Scalable and Sound Low-Rank Tensor Learning

Hao Cheng[♯] **Yaoliang Yu**[*] **Xinhua Zhang**[‡] **Eric Xing**[*] **Dale Schuurmans**[†]
University of Washington[♯] CMU[*] NICTA and ANU[‡] University of Alberta[†]

Abstract

Many real-world data arise naturally as tensors. Equipped with a low rank prior, learning algorithms can benefit from exploiting the rich dependency encoded in a tensor. Despite its prevalence in low-rank matrix learning, trace norm ceases to be tractable in tensors and therefore most existing works resort to matrix unfolding. Although some theoretical guarantees are available, these approaches may lose valuable structure information and are not scalable in general. To address this problem, we propose directly optimizing the tensor trace norm by approximating its *dual* spectral norm, and we show that the approximation bounds can be *efficiently* converted to the original problem via the generalized conditional gradient algorithm. The resulting approach is scalable to large datasets, and matches state-of-the-art recovery guarantees. Experimental results on tensor completion and multitask learning confirm the superiority of the proposed method.

1 Introduction

Real-world data are complex and usually exhibit rich structures that learning algorithms can significantly benefit from. For instance, data in the vectorial form can be sparse while in the matrix form can have low rank. More general multi-way correlations are often observed in tensor form data, i.e. multi-dimensional arrays [1]. Examples include multi-channel images, video sequences, chemical compound processes, brain EEG signals, etc. Additionally, tensors are often used as important tools for modeling and estimation. Most notably, they can be applied as the high order moments in latent variable models [2] such as independent component analysis, mixture and topic models, etc. Due to the wide applicability, a lot of works have focused on factorizing, completing, and learning a tensor of interest.

The most straightforward solution to low-rank tensor decomposition is arguably alternating least squares (ALS) or block-coordinate based methods [1, 3]. With relatively high efficiency, they are generically applicable in practice. In the case of completing a $d \times d \times d$ tensor using z observed entries, the complexity is only $O(zr)$, where r is the intended rank. However, they are based on nonconvex optimization whose globally optimal solution is generally intractable to find. Therefore, the analysis of sample complexity and recovery bounds is generally hard, leaving the recent results restricted to strong assumptions such as the absence of noise or the access to the truth rank [e.g. 2, 4, 5].

A prevalent class of theoretically sound approaches are based on convex relaxations of the rank function. In low-rank matrix learning, the utilization of trace norm has achieved remarkable success since the seminal work of [6]. Extension to tensors is conceptually straightforward, based on the generic framework of atomic norm [7]. Recently, [8] provided excellent theoretical support for the tensor trace norm. However, many computational tractability properties of matrices do not carry over to tensors. For instance, the tensor rank can (far) exceed the maximum dimension, and a low-rank approximation may not even exist [9]. Indeed, it is also NP-hard to decide the tensor rank and tensor trace norm, see e.g. [10].

To work around the tractability barrier, approximations are in order. A large body of algorithms unfold the tensor into matrices, and then apply well established matrix techniques, e.g., minimizing the (matrix) trace norm as a proxy of the rank [11–16]. However, this may be unsatisfying because a genuinely "low-rank" tensor can nevertheless have large or even full rank in *all* matrix unfoldings. It also brings about scalability issues as the most efficient implementation of proximal gradient methods or ADMM costs at least $O(zd)$ time. As a result, most matricization based algorithms do not scale to large tensors.

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

The aim of this paper is therefore to design approximate models which are both scalable and theoretically sound. Instead of directly approximating the tensor trace norm, we resort to its dual norm—tensor spectral norm, because it is much easier to approximate. We provide a simple algorithm to compute the tensor spectral norm with sound approximation bounds, and its computational cost is similar to that of ALS (Section 4). A key challenge is then to utilize this approximation as an oracle, and to find a solution for the original optimization problem with suboptimality guarantee. We show that this translation can be achieved by using the recent generalized conditional gradient algorithm [17, 18], and furthermore $O(1/\epsilon)$ rates of convergence can be derived for this approximate oracle setting. Finally we study the sample complexity of the proposed algorithm in Section 5, and demonstrate that it does match the state-of-the-art bounds of matricization based models. Therefore, we obtain an algorithm which is efficient in both computation and inference. In experiments on tensor completion and multitask learning, our method significantly outperforms existing methods in generalization and efficiency.

2 Preliminaries

We first review some preliminaries. A tensor is identified as a multi-dimensional array $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_K}$, where K is the order and d_k is the dimension of the k-th mode. Without loss of generality, we assume throughout that $d_1 \geq d_2 \geq \cdots \geq d_K$. Following [1], we define the mode-k multiplication of a tensor with some matrix $U \in \mathbb{R}^{\hat{d}_k \times d_k}$ as

$$(\mathcal{A} \times_k U)_{i_1,\ldots,i_{k-1},\hat{i}_k,i_{k+1},\ldots,i_K} = \sum_{i_k=1}^{d_k} \mathcal{A}_{i_1,\ldots,i_{k-1},i_k,i_{k+1},\ldots,i_K} U_{\hat{i}_k,i_k}.$$

Note that the dimension of the k-th mode of the product $\mathcal{A} \times_k U$ changes from d_k to $\hat{d}_k$. As usual, we define the inner product of two tensors with the same size as $\langle \mathcal{A}, \mathcal{B} \rangle := \sum_{i_1,\ldots,i_K} \mathcal{A}_{i_1,\ldots,i_K} \mathcal{B}_{i_1,\ldots,i_K}$. and the induced Frobenius norm $\|\mathcal{A}\|_{\mathsf{F}} := \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$.

We can unfold (or flatten) a tensor into a 2-D matrix as follows. For any $1 \leq k \leq K$, $\mathcal{A}_{(k)} \in \mathbb{R}^{d_k \times (\prod_{j \neq k} d_j)}$, with its $\big(i_k, 1 + \sum_{j=1, j\neq k}^{K} (i_j - 1) \prod_{m=j+1, m\neq k}^{K} d_m\big)$-th entry being $\mathcal{A}_{i_1,\ldots,i_K}$. Like matrices, we can decompose a tensor into a linear combination of *primitives*. First, we define rank-1 (pure) tensors as $\mathcal{T} = \mathbf{u} \otimes \mathbf{v} \otimes \cdots \otimes \mathbf{z}$, where the K factors $\mathbf{u} \in \mathbb{R}^{d_1}, \ldots, \mathbf{z} \in \mathbb{R}^{d_K}$ form the outer product $\mathcal{T}_{i_1,i_2,\ldots,i_K} = u_{i_1} v_{i_2} \cdots z_{i_K}$, $\forall i_1, \ldots, i_K$. Then, the Candecomp-Parafac (CP) decomposition of the tensor $\mathcal{A}$ refers to

$$\mathcal{A} = \sum_{i=1}^{r} \mathbf{u}_i \otimes \mathbf{v}_i \otimes \cdots \otimes \mathbf{z}_i, \tag{1}$$

and the lowest possible value of r is called the *rank* of $\mathcal{A}$. Surprisingly, the tensor rank could well exceed any dimension d_k. In general, a well known bound states

$$\forall k, \ \ \mathrm{rank}(\mathcal{A}_{(k)}) \leq \mathrm{rank}(\mathcal{A}) \leq \textstyle\prod_{j \neq k} \mathrm{rank}(\mathcal{A}_{(j)}), \tag{2}$$

where recall that $\mathrm{rank}(\mathcal{A}_{(k)})$ is upper bounded by the k-th dimension d_k. Note that even when the tensor rank is small, the mode-k unfolding $\mathcal{A}_{(k)}$ can still have large or even full matrix rank for all k, which suggests that treating a genuine tensor as a folded matrix could lose a lot of structure.

Numerically finding a CP decomposition or the tensor rank is intractable [10]. Block coordinate descent, i.e. alternatively optimizing each factor matrix with all others fixed, can lead to a *local* optimum. We caution that when $K > 2$, deflation, i.e. finding the rank-1 factors successively, can be far from optimal [19].

3 Low-Rank Tensor Learning

Learning a low-rank tensor can be formulated as the following optimization problem:

$$\inf_{\mathcal{W}} \ \ \ell(\mathcal{W}) + \lambda \cdot \mathrm{rank}(\mathcal{W}), \tag{3}$$

where $\lambda \geq 0$ is the regularization constant. The loss function ℓ measures the discrepancy between the estimate tensor $\mathcal{W}$ and the data tensor $\mathcal{X}$ (suppressed in our notation). Some useful instantiations of ℓ can be found in Section 6, and here we use as a concrete example the tensor completion problem, with $\ell(\mathcal{W}) = \ell_{\mathrm{tc}}(\mathcal{W}) = \|\mathscr{P}(\mathcal{W} - \mathcal{X})\|_{\mathsf{F}}^2$, where $\mathscr{P} : \mathbb{R}^{d_1 \times \cdots \times d_K} \to \mathbb{R}^{d_1 \times \cdots \times d_K}$ is the (linear) sampling operator that fills unobserved entries with 0. We are particularly interested in finding a low-rank estimate, induced by penalizing the tensor rank. This bias can be beneficial in multiple ways: a). It is a valid prior in many applications; b). It leads to simpler models with potentially better generalization performance; c). It improves scalability by reducing the cost in computation and storage, as we shall see.

Although appealing in theory, the tensor rank is computationally intractable [10]. For matrices, the trace norm relaxation [6, 20] has been remarkably successful, serving as a convex surrogate of the matrix rank. It is thus natural to extend this principle to tensors, using the atomic norm framework of [7]. We start with a set of atoms (primitives), convexify it if necessary, and then construct the Minkowski gauge function as the appropriate regularizer for promoting the atoms. It is demonstrated in [7] that this yields the "best" convex regularizer in an appropriate sense.

Adapting to the low-rank tensor learning problem (3), we choose the atoms to be rank-1 tensors:

$$\mathsf{A} := \{\mathbf{u}_1 \otimes \cdots \otimes \mathbf{u}_K : \forall k, \mathbf{u}_k \in \mathbb{R}^{d_k}, \|\mathbf{u}_k\|_2 \leq 1\}. \tag{4}$$

Upon convexification we can construct the tensor trace norm [TTN, e.g. 21, 22]:

$$\|\mathcal{A}\|_{\text{tr}} := \inf\{\rho \geq 0 : \mathcal{A} \in \rho \cdot \text{conv}(\mathsf{A})\}. \quad (5)$$

Clearly (5) is a norm on $\mathbb{R}^{d_1 \times \cdots \times d_K}$, and its dual norm—tensor spectral norm (TSN)—is given by

$$\|\mathcal{A}\|_{\text{sp}} := \max_{\|\mathcal{B}\|_{\text{tr}} \leq 1} \langle \mathcal{A}, \mathcal{B} \rangle = \max_{\mathcal{B} \in \mathsf{A}} \langle \mathcal{A}, \mathcal{B} \rangle \quad (6)$$

$$= \max_{\forall k, \|\mathbf{u}_k\|_2 \leq 1} \langle \mathcal{A}, \mathbf{u}_1 \otimes \cdots \otimes \mathbf{u}_K \rangle . \quad (7)$$

Specializing to matrices ($K = 2$) we recover the familiar trace norm and the usual spectral norm.

3.1 Main formulation

We can now obtain a convex relaxation of (3) by replacing the tensor rank with the TTN,

$$\min_{\mathcal{W}} \ \ell(\mathcal{W}) + \lambda \cdot \|\mathcal{W}\|_{\text{tr}} . \quad (8)$$

Assuming ℓ is convex, (8) is a convex problem. This convexity appears to be a significant step towards tractability, and numerous efficient algorithms have been proposed for the matrix case ($K = 2$). However, almost no[1] significant effort has been made for $K \geq 3$. [8], for instance, proved some theoretical advantages of using TTN, but no practical algorithm was provided. The underlying reason is that TTN, albeit being convex, is itself intractable [10]! Consequently, one might wonder what is the benefit of replacing the original intractable problem (3) with another convex but still intractable one (8)? The answer in short is: the latter provides more favorable approximation bounds and facilitates more effective algorithms in practice—not all intractable problems are equally "hard".

Indeed, it is a standard practice to attack NP-hard problems with *approximate* guarantees (additive or multiplicative). We recall the definition of a multiplicative α-approximate algorithm:

Definition 1. *Consider a class of optimization problems $0 \leq \texttt{OPT}_f = \inf_{\mathbf{w}} f(\mathbf{w})$, where f belongs to some function class $\mathcal{F}$. Fix $\alpha > 0$. An algorithm is α-approximate if for all $f \in \mathcal{F}$ it always outputs $\mathbf{w}_f^*$ such that $f(\mathbf{w}_f^*) \leq \frac{1}{\alpha}\texttt{OPT}_f$.*

There is a hierarchy of NP-hard problems that can be solved approximately: some admits polynomial time approximation (e.g. Knapsack); some admits a constant approximation (e.g. max-cut); and still some has guarantees depending on the problem size, which is the case in (6) as will be shown below. We will develop an efficient α-approximate algorithm for solving the relaxation (8). The convexity in (8), although not leading directly to tractability, will play a key role in our reasoning and development.

[1] After this work was completed, N. Rao kindly brought to our attention a related work [23], which also applied an improved conditional gradient to tensor completion, but no approximation guarantee was provided.

4 Efficient Approximate Algorithm

The most straightforward attempt to solve (8)is to approximate the TTN. However, different from the matrix trace norm, it is not the sum of (natural) "singular values", and the definition in (5) does not provide an explicit form that allows convenient and analyzable approximation. By contrast, the definition of TSN in (6) exhibits much "simpler" and more explicit structure (though also NP-hard), where the constraints decouple over modes with variables $\mathbf{u}_k$ lying in standard Euclidean balls. So a natural strategy is to first approximate the TSN with good guarantee, and then design optimization algorithms that convert this bound to the optimization bound of the original problem (8).

4.1 Approximating the Tensor Spectral Norm

To design an efficient algorithm for approximating TSN, we first resort to a celebrated result from convex geometry [24]. Let B_2^d be the Euclidean norm ball in a d dimensional space (d is a superscript in B_2^d). We can find in polynomial time (at most) d points $\{\mathbf{p}_i\}$, such that their convex hull P^d (a polytope) satisfies $\mathsf{P}^d \subseteq \mathsf{B}_2^d \subseteq c\sqrt{d}\mathsf{P}^d$, where c is some universal constant. In particular we can use the unit L_1 ball B_1^d: clearly $\mathsf{B}_1^d \subseteq \mathsf{B}_2^d \subseteq \sqrt{d}\mathsf{B}_1^d$. By counting the volume, it can be proved that the factor $\sqrt{d}$ here is the best possible [25]. Specializing to the TSN, we simply replace each Euclidean ball constraint with an L_1 ball, and evaluate the inner product in (6) at each of the vertices of the polytope. Since the matrix trace norm is tractable, we need only execute this polytopal approximation for the last $K - 2$ modes, i.e., solving the approximation

$$\max \langle \mathcal{A}, \mathbf{u}_1 \otimes \cdots \otimes \mathbf{u}_K \rangle , \text{ s.t. } \forall i \geq 3, \ \mathbf{u}_i \in \mathsf{P}^{d_i}, \quad (9)$$

where also $\mathbf{u}_1 \in \mathsf{B}_2^{d_1}, \mathbf{u}_2 \in \mathsf{B}_2^{d_2}$.

For each of the $\prod_{k=3}^K d_k$ vertices $\mathbf{p}^3 \otimes \cdots \otimes \mathbf{p}^K$, we evaluate the matrix spectral norm $\|\mathcal{A} \times_3 \mathbf{p}^3 \cdots \times_K \mathbf{p}^K\|_2$ and take the maximum among them. Since the vertices of L_1 balls are canonical basis vectors, this amounts to taking each matrix *slice* of $\mathcal{A}$, hence we call it the ***slicing* approach**. It yields the optimal solution for (9), and translates to an $\alpha = \prod_{k=3}^K \sqrt{1/d_k}$ approximate solution for the TSN (6). The overall computational cost is $O(\prod_{k=1}^K d_k)$, which can be further reduced after embarrassing parallelization. This is much faster than existing matricization approaches [11–16], which cost $O(\prod_{k=1}^K d_k \sum_{k=1}^K d_k)$ due to multiple *full* matrix SVDs.

A number of approximations of TSN is available in the literature [e.g., 26], all leading to the same α guarantee here. We can easily combine them by picking the best one. Although the approximation ratio depends on

the problem size, it is only a worst case bound which is likely not tight. Perhaps more surprisingly, this is (up to logarithmic factors) the best result that we are aware of, despite being derived from such a straightforward polytopal approximation (more delicate tradeoff between computational cost and approximation quality is available in Appendix B). Significantly new ideas seem necessary for further improvement. In practice it is much more effective to perform block coordinate ascent (BCA) over (6) through all $\mathbf{u}_k$, initialized with the solution of (9). This is the strategy we use in our experiment. However, we also observed that randomly initializing BCA did not yield a result as good—a provably good initialization such as using (9) turns out to be crucial.

Computational efficiency. A key advantage of this scheme is that the computational efficiency can be further improved by utilizing the sparsity in $\mathcal{A}$. Suppose $\mathcal{A}$ has z nonzeros entries. Then the optimization in (9) can be solved in $\tilde{O}(z)$ time (independent of dimension), because the cost of computing the spectral norm of a matrix is linear in the number of nonzero entries. The subsequent BCA can also be performed in $\tilde{O}(z)$ time per iteration, and [27] proposed a highly optimized algorithm.

4.2 Conditional Gradient with Approximate Spectral Norm

Equipped with an effective approximation of TSN, the challenge remains to design an *efficient* optimization algorithm which leverages this approximation guarantee and finds an α-approximate solution to the original TTN regularized problem (8). We discover that this conversion can be accomplished by the recently developed generalized conditional gradient [GCG, 17, 18], which extends the traditional conditional gradient algorithm [28] to the (unconstrained) regularized problem. Technically, we further assume ℓ is smooth (i.e. its gradient is Lipschitz continuous),

The basic GCG algorithm in [17] successively linearizes the loss ℓ at the current iterate $\mathcal{W}_t$, finds an update direction in the unit ball of TTN:

$$\textbf{Oracle:} \quad \mathcal{Z}_t \in \operatorname{argmin}_{\mathcal{Z}:\|\mathcal{Z}\|_{\mathrm{tr}}\leq 1} \langle \mathcal{Z}, \nabla \ell(\mathcal{W}_t)\rangle , \quad (10)$$

and update by $\mathcal{W}_{t+1} = (1-\eta_t)\mathcal{W}_t + \eta_t\beta_t\mathcal{Z}_t$, with some step size $\eta_t \in [0,1]$ and scaling factor

$$\beta_t = \operatorname{argmin}_{\beta\geq 0} \ \ell((1-\eta_t)\mathcal{W}_t + \eta_t\beta\mathcal{Z}_t) + \lambda\cdot\eta_t\beta. \quad (11)$$

It is crucial to observe that the oracle problem (10) is exactly the same as the optimization involved in the definition of TSN (6). Therefore the above approximation of TSN readily provides an α-approximate solution to the oracle problem here. So the final question to resolve is whether or how GCG is "robust" against such approximate oracles. Theorem 1 provides an encouraging answer.

Theorem 1. *Let $\ell \geq 0$ be convex, smooth, and have bounded sublevel sets. Denote $f(\mathcal{W}) = \ell(\mathcal{W}) + \lambda\cdot\|\mathcal{W}\|_{\mathrm{tr}}$. Suppose in each iteration t, we find $\mathcal{Z}_t$ that solves the oracle (10) α-approximately, Then for all $\mathcal{W}$ and for all $t \geq 1$, running GCG with $\eta_t = \frac{2}{t+2}$ leads to $f(\mathcal{W}_t) - \frac{f(\mathcal{W})}{\alpha} \leq \frac{2C}{t+3}$, where C is some constant that does not depend on t or α.*

The proof is relegated to Appendix C. Theorem 1 ensures that the GCG procedure is (asymptotically) α-approximate when equipped with the α-approximate oracle in (10). For instance, if $\alpha = 1/2$, then GCG is at most twice worse than the (possibly intractable) optimum. We note that [29] considered a different multiplicative approximation, requiring in each step an approximate solution of $\min_{\|\mathcal{Z}\|\leq 1}\langle\mathcal{Z}, \nabla\ell(\mathcal{W}_t)\rangle - \langle\mathcal{W}_t, \nabla\ell(\mathcal{W}_t)\rangle$. Unfortunately, this is usually hard to achieve due to the second changing term. On the other hand, [29] were still able to prove *exact* optimality while Theorem 1 here yields only an approximate guarantee. We noted in passing that a similar strategy appeared independently in [30] for hard matrix factorizations. In Appendix C, we further show that Theorem 1 can be generalized by replacing TTN with any convex positive homogeneous function.

Note this conversion is enabled by the convexity of f, which is facilitated by the use of TTN despite its own intractability. The polytopal approximation technique used for TSN, however, cannot be directly applied to the original problem (8), or in other numerical algorithms (e.g., proximal gradient). It is enabled by the linearization step (10) in GCG, where constraints in (6) decouple over modes and Euclidean balls can be approximated analytically and efficiently. Happily, any improvement on computing TSN immediately translates to the same amount for the tensor learning problem (8).

4.3 Local acceleration

A very efficient acceleration strategy was proposed in [17] for matrices. Pleasantly, we can extend their trick to the new tensor setting, starting with a variational form for the tensor trace norm:

Theorem 2 (Variational formula)**.** *Fix $\mathcal{A} \in \mathbb{R}^{d_1\times\cdots\times d_K}$ and let $t \geq \prod_{k=1}^K d_k$. Then,*

$$\|\mathcal{A}\|_{\mathrm{tr}} = \min\left\{\Sigma_{i=1}^t \|\mathbf{u}_i\|_2\cdots\|\mathbf{z}_i\|_2\right\} \quad (12)$$

$$= \min\left\{\tfrac{1}{K}\Sigma_{i=1}^t \|\mathbf{u}_i\|_2^K + \cdots + \|\mathbf{z}_i\|_2^K\right\}, \quad (13)$$

where the minimum is taken w.r.t. all factorizations $\mathcal{A} = \sum_{i=1}^t \mathbf{u}_i\otimes\cdots\otimes\mathbf{z}_i$, $\mathbf{u}_i \in \mathbb{R}^{d_1}, \ldots, \mathbf{z}_i \in \mathbb{R}^{d_K}$.

Algorithm 1: α-Approximate GCG for solving low-rank tensor learning (8) with local search

1 Set $\mathcal{W}_0 = \mathbf{0}$, $U_0 = \ldots = Z_0 = [\,]$, $s_0 = 0$.
2 **for** $t = 1, 2, \ldots$ **do**
3 find $(\mathbf{u}_t, \ldots, \mathbf{z}_t)$ that yields α-approximate spectral norm of $-\nabla\ell(\mathcal{W}_{t-1})$; // §4.1
4 $(a_t, b_t) \leftarrow \operatorname{argmin}_{a\geq 0, b\geq 0} \ell(a \cdot \mathcal{W}_{t-1} + b \cdot \mathbf{u}_t \otimes \cdots \otimes \mathbf{z}_t) + \lambda(a \cdot s_t + b)$; // §4.2
5 $U_{\text{init}} \leftarrow (\sqrt[K]{a_t} U_{t-1}, \sqrt[K]{b_t}\mathbf{u}_t), \ldots, Z_{\text{init}} \leftarrow (\sqrt[K]{a_t} Z_{t-1}, \sqrt[K]{b_t}\mathbf{z}_t)$;
6 find $U_t, \ldots, Z_t$ such that $\mathfrak{F}_t(U_t, \ldots, Z_t) \leq \mathfrak{F}_t(U_{\text{init}}, \ldots, Z_{\text{init}})$; // §4.3
7 $\mathcal{W}_t \leftarrow \mathcal{I}_t \times_1 U_t \cdots \times_K Z_t$; // Implicitly maintained in implementation
8 $s_t \leftarrow \frac{1}{K}\sum_{i=1}^t \|(U_t)_{:i}\|_2^K + \cdots + \|(Z_t)_{:i}\|_2^K$; // Upper bound on TTN, Thm 2

The first equality is well-known thanks to Grothendieck's work on the tensor product of Banach spaces. Our proof in Appendix C gives an explicit bound on t, the number of rank-1 factors. In Appendix D we argue that the above variational forms should be preferred than some existing ones.

We now make use of Theorem 2 to further accelerate GCG. The entire procedure for solving the tensor learning problem (8) is summarized in Algorithm 1. After the $(t-1)$-th iteration of GCG (line 7), we get an explicit low-rank representation of the current iterate $\mathcal{W}_{t-1} = \sum_{i=1}^{t-1} \mathbf{u}_i \otimes \cdots \otimes \mathbf{z}_i$. Instead of using the fixed step size $\eta_t = \frac{2}{t+2}$ to combine $\mathcal{W}_{t-1}$ and the newly found atom $\mathbf{u}_t \otimes \cdots \otimes \mathbf{z}_t$ (line 3) as in Theorem 1, we optimize it along with the scaling factor β (line 4, after change of variable to a, b). To improve convergence, we locally minimize the surrogate function (line 6):

$$\mathfrak{F}_t(\tilde{U}, \ldots, \tilde{Z}) := \ell(\textstyle\sum_{i=1}^t \tilde{U}_{:i} \otimes \cdots \otimes \tilde{Z}_{:i}) + \frac{\lambda}{K}\sum_{i=1}^t \|(\tilde{U})_{:i}\|_2^K + \cdots + \|(\tilde{Z})_{:i}\|_2^K, \quad (14)$$

simply replacing the intractable TTN with the variational form (12). We expect the surrogate problem (with fixed t) to be reasonably close to the original problem (8), hence a local (smooth) unconstrained minimization of (14) should improve the iterate $\mathcal{W}_{t-1}$ of GCG. Adopting proper initialization to ensure monotonic progress (line 5), the convergence guarantee in Theorem 1 is retained.

Although local optimization does not improve convergence rate in the worst case, empirically it much accelerates convergence by allowing the factors found in past iterations to be re-optimized in the context of new ones. This provides much more freedom than merely optimizing their weights as in the vanilla GCG. Faster convergence also results in simpler models which improves generalization performance in general. Interestingly, GCG with local search bears much resemblance to ALS, but with the key difference being that the number of factors is not fixed, and new factors are incorporated in a greedy fashion. This potentially alleviates the issue of local optimum suffered by ALS.

Computational efficiency. Note in Algorithm 1, $\mathcal{W}_t$ is always represented via CP decomposition $\mathcal{W}_t = \sum_{i=1}^t \mathbf{u}_i \otimes \cdots \otimes \mathbf{z}_i$, which is cheap to maintain and underpins the seamless integration of GCG and local search. The most costly step in Algorithm 1 is computing the gradients $\frac{\partial \ell}{\partial \mathbf{u}_i}, \ldots, \frac{\partial \ell}{\partial \mathbf{z}_i}$ given $\nabla\ell(\mathcal{W}_{t-1})$, for which a naive approach takes $O(K\prod_{k=1}^K d_k)$ time. We show in Appendix E that this cost can be reduced to $O(\prod_{k=1}^K d_k)$ via dynamic programming.

A key merit of GCG lies in its ability to effectively utilize both the sparsity in tensor completion, and the low rank of the optimal solution. Assuming the tensor has z nonzeros, the approximate oracle costs $O(z)$ time, and at iteration t the gradients in local search cost $O(zt)$ [27], on par with ALS. t is typically low since the optimal solution has low rank, allowing GCG to converge fast. As a result, its overall cost is much cheaper than matricization based methods, which require $O(z\sum_k d_k)$ computation for singular value thresholding on a sparse matrix that is *not* necessarily low-rank in intermediate steps. Finally, the space cost is only $O(z + t\sum_k d_k)$, much less than the $O(\prod_k d_k)$ of matricization approaches.

5 Sample Complexity Comparisons

Arguably, for any tractable approximation of TSN, we can directly use its dual in the optimization objective (8), in lieu of TTN. This raises the question of why not directly analyze their sample complexity, and why do we analyze its deviation from TTN. This is because using TTN as a gold standard allows us to infer *suggestively* that a solution based on tighter approximation of TSN will lead to as good or better sample complexity. For example, once we establish the sample complexity for the slicing algorithm (below), then intuitively it should not deteriorate when conjoined with BCA, which improves the approximation of TSN. We leave a rigorous analysis for future work.

To give some theoretical underpinnings, let us consider

concretely the tensor completion problem:

$$\min_{\mathcal{W}} \sum_{i=1}^{q} \|\mathscr{L}_i(\mathcal{W})\|_{(i)} \quad \text{s.t.} \quad \langle \mathcal{A}_i, \mathcal{W} \rangle = b_i, \quad (15)$$

where each $\mathcal{A}_i$ is a Gaussian random tensor, e.g., the entries of $\mathcal{A}_i$ are i.i.d. standard normal, and $b_i = \langle \mathcal{A}_i, \mathcal{W}_0 \rangle$ for some unknown low-rank tensor $\mathcal{W}_0$. For simplicity, let $d_1 = \cdots = d_K = d$. The linear maps $\mathscr{L}_i$ are from $\mathbb{R}^d \times \cdots \times \mathbb{R}^d$ to $\mathbb{R}^{m_i}$, and the norm $\|\cdot\|_{(i)}$ is defined on the corresponding range space $\mathbb{R}^{m_i}$. In particular, we will compare the following two approaches:

- Matricization: $q = K$, and for $i = 1, \ldots, q$, $\mathscr{L}_i : \mathbb{R}^d \times \cdots \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^{d^{K-1}}$, $\mathcal{W} \mapsto \mathcal{W}_{(i)}$, and $\|\cdot\|_{(i)}$ is the matrix trace norm on $\mathbb{R}^d \times \mathbb{R}^{d^{K-1}}$.
- Slicing: $q = d^{K-2}$, and for $i = 1, \ldots, q$, $\mathscr{L}_i : \mathbb{R}^d \times \cdots \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$, $\mathcal{W} \mapsto \mathcal{W}_{:,:,i_3,\ldots,i_K}$, and $\|\cdot\|_{(i)}$ is the matrix trace norm on $\mathbb{R}^d \times \mathbb{R}^d$, where $i = \sum_{j=3}^{K} d^{j-3} i_j$ is the d-ary expansion.

The matricization case has been widely studied in [e.g. 11–15, 31], while the slicing case does not appear to be widely appreciated. We used slicing in Section 4.1 to approximate the TSN.

We are interested in deciding the least number of observations m that still allows recovering $\mathcal{W}_0$ (with high probability) by solving (15). Obviously, the recovery is successful iff $\mathcal{W}_0$ is the unique minimizer of (15). Let us recall the following result:

Theorem 3 ([31, 32]). *Suppose $\mathcal{W}_0 \neq \bigcap_{i \leq q} \operatorname{null}(\mathscr{L}_i)$. For each i, define $L_i = \sup_{0 \neq \mathbf{x} \in \mathbb{R}^{m_i}} \frac{\|\mathbf{x}\|_{(i)}}{\|\mathbf{x}\|_{\mathsf{F}}}$. Set $\kappa_i = \frac{d^K \|\mathscr{L}_i(\mathcal{W}_0)\|_{(i)}^2}{L_i^2 \|\mathscr{L}_i\|^2 \|\mathcal{W}_0\|_{\mathsf{F}}^2}$. Then for $m \leq O(\min_i \kappa_i)$, with high probability $\mathcal{W}_0$ is not the unique minimizer of (15), while for $m \geq O(\max_i \kappa_i)$, with high probability $\mathcal{W}_0$ is the unique minimizer.*

We apply this theorem to the matricization and the slicing approach above. Note first that in both cases $L_i = \sqrt{d}$ and the operator norm $\|\mathscr{L}_i\| = 1$. Thus, for successful recovery we need to compare:

$$\frac{d^{K-1}}{\|\mathcal{W}_0\|_{\mathsf{F}}^2} \max_i \|(\mathcal{W}_0)_{(i)}\|_{\mathrm{tr}}^2 \text{ vs. } \frac{d^{K-1}}{\|\mathcal{W}_0\|_{\mathsf{F}}^2} \max_i \|(\mathcal{W}_0)_{:,:,i_3,\ldots,i_K}\|_{\mathrm{tr}}^2.$$

Clearly, both are on the same order $O(d^{K-1})$ but we always have $\max_i \|(\mathcal{W}_0)_{(i)}\|_{\mathrm{tr}}^2 \geq \max_i \|(\mathcal{W}_0)_{:,:,i_3,\ldots,i_K}\|_{\mathrm{tr}}^2$, i.e., the slicing approach (latter) always has a smaller problem dependent constant. This can make a big difference: We could obtain the correct order $O(d^{K-1})$ by using say even the ℓ_1 norm with $\mathscr{L}_i : \mathcal{W} \mapsto \mathcal{W}_{:,i_2,\ldots,i_K}$. Only the problem dependent term inside the max function reflects how different norms match the problem structure (e.g. low-rank). We note that by using a more balanced unfolding, [31] was able to improve the order $O(d^{K-1})$, at the expense of potentially increasing the problem dependent term. On the other hand, the slicing norm is particularly computational friendly: it can be embarrassingly parallelized. Moreover, when $\mathcal{A}_i$ are sampling operators, the problem becomes completely separable and we need only complete each slice independently.

We wish to point out a second difference between the matricization approach and the proposed α-approximate GCG with the slicing norm: The latter not only recovers a low-rank tensor, it also finds a CP decomposition that certifies the low-rank assumption. In contrast, the matricization approach can only recover the tensor, but it never explicitly finds a usable CP decomposition.

A large body of works deal directly with the tensor rank function, e.g. [1, 33, 34]. Nonconvex optimization tools are employed to get a local minima, whose formal guarantee, to our best knowledge, is largely unknown. Under the restrictive orthogonal CP assumption (which may not exist in general), [4] justified an extended form of power iteration while [2] proved strong guarantees by restricting the rank. It may be possible to strengthen our results by combining ideas from them.

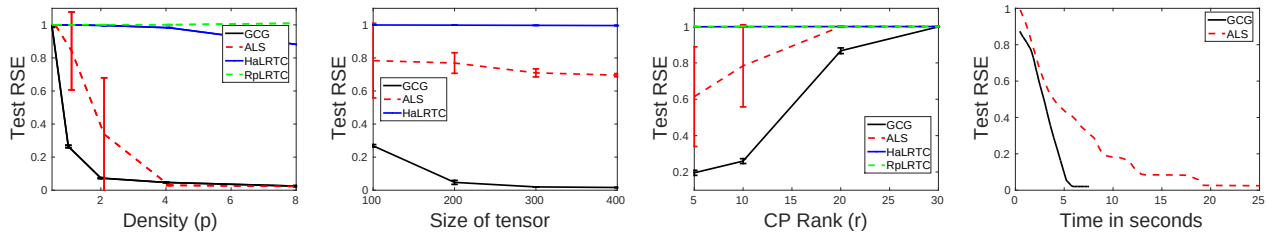
6 Experiments

We study the empirical performance of Algorithm 1 on two low-rank learning problems: tensor completion and multitask learning. Local optimization (14) in GCG was solved by LBFGS. ALS was tested in all cases, with gradients computed by the state-of-the-art algorithm for efficiency [27].

6.1 Low-Rank Tensor Completion

Two state-of-the-art algorithms were used for comparison: HaLRTC [14] which uses the weighted matrix trace norm; and a tighter relaxation by [35], referred to as RpLRTC. [14, Eq 11] also considered a non-convex Parafac regularizer that is similar to the variational form of tensor trace norm. Since its performance has been shown inferior to HaLRTC, we will not include it in the comparison. Also note the analysis of [4] does not consider our noisy setting. For all methods, we selected the value of λ via a validation set, which always consisted of 10% of the whole dataset. The rank in ALS was set to the true value on synthetic data, and was selected via a validation set on real data.

Synthetic data. We first generated a 3-order tensor $\mathcal{Z}^0 \in \mathbb{R}^{d \times d \times d}$ using the CP decomposition (1), where $\mathbf{u}_i, \mathbf{v}_i, \mathbf{z}_i$ were drawn i.i.d. from the standard normal



(a) RSE vs p (train%) (b) RSE vs d (tensor size) (c) RSE vs r (CP rank) (d) RSE vs training time

Figure 1: Test RSE on synthetic data generated by CP decomposition. RpLRTC is not included in (b) because it gets very slow when $d \geq 200$.

distribution. So $\text{rank}(\mathcal{Z}^0) \leq r$ and we loosely call r the rank. $\mathcal{Z}_{i,j,k} := (\mathcal{Z}^0_{i,j,k} - \text{mean}(\mathcal{Z}^0)) \ /(d^{1.5}\text{std}(\mathcal{Z}^0)) + \xi_{i,j,k}$, where mean and std stand for the mean and standard deviation, respectively, and $\xi_{i,j,k}$ are i.i.d. Gaussians with zero mean and variance 0.1^2. We randomly picked $p\%$ elements of $\mathcal{Z}$ as observations for training, and the reconstructed tensor $\mathcal{W}$ was compared with $\mathcal{Z}^0$ on the rest $(90-p)\%$ entries. The whole process was repeated 10 times, and we report the average of $\text{RSE} := \left\|\mathcal{W}_{\text{te}} - \mathcal{Z}^0_{\text{te}}\right\|_{\text{F}} / \left\|\mathcal{Z}^0_{\text{te}}\right\|_{\text{F}}$ [14].

Figures 1(a) to 1(c) show the test RSE as a function of density p of the training data, the size d, and the CP rank r respectively. In all plots, parameters not varying in the x-axis were set to the default values: $p = 1$, $r = 10$, and $d = 100$. Clearly, in all cases GCG yields significantly more accurate completions than ALS, HaLRTC, and RpLRTC. Fixing d and r, GCG clearly outperforms when the tensor is sparse ($p = 1$ or 2). In real datasets such as Yelp and Nell below, less than 1% entries are observed, and therefore it is indeed an interesting regime. ALS exhibits higher variance suggesting susceptibility to local minima. With fixed p and r, increasing the size of tensor allows both ALS and GCG to reduce test RSE. But it benefits GCG much more thanks to its convexity. Finally, increasing the CP rank makes the problem challenging for all methods. This is reasonable because the sample size is fixed. Overall, the results confirm higher faithfulness of GCG in low-rank tensor completion than nonconvex methods and matricization based regularizers when the tensor is sparse.

Computational time is another key aspect of comparison. To make the comparison fair, we set $d = 100, p = 8, r = 10$ so that ALS and GCG achieve similar RSE. HaLRTC and RpLRTC are not included because they scale much more poorly. Figure 1(d) shows that GCG is much more efficient than ALS in reducing the test RSE.

Image inpainting. Images are naturally represented as a "width (259)" × "height (247)" × "RGB (3)" tensor. We used an image of facade [14] and randomly sampled $p\%$ pixels from the original color image for training. All pixel values were rescaled to $[0, 1]$. In addition to the competing methods used above, we also compared with the latent Schatten norm [15] as the dataset does not necessarily have low CP rank any more. Figure 2 reports the RSE on the remaining $(90-p)\%$ pixels with varied p. Clearly, GCG is superior in propagating global structure from a small number of observed pixels. Again ALS exhibits high variance, probably due to local minima.

Multiway recommendation. We next study tensor completion for recommendation systems, and the focus is on large-scale real data. The Yelp dataset is from the Yelp Data Challenge [36] which contains (user, items, word) triples for business reviews. It is sized $46K \times 12K \times 85K$ with 9.9 million nonzeros. The Nell dataset is from [37] which consists of (noun phrase 1, context, noun phrase 2) triples. It is sized $12K \times 9K \times 29K$ with 77 million nonzeros, and we randomly subsampled 10% for experiment. None of the matricization based methods can scale to such large datasets, and therefore we only compared GCG with ALS. We varied the number of training (observed) entries, used another 10% entries for validation, and tested on the rest entries. The random partition was repeated for 10 times, and the mean test RSE is presented in Figure 3 and 4. For a range of training set size, GCG yields significantly lower error than ALS.

6.2 Multitask Learning

Restaurant recommendation [13]. The tasks are to predict the rating that a restaurant would receive from each of the $I_2 = 138$ customers, in $I_3 = 3$ aspects (food, service, overall). So there are $I_2 \times I_3$ tasks, with each task employing a $I_1 = 45$ dimensional weight vector. The $I_1 \times I_2 \times I_3$ weight tensor is assumed to have low rank. Each of the 3483 data points is a restaurant represented by an I_1 dimensional feature vector (e.g. location and cuisine type), as well as its

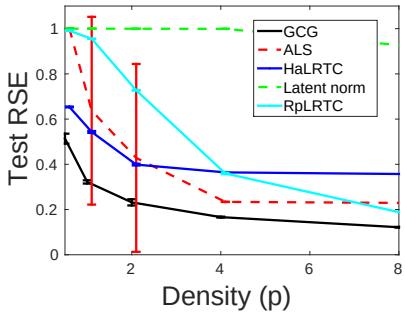


Figure 2: Image inpainting

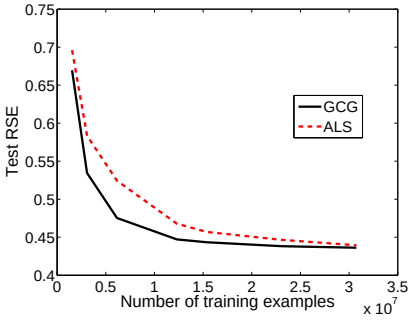


Figure 3: Test RSE on Yelp

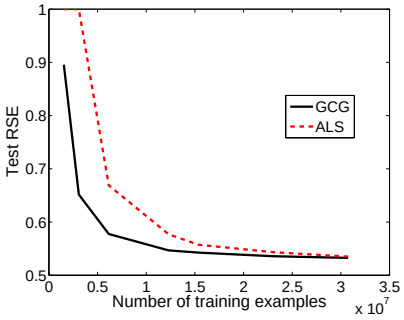


Figure 4: Test RSE on Nell

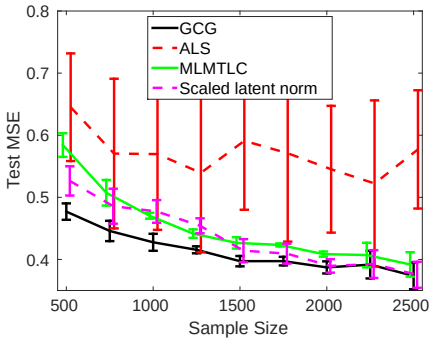


Figure 5: Restaurant rating prediction

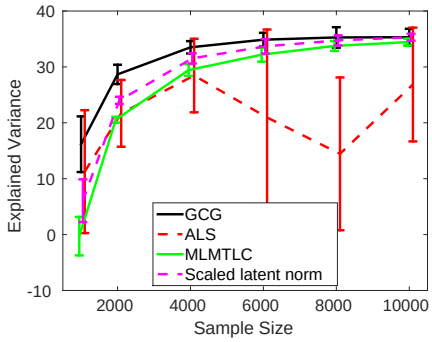


Figure 6: School grade prediction

$I_2 \times I_3$ number of ratings. Predicting these ratings constitutes $I_2 \times I_3$ regression problems, each with an ℓ_2 loss.

GCG was compared with ALS, a convex multilinear multitask learning model (MLMTL-C) [13], and scaled latent trace norm proposed recently in [38]. They have been shown to predict more accurately than a number of other multitask algorithms (hence not included here). We randomly sampled m ratings (across all tasks) for training, 250 ratings for validation (used by GCG only), and the rest for testing. All competing algorithms selected models by optimizing the test mean squared error (MSE), an extra advantage that was not available to GCG. Figure 5 plots the test MSE as a function of m, averaged over 20 random repetitions. Clearly, GCG yields significantly lower test error over a range of training set sizes, with the edge diminishing as the sample size grows. It is also more efficient. For example, it took 0.5 seconds for GCG to train on 1500 examples, while MLMTL-C and scaled latent norm spent about 9 seconds and 12 seconds respectively.

School grade prediction. Following [38], we used multitask learning to model the Inner London Education Authority dataset. It records the score of 15,362 students at 139 schools over 3 years, with each student represented by 24 attributes. The weight tensor is $24 \times 139 \times 3$, and following [38] we measured the percentage of explained variance: 100·(1 - (test MSE) / (variance of score)). Model selection in GCG was based on a validation set of 1000 samples, while all competing algorithms were allowed to select models by directly optimizing the test measure. As shown in Figure 6, GCG again achieves the highest performance, and is particularly advantageous when the sample size is small (2000 to 6000). The results of MLMTL-C and scaled latent norm recover those in [38].

7 Conclusion

Learning a low-rank tensor is an important task in many real-world applications. In this work we consider the genuine tensor trace norm regularization, as a proxy of the nonconvex tensor rank function. Building on a simple polytopal approximation of the dual spectral norm, we demonstrate how to solve the tensor trace norm problem using the recent GCG algorithm. We establish the $O(1/t)$ rate of convergence of GCG with such an approximate dual spectral norm. Further accelerations using an unconstrained smooth surrogate is discussed. Compared to matricization approaches, our algorithm achieves the same order of sample complexity but with potentially a smaller problem dependent constant. Computationally, our algorithm benefits from both the sparse and low-rank structure hence scales much better. Experiments on a variety of real datasets confirm the effectiveness of our algorithm.

References

[1] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.

[2] A. Anandkumar, R. Ge, and M. Janzamina. Guaranteed non-orthogonal tensor decomposition via alternating rank-1 updates, 2014. ArXiv:1402.5180v2.

[3] M. Mørup and L. K. Hansen. Automatic relevance determination for multi-way models. *Journal of Chemometrics*, 23(7-8):352–363, 2009.

[4] P. Jain and S. Oh. Provable tensor factorization with missing data. *In NIPS*. 2014.

[5] V. Kuleshov, A. Chaganty, and P. Liang. Tensor factorization via matrix factorization. *In AISTATS*. 2015.

[6] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2009.

[7] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S.Willsky. The convex geometry of linear inverse problems. *Foundations of Computational Mathematics*, 12(6):805–849, 2012.

[8] M. Yuan and C.-H. Zhang. On tensor completion via nuclear norm minimization. *Foundations of Computational Mathematics*, to appear.

[9] V. D. Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.

[10] C. J. Hillar and L.-H. Lim. Most tensor problems are NP-hard. *Journal of ACM*, 60(6):45:1–45:39, 2013.

[11] M. Signoretto, L. D. Lathauwer, and J. A. K. Suykens. Nuclear norms for tensors and their use for convex multilinear estimation. *Tech. Rep. 10-186*, ESAT-SISTA, K. U. Leuven, 2010.

[12] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27:1–19, 2011.

[13] B. Romera-Paredes, M. Aung, N. Bianchi-Berthouze, and M. Pontil. Multilinear multitask learning. *In ICML*. 2013.

[14] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013.

[15] R. Tomioka and T. Suzuki. Convex tensor decomposition via structured schatten norm regularization. *In NIPS*. 2013.

[16] D. Goldfarb and Z. Qin. Robust low-rank tensor recovery: Models and algorithms. *SIAM Journal on Matrix Analysis and Applications*, 35:225–253, 2014.

[17] X. Zhang, Y. Yu, and D. Schuurmans. Accelerated training for matrix-norm regularization: A boosting approach. *In NIPS*. 2012.

[18] Z. Harchaoui, A. Juditsky, and A. Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152:75–112, 2015.

[19] A. Stegeman and P. Comon. Subtracting a best rank-1 approximation may increase tensor rank. *Linear Algebra and its Applications*, 433:1276–1300, 2010.

[20] M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. *In American Control Conference*, pp. 4734–4739. 2001.

[21] L.-H. Lim and P. Comon. Blind multilinear identification. *IEEE Transactions on Information Theory*, 60(2):1260–1280, 2014.

[22] H. Derksen. On the nuclear norm and the singular value decomposition of tensors. *Foundations of Computational Mathematics*, to appear.

[23] N. Rao, P. Shah, and S. Wright. Forward - backward greedy algorithms for atomic norm regularization, 2014. ArXiv:1404.5692.

[24] M. Kochol. Constructive approximation of a ball by polytopes. *Math Slovaca*, 44(1):99–105, 1994.

[25] I. Bàràny and Z. Füredi. Approximation of the sphere by polytopes having few vertices. *Proceedings of the American Mathematical Society*, 102(3):651–659, 1988.

[26] Z. Li, S. He, and S. Zhang. *Approximation Methods for Polynomial Optimization: Models, Algorithms, and Applications*. Springer, 2012.

[27] J. H. Choi and S. Vishwanathan. Dfacto: Distributed factorization of tensors. *In NIPS*. 2014.

[28] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[29] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-Coordinate Frank-Wolfe optimization for structured svms. *In ICML*. 2013.

[30] F. Bach. Convex relaxations of structured matrix factorizations, 2013. HAL:00861118.

[31] C. Mu, B. Huang, J. Wright, and D. Goldfarb. Square deal: Lower bounds and improved relaxations for tensor recovery. *In ICML*. 2014.

[32] D. Amelunxen, M. Lotz, M. B. McCoy, and J. A. Tropp. Living on the edge: phase transitions in convex programs with random data. *Information and Inference*, 3(3):224–294, 2014.

[33] J. Nie and L. Wang. Semidefinite relaxations for best rank-1 tensor approximations. *SIAM Journal on Matrix Analysis and Applications*, 35(3):1155–1179, 2014.

[34] B. Jiang, S. Ma, and S. Zhang. Tensor principal component analysis via convex optimization. *Mathematical Programming*, 150:423–457, 2015.

[35] B. Romera-Paredes and M. Pontil. A new convex relaxation for tensor completion. *In NIPS*. 2013.

[36] Https://www.yelp.com/dataset_challenge/dataset.

[37] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos. Gigatensor: Scaling tensor analysis up by 100 times - algorithms and discoveries. *In KDD*. 2012.

[38] K. Wimalawarne, M. Sugiyama, and R. Tomioka. Multitask learning meets tensor factorization: task imputation via convex optimization. *In NIPS*. 2014.

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

[40] R. Bro. PARAFAC: Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38:147–171, 1997.

[41] S. Engelen, S. Frosch, and B. Jorgensen. A fully robust parafac method for analyzing fluorescence data. *Journal of Chemometrics*, 23(3-4):124–131, 2009.