Decentralized Robust Subspace Clustering

Bo Liu†, Xiao-Tong Yuan‡, Yang Yu†, Qingshan Liu‡, Dimitris N. Metaxas†
†Department of Computer Science, Rutgers, The State University of New Jersey
‡Jiangsu Province Key Laboratory of Big Data Analysis Technology,
Nanjing University of Information Science and Technology
lb507@cs.rutgers.edu

Abstract

We consider the problem of subspace clustering using the SSC (Sparse Subspace Clustering) approach, which has several desirable theoretical properties and has been shown to be effective in various computer vision applications. We develop a large scale distributed framework for the computation of SSC via an alternating direction method of multiplier (ADMM) algorithm. The proposed framework solves SSC in column blocks and only involves parallel multivariate Lasso regression subproblems and sample-wise operations. This appealing property allows us to allocate multiple cores/machines for the processing of individual column blocks. We evaluate our algorithm on a shared-memory architecture. Experimental results on real-world datasets confirm that the proposed block-wise ADMM framework is substantially more efficient than its matrix counterpart used by SSC, without sacrificing accuracy. Moreover, our approach is directly applicable to decentralized neighborhood selection for Gaussian graphical models structure estimation.

Introduction

In high-dimensional analysis, it is commonly assumed that data lie in certain more interpretable low-dimensional structures instead of being randomly distributed across the ambient space. Subspace Clustering (SC) is such a tool that assumes the intrinsic structure of data can be described by subspace and aims to recover the underlying subspace structures from the observed high-dimensional ambient data. In fact, when data are generated from multiple classes, it is reasonable to assume that each class can be well represented by a subspace. Given $\mathcal{X}$ as a collection of samples in a d-dimension ambient space; these samples may be partitioned as $\mathcal{X} = \bigcup_{k=1}^{K} \mathcal{X}_k$ with each $\mathcal{X}_k$ being a collection of samples distributed around (unknown) subspace $\mathcal{S}_k \subset \mathbb{R}^d$. The task of SC is to approximately group the points in $\mathcal{X}$ into their respective subspaces. The SC model is gaining increasing attention in computer vision and machine learning due to its desirable statistical properties and promising performance on real-world datasets (Candès et al. 2011; Elhamifar and Vidal 2013; Feng et al. 2014).

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

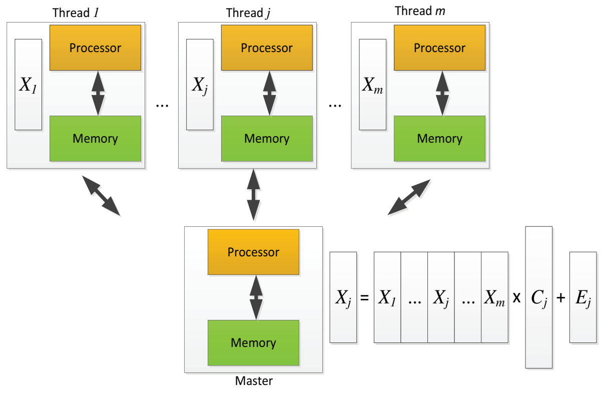

Figure 1: The framework of the proposed distributed sparse subspace clustering algorithm. The overall dataset is decomposed and assigned to multiple cores or machines. $\boldsymbol{C}_i$ and $\boldsymbol{E}_i$ is learned through the communication of master core or machine with others. Finally $\boldsymbol{C}$ and $\boldsymbol{E}$ is derived by aggregating $\{\boldsymbol{C_i}\}_{i=1}^{m}$ and $\{\boldsymbol{E_i}\}_{i=1}^{m}$.

Sparse subspace clustering

Let $\boldsymbol{X} = [\boldsymbol{x}_1, ...\boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ be the data matrix whose columns are the samples in $\mathcal{X}$. In real-world problems, due to the presence of noise, it is usually assumed that $\boldsymbol{X}$ is generated from the following superposition model:

$$\boldsymbol{X} = \boldsymbol{Z} + \boldsymbol{Y}_0 + \boldsymbol{E}_0 \quad (1)$$

where $\boldsymbol{Z} = [\boldsymbol{z}_1, ...\boldsymbol{z}_n]$ is the clean data matrix with columns sampled exactly from $\{\mathcal{S}_k\}_{k=1}^{K}$, $\boldsymbol{Y}_0$ is a dense random matrix modeling noise and $\boldsymbol{E}_0$ is a matrix with certain sparsity structures modeling sparse outliers. The state-of-the-art robust subspace clustering algorithms take advantage of the so-called self-expressiveness property of linear subspaces, i.e., each noiseless data point from one of the subspaces can be reconstructed by a combination of the other noiseless data points from the same subspace. Formally, the self-expressiveness model is defined over the clean data as:

$$\boldsymbol{Z} = \boldsymbol{Z}\boldsymbol{C} \quad s.t.\ \mathrm{diag}(\boldsymbol{C}) = 0$$

where $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ is the representation coefficient matrix and $\mathrm{diag}(\boldsymbol{C})$ is the vector of the diagonal elements of $\boldsymbol{C}$.

Ideally, it is expected that the non-zero representation coefficients for each sample $\boldsymbol{z}_i$ are from those samples belonging to the same subspace as $\boldsymbol{z}_i$, and thus the parameter matrix $\boldsymbol{C}$ has a block diagonal structure with blocks corresponding to clusters. The self-expressive property of the clean data can be propagated to the noisy model in (1) as

$$\boldsymbol{X} = \boldsymbol{X}\boldsymbol{C} + \boldsymbol{Y} + \boldsymbol{E} \quad s.t.\ \text{diag}(\boldsymbol{C}) = 0 \tag{2}$$

where $\boldsymbol{Y} = \boldsymbol{Y}_0 - \boldsymbol{Y}_0\boldsymbol{C}$ and $\boldsymbol{E} = \boldsymbol{E}_0 - \boldsymbol{E}_0\boldsymbol{C}$. Since $\boldsymbol{C}$ is expected to be sparse, the perturbation matrices $\boldsymbol{Y}$ and $\boldsymbol{E}$ are expected to preserve the structures of $\boldsymbol{Y}_0$ and $\boldsymbol{E}_0$. The task of Sparse Subspace Clustering (SSC) (Elhamifar and Vidal 2013) is then to find a solution $(\boldsymbol{C}, \boldsymbol{Y}, \boldsymbol{E})$ for (2) such that $\boldsymbol{C}$ is sparse, $\boldsymbol{Y}$ is bounded and $\boldsymbol{E}$ has certain desired sparsity structure, e.g., column-wise sparsity. To do so, the following optimization problem is considered:

$$\begin{aligned} &\min_{\boldsymbol{C},\boldsymbol{Y},\boldsymbol{E}} \frac{1}{2}\|\boldsymbol{Y}\|_F^2 + \lambda_1\|\boldsymbol{C}\|_1 + \lambda_2\|\boldsymbol{E}\|_{p,1} \\ &s.t.\ \boldsymbol{X} = \boldsymbol{X}\boldsymbol{C} + \boldsymbol{Y} + \boldsymbol{E}, \quad diag(\boldsymbol{C}) = 0 \end{aligned} \tag{3}$$

where $\|\boldsymbol{Y}\|_F$ is the Frobenius norm which promotes having small entries in $\boldsymbol{Y}$, $\|\boldsymbol{C}\|_1$ is the element-wise ℓ_1-norm which promotes sparsity of the entries of $\boldsymbol{C}$, and $\|\boldsymbol{E}\|_{p,1}$ is $\ell_{p,1}$-norm (i.e.,sum of ℓ_p-norm of the columns) which promotes structural sparsity of $\boldsymbol{E}$. For instances, $\|\boldsymbol{E}\|_{1,1}$ encourages element-wise sparsity while $\|\boldsymbol{E}\|_{2,1}$ encourages column-wise sparsity of $\boldsymbol{E}$. The two regularization parameters λ_1 and λ_2 balance the three terms in the objective function. After the optimal representation matrix $\boldsymbol{C}$ is obtained, it can be used to construct an affinity matrix for graph based clustering algorithms such as spectral clustering. Strong statistical guarantees and promising empirical performance have been established for SSC (Elhamifar and Vidal 2013; Candès et al. 2011).

Motivation and contribution

Although being statistically efficient and empirically effective for robust subspace clustering, a bottleneck of SSC is its computational complexity with respect to the sample size n. This can be seen from the fact that SSC in (3) can be decomposed into n independent Lasso-type regression subproblems, each of which is of feature size $n - 1$. This separable structure has motivated the authors of (Elhamifar and Vidal 2009; 2013) to solve the Lasso-type program column-by-column using first-order methods such as alternating direction method of multipliers (ADMM) (Boyd et al. 2011). However, the existing SSC solvers do not scale well to large scale problems as they are not designed to run on multiple cores or distributed computing equipments.

In this paper we present DSSC-ADMM as an efficient variant of ADMM algorithm for *Decentralized* computation of SSC. The framework of the proposed method is illustrated in Fig. 1. The scalability of our method relies on a key innovation that we solve (3) in column-blocks of the coefficient and noise matrices at a time, rather than one column at a time. At a first glance, since (3) already decomposes column-wise, solving multiple columns together in blocks might not seem worthwhile. However, as we will show that DSSC-ADMM working with column-blocks can be compactly decentralized on multiple cores and make more efficient use of each core.

We experiment with shared-memory to illustrate the above points. Empirically, DSSC-ADMM is shown to be much faster than existing methods for SSC, and scales well to large scale problems.

Related work

Subspace clustering. Existing subspace clustering methods (Costeira and Kanade 1998; Bradley and Mangasarian 2000; Ma et al. 2008; Tipping and Bishop 1999) can be roughly divided into four categories: the iterative methods, the algebraic methods, the statistical methods and the spectral based methods. The interested readers can refer to (Vidal 2010) for a comprehensive survey. The SSC method (Elhamifar and Vidal 2013) addressed in this paper belongs to the category of spectral based methods which is built upon the self-expressive model in (2). It is argued in (Liu et al. 2013) that $\boldsymbol{C}$ should have a low-rank structure and impose a nuclear-norm constraint on $\boldsymbol{C}$ in the objective function. In (Luo et al. 2011), sparsity and low rank constraints are combined as a regularizer. Lu *et al.* proposed to estimate $\boldsymbol{C}$ via least square regression (Lu et al. 2012). Feng *et al.* imposed a Laplacian constraint on $\boldsymbol{C}$ to enforce diagonal structure (Feng et al. 2014).

Distributed learning and computing. The era of “Big Data” proposes many new challenges on data scalability. Our framework can be positioned as a part of the recent surge of effort in scaling up machine learning algorithms to big data. Parallel and distributed learning is a promising method to alleviate the pressure of increasing dataset size. Extensive research has been conducted on this topic including platform, communication scheme and algorithm design. Various practical tools and have been developed to facilitate using distributed learning, such as MLI (Sparks et al. 2013) and parameter server (Li et al. 2013). Many popular machine learning algorithms have been reinvestigated to be applicable in distributed computing environment, such as spectral clustering (Chen et al. 2011), K-means clustering (Liang, Balcan, and Kanchanapally 2013), Gaussian graphical models learning (Wang et al. 2013) and matrix factorization (Zhang and Kwok 2014a). Scaling up machine learning algorithms through parallelization and distribution has been heavily explored on various architectures, including shared-memory architectures (Niu et al. 2011), distributed memory architectures (Fu, Wang, and Banerjee 2013) and GPUs (Raina, Madhavan, and Ng 2009).

Due to the high flexibility of ADMM to solve various problems, it has been extensively studied and applied (Boyd et al. 2011; Suzuki 2013; Sedghi, Anandkumar, and Jonckheere 2014; Mota et al. 2013; Wei and Ozdaglar 2012). More recently, Distributed ADMM are being extended to asynchronous conditions. In (Wei and Ozdaglar 2013), a asynchronous ADMM algorithm is developed by partitioning the sub-threads into groups. At each iteration, one group is randomly updated. (Zhang and Kwok 2014b), in which a asynchronous distributed ADMM algorithm that solves consensus constraint. However, owing to the different constraint form it can not be directly applied to our problem.

Decentralized SSC

Since the SSC model in (3) is sparable in columns, for the purpose of parallel computing, it is a natural idea to decompose the model into a group of mutually interacted submodels by dividing the data samples into column blocks.

Column-block decomposition

We assume that the data matrix $\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_m]$ are divided into m blocks in columns (samples) with $\boldsymbol{X}_j \in R^{d\times n_j}$, $\sum_{j=1}^m n_j = n$. With this column-block partition, the structural sparsity outliers can be expressed as $\boldsymbol{E} = [\boldsymbol{E}_1, \boldsymbol{E}_2, ..., \boldsymbol{E}_m]$ and the self-expression coefficient matrix $\boldsymbol{C}$ can be accordingly decomposed as

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} & \dots & \boldsymbol{C}_{1m} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} & \dots & \boldsymbol{C}_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{C}_{m1} & \boldsymbol{C}_{m2} & \vdots & \boldsymbol{C}_{mm} \end{pmatrix}$$

where the block $\boldsymbol{C}_{ij} \in \mathbb{R}^{n_i\times n_j}$ contains the mutual representation coefficients between blocks $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$. Note that $\boldsymbol{E}_{p,1}$ is separable with respect to the column blocks, i.e., $\|\boldsymbol{E}\|_{p,1} = \sum_{i=1}^m \|\boldsymbol{E}_i\|_{p,1}$; the ℓ_1-norm regularization term $\|\boldsymbol{C}\|_1$ is separable with respect to the blocks $\boldsymbol{C}_{ij}$, i.e., $\|\boldsymbol{C}\|_1 = \sum_{i=1}^m\sum_{j=1}^m \|\boldsymbol{C}_{ij}\|_1$. Therefore the SSC model in (3) can be decomposed into m individual Lasso-type subproblems for the column blocks $i = 1, ..., m$:

$$\min_{\{\boldsymbol{C}_{ji}\},\boldsymbol{E}_i} \frac{1}{2}\|\boldsymbol{X}_i - \sum_{j=1}^m \boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{E}_i\|_F^2 + \lambda_1 \sum_{j=1}^m \|\boldsymbol{C}_{ji}\|_1$$
$$+\lambda_2\|\boldsymbol{E}_i\|_{p,1} \quad s.t.\ \text{diag}(\boldsymbol{C}_{ii}) = 0 \tag{4}$$

If only a fraction of the data samples are grossly, then we can use $p = 2$ to enforce column-wise sparsity of $\boldsymbol{E}_i$. If it is a prior that the all the data samples are contaminated by element-wise sparse outliers, then we can use $p = 1$ to encourage entry-wise sparsity of $\boldsymbol{E}_i$.

DSSC-ADMM algorithm

We develop a variant of ADMM (Boyd et al. 2011) to solve each individual subproblem (4). The reason to resort to ADMM is because it can be naturally decentralized for parallel optimization. Let $\boldsymbol{X}_j\boldsymbol{C}_{ji} = \boldsymbol{A}_{ji}$. Then the subproblem (4) can be equivalently formulated as

$$\min_{\{\boldsymbol{A}_{ji},\boldsymbol{C}_{ji}\},\boldsymbol{E}_i} \frac{1}{2}\|\boldsymbol{X}_i - \sum_{j=1}^m \boldsymbol{A}_{ji} - \boldsymbol{E}_i\|_F^2 + \lambda_1 \sum_{j=1}^m \|\boldsymbol{C}_{ji}\|_1$$
$$+\lambda_2\|\boldsymbol{E}_i\|_{p,1} \quad s.t.\ \boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{A}_{ji} = 0, \forall j,\ \text{diag}(\boldsymbol{C}_{ii}) = 0$$

Equivalently, we introduce a squared loss penalty term for each constraint $\boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{A}_{ji} = 0$ regularized by ρ:

$$\min_{\{\boldsymbol{A}_{ji},\boldsymbol{C}_{ji}\},\boldsymbol{E}_i} \frac{1}{2}\|\boldsymbol{X}_i - \sum_{j=1}^m \boldsymbol{A}_{ji} - \boldsymbol{E}_i\|_F^2 + \lambda_1 \sum_{j=1}^m \|\boldsymbol{C}_{ji}\|_1$$
$$+\lambda_2\|\boldsymbol{E}_i\|_{p,1} + \frac{\rho}{2}\sum_{j=1}^m \|\boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{A}_{ji}\|_F^2$$
$$s.t.\ \boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{A}_{ji} = 0, \forall j,\ \text{diag}(\boldsymbol{C}_{ii}) = 0 \tag{5}$$

Then we introduce a matrix $\boldsymbol{\Delta}_{ji}$ as Lagrangian multiplier for each individual equality constraint $\boldsymbol{X}_{ji}\boldsymbol{C}_{ji} - \boldsymbol{A}_{ji} = 0$ and arrive at the following Lagrangian for (5):

$$\min_{\{\boldsymbol{A}_{ji},\boldsymbol{C}_{ji}\},\boldsymbol{E}_i} \frac{1}{2}\|\boldsymbol{X}_i - \sum_{j=1}^m \boldsymbol{A}_{ji} - \boldsymbol{E}_i\|_F^2 + \lambda_1 \sum_{j=1}^m \|\boldsymbol{C}_{ji}\|_1$$
$$+\lambda_2\|\boldsymbol{E}\|_{p,1} + \frac{\rho}{2}\sum_{j=1}^m \|\boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{A}_{ji}\|_F^2$$
$$+\sum_{j=1}^m Tr(\boldsymbol{\Delta}_{ji}^\top(\boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{A}_{ji})) \quad s.t.\ \text{diag}(\boldsymbol{C}_{ii}) = 0$$

The scaled form of the above is

$$\min_{\{\boldsymbol{A}_{ji},\boldsymbol{C}_{ji}\},\boldsymbol{E}_i} \frac{1}{2}\|\boldsymbol{X}_i - \sum_{j=1}^m \boldsymbol{A}_{ji} - \boldsymbol{E}_i\|_F^2 + \lambda_1 \sum_{j=1}^m \|\boldsymbol{C}_{ji}\|_1$$
$$+\lambda_2\|\boldsymbol{E}_i\|_{p,1} + \frac{\rho}{2}\sum_{j=1}^m \|\boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{A}_{ji} + \boldsymbol{U}_{ji}\|_F^2$$
$$s.t.\ \text{diag}(\boldsymbol{C}_{ii}) = 0 \tag{6}$$

where $\boldsymbol{U}_{ji} = \boldsymbol{\Delta}_{ji}/\rho$ is the scaled dual variable. This above optimization problem is jointly convex in $\boldsymbol{C}_{ji}$, $\boldsymbol{A}_{ji}$ and $\boldsymbol{E}_i$, and thus can be solved in a alternating minimization procedure. The algorithmic procedure contains two levels of iterations. The outer loop visits all the column-blocks while the inner loop optimizes the parameters in the subproblem (6) associate with the current block.

A high level description of DSSC-ADMM is summarized in Algorithm 1. We denote the clock value as t_i which is initialized to be 0. Everytime the master thread finishes updating $\overline{\boldsymbol{A}_i}$, $\boldsymbol{E}_i$ and $\boldsymbol{U}_i$, the clock value t_i is increased by 1.

The inner loop iteration continues until the following convergence criterion is met or a desired number of iterations are reached:

$$\|\big(\overline{\boldsymbol{A}_i}^{(t_i)} - \overline{\boldsymbol{A}_i}^{(t_i-1)}\big)^T\|_\infty < \epsilon_1,\ \|\big(\boldsymbol{E}_i^{(t_i)} - \boldsymbol{E}_i^{(t_i-1)}\big)^T\|_\infty < \epsilon_2 \tag{7}$$

where ϵ_1 and ϵ_2 denote the residual error tolerance. We assemble the obtained blocks $\boldsymbol{C}_{ji}$ as the output coefficient matrix $\boldsymbol{C}$. The inner loop iteration will be described in details in the subsections to come. The following quantities are needed to simplify and decentralize our algorithm:

$$\overline{\boldsymbol{X}\boldsymbol{C}_i}^{(t_i)} = \frac{1}{m}\sum_{j=1}^m \boldsymbol{X}_j\boldsymbol{C}_{ji}^{(t_i)},\ \overline{\boldsymbol{A}_i}^{(t_i)} = \frac{1}{m}\sum_{j=1}^m \boldsymbol{A}_{ji}^{(t_i)}$$

Algorithm 1: The processing of ADMM-DSSC when calculating $\boldsymbol{C}_i$ and $\boldsymbol{E}_i$.

```
Input: $\boldsymbol{X}$, $\{\boldsymbol{X}_i\}_{i=1}^m$, $\lambda_1$, $\lambda_2$ and $\rho$
Initialize Set $t_i$ to be 0. $\overline{\boldsymbol{A}}_i^{(0)}$, $\{\boldsymbol{C}_{ji}^{(0)}\}_{j=1}^m$, $\boldsymbol{U}_i^{(0)}$ and
$\boldsymbol{E}_i^{(0)}$ to be zero matrices.
repeat
    (S1) $\boldsymbol{C}$-Update: $\forall j$, update $\boldsymbol{C}_{ji}^{(t_{ji})}$ given $\overline{\boldsymbol{A}}_i^{(t_i-1)}$,
    $\boldsymbol{E}_i^{(t_i-1)}$ and $\boldsymbol{U}_i^{(t_i-1)}$ fixed.
    (S2) $\boldsymbol{A}$-Update: Update $\overline{\boldsymbol{A}}_i$ given $\{\boldsymbol{C}_{ji}^{(t_i)}\}_{j=1}^m$,
    $\{\boldsymbol{U}_i^{(t_i-1)}\}$ and and $\{\boldsymbol{E}_i^{(t_i-1)}\}$ fixed.
    (S3) $\boldsymbol{E}$-Update: Update $\boldsymbol{E}_i^{(t_i)}$ given $\overline{\boldsymbol{A}}_i^{(t_i)}$,
    $\{\boldsymbol{C}_{ji}^{(t_i)}\}_{j=1}^m$ and $\boldsymbol{U}_i^{(t_i-1)}$ fixed.
    (S4) $\boldsymbol{U}$-Update: Update $\boldsymbol{U}_i^{(t_i)}$ given $\overline{\boldsymbol{A}}_i^{(t_i)}$ and
    $\{\boldsymbol{C}_{ji}^{(t_i)}\}_{j=1}^m$ fixed.
    $t_i \leftarrow t_i + 1$
until halting condition holds;
Output: Weight coefficient matrix $\boldsymbol{C}$.
```

$\boldsymbol{C}$-Update In this step, we fix $\overline{\boldsymbol{A}}_i^{(t_i-1)}$, $\boldsymbol{U}_i^{(t_i-1)}$ and $\boldsymbol{E}_i^{(t_i-1)}$ and update the blocks $\boldsymbol{C}_{ji}$.

- For $j \neq i$, we update $\boldsymbol{C}_{ji}^{(t_i)}$ by optimizing

$$\boldsymbol{C}_{ji}^{(t_i)} = \arg\min_{\boldsymbol{C}_{ji}} \frac{\rho}{2}\|\boldsymbol{X}_j\boldsymbol{C}_{ji} - \boldsymbol{B}_{ji}^{(t_i-1)}\|_F^2 + \lambda_1\|\boldsymbol{C}_{ji}\|_1$$

  where

$$\boldsymbol{B}_{ji}^{(t_i-1)} = \boldsymbol{X}_j\boldsymbol{C}_{ji}^{(t_i-1)} - \boldsymbol{U}_{ji}^{(t_i-1)}$$

  This is a standard multivariate Lasso regression problem which can be efficiently solved via first-order solvers such as proximal gradient method and ADMM.

- For $j = i$, we update $\boldsymbol{C}_{ii}^{(t_i)}$ as

$$\boldsymbol{C}_{ii}^{(t_i)} = \arg\min_{\boldsymbol{C}_{ii}} \frac{\rho}{2}\|\boldsymbol{X}_i\boldsymbol{C}_{ii} - \boldsymbol{B}_{ii}^{(t_i-1)}\|_F^2 + \lambda_1\|\boldsymbol{C}_{ii}\|_1$$
$$s.t \quad \operatorname{diag}(\boldsymbol{C}_{ii}) = 0$$

  This is a constrained multivariate Lasso regression problem which can be efficiently solved by applying a similar variant of ADMM as used in (Elhamifar and Vidal 2013).

This $\boldsymbol{C}$-update step is the most expensive one in our framework as it needs to solve m Lasso sub-problems. Fortunately, these sub-problems are independent to each other, thus can be solved in parallel on m different cores. By distributing the computation task, each subproblem is computationally more stable and efficient than the large scale Lasso estimator as used in the original SSC. Empirically we find that solving the above subproblem by using warm start, i.e., at the t_i-th iteration, initializing C_{ji} to be $C_{ji}^{(t_i-1)}$ can significantly boost the overall optimization efficiency.

$\boldsymbol{A}$-Update In this step, we fix $\boldsymbol{C}_{ji}^{(t_{ji})}$ and $\boldsymbol{U}_{ji}^{(t_{ji}-1)}$, $j = 1, ..., m$ and update $\overline{\boldsymbol{A}}_i^{(t_i)}$ as

$$\overline{\boldsymbol{A}}_i^{(t_i)} = \frac{1}{m+\rho}(\boldsymbol{X}_i - \boldsymbol{E}_i^{(t_i-1)} + \rho\overline{\boldsymbol{X}\boldsymbol{C}}_i^{(t_i)} + \rho\boldsymbol{U}_i^{(t_i-1)})$$

where

$$\boldsymbol{U}_i^{(t_i-1)} = \frac{1}{m}\sum_{m=1}^{M}\boldsymbol{U}_{ji}^{(t_i-1)}$$

Given $\overline{\boldsymbol{A}}_i^{(t_i)}$, $\boldsymbol{A}_{ji}^{(t_{ji})}$ can be formulated as:

$$\boldsymbol{A}_{ji}^{(t_i)} = \boldsymbol{X}_i\boldsymbol{C}_{ji}^{(t_i)} + \boldsymbol{U}_{ji}^{(t_i-1)} + \overline{\boldsymbol{A}}_i^{(t_i)} - \overline{\boldsymbol{X}\boldsymbol{C}}_i^{(t_i)} - \boldsymbol{U}_i^{(t_i-1)} \tag{8}$$

$\boldsymbol{E}$-Update In this step, we fix $\boldsymbol{C}_{ji}^{(t_i)}$, $\overline{\boldsymbol{A}}_i^{(t_i)}$ and $\boldsymbol{U}_{ji}^{(t_i-1)}$, $j = 1, ..., m$ and update $\boldsymbol{E}_i^{(t_i)}$ by solving

$$\boldsymbol{E}_i^{(t_i)} = \arg\min_{\boldsymbol{E}_i} \frac{1}{2}\|\boldsymbol{X}_i - m\overline{\boldsymbol{A}}_i^{(t_i)} - \boldsymbol{E}_i\|_F^2 + \lambda_2\|\boldsymbol{E}_i\|_{p,1}$$

Particularly, when $p = 1$, the entries of $\boldsymbol{E}_i$ can be updated using the following soft-thresholding operation

$$\boldsymbol{E}_i^{(t_i)}(l,k) = \begin{cases} \alpha(l,k)\boldsymbol{M}_i^{(t_i)}(l,k) & \text{if } |\boldsymbol{M}_i^{(t_i)}(l,k)| > \lambda_2 \\ 0 & \text{otherwise} \end{cases}$$
$$l = 1, ..., d, \ k = 1, 2, ..., n_i$$

where $\alpha(l,k) = \frac{|\boldsymbol{M}_i^{(t_i)}(l,k)| - \lambda_2}{|\boldsymbol{M}_i^{(t_i)}(l,k)|}$, $\boldsymbol{E}_i^{(t_i)}(l,k)$ is the value of $\boldsymbol{E}_i^{(t_i)}$ at entry (l,k) and $\boldsymbol{M}_i^{(t_i)} = \boldsymbol{X}_i - m\overline{\boldsymbol{A}}_i^{(t_i)}$. When $p = 2$, $\boldsymbol{E}_i^{(t)}$ has the following closed form (Liu et al. 2013):

$$\boldsymbol{E}_i^{(t_i)}(k) = \begin{cases} \beta(k)\boldsymbol{M}_i^{(t_i)}(k) & \text{if } \|\boldsymbol{M}_i^{(t_i)}(k)\|_2 > \lambda_2 \\ 0 & \text{otherwise} \end{cases}$$
$$k = 1, 2, ..., n_i$$

where $\beta(k) = \frac{\|\boldsymbol{M}_i^{(t_i)}(k)\|_2 - \lambda_2}{\|\boldsymbol{M}^{(t_i)}(k)\|_2}$, $\boldsymbol{E}_i^{(t_i)}(k)$ is the k-th column of $\boldsymbol{E}_i^{(t_i)}$. This step update can be conducted on the main thread and it only involves elementary vector operation.

$\boldsymbol{U}$-Update In this step, we firstly fix $\boldsymbol{C}_{ji}^{(t_i)}$ and $\overline{\boldsymbol{A}}_i^{(t_i)}$ and update $\boldsymbol{U}_i^{(t_i)}$ as follows:

$$\boldsymbol{U}_i^{(t_i)} = \boldsymbol{U}_i^{(t_i-1)} + \overline{\boldsymbol{X}\boldsymbol{C}}_i^{(t_i)} - \overline{\boldsymbol{A}}_i^{(t_i)}$$

After that $\boldsymbol{U}_i^{(t_i)}$ was sent back to all of the sub-treads. $\boldsymbol{U}_{ji}^{(t_i)}$ is updated through

$$\boldsymbol{U}_{ji}^{(t_i)} = \boldsymbol{U}_{ji}^{(t_i-1)} + \boldsymbol{X}_j\boldsymbol{C}_{ji}^{(t_i)} - \boldsymbol{A}_{ji}^{(t_i)}$$

according to (8) it equals to $\boldsymbol{U}_i^{(t_i)}$.

Subspace segmentation via spectral clustering

Given the output coefficient matrix $\boldsymbol{C}$, we normalize its columns as $c_i = \frac{c_i}{\|c_i\|_\infty}$. A graph $G = (V, E)$ is constructed, where V is the set of N nodes corresponding to N samples, E is the edge of the graph connecting those samples. The weight of the graph is a nonnegative sample similarity matrix $\boldsymbol{W}$ obtained by $\boldsymbol{W} = |\boldsymbol{C}| + |\boldsymbol{C}|^T$ where each value in $|\boldsymbol{C}|$ is absolute value of corresponding element in $\boldsymbol{C}$. We then feed $\boldsymbol{W}$ into the spectral clustering algorithm to divide the data points into different groups.

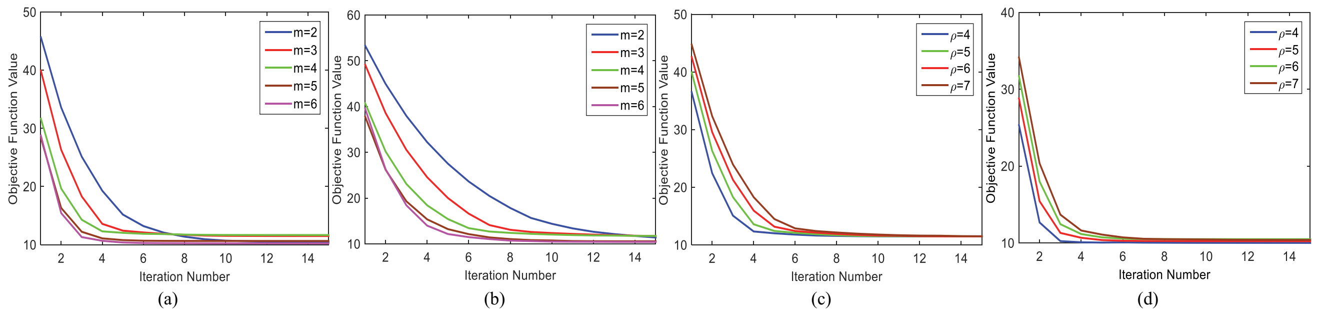

Figure 2: The convergence curves evaluated by $\|(X_i - m\overline{A_i})^T\|_\infty$ given different parameter settings. (a) The convergence curve of parameter $m = 2, 3, 4, 5, 6$, $\rho = 5$. (b) The convergence curve of parameter $m = 2, 3, 4, 5, 6$, $\rho = 10$. (c) The convergence curve of parameter $m = 3$, $\rho = 4, 5, 6, 7$. (c) The convergence curve of parameter $m = 6$, $\rho = 4, 5, 6, 7$.

Experiments

We conduct data clustering and graph based semi-supervised learning experiments on real-world data to evaluate the effectiveness and efficiency of our algorithm. In each experiment we will use the term DSSC-m to abbreviate the DSSC-ADMM algorithm with data matrix being randomly divided into m column blocks.

The algorithm is implemented in Matlab and tested on a cluster with 2.7GHz CPUs and 128GB RAM. The decentralized computing is conducted on a shared-memory architecture (Niu et al. 2011). To be specific, all samples are loaded to the memory and partitioned into column blocks.

Experiments on subspace segmentation and outlier detection

We use the clustering error $CE = N_{mis}/N$ as a metric to measure data clustering performance. Here N_{mis} is the number of misclustered samples and N is the total number of samples.

In outlier detection experiment we use the areas under the receiver operator characteristic (ROC) curve, known as AUC, for evaluation (Liu et al. 2013). To evaluate the speedup performance of DSSC-ADMM, we report the computation time of each considered DSSC-ADMM-i and calculate the speedup (SP)[1] as $SP = \frac{T_{SSC}}{T_{DSSC-ADMM-i}}$ where T_{SSC} is the execution time of the traditional SSC (Elhamifar and Vidal 2013)[2] and $T_{DSSC-ADMM-i}$ is the execution time of our method on i cores.

In this expeiment, we test DSSC-ADMM on the hand written digits image dataset MINIST (LeCun et al. 1998). The size of the original images are uniformly 28×28. The digit numbers are centered by translation according to the center of mass of pixels.

Subspace segmentation with clean data We randomly select 500 images from the dataset for each of the numbers from 0 to 9 hence there are 5000 images in total. All the images are resized to 14×14 using bilinear interpolation so that each image is represented by a 196-dimension gray value vector. We empirically set $\lambda_1 = 0.67$. m is chosen to be 2,3,4,5 and 6, respectively.

We analyze the relationship between model parameter value and objective function convergence. There are two important parameters in our model, which are the number of block m and the value of ρ, respectively. We illustrate the model convergence through the value of $\|(X_i - m\overline{A_i}^{(t_i)})^T\|_\infty$. We first fix ρ to be 5 and 10, then test the convergence under different value of m. The results are shown in Fig. 2(a) and Fig. 2(b). After that we fix the value of m to be 3 and 6, then test the convergence under different value of ρ. Fig. 2(c) and Fig. 2(d) illustrate the results. The analysis indicates that by properly choosing the parameters, the proposed method can converge fast after several iterations.

The CE of SSC and DSSC-ADMM is shown in Fig. 3(a). The time cost and SP are shown in Fig. 3(b) and Fig. 3(c). It can be observed that DSSC-ADMM method is able to reduce around $55\% - 85\%$ time cost if the computation are distributed on $2 - 6$ cores, with comparable accuracy.

Joint subspace segmentation and outlier detection We next test the performance of jointly clustering the digit images and detecting the outliers. The pixels of outlier images are randomly generated as uniform noise from $[0, 255]$. 1000 outlier images are produced and merged with 5000 digit images as dataset. The parameters λ_1 and λ_2 are set to be 0.67 and 1.5, respectively. We evaluate CE of digit images and AUC of outlier detection result. The performance is also shown in Fig. 3(a). According to the above results, the CE result is almost unchanged and outliers can be perfectly detected by both of DSSC-ADMM and SSC. The time cost and SP are shown in Fig. 3(b) and Fig. 3(c), respectively. The proposed method can reduce around $68\% - 90\%$ time cost by using 2 to 6 cores.

Experiments on semi-supervised learning

In this part we consider applying the proposed method to graph based semi-supervised learning. Recent works that formulate the graph construction into a sparse data recon-

[1]Please refer to http://en.wikipedia.org/wiki/Speedup.

[2]In our implementation the parameters of SSC are learned by solving (3) through ADMM. The optimization termination criterion is set as $\|(C^{(t)} - C^{(t-1)})^T\|_\infty \leq 0.01$, $\|(E^{(t)} - E^{(t-1)})^T\|_\infty \leq 0.01$ or 200 iterations is reached.

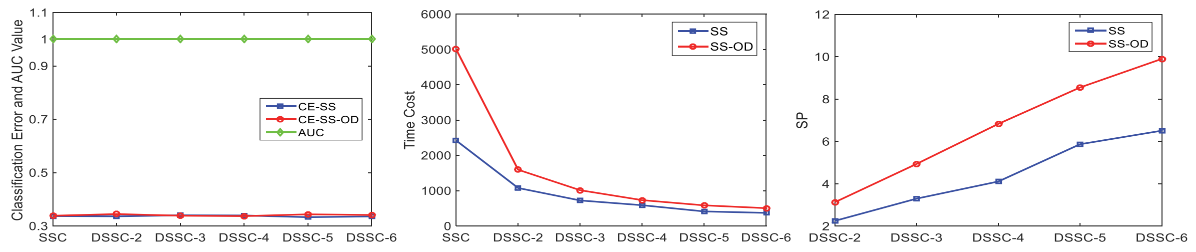

Figure 3: Performance and time cost comparison of subspace segmentation and outlier detection between SSC and DSSC-ADMM on 5000 MNIST digit image dataset. (a) Performance comparison. CE-SS is the CE value of subspace segmentation experiment; CE-SS-OD and AUC are the CE and AUC value of joint data clustering and outlier detection experiment. (b) Time cost comparison (in seconds). (c) SP values.

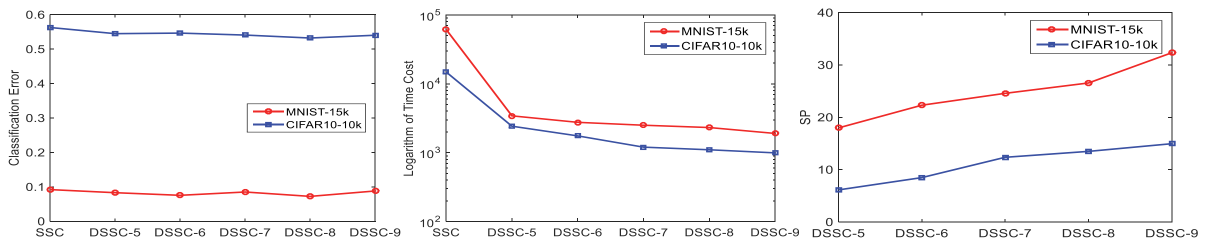

Figure 4: Performance and time cost comparison on MNIST-15k and CIFAR10-10k datasets. (a) Classification error comparison on both of the two datasets. (b) Logarithm of time cost comparison (in seconds). (c) SP values.

struction problem have achieved great success (Yan and Wang 2009; Cheng et al. 2010; Wang, Liu, and Wu 2015). The datasets we test include:

- MINIST-15k dataset. 1500 images per digit are selected to form the dataset. To evaluate the classification error, 10 labeled samples are randomly selected from each digit. We set $\lambda_1 = 1$ to build the sparse graph.
- CIFAR10-10k dataset. CIFAR10 is composed of 60000 images from 10 categories (Krizhevsky and Hinton 2009). Each image has 32×32 pixels. We collect 1000 images from each category. 512-dimension gist feature (Oliva and Torralba 2001) is extracted as feature representation for each image. 100 labelled samples are randomly selected from each category. The parameter λ_1 is set to be 0.1.

We repeat the semi-supervised classification testing 10 times to calculate the average classification error. The average classification errors of two datasets are shown in Fig. 4(a). We can conclude that the proposed DSSC-ADMM introduces no loss to the semi-supervised classification accuracy. According to the time cost and SP value shown in Fig.4(b) and Fig. 4(c), only less than 6% time cost is needed for MNIST-15k dataset and less than 17% time cost is needed for CIFAR10-10k if DSSC-ADMM is run on 5 to 9 cores.

Conclusion and Discussion

In this paper, we proposed DSSC-ADMM as a large scale decentralized framework for computing SSC via an alternating direction method of multiplier (ADMM) algorithm. The key innovation of DSSC-ADMM is to solve SSC in column blocks which can be compactly distributed on multiple cores for estimating the parameters associated with these blocks. This leads to substantially more efficient optimization than the traditional column-wise estimation based SSC methods. We evaluate our algorithm on a shared-memory architecture with multiple cores. Extensive experiments on both synthetic and read-world datasets confirm that the proposed method can significantly accelerate SSC, at almost no cost of clustering accuracy. It is interesting to note that DSSC-ADMM can be directly applied to implement a distributed variant of the Neighborhood Selection Lasso (NS-Lasso) method for Gaussian graphical models estimation (Meinshausen and Bühlmann 2006). The NS-Lasso method directly estimates the support of the sparse precision matrix using separate neighborhood estimations for each variable followed by a proper aggregation rule. In formulation, NS-Lasso is equivalent to SSC in (3) without considering the noise term $\boldsymbol{E}$. Therefore, DSSC-ADMM method can be directly applied to decentralized neighborhood selection.

Acknowledgments

Xiao-Tong Yuan was supported partially by the National Natural Science Foundation of China under Grant 61402232, 61522308, and partially by the Natural Science Foundation of Jiangsu Province of China under Grant BK20141003. Qingshan Liu was supported partially by National Natural Science Foundation of China under Grant 61532009.

References

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* 3(1):1–122.

Bradley, P. S., and Mangasarian, O. L. 2000. K-plane clustering. *Journal of Global Optimization* 16(1):23–32.

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *Journal of the ACM* 58(3):11.

Chen, W.-Y.; Song, Y.; Bai, H.; Lin, C.-J.; and Chang, E. Y. 2011. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(3):568–586.

Cheng, B.; Yang, J.; Yan, S.; Fu, Y.; and Huang, T. S. 2010. Learning with l1 graph for image analysis. *IEEE Transactions on Image Processing* 19(4):858–866.

Costeira, J. P., and Kanade, T. 1998. A multibody factorization method for independently moving objects. *International Journal of Computer Vision* 29(3):159–179.

Elhamifar, E., and Vidal, R. 2009. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Elhamifar, E., and Vidal, R. 2013. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(11):2765–2781.

Feng, J.; Lin, Z.; Xu, H.; and Yan, S. 2014. Robust subspace segmentation with block-diagonal prior. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Fu, Q.; Wang, H.; and Banerjee, A. 2013. Bethe-ADMM for tree decomposition based parallel map inference. In *Conference on Uncertainty in Artificial Intelligence*.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Li, M.; Zhou, L.; Yang, Z.; Li, A.; Xia, F.; Andersen, D. G.; and Smola, A. 2013. Parameter server for distributed machine learning. In *Big Learning Workshop at NIPS*.

Liang, Y.; Balcan, M.-F.; and Kanchanapally, V. 2013. Distributed pca and k-means clustering. In *The Big Learning Workshop at NIPS*.

Liu, G.; Lin, Z.; Yan, S.; Sun, J.; Yu, Y.; and Ma, Y. 2013. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(1):171–184.

Lu, C.-Y.; Min, H.; Zhao, Z.-Q.; Zhu, L.; Huang, D.-S.; and Yan, S. 2012. Robust and efficient subspace segmentation via least squares regression. In *European Conference on Computer Vision*.

Luo, D.; Nie, F.; Ding, C.; and Huang, H. 2011. Multisubspace representation and discovery. In *Machine Learning and Knowledge Discovery in Databases*. Springer. 405–420.

Ma, Y.; Yang, A. Y.; Derksen, H.; and Fossum, R. 2008. Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM Review* 50(3):413–458.

Meinshausen, N., and Bühlmann, P. 2006. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics* 34(3):1436–1462.

Mota, J. F.; Xavier, J. M.; Aguiar, P. M.; and Puschel, M. 2013. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing* 61(10):2718–2723.

Niu, F.; Retcht, B.; Re, C.; and Wright, S. J. 2011. Hogwild! a lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*.

Oliva, A., and Torralba, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision* 42(3):145–175.

Raina, R.; Madhavan, A.; and Ng, A. Y. 2009. Large-scale deep unsupervised learning using graphics processors. In *International Conference on Machine Learning*.

Sedghi, H.; Anandkumar, A.; and Jonckheere, E. 2014. Multistep stochastic ADMM in high dimensions: applications to sparse optimization and matrix decomposition. In *Advances in Neural Information Processing Systems*.

Sparks, E. R.; Talwalkar, A.; Smith, V.; Kottalam, J.; Pan, X.; Gonzalez, J.; Franklin, M. J.; Jordan, M. I.; and Kraska, T. 2013. MLI: An API for distributed machine learning. In *IEEE International Conference on Data Mining*.

Suzuki, T. 2013. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *International Conference on Machine Learning*.

Tipping, M., and Bishop, C. 1999. Mixtures of probabilistic principal component analyzers. *Neural Computation* 11(2):443–482.

Vidal, R. 2010. A tutorial on subspace clustering. *IEEE Signal Processing Magazine* 28(2):52–68.

Wang, H.; Banerjee, A.; jui Hsieh, C.; Ravikumar, P.; and Dhillon, I. 2013. Large scale distributed sparse precision estimation. In *Advances in Neural Information Processing Systems*.

Wang, M.; Liu, X.; and Wu, X. 2015. Visual classification by l1-hypergraph modeling. *IEEE Transactions on Knowledge and Data Engineering* 27(9):2564–2574.

Wei, E., and Ozdaglar, A. 2012. Distributed alternating direction method of multipliers. In *IEEE Annual Conference on Decision and Control*.

Wei, E., and Ozdaglar, A. 2013. On the O$(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. In *IEEE Global Conference on Signal and Information Processing*.

Yan, S., and Wang, H. 2009. Semi-supervised learning by sparse representation. In *SIAM International Conference on Data Mining*.

Zhang, R., and Kwok, J. 2014a. Asynchronous distributed ADMM algorithm for global variable consensus optimization. In *International Conference on Machine Learning*.

Zhang, R., and Kwok, J. 2014b. Asynchronous distributed ADMM for consensus optimization. In *International Conference on Machine Learning*.