User's Guide for TVAL3: TV Minimization by Augmented Lagrangian and Alternating Direction Algorithms

Chengbo Li, Wotao Yin, and Yin Zhang
Department of CAAM
Rice University, Houston, Texas, 77005

(Version 1.0, 12-01-2010)

Abstract

This User's Guide describes the functionality and basic usage of the Matlab package TVAL3 for total variation minimization. The main algorithm used in TVAL3 is briefly introduced in the appendix.

Contents

1 Introduction

TVAL3 is short for "Total Variation Minimization by Augmented Lagrangian and Alternating Direction Algorithms". It is a Matlab solver that at present can be applied to the following four total variation (TV) based minimization models for reconstructing an image u from its (linear, incomplete, and/or degraded) observations b:

$$\begin{aligned}
&(\text{TV}) && \min_{u\in\mathbb{C}^n} \textstyle\sum_i \|D_i u\|_p, \quad \text{s.t. } Au = b, && (1)\\
&(\text{TV+}) && \min_{u\in\mathbb{R}^n} \textstyle\sum_i \|D_i u\|_p, \quad \text{s.t. } Au = b \;\; u \geq 0, && (2)\\
&(\text{TV/L2}) && \min_{u\in\mathbb{C}^n} \textstyle\sum_i \|D_i u\|_p + \frac{\mu}{2}\|Au - b\|_2^2, && (3)\\
&(\text{TV/L2+}) && \min_{u\in\mathbb{R}^n} \textstyle\sum_i \|D_i u\|_p + \frac{\mu}{2}\|Au - b\|_2^2, \quad \text{s.t. } u \geq 0, && (4)
\end{aligned}$$

where $\|\cdot\|_p$ for $p = 1$ or 2 is the 1-norm or 2-norm, respectively, $n = n_1 \times n_2$ is the size of signals or images, $D_i u$ ($\in \mathbb{C}^2$ or $\mathbb{R}^2$ depending on $u \in \mathbb{C}^n$ or $\mathbb{R}^n$) is the discrete gradient vector of u at position i, $A \in \mathbb{C}^{m\times n}$ ($m < n$) is the measurement matrix, $b \in \mathbb{C}^m$ is the observation of u via some linear measurements, and $\mu > 0$ is the penalty parameter for the TV/L2 models.

The first terms in the objective functions are the TV regularization terms, which are isotropic for $p = 2$, and anisotropic for $p = 1$. Using the isotropic ones is often preferred, and is the default in TVAL3, since it results in fewer zig-zagging object boundaries in the reconstructed image. The second terms in objective functions are commonly referred to as fidelity terms.

2 Quick Start

TVAL3 can be downloaded from the URL:

`http://www.caam.rice.edu/~optimization/L1/TVAL3/`.

It has a simple Matlab interface with 5 input arguments and either 1 or 2 output arguments:

```
             U = TVAL3(A,b,n1,n2,opts);
or     [U,out] = TVAL3(A,b,n1,n2,opts);
```

where the input A is either a matrix in $\mathbb{C}^{m\times n}$ or a function handle (see more information below), $b \in \mathbb{C}^m$ is the observation, n_1 and n_2 represent the size of the signal/image, and the output $U \in \mathbb{C}^{n_1 \times n_2}$ is the recovered signal/image. All these quantities can be real or complex. The input argument `opts` is a structure carrying control options. The optional output argument `out` contains some secondary output information.

Unzipping the package creates the directory `TVAL3_xx` where "xx" is a version number. Please start by running `warm_up.m`, which updates Matlab's search path and calls `mex` to compile a C++ file for a fast Walsh-Hadamard transform into a Matlab mex file (as such you will need a compiler installed on your system). Besides, running the Matlab script `demo.m` in the current directory would also help users set necessary path, but without compiling the C++ file.

Upon successful setup, four more `demo` files in the `Demos` directory are ready to run.

The input argument `A` should be either an $m \times n$ matrix or a Matlab function handle corresponding to a given linear transform A from $\mathbb{C}^n$ to $\mathbb{C}^m$ and its adjoint A^* in the way such that

`A(x,1) returns` Ax,
`A(y,2) returns` A^*y.

For an example of defining such a function handle `A`, see the function `dfA` at the bottom of the function `demo_lina.m` under the folder `Demos`.

TVAL3 accepts all kinds of measurement matrices A or corresponding function handles. It is preferred, but not required, for A to have orthogonal and normalized rows. The speed of TVAL3 is largely affected by how fast Ax and A^*y can be computed.

TVAL3 requires `opts` to contain at least one field. If users choose TV/L2 or TV/L2+ model, `opts.mu` must be set according to the value of μ in the model. All the fields of `opts` are described in Section 4 below.

3 Model Selection

TVAL3 can solve one of the four supported models, TV, TV+, TV/L2, and TV/L2+, while each one can be either isotropic or anisotropic. A model is selected according to options supplied in `opts`.

- **The isotropic TV model**
  This model is solved by default. (However, please specify at least one field of `opts`, which can be any one compatible with this model. For example, `opts.disp = false`.)

- **The isotropic TV+ model**
  Set `opts.nonneg = true`.

- **The isotropic TV/L2 model**
  Set `opts.TVL2 = true`.

- **The isotropic TV/L2+ model**
  Set `opts.nonneg = true` and `opts.TVL2 = true`.

- **One of the above models with anisotropic TV**
  To solve any of above four models with *anisotropic* TV corresponding to $p = 1$, set `opts.TVnorm = 1` in addition to setting a corresponding field in the way described above.

For the efficiency of TVAL3, we always suggest users to avoid TV/L2 or TV/L2+ model unless necessary, since TV or TV+ model could be faster to obtain a certain accuracy. Even though the noise exists in your cases, TV or TV+ model still works fairly well in practice.

4 Fields of `opts`

The following fields of `opts` can be specified by users, and their default values are given in brackets "[ ]". They are roughly ordered by the importance according to the authors' experience.

```
opts.mu = [2^8]                   (primary penalty parameter)
opts.beta = [2^5]                 (secondary penalty parameter)
opts.mu0 = opts.mu                (initial mu for continuation)
opts.beta0 = opts.beta            (initial beta for continuation)
opts.tol = [1.e-6]                (outer stopping tolerance)
opts.tol_inn = [1.e-3]            (inner stopping tolerance)
opts.maxit = [1025]               (maximum total iterations)
opts.maxcnt = [10]                (maximum outer iterations)
opts.TVnorm = [2]                 (isotropic or anisotropic TV)
opts.nonneg = [false]             (switch for nonnegative models)
opts.TVL2 = [false]               (switch for TV/L2 models)
opts.isreal = [false]             (switch for real signals/images)
```

```
opts.scale_A = [true]              (switch for scaling A)
opts.scale_b = [true]              (switch for scaling b)
opts.disp = [false]                (switch for iteration info printout)
opts.init = [1]                    (initial guess)
```

Among fields, `opts.mu` appears to be the most important one. To get the best performance, the value of `opts.mu` should be set in accordance with both the noise level in the observation b and the sparsity level of the underlying signal/image u. For example, the higher the noise level is, the smaller `opts.mu` should be (of course it is difficult to estimate the noise level without knowing the true solution). Based on our experience, the simplest way to choose `mu` is to try different values from 2^4 up to 2^{13} and compare the recovered signals/images. The value of `opts.beta` also affects the performance of TVAL3, but it is much less important than `opts.mu`. Users can also decide `opts.beta` by trying with values from 2^4 up to 2^{13} if necessary. Options `opts.mu0` and `opts.beta0` suggest if the continuation scheme is applied. The default values mean no need for continuation. users can trigger it by setting both `opts.mu0` and `opts.beta0` much smaller than `opts.mu` and `opts.beta`, respectively (see Demos). In some scenarios, continuation scheme could accelerate the convergence and reduce the elapsed time. Both `opts.tol` and `opts.tol_inn` determine the solution accuracy. Their smaller values result in a longer elapsed time and usually a better solution quality. If the observation is noisy or the problem is large-scale, `opts.tol = 1.e-2 or 1.e-3` might be sufficient. The options `opts.maxit` and `opts.maxcnt` set limits for the numbers of total and outer iterations, respectively. `opts.TVnorm`, `opts.nonneg`, and `opts.TVL2` determines which one of the four models is solved. If the true solution is real (as opposed to complex), `opts.isreal` should be set as *true*. The options `opts.scale_A` and `opts.scale_b` determine whether `A` and `b` should be scaled, respectively. In general, decisions are made automatically so assigning non-default values to these two options is not recommended. The field `opts.disp` controls whether iteration information is displayed or not. Furthermore, the initial solution is assigned according to `opts.init`. `opts.init = 1` assigns A^*b, `opts.init = 0` assigns the zero matrix, and `opts.init = U0` assigns a user-provided matrix U0.

Getting the best solution quality often requires tuning certain options. Among the most important ones are `opts.mu`, `opts.tol`, `opts.beta`, and `opts.maxcnt`. It is advisable to try the default values first before any tuning.

5 Feedback

Your feedback is welcome and appreciated! You can send your questions, bug reports, and suggestions to `cl9@rice.edu`.

Acknowledgments

The authors are grateful to a group of users, colleagues, and students from Rice University, especially those in the ECE department, who helped test previous beta versions of the code and provided useful feedback. The first author would like to thank Dr. Junfeng Yang of Nanjing University for his tremendous help at the beginning of the project and Ting Sun of Rice University for providing test data. The work of C. Li has been supported in part by NSF Grant DMS-0811188, ONR Grant N00014-08-1-1101, and AFOSR Grant FA9550-09-C-0121. The work of W. Yin has been supported in part by NSF CAREER Award DMS-0748839, ONR Grant N00014-08-1-1101, AFOSR Grant FA9550-09-C-0121, and an Alfred P. Sloan Research Fellowship. The work of Y. Zhang has been supported in part by NSF Grant DMS-0811188 and ONR Grant N00014-08-1-1101.

Appendix: Algorithm

Our algorithmic framework is a Lagrangian multiplier method applied to a particular augmented Lagrangian function; that is,

Algorithm 1 *Input A, b, n_1, n_2, and opts.*

> **While** *"not converge"* **Do**
>
> — *Approximately minimize the augmented Lagrangian function by an alternating direction scheme.*
>
> — *Update multipliers.*
>
> **End Do**

The convergence properties of algorithms in this framework have been well analyzed in the optimization literature (see [1], for example).

To briefly describe our algorithm, we take the real isotropic TV model

$$\min_{u \in \mathbb{R}^n} \sum_i \|D_i u\|, \quad \text{s.t. } Au = b,$$

(where we use $\|.\|$ for $\|.\|_2$ for simplicity) for example. This TV model is clearly equivalent to

$$\min_{w_i \in \mathbb{R}^2, u \in \mathbb{R}^n} \sum_i \|w_i\|, \quad \text{s.t. } Au = b \text{ and } D_i u = w_i \text{ for all } i.$$

Its corresponding augmented Lagrangian problem is

$$\min_{w_i, u} \sum_i (\|w_i\| - \nu_i^T (D_i u - w_i) + \frac{\beta}{2} \|D_i u - w_i\|^2) - \lambda^T (Au - b) + \frac{\mu}{2} \|Au - b\|^2. \quad \text{(A-1)}$$

An alternating minimization scheme is applied to solving (A-1). For a fixed u, the minimizers w_i for all i can be obtained via the formula

$$w_i \quad = \quad \max\left\{ \left\| D_i u - \frac{\nu_i}{\beta} \right\| - \frac{1}{\beta}, 0 \right\} \frac{D_i u - \nu_i / \beta}{\|D_i u - \nu_i / \beta\|}. \quad \text{(A-2)}$$

On the other hand, for fixed w_i, we approximately minimize the quadratic with respect to u by taking one steepest descent step with the steplength computed by a back-tracking non-monotone line search scheme [2] starting from a Barzilai-Borwein (BB) step length [3]. After each steepest descent step, we update w_i and repeat the process until (A-1) is approximately solved within a prescribed tolerance.

Let $\hat{u}$ and $\{\hat{w}_i\}$ represent an approximate solution to (A-1). The multipliers are then updated through the well-known formulas: for all i

$$\begin{aligned} \nu_i &\leftarrow \nu_i - \beta (D_i \hat{u} - \hat{w}_i), \\ \lambda &\leftarrow \lambda - \mu (A\hat{u} - b). \end{aligned}$$

Combining the framework Algorithm 1 with the inner iterations described above leads to the core algorithm of the solver TVAL3 in version 1.0.

For anisotropic models, all formulas remain the same except a slight change in the formula (A-2) for updating w_i. For models with nonnegativity constraints, a projected gradient method instead of the steepest descent method was used for updating u.

For more details, an elaborate description of TVAL3 including theoretical and numerical results will be fully stated in a forthcoming paper.

References

[1] Jorge Nocedal and Stephen J. Wright. Numerical Optimization. Springer-Verlag, New York, U.S.A., (2006).

[2] H. Zhang and W. W. Hager. A nonmonotone line search technique and its application to unconstrained optimization. SIAM J. Optim., 14 (2004), pp. 1043–1056.

[3] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. IMA J. Numer. Anal., 8 (1988), pp. 141C148.