Optimization for Machine Learning

Pablo Mollá Chárlez

Contents

1 Introduction

Optimization is a fundamental aspect of machine learning and data science, playing a critical role in **model training** and **parameter tuning**. It involves textcolorbluefinding the best solution for a given problem by **minimizing** or **maximizing** an objective function while satisfying specific constraints. From simple regression models to complex deep learning architectures, optimization techniques ensure efficient learning and convergence. This summary outlines the key components of optimization problems and techniques.

2 Optimization Problem

2.1 Formal Definition

- Objective Function: The function $f(x)$ that we aim to maximize or minimize. For example, in machine learning, this could be a prediction error.
- Constraints: Conditions that the solutions must satisfy, including:
  - **Inequality Constraints:** $g(x) \leq 0$
  - **Equality Constraints:** $h(x) = 0$
- Feasible Set: The set of all possible solutions that satisfy the constraints:

  $$S = \{x \in \mathrm{Dom}\ f \mid g_i(x) \leq 0\ \forall i,\ h_j(x) = 0\ \forall j\}.$$

- Optimal Solution: The solution $x^\star$ that maximizes or minimizes the objective function within the feasible set:

  $$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} h_j(x) = 0, & \forall j = 1, \ldots, \ell, \\ g_i(x) \leq 0, & \forall i = 1, \ldots, q. \end{cases}$$

  The optimal solution is given by:

  $$x^\star = \arg\min_{x \in \mathbb{R}^n} f(x).$$

2.2 Challenges in Optimization

- **Global vs. Local Optima:** A global optimum represents the absolute best solution, whereas a local optimum is only the best within a specific neighborhood of the domain of f. Non-convex problems often cause algorithms to get trapped in local optima.
- **Uniqueness:** While a unique solution is ideal, non-convex problems frequently have multiple optima, necessitating careful consideration in selecting the most appropriate solution.
- **Multimodality:** Multimodal functions, characterized by multiple peaks and valleys, pose significant challenges. Algorithms such as gradient descent may converge to a local rather than the global optimum.
- **Convexity:** Convex functions simplify optimization, as any local minimum is also a global minimum. Non-convex functions, however, lack this property, making them harder to optimize.
- **Differentiability:** Differentiable functions, with smooth and continuous slopes, are well-suited for gradient-based methods. Non-differentiable points, such as sharp corners or sudden changes, require alternative approaches like subgradient methods.
- **Curse of Dimensionality:** As the number of variables increases, the complexity of the problem grows significantly, making it more challenging to visualize, analyze, and solve.
- **Non-separability:** Dependencies between optimization variables prevent them from being optimized independently, complicating the optimization process and often requiring joint optimization strategies.

- **Ill-posedness:** An ill-posed problem is one where the solution does not depend continuously on the data, meaning small changes in input can lead to large variations in the output, or the solution might not even exist or be unique. Ill-posedness can lead to instability in optimization, especially in numerical methods.

- **Ill-conditioning:** The function may have a poorly scaled gradient, meaning small changes in certain directions lead to disproportionately large changes in others. This imbalance can cause slow or unstable convergence during optimization, as it makes it difficult for gradient-based methods to find a stable path to the optimum.

  The condition number of a function or optimization problem provides insight into how sensitive the function is to numerical errors:

  - **Well-Conditioned:**
    - Condition number: $\mathcal{K}$ is close to 1 or moderately larger.
    - A condition number of 1 implies that the function or matrix is perfectly well-conditioned (no numerical sensitivity).
    - Typically, a condition number up to 100 or so is considered well-conditioned in practical numerical methods.

    Interpretation: The optimization problem is stable, and numerical algorithms (like gradient descent) are likely to converge smoothly without excessive rounding errors or instability.

  - **Not Ill-Conditioned (Moderately Conditioned):**
    - Condition number: $\mathcal{K}$ is between 100 and 1000.
    - While this range still indicates a function or problem that is not "badly" conditioned, it shows moderate sensitivity to small perturbations in the data or parameters.

    Interpretation: The optimization process might still converge, but care may need to be taken with step sizes and numerical precision. Small perturbations may cause some inaccuracies.

  - **Ill-Conditioned:**
    - Condition number: $\mathcal{K}$ is greater than 1000, and it can grow exponentially as the problem becomes more complex.
    - Functions with a very large condition number (e.g., $\mathcal{K} > 10^6$) are considered ill-conditioned.

    Interpretation: The problem is highly sensitive to small changes or errors in the data or parameters. Numerical methods might experience slow convergence or instability. The optimization process can become highly inefficient, and you may need special techniques like regularization, preconditioning, or smaller step sizes.

3 Sets and Functions

3.1 Convex Sets

A set $S \subset \mathbb{R}^n$ is convex if, for any $x, y \in S$ and $\lambda \in [0,1]$, $\lambda x + (1-\lambda)y \in S$. This means that the line segment connecting any two points within the set lies entirely inside the set.

3.2 Convex Functions

3.2.1 Definition 1

A function f is convex if, for any x, y and $\lambda \in [0,1]$, $f(\lambda \cdot x + (1-\lambda) \cdot y) \leq \lambda \cdot f(x) + (1-\lambda) \cdot f(y)$. This implies that the line segment connecting any two points on the graph of f lies above the graph itself. A twice-differentiable function of a single variable is convex if and only if its second derivative is non-negative on its entire domain.

3.2.2 Definition 2

A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be convex if and only if $\forall x, \theta \in \mathbb{R}^d$, we have that:

$$f(x) \geq f(\theta) + \nabla f(\theta)^T (x - \theta)$$

3.2.3 Strong Convexity

A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be μ-strongly convex if, for all $x, \theta \in \mathbb{R}^d$,

$$f(x) \geq f(\theta) + \nabla f(\theta)^\top (x - \theta) + \frac{\mu}{2} \|x - \theta\|^2,$$

where $\mu > 0$ is the strong convexity parameter.

3.2.4 Properties of Convex Functions

The following operations preserve convexity:

- **Non-Negative Weighted Sum:** If f_1 and f_2 are convex functions, then $\alpha f_1 + \beta f_2$ is also convex for $\alpha, \beta \geq 0$.
- **Composition Rules:** If f is convex and increasing, and g is convex, then the composition $f(g(x))$ is convex.
- **Jensen's Inequality:** If f is a convex function and X is a random variable, then: $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$

3.3 Gradients

The gradient of a differentiable function $f(x)$, denoted as $\nabla f(x)$, is a vector of partial derivatives that represents the rate of change of f with respect to each variable. Mathematically, for $f : \mathbb{R}^n \to \mathbb{R}$,

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right]^T.$$

For a differentiable function $f(x)$, the gradient $\nabla f(x)$ points in the direction of the steepest ascent. To minimize the function, one moves in the direction of $-\nabla f(x)$, which is the direction of steepest descent.

3.4 Hessians

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice-differentiable scalar-valued function. The Hessian of f is a square matrix of second-order partial derivatives, defined as:

$$H(f) = \nabla^2 f(w) = \begin{bmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \frac{\partial^2 f}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{bmatrix}.$$

In compact notation:

$$(H(f))_{ij} = \frac{\partial^2 f}{\partial w_i \partial w_j}, \quad \forall i, j \in \{1, 2, \ldots, n\}.$$

The Hessian provides information about the curvature of f:

- If $H(f)$ is positive semi-definite at z, f is convex at $z \longleftrightarrow \forall z \in R^n, z^T \cdot H_f \cdot z \geq 0$.
- If $H(f)$ is negative definite at z, f is locally concave at z.
- If $H(f)$ is indefinite, z may be a saddle point.

3.4.1 Least Square Function: Convexity, Gradient and Hessian

Let's prove that the least square function is convex, i.e:

$$f(w) = \frac{1}{2}\|y - Xw\|^2,$$

where $y \in \mathbb{R}^m$, $X \in \mathbb{R}^{m\times n}$, and $w \in \mathbb{R}^n$. To prove $f(w)$ is convex, we need to show:

$$f(\lambda w_1 + (1-\lambda)w_2) \leq \lambda f(w_1) + (1-\lambda)f(w_2), \quad \forall w_1, w_2 \in \mathbb{R}^n,\ \lambda \in [0,1].$$

We have:

$$f(\lambda w_1 + (1-\lambda)w_2) = \frac{1}{2}\|y - X(\lambda w_1 + (1-\lambda)w_2)\|^2.$$

Using the linearity of X, this becomes:

$$f(\lambda w_1 + (1-\lambda)w_2) = \frac{1}{2}\|\lambda(y - Xw_1) + (1-\lambda)(y - Xw_2)\|^2.$$

Using the property of the squared norm, $\|u+v\|^2 = \|u\|^2 + \|v\|^2 + 2u^Tv$, we get:

$$f(\lambda w_1 + (1-\lambda)w_2) = \frac{1}{2}\left(\lambda^2\|y - Xw_1\|^2 + (1-\lambda)^2\|y - Xw_2\|^2 + 2\lambda(1-\lambda)(y - Xw_1)^T(y - Xw_2)\right).$$

Next, we calculate:

$$\lambda f(w_1) + (1-\lambda)f(w_2) = \lambda \cdot \frac{1}{2}\|y - Xw_1\|^2 + (1-\lambda)\cdot\frac{1}{2}\|y - Xw_2\|^2.$$

This simplifies to:

$$\lambda f(w_1) + (1-\lambda)f(w_2) = \frac{1}{2}\left(\lambda\|y - Xw_1\|^2 + (1-\lambda)\|y - Xw_2\|^2\right).$$

Now, compute the difference:

$$\text{Difference} := f(\lambda w_1 + (1-\lambda)w_2) - \left(\lambda f(w_1) + (1-\lambda)f(w_2)\right).$$

Substituting the expressions, we get:

$$\text{Difference} = \frac{1}{2}\left(\lambda^2\|y - Xw_1\|^2 + (1-\lambda)^2\|y - Xw_2\|^2 + 2\lambda(1-\lambda)(y - Xw_1)^T(y - Xw_2)\right)$$

$$-\frac{1}{2}\left(\lambda\|y - Xw_1\|^2 + (1-\lambda)\|y - Xw_2\|^2\right).$$

After some algebra and factoring out $\lambda(1-\lambda)$, the difference becomes:

$$\text{Difference} = -\frac{\lambda(1-\lambda)}{2}\|Xw_1 - Xw_2\|^2.$$

Since $\|Xw_1 - Xw_2\|^2 \geq 0$ and $\lambda(1-\lambda) \geq 0$ for $\lambda \in [0,1]$, the difference is non-positive. Hence:

$$f(\lambda w_1 + (1-\lambda)w_2) \leq \lambda f(w_1) + (1-\lambda)f(w_2).$$

This proves that $f(w)$ is convex. The gradient calculation details of the function can be found on the exercise sheet, although here you can find the results for the gradient and the hessian of the least square function.

- **Gradient**: $\nabla f(x) = -X^T \cdot (y - Xw)$
- **Hessian**: $H_f(w) = X^TX$

3.5 Taylor's Expansion

3.5.1 Taylor's Expansion in $\mathbb{R}$

Let$k$ be a natural number, $x_0 \in \mathbb{R}$, and f a function that is k-times continuously differentiable on an interval containing x_0 and x. Taylor's theorem states that there exists some $\xi \in (x_0, x)$ such that:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(k)}(\xi)}{k!}(x - x_0)^k.$$

Implication: Taylor's theorem allows us to approximate $f(x)$ around x_0 using increasingly accurate terms based on the derivatives of f at x_0.

3.5.2 Taylor's Expansion in $\mathbb{R}^n$

For a function $f : \mathbb{R}^n \to \mathbb{R}$ that is continuously twice differentiable, the Taylor approximation around a point $x_0 \in \mathbb{R}^n$ is given by:

$$f(x) = f(x_0) + \nabla f(x_0)^T(x - x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0) + R_3(x),$$

where $R_3(x)$ is the remainder term:

$$R_3(x) = O(\|x - x_0\|^3), \quad \text{which vanishes as } x \to x_0.$$

3.6 Descent Direction

The concept of a descent direction identifies directions $\mathbf{d}$ in which the function f decreases locally. Let $\mathbf{x}$ be a point in the domain of f such that $\nabla f(\mathbf{x}) \neq 0$, meaning $\mathbf{x}$ is not a critical point of f. A vector $\mathbf{d} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is a descent direction for f at $\mathbf{x}$ if there exists $\bar{\alpha} > 0$ such that:

$$f(\mathbf{x} + \alpha\mathbf{d}) < f(\mathbf{x}), \quad \forall \alpha \in (0, \bar{\alpha}).$$

This means that f strictly decreases along the half-line $\{\mathbf{x} + \alpha\mathbf{d} : \alpha > 0\}$ for sufficiently small step sizes $\alpha > 0$.

3.6.1 Descent Direction Lemma

Let x be a non-critical point of f $(\nabla f(x) \neq 0)$, and $d \in \mathbb{R}^n - \{0\}$. If $\nabla f(x)^T d < 0$, then d is a descent direction for f at x.

Proof

Since f is differentiable, by the first-order Taylor expansion theorem, we can approximate $f(x + \alpha d)$ for small $\alpha > 0$ as:

$$f(x + \alpha d) = f(x) + \alpha \nabla f(x)^\top d + o(\alpha),$$

where $o(\alpha)$ represents higher-order terms that vanish as $\alpha \to 0$.
If $\nabla f(x)^\top d < 0$, then for small $\alpha > 0$, the term $\alpha \nabla f(x)^\top d$ is negative, implying:

$$f(x + \alpha d) < f(x).$$

Therefore, d is a descent direction for f at x.

3.6.2 Steepest Descent Direction

The (unnormalized) direction $\mathbf{d} = -\nabla f(x)$ (anti-gradient) is called the steepest descent direction of f at x, as it yields the greatest decrease in f.

3.6.3 Gradient Descent Algorithm

To minimize a differentiable function f, the Gradient Descent algorithm operates with the following sequence of iterates:

- **Initialization:** Start with an initial point $x^{(0)}$.
- **Iteration:** For $k = 0, 1, 2, \ldots$:

  $$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \mathbf{d}^{(k)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}),$$

  Where:

  - $\mathbf{d}^{(k)} = -\nabla f(x^{(k)})$: The descent direction (negative gradient).
  - $\alpha^{(k)}$: The step size (learning rate).

The iterations continue until a stopping criterion is reached, such as when $\|\nabla f(x^{(k)})\|$ is sufficiently small or the change in $f(x^{(k)})$ becomes negligible.

3.6.4 L-Smoothness

A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be L-smooth if and only if,

$$f(x) \leq f(\theta) + \nabla f(\theta)^\top (x - \theta) + \frac{L}{2} \|x - \theta\|^2 \quad \forall \theta, x \in \mathbb{R}^d$$

where $L > 0$ is the smoothness parameter.

3.6.5 Condition Number

The condition number κ measures how "well-conditioned" the optimization problem is. When a function $f : \mathbb{R}^n \to \mathbb{R}$ is both L-smooth and μ-strongly convex, we define its condition number κ as:

$$\kappa = \frac{L}{\mu},$$

where L is the smoothness constant and μ is the strong convexity constant. When $L = \mu$, the function is **perfectly conditioned** ($\sim \kappa = 1$). Besides, a small condition number $\kappa \approx 1$ results in fast convergence and, a large condition number $\kappa >> 1$ leads to slow convergence and oscillations (zigzag). In terms of Linear Algebra, the condition number of a matrix A is given by:

$$\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}},$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of A.

3.6.6 Level Sets

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, the level set of f corresponding to a scalar $c \in \mathbb{R}$ is the set of all points $x \in \mathbb{R}^n$ such that:

$$L_c = \{x \in \mathbb{R}^n \mid f(x) = c\}.$$

4 Convergence Rates

4.1 Definition and Importance

The convergence rate describes how quickly an optimization algorithm approaches the optimal solution. It is an important metric for comparing algorithms, as faster convergence leads to fewer iterations and faster solutions.

4.2 Types of Convergence

4.2.1 Linear Convergence:

Linear convergence occurs when the error decreases by a constant fraction with each iteration. Formally, this can be expressed as:

$$\|x^{(k+1)} - x^*\| \leq c \cdot \|x^{(k)} - x^*\|,$$

where c is a constant that represents the rate of error decrease, satisfying $0 < c < 1$. A smaller c indicates faster convergence.

4.2.2 Quadratic Convergence:

Quadratic convergence occurs when the error decreases by the square of the previous error at each iteration, leading to very rapid convergence near the solution. Formally, this is expressed as:

$$\|x^{(k+1)} - x^*\| \leq c \cdot \|x^{(k)} - x^*\|^2,$$

where c is a positive constant ($c > 0$). Quadratic convergence is significantly faster than linear convergence, especially near the optimal solution.

4.3 Theorem: Convergence Rate of Gradient Descent for Strongly Convex Functions

Assume f is L-smooth and μ-strongly convex. For gradient descent with a fixed step size $\alpha_k = \frac{1}{L}$, the iterates $(x_k)_{k\geq 0}$ satisfy:

$$f(x_k) - f(x^*) \leq e^{-\frac{k\mu}{L}} \cdot (f(x_0) - f(x^*)),$$

where:

- x^* is the minimizer of f,
- $\frac{\mu}{L}$ determines the rate of convergence and depends on the condition number $\kappa = \frac{L}{\mu}$.

Gradient descent therefore achieves exponential (linear in log-scale) convergence rate for strongly convex functions.

4.4 Theorem: Convergence of Gradient Descent for Smooth and Convex Functions

For a convex and L-smooth function f, gradient descent with a step size $\alpha = \frac{1}{L}$ satisfies:

$$f(x_k) - f(x^*) = O\left(\frac{1}{k}\right),$$

where x^* is the minimizer of f. If f is only assumed to be smooth and convex, gradient descent with a constant step size $\alpha = \frac{1}{L}$ still converges, but at a slower rate (sublinear rate).

4.5 Rate Convergence Algorithms

4.5.1 Gradient Descent (GD)

The basic gradient descent algorithm updates the current value of x by moving against the gradient of the function f with a fixed step size α.

```
Initialize $x = x_0$ (starting point)
Set learning rate $\alpha$
Repeat until convergence:
    Compute: $\nabla f(x)$
    Update: $x_{k+1} = x_k - \alpha \cdot \nabla f(x)$
    If convergence criterion is met:
        Break
```

4.5.2 Classical Momentum

Classical momentum introduces a velocity term v that helps the algorithm gain speed along the direction of the gradient and avoid oscillations.

```
Initialize $x = x_0, v = 0$ (starting point and velocity)
Set learning rate $\alpha$, momentum $\mu$
Repeat until convergence:
    Compute Gradient: $\nabla f(x)$
    Update velocity: $v = \mu \cdot v + \alpha \cdot \nabla f(x)$
    Update: $x = x - v$
    If convergence criterion is met:
        Break
```

4.5.3 Nesterov Accelerated Gradient Descent (NAG)

Nesterov's method is similar to classical momentum, but it looks ahead in the direction of the momentum to compute the gradient, resulting in faster convergence.

```
Initialize $x = x_0, v = 0$ (starting point and velocity)
Set learning rate $\alpha$, momentum $\mu$
Repeat until convergence:
    Compute the lookahead position: $x_{\text{hat}} = x - \mu \cdot v$
    Compute gradient at the lookahead position: $\nabla f(x_{\text{hat}})$
    Update velocity: $v = \mu \cdot v + \alpha \cdot \nabla f(x_{\text{hat}})$
    Update: $x = x - v$
    If convergence criterion is met:
        Break
```

4.5.4 Newton's Method

Newton's method updates the current point x using both the gradient $\nabla f(x)$ and the Hessian $H(x) = \nabla^2 f(x)$ (the second derivative), allowing for faster convergence especially when the function is not well-behaved.

```
Initialize $x = x_0$ (starting point)
Set learning rate $\alpha$
Repeat until convergence:
    Compute Gradient: $\nabla f(x)$
    Compute Hessian: $H(x) = \nabla^2 f(x)$
    Update: $x = x - H(x)^{-1} \cdot \nabla f(x)$
    If convergence criterion is met:
        Break
```

4.5.5 Gradient Descent with Step Size Chosen by Armijo's Rule

Armijo's rule dynamically adjusts the step size α based on the current gradient $\nabla f(x)$ and the function value $f(x)$. The step size is chosen to satisfy a decrease in the function value proportional to the gradient direction, and $c \in (0, 1)$.

```
Initialize $x = x_0$ (starting point)
Set $\alpha$ (learning rate), $\beta$ (factor for decreasing learning rate)
and $c$ (sufficient decrease condition)
Repeat until convergence:
    Compute Gradient: $\nabla f(x)$
    While: $f(x - \alpha \cdot \nabla f(x)) > f(x) - c\alpha \cdot \|\nabla f(x)\|^2$    (Armijo condition)
        Update: $\alpha = \beta \cdot \alpha$   (if while not satisfied $\alpha$ needs to be reduced)
    Update: $x = x - \alpha \cdot \nabla f(x)$
    If convergence criterion is met:
        Break
```

4.5.6 Secant Method

The Secant Method is an iterative method used to approximate the derivative of a function when the exact derivative is hard or expensive to compute. It is often used for root-finding. Given the function $f(x)$, the Secant Method approximates the derivative using two points.

```
Initialize $x_0, x_1$ (starting points)
Repeat until convergence:
    Compute: $f(x_0),\ f(x_1)$
    Compute approximate derivative: $f'(x_1) \approx \frac{f(x_1) - f(x_0)}{x_1 - x_0}$
    Compute the next point: $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$
    Update: $x_0 = x_1,\ x_1 = x_2$
    If convergence criterion is met:
        Break
```

In the Secant Method, instead of calculating the second derivative $\nabla^2 f(x)$, we approximate it using the difference of the gradients at two previous points.

4.5.7 Iteratively Reweighted Least Squares (IRLS) Newton Descent

The IRLS method is an iterative optimization approach that is often used when dealing with problems that involve a weighted least squares minimization, such as robust regression. The method iteratively updates the weights applied to the least squares error to improve convergence. This method often uses a Newton-type descent for each iteration, which updates the weights based on the residuals from the previous iteration.

```
Initialize $x = x_0$ (starting point), $y$ (target vector)
Set maximum iterations and tolerance for convergence
Repeat until convergence or maximum iterations reached:
    Compute the residuals: $r = y - f(x)$
    Compute the weights $W$ based on residuals: $W = \text{diag}(w_1, w_2, ..., w_n) = \text{diag}(|\frac{1}{r_1}|, |\frac{1}{r_2}|, ..., |\frac{1}{r_n}|)$
    Solve the weighted least squares problem using Newton's method:
        Compute the gradient $\nabla f(x) = W \cdot (f(x) - y)$ and the hessian $H(x) = W$
        Solve for $\Delta x$: $\nabla^2 f(x)\Delta x = -\nabla f(x)$
    Update: $x_{k+1} = x_k + \Delta x$                                        If conver
        Break
```

4.5.8 Quasi-Newton Methods (Including BFGS Update)

Quasi-Newton methods aim to approximate the Newton's method in a computationally efficient way, especially when the Hessian matrix is difficult to compute or invert. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) update is a widely used Quasi-Newton method that iteratively approximates the Hessian.

```
Initialize $x_0$ (starting point), $B_0 = I_n$ (initial approximation for Hessian)
Repeat until convergence:
    Compute: $\nabla f(x_k)$
    Compute: $s_k = x_{k+1} - x_k$
    Compute the difference: $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
    Update $B_k$: $B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$ (BFGS update rule)
    Solve for the descent direction: $p = -B_k^{-1} \nabla f(x_k)$
    Update: $x_{k+1} = x_k + p$
    If convergence criterion is met:
        Break
```

The BFGS method is an iterative process for updating an approximation to the Hessian matrix, and it uses this approximation to compute the descent direction and update the parameters efficiently.

5 Continuous Optimization

5.1 Unconstrained vs. Constrained Optimization

5.1.1 Unconstrained Optimization:

In unconstrained optimization, the goal is to minimize a function $f(x)$ over its domain $D = \text{dom}\, f$, without any explicit constraints on x:

$$\min_{x \in D} f(x).$$

For these problems, the feasible set is simply D, the domain of f.

5.1.2 Constrained Optimization:

If restrictions are imposed on x (e.g., $g_i(x) \leq 0$ for certain constraint functions $g_i(x)$), the problem becomes a constrained optimization problem, where solutions must satisfy these additional conditions.

6 Constrained Optimization

Now, we want to optimize a problem under certain constraints:

- **Objective:** Minimize or Maximize a function $f(x)$ subject to constraints.
- **General Form:** We will denote this form as the Primal Form or Primal Problem .
  - $\min_{x \in \mathbb{R}^{\ltimes}} f(x)$
  - subject to: $g_i(x) \leq 0, \ i = 1, \ldots, m$
    $h_j(x) = 0, \ j = 1, \ldots, p$

6.1 Definitions

- **Feasible Set:** The feasible set (or feasible region) is the set of all points that satisfy the constraints of an optimization problem. Formally, for a problem with constraints $g_i(x) \leq 0$ and $h_j(x) = 0$, the feasible set S is:
$$S = \{x \in \mathbb{R}^{\ltimes} \mid g_i(x) \leq 0, \ h_j(x) = 0, \ \forall i, j\}$$
- **Feasible Solution:** A feasible solution is any point $x \in S$ that satisfies all problem constraints.
- **Optimal solution:** If it exists, is a feasible solution that minimizes (or maximizes) the objective function within the feasible set.
- **Lagrangian Function:** The Lagrangian function is defined as:
$$\mathcal{L}(x, \lambda) = f(x) + \sum_{j=1}^{p} \lambda_j h_j(x)$$
  where λ_j are the Lagrange multipliers.
- Dual Problem: Minimize with respect to x and $\lambda_i' s$ the Lagrangian $\mathcal{L}(x, \lambda)$:
  - $\min_{x \in \mathbb{R}^{\ltimes}} \mathcal{L}(x, \lambda)$
  - subject to: $g_i(x) \leq 0, \ i = 1, \ldots, m$
    $h_j(x) = 0, \ j = 1, \ldots, p$

6.2 Optimization with Equality Constraints

6.2.1 Theorem: First-Order Optimality Conditions

Let x^* be a local minimum of $f(x)$ subject to equality constraints $h_j(x) = 0$ for $j = 1, \ldots, p$. If x^* is a regular point (the gradients $\nabla h_1(x^*), \ldots, \nabla h_p(x^*)$ are linearly independent), there exist Lagrange multipliers $\lambda_1, \lambda_2, \ldots, \lambda_p$ such that:

$$\nabla f(x^*) + \sum_{j=1}^{p} \lambda_j \nabla h_j(x^*) = 0, \ h_j(x^*) = 0, \ j = 1, \ldots, p$$

6.3 Optimization with Equality and Inequality Constraints

The Karush-Kuhn-Tucker (KKT) conditions are necessary conditions to check optimality in problems involving both equality and inequality constraints. They extend the method of Lagrange multipliers to handle inequality constraints. The primal problem which aims at minimizing the objective function $f(x)$ is turned into the minimization of the following Lagrangian function:

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i \cdot g_i(x) + \sum_{j=1}^{p} \mu_j \cdot h_j(x)$$

We call the primal feasibility the conditions that guarantee that the solution lies within the feasible region in the primal problem. Besides, we call dual feasibility the condition that forces the Lagrange multipliers associated with the inequality constraints must be non-negative.

6.3.1 Theorem : KKT Conditions

Let $f(x)$, $g_i(x)$, and $h_j(x)$ be continuously differentiable. If x^* is a local minimum and satisfies certain regularity conditions, then there exist $\lambda_i \geq 0$ and μ_j such that the KKT conditions hold.

6.3.2 Dual Problem

The dual problem is derived by minimizing the Lagrangian over x:

$$g(\lambda, \mu) = \inf_{x} \mathcal{L}(x, \lambda, \mu)$$

Then, the dual problem is:

$$\max_{\lambda \geq 0, \mu} g(\lambda, \mu)$$

6.3.3 Strong & Weak Duality

After solving both, the primal problem (x^*) and dual problem (λ, μ), we have the following 2 properties:

1. **Weak Duality:** $f(x^*) \geq g(\lambda, \mu)$
2. **Strong Duality:** $f(x^*) = g(\lambda^*, \mu^*)$

6.3.3.1 Theorem: Slater's Condition

For a convex optimization problem (**objective function and inequality contraint functions are convex while the equality constraint functions are affine**), if there exists a strictly feasible point x (one that satisfies $g_i(x) < 0,\ h_j(x) = 0$), then strong duality holds.

6.4 Example 1

Let's consider the following optimization problem we want to solve:

$$\text{Minimize: } f(x) = x^2 \quad \text{subject to: } g(x) = x - 2 \leq 0.$$

To handle the constraint, we define the Lagrangian function:

$$\mathcal{L}(x, \lambda) = f(x) + \lambda \cdot g(x),$$

Where:

- $f(x)$ is the objective function (x^2),
- $g(x)$ is the constraint $(x - 2)$,

- $\lambda \geq 0$ is the Lagrange multiplier.

Then, in our case:

$$\mathcal{L}(x, \lambda) = x^2 + \lambda(x - 2).$$

The dual function $g(\lambda)$ is obtained by minimizing the Lagrangian over x:

$$g(\lambda) = \inf_x \mathcal{L}(x, \lambda).$$

Therefore, by minimizing $\mathcal{L}(x, \lambda)$ with respect to x:

$$\frac{\partial \mathcal{L}}{\partial x} = 2x + \lambda \longleftrightarrow 2x + \lambda = 0 \longleftrightarrow x = -\frac{\lambda}{2}$$

By substituting $x = -\frac{\lambda}{2}$ into $\mathcal{L}(x, \lambda)$ to compute $g(\lambda)$:

$$\mathcal{L}\left(-\frac{\lambda}{2}, \lambda\right) \;=\; \left(-\frac{\lambda}{2}\right)^2 + \lambda\left(-\frac{\lambda}{2} - 2\right) \;=\; \frac{\lambda^2}{4} - \frac{\lambda^2}{2} - 2\lambda \;=\; -\frac{\lambda^2}{4} - 2\lambda$$

The dual function has to be maximized $g(\lambda)$ subject to $\lambda \geq 0$:

$$\max_{\lambda \geq 0} g(\lambda) = -\frac{\lambda^2}{4} - 2\lambda.$$

Thus, we need to differentiate $g(\lambda)$ with respect to λ:

$$\frac{dg}{d\lambda} = -\frac{\lambda}{2} - 2 = 0 \longleftrightarrow \lambda = -4 \quad \text{(but this violates } \lambda \geq 0\text{)}$$

Therefore, $\lambda = 0$, and:

$$g(0) = -\frac{(0)^2}{4} - 2(0) = 0.$$

The primal solution gives $x^* = 2$ and $f(x^*) = 4$, the dual solution gives $g(\lambda) = 0$, which satisfies weak duality ($4 = f(x^*) \geq g(\lambda) = 0$). If Slater's Condition holds (i.e., $g(x) < 0$ for some x) (which doesn't because $g(2) = 0$), then strong duality ensures $f(x^*) = g(\lambda^*)$.

6.5 Example 2

We want to solve the following optimization problem:

$$\text{Minimize: } f(x_1, x_2) = x_1^2 + x_2^2$$

Subject to:

- Inequality constraint: $g(x_1, x_2) = x_1 + x_2 - 1 \leq 0$,
- Equality constraint: $h(x_1, x_2) = x_1 - x_2 = 0$.

The Lagrangian for this problem is defined as:

$$\mathcal{L}(x_1, x_2, \lambda, \mu) = f(x_1, x_2) + \lambda g(x_1, x_2) + \mu h(x_1, x_2) = x_1^2 + x_2^2 + \lambda(x_1 + x_2 - 1) + \mu(x_1 - x_2)$$

Where:

- $\lambda \geq 0$ is the Lagrange multiplier for the inequality constraint,
- μ is the Lagrange multiplier for the equality constraint.

The dual function $g(\lambda, \mu)$ is obtained by minimizing the Lagrangian over x_1 and x_2:

$$g(\lambda, \mu) = \inf_{x_1, x_2} \mathcal{L}(x_1, x_2, \lambda, \mu).$$

Therefore, we need to compute the gradient of the Lagrangian with respect to x_1 and x_2 by taking partial derivatives of $\mathcal{L}$ with respect to x_1 and x_2:

- $\frac{\partial \mathcal{L}}{\partial x_1} = 2x_1 + \lambda + \mu \longleftrightarrow \frac{\partial \mathcal{L}}{\partial x_1} = 0 \longleftrightarrow 2x_1 + \lambda + \mu = 0 \longleftrightarrow x_1 = -\frac{\lambda+\mu}{2}$
- $\frac{\partial \mathcal{L}}{\partial x_2} = 2x_2 + \lambda - \mu \longleftrightarrow \frac{\partial \mathcal{L}}{\partial x_2} = 0 \longleftrightarrow 2x_2 + \lambda - \mu = 0 \quad \Rightarrow \quad x_2 = -\frac{\lambda-\mu}{2}$

By substituting $x_1 = -\frac{\lambda+\mu}{2}$ and $x_2 = -\frac{\lambda-\mu}{2}$ into $\mathcal{L}$, we would obtain the dual function that would need to be maximized:

$$\mathcal{L}\left(-\frac{\lambda+\mu}{2}, -\frac{\lambda-\mu}{2}, \lambda, \mu\right) = \left(-\frac{\lambda+\mu}{2}\right)^2 + \left(-\frac{\lambda-\mu}{2}\right)^2 + \lambda\left(-\frac{\lambda+\mu}{2} - \frac{\lambda-\mu}{2} - 1\right) + \mu\left(-\frac{\lambda+\mu}{2} + \frac{\lambda-\mu}{2}\right).$$

7 Stochastic Optimization

Stochastic optimization refers to optimization methods that introduce randomness to address uncertainty in various areas, such as data, models, or the optimization process itself. Unlike deterministic methods, stochastic optimization uses probabilistic techniques to find optimal solutions.

- **Data Sampling:** Methods like Stochastic Gradient Descent (SGD) work with random subsets of data to handle large-scale datasets efficiently.
- **Latent Variables:** Algorithms like the Expectation-Maximization (EM) algorithm iteratively estimate unobserved or latent variables, optimizing parameters alternately.

Stochastic methods, such as SGD and EM, are particularly useful when dealing with uncertainty, large datasets, and models with latent variables. They balance randomness and optimization to achieve effective solutions.

7.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an optimization algorithm used to minimize a function by iteratively updating its parameters based on the gradient of the function. Unlike traditional gradient descent, which computes the gradient using the entire dataset, SGD updates the parameters using the gradient of a randomly chosen subset (or a single data point) at each step. SGD is efficient for large-scale optimization. Convergence rates are slower than full gradient methods.

7.1.1 SGD Update Rule

Each function $f_i(x)$ represents the loss or cost associated with a specific data point or a subset of data from the original dataset in which we want to optimize the function $f(x)$.

- **Objective:** Minimize the average of functions:

$$\min_x \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

- **Gradient Descent Update:**

$$x^{(k+1)} = x^{(k)} - \alpha_k \cdot \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(x^{(k)})$$

- **Stochastic Gradient Descent Update (SGD):**

$$x^{(k+1)} = x^{(k)} - \alpha_k \cdot \nabla f_{i_k}(x^{(k)})$$

  where i_k is chosen randomly or cyclically.

7.1.2 Choosing i_k

There are 2 possible was to choose the indices i_k which determine the randomness of the SGD method:

1. **Randomized Rule:** Choose i_k uniformly at random from $\{1, \ldots, m\}$. This rule provides an unbiased estimate of the gradient.
2. **Cyclic Rule:** Choose i_k cyclically (e.g., iterate through all indices in order).

7.1.3 Step Size

The step size α_k controls the magnitude of updates. The standard practice is to use diminishing step sizes (e.g., $\alpha_k = \frac{1}{k}$) to reduce the impact of noisy gradients over time.

7.2 Convergence

SGD with diminishing step sizes converges at a sublinear rate:

$$E[f(x^{(k)})] - f^* = O\left(\frac{1}{\sqrt{k}}\right)$$

If $f(x)$ is strongly convex and has a Lipschitz gradient, convergence is faster:

$$E[f(x^{(k)})] - f^* = O\left(\frac{1}{k}\right)$$

In general, the convergence rate is slower than standard gradient descent due to gradient noise.

7.2.1 Mini-batch SGD

In **mini-batch SGD**, a random subset of indices $I_k \subset \{1, \ldots, m\}$ such that $\#I_k = b$ is chosen. The update rule for mini-batches:

$$x^{(k+1)} = x^{(k)} - \alpha_k \cdot \frac{1}{b} \sum_{i \in I_k} \nabla f_i(x^{(k)})$$

The benefits include:

- Reduces variance by a factor of $\frac{1}{b}$.
- Suitable for parallel computation.

The trade-offs include:

- Mini-batches reduce variance but are b-times more expensive than pure SGD.
- Convergence rate: $O\left(\frac{1}{\sqrt{bk} + \frac{1}{k}}\right)$.

7.3 EM Algorithm

The Expectation-Maximization (EM) Algorithm is used for maximum likelihood estimation (MLE) in probabilistic models with latent (unobserved) variables. The main purpose of the algorithm is to help solve MLE problems for latent variable models where we have observed data X and latent data Z, and the goal is to find the model parameters θ that maximize the observed data likelihood.

7.4 Goal

Maximize the likelihood: The goal is to compute:

$$\hat{\theta} = \arg\max_{\theta} \log p(X|\theta)$$

where $\log p(X|\theta)$ is the observed data log-likelihood function. The EM algorithm alternately optimizes the log-likelihood by using both the observed and latent variables:

1. E-step (Expectation): Compute the expected log-likelihood, assuming the latent variables Z are available.
2. M-step (Maximization): Maximize the expected log-likelihood with respect to the model parameters θ.

7.5 EM Steps

1. Initialize with an initial estimate of $\theta^{(0)}$.
2. Repeat until convergence:
   - **E-Step:** Compute the expectation of the complete-data log-likelihood:

     $$Q(\theta|\theta^{(k)}) = \mathbb{E}_{Z|X,\theta^{(k)}}[\log p(X, Z|\theta)]$$

   - **M-Step:** Maximize $Q(\theta|\theta^{(k)})$ to update θ:

     $$\theta^{(k+1)} = \arg\max_{\theta} Q(\theta|\theta^{(k)})$$

7.6 Convergence

The EM algorithm guarantees that the observed data log-likelihood increases with each iteration. It converges to a local maximum or stationary point, but not necessarily to the global maximum.

7.7 Advantages & Limitations

- **Advantages:**
  - Natural handling of latent variables: The EM algorithm is well-suited for problems with hidden or unobserved variables.
  - Straightforward implementation: It is easy to implement for many probabilistic models.
  - Widely used in probabilistic machine learning models.
- **Limitations:**
  - Non-convex problems: EM is generally used for non-convex problems where multiple local optima may exist.
  - Local optima: The algorithm may converge to a local optimum instead of the global one.
  - Slow convergence: Near the optimum, the convergence rate can be slow.
  - Sensitivity to initialization: The final solution can be sensitive to the initial parameter guess, leading to different results based on initialization.

8 Linear Algebra Tips

★ $\dfrac{\partial a^T x}{\partial x} = \dfrac{\partial x^T a}{\partial x} = a$

★ $\dfrac{\partial x^T A x}{\partial x} = (A + A^T)x$

★ $\dfrac{\partial \log |A|}{\partial A} = A^{-1}$

★ $x^T A x = \text{trace}(x^T A x)$

★ $\text{trace}(x^T A x) = \text{trace}(x x^T A)$

★ $\dfrac{\partial \text{ trace}(BA)}{\partial A} = B^T$

Example of last property:

Let A be a 2×2 matrix $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and B be a 2×2 matrix $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$. The product BA is:

$$BA = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} b_{11}a_{11} + b_{12}a_{21} & b_{11}a_{12} + b_{12}a_{22} \\ b_{21}a_{11} + b_{22}a_{21} & b_{21}a_{12} + b_{22}a_{22} \end{bmatrix}.$$

The trace of BA is the sum of the diagonal elements:

$$\text{trace}(BA) = (b_{11}a_{11} + b_{12}a_{21}) + (b_{21}a_{12} + b_{22}a_{22}).$$

The derivative of trace(BA) with respect to A results in:

$$\frac{\partial \text{trace}(BA)}{\partial A} = B^T = \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{bmatrix}.$$

Thus, the property holds true.