The Split Bregman Method for $L_1$ Regularized Problems: An Overview

Pardis Noorzad$^1$

$^1$Department of Computer Engineering and IT
Amirkabir University of Technology

Khordad 1389
1 Introduction

2 TV Denoising
   The ROF Model
   Iterated Total Variation

3 $L_1$ Regularization
   Easy vs. Hard Problems

4 Split Bregman Method
   Split Bregman Formulation
   Bregman Iteration
   Applying SB to TV Denoising

5 Results
   Fast Convergence
   Acceptable Intermediate Results
Image Restoration and Variational Models

- Fundamental problem in image restoration: **denoising**
- Denoising is an important step in machine vision tasks
- Concern is to **preserve** important image features
  - edges, texture
  while **removing** noise
- Variational models have been very successful
TV-based Image Restoration

- **Total variation based** image restoration models first introduced by Rudin, Osher, and Fatemi [ROF92]
- An early example of PDE based **edge preserving** denoising
- Has been extended and solved in a variety of ways
- Here, the **Split Bregman** method is introduced
## Denoising

### Decomposition

\[ f = u + v \]

- \( f : \Omega \rightarrow \mathbb{R} \) is the noisy image
- \( \Omega \) is the bounded open subset of \( \mathbb{R}^2 \)
- \( u \) is the true signal
- \( v \sim N(0, \sigma^2) \) is the white Gaussian noise
Conventional Variational Model
Easy to solve — results are disappointing

\[
\min \int_{\Omega} (u_{xx} + u_{yy})^2 dx
dy
\]

such that

\[
\int_{\Omega} ud\Omega = \int_{\Omega} f d\Omega
\]

(white noise is of zero mean)

\[
\int_{0}^{1} \frac{1}{2} (u - f)^2 dx
dy = \sigma^2,
\]

(a priori information about \( v \))
The ROF Model

Difficult to solve — successful for denoising

\[
\min_{u \in BV(\Omega)} \left\{ \|u\|_{BV} + \lambda \|f - u\|_2^2 \right\}
\]

- \( \lambda > 0 \): scale parameter
- \( BV(\Omega) \): space of functions with \textbf{bounded variation} on \( \Omega \)
- \( \| . \| \): \textbf{BV seminorm} or \textbf{total variation} given by,
  \[
  \|u\|_{BV} = \int_{\Omega} |\nabla u|
  \]
The ROF Model

BV seminorm

- It’s use is essential — allows image recovery with edges
- What if first term were replaced by \( \int_{\Omega} |\nabla u|^p \)?
  - Which is both differentiable and strictly convex
- No good! For \( p > 1 \), its derivative has smoothing effect in the optimality condition
- For TV however, the operator is degenerate, and affects only level lines of the image
Iterative Regularization

Adding back the noise

- In the ROF model, $u - f$ is treated as error and discarded
- In the decomposition of $f$ into signal $u$ and additive noise $v$
  - There exists some signal in $v$
  - And some smoothing of textures in $u$
- Osher et al. [OBG+05] propose an iterated procedure to add the noise back
Iterative Regularization

The iteration

**Step 1:** Solve the ROF model to obtain:

\[
    u_1 = \arg \min_{u \in BV(\Omega)} \left\{ \int |\nabla u| + \lambda \int (f - u)^2 \right\}
\]

**Step 2:** Perform a correction step:

\[
    u_2 = \arg \min_{u \in BV(\Omega)} \left\{ \int |\nabla u| + \lambda \int (f + v_1 - u)^2 \right\}
\]

\(v_1\) is the noise estimated by the first step, \(f = u_1 + v_1\)
Definition

$L_1$ regularized optimization

$$
\min_u \|\Phi(u)\|_1 + H(u)
$$

- Many important problems in imaging science (and other problems in engineering) can be posed as $L_1$ regularized optimization problems

- $\|\cdot\|_1$: the $L_1$ norm

- both $\|\Phi(u)\|_1$ and $H(u)$ are convex functions
Easy vs. Hard Problems

**Easy Instances**

\[
\arg\min_u \|Au - f\|^2_2 \quad \text{differentiable}
\]

\[
\arg\min_u \|u\|_1 + \|u - f\|^2_2 \quad \text{solvable by shrinkage}
\]
Shrinkage
or Soft Thresholding

Solves the $L_1$ problem of the form ($H(.)$ is convex and differentiable):

$$\arg \min_u \mu \|u\|_1 + H(u)$$

Based on this iterative scheme

$$u^{k+1} \rightarrow \arg \min_u \mu \|u\|_1 + \frac{1}{2\delta^k} \|u - (u^k - \delta^k \nabla H(u^k))\|^2$$
Shrinkage
Continued

Since unknown $u$ is componentwise separable, each component can be independently obtained:

$$u^{k+1}_i = \text{shrink}((u^k - \delta^k \nabla H(u^k))_i, \mu \delta^k), \; i = 1, \ldots, n,$$

$$\text{shrink}(y, \alpha) := \text{sgn}(y) \max\{|y| - \alpha, 0\} = \begin{cases} y - \alpha, & y \in (\alpha, \infty), \\ 0, & y \in [-\alpha, \alpha], \\ y + \alpha, & y \in (-\infty, -\alpha). \end{cases}$$
Hard Instances

\[
\begin{align*}
\text{arg min}_u & \quad \|\Phi(u)\|_1 + \|u - f\|^2_2 \\
\text{arg min}_u & \quad \|u\|_1 + \|Au - f\|^2_2 
\end{align*}
\]

What makes these problems hard?
The coupling between the $L_1$ and $L_2$ terms.
Split the $L_1$ and $L_2$ components

To solve the general regularization problem:

$$\arg\min_u \|\Phi(u)\|_1 + H(u)$$

Introduce $d = \Phi(u)$ and solve the constrained problem

$$\arg\min_{u,d} \|d\|_1 + H(u) \text{ such that } d = \Phi(u)$$
Split the $L_1$ and $L_2$ components

Add an $L_2$ penalty term to get an unconstrained problem

$$\arg \min_{u,d} \|d\|_1 + H(u) + \frac{\lambda}{2} \|d - \Phi(u)\|^2$$

• Obvious way is to use the penalty method to solve this

• However, as $\lambda_k \to \infty$, the condition number of the Hessian approaches infinity, making it impractical to use fast iterative methods like Conjugate Gradient to approximate the inverse of the Hessian.
The optimization problem is solved by iterating

\[(u^{k+1}, d^{k+1}) = \arg \min_{u,d} \|d\|_1 + H(u) + \frac{\lambda}{2}\|d - \Phi(u) - b^k\|^2\]

\[b^{k+1} = b^k + (\Phi(u) - d^k)\]

The iteration in the first line can be done separately for \(u\) and \(d\).
3-step Algorithm

Step 1: \( u^{k+1} = \arg \min_u \ H(u) + \frac{\lambda}{2} \|d^k - \Phi(u) - b^k\|_2^2 \)

Step 2: \( d^{k+1} = \arg \min_d \|d\|_1 + \frac{\lambda}{2} \|d^k - \Phi(u) - b^k\|_2^2 \)

Step 3: \( b^{k+1} = b^k + \Phi(u^{k+1}) - d^{k+1} \)

- Step 1 is now a differentiable optimization problem, we'll solve with **Gauss Seidel**
- Step 2 can be solved efficiently with shrinkage
- Step 3 is an explicit evaluation
Anisotropic TV

\[
\arg\min_u |\nabla_x u| + |\nabla_y u| + \frac{\mu}{2} \|u - f\|_2^2
\]
Anisotropic TV

The steps

Step 1: \( u^{k+1} = G(u^k) \)

Step 2: \( d_x^{k+1} = \text{shrink}(\nabla_x u^{k+1} + b_x^k, \frac{1}{\lambda}) \)

Step 3: \( d_y^{k+1} = \text{shrink}(\nabla_y u^{k+1} + b_y^k, \frac{1}{\lambda}) \)

Step 4: \( b_x^{k+1} = b_x^k + (\nabla_x u - x) \)

Step 5: \( b_y^{k+1} = b_y^k + (\nabla_y u - y) \)

- \( G(u^k) \): result of one Gauss-Seidel sweep for the corresponding \( L_2 \) optimization

- This algorithm is cheap — each step is a few operations per pixel
**Isotropic TV**

With similar steps

\[
\arg \min_u \sum_i \sqrt{(\nabla_x u)_i^2 + (\nabla_y u)_i^2} + \frac{\mu}{2} \|u - f\|^2_2
\]
Split Bregman is fast

Intel Core 2 Duo desktop (3 GHz), compiled with g++

<table>
<thead>
<tr>
<th>Anisotropic</th>
<th>Time/cycle (sec)</th>
<th>Time Total (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256 × 256 Blocks</td>
<td>0.0013</td>
<td>0.068</td>
</tr>
<tr>
<td>512 × 512 Lena</td>
<td>0.0054</td>
<td>0.27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Isotropic</th>
<th>Time/cycle (sec)</th>
<th>Time Total (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256 × 256 Blocks</td>
<td>0.0018</td>
<td>0.0876</td>
</tr>
<tr>
<td>512 × 512 Lena</td>
<td>0.011</td>
<td>0.55</td>
</tr>
</tbody>
</table>
Split Bregman is fast
Compared to Graph Cuts

<table>
<thead>
<tr>
<th>Image</th>
<th>Split Bregman</th>
<th>Graph Cuts (4 point)</th>
<th>Graph Cuts (16 point)</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 × 256 Blocks</td>
<td>0.0732</td>
<td>0.214</td>
<td>0.468</td>
</tr>
<tr>
<td>512 × 512 Lena</td>
<td>0.2412</td>
<td>0.709</td>
<td>1.51</td>
</tr>
</tbody>
</table>
Intermediate images are smooth

Original

Noisy (sigma=25)

10 Iterations

50 Iterations
Intermediate images are smooth

Noisy (sigma=15)

10 Iterations

50 Iterations
References


Thank you for your attention. Any questions?