Alternatives to conjugate direction optimization for sparse solutions to geophysical problems

Nader Moussa

ABSTRACT

Throughout much of this summer, we experimented with extensions to the conjugate direction method to find optimal solutions to sparse geophysical problems. However, this category of techniques is not unique in its ability to optimize L1-styled fitting goals. We also investigated a variety of other techniques, including a pure L1 solution via the weighted median; a steepest-descent algorithm using the signum-function as a gradient of the true L1 norm; and a totally different approach using the Simplex Algorithm, by mapping our objective function into a linear programming form. Categorically, the approaches that relied on the true L1 method failed due to what we believe is a theoretical shortcoming of the direct application of the pure L1 norm to geophysical optimization problems. The use of linear programming turned out to be quite successful. This could be an interesting option for future research in geophysical optimization.

INTRODUCTION

Part of our objective in this summer's study of the L_1 optimization criteria was motivated by new theoretical ideas for the conjugate direction solver (Claerbout, 2009), and its corresponding implementation (Maysami and Moussa, 2009). In addition to this new technique, we also extensively investigated the basic theory of convex optimization, motivated by our ultimate desire to find the most generally applicable toolkit for geophysical inversion on sparse or "blocky" models. Optimization theory has been subject to much research at Stanford across many fields. Prior art that is directly applicable to L_1 minimization spans the departments of Geophysics (Claerbout, 2008; Guitton, 2000), Computer Science (Golub and Van Loan, 1996), Operations Research, Management Science & Engineering (Paige and Saunders, 1982), and Electrical Engineering (Boyd and Vandenberghe, 2009). The enormous wealth of prior research across so many different disciplines has produced numerous algorithms and mathematical techniques which superficially bear no resemblence to each other – but all share the same final goal, which is the minimization of a generalized convex objective function. For the case of conventional geophysical inversion, this objective is some measure of the error between modeled- and recorded- data.

This broad-based investigation brought attention to techniques, such as linear programming, which are well-developed and widely used in other fields. However, these tools were rarely utilized by SEP (and presumably in the geophysical inversion community outside Stanford). Our efforts have developed a formulation of these techniques to convert a standard form geophysical data-fitting and inversion problem ($\mathbf{L}\ \mathbf{m} = \mathbf{d}$) into a linear programming problem.

I demonstrate in the following sections the efforts to construct a pure L_1 solver, and its associated numerical difficulties. Next is our foray into the realm of linear programming – a well-developed toolset that has seen little application in geophysical inversion.

PURE L1 SOLUTIONS

Early work focused on a pure conjugate-gradient or steepest-descent method using the true L_1 norm. We implemented a modified version of `cg_step`, implementing a line search first as a weighted median search, and also with a full, explicit calculation of the Frechet derivative. It was conclusively shown that the gradient of a pure L_1 solution resulted in introduction of local minima, resulting in a failure to converge. This has been noted in prior work (Bube and Langan, 1997) – the pure L_1 norm is not strictly convex. This can be shown with the trivial example of finding the L_1 minimum for a median problem with even number of elements:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} m_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{1}$$

The residual in the L_1 case is the sum $|m-1| + |m-2|$, which has nonunique global minima for all $m \in \big[1, 2\big]$. The gradient, in this entire region, is exactly zero. In a larger, non-trivial data-fitting problem, these zero-value gradients can occur at locations other than the global optimum - and thus the gradient-based methods do not function properly.

LINEAR PROGRAMMING

Theory

The goal of Linear Programming, like optimization in general, is to maximize an objective function, subject to a set of constraints. In this case, the objective function is a linear equation in $\mathbf{x}$.

Linear programming developed out of the Operations Research community following World War II (Dantzig, 1963). It developed from earlier planning algorithms and optimization methods, finally emerging as one of the simplest tools that could optimally satisfy a set of competing equations. It is commonly used in business analytics, supply-chain management, and path planning. Variations of the concept have been applied to geophysical problems, including tomography inversion (Berryman, 1989); but in general, the geophysical inversion community has not shown strong adoption of this body of techniques. This is unfortunate, because the methodology of constructing and navigating within a feasible region of solutions enables both numerical optimization to find a global minimum, as well as heuristic "picking" interpretations of other valid, non-minimum solutions that satisfy the problem constraints.

We have successfully mapped general-purpose geophysical optimization problems into this numerical framework, particularly emphasizing data-fitting with linear operators. Due to its widespread use in other fields, a variety of software tools exist to find solutions to linear programming problems. A variety of numerical programming environments provide linear programming solvers natively; we explored several of these environments and evaluated their solver implementations. The most promising environment of the ones we considered is the GNU Scientific Library and the GNU Linear Programming Kit (GLPK), with language bindings for C, Python, and FORTRAN. Tools and libraries are also available for GNU Octave and its commercial equivalent, MATLAB.

The setup for linear programming revolves around a generalized parameter space, $\mathbf{x}$, which we seek to modify until an optimum is found with respect to some objective function $\mathbf{c}$ subject to constraints $\mathbf{b}$.

We have:

$$x_i = \text{the parameter space} \tag{2}$$
$$c_i = \text{the objective definition} \tag{3}$$
$$a_{ij} = \text{the definition of the constraints} \tag{4}$$
$$b_j = \text{the constraints vector} \tag{5}$$

where i ranges from 1 to the number of parameters; and j ranges from 1 to the number of constraints.

Additional physical constraints are represented numerically by introducing *auxiliary variables*, denoted $\mathbf{x_s}$, and adding an objective coefficient c_i for each introduced constraint. Minimization of $|\mathbf{x_s}|$ is equivalent to optimal satisfaction of these physical constraints; this objective competes with the minimization of $|\mathbf{x}|$. This relationship between model- and data-fitting constraints is the basis of the optimzation problem.

General solutions to this optimization can be found according to the *Simplex Algorithm* (Dantzig, 1963), design of which is rooted in topological graph theory. To find an optimal solution, the system is represented in augmented matrix form. A scalar value Z is constructed, representing the objective function minus the constraints weightings, for any given intermediate solution. At each iteration, the simplex solver evaluates the maximal gradient of Z, with respect to a current set of basis equations, $\left[x|x_s\right]^T$.

This gradient provides a descent direction; the model x is updated accordingly, traversing the linear system until a vertex of the solution graph is found. At each vertex, a new basis set is calculated by matrix transvection (or equivalent, but slower, Gaussian elimination). This allows the objective scalar Z to be expressed in terms of the new basis $\left[x|x_s\right]^T$ – in other words, by rotating the state-matrix according to a transvection operator. (Transvection simply involves adding a scalar multiple of one row to another row). By choosing the correct transvections, the next descent direction can be directly read as the coefficients of the matrix representation. It can be shown that an optimum solution, if one exists, must lie on either a vertex or as a set of equally-optimum elements on a single edge of the solution space. This lies within the *feasible region* defined by the constraining equations.

Mapping Optimization Problems to Linear Programming

This maps to our conventional model-fitting treatment using optimization theory according to the following:

$$x_i \xrightarrow[\text{parameter space}]{} \text{model space} \tag{6}$$

$$c_i \xrightarrow[\text{objective function}]{} \text{regularization} \tag{7}$$

$$a_{ij} \xrightarrow[\text{constraint functions}]{} \text{forward operator} \tag{8}$$

$$b_j \xrightarrow[\text{constraint vector}]{} \text{data space} \tag{9}$$

with $i = 1..(\text{model size})$ and $j = 1..(\text{data size})$ Conveniently, this allows a direct representation of geophysical data fitting, including regularization on the model space. This linear programming setup can be represented in matrix form, as follows:

$$\mathbf{c}^T\,\mathbf{x} = Z \tag{10}$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{11}$$

I have introduced the scalar, Z, which is the value of the objective. This will be either minimized or maximized depending on the particular geophysical problem. Effectively, this means finding a value for $\mathbf{x}$ that optimally aligns with the model-fitting and data-fitting goals.

Correspondingly, for a simple 2×3 example:

$$\begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = Z \tag{12}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{13}$$

To solve according to the L_1 norm, we need to take account of the absolute value in the definition of the L_1 error criteria, noting that the i^{th} element of the residual is defined as an absolute value of the error term:

$$r_i = |\mathbf{L}\ \mathbf{m} - \mathbf{d}| \tag{14}$$

To account for this, we extend the linear programming matrix representation to *augmented form.* This requires an extension of the $\mathbf{x}$ vector. The approach is not entirely dissimilar to the placement of a regularization in the model vector in a conventional setup, in that it is reformulating the equations to provide us with a fit in compliance with our *a priori* geophyiscal knowledge.

The augmented representation is written in matrix form as

$$\begin{bmatrix} \mathbf{1} & -\mathbf{c}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{I} \end{bmatrix} \begin{bmatrix} Z \\ \mathbf{x} \\ \mathbf{x_s} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} \tag{15}$$

The first row of the augmented form results in the optimization criteria:

$$Z - \mathbf{c}^T\mathbf{x} = 0 \tag{16}$$

A large value of Z maximizes the original objective function, subject to the constraints in the $\mathbf{c}$ matrix. The goal is to put as much "energy" in the Z (objective function) with as little energy in all other rows of the augmented matrix; this is accomplished with the *Simplex Algorithm*, simultaneously satisfying the minimization of the pure L_1 norm criteria.

L_1 Formulation

To satisfy the L_1 optimization criteria for a data fitting problem, with data $\mathbf{d}$ of length N_D, and model $\mathbf{m}$ of size N_M, modeled by $|\mathbf{Lm} - \mathbf{d}| = r$, we set up the following to specify the terms in (15) and (16):

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{2N_D} & -\mathbf{L} \\ \mathbf{I}_{2N_D} & -\mathbf{L} \end{bmatrix} \tag{17}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{d} \\ -\mathbf{d} \end{bmatrix} \tag{18}$$

$$\mathbf{c}^T = \begin{bmatrix} 0 & \mathbf{1}_{2N_M} \end{bmatrix} \tag{19}$$

The initial model can be stored in $\mathbf{x}$, which will be updated. The final value of auxiliary variables $\mathbf{x_s}$ represents the status of the model regularization constraints (which can also be assigned an initial value).

$$\mathbf{x} = \mathbf{m} \tag{20}$$
$$\mathbf{x_s} = \mathbf{m}_{regularization} \tag{21}$$

In this format, the matrices can be fed directly into the GLPK function call in C or Octave.

RESULTS

The linear programming formulation was among our most robust techniques for estimating and tracking water-level drift in our synthetic 1-D Sea of Galilee. Below are results plotted from a sample experiment. In this case, every single data point was corrupted by a water-level drift of unknown magnitude, and water-level was allowed to vary at any time during synthetic data collection. After ten sweeps, the linear programming solver is able to nearly perfectly reconstruct the drift profile and correctly estimates the true lake depth profile (except for minor artifacts).

This problem is an example of an underconstrained inversion problem. The source data is corrupted by an unknown data drift error function, which we seek to estimate based on our approximation that it should be defined by a sparse derivative. This model approximates a boat that is sampling lake-depth while floating on an unknown water level, illustrated in Figure 1. The problem is under-constrained, because we do not know the nature of the drift function; but by assuming that its derivative is sparse, an L_1 or linear programming optimization problem is set up. In Figure 2, the result of the linear programming drift estimation is shown.

I conclude that linear programming yields a fairly deterministic result, even in the case of an underconstrained inversion problem. The method is stable and straightforward. The most serious drawback is that most implementations of the Simplex Algorithm require an explicit definition of the forward operator (in matrix form) – so for very large geophysical problems, this can severely limit its applicability.

ACKNOWLEDGEMENTS

This work was spurned by Jon Claerbout's theoretical appreciation for alternate methods of optimization. Mohammad Maysami experimented with a Python linear programming code which successfully solved many test cases. The matrix mapping between Linear Programming and more conventional SEP-style data-fitting was largely developed with Elita Li. Bob Clapp's programming and technical assistance also expedited our progress.

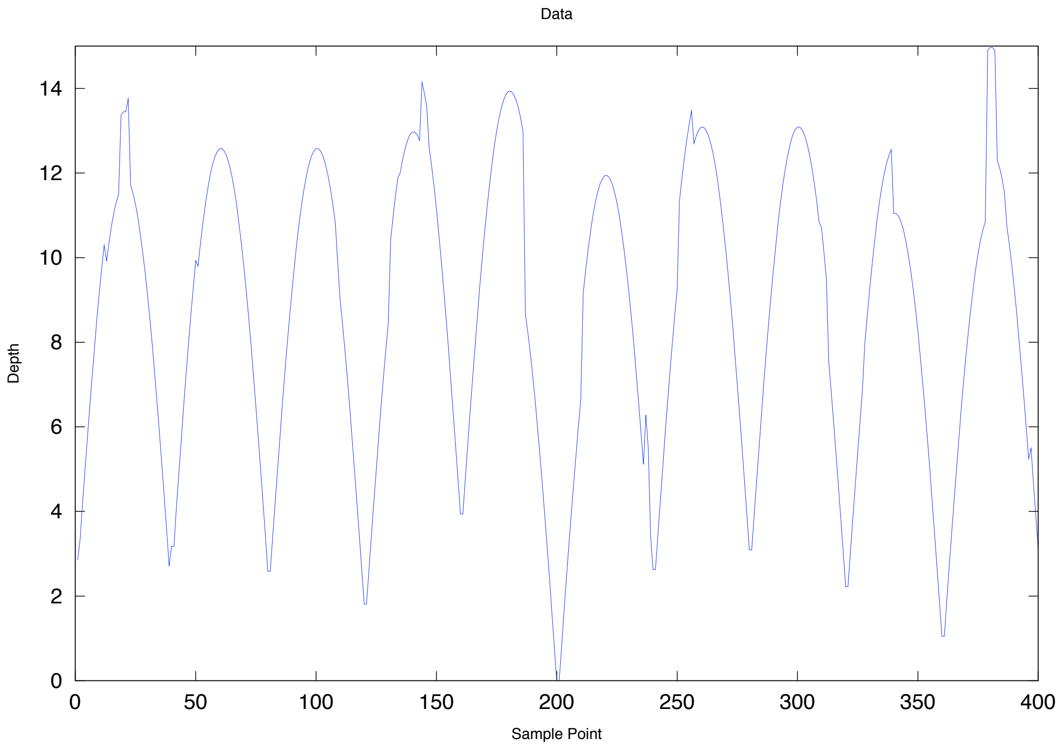

Figure 1: The input data for the linear programming Galilee estimation. **[CR]**

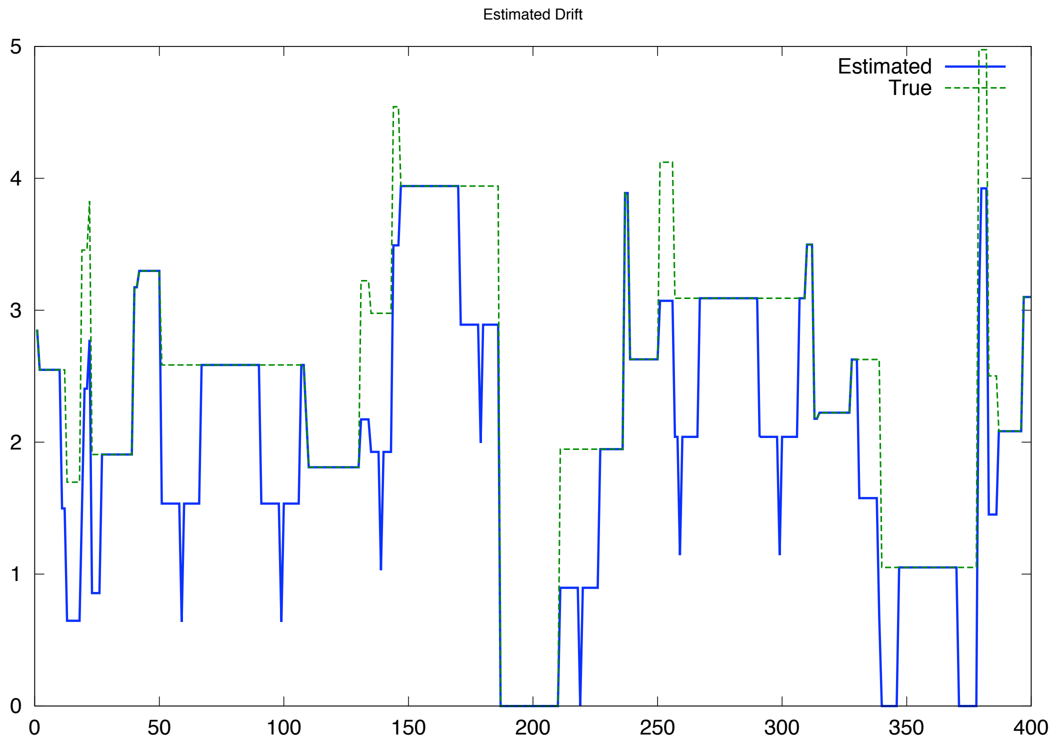

Figure 2: The estimated lake drift. **[CR]**

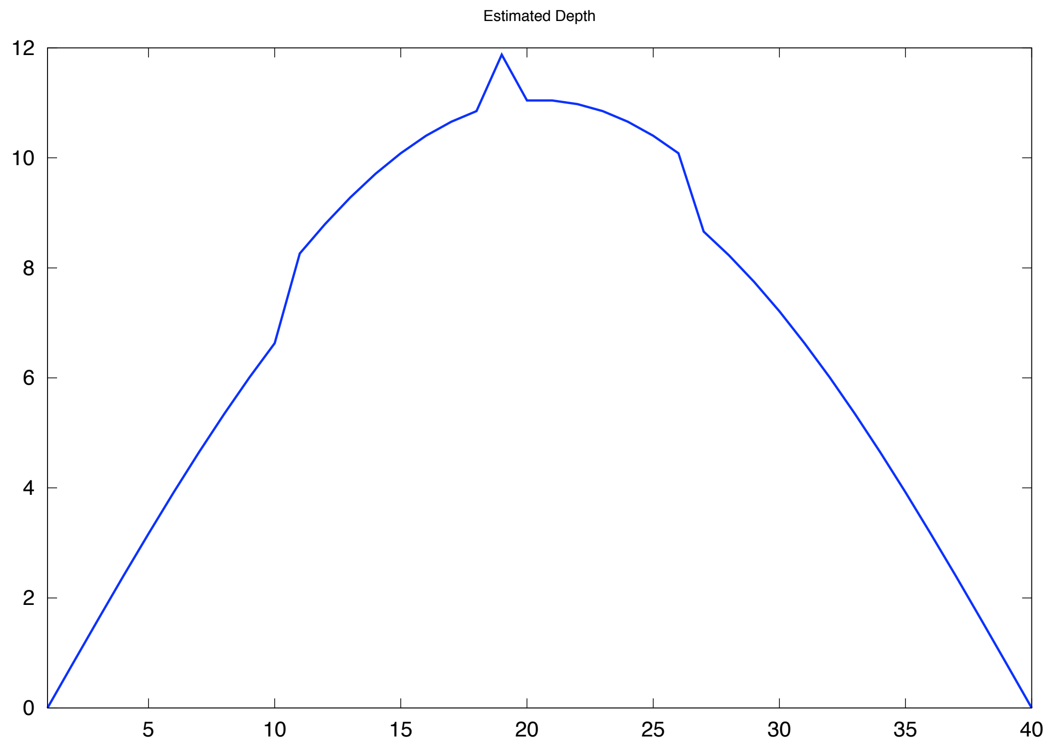

Figure 3: The estimated lake depth. [**CR**]

REFERENCES

Berryman, J. G., 1989, Seismic crosshole tomography and nonlinear constrained optimization: Proceedings of the SIAM Geophysical Inversion Workshop, 396–414.

Boyd, S. and L. Vandenberghe, 2009, Convex optimization, seventh printing ed.

Bube, K. P. and R. T. Langan, 1997, Hybrid l^1/l^2 minimization with applications to tomography: Geophysics, **62**, 1183–1195.

Claerbout, J. F., 2008, Image estimation by example.

———, 2009, Blocky models via the $l1/l2$ hybrid norm: SEP-Report, **139**, 1–10.

Dantzig, G. B., 1963, Linear programming and extensions.

Golub, G. H. and C. F. Van Loan, 1996, Matrix computations, 3rd ed.: The Johns Hopkins University Press.

Guitton, A., 2000, Huber solver versus IRLS algorithm for quasi L1 inversion: SEP-Report, **103**, 255–270.

Maysami, M. and N. Moussa, 2009, Generalized-norm conjugate direction solver: SEP-Report, **139**, 11–22.

Paige, C. C. and M. A. Saunders, 1982, Algorithm 583 LSQR: spare linear equations and sparse least squares problems: ACM Transaction on Mathematical Software, **8**, 195–209.