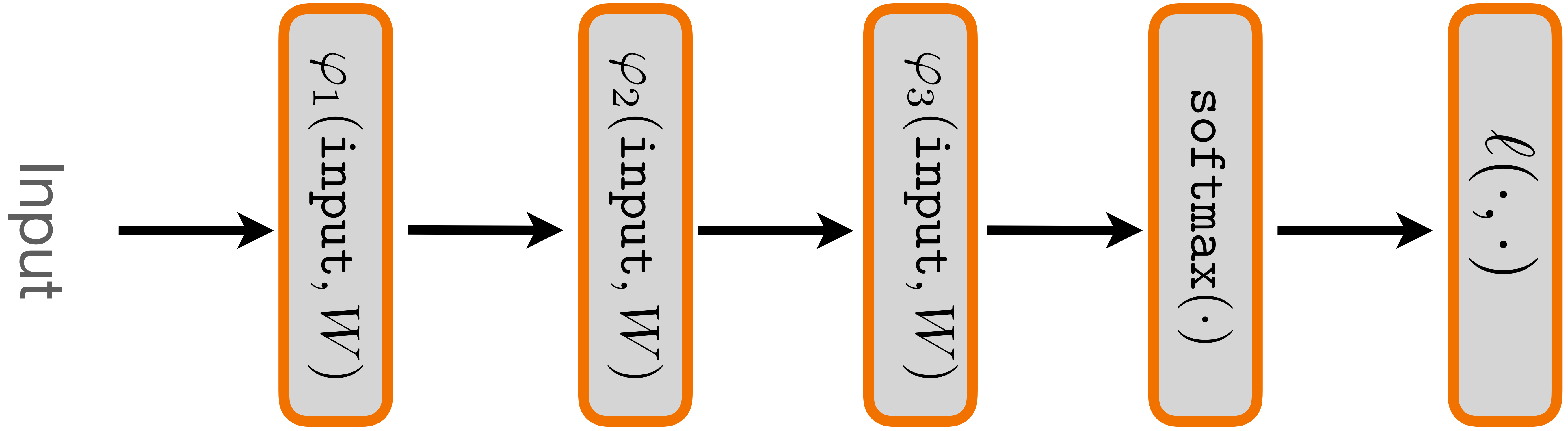


$$\ell\left(\texttt{softmax}\left(\varphi_3(\varphi_2(\varphi_1(x_i, W), W), W)\right), \quad y_i\right)$$

Softmax: function that normalizes and outputs a vector of probabilities on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot,\cdot)$

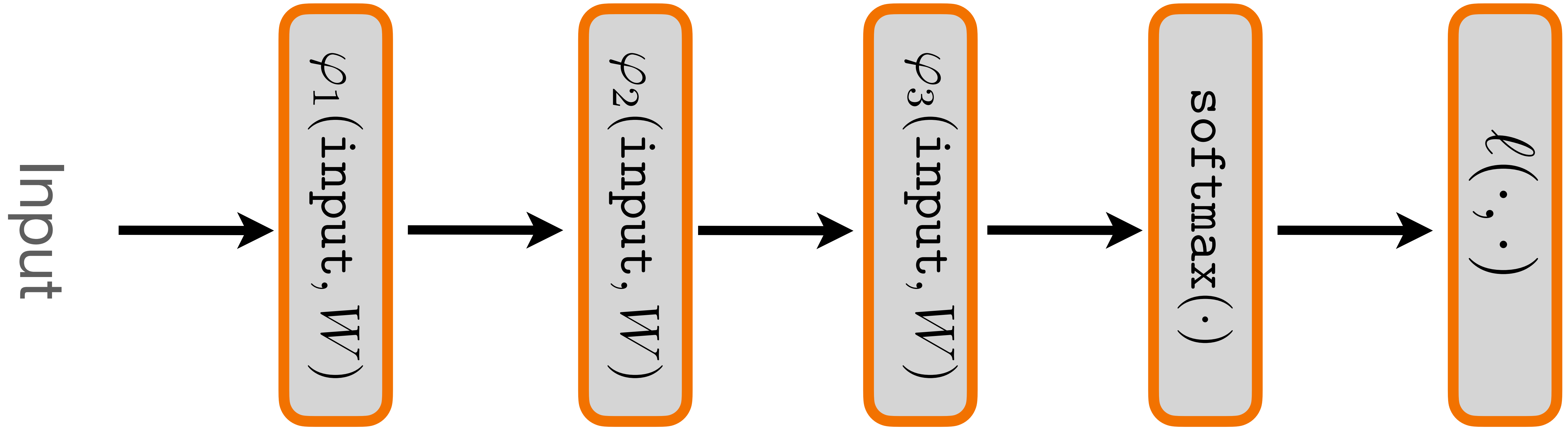


$$\ell\left(\texttt{softmax}\left(\varphi_3(\varphi_2(\varphi_1(x_i, W), W), W)\right), \; y_i\right)$$

Softmax: function that normalizes
and outputs a vector of probabilities
on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot,\cdot)$

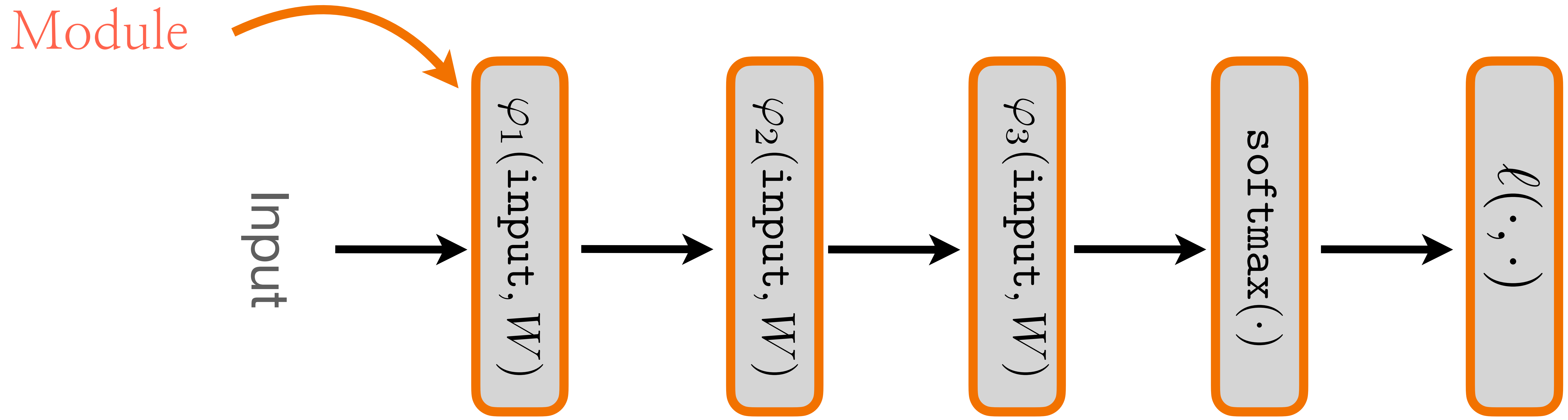


$$\ell\left(\texttt{softmax}\left(\varphi_3(\varphi_2(\varphi_1(x_i, W), W), W\right), \quad y_i\right)$$

Softmax: function that normalizes and outputs a vector of probabilities on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot, \cdot)$

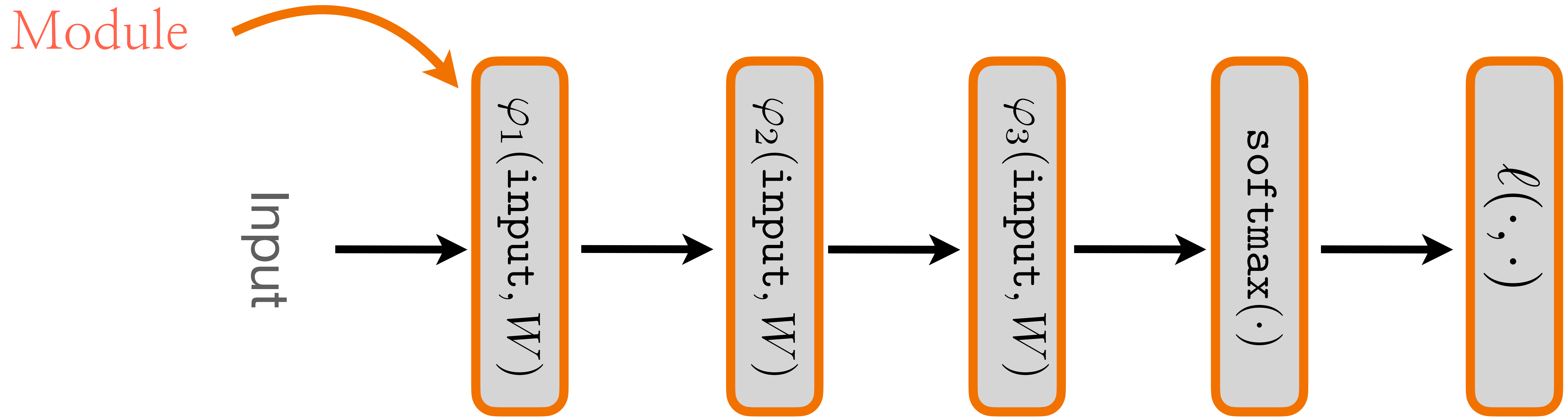


$$\ell\left(\texttt{softmax}\left(\varphi_3(\varphi_2(\varphi_1(x_i, W), W), W\right), \quad y_i\right)$$

Softmax: function that normalizes and outputs a vector of probabilities on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot,\cdot)$

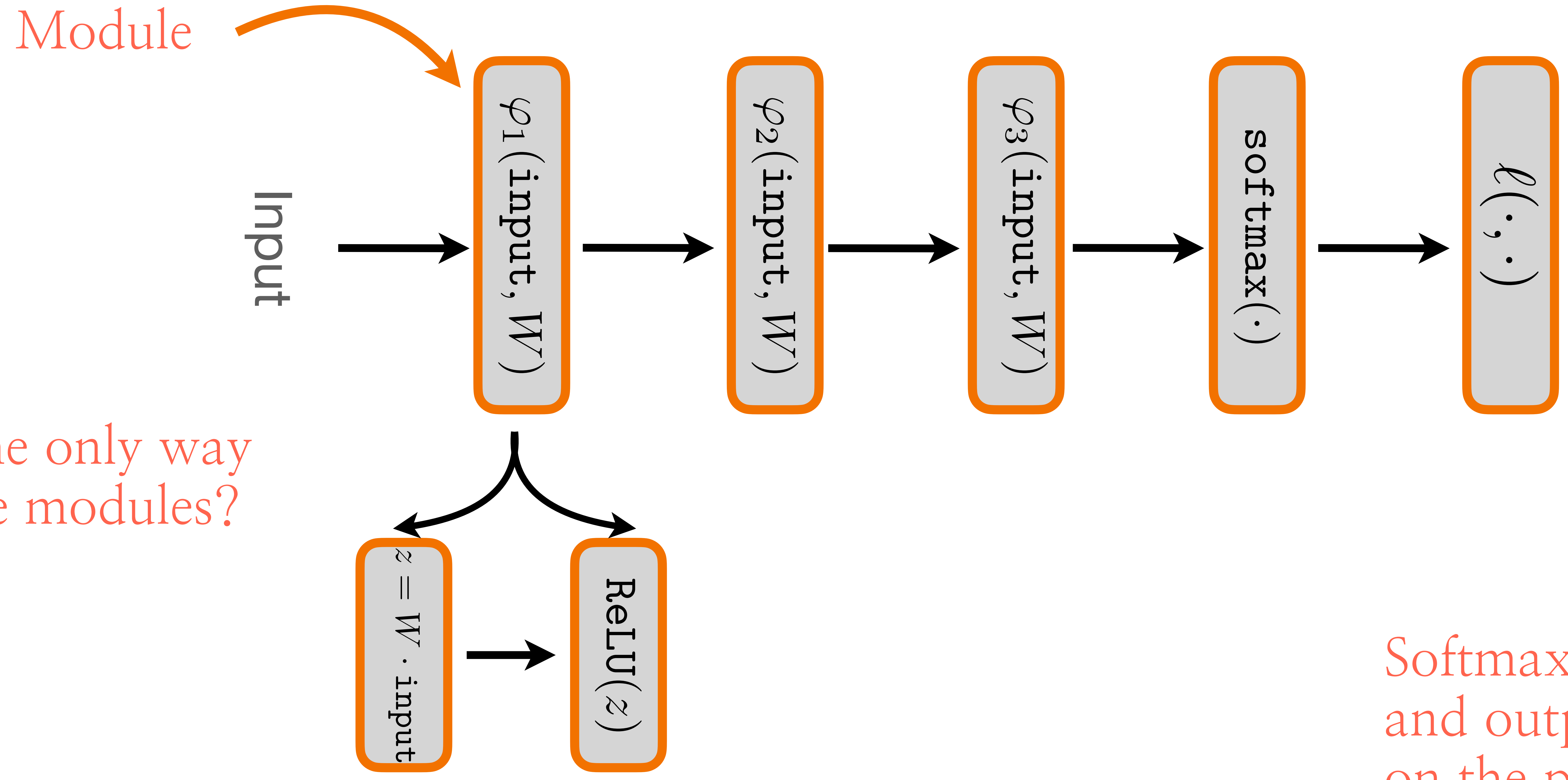


Is this the only way to define modules?

Softmax: function that normalizes and outputs a vector of probabilities on the possible outcomes

Neural networks represented as boxes

– Most used representation of neural networks (but math will get us the details)
– Example: feedforward neural network, with 3 hidden layers, and ReLUs
Softmax before output, loss function $\ell(\cdot,\cdot)$

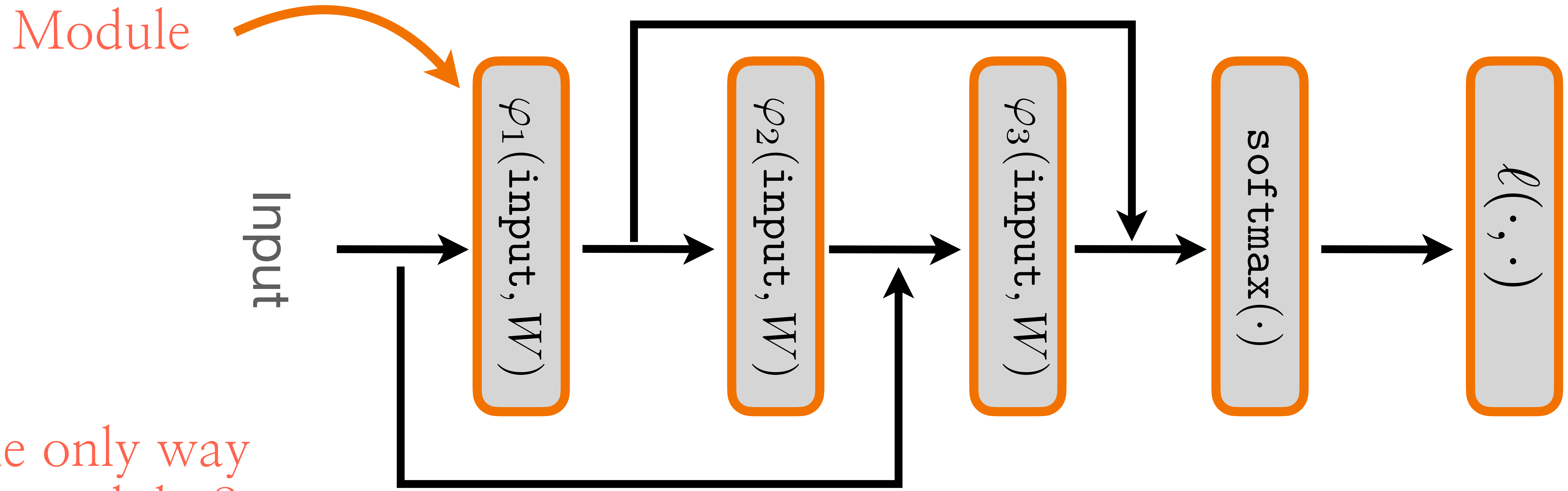


Is this the only way to define modules?

Softmax: function that normalizes and outputs a vector of probabilities on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Example: feedforward neural network, with 3 hidden layers, and ReLUs
  Softmax before output, loss function $\ell(\cdot,\cdot)$



Is this the only way to define modules?

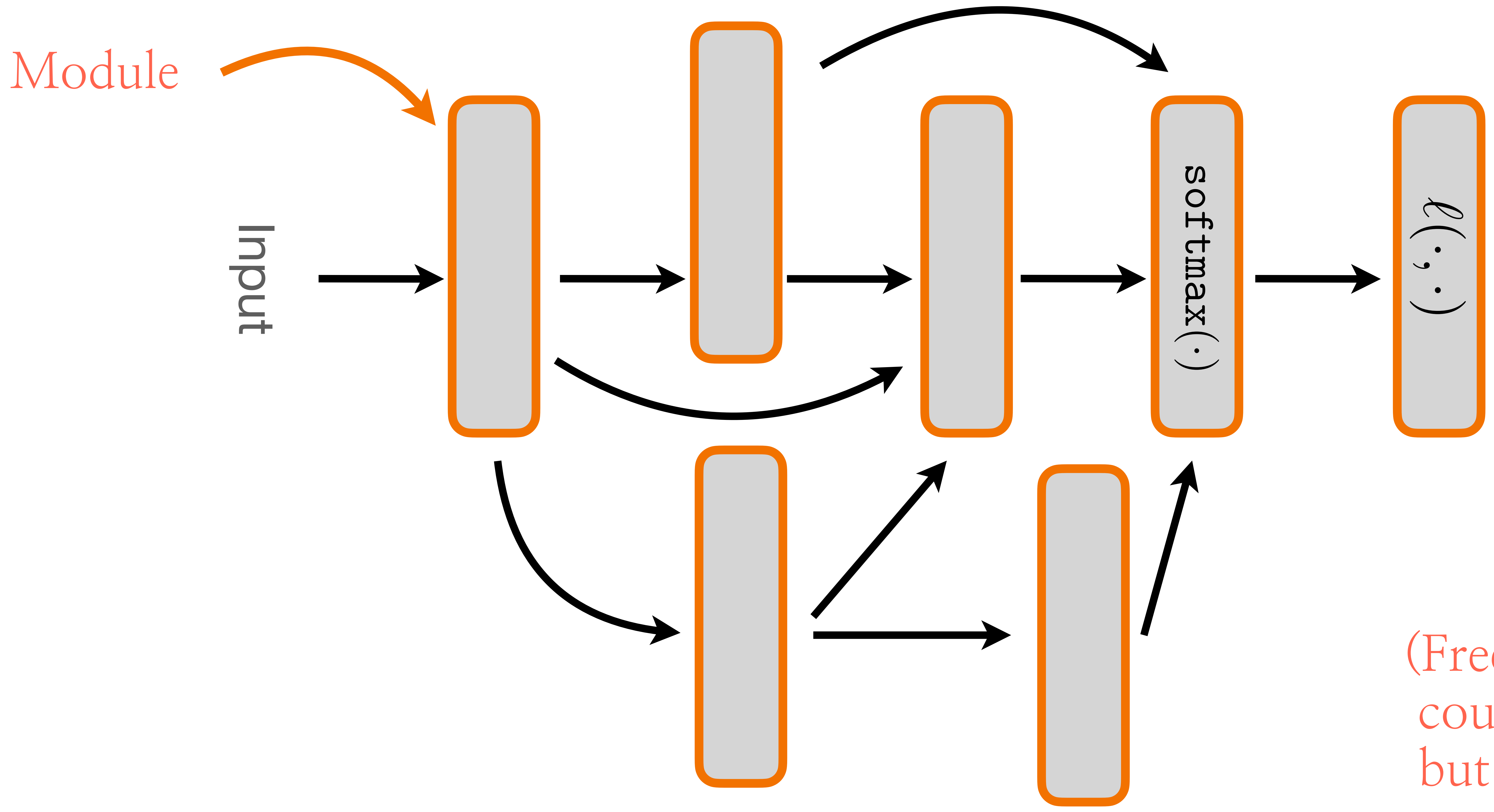
Softmax: function that normalizes and outputs a vector of probabilities on the possible outcomes

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Does something restrict us from using more complicated models? NO!

Neural networks represented as boxes

- Most used representation of neural networks (but math will get us the details)
- Does something restrict us from using more complicated models? NO!



(Free lunch theorem: user bias could help you in one case, but can hurt you in another)

Neural networks represented as boxes

- In an abstract sense, a module is a (smaller) function
  - Receives an input, has variables, provides an output

Neural networks represented as boxes

- In an abstract sense, a module is a (smaller) function
  - Receives an input, has variables, provides an output
- (Desirable) properties of modules:
  - Easy to evaluate
  - Easy to compute its derivatives (mostly first-order / gradients) almost everywhere
  - Efficient implementation
  - Complex enough to represent/learn data well

Neural networks represented as boxes

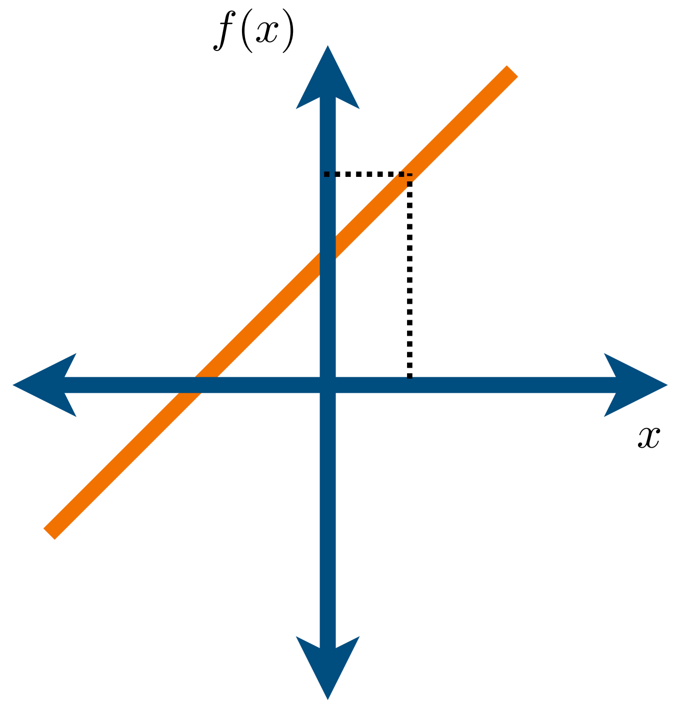
- In an abstract sense, a module is a (smaller) function
  - Receives an input, has variables, provides an output
- (Desirable) properties of modules:
  - Easy to evaluate
  - Easy to compute its derivatives (mostly first−order / gradients) almost everywhere
  - Efficient implementation
  - Complex enough to represent/learn data well
- Examples:

$$z = W \cdot \texttt{input}$$

Neural networks represented as boxes

- In an abstract sense, a module is a (smaller) function
  - Receives an input, has variables, provides an output
- (Desirable) properties of modules:
  - Easy to evaluate
  - Easy to compute its derivatives (mostly first-order / gradients) almost everywhere
  - Efficient implementation
  - Complex enough to represent/learn data well
- Examples:

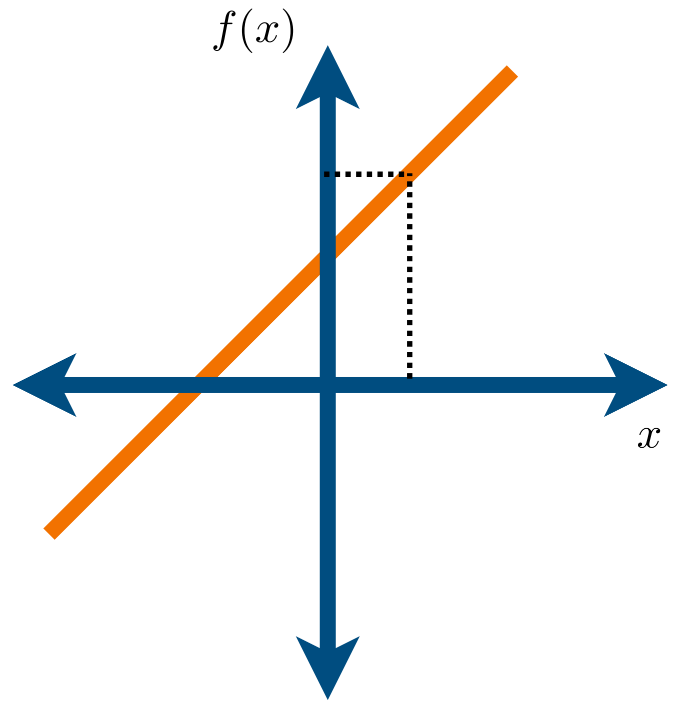
$$z = W \cdot \texttt{input}$$

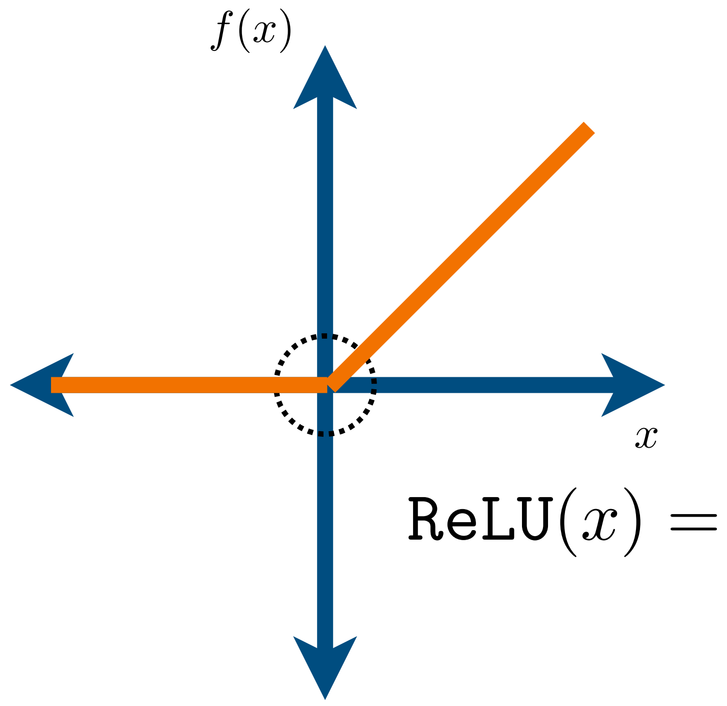


Neural networks represented as boxes

- In an abstract sense, a module is a (smaller) function
  - Receives an input, has variables, provides an output
- (Desirable) properties of modules:
  - Easy to evaluate
  - Easy to compute its derivatives (mostly first-order / gradients) almost everywhere
  - Efficient implementation
  - Complex enough to represent/learn data well
- Examples:

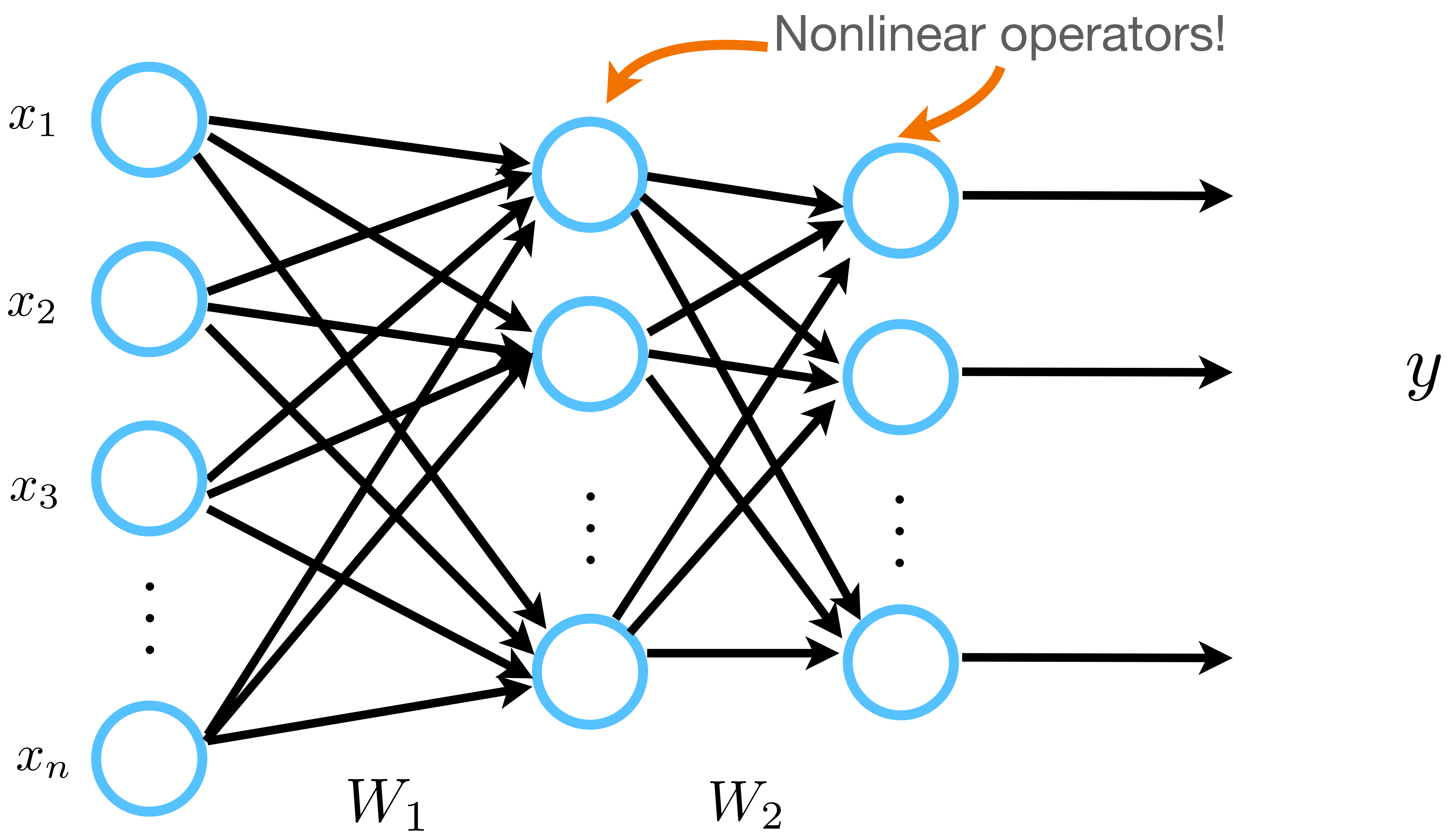
$$z = W \cdot \texttt{input}$$



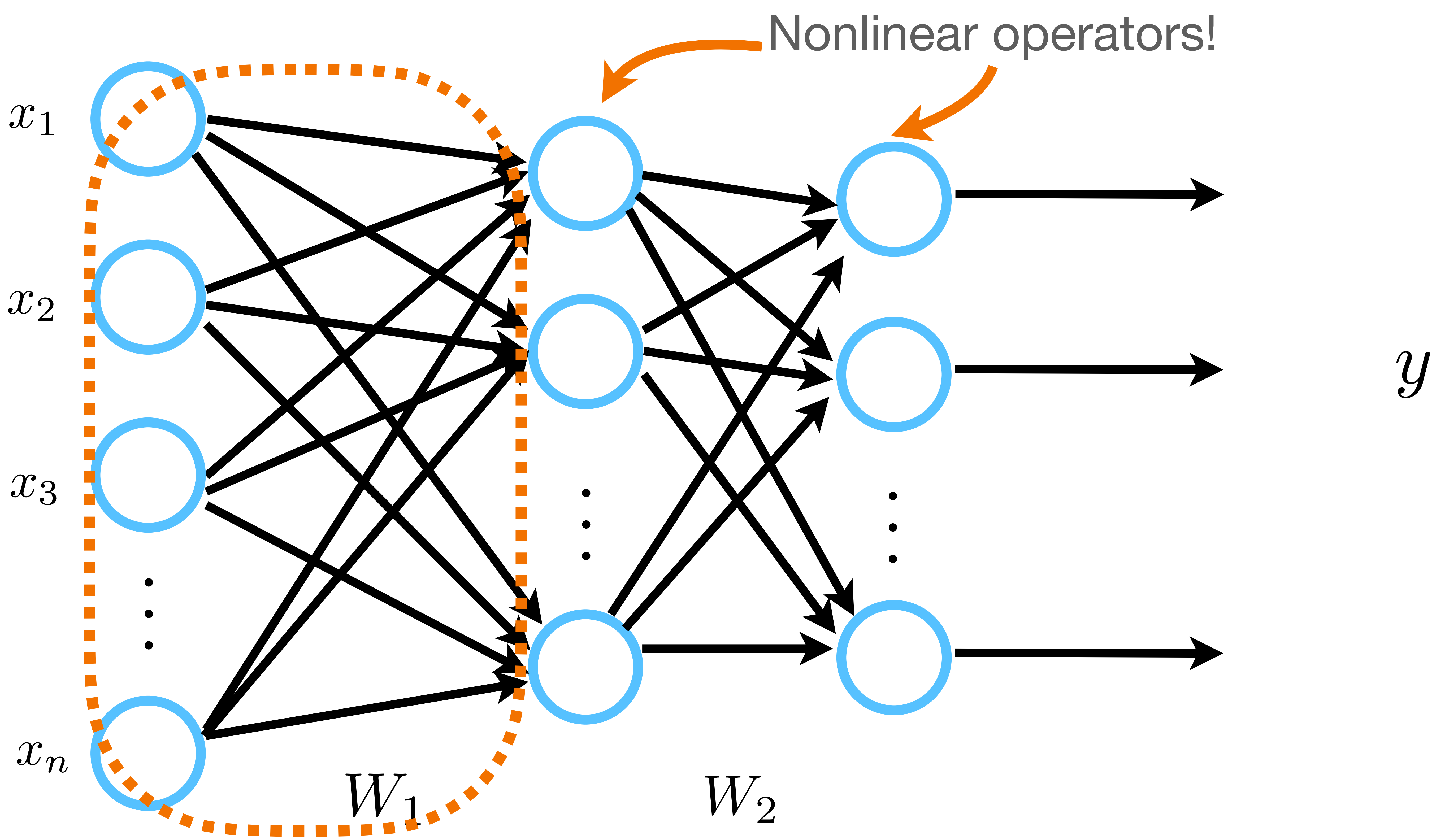


$$\texttt{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Fully connected neural networks or MLPs

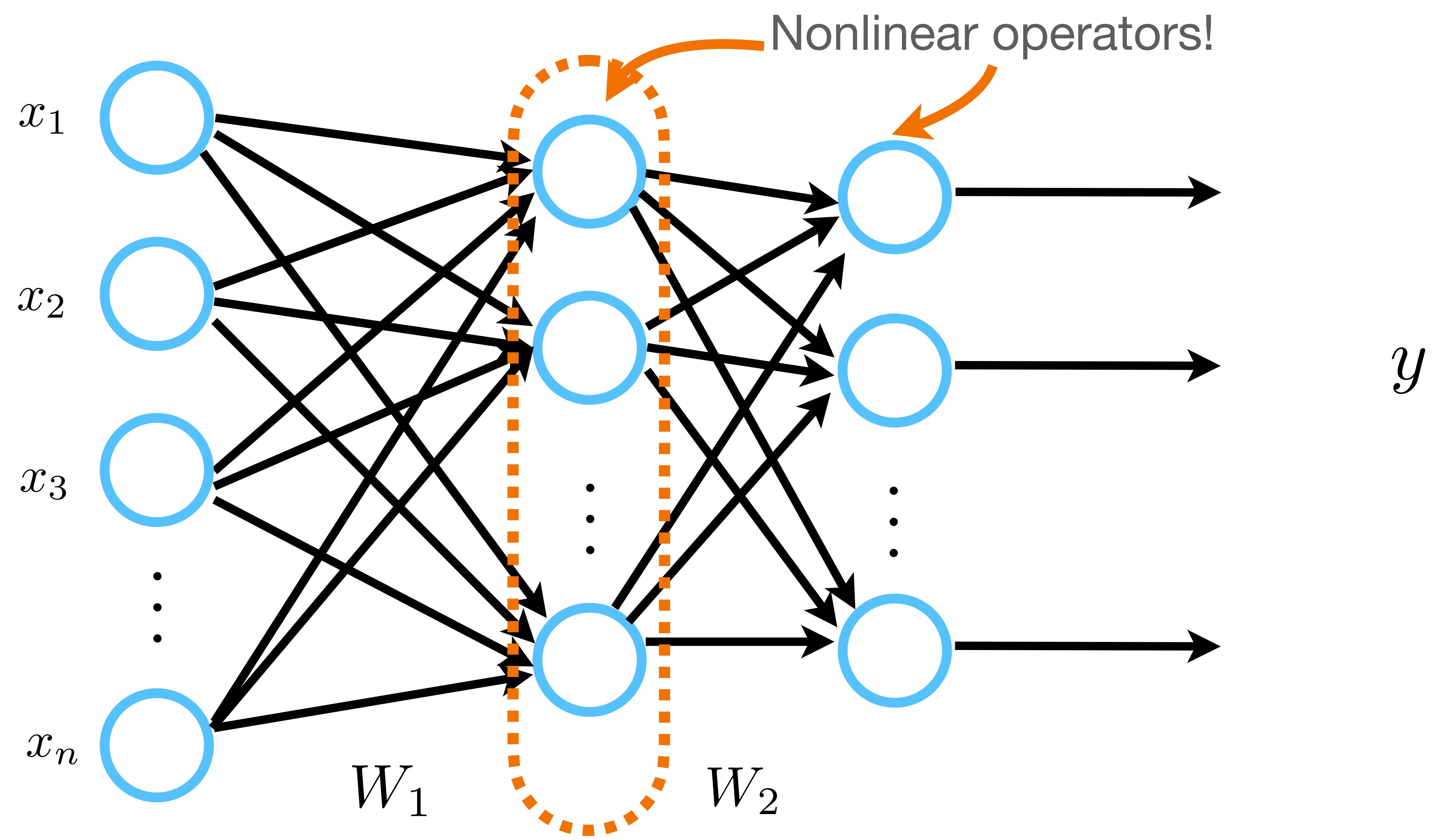


Fully connected neural networks or MLPs



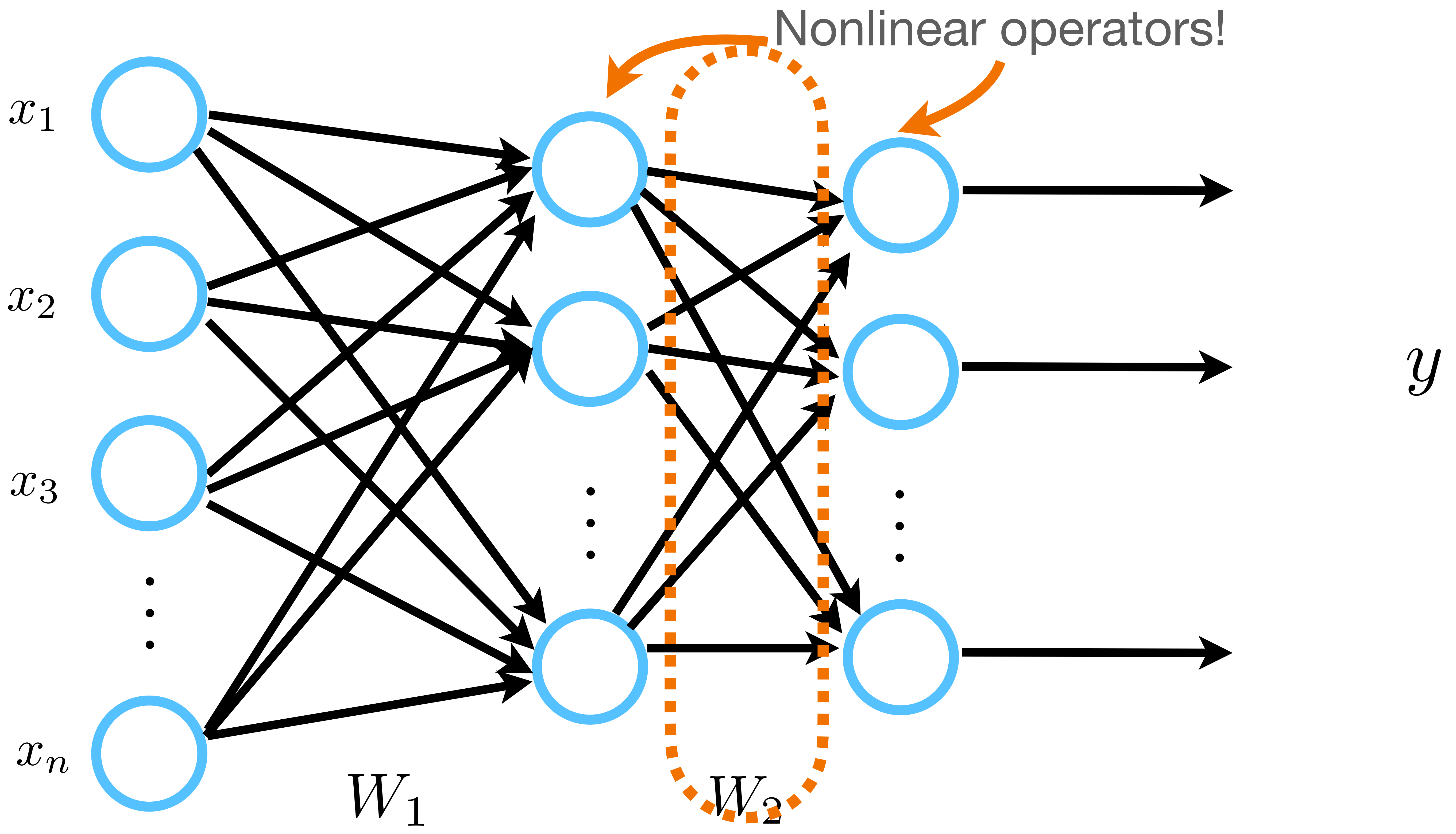
1. $z_1 = W_1 \cdot x$

Fully connected neural networks or MLPs



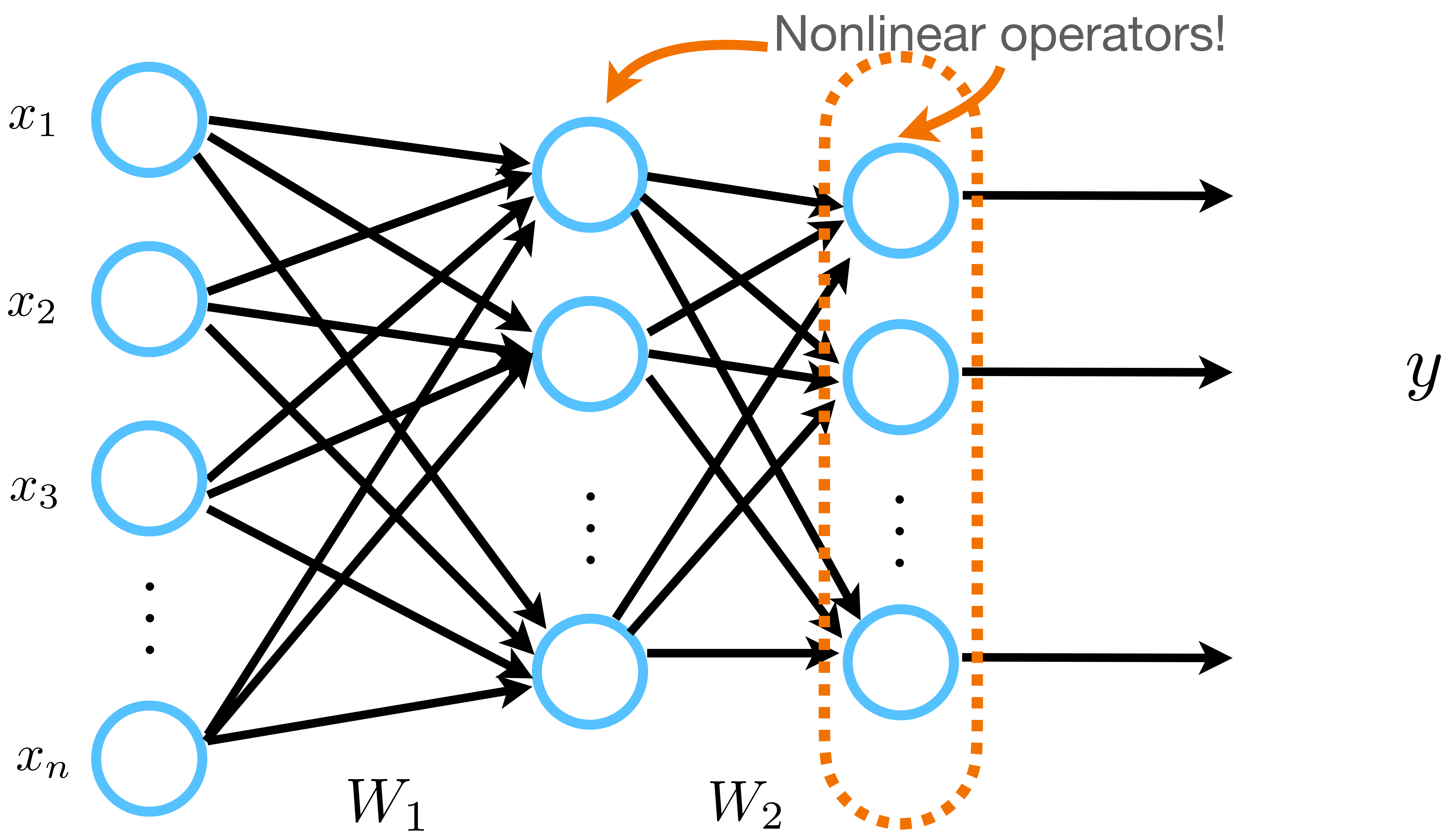
1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$

Fully connected neural networks or MLPs



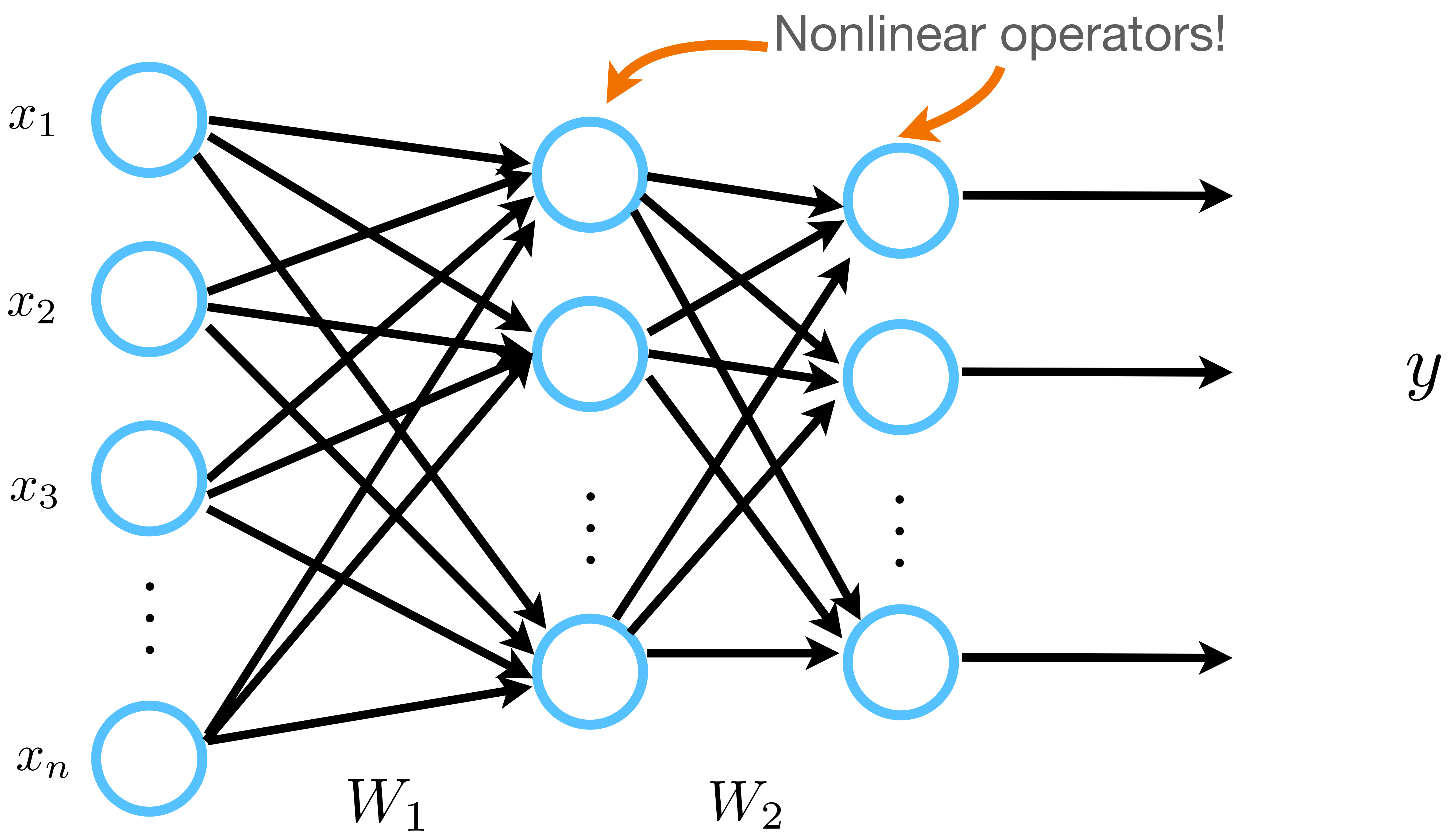
1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$
3. $z_2 = W_2 \cdot \sigma(z_1) = W_2 \cdot \sigma(W_1 \cdot x)$

Fully connected neural networks or MLPs



1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$
3. $z_2 = W_2 \cdot \sigma(z_1) = W_2 \cdot \sigma(W_1 \cdot x)$
4. $\texttt{softmax}(z_2) = \texttt{softmax}(W_2 \cdot \sigma(W_1 \cdot x))$

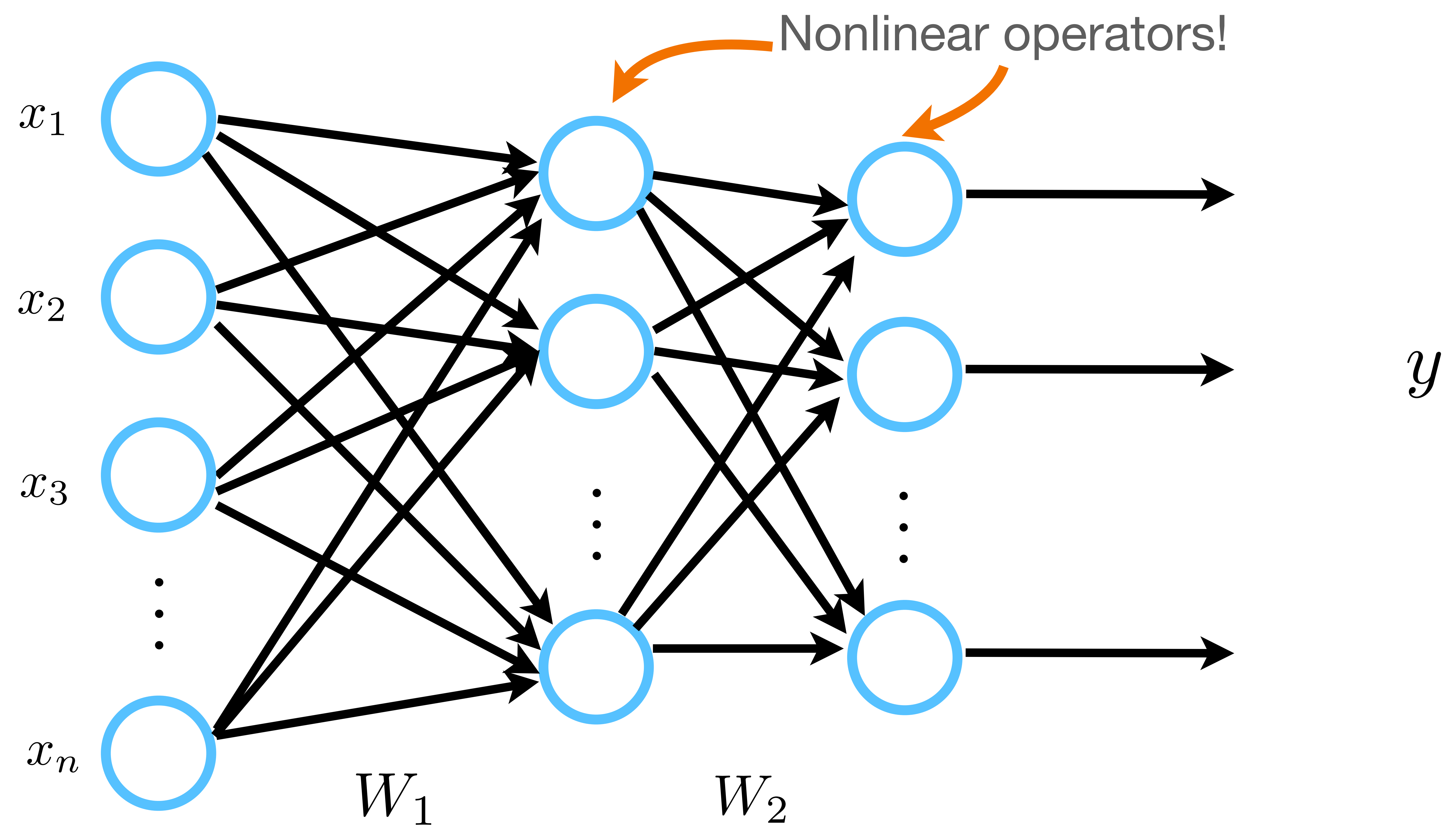
Fully connected neural networks or MLPs



1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$
3. $z_2 = W_2 \cdot \sigma(z_1) = W_2 \cdot \sigma(W_1 \cdot x)$
4. $\texttt{softmax}(z_2) = \texttt{softmax}(W_2 \cdot \sigma(W_1 \cdot x))$

- Nothing more complicated
- Boils down to what modules will be used
- Famous architecture due to simplicity and theoretical guarantees

Fully connected neural networks or MLPs



This sequence of operations is also known as the **forward pass on the neural network**

You can think of forward pass as function evaluation

1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$
3. $z_2 = W_2 \cdot \sigma(z_1) = W_2 \cdot \sigma(W_1 \cdot x)$
4. $\texttt{softmax}(z_2) = \texttt{softmax}(W_2 \cdot \sigma(W_1 \cdot x))$

- Nothing more complicated
- Boils down to what modules will be used
- Famous architecture due to simplicity and theoretical guarantees

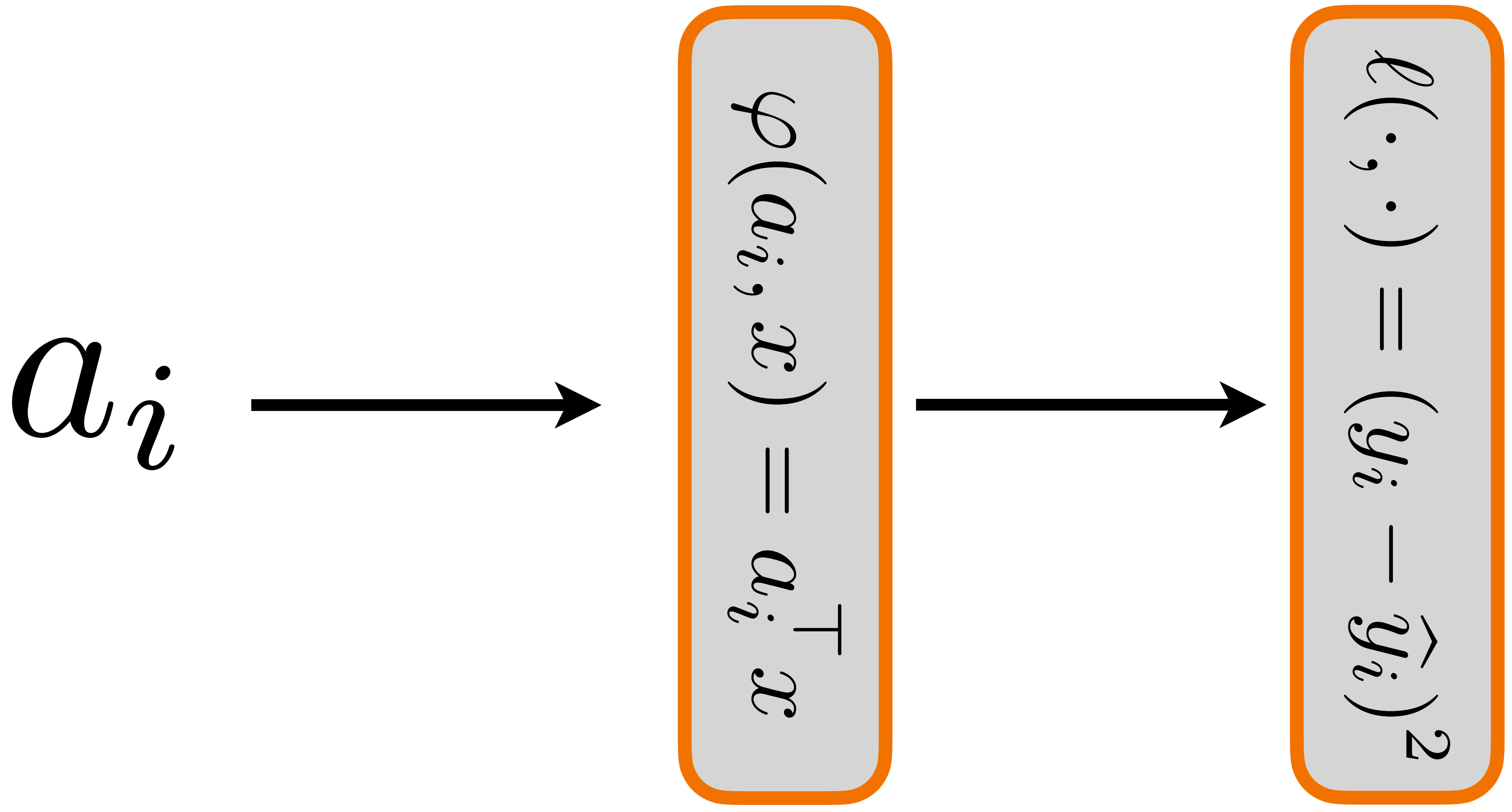
Motivation: Gradient descent for least-squares

$$\min_x f(x) := \frac{1}{n}\sum_{i=1}^{n}(y_i - a_i^\top x)^2$$

Motivation: Gradient descent for least-squares

$$\min_x f(x) := \frac{1}{n}\sum_{i=1}^{n}(y_i - a_i^\top x)^2$$

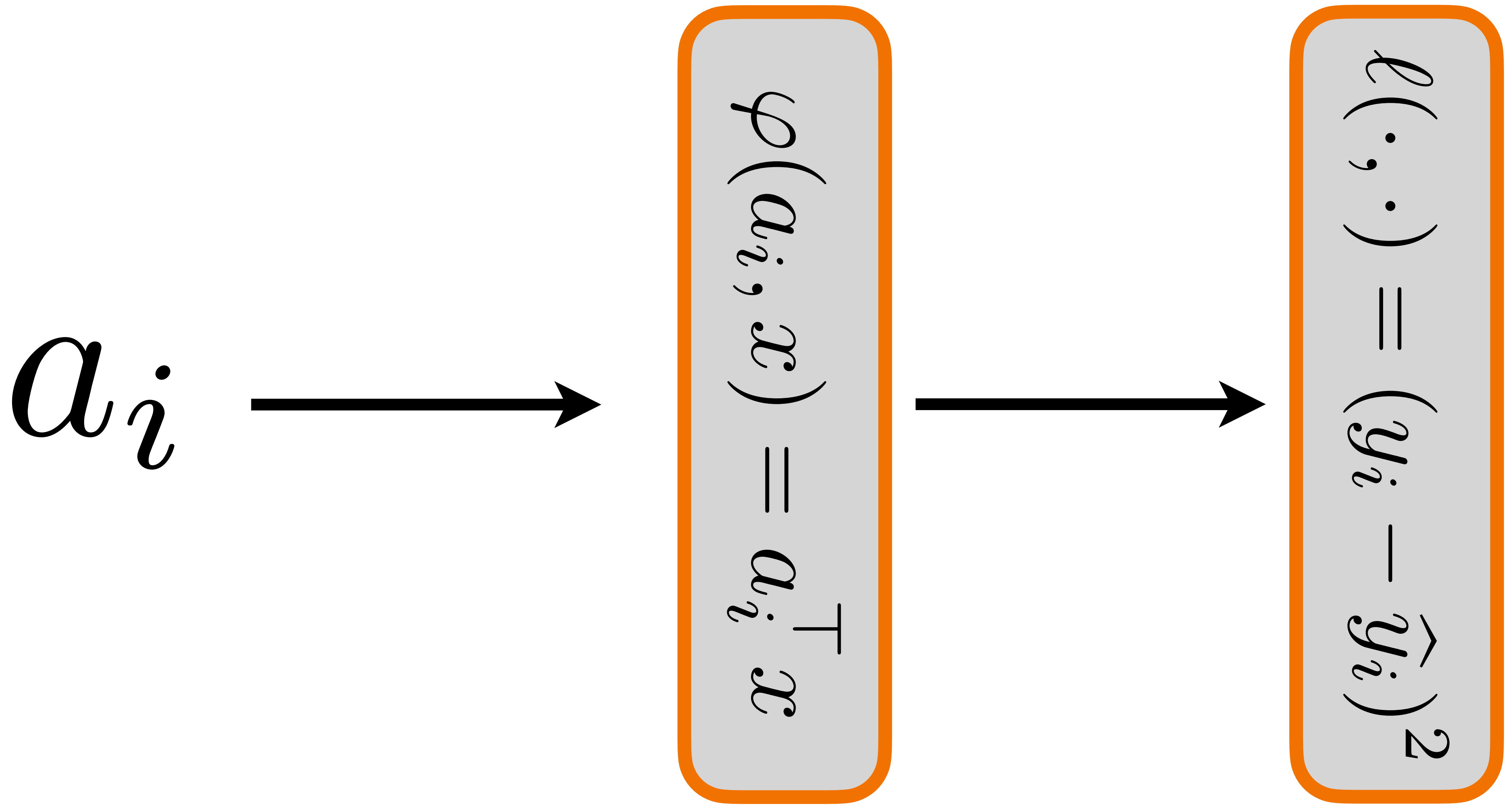
– Using modules:



Motivation: Gradient descent for least-squares

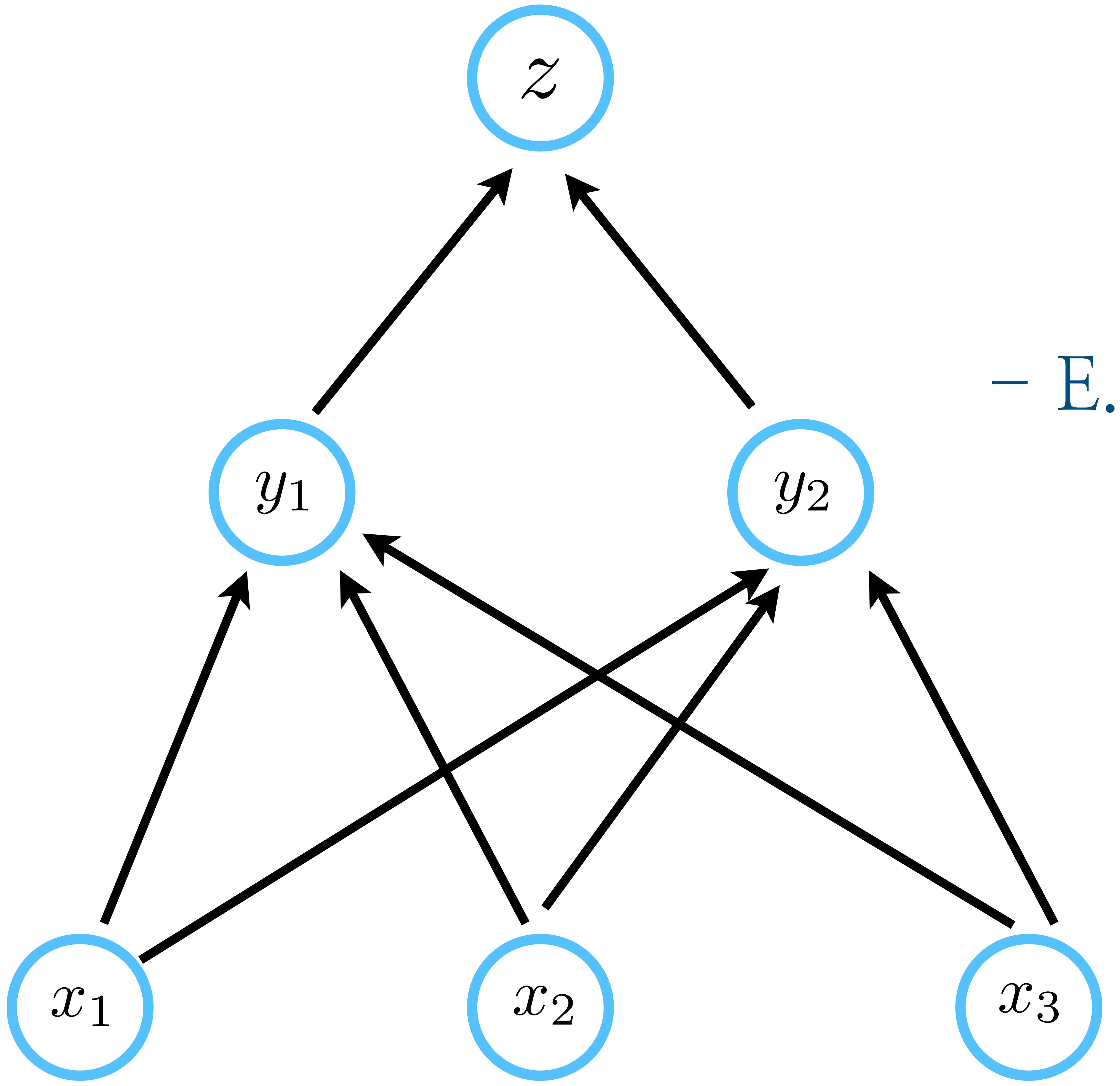
$$\min_x f(x) := \frac{1}{n}\sum_{i=1}^{n}(y_i - a_i^\top x)^2$$

– Using modules:



$$\begin{aligned}
\nabla f_i(x) &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial x} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} \cdot \frac{\partial \widehat{y}_i}{\partial x} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} \cdot \frac{a_i^\top x}{\partial x} \\
&= -2a_i(y_i - a_i^\top x)
\end{aligned}$$

(chain rule of derivatives)



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

– E.g.,

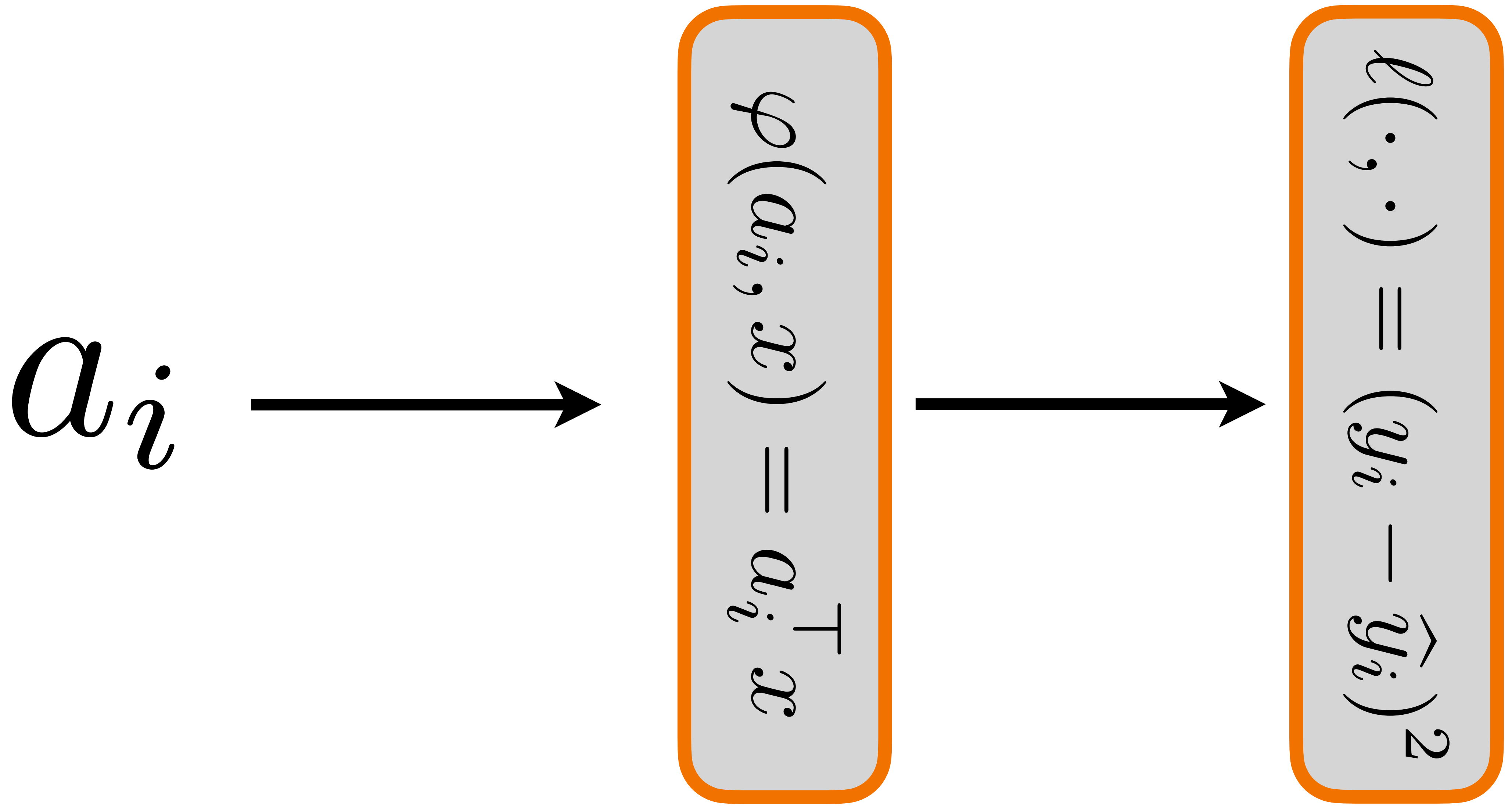
$$\frac{\partial z}{\partial x_3} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_3} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_3}$$

(Follow all relevant paths)

Motivation: Gradient descent for least-squares

$$\min_x f(x) := \frac{1}{n}\sum_{i=1}^{n}(y_i - a_i^\top x)^2$$

- Using modules:



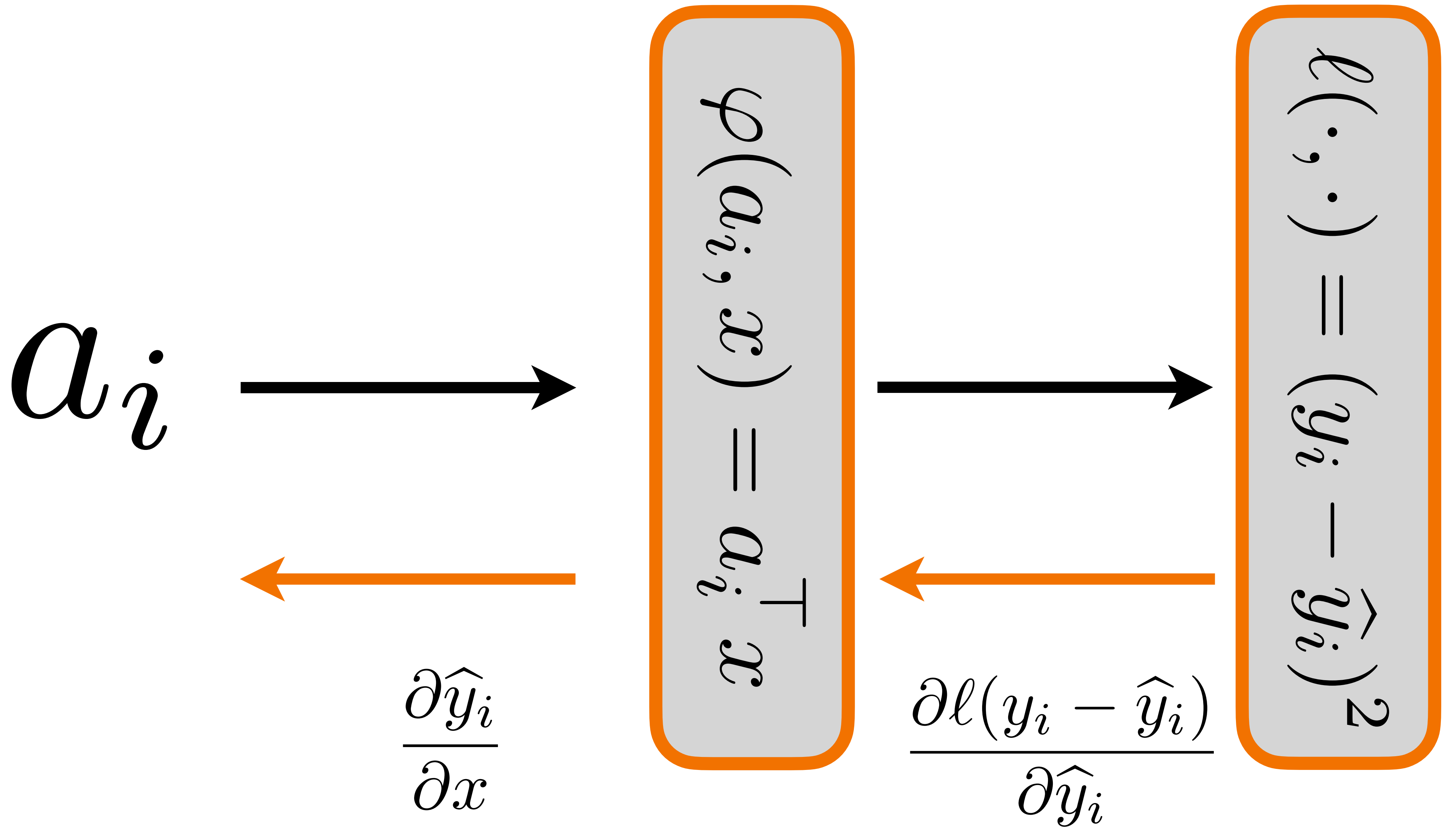
$$\begin{aligned}
\nabla f_i(x) &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial x} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} \cdot \frac{\partial \widehat{y}_i}{\partial x} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} \cdot \frac{a_i^\top x}{\partial x} \\
&= -2a_i(y_i - a_i^\top x)
\end{aligned}$$

(chain rule of derivatives)

Motivation: Gradient descent for least-squares

$$\min_x f(x) := \frac{1}{n}\sum_{i=1}^{n}(y_i - a_i^\top x)^2$$

- Using modules:



(backward pass on modules!)

$$\begin{aligned}
\nabla f_i(x) &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial x} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} \cdot \frac{\partial \widehat{y}_i}{\partial x} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} \cdot \frac{a_i^\top x}{\partial x} \\
&= -2a_i(y_i - a_i^\top x)
\end{aligned}$$

(chain rule of derivatives)

Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n}\sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n}\sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

– Using modules:

Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n}\sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

– Using modules:

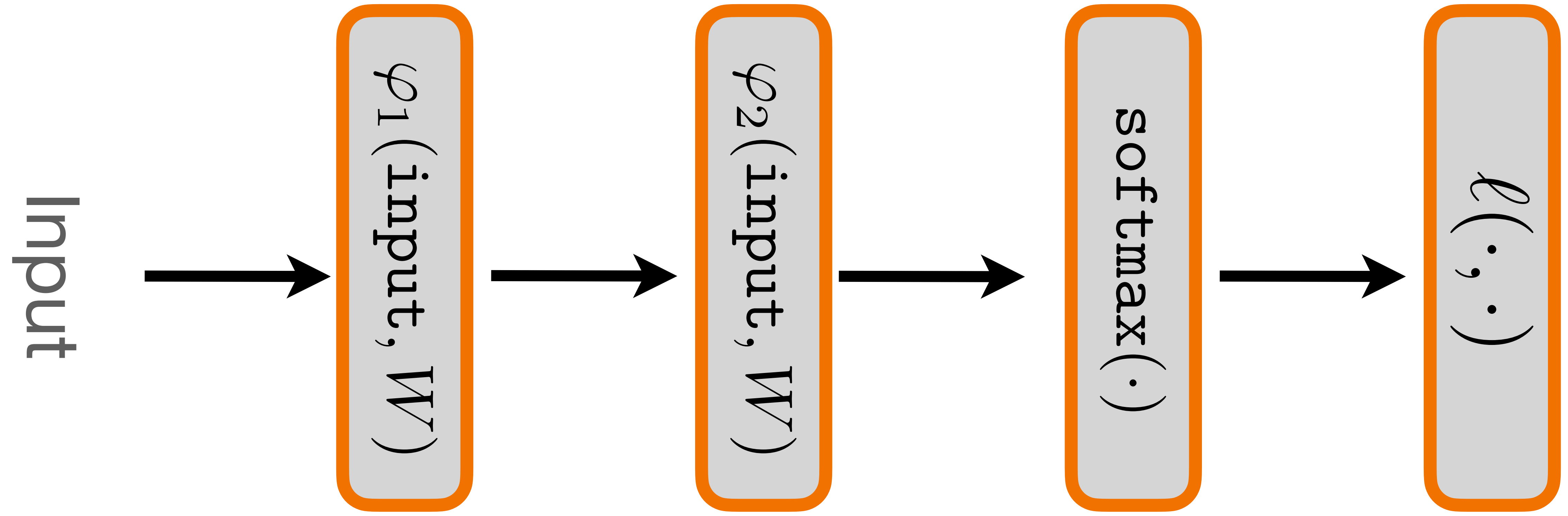
$$\varphi_1(\texttt{input}, W) = \sigma(W_1 \cdot x_i)$$
$$\varphi_2(\texttt{input}, W) = \sigma(W_2 \cdot \varphi_2(\cdot))$$

Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n}\sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

- Using modules:

$\varphi_1(\texttt{input}, W) = \sigma(W_1 \cdot x_i)$
$\varphi_2(\texttt{input}, W) = \sigma(W_2 \cdot \varphi_2(\cdot))$



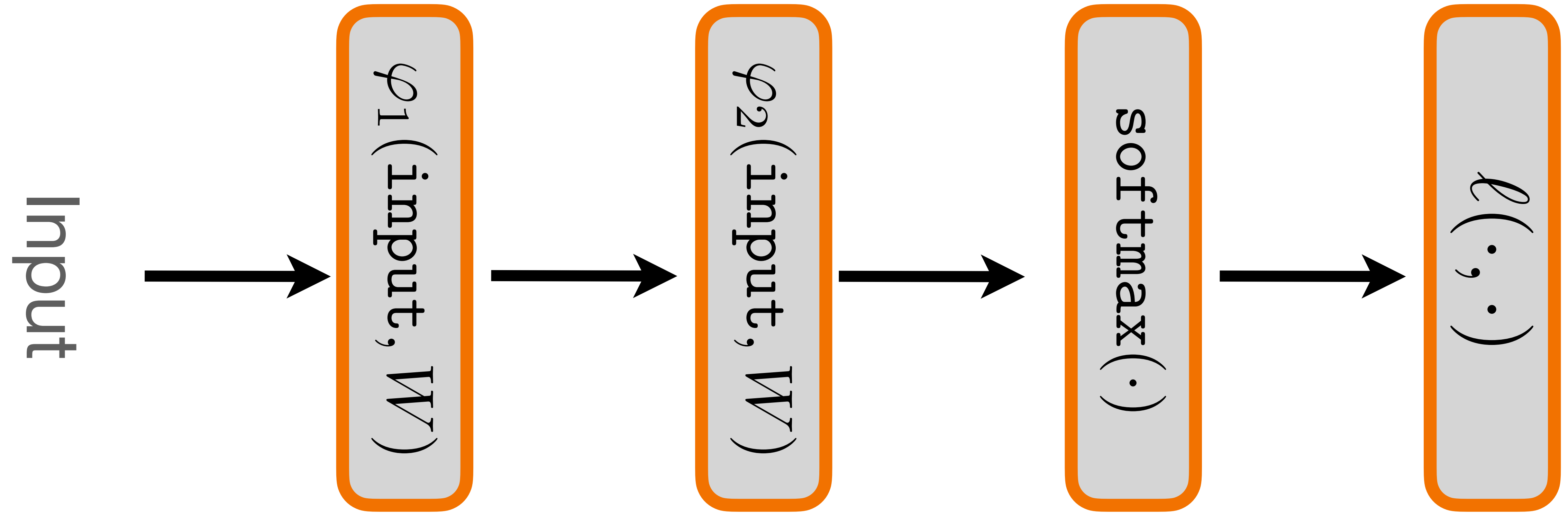
Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n} \sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

– Using modules:

$\varphi_1(\texttt{input}, W) = \sigma(W_1 \cdot x_i)$

$\varphi_2(\texttt{input}, W) = \sigma(W_2 \cdot \varphi_2(\cdot))$



$$\begin{aligned}
\nabla f_i(x) &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial W} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial W} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial \varphi(\cdot, W_2)} \cdot \frac{\partial \varphi(\cdot, W_2)}{\partial W} \\
&= \cdots
\end{aligned}$$

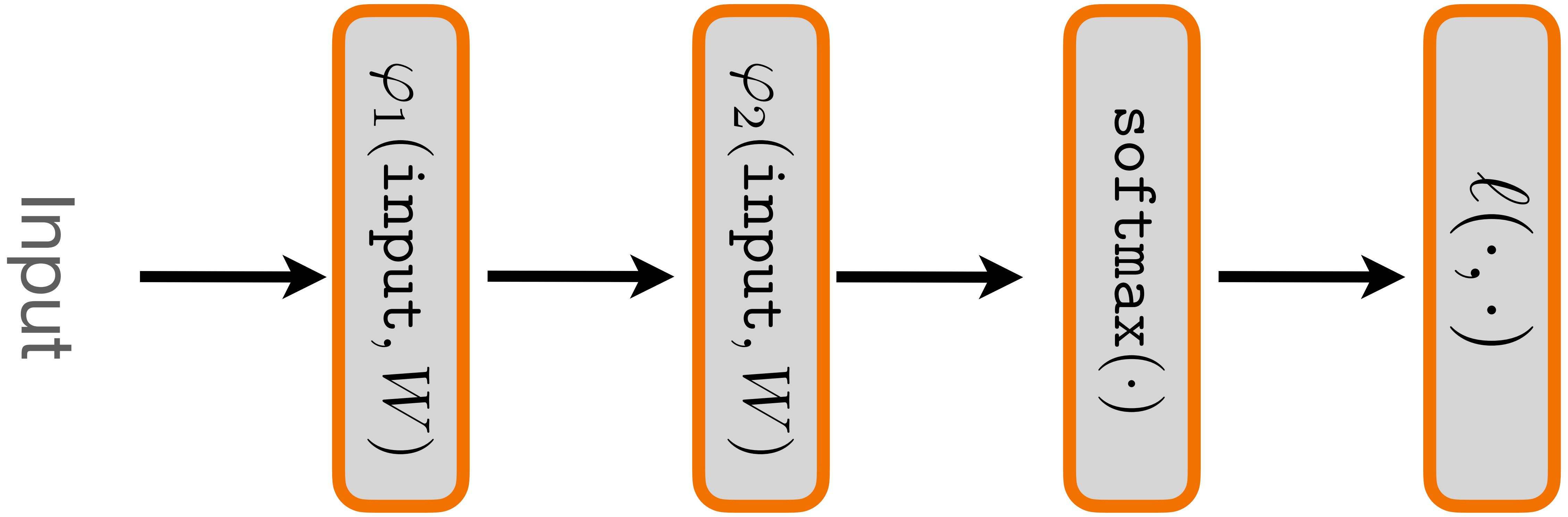
(chain rule of derivatives)

Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n} \sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

– Using modules:

$$\varphi_1(\texttt{input}, W) = \sigma(W_1 \cdot x_i)$$
$$\varphi_2(\texttt{input}, W) = \sigma(W_2 \cdot \varphi_2(\cdot))$$



$$\begin{aligned}
\nabla f_i(x) &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial W} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial W} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial \varphi(\cdot, W_2)} \cdot \frac{\partial \varphi(\cdot, W_2)}{\partial W} \\
&= \cdots
\end{aligned}$$

(chain rule of derivatives)

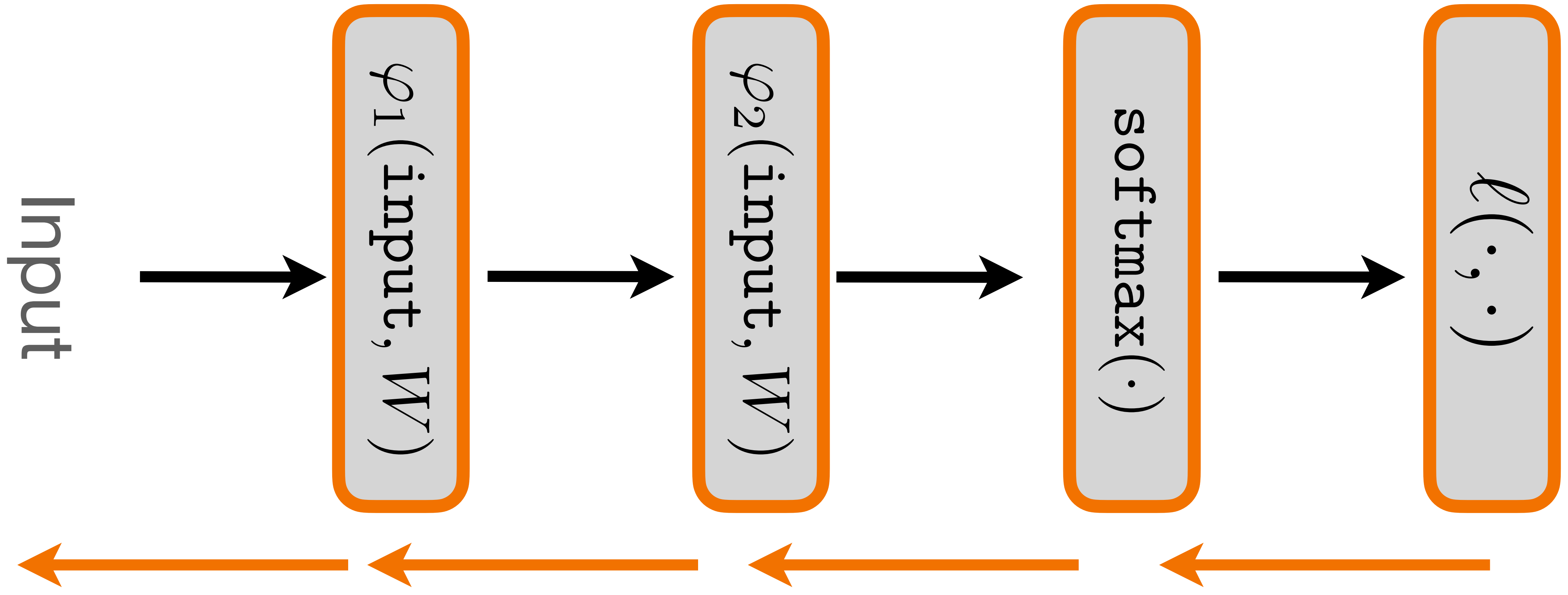
Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n}\sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

– Using modules:

$\varphi_1(\texttt{input}, W) = \sigma(W_1 \cdot x_i)$

$\varphi_2(\texttt{input}, W) = \sigma(W_2 \cdot \varphi_2(\cdot))$



(backward pass on modules!)

$$\begin{aligned}\nabla f_i(x) &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial W} \\ &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial W} \\ &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial \varphi(\cdot, W_2)} \cdot \frac{\partial \varphi(\cdot, W_2)}{\partial W} \\ &= \cdots\end{aligned}$$

(chain rule of derivatives)

Backpropagation = Gradient descent

(Just done efficiently on graphs, without redoing calculations)

Conclusion

- We have set up background of smooth optimization
- We have provided the first convergence rate result, and defined different convergence rates that could be attainable

Conclusion

– We have set up background of smooth optimization

– We have provided the first convergence rate result, and defined different convergence rates that could be attainable

Next lecture

– Brief introduction to convex optimization and related topics