EFFICIENT ALGORITHMS FOR 'UNIVERSALLY' CONSTRAINED MATRIX AND TENSOR FACTORIZATION

Kejun Huang, Nicholas D. Sidiropoulos *

Dept. of ECE, Univ. of Minnesota
Minneapolis, MN 55455, USA

Athanasios P. Liavas†

Dept. of ECE, Technical Univ. of Crete
Chania 73100, Greece

ABSTRACT

We propose a general algorithmic framework for constrained matrix and tensor factorization, which is widely used in unsupervised learning. The new framework is a hybrid between alternating optimization (AO) and the alternating direction method of multipliers (ADMM): each matrix factor is updated in turn, using ADMM. This combination can naturally accommodate a great variety of constraints on the factor matrices, hence the term 'universal'. Computation caching and warm start strategies are used to ensure that each update is evaluated efficiently, while the outer AO framework guarantees that the algorithm converges monotonically. Simulations on synthetic data show significantly improved performance relative to state-of-the-art algorithms.

1. INTRODUCTION

Constrained matrix and tensor factorization techniques are widely used for latent parameter estimation and blind source separation in signal processing, dimensionality reduction and clustering in machine learning, and numerous other applications in diverse disciplines, such as chemistry and psychology. Unconstrained rank factorization of matrices and tensors is relatively well-studied, as in the matrix case the basis of any solution is simply the principal components of the singular value decomposition (SVD), and in the tensor case alternating least squares (ALS) and other algorithms usually yield satisfactory results. However, the available algorithms for *constrained* matrix and tensor factorization leave much to be desired as of this writing, and a unified framework that can easily and naturally incorporate multiple constraints on the latent factors is sorely missing. Existing algorithms are usually only able to handle one or at most few specialized constraints, and/or the algorithm needs to be redesigned carefully if new constraints are added.

In this paper, we propose a general algorithmic framework that seamlessly and relatively effortlessly incorporates many common types of constraints and compositions thereof, building upon and bridging together the alternating optimization (AO) framework and the alternating direction method of multipliers (ADMM). While the AO framework provides monotone convergence for these (NP-)hard factorization problems, ADMM serves as a sub-routine that handles almost 'universal' constraints very efficiently, with per-iteration complexity similar to the unconstrained LS algorithm or sub-optimal approaches like the multiplicative non-negative matrix factorization (NMF) update.

1.1. Notation

We denote the (approximate) factorization of matrix $\mathbf{Y} \approx \mathbf{W}\mathbf{H}^T$, where $\mathbf{Y}$ is $m \times n$, $\mathbf{W}$ is $m \times k$, and $\mathbf{H}$ is $n \times k$, $k \leq m, n$, and in most cases much smaller. Note that adding constraints on $\mathbf{W}$ and $\mathbf{H}$ may turn the solution from easy to find (via SVD) but non-identifiable, to NP-hard to find but identifiable. It has been shown that simple constraints like non-negativity and sparsity can make the factors essentially unique, but at the same time, computing the optimal solution becomes NP-hard – see [1] and references therein.

For simplicity, we only consider 3-way tensors in this paper, which are denoted with an under-score $\underline{\mathbf{Y}}$, of size $n_1 \times n_2 \times n_3$. In what follows, we focus on the so-called parallel factor analysis (PARAFAC) model, also known as canonical decomposition (CANDECOMP) or canonical polyadic decomposition (CPD), which is essentially unique under mild conditions [2], but constraints certainly help enhance estimation performance, and even identifiability. One compact way to represent the CPD model is by writing it in matricized form as $\mathbf{Y}_{(1)}^T \approx (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T$, where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are the three factors sharing the same number of columns k, $\mathbf{Y}_{(1)}$ is the mode-1 matricization of $\underline{\mathbf{Y}}$, and $\odot$ denotes the Khatri-Rao product. Details of these tensor related operations can be found in tutorial papers like [3], and are omitted here.

1.2. Alternating direction method of multipliers

The ADMM solves convex problems that can be put in the following form

$$\begin{aligned} \underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \end{aligned}$$

*Email: (huang663,nikos)@umn.edu. Supported in part by NSF IIS-1247632, IIS-1447788, and a UM Informatics Institute fellowship.

†Email: liavas@telecom.tuc.gr.

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

by iterating the following updates

$$\begin{aligned}\mathbf{x} &\leftarrow \arg\min_{\mathbf{x}} f(\mathbf{x}) + (\rho/2)\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}\|_2^2,\\ \mathbf{z} &\leftarrow \arg\min_{\mathbf{z}} g(\mathbf{z}) + (\rho/2)\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}\|_2^2,\\ \mathbf{u} &\leftarrow \mathbf{u} + (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}),\end{aligned} \tag{1}$$

where $\mathbf{u}$ is a scaled version of the dual variable corresponding to the equality constraint $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$, and ρ is specified by the user.

A comprehensive review of the ADMM algorithm can be found in [4] and the references therein. ADMM converges under very mild conditions (f and g being closed, proper, and convex, and that an optimal solution exists). Notice that smoothness is not required, which means inequality constraints can be incorporated into the cost function by setting the function value to be $+\infty$ if the constraints are not satisfied. Moreover, the splitting of $\mathbf{x}$ and $\mathbf{z}$ can be carefully chosen so that each of the updates is trivial (possibly in closed-form).

2. CONSTRAINED MATRIX FACTORIZATION

We start with an algorithm for constrained matrix factorization. We adopt the common matrix factorization setting by formulating it as the optimization problem

$$\underset{\mathbf{W},\mathbf{H}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{Y} - \mathbf{W}\mathbf{H}^T\|_F^2 + r_W(\mathbf{W}) + r_H(\mathbf{H}), \tag{2}$$

where $r_W(\cdot)$ and $r_H(\cdot)$ are regularizations imposed on the latent factors $\mathbf{W}$ and $\mathbf{H}$, and can take value $+\infty$, so that any hard constraints can also be included.

The problem (2) is not convex in both $\mathbf{W}$ and $\mathbf{H}$, but is convex in one if we fix the other, thus alternating optimization is usually employed. We first describe how to efficiently update $\mathbf{H}$ while fixing $\mathbf{W}$ using ADMM, and then treat this as a sub-routine to solve (2) efficiently.

2.1. ADMM for regularized least-squares

We first reformulate the subproblem for the update of $\mathbf{H}$ as

$$\begin{aligned}\underset{\mathbf{H},\tilde{\mathbf{H}}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{Y} - \mathbf{W}\tilde{\mathbf{H}}\|_F^2 + r_H(\mathbf{H})\\ \text{subject to} \quad & \mathbf{H} = \tilde{\mathbf{H}}^T.\end{aligned} \tag{3}$$

It is easy to adopt the ADMM algorithm and derive the following iterates:

$$\begin{aligned}\tilde{\mathbf{H}} &\leftarrow (\mathbf{W}^T\mathbf{W} + \rho\mathbf{I})^{-1}(\mathbf{W}^T\mathbf{Y} + \rho(\mathbf{H} + \mathbf{U})^T),\\ \mathbf{H} &\leftarrow \arg\min_{\mathbf{H}} r_H(\mathbf{H}) + \frac{\rho}{2}\|\mathbf{H} - \tilde{\mathbf{H}}^T + \mathbf{U}\|_F^2,\\ \mathbf{U} &\leftarrow \mathbf{U} + \mathbf{H} - \tilde{\mathbf{H}}^T.\end{aligned} \tag{4}$$

One important observation is that, throughout the iterations the same matrix $\mathbf{W}^T\mathbf{Y}$ and matrix inversion $(\mathbf{W}^T\mathbf{W}+\rho\mathbf{I})^{-1}$ are used. Therefore, to save computations, we can cache $\mathbf{W}^T\mathbf{Y}$ and the Cholesky decomposition of $\mathbf{W}^T\mathbf{W} + \rho\mathbf{I} = \mathbf{L}\mathbf{L}^T$. Then the update of $\tilde{\mathbf{H}}$ is dominated by one forward substitution and one backward substitution, resulting in a complexity of $\mathcal{O}(k^2 n)$.

The update of $\mathbf{H}$ is the so-called *proximal operator* of the function $(1/\rho)r_H(\cdot)$ around point $(\tilde{\mathbf{H}}^T - \mathbf{U})$, and in particular if $r_H(\cdot)$ is an indicator function of a convex set, then the update of $\mathbf{H}$ becomes a projection operator, a special case of the proximal operator. For a lot of regularizations/constraints, especially those that are often used in matrix factorization problems, the update of $\mathbf{H}$ boils down to element-wise updates, i.e., costing $\mathcal{O}(kn)$ flops. Many efficiently implementable proximal operators can be found in [5, §6]. Here we list some of the most commonly used constraints/regularizations in the matrix factorization problem. For simplicity of notation, let us define $\bar{\mathbf{H}} = \tilde{\mathbf{H}}^T - \mathbf{U}$.

- Non-negativity. In this case $r_H(\cdot)$ is the indicator function of $\mathbb{R}_+$, and the update is simply zeroing out the negative values of $\bar{\mathbf{H}}$. In fact, any element-wise bound constraints can be handled similarly, since element-wise projection is trivial.
- Lasso regularization. For $r_H(\mathbf{H}) = \lambda\|\mathbf{H}\|_1$, the sparsity inducing regularization, the update is the well-known *soft-thresholding* operator: $h_{ij} = [1-(\lambda/\rho)|\bar{h}_{ij}|^{-1}]_+\bar{h}_{ij}$. The element-wise thresholding can also be converted to block-wise thresholding if we know *a priori* that the zeros are grouped, leading to the group sparsity regularization.
- Simplex constraint. In some probabilistic model analysis we need to constrain the columns or rows to be element-wise non-negative and sum up to one. As described in [6], this projection can be done with a randomized algorithm with linear-time complexity on average.

We found empirically that by setting $\rho = \|\mathbf{W}\|_F^2/k$, the ADMM iterates for the regularized least-squares problem converge very fast. With a good initialization (which will be discussed later), the update of $\mathbf{H}$ usually does not take more than 5 or 10 ADMM iterations. The proposed algorithm for the sub-problem (3) is summarized in Alg. 1. As we can see, the pre-calculation step takes $\mathcal{O}(k^2m + k^3)$ flops to perform the Cholesky decomposition, and $\mathcal{O}(mnk)$ flops to form $\mathbf{F}$. In the iterations the complexity is dominated by the $\tilde{\mathbf{H}}$-update, with complexity $\mathcal{O}(k^2 n)$. Notice that for the unconstrained problem, the complexity is essentially the same as the pre-calculation step plus one iteration. For a small number of ADMM iterations, the complexity of Alg. 1 is of the same order as the unconstrained problem.

Algorithm 1: Solve (3) using ADMM

Input: $\mathbf{Y}$, $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{U}$
Initialize $\mathbf{H}$ and $\mathbf{U}$;
$\rho = \|\mathbf{W}\|_F^2/k$;
Calculate $\mathbf{L}$ from the Cholesky decomposition of $\mathbf{W}^T\mathbf{W} + \rho\mathbf{I} = \mathbf{L}\mathbf{L}^T$;
$\mathbf{F} = \mathbf{W}^T\mathbf{Y}$;
repeat
  $\tilde{\mathbf{H}} \leftarrow \mathbf{L}^{-T}\mathbf{L}^{-1}(\mathbf{F} + \rho(\mathbf{H} + \mathbf{U})^T)$ using forward/backward substitution ;
  $\mathbf{H} \leftarrow \arg\min_{\mathbf{H}} r_H(\mathbf{H}) + \frac{\rho}{2}\|\mathbf{H} - \tilde{\mathbf{H}}^T + \mathbf{U}\|_F^2$;
  $\mathbf{U} \leftarrow \mathbf{U} + \mathbf{H} - \tilde{\mathbf{H}}^T$;
until *convergence*;
return $\mathbf{H}$ and $\mathbf{U}$.

2.2. Alternating optimization

Now, naturally, Alg. 1 is used as the sub-routine for alternating optimization. Since $\mathbf{W}$ and $\mathbf{H}$ are updated alternatingly, one would expect that the $\mathbf{W}$ and $\mathbf{H}$ (and their corresponding dual variables) obtained in the previous iteration should not be very far away from the updates for the current iteration. Thus, we can initialize the current ADMM update using the previous results. Experiments show that this can greatly reduce the number of inner-iterations required for ADMM. Soon after an initial transient stage, the sub-problems can be solved in just one iteration. The proposed algorithm for constrained matrix factorization is summarized in Alg. 2.

Algorithm 2: Proposed algorithm for (2)

Initialize $\mathbf{W}$ and $\mathbf{H}$;
$\mathbf{U}_W \leftarrow 0, \mathbf{U}_H \leftarrow 0$;
repeat
  $(\mathbf{H}, \mathbf{U}_H) \leftarrow \arg\min_{\mathbf{H}} \frac{1}{2}\|\mathbf{Y} - \mathbf{W}\mathbf{H}^T\|_F^2 + r_H(\mathbf{H})$ using Alg. 1 initialized with $(\mathbf{H}, \mathbf{U}_H)$;
  $(\mathbf{W}, \mathbf{U}_W) \leftarrow \text{argmin}_{\mathbf{W}} \frac{1}{2}\|\mathbf{Y}–\mathbf{W}\mathbf{H}^T\|_F^2 + r_W(\mathbf{W})$ using Alg. 1 initialized with $(\mathbf{W}, \mathbf{U}_W)$;
until *convergence*;

The convergence of Alg. 2 follows from that of the general block coordinate descent algorithm framework, i.e., the objective is monotonically decreasing, and every limit point is a stationary point, if there are only two blocks [7]. If we limit the maximum number of ADMM inner-loops, the updates at the initial phase will not be guaranteed to be conditionally optimal – but this can be treated as a generalized initialization step. By using the proposed warm-start strategy, the ADMM inner-loops soon become conditionally optimal, thus ensuring convergence of the outer-loop.

3. EXTENSION TO TENSOR FACTORIZATION

Going from two-way to higher-way data arrays is easy when using our approach: we can again adopt AO over the matrix factors, and solve each of the matrix sub-problems using Alg. 1. Without loss of generality, let us consider the update of $\mathbf{A}$, which is $\arg\min_{\mathbf{H}}$ of (3) with $\mathbf{W} = \mathbf{C} \odot \mathbf{B}$, the Khatri-Rao product of $\mathbf{C}$ and $\mathbf{B}$, and $\mathbf{Y} = \mathbf{Y}_{(1)}^T$, the mode-1 matricization of the data tensor $\underline{\mathbf{Y}}$.

Due to the Khatri-Rao structure of $\mathbf{W}$, $\mathbf{W}^T\mathbf{W}$ can be computed efficiently as $\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B}$, where $*$ is the element-wise (Hadamard) product. How to efficiently calculate $\mathbf{W}^T\mathbf{Y} = (\mathbf{C} \odot \mathbf{B})^T\mathbf{Y}_{(1)}^T$ depends on how the data tensor $\underline{\mathbf{Y}}$ is stored, and various efficient implementations have been proposed [8–10]. For simplicity, we only consider the dense case, for which a computationally (but not memory-) efficient way is to replicate the tensor in three matricizations $\mathbf{Y}_{(1)}$, $\mathbf{Y}_{(2)}$, and $\mathbf{Y}_{(3)}$, so that in each iteration they are readily available. Beyond the computation of these special products, the rest of the computations involved in the ADMM inner-loops are all very cheap, as we have explained in the matrix case. Detailed algorithm description in the tensor case is omitted due to space limitations, and will be given in the journal version.

As we can see, the ADMM is an appealing sub-routine for alternating optimization, leading to a simple plug-and-play generalization of the workhorse ALS algorithm. Theoretically, they share the same per-iteration complexity if the number of inner ADMM iterations is small, which is true in practice, after an initial transient. Efficient implementation of the overall algorithm should include data-structure-specific algorithms for $(\mathbf{C} \odot \mathbf{B})^T\mathbf{Y}_{(1)}^T$, and may include parallel/distributed computation along the lines of [11]. Related developments will be included in the journal version.

4. RELATED WORK

The most common constraint imposed on the latent factors is non-negativity. Traditional methods used for the alternating sub-problems are active-set (AS) [12] and its variations like block principle pivoting [13, 14]. The main idea is that the problem becomes unconstrained if we figure out the variables that should be equal to zero. However, even if we know exactly where the zeros should be (which is approximately so in the alternating optimization framework), the least-squares problem for each row of $\mathbf{H}$ is different. One can form $\mathbf{W}^T\mathbf{W}$ and $\mathbf{W}^T\mathbf{Y}$ once, but for each row of $\mathbf{H}$ different subsets of entries in $\mathbf{W}^T\mathbf{W}$ should be inverted. This becomes unappealing when k grows large. Some other methods, like the multiplicative-update (MU) [15] or hierarchical alternating least squares (HALS) [16], ensure that the per-iteration complexity is dominated by calculating $\mathbf{W}^T\mathbf{W}$ and $\mathbf{W}^T\mathbf{Y}$, although more outer-loops are needed for convergence. To move beyond non-negativity, MU is not applicable, AS is di-

rectly amendable to sparsity regularization only, and HALS is able to handle all constraints imposed on the columns of $\mathbf{H}$ [17], but not the rows.

The ADMM algorithm for constrained/regularized linear least-squares problems is widely used in large scale compressive sensing and image processing [18, 19] applications, thanks to its nice convergence properties and the ability of Cholesky caching. However, to the best of our knowledge, this is the first time that it is used as a sub-routine for constrained matrix/tensor factorization. ADMM is a perfect fit for this kind of problem, because essentially each sub-problem is a large number of constrained least-squares *sharing the same mixing matrix*, thus one matrix inversion is not only amortized throughout the inner iterations but also across different rows (as opposed to, say, what happens with the active-set method), resulting in per-iteration complexity that is almost the same as the unconstrained one. Furthermore, the alternating optimization framework naturally provides good initializations, which greatly reduces the number of inner-iterations required.

We should emphasize that our proposed algorithm is in general alternating optimization, and ADMM is only applied to the sub-problems. There are also algorithms that directly apply the ADMM approach to the whole problem [11, 20]. In that case, the per-iteration complexity is also the same as the unconstrained alternating least-squares. However, due to the non-convexity of the problem, the objective is not guaranteed to decrease monotonically, unlike alternating optimization. Moreover, both ADMM and AO guarantee that every limit point is a stationary point, but in practice AO almost always converges, which is not the case for ADMM (applied to the whole problem).

5. NUMERICAL EXAMPLES

In this section we provide some illustrative examples to showcase the numerical performance of our proposed algorithm on synthetic data, compared to some of the existing algorithms mentioned in the previous section. All experiments are performed in MATLAB 2013a on a Linux server with 32 Xeon 2.00GHz cores and 128GB memory.

5.1. Non-negative matrix factorization

Here we compare the proposed algorithm with some state-of-the-art NMF algorithms on synthetic data. By specifying the values of m, n, and k, the true $\mathbf{W}$ and $\mathbf{H}$ are generated by drawing the elements from an i.i.d. exponential distribution with mean 1, and then 50% of the elements are randomly set to 0. The data matrix $\mathbf{Y}$ is then set to be their product $\mathbf{Y} = \mathbf{W}\mathbf{H}^T + \mathbf{N}$, where the elements of $\mathbf{N}$ are drawn from an i.i.d. Gaussian distribution with variance $\sigma^2 = 10^{-2}$.

Three algorithms are used for comparison:

HALS Hierarchical alternating least-squares [16];

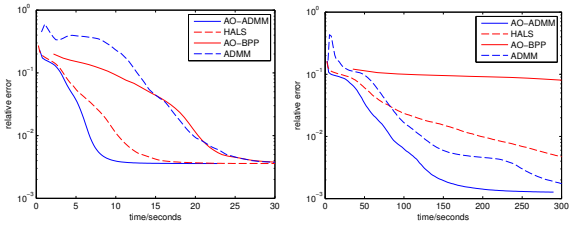

Fig. 1: Convergence of some NMF algorithms.

Algorithm	$\|\mathbf{Y} - \mathbf{W}\mathbf{H}^T\|_F$	run time	iterations
AO-ADMM	193.0702	22.0218s	109.3
HALS	193.1642	28.4811s	205.0
AO-BPP	197.1617	33.6751s	50.4
ADMM	198.1530	29.4779s	102.9

Table 1: Averaged performance of NMF algorithms.

AO-BPP Alternating optimization using block principle pivoting [13][1];

since these two algorithms are reported in [13] to out-perform other NMF algorithms, and

ADMM ADMM applied to the whole problem [20][2].

The proposed algorithm is denoted as **AO-ADMM**. All algorithms are initialized with the same random point.

The convergence time in seconds versus the relative error $\|\mathbf{Y} - \mathbf{W}\mathbf{H}^T\|_F / \|\mathbf{Y}\|_F$ of the aforementioned algorithms for two indicative problem instances is shown in Fig. 1. On the left, the problem size is $m = n = 2000$ and $k = 100$. As we can see, our proposed algorithm converges with the fastest rate. On the right, the problem size is increased to $m = n = 5000$, and $k = 300$, and the gap becomes larger, indicating that our proposed algorithm has better scalability, thus it is more suitable for big data. The first case is also repeated 10 times with random synthetic data, and the averaged result is given in Table 1.

5.2. Non-negative PARAFAC

We now test the algorithms in the case of tensors. For different values of n_1, n_2, n_3, and k, the true latent factors $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are generated in the same manner as $\mathbf{W}$ and $\mathbf{H}$ in the NMF case. The data tensor $\underline{\mathbf{Y}}$ is then generated as a low-rank PARAFAC model with additive noise $y_{ijl} = \sum_{f=1}^{k} a_{if} b_{jf} c_{lf} + \nu_{ijl}$, where ν_{ijl} are i.i.d. Gaussian noise with variance $\sigma^2 = 10^{-2}$.

Again, our proposed algorithm, denoted as AO-ADMM, is compared with HALS [16][1], AO-BPP [14][1], and ADMM [11], and the convergence time in seconds versus the relative error in indicative problem instances is shown in Fig. 2.

[1] Matlab code downloaded from `http://www.cc.gatech.edu/~hpark/nmfsoftware.php`

[2] Matlab code downloaded from `http://mcnf.blogs.rice.edu/`

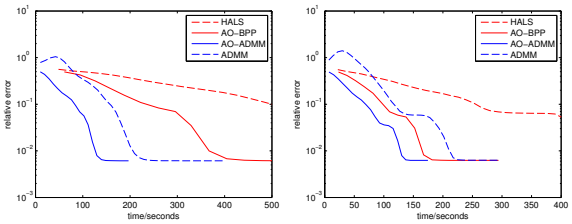

Fig. 2: Convergence of some NTF algorithms.

Algorithm	$\|\mathbf{Y}_{(1)}^T - (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T\|_F$	run time	iterations
AO-ADMM	1117.634	222.4882s	32.0
AO-BPP	1117.674	723.8094s	23.6
HALS	1117.665	1943.9312s	146.7
ADMM	1163.799	427.9522s	77.0
tensorlab	1119.867	369.9924s	N/A

Table 2: Averaged performance of NTF algorithms

All algorithms are initialized with the same random point[3]. On the left, $n_1 = n_2 = n_3 = 500$ and $k = 100$, while a somewhat unbalanced case is shown on the right with $n_1 = 1000$, $n_2 = 500$, $n_3 = 200$ and $k = 100$. The first case is also repeated 10 times with random synthetic data, and the averaged result is given in Table 2. For this experiment we have also included Tensorlab [21], which handles non-negative PARAFAC using "all-at-once" derivative-based nonlinear least-squares optimization techniques. As we can see, AO-ADMM again outperforms all other algorithms in all cases.

6. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a general algorithmic framework for matrix and tensor factorization that aims to approximate the data matrix/tensor in the least-squares sense, with the ability to efficiently handle almost 'universal' constraints/regularizations imposed on the latent factors. The algorithm builds upon the widely used alternating optimization framework, but the constrained sub-problems are solved via the alternating direction method of multipliers. Simulations on synthetic data, with non-negativity constraints imposed, showed great performance compared to the state-of-the-art algorithms. More simulations of this algorithm with more constraints imposed and applied to real data will be given in the journal version.

We are also working on applying this algorithmic framework for factorization problems where the error measure is different from the least-squares criterion, e.g., the l_1 loss, the Kullback-Leibler divergence, and/or loss evaluated only at a subset of entries in $\mathbf{Y}$ when there are missing values. The per-iteration complexity is still similar to the unconstrained ALS algorithm, again by exploiting Cholesky caching and warm start. The goal is to provide a simple algorithmic framework for constrained matrix and tensor factorization which maintains the cheap per-iteration complexity and monotone reduction of the cost function, while providing flexibility in terms of constraints and faster convergence.

[3]However, ADMM is reported to be more sensitive to 'good' initializations [11], thus ADMM is probable to be restarted at a different initialization.

REFERENCES

[1] K. Huang, N. D. Sidiropoulos, and A. Swami, "Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition," *IEEE Trans. on Signal Processing*, vol. 62, no. 1, pp. 211–224, 2014.

[2] N. D. Sidiropoulos and R. Bro, "On the uniqueness of multilinear decomposition of N-way arrays," *Journal of chemometrics*, vol. 14, no. 3, pp. 229–239, 2000.

[3] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[4] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[5] N. Parikh and S. P. Boyd, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 123–231, 2014.

[6] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l_1-ball for learning in high dimensions," in *Proc. ACM ICML*, 2008, pp. 272–279.

[7] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear Gauss-Seidel method under convex constraints," *Operations Research Letters*, vol. 26, no. 3, pp. 127–136, 2000.

[8] B. W. Bader and T. G. Kolda, "Efficient matlab computations with sparse and factored tensors," *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 205–231, 2007.

[9] U Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries," in *Proc. ACM SIGKDD*, 2012, pp. 316–324.

[10] N. Ravindran, N.D. Sidiropoulos, S. Smith, and G. Karypis, "Memory-efficient parallel computation of tensor and matrix products for big tensor decomposition," in *Asilomar Conference on Signals, Systems, and Computers*, 2014.

[11] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via the alternating direction method of multipliers," *IEEE Trans. on Signal Processing*, 2014, submitted.

[12] R. Bro and S. De Jong, "A fast non-negativity-constrained least squares algorithm," *Journal of Chemometrics*, vol. 11, no. 5, pp. 393–401, 1997.

[13] J. Kim and H. Park, "Fast nonnegative matrix factorization: An active-set-like method and comparisons," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, 2011.

[14] J. Kim and H. Park, "Fast nonnegative tensor factorization with an active-set-like method," in *High-Performance Scientific Computing*, pp. 311–326. Springer, 2012.

[15] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems*, 2001, pp. 556–562.

[16] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 92, no. 3, pp. 708–721, 2009.

[17] R. Bro and N. D Sidiropoulos, "Least squares algorithms under unimodality and non-negativity constraints," *Journal of Chemometrics*, vol. 12, no. 4, pp. 223–247, 1998.

[18] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Trans. on Image Processing*, vol. 19, no. 9, pp. 2345–2356, 2010.

[19] E. Esser, Y. Lou, and J. Xin, "A method for finding structured sparse solutions to nonnegative least squares problems with applications," *SIAM Journal on Imaging Sciences*, vol. 6, no. 4, pp. 2010–2046, 2013.

[20] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Frontiers of Mathematics in China*, vol. 7, no. 2, pp. 365–384, 2012.

[21] L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab v2.0," `http://www.tensorlab.net/`, Jan. 2014.