Lifted coordinate descent for learning with trace-norm regularization

Miroslav Dudík
Yahoo! Research, NY

Zaid Harchaoui
INRIA and LJK, Grenoble

Jérôme Malick
CNRS and LJK, Grenoble

Abstract

We consider the minimization of a smooth loss with trace-norm regularization, which is a natural objective in multi-class and multi-task learning. Even though the problem is convex, existing approaches rely on optimizing a non-convex variational bound, which is not guaranteed to converge, or repeatedly perform singular-value decomposition, which prevents scaling beyond moderate matrix sizes. We lift the non-smooth convex problem into an infinitely dimensional smooth problem and apply coordinate descent to solve it. We prove that our approach converges to the optimum, and is competitive or outperforms state of the art.

1 Introduction

A large set of machine learning techniques including SVMs, logistic regression or boosting can be phrased as convex optimization among the set of linear predictors. The optimization objective has typically two parts: *empirical risk*, measuring a goodness of fit to the data, and *regularization*, measuring the complexity of the model and thus controlling its capacity to overfit. Here, we study variants where the model takes the form of a matrix, such as in multi-class or multi-task learning problems, and the regularization takes the form of the trace norm [36, 1, 2, 24, 30]. The trace norm (i.e., the sum of singular values) is the convex envelope of the rank of a matrix (on a special ball, see [15, 31]); thus it encourages matrices of low rank. Even though the resulting problems are convex, the existing techniques for trace-norm regularized optimization [36, 1, 24, 30] either do not scale beyond moderately sized matrices, or are not guaranteed to converge to the optimum. Here we propose a simple approach based on coordinate descent which is guaranteed to converge to the minimum of the convex objective and also scales to large matrix sizes.

Previous approaches for trace-norm regularized problems fall into two categories. The first set of approaches adapts general-purpose convex optimization to trace norm. Here, the most scalable ones are composite optimization techniques [24, 30]. Composite optimization relies on the implementation of the *proximal operator*. For trace-norm regularization, the key operation in the proximal operator is singular-value decomposition (SVD), which is a bottleneck in scaling to large matrix sizes. The second set of approaches uses variational characterizations of trace norm [36, 1] and proceeds by alternating minimization. These techniques either rely on SVD calculations [1], which prevents their scaling, or lack global convergence guarantees [36], and thus they are sensitive to the starting point and may require extensive problem-specific tuning.

Our approach is based on coordinate descent, which has demonstrated extremely good performance in ℓ_1-regularized optimization [17, 39]. Since the trace norm is a natural generalization of the ℓ_1-norm to matrices with the coordinates replaced by rank-one matrices (see, e.g., [31]), we should expect that coordinate descent algorithms will generalize to matrix settings as long as we replace the coordinates by the suitable rank-one matrices. The challenge is determining *which* rank-one matrices. Ideally, we would like to use the rank-one matrices from the SVD of the solution. However, except for least-squares regression problems, it is not clear how to determine these matrices without first finding the solution. This conceptual obstacle has prevented the application of coordinate-descent techniques to matrix setting [35, Chapter 11].

In this paper, we overcome this conceptual obstacle by considering *all possible* (normalized) rank-one matrices as coordinates. This set of matrices forms an overcomplete and uncountable infinite basis of the space of matrices. We show that a simple strategy of performing a coordinate descent on this lifted space actually

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX.

converges to the right solution. Picking the coordinate corresponding to the steepest descent direction amounts to calculating the top singular-vector pair. This operation is an order of magnitude faster than SVD. In our experiments we demonstrate that our approach is competitive and often outperforms existing approaches. Various tricks of the trade from ℓ_1-based optimization, such as regularization path calculation [17], can be adapted to our setting.

The idea of coordinate descent in infinite dimensional (or functional) spaces is explored in the boosting literature [25, 11, 40]. In linear programming, this technique is known as column generation [12]. However, these techniques have not been previously applied and analyzed for trace-norm objectives. Several authors [19, 22, 33] have used rank-one updates in convex optimization settings, but their techniques were tailored to quadratic objectives and the focus was on the case of low-rank matrix approximation (e.g., in collaborative filtering) rather than the general purpose trace-norm optimization, which is the subject of this paper. Finally, there are some similarities between our algorithm and algorithms of [34] and [38] for ℓ_1 regularized least-squares regression. Both algorithms rely on coordinate descent techniques to identify a subspace of interesting "active" variables.

In Sec. 2, we describe our problem setting in more detail. In Sec. 3, we lift the non-smooth matrix objective to a smooth objective in infinite dimensions, describe our algorithm and prove its convergence. Finally, in Sec. 4 we evaluate our method on several synthetic and real-world data sets.

2 Supervised learning with trace-norm regularization

We begin by reviewing two supervised learning problems and the associated optimization objectives where the parameters naturally form a matrix. Then we introduce trace-norm regularized optimization.

2.1 Multi-class classification

Our first problem is multi-class classification by multinomial logistic regression. For example, in image classification classes correspond to different object categories, such as human faces, cars, or animals, and each training example is described by a vector of features derived from the pixel representation of the image.

Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ be a set of labeled training data, where $\mathbf{x}_i \in \mathbb{R}^d$ are feature vectors and $y_i \in \mathcal{Y} := \{1, \ldots, k\}$ are the class labels. The linear classifier is specified by a separate weight vector $\mathbf{w}_y \in \mathbb{R}$ for each class. For a given test example $\mathbf{x} \in \mathbb{R}^d$, the target class is predicted according to $\hat{y} = \operatorname{Arg\,max}_y \mathbf{w}_y^\top \mathbf{x}$. Class-wise weight vectors form the weight matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k] \in \mathbb{R}^{d \times k}$. In regularized multinomial logistic regression, the classifier is obtained by solving the optimization problem:

$$\underset{\mathbf{W} \in \mathbb{R}^{d \times k}}{\text{Minimize}} \quad \lambda \Omega(\mathbf{W}) + \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{W}; \mathbf{x}_i, y_i) \qquad (1)$$

where $\Omega(\mathbf{W})$ is the regularization and

$$L(\mathbf{W}; \mathbf{x}, y) = \log\Bigl(1 + \sum_{\ell \in \mathcal{Y} \setminus \{y\}} \exp\left\{\mathbf{w}_\ell^\top \mathbf{x} - \mathbf{w}_y^\top \mathbf{x}\right\}\Bigr)$$

is the multinomial logistic loss.

2.2 Multi-task classification

Our second example is multi-task learning, where the goal is to solve multiple classification problems simultaneously. Each individual problem is modeled by multinomial logistic regression. For example, in Sec. 4, we consider the problem of predicting user preferences for different products within some category (such as personal computers). Here, users correspond to tasks. For each user we collect relative preferences over various pairs of products within our category. Each product is described by a vector of features. Examples represent pairs of products, with the feature vector being the difference of the feature vectors of the two products, and the relative preference (for the first or the second product) being the target class. Thus, each individual task is a binary logistic regression problem.

Formally, we are given m data sets (tasks) indexed by $j = 1, \ldots, m$, each consisting of n_j examples $(\mathbf{x}_{j,1}, y_{j,1}), \ldots, (\mathbf{x}_{j,n_j}, y_{j,n_j})$ where $\mathbf{x}_{j,i} \in \mathbb{R}^d$ is the feature vector and $y_{j,i} \in \mathcal{Y}_j := \{1, \ldots, k_j\}$ is a class label. For each task, we fit a matrix of linear predictors $\mathbf{W}_j = [\mathbf{w}_{j,1}, \ldots, \mathbf{w}_{j,k_j}]$. We combine matrices across all tasks to obtain the joint matrix $\mathbf{W} = [\mathbf{W}_1, \ldots, \mathbf{W}_m] \in \mathbb{R}^{d \times k}$, where $k = k_1 + \cdots + k_m$, which is fitted by regularized multi-task logistic regression as follows:

$$\underset{\mathbf{W} \in \mathbb{R}^{d \times k}}{\text{Minimize}} \quad \lambda \Omega(\mathbf{W}) + \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} L_j(\mathbf{W}; \mathbf{x}_{j,i}, y_{j,i}) \quad (2)$$

where $\Omega(\mathbf{W})$ is the regularization, $n = n_1 + \cdots + n_m$ is the total number of examples, and L_j is the multinomial logistic loss over the submatrix $\mathbf{W}_j$.

2.3 Matrix norms

Before we introduce trace-norm regularization and regularized optimization, some notation is in order. In a

vector space $\mathbb{R}^p$, we use notation $\|\cdot\|_2$, $\|\cdot\|_1$, $\|\cdot\|_\infty$, respectively, for ℓ_2-norm (the Euclidean norm), ℓ_1-norm (the sum of absolute values), and ℓ_∞-norm (the maximum of absolute values).

For a matrix $\mathbf{W} \in \mathbb{R}^{d\times k}$, we write $\sigma(\mathbf{W})$ for the spectrum of the matrix, viewed as a vector of its singular values, and define the so-called Schatten p-norm as

$$\|\mathbf{W}\|_{\sigma,p} \coloneqq \|\sigma(\mathbf{W})\|_p \ .$$

In this paper we only use $p = 1$ and $p = \infty$. For $p = \infty$, we obtain the maximum-singular value norm. For $p = 1$, we obtain the so-called trace norm of the matrix (also called nuclear norm). Note that for a positive-definite matrix $\mathbf{W}$, $\|\mathbf{W}\|_{\sigma,1}$ equals the trace of $\mathbf{W}$, hence the name trace norm.

2.4 Learning with trace-norm penalty

There are many choices of regularization functions in the two problems introduced in the previous sections. We focus on regularization by trace norm, i.e., $\Omega(\mathbf{W}) = \|\mathbf{W}\|_{\sigma,1}$. As we mention in the introduction, trace norm encourages low rank solutions. Hence, it corresponds to the assumption that the linear predictors lie in the same linear subspace of $\mathbb{R}^d$ [1].

The topic of this paper is the optimization problem

$$\underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\text{Minimize}} \quad \phi_\lambda(\mathbf{W}) := \lambda\|\mathbf{W}\|_{\sigma,1} + \phi(\mathbf{W}) \qquad (3)$$

where $\phi : \mathbb{R}^{d\times k} \to \mathbb{R}$ is the empirical risk (i.e., average loss across training examples). We assume it satisfies the following conditions:

(A) **convexity**: ϕ is convex

(B) **lower-boundedness**: ϕ is bounded below; we assume $\phi(\mathbf{W}) \geq 0$ (otherwise ϕ can be shifted)

(C) **smoothness**: ϕ is differentiable and there exists a norm $\|\cdot\|$, and a constant $H > 0$ such that

$$\big\langle \mathbf{W}' - \mathbf{W},\, \nabla\phi(\mathbf{W}') - \nabla\phi(\mathbf{W})\big\rangle \leq H\|\mathbf{W}' - \mathbf{W}\|^2$$

for all $\mathbf{W}'$, $\mathbf{W} \in \mathbb{R}^{d\times k}$.

For example, the empirical risks in Eqs. (1) and (2) satisfy these conditions. This result, based on properties of the multinomial logistic loss function L, is proved in Appendix B.

2.5 Matrix optimization

The learning problem (3) is a convex non-smooth matrix optimization problem. Let us briefly present some of its basic properties.

First, it is easy to see that a solution to the problem (3) always exists. This is because the minimization can be restricted to the level-set $\{\mathbf{W} :\ \phi_\lambda(\mathbf{W}) \leq \phi_\lambda(\mathbf{0})\}$, which is compact (since ϕ is bounded below). By continuity, ϕ_λ attains a minimum over this set.

The necessary and sufficient condition for optimality of $\mathbf{W}$ is that $\mathbf{0}$ lies in the subdifferential of ϕ_λ

$$\mathbf{0} \in \partial\phi_\lambda(\mathbf{W}) \ .$$

By subdifferential calculus [20], this is equivalent to

$$-\nabla\phi(\mathbf{W})/\lambda \in \partial\|\mathbf{W}\|_{\sigma,1} \ .$$

Now, since $\partial\|\mathbf{W}\|_{\sigma,1}$ is exactly (see, e.g., [15])

$$\{\mathbf{M} \in \mathbb{R}^{d\times k} :\ \|\mathbf{M}\|_{\sigma,\infty} \leq 1,\, \langle\mathbf{M}, \mathbf{W}\rangle = \|\mathbf{W}\|_{\sigma,1}\},$$

we obtain the following proposition:

Proposition 2.1. *A matrix $\mathbf{W} \in \mathbb{R}^{d\times k}$ solves Eq. (3) if and only if*

(i) $\|\nabla\phi(\mathbf{W})\|_{\sigma,\infty} \leq \lambda$, *and*

(ii) $\langle\nabla\phi(\mathbf{W}), \mathbf{W}\rangle = -\lambda\|\mathbf{W}\|_{\sigma,1}$.

We say that $\mathbf{W}$ is an ε-approximate solution, or simply an ε-solution, if the optimality conditions are approximately satisfied:

(i') $\|\nabla\phi(\mathbf{W})\|_{\sigma,\infty} \leq \lambda + \varepsilon$, and

(ii') $\Big|\langle\nabla\phi(\mathbf{W}), \mathbf{W}\rangle + \lambda\|\mathbf{W}\|_{\sigma,1}\Big| \leq \varepsilon\|\mathbf{W}\|_{\sigma,1}$.

3 Trace-norm optimization via lifted coordinate descent

This section presents our approach for solving the learning problem (3). The idea is to recast this non-smooth optimization problem in $\mathbb{R}^{d\times k}$ as a smooth optimization problem in an infinite dimensional space. We then exploit the simplicity of the new formulation to design a coordinate descent algorithm.

3.1 Lifting to an infinite dimensional space

We do not have a basis on which we could design a coordinate descent algorithm. So we construct, as follows, an *overcomplete and uncountable infinite "basis"* for the set of matrices living in $\mathbb{R}^{d\times k}$, by considering *all possible* (normalized) rank-one matrices.

Let $\mathcal{M}$ denote the set of rank-one matrices

$$\mathcal{M} = \{\mathbf{u}\mathbf{v}^\top :\ \mathbf{u} \in \mathbb{R}^d,\, \mathbf{v} \in \mathbb{R}^k,\, \|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1\} \ .$$

Let $\mathcal{I}$ be an index set for the elements of $\mathcal{M}$, i.e.,

$$\mathcal{M} = \{\mathbf{M}_i \in \mathbb{R}^{d\times k} :\ i \in \mathcal{I}\} = \{\mathbf{u}_i\mathbf{v}_i^\top :\ i \in \mathcal{I}\} \ .$$

A function from $\mathcal{I}$ to $\mathbb{R}$ can be written $\boldsymbol{\theta} = (\theta_i)_{i\in\mathcal{I}} \in \mathbb{R}^{\mathcal{I}}$; its support is the set of indices which are nonzero,

i.e., $\mathrm{supp}(\boldsymbol{\theta}) = \{i \in \mathcal{I} : \theta_i \neq 0\}$. We consider the vector space of functions $\boldsymbol{\theta}$ with finite support, denoted as Θ,

$$\Theta \coloneqq \{\boldsymbol{\theta} \in \mathbb{R}^{\mathcal{I}} : \mathrm{supp}(\boldsymbol{\theta}) \text{ is finite}\}$$

equipped with the natural ℓ_1-norm defined by $\|\boldsymbol{\theta}\|_1 = \sum_{i\in\mathcal{I}} |\theta_i|$. Basic properties of the space and its elements are recalled in Appendix E. The connection between Θ and $\mathbb{R}^{d\times k}$ is the following: each $\boldsymbol{\theta} \in \Theta$ defines a unique matrix in $\mathbb{R}^{d\times k}$

$$\mathbf{W}_{\boldsymbol{\theta}} = \sum_{i\in\mathcal{I}} \theta_i \mathbf{M}_i \ . \tag{4}$$

The properties of the map $\boldsymbol{\theta} \mapsto \mathbf{W}_{\boldsymbol{\theta}}$ are simple, but important in our developments; we summarize them in the next proposition. It concerns the *non-negative orthant* of Θ:

$$\Theta^+ = \{\boldsymbol{\theta} \in \Theta : \theta_i \geq 0 \text{ for all } i \in \mathcal{I}\} \ .$$

Proposition 3.1. *The map $\boldsymbol{\theta} \mapsto \mathbf{W}_{\boldsymbol{\theta}}$ is a continuous linear map from Θ to $\mathbb{R}^{d\times k}$. Moreover, for all $\boldsymbol{\theta} \in \Theta^+$, we have*

$$\|\mathbf{W}_{\boldsymbol{\theta}}\|_{\sigma,1} \leq \sum_{i\in\mathcal{I}} \theta_i = \|\boldsymbol{\theta}\|_1$$

and for any $\mathbf{W} \in \mathbb{R}^{d\times k}$, the vector of its singular values corresponds to $\boldsymbol{\theta} \in \Theta^+$ such that $|\mathrm{supp}(\boldsymbol{\theta})| = \mathrm{rank}(\mathbf{W})$, $\mathbf{W}_{\boldsymbol{\theta}} = \mathbf{W}$ and $\|\boldsymbol{\theta}\|_1 = \|\mathbf{W}\|_{\sigma,1}$.

Thus the trace norm in $\mathbb{R}^{d\times k}$ and the ℓ_1-norm in Θ^+ almost coincide [23]. It is tempting to replace the optimization in $\mathbf{W}$ by optimization in $\boldsymbol{\theta}$ over Θ^+. Consider $\psi(\boldsymbol{\theta}) \coloneqq \phi(\mathbf{W}_{\boldsymbol{\theta}})$, the infinite dimensional version of ϕ, and the optimization problem

$$\underset{\boldsymbol{\theta}\in\Theta^+}{\text{Minimize}} \quad \psi_\lambda(\boldsymbol{\theta}) \coloneqq \lambda \sum_{i\in\mathcal{I}} \theta_i + \phi(\mathbf{W}_{\boldsymbol{\theta}}) \ . \tag{5}$$

By Prop. 3.1, for all $\boldsymbol{\theta} \in \Theta^+$, we have an upper bound

$$\phi_\lambda(\mathbf{W}_{\boldsymbol{\theta}}) \leq \psi_\lambda(\boldsymbol{\theta}).$$

The next theorem shows that minimizing ϕ_λ and ψ_λ is actually equivalent.

Theorem 3.2. *The function $\psi_\lambda \colon \Theta \to \mathbb{R}$ is convex and differentiable. The following optimization problems are equivalent, i.e., they have the same optimal value and correspondence of optimal solutions as*

$$\hat{\boldsymbol{\theta}} \in \underset{\boldsymbol{\theta}\in\Theta^+}{\mathrm{Arg\,min}}\, \psi_\lambda(\boldsymbol{\theta}) \quad \textit{iff} \quad \mathbf{W}_{\hat{\boldsymbol{\theta}}} \in \underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\mathrm{Arg\,min}}\, \phi_\lambda(\mathbf{W}) \ .$$

Note that the first-order optimality conditions for problem (5) are

(a) $\forall i \in \mathcal{I} : \ \frac{\partial\psi}{\partial\theta_i}(\boldsymbol{\theta}) \geq -\lambda$

(b) $\forall i \in \mathrm{supp}(\boldsymbol{\theta}) : \ \frac{\partial\psi}{\partial\theta_i}(\boldsymbol{\theta}) = -\lambda.$

The ε-approximate optimality is defined by

(a') $\forall i \in \mathcal{I} : \ \frac{\partial\psi}{\partial\theta_i}(\boldsymbol{\theta}) \geq -\lambda - \varepsilon$

(b') $\forall i \in \mathrm{supp}(\boldsymbol{\theta}) : \ \left|\frac{\partial\psi}{\partial\theta_i}(\boldsymbol{\theta}) + \lambda\right| \leq \varepsilon$

The correspondence of Theorem 3.2 between the optimal solutions of the two problems is in fact even stronger, as it extends to approximate solutions.

Theorem 3.3. *Let ε be such that $0 \leq \varepsilon \leq \lambda$. If $\boldsymbol{\theta}$ is an ε-solution of (5), then $\mathbf{W}_{\boldsymbol{\theta}}$ is an ε-solution of (3).*

We now use this infinite dimensional embedding to design our learning algorithm.

3.2 Coordinate descent algorithm

This section presents our coordinate descent algorithm for optimizing the function ψ_λ over Θ^+, and thus solving the original learning problem (3).

At the current iterate $\boldsymbol{\theta}$, we pick the coordinate along which we can achieve the steepest descent while remaining in Θ^+. For coordinates $i \notin \mathrm{supp}(\boldsymbol{\theta})$, we can only move in the positive direction. So it suffices to pick $i \in \mathcal{I}$ with the smallest $\frac{\partial\psi_\lambda}{\partial\theta_i}(\boldsymbol{\theta})$. This problem is equivalent to calculating the top singular-vector pair of the matrix $-\nabla\phi(\mathbf{W})$ (where $\mathbf{W} = \mathbf{W}_{\boldsymbol{\theta}}$), since

$$\begin{aligned}\mathrm{Arg\,min}_{i\in\mathcal{I}}\, \tfrac{\partial\psi_\lambda}{\partial\theta_i}(\boldsymbol{\theta}) &= \mathrm{Arg\,min}_{i\in\mathcal{I}} \big(\lambda + \langle \mathbf{M}_i, \nabla\phi(\mathbf{W})\rangle\big) \\ &= \mathrm{Arg\,min}_{i\in\mathcal{I}}\, \langle \mathbf{u}_i\mathbf{v}_i^\top, \nabla\phi(\mathbf{W})\rangle \\ &= \mathrm{Arg\,max}_{i\in\mathcal{I}}\ \mathbf{u}_i^\top\big(-\nabla\phi(\mathbf{W})\big)\mathbf{v}_i \ .\end{aligned}$$

For coordinates $i \in \mathrm{supp}(\boldsymbol{\theta})$, it is also possible to move in the negative direction. Here, we perform a traditional ℓ_1-style coordinate descent. We can update coordinates either cyclically [17] (which is what we do here), or uniformly at random [27]. We optimize over $\mathrm{supp}(\boldsymbol{\theta})$ until the optimality conditions are satisfied.

The final algorithm (Algorithm 1) is called **R1D**, which stands for *rank-one descent*. In our algorithm, we do not compute the steepest direction $\frac{\partial\psi_\lambda}{\partial\theta_i}(\boldsymbol{\theta})$, but only use a steep-enough direction (steepest up to $\varepsilon/2$). The following proposition shows that **R1D** is guaranteed to make fixed progress provided that the corresponding partial derivative is large enough.

Proposition 3.4. *There exist $\alpha, \delta > 0$ such that for all $\varepsilon > 0$, $\boldsymbol{\theta} \in \Theta^+$ and $i \in \mathcal{I}$ such that $\frac{\partial\psi_\lambda}{\partial\theta_i}(\boldsymbol{\theta}) \leq -\varepsilon$, we have*

$$\psi_\lambda(\boldsymbol{\theta} + \delta\,\mathbf{e}_i) \leq \psi_\lambda(\boldsymbol{\theta}) - \alpha\varepsilon^2. \tag{6}$$

We deduce that the algorithm converges to an ε-optimal solution in a finite number of iterations.

Theorem 3.5. R1D *provides ε-optimal solutions $\boldsymbol{\theta}_\varepsilon$ and $\mathbf{W}_\varepsilon$ after at most $8\psi_\lambda(\boldsymbol{\theta}_0)/\alpha\varepsilon^2$ iterations.*

Algorithm 1 R1D$(\phi, \lambda, \boldsymbol{\theta}_0, \varepsilon)$

Input: empirical risk ϕ, regularization λ
initial point $\mathbf{W}_{\boldsymbol{\theta}_0}$, convergence threshold ε

Output: ε-optimal $\mathbf{W}_{\boldsymbol{\theta}}$

Notation: $\mathbf{W}_t \coloneqq \mathbf{W}_{\theta_t}$, $\mathbf{u}_t \coloneqq \mathbf{u}_{i_t}$, $\mathbf{v}_t \coloneqq \mathbf{v}_{i_t}$, $\mathbf{e}_t \coloneqq \mathbf{e}_{i_t}$

Algorithm:

For $t = 0, 1, 2, \ldots$:

1. Find an approximate top singular-vector pair of $(-\nabla\phi(\mathbf{W}_t))$, i.e., $i_t \in \mathcal{I}$ such that
$$\mathbf{u}_t^\top \big(-\nabla\phi(\mathbf{W}_t)\big)\mathbf{v}_t \geq \|\nabla\phi(\mathbf{W}_t)\|_{\sigma,\infty} - \varepsilon/2$$
2. Let $g_t \coloneqq \frac{\partial\psi_\lambda}{\partial\theta_{i_t}}(\boldsymbol{\theta}_t) = \lambda + \langle \nabla\phi(\mathbf{W}_t), \mathbf{u}_t\mathbf{v}_t^\top\rangle$
3. If $g_t \leq -\varepsilon/2$
$$\mathbf{W}_{t+1} = \mathbf{W}_t + \delta\,\mathbf{u}_t\mathbf{v}_t^\top \text{ with } \delta \text{ given by Prop. 3.4}$$
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \delta\,\mathbf{e}_t$$
4. Else (i.e., $g_t > -\varepsilon/2$)
If $\boldsymbol{\theta}_t$ satisfies (b'), terminate and return $\boldsymbol{\theta}_t$
Otherwise, compute $\boldsymbol{\theta}_{t+1}$ as an ε-solution of the restricted problem $\min_{\boldsymbol{\theta}\in\mathbb{R}_+^{\mathrm{supp}(\boldsymbol{\theta}_t)}} \psi_\lambda(\boldsymbol{\theta})$

If the upper bound H is not known ahead of time, it is possible to use a search strategy as in [26]. For special loss functions, it is possible to derive a tighter upper bound than implied by Prop. 3.4. For instance, for multi-class loss, we use an upper bound along the lines of [14]. When a specialized bound is not available or too loose (as we found in the case of multi-task loss), we observed that it helps to augment the rule of Prop. 3.4 by line search. Details of these strategies will be provided in the extended version of this paper.

If instead of terminating **R1D** after reaching ε-optimality, we decrease ε according to a predefined sequence $(\varepsilon_\ell)_\ell$ converging to 0, we obtain an asymptotic convergence.

Theorem 3.6. *Let $\varepsilon_\ell \to 0$. Define $\mathbf{W}_{\boldsymbol{\theta}_\ell}$ as the solution generated by* **R1D** *with $\varepsilon = \varepsilon_\ell$. Then a subsequence of $(\mathbf{W}_{\boldsymbol{\theta}_\ell})_\ell$ converges to a solution of* (3).

Running time We discuss the running time of our algorithm focusing on the specific losses associated with multi-class and multi-task learning. Recall that the number of training examples is n, the number of linear predictors (matrix columns) is k and the number of features is d. The final parameter is the size of the current $\mathrm{supp}(\boldsymbol{\theta}_t)$ which we denote r. The key operations are calculations of $\nabla\phi(\mathbf{W})$, approximate top singular-vector pair, and $\nabla_{\mathrm{supp}(\boldsymbol{\theta})}\psi_\lambda(\boldsymbol{\theta})$ in Step 4. Their running times are as follows:

- $\nabla\phi(\mathbf{W})$: bottleneck of this operation is the matrix-vector product $\mathbf{W}^\top\mathbf{x}_i$ (in multi-class problem) or $\mathbf{W}_j^\top\mathbf{x}_i$ (in multi-task problem), respectively, for $i = 1, \ldots, n$ and $i = 1, \ldots, n_j$. Naively, the running time is $O(ndk)$ and $O(\sum_j n_j dk_j)$, but exploiting special properties (e.g., sparsity of feature vectors), this step can be much faster. The representation $\mathbf{W} = \mathbf{W}_{\boldsymbol{\theta}}$ as a sum of rank-one matrices presents an additional opportunity which (even without special structure) immediately yields the running times $O(n(d+k)r)$ and $O(\sum_j n_j(d+k_j)r)$. If n is very large, the summation across examples can be parallelized.
- **approximate top singular-vector pair**: this can be calculated in time $O(dk)$ by a few steps of the power method or Lanczos iterations [9].
- $\nabla_{\mathrm{supp}(\boldsymbol{\theta})}\psi_\lambda(\boldsymbol{\theta})$: without additional structure this can be done in time $O(nrk)$ and $O(\sum_j n_j rk_j)$, assuming that $\mathbf{u}_\ell^\top\mathbf{x}_i$ values are precomputed for all $\ell \in \mathrm{supp}(\boldsymbol{\theta})$ and all $i = 1, \ldots, n$ (this is amortized into the calculation of $\nabla\phi(\mathbf{W})$). Again, summation over n can be parallelized if needed.

Algorithm 2 ContR1D$(\phi, \lambda_0, \alpha, N)$

Input: empirical risk ϕ, initial regularization λ_0
multiplicative step $\alpha \in (0, 1)$
number of steps $N \geq 1$

Output: $(\mathbf{W}_{\boldsymbol{\theta}_\ell})_{\ell=0}^N$ minimizing ϕ_{λ_ℓ} with $\lambda_\ell = \lambda_0\alpha^\ell$

Algorithm:

Let $\beta = \dfrac{1-\alpha}{1+\alpha}$, $\lambda_\ell = \lambda_0\alpha^\ell$, $\varepsilon_\ell = \beta\lambda_\ell$

$\boldsymbol{\theta}_0 = \mathbf{R1D}(\phi, \lambda_0, \mathbf{0}, \varepsilon_0)$

For $\ell = 1, 2, \ldots, N$:

$\boldsymbol{\theta}_\ell = \mathbf{R1D}(\phi, \lambda_\ell, \boldsymbol{\theta}_{\ell-1}, \varepsilon_\ell)$

Continuation It has been noted that solving ℓ_1-regularized problems is faster when λ is large [17]. In fact, these instances give coordinate descent an edge over other techniques since the solutions for larger λ tend to be sparser. This can be extended to trace-norm problems, with the sparsity replaced by low rank. We can accelerate **R1D** by taking a sequence of problems with decreasing values of the λ, and using the intermediate solution as a warm start for the next problem. In addition to the benefit from the warm-starting, we obtain a sequence of models optimizing the same empirical risk with different values of regularization. Such a sequence is called a *regularization path* [18] and is in itself a useful output since in practice we typically choose among several values of λ by cross-validation.

ContR1D (Algorithm 2) is the continuation version of our **R1D**. It returns a regularization path for

a geometrically spaced sequence of λ's of the form $\lambda_\ell = \lambda_0 \alpha^\ell$ where $\alpha \in (0,1)$. As a convergence criterion we use $\varepsilon_\ell = \beta\lambda_\ell$. We select the largest $\beta \in (0,1)$ that guarantees that sets of ε_ℓ-approximate minimizers of ϕ_{λ_ℓ} *do not intersect* (with the exception of the case $\nabla\phi(\mathbf{0}) = \mathbf{0}$, for which the neighborhoods of $\mathbf{0}$ are approximate minimizers for any $\varepsilon_\ell > 0$). By investigating ε-optimality conditions for consecutive λ's, we obtain the setting $\beta = (1-\alpha)/(1+\alpha)$.

Say that our goal is to calculate an $\bar{\varepsilon}$-solution for a specific regularization coefficient $\bar{\lambda}$. Then we can use the algorithm **ContR1D** as follows. We set $\lambda_0 = \|\nabla\phi(\mathbf{0})\|_{\sigma,\infty}$ to guarantee that $\mathbf{0}$ is a minimizer of ϕ_{λ_0}. We set $\bar{\beta} = \bar{\varepsilon}/\bar{\lambda}$ and invert the formulas in **ContR1D** to obtain α and N such that $\lambda_N = \bar{\lambda}$ and $\varepsilon_N \leq \bar{\varepsilon}$. We run **ContR1D** with the calculated α and N, and obtain an $\bar{\varepsilon}$-solution for $\bar{\lambda}$ as our last iterate $\mathbf{W}_{\boldsymbol{\theta}_N}$.

3.3 Previous algorithms

Existing approaches for trace-norm penalized learning fall into three categories: (1) proximal gradient methods [24, 30]; (2) alternating direction methods, based on variational characterizations of trace norm, such as the iterative rescaling [1], or low-norm factorization [36]; and (3) conditional gradient methods [22]. These methods are described in Appendix C. Here we stress the main differences between them and **R1D**.

Proximal gradient An iteration of a basic proximal gradient method [5, 3] consists of a gradient step on ϕ followed by a "correction" (proximal step) according to the trace norm. An accelerated version has good rate of convergence and has been shown to be effective for trace-norm problems [30]. Roughly speaking, $O(1/\sqrt{\varepsilon})$ iterations are required to achieve ε-accuracy, while **R1D** converges in $1/\varepsilon^2$ iterations. However, computing the proximal step for the trace norm requires solving an SVD, i.e., the running time $O(kd\,\mathrm{rank}(\mathbf{W}))$, while **R1D** needs only an approximate top singular-vector pair with the running time $O(kd)$. As shown in Sec. 4, faster iterations are the key to scaling up to larger problems. Additionally, since our algorithm is incrementally adding rank-one matrices, it automatically maintains a matrix factorization of the approximate solution and has an explicit control over an upper bound on the rank.

Iterative rescaling The approach of [1] is based on reformulating the trace norm as an infimum of reweighted Frobenius norms, which bypasses the non-smoothness of the problem. While the resulting smooth convex optimization problems are easier to solve (by stabilized gradient-like methods for example), we lose (part of) the benefit of the trace-norm regularization, which is that the trace-norm penalty is enforcing low rank. The numerical experiments will show that this method does not approximate well optimal solutions when they are of low rank.

Low-norm factorization The approach of [36] is to use the decomposition $\mathbf{W} = \mathbf{U}\mathbf{V}^\top$ to reformulate the trace norm as the minimum of the sum of squared Frobenius norms. A block-coordinate descent can then be applied since the minimizations with respect to each factor $\mathbf{U}$ and $\mathbf{V}$ are smooth optimization problems, tackled by stabilized gradient-like methods. This approach however breaks the convexity of the original problem (3), as the reformulated problem is not jointly convex with respect to $(\mathbf{U}, \mathbf{V})$. As we will see in our experiments, the algorithm is very sensitive to starting points, and gets stuck in non-optimal critical points.

Conditional gradient Recently, conditional gradient algorithms were proposed for squared-loss problems with a trace-norm *constraint* [22], frequently used in collaborative filtering. We carried out preliminary experiments with the algorithm of [22], but we observed slow convergence. We attribute it to the fact that the algorithm was designed and evaluated on squared loss. While the recommended theoretical stepsize works fine for squared loss, we believe it might be too conservative for multinomial logistic losses. Furthermore, it is difficult to conduct a fair experimental comparison since [22] works on the constrained formulation while our algorithm solves the penalized formulation. We defer a detailed comparison to future work.

4 Experimental results

We conducted experiments covering a wide range of problem scales, feature correlation amounts, and regularization amounts. We use the following acronyms for the four compared methods: **ContR1D** for our algorithm, **Prox++** for accelerated proximal gradient, **IR** for iterative rescaling, and **AM** for alternating minimization for the low-norm factorization objective.

4.1 Synthetic data

We first evaluate the methods on a synthetic multinomial logistic regression problem, following a similar protocol as in [35, Chapter 11].

We use $d = 250$ features and $k = 500$ classes. All our training data sets have 10 examples per class, yielding 5000 examples in total. Examples for each class are sampled from a separate multivariate normal distribution in $\mathbb{R}^d$ with means determined by choosing the first $0.2d$ coordinates independently uniformly at random from $\{-1, 1\}$, and setting the remaining $0.8d$

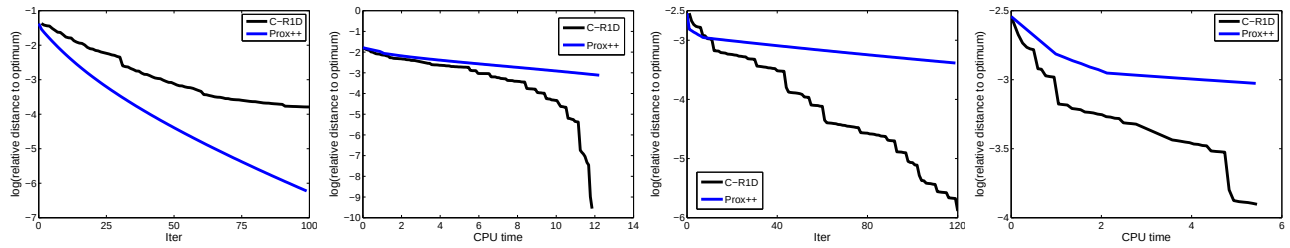


Figure 1: Optimization accuracy of **ContR1D** and **Prox++** in high correlation settings. Left two plots: accuracy versus the number of iterations and CPU time for light regularization. Right two plots: heavy regularization.

coordinates to zero. The covariance matrix for each class is $(\Sigma)_{i,j} = \sigma^2 \rho^{|i-j|}$, where we use σ to control the separation of the classes and ρ to control correlation among features. We consider $\rho = 0.1$ and $\rho = 0.9$ to obtain *low correlation* and *high correlation* settings. We set σ so that the average distance among all pairs of class means is 3σ. We compared performance for $\lambda = 0.5$ (heavy regularization) and $\lambda = 0.001$ (light regularization). We report performance results averaged over 10 replications by plotting the relative accuracy $|(f - f^\star)/f^\star|$ against the computation time. We used the smallest value of the objective function attained by the best performing method after a large number of iterations as a proxy for $f^\star$. As we observed similar or worse behavior for the other algorithms compared to **Prox++**, we only compare here our algorithm to **Prox++**.

In Fig. 1, we highlight the main strengths and weaknesses of our approach in high-correlation situations. We see that **ContR1D** outperforms **Prox++** in terms of CPU time in both high and low regularization settings. As a function of iteration, **Prox++** converges faster for low regularization since its convergence rate is directly controlled by the Lipschitz constant of the gradient (fast convergence for gradient with low Lipschitz constant). However, since its iterations are more costly, **ContR1D** achieves more accurate solutions faster. Note also the "staircase phenomenon" in the curves for **ContR1D**, which can be attributed to alternation between Steps 3 and 4. Our approach achieved similar performance to **Prox++** in low correlation situations, so the results are omitted.

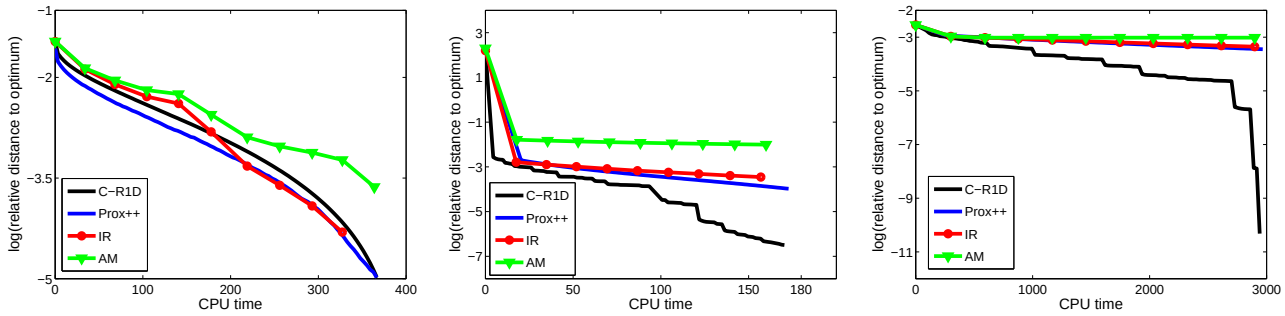
4.2 Real-world data

We considered two conjoint analysis datasets [1] and [8], which we refer to as *Conjoint (I)* and *Conjoint (II)*, and a subset of the *ImageNet* dataset 2010 [4]. In Fig. 2 (top), we compare our algorithm to the others in terms of optimization accuracy. In Fig. 2 (bottom), we plot the average test error as a function of training time (averaged over 10 cross-validation splits, using the best performing regularization coefficient λ).

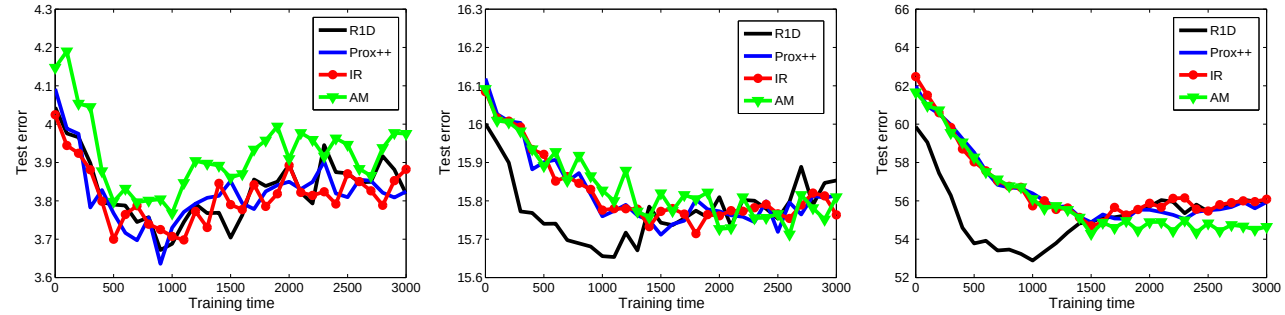
Conjoint analysis The goal of conjoint analysis is modeling people's preferences among choices in some set (e.g., products in a certain category). We view it as a multi-task problem with individuals corresponding to tasks and *pairs of choices* corresponding to examples. Each individual choice is modeled by a feature vector, the pair is modeled as the difference of the two vectors, with the target class being the preferred choice between the two listed in the pair, i.e., each task is a binary logistic regression. Dataset *Conjoint (I)* was taken from a survey of 180 individuals, each providing on average preferences for 8 pairs of PCs (among 20 different PCs), parameterized by 13 features. Dataset *Conjoint (II)* is another survey regarding 1187 individuals, each providing on average preferences for 10 pairs of options (among the total of 5), parameterized by 22 features.

On *Conjoint (I)*, all algorithms show similar optimization performance. Since the alternating minimization algorithm **AM** is heavily dependent on the initialization because of the non-convexity of the objective, we only reported the best performance for this algorithm. In terms of test error, three algorithms **ContR1D**, **Prox++**, and **IR** achieve similar performance, whereas the alternating minimization algorithm **AM** shows higher variance and worse accuracy. The algorithm gets easily trapped in local minima of the objective function which might not correspond to solutions yielding low test error.

On *Conjoint (II)*, the algorithms show different optimization performance. In particular, our algorithm clearly outperforms the other methods. The accelerated proximal gradient **Prox++** and the iterative rescaling **IR** show similar optimization performance. In terms of test error, our algorithm **ContR1D** performs better than the other three and in fact manages to reach the lowest test error in the early stages of training. This phenomenon might be due to the fact that early iterates of our algorithm tend to have lower rank, which might yield better generalization than the



(a) Conjoint (I) (b) Conjoint (II) (c) ImageNet



(a) Conjoint (I) (b) Conjoint (II) (c) ImageNet

Figure 2: Comparison of optimization performance and test error (in percentage) on real-world data.

final solution of the trace-norm regularized optimization. This behavior is reminiscent of the behavior of coordinate descent algorithms for ℓ_1-regularized problems (see, e.g., [39]), which tend to achieve better test error in early iterations as well.

ImageNet Here, we tackle a multi-class classification problem with the multinomial logistic loss. We chose a subset of 281 classes from the *ImageNet* dataset, corresponding to various kinds of birds, carnivores, and flowers. Each example is described by 4096 features. We use 10 training and 10 test examples per class.

Our algorithm shows superior optimization performance. This could be explained by the strong correlation of the visual features in this application. Such correlation harms the performance of the other algorithms whereas our algorithm remains robust.

As in the *Conjoint (II)* dataset, our coordinate descent algorithm reaches the lowest test error quickly, but the generalization performance then slightly deteriorates. It is worthwhile to note that, on this dataset, the alternating minimization algorithm **AM** reaches lower test error than the other algorithms after a long training time for a particular run. We interpret this by relating it to the low rank of the weight matrices at the end of the training phase. Since the optimization process of **AM** is not guaranteed to converge to the global optimum and is strongly biased by the structure of the initial weight matrix, **AM** displays good test-error performance when this structure is particularly tailored to the data at hand. Here, we observe that for the best run of **AM** the initial matrix was of very low rank.

5 Conclusion

We have introduced a new fast coordinate descent algorithm for a wide range of trace-norm regularized learning problems. We have shown that in problems with large matrices, our approach is competitive or outperforms existing optimization algorithms. Our work paves the way for the design of efficient and scalable learning approaches for large-scale matrix problems such as the full *ImageNet* dataset.

Acknowledgements

This work was funded by a Math-STIC project from Grenoble University and the PASCAL 2 Network of Excellence.

References

[1] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[2] F. Bach. Consistency of trace norm minimization. *JMLR*, 9:1019–1048, 2008.

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

[4] A. Berg, J. Deng, and F.-F. Li. ImageNet large scale visual recognition challenge, 2010. `http://www.image-net.org/`.

[5] D. Bertsekas. *Nonlinear Programming (2nd ed.)*. Athena Scientific, 2004.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge UP, 2004.

[7] V. Chandrasekaran. *Convex Optimization Methods for Graphs and Statistical Modeling*. PhD thesis, MIT, 2011.

[8] O. Chapelle and Z. Harchaoui. A machine learning approach to conjoint analysis. In *Adv. NIPS*. 2005.

[9] K. Chen. *Matrix Preconditioning Techniques and Applications*. Cambridge UP, 2005.

[10] K. L. Clarkson. Coresets, sparse greedy approximation, and the Frank-wolfe algorithm. In *Proc. SODA*, 2008.

[11] M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 47, 2002.

[12] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46, 2002.

[13] V. Demyanov and A. Rubinov. *Approximate Methods in Optimization Problems*. American Elsevier, 1970.

[14] M. Dudík, S. J. Phillips, and R. E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *JMLR*, 8:1217–1260, 2007.

[15] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford, 2002.

[16] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[17] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010.

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning (2nd Ed.)*. Springer Series in Statistics. Springer, 2008.

[19] E. Hazan. Sparse approximate solutions to semidefinite programs. In *Proc. 8th Latin American Conf. Theor. Informatics*, pages 306–316, 2008.

[20] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer Verlag, Heidelberg, 1993. Two volumes.

[21] E. J. C. J-F Cai and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. Optimization*, 20(4):1956–1982, 2008.

[22] M. Jaggi and M. Sulovský. A simple algorithm for nuclear norm regularized problems. In *ICML*, 2010.

[23] G. Jameson. *Summing and nuclear norms in Banach space theory*. London Mathematical Society Student Texts, 8. Cambridge University Press. XI, 1987.

[24] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *ICML*, 2009.

[25] L. Mason, P. Bartlett, J. Baxter, and M. Frean. Functional gradient techniques for combining hypotheses. In B. Schölkopf, A. Smola, P. Bartlett, and D. Schuurmans, editors, *Adv. Large Margin Classifiers*. MIT Press, 2000.

[26] Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE, 2007.

[27] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. Technical report, CORE, 2010.

[28] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 2010.

[29] R. Phelps. *Convex functions, monotone operators, and differentiability*. Lecture notes in mathematics. Springer-Verlag, 1993.

[30] T. K. Pong, S. J. Paul Tseng, and J. Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM J. Optimization*, 20(6):3465–3489, 2010.

[31] B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[32] R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[33] S. Shalev-Shwartz, A. Gonen, and O. Shamir. Large-scale convex minimization with a low-rank constraint. In *ICML*, 2011.

[34] W. Shi, G. Wahba, S. Wright, K. Lee, R. Klein, and B. Klein. Lasso-patternsearch algorithm with application to ophthalmology and genomic data. *ASA Proceedings of the Joint Statistical Meetings*, 2006.

[35] S. Sra, S. Nowozin, and S. J. Wright. *Optimization for Machine Learning*. The MIT Press, 2010.

[36] N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-margin matrix factorization. In *Adv. NIPS*, 2005.

[37] A. Tewari, P. K. Ravikumar, and I. S. Dhillon. Greedy algorithms for structurally constrained high dimensional problems. In *Adv. NIPS*, 2011.

[38] S. Wright. Accelerated block-coordinate relaxation for regularized optimization. Technical report, 2010.

[39] G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *JMLR*, 11, December 2010.

[40] T. Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transaction on Information Theory*, 49:682–691, 2003.