CS 412 — Introduction to Machine Learning (UIC) February 18, 2025

Lecture [9]

Instructor: Aadirupa Saha *Scribe(s): Datta Sai V V N*

Overview

In the last lecture, we covered the following main topics:

1. **Properties of Cvx funcs**
2. **Gradient descent (GD)**
3. **Convergence rates**

This lecture focuses on:

1. **Convergence of GD**
2. **Taylor Series Approximation**
3. **Newton's Method**

1 Conversion Rates & Gradient Descent

Gradient Descent (GD) is an optimization algorithm used to find a local minimum of a function. The learning rate plays a crucial role in determining how quickly the algorithm converges.

1.1 Convergence Rate of Gradient Descent

Algorithm 1.1: Gradient Descent Update Rule

Update Rule:
$x_{t+1} = x_t - \eta \nabla f(x_t)$

where:
x_t is the current point
η is the learning rate
$\nabla f(x_t)$ is the gradient at x_t

1.1.1 Assumption

- The function $f : \mathbb{R}^d \to \mathbb{R}$ is convex and **L-Lipschitz**.

Theorem 1.1:

Convergence Rate of Gradient Descent

Theorem 1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex and L-Lipschitz function. Then the gradient descent algorithm satisfies:*

$$f(\bar{x}_T) - f(x^*) \leq \frac{\|x_1 - x^*\|^2 L}{\sqrt{T}}$$

where:
*- x^* is the optimal point (minimizer of $f(x)$). - $\bar{x}_T$ is the **averaged iterate**:*

$$\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$$

This result shows that gradient descent has a convergence rate of $O(1/\sqrt{T})$ for general convex functions.

1.2 Sublinear Convergence of Gradient Descent: $O(1/\sqrt{T})$

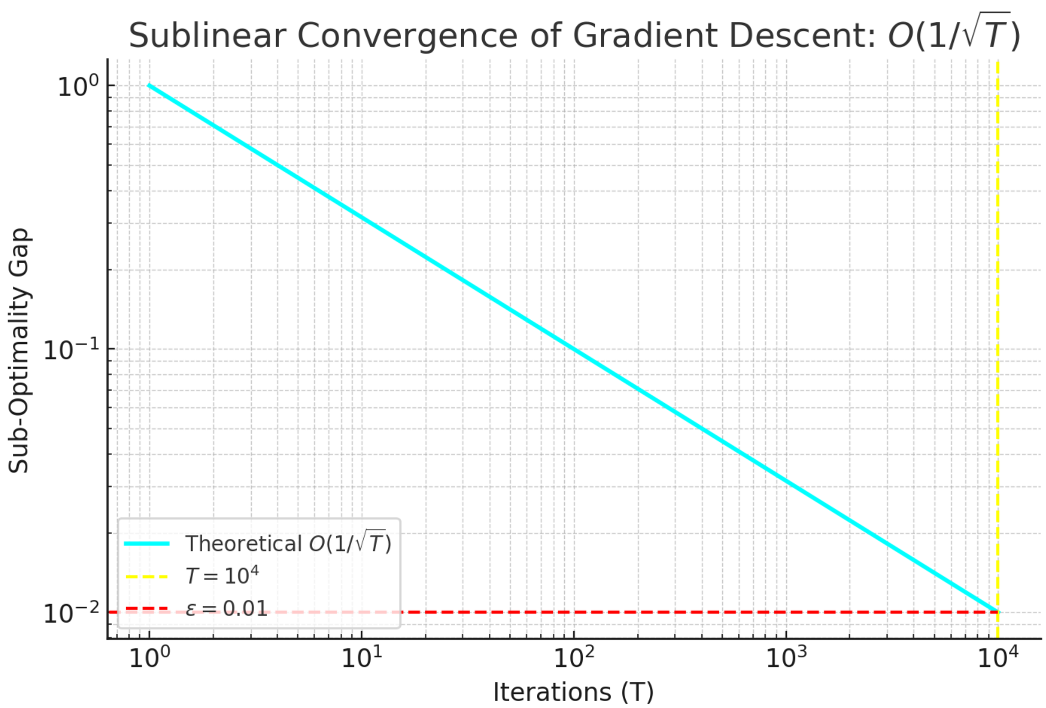

Figure 1: **Sublinear Convergence of Gradient Descent:** $O(1/\sqrt{T})$

Graph Interpretation:

- The **cyan curve** represents the theoretical convergence rate $O(1/\sqrt{T})$, showing how the sub-optimality gap shrinks as T increases.

- The **red dashed line** at $\varepsilon = 0.01$ represents the target error threshold.
- The **yellow vertical line** at $T = 10^4$ marks the required iterations for Gradient Descent to ensure that the function value is within 0.01 of the optimal.
- This graph visually confirms the theoretical result that Gradient Descent exhibits sublinear convergence.

1.3 Sub-Optimality Gap

Sub-Optimality Gap **Definition 1** The **Sub-Optimality Gap** of a point x is defined as:

$$\text{Sub-Opt Gap}(x) = f(x) - f(x^*)$$

where:

- $f(x)$ is the function value at x,
- $f(x^*)$ is the optimal function value (at the minimizer x^*).

◊

Theorem 1.2:

Sub-Optimality Gap for Averaged Iterate

Theorem 2. *For the averaged iterate $\bar{x}_T$, the **Sub-Optimality Gap** satisfies:*

$$\textit{Sub-Opt Gap}(\bar{x}_T) = \frac{\|x_1 - x^*\|^2 L}{\sqrt{T}}$$

*This result matches the **convergence rate** shown in Algorithm 1.1*

1.4 Real-World Application: Sub-Optimality Gap in Machine Learning Optimization

Why is the Sub-Optimality Gap Important?
In practical machine learning, the sub-optimality gap measures how far a current solution is from the optimal model parameters. It is widely used to evaluate optimization algorithms in deep learning and convex optimization.

Key Applications:

- **Neural Network Training:** - During training, loss minimization follows a sub-optimality gap reduction. - Convergence analysis ensures models are optimized efficiently.
- **Hyperparameter Tuning:** - Gradient-based optimization methods rely on tracking sub-optimality gap for learning rate adjustments.
- **Convex Optimization Problems:** - Used in Lasso regression, SVM training, and logistic regression. - Ensures optimal parameter selection over iterations.

1.5 Example: Finding T for a Given Sub-Optimality Gap

Exercise 1.1:

Problem Setup

- Suppose we have a **convex function** $f : \mathbb{R}^d \to \mathbb{R}$.
- It is **1-Lipschitz** (i.e., $L = 1$).
- The initial distance from the optimum is known:

$$\|x_1 - x^*\|^2 = 1$$

Objective: Find T such that the **Sub-Optimality Gap** satisfies:

$$f(\bar{x}_T) - f(x^*) = \varepsilon$$

Step 1: Using the Convergence Rate Formula
We know that for convex functions, Gradient Descent satisfies:

$$f(\bar{x}_T) - f(x^*) \leq \frac{\|x_1 - x^*\|^2 L}{\sqrt{T}}$$

Substituting known values:

$$\frac{1 \cdot 1}{\sqrt{T}} = \varepsilon$$

Step 2: Solving for T
For a given $\varepsilon = 0.01$, we set up the equation:

$$\frac{1}{\sqrt{T}} = 0.01$$

Squaring both sides:

$$T = \frac{1}{(0.01)^2} = 10^4$$

Step 3: Interpretation

- If Gradient Descent is run for 10^4 steps, then we guarantee:

$$f(x_{10^4}) - f(x^*) \leq 0.01$$

- This means that after **10,000 iterations**, the function value is at most **0.01** away from the optimal function value.

1.6 Visual Representation for a Given Sub-Optimality Gap:

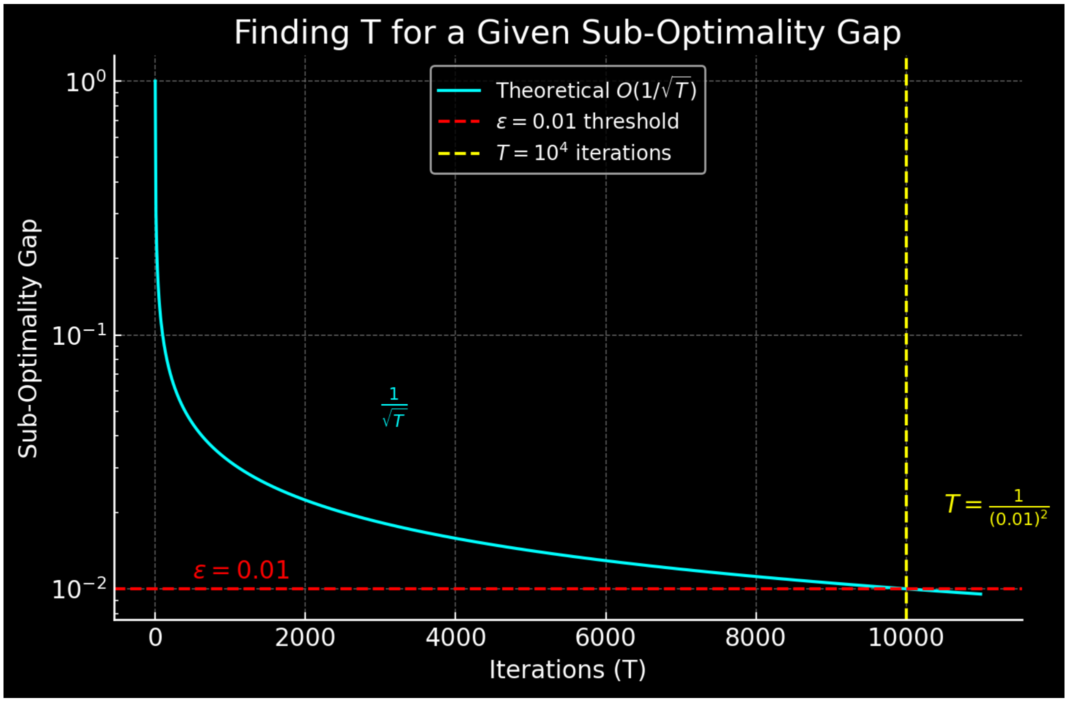

Figure 2: **Finding T for a Given Sub-Optimality Gap**

Graph Interpretation:

- The red dashed line marks the error threshold $\varepsilon = 0.01$, indicating the desired level of accuracy.
- The yellow vertical line at $T = 10^4$ shows the number of iterations required to ensure that the function value is at most 0.01 away from the optimal function value.
- This visualization confirms that Gradient Descent requires at least 10^4 iterations to meet the specified accuracy.
- Thus, the graph provides an intuitive visual confirmation of the theoretical convergence rate and the required iterations for a given accuracy.

1.7 Comparison of Convergence Rates

Function Type	**Convergence Rate**	**Steps for $\varepsilon = 0.01$**
Convex Only	$O(1/\sqrt{T})$	10^4
Convex + Smooth	$O(1/T)$	201
Strongly Convex	$O(1/T)$	201
Strongly Convex + Smooth	$O(e^{-T/\kappa})$	10

2 Taylor Series Approximation

2.1 Definition

Theorem 2.1:

Taylor Series Expansion For a function $f : \mathbb{R} \to \mathbb{R}$, the Taylor series expansion around a point x is:

$$f(x+\delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \frac{\delta^3}{3!} f'''(x) + \dots \tag{1}$$

where:

- $f'(x) = \frac{df}{dx}$ (first derivative)
- $f''(x) = \frac{d^2 f}{dx^2}$ (second derivative)
- $f'''(x)$, etc., represent higher-order derivatives.

2.2 Example: $f(x) = ax^2$

Let's approximate $f(x) = ax^2$ using Taylor series.

Exercise 2.1:

First, compute derivatives:

$$f'(x) = 2ax$$
$$f''(x) = 2a$$
$$f'''(x) = 0 \quad \text{(all higher-order derivatives are zero)}$$

Applying Taylor expansion:

$$f(x+\delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + 0 \tag{2}$$

Substituting values:

$$f(x+\delta) = ax^2 + \delta(2ax) + \frac{\delta^2}{2}(2a)$$

$$= ax^2 + 2ax\delta + a\delta^2$$

$$= a(x+\delta)^2$$

Thus, **LHS = RHS**, verifying that Taylor series provides an **exact approximation** for quadratic functions.

3 Intuition: Why Gradient Descent (GD) Works

Gradient Descent (GD) is based on the **first-order approximation** of a function using **Taylor series**.

3.1 Taylor Series Approximation

Taylor series allows us to approximate a function $f(x)$ around a point by expanding it in terms of its derivatives. This helps us analyze how function values change with small steps in x, which is fundamental to the working of GD.

Theorem 3.1:

Taylor Series Expansion For a function $f : \mathbb{R} \to \mathbb{R}$, expanding $f(x + \delta)$ gives:

$$f(x+\delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \frac{\delta^3}{3!} f'''(x) + \dots \tag{3}$$

where:

- $f'(x)$ is the first derivative,
- $f''(x)$ is the second derivative,
- Higher-order terms contain $\delta^2, \delta^3, \ldots$, which become **very small** for small δ.

3.2 Ignoring Higher-Order Terms

In optimization, small step sizes δ make higher-order terms insignificant. This allows us to approximate the function using only the first derivative, leading to a simpler and computationally efficient model for optimization.

Exercise 3.1: I

f δ is small (e.g., $\delta = 0.001$), then:

$$\delta^2 = (0.001)^2 = 10^{-6}, \quad \delta^3 = 10^{-9}, \dots$$

These higher-order terms become negligible, leaving us with:

$$f(x+\delta) \approx f(x) + \delta f'(x) \tag{4}$$

This is a **first-order approximation**, meaning that for small steps, we can approximate function behavior using just the **gradient** $f'(x)$.

3.3 How Gradient Descent Uses This Approximation

Exercise 3.2:

Gradient Descent chooses δ such that:

$$\delta = -\eta f'(x) \tag{5}$$

where η is the step size (learning rate).
Plugging into the approximation:

$$f(x - \eta f'(x)) \approx f(x) - \eta f'(x)^2 \tag{6}$$

Ensuring Descent:
Since $\eta f'(x)^2 \geq 0$, we guarantee:

$$f(x_{t+1}) < f(x_t) \tag{7}$$

Thus, the function **value always decreases**, ensuring descent.

3.4 Visualization of Gradient Descent Using Taylor Approximation

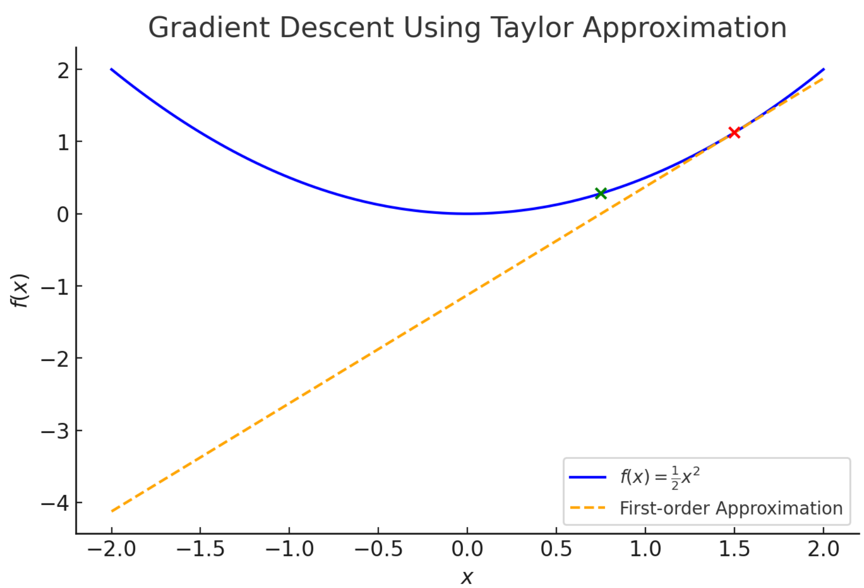

Figure 3: **Visualization of Gradient Descent Using Taylor Approximation**

The blue curve represents the function $f(x) = \frac{1}{2}x^2$. The red point is the current position x_t, and the green point is the updated position after applying gradient descent. The orange dashed line represents the first-order Taylor approximation (tangent line), demonstrating how GD follows the gradient to minimize the function.

4 Newton's Method: A Second-Order Optimization Approach

Newton's Method is an optimization technique that **uses second-order information (Hessian matrix or second derivative) to accelerate convergence**.

4.1 Second-Order Approximation using Taylor Series

Newton's Method relies on a **second-order Taylor expansion** to approximate a function more accurately than first-order methods like Gradient Descent. By incorporating the **second derivative** $f''(x)$, this approach provides a better understanding of the function's curvature, helping optimize step sizes. The higher-order terms are ignored as δ is assumed to be small, making the approximation computationally efficient.

Theorem 4.1:

Second-Order Taylor Approximation We approximate $f(x + \delta)$ using **Taylor expansion up to the second order**:

$$f(x + \delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \dots \tag{8}$$

Higher-order terms (δ^3, etc.) are **ignored** since δ is assumed to be small.

4.2 Finding the Optimal Step δ

To ensure maximum decrease in the function $f(x)$, Newton's Method derives an **optimal step size** using second-order information. By setting the derivative of the Taylor approximation to zero, we solve for δ as:

Exercise 4.1: T

o **maximize the decrease in** $f(x + \delta)$, we solve:

$$\min_{\delta} f(x + \delta) = \min_{\delta} \left[f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) \right] \tag{9}$$

Taking the derivative w.r.t. δ **and setting it to zero:**

$$f'(x) + \frac{\delta f''(x)}{2} = 0 \tag{10}$$

Solving for δ:

$$\delta = -\frac{f'(x)}{f''(x)} \tag{11}$$

This is **Newton's optimal step size**.

4.3 Newton's Algorithm

Algorithm 4.1: Newton's Method Update Rule

Using the **Newton step** δ in an iterative update:

$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)} \tag{12}$$

For **multivariate cases** ($d > 1$), the update rule is:

$$x_{t+1} = x_t - [\nabla^2 f(x_t)]^{-1} \nabla f(x_t) \tag{13}$$

where:

- $\nabla f(x_t)$ is the **gradient**.
- $\nabla^2 f(x_t)$ is the **Hessian matrix** (second-order derivatives).

4.4 Advantages of Newton's Method

- **Faster Convergence**: Quadratic convergence near optimal solutions.
- **Better Step Size Selection**: Uses second-derivative information instead of a manually chosen learning rate.
- **More Precise**: Effective for convex functions with well-conditioned Hessians.

4.5 Limitations

- **Computationally Expensive**: Requires computing the Hessian and its inverse.
- **Not Always Feasible**: Hessian inversion is difficult for high-dimensional problems.

4.6 Comparison with Gradient Descent

Method	Update Rule	Convergence Rate
Gradient Descent	$x_{t+1} = x_t - \eta \nabla f(x_t)$	$O(1/T)$ (for smooth & convex)
Newton's Method	$x_{t+1} = x_t - [\nabla^2 f(x_t)]^{-1} \nabla f(x_t)$	$O(\log T)$ (quadratic convergence)

Table 1: Comparison of Gradient Descent and Newton's Method

5 Implemented a Project Using a Dataset for the Above Lecture

To complement the theoretical concepts discussed in the lecture, I have implemented a project that applies Gradient Descent and Newton's Method to a real-world dataset. This project provides a visual and practical understanding of these optimization techniques.

5.1 Project Overview

The project involves:

- Implementing Gradient Descent and analyzing its convergence rate.
- Implementing Newton's Method and observing its faster convergence.
- Comparing both methods to highlight their strengths and trade-offs.

5.2 Accessing the Project

The project has been implemented in a Jupyter Notebook. You can download and explore the implementation using the link below:

Download Gradient Descent & Newton's Method Project

By referring to this notebook, you can visualize the optimization process, understand the theoretical concepts in action, and see how these methods perform on real-world data.

Next Lecture

The next lecture will cover the following topics:
(i) GD convergence analysis,
(ii) SGD + Convergence guarantees,
(iii) Batched SGD,
(iv) Variants of GD.

References:

1. Lecture notes by Prof. Aadirupa Saha from course CS 412 Intro to ML **Lec.9.pdf**