CSCE 633: Machine Learning

Lecture 9: Gradient Descent

Texas A&M University
Bobak Mortazavi

Goals for this Lecture

- First Order Optimization – The Gradient Function
- Understand Gradient Descent
- Understanding Limitations to Gradient Descent
- Second Order Optimization – Convexity/Concavity
- Newton’s Method for Descent

Why does all this work? Convexity of loss function!

- A Set S is convex if for any w, $w' \in S$ there exists
- $\lambda \mathrm{w} + (1-\lambda)w' \in S$ for $\lambda \in [0,1]$
- In practice this means draw a line between any two points in a set and if it is convex, every point on the line still lies within the set
- Now, a function g(w) is convex if its set of points defines a convex set
- In other words

$$g(\lambda(\mathrm{w} + (1-\lambda)w') \leq \ \lambda g(\mathrm{w} + (1-\lambda)g(w') \ \ \lambda \in [0,1]$$

The Zero-Order Optimality Condition

Find the smallest point(s) of a function.

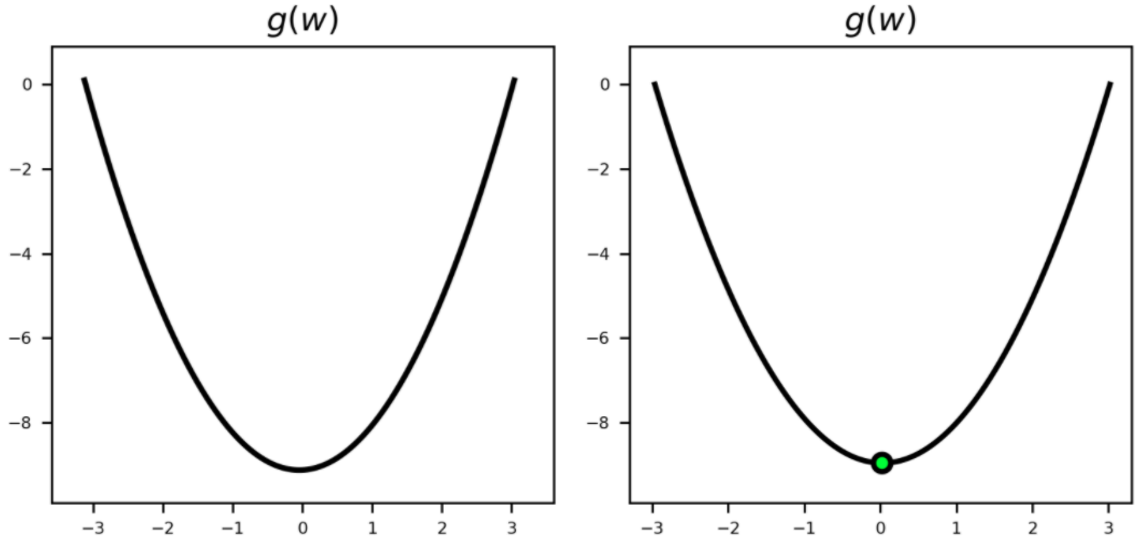
$$\underset{w}{\text{minimize}}\; g(\boldsymbol{w})$$

- Approach:
  - Identify the minimum visually by plotting it over a large swath of its input space.

The Zero-Order Optimality Condition

- Example 1: Global minimum of a quadratic

$$g(w) = w^2$$



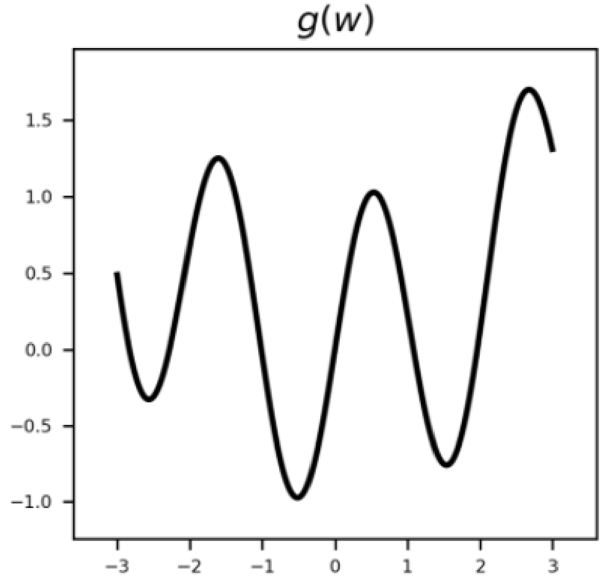
- **Global** minimum point w^*

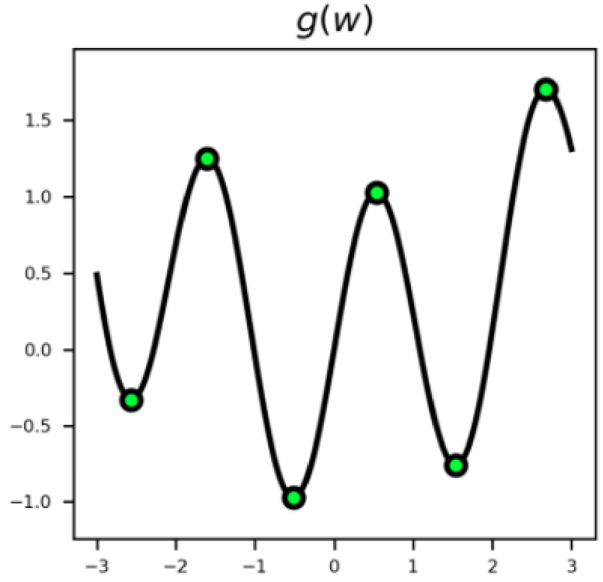
$$g(w^*) \leq g(w) \text{ for all } w$$

The Zero-Order Optimality Condition

- Example 4: **local** maximum/minimum of the sum of a sinusoid and a quadratic

$$g(w) = \sin(3w) + 0.1w^2$$





- **Local** minimum point w^*

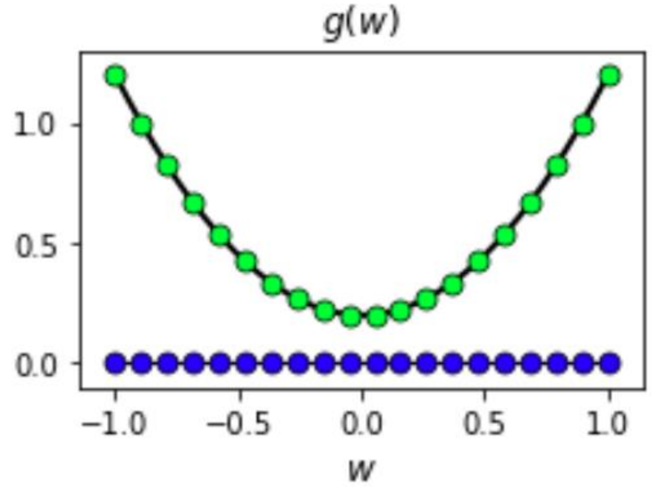
$$g(w^*) \leq g(w) \text{ for all } w \text{ near } w^*$$

The zero order condition for optimality

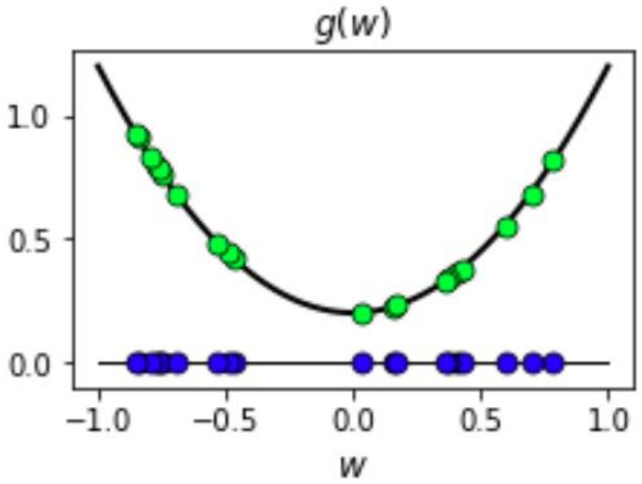
- The zero order condition for optimality: A point w^* is:
  - a global minimum of $g(w)$ if and only if $g(w^*) \leq g(w)$ for all w.
  - a global maximum of $g(w)$ if and only if $g(w^*) \geq g(w)$ for all w.
  - a local minimum of $g(w)$ if and only if $g(w^*) \leq g(w)$ for all $w \text{ near } w^*$.
  - a local maximum of $g(w)$ if and only if $g(w^*) \geq g(w)$ for all $w \text{ near } w^*$.

Global Optimization Methods

- More samples



Evenly sampling



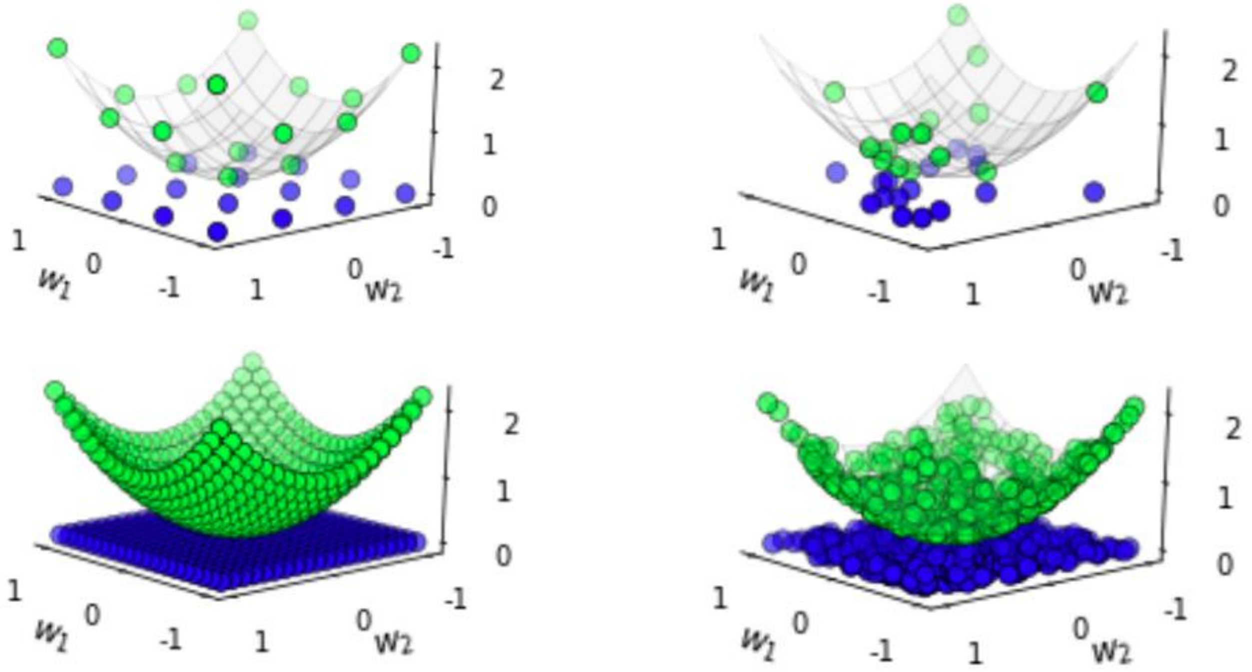
Randomly sampling

- When given enough samples, the minimized point can be close to global minimum.
- Either approach is able to find global minimum.

Global Optimization Methods

Example 2: 3-d quadratic

$$g(w_1, w_2) = w_1^2 + w_2^2 + 0.2$$



Evenly sampling

Randomly sampling

Local Optimization Methods

Framework

- $\mathbf{w}^0$: initial point.
- $\mathbf{w}^1$: the first updated point
- $\mathbf{d}^0$: direction vector from $\mathbf{w}^0$ to $\mathbf{w}^1$

$$\mathbf{w}^1 = \mathbf{w}^0 + \mathbf{d}^0$$

- Similarly
- $\mathbf{w}^2$: the second updated point
- $\mathbf{d}^1$: direction vector from $\mathbf{w}^1$ to $\mathbf{w}^2$

$$\mathbf{w}^2 = \mathbf{w}^1 + \mathbf{d}^1$$

Local Optimization Methods

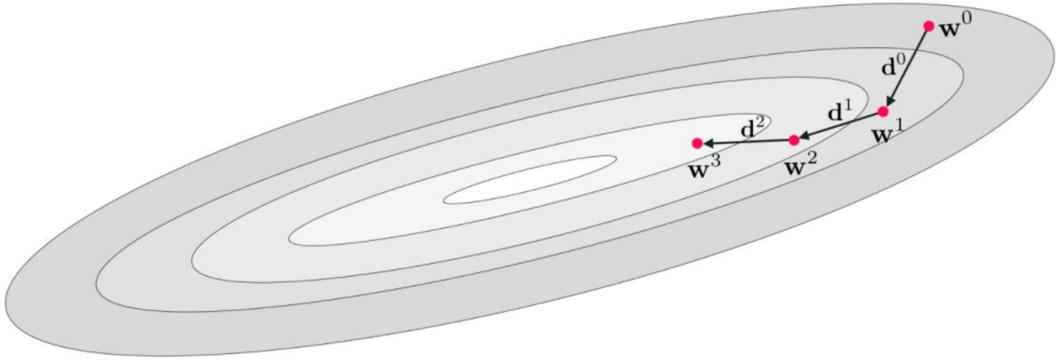
$$
\begin{aligned}
&\mathbf{w}^0 \\
&\mathbf{w}^1 = \mathbf{w}^0 + \mathbf{d}^0 \\
&\mathbf{w}^2 = \mathbf{w}^1 + \mathbf{d}^1 \\
&\mathbf{w}^3 = \mathbf{w}^2 + \mathbf{d}^2 \\
&\vdots \quad \vdots \quad \vdots \\
&\mathbf{w}^K = \mathbf{w}^{K-1} + \mathbf{d}^{K-1}
\end{aligned}
$$

$\mathbf{d}^{k-1}$ is the descent direction defined at the k^{th} step of process

$$\mathbf{w}^k = \mathbf{w}^{k-1} + \mathbf{d}^{k-1}$$

and

$$g(\mathbf{w}^0) > g(\mathbf{w}^1) > g(\mathbf{w}^2) > \cdots > g(\mathbf{w}^K)$$



Schematic illustration of a generic local optimization scheme.

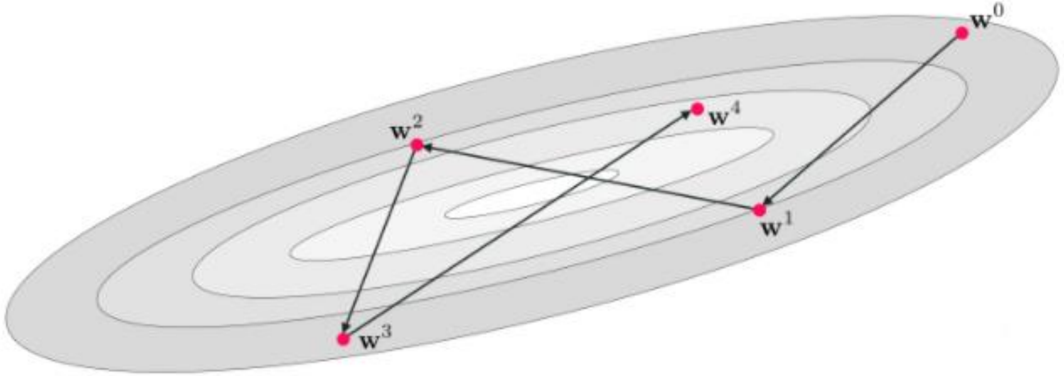
Local Optimization Methods

The steplength parameter

- Distance of updating at k^{th} step:

$$\left\|\mathbf{w}^k - \mathbf{w}^{k-1}\right\|_2 = \left\| \left(\mathbf{w}^{k-1} + \mathbf{d}^{k-1}\right) - \mathbf{w}^{k-1}\right\|_2 = \left\|\mathbf{d}^{k-1}\right\|_2.$$

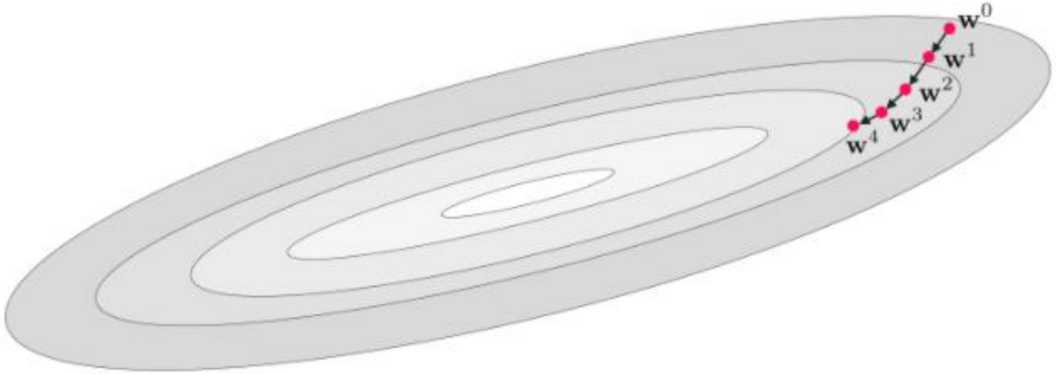
- Correct direction wrong length
  - Large direction vectors: can never reach approximate minimum.



Direction vectors are too large causing a wild oscillatory behavior around the minimum.

Local Optimization Methods

- Correct direction wrong length
  - Short updating distance: move too slow and too many steps are required.



Direction vectors are too small, requiring a large number of steps be taken to reach the minimum.

Local Optimization Methods

- Steplength parameter/Learning rate parameter:

$$\mathbf{w}^k = \mathbf{w}^{k-1} + \alpha \mathbf{d}^{k-1}$$

- The entire sequence of K steps:

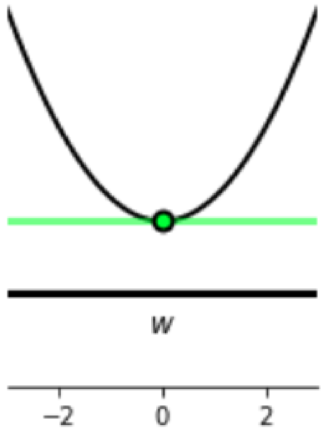
$$\begin{aligned}
&\mathbf{w}^0 \\
&\mathbf{w}^1 = \mathbf{w}^0 + \alpha \mathbf{d}^0 \\
&\mathbf{w}^2 = \mathbf{w}^1 + \alpha \mathbf{d}^1 \\
&\mathbf{w}^3 = \mathbf{w}^2 + \alpha \mathbf{d}^2 \\
&\vdots \quad \vdots \quad \vdots \\
&\mathbf{w}^K = \mathbf{w}^{K-1} + \alpha \mathbf{d}^{K-1}
\end{aligned}$$

- Distance vector:

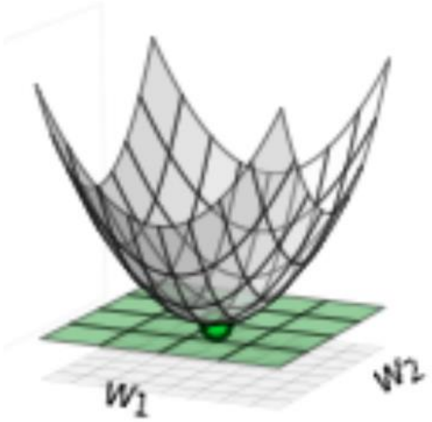
$$\left\| \mathbf{w}^k - \mathbf{w}^{k-1} \right\|_2 = \left\| \left(\mathbf{w}^{k-1} + \alpha \mathbf{d}^{k-1} \right) - \mathbf{w}^{k-1} \right\|_2 = \alpha \left\| \mathbf{d}^{k-1} \right\|_2$$

The First-Order Optimality Condition

The first order condition



2-D quadratic:
tangent line is flat



3-D quadratic:
tangent hyperplane is flat

- The first derivative(s) is exactly zero at the function's minimum.
  - Minimum values of a function are naturally located at 'valley floors'.

The First-Order Optimality Condition

- Potential minimum points v from first order derivatives
  - Input dimension $N = 1$:

$$\frac{\mathrm{d}}{\mathrm{d}w} g\left(v\right) = 0$$

  - Input dimension N:

$$\begin{aligned} \frac{\partial}{\partial w_1} g(\mathbf{v}) &= 0 \\ \frac{\partial}{\partial w_2} g(\mathbf{v}) &= 0 \\ &\vdots \\ \frac{\partial}{\partial w_N} g(\mathbf{v}) &= 0 \end{aligned}$$

- First order system:

$$\nabla g\left(\mathbf{v}\right) = \mathbf{0}_{N \times 1}$$

The First-Order Optimality Condition

- The first order optimality condition translates the problem of identifying a function's minimum points into the task of solving a system of N first order equations.

- Problems:
  - With few exceptions, it is virtually impossible to solve a general function's first order systems of equations 'by hand'.
  - The *first order optimality condition* does not only define minima of a function, but other points as well.

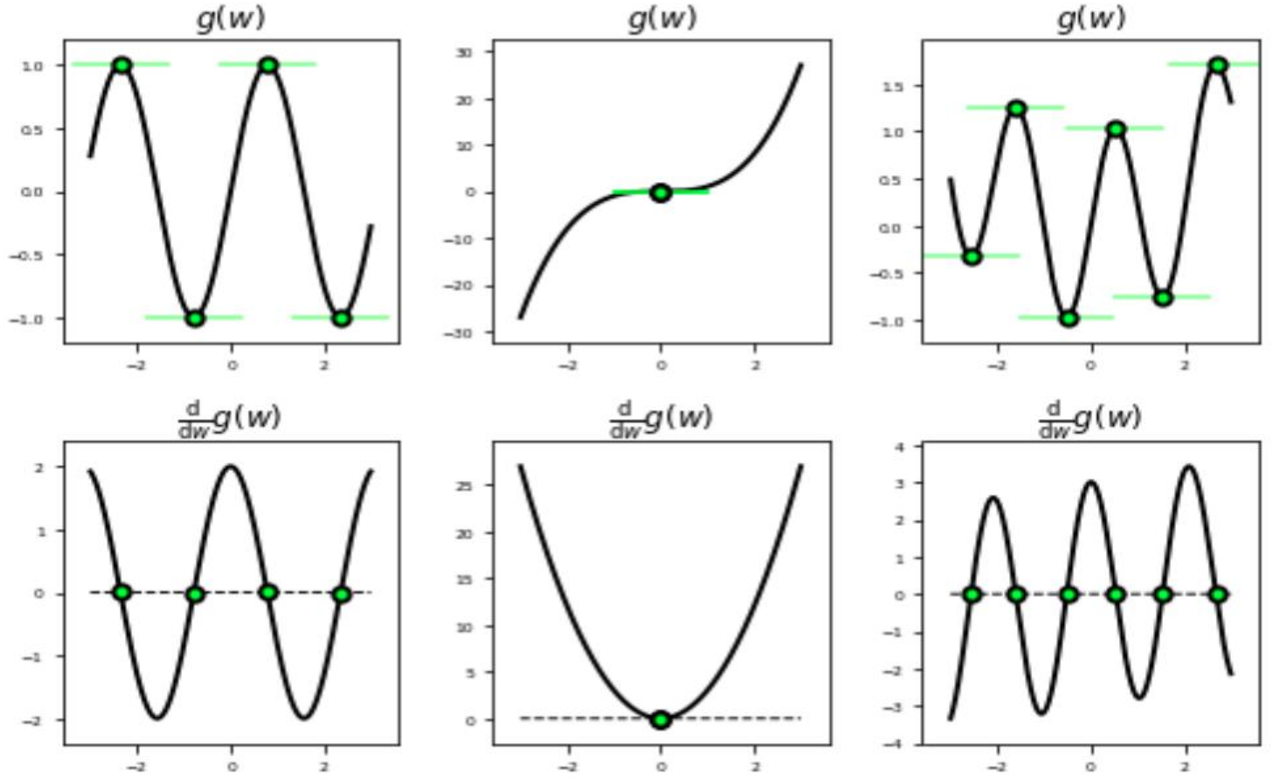
The First-Order Optimality Condition

- Examples: not only *global* minima that have zero derivatives

$g(w) = \sin(2w)$ $\qquad$ $g(w) = w^3$ $\qquad$ $g(w) = \sin(3w) + 0.1w^2$

Zero-order

First-order



TEXAS A&M UNIVERSITY

The First-Order Optimality Condition

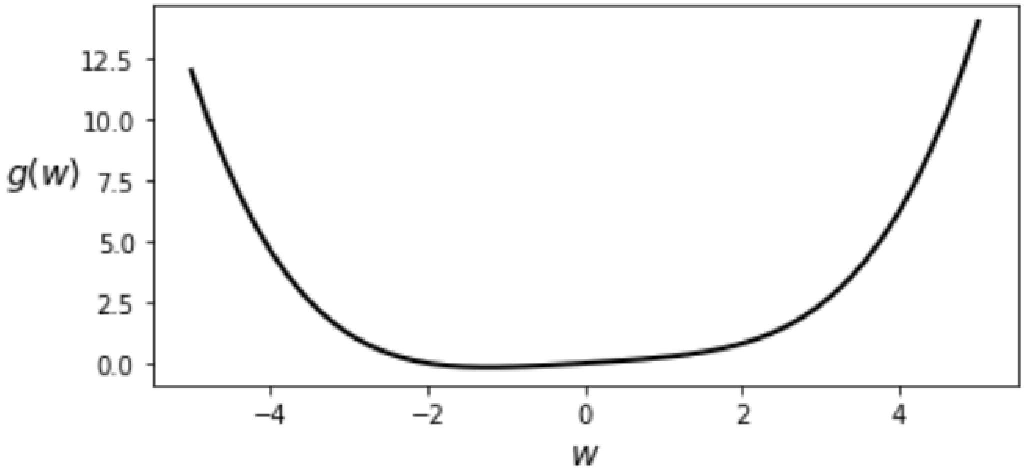
- Zero-valued derivative(s):
  - Local/global minima
  - Local/global maxima
  - Saddle points

- The first order condition for optimality:
  - Stationary points of a function g (including minima, maxima, and saddle points) satisfy the first order condition $\nabla g(v) = 0_{N\times 1}$.
  - Finding global minima → solving a system of (typically nonlinear) equations.
  - Note: if a function is *convex* (e.g., quadratic function), then any point of such a function satisfying the first order condition must be a global minima.

The First-Order Optimality Condition

- Example: global minimum
  - Function:

$$g(w) = \frac{1}{50}\left(w^4 + w^2 + 10w\right)$$



The First-Order Optimality Condition

- Compute first order system:

$$\frac{\mathrm{d}}{\mathrm{d}w}g(w) = \frac{1}{50}\left(4w^3 + 2w + 10\right) = 0$$

- Simplify:

$$2w^3 + w + 5 = 0$$

- Solution:
  - Three possible solutions, but only one is global minimum

$$w = \frac{\sqrt[3]{\sqrt{2031} - 45}}{6^{\frac{2}{3}}} - \frac{1}{\sqrt[3]{6\left(\sqrt{2031} - 45\right)}}$$

The First-Order Optimality Condition

- Example: a general multi-input quadratic function
  - Function:

$$g(\mathbf{w}) = a + \mathbf{b}^T\mathbf{w} + \mathbf{w}^T\mathbf{C}\mathbf{w}$$

  - First derivative (gradient):

$$\nabla g(\mathbf{w}) = 2\mathbf{C}\mathbf{w} + \mathbf{b}$$

  - Setting first derivative to zero gives the form of stationary points:

$$\mathbf{C}\mathbf{w} = -\frac{1}{2}\mathbf{b}$$

The First-Order Optimality Condition

Coordinate descent and the first order optimality condition

- First-order derivative of N dimensional input function g:

$$\nabla g\left(\mathbf{v}\right) = \mathbf{0}_{N\times 1}$$

- On each coordinate:

$$\begin{aligned}
\frac{\partial}{\partial w_1} g(\mathbf{v}) &= 0 \\
\frac{\partial}{\partial w_2} g(\mathbf{v}) &= 0 \\
&\vdots \\
\frac{\partial}{\partial w_N} g(\mathbf{v}) &= 0
\end{aligned}$$

- Hard to solve ‘by hand’.

The First-Order Optimality Condition

- Coordinate-wise: *sequentially* solving one of these equations (or one batch).

$$\frac{\partial}{\partial w_n} g(\mathbf{v}) = 0$$

- Method:
  - First initialize at an input point $\mathbf{w}^0$, and begin by updating the first coordinate

$$\frac{\partial}{\partial w_1} g\left(\mathbf{w}^0\right) = 0$$

  - After obtaining the optimal first weight $\mathbf{w}_1^*$, update the first coordinate $\mathbf{w}^0$, and call the updated set of weights $\mathbf{w}^1$.
  - Continue this pattern to update the n^{th} weight.
  - After going through all N weights a single time, the solution can be refined by sweeping through the weights again.
  - At the k^{th} such sweep the n^{th} weight is updated by solving:

$$\frac{\partial}{\partial w_n} g\left(\mathbf{w}^{k+n-1}\right) = 0$$

The First-Order Optimality Condition

- Example: Minimizing convex quadratic functions via first order coordinate descent
  - Function:

$$g(w_0, w_1) = w_0^2 + w_1^2 + 2$$

  - Written in vector-matrix:

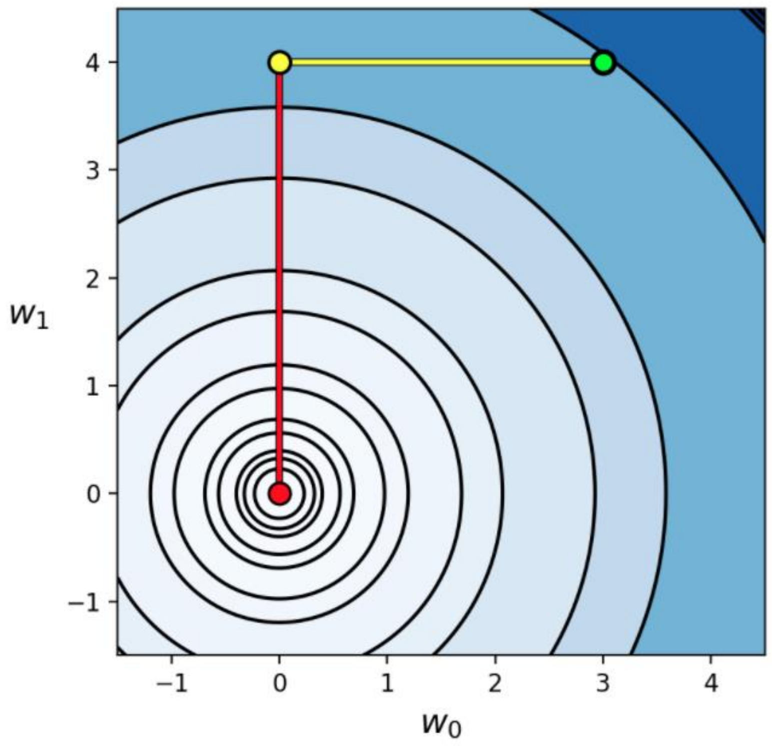
$$a = 2, \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ and } \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

  - Initialization:

$$\mathbf{w} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

The First-Order Optimality Condition

- Run 1 iteration of the algorithm: minimum is found

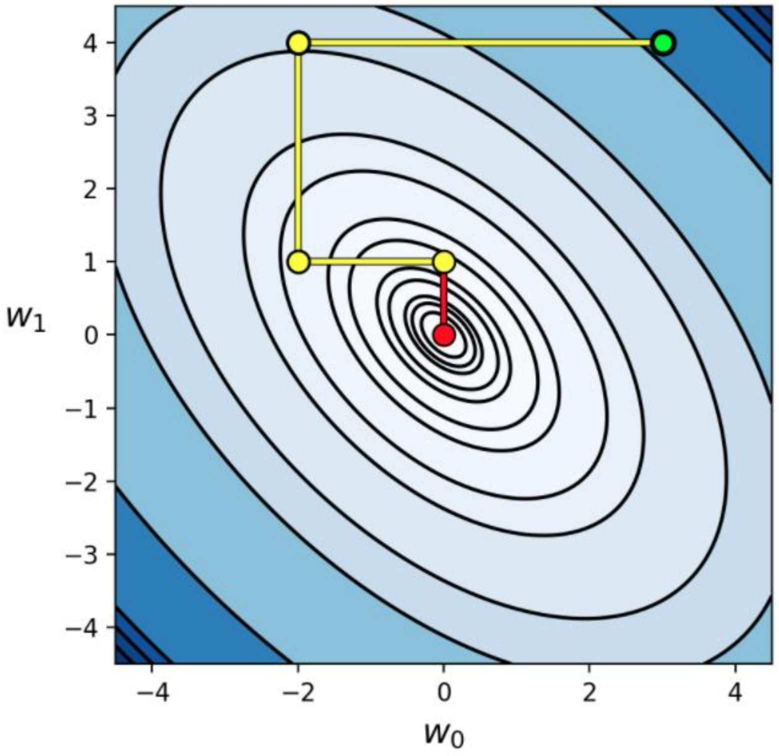


The First-Order Optimality Condition

- For another convex quadratic:

$$a = 20, \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{and } \mathbf{C} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

- The same initialization, run the methods for 2 iterations:



The Geometry of First-Order Taylor Series

Single-input function derivatives and the steepest ascent/descent

- The derivative of a single-input function defines a tangent line at each point its input domain - called its *first order Taylor series approximation.*
- For a differentiable function $g(w)$, the tangent line at each point w^0 is:

$$h(w) = g(w^0) + \frac{\mathrm{d}}{\mathrm{d}w} g(w^0)(w - w^0)$$

- The *steepest ascent* direction is the is the slope of this line (derivative):

$$\text{steepest ascent direction of tangent line} = \frac{\mathrm{d}}{\mathrm{d}w} g(w^0)$$

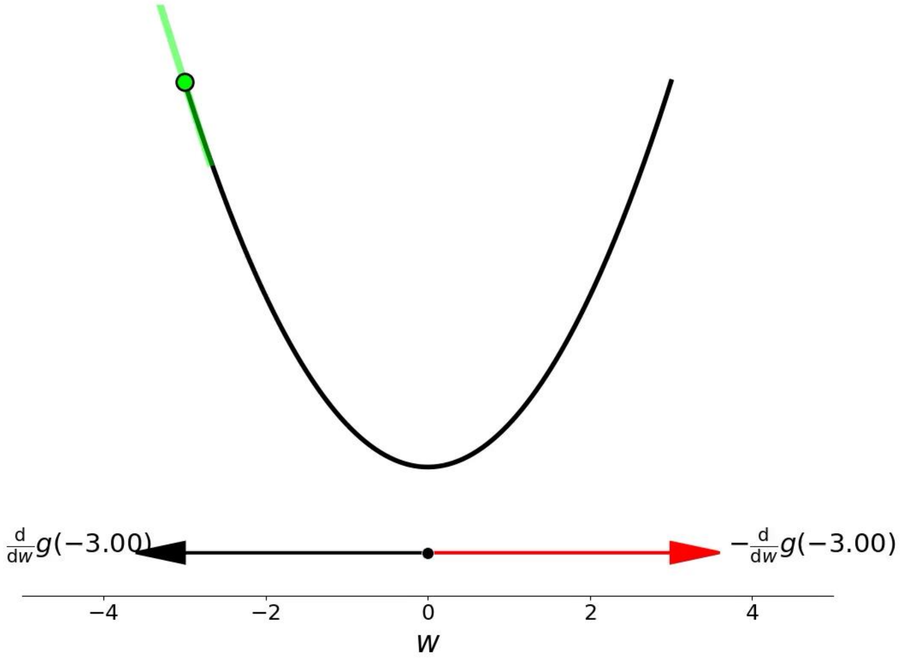
- The *steepest descent direction* is the negative slope of this line (*negative* derivative)

$$\text{steepest descent direction of tangent line} = -\frac{\mathrm{d}}{\mathrm{d}w} g(w^0)$$

The Geometry of First-Order Taylor Series

- Example: the derivative as a direction of ascent/descent for a 2-d quadratic
  - Function:

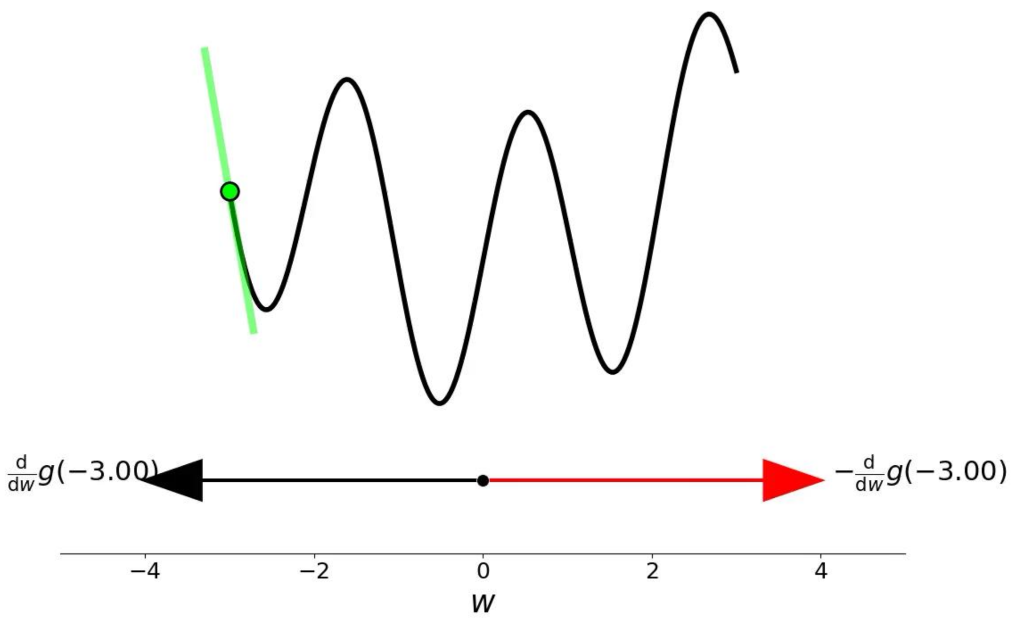
$$g(w) = 0.5w^2 + 1$$



The Geometry of First-Order Taylor Series

- Example: the derivative as a direction of ascent/descent for a 2-d wavy function
  - Function:

$$g(w) = \sin(3w) + 0.1w^2 + 1.5$$



The Geometry of First-Order Taylor Series

Multi-input function derivatives and the direction of greatest ascent / descent

- N dimensional input function $g(w)$: N *partial* derivatives, one in each direction

$$\nabla g(\mathbf{w}) = \begin{bmatrix} \frac{\partial}{\partial w_1} g(\mathbf{w}) \\ \frac{\partial}{\partial w_2} g(\mathbf{w}) \\ \vdots \\ \frac{\partial}{\partial w_N} g(\mathbf{w}). \end{bmatrix}$$

- First order tangent hyperplane at point $\mathbf{w}^0$:

$$h(\mathbf{w}) = g(\mathbf{w}^0) + \nabla g(\mathbf{w}^0)^T(\mathbf{w} - \mathbf{w}^0)$$

The Geometry of First-Order Taylor Series

- The steepest ascent/descent direction along each coordinate axis:

$$\text{steepest ascent direction along } n^{th} \text{ axis} = \frac{\partial}{\partial w_n} g(\mathbf{w}^0)$$

$$\text{steepest descent direction along } n^{th} \text{ axis} = -\frac{\partial}{\partial w_n} g(\mathbf{w}^0)$$

- The steepest ascent/descent direction on the entire N dimensional input space:

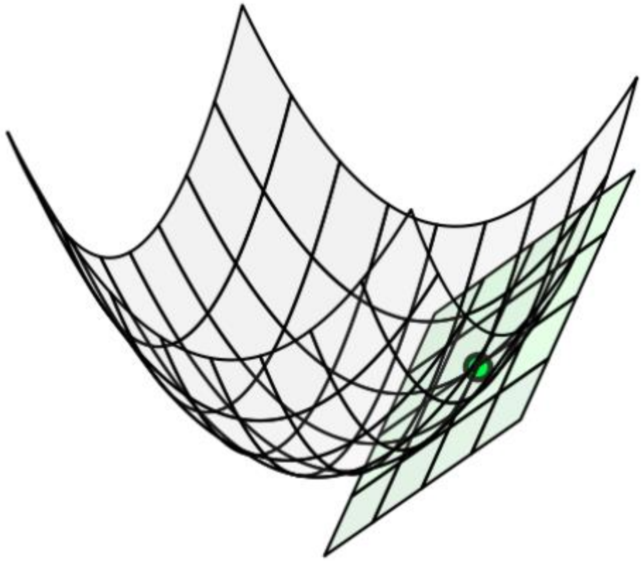
$$\text{ascent direction of tangent hyperplane} = \nabla g(\mathbf{w}^0)$$

$$\text{descent direction of tangent hyperplane} = -\nabla g(\mathbf{w}^0)$$

The Geometry of First-Order Taylor Series

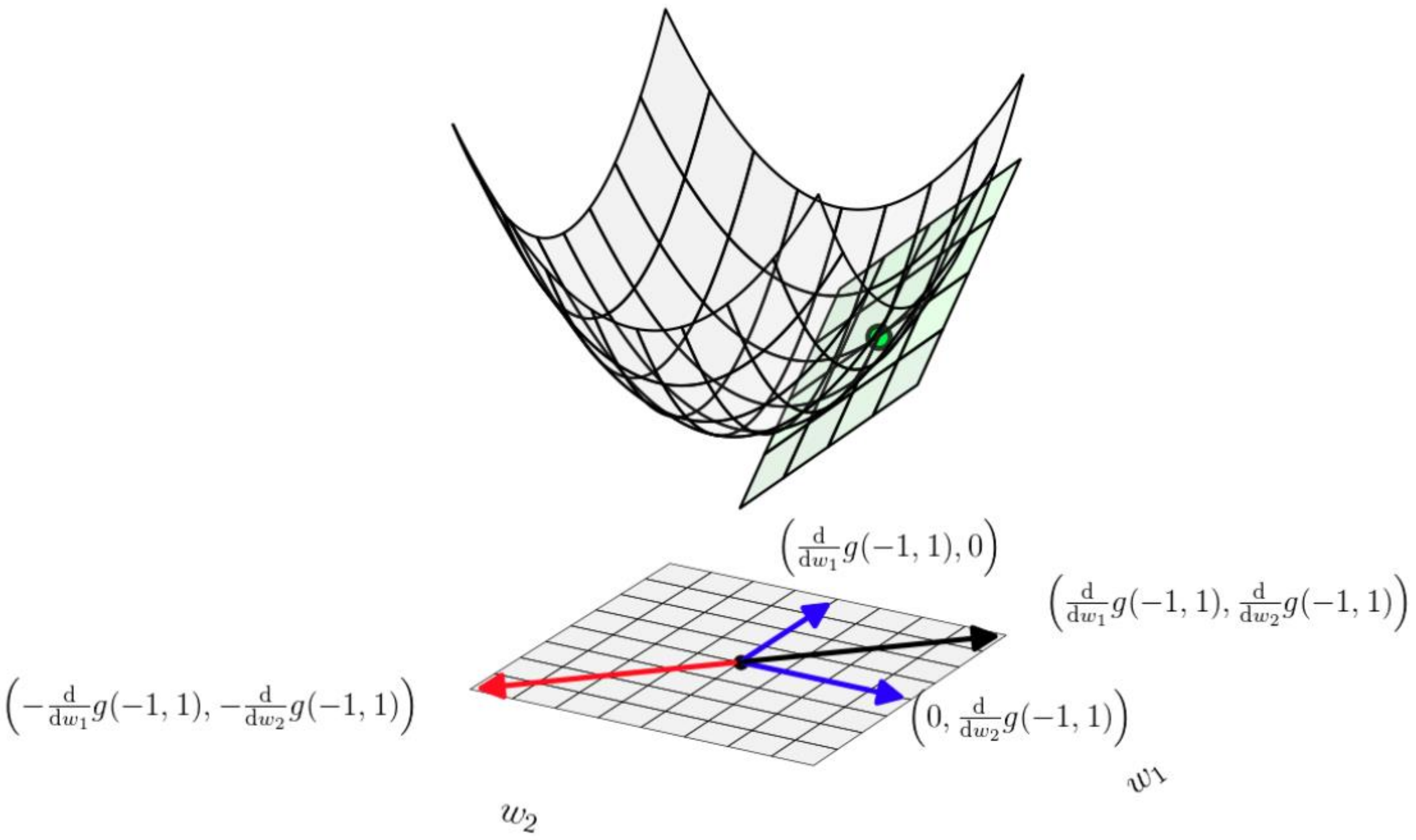
- Example: direction of ascent / descent for a multi-input quadratic function
  - Function:

$$g(w_1, w_2) = w_1^2 + w_2^2 + 6$$



$$\mathbf{w}^0 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

The Geometry of First-Order Taylor Series



Descent direction

Ascent direction

Gradient Descent

The gradient descent algorithm

- Find minima of a given function $g(\mathbf{w})$:

$$\mathbf{w}^{k} = \mathbf{w}^{k-1} + \alpha \mathbf{d}^{k}$$

- $\mathbf{d}^k$ are *descent direction* vectors:

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

- The sequence of steps then take the form:

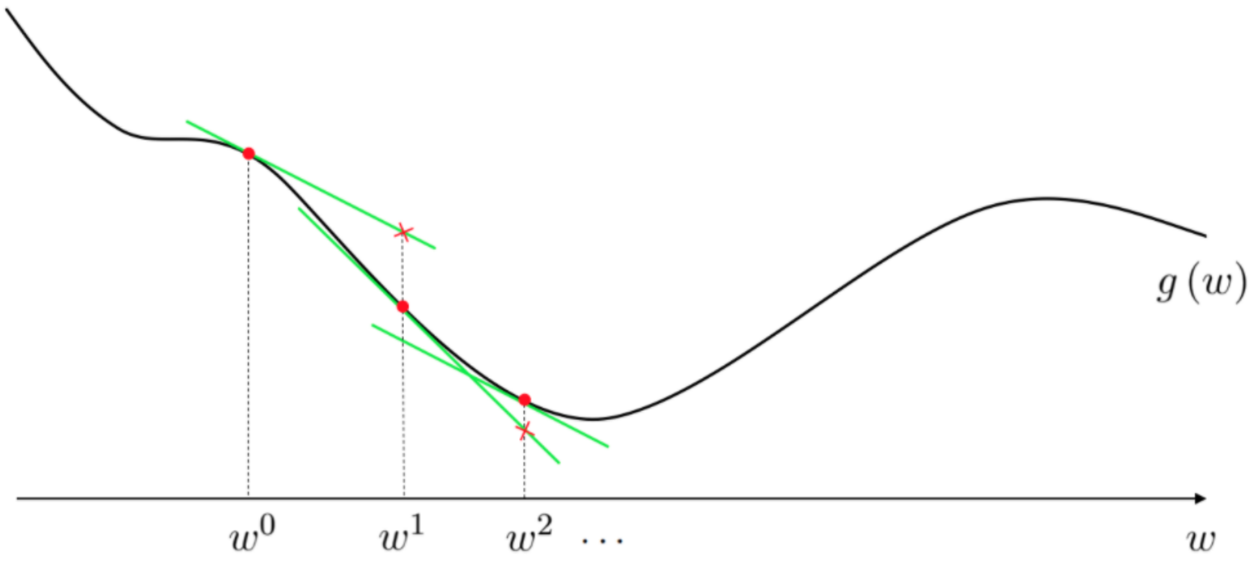
$$\mathbf{w}^{k} = \mathbf{w}^{k-1} - \alpha \nabla g\left(\mathbf{w}^{k-1}\right)$$

- The **gradient descent algorithm**: a local optimization method where the negative gradient is employed as the descent direction at each step.

Gradient Descent

- The gradient descent algorithm pseudo-code

1: **input:** function g, steplength α, maximum number of steps K, and initial point $\mathbf{w}^0$
2: `for` $k = 1 \ldots K$
3: $\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha \nabla g\left(\mathbf{w}^{k-1}\right)$
4: **output:** history of weights $\left\{\mathbf{w}^k\right\}_{k=0}^{K}$ and corresponding function evaluations $\left\{g\left(\mathbf{w}^k\right)\right\}_{k=0}^{K}$



Gradient Descent

- How to set the α parameter (learning rate)?
  - Fixed steplegnth
  - Diminishing steplegnth
- When does gradient descent stop?
  - The algorithm will halt near stationary points of a function (minima or saddle points) if the steplength is chosen wisely.
  - If the step does not move from the prior point $\mathbf{w}^{k-1}$ significantly:
    - The direction we are traveling in is vanishing i.e., $-\nabla g\left(\mathbf{w}^k\right) \approx \mathbf{0}_{N\times 1}$
    - A *stationary point* of the function

Gradient Descent

- Example 1: A convex single input example
  - Minimize the polynomial function:

$$g(w) = \frac{1}{50}\left(w^4 + w^2 + 10w\right)$$

  - First order optimality condition (difficulty to calculate by hand)

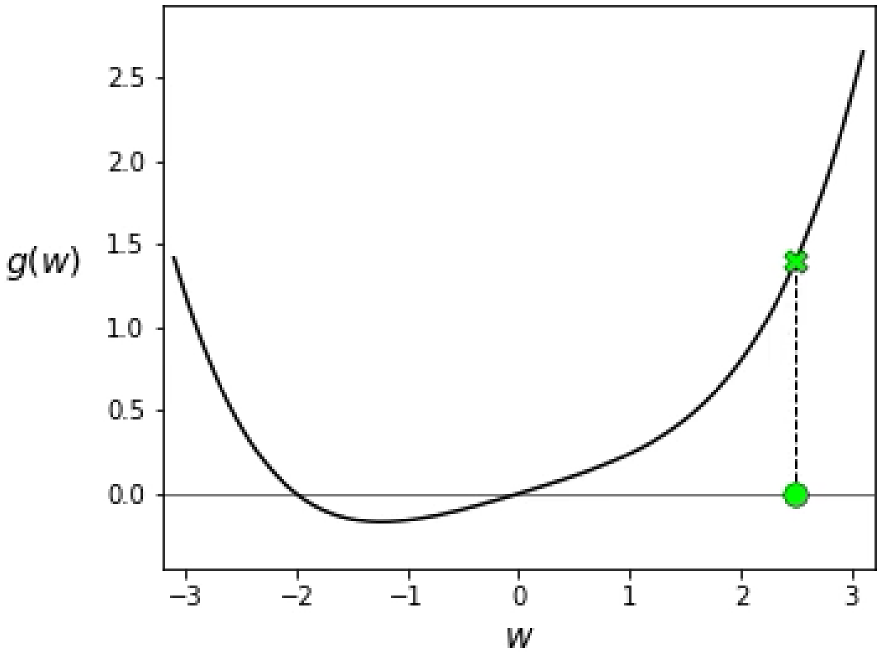
$$w = \frac{\sqrt[3]{\sqrt{2031} - 45}}{6^{\frac{2}{3}}} - \frac{1}{\sqrt[3]{6\left(\sqrt{2031} - 45\right)}}$$

  - Computing the gradient

$$\frac{\partial}{\partial w} g\left(w\right) = \frac{2}{25} w^3 + \frac{1}{25} w + \frac{1}{5}$$

Gradient Descent

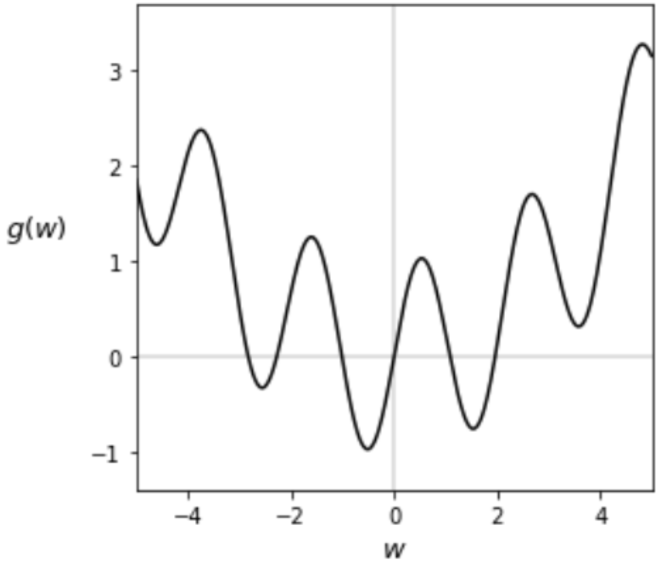
- Initialization $\mathbf{w}^0 = 2.5$
- Steplength/learning rate $\alpha = 1$
- 25 iterations



Gradient Descent

- Example 2: A non-convex single input example (Lecture 4)
  - Function:

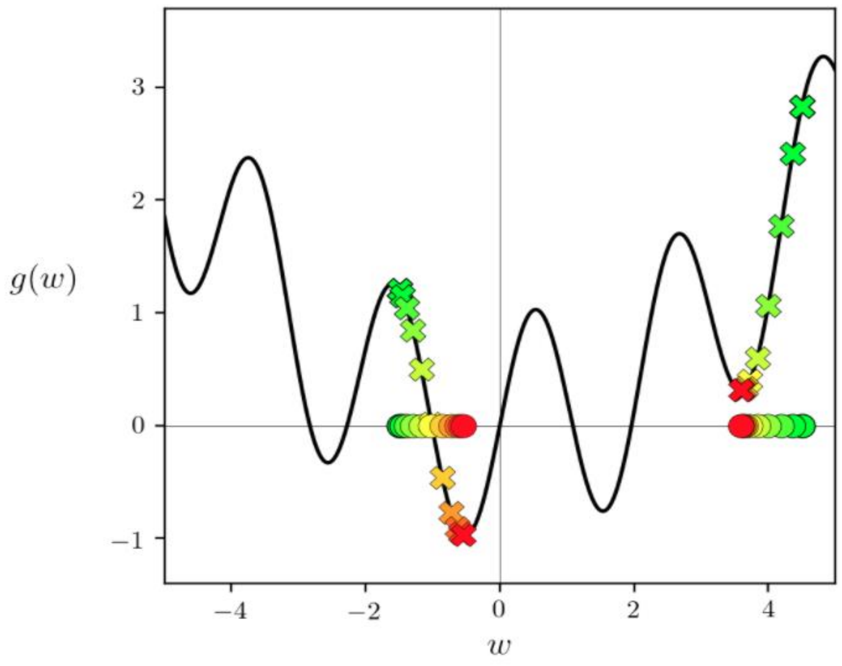
$$g(w) = \sin(3w) + 0.1w^2$$



  - Algorithm parameters:
    - Steplength parameter: $\alpha = 0.1$
  - Starting point:
    - Run 1: $w^0 = 4.5$
    - Run 2: $w^0 = -1.5$

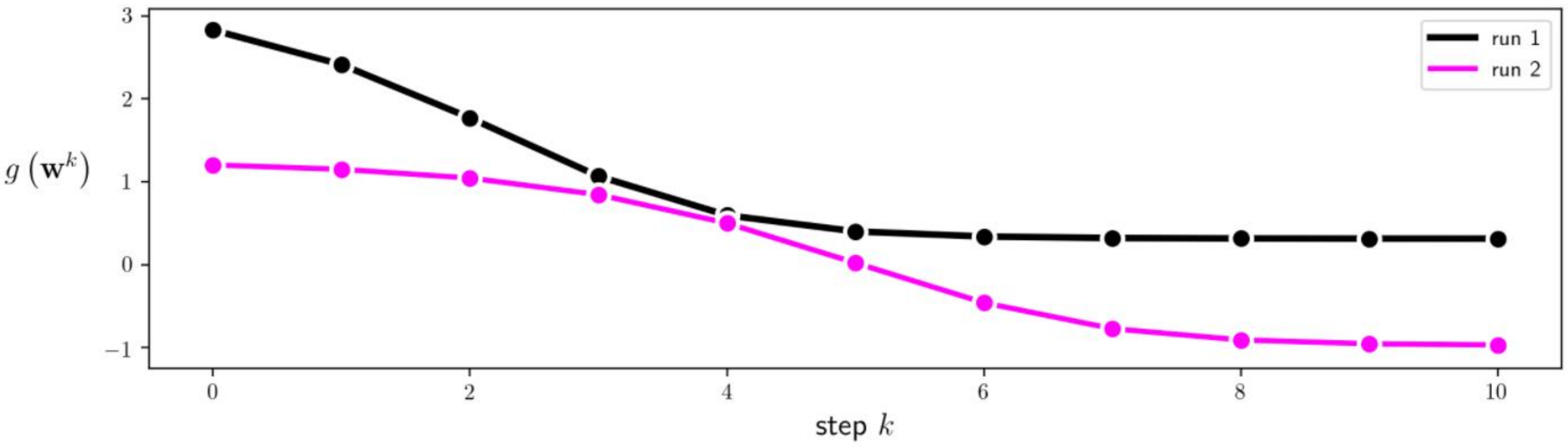
Gradient Descent

- Gradient descent path: green to red
  - Run 1: right
  - Run 2: left



Gradient Descent

- Cost function history plots
  - Run 2 cost is lower than run 1
  - Run 1 is a local minimum



Gradient Descent

- Example 3: A convex multi-input example
  - Function: a multi-input quadratic function

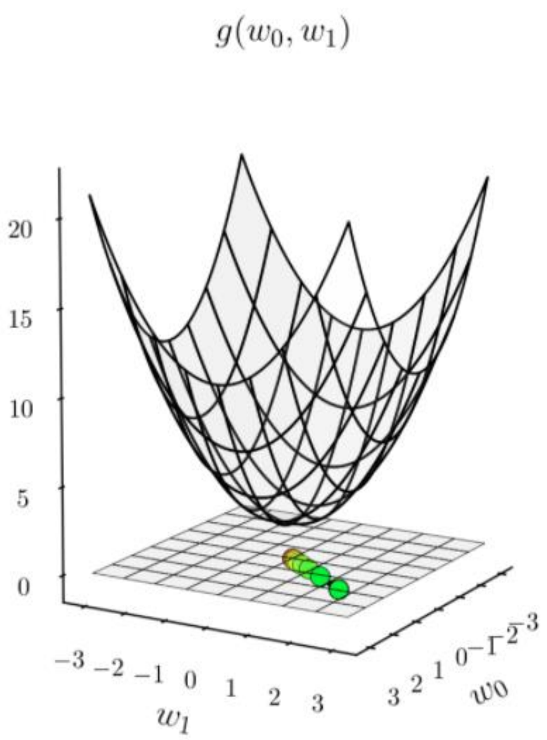
$$g(w_1, w_2) = w_1^2 + w_2^2 + 2$$

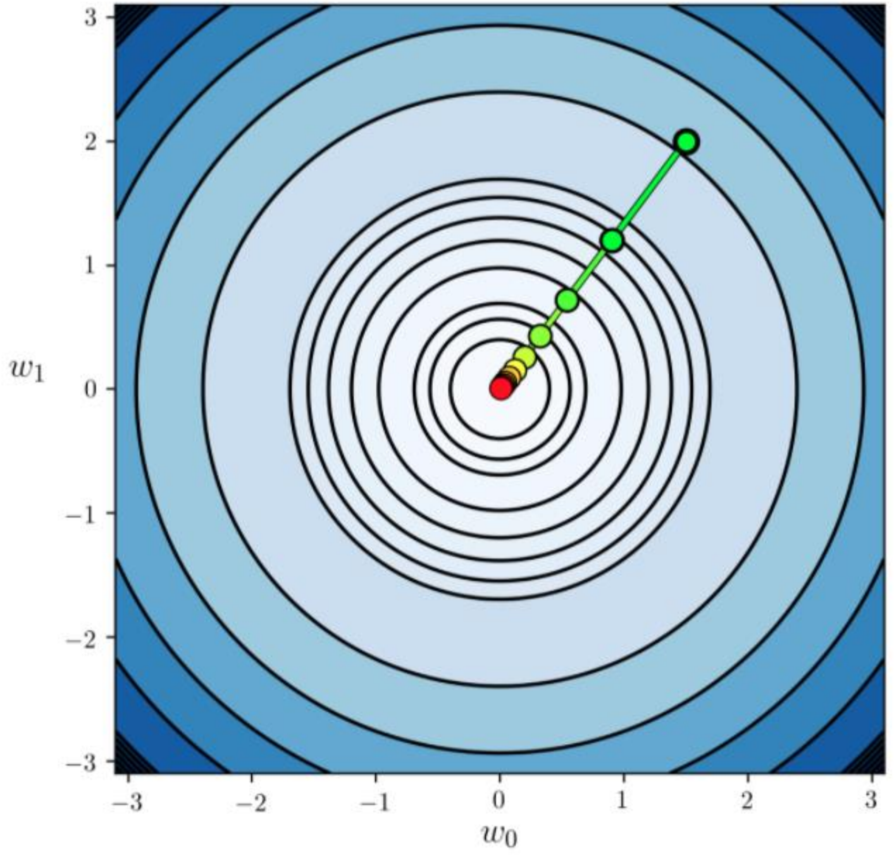
  - Run: 10 steps with the steplength / learning rate $\alpha = 0.1$
  - Gradient:

$$\nabla g(\mathbf{w}) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$

Gradient Descent

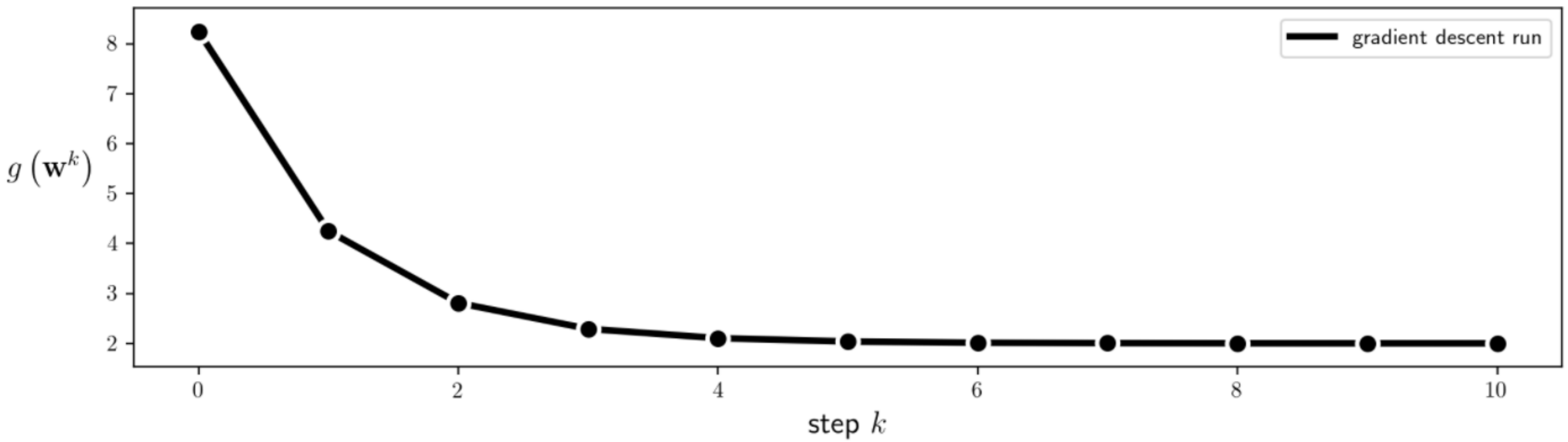
- Gradient descent path





Gradient Descent

- Cost function history plot



- Cost function history plots are a valuable debugging tool, particularly true with higher dimensional functions that we cannot visualize.

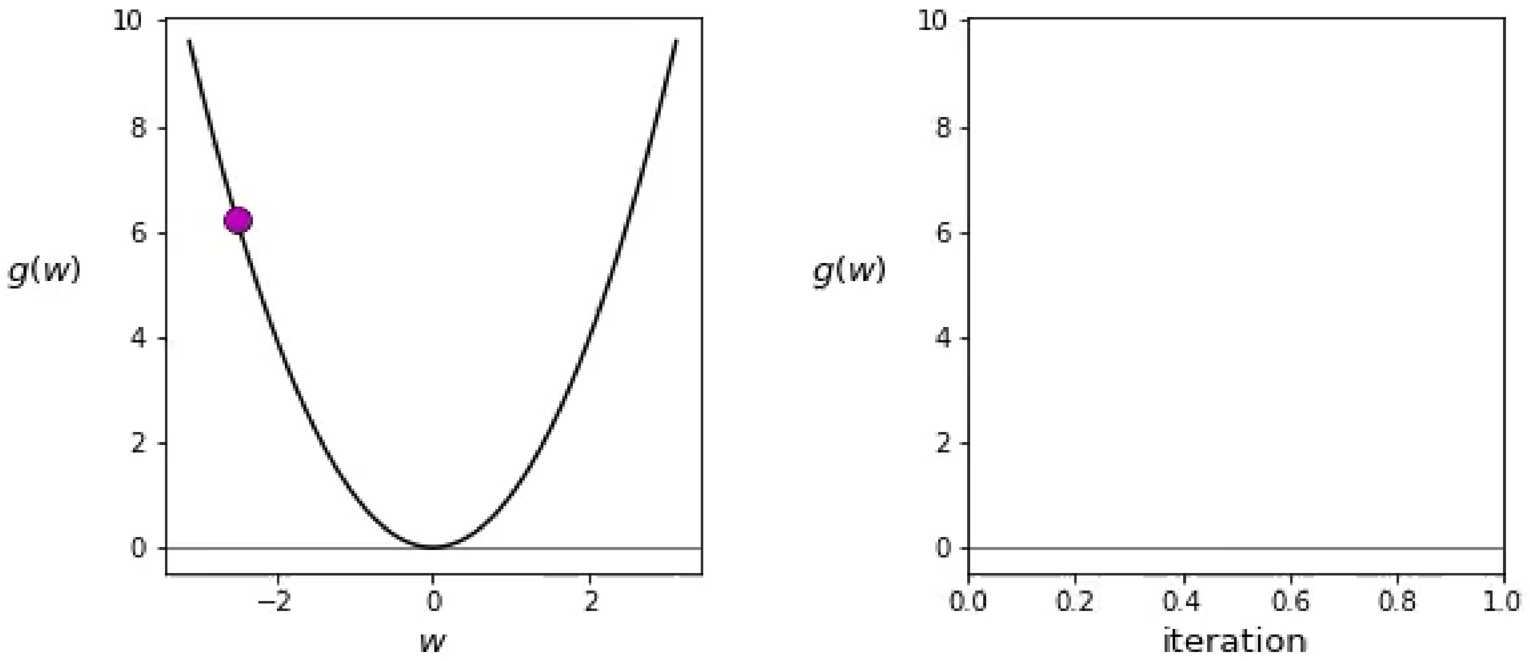
Gradient Descent

Basic steplength choices for gradient descent

- Common choices:
  - Fixed α: $10^{-\gamma}$ where γ is an integer.
  - Diminishing α: $\frac{1}{k}$ where at k^{th} step of a run.
- Choosing a particular value for the steplength / learning rate α at each step of gradient descent mirrors that of any other local optimization method: α should be chosen to induce the most rapid minimization possible.

Gradient Descent

- Example 4: fixed steplength for a single input convex function
  - Function: $g(w) = w^2$
  - Right panel: cost function plot

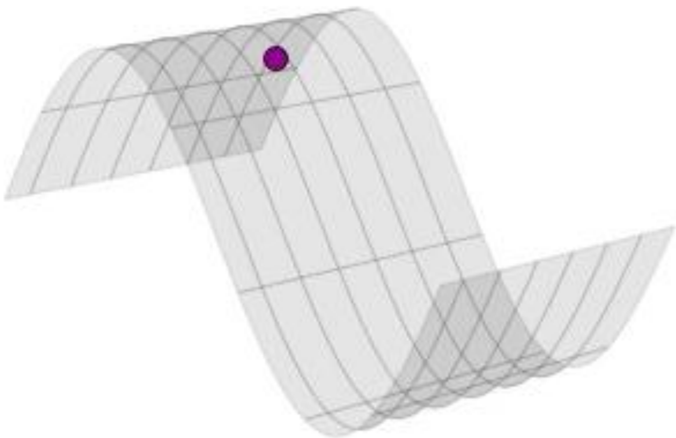


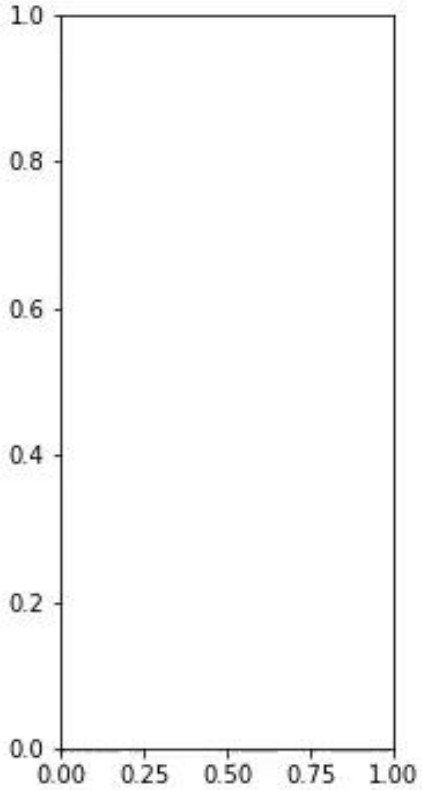
Gradient Descent

- Setting
  - Initialization: $w^0 = -2.5$
  - Five steps of gradient descent (unnormalized)
- Fixed steplength/learning rate:
  - When the steplength parameter is too large, the sequence of evaluations begins to rocket out of control.
  - Keep track of the best weights seen thus far in the process when implementing gradient descent.
  - The final weights resulting from the run may not in fact provide the lowest value depending on function, steplength parameter, etc.

Gradient Descent

- Example 5: fixed steplength selection for a multi-input non-convex function
  - Function: $g(w_1, w_2) = \sin(w_1)$





Gradient Descent

- Comparing fixed and diminishing steplengths
  - Function:

$$g(w) = |w|$$

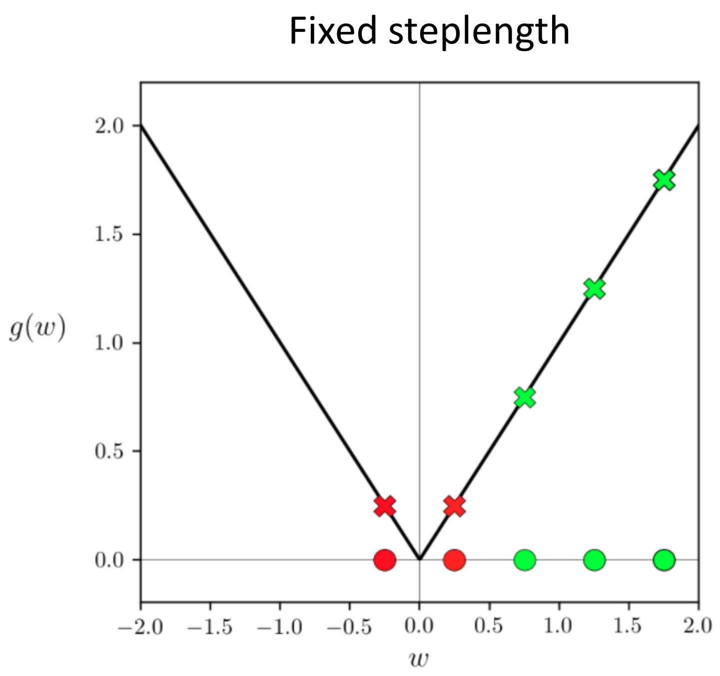
  - Single global minimum: $w = 0$
  - Gradient: everywhere but at $w = 0$

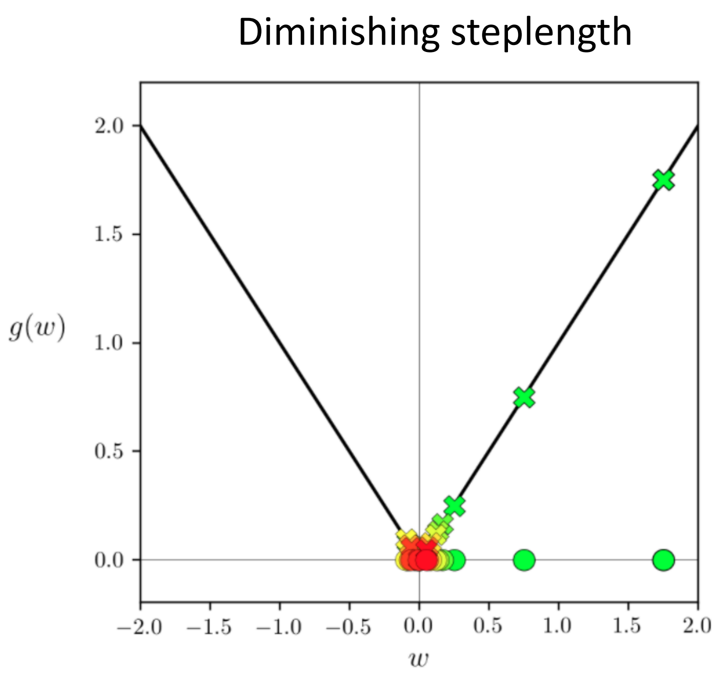
$$\frac{\mathrm{d}}{\mathrm{d}w} g(w) = \begin{cases} +1 & \text{if } w > 0 \\ -1 & \text{if } w < 0 \end{cases}$$

  - Initialization: $w^0 = 2$
  - Comparison:
    - Fixed steplength: $\alpha = 0.5$
    - Diminishing steplength: $\alpha = \frac{1}{k}$

Gradient Descent

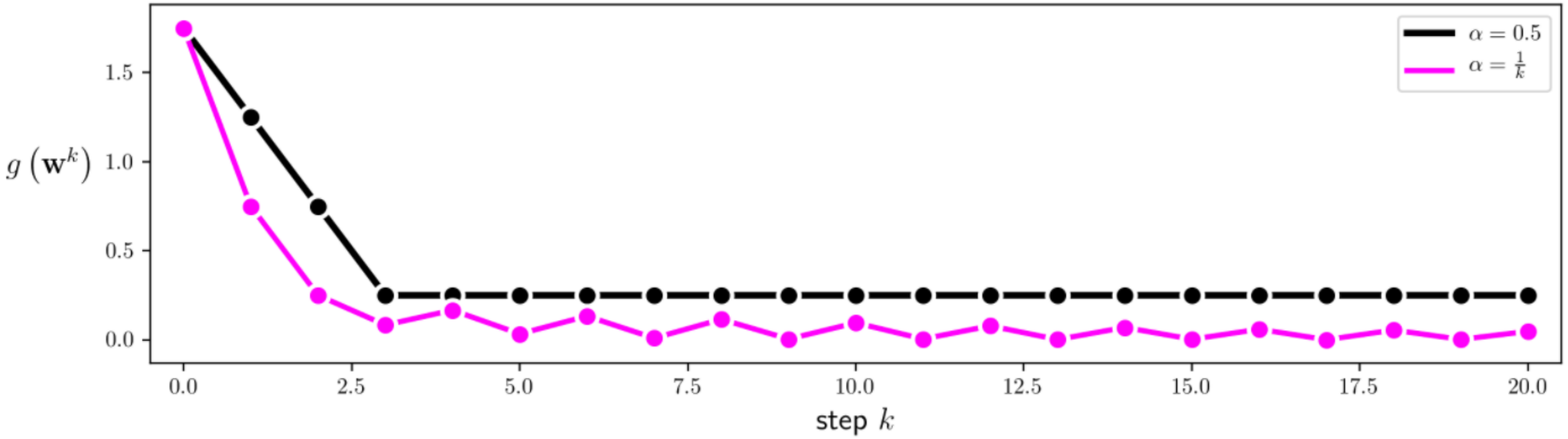
- Gradient descent path:





Gradient Descent

- Cost function plot
  - A diminishing steplength is absolutely necessary in order to reach a point close to the minimum of this function



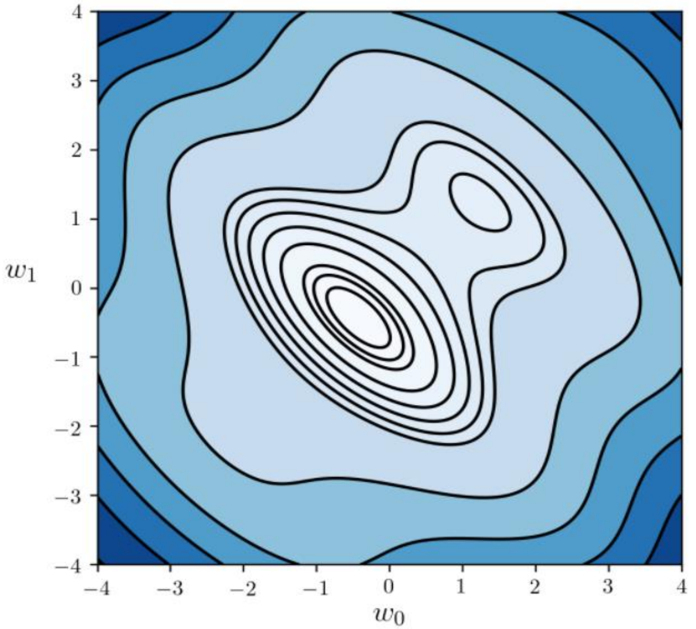
Gradient Descent

Oscillation in the cost function history plot is not always a bad thing

- Example:
  - Function:

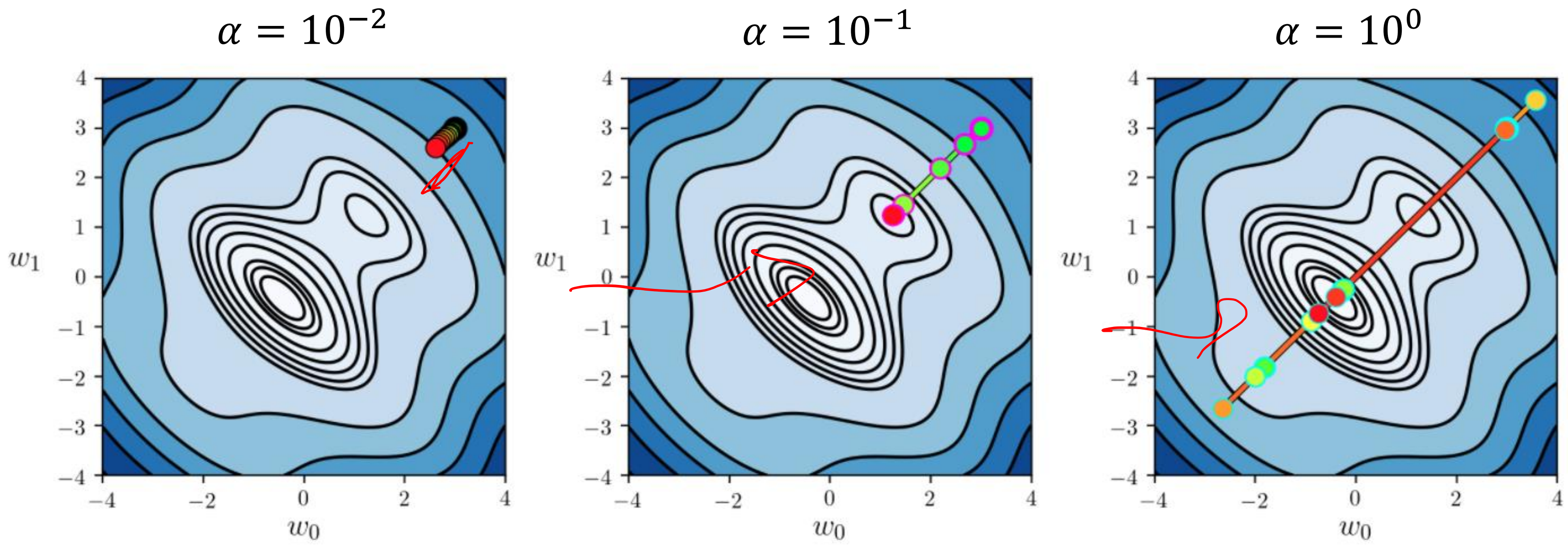
$$g(\mathbf{w}) = w_0^2 + w_1^2 + 2\sin(1.5\,(w_0 + w_1))^2 + 2$$

  - Local minimum: $\begin{bmatrix}1.5\\1.5\end{bmatrix}$
  - Global minimum: $\begin{bmatrix}-0.5\\-0.5\end{bmatrix}$
  - Initial point: $\begin{bmatrix}3\\3\end{bmatrix}$
  - Steplength: fixed



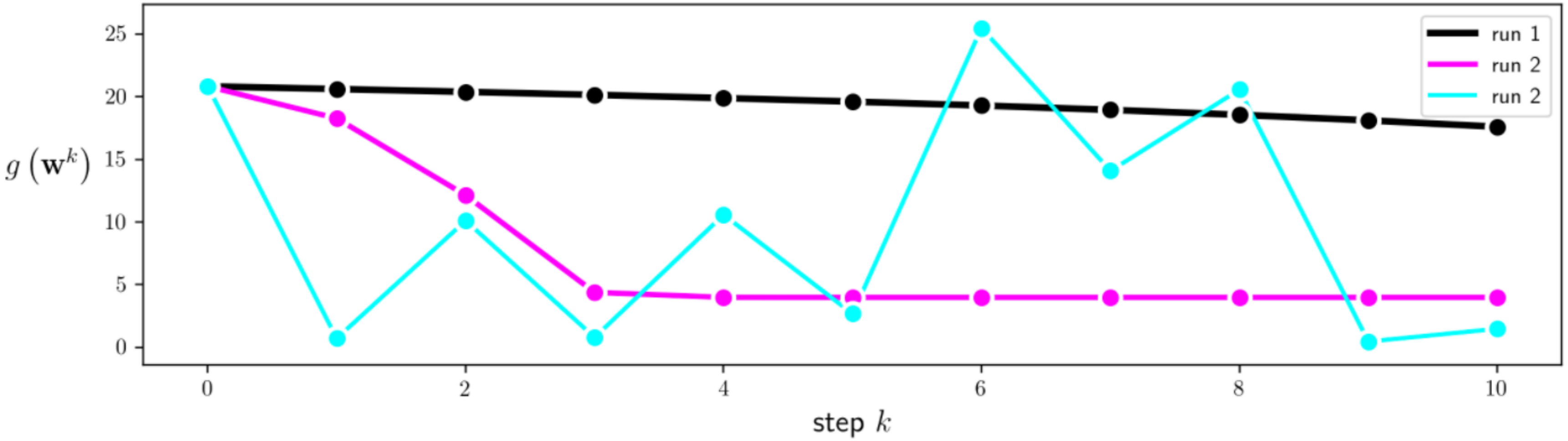
Gradient Descent

- Run 1: steplength too small
- Run 2: local minimum near $\begin{bmatrix}1.5\\1.5\end{bmatrix}$
- Run 3: global minimum near $\begin{bmatrix}-0.5\\-0.5\end{bmatrix}$



Gradient Descent

- Cost function plot



- Run 1: not strictly decreasing at each step
- Run 3: lead to oscillatory but indeed find the lowest point out of all three runs performed.

Takeaways

- How to Compute Gradients
- How to Run Gradient Descent
- Understanding impact of each iteration's step size