Fast Newton-type Methods for Total Variation Regularization

Álvaro Barbero ALVARO.BARBERO@UAM.ES
Universidad Autónoma de Madrid and Instituto de Ingeniería del Conocimiento, Madrid, Spain

Suvrit Sra SUVRIT@TUEBINGEN.MPG.DE
MPI for Intelligent Systems, Tübingen, Germany

Abstract

Numerous applications in statistics, signal processing, and machine learning regularize using Total Variation (TV) penalties. We study anisotropic (ℓ_1-based) TV and also a related ℓ_2-norm variant. We consider for both variants associated (1D) *proximity operators*, which lead to challenging optimization problems. We solve these problems by developing Newton-type methods that outperform the state-of-the-art algorithms. More importantly, our 1D-TV algorithms serve as building blocks for solving the harder task of computing 2- (and higher)-dimensional TV proximity. We illustrate the computational benefits of our methods by applying them to several applications: (i) image denoising; (ii) image deconvolution (by plugging in our TV solvers into publicly available software); and (iii) four variants of fused-lasso. The results show large speedups—and to support our claims, we provide software accompanying this paper.

1. Introduction

Applications in statistics, signal processing, and machine learning frequently involve problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad L(\boldsymbol{x}) + R(\boldsymbol{x}), \tag{1}$$

where L is a differentiable, convex loss, and R is a convex, possibly nonsmooth regularizer. Nonsmoothness of R makes optimizing (1) hard; but often the difficulties raised by this nonsmoothness can be alleviated by passing to R's *proximity operator*, defined by the following operation:

$$\operatorname{prox}_R(\boldsymbol{y}) := \operatorname{argmin}_{\boldsymbol{x}} \quad \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 + R(\boldsymbol{x}). \tag{2}$$

Appearing in *Proceedings of the 28^{th} International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

Within machine learning and related fields, the benefits of invoking the proximity operator (2) are well-recognized (Nesterov, 2007; Combettes & Pesquet, 2009; Duchi & Singer, 2009), and several choices of R have already been considered.

We study a special choice for R: one and higher dimensional *total-variation* (TV)[1]; for $\boldsymbol{x}\in\mathbb{R}^n$, this is defined as

$$\mathrm{Tv}_p^{1\mathrm{D}}(\boldsymbol{x}) := \Big(\sum\nolimits_{i=1}^{n-1} |x_{i+1}-x_i|^p\Big)^{1/p}, \tag{3}$$

and for matrices $\boldsymbol{X}\in\mathbb{R}^{m\times n}$ it is defined as

$$\mathrm{Tv}_{p,q}^{2\mathrm{D}}(\boldsymbol{X}) := \sum\nolimits_{i=1}^{m}\Big(\sum\nolimits_{j=1}^{n-1} |x_{i,j+1}-x_{i,j}|^p\Big)^{1/p} + \sum\nolimits_{j=1}^{n}\Big(\sum\nolimits_{i=1}^{m-1} |x_{i+1,j}-x_{i,j}|^q\Big)^{1/q}, \tag{4}$$

where usually $p,q \in \{1,2\}$. We focus on two key variants of (3) and (4): *anisotropic*-TV (see e.g. Bioucas-Dias & Figueiredo, 2007), with p, $q=1$; and TV with both p and $q=2$. Extension of (3) to tensor data is relegated to (Barbero & Sra, 2011), for paucity of space.

The regularizers $\mathrm{Tv}_1^{1\mathrm{D}}$ and $\mathrm{Tv}_{1,1}^{2\mathrm{D}}$ arise in many applications—e.g., image denoising and deconvolution (Dahl et al., 2010), fused-lasso (Tibshirani et al., 2005), logistic fused-lasso (Kolar et al., 2010), and change-point detection (Harchaoui & Lévy-Leduc, 2010); also see the related work (Vert & Bleakley, 2010). This fairly broad applicability motivates us to develop efficient proximity operators for TV. Before beginning the technical discussion, let us summarize our key contributions.

Algorithms: For $\mathrm{Tv}_1^{1\mathrm{D}}$- and $\mathrm{Tv}_2^{1\mathrm{D}}$-proximity, we derive efficient Newton-type algorithms, which we subsequently use as building blocks for rapidly solving the harder case of $\mathrm{Tv}_{p,q}^{2\mathrm{D}}$-proximity (also higher-D TV) with $p,q\in\{1,2\}$.

Applications: We highlight some of the benefits of our fast algorithms by showing their application to image denoising; we also show their use as efficient subroutines in

[1] Our definitions of TV are different from the original ROF model of TV (Rudin et al., 1992); also see §5.2.

larger solvers for image deconvolution and for solving four variants of fused-lasso.

Software: To support our numerical results, we provide efficient implementations of our algorithms.[2]

An additional important message of our paper is: even for large-scale machine learning problems, Newton-type methods can be superior to first-order methods, provided the problem has enough structure. This viewpoint, though obvious, seems to be espoused by a surprisingly small fraction of researchers within the machine learning community.

The literature on TV is huge, so it cannot be summarized here. However, we do mention here some of the most directly relevant work.

Previously, Vogel & Oman (1996) suggested a Newton approach for TV; they smoothed the objective, but noted that it leads to numerical difficulties. In contrast, we solve the nonsmooth problem directly. Recently, Liu et al. (2010) presented tuned algorithms for Tv_1^{1D}-proximity based on a careful "restart" heuristic; their methods show strong empirical performance but do not extend easily to higher-D TV. Our Newton-type methods outperform the tuned methods of (Liu et al., 2010), and fit nicely in a general algorithmic framework that allows tackling the harder two- and higher-D TV problems.

TV regularization in itself arises frequently in image denoising, whereby a large number of TV-based denoising algorithms exist (see e.g. Zhu & Chan, 2008). However, in contrast to our paper, most TV-based methods use the standard isotropic TV or ROF model (Rudin et al., 1992), and there are few methods tailored to anisotropic TV, except those developed in the context of fused-lasso (Friedman et al., 2007; Liu et al., 2010).

It is tempting to assume that existing isotropic algorithms, such as the state-of-the-art PDHG (Zhu & Chan, 2008) method, can be easily adapted. But this is not so. PDHG requires fine-tuning of its parameters, and to obtain fast performance its authors apply non-trivial adaptive rules that fail on our anisotropic model. ADMM-style algorithms (Combettes & Pesquet, 2009), whose convergence speed is highly sensitive to their stepsize parameters, also pose similar problems. In stark contrast, our solvers do not require *any parameter tuning*, and run rapidly.

Our TV methods can be plugged in directly into solvers such as TwIST (Bioucas-Dias & Figueiredo, 2007) or SALSA (Afonso et al., 2010) for image deblurring, and into methods such as FISTA (Beck & Teboulle, 2009) or TRIP (Kim et al., 2010), for TV-regularized optimization.

[2] See *http://arantxa.ii.uam.es/∼gaa/software.html*

2. One dimensional TV-Proximity

We begin with 1D-TV proximity, and devote most attention to it, since it forms a crucial part of our 2D-TV methods. Introduce the *differencing matrix* $\boldsymbol{D} \in \mathbb{R}^{(n-1)\times n}$ with $d_{ij} = 0$, except for $d_{ii} = -1$ and $d_{i,i+1} = 1$. Let $\text{Tv}_p^{\text{1D}}(\boldsymbol{x}) = \|\boldsymbol{D}\boldsymbol{x}\|_p$; then the TV-proximity problem is

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{D}\boldsymbol{x}\|_p. \tag{5}$$

It is often easier to solve (5) via its dual

$$\max_{\boldsymbol{u}} \quad -\tfrac{1}{2}\|\boldsymbol{D}^T\boldsymbol{u}\|_2^2 + \boldsymbol{u}^T\boldsymbol{D}\boldsymbol{y}, \quad \text{s.t. } \|\boldsymbol{u}\|_q \le \lambda, \tag{6}$$

where $\|\cdot\|_q$ is the *dual-norm* to $\|\cdot\|_p$. If $\boldsymbol{u}$ is dual feasible, then the primal variable $\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{D}^T\boldsymbol{u}$. The corresponding *duality-gap* is easily computed as

$$\text{gap}(\boldsymbol{x},\boldsymbol{u}) := \lambda\|\boldsymbol{D}\boldsymbol{x}\|_p - \boldsymbol{u}^T\boldsymbol{D}\boldsymbol{x}. \tag{7}$$

If $\boldsymbol{u}^*$ is the optimal dual solution, then the optimal primal solution is given by $\boldsymbol{x}^* = \boldsymbol{y} - \boldsymbol{D}^T\boldsymbol{u}^*$.

2.1. TV-L1: Proximity for Tv_1^{1D}

For 1D anisotropic TV, the dual (6) becomes

$$\min_{\boldsymbol{u}} \ \phi(\boldsymbol{u}) := \tfrac{1}{2}\|\boldsymbol{D}^T\boldsymbol{u}\|_2^2 - \boldsymbol{u}^T\boldsymbol{D}\boldsymbol{y}, \text{ s.t. } \|\boldsymbol{u}\|_\infty \le \lambda. \tag{8}$$

This is a box-constrained quadratic program; so it can be solved by methods such as TRON (Lin & Moré, 1999), L-BFGS-B (Byrd et al., 1994), or projected-Newton (PN) (Bertsekas, 1982). But these methods can be inefficient if invoked out-of-the-box, and carefully exploiting problem structure is a must. PN lends itself well to such structure exploitation, and we adapt it to develop a highly competitive method for solving the dual problem (8).

The generic PN procedure runs iteratively: it first identifies a special subset of the active variables and uses these to compute a reduced Hessian. Then, this Hessian is used to scale the gradient and move in direction opposite to it, scaling by a stepsize, if needed. Finally, the next iterate is obtained by projecting onto the constraints, and the cycle repeats. At each iteration we select the active variables:

$$I := \{i \mid \big(u_i = -\lambda \text{ and } [\nabla\phi(\boldsymbol{u})]_i > \epsilon\big) \text{ or } \big(u_i = \lambda \text{ and } [\nabla\phi(\boldsymbol{u})]_i < -\epsilon\big)\},$$

where $\epsilon \ge 0$ is small scalar. Let $\bar{I} := \{1\ldots n\}\setminus I$ be the set of indices not in I. From the Hessian $\boldsymbol{H} = \nabla^2\phi(u)$ we extract the *reduced Hessian* $\boldsymbol{H}_{\bar{I}}$ by selecting rows and columns indexed by $\bar{I}$, and compute the "reduced" update

$$\boldsymbol{u}_{\bar{I}} \leftarrow P(\boldsymbol{u}_{\bar{I}} - \alpha\boldsymbol{H}_{\bar{I}}^{-1}[\nabla\phi(\boldsymbol{u})]_{\bar{I}}), \tag{9}$$

where α is a stepsize, and P denotes elementwise projection onto the constraints. Let us now see how to exploit structure to efficiently perform the above steps.

First, notice that the Hessian $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{D}^T$ is symmetric and tridiagonal, with 2s on the main diagonal and -1s on the sub- and superdiagonals. Next, observe that whatever the active set I, the corresponding reduced Hessian $\boldsymbol{H}_{\bar{I}}$ remains symmetric tridiagonal. This is crucial because then we can quickly compute the updating direction $\boldsymbol{d}_{\bar{I}}$ by solving $\boldsymbol{H}_{\bar{I}}\boldsymbol{d}_{\bar{I}} = [\nabla\phi(\boldsymbol{u}^t)]_{\bar{I}}$. This linear system can be solved by computing the Cholesky decomposition $\boldsymbol{H}_{\bar{I}} = \boldsymbol{R}^T\boldsymbol{R}$ in *linear time* using the LAPACK routine DPTTRF (Anderson et al., 1999). The resulting $\boldsymbol{R}$ is bidiagonal, so we can solve for $\boldsymbol{d}_{\bar{I}}$ in linear time too. Thus, a full PN iteration takes $O(n)$ time.

The next crucial ingredient is stepsize selection. Bertsekas (1982) recommends Armijo-search along projection arc. However, for our problem Armijo-search is disproportionately expensive. So we resort to a backtracking strategy using quadratic interpolation (Nocedal & Wright, 2000). This strategy is as follows: if the current stepsize α_k does not provide enough decrease in ϕ, we build a quadratic model using $\phi(\boldsymbol{u})$, $\phi(\boldsymbol{u}-\alpha_k\boldsymbol{d})$, and $\partial_\alpha\phi(\boldsymbol{u}-\alpha_k\boldsymbol{d})$. Then, stepsize α_{k+1} is set to the value that minimizes this model. If the new α_{k+1} is larger than or too similar to α_k, its value is halved. Note that the gradient $\nabla\phi(\boldsymbol{u})$ might be misleading if $\boldsymbol{u}$ has components at the boundary and $\boldsymbol{d}$ points outside this boundary (because then, due to the subsequent projection no real improvement would be obtained by stepping outside the feasible region). To address this concern, we modify the computation of the gradient $\nabla\phi(\boldsymbol{u})$, replacing by zeros the entries that relate to direction components pointing outside the feasible set.

Algorithm 1 PN algorithm for TV-L1-proximity

Solve $\boldsymbol{D}\boldsymbol{D}^T\boldsymbol{u}^* = \boldsymbol{D}\boldsymbol{y}$.
if $\|\boldsymbol{u}^*\|_\infty \le \lambda$ **return** $\boldsymbol{u}^*$; **end if**
$\boldsymbol{u}^0 = P[\boldsymbol{u}^*]$, $t = 0$
while duality-gap $>$ tolerance **do**
  Identify set of active constraints I; let $\bar{I} = \{1\dots n\} \setminus I$
  Construct reduced Hessian $\boldsymbol{H}_{\bar{I}}$
  Solve $\boldsymbol{H}_{\bar{I}}\boldsymbol{d}_{\bar{I}} = [\nabla\phi(\boldsymbol{u}^t)]_{\bar{I}}$
  Compute stepsize α using backtracking + interpolation
  Update $\boldsymbol{u}_{\bar{I}}^{t+1} = P[\boldsymbol{u}_{\bar{I}}^t - \alpha\boldsymbol{d}_{\bar{I}}]$.
  $t \leftarrow t+1$.
end while
return $\boldsymbol{u}^t$.

Finally, we must account for the case when λ is so large that the unconstrained optimum coincides with the constrained one. In this case, we just obtain $\boldsymbol{u}^*$ via $\boldsymbol{D}\boldsymbol{D}^T\boldsymbol{u}^* = \boldsymbol{D}\boldsymbol{y}$. All the above ideas are encapsulated as Algorithm 1.

2.2. TV-L2: Proximity for Tv_2^{1D}

For TV-L2 proximity $p = 2$, so the dual (6) becomes

$$\min_{\boldsymbol{u}}\ \phi(\boldsymbol{u}) := \tfrac{1}{2}\|\boldsymbol{D}^T\boldsymbol{u}\|_2^2 - \boldsymbol{u}^T\boldsymbol{D}\boldsymbol{y},\ \text{ s.t. } \|\boldsymbol{u}\|_2 \le \lambda. \tag{10}$$

Algorithm 2 MSN based TV-L2 proximity

Initialize: $\alpha^0 = 0$, $t = 0$.
while ($\neg$ converged) **do**
  Compute Cholesky decomp. $\boldsymbol{D}\boldsymbol{D}^T + \alpha^t\boldsymbol{I} = \boldsymbol{R}^T\boldsymbol{R}$.
  Obtain $\boldsymbol{u}$ by solving $\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{u} = \boldsymbol{D}\boldsymbol{y}$.
  Obtain $\boldsymbol{q}$ by solving $\boldsymbol{R}^T\boldsymbol{q} = \boldsymbol{u}$.
  Update α using (15)
  $t \leftarrow t+1$.
end while
return $\boldsymbol{u}^t$

Problem (10) is an instance of the well-known trust-region subproblem, whereby a variety of numerical methods are available for it (Conn et al., 2000). Below we derive an algorithm based on the Moré-Sorensen Newton (MSN) iteration (Moré & Sorensen, 1983), which in general is expensive, but in our case proves to be efficient thanks to the tridiagonal Hessian. Curiously, experiments show that for a certain range of λ values, gradient-projection (GP) (without line-search though) can also be competitive. Thus, for best performance we prefer a hybrid MSN-GP method for (10).

Consider the KKT conditions for (10):

$$\begin{aligned}&(\boldsymbol{D}\boldsymbol{D}^T + \alpha\boldsymbol{I})\boldsymbol{u} = \boldsymbol{D}\boldsymbol{y},\\ &\alpha(\|\boldsymbol{u}\|_2 - \lambda) = 0, \quad \alpha \ge 0,\end{aligned} \tag{11}$$

where α is a Lagrange multiplier. There are two cases: $\|\boldsymbol{u}\|_2 < \lambda$; or $\|\boldsymbol{u}\|_2 = \lambda$. If $\|\boldsymbol{u}\|_2 < \lambda$, then $\alpha = 0$ and $\boldsymbol{u}$ is obtained by solving $\boldsymbol{D}\boldsymbol{D}^T\boldsymbol{u} = \boldsymbol{D}\boldsymbol{y}$. Conversely, if the solution to $\boldsymbol{D}\boldsymbol{D}^T\boldsymbol{u} = \boldsymbol{D}\boldsymbol{y}$ lies in the interior, then it solves (11). Thus, we need to only consider $\|\boldsymbol{u}\|_2 = \lambda$.

Given α, one has $\boldsymbol{u}(\alpha) = (\boldsymbol{D}\boldsymbol{D}^T + \alpha\boldsymbol{I})^{-1}\boldsymbol{D}\boldsymbol{y}$. So we must compute the "true" α. This can be done by solving $\|\boldsymbol{u}(\alpha)\|_2^2 = \lambda^2$, or alternatively solving the MSN equation

$$h(\alpha) := \lambda^{-1} - \|\boldsymbol{u}(\alpha)\|_2^{-1} = 0, \tag{12}$$

which is written so, as it is almost linear in the search interval, resulting in fast convergence (Moré & Sorensen, 1983). Newton's iteration for (12) is

$$\alpha \leftarrow \alpha - h(\alpha)/h'(\alpha), \tag{13}$$

and a simple calculation shows that

$$\frac{1}{h'(\alpha)} = \frac{\|\boldsymbol{u}(\alpha)\|_2^3}{\boldsymbol{u}(\alpha)^T(\boldsymbol{D}\boldsymbol{D}^T + \alpha\boldsymbol{I})^{-1}\boldsymbol{u}(\alpha)}. \tag{14}$$

The key idea in MSN is to eliminate the matrix inverse in (14) by introducing the Cholesky decomposition $\boldsymbol{D}\boldsymbol{D}^T + \alpha\boldsymbol{I} = \boldsymbol{R}^T\boldsymbol{R}$ and defining a vector $\boldsymbol{q} = (\boldsymbol{R}^T)^{-1}\boldsymbol{u}$. As a result, iteration (13) becomes

$$\alpha \leftarrow \alpha - \frac{\|\boldsymbol{u}\|_2^2}{\|\boldsymbol{q}\|_2^2}\left(1 - \frac{\|\boldsymbol{u}\|_2}{\lambda}\right). \tag{15}$$

Observe that both $\boldsymbol{R}$ and $\boldsymbol{q}$ can be computed in linear time, so the overall iteration (15) runs in linear time.

The MSN iteration (15) is fairly sophisticated. Let us look at a much simpler one: GP with a *fixed* stepsize α_0

$$\boldsymbol{u}^{t+1} = P_{\|\cdot\|_2 \leq \lambda}(\boldsymbol{u}^t - \alpha_0 \nabla \phi(\boldsymbol{u}^t)), \tag{16}$$

which is set to the inverse of the largest eigenvalue of the Hessian $\boldsymbol{D}\boldsymbol{D}^T$. This is easily done as the eigenvalues have a closed-form expression, namely $\lambda_i = 2 - 2\cos\left(\frac{i\pi}{n}\right)$ (for $0 \leq i \leq n$). The largest $\lambda_{n-1} = 2 - 2\cos\left(\frac{(n-1)\pi}{n}\right)$, which tends to 4 as $n \to \infty$; so $\alpha_0 = 1/4$ is a good approximation. Furthermore, the projection $P_{\|\cdot\|_2 \leq \lambda}$ is also trivial. Thus, the GP iteration (16) can be attractive, and it indeed can outperform the more sophisticated MSN method, though only for a very limited range of λ values. Therefore, in practice we recommend a hybrid of GP and MSN.

3. Two-dimensional TV Proximity

Now we advance to the harder problem of two-dimensional TV. Let $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ be an input matrix, and let $\boldsymbol{x}^i$ denote its i-th row, and $\boldsymbol{x}_j$ its j-th column. Further, let $\boldsymbol{D}_n$ and $\boldsymbol{D}_m$ be differencing matrices for the row and column dimensions. Then, the regularizer (4) can be written as

$$\mathrm{Tv}^2_{p,q}(\boldsymbol{X}) = \sum\nolimits_i \|\boldsymbol{D}_n \boldsymbol{x}^i\|_p + \sum\nolimits_j \|\boldsymbol{D}_m \boldsymbol{x}_j\|_q. \tag{17}$$

The corresponding $\mathrm{Tv}^{2\mathrm{D}}_{p,q}$-proximity problem is

$$\min_{\boldsymbol{X}} \quad \tfrac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2 + \lambda\, \mathrm{Tv}^2_{p,q}(\boldsymbol{X}), \tag{18}$$

where $\lambda > 0$ is a penalty parameter. Unfortunately, in general, the proximity operator for a sum of convex functions is difficult to compute. However, if we could split (18) into

$$\min_{\boldsymbol{X}} \tfrac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2 + \lambda \sum\nolimits_i \|\boldsymbol{D}_n \boldsymbol{x}^i\|_p \tag{19}$$

$$\min_{\boldsymbol{X}} \tfrac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2 + \lambda \sum\nolimits_j \|\boldsymbol{D}_m \boldsymbol{x}_j\|_q, \tag{20}$$

then our task would be greatly simplified, especially because (19) and (20) themselves further decompose into 1D-TV proximity problems. Fortunately, at the cost of slight additional storage, we can do precisely this splitting via the *proximal Dykstra* method (Combettes & Pesquet, 2009). Algorithm 3 presents pseudocode.

Remarks: Since (19) and (20) decompose into independent 1D-TV subproblems, we could solve these subproblems in parallel if desired. Also, as shown, Algorithm (3) cannot solve (19) and (20) in parallel due to the shared dependence on $\boldsymbol{Z}_t$. A variant of Algorithm 3 allows us to overcome this limitation, though it is usually more preferable for multi-dimensional TV (Barbero & Sra, 2011). Empirically, Algorithm 3 converges rapidly and usually (19) and (20) must be solved only about 4–6 times. When $p, q \in \{1, 2\}$, we invoke our Newton-type methods to efficiently solve the corresponding 1D-TV subproblems.

Algorithm 3 Proximal Dykstra Algorithm for (18)

Initialize $\boldsymbol{X}_0 = \boldsymbol{Y}$, $\boldsymbol{P}_0 = 0$, $\boldsymbol{Q}_0 = 0$, $t = 0$
while ($\neg$ converged) **do**
  $\boldsymbol{Z}_t$ = Solve (19) with $\boldsymbol{Y} = \boldsymbol{X}_t + \boldsymbol{P}_t$
  $\boldsymbol{P}_{t+1} = \boldsymbol{X}_t + \boldsymbol{P}_t - \boldsymbol{Z}_t$
  $\boldsymbol{X}_{t+1}$ = Solve (20) with $\boldsymbol{Y} = \boldsymbol{Z}_t + \boldsymbol{Q}_t$
  $\boldsymbol{Q}_{t+1} = \boldsymbol{Z}_t + \boldsymbol{Q}_t - \boldsymbol{X}_{t+1}$
  $t \leftarrow t + 1$
end while
return $\boldsymbol{X}_t$

4. Numerical Results: Proximity operators

In this section, we provide experimental results illustrating the performance of our Newton-type algorithms for 1D-TV proximity. We test them under two scenarios: (i) with increasing input size n; and (ii) with varying penalty parameter λ. For scenario (i) we select a random $\lambda \in [0, 50]$ for each run; the data vector $\boldsymbol{y}$ is also generated randomly by picking $y_i \in [-2\lambda, 2\lambda]$ (proportionally scaled to λ) for $1 \leq i \leq n$. For scenario (ii), y_i ranges in $[-2, 2]$, while the penalty λ is varied from 10^{-3} (negligible regularization) to 10^3 (the TV-term dominates).

4.1. Results for TV-L1 proximity

We compare running times of our PN approach (C implementation) against two methods: (i) the FLSA function (C implementation) of the SLEP library (Liu et al., 2009), which seems to be the state-of-the-art method for $\mathrm{Tv}^{1\mathrm{D}}_1$-proximity (Liu et al., 2010); and (ii) the Pathwise Coordinate Optimization method (R + FORTRAN implementation) from (Friedman et al., 2007). For PN and SLEP, we use duality gap of 10^{-5} as the stopping criterion. For Coordinate Optimization, duality gap is not supported so we use its default stopping criteria. Timing results are presented in Figure 1 (left panel) for increasing input sizes and penalties. From the plots we see that both SLEP and PN are much faster than Coordinate Optimization. Though, it must be mentioned that the latter returns the full regularization path, while SLEP and PN compute the solution for only one λ. But this is no limitation; SLEP and PN run much faster and with warm-starts one can rapidly compute solutions for several λ values, if needed.

With increasing input sizes PN finds a solution faster than SLEP, taking roughly at most 60% of the time: explicit numerical values are reported Table 1 for easy reference. Figure 1 indicates that larger speedups are observed for small λ, while for large λ, both SLEP and PN perform similarly. The rationale behind this behavior is simple: for smaller λ the active set I is prone to become larger, and PN ex-

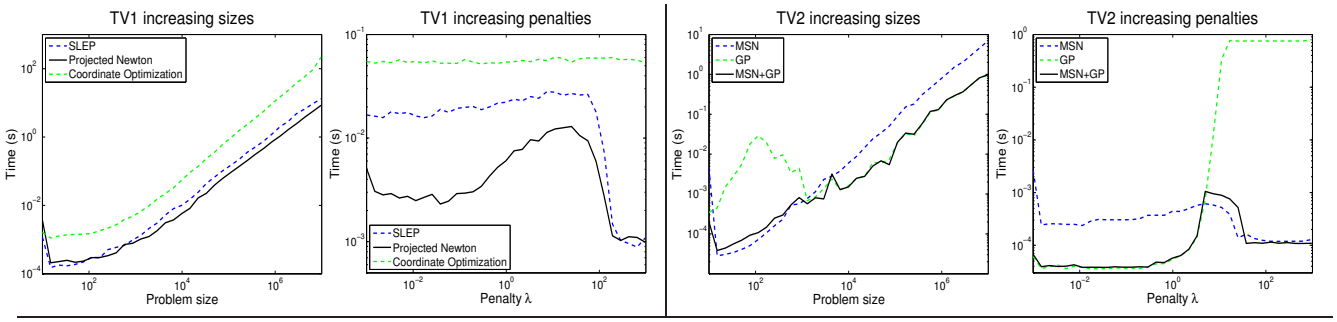

Figure 1. Running times (in secs). Left panel: PN, SLEP and Coordinate Optimization solvers for Tv_1^{1D}-proximity with increasing input sizes and penalties. Right panel: MSN, GP, and hybrid MSN-GP solvers for Tv_2^{1D}-proximity with increasing input sizes and penalties.

Table 1. Running times (in milliseconds) for PN, SLEP and Coordinate Optimization solvers for TV-L1 problems with increasing input sizes (in log-scale); n denotes problem size.

$\log_{10} n$	SLEP	PN	COORD.
1.00	**1.19**	3.6561	1.63
1.53	**0.17**	0.2476	1.37
2.06	0.30	**0.29**	1.52
2.58	0.59	**0.41**	2.69
3.11	1.33	**1.04**	6.74
3.64	5.25	**3.10**	22.20
4.17	15.10	**8.22**	92.41
4.70	67.60	**39.35**	359.50
5.23	221.58	**137.81**	1550.27
5.75	759.62	**464.32**	5678.25
6.28	2874.83	**1655.25**	23285.00
6.81	9457.11	**5659.42**	93366.00

plicitly takes advantage of this set by updating only the not indexed by I. On the other hand, for large λ, PN's strategy becomes similar to that of SLEP, hence the similar performance. Finally, as Coordinate Optimization computes the full regularization path, its runtime is invariant to λ.

4.2. Results for TV-L2

To compare the running times of MSN and GP, we again use duality gap of 10^{-5} as the stopping criterion. Further, as MSN might generate infeasible solutions during the optimization, we also apply a boundary proximity criterion for MSN with tolerance 10^{-6}. Looking at the results it can be seen that the performance of MSN and GP differs noticeably in the two experimental scenarios. While Figure 1 (first plot; right panel) might indicate that GP converges faster than MSN for large inputs, it does so depending on the size of λ relative to $\|\boldsymbol{y}\|_2$. Indeed, Figure 1 (last plot) shows that although for small values of λ, GP runs faster than MSN, as λ increases, GP's performance worsens dramatically, so much that for moderately large λ it is unable to find an acceptable solution even after 10000 iterations (an upper limit imposed in our implementation). Conversely, MSN finds a solution satisfying the stopping criterion under every situation, thus showing a more robust behavior. Therefore, we propose a hybrid approach that combines the strengths of MSN and GP. This hybrid is guided using the following (empirically determined) rule of thumb: if $\lambda < \|\boldsymbol{y}\|_2$ use GP, otherwise use MSN. Further, as a safeguard, if GP is invoked but fails to find a solution within 50 iterations, the hybrid should switch to MSN. This combination guarantees rapid convergence in practice. Results for this hybrid approach are included in the plots in Figure 1, and we see that it successfully mimics the behavior of the better algorithm amongst MSN and GP.

5. Numerical Results: Applications

To highlight the potential benefits of our algorithms we show below three main applications: (i) fused-lasso; (ii) image denoising; and (iii) image deblurring. However, we note here that the exact application itself is not as much a focus as the fact that our solvers apply to a variety of applications while leading to noticeable empirical speedups.

5.1. Results for 1D Fused-Lasso

Our first application is to fused-lasso for which we plug in our algorithms as subroutines into the generic TRIP solver of Kim et al. (2010). We then apply TRIP to solve the following variants of fused-lasso:

1. **Fused-lasso (FL):** Here $L(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$, and $R(\boldsymbol{x}) = \lambda_1\|\boldsymbol{x}\|_1 + \lambda_2\|\boldsymbol{D}\boldsymbol{x}\|_1$; this is the original fused-lasso problem introduced in (Tibshirani et al., 2005), and used in several applications, such as in bioinformatics (Tibshirani & Wang, 2008; Rapaport & Vert, 2008; Friedman et al., 2007).
2. **ℓ_2-variable fusion (VF):** Same as FL but with $\lambda_2\|\boldsymbol{D}\boldsymbol{x}\|_2$ instead. This FL variant seems to be new.
3. **Logistic-fused lasso (LFL):** A logistic loss $L(\boldsymbol{x}, c) = \sum_i \log\left(1 + e^{-\boldsymbol{y}_i(\boldsymbol{a}_i^T\boldsymbol{x}+c)}\right)$ can be introduced in the FL formulation to obtain a more appropriate model for

classification on a dataset $\{(\boldsymbol{a}_i, y_i)\}$. For an application to time-varying networks see Kolar et al. (2010).

4. **Logistic + ℓ_2-fusion (LVF):** This model combines logistic loss with the VF setting.

Table 2. Running times (secs) for SLEP and TRIP for optimizing different versions of fused-lasso with increasing input sizes. Both methods were run to satisfy the same convergence criterion.

MODEL	SLEP			TRIP		
n	10^3	10^4	10^5	10^3	10^4	10^5
FS	0.089	1.43	41.80	**0.02**	**0.10**	**0.86**
VS	0.16	1.26	35.77	**0.02**	**0.10**	**0.90**
LFL	**0.21**	15.01	144.81	0.78	**5.35**	**53.88**
LVF	0.86	**0.02**	132.13	**0.81**	0.15	**11.24**

Synthetic data. We first compare TRIP equipped with our proximity solvers with the approach of Liu et al. (2010). Here random matrices $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ are generated, whose entries are selected to follow a zero mean, unit variance normal distribution. We fix $m = 100$, and set $\lambda_1 = \lambda_2 = 0.01$. Then, we sample matrices with number of columns n varying as 10^3, 10^4, and 10^5. To select the vector of responses $\boldsymbol{y}$, we use the formula $\boldsymbol{y} = \operatorname{sgn}(\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{v})$, where $\boldsymbol{x}_t$, and $\boldsymbol{v}$ are random vectors whose entries have variances 1 and 0.01, respectively. The numerical results are summarized in Table 2, where we compare SLEP (version 4.0) (Liu et al., 2009) against the TRIP-based approach.[3] While for smaller matrices with $n = 10^3$ both methods run similarly fast, as the size of the input matrices increases, the TRIP-based fused-lasso solvers run much faster than SLEP.

Real Data. We tested each of the four FL models on binary classification tasks for the following microarray datasets: ArrayCGH (Stransky et al., 2006), Leukemias (Golub et al., 1999), Colon (U. Alon et al., 1999), Ovarian (Rogers et al., 2005) and Rat (Hua et al., 2009). Each dataset was split into three equal parts (ensuring both classes are present in every split) for training, validation and test. The penalty parameters where found by grid search in the range $\lambda_1, \lambda_2 \in [10^{-3}, 10^1]$ to maximize classification accuracy on the validation splits.

Table 3 shows test accuracies. We see that in general, logistic-loss based FL models yield better classification accuracies than those based on least-squares. This result is natural: logistic-loss is more suited for classification in tasks like the ones proposed for these datasets. Regarding the TV-regularizer, three out of five datasets seem to be insensitive to this choice, though the Tv_1^{1D}-penalty performs better for Ovarian, while Tv_2^{1D} works best for ArrayCGH.

[3]Both TRIP and SLEP are implemented in MATLAB; only the crucial proximity operators are implemented in C.

Table 3. Classification accuracies on microarray data.

DATASET	FL	VF	LFL	LVF
ARRAYCGH	73.6%	**78.9%**	73.6%	73.6%
LEUKEMIAS	92.0%	92.0%	**96.0%**	**96.0%**
COLON	**77.2%**	**77.2%**	**77.2%**	**77.2%**
OVARIAN	**88.8%**	83.3%	77.7%	77.7%
RAT	67.2%	65.5%	**70.4%**	**70.4%**

5.2. Results for 2D-TV

We now show application of our two-dimensional $\text{Tv}_{1,1}^{\text{2D}}$-proximity solver. We are not aware of natural applications for two or higher-dimensional $\text{Tv}_{2,2}^{\text{2D}}$-proximity, so we do not discuss it further. The most basic and natural application of our $\text{Tv}_{1,1}^{\text{2D}}$-proximity is to image denoising. Among the vast number of denoising methods, we compare against the well-established method based on the classic ROF-TV model (Rudin et al., 1992). This model takes an $n\times n$ noisy image $\boldsymbol{Y}$ and denoises it by solving

$$\min_{\boldsymbol{X}} \quad \tfrac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_{\text{F}}^2 + \lambda \text{Tv}_{\text{rof}}(\boldsymbol{X}), \tag{21}$$

where the ROF version of TV is defined as

$$\text{Tv}_{\text{rof}}(\boldsymbol{X}) = \sum\nolimits_{1\le i,j<n} \|(\nabla x)_{i,j}\|_2,$$
$$(\nabla x)_{i,j} = \begin{bmatrix} x_{i+1,j} - x_{i,j} \\ x_{i,j+1} - x_{i,j} \end{bmatrix}.$$

That is, the TV operator is applied on the discrete gradient over the image. This TV regularization is known as isotropic TV, in contrast to our anisotropic TV. Although often the isotropic version Tv_{rof} is preferred, for some applications anisotropic TV shows superior denoising. We show a simple example that illustrates this setting naturally, namely, denoising of two-dimensional barcodes (Choksi et al., 2010). We apply our 2D-TV operator to this setting and compare against the isotropic model which we solved using the state-of-the-art PDHG method (Zhu & Chan, 2008). For further reference we also compare against: (i) the anisotropic TV solver proposed in (Friedman et al., 2007); (ii) an adapted (anisotropic) PDHG solver obtained easily by modifying the original formulation; and (iii) a median filter. We note that the stepsize selection rules recommended for PDHG, failed to produce fast runtimes when applied to anisotropic TV. Thus, to make PDHG competitive, we searched for optimal stepsize parameters for it by exhaustive grid search.

Table 4 presents runtimes and Improved Signal-to-Noise Ratio (ISNR) values obtained for a series of denoising experiments on barcode images that were corrupted by additive (variance 0.2) and multiplicative (variance 0.3) gaussian noise. To compensate for the loss of contrast produced by TV filtering, intensity values are rescaled to the range

Table 4. Barcodes denoising results obtained via PN, Coordinate Optimization and adapted PDHG for the anisotropic model, genuine PDHG for the isotropic model, and a median filter. ISNR (dB) values (higher is better) and running times in seconds are shown.

	ANISOTROPIC				ISOTROPIC		MEDIAN	
SIZE	ISNR	TIME PN	TIME COORD.	TIME PDHG	ISNR	TIME PDHG	ISNR	TIME
100×100	2.39	0.11	2.85	0.64	2.04	0.03	1.24	0.00
175×175	4.14	0.27	15.99	8.71	3.38	0.11	1.74	0.02
300×300	5.48	0.88	140.78	128.72	4.38	0.37	2.35	0.03
375×375	6.04	1.39	167.68	93.87	4.39	0.76	2.42	0.07
500×500	4.42	2.59	228.55	203.19	3.58	1.30	2.18	0.09

(a) (b) (c) (d) (e)

Figure 2. Example of barcode denoising for the isotropic and anisotropic models on a 175×175 image. a) Clean image. b) Noisy image. c) Anisotropic denoising. d) Isotropic denoising. e) Median filter.

of the original image. The penalty parameter λ for each model was chosen so as to maximize the on the 300×300 image. As expected, the anisotropic TV regularizer is more appropriate for the underlying structure of the image, and thus obtains lower reconstruction errors. An example is shown in Figure 2, where we also observe visually better reconstructions via the anisotropic model. Additional experimental results are in (Barbero & Sra, 2011).

Regarding running times, our PN solver vastly outperforms Coordinate Optimization and anisotropic PDHG. The isotropic version of the problem is simpler than the anisotropic one, so it is no surprise that the carefully tuned PDHG approach requires less time than PN. It is also worth mentioning that in (Choksi et al., 2010) an ℓ_1 loss is used, and denoising cast as a Linear Program, to which a generic solver is applied; this approach requires runtimes of over 10^3 seconds for the largest image.

5.3. Image deconvolution

With little added effort our two-dimensional TV solver can be employed for the harder problem of image deconvolution, which takes the form

$$\min_{\boldsymbol{x}} \quad \tfrac{1}{2}\|\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda R(\boldsymbol{x}),$$

where K is a blur operator, R is a regularizer, and $\boldsymbol{x}$ encodes an image. As stated before, this problem can also be solved using prox_R as a building block. Precisely this is done by the solver SALSA (Afonso et al., 2010). We plug our 2D-TV solver directly into SALSA to obtain a fast anisotropic deconvolution algorithm. Table 5 presents numerical results (visual results are in (Barbero & Sra, 2011)) for deconvolution of noisy barcode images subjected to motion blurring. Comparing against SALSA's default isotropic denoising operator, again an anisotropic model produces a better reconstruction. Results for Richardson-Lucy (RL) (Biggs & Andrews, 1997) as implemented in Matlab are also presented, showing much faster filtering times but inferior reconstruction quality.

Table 5. Deconvolution results for anisotropic and isotropic models using the SALSA solver, and MATLAB's Richardson-Lucy (RL) method. ISNR (dB) values and runtimes (in secs) are shown.

	ANISOTROPIC		ISOTROPIC		RL	
n	ISNR	TIME	ISNR	TIME	ISNR	TIME
100	1.55	1.19	1.10	0.12	0.73	0.04
175	2.79	0.81	2.15	0.55	0.79	0.18
300	4.07	3.34	3.07	2.40	1.07	0.46
375	4.05	5.41	2.92	3.71	1.13	0.61
500	3.21	8.98	2.37	5.71	1.04	1.26

Acknowledgments

With partial support from Spain's TIN 2010-21575-C02-01 and FPU–MEC grant reference AP2006–02285.

References

Afonso, Manya V., Bioucas-Dias, José M., and Figueiredo, Mário A. T. Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 19(9), September 2010.

Anderson, E. et al. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. ISBN 0-89871-447-8 (paperback).

Barbero, Á. and Sra, S. *http://people.kyb.tuebingen.mpg.de /suvrit/work/icml11.pdf*, 2011.

Beck, A. and Teboulle, M. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM J. Imgaging Sciences*, 2(1):183–202, 2009.

Bertsekas, Dimitri P. Projected newton methods for optimization problems with simple constraints. *SIAM J. Control and Optimization*, 20(2), March 1982.

Biggs, David S. C. and Andrews, Mark. Acceleration of iterative image restoration algorithms. *Applied Optics*, 36(8), March 1997.

Bioucas-Dias, José M. and Figueiredo, Mário A. T. A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16(12): 2992–3004, December 2007.

Byrd, Richard H., Lu, Peihuang, Nocedal, Jorge, and Zhu, Ciyou. A limited memory algorithm for bound constrained optimization. Technical report, Northwestern University, 1994.

Choksi, Rustum, van Gennip, Yves, and Oberman, Adam. Anisotropic total variation regularized l1-approximation and denoising/deblurring of 2d bar codes. Technical report, Department of Mathematics and Statistics, McGill University, July 2010.

Combettes, Patrick L. and Pesquet, Jean-Christophe. Proximal splitting methods in signal processing. *arXiv*, 2009.

Conn, A. R., Gould, N. I. M., and Toint, P. L. *Trust-Region Methods*. SIAM, 2000.

Dahl, Joachim, Hansen, Per Christian, Jensen, Søren Holdt, and Jensen, Tobias Lindstrøm. Algorithms and software for total variation image reconstruction via first-order methods. *Numer Algor*, (53):67–92, 2010.

Duchi, J. and Singer, Y. Online and Batch Learning using Forward-Backward Splitting. *JMLR*, Sep. 2009.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1 (2):302–332, August 2007.

Golub, T. R. et al. Molecular classification of cancer. *Science*, 286(5439):531–537, October 1999.

Harchaoui, Z. and Lévy-Leduc, C. Multiple Change-Point Estimation With a Total Variation Penalty. *Journal of the American Statistical Association*, 105(492):1480–1493, 2010.

Hua, Jianping, Tembe, Waibhav D., and Dougherty, Edward R. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42:409–424, 2009.

Kim, Dongmin, Sra, Suvrit, and Dhillon, Inderjit. A scalable trust-region algorithm with application to mixed-norm regression. In *ICML*, 2010.

Kolar, M., Song, L., Ahmed, A., and Xing, E. Estimaging time-varying networks. *The Annals of Applied Statistics*, 4(1):94–123, 2010.

Lin, Chih-Jen and Moré, Jorge J. Newton's method for large bound-constrained optimization problems. *SIAM J. Optim.*, 9 (4):1100–1127, 1999.

Liu, J., Ji, S., and Ye, J. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009. *http://www.public.asu.edu/ jye02/Software/SLEP*.

Liu, J., Yuan, L., and Ye, J. An efficient algorithm for a class of fused lasso problems. In *SIGKDD*, 2010.

Moré, Jorge J. and Sorensen, D. C. Computing a trust region step. *SIAM J. Sci. Stat. Comput.*, 4(3), September 1983.

Nesterov, Y. Gradient methods for minimizing composite objective function. Technical Report 76, Catholic University of Louvain, CORE, 2007.

Nocedal, Jorge and Wright, Stephen J. *Numerical Optimization*. Springer Verlag, 2000.

Rapaport, F. and Vert, E. Barillot J.-P. Classification of array-CGH data using fused SVM. *Bioinformatics*, 24(13):i375–i382, 2008.

Rogers, Simon, Girolami, Mark, Campbell, Colin, and Breitling, Rainer. The latent process decomposition of cdna microarray data sets. *IEEE/ACM Trans. Comp. Bio. and Bioinformatics*, 2 (2), April-June 2005.

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

Stransky, Nicolas et al. Regional copy number-independent deregulation of transcription in cancer. *Nature Genetics*, 38 (12):1386–1396, December 2006.

Tibshirani, R. and Wang, P. Spatial smoothing and hot spot detection for CGH data using the fused lasso. *Biostatistics*, 9(1): 18–29, 2008.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. Sparsity and smoothness via the fused lasso. *J. Royal Stat. Soc.: Series B*, 67(1):91–108(18), 2005.

U. Alon et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA*, 96:6745–6750, June 1999.

Vert, J.-P. and Bleakley, K. Fast detection of multiple change-points shared by many signals using group LARS. In *Advances in Neural Information Processing Systems*, 2010.

Vogel, C. R. and Oman, M. E. Iterative methods for total variation denoising. *SIAM J. Sci. Comput.*, 17(1):227–238, 1996.

Zhu, Mingqiang and Chan, Tony. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. Technical report, UCLA CAM, 2008.