Simultaneous Clustering and Optimization for Evolving Datasets

Yawei Zhao, En Zhu, Xinwang Liu, Chang Tang, Deke Guo, *Senior Member, IEEE*, and Jianping Yin

Abstract—Simultaneous clustering and optimization (SCO) has recently drawn much attention due to its wide range of practical applications. Many methods have been previously proposed to solve this problem and obtain the optimal model. However, when a dataset evolves over time, those existing methods have to update the model frequently to guarantee accuracy; such updating is computationally infeasible. In this paper, we propose a new formulation of SCO to handle evolving datasets. Specifically, we propose a new variant of the alternating direction method of multipliers (ADMM) to solve this problem efficiently. The guarantee of model accuracy is analyzed theoretically for two specific tasks: ridge regression and convex clustering. Extensive empirical studies confirm the effectiveness of our method.

Index Terms—Simultaneous clustering and optimization, evolving datasets, sum-of-norms regularizer, ADMM

1 Introduction

Simultaneous clustering and optimization (SCO) has recently drawn much attention in the machine learning and data mining community [1], [2]. Let us consider an example to explain this. Suppose that we want to predict the price of houses in New York City. The prices of houses located in the same region should be predicted by using similar prediction models. The prices of houses located in different regions should be predicted by using different prediction models. Traditional methods usually involve two separate steps. Such methods first learn prediction models for every house and then use clustering methods such as k-means clustering [3], [4], [5], [6] to determine the similarity among the obtained prediction models. However, the purpose of the SCO task is to perform prediction and identify the similarity among prediction models simultaneously; this approach usually outperforms traditional solutions.

Previous methods such as network lasso [1], [2], [7] formulate the SCO problem as that of a convex objective function with a sum-of-norms regularizer, which usually leads to a high computational cost for to following reasons:

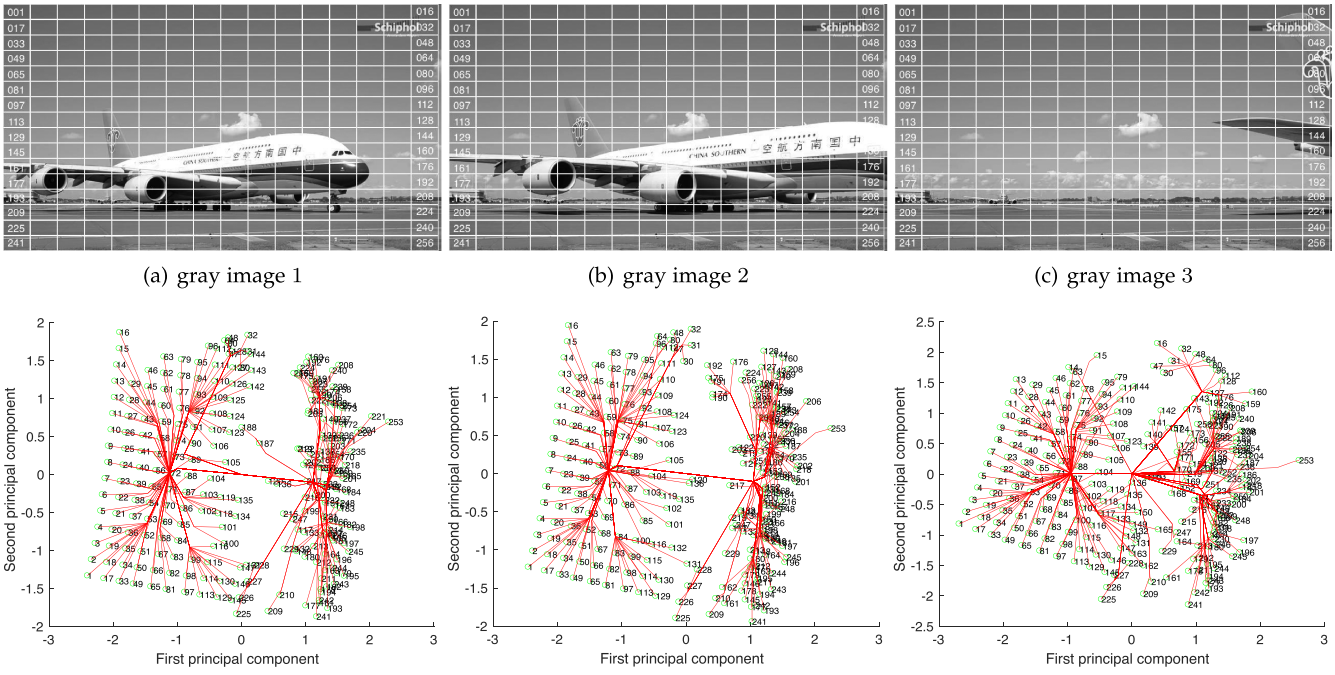
- The number of optimization variables increases linearly with the number of instances and features.
- The optimization variables are highly nonseparable due to the sum-of-norms regularization.
- The objective function is extremely nonsmooth.

Due to those challenges, many optimization methods such as the alternating minimization algorithm (AMA) [8] and the alternating direction method of multipliers (ADMM) [1], [2] have been developed to reduce the computational cost. Although these existing methods obtain great efficiency improvement for a static dataset, they are unable to handle an evolving dataset directly due to exceptionally high computational complexity. The reason is that when a dataset evolves over time, the optimal model of SCO has to be frequently updated over time. Otherwise, the model accuracy cannot be guaranteed. However, it is impractical to update the optimal model frequently due to the high computational cost. Therefore, finding an effective method of performing SCO on an evolving dataset with a guarantee of model accuracy is a meaningful problem.

Let us consider an example to explain our motivation. Determining a cluster path [8], [9], [10] is one of SCO tasks. This is done by performing convex clustering over multiple rounds. As illustrated in Fig. 1, the cluster paths for images 1 and 2 are very similar due to few changes of pixels. However, the cluster path of image 3, namely, Fig. 1f, is significantly different from them. It is impractical to update the optimal cluster path for every image timely because it takes at least 18 seconds to obtain a cluster path. Additionally, compared with Figs. 1d and 1e, we observe that a cluster path for image 1, i.e., Fig. 1d, can be used for a similar image 2 with some slight loss of accuracy. Compared with Figs. 1d and 1f, we observe that a cluster path should be updated if the image is changed significantly. In a general scenario, we are thus motivated by the following two nontrivial and challenging problems:

- *Y. Zhao, E. Zhu, and X. Liu are with the College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China. E-mail: {zhaoyawei, enzhu, xinwangliu}@nudt.edu.cn.*
- *C. Tang is with the School of Computer Science, China University of Geosciences, Wuhan, Hubei 430074, China. E-mail: tangchang@cug.edu.cn.*
- *D. Guo is with the Science, Technology, and Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, Hunan 410073, China. E-mail: guodeke@gmail.com.*
- *J. Yin is with the Dongguan University of Technology, Dongguan, Guangdong 523000, China. E-mail: jpyin@dgut.edu.cn.*

Manuscript received 18 July 2018; revised 25 Apr. 2019; accepted 6 June 2019. Date of publication 17 June 2019; date of current version 7 Dec. 2020. (Corresponding author: Yawei Zhao.) Recommended for acceptance by F. Afrati. Digital Object Identifier no. 10.1109/TKDE.2019.2923239



(a) gray image 1 (b) gray image 2 (c) gray image 3

(d) cluster path 1, 19.7 seconds (e) cluster path 2, 19.0 seconds (f) cluster path 3, 18.8 seconds

Fig. 1. There is a plane at an airport, and it is crossing the field of view. Fig. 1(b) shows the next frame in the video after the frame shown in Fig. 1(a). Fig. 1(a) shows that the plane is nearly outside the field of view after 2 seconds. We partition every image into 256 blocks and identify each block with a unique number. The maximum gray value corresponding to pixels in a block is used to represent the block. The cluster paths of Figs. 1(a), 1(b), and 1(c) are illustrated in Figs. 1(d), 1(e), and 1(f), respectively. Note that the cluster path of an image is obtained by running the previous method for more than 18 seconds, which is unacceptably long in video analysis. This illustrative example shows that it is vitally important to find an effective method for obtaining the cluster path with a guaranteed accuracy for a video dataset.

- How to obtain a model with a guaranteed accuracy that can be used for a dataset undergoing slight changes, and
- When to update the model if the dataset has evolved to become sufficiently different.

In this paper, we aim to answer the above two challenging questions. We reformulate the problem of SCO as a convex problem with cone constraints in the dual space. Compared with the formulation in the previous studies, the new formulation does not contain the sum-of-norms regularizer and is thus much easier to solve. Then, a new regularizer is proposed to allow for the optimal model to be insensitive to slightly evolving data. A new metric is proposed for deciding when to update the model for a significant change of a dataset. Additionally, we propose a new variant of the alternating direction method of multipliers (ADMM) to solve the proposed problem efficiently. Furthermore, the guarantee of the model accuracy is analyzed theoretically for convex clustering and ridge regression tasks. Extensive empirical studies show the advantages of the proposed method. In brief, our contributions are summarized as follows:

- A new formulation of the SCO problem is proposed to handle an evolving dataset.
- An efficient variant of ADMM is proposed to solve the proposed problem efficiently.
- The accuracy of the model is analyzed theoretically for convex clustering and ridge regression tasks.
- Extensive empirical studies show the superiority of the proposed method.

The paper is organized as follows. Section 2 reviews the related studies. Section 3 introduces the preliminaries. Section 4 describes a new formulation of SCO for an evolving dataset. Section 5 discusses a novel ADMM method for solving the problem. Section 6 presents a theoretical analysis. Section 7 details the experiments. Section 8 concludes the paper.

2 Related Studies

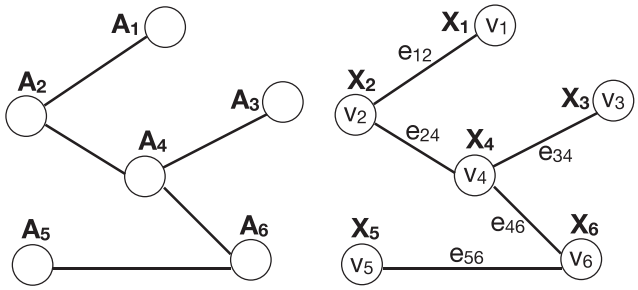
In this section, we briefly review the related literature.

2.1 Network Lasso

Network lasso was the first method proposed to solve the SCO problem [1]. It was subsequently implemented and published as a general optimization tool for graph analysis in [11]. Recently, network lasso has drawn much attention in various application scenarios [2], [7], [12]. It was used in [7] to predict the location of a shared bike. The study [12] examined a sufficient condition for the network topology to yield an accurate solution. Network lasso was extended in [2] to handle noisy and missing data. The cited existing studies extended network lasso for static datasets; however, such approaches are unsuitable for handling evolving datasets due to a high computational cost.

2.2 Convex Clustering

The special case of SCO, i.e., convex clustering [13], has been extensively studied for several years. Compared to the traditional clustering methods, it has a convex formulation, leading to a robust clustering result. For example, the clustering



(a) data graph (b) variable graph $\mathcal{G}$

Fig. 2. Illustrative example of the graph abstraction. For any i such that $1 \leq i \leq 6$, $\mathbf{A}_i$ represents an instance of the dataset, $\mathbf{X}_i$ represents the corresponding optimization variable, v_i represents a vertex of graph $\mathcal{G}$, and e_{ij} represents the edge connecting v_i and v_j.

result of k-means is sensitive to the seeds, and picking good seeds is challenging [3], [14]. However, due to a convex objective function, the result of convex clustering can be determined. It is not impacted by any heuristic rules used in traditional clustering methods. A formulation of convex clustering was proposed in [13] by relaxing the formulation of k-means clustering. Subsequently, [15] and [16] provided several sufficient conditions for recovering the clustering membership theoretically. Other studies, e.g., [8], [17], focus on improving the efficiency of convex clustering. Although those previous studies attained great improvement of convex clustering for static datasets, they are unsuitable for handling evolving datasets due to a high computational cost. The method proposed in the paper reduces such computational cost and makes a good tradeoff between efficiency and accuracy.

3 Preliminaries

In this section, several important notations are introduced. Then, the problem of simultaneous clustering and optimization is presented.

3.1 SCO: Simultaneous Clustering and Optimization

SCO is an optimization framework with a formulation that is usually presented as a convex objective function with a sum-of-norms regularizer. Before discussing its formulation, we introduce the data model—a graph abstraction of a dataset.

The basic data model consists of two graphs—a data graph $\mathcal{G}^{\text{data}}$ and a variable graph $\mathcal{G}^{\text{variable}}$—that are generated as follows.

- *Data graph* $\mathcal{G}^{\text{data}}$. Every instance in a dataset is represented by a vertex. Generally, the K-nearest neighbors (K-NN) method is performed on the dataset. If an instance is one of the K nearest neighbors of another instance, then an edge is generated to connect them. Thus, we obtain a graph $\mathcal{G}^{\text{data}}$ that measures the similarity among instances.
- *Variable graph* $\mathcal{G}^{\text{variable}}$. Variable graph $\mathcal{G}^{\text{variable}}$ is generated based on graph $\mathcal{G}^{\text{data}}$. An optimization variable, e.g., $\mathbf{X}_i$, is represented by a vertex v_i. If two instances, e.g., $\mathbf{A}_i$ and $\mathbf{A}_j$ in graph $\mathcal{G}^{\text{data}}$, are connected by an edge, then vertices v_i and v_j are connected by edge e_{ij}.

Let us consider an illustrative example as a further explanation. Fig. 2a shows a data graph for a dataset consisting of 6 instances. Fig. 2b shows the variable graph for the data graph $\mathcal{G}^{\text{data}}$ in the left panel. Since SCO is formulated based on the variable graph $\mathcal{G}^{\text{variable}}$, we denote the variable graph $\mathcal{G}^{\text{variable}}$ by $\mathcal{G}$ by default in the following content. The vertex set of $\mathcal{G}$ is denoted by $\mathcal{V}$, and the edge set of $\mathcal{G}$ is denoted by $\mathcal{E}$. Thus, SCO [1] is formulated as

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times d}} \underbrace{\sum_{v_i\in\mathcal{V}} f_i(\mathbf{X}_i)}_{\text{the empirical loss}} + \alpha \underbrace{\sum_{e_{ij}\in\mathcal{E}} w_{ij}\left\|\mathbf{X}_i - \mathbf{X}_j\right\|_p}_{\text{the sum-of-norms regularizer}}. \tag{1}$$

Here, p is picked from $1, 2, \infty$. Parameter α is a hyperparameter that should be specified. Parameter w_{ij} is the weight of the edge e_{ij}. When the variable graph $\mathcal{G}$ is obtained, w_{ij} is usually set to be inversely proportional to the distance between $\mathbf{A}_i$ and $\mathbf{A}_j$ [8]. The optimal value of $\mathbf{X}_i$, denoted by $\mathbf{X}_{i*}$, represents the prediction model for the ith instance, i.e., $\mathbf{A}_i$.

The complete objective function of (1) consists of two parts explained as follows.

- *Empirical loss.* SCO can be used on various data analysis tasks by instantiating the empirical loss of $f_i(\mathbf{X}_i)$. Function $f_i(\cdot)$ usually represents the empirical loss due to the ith instance, which is determined by a specific machine learning task, such as regression, classification, and clustering. For example, if

  $$\sum_{v_i\in\mathcal{V}} f_i(\mathbf{X}_i) = \sum_{v_i\in\mathcal{V}} \|\mathbf{X}_i - \mathbf{A}_i\|_2^2 = \|\mathbf{X} - \mathbf{A}\|_F^2,$$

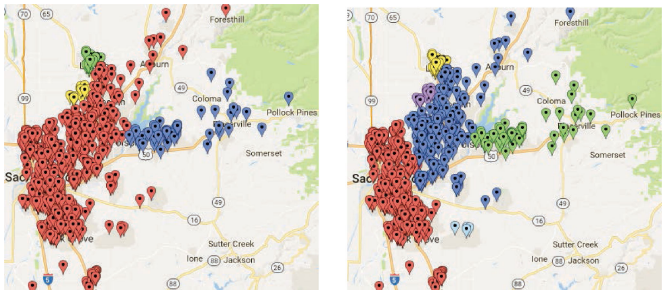
  holds, it represents the empirical loss of convex clustering. If the number of vertices in $\mathcal{V}$ is n, and

  $$\sum_{v_i\in\mathcal{V}} f_i(\mathbf{X}_i) = \sum_{i=1}^{n}\left(\left\|\mathbf{A}_i\mathbf{X}_i^{\mathrm{T}} - \mathbf{y}_i\right\|_2^2 + \gamma\|\mathbf{X}_i\|_2^2\right),$$

  holds, it represents the empirical loss of ridge regression.
- *Sum-of-norms regularizer.* The sum-of-norms regularizer $\left\|\mathbf{X}_i - \mathbf{X}_j\right\|_p$ is essential for performing simultaneous clustering and optimization. As illustrated in (1), SCO learns a model, e.g., $\mathbf{X}_i$, for every instance, e.g., $\mathbf{A}_i$. The most noticeable difference between SCO and the traditional machine learning task is the *sum-of-norms regularizer*. The regularizer controls the similarity among those models. If $\alpha = 0$, every instance has a different model from those of others. With the increase in α, some instances tend to have similar or even identical models. If α is sufficiently large, all instances may have the same model. We refer to an illustrative example [2] in Fig. 3 to provide further explanations about the regularizer.

3.2 Notation

Suppose that graph $\mathcal{G}$ consists of n vertices and m edges. The ith vertex is represented by v_i, and its corresponding optimization variable is $\mathbf{X}_i \in \mathbb{R}^{1\times d}$. The edge connecting v_i and v_j is represented by e_{ij}. The weight of e_{ij} is denoted by w_{ij}. $\mathbf{A} \in \mathbb{R}^{n\times d}$ represents the data matrix consisting of n instances, and every instance is characterized by d features.



(a) $\alpha = 6$ (b) $\alpha = 4$

Fig. 3. Example illustrating SCO if $f_i(\mathbf{X}_i)$ is instantiated to be the empirical loss of ridge regression [2]. The task needs to learn a prediction model for every house in the Greater Sacramento area. A marker represents a house. If markers have the same color, the corresponding houses have identical prediction models. As we observe, with the increase in α, the same model is used for more houses. The experimental details are described in [2].

$\mathbf{X} \in \mathbb{R}^{n\times d}$ represents the matrix of optimization variables. The other important notations are shown as follows.

- Ordinary lowercase letters, e.g., α and β, represent constant scalars. Bold lowercase letters, e.g., $\mathbf{y}$, represent vectors. Bold capital letters, e.g., $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $\mathbf{Q} \in \mathbb{R}^{m\times n}$, represent matrices.
- A bold capital letter with a subscript, e.g., $\mathbf{A}_i \in \mathbb{R}^{1\times d}$, represents the ith row of a matrix. A bold capital letter with two subscripts, e.g., $\mathbf{Q}_{ij}$, represents the element located at the ith row and the jth column.
- $\mathrm{vec}(\cdot)$ represents the column stacking vectorization of a matrix.
- $\mathrm{diag}(\mathbf{v})$ represents the diagonal matrix consisting of the elements of vector $\mathbf{v}$.
- $\otimes$ represents the Kronecker product. $\odot$ represents the Hadamard product.
- $\|\cdot\|$ represents a norm of a vector, and $\|\cdot\|_*$ represents its dual norm.
- $\mathbf{I}_d$ represents the identity matrix. $\mathbf{0}$ and $\mathbf{1}$ represent constant matrices with elements of 0 and 1, respectively.

4 Simultaneous Clustering and Optimization for Evolving Datasets

In this section, we propose a new formulation for performing SCO. Then, several examples are provided to present more details about the formulation.

4.1 Formulation

Since the dataset is evolving, the change in the data matrix $\mathbf{A}$ will impact the solution. Without a loss of generality, we reformulate (1) as follows:

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times d}} f(\mathbf{X};\mathbf{A}) + \alpha \sum_{e_{ij}\in\mathcal{E}} w_{ij}\left\|\mathbf{X}_i - \mathbf{X}_j\right\|_p. \tag{2}$$

Suppose that the number of vertices in $\mathcal{V}$ is n, i.e., $n = |\mathcal{V}|$, and the number of edges in $\mathcal{E}$ is m, i.e., $m = |\mathcal{E}|$. We introduce an auxiliary matrix $\mathbf{Q} \in \mathbb{R}^{m\times n}$ to help reformulate the optimization's objective function. Any row of $\mathbf{Q}$ consists of one positive element, one negative element and the remaining zero elements. Any row of $\mathbf{Q}$ corresponds to an edge of graph $\mathcal{G}$. If the kth row of $\mathbf{Q}$, i.e., $\mathbf{Q}_k$, has a positive value at the ith element and a negative value at the jth element, it corresponds to edge e_{ij}. In this case, the positive element of $\mathbf{Q}_k$ is $\mathbf{Q}_{ki} := \alpha w_{ij}$, and the negative element of $\mathbf{Q}_k$ is $\mathbf{Q}_{kj} := -\alpha w_{ij}$. In other words,

$$\mathbf{Q}_k = (\mathbf{0}, \underbrace{\mathbf{Q}_{ki}}_{\mathbf{Q}_{ki}:=\alpha w_{ij}}, \mathbf{0}, \underbrace{\mathbf{Q}_{kj}}_{\mathbf{Q}_{kj}:=-\alpha w_{ij}}, \mathbf{0}). \tag{3}$$

Additionally, the sum-of-norms regularization is usually denoted by the $l_{1,p}$ norm, i.e., $\|\cdot\|_{1,p}$ [1], [8], [16], [17]. It is defined by $\|\cdot\|_{1,p} := \sum_{e_{ij}\in\mathcal{E}} \|\cdot\|_p$. Using $\mathbf{Q}$ to reformulate the optimization objective function of (2), we thus obtain

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times d}} f(\mathbf{X};\mathbf{A}) + \|\mathbf{Q}\mathbf{X}\|_{1,p}.$$

TABLE 1
l_p Norm and Its Dual Norm l_q

$\|\cdot\|_p$	$\|\cdot\|_1$	$\|\cdot\|_2$	$\|\cdot\|_\infty$
$\|\cdot\|_q$	$\|\cdot\|_\infty$	$\|\cdot\|_2$	$\|\cdot\|_1$
constraints for $\|\cdot\|_q$	box	ball	simplex

Theorem 1. *The dual formulation of (2) is equivalent to*

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}^{m\times d}} f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q}); \mathbf{A}), \tag{4}$$

subject to

$$\|\boldsymbol{\lambda}_i\|_q \leq 1, 1 \leq i \leq m.$$

Once the quantity $\boldsymbol{\lambda}_*$ that minimizes (4) has been obtained, the quantity $\mathbf{X}_*$ that minimizes the primal problem (2) is obtained by

$$\nabla f(\mathbf{X}_*;\mathbf{A}) + (\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}_*) = 0. \tag{5}$$

Here, $p \in \{1, 2, \infty\}$, and $\frac{1}{p} + \frac{1}{q} = 1$. Table 1 shows the specific values of p and q.

Therefore, our new formulation of the dual problem is

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}^{m\times d}} f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q}); \mathbf{A}) + \beta\left\|(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\right\|_s \tag{6}$$

subject to

$$\|\boldsymbol{\lambda}_i\|_q \leq 1, 1 \leq i \leq m.$$

Here, s can be 1, 2 or ∞. Parameter β controls the accuracy of the solution for the evolving dataset. A small β means that the solution of (6) is sensitive to a perturbation of the data matrix $\mathbf{A}$. A large β causes the solution of (6) to be robust to a perturbation of the data matrix.

Intuitive Idea. When $\mathbf{A}$ evolves to be $\mathbf{A} + \boldsymbol{\Delta}$, the minimizer $\boldsymbol{\lambda}_*$ is not the true minimizer $\tilde{\boldsymbol{\lambda}}_*$ of (4) based on the evolving dataset $\mathbf{A} + \boldsymbol{\Delta}$. However, if the difference between $\mathbf{A}$ and $\mathbf{A} + \boldsymbol{\Delta}$ is not significant, it is reasonable to use $\boldsymbol{\lambda}_*$ as an approximation of the true minimizer of (4). Note that

$$\|\nabla f(\mathbf{X}_*;\mathbf{A})\| = \left\|(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}_*)\right\|.$$

Consider a fixed $\mathbf{X}_*$; then, $\|\nabla f(\mathbf{X}_*;\mathbf{A})\|$ changes when $\mathbf{A}$ evolves into $\mathbf{A}+\mathbf{\Delta}$. The smaller the change is between $\|\nabla f(\mathbf{X}_*;\mathbf{A})\|$ and $\|\nabla f(\mathbf{X}_*;\mathbf{A}+\mathbf{\Delta})\|$, the closer $\mathbf{X}_*$ and the true minimizer $\tilde{\mathbf{X}}_*$ based on $\mathbf{A}+\mathbf{\Delta}$. Making $\|\nabla f(\mathbf{X}_*;\mathbf{A})\|$ insensitive to $\mathbf{\Delta}$ is equivalent to making $\|(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}_*)\|$ insensitive to $\mathbf{\Delta}$. The reason is that $\boldsymbol{\lambda}_*$ determines $\mathbf{X}_*$ according to (5). We add a regularizer $\|(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\|_s$ into the objective function of (4), which penalizes the objective function for a large $\|(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\|_s$. When $\mathbf{A}$ evolves, the regularizer guarantees that the change in $\|(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\|_s$ is bounded within the user's control.

We define a new metric to decide when to update the model for the evolving dataset.

$$\begin{aligned}\Delta_{f^*} :=& |f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda}_*)(\mathbf{I}_d\otimes\mathbf{Q});\mathbf{A}+\mathbf{\Delta})\\ &- f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda}_*)(\mathbf{I}_d\otimes\mathbf{Q});\mathbf{A})|.\end{aligned} \tag{7}$$

If this metric exceeds a threshold, then the model needs to be updated. Finally, we propose Algorithm 1 to perform SCO for evolving datasets.

Algorithm 1. SCO for Evolving Datasets

Require: Data matrix $\mathbf{A}$, a positive β and a threshold c to control the accuracy of the solution for evolving datasets.

1: **while** The change in $\mathbf{A}$, i.e., $\mathbf{A}+\mathbf{\Delta}$, is detected. **do**
2: Obtain Δ_{f^*} according to (7).
3: **if** $\Delta_{f^*}\geq c$ **then**
4: $\mathbf{A}\leftarrow\mathbf{A}+\mathbf{\Delta}$.
5: Solve the optimization problem (6), and obtain the minimizer $\boldsymbol{\lambda}_*$.
6: Obtain $\mathbf{X}_*$ according to (5).

4.2 Examples

We provide two examples—convex clustering [13] and ridge regression [18]—to present additional explanations of the proposed formulation.

Convex Clustering. The optimization objective function of convex clustering is

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times d}}\|\mathbf{X}-\mathbf{A}\|_F^2+\|\mathbf{Q}\mathbf{X}\|_{1,p}.$$

In this case, $f(\mathbf{X};\mathbf{A})=\|\mathbf{X}-\mathbf{A}\|_F^2$, and $f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda};\mathbf{A})\ (\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}})=-\mathrm{vec}^{\mathrm{T}}(\mathbf{A})(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})+\frac{1}{4}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d\otimes\mathbf{Q})(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})$. We thus obtain the formulation of the dual problem of convex clustering as

$$\begin{aligned}\min_{\boldsymbol{\lambda}\in\mathbb{R}^{m\times d}}\ &-\mathrm{vec}^{\mathrm{T}}(\mathbf{A})(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\\ &+\frac{1}{4}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d\otimes\mathbf{Q})(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}),\end{aligned}$$

subject to

$$\|\boldsymbol{\lambda}_i\|_q\leq 1, 1\leq i\leq m.$$

The dual problem of convex clustering is

$$\begin{aligned}\min_{\boldsymbol{\lambda}\in\mathbb{R}^{m\times d}}\ &-\mathrm{vec}^{\mathrm{T}}(\mathbf{A})(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\\ &+\frac{1}{4}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d\otimes\mathbf{Q})(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\\ &+\beta\left\|(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\right\|_s\end{aligned}$$

subject to

$$\|\boldsymbol{\lambda}_i\|_q\leq 1, 1\leq i\leq m.$$

Once the minimizer $\boldsymbol{\lambda}_*$ of (6) has been obtained, the minimizer of the primal problem (2) is obtained according to (5), i.e., $2(\mathrm{vec}(\mathbf{X})-\mathrm{vec}(\mathbf{A}))+(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}_*)=0$ in this case.

Ridge Regression. The objective function of ridge regression with the sum-of-norms regularizer is

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times d}}\sum_{i=1}^{n}\left(\left\|\mathbf{A}_i\mathbf{X}_i^{\mathrm{T}}-\mathbf{y}_i\right\|_2^2+\gamma\|\mathbf{X}_i\|_2^2\right)+\|\mathbf{Q}\mathbf{X}\|_{1,p}.$$

After the constant item is discarded, the above becomes equivalent to

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times d}}\mathrm{vec}^{\mathrm{T}}(\mathbf{X})\mathbf{\Omega}\mathrm{vec}(\mathbf{X})-2\mathbf{y}^{\mathrm{T}}\mathbf{\Lambda}\mathrm{vec}(\mathbf{X})+\|\mathbf{Q}\mathbf{X}\|_{1,p},$$

where $\mathbf{\Omega}\in\mathbb{R}^{(nd)\times(nd)}$ is defined by

$$\mathbf{\Omega}:=\mathrm{diag}(\mathrm{vec}(\mathbf{A}))\mathrm{diag}(\mathrm{vec}(\mathbf{A}))+\gamma\mathbf{I}_{nd},$$

and $\mathbf{\Lambda}\in\mathbb{R}^{n\times(nd)}$ is defined by

$$\mathbf{\Lambda}:=(\mathbf{1}_{1\times d}\otimes\mathbf{I}_n)\mathrm{diag}(\mathrm{vec}(\mathbf{A})).$$

Therefore, the final dual problem of ridge regression is formulated as

$$\begin{aligned}\min_{\boldsymbol{\lambda}\in\mathbb{R}^{m\times d}}\ &\frac{1}{4}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d\otimes\mathbf{Q})\mathbf{\Omega}^{-1}(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\\ &+\frac{1}{2}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d\otimes\mathbf{Q})\mathbf{\Omega}^{-1}\mathbf{\Lambda}\mathbf{y}+\beta\left\|(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})\right\|_s,\end{aligned}$$

subject to

$$\|\boldsymbol{\lambda}_i\|_q\leq 1, 1\leq i\leq m.$$

Once the minimizer $\boldsymbol{\lambda}_*$ of (6) has been obtained, the minimizer of the primal problem (2) is obtained according to (5), i.e., $2\mathbf{\Omega}\mathrm{vec}(\mathbf{X}_*)-2\mathbf{\Lambda}^{\mathrm{T}}\mathbf{y}+(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}_*)=0$ in this case.

5 New Variant of ADMM for Solving (6)

In this section, we describe a novel ADMM method for efficiently solving the proposed formulation, i.e., (6).

5.1 Algorithmic Framework

The constraint set $\mathcal{C}$ of (6) is defined by

$$\mathcal{C}=\{\boldsymbol{\lambda}|\|\boldsymbol{\lambda}_i\|_q\leq 1, 1\leq i\leq m\}.$$

Similarly, a new notation $\mathbf{u}\in\mathbb{R}^{md\times 1}$ is defined by

$$\mathbf{u}:=(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}).$$

Thus, (6) is reformulated as

$$\min_{\boldsymbol{\lambda}\in\mathcal{C}}f^*(-(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda});\mathbf{A})+\beta\|\mathbf{u}\|_s$$

subject to

$$(\mathbf{I}_d\otimes\mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})-\mathbf{u}=0.$$

Denote $h(\boldsymbol{\lambda}) := f^*(-(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}); \mathbf{A})$. Recall that $g(\mathbf{u}) = \|\mathbf{u}\|_{1,p}$. Its augmented Lagrangian multiplier is

$$L(\boldsymbol{\lambda}, \mathbf{u}, \boldsymbol{\mu}) = h(\boldsymbol{\lambda}) + g(\mathbf{u}) + \boldsymbol{\mu}^{\mathrm{T}}\Big((\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}) - \mathbf{u}\Big) + \frac{\rho}{2}\Big\|(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}) - \mathbf{u}\Big\|_2^2.$$

Update $\boldsymbol{\lambda}$. The $(t+1)$-th update of $\boldsymbol{\lambda}$ is

$$\begin{aligned}\boldsymbol{\lambda}^{(t+1)} &= \underset{\boldsymbol{\lambda}\in\mathcal{C}}{\operatorname{argmin}} L(\boldsymbol{\lambda}, \mathbf{u}^{(t)}, \boldsymbol{\mu}^{(t)}) \\ &= \underset{\boldsymbol{\lambda}\in\mathcal{C}}{\operatorname{argmin}} h(\boldsymbol{\lambda}) + \big(\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q})\big)\boldsymbol{\mu}^{(t)} \\ &\quad + \frac{\rho}{2}\Big\|(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}) - \mathbf{u}^{(t)}\Big\|_2^2.\end{aligned} \tag{8}$$

It is worth noting that most of the previous studies [1], [8], [13], [16], [17] investigate the case of $p = q = 2$. However, we observe that the case of $p = 1$ and $q = \infty$ is more efficient than that of $p = q = 2$. If $p = 1$ and $q = \infty$, the update of $\boldsymbol{\lambda}$ can be naturally parallelized. We present the details in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109//TKDE.2019.2923239.

Algorithm 2. New Variant of ADMM for Solving (6)

Require: Data matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ and a positive β.
1: Initialize $\boldsymbol{\lambda}^{(0)}$ and $\boldsymbol{\mu}^{(0)}$, and set $t = 0$.
2: **for** Stopping criterion is not satisfied **do**
3: Update $\boldsymbol{\lambda}^{(t+1)}$ by solving the optimization problem (8).
4: Update $\mathbf{u}^{(t+1)}$ according to (9).
5: Update $\boldsymbol{\mu}^{(t+1)}$ according to (10).
6: $t = t + 1$;
7: **return** The final value of $\boldsymbol{\lambda}$.

Remark 1. The parallelization speed-up of the update of $\boldsymbol{\lambda}$ is up to $O(d)$ if $p = 1$ and $q = \infty$.

Update $\mathbf{u}$. The $(t+1)$-th update of $\mathbf{u}$ is formulated as follows:

Theorem 2.

$$\mathbf{u}^{(t+1)} = \boldsymbol{\omega}^{+} \odot \boldsymbol{\omega}, \tag{9}$$

where

$$\boldsymbol{\omega} := \frac{1}{\rho}\boldsymbol{\mu}^{(t)} + (\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}^{(t+1)}).$$

According to Theorem 2, the update of $\mathbf{u}$ has a closed form that can be computed very efficiently.

Update $\boldsymbol{\mu}$. The $(t+1)$-th update of $\boldsymbol{\mu}$ is simply

$$\boldsymbol{\mu}^{(t+1)} = \boldsymbol{\mu}^{(t)} + \rho\Big((\mathbf{I}_d \otimes \mathbf{Q})\mathrm{vec}(\boldsymbol{\lambda}^{(t+1)}) - \mathbf{u}^{(t+1)}\Big). \tag{10}$$

The new variant of ADMM is presented in Algorithm 2.

5.2 Complexity Analysis

If $f(\cdot)$ is convex, ADMM has been proven to be convergent [19]. A new norm $\|\mathbf{w}\|_{\mathbf{H}}$ is defined by

$$\|\mathbf{w}\|_{\mathbf{H}} := \mathbf{w}^{\mathrm{T}}\mathbf{H}\mathbf{w},$$

where $\mathbf{w} \in \mathbb{R}^{(nd+2md)}$ is defined by

$$\mathbf{w} := \big(\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda}); \mathbf{u}; \boldsymbol{\mu}\big),$$

and $\mathbf{H}$ is defined by

$$\mathbf{H} := \begin{pmatrix} \mathbf{0}_{nd\times nd} & & \\ & \rho\mathbf{I}_{md} & \\ & & \frac{1}{\rho}\mathbf{I}_{md.} \end{pmatrix},$$

Therefore, we obtain the following theorem [20]:

Theorem 3 (Theorem 5.1 in [20]). *Assume that $f(\cdot)$ is convex. If the ADMM is run for T iterations, we have*

$$\big\|\mathbf{w}^T - \mathbf{w}^{T+1}\big\|_{\mathbf{H}}^2 \le \frac{1}{T+1}\big\|\mathbf{w}^0 - \mathbf{w}_*\big\|_{\mathbf{H}}^2,$$

where $\mathbf{w}_* = \big(\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda}_*); \mathbf{u}_*; \boldsymbol{\mu}_*\big)$.

It means that the convergence rate of the ADMM method is $O(\frac{1}{T})$ [20]. Formally, the quality of the solution obtained by the ADMM method is presented as follows.

Remark 2. If we want to obtain an ϵ-accurate ($\epsilon > 0$) solution, i.e., $\big\|\mathbf{w}^T - \mathbf{w}^{T+1}\big\|_{\mathbf{H}}^2 \le \epsilon$, the ADMM method needs to perform $O(\frac{1}{\epsilon})$ iterations.

6 Theoretical Analysis

In this section, we analyze the accuracy of the approximate solution obtained by Algorithm 1 for two specific tasks: convex clustering and ridge regression.

6.1 Meta Framework for Analysis

We define the constraints of (6), i.e., $\mathcal{C}$ as

$$\mathcal{C} = \{\boldsymbol{\lambda} \in \mathbb{R}^{m\times d} | \|\boldsymbol{\lambda}_i\|_q \le 1,\ 1 \le i \le m\}.$$

Define the minimizer of (6) based on the evolving dataset $\mathbf{A} + \boldsymbol{\Delta}$ as

$$\begin{aligned}\tilde{\boldsymbol{\lambda}}_* := &\underset{\boldsymbol{\lambda}\in\mathcal{C}}{\operatorname{argmin}} f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q}); \mathbf{A} + \boldsymbol{\Delta}) \\ &+ \beta\|(\mathbf{I}_d \otimes \mathbf{Q})\mathrm{vec}(\boldsymbol{\lambda})\|_s.\end{aligned}$$

Recall that $\boldsymbol{\lambda}_*$ is defined by

$$\begin{aligned}\boldsymbol{\lambda}_* = &\underset{\boldsymbol{\lambda}\in\mathcal{C}}{\operatorname{argmin}} f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q}); \mathbf{A}) \\ &+ \beta\|(\mathbf{I}_d \otimes \mathbf{Q})\mathrm{vec}(\boldsymbol{\lambda})\|_s.\end{aligned}$$

According to Algorithm 1, we have

$$\begin{aligned}&f^*(-\mathrm{vec}^{\mathrm{T}}(\tilde{\boldsymbol{\lambda}}_*)(\mathbf{I}_d \otimes \mathbf{Q}); \mathbf{A} + \boldsymbol{\Delta}) \\ &\le f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda}_*)(\mathbf{I}_d \otimes \mathbf{Q}); \mathbf{A} + \boldsymbol{\Delta}) \\ &\le f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda}_*)(\mathbf{I}_d \otimes \mathbf{Q}); \mathbf{A}) + c.\end{aligned} \tag{11}$$

This inequality provides a method for bounding the difference between $\boldsymbol{\lambda}_*$ and $\tilde{\boldsymbol{\lambda}}_*$. Additionally, according to (5), we have

$$\begin{aligned}&\nabla f(\tilde{\mathbf{X}}_*; \mathbf{A} + \boldsymbol{\Delta}) - \nabla f(\mathbf{X}_*; \mathbf{A}) \\ &= (\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\big(\tilde{\boldsymbol{\lambda}}_* - \boldsymbol{\lambda}_*\big).\end{aligned} \tag{12}$$

This equality transforms the bound on the difference between $\tilde{\mathbf{X}}_*$ and $\mathbf{X}_*$ into the bound on the difference between $\boldsymbol{\lambda}_*$ and $\tilde{\boldsymbol{\lambda}}_*$. Therefore, by combining (11) and (12), we are able to bound the difference between $\tilde{\mathbf{X}}_*$ and $\mathbf{X}_*$.

Remark 3. The difference between $\tilde{\mathbf{X}}_*$ and $\mathbf{X}_*$ may be bounded by (11) and (12).

6.2 Guarantee of Model Accuracy for Convex Clustering

In convex clustering, $f(\mathbf{X};\mathbf{A}) = \|\mathbf{X}-\mathbf{A}\|_F^2$, $\nabla f(\mathbf{X};\mathbf{A}) = 2(\mathrm{vec}(\mathbf{X}) - \mathrm{vec}(\mathbf{A}))$, and $f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}};\mathbf{A}) = -\mathrm{vec}^{\mathrm{T}}(\mathbf{A})(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}) + \frac{1}{4}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q})(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda})$. The difference between $\tilde{\mathbf{X}}_*$ and $\mathbf{X}_*$ is bounded according to the following theorem:

Theorem 4. *When the data matrix $\mathbf{A}$ evolves to be $\mathbf{A}+\boldsymbol{\Delta}$, and Algorithm 1 is run to perform convex clustering, the difference between $\mathbf{X}_*$ obtained by Algorithm 1 based on $\mathbf{A}$ and the true solution $\tilde{\mathbf{X}}_*$ based on $\mathbf{A}+\boldsymbol{\Delta}$ is bounded as*

$$\begin{aligned}&\mathrm{vec}^{\mathrm{T}}(\mathbf{A})\big(\mathrm{vec}(\tilde{\mathbf{X}}_*) - \mathrm{vec}(\mathbf{X}_*)\big)\\ &\le \mathrm{vec}^{\mathrm{T}}(\mathbf{A})\mathrm{vec}(\boldsymbol{\Delta}) + \frac{c}{2}\\ &\quad + \frac{1}{2\beta}\|\mathrm{vec}(\boldsymbol{\Delta})\|\mathrm{vec}^{\mathrm{T}}(\mathbf{A}+\boldsymbol{\Delta})\mathrm{vec}(\mathbf{A}+\boldsymbol{\Delta}).\end{aligned}$$

6.3 Guarantee of Model Accuracy for Ridge Regression

In Section 4.2, we showed that $f(\mathbf{X};\mathbf{A}) = \mathrm{vec}^{\mathrm{T}}(\mathbf{X})\boldsymbol{\Omega}\mathrm{vec}(\mathbf{X}) - 2\mathbf{y}^{\mathrm{T}}\boldsymbol{\Lambda}\mathbf{X}$ and $f^*(-\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q});\mathbf{A}) = \frac{1}{4}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q})\boldsymbol{\Omega}^{-1}(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}) + \frac{1}{2}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q})\boldsymbol{\Omega}^{-1}\boldsymbol{\Lambda}\mathbf{y}$ for the ridge regression task. Define two new notations, $\tilde{\boldsymbol{\Omega}} \in \mathbb{R}^{(nd)\times(nd)}$ and $\tilde{\boldsymbol{\Lambda}} \in \mathbb{R}^{n\times(nd)}$, as

$$\tilde{\boldsymbol{\Omega}} := \mathrm{diag}(\mathrm{vec}(\mathbf{A}+\boldsymbol{\Delta}))\mathrm{diag}(\mathrm{vec}(\mathbf{A}+\boldsymbol{\Delta})) + \gamma\mathbf{I}_{nd},$$

and

$$\tilde{\boldsymbol{\Lambda}} := (\mathbf{1}_{1\times d} \otimes \mathbf{I}_n)\mathrm{diag}(\mathrm{vec}(\mathbf{A}+\boldsymbol{\Lambda})).$$

According to (11), we have

$$\begin{aligned}c \ge\ & \frac{1}{4}\mathrm{vec}^{\mathrm{T}}(\tilde{\boldsymbol{\lambda}}_*)(\mathbf{I}_d \otimes \mathbf{Q})\tilde{\boldsymbol{\Omega}}^{-1}(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\tilde{\boldsymbol{\lambda}}_*)\\ & - \frac{1}{4}\mathrm{vec}^{\mathrm{T}}(\boldsymbol{\lambda})(\mathbf{I}_d \otimes \mathbf{Q})\boldsymbol{\Omega}^{-1}(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}).\end{aligned} \tag{13}$$

Define a new notation $\boldsymbol{\Phi} \in \mathbb{R}^{(nd)\times(nd)}$ as

$$\boldsymbol{\Phi} := 2\mathrm{diag}(\mathrm{vec}(\boldsymbol{\Delta}))(\mathbf{1}_{d\times d} \otimes \mathbf{I}_n)\mathrm{diag}(\mathrm{vec}(\mathbf{A})).$$

Theorem 5. *When the data matrix $\mathbf{A}$ evolves to be $\mathbf{A}+\boldsymbol{\Delta}$, and Algorithm 1 is run to perform ridge regression, the difference between $\mathbf{X}_*$ obtained by Algorithm 1 based on $\mathbf{A}$ and the true solution $\tilde{\mathbf{X}}_*$ based on $\mathbf{A}+\boldsymbol{\Delta}$ is bounded as*

$$\begin{aligned}&\mathrm{vec}^{\mathrm{T}}(\tilde{\mathbf{X}}_*)\boldsymbol{\Omega}\mathrm{vec}(\tilde{\mathbf{X}}_*) - \mathrm{vec}^{\mathrm{T}}(\mathbf{X}_*)\boldsymbol{\Omega}\mathrm{vec}(\mathbf{X}_*)\\ &\le \frac{1}{16\beta^2}\left\|\mathbf{y}^{\mathrm{T}}\tilde{\boldsymbol{\Omega}}^{-1}\tilde{\boldsymbol{\Lambda}}\mathbf{y}\right\|^2\left\|\tilde{\boldsymbol{\Omega}}^{-1}\boldsymbol{\Phi}\tilde{\boldsymbol{\Omega}}^{-1}\right\|\\ &\quad + \frac{1}{16\beta^2}\left\|\mathbf{y}^{\mathrm{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{\Lambda}\mathbf{y}\right\|^2\left\|\boldsymbol{\Omega}^{-1}\boldsymbol{\Phi}\boldsymbol{\Omega}^{-1}\right\| + 4c.\end{aligned}$$

Remark 4. Given $\mathbf{A}$ and $\boldsymbol{\Delta}$, it is possible to obtain a relatively accurate solution for convex clustering and ridge regression by setting a small c and a large β in Algorithm 1.

7 Empirical Studies

In the section, we first present the experimental settings. Then, we perform convex clustering, obtain the cluster path for video datasets, and perform the ridge regression tasks to evaluate our method.

7.1 Experimental Settings

The experiments are performed on a server equipped with an Intel Xeon(R) 32-core E5-2610 CPU and 48 GB of RAM. We implement all the algorithms by using Matlab 2016b and the CVX solver [21].

We use six datasets: *moon*,[1] *iris*,[2] *segment*,[3] *svm-guide*,[4] *space_ga*,[5] and *airfoil*.[6] Given graph $\mathcal{G}$, the weight for every edge is set to be inversely proportional to the distance between the vertices; this approach is widely used in [8], [9], [22]. Additionally, our method is compared with many existing methods. These existing methods are presented briefly as follows:

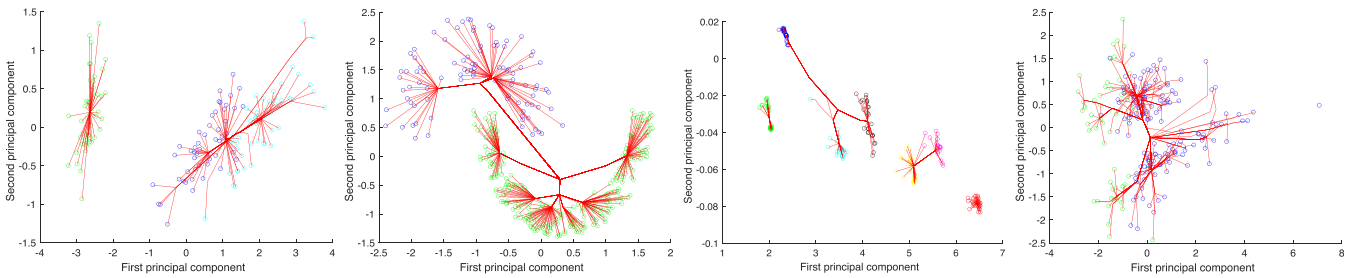
- PRIMAL-CC [13] has been proposed to perform convex clustering. It solves the primal problem, i.e., (2), directly.
- SSNAL-CC [17] is a semi-smooth Newton-based augmented Lagrangian method. It has been proposed to perform convex clustering efficiently.
- NET-LASSO [1] has been proposed as a general framework for performing SCO for various tasks, e.g., convex clustering and ridge regression.
- AMA-CC [8] is an alternating minimization algorithm. It has been proposed to perform convex clustering efficiently.

We follow the above notation, whereby the minimizer of (6) for $\mathbf{A}$ is denoted by $\mathbf{X}_*$, and the true minimizer of (6) for $\mathbf{A}+\boldsymbol{\Delta}$ is denoted by $\tilde{\mathbf{X}}_*$. Their difference, i.e., $\left\|\mathbf{X}_* - \tilde{\mathbf{X}}_*\right\|_F^2$, is used to measure the approximation of $\mathbf{X}_*$ against $\tilde{\mathbf{X}}_*$. Here, $\|\cdot\|_F$ represents the Frobenius norm. The threshold c in Algorithm 1 is set to be $c = 10$ by default.

7.2 Convex Clustering

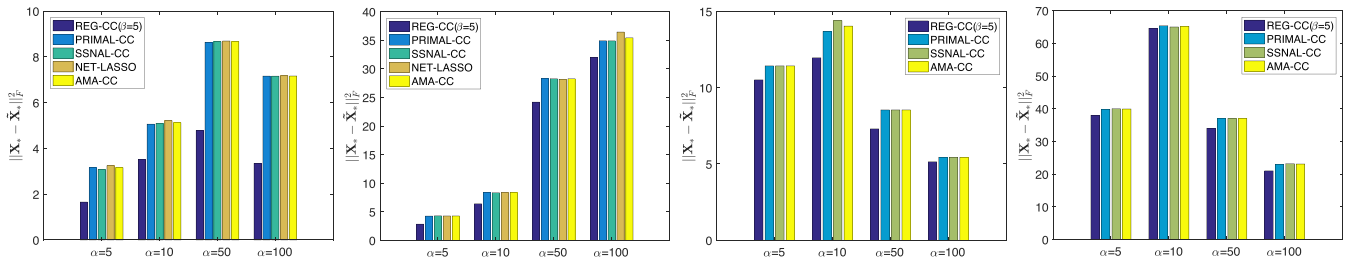
As we have shown in the previous section, $f(\mathbf{X};\mathbf{A}) = \|\mathbf{X}-\mathbf{A}\|_F^2$ holds for convex clustering. Given a value α, convex clustering recovers the clustering membership. If an instance contains more than 2 features, we present the cluster path by using its first and second principal components. The edges in graph $\mathcal{G}$ are generated by running the K-NN method with $K = 10$ for *moon* and *iris* and $K = 5$ for *segment* and *svm-guide*. For datasets *segment* and *svm-guide*, we pick

1. https://cs.joensuu.fi/sipu/datasets/jain.txt
2. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#iris
3. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#*segment*
4. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#svmguide1
5. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#space_ga
6. http://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise



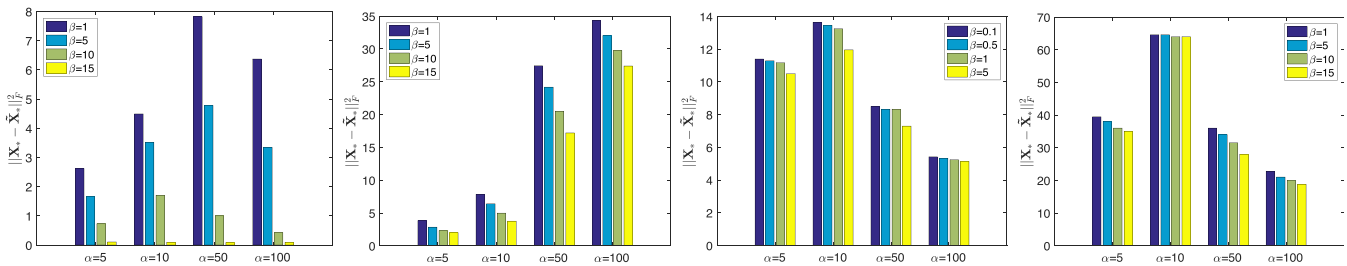
(a) *iris* (b) *moon* (c) *segment* (d) *svm-guide*

Fig. 4. Cluster paths obtained for the datasets *iris*, *moon*, *segment*, and *svm-guide*.



(a) *iris* (b) *moon* (c) *segment* (d) *svm-guide*

Fig. 5. When datasets evolve, our REG-CC method yields more accurate clustering results than those of its counterparts by varying α.



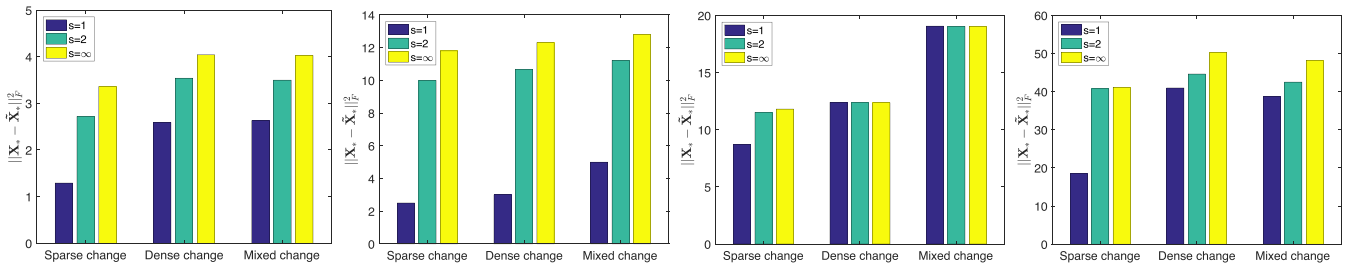
(a) *iris* (b) *moon* (c) *segment* (d) *svm-guide*

Fig. 6. When datasets evolve, our REG-CC method recovers the clustering membership more accurately with a large β.

200 instances randomly and draw their cluster paths as illustrative examples. We increase α gradually and obtain a cluster path for every dataset. Additionally, we use the Gaussian distribution $N(0, 0.1^2)$ to generate the perturbed matrix $\mathbf{\Delta}$ that is used to simulate the evolving dataset. Parameter β is set to 5 by default in the evaluation. Our method is denoted by *REG-CC* for convex clustering. Note that the existing NET-LASSO method is unable to handle the datasets *segment* and *svm-guide* due to running out of memory.

As illustrated in Fig. 4, every color represents a true cluster from the ground truth. If α is small, local communities of instances are detected. As α increases, different local communities are fused into a large community. The number of clusters declines for a large α. Furthermore, as illustrated in Fig. 5, our *REG-CC* method yields more accurate clustering results than do the existing methods for various values of α. Due to the regularized item in the dual problem of (6), our *REG-CC* method penalizes a large fluctuation in $\mathbf{X}_*$. Thus, with the control of accuracy, the regularized item makes $\mathbf{X}_*$ robust to a perturbation of the data matrix. In other words, $\mathbf{X}_*$ is a satisfactory approximation of $\hat{\mathbf{X}}_*$ for the evolving dataset $\mathbf{A} + \mathbf{\Delta}$. Additionally, when we vary β, Fig. 6 shows that a large β yields more accurate clustering results than does a small β. The reason is that a large β imposes a greater penalty on the fluctuation in $\mathbf{X}_*$, which makes $\mathbf{X}_*$ insensitive to the change in $\mathbf{A}$.

Additionally, we evaluate the performance of our method by varying s in the regularizer. We test it in three kinds of evolving environments. In the first evolving environment, the perturbed matrix $\mathbf{\Delta}$ is sparse, and its nonzero values are generated from a Gaussian distribution, i.e., $N(0, 0.1^2)$. This leads to a sparse change in the data matrix. In this case, the number of nonzero elements in the perturbed matrix is set to be $0.2n$, where n is the number of instances in the dataset. In the second evolving environment, the perturbed matrix $\mathbf{\Delta}$ is dense, and its values are generated from the Gaussian distribution $N(0, 0.01^2)$. This leads to a dense and relatively insignificant change in the data matrix. The third evolving environment generates the mixed perturbed matrix $\mathbf{\Delta}$ with $0.2n$ elements generated from the first case and the other elements generated from the second case. As illustrated in Fig. 7, our method performs best in the setting of $s = 1$ in all of these cases. In this setting, the regularizer favors obtaining a sparse $(\mathbf{I}_d \otimes \mathbf{Q})^{\mathrm{T}}\mathrm{vec}(\boldsymbol{\lambda}_*)$ in the evolving environment. Although the data matrix changes in the evolving environment, the sparsity makes the minimizer $\boldsymbol{\lambda}_*$ more robust to the change in the data matrix.



(a) *iris* (b) *moon* (c) *segment* (d) *svm-guide*

Fig. 7. When the environment evolves, our REG-CC method performs best at the setting of $s = 1$.

TABLE 2
Average CPU Time, in Seconds, of Running the Proposed REG-CC Method by Varying Constraints

dataset/algos	REG-CC(l_1)	REG-CC(l_2)	REG-CC(l_∞)	parallel REG-CC(l_∞)
iris	5.53(0.05)	24.83(0.81)	29.86(4.04)	**1.20**(**0.0004**)
moon	4.26(0.02)	33.25(0.85)	29.50(0.71)	**2.36**(**0.0137**)
segment	54.29(11.94)	106.05(4.60)	147.16(42.03)	**6.44**(**0.06**)
svm-guide	318.58(24.70)	731.83(38.28)	1209.35(5.44)	**20.32**(**0.04**)

The numbers in parentheses represent the respective variances.

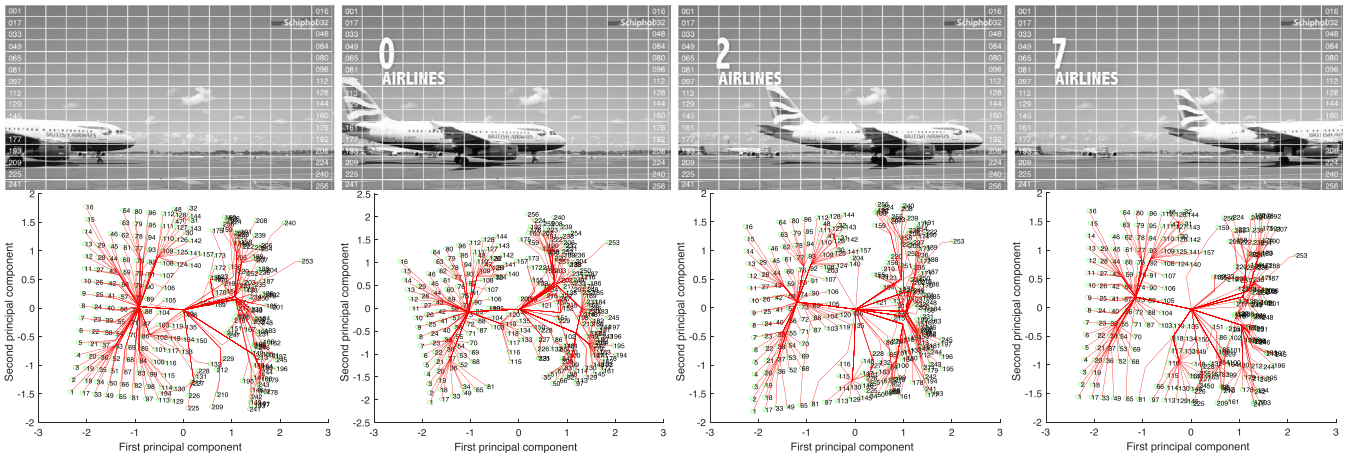


Fig. 8. Illustrative examples of cluster paths.

Finally, we test the efficiency of our method by varying q. Various values of q imply different types of constraints in the dual problem (6). Variable q is set to 1, 2 and ∞. In particular, if $q = \infty$, our method supports parallel computing naturally. It is implemented by using multiple cores. According to Fig. 7, performance at $s = 1$ is usually better than at $s = 2$ and at ∞. Thus, we set $s = 1$ in this experiment. For all settings, our algorithm is executed three times. We show the mean and the variance (shown as the number in parentheses) of the used CPU time in seconds in Table 2. As shown in Table 2, our method is most efficient at $q = \infty$, when it is executed in parallel.

7.3 Cluster Paths for Video Datasets

We obtain cluster paths for the DAVIS 2017 video dataset.[7] The videos in the dataset are at 480 p resolution. In the experiment, each raw image is transformed into the corresponding grayscale image, and we determine the cluster path by using the grayscale image. Specifically, we partition every image into 256 blocks according to the size of the image and assign every block a unique ID number. The number near a point in the cluster path represents the corresponding block in the grayscale image. Together with the position representing location in the image, every block is represented by a 3×1 vector. Additionally, we set the number of neighbors to $K = 4$ when K-NN is used to construct the sparse graph $\mathcal{G}$ for every image. Other parameters such as q and s are set to $q = 1$ and $s = \infty$ by default.

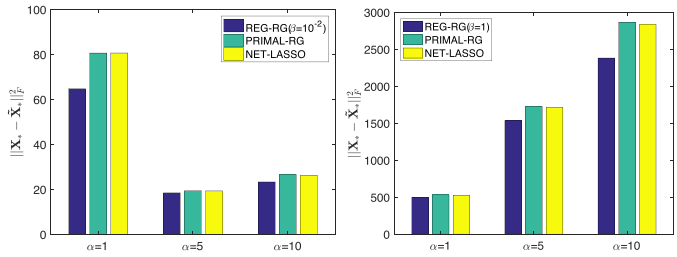
Fig. 8 shows the cluster paths. We obtain several interesting observations from cluster paths. At a low level, the cluster path shows the clustering membership among blocks of the grayscale image. For example, the blocks with id of 1, 17, and 33 (blocks in the top-left region) belong to a cluster due to their similar gray values. Blocks 112, 109, and 127 (those in the center-right region) belong to a cluster. This captures the similarity of the blocks at a low level. At a high level, we compare the four images and their corresponding

7. https://data.vision.ee.ethz.ch/csergi/share/davis/DAVIS-2017-test-challenge-480p.zip

Fig. 9. Illustrative examples of segmentation of moving objects by using convex clustering.

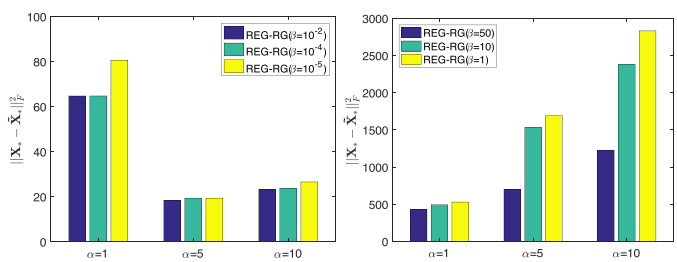
cluster paths. The event of the plane crossing is detected by analyzing the change of the structures of those cluster paths. In particular, the lower-right part of cluster paths changes from sparse paths to dense paths. The change in the cluster path reflects the event of the plane crossing. The above low-level and high-level observations provide significant insights for understanding the video.

Additionally, as we have shown in the previous section, if $f(\mathbf{X};\mathbf{A}) = \|\mathbf{X}-\mathbf{A}\|_F^2$ holds, our method can be used to perform object segmentation for a video. Similarly, we partition every image into 256 blocks. Every block is represented by its average gray value. The illustrative examples are presented in Fig. 9. As we observe, it is significant to detect the moving object in the video. Although clustering is a traditional method of performing image segmentation, convex clustering is more robust than the classic clustering method such as k-means clustering. The clustering result of convex clustering is determined and is not impacted by the seeds or heuristic rules used in k-means clustering.



(a) *space-ga* (b) *airfoil*

Fig. 10. When the datasets evolve, our REG-RG method yields more accurate prediction models than do its counterparts by varying α.



(a) *space-ga* (b) *airfoil*

Fig. 11. When the datasets evolve, our REG-RG method yields more accurate prediction models with a large β.

7.4 Ridge Regression

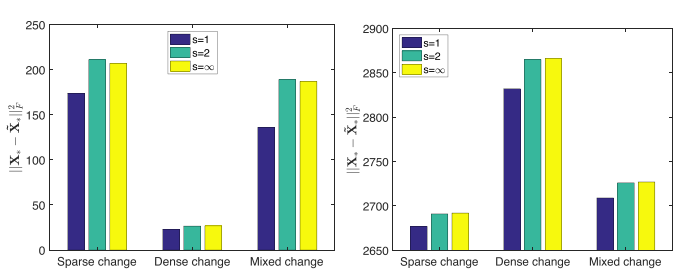
We further evaluate the performance of the proposed method by conducting the ridge regression task on the *space-ga* and *airfoil* datasets. As we have shown, $f(\mathbf{X};\mathbf{A})$ in (2) is instantiated to be

$$f(\mathbf{X};\mathbf{A}) = \sum_{i=1}^{n}\left(\left\|\mathbf{A}_i\mathbf{X}_i^{\mathrm{T}} - \mathbf{y}_i\right\|_2^2 + \gamma\|\mathbf{X}_i\|_2^2\right),$$

for the ridge regression task. Here, $\gamma = 5$ in the experiment, and we construct the network $\mathcal{G}$ by running the K-NN method on the chosen datasets with $K = 5$. Additionally, in this analysis, the proposed method is denoted by *REG-RG* for the ridge regression task. The state-of-the-art method is network lasso [1], denoted by *NET-LASSO*. The proposed method is also compared with the method of solving the primal problem (2) directly, which is denoted by *PRIMAL-RG*.

As illustrated in Fig. 10, the proposed REG-RG method yields more accurate prediction models than do the existing methods when datasets evolve. The superiority becomes significant with the increase in β, as shown in Fig. 11. The reason is that the proposed REG-RG method tends to yield the solution that is relatively robust to the evolving data. The large β means that REG-RG applies a larger penalty to the solution, which encourages it to be insensitive to the evolving data.

Next, we evaluate our method on three kinds of evolving data, including the sparse evolving data, the dense evolving data, and a mixture of both. As we have shown, the sparse



(a) *space-ga* (b) *airfoil*

Fig. 12. Our method yields the best solution at $s = 1$.

TABLE 3
Average CPU Time, in Seconds, Needed to Execute the Proposed REG-RG Method by Varying the Constraints

dataset/algos	REG-RG(l_1)	REG-RG(l_2)	REG-RG(l_∞)	parallel REG-RG(l_∞)
space-ga	26.41(0.145)	25.25(1.586)	26.63(0.225)	**9.43**(**0.34**)
airfoil	29.43(0.52)	29.61(0.004)	29.40(0.13)	**5.68**(**0.051**)

The numbers in the parentheses represent variance.

evolving data consist of $0.2n$ nonzero values generated from the Gaussian distribution $N(0, 0.1^2)$. The dense evolving data are generated from the Gaussian distribution $N(0, 0.01^2)$. A mixture of them is obtained by generating $0.2n$ values from the sparse case, while the other values are generated from the dense case. As shown in Fig. 12, it is more effective to obtain an approximate solution for the proposed method at $s = 1$ than at other values of s. Finally, we evaluate the efficiency of the proposed method by varying the constraints. We execute every algorithm three times and record the average and the variance (the number in the parentheses) of CPU time, in seconds. Table 3 shows that the efficiency varies slightly when different types of constraints are used. However, due to the use of parallel computing, the proposed method is most efficient at $q = \infty$.

8 Conclusions

We investigate SCO in an evolving environment. We first reformulate the problem into a convex problem with cone constraints in a dual space. Then, a new regularizer is proposed to obtain an approximate solution for the evolving dataset. A novel ADMM method is proposed to solve the problem efficiently. Afterward, we analyze the quality of the solution theoretically for the cases of the convex clustering and ridge regression tasks. Finally, extensive empirical studies show the advantages of the proposed method.

Acknowledgments

This work was supported by the National Key R & D Program of China (Grant No. 2018YFB1003203), the National Natural Science Foundation of China (Grant Nos. 61672528, 61773392, 61701451, 61671463, and 61772544), and the National Basic Research Program (the 973 program) under Grant No. 2014CB347800.

References

[1] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *Proc. ACM Proc. Int. Conf. Knowl. Discovery Data Mining*, 2015, Art. no. 387.
[2] Y. Zhao, K. Xu, L. Xinwang, E. Zhu, X. Zhu, and J. Yin, "Triangle lasso for simultaneous clustering and optimization in graph datasets," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2018.2865342.
[3] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[4] L. Bottou, Y. Bengio, et al., "Convergence properties of the k-means algorithms," in *Proc. Advances Neural Inf. Process. Syst.*, 1995, pp. 585–592.
[5] C. Tang, X. Zhu, X. Liu, M. Li, P. Wang, C. Zhang, and L. Wang, "Learning joint affinity graph for multi-view subspace clustering," *IEEE Trans. Multimedia*, vol. 21, no. 7, pp. 1724–1736, Jul. 2019.
[6] X. Liu, X. Zhu, M. Li, L. Wang, C. Tang, J. Yin, D. Shen, H. Wang, and W. Gao, "Late fusion incomplete multi-view clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2018.2879108.
[7] S. Ghosh, K. Page, and D. D. Roure, "An application of network lasso optimization for ride sharing prediction," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1–21.
[8] E. C. Chi and K. Lange, "Splitting methods for convex clustering," *Prof. Geographer*, vol. 46, no. 1, pp. 80–89, 2014.
[9] K. M. Tan and D. Witten, "Statistical properties of convex clustering," *Electron. J. Statist.*, vol. 9, no. 2, 2015, Art. no. 2324.
[10] P. Radchenko and G. Mukherjee, "Convex clustering via l1 fusion penalization," *J. Roy. Statistical Soc.: Series B (Statistical Methodology)*, vol. 47, Feb. 2017, Art. no. 67.
[11] J. Leskovec and R. Sosi, "Snap: A general purpose network analysis and graph mining library," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 1, 2016, Art. no. 1.
[12] A. Jung, N. Tran, and A. Mara, "When is network lasso accurate?" Frontiers in Applied Mathematics and Statistics, vol. 3, pp. 1–11, 2018.
[13] F. Lindsten, H. Ohlsson, and L. Ljung, "Just relax and come clustering! a convexification of k-means clustering," Tech. Rep. LiTH-ISY-R-2992, Dept. Elect. Eng., Linkping University, Linkping, Sweden, 2011.
[14] Y. Zhao, Y. Ming, X. Liu, E. Zhu, K. Zhao, and J. Yin, "Large-scale k-means clustering via variance reduction," *Neurocomputing*, vol. 307, pp. 184–194, 2018.
[15] C. Zhu, H. Xu, C. Leng, and S. Yan, "Convex optimization procedure for clustering - theoretical revisit," in *Proc. Advances Neural Inf. Process. Syst.*, 2014, pp. 1619–1627.
[16] A. Panahi, D. Dubhashi, F. D. Johansson, and C. Bhattacharyya, "Clustering by sum of norms: Stochastic incremental algorithm, convergence and cluster recovery," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2769–2777.
[17] Y. Yuan, D. Sun, and K. C. Toh, "An efficient semismooth newton based algorithm for convex clustering," in *Proc. Advances Neural Inf. Process. Syst.*, 2018, pp. 5718–5726.
[18] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.
[20] B. He and X. Yuan, "On non-ergodic convergence rate of Douglas-Rachford alternating direction method of multipliers," *Numerische Mathematik*, vol. 130, no. 3, pp. 567–577, 2015.
[21] "CVX research project," [Online]. Available: http://cvxr.com/cvx/, Accessed: Jul. 1, 2018.
[22] G. K. Chen, E. C. Chi, J. M. O. Ranola, and K. Lange, "Convex clustering - an attractive alternative to hierarchical clustering," *PLoS Comput. Biol.*, vol. 11, no. 5, 2015, Art. no. e1004228.

Yawei Zhao received the BE and MS degrees in computer science from the National University of Defense Technology, China, in 2013 and 2015, respectively. He is currently working toward the PhD degree in computer science at the National University of Defense Technology, China. His research interests include asynchronous and parallel optimization algorithms, pattern recognition, and machine learning.

En Zhu received the MS and PhD degrees in computer science from the National University of Defense Technology, China, in 2001 and 2005, respectively. He currently works as a full professor with the School of Computer Science, National University of Defense Technology, China. His main research interests include pattern recognition, image processing, and information security.

Xinwang Liu received the PhD degree from the National University of Defense Technology (NUDT), China. He is currently an assistant researcher with the School of Computer Science, NUDT. His current research interests include kernel learning and unsupervised feature learning. He has published more than 40 peer-reviewed papers, including those in highly regarded journals and conferences, such as the *IEEE Transactions on Image Processing*, the *IEEE Transactions on Neural Networks and Learning Systems*, ICCV, AAAI, IJCAI, etc. He served on the technical program committees of IJCAI 2016-2017 and AAAI 2018-2018.

Chang Tang received the PhD degree from Tianjin University, Tianjin, China, in 2016. He joined the AMRL Lab at the University of Wollongong between September 2014 and September 2015. He is currently an associate professor with the School of Computer Science, China University of Geosciences, Wuhan, China. He has published more than 20 peer-reviewed papers, including those in highly regarded journals and conferences, such as the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, the *IEEE Transactions on Multimedia*, the *IEEE Transactions on Human-Machine Systems*, the *IEEE Signal Processing Letters*, ICCV, CVPR, AAAI, ACMMM, etc. He served on the technical program committees of IJCAI 2018/2019, ICME 2018/2019, AAAI 2019, CVPR 2019, and ICCV 2019. His current research interests focus on building machine learning models for solving computer vision and data mining problems.

Deke Guo received the BS degree in industry engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the PhD degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a professor with the College of Systems Engineering, National University of Defense Technology. His research interests include distributed systems, software-defined networking, datacenter networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE and a member of the ACM.

Jianping Yin received the MS and PhD degrees in computer science from the National University of Defense Technology, China, in 1986 and 1990, respectively. He is a professor of computer science at the Dongguan University of Technology. His research interests involve artificial intelligence, pattern recognition, algorithm design, and information security.