CSCE 421: Machine Learning

Lecture 5: Gradient Descent

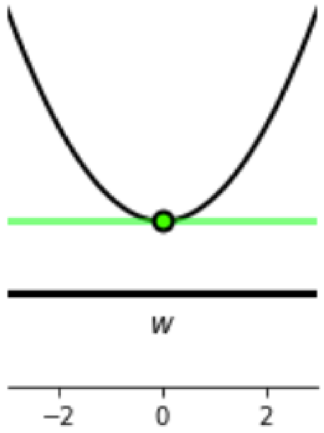
Texas A&M University
Bobak Mortazavi
Ryan King
Zhale Nowroozilarki

Goals for this Lecture

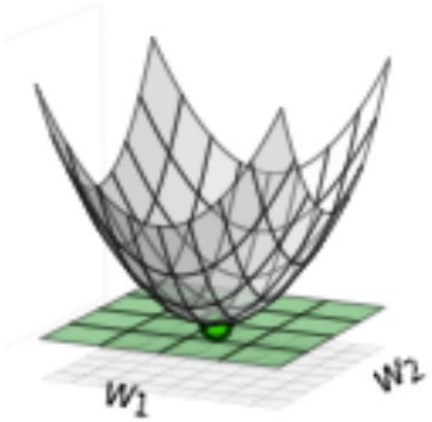
- First Order Optimization – The Gradient Function
- Understand Gradient Descent
- Understanding Limitations to Gradient Descent
- Second Order Optimization – Convexity/Concavity
- Newton's Method for Descent

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 $N = 1$:

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

  - Input dimension N:

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

- First order system:

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

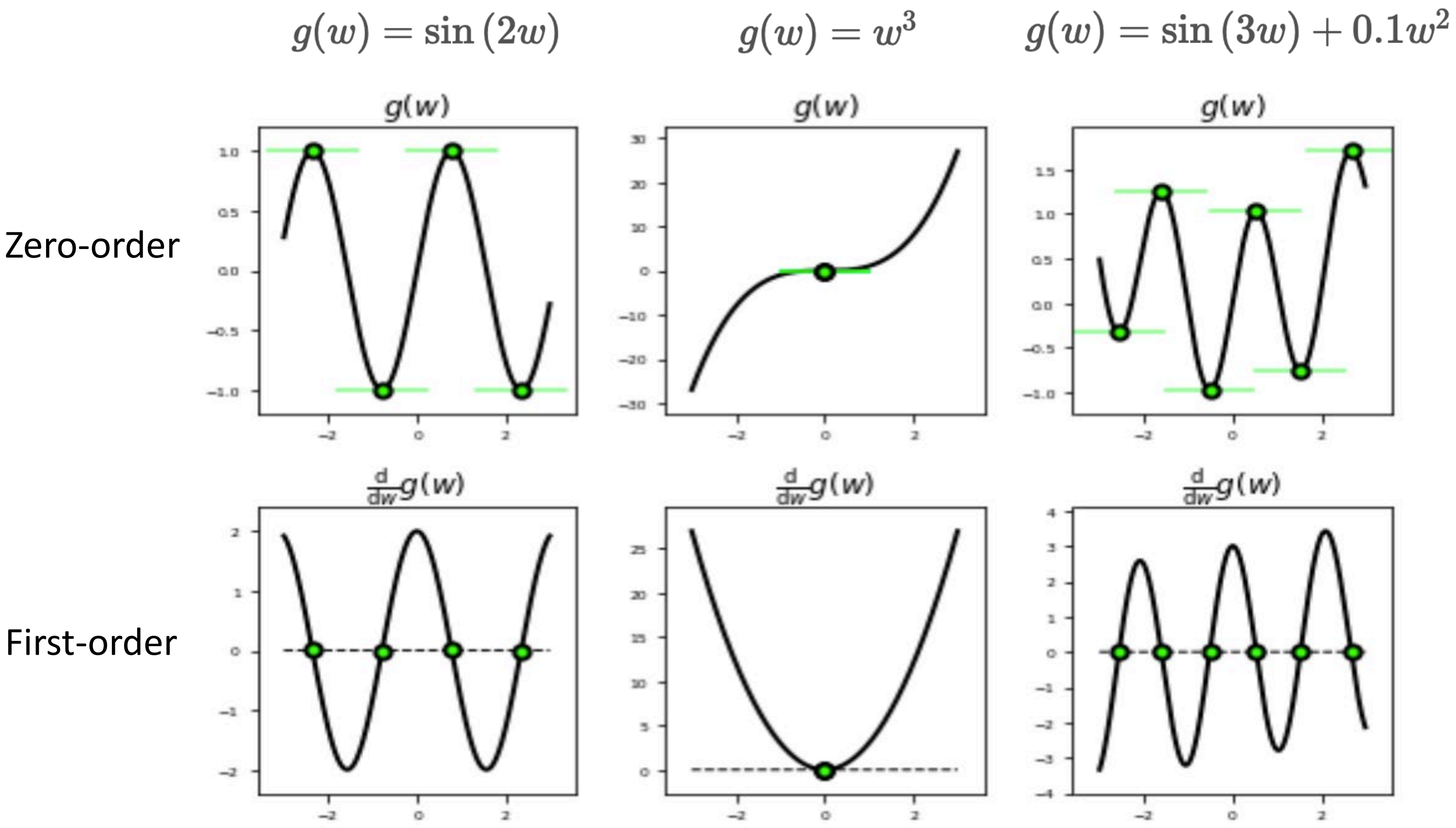
The First-Order Optimality Condition

- 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



The First-Order Optimality Condition

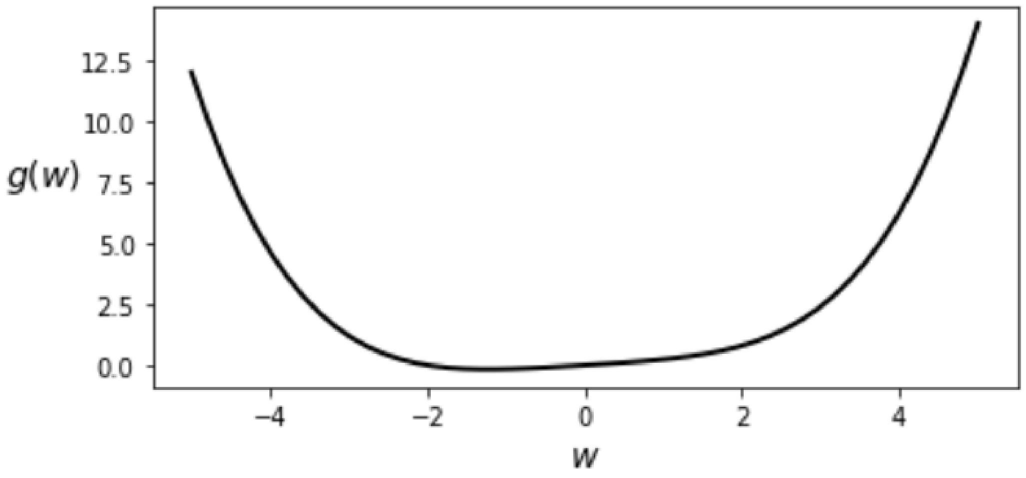
- Zero-valued derivative(s):
  - Local/global minima
  - Local/global maxima
  - Saddle points

- 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:

$$\nabla g\left(\mathbf{v}\right) = \mathbf{0}_{N\times1}$$

- On each coordinate:

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

- 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:

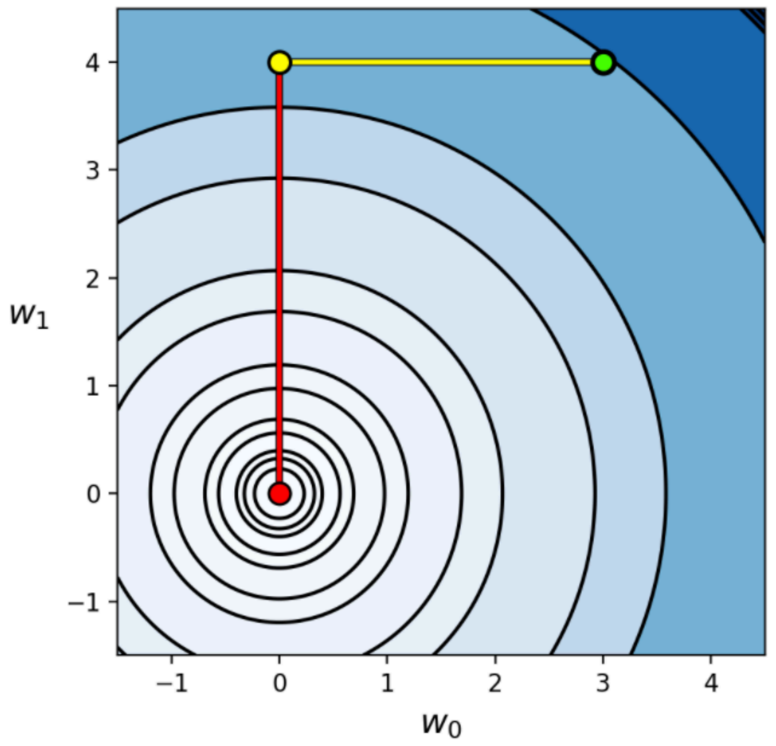
$$a = 2,\ \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ \text{and}\ \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

  - Initialization:

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

The First-Order Optimality Condition

- Run 1 iteration of the algorithm: minimum is found

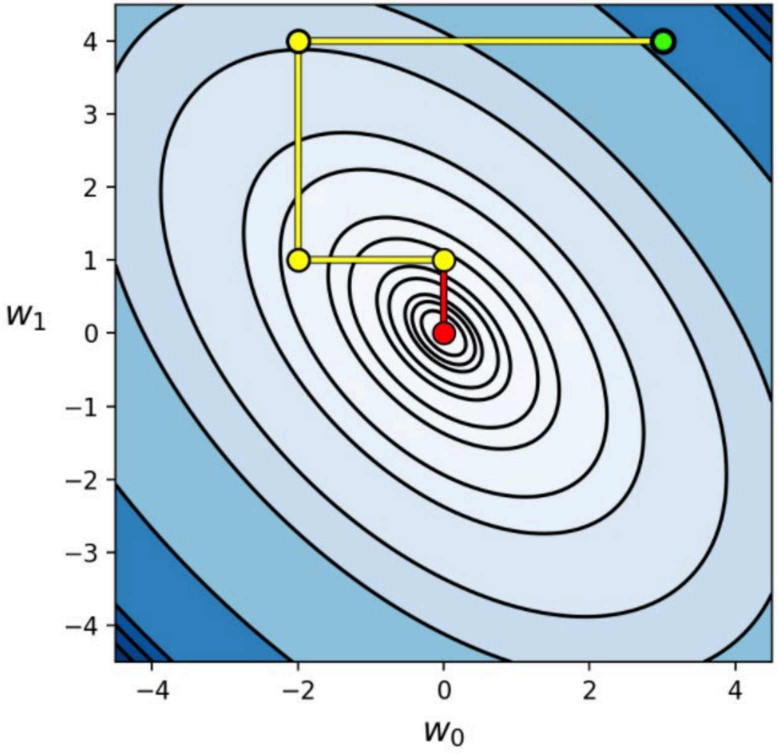


The First-Order Optimality Condition

- 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):

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

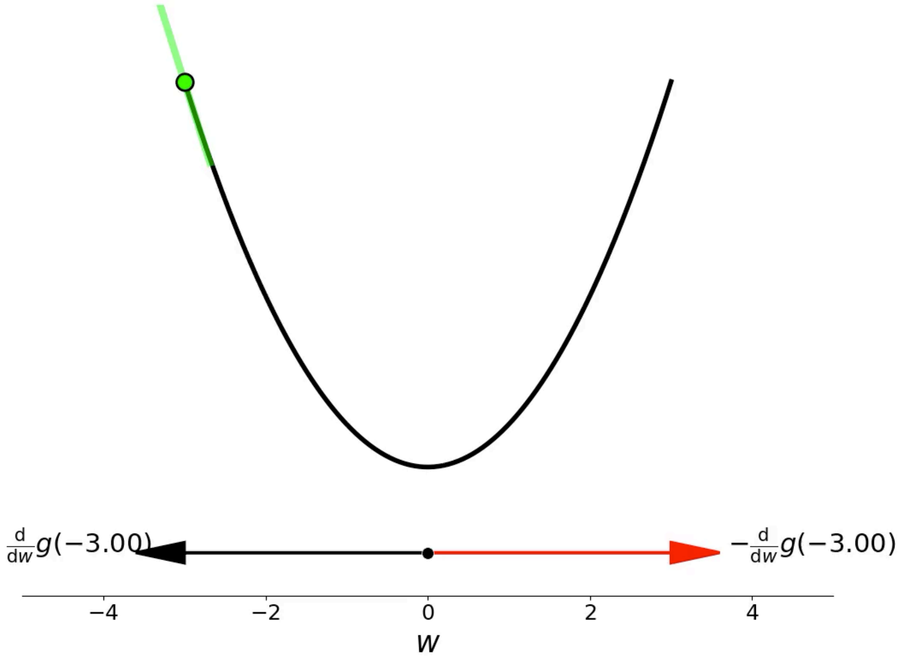
- The *steepest descent direction* is the negative slope of this line (*negative* derivative)

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

The Geometry of First-Order Taylor Series

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

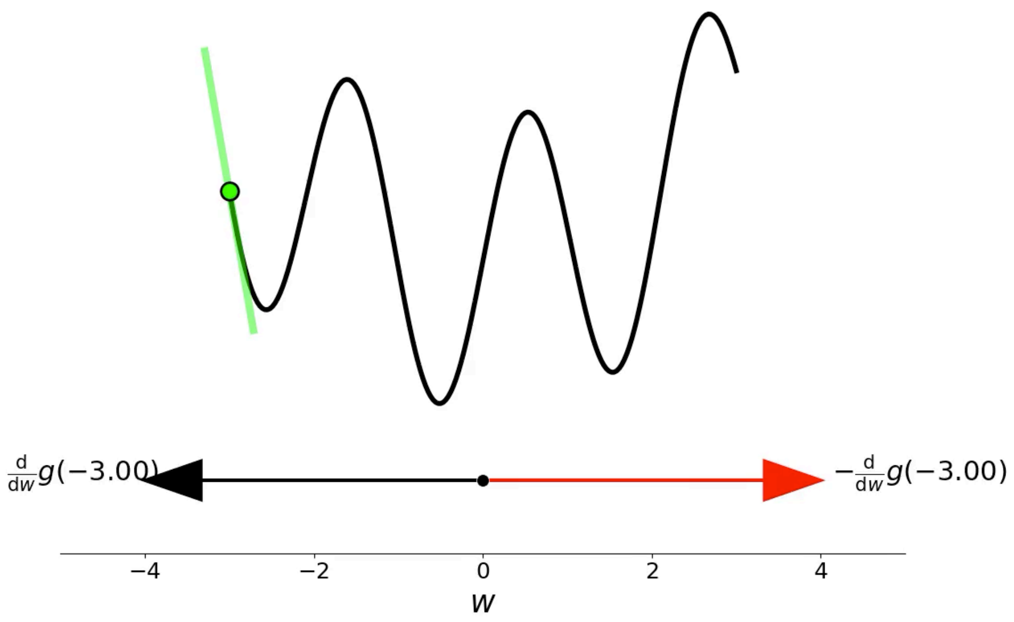
$$g(w) = 0.5w^2 + 1$$



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:

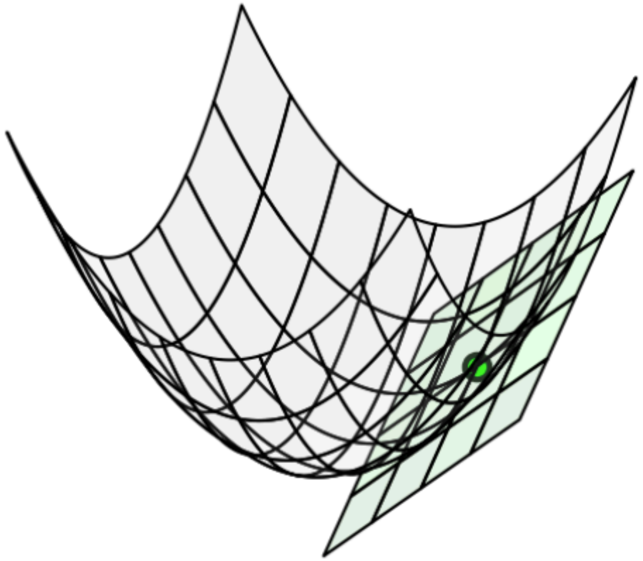
$$\text{ascent direction of tangent hyperplane} = \nabla g(\mathbf{w}^0)$$

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

The Geometry of First-Order Taylor Series

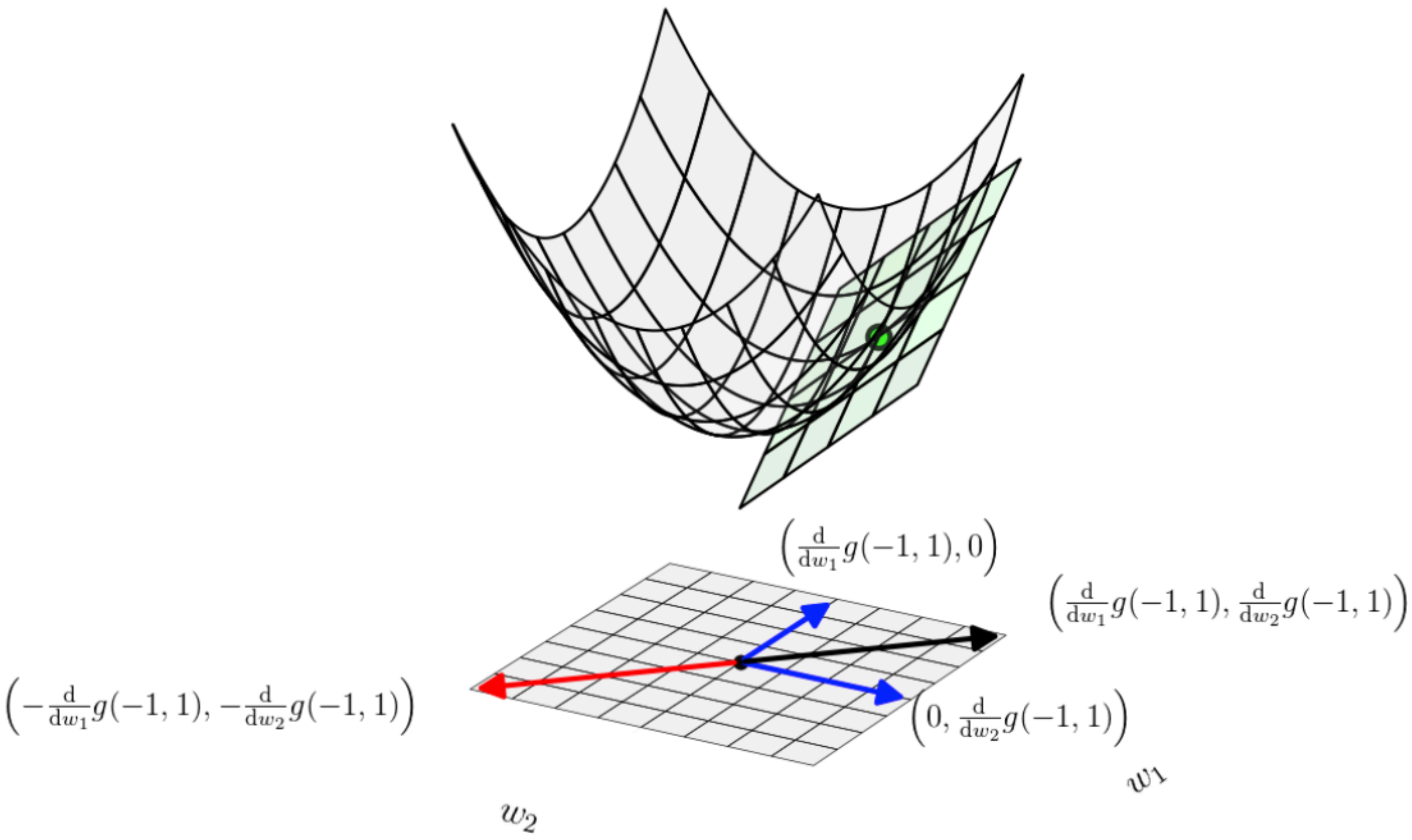
- Example: direction of ascent / descent for a multi-input quadratic function
  - Function:

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



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

The Geometry of First-Order Taylor Series



Descent direction

Ascent direction

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

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:

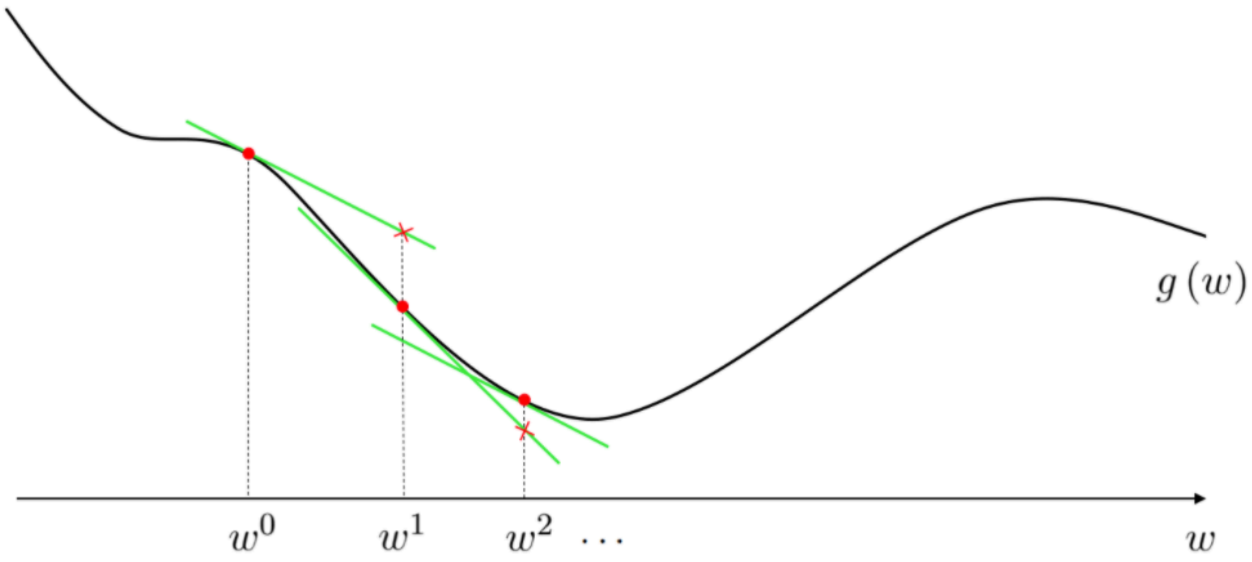
$$\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha \nabla g\left(\mathbf{w}^{k-1}\right)$$

- 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` $k = 1...K$
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)

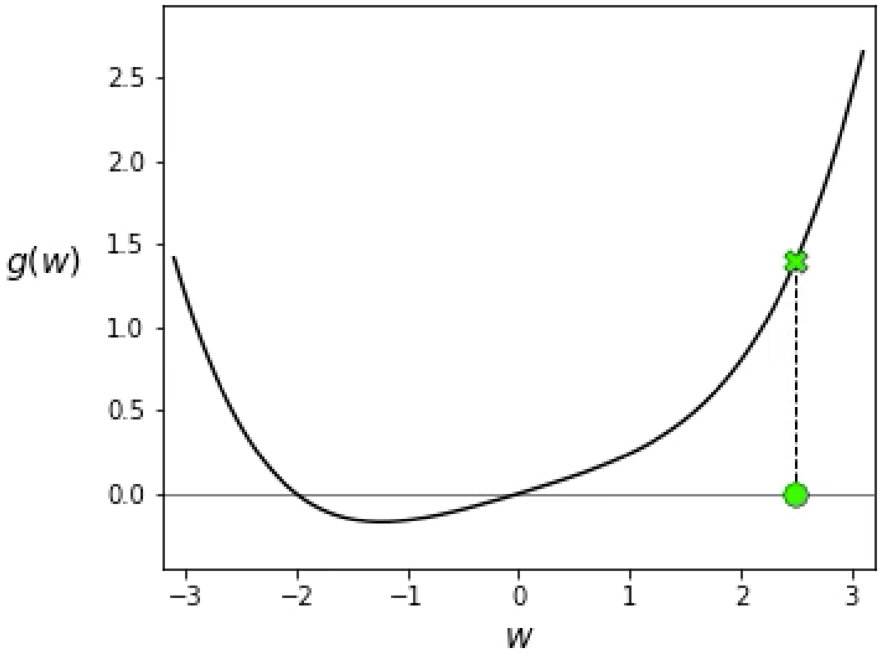
$$w = \frac{\sqrt[3]{\sqrt{2031} - 45}}{6^{\frac{2}{3}}} - \frac{1}{\sqrt[3]{6\left(\sqrt{2031} - 45\right)}}$$

  - 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

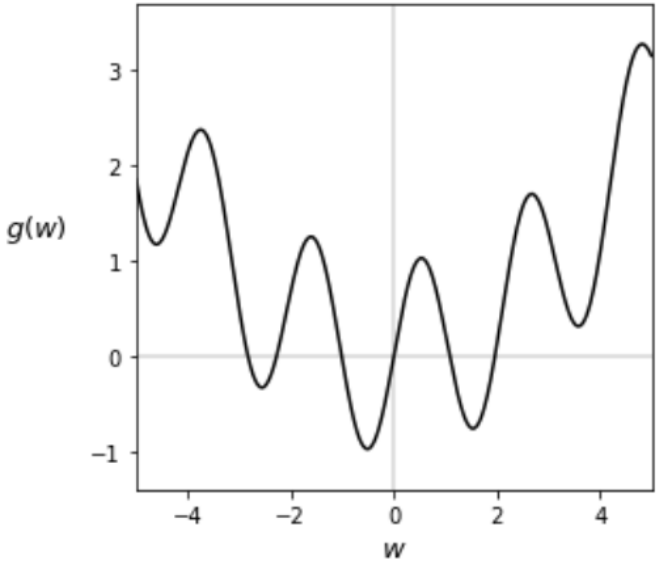


Gradient Descent

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

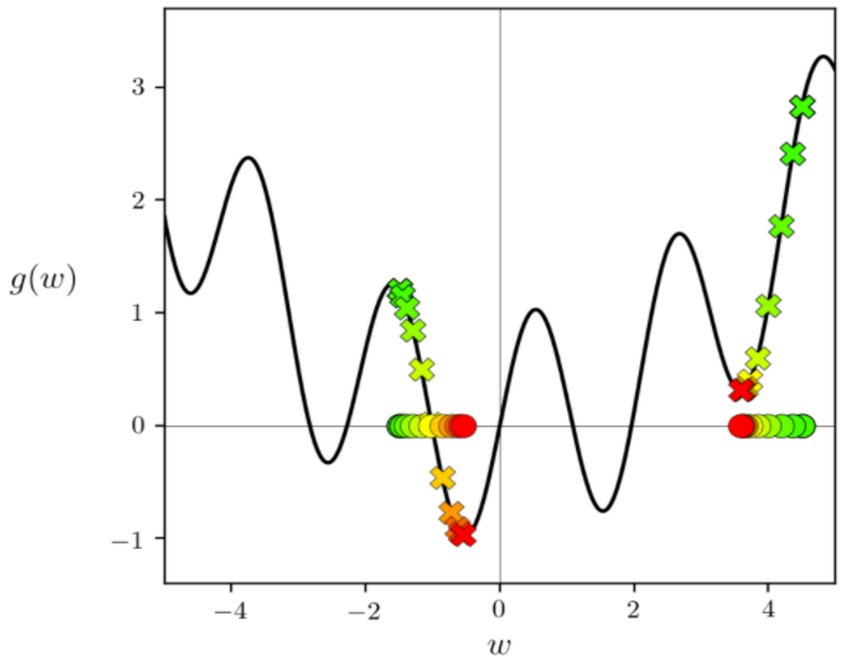
$$g(w) = \sin(3w) + 0.1w^2$$

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



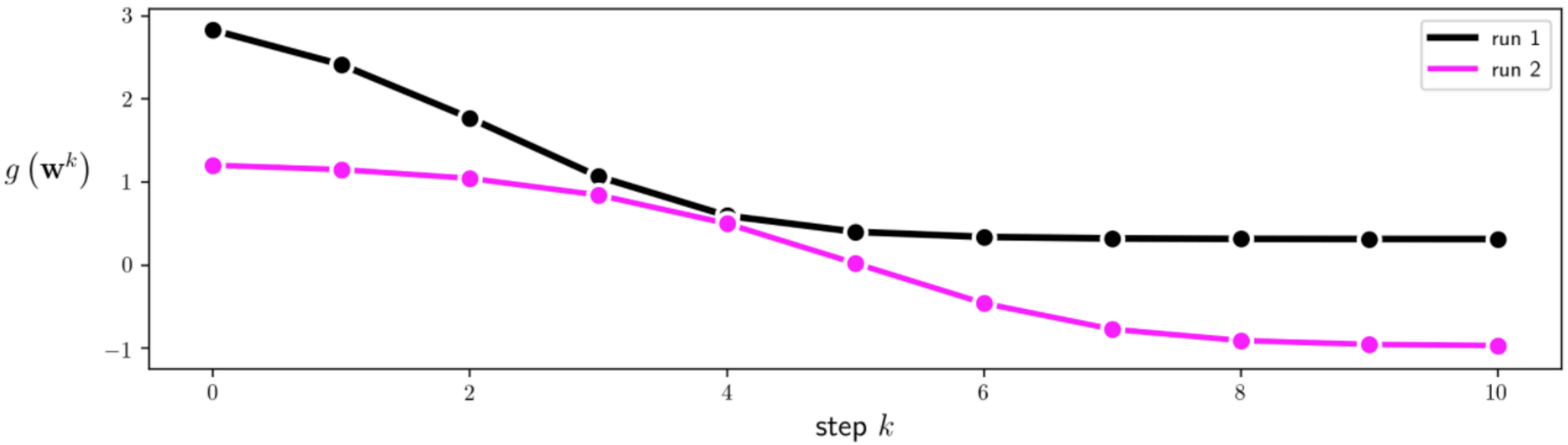
Gradient Descent

- 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

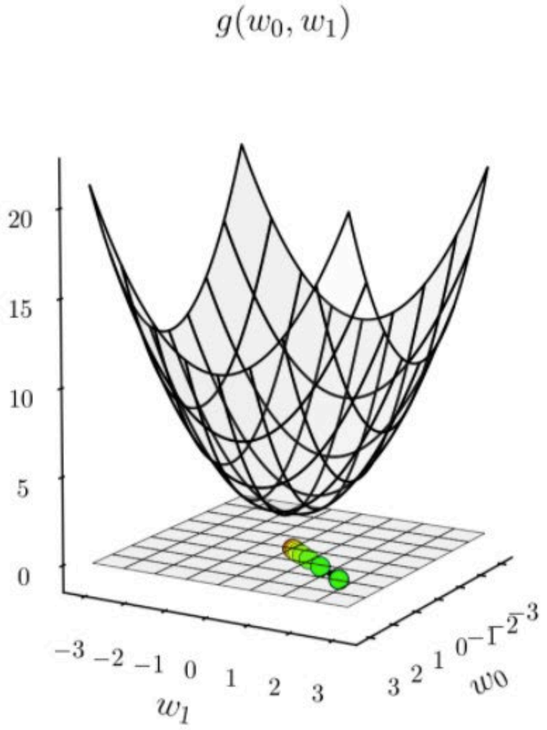
$$g(w_1, w_2) = w_1^2 + w_2^2 + 2$$

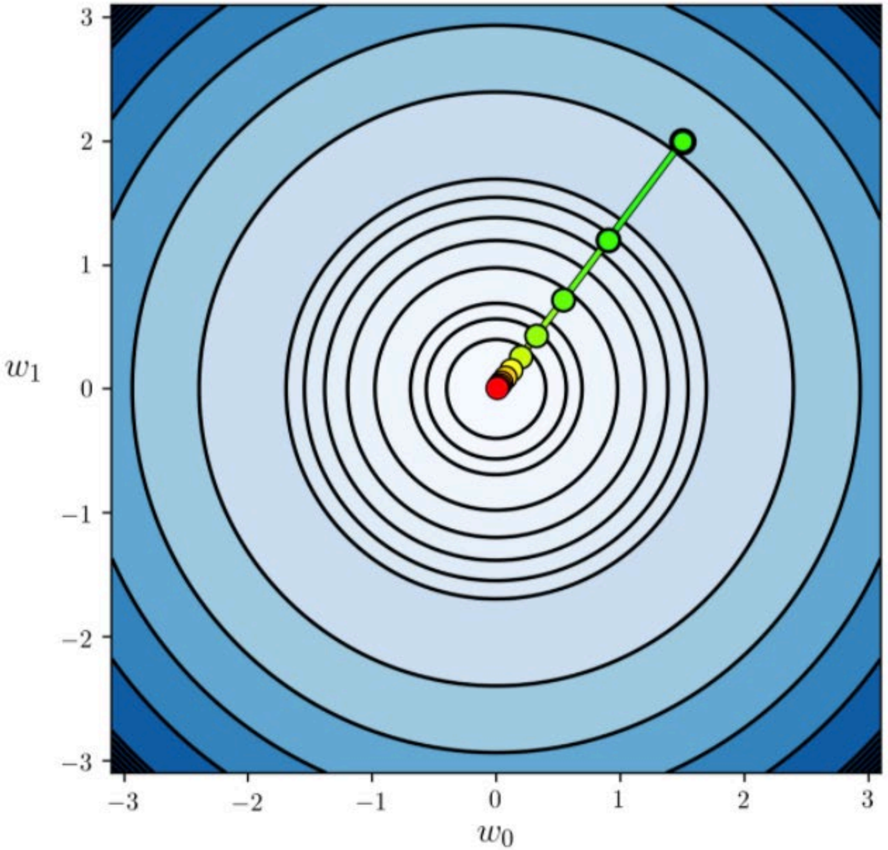
  - Run: 10 steps with the steplength / learning rate $\alpha = 0.1$
  - Gradient:

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

Gradient Descent

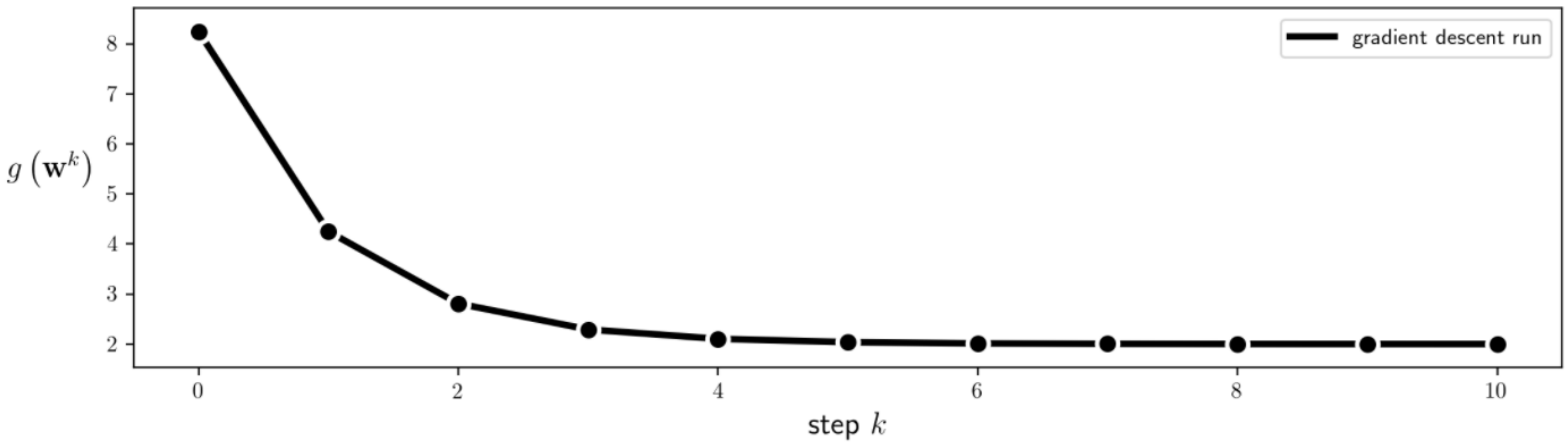
- Gradient descent path





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.

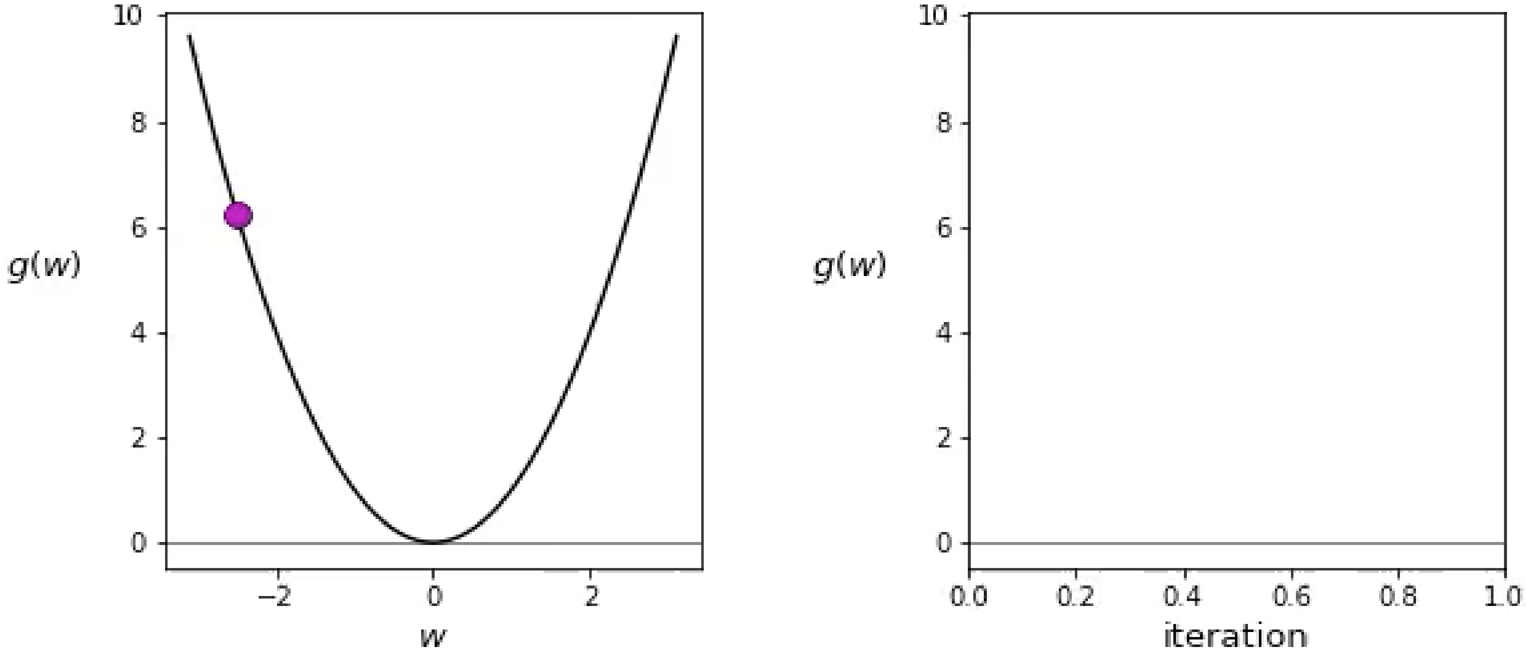
Gradient Descent

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

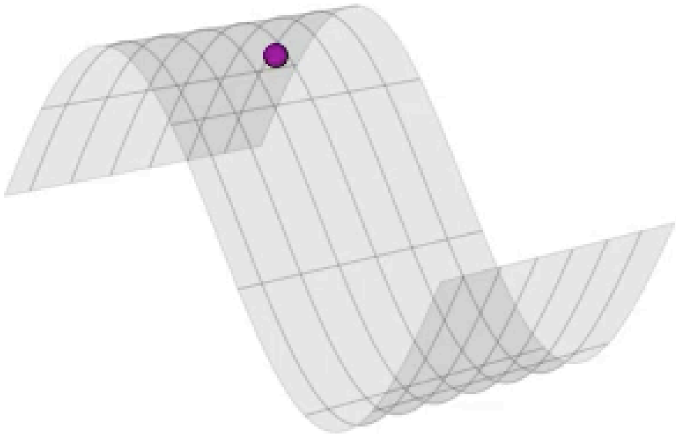


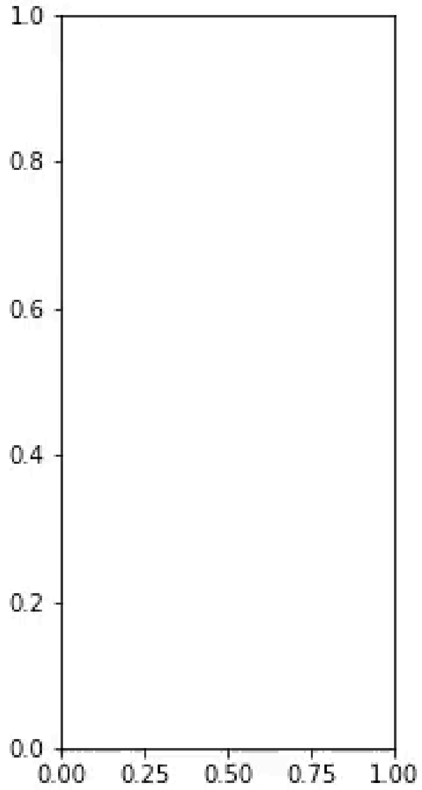
Gradient Descent

- 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 $w = 0$

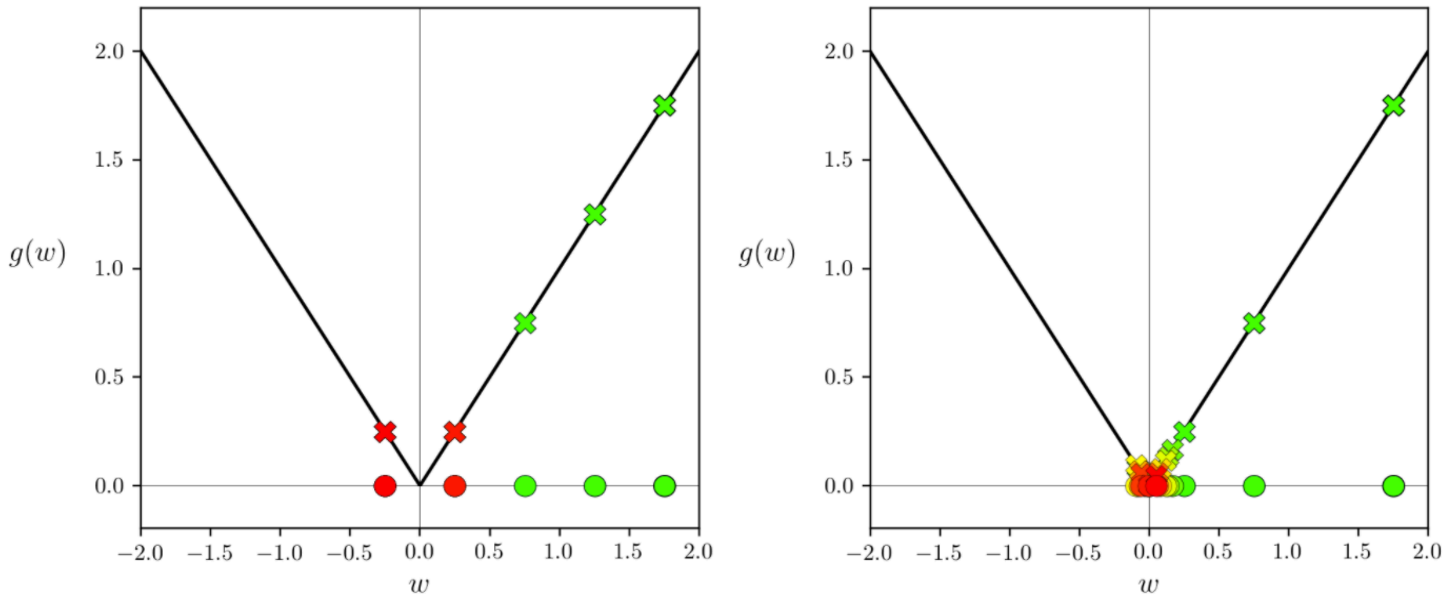
$$\frac{\mathrm{d}}{\mathrm{d}w} g(w) = \begin{cases} +1 & \text{if } w > 0 \\ -1 & \text{if } w < 0 \end{cases}$$

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

Gradient Descent

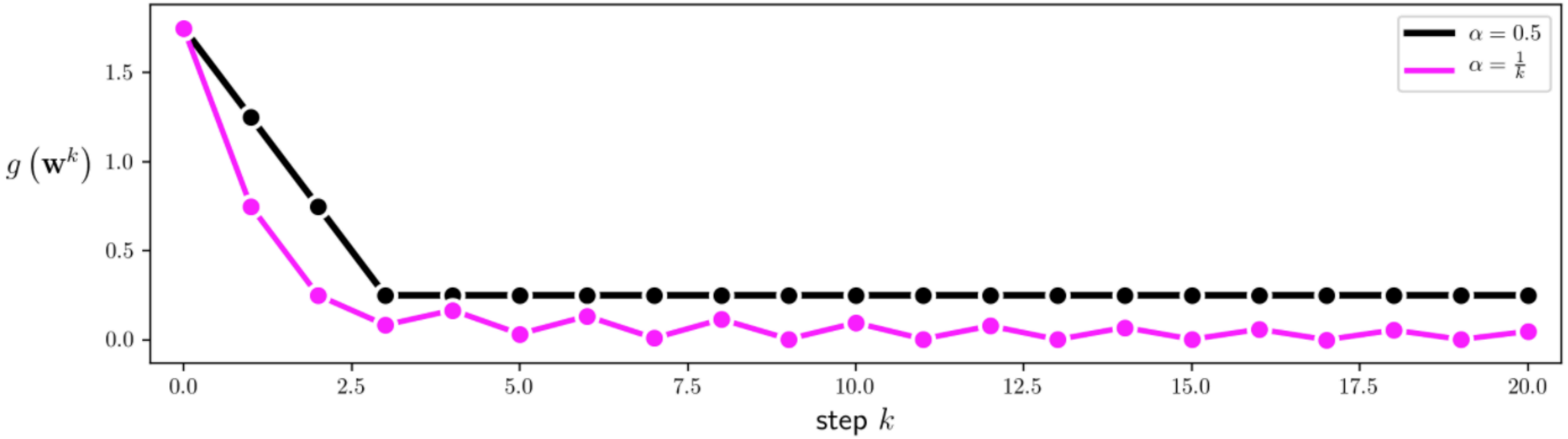
- Gradient descent path:

Fixed steplength Diminishing steplength



Gradient Descent

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



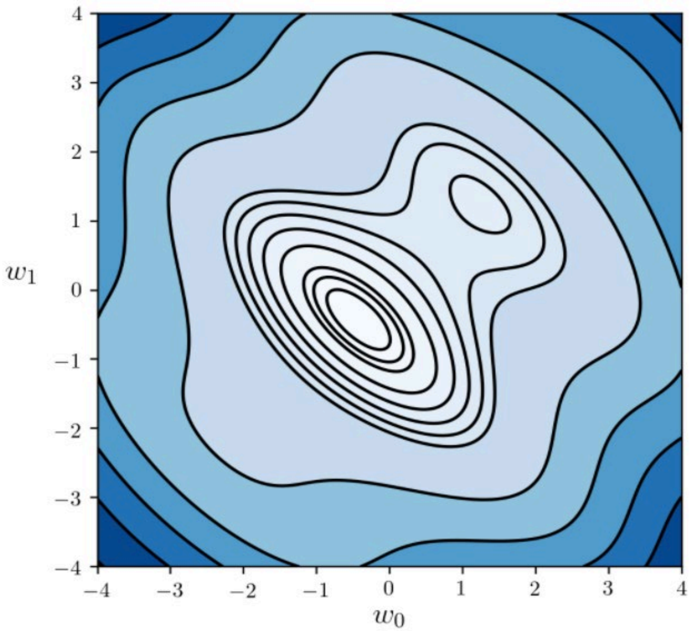
Gradient Descent

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

- Example:
  - Function:

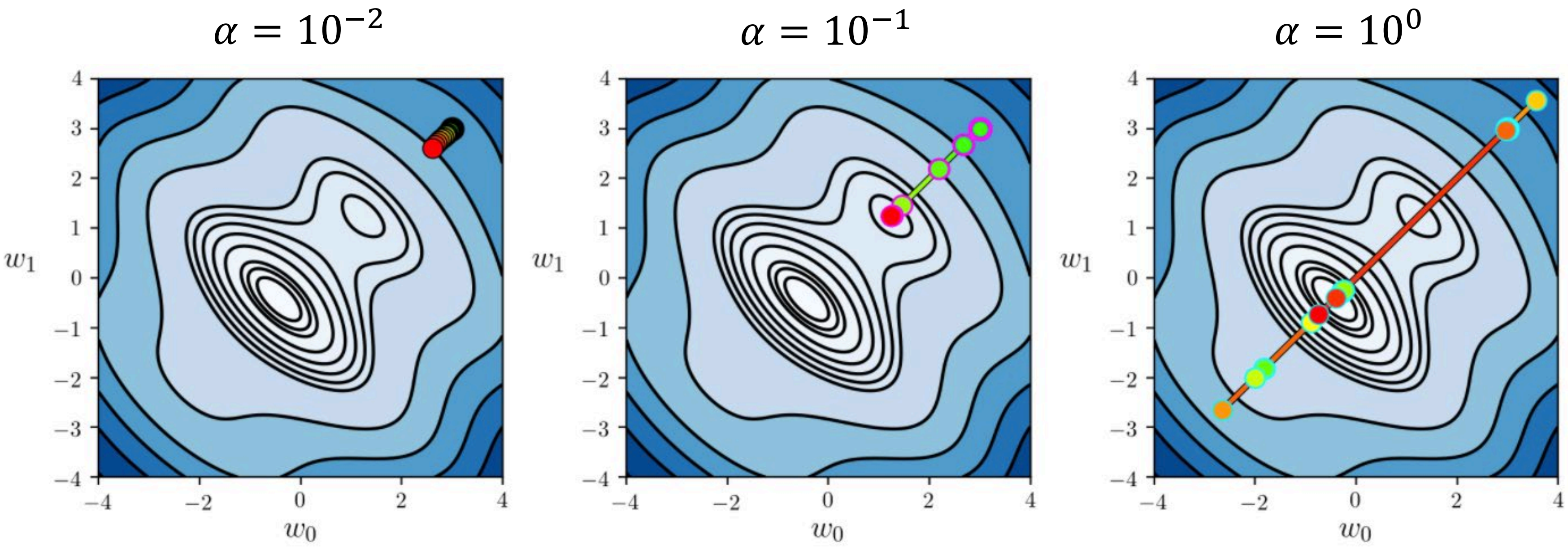
$$g(\mathbf{w}) = w_0^2 + w_1^2 + 2\sin(1.5\,(w_0 + w_1))^2 + 2$$

  - 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



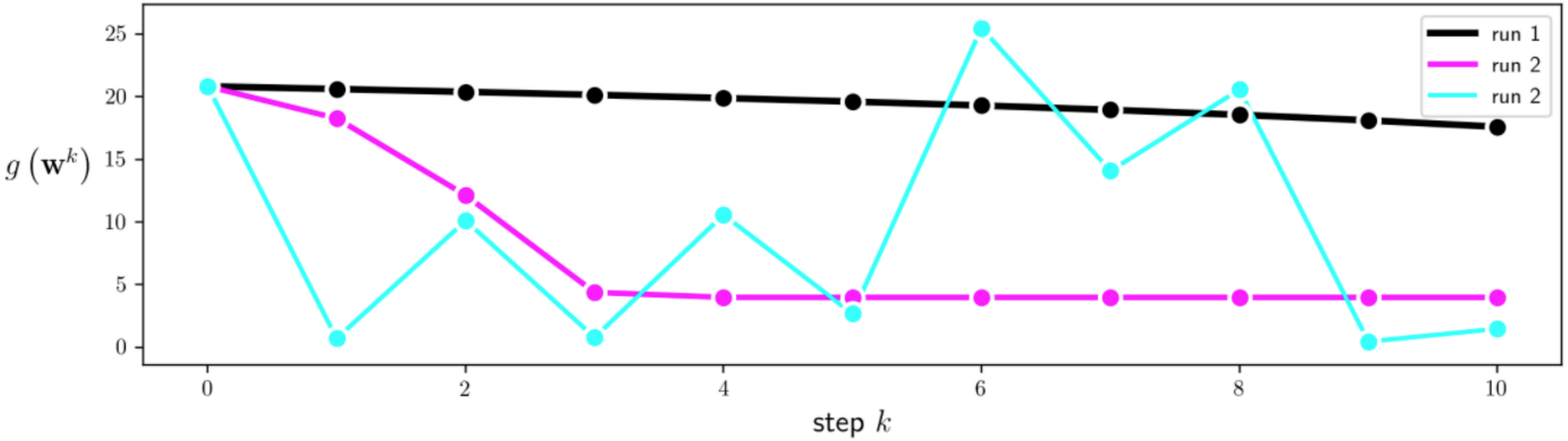
Gradient Descent

- 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.

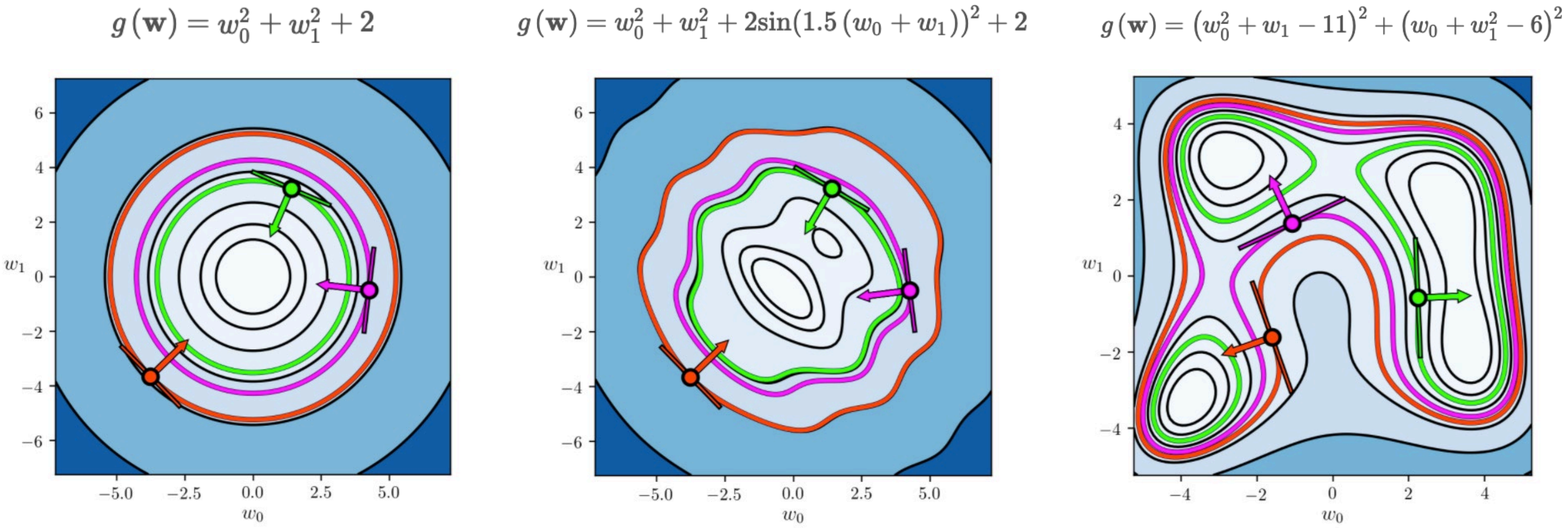
Goals for this Lecture

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

Two Natural Weaknesses of Gradient Descent

Problem 1: The 'zig-zagging' behavior of gradient descent

- The (negative) gradient direction points perpendicular to the contours of any function



Two Natural Weaknesses of Gradient Descent

- The negative gradient direction oscillate rapidly or zig-zag
- Example
  - Functions: three N=2N=2 dimensional quadratic

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

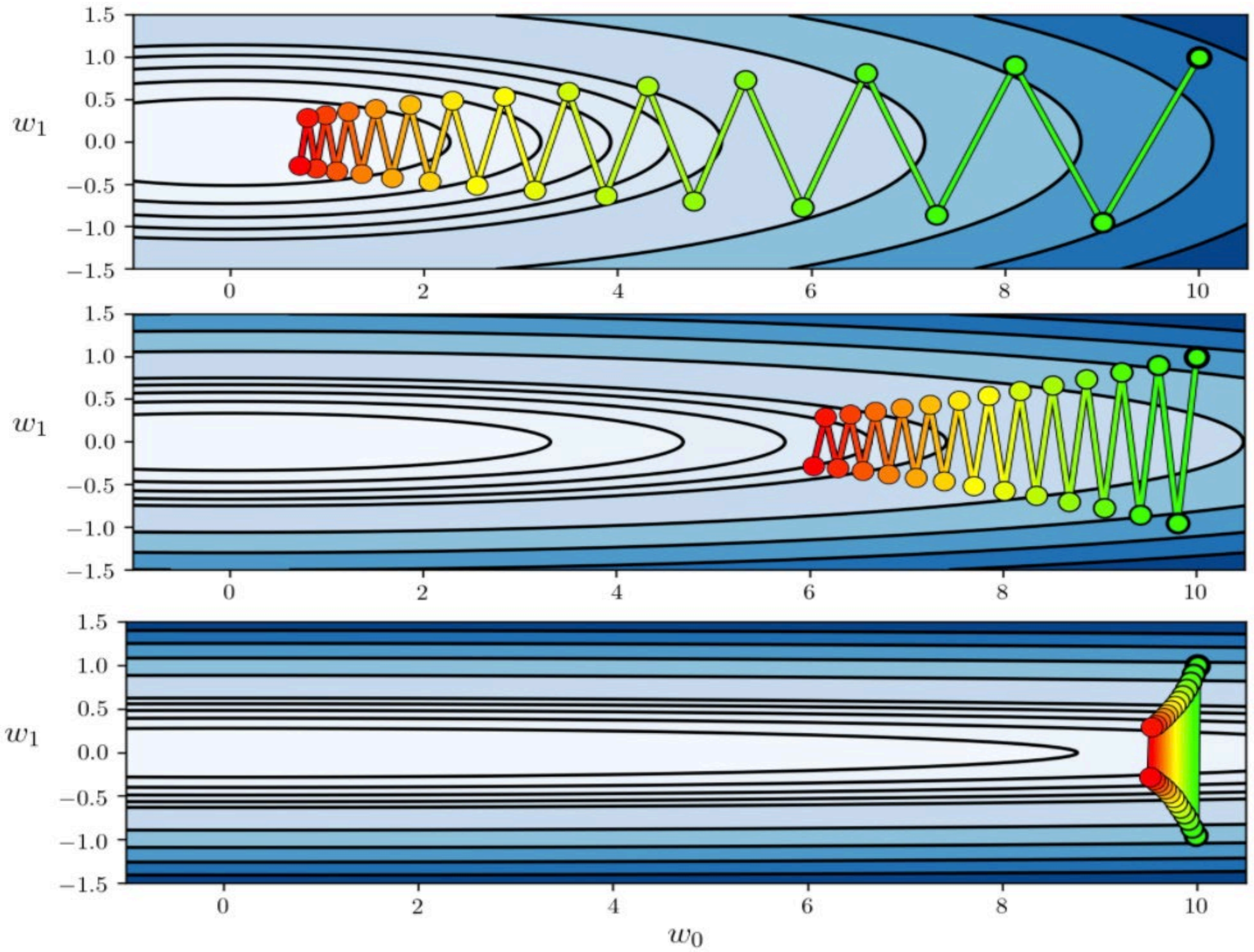
    - The first quadratic: $\mathbf{C} = \begin{bmatrix} 0.5 & 0 \\ 0 & 12 \end{bmatrix}$
    - The second quadratic: $\mathbf{C} = \begin{bmatrix} 0.1 & 0 \\ 0 & 12 \end{bmatrix}$
    - The third quadratic: $\mathbf{C} = \begin{bmatrix} 0.01 & 0 \\ 0 & 12 \end{bmatrix}$
  - Same global minimum: $\mathbf{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ where $g(\mathbf{w}) = 0$
  - Initialization: $\mathbf{w}^{\mathbf{0}} = \begin{bmatrix} 10 \\ 1 \end{bmatrix}$
  - Steplength / learning rate value: $\alpha = 0.1$

Two Natural Weaknesses of Gradient Descent

The first quadratic

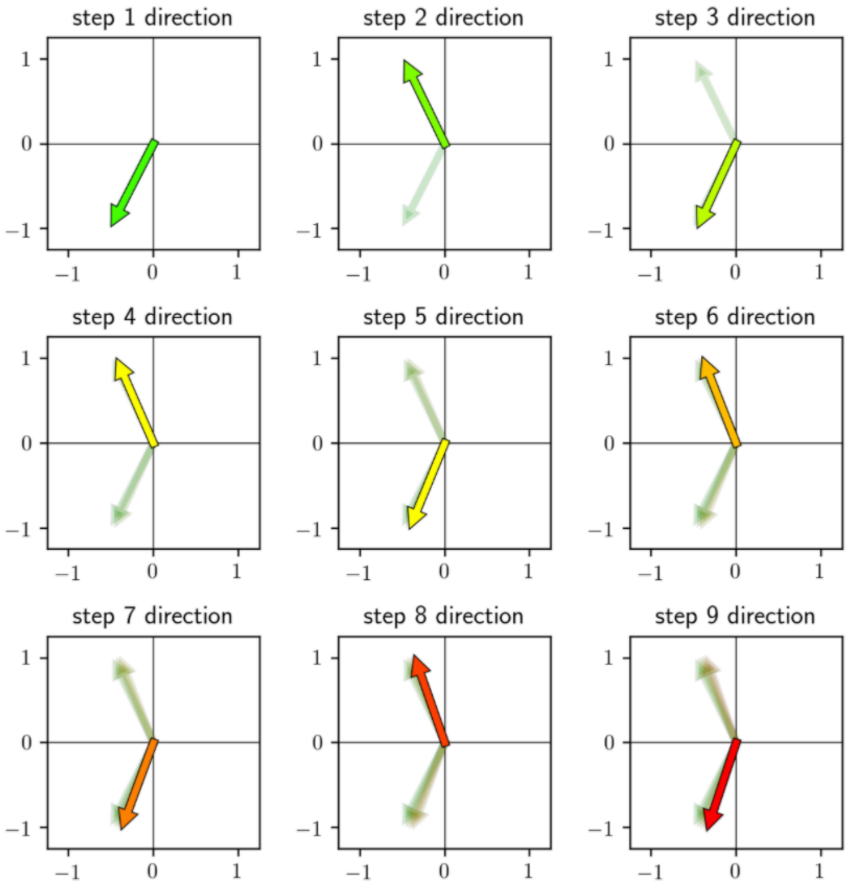
The second quadratic

The third quadratic



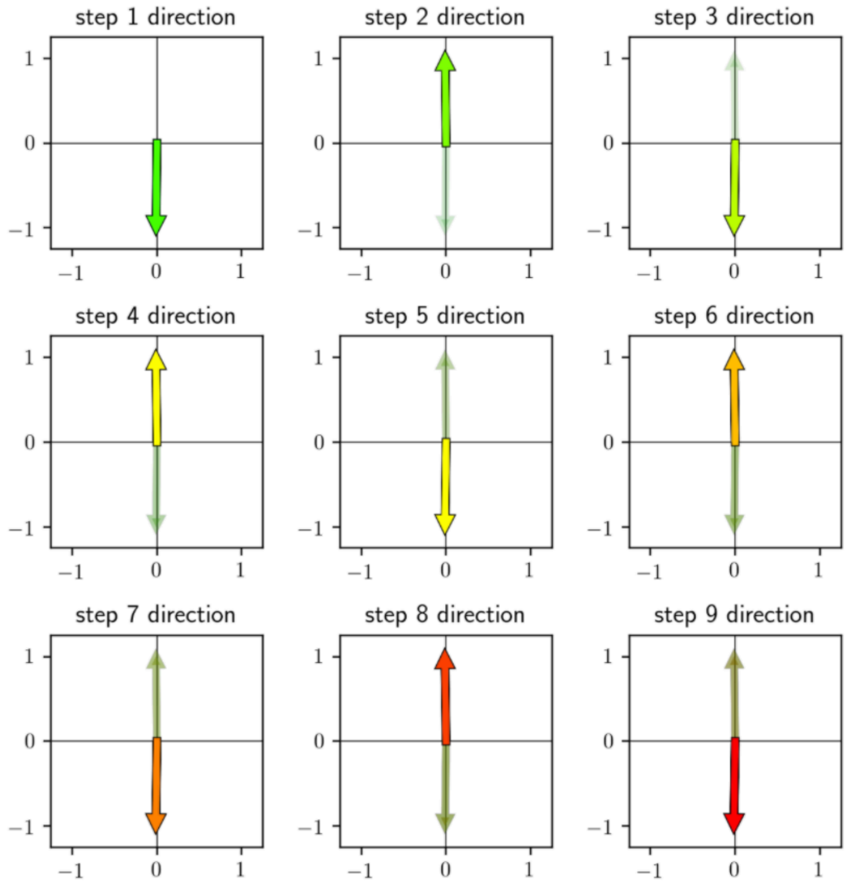
Two Natural Weaknesses of Gradient Descent

- Descent direction on the **first** quadratic



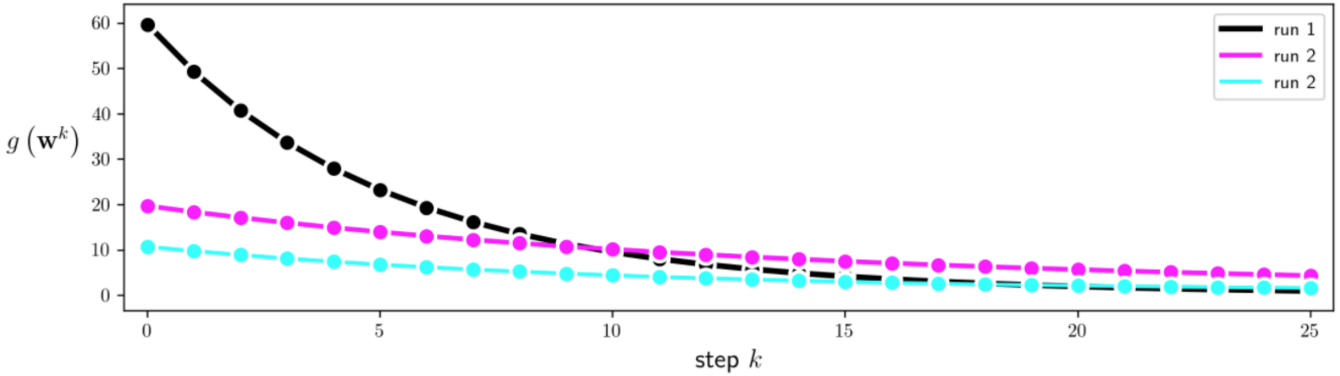
Two Natural Weaknesses of Gradient Descent

- Descent direction on the **third** quadratic

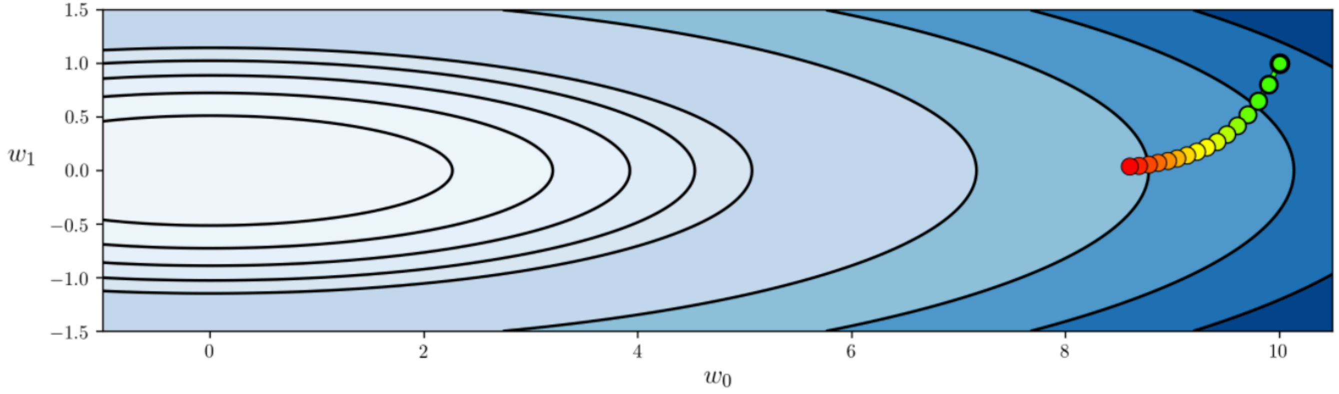


Two Natural Weaknesses of Gradient Descent

- Cost function plot



- Reducing the steplength value can ameliorate this zig-zagging behavior.
- Do not solve the underlying problem that zig-zagging produces - slow convergence



Two Natural Weaknesses of Gradient Descent

Problem 2: The slow-crawling behavior of gradient descent

- The vanishing behavior of the negative gradient magnitude near stationary points has a natural consequence for gradient descent steps - they progress very slowly, or 'crawl', near stationary points.
- Unlike zero order methods, the distance traveled during each step of gradient descent is not completely determined by the steplength/learning rate value α.

Two Natural Weaknesses of Gradient Descent

- The general local optimization step:

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

- Zero order: $\mathbf{d}^{k-1}$ is a unit length descent direction

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

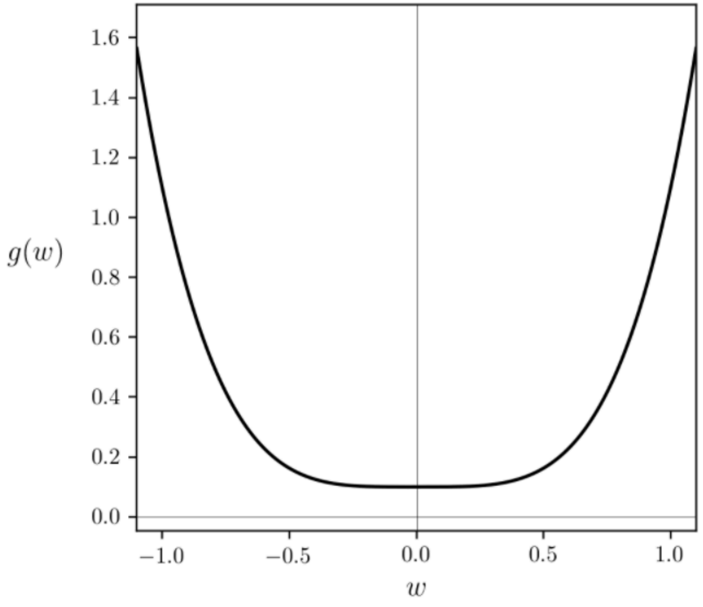
- Gradient descent: $\mathbf{d}^{k-1} = -\nabla g(\mathbf{w}^{k-1})$

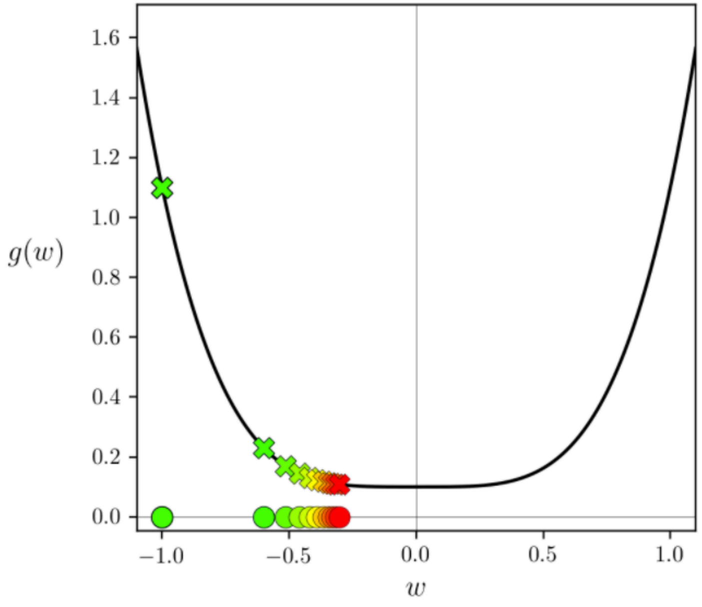
$$\left\|\mathbf{w}^k - \mathbf{w}^{k-1}\right\|_2 = \left\|\left(\mathbf{w}^{k-1} - \alpha \nabla g\left(\mathbf{w}^{k-1}\right)\right) - \mathbf{w}^{k-1}\right\|_2 = \alpha \left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2$$

Two Natural Weaknesses of Gradient Descent

- Example 1: Slow-crawling behavior of gradient descent near the minimum of a function
  - Function:
  - Minimum: $w = 0$
  - Steplength: $\alpha = 0.1$

$$g(w) = w^4 + 0.1$$



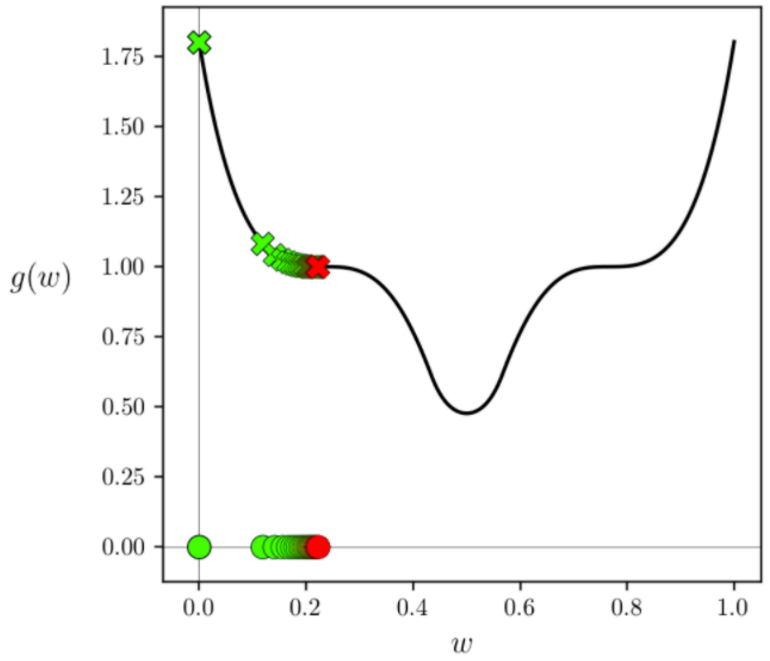


Two Natural Weaknesses of Gradient Descent

- Example 2: Slow-crawling behavior of gradient descent near saddle points
  - Function:

$$g(w) = \text{maximum}(0, (3w - 2.3)^3 + 1)^2 + \text{maximum}(0, (-3w + 0.7)^3 + 1)^2$$

  - Minimum: $w = \frac{1}{2}$
  - Saddle points:
    - $w = \frac{7}{30}$
    - $w = \frac{23}{30}$
  - Gradient descent: 50 steps
  - Steplength: $\alpha = 0.01$
  - Initialization: $w = 0$

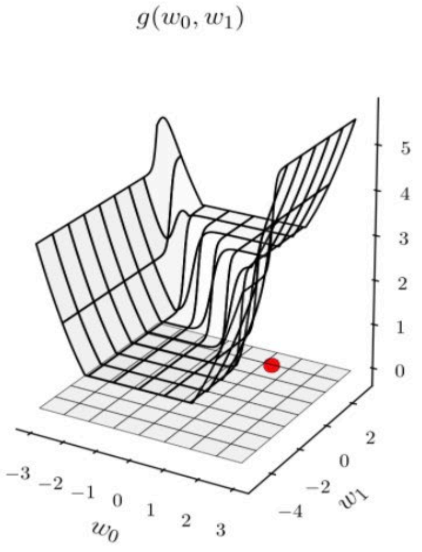


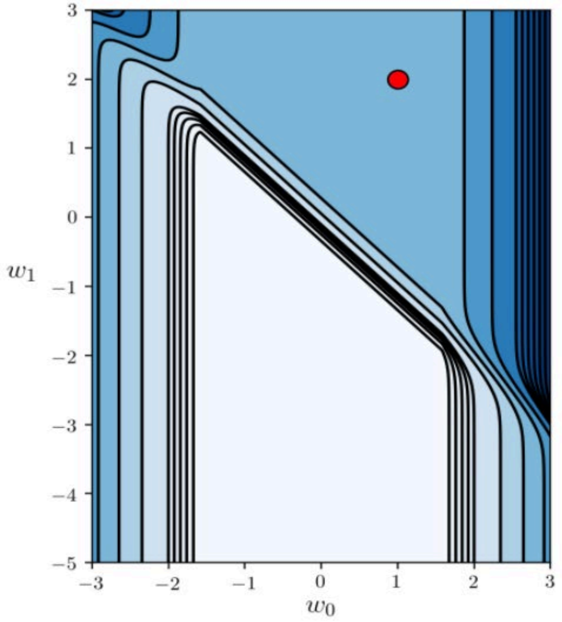
Two Natural Weaknesses of Gradient Descent

- Example 3: Slow-crawling behavior of gradient descent in large flat regions of a function
  - Function:

$$g(w_0, w_1) = \tanh(4w_0 + 4w_1) + \max(1, 0.4w_0^2) + 1$$

  - Initialization: $w^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
  - 1000 steps of gradient descent with a steplength $\alpha = 0.1$





Goals for this Lecture

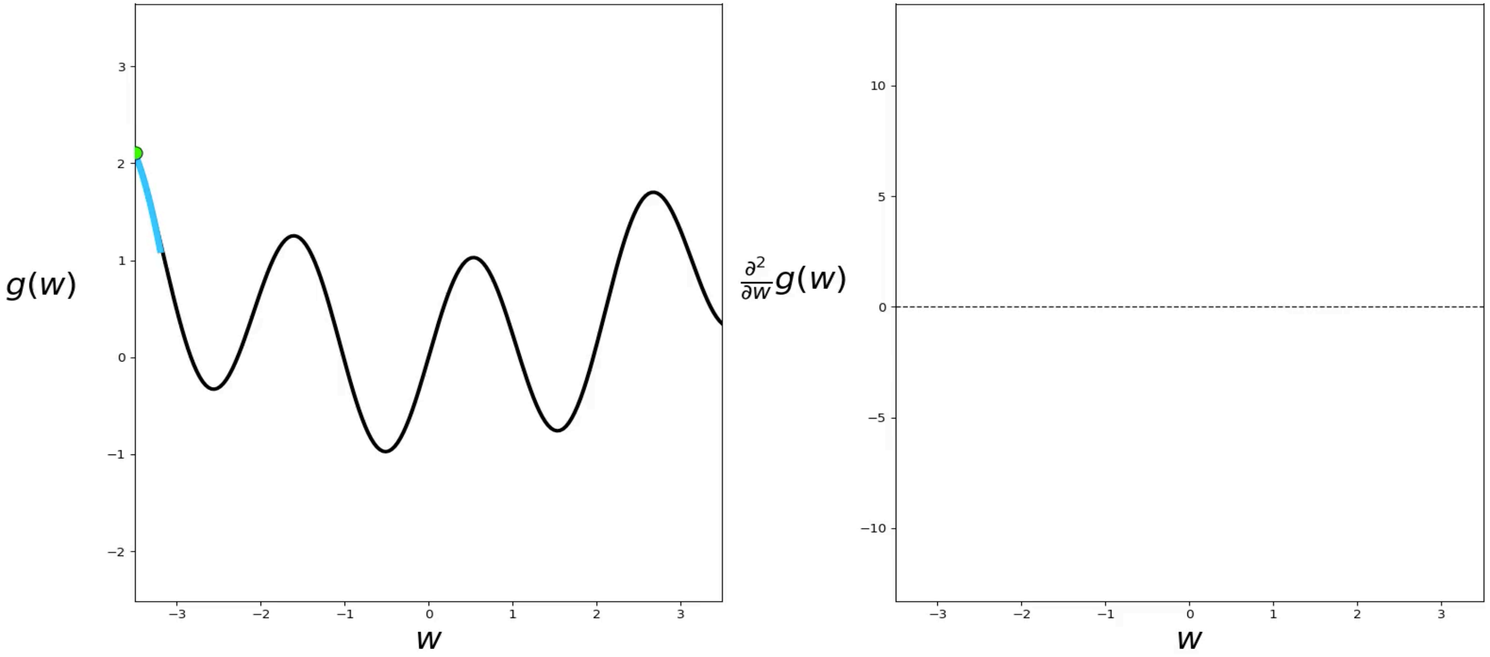
- First Order Optimization – The Gradient Function
- Understand Gradient Descent
- Understanding Limitations to Gradient Descent
- Second Order Optimization – Convexity/Concavity
- Newton's Method for Descent

The Second-Order Optimality Condition

Curvature and single-input functions

- The second order Taylor series approximation of function

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



The Second-Order Optimality Condition

- The second order approximation appears to match the local convexity/concavity of the underlying function near the point on which it is defined.
  - If at this point the function appears to be convex locally, the second order approximation is too convex and upward facing.
  - If the point is on a part of the function where it is facing downward or concave, the second order approximation is also concave and facing downward.
- The second order Taylor Series is a quadratic built to match a function locally.

The Second-Order Optimality Condition

- Quadratic functions are easy to determine convex or concave
- A general single input quadratic

$$g(w) = a + bw + cw^2$$

  - $c > 0$: convex
  - $c < 0$: concave
  - $c = 0$: both convex and concave (a line)
- The second order Taylor Series $h(w)$ of a single input function $g(w)$ at a point w_0 is:

$$h(w) = g(w^0) + \left(\frac{\mathrm{d}}{\mathrm{d}w} g(w^0)\right)(w - w^0) + \frac{1}{2}\left(\frac{\mathrm{d}^2}{\mathrm{d}w^2} g(w^0)\right)(w - w^0)^2$$

The Second-Order Optimality Condition

$$c = \frac{1}{2} \frac{\mathrm{d}^2}{\mathrm{d}w^2} g(w^0)$$

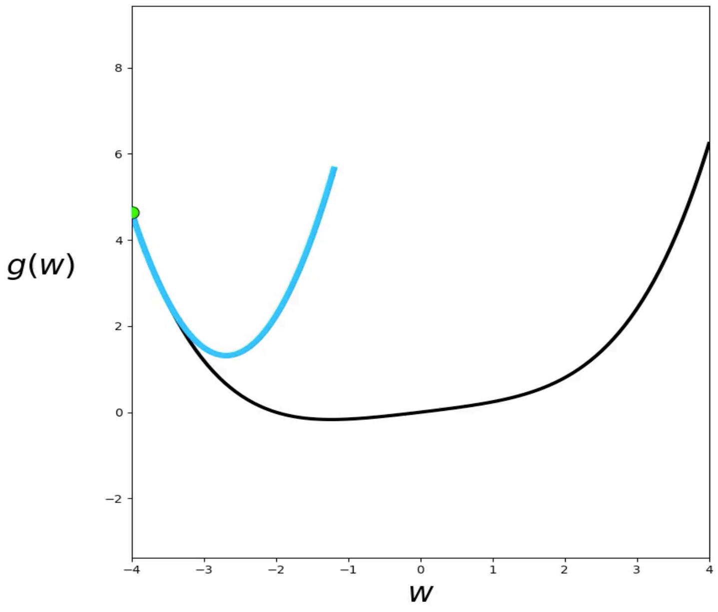
- $\frac{d^2}{dw^2} g(w^0) \geq 0$: convex at w^0
- $\frac{d^2}{dw^2} g(w^0) \leq 0$: concave at w^0

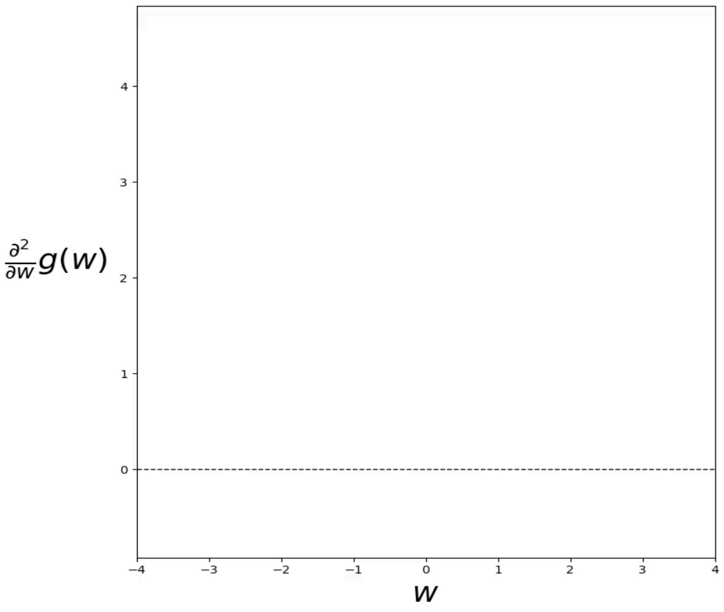
- A function g is convex if it is convex at each of its input points. ($\frac{d^2}{dw^2} g(w^0) \geq 0$ everywhere)
- A function g is concave if it is convex at each of its input points. ($\frac{d^2}{dw^2} g(w^0) \leq 0$ everywhere)

The Second-Order Optimality Condition

- Example: single-input plot
  - Function:

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





The Second-Order Optimality Condition

Curvature and multi-input functions

- The general multi-input quadratic function

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

- The convexity/concavity is determined by the eigenvalues of $\mathbf{C}$
  - The quadratic is convex along its n^{th} input iif its n^{th} eigenvalue $d_n \geq 0$
  - The quadratic is convex along its n^{th} input iif its n^{th} eigenvalue $d_n \leq 0$

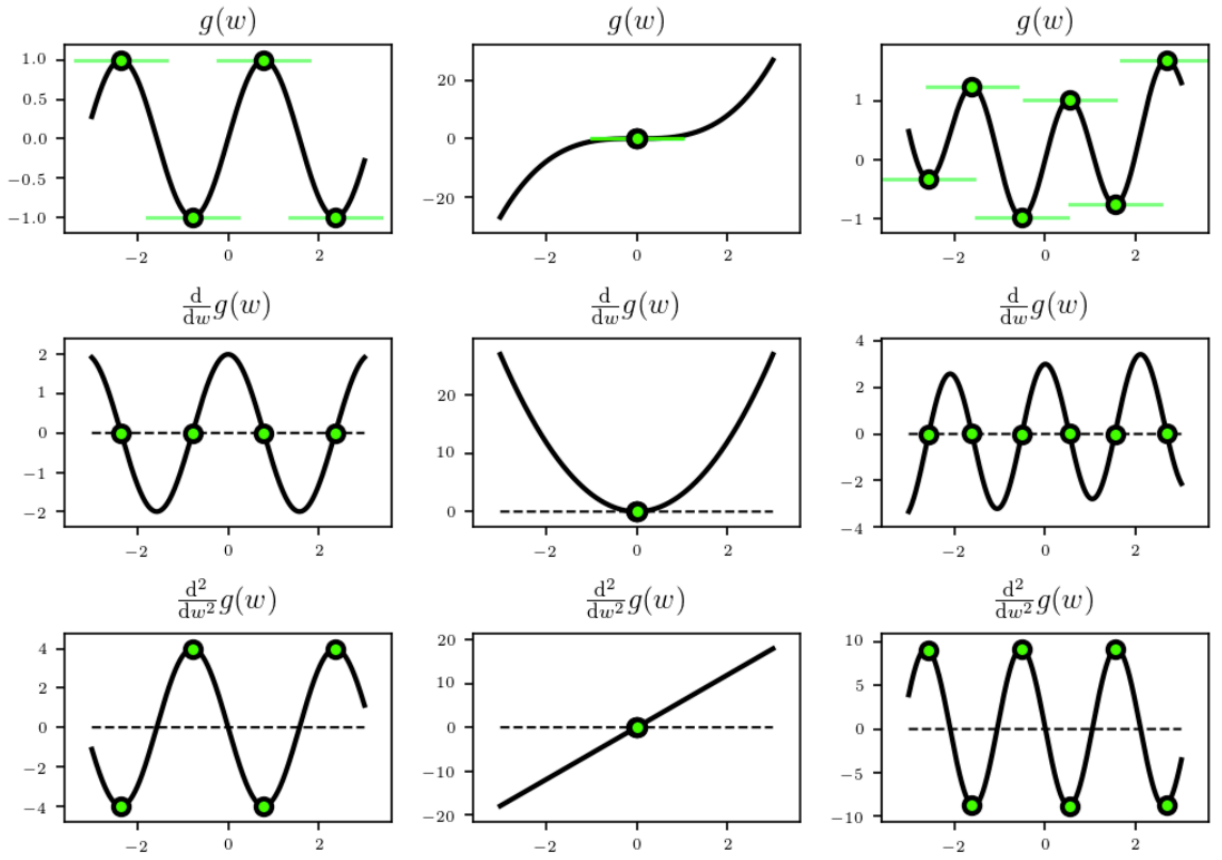
The Second-Order Optimality Condition

- Convexity/concavity at $\mathbf{w}^0$
  - g is convex at $\mathbf{w}^0$ iif the second order Taylor Series approximation is convex in every one of its input dimension, i.e., $\nabla^2 g(\mathbf{w}^0)$ has no negative eigenvalues.
  - g is concave at $\mathbf{w}^0$ iif the second order Taylor Series approximation is concave in every one of its input dimension, i.e., $\nabla^2 g(\mathbf{w}^0)$ has no positive eigenvalues.
- Convex/concave function
  - g is a *convex function* if it is convex everywhere, or if $\nabla^2 g(\mathbf{w}^0)$ has all nonnegative eigenvalues at every input.
  - g is a *concave function* if it is concave everywhere, or if $\nabla^2 g(\mathbf{w}^0)$ has all non-positive eigenvalues at every input.

The Second-Order Optimality Condition

The second order condition

- Comparison of zero, first, second order



The Second-Order Optimality Condition

- Single-input functions
  - Local/global minimum: $\frac{\partial^2}{\partial w^2} g(w) > 0$
  - Local/global maximum: $\frac{\partial^2}{\partial w^2} g(w) < 0$
  - A saddle point: $\frac{\partial^2}{\partial w^2} g(w) = 0$ and $\frac{\partial^2}{\partial w^2} g(w)$ changes sign at w.

- Multi-input functions
  - Local minimum: all eigenvalues of $\nabla^2 g(\mathbf{w}^0)$ are positive
  - Local maximum: all eigenvalues of $\nabla^2 g(\mathbf{w}^0)$ are negative
  - A saddle points: all eigenvalues of $\nabla^2 g(\mathbf{w}^0)$ are mixed (have both positive and negative.

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

Newton's Method

- Newton’s method: a local optimization algorithm produced by repeatedly taking steps that are stationary points of the second order Taylor series approximations to a function.

- Method:
  - At k^{th} step move to the stationary point of the quadratic approximation generated at the previous step $\mathbf{w}^{k-1}$:
  - A stationary p $h(\mathbf{w}) = g(\mathbf{w}^{k-1}) + \nabla g(\mathbf{w}^{k-1})^T(\mathbf{w} - \mathbf{w}^{k-1}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^{k-1})^T \nabla^2 g\left(\mathbf{w}^{k-1}\right)\left(\mathbf{w} - \mathbf{w}^{k-1}\right)$

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \left(\nabla^2 g(\mathbf{w}^{k-1})\right)^{-1} \nabla g(\mathbf{w}^{k-1})$$

Newton's Method

– For single input functions:

$$w^k = w^{k-1} - \frac{\frac{\mathrm{d}}{\mathrm{d}w} g(w^{k-1})}{\frac{\mathrm{d}^2}{\mathrm{d}w^2} g(w^{k-1})}$$

– This local optimization fits:

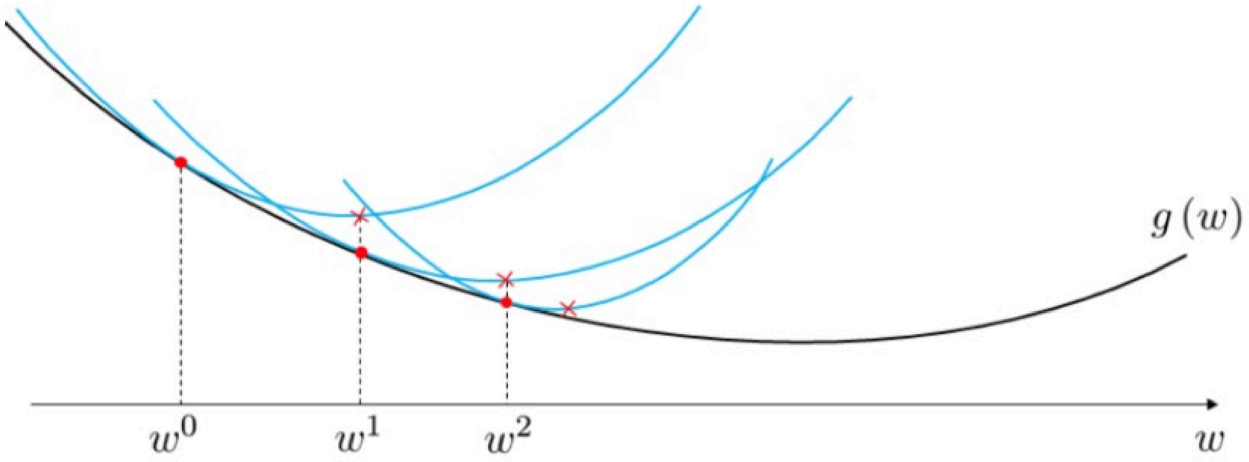
$$\mathbf{w}^k = \mathbf{w}^{k-1} + \alpha \mathbf{d}^k$$

where

$$\mathbf{d}^k = -\left(\nabla^2 g(\mathbf{w}^{k-1})\right)^{-1} \nabla g(\mathbf{w}^{k-1})$$

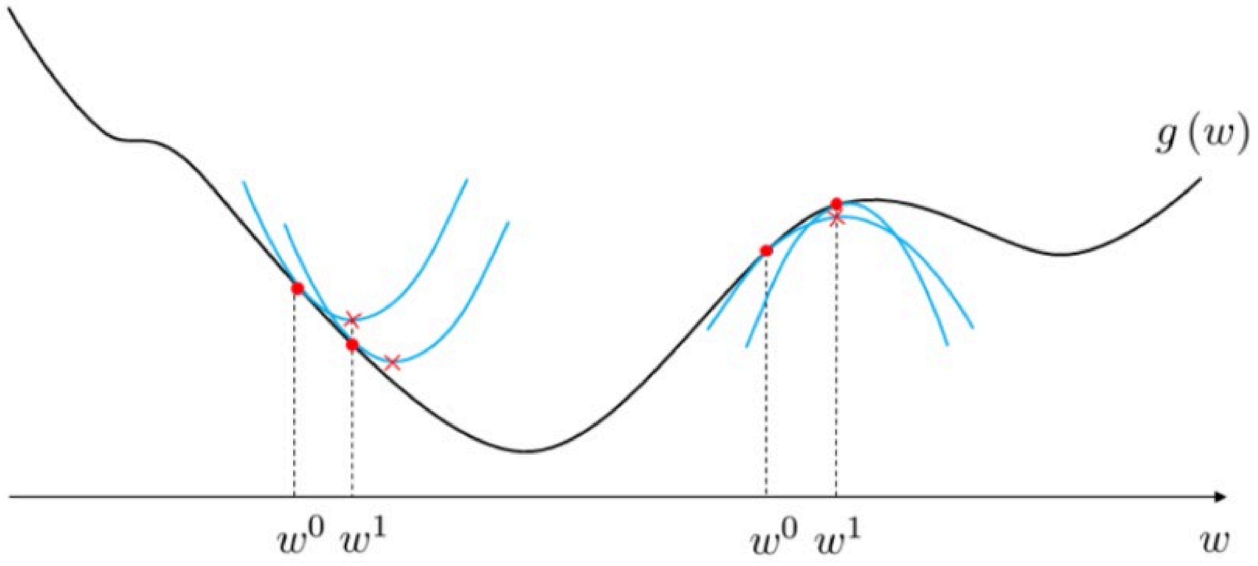
Newton's Method

- Example 1: Newton's method on a convex function
  - The quadratic approximations are themselves always convex
  - The stationary points are minima
  - The sequence leads to a minimum of the original function



Newton's Method

- Example 2: Newton's method on a non-convex function
  - The quadratic approximations can be concave or convex
  - Lead the algorithm to possibly converge to a maximum.



Newton's Method

Ensuring numerical stability

- The single-input Newton step

$$w^k = w^{k-1} - \frac{\frac{\mathrm{d}}{\mathrm{d}w} g(w^{k-1})}{\frac{\mathrm{d}^2}{\mathrm{d}w^2} g(w^{k-1})}$$

  – Near flat portions of a function, both $\frac{\mathrm{d}}{\mathrm{d}w} g(w^{k-1})$ and $\frac{\mathrm{d}^2}{\mathrm{d}w^2} g(w^{k-1})$ can be nearly zero valued.

- Regularized Newton: add a very small positive value ϵ to the second derivative:

$$w^k = w^{k-1} - \frac{\frac{\mathrm{d}}{\mathrm{d}w} g(w^{k-1})}{\frac{\mathrm{d}^2}{\mathrm{d}w^2} g(w^{k-1}) + \epsilon}$$

Newton's Method

- Multi-input functions
  - Regularized Newton: add $\epsilon\mathbf{I}_{N\times N}$, a $N \times N$ identity matrix scaled by a small positive ϵ value, to the Hessian matrix:

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \left(\nabla^2 g(\mathbf{w}^{k-1}) + \epsilon\mathbf{I}_{N\times N}\right)^{-1} \nabla g(\mathbf{w}^{k-1})$$

  - $\left(\nabla^2 g(\mathbf{w}^{k-1}) + \epsilon\mathbf{I}_{N\times N}\right)$hen

$$\left(\nabla^2 g(\mathbf{w}^{k-1}) + \epsilon\mathbf{I}_{N\times N}\right)\mathbf{w} = \left(\nabla^2 g(\mathbf{w}^{k-1}) + \epsilon\mathbf{I}_{N\times N}\right)\mathbf{w}^{k-1} - \nabla g(\mathbf{w}^{k-1})$$

Newton's Method

- Newton's method:

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

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

3: $\mathbf{w}^k = \mathbf{w}^{k-1} - \left(\nabla^2 g(\mathbf{w}^{k-1}) + \epsilon \mathbf{I}_{N\times N}\right)^{-1} \nabla g(\mathbf{w}^{k-1})$

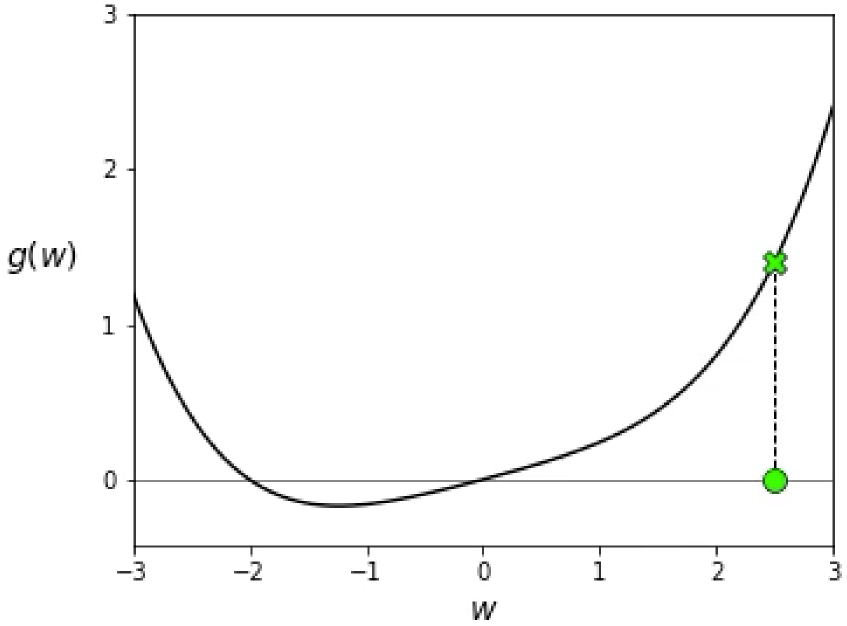
4: **output:** history of weights $\left\{\mathbf{w}^k\right\}_{k=0}^{K}$ and corresponding function evaluations $\left\{g\left(\mathbf{w}^k\right)\right\}_{k=0}^{K}$

Newton's Method

- Example 1: Newton's method applied to a convex single-input function
  - Function:

$$g(w) = \frac{1}{50}\left(w^4 + w^2 + 10w\right) + 0.5$$

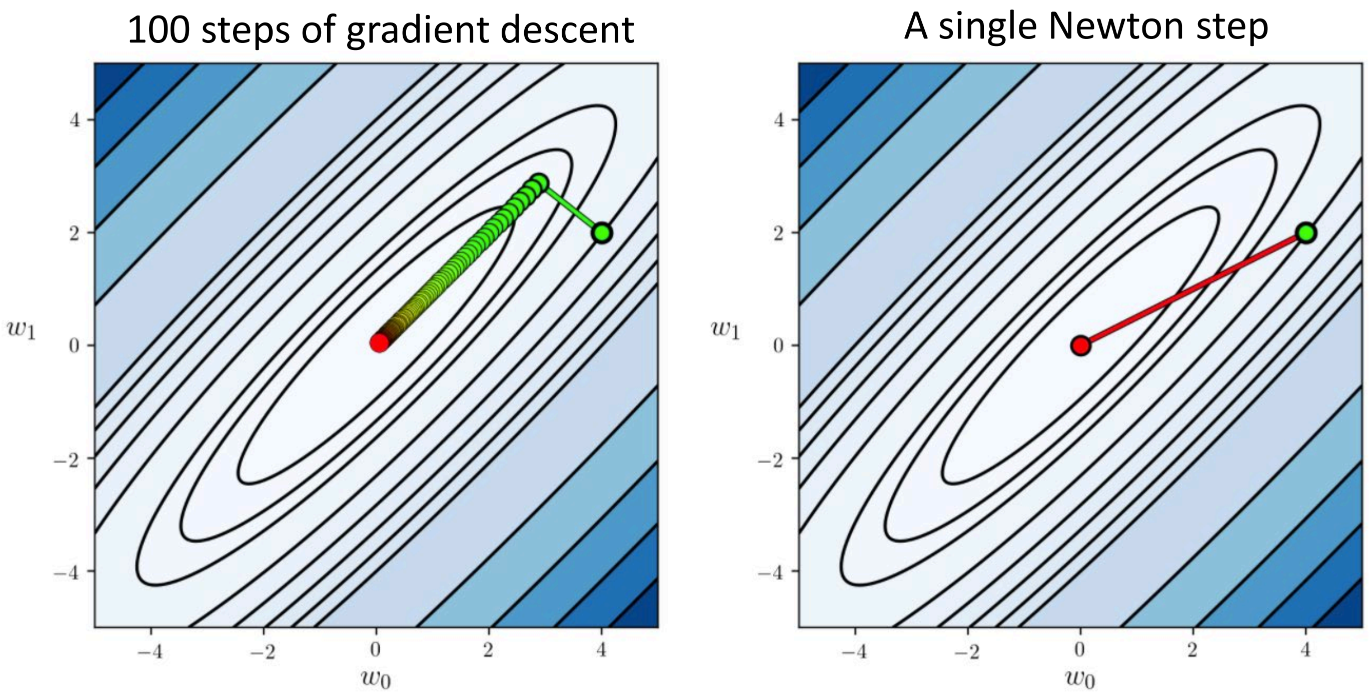
  - Initialization: $w = 2.5$



Newton's Method

- Example 2: Minimizing a quadratic function with a single Newton step
  - Function:

$$g(w_1, w_2) = 0.26(w_1^2 + w_2^2) - 0.48w_1w_2$$



Limitation of Newton’s Method

A Newton's method step requires far more in terms of storage and computation than a first order step

- Requires the storage and computation of not just a gradient but an entire $N \times N$ Hessian matrix of second derivative information.
- In machine learning, this can easily have tens of thousands to hundreds of thousands or even hundreds of millions of inputs, making the complete storage of an associated Hessian impossible.

Takeaways

- Understand Gradient Descent
- Understanding Selection of Step Size for Gradient Descent
- Understanding Newton's Method
- Be able to compute Gradients
- Be able to evaluate function optimality based upon Gradient Descent
- **Next Time: Logistic Regression**