Conjugate Gradient in Large-Scale Optimization

Ronald Nap, Cristian Espinosa

May 1, 2024

Abstract

This report explores the application of the Conjugate Gradient (CG) method in large-scale optimization problems. With the increasing scale of data in fields like machine learning, traditional optimization methods often struggle to maintain efficiency. The CG method, known for its efficiency in handling large, sparse systems, is evaluated for its performance and potential improvements through preconditioning.

1 Introduction

The need for efficient optimization algorithms grows as the size and complexity of datasets increase, particularly in areas such as machine learning and data analytics where high-dimensional spaces are common. The Conjugate Gradient (CG) method offers a promising solution for solving large-scale problems efficiently. This report examines the CG method both as a standalone approach and as an enhancement to existing optimization methods. By incorporating CG, we aim to improve the performance of traditional techniques, demonstrating its suitability and enhanced efficiency in handling high-dimensional optimization challenges.

2 Problem Description

The objective of this study is to evaluate the performance of the Conjugate Gradient (CG) method against other optimization algorithms on quadratic and non-convex functions. The CG method is particularly suitable for large-scale problems where matrices are large and sparse. This study focuses on two types of functions:

- **Quadratic Function:** Defined by $f(x) = \frac{1}{2}x^T Ax - b^T x + c$, where $A \in \mathbb{R}^{5000\times5000}$ is a symmetric positive-definite matrix. The gradient of this function is $\nabla f(x) = Ax - b$, and the Hessian is simply the matrix A, which is constant.
- **Generalized Rosenbrock Function:** Defined by $g(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1-x_i)^2]$. This non-convex function is commonly used as a performance test problem for optimization algorithms. For simplicity and illustration, the gradient and Hessian calculations are presented for $n = 2$:

$$g(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$\nabla g(x_1, x_2) = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}$$

$$\text{Hessian: } H = \begin{bmatrix} -400(x_2 - 3x_1^2) + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$$

Usually the Hessian is tridiagonal for larger n but is simplified to a 2x2 matrix for $n = 2$.

Both functions present different challenges: the quadratic function tests the ability of the CG method to efficiently solve large systems of linear equations, while the Rosenbrock function tests the method's performance on a more complex, non-linear landscape.

3 Methodology

The Conjugate Gradient (CG) method is an algorithm for the numerical solution of particular systems of linear equations, specifically those where the matrix A is symmetric and positive-definite. It is often used in practical applications to solve sparse systems that arise from numerical methods.

3.1 Background

The CG method can be viewed both as an iterative method for solving linear systems and as a method to solve optimization problems of the form $f(x) = \frac{1}{2}x^T Ax - b^T x$.

3.2 Derivation

The Conjugate Gradient (CG) method efficiently finds solutions to systems where the matrix A is symmetric and positive-definite, by leveraging the mathematical properties of conjugacy and orthogonality.

- **Initialization:** The method initiates with an estimation x_0 and calculates the residual $r_0 = b - Ax_0$, which represents the negative gradient of the quadratic function at x_0 and indicates the steepest descent direction. This residual is used as the initial search direction, $p_0 = r_0$.
- **Iterative Process:**
  1. **Step Size Calculation:** For each iteration k, the CG method aims to minimize the quadratic function along the search direction p_k. The step size α_k is crucial and is computed as:
  $$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$$
  This formula determines how far along p_k the next point x_{k+1} should be from x_k. It is derived by ensuring that the new residual r_{k+1} is orthogonal to the previous residual r_k, minimizing the function $f(x_k + \alpha_k p_k)$ by taking its derivative with respect to α, setting it to zero, and solving for α_k.
  2. **Update Equations:**
     - Update the estimate of the solution: $x_{k+1} = x_k + \alpha_k p_k$.
     - Update the residual: $r_{k+1} = r_k - \alpha_k A p_k$.
     - Update the direction: The new direction p_{k+1} is adjusted to be conjugate to the previous directions under matrix A, ensuring efficient navigation through the problem space. This is done using the formula:
     $$p_{k+1} = r_{k+1} + \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} p_k$$
  This update ensures that each search direction is A-conjugate to the previous one, enhancing the method's efficiency by avoiding redundant directions.

3.3 Algorithm

The CG algorithm proceeds as follows:

1. Initialize x_0, compute $r_0 = b - Ax_0$, set $p_0 = r_0$.
2. For $k = 0, 1, 2, \ldots$ until convergence:
   - Compute $\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$
   - Update $x_{k+1} = x_k + \alpha_k p_k$
   - Compute $r_{k+1} = r_k - \alpha_k A p_k$
   - Update $p_{k+1} = r_{k+1} + \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} p_k$

4 Results

Function	Method	Iterations	Time (s)	Rel. Error
Quadratic	Steepest Descent	100	18.8564	1.890×10^{-6}
	Newton's Method	3	1.4591	$\mathbf{3.603 \times 10^{-15}}$
	Newton + CG	4	1.9177	5.818×10^{-7}
	BFGS	100	179.2829	1.257×10^{-6}
	Trust Region	100	174.7288	7.535×10^{-7}
	Trust Region + CG	100	1.1549	1.460×10^{-7}
	Conjugate Gradient	3	**0.0142**	5.636×10^{-7}
	Preconditioned CG	**2**	1.3991	3.178×10^{-9}
Rosenbrock	Steepest Descent	100	0.0346	0.541
	Newton's Method	14	0.0047	$\mathbf{4.721 \times 10^{-15}}$
	Newton's Method + CG	**2**	**0.0017**	1.652×10^{-14}
	BFGS	23	0.0027	7.609×10^{-13}
	Trust Region	100	1.8797	0.849
	Trust Region + CG	100	0.0096	1.63×10^{-4}

The results presented highlight the efficiency of integrating the Conjugate Gradient (CG) method into classical optimization frameworks like Newton's and Trust Region methods. This integration primarily aims to enhance computational speed in applications where high precision may be secondary.

4.1 Enhancing Newton's Method with CG

Newton's Method, traditionally known for its quadratic convergence rate, can sometimes be computationally demanding due to the necessity of inverting the Hessian matrix at each iteration. By incorporating CG into Newton's framework (Newton + CG), we leverage CG to solve the linear system approximately. This approach significantly reduces the computational burden per iteration.

- Newton's Method with CG achieved a commendable balance between speed and accuracy, completing in only 4 iterations with a relative error of 5.818×10^{-7}.
- This method was faster than the standard Newton's Method, which, while slightly more accurate, required more computational time.

4.2 Trust Region with CG

The Trust Region method, known for its robustness in handling non-linear optimization problems, typically involves solving a subproblem at each iteration which can be computationally intensive. Integrating CG into the Trust Region method offers a way to solve these subproblems more efficiently. The focus is on quickly finding a satisfactory solution rather than the most accurate one, which can be particularly beneficial in large-scale applications where computational resources are a limiting factor.

- Trust Region enhanced with CG (Trust Region + CG) required fewer iterations and significantly less time compared to the traditional Trust Region approach.
- With a relative error of 1.63×10^{-4}, Trust Region + CG offers a viable trade-off between computational speed and solution accuracy.

4.3 Implications

The integration of CG into Newton's and Trust Region methods underscores a practical approach in optimization practices—prioritizing speed and resource efficiency over ultimate precision. Such strategies are crucial in real-world scenarios where decisions must be made rapidly, or where computational resources are constrained.

In conclusion, the modified approaches using CG not only retain the core advantages of the original methods but also introduce significant efficiency improvements. These adaptations make the methods particularly suitable for large-scale problems or in applications where faster convergence is more critical than achieving the lowest possible error.

5 Conclusion

The Conjugate Gradient method proves to be a robust and efficient tool for large-scale optimization problems, particularly when enhanced by preconditioning. Future work will focus on improving preconditioning techniques and exploring their effects on convergence rates.