Optimization

CMPUT 267: Basics of Machine Learning

Textbook §4.1-4.4

Comments

- Assignment 1 due this week
- For next Reading Exercises, we will release some Practice Questions
- Two advertisements for student clubs:
  - Undergraduate AI Society is hosting a Computer Hex Tournament: https://hex-tournament.devpost.com/
  - Students for Machine Learning in Business (interdisciplinary, including CS, Engineering, Business, etc) are looking for a Co-Director of Data Science

Clarification for Q5 in Assignment 1

- There is a natural tendency to understand things intuitively. Sometimes mathematical formulas are not intuitive. Intuitive is not necessarily better, applying formulas/algorithms is also important
  - As time passes, having intuition about math actually gets easier. Sometimes in the beginning it is simply better to try executing
  - If you cannot intuit something, then that is ok!
- Simplified Q5b and gave more details about Chebyshev's
- Let's go over what it means to take the expectation of a sample average

Sample Average

- Imagine we flip a fair coin 3 times and take $\bar{X} = \frac{1}{3} \sum_{i=1} X_i$
- 8 possible datasets, $\mathscr{D}_1 = \{0,0,0\}$, $\mathscr{D}_2 = \{0,0,1\}$, $\mathscr{D}_3 = \{0,1,0\}$, $\mathscr{D}_4 = \{0,1,1\}$, $\mathscr{D}_5 = \{1,0,0\}$, $\mathscr{D}_6 = \{1,0,1\}$, $\mathscr{D}_7 = \{1,1,0\}$, $\mathscr{D}_8 = \{1,1,1\}$
- $\mathbb{E}[\bar{X}] = \sum_{j=1}^{8} p(\mathscr{D}_j)\text{average}(\mathscr{D}_j)$ where $\bar{x} = \text{average}(\mathscr{D}_j)$ is a possible outcome

Sample Average for continuous RVs

- Imagine we sample from a Gaussian 3 times and take $\bar{X} = \frac{1}{3}\sum_{i=1} X_i$
- Uncountably many possible datasets, $\mathscr{D}_1 = \{1.3, 0.11, 0.35674\}$, $\mathscr{D}_2 = \{-0.33, 0, 9.45\}$, …
- $\mathbb{E}[\bar{X}] = \int p(\mathscr{D})\text{average}(\mathscr{D})d\mathscr{D} = \int p(x_1, x_2, x_3)\text{average}(x_1, x_2, x_3)dx_1 dx_2 x_3$

Relevance to your assignment

- In the real world, you see precisely one of these dataset and one $\bar{x}$
- We reason about other datasets you could see, because we want to know: for equally likely datasets, do they have a similar $\bar{x}$? If yes, then you all agree and are probably all correct. If no, then who is right? You in a parallel universe or you in this universe? You can’t know. So we give an interval around your $\bar{x}$ to say: at least I am confident its somewhere in here
- In your assignment, you get to have multiple universes! We have synthetic data, so we simulate what it would be like to have multiple estimates of the sample average. (You do not ever do this in practice)

Question 5 on the assignment

- Changed to only ask you to give the confidence interval for the sample variance
- The sample variance is also an estimator, and we can reason about its bias and variance
- Example: $\bar{V} = \frac{1}{n}\sum_{i=1} X_i^2$ has $\mathbb{E}[\bar{V}] = \int p(\mathscr{D})\text{squared-average}(\mathscr{D})d\mathscr{D}$
- We can use our CI approaches for this estimator to ask: how much does it deviate from its true mean?

Back to Parameter Estimation

- In class, we started discussing that we will need to solve optimization problems so that we can find the parameters for our distributions/functions
- We won’t talk about those optimization problems just yet. Let’s first ask: in general, how do we solve optimization problems?

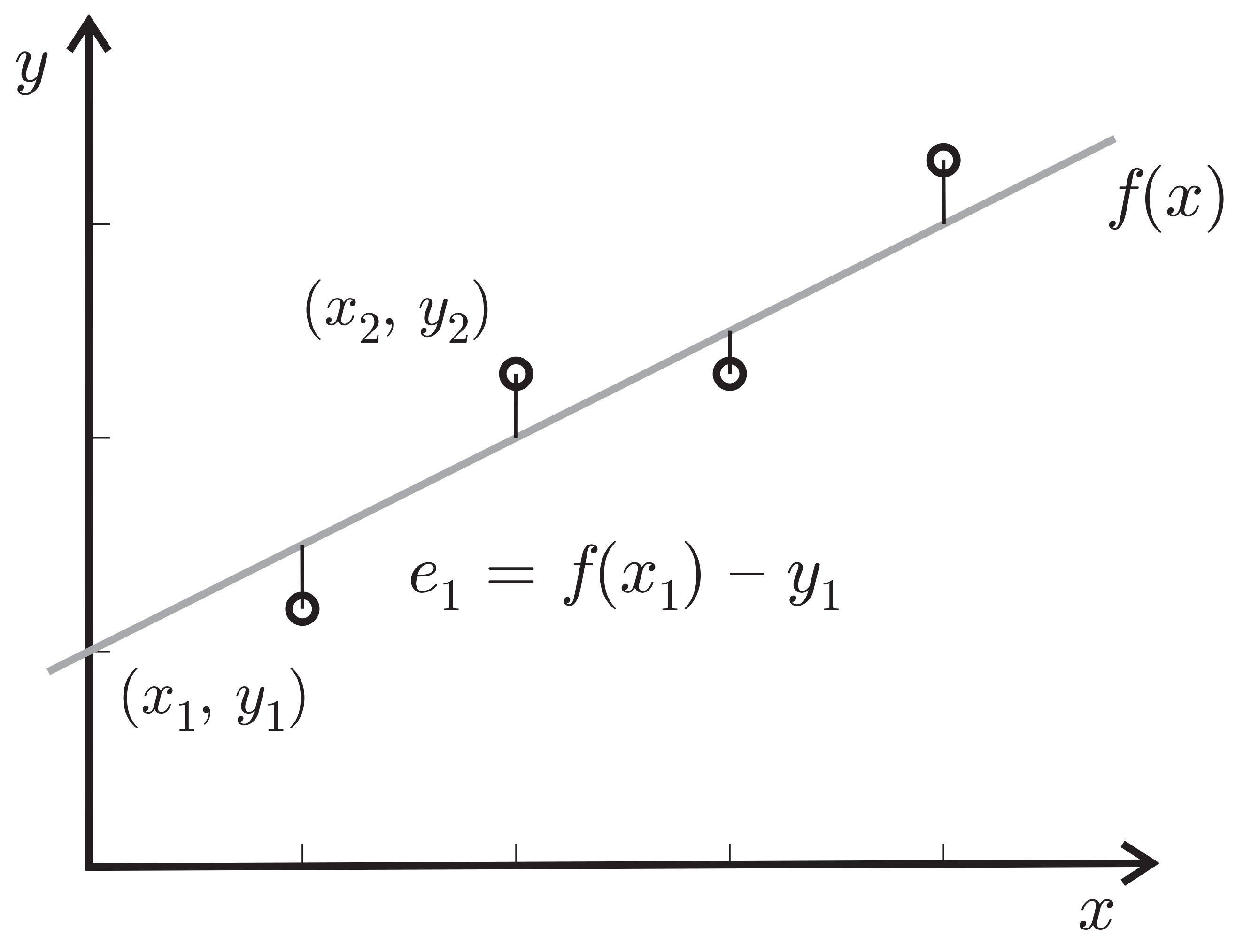
Optimization

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

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

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** $c(\mathbf{w}) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$



The set $\mathscr{W} = \mathbb{R}^2$. What if instead you wanted to find weights between [-10,10]?

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 $w \in \mathscr{W} = \{1,2,3,...,1000\}$. Now how would you solve

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

Optimization Properties

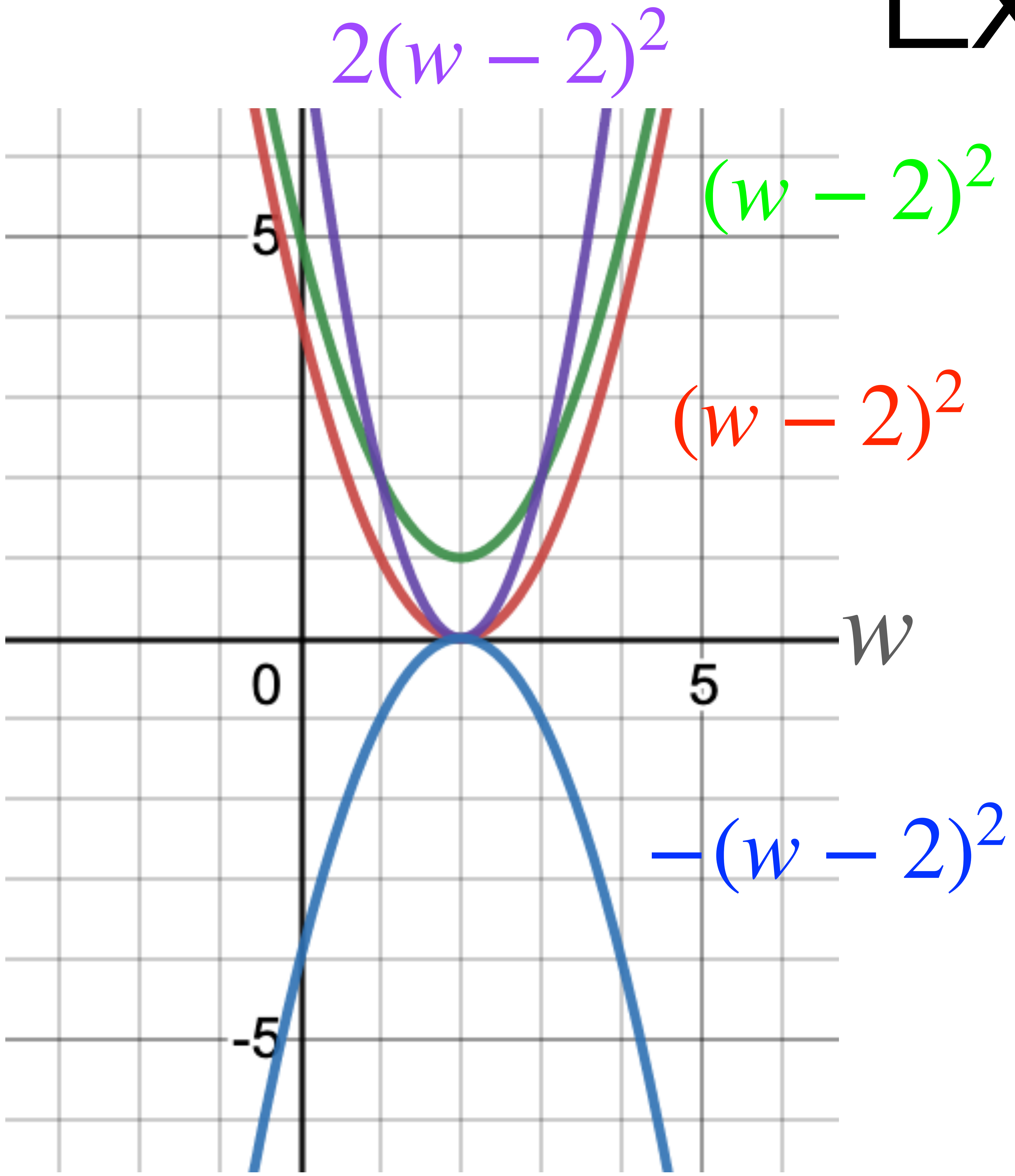
1. **Maximizing** $c(w)$ is the same as minimizing $-c(w)$:

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

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

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

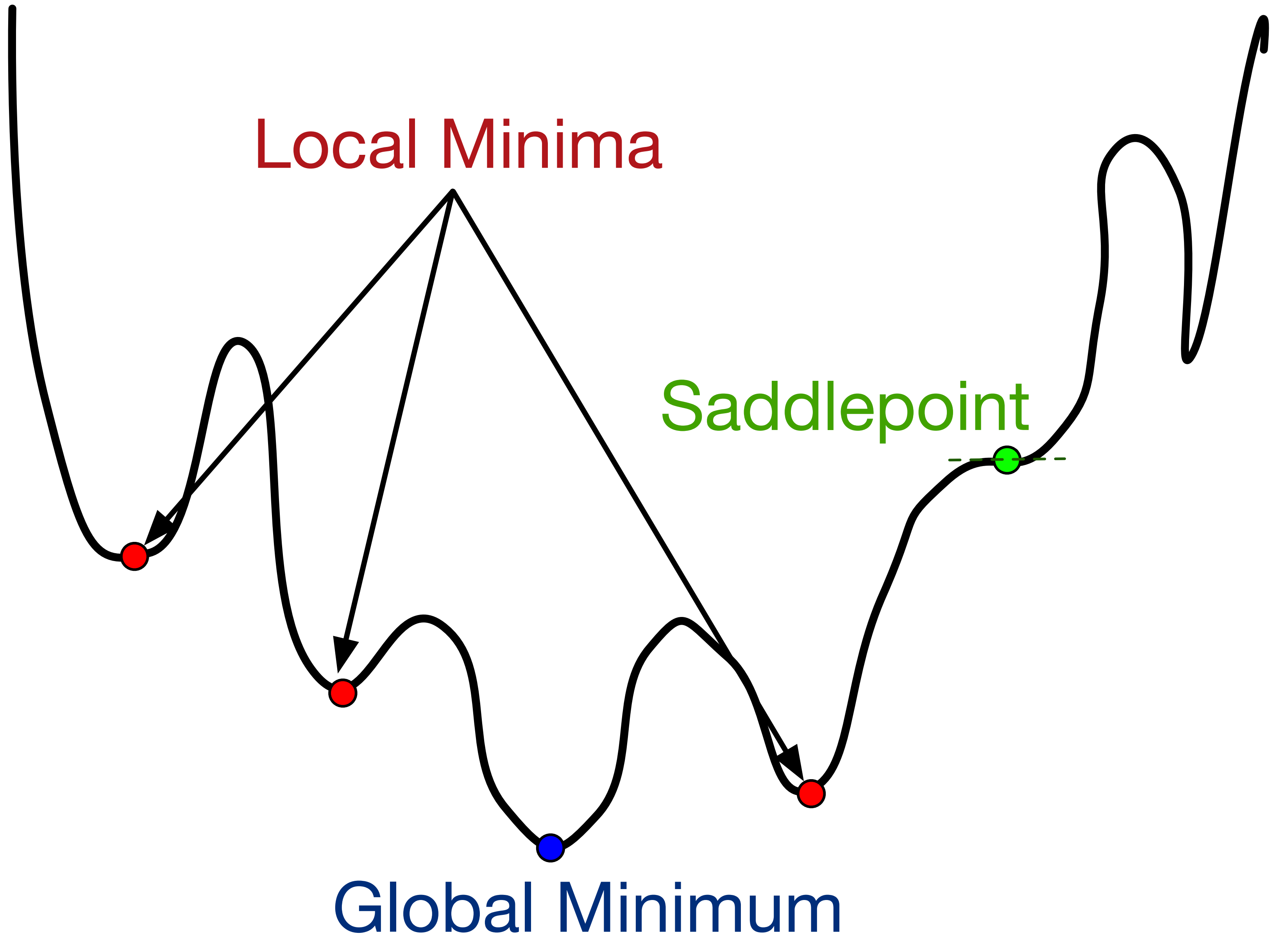
Example



$$
\begin{aligned}
&\arg\min_{w\in\mathbb{R}} \; (w-2)^2 \\
&= \arg\min_{w\in\mathbb{R}} \; 2(w-2)^2 \\
&= \arg\min_{w\in\mathbb{R}} \; (w-2)^2 + 1 \\
&= \arg\max_{w\in\mathbb{R}} \; -(w-2)^2 \\
&= 2
\end{aligned}
$$

Stationary Points

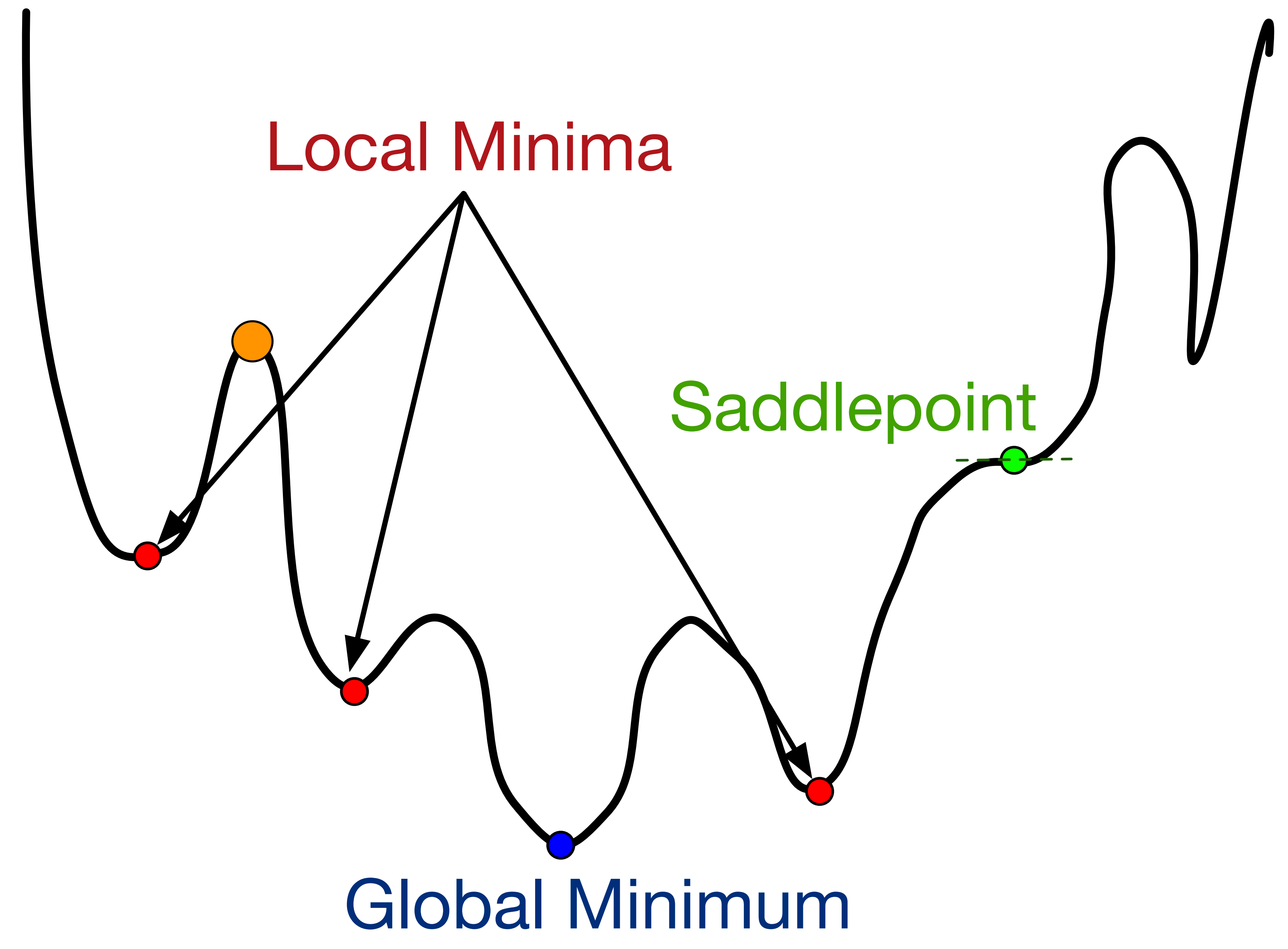
- Every minimum of an everywhere-differentiable function $c(w)$ **must** occur at a **stationary point**: A point at which $c'(w) = 0$
- However, not every stationary point is a minimum
- Every stationary point is either:
  - A **local minimum**
  - A **local maximum**
  - A **saddlepoint**
- The **global minimum** is either a local minimum (or a boundary point)



Let's assume for now that w is unconstrained (i.e, $w \in \mathbb{R}$ rather than $w \geq 0$ or $w \in [0,1]$)

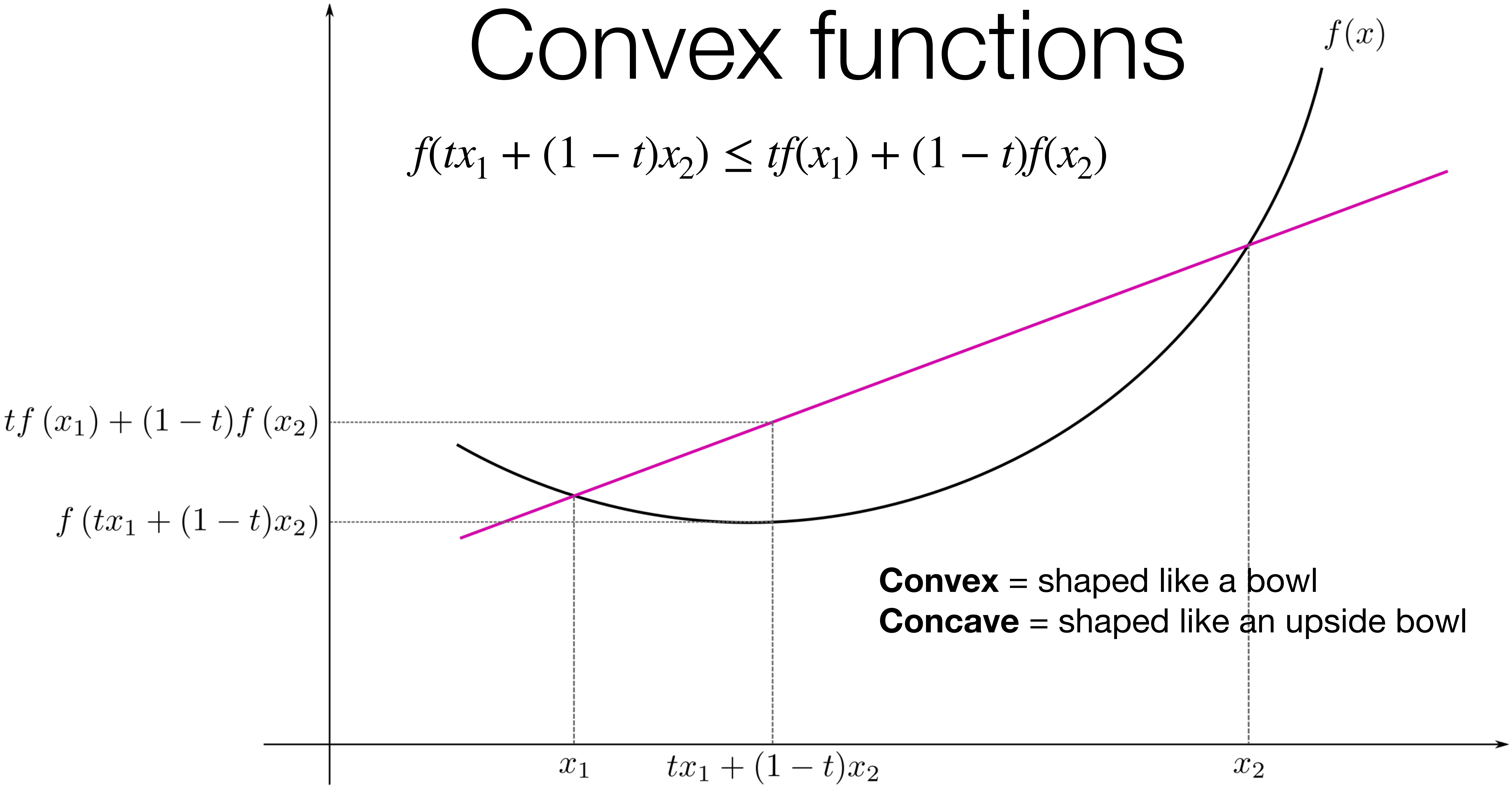
Identifying the type of the stationary point

- If function curved upwards (**convex**) locally, then **local minimum**



Convex functions

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

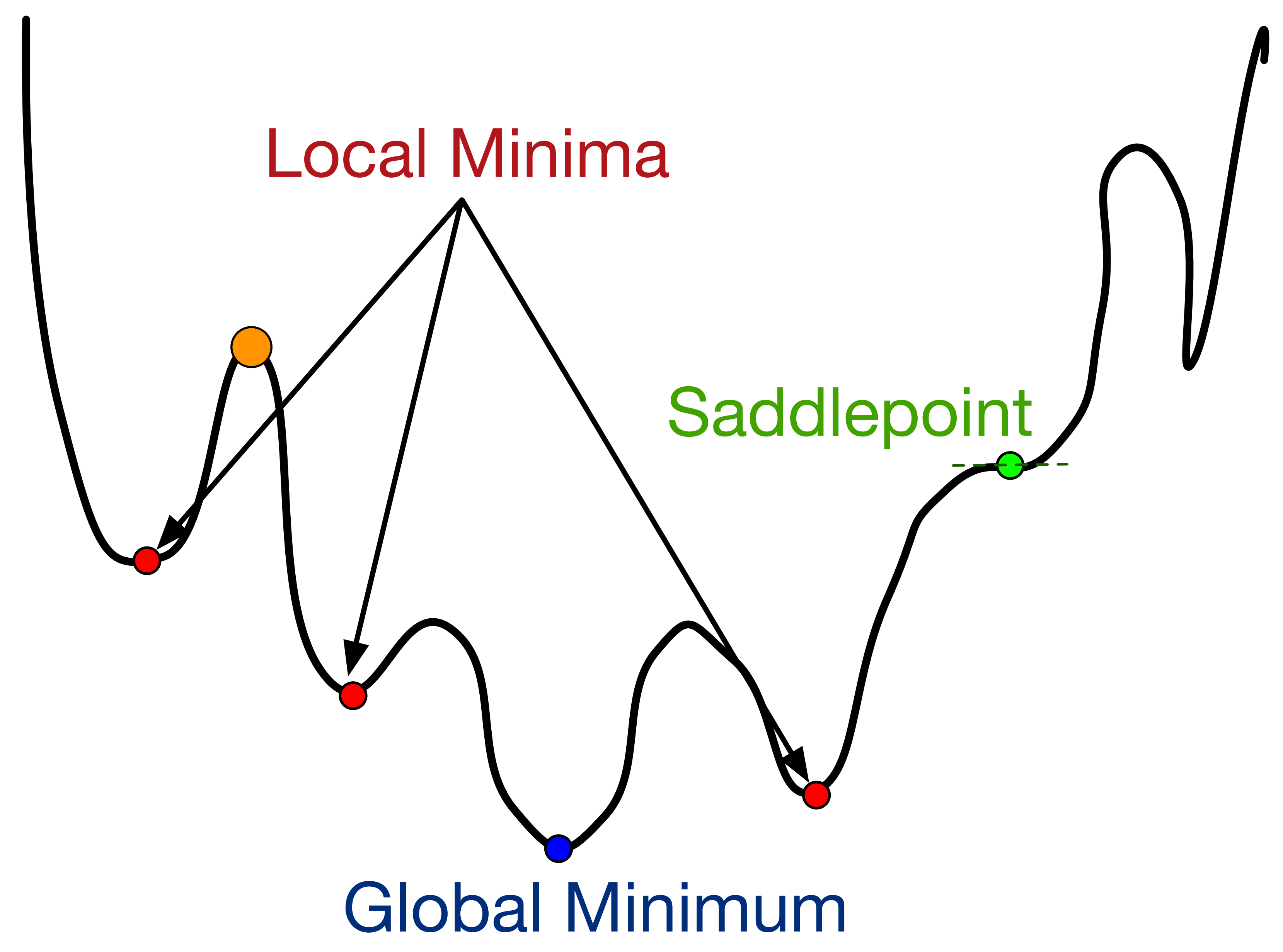


Convex = shaped like a bowl
Concave = shaped like an upside bowl

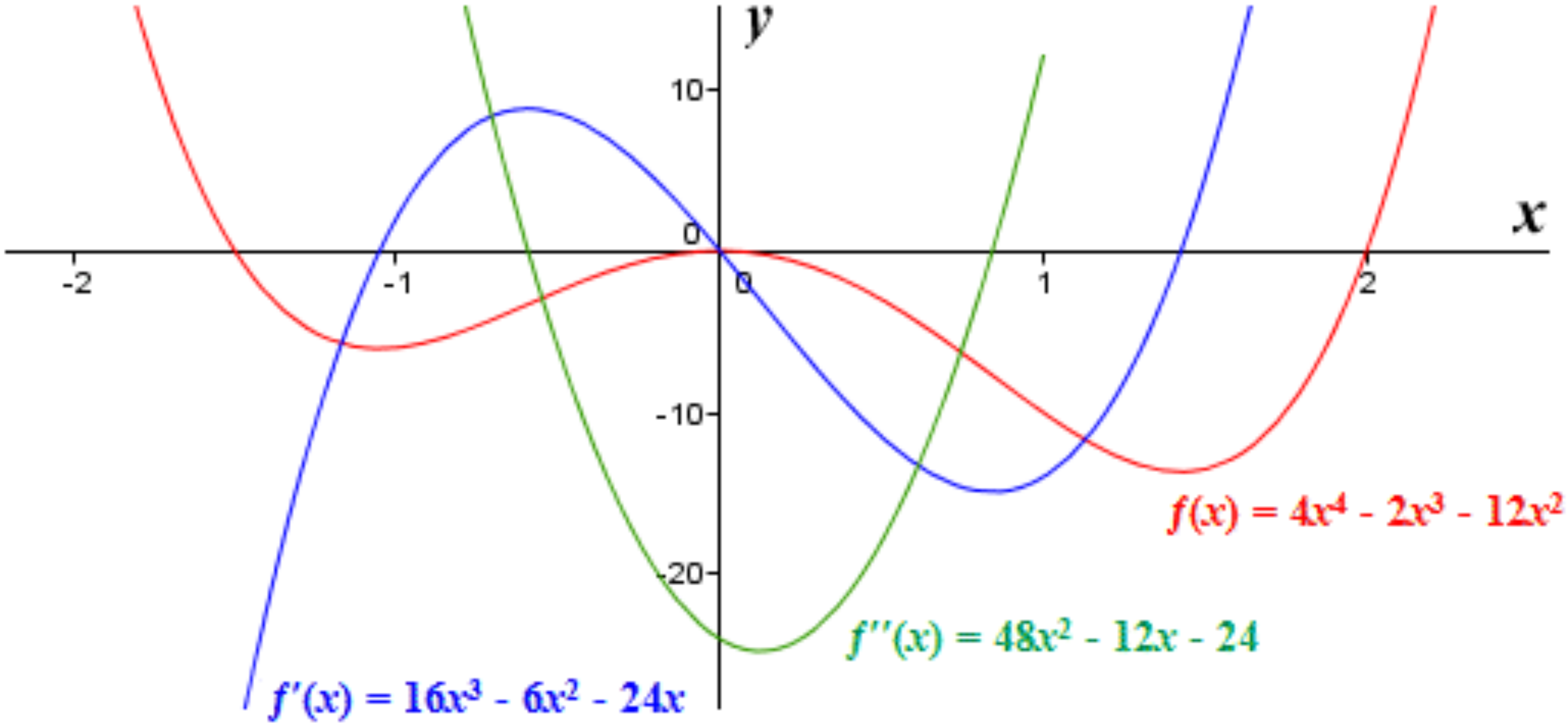
* from Wikipedia

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 might be a **saddlepoint** but could also be a local min or local max
- Locally, cannot distinguish between local min and global min (its a global property of the surface)

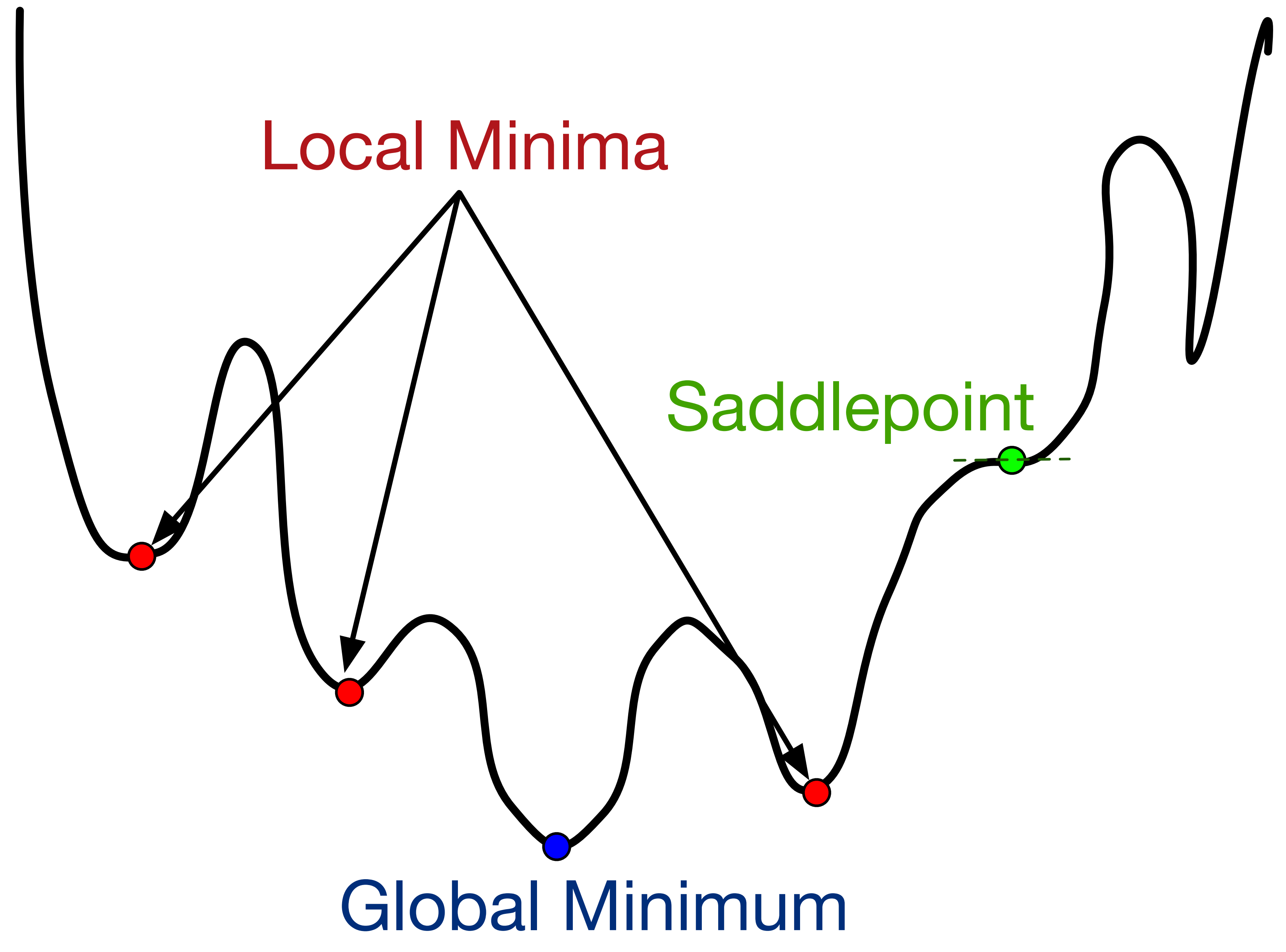


Second derivative reflects curvature



Second derivative test

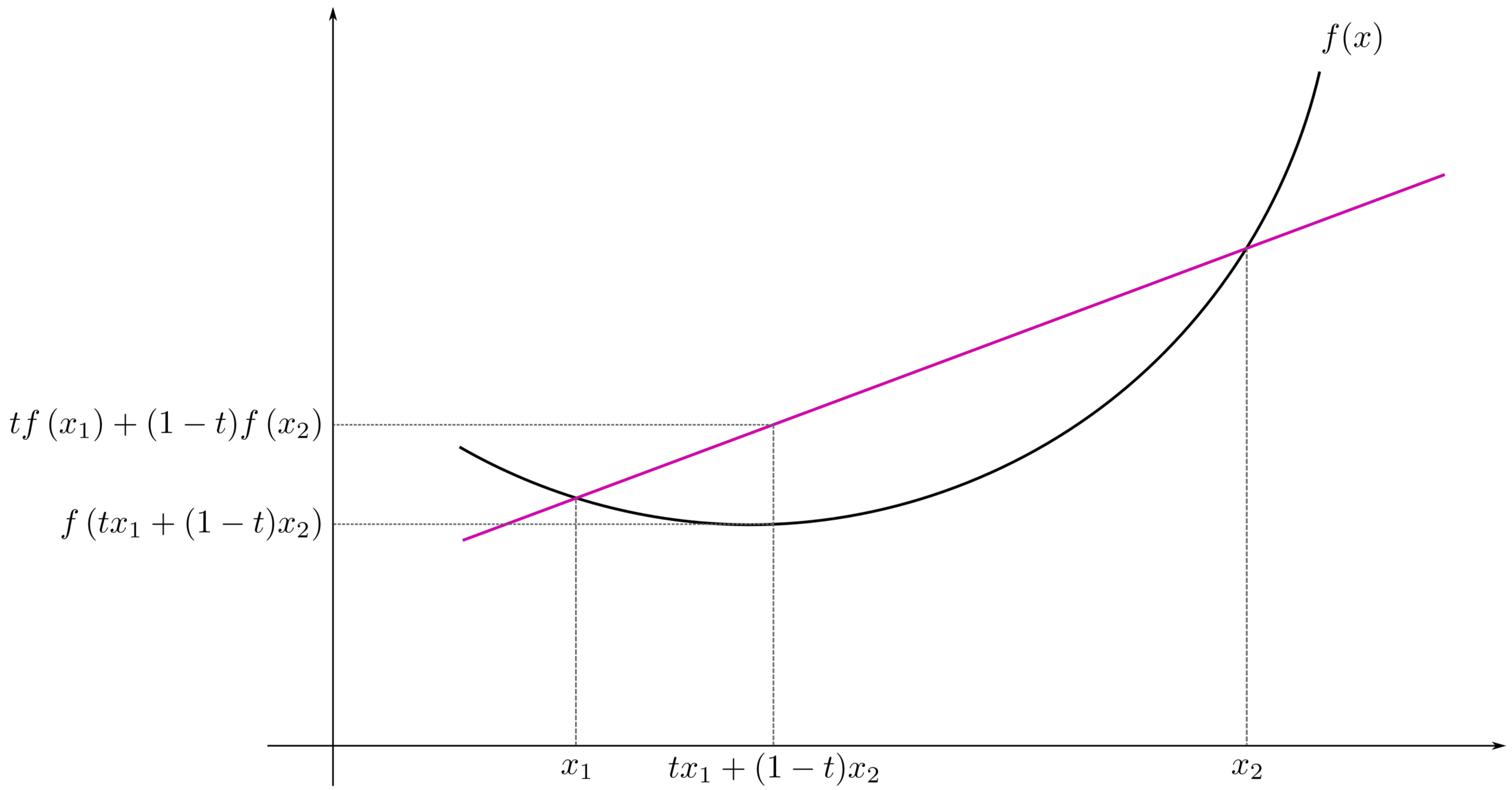
1. If $c''(w_0) > 0$ then w_0 is a local minimum.
2. If $c''(w_0) < 0$ then w_0 is a local maximum.
3. If $c''(w_0) = 0$ then the test is inconclusive: we cannot say which type of stationary point we have and it could be any of the three.



Testing optimality without the second derivative test

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

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



Procedure

- Find a stationary point, namely w_0 such that $c'(w_0) = 0$
  - Sometimes we can do this analytically (closed form solution, namely an explicit formula for w_0)
- Reason about if it is optimal
  - Check if your function is convex
  - If you have only one stationary point and it is a local minimum, then it is a global minimum
  - Otherwise, if second derivate test says its a local min, can only say that

Exercise

- Find the solution to the optimization problem $\min_{w \in \mathbb{R}} (w-2)^2 + (w-3)^2$
- Recall that the procedure is:
  - 1. Find a stationary point, namely w_0 such that $c'(w_0) = 0$
  - 2. Do the second derivative test (or reason about if this function is convex)

Solution

- $c(w) = (w-2)^2 + (w-3)^2$
- $c'(w) = 2(w-2) + 2(w-3) = 4w - 10$
- $c''(w) = 4$
- $c'(w_0) = 0 = 4w_0 - 10 \implies w_0 = 10/4 = 2.5$
- $c''(w_0) = 4 > 0$, so a local min. Only one stationary point, so its a global min.

Exercise: Prove equivalence under constant shifts

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

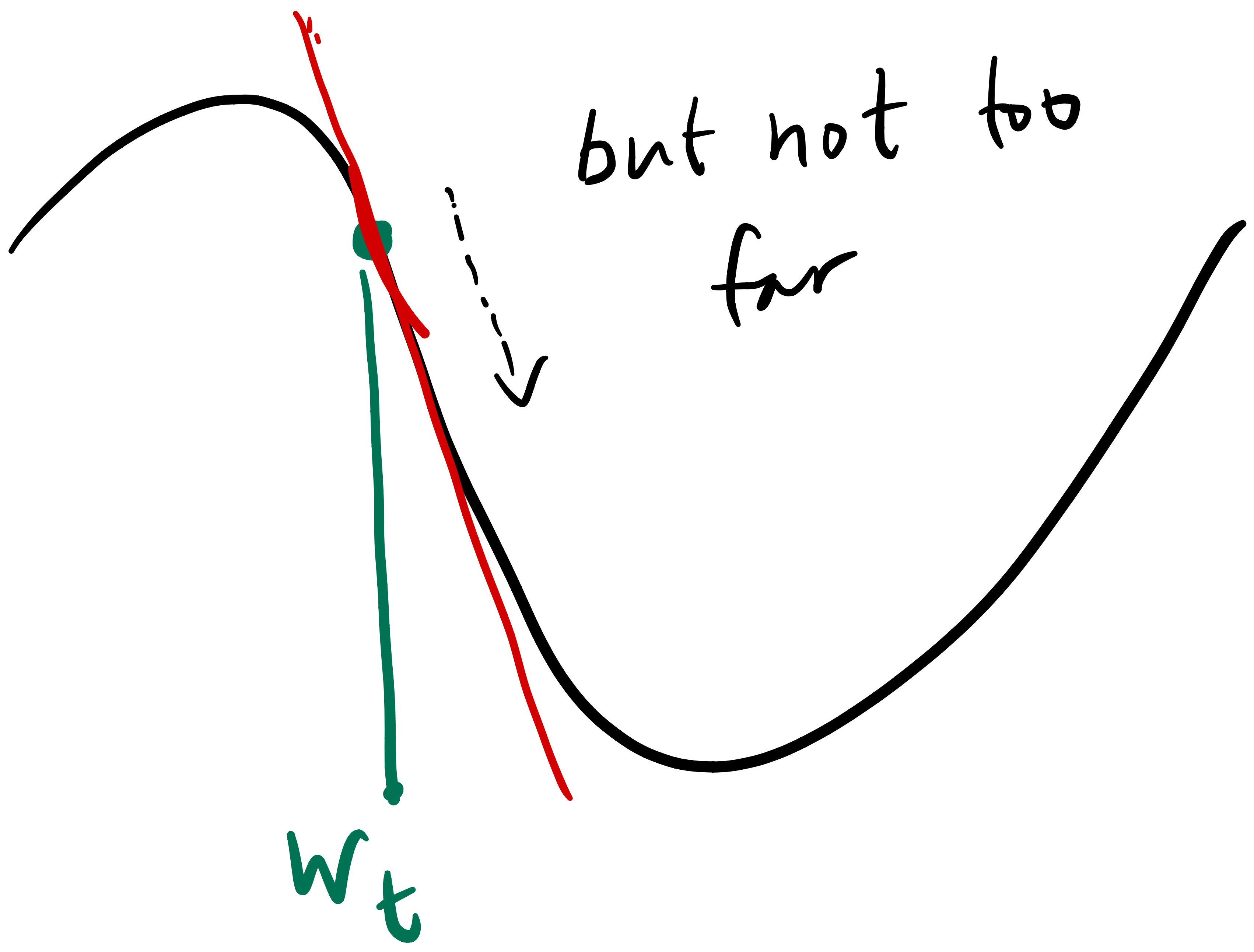
$$\arg\min_{w} c(w) = \arg\min_{w} c(w) + k = \arg\min_{w} c(w) - k = \arg\min_{w} k c(w) \quad \forall k \in \mathbb{R}^+$$

Show that all of these have the same set of stationary points, namely points w where c'(w) = 0

Numerical Optimization

- 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**

Intuitive explanation of gradient descent



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

$\eta > 0$ a small stepsize

A more careful explanation

- Why would it work to take small steps?
- What are we really doing?
- We are locally approximating the function, using a Taylor series
  - Note: I won't test you on the Taylor series derivation; the goal here is for you to understand where the algorithm comes from and help explain the differences between first and second-order gradient descent

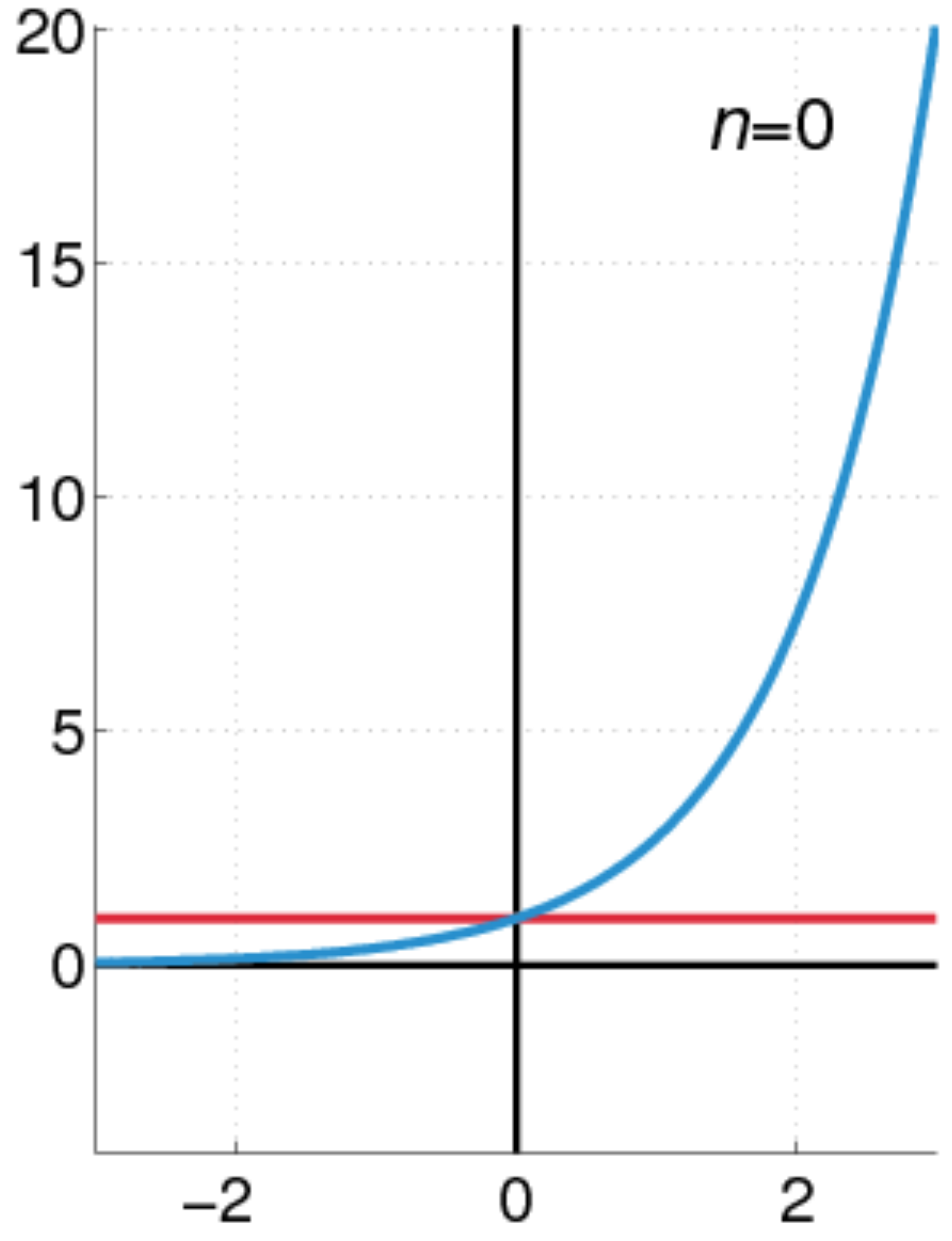
Taylor Series

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

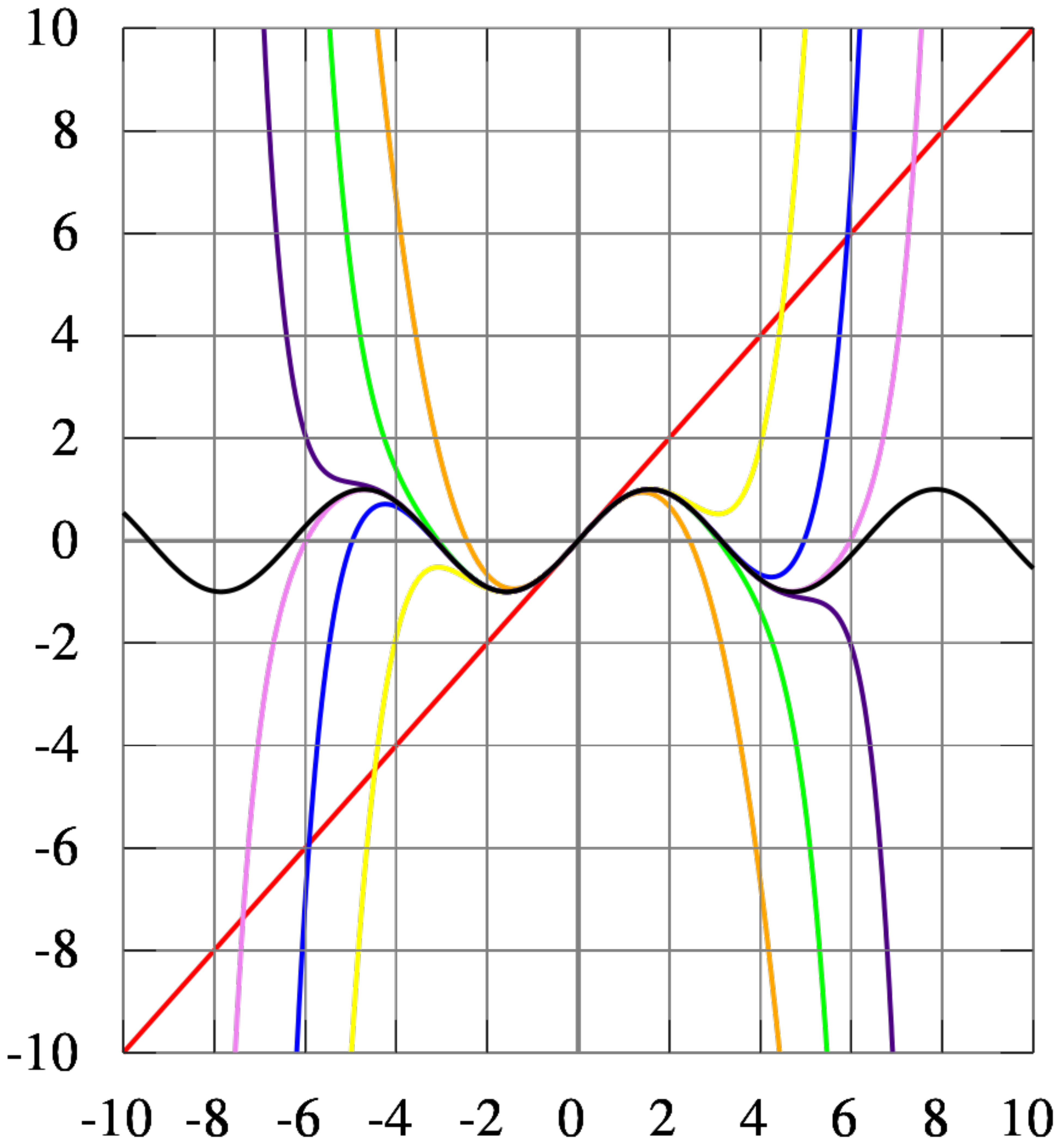


Taylor Series Visualization (2)

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

Approximating sin function
at point x0 = 0
(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:

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

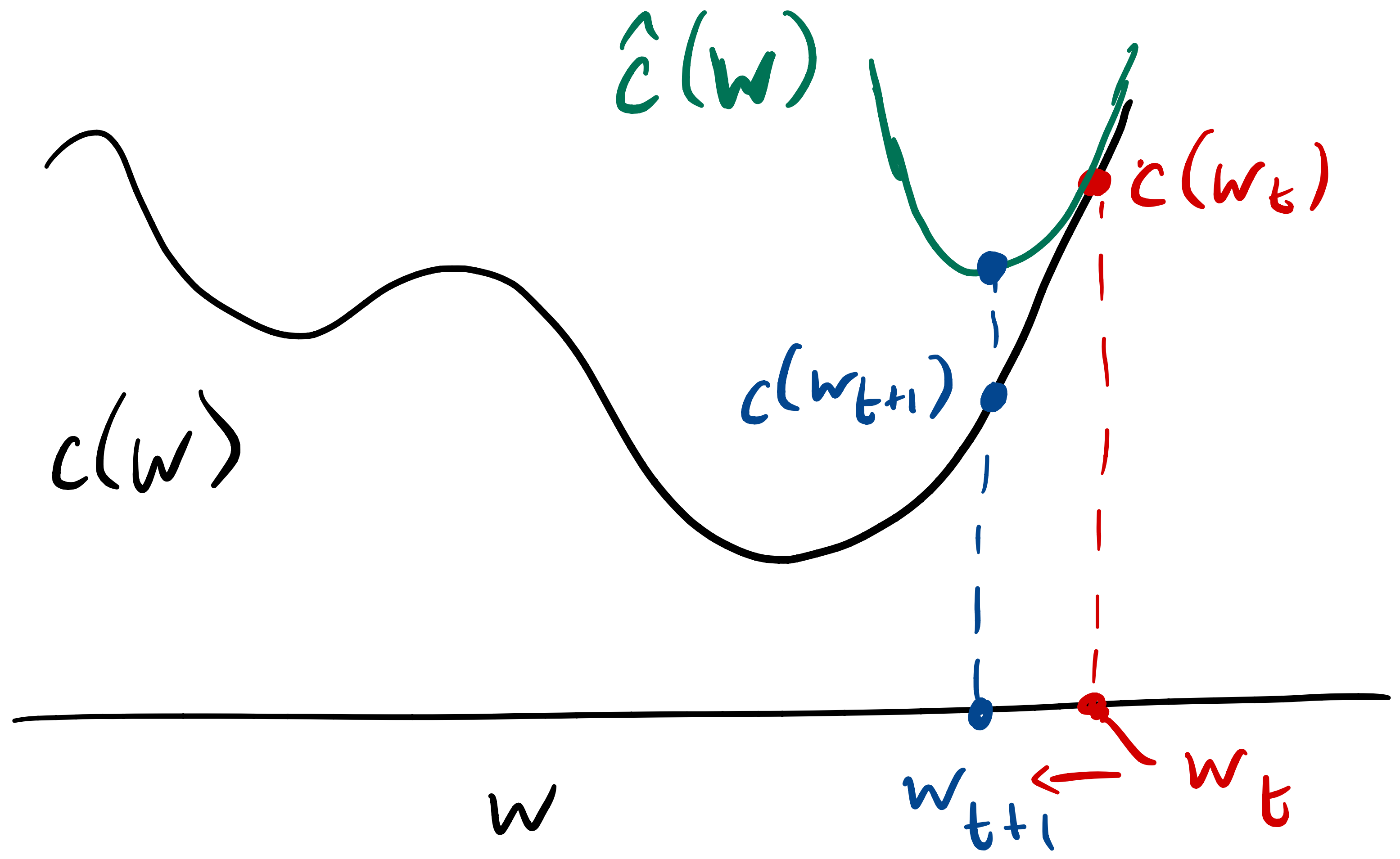
$$= c(a) + \sum_{i=1}^{k} \frac{c^{(i)}(a)}{i!}(w-a)^i$$

- *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

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.

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

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

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

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

(First-Order) Gradient Descent

- We can run Second-order GD whenever we have access to both the first and second derivatives of the target function
- Often we want to only use the **first derivative**
  - Not obvious yet why, but for the multivariate case second-order is computationally intensive
- **First-order gradient descent:** Replace the **second derivative** with a constant $\frac{1}{\eta}$ (the **step size**) in the approximation:

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

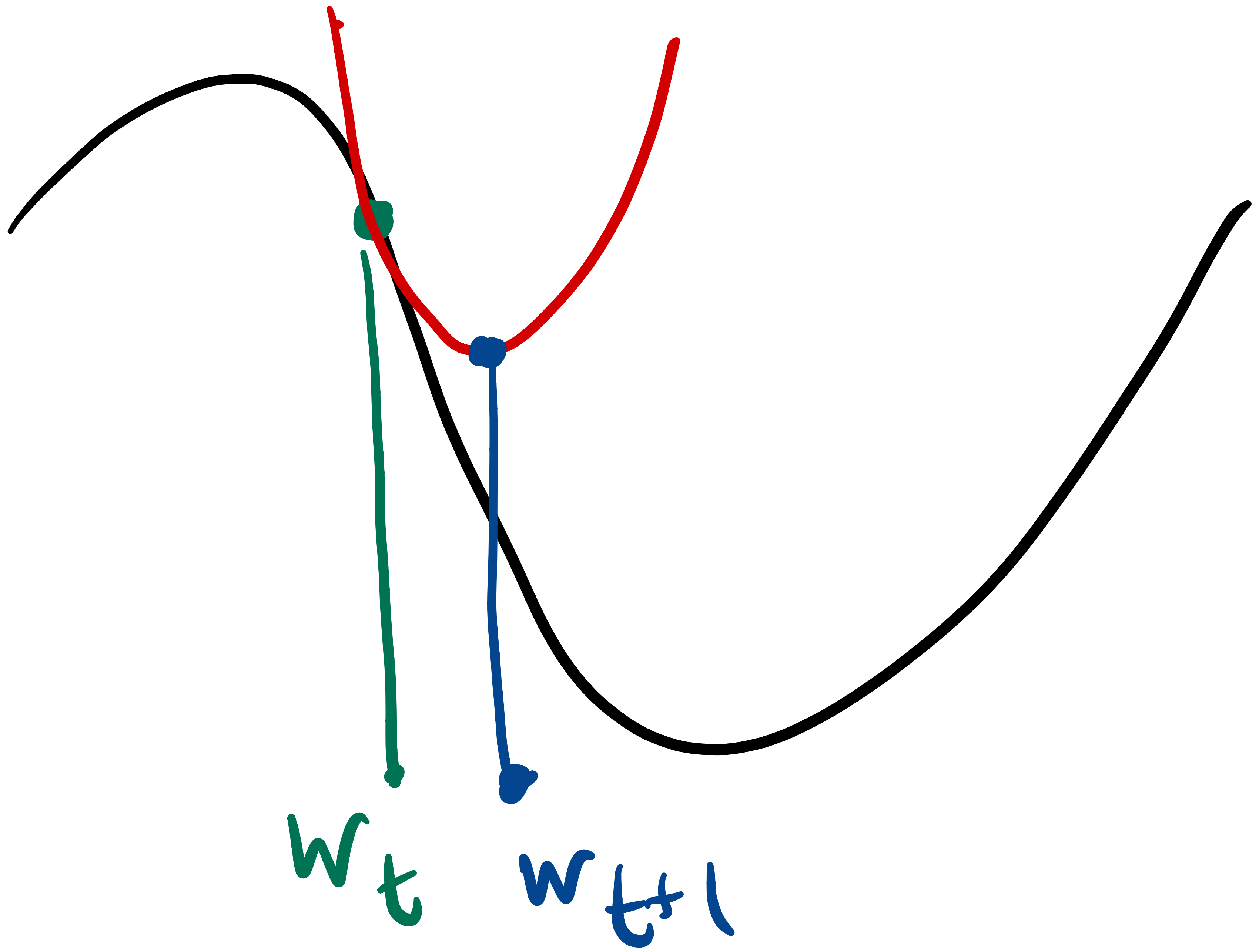
$$\hat{c}(w) = c(w_t) + c'(w_t)(w - w_t) + \frac{1}{2\eta}(w - w_t)^2$$

- By exactly the same derivation as before:

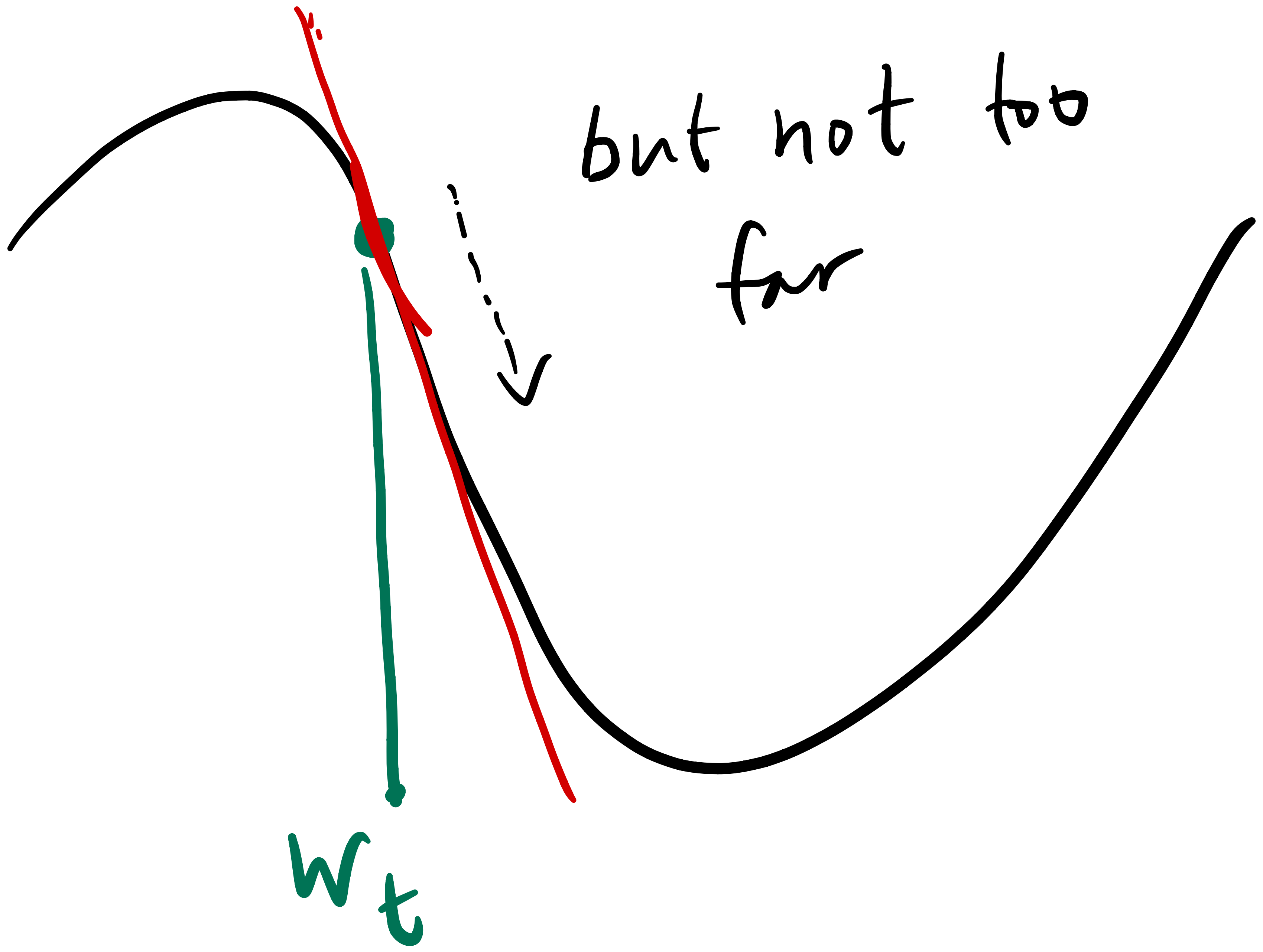
$$\boxed{w_{t+1} \leftarrow w_t - \eta c'(w_t)}$$

1st and 2nd order

OR



2nd order



1st order, distance controlled by stepsize

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 **H U G E** ($d \gg 10{,}000$ 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

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

Example

- $c(w_1, w_2) = (2w_1 + 4w_2 - 7)^2$
- $\dfrac{\partial c}{\partial w_1}(w_1, w_2) = 4(2w_1 + 4w_2 - 7)$
- Then we query at a particular point, e.g., $(w_1, w_2) = (1, -1)$, giving $\dfrac{\partial c}{\partial w_1}(1, -1) = 4(2 - 4 - 7) = -36$
- Equivalently, let $f(w_1) = c(w_1, -1)$ for this fixed w_2
- Then $f'(w_1) = \dfrac{\partial c}{\partial w_1}(w_1, -1)$, i.e., $f'(1) = \dfrac{\partial c}{\partial w_1}(1, -1) = -36$

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 $c : \mathbb{R}^d \to \mathbb{R}$ is just:

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

$$\begin{bmatrix} w_{t+1,1} \\ w_{t+1,2} \\ \vdots \\ w_{t+1,d} \end{bmatrix} = \begin{bmatrix} w_{t,1} \\ w_{t,2} \\ \vdots \\ w_{t,d} \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial c}{\partial_{w_1}}(\mathbf{w}_t) \\ \frac{\partial c}{\partial_{w_2}}(\mathbf{w}_t) \\ \vdots \\ \frac{\partial c}{\partial_{w_d}}(\mathbf{w}_t) \end{bmatrix}$$

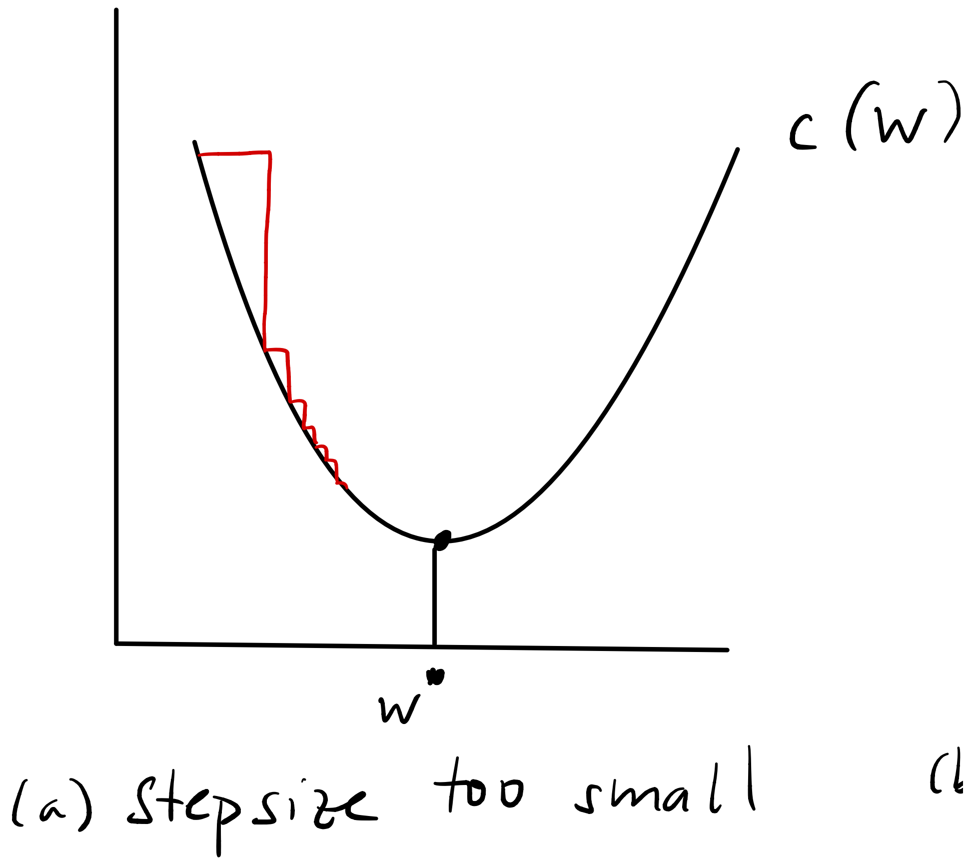
Extending to stepsize per timestep

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

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

- Notice the t subscript on η
- We can choose a **different** η_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



- If the step size is **too small**, gradient descent will "work", but take forever
- **Too big**, and we can overshoot the optimum
- There are some heuristics that we can use to **adaptively** guess good values for η_t
- 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!

Line Search

A simple heuristic: **line search**

1. Try some largest-reasonable step size $\eta_t^{(0)} = \eta_{\max}$
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 $\eta_{\max}$ again.

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

Adaptive stepsize algorithms

- Stepsize selection is very important, and so there is a vast array of algorithms for adaptive stepsizes
- Line search is a bit onerous to use, and not common with something called stochastic gradient descent (which is what we will use later)
- We will see smarter stepsize algorithms then, and in your assignment

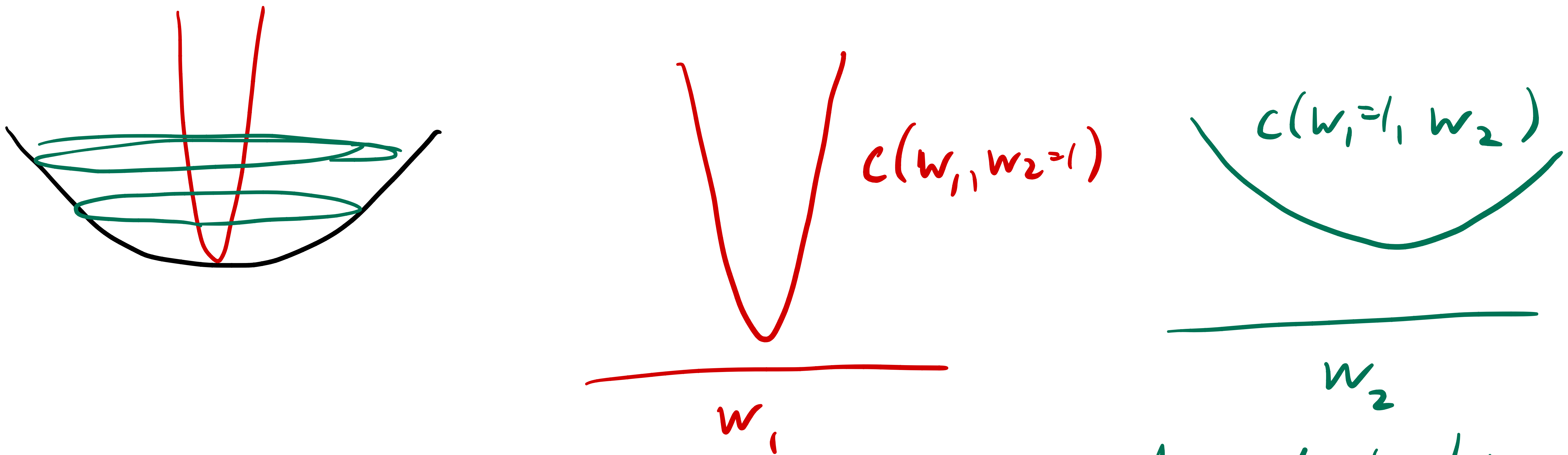
Do we have to use a scalar stepsize?

- 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



Stepsize 1 should be small

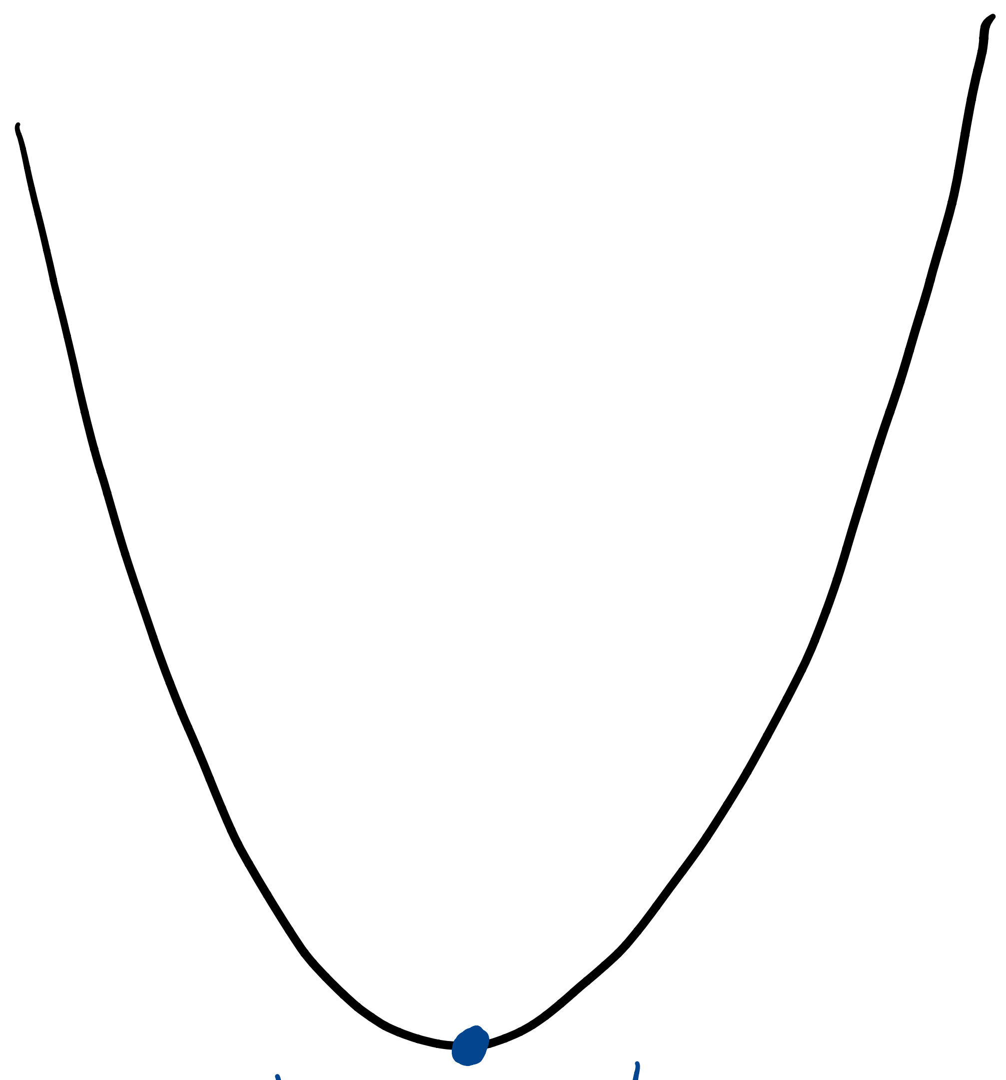
Stepsize 2 should be big

Now what if we have constraints?

- For this course, we almost always only deal with unconstrained problems
- We will only consider constraints like $w \geq 0$ or $w \in [a, b]$
- Then the procedure is:
  - 1. Find a stationary point
  - 2. Verify that it is the only stationary point, and a local min according to the second derivative test
  - 3. Additionally check if the boundary points have a smaller value

Visualizing the effect of constraints

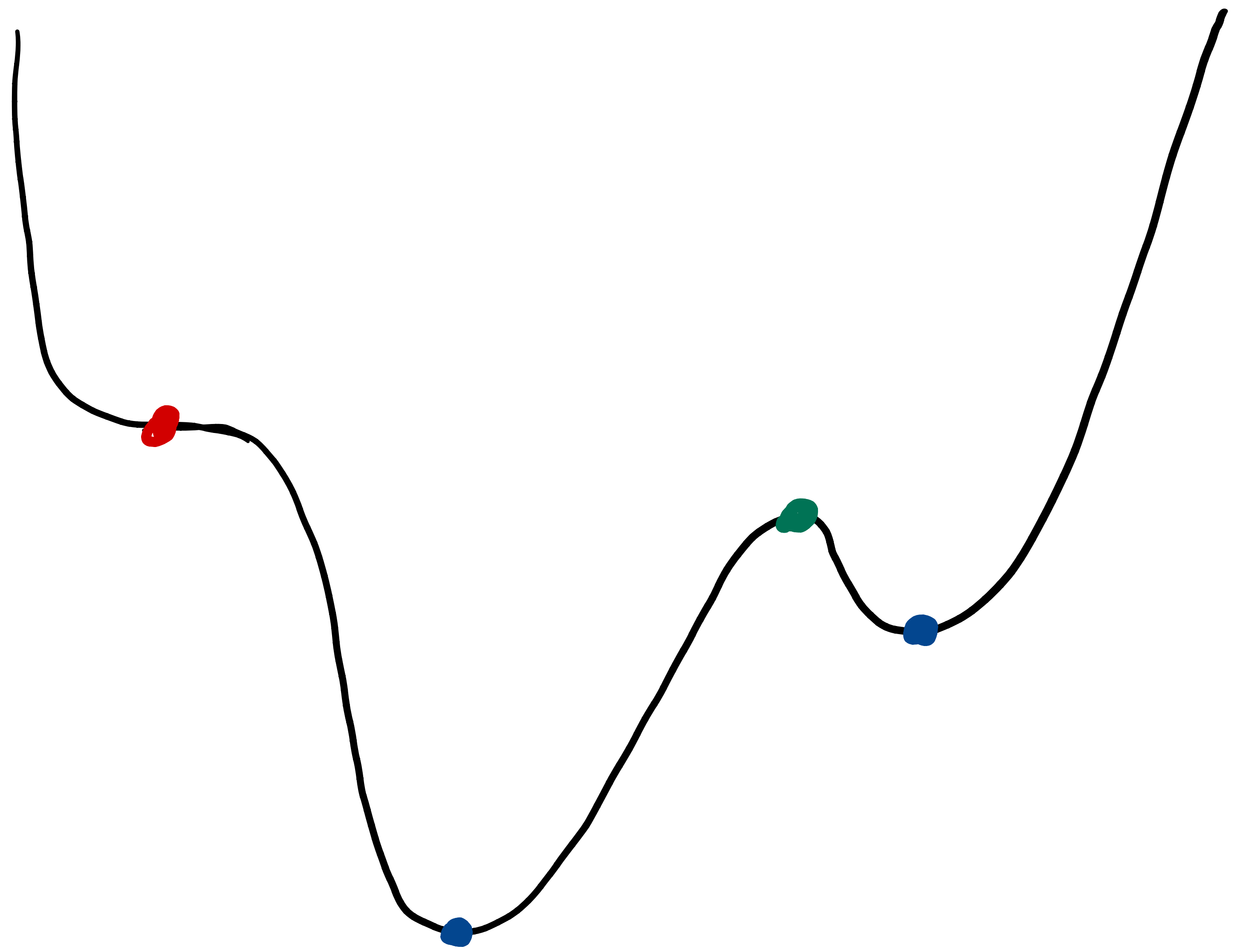
Convex function



Only stationary point

Global min

Nonconvex function

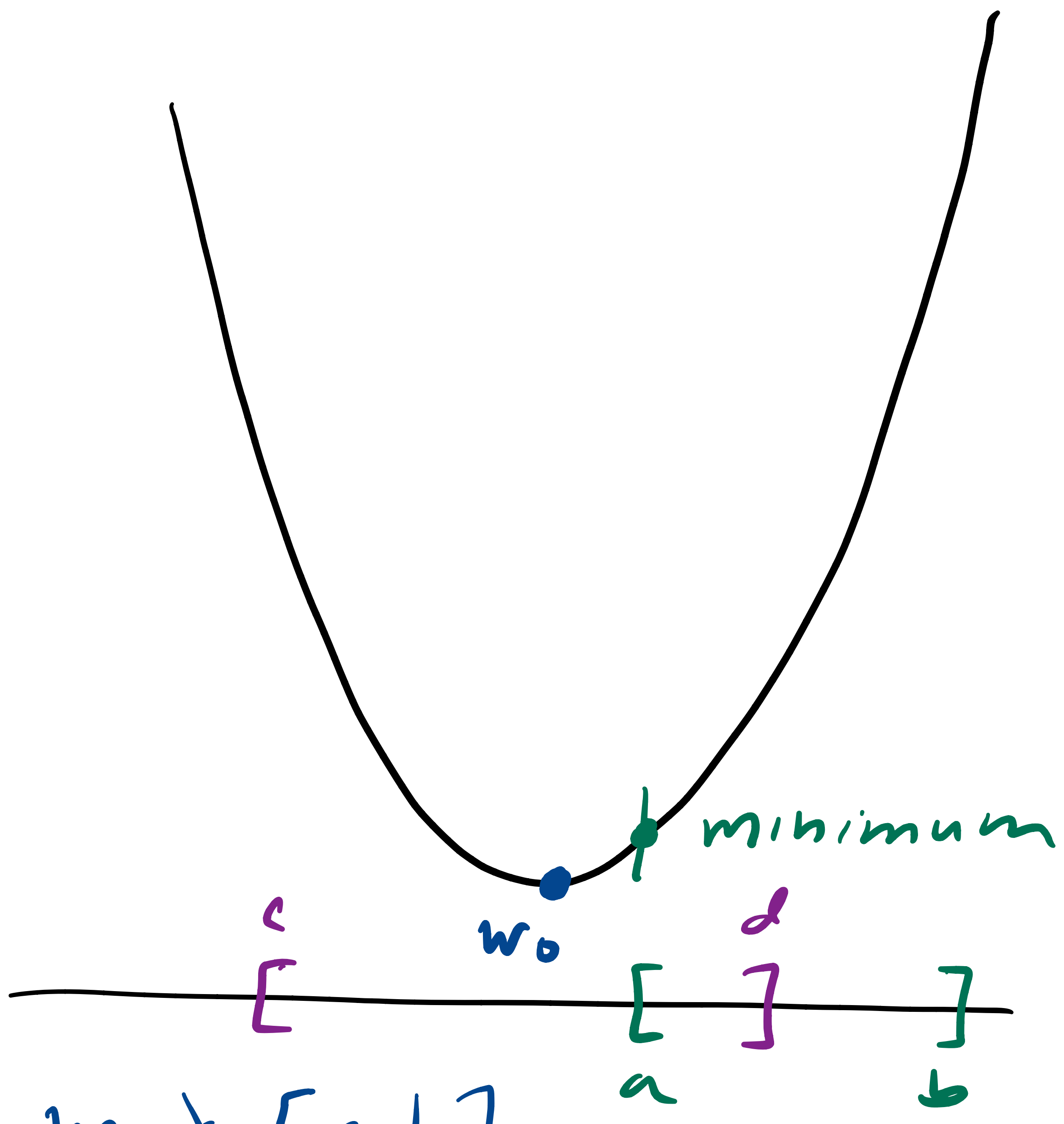


Local minima

Local maximum

Saddlepoint (c'' is 0)

Constraints on w



$w_0 \notin [a,b]$

$w_0 \in [c,d]$

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**