Math 6590 Project III: Implementation of Split Bregman in three L1-regularized problems: image denoising and inpainting and compressive sensing

Name: Li Dong
RIN: 661243168

Apr 3, 2016

Abstract

In this report, split Bregman, as known as augmented Lagrangian or alternating direction method of multipliers(ADMM), is implemented for three specific L1-regularized problems: image denoising, image inpainting and compressive sensing. To extend the algorithm in image denoising problem to image inpainting problem and compressive sensing, only slight modification is needed. The effect of the penalization parameter on the convergence is investigated and the convergence of the algorithm for the compressive sensing is showed.

1 Problems

Ever since the proposition of the Rudin-Osher-Fatemi (ROF) model [1], mathematicians have been looking for an efficient and robust algorithm to solve the ROF model. Various methods are invented, such as gradient flow, lagged diffusivity fixed point iteration [2, 3], Chambolle projection [4] and split Bregman [5], among which split Bregman, also know as augmented Lagrangian or alternating direction method of multipliers(ADMM), is broadly-applicable and easy to be adapted to other L1-regularized problems, such as image inpainting and compressive sensing. The connection among split Bregman, augmented Lagrangian and ADMM as well as the convergence of split Bregman can be found in [6] In this report, the following three problems are solved with the split Bregman method.

1. Describe the augmented Lagrangian (split-Bregman/ADM) method for solving the Rudin-Osher-Fatemi model for 2D image denosing.

$$\min_u \mathcal{E}(u) = \int_\Omega |Du| \mathrm{d}x + \frac{\lambda}{2} \int_\Omega (u - u_0)^2 \mathrm{d}x \tag{1}$$

   Use Matlab to implement the above algorithm and test your code for the attached data.

2. Extend the above total variational method for image inpainting. Describe the model, the algorithm based on augmented Lagrangian method and test your code using the attached data.

3. Consider the compressive sensing problem

$$\min_{x \in \mathbb{R}^n} |x|, \qquad \text{s.t.} \qquad Ax = b \tag{2}$$

   Design an algorithm based on the augmented Lagrangian method to solve the above problem. Show the convergence of your algorithm and test your code using the attached data.

2 Methodology

2.1 Image denoising

Given the Eq.(1), we can first split the L1-regularization term ∇u and let $p = \nabla u$. Then we can write the augmented Lagrangian formulation as

$$\mathcal{L}(u, p; B) = \int_\Omega |p| \mathrm{d}x + \frac{\lambda}{2} \int_\Omega (u - u_0)^2 \mathrm{d}x + \frac{r}{2} \int_\Omega (\nabla u - p + B)^2 \mathrm{d}x, \tag{3}$$

where u_0 is the noisy image, $\lambda, r and B$ are regularization parameter, penalization parameter and Lagrangian multiplier, respectively. In order to solve 3, u, p and B are updated one by one and the following iteration is adopted:

$$\begin{aligned} u^k &= \min_u \frac{\lambda}{2} ||u - u_0||^2 + \frac{r}{2} ||\nabla u - p^{k-1} + B^{k-1}||^2, & (4) \\ p^k &= \min_p \int_\Omega |p| + \frac{r}{2} ||\nabla u^k - p + B^{k-1}||^2, & (5) \\ B^k &= B^{k-1} + \nabla u^k - p^k. & (6) \end{aligned}$$

An Euler-Lagrange equation is solved to find the solution to Eq.(4), a shrinkage operator is used to find the solution to Eq.5) and the solution to Eq.(6) is obvious. Thus, we have

$$\begin{aligned} (-r\Delta + \lambda) u^k &= \lambda u_0 + r \nabla \cdot (B^{k-1} - p^{k-1}), & (7) \\ p^k &= shrink(\nabla u^k + B^{k-1}, r) = max\{0, 1 - \frac{1}{r|\nabla u^k + B^{k-1}|}\}(\nabla u^k + B^{k-1}), & (8) \\ B^k &= B^{k-1} + \nabla u^k - p^k. & (9) \end{aligned}$$

Finite difference is used to solve Eq.(7) and the domain is discretized pixel-wise with unit element length. It is noted that Eq.(8) is component-wise operation.

2.2 Image inpainting

In the image inpainting problem, we have to complete the image where there is no information in certain domain. If we assume the domain where we have to fill in information is D, the augmented Lagrangian formulation can be written as

$$\mathcal{L}(u, p; B) = \int_\Omega |p| \mathrm{d}x + \frac{\lambda}{2} \int_{\Omega/D} (u - u_0)^2 \mathrm{d}x + \frac{r}{2} \int_\Omega (\nabla u - p + B)^2 \mathrm{d}x. \tag{10}$$

For the ease of computation, we introduce an indicator function

$$\chi_0 = \begin{cases} 0, & \text{in } D, \\ 1, & \text{otherwise.} \end{cases}$$

It is then obvious that we can rewrite Eq.(10) as

$$\mathcal{L}(u, p; B) = \int_\Omega |p| \mathrm{d}x + \frac{\lambda}{2} \int_\Omega \chi_0 (u - u_0)^2 \mathrm{d}x + \frac{r}{2} \int_\Omega (\nabla u - p + B)^2 \mathrm{d}x. \tag{11}$$

This means that we only need to multiply λ in Eq.(7) with a diagonal matrix which has zeros in entries in domain D and ones elsewhere.

2.3 Compressive sensing

The objective function of the compressive sensing problem can be written as

$$\min_x |x| \quad \text{subject to } Ax = b, \tag{12}$$

and the corresponding augmented Lagrangian formulation can be written as

$$\mathcal{L}(x, p; B) = |p| + \frac{\lambda}{2}||Ax - b||^2 + \frac{r}{2}||x - p + B||^2, \tag{13}$$

where $p = x$, λ is the regularization parameter and r is the penalization parameter. This formulation is almost identical to Eq.(3) except the fidelity term. Thus, we write a similar iteration strategy as following.

$$\begin{aligned}
(\lambda A^T A + r)x^k &= r(p^{k-1} - B^{k-1}) + \lambda A^T b, & (14)\\
p^k &= shrink(x^k + B^{k-1}, r) = max\{0, 1 - \frac{1}{r|x^k + B^{k-1}|}\}(x^k + B^{k-1}), & (15)\\
B^k &= B^{k-1} + x^k - p^k. & (16)
\end{aligned}$$

Theorem 1. *In the above algorithm, the sequence $\{p^k, x^k\}_{k=1}^{\infty}$ converges to a minimizer $\{p^*, x^*\}$ of the variation problem of Eq.(13).*

Proof. First we denote that $f(p) = |p|$ and $g(x) = \frac{\lambda}{2}||Ax - b||^2$ and both of them are clearly convex. The augmented Lagrangian formulation of the compressive sensing problem can be written as

$$\mathcal{L} = f(p) + g(x) + r||x - p||^2 + rB(x - p), \tag{17}$$

At the beginning of this proof, we need to introduce the concept of saddle point. (x^*, p^*, B^*) is called a saddle point of $\mathcal{L}(x, p; B)$ if

$$\mathcal{L}(x^*, p^*; B) \le \mathcal{L}(x^*, p^*; B^*) \le \mathcal{L}(x, p; B^*) \quad \forall x, p, B. \tag{18}$$

Now we make a claim and prove some properties of the saddle point.
Claim: If (x^*, p^*, B^*) is a saddle point of $\mathcal{L}(x, p; B)$ if and only if $p^* = x^*$ and x^* is a minimizer of Eq.(17).
Proof:

If (x^*, p^*, B^*) is a saddle point, the first inequality of (18) gives

$$\begin{aligned}
&f(p^*) + g(x^*) + \frac{r}{2}||x^* - p^*||^2 + r < x^* - p^*, B > \le f(p^*) + g(x^*) + \frac{r}{2}||x^* - p^*||^2 + r < x^* - p^*, B^* >,\\
&\Rightarrow r < x^* - p^*, B > \le r < x^* - p^*, B^* >, \quad \forall B,\\
&\Rightarrow x^* = p^*.
\end{aligned}$$

Considering $x^* = p^*$,the second inequality of (18) gives

$$f(p^*) + g(x^*) \le f(p) + g(x) + \frac{r}{2}||x - p||^2 + r < x - p, B^* > . \tag{19}$$

If we choose $p = x$, we can easily have $f(x^*) + g(x^*) \le f(x) + g(x)$.

On the other hand, suppose (x^*, p^*) is a solution of Eq.(17). The first inequality of (18) is trivial.

Now let's compute the derivatives of Eq.(17).

$$\frac{\delta \mathcal{L}}{\delta p} = \delta f(p) - r(x-p) - rB, \tag{20}$$

$$\frac{\delta \mathcal{L}}{\delta x} = \delta g(x) + r(x-p) + rB, \tag{21}$$

where $\delta f(p)$ and $\delta g(x)$ represent the corresponding subdifferential sets. From the problem statement, we have

$$p^* = \min_p \mathcal{L}(p, x^*; B^*) \Rightarrow \frac{\delta \mathcal{L}}{\delta p} = 0 \Rightarrow rB^* \in \delta f(p^*), \tag{22}$$

$$x^* = \min_x \mathcal{L}(p^*, x; B^*) \Rightarrow \frac{\delta \mathcal{L}}{\delta x} = 0 \Rightarrow -rB^* \in \delta g(x^*). \tag{23}$$

Here please note that $p^* = x^*$. According to the properties of subdifferential set, we can write,

$$f(p) - f(p^*) - r < B^*, p - p^* > \geq 0 \quad \forall p, \tag{24}$$

$$g(x) - g(x^*) + r < B^*, x - x^* > \geq 0 \quad \forall x. \tag{25}$$

where $< \cdot, \cdot >$ defines inner product.

Now if we sum up (24) and (25), we can have

$$f(p) + g(x) \geq f(p^*) + g(x^*) + r < B^*, p - x > . \tag{26}$$

Now if we go back to Eq.(17), we have

$$\begin{aligned} \mathcal{L}(x^*, p^*; B^*) &= f(p^*) + g(x^*), \\ &\leq f(p) + g(x) + r < x - p, B^* >, \\ &\leq f(p) + g(x) + r < x - p, B^* > + \frac{r}{2}||x - p||^2 = \mathcal{L}(x, p; B^*), \end{aligned}$$

Hence the second inequality of (18). This completes the proof for this claim.

■

Denote $\bar{p}^* = p^k - p^*$, $\bar{x}^* = x^k - x^*$ and $\bar{B}^* = B^k - B^*$, where $\{p^*, x^*, B^*\}$ is the saddle point of $\mathcal{L}$. Again, for p^* and x^* we have results (22) and (23). Similarly, for p^k and x^k, we have

$$f(p) - f(p^k) - r < x^k - p^k + B^{k-1}, p - p^k > \geq 0, \tag{27}$$

$$g(x) - g(x^k) + r < x^k - p^{k-1} + B^{k-1}, x - x^k > \geq 0. \tag{28}$$

If we let $p = p^k$ in (24) and $p = p^*$ in (27) and sum the two inequalities, this gives

$$r < \bar{x}^k - \bar{p}^k + \bar{B}^{k-1}, \bar{p}^k > \geq 0. \tag{29}$$

Also, we let $x = x^k$ in (25) and $x = x^*$ in (28) and sum the two, this gives

$$r < \bar{x}^k - \bar{p}^{k-1} + \bar{B}^{k-1}, -\bar{x}^k > \geq 0. \tag{30}$$

Summing inequalities (29) and (30) gives

$$-r||\bar{x}^k - \bar{p}^k||^2 - r < \bar{B}^{k-1}, \bar{x}^k - \bar{p}^k > -r < \bar{p}^k - \bar{p}^{k-1}, \bar{x}^k > \geq 0. \tag{31}$$

From Eq.(6), we can write

$$r||\bar{B}^{k-1}||^2 - r||\bar{B}^k||^2 = -2r < \bar{B}^{k-1}, \bar{x}^k - \bar{p}^k > -r||\bar{x}^k - \bar{p}^k||^2 \geq r||\bar{x^k} - \bar{p}^k||^2 + 2r < \bar{x}^k, \bar{p}^k - \bar{-}^{k-1} > \quad (32)$$

where the latter inequality is obtained by plugging in the inequality (31). Now, we focus on $< \bar{x}^k, \bar{p}^k - \bar{p}^{k-1} >$.

$$< \bar{x}^k, \bar{p}^k - \bar{p}^{k-1} > = < \bar{x}^k - \bar{x}^{k-1}, \bar{p}^k - \bar{p}^{k-1} > + < \bar{x}^{k-1} - \bar{p}^{k-1}, \bar{p}^k - \bar{p}^{k-1} > + < \bar{p}^{k-1}, \bar{p}^k - \bar{p}^{k-1} > . \quad (33)$$

Further, $p^{k-1} \min_p \mathcal{L}(p, x^{k-1}, B^{k-2})$, similar as (24), gives

$$f(p) - f(p^{k-1}) + r < -(x^{k-1} - p^{k-1} + B^{k-2}), p - p^{k-1} > \geq 0 \quad \forall p. \quad (34)$$

Also, we let $p = p^k$ in (34) and $p = p^{k-1}$ in (27) and sum the two inequalities. This gives

$$< B^{k-1} - B^{k-2}, p^k - p^{k-1} > + < p^{k-1} - p^k + x^k - x^{k-1}, p^k - p^{k-1} > \geq 0. \quad (35)$$

' Since we have

$$\begin{aligned} B^{k-1} - B^{k-2} &= x^{k-1} - pk - 1 = x^{\bar{k-1}} - p^{\bar{k-1}}, & (36) \\ p^k - p^{k-1} &= \bar{p}^k - \bar{p}^{k-1}, & (37) \\ x^k - x^{k-1} &= \bar{x}^k - \bar{x}^{k-1}, & (38) \end{aligned}$$

we can reduce inequality (35) to

$$< \bar{x}^{k-1} - \bar{p}^{k-1}, \bar{p}^k - \bar{p}^{k-1} > + < \bar{x}^k - \bar{x}^{k-1}, \bar{p}^k - \bar{p}^{k-1} > \geq ||\bar{p}^k - \bar{p}^{k-1}||^2. \quad (39)$$

Thus, the Eq.(33) can be written as

$$< \bar{x}^k, \bar{p}^k - \bar{p}^{k-1} > \geq ||\bar{p}^k - \bar{p}^{k-1}||^2 + < \bar{p}^{k-1}, \bar{p}^k - \bar{p}^{k-1} > . \quad (40)$$

Now we return to the inequality (32),

$$\begin{aligned} r||\bar{B}^{k-1}||^2 - r||\bar{B}^k||^2 &\geq r||\bar{x}^k - \bar{p}^k||^2 + 2r(||\bar{p}^k - \bar{p}^{k-1}||^2 + < \bar{p}^{k-1}, \bar{p}^k - \bar{p}^{k-1} >), \\ &= r||\bar{x}^k - \bar{p}^k||^2 + r(||\bar{p}^k||^2 + ||\bar{p}^{k-1}||^2 + ||\bar{p}^k - \bar{p}^{k-1}||^2). \end{aligned}$$

By rearranging the above inequality, we can have

$$r(||\bar{B}^{k-1}||^2 + ||\bar{p}^{k-1}||^2) - r(||\bar{B}^k||^2 + ||\bar{p}^k||^2) \geq r||\bar{x}^k - \bar{p}^k||^2 + ||\bar{p}^k - \bar{p}^{k-1}||^2, \quad (41)$$

which clearly demonstrates that $\{r(||\bar{B}^k||^2 + ||\bar{p}^k||^2)\}_{k=1}^{\infty}$ is a non-increasing sequence and converges. Now we can conclude that $\{B^k\}$, $\{p^k\}$ and $\{x^k\}$ are bounded and $\lim_{k\to\infty} ||x^k - p^k|| = 0$, $\lim_{k\to\infty} ||p^k - p^{k-1}|| = 0$.

Next, we claim that

Claim: $\lim_{k\to\infty} f(p^k) + g(x^k) = f(p^*) + g(x^*)$

Proof: Since (x^*, p^*, B^*) is a saddle point $\forall x and p$, $\mathcal{L}(x^*, p^*; B^*) \leq \mathcal{L}(x, p; B^*)$. Choose $x = x^*$ and $p = p^*$, we have

$$f(p^*) + g(x^*) \leq f(p^k) + g(x^k) + \frac{r}{2}||x^k - p^k|| + r < x^k - p^k, B^* > . \quad (42)$$

Let $p = p^*$ in (27) and $x = x^*$ in (28) and sum the two up, we have

$$\begin{aligned} f(p^*) + g(x^*) &\geq f(p^k) + g(x^k) + r < x^k - p^k + B^{k-1}, p^* - p^k > -r < x^k - p^k + B^{k-1}, x^* - x^k >, \\ &= f(p^k) + g(x^k) + r < x^k - p^k, p^* - p^k > -r < x^k - p^{k-1}, x^* - x^k > + r < B^{k-1}, x^k - p^k > . \end{aligned}$$

Since $\lim_{k\to\infty} ||x^k - p^k|| = 0$, Claim is proved. ■

Again, if we let $p = p^k$ and $x = x^k$ in (24) and (25) and sum the two up, this gives

$$\begin{aligned} f(p^k) + g(x^k) + r < B^*, x^k - p^k > &\geq f(p^*) + g(x^*) - r < B^*, p^* - p^k > + r < B^*, x^k - p^k >, \\ &= f(p^*) + g(x^*) - g(x^*) + g(x^k) + r < B^*, x^k - p^* >, \\ &= f(p^*) + g(u^*) - \frac{\lambda}{2}||Ax^* - b||^2 + \frac{\lambda}{2}||Ax^k - b||^2 + r < B, x^k - p^* > . \end{aligned}$$

Since $D_x\mathcal{L}(x^*) = 0$ gives $-rB^* = \lambda A^T(Ax^* - b)$ as well as $p^* = x^*$, the above equation can be written as

$$\begin{aligned} &= f(p^*) + g(u^*) - \frac{\lambda}{2}||Ax^* - b||^2 + \frac{\lambda}{2}||Ax^k - b||^2 - \lambda < A^TAx^* - b, x^k - x^* >, \\ &= f(p^*) + g(x^*) + \frac{\lambda}{2}||A(x^k - x^*)||^2. \end{aligned}$$

Now that we know $\lim_{k\to\infty} f(p^k) + g(x^k) = f(p^*) + g(x^*)$ and $\lim_{k\to\infty} ||x^k - p^k|| = 0$, it is easy to see that $\lim_{k\to\infty} ||x^k - x^*|| = 0$.

Consider $||p^k - p^*|| \leq ||p^k - x^k|| + ||x^k - p^*||$, since $p^* = x^*$, $\lim_{k\to\infty} ||x^k - x^*|| = 0$ and $\lim_{k\to\infty} ||p^k - x^k|| = 0$, hence $\lim_{k\to\infty} ||p^k - p^*|| = 0$. This completes the proof.

□

3 Results

In this part, results for the above three problems and the convergence of the corresponding algorithms are presented. In this study, we explored the effects of the penalization parameter r as well as the regularization parameter λ. Three convergence metrics are plotted here, the convergence of the auxiliary variable p, the difference between the reconstructed solution and the true solution and the objective function value. The termination criterion for all three algorithms is the iteration number.

3.1 Image denoising

Figure 1 shows the result of the image denoising problem and Figure 2 displays the convergence when varying the penalization parameter r. In Figure 2, the first row is when $r = 0.0001$ and the second row is when $r = 0.001$. As stated in the beginning of this section, we are using three convergence metrics as shown in the three columns, specifically $||p^k - \nabla u^k||_F$, the PSNR between the reconstructed image and the original image and value of Eq.(1). By comparing the two rows of Figure 2, it is easy to see that the second row where $r = 0.001$ is more appropriate than the first row where $r = 0.0001$ because the algorithm converges much faster in the second row. The fact that the first column approaches to zero means that the auxiliary variable p approaches to ∇u. The PSNR of the image first quickly grows larger and then drops a little bit indicates that the quality of the image become better during the progression of the algorithm and it degrades somewhat after certain iterations. This degradation might be caused by the instability of the finite difference solving strategy, though both the first and the last columns give consistent convergence.

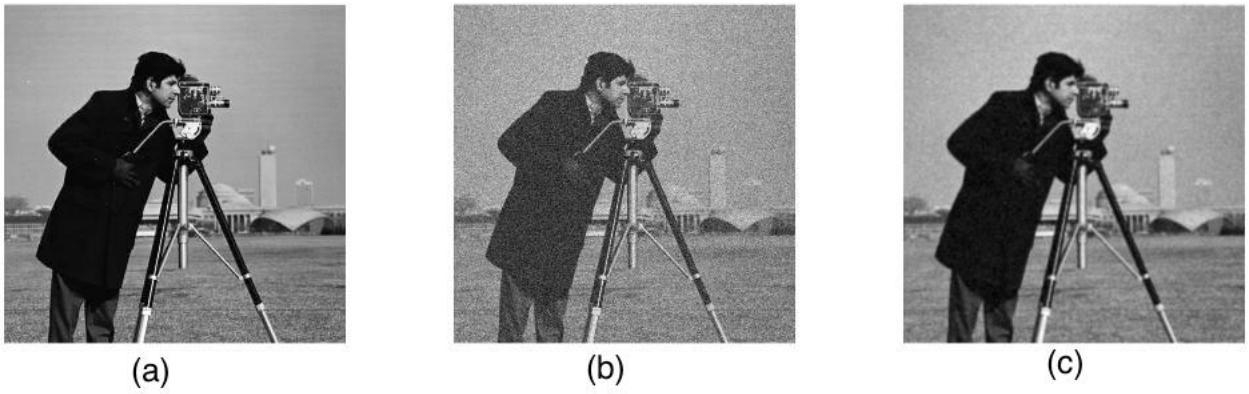

Figure 1: Results for image denoising problem, (a)original image, (b)noisy image, (c)reconstructed image, $\lambda = 0.01, r = 0.001$.

One thing that is worth mentioning in the implementation is that only one conjugate gradient solve was performed to find the inexact solution of the Euler-Lagrange Eq.(7). This greatly accelerated the iteration. This is especially favorable when the Euler-Lagrange equation is expensive to solve in some circumstances.

3.2 Image inpainting

Figure 3 shows the result from the image inpainting problem. Since we have shown the effects of the penalization parameter r in the above problem and this inpainting problem resembles that a lot, we only show the effects of the regularization parameter λ. As displayed in Figures 3(c) and 3(d), λ were chosen to be 100 and 1, respectively. Because there are no noise in the contaminated image, we can set λ to be relatively large meaning more confidence is put in the fidelity term. From the comparison between the results from the two λ, it is clear that smaller value tends to smooth out, or blur, the image.

In this problem, somehow inexact solution from the conjugate gradient solver to the Euler-Lagrange equation does not lead to good quality images. Even more exact solution from the conjugate gradient solver in each iteration is not able to improve the quality of the images to a satisfied criterion, which means there are still some visible black lines in the image. Thus, in the implementation, backslash is used to solve the Euler-Lagrange equation. Surprisingly, only one iteration is needed to finish the inpainting and results in good quality images. However, what is not intuitive in this case is that more iterations lead to completely degraded images, meaning the image becomes a single-color chunk. That is also the reason that the convergence plots are not recorded for this problem. In my opinion, this instability might be caused by the finite difference strategy to solve the Euler-Lagrange equation, because people have used FFT solving the Euler-Lagrange equation without any problem.

3.3 Compressive sensing

Figures 4 and 5 show the results and the convergence of the algorithm developed in Section 2.3. Figure 4(a) displays the ground truth of the signal and Figure 4(b) displays the reconstructed signal. In this case, the regularization parameter λ is fixed to 10 in all cases and only the penalization parameter r is varied with 100, 1 and 10. Since the results do not depend on the choices of r, only one result is presented here. As shown in Figure 5, $r = 10$, the third row, is the most appropriate among the three choices. In Figure 5, as usual, three convergence metrics are presented in each column and the first column is, again, the auxiliary variable p, the second column is $|x^k - x_true|$ and the third column is the value of

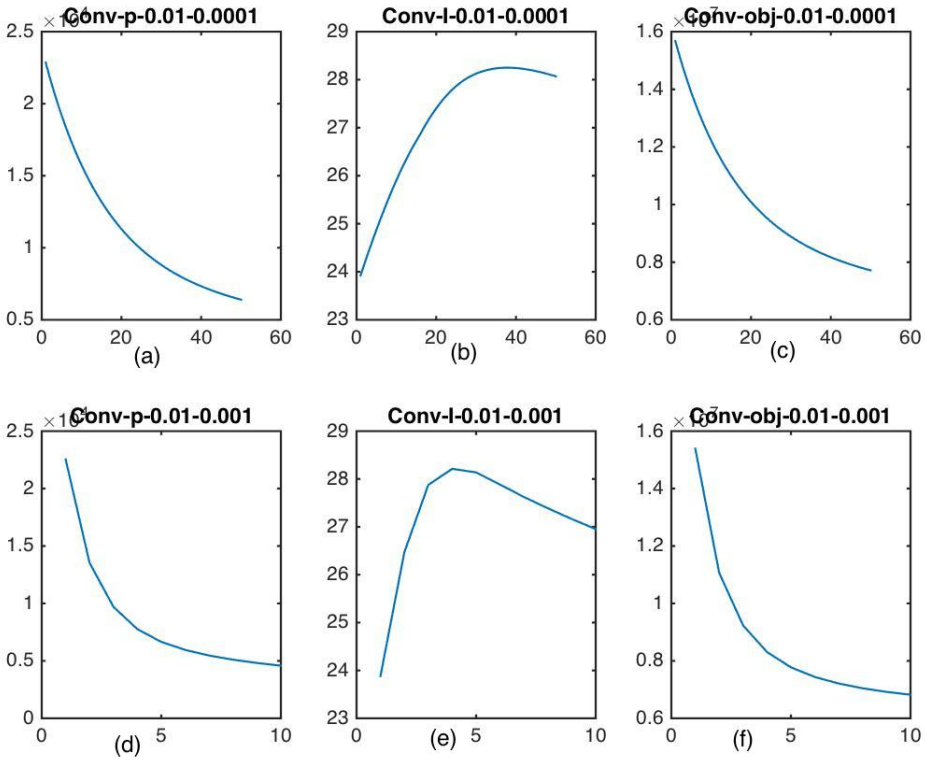

Figure 2: Convergence for image denoising problem with different penalization parameter r, $r = 0.0001$ for the first row and $r = 0.001$ for the second row, (a) and (d) are convergence of the auxiliary variable p, (b) and (e) are the convergence in terms of the image PSNR, (d) and (f) are the convergence of the objective function (1). The horizontal axes represent the iteration number and the vertical axes represent the corresponding values.

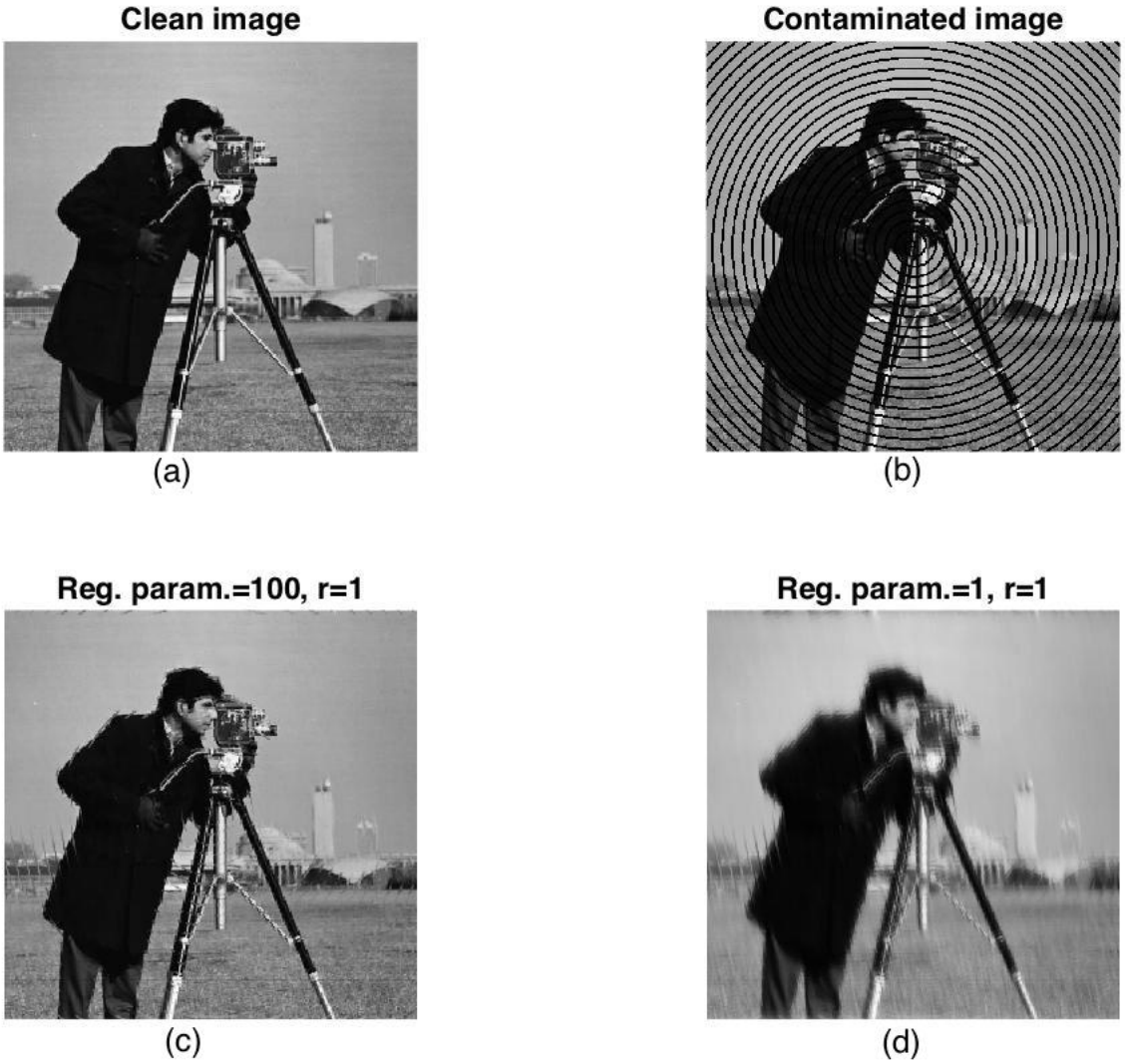

Figure 3: Results for image inpainting problem, (a)original image, (b)noisy image, (c)reconstructed image with $\lambda = 100, r = 1$, (d)reconstructed image with $\lambda = 1, r = 1$.

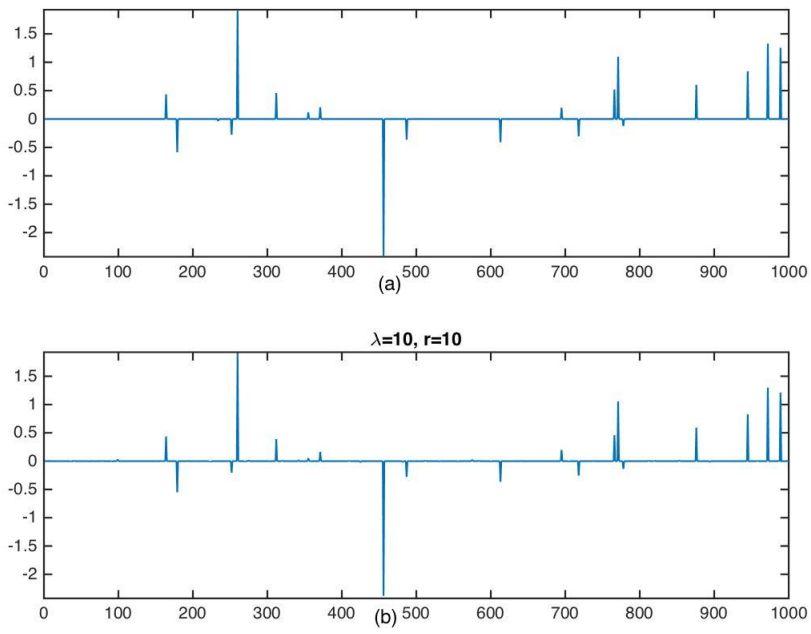

Figure 4: Results for the compressed sensing problem, (a)original signal, (b)reconstructed signal.

the objective function Eq.(12). Similarly, the first column of plots shows that the auxiliary variable p converges to x, the second column of plots shows that x converges to $x_t rue$ and the third column of plots shows that the objective function is getting smaller as the algorithm progressed. The first row of the plots clearly indicates the consequence of selecting a relatively large penalization parameter, converging slowly. In the second row of the plots, the choice of a relatively smaller r also show relatively slow convergence comparing to the third row of plots, where r is chosen to be 10. Especially, by inspecting Figures 5(e) and 5(h), we can see that when $r = 10$, it converged only after around 60 iterations; When $r = 1$, the convergence curve is not smooth which indicates some instability and it only converged after roughly iterations.

It is also noted that in this case the Euler-Lagrange equation is solve with backslash, since the inexact solution, i.e. one solve from conjugate gradient, does not give reasonable results.

4 Observation and Conclusions

1. Split Bregman is a relatively general framework to tackle L1-regularized problems and provides excellent convergence.
2. The regularization parameter λ works in the same way as the previous two reports. It is noted that larger value can be and should be used when the noise in the image is not much or none, such as the image inpainting case, or we want a relatively hard constrain, such as the compressive sensing case.
3. The penalization parameter r controls the convergence rate yet the result does not depend on it. r can neither be too large nor too small, in both cases it will slow the convergence though the final result should be the same when it converges.
4. Although technically, the inexact solution to the Euler-Lagrange equation can lead to convergence, this might be affected by certain circumstances in practice. In this report, the inexact solution

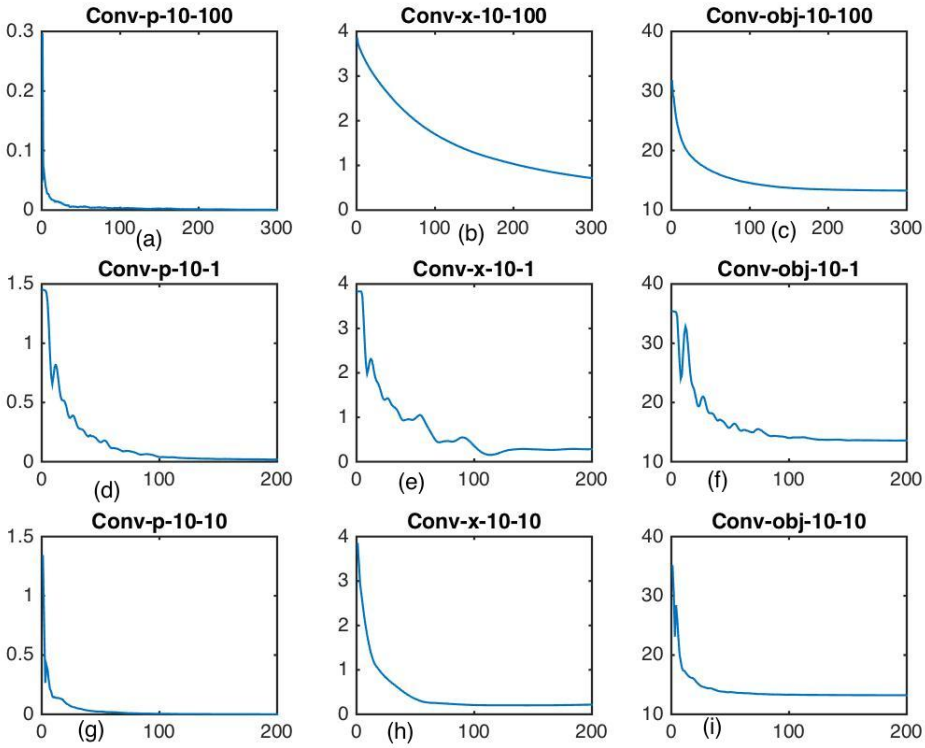

Figure 5: Convergence for the compressed sensing problem with different penalization parameter r, $r = 100$ for the first row, $r = 1$ for the second row and $r = 10$ for the third row, (a), (d) and (g) are convergence of the auxiliary variable p, (b), (e) and (h) are the convergence of the signal x, (d), (f) and (i) are the convergence of the objective function (12). The horizontal axes represent the iteration number and the vertical axes represent the corresponding values.

technique works for the image denoising problem but not for the other two problems. The reason might be the finite difference strategy but need further verification.

5. In the image inpainting problem, the algorithm is finished after one iteration only and more iterations degrade the image significantly. This might be caused by the finite difference strategy, since there is a successful case where people implement exactly the same function with the only difference being the FFT solver.

References

[1] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[2] Curtis R Vogel and Mary E Oman. Iterative methods for total variation denoising. *SIAM Journal on Scientific Computing*, 17(1):227–238, 1996.

[3] Curtis R Vogel. *Computational methods for inverse problems*, volume 23. Siam, 2002.

[4] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2):89–97, 2004.

[5] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[6] Chunlin Wu and Xue-Cheng Tai. Augmented lagrangian method, dual methods, and split bregman iteration for rof, vectorial tv, and high order models. *SIAM Journal on Imaging Sciences*, 3(3):300–339, 2010.

Appendix

MATLAB codes.

main.m calls three problems and save results.

```
clear
close all



problem = 1;

switch problem

    case 1
        [I_exact, I_noise, I_contaminated, D] = LoadData;

        iter=[50 10];
        D=ones(size(D));
        lambda=[1e-2 1e-2]; r=[1e-4 1e-3];

        h=figure; subplot(1,3,1); imshow(I_exact,[]); title('Clean image');
                  subplot(1,3,2); imshow(I_noise,[]); title('Noisy image');
        c=figure;
        for j=1:length(lambda)
```

```
        conv_p=zeros(iter(j),1);
        conv_I=zeros(iter(j),1);
        conv_obj=zeros(iter(j),1);
        p_k=zeros([size(I_noise), 2]); B_k=zeros([size(I_noise), 2]);
        I_k=I_noise;
        for i = 1:iter(j)
           I_k=solveEulerLagrange(I_noise,p_k,B_k,lambda(j),r(j),D,2);
           p_k=shrink(I_k,B_k,r(j));
           grad_I_k=grad(I_k);
           B_k=B_k + grad_I_k - p_k;

           conv_p_tmp=p_k-grad(I_k);
           conv_p(i)=norm(conv_p_tmp(:));
           conv_I(i)=psnr(I_k,I_exact); %norm(I_k(:)-I_exact(:));
           tmp=sqrt(grad_I_k(:,:,1).^2+grad_I_k(:,:,2).^2);
           conv_obj(i)=sum(tmp(:)) + 0.5*lambda(j)*norm(I_k-I_contaminated,'fro')^2;

            figure(c); subplot(2,3,j*j); plot(conv_p);
            title(sprintf('Conv-p-%g-%g',lambda(j),r(j)));
            figure(c); subplot(2,3,j*j+1); plot(conv_I);
            title(sprintf('Conv-I-%g-%g',lambda(j),r(j)));
            figure(c); subplot(2,3,j*j+2); plot(conv_obj);
            title(sprintf('Conv-obj-%g-%g',lambda(j),r(j)));
            figure(h); subplot(1,3,3); %subplot(2,2,j+2);
            imshow(I_k(:,:),[]);
            title(sprintf('Reg. param.=%g, r=%g',lambda(j),r(j)));

            drawnow
        end
    end
    saveas(h,'1_Denoise.jpg');
    saveas(c,'1_Conv_p_I.jpg');
case 2
    [I_exact, I_noise, I_contaminated, D] = LoadData;

    iter=[1 1];
    D=1.-D;
    lambda=[1e2 1]; r=[1 1];

    h=figure; subplot(2,2,1); imshow(I_exact,[]); title('Clean image');
              subplot(2,2,2); imshow(I_contaminated,[]); title('Contaminated image');
    c=figure;
    for j=1:length(lambda)
        conv_p=zeros(iter(j),1);
        conv_I=zeros(iter(j),1);
        conv_obj=zeros(iter(j),1);
        p_k=zeros([size(I_contaminated), 2]); B_k=zeros([size(I_contaminated), 2]);
        I_k=I_contaminated;
        for i = 1:iter(j)
           I_k=solveEulerLagrange(I_contaminated,p_k,B_k,lambda(j),r(j),D,1);
           p_k=shrink(I_k,B_k,r(j));
           grad_I_k=grad(I_k);
           B_k=B_k + grad_I_k - p_k;
```

```
                conv_p_tmp=p_k-grad(I_k);
                conv_p(i)=norm(conv_p_tmp(:));
                conv_I(i)=norm(I_k(:)-I_exact(:));
                tmp=sqrt(grad_I_k(:,:,1).^2+grad_I_k(:,:,2).^2);
                conv_obj(i)=sum(tmp(:)) + 0.5*lambda(j)*norm(I_k-I_contaminated,'fro')^2;

                 figure(c); subplot(2,3,j*j); plot(conv_p);
                 title(sprintf('Conv-p-%g-%g',lambda(j),r(j)));
                 figure(c); subplot(2,3,j*j+1); plot(conv_I);
                 title(sprintf('Conv-I-%g-%g',lambda(j),r(j)));
                 figure(c); subplot(2,3,j*j+2); plot(conv_obj);
                 title(sprintf('Conv-obj-%g-%g',lambda(j),r(j)));
                 figure(h); subplot(2,2,j+2); imshow(I_k(:,:),[]);
                 title(sprintf('Reg. param.=%g, r=%g',lambda(j),r(j)));

                 drawnow
             end
         end
         saveas(h,'2_Inpaint.jpg');
         saveas(c,'2_Conv_p_I.jpg');
     case 3
         [A, b, x_t] = LoadDataForCompressiveSensing;
         c=figure;
         lambda=1e1;
         r=[1e2; 1; 1e1]; iter=[300 200 200];
         h=figure; subplot(2,1,1); %subplot(length(iter),1,1);
         plot(x_t); ylim([min(x_t) max(x_t)]);
         for j=1:length(iter)
             B1_k=zeros(size(x_t)); B2_k=zeros(size(b));
             p_k=zeros(size(x_t));
             x_k=zeros(size(x_t));
             conv_obj=zeros(size(iter(j)));
             conv_x=zeros(size(iter(j)));
             conv_p=zeros(size(iter(j)));
            for i=1:iter(j)
                x_k=solveEulerLagrangeCS(A,b,p_k,B1_k,B2_k,r(j),lambda);
                p_k=shrink(x_k,B1_k,r(j));
                B1_k=B1_k + x_k - p_k;
                B2_k=0;%B2_k+A*x_k-b;

                conv_p(i)=norm(p_k-x_k);
                conv_x(i)=norm(x_k-x_t);
                conv_obj(i)=sum(abs(x_k));

                 figure(c); subplot(length(iter),3,3*j-2); plot(conv_p);
                 title(sprintf('Conv-p-%g-%g',lambda,r(j)));
                 figure(c); subplot(length(iter),3,3*j-2+1); plot(conv_x);
                 title(sprintf('Conv-x-%g-%g',lambda,r(j)));
                 figure(c); subplot(length(iter),3,3*j-2+2); plot(conv_obj);
                 title(sprintf('Conv-obj-%g-%g',lambda,r(j)));

%                  figure(h); subplot(length(iter)+1,1,j+1); plot(x_k);
                 figure(h); subplot(2,1,2); plot(x_k); ylim([min(x_t) max(x_t)]);
                 title(sprintf('\\lambda=%g, r=%g',lambda,r(j,1)));
```

```
                drawnow
            end
        end
        saveas(h,'3_CS.jpg');
        saveas(c,'3_Conv_p_I.jpg');
end
```

solveEulerLagrange.m solves the Euler-Lagrange equation in the image denoising and inpainting problems.

```
function [I_kk]=solveEulerLagrange(I_0,p_k,B_k,lambda,r,D,solver)

% laplacian operator
[ny, nx] = size(I_0);
ex = ones(nx,1);
Dxx = spdiags([ex -2*ex ex], [-1 0 1], nx, nx);
Dxx(1,1) = -1; Dxx(1,2) = 1; % impose homogeneous Neumann
Dxx(end,end-1) = 1; Dxx(end,end) = -1;
ey = ones(ny,1);
Dyy = spdiags([ey, -2*ey ey], [-1 0 1], ny, ny);
Dyy(1,1) = -1; Dyy(1,2) = 1; % impose homogeneous Neumann
Dyy(end,end-1) = 1; Dyy(end,end) = -1;
Lap2 = kron(Dyy, speye(nx)) + kron(speye(ny), Dxx);

div_B_p=div(B_k(:,:,1)-p_k(:,:,1), B_k(:,:,2)- p_k(:,:,2));
A=(-r.*Lap2 + spdiags(D(:),0,speye(size(Lap2)))*lambda);
b=(D(:)*lambda.*I_0(:) + r.*div_B_p(:));
switch solver
    case 1
        I_kk= A \ b;
    case 2
        I_kk = pcg(A,b,[],1);
end
I_kk=reshape(I_kk,size(I_0));


function fd = div(Px,Py)

nbdims = 2;
if size(Px,1)==1 || size(Px,2)==1
    nbdims = 1;
end

fx = Px-Px([1 1:end-1],:);
if nbdims>=2
    fy = Py-Py(:,[1 1:end-1]);
end

if nbdims==2
    fd = fx+fy;
else
    fd = fx;
end
```

solveEulerLagrangeCS.m solves the Euler-Lagrange equation in the compressive sensing problem.

```
function [x]=solveEulerLagrangeCS(A0,b0,p_k,B1_k,B2_k,r1,lambda)

A_tmp=lambda*(A0'*A0)+r1*speye(size(A0,2));
b_tmp=r1*(p_k-B1_k)+lambda*A0'*(b0-B2_k);
% x = pcg(A_tmp,b_tmp,[],1);
x=A_tmp\b_tmp;
```

shrink.m implements the shrinkage operator for all three problems.

```
function p_k=shrink(I_k,B_k,r)

if size(I_k,2)~=1
    grad_I_k=grad(I_k);
    IkBk_1=grad_I_k(:,:,1)+B_k(:,:,1);
    IkBk_2=grad_I_k(:,:,2)+B_k(:,:,2);
    IkBk=sqrt(IkBk_1.^2 + IkBk_2.^2);
    tmp=1-1./(r*IkBk);
    tmp(tmp<0)=0;
    p_k(:,:,1)=tmp .* (grad_I_k(:,:,1)+B_k(:,:,1));
    p_k(:,:,2)=tmp .* (grad_I_k(:,:,2)+B_k(:,:,2));
else
    IkBk=abs(I_k+B_k);
    tmp=1-1./(r*IkBk);
    tmp(tmp<0)=0;
    p_k=tmp .* (I_k+B_k);
end
```

grad.m implements the gradient operator that is used in the image denoising and inpainting problems.

```
function [fx] = grad(M)

nbdims = 2;
if size(M,1)==1 || size(M,2)==1
    nbdims = 1;
end

fx = M([2:end end],:)-M;

if nbdims>=2
    fy = M(:,[2:end end])-M;
end

if nbdims==2
    fx = cat(3,fx,fy);
end
```