Compressed Sensing - Reading Course Project

Arun Suresh (under the advisement of Dr. Xiaojing Ye)

November 2020

1 Introduction

Given a sparse signal $x \in \mathbb{R}^n$ with p non-zero entries (p much lesser than n), it is computationally expensive to sample, store and transmit them, especially when n is very large. The quest to solve this problem effectively, leads us to consider a different problem where instead of sampling x directly, we transform it through a pre-defined matrix $A \in \mathbb{R}^{m \times n}$, with m significantly less than n. So given $Ax = b$, it is much easier to sample, store and transmit b because of low dimensionality. However, the problem becomes, given such a b and A, how do we get back x? Under some relaxation, this leads us to solve the following problem

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

where $||\cdot||_1$ is the ℓ_1 norm of x and $||\cdot||$ is the ℓ_2 norm of x. In this report, the minimization problem (1) is solved using the regular proximal gradient method (RPG) and accelerated proximal gradient method.

2 Regular proximal Gradient method

Following the notations outlined in the main reading, we let $\phi(x) = ||x||_1$ and $f(x) = (\lambda/2)||Ax - b||$. Notice that we have

$$\text{prox}_{\tau\phi}(z) = \text{argmin}_x \left\{ \phi(x) + \frac{1}{2\tau}||x - z||^2 \right\} \tag{2}$$

Since $\phi(x) = ||x||_1 = \sum_i |x_i|$, we can in fact obtain a closed form for $\text{prox}_{\tau\phi}(z)$. The derivation is presented below. In order to obtain a closed form for $\text{prox}_{\tau\phi}$, let us start with its 1 dimensional analog, namely $\text{prox}_{\tau|\cdot|}$. So, notice that

$$\text{prox}_{\tau|\cdot|}(z) = \text{argmin}_x \left\{ |x| + \frac{1}{2\tau}|x - z|^2 \right\} \tag{3}$$

It is straightforward to see that

$$|x| + \frac{1}{2\tau}|x - z|^2 = \begin{cases} x + \frac{1}{2\tau}(x - z)^2 & x > 0 \\ -x + \frac{1}{2\tau}(x - z)^2 & x < 0 \\ \frac{1}{2\tau}z^2 & x = 0 \end{cases} \tag{4}$$

Notice that splitting it into these three cases avoids differentiability issues locally when $x > 0$ or $x < 0$. Differentiating the above function and setting it equal to 0, we see that in the first case, we have $\hat{x} = z - t$, which is true if and only if $z \geq \tau$, and similarly, in the second case $\hat{x} = z + \tau$ which is true if and only if $z \leq -\tau$. Finally when $|z| < \tau$, we see that both cases lead to a contradiction. This means the function above can not attain a minimum at a positive or negative point. Moreover, upon expanding and simplifying, it is clear that both $x + \frac{1}{2\tau}(x - z)^2$ and $-x + \frac{1}{2\tau}(x - z)^2$ converge down to $\frac{1}{2\tau}z^2$. Therefore in the third case, $\hat{x} = 0$. This gives a closed form of (4) given by

$$\text{prox}_{\tau|\cdot|}(z) = \begin{cases} z - \tau & z \geq \tau \\ z + \tau & z \leq -\tau \\ 0 & |z| < \tau \end{cases} \tag{5}$$

Generalizing to $\mathbb{R}^n$ is now straightforward, if we apply (5) to each component. Notice that this only works because our original function is convex.

3 Accelerated proximal Gradient method

Now, we consider the accelerated proximal gradient method. Considering (7) (and the preceeding equation) in [3], the obvious choice of $h(x)$ is

$$h(x) = \frac{1}{2}||x||^2 \tag{6}$$

Notice that this gives the following proximity function

$$D(x,y) = \frac{1}{2}||x||^2 - \frac{1}{2}||y^2|| - \langle y, x-y\rangle \tag{7}$$

which easily satisfies (7)[†]. Similarly, the obvious choice of $P(x)$ is $||x||_1$ and $f(x) = (\lambda/2)||Ax-b||^2$. Since we are working in the metric space $\mathbb{R}^n$, which is convex (trivially), closed under the usual topology (once again, trivially), and $\text{dom}(P) = \mathbb{R}^n$, we can simply let $X_k = \mathcal{E} = \mathbb{R}^n$ for all k. The non-trivial part of implementing the algorithm now becomes the computation of

$$z_{k+1} = \text{argmin}_{x\in\mathbb{R}^n}\{\ell_f(x;y_k) + \theta_k LD(x;z_k)\} \tag{8}$$

Once again, as in the case of the regular proximal gradient method, we hope to find a closed form for computing z_{k+1}. To this end, let us investigate $l_f + \theta_k LD$ and study its 1 dimensional analog. It is straightforward to see that

$$\ell_f + \theta_k LD = ||x||_1 + \frac{\lambda}{2}||Ay-b||^2 + \frac{\lambda}{2}\langle \nabla f(y), x-y\rangle + \frac{\theta_k L}{2}||x-z_k||^2 \tag{9}$$

which we need to minimize. The one dimensional analog of (9) is a function of the form

$$\tilde{f}(x) = |x| + px + \frac{q}{2}|x-z|^2 + r \tag{10}$$

which can then be written as

$$\tilde{f}(x) = \begin{cases} x + px + \frac{q}{2}(x-z)^2 + r & x > 0 \\ -x + px + \frac{q}{2}(x-z)^2 + r & x < 0 \\ qz^2 + r & x = 0 \end{cases} \tag{11}$$

Performing a similar analysis as done in the regular proximal gradient method, we have a closed form for the argmin of $\tilde{f}$ given by

$$\hat{x} = \begin{cases} z + \frac{1-p}{q} & z < \frac{p-1}{q} \\ z - \frac{1+p}{q} & z > \frac{p+1}{q} \\ 0 & otherwise \end{cases} \tag{12}$$

Generalizing this into $\mathbb{R}^n$, once again, since our original function $\ell_f + \theta_k LD$ is clearly convex, we can simply apply (12) to each component of our z and y_k

[†]Notice that upon simplifying, we have $D(x,y) = \frac{1}{2}||x-y||^2$ since $\langle y, x-y\rangle = \langle y, x\rangle - ||y||^2$.

vectors. The natural n dimensional vector analog of p, q and r are simply

$$\begin{aligned} p &= \frac{\lambda}{2}(\nabla f(y_k)) = \frac{\lambda}{2}(A^T(Ay_k - b)) \\ q &= \theta_k L \\ r &= \frac{\lambda}{2} b^T(Ay_k - b) \end{aligned} \tag{13}$$

With this in place, it is now straightforward to implement algorithm 1 in [3].

4 Results

4.1 Sparse $\hat{x}$

Notice that when $x = \hat{x}$, we have $\phi(\hat{x}) + f(\hat{x}) = \phi(\hat{x}) = ||x||_1$. Suppose there was a $y \in \mathbb{R}^n$ with $\phi(y) + f(y) < ||\hat{x}||_1$, then we have that there is some m such that $||\hat{x}||_1 = m||y||_1$. This gives,

$$\frac{1}{m}||\hat{x}||_1 + \frac{\lambda}{2}||Ay - b||^2 \leq ||\hat{x}||_1$$

Now the choice of λ is clearly very important to the accuracy of these search models. Assume $\bar{x} = ay$ for some $a \geq 1$. Then

$$\|y\|_1 + \frac{\lambda}{2}\|Ay - b\|^2 = \frac{1}{a}\|\bar{x}\|_1 + \frac{\lambda}{2}(1 - \frac{1}{a})^2\|b\|^2. \tag{14}$$

Thus if $f_\phi(y) \leq f_\phi(\bar{x})$ from (14) we would have

$$\frac{\lambda}{2}(1 - \frac{1}{a})\|A\bar{x}\|^2 \leq \|\bar{x}\|_1. \tag{15}$$

From (15) we see that with proper A and λ large enough we can have strong confidence that $\bar{x}$ is indeed a solution to the main problem. So, with an appropriate choice of λ, we can be certain that $\hat{x}$ is a solution to our minimization problem in a neighborhood of $\hat{x}$. Since $\phi + f$ is convex, we know for a fact that there is only one such solution.

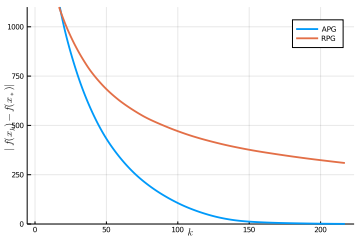
The regular proximal gradient and the accelerated proximal gradient algorithms were implimented to a problem with $n = 1000$, $m = 100$ and $p = 10$. The $m \times n$ matrix A was pre-defined with entries generated from a standard normal distribution and was normalized so that each column had ℓ_2 norm equal to 1. A random n dimensional vector $\hat{x}$ with p randomly initialized non-zero integer entries. In what follows, we summarize the results obtained using both algorithms. The user chosen parameter λ was set equal to 2 and the tolerance was chosen to be $\epsilon = 10^{-3}$. The Lipschitz constant was computed to be $\lambda||A^TA||_2$, and the step size τ was set randomly initialized to a number between 0 and $2/L$. After generating the matrix A and $\hat{x}$, two separate computations were performed.

The first computation, inspired from N. Gaby, held $\theta_k = 1/L$ constant. It can be checked that this satisfies (15) in [3] quite easily. The plots obtained in this run are presented below (Figure 1). It is clear that APG performs much better than RPG. It is also clear from Figure 2, that RPG is $O(1/k)$ while APG is $O(1/\sqrt{k})$, since there exists $c_1 = 10^5$ and $c_2 = 10^3$ and x_0, x_1 such that the RPG plot is under $10^5/k$ for all $k > x_0$ and the APG plot is under $10^3/\sqrt{k}$ for all $k > x_1$[‡]. Figure 4 (a) further illustrates this by producing a semilog plot and

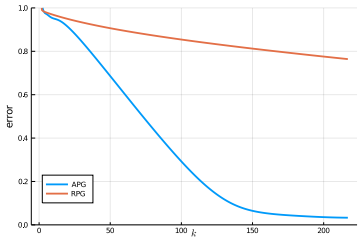
[‡]There seems to be a mistype in figure 2 (b), as it should read "APG" in the legend.

it is clear that both methods approximate straight lines with slope proportional to ϵ and $\sqrt{\epsilon}$.

Moreover, one can conclude from Figure 1 (b) that the error goes to zero much faster in APG than in RPG. In fact, it is noteworthy that it took more 3000 iterations for the error from the RPG method to get below 0.1, while it simply took 135 iterations for the error obtained from APG to get there.

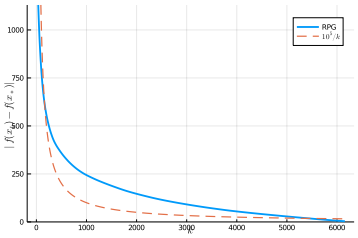


(a) $y = |f(x_k) - f(\hat{x})|$

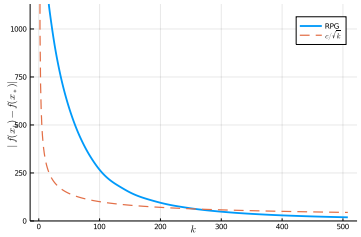


(b) $y = \frac{||x_k - \hat{x}||}{||\hat{x}||}$

Figure 1: Function plots and error plots for constant $\theta_k = 1/L$



(a) k vs. RPG, $10^5/k$



(b) k vs. APG, $10^3/\sqrt{k}$

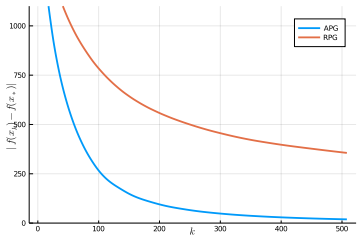
Figure 2: Comparison of methods alongside $10^5/k$ and $10^3/\sqrt{k}$ with $\theta_k = 1/L$

The second computation, followed (20) from [3] for the choice of θ_k. This meant that, θ_0 was initialized randomly, and θ_k was recursively obtained using

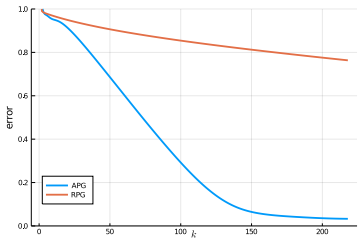
$$\theta_{k+1} = \frac{\sqrt{\theta_k^4 - 4\theta_k^2} - \theta_k^2}{2}$$

The plots obtained in this run are presented below in figure 3. It is clear that APG once again performs much better than RPG. It has been verified in a fashion similar to figure 2, that these methods are also $O(1/k)$ and $O(1/\sqrt{k})$

respectively. The plots are not reproduced below to reduce redundancy. Figure 4 (b) further produces a semilog plot and it is clear that both methods approximate straight lines with slope proportional to ϵ and $\sqrt{\epsilon}$.



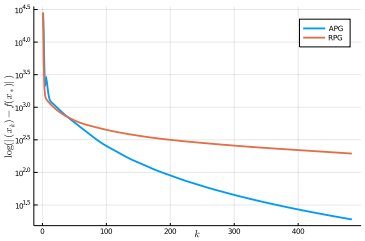
(a) $y = |f(x_k) - f(\hat{x})|$



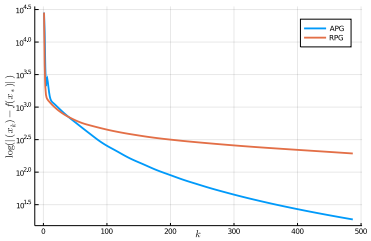
(b) $y = \frac{||x_k - \hat{x}||}{||\hat{x}||}$

Figure 3: Function plots and error plots for θ_k following (20) in [3]

It is clear that $\hat{x}$ is the solution to the minimization problem. It is noteworthy that when θ_k was held constant, the convergence was observed much faster than for a varying θ_k dictated by (20) in [3]. This is something that we expect because, in the latter case $\theta_k \to 0$ (much faster than any other choice in fact), which reduces the influence of the quadratic factor altogether and we are left with only a linear approximation of f^P, which, like we would expect doesn't do the best job in faithfully approximating f^P.



(a) $y = \log(|f(x_k) - f(\hat{x})|)$, $\theta_k = 1/L$



(b) $y = \log(|f(x_k) - f(\hat{x})|)$, varying θ_k

Figure 4: semilog plots for both cases

Moreover the choice of λ is also crucial in convergence, as for smaller values of λ – much slower, or in some cases, no convergence was observed. This agrees with our computations from the start of this section. In order to reduce redundancy, those plots are not reproduced in this report. Finally, in order to quantitatively describe the performance of both the models, the average run time of the core algorithm (after initialization, before plotting) over 50 runs, in

both cases are tabulated below

Algorithm	θ_k variable	θ_k constant
RPG	0.993444s	0.993444s
APG	0.202501s	0.121701s

and it is clear that on average, APG runs 4 times faster than RPG (and that as expected, holding θ_k constant further cuts down some computation time).

4.2 Non-sparse $\hat{x}$ (Variation of CS)

So far, we have implemented the compressed sensing algorithm for sparse signals with $p = 10$ non-zero entries with p significantly smaller than $n = 1000$. However, it is clear that a lot of real life signals are non-sparse, and therefore there is a need for adapting CS to non-sparse signals. Wavelet analysis has been an area of active study and research since the 1980s and has been used for various applications. The discrete wavelet transform is a very powerful tool for image and signal analysis, but our main focus is going to revolve around the fact that, with a realistic signal and a corresponding choice of basis wavelet, the discrete wavelet transform of the signal will often be a sparse vector in the wavelet domain. We will then use the sparse vector as our ground truth $\hat{x}$. So, we set up our APG/RPG problem around this $\hat{x}$, converge to our solution x^* and finally apply the inverse transform to reconstruct our original signal. In order to exemplify this, we will consider our input signal to be the Heavisine function H (stretched by a factor of 1024) given by

$$H(x) = 4\sin(4\pi x) - \text{sign}(x - 307.2) - \text{sign}(x - 737.28)$$

and infuse it with some Gaussian noise. The resulting input signal looks like –

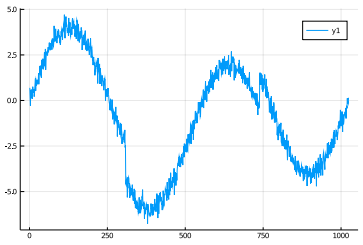


Figure 5: Input Signal

We will now take this input signal, and perform a single layer wavelet transform with the basis given by the sym8 wavelet. Sym8 was chosen here since the heavisine function, even with its abrupt bumps, closely resembles a sine

function in nature. In particular, it is noteworthy that, the heavisine function is differentiable and smooth in all but exactly four points. Therefore, a basis with the mother wavelet and the scaling function both differentiable everywhere serves as a more robust choice.

A single layer discrete wavelet transform is then applied to the input signal with hard thresholding (with a threshold value of $\mu = 1.0$), and the resulting distribution of coefficients in the wavelet domain (as shown in figure 6) is considered to be our ground truth $\hat{x}$ for CS. It is to be noted that in order to apply the discrete wavelet transform, it was crucial that the number of data points is a power of 2. Therefore, n was redefined to be $2^{10} = 1024$. A hard thresholding

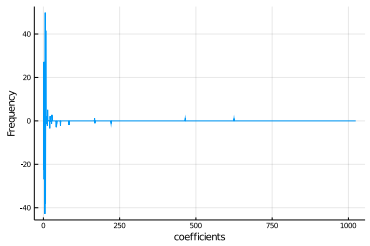


Figure 6: Transformed signal in wavelet Domain

was chosen here because the goal of the transform was to induce sparsity and not to reduce the magnitude of the resultant coefficients. To that end, notice that in the wavelet domain, the resultant distribution of coefficients is sparse, and is ideal for APG (or RPG) as in section 4.1. It is also noted in [4] that the sparsity of $\hat{x}$ can be further enhanced by transforming $\hat{x}$ via

$$W = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \frac{n-1}{n} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \frac{1}{n} \end{bmatrix}$$

However, for this toy example, we do not wish to add this further layer of complexity.

Upon obtaining the sparse $\hat{x}$, we had all the ingredients to perform an APG (or RPG which wasn't performed here to remove redundancies). The algorithm ran in 0.102457s under holding $\theta_k = 1/L$ constant. The resulting error plot is presented below (Figure 7).

It is noteworthy that the error decays to zero once again very fast, however, not as fast as observed in section 4.1, and this is due to the excessive noise that is present in the input signal.

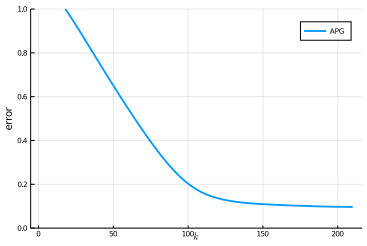


Figure 7: APG error plot

The inverse wavelet transform was then applied to x^* (the APG output), which then gave our reconstructed signal as presented below. Notice that we have

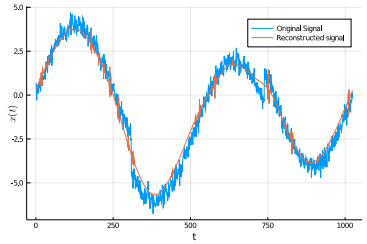


Figure 8: Original signal vs. Reconstructed signal

now not only sampled the original signal back to a good extent but also used the wavelet transform to denoise the signal in addition to the reconstruction. This is to be expected out of the (hard) thresholding condition that reduced most noisy coefficients to zero. Tweaking the threshold value from $\mu = 1.0$ to a smaller value will bring back the noise upon reconstruction. It is clear that the choice of μ depends on the application and the extent to which one would want to preserve the noisiness in a given signal.

[Ye: Very good!]

5 Acknowledgements

In the process of studying the accelerated proximal gradient method, and gathering inspiration for the proper writing of the the one dimensional analog of the problem - N. Gaby's very well written report was consulted. Suggestions and fruitful conversations with W. Phanthavong also aided the completion of this project.

6 Further Directions

Performing non-sparse CS for various other noisy signals under different basis wavelets is a promising endeavor. It is clear that in our non-sparse computation, the error did not quite go to zero as fast as our expectations. Optimizing our threshold constant μ depending on the inherent noise present in the input signal will definitely help the error reduce to zero much faster. One may also think about denoising the signal before doing CS as a way to optimize.

7 Code

The algorithms were implemented in the Julia programing language (Ver 1.4.2). The code for section 4.1 should be accessible from the `coding_project.jl` file. The code for section 4.2 should be accessible from the `variation_of_CS.jl` file.

[Ye: Good job! I didn't expect that you also completed the variation of CS using wavelet, which is very nice.

I gave similar final remark to Nathan which you may also consider: Now you can search "FISTA" on Google to find the paper by Beck and Teboulle. What you presented in this report is the essential main idea of that paper, except for the back-tracking part which is a standard technique (hence a minor contribution) in the optimization community (as in practice the Lipschitz constant L may be unknown or difficult to compute, and one needs to search for a proper step size without knowing L). Read that paper carefully, and learn the way they organize the content and present their theoretical and numerical results. This prepares you to convert your initial report (when we will consider new problem and propose new method) into a full research paper for publication in the future.

A small suggestion I mentioned above: use the loglog or semilogy plots which may help you to visualize the convergence rate, since you would observe "straight line" in your plot. If the convergence is sublinear, i.e., error $e_k := f(x_k) - f(x^*) = O(1/k^\alpha)$ versus iteration k, then $\log(e_k) \propto -\alpha \log k$, which means you'll see a line with slope $-\alpha$ in the loglog plot. Similarly, if the convergence is linear and you have $e_k = O(\theta^k)$ for some $\theta \in (0, 1)$, then you will see a straight line in the semilogy plot.]

References

[1] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52(2):489–509,2006

[2] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer,2004

[3] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2008.

[4] W. Ziran, W. Huachuang and Z. Jianlin. (2017). *Wavelet sparse transform optimization in image reconstruction based on compressed sensing*. IOP Conference Series: Earth and Environmental Science.