# EECS 395/495 Machine Learning


## Aggelos K. Katsaggelos

Joseph Cummings Professor
Northwestern University
Department of EECS
Department of Linguistics
Argonne National Laboratory
NorthShore University Health System
Evanston, IL 60208
http://ivpl.eecs.northwestern.edu


NORTHWESTERN UNIVERSITY · QUAECUMQUE SUNT VERA · 1851

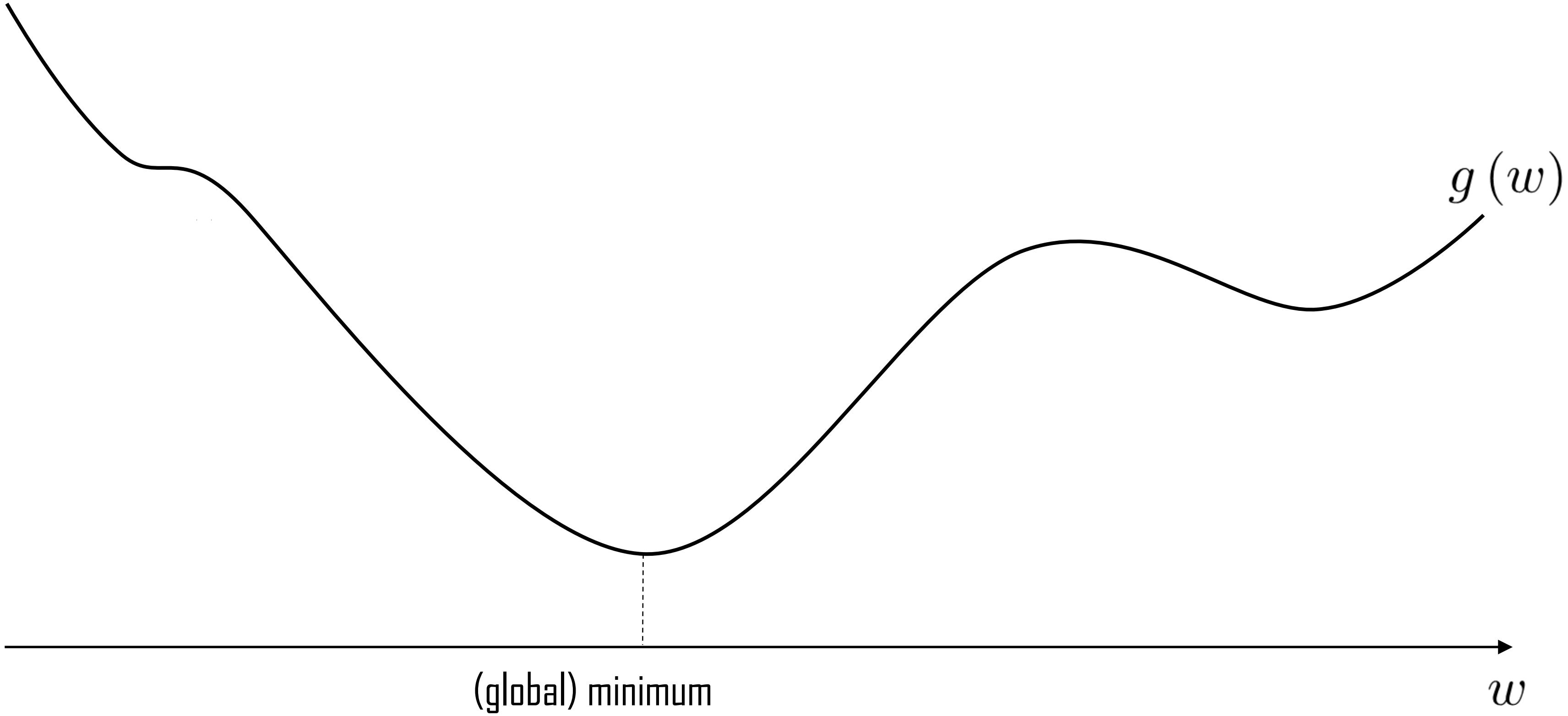
iVPL image & video processing lab

# very big picuture view on numerical optimization

# Why learn numerical optimization?

- In virtually all machine learning applications we look to find the (global) minimum of an associated cost function

- This (global) minimum corresponds to the *optimal* parameters/weights for the model at hand

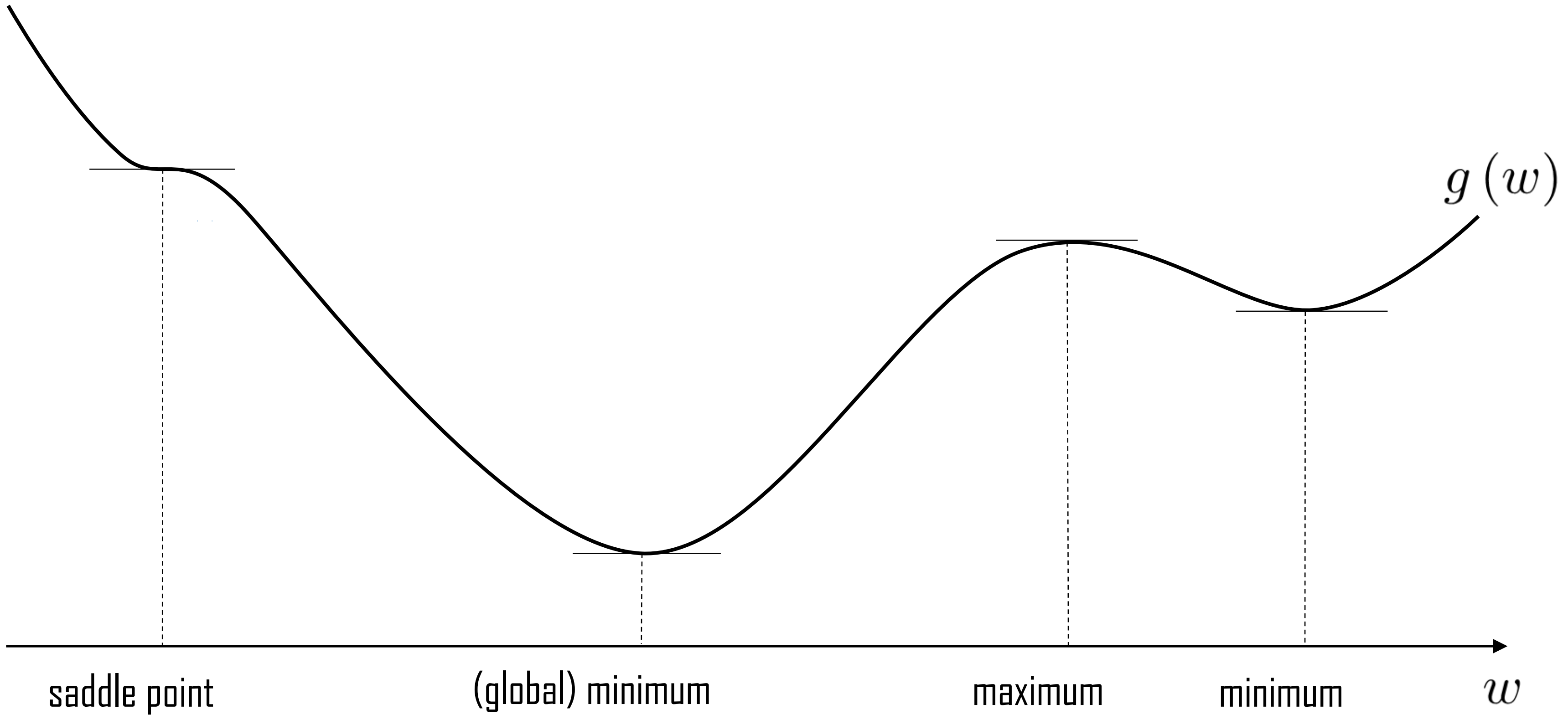


# A useful tool from Calculus

- **The first order condition for optimality** gives a nice criterion for finding all *stationary* points:

$w$ is a stationary point of $g$ if $g'(w) = 0$

For a general *N*-d input $\mathbf{w}_{N\times 1}$ we have the analogous condition $\nabla g(\mathbf{w}) = \mathbf{0}_{N\times 1}$



# Stationary points

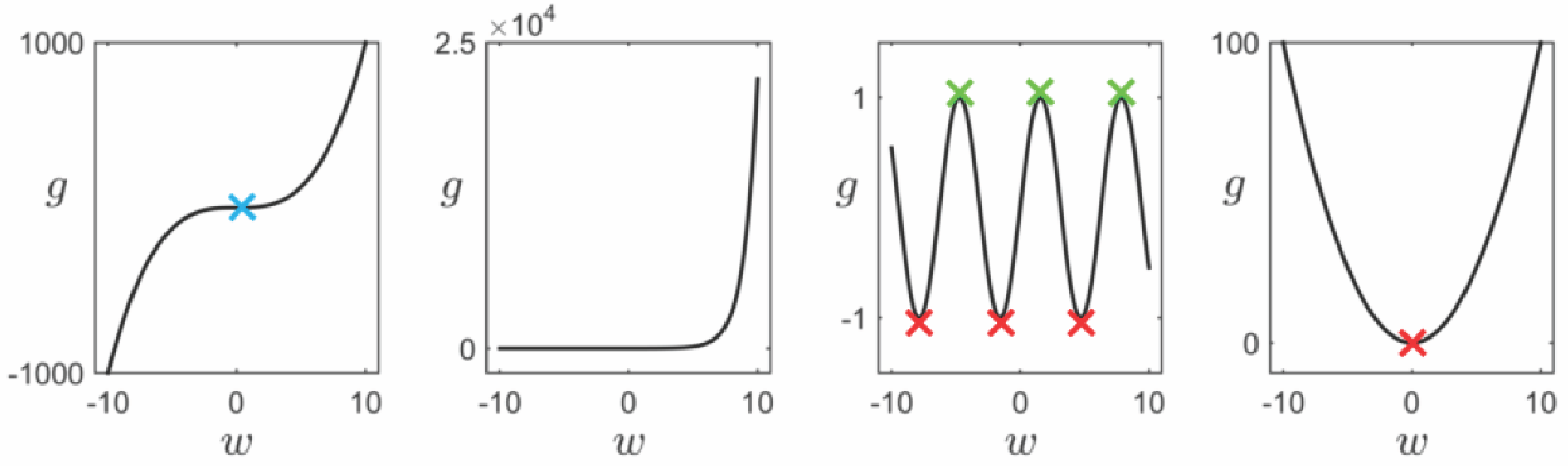



**Figure 2.3.** *From left to right, plots of four functions* $g(w) = w^3$, $e^w$, $sin(w)$, *and* $w^2$, *with their stationary points marked with red, green, and blue cross symbols corresponding to minima, maxima, and saddle points, respectively.*

# Convexity

A twice differentiable function $g$ is convex if and only if it has nonnegative curvature, i.e., $\nabla^2 g(\mathbf{w}) \succeq \mathbf{0}_{N\times N}$ for every $\mathbf{w}$ in its domain.

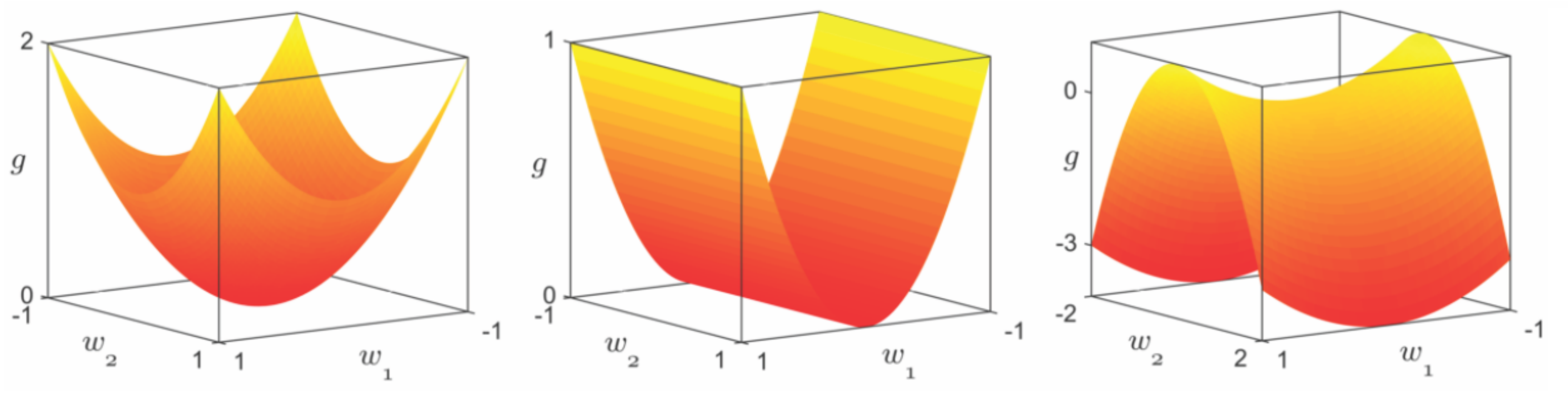



**Figure 2.5.** *Three quadratic cost function* $g(\mathbf{w}) = \mathbf{w}^T\mathbf{A}\mathbf{w} + \mathbf{b}^T\mathbf{w} + c$ *generated by different instances of matrix* $\mathbf{A}$*. In all three cases* $\mathbf{b}$ *and* $c$ *are set to zero. (left)* $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ *generates a (strictly) convex upward facing cup. (middle)* $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ *generates a long narrow half-pipe with infinitely many global minima along the bottom. (right)* $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ *produces the upward and downward curving quadratic surface with a saddle point at* $\mathbf{w} = \mathbf{0}$.

$$\nabla^2 g(\mathbf{w}) = \mathbf{A} + \mathbf{A}^T$$

## Solving the first order system

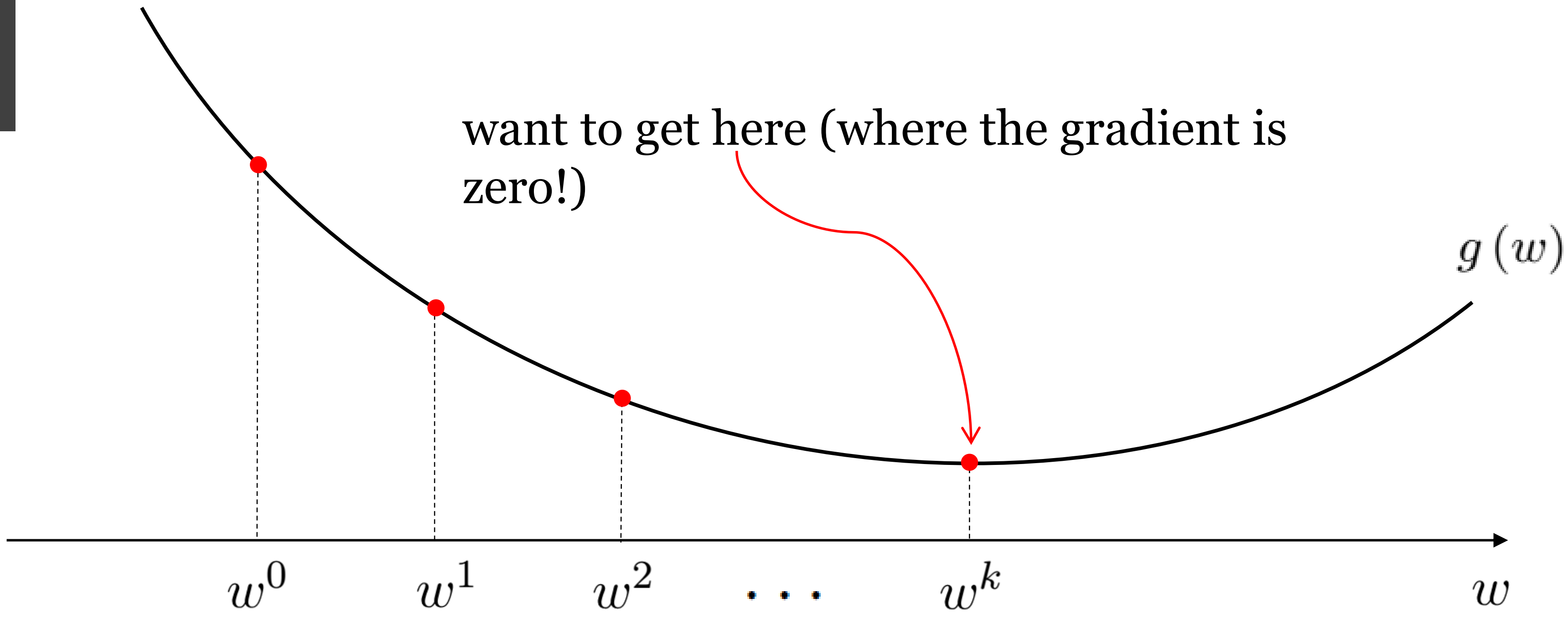
- $\nabla g(\mathbf{w}) = \mathbf{0}_{N\times 1}$ is a system of $N$ equations
$$\begin{cases} \frac{\partial}{\partial w_1} g = 0 \\ \frac{\partial}{\partial w_2} g = 0 \\ \vdots \\ \frac{\partial}{\partial w_N} g = 0 \end{cases}$$

- Easy to solve when linear (e.g., linear regression)

- But in most cases nonlinear in $\mathbf{w}$ with no closed form solution

- Iterative methods (e.g., gradient descent or Newton's method) are used to *approximately* solve this system

# Iterative methods

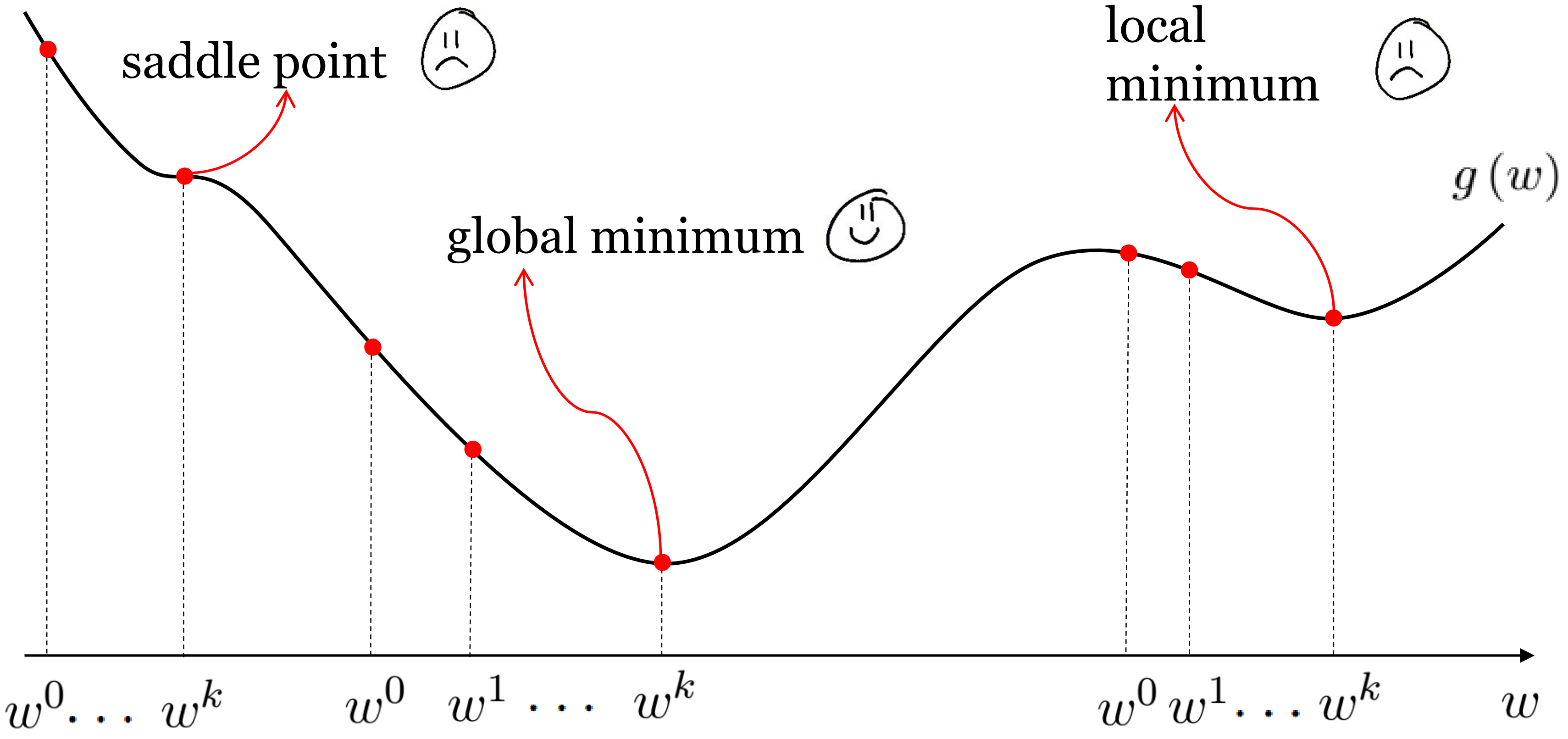



(1) Start the minimization process from some *initial point* $\mathbf{w}^0$.

(2) Take *iterative* steps denoted by $\mathbf{w}^1$, $\mathbf{w}^2$, $\ldots$, going "downhill" towards a stationary point of $g$.

(3) Repeat step (2) until the sequence of points converges to a stationary point of $g$.

# Initial point




1. Start the minimization process from some *initial point* $\mathbf{w}^0$.

2. Take *iterative* steps denoted by $\mathbf{w}^1, \mathbf{w}^2, \ldots$, going "downhill" towards a stationary point of $g$.

3. Repeat step 2 until the sequence of points converges to a stationary point of $g$.

# Initial point

- With non-convex cost functions it is possible to end up at a saddle point or a local minimum depending on the initialization
- In such cases run the iterative method multiple times with different initializations and take the lowest result
- nonconvex $\neq$ bad, simply worth being aware of

1 Start the minimization process from some *initial point* $\mathbf{w}^0$.

2 Take *iterative* steps denoted by $\mathbf{w}^1$, $\mathbf{w}^2$, $\ldots$, going "downhill" towards a stationary point of $g$.

3 Repeat step 2 until the sequence of points converges to a stationary point of $g$.

# Stopping criteria

1. When a pre-specified number of iterations, say $T$, has been taken, i.e., $k > T$

2. When the gradient is small enough, i.e., $\|\nabla g\left(\mathbf{w}^k\right)\|_2 < \epsilon$ for some small $\epsilon > 0$

3. When the objective $g\left(\mathbf{w}^k\right)$ does not change much from step to step, i.e., $\frac{\left|g\left(\mathbf{w}^k\right)-g\left(\mathbf{w}^{k-1}\right)\right|}{\left|g\left(\mathbf{w}^k\right)\right|} < \epsilon$ for some small $\epsilon > 0$

4. When the solution $\mathbf{w}^k$ does not change much from step to step, i.e., $\frac{\|\mathbf{w}^k-\mathbf{w}^{k-1}\|_2}{\|\mathbf{w}^k\|_2} < \epsilon$ for some small $\epsilon > 0$

# The iterative steps

- The iterative steps are taken based on the *Taylor series approximation* of the cost function
- Two popular methods: **gradient descent** (based on $1^{st}$ order Taylor approximation), and **Newton's method** (based on $2^{nd}$ order Taylor approximation)

(1) Start the minimization process from some *initial point* $\mathbf{w}^0$.

(2) Take *iterative* steps denoted by $\mathbf{w}^1$, $\mathbf{w}^2$, $\ldots$, going "downhill" towards a stationary point of $g$.

(3) Repeat step (2) until the sequence of points converges to a stationary point of $g$.

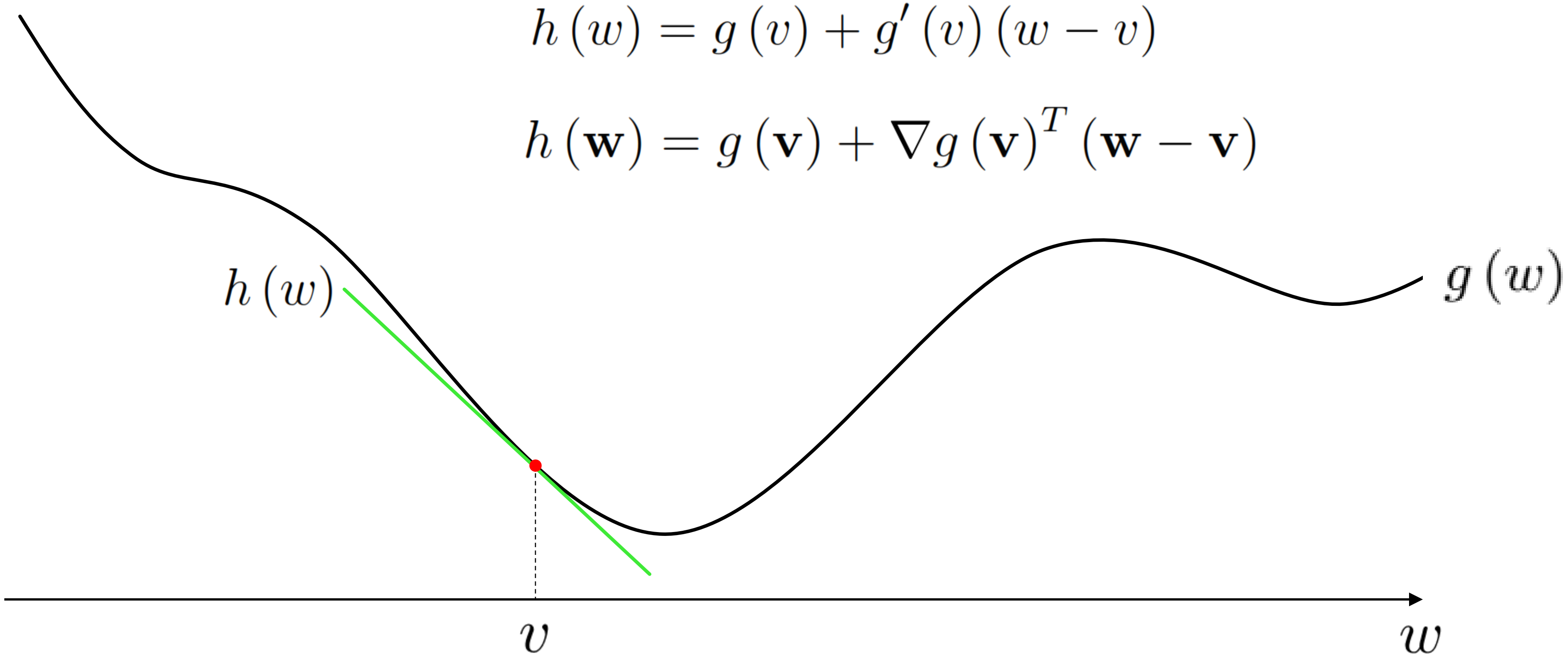
# Gradient Descent

# Gradient descent

- The most basic yet extremely popular numerical optimization method
- Iterative steps are taken based on the **1st order** Taylor series approximation
- The minimum is sought by traveling downward on the **hyperplanes** tangent to $g$ at each iteration

$$h(w) = g(v) + g'(v)(w - v)$$

$$h(\mathbf{w}) = g(\mathbf{v}) + \nabla g(\mathbf{v})^T (\mathbf{w} - \mathbf{v})$$



# Gradient descent

- The idea is to minimize $h$ instead of $g$
- But $h$ is a hyperplane with its minimum at -∞ !
- So we take a finite-length step in the *steepest descent* direction

$$\underset{\|\mathbf{w}\|_2=1}{\text{minimize}} \quad \|\nabla g(\mathbf{v})\|_2 \, \|\mathbf{w}\|_2 \cos(\theta)$$

$$\theta = \pi$$

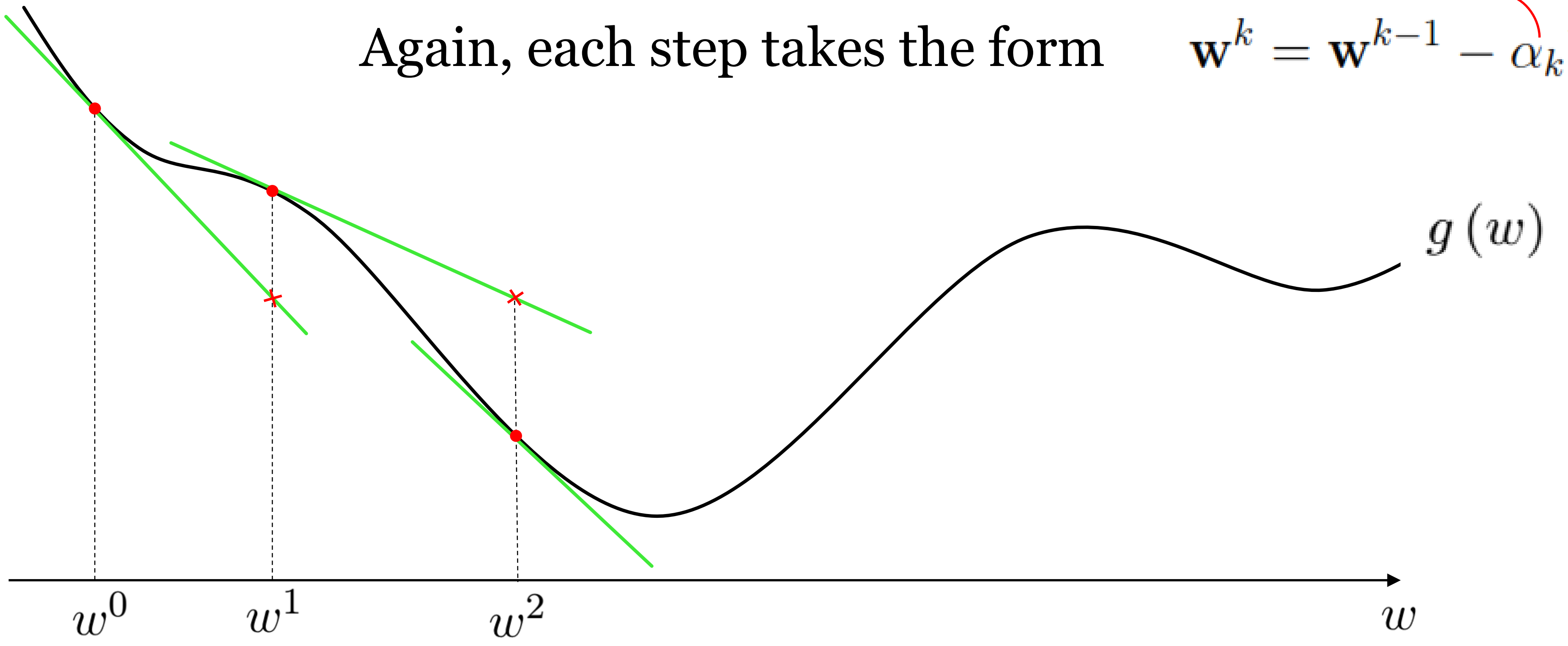
$$\mathbf{w} = \frac{-\nabla g(\mathbf{v})}{\|\nabla g(\mathbf{v})\|_2}$$

# Gradient descent

- Begin at a point $\mathbf{w}^o$
- Travel downward on tangent hyperplane to $g$ at $\mathbf{w}^o$ in the direction of negative gradient
- (Hop back onto the function)
- Repeat until a stationary point is reached

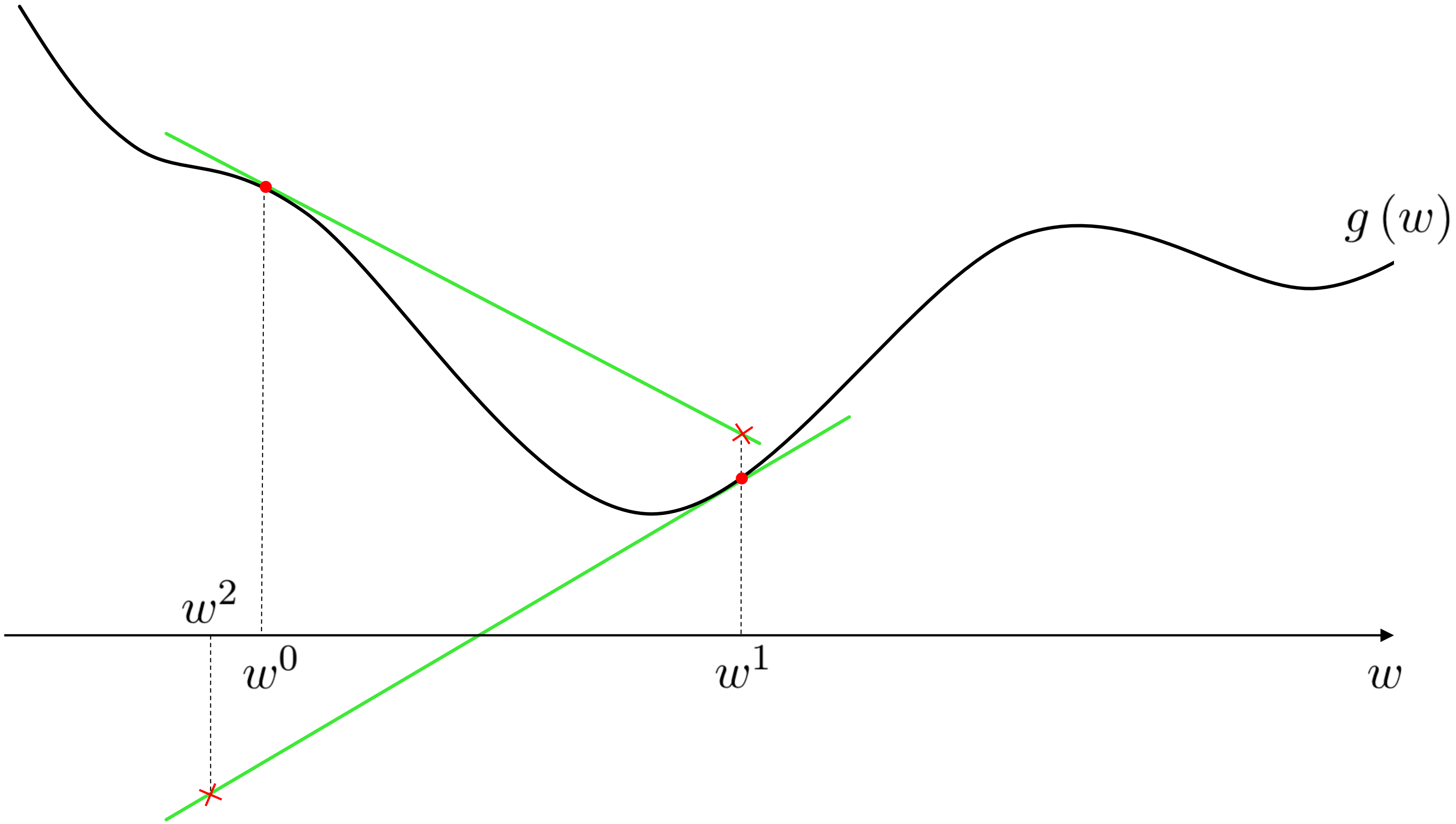
step length (aka learning rate) determines how far to travel down the negative gradient direction

Again, each step takes the form $\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha_k \nabla g\left(\mathbf{w}^{k-1}\right)$



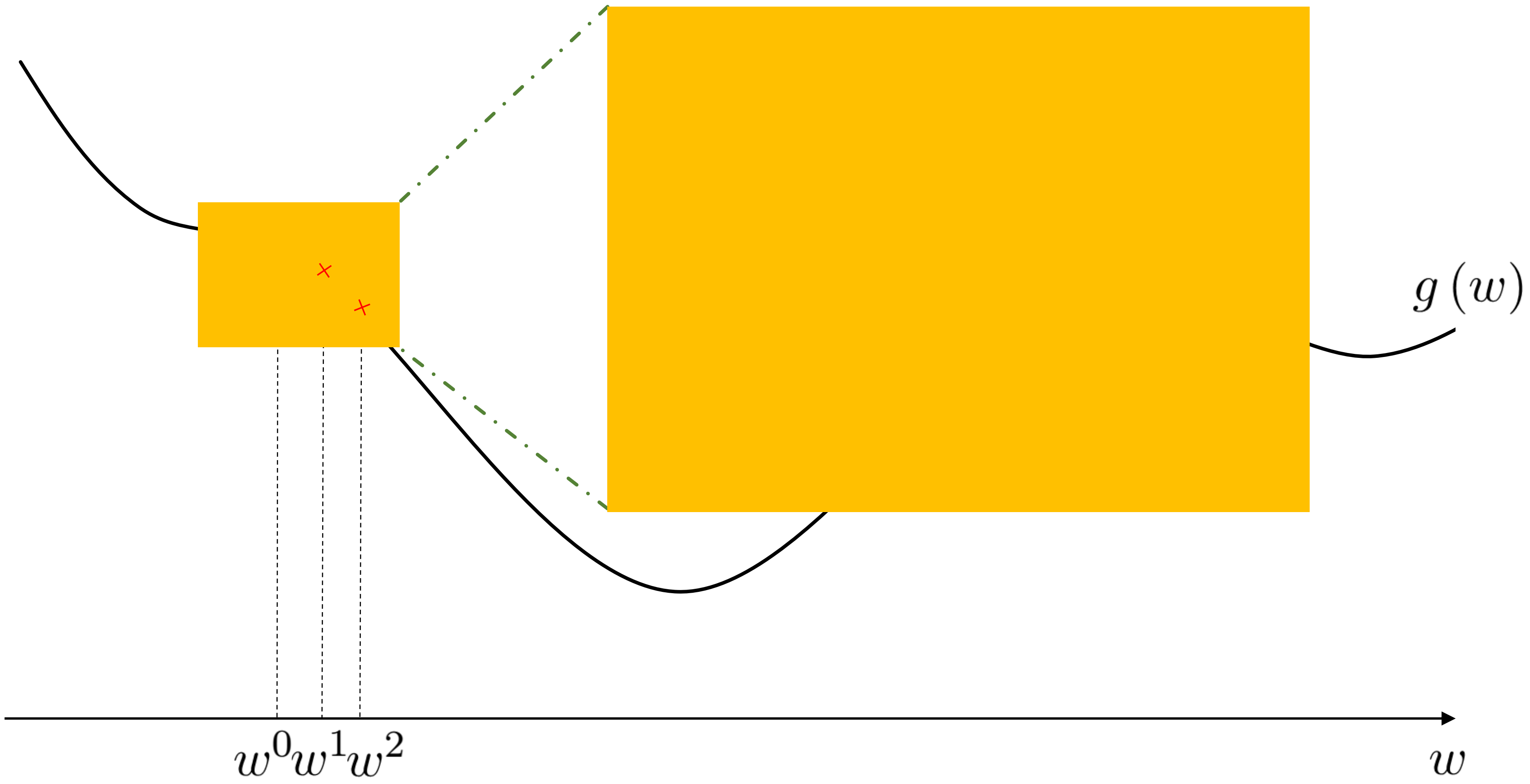
# Gradient descent step length

- What happens when the step length is too large?



# Gradient descent step length

- What happens when the step length is too small?



## Gradient descent step length

- How to select a "good" step length?

1. **Naïve trial and error**: Start with a fixed large step size for all iterations, if not produce decreasing values in then $g$ecrease and try again!

2. **Does $g$ have bounded curvature?** If so, a fixed step length can be provable convergence

3. **Adaptive step length selection:** Use an adaptive step length procedure essentially does "trial and error" at each step

# Pseudo-code

**Algorithm** Gradient descent (with fixed step length)

**Input:** differentiable function $g$, fixed step length $\alpha$, and initial point $\mathbf{w}^0$

$k = 1$

**Repeat until stopping condition is met:**

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

$$k \leftarrow k + 1$$

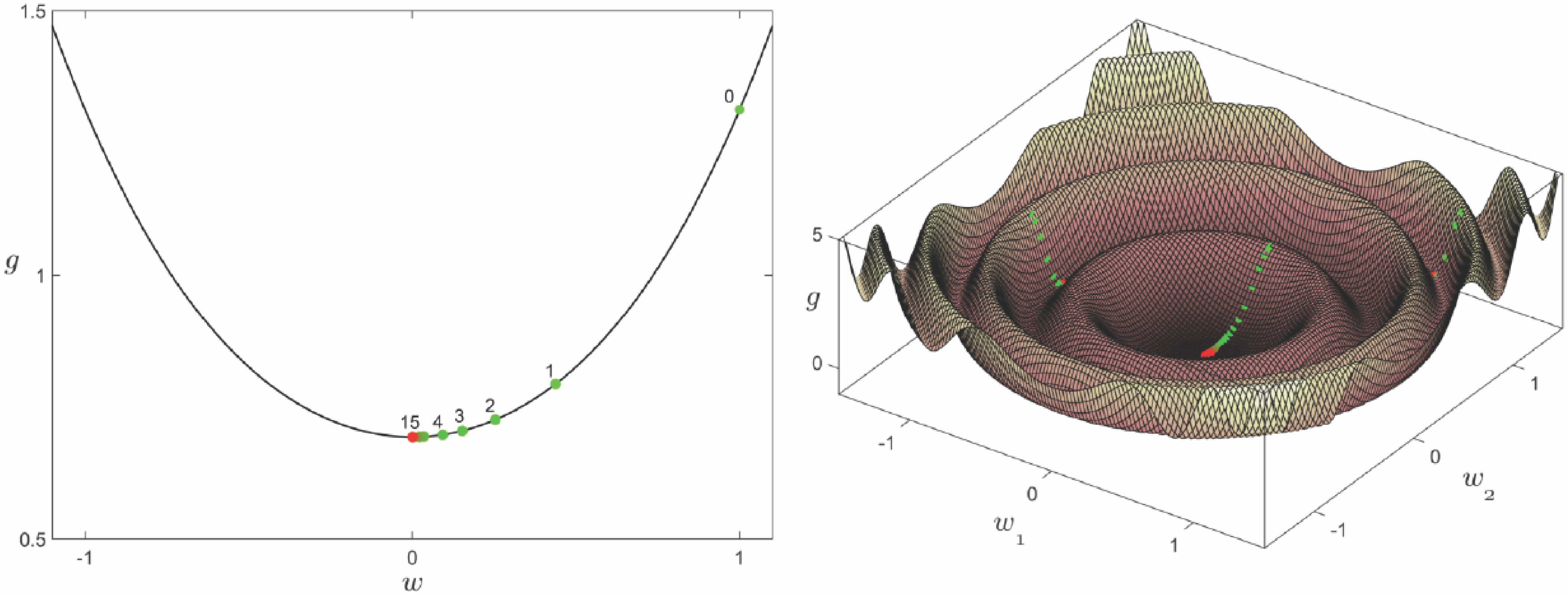



**Figure 2.11.** *Two examples of gradient descent detailed in the text. (left) A convex 1-dimensional function with many gradient descent steps, initialized at* $w^0 = 1$*, required for convergence to the global minimum. Only the first* 15 *steps are numbered since the rest blur together as the minimum is approached. (right) A 2-dimensional nonconvex function with gradient descent steps towards a minimum, initialized at three distinct values leads to three local minima of the function (one of which, the center-most, is the global minimum as well.*

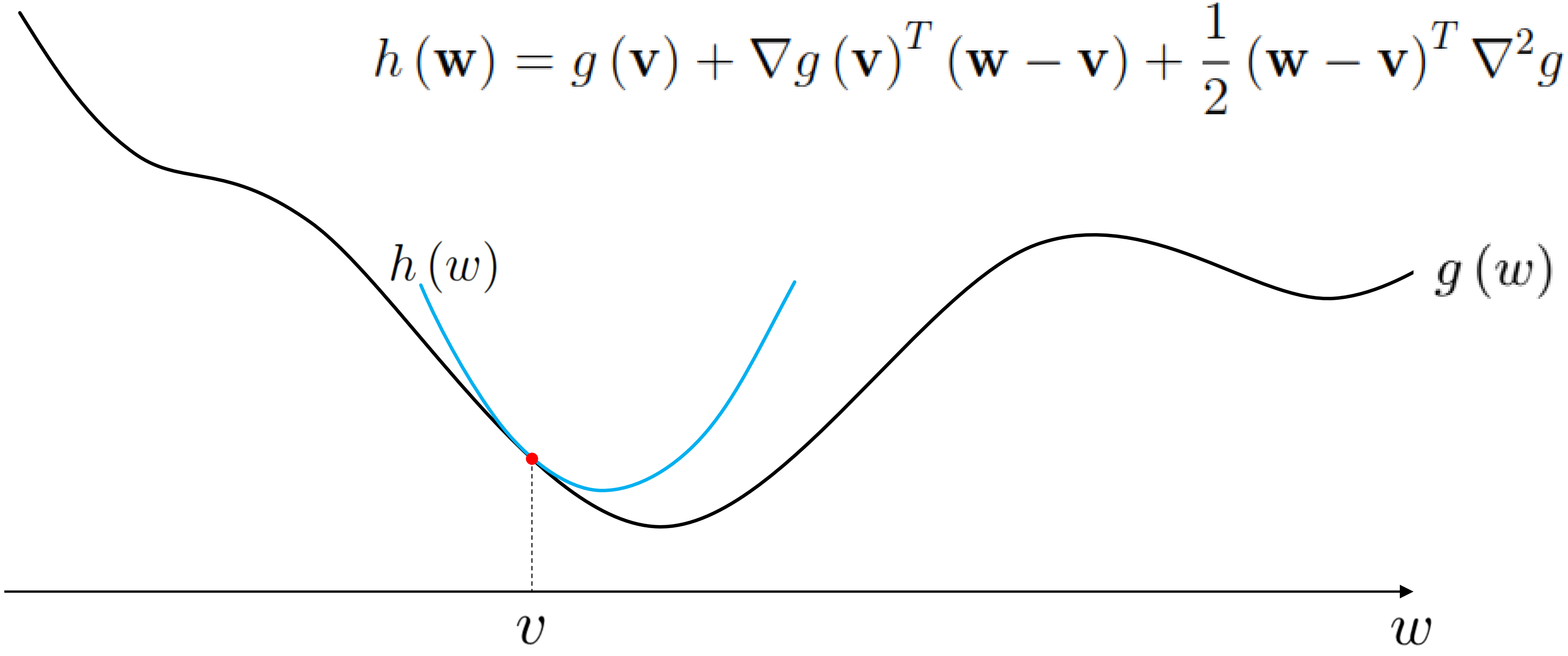
# Newton’s Method

## Newton’s method

- Iterative steps are taken based on the **2nd order** Taylor series approximation
- The minimum is sought by traveling downward on the **quadratics** tangent t $g$ at each iteration

$$h(w) = g(v) + g'(v)(w - v) + \frac{1}{2} g''(v)(w - v)^2$$

$$h(\mathbf{w}) = g(\mathbf{v}) + \nabla g(\mathbf{v})^T (\mathbf{w} - \mathbf{v}) + \frac{1}{2} (\mathbf{w} - \mathbf{v})^T \nabla^2 g(\mathbf{v}) (\mathbf{w} - \mathbf{v})$$



# Newton's method

- Once again the idea is to minimize $h$ at each step
- $h$ is quadratic

$$h(\mathbf{w}) = g(\mathbf{v}) + \nabla g(\mathbf{v})^T (\mathbf{w} - \mathbf{v}) + \frac{1}{2} (\mathbf{w} - \mathbf{v})^T \nabla^2 g(\mathbf{v}) (\mathbf{w} - \mathbf{v})$$

- So we can find its stationary point using the first order condition

$$\nabla h(\mathbf{w}) = \nabla g(\mathbf{v}) + \nabla^2 g(\mathbf{v}) (\mathbf{w} - \mathbf{v})$$

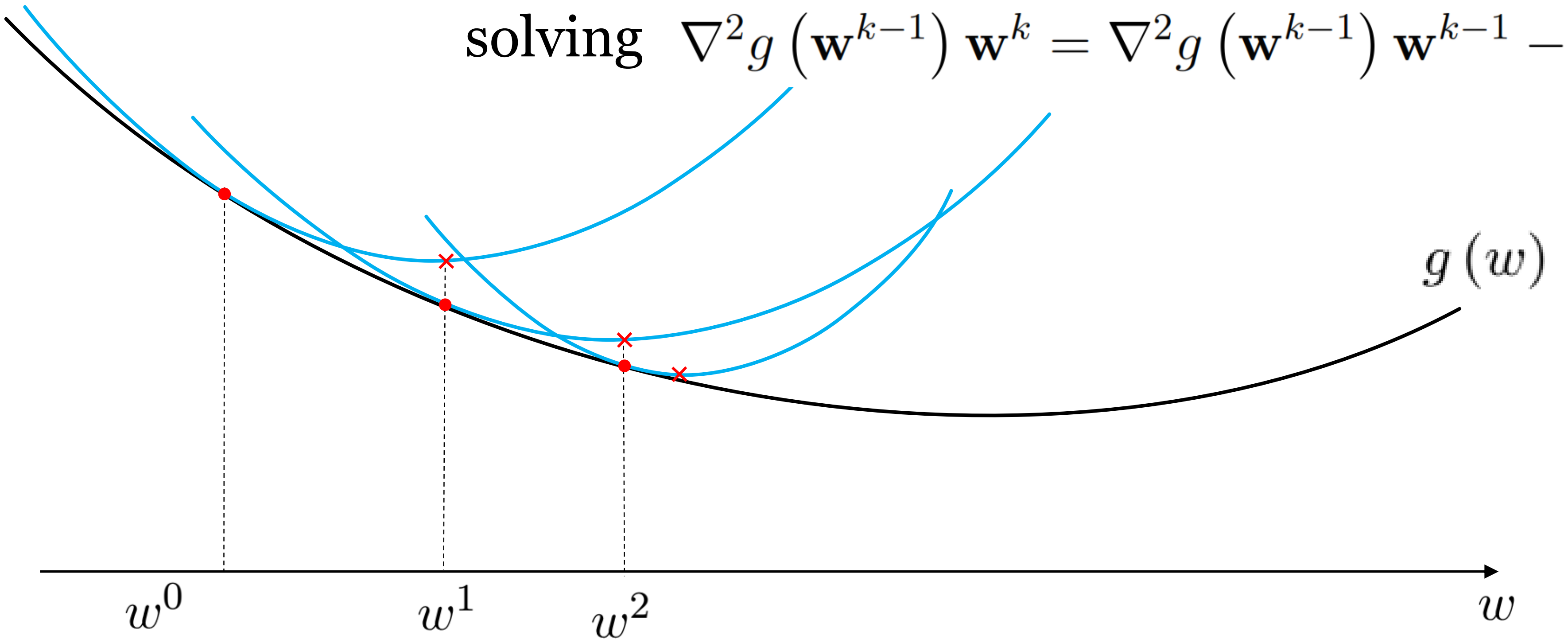
- Setting $\nabla h$ to zero gives the following linear system in $\mathbf{w}$

$$\nabla^2 g(\mathbf{v}) \mathbf{w} = \nabla^2 g(\mathbf{v}) \mathbf{v} - \nabla g(\mathbf{v})$$

# Newton's method

- Begin at a point $\mathbf{w}^0$
- Travel to the minimum of the tangent quadratic
- (Hop back onto the function)
- Repeat until a stationary point is reached

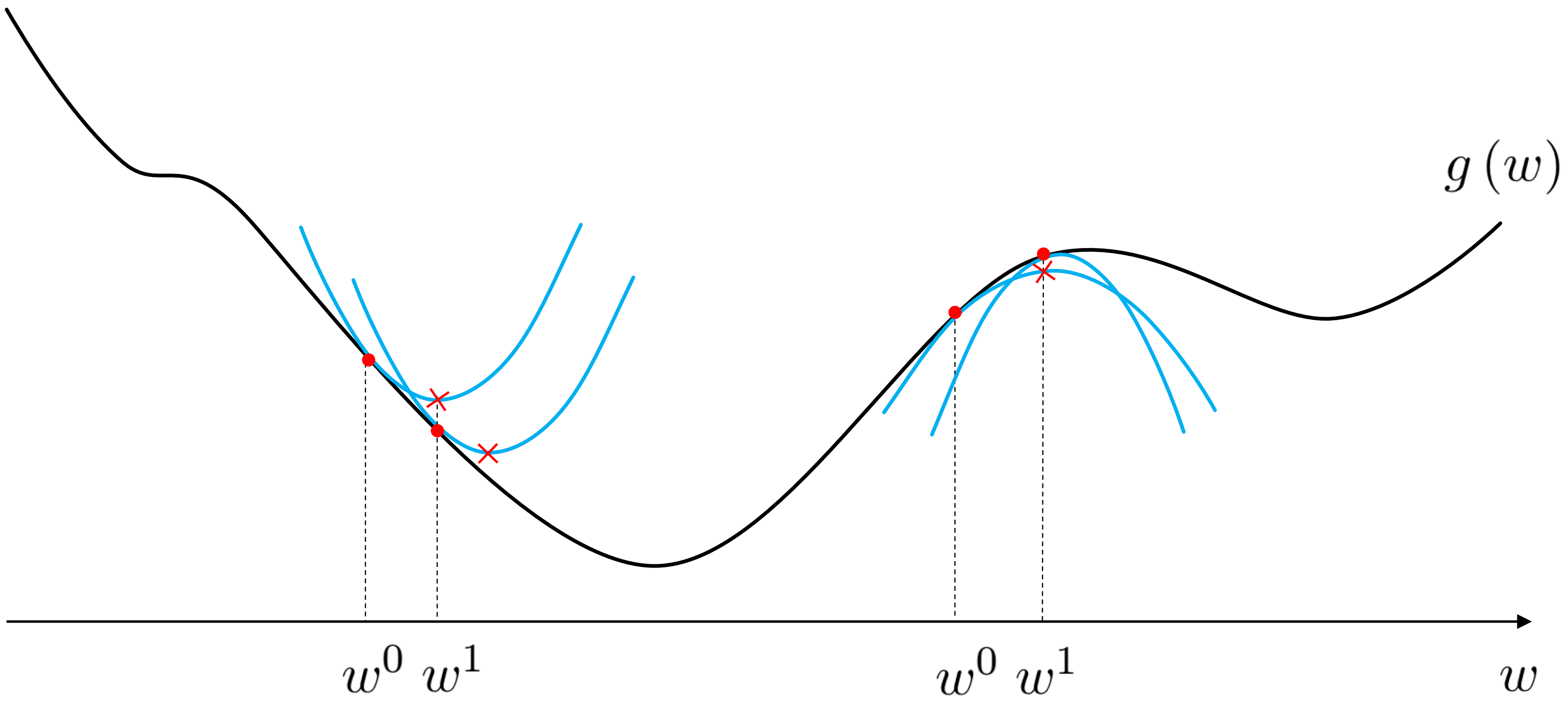
Again, the minimum of each quadratic is found via solving $\nabla^2 g\left(\mathbf{w}^{k-1}\right)\mathbf{w}^k = \nabla^2 g\left(\mathbf{w}^{k-1}\right)\mathbf{w}^{k-1} - \nabla g\left(\mathbf{w}^{k-1}\right)$



# Warning!

- For a non-convex function quadratics can be **concave**

- With such functions Newton's method can *climb* to a maximum, or diverge!



# Pseudo-code

**Algorithm** Newton's method

**Input:** twice differentiable function $g$, and initial point $\mathbf{w}^0$

$k = 1$

**Repeat until stopping condition is met:**

Solve the system $\nabla^2 g\left(\mathbf{w}^{k-1}\right)\mathbf{w}^k = \nabla^2 g\left(\mathbf{w}^{k-1}\right)\mathbf{w}^{k-1} - \nabla g\left(\mathbf{w}^{k-1}\right)$ for $\mathbf{w}^k$.

$k \leftarrow k + 1$

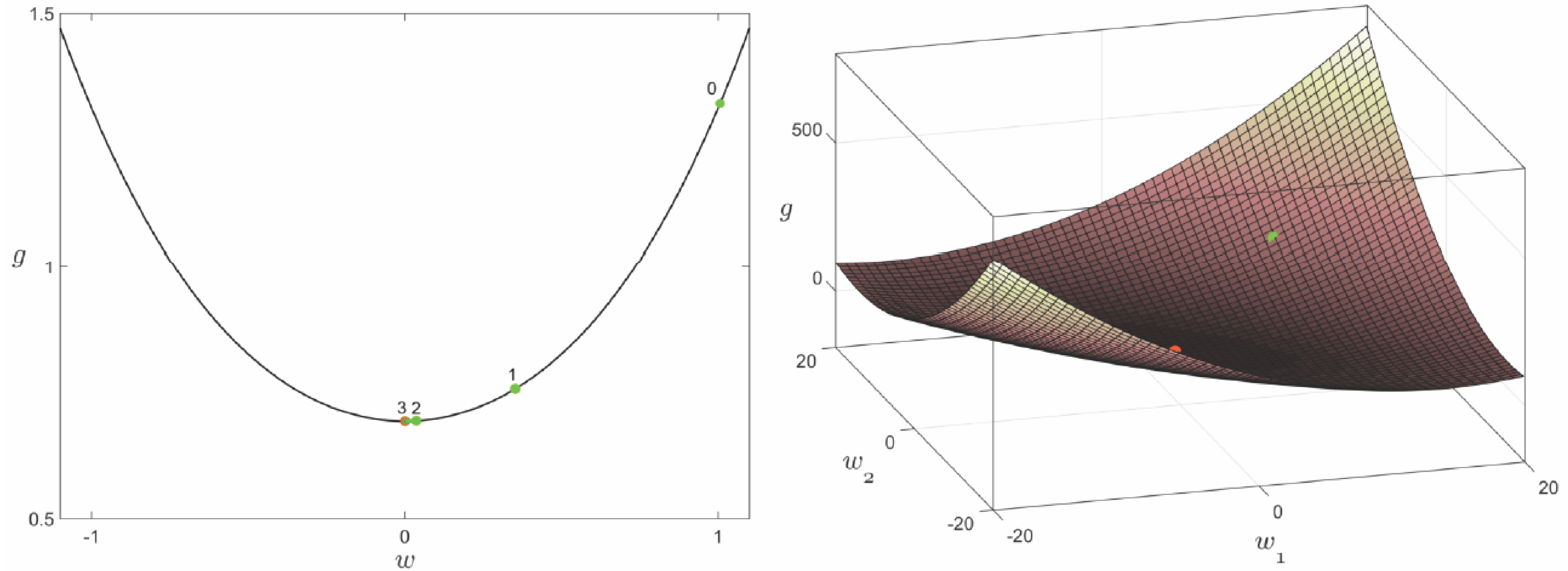



**Figure 2.13.** *Two examples of Newton's method detailed in the text. (left) A convex 1-dimensional function with only three Newton steps, initialized at $w^0 = 1$, required for convergence to the global minimum. (right) A 2-dimensional quadratic function. Initialized at any point, only one Newton step is required to reach the global minimum of this function.*

# Gradient descent vs. Newton's method

- Gradient descent uses 1st order (linear) approximations while Newton's method uses 2nd order (quadratic) approximations

- Newton's method converges in fewer steps as a result

- Unlike Newton's method, gradient descent needs properly chosen step length for convergence

- Newton's method requires calculation/storage of the Hessian (in addition to the gradient) at each iteration, thus not suitable for large scale problems

- Newton's method can be problematic when applied to non-convex functions