Quebec
Artificial
Intelligence
Institute

Mila

Optimization
in deep learning

Gaétan Marceau Caron
Jeremy Pinto
Applied research scientists, Mila
gaetan.marceau.caron@mila.quebec
jeremy.pinto@mila.quebec

Ingredients for supervised deep learning

- A task definition
- An evaluation metric
- A large amount of high-quality labeled data (>100 000)
- A learning algorithm:
  - End-to-end differentiable computational graph
  - A gradient calculation algorithm: backpropagation
  - **A parameter optimizer**

Ingredients for supervised deep learning

- A task definition
- **An evaluation metric**: we want to maximize the model performance.
- A large amount of high-quality labeled data (>100 000)
- A learning algorithm:
    - End-to-end differentiable computational graph
    - A gradient calculation algorithm: backpropagation
    - **A parameter optimizer**

Why optimization is important in ML?

- We want to maximize the model performance evaluated by the metric.
- Usually, the evaluation metric is not differentiable (e.g., accuracy).
- We need a differentiable surrogate function for training the model.

Objectives of the presentation

Give the intuition to the basic concepts of optimization for deep learning.

Three concepts to remember:

- Preconditioning
- Momentum
- Stochastic optimization

Global minimum

x^* is a *global minimum* of $f : \mathbb{R}^n \rightarrow \mathbb{R}$ iif

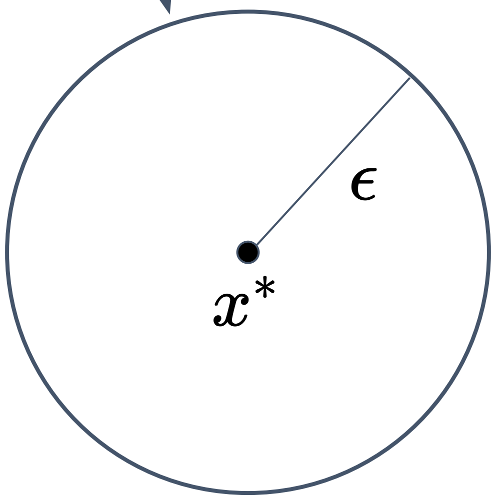
$$\forall x \in \mathbb{R}^n, f(x^*) \leq f(x)$$

Mila

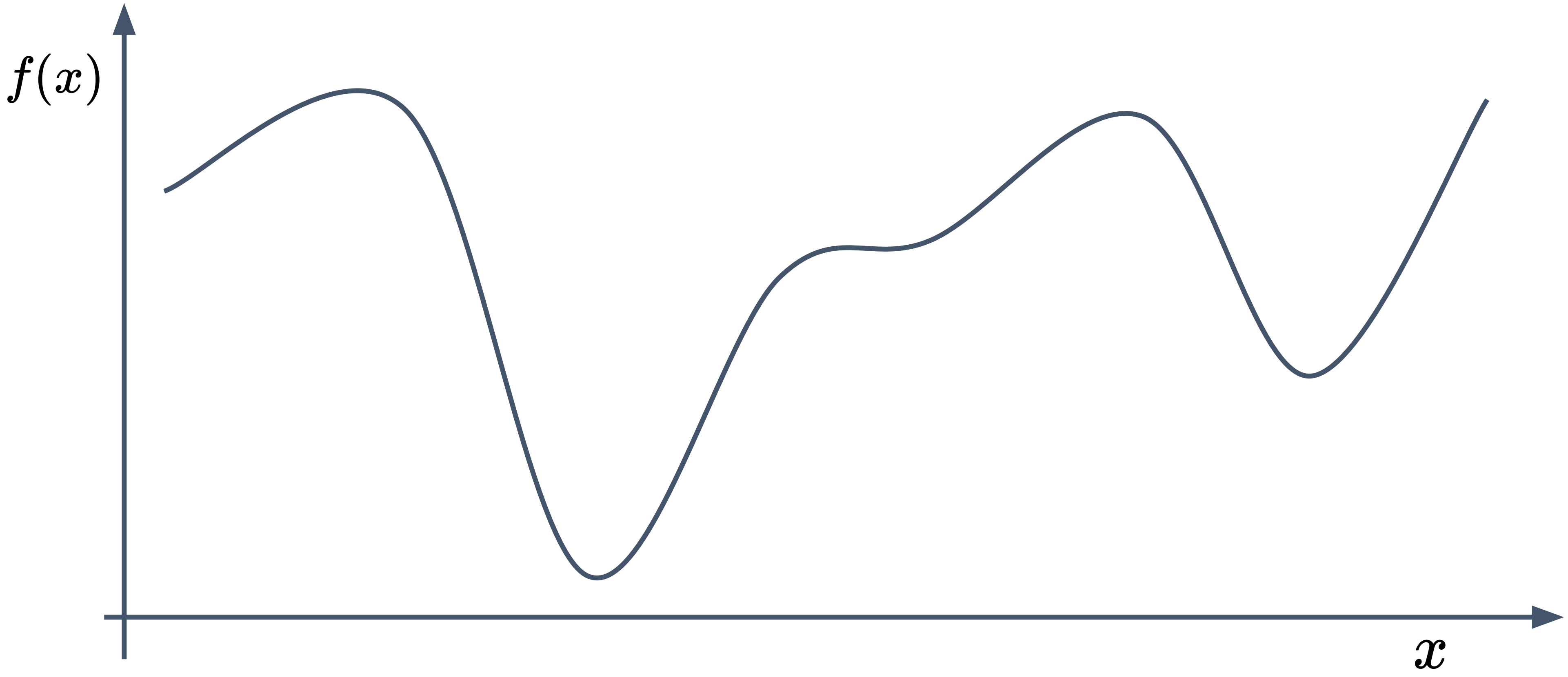
Local minimum

x^* is a *local minimum* of $f : \mathbb{R}^n \to \mathbb{R}$ iif

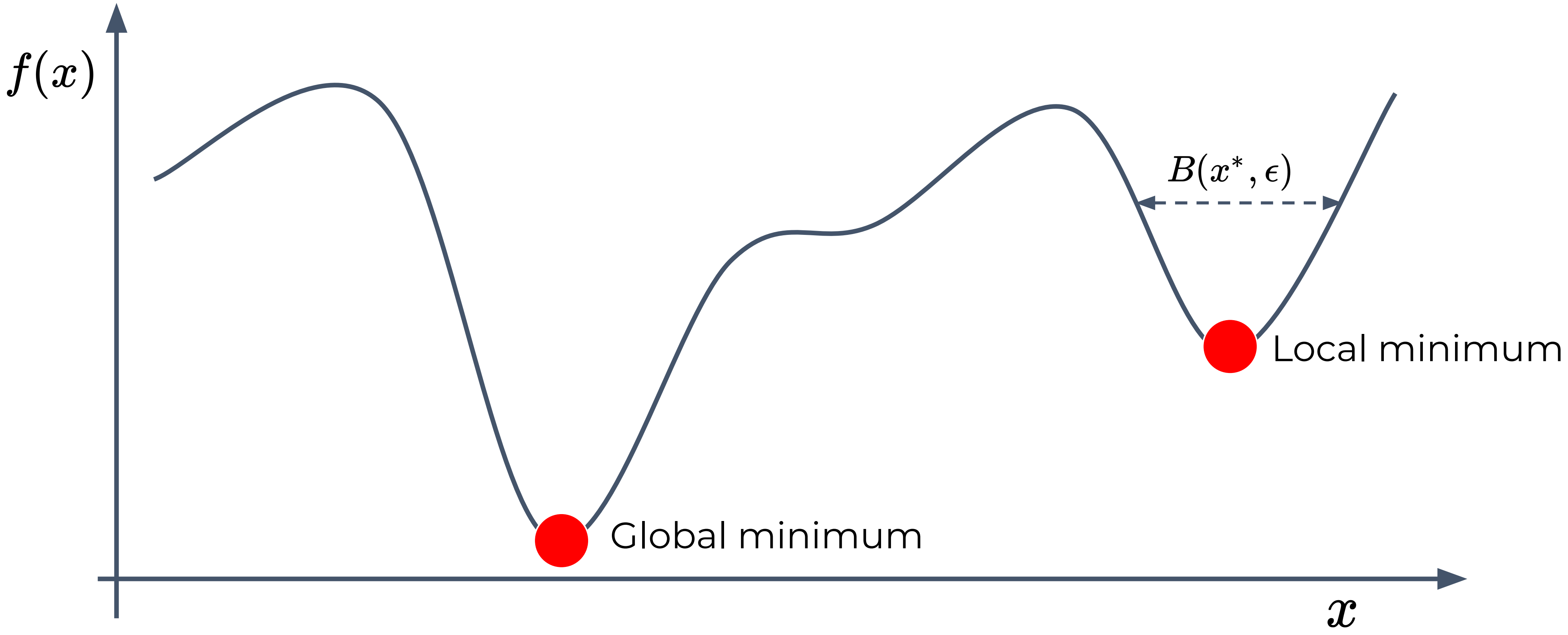
$$\exists \epsilon > 0, \forall x \in B(x^*, \epsilon), f(x^*) \leq f(x)$$



Example in 1D



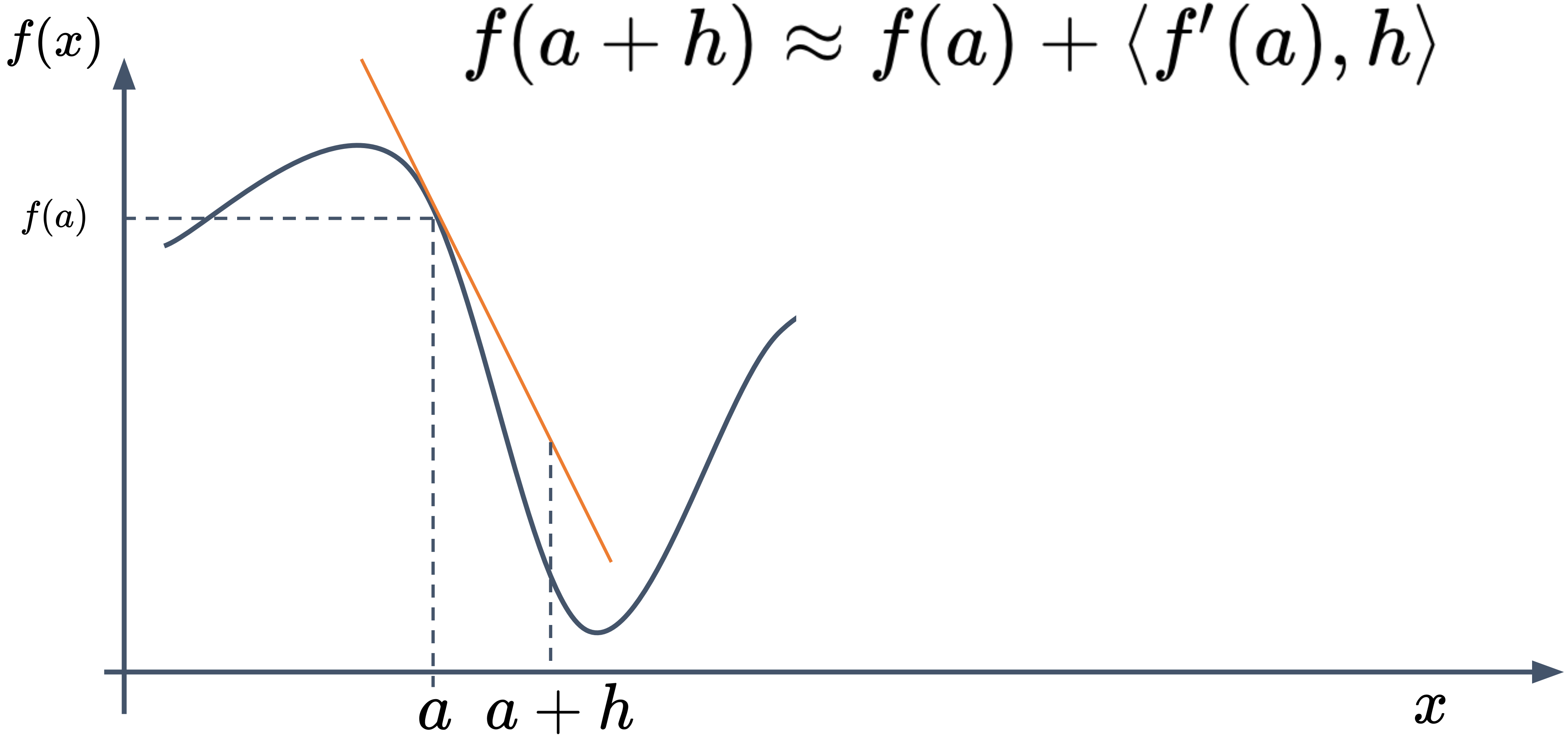
Example



Mila

Linear approximation

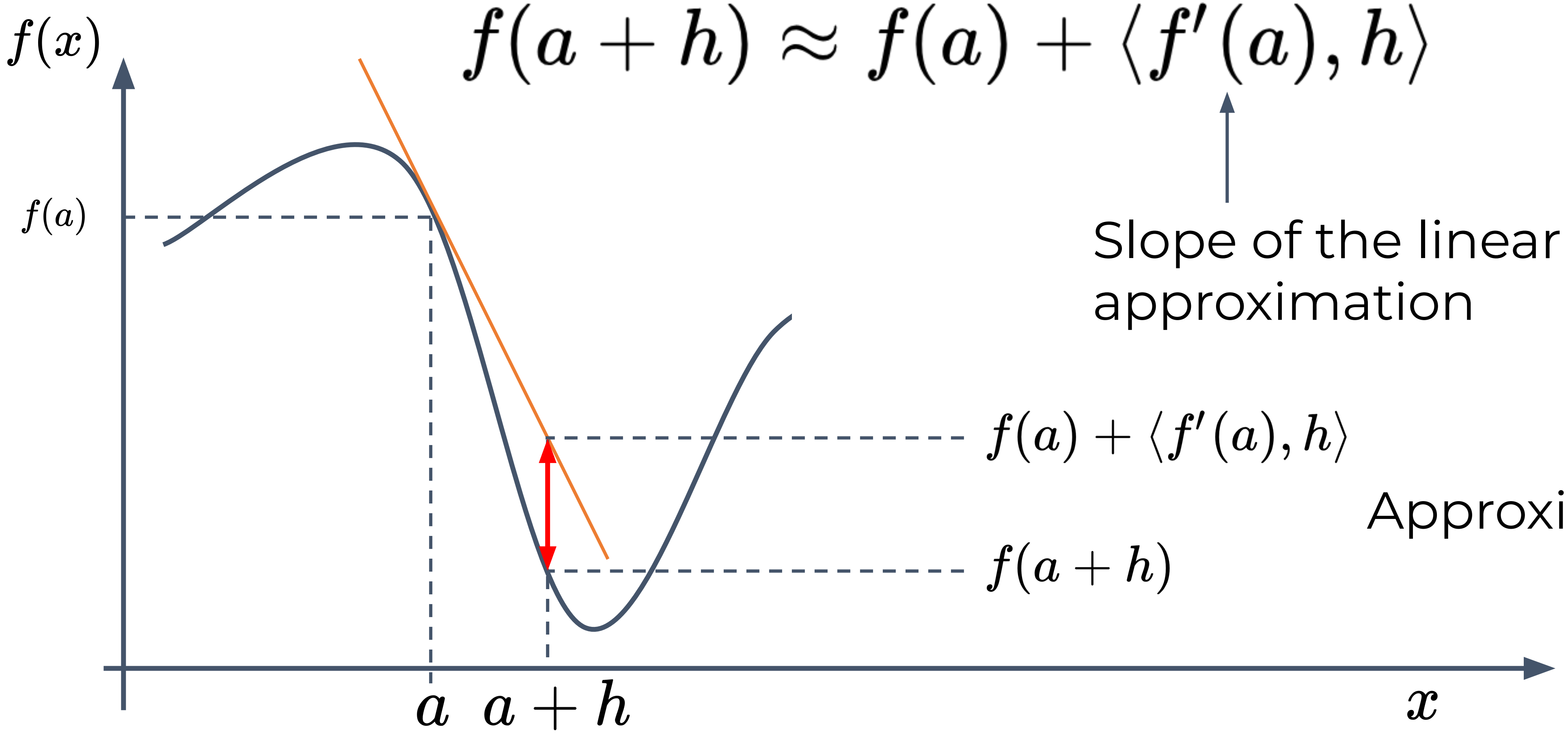
$$f(a+h) \approx f(a) + \langle f'(a), h \rangle$$



Linear approximation

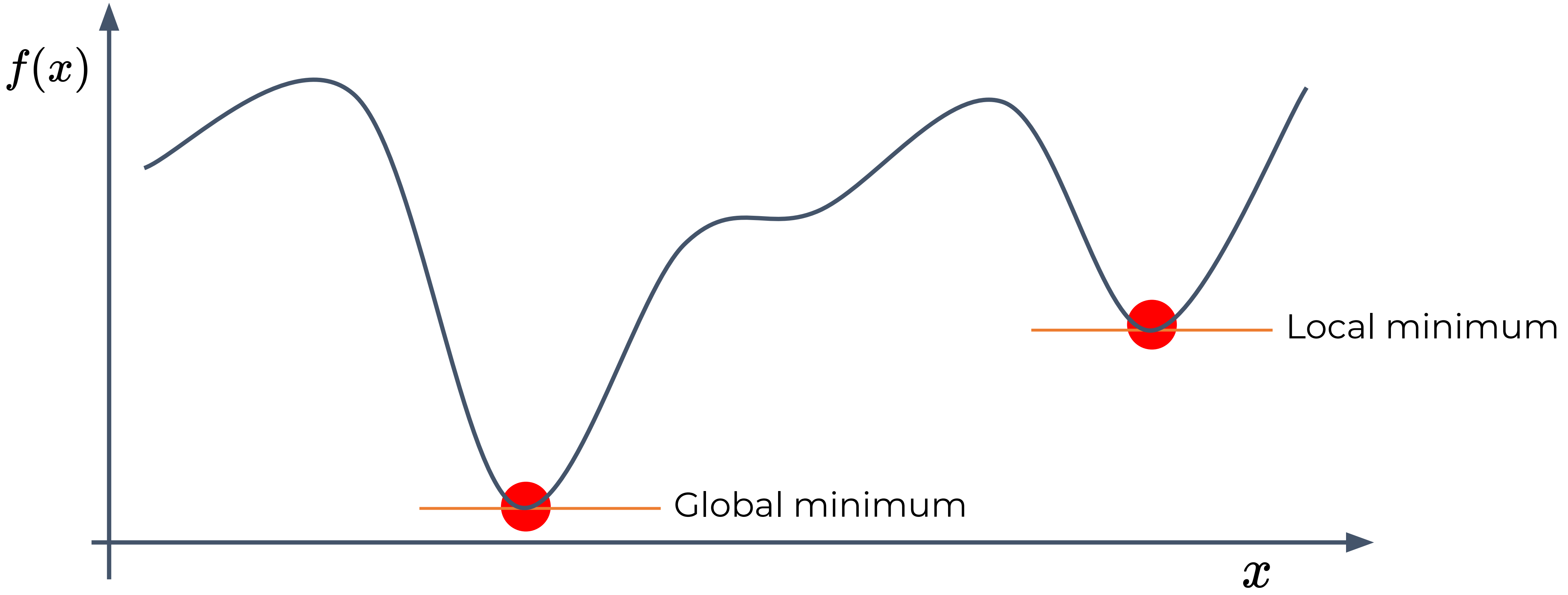
$$f(a+h) \approx f(a) + \langle f'(a), h \rangle$$

Slope of the linear approximation



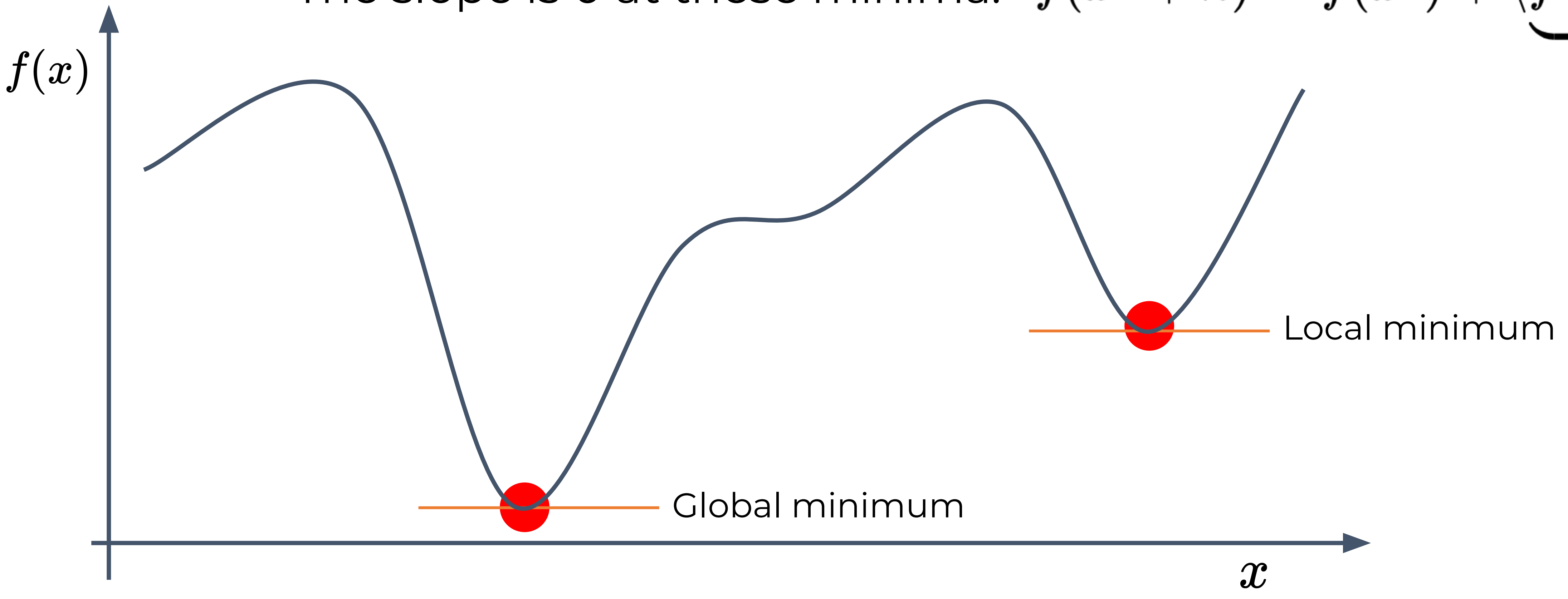
Approximation error

How to find extrema?



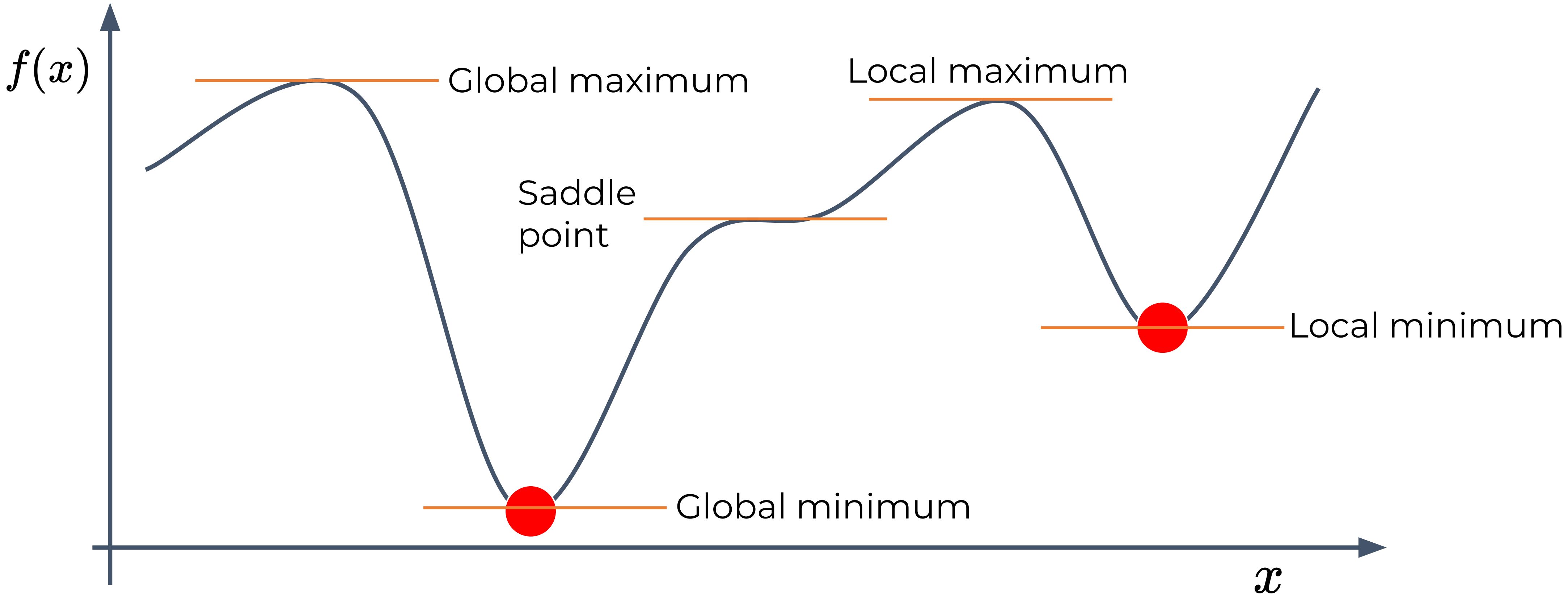
How to find extrema?

The slope is 0 at these minima: $f(x^* + h) \approx f(x^*) + \langle \underbrace{f'(x^*)}_{0}, h \rangle$



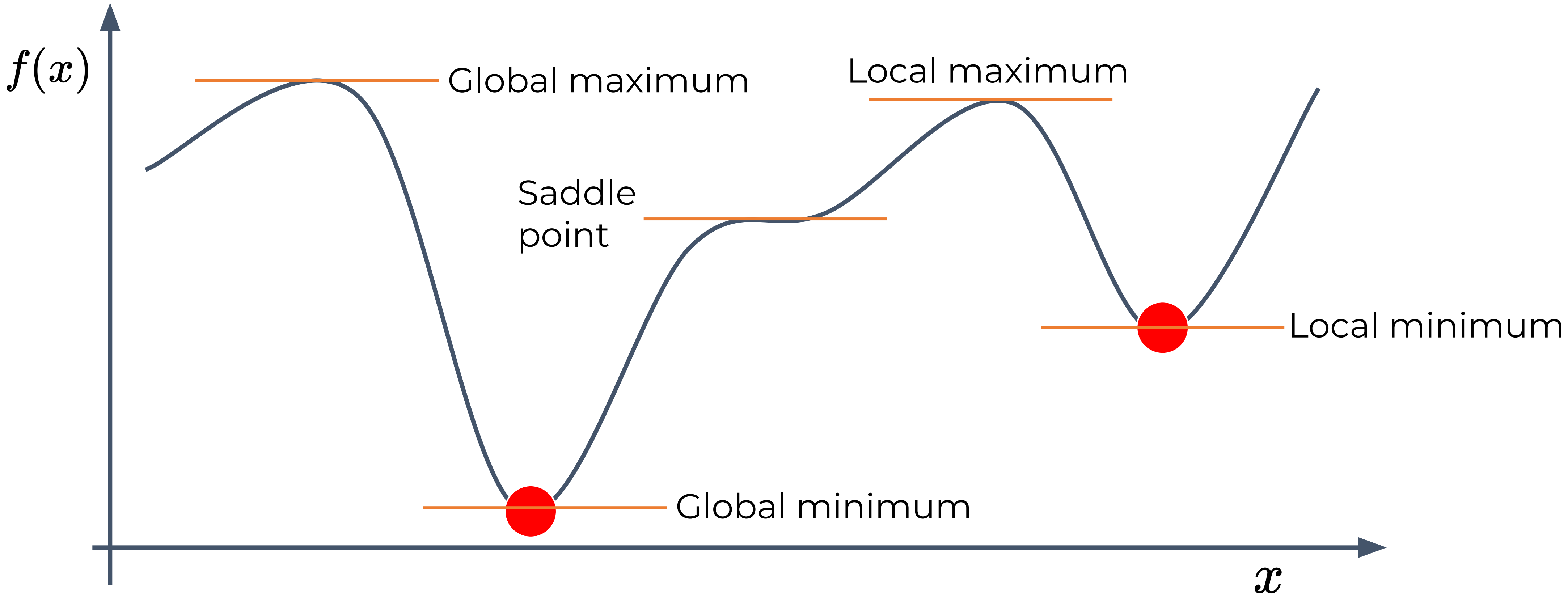
How to find extrema?

However, there are other points with a slope that equals 0.



Critical point classification

x^* is a critical point iif $f'(x^*) = 0$

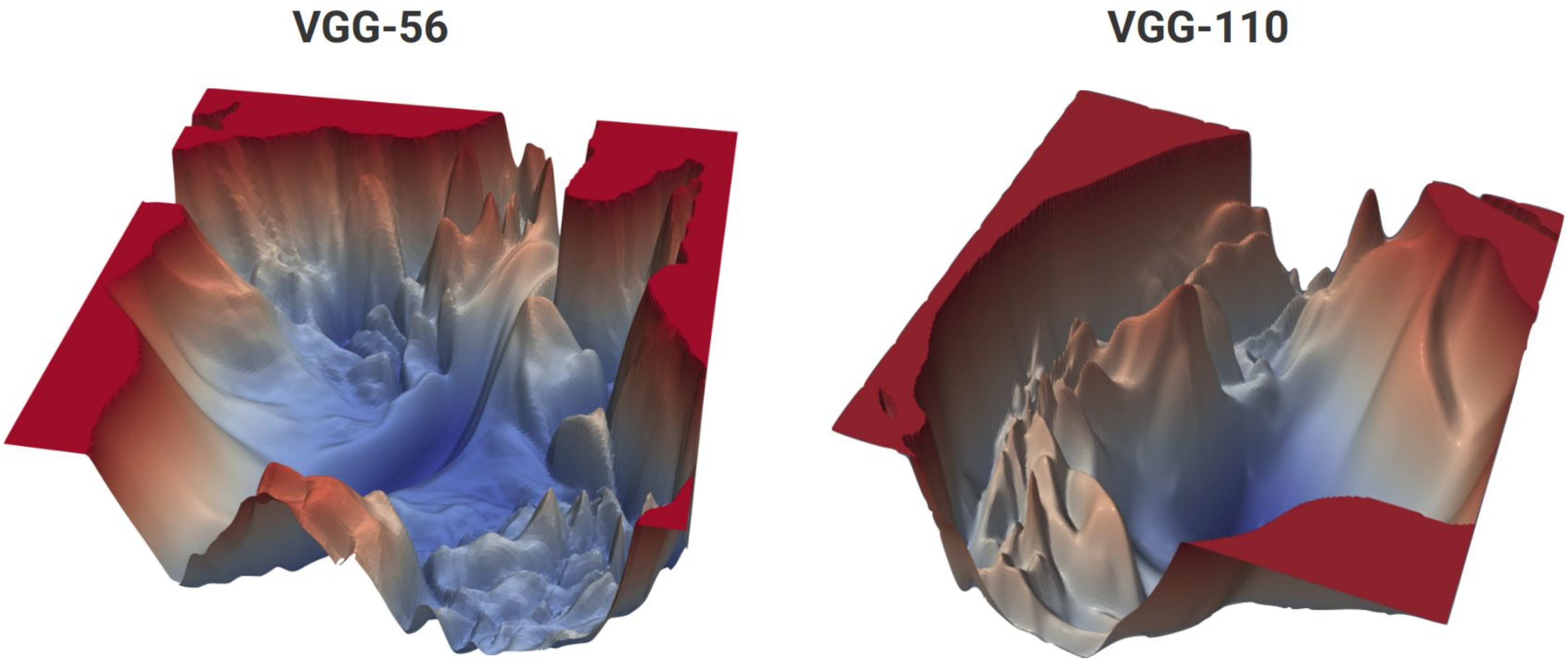


Fermat's theorem

If x^* is an extremum (local or global), then it is a critical point.

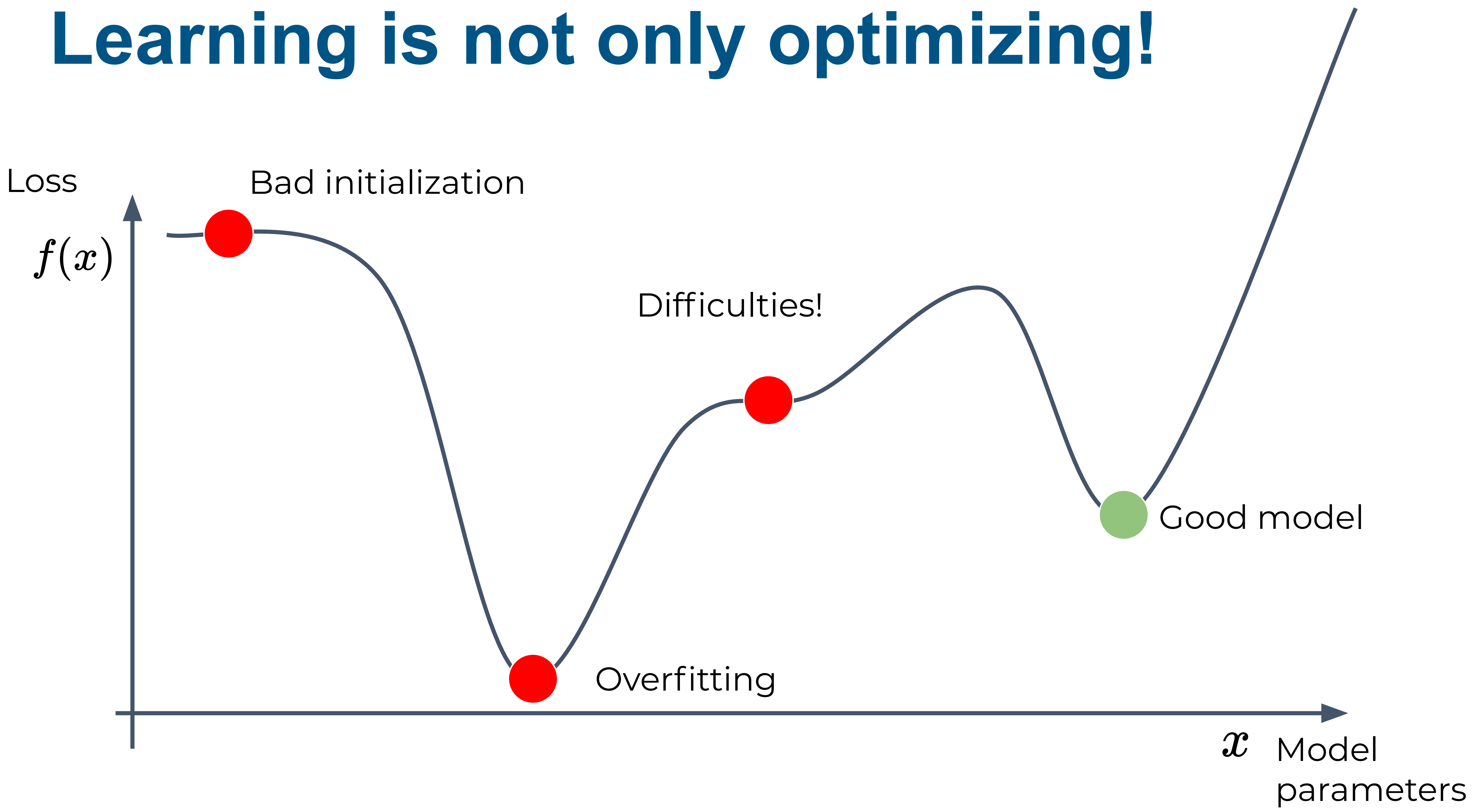
However, a critical point does not imply an extremum (e.g., a saddle point is not an extremum).

Parameter landscape for DL models



Li, Hao, et al. "Visualizing the loss landscape of neural nets." arXiv:1712.09913 (2017).

Learning is not only optimizing!



Critical point classification

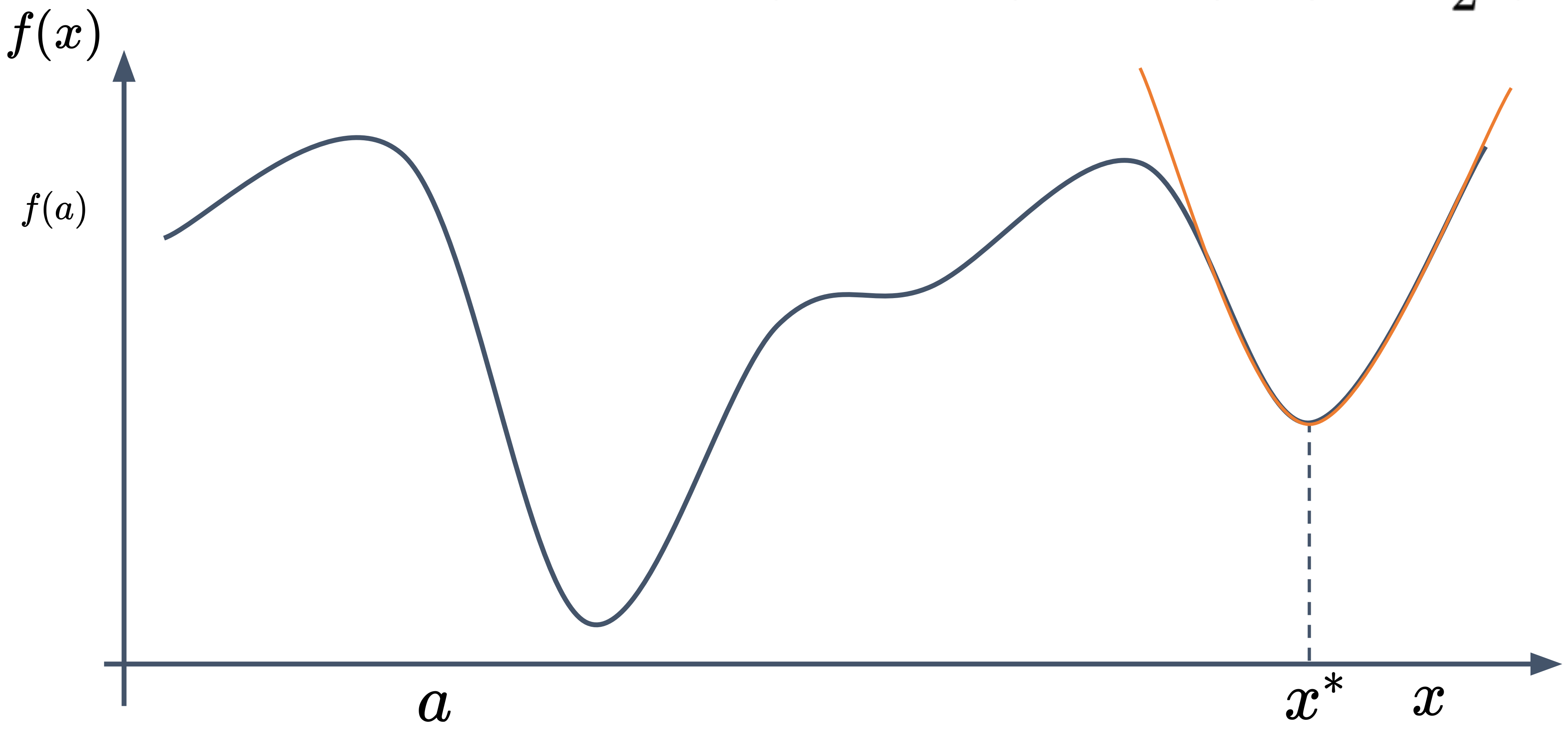
$$f(a+h) \approx f(a) + \langle f'(a), h \rangle + \frac{1}{2}\langle f''(a)h, h \rangle$$

Suppose that x^* is a minimum, then

$$f(x^* + h) \approx f(x^*) + \underbrace{\langle f'(x^*), h \rangle}_{0} + \underbrace{\frac{1}{2}\langle f''(x^*)h, h \rangle}_{q(h)}$$

Example

$$f(x^* + h) \approx f(x^*) + \frac{1}{2}\langle f''(x^*)h, h\rangle$$



Quadratic function in 2D

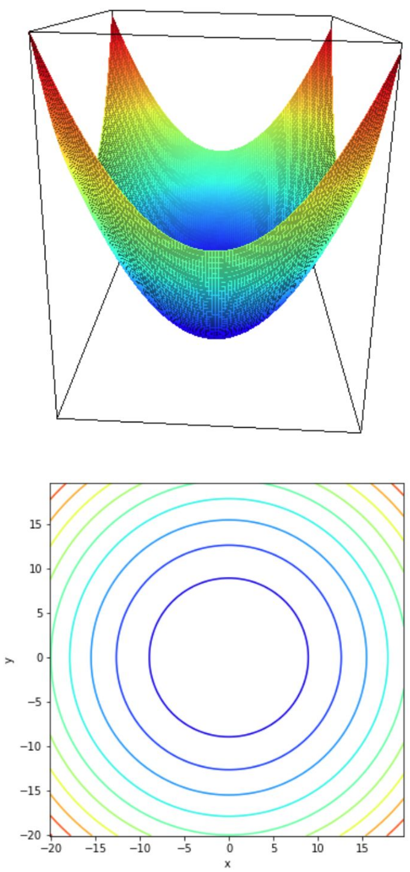
$$q(h) = \frac{1}{2}\langle f''(a)h, h\rangle$$

$$= \frac{1}{2}h^\top Hh = \frac{1}{2}\begin{bmatrix} h_1 & h_2 \end{bmatrix}\begin{bmatrix} \partial_{11}f(a) & \partial_{12}f(a) \\ \partial_{21}f(a) & \partial_{22}f(a) \end{bmatrix}\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

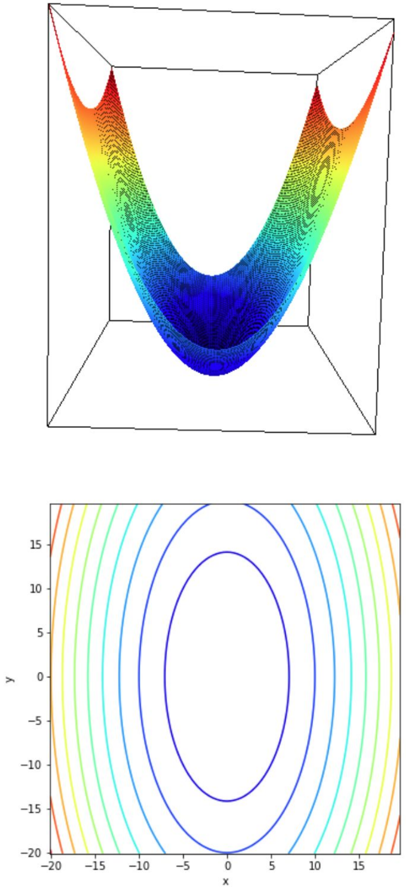
$$q'(h) = f''(a)h$$

$$\boxed{q''(h) = f''(a)}$$

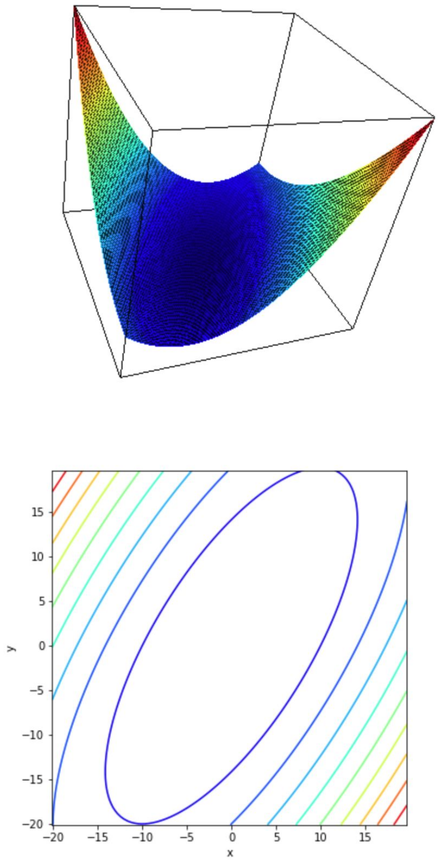
Positive definite in 2D



$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

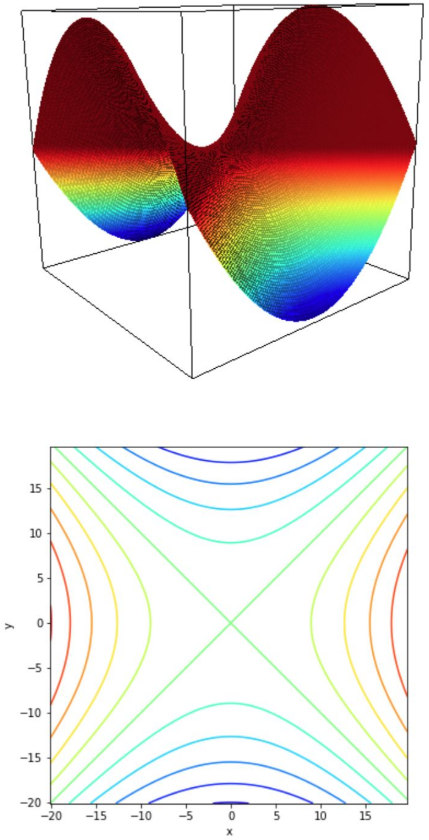


$$H = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

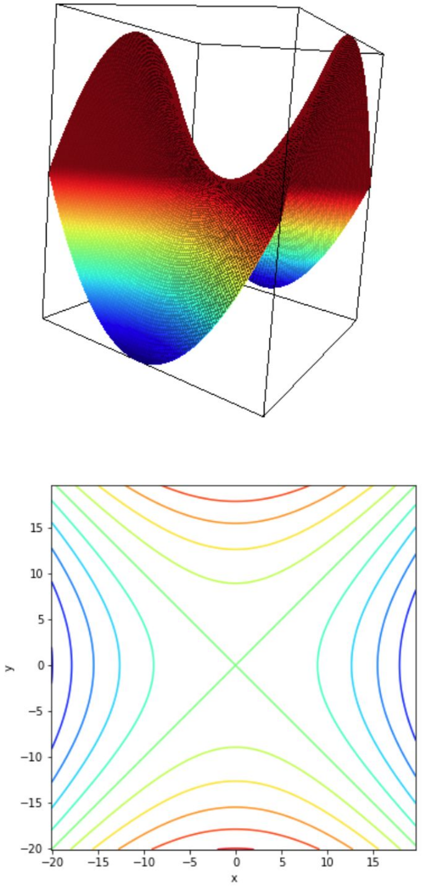


$$H = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

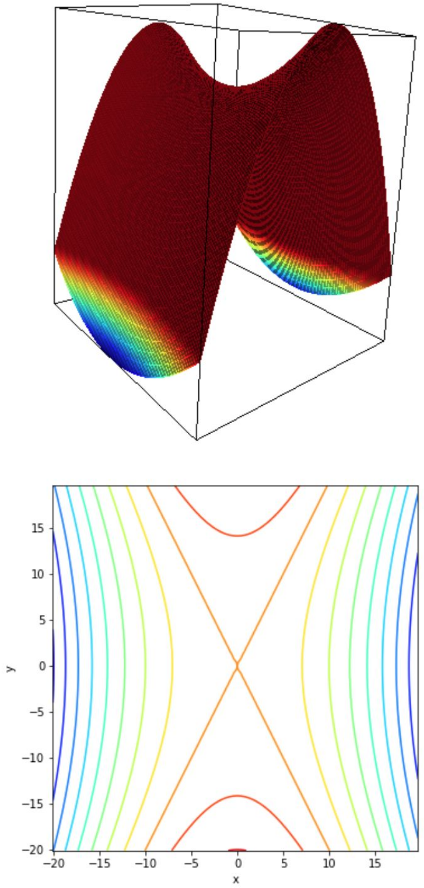
Saddle point in 2D



$$H = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$



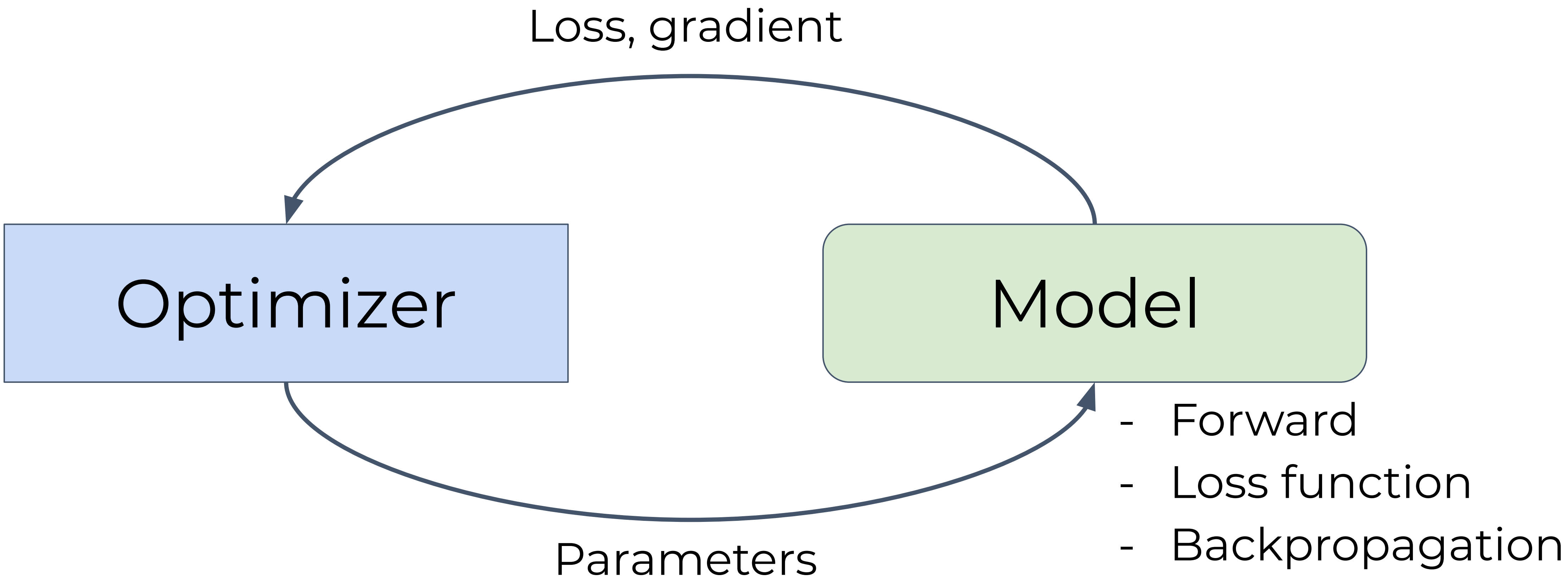
$$H = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$H = \begin{bmatrix} -4 & 0 \\ 0 & 1 \end{bmatrix}$$

Optimization algorithm

Iterative procedure



Why gradient is important for optimization?

$$f(a+h) - f(a) \approx \langle f'(a), h \rangle$$

h is a vector that gives the direction to go.

$$\begin{aligned} \min_{h, \|h\|=1} \langle f'(a), h \rangle &= \|f'(a)\| \|h\| \cos(\angle(f'(a), h)) \\ &= C \cos(\angle(f'(a), h)) \end{aligned}$$

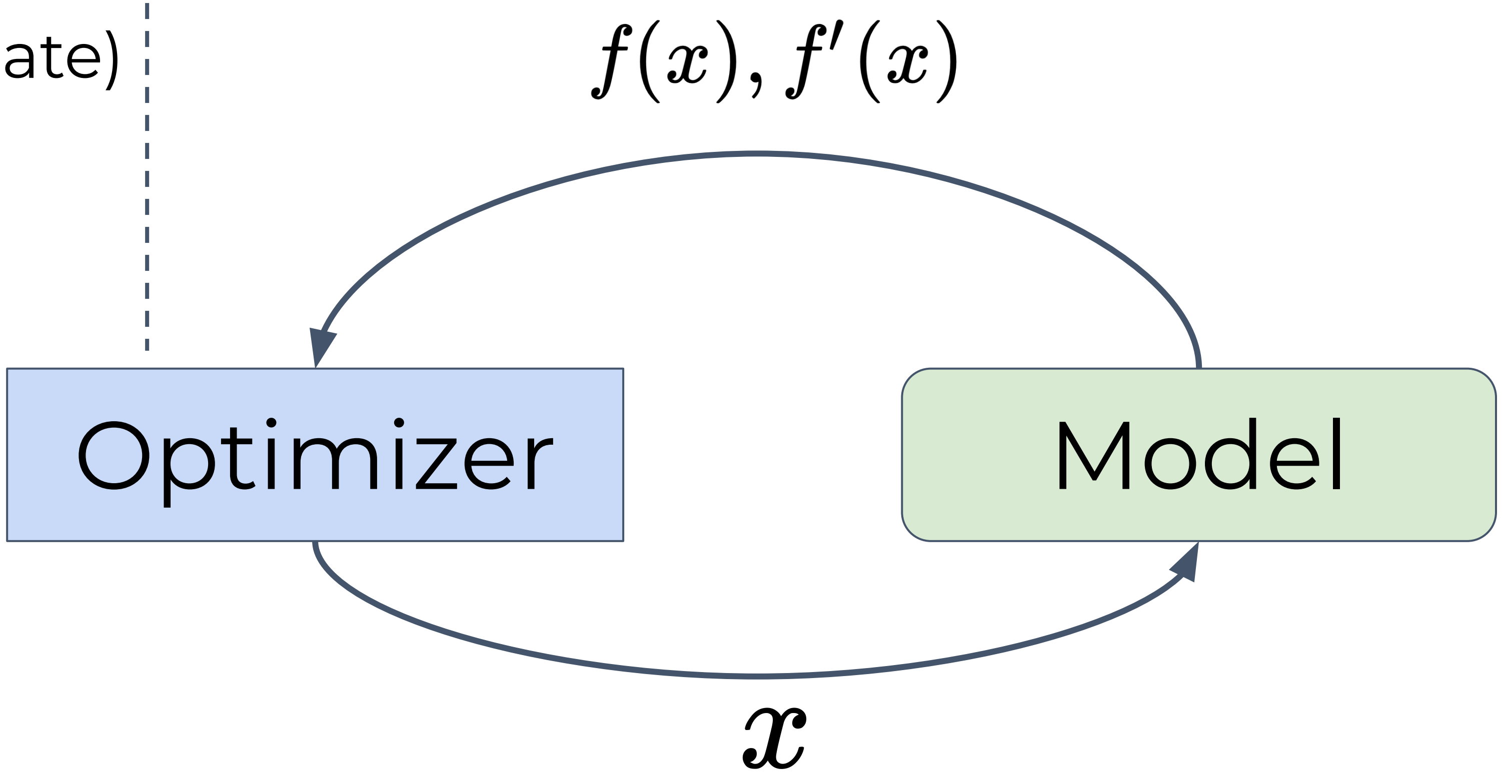
The gradient indicates
the direction of the **steepest slope** at point a

Mila

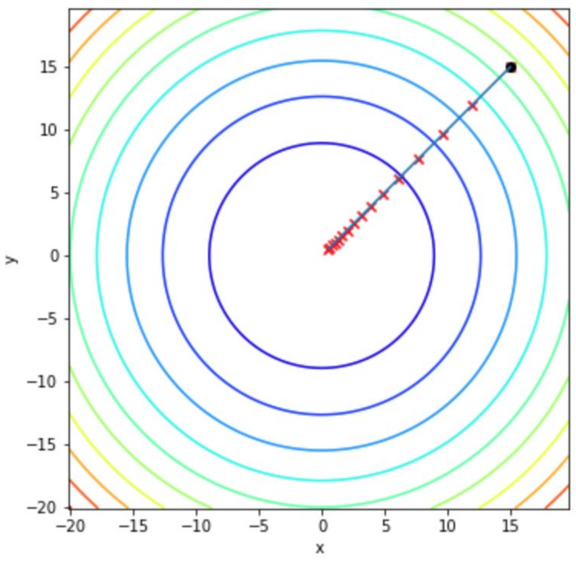
Steepest descent method (Gradient descent)

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

$\eta > 0$ (step-size, learning rate)

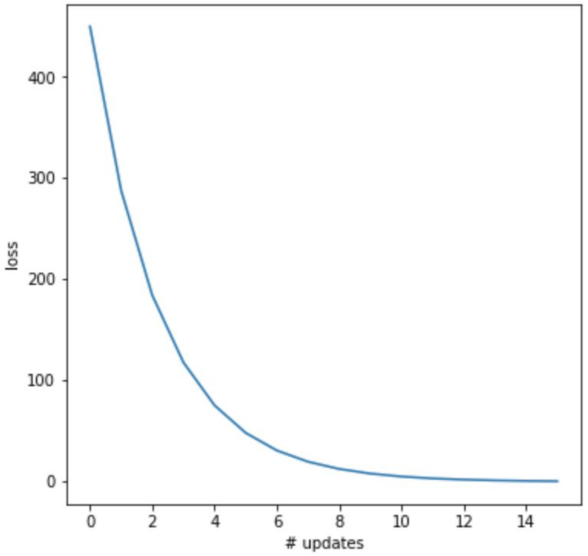


Gradient descent

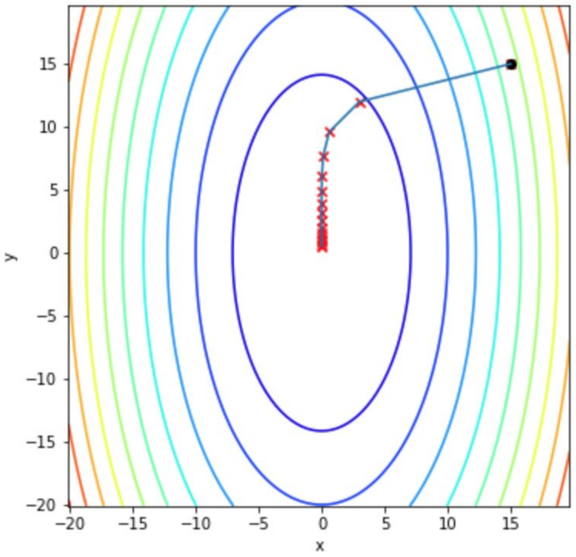


$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\eta = 0.1$$

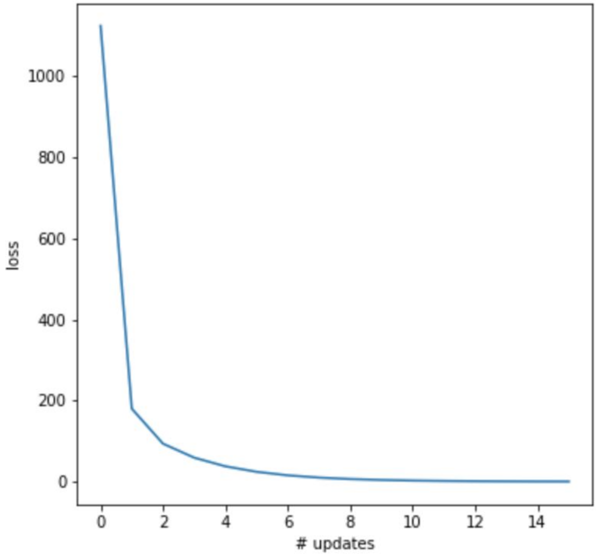


Gradient descent

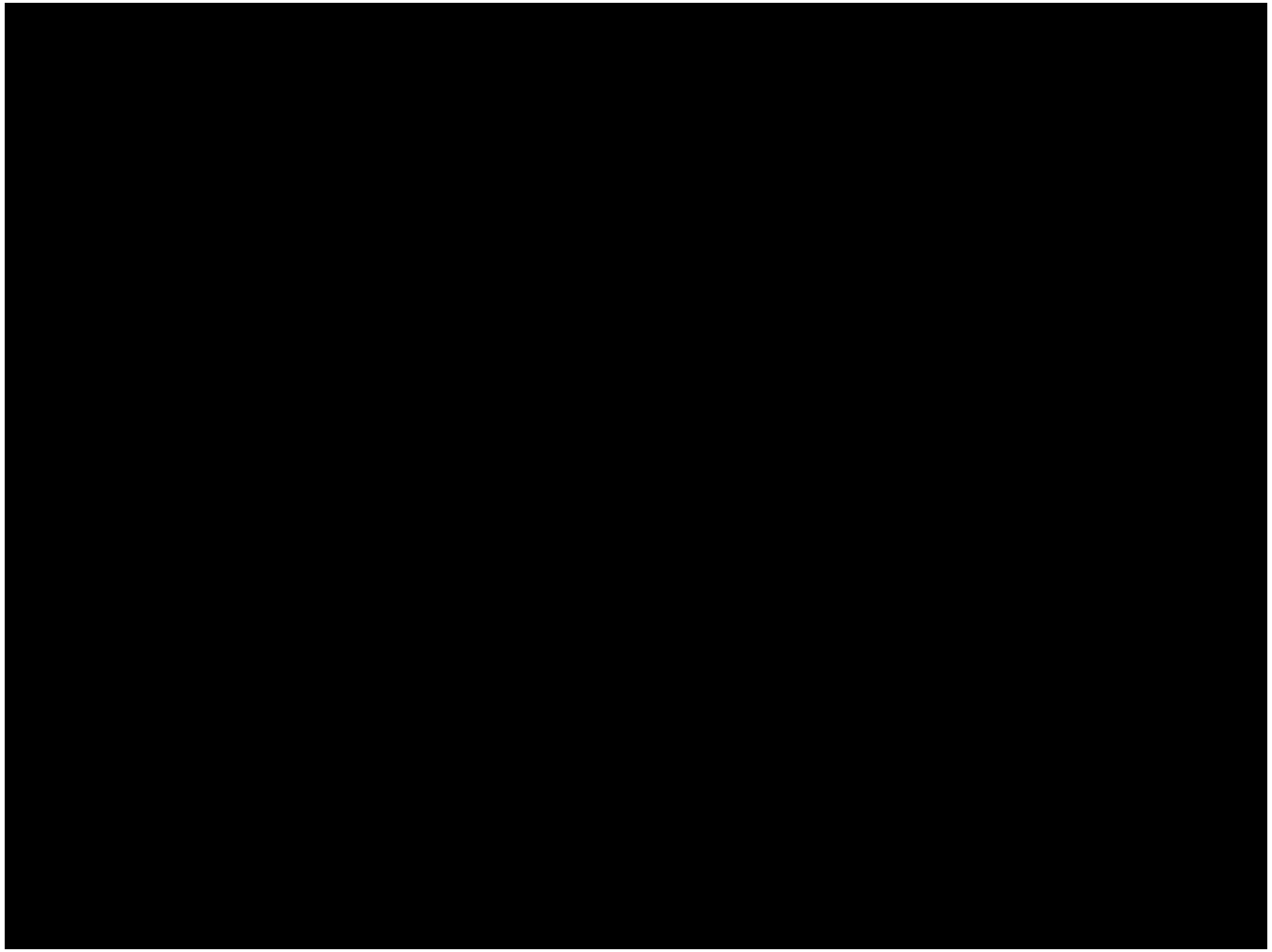


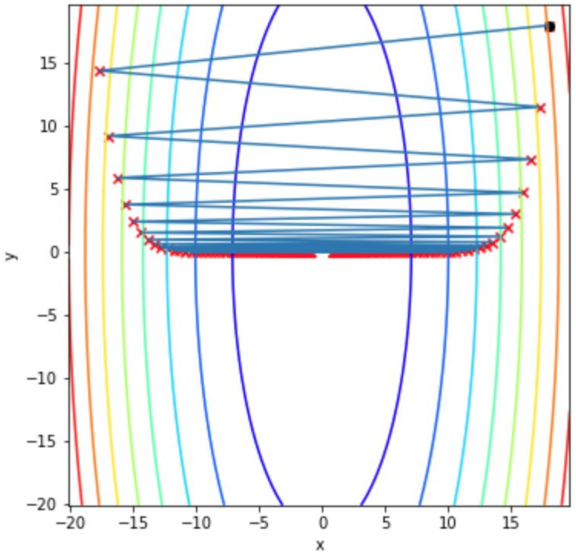
$$H = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\eta = 0.1$$



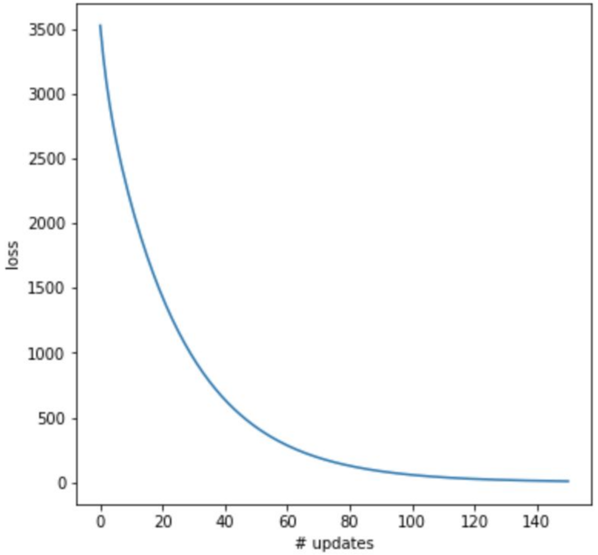
Gradient descent





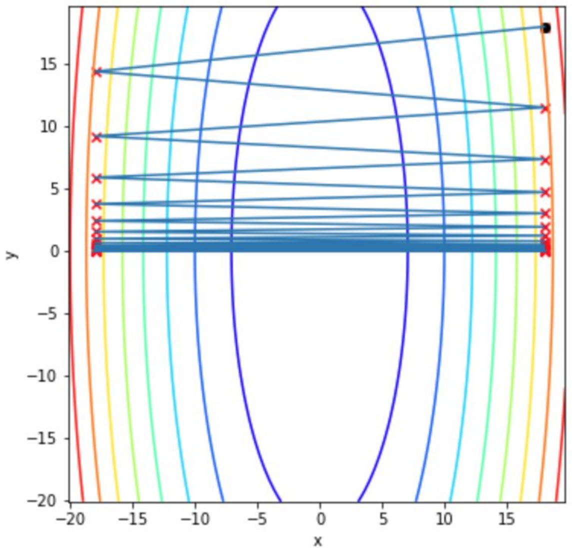
$$H = \begin{bmatrix} 9.9 & 0 \\ 0 & 1 \end{bmatrix}$$

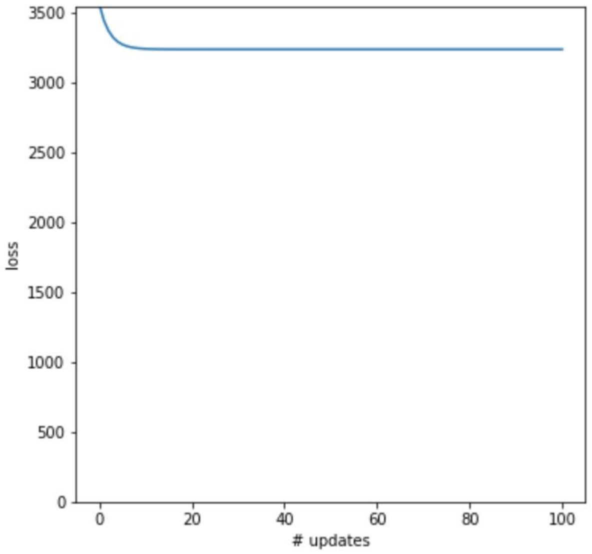
$$\eta = 0.1$$



Notice the rescaling of the axis.

Gradient descent





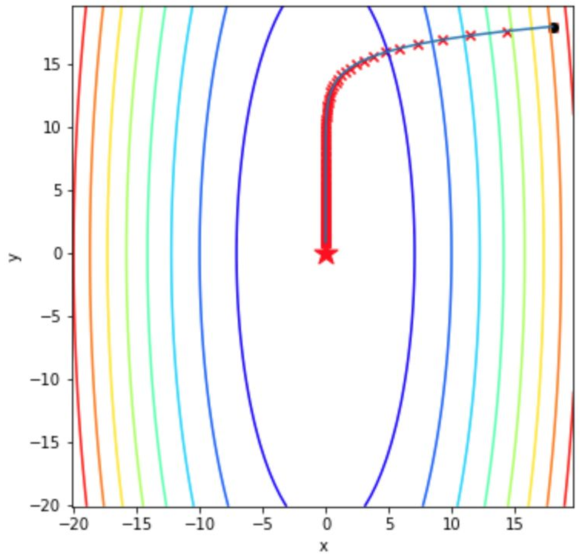
$$H = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\eta = 0.1$$

Notice the rescaling of the axis.

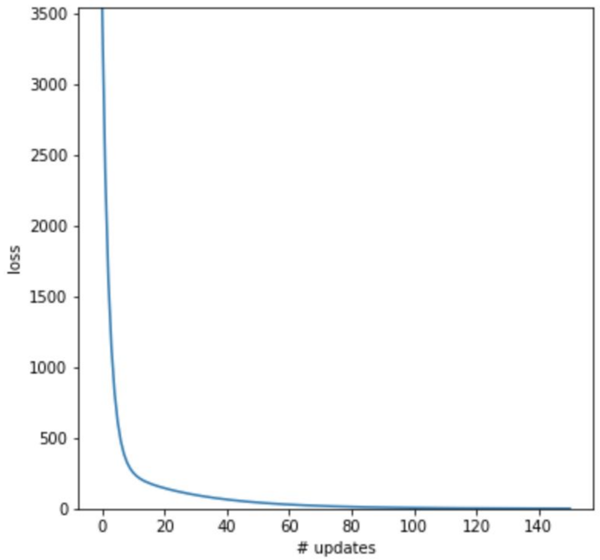
Mila

Gradient descent



$$H = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boxed{\eta = 0.01}$$



Notice the rescaling of the axis.

Mila

Conditioning problem

How to quantify the problem? Condition number.

$$H = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \qquad \kappa(H) = \frac{|10|}{|1|}$$

We want to have a condition number close to 1.

Problem of physical units

What are the physical units of this equation?

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

2D example:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \eta \begin{bmatrix} \partial_1 f(x) \\ \partial_2 f(x) \end{bmatrix}$$

x_1 : [km] x_2 : [mm] η : [?]

$\partial_1 f(x)$: [J/km] $\partial_2 f(x)$: [J/mm]

Problem of physical units

What are the physical units of this equation?

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

2D example:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \eta \begin{bmatrix} M_{11} & 0 \\ 0 & M_{22} \end{bmatrix} \begin{bmatrix} \partial_1 f(x) \\ \partial_2 f(x) \end{bmatrix}$$

x_1 : [km] x_2 : [mm] M_{11} : [km²/J]

$\partial_1 f(x)$: [J/km] $\partial_2 f(x)$: [J/mm] M_{22} : [mm²/J]

Mila

Preconditioning

General update equation for gradient descent.

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

M : preconditioning matrix

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \eta \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \partial_1 f(x) \\ \partial_2 f(x) \end{bmatrix}$$

Preconditioning

General update equation for gradient descent.

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

The size of M_t is (number of parameters)^2!

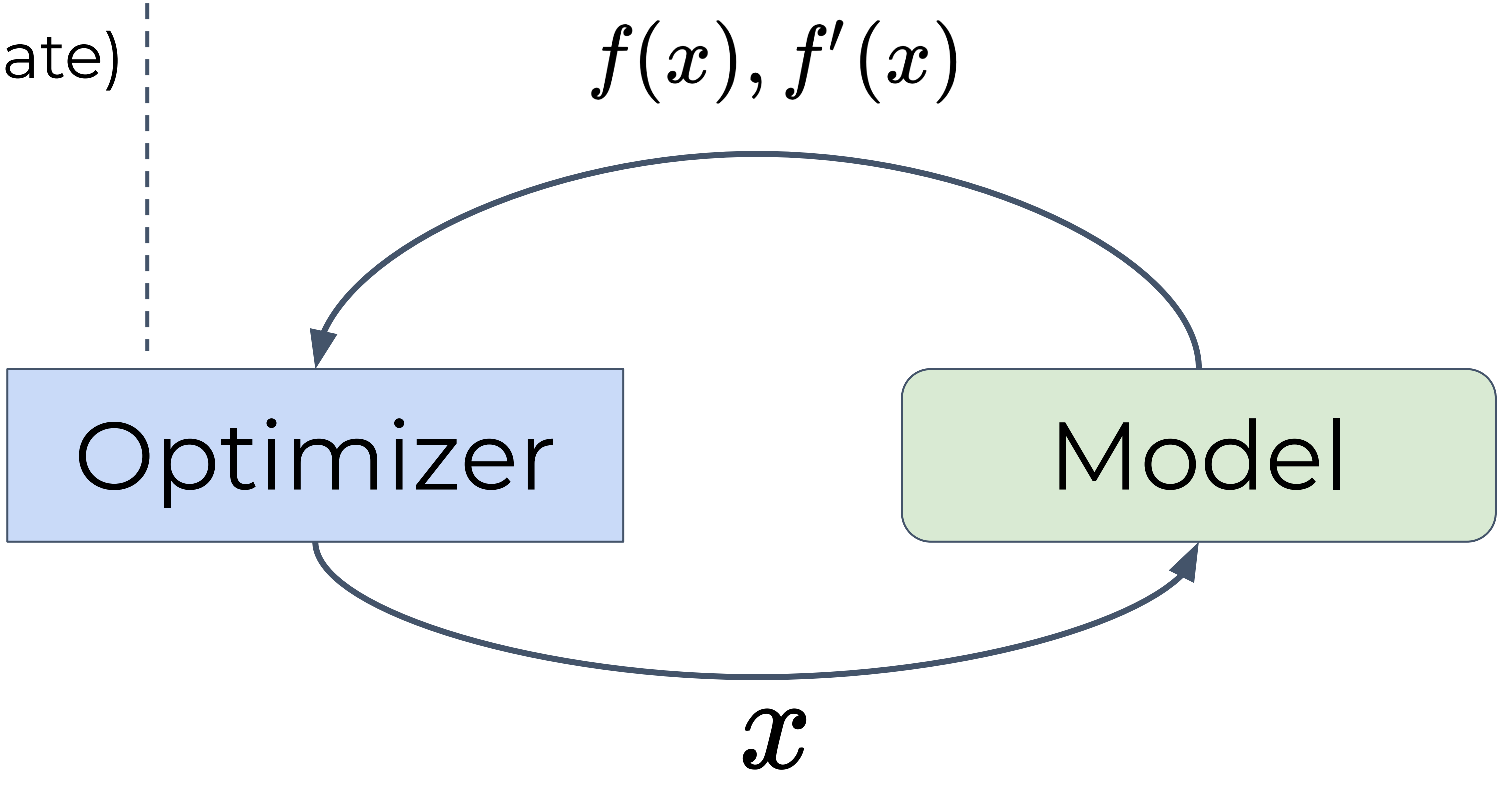
M : preconditio

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \eta \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \partial_1 f(x) \\ \partial_2 f(x) \end{bmatrix}$$

Preconditioned gradient descent

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

$\eta > 0$ (step-size, learning rate)



RMSProp

Component-wise preconditioning

$$v_t = \beta v_{t-1} + (1-\beta) f'(x_t)^2$$

Preconditioning term

$$x_{t+1} = x_t + \frac{1}{\sqrt{v_t + \epsilon}} f'(x_t)$$

Update rule

Momentum

- Potential energy $U(x) = mgh(x)$

$$F = ma = -U'(x) = -mgh'(x)$$

Source:Sam Poullain, Unsplash

Initial conditions: x_0, v_0

Physics	Optimization
$a_t \leftarrow -gh'(x_t)$	$a_t \leftarrow -\eta f'(x_t)$
$v_{t+1} \leftarrow \mu v_t + a_t \Delta t$	$v_{t+1} \leftarrow \mu v_t + a_t$
$x_{t+1} \leftarrow x_t + v_{t+1} \Delta t$	$x_{t+1} \leftarrow x_t + v_{t+1}$

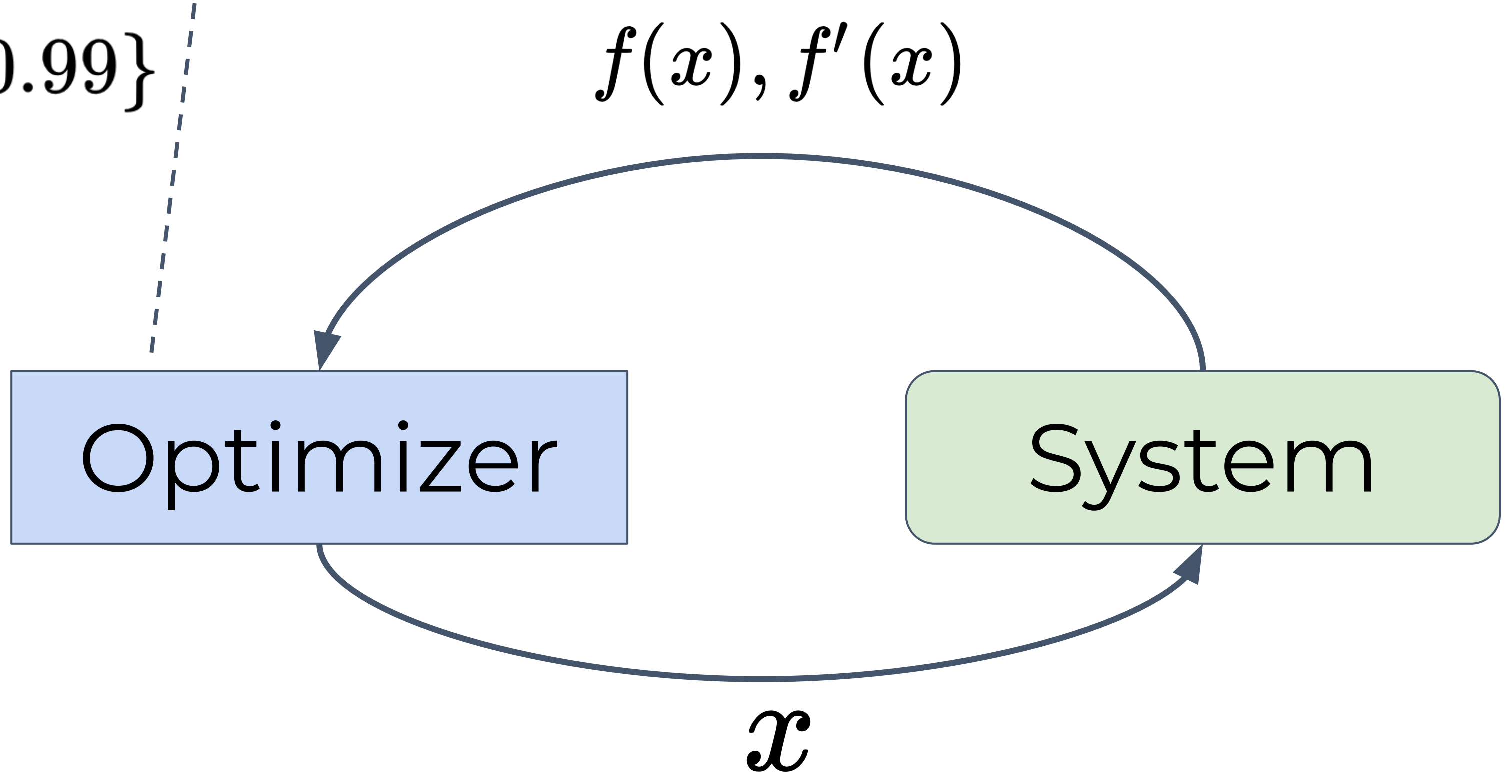
μ : friction

Gradient descent with momentum

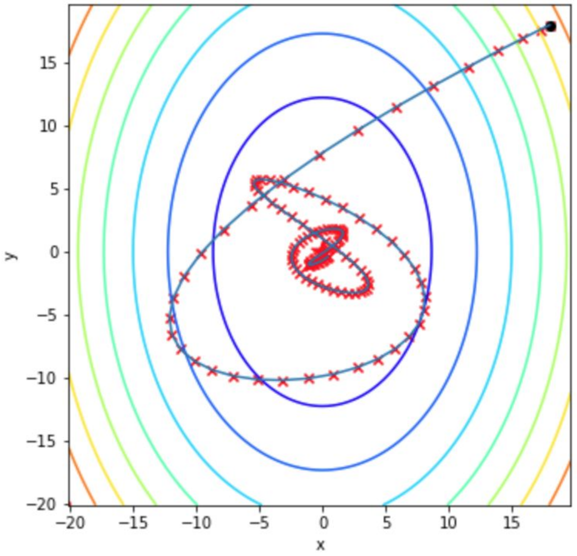
$$v \leftarrow \mu v - \eta f'(x)$$

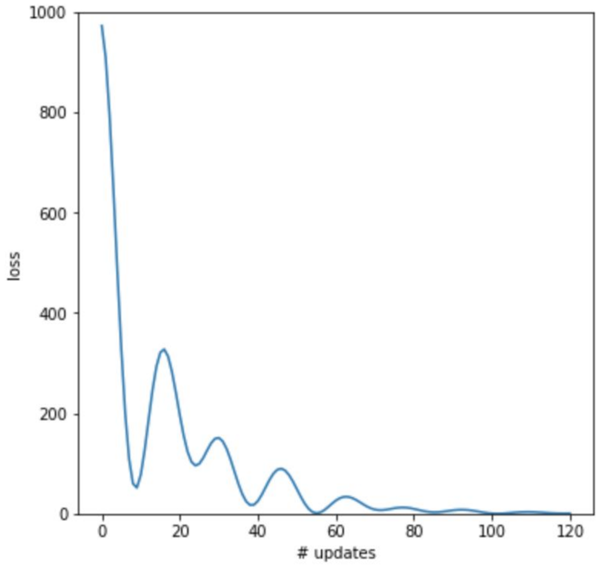
$$x \leftarrow x + v$$

$\eta > 0 \quad \mu \in \{0.5, 0.9, 0.95, 0.99\}$



Momentum



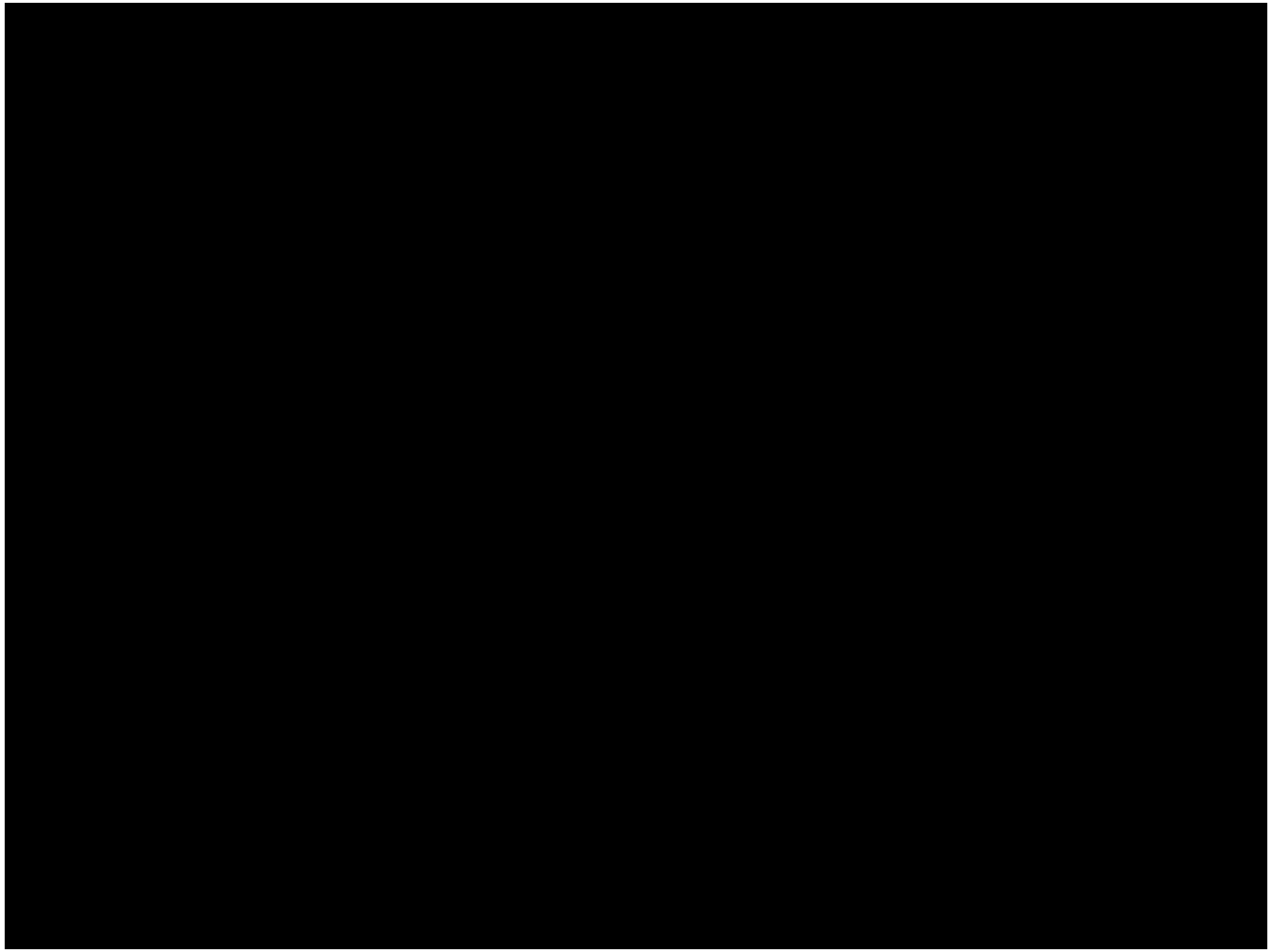


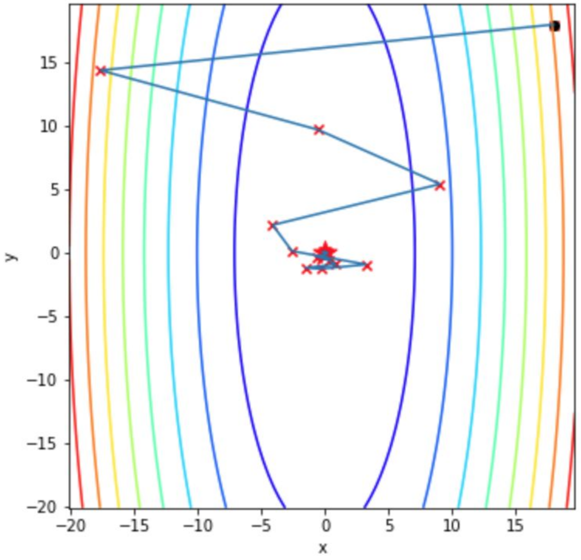
$$H = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

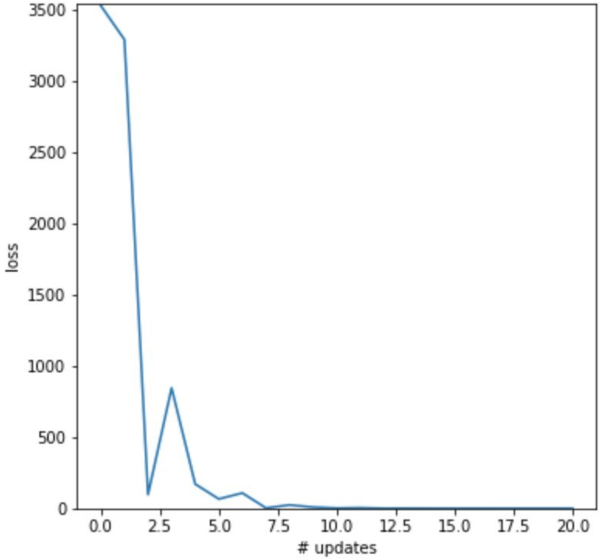
$$\eta = 0.01$$

$$\mu = 0.95$$

Momentum







$$H = \begin{bmatrix} 9.9 & 0 \\ 0 & 1 \end{bmatrix}$$

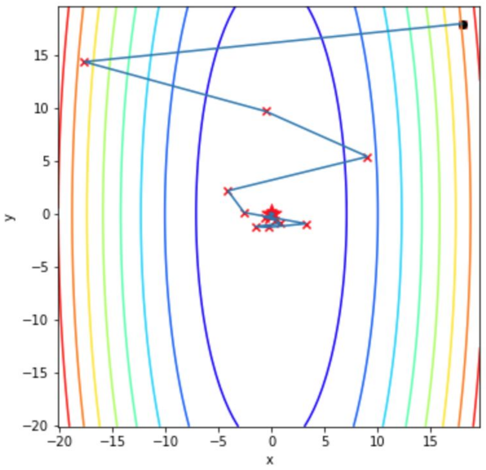
$$\eta = 0.1$$

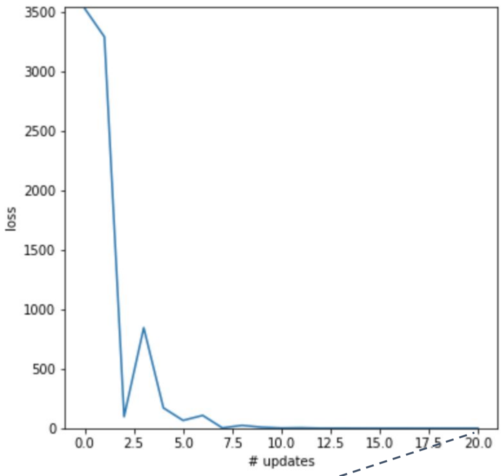
$$\mu = 0.5$$

Comparison

$$H = \begin{bmatrix} 9.9 & 0 \\ 0 & 1 \end{bmatrix}$$

With Momentum

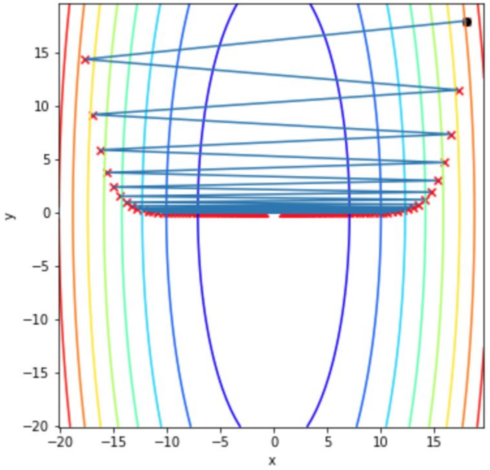


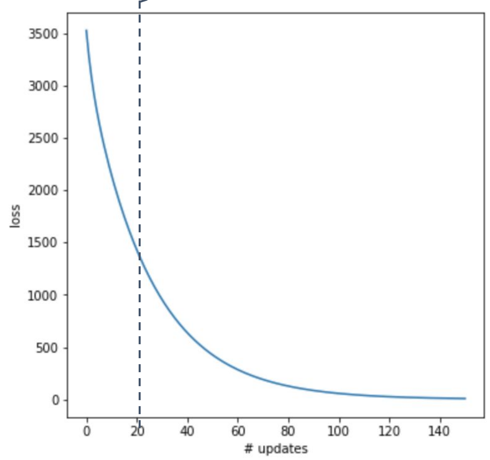


$$\eta = 0.1$$

$$\mu = 0.5$$

Without Momentum

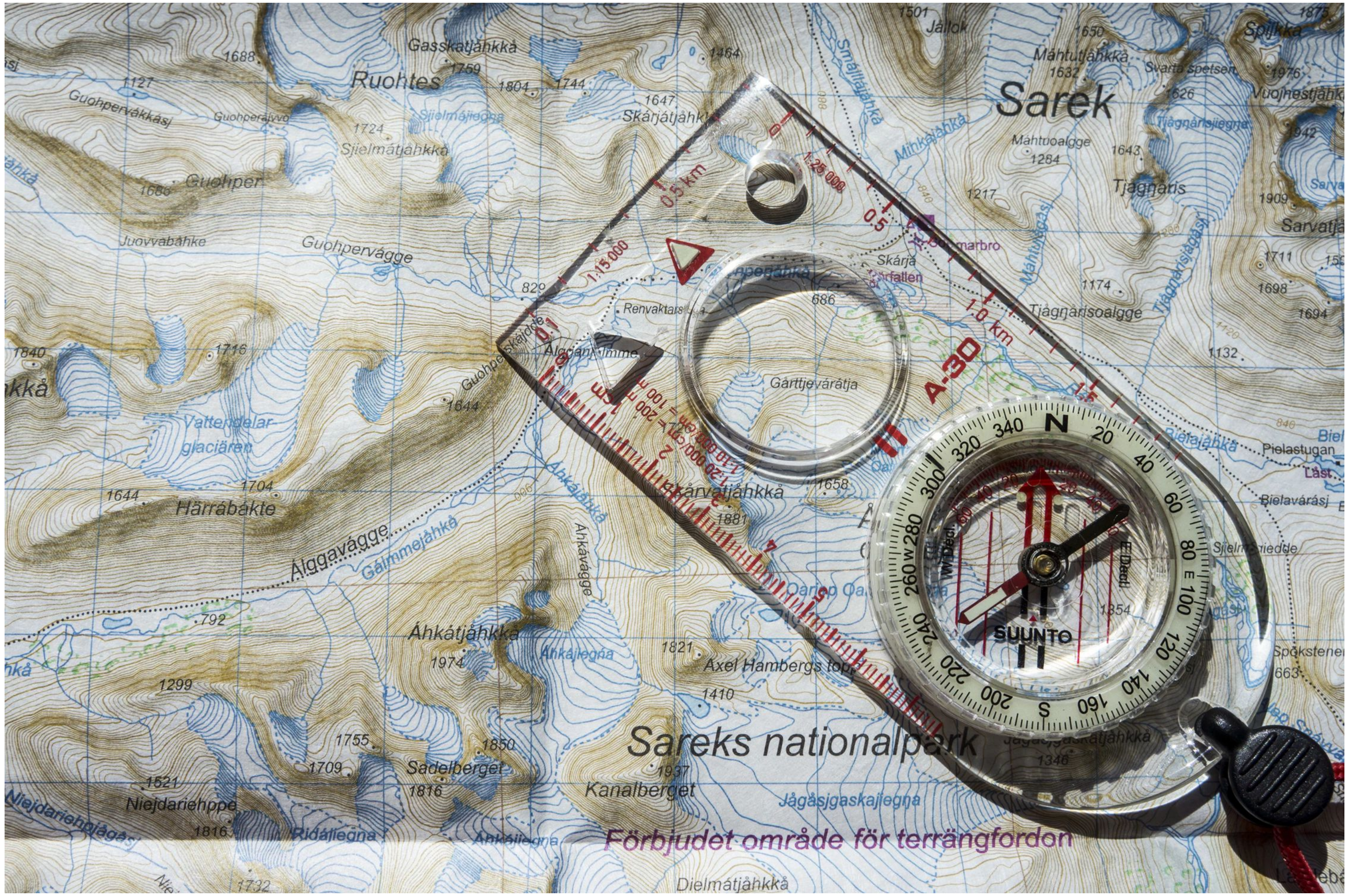




$$\eta = 0.1$$

A family of algorithms

- Adagrad
- Adadelta
- AdaMax
- RMSProp
- **Adam**
- AMSGrad
- ...



Source: Hendrik Morkel, Unsplash

Adaptive moment estimation (Adam)
RMSProp with momentum

Component-wise preconditioning

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) f'(x_t)$$ Momentum term

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) f'(x_t)^2$$ Preconditioning term

$$x_{t+1} = x_t + \frac{1}{\sqrt{v_t} + \epsilon} m_t$$ Update rule

This is a simplified version without the bias correction terms!

Quebec
Artificial
Intelligence
Institute

Mila

Stochastic optimization

Stochastic gradient descent

Risk

$$L(\theta) = \mathbb{E}_{z \sim \mathcal{D}} \left[l(\theta; z) \right]$$

Stochastic gradient descent

Risk

Empirical risk

$$L(\theta) = \mathbb{E}_{z\sim\mathcal{D}}\left[l(\theta; z)\right] \xrightarrow{\text{Approximation 1}} \hat{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(\theta; z_i)$$

Mila

Stochastic gradient descent

Risk

$$L(\theta) = \mathbb{E}_{z\sim\mathcal{D}}\left[l(\theta; z)\right]$$

Empirical risk

$$\hat{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(\theta; z_i)$$

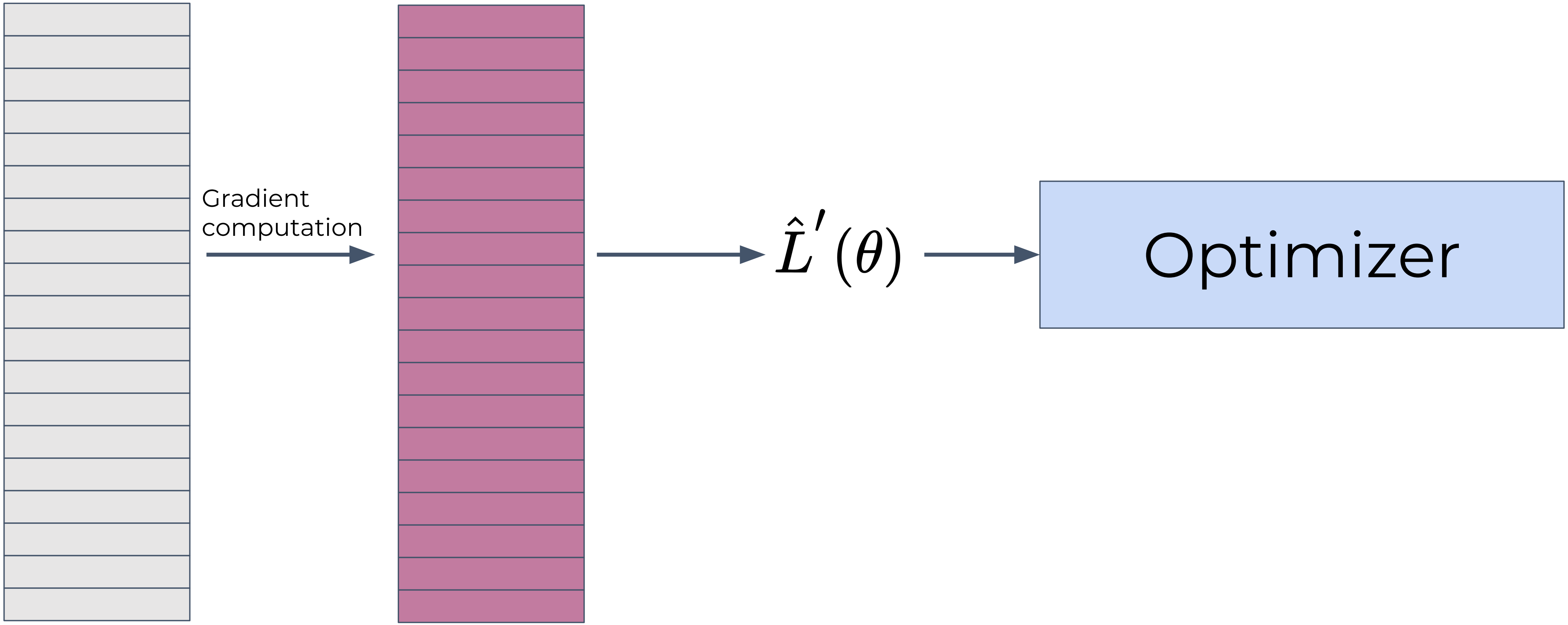
Approximation 1

Approximation 2

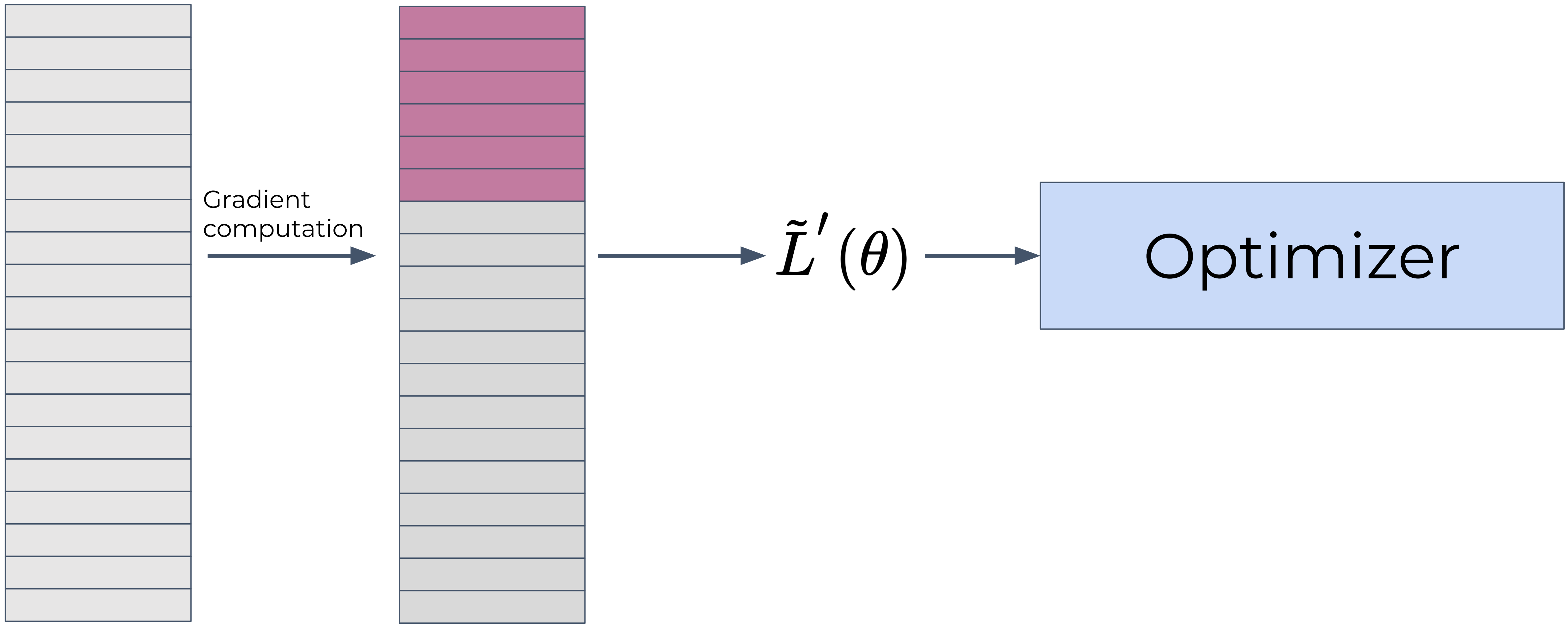
$$\tilde{L}(\theta) = \frac{1}{K}\sum_{i=1}^{K} l(\theta; z_{\sigma(i)})$$

K<<N : Number of examples (Mini-batch)

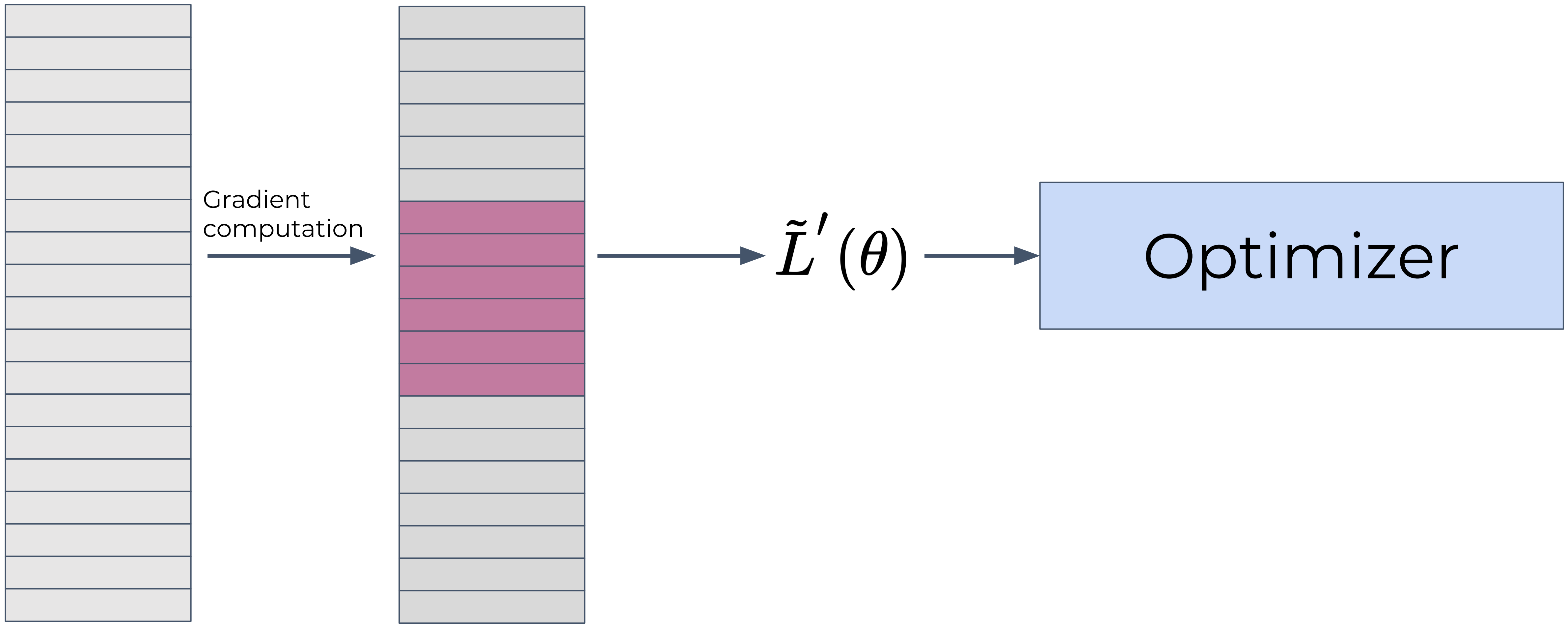
Gradient descent



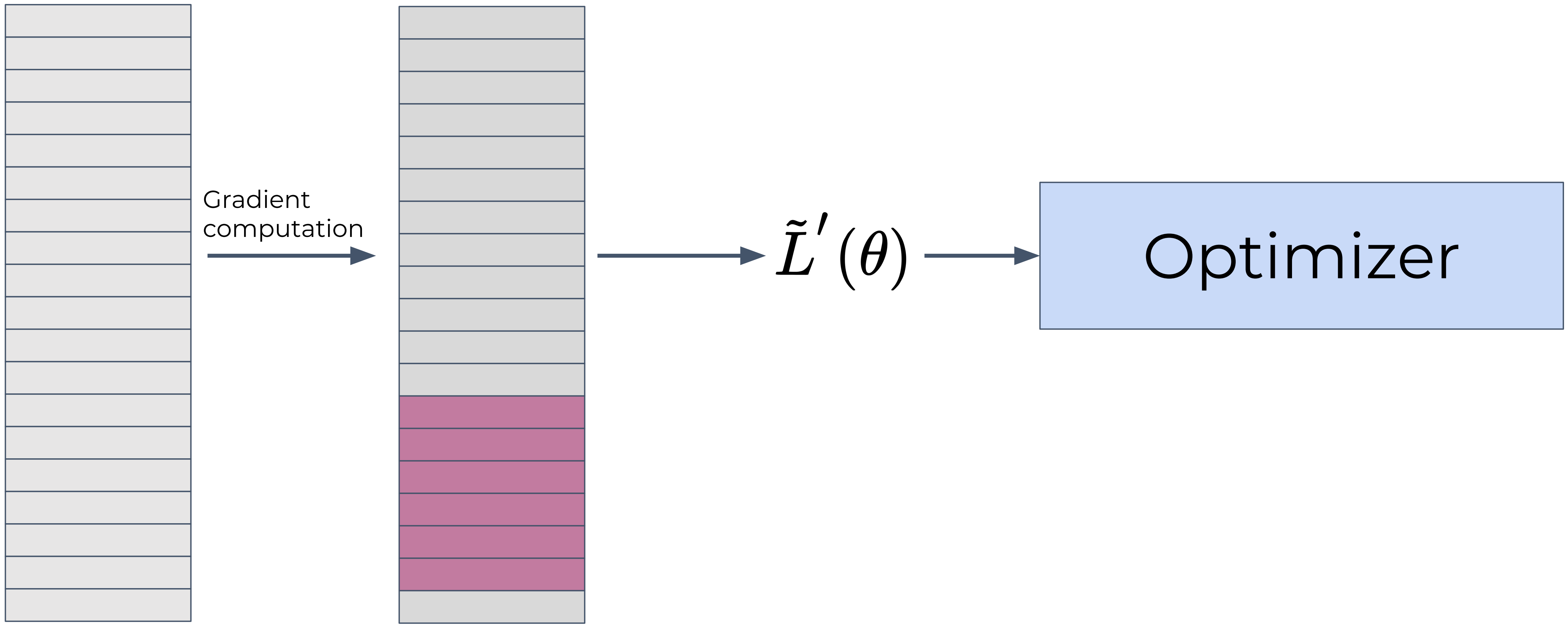
Stochastic gradient descent



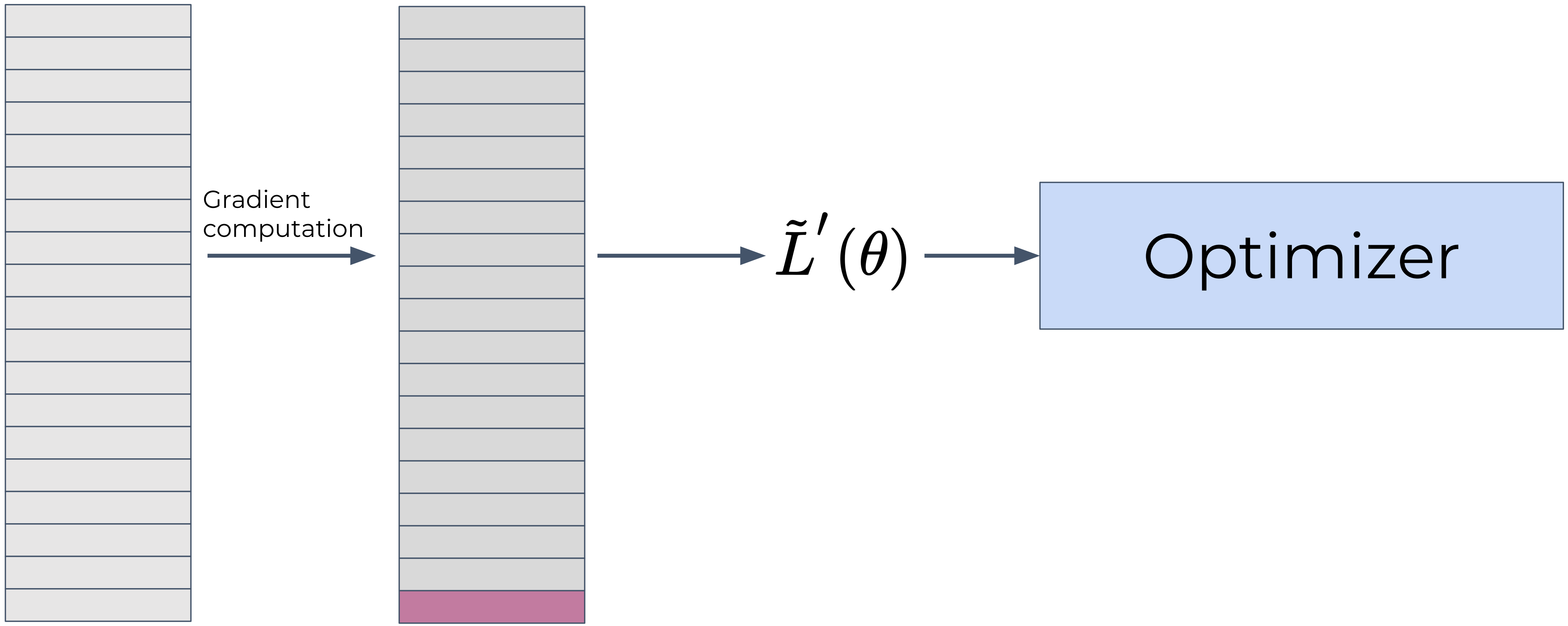
Stochastic gradient descent



Stochastic gradient descent

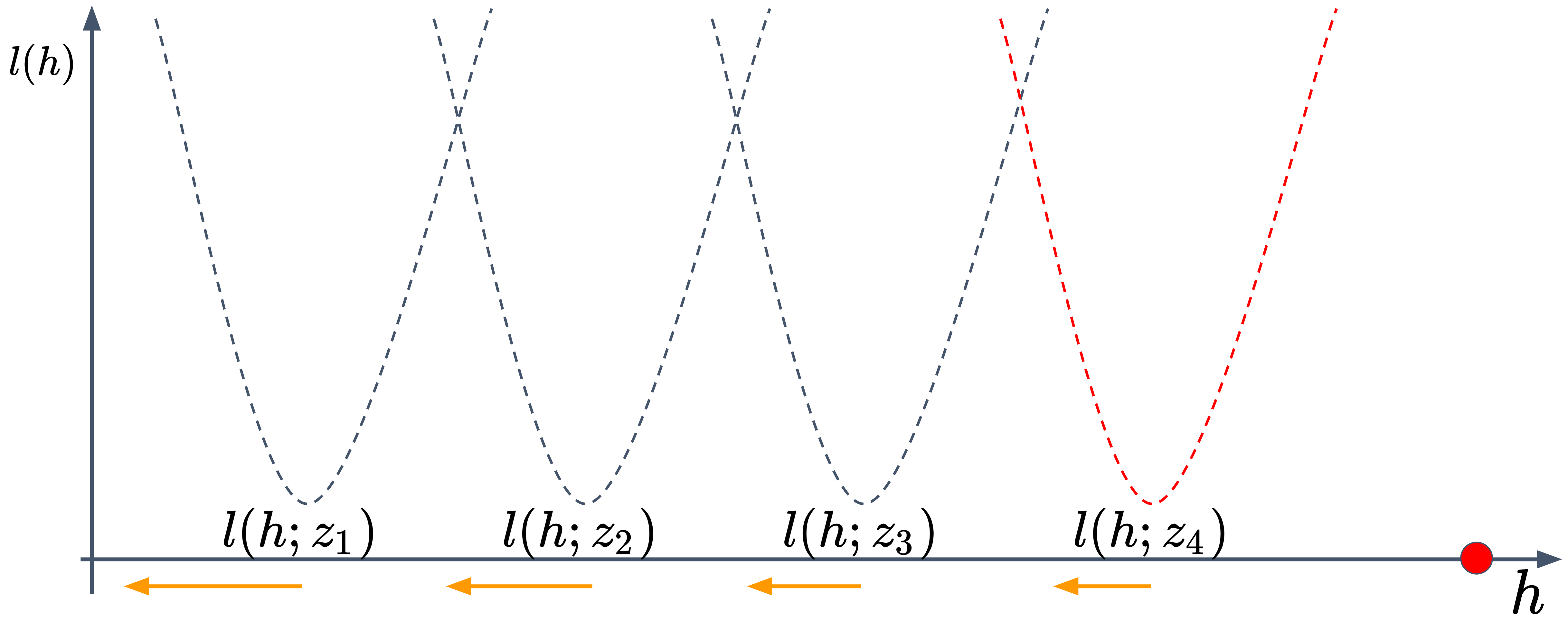


Stochastic gradient descent

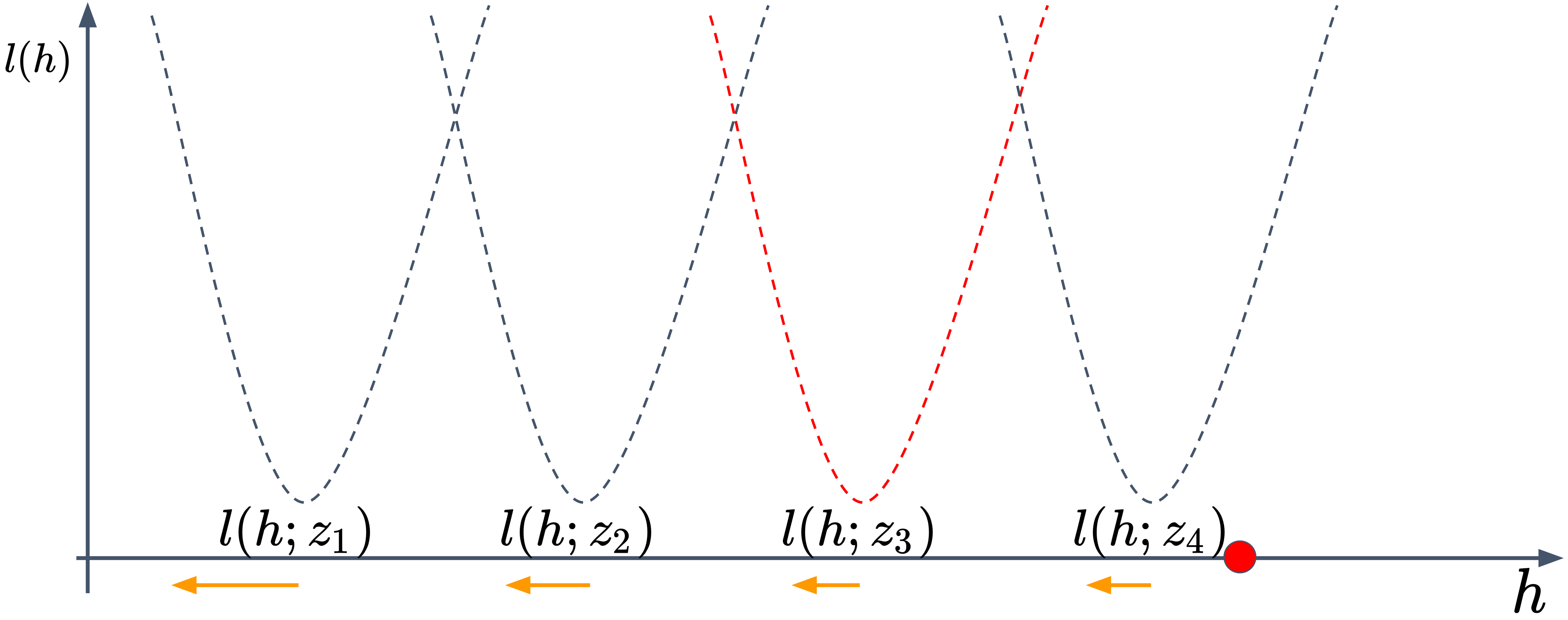


Mila

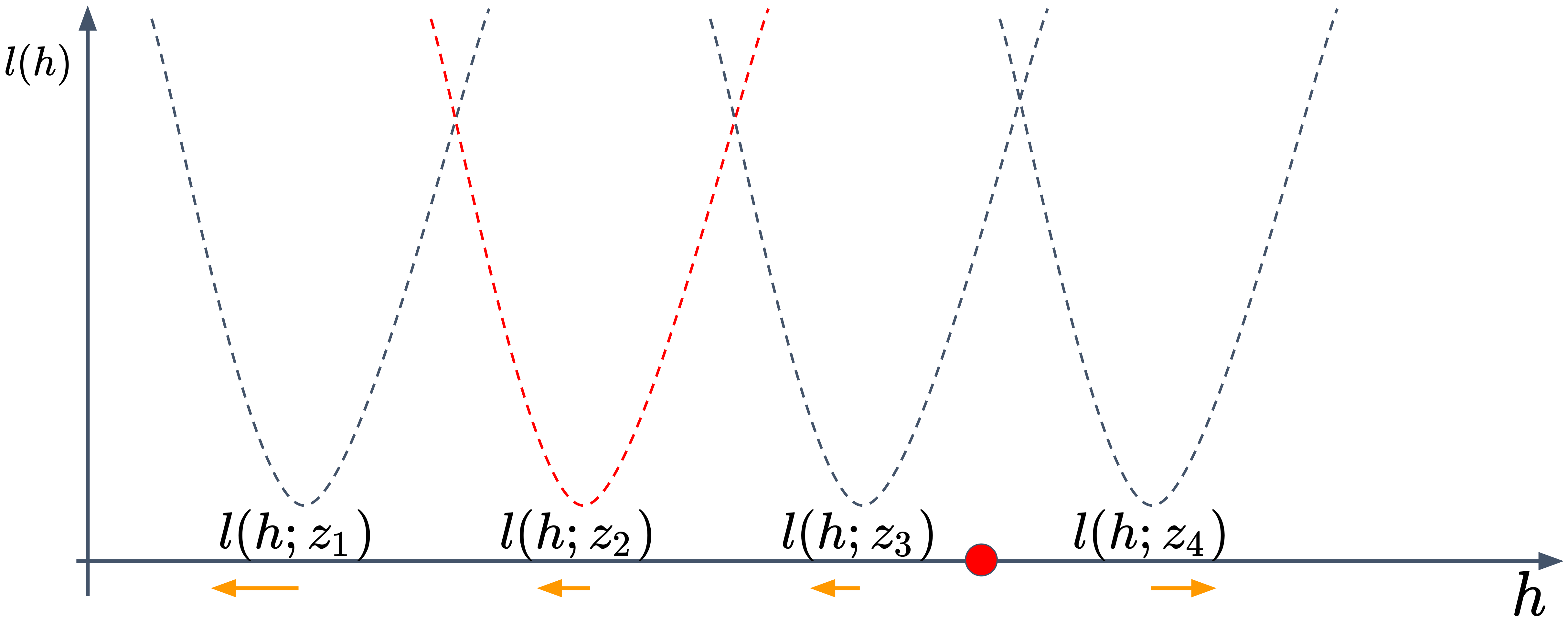
Convergence problem



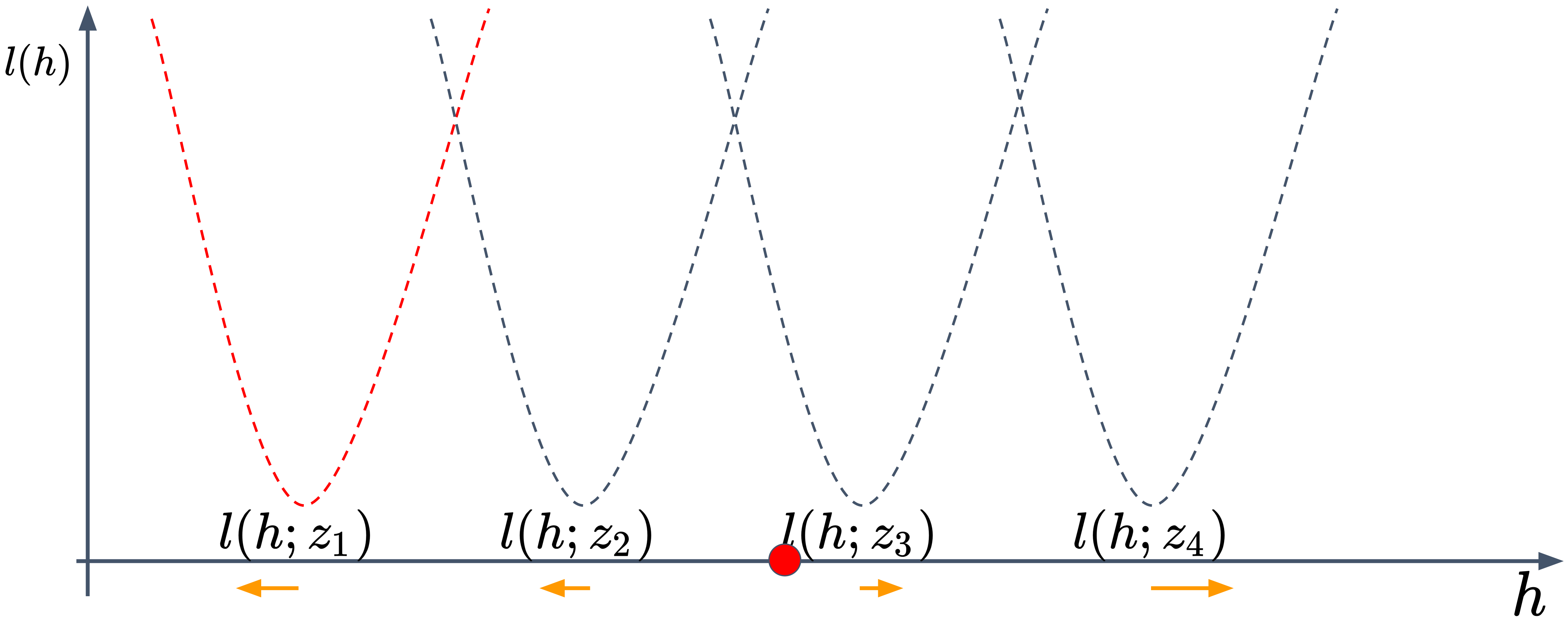
Convergence problem



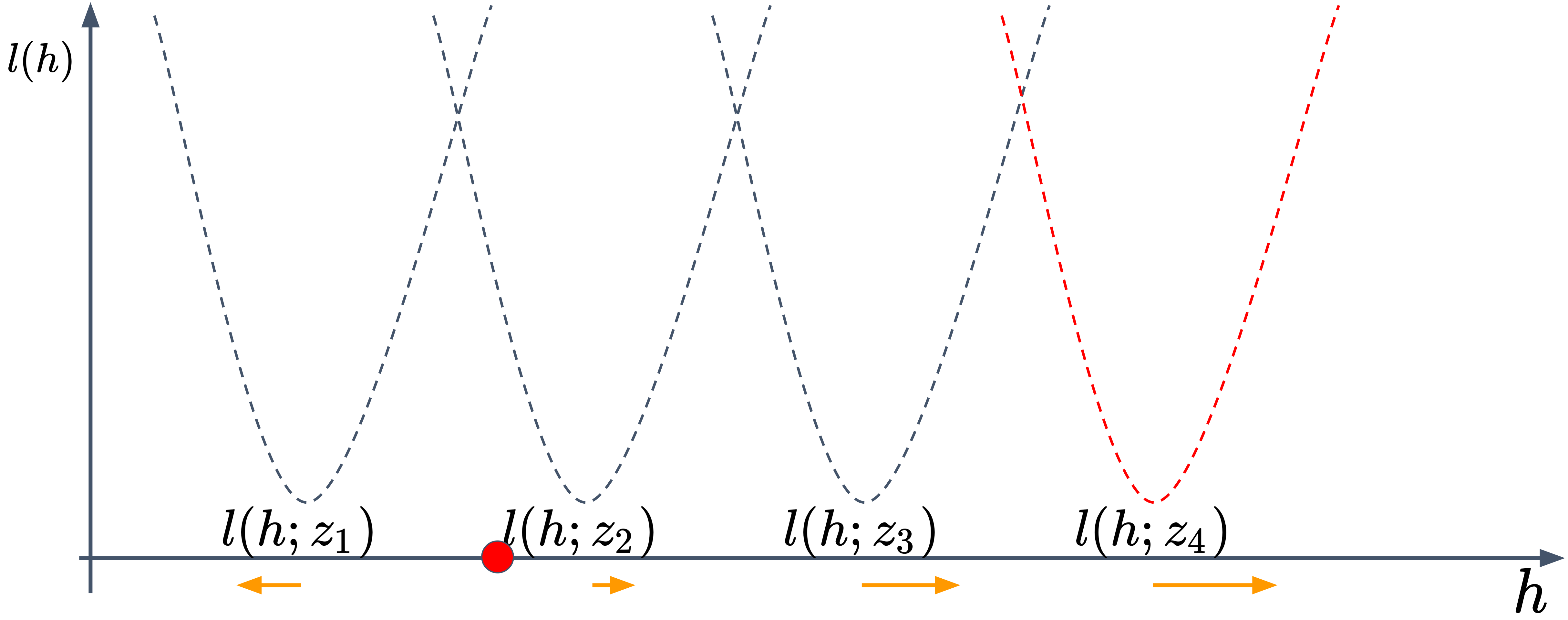
Convergence problem



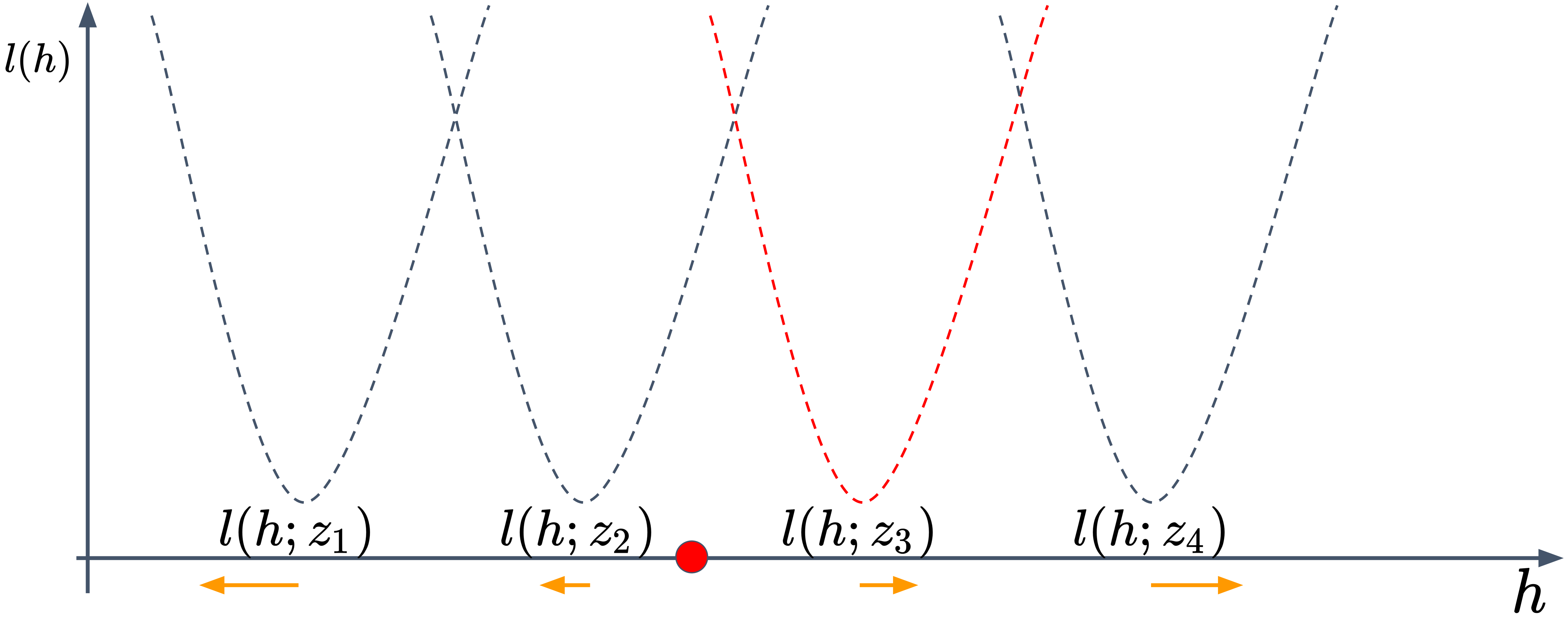
Convergence problem



Convergence problem

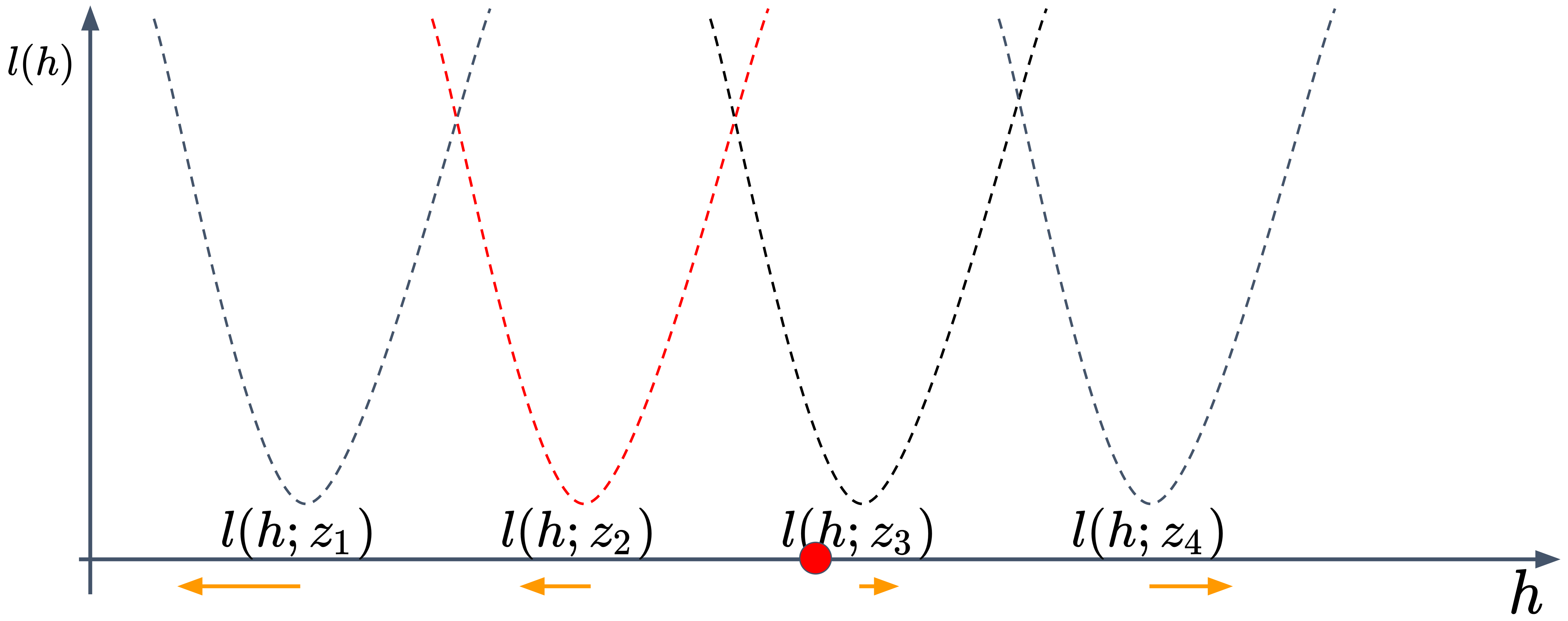


Convergence problem

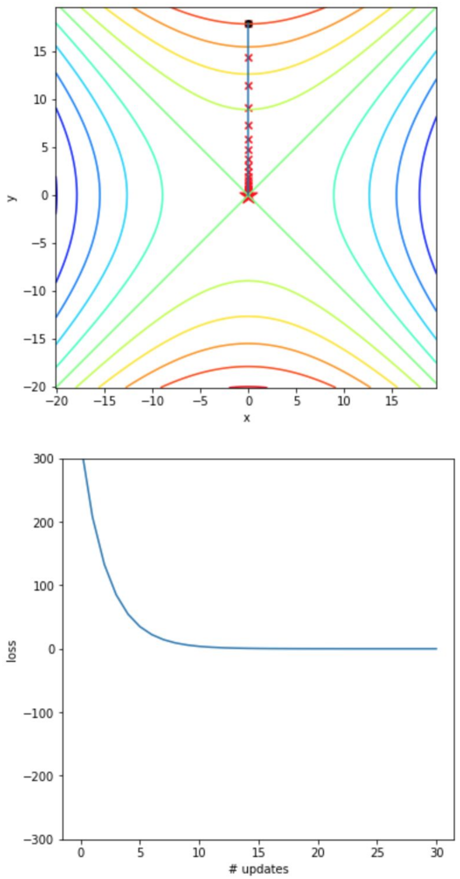


Mila

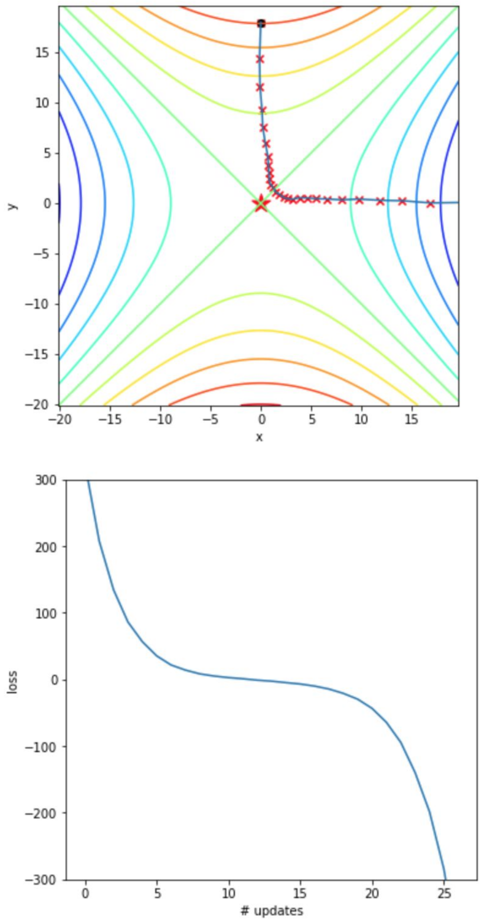
Convergence problem



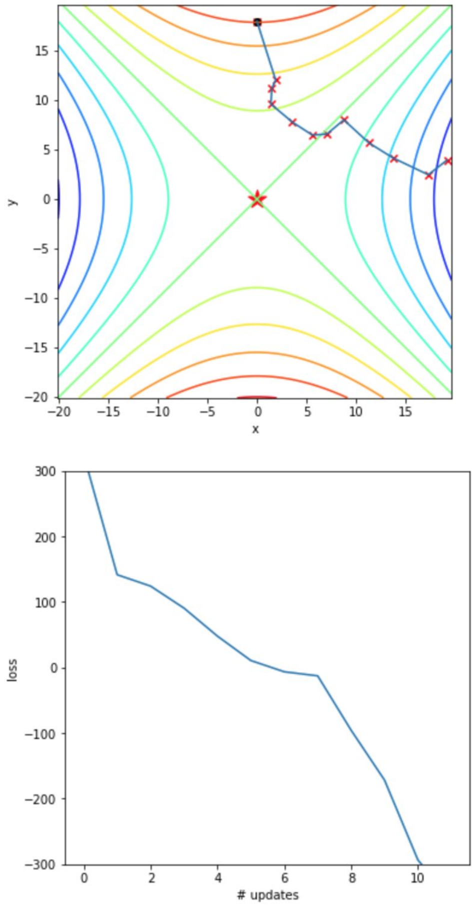
Escape the saddle points



$$\sigma^2 = 0$$



$$\sigma^2 = 0.1$$



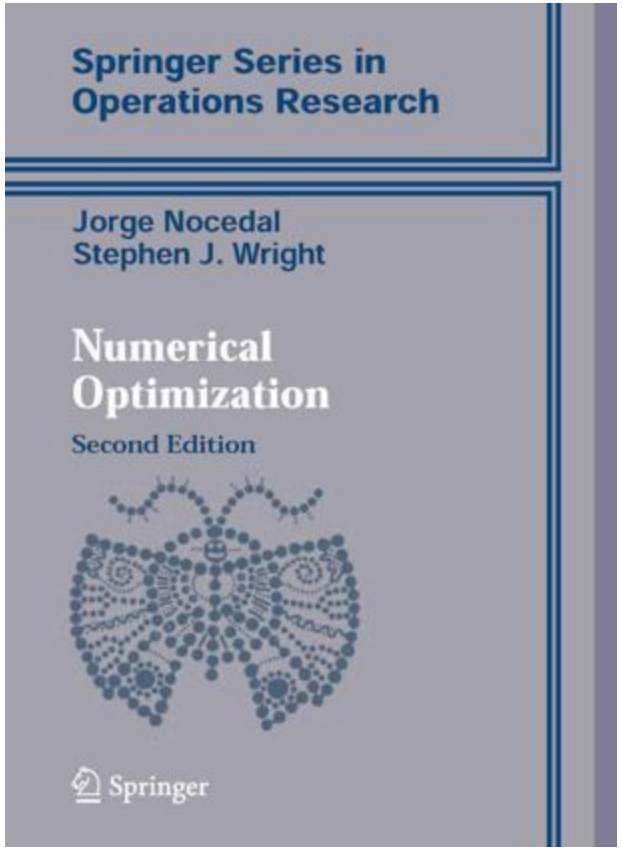
$$\sigma^2 = 1$$

Take-home message

- Machine learning is not only about optimization.
- Preconditioning facilitates optimization.
- Momentum leads to better results.
- Noise can have a beneficial effect for escaping saddle points.

References

- Explanation of the effect of noise on saddle points
- Explanation of the effect of momentum
- The chapter on numerical calculations in the book
  Deep Learning by Goodfellow, Bengio and Courville
- Explanation of Adam by Andrew Ng
- How to reduce learning rate by Andrew Ng
- Description of the different optimizers in DL



Mila