Proximal Algorithms for Basis Pursuit Denoising

Francis Moran *fdm31@scarletmail.rutgers.edu*

Ali Zafari *ali.zafari@rutgers.edu*

Github Repository: ***[link]***

Abstract

This paper aims to describe the theory and application of modern proximal optimization algorithms with respect to the Basis Pursuit Denoising problem. First, a look at proximal optimization theory will take place, followed by the construction of fundamental algorithms. Next, the problem statement will be described and potential algorithms will be discussed. Specifically, this paper will focus on three algorithms: Iterative Shrinking Thresholding Algorithm (ISTA), Fast Iterative Shrinking Thresholding Algorithm (FISTA), and Sparse Reconstruction by Separable Approximation (SpaRSA). Experimentation will be performed on a sample image and results will be compared.

1 Introduction

1.1 Motivating Proximal Algorithms

We have seen gradient based algorithms to find solutions of unconstrained minimization problems over the semester [1]. When the objective function is non-smooth (non-differentiable) these algorithms and their convergence analysis are no longer valid. The basic solution when the objective function is still convex but not necessarily differentiable is to use the method of subgradients. Although it fixes the differentiability requirement of gradient-based methods, its notorious slow convergence rate prohibits its practical usage. This disadvantage is where the proximal algorithms shine as they can provide the same convergence rate offered by gradient-based methods for non-smooth convex objective functions [2]. In the next section we briefly analyze the convergence rate of subgradient methods to motivate the topic of this project.

1.1.1 Convergence of Analysis of Subgradient Methods

The subgradient method is the non-smooth version of gradient descent. The basic algorithm is straightforward, consisting of the iterations:

$$x_{k+1} = x_k - \alpha_k \Delta x_k$$

where the Δx_k is any member of $\partial f(x_k)$.

Definition 1.1 (Subgradient) *Subgradient of function f at x is a vector $g \in \mathbb{R}^n$ such that*

$$f(y) \geq f(x) + g^T(x - y) \;\; \forall y \in \mathrm{dom} f$$

Collection of subradients at x is called the subdifferential at x denoting as $\partial f(x)$.

The major disadvantage of subgradient methods is their extremely slow rate of convergence. Further, some cases may provide no guarantee for convergence. The next two theorems show this claim, the proofs can be found in [3].

Theorem 1 (Subgradient Non-convergence!) *Let f be a convex Lipschitz continuous function with $M > 0$. Suppose that step size $\alpha_k = \alpha > 0$ is fixed for all k. Then*

$$f_k^{best} - f(x^\star) \leq \frac{1}{2\alpha k}\|x_0 - x^\star\|_2^2 + \frac{\alpha M^2}{2}$$

Subgradient method does not guarantee a decrease in function value at each iteration, so we keep the best after k^{th} iteration, f_k^{best}. For fixed step size, convergence is not guaranteed.

Theorem 2 (Subgradient Convergence) *Let f be a convex Lipschitz continuous function with $M > 0$. Suppose that step sizes satisfy $\alpha_k \to 0$ as $k \to \infty$ and $\sum_{k=1}^{\infty} \alpha_k = \infty$. Then the achievable rate for a general f is*

$$f_k^{best} - f(x^\star) \leq \frac{1}{\sqrt{k}}\|x_0 - x^\star\|_2^2 + Const.\frac{M^2 \log k}{\sqrt{k}}$$

This convergence rate is slow compared to gradient descent. The choice of step size heavily affects the convergence rate of subgradient method.

1.2 Search for Acceleration

To avoid the extremely slow convergence rate of subgradient methods, in cases which the proximal mapping (to be defined in the next section) can be done efficiently proximal algorithms are introduced. These algorithms have similar convergence rates to those of gradient descent methods.
It has also been shown that the convergence rate of gradient descent methods can be improved significantly by momentum-based methods. Informally, this involves mixing some history of gradient updates into the current estimate. Using the same gradient descent method but with new updates:

$$y_k = x_k + \beta_k(x_k - x_{k-1})$$
$$x_{k+1} = y_k - \alpha_k \nabla f(y_k)$$

where $\beta_k = \frac{k-1}{k+2}$. The next theorem indicates this claim.

Theorem 3 (Nestrov's Optimal Method) *Let f be a convex L-smooth function ($L > 0$). Nestrov's updates for suitable choices of step size α_k using gradient-based descent optimization yields [4]:*

$$f(x_k) - f(x^\star) \leq \frac{2L}{(k+1)^2}\|x_0 - x^\star\|_2^2$$

This same idea was explored for proximal algorithms applied to the BPDN problem. The papers [5, 6] discussed in this project have this goal of accelerating the regular proximal algorithm.

1.3 Proximal Operators

Proximal Algorithms are a type of optimization algorithms that excel at solving non-smooth and constrained convex optimization problems. The fundamental mechanism behind these algorithms is shown in the proximal operator $\mathbf{prox}_f$. The proximal operator maps a point $v \in \mathcal{R}^n$ to $x \in \mathcal{R}^n$ based on the function value $f(x)$ and the proximity between v and x.

Let $f : \mathcal{R}^n \to \mathcal{R} \cup \{+\infty\}$ be a closed convex function with domain

$$\mathbf{dom} f = \{x \in \mathbf{dom} f \mid f(x) < +\infty\}$$

The function f will have the extended value of $+\infty$ for all $x \notin \mathbf{dom} f$. The proximal operator $\mathbf{prox}_f$ is then given by:

$$\mathbf{prox}_{\lambda f}(v) = \arg\min_x (f(x) + \frac{1}{2\lambda}\|x - v\|_2^2)$$

where λ is a parameter that determines the weight of the proximity term.

The proximal operators mapping can be described in two cases. In case one, $v \notin \mathbf{dom} f$. Here, the extended value of f will ensure the mapped point $x \in \mathbf{dom} f$. In the other case, $v \in \mathbf{dom} f$. Now the proximal operator will map x such that $f(x) \leq f(v)$.

It should also be noted that when $f(x) = f(v)$, or the operator returns a fixed point, then x (or v) minimizes f. In other words, x^* is a minimizer of f if and only if:

$$x^* = \mathbf{prox}_{\lambda f}(x^*)$$

This property of proximal operators is very useful in algorithm development.

Another property that will be useful is the evaluation of a proximal operator $\mathbf{prox}_{\lambda f}$ when f is the l_1 norm, $f = \|\cdot\|_1$. The solution to this problem is given by the soft thresholding operator, which is given by

$$\mathbf{prox}_{\lambda f}(v) = (v - \lambda)_+ - (-v - \lambda)_+.$$

This will be evaluated to zero when $|v| \leq \lambda$.

1.4 Proximal Gradient Descent

As mentioned in the previous section, the fixed point property of the proximal operator allows for simple algorithm development. This fact along with the guarantee that the proximal operator given point v will return point x where $f(x) \leq f(v)$ allows for the creation of simple algorithms.

First, the proximal minimization algorithm simply finds the next point by taking the proximal operator of the previous. This algorithm is defined as:

$$x_{k+1} := \mathbf{prox}_{\lambda f}(x_k)$$

This algorithm is guaranteed to converge given that f has a minimum and $\lambda > 0$. Although this will converge to a solution, it is computationally expensive and difficult to minimize. Proximal methods become much easier when they take the form

$$\text{minimize } f(x) = \text{minimize } g(x) + h(x)$$

where g and h are closed convex functions and g is differentiable. Such an algorithm will aim to minimize g with constraints encoded by h. Once such method with this form is the proximal gradient method. This method first performs gradient descent on g, then takes the proximal operator of h. The algorithm can be thought of as first minimizing g and then moving the minimized point closer to the behavior described in h. This cycle ensures a minimized solution that adheres to the nature of h. Choosing h as I_C with constraints set C results in the projection onto the constraints set. A norm l_1 can also be chosen for h, resulting in a sparse solution, for example. The proximal gradient methods iterations are defined by

$$x_{k+1} := \mathbf{prox}_{\lambda_k h}(x_k - \lambda_k \nabla g(x_k))$$

where λ^k is found using a line search. The algorithm may be stopped when $f(x_k) - f(x_{k+1}) \leq \epsilon$, where ϵ is a sufficiently small value.

Theorem 4 (Proximal Convergence) *Consider $f(x) = g(x) + h(x)$ where g is L-smooth and convex and h is convex. Using fixed step size $\alpha_k = \frac{1}{L}$, and denoting $x^\star$ as the minimizer of f, for the proximal gradient*

descent algorithm we have:

$$f(x_k) - f(x^\star) \leq \frac{L}{2k}\|x_0 - x^\star\|_2^2$$

There exists an accelerated proximal gradient method that makes use of an extrapolation parameter ω paired with a momentum term. A basic implementation of this idea uses a similar framework as the proximal gradient method,

$$x^{k+1} := \mathbf{prox}_{\lambda^k h}(y^{k+1} - \lambda^k \nabla g(y^{k+1}))$$

where y is determined by

$$y^{k+1} := x^k + \omega^k(x^k - x^{k-1})$$

The parameter $\omega^k \in [0, 1)$ is chosen specifically to achieve the accelerated performance. Note that when $\omega^k = 0$, the algorithm is equivalent to the proximal gradient method. Another important aspect of this algorithm is the use of $(x^k - x^{k-1})$. This results in a vector pointing near the solution x^*, while taking the reasonable assumption that the general descent direction is correct. This results in accelerated speed in convergence and number of iterations as each iteration possesses momentum towards the solution.

The step size λ is still chosen/ determined in the same manner as in proximal gradient method. When ω^k is chosen correctly, this method achieves a faster worst case convergence rate that that of the proximal gradient method.

2 Basis Pursuit Denoising Problem

2.1 Problem Description

The basis pursuit denoising problem (BPDN) is an optimization problem that takes the form

$$\min_x \tfrac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1$$

where b is a noisy input image, A is a randomly generated observation matrix, and x is the recovered signal. The optimization aims to balance minimizing the residual ($b = Ax$) while maintaining a sparse solution ($\|x\|_1$). This problem arises from the similar basis pursuit problem, which takes the form

$$\min_x \|x\|_1 \text{ subject to } y = Ax.$$

Loosening the equality constraint recreates BPDN and allows the optimization problem to better handle noise and produce improved image denoising results.

3 Proximal Algorithms to BPDN

Implementing ISTA, FISTA, and SpaRSA for BPDN will be the focus of this paper. A brief description of each algorithm and a problem-specific implementation and structure is given.

3.1 ISTA

The Iterative Shrinking Thresholding Algorithm employs proximal gradient descent to solve the lasso problem. The lasso problem consists of a smooth function alongside a penalty term who's function is to encourage sparsity in the result. The lasso problem can be stated as:

$$\text{minimize } \tfrac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1$$

where the parameter λ determines the sparsity level of the solution. Implementing a solution using proximal gradient method, we can split the lasso problem into functions $g(x)$ and $h(x)$:

$$g(x) = \tfrac{1}{2}\|Ax-b\|_2^2, \quad h(x) = \lambda\|x\|_1$$

with gradient

$$\nabla f(x) = A^T(Ax-b)$$

The proximal operator of the l_1 norm is given as

$$\mathbf{prox}_h(x) = \text{sign}(x)\max\{|x|-\lambda, 0\}$$

This is equivalent to the soft thresholding operator. In its other form

$$\mathbf{prox}_{\lambda g}(x) = \begin{cases} v_i - \lambda & v_i \geq \lambda \\ 0 & |v_i| \leq \lambda \\ v_i + \lambda & v_i \leq -\lambda, \end{cases}$$

it is easier to see what this operator is doing. Much like the name of the algorithm, this is a shrinking and thresholding operator. All values with magnitude less than γ are set to zero, and all other values are shrunk by the same λ. This behavior helps in minimizing $g(x) = \lambda\|x\|_1$ and creating a sparse solution.

Finally, the algorithm is defined as

$$x^{k+1} := \mathbf{prox}_{\lambda h}(x^k + t^k \nabla g(x^k))$$

where step size t^k is found using line search. In its most basic form, ISTA performs two fundamental tasks. First, a gradient descent is performed on some x^k with step size λ. Then, the proximal operator (specifically the shrinking/ thresholding operator) is applied. The result of this operator is the next point x^{k+1}. This repeats until a suitable x is found.

An ISTA algorithm can now be devised for basis pursuit. The algorithm for BPDN can be simply built upon the basic iterative shrinking method given by

$$x^{k+1} = \mathbf{prox}_{\lambda h}(x^k - 2t^k A^T(Ax^k - b))$$

with step size t and the proximal operator equivalent to the soft thresholding operator. This solution will be defined as p_L where L is a Lipschitz constant of ∇f. Now, the algorithm is defined.

Consider $f(x) = g(x) + h(x)$ where g is L-smooth and convex and h is convex.

Algorithm 1: ISTA Proximal Gradient Descent Algorithm

given $x_0 \in \mathbb{R}^n$;
repeat
  $x_k = \mathbf{prox}_{\frac{1}{L}h}(x_k - \frac{1}{L}\nabla g(x_k))$;
until *stopping criterion is satisfied*;

3.2 FISTA

The Fast Iterative Shrinking Thresholding Algorithm is similar to ISTA, but instead of using the proximal gradient method for the framework of the algorithm, the accelerated proximal gradient method is used. This results in an algorithm defined by

$$x_{k+1} := \mathbf{prox}_{\gamma\alpha_k h}(y_{k+1} + \alpha_k \nabla g(x_k))$$

FISTA chooses ω^k to be

$$\frac{t_k - 1}{t_{k+1}}$$

with

$$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}.$$

resulting in iterations

$$y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}).$$

It can be clearly seen that when $x^k \approx x^{k+1}$ the iterations reduce to those of proximal gradient method (ISTA). However, the speed increase of this algorithm arises from the momentum provided when $x^k > x^{k+1}$. As mentioned while discussing accelerated gradient method, the term $(x^k - x^{k-1})$ is a vector pointing in the direction currently chosen by the algorithm. A scaled version of this vector is added to the current iterate x^k, pushing x^k further in the chosen descent direction. It should also be noted that ω^k starts at zero and approaches 1 as t $\to \infty$. In other words, FISTA gains confidence in the chosen direction with each passing iteration. The slowing of the algorithm is left to the momentum vector approaching zero.

The convergence rate of the FISTA is proven to be similar to the convergence rate of accelerated gradient descent as shown in the next theorem.

Theorem 5 (FISTA Convergence) *Consider $f(x) = g(x) + h(x)$ where g is L-smooth and convex and h is convex. Using fixed step size $\alpha_k = \frac{1}{L}$, and denoting $x^\star$ as the minimizer of f:*

$$f(x_k) - f(x^\star) \leq \frac{2L}{(k+1)^2}\|x_0 - x^\star\|_2^2$$

The full algorithm takes the form:

Consider $f(x) = g(x) + h(x)$ where g is L-smooth and convex and h is convex.

Algorithm 2: FISTA Proximal Gradient Descent Algorithm

given $y_1 = x_0 \in \mathbb{R}^n, t_1 = 1$;
repeat
  $x_k = \mathbf{prox}_{\frac{1}{L}h}(y_k - \frac{1}{L}\nabla g(y_k))$;
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$;
  $y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1})$;
until *stopping criterion is satisfied*;

3.3 SpaRSA

The SpaRSA algorithm aims to solve the lasso problem described above with increased convergence time while maintaining a general solution. The implementation of SpaRSA differs in implementation through the α term. In SpaRSA, α is an iteration based parameter. The algorithm can then be defined as

$$x_{k+1} = \arg\min_x \tfrac{1}{2}\|x - u_k\|_2^2 + \tfrac{\tau}{\alpha'_k}\|x\|_1.$$

Where τ is the sparsity level. Written in algorithmic form we have

$$x_{k+1} := \mathbf{prox}_{\alpha'_k h}(x_k - \nabla g(x_k))$$

where the l_1 proximal operator is used. The parameter α is determined by solving a smaller optimization problem each iteration. This optimization problem is a variant of the Barzilai-Borwein approach, which, similar to the Newton method sets α_k in relation to the Hessian at the latest point [7]. α_k is defined as

$$\alpha'_k = \arg\min_\alpha \|\alpha s_k - r_k\|_2^2$$

where

$$s_k = x_k - x_{k-1},$$
$$r_k = \nabla f(x_k) - \nabla f(x_{k-1})$$

α is bounded based on the implementation. Computing a new α each iteration has shown an increased rate of convergence in implementation.

Consider $f(x) = g(x) + h(x)$ where g is L-smooth and convex and h is convex.

Algorithm 3: SpaRSA Proximal Gradient Descent

Given
$\eta > 1, [0 < \alpha'_{min} < \alpha'_{max}], x_0 \in \mathbb{R}^n$
repeat
  $\alpha'_k \in [\alpha'_{min}, \alpha'_{max}]$; (Safeguarded Barzilai-Borwein)
  repeat
    $x_{k+1} = \mathbf{prox}_{\frac{1}{\alpha'_k} h}(x_k - \frac{1}{\alpha'_k}\nabla g(x_k))$;
    $\alpha'_{k+1} \leftarrow \eta\alpha'_k$;
  until x_{k+1} *satisfies an acceptance criterion*;
  $k \leftarrow k + 1$;
until *stopping criterion is satisfied*;

SpaRSA is the most computationally complex of the described algorithms due to the mini convex problem that needs to be solved in calculating α'_k.

4 Experimental Results

4.1 Experimental Setup

Next, experimentation was completed to show the performance differences between ISTA, FISTA, and SpaRSA. The basis pursuit denoising problem was implemented with the famous cameraman test image. This 256×256 image was blurred with a Gaussian kernel, as shown in Figure 1.

The algorithms are then implemented with y containing the blurred image and A composed of a Gaussian blur filter and wavelet transform. For all algorithms the regularization parameter is set to $\lambda = 0.035$, although there exists a large range of acceptable λ values.

4.2 Results

The cameraman test image was first blurred using a Gaussian blur using MATLAB function 'imfilter', as seen in figure 1. Each algorithm was run until reaching a specified function value of $1.2e5$. Results in figure 2 show the speed of each algorithm. As expected, ISTA was the slowest algorithm, taking around 4-5 times the number of iterations as FISTA and SpaRSA. These faster algorithms were close in terms of number of iterations, with SpaRSA finishing in 20 iterations compared to FISTA's 28.

The speed of FISTA and SpaRSA is also displayed in figure 4. SpaRSA is able to achieve the same reduction in 5 iterations as ISTA could in 10. This gap only increases with the number of iterations. with ISTA needing close to 100 iterations to gain the reduction SpaRSA achieved in 20.

Our results matched what was expected out of the three algorithms. The more computationally intensive SpaRSA converged in the least number of iterations, and FISTA outperformed its slower counterpart. The low cost increase of FISTA should also be noted. FISTA requires the same number of gradient evaluations as ISTA, with the additions of some negligible multiplications. The relative increase in computation compared to the increase in speed is remarkably small.

Figure 1: Original and Blurred Cameraman Image.

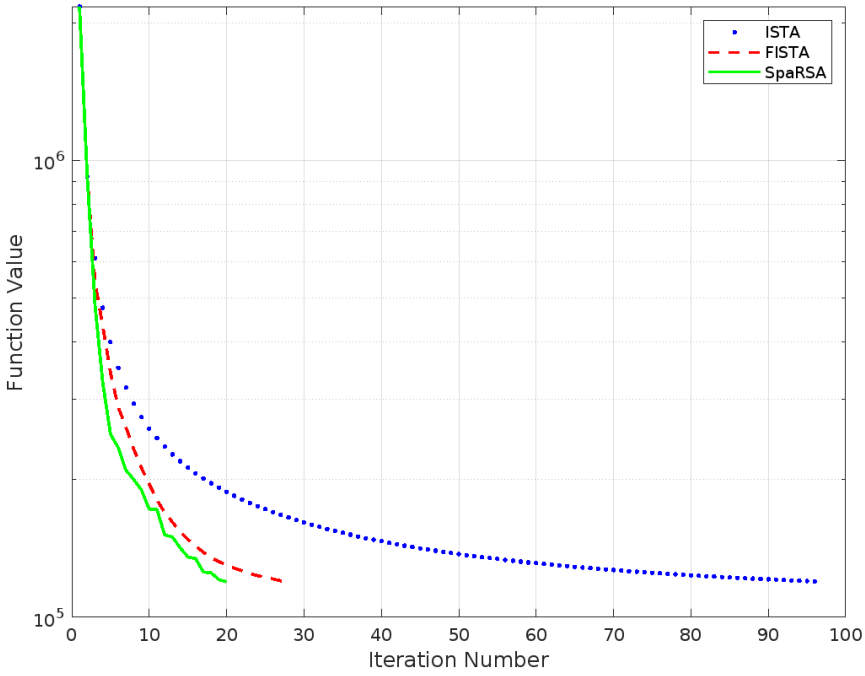

Figure 2: Function Value vs Iteration Number of ISTA, FISTA, and SpaRSA

Figure 3: Recovered Images from each Algorithm.

5 Conclusion

We have discussed the proximal methods as a proxy of gradient based methods to the non-smooth objective function. We focused on the BPDN problem and two accelerated proximal methods proposed to enhance convergence speed of solving it. Our numerical experiments with explained FISTA [5] and SpaRSA [6] algorithms verify the observations in the literature [8, 9] that the latter in practice converges faster than the former. The fundamental algorithms used for ISTA and SpaRSA were given by each respective publication,

Figure 4: Outputs of ISTA, FISTA, and SpaRSA at selected iterations.

and the FISTA algorithm consisted of minor changes to ISTA. It should also be noted that the convergence analysis of SpaRSA is not presented, unlike that of FISTA.

References

[1] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004. 1

[2] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014. 1

[3] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, no. 01, 2003. 1

[4] J. Romberg and M. Davenport, "Lecture notes of ece6270," *Georgia Institute of Technology, Fall Semester*, 2022. 2

[5] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009. 2, 9

[6] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on signal processing*, vol. 57, no. 7, pp. 2479–2493, 2009. 2, 9

[7] W. Yin, "Math 164: Optimization barzilai-borwein method," *University of California, Los Angeles, Spring Semester*, 2015. 7

[8] S. Becker, J. Bobin, and E. J. Candès, "Nesta: A fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011. 9

[9] W. W. Hager, D. T. Phan, and H. Zhang, "Gradient-based methods for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 146–165, 2011. 9