doi:10.11887/j.cn.201802018 http://journal.nudt.edu.cn

图像恢复中的稳健交替方向乘子法*

吴 越,曾向荣,周典乐,刘 衍,周家庆,周经纶
(国防科技大学 系统工程学院,湖南 长沙 410073)

摘 要:交替方向乘子法在解决线性逆问题(包括图像恢复)中取得了良好的效果,但是其效果对惩罚参数的选择非常敏感,不利于具体的应用。提出基于惩罚参数自适应选择原理的稳健交替方向乘子法,对其优化条件和收敛性进行了详细分析。实验表明,在基于 Parseval 紧框架的图像恢复应用中,该算法不但对惩罚参数的选择表现出良好的稳健性,而且效果优于交替方向乘子法,并优于其他目前热门的算法。

关键词:交替方向乘子法;图像恢复;压缩感知;紧框架
中图分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2018)02-112-07

Image restoration via robust alternating direction method of multipliers

WU Yue, ZENG Xiangrong, ZHOU Dianle, LIU Yan, ZHOU Jiaqing, ZHOU Jinglun
(College of Systems Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: The ADMM (alternating direction method of multipliers) with appropriate parameters plays a successful role in solving linear inverse problems (including image restoration). But the results obtained by ADMM are sensitive to the choices of the penalty parameter, and this bad robustness brought some troubles in its applications. Based on a scheme of choosing the penalty parameter adaptively, a RADMM (robust ADMM) was proposed to tackle this shortcoming. Through analyzing optimization conditions and convergence of RADMM, we can conclude that the adaptive control of the penalty parameter provides good robustness, faster speed of convergence and better solution. And the experiments show that, in the application of image restoration based on the Parseval tight frame, the RADMM is robust to the choices of the penalty parameter, outperforms the ADMM, and is far superior to other alternative state-of-the-art methods.

Key words: alternating direction method of multipliers; image restoration; compressed sensing; tight frame

图像恢复是一个经典的线性逆问题:

$$\boldsymbol{y}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{n} \tag{1}$$

其中:$\boldsymbol{x}$ 是原图像,即目标图像;$\boldsymbol{A}$ 是线性算子的矩阵表示,一般为模糊矩阵;$\boldsymbol{y}$ 为观测图像,即需要修复的图像;$\boldsymbol{n}$ 则为噪声。众所周知,从 $\boldsymbol{y}$ 中修复出 $\boldsymbol{x}$ 是一个欠定问题,即存在多个解。幸运的是,如果 $\boldsymbol{x}$ 是稀疏的或在某个特定的域里稀疏,可以利用压缩感知(compressed sensing)理论[1-2]以较少的代价得到满意结果。

对于图像来说,$\boldsymbol{x}$ 表示的是各个像素点的灰度值,因此 $\boldsymbol{x}$ 不是稀疏变量。但是图像在某些特定的小波或框架(一般假设为紧框架)中表示的系数是稀疏的。文献[3]提供了这两种表示方法——分析(analysis)和合成(synthesis)方法。对于前者来说,其模型为:

$$\min_{\boldsymbol{x}}\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\lambda\|\boldsymbol{P}\boldsymbol{x}\|_1 \tag{2}$$

其中,λ 为正则化参数,$\boldsymbol{P}$ 为分析算子,则 $\boldsymbol{P}\boldsymbol{x}$ 表示的是在小波基或紧框架下的稀疏系数[4]。后者的模型为:

$$\min_{\boldsymbol{s}}\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{W}\boldsymbol{s}\|_2^2+\lambda\|\boldsymbol{s}\|_1 \tag{3}$$

其中,$\boldsymbol{W}$ 为框架矩阵,则图像 $\boldsymbol{x}=\boldsymbol{W}\boldsymbol{s}$,其中 $\boldsymbol{s}$ 为相应的稀疏系数。注意式(2)和式(3)都是凸优化问题,其正则化项 l_1 范数惩罚的都是稀疏系数,从而达到其正则化的目的。

解决问题(2)~(3)最经典的方法是迭代阈值收缩(Iterative Shrinkage/Thresholding, IST)算法,其核心是所谓的阈值收缩函数,也被称为 Moreau 近似映射或去噪函数,定义为:

* **收稿日期**:2016-12-01
基金项目:国家自然科学基金资助项目(61602494);国防科技大学科研计划资助项目(ZK16-03-16)
作者简介:吴越(1977—),男,浙江绍兴人,博士研究生,E-mail:wuyuework@163.com;
曾向荣(通信作者),男,湖南涟源人,讲师,博士,E-mail:zengxrong@gmail.com

$$Prox_{\lambda f}(\boldsymbol{t}) = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{t}-\boldsymbol{x}\|_2^2 + \lambda f(\boldsymbol{x}) \quad (4)$$

当 $f(\boldsymbol{x})$ 为 l_1 范数时,式(4)即为众所周知的软阈值(soft thresholding):

$$Soft_{\lambda}(\boldsymbol{t}) = Prox_{\lambda\|\cdot\|_1}(\boldsymbol{t}) = \mathrm{sign}(\boldsymbol{t})\max\{|\boldsymbol{t}|-\lambda, 0\} \quad (5)$$

但是当矩阵 $\boldsymbol{A}$ 处在病态条件下,IST 算法在解决问题(2)~(3)时会变得很慢。为了克服这一点,许多加速版本的 IST 算法被提出来了。例如,TwIST(Two step IST)[5] 中每一次的迭代取决于之前的两次迭代,而在 IST 算法中,其只取决于之前一次的迭代。在许多基于小波和全方差(Total Variation,TV)的图像恢复问题中,它比 IST 算法在计算速度上有了很大幅度的提高。另外一种经典的快速 IST 算法(Fast IST Algorithm,FISTA)[6] 也在速度上比 IST 算法有明显优势。类似于 TwIST,FISTA 也是一种两步版本的 IST 算法,并且是 Nesterov 优化梯度算法(针对光滑凸问题)[7] 的非光滑变体。还有一种 IST 算法的变体——SpaRSA 算法[8] 通过每一步选择不同的步长来提高算法的速度。

近几年来,除了以上的 IST 系列算法,交替方向乘子法(Alternating Directions Method of Multipliers,ADMM)[9] 受到很多学者的关注。ADMM 融合了变量分裂技术[10] 和增广拉格朗日方法[11],其与最近被提出来的 Bregman 迭代方法[12-13] 有着紧密的联系。作为典型的 ADMM 算法,文献[14]针对非约束优化问题,提出了一种分裂增广拉格朗日收缩算法(Split Augmented Lagrangian Shrinkage Algorithm,SALSA),在图像去卷积、图像恢复以及核磁共振成像中,SALSA 的表现都优于 FISTA、TwIST 和 SpaRSA。

尽管 SALSA 相对于其他的算法表现出了很大的优势,但是它在具体实施中怎样选择惩罚参数是一个难题。事实上,SALSA 的结果对惩罚参数的选择非常敏感,为了达到满意的结果,在实施中不得不做大量的尝试以选择最合适的惩罚参数。本文正是为了克服这一难题,利用惩罚参数自适应选择原理[15],提出了一种稳健交替方向乘子法,它对惩罚参数的选择具有非常好的稳健性,并且其在各项评价指标上也优于 SALSA 和其他算法。

1　ADMM 算法框架

1.1　优化问题和 ADMM 算法

首先考虑下面的非约束优化问题:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + g(\boldsymbol{B}\boldsymbol{x}) \quad (6)$$

其中,$\boldsymbol{B}\in\mathbb{R}^{M\times N}$,$x\in\mathbb{R}^N$,$f:\mathbb{R}^N\to\overline{\mathbb{R}}$ 和 $g:\mathbb{R}^M\to\overline{\mathbb{R}}$。变量分裂技术就是通过约束 $\boldsymbol{v}=\boldsymbol{B}\boldsymbol{x}$ 来增加新的变量 $\boldsymbol{v}\in\mathbb{R}^M$,使式(6)变成一个约束优化问题:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + g(\boldsymbol{v}) \text{ s. t. } \boldsymbol{v}=\boldsymbol{B}\boldsymbol{x} \quad (7)$$

而 ADMM[9] 算法就是通过交替优化由式(7)生成的增广拉格朗日方程得到的,具体见算法 1。

算法 1　ADMM 算法

Alg. 1　ADMM algorithm

Step 1:设 $k=0$,选择 $\mu>0$,$\boldsymbol{v}_0$ 和 $\boldsymbol{d}_0$

Step 2:$\boldsymbol{x}_{k+1}\in\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{B}\boldsymbol{x}-\boldsymbol{v}_k-\boldsymbol{d}_k\|^2$

Step 3:$\boldsymbol{v}_{k+1}\in\arg\min_{\boldsymbol{v}} g(\boldsymbol{v}) + \frac{\mu}{2}\|\boldsymbol{B}\boldsymbol{x}_{k+1}-\boldsymbol{v}-\boldsymbol{d}_k\|^2$

Step 4:$\boldsymbol{d}_{k+1}=\boldsymbol{d}_k-(\boldsymbol{B}\boldsymbol{x}_{k+1}-\boldsymbol{v}_{k+1})$

Step 5:$k=k+1$

Step 6:如果满足停止条件就停止,否则转 Step 2

1.2　ADMM 收敛性定理

对于问题(7)和相应的 ADMM 算法(算法 2),Eckstein 和 Bertsekas[16] 给出了收敛性定理:

定理 1[14,16]　针对问题(7),对于任意 $\mu>0$,$\boldsymbol{v}_0,\boldsymbol{d}_0\in\mathbb{R}^M$,假设下面四个假设成立:

假设 1:$\boldsymbol{B}\in\mathbb{R}^{M\times N}$ 是列满秩矩阵;

假设 2:f,g 是正常的闭凸函数;

假设 3:两个序列 $(\boldsymbol{\zeta}_k)_{k\in\mathbf{N}}\subseteq[0,+\infty)$ 和 $(\boldsymbol{\eta}_k)_{k\in\mathbf{N}}\subseteq[0,+\infty)$ 分别满足 $\sum_k^{\infty}\boldsymbol{\zeta}_k<\infty$ 和 $\sum_k^{\infty}\boldsymbol{\eta}_k<\infty$;

假设 4:三个序列 $(\boldsymbol{x}_k)_{k\in\mathbf{N}}\subseteq\mathbb{R}^N$,$(\boldsymbol{v}_k)_{k\in\mathbf{N}}\subseteq\mathbb{R}^M$ 和 $(\boldsymbol{d}_k)_{k\in\mathbf{N}}\subseteq\mathbb{R}^M$ 满足:

$$\begin{cases}\left\|\boldsymbol{x}_{k+1}-\arg\min_{\boldsymbol{x}}\left(f(\boldsymbol{x})+\frac{\mu}{2}\|\boldsymbol{B}\boldsymbol{x}-\boldsymbol{v}_k-\boldsymbol{d}_k\|^2\right)\right\|\le\boldsymbol{\zeta}_k\\ \left\|\boldsymbol{v}_{k+1}-\arg\min_{\boldsymbol{y}}\left(g(\boldsymbol{v})+\frac{\mu}{2}\|\boldsymbol{B}\boldsymbol{x}_{k+1}-\boldsymbol{v}-\boldsymbol{d}_k\|^2\right)\right\|\le\boldsymbol{\eta}_k\\ \boldsymbol{d}_{k+1}=\boldsymbol{d}_k-(\boldsymbol{B}\boldsymbol{x}_{k+1}-\boldsymbol{v}_{k+1})\end{cases}$$

那么,如果问题(7)有解,序列 $(\boldsymbol{x}_k)_{k\in\mathbf{N}}$ 收敛到它的一个解;如果问题(7)无解,则两个序列 $(\boldsymbol{x}_k)_{k\in\mathbf{N}}$ 和 $(\boldsymbol{d}_k)_{k\in\mathbf{N}}$ 中至少一个发散。

1.3　优化条件分析

对于问题(7),其优化的充分必要条件如下。

假设 $\boldsymbol{x}^*$、$\boldsymbol{v}^*$ 是原始可行解,$\boldsymbol{d}^*$ 为对偶可行解,则对于原始可行性来说,有:

$$\boldsymbol{v}^*=\boldsymbol{B}\boldsymbol{x}^* \quad (8)$$

其对偶可行性有

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*) - \mu \boldsymbol{B}^{\mathrm{T}} \boldsymbol{d}^* \tag{9}$$

$$\mathbf{0} \in \partial g(\boldsymbol{v}^*) + \mu \boldsymbol{d}^* \tag{10}$$

当 f 和 g 都为可微分函数时,其次微分为它们的梯度,即 $\partial f = \{\nabla f\}$ 和 $\partial g = \{\nabla g\}$,则式(9)和式(10)为:

$$\mathbf{0} = \nabla f(\boldsymbol{x}^*) - \mu \boldsymbol{B}^{\mathrm{T}} \boldsymbol{d}^* \tag{11}$$

$$\mathbf{0} = \nabla g(\boldsymbol{v}^*) + \mu \boldsymbol{d}^* \tag{12}$$

由算法 1 中的 Step2, $\boldsymbol{x}_{k+1}$ 是 $\min\limits_{x} f(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{B}\boldsymbol{x} - \boldsymbol{v}_k - \boldsymbol{d}_k\|^2$ 的优化解,有:

$$\begin{aligned}
&\mathbf{0} \in \partial f(\boldsymbol{x}_{k+1}) + \mu \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{B}\boldsymbol{x}_{k+1} - \boldsymbol{v}_k - \boldsymbol{d}_k) \\
&= \partial f(\boldsymbol{x}_{k+1}) + \mu \boldsymbol{B}^{\mathrm{T}}[\underbrace{(\boldsymbol{d}_k - \boldsymbol{d}_{k+1} + \boldsymbol{v}_{k+1})}_{\text{由算法1中的Step4得到}} - \boldsymbol{v}_k - \boldsymbol{d}_k] \\
&= \partial f(\boldsymbol{x}_{k+1}) - \mu \boldsymbol{B}^{\mathrm{T}} \boldsymbol{d}_{k+1} + \mu \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{v}_{k+1} - \boldsymbol{v}_k)
\end{aligned} \tag{13}$$

同理, $\boldsymbol{v}_{k+1}$ 是 $\min\limits_{v} g(\boldsymbol{v}) + \frac{\mu}{2}\|\boldsymbol{B}\boldsymbol{x}_{k+1} - \boldsymbol{v} - \boldsymbol{d}_k\|^2$ 的优化解,因此

$$\begin{aligned}
&\mathbf{0} \in \partial g(\boldsymbol{v}_{k+1}) + \mu \underbrace{[\boldsymbol{d}_k - (\boldsymbol{B}\boldsymbol{x}_{k+1} - \boldsymbol{v}_{k+1})]}_{\text{算法1中的Step4}} \\
&= \partial g(\boldsymbol{v}_{k+1}) + \mu \boldsymbol{d}_{k+1}
\end{aligned} \tag{14}$$

分别比较式(10)和式(14),式(9)和式(13),不难发现 $\boldsymbol{v}_{k+1}$ 和 $\boldsymbol{d}_{k+1}$ 通常满足式(10),而式(9)是通过迭代满足的,即当 $\boldsymbol{v}_{k+1} - \boldsymbol{v}_k \to \mathbf{0}$ 时,式(13)→式(9)。因此

$$\boldsymbol{dr}_{k+1} = \mu \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{v}_{k+1} - \boldsymbol{v}_k) \tag{15}$$

被当成是式(13)的剩余值,也称为第 $k+1$ 迭代步时的对偶剩余值(dual residual)。而

$$\boldsymbol{pr}_{k+1} = \boldsymbol{B}\boldsymbol{x}_{k+1} - \boldsymbol{v}_{k+1} \tag{16}$$

被称为原始剩余值(primal residual)。

总之,ADMM 算法的优化条件是式(8)、式(9)和式(10)。式(10)在迭代过程中总是满足的,而原始和对偶剩余值: $\boldsymbol{pr}_{k+1}$ 和 $\boldsymbol{dr}_{k+1}$ 在迭代中收敛至 $\mathbf{0}$,分别使得式(8)和式(9)满足。

2 稳健 ADMM 算法框架

基于上面的 ADMM 算法框架,可以结合基于原始和对偶剩余值的惩罚参数 μ 自适应选择原理[15],得到稳健 ADMM(Robust ADMM, RADMM)算法。

2.1 惩罚参数自适应选择原理

为了加快 ADMM 算法的收敛速度,可以考虑每一个迭代中采用不同的惩罚参数 μ_k,这样也减少了算法对初始值的依赖。由 ADMM 算法的迭代过程可知, μ 越大,对远离初始可行解的数值惩罚越大,从而产生小的初始剩余值 $\boldsymbol{pr}_{k+1}$。但是 μ 越大,会使对偶剩余值 $\boldsymbol{dr}_{k+1}$ 增大;相反,如果 μ 越小,能减少 $\boldsymbol{dr}_{k+1}$,但是会增加 $\boldsymbol{pr}_{k+1}$。基于这些考虑,产生了惩罚参数自适应选择原理[15]:

$$\mu_{k+1} = \begin{cases} \alpha\mu_k & \|\boldsymbol{pr}_{k+1}\|_2 > \delta\|\boldsymbol{dr}_{k+1}\|_2 \\ \mu_k/\beta & \|\boldsymbol{dr}_{k+1}\|_2 > \delta\|\boldsymbol{pr}_{k+1}\|_2 \\ \mu_k & \text{其他} \end{cases} \tag{17}$$

其中, $\alpha > 1, \beta > 1, \delta > 1$。式(17)的目的就是用 δ 来捆绑 $\boldsymbol{pr}_{k+1}$ 和 $\boldsymbol{dr}_{k+1}$,使它们都收敛至 $\mathbf{0}$。例如,当 $\boldsymbol{pr}_{k+1}$ 相对于 $\boldsymbol{dr}_{k+1}$ 过大($\|\boldsymbol{pr}_{k+1}\|_2 > \delta\|\boldsymbol{dr}_{k+1}\|_2$),惩罚参数增加至 $\alpha\mu_k$;相反,当 $\boldsymbol{pr}_{k+1}$ 相对于 $\boldsymbol{dr}_{k+1}$ 过小($\|\boldsymbol{dr}_{k+1}\|_2 > \delta\|\boldsymbol{pr}_{k+1}\|_2$),惩罚参数减少至 μ_k/β。事实上,为了进一步减少计算,式(17)中的距离标量 2 - 范数: $\|\boldsymbol{pr}\|_2$ (和 $\|\boldsymbol{dr}\|_2$)可用 $|\boldsymbol{pr}|$ (和 $|\boldsymbol{dr}|$)来代替,后者被定义为矩阵各元素的绝对值之和,例如 $|\boldsymbol{pr}| = \sum\limits_i \sum\limits_j |\boldsymbol{pr}_{ij}|$,其类似于向量的 1 - 范数。

值得一提的是,在 ADMM 算法中惩罚参数 μ_{k+1} 调整的时候,根据 $d = \lambda/\mu$ 也需要及时更新 d_{k+1},即:

$$\boldsymbol{d}_{k+1} = \begin{cases} \boldsymbol{d}_k/\alpha & |\boldsymbol{pr}_{k+1}| > \delta|\boldsymbol{dr}_{k+1}| \\ \beta\boldsymbol{d}_k & |\boldsymbol{dr}_{k+1}| > \delta|\boldsymbol{pr}_{k+1}| \\ \boldsymbol{d}_k & \text{其他} \end{cases} \tag{18}$$

2.2 稳健 ADMM 算法

结合算法 1,式(15)、式(16)和式(18),可以得到基于惩罚参数自适应选择的 RADMM 算法,见算法 2。

算法 2 RADMM 算法

Alg. 2 RADMM algorithm

Step 1: 设 $k = 0$,选择 $\mu_0 > 0, \delta > 0, \alpha > 0, \beta > 0, \boldsymbol{v}_0$ 和 $\boldsymbol{d}_0$

Step 2: $\boldsymbol{x}_{k+1} \in \arg\min\limits_{x} f(\boldsymbol{x}) + \frac{\mu_k}{2}\|\boldsymbol{B}\boldsymbol{x} - \boldsymbol{v}_k - \boldsymbol{d}_k\|^2$

Step 3: $\boldsymbol{v}_{k+1} \in \arg\min\limits_{v} g(\boldsymbol{v}) + \frac{\mu_k}{2}\|\boldsymbol{B}\boldsymbol{x}_{k+1} - \boldsymbol{v} - \boldsymbol{d}_k\|^2$

Step 4: $\bar{\boldsymbol{d}}_{k+1} = \boldsymbol{d}_k - (\boldsymbol{B}\boldsymbol{x}_{k+1} - \boldsymbol{v}_{k+1})$

Step 5: $\boldsymbol{pr}_{k+1} = \boldsymbol{B}\boldsymbol{x}_{k+1} - \boldsymbol{v}_{k+1}$

Step 6: $\boldsymbol{dr}_{k+1} = \mu_k \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{v}_{k+1} - \boldsymbol{v}_k)$

Step 7: 判定: 如果 $|\boldsymbol{pr}_{k+1}| > \delta|\boldsymbol{dr}_{k+1}|$

Step 8: $\mu_{k+1} = \alpha\mu_k$

Step 9: $\boldsymbol{d}_{k+1} = \bar{\boldsymbol{d}}_{k+1}/\alpha$

Step 10: 否则如果 $|\boldsymbol{dr}_{k+1}| > \delta|\boldsymbol{pr}_{k+1}|$

Step 11: $\mu_{k+1} = \mu_k/\beta$

Step 12: $\boldsymbol{d}_{k+1} = \beta\bar{\boldsymbol{d}}_{k+1}$

Step 13: 否则

Step 14: $\mu_{k+1} = \mu_k$

Step 15: $\boldsymbol{d}_{k+1} = \bar{\boldsymbol{d}}_{k+1}$

Step 16: 结束判定

Step 17: $k = k + 1$

Step 18: 如果满足停止条件就停止,否则转 Step 2

2.3　收敛性分析

为了证明 RADMM 收敛,可把 RADMM 的算法过程看作是两个部分组成:①运行 RADMM 有限个迭代;②接下来运行 SALSA。事实上,在 RADMM 的实施中,由于惩罚参数自适应选择,原始和对偶剩余值随着迭代过程将会相互趋近,在一定的迭代步骤后,一般最后会出现:$|\boldsymbol{pr}_{k+1}|/\delta < |\boldsymbol{dr}_{k+1}| < \delta\ |\boldsymbol{pr}_{k+1}|$(或 $|\boldsymbol{dr}_{k+1}|/\delta < |\boldsymbol{pr}_{k+1}| < \delta|\boldsymbol{dr}_{k+1}|$),此时 $\mu_{k+1}=\mu_k$,就退化为 SALSA 算法了。注意,以上并没有说明当 RADMM 算法中一定出现 $\mu_{k+1}=\mu_k$,就会从此退化成 SALSA 算法,因此在下一个迭代中有可能满足使 μ_{k+1} 增大或缩小的条件。但是,由 2.1 节的分析可知,正是由于对惩罚参数的自适应控制,使算法能收敛得更快,使原始和对偶剩余值同步和同水平减少,提高了结果的精度。

无论如何,RADMM 都可以看作是:在前面有限迭代中通过调整惩罚参数 μ 来控制原始和对偶剩余值,然后停止使用惩罚参数自适应选择,从而退化成 SALSA 算法。基于此,如果 SALSA 收敛,则 RADMM 收敛。而 1.2 节保证了 SALSA 的收敛,从而保证了 RADMM 的收敛。下面将利用 RADMM 进行图像恢复。

3　基于 Parseval 紧框架的图像恢复

这一部分采用式(3)作为优化模型对图像进行修复,并进一步假设 $\boldsymbol{A}$ 为块轮换卷积矩阵(block-circulant convolution matrix),$\boldsymbol{W}$ 为标准化紧框架矩阵(即 Parseval 紧框架),则有:

$$\min_{x}\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{AWs}\|_2^2+\lambda\|\boldsymbol{s}\|_1 \text{ s. t. } \boldsymbol{WW}^{\mathrm{H}}=\boldsymbol{I} \tag{19}$$

其中,$\boldsymbol{W}^H$ 为 $\boldsymbol{W}$ 的共轭转置,$\boldsymbol{I}$ 为单位矩阵。对于问题(19),结合问题(6)和算法 2,设置:

$$\begin{cases} f(\boldsymbol{s})=\dfrac{1}{2}\|\boldsymbol{y}-\boldsymbol{AWs}\|_2^2 \\ g(\boldsymbol{v})=\lambda\|\boldsymbol{v}\|_1 \\ \boldsymbol{B}=\boldsymbol{I} \end{cases} \tag{20}$$

对于式(20),算法 2 中的 Step 2 ~3 变为:

$$\boldsymbol{s}_{k+1}=\arg\min_{\boldsymbol{s}}\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{AWs}\|_2^2+\frac{\mu_k}{2}\|\boldsymbol{x}-\boldsymbol{v}_k-\boldsymbol{d}_k\|^2 \tag{21}$$

$$\boldsymbol{v}_{k+1}=\arg\min_{\boldsymbol{v}}\lambda\|\boldsymbol{v}\|_1+\frac{\mu_k}{2}\|\boldsymbol{s}_{k+1}-\boldsymbol{v}-\boldsymbol{d}_k\|^2 \tag{22}$$

简单分析可知,在式(21)中,通过设置目标函数的导数为 $\boldsymbol{0}$,可以得到:

$$\boldsymbol{s}_{k+1}=(\boldsymbol{W}^{\mathrm{H}}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{AW}+\mu_k\boldsymbol{I})^{-1}\boldsymbol{\omega}_k \tag{23}$$

其中,$\boldsymbol{\omega}_k=\boldsymbol{W}^{\mathrm{H}}\boldsymbol{A}^{\mathrm{H}}y+\mu_k(\boldsymbol{v}_k+\boldsymbol{d}_k)$。而式(22)是软阈值(见式(5)):

$$\boldsymbol{v}_{k+1}=Soft_{\lambda/\mu_k}(\boldsymbol{s}_{k+1}-\boldsymbol{d}_k) \tag{24}$$

在式(23)中,根据 Sherman-Morrison-Woodbury 矩阵求逆公式,有$(\boldsymbol{W}^{\mathrm{H}}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{AW}+\mu_k\boldsymbol{I})^{-1}=\frac{1}{\mu_k}[\boldsymbol{I}-\boldsymbol{W}^{\mathrm{H}}\boldsymbol{A}^{\mathrm{H}}(\boldsymbol{AA}^{\mathrm{H}}+\mu_k\boldsymbol{I})^{-1}\boldsymbol{AW}]$,又由于 $\boldsymbol{A}$ 是块轮换矩阵,因此其能被分解为:

$$\boldsymbol{A}=\boldsymbol{U}^{\mathrm{H}}\boldsymbol{DU} \tag{25}$$

其中:$\boldsymbol{U}$ 为酉(unitary)矩阵(即有 $\boldsymbol{U}^{\mathrm{H}}\boldsymbol{U}=\boldsymbol{UU}^{\mathrm{H}}=\boldsymbol{I}$ 或 $\boldsymbol{U}^{\mathrm{H}}=\boldsymbol{U}^{-1}$),代表了二维离散傅里叶变换(Discrete Fourier Transform, DFT),通过快速傅里叶(Fast Fourier Transform, FFT)算法来计算 $\boldsymbol{U}$ 或 $\boldsymbol{U}^{\mathrm{H}}$ 的计算复杂度为 $O(N\log N)$;$\boldsymbol{D}$ 为对角矩阵,其对角元素即为 $\boldsymbol{A}$ 的 DFT 系数。因此,

$$(\boldsymbol{AA}^{\mathrm{H}}+\mu_k\boldsymbol{I})^{-1}=[\boldsymbol{U}^{\mathrm{H}}\boldsymbol{DU}(\boldsymbol{U}^{\mathrm{H}}\boldsymbol{DU})^{\mathrm{H}}+\mu_k\boldsymbol{I}]^{-1}=(\boldsymbol{U}^{\mathrm{H}}\boldsymbol{D}^{\mathrm{H}}\boldsymbol{DU}+\mu_k\boldsymbol{U}^{\mathrm{H}}\boldsymbol{U})^{-1}=\boldsymbol{U}^{\mathrm{H}}(|\boldsymbol{D}|^2+\mu_k\boldsymbol{I})^{-1}\boldsymbol{U} \tag{26}$$

其中:$|\boldsymbol{D}|^2$ 是把对角矩阵 $\boldsymbol{D}$ 所有的元素取绝对值的平方,因此也为一个对角矩阵。则 $|\boldsymbol{D}|^2+\mu_k\boldsymbol{I}$ 为对角矩阵,所以很容易得到$(|\boldsymbol{D}|^2+\mu_k\boldsymbol{I})^{-1}$(只需把 $|\boldsymbol{D}|^2+\mu_k\boldsymbol{I}$ 的对角元素取倒数,其计算复杂度为 $O(N)$)。综合得到:

$$\boldsymbol{s}_{k+1}=\frac{1}{\mu_k}[\boldsymbol{I}-\boldsymbol{W}^{\mathrm{H}}\boldsymbol{U}^{\mathrm{H}}\boldsymbol{D}^{\mathrm{H}}(|\boldsymbol{D}|^2+\mu_k\boldsymbol{I})^{-1}\boldsymbol{DUW}]\boldsymbol{\omega}_k \tag{27}$$

如果直接用 FFT 算法来表示式(27),则有:

$$\boldsymbol{s}_{k+1}=\frac{1}{\mu_k}\left\{\boldsymbol{\omega}_k-\boldsymbol{W}^{\mathrm{H}}\left[F^{-1}\left(\cdot\frac{|F(\boldsymbol{A})|\circ|F(\boldsymbol{A})|}{|F(\boldsymbol{A})|\circ|F(\boldsymbol{A})|+\mu_k}\circ F(\boldsymbol{W}\boldsymbol{\omega}_k)\right)\right]\right\} \tag{28}$$

其中:"$\circ$"和"$\cdot$—"分别表示点乘和点除,即对应的元素相乘和相除;F 表示 FFT 算法,则F^{-1}表示反 FFT 算法。

值得一提的是,由于采取的是设置方案(20),因此在算法 2 中相应的 $\boldsymbol{B}=\boldsymbol{I}$。算法 2 中计算复杂度最大的两步是 Step 2 和 Step 3,为 $O(N\log N)$,其他迭代步骤的计算复杂度为 $O(N)$ 或 $O(1)$。因此,总体计算复杂度为 $O(N\log N)$。

4　实验和结果

本实验中,对于模型(1),原始图像 $\boldsymbol{x}$ 选用图像"Lena"(见图 1,512 × 512 像素),$\boldsymbol{A}$ 对应 9 × 9 的均匀模糊算子,其通过构造相应的循环矩阵并进行 FFT 变换得到,$\boldsymbol{n}$ 为高斯噪声。$\boldsymbol{y}$ 为模糊和加噪声后的观测图像,例如,信噪比(Signal to Noise Ratio, SNR)为 40 dB 的模糊图片见图 2。

图1 原始图像
Fig. 1 Original image

图2 观测图像
Fig. 2 Observed image

下面对 RADMM 和目前热门的算法(重点是 SALSA)从不同角度进行比较。所采取的主要评价指标有:时间、迭代数、均方差(Mean Square Error, MSE)和改进信噪比(Improved SNR, ISNR),其定义为 $ISNR = 10\lg_{10}\dfrac{\|\boldsymbol{x}-\boldsymbol{y}\|_F^2}{\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|_F^2}$,其中 $\boldsymbol{x}$ 为原始图像,$\boldsymbol{y}$ 为观测图像,$\tilde{\boldsymbol{x}}$ 表示算法得到的重构图像,$\|\boldsymbol{H}\|_F$ 表示矩阵 $\boldsymbol{H}$ 的 Frobenius 范数,定义为 $\|\boldsymbol{H}\|_F = \sqrt{\sum_{ij}|\boldsymbol{H}_{ij}|^2}$。

4.1 比较 RADMM 和其他算法

RADMM 的特定参数设置为 $\delta = 10$,$\alpha = \beta = 2$。其和 SALSA[14] 中的停止条件可设定为 $Tolerance = \dfrac{|Objective(\boldsymbol{s}_k) - Objective(\boldsymbol{s}_{k-1})|}{Objective(\boldsymbol{s}_k)} \leqslant 0.001$,其中 $Objective(\boldsymbol{s}_k) = \dfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{AWs}_k\|_2^2 + \lambda\|\boldsymbol{s}_k\|_1$。为了方便比较算法,算法 TwIST[5]、SpaRSA[8] 和 FISTA[6] 的停止条件为其目标函数值 $Objective \leqslant Objective_{\mathrm{SALSA}}(\boldsymbol{s}_k)$,其中 $Objective_{\mathrm{SALSA}}(\boldsymbol{s}_k)$ 为算法 SALSA 停止时的目标函数值,并设置算法的最大迭代步数不超过 500($Iterations \leqslant 500$)。正则化参数 λ 的选取则是基于实验效果(如 ISNR 最好),一般为 0.01 左右。经过不同的尝试,当 $\mu = 0.001$ 时,RADMM、SALSA、TwIST、SpaRSA 和 FISTA 都达到了算法本身所能达到的较好结果,当 SNR 为 40 dB 时,不同算法重构的图像如图 3 ~ 7 所示。当 SNR 分别为 40 dB、35 dB 和 30 dB 时,运算时间、迭代数、MSE 和 ISNR 的对比如表 1 所示。

图3 RADMM 重构图像
Fig. 3 Reconstructed image by RADMM

图4 SALSA 重构图像
Fig. 4 Reconstructed image by SALSA

图5 TwIST 重构图像
Fig. 5 Reconstructed image by TwIST

图6 SpaRSA 重构图像
Fig. 6 Reconstructed image by SpaRSA

图7 FISTA 重构图像
Fig. 7 Reconstructed image by FISTA

表1 不同算法结果比较

Tab. 1 The comparison among algorithms

算法	SNR	时间/s	迭代数	MSE	ISNR/dB
RADMM	40	15.2	8	38	6.49
	35	15.3	8	44.8	5.78
	30	16.7	9	80.3	3.29
SALSA	40	30.7	17	45.7	5.68
	35	30.5	17	97.3	2.41
	30	32.1	19	159.3	0.21
TwIST	40	290	352	43.1	5.94
	35	313	361	95.8	2.53
	30	318	362	154.2	0.25
SpaRSA	40	>354	>500	40.3	6.23
	35	>356	>500	92.7	2.86
	30	>354	>500	142.2	0.43
FISTA	40	>259	>500	59.7	4.52
	35	>259	>500	112.4	1.98
	30	>258	>500	173.2	0.11

从表 1 可以看出,在不同噪声水平下,对于指标:时间、迭代数、MSE 和 ISNR,RADMM 都好于 SALSA,并远远优于其他热门的算法。

4.2 比较 RADMM 和 SALSA

4.2.1 改进信噪比和均方差

SALSA 算法具体实施的一个难点就是选择

合适的惩罚参数 μ,这也给算法的结果带来很多不确定性。而 RADMM 则很好地克服了这个问题。为了验证这一点,在 4.1 节的实验设置中改变 μ 从 10^{-6} 至 10^{4},其他设置不变。对图像"Lena"实验得到的结果如图 8 ~9 所示。

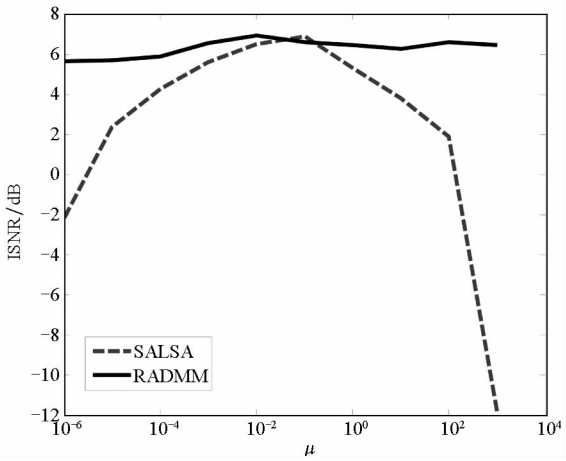

图 8　不同 μ 下的 RADMM 和 SALSA 所得 ISNR 对比

Fig. 8　ISNRs obtained by RADMM and SALSA over μ

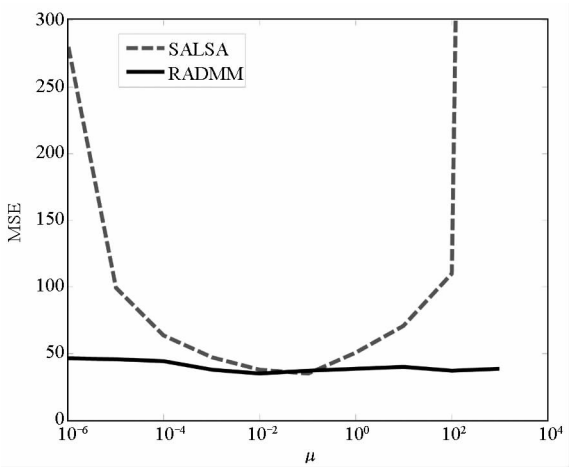

图 9　不同 μ 下的 RADMM 和 SALSA 所得 MSE 对比

Fig. 9　MSEs obtained by RADMM and SALSA over μ

从图 8 可以看出,SALSA 得到的 ISNR 对 μ 的选择非常敏感,当 $\mu = 0.01$ 左右时,SALSA 取得较好的 ISNR(这也是 4.1 节设置 μ 为 0.01 的原因),但当随着 μ 减少或增大时,ISNR 急剧减少,因此 SALSA 需要不断地尝试 μ 才能得到满意的结果。与之形成对比的是,RADMM 得到的 ISNR 对 μ 的选择很稳健,所以在使用 RADMM 时,对 μ 的选择相对随意。

同理,依照图 9,SALSA 得到的 MSE 对 μ 的选择非常敏感,当 $\mu = 0.01$ 左右时,SALSA 得到较小的 MSE,而当随着 μ 减少或增大时,MSE 急剧增大;但是 RADMM 却表现出了相当好的稳健性。

4.2.2 原始剩余值和对偶剩余值

除了把停止条件只限定为迭代数 *Iterations* ≤ 500,其他的实验设置和 4.1 节相同。对图像"Lena"实验得到的原始和对偶剩余值的演变过程如图 10 所示。注意,由于 SALSA 只是 RADMM 的一个特殊情况,因此 SALSA 的原始剩余值 ***pr*** 和对偶剩余值 ***dr*** 可由在 RADMM 中设置 $\alpha = \beta = 1$ 得到。RADMM 中原始和对偶剩余值分别为 $\boldsymbol{pr}_{k+1} = \boldsymbol{s}_{k+1} - \boldsymbol{v}_{k+1}$ 和 $\boldsymbol{dr}_{k+1} = \mu(\boldsymbol{v}_{k+1} - \boldsymbol{v}_k)$,而 SALSA 中的原始和对偶剩余值也可这样得到。

从图 10 不难看出,随着迭代的进行,RADMM 使 ***pr*** 和 ***dr*** 收敛到同一水平。根据 2.1 节的分析,这非常有利于算法快速地收敛到全局最优解。而 SALSA 中由于没有在迭代过程中对惩罚参数 μ 进行控制,导致 ***pr*** 和 ***dr*** 各行其道,不利于其收敛到全局最优解,这也是 4.1 节所显示出 SALSA 劣于 RADMM 的原因。

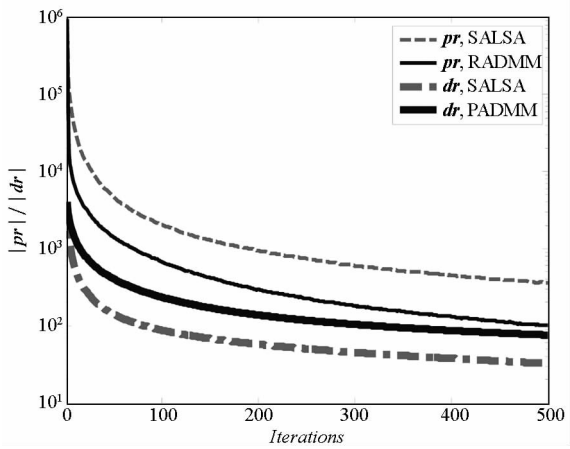

图 10　RADMM 和 SALSA 中 ***pr*** 和 ***dr*** 的演变过程

Fig. 10　Evolution of ***pr*** and ***dr*** over iterations

4.2.3　迭代扰动

RADMM 和 SALSA 中的第 k 步迭代扰动被定义为:

$$violations_k = \frac{\|\boldsymbol{v}_k - \boldsymbol{s}_k\|_F}{\sqrt{\|\boldsymbol{v}_k\|_F^2 + \|\boldsymbol{s}_k\|_F^2}}$$

从定义可以看出,迭代扰动衡量的是每一个迭代中,不同分裂变量之间的差异。其一定程度上反映了算法的收敛状况,因为在算法的不断迭代过程中,不同的分裂变量会最终收敛到全局最优解。这里仍然设定算法停止条件为 *Iterations* ≤500,并取 $\mu = 0.1$、0.01 和 0.001,则相应的 RADMM 和 SALSA 的迭代扰动如图 11 所示。

分析图 11 可以得出:RADMM 的迭代扰动总是低于 SALSA,并且随着 μ 的减少(增大),这两

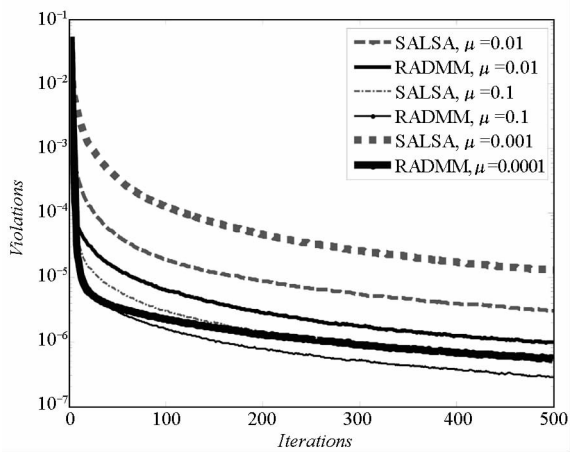

图 11 RADMM 和 SALSA 中迭代扰动的演变过程

Fig. 11 Evolution of violations over iterations

种算法迭代扰动的差距增大(减少)。这也从侧面反映出 RADMM 性能上的优势。

5 结论

为了取得良好的效果,交替方向乘子法在实际应用中不得不做大量的尝试,以选择合适的惩罚参数。针对其对惩罚参数这一较差的稳健性,提出了稳健交替方向乘子法。正是采用了惩罚参数自适应选择原理,对原始和对偶剩余值进行绑定,使两者同步和同水平减少,不但使收敛加快,并且提高了结果的精度。在基于紧框架的图像恢复应用中,稳健交替方向乘子法不但对惩罚参数的选择表现出良好的稳健性,而且在时间、迭代数、MSE 和 ISNR 上都优于 SALSA,尤其在前两项指标上远远优于其他目前热门的算法 TwIST、SpaRSA 和 FISTA。

尽管稳健交替方向乘子法表现出了很多优势,但是它只适合两项优化问题,例如类似于式(2)和式(3)这样的问题:一项是数据保真项,另一项是正则化项。因此,怎么将稳健交替方向乘子法框架扩展到三项甚至更多项的优化问题将是以后研究的内容。

参考文献(References)

[1] Donoho D L. Compressed sensing[J]. IEEE Transactions on Information Theory, 2006, 52(4): 1289 – 1306.

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

[3] Elad M, Milanfar P, Rubinstein R. Analysis versus synthesis in signal priors[J]. Inverse Problems, 2007, 23(3): 947 – 968.

[4] Mallat S. A wavelet tour of signal processing[M]. New York, USA: Academic Press, 2009.

[5] Bioucas-Dias J M, Figueiredo M A T. A new TwIST: two-step iterative shrinkage/thresholding algorithms for image restoration[J]. IEEE Transactions on Image Processing, 2007, 16(12): 2992 – 3004.

[6] Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems[J]. SIAM Journal on Imaging Sciences, 2009, 2(1): 183 – 202.

[7] Nesterov Y. Introductory lectures on convex optimization[M]. Boston, MA, USA: Springer, 2004.

[8] Wright S J, Nowak R D, Figueiredo M A T. Sparse reconstruction by separable approximation[J]. IEEE Transactions Signal Processing, 2009, 57(7): 2479 – 2493.

[9] Boyd S, Parikh N, Chu E, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers[J]. Foundations and Trends in Machine Learning, 2011, 3(1): 1 – 122.

[10] Courant R. Variational methods for the solution of problems with equilibrium and vibration[J]. Bulletin of the American Mathematical Society, 1943, 49: 1 – 23.

[11] Bertsekas D P. Nonlinear programming[M]. 2nd ed. USA: Athena Scientific, 1992.

[12] Yin W, Osher S, Goldfarb D, et al. Bregman iterative algorithms for l1 minimization with applications to compressed sensing[J]. SIAM Journal on Imaging Science, 2008, 1(1): 143 – 168.

[13] Goldstein T, Osher S. The split Bregman method for l1 regularized problems[J]. SIAM Journal on Imaging Sciences, 2009, 2(2): 323 – 343.

[14] Afonso M V, Bioucas-dias J M, Figueiredo M A T. Fast image recovery using variable splitting and constrained optimization[J]. IEEE Transactions on Image Processing, 2009, 19(9): 2345 – 2356.

[15] He B S, Yang H, Wang S L. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities[J]. Journal of Optimization Theory and Applications, 2000, 106(2): 337 – 356.

[16] Eckstein J, Bertsekas D P. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators[J]. Mathematical Programming, 1992, 55(3): 293 – 318.