A memory gradient method for non-smooth convex optimization

V. Bridge, T. Kostyuk, S. Machado-Marques

MATH6651
Final Project

December 2022

Summary

Introduction

Let $f : R^n \to R$ is a convex function and consider the unconstrained optimization problem

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

Common approaches:

- *Steepest Descent method:*

$$x_k = x_{k-1} + \alpha_k d_k, \ k \geq 1, \tag{1}$$

where $d_k = -\nabla f(x)$ is the k^{th} search direction, α_k is the k^{th} step size, and x_0 is some initial point. Step length α_k is chosen according to a line search method, such that:

$$\alpha_k = arg \min_{\alpha} h(\alpha) = arg \min_{\alpha} f(x_k + \alpha d_k) \tag{2}$$

Introduction

- *Conjugate Gradient method:*

  Redefines $d_k = -\nabla f(x_{k-1}) + \beta_k d_{k-1},\ k \geq 1,$
  where $d_0 = -\nabla f(x_0)$ for some initial x_0.

  For Fletcher-Reeves method,

$$\beta_k = \frac{||\nabla f(x_{k+1})||^2}{||\nabla f(x_k)||^2},\ k \geq 1,$$

Q: What if f is non-differentiable?

Problem Description

Our original unconstrained optimization problem is associated with another problem:

$$\min_{x \in R^n} F(x), \tag{3}$$

where $F : R^n \to R$ is the so-called Moreau–Yosida regularization of f, defined by

$$F(x) = \min_{z \in R^n} \left\{ f(z) + \frac{1}{2\lambda} ||z - x||^2 \right\}, \tag{4}$$

where $\lambda > 0$ and $|| * ||$ denotes the Euclidean norm.

Importantly:

- F is differentiable
- ∇F is Lipschitz continuous
- The solution sets of f and F coincide
- Finding the exact solution is *difficult or impossible*

The paper aims to use a memory gradient method to minimize F.

Preliminaries

Let $p(x)$ be the unique minimizer in (4), i.e.

$$p(x) = arg \min_{z \in R^n} \left\{ f(z) + \frac{1}{2\lambda} ||z - x||^2 \right\}, \tag{5}$$

then $p^\alpha(x, \epsilon) \in R^n$ is an approximation of $p(x)$ such that

$$f(p^\alpha(x, \epsilon)) + \frac{1}{2\lambda} ||p^\alpha(x, \epsilon) - x||^2 \leq F(x) + \epsilon \tag{6}$$

for any $x \in R^n$ and $\epsilon > 0$. Then we can use $p^\alpha(x, \epsilon)$ to define the approximations of $F(x)$ and $g(x) = \nabla F$, by

$$F^\alpha(x, \epsilon) = f(p^\alpha(x, \epsilon)) + \frac{1}{2\lambda} ||p^\alpha(x, \epsilon) - x||^2, \tag{7}$$

and

$$g^\alpha(x, \epsilon) = \frac{x - p^\alpha(x, \epsilon)}{\lambda} \tag{8}$$

Algorithm Definition

INPUT: tolerance ε, number of iterations in memory m, other parameters $\rho, \gamma, \sigma, \beta, \tau = \lambda, \theta_k, \epsilon_k$
OUTPUT: $\min\limits_{x \in R^n} f(x)$

Step 1. Compute $g^\alpha(x_k, \epsilon_k)$. If $||g^\alpha(x_k, \epsilon_k)|| \leq \varepsilon$, then stop.
Step 2. Calculate search direction
if $k \leq m$ **then** $d_k = -g^\alpha(x_k, \epsilon_k)$
else if $k \geq m+1$ **then** $d_k = -\lambda_k g^\alpha(x_k, \epsilon_k) - \sum\limits_{i=1}^{m} \lambda_{k-i} d_{k-i}$
Step 3. determine a step size α_k. Set $x_{k+1} = x_k + \alpha_k d_k$
Step 4. If $\epsilon_k \leq \alpha_k^2 ||d_k||^2$, set $0 < \epsilon_{k+1} \leq \epsilon_k$ and go to next step; otherwise set $\epsilon_k := 0.5\epsilon_k$ and go to Step 4.
Step 5. Set $k := k+1$ and go to Step 1.

The algorithm is doubly iterative: outer cycle consists of steps 1 through 5, inner cycle consists of Step 4 through Step 4.

Convergence Analysis

There are two assumptions and few lemmas that are necessary for convergence analysis.

Assumption A. The level set $\mathcal{L}_0 = \{x \in R^n | F(x) \leq F^{\alpha}(x_0, \epsilon_0) + \sum_{i=1}^{+\infty} \epsilon_i\}$ is bounded, where x_0 is an available initial point.

This assumption is a weaker condition than the strong convexity of f as normally required by other methods for non-smooth convex optimization.
Assumption B f is strongly convex with modulus $c > 0$, i.e.

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) - \frac{1}{2}c\alpha(1-\alpha)||x-y||^2 \quad (9)$$

for all $x, y \in R^n$ and $\alpha \in [0, 1]$

This can be used to show that the Moreau-Yosida regularization F is also strongly convex with modulus $\frac{c}{c\lambda+1}$

Convergence Analysis (continued)

The authors prove two theorems related to the convergence of the algorithm.

Theorem

Suppose that Assumption A holds. If the algorithm generates an infinite sequence $\{x_k\}$*, then any accumulation point of* $\{x_k\}$ *is an optimal solution of problem* (3)*.*

Its main significance is that it proves that with each iteration the solution to Moreau-Yosida regularisation is improved, and hence, at any point we have an optimal solution to the regularisation and the original problem, due to the equivalence of their solutions set.

Convergence Analysis (continued)

The second theorem is about the rate of convergence of the proposed algorithm.

Theorem

Suppose that Assumption B holds. Then the sequence $\{x_k\}$ *generated by the algorithm converges to a unique minimizer* x^* *of* $F(x)$*. Suppose further that*

$$\epsilon_k \leq \beta_k \alpha_k^2 ||d_k||^2, \tag{10}$$

where $\{\beta_k\}$ *is a decreasing positive sequence satisfying* $\lim_{k\to+\infty} \beta_k = 0$*. Then, there exist an infinite subset* $K \implies \{m+1, m+2, ...\}$ *and* $i_0 : 1 \leq i_0 \leq m$ *such that*

$$\lim_{k\in K, k\to+\infty} \frac{||g^{\alpha}(x_k, \epsilon_k)||}{||d_{k-i_0}||} = 0 \tag{11}$$

or $\{x_k\}$ *converges to* x^* *at least R-linearly.*

Convergence Analysis (continued)

Remark To better understand what this theorem states, it is worth looking into the concept of convergence. Rate of convergence generally quantifies how quickly the solution sequence approaches its limit.

$$\lim_{k\to\infty} \frac{|x_{k+1} - x_\infty|}{|x_k - x_\infty|^q} = \mu \tag{12}$$

Such convergence is called "Q-linear".
R-linear convergence means that the rate of convergence is reasonably fast, but it is variable. The definition of convergence in this case is as follows:

Definition

The sequence $\{x_k\}$ is said to converge R-linearly to its limit L if there exist a sequence ϵ_k such that $|x_k - L| \leq \epsilon_k$ for all k, and ϵ_k converges Q-linearly to zero.

Hence, to prove the theorem, it will be shown that there exists such a sequence for $||x_k - x^*||$.

The complete proof is too long for the presentation, so we will only describe the main results.

Proof.

First, it is shown that $\epsilon_k \to 0$ as $k \to \infty$ and $\{x_k\} \to x^*$.

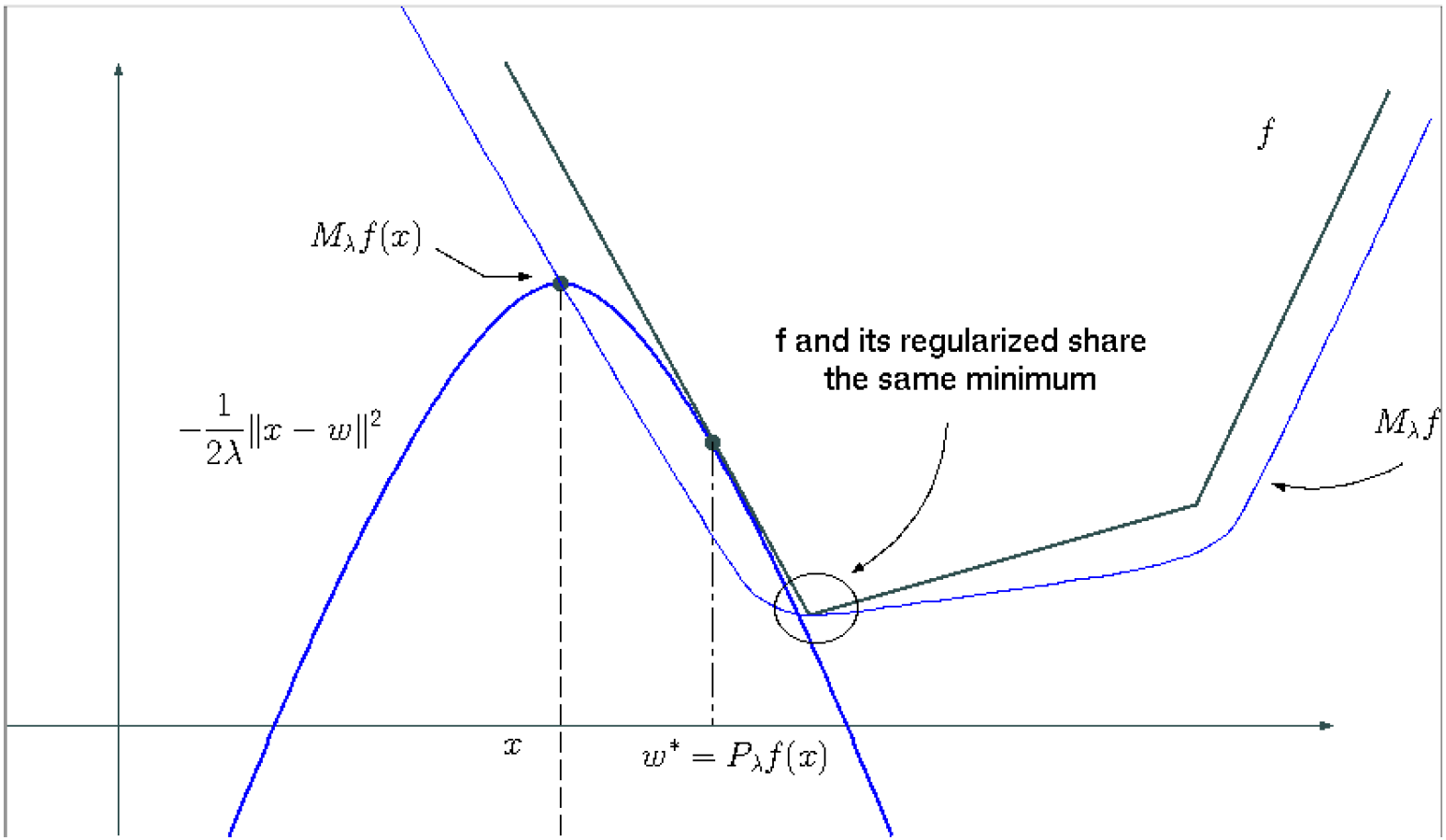
Then, it is shown that

$$||x_k - x^*|| \leq \sqrt{\frac{2(c\lambda + 1)(D_{k_0} - F(x^*))}{c\theta^{k_0}}}(\sqrt{\theta})^k, \forall k \geq k_0,$$

and hence $\{x_k\}$ converges to x^* at least R-linearly. □

Moreau Yosida In Practice

How the algorithm approximates the optimal value we can examine this graph:



Problems Tested

- 1) Generalization of MAXQ:

$$f(x) = \max_{1 \leq i \leq n} (x_i)^2, \; x_0 = (1, 2, ..., \frac{n}{2}, \frac{-n}{2} - 1, ..., -n)$$

- 3) Chained LQ:

$$f(x) = \sum_{i=1}^{n-1} \max\{-x_i - x_{i+1}, -x_i - x_{i+1} + (x_i^2 + x_{i+1}^2 - 1)\},$$

$$x_0 = (-0.5, -0.5, ..., -0.5)$$

- 6) Chained Mifflin 2:

$$f(x) = \sum_{i=1}^{n-1} (-x_i + 2(x_i^2 + x_{i+1}^2 - 1) + 1.75|x_i^2 + x_{i+1}^2 - 1|),$$

$$x_0 = (-1, -1, ..., -1)$$

Implementation Details

As in the paper, parameters chosen:

1. $\rho = 0.001$,
2. $\gamma = 0.85$,
3. $\sigma = 0.01$,
4. $\beta = 0.5$,
5. $\tau = 1$,
6. $\lambda = 1$,
7. m $= 5$,
8. $\theta_k \equiv 0.5$,
9. $\varepsilon = 10^{-10}$,
10. and $\epsilon_k = 1/(k+2)^2$.

Table 2. Numerical results for related algorithms.

P	n	Algorithm 3.1	CG-YWL
1	1000	197/1526/1.7680e−10	225/4710/6.9354e−8
	5000	239/2856/2.3657e−10	250/5235/6.8798e−8
2	1000	96/1297/2.7360e−9	91/1482/8.2738e−9
	5000	121/1568/1.6073e−9	111/1938/9.7206e−9
3	1000	34/112/−1.3740e+3	37/114/7.2687e−9
	5000	39/116/−7.2964e+3	39/120/9.0932e−9
4	1000	73/872/4.1992e−11	77/1026/6.8037e−9
	5000	79/901/8.9864e−11	90/1281/7.8405e−9
5	1000	36/115/1.3470e−11	38/117/7.2687e−9
	5000	41/126/6.3525e−11	40/123/9.0932e−9
6	1000	37/105/−6.9813e+3	37/114/−2.4975e+4
	5000	40/124/−3.4917e+4	39/120/−1.2498e+5
7	1000	32/98/8.4532e−11	37/114/5.4897e−9
	5000	36/107/4.3257e−11	39/120/6.8294e−9
8	1000	35/103/7.3727e−11	39/120/6.8185e−9
	5000	38/115/1.1439e−11	41/126/8.5258e−9

Figure: Optimality Results: Memory Gradient vs Conjugate Gradient

Numerical Results

To test out the algorithm proposed by the authors:

1. We implemented our own version of the memory algorithm in Python
2. We looked for a method of approximating $p(x)$.
3. We selected a minimization Nelder-Mead algorithm from the Python library: *scipy*

We choose a different stop condition due to computational limitations. We apply these parameters to both our Memory Gradient method and Conjugate Gradient method algorithms, instead we chose an iteration number instead of using the stop condition of $||g^{\alpha}(x_k, \epsilon_k)|| \leq \varepsilon$, where $\varepsilon = 10^{-10}$.

Note: While the authors do not clearly indicate their method of approximation, we selected a similar approach to the one described in [2].

Results

Results obtained by different algorithms to solve non-smooth problems.

P	**n**	**Memory Algorithm**	**CG Method**	**Simplex Method**	**Expected Value** [3]
1	20	100/28.4272	100/114.6783	100/307.8883	0
		200/28.3831	200/112.8467	200/256.1166	0
3	2	20/-1.4142	20/0.8237	20/-1.3941	-1.4142
		40/-1.4142	40/0.7632	40/-1.4138	-1.4142
6	2	20/-1.0000	20/-8.3245	20/-1.4000	-1
		40/-1.0000	40/-12.48845	40/-0.9999	-1

References

1. Y. Ou and Y. Liu. A memory gradient method for non-smooth convex optimization, International Journal of Computer Mathematics, 92:8 (2015), 1625-1642.
2. G. Yuan, Z. Wei, and Z. Wang, Gradient trust region algorithm with limited memory BFGS update for nonsmooth convex minimization. Comput Optim Appl 54, 45–64 (2013).
3. A. Hakan TOR, Comparative numerical results on HANSO (Hybrid Algorithm for Nonsmooth Optimization), Abdullah Gül University, Computer Science Faculty, Dept. of Applied Mathematics (2020).