STA378 Final Report

Felix Gao

September 2025

1 Abstract

2 Background and Introduction

Unconstrained optimization algorithms

In the field of continuous numerical optimization, the aim is to develop solvers that can locate a local minimum of unconstrained optimization problems:

$$\min_{x \in \mathbb{R}^n} f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and using first-order (gradient vector of size n) and eventually using second-order derivatives (Hessian matrix - symmetric matrix of size n) of the objective function f. There exists a variety of algorithms for such problem and you can find a list of implementations in the Julia package JSOSolver.jl. These algorithms are iterative algorithms that start from an initial guess $x_0 \in \mathbb{R}^n$ and compute a follow-up iterate until a stationary point is reached, i.e, $\nabla f(x) \approx 0$.

In Calculus, the most common optimization algorithms are **Gradient Descent** and **Newton's Method**. However, when the problem becomes very large, it can be very inefficient and hardware expensive to locate a local minimum. For example,

3 Methodology and Implementation

4 Results

Dataframe Description

Column	**Type**	**Description**
`status`	Symbol	**first_order**: solver successfully reached a first-order stationary point; **max_time**: solver couldn't solve the problem within the time limit; **unbounded**: the minimum of the objective function goes to $-\infty$.
`name`	String	Identifier for the optimization problem.
`solver`	String	Name of the solver/algorithm applied.
`mem`	Int	Hyperparameter of the solver.
`nvar`	Int	Number of decision variables in the problem.
`time`	Float64	Total time(s) for solving the problem.
`memory`	Float64	Total memory(MB) used for solving the problem.
`num_iter`	Int	Total number of iterations performed.
`nvmops`	Int	Number of vector–matrix products.
`neval_obj`	Int	Number of objective function evaluations.
`init_eval_obj_time`	Float64	Time (seconds) for the initial objective evaluation.
`init_eval_obj_mem`	Float64	Memory (MB) for the initial objective evaluation.
`init_eval_obj_alloc`	Float64	Number of heap allocations for the initial objective evaluation.
`neval_grad`	Int	Number of gradient evaluations.
`init_eval_grad_time`	Float64	Time (seconds) for the initial gradient evaluation.
`init_eval_grad_mem`	Float64	Memory (MB) for the initial gradient evaluation.
`init_eval_grad_alloc`	Float64	Number of heap allocations for the initial gradient evaluation.
`is_init_run`	Bool	Indicates whether this is the initial run.

Table 1: Description of columns in the solver benchmark data table.

5 Conclusions

6 Future work