FaLRR: A Fast Low Rank Representation Solver

Shijie Xiao[†], Wen Li[†], Dong Xu[†], Dacheng Tao[‡]
[†]School of Computer Engineering, Nanyang Technological University, Singapore
[‡]Centre for Quantum Computation & Intelligent Systems and the Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia
{XIAO0050, WLI1, DongXu}@ntu.edu.sg, Dacheng.Tao@uts.edu.au

Abstract

Low rank representation (LRR) has shown promising performance for various computer vision applications such as face clustering. Existing algorithms for solving LRR usually depend on its two-variable formulation which contains the original data matrix. In this paper, we develop a fast LRR solver called FaLRR, by reformulating LRR as a new optimization problem with regard to factorized data (which is obtained by skinny SVD of the original data matrix). The new formulation benefits the corresponding optimization and theoretical analysis. Specifically, to solve the resultant optimization problem, we propose a new algorithm which is not only efficient but also theoretically guaranteed to obtain a globally optimal solution. Regarding the theoretical analysis, the new formulation is helpful for deriving some interesting properties of LRR. Last but not least, the proposed algorithm can be readily incorporated into an existing distributed framework of LRR for further acceleration. Extensive experiments on synthetic and real-world datasets demonstrate that our FaLRR achieves order-of-magnitude speedup over existing LRR solvers, and the efficiency can be further improved by incorporating our algorithm into the distributed framework of LRR.

1. Introduction

Data in many real-world problems often lies in a union of low-dimensional subspaces. Given data sampled from multiple subspaces, the goal of *subspace clustering* is to partition the data into a pre-defined number of groups, such that each group corresponds to exactly one subspace. The subspace clustering problem has been extensively studied [20, 7, 14] for various real-world applications such as face clustering [14, 7]. Recently, there are growing research interests in the spectral clustering based methods [7, 14], which solve the subspace clustering problem by applying spectral clustering [21, 15] on a desired similarity matrix.

Among the spectral clustering based approaches, the low-rank representation (LRR) method [14, 13] aims to learn a low rank data representation matrix for constructing the desired similarity matrix. Based on a convex formulation, LRR is robust to noise (see [13] and references therein), and thus it has achieved promising performance in dealing with real-world vision problems [14, 13, 9].

Regarding optimization, several algorithms [14, 12, 13] were proposed to exactly solve LRR. Moreover, to efficiently obtain an approximated solution of LRR, a distributed framework [17] was developed. However, existing algorithms are usually based on the original formulation [14] or a similar variant [13], either of which is a two-variable problem with regard to the original data matrix.

In this paper, we reformulate LRR based on the factorized data, so that the resultant problem contains the components from skinny singular value decomposition (SVD) of the original data matrix. Interestingly, the new problem benefits both the optimization and the theoretical study. In terms of optimization, we develop an efficient algorithm based on the alternating direction method (ADM) [2, 12], in which both resultant subproblems can be solved exactly. We show that our new optimization algorithm achieves a global optimum in theory and it outperforms the existing LRR solvers [13, 12] in terms of efficiency. Moreover, based on the new form, we provide theoretical analysis regarding the influence of the parameter in LRR. Last but not least, the proposed algorithm can be readily incorporated into the distributed framework [17] of LRR, to further improve the efficiency.

Notations: For convenient presentation, we denote a vector (*resp.*, matrix) with a lowercase (*resp.*, uppercase) letter in boldface. The transpose of a vector/matrix is denoted by the superscript $'$. $\mathbf{I}_n \in \mathbb{R}^{n\times n}$, $\mathbf{O}_{m\times n} \in \mathbb{R}^{m\times n}$ denote the $n \times n$ identity matrix and the $m \times n$ zero matrix, respectively. $\mathbf{0}_n \in \mathbb{R}^n$, $\mathbf{1}_n \in \mathbb{R}^n$ denote the column vectors with all elements being zeros and ones, respectively. For simplicity, we drop the subscripts of $\mathbf{I}_n$, $\mathbf{O}_{m\times n}$, $\mathbf{0}_n$ and $\mathbf{1}_n$ when the dimension is obvious.

The norms of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ are defined as follows. $\|\mathbf{A}\|_F = \sqrt{trace(\mathbf{A}\mathbf{A}')}$ is the Frobenius norm. $\|\mathbf{A}\|_*$ is the nuclear norm, which is the sum of singular values of $\mathbf{A}$. $\|\mathbf{A}\|_2$ is the spectral norm, which is the maximal singular value of $\mathbf{A}$. $\|\mathbf{A}\|_{max}$ is the max norm, which is the maximal absolute value of elements in $\mathbf{A}$. The $\ell_{2,1}$ norm [18] is defined as $\|\mathbf{A}\|_{2,1} = \sum_{i=1}^{n} \|\mathbf{a}_i\|$, where $\mathbf{a}_i \in \mathbb{R}^m$ is the i-th column of $\mathbf{A}$, and $\|\mathbf{a}\| = \sqrt{\mathbf{a}'\mathbf{a}}$ is the ℓ_2 norm of a vector $\mathbf{a}$. Besides, $rank(\mathbf{A})$ denotes the rank of matrix $\mathbf{A}$, $\langle \mathbf{A}, \mathbf{B} \rangle$ denotes the inner product of two matrices $\mathbf{A}$ and $\mathbf{B}$ (*i.e.*, $\langle \mathbf{A}, \mathbf{B} \rangle = trace(\mathbf{A}'\mathbf{B})$), and $diag(\cdot)$ denotes the diagonalization operator by converting a vector into a diagonal matrix.

2. The LRR Problem

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ be a set of data samples drawn from a union of several subspaces, where d is the feature dimension and n is the total number of data samples. LRR [14, 13] seeks a low-rank data representation matrix $\mathbf{Z} \in \mathbb{R}^{n\times n}$ such that $\mathbf{X}$ can be self-expressed (*i.e.*, $\mathbf{X} = \mathbf{X}\mathbf{Z}$) if the data is clean. Considering that the input data may contain outliers (*i.e.*, some columns of $\mathbf{X}$ may be corrupted), the LRR problem can be formulated as,

$$\min_{\mathbf{Z},\mathbf{E}} \quad \|\mathbf{Z}\|_* + \lambda\|\mathbf{E}\|_{2,1} \tag{1}$$

$$\text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E}, \tag{2}$$

where λ is a tradeoff parameter and $\mathbf{E} \in \mathbb{R}^{d\times n}$ denotes the representation error. The nuclear norm based term $\|\mathbf{Z}\|_*$ acts as an approximation of the rank regularizer [25, 24, 19], and $\ell_{2,1}$ norm based term $\|\mathbf{E}\|_{2,1}$ encourages $\mathbf{E}$ to be column-sparse.

Liu *et al*. [14] proposed a three-block Alternating Direction Method (3BADM) to solve the problem in (1). In particular, due to the term $\mathbf{X}\mathbf{Z}$ in the constraint, it is nontrivial to directly apply ADM for efficiently solving the original LRR problem. So they introduce an auxiliary variable $\mathbf{J} = \mathbf{Z}$ to solve the problem using ADM with three blocks of variables. However, this LRR solver suffers from $\mathcal{O}(n^3)$ computational complexity per iteration, and the introduction of the auxiliary variable may slow down the convergence and increase the memory consumption [12]. Moreover, it is difficult to generally ensure the convergence of ADM with three or more blocks [13, 12].

To address the above drawbacks, Lin *et al*. [12] proposed the accelerated linearized alternating direction method with adaptive penalty (aLADMAP) to solve the LRR problem, in which they solve LRR by using a variant of ADM. Specifically, to address the challenging subproblem w.r.t. $\mathbf{Z}$, they proposed to linearize the quadratic term in the subproblem. Moreover, to efficiently solve this subproblem, they use a strategy [12] to predict the rank of $\mathbf{Z}$ in each iteration and solve this subproblem based on truncated SVD with the predicted rank, because truncated SVD is usually faster than skinny SVD when the user-defined rank is small. They also proved that the theoretical convergence of aLADMAP can be guaranteed, under the assumption that both the subproblems in aLADMAP are exactly solved at each iteration. However, aLADMAP may not converge to the global optimum in practice, possibly because the subproblem related to the nuclear norm based term may be solved inexactly when the prediction of the rank of $\mathbf{Z}$ is inappropriate. In addition, although the time complexity per iteration for this solver is lower than that in [14], the complexity is still quadratic w.r.t. n.

In [13], Liu *et al*. improved the LRR solver in [14]. Specifically, based on the observation that the optimal solution (w.r.t. the variable $\mathbf{Z}$) of LRR is in $span(\mathbf{X}')$ [13], they first pre-calculate a matrix by orthogonalizing the columns of $\mathbf{X}'$, and then reformulate LRR as a problem w.r.t. $\{\hat{\mathbf{Z}}, \mathbf{E}\}$, where $\hat{\mathbf{Z}} \in \mathbb{R}^{r\times n}$ and $\mathbf{E} \in \mathbb{R}^{d\times n}$. By using 3BADM to solve the new problem, the time complexity per iteration is at most $\mathcal{O}(d^2n + d^3)$ (assuming $d \leq n$) [13]. However, this LRR solver still suffers from the convergence issue and additional space cost, similarly as that in [14].

In a recent work [17], Talwalkar *et al*. proposed a divide-and-conquer algorithm named Divide-Factor-Combine LRR (DFC-LRR) to accelerate the optimization of LRR. Specifically, they randomly partition the columns of $\mathbf{X}$ into several submatrices, and solve the resultant subproblem w.r.t. each submatrix by using the solver in [13]. However, the performance (*e.g.*, efficiency) of DFC-LRR essentially depends on the solver of its subproblem. Besides, there are several works (*e.g.*, [28, 27, 26]) that study some variants of LRR. Note those works are out of the scope of this paper, because our main goal is to improve the efficiency when exactly solving the LRR problem in (1).

3. Reformulating LRR via Factorized Data

In this section, we reformulate LRR via factorized data, based on which we propose an efficient algorithm for solving LRR, and also conduct theoretical studies with regard to the parameter λ.

Specifically, we study a more general formulation of LRR as follows,

$$\min_{\mathbf{Z}\in\mathbb{R}^{n\times m},\mathbf{E}\in\mathbb{R}^{d\times m}} \quad \|\mathbf{Z}\|_* + \lambda\|\mathbf{E}\|_{2,1} \tag{3}$$

$$\text{s.t.} \quad \mathbf{X}\mathbf{D} = \mathbf{X}\mathbf{Z} + \mathbf{E}$$

where $\mathbf{X} \in \mathbb{R}^{d\times n}$ is the data matrix, and $\mathbf{D} \in \mathbb{R}^{n\times m}$ is a pre-defined matrix. We assume that $\mathbf{X}\mathbf{D}$ is not a zero matrix, otherwise it will result in the trivial (zero) solution. Note that, when setting $\mathbf{D} = \mathbf{I}_n$ and $m = n$, the above problem is reduced to LRR in (1). We will discuss the use of $\mathbf{D}$ when $\mathbf{D} \neq \mathbf{I}_n$ in Section 6.

To derive our reformulation of the problem in (3), we first make the following definitions. Let r denote the rank of $\mathbf{X}$. Moreover, let us factorize $\mathbf{X}$ via the *skinny* singular value decomposition (SVD):

$$\mathbf{X} = \mathbf{U}_r \mathbf{S}_r \mathbf{V}_r' ,$$

where $\mathbf{U}_r \in \mathbb{R}^{d\times r}$ and $\mathbf{V}_r \in \mathbb{R}^{n\times r}$ are two column-wise orthogonal matrices that satisfy $\mathbf{U}_r'\mathbf{U}_r = \mathbf{V}_r'\mathbf{V}_r = \mathbf{I}_r$, $\mathbf{S}_r \in \mathbb{R}^{r\times r}$ is a diagonal matrix defined as

$$\mathbf{S}_r = diag([\sigma_1, \ldots, \sigma_r]') ,$$

in which $\{\sigma_i\}_{i=1}^r$ are the r positive singular values of $\mathbf{X}$ sorted in descending order. Based on the definitions above, we present the reformulation by the following theorem:

Theorem 1 *Let $\mathbf{W}^*$ denote the optimal solution of the following problem,*

$$\min_{\mathbf{W}\in\mathbb{R}^{r\times m}} \|\mathbf{W}\|_* + \lambda\|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W})\|_{2,1} . \tag{4}$$

Then, $\{\mathbf{Z}^*, \mathbf{E}^*\}$, defined as

$$\begin{cases} \mathbf{Z}^* = \mathbf{V}_r\mathbf{W}^* \\ \mathbf{E}^* = \mathbf{X}\mathbf{D} - \mathbf{X}\mathbf{V}_r\mathbf{W}^* \end{cases} ,$$

is the optimal solution of the problem in (3). In particular, $\|\mathbf{Z}^*\|_* = \|\mathbf{W}^*\|_*$ and $\|\mathbf{E}^*\|_{2,1} = \|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W}^*)\|_{2,1}$ always hold, implying that the two problems in (3) and (4) have equal optimal objective values.

Proof 1 *Given page limitation, we sketch the outline of the proof of Theorem 1 here. First, the feasibility of $\{\mathbf{Z}^*, \mathbf{E}^*\}$ can be easily verified by*

$$\mathbf{X}\mathbf{Z}^* + \mathbf{E}^* = \mathbf{X}\mathbf{V}_r\mathbf{W}^* + (\mathbf{X}\mathbf{D} - \mathbf{X}\mathbf{V}_r\mathbf{W}^*) = \mathbf{X}\mathbf{D}.$$

To prove that $\{\mathbf{Z}^*, \mathbf{E}^*\}$ is optimal to the problem in (3), we need to show that for any feasible solution $\{\mathbf{Z}, \mathbf{E}\}$, the following inequality holds:

$$\|\mathbf{Z}\|_* + \lambda\|\mathbf{E}\|_{2,1} \geq \|\mathbf{Z}^*\|_* + \lambda\|\mathbf{E}^*\|_{2,1} .$$

To this end, we prove the following lines.

$$\begin{aligned} & \|\mathbf{Z}\|_* + \lambda\|\mathbf{E}\|_{2,1} \\ \geq\ & \|\mathbf{V}_r'\mathbf{Z}\|_* + \lambda\|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{V}_r'\mathbf{Z})\|_{2,1} \\ \geq\ & \|\mathbf{W}^*\|_* + \lambda\|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W}^*)\|_{2,1} \\ =\ & \|\mathbf{Z}^*\|_* + \lambda\|\mathbf{E}^*\|_{2,1} . \end{aligned}$$

In fact, the first inequality holds because we have $\|\mathbf{E}\|_{2,1} = \|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{V}_r'\mathbf{Z})\|_{2,1}$ and $\|\mathbf{Z}\|_* \geq \|\mathbf{V}_r'\mathbf{Z}\|_*$ based on the feasibility of $\{\mathbf{Z}, \mathbf{E}\}$ and some properties of the $\ell_{2,1}$ norm and the nuclear norm, the second inequality holds because $\mathbf{W}^*$ is the optimal solution of the problem in (4), and the last equality can be verified by proving $\|\mathbf{Z}^*\|_* = \|\mathbf{W}^*\|_*$ and $\|\mathbf{E}^*\|_{2,1} = \|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W}^*)\|_{2,1}$.

Note that, the size of $\mathbf{W} \in \mathbb{R}^{r\times m}$ is no larger than $\mathbf{Z} \in \mathbb{R}^{n\times m}$ since we have $r \leq n$. Moreover, considering that $\mathbf{S}_r$ is a diagonal matrix and $\mathbf{V}_r'\mathbf{D}$ can be treated as a pre-computed matrix, the new formulation in (4) does not contain matrix-matrix multiplication between *full* matrices.

Based on Theorem 1, we can analyze the problem in (3) by studying the problem in (4) instead. In particular, we propose a fast LRR solver called *FaLRR* to obtain the solution of the LRR problem in (1) based on Theorem 1, for which the core part is to solve the problem in the form of (4). In Section 4, we show that the problem in (4) can be efficiently solved by ADM with two blocks of variables, where the resultant subproblems can be exactly solved. Thus, the global optimum can be achieved. Moreover, based on the new problem in (4), we present our theoretical results regarding the influence of the parameter λ in Section 5.

4. Optimization

To solve the problem in (4), we first equivalently rewrite it as a two-variable optimization problem for ease of optimization. Specifically, by introducing another variable $\mathbf{Q} \in \mathbb{R}^{r\times m}$, we rewrite the problem in (4) as follows:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{Q}\in\mathbb{R}^{r\times m}} \quad & \|\mathbf{W}\|_* + \lambda\|\mathbf{S}_r\mathbf{Q}\|_{2,1} \\ s.t. \quad & \mathbf{W} + \mathbf{Q} = \mathbf{V}_r'\mathbf{D}. \end{aligned} \tag{5}$$

The corresponding augmented Lagrangian [2] is

$$\begin{aligned} & \mathcal{L}_\rho(\mathbf{W}, \mathbf{Q}, \mathbf{L}) \\ = \ & \|\mathbf{W}\|_* + \lambda\|\mathbf{S}_r\mathbf{Q}\|_{2,1} + \langle \mathbf{L}, \mathbf{V}_r'\mathbf{D} - \mathbf{W} - \mathbf{Q}\rangle \\ & + \frac{\rho}{2}\|\mathbf{V}_r'\mathbf{D} - \mathbf{W} - \mathbf{Q}\|_F^2, \end{aligned}$$

where $\mathbf{L} \in \mathbb{R}^{r\times m}$ is the Lagrangian multiplier and $\rho > 0$ is the penalty parameter.

By employing ADM, we iteratively update the variables $\{\mathbf{W}, \mathbf{Q}\}$, the Lagrange multiplier $\mathbf{L}$ and the penalty parameter ρ until convergence, as shown in Algorithm 1. In particular, we introduce how to efficiently and exactly solve the subproblems for updating the variables $\{\mathbf{W}, \mathbf{Q}\}$ as follows.

4.1. Updating W

The subproblem $\min_{\mathbf{W}} \mathcal{L}_\rho(\mathbf{W}, \mathbf{Q}_t, \mathbf{L}_t)$ for updating $\mathbf{W}$ is given by

$$\min_{\mathbf{W}\in\mathbb{R}^{r\times m}} \|\mathbf{W}\|_* + \frac{\rho}{2}\|\mathbf{W} - \mathbf{G}\|_F^2 , \tag{6}$$

where $\mathbf{G} \in \mathbb{R}^{r\times m}$ is defined as $\mathbf{G} = \mathbf{V}_r'\mathbf{D} - \mathbf{Q}_t + \mathbf{L}_t/\rho$.

Based on Theorem 2.1 in [4], the optimal solution of the problem (6) can be obatined by applying the singular value shrinkage operator to $\mathbf{G}$, *i.e.*,

$$\mathcal{D}_{1/\rho}(\mathbf{G}) = \mathbf{U}_G diag\left([\mathbf{s}_G - \frac{1}{\rho}\mathbf{1}]_+\right)\mathbf{V}_G', \tag{7}$$

Algorithm 1 Solving the problem in (4).

Input: $\mathbf{S}_r$, $\mathbf{V}_r$, $\mathbf{D}$ and λ.
Initialize $\mathbf{W}_0 = \mathbf{Q}_0 = \mathbf{L}_0 = \mathbf{O}_{r\times m}$, $t = 0$, and set the parameters ρ, ρ_{max}, γ and ε.
while not converged **do**
1. $\mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \mathcal{L}_\rho(\mathbf{W}, \mathbf{Q}_t, \mathbf{L}_t)$.
2. $\mathbf{Q}_{t+1} = \arg\min_{\mathbf{Q}} \mathcal{L}_\rho(\mathbf{W}_{t+1}, \mathbf{Q}, \mathbf{L}_t)$.
3. $\mathbf{L}_{t+1} = \mathbf{L}_t + \rho(\mathbf{V}_r'\mathbf{D} - \mathbf{W}_{t+1} - \mathbf{Q}_{t+1})$.
4. $\rho = \min(\gamma\rho, \rho_{max})$.
5. $t = t + 1$.
6. Check whether the following convergence condition is satisfied: $\|\mathbf{V}_r'\mathbf{D} - \mathbf{W}_t - \mathbf{Q}_t\|_{max} \leq \epsilon$.

end while
Output: $\mathbf{W}^* = \mathbf{W}_t$

where $\mathbf{U}_G diag(\mathbf{s}_G)\mathbf{V}_G'$ is the skinny SVD of $\mathbf{G}$, and $[\cdot]_+$ is a thresholding operator by setting the negative elements to zeros.

Interestingly, $\mathcal{D}_{1/\rho}(\mathbf{G})$ can be calculated in another way. Assume that we know the number of singular values in $\mathbf{s}_G$ that are greater than $1/\rho$, and let p denote it. Then, $\mathcal{D}_{1/\rho}(\mathbf{G})$ can also be calculated as

$$\mathcal{D}_{1/\rho}(\mathbf{G}) = \hat{\mathbf{U}}_G \left(diag(\hat{\mathbf{s}}_G) - \frac{1}{\rho}\mathbf{I}_p \right) \hat{\mathbf{V}}_G', \tag{8}$$

where $\hat{\mathbf{U}}_G \in \mathbb{R}^{r\times p}$, $\hat{\mathbf{V}}_G \in \mathbb{R}^{m\times p}$ and $\hat{\mathbf{s}}_G \in \mathbb{R}^p$ are obtained from the *truncated* SVD (with rank p) of $\mathbf{G}$, *i.e.*, $\hat{\mathbf{U}}_G diag(\hat{\mathbf{s}}_G)\hat{\mathbf{V}}_G'$.

Since truncated SVD only calculates the largest p singular values rather than all positive singular values, it is usually faster than skinny SVD when p is not very large. Therefore, it is used in [12] to improve the efficiency when solving the nuclear norm related problem which is similar to (6). However, when p is large, truncated SVD might be even slower than skinny SVD in practice. Therefore, such a strategy cannot always guarantee the improvement of efficiency. Moreover, given $\mathbf{G}$ and ρ, it is non-trivial to accurately estimate p, and inappropriate estimation may result in a suboptimal solution of (6).

Therefore, we propose a *tentative* strategy. In particular, we first progressively calculate at most $\min(r, p_{max})$ largest singular values of $\mathbf{G}$ (where p_{max} is empirically set to 5 in this work), and compare the smallest one of them with $\frac{1}{\rho}$. If it is not larger than $\frac{1}{\rho}$, which indicates $p \leq p_{max}$, then we obtain $\hat{\mathbf{s}}_G$ based on these singular values and recover $\{\hat{\mathbf{U}}_G, \hat{\mathbf{V}}_G\}$ to calculate $\mathcal{D}_{1/\rho}(\mathbf{G})$ as in (8). Otherwise, we directly perform skinny SVD on $\mathbf{G}$ to calculate $\mathcal{D}_{1/\rho}(\mathbf{G})$ as in (7), and we will not do such attempt in the subsequent iterations in Algorithm 1, because p (depends on ρ and $\mathbf{G}$) usually increases as the number of iterations increases according to our experimental observations. Note that, our method guarantees the exact solution to the problem in (6) without estimating p.

4.2. Updating Q

The subproblem $\min_{\mathbf{Q}} \mathcal{L}_\rho(\mathbf{W}_{t+1}, \mathbf{Q}, \mathbf{L}_t)$ for updating $\mathbf{Q}$ is given by,

$$\min_{\mathbf{Q}\in\mathbb{R}^{r\times m}} \lambda\|\mathbf{S}_r\mathbf{Q}\|_{2,1} + \frac{\rho}{2}\|\mathbf{Q} - \mathbf{C}\|_F^2, \tag{9}$$

where $\mathbf{C} \in \mathbb{R}^{r\times m}$ is defined as $\mathbf{C} = \mathbf{V}_r'\mathbf{D} - \mathbf{W}_{t+1} + \mathbf{L}_t/\rho$.

Let us denote the i-th column of $\mathbf{Q}$ and $\mathbf{C}$ as $\mathbf{q}_i$ and $\mathbf{c}_i$, respectively. Then the above problem can be rewritten as

$$\min_{\{\mathbf{q}_i\}_{i=1}^m} \lambda\sum_{i=1}^m \|\mathbf{S}_r\mathbf{q}_i\| + \frac{\rho}{2}\sum_{i=1}^m \|\mathbf{q}_i - \mathbf{c}_i\|^2, \tag{10}$$

which is separable w.r.t. $\{\mathbf{q}_i\}_{i=1}^m$. Therefore, we solve it by optimizing m subproblems, with each in the following form:

$$\min_{\mathbf{q}\in\mathbb{R}^r} \|\mathbf{S}_r\mathbf{q}\| + \frac{\mu}{2}\|\mathbf{q} - \mathbf{c}\|^2, \tag{11}$$

where the subscripts of $\mathbf{q}_i$ and $\mathbf{c}_i$ are dropped for convenient presentation, μ is defined as $\mu = \frac{\rho}{\lambda}$. The corresponding optimal solution is

$$\mathbf{q}^* = \begin{cases} [\frac{\mu\alpha c_1}{\mu\alpha+\sigma_1^2}, \ldots, \frac{\mu\alpha c_r}{\mu\alpha+\sigma_r^2}]', & \text{if } \|\mathbf{S}_r^{-1}\mathbf{c}\| > \frac{1}{\mu} \\ \mathbf{0}_r, & \text{otherwise} \end{cases}, \tag{12}$$

where c_i denotes the i-th element of $\mathbf{c}$ $\forall i = 1, \ldots, r$, and in the case of $\|\mathbf{S}_r^{-1}\mathbf{c}\| > \frac{1}{\mu}$, α can be obtained as the unique positive root of $\sum_{i=1}^r (\frac{c_i\sigma_i}{\mu\alpha+\sigma_i^2})^2 = \frac{1}{\mu^2}$ by using the bisection method [3].

4.3. Time Complexity

The time complexity for iterative algorithms (such as 3BADM [13], aLADMAP [12] and the proposed Algorithm 1) mainly depends on two aspects: the total number of iterations and the computational cost per iteration. We observe that the total number of iterations of the proposed Algorithm 1 is often relatively small (see Figure 3). So we focus on the discussion of the computational cost per iteration as follows.

For Algorithm 1, the most time-consuming steps at each iteration are updating $\mathbf{W}$ and updating $\mathbf{Q}$. In particular, for updating $\mathbf{W}$, the time complexity is usually no more than $\mathcal{O}(rm\min(r, m))$, given that p_{max} is small. To update $\mathbf{Q}$, we solve m subproblems, each with $\mathcal{O}(r)$ time complexity, so the time complexity for updating $\mathbf{Q}$ is $\mathcal{O}(rm)$. In summary, for Algorithm 1, the total time complexity of each iteration is $\mathcal{O}(rm\min(r, m) + rm)$. Particularly, for solving the LRR problem in (1) where $m = n$ and $r \leq n$, the

time complexity per iteration for FaLRR is $\mathcal{O}(nr^2 + nr)$. In contrast, the time complexity per iteration is at most $\mathcal{O}(nd^2 + d^3)$ (assuming $d \leq n$) for 3BADM [13], and $\mathcal{O}(n^2\hat{r})$ [12] for aLADMAP, where $\hat{r}$ denotes the rank of $\mathbf{Z}$ in one iteration of aLADMAP.

5. Theoretical Study

In this section, based on the new formulation (4), we present some theoretical results w.r.t. the influence of the parameter λ.

To avoid confusion, let $\mathbf{W}_\lambda^*$ denote the optimal solution to the problem in (4) given the parameter λ, *i.e.*,

$$\mathbf{W}_\lambda^* = \arg\min_{\mathbf{W}\in\mathbb{R}^{r\times m}} \|\mathbf{W}\|_* + \lambda\|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W})\|_{2,1}.$$

In addition, let

$$F(\lambda) = \|\mathbf{W}_\lambda^*\|_* + \lambda\|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W}_\lambda^*)\|_{2,1}$$

denote the optimal objective value of the problem in (4).

Firstly, we have the following lemma that shows the behaviors of $F(\lambda)$, $\|\mathbf{W}_\lambda^*\|_*$ and $\|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W}_\lambda^*)\|_{2,1}$ when the positive parameter λ increases.

Lemma 1 *$F(\lambda)$ and $\|\mathbf{W}_\lambda^*\|_*$ are both non-decreasing w.r.t. $\lambda > 0$, while $\|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W}_\lambda^*)\|_{2,1}$ is non-increasing w.r.t. $\lambda > 0$.*

Moreover, it is clear that $0 \leq rank(\mathbf{W}_\lambda^*) \leq \min(r, m)$, given that $\mathbf{W}_\lambda^* \in \mathbb{R}^{r\times m}$. Based on Lemma 1, we show a lower-bound of $rank(\mathbf{W}_\lambda^*)$ in the following theorem.

Theorem 2 *For any positive λ, the rank of $\mathbf{W}_\lambda^*$ satisfies*

$$rank(\mathbf{W}_\lambda^*) \geq \frac{\|\mathbf{W}_\lambda^*\|_*}{\|\mathbf{V}_r'\mathbf{D}\|_2 + \frac{1}{\sigma_r}\|\mathbf{S}_r(\mathbf{V}_r'\mathbf{D} - \mathbf{W}_\lambda^*)\|_{2,1}}, \quad (13)$$

where the right-hand side is non-decreasing *w.r.t. λ.*

In particular, the *non-decreasing* property of the right-hand side of (13) can be easily obtained according to Lemma 1. We omit the detailed proofs of Lemma 1 and Theorem 2, due to the page limitation.

Based on Theorem 2, if λ is large, $rank(\mathbf{W}_\lambda^*)$ tends to be large since its lower-bound is non-decreasing w.r.t. λ. In this case, $\mathbf{W}$ in the last few iterations of Algorithm 1 tends to have a large rank, so updating $\mathbf{W}$ via (8) based on truncated SVD may be inefficient. This motivated us to design the tentative strategy in Section 4.1. In addition, regarding the influence of λ for the LRR problem in (1), we can draw similar conclusions to Lemma 1 and Theorem 2 according to Theorem 1.

Algorithm 2 Incorporating Algorithm 1 into DFC-LRR.

Input: $\mathbf{S}_r$, $\mathbf{V}_r$, λ, q.
1. Randomly generate $\hat{\mathbf{I}}$ and split $\hat{\mathbf{I}}$ into $\{\mathbf{D}_i\}_{i=1}^q$.
2. $\forall i = 1, \ldots, q$, obtain $\mathbf{W}_i^*$ via Algorithm 1 with input $\{\mathbf{S}_r, \mathbf{V}_r, \mathbf{D}_i, \lambda_i\}$, then calculate $\mathbf{Z}_i^* = \mathbf{V}_r\mathbf{W}_i^*$.
3. $\mathbf{Z}_{DFC}^* = ColumnProjection([\mathbf{Z}_1^*, \ldots, \mathbf{Z}_q^*], \mathbf{Z}_1^*)$ [17].
Output: $\mathbf{Z}_{DFC}^*$.

6. Incorporation into DFC-LRR

The distributed framework DFC-LRR [17] involves three steps. Firstly, the columns of $\mathbf{X}$ are randomly partitioned into q submatrices $\{\mathbf{X}_i \in \mathbb{R}^{d\times n_i}\}_{i=1}^q$, where n_i's are approximately equal, and $\sum_{i=1}^q n_i = n$. After that, we obtain the minimizers $\{\mathbf{Z}_i^*\}_{i=1}^q$ by solving q subproblems, with each in the following form:

$$\min_{\mathbf{Z}_i\in\mathbb{R}^{n\times n_i}, \mathbf{E}_i\in\mathbb{R}^{d\times n_i}} \|\mathbf{Z}_i\|_* + \lambda_i\|\mathbf{E}_i\|_{2,1} \quad (14)$$

$$s.t. \quad \mathbf{X}_i = \mathbf{X}\mathbf{Z}_i + \mathbf{E}_i \quad (15)$$

where $\lambda_i = \lambda\sqrt{n/n_i}$ is a rescaled parameter. Finally, we calculate $\mathbf{Z}_{DFC}^* \in \mathbb{R}^{n\times n}$ based on $\{\mathbf{Z}_i^*\}_{i=1}^q$ via column projection (see [17] for the details), so that $\mathbf{Z}_{DFC}^*$ is an approximated solution of the LRR problem in (1).

Interestingly, the problem in (14) can be rewritten in the form of (3), and thus can be solved using the proposed Algorithm 1. To be exact, we can replace $\mathbf{X}_i \in \mathbb{R}^{d\times n_i}$ in (15) with $\mathbf{X}\mathbf{D}_i$, where each $\mathbf{D}_i \in \{0, 1\}^{n\times n_i}$ is a column submatrix of $\hat{\mathbf{I}}$ (in other words, $[\mathbf{D}_1, \ldots, \mathbf{D}_q] = \hat{\mathbf{I}}$), in which $\hat{\mathbf{I}}$ is obtained by randomly reordering the columns of an identity matrix $\mathbf{I}_n$. Algorithm 2 details how Algorithm 1 is incorporated into DFC-LRR, where the second step can be implemented in parallel.

7. Experiments

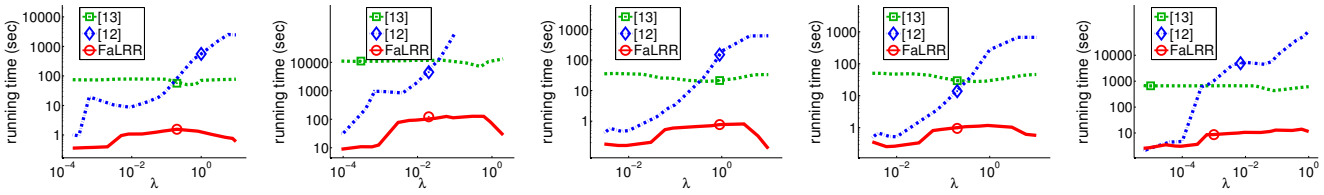
In the experiments, we compare our FaLRR with the existing LRR solvers in [12, 13] for exactly solving the LRR problem. Moreover, we compare the efficiency of Algorithm 2 with that of the original DFC-LRR proposed in [17].

All the experiments are conducted on a desktop with Intel Xeon CPU (3.2GHz), 16GB memory and MATLAB R2012b, and we record the running time for each algorithm. For [13], [12] and [17], we use the codes which are available online [13, 12] or provided by the authors [17]. We preprocess the data by orthogonalizing the columns of $\mathbf{X}'$ for the LRR solver in [13] and DFC-LRR [17], or by performing[1] skinny SVD on $\mathbf{X}$ for FaLRR and Algorithm 2. The time of preprocessing is not counted into the running time of these

[1]On the real-world datasets, namely the ExtYaleB, NH, LFW100+, LFW50+ and HARUS datasets, it takes 0.13, 19.64, 0.07, 0.09 and 0.99 seconds, respectively.

Table 1: The running time (in seconds) and subspace clustering accuracies (%) for solving the LRR problem on the ExtYaleB, NH, LFW 100+, LFW 50+ datasets and the HARUS dataset. "SPEEDUP" denotes the ratio between the running time of a baseline method and that of our FaLRR.

DATASET	d	n	r	RUNNING TIME (SEC) / CLUSTERING ACCURACY (%)			SPEEDUP	
				FALRR	[12]	[13]	VS. [12]	VS. [13]
EXTYALEB	2016	640	640	1.56/79.22	557.20/79.53	56.50/79.22	**357**	**36**
NH	6608	4660	4660	126.68/99.94	4506.52/99.81	11033.27/99.98	**36**	**87**
LFW 100 +	1000	500	500	0.77/99.40	149.61/99.40	21.36/99.40	**194**	**28**
LFW 50 +	1000	600	600	0.95/98.00	13.89/98.00	29.60/98.00	**15**	**31**
HARUS	561	10299	471	8.64/82.93	4924.40/71.57	662.78/51.37	**570**	**77**



(a) ExtYaleB (b) NH (c) LFW 100+ (d) LFW 50+ (e) HARUS

Figure 1: The running time (in seconds) w.r.t. λ, for solving the LRR problem on the real-world datasets. The positions of markers indicate the optimal parameters for the three LRR solvers, respectively.

algorithms, because it only needs to be conducted once on each dataset and the computation is fast. Following [17], for DFC-LRR and our Algorithm 2, we report the parallel running time, namely the longest one of running times for solving the subproblems in DFC-LRR plus the running time for combining the minimizers of subproblems via column projection.

To obtain the clustering results, we follow [13] to build the affinity matrix based on the resultant solution of LRR, and then perform spectral clustering on it. To evaluate the clustering performance, we calculate the clustering accuracy (also called segmentation accuracy) [14, 13, 17]. Following [14, 13], we tune the parameter λ for each LRR solver and report the best clustering performance.

7.1. Experiments on real-world datasets

Datasets: In this experiment, we compare different methods for solving the LRR problem for clustering face images or human activities. To this end, we use the following datasets:

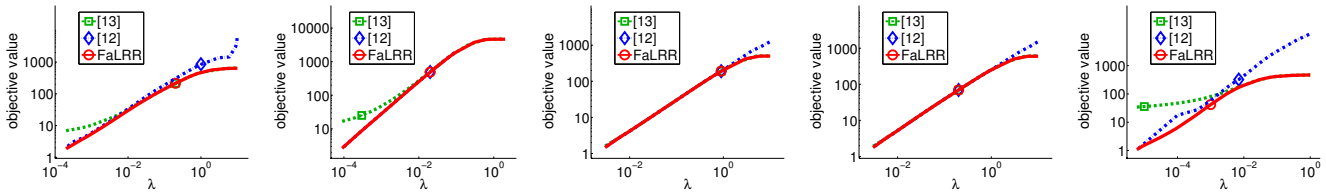
The Extended Yale Face Database B (**ExtYaleB**) contains $2,414$ frontal face images of 38 subjects, with different lighting, poses and illumination conditions. There are about 64 faces for each subject. In the experiment, we follow [14, 17] to use 640 faces corresponding to the first 10 subjects. We resize each face image to 48×42 pixels and extract the 2016-dimensional gray-level intensity feature for representing each face image.

The Notting-Hill (**NH**) dataset [22, 23, 5] consists of $4,660$ face images of 5 main casts detected in a movie called "Notting Hill". The face images in this dataset are taken in the unconstrained environments, with variations in poses, facial expressions, illumination and occlusions. We resize each face image to 48×42 pixels, and uniformly divide it into 16×7 non-overlapping blocks. Then, we extract 59-dimensional Local Binary Patterns (LBP) [16, 11] descriptor from each block and concatenate them to form a $6,608$-dimensional feature vector.

The Labeled Faces in the Wild (**LFW**) dataset [10, 6] is a benchmark face database. It contains more than $10,000$ in-the-wild faces from $5,749$ subjects, with large variations in poses, facial expressions, illumination and occlusions. Note that in this database, many subjects have only a few faces. To guarantee that we have sufficient faces per subject, we use two subsets of this database. Each subset consists of the subjects with at least n_{face} faces, where n_{face} is 100 or 50. Accordingly, the two subsets are referred to as "LFW 100+" and "LFW 50+", respectively. The first n_{face} faces for each subject are used in each subset. To represent each face, we use the 1000-dimensional deep learning based feature [8] provided by Face++[2], because of the excellent face recognition performance on the LFW benchmark.

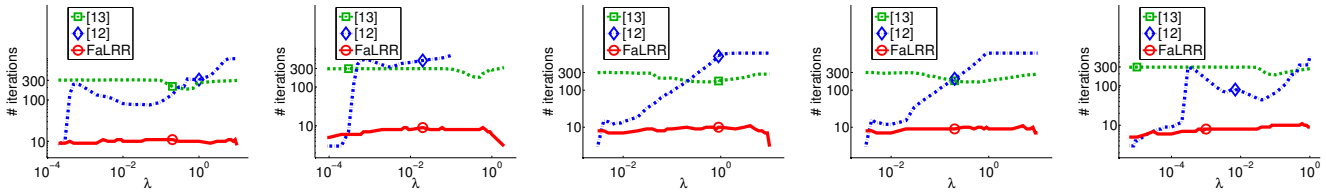
The Human Activity Recognition Using Smartphones (**HARUS**) dataset [1] is a large dataset with data collected

[2] http://www.faceplusplus.com/



(a) ExtYaleB (b) NH (c) LFW 100+ (d) LFW 50+ (e) HARUS

Figure 2: The resultant objective value w.r.t. λ, for solving the LRR problem on the real-world datasets. The positions of markers indicate the optimal parameters for the three LRR solvers, respectively. Note the curves may overlap when different LRR solvers achieve the same objective values.



(a) ExtYaleB (b) NH (c) LFW 100+ (d) LFW 50+ (e) HARUS

Figure 3: The total number of iterations w.r.t. λ, for solving the LRR problem on the real-world datasets. The positions of markers indicate the optimal parameters for the three LRR solvers, respectively. "# iterations" denotes the total number of iterations.

using embedded sensors (*i.e.* accelerometer and gyroscope) on the smartphones. To collect such data, the smartphones with embedded sensors are carried by volunteers on their waists, when they are conducting daily activities (*e.g.*, walking, sitting, laying). After that, the captured sensor signals (3-axial linear acceleration and 3-axial angular velocity) are pre-processed to filter noise and post-processed (*e.g.*, by sampling). Finally, for each signal, a 561-dimensional feature vector with time and frequency domain variables is extracted. In our experiment, we use the whole dataset with 10,299 signals w.r.t. 6 activities.

Experimental results: For the face datasets and the human activity dataset, the running times and clustering accuracies of LRR solvers in [12] and [13] as well as our FaLRR are reported in Table 1, where the running time and clustering accuracy for each method are obtained by using the optimal parameter from the best clustering performance. Moreover, Figure 1, Figure 2 and Figure 3 show the running time, the resultant objective value and the total number of iterations w.r.t. λ, for solving the LRR problem on those datasets[3], respectively. Based on these results, we have the following observations:

According to Table 1, our FaLRR consistently achieves order-of-magnitude speedup over the LRR solvers in [13] and [12] on all datasets, when all of the methods use their optimal parameter values.

[3]For λ with relatively large values, some results of the LRR solver in [12] are not available since it takes too long time.

Based on Figure 1, our FaLRR tends to be more efficient when the parameter is relatively large or small, because either the number of iterations is generally smaller (see Figure 3) or $\mathbf{W}$ in Algorithm 1 can be updated efficiently based on truncated SVD according to our tentative strategy. Moreover, the running time of the LRR solver in [13] is usually large for all values of λ, since its total number of iterations is often large (see Figure 3), and it frequently uses matrix-matrix multiplication (such as $\mathbf{XZ}$) and skinny SVD operations in its optimization. In addition, the running time of the LRR solver in [12] heavily depends on λ, possibly because its number of iterations is relatively sensitive to λ (see Figure 3). Moreover, it always adopts truncated SVD when solving its nuclear norm related subproblem, which may be time-consuming when λ is large.

Comparing the clustering performance of the three LRR solvers, we observe that they achieve comparable results on the first four datasets listed in Table 1, but there is obvious difference between their results on the HARUS dataset. Such difference may be related to the corresponding objective values in Figure 2. On the LFW 50+ and LFW 100+ datasets, these three LRR solvers usually achieve the same objective value and they also achieve the same best clustering accuracy (see Table 1) when using the same value of λ (see Figure 2). However, the LRR solvers in [13, 12] cannot result in a low objective value as our FaLRR on the HARUS dataset (see Figure 2), which provides a possible explanation of their lower clustering accuracies than ours.

Regarding the resultant objective value, our FaLRR usually achieves the minimum value. The other LRR solvers may not always arrive at a global optimum, possibly because of inappropriate rank prediction or the lack of theoretical guarantee.

7.2. Experiments on synthetic datasets

In the previous experiments, we have shown the efficiency of our FaLRR for solving the LRR problem on several real-world datasets. In this experiment, we perform a more comprehensive evaluation of its efficiency by using synthetic datasets with different sizes and ranks of data. The synthetic datasets are constructed similarly as those in [14, 12]. Considering that the time complexity of Algorithm 1 depends on $\{n, r\}$ and is irrelevant of d, we generate synthetic data $\mathbf{X} \in \mathbb{R}^{d\times n}$ with $rank(\mathbf{X}) = r$, where we fix d to 4000 and set $\{n, r\}$ to various values.

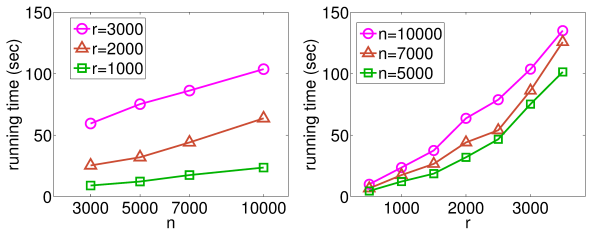
To generate the synthetic dataset, we firstly generate the bases $\{\mathbf{B}_i\}_{i=1}^{s}$ for s independent subspaces, where s is set to 10. Specifically, we generate $\mathbf{B}_1 \in \mathbb{R}^{d\times \frac{r}{s}}$ as a random orthogonal matrix. Then, we generate $\mathbf{T} \in \mathbb{R}^{d\times d}$ as a random rotation matrix, and produce the rest $(s-1)$ bases by $\mathbf{B}_{i+1} = \mathbf{T}\mathbf{B}_i, i = 1, \ldots, s-1$. After that, we obtain $\mathbf{B}_i\mathbf{Q}_i$ as data points sampled from each subspace $\forall i = 1, \ldots, s$, where each $\mathbf{Q}_i \in \mathbb{R}^{\frac{r}{s}\times\frac{n}{s}}$ is a matrix of independent entries with each distributed uniformly in $[0, 1]$. Since the experiments on the synthetic datasets are used for efficiency evaluation of FaLRR, we simply use clean data without adding outliers. Finally, we obtain the matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ as $\mathbf{X} = [\mathbf{B}_1\mathbf{Q}_1, \ldots, \mathbf{B}_s\mathbf{Q}_s]$.

For solving the LRR problem on each $\mathbf{X}$ whose scale is characterized by $\{n, r\}$, we set the parameter λ in the form of $\lambda = a \times 10^{\kappa}$, where $a \in \{1, \ldots, 9\}$, $\kappa \in \mathbb{Z}$ lies in a sufficiently large range. After running FaLRR with λ set to these values, we report the running time corresponding to the parameter value that leads to the *longest* time.

Figure 4a and Figure 4b show the running times of our FaLRR w.r.t. n and r, respectively, for solving the LRR problem on the synthetic datasets. It is observed that, even for data with relatively large size and rank, *e.g.*, $n = 10000$ and $r = 3000$, FaLRR is still efficient, with its running time around 2 minutes. According to Figure 4a and Figure 4b, the running time of FaLRR is approximately linear w.r.t. n and quadratic w.r.t. r.

7.3. Experiment with DFC-LRR

In this experiment, we compare the efficiency of Algorithm 2, the original DFC-LRR in [17], FaLRR and the LRR solver in [13]. We conduct this experiment on the NH dataset, since solving the LRR problem on this real-world dataset is relatively time-consuming for the LRR solvers (see Table 1 and Figure 1). For Algorithm 2 and the original DFC-LRR, the number of submatrices is set to 10.



(a) Running time w.r.t. n (b) Running time w.r.t. r

Figure 4: The running time of FaLRR w.r.t. n and r, for solving the LRR problem on the synthetic datasets .

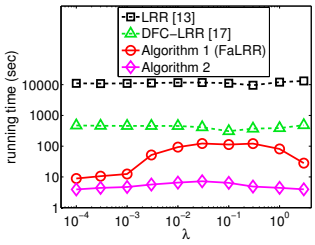


Figure 5: The running time of different algorithms w.r.t. λ, on the NH dataset.

Figure 5 shows the running times of these algorithms on the NH dataset.

It is observed that, Algorithm 2 is consistently faster than FaLRR, with the longest running time reduced from over 100 seconds to less than 10 seconds. Similarly, DFC-LRR proposed in [17] usually achieves more than 10 times speedup over the LRR solver in [13]. Moreover, we also observe that our FaLRR already outperforms DFC-LRR in [17] in terms of the running time, which again demonstrates the efficiency of our FaLRR. Last but not least, by incorporating Algorithm 1 into the distributed framework DFC-LRR, our Algorithm 2 achieves an impressive speedup (about 1000 times speedup) over the LRR solver in [13], with the running time reduced from about 3 hours to less than 10 seconds.

8. Conclusion

In this paper, we have developed a fast LRR solver called FaLRR which is guaranteed to result in a global optimum, based on our new reformulation of the LRR problem. Moreover, our theoretical study shows the influence of the parameter in LRR. In addition, our proposed algorithm can be easily incorporated into the distributed framework called DFC-LRR. The extensive experiments have demonstrated that our FaLRR is usually much faster than existing solvers [12, 13] for solving the LRR problem, and that incorporating our algorithm into the distributed framework DFC-LRR can further improve the efficiency.

Acknowledgement

This research is supported in part by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority, the Singapore MoE Tier 2 Grant (ARC42/13), as well as Australian Research Council Projects (FT-130101457 and DP-140102164).

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Ambient assisted living and home care*, pages 216–223. 2012.

[2] 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*, 3(1):1–122, 2011.

[3] R. Burden and J. Faires. *Numerical analysis*. Cengage Learning, 2011.

[4] J. Cai, C. Emmanuel, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[5] X. Cao, C. Zhang, H. Fu, S. Liu, and H. Zhang. Diversity-induced multi-view subspace clustering. In *CVPR*, 2015.

[6] Z. Cui, W. Li, D. Xu, S. Shan, and X. Chen. Fusing robust face region descriptors via multiple metric learning for face recognition in the wild. In *CVPR*, pages 3554–3561, 2013.

[7] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *T-PAMI*, 35(11):2765–2781, 2013.

[8] H. Fan, Z. Cao, Y. Jiang, Q. Yin, and C. Doudou. Learning deep face representation. *arXiv preprint arXiv:1403.2802*, 2014.

[9] H. Fu, D. Xu, S. Lin, D. W. K. Wong, and J. Liu. Automatic optic disc detection in oct slices via low-rank reconstruction. *T-BME*, 62(4):1151–1158, 2015.

[10] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[11] M. Kan, D. Xu, S. Shan, W. Li, and X. Chen. Learning prototype hyperplanes for face verification in the wild. *T-IP*, 22(8):3310–3316, 2013.

[12] Z. Lin, R. Liu, and Z. Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In *NIPS*, pages 612–620, 2011.

[13] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *T-PAMI*, 35(1):171–184, 2013.

[14] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *ICML*, pages 663–670, 2010.

[15] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2002.

[16] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *T-PAMI*, 24(7):971–987, 2002.

[17] A. Talwalkar, L. Mackey, Y. Mu, S.-F. Chang, and M. I. Jordan. Distributed low-rank subspace segmentation. In *ICCV*, pages 3543–3550, 2013.

[18] M. Tan, Q. Shi, A. van den Hengel, C. Shen, J. Gao, F. Hu, and Z. Zhang. Learning graph structure for multi-label image classification via clique generation. In *CVPR*, 2015.

[19] M. Tan, I. W. Tsang, L. Wang, B. Vandereycken, and S. J. Pan. Riemannian pursuit for big matrix recovery. In *ICML*, pages 1539–1547, 2014.

[20] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *T-PAMI*, 27(12):1945–1959, 2005.

[21] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[22] B. Wu, Y. Zhang, B.-G. Hu, and Q. Ji. Constrained clustering and its application to face clustering in videos. In *CVPR*, pages 3507–3514, 2013.

[23] S. Xiao, M. Tan, and D. Xu. Weighted block-sparse low rank representation for face clustering in videos. In *ECCV*, pages 123–138, 2014.

[24] Z. Xu, W. Li, L. Niu, and D. Xu. Exploiting low-rank structure from latent domains for domain generalization. In *ECCV*, pages 628–643, 2014.

[25] Z. Zeng, S. Xiao, K. Jia, T. Chan, S. Gao, D. Xu, and Y. Ma. Learning by associating ambiguously labeled images. In *CVPR*, pages 708–715, 2013.

[26] H. Zhang, Z. Yi, and X. Peng. fLRR: fast low-rank representation using Frobenius-norm. *Electronics Letters*, 50(13):936–938, 2014.

[27] X. Zhang, F. Sun, G. Liu, and Y. Ma. Fast low-rank subspace segmentation. *T-KDE*, 26(5):1293–1297, 2014.

[28] L. Zhuang, H. Gao, Z. Lin, Y. Ma, X. Zhang, and N. Yu. Non-negative low rank and sparse graph for semi-supervised learning. In *CVPR*, pages 2328–2335, 2012.