New York University

Master's Thesis

An L-BFGS-B Optimizer for Non-Smooth Functions and some experimentations

Author:
Wilmer Henao

Supervisor:
Michael L. Overton

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science in Scientific Computing

in the

Courant Institute of Mathematical Sciences
Department of Mathematics

March 2014

Declaration of Authorship

I, Wilmer Henao, declare that this thesis titled, 'An L-BFGS-B Optimizer for Non-Smooth Functions and some experimentations' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

NEW YORK UNIVERSITY

Abstract

Faculty Name
Department of Mathematics

Master of Science in Scientific Computing

An L-BFGS-B Optimizer for Non-Smooth Functions and some experimentations

by Wilmer Henao

The Thesis Abstract is written here ...

Acknowledgements

I would like to thank my advisor Michael Overton for all the hours of hard work and for all the great recommendations and changes that led to this thesis.

I also would like to thank Allan Kaku and Anders Skaaja for earlier contributions to other versions of the code.

Finally, I would like to thank High Performance Computing at NYU.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

Contents iv

1 Introduction 1

2 L-BFGS-B 3
2.1 BFGS . . . 3
2.2 L-BFGS . . . 4
2.3 L-BFGS-B . . . 4
2.3.1 Gradient Projection . . . 4
2.3.2 Subspace Minimization . . . 5

3 Modifications to the L-BFGS-B algorithm 7
3.1 The Wolfe conditions . . . 7
3.2 The line search methodology . . . 8
3.3 The termination condition . . . 8
3.3.1 The Solution of the quadratic program . . . 9

4 the solution test functions 10

A Appendix Title Here 12

Bibliography 13

LAH **L**ist **A**bbreviations **H**ere

For/Dedicated to/To my...

Chapter 1

Introduction

The problem addressed is to find a local minimizer of the nonsmooth minimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & l_i \leq x_i \leq u_i, \\ & i = 1, \ldots, n. \end{aligned} \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, is continuous but not differentiable everywhere and n is a very large but finite number.

The $L - BFGS - B$ algorithm [1] is a standard method for solving large instances of 1.1 when f is a smooth function. The original name of $BFGS$ stands for Broyden, Fletcher, Goldfarb and Shanno, the authors of the original "BFGS" quasi-Newton algorithm for unconstrained optimization discovered and published independently by them in 1970 [give the 4 references here]. This method requires storing and updating a matrix which approximates the inverse of the Hessian matrix $\nabla^2 f(x)$ and hence requires $\mathcal{O}(n^2)$ operations per iteration. The $L - BFGS$ variant [give reference] is based on BFGS but requires only $\mathcal{O}(mn)$ operations per iteration: thus, the L stands for Large. Finally, the last letter B in $L - BFGS - B$ stands for bounds, meaning the lower and upper bounds l_i and u_i in 1.1. $TheL - BFGS - B$ algorithm is implemented in a well known $FORTRAN$ software package by the same name [give reference].

In this thesis, there is a brief description of the $L - BFGS - B$ algorithm at a high level and then explain how the modified algorithm is more suitable for functions f which may not be differentiable at their local or global optimizers. We call the new algorithm L-BFGS-B-NS where NS stands for Non-Smooth. We implemented these changes in a

modified version of the Fortran code [ref] which is can be downloaded from the website for this thesis [give URL]. We report on some numerical experiments that strongly suggest that the new code should be useful for the nonsmooth bound-constrained optimization problem (1.1).

We are grateful to Jorge Nocedal and his coauthors for allowing us to modify the L-BFGS-B code and post the modified version.

Chapter 2

L-BFGS-B

This section is a description of the original $L-BFGS-B$ code at a very high level [2]. The original software is intended to work well with smooth functions. This thesis discusses how to modify the algorithm for Non-Smooth functions.

2.1 BFGS

$BFGS$ is a standard tool for optimization of smooth functions. It is a line search method and its goal is to find a search direction starting from its current position x. The search direction is of type $d = -B\nabla f$ [1] where B is an approximation to the inverse Hessian. [2] This k^{th} step approximation is calculated via the $BFGS$ formula

$$B_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) B_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k} \tag{2.1}$$

where $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and $s_k = x_{k+1} - x_k$ and where the first approximation of B_0 is assumed to be the the identity matrix I in this thesis. $BFGS$ exhibits superlinear convergence but it also requires $\mathcal{O}(n^2)$ operations per iteration. [3]

In the case of Non-Smooth functions. $BFGS$ typically succeeds in finding a local minimizer. This is partly because $BFGS$ has some nice self-correcting properties, however this requires some modifications of the line search conditions. This change to the line search conditions is known as the weak Wolfe line search and it will be explained later in this thesis.

[1] Notice that when B is the identity, this is the same direction as steepest descent. Another common line search method of optimization

[2] When it is exactly the inverse Hessian the method is known as Newton's method. Newton's method has quadratic convergence but requires the explicit calculation of the Hessian at every single step.

2.2 L-BFGS

$L-BFGS$ stands for Limited-memory $BFGS$. This algorithm approximates $BFGS$ using only a limited amout of computer memory to approximate the inverse of the Hessian B. So Instead of storing a dense $n \times n$ matrix, $L-BFGS$ keeps a record of the last m iterations where m is a small number that is chosen according to the problem at hand. [3] It is for this reason that during the first m iterations, $BFGS$ and $L-BFGS$ produce exactly the same search directions.

Because of this construction, the $L-BFGS$ algorithm is less computationally intensive and it only requires $\mathcal{O}(mn)$ iterations. So it is much better suited for problems where the number of dimensions n is large. For this reason it is the algorithm of choice in this thesis.

2.3 L-BFGS-B

Finally $L-BFGS-B$ comes naturally as an extension of $L-BFGS$. The B stands for the inclusion of Boundaries. $L-BFGS-B$ requires two extra steps on top of $L-BFGS$. First, a gradient projection that reduces the dimensionality of the problem. Depending on the problem, the gradient projection could potentially save a lot of iterations by eliminating those variables that are at bound at the optimum. After that, the subspace minimization; here, during the search step phase the step length is restricted as much as necessary in order to remain within the box defined by 1.1.

2.3.1 Gradient Projection

The original algorithm was created for the case when n is large and f is smooth. Its first step is a gradient projection similar to the one outlined in [4, 5] which is used to determine an active set corresponding to those variables that are on either their lower or upper bounds. The active set defined at point x^* is:

$$\mathcal{A}(x^*) = \{i \in \{1 \dots n\} | x_i^* = l_i \vee x_i^* = u_i\} \tag{2.2}$$

Working with this active set is efficient in large scale problems. A pure line search algorithm would have to choose a step length short enough to remain within the box defined by u_i and l_i. So if at the optimum, a large number $\mathcal{B}$ of variables are either

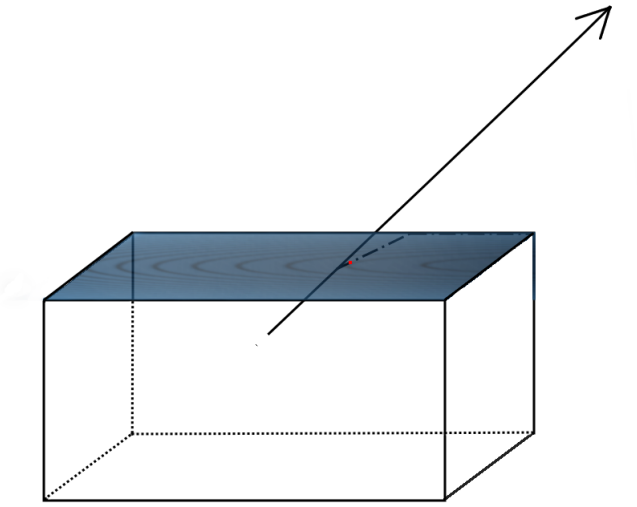
[3] In this thesis $m < 20$, and in practice numbers between 5 and 10 are regularly used. There is no way of knowing a priori what choice of m will provide the best results

on the lower or the upper bound. At least a number $\mathcal{B}$ of iterations might be needed. Gradient projection tries to reduce this number of iterations. In the best case, only 1 iteration is needed instead of $\mathcal{B}$.

Gradient projection works on the approximation model:

$$m_k(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{(x - x_k)^T H_k(x - x_k)}{2} \tag{2.3}$$

where H_k is a $L - BFGS - B$ approximation to the Hessian $\nabla^2 f$ stored in the implicit way defined by $L - BFGS$.



In this first stage of the algorithm a piecewise linear segment starts on the current point x_k in the direction $-\nabla f(x_k)$. Whenever this direction encounters one of the constraints, the line segment turns corners in order to remain feasible. The path is nothing but the feasible piecewise projection of the negative gradient direction on the constraint box determined by the values $\overrightarrow{l}$ and $\overrightarrow{u}$. At the end of this stage, the value of x that minimizes $m_k(x)$ restricted to this piecewise gradient projection path is known as the 'Cauchy point' x^c.

2.3.2 Subspace Minimization

The problem with gradient projection is that its search direction does not take advantage of information provided implicitly by H_k, and therefore the speed of convergence is at best linear. It is for this reason that a stage two is necessary. Stage 2 or subspace

minimization uses an $L-BFGS$ implicit approximation of the Inverse Hessian matrix restricted to the free variables that are not in the active set $\mathcal{A}(x^c)$.

The idea at a higher level is to solve the constrained problem 2.3, but only on those dimensions that are free. The starting point for this new problem will be the previously found Cauchy point x^c, the $L-BFGS$ approximation will provide a new search direction $\hat{d}^u$ that takes implicit advantage of second order approximations of the Hessian matrix.

The algorithm will move in the direction $\hat{d}^* = \alpha^* \hat{d}^u$ where α^* (the step length) is chosen so that the new point $\bar{x}_i$ satisfies weak wolfe descent and curvature conditions, A third restriction on the step length is added so that the next iteration stays feasible. Once this step is finished, the next and final step will be the termination condition. If the termination condition fails a new gradient projection and subspace minimization will be needed.

Chapter 3

Modifications to the L-BFGS-B algorithm

We made three main changes to the original $L-BFGS-B$ algorithm. They concern the line search Wolfe conditions, the line search methodology, and the termination condition.

3.1 The Wolfe conditions

Probably the most important change made to the original code was the change in the curvature condition. It is accepted that the Wolfe conditions work very well whenever the function f is smooth [6]. Originally there are two Wolfe conditions: one of them is the Armijo condition, also known as the sufficient decrease requirement. The other one is the curvature condition, of which the most popular version is the "strong Wolfe" curvature condition:

$$|d_k^T \nabla f(x_k + \alpha_k d_k)| \leq |d_k^T \nabla f(x_k)| \tag{3.1}$$

Where d_k represents the search direction. The strong Wolfe condition is natural for smooth optimization. Its goal is to find a step length long enough that the slope has been reduced "sufficiently", but the problem is that it does not work well for the Non-Smooth case. This is because near the minimal points, there may be abrupt changes in the gradient. For example the curvature condition can never be satisfied when $f(x) = |x|$, because the slope never becomes flat. The "weak wolfe" condition

$$d_k^T \nabla f(x_k + \alpha_k d_k) \geq d_k^T \nabla f(x_k) \tag{3.2}$$

is all that is needed to guarantee that the $BFGS$ updated inverse Hessian approximation is positive definite [7]. This weak version is the one that will be used in this thesis.

3.2 The line search methodology

The original software by Nocedal[2], included a cubic line search. The idea of a cubic interpolation line search is to take advantage of the smooth properties of function f in order to find a more convenient point. But in the case of nonsmooth functions, this line search does not serve our purposes and therefore a more typical line search that implements a bisection has to be used.

A step length is selected. If this step length does not satisfy the sufficient decrease and curvature conditions then a step length of half or double the size is selected. The algorithm is guaranteed to converge under a careful selection of the parameters.

3.3 The termination condition

One important requirement of a practical algorithm is that it ends in a finite time. For the case of smooth functions, the usuall way to check whether the algorithm has converged, is by using the projected gradient which is nothing but the projection of the negative gradient onto the bounding box defined by l and u. If this projected gradient has a small norm the algorithm has converged. In the case of nonsmooth functions however, this is not necessarily true and the function at the minimum, may have a wedge. In this wedge the projected gradient may not vanish. Furthermore, if there is a sequence of points that approaches the optimum x from the right, the projected gradients corresponding to this sequence of points might be completely different from the projected gradients associated to a sequence of points that approach the optimum x from the left.

Given this set of conditions, there is the need for a special set of rules to establish the finalization of each optimization.

The best practical methodology should be to calculate wheter zero 0 or a very small number is the norm of a vector that is part of the convex hull of gradients evaluated at points near the optimum candidate x [7]. In order to make sure that the gradient zero $\vec{0}$ or a vector with a very small norm is part of the convex hull calculated near a neighbourhood of the optimum, the algorithm keeps a record of the latest gradient vectors in this small neighbourhood of the point where it suspect that the optimum is located.

If there exists a vector with such a small norm, the algorithm ends. The best way to find a vector with such properties is to find the vector with the minimal norm that resides in the convex hull. If the minimum such norm is larger than the tolerance, the algorithm must continue to the following iteration and not terminate.

3.3.1 The Solution of the quadratic program

Practical experience has shown that the termination condition is able to show when the function has converged to a local optimum. However in order to find the vector with the minimal norm one subalgorithm needs to be solved. This subalgorithm is a practical primal-dual algorithm implemented in [8]. In this case in particular the best solution is to implement a variation of Mehrotra's Predictor-Corrector algorithm applied to quadratic programming.

The algorithm is widely used in practice with great results. The implementation on [8] is is exactly the one on [3]. In this thesis it was implemented in $FORTRAN$ as part of the optimizer.

Chapter 4

the solution test functions

In order to make some tests, a few functions will be evaluated. The most important function to test this non-smooth optimizer is a modified version of rosenbrock's:

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} |x_i - x_{i-1}^2|^p \tag{4.1}$$

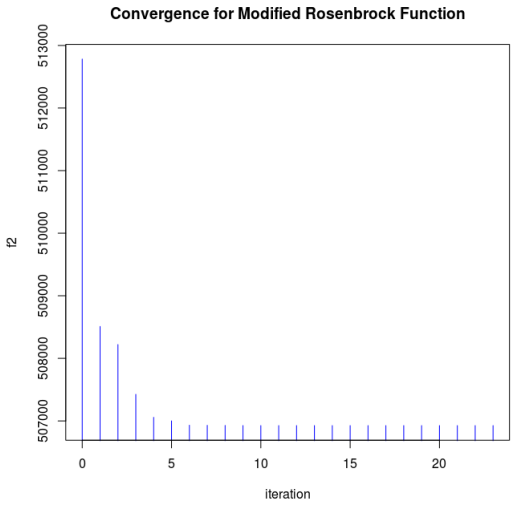
Where the value of p changes the behaviour of the optimizer. This function can be proven to be lipschitz continuous whenever $p > 1$ if restricted to the domain defined by

$$x_i = \begin{cases} [-100, 100] & \text{if } i \in \text{ even numbers} \\ [10, 100] & \text{if } i \in \text{ odd numbers} \end{cases} \tag{4.2}$$

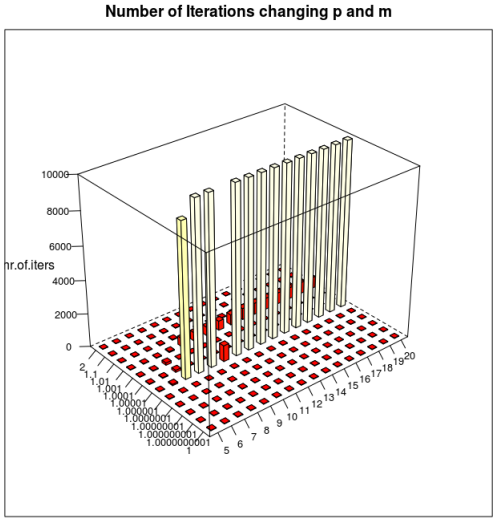
and in fact, whenever the function is restricted to a finite domain this function will be lipschitz continuous for $p > 1$. Whenever $p > 1$ the function $f(z) = |z|^p$ is zero 0 around zero because the derivative $p|z|^{p-1}$ is zero whenever z tends to zero from the right. (this is also the case from the left because it is an even function). However the second derivative will not be as nice.

For the case when $p \leq 1$ the second derivative tends to infinity. $\lim_{x \to 0^+} f' = \infty$. Which is already well known given the "heavyside" look of $f(z) = |z|$.

The convergence of the algorithm smoothly descends to the objective



The converge is adversely affected by the selection of p as one would expect. Values of p descending to 1 make the function less "smooth" and have the adverse effect of making the convergence much more difficult. In this exercise it is noticeable how slow the convergence becomes for a few specific values of p. In particular for 1.0001



$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k p_k^T \nabla f(x_k) \tag{4.3}$$

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

[1] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ci You Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, 1995. ISSN 1064-8275. doi: 10.1137/0916069. URL http://dx.doi.org/10.1137/0916069.

[2] Ciyou Zhu, Richard Byrd, Jorge Nocedal, and Jose Luis Morales. Lbfgsb 3.0. http://www.ece.northwestern.edu/~nocedal/lbfgsb.html, 2011.

[3] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999. ISBN 0-387-98793-2. doi: 10.1007/b98874. URL http://dx.doi.org/10.1007/b98874.

[4] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM J. Numer. Anal.*, 25 (2):433–460, 1988. ISSN 0036-1429. doi: 10.1137/0725029. URL http://dx.doi.org/10.1137/0725029.

[5] Jorge J. Moré and Gerardo Toraldo. Algorithms for bound constrained quadratic programming problems. *Numer. Math.*, 55(4):377–400, 1989. ISSN 0029-599X. doi: 10.1007/BF01396045. URL http://dx.doi.org/10.1007/BF01396045.

[6] Dong-Hui Li and Masao Fukushima. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. *SIAM J. Optim.*, 11(4):1054–1064 (electronic), 2001. ISSN 1052-6234. doi: 10.1137/S1052623499354242. URL http://dx.doi.org/10.1137/S1052623499354242.

[7] Adrian S. Lewis and Michael L. Overton. Nonsmooth optimization via quasi-Newton methods. *Math. Program.*, 141(1-2, Ser. A):135–163, 2013. ISSN 0025-5610. doi: 10.1007/s10107-012-0514-2. URL http://dx.doi.org/10.1007/s10107-012-0514-2.

[8] Anders Skaaja. Limited memory bfgs for nonsmooth optimization. Master's thesis, New York University, Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012, 2010.