Scalable Algorithms for Locally Low-Rank Matrix Modeling

Qilong Gu
Dept. of Computer Science and Engineering
University of Minnesota, Twin Cities
guxxx396@cs.umn.edu

Joshua D. Trzasko
Dept. of Radiology
Mayo Clinic
Trzasko.Joshua@mayo.edu

Arindam Banerjee
Dept. of Computer Science and Engineering
University of Minnesota, Twin Cities
banerjee@cs.umn.edu

***Abstract*—We consider the problem of modeling data matrices with locally low rank (LLR) structure, a generalization of the popular low rank structure widely used in a variety of real world application domains ranging from medical imaging to recommendation systems. While LLR modeling has been found to be promising in real world application domains, limited progress has been made on the design of scalable algorithms for such structures. In this paper, we consider a convex relaxation of LLR structure, and propose an efficient algorithm based on dual projected gradient descent (D-PGD) for computing the proximal operator. While the original problem is non-smooth, so that primal (sub)gradient algorithms will be slow, we show that the proposed D-PGD algorithm has geometrical convergence rate. We present several practical ways to further speed up the computations, including acceleration and approximate SVD computations. With experiments on both synthetic and real data from MRI (magnetic resonance imaging) denoising, we illustrate the superior performance of the proposed D-PGD algorithm compared to several baselines.**

I. Introduction

Recent years have seen considerable advances in modeling 'corrupted' data matrices by assuming the true underlying matrix has some suitable structure, such as low-rank or variants [1], [2]. Two popular forms of corruption which have been studied in the literature include matrices with noisy observations, typically studied in computer vision or medical imaging domains [1], [2], and matrices with missing observations, typically studied in the context of recommendation systems [3]–[5]. Assuming the true underlying matrix has a structure, such as low rank, recovering the true matrix involves an estimation algorithm often based on a convex relaxation of a suitable structured estimation problem [6]–[8]. Irrespective of the exact nature of corruption, say noisy vs. missing entries, the core challenge in designing efficient algorithms for such estimation typically involves being able to efficiently compute the proximal operator of the structure under consideration [3], [4].

While low-rank models have been successful in many application domains, including medical imaging [1], [9], [10] and recommendation systems [4], [11], recent work have found that *locally low-rank* (LLR) modeling of matrices can perform better than a low-rank model. In particular, recent work have found that by promoting *locally low-rank* (LLR) structure, the denoising performance on MRI can be greatly improved [12]. Related developments have started in the domain of recommendation systems [3], [13]. Locally low-rank (LLR) structures can be viewed as a generalization of (global) low-rank structures, and allows for more fine-tuned analysis of large data matrices for complex problems [1], [3], [12]. A key challenge in modeling matrices as LLR is computational: unlike widely studied structures such as sparsity or low-rank, currently there is limited progress on efficient algorithms for estimating LLR approximations of a given corrupted matrix, with noisy or missing entries. In particular, the proximal operator for the convex relaxation of the locally low rank structure does not have a closed form solution. In this paper, we introduce an efficient algorithm for computing the proximal operator, and illustrate applications of the idea on both synthetic and real datasets.

To keep the exposition concrete, we focus on noisy corruptions, noting again that recovering structured matrices from noisy or missing entries involve computing the proximal operator of the structure of interest [11], [14]. In the context of matrix denoising, one observes a 'noisy' matrix $Z \in \mathbb{R}^{n\times m}$, which is assumed to be the sum of a 'clean' matrix $X \in \mathbb{R}^{n\times m}$ and noise $W \in \mathbb{R}^{n\times m}$ [1], [2]. One often assumes the clean matrix X to have some specific structure, such as low rank, which is utilized to accurately estimate X from Z. For estimating X, the literature considers a convex relaxation of the estimation problem, replacing the low-rank structure of X with a regularization based on the nuclear norm of X, i.e., an estimator of the form

$$\min_X \ \frac{1}{2}\|Z - X\|_F^2 + \lambda\|X\|_* \ , \tag{1}$$

where $\|.\|_*$ denotes the nuclear norm, and $\lambda > 0$ is a regularization parameter [1]. One can show that (1) can be solved using the so-called singular value thresholding (SVT) [1], [2]. In particular, considering the singular value decomposition (SVD) of Z to be $Z = U\Sigma V^T$ where $\Sigma = \mathrm{diag}(\sigma_i)$, the diagonal matrix of singular values, the solution to (1) is given by $X_\lambda = U\Sigma_\lambda V^T$ where $\Sigma_\lambda = \mathrm{diag}((\sigma_i - \lambda)_+)$, the diagonal matrix of soft-thresholded singular values. The SVT approach has been empirically and statistically analyzed, and is considered an effective approach to image denoising [1].

In this paper, we say a matrix $X \in \mathbb{R}^{n\times m}$ is *locally low-rank* (LLR) [15] if there are sub-sets of rows of X which are low-rank. In general, the sub-sets of rows can be overlapping, and the setting of overlapping sub-sets constitute

the more interesting case for the estimation problem. Let $\mathcal{G}_i \subseteq \{1, 2, \ldots, n\}, i = 1, \ldots, L$, be the indices of the i-th subset of rows so that $|\mathcal{G}_i| \leq n$, where $|\mathcal{G}_i|$ denotes the cardinality of that subset. For each subset $\mathcal{G}_i$, there is a corresponding row-selection matrix $Q_{\mathcal{G}_i} \in \{0,1\}^{|\mathcal{G}_i| \times m}$ so that $Q_{\mathcal{G}_i} X$ extracts rows in $\mathcal{G}_i$ from X and forms a $|\mathcal{G}_i| \times m$ matrix. By the locally low-rank assumption, each submatrix $Q_{\mathcal{G}_i} X$ is low-rank. Then, given a collection of row-selection matrices $\{Q_{\mathcal{G}_i}\}_{i=1}^L$ that cover all rows of X, the denoising problem can be posed as:

$$\min_X \ \frac{1}{2}\|Z - X\|_F^2 \quad \text{s.t.} \quad Q_{\mathcal{G}_i} X, i = 1, \ldots, L \text{ are low rank} \ . \tag{2}$$

In this paper, we consider a convex relaxation of the problem in (2) where the low-rank structure is captured by a regularization based on the nuclear norm of the sub-matrices. In particular, we focus on the following convex optimization problem:

$$\min_X \ g(X) = \frac{1}{2}\|Z - X\|_F^2 + \lambda \sum_{i=1}^{L} \|Q_{\mathcal{G}_i} X\|_* \ , \tag{3}$$

where $\lambda > 0$ is a constant, and the collection of row-selection matrices $\{Q_{\mathcal{G}_i}\}_{i=1}^L$ is pre-specified. Note that such pre-specification is possible in the context of MRI denoising using domain knowledge [12], [15]. In settings such as recommendation systems, one can consider overlapping clustering of the users (rows) [3], [16], so that each user cluster is essentially assumed to be low-rank.

If the collection $\{Q_{\mathcal{G}_i}\}_{i=1}^L$ correspond to non-overlapping rows, then (3) has a closed form solution, which can be obtained by SVT of each block of Z corresponding to $Q_{\mathcal{G}_i} X$. In this paper, we focus on the general case when the subsets are allowed to overlap.

We introduce a novel dual projected gradient descent (D-PGD) algorithm for solving (3), i.e., computing the proximal operator of LLR structures. Note that since the primal problem is strongly convex but non-smooth, a sub-gradient descent algorithm based on the primal will have a sublinear $O(1/\sqrt{t})$ rate of convergence [17], [18]. If we apply some acceleration strategies, then the convergence rate of gradient algorithm can be improved to $O(1/t^2)$ [19]. Interestingly, we show that the proposed D-PGD algorithm will converge geometrically, i.e., a $O(\gamma^t), \gamma < 1$ rate of convergence. To make the D-PGD algorithm scalable, we consider Nesterov acceleration [20] and also partial SVD computations [21] to speed up each iteration. Further, we consider an Adaptive Acceleration (AA) algorithm based on D-PGD, which does a mix of basic D-PGD and accelerated D-PGD on the dual, adaptively choosing when to use (restart) the acceleration. Based on our results for D-PGD, we can show that AA will also converge geometrically. We discuss several alternative approaches to solving the problem, including ADMM (alternating direction method of multipliers) [22], block coordinate descent (BCD) [17], and block singular value theresholding (B-SVT) [1], which are used as baselines for empirical comparison. In application of MRI denoising, we propose a parallel algorithm for D-PGD. Previous works have shown that for parallel proximal algorithm [22], [23], we need a master process, and in each step the master process has to collect data from all other processes. In this work, we propose a method to avoid master process, therefore the speed up thought parallelization can be almost linear.

The performance of D-PGD is empirically evaluated using both synthetic and real datasets. For real datasets, we consider MRI (magnetic resonance imaging) denoising problems, focusing on time-resolved (i.e., dynamic) cardiac MR imaging [15] and multi-channel brain imaging [12]. We report results on different variants of the proposed D-PGD algorithm, illustrating that adaptive acceleration (AA) and partial SVD calculations lead to the most efficient versions of D-PGD, which outperforms baselines based on ADMM and BCD. For MRI denoising, we illustrate that D-PGD reaches the same quality of denoised images in much shorter time compared with the baselines for both cardiac MRI and brain MRI denoising.

The rest of the paper is organized as follows. In Section II, we present the D-PGD algorithm for solving (3) and establish its geometrical rate of convergence. We present scalable variants of D-PGD in Section III. We discuss alternative algorithms for LLR modeling in Section IV. In Section V, we discuss the MRI denoising application and datasets, and approaches to parallelize the computation for this application. We present experimental results on synthetic and real datasets in Section VI, and conclude in Section VII.

II. Algorithms for LLR Modeling

The estimation problem (3) is essentially about computing the proximal operator of the norm obtained as a convex relaxation of the LLR structure. For convenience, we will refer to the norm as *overlapping nuclear norm*. In this section, we outline our way of computing the proximal operator of the overlapping nuclear norm based on D-PGD. We show that our algorithm in fact has a geometrical convergence rate.

A. Dual Projected Gradient Descent (D-PGD)

By making use of the dual norm of the nuclear norm [24], we rewrite problem (3) as

$$\min_X \max_{Y_i \in \Omega_i^\lambda, i=1,\ldots,L} \mathcal{L}(X, \{Y_i\}) = \frac{1}{2}\|Z - X\|_F^2 + \sum_{i=1}^{L} \langle Q_{\mathcal{G}_i} X, Y_i \rangle \ , \tag{4}$$

where we define

$$\Omega_i^\lambda = \{Y_i \in \mathbb{R}^{|\mathcal{G}_i| \times n} : \|Y_i\|_2 \leq \lambda\} \ . \tag{5}$$

Algorithm 1 Dual Projected Gradient Descent Algorithm (D-PGD)

Inputs: $\{Q_{\mathcal{G}_i}\}_{i=1}^L$, Z.
Initialize: $\{Y_i^0\}_{i=1}^L = \{0\}$.
for $t = 0, 1, \ldots, T$ **do**
  for $i = 1, \ldots, L$ **do**
    $Y_i^{t+1} = \Pi_{\Omega_i^\lambda}(Y_i^t - \frac{1}{d_{max}} Q_{\mathcal{G}_i}(\sum_{i'=1}^L Q_{\mathcal{G}_{i'}}^T Y_{i'}^t - Z)))$
  end for
end for

It is easy to verify that $\mathcal{L}(X, \{Y_i\})$ is convex in X and concave in $\{Y_i\}$. We can change the order of min and max. By minimizing $\mathcal{L}(X, \{Y_i\})$ over X and reordering, we get the dual problem

$$\min_{Y_i \in \Omega_i^\lambda, i=1,\dots,L} \quad f(\{Y_i\}_{i=1}^L) = \frac{1}{2} \left\| Z - \sum_{i=1}^L Q_{\mathcal{G}_i}^T Y_i \right\|_F^2 \tag{6}$$

In the dual problem (6), the overlapping part has been separated into different blocks. We make use of projected gradient descent (PGD) [17] to solve problem (6) (Algorithm 1). Given the current iterate $\{Y_i^t\}_{i=1}^L$, PGD takes a gradient step and projects onto the feasible set as follows:

$$Y_i^{t+1} = \Pi_{\Omega_i^\lambda} \left(Y_i^t - \frac{1}{d_{\max}} \nabla f(\{Y_i^t\}_{i=1}^L) \right) \ , \quad i = 1, \dots, L \ , \tag{7}$$

where the step size is determined by

$$d_{max} = \max_{j=1,\dots,n} \ |\{i : j \in \mathcal{G}_i\}| \ , \tag{8}$$

the maximum number of groups a row belongs to. The demanding aspect of the computation is a projection onto the feasible set Ω_i^λ, and we denote the projection operator as $\Pi_{\Omega_i^\lambda}(\cdot)$. Note that for any matrix W, the projection $\Pi_{\Omega_i^\lambda}(W)$ can be computed exactly. If the singular value decomposition of the matrix $W = U\Sigma V^T$, where $\Sigma = \operatorname{diag}(\sigma(W))$ and $\sigma(W)$ is the vector of all singular values of W, then

$$\Pi_{\Omega_i^\lambda}(W) = U \operatorname{diag}(\min\{\sigma(W), \lambda\}) V^T \ . \tag{9}$$

Note that Algorithm 1 is parameter free, therefore its performance is stable.

B. Geometrical Convergence of D-PGD

The convergence analysis of PGD has been well studied in the literature. For general convex function and convex set Ω_i, the convergence of algorithm is known to be sub-linear, with a $O(1/t)$ rate of convergence [25]. While better rates are possible for strongly convex and smooth functions, the problem (6) is not strongly convex.

Let us first characterize the optimal solution set $\mathcal{Y} = \{Y_i^*\}_{i=1}^L$ of (6). A collection of variables $\{Y_i\}_{i=1}^L$ satisfy the Karush-Kuhn-Tucker (KKT) conditions of (6) if

$$\begin{aligned} Q_{\mathcal{G}_i} \left(Z - \sum_{i'=1}^L Q_{\mathcal{G}_{i'}} Y_{i'}^* \right) \in \mu_i \partial \|Y_i^*\|_2 \\ \mu_i(\|Y_i^*\|_2 - \lambda) = 0 \ , \ \mu_i \geq 0 \\ Y_i \in \Omega_i^\lambda \ , \ i = 1, \dots, L. \end{aligned} \tag{10}$$

Since (6) is a convex optimization problem, the KKT conditions are both necessary and sufficient for optimality. Note that if we introduce a new variable Y such that $Y = \sum_{i=1}^L Q_{\mathcal{G}_i}^T Y_i$, then the new objective function $g(Y) = \frac{1}{2}\|Z - Y\|_F^2$ is strongly convex. Then, using [26, Proposition 1], we can show the following result:

Lemma 1: For problem (6), there are a matrix Y^* and $\mu_i \geq 0, i = 1, \dots, L$ such that for all $\{Y_i\}_{i=1}^L \in \mathcal{Y}$

$$\sum_{i=1}^L Q_{\mathcal{G}_i} Y_i^* = Y^*, \ Q_{\mathcal{G}_i} X^* \in \mu_i \partial \|Y_i^*\|_2, \ \mu_i(\|Y_i\|_2 - \lambda) = 0 \ , \tag{11}$$

where $X^* = Z - Y^*$.

Lemma 1 shows that $\mathcal{Y}$ can be characterized by Y^* and μ_i. By convex analysis, we also have

$$\mathcal{Y} = \left\{ \{Y_i\}_{i=1}^L : \sum_{i=1}^L Q_{\mathcal{G}_i}^T Y_i = Y^*, Y_i \in \lambda \partial \|Q_{\mathcal{G}_i} X^*\|_* \right\}, \tag{12}$$

which gives alternative optimality conditions that do not require μ_i. The characterization of $\mathcal{Y}$ as in (12) plays an important role in our analysis.

Our convergence analysis is based on the *error bound property* (EBP) of problem (6). Let

$$d(\{Y_i\}_{i=1}^L, \mathcal{Y}) = \inf_{\{Y_i'\}_{i=1}^L \in \mathcal{Y}} \sqrt{\sum_{i=1}^L \|Y_i - Y_i'\|_F^2} \tag{13}$$

be the distance of any collection $\{Y_i\}_{i=1}^L$ to optimal solution set $\mathcal{Y}$. Further, for any collection $\{Y_i\}_{i=1}^L$, let

$$R_i(Y_i) = \Pi_{\Omega_i^\lambda}(Y_i + Q_{\mathcal{G}_i}(Z - \sum_{i'=1}^L Q_{\mathcal{G}_{i'}} Y_{i'})) - Y_i, i = 1, \dots, L \tag{14}$$

which characterizes the residual corresponding to one gradient update of Algorithm 1.

Our first key result (Theorem 2) shows that under mild conditions the EBP based characterization $d(\{Y_i\}_{i=1}^L, \mathcal{Y})$ is of the order $\sqrt{\sum_{i=1}^L \|R_i(Y_i)\|_F^2}$ for all feasible $\{Y_i\}_{i=1}^L$.

Theorem 2: Suppose there exists an $\{Y_i^*\}_{i=1}^L \in \mathcal{Y}$ such that

$$\lambda \notin \sigma(Y_i^* + Q_{\mathcal{G}_i} X^*), i = 1, \dots, L \ , \tag{15}$$

so λ is not one of the singular values of $(Y_i^* + Q_{\mathcal{G}_i} X^*)$. Then there exist constant $\kappa > 0$ such that

$$\begin{aligned} d(\{Y_i\}_{i=1}^L, \mathcal{Y}) \leq \kappa \sqrt{\sum_{i=1}^L \|R_i(Y_i)\|_F^2} \\ \text{for any} \quad Y_i \in \Omega_i^\lambda, i = 1, \dots, L \end{aligned} \tag{16}$$

We present a sketch of the proof below.

Proof sketch: The structure of our proof follows the framework in [26]. Let

$$\Gamma(Y) = \left\{ \{Y_i\}_{i=1}^L : \sum_{i=1}^L Q_{\mathcal{G}_i}^T Y_i = Y \right\} \ , \tag{17}$$

which is the solution set of a linear system. It follows from (12) that the solution set $\mathcal{Y} = \Gamma(Y^*) \cap \bigotimes_{i=1}^L \lambda \partial \|Q_{\mathcal{G}_i} X^*\|_*$ where $\bigotimes$ is the Cartesian product. If condition (15) holds, then we can decouple the pair $(\Gamma(Y^*), \bigotimes_{i=1}^L \lambda \partial \|Q_{\mathcal{G}_i} X^*\|_*)$ by the following lemma.

Lemma 3: Suppose there exists a $\{Y_i^*\}_{i=1}^L \in \mathcal{Y}$ satisfying

$$\lambda \notin \sigma(Y_i^* + Q_{\mathcal{G}_i} X^*), i = 1, \ldots, L$$

Then there exists a constant $\kappa > 0$ such that

$$\begin{aligned} d(\{Y_i\}_{i=1}^L, \mathcal{Y}) \leq \quad & \kappa(d(\{Y_i\}_{i=1}^L, \Gamma(Y^*)) + \\ & d(\{Y_i\}_{i=1}^L, \bigotimes_{i=1}^L \partial \|Q_{\mathcal{G}_i} X^*\|_*)) \end{aligned} \tag{18}$$

By Hoffman's bound [27], there is a constant $\kappa_2 > 0$ such that

$$d(\{Y_i\}_{i=1}^L, \Gamma(Y^*)) \leq \kappa_2 \| \sum_{i=1}^L Q_{\mathcal{G}_i}^T Y_i - Y^* \|_F \tag{19}$$

Further, the second term on right hand side of (18) can be bounded by

$$\begin{aligned} & d(\{Y_i\}_{i=1}^L, \bigotimes_{i=1}^L \lambda \partial \|Q_{\mathcal{G}_i} X^*\|_*)) \leq \kappa_3 \sqrt{\sum_{i=1}^L \|X_i - Q_{\mathcal{G}_i} X^*\|_F^2} \\ & \text{for all } Y_i \in \partial \|X_i\|_* \end{aligned} \tag{20}$$

which follows the following bound for each block

Lemma 4: For any Y_i^* such that $Y_i^* \in \lambda \partial \|Q_{\mathcal{G}_i} X^*\|_*$, there exist constants $\kappa_i' > 0$ such that

$$d(Y_i, \lambda \partial \|Q_{\mathcal{G}_i} X^*\|_*) \leq \kappa_i' \|X_i - Q_{\mathcal{G}_i} X^*\|_F \tag{21}$$

for all matrices $\{Y_i, X_i\}$ that satisfy $Y_i \in \lambda \partial \|X_i\|_*$
Let $\kappa_4 = \max\{\kappa_2, \kappa_3\}$. Replace the bounds from (19) and (20) in (18), we have

$$\begin{aligned} d(\{Y_i\}_{i=1}^L, \mathcal{Y}) \leq & \kappa_4 (\| \sum_{i=1}^L Q_{\mathcal{G}_i}^T Y_i - Y^* \|_F + \\ & \sqrt{\sum_{i=1}^L \|X_i - Q_{\mathcal{G}_i} X^*\|_F^2}) \\ & \text{for all } Y_i \in \partial \|X_i\|_* \end{aligned} \tag{22}$$

Then follows by Lemma 5.

Lemma 5: Suppose there is a constant $\kappa_4 > 0$ such that (22) holds. Then, there exist constants $\kappa > 0$ such that

$$\begin{aligned} & d(\{Y_i\}_{i=1}^L, \mathcal{Y}) \leq \kappa \sqrt{\sum_{i=1}^L \|R_i(Y_i)\|_F^2} \\ & \text{for any } Y_i \in \Omega_i^\lambda, i = 1, \ldots, L \end{aligned}$$

the error bound (16) holds. ∎

Note that the condition (15) under which the result holds is mild, since one just needs to choose λ not to be a singular value of the matrix $Y_i^* + Q_{\mathcal{G}_i} X^*$ for some Y_i^* Using the EBP bound in Theorem 2 and the framework in [28], we now establish the geometrically convergence of D-PGD in Algorithm 1.

Theorem 6: Suppose there exists an $\{Y_i^*\}_{i=1}^L \in \mathcal{Y}$ such that

$$\lambda \notin \sigma(Y_i^* + Q_{\mathcal{G}_i} X^*), i = 1, \ldots, L \ . \tag{23}$$

Then the sequence $\{f(\{Y_i^t\}_{i=1}^L)\}_{t \geq 0}$ generated by Algorithm 1 converge to the optimal value f^* and there is a constant $\eta \in (0, 1)$ such that all $t \geq 0$ satisfies

$$f(\{Y_i^{t+1}\}_{i=1}^L) - f^* \leq \frac{\eta}{\eta + 1} (f(\{Y_i^t\}_{i=1}^L) - f^*) \ . \tag{24}$$

We present a sketch of the proof below.

Proof sketch: By our assumption, and the following lemmas

Lemma 7: For every Y_i, function

$$g_i(s) = \frac{1}{s} \|Y_i - \Pi_{\Omega_i^\lambda}(Y_i - s \nabla f_i(Y_i))\|_F, s > 0$$

is monotonically nonincreasing.

Lemma 8: The sequence $\{f(\{Y_i^t\}_{i=1}^L)\}_{t \geq 0}$ generated by PGD (1) satisfies

$$\begin{aligned} f(\{Y_i^{t+1}\}_{i=1}^L) - f^* \leq d_{max} \Bigg(& \sqrt{\sum_{i=1}^L \|Y_i^t - Y_i^{t+1}\|_F^2} + \\ & \sqrt{\sum_{i=1}^L \|Y_i^t - Y_i^*\|_F^2} \Bigg) \sqrt{\sum_{i=1}^L \|Y_i^t - Y_i^{t+1}\|_F^2} \end{aligned} \tag{25}$$

we have

$$f(\{Y_i^{t+1}\}_{i=1}^L) - f^* \leq d_{max}(1 + \kappa) \sum_{i=1}^L \|Y_i^t - Y_i^{t+1}\|_F^2 \tag{26}$$

We bound right hand side of (26) by

Lemma 9: The sequence $\{f(\{Y_i^t\}_{i=1}^L)\}_{t \geq 0}$ generated by PGD (1) satisfies

$$f(\{Y_i^t\}_{i=1}^L) - f(\{Y_i^{t+1}\}_{i=1}^L) \geq \frac{d_{max}}{2} \sum_{i=1}^L \|Y_i^t - Y_i^{t+1}\|_F^2 \tag{27}$$

and get $f(\{Y_i^{t+1}\}_{i=1}^L) - f^* \leq 2(1 + \kappa)(f(\{Y_i^t\}_{i=1}^L) - f(\{Y_i^{t+1}\}_{i=1}^L))$, which implies

$$f(\{Y_i^{t+1}\}_{i=1}^L) - f^* \leq \frac{\eta}{\eta + 1} (f(\{Y_i^t\}_{i=1}^L) - f^*)$$

where $\eta = 2(1 + \kappa)$. ∎

Thus, the sequence $\{f(\{Y_i^t\}_{i=1}^L)\}_{t \geq 0}$ in Theorem 6 converge geometrically, i.e., rate of convergence γ^t where $\gamma = \frac{\eta}{\eta + 1}$.

III. Scalable D-PGD Updates

In D-PGD, the projection in (9) can be computationally demanding for large matrix W, since it requires the full SVD of matrix W. In this section, we propose a few practical strategies to speed up the algorithm, including efficiently computing the projection operator using adaptive fixed rank SVD [21], doing Nesterov acceleration [18], [20], and doing adaptive acceleration.

A. Efficient Projection using Partial SVD

Recall that for a proper convex function f, its proximal operator is defined as [25] $\mathbf{prox}_f(W) = \operatorname{argmin}_U \left\{ \frac{1}{2} \|W - U\|_F^2 + f(U) \right\}$. Let f^* be the convex conjugate of f, the Moreau decomposition [25], [29] gives

$W = \mathbf{prox}_f(W) + \mathbf{prox}_{f^*}(W)$. For our algorithm, we know that the convex conjugate of indicator function $\mathbb{I}_{\Omega_i}(W)$ is $\lambda\|W\|_*$, and we can compute the projection as

$$\Pi_{\Omega_i}(W) = W - \mathbf{prox}_{\lambda\|\cdot\|_*}(W) \ . \tag{28}$$

Note that the proximal operator in the r.h.s. of (28) involves SVT. To speed up the computation in practice, one can do adaptive fixed rank SVD decompositions, say rank sv_i^t for the i-th block and step t, and adjusting the rank for the next iteration based on [21, Section 3]

$$\mathrm{sv}_i^{t+1} = \begin{cases} \mathrm{svp}_i^t + 1 & \mathrm{svp}_i^t < \mathrm{sv}_i^t \\ \min(\mathrm{svp}_i^t + \mathrm{round}(0.05|\mathcal{G}_i|, |\mathcal{G}_i|)) & \mathrm{svp}_i^t = \mathrm{sv}_i^t \end{cases} \tag{29}$$

where svp_i^t is the number of singular values in the sv_i^t singular values that are larger than λ.

B. Acceleration Scheme

We apply Nesterov's acceleration scheme introduced in [18], [20]. The scheme introduced a combination coefficient θ^t. Starting from 0, this coefficient is given by $\theta^{t+1} = \frac{1+\sqrt{1+4(\theta^t)^2}}{2}$. In each update, in addition to pure gradient update, the output is given by a combination $\tilde{Y}_i^{t+1} = Y_i^{t+1} + \frac{\theta^t - 1}{\theta^{t+1}}(Y_i^{t+1} - Y_i^t)$, where Y_i^{t+1} and Y_i^t are gradient updates. It is unclear if this algorithm will have a linear convergence rate. Therefore we can consider a hybrid strategy which, after a few steps of Nesterov update, considers a basic gradient update if that leads to a lower objective. One way to determine which update rule we should use is by adaptive restarting. Based on [30], in each step we obtain both the gradient update and the accelerated update. We compare the objective value, and decide whether to do a gradient step or continue with acceleration. If we choose to do pure gradient update, we also reset the combination coefficient to 0. We refer to this method as Adaptive Acceleration (AA).The linear convergence rate of AA is stated in Corollary 10 and follows from Theorem 6. Empirical results in Section VI-A3 also shows that AA improves over the performance of both PGD and ACC.

Corollary 10: Suppose there exists an $\{Y_i^*\}_{i=1}^L \in \mathcal{Y}$ such that condition (15) holds, then algorithm AA converges linearly.

IV. Alternative Algorithms for LLR Modeling

In this section, we briefly discuss three other optimization algorithms for LLR modeling, respectively based on the alternating direction method of multiplier (ADMM), block coordinate descent (BCD), and blockwise singular value thresholding (B-SVT).

A. ADMM Algorithm

The optimization (3) is challenging due to the overlap of the subsets. A simple approach is to decouple the overall problem into smaller problems corresponding to the sub-matrices and iteratively solve the smaller subproblems. A direct decoupling reformulation of (3) is

$$\min_{X, \{X_i\}_{i=1}^L} \ \frac{1}{2}\|Z - X\|_F^2 + \lambda \sum_{i=1}^L \|X_i\|_* \quad \text{s.t.} \quad Q_{\mathcal{G}_i} X = X_i \ , \tag{30}$$

which can be solved using the Alternating Direction Method of Multipliers (ADMM) [22]. The main steps of ADMM are two primal steps

$$X_i^{t+1} = \mathrm{argmin}_{X_i} \ \frac{1}{2}\|Q_{\Omega_i} X^t + Y_i^t - X_i\|_F^2 + \frac{\lambda}{\rho}\|X_i\|_*,$$

$$i = 1, \ldots, L$$

$$X^{t+1} = (I + \rho \sum_{i=1}^L Q_i^T Q_i)^{-1}(Z + \rho \sum_{i=1}^L Q_i^T X_i^t - \rho \sum_{i=1}^L Q_i^T Y_i^t)$$

and a dual step $Y_i^{t+1} = Y_i^t + (Q_{\mathcal{G}_i} X - X_i)$. The matrix inverse in the X^{t+1} can be pre-computed. The updates for X_i^{t+1} can be solved using SVT as discussed in Section I, but can be slow if the sub-matrices are large. However, the X_i^{t+1} updates can in principle be done in parallel over $i = 1, \ldots, L$. One needs to choose the parameter $\rho > 0$, and in practice the convergence can vary based on this choice. The ADMM algorithm is given by Algorithm 2

Algorithm 2 Alternating Direction Method of Multipliers (ADMM)

Inputs: $\{Q_{\mathcal{G}_i}\}_{i=1}^L$, Z; $\rho > 0$
Initialize: $\{Y_i^0\}_{i=1}^L = \{0\}$.
for $t = 0, 1, \ldots, T$ **do**
  for $i = 1, \ldots, L$ **do**
    $X_i^{t+1} = \mathrm{argmin}_{X_i} \ \frac{1}{2}\|Q_{\Omega_i^\lambda} X^t + Y_i^t - X_i\|_F^2 + \frac{\lambda}{\rho}\|X_i\|_*$
  end for
  $X^{t+1} = (I + \rho \sum_{i=1}^L Q_i^T Q_i)^{-1}(Z + \rho \sum_{i=1}^L Q_i^T X_i^t - \rho \sum_{i=1}^L Q_i^T Y_i^t)$
  $Y_i^{t+1} = Y_i^t + (Q_{\mathcal{G}_i} X - X_i)$
end for

B. BCD Algorithm

For dual problem (6), instead of updating all blocks at the same time like DPGD, we can only update one or several blocks. We fixed the other blocks, and minimizing the objective function with respect to the chosen blocks. The BCD algorithm is given by Algorithm 3.

C. B-SVT Algorithm

BSVT [1] is another method for denoising. In BSVT, we first solve singular value thresholding on each block

$$\min_{X_{\mathcal{G}_i}} \frac{1}{2}\|Q_{\mathcal{G}_i} Z - X_{\mathcal{G}_i}\|_F^2 + d_{\max}\lambda\|X_{\mathcal{G}_i}\|_* \tag{31}$$

then we take the average of all solutions

$$\hat{X} = \frac{1}{d_{\max}} Q_{\mathcal{G}_i}^T X_{\mathcal{G}_i} \tag{32}$$

Algorithm 3 Block Coordinate Descent (BCD)

Inputs: $\{Q_{\mathcal{G}_i}\}_{i=1}^L$, Z.
Initialize: $\{Y_i^0\}_{i=1}^L = \{0\}$, $\theta_0 = 0$, and choose a step size $\rho > 0$.
for $t = 0, 1, \ldots, T$ **do**
  choose some blocks $\mathcal{B}_t$
  for $i = 1, \ldots, L$ **do**
    if $i \in \mathcal{B}_t$ **then**
      $Y_i^{t+1} = \Pi_{\Omega_i^\lambda}(Y_i^t - \rho \cdot Q_{\mathcal{G}_i}(\sum_{i'=1}^L Q_{\mathcal{G}_{i'}}^T Y_{i'}^t - Z)))$
    else
      $Y_i^{t+1} = Y_i^t$
    end if
  end for
end for

BSVT is a one pass algorithm therefore we do not need iterations.

V. Application: MRI Denoising

In this section, we consider an important real world application for LLR modeling: Magnetic Resonance Imaging (MRI) denoising. The proposed algorithms were applied to locally low rank image denoising for both time-resolved (i.e., dynamic) cardiac MR imaging [15] and multi-channel brain imaging [12], two applications where locally low rank-based processing techniques have proven beneficial. We briefly discuss the datasets, and also how to make use of parallel updates and BCD in these applications.

Cardiac MRI. The first data set tested represents a first pass myocardial perfusion cardiac MRI exam performed at 3.0 T using a standard gradient-recalled echo (GRE) acquisition sequence, wherein imaging was continuously performed following intravascular contrast administration to observe. In such cases, image quality (i.e., SNR) is typically sacrificed in exchange for high spatiotemporal resolution. The second and third data sets represent short- and long axis cine cardiac exams acquired using a 2D balanced steady state free precession (bSSFP) sequence at 1.5 T. Due to RF specific absorption ratio (SAR) limitations, cine cardiac imaging is often performed at lower field strengths, which in turn limits the amount of signal generated during the scan. For these data sets, the auxiliary non-spatial dimension considered for locally low rank processing was time. For additional details about these time-resolved cardiac data sets, see [1], [9].

Brain MRI. The proposed algorithms were also tested on two T1-weighted brain image data sets obtained using a T1-weighted 3D spoiled gradient echo (SPGR) sequence with an 8-channel phased array receiver and RF flip angle of $\theta = 25$. For these data sets, the auxiliary non-spatial dimension considered for locally low rank processing was receiver channel index. For additional details about these multi-channel data sets, see [31].

Formulation Consider a series of T frames of $N_x \times N_y$ 2D MR images in complex number. To apply local low rank model, we first transform it into a $N_x N_y \times T$ Casorati matrix $Z = X^* + W$.

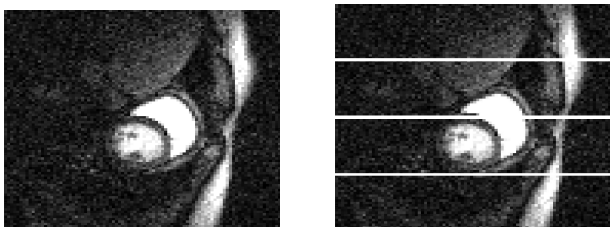


Fig. 1: In our parallel implementation, we only decompose along the rows of the image. In this picture, we want to denoise the image on the left hand side using 4 cores, so we first decompose it like the right hand side.

Denote $\mathcal{G}_{ij}$ an image block indexed by (i, j), and the size of $\mathcal{G}_{ij}$ is $B \times B$. In MR series, $\mathcal{G}_{ij}$ starts from the pixel located at (i, j), and extends to the $i + B$-th column and the $j + B$-th row. Operator $Q_{\mathcal{G}_{ij}}$ extracts a $B^2 \times T$ submatrix from Z with rows corresponding to block $\mathcal{G}_{ij}$ in the series. In our application for MRI, we construct a block for each pixel in the MR image by using cyclic way.

Parallel Updates for MRI We designed a parallel D-PGD algorithm to work in a distributed memory environment. In previous works for parallel proximal algorithm, one needs an all reduce operation in each step [22], [23]. In this work, we propose a decomposition method for MRI to avoid the all reduce operation in our parallel implementation, therefore the speed up through parallelization can be almost linear. Our decomposition method makes use of structure specific to MR images to reduce the overhead.

From our algorithm 1, update of dual variable Y_{ij} corresponding to block $\mathcal{G}_{ij}$ is given by

$$Y_{ij}^{t+1} = \Pi_{\Omega_{ij}^\lambda}(Y_{ij}^t - \rho \cdot Q_{\mathcal{G}_{ij}}(\sum_{i,j} Q_{\mathcal{G}_{ij}}^T Y_{ij}^t - Z))). \quad (33)$$

In a parallel algorithm of update (33), communication between cores is required when computing $\sum_{i,j} Q_{\mathcal{G}_{ij}}^T Y_{ij}^t$, and we need all blocks to get it. However, in our application we only need $Q_{\mathcal{G}_{ij}} \sum_{i,j} Q_{\mathcal{G}_{ij}}^T Y_{ij}^t$. Therefore we can merely compute $\sum_{\mathcal{G}_{i'j'} \cap \mathcal{G}_{ij} \neq \emptyset} Q_{\mathcal{G}_{i'j'}}^T Y_{i'j'}^t$ in one core, and communicate Y_{ij} overlapping with other cores.

In our algorithm, we decompose the image along the rows (see figure 1), so that there will be no communication along rows indexed by j. Denote $\sum_{i,j} Q_{\mathcal{G}_{ij}}^T Y_{ij}^t$ by H^t. Suppose we have P cores. In the p-th core ($p = 1 \ldots P$), we store row $(p-1) \cdot \frac{N_x}{P} + 1$ to row $p \cdot \frac{N_x}{P} + B - 1$ of H^t, and denote this submatrix as H_p^t. We can compute all Y_{ij}^{t+1} with i indexed by $(p-1) \cdot \frac{N_x}{P} + 1$ through $p \cdot \frac{N_x}{P}$ using H_p^t. Then in core p, we introduce a matrix G_p^{t+1}, and we update it locally by

$$G_p^{t+1} = \sum_{i=(p-1)\cdot\frac{N_x}{P}+1}^{p\cdot\frac{N_x}{P}} \sum_j Q_{\mathcal{G}_{ij}}^T Y_{ij}^t$$

Notice that $G_p^{t+1} \neq H_p^{t+1}$ since in the first and last $B-1$ rows of G_p^{t+1} we need to add Y^{t+1} from core $p-1$ and core

$p+1$. By using matrix G_p^{t+1}, we can simply do this by

$$H_p^{t+1}(1\colon B-1,:) = G_p^{t+1}(1\colon B-1,:)+G_{p-1}^{t+1}(\frac{N_x}{P}-B+1\colon \frac{N_x}{P},:)$$

where for a matrix A we denote $A(i\colon j,:)$ as a submatrix extracted from the i-th to the j-th rows of each frame of corresponding MR series. We do this by sending $G_{p-1}^{t+1}(\frac{N_x}{P} - B+1\colon \frac{N_x}{P},:)$ to the p-th core, and the send operation can be done from all cores in parallel. Then we get the last $B-1$ rows of H_p^{t+1} by

$$H_p^{t+1}(\frac{N_x}{P} - B+1\colon \frac{N_x}{P},:) = H_{p+1}^{t+1}(1\colon B-1,:)$$

We only need to send $H_{p+1}^{t+1}(1\colon B-1,:)$ to the p-th core for all possible p in parallel. In our algorithm, the overlapping part has memory continuity, and we have no pain manipulating such data. In theory, the speed up of our parallel implementation will be linear.

Another method we can apply to MRI series is block coordinate descent (BCD). One way to do this is to randomly choose some blocks and do gradient descent on these blocks. However this does not work for our application because it has too many blocks. If the number of blocks we choose is small then it will take too many steps to update and too long to extract these blocks. Our method here is to first divide the 2D matrix into many non-overlapping blocks. In other words, for any two blocks $\mathcal{G}_i$ and $\mathcal{G}_j$ in this update, we have $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$. We update these blocks in parallel. In the next step we modify our previous division into blocks, work with new non-overlapping blocks, and do another update. The process continues till all blocks have been updated. In this way we can make the parallel updates communication efficient.

VI. Experimental Results

In this section, we focus on experiments to illustrate the efficiency of our algorithms. We do a comparison of different algorithms on synthetic matrices of different sizes.

A. Synthetic Data

We fix $n = 3000$, and choose $m = 1000$, $m = 3000$ and $m = 5000$. We use a sliding window of 1000 adjacent rows and stepsize 250, so that the set of row subsets with low rank are

$$\mathcal{G} = \{\{1,\ldots,1000\},\{251,\ldots,1250\},\ldots,\{(J+1)\bmod m,\ldots,(J+1000)\bmod m\}\}$$

where $J = 250(|\mathcal{G}|-1)$. We sample the local low-rank matrix in a hierarchically way, and each low rank matrix is generated in the same way as [25] [1]. We first sample a rank 2 matrix X_1 of size $n \times m$, then we sample a rank 1 matrix $X_2^{(j)}$ on each small block of size $250 \times m$ for $j = 1,\ldots,|\mathcal{G}|$. At last we sample a rank 1 matrix $X_{\mathcal{G}_i}$ on each group for $i = 1,\ldots,|\mathcal{G}|$. Matrix X^* is the sum up of all the samples, that is

$$X^* = X_1 + \begin{pmatrix} X_2^{(1)} \\ \vdots \\ X_2^{(|\mathcal{G}|)} \end{pmatrix} + Q_{\mathcal{G}_1}^T X_{\mathcal{G}_1} + \ldots + Q_{\mathcal{G}_{|\mathcal{G}|}}^T X_{\mathcal{G}_\mathcal{G}} \ . \quad (34)$$

The rank of each block $\mathcal{G}_i$ is thus less that 10. We sample a noise matrix E with i.i.d. entries from a Gaussian distribution, and generate the output from $Z = X^* + \frac{\sigma}{\sqrt{m}+\sqrt{n}}E$, where σ is the standard deviation.

1) Setup: In our experiment, we set $\lambda = \frac{\sigma}{d_{max}}$. We use the Frobenius norm $\epsilon^t = \|X^t - X^*\|_F$ to measure the closeness of X^t to X^*, where $X^t = Z - \sum_{i=1}^L Q_{\mathcal{G}_i}^T Y_i^t$, and X^* is the "optimal" solution to problem (3) after running algorithm AA for a long time. In our experiment, we stop our algorithm when absolute error is less than 10^{-5}. We also compare the convergence behavior of primal objective function error $g(X^t) - g(X^*)$. For ADMM, stepsize ρ is chosen to be $\frac{1}{d_{\max}}$, and only full SVD is used. This experiment is performed on core i7-6700 4 cores with 16 GB memory, our program is implemented using Matlab.

2) Effect of Partial SVD: In this experiment, we test the performance of partial SVD on synthetic data. Figure 2 shows the convergence and running time of D-PGD using partial SVD and full SVD when $n = 5000$. Both algorithms stop after similar number of iterations. However, we can see that D-PGD with partial SVD considerably outperforms D-PGD with full SVD in terms of actual running time.

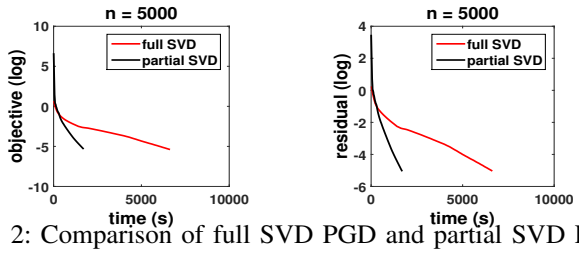


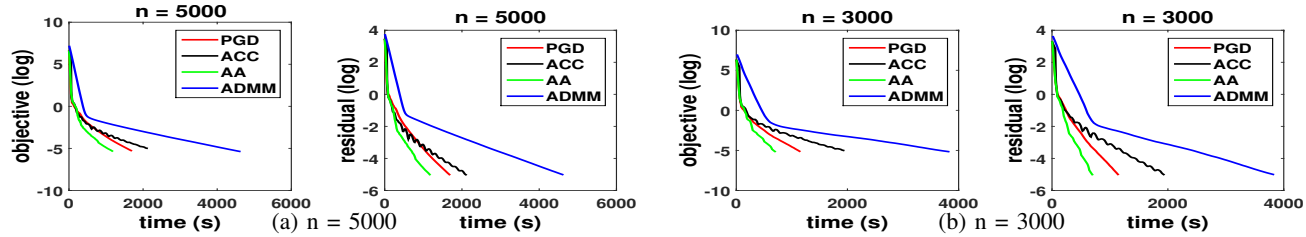
Fig. 2: Comparison of full SVD PGD and partial SVD PGD in computational timewhen $n = 5000$. Both algorithm use the same number of iterations, and converge linearly. We can see that Partial SVD based D-PGD uses considerably less time.

3) Performance Comparison: In this experiment, we compare the performance of different algorithms on synthetic data. We use partial SVD for D-PGD, ACC and AA. From Figure 3, we see that D-PGD, ACC and ADMM take similar number of iterations to converge. Interestingly, ACC has similar or sometimes worse behavior compared to D-PGD and ADMM in terms of number of iterations, and worse than D-PGD in terms of time. ADMM looks bad in terms of time, but that can be attributed to ADMM using full SVD. Interestingly, the AA algorithm outperforms all the others, both in number of iterations and running time.

B. Sequential Deonising

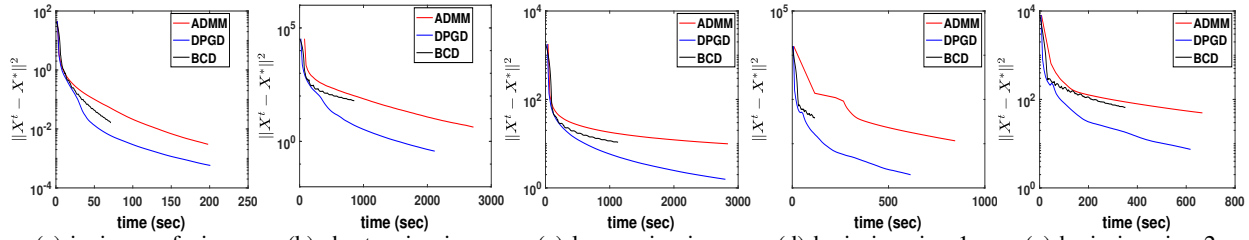
In this experiment, we apply our algorithm to denoising dynamic cardiac MRI series and brain image. The size of our dataset is given in table (I). We have three cardiac dataset:

[1] http://stanford.edu/~boyd/papers/prox_algs/matrix_decomp.html



(a) n = 5000 (b) n = 3000

Fig. 3: Comparison of PGD, ACC, AA, and ADMM in terms of number of iterations and computational time when $n = 5000$ and $n = 3000$. All algorithms converge linearly. AA is the fastest algorithm in running time.



(a) invivo_perfusion (b) short_axis_cine (c) long_axis_cine (d) brain imaging 1 (e) brain imaging 2

Fig. 4: Comparison of ADMM, DPDG, and BCD in terms of computational time on different dataset. We can see that on all datasets DPGD outperforms other algorithms.

Dataset	N_x	N_y	N_c	T
invivo_perfusion	90	90	1	30
short_axis	144	192	8	19
long_axis	192	192	8	19
brain imaging 1	256	256	8	1
brain imaging 2	256	256	8	1

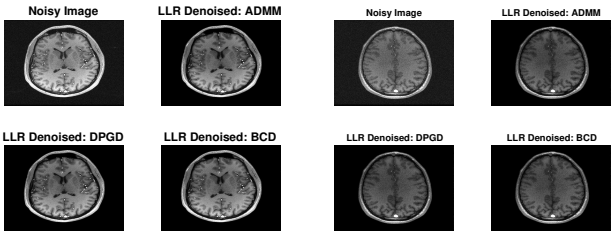
TABLE I: Size of different set

invivo perfusion, short axis, and long axis. The other two datasets are brain imaging dataset.

1) Setup: We compare the speed of the three algorithms: ADMM, DPGD with adaptive acceleration, and PGD. We first run our DPGD algorithm for a very long time to get an "optimal" solution X^*, then we compare the immediate result of each iteration of our algorithms to X^*. Parameter λ is given by "cross validation", we run our algorithm using different λ, and the best λ is chosen by experts by looking at the denoised images. The best λ we get are: invivo_perfusion, 0.05; short_axis, 20; long_axis, 1; brain imaging 1, 25; brain imaging 2, 10. We choose block size B to be 5 for all dataset. For ADMM, we choose its step size ρ to be $\frac{1}{d_{\max}}$. For BCD, we choose its step size ρ to be 1. We run ADMM and DPGD for 100 iterations and BCD for 200 iterations. This experiment is performed on core i7-6700 4 cores with 16 GB memory, our program is implemented using Matlab.

2) Performance Comparison: We plot running time optimization error $\|X^t - X^*\|^2$ of different algorithms. We present our plot in figure 4, in this plot $\|X^t - X^*\|^2$ is in log scale. From figure 4 we can see that all three algorithms converge linearly. The plot also shows the efficiency of the three algorithms. BCD takes much less time for each iteration, but it is not as good as DPGD. Besides, the BCD curve oscillate while going down. The performance of Vanilla ADMM is not as good. Our algorithm is the most efficient among the three algorithms.

3) Effect of Our Algorithm: We make use of the result in section (VI-B2), and show the quality of denoised images. For cardiac MRI series, we choose one frame from each of them. The result is presented as figure 6. For braining imaging data, we show both of them. The result is presented as figure 5. From these results we can see that the denoised image become clearer than the noisy image. We also know from the result in (VI-B2) that to reach the same denoising quality, DPGD takes less time than ADMM and BCD.



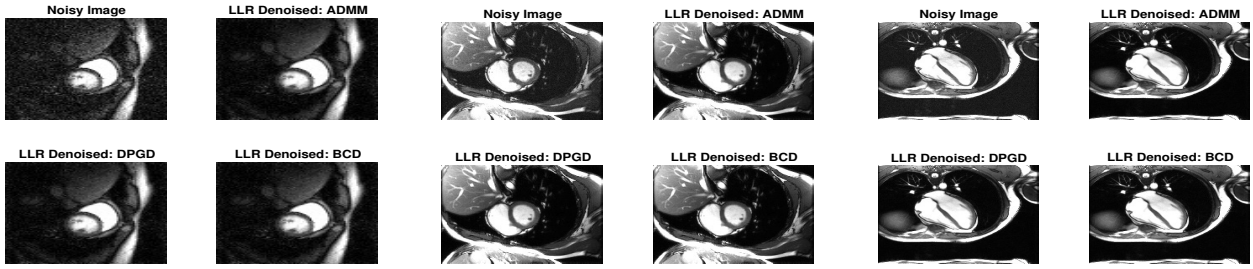
(a) brain imaging 1 (b) brain imaging 2

Fig. 5: Comparison of denoised images using using ADMM, DPGD with adaptive acceleration, and BCD in Brain Imaging. Original noisy images (upper left); Denoised images using ADMM (upper right); Denoised images using DPGD (bottom left); Denoised images using BCD (bottom right).

4) Comparison with BSVT: We compare LLR denoising with BSVT. Our result is presented as Figure 7. In figure 7, we show one frame from each cardiac MRI series and visualization of two brain imaging datasets. In the top row we show the noisy image without denoising, in the middle we show the denoising result of $BSVT$, and in the last row we show the denoising result of locally low rank. As BSVT is a heuristic algorithm, it is not guaranteed to solve the locally low-rank problem. Besides, DPGD can be applied to problems with missing values.

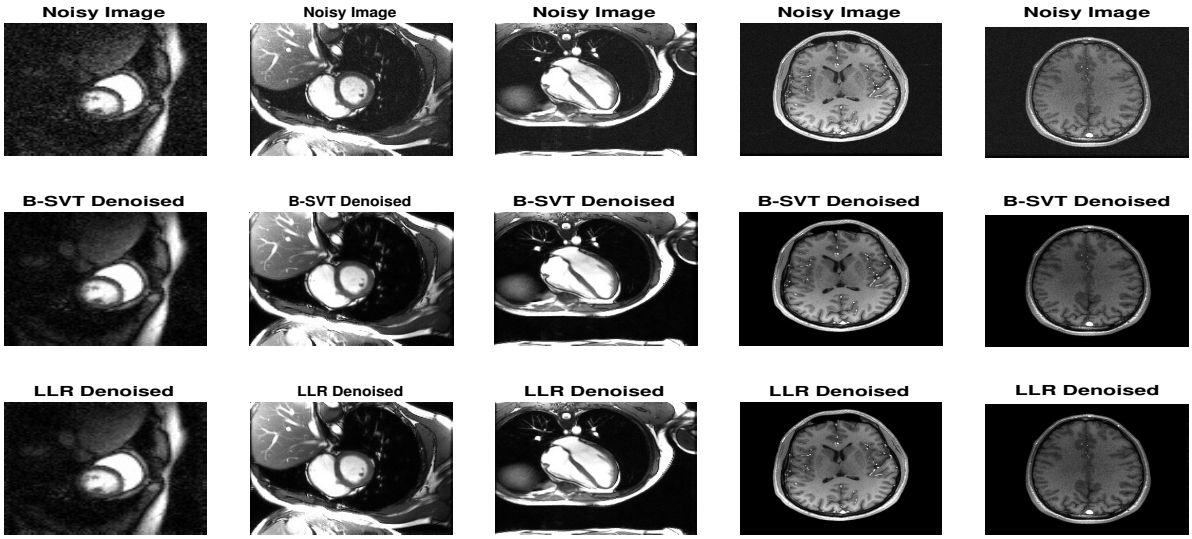
C. Parallel Denoising

In this experiment, we apply parallel update for DPGD to the same dataset in section (VI-B). We show show the speed up of our algorithm giving increasing number of cores.



(a) invivo_perfusion (b) short_axis (c) long_axis

Fig. 6: Comparison of denoised frames using ADMM, DPGD with adaptive acceleration, and BCD in cardiac MRI. Original noisy images (upper left); Denoised images using ADMM (upper right); Denoised images using DPGD (bottom left); Denoised images using BCD (bottom right).



(a) invivo_perfusion (b) short_axis (c) long_axis (d) brain imaging 1 (e) brain imaging 2

Fig. 7: Comparison of BSVT denoised image and LLR denoised image. Original images (top), Denoised images using BSVT (middle), Denoised images using LLR (bottom)

1) Setup: We implemented a parallel version of DPGD algorithm with adaptive acceleration using C, Open BLAS, and Open MPI. We run our code on a node with 32 cores provided by Intel Haswell E5-2680v3 processors and 16 GB memory. In each run we update 50 iterations.

2) Effect of Parallel Update: We present our result in figure (8). In this figure the top row shows the plot of running time versus number of cores, while the row below shows the plot of speed up versus number cores. In the top figures the blue circles are the number of cores that we used. In the figures below the red line is the ideal speed up, that the speed up is the same as number of cores. From these figures we can see that the speedup of our algorithm is almost linear with the number of cores. This accord with our discussion in section (V), because communication time should not change with number of cores.

VII. Conclusions

In this paper, we considered the locally low rank (LLR) matrix denoising problem, and proposed an efficient algorithm D-PGD for solving the problem. Even though the formulation is not strongly convex, we proved that D-PGD converges linearly under mild assumptions. We introduced strategies to speed up the algorithms in practice and illustrated performance on synthetic and real data. Applications of the LLR formulation for missing data reconstruction will be investigated in future work.

Acknowledgments

The research was supported by NSF grants IIS-1447566, IIS-1422557, CCF-1451986, CNS-1314560, IIS-0953274, IIS-1029711, NASA grant NNX12AQ39A, NSF CCF:CIF:Small:1318347 and "Mayo Clinic Discovery Translation Program".

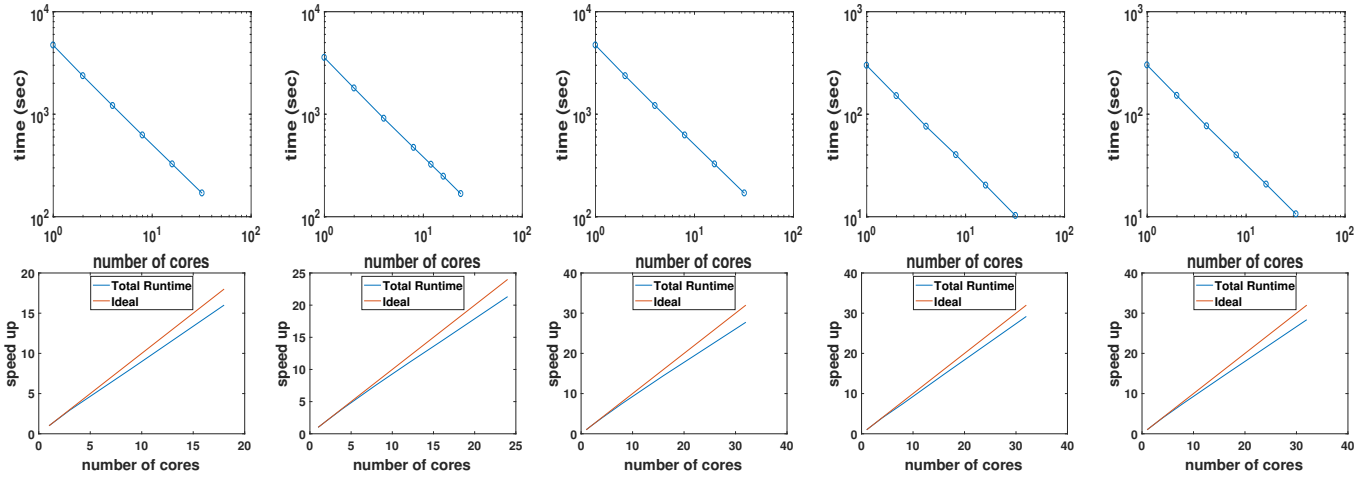
References

[1] E. J. Candès, C. A. Sing-long, and J. D. Trzasko, "Unbiased Risk Estimates for Singular Value Thresholding and Spectral Estimators," *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4643–4657, 2013.

[2] D. Donoho and M. Gavish, "Minimax risk of matrix denoising by singular value thresholding," *Annals of Statistics*, vol. 42, no. 6, pp. 2413–2440, 2014.

[3] J. Lee, S. Kim, G. Lebanon, Y. Singer, and S. Bengio, "Llorma: Local low-rank matrix approximation," *Journal of Machine Learning Research*, vol. 17, no. 15, pp. 1–24, 2016.

[4] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.



(a) invivo_perfusion (b) short_axis (c) long_axis (d) brain imaging 1 (e) brain imaging 2

Fig. 8: Running time of parallel DPGD using different number of cores (top). Almost linear speed up in the number of cores (down).

[5] S. Gunasekar, A. Banerjee, and J. Ghosh, "Unified view of matrix completion under general structural constraints," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 1180–1188.

[6] S. Negahban and M. J. Wainwright, "Estimation of (near) low-rank matrices with noise and high-dimensional scaling," *The Annals of Statistics*, vol. 39, no. 2, pp. 1069–1097, Apr. 2011.

[7] S. Chen and A. Banerjee, "Structured matrix recovery via the generalized dantzig selector," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3252–3260. [Online]. Available: http://papers.nips.cc/paper/6394-structured-matrix-recovery-via-the-generalized-dantzig-selector.pdf

[8] A. Banerjee, S. Chen, F. Fazayeli, and V. Sivakumar, "Estimation with Norm Regularization," in *Advances in Neural Information Processing Systems*, 2014.

[9] S. Goud, Y. Hu, and M. Jacob, "Real-time cardiac MRI using low-rank and sparsity penalties," in *ISBI*, 2010, pp. 988–991.

[10] J. P. Haldar and Z.-P. Liang, "Spatiotemporal imaging with partially separable functions: A matrix recovery approach," in *ISBI*, 2010, pp. 716–719.

[11] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[12] J. D. Trzasko and A. Manduca, "Calibrationless parallel mri using clear," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Nov 2011, pp. 75–79.

[13] Q. Yao and J. T. Kwok, "Colorization by patch-based local low-rank matrix completion," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, pp. 1959–1965. [Online]. Available: http://dl.acm.org/citation.cfm?id=2886521.2886593

[14] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and bregman iterative methods for matrix rank minimization," *Mathematical Programming*, vol. 128, no. 1, pp. 321–353, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10107-009-0306-5

[15] J. D. Trzasko, "Exploiting local low-rank structure in higher-dimensional mri applications," pp. 885 821–885 821–8, 2013.

[16] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 426–434.

[17] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[18] Y. Nesterov, *Introductory lectures on convex optimization : a basic course*, ser. Applied optimization. Boston, Dordrecht, London: Kluwer Academic Publ., 2004.

[19] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of Optimization*, 2010. [Online]. Available: http://www.math.nus.edu.sg/~{}mattohkc/papers/mc11.pdf

[20] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009. [Online]. Available: http://dx.doi.org/10.1137/080716542

[21] Z. Lin, R. Liu, and Z. Su, "Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation," in *Advances in Neural Information Processing Systems 24*, no. 1, 2011, pp. 1–9. [Online]. Available: http://arxiv.org/abs/1109.0367

[22] S. 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, 2010.

[23] Z. Peng, M. Yan, and W. Yin, "Parallel and distributed sparse optimization," in *2013 Asilomar Conference on Signals, Systems and Computers*, Nov 2013, pp. 659–646.

[24] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010. [Online]. Available: https://doi.org/10.1137/070697835

[25] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[26] Z. Zhou and A. M.-C. So, "A Unified Approach to Error Bounds for Structured Convex Optimization Problems," *arXiv:1512.03518*, pp. 1–32, 2015. [Online]. Available: http://arxiv.org/abs/1512.03518$\delimiter"026E30F$nhttp://www.arxiv.org/pdf/1512.03518.pdf

[27] A. J. Hoffman, " On Approximate Solutions of Systems of Linear Inequalities," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 4, pp. 263–265, 1952.

[28] Z. Q. Luo and P. Tseng, "Error bounds and convergence analysis of feasible descent methods: a general approach," *Annals of Operations Research*, vol. 46-47, no. 1, pp. 157–178, 1993.

[29] J. J. Moreau, "Decomposition orthogonale d'un espace hilbertien selon deux cones mutuellement polaires." *Comptes Rendus de l'Académie des Sciences*, vol. 255, pp. 238–240, 1962.

[30] B. O'Donoghue and E. Candés, "Adaptive Restart for Accelerated Gradient Schemes," *Foundations of Computational Mathematics*, vol. 15, no. 3, pp. 715–732, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10208-013-9150-3

[31] J. D. Trzasko, P. M. Mostardi, S. J. Riederer, and A. Manduca, "Estimating t1 from multichannel variable flip angle SPGR sequences," *Magnetic Resonance in Medicine*, vol. 69, no. 6, pp. 1787–1794, 6 2013.