A scalable trust-region algorithm with application to mixed-norm regression

Dongmin Kim[*] DMKIM@CS.UTEXAS.EDU
Suvrit Sra[†] SUVRIT@TUEBINGEN.MPG.DE
Inderjit Dhillon[*] INDERJIT@CS.UTEXAS.EDU

[*]Dept. of Computer Science, University of Texas, Austin, TX 78712, USA

[†]Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076, Tübingen, Germany

Abstract

We present a new algorithm for minimizing a convex loss-function subject to regularization. Our framework applies to numerous problems in machine learning and statistics; notably, for sparsity-promoting regularizers such as ℓ_1 or $\ell_{1,\infty}$ norms, it enables efficient computation of sparse solutions. Our approach is based on the trust-region framework with nonsmooth objectives, which allows us to build on known results to provide convergence analysis. We avoid the computational overheads associated with the conventional Hessian approximation used by trust-region methods by instead using a simple *separable* quadratic approximation. This approximation also enables use of proximity operators for tackling nonsmooth regularizers. We illustrate the versatility of our resulting algorithm by specializing it to three mixed-norm regression problems: group lasso [36], group logistic regression [21], and multi-task lasso [19]. We experiment with both synthetic and real-world large-scale data—our method is seen to be competitive, robust, and scalable.

1. Introduction

We present a new algorithm for solving optimization problems of the form

$$\min_{\boldsymbol{w}} \quad \Phi(\boldsymbol{w}) = L(\boldsymbol{w}) + R(\boldsymbol{w}), \tag{1.1}$$

where L is a continuously differentiable, convex loss-function, and R is a convex and continuous, usually non-differentiable regularizer. This generic setup enjoys great importance as it encompasses several basic problems in machine learning and statistics: e.g., Lasso [29], Group Lasso [36], ℓ_1-logistic regression [16], and multi-task feature selection [25].

Since Φ is convex, a variety of methods can be used to solve (1.1)—ranging from the subgradient method [5] to sophisticated interior point methods [24]; we summarize several approaches in §2. Standard methods, however, do not always scale to problem sizes encountered in machine learning, where datasets with several million points are not uncommon. Lack of scalability is even more pronounced when R is non-differentiable, so that many otherwise scalable methods can also become unbearably slow. To overcome some of the challenges posed by (1.1), we design and analyze a new trust-region (TR) algorithm, which we call TRIP. For concrete instances of (1.1), TRIP has a worst-case runtime bounded by $O(1/\epsilon)$ where ϵ denotes the desired solution accuracy (Theorem 3.5).

In contrast to conventional trust-region setups, TRIP attains greater scalability by using only first-order information at the current iteration to build a scalar approximation to the Hessian. This strategy enables the use of proximity operators (§3.2), which allow tackling nonsmooth regularizers efficiently. As a result, TRIP is highly competitive with state-of-the-art methods. We illustrate TRIP's empirical effectiveness by applying it to three important regression problems regularized by mixed-norms (e.g., $\ell_{1,2}$, $\ell_{1,\infty}$): group lasso (GLASSO) [3, 36]; group lasso for logistic regression (GL-LR) [21]; and multi-task lasso (MTL) [19]. These three problems are not only widely applicable but also representative of modern challenging problems that are solvable via our method—see §4 for details.

In summary, our algorithm TRIP is (i) *scalable*, as it uses separable quadratic approximation; (ii) *flexible*, as it depends only on proximity operators for solving subproblems; and (iii) *robust*, because despite using several trust-region parameters, it requires minimal tuning. Empirical results (see §4) on both synthetic and real-world data corroborate these claims.

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

2. Related Work & Background

The simplest approach to solving (1.1) is perhaps the subgradient method [5]. Here one iterates $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta^t \boldsymbol{g}^t$; η^t is a step-size and $\boldsymbol{g}^t$ is a subgradient in $\partial\Phi(\boldsymbol{w}^t)$. This method is, however, overly simple: its iterates are hardly ever at the points of non-differentiability [28, 11], which are usually the actual points of interest as they tend to reveal problem structure such as sparsity [37, 2]. As noted in [28], a similar fate is met by stochastic gradient descent.

A more attractive choice is to tackle non-differentiability of R via proximity operators (see [6], §3.2). These operators are central to forward-backward splitting methods (FBS) [6] (FBS includes methods such as *iterative shrinkage-thresholding* as special cases), as well as to methods based on surrogate optimization [13, 10], and separable approximation [34]. If even the loss is non-differentiable, then the recent FBS method FOBOS [11] may be used. A potential drawback of FBS methods is that often the step-sizes must be carefully tuned. In contrast, our method TRIP usually succeeds without user intervention.

In recent development for mixed-norm problems, block-coordinate descent procedures have been popular: e.g., for separable regularizers [30]; for MTL [19], GLASSO [36], and GL-LR [22]. However, for large-scale data, coordinate-wise approaches are often non-competitive [32, 28].

Beyond FBS there are two more noteworthy methods: SpaRSA [34] and PQN [27]. SpaRSA is a state-of-the-art framework for solving (1.1); our method shares its separable Hessian approximation, but differs significantly with respect to the convergence analysis. PQN is a limited memory quasi-Newton method with excellent empirical performance, but tackles R by passing to the constrained form $R(\boldsymbol{w}) \leq \gamma$. We compare TRIP with FBS, SpaRSA, PQN, and observe highly competitive empirical behavior.

2.1. Nonsmooth Trust-Region Method

Broadly viewed, our method is a variant of the basic trust-region (TR) method which has been successfully applied to some machine learning problems [18, 35]. Like the basic TR method, *the TR method for nonsmooth* Φ also consists of three main steps: (i) approximate the objective function within a *trust-region*[1] via a model function; (ii) minimize the resulting model; and (iii) update the current iterate and the trust-region. Here are some details on these steps.

Let the current iterate be $\boldsymbol{w}^k$. The model function m is designed to satisfy

$$m(\boldsymbol{w}^k, \boldsymbol{p}^k, \boldsymbol{s}) \approx \Phi(\boldsymbol{w}^k + \boldsymbol{s}), \quad \text{for } \|\boldsymbol{s}\| \leq \Delta^k,$$

where $\boldsymbol{p}^k$ parametrizes the model and Δ^k defines the trust-region. Since Φ is non-differentiable a conventional choice such as a second-order Taylor approximation is not directly applicable (see [8, Chapter 11] for more technical details).

Once a proper model has been built, we minimize it to obtain a "step" (minimizer) $\boldsymbol{s}^k$, and then test whether this step is acceptable by computing the ratio

$$\rho_k = \frac{\Phi(\boldsymbol{w}^k) - \Phi(\boldsymbol{w}^k + \boldsymbol{s}^k)}{m(\boldsymbol{w}^k, \boldsymbol{p}^k, \boldsymbol{0}) - m(\boldsymbol{w}^k, \boldsymbol{p}^k, \boldsymbol{s}^k)}, \tag{2.1}$$

which roughly measures the *accuracy* of m. When decrease in the model m matches that in the objective Φ, then $\rho_k \approx 1$ and the trust-region can be increased; if ρ_k is too small, the trust-region must be shrunk. More formally, let $0 < \eta_1 \leq \eta_2 < 1$, $0 < \frac{1}{\gamma_3} \leq \gamma_1 \leq \gamma_2 < 1 < \gamma_3$, and $\Delta_U > 0$ be TR parameters [8]. If $\rho_k \geq \eta_1$, then we update $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \boldsymbol{s}^k$, while the TR is updated via some

$$\Delta^{k+1} \in \begin{cases} [\gamma_1\Delta^k, \gamma_2\Delta^k], & \text{if } \rho_k < \eta_1, \\ [\gamma_2\Delta^k, \Delta^k], & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_3\Delta^k, \Delta_U], & \text{if } \rho_k \geq \eta_2. \end{cases} \tag{2.2}$$

3. The Trust-Region Proximal method: TRIP

Now we are ready to present our approach: the Trust-RegIon Proximal method, or TRIP for short. A crucial feature of TRIP that differentiates it from other TR methods is the particular model function that we use.

Normally, the model function captures second-order information of the overall objective Φ. We do not approximate Φ directly, but instead approximate only L, that too using essentially first-order information. More precisely, we replace the Hessian of L by a *separable* quadratic approximation, which reduces minimization of the model function to applying a proximity operator. Though our simplification sacrifices superior theoretical convergence rates of second-order methods, it does so in exchange for cheaper per-iteration cost. Such tradeoffs are common to machine learning applications, and are often crucial to tackling large-scale data.

3.1. The model function

For *differentiable* Φ, a typical second-order model is

$$m(\boldsymbol{w}^k, \boldsymbol{H}^k, \boldsymbol{s}) = \Phi(\boldsymbol{w}^k) + \left\langle \boldsymbol{s}, \nabla\Phi(\boldsymbol{w}^k) \right\rangle + \tfrac{1}{2}\left\langle \boldsymbol{s}, \boldsymbol{H}^k \boldsymbol{s} \right\rangle,$$

where $\boldsymbol{H}^k$ is a positive-definite matrix that captures curvature information (e.g., for twice-differentiable Φ, one could have $\boldsymbol{H}^k = \nabla^2\Phi(\boldsymbol{w}^k)$). Since $\Phi = L + R$ where only L is differentiable, a second-order model could be

$$\begin{aligned} m(\boldsymbol{w}^k, \boldsymbol{H}^k, \boldsymbol{s}) = L(\boldsymbol{w}^k) + \left\langle \boldsymbol{s}, \nabla L(\boldsymbol{w}^k) \right\rangle + \\ \tfrac{1}{2}\left\langle \boldsymbol{s}, \boldsymbol{H}^k \boldsymbol{s} \right\rangle + R(\boldsymbol{w}^k + \boldsymbol{s}), \end{aligned} \tag{3.1}$$

[1] The trust-region captures a neighborhood of the current iterate where minimizing a model function, more or less guarantees minimization of the original objective.

where $\boldsymbol{H}^k$ now captures curvature information for $L(\boldsymbol{w}^k)$. However, the matrix $\boldsymbol{H}^k$ makes minimizing (3.1) non-trivial, so we wish to replace it with a simpler choice.

An immediate choice is inspired by quasi-Newton methods that seek a matrix $\boldsymbol{H}^k$ which satisfies the secant equation

$$\boldsymbol{H}^k \boldsymbol{u}^k = \boldsymbol{v}^k, \text{ or equivalently, } \boldsymbol{u}^k = \left(\boldsymbol{H}^k\right)^{-1} \boldsymbol{v}^k, \quad (3.2)$$

where the difference vectors $\boldsymbol{u}^k$ and $\boldsymbol{v}^k$ are defined as

$$\boldsymbol{u}^k = \boldsymbol{w}^k - \boldsymbol{w}^{k-1} \text{ and } \boldsymbol{v}^k = \nabla L(\boldsymbol{w}^k) - \nabla L(\boldsymbol{w}^{k-1}).$$

The secant equation (3.2) can be solved in various ways. One extreme, but surprisingly effective choice is to use the spectral formulae proposed by [4] which yield *scalar* solutions to (3.2) by computing

$$\begin{aligned} \alpha^k &= \operatorname{argmin}_\alpha \|\alpha \boldsymbol{u}^k - \boldsymbol{v}^k\|_2^2, \qquad \text{or} \\ \alpha^k &= \operatorname{argmin}_\alpha \|\boldsymbol{u}^k - \alpha^{-1}\boldsymbol{v}^k\|_2^2, \end{aligned}$$

which result in the closed form solutions

$$\alpha^k = \langle \boldsymbol{u}^k, \boldsymbol{v}^k\rangle / \|\boldsymbol{u}^k\|^2, \quad \text{and} \quad \alpha^k = \|\boldsymbol{v}^k\|^2 / \langle \boldsymbol{u}^k, \boldsymbol{v}^k\rangle. \quad (3.3)$$

The scalars α^k yield diagonal Hessian approximations $\alpha^k \boldsymbol{I}$. This is one of the key features of SpaRSA [34] and we too adopt it in our method. However, we note that this α^k might be unbounded for our problem (1.1), hence we impose a pre-specified lower bound α_L to keep $\alpha^k \boldsymbol{I}$ positive-definite, and an upper bound α_U to prevent degeneracy.

Finally, we plug either choice of α^k into (3.1) to obtain our *separable* quadratic model,

$$\begin{aligned} m(\boldsymbol{w}^k, \alpha^k, \boldsymbol{s}) = L(\boldsymbol{w}^k) + \big\langle \boldsymbol{s}, \nabla L(\boldsymbol{w}^k)\big\rangle + \\ \tfrac{1}{2}\alpha^k \|\boldsymbol{s}\|_2^2 + R(\boldsymbol{w}^k + \boldsymbol{s}). \end{aligned} \quad (3.4)$$

Minimization of (3.4) can be written as

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{s}} m(\boldsymbol{w}^k, \alpha^k, \boldsymbol{s}) \\ &= \operatorname{argmin}_{\boldsymbol{s}} \tfrac{1}{2}\|\boldsymbol{s} + \tfrac{1}{\alpha^k}\nabla L(\boldsymbol{w}^k)\|_2^2 + \tfrac{1}{\alpha^k} R(\boldsymbol{w}^k + \boldsymbol{s}), \end{aligned} \quad (3.5)$$

Problem (3.5) is recognized to be a *proximity operator* (for R) [7, 6], and since it plays a crucial role in our method, let us look at proximity operators in more detail.

3.2. Proximity operators

Let $R : X \subseteq \mathbb{R}^d \to (-\infty, \infty]$ be a lower semicontinuous, proper convex function. For a point $\boldsymbol{v} \in X$, the *proximity operator* for R applied to $\boldsymbol{v}$ is defined as

$$\operatorname{Prox}_R \boldsymbol{v} = \operatorname*{argmin}_{\boldsymbol{s} \in X} \quad \tfrac{1}{2}\|\boldsymbol{s} - \boldsymbol{v}\|_2^2 + R(\boldsymbol{s}). \quad (3.6)$$

Several examples of proximity operators are also discussed in [11], and some additional details on proximity operators may be found in [7]; one important property that we use is *Moreau's decomposition*:

$$\boldsymbol{v} = \operatorname{Prox}_R \boldsymbol{v} + \operatorname{Prox}_{R^*} \boldsymbol{v}, \quad (3.7)$$

where R^* is the Fenchel conjugate to R. For R equal to ℓ_1, ℓ_2, or ℓ_∞ norms, after simple algebra (3.7) yields

$$\operatorname{Prox}_{\lambda\|\cdot\|_1} \boldsymbol{v} = \operatorname{sgn}(\boldsymbol{v}) \odot (|\boldsymbol{v}| - \lambda)_+, \quad (3.8)$$

$$\operatorname{Prox}_{\lambda\|\cdot\|_2} \boldsymbol{v} = (1 - \|\boldsymbol{v}\|_2^{-1}\lambda)_+ \boldsymbol{v}, \quad (3.9)$$

$$\operatorname{Prox}_{\lambda\|\cdot\|_\infty} \boldsymbol{v} = \boldsymbol{v} - \mathrm{P}_{\|\cdot\|_1 \le \lambda} \boldsymbol{v}, \quad (3.10)$$

where $(x)_+ = \max(x, 0)$. The first two can be computed in $O(n)$ time; (3.10) too can be computed in $O(n)$ time by using the algorithm of [12] for the projection $\mathrm{P}_{\|\cdot\|_1 \le \lambda}(\boldsymbol{v})$.

3.3. The Algorithm

We now have all the basic ingredients needed to present TRIP—Algorithm 1 provides the pseudo-code.

TRIP operates in two phases: *null* and *monotonic*. In the null phase, we solve (3.5) using appropriate proximity operators. But the null phase relies on the separable model (3.4), which is based on spectral formulae that exhibit a characteristic non-monotonic behavior [4]. This non-monotonicity, while empirically very effective, makes it extremely difficult to analyze and prove convergence. The difficulty essentially stems from potential violation of the following descent condition:

Definition 3.1 (Descent)**.** Let $m(\boldsymbol{w}^k, \boldsymbol{p}^k, \boldsymbol{s}^k)$ be the model function, and $\boldsymbol{g}^*$ the *minimum norm generalized gradient*[2] at $\boldsymbol{w}^k$. The *descent condition* requires $\boldsymbol{s}^k$ to satisfy

$$\begin{aligned} m(\boldsymbol{w}^k, \boldsymbol{p}^k, \boldsymbol{0}) - m(\boldsymbol{w}^k, \boldsymbol{p}^k, \boldsymbol{s}^k) \\ > \sigma \|\boldsymbol{g}^*\| \min\{\Delta_L, \Delta^k\}, \end{aligned} \quad (3.11)$$

where $\sigma \in (0, 1)$, and Δ_L, Δ^k denote a pre-specified lower-bound and the current trust-region size, respectively.

The monotonic phase of the algorithm ensures (3.11), so that the adverse null iterations can be "corrected." To this end, we limit the number of allowable *consecutive* non-monotonic iterations. At any moment, the algorithm maintains a *reference iteration* $\boldsymbol{w}^r$, for which the last descent occurred. From $\boldsymbol{w}^r$ onwards the algorithm may move non-monotonically for at most $\bar{r}$ iterations. If there is descent, the reference iteration is reset, while if for $\bar{r}$ iterations the method fails to descend, it enters the monotonic phase that guarantees (3.11). Consequently, we can guarantee convergence of the overall algorithm, as shown below.

[2]Defined as $\boldsymbol{g}^* = \operatorname{argmin}_{\boldsymbol{g} \in \partial_G \Phi(\boldsymbol{w})} \|\boldsymbol{g}\|$, where $\partial_G \Phi(\boldsymbol{w})$ is the set of *generalized gradients* at $\boldsymbol{w}$, i.e., vectors $\boldsymbol{g} \in \partial_G \Phi(\boldsymbol{w})$, s.t., $\forall \boldsymbol{d} \in \mathbb{R}^n$, $\langle \boldsymbol{g}, \boldsymbol{d}\rangle \le \lim_{t \to 0+} \sup_{\boldsymbol{y} \to \boldsymbol{w}} \frac{\Phi(\boldsymbol{y} + t\boldsymbol{d}) - \Phi(\boldsymbol{y})}{t}$

Algorithm 1 Trust-RegIon Proximal method (TRIP).

```
input TR parameters (2.2), descent parameters (3.11),
  model parameters $\alpha_L, \alpha_U$ (3.4) and $\tau \in (0,1)$
  Initialize $k \leftarrow 0,\ r \leftarrow 0,\ \Delta \leftarrow \Delta^0$, and
      $\boldsymbol{w}^k, \boldsymbol{w}^{k-1} \in \mathbb{R}^n$ such that $\Phi(\boldsymbol{w}^k) < \Phi(\boldsymbol{w}^{k-1})$
  repeat
    $\boldsymbol{s}^k \leftarrow -\boldsymbol{w}^k + \mathrm{Prox}_{R/\alpha^k}\left[\boldsymbol{w}^k - (\alpha^k)^{-1}\nabla L(\boldsymbol{w}^k)\right]$
    if $k - r < \bar{r}$ then
      {Potential non-monotonic descent}
      $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \boldsymbol{s}^k$; Compute $\alpha^{k+1}$;
      if $\Phi(\boldsymbol{w}^k + \boldsymbol{s}^k) < \Phi(\boldsymbol{w}^r)$ then
        $r \leftarrow k$; {Update reference iterate}
      end if
      $k \leftarrow k+1$; Continue;
    end if
    {Enforce monotonicity}
    repeat
      $\boldsymbol{s}^k \leftarrow -\mathrm{P}_{\|\cdot\| \le \Delta^k}\left[(\alpha^k)^{-1}\boldsymbol{g}^*\right]$;   $\alpha^k \leftarrow \tau\alpha^k$
    until (3.11) is satisfied
    Compute $\rho_k$ and update TR using (2.2);
    Update $\alpha^{k+1}$;   $r \leftarrow k$;   $k \leftarrow k+1$
  until Stopping criteria met
```

3.4. Convergence Analysis of TRIP

Before we state the main convergence theorem for TRIP, we need to prove a technical but crucial proposition.

Proposition 3.2. *The following properties hold:*

(1) The set of model parameters Λ is closed and bounded.

(2) The function $\Phi(\boldsymbol{w})$ from (1.1) *is locally Lipschitz continuous and regular on $\mathbb{R}^n$.*

(3) The model $m(\boldsymbol{w}, \alpha, \boldsymbol{s})$ from (3.4) *is locally Lipschitz continuous and regular with respect to $\boldsymbol{s}$ for all $(\boldsymbol{w}, \alpha) \in \mathbb{R}^n \times \Lambda$, and continuous in $(\boldsymbol{w}, \alpha)$ for all $\boldsymbol{s} \in \mathbb{R}^n$.*

(4) The values of the objective function and of the model coincide when $\boldsymbol{s} = \boldsymbol{0}$, i.e., $m(\boldsymbol{w}, \alpha, \boldsymbol{0}) = \Phi(\boldsymbol{w})$ for all $(\boldsymbol{w}, \alpha) \in \mathbb{R}^n \times \Lambda$.

(5) The generalized directional derivatives of the model and of the objective function coincide in every nonzero direction $\boldsymbol{d}$ when $\boldsymbol{s} = \boldsymbol{0}$ for all $(\boldsymbol{w}, \alpha) \in \mathbb{R}^n \times \Lambda$.

Proof: 3.2 (1). Immediate because $\Lambda = [\alpha_L, \alpha_U]$.

Proof: 3.2 (2). Since by definition both L and R are convex, continuous, the objective $\Phi = L + R$ is locally Lipschitz continuous [26], and also regular [1].

Proof: 3.2(3). Recall that the model $m(\boldsymbol{w}, \alpha, \boldsymbol{s})$ is

$$m(\boldsymbol{w}, \alpha, \boldsymbol{s}) = L(\boldsymbol{w}) + \langle \boldsymbol{s}, \nabla L(\boldsymbol{w})\rangle + \tfrac{1}{2}\alpha\|\boldsymbol{s}\|_2^2 + R(\boldsymbol{w}+\boldsymbol{s})$$

Each of the above terms on the right hand side of m is convex and continuous w.r.t. $\boldsymbol{s}$. Thus, $m(\boldsymbol{w}, \alpha, \boldsymbol{s})$ is convex and continuous, and thereby also locally Lipschitz continuous and regular [26, 1] in $\boldsymbol{s}$. Furthermore, given $\boldsymbol{s}$ it is easy to check that all the terms are continuous in $(\boldsymbol{w}, \alpha)$ (recall that L is continuously differentiable), whereby m itself is continuous in $(\boldsymbol{w}, \alpha)$ for all $\boldsymbol{s} \in \mathbb{R}^n$.

To prove 3.2(4) and 3.2(5) we use the following lemma.

Lemma 3.3 ([8, Lemma 11.2.1])**.** *If 3.2(2) and 3.2(3) hold, then the following limit*

$$\lim_{\|\boldsymbol{s}\| \to 0} \frac{\Phi(\boldsymbol{w}+\boldsymbol{s}) - m(\boldsymbol{w}, \alpha, \boldsymbol{s})}{\|\boldsymbol{s}\|} = 0, \tag{3.12}$$

is equivalent to the combination of 3.2(4) and 3.2(5).

We now prove 3.2(4), 3.2(5) by showing that (3.12) holds.

Proof: 3.2(4,5). By definition of $m(\boldsymbol{w}, \alpha, \boldsymbol{s})$ we have

$$\begin{aligned}
&\lim_{\|\boldsymbol{s}\| \to 0} \frac{\Phi(\boldsymbol{w}+\boldsymbol{s}) - m(\boldsymbol{w}, \alpha, \boldsymbol{s})}{\|\boldsymbol{s}\|} \\
&= \lim_{\|\boldsymbol{s}\| \to 0} \frac{L(\boldsymbol{w}+\boldsymbol{s}) + R(\boldsymbol{w}+\boldsymbol{s})}{\|\boldsymbol{s}\|} \\
&\quad - \frac{L(\boldsymbol{w}) + \langle \boldsymbol{s}, \nabla L(\boldsymbol{w})\rangle + \frac{1}{2}\alpha\|\boldsymbol{s}\|_2^2 + R(\boldsymbol{w}+\boldsymbol{s})}{\|\boldsymbol{s}\|} \\
&= \lim_{\|\boldsymbol{s}\| \to 0} \frac{L(\boldsymbol{w}+\boldsymbol{s}) - L(\boldsymbol{w})}{\|\boldsymbol{s}\|} - \lim_{\|\boldsymbol{s}\| \to 0} \frac{\langle \boldsymbol{s}, \nabla L(\boldsymbol{w})\rangle}{\|\boldsymbol{s}\|} \\
&\quad - \lim_{\|\boldsymbol{s}\| \to 0} \frac{\alpha\|\boldsymbol{s}\|_2^2}{2\|\boldsymbol{s}\|}.
\end{aligned} \tag{3.13}$$

Let $\{\boldsymbol{s}^k\}$ be a sequence converging to zero; let $h^k = \|\boldsymbol{s}^k\|$, and $\boldsymbol{d}^k = \boldsymbol{s}^k/h^k$. Then,

$$\lim_{k\to\infty} h^k = 0 \quad \text{while} \quad \|\boldsymbol{d}^k\| = 1, \quad \forall k.$$

Using this notation we rewrite (3.13) as

$$\begin{aligned}
&\lim_{k\to\infty} \frac{L(\boldsymbol{w}+h^k\boldsymbol{d}^k) - L(\boldsymbol{w})}{h^k\|\boldsymbol{d}^k\|} - \lim_{k\to\infty} \frac{h^k\left\langle \boldsymbol{d}^k, \nabla L(\boldsymbol{w})\right\rangle}{h^k\|\boldsymbol{d}^k\|} \\
&- \lim_{k\to\infty} \frac{h^k\alpha\|\boldsymbol{d}^k\|_2^2}{2\|\boldsymbol{d}^k\|}.
\end{aligned}$$

Since L is continuously differentiable we obtain

$$\lim_{k\to\infty} \frac{\left\langle \boldsymbol{d}^k, \nabla L(\boldsymbol{w})\right\rangle}{\|\boldsymbol{d}^k\|} - \lim_{k\to\infty} \frac{\left\langle \boldsymbol{d}^k, \nabla L(\boldsymbol{w})\right\rangle}{\|\boldsymbol{d}^k\|} - \lim_{k\to\infty} h^k \cdot C = 0,$$

where $C = \alpha\|\boldsymbol{d}^k\|_2^2/2\|\boldsymbol{d}^k\|$. Thus (3.12) holds. □

With Proposition 3.2 in hand, the global convergence of TRIP can be established provided that the sequence $\{\boldsymbol{w}^r\}$ of reference iterations has a limit point. This can be ensured using some additional assumptions on Φ, for instance, when Φ is bounded below and has a bounded level set. In this paper, we assume the existence of a limit point and show that it is first-order critical.

Theorem 3.4 (Convergence)**.** *Suppose $\boldsymbol{w}^*$ is a limit point of the sequence $\{\boldsymbol{w}^r\}$ generated by Algorithm 1. Then $\boldsymbol{w}^*$ is a first-order critical point of $\Phi(\boldsymbol{w})$.*

Proof. Since Proposition 3.2 holds, the TR framework of Theorem 11.2.5 in [8] yields the proof. □

Let us now look at TRIP's convergence rate.

Theorem 3.5 (Convergence rate)**.** *The sequence $\{\Phi(\boldsymbol{w}^r)\}$ generated by Algorithm 1 converges at the rate $O(1/r)$.*

Proof sketch. Suppose $\boldsymbol{w}^*$ is the solution to (1.1). For brevity, denote $\Phi(\boldsymbol{w}^r)$ by Φ^r, and $m(\boldsymbol{w}^r, \alpha^r, \boldsymbol{s}^r)$ by m^r. Since $\Phi^{r+1} < \Phi^r$, there exists a constant ξ such that $\|\boldsymbol{w}^r - \boldsymbol{w}^*\| \leq \xi$ for all $\boldsymbol{w}^r$. From (2.1), the computation of ρ, and from the trust-region update (2.2), we have

$$\frac{\Phi^r - \Phi^{r+1}}{\Phi^r - m^r} \geq \eta_1, \quad \text{that is} \quad \Phi^{r+1} \leq (1-\eta_1)\Phi^r + \eta_1 m^r.$$

Following Nesterov [23] we see that if $\Phi^0 - \Phi^* \geq \eta_1 \alpha_U \xi^2$, then $\Phi^1 - \Phi^* \leq \eta_1 \alpha_U \xi^2/2$, otherwise

$$\Phi^{r+1} - \Phi^* \leq \Phi^r - \Phi^* - \eta_1 \frac{(\Phi^r - \Phi^*)^2}{2\alpha_U \xi^2}. \tag{3.14}$$

Using the shorthand $\theta^r = 1/(\Phi^r - \Phi^*)$, (3.14) yields

$$\frac{1}{\theta^{r+1}} \leq \frac{1}{\theta^r} - \frac{\eta_1}{2\alpha_U \xi^2 (\theta^r)^2} = \frac{2\alpha_U \xi^2 \theta^r - \eta_1}{2\alpha_U \xi^2 (\theta^r)^2},$$

which further implies that

$$\begin{aligned} \theta^{r+1} &\geq \frac{2\alpha_U \xi^2 (\theta^r)^2}{2\alpha_U \xi^2 \theta^r - \eta_1} = \theta^r + \frac{\eta_1 \theta^r}{2\alpha_U \xi^2 \theta^r - \eta_1} \\ &> \theta^r + \frac{\eta_1 \theta^r}{2\alpha_U \xi^2 \theta^r} = \theta^r + \frac{\eta_1}{2\alpha_U \xi^2}. \end{aligned}$$

Repeating the argument for all t we immediately obtain

$$\theta^r \geq \theta^{r-1} + \frac{\eta_1}{2\alpha_U \xi^2} \geq \cdots \geq \theta^0 + \frac{\eta_1 r}{2\alpha_U \xi^2} > \frac{\eta_1 r + 2}{2\alpha_U \xi^2},$$

which yields the desired convergence rate

$$\Phi^r - \Phi^* < \frac{2\alpha_U \xi^2}{\eta_1 r + 2} < \frac{2\alpha_U \xi^2}{\eta_1} \cdot \frac{1}{r} = O(1/r). \quad \square$$

3.5. Implementation

At each iteration of TRIP, we need to apply proximity operators for mixed-norms (null phase) or compute minimum-norm generalized gradients (monotonic phase). Proximity operators can be applied efficiently as the task breaks down into n separate operators for GLASSO and GL-LR (d for MTL)—a point also noticed by [34, 11]. The monotonic phase requires minimizing the generalized gradients $\|\boldsymbol{g}\|_1$, $\|\boldsymbol{g}\|_2$ or $\|\boldsymbol{g}\|_\infty$—a task that is easy but having a somewhat technical derivation, so we defer it to [15, §2.1] for brevity.

4. Experiments

In this section we apply TRIP to the following three mixed-norm[3] regression problems:

1. **Group lasso** (GLASSO) [36]. Let $\boldsymbol{X}_j \in \mathbb{R}^{m \times d_j}$, $\boldsymbol{w}_j \in \mathbb{R}^{d_j}$ ($1 \leq j \leq n$), $\boldsymbol{y} \in \mathbb{R}^m$, and $\lambda > 0$. Then, the GLASSO problem ($\ell_{1,2}$-regularized) is:
$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_n} \tfrac{1}{2}\|\boldsymbol{y} - \sum\nolimits_{j=1}^n \boldsymbol{X}_j \boldsymbol{w}_j\|_2^2 + \lambda \sum\nolimits_{j=1}^n \|\boldsymbol{w}_j\|_2. \tag{4.1}$$
Formulation (4.1) is the most frequently studied variant of GLASSO; see for instance [36, 2].
2. **Group-lasso for logistic regression** (GL-LR) [21]. Same as GLASSO except that instead of a quadratic loss, the logistic loss
$$L(\boldsymbol{w}) = \boldsymbol{y}^T \boldsymbol{\eta} - \sum\nolimits_{i=1}^m \log\bigl(1 + e^{\eta_i}\bigr), \tag{4.2}$$
is used; $\boldsymbol{\eta} = b\mathbf{1} + \sum_{j=1}^n \boldsymbol{X}_j \boldsymbol{w}_j$ for a bias term b.
3. **Multi-task lasso** (MTL) [19]. Let $\boldsymbol{X}_j \in \mathbb{R}^{m_j \times d}$, and parameter matrix $\boldsymbol{W} \in \mathbb{R}^{d \times n}$. Each column $\boldsymbol{w}_j$ of $\boldsymbol{W}$ corresponds to a task, but the mixed-norm is applied over the rows $\boldsymbol{w}^i$ of $\boldsymbol{W}$. The optimization problem is:
$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_n} \sum\nolimits_{j=1}^n \tfrac{1}{2}\|\boldsymbol{y}_j - \boldsymbol{X}_j \boldsymbol{w}_j\|_2^2 + \lambda \sum\nolimits_{i=1}^d \|\boldsymbol{w}^i\|_\infty, \tag{4.3}$$
A more general setup is described in [25].

These three problems and their variations have attracted a lot of attention. Notable applications include: feature-selection [36]; groupwise, multi-task feature-selection [25, 31, 38]; cognitive neuroscience [19]; bioinformatics [20]; computer vision [33]. We conduct experiments on both synthetic and real-world datasets and compare four algorithms:[4] TRIP, PQN,[5] SpaRSA,[6] and FBS[7]. Though we focus on the three applications listed above, we remark in passing that TRIP also easily applies to other problems such as lasso, ℓ_1-penalized logistic-regression [16], Elastic-net [39], and sparse group lasso [14].

4.1. Synthetic Data

Table 1 summarizes the synthetic datasets used. Due to space limitations, we include only a few of our results

[3]Let $\boldsymbol{w} \in \mathbb{R}^d$ be partitioned into subvectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$, where $\boldsymbol{w}_j \in \mathbb{R}^{d_j}$ for $1 \leq j \leq n$. The $\ell_{p,q}$ *mixed-norm* (aka group-norm) for $\boldsymbol{w}$ is defined as

$$\|\boldsymbol{w}\|_{p,q} = \bigl\| \, [\|\boldsymbol{w}_1\|_q; \|\boldsymbol{w}_2\|_q; \ldots, \|\boldsymbol{w}_n\|_q] \, \bigr\|_p.$$

The most commonly used mixed-norms are $\ell_{1,2}$ and $\ell_{1,\infty}$.

[4]We remark again that block-coordinate descent (BCD) methods do not scale well. For example, the BCD method of [19] requires pre-processing steps that compute $\boldsymbol{X}_j^T \boldsymbol{X}_j$; these products are prohibitively expensive to compute, and demand huge storage—e.g., for dataset S3 (Table 1) more than 1800GB would be needed!

[5]Obtained from: http://people.cs.ubc.ca/ schmidtm/Software/PQN.html

[6]Obtained from: http://www.lx.it.pt/∼mtf/SpaRSA/

[7]Carefully implemented ourselves; no reference implementation was available.

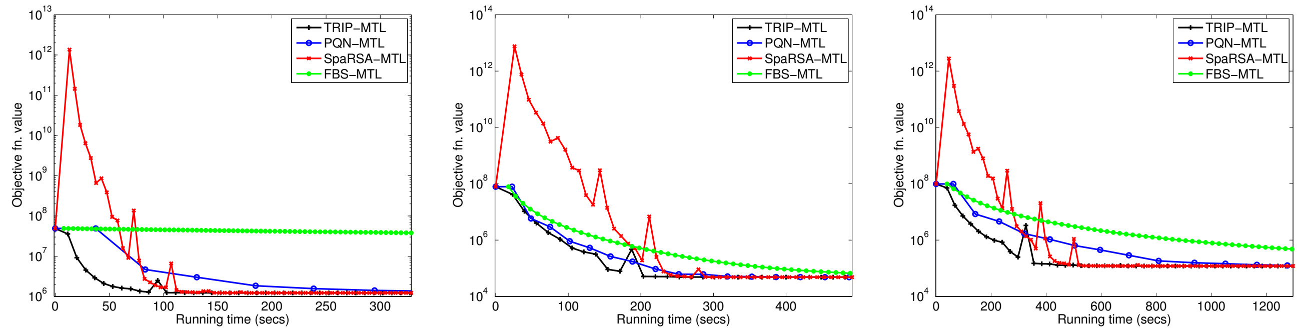

Figure 1. MTL experiments: objective function vs. runtime for datasets S1, S2, and S3. FBS eventually converges (not shown) after a long time. On S2 PQN is very competitive. SpaRSA is seen to be nonmonotonic. Overall TRIP outperforms the other methods.

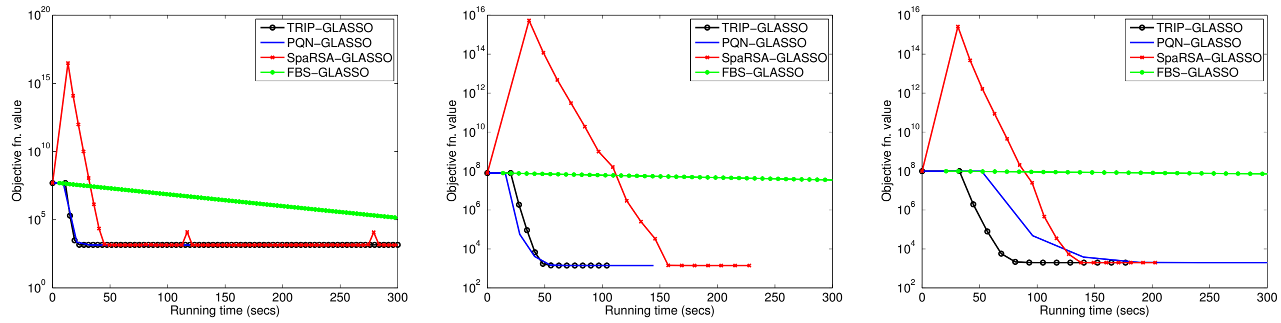

Figure 2. GLASSO experiments: objective function vs. runtime for datasets S1, S2, and S3. FBS eventually converges (not shown) after a long time; TRIP and PQN behave similarly, except on S3 where TRIP outperforms PQN.

(more may be found in [15]). We first show results for MTL. Here, the observations are generated following the model $\boldsymbol{y}_j = \boldsymbol{X}_j \boldsymbol{w}_j + \epsilon_j$, where ϵ_j is Gaussian noise. Figure 1 plots objective values versus runtimes for the four algorithms. We adjusted the parameters for PQN, SpaRSA, and FBS, to make them perform competitively; for FBS we searched over a grid of step-sizes and have reported the best results obtained. We see that TRIP converges the fastest, followed by PQN SpaRSA, and FBS. Notice how SpaRSA exhibits highly non-monotonic behavior.

Next, we show synthetic results for GLASSO. Here the observation was generated as per $\boldsymbol{y} = \sum_{j=1}^{n} \boldsymbol{X}_j \boldsymbol{w}_j + \epsilon$. The data matrices used were the same as for MTL. We again adjusted parameters of the other approaches to make them run competitively. Figure 2 presents objective function versus runtime plots. Here, as opposed to MTL, we observe that PQN and TRIP perform similarly on S1 and S2, while for the larger dataset TRIP converges much faster.

4.2. Real-world Datasets

The main goal of this section is to compare the computational efficiency of the different methods on some real-world datasets, once again with application to MTL, GLASSO, and GL-LR. We run all these regressions for performing explanatory feature selection, a problem for which fast optimization of the objective function is always beneficial. We begin with a brief demonstrative sample, which is followed by the main results.

Table 1. Dimensions of the various synthetic datasets. The values correspond to matrices in n groups, $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \in \mathbb{R}^{m \times d}$. The largest dataset S3 contains over 500 million nonzero elements.

Name	m	d	n	# nonzeros
S1	10,000	5,000	500	$2.49 \cdot 10^8$
S2	50,000	20,000	100	$3.99 \cdot 10^8$
S3	100,000	50,000	100	$5.01 \cdot 10^8$

The first experiment solves MTL for the wine quality dataset [9]. This dataset contains two sub-datasets that measure physicochemical properties of red wines and white wines. The response variable is the quality for each wine, and takes values from 0 to 10.

Table 2 shows the weights of the features selected by linear regression and by MTL. By applying the linear regression *separately*, one obtains (Density, Alcohol) as the most influential features for red wine, and (Density, Residual sugar) for white wine. MTL, however, provides a different explanation: Alcohol is the co-dominating feature to determine the quality, regardless of the wine type, while Volatile

Feature	Linear Regression	MTL $\lambda = 1e + 2$	MTL $\lambda = 3e + 3$
Fixed acidity	0.0253 / 0.0624	0.0386 / -0.0201	0/0
Volatile acidity	**-0.1104 / -0.2120**	**0.0193 / -0.2043**	**0.0768/-0.0768**
Citric acid	-0.0120 / 0.0030	0.0164 / 0.0013	0/0
Residual sugar	0.0037 / 0.4666	0.0389 / 0.2183	0/0
Chlorides	-0.0325 / -0.0061	0.0045 / -0.0045	0/0
Free sulfur dioxide	0.0154 / 0.0717	0.0265 / 0.0521	0/0
Total sulfur dioxide	-0.0335 / -0.0137	0 0	0/0
Density	0.6634 / -0.5075	0.1579 / -0.1579	0/0
pH	-0.1993 / 0.1170	0.0484 / 0.0484	0/0
Sulphates	0.1070 / 0.0814	0.0833 / 0.0604	0/0
Alcohol	**0.5467 / 0.2688**	**0.5604 / 0.4289**	**0.3731/0.3731**

Table 2. Feature selection via multi-task lasso

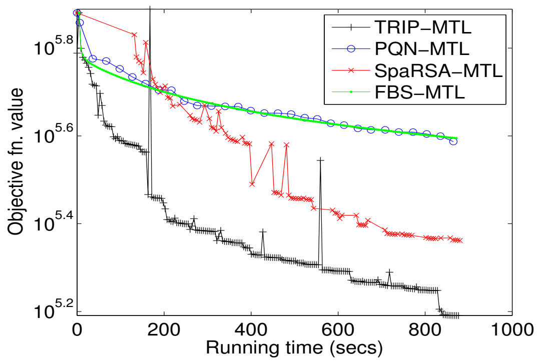

Figure 3. Running time plots. The 20 Newsgroups dataset was pre-processed to fit into the MTL framework. Five newsgroup topics (computer, recreation, science, politics, religion) were reduced to five different lasso tasks of size 2907 $\times$ 53975. The "forsale" group is omitted since it does not have any subgroups.

acidity contrasts different types. Notice that it can be difficult to identify such contrasting features via standard lasso, and it is a unique ability of MTL to be able to do so.

In our next experiment we run MTL on the 20 newsgroups dataset, using five tasks; each task has data of size 2907 $\times$ 53975. As already seen on synthetic data, TRIP minimizes the objective function effectively. Here too, from Figure 3 we see that TRIP generates solution comparable with other methods in a fraction of their elapsed time.

The key difference between MTL and GLASSO is how the grouping happens. In MTL, each feature is grouped across multiple tasks, while in GLASSO, features in a *single* problem are grouped. To compare the methods for GLASSO, we chose the MNIST handwritten digits dataset [17] and formed a GLASSO instance. Specifically, we formulate the problem as selecting a limited number of row-pixel locations and computing the weights for pixels in the selected rows. This setup yields 28 data matrices of size $60,000 \times 28$ with a total of 8,994,516 non-zero entries and an observation vector of size $60,000 \times 1$. Figure 4 shows the running time of various methods on this dataset—we observe that TRIP significantly outperforms the other methods.

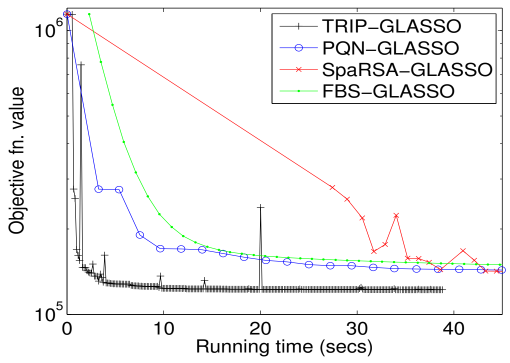

Figure 4. Running time plots on the MNIST handwritten digits dataset. Each image in the dataset is partitioned with respect to its rows, i.e., each 28 $\times$ 28 digit image is considered as a collection of 28 row-pixels. We remark that we had to turn on the 'safeguard' option of SpaRSA to make it converge on this dataset; this generally results in the prolonged first iteration as shown in the figure. Considering that FBS requires extensive parameter tuning, we would still regard SpaRSA as highly competitive, as it rapidly catches up with other methods in a few "normal" iterations.

We also produced a GL-LR problem on the MNIST dataset. We, however, simplified the problem to learning a binary classifier for the digit "1". The result from this GL-LR problem is shown in Figure 5. Here we could not show SpaRSA as the publicly available implementation supports only quadratic losses. TRIP is seen to outperform PQN and FBS, both of which are competitive with each other.

5. Conclusions and Future Work

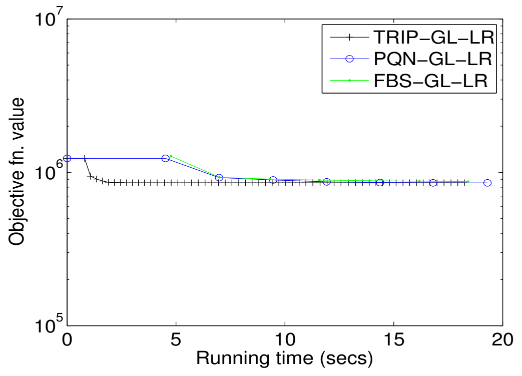

Figure 5. Running time plots for solving a GL-LR problem on the MNIST handwritten digits dataset.

We presented a new method based on a nonsmooth trust-region framework for solving regularized convex problems. We applied our algorithm to three mixed-norm regularized regression problems, for which it exhibited highly competitive performance on both synthetic and real-world datasets, while enjoying solid theoretical convergence properties.

We emphasize that the applicability of our method is not limited to the problems presented in this paper. Future challenges include further algorithmic developments such as employing better, potentially more sophisticated yet scalable models, and developing efficient proximity operators for more complex regularizers, e.g., tree-regularized mixed-norms and overlapping mixed-norms.

References

[1] Aubin, J.-P. *L'analyse non lineaire et ses motivations economiques*. Masson, 1984.

[2] Bach, F. R. Consistency of the Group Lasso and Multiple Kernel Learning. *J. Mach. Learn. Res.*, 9:1179–1225, 2008.

[3] Bakin, S. *Adaptive regression and model selection in data mining problems*. PhD thesis, Australian National University, Canberra, 1999.

[4] Barzilai, J. and Borwein, J. M. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1): 141–148, 1988.

[5] Bertsekas, D. P. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

[6] Combettes, P. L. and Pesquet, J.-C. Proximal Splitting Methods in Signal Processing. Dec. 2009.

[7] Combettes, P. L. and Wajs, V. R. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005.

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

[9] Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 1998.

[10] Daubechies, I., Defrise, M., and Mol, C. De. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 11:1413–1457, 2004.

[11] Duchi, J. and Singer, Y. Online and Batch Learning using Forward-Backward Splitting. *J. Mach. Learning Res. (JMLR)*, Sep. 2009.

[12] Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient Projections onto the ℓ_1-Ball for Learning in High Dimensions. In *ICML*, 2008.

[13] Figueiredo, MAT and Nowak, RD. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

[14] Friedman, J., Hastie, T., and Tibshirani, R. A note on the group lasso and a sparse group lasso. *arXiv:1001.0736v1 [math.ST]*, Jan. 2010.

[15] Kim, D., Sra, S., and Dhillon, I. S. A scalable trust-region algorithm with application to mixed-norm regression: Supplementary Material. In *ICML*, 2010.

[16] Koh, K., Kim, S.-J., and Boyd, S. An Interior-Point Method for Large-Scale ℓ_1-Regularized Logistic Regression. *JMLR*, 8, 2007.

[17] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[18] Lin, C.J., Weng, R.C., and Keerthi, S.S. Trust region newton method for logistic regression. *The Journal of Machine Learning Research*, 9:627–650, 2008.

[19] Liu, H., Palatucci, M., and Zhang, J. Blockwise Coordinate Descent Procedures for the Multi-task Lasso, with Applications to Neural Semantic Basis Discovery. In *Int. Conf. Machine Learning*, Jun. 2009.

[20] Ma, S., Song, X., and Huang, J. Supervised group lasso with applications to microarray data analysis. *BMC Bioinformatics*, 2007.

[21] Meier, L., van de Geer, S., and Bühlmann, P. The group lasso for logistic regression. *J. Royal Stat. Soc.: Ser. B*, 70 (1):53–71, 2008.

[22] Meinshausen, N. and Yu, B. Lasso-type recovery of sparse representations for high-dimensional data. *Ann. Stat.*, 37(1): 246–270, 2009.

[23] Nesterov, Y. Gradient methods for minimizaing composite objective function. *Center for Operations Research and Econometrics (CORE0, Catholic University of Louvain, Tech. Rep*, 76, 2007.

[24] Nocedal, J. and Wright, S. *Numerical Optimization*. Springer, 1999.

[25] Obonzinski, G., Taskar, B., and Jordan, M. Multi-task feature selection. Technical report, UC Berkeley, Jun. 2006.

[26] Roberts, A. W. and Varberg, D. E. Another proof that convex functions are locally Lipschitz. *The American Mathematical Monthly*, 81(9):1014–1016, 1974.

[27] Schmidt, M., van den Berg, E., Friedlander, M., and Murphy, K. Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm. In *AISTATS*, 2009.

[28] Shalev-Shwartz, S. and Tewari, A. Stochastic methods for ℓ_1-regularized loss minimization. In *ICML*, 2009.

[29] Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[30] Tseng, P. and Yun, S. A block-coordinate descent method for linearly constrained nonsmooth separable optimization. *J. Optim. Theory and Appl.*, 140, 2009.

[31] Turlach, B. A., Venables, W. N., and Wright, S. J. Simultaneous Variable Selection. *Technometrics*, 27:349–363, 2005.

[32] van den Berg, E., Schmidt, M., Friedlander, M. P., and Murphy, K. Group sparsity via linear-time projection. Technical Report TR-2008-09, Univ. British Columbia, Jun. 2008.

[33] Wang, X. G., Zhang, C., and Zhang, Z. Y. Boosted multi-task learning for face verification with applications to web image and video search. In *CVPR*, pp. 142–149, 2009.

[34] Wright, S. J., Nowak, R. D., and Figueiredo, M. A. T. Sparse reconstruction by separable approximation. *IEEE Trans. Sig. Proc.*, 57(7):2479–2493, 2009.

[35] Yuan, G.X., Chang, K.W., Hsieh, C.J., and Lin, C.J. A comparison of optimization methods for large-scale l1-regularized linear classification. Technical report, Department of Computer Science, National Taiwan University, http://www. csie. ntu. edu. tw/˜ cjlin/papers/l1. pdf, 2009.

[36] Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *J. R. Statist. Soc. B*, 68 (1):49–67, 2006.

[37] Zhao, P. and Yu, B. On Model Selection Consistency of Lasso. *JMLR*, 7:2541–2563, 2006.

[38] Zhao, P., Rocha, G., and Yu, B. The composite absolute penalties family for grouped and hierarchical variable selection. *Ann. Stat.*, 37(6A):3468–3497, 2009.

[39] Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*, 67(2):301–320, 2005.