SOICT

HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

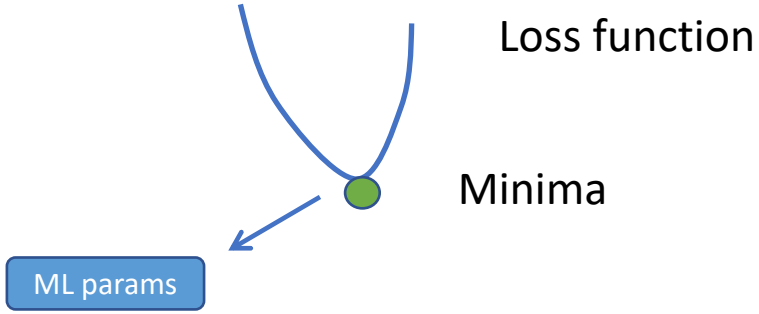
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Optimization in ML

Dam Quang Tuan

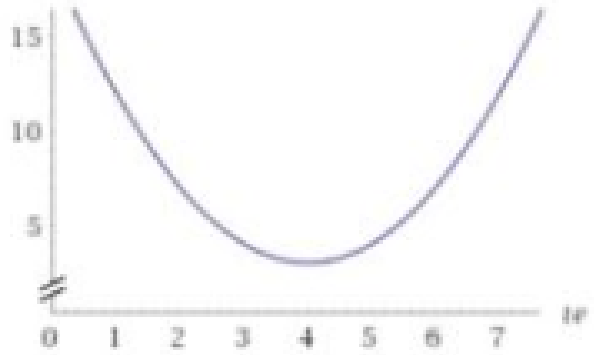
Optimization in ML

- We treat the parameter estimation problem as a problem of function optimization



Example

$$L(w) = 3 + (w - 4)^2$$



- Easy way to find minimum of this function:

$$\frac{\partial}{\partial w} L(w) = 0$$

$$\frac{\partial}{\partial w} L(w) = 2(w - 4)$$

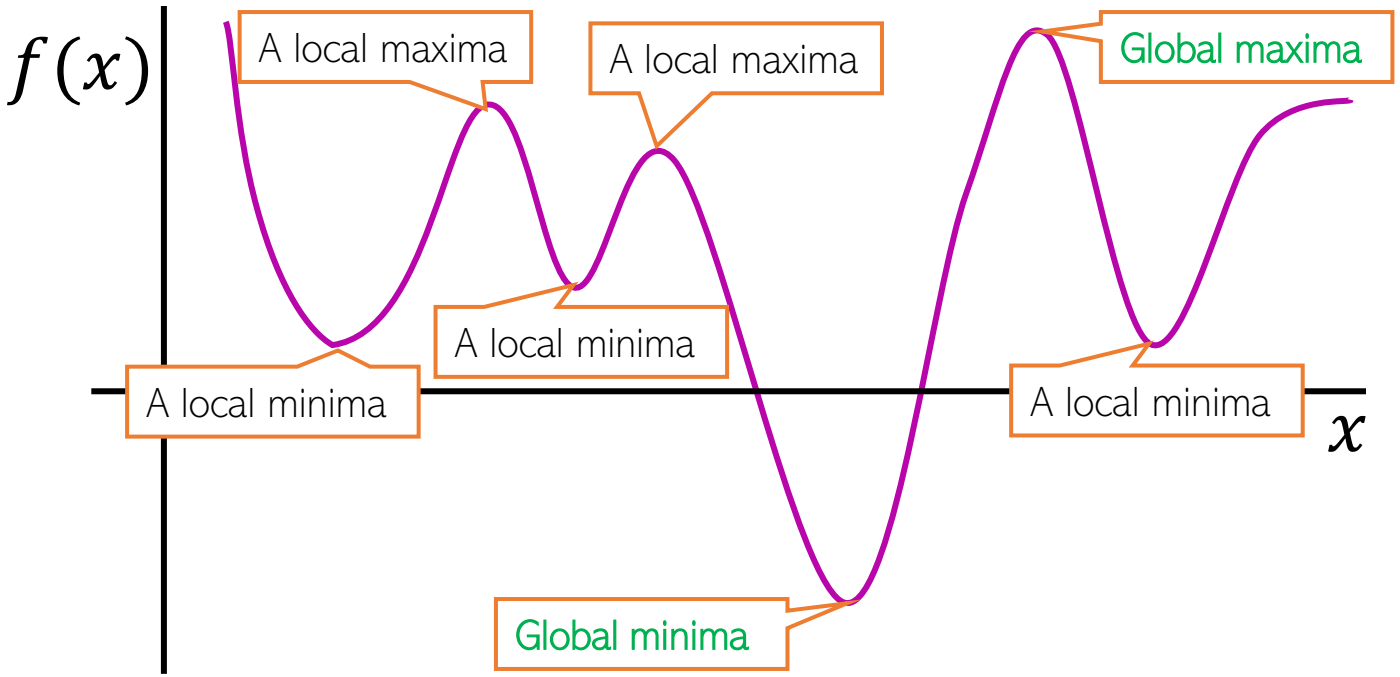
Functions and their optima

- Many ML problems require us to optimize a function f of some variable(s) x
- For example: the objective function of the ML problem we are solving (e.g., squared loss for regression)

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Functions and their optima

- For simplicity, assume f is a scalar-valued function of a scalar x ($f: \mathbb{R} \rightarrow \mathbb{R}$)

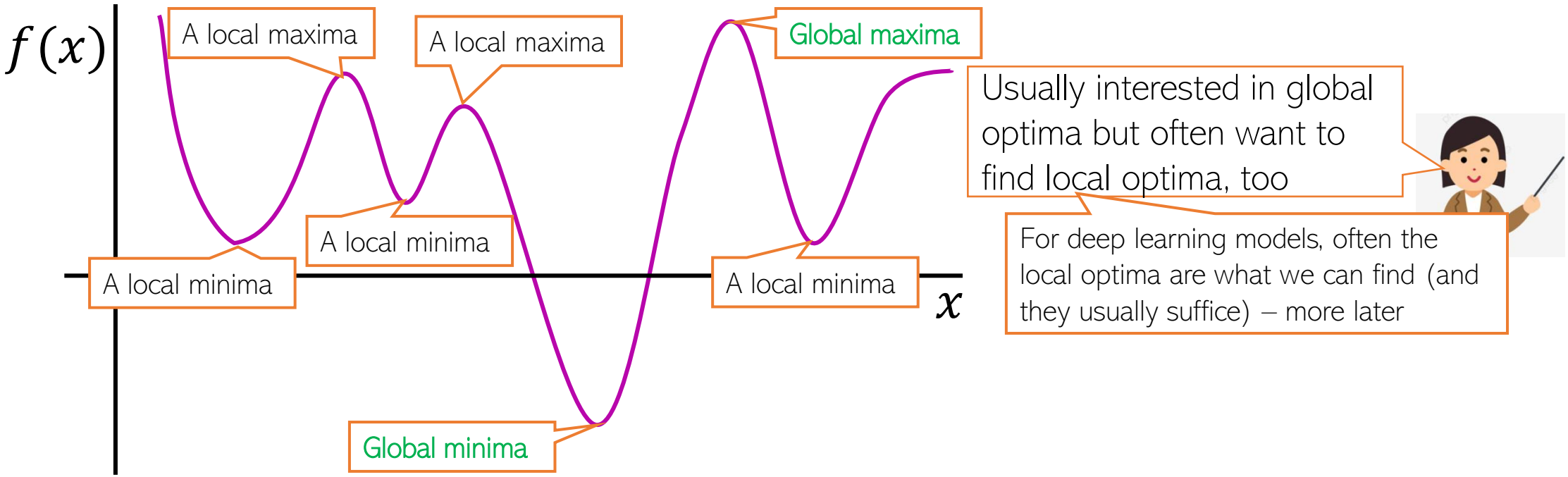


- Any function has one/more optima (maxima, minima), and maybe saddle points

- Finding the optima or saddles requires derivatives/gradients of the function

Functions and their optima

- For simplicity, assume f is a scalar-valued function of a scalar x ($f: \mathbb{R} \rightarrow \mathbb{R}$)



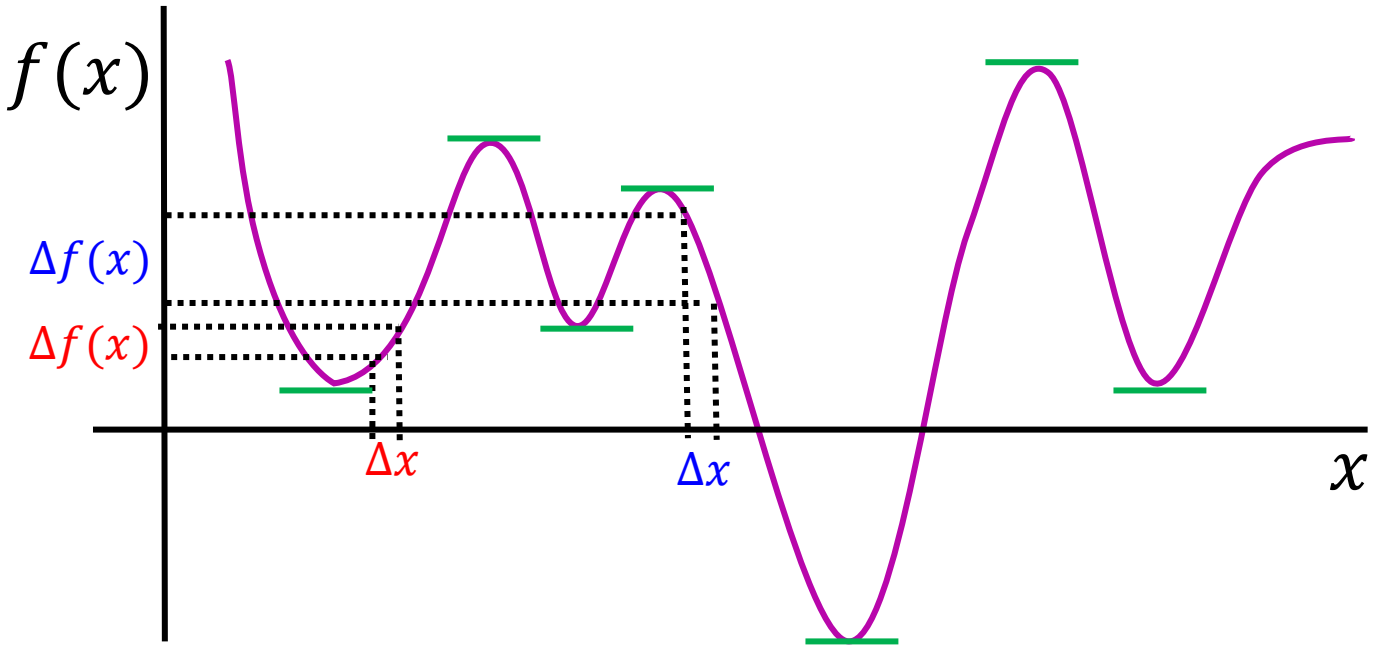
- Any function has one/more optima (maxima, minima), and maybe saddle points

- Finding the optima or saddles requires derivatives/gradients of the function

Derivatives

- Magnitude of derivative at a point is the rate of change of the func at that point

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}$$



Derivatives

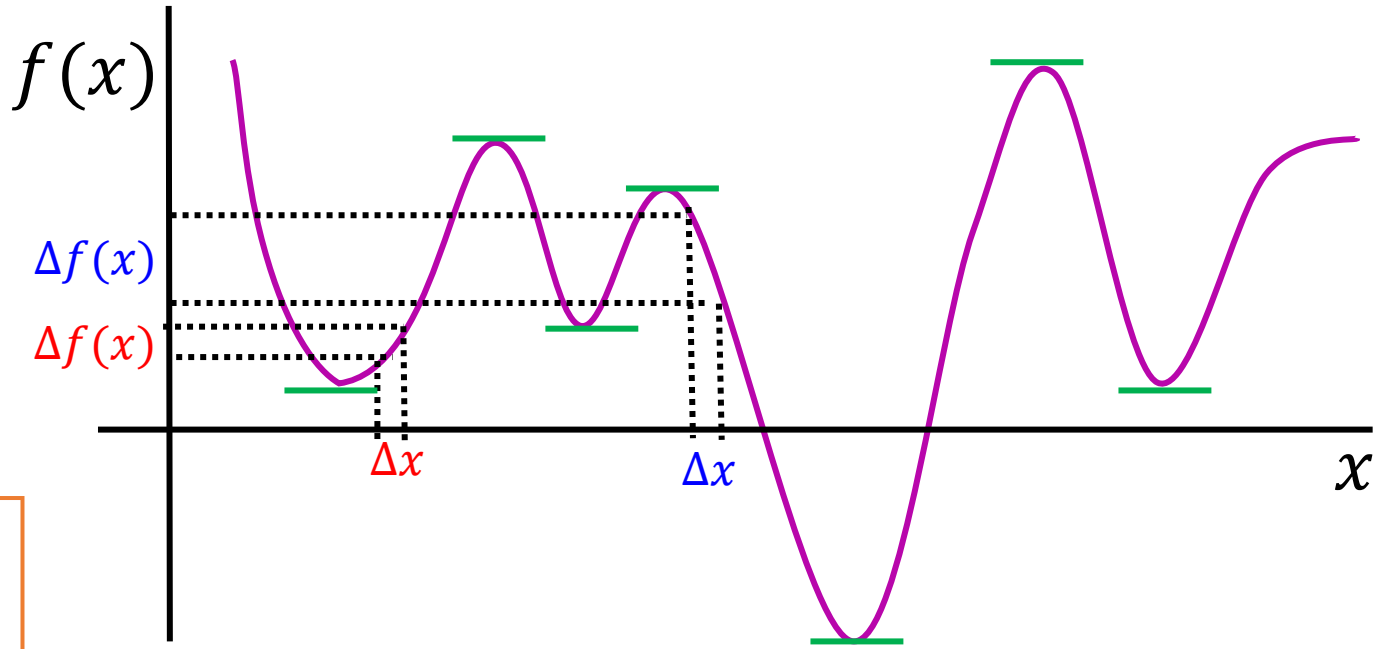
Will sometimes use $f'(x)$ to denote the derivative

- Magnitude of derivative at a point is the rate of change of the func at that point

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}$$

Sign is also important: Positive derivative means f is increasing at x if we increase the value of x by a very small amount; negative derivative means it is decreasing

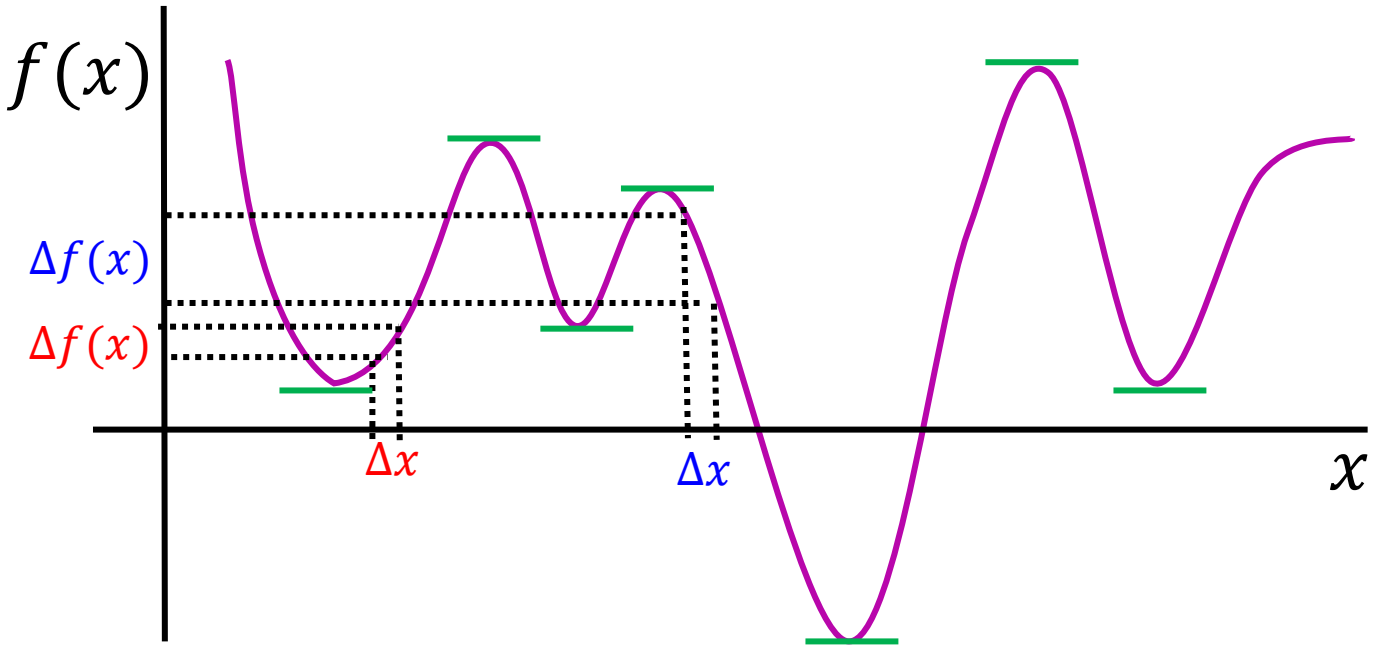
Understanding how f changes its value as we change x is helpful to understand optimization (minimization/maximization) algorithms



Derivatives

- Magnitude of derivative at a point is the rate of change of the func at that point

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}$$

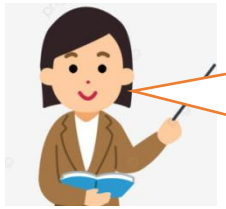


- Derivative becomes zero at stationary points (optima or saddle points)
  - The function becomes "flat" ($\Delta f(x) = 0$ if we change x by a very little at such points)
  - These are the points where the function has its maxima/minima (unless they are saddles)

Rules of Derivatives

Some basic rules of taking derivatives

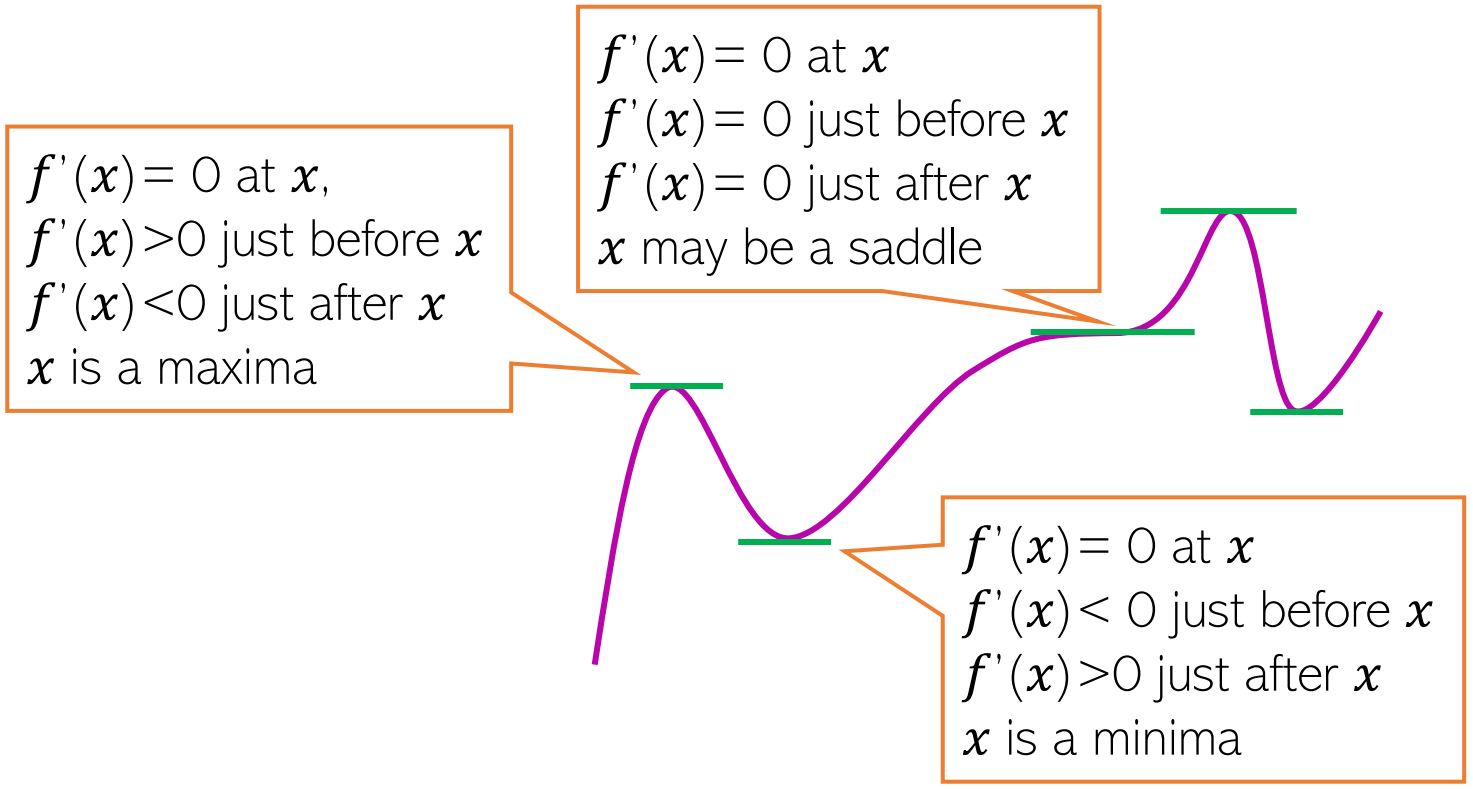
- Sum Rule: $\big(f(x)+g(x)\big)' = f'(x)+g'(x)$
- Scaling Rule: $\big(a \cdot f(x)\big)' = a \cdot f'(x)$ if a is not a function of x
- Product Rule: $\big(f(x) \cdot g(x)\big)' = f'(x) \cdot g(x) + g'(x) \cdot f(x)$
- Quotient Rule: $\big(f(x)/g(x)\big)' = \big(f'(x) \cdot g(x) - g'(x)f(x)\big)/\big(g(x)\big)^2$
- Chain Rule: $\Big(f\big(g(x)\big)\Big)' \stackrel{\text{def}}{=} (f \circ g)'(x) = f'\big(g(x)\big) \cdot g'(x)$



We already used some of these (sum, scaling and chain) when calculating the derivative for the linear regression model

Derivatives

- How the derivative itself changes tells us about the function's optima



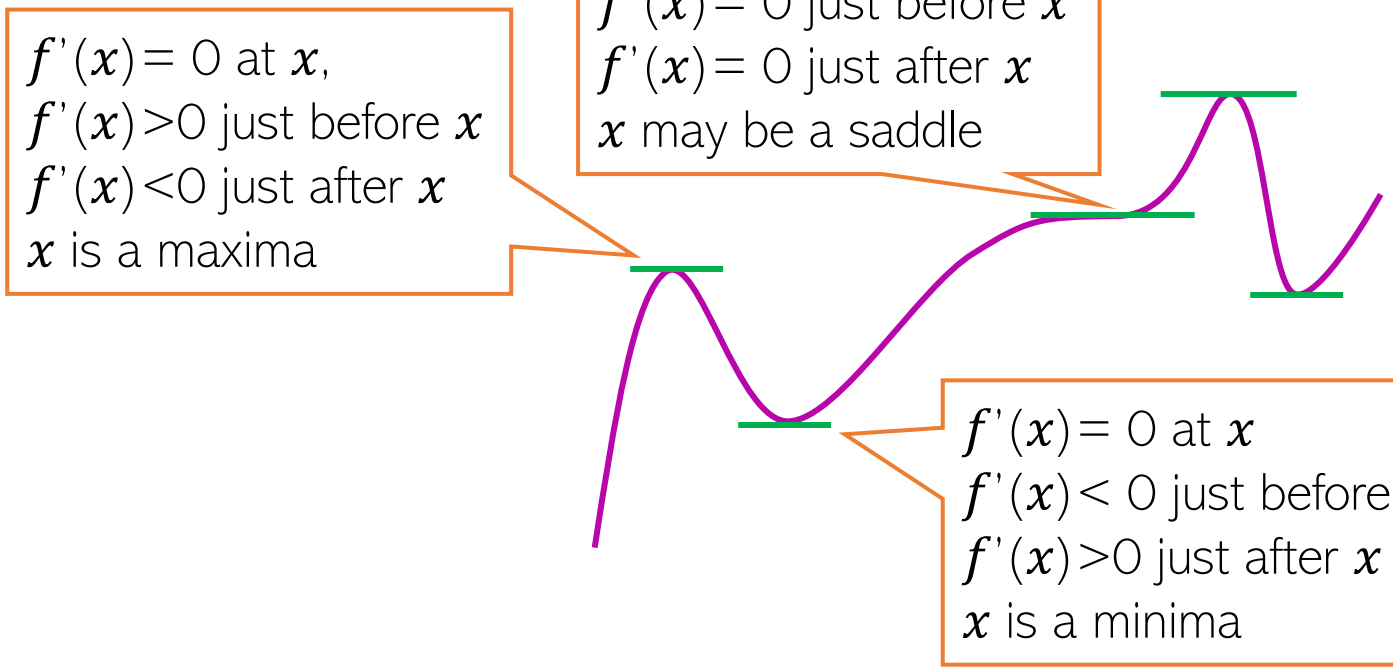
Derivatives

- How the derivative itself changes tells us about the function's optima

$f'(x) = 0$ at x,
$f'(x) > 0$ just before x
$f'(x) < 0$ just after x
x is a maxima

$f'(x) = 0$ at x
$f'(x) = 0$ just before x
$f'(x) = 0$ just after x
x may be a saddle

$f'(x) = 0$ at x
$f'(x) < 0$ just before x
$f'(x) > 0$ just after x
x is a minima



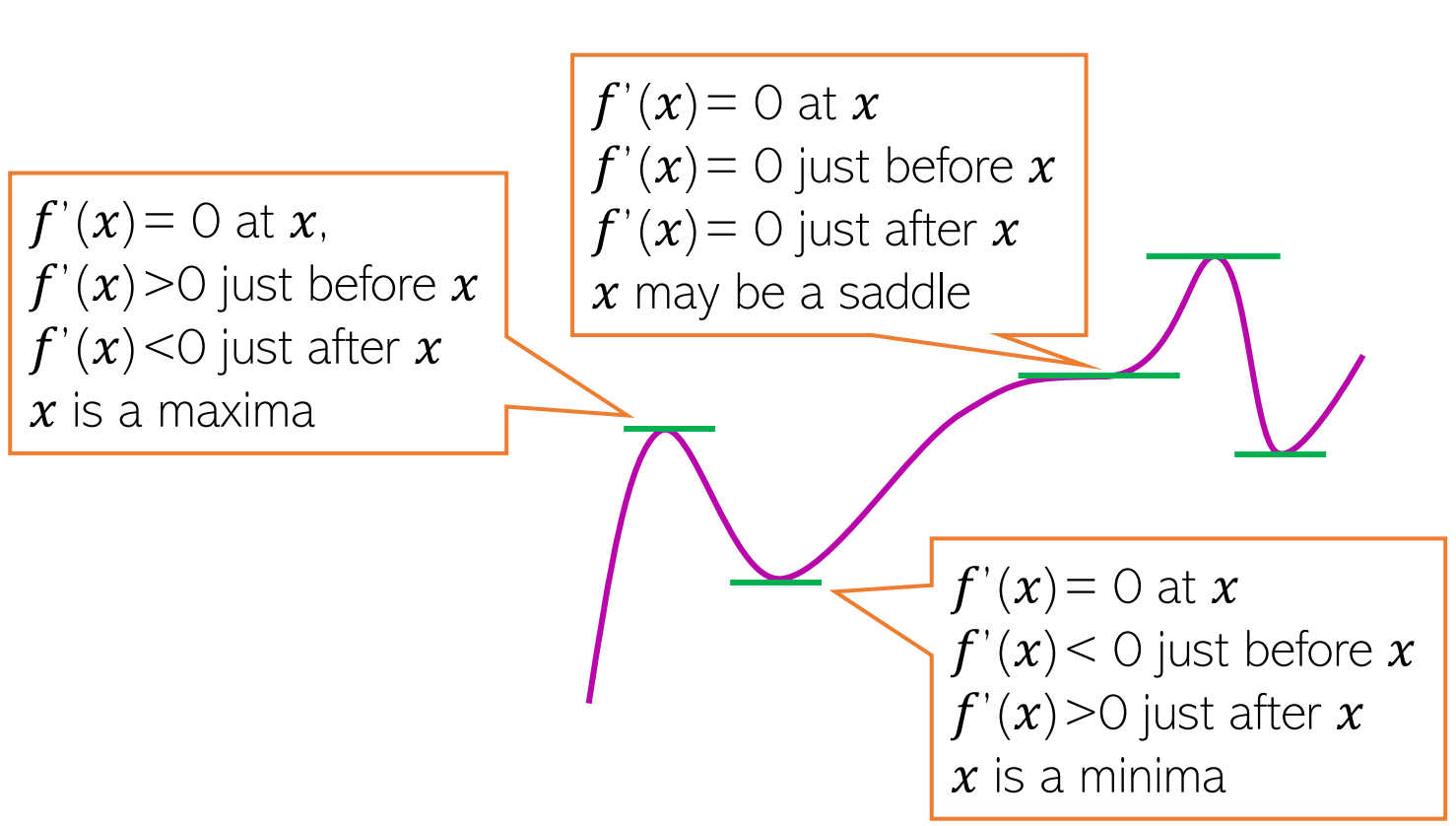
$f'(x) = 0$ and $f''(x) < 0$
x is a maxima

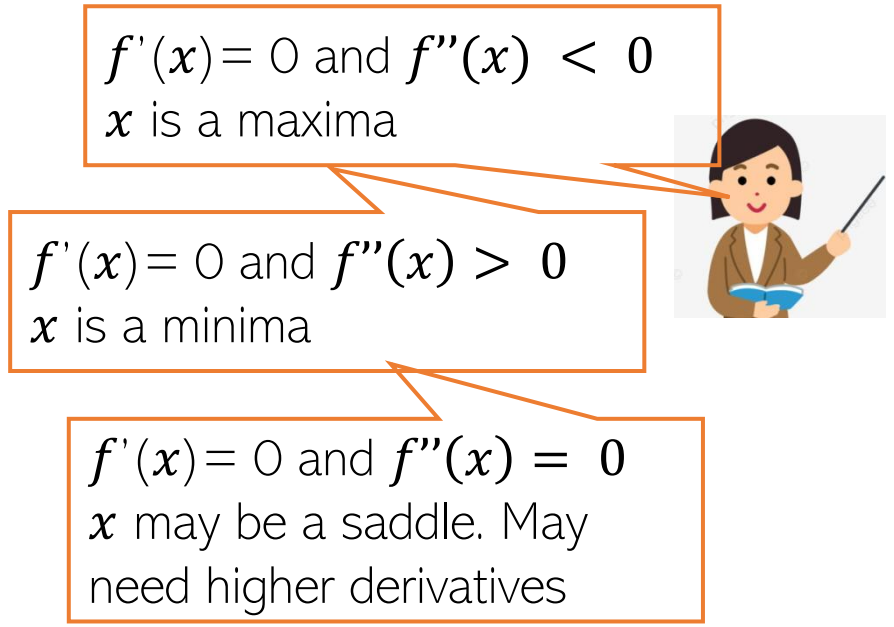
$f'(x) = 0$ and $f''(x) > 0$
x is a minima

$f'(x) = 0$ and $f''(x) = 0$
x may be a saddle. May need higher derivatives

Derivatives

- How the derivative itself changes tells us about the function's optima

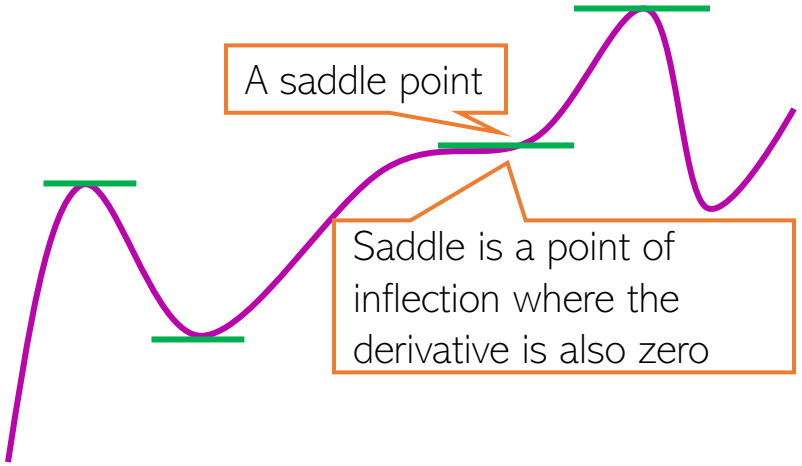


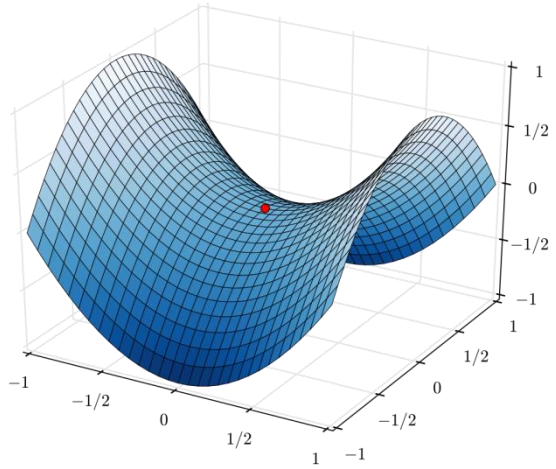


- The second derivative $f''(x)$ can provide this information

Saddle Points

- Points where derivative is zero but are neither minima nor maxima





- Saddle points are very common for loss functions of deep learning models
  - Need to be handled carefully during optimization

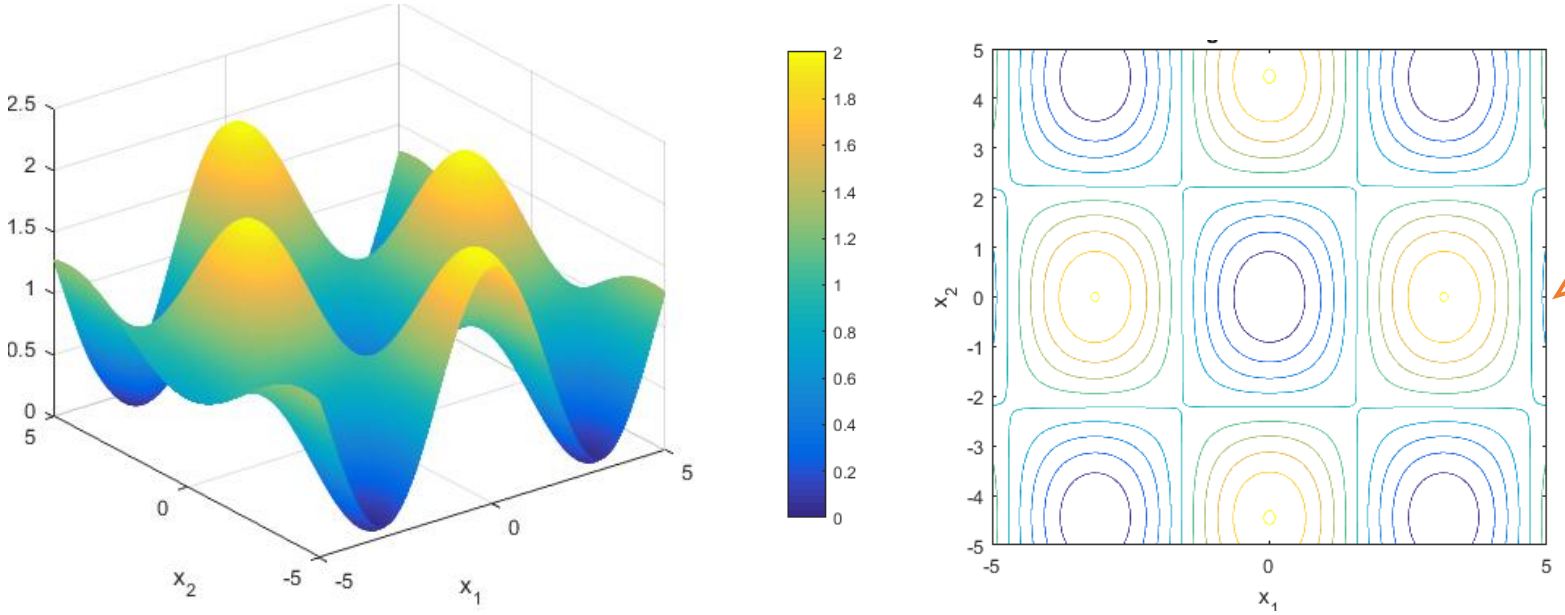
- Second or higher derivative may help identify if a stationary point is a saddle

Multivariate Functions

- Most functions that we see in ML are multivariate function
- Example: Loss fn $\boldsymbol{L(w)}$ in lin-reg was a multivar function of $\boldsymbol{D}$-dim vector $\boldsymbol{w}$

$$\boldsymbol{L(w)}: \mathbb{R}^D \rightarrow \mathbb{R}$$

- Here is an illustration of a function of 2 variables (4 maxima and 5 minima)



Two-dim contour plot of the function (i.e., what it looks like from the above)

Plot courtesy: http://benchmarkfcns.xyz/benchmarkfcns/griewankfcn.html

Derivatives of Multivariate Functions

- Can define derivative for a multivariate functions as well via the gradient
- Gradient of a function $f(\boldsymbol{x})$: $\mathbb{R}^D \rightarrow \mathbb{R}$ is a $D \times 1$ vector of partial derivatives

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_D}\right)$$

Each element in this gradient vector tells us how much f will change if we move a little along the corresponding (akin to one-dim case)

- Optima and saddle points defined similar to one-dim case
  - Required properties that we saw for one-dim case must be satisfied along all the directions
- The second derivative in this case is known as the Hessian

The Hessian

- For a multivar scalar valued function $f(\boldsymbol{x})$: $\mathbb{R}^D \rightarrow \mathbb{R}$, Hessian is a $D \times D$ matrix

$$\nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_D} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_D x_1} & \frac{\partial^2 f}{\partial x_D x_2} & \cdots & \frac{\partial^2 f}{\partial x_D^2} \end{bmatrix}$$

Gives information about the curvature of the function at point $\boldsymbol{x}$

Note: If the function itself is vector valued, e.g., $f(\boldsymbol{x})$: $\mathbb{R}^D \rightarrow \mathbb{R}^K$ then we will have K such $D \times D$ Hessian matrices, one for each output dimension of f

A square, symmetric $D \times D$ matrix M is PSD if $\boldsymbol{x}^\top M\boldsymbol{x} \geq \boldsymbol{0}\ \forall\ \boldsymbol{x} \in \mathbb{R}^D$
Will be NSD if $\boldsymbol{x}^\top M\boldsymbol{x} \leq \boldsymbol{0}\ \forall\ \boldsymbol{x} \in \mathbb{R}^D$

PSD if all eigenvalues are non-negative

- The Hessian matrix can be used to assess the optima/saddle points
  - $\nabla f(\boldsymbol{x}) = 0$ and $\nabla^2 f(\boldsymbol{x})$ is a positive semi-definite (PSD) matrix then $\boldsymbol{x}$ is a minima
  - $\nabla f(\boldsymbol{x}) = 0$, and $\nabla^2 f(\boldsymbol{x})$ is a negative semi-definite (NSD) matrix then $\boldsymbol{x}$ is a maxima

The Hessian: Proof

- The Hessian matrix can be used to assess the optima/saddle points
  - $\nabla f(\boldsymbol{x}) = 0$ and $\nabla^2 f(\boldsymbol{x})$ is a positive semi-definite (PSD) matrix then $\boldsymbol{x}$ is a minima
  - $\nabla f(\boldsymbol{x}) = 0$, and $\nabla^2 f(\boldsymbol{x})$ is a negative semi-definite (NSD) matrix then $\boldsymbol{x}$ is a maxima

Proof

1. Critical Point Condition:

We start with the Taylor series expansion of a twice-differentiable scalar function $f(\mathbf{x})$ around a point $\mathbf{x}_0$:

$$f(\mathbf{x}_0 + \mathbf{h}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top \mathbf{h} + \frac{1}{2}\mathbf{h}^\top \nabla^2 f(\mathbf{x}_0)\mathbf{h} + o(\|\mathbf{h}\|^2).$$

Here:

- $\nabla f(\mathbf{x}_0)$ is the gradient vector (first-order partial derivatives).
- $\nabla^2 f(\mathbf{x}_0)$ is the Hessian matrix (second-order partial derivatives).
- $\mathbf{h}$ is a small perturbation around $\mathbf{x}_0$.

The Hessian: Proof

- The Hessian matrix can be used to assess the optima/saddle points
  - $\nabla f(\boldsymbol{x}) = 0$ and $\nabla^2 f(\boldsymbol{x})$ is a positive semi-definite (PSD) matrix then $\boldsymbol{x}$ is a minima
  - $\nabla f(\boldsymbol{x}) = 0$, and $\nabla^2 f(\boldsymbol{x})$ is a negative semi-definite (NSD) matrix then $\boldsymbol{x}$ is a maxima

Proof

2. Stationary Point Condition:

Given $\nabla f(\mathbf{x}_0) = 0$, the Taylor expansion simplifies to:

$$f(\mathbf{x}_0 + \mathbf{h}) = f(\mathbf{x}_0) + \frac{1}{2}\mathbf{h}^\top \nabla^2 f(\mathbf{x}_0)\mathbf{h} + o(\|\mathbf{h}\|^2).$$

The nature of the critical point $\mathbf{x}_0$ now depends on the **sign of the quadratic term** $\mathbf{h}^\top \nabla^2 f(\mathbf{x}_0)\mathbf{h}$.

3. Positive Semi-Definite (PSD) Case (Local Minimum):

- If $\nabla^2 f(\mathbf{x}_0)$ is **positive semi-definite**, then:

$$\mathbf{h}^\top \nabla^2 f(\mathbf{x}_0)\mathbf{h} \geq 0 \quad \forall \mathbf{h}.$$

- Thus, for any small perturbation $\mathbf{h}$, we get:

$$f(\mathbf{x}_0 + \mathbf{h}) \geq f(\mathbf{x}_0).$$

- This implies that $\mathbf{x}_0$ is a **local minimum**.

The Hessian: Proof

- The Hessian matrix can be used to assess the optima/saddle points
  - $\nabla f(\boldsymbol{x}) = 0$ and $\nabla^2 f(\boldsymbol{x})$ is a positive semi-definite (PSD) matrix then $\boldsymbol{x}$ is a minima
  - $\nabla f(\boldsymbol{x}) = 0$, and $\nabla^2 f(\boldsymbol{x})$ is a negative semi-definite (NSD) matrix then $\boldsymbol{x}$ is a maxima

Proof

4. Negative Semi-Definite (NSD) Case (Local Maximum):

- If $\nabla^2 f(\mathbf{x}_0)$ is **negative semi-definite**, then:

$$\mathbf{h}^\top \nabla^2 f(\mathbf{x}_0)\mathbf{h} \leq 0 \quad \forall \mathbf{h}.$$

- Thus, for any small perturbation $\mathbf{h}$, we get:

$$f(\mathbf{x}_0 + \mathbf{h}) \leq f(\mathbf{x}_0).$$

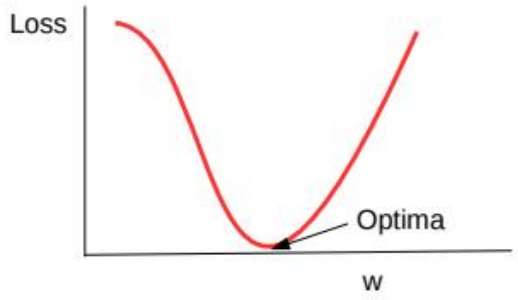
- This implies that $\mathbf{x}_0$ is a **local maximum**.

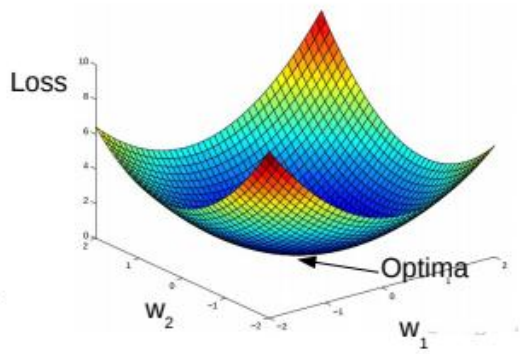
5. Indefinite Case (Saddle Point):

- If the Hessian $\nabla^2 f(\mathbf{x}_0)$ has both positive and negative eigenvalues, the quadratic term can be positive for some $\mathbf{h}$ and negative for others.
- In this case, $\mathbf{x}_0$ is a **saddle point**.

Convex and Non-Convex Functions

- A function being optimized can be either convex or non-convex
- Here are a couple of examples of convex functions

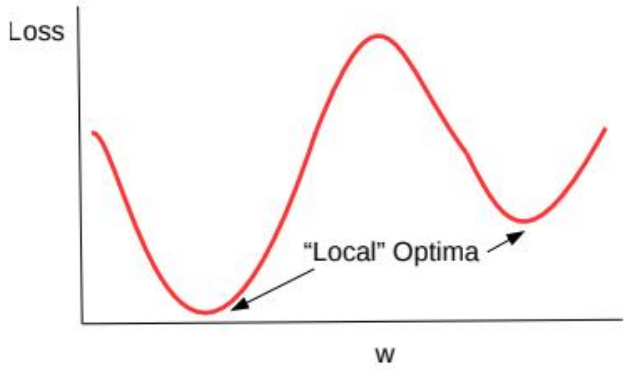


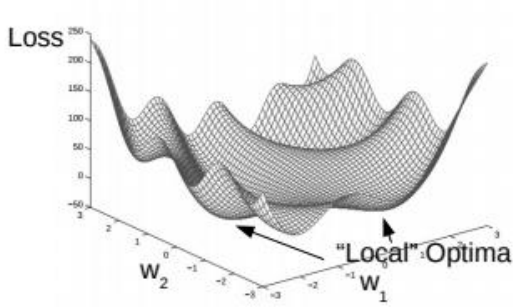


Convex functions are bowl-shaped. They have a unique optima (minima)

Negative of a convex function is called a concave function, which also has a unique optima (maxima)

- Here are a couple of examples of non-convex functions





Non-convex functions have multiple minima. Usually harder to optimize as compared to convex functions

Loss functions of most deep learning models are non-convex

Convex Sets

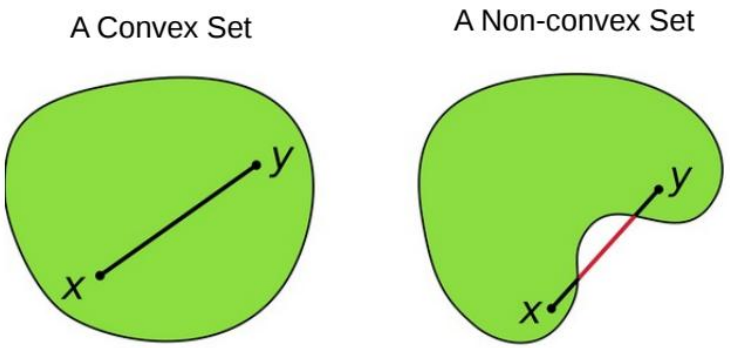
- A set S of points is a convex set, if for any two points $x, y \in S$, and $0 \leq \alpha \leq 1$

z is also called a "convex combination" of two points

$$z = \alpha x + (1 - \alpha) y \in S$$

Can also define convex combination of N points $x_1, x_2, \ldots, x_N$ as $z = \sum_{i=1}^{N} \alpha_i x_i$

- Above means that all points on the line-segment between x and y lie within S

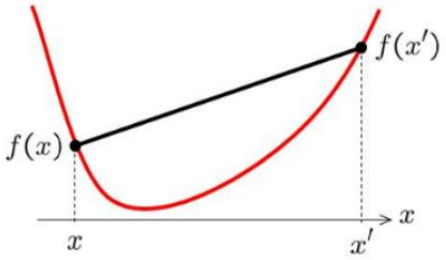


- The domain of a convex function needs to be a convex set

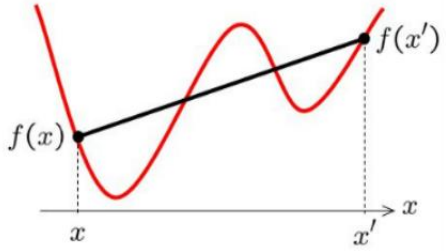
Convex Functions

- Informally, $f(x)$ is convex if all of its chords lie above the function everywhere

Convex Function



Non-convex Function



Note: "Chord lies above function" more formally means

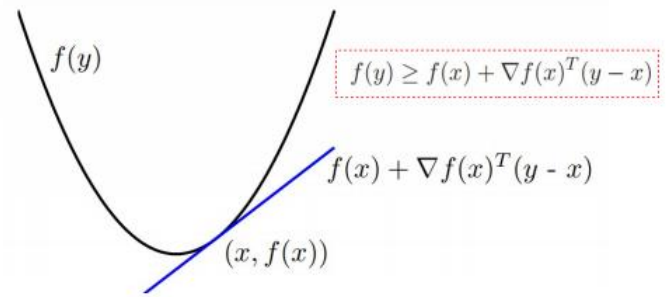
If f is convex then given

$$\alpha_1, \ldots, \alpha_n \text{ s.t } \sum_{i=1}^{n} \alpha_i = 1$$

$$f\left(\sum_{i=1}^{n} \alpha_i x_i\right) \leq \sum_{i=1}^{n} \alpha_i f(x_i)$$

Jensen's Inequality

- Formally, (assuming differentiable function), some tests for convexity:
  - First-order convexity (graph of f must be above all the tangents)

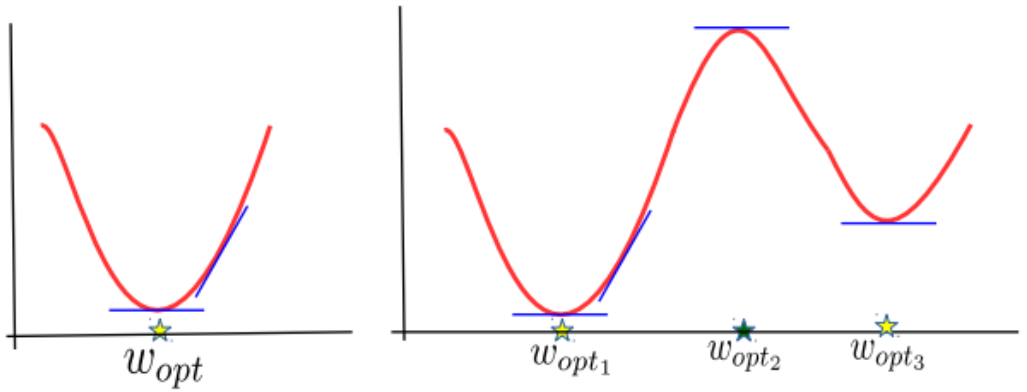


Exercise: Show that ridge regression objective is convex

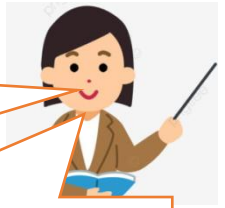
  - Second derivative a.k.a. Hessian (if exists) must be positive semi-definite

Optimization Using First-Order Optimality

- Very simple. Already used this approach for linear and ridge regression



Called "first order" since only gradient is used and gradient provides the first order info about the function being optimized



The approach works only for very simple problems where the objective is convex and there are no constraints on the values $\boldsymbol{w}$ can take

- First order optimality: The gradient $\boldsymbol{g}$ must be equal to zero at the optima

$$\boldsymbol{g} = \nabla_{\boldsymbol{w}}[L(\boldsymbol{w})] = \mathbf{0}$$

- Sometimes, setting $\boldsymbol{g} = \mathbf{0}$ and solving for $\boldsymbol{w}$ gives a closed form solution

- If closed form solution is not available, the gradient vector $\boldsymbol{g}$ can still be used in iterative optimization algos, like gradient descent

Optimization via Gradient Descent

Can I used this approach to solve maximization problems?

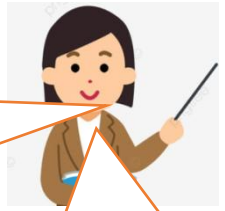
For max. problems we can use gradient ascent

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta_t \boldsymbol{g}^{(t)}$$

Will move in the direction of the gradient

Iterative since it requires several steps/iterations to find the optimal solution

For convex functions, GD will converge to the global minima



Good initialization needed for non-convex functions

Fact: Gradient gives the direction of steepest change in function's value

Gradient Descent

- Initialize $\boldsymbol{w}$ as $\boldsymbol{w}^{(0)}$
- For iteration $t = 0,1,2, \ldots$ (or until convergence)
  - Calculate the gradient $\boldsymbol{g}^{(t)}$ using the current iterates $\boldsymbol{w}^{(t)}$
  - Set the learning rate η_t
  - Move in the opposite direction of gradient

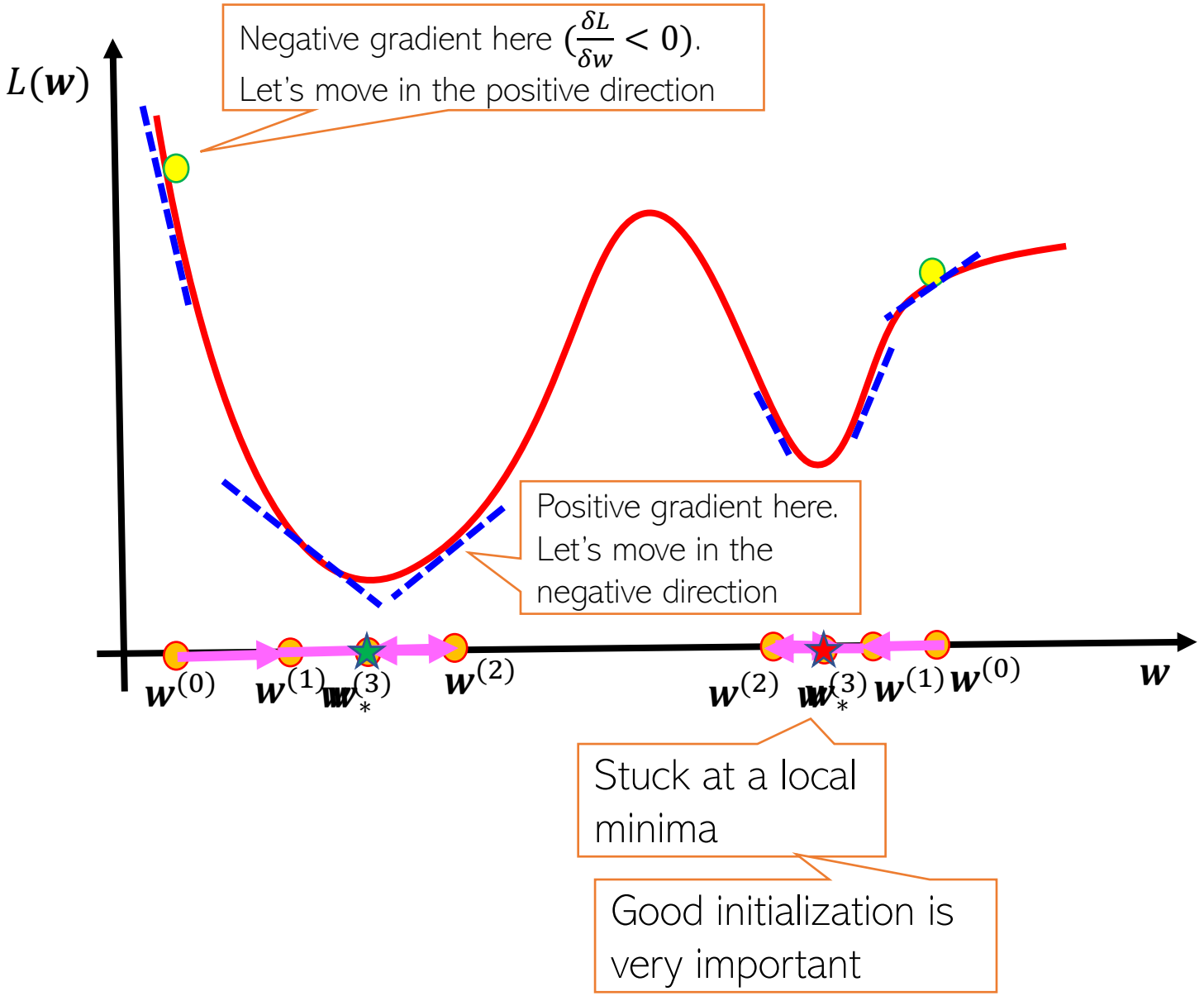
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_t \boldsymbol{g}^{(t)}$$

Will see the justification shortly

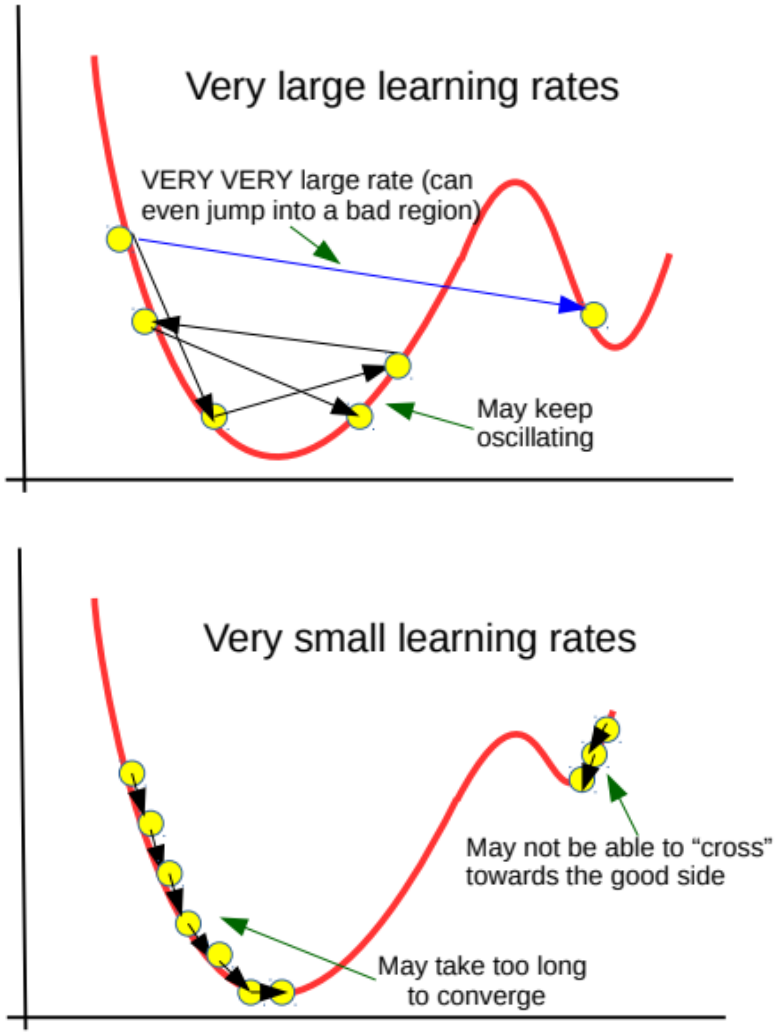
The learning rate very imp. Should be set carefully (fixed or chosen adaptively). Will discuss some strategies later

Sometimes may be tricky to to assess convergence? Will see some methods later

Gradient Descent: An Illustration



Learning rate is very important



GD: An Example

- Let's apply GD for least squares linear regression

$$\boldsymbol{w}_{ridge} = \arg\min_{\boldsymbol{w}} L_{reg}(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

- The gradient: $\boldsymbol{g} = -\sum_{n=1}^{N} 2(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)\boldsymbol{x}_n$

- Each GD update will be of the form

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta_t \sum_{n=1}^{N} 2\left(y_n - \boldsymbol{w}^{(t)\top} \boldsymbol{x}_n\right)\boldsymbol{x}_n$$

Prediction error of current model $\boldsymbol{w}^{(t)}$ on the n^{th} training example

Training examples on which the current model's error is large contribute more to the update

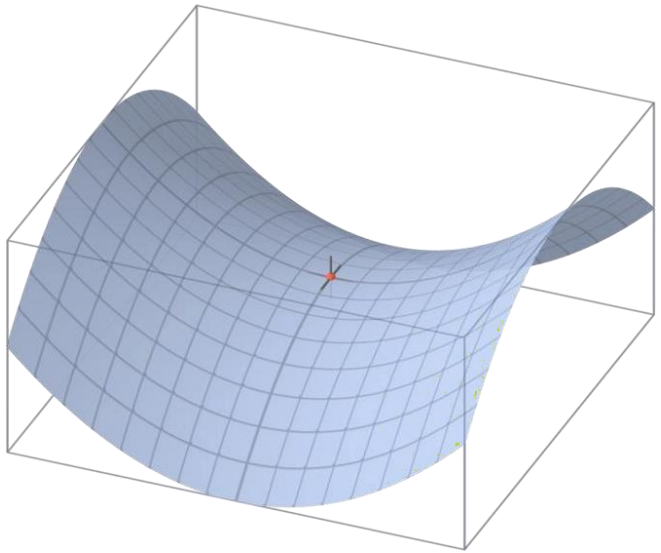
- Exercise: Assume $N = 1$, and show that GD update improves prediction on the training input $(\boldsymbol{x}_n, y_n)$, i.e, y_n is closer to $\boldsymbol{w}^{(t+1)\top} \boldsymbol{x}_n$ than to $\boldsymbol{w}^{(t)\top} \boldsymbol{x}_n$
  - This is sort of a proof that GD updates are "corrective" in nature (and it actually is true not just for linear regression but can also be shown for various other ML models)

Challenges

- Choosing a proper learning rate can be difficult.
  - Small learning rate leads to painfully slow convergence
  - Large learning rate can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge
- Learning rate schedules and thresholds
  - It has to be defined in advance and unable to adapt to a dataset's characteristics.
- Same learning rate applies to all parameter updates.
  - If our data is sparse and our features have very different frequencies, we might not want to update all of them to the same extent, but perform a larger update for rarely occurring features.

Challenges

- Avoiding getting trapped in their numerous suboptimal local minima and saddle points
  - Dauphin et al. argue that the difficulty arises in fact not from local minima but from saddle points.
  - These saddle points are usually surrounded by a plateau of the same error, which makes it notoriously hard for SGD to escape, as the gradient is close to zero in all dimensions.



Momentum

- One of the main limitations of gradient descent) is local minima
  - When the gradient descent algorithm reaches a local minimum, the gradient becomes zero and the weights converge to a sub-optimal solution
- A very popular method to avoid local minima is to compute a temporal average direction in which the weights have been moving recently
  - An easy way to implement this is by using an exponential average

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$$\theta^{new} = \theta^{old} - v_t$$

  - The term γ is called the momentum
  - The momentum has a value between 0 and 1 (typically 0.9)
- Properties
  - Fast convergence
  - Less oscillation

Momentum

- Essentially, when using momentum, we push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way (until it reaches its terminal velocity if there is air resistance).

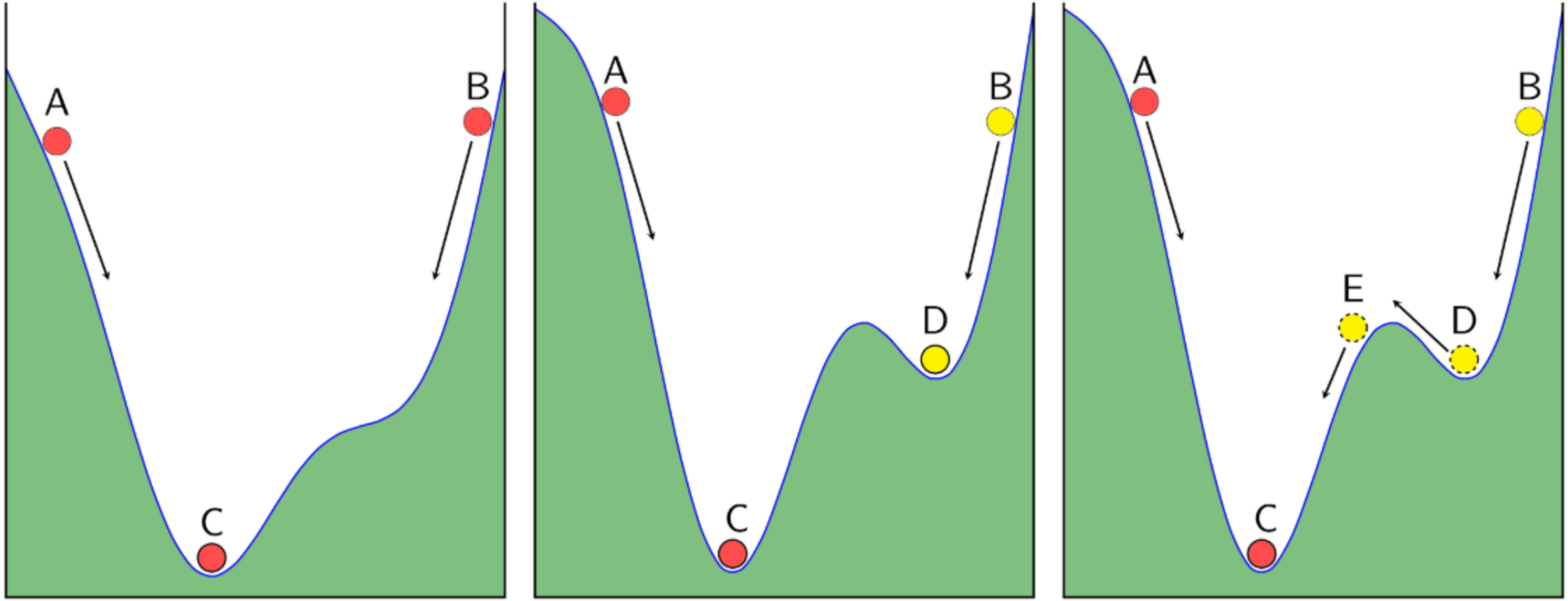
- The same thing happens to our parameter updates: The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.

Momentum

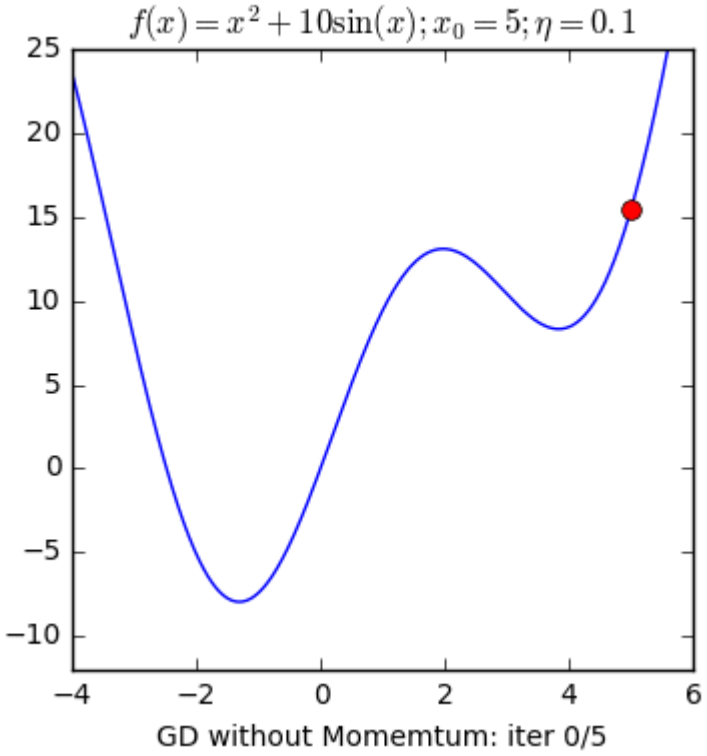
- Essentially, when using momentum, we push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way (until it reaches its terminal velocity if there is air resistance).

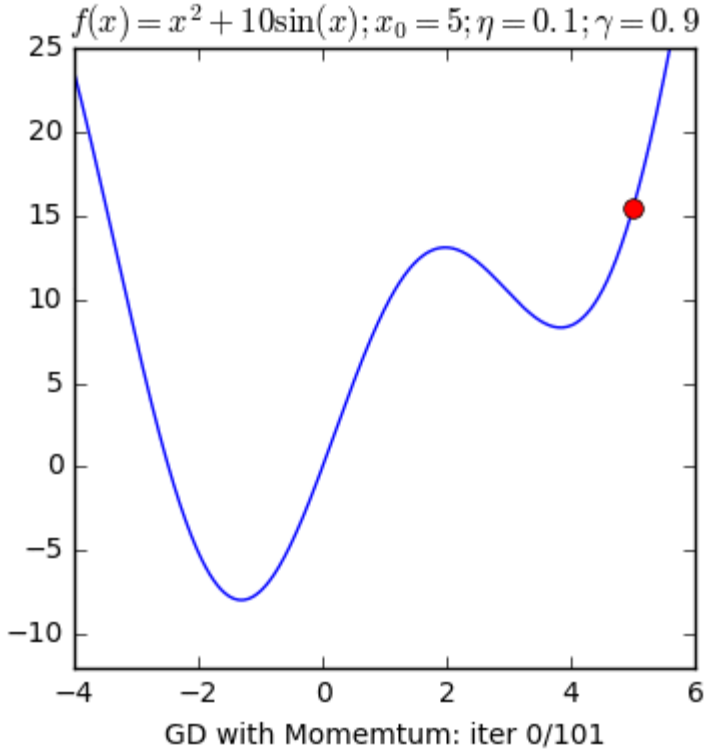
- The same thing happens to our parameter updates: The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.

Momentum



Momentum


$f(x) = x^2 + 10\sin(x); x_0 = 5; \eta = 0.1$
25
20
15
10
5
0
−5
−10
−4
−2
0
2
4
6
GD without Momemtum: iter 0/5


$f(x) = x^2 + 10\sin(x); x_0 = 5; \eta = 0.1; \gamma = 0.9$
25
20
15
10
5
0
−5
−10
−4
−2
0
2
4
6
GD with Momemtum: iter 0/101

Momentum

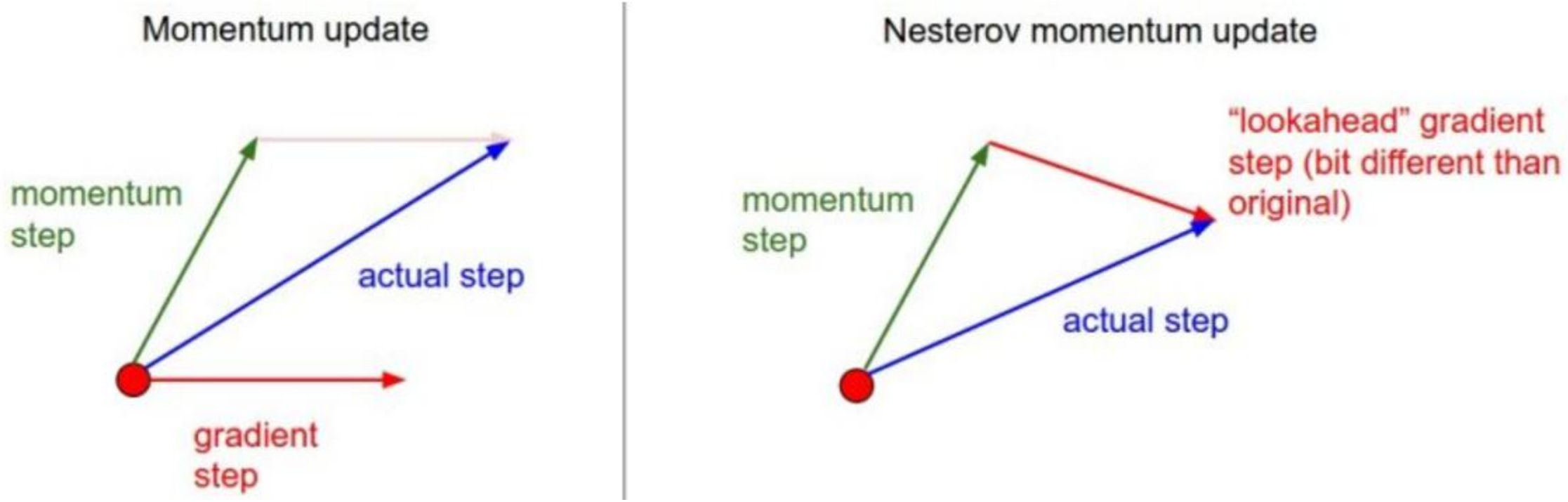
- However, a momentum term can hurt when the search is close to the minima (think of the error surface as a bowl)
  - As the network approaches the bottom of the error surface, it builds enough momentum to propel the weights in the opposite direction, creating an undesirable oscillation that results in slower convergence

NGD (Nesterov accelerated gradient)

- Nesterov accelerated gradient improved on the basis of Momentum algorithm
  - Approximation of the next position of the parameters.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1}).$$

$$\theta^{new} = \theta^{old} - v_t$$



Adagrad

- Adagrad modifies the general learning rate η at each time step t for every parameter θ_i based on the past gradients that have been computed for θ_i:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \frac{\eta}{\sqrt{G_{t,i} + \epsilon}} g_{t,i}$$

$$G_{t,i} = \sum_{k \leq t} \left(g_i^{(k)} \right)^2$$

$$g_i^{(k)} = \frac{\partial J(\theta)}{\partial \theta_i} \Big]_{\theta^{(k)}}$$

Adaptive Gradient Methods

- These methods use a different learning rate for each parameter $\theta_i \in \Re$ at every time step t.
  - For brevity, we set $g^{(t)}$ to be the gradient of the objective function w.r.t. $\theta_i \in \Re$ at time step t:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \boldsymbol{\eta} \cdot g_i^{(t)}$$

- These methods modify the learning rate $\boldsymbol{\eta}$ at each time step (t) for every parameter θ_i based on the past gradients that have been computed for θ_i.

References

CS771: Intro to Machine Learning (Fall 2021), Nisheeth Srivastava, IIT Kanpur