04 Numerical Optimization

May 29, 2021

Information: *Brief introduction to convexity, optimization, and gradient descent*

Written by: *Zihao Xu*

Last update date: *05.29.2021*

1 Convexity

1.1 Introduction

- Convexity plays a vital rule in the design of optimization algorithms, which is largely due to fact that it is much easier to analyze and test algorithms in such a context.
- If the algorithm performs poorly even in the convex setting, typically we should not hope to see great results otherwise.
- Even though the optimization problems in ML/DL are generally non-convex, they often exhibit some properties of convex ones near local minimum.

1.2 Open and Closed Sets

- Define
$$A \subset \mathbb{R}^n$$
and open ball of diameter ϵ is $B(r, \epsilon) = \{r \in \mathbb{R}^n : \|r - r_o\| < \epsilon\}$
- A set A is **open** if
  - At every point, there is an open ball contained in A
  - $\forall r \in A,\ \exists \epsilon > 0$ s.t. $B(r, \epsilon) \subset A$
- A set A is **closed** if $A^c = \mathbb{R}^n - A$ is open
- A set A is compact if it is closed and bounded
- Facts:
  - $\mathbb{R}^N$ is both open and closed, but it is not compact
  - If A is compact, then every sequence in A has a limit point in A

1.3 Convex Sets

1.3.1 Definition

- A set C is convex if, for any $x, y \in C$ and $\theta \in \mathbb{R}$ with $0 \le \theta \le 1$:
$$\theta x + (1 - \theta) y \in C$$
  - Intuitively, it means if we take any two elements in C and draw a line segment between these two elements, then every point on that line segment also belongs to C

- The point $\theta x + (1-\theta)y$ is called a **convex combination** of the points x and y

1.3.2 Examples

- **All of $\mathbb{R}^n$.**
  - Given any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,
$$\theta \mathbf{x} + (1-\theta)\mathbf{y} \in \mathbb{R}^n$$
- **The non-negative orthant $\mathbb{R}^n_+$.**
  - $\mathbb{R}^n_+$ consists of all vectors in $\mathbb{R}^n$ whose elements are all non-negative
$$\mathbb{R}^n_+ = \{\mathbf{x} : x_i \geq 0 \ \forall i = 1, \cdots, n\}$$
  - Given any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N_+$ and $0 \leq \theta \leq 1$,
$$(\theta \mathbf{x} + (1-\theta)\mathbf{y})_i = \theta x_i + (1-\theta) y_i \geq 0 \ \forall i$$
- **Norm balls**
  - Let $\|\cdot\|$ be some norm on $\mathbb{R}^n$ (e.g., the Euclidean norm $\|\mathbf{x}\|_2 = \sqrt{\Sigma_{i=1}^n x_i^2}$). Then the set $\{\mathbf{x} : \|\mathbf{x}\| \leq 1\}$ is a convex set.
  - Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ with $\|\mathbf{x}\| \leq 1, \|\mathbf{y}\| \leq 1$ and $0 \leq \theta \leq 1$. Then
$$\|\theta \mathbf{x} + (1-\theta)\mathbf{y}\| \leq \|\theta \mathbf{x}\| + \|(1-\theta)\mathbf{y}\| = \theta\|\mathbf{x}\| + (1-\theta)\|\mathbf{y}\| \leq 1$$
    where the **triangle inequality** and the **positive homogeneity** of norms are used
- **Affine subspaces and polyhedra**
  - Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{R}^m$, an affine subspace is the set $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}\}$ (note this could possible be empty if $\mathbf{b}$ is not in range of $\mathbf{A}$).
  - Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ s.t. $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{y} = \mathbf{b}$, then for $0 \leq \theta \leq 1$:
$$\mathbf{A}(\theta \mathbf{x} + (1-\theta)\mathbf{y}) = \theta \mathbf{A}\mathbf{x} + (1-\theta)\mathbf{A}\mathbf{y} = \theta \mathbf{b} + (1-\theta)\mathbf{b} = \mathbf{b}$$
  - Similarly, a polyhedron is the set $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \preceq \mathbf{b}\}$ (also possibly empty), where $\preceq$ denotes componentwise inequality
    - All the entries of $\mathbf{A}\mathbf{x}$ are less than or equal to their corresponding element in $\mathbf{b}$
  - Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ that satisfy $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ and $\mathbf{A}\mathbf{y} \leq \mathbf{b}$ and $0 \leq \theta \leq 1$:
$$\mathbf{A}(\theta \mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta \mathbf{b} + (1-\theta)\mathbf{b} = \mathbf{b}$$
- **Intersection of convex sets**
  - Suppose $C_1, C_2, \cdots, C_k$ are convex sets. Then their intersection
$$\bigcap_{i=1}^k C_i = \{x : x \in C_i \ \forall i = 1, \cdots, k\}$$
    is also a convex set
  - Given $x, y \in \bigcap_{i=1}^k C_i$ and $0 \leq \theta \leq 1$. Then
$$\theta x + (1-\theta)y \in C_i \ \forall i = 1, \cdots, k$$
    by the definition of a convex set. Therefore
$$\theta x + (1-\theta)y \in \bigcap_{i=1}^k C_i$$

- Note that the *union* of convex sets in general will not be convex

- **Positive semidefinite matrices**
  - The set of all symmetric positive semidefinite matrices, often times called the *positive semidefinite cone* and denoted $\mathbb{S}^n_+$ is a convex set (in general, $\mathbb{S}^n \subset \mathbb{R}^{n\times n}$ denotes the set of symmetric $n \times n$ matrices).
  - A matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite if and only if $\mathbf{A} = \mathbf{A}^T$ and for all $\mathbf{x} \in \mathbb{R}^n, \mathbf{x}^T\mathbf{A}\mathbf{x} \leq 0$
  - Given two symmetric positive semidefinite matrices $\mathbf{A}, \mathbf{B} \in \mathbb{S}^n_+$ and $0 \leq \theta \leq 1$, then for any $\mathbf{x} \in \mathbb{R}^n$,
$$\mathbf{x}^T(\theta\mathbf{A} + (1-\theta)\mathbf{B})\mathbf{x} = \theta\mathbf{x}^T\mathbf{A}\mathbf{x} + (1-\theta)\mathbf{x}^T\mathbf{B}\mathbf{x} \geq 0$$
  - The logic to show that all **positive definite**, **negative definite**, and **negative semidefinite** matrices are each also convex

1.4 Convex Functions

1.4.1 Definition

- A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain (denoted $\mathcal{D}(f)$) is a *convex set*, and if, for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}(f)$ and $\theta \in \mathbb{R}, 0 \leq \theta \leq 1$:
$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})$$
  - Intuitively, it means if we pick any two points on the graph pf a convex function and draw a straight line between them, then the portion of function between these two points will lie below this straight line.
- A function is called **strictly convex** if the definition holds with strict inequality for $\mathbf{x} \neq \mathbf{y}$ and $0 < \theta < 1$
- A function f is called **concave** if $-f$ is convex
- A function f is called **strictly concave** if $-f$ is strictly convex

1.4.2 First Order Condition for Convexity

- Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable. Then f is convex if and only if $\mathcal{D}(f)$ is a convex set and for $\mathbf{x}, \mathbf{y} \in \mathcal{D}(f)$,
$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla_x f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})$$
where the function $f(\mathbf{x}) + \nabla_x f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})$ is called the **first-order approximation** to the function f at the point $\mathbf{x}$
  - Intuitively, this can be thought of as approximating f with its tangent line at the point $\mathbf{x}$.
- Similarly, f would be
  - strictly convex if this holds with strict inequality
  - concave if the inequality is reversed
  - strictly concave if the reverse inequality is strict

1.4.3 Second Order Condition for Convexity

- Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable. Then f is convex if and only if $\mathcal{D}(f)$ is a convex set and its *Hessian* is positive semidefinite:
$$\forall x \in \mathcal{D}(f),\ \nabla^2_x f(x) \succeq 0$$

  - Here the notation $\succeq$ refers to positive semidefiniteness
  - In one dimension, this is equivalent to the condition that the second derivative $f''(x)$ always be positive
- Similarly, f is
  - strictly convex if its Hessian is positive definite
  - concave if the Hessian is negative semidefinite
  - strictly concave if the Hessian is negative definite

1.4.4 Jensen's Inequality

- Start with the inequality in the basic definition of a convex function

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \text{ for } 0 \leq \theta \leq 1$$

Using induction, extend this definition to convex combinations of more than one point

$$f\left(\sum_{i=1}^{k} \theta_i x_i\right) \leq \sum_{i=1}^{k} \theta_i f(x_i) \;\text{ for } \sum_{i=1}^{k} \theta_i = 1,\; \theta_i \geq 0\; \forall i$$

This can also extend to infinite sums or integrals. In the latter case, the inequality can be written as

$$f\left(\int p(x)x dx\right) \leq \int p(x)f(x)dx \quad \text{for} \quad \int p(x)dx = 1,\; p(x) \geq 0 \;\; \forall x$$

Since $\int p(x)dx = 1$, it is common to consider it a probability density, in which case the previous equation can be written in terms of expectations

$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)]$$

which is called **Jensen's inequality**

1.4.5 Examples

- **Exponential**
  - Let $f : \mathbb{R} \to \mathbb{R}, f(x) = e^{ax}$ for any $a \in \mathbb{R}$.
  - $f''(x) = a^2 e^{ax}$ is positive for all x
- **Negative logarithm**
  - Let $f : \mathbb{R} \to \mathbb{R}, f(x) = -\log x$ with domain $\mathcal{D}(f) = \mathbb{R}_{++} = \{x : x > 0\}$
  - $f''(x) = \frac{1}{x^2} > 0$ for all x
- **Affine functions**
  - Let $f : \mathbb{R}^n \to \mathbb{R}, f(\mathbf{x}) = \mathbf{b}^T\mathbf{x} + c$ for some $\mathbf{b} \in \mathbb{R}^n, c \in \mathbb{R}$
  - The Hessian $\nabla^2_{\mathbf{x}} f(\mathbf{x}) = 0$ for all $\mathbf{x}$
  - Affine functions of this form are the **only** functions that are **both convex and concave**
- **Quadratic function**
  - Let $f : \mathbb{R}^n \to \mathbb{R}, f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$ for a symmetric matrix $\mathbf{A} \in \mathbb{S}^n, b \in \mathbb{R}^n$ and $c \in \mathbb{R}$
  - The Hessian for this function is $\nabla^2_{\mathbf{x}} f(\mathbf{x}) = \mathbf{A}$
  - The convexity or non-convexity of f is determined entirely by whether or not $\mathbf{A}$ is positive semidefinite

- The **squared Euclidean norm** $f(\mathbf{x}) = \|\mathbf{x}\|_2^2 = \mathbf{x}^T\mathbf{x}$ is a special case of quadratic functions where $\mathbf{A} = \mathbf{I}, \mathbf{b} = \mathbf{0}, c = 0$, so it is therefore a **strictly convex function**

- **Norms**
  - Let $f : \mathbb{R}^n \to \mathbb{R}$ be some norm on $\mathbb{R}^n$.
  - By the **triangle inequality** and **positive homogeneity** of norms, for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n, 0 \leq \theta \leq 1$,

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq f(\theta\mathbf{x}) + f((1-\theta)\mathbf{y}) = \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})$$

  - Not possible to prove convexity based on the first or second order conditions because norms are not generally differentiable

- **Nonnegative weighted sums of convex functions**
  - Let $f_1, f_2, \cdots, f_k$ be convex functions and $w_1, w_2, \cdots, w_k$ be nonnegative real numbers. Then

$$f(x) = \sum_{i=1}^{k} w_i f_i(x)$$

    is a convex function, since

$$\begin{aligned}
f(\theta x + (1-\theta)y) &= \sum_{i=1}^{k} w_i f_i(\theta x + (1-\theta)y) \\
&\leq \sum_{i=1}^{k} w_i(\theta f_i(x) + (1-\theta)f_i(y)) \\
&= \theta \sum_{i=1}^{k} w_i f_i(x) + (1-\theta) \sum_{i=1}^{k} w_i f_i(y) \\
&= \theta f(x) + (1-\theta)f(x)
\end{aligned}$$

2 Optimization

2.1 Motivation

- Most ML/DL algorithms involve **optimization** of some sort.
  - Optimization refers to the task of either **minimizing** or maximizing some function $f(\mathbf{x})$ by altering $\mathbf{x}$
  - Usually phrase most optimization problems in terms of minimizing $f(\mathbf{x})$
  - Maximization may be accomplished via s minimization algorithm by minimizing $-f(\mathbf{x})$
- Usually the function we want to minimize is called the **objective function**, or **criterion**. In ML/DL contexts, the name **loss function** is often used.
  - As mentioned in introduction, a loss function quantifies the *distance* between the **real** and **predicted** value of the target.
  - Usually be a non-negative number where smaller values are better and perfect predictions incur a loss of 0
  - Usually denoted as $L(\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is usually the parameter of ML/DL models
- Usually denote the value that minimizes a function with a superscript $*$
  - $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} L(\boldsymbol{\theta})$
- Most ML/DL algorithms are so complex that it is difficult or impossible to find the closed form solution for the optimization problem
  - Use numerical optimization method instead
- One common algorithm is **gradient descent**, other optimization algorithms are
  - Expectation Maximization
  - Sampling-based optimization
  - Greedy optimization

2.2 Local Minimum and Global Minimum

2.2.1 Local Minimum

- Let $f : A \to \mathbb{R}$ where $A \subset \mathbb{R}^N$, a point x is locally minimal if it is available and if there exists some $R < 0$ such that all feasible points z with $\|x - z\|_2 \leq R$, satisfy $f(x) \leq f(z)$
- **Necessary** condition for local minimum
  - Let f be continuously differentiable and let $x \in A$ be a local minimum, then $\nabla f(x) = 0$
- **Saddle Point**:
  - We say that $x \in A$ is a saddle point of f if $\nabla f(x) = 0$ and r_o is not a local minimum

2.2.2 Global Minimum

- A point x is globally minimal if it is available and for all feasible points z, $f(x) \leq f(z)$
  - A global minimum must also be a local minimum

2.3 Optimization Theorems

- Let $f : A \to \mathbb{R}$ where $A \subset \mathbb{R}^N$
  - If f is continuous and A is **compact**, then f takes on a global minimum in A
  - If f is **convex** on A, then any local minimum is a global minimum
  - If f is continuously differentiable and convex on A, then $\nabla f(x) = 0$ implies the $x \in A$ is a global minimum of f

- **Important Facts**:
  - Global minimum **may not be unique**
  - If A is closed but not bounded, then f may not take on a global minimum
  - Most interesting functions in ML/DL are **not** convex

2.4 Convex Optimization

- Formally, a convex optimization problem is an optimization problem of the form

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in C \end{aligned}$$

  where f is a convex function, C is a convex set, and x is the optimization variable
- Often written as

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_i(x) \leq 0 \quad i = 1, \cdots, m \\ & && h_i(x) = 0 \quad i = 1, \cdots, p \end{aligned}$$

  where f is a convex function, g_i are convex functions and h_i are affine functions and x is the optimization variable

2.5 Constrained Optimization

2.5.1 Definition

- Sometimes we wish not only to maximize or minimize a function $f(\mathbf{x})$ over all possible values of $\mathbf{x}$. Instead the maximal or minimal value of $f(\mathbf{x})$ for values of $\mathbf{x}$ in some set $\mathbb{S}$. This is known as **constrained optimization**
- Points $\mathbf{x}$ that lies within the set $\mathbb{S}$ are called **feasible** points in constrained optimization terminology

2.5.2 Solution

- **Intuition**: Design a different, **unconstrained** optimization problem whose solution can be converted into a solution to the original constrained optimization problem
  - For example, to minimize $f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^2$ with $\mathbf{x}$ constrained to have exactly unit L^2 norm, we can instead minimize

$$g(\theta) = f([\cos\theta, \sin\theta]^T)$$

    with respect to θ, then return $[\cos\theta, \sin\theta]$ as the solution to the original problem
  - Requires creativity
  - The transformation between optimization problems must be designed specifically for each case we counter
- **Karush-Kuhn-Tucker** (KKT) approach
  - Provides a very general solution to constrained optimization

3 Gradient Descent

3.1 Basic Concepts

3.1.1 Definition

- **Definition**:
  - A **first-order iterative** optimization algorithm for finding **local minimum** of a **differential** function.
    * The idea is to take *repeated steps* in the opposite direction of the *gradient* of the function at the current point, because this is the direction of steepest descent.
    * As it only calculates the *first-order* derivative, it requires the objective function to be *differential* and is called *first-order optimization algorithms*
      · Some optimization algorithms that also use the Hessian matrix are called *second-order optimization algorithms*
    * Converge when first-order derivative is zero, which only ensures reaching **local minimum** for general functions
      · That is to say, the start point will sometimes affect final convergence
    * Generally speaking, gradient descent algorithms converge to the **global minimum** of continuously differentiable **convex** functions
- **Theory**:
  - Based on the observation that if the multi-variable function $F(\mathbf{x})$ is defined and differentiable in a neighborhood of a point $\mathbf{a}$, then $F(\mathbf{x})$ decreases **fastest** if one goes from $\mathbf{a}$ in the direction of the negative gradient of F at $\mathbf{a}$, which is $-\nabla F(\mathbf{a})$. It follows that if

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

    for a $\gamma \in \mathbb{R}_+$ small enough, then

$$F(\mathbf{a}_n) \geq F(\mathbf{a}_{n+1})$$

- Simple form of **vanilla gradient descent** (GD):
  1. Start at random parameter $\boldsymbol{\theta}$
  2. Repeat until converged
     - $\mathbf{d} \leftarrow -\nabla L(\boldsymbol{\theta})$
     - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \mathbf{d}^T$
  - α is called **learning rate** or **step size**

3.1.2 Compute Loss Gradient

- Take the **mean square error** as an example:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} L_{MSE}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)\|^2 \right\} \\ &= \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \left\{ (\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i))^T (\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \right\}\end{aligned}$$

  Use the chain rule and scale-by-vector matrix calculus identity that

$$\frac{\partial \mathbf{x}^T \mathbf{x}}{\partial \mathbf{x}} = 2\mathbf{x}^T$$

We can get

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} L_{MSE}(\boldsymbol{\theta}) &= \frac{2}{N}\sum_{i=1}^{N}(\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i))^T \nabla_{\boldsymbol{\theta}}(\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \\ &= \frac{2}{N}\sum_{i=1}^{N}(\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i))^T \nabla_{\boldsymbol{\theta}}(-f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \\ &= -\frac{2}{N}\sum_{i=1}^{N}(\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i))^T \nabla_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))\end{aligned}$$

- The result of the gradient usually includes three parts:
  - Sum over training data. It consists of a lot of computations but the way of computation is relatively easy and straight forward
  - Prediction error term such as $\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)$ in MSE, which is usually easy to get
  - Gradient of inference function $\nabla_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))$, which is difficult to solve
    * Enabled by **automatic differentiation** built into modern domain specific languages such as Pytorch, Tensorflow, ...
    * For neural networks, this is known as **back propagation**

3.1.3 Select appropriate learning rate

- Too large α leads to instability and even divergence
- Too small α leads to slow convergence
- **Steepest gradient descent** use **line search** to compute the best α
  1. Start at random parameter $\boldsymbol{\theta}$
  2. Repeat until converged
     - $\mathbf{d} \leftarrow -\nabla L(\boldsymbol{\theta})$
     - $\alpha^* \leftarrow \underset{\alpha}{\operatorname{argmin}}\{L(\boldsymbol{\theta} + \alpha\mathbf{d}^T)\}$
     - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^*\mathbf{d}^T$
- **Adaptive learning rates** may help, but not always
  - $\alpha = \frac{1}{t}$, approaches 0 but can cover an infinite distance since $\lim_{a\to\infty}\sum_{t=1}^{a}\frac{1}{t} = \infty$
- **Coordinate Descent** update one parameter at a time
  - Removes problem of selecting step size
  - Each update can be very fast, but lots of updates

3.1.4 Slow convergence due to Poor Conditioning

- **Conditioning** refers to how rapidly a function changes with respect to small changes in its inputs.
- Consider the function

  $$f(x) = \mathbf{A}^{-1}\mathbf{x}$$

  When $\mathbf{A} \in \mathbb{R}^{n\times n}$ has an eigenvalue decomposition, its **condition number** is

  $$\max_{i,j}\left|\frac{\lambda_i}{\lambda_j}\right|$$

  This is the ratio of the magnitude of the largest and smallest eigenvalue
- A problem with a **low condition number** is said to be **well-conditioned**, while a problem with a high condition number is said to be ill-conditioned

- In non-mathematical terms, an ill-conditioned problem is one where, for a small change in the inputs there is a large change in the answer or dependent variable, which means the correct solution to the equation becomes hard to find
    - Condition number is a property of the problem
- **Gradient descent** is very sensitive to **condition number** of the problem
    - No good choice of step size. Tiny change in one variable could lead to great change in dependent variable.
- **Solutions:**
    - **Newton's method:** Correct for local second derivative.
        * Too much computation and too difficult to implement
        * Harmful when near saddle points
    - **Alternative methods**:
        * Preconditioning: Easy, but tends to be ad-hoc, not so robust
        * Momentum

3.1.5 Vanishing Gradients

- The most insidious problem to encounter
- Some function leads to almost zero gradients far away from local minimums, which makes the optimization stuck for a long time or even stop.
- For example, assume that we want to minimize the function

  $$f(x) = \tanh(x)$$

  The derivative is

  $$f'(x) = 1 - \tanh^2(x)$$

  If we happen to get started at $x = 4$ then the derivative at that point is

  $$f'(4) = 0.0013$$

  The gradient is close to nil. Consequently, optimization will get stuck for a long time before we make progress
- **Possible Solutions**:
    - Reparameterize the problem
    - Good initialization of the parameter
    - Reconstruct the objective function (e.g., change activation function in neural networks)