10/10

Reading Report of Beck et al.

(ECE 695, Reading Assignment 04, Spring 2018)

Overview of the Paper

The paper proposed an iterative method for solving inverse linear problems in general signal processing or image processing. The method is a close relative to the well studied iterative shrinkage-thresholding algorithms (ISTA) that has a simple framework which is easy to implement and suitable for solving large scale problems. The term 'shrinkage' is originated from the fact that when the regularization term of the objective function is a l_1, then the step inside ISTA can be simply reduced into a shrinkage operation. The proposed fast ISTA is similar to ISTA in structure, but introduces a new way to update the input to the shrinkage operator such that the convergence rate of the algorithm greatly improves compared to the ISTA algorithm. Also by experiments, the performance of the proposed algorithm seems to also surpass that of the original vanilla ISTA method.

Prior Work

The proposed FISTA algorithm is a direct descendant of ISTA, which can be used for solving exactly the same groups of linear inverse problems. Due to the slow convergence rate of ISTA, methods other than FISTA also attempted to improve the convergence rate of ISTA. TWIST is in the line of works that attempted this. It proposed an interesting two-step ISTA that is proven to converge to a minimizer of the original objective function under some assumptions about the data of the problem. Another work by Nesterov also proposed a multi-step ISTA that demonstrates a convergence rate the same as FISTA, but this method requires two projection-like steps different from FISTA that requires only one. When the regularization term is l_1, then there are a lot of competitive methods. Some notable ones are MOSEK, LARS, SEDUMI, PDCO, l1ls and SPGL. However, these methods are not as general as the proposed FISTA.

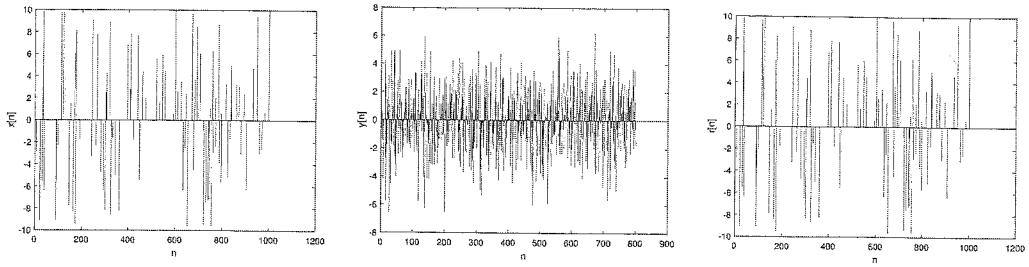
Key Ideas of the Paper

It seems there is only one main key point from this paper. The proposed method developed an update rule for the input to the minimization problem within each iteration of ISTA. Specifically the update rule is shown below:

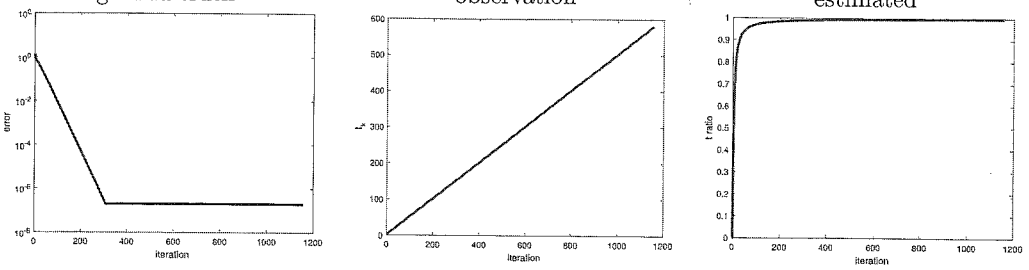
$$\begin{aligned} \boldsymbol{x}_k &= pL_k(\boldsymbol{y}_k) \\ t_{k+1} &= \frac{1+\sqrt{1+4t_k^2}}{2} \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + (\frac{t_k-1}{t_{k+1}})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) \end{aligned}$$

Other than this update rule, the rest of the proposed algorithm is exactly the same as the original ISTA algorithm. The beautiful part of this update rule is the t_k step. It seems t_k is only dependent on t_{k-1} but not $f(*)$ or $g(*)$, which means the sequence of t_k is only dependent on t_1 which is initialized as some input constant by the user. This sequence of number then just magically enables the proof and faster convergence rate of the algorithm automatically. This is almost jaw-dropping. Experimentally speaking, when $t_1 = 1$, t_k is just a sequence of increasing number. Which means the ratio $\frac{t_k-1}{t_{k+1}}$ is always smaller than 1. Other than the update step for t_k, the update rule for $\boldsymbol{y}_k$ is also interesting. The second term $(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$ is basically a vector pointing from the result of previous solution to the result of current solution. This vector is basically a vector of progression. If we assume the direction of progression is correct, which is probably true for any working iterative algorithm, then by adding that vector to $\boldsymbol{x}_k$ we simply get a more 'bold' or more 'progressive' solution $\boldsymbol{y}_k$ compared to the more conservative solution $\boldsymbol{x}_k$. This then makes sense if FISTA has a faster convergence rate than the original ISTA.

Very good!



ground truth observation estimated



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

Comments

I ran the code provided by researchers at UC Berkeley on a sparse signal reconstruction problem. The result of FISTA is rather accurate considering noise is present in this simulation. We can see that the error plot under semilog scale is a straight line, meaning the FISTA algorithm has a convergence rate of $\frac{1}{k^2}$ as reported by the paper. On the other hand, the t_k curve is basically a straight line. Again, it is very interesting that a sequence of t_k from a straight line can make the algorithm convergent and converge faster than ISTA. While, the ratio $\frac{t_k-1}{t_{k+1}}$ used for updating $\boldsymbol{y}_{k+1}$ is a rather steep curve that increases very fast at the early iterations of the algorithm.

Since the t_k curve approximates a straight line with slope being $\frac{1}{2}$, and we also know it passes point $(1,1)$ as $t_1 = 1$. Then we can approximate the sequence as $t_k = 1 + \frac{1}{2}k$. Then running FISTA with this approximation, I'm able to get an almost exactly the same convergence plot, and the final errors when running the two update schemes for running the same number of iterations are also basically the same with difference at around 1e-10. This means the performance of the algorithm is not sensitive to the t_k sequence, meaning an approximation is more than enough in practical use. However, the nice theorems and proofs in the paper are no longer valid. Then it is conclusive that the definition of t_k in the paper is more for proof purpose than actual usefulness. Changing the slope of the linear approximate curve from $\frac{1}{2}$ to $\frac{1}{3}$ also works and produce the same(to human eyes) convergence curve. Changing the offset of the curve from 1 to 2 works as well and not surprisingly generates the same convergence curve as well. Turning the linear function to a quadratic function $t_{k+1} = 2 + \frac{t_k^2}{3}$ also works and generates the same convergence curve. Finally using a linear decreasing function does not work at all. It seems as long as the t_k sequence is somewhat increasing the performance of FISTA would be the same. Although this fact hurts the importance of the update rule presented by the paper, it actually makes the algorithm quite robust.

Another thing worth trying is to replace the $\frac{t_k-1}{t_{k+1}}$ with some other curve and see the effect. I designed a function in matlab using cdf('Normal',k,0,10)-cdf('Normal',-k,0,10), which is the area under the Gaussian distribution within the bound $[x, -x]$, and the shape of this curve resembles the shape of $\frac{t_k-1}{t_{k+1}}$. Again using this ad-hoc function to generate the weight for $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ inside the update step for $\boldsymbol{y}_{k+1}$ also works and generate exactly the same convergence curve viewed by human eyes. However, the speed decreases a little bit due to the fact that cdf command in Matlab is slower than simple scalar division. However, this experiment, similarly to the experiments on t_k sequence, shows that the FISTA algorithm is not sensitive to the update rule. So I would guess the most significant contribution of the paper is the adoption of $\boldsymbol{x}_{k-1} - \boldsymbol{x}_k$ for updating $\boldsymbol{y}_{k+1}$.

Very nice and very unique finding!

Reading Report of a Fast Iterative Shrinkage-Thresholding Algorithm by Beck et. al.

(ECE 695, Reading Assignment IV, Spring 2018)

Overview of the Paper

This paper talks in detail about the Iterative Shrinkage-Thresholding Algorithm (ISTA) and its accelerated version Fast Iterative Shrinkage-Thresholding Algorithm (FISTA), both of which are methods to solve the l1 regularized least squares problems. The authors give a detailed analysis of the global convergence properties of both the ISTA and FISTA and a few examples are given to show the superior convergence of FISTA for applications such as Image De-blurring.

Prior Work

There are a wide plethora of algorithms that have been utilized based on different ideas to tackle the l_1 regularized QP's.Homotopy algorithms such as Least Angle Regression, Interior Point methods such as l_1-ls which utilize preconditioned CG to accelerate the convergence rate, IST(Iterative Shrinkage/Thresholding) formulated as an MM algorithms that takes advantage of the convexity of the problem, and greedy algorithms such as OMP.

Key Ideas of the Paper & Assumptions

The underlying principle of both FISTA and ISTA are the same. They compose the given objective function into 2 components $f(x) \rightarrow \|Ax-b\|_2^2$ and $g(x) \rightarrow \lambda\|x\|_1$.The method works If f(x) satisfies the property that it is continuously differentiable with Lipschitz continuous gradient and if the sub-differential of g(x) exists and can be easily computed. If this condition is met, we can compute a quadratic approximation of f(x) at each iteration that uses only the gradient information and the Lipschitz constant L(f) (= max eigenvalue of $\boldsymbol{A}^T\boldsymbol{A}$) and leave h(x) alone. The problem can be recast as shown in equation 2.3[1] and the solution of this equation is given by the proximal operator, which is a function of both h(x) and the parameter t. The proximal operator is nothing but a (λt) shifted version of the soft thresholding operator. This is what induces sparsity in our solution. The parameter t weighs the relative importance of the l1 term and gradient term (f(x)). As long as $t < 1/L(f)$, both methods converge. FISTA is a modification of ISTA in which we use Nesterov's acceleration scheme to speed up convergence (more details in the Comments section)

Comments

Image De-blurring Example- My first experiment was to use the Image Deblurring application to test the power of FISTA with respect to GPSR(Gradient Projection Sparse Reconstruction) that we studied in Reading 2. Both these methods are first-order methods in that they utilize the gradient information and objective function value to arrive at the solution.

Experimental Setup: I chose a grayscale image 512×512.The given image was blurred using a 2-D Gaussian kernel[2]with standard deviation = 4 in both the horizontal and vertical directions. For the FISTA implementation, the matrix $\boldsymbol{A}$ has been implemented as a fast operator, using the Discrete Cosine transform, assuming reflexive boundary conditions for the de-blurred image. I chose the same reflexive boundary conditions for both FISTA and GPSR. The FISTA code does not have a defined stopping criterion other than maximum iterations. For simplicity, I added a stopping criterion based on the relative error in objective function with respect to the previous iteration ($tol = 1e^{-6}$). The current FISTA code uses a fixed value of $t = 1/L(f)$ for the soft-thresholding step, where $L(f)$ = Lipschitz constant of the gradient of f as mentioned before. For GPSR, both the monotone and non-monotone version of the algorithm were tested for the given problem. To quickly compute the optimum regularization parameter λ, I ran the code for a range of values of λ using both cold start and warm-start. There was hardly any difference in the computation time using these approaches. I chose $\lambda = 1e^{-4}$ as the regularization parameter as it provided a good balance between the sparsity in the solution and the accuracy of the estimate of the true image. Note, FISTA solves

$\|Ax-b\|_2^2+\lambda\|Wx\|_1$ whereas GPSR solves $\frac{1}{2}\|Ax-b\|_2^2+\tau\|Wx\|_1$, thus the objective function value of FISTA was scaled by 0.5 and equivalently,$\tau = 0.5e^{-4}$ was chosen for GPSR. There were 2 sets of studies done, one with noiseless blurred image and the other with 1% random image noise added to the blurred image

Observations :Figure 1(c) and 1(d) show the de-blurred image obtained using FISTA and GPSR corresponding to the noiseless blurred image. While both algorithms give comparable rate of convergence (Figure 2) and the solution of both have nearly the same MSE (0.0012), the estimate from FISTA is more sparse than the estimate from GPSR in the wavelet domain. This observation is intuitive because at every iteration we compute the proximal gradient by using the soft thresholding operator on the gradient in FISTA. It was also observed that GPSR has a tendency to get stuck as we are close to the solution, this was observed in both the monotone and non-monotone versions of the algorithm and is represented by the long tail observed in Figure 2a). As the gradient approaches 0, the subsequent iterations move more slowly towards the solution in the case of GPSR.The non-monotone version of GPSR which uses the Barzilae-Bordwein step length has marginally better convergence than the monotone version. The non-monotonic behavior can be explained using the eigenvalues of $\boldsymbol{A^TA}$ as explained in Reading 3. A more interesting observation is made when noise is added to the measurement. With 1% added noise, GPSR outperforms FISTA, although both algorithms converge to nearly the same objective function value. The MSE wrt to the true image, is much lower for GPSR as compared to the MSE for FISTA. Although MSE as a metric is biased, visually it can clearly be seen that the FISTA solution is plagued by noise. This performance degrades further as more noise is added.

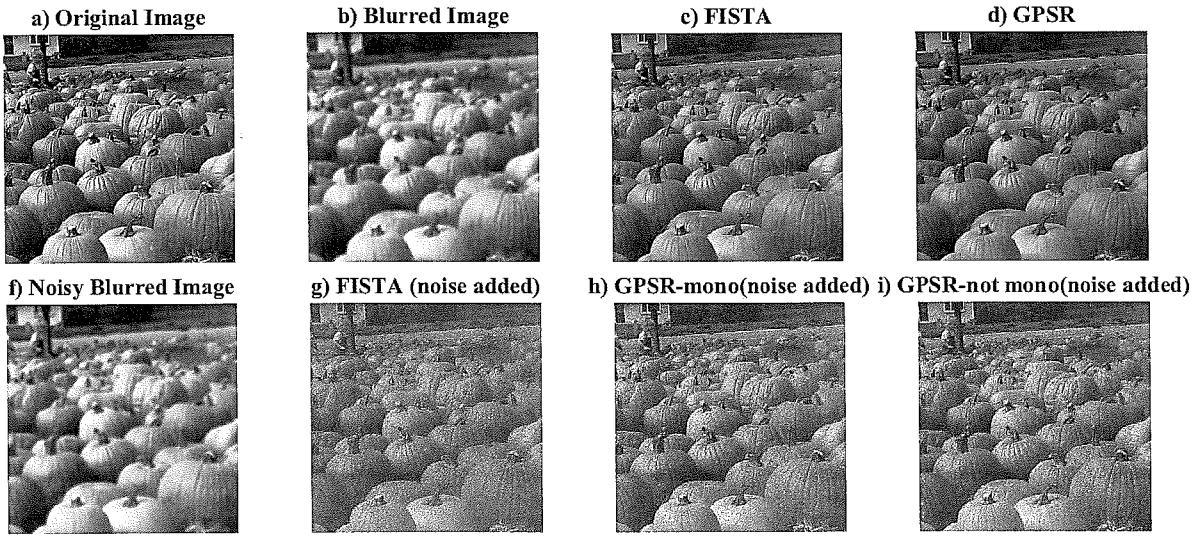


Figure 1: Outputs of FISTA & GPSR for the given problem

FISTA De-Mystified

The acceleration[3] in performance shown by FISTA as compared to ISTA can be attributed to 2 key equations $\boldsymbol{t_{k+1}} = \frac{1+\sqrt{1+4t_k^2}}{2}$ and $y_{k+1} = x_k + \frac{t_k-1}{t_{k+1}}(x_k - x_{k-1})$. We add a momentum term $\frac{t_k-1}{t_{k+1}}(x_k - x_{k-1})$ to the current iterate x_k and then evaluate the proximal gradient at this new intermediate point, shifting the iterate in the direction of largest momentum. Figure 2(c) is a plot of t_k vs k.In the initial iterations, we are accelerating the point much higher than the later iterations when we start approaching the true solution. This acceleration scheme was first proposed by Nesterov for the unconstrained minimization case and has been adapted to work well for FISTA. Now, this scheme works well if our measurements don't have noise. If we add noise, it is likely that the intermediate point y_k will start deviating from the true direction of largest momentum. In this scenario acceleration doesn't work very well. This phenomenon also explains why there was no benefit in using warm-starts. Warm-starts are useful under the assumption that the solution path is a continuous function of the regularization parameter, but in FISTA we compute the next iterate based on an intermediate point.

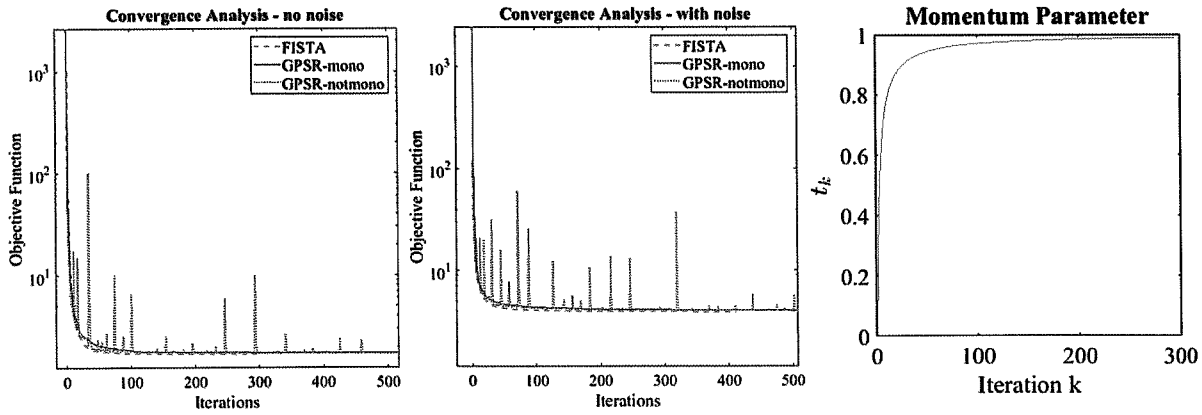


Figure 2: Outputs of FISTA & GPSR for the given problem

References

[1] Amir Beck and Marc 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] James G. Nagy Christian Hansen and Dianne P. O'Leary, "http://www.imm.dtu.dk/ pcha/hno/," *Deblurring Images ,Matrices, Spectra, and Filtering.*

[3] Ryan Tibshirani, "http://www.stat.cmu.edu/ ryantibs/convexopt/," *Convex Optimization: Fall 2016.*

10/10

Reading Report of Beck et al.

(ECE 695, Reading Assignment 4, Spring 2018)

Overview of the Paper

This paper introduces an algorithm of iterative shrinkage-thresholding algorithms (ISTA) class for solving linear inverse problems and the algorithm introduced is demonstrated in paper to be significantly faster (both theoretically and empirically) than all other ISTA's. The paper first introduces why we are interested in solving inverse linear problems and what common methods are for solving these problems. Then, the paper establishes the theoretical upper bound of rate of convergence for both traditional ISTA and their new FISTA. Finally, the paper shows some experimental results on applying FISTA to image deblurring problems.

Prior Work

Numerous gradient based algorithms have been developed for solving L_1 regularized least square problems (l1ls, GPSR, BPDN, ones that we have seen). ISTA is another popular method class due to its simplicity. However, a naive ISTA has a relatively low global convergence rate. Several alternative algorithms have been proposed to accelerate the convergence, for example two-step ISTA (TWIST).

Key Ideas of the Paper

The L_1 regularized least square objective function is just a particular case of the more general problem and the algorithm introduced in this paper does not only apply to the particular case of L_1 but also to the more general problem.

To establish an upper bound for global convergence rate of the algorithm, the authors made following assumptions about the objective function to be optimized. The function to be minimized is the sum of a smooth convex function f and a continuous but possibly non-smooth function g. The gradient of gradient function of f needs to be bounded (Lipschitz continuous).

The key idea for establishing upper bound for global convergence rate is to use a quadratic equation to approximate the original objective function. The authors show that for any vector $y \in \mathbb{R}^n$, the objective function F evaluated at the minimizer $p_L(y)$ for quadratic model is upper bounded by the quadratic approximation model evaluated at minimizer.

After a series of complicated algebraic derivation, the authors show that the convergence rate of a traditional ISTA is $O(\frac{1}{k})$ and their FISTA is $O(\frac{1}{k^2})$

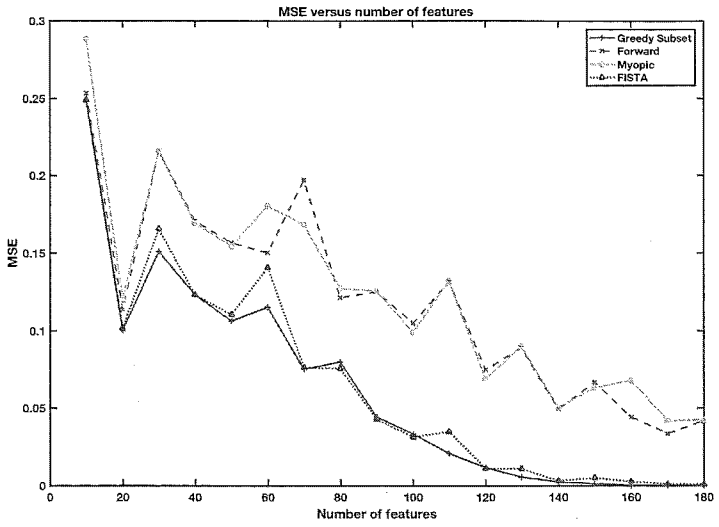
The key difference that drives FISTA significantly faster than ISTA is the input to the shrinkage operator (the minimizer for the quadratic approximation function). Instead of a simple previous point, FISTA uses a linear combination of the previous two points as the input to the shrinkage operator.

Comments

First, I would like to make a trivial comment that the theoretical convergence rates $O(\frac{1}{k})$ and $O(\frac{1}{k^2})$ themselves do not really show that FISTA is faster than ISTA in general cases in that both are just upper bounds. It is possible that the derivation used cannot establish a tighter upper bound for ISTA. To compare convergence rate, I suppose a θ bound or a Ω bound would be more convincing and informative. Of course, the specific experiments in this paper indeed demonstrate that FISTA is orders of magnitude faster than ISTA.

Second, I have to admit that after reading section 2 several times, I am still not able to understand how to compute this minimizer ($p_L(y)$) for quadratic approximation model. The tricky part is the non-differentiable $g(x)$ and the paper does not explicitly tell how to deal with it. Comparing the ISTA algorithm shown in section 3 and equation (1.4), I suppose $p_L(y)$ is that shrinkage operator in equation (1.4)?

In my experiment, I am mainly interested in applying this algorithm to feature selection problems in machine learning and I compare its results with some greedy algorithms (greedy subset, forward fitting, and myopic fitting). Default parameters were used for the MATLAB package of this algorithm. The data



is created by equation below. X_{nxd} is input data matrix where each row is a sample and each column is a feature. $\theta \in \mathbb{R}^d$ is the weight vector, and $\epsilon \in \mathbb{R}^n$ is some random noise. For feature selection, we try to pick F most significant features in X by minimizing the squared error between ground truth y and y calculated with only selected features. θ and selected feature set S are the outputs of each algorithm. F is a hyperparameter. In all experiments below, number of samples n was chosen to be 100, F was chosen to be half of the number of available features d.

$$y_{gt} = X\theta + \epsilon \tag{1}$$

$$y = X_S\theta_S, S \subset \{1, 2, ..., d\} \tag{2}$$

The most significant 5 features for each trial were shown below in the table. Feature index are sorted in descending significance order. As can be seen, when the number of features is relatively small compared to number of samples, although the ordering of selected features might be a little bit different, the selected set for FISTA is almost the same as greedy search algorithms. I also plotted the mean squared error between calculated y from selected features and ground truth y, as shown in figure above. It can be seen that the MSE for using features selected by FISTA is almost the same as the Greedy subset algorithm. It should be noted that forward fitting and myopic fitting can actually have MSE similar to greedy subset by refitting θ after feature selection. The MSE for all four algorithms is decreasing as number of features increases. This is easy to understand because the system gradually shifts toward an underdetermined linear system from an over-determined linear system. Based on the table and figure shown, I believe that FISTA can be used for feature selection purpose.

	Greedy Subset	Forward Fitting	Myopic Fitting	FISTA
d = 10	6,10,1,7,3	6,10,1,7,3	6,10,1,7,8	1,10,7,3,6
d = 10	4,10,6,7,3	4,10,6,7,3	4,10,6,7,3	4,6,7,10,2
d = 10	7,10,2,9,5	7,10,2,1,5	7,10,2,5,9	7,10,2,9,5
d = 40	20,2,26,4,39	20,2,26,25,6	20,2,26,4,39	20,39,2,26,16
d = 40	6,21,39,37,35	6,21,35,24,39	6,39,21,37,13	6,21,39,35,37
d = 80	9,13,50,30,70	9,13,50,30,70	70,50,30,13,11	62,23,28,45,58
d = 80	38,46,32,60,2	38,46,32,60,2	46,38,60,32,37	38,54,46,52,10
d = 120	76,22,75,113,50	76,22,75,113,50	22,76,75,113,50	76,119,23,99,81
d = 120	92,17,76,61,66	92,17,76,61,88	92,17,76,61,88	32,27,35,105,95
d = 180	61,111,162,107,2	61,111,162,107,2	79,61,144,119,111	158,120,8,2,112

Is there any way we can quantitatively measure the quality of different algorithms' results?

Reading Report of: *A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems* by Beck et al.

(ECE 695, Reading Assignment 01, Spring 2018)

10/12

Overview of the Paper

The paper deals with a fast solution of optimization problems similar but not limited to the ℓ-1 regularized least squares problem and is based on the Iterative Shrinkage-Thresholding Algorithm (ISTA). The premise is that the proposed algorithm, FISTA, keeps the simplicity of the ISTA algorithm while improving the worst case non-asymptotic global rate of convergence from $F(x_k) - F(x^*) = O(1/k)$ to $F(x_k) - F(x^*) = O(1/k^2)$, where $F(x_k)$ is the objective function, evaluated at the k-th estimate. The authors prove that the algorithm attains the claimed algorithmic complexity for any objective function $F(x) = f(x) + g(x)$, when $f(x)$ is a smooth, convex function that is Lipschitz-continuously differentiable and $g(x)$ is convex but not necessarily smooth. The improved performance of the algorithms is demonstrated on ill-conditioned deblurring problems and compared to the similar ISTA and MTWIST algorithms.

Prior Work

The paper mentions the existence of interior point methods which could be applied to the problem at hand, which however in the case of many practical problems involve dense matrix operations that inhibit the use on large scale problems. Furthermore, in the recent past of the publication of this paper there emerged several algorithms that are also based on the ISTA approach, one of which is the (M)TWIST. The FISTA algorithm builds upon an idea posed by Nestorov in 1983 for minimizing smooth convex functions, however the prove of algorithmic complexity is based upon a different method.

Key Ideas of the Paper

The key ideas of this paper are grounded in the ISTA algorithm. The authors improve the recursive update of the iterate and prove algorithmic complexity properties with respect to ISTA, thus we'll need some background on it's main ideas first.

Essential to ISTA is the quadratic approximation of $F(x) = f(x) + g(x)$ and resulting update:

$$p_L(y) = \arg\min_x \left\{ g(x) + \frac{L}{2} \left\| x - \left(y - \frac{1}{L} \nabla f(y) \right) \right\|^2 \right\} \tag{1}$$

where y could be an initial estimate of x and L is a step size. First, note that this optimization is relatively easy to compute when $g(x) = \lambda ||x||_1$ and $f(x) = ||Ax - b||^2$. That is because the coordinates of x are all separable and can be computed independent of each other. The remaining work is mostly the computation of the gradient, $\nabla f(y) = 2A^\top Ay - 2A^\top b$ and applying the shrinkage operator once per iteration. Both algorithms, ISTA and FISTA, may deploy a constant step size, L, or utilize a backtracking algorithm to find an appropriate step size at every step of the algorithm. The core difference is that the ISTA algorithm only uses the previous iterate, x_{k-1}, to determine the variable y, whereas the FISTA algorithm utilized the previous two iterates, x_{k-1} and x_{k-2}:

$$\text{ISTA}: \quad y_k = x_{k-1} \qquad x_k = p_L(y_k) \tag{2}$$

$$\text{FISTA}: \quad y_k = x_{k-1} + \left(\frac{t_k - 1}{t_k + 1} \right) (x_{k-1} - x_{k-2}) \qquad x_k = p_L(y_k) \tag{3}$$

where t_k follows a simple recurrence relation.

Comments

For my experiments I downloaded the code package at the provided URL that included a benchmark script for several well known ℓ-1 LS minimizers. First, I observed that not only do the solvers vary significantly in

the speed of the algorithms but also in terms of the quality of the solution. This can be seen in Figure 1. I wanted to analyze how different algorithms compare in these metrics and how that varies when the sparsity of x changes.

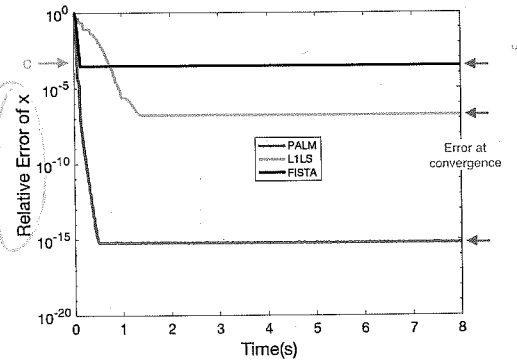


Figure 1: RMSE over time for choice of algorithms

Experimental Setup: A is an $m \times n = 800 \times 100$ matrix with random, zero-mean columns with unit variance. X_j is uniformly random in [-10, 10] where it is nonzero. $\lambda = 0.001$ and there is no noise. For the speed comparison I compare the time it takes until the RMSE is smaller than c (compare Figure 1) where $c = (1+\epsilon)\max_{\text{algorithm}} \min_k \text{RMSE}(x_k^{\text{algorithm}})$ is slightly larger than the final RMSE of the worst algorithm. For the quality comparison I ran the algorithms sufficiently long while registering the RMSE that they finally converge to (even if it takes them differently long to compute). Averages and error bars are shown for 10 runs at each data point in Figure 2. The PALM algorithm is arbitrarily chosen as a comparison among the choices that result in low error.

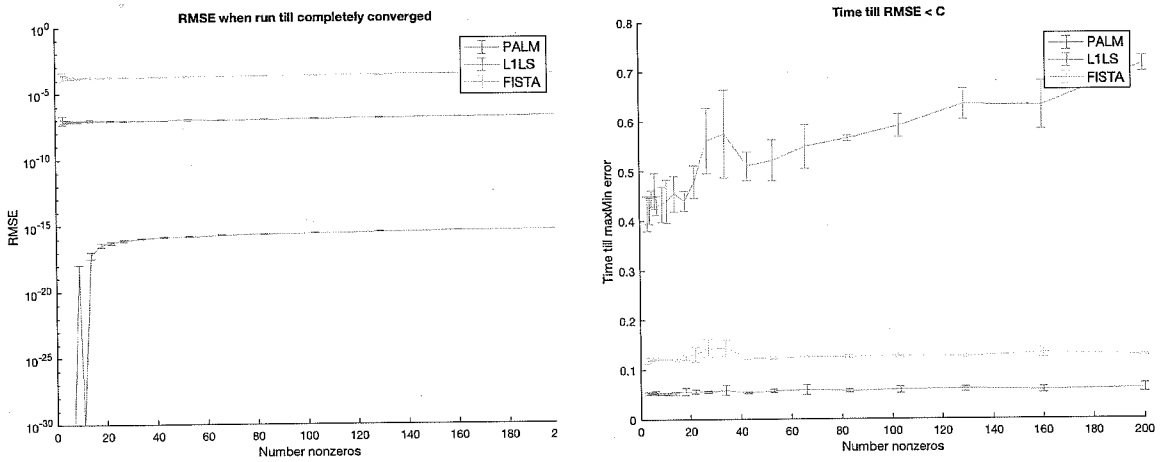


Figure 2: Left: RMSE at convergence. Right: Time RMSE is below threshold

Observations: When I started this experiment I was not sure whether the different RMSE's are due to better accuracy in terms of floating point operations or whether the algorithms actually produced superior solutions. To my surprise one can see that the RMSE of the PALM algorithm is not just always much smaller than the L1LS and FISTA algorithm but also gives RMSE $= 0$ (clipped to 10^{-30}) for small number of zeros when the others don't. The PALM algorithm is also much faster at arriving at a comparable solution.

The FISTA algorithm tends to be faster than the L1LS, however arrives at a solution with more RMSE.