Gradient descent

Iterative optimization

Main reference: J. Watt et al., “Machine Learning refined”, Cambridge University Press

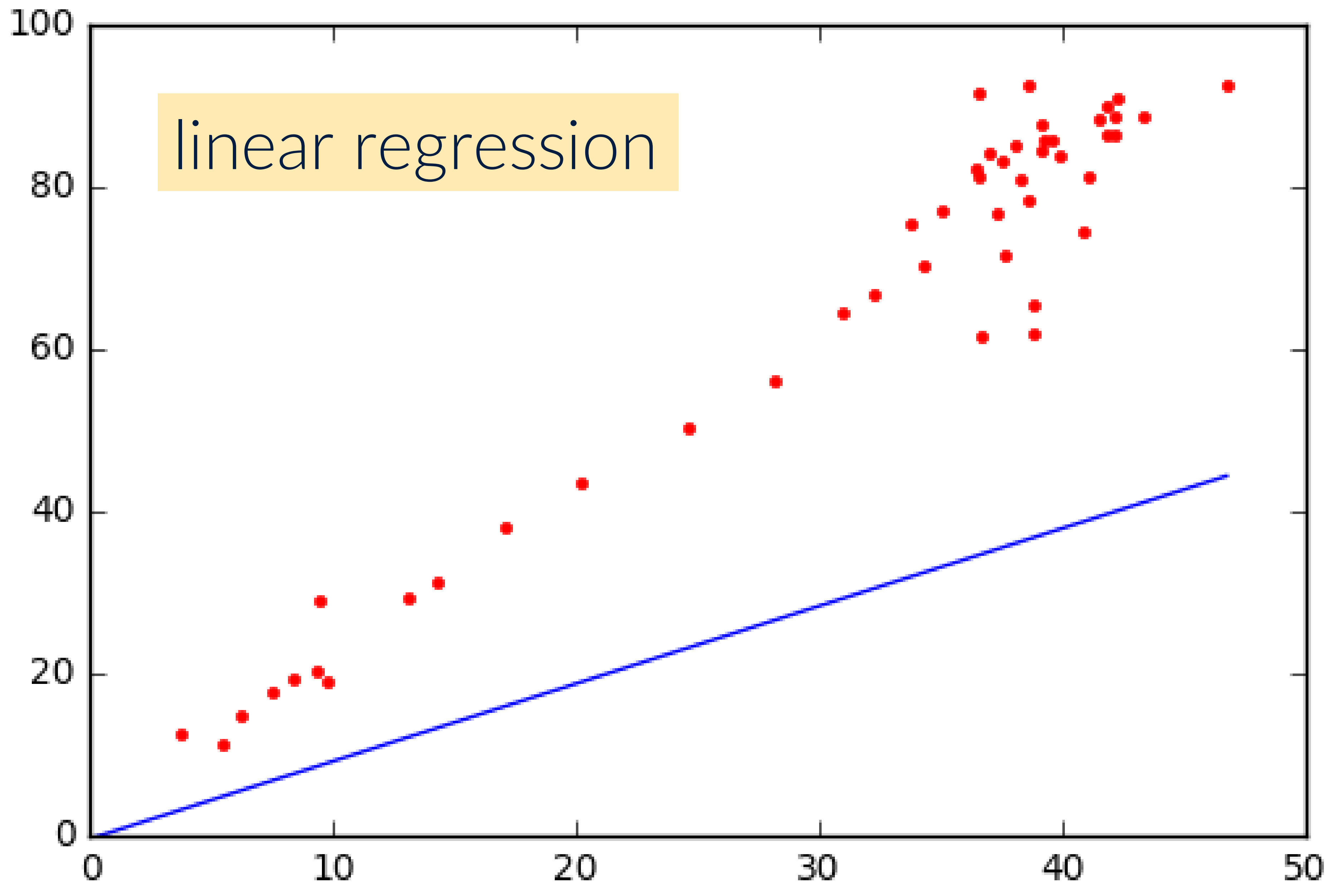
EPFL

ETH zürich

What is wrong with OLS?

- Nothing! **Gradient Descent** is more general in that it can apply to **any optimization problem** by an **iterative** process
- The **gradient descent algorithm** is a local optimization method where - at each step - we employ the **negative gradient** as our descent direction

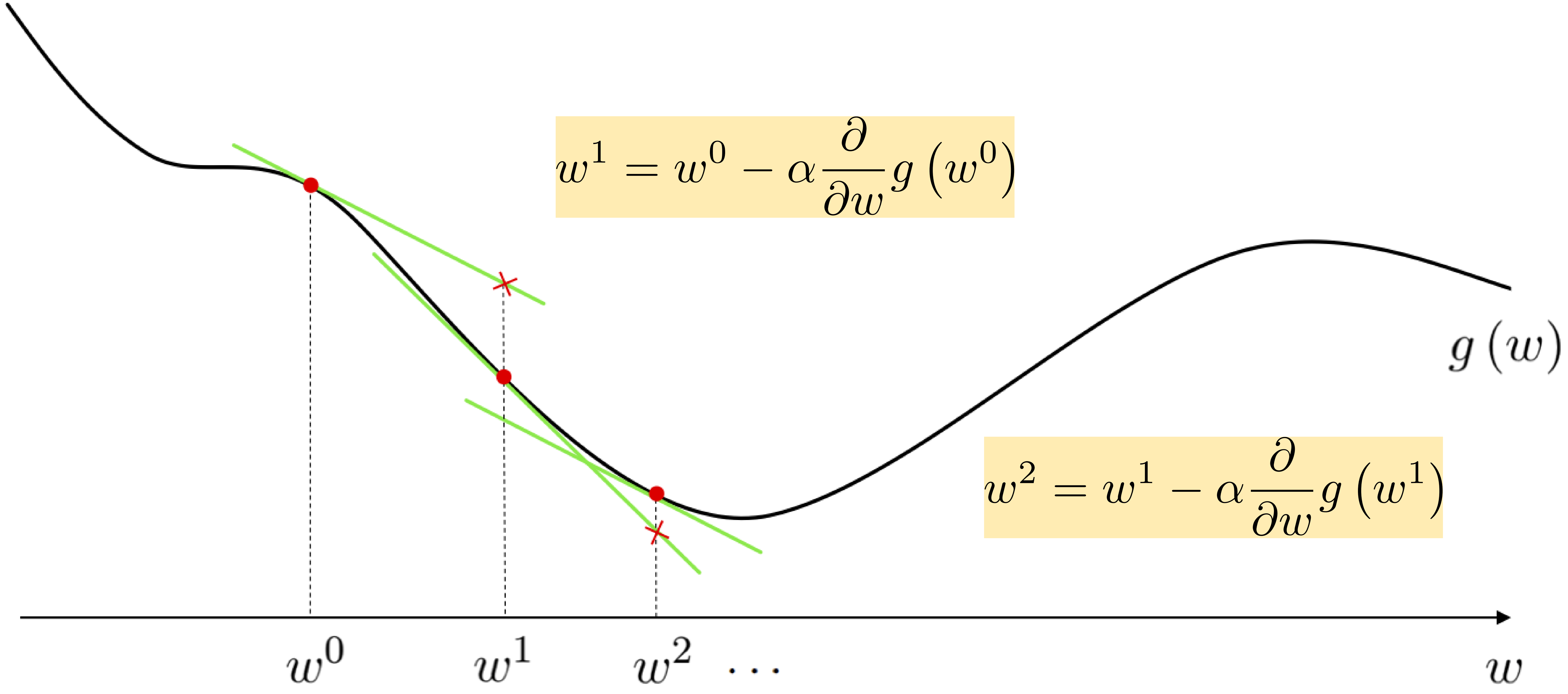
We start with a random parameter vector $\mathbf{w}^0$
At each step, we obtain a new parameter vector $\mathbf{w}^k$



From https://muthu.co/linear-regression-using-gradient-descent-algorithm/

A figurative drawing

- Let's consider a **cost function** $g(w)$, for example the mean squared cost
- The negative **gradient** of the function provides an excellent and easily computed descent direction at each step of this local optimization method



Iterative algorithm

- Let's consider a **cost function** $g(w)$, for example the mean squared cost

1: **input:** function g, steplength α, maximum number of steps K, and initial point $\mathbf{w}^0$

2: `for` $k = 1...K$

3: $\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha \nabla g\left(\mathbf{w}^{k-1}\right)$

with $\nabla g := \begin{bmatrix} \frac{\partial}{\partial w_0} \\ \frac{\partial}{\partial w_1} \\ \vdots \\ \frac{\partial}{\partial w_m} \end{bmatrix} g$

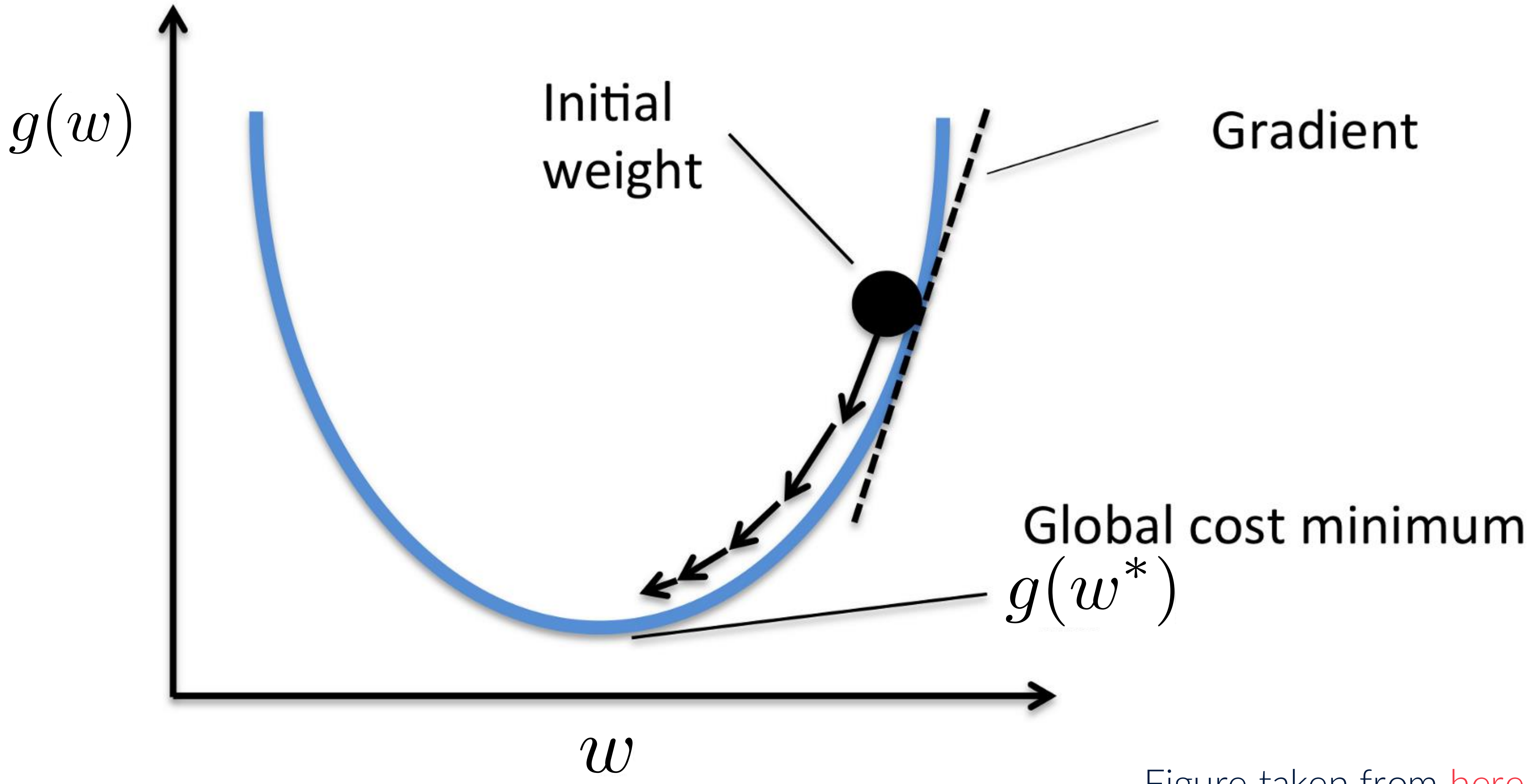
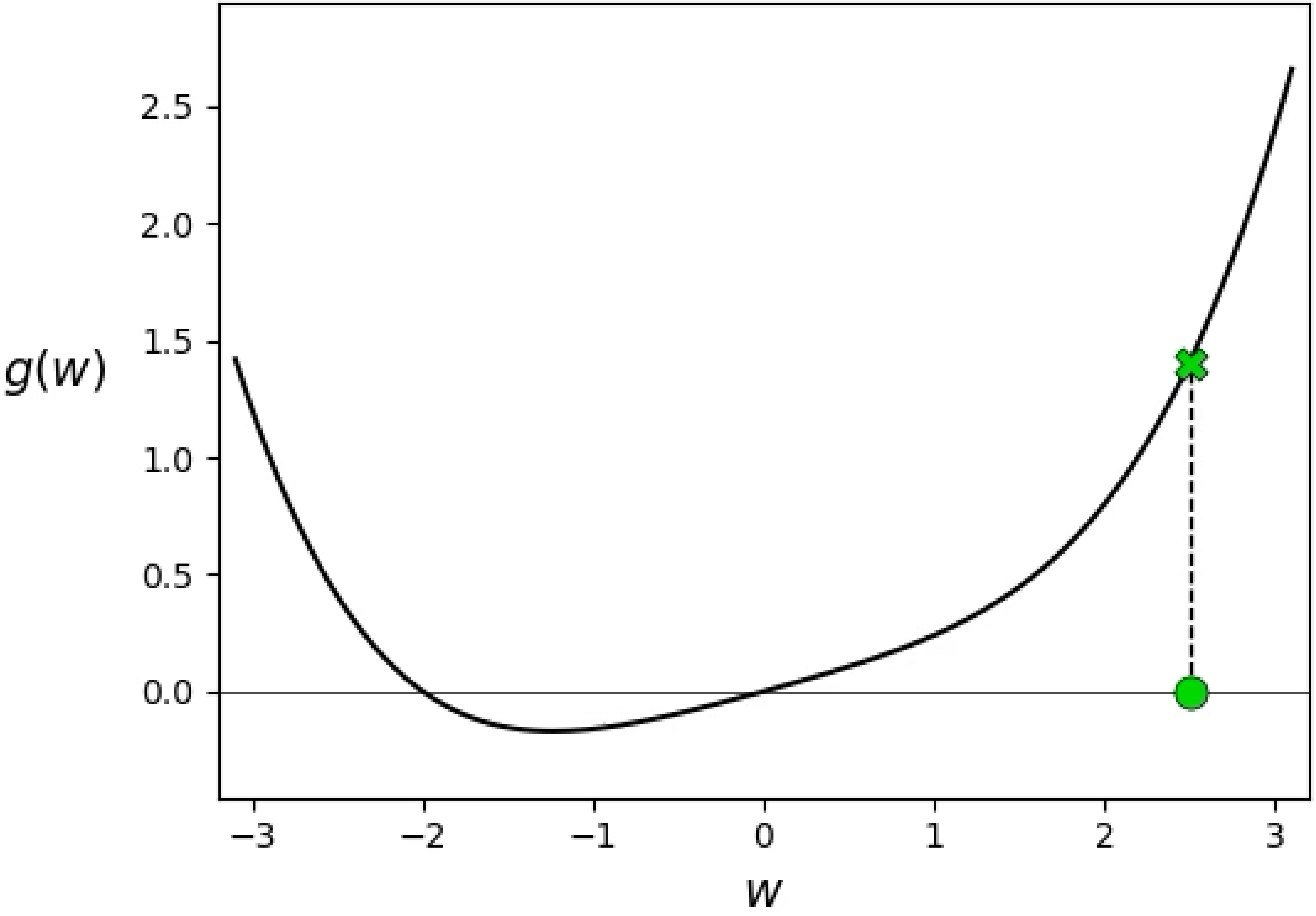
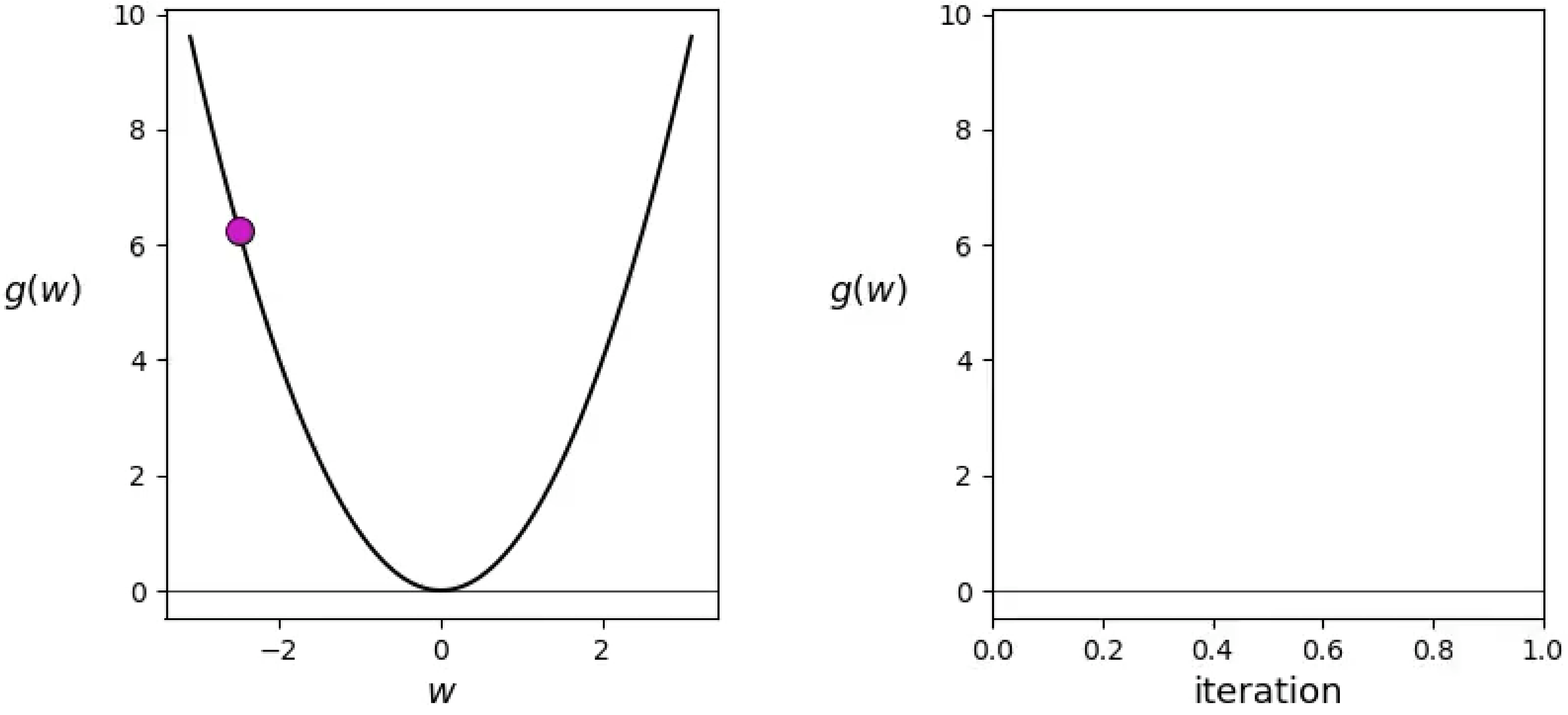


Figure taken from here

Animated illustration

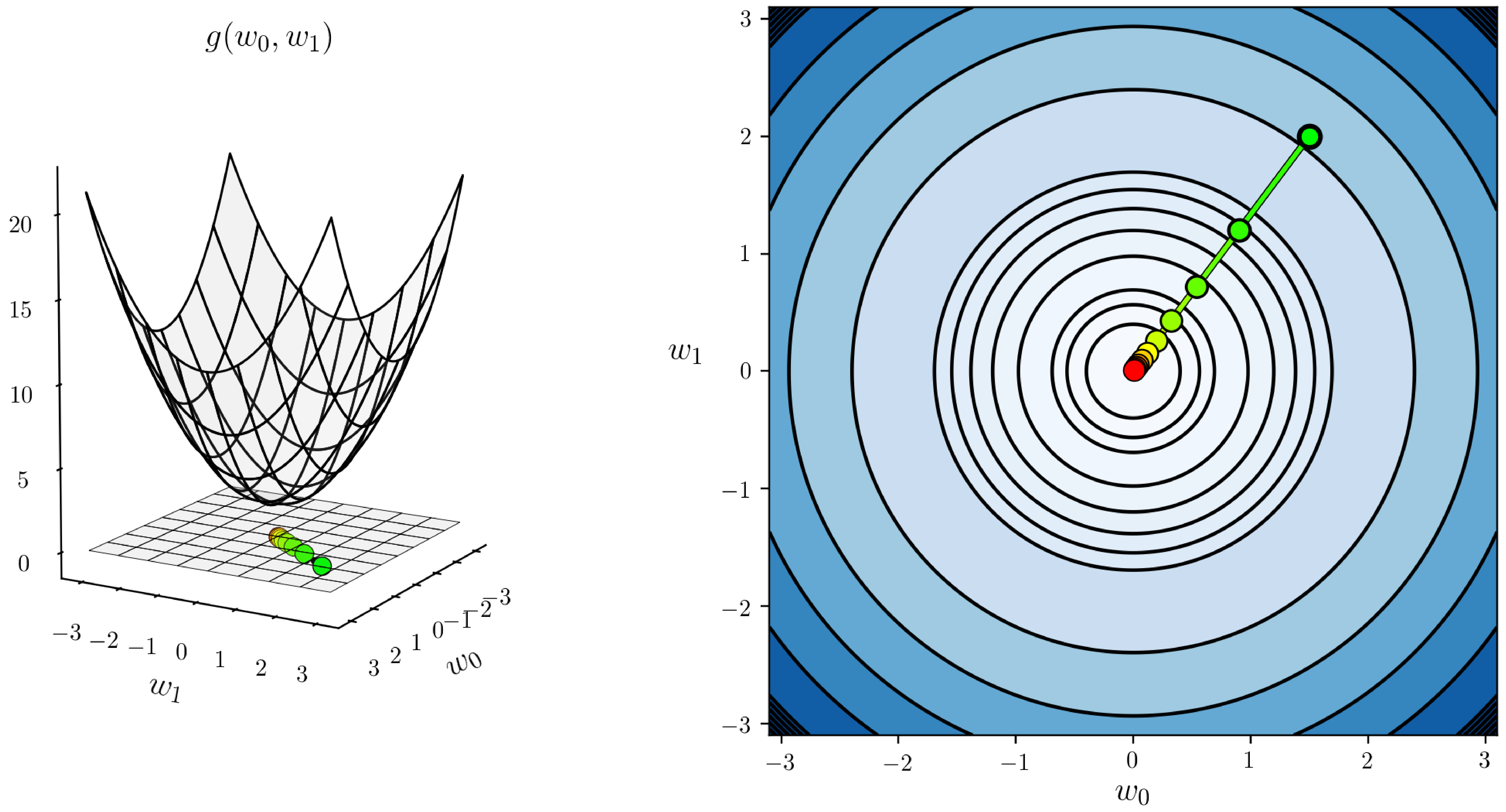


Choice of steplength α (aka learning rate)



Example #1: 2 variables, quadratic cost

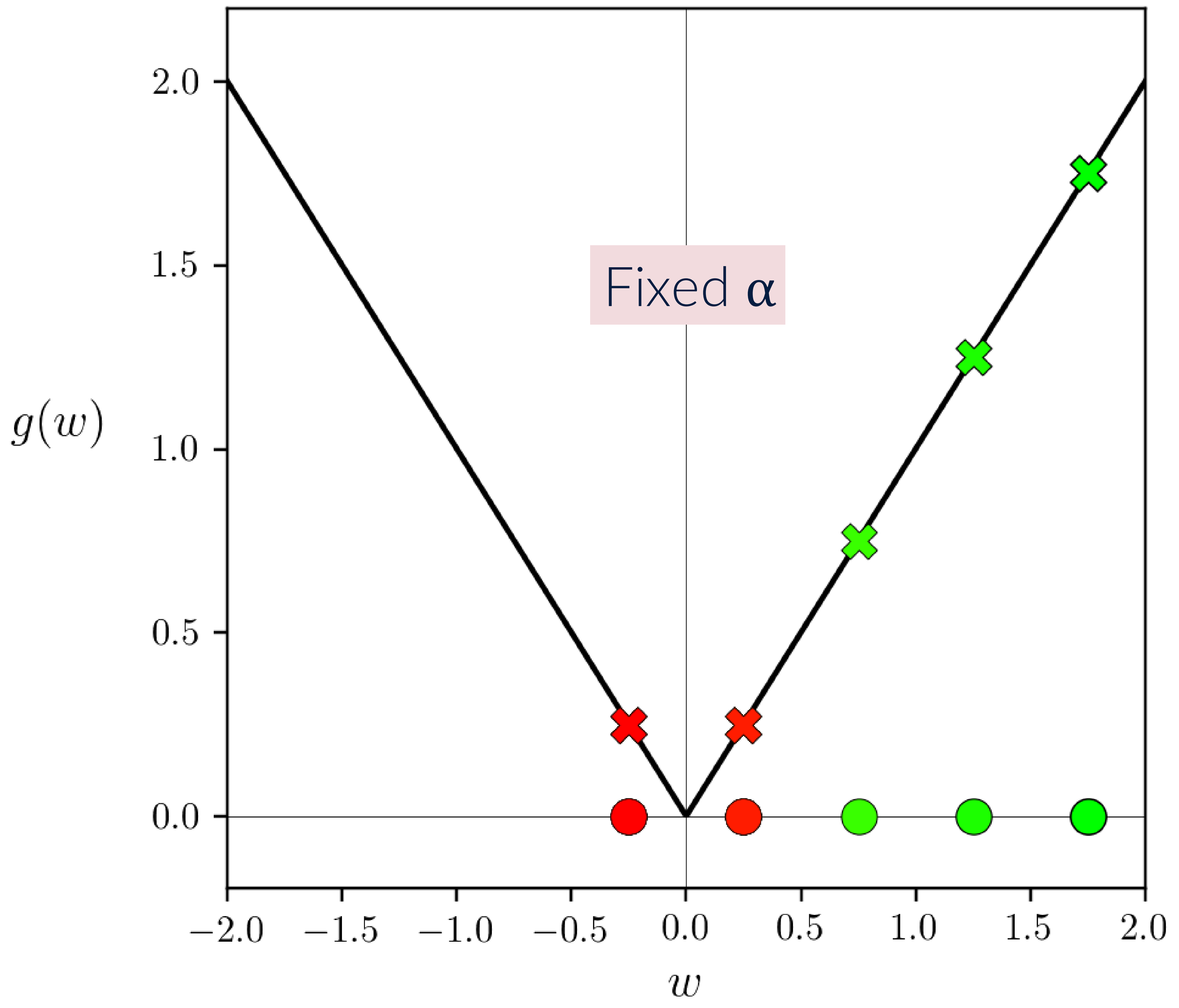
- Let's consider $g(w_0, w_1) = w_0^2 + w_1^2 + 2$ $\qquad \nabla g(\mathbf{w}) = \begin{bmatrix} 2w_0 \\ 2w_1 \end{bmatrix}$



Example #2: 1 variable, absolute cost

- Let's consider $g(w) = |w|$

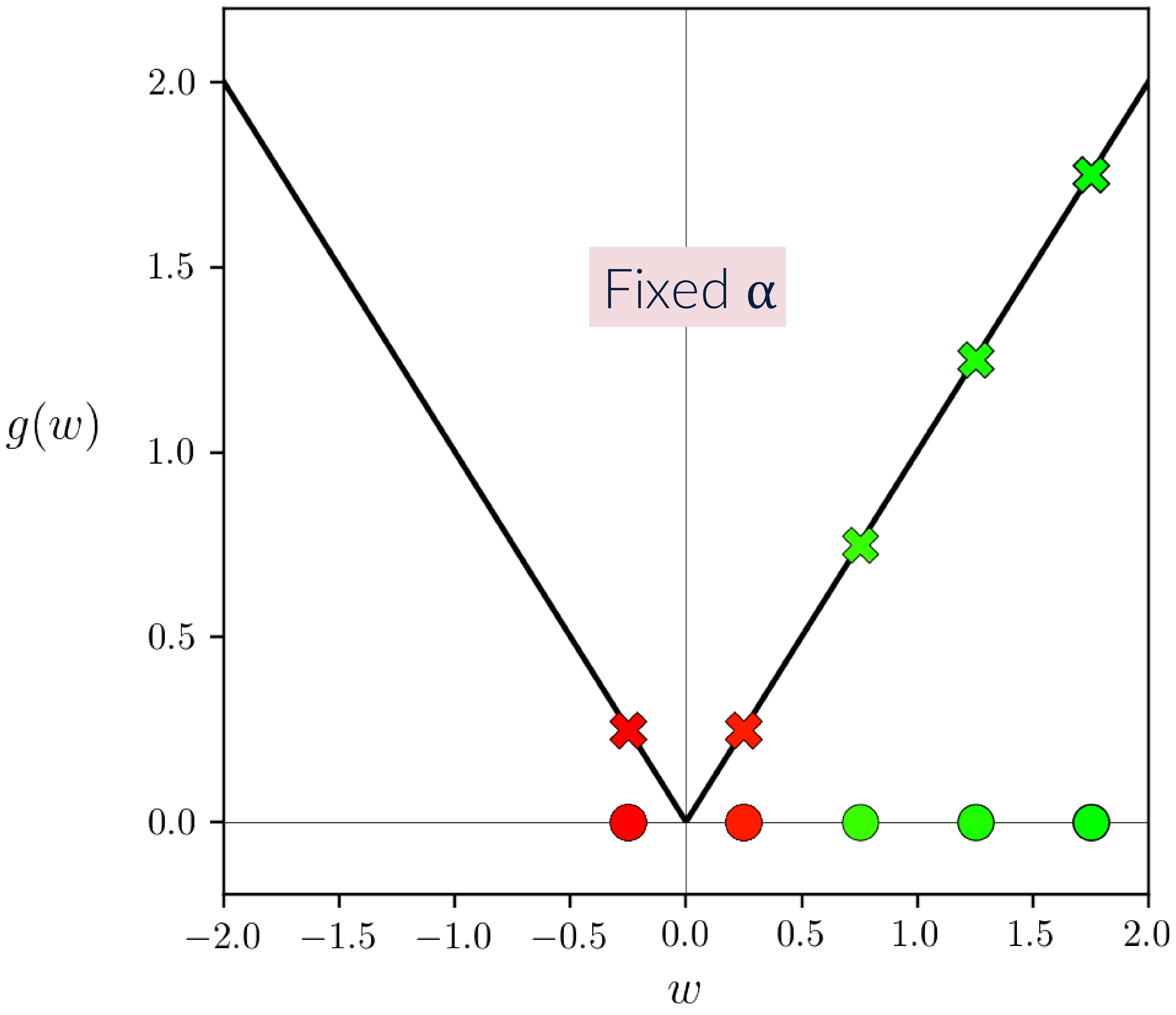
$$\frac{\mathrm{d}}{\mathrm{d}w} g(w) = \begin{cases} +1 & \text{if } w > 0 \\ -1 & \text{if } w < 0. \end{cases}$$

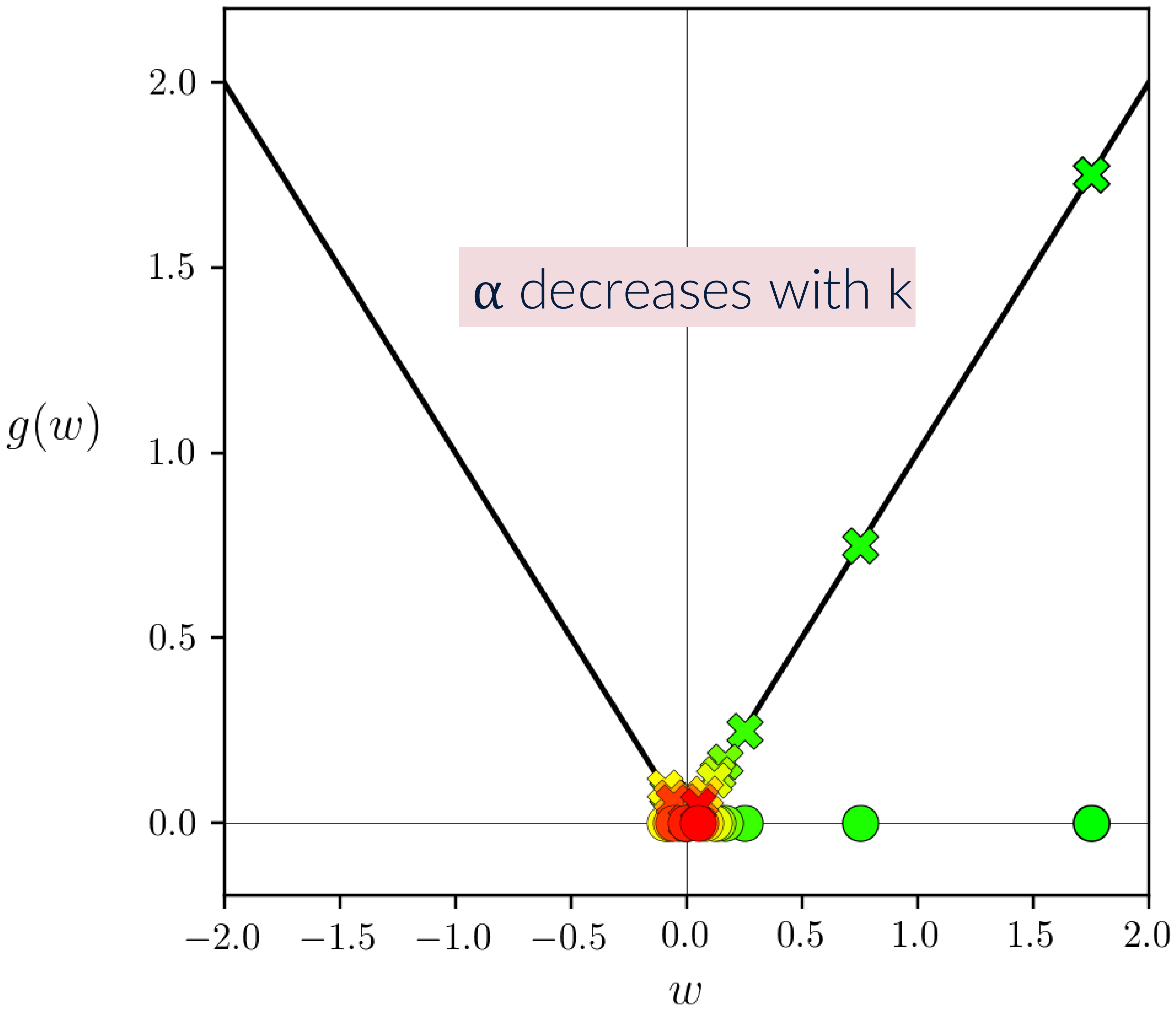


Example #2: 1 variable, absolute cost

- Let's consider $g(w) = |w|$

$$\frac{\mathrm{d}}{\mathrm{d}w} g(w) = \begin{cases} +1 & \text{if } w > 0 \\ -1 & \text{if } w < 0. \end{cases}$$





Common cost functions and their gradient

Mean squared cost function:

$$g_{\mathrm{sl}}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}\left(y^i - \tilde{\mathbf{X}}^i\mathbf{w}\right)^2$$

$$\nabla g_{\mathrm{sl}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\tilde{\mathbf{X}}^i\left(\tilde{\mathbf{X}}^i\mathbf{w} - y^i\right)$$

Mean absolute cost function:

$$g_{\mathrm{al}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left|y^i - \tilde{\mathbf{X}}^i\mathbf{w}\right|$$

$$\nabla g_{\mathrm{al}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\tilde{\mathbf{X}}^i \text{ if } \left(\tilde{\mathbf{X}}^i\mathbf{w} - y^i\right) \geq 0$$

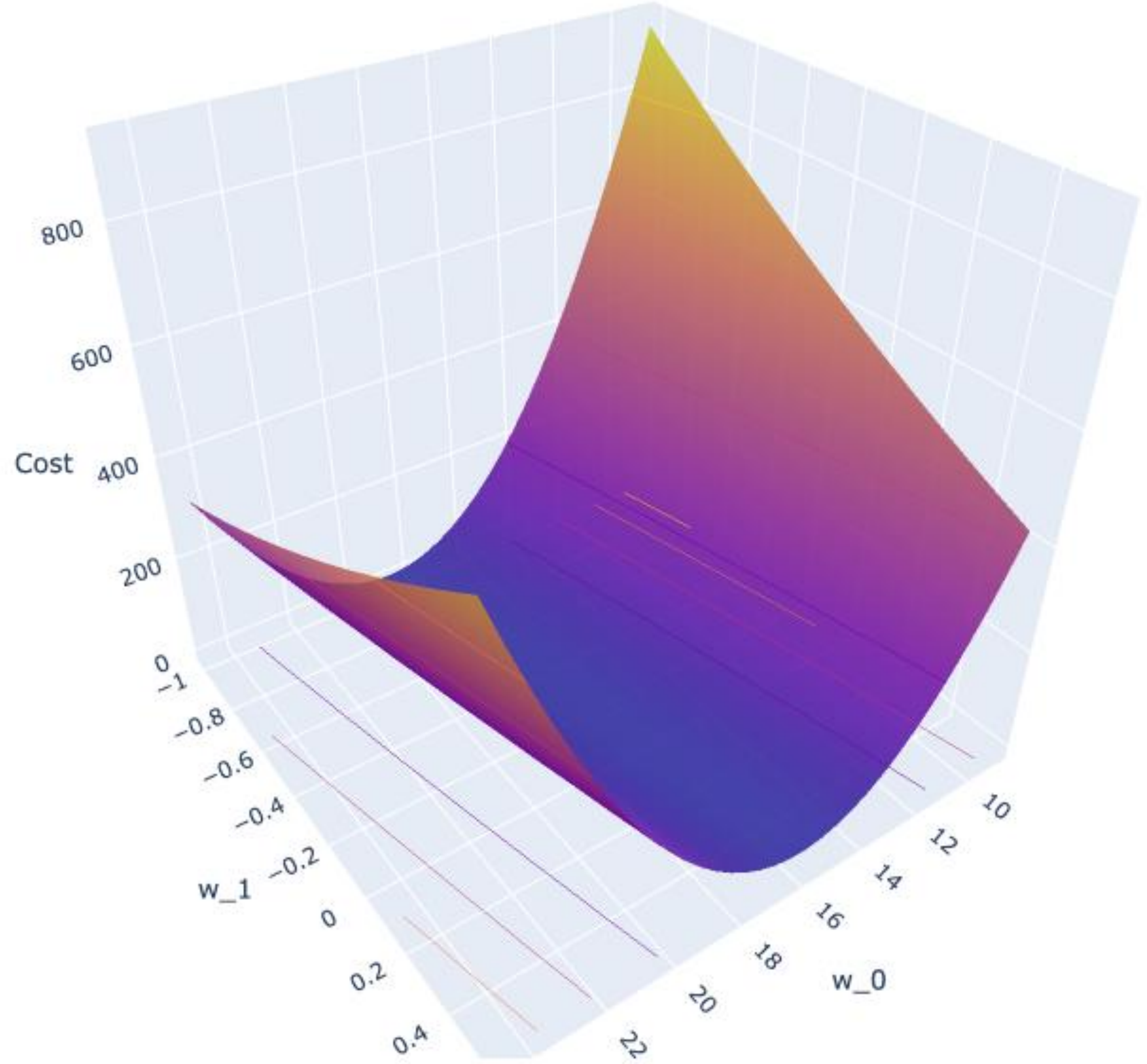
$$= -\frac{1}{N}\sum_{i=1}^{N}\tilde{\mathbf{X}}^i \text{ otherwise.}$$

Standardizing / rescaling input variables

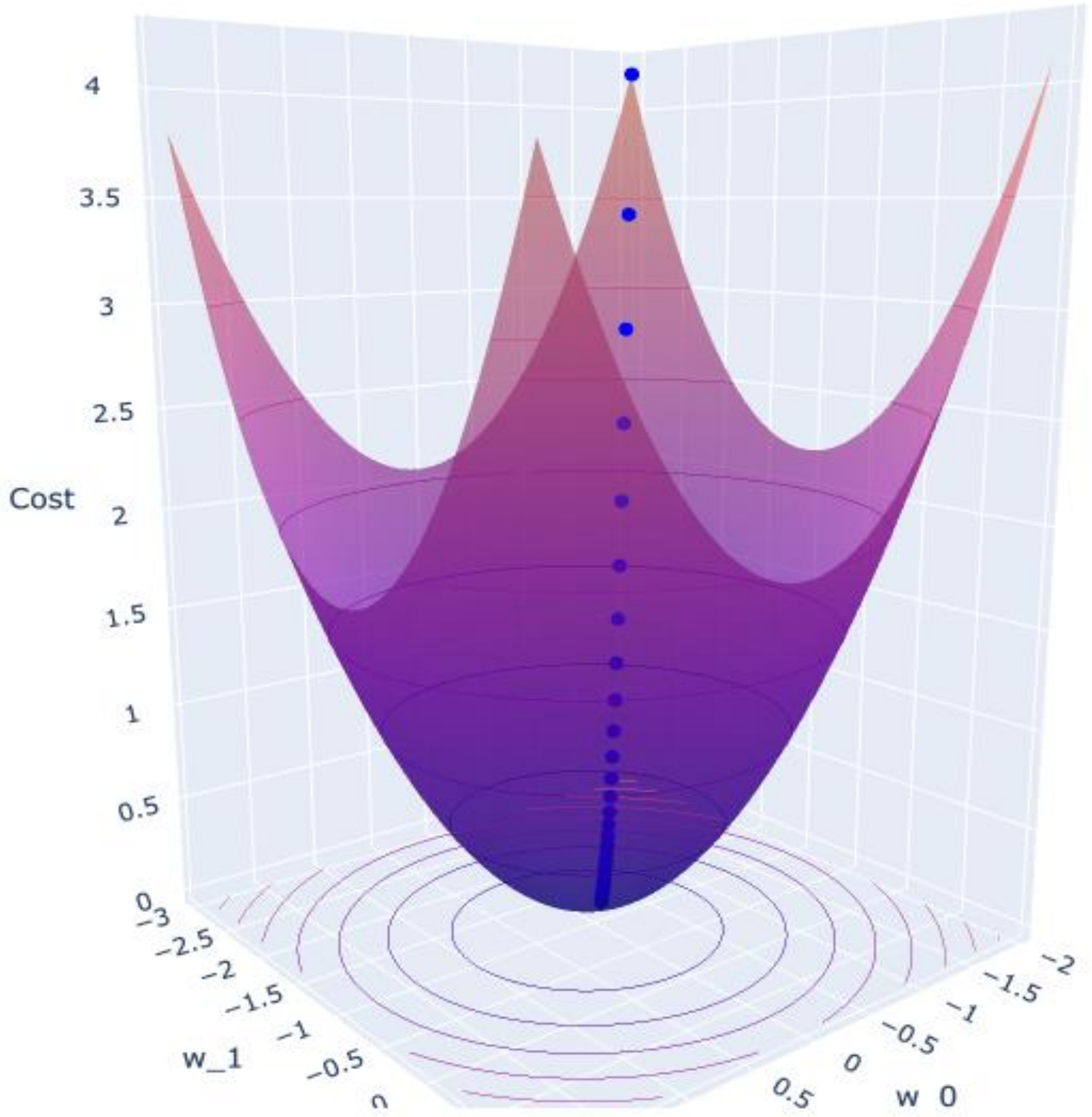
- Rescaling all (in)dependent variables help with convergence
- A common method is to scale the data to have 0 mean and unit variance
- So, $x = \dfrac{(x - \mu_x)}{\sigma_x}$

- Example:
  - Biking / heart disease
  - Mean squared cost

without standardization

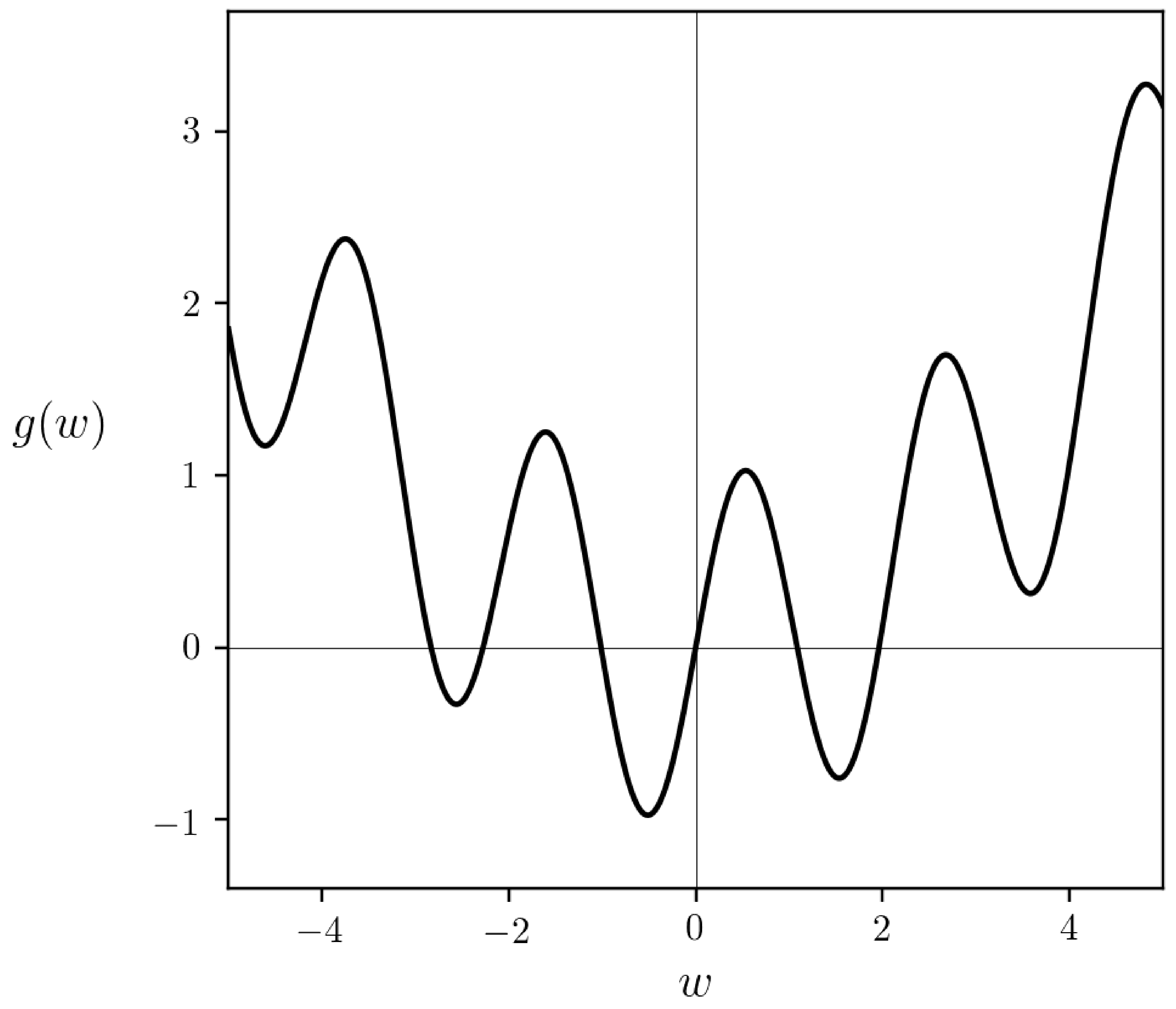


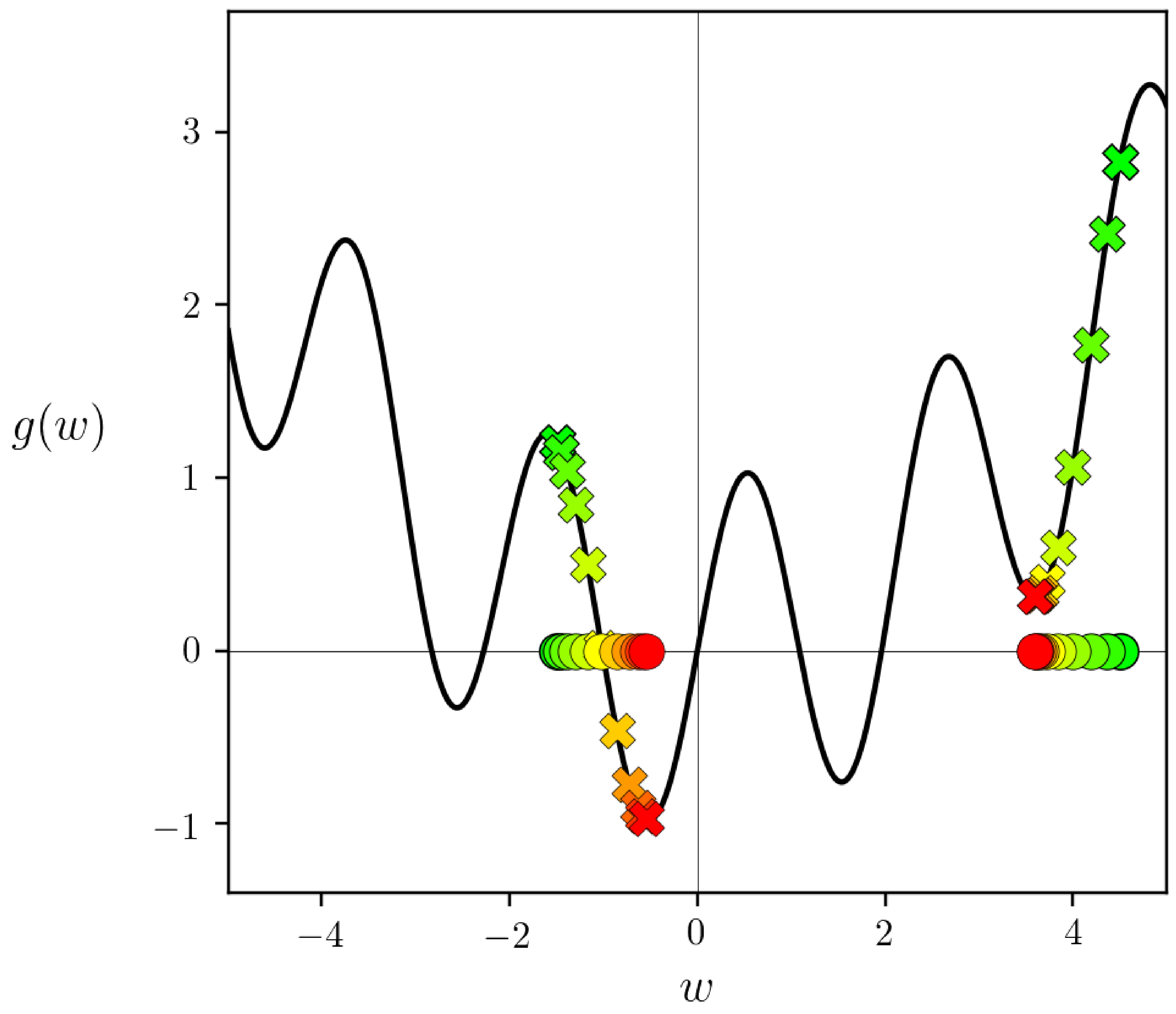
without standardization



Non-convex cost functions

In theory for the most general non-convex functions, in order to find the global minimum of a function using gradient descent one may need to run it several times with different initializations and/or steplengths





What did I learn?

- **Gradient descent** is an iterative optimization algorithm
- Used to **approximate** the local minima of a differentiable (cost) function
- Single local minimum of a **convex (cost) function**; i.e., global minimum
- Two common **convex** cost functions: (mean) squared and (mean) absolute
- scikit-learn proposes **SGDregressor()** – the gradient of the cost (loss) is estimated each sample at a time and the model is updated along the way with a decreasing steplength (aka learning rate) – cf. notebook

QUESTIONS ?