# Optimization

CMPUT 296: Basics of Machine Learning

Textbook §4.1-4.4

# Logistics

**Updates:**

- Delay Assignment 2 deadline by 1 week
    - Now Friday, March 19
- Delay Midterm by 1 week
    - Now Thursday, March 25
- Thought Question 3 due sooner, but only for Chapter 7 and 8
    - Now due Monday, March 15 instead of Thursday March 25

**Lab this Week:**

- Q&A for Assignment 2

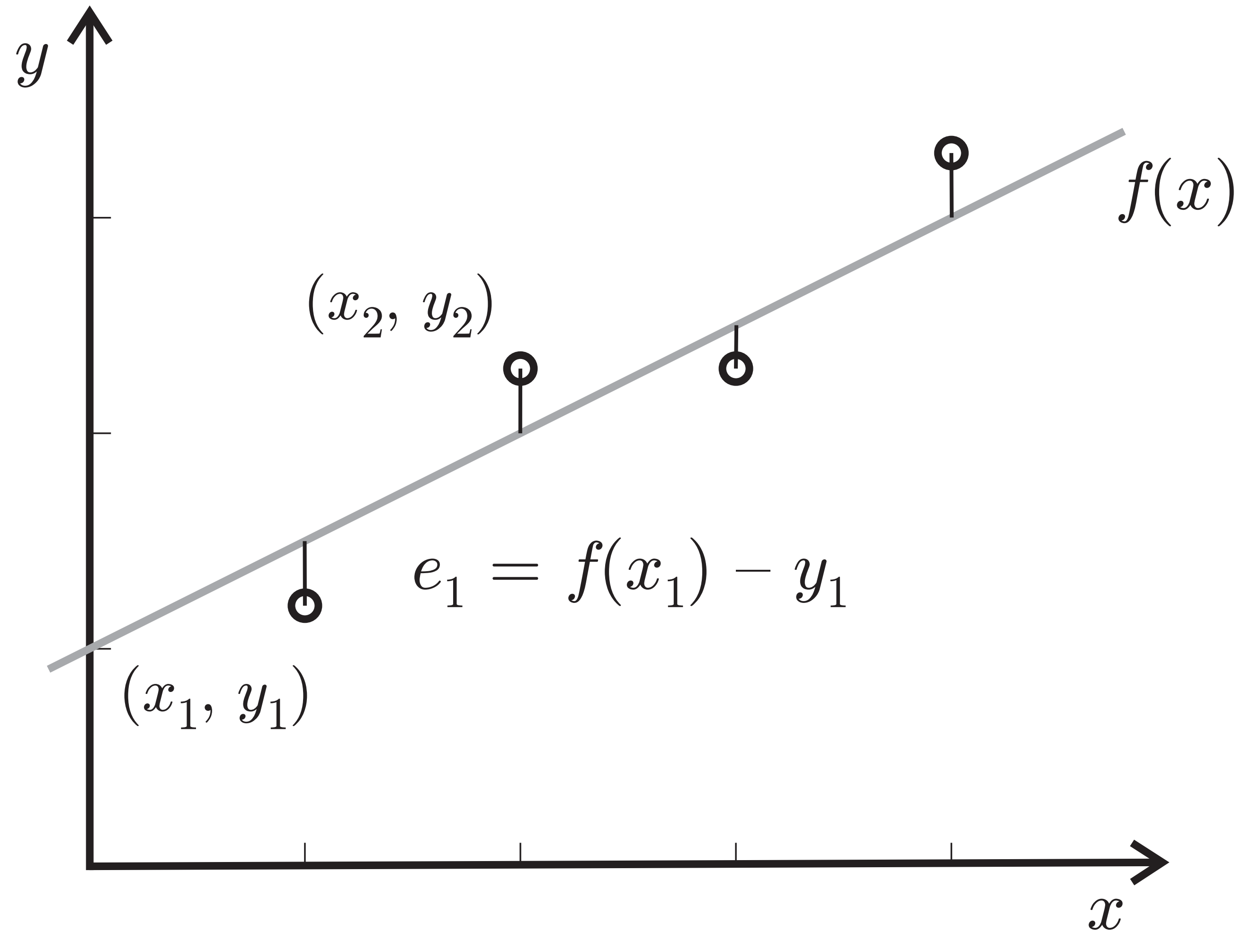
# Optimization

We often want to find the argument $w^*$ that **minimizes** an **objective function** $c$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} c(\mathbf{w})$$

**Example:** Using linear regression to fit a dataset $\{(x_i, y_i)\}_{i=1}^{n}$

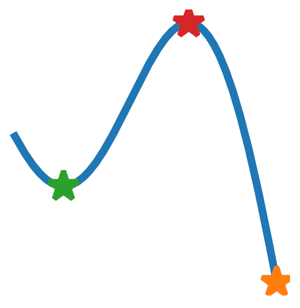
- Estimate the targets by $\hat{y} = f(x) = w_0 + w_1 x$
- Each vector $\mathbf{w}$ specifies a particular $f$
- Objective is the **total error** $c(\mathbf{w}) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$



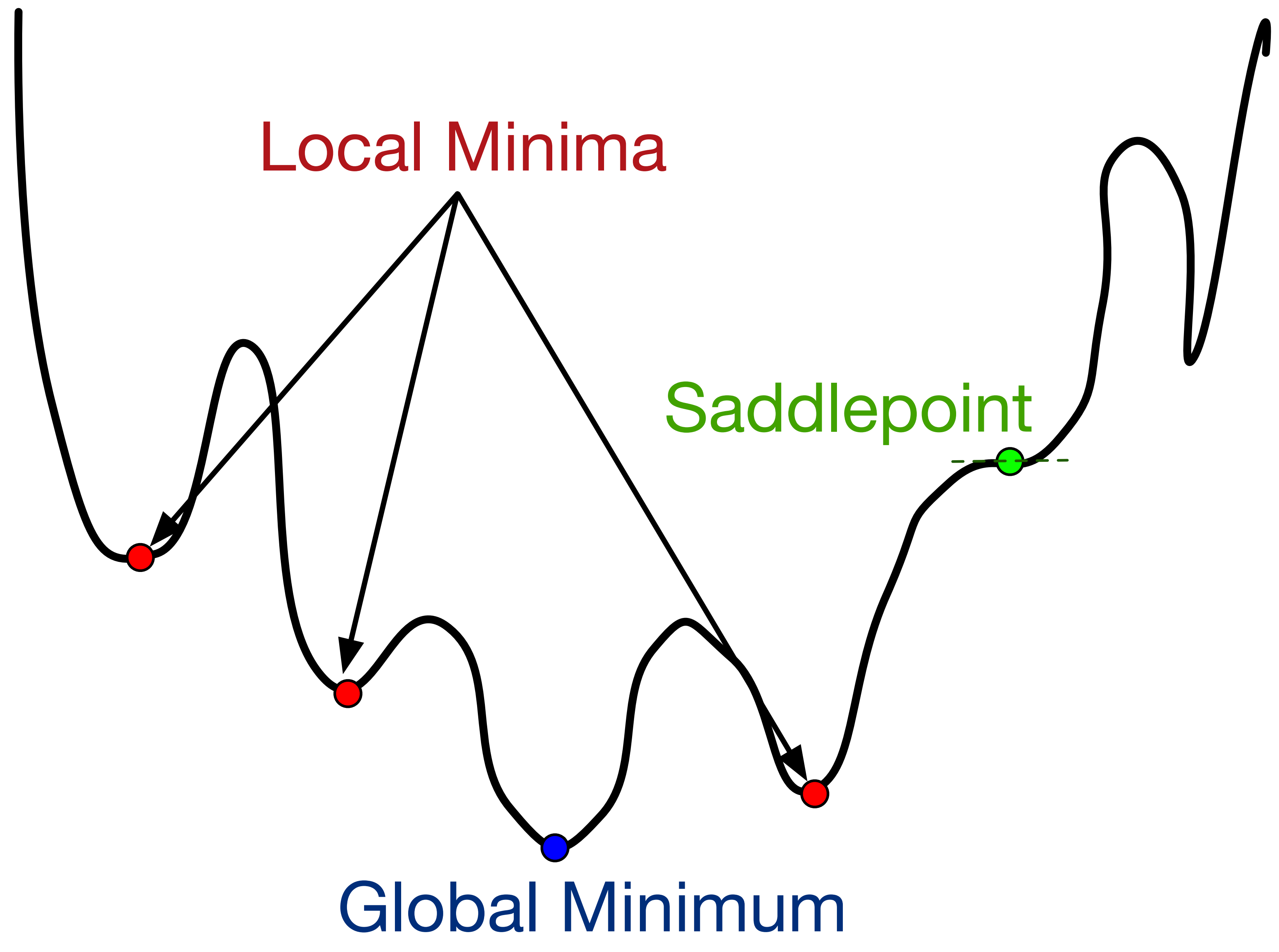
# Stationary Points

- Recall that every minimum of an everywhere-differentiable function $c(w)$ must* occur at a **stationary point**: A point at which $c'(w) = 0$

✳ **Question:** What is the exception?



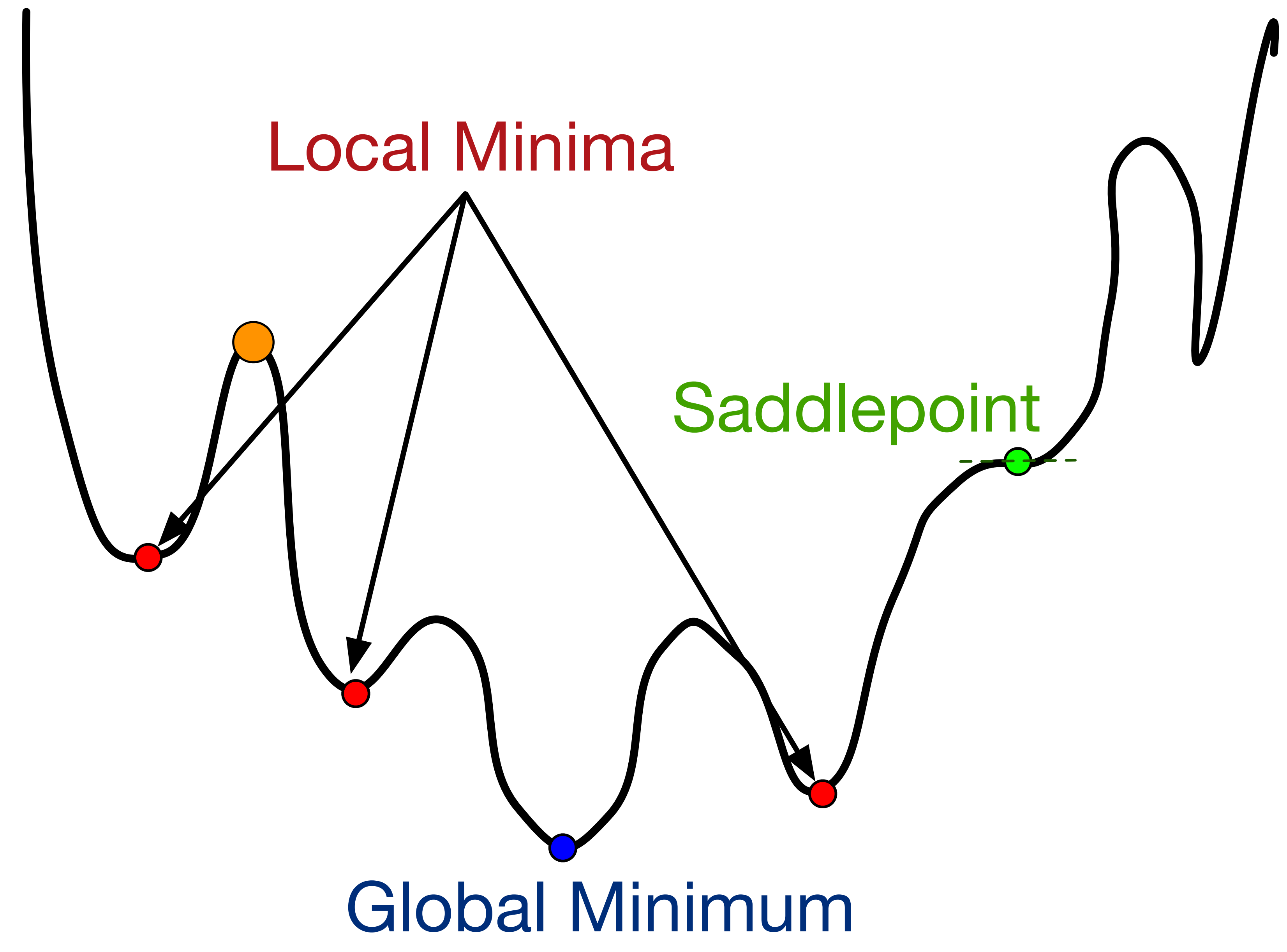
- However, not every stationary point is a minimum



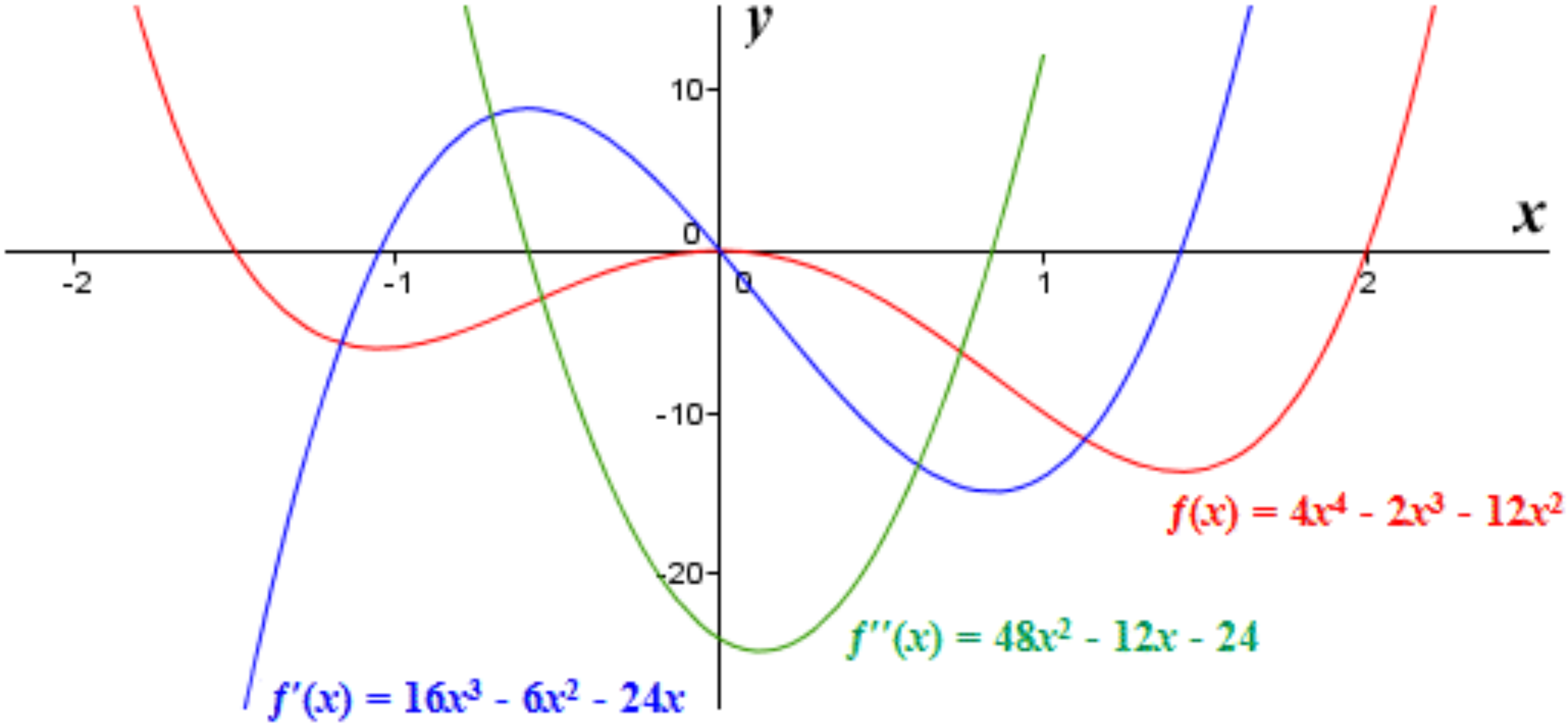
- Every stationary point is either:
  - A **local minimum**
  - A **local maximum**
  - A **saddlepoint**
- The **global minimum** is either a local minimum, or a boundary point

# Identifying the type of the stationary point

- If function curved upwards (**convex**) locally, then **local minimum**
- If function curved downwards (**concave**) locally, then **local maximum**
- If function **flat** locally, then **saddlepoint**
- Locally, cannot distinguish between local min and global min (its a global property of the surface)



# Second derivative reflects curvature



# Numerical Optimization

- So a simple recipe for optimizing a function is to find its stationary points; one of those must be the minimum (as long as domain is unbounded)
    - **Question:** Why don't we always just do that?
- We will *almost never* be able to **analytically** compute the minimum of the functions that we want to optimize
    - ✴ (Linear regression is an important exception)
- Instead, we must try to find the minimum **numerically**
- Main techniques: First-order and second-order **gradient descent**

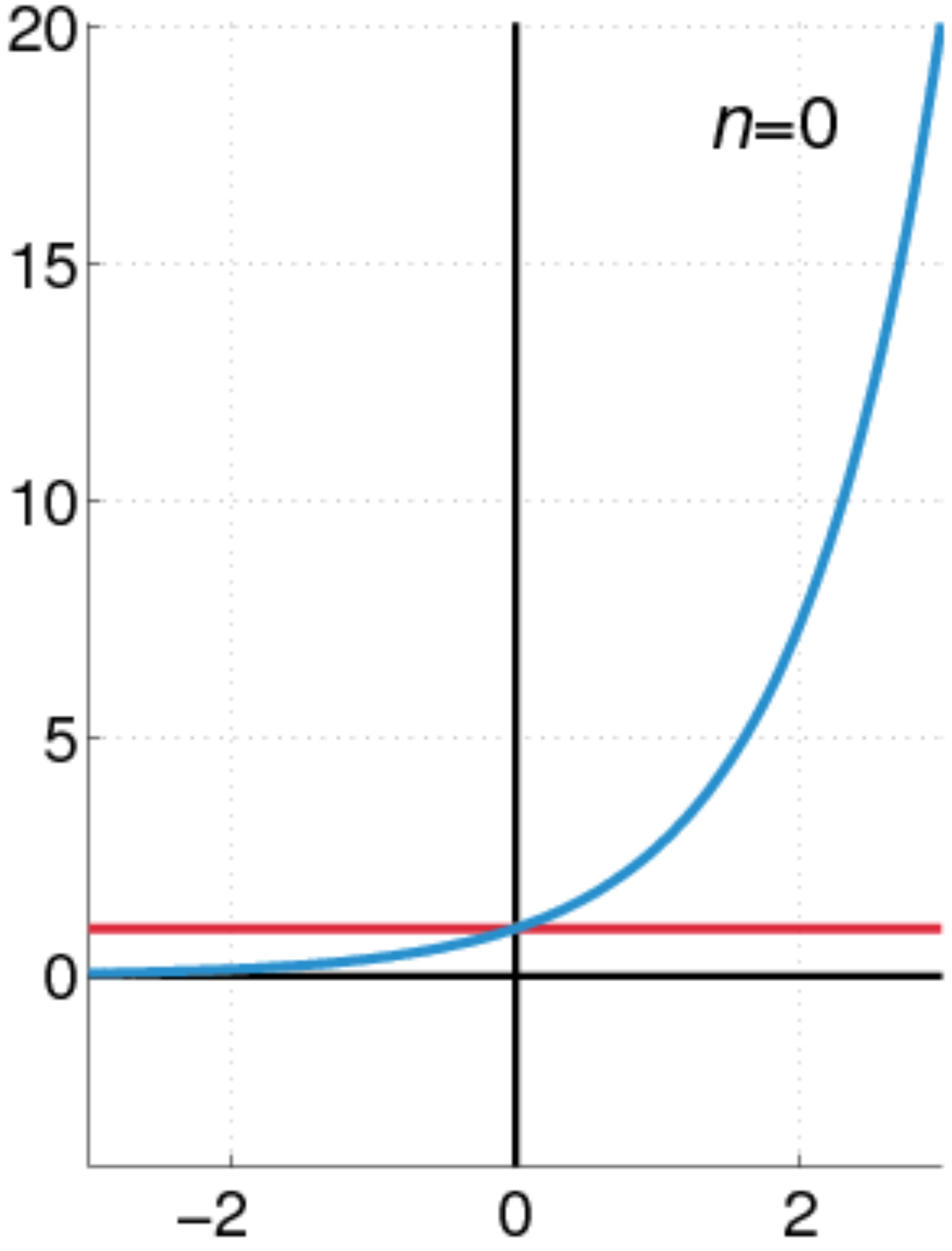
# Taylor Series

**Definition:** A **Taylor series** is a way of approximating a function $c$ in a small neighbourhood around a point $a$:

$$c(w) \approx c(a) + c'(a)(w-a) + \frac{c''(a)}{2}(w-a)^2 + \cdots + \frac{c^{(k)}(a)}{k!}(w-a)^k$$

$$= c(a) + \sum_{i=1}^{k} \frac{c^{(i)}(a)}{i!}(w-a)^i$$

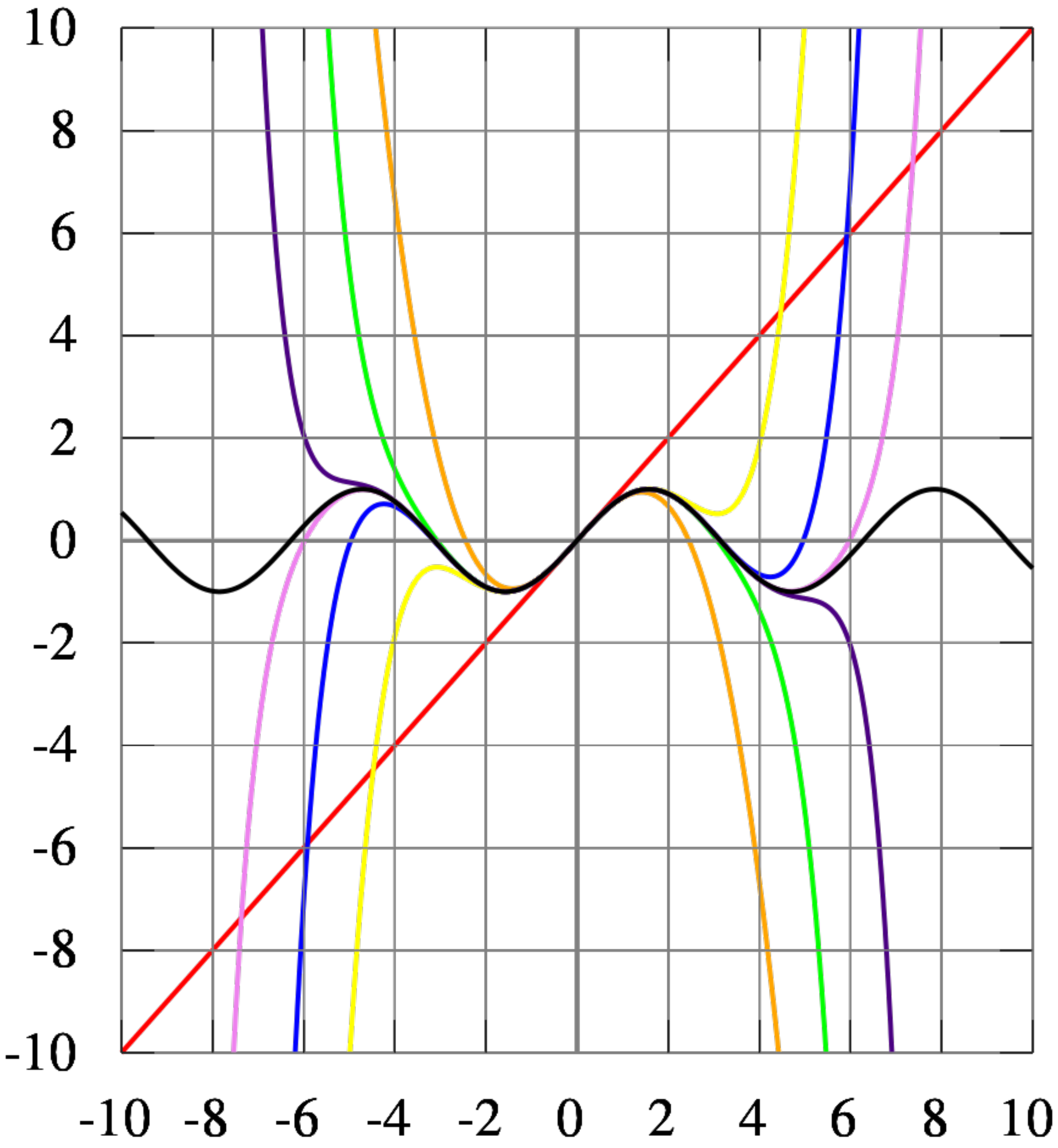
# Taylor Series Visualization



# Taylor Series Visualization (2)

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

Approximating sin function
at point x0.
What is x0?
How can you tell?



degree 1, 3, 5, 7, 9, 11 and 13.

# Taylor Series

**Definition:** A **Taylor series** is a way of approximating a function $c$ in a small neighbourhood around a point $a$:

$$c(w) \approx c(a) + c'(a)(w-a) + \frac{c''(a)}{2}(w-a)^2 + \cdots + \frac{c^{(k)}(a)}{k!}(w-a)^k$$

$$= c(a) + \sum_{i=1}^{k} \frac{c^{(i)}(a)}{i!}(w-a)^i$$

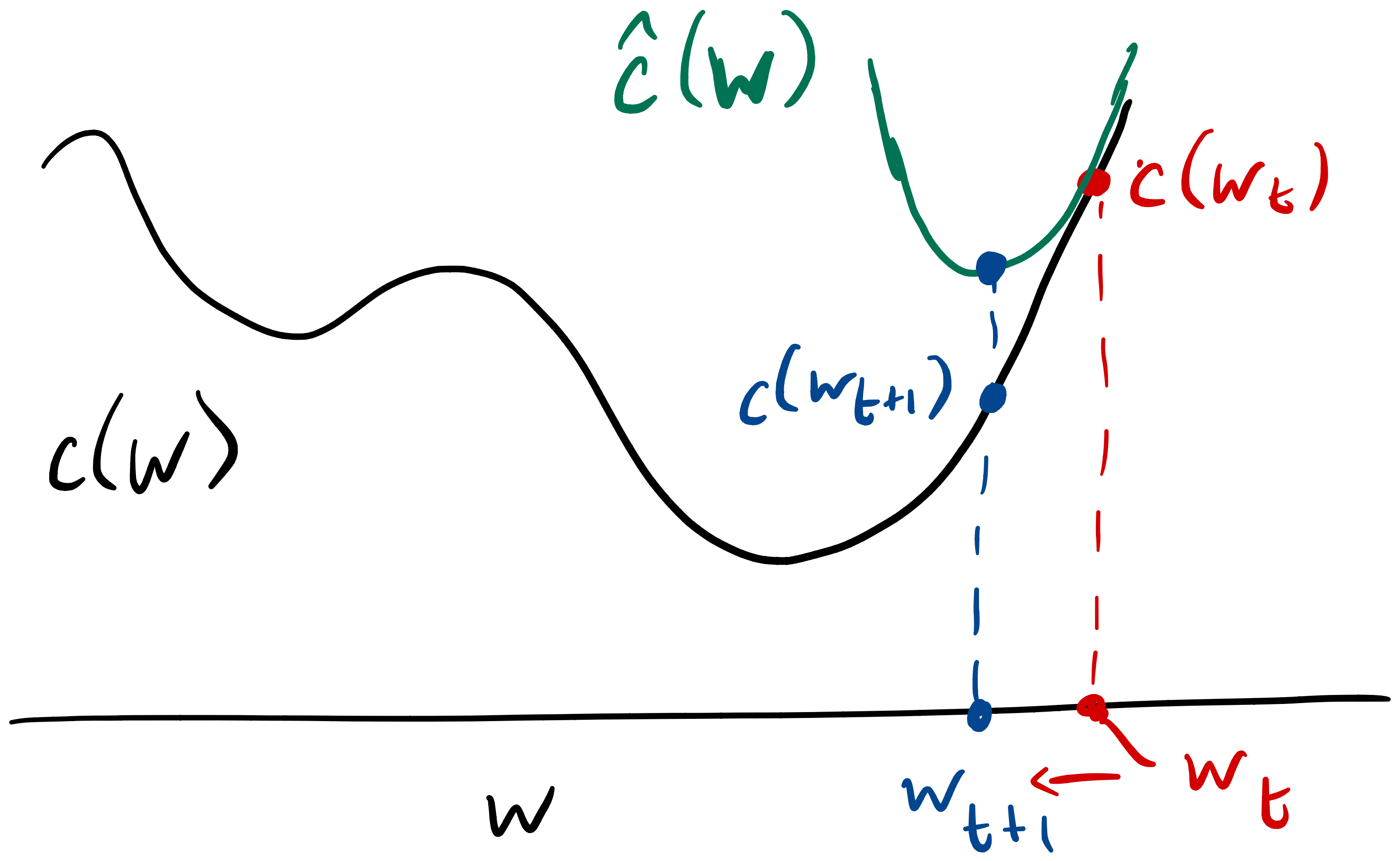
- *Intuition:* Following tangent line of the function approximates how it changes
  - i.e., following a function with the same first derivative
  - Following a function with the same first and second derivatives is a better approximation; with the same first, second, third derivatives is even better; etc.

# Second-Order Gradient Descent (Newton-Raphson Method)

1. Approximate the target function with a **second-order Taylor series** around the current guess $w_t$: $$\hat{c}(w) = c(w_t) + c'(w_t)(w - w_t) + \frac{c''(w_t)}{2}(w - w_t)^2$$

2. Find the stationary point of the approximation

$$\boxed{w_{t+1} \leftarrow w_t - \frac{c'(w_t)}{c''(w_t)}}$$



$w_{t+1}$ minimum of $\hat{c}$

Notice $c(w_{t+1}) < c(w_t)$

# Second-Order Gradient Descent (Newton-Raphson Method)

1. Approximate the target function with a **second-order Taylor series** around the current guess $w_t$:

$$\hat{c}(w) = c(w_t) + c'(w_t)(w - w_t) + \frac{c''(w_t)}{2}(w - w_t)^2$$

2. Find the stationary point of the approximation

$$\boxed{w_{t+1} \leftarrow w_t - \frac{c'(w_t)}{c''(w_t)}}$$

3. If the stationary point of the approximation is a (good enough) stationary point of the objective, then stop. Else, goto 1.

$$0 = \frac{d}{dw}\left[c(a) + c'(a)(w - a) + \frac{c''(a)}{2}(w - a)^2\right]$$

$$= c'(a) + 2\frac{c''(a)}{2}w - 2\frac{c''(a)}{2}a$$

$$= c'(a) + c''(a)(w - a)$$

$$\iff \quad -c'(a) = c''(a)(w - a)$$

$$\iff \quad (w - a) = -\frac{c'(a)}{c''(a)}$$

$$\iff \quad w = a - \frac{c'(a)}{c''(a)}$$

# (First-Order) Gradient Descent

- We can run Newton-Raphson whenever we have access to both the first and second derivatives of the target function
- Often we want to only use the **first derivative** (**why?**)
- **First-order gradient descent:** Replace the **second derivative** with a constant $\frac{1}{\eta}$ (the **step size**) in the approximation:

$$\hat{c}(w) = c(w_t) + c'(w_t)(w - w_t) + \frac{c''(w_t)}{2}(w - w_t)^2$$

$$\hat{c}(w) = c(w_t) + c'(w_t)(w - w_t) + \frac{1}{2\eta}(w - w_t)^2$$

- By exactly the same derivation as before:

$$\boxed{w_{t+1} \leftarrow w_t - \eta c'(w_t)}$$

# Partial Derivatives

- **So far:** Optimizing univariate function $c : \mathbb{R} \to \mathbb{R}$
- **But actually:** Optimizing multivariate function $c : \mathbb{R}^d \to \mathbb{R}$
  - $d$ is typically **HUGE** ($d \gg 10{,}000$ is not uncommon)
- First derivative of a multivariate function is a vector of partial derivatives

**Definition:**

The **partial derivative** $\frac{\partial f}{\partial x_i}(x_1, \ldots, x_d)$

of a function $f(x_1, \ldots, x_d)$ at $x_1, \ldots, x_d$ with respect to $x_i$ **is** $g'(x_i)$, where

$$g(y) = f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_d)$$

# Gradients

The multivariate analog to a **first derivative** is called a **gradient**.

**Definition:**

The **gradient** $\nabla f(\mathbf{x})$ of a function $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{x} \in \mathbb{R}^d$ is a vector of all the partial derivatives of $f$ at $\mathbf{x}$:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial_{x_1}}(\mathbf{x}) \\ \frac{\partial f}{\partial_{x_2}}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial_{x_d}}(\mathbf{x}) \end{bmatrix}$$

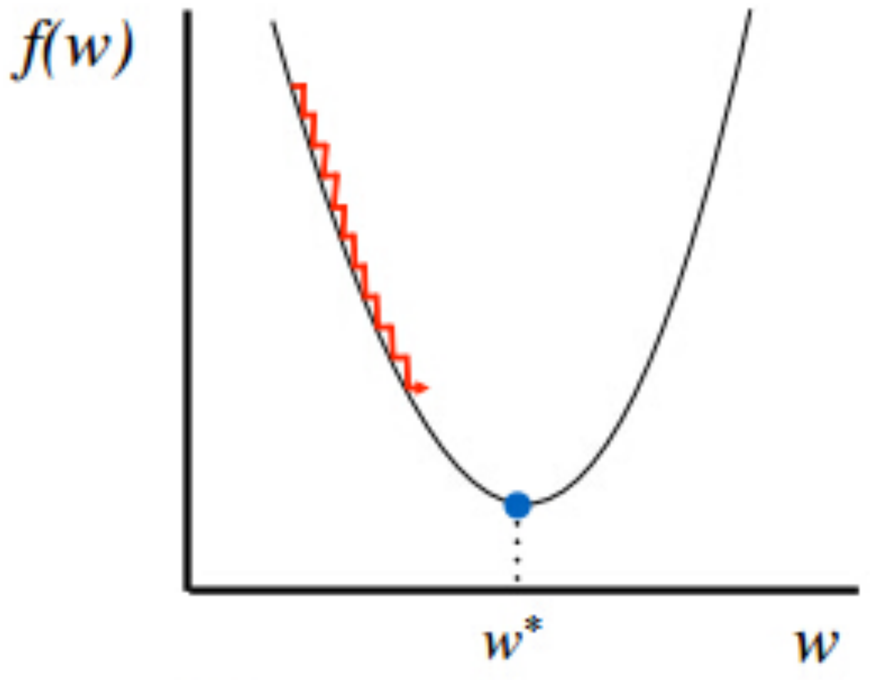
# Multivariate Gradient Descent

First-order gradient descent for multivariate functions $c : \mathbb{R} \to \mathbb{R}$ is just:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla c(\mathbf{w}_t)$$

- Notice the $t$ subscript on $\eta$
- We can choose a **different** $\eta_t$ for each iteration
  - Indeed, for univariate functions, Newton-Raphson can be understood as first-order gradient descent that chooses a step size of $\eta_t = \dfrac{1}{c''(w_t)}$ at each iteration.
- Choosing a good step size is crucial to efficiently using first-order gradient descent

# Adaptive Step Sizes



(a) Step-size too small

- If the step size is **too small**, gradient descent will "work", but take forever
- **Too big**, and we can overshoot the optimum
- Ideally, we would choose $\eta_t = \arg\min_{\eta \in \mathbb{R}^+} c\left(\mathbf{w}_t - \eta \nabla c(\mathbf{w}_t)\right)$
  - But that's another optimization!
- There are some heuristics that we can use to **adaptively** guess good values for $\eta_t$

# Line Search

A simple heuristic: **line search**

1. Try some largest-reasonable step size $\eta_t^{(0)} = \eta_{\text{max}}$
2. Is $c\left(w_t - \eta_t^{(s)} \nabla c(w_t)\right) < c(w_t)$?
   If yes, $w_{t+1} \leftarrow w_t - \eta_t^{(s)} \nabla c(w_t)$
3. Otherwise, try $\eta_t^{(s+1)} = \tau \eta_t^{(s)}$
   (for $\tau < 1$) and goto 2

**Intuition:**

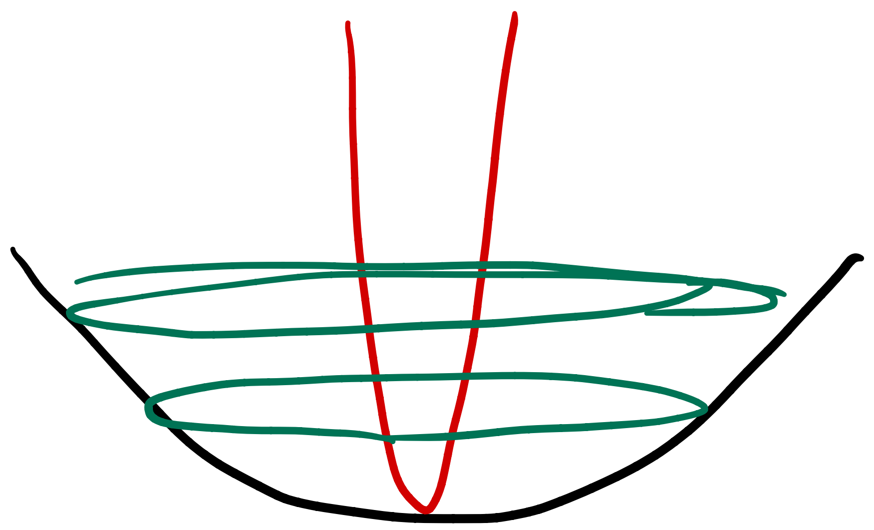
- Big step sizes are better so long as they don't overshoot
- Try a big step size! If it *increases* the objective, we must have overshot, so try a smaller one.
- Keep trying smaller ones until you *decrease* the objective; then start iteration $t+1$ from $\eta_{\text{max}}$ again.

- Typically $\tau \in [0.5, 0.9]$

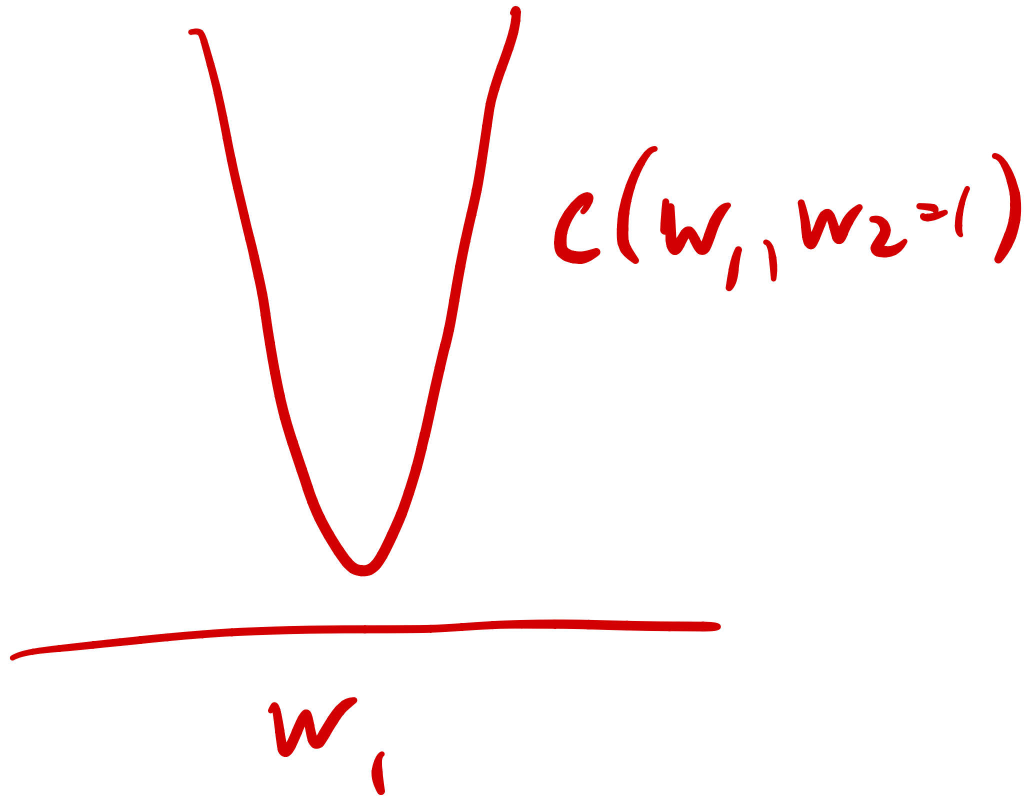
# Do we have to use a scalar stepsize?

- Or can we use a different stepsize per dimension? And why would we?

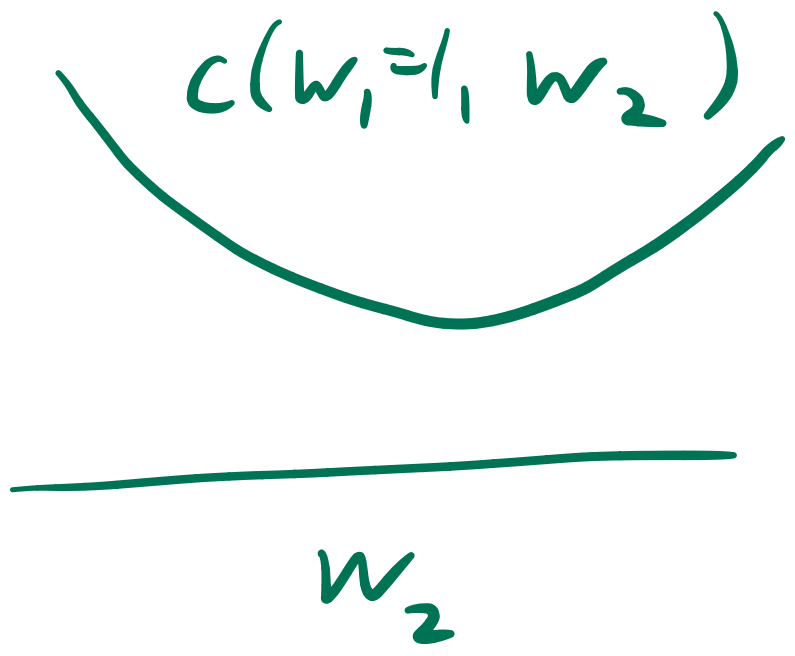
Weights $w_1$ and $w_2$



For fixed $w_2 = 1$



For fixed $w_1$



$\alpha_1$ should be small

$\alpha_2$ should be big

# Optimization Properties

1. **Maximizing** $c(w)$ is the same as minimizing $-c(w)$:

$$\arg\max_{w} c(w) = \arg\min_{w} - c(w)$$

2. **Equivalence under constant shifts:** Adding, subtracting, or multiplying by a positive constant does not change the minimizer of a function:

$$\arg\min_{w} c(w) = \arg\min_{w} c(w) + k = \arg\min_{w} c(w) - k = \arg\min_{w} kc(w) \quad \forall k \in \mathbb{R}^{+}$$

3. **Convex functions** have a global minimum at every stationary point

$$c \text{ is convex} \iff c(t\mathbf{w}_1 + (1-t)\mathbf{w}_2) \leq tc(\mathbf{w}_1) + (1-t)c(\mathbf{w}_2)$$

# Summary

- We often want to find the argument $w^*$ that **minimizes** an **objective function** $c$:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} c(\mathbf{w})$$

- Every interior minimum is a **stationary point**, so check the stationary points
- Stationary points usually identified **numerically**
  - Typically, by **gradient descent**
- Choosing the **step size** is important for efficiency and correctness
  - Common approach: Adaptive step size
  - E.g., by **line search**

# Exercise: Making your own optimization algorithm

- Imagine I told you that you need to find

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^d} c(\mathbf{w})$$

- Pretend you have never heard of gradient descent. What algorithm might you design to find this?

- Now what if I told you that $w \in \mathscr{W} = \{1,2,3,\ldots,1000\}$. Now how would you solve

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathscr{W}} c(\mathbf{w})$$