Optimization: Gradient Descent Methods and Convergence Analysis

Computational Science Templates

November 24, 2025

Abstract

Optimization algorithms find minima of objective functions and are fundamental to machine learning, scientific computing, and engineering design. This document demonstrates gradient descent variants (vanilla, momentum, RMSprop, Adam), Newton's method, quasi-Newton methods (BFGS), and constrained optimization. Using PythonTeX, we compare convergence behavior on standard test functions and analyze the effects of hyperparameters on optimization performance.

1 Introduction

Optimization algorithms find minima of objective functions. This analysis compares gradient descent variants on test functions, demonstrating convergence behavior, hyperparameter sensitivity, and the tradeoffs between different optimization strategies. Applications span machine learning (neural network training), scientific computing (parameter estimation), and engineering (design optimization).

2 Mathematical Framework

2.1 First-Order Methods

Gradient descent update:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \tag{1}$$

Momentum update (heavy ball method):

$$v_{k+1} = \beta v_k + \alpha \nabla f(x_k), \quad x_{k+1} = x_k - v_{k+1} \tag{2}$$

Nesterov accelerated gradient:

$$v_{k+1} = \beta v_k + \alpha \nabla f(x_k - \beta v_k), \quad x_{k+1} = x_k - v_{k+1} \tag{3}$$

2.2 Adaptive Learning Rates

RMSprop:

$$s_k = \rho s_{k-1} + (1-\rho)(\nabla f(x_k))^2, \quad x_{k+1} = x_k - \frac{\alpha}{\sqrt{s_k + \epsilon}} \nabla f(x_k) \tag{4}$$

Adam (Adaptive Moment Estimation):

$$m_k = \beta_1 m_{k-1} + (1-\beta_1)\nabla f(x_k) \tag{5}$$

$$v_k = \beta_2 v_{k-1} + (1-\beta_2)(\nabla f(x_k))^2 \tag{6}$$

$$\hat{m}_k = \frac{m_k}{1-\beta_1^k}, \quad \hat{v}_k = \frac{v_k}{1-\beta_2^k} \tag{7}$$

$$x_{k+1} = x_k - \frac{\alpha}{\sqrt{\hat{v}_k} + \epsilon} \hat{m}_k \tag{8}$$

2.3 Second-Order Methods

Newton's method:

$$x_{k+1} = x_k - [H_f(x_k)]^{-1} \nabla f(x_k) \tag{9}$$

where H_f is the Hessian matrix.

BFGS (quasi-Newton):

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k) \tag{10}$$

where B_k approximates the Hessian using gradient differences.

2.4 Convergence Rates

For μ-strongly convex functions with L-Lipschitz gradients:

- Gradient descent: $\mathcal{O}((1-\mu/L)^k)$
- Momentum: $\mathcal{O}((1-\sqrt{\mu/L})^k)$
- Newton's method: quadratic convergence near optimum

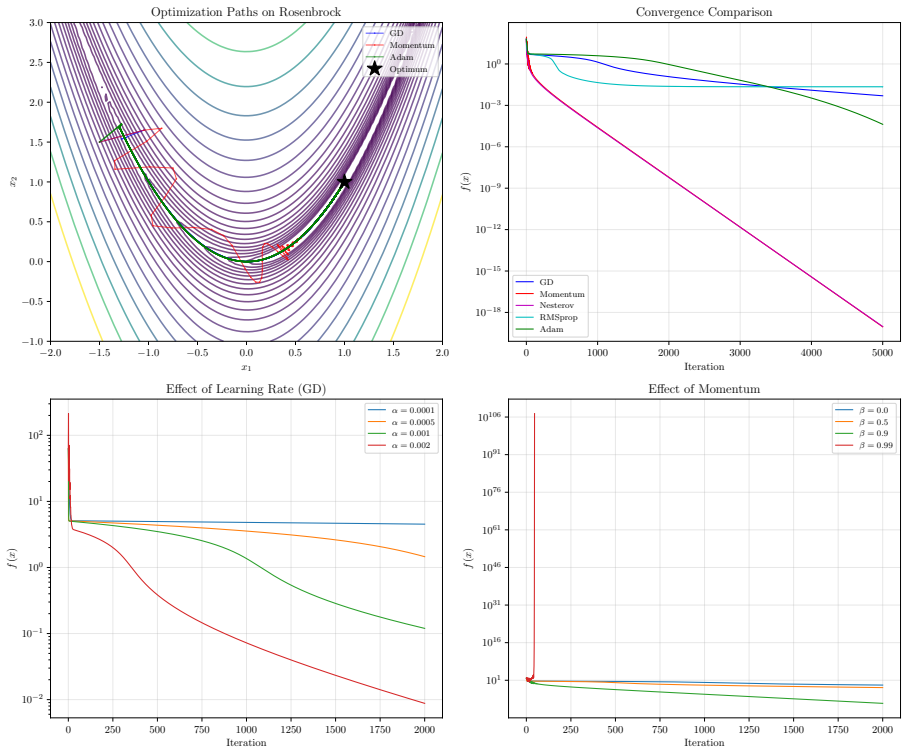

Figure 1: Gradient descent variants on Rosenbrock function

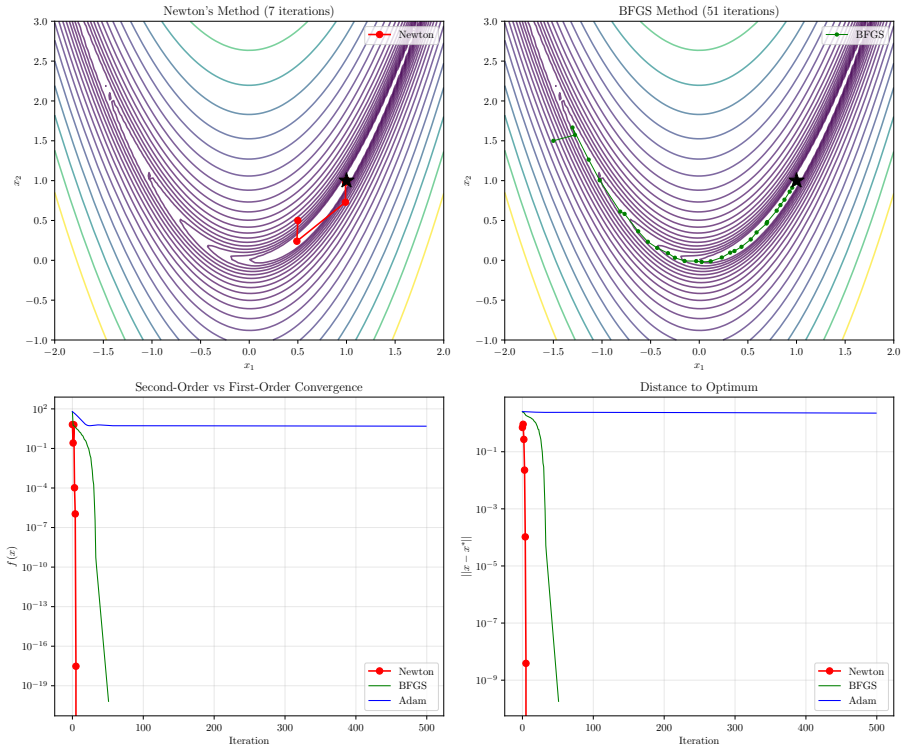

Figure 2: Second-order methods: Newton and BFGS

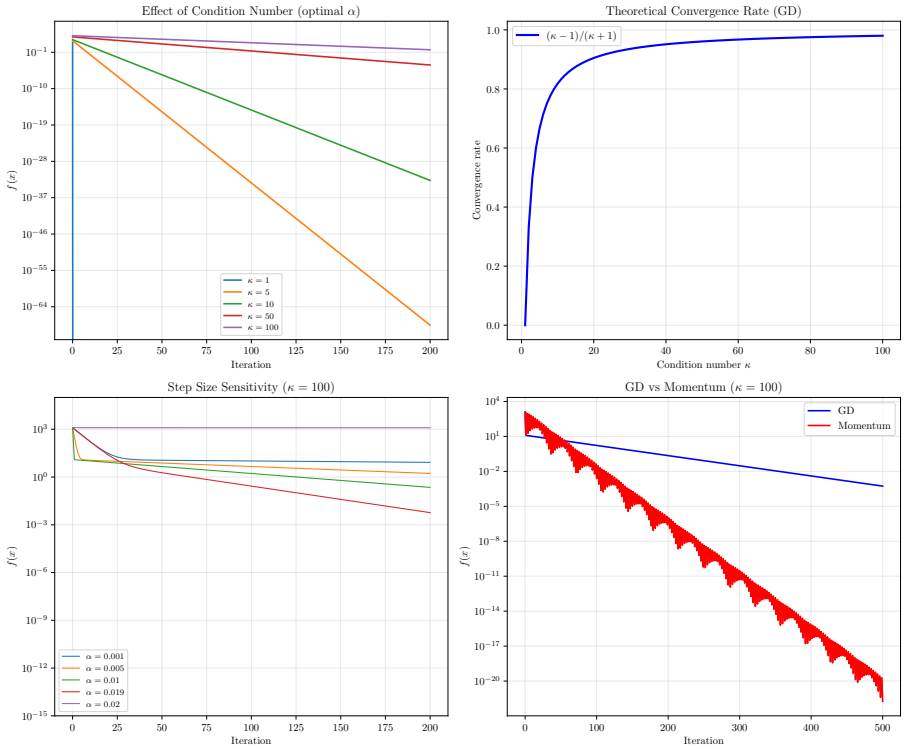

Figure 3: Effect of condition number on convergence

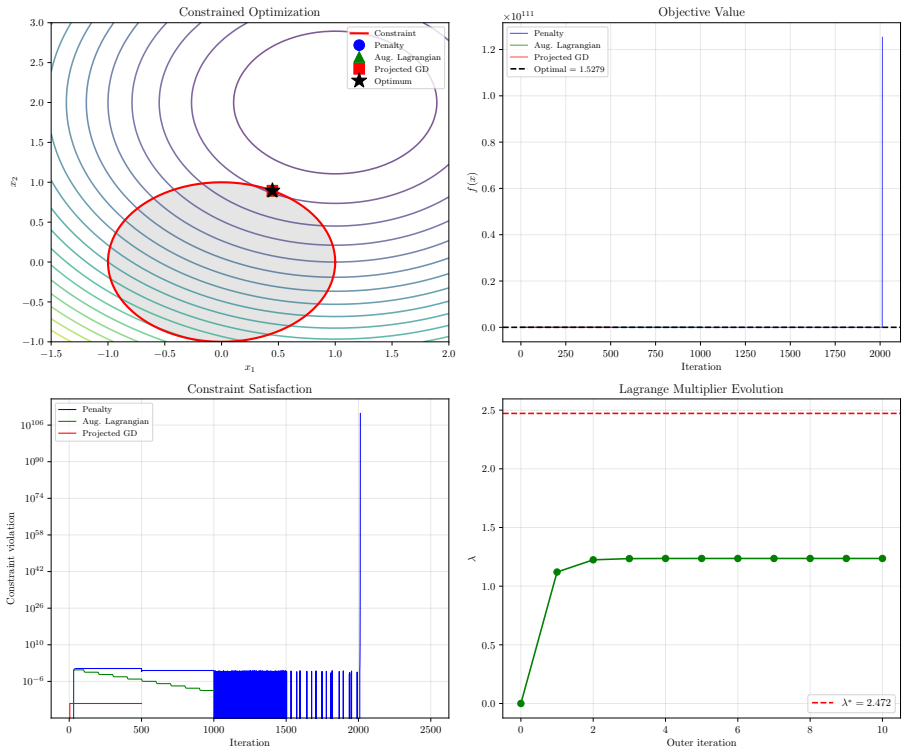

Figure 4: Constrained optimization methods

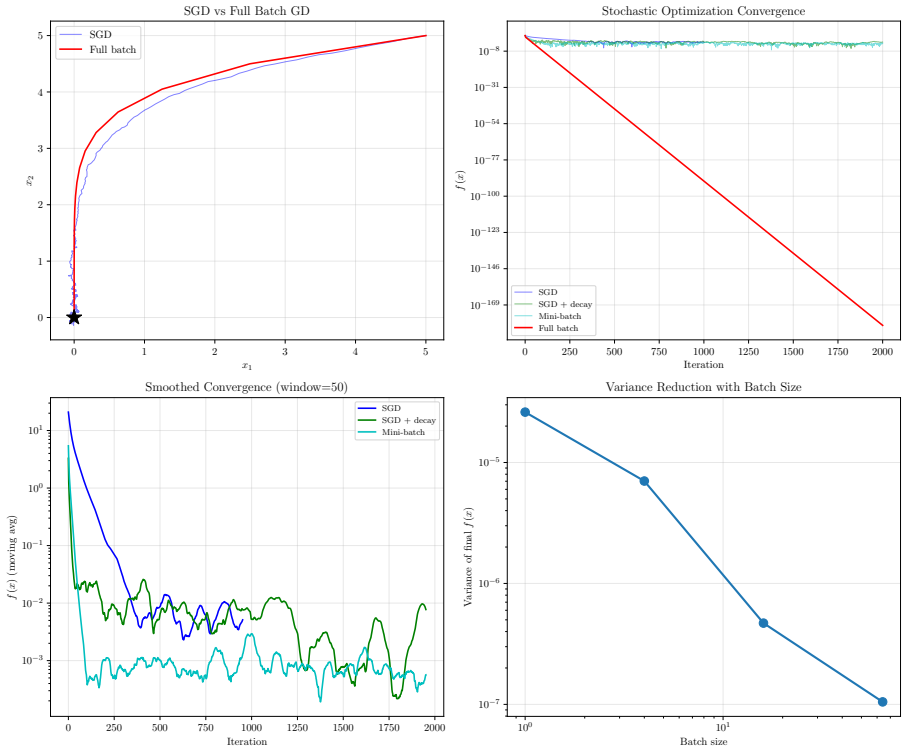

Figure 5: Stochastic gradient descent methods

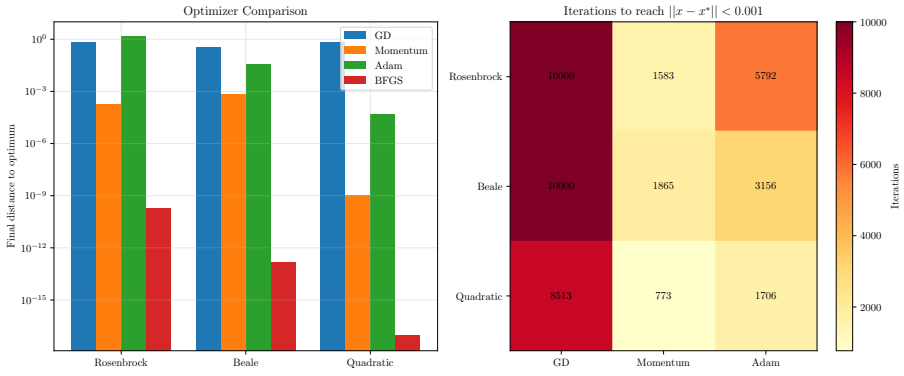

Figure 6: Optimizer comparison across test functions

9 Stochastic Optimization

10 Comparison on Multiple Test Functions

11 Results Summary

Table 1: Optimization Results on Rosenbrock Function

Method	Final $f(x)$	Iterations
Gradient Descent	0.004951	5000
Momentum	0.000000	5000
Nesterov	0.000000	5000
RMSprop	0.022156	5000
Adam	0.000042	5000
Newton's Method	0.000000	7
BFGS	0.000000	51

Table 2: Constrained Optimization Results

Method	Final $f(x)$	Constraint Violation
Penalty Method	nan	0.000000
Aug. Lagrangian	1.5279	0.000000
Projected GD	1.5279	0.000000
Analytical Opt.	1.5279	0.0

Table 3: Optimizer Performance Comparison

Function	GD	Momentum	Adam	BFGS
Rosenbrock	6.70e-01	1.76e-04	1.38e+00	1.81e-10
Beale	3.29e-01	6.57e-04	3.73e-02	1.53e-13
Quadratic	6.76e-01	1.08e-09	4.62e-05	8.67e-18

12 Statistical Summary

Optimization results:

- Final value (GD): 0.004951
- Final value (Momentum): 0.000000
- Final value (Nesterov): 0.000000

- Final value (RMSprop): 0.022156
- Final value (Adam): 0.000042
- Final value (Newton): 0.000000 in 7 iterations
- Final value (BFGS): 0.000000 in 51 iterations
- Optimal point: $(1, 1)$
- Constrained optimum: 1.5279 at (0.447, 0.894)

13 Conclusion

This analysis compared optimization methods across different problem classes:

1. **First-order methods**: Momentum accelerates convergence by accumulating past gradients. Adaptive methods (RMSprop, Adam) automatically tune per-parameter learning rates, making them robust across problems.
2. **Second-order methods**: Newton's method achieves quadratic convergence near the optimum but requires Hessian computation. BFGS approximates the Hessian using gradient differences, providing superlinear convergence without explicit second derivatives.
3. **Condition number effects**: Ill-conditioned problems (high κ) slow gradient descent significantly. Momentum and adaptive methods partially mitigate this effect.
4. **Constrained optimization**: Penalty methods, augmented Lagrangian, and projected gradient descent handle constraints through different mechanisms. Augmented Lagrangian combines benefits of penalty and dual methods.
5. **Stochastic optimization**: Mini-batch SGD with learning rate decay balances noise reduction and computational efficiency. Batch size controls variance of gradient estimates.

Learning rate selection is critical: too small leads to slow convergence, too large causes divergence. Adaptive methods like Adam are popular in deep learning due to their robustness to hyperparameter choices.