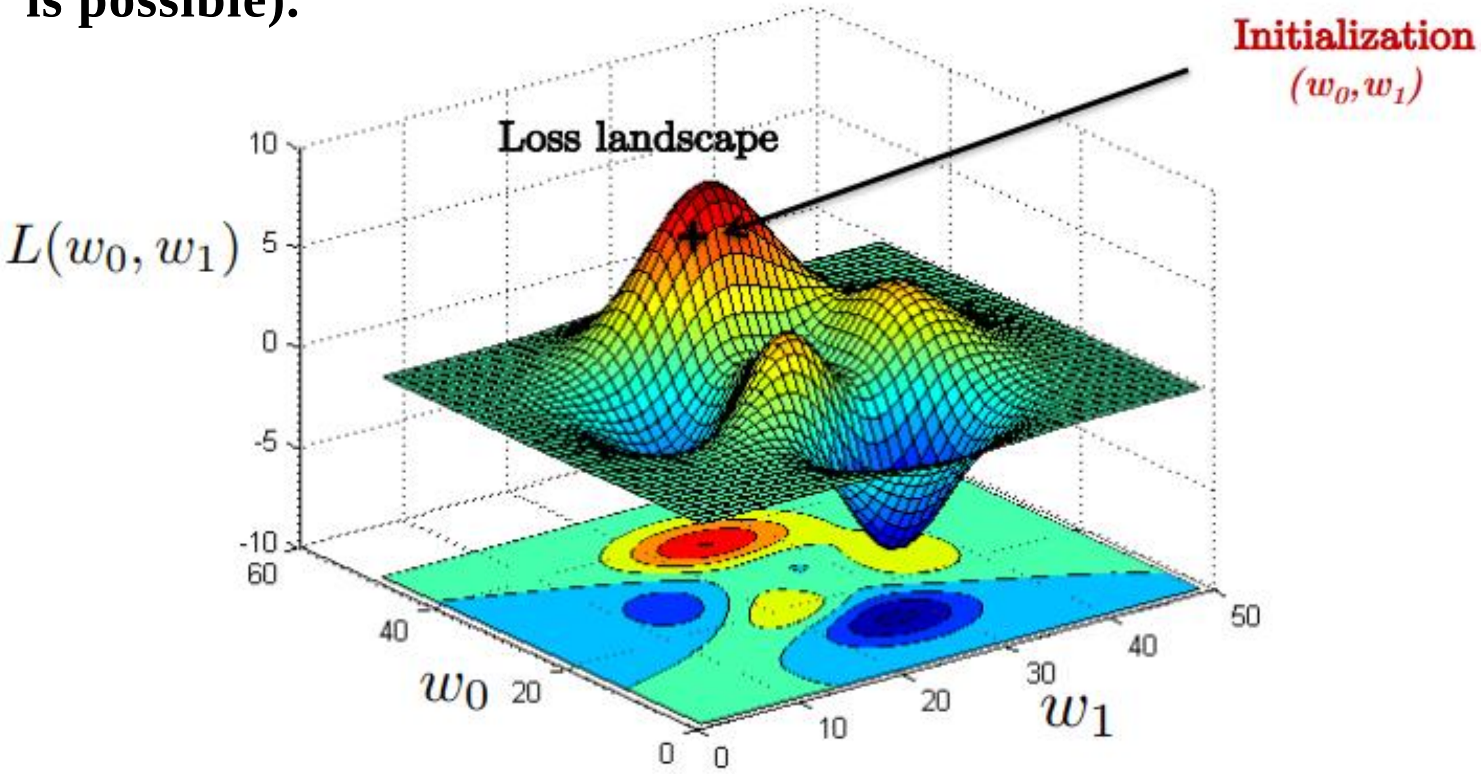
인공지능과 수학적 배경

Optimization by gradient descent

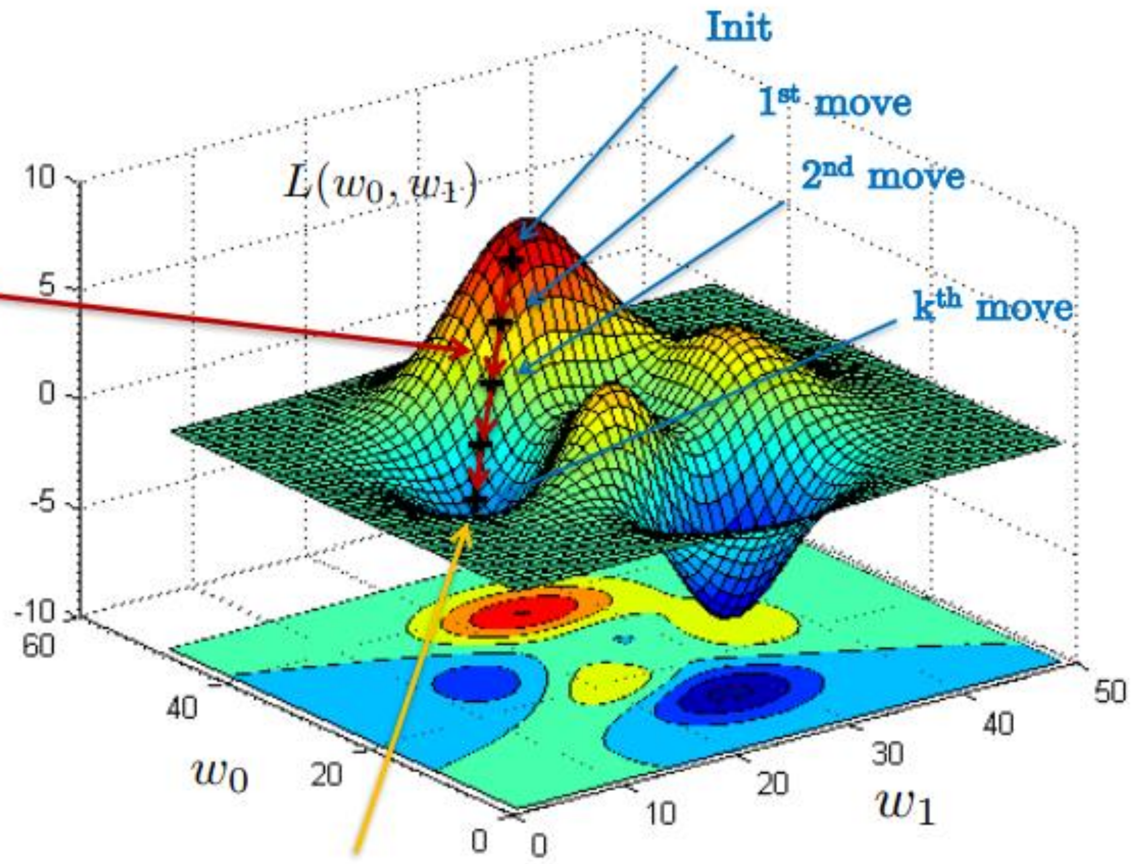
- **Gradient descent technique:**
  - **Initialization**: Start with some values (w_0, w_1).
  - **Iterate (loop)**: Keep updating (w_0, w_1) to reduce the loss value $L(w_0, w_1)$ until a minimum is reached (when no possible update is possible).



Linear Regression

Dynamic process

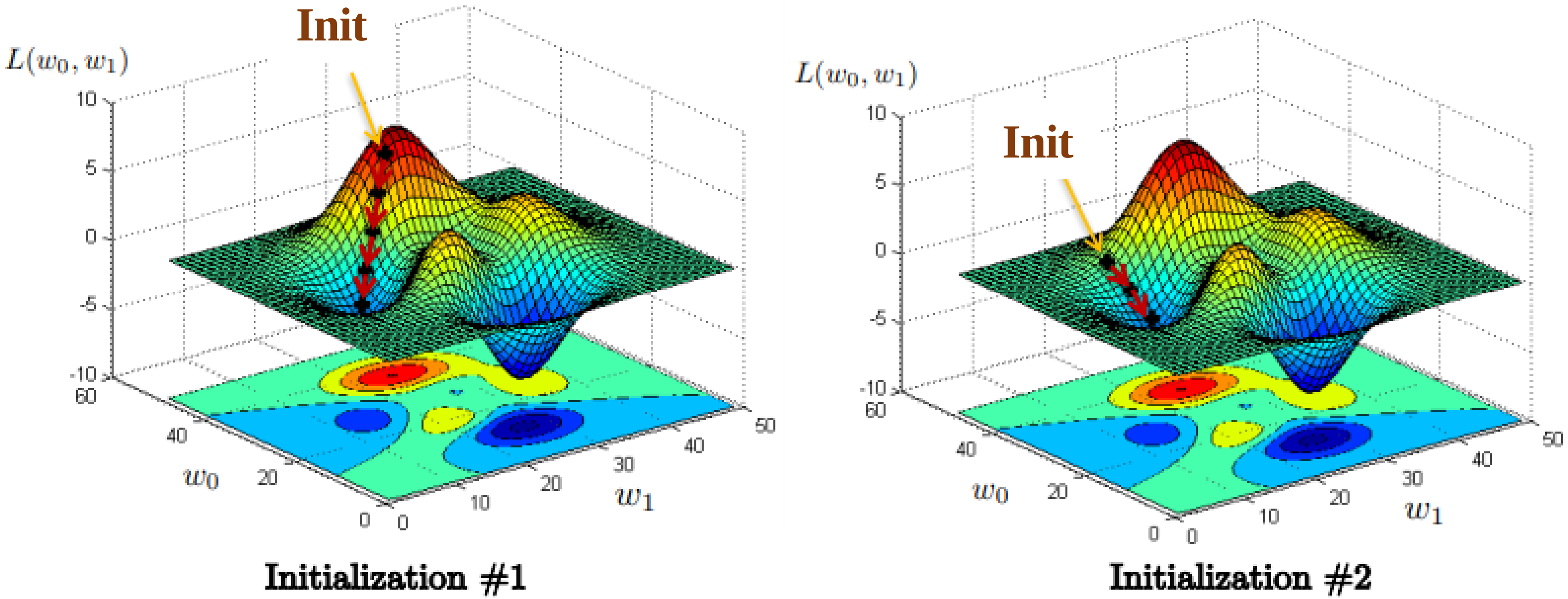
Idea: Move the values of (w_0, w_1) in the direction of the steepest descent to decrease the value of the loss L as fast as possible.



Minimum
(w_0, w_1)

Linear Regression
Initialization

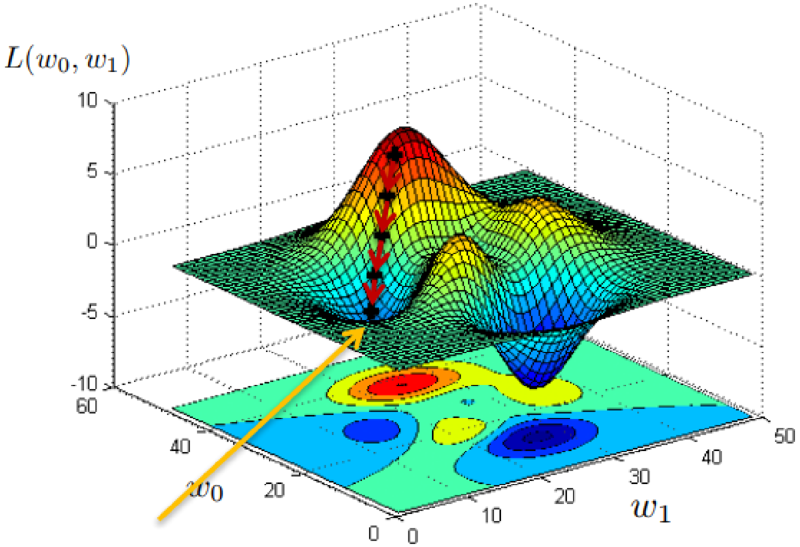
- **How to start?**
  - **Initialization can be random.**
  - **It can also be estimated if additional information on L is available.**



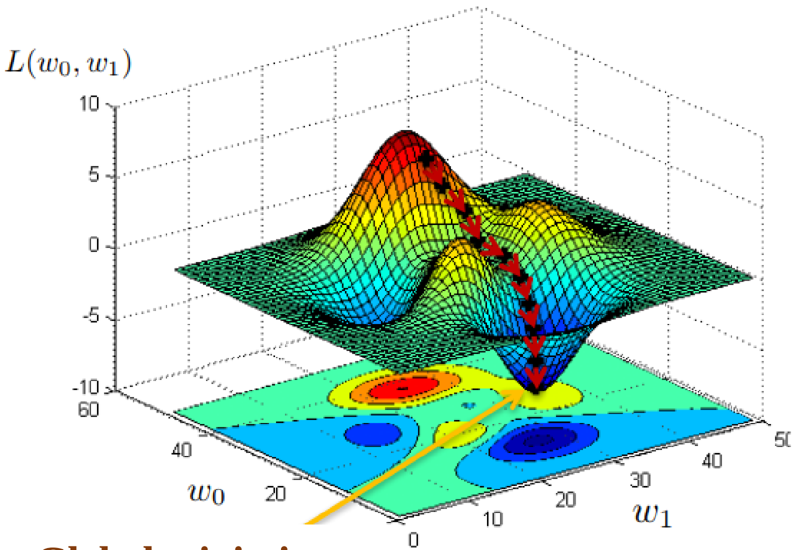
- **Optimization scheme is faster with initialization #2 because the optimization starts closer to the solution.**

Linear Regression
Initialization

- **Good initialization can be essential:**
  - **Start closer to the minimizer $\Rightarrow$ Quicker convergence.**
  - **If bad choice then gradient descent captures local minimizer (not as good as global minimizer).**



Local minimizer

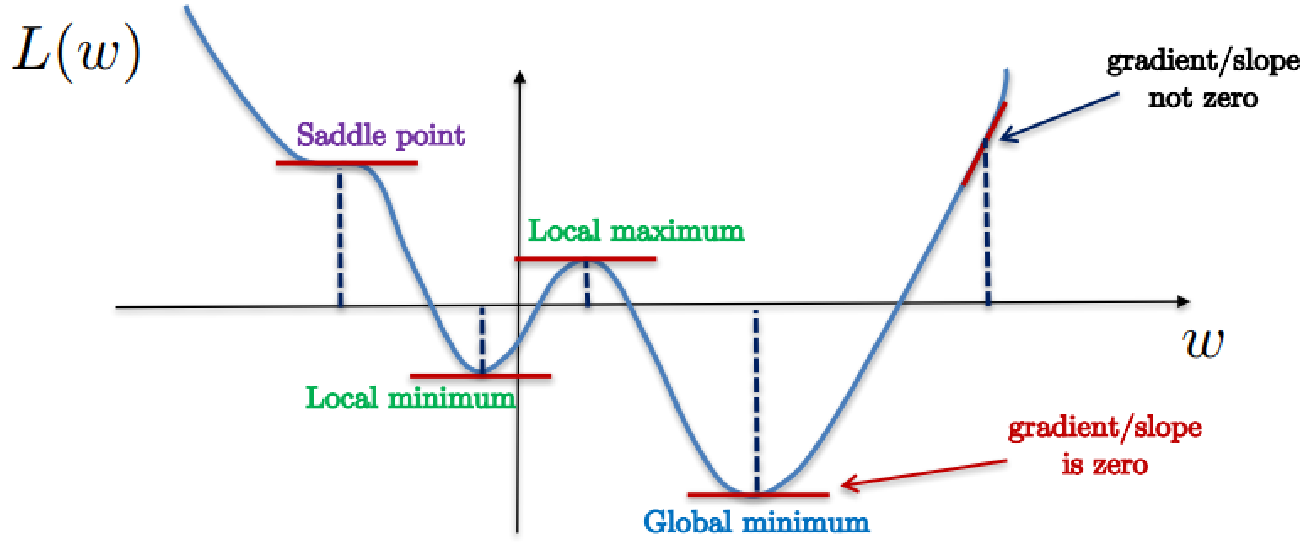


Global minimizer

$$\min_{w=(w_0,w_1)} L(w_0, w_1)$$

Minimizers

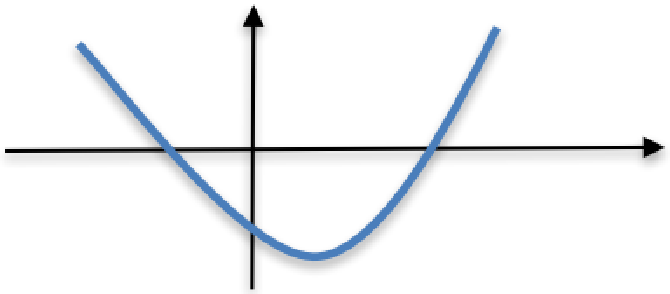
- **Stationary/critical points**: Points where the **gradient/slope** of the function is **zero**.
- **Characterization** of stationary points:
  - **Global** minimizers (maximizers)
  - **Local** minimizers (maximizers)
  - **Saddle** points



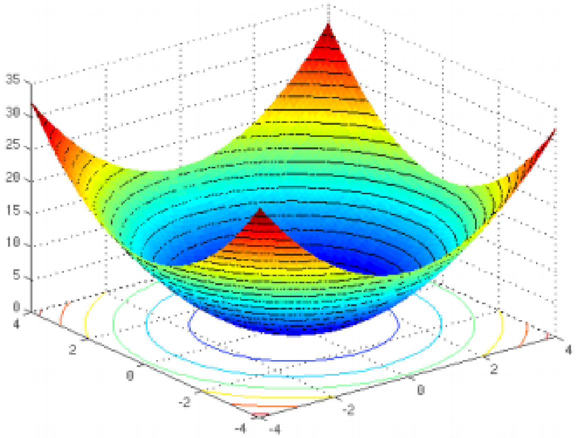
$$\min_{w=(w_0,w_1)} L(w_0, w_1)$$

Convexity vs non-convexity

- **Convex losses are mathematically well studied: Fast optimization techniques, well understood behaviors and properties (existence of global minimum).**



1D convex function

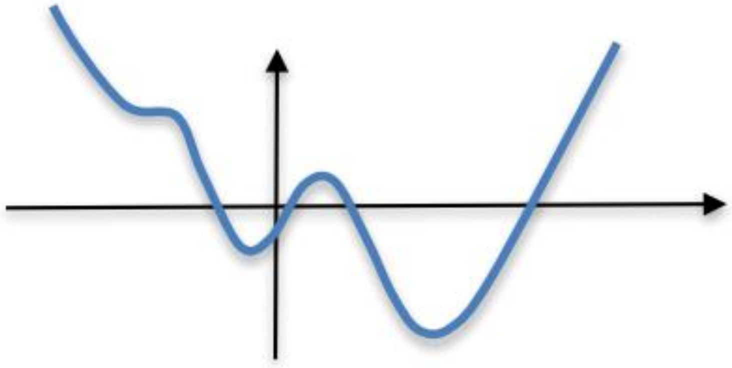


2D convex function

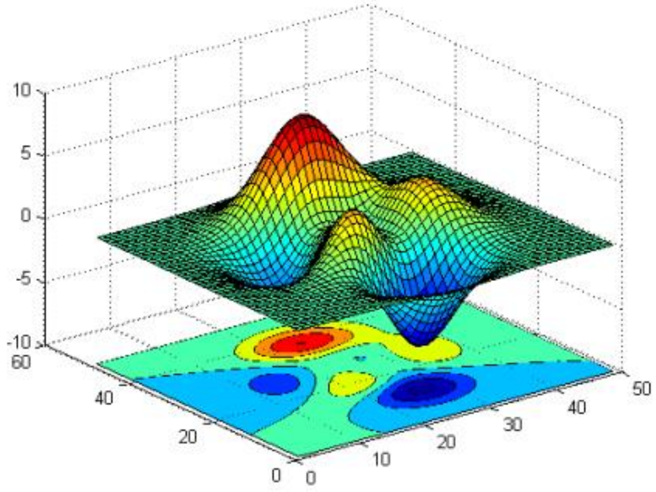
Linear Regression

Convexity vs non-convexity

- Non-convex losses are in general not well understood, are slow to optimize (gradient descent), have critical points, but they offer large learning capacity.



1D non-convex function



2D non-convex function

Formalization

- **Repeat until convergence** **(until reaching a stationary point):**

$$\underbrace{w_j}_{\substack{j^{th}\text{ variable of } w \\ w=(w_0,w_1,w_2,\dots) \\ j=(0,1,2,\dots)}} \overbrace{\leftarrow}^{\substack{\text{Assignment} \\ \text{operator}}} w_j - \overbrace{\tau}^{\substack{\text{Learning} \\ \text{rate}}} \underbrace{\underbrace{\frac{\partial}{\partial w_j}}_{\substack{\textit{Derivative w.r.t.} \\ j^{th}\textit{ variable}}} L(w)}_{\substack{\textit{Gradient of } L \\ \textit{w.r.t. } j^{th}\textit{ variable}}}$$

Computer notation

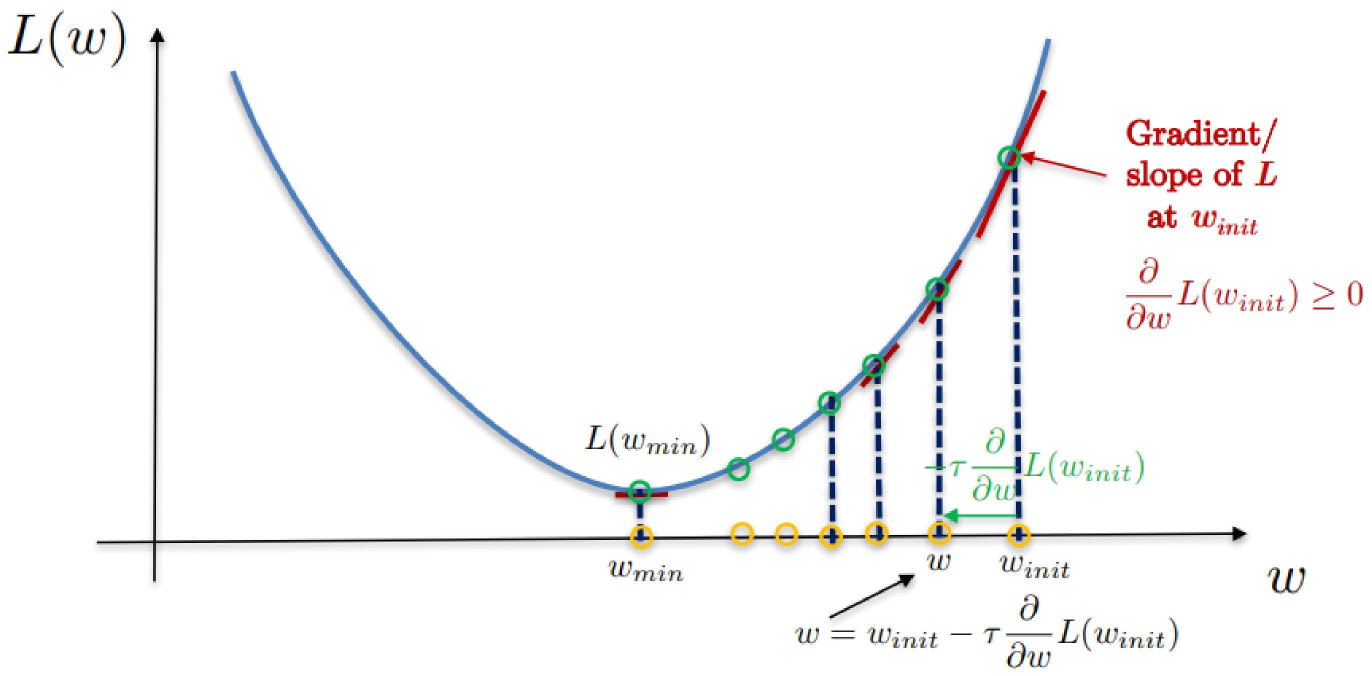
$$\overbrace{w_j^{k+1}}^{\substack{\text{Iteration} \\ \text{number } k+1}} = \overbrace{w_j^{k}}^{\substack{\text{Iteration} \\ \text{number } k}} - \tau \frac{\partial}{\partial w_j} L(w^k)$$

Math notation

Linear Regression

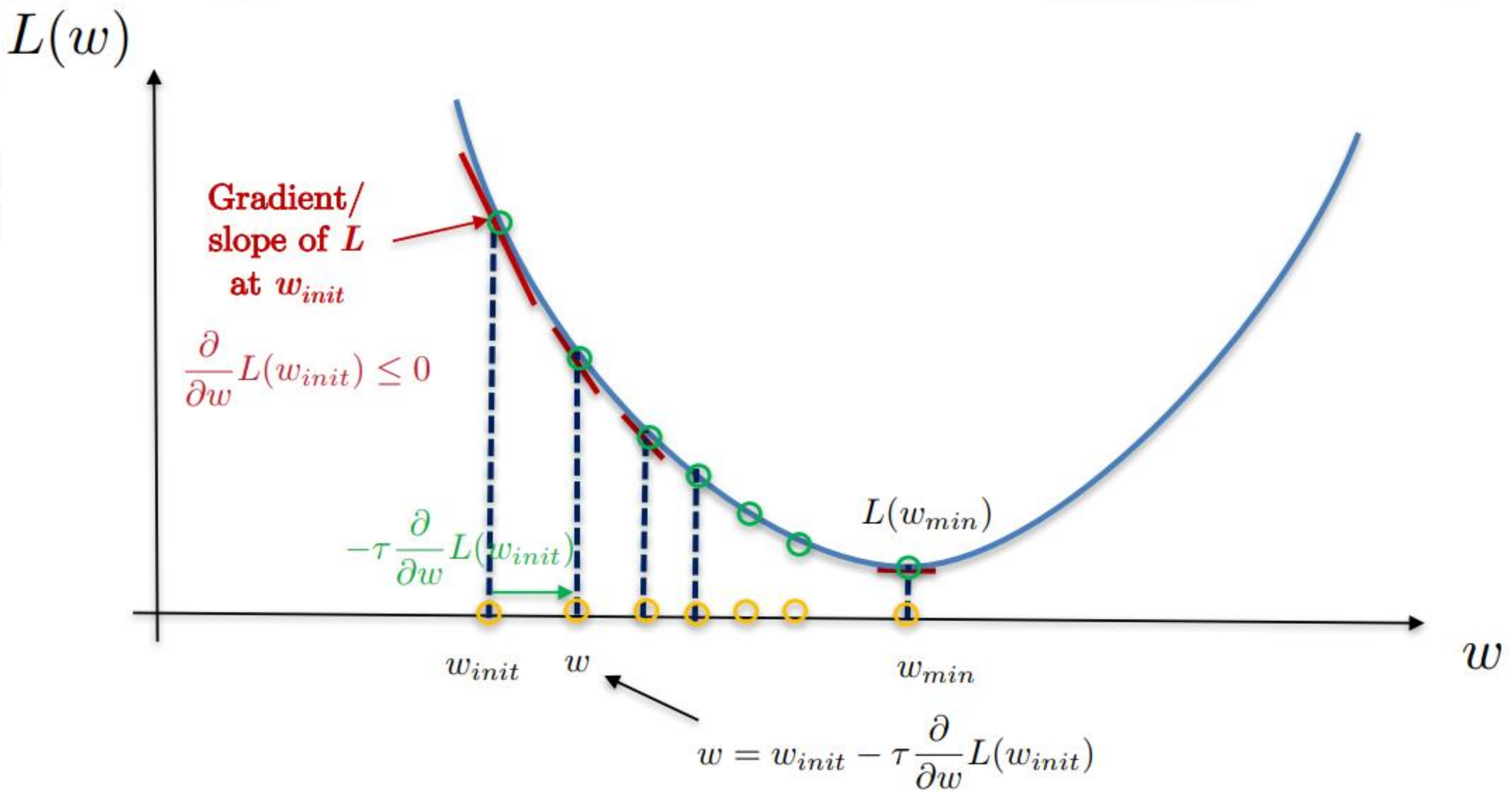
Gradient descent with one variable

- **Let us consider a single parameter w:**
  - **Prediction function:** $f_w(x) = wx$
  - **Loss optimization:** $\min_w \; L(w) = \frac{1}{n}\sum_{i=1}^{n}\left(wx_i - y_i\right)^2$
  - **Gradient descent:** $w \leftarrow w - \tau\frac{\partial}{\partial w}L(w)$



Linear Regression

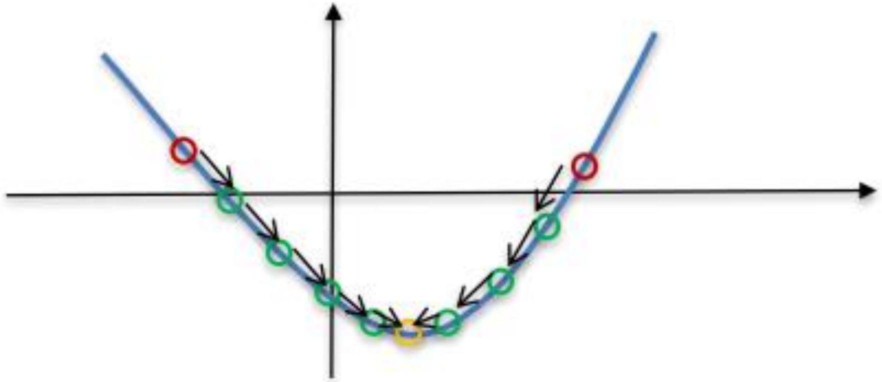
Different initialization



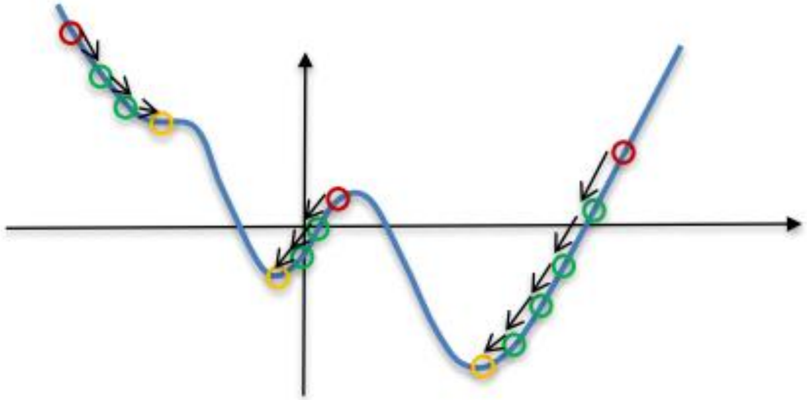
- **Any initialization of the gradient descent does converge to the solution of the optimization problem. Only the optimization time is affected.**
  Why? Loss is convex.

Convexity vs non-convexity with GD

- Gradient descent techniques with convex function: Only global minimizers, GD will converge for all initializations.



- Gradient descent techniques with non-convex function: Critical points ⇒ Choice of initialization is critical for global minimizers, local minimizer, saddle points.

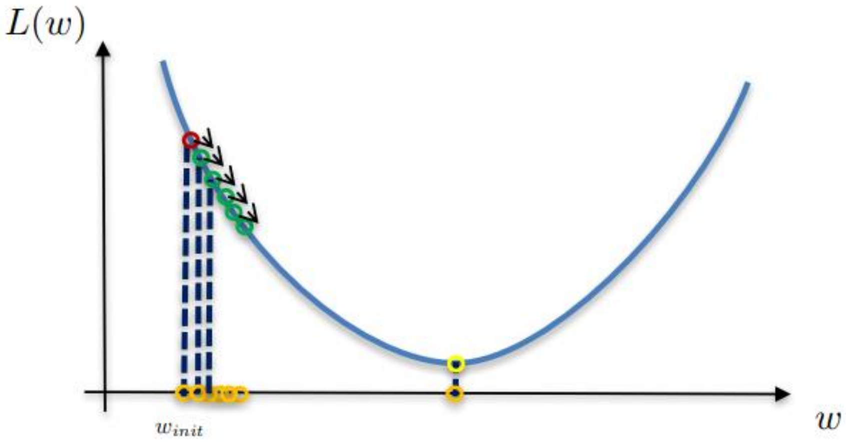


Linear Regression

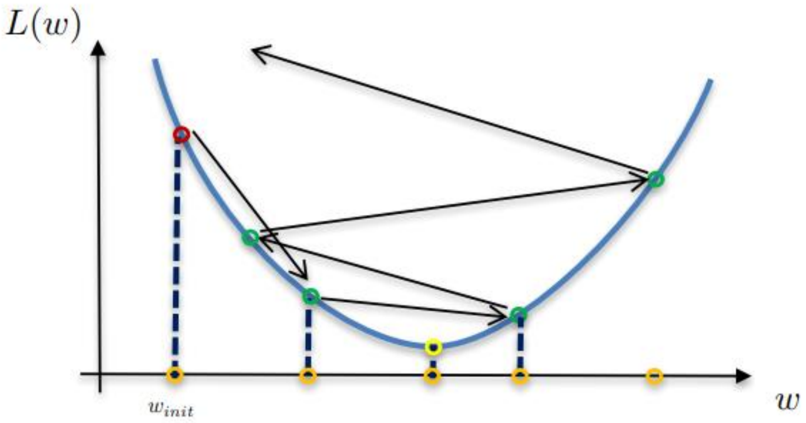
Learning rate

- Influence of learning rate/ time step in gradient descent: $$w \leftarrow w - \tau \frac{\partial}{\partial w} L(w)$$

- If τ is too small, then the optimization process requires a lot of iterations to converge to the solution.

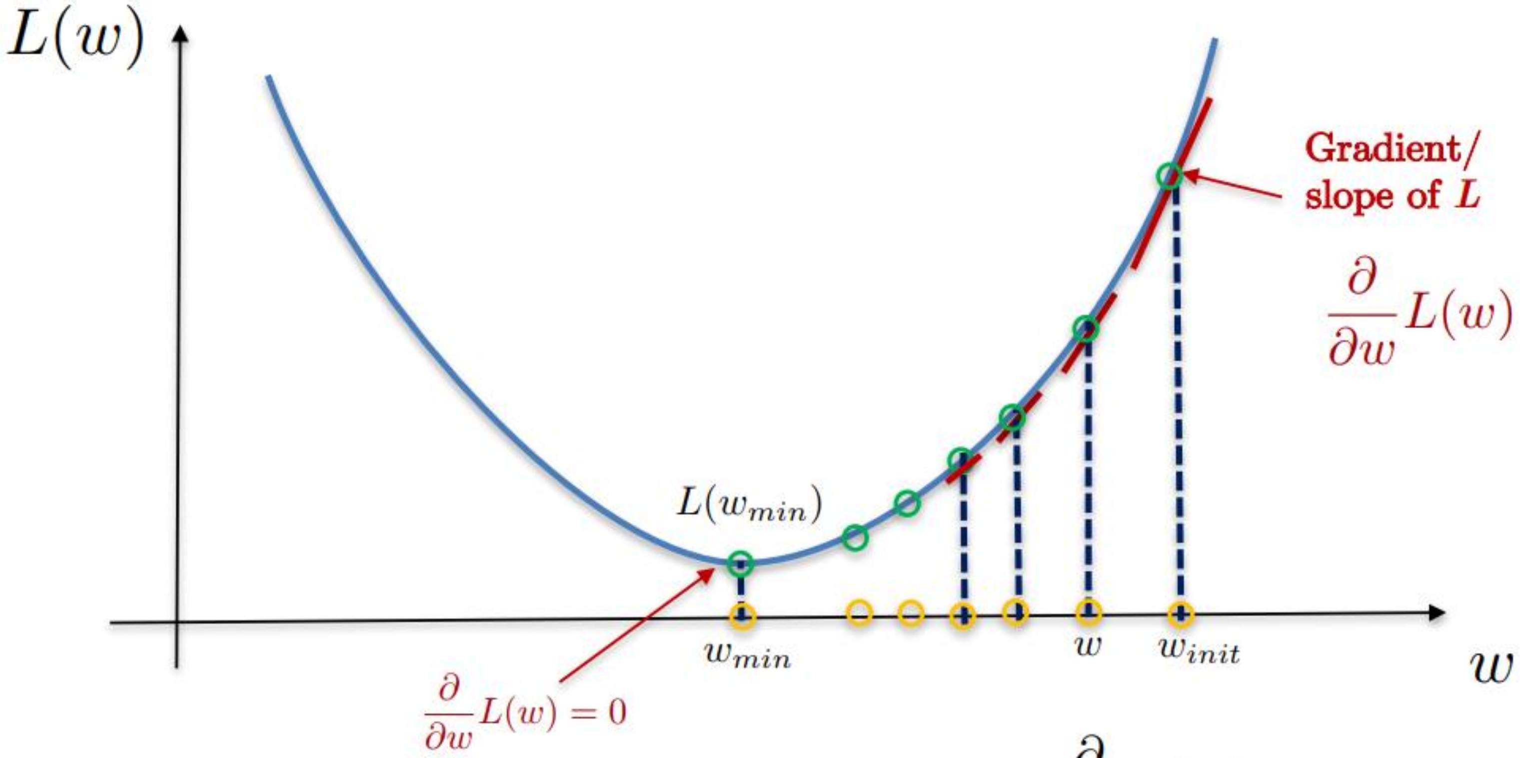


- If τ is too large, then the optimization process does not converge (diverges) and cannot capture the minimum (loss value explodes).



Convergence speed

- **Do we need to decrease the learning rate τ to guarantee convergence?**
  No. The slope/gradient decreases its value when we get closer to the minimum.



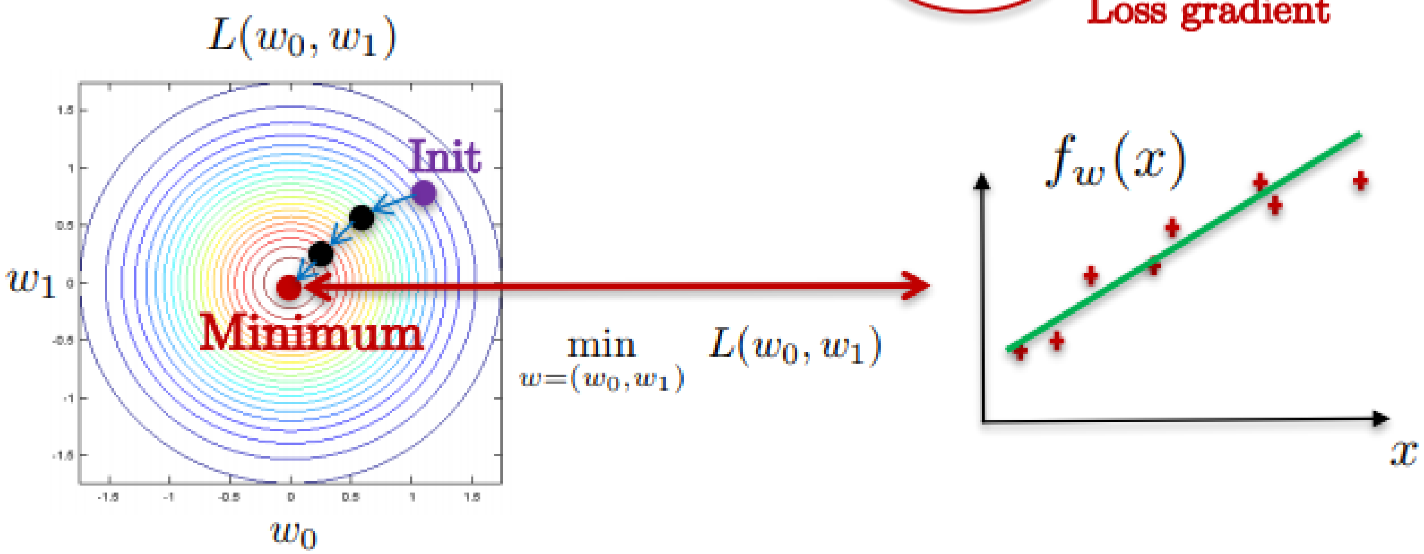
$$w \leftarrow w - \tau \frac{\partial}{\partial w} L(w)$$

Linear Regression

Gradient descent for linear regression

- **Prediction function:** $f_w(x) = w_0 + w_1 x$
- **Parameters:** w_0, w_1
- **Loss function:** $L(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_i - y_i\right)^2$
- **Optimization:** $\min_{w=(w_0,w_1)} L(w_0, w_1)$
- **Gradient descent:** $w_j \leftarrow w_j - \tau \frac{\partial}{\partial w_j} L(w)$

Loss gradient



Linear Regression

Loss gradient

- **For any w_j:**

$$\frac{\partial}{\partial w_j} L(w_0, w_1) = \frac{\partial}{\partial w_j}\left[\frac{1}{n}\sum_{i=1}^{n}\Big(w_0 + w_1 x_i - y_i\Big)^2\right]$$

- **For w_0:**

$$\begin{aligned}\frac{\partial}{\partial w_0} L(w_0, w_1) &= \frac{\partial}{\partial w_0}\left[\frac{1}{n}\sum_{i=1}^{n}\Big(w_0 + w_1 x_i - y_i\Big)^2\right]\\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w_0}\Big(w_0 + w_1 x_i - y_i\Big)^2\\ &= \frac{2}{n}\sum_{i=1}^{n}(w_0 + w_1 x_i - y_i).\frac{\partial}{\partial w_0}(w_0 + w_1 x_i - y_i)\\ &= \frac{2}{n}\sum_{i=1}^{n}(w_0 + w_1 x_i - y_i).1\end{aligned}$$

Linear Regression

Loss gradient

- **For any w_1:**

$$
\begin{aligned}
\frac{\partial}{\partial w_1} L(w_0, w_1) &= \frac{\partial}{\partial w_1}\left[\frac{1}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_i - y_i\right)^2\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w_1}\left(w_0 + w_1 x_i - y_i\right)^2 \\
&= \frac{2}{n}\sum_{i=1}^{n}(w_0 + w_1 x_i - y_i).\frac{\partial}{\partial w_1}(w_0 + w_1 x_i - y_i) \\
&= \frac{2}{n}\sum_{i=1}^{n}(w_0 + w_1 x_i - y_i).x_i
\end{aligned}
$$

Gradient descent equation

- **Initialize:** w_0, w_1

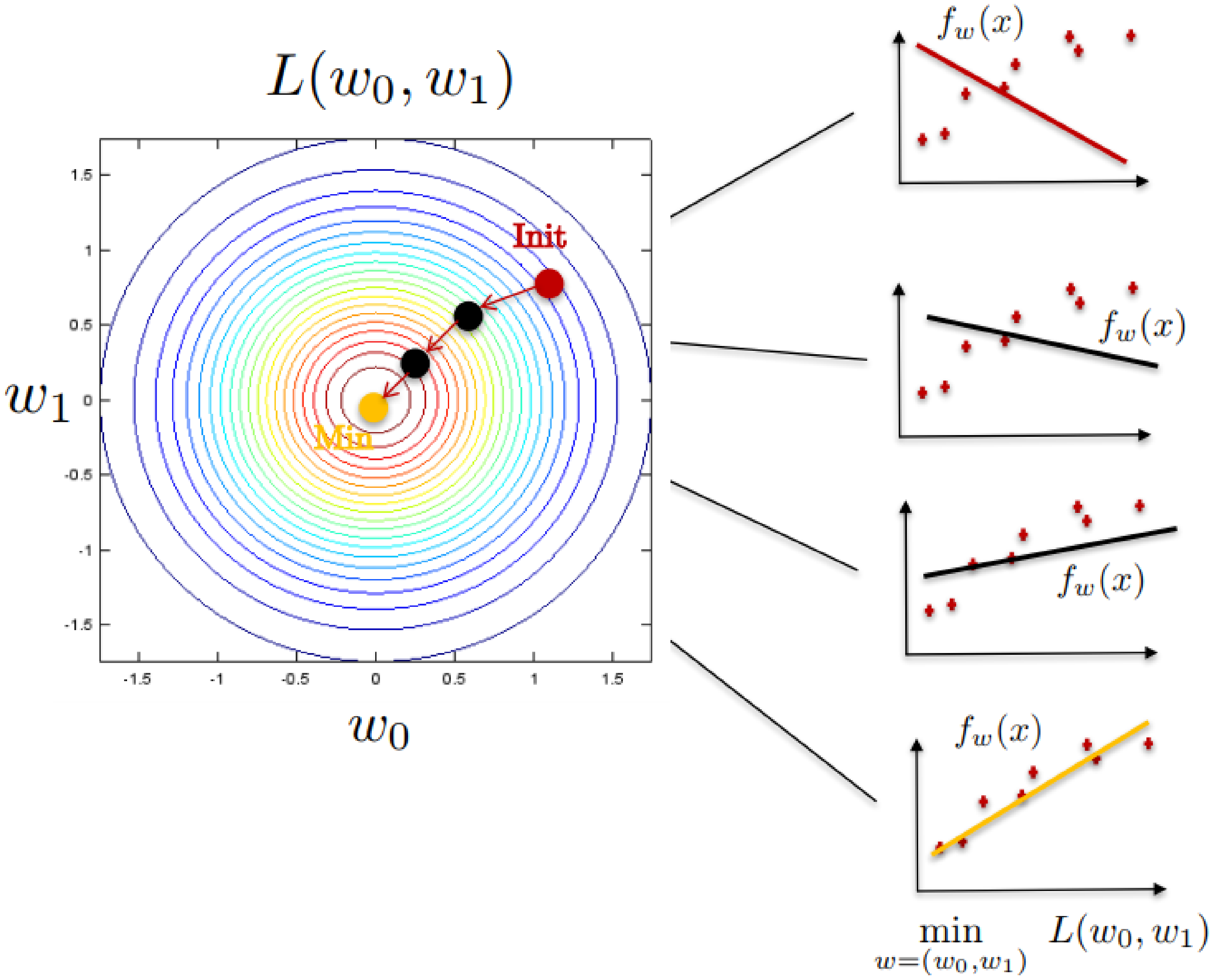
- **Iterate until convergence:**

$$w_0 \leftarrow w_0 - \tau \frac{2}{n} \sum_{i=1}^{n} (w_0 + w_1 x_i - y_i)$$

$$w_1 \leftarrow w_1 - \tau \frac{2}{n} \sum_{i=1}^{n} (w_0 + w_1 x_i - y_i) x_i$$

Linear Regression

Gradient descent in action



Stochastic vs deterministic GD

- **Training set size is *n*:**

$$w_j \leftarrow w_j - \tau \frac{\partial}{\partial w_j} L(w)$$

$$\text{with } L(w) = \frac{1}{n} \sum_{i=1}^{n} \left(w_0 + w_1 x_i - y_i \right)^2$$

$$\Downarrow$$

$$w_j \leftarrow w_j - \tau \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w_j} \left(w_0 + w_1 x_i - y_i \right)^2$$

- **If n is small a few K, but if n is as large as M or B?**

  ⇒ **Stochastic gradient descent (neural networks)**