OptLearn: A Numerical Optimization Framework

Kevin Mota da Costa

January 15, 2026

Abstract

This document presents the mathematical formulation and algorithmic structure of *OptLearn*, a numerical optimization framework designed to solve unconstrained minimization problems using first-order methods. The framework implements classical gradient-based optimization algorithms and supports arbitrary objective functions, including non-linear and non-convex problems.

The emphasis of this document is on the mathematical foundations of the optimization algorithms, the numerical gradient approximation employed, and the convergence behavior observed in practical examples, including analytical benchmark functions and neural network parameter optimization.

1 Introduction

Many scientific and engineering problems can be formulated as the minimization of a scalar objective function:

$$\min_{\theta \in \mathbb{R}^n} f(\theta), \tag{1}$$

where θ denotes a vector of free parameters and $f(\theta)$ is a real-valued cost function.

OptLearn provides a unified computational framework for addressing such problems using gradient-based iterative methods. The framework is agnostic to the functional form of $f(\theta)$ and relies only on function evaluations, enabling its application to a wide range of optimization tasks.

2 Numerical Gradient Approximation

In *OptLearn*, gradients are computed numerically using central finite differences. For a scalar function $f(\theta)$, the partial derivative with respect to parameter θ_i is approximated as:

$$\frac{\partial f}{\partial \theta_i} \approx \frac{f(\theta_i + \varepsilon) - f(\theta_i - \varepsilon)}{2\varepsilon}, \tag{2}$$

where ε is a small perturbation.

This approximation provides second-order accuracy in ε and allows the optimization algorithms to operate without explicit analytical gradients.

3 Generic Optimization Procedure

All optimization algorithms in *OptLearn* follow the same iterative structure:

1. Initialize the parameter vector $\theta^{(0)}$,
2. Compute the numerical gradient $\nabla f(\theta^{(t)})$,
3. Update parameters according to an optimizer-specific rule,
4. Record diagnostic quantities such as function value and gradient norm,
5. Terminate when a convergence criterion is satisfied.

Convergence is detected when the Euclidean norm of the gradient satisfies:

$$\|\nabla f(\theta^{(t)})\| < \tau, \tag{3}$$

where τ is a user-defined tolerance.

4 Gradient Descent

The simplest optimizer implemented is gradient descent, with update rule:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla f(\theta^{(t)}), \tag{4}$$

where η is the learning rate.

Gradient descent provides a baseline method whose convergence properties are well understood for convex and smooth objective functions.

5 Momentum-Based Optimization

Momentum-based optimization augments gradient descent by introducing a velocity variable $v^{(t)}$:

$$v^{(t)} = \beta v^{(t-1)} + (1-\beta)\nabla f(\theta^{(t)}), \tag{5}$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta v^{(t)}. \tag{6}$$

This formulation reduces oscillatory behavior in directions of high curvature and accelerates convergence along consistent descent directions.

6 RMSProp

RMSProp adapts the learning rate independently for each parameter by maintaining an exponentially decaying average of squared gradients:

$$s^{(t)} = \beta s^{(t-1)} + (1-\beta)(\nabla f(\theta^{(t)}))^2, \tag{7}$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{\nabla f(\theta^{(t)})}{\sqrt{s^{(t)}} + \epsilon}. \tag{8}$$

This normalization mitigates issues caused by uneven gradient magnitudes across dimensions.

7 Adam Optimizer

Adam combines momentum and RMSProp by estimating both first and second moments of the gradient:

$$m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1)\nabla f(\theta^{(t)}), \tag{9}$$

$$v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2)(\nabla f(\theta^{(t)}))^2. \tag{10}$$

Bias-corrected estimates are given by:

$$\hat{m}^{(t)} = \frac{m^{(t)}}{1-\beta_1^t}, \tag{11}$$

$$\hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^t}. \tag{12}$$

The update rule is:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{\hat{m}^{(t)}}{\sqrt{\hat{v}^{(t)}} + \epsilon}. \tag{13}$$

Adam provides robust performance across a wide range of optimization problems.

8 Optimization of Neural Network Parameters

As a representative non-linear optimization problem, *OptLearn* is applied to the minimization of a neural network loss function.

Let $\hat{y}(x;\theta)$ denote the network output with parameters θ. Given observations $(x_i, y_i, \delta y_i)$, the objective function is:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left(\frac{y_i - \hat{y}(x_i;\theta)}{\delta y_i} \right)^2 + \lambda \|\theta\|_2^2, \tag{14}$$

where λ controls L_2 regularization.

Although the example involves a neural network, the optimization procedure remains identical to that used for analytical test functions.

9 Convergence Diagnostics

During optimization, the framework records:

- Objective function value $f(\theta^{(t)})$,
- Gradient norm $\|\nabla f(\theta^{(t)})\|$,

- Parameter values at each iteration.

These quantities enable detailed analysis of convergence behavior and optimizer performance.

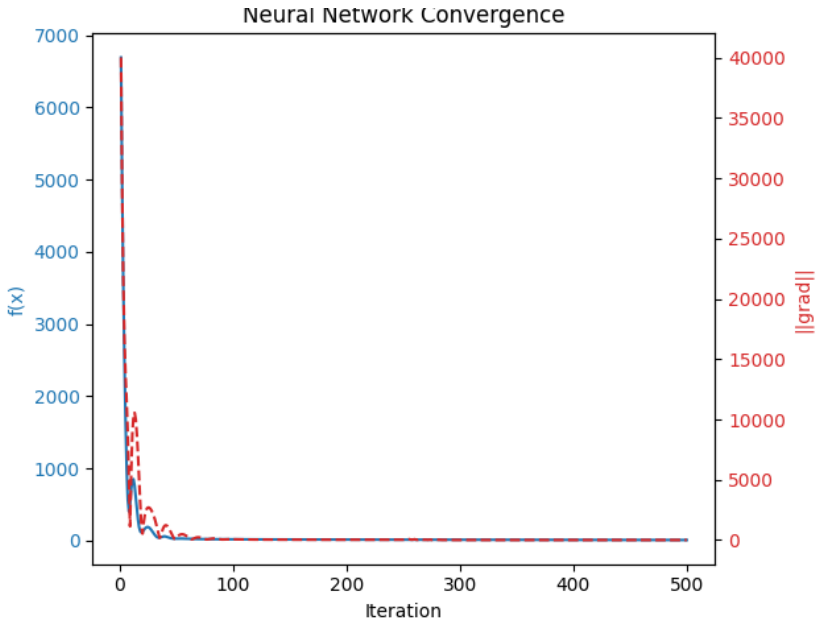

Figure 1: Convergence of the objective function and gradient norm during optimization.

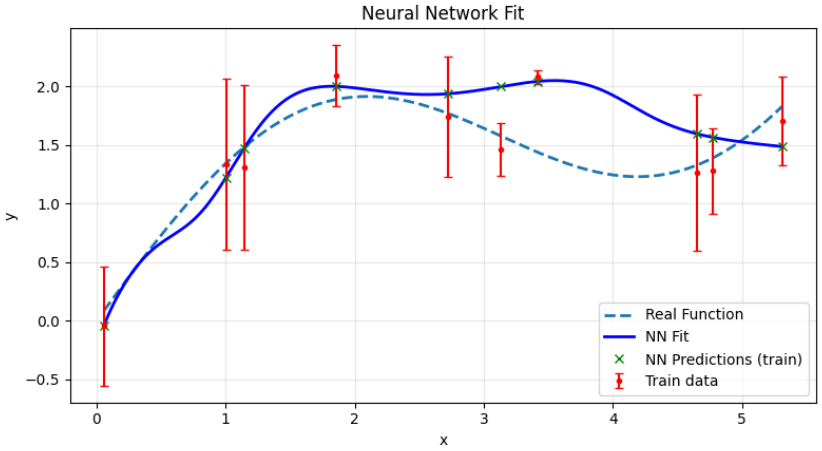

Figure 2: Neural network solution obtained via numerical optimization.

10 Conclusion

OptLearn provides a general-purpose framework for numerical optimization based on first-order methods. By supporting multiple optimization algorithms within a unified structure, the framework enables systematic comparison of convergence properties and behavior across different objective functions.

The design emphasizes mathematical transparency, algorithmic clarity, and reproducible diagnostics, making *OptLearn* suitable for scientific and engineering optimization tasks.

11 References

1. Nocedal, J., & Wright, S., *Numerical Optimization*, Springer (2006).
2. Kingma, D. P., & Ba, J., *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980.
3. Ruder, S., *An Overview of Gradient Descent Optimization Algorithms*, arXiv:1609.04747.
4. Bertsekas, D. P., *Nonlinear Programming*, Athena Scientific (1999).