Journal of Machine Learning Research 18 (2017) 1-46 Submitted 8/14; Revised 10/17; Published 11/17

Generalized Conditional Gradient for Sparse Estimation

Yaoliang Yu YAOLIANG.YU@UWATERLOO.CA
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada

Xinhua Zhang ZHANGX@UIC.EDU
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607

Dale Schuurmans DAES@UALBERTA.CA
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2E8, Canada

Editor: Koby Crammer

Abstract

Sparsity is an important modeling tool that expands the applicability of convex formulations for data analysis, however it also creates significant challenges for efficient algorithm design. In this paper we investigate the generalized conditional gradient (GCG) algorithm for solving sparse optimization problems—demonstrating that, with some enhancements, it can provide a more efficient alternative to current state of the art approaches. After studying the convergence properties of GCG for general convex composite problems, we develop efficient methods for evaluating polar operators, a subroutine that is required in each GCG iteration. In particular, we show how the polar operator can be efficiently evaluated in learning low-rank matrices, instantiated with detailed examples on matrix completion and dictionary learning. A further improvement is achieved by interleaving GCG with fixed-rank local subspace optimization. A series of experiments on matrix completion, multi-class classification, and multi-view dictionary learning shows that the proposed method can significantly reduce the training cost of current alternatives.

Keywords: generalized conditional gradient, frank-wolfe, dictionary learning, matrix completion, multi-view learning, sparse estimation

1. Introduction

Sparsity is an important concept in high-dimensional statistics (Bühlmann and van de Geer, 2011) and signal processing (Eldar and Kutyniok, 2012), which has led to important application successes by reducing model complexity and improving interpretability of the results. Although it is common to promote sparsity by adding appropriate regularizers, such as the $\mathfrak{l}_1$ norm, sparsity can also present itself in more sophisticated forms, such as group sparsity (Yuan and Lin, 2006), graph sparsity (Kim and Xing, 2009; Jacob et al., 2009), matrix rank sparsity (Candès and Recht, 2009), and other combinatorial sparsity patterns (Obozinski and Bach, 2012). These recent notions of *structured* sparsity (Bach et al., 2012;

©2017 Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans.

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`. Attribution requirements are provided at `http://jmlr.org/papers/v18/14-348.html`.

Micchelli et al., 2013) have greatly enhanced the ability to model structural relationships in data; however, they also create significant computational challenges.

A currently popular optimization scheme for training sparse models is *accelerated proximal gradient* (APG) (Beck and Teboulle, 2009; Nesterov, 2013), which enjoys an optimal rate of convergence among black-box first-order procedures (Nesterov, 2013). An advantage of APG is that each iteration consists solely of computing a *proximal update* (PU), which for simple regularizers can have a very low complexity. For example, under $\mathfrak{l}_1$ norm regularization each iterate of APG reduces to a soft-shrinkage operator that can be computed in linear time, which partly explains its popularity for such problems. Unfortunately, for matrix low-rank regularizers the PU soon becomes a computational bottleneck. For example, the trace norm is often used to promote low rank solutions in matrix variable problems such as matrix completion (Candès and Recht, 2009); but here the associated PU requires a *full* singular value decomposition (SVD) of the gradient matrix on each iteration, which prevents APG from being applied to large problems. Not only does the PU require nontrivial computation for matrix regularizers, it also yields *dense* intermediate iterates.

To address the primary shortcomings with APG, we investigate the *generalized conditional gradient* (GCG) strategy for sparse optimization, which has been motivated by the promise of recent sparse approximation methods (Hazan, 2008; Clarkson, 2010). A special case of GCG, known as the conditional gradient (CG), was originally proposed by Frank and Wolfe (1956) and has received significant renewed interest (Bach, 2015; Clarkson, 2010; Freund and Grigas, 2016; Hazan, 2008; Jaggi and Sulovsky, 2010; Jaggi, 2013; Shalev-Shwartz et al., 2010; Tewari et al., 2011; Yuan and Yan, 2013). The key advantage of GCG for sparse estimation is that it need only compute the *polar* of the regularizer in each iteration, which sometimes admits a much more efficient update than the PU in APG. For example, under trace norm regularization, GCG only requires the *spectral norm* of the gradient to be computed in each iteration, which is an order of magnitude cheaper than evaluating the full SVD as required by APG. Furthermore, the greedy nature of GCG affords explicit control over the sparsity of intermediate iterates, which is not available in APG. Although existing work on GCG has generally been restricted to *constrained* optimization (i.e. enforcing an upper bound on the sparsity-inducing regularizer), in this paper we consider the more general *regularized* version of the problem. Despite their theoretical equivalence, the regularized form allows more efficient local optimization to be interleaved with the primary update, which provides a significant acceleration in practice.

This paper is divided into two major parts: first we present a general treatment of GCG and establish its convergence properties for convex optimization in Section 3, with a special focus on convex gauge regularizers; then we apply the method to important case studies to demonstrate its effectiveness on large-scale, sparse estimation problems in Section 4, with experiments presented in Section 5.

In particular, in the first part, after establishing notation and briefly discussing relevant optimization schemes (Section 2), we present our first main contribution (Section 3): the convergence properties of GCG for general convex composite optimization. The algorithm is then specialized to convex gauge regularizers with refined analysis (Section 3.2). Indeed we prove that, under standard assumptions, GCG converges to the global optimum at the rate of $O(1/t)$—which is a significant improvement over previous bounds (Dudik et al., 2012). Although the convergence rate for GCG remains inferior to that of APG, these two algorithms demonstrate alternative trade-offs between the number of iterations required and the per-step complexity. As shown in our experiments, GCG can be significantly faster than APG in terms of *overall* computation time for large problems. The results in Section 3 are new.

Equipped with the technical results of GCG on convex optimization, the second part of the paper then applies the algorithm to the important application of low rank learning (Section 4). Since imposing a hard bound on matrix rank generally leads to an intractable problem, in Section 4.2 we first present a generic relaxation strategy based on convex gauges (Chandrasekaran et al., 2012; Tewari et al., 2011) that yields a convex program (Bach et al., 2008; Bradley and Bagnell, 2009; Zhang et al., 2012). Conveniently, the resulting problem can be easily optimized using the GCG algorithm from Section 3.2. To further reduce computation time, we introduce in Section 4.3 an auxiliary fixed rank subspace optimization within each iteration of GCG. Although similar hybrid approaches have been previously suggested (Burer and Monteiro, 2005; Laue, 2012; Mishra et al., 2013), we propose an efficient new alternative that, instead of locally optimizing a *constrained* fixed rank problem, optimizes an *unconstrained* surrogate objective based on a variational representation of matrix norms. This alternative strategy allows a far more efficient local optimization without compromising GCG's convergence properties. In Section 4.4 we show that the approach can be applied to dictionary learning and matrix completion problems, as well as a non-trivial multi-view form of dictionary learning (White et al., 2012).

Finally, in Section 5 we provide an extensive experimental evaluation that compares the performance of GCG (augmented with interleaved local search) to state-of-the-art optimization strategies, across a range of problems, including matrix completion, multi-class classification, and multi-view dictionary learning.

1.1 Extensions over Previously Published Work

Some of the results in this article have appeared in a preliminary form in two conference papers (Zhang et al., 2012; White et al., 2012); however, several specific extensions have been added in this paper, including:

- A more extensive and unified treatment of GCG in general Banach spaces.
- A more refined analysis of the convergence properties of GCG.
- A new application of GCG to multi-view dictionary learning.
- New experimental evaluations on larger data sets, including new results for matrix completion and image multi-class classification.
- Open source Matlab code released at `https://www.cs.uic.edu/~zhangx/GCG`.

2. Preliminaries

We first establish some of the definitions and notation that will be used throughout the paper, before providing a brief background on the relevant optimization methods that establish a context for discussing GCG.

2.1 Notation and Definitions

Throughout this paper we assume that the underlying space, unless otherwise stated, is a real Banach space $\mathcal{B}$ with norm $\|\cdot\|$; for example, $\mathcal{B}$ could be the Euclidean space $\mathbb{R}^d$ with any norm. The dual space (the set of all real-valued continuous linear functions on $\mathcal{B}$) is denoted $\mathcal{B}^*$ and is equipped with the dual norm $\|\mathbf{g}\|^\circ = \sup\{\mathbf{g}(\mathbf{w}) : \|\mathbf{w}\| \leq 1\}$. Note that for a differentiable function $f : \mathcal{B} \to \overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$, its (Frechét) derivative ∇f is in the dual space $\mathcal{B}^*$.

We use bold lowercase letters to denote vectors, where the i-th component of a vector $\mathbf{w}$ is denoted w_i, while $\mathbf{w}_t$ denotes some other vector. We use the shorthand $\langle \mathbf{w}, \mathbf{g} \rangle := \mathbf{g}(\mathbf{w})$ for any $\mathbf{g} \in \mathcal{B}^*$ and $\mathbf{w} \in \mathcal{B}$. We call a function $f : \mathcal{B} \to \overline{\mathbb{R}}$ σ-strongly convex (w.r.t. the norm $\|\cdot\|$) if there exists some $\sigma \geq 0$ such that for all $0 \leq \lambda \leq 1$ and $\mathbf{w}, \mathbf{z} \in \mathcal{B}$,

$$f(\lambda \mathbf{w} + (1-\lambda)\mathbf{z}) + \tfrac{1}{2}\sigma\lambda(1-\lambda)\|\mathbf{w} - \mathbf{z}\|^2 \leq \lambda f(\mathbf{w}) + (1-\lambda) f(\mathbf{z}). \tag{1}$$

In the case where $\sigma = 0$ we simply say f is convex. A set $C \subseteq \mathcal{B}$ is called convex if for all $\mathbf{w}, \mathbf{z} \in C \implies \lambda\mathbf{w} + (1-\lambda)\mathbf{z} \in C$ for all $0 \leq \lambda \leq 1$. It follows from the definition that the domain of a convex function f, denoted as $\operatorname{dom} f := \{\mathbf{w} : f(\mathbf{w}) < \infty\}$, is a convex set. The function f is called proper if $\operatorname{dom} f \neq \emptyset$, and closed if all sublevel sets $\{\mathbf{w} : f(\mathbf{w}) \leq \alpha\}$ are closed for all $\alpha \in \mathbb{R}$. The subdifferential of a convex function $f : \mathcal{B} \to \overline{\mathbb{R}}$ at $\mathbf{w}$ is defined as the set $\partial f(\mathbf{w}) = \{\mathbf{g} \in \mathcal{B}^* : f(\mathbf{z}) \geq f(\mathbf{w}) + \langle \mathbf{z} - \mathbf{w}, \mathbf{g}\rangle, \forall \mathbf{z} \in \mathcal{B}\}$.

Key to our subsequent development is two particular functions associated with a nonempty set in $\mathcal{B}$: the indicator function

$$\iota_C(\mathbf{w}) = \begin{cases} 0, & \text{if } \mathbf{w} \in C \\ \infty, & \text{otherwise} \end{cases}, \tag{2}$$

and the gauge function

$$\kappa(\mathbf{w}) = \kappa_A(\mathbf{w}) := \inf\{\rho \geq 0 : \mathbf{w} \in \rho A\}, \tag{3}$$

where $\rho A := \{\rho \mathbf{w} : \mathbf{w} \in A\}$, and the infimum is taken to be infinity over an empty set (i.e. when the condition $\mathbf{w} \in \rho A$ is not satisfied for any $\rho \geq 0$). Intuitively, $\kappa(\mathbf{w})$ is the smallest rescaling of the set A to "barely" contain $\mathbf{w}$; see Figure 1. Note that we always have $\kappa(\mathbf{0}) = 0$, and the gauge function is always positive homogeneous (i.e., $\kappa(t\mathbf{w}) = t\kappa(\mathbf{w})$ for all $t \geq 0$). Let $\mathsf{B} = \mathsf{B}_\kappa := \{\mathbf{w} : \kappa(\mathbf{w}) \leq 1\}$ denote the "unit ball" of κ, and then $\kappa_{\mathsf{B}} = \kappa_A$. B_κ is obviously intrinsic to κ, while different choices of A may yield the same gauge κ_A. Clearly, the indicator function ι_C is convex iff C is convex, while the gauge function κ is convex iff its unit ball is convex. Moreover, the gauge κ is a semi-norm iff its unit ball B is convex and symmetric (i.e. $\mathsf{B} = -\mathsf{B}$). Specifically, all semi-norms are convex gauge functions.

In practice, it is natural to engineer κ_A from a general "atomic set" A, which does not have to be convex (and can even be discrete) or contain the origin. Examples include the set of permutation matrices (Chandrasekaran et al., 2012), and the set of $\mathbf{x}\mathbf{x}^\top$ with $\|\mathbf{x}\|_2 = 1$ and $x_i \geq 0$ for low-rank nonnegative matrix factorization ($\mathbf{x}^\top$ is the transpose of $\mathbf{x}$). In this paper, we will work only with *convex closed* gauge functions, for which a sufficient *but not necessary* condition is that A is convex, closed, and containing the origin (an assumption

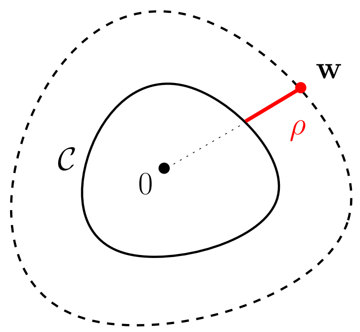


Figure 1: Defining a convex gauge via the Minkowski functional of a convex set $\mathcal{C}$.

made by many textbooks). For practical flexibility, we intentionally avoid making the latter assumptions in the first place, and the resulting technical complication is minimal.

A related function that underpins our algorithm is the *polar* of a gauge function κ_A, defined for all $\mathbf{g} \in \mathcal{B}^*$ as

$$\kappa^\circ(\mathbf{g}) = \kappa_A^\circ(\mathbf{g}) := \inf\{\mu \geq 0 : \langle \mathbf{w}, \mathbf{g} \rangle \leq \mu\kappa(\mathbf{w}),\ \forall \mathbf{w} \in \mathcal{B}\} \tag{4}$$

$$= \sup_{\mathbf{w} \in \mathsf{B}_\kappa} \langle \mathbf{w}, \mathbf{g} \rangle = \sup_{\mathbf{w} \in A \cup \{\mathbf{0}\}} \langle \mathbf{w}, \mathbf{g} \rangle . \tag{5}$$

As the notation suggests, the polar of a norm (a *bona fide* convex gauge) is its dual norm.

Lastly, a differential function $\ell : \mathcal{B} \to \mathbb{R}$ is said to be L-smooth (w.r.t. the norm $\|\cdot\|$) if for all $\mathbf{w}$, $\mathbf{z} \in \mathcal{B}$,

$$\ell(\mathbf{w}) \leq \tilde{\ell}_L(\mathbf{w}; \mathbf{z}) := \ell(\mathbf{z}) + \langle \mathbf{w} - \mathbf{z}, \nabla\ell(\mathbf{z}) \rangle + \tfrac{L}{2} \|\mathbf{w} - \mathbf{z}\|^2 , \tag{6}$$

i.e., ℓ is upper bounded by the quadratic approximation around $\mathbf{z}$. It is well-known that if the gradient $\nabla\ell$ is L-Lipschitz continuous, i.e., $\|\nabla\ell(\mathbf{w}) - \nabla\ell(\mathbf{z})\|^\circ \leq L\|\mathbf{w} - \mathbf{z}\|$ for all $\mathbf{w}, \mathbf{z}$, then ℓ is L-smooth. The converse is also true if ℓ is additionally convex. We will make repeated use of this quadratic upper bound in our analysis below.

2.2 Composite Minimization Problem

Many machine learning problems, particularly those formulated in terms of regularized risk minimization, can be formulated as:

$$\inf_{\mathbf{w} \in \mathcal{B}} F(\mathbf{w}), \text{ such that } F(\mathbf{w}) := \ell(\mathbf{w}) + f(\mathbf{w}), \tag{7}$$

where f is convex and ℓ is convex and continuously differentiable. Typically, ℓ is a loss function and f is a regularizer although their roles can be reversed in some common examples, such as support vector machines and Lasso.

Due to the importance of this problem in machine learning, significant effort has been devoted to designing efficient algorithms for solving (7). Standard methods are generally based on computing an update sequence $\{\mathbf{w}_t\}$ that converges to a global minimizer of F when F is convex, or converges to a stationary point otherwise. For example, a popular example is the mirror descent (MD) algorithm (Beck and Teboulle, 2003), which takes an arbitrary subgradient $\mathbf{g}_t \in \partial F(\mathbf{w}_t)$ and successively updates by

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left\{ \langle \mathbf{w}, \mathbf{g}_t \rangle + \frac{1}{2\eta_t} \mathsf{D}(\mathbf{w}, \mathbf{w}_t) \right\} \tag{8}$$

with a suitable step size $\eta_t \geq 0$; here $\mathsf{D}(\mathbf{w}, \mathbf{z}) := \mathsf{d}(\mathbf{w}) - \mathsf{d}(\mathbf{z}) - \langle \mathbf{w} - \mathbf{z}, \nabla \mathsf{d}(\mathbf{z}) \rangle$ denotes a Bregman divergence induced by some differentiable 1-strongly convex function $\mathsf{d} : \mathcal{B} \to \overline{\mathbb{R}}$. Under mild assumptions, MD converges (in terms of the function value) at a rate of $O(1/\sqrt{t})$ (Beck and Teboulle, 2003). However, this algorithm ignores the composite structure in (7) and can be impractically slow for many applications.

Another widely used algorithm is the proximal gradient (PG) (e.g. Fukushima and Mine, 1981), also known as the forward-backward splitting procedure, which iteratively performs the proximal update (PU)

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left\{ \langle \mathbf{w}, \nabla \ell(\mathbf{w}_t) \rangle + f(\mathbf{w}) + \frac{1}{2\eta_t} \mathsf{D}(\mathbf{w}, \mathbf{w}_t) \right\}. \tag{9}$$

PU differs from MD updates in that it does not linearize the regularizer f. The downside is that (9) is usually harder to solve than (8) for each iteration, with an upside that a faster $O(1/t)$ rate of convergence can be achieved (Tseng, 2010). This rate can be further improved to $O(1/t^2)$ with the extrapolation variant known as accelerated proximal gradient (APG) (Beck and Teboulle, 2009; Nesterov, 2013). In particular, suppose $\mathcal{B} = \mathbb{R}^d$, $\mathsf{D}(\mathbf{w}, \mathbf{z}) = \frac{1}{2} \|\mathbf{w} - \mathbf{z}\|_2^2$, and $f(\mathbf{w}) = \|\mathbf{w}\|_1$, where the $\mathfrak{l}_\mathsf{p}$ norm is defined as $\|\mathbf{w}\|_\mathsf{p} := (\sum_{i=1}^d |w_i|^\mathsf{p})^{1/\mathsf{p}}$ for $\mathsf{p} \geq 1$. Then PU recovers the soft-shrinkage operator, which plays an important role in sparse estimation:

$$\mathbf{z}_t = \mathbf{w}_t - \eta_t \nabla \ell(\mathbf{w}_t), \qquad \text{and} \qquad \mathbf{w}_{t+1} = (1 - \eta_t / |\mathbf{z}_t|)_+ \, \mathbf{z}_t. \tag{10}$$

Here the algebraic operations in the latter formula are performed componentwise. Similarly, when $\mathcal{B} = \mathbb{R}^{m \times n}$, $\mathsf{D}(W, Z) = \frac{1}{2} \|W - Z\|_\mathrm{F}^2$ (Frobenius norm) and $f(W) = \|W\|_\mathrm{tr}$ (trace norm, i.e. sum of singular values), one recovers the matrix analogue of soft-shrinkage used for matrix completion (Candès and Recht, 2009): here the same gradient update of $\mathbf{z}_t$ in (10) is applied, but the shrinkage operator on $\mathbf{w}_{t+1}$ in (10) is applied to the singular values of Z_t while keeping the singular vectors intact. Unfortunately, such a PU can become prohibitively expensive for large matrices, since it requires a *full* singular value decomposition (SVD) at each iteration, at a cubic time cost. We will see in Section 3.2 that an alternative algorithm only requires computing the dual of the trace norm (namely the spectral norm $\|\cdot\|_\mathrm{sp}$, i.e. the greatest singular value), which requires only quadratic time per iteration.

3. Generalized Conditional Gradient for Convex Optimization

In this section we present the generalized conditional gradient (GCG) algorithm, which has recently been receiving renewed attention (Bredies and Lorenz, 2008; Hazan, 2008; Jaggi, 2013; Shalev-Shwartz et al., 2010; Yuan and Yan, 2013; Zhang et al., 2012). Our goal in this section will be to develop efficient solution methods for (7), where f is convex and ℓ

is convex and L-smooth (cf. (6)). In particular, we will refine our algorithms to convex gauge regularized problems where special computational savings can be achieved through its polar. These are clearly among the most important settings in practice, and after some enhancements the algorithm and convergence guarantees we establish in this section will be applied to dictionary learning in Section 4.

3.1 General Setting with Convex and Smooth ℓ

We first develop the framework of the GCG algorithm and its analysis by focusing on general convex functions f. Formally, we make the following assumption throughout this Section 3.

Assumption 1 *ℓ is convex and L-smooth, and f is convex.*

Protocol: To avoid unnecessary pathologies, we tacitly assume all functions considered in this work are proper and closed.

To derive the optimization algorithm, first observe that given the above assumption, $F = \ell + f$ is convex and so any $\mathbf{w} \in \mathcal{B}$ is globally optimal for (7) if, and only if,

$$\mathbf{0} \in \partial F(\mathbf{w}) = \nabla \ell(\mathbf{w}) + \partial f(\mathbf{w}). \tag{11}$$

Let $f^*(\mathbf{g}) := \sup_{\mathbf{w}} \langle \mathbf{w}, \mathbf{g} \rangle - f(\mathbf{w})$ denote the Fenchel conjugate of f, and using the fact that $\mathbf{g} \in \partial f(\mathbf{w})$ iff $\mathbf{w} \in \partial f^*(\mathbf{g})$ when f is closed (Borwein and Vanderwerff, 2010), one can observe that the necessary and sufficient condition for $\mathbf{w}$ to be globally optimal can also be expressed as

$$(11) \quad \Longleftrightarrow \quad \mathbf{w} \in \partial f^*(-\nabla \ell(\mathbf{w})) \quad \Longleftrightarrow \quad \mathbf{w} \in (1-\eta)\mathbf{w} + \eta \partial f^*(-\nabla \ell(\mathbf{w})), \tag{12}$$

where $\eta \in (0,1)$. Then, by the definition of f^*, we have

$$\mathbf{d} \in \partial f^*(-\nabla \ell(\mathbf{w})) \quad \Longleftrightarrow \quad \mathbf{d} \in \arg\min_{\mathbf{d}} \left\{ \langle \mathbf{d}, \nabla \ell(\mathbf{w}) \rangle + f(\mathbf{d}) \right\}. \tag{13}$$

Thus, the condition for a global minimum can be characterized by the fixed-point inclusion

$$\mathbf{w} \in (1-\eta)\mathbf{w} + \eta \mathbf{d} \quad \text{for some} \quad \mathbf{d} \in \arg\min_{\mathbf{d}} \left\{ \langle \mathbf{d}, \nabla \ell(\mathbf{w}) \rangle + f(\mathbf{d}) \right\}. \tag{14}$$

This particular fixed-point condition immediately suggests an update that provides the foundation for the generalized conditional gradient algorithm (GCG) outlined in Algorithm 1. Each iteration of this procedure involves linearizing the smooth loss ℓ, solving the subproblem (13), selecting a step size, taking a convex combination, then conducting some form of local improvement. The GCG update based on (14) naturally generalizes the original conditional gradient update, which was first studied by (Frank and Wolfe, 1956) for the case when $f = \iota_C$ for a polyhedral set C.

Note that the subproblem (13) shares some similarity with the proximal update (9): both choose to leave the potentially nonsmooth function f intact, but here the smooth loss ℓ is replaced by its plain linearization rather than its linearization *plus* a (strictly convex) proximal regularizer such as the quadratic upper bound $\tilde{\ell}_L(\cdot)$ in (6). Consequently, there might be *multiple* solutions, in which case we simply adopt any one, or no solution (e.g.

Algorithm 1 Generalized Conditional Gradient (GCG).

```
1: Initialize: $\mathbf{w}_0 \in \operatorname{dom} f$.
2: for $t = 0, 1, \ldots$ do
3:     Compute $\mathbf{d}_t \in \arg\min_{\mathbf{d}} \{\langle \mathbf{d}, \mathbf{g}_t \rangle + f(\mathbf{d})\}$, where $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t)$.
4:     Choose step size $\eta_t \in [0, 1]$ and perform update $\tilde{\mathbf{w}}_{t+1} = (1 - \eta_t)\mathbf{w}_t + \eta_t \mathbf{d}_t$.
5:     $\mathbf{w}_{t+1} = \texttt{Improve}(\tilde{\mathbf{w}}_{t+1}, \ell, f)$          ▷ Subroutine, see Definition 1
6: end for
```

divergence), in which case we will impose extra assumptions to ensure boundedness (details below, especially Assumption 2 and Theorem 2).

An important component of Algorithm 1 is the final step, `Improve`, in Line 5, which is particularly important in practice: it allows for heuristic local improvement to be conducted on the iterates without sacrificing any theoretical convergence properties of the overall procedure (see Section 4.3). A precursor of `Improve` appeared in (Meyer, 1974, Algorithm 2). Since `Improve` has access to both ℓ and f it can be very powerful in principle—we will consider the following variants that have respective consequences for practice and subsequent analysis.

Definition 1 *The subroutine `Improve` in Algorithm 1 is called `Null` if for all t, $\mathbf{w}_{t+1} = \tilde{\mathbf{w}}_{t+1}$; `Descent` if $F(\mathbf{w}_{t+1}) \le F(\tilde{\mathbf{w}}_{t+1})$; `Monotone` if $F(\mathbf{w}_{t+1}) \le F(\mathbf{w}_t)$; and `Relaxed` if*

$$F(\mathbf{w}_{t+1}) \le \tilde{\ell}_L(\tilde{\mathbf{w}}_{t+1}; \mathbf{w}_t) + (1 - \eta_t) f(\mathbf{w}_t) + \eta_t f(\mathbf{d}_t). \tag{15}$$

Obviously, `Relaxed` can be easily morphed to further satisfy `Descent` and/or `Monotone` by comparing $F(\mathbf{w}_{t+1})$ with $F(\tilde{\mathbf{w}}_{t+1})$ and/or $F(\mathbf{w}_t)$, whereas `Null` and `Descent` must be `Relaxed` (since ℓ is L-smooth).

Our main goal remains to establish that Algorithm 1 indeed converges to a global optimum under general conditions given in Assumption 1. In order to address the aforementioned issue of bounding $\mathbf{d}_t$, we next introduce one more assumption.

Assumption 2 *The sequences $\{\mathbf{d}_t\}$ and $\{\mathbf{w}_t\}$ generated by the algorithm are bounded for any choice of $\eta_t \in [0, 1]$.*

Although this assumption is more stringent, it can fortunately be achieved under various conditions, which we summarize as follows. The proof is given in Appendix A.1.

Proposition 2 *Let Assumption 1 hold. Then Assumption 2 is satisfied if any of the following holds.*

(a) The subroutine `Improve` is `Monotone`; the sublevel set $\{\mathbf{w} \in \operatorname{dom} f : F(\mathbf{w}) \le F(\mathbf{w}_0)\}$ is compact; and f is cofinite, i.e., its Fenchel conjugate f^* has full domain.

(b) The subroutine `Improve` is `Monotone`; the sublevel set $\{\mathbf{w} \in \operatorname{dom} f : F(\mathbf{w}) \le F(\mathbf{w}_0)\}$ is bounded; and f is super-coercive, i.e., $\lim_{\|\mathbf{w}\| \to \infty} f(\mathbf{w}) / \|\mathbf{w}\| \to \infty$.

(c) $\operatorname{dom} f$ is bounded.

In particular, under any of the conditions in Theorem 2, Line 3 of Algorithm 1 is well-defined. Note that if $\mathcal{B} = \mathbb{R}^d$, then condition (a) is in fact equivalent to condition (b). On the other hand, if f is a convex gauge, then none of the three conditions above holds, and we will deal with this in Section 3.2.

Finally, for each $\mathbf{w} \in \mathcal{B}$ we define a quantity, referred to as the *duality gap*, that will be useful in understanding the convergence properties of GCG:

$$\begin{aligned}\mathsf{G}(\mathbf{w}) &:= F(\mathbf{w}) - \inf_{\mathbf{d}} \{\ell(\mathbf{w}) + \langle \mathbf{d} - \mathbf{w}, \nabla\ell(\mathbf{w})\rangle + f(\mathbf{d})\} \quad \text{(linearizing } \ell \text{ at } \mathbf{w}) &(16)\\ &= \langle \mathbf{w}, \nabla\ell(\mathbf{w})\rangle + f(\mathbf{w}) - \inf_{\mathbf{d}} \{\langle \mathbf{d}, \nabla\ell(\mathbf{w})\rangle + f(\mathbf{d})\} \\ &= \langle \mathbf{w}, \nabla\ell(\mathbf{w})\rangle + f(\mathbf{w}) + f^*(-\nabla\ell(\mathbf{w})). \quad \text{(definition of Fenchel dual)} &(17)\end{aligned}$$

Since f and ℓ are convex in Assumption 1, the duality gap provides an upper bound on the suboptimality of any search point, as established in the following proposition.

Proposition 3 *Let Assumption 1 hold. Then* $\mathsf{G}(\mathbf{w}) \geq F(\mathbf{w}) - \inf_{\mathbf{d}} F(\mathbf{d}) \geq 0$ *for all* $\mathbf{w} \in \mathcal{B}$. *Furthermore,* $\mathsf{G}(\mathbf{w}) = 0$ *iff* $\mathbf{w}$ *is globally optimal, i.e.* $\mathbf{w}$ *satisfies the condition* (11)*.*

Proof As ℓ is convex, (16) implies $\mathsf{G}(\mathbf{w}) \geq F(\mathbf{w}) - \inf_{\mathbf{d}}\{\ell(\mathbf{d}) + f(\mathbf{d})\} = F(\mathbf{w}) - \inf_{\mathbf{d}} F(\mathbf{d}) \geq 0$. By the Fenchel-Young inequality, (17) implies $\mathsf{G}(\mathbf{w}) = 0$ iff $\mathbf{w} \in \partial f^*(-\nabla\ell(\mathbf{w}))$, i.e. (11) holds. ■

Since $\mathsf{G}(\mathbf{w}_t)$ is an upper bound on the suboptimality of $\mathbf{w}_t$, the duality gap gives a natural stopping criterion for (7) that can be easily computed in each iteration of Algorithm 1.

We are now ready for the first convergence result regarding Algorithm 1.

Theorem 4 *Let Assumptions 1 and 2 hold. Also assume the subroutine is* `Relaxed`*, and the subproblem* (13) *is solved up to some additive error* $\varepsilon_t \geq 0$*. Then, for any* $\mathbf{w} \in \operatorname{dom} F$ *and* $t \geq 0$*, Algorithm 1 yields*

$$F(\mathbf{w}_{t+1}) \leq F(\mathbf{w}) + \pi_t(1-\eta_0)(F(\mathbf{w}_0) - F(\mathbf{w})) + \sum_{s=0}^{t} \frac{\pi_t}{\pi_s}\eta_s^2(\varepsilon_s/\eta_s + \tfrac{L}{2}\|\mathbf{d}_s - \mathbf{w}_s\|^2), \quad (18)$$

where $\pi_t := \prod_{s=1}^{t}(1-\eta_s)$ *with* $\pi_0 = 1$*. Furthermore, for all* $t \geq k \geq 0$*, the minimal duality gap* $\tilde{\mathsf{G}}_k^t := \min_{k\leq s\leq t} \mathsf{G}(\mathbf{w}_s)$ *satisfies*

$$\tilde{\mathsf{G}}_k^t \leq \frac{1}{\sum_{s=k}^{t}\eta_s}\left[F(\mathbf{w}_k) - F(\mathbf{w}_{t+1}) + \sum_{s=k}^{t}\eta_s^2\left(\varepsilon_s/\eta_s + \tfrac{L}{2}\|\mathbf{d}_s - \mathbf{w}_s\|^2\right)\right]. \quad (19)$$

From Theorem 4 we derive the following concrete rates of convergence.

Corollary 5 *Under the same assumptions as Theorem 4, also let* $\eta_t = 2/(t+2)$, $\varepsilon_t \leq \delta\eta_t/2$ *for some* $\delta \geq 0$*, and* $L_F := \sup_t L\|\mathbf{d}_t - \mathbf{w}_t\|^2$*. Then, for all* $t \geq 1$ *and* $\mathbf{w}$*, Algorithm 1 yields*

$$F(\mathbf{w}_t) \leq F(\mathbf{w}) + \frac{2(\delta + L_F)}{t+3}, \qquad \textit{and} \qquad \tilde{\mathsf{G}}_1^t \leq \frac{3(\delta + L_F)}{t\ln 2} \leq \frac{4.5(\delta + L_F)}{t}. \quad (20)$$

The proofs of Theorem 4 and Theorem 5 can be found in Appendix A.2 and Appendix A.3 respectively. Note that the simple step size rule $\eta_t = 2/(t+2)$ already leads to an $O(1/t)$ bound on the objective value attained. Of course, it is possible to use other step size rules. For instance, both $\eta_s = 1/(s+1)$ and the constant rule $\eta_s = 1-(t+1)^{1/t}$ lead to an $O(\frac{1+\log t}{t+1})$ rate; see (Freund and Grigas, 2016) for detailed calculations.

We emphasize that GCG with the open-loop step size rule $\eta_t = 2/(t+2)$ requires neither the Lipschitz constant L nor the norm $\|\cdot\|$ to be known. We can therefore freely adopt the best choices for the problem at hand in the analysis. Note that, even though the rate does not depend on the initial point $\mathbf{w}_0$ provided $\eta_0 = 1$, by letting $\eta_0 \neq 1$ the bound can be slightly improved (Freund and Grigas, 2016).

Remark 6 *The observation that an $O(1/t)$ rate can be obtained by an open-loop step size rule such as $\eta_t = O(1/t)$ appears to have been first made by Dunn and Harshbarger (1978). The same rate can be achieved by the closed-loop step size rule where η_t is selected to minimize $\ell((1-\eta_t)\mathbf{w}_t+\eta_t\mathbf{d}_t)$ or its quadratic upper bound $\tilde{\ell}((1-\eta_t)\mathbf{w}_t+\eta_t\mathbf{d}_t;\mathbf{w}_t)$ (Frank and Wolfe, 1956; Levitin and Polyak, 1966; Dem'yanov and Rubinov, 1967). A similar rate on the minimal duality gap was also given in (Clarkson, 2010), and extended by (Jaggi, 2013). However, all these previous works focused on the special case where $f = \iota_C$ for some compact set C. Fukushima and Mine (1981) was the first to consider a general convex function f (and possibly a nonconvex ℓ), but they only established convergence for the algorithm. Bredies and Lorenz (2008) proved the $O(1/t)$ rate by using a closed-loop step size rule. More recently, Bach (2015) considered the special case where f is strongly convex, and identified the equivalence between GCG and MD, hence also establishing the $O(1/t)$ rate for this special case.*

3.2 Improved Algorithm and Refined Analysis when f is a Convex Gauge

Although the results in Theorem 4 and Theorem 5 hold for general convex functions f, they require $\{\mathbf{d}_t\}$ and $\{\mathbf{w}_t\}$ to be bounded as in Assumption 2. Some sufficient conditions were provided by Theorem 2. Unfortunately, these conditions cannot be satisfied by an important class of regularizers in machine learning: norms, semi-norms, and more generally convex *gauge* functions, cf. Section 2 for the detailed definition. Examples include the ℓ_p norms ($p \geq 1$), the total variation semi-norm, the trace norm for matrices, etc. Indeed, the subproblem (13) in Line 3 of Algorithm 1 might diverge to $-\infty$, leaving the update undefined. In this subsection, we develop a specialized form of GCG that circumvents this problem and exploits properties of convex gauges to significantly reduce computational overhead. The resulting GCG formulation, which is one of our main contributions, will be used to achieve efficient algorithms in important case studies in the second part of the paper, such as matrix completion under trace norm regularization (Section 4.2.1 below).

In the literature, two main approaches have been used to bypass the unboundedness in solving (13); unfortunately, both remain unsatisfactory in different ways. First, one obvious way to restore boundedness to (13) is to reformulate the regularized problem (7) in its equivalent constrained form

$$\inf_{\mathbf{w}} \ \ell(\mathbf{w}) \ \text{ s.t. } \ f(\mathbf{w}) \leq \zeta, \tag{21}$$

where it is well-known that a correspondence between (7) and (21) can be achieved if the constant ζ is chosen appropriately. In this case, Algorithm 1 can be directly applied as long as (21) has a bounded domain. Much recent work in machine learning has investigated this variant (Hazan, 2008; Jaggi and Sulovsky, 2010; Jaggi, 2013; Shalev-Shwartz et al., 2010; Tewari et al., 2011; Yuan and Yan, 2013). However, finding the appropriate value of ζ is costly in general, and often one cannot avoid considering the penalized formulation (e.g. when it is nested in another problem). Furthermore, the constraints in (21) preclude the application of many efficient local `Improve` techniques designed for *unconstrained* problems.

A second, alternative, approach is to square f, which ensures that it becomes super-coercive (Bradley and Bagnell, 2009). However, this modification is somewhat arbitrary, in the sense that super-coerciveness can be achieved by raising the norm to the pth power for any $p > 1$ (when $\mathcal{B} = \mathbb{R}^d$). Moreover, the insertion of a local `Improve` step in such an approach requires the regularizer f to be evaluated at all iterates, which is expensive for applications such as matrix completion.

3.2.1 Generalized Convex Gauge Regularization

To address the issue of ensuring well defined iterates, while also reducing their cost, we develop an alternative modification of GCG. The algorithm we develop also allows a mild generalization of convex gauge regularization by introducing a composition with an increasing convex function. In particular, let κ denote a convex gauge induced by a convex, closed set $\mathcal{C}$ containing the origin (cf. (3)) and let $h : \mathbb{R}_+ \to \overline{\mathbb{R}}$ denote an *increasing convex* function over $\mathbb{R}_+ := [0, \infty]$. Then regularizing by $f(\cdot) = h(\kappa(\cdot))$ yields the particular form of (7) that we will address with this modified approach:

$$\inf_{\mathbf{w}} \ \ell(\mathbf{w}) + h(\kappa(\mathbf{w})). \tag{22}$$

For the structured sparse regularizers typically considered in machine learning, the set $\mathcal{C}$ used to construct the gauge function κ via (3) has additional structure that admits efficient computation (Chandrasekaran et al., 2012). In particular, $\mathcal{C}$ is often constructed by taking the convex hull of some *compact* set of "atoms" $\mathcal{A}$ specific to the problem; i.e., $\mathcal{C} = \text{conv}\mathcal{A}$. Such a structured formulation allows one to re-express the gauge as the "$\mathfrak{l}_1$ norm" over an appropriate "atomic decomposition." In details:

$$\kappa_{\mathcal{C}}(\mathbf{w}) = \inf\left\{\sum_{i=1}^{r} \lambda_i : \mathbf{w} = \sum_{i=1}^{r} \lambda_i \mathbf{a}_i, \mathbf{a}_i \in \mathcal{A}, \lambda_i \geq 0\right\} \tag{23}$$

$$\kappa_{\mathcal{C}}^{\circ}(\mathbf{g}) = \sup_{\mathbf{w} \in \mathcal{A} \cup \{\mathbf{0}\}} \langle \mathbf{w}, \mathbf{g} \rangle . \tag{24}$$

That is, if we decompose $\mathbf{w}$ as a conic combination of the "atoms" in $\mathcal{A}$, then $\kappa(\mathbf{w})$ is the least $\mathfrak{l}_1$ norm of the decomposition coefficients. Determining the convex gauge $\kappa_{\mathcal{C}}$ directly from the set $\mathcal{C}$ is not always easy, and the following duality result may come into aid (Rockafellar and Wets, 1998, p. 491):

$$\mathcal{C} \text{ is closed convex and } \mathbf{0} \in \mathcal{C} \implies \kappa_{\mathcal{C}} = (\kappa_{\mathcal{C}}^{\circ})^{\circ}. \tag{25}$$

Since $\mathcal{A}$ is compact, the polar $\kappa_{\mathcal{C}}^{\circ}$ has full domain. Intuitively, κ° has a much simpler form than κ if the structure of $\mathcal{A}$ is "simple." In such cases, it is advantageous to design an optimization algorithm that circumvents evaluation of κ, by instead using κ° which is generally far less expensive to evaluate. This will be the underpinning motivation of our algorithm design.

3.2.2 Modified GCG Algorithm for Generalized Gauge Regularization

The key to the modified algorithm is to avoid the possibly unbounded solution to (13) while also avoiding explicit evaluation of the gauge κ. First, to avoid unboundedness, we adopt the technique of moving the regularizer to the constraint, as in (21), but rather than applying the modification directly to (22) we only use it in the subproblem (13) via:

$$\mathbf{d}_t \in \arg\min_{\mathbf{d}:h(\kappa(\mathbf{d}))\le\zeta} \langle \mathbf{d}, \nabla\ell(\mathbf{w}_t)\rangle . \tag{26}$$

Then, to bypass the complication of dealing with h and the unknown bound ζ, we decompose $\mathbf{d}_t$ into its normalized direction $\mathbf{a}_t$ and scale θ_t (i.e., such that $\mathbf{d}_t = \mathbf{a}_t\theta_t/\eta_t$), determining each separately. In particular, from this decomposition, the update from Line 4 of Algorithm 1 can be equivalently expressed as

$$\mathbf{w}_{t+1} = (1-\eta_t)\mathbf{w}_t + \eta_t\mathbf{d}_t = (1-\eta_t)\mathbf{w}_t + \theta_t\mathbf{a}_t, \tag{27}$$

hence the modified algorithm need only work with $\mathbf{a}_t$ and θ_t directly.

Determining the direction. First, the normalized direction, $\mathbf{a}_t$, can be recovered via

$$\mathbf{a}_t \in \arg\min_{\mathbf{a}:\kappa(\mathbf{a})\le 1} \langle \mathbf{a}, \nabla\ell(\mathbf{w}_t)\rangle \iff \mathbf{a}_t \in \arg\min_{\mathbf{a}\in\mathcal{A}\cup\{\mathbf{0}\}} \langle \mathbf{a}, \nabla\ell(\mathbf{w}_t)\rangle \qquad \text{(by (4))} \tag{28}$$

$$\iff \mathbf{a}_t \in \arg\max_{\mathbf{a}\in\mathcal{A}\cup\{\mathbf{0}\}} \langle \mathbf{a}, -\nabla\ell(\mathbf{w}_t)\rangle . \tag{29}$$

Observe that (29) effectively involves computation of the polar $\kappa^\circ(-\nabla\ell(\mathbf{w}))$ only, by (4). Since this optimization might still be challenging, we further allow it to be solved only *approximately* to within an additive error $\varepsilon_t \ge 0$ and a multiplicative factor $\alpha_t \in (0,1]$; that is, by relaxing (28) the modified algorithm only requires an $\mathbf{a}_t \in \mathcal{A}$ to be found that satisfies

$$\langle \mathbf{a}_t, \nabla\ell(\mathbf{w}_t)\rangle \le \alpha_t\left(\varepsilon_t + \min_{\mathbf{a}\in\mathcal{A}\cup\{\mathbf{0}\}} \langle \mathbf{a}, \nabla\ell(\mathbf{w}_t)\rangle\right) = \alpha_t\left(\varepsilon_t - \kappa^\circ(-\nabla\ell(\mathbf{w}_t))\right). \tag{30}$$

Intuitively, this formulation computes the normalized direction that has (approximately) the greatest negative "correlation" with the gradient $\nabla\ell$, yielding the steepest local decrease in ℓ. Importantly, this formulation does *not* require evaluation of the gauge function κ.

Determining the scale. Next, to choose the scale θ_t, it is natural to consider minimizing a simple upper bound on the objective that is obtained by replacing $\ell(\cdot)$ with its quadratic upper approximation $\tilde{\ell}_L(\cdot;\mathbf{w}_t)$ given in (6):

$$\min_{\theta\ge 0} \tilde{\ell}_L((1-\eta_t)\mathbf{w}_t + \theta\mathbf{a}_t;\mathbf{w}_t) + h(\kappa((1-\eta_t)\mathbf{w}_t + \theta\mathbf{a}_t)). \tag{31}$$

Unfortunately, κ still participates in (31), so a further approximation is required. Here is where the decomposition of $\mathbf{d}_t$ into $\mathbf{a}_t$ and θ_t is particularly advantageous. Observe that

$$\begin{aligned} h(\kappa((1-\eta_t)\mathbf{w}_t + \theta\mathbf{a}_t)) &\le h((1-\eta_t)\kappa(\mathbf{w}_t) + \eta_t\kappa(\theta\mathbf{a}_t/\eta_t)) \\ &\le (1-\eta_t)h(\kappa(\mathbf{w}_t)) + \eta_t h(\kappa(\theta\mathbf{a}_t/\eta_t)) \qquad (32) \\ &\le (1-\eta_t)h(\kappa(\mathbf{w}_t)) + \eta_t h(\theta/\eta_t)), \qquad (33) \end{aligned}$$

Algorithm 2 GCG for positively homogeneous regularizers.

```
Require: The set $\mathcal{A}$ whose convex hull $\mathcal{C}$ defines the gauge $\kappa$.
1: Initialize $\mathbf{w}_0$ and $\rho_0 \geq \kappa(\mathbf{w}_0)$.
2: for $t = 0, 1, \ldots$ do
3:     Choose normalized direction $\mathbf{a}_t$ that satisfies (30).
4:     Choose step size $\eta_t \in [0,1]$ and set the scaling $\theta_t \geq 0$ by (34)
5:     $\tilde{\mathbf{w}}_{t+1} = (1-\eta_t)\mathbf{w}_t + \theta_t \mathbf{a}_t$, and $\tilde{\rho}_{t+1} = (1-\eta_t)\rho_t + \theta_t$
6:     $(\mathbf{w}_{t+1}, \rho_{t+1}) = \mathtt{Improve}(\tilde{\mathbf{w}}_{t+1}, \tilde{\rho}_{t+1}, \ell, f)$        ▷ Subroutine, see Theorem 8
7: end for
```

where the first inequality follows from the convexity of κ and the fact that h is increasing, the second inequality follows from the convexity of h, and (33) follows from the fact that $\kappa(\mathbf{a}_t) \leq 1$ by construction ($\mathbf{a}_t \in \mathcal{A}$ in (4)). Although κ still participates in (33) we can now bypass evaluation of κ by maintaining a simple upper bound $\rho_t \geq \kappa(\mathbf{w}_t)$, yielding the relaxed update

$$\theta_t = \arg\min_{\theta \geq 0} \left\{ \tilde{\ell}_L((1-\eta_t)\mathbf{w}_t + \theta \mathbf{a}_t; \mathbf{w}_t) + (1-\eta_t)h(\rho_t) + \eta_t h(\theta/\eta_t) \right\}. \tag{34}$$

Crucially, the upper bound $\rho_t \geq \kappa(\mathbf{w}_t)$ can be maintained by the simple update $\rho_{t+1} = (1-\eta_t)\rho_t + \theta_t$, which is sufficient to ensure that ρ_{t+1} upper bounds $\kappa(\mathbf{w}_{t+1})$:

$$\rho_{t+1} = (1-\eta_t)\rho_t + \theta_t \geq (1-\eta_t)\kappa(\mathbf{w}_t) + \theta_t \kappa(\mathbf{a}_t) \geq \kappa((1-\eta_t)\mathbf{w}_t + \theta_t \mathbf{a}_t) \geq \kappa(\mathbf{w}_{t+1}). \tag{35}$$

From these components we obtain the final modified procedure, Algorithm 2: a variant of GCG (Algorithm 1) that avoids the potential unboundedness of the subproblem (13) by replacing it with (30), while also completely avoiding explicit evaluation of the gauge κ. The `Improve` subroutine can be adapted by mirroring Theorem 1.

Definition 7 *The subroutine `Improve` in Algorithm 2 is called `Null` if for all t, $\mathbf{w}_{t+1} = \tilde{\mathbf{w}}_{t+1}$ and $\rho_{t+1} = \tilde{\rho}_{t+1}$. Provided that $\rho_{t+1} \geq \kappa(\mathbf{w}_{t+1})$, `Improve` is called `Descent` if $\ell(\mathbf{w}_{t+1}) + h(\rho_{t+1}) \leq \ell(\tilde{\mathbf{w}}_{t+1}) + h(\tilde{\rho}_{t+1})$; `Monotone` if $\ell(\mathbf{w}_{t+1}) + h(\rho_{t+1}) \leq \ell(\mathbf{w}_t) + h(\rho_t)$; and `Relaxed` if*

$$\ell(\mathbf{w}_{t+1}) + h(\rho_{t+1}) \leq \tilde{\ell}_L(\tilde{\mathbf{w}}_{t+1}; \mathbf{w}_t) + (1-\eta_t)h(\rho_t) + \eta_t h(\theta_t/\eta_t). \tag{36}$$

Similar to Theorem 1, `Relaxed` can be easily morphed to further satisfy `Descent` and/or `Monotone` by comparing $(\mathbf{w}_{t+1}, \rho_{t+1})$ with $(\tilde{\mathbf{w}}_{t+1}, \tilde{\rho}_{t+1})$ and/or $(\mathbf{w}_t, \rho_t)$ over $\ell(\mathbf{w}) + h(\rho)$. As before `Null` and `Descent` are specializations of `Relaxed`.

For problems of the form (22), Algorithm 2 can provide significantly more efficient updates than Algorithm 1. Despite the relaxations introduced in Algorithm 2 to achieve faster iterates, the following theorem establishes that the procedure is still sound, preserving essentially the same convergence guarantees as the more expensive Algorithm 1.

Theorem 8 *Let $h : \mathbb{R}_+ \to \overline{\mathbb{R}}$ be an increasing convex function, κ be a convex gauge induced by the convex hull of a compact atomic set $\mathcal{A} \cup \{\mathbf{0}\}$, and ℓ be L-smooth and convex. Let $F = \ell + h \circ \kappa$ and assume Algorithm 2 generates a bounded sequence $\{\mathbf{w}_t\}$. Let $\alpha_t > 0$,*

$0 \le \eta_t \le 1$, *and the subroutine* **`Improve`** *be* **`Relaxed`**. *Then for any* $\mathbf{w} \in \operatorname{dom} F$ *and* $t \ge 0$, *we have*

$$F(\mathbf{w}_{t+1}) \le F(\mathbf{w}) + \pi_t(1-\eta_0)\big(F(\mathbf{w}_0) - F(\mathbf{w})\big) \\ + \sum_{s=0}^{t} \frac{\pi_t}{\pi_s}\eta_s^2 \Big(\big(\rho\varepsilon_s + h(\rho/\alpha_s) - h(\rho)\big)/\eta_s + \tfrac{L}{2}\left\|\tfrac{\rho}{\alpha_s}\mathbf{a}_s - \mathbf{w}_s\right\|^2\Big), \tag{37}$$

where $\rho := \kappa(\mathbf{w})$ *and* $\pi_t := \prod_{s=1}^{t}(1-\eta_s)$ *with* $\pi_0 = 1$.

The proof is given in Appendix A.4. From this theorem, we obtain the following rate of convergence by pursuing a similar argument as in the proof of Theorem 5.

Corollary 9 *Under the same setting as in Theorem 8, let* $\eta_t = 2/(t+2)$, $\varepsilon_t \le \delta\eta_t/2$ *for some absolute constant* $\delta > 0$, $\alpha_t = \alpha > 0$, $\rho = \kappa(\mathbf{w})$, *and* $L_F := L\ \sup_t \left\|\frac{\rho}{\alpha}\mathbf{a}_t - \mathbf{w}_t\right\|^2$. *Then for all* $t \ge 1$ *and* $\mathbf{w}$, *we have*

$$F(\mathbf{w}_t) \le F(\mathbf{w}) + \frac{2(\rho\delta + L_F)}{t+3} + h(\rho/\alpha) - h(\rho). \tag{38}$$

Moreover, if $\ell \ge 0$ *and* h *is the identity map, then*

$$F(\mathbf{w}_t) \le \frac{1}{\alpha}F(\mathbf{w}) + \frac{2(\rho\delta + L_F)}{t+3}. \tag{39}$$

Some remarks concerning this algorithm and its convergence properties are in order.

Remark 10 *When* $\alpha_t = 1$ *for all* t, *and* h *is the indicator* $\iota_{[0,\zeta]}$ *for some* $\zeta > 0$, *Theorem 9 implies the same convergence rate for the constrained problem* (21), *which immediately recovers (some of) the more specialized results discussed in (Clarkson, 2010; Hazan, 2008; Jaggi, 2013; Shalev-Shwartz et al., 2010; Tewari et al., 2011; Yuan and Yan, 2013). For the special case where* h *is the identity map, a similar* $O(1/t)$ *rate achieved by a similar algorithm appeared in Harchaoui et al. (2015), whose workshop version appeared around the same time as our conference version (Zhang et al., 2012). Harchaoui et al. (2015) considered the slightly more restrictive setting where the convex gauge* κ *is the restriction of a norm into a convex cone and they did not consider (multiplicative) approximate polar computations. Their algorithm also needs to evaluate* κ *explicitly, while the key advantage of our method is to avoid this expensive computation by an upper bound. On the other hand, Harchaoui et al. (2015) investigated the memory based acceleration in (Holloway, 1974; Meyer, 1974) while we develop in the next section a fixed-rank local improvement step.*

Remark 11 *The additional factor* ρ *in the above bounds is necessary because the approximation in* (30) *is not invariant to scaling, requiring some compensation. Note also that* $\alpha \le 1$, *since the right-hand side of* (30) *must become negative as* $\varepsilon_t \to 0$. *The result in* (39) *states roughly that an* α*-approximate subroutine (for computing the polar of* κ*) leads to an* α*-approximate "minimum", also at the rate of* $O(1/t)$. *This observation was independently made in (Bach, 2013), where the multiplicative factor* ρ *is introduced to solve approximately some matrix factorization problems. In a similar vein, Cheng et al. (2016) leveraged the multiplicative guarantee to solve some tensor problems.*

Remark 12 *The assumption that $\{\mathbf{w}_t\}$ is bounded can be satisfied easily, which in turn ensures $L_F < \infty$. The simplest way is to postulate that the sublevel set $\{\mathbf{w} \in \operatorname{dom} F : F(\mathbf{w}) \le \ell(\mathbf{w}_0) + h(\rho_0)\}$ is bounded. In this case $\{\mathbf{w}_t\}$ is trivially bounded if `Monotone` is used. When `Relaxed` is used instead (which includes `Null` and `Descent`), we can strengthen it into `Monotone` via the roll-back trick in Theorem 7. At first sight, the algorithm appears to be stuck, but in fact the diminishing step size η_t will eventually lead to progresses as guaranteed by Theorem 9. Computationally, keeping track of $\ell(\mathbf{w}_t)+h(\rho_t)$ does not incur any overhead. We will provide concrete values/bounds of L_F for the case studies in Section 4.2 and Section 4.4.*

Remark 13 *The convergence results in Theorem 8 and Theorem 9 obviously hold when `Null` or `Descent` is employed for `Improve`, because they are both specializations of `Relaxed`. The update for θ in (34) requires knowledge of the Lipschitz constant L and the norm $\|\cdot\|$. If the norm $\|\cdot\|$ is Hilbertian and h is the identity map, we have the formula*

$$\theta_t = \tfrac{1}{L\|\mathbf{a}_t\|^2} \max\left\{\langle \mathbf{a}_t, L\eta_t \mathbf{w}_t - \nabla\ell(\mathbf{w}_t)\rangle - 1, 0\right\}.$$

However, it is also easy to devise other scaling and step size updates that do not require the knowledge of L or the norm $\|\cdot\|$, such as

$$(\eta_t^*, \theta_t^*) \in \arg\min_{1\ge\eta\ge0, \theta\ge0} \ell((1-\eta)\mathbf{w}_t + \theta\mathbf{a}_t) + (1-\eta)h(\rho_t) + \eta h(\theta/\eta), \tag{40}$$

followed by the `Null` step. Note that (40) is jointly convex in η and θ, since $\eta h(\theta/\eta)$ is a perspective function. Evidently this update can be treated as a `Relaxed` subroutine, since for any (η_t, θ_t) chosen in step 4 of Algorithm 2, it follows that

$$\begin{aligned}
\ell(\mathbf{w}_{t+1}) + h(\rho_{t+1}) &\le \ell((1-\eta_t^*)\mathbf{w}_t + \theta_t^*\mathbf{a}_t) + (1-\eta_t^*)h(\rho_t) + \eta_t^* h(\theta_t^*/\eta_t^*) \\
&\le \ell((1-\eta_t)\mathbf{w}_t + \theta_t\mathbf{a}_t) + (1-\eta_t)h(\rho_t) + \eta_t h(\theta_t/\eta_t) \quad \text{(by (40))} \\
&\le \tilde{\ell}_L(\tilde{\mathbf{w}}_{t+1}; \mathbf{w}_t) + (1-\eta_t)h(\rho_t) + \eta_t h(\theta_t/\eta_t).
\end{aligned}$$

The upper bound $\rho_{t+1} \ge \kappa(\mathbf{w}_{t+1})$ can be verified in the same way as for (35); hence the update (40) enjoys the same convergence guarantee as in Theorem 9.

Remark 14 *An alternative workaround for solving (7) with gauge regularization would be to impose an upper bound $\zeta \ge \kappa(\mathbf{w})$ and consider $\min_{\mathbf{w}:\kappa(\mathbf{w})\le\zeta} \{\ell(\mathbf{w}) + \kappa(\mathbf{w})\}$, while dynamically adjusting ζ. A standard GCG approach, similar to (Harchaoui et al., 2015), would then require solving*

$$\mathbf{d}_t \in \arg\min_{\mathbf{d}:\kappa(\mathbf{d})\le\zeta} \{\langle \mathbf{d}, \nabla\ell(\mathbf{w}_t)\rangle + \kappa(\mathbf{d})\} \tag{41}$$

as a subproblem, which can be as easy to solve as (30). If, however, one wished to include a local improvement step `Improve` (which is essential in practice), it is not clear how to efficiently maintain the constraint $\kappa(\mathbf{w}) \le \zeta$. Moreover, solving (41) up to some multiplicative *factor might* not *be as easy as (30).*

3.3 Additional Discussions and Related Work

In this section we discuss some further properties and extensions of the GCG algorithm.

Smoothing: The convergence rates we have established require the convex loss ℓ to be L-smooth. Although this holds for a variety of losses such as the square and logistic loss, it does not hold for the hinge loss used in support vector machines. However, one can always approximate a nonsmooth (Lipschitz) loss by a smooth function (Nesterov, 2005) before applying GCG, at the expense of getting a slower $O(1/\sqrt{t})$ rate of convergence.

Totally Corrective: As given, Algorithm 2 adds a single atom in each iteration, with a conic combination between the new atom and the previous aggregate. We will refer to it as *vanilla* GCG if `Improve` is further set to `Null`. If $f = \iota_C$ for some compact and convex set C, we simply call it *vanilla* CG for disambiguation. An even more aggressive scheme is to re-optimize the coefficients of all (or a subset of) existing atoms in each iteration. Such a procedure was first studied in (Meyer, 1974; Holloway, 1974), which is often known as the "totally (or fully) corrective update" in the boosting literature and as the "restricted simplicial decomposition" in optimization. More generally, for the convex gauge κ, each iteration of a totally corrective procedure would involve solving

$$\min_{\boldsymbol{\sigma} \geq 0} \ell\left(\sum_{\tau \in A_t} \sigma_\tau \mathbf{b}_\tau\right) + \sum_{\tau \in A_t} \sigma_\tau, \tag{42}$$

where A_t is a set of atoms accumulated up to the t-th iteration. When $f = \iota_C$, the totally corrective CG solves $\min_{\boldsymbol{\sigma}} \ell\left(\sum_{\tau \in A_t} \sigma_\tau \mathbf{b}_\tau\right)$ over $\boldsymbol{\sigma} \geq 0$ and $\mathbf{1}'\boldsymbol{\sigma} = 1$. As pointed out by Holloway (1974), as long as $\{\mathbf{w}_t, \mathbf{a}_t\} \subseteq A_t$, the totally corrective CG will converge at least as fast as the vanilla CG. Much faster convergence is usually observed in practice, although this advantage must be countered by the extra effort spent in solving (42) and alike, which itself need not be trivial. In a finite dimensional setting, provided that the atoms are linearly independent and some restricted strong convexity is present, it is possible to prove that the totally corrective CG converges at a linear rate; see (Shalev-Shwartz et al., 2010; Yuan and Yan, 2013). Recent work of Lacoste-Julien and Jaggi (2015) has further relaxed the requirement of linear independence by making a weaker assumption that the number of atoms is finite (i.e. the feasible region is a polytope).

GCG v.s. PG: The convergence rate established for GCG is on par with PG. When $f = \iota_C$, Canon and Cullum (1968) showed that the rate of *vanilla* CG cannot be improved in general even when ℓ is strongly convex.[1] In this sense, *vanilla* GCG (which subsumes *vanilla* CG) is slower than an "optimal" algorithm like APG. GCG's advantage, however, is that it only needs to solve a linear subproblem (13) in each iteration (i.e., computing the polar κ°), whereas APG (or PG) needs to compute the proximal update, which is a quadratic problem for the least square prox-function. The two approaches are complementary in the sense that the polar can be easier to compute in some cases, whereas in other cases the proximal update can be computed analytically. An additional advantage GCG has over APG, however, is its greedy sparse nature: Each iteration of GCG amounts to a single atom, hence the number of atoms never exceeds the total number of iterations for GCG. By contrast, APG can produce a dense update in each iteration, although in later stages the

1. Improvements in specific cases are still possible; see the next paragraph. Indeed the counter-example of Canon and Cullum (1968) only applies when the solution lies at the boundary of a polyhedral set.

estimates may become sparser due to the shrinkage effect of the proximal update. Moreover, GCG is "robust" with respect to α-approximate atom selection (cf. (39) in Theorem 9), whereas similar results do not appear to be available for PG or APG when the proximal update is computed up to a multiplicative factor.

Faster Rates: Faster rates of convergence for CG (with a closed-loop step size rule), under the restriction $f = \iota_C$, have been considered by a number of works. Levitin and Polyak (1966) first showed that if C is strongly convex and the gradient $\nabla \ell$ is bounded away from $\mathbf{0}$, then a linear rate of convergence can be achieved by CG. Guélat and Marcotte (1986) showed that if there is a minimizer in the interior of C, then again a linear rate of convergence can be obtained. Wolfe (1970) proposed an away step modification of CG, and a variant of it designed by Guélat and Marcotte (1986) attained linear rate of convergence when C is polyhedral. The analysis was further improved into a global linear rate in (Lacoste-Julien and Jaggi, 2015; Beck and Shtern, 2015). Dunn (1979) proved $o(1/t)$ rate, linear rate and finite convergence for CG, depending on how "continuous" the polar operation is. Lastly, Garber and Hazan (2015) showed that for polytope C, a modification of CG achieves the linear rate if ℓ is of quadratic growth (in particular, strongly convex), and Garber and Hazan (2016) proved the $O(1/t^2)$ rate if ℓ is of quadratic growth and C is strongly convex. Note that these faster rates are possible by putting restrictive assumptions on the constraint set C (polyhedral or strongly convex). Interestingly, results on faster rates for *general convex* functions f are scarce, although establishing it for GCG is conceivably not hard. For example, in (Zhang et al., 2012) we proved a linear rate for the totally corrective GCG when ℓ is strongly convex (in a slightly stronger sense), f is a convex gauge, and $\mathcal{B} = \mathbb{R}^n$.

4. Application to Low Rank Learning

As a first application of GCG, we consider the problem of optimizing low rank matrices; a problem that forms the cornerstone of many machine learning tasks such as dictionary learning and matrix completion. Since imposing a bound on matrix rank generally leads to an intractable problem, much recent work has investigated approaches that relax the hard rank constraint with carefully designed regularization penalties that indirectly encourage low rank structure (Bach et al., 2008; Bradley and Bagnell, 2009). In Section 4.2 we first formulate a generic convex relaxation scheme for low rank problems arising in matrix approximation tasks, including dictionary learning and matrix completion. Conveniently, the modified GCG algorithm developed in Section 3.2, Algorithm 2, provides a general solution method for the resulting problems. Next, in Section 4.3 we show how the local improvement component of the modified GCG algorithm can be provided in this case by a procedure that optimizes an unconstrained surrogate (instead of a constrained problem) to obtain greater efficiency. This modification significantly improves the practical efficiency of GCG without affecting its convergence properties. We illustrate these contributions by highlighting the example of matrix completion under trace norm regularization, where the improved GCG algorithm demonstrates state of the art performance. Finally, we demonstrate how the convex relaxation can be generalized to handle latent multi-view models (White et al., 2012) in Section 4.4, showing how the improved GCG algorithm can still be applied once the major challenge of efficiently computing the polar κ° has been solved.

4.1 Low Rank Learning Problems

Many machine learning tasks, such as dictionary learning and matrix completion, can be formulated as seeking low rank matrix approximations of a given data matrix. In these problems, one is presented with an $n \times m$ matrix X (perhaps only partially observed) where each column corresponds to a training example and each row corresponds to a feature across examples; the goal is to recover a low rank matrix W that approximates (the observed entries of) X. To achieve this, we assume we are given a convex loss function $\ell(W) := L(W, X)$ that measures how well W approximates X.[2] If we impose a hard bound, t, on the rank of W, the core training problem can then be expressed as

$$\inf_{W:\operatorname{rank}(W)\leq t} \ell(W) = \inf_{U\in\mathbb{R}^{n\times t}} \inf_{V\in\mathbb{R}^{t\times m}} \ell(UV). \tag{43}$$

Unfortunately, this problem is not convex, and except for special cases (such as SVD) is believed to be intractable.

The two forms of this problem that we will explicitly illustrate in this paper are dictionary learning and matrix completion. First, for dictionary learning, the $n \times m$ data matrix X is fully observed, the $n \times t$ matrix U is interpreted as a "dictionary" consisting of t basis vectors, and the $t \times m$ matrix V is interpreted as the "coefficients" consisting of m code vectors. For this problem, the dictionary and coefficient matrices need to be inferred *simultaneously*, in contrast to traditional signal approximation schemes where the dictionary (e.g., the Fourier or a wavelet basis) is fixed *a priori*. Unfortunately, learning the dictionary with the coefficients creates significant computational challenge since the formulation is no longer *jointly* convex in the variables U and V, even when the loss ℓ is convex. Indeed, for a fixed dictionary size t, the problem is known to be NP-hard (Vavasis, 2010) except for very special cases (Yu and Schuurmans, 2011).

The second learning problem we illustrate is low rank matrix completion. Here, one observes a small number of entries in $X \in \mathbb{R}^{n\times m}$ and the task is to infer the remaining entries. In this case, the optimization objective can be expressed as $\ell(W) := L(\mathcal{P}(W - X))$, where L is a reconstruction loss (such as Frobenius norm squared) and $\mathcal{P} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ is a "mask" operator that simply fills the entries corresponding to unobserved indices in X with zero. Various recommendation problems, such as the Netflix prize,[3] can be cast as matrix completion. Due to the ill-posed nature of the problem it is standard to assume that the matrix X can be well approximated by a low rank reconstruction W. Unfortunately, the rank function is not convex, hence hard to minimize in general. Conveniently, both the dictionary learning and matrix completion problems are subsumed by the general formulation we consider in (43).

4.2 General Convex Relaxation and Solution via GCG

To develop a convex relaxation of (43), we first consider the factored formulation expressed in terms of U and V on the right hand side. Although it only involves an unconstrained minimization which can be convenient, care is needed to handle the scale invariance introduced by the factored form: since U can always be scaled and V counter-scaled to preserve

2. Note that the matrix W need not reconstruct the entries of X directly: a nonlinear transfer could be interposed while maintaining convexity of ℓ; for example, by using an appropriate Bregman divergence.
3. `http://www.netflixprize.com//community/viewtopic.php?id=1537`

the product UV, some form of penalty or constraint is required to restore well-posedness. A standard strategy is to constrain U, for which a particularly common choice is to constrain its columns to have unit length; i.e. $\|U_{:i}\|_c \le 1$ for all i, where $U_{:i}$ stands for the i-th column of U, and c simply denotes some norm on the columns. In dictionary learning, for example, this amounts to constraining the basis vectors in the dictionary U to lie within the unit ball of $\|\cdot\|_c$. Similar normalization can be used to represent the rows $V_{i:}$, with the magnitude encoded by $\sigma_i \ge 0$. In consequence, the unconstrained optimization over $\ell(UV)$ in (43) can be reformulated as

$$\min_{U\in\mathbb{R}^{n\times t}, V\in\mathbb{R}^{t\times m}, \boldsymbol{\sigma}\in\mathbb{R}^t} \ell(U\,\mathrm{diag}(\boldsymbol{\sigma})V), \quad s.t. \quad \|U_{:i}\|_c \le 1,\ \|V_{i:}\|_r \le 1,\ \text{and } \sigma_i \ge 0. \tag{44}$$

Here $\boldsymbol{\sigma} := (\sigma_1, \sigma_2, \ldots, \sigma_t)^\top$, and $\mathrm{diag}(\boldsymbol{\sigma})$ is a diagonal matrix with the i-th diagonal entry being σ_i. The specific form of the row norm $\|\cdot\|_r$ provides additional flexibility in promoting different structures; for example, the $\mathfrak{l}_1$ norm leads to sparse solutions, the $\mathfrak{l}_2$ norm yields low rank solutions, and block structured norms generate group sparsity. The specific form of the column norm $\|\cdot\|_c$ also has an effect, as discussed in Section 4.4 below.

Of course the problem remains non-convex. However, it has recently been observed that a simple relaxation of the rank constraint in (43) allows a convex reformulation to be achieved (Argyriou et al., 2008; Bach et al., 2008; Bradley and Bagnell, 2009; Zhang et al., 2011). Observe that replacing the rank constraint with a regularization penalty on the magnitude σ_i of each basis pair $(U_{:i}, V_{i:})$ yields a relaxed form of (43):

$$\inf_{U:\|U_{:i}\|_c\le 1}\ \inf_{V:\|V_{i:}\|_r\le 1}\ \inf_{\boldsymbol{\sigma}:\sigma_i\ge 0} \ell(U\,\mathrm{diag}(\boldsymbol{\sigma})V) + \lambda\|\boldsymbol{\sigma}\|_1, \tag{45}$$

where the number of columns in U (and also the number of rows in V) is *not* restricted. Similar to Lasso, the $\mathfrak{l}_1$ norm on the "singular values" (combination weights) $\boldsymbol{\sigma}$ is used as a penalty that serves as a convex proxy for rank. By the shrinkage effect of the $\mathfrak{l}_1$ norm, many σ_i will become zero, implying that the corresponding columns of U (and rows of V) can be dropped. As a result, the rank of the solution—which is indeed unknown *a priori*—can be chosen adaptively with the trade-off parameter $\lambda \ge 0$.

Importantly, although the problem (45) is still not jointly convex in the factors U, V and $\boldsymbol{\sigma}$, it can be exactly reformulated as a convex optimization as long as ℓ is convex:

$$(45) = \min_W\ \ell(W) + \lambda\ \inf\left\{\sum_i \sigma_i : \boldsymbol{\sigma} \ge 0, W = \sum_i \sigma_i U_{:i}V_{i:}, \|U_{:i}\|_c \le 1, \|V_{i:}\|_r \le 1\right\} \tag{46}$$

$$= \min_W\ \ell(W) + \lambda\,\kappa(W), \tag{47}$$

where the gauge κ is defined by the following set $\mathcal{A}$ with uncountably many elements

$$\mathcal{A} := \{\mathbf{u}\mathbf{v}^\top : \mathbf{u}\in\mathbb{R}^n, \mathbf{v}\in\mathbb{R}^m, \|\mathbf{u}\|_c \le 1, \|\mathbf{v}\|_r \le 1\}. \tag{48}$$

The resulting convex formulation (47) has been observed, in various forms, by, e.g., Argyriou et al. (2008); Bach et al. (2008); Zhang et al. (2011); White et al. (2012). However our derivation, first presented in (Zhang et al., 2012), provides a much simpler and flexible view based on convex gauges, which also establishes a direct connection to the GCG method outlined in Section 3.2, yielding an efficient solution method with no additional effort. The reformulation (47) reveals that the relaxed learning problem (45) is essentially considering a rank-one decomposition of W, which generalizes the singular value decomposition. Furthermore it provides a clearer understanding of the computational properties of the regularizer in (47) via its polar:

$$\kappa^\circ(G) = \sup_{A\in\mathcal{A}} \langle A, G\rangle = \sup_{\|\mathbf{u}\|_c\le 1, \|\mathbf{v}\|_r\le 1} \mathbf{u}^\top G\mathbf{v} \tag{49}$$

$$= \sup_{\|\mathbf{v}\|_r\le 1} \|G\mathbf{v}\|_c^\circ = \sup_{\|\mathbf{u}\|_c\le 1} \left\|G^\top \mathbf{u}\right\|_r^\circ. \tag{50}$$

Recall that for gauge regularized problems in the form (47), Algorithm 2 only requires approximate computation of the polar (30) in Line 3; hence an efficient method for (approximately) solving (49) immediately yields an efficient implementation. However, this simplicity must be countered by the fact that the polar is not always tractable, since it involves *maximizing* a norm subject to a different norm constraint.[4] Nevertheless, there exist important cases where the polar can be efficiently evaluated, which we illustrate below.

The polar also allows one to gain insight into the structure of the original gauge regularizer, since by duality $\kappa = (\kappa^\circ)^\circ$ yields an explicit formula for κ. For example, consider the important special case where $\|\cdot\|_r = \|\cdot\|_c = \|\cdot\|_2$. In this case, using (49) one can infer that the polar $\kappa^\circ(\cdot) = \|\cdot\|_{\text{sp}}$ is the spectral norm, and therefore the corresponding gauge regularizer $\kappa(\cdot) = \|\cdot\|_{\text{tr}}$ is the trace norm. In this way, the trace norm regularizer, which is known to provide a convex relaxation of the matrix rank (Chandrasekaran et al., 2012), can be viewed from the perspective of dictionary learning as an *induced* norm that arises from a simple 2-norm constraint on the magnitude of the dictionary entries $U_{:i}$ and a simple 2-norm penalty on the magnitudes of the code vectors $V_{i:}$.

This convex relaxation strategy based on convex gauges is quite flexible and has been recently studied for some structured sparse problems (Chandrasekaran et al., 2012; Tewari et al., 2011; Zhang et al., 2012; Bach, 2013). Importantly, the generality of the column and row norms in (45) proffer considerable flexibility to finely characterize the structures sought by low rank learning methods. We will exploit this flexibility in Section 4.4 below to achieve a novel extension of this framework to multi-view dictionary learning (White et al., 2012).

4.2.1 Application to Matrix Completion

We end this subsection with a brief illustration of how the framework can be applied to matrix completion. Recall that in this case one is given a partially observed data matrix X, and the goal is to find a low rank approximator W that minimizes the reconstruction error on observed entries. In particular, consider the following standard specialization of (47):

$$\min_{W\in\mathbb{R}^{n\times m}} \frac{1}{2\|\mathcal{P}\|_0} \|\mathcal{P}(X-W)\|_{\text{F}}^2 + \lambda \|W\|_{\text{tr}}, \tag{51}$$

where, as noted above, $\mathcal{P} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ is the mask operator that fills entries where X is unobserved with zero. $\|\mathcal{P}\|_0$ is the number of observed entries. Surprisingly, Candès and

4. In this regard, the α-approximate polar oracle introduced in (30) and Theorem 9 might be useful, although we shall not develop this idea further here. See the work of Bach (2013) for some applications of this idea to matrix factorization and the work of Cheng et al. (2016) to tensor problems.

Recht (2009) have proved that, assuming X is (approximately) low-rank, the solution of the convex relaxation (51) will recover the true matrix X with high probability, even when only a small *random* portion of X is observed.

Note that, since (51) is a specialization of (47), the modified GCG algorithm, Algorithm 2, can be immediately applied. Interestingly, such an approach contrasts with the most popular algorithm favored in the current literature: PG (or its accelerated variant APG; cf. Section 2). The two strategies, GCG versus PG, exhibit an interesting trade-off. Each iteration of PG (or APG) involves solving the PU (9), which in this case requires a *full* singular value decomposition (SVD) of the current approximant W. The modified GCG method, Algorithm 2, by contrast, only requires the polar of the gradient matrix $\nabla\ell(W)$ to be (approximately) computed in each iteration (i.e. dual of the trace norm; namely, the spectral norm), which is an order of magnitude cheaper to evaluate than a full SVD (cubic versus squared worst case time).[5] On the other hand, GCG has a slower theoretical rate than APG, $O(1/t)$ versus $O(1/t^2)$, in terms of the number of iterations required to achieve a given accuracy. Once the enhancements in the next section have been incorporated, our experiments in Section 5.1 will demonstrate that GCG is far more efficient than APG for solving the matrix completion problem (51) to tolerances used in practice.

Given the specific objective (51), we may derive a concrete bound on L_F used in Theorem 9. Obviously, the smoothness constant is $L = 1/\left\|\mathcal{P}\right\|_0$. Suppose we use `Monotone` with $W_0 = \mathbf{0}$. So all W_t (and the optimal W) need to satisfy $\lambda \left\|W_t\right\|_{\mathrm{tr}} \leq C$, where $C = \left\|\mathcal{P}X\right\|_{\mathrm{F}}^2 / (2\left\|\mathcal{P}\right\|_0)$. Due to the normalization, C can be considered a universal constant. All matrices $A \in \mathcal{A}$ in (48) satisfy $\left\|A\right\|_{\mathrm{F}} \leq 1$. Obviously $\frac{1}{n}\left\|W_t\right\|_{\mathrm{tr}}^2 \leq \left\|W_t\right\|_{\mathrm{F}}^2 \leq \left\|W_t\right\|_{\mathrm{tr}}^2$. So

$$L_F \leq 2L \sup_t \left\{ \frac{\rho^2}{\alpha^2}\left\|A_t\right\|_{\mathrm{F}}^2 + \left\|W_t\right\|_{\mathrm{F}}^2 \right\} \leq 2L \sup_t \left\{ \frac{\rho^2}{\alpha^2} + \left\|W_t\right\|_{\mathrm{tr}}^2 \right\} \leq \frac{2}{\left\|\mathcal{P}\right\|_0}\left(\frac{1}{\alpha^2}+1\right)\frac{C^2}{\lambda^2}. \quad (52)$$

If λ needs to be less than $\frac{1}{\left\|\mathcal{P}\right\|_0}$, then the upper bound on L_F will be $O(\left\|\mathcal{P}\right\|_0)$. Notice that under `Monotone`, the expression of L_F in Theorem 5 is on par with the condition number $L_f D_*^2$ in (Harchaoui et al., 2015, Theorem 3).

4.3 Fixed-rank Local Optimization

Although the modified GCG algorithm, Algorithm 2, can be readily applied to the reformulated low rank learning problem (47), the sublinear rate of convergence established in Theorem 9 is still too slow in practice. Nevertheless, an effective local improvement scheme can be developed that satisfies the properties of a `Relaxed` subroutine in Algorithm 2, which allows empirical convergence to be greatly accelerated. Recall that in Algorithm 2, the intermediate iterate $\tilde{W}_{t+1}$ is determined by a linear combination of the previous iterate W_t and the newly added atom A_t (we changed the notation from $\tilde{\mathbf{w}}_{t+1}$ to $\tilde{W}_{t+1}$ since

5. The per-iteration cost of computing the spectral norm can be further reduced for the sparse matrices occuring naturally in matrix completion, since the gradient matrix in (51) contains at most k nonzero elements for k equal to the number of observed entries in X. In such cases, an ϵ accurate estimate of the squared spectral norm can be obtained in $O\left(\frac{k\log(n+m)}{\sqrt{\epsilon}}\right)$ time (Jaggi, 2013; Kuczyński and Woźniakowski, 1992), offering a significant speed up over the general case. Although some speed ups might also be possible when computing the SVD for sparse matrices, the gap between the computational cost for evaluating the spectral norm versus the trace norm for sparse matrices remains significant.

the optimization variables are now matrices). We will next demonstrate that $\tilde{W}_{t+1}$ can be further improved by solving a reformulated local problem, and then show how this can still preserve the convergence guarantees of Algorithm 2.

The key reformulation relies on the following fact.

Proposition 15 *The convex gauge κ induced by the set $\mathcal{A}$ in* (48) *can be re-expressed as*

$$\kappa(W) = \min\left\{\sum_{i=1}^{t} \|U_{:i}\|_c \ \|V_{i:}\|_r : UV = W\right\} = \min\left\{\frac{1}{2}\sum_{i=1}^{t}\left(\|U_{:i}\|_c^2 + \|V_{i:}\|_r^2\right) : UV = W\right\},$$

for $U \in \mathbb{R}^{n\times t}$ and $V \in \mathbb{R}^{t\times m}$, as long as $t \geq mn$.

In Appendix B.1 we provide a direct proof that allows us to bound the number of terms involved in the summation. The above lower bound mn on t may not be sharp. For example, if $\|\cdot\|_r = \|\cdot\|_c = \|\cdot\|_2$, then κ is the trace norm, where $\sum_{i=1}^{t}(\|U_{:i}\|_c^2 + \|V_{i:}\|_r^2)$ is simply $\|U\|_{\mathrm{F}}^2 + \|V\|_{\mathrm{F}}^2$, the sum of the squared Frobenius norms; that is, Proposition 15 yields the well-known variational form of the trace norm in this case (Srebro et al., 2005). Observe that t in this case can be as low as the rank of W. Theorem 15 also appeared in (Bach et al., 2008; Bach, 2013), although without the explicit bound $t \geq mn$.

Based on Theorem 15, the objective in (47) can then be *approximated* at iteration t as

$$\mathcal{F}_{t+1}(U, V) := \ell(UV) + \frac{\lambda}{2}\sum_{i=1}^{t+1}\left(\|U_{:i}\|_c^2 + \|V_{i:}\|_r^2\right). \tag{53}$$

Our key intuition is to first factorize $\tilde{W}_{t+1}$ into $U_{\text{init}}V_{\text{init}}$, and then find improvement to (U, V) with respect to $\mathcal{F}_{t+1}$ (*not* the original F). This results in U_{t+1} and V_{t+1} satisfying $\mathcal{F}_{t+1}(U_{t+1}, V_{t+1}) \leq \mathcal{F}_{t+1}(U_{\text{init}}, V_{\text{init}})$. Finally we restore the iterates based on Theorem 15

$$W_{t+1} = U_{t+1}V_{t+1} \qquad \text{and} \qquad \rho_{t+1} = \frac{1}{2}\sum_{i=1}^{t+1}\left(\|(U_{t+1})_{:i}\|_c^2 + \|(V_{t+1})_{i:}\|_r^2\right). \tag{54}$$

Using $\mathcal{F}_{t+1}$ as a surrogate objective for local improvement is particularly advantageous in computation, because the problem is now unconstrained and the gauge function has been replaced by much simpler forms such as $\|U_{:i}\|_c^2$ and $\|V_{i:}\|_r^2$. This allows efficient unconstrained minimization routines (e.g. L-BFGS) to be applied directly, improving the quality of the current iterate *without* increasing the size of the dictionary. Note however that (53) is not *jointly* convex in (U, V) since the size $t + 1$ (the iteration counter) is fixed. When t is sufficiently large, any local minimizer of (53) *globally* solves the original problem (47) (Burer and Monteiro, 2005)[6], but we locally reduce (53) in each iteration even for small t.

Once this modification is added to Algorithm 2 we obtain the final variant, Algorithm 3. In this algorithm, Line 3 remains equivalent to before, and Line 4 adopts the slightly more sophisticated update rule in (40), although one can also use (34) with $\eta_t = \frac{2}{t+2}$. Lines 5 to 8 *collectively* implement the `Improve` subroutine: Line 5 splits the iterate into two parts

6. The catch here is that seeking a local minimizer can still be highly non-trivial, sometimes it is even hard to certify the local optimality.

Algorithm 3 GCG variant for low rank learning.

```
Require: The atomic set $\mathcal{A}$.
1: Initialize $W_0 = \mathbf{0}$, $\rho_0 = 0$, $U_0 = V_0 = \Lambda_0 = \emptyset$.
2: for $t = 0, 1, \ldots$ do
3:     $(\mathbf{u}_t, \mathbf{v}_t) \leftarrow \arg\min_{\mathbf{u}\mathbf{v}^\top \in \mathcal{A}} \left\langle \nabla \ell(W_t), \mathbf{u}\mathbf{v}^\top \right\rangle$
4:     $(\eta_t, \theta_t) \leftarrow \arg\min_{0 \le \eta \le 1, \theta \ge 0} \ell((1-\eta)W_t + \theta\, \mathbf{u}_t \mathbf{v}_t^\top) + \lambda((1-\eta)\rho_t + \theta)$
5:     $U_{\text{init}} \leftarrow (\sqrt{1-\eta_t} U_t, \sqrt{\theta_t}\mathbf{u}_t)$, $V_{\text{init}} \leftarrow (\sqrt{1-\eta_t} V_t^\top, \sqrt{\theta_t}\mathbf{v}_t)^\top$
6:     $(U_{t+1}, V_{t+1}) \leftarrow \texttt{Reduce}(\mathcal{F}_{t+1}, U_{\text{init}}, V_{\text{init}})$ s.t. $\mathcal{F}_{t+1}(U_{t+1}, V_{t+1}) \le \mathcal{F}_{t+1}(U_{\text{init}}, V_{\text{init}})$
       e.g. find a local minimum of $\mathcal{F}_{t+1}$ with $U$ and $V$ initialized to $U_{\text{init}}$ and $V_{\text{init}}$ resp.
7:     $W_{t+1} \leftarrow U_{t+1} V_{t+1}$        ▷ Nominal as we only maintain $(U_t, V_t)$ in practice
8:     $\rho_{t+1} \leftarrow \frac{1}{2} \sum_{i=1}^{t+1} (\|(U_{t+1})_{:i}\|_c^2 + \|(V_{t+1})_{i:}\|_r^2)$
9: end for
```

U_{init} and V_{init}, while Line 6 further reduces the reformulated local objective (53) with the designated initialization (e.g. by finding a local minimum). We referred to it as `Reduce` in order to be distinct from `Improve`. The last two steps restore the GCG iterate by (54).

The only major challenge left for inserting a local optimization step in Algorithm 2 is that the `Relaxed` condition (36) in Theorem 7 must be maintained. This is the condition under which Algorithm 3 retains the $O(1/t)$ rate of convergence established in Theorem 9. Fortunately this holds true as stated in Theorem 16 and its proof is given in Appendix B.2.

Proposition 16 *(U_{init}, V_{init}) in Line 5 is a factorization of the intermediate iterate $\tilde{W}_{t+1} = (1-\eta_t)W_t + \theta_t \mathbf{u}_t \mathbf{v}_t^\top$. Let (U_{t+1}, V_{t+1}) improve upon it in the sense that $\mathcal{F}_{t+1}(U_{t+1}, V_{t+1}) \le \mathcal{F}_{t+1}(U_{init}, V_{init})$. Then the W_{t+1} and ρ_{t+1} recovered by* (54) *satisfy the `Relaxed` condition.*

Although the `Reduce` subroutine is not guaranteed to achieve *strict* improvement in every iteration, we observe in our experiments in Section 5 below that Algorithm 3 is almost always significantly faster than the `Null` version of Algorithm 2. In other words, local acceleration appears to make a big difference in practice.

Remark 17 *Interleaving local improvement with a globally convergent procedure has been considered by Mishra et al. (2013) and Laue (2012). In particular, Laue (2012) considered the constrained problem* (21)*, which unfortunately leads to considering a constrained minimization for local improvement that can be less efficient than the unconstrained approach considered in* (53)*. In addition, Laue (2012) used the original objective F directly in local optimization. This leads to significant challenge when the regularizer κ is complicated such as the trace norm, and we avoid this issue by using a smooth surrogate objective $\mathcal{F}_{t+1}$.*

Focusing on trace norm regularization only, Mishra et al. (2013) proposed a trust-region procedure that locally minimizes the original *objective on the Stiefel manifold and the positive semidefinite cone, which unfortunately also requires constrained minimization and dynamic maintenance of the singular value decomposition of a small matrix. The rate of convergence of this procedure has not been analyzed.*

Remark 18 *We remark that local minimization of* (53) *constitutes the primary bottleneck of Algorithm 3, consuming over 50% of the computation. Since* (54) *is the same objective* *considered by the standard fixed rank approach to matrix factorization, the overall performance of Algorithm 3 can be substantially improved by leveraging the techniques that have been developed for this specific objective. These improvements include system level optimizations such as stochastic solvers on distributed, parallel and asynchronous infrastructures (Zhuang et al., 2013; Yun et al., 2014). In the experiments conducted in Section 5 below, we merely solve* (54) *using a generic L-BFGS solver to highlight the efficiency of the GCG framework itself. Nevertheless, such a study leaves significant room for further straightforward acceleration.*

Remark 19 *Much of our development in this section was inspired by the work of Bach et al. (2008), who first demonstrated, although in a slightly less clear way, that the convex relaxation in* (47) *is possible, but did not observe the computational connection to GCG. Our contribution here is to present this result succinctly using the notion of convex gauges, connect it naturally to the GCG algorithm we developed in Section 3, and propose an efficient fixed-rank local improvement subroutine* `Reduce` *that does not affect theoretical convergence guarantees. After our conference version (Zhang et al., 2012), Bach (2013) also independently extended the previous work (Bach et al., 2008) using convex gauges, and discussed a multiplicative approximate GCG variant similar to our Algorithm 2. However, the possibility to intervene fixed-rank local subroutines based on Theorem 15 with GCG, and its practical computational consequences, were not observed in (Bach, 2013).*

4.4 Multi-view Dictionary Learning

Finally, in this section we show how low rank learning can be generalized to a multi-view setting where, once again, the modified GCG algorithm can efficiently provide optimal solutions. The key challenge in this case is to develop column and row norms that can capture the more complex problem structure.

Consider a two-view learning task where one is given m paired observations $\left\{ \begin{bmatrix} \mathbf{x}_j \\ \mathbf{y}_j \end{bmatrix} \right\}$ consisting of an x-view and a y-view with lengths n_1 and n_2 respectively. The goal is to infer a set of latent representations, $\mathbf{h}_j$ (of dimension $t < \min(n_1, n_2)$), and reconstruction models parameterized by the matrices A and B, such that $A\mathbf{h}_j \approx \mathbf{x}_j$ and $B\mathbf{h}_j \approx \mathbf{y}_j$ for all j. In particular, let X denote the $n_1 \times m$ matrix of x-view observations, Y the $n_2 \times m$ matrix of y-view observations, and $Z = \begin{bmatrix} X \\ Y \end{bmatrix}$ the concatenated $(n_1 + n_2) \times m$ matrix. The problem can then be expressed as recovering an $(n_1 + n_2) \times t$ matrix of model parameters, $C = \begin{bmatrix} A \\ B \end{bmatrix}$, and a $t \times m$ matrix of latent representations, H, such that $Z \approx CH$.

The key assumption behind multi-view learning is that each of the two views, $\mathbf{x}_j$ and $\mathbf{y}_j$, is conditionally independent given the shared latent representation, $\mathbf{h}_j$. Although multi-view data can always be concatenated hence treated as a single view, explicitly representing multiple views enables more accurate identification of the latent representation (as we will see), when the conditional independence assumption holds. To better motivate this model, it is enlightening to first reinterpret the classical formulation of multi-view dictionary learning, which is given by *canonical correlation analysis* (CCA). Typically, it is expressed as the problem of projecting two views so that the correlation between them is maximized. Assuming the data is centered (i.e. $X\mathbf{1} = \mathbf{0}$ and $Y\mathbf{1} = \mathbf{0}$), the sample covariance of X and Y is given by $XX^\top/m$ and $YY^\top/m$ respectively. CCA can then be expressed as an optimization over matrix variables

$$\max_{U,V} \operatorname{tr}(U^\top XY^\top V) \qquad \text{s.t.} \qquad U^\top XX^\top U = V^\top YY^\top V = I \tag{55}$$

for $U \in \mathbb{R}^{n_1 \times t}$, $V \in \mathbb{R}^{n_2 \times t}$ (De Bie et al., 2005). Here I is the identity matrix whose size is clear from the context. Although this classical formulation (55) does not make the shared latent representation explicit, CCA can be expressed by a generative model: given a latent representation, $\mathbf{h}_j$, the observations $\mathbf{x}_j = A\mathbf{h}_j + \boldsymbol{\epsilon}_j$ and $\mathbf{y}_j = B\mathbf{h}_j + \boldsymbol{\nu}_j$ are generated by a linear mapping plus independent zero mean Gaussian noise, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \Sigma_x)$, $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \Sigma_y)$ (Bach and Jordan, 2006). Interestingly, one can show that the classical CCA problem (55) is equivalent to the following multi-view dictionary learning problem.

Proposition 20 (White et al. 2012, Proposition 1) *Fix t, let*

$$(A, B, H) = \arg \min_{A:\|A_{:i}\|_2 \le 1} \min_{B:\|B_{:i}\|_2 \le 1} \min_{H} \left\| \begin{bmatrix} (XX^\top)^{-1/2} X \\ (YY^\top)^{-1/2} Y \end{bmatrix} - \begin{bmatrix} A \\ B \end{bmatrix} H \right\|_{\mathrm{F}}^2 . \tag{56}$$

Then $U = (XX^\top)^{-\frac{1}{2}} A$ and $V = (YY^\top)^{-\frac{1}{2}} B$ provide an optimal solution to (55).

Proposition 20 demonstrates how formulation (56) respects the conditional independence of the separate views: given a latent representation $\mathbf{h}_j$, the reconstruction losses on the two views, $\mathbf{x}_j$ and $\mathbf{y}_j$, cannot influence each other, since the reconstruction models A and B are *individually* constrained. By contrast, in single-view dictionary learning (i.e. *principal component analysis*) A and B are concatenated in the larger variable $C = \left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$, where C as a whole is constrained but A and B are not. A and B must then compete against each other to acquire magnitude to explain their respective "views" given $\mathbf{h}_j$ (i.e. conditional independence is not enforced). Such sharing can be detrimental if the two views really are conditionally independent given $\mathbf{h}_j$.

This matrix factorization viewpoint has recently allowed dictionary learning to be extended from the single-view setting to the multi-view setting (Jia et al., 2010). However, despite its elegance, the computational tractability of CCA hinges on its restriction to squared loss under a particular normalization, precluding other important losses. To combat these limitations, we apply the same convex relaxation principle as in Section 4.2. In particular, we reformulate (56) by first incorporating an arbitrary loss function ℓ that is convex in its first argument, and then relaxing the rank constraint by replacing it with the $\mathfrak{l}_1$ norm of the "singular values". This amounts to the following training problem that extends (Jia et al., 2010):

$$\min_{A,B,H,\boldsymbol{\sigma}} \ell\left(\begin{bmatrix} A \\ B \end{bmatrix} \operatorname{diag}(\boldsymbol{\sigma}) H; Z \right) + \lambda \|\boldsymbol{\sigma}\|_1, \quad \text{s.t. } \|H_{i:}\|_2 \le 1, \quad \sigma_i \ge 0, \quad \begin{bmatrix} A_{:i} \\ B_{:i} \end{bmatrix} \in \mathcal{D} \ \forall i,$$
$$\text{where} \quad \mathcal{D} := \left\{ \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} : \|\mathbf{a}\|_2 \le \beta, \ \|\mathbf{b}\|_2 \le \gamma \right\}. \tag{57}$$

The $\mathfrak{l}_2$ norm on the *rows* of H is a regularizer that encourages the rows to become sparse, hence reducing the size of the learned representation (Argyriou et al., 2008). Since the size of the latent representation (the number of rows of H) is not fixed, our results in Section 4.2

immediately allows the problem to be solved globally and efficiently for A, B and H. This considerably improves the previous local solutions in (Jia et al., 2010) which fixed the latent dimension and approached the non-convex problem by alternating block coordinate descent between A, B and $\operatorname{diag}(\boldsymbol{\sigma})H$.

Indeed, (57) fits exactly in the framework of (45), with $\|\cdot\|_r$ being the $\mathfrak{l}_2$ norm and

$$\|\mathbf{c}\|_c = \max\left(\frac{1}{\beta}\|\mathbf{a}\|_2, \frac{1}{\gamma}\|\mathbf{b}\|_2\right) \quad \text{for} \quad \mathbf{c} = \begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}. \tag{58}$$

Next we will show how to efficiently compute the polar operator (50) that underpins Algorithm 3 in this multi-view model.

4.4.1 Efficient Polar Operator

The polar operator (50) for problem (57) turns out to admit an efficient but nontrivial solution. For simplicity, we assume $\beta = \gamma = 1$ while the more general case can be dealt with in exactly the same way. The proof of the following key theorem is in Appendix B.3.

Proposition 21 *Let $I_1 := \operatorname{diag}(\begin{bmatrix}\mathbf{1}\\ \mathbf{0}\end{bmatrix})$ and $I_2 := \operatorname{diag}(\begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix})$ be the identity matrices on the subspaces spanned by x-view and y-view, respectively. For any $\mu > 0$, denote $D_\mu = \sqrt{1+\mu}I_1 + \sqrt{1+1/\mu}I_2$ as a diagonal scaling of the two identity matrices. Then, for the row norm $\|\cdot\|_r = \|\cdot\|_2$ and column norm $\|\cdot\|_c$ defined as in* (58)*, the corresponding polar* (49) *is given by*

$$\kappa^\circ(G) = \min_{\mu \geq 0} \|D_\mu G\|_{\text{sp}}. \tag{59}$$

Despite its variational form, the polar (59) can still be efficiently computed by considering its squared form and expanding:

$$\|D_\mu G\|_{\text{sp}}^2 = \left\|G^\top D_\mu D_\mu G\right\|_{\text{sp}} = \left\|G^\top \left[(1+\mu)I_1 + (1+1/\mu)I_2\right]G\right\|_{\text{sp}}.$$

Clearly, the term inside the spectral norm $\|\cdot\|_{\text{sp}}$ is convex in μ on $\mathbb{R}_+$ while the spectral norm is convex and increasing on the cone of positive semidefinite matrices, hence by the usual rules of convex composition the left hand side above is a *convex* function of μ. Consequently we can use a subgradient algorithm, or even golden section search, to find the optimal μ in (59). Given an optimal μ, an optimal pair of "atoms" $\mathbf{a}$ and $\mathbf{b}$ can be recovered via the KKT conditions; see Appendix B.4 for details. Thus, by using (59) Algorithm 3 is able to solve the resulting optimization problem far more efficiently than the initial procedure offered by White et al. (2012).

Similar to the case of matrix completion for single view in Section 4.2.1, the L_F in Theorem 5 for this multi-view setting can be bounded in exactly the same way as in (52). The only difference is that instead of $\|W\|_{\text{tr}} \geq \|W\|_{\text{F}}$, we now have $\kappa(W) \geq \frac{1}{\sqrt{2}}\|W\|_{\text{F}}$. To see it, just set $\mu = 1$ in (59), and we immediately obtain $\kappa^\circ(G) \leq \sqrt{2}\|G\|_{\text{sp}}$. Dualizing both sides leads to $\kappa(W) \geq \frac{1}{\sqrt{2}}\|W\|_{\text{tr}} \geq \frac{1}{\sqrt{2}}\|W\|_{\text{F}}$. Therefore we can proceed in analogy to (52):

$$L_F \leq 2L \sup_t \left\{\frac{\rho^2}{\alpha^2}\|A_t\|_{\text{F}}^2 + \|W_t\|_{\text{F}}^2\right\} \leq 2L \sup_t \left\{\frac{2\rho^2}{\alpha^2} + 2\kappa(W_t)^2\right\} \leq \frac{4}{\|\mathcal{P}\|_0}\left(\frac{1}{\alpha^2}+1\right)\frac{C^2}{\lambda^2}. \tag{60}$$

4.4.2 A Modified Power Iteration

In practice, one can compute the polar (59) much more efficiently by deploying a heuristic variant of the power iteration method, whose motivation arises from the KKT conditions in the proof of Theorem 21 (see (76) in Appendix B.4). In particular, two normalized vectors $\mathbf{a}$ and $\mathbf{b}$ are optimal for the polar (50) if and only if there exist nonnegative scalars μ_1 and μ_2 such that

$$\begin{bmatrix} \mu_1 \mathbf{a} \\ \mu_2 \mathbf{b} \end{bmatrix} = GG^\top \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \qquad \text{and} \qquad \mu_1 I_1 + \mu_2 I_2 \succeq GG^\top. \tag{61}$$

Here $A \succeq B$ asserts that $A - B$ is positive semidefinite. Therefore, after randomly initializing $\mathbf{a}$ and $\mathbf{b}$ with unit norm, it is natural to extend the power iteration method by repeating the update:

$$\begin{bmatrix} \mathbf{s} \\ \mathbf{t} \end{bmatrix} \leftarrow GG^\top \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \quad \text{then} \quad \begin{bmatrix} \mathbf{a}^{\text{new}} \\ \mathbf{b}^{\text{new}} \end{bmatrix} \leftarrow \begin{bmatrix} \frac{1}{\|\mathbf{s}\|_2}\mathbf{s} \\ \frac{1}{\|\mathbf{t}\|_2}\mathbf{t} \end{bmatrix}.$$

This procedure usually converges to a solution where the Lagrange multipliers μ_1 and μ_2, recovered from (61), satisfy (61) hence confirming their optimality. While we leave the investigation of the theoretical guarantees of this power iteration heuristic for future work, let us mention that it works very well in practice (as in the single view case) and when the occasional failure occurs we revert to (59) and follow the recovery rule detailed in Appendix B.4.

5. Experimental Evaluation

We evaluate the extended GCG approach developed above in three sparse learning problems: matrix completion, multi-class classification, and multi-view dictionary learning. All algorithms were implemented in Matlab unless otherwise noted.[7] All experiments were run on a single core of a cluster housed at National ICT Australia with AMD 6-Core Opteron 4184 (2.8GHz, 3M L2/6M L3 Cache, 95W TDP, 64 GB memory).

5.1 Matrix Completion with Trace Norm Regularization

Our first set of experiments is on matrix completion (51), where the division by $\|\mathcal{P}\|_0$ is omitted to be consistent with existing solvers. Five data sets are used and their statistics are given in Table 1. The data sets `Yahoo252M` (Dror et al., 2012) and `Yahoo60M`[8] provide music ratings, and the data sets `Netflix` (Dror et al., 2012), `MovieLens1M`, and `MovieLens10M`[9] all provide movie ratings. For `MovieLens1M` and `Yahoo60M`, we randomly sampled 75% ratings as training data, using the rest for testing. The other three data sets came with training and testing partition.

We compared the extended GCG approach (Algorithm 3) with three state-of-the-art solvers for trace norm regularized objectives: MMBS[10] (Mishra et al., 2013), DHM (Dudik

7. All code is available for download at `https://www.cs.uic.edu/~zhangx/GCG`.
8. `Yahoo60M` is the R1 data set at `http://webscope.sandbox.yahoo.com/catalog.php?datatype=r`.
9. Both MovieLens1M and MovieLens10M were downloaded from `http://www.grouplens.org/datasets/movielens`.
10. Downloaded from `http://www.montefiore.ulg.ac.be/~mishra/softwares/traceNorm.html`.

data set	n (#user)	m (#product)	#train	#test	λ
`MovieLens1M`	6,040	3,952	750,070	250,024	50
`MovieLens10M`	71,567	65,133	9,301,174	698,599	100
`Netflix`	2,549,429	17,770	99,072,112	1,408,395	1
`Yahoo60M`	98,130	1,738,442	60,021,577	20,872,219	300
`Yahoo252M`	1,000,990	624,961	252,800,275	4,003,960	100

Table 1: Summary of the data sets used in matrix completion experiments. Here n is the total number of users, m is the number of products (movies or music), #train and #test are the number of ratings used for training and testing respectively.

et al., 2012), and JS (Jaggi and Sulovsky, 2010). The local optimization in GCG was performed by L-BFGS with the maximum number of iteration set to 20.[11] Traditional solvers such as proximal methods (Toh and Yun, 2010; Pong et al., 2010) were not included in this comparison because they are much slower.

MMBS is confined to trace norm regularized problems. At each iteration t, they solve

$$\min_{U\in\mathbb{R}^{n\times t},\ B\in\mathbb{R}^{t\times t},\ V\in\mathbb{R}^{t\times m}} \ell(UBV) + \lambda\,\mathrm{tr}(B), \quad \text{s.t. } U^\top U = VV^\top = I, \text{ and } B \succeq \mathbf{0}. \tag{62}$$

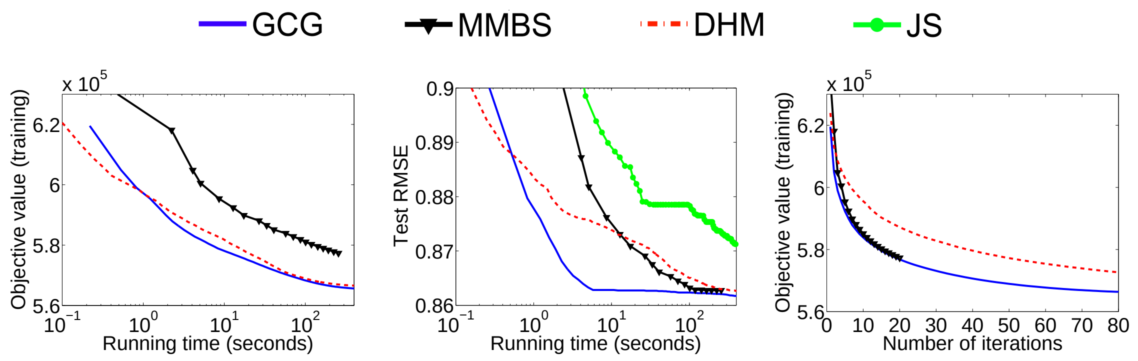
After obtaining a stationary point, the gradient of ℓ is computed, with its top left and right singular vectors appended to U and V respectively, and repeat. Unfortunately, the optimization in (62) is highly challenging on the Stiefel manifold, and their algorithm relies on the directional Hessian which becomes rather complicated in computation for general ℓ. By contrast, the local surrogate (53) used in our approach is substantially easier to optimize.

DHM is a special case of GCG, with `Improve` instantiated with a line search procedure (Dudik et al., 2012). The detail of line search is intricate, but in practice it is slower than totally corrective updates (42) solved by L-BFGS. To further streamline the optimization for (42), we buffered the quantities $\langle \mathcal{P}(\mathbf{u}_t\mathbf{v}_t'), \mathcal{P}(\mathbf{u}_s\mathbf{v}_s')\rangle$ and $\langle \mathcal{P}(\mathbf{u}_t\mathbf{v}_t'), \mathcal{P}(X)\rangle$, where $(\mathbf{u}_t, \mathbf{v}_t)$ is the result of polar operator at iteration t (Line 3 of Algorithm 3). This allowed the objective (42) to be evaluated efficiently without having to compute matrix inner product.

JS was proposed for solving the constrained problem: $\min_W \ell(W)$ s.t. $\|W\|_{\mathrm{tr}} \le \zeta$, which makes it hard to directly compare with solvers for the penalized problem (51). As a workaround, given λ we first found the optimal solution W^* for the penalized problem, and then we set $\zeta = \|W^*\|_{\mathrm{tr}}$. To solve the constrained problem, JS lifts it to an SDP by observing that $\|W\|_{\mathrm{tr}} \le \frac{t}{2}$ iff there exist symmetric matrices A and B such that $Z := \begin{pmatrix} A & W \\ W^\top & B \end{pmatrix} \succeq \mathbf{0}$ and $\mathrm{tr}(A) + \mathrm{tr}(B) = t$. Then the objective $\ell(W)$ can be rewritten in terms of Z, solved by the conventional conditional gradient algorithm for constrained problems.

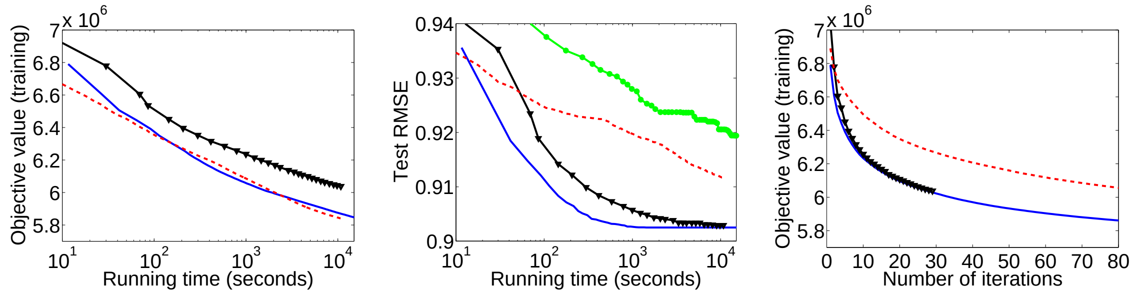
Figures 2 to 6 show how quickly the various algorithms drive the training objective down, while also providing the root mean square error (RMSE) on test data (i.e. comparing the reconstructed matrix each iteration to the ground truth). JS is comparable only on RMSE. The regularization constant λ, given in Table 1, was chosen from

11. Downloaded from `http://www.cs.ubc.ca/~pcarbo/lbfgsb-for-matlab.html`. Recall a suboptimal solution is sufficient for the fixed-rank local optimization in `Relaxed`.



(a) Objective value vs CPU time (b) Test RMSE vs CPU time (c) Objective value vs #iteration

Figure 2: Matrix completion on `MovieLens1M`



(a) Objective value vs CPU time (b) Test RMSE vs CPU time (c) Objective value vs #iteration

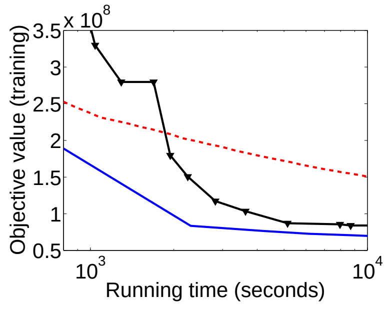
Figure 3: Matrix completion on `MovieLens10M`

$\{1, 10, 50, 100, 200, 300, 400, 500, 1000\}$ to minimize the test RMSE. On the `MovieLens1M` and `MovieLens10M` data sets, GCG and DHM exhibit similar efficiency in reducing the objective value with respect to CPU time (see Figure 2(a) and 3(a)). However, GCG achieves comparable test RMSE an order of magnitude faster than DHM, as shown in Figure 2(b) and 3(b). Interestingly, this discrepancy can be explained by investigating the rank of the solution, i.e. the number of outer iterations taken. In Figure 2(c) and 3(c), DHM clearly converges much more slowly in terms of the number of iterations, which is expected because it does not re-optimize the basis at each iteration, unlike GCG and MMBS. Therefore, although the overall computational cost for DHM appears on par with that for GCG, its solution has a much higher rank, resulting in a much higher test RMSE in these cases.

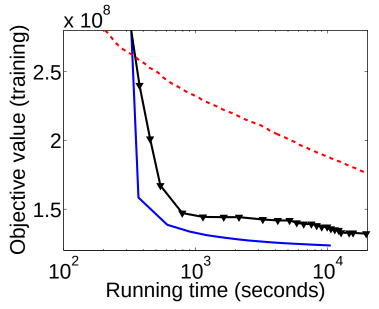
MMBS takes nearly the same number of iterations as GCG to achieve the same objective value (see Figure 2(c) and 3(c)). However, its local search at each iteration is conducted on a *constrained* manifold, making it much more expensive than locally optimizing the unconstrained smooth surrogate (53). As a result, the overall computational complexity of MMBS is an order of magnitude greater than that of GCG, with respect to optimizing both the objective value and the test RMSE.

The observed discrepancies between the relative efficiency of optimizing the objective value and test RMSE becomes much smaller on the other three data sets, which are much larger in size. In Figure 4 to 6, it is clear that GCG takes significantly less CPU time to

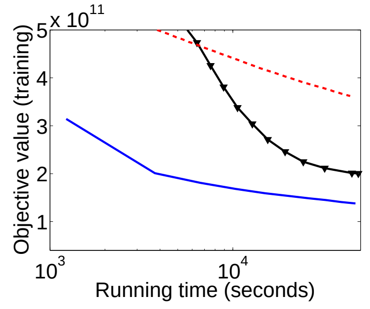
GCG
MMBS
DHM
JS



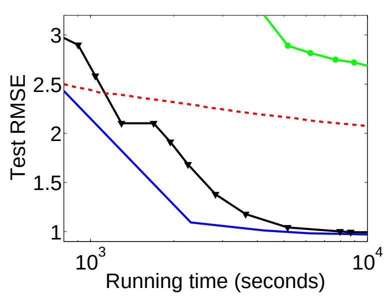
(a) Objective value vs CPU time



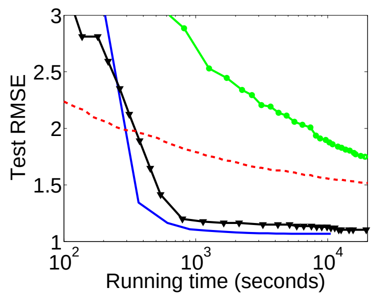
(a) Objective value vs CPU time



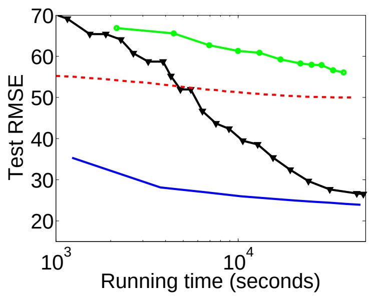
(a) Objective value vs CPU time



(b) Test RMSE vs CPU time



(b) Test RMSE vs CPU time



(b) Test RMSE vs CPU time

Figure 4: Matrix completion on `Netflix`

Figure 5: Matrix completion on `Yahoo100m`

Figure 6: Matrix completion on `Yahoo252m`

optimize both criteria. The declining trend of the objective value has a similar shape to that of the test RMSE. Here we observe that DHM is substantially slower than GCG and MMBS, confirming the importance of optimizing the bases at the same time as optimizing their weights. JS, which does not exploit the penalized form of the objective, is considerably slower with respect to reducing the test RMSE.

5.2 Multi-class Classification with Trace Norm Regularization

Next we compared the four algorithms on multi-class classification problems also in the context of trace norm regularization. Here the task is to predict the class membership of input instances, $\mathbf{x}_i \in \mathbb{R}^n$, where each instance is associated with a unique label $y_i \in \{1, \ldots, C\}$. In particular, we consider a standard linear classification model where each class c is associated with a weight vector in $\mathbb{R}^n$, stacked in a model matrix $W \in \mathbb{R}^{n \times C}$. Then for each training example $(\mathbf{x}_i, y_i)$, the individual logistic loss $L(W; \mathbf{x}_i, y_i)$ is given by

$$L(W; \mathbf{x}_i, y_i) = -\log \frac{\exp(\mathbf{x}_i' W_{:,y_i})}{\sum_{c=1}^{C} \exp(\mathbf{x}_i' W_{:c})},$$

data set	n	C	#train	#test	λ
`Fungus10`	4,096	10	10 per class	10 per class	10^{-2}
`Fungus134`	4,096	134	50 per class	50 per class	10^{-3}
`k1024`	131,072	50	50 per class	50 per class	10^{-3}

Table 2: Summary of the data sets used in multi-class classification experiments. Here n is the number of features, C is the number of classes, #train and #test are the number of training and test images respectively.

and the complete loss is given by averaging over the training set, $\ell(W) := \frac{1}{m}\sum_{i=1}^{m} L(W; \mathbf{x}_i, y_i)$. It is reasonable to assume that W has a low rank; i.e. the weight vector of all classes lie in a low dimensional subspace (Akata et al., 2014), which motivates the introduction of a trace norm regularizer on W, yielding a training problem in the form of (47).

We conducted experiments on three data sets extracted from the ImageNet repository (Deng et al., 2009), with characteristics shown in Table 2 (Akata et al., 2014).[12] `k1024` is from the ILSVRC2011 benchmark, and `Fungus10` and `Fungus134` are from the Fungus group. All images are represented by the Fisher descriptor.

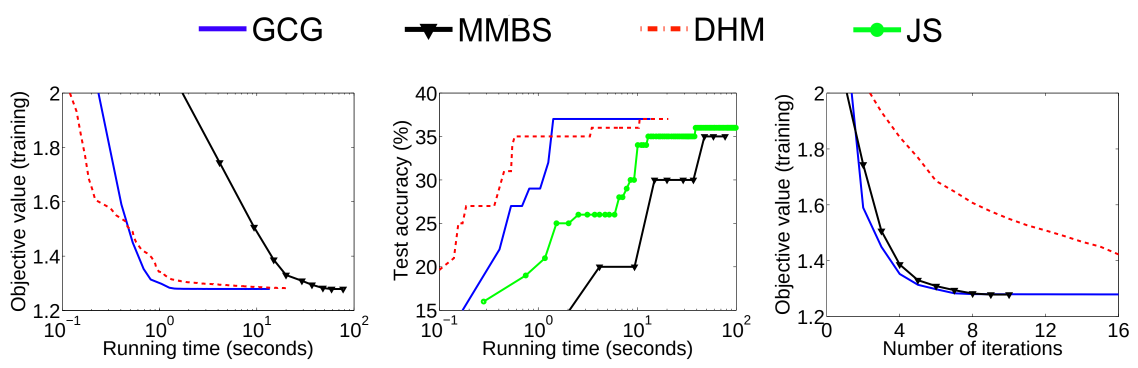
As in the matrix completion case above, we again compared how fast the algorithms reduced the objective value and test accuracy. Here the value of λ was chosen from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ to maximize test accuracy. Figures 7 to 9 show that the models produced by GCG achieve comparable (or better) training objectives and test accuracies in far less time than MMBS, DHM and JS. Although DHM is often more efficient than GCG early on, it is soon overtaken. This observation could be related to the diminishing return of adding bases, as shown in Figure 7(c), 8(c), and 9(c), since the addition of the first few bases yields a much steeper decrease in objective value than that achieved by later bases. Hence, after the initial stage, the computational efficiency of a solver turns to be dominated by better exploitation of the bases. That is, using added bases as an initialization for local improvement (as in GCG) appears to be more effective than simply finding an optimal conic combination of added bases (as in DHM).

Also notice from Figures 7(c), 8(c) and 9(c) that MMBS and GCG are almost identical in terms of the number of iterations required to achieve a given objective value. However, the manifold based local optimization employed by MMBS at each iteration makes its overall computational time worse than that of DHM and JS. This is a different outcome from the matrix completion experiments we saw earlier, and seems to suggest that the manifold approach in MMBS is more effective for the least square loss than for the logistic loss.

5.3 Multi-view Dictionary Learning

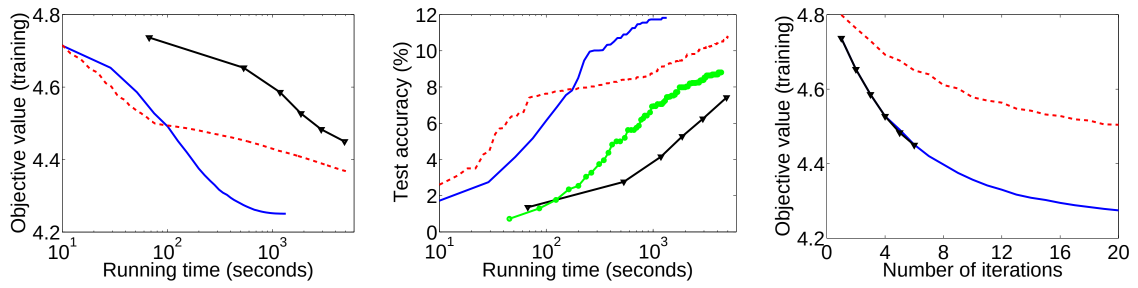
Our third set of experiments focuses on optimizing the objective for the multi-view dictionary learning model (57) presented in Section 4.4. Here we compared the GCG approach outlined in Section 4.4 with a straightforward *local* solver that is based on block coordinate descent (BCD), which alternates between: (a) fixing H and optimizing A and B (with norm

12. The three data sets were all downloaded from `http://lear.inrialpes.fr/src/jsgd/`.



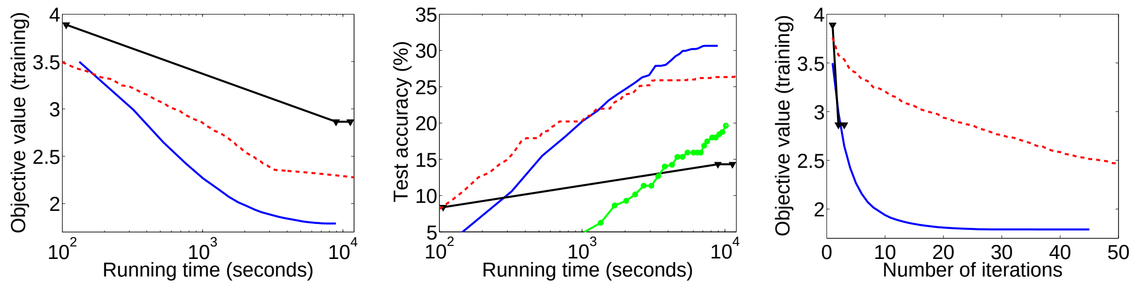
(a) Objective value vs CPU time (b) Test accuracy vs CPU time (c) Test accuracy vs #iteration

Figure 7: Multi-class classification on `Fungus10`



(a) Objective value vs CPU time (b) Test accuracy vs CPU time (c) Objective value vs #iteration

Figure 8: Multi-class classification on `Fungus134`



(a) Objective value vs CPU time (b) Test accuracy vs CPU time (c) Objective value vs #iteration

Figure 9: Multi-class classification on `k1024`

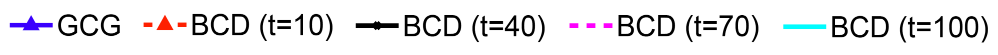
ball constraints), then (b) fixing A and B and optimizing H (with nonsmooth regularizer). Both steps were solved by FISTA (Beck and Teboulle, 2009), initialized with the solution from the previous iteration. We tested several values of t, the dimensionality of the latent subspace of the solution, in these experiments. The GCG approach used Algorithm 3 with the local objective (53) instantiated by the norm expression in (58) and solved by L-BFGS with max iteration set to 20. We initialized the U and V matrices with 20 columns and rows respectively, each drawn i.i.d. from a uniform distribution on the unit sphere. Based on the noise model we consider in these experiments, we started with a base loss ℓ set to $\mathfrak{l}_1(Z, Z^*) = \sum_{ij} |Z_{ij} - Z^*_{ij}|$ where Z^* is the true underlying matrix; however, following (Becker et al., 2009, Eq 3.1), we smoothed this $\mathfrak{l}_1$ loss by adding a quadratic prox-function with strong convexity modulus 10^{-4}. Under the resulting loss, the optimization over A and B is decoupled given H. We terminated all algorithms after a maximum of running time (3,600 seconds) is reached.

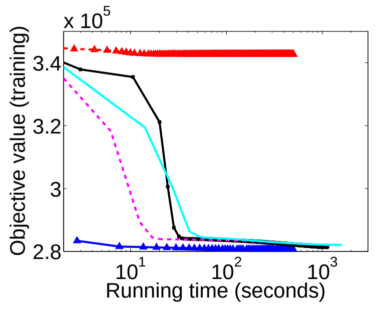
5.3.1 Synthetic Data

We first constructed experiments on a synthetic data set that fulfills the conditional independence assumption (57) of the multi-view model (as discussed in Section 4.4). Here the x- and y-views have basis matrices $A^* \in \mathbb{R}^{n_1 \times t^*}$ and $B^* \in \mathbb{R}^{n_2 \times t^*}$ respectively, where all columns of A^* were drawn i.i.d. from the $\mathfrak{l}_2$ sphere of radius $\beta = 1$, and all columns of B^* were drawn i.i.d. from the $\mathfrak{l}_2$ sphere of radius $\gamma = 5$. We set $n_1 = 800$, $n_2 = 1000$, and $t^* = 10$ (so that the dictionary size is indeed small). The latent representation matrix $H^* \in \mathbb{R}^{t^* \times m}$ has all elements drawn i.i.d. from the zero-mean unit-variance Gaussian distribution. The number of training examples is set to $m = 200$, which is much lower than the number of features n_1 and n_2. Then the clean versions of x-view and y-view were generated by $X^* = A^*H^*$ and $Y^* = B^*H^*$ respectively, while the noisy versions were obtained by adding i.i.d. noise to 15% entries of X^* and Y^* that were selected uniformly at random, yielding $\hat{X}$ and $\hat{Y}$ respectively. The noise is uniformly distributed in $[0, 10]$. Given the noisy observations, the denoising task is to find reconstructions X and Y for the two views, such that the error $\|X - \hat{X}\|_F^2 + \|Y - \hat{Y}\|_F^2$ is small. The composite training problem is formulated in (57).

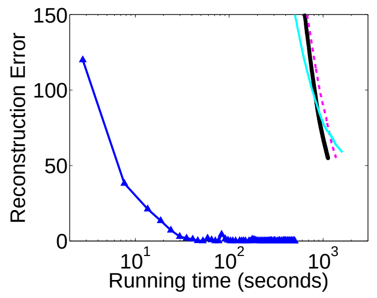
The results of comparing the modified GCG method with local optimization are presented in Figures 10 to 12, where the regularization parameter λ has been varied among $\{80, 60, 40\}$. In each case, GCG optimizes the objective value significantly faster than BCD for all values of t chosen for BCD. The differences between methods are even more evident when considering the reconstruction error. In particular, although the local optimization in GCG already achieves a low objective value in the first outer iteration (see Figure 10(a) and 11(a)), the reconstruction error remains high, thus requiring further effort to reduce it.

When $t = 10$ which equals the true rank $t^* = 10$, BCD is more likely to get stuck in a poor local minimum when λ is large, for example as shown in Figure 10(a) for $\lambda = 80$. This occurs because when optimizing H, the strong shrinkage in the proximal update stifles major changes in H (and in A and B consequently). This problem is exacerbated when the rank of the solution is overly restricted, leading to even worse local optima. On the other hand, under-regularization does indeed cause overfitting in this case; for example, as shown in Figure 12 for $\lambda = 40$. Interestingly, BCD gets trapped in local optima for all t, while GCG eventually escapes suboptimal solutions and eventually finds a way to considerably reduce the objective value. However, this simply creates a dramatic *increase* in the test reconstruction error, a typical phenomenon of overfitting.



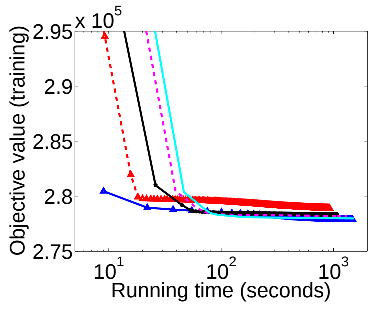


(a) Objective value vs CPU time

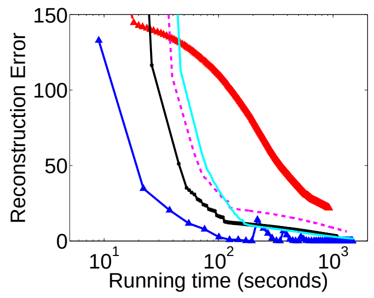


(b) Reconst. error vs CPU time

Figure 10: Denoising on synthetic data $\lambda = 80$

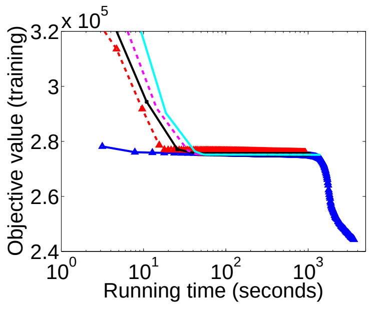


(a) Objective value vs CPU time

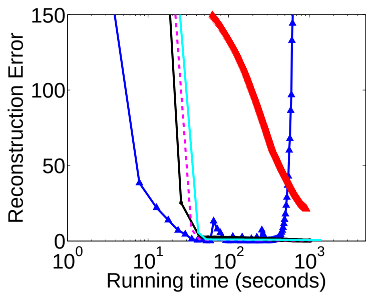


(b) Reconst. error vs CPU time

Figure 11: Denoising on synthetic data $\lambda = 60$



(a) Objective value vs CPU time



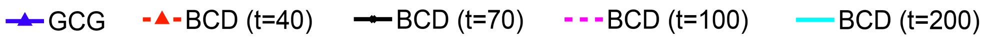
(b) Reconst. error vs CPU time

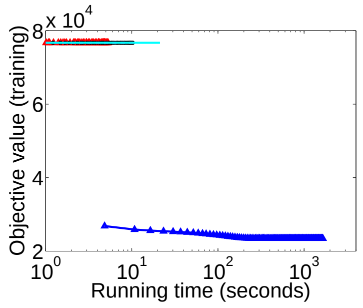
Figure 12: Denoising on synthetic data $\lambda = 40$

5.3.2 Image Denoising

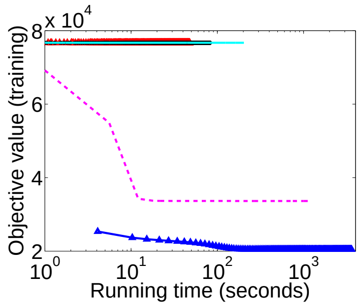
We then conducted multi-view dictionary learning experiments on natural image data. The data set we considered is based on the Extended Yale Face Database B[13] (Georghiades et al., 2001). In particular, the data set we used consists of grey level face images of 28 human subjects, each with 9 poses and 64 lighting conditions. To construct the data set, we set the x-view to a fixed lighting (+000E+00) and the y-view to a different fixed lighting (+000E+20). We obtained paired views by randomly drawing a subject and a pose under the two lighting conditions. The underlying assumption is that each lighting has its own set of bases (A and B) and each (person, pose) pair has the same latent representation for the two lighting conditions. All images were down-sampled to 50-by-50, meaning $n_1 = n_2 = 2500$, and we randomly selected $t = 50$ pairs of (person, pose) for training. The x-view was kept clean, while pixel errors were added to the y-view, randomly setting 10% of the pixel values to 1. This noise model mimics natural phenomena such as occlusion and loss of pixel information from image transfer. The goal is again to enable appropriate reconstruction of

13. Data downloaded from `http://vision.ucsd.edu/content/extended-yale-face-database-b-b`.

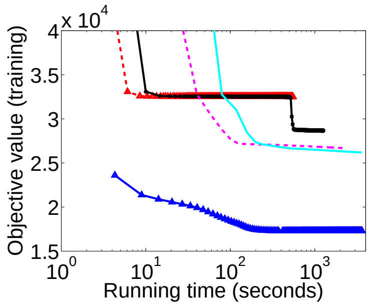




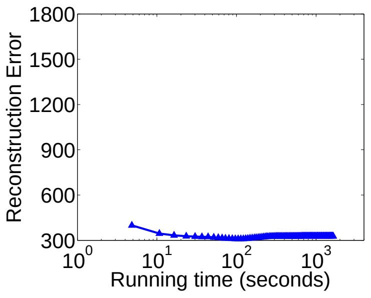
(a) Objective value vs CPU time



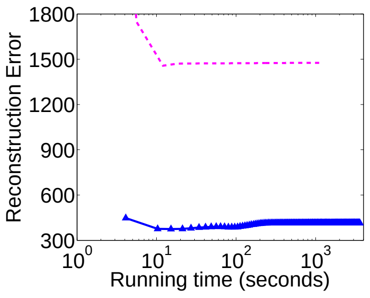
(a) Objective value vs CPU time



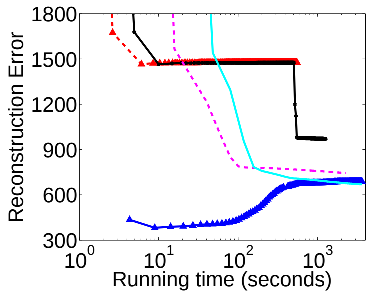
(a) Objective value vs CPU time



(b) Reconst. error vs CPU time



(b) Reconst. error vs CPU time



(b) Reconst. error vs CPU time

Figure 13: Denoising on face data with $\lambda = 40$

Figure 14: Denoising on face data with $\lambda = 30$

Figure 15: Denoising on face data with $\lambda = 20$

a noisy image using other views. Naturally, we set $\beta = \gamma = 1$ since the data is in the $[0, 1]$ interval.

Figures 13 to 15 show the results comparing GCG with the local BCD for different choices of t for BCD. Clearly the value of λ that yields the lowest reconstruction error is 40. However, here BCD simply gets stuck at the trivial local minimum $H = \mathbf{0}$, as shown in Figure 13(a). This occurs because under the random initialization of A and B, the proximal update simply sets H to $\mathbf{0}$ when the regularization is too strong. When λ is decreased to 30, as shown in Figure 14(a), BCD can escape the trivial $H = \mathbf{0}$ solution for $t = 100$, but again it becomes trapped in another local minimum that remains much worse than the globally optimal solution found by GCG. When λ is further reduced to 20, the BCD algorithms finally escape the trivial $H = \mathbf{0}$ point for all t, but still fail to find an acceptable solution. Although GCG again converges to a much better global solution in terms of the objective value, overfitting can still occur: in Figure 15(b) the reconstruction error of GCG increases significantly after an initial decline.

6. Conclusion

We have presented a unified treatment of the generalized conditional gradient (GCG) algorithm that covers several useful extensions. After a thorough treatment of its convergence properties, we illustrated how an extended form of GCG can efficiently handle a convex gauge regularizer in particular. We illustrated the application of these GCG methods in learning low rank matrices, instantiated with examples on matrix completion, dictionary learning, and multi-view learning. In each case, we considered a generic convex relaxation procedure and focused on the efficient computation of the polar operator—one of the key steps to obtaining an efficient implementation of GCG. To further accelerate its convergence, we interleaved the standard GCG update with a fixed rank subspace optimization, which greatly improves performance in practice without affecting the theoretical convergence properties. Extensive experiments on matrix completion, multi-class classification, multi-view learning confirmed the superiority of the modified GCG approach in practice.

For future work, several interesting extensions are worth investigating. The current form of GCG is inherently framed as a batch method, where each iteration requires gradient of the loss to be computed over *all* training data. When the training data is large, it is preferable to only use a random subset, or distribute the training data across several processors. Another interesting direction is to interpolate between the polar operator and proximal updates; that is, one could incorporate a mini-batch of atoms at each iteration. For example, for trace norm regularization, one can consider adding k instead of just the 1 largest singular vectors. It remains an open question whether such a "mini-batch" of atoms can provably accelerate the overall optimization, but some recent work (Hsieh and Olsen, 2014) has demonstrated promise in practice.

Acknowledgments

This research was supported by the Alberta Innovates Center for Machine Learning (AICML), the Canada Research Chairs program, and NSERC. Most of this work was carried out when Y Yu and X Zhang were with the University of Alberta and AICML. X Zhang would like to thank the support of NICTA, where he worked as a researcher from Oct 2012 to Oct 2015 and conducted 50% of his work on this paper. We thank Csaba Szepesvári, the action editor, and the reviewers at various stages of this work for the many constructive comments.

Appendix A. Proofs for Section 3

A.1 Proof of Theorem 2

Since $\nabla \ell$ is Lipschitz continuous, it must be uniformly continuous on bounded set.

(a): Let C be the closure of the sequence $\{\mathbf{w}_t\}_t$. Due to the compactness assumption on the sublevel set and the monotonicity of $F(\mathbf{w}_t)$, C is compact. Moreover $C \subseteq \operatorname{dom} f$ since for all cluster points, say $\mathbf{w}$, of $\mathbf{w}_t$ we have from the closedness of F that $F(\mathbf{w}) \leq \liminf F(\mathbf{w}_{t_k}) \leq F(\mathbf{w}_0) < \infty$. Since $-\nabla \ell$ is continuous, $-\nabla \ell(C)$ is a compact subset of $\operatorname{int}(\operatorname{dom} f^*)$, the interior of $\operatorname{dom} f^*$. Note that f^* is continuous on the interior of its domain, therefore its subdifferential is locally bounded on $-\nabla \ell(C)$, see e.g. (Borwein and

Vanderwerff, 2010, Proposition 4.1.26). A standard compactness argument then establishes the boundedness of $(\partial f^*)(-\nabla \ell(C))$. Thus $\{\mathbf{d}_t\}_t$ is bounded.

(b): Note first that the boundedness of $\{\mathbf{w}_t\}_t$ follows immediately from the boundedness assumption on the sublevel set, thanks to the monotonic property of $F(\mathbf{w}_t)$. Since $\nabla \ell$ is uniformly continuous, the set $\{-\nabla \ell(\mathbf{w}_t)\}_t$ is again bounded. On the other hand, we know from (Borwein and Vanderwerff, 2010, Theorem 4.4.13, Proposition 4.1.25) that f is super-coercive iff ∂f^* maps bounded sets into bounded sets. Therefore $\{\mathbf{d}_t\}_t$ is again bounded.

(c): This is clear.

Note that the convexity of ℓ is not used in this proof.

A.2 Proof of Theorem 4

Since ℓ is L-smooth, for the sequences $\{\mathbf{w}_t\}$ and $\{\mathbf{d}_t\}$ generated by Algorithm 1, and for all $\eta \in [0,1]$, we have

$$\ell(\mathbf{w}_t + \eta(\mathbf{d}_t - \mathbf{w}_t)) \le \ell(\mathbf{w}_t) + \eta \langle \mathbf{d}_t - \mathbf{w}_t, \nabla \ell(\mathbf{w}_t)\rangle + \tfrac{L\eta^2}{2} \|\mathbf{d}_t - \mathbf{w}_t\|^2 . \tag{63}$$

Since the subroutine is `Relaxed` and the subproblem (13) is solved up to some $\varepsilon_t \ge 0$,

$$\begin{aligned} F(\mathbf{w}_{t+1}) &\le \ell(\mathbf{w}_t) + \eta_t \langle \mathbf{d}_t - \mathbf{w}_t, \nabla \ell(\mathbf{w}_t)\rangle + \tfrac{L\eta_t^2}{2} \|\mathbf{d}_t - \mathbf{w}_t\|^2 + (1-\eta_t) f(\mathbf{w}_t) + \eta_t f(\mathbf{d}_t) \\ &\le F(\mathbf{w}_t) - \eta_t \mathsf{G}(\mathbf{w}_t) + \tfrac{L\eta_t^2}{2} \|\mathbf{d}_t - \mathbf{w}_t\|^2 + \eta_t \varepsilon_t \\ &= F(\mathbf{w}_t) - \eta_t \mathsf{G}(\mathbf{w}_t) + \eta_t^2 (\varepsilon_t/\eta_t + \tfrac{L}{2} \|\mathbf{d}_t - \mathbf{w}_t\|^2). \end{aligned}$$

Define $\Delta_t := F(\mathbf{w}_t) - F(\mathbf{w})$ and $\mathsf{G}_t := \mathsf{G}(\mathbf{w}_t)$. Thus

$$\Delta_{t+1} \le \Delta_t - \eta_t \mathsf{G}_t + \eta_t^2(\varepsilon_t/\eta_t + \tfrac{L}{2}\|\mathbf{d}_t - \mathbf{w}_t\|^2), \qquad \text{and} \qquad \Delta_t \le \mathsf{G}_t. \tag{64}$$

Plug the latter into the former and expand:

$$\Delta_{t+1} \le \pi_t(1-\eta_0)\Delta_0 + \sum_{s=0}^{t} \frac{\pi_t}{\pi_s} \eta_s^2 (\varepsilon_s/\eta_s + \tfrac{L}{2}\|\mathbf{d}_s - \mathbf{w}_s\|^2). \tag{65}$$

To prove the second claim, we have from (64) $\eta_t \mathsf{G}_t \le \Delta_t - \Delta_{t+1} + \eta_t^2(\varepsilon_t/\eta_t + \frac{L}{2}\|\mathbf{d}_t - \mathbf{w}_t\|^2)$. Summing from k to t:

$$\left(\min_{k \le s \le t} \mathsf{G}_s\right) \sum_{s=k}^{t} \eta_s \le \sum_{s=k}^{t} \eta_s \mathsf{G}_s \le F(\mathbf{w}_k) - F(\mathbf{w}_{t+1}) + \sum_{s=k}^{t} \eta_s^2 (\varepsilon_s/\eta_s + \tfrac{L}{2}\|\mathbf{d}_s - \mathbf{w}_s\|^2).$$

Rearranging completes the proof.

A.3 Proof of Theorem 5

As $\eta_t = \frac{2}{t+2}$, we have $\eta_0 = 1$ and $\pi_t = \frac{2}{(t+1)(t+2)}$. For the first claim (upper bound on $F(\mathbf{w}_t)$) all we need to verify is that, by induction, $\frac{1}{(t+1)(t+2)} \sum_{s=0}^{t} \frac{s+1}{s+2} \le \frac{1}{t+4}$. This is elementary.

We prove the second claim by a sequence of calculations similar to that in (Freund and Grigas, 2016). First, using (19) and the bound on $F(\mathbf{w}_t)$ in (20) with $t = 1$ and $\mathbf{w} = \mathbf{w}_2$:

$$\tilde{\mathsf{G}}_1^1 \le \tfrac{1}{\eta_1}\left(F(\mathbf{w}_1) - F(\mathbf{w}_2) + \tfrac{\eta_1^2}{2}(\delta + L_F)\right) \le \tfrac{3}{2}\left(\tfrac{1}{2}(\delta + L_F) + \tfrac{2}{9}(\delta + L_F)\right) = \tfrac{13}{12}(\delta + L_F),$$

proving the second claim on $\tilde{\mathsf{G}}_1^t$ for $t = 1, 2$ (note that $\tilde{\mathsf{G}}_1^2 \le \tilde{\mathsf{G}}_1^1$ by definition). For $t \ge 3$, we consider $k = t/2 - 1$ if t is even and $k = (t+1)/2 - 1$ otherwise. Clearly $k \ge 1$, hence

$$\sum_{s=k}^{t} \eta_s = 2\sum_{s=k}^{t} \frac{1}{s+2} \ge 2\int_{k-1}^{t} \frac{1}{s+2}\mathrm{d}s = 2\ln\frac{t+2}{k+1} \ge 2\ln 2,$$
$$\sum_{s=k}^{t} \eta_s^2/2 = 2\sum_{s=k}^{t} \frac{1}{(s+2)^2} \le 2\int_{k}^{t+1} \frac{1}{(s+2)^2}\mathrm{d}s = 2\left(\frac{1}{k+2} - \frac{1}{t+3}\right).$$

Using again (19), and the bound on $F(\mathbf{w}_t)$ in (20) with $t = k$ and $\mathbf{w} = \mathbf{w}_{t+1}$:

$$\tilde{\mathsf{G}}_1^t \le \tilde{\mathsf{G}}_k^t \le \frac{\delta + L_F}{2\ln 2}\left(\frac{2}{k+3} + \frac{2}{k+2} - \frac{2}{t+3}\right) \le \frac{\delta + L_F}{\ln 2}\left(\frac{2}{t+2} + \frac{1}{t+3}\right) \le \frac{3(\delta + L_F)}{t\ln 2}.$$

A.4 Proof of Theorem 8

A proof can be based on the simple observation that the objective value $F^\star$ satisfies

$$F^\star := \inf_{\mathbf{w}}\{\ell(\mathbf{w}) + f(\mathbf{w})\} = \inf_{\mathbf{w},\rho:\kappa(\mathbf{w})\le\rho} \ell(\mathbf{w}) + h(\rho). \tag{66}$$

Note that if ρ were known, the theorem could have been proved as before. The key idea behind the step size choice (34) is to ensure that Algorithm 2 performs almost the same as if it knew the *unknown* but fixed constant $\rho = \kappa(\mathbf{w})$ beforehand.

Consider an arbitrary $\mathbf{w}$ and let $\rho = \kappa(\mathbf{w})$. We also use the shorthand $\hat{F}_t := \ell(\mathbf{w}_t) + h(\rho_t) \ge \ell(\mathbf{w}_t) + f(\mathbf{w}_t) = F(\mathbf{w}_t)$. Then, the following chain of inequalities can be verified:

$$\begin{aligned}
\hat{F}_{t+1} &:= \ell(\mathbf{w}_{t+1}) + h(\rho_{t+1}) \\
&\le \hat{F}_t + \langle \theta_t\mathbf{a}_t - \eta_t\mathbf{w}_t, \nabla\ell(\mathbf{w}_t)\rangle + \tfrac{L}{2}\left\|\theta_t\mathbf{a}_t - \eta_t\mathbf{w}_t\right\|^2 - \eta_t h(\rho_t) + \eta_t h(\theta_t/\eta_t) \\
&\le \hat{F}_t + \eta_t\left\langle \tfrac{\rho}{\alpha_t}\mathbf{a}_t - \mathbf{w}_t, \nabla\ell(\mathbf{w}_t)\right\rangle + \tfrac{L\eta_t^2}{2}\left\|\tfrac{\rho}{\alpha_t}\mathbf{a}_t - \mathbf{w}_t\right\|^2 - \eta_t h(\rho_t) + \eta_t h(\rho/\alpha_t) \\
&\le \hat{F}_t + \min_{\mathbf{z}:\kappa(\mathbf{z})\le\rho} \eta_t\langle \mathbf{z} - \mathbf{w}_t, \nabla\ell(\mathbf{w}_t)\rangle + \eta_t\rho\varepsilon_t + \tfrac{L\eta_t^2}{2}\left\|\tfrac{\rho}{\alpha_t}\mathbf{a}_t - \mathbf{w}_t\right\|^2 - \eta_t h(\rho_t) + \eta_t h(\rho/\alpha_t) \\
&= \hat{F}_t + \min_{\mathbf{z}:\kappa(\mathbf{z})\le\rho} \eta_t\langle \mathbf{z} - \mathbf{w}_t, \nabla\ell(\mathbf{w}_t)\rangle - \eta_t h(\rho_t) + \eta_t h(\rho) \\
&\qquad + \eta_t^2\underbrace{\left(\tfrac{L}{2}\left\|\tfrac{\rho}{\alpha_t}\mathbf{a}_t - \mathbf{w}_t\right\|^2 + (\rho\varepsilon_t + h(\rho/\alpha_t) - h(\rho))/\eta_t\right)}_{:=\delta_t} \\
&= \hat{F}_t + \eta_t^2\delta_t - \eta_t\underbrace{\Big[\langle \mathbf{w}_t, \nabla\ell(\mathbf{w}_t)\rangle + h(\rho_t) - \min_{\mathbf{z}:\kappa(\mathbf{z})\le\rho}\langle \mathbf{z}, \nabla\ell(\mathbf{w}_t)\rangle + h(\rho)\Big]}_{:=\hat{\mathsf{G}}(\mathbf{w}_t)},
\end{aligned}$$

where the first inequality is because the subroutine is `Relaxed`, the second inequality follows from the minimality of θ_t in (34), and the third inequality is due to the choice of $\mathbf{a}_t$ in Line 3 of Algorithm 2.

Recall that $\rho = \kappa(\mathbf{w})$. Moreover, due to the convexity of ℓ,

$$\begin{aligned}\hat{\mathsf{G}}(\mathbf{w}_t) &= \langle \mathbf{w}_t, \nabla\ell(\mathbf{w}_t)\rangle + h(\rho_t) - \min_{\mathbf{z}:\kappa(\mathbf{z})\le\rho} \big(\langle \mathbf{z}, \nabla\ell(\mathbf{w}_t)\rangle + h(\rho)\big)\\ &= h(\rho_t) - h(\kappa(\mathbf{w})) + \max_{\mathbf{z}:\kappa(\mathbf{z})\le\kappa(\mathbf{w})} \langle \mathbf{w}_t - \mathbf{z}, \nabla\ell(\mathbf{w}_t)\rangle\\ &\ge h(\rho_t) - h(\kappa(\mathbf{w})) + \max_{\mathbf{z}:\kappa(\mathbf{z})\le\kappa(\mathbf{w})} \ell(\mathbf{w}_t) - \ell(\mathbf{z})\\ &\ge \hat{F}_t - F(\mathbf{w}).\end{aligned}$$

Thus we have retrieved the recursion:

$$\hat{F}_{t+1} - F(\mathbf{w}) \le \hat{F}_t - F(\mathbf{w}) - \eta_t\hat{\mathsf{G}}(\mathbf{w}_t) + \eta_t^2\delta_t, \qquad \text{and} \qquad \hat{F}_t - F(\mathbf{w}) \le \hat{\mathsf{G}}(\mathbf{w}_t).$$

Summing the indices as in the proof of Theorem 4 and noting that $F(\mathbf{w}_t) \le \hat{F}_t$ for all t completes the proof.

Appendix B. Proofs for Section 4

B.1 Proof of Theorem 15

The first equivalence in Theorem 15 is well-known since Grothendieck's work on tensor products of Banach spaces (where it is usually called the projective tensor norm). The second equality follows directly from the arithmetic-geometric mean inequality.

We first note that the atomic set $\mathcal{A}$ in (48) is compact, so is its convex hull $\operatorname{conv}\mathcal{A}$. It is also easy to see that $\mathcal{A}$ is a connected set (by considering the continuous map $(\mathbf{u}, \mathbf{v}) \mapsto \mathbf{u}\mathbf{v}^\top$).

$$\begin{aligned}\text{By (4), } \kappa(W) &= \inf\Big\{\rho \ge 0 : W = \rho\sum\nolimits_i \sigma_i\mathbf{a}_i, \sigma_i \ge 0, \sum\nolimits_i \sigma_i = 1, \mathbf{a}_i \in \mathcal{A}\Big\} \qquad (67)\\ &= \inf\{\rho \ge 0 : W \in \rho\ \operatorname{conv}\mathcal{A}\}.\end{aligned}$$

Since $\operatorname{conv}\mathcal{A}$ is compact with $\mathbf{0} \in \operatorname{int}(\operatorname{conv}\mathcal{A})$ we know the infimum above is attained. Thus there exist $\rho \ge 0$ and $C \in \operatorname{conv}\mathcal{A}$ so that $W = \rho C$ and $\kappa(W) = \rho$. Applying Caratheodory's theorem for connected sets (Rockafellar and Wets, 1998, Theorem 2.29) we know $C = \sum_{i=1}^t \sigma_i\mathbf{u}_i\mathbf{v}_i^\top$ for some $\sigma_i \ge 0$, $\sum_i \sigma_i = 1$, $\mathbf{u}_i\mathbf{v}_i^\top \in \mathcal{A}$ and $t \le mn$. Let $U = \sqrt{\rho}\,[\sqrt{\sigma_1}\mathbf{u}_1, \ldots, \sqrt{\sigma_t}\mathbf{u}_t]$, and $V = \sqrt{\rho}\,[\sqrt{\sigma_1}\mathbf{v}_1, \ldots, \sqrt{\sigma_t}\mathbf{v}_t]^\top$. We then have $W = UV$ and $\kappa(W) = \rho \ge \frac{1}{2}\sum_{i=1}^t \Big(\|U_{:i}\|_c^2 + \|V_{i:}\|_r^2\Big)$, which proves one side of Theorem 15.

On the other hand, consider any U and V that satisfy

$$W = UV = \sum_{j=1}^t \|U_{:j}\|_c \|V_{j:}\|_r \sum_{i=1}^t \frac{\|U_{:i}\|_c \|V_{i:}\|_r}{\sum_{j=1}^t \|U_{:j}\|_c \|V_{j:}\|_r} \frac{U_{:i}}{\|U_{:i}\|_c} \frac{V_{i:}}{\|V_{i:}\|_r},$$

assuming w.l.o.g. that $\|U_{:i}\|_c \|V_{i:}\|_r \ne 0$ for all $1 \le i \le t$. This, together with the definition (67), gives the other half of the equality: $\kappa(W) \le \sum_{i=1}^t \|U_{:i}\|_c \|V_{i:}\|_r$. The proof is completed by making the elementary observation that $\sum_{i=1}^t \|U_{:i}\|_c \|V_{i:}\|_r \le \frac{1}{2}\sum_{i=1}^t \Big(\|U_{:i}\|_c^2 + \|V_{i:}\|_r^2\Big)$.

B.2 Proof of Theorem 16

Since $U_{\text{init}} = (\sqrt{1-\eta_t}U_t, \sqrt{\theta_t}\mathbf{u}_t)$ and $V_{\text{init}} = (\sqrt{1-\eta_t}V_t^\top, \sqrt{\theta_t}\mathbf{v}_t)^\top$, it is obvious that $\tilde{W}_{t+1} = U_{\text{init}}V_{\text{init}}$. Note $\|\mathbf{u}_t\|_c \le 1$ and $\|\mathbf{v}_t\|_r \le 1$. By construction

$$\begin{aligned}
\ell(W_{t+1}) + \lambda\rho_{t+1} &= \ell(U_{t+1}V_{t+1}) + \frac{\lambda}{2}\sum_{i=1}^{t+1}\left(\|(U_{t+1})_{:i}\|_c^2 + \|(V_{t+1})_{i:}\|_r^2\right) \\
&= \mathcal{F}_{t+1}(U_{t+1}, V_{t+1}) \ \le\ \mathcal{F}_{t+1}(U_{\text{init}}, V_{\text{init}}) \\
&\le \ell((1-\eta_t)U_tV_t + \theta_t\mathbf{u}_t\mathbf{v}_t^\top) + \lambda\theta_t + \lambda\frac{(1-\eta_t)}{2}\sum_{i=1}^{t}\left(\|(U_t)_{:i}\|_c^2 + \|(V_t)_{i:}\|_r^2\right) \\
&= \ell((1-\eta_t)W_t + \theta_t\mathbf{u}_t\mathbf{v}_t^\top) + \lambda(1-\eta_t)\rho_t + \lambda\theta_t, \qquad (68)
\end{aligned}$$

where the inequality is by the definition of U_{init} and V_{init}, and the last equality is by the definition of ρ_t in (54). In addition, $\kappa(W_{t+1}) \le \rho_{t+1}$ follows from (54) and Theorem 15.

B.3 Proof of Theorem 21

By (50), the square of the polar is

$$(\kappa^\circ(G))^2 = \max\left\{\mathbf{c}^\top ZZ^\top\mathbf{c} : \|\mathbf{a}\|_2 \le 1, \|\mathbf{b}\|_2 \le 1\right\}, \quad \text{where} \quad \mathbf{c} = \begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}. \qquad (69)$$

Since it maximizes a convex quadratic function over a convex set, there must be a maximizer attained at an extreme point of the feasible region. Therefore the problem is equivalent to

$$\begin{aligned}
(\kappa^\circ(G))^2 &= \max\left\{\mathbf{c}^\top ZZ^\top\mathbf{c} : \|\mathbf{a}\|_2 = 1, \|\mathbf{b}\|_2 = 1\right\} \qquad (70)\\
&= \max\left\{\operatorname{tr}(SZZ^\top) : \operatorname{tr}(SI_1) = 1, \operatorname{tr}(SI_2) = 1, S \succeq 0, \operatorname{rank}(S) = 1\right\}, \qquad (71)
\end{aligned}$$

where $S = \mathbf{c}\mathbf{c}^\top$. Key to this proof is the fact that the rank one constraint can be dropped without breaking the equivalence. To see why, notice that without this rank constraint the feasible region is the intersection of the positive semidefinite cone with two hyperplanes, hence the rank of all its extreme points must be upper bounded by one (Pataki, 1998). Furthermore the linearity of the objective implies that there must be a maximizing solution attained at an extreme point of the feasible region. Therefore we can continue by

$$\begin{aligned}
(\kappa^\circ(G))^2 &= \max\left\{\operatorname{tr}(SZZ^\top) : \operatorname{tr}(SI_1) = 1, \operatorname{tr}(SI_2) = 1, S \succeq 0\right\} \\
&= \max_{S\succeq 0}\ \min_{\mu_1,\mu_2\in\mathbb{R}} \operatorname{tr}(SGG^\top) - \mu_1(\operatorname{tr}(SI_1) - 1) - \mu_2(\operatorname{tr}(SI_2) - 1) \\
&= \min_{\mu_1,\mu_2\in\mathbb{R}}\ \max_{S\succeq 0} \operatorname{tr}(SGG^\top) - \mu_1(\operatorname{tr}(SI_1) - 1) - \mu_2(\operatorname{tr}(SI_2) - 1) \\
&= \min\left\{\mu_1 + \mu_2 : \mu_1, \mu_2 \in \mathbb{R}, GG^\top \preceq \mu_1 I_1 + \mu_2 I_2\right\} \qquad (72)\\
&= \min\left\{\mu_1 + \mu_2 : \mu_1 \ge 0, \mu_2 \ge 0, \left\|D_{\mu_2/\mu_1}G\right\|_{\text{sp}}^2 \le \mu_1 + \mu_2\right\} \qquad (73)\\
&= \min\left\{\|D_\mu G\|_{\text{sp}}^2 : \mu \ge 0\right\}, \qquad (74)
\end{aligned}$$

where the third equality is due to the Lagrangian duality, the fifth equality follows from the equivalence $\left\|D_{\mu_2/\mu_1}G\right\|_{\text{sp}}^2 \le \mu_1+\mu_2 \iff D_{\mu_2/\mu_1}GG^\top D_{\mu_2/\mu_1} \preceq (\mu_1+\mu_2)I \iff GG^\top \preceq (\mu_1+\mu_2)D^{-1}_{\mu_2/\mu_1}D^{-1}_{\mu_2/\mu_1} = \mu_1 I_1 + \mu_2 I_2$, and the last equality is obtained through the re-parameterization $\mu = \mu_2/\mu_1, \nu = \mu_1+\mu_2$ and the elimination of ν.

B.4 Solving the multi-view polar

Here we show how to efficiently compute the polar (dual norm) in the multi-view setting.

Let us first backtrack in (73), which produces the optimal μ_1 and μ_2 via

$$\mu_1+\mu_2 = \left\|D_\mu G\right\|_{\text{sp}}^2, \qquad \text{and} \qquad \frac{\mu_2}{\mu_1} = \mu. \tag{75}$$

Then the KKT condition for the step from (72) to (73) can be written as

$$M \succeq \mathbf{0}, \qquad M\mathbf{c} = \mathbf{0}, \qquad \|\mathbf{a}\|_2 = \|\mathbf{b}\|_2 = 1, \quad \text{where} \quad M := \mu_1 I_1 + \mu_2 I_2 - GG^\top. \tag{76}$$

By duality, this is a sufficient and necessary condition for $\mathbf{a}$ and $\mathbf{b}$ to be the solution of the polar operator. Given the optimal μ_1 and μ_2, the first condition $M \succeq \mathbf{0}$ must have been satisfied already. Let the null space of M be spanned by an orthonormal basis $\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$. Then we can parameterize $\mathbf{c}$ as $\mathbf{c} = Q\boldsymbol{\alpha}$ where $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_k] =: \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$. By (76),

$$0 = \|\mathbf{a}\|_2^2 - \|\mathbf{b}\|_2^2 = \boldsymbol{\alpha}'\left(Q_1^\top Q_1 - Q_2^\top Q_2\right)\boldsymbol{\alpha}. \tag{77}$$

Let $P\Sigma P^\top$ be the eigen-decomposition of $Q_1^\top Q_1 - Q_2^\top Q_2$ with Σ being diagonal and P being unitary. Let $\mathbf{v} = P^\top\boldsymbol{\alpha}$. Then $\mathbf{c} = QP\mathbf{v}$ and (77) simply becomes $\mathbf{v}^\top\Sigma\mathbf{v} = 0$. In addition, (76) also implies $2 = \|\mathbf{c}\|_2^2 = \mathbf{v}^\top P^\top Q^\top QP\mathbf{v} = \mathbf{v}^\top\mathbf{v}$. So finally the optimal solution to the polar problem (70) can be completely characterized by

$$\left\{ \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = QP\mathbf{v} : \mathbf{v}^\top\Sigma\mathbf{v} = 0, \|\mathbf{v}\|_2^2 = 2 \right\}, \tag{78}$$

which is simply a linear system after a straightforward change of variable. The major computational cost for recovery is to find the null space of M, which can be achieved by QR decomposition. The complexity of eigen-decomposition for $Q_1^\top Q_1 - Q_2^\top Q_2$ depends on the dimension of the null space of M, which is usually low in practice.

References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Good practice in large-scale learning for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):507–520, 2014.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

Francis Bach. Convex relaxations of structured matrix factorizations. HAL:00861118, 2013.

Francis Bach. Duality between subgradient and conditional gradient methods. *SIAM Journal of Optimization*, 25(1):115–129, 2015.

Francis Bach and Michael Jordan. A probabilistic interpretation of canonical correlation analysis. Technical Report 688, Department of Statistics, University of California, Berkeley, 2006.

Francis Bach, Julien Mairal, and Jean Ponce. Convex sparse matrix factorizations. arXiv:0812.1869v1, 2008.

Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012.

Amir Beck and Shimrit Shtern. Linearly convergent away-step conditional gradient for non-strongly convex functions, 2015.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.

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

Stephen Becker, Jérôme Bobin, and Emmanuel J. Candès. NESTA: A fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*, 4(1):1–39, 2009.

Jonathan M. Borwein and John D. Vanderwerff. *Convex Functions: Constructions, Characterizations and Counterexamples*. Cambridge University Press, 2010.

David M. Bradley and James Andrew Bagnell. Convex coding. In *Conf. on Uncertainty in Artificial Intelligence*, 2009.

Kristian Bredies and Dirk A. Lorenz. Iterated hard shrinkage for minimization problems with sparsity constraints. *SIAM Journal on Scientific Computing*, 30(2):657–683, 2008.

Peter Bühlmann and Sara van de Geer. *Statistics for High-Dimensional Data*. Springer, 2011.

Samuel Burer and Renato D C Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2009.

Michael D. Canon and Clifton D. Cullum. A tight upper bound on the rate of convergence of Frank-Wolfe algorithm. *SIAM Journal on Control*, 6:509–516, 1968.

Venkat Chandrasekaran, Benjamin Recht, Pablo A. Parrilo, and Alan S.Willsky. The convex geometry of linear inverse problems. *Foundations of Computational Mathematics*, 12(6): 805–849, 2012.

H. Cheng, Y. Yu, X. Zhang, E. Xing, and D. Schuurmans. Scalable and sound low-rank tensor learning. In *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, 2016.

Kenneth L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms*, 6(4):1–30, 2010.

Tij De Bie, Nello Cristianini, and Roman Rosipal. Eigenproblems in pattern recognition. In *Handbook of Geometric Computing*, pages 129–170, 2005.

Vladimir Fedorovich Dem'yanov and Alexander M. Rubinov. The minimization of a smooth convex functional on a convex set. *SIAM Journal on Control*, 5:280–294, 1967.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-fei. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The Yahoo! music dataset and KDD-Cup'11. *JMLR Workshop and Conference Proceedings: Proceedings of KDD Cup 2011 Competition*, 18:3–18, 2012.

Miroslav Dudik, Zaid Harchaoui, and Jerome Malick. Lifted coordinate descent for learning with trace-norm regularizations. In *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, 2012.

J. Dunn. Rates of convergence for conditional gradient algorithms near singular and non-singular extremals. *SIAM Journal on Control and Optimization*, 17:187–211, 1979.

J. Dunn and S. Harshbarger. Conditional gradient algorithms with open loop step size rules. *Journal of Mathematical Analysis and Applications*, 62:432–444, 1978.

Yonina C. Eldar and Gitta Kutyniok, editors. *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

Robert Freund and Paul Grigas. New analysis and results for the Frank-Wolfe method. *Mathematical Programming*, 155(1):199–230, 2016.

Masao Fukushima and Hisashi Mine. A generalized proximal point algorithm for certain non-convex minimization problems. *International Journal of Systems Science*, 12(8): 989–1000, 1981.

Dan Garber and Elad Hazan. Faster rates for the Frank-Wolfe method over strongly-convex sets. In *Proc. Intl. Conf. on Machine Learning*, 2015.

Dan Garber and Elad Hazan. A linearly convergent variant of the conditional gradient algorithm under strong convexity, with applications to online and stochastic optimization. *SIAM Journal on Optimization*, 26(3):1493–1528, 2016.

Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:643–660, 2001.

J. Guélat and P. Marcotte. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35:110–119, 1986.

Zaid Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152:75–112, 2015.

Elad Hazan. Sparse approximate solutions to semidefinite programs. In *Latin American Conference on Theoretical Informatics*, 2008.

Charles A. Holloway. An extension of the Frank and Wolfe method of feasible directions. *Mathematical Programming*, 6(1):14–27, 1974.

Cho-Jui Hsieh and Peder A. Olsen. Nuclear norm minimization via active subspace selection. In *Proc. Intl. Conf. on Machine Learning*, 2014.

Laurent Jacob, Guillaume Obozinski, and Jean P. Vert. Group lasso with overlap and graph lasso. In *Proc. Intl. Conf. on Machine Learning*, 2009.

Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proc. Intl. Conf. on Machine Learning*, 2013.

Martin Jaggi and Marek Sulovsky. A simple algorithm for nuclear norm regularized problems. In *Proc. Intl. Conf. on Machine Learning*, 2010.

Yangqing Jia, Mathieu Salzmann, and Trevor Darrell. Factorized latent spaces with structured sparsity. In *Advances in Neural Information Processing Systems*, pages 982–990, 2010.

Seyoung Kim and Eric P. Xing. Statistical estimation of correlated genome associations to a quantitative trait network. *PLoS Genetics*, 5(8):1–18, 2009.

Jacek Kuczyński and Henryk Woźniakowski. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1094–1122, 1992.

S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems*, 2015.

Sören Laue. A hybrid algorithm for convex semidefinite optimization. In *Proc. Intl. Conf. on Machine Learning*, 2012.

Evgeny S. Levitin and Boris T. Polyak. Constrained minimization problems. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966.

Gerard Meyer. Accelerated Frank–Wolfe algorithms. *SIAM Journal on Control*, 12:655–655, 1974.

Charles A. Micchelli, Jean M. Morales, and Massimiliano Pontil. Regularizers for structured sparsity. *Advances in Computational Mathematics*, 38(3):455–489, 2013.

Bamdev Mishra, Gilles Meyer, Francis Bach, and Rodolphe Sepulchre. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization*, 23(4):2124–2149, 2013.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming, Series B*, 140:125–161, 2013.

Guillaume Obozinski and Francis Bach. Convex relaxation for combinatorial penalties. Technical Report HAL 00694765, 2012.

Gabor Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of Operations Research*, 23(2):339–358, 1998.

Ting Kei Pong, Paul Tseng, Shuiwang Ji, and Jieping Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20 (6):3465–3489, 2010.

Ralph Tyrell Rockafellar and Roger J-B Wets. *Variational Analysis.* Springer, 1998.

Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. Trading accuracy for sparsity in optimization problem with sparsity constraint. *SIAM Journal on Optimization*, 20(6): 2807–2832, 2010.

Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems*, 2005.

Ambuj Tewari, Pradeep Ravikumar, and Inderjit S. Dhillon. Greedy algorithms for structurally constrained high dimensional problems. In *Advances in Neural Information Processing Systems*, 2011.

Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 6:615–640, 2010.

Paul Tseng. Approximation accuracy, gradient methods, and error bound for structured convex optimization. *Mathematical Programming, Series B*, 125:263–295, 2010.

Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2010.

Martha White, Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans. Convex multi-view subspace learning. In *Advances in Neural Information Processing Systems*, 2012.

P. Wolfe. Convergence theory in nonlinear programming. In *Integer and Nonlinear Programming.* North-Holland, 1970.

Yaoliang Yu and Dale Schuurmans. Rank/norm regularization with closed-form solutions: Application to subspace clustering. In *Conf. on Uncertainty in Artificial Intelligence*, 2011.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of The Royal Statistical Society Series B*, 68(1):49–67, 2006.

Xiaotong Yuan and Shuicheng Yan. Forward basis selection for pursuing sparse representations over a dictionary. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):3025–3036, 2013.

Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, S V N Vishwanathan, and Inderjit Dhillon. NOMAD: Nonlocking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. In *Proc. Intl. Conf. on Very Large Data Bases*, 2014.

Xinhua Zhang, Yaoliang Yu, Martha White, Ruitong Huang, and Dale Schuurmans. Convex sparse coding, subspace learning, and semi-supervised extension. In *Proc. Nat'l. Conf. Artificial Intelligence*, 2011.

Xinhua Zhang, Yaoliang Yu, and Dale Schuurmans. Accelerated training for matrix-norm regularization: A boosting approach. In *Advances in Neural Information Processing Systems*, 2012.

Yong Zhuang, Wei-Sheng Chin, Yu Juan, and Chih-Jen Lin. A fast parallel SGD for matrix factorization in shared memory systems. In *Proc. ACM Recommender System Conference*, 2013.