On the Cluster-aware Supervised Learning (CluSL): Frameworks, Convergent Algorithms, and Applications

Shutong Chen
School of Business and Management, Donghua University, No. 1882 West Yanan Road, Shanghai, China 200051, cst950928@hotmail.com.

Weijun Xie
Department of Industrial & Systems Engineering, Virginia Tech, Blacksburg, VA 24061, wxie@vt.edu.

This paper proposes a cluster-aware supervised learning (CluSL) framework, which integrates the clustering analysis with supervised learning (SL). The objective of CluSL is to simultaneously find the best clusters of the data points and minimize the sum of loss functions within each cluster. This framework has many potential applications in healthcare, operations management, manufacturing, and so on. Since CluSL, in general, is nonconvex, we develop a regularized alternating minimization (RAM) algorithm to solve it, where at each iteration, we penalize the distance between the current clustering solution and the one from the previous iteration. By choosing a proper penalty function, we show that each iteration of the RAM algorithm can be computed efficiently. We further prove that the proposed RAM algorithm will always converge to a stationary point within a finite number of iterations. This is the first known convergence result in cluster-aware learning literature. Furthermore, we extend CluSL to the high-dimensional datasets, termed F-CluSL framework. In F-CluSL, we cluster features and minimize loss function at the same time. Similarly, to solve F-CluSL, a variant of RAM algorithm (i.e., F-RAM) is developed and proven to be convergent to an ϵ-stationary point. Our numerical studies demonstrate that the proposed CluSL and F-CluSL can outperform the existing ones such as random forests and support vector classification, both in the interpretability of learning results and prediction accuracy.

Key words: Clustering, Supervised Learning, Regularization, Alternating Minimization, Globally Convergent, Feature Extraction.

1. Introduction

In the wake of big data, Supervised Learning (SL) techniques have been broadly applied to analyze the possibly large-scale datasets as well as provide meaningful information for decision making (Duda et al. 2012, Choi et al. 2016). The goal of SL is to learn the hypothesis or describe the interrelationship between input variables and output responses using a set of training samples with known responses. Two typical examples of SL are regression and classification problems, corresponding to continuous and discrete responses, respectively. For example, in healthcare systems, there are enormous patient data, including personal information, treatment plan, required drugs, etc., are incorporated to classify patients for the different discrete levels of care upon arrival with corresponding medical resources allocation. It is important for the nurses to effectively assign each patient the right level of care to maximize the healthcare resources. In the real estate industry, housing data such as the geographic location, square feet, number of bedrooms, built year, historical sales prices, etc., are useful to develop the learning models to predict the future sales prices, which is a regression problem. In many SL problems, it often assumes that the data points are homogeneous, which may not hold in practice. In the aforementioned examples, it is very natural that the patients might belong to different groups, based upon their family histories or their physical conditions such as age; or in the housing pricing example, the housing pricing can vary significantly according to the different school zones or the different neighborhoods. Ignoring such heterogeneity within datasets can cause very poor learning results (please see Section 5 for illustrative examples). Therefore, to improve the learning results and explore the cluster structures within the dataset, this paper studies Cluster-aware Supervised Learning (CluSL), which simultaneously identifies the disparity among the different groups and trains the learning models.

1.1. Relevant Literature

Different from conventional SL, this paper explores the hidden structures in the datasets to enhance SL. Below we review the most relevant works, reveal existing research gaps, and further distinguish our work.

Cluster-wise linear regression. The cluster-wise linear regression (CLR) is to find clusters based on linear regression, where the data points having similar regression effects belong to the same cluster. It turns out that CLR can be viewed as a special case of CluSL, which is illustrated in Section 2. Due to its appealing prediction accuracy compared with regular linear regression (Park et al. 2017), many variants of CLR have been proposed in literature such as Zhu et al. (2012), Liu et al. (2017), Devijver (2017), Khadka et al. (2018), etc. For example, Zhu et al. (2012) proposed a mixed-integer program with the least sum of absolute deviations to model CLR. A likelihood-based CLR was proposed to determine the potential sets of relevant predictors in Devijver (2017). Similar work can be found in (Di Mari et al. 2017, Khadka et al. 2018). However, merely focusing on training errors, these CLRs do not explore inherent clustering structures, and then might result in dramatic prediction errors by ignoring the clustering structures of the incoming data. This paper fills this gap, and the proposed CluSL allows to both explore interpretable clusters and deliver accurate learning results. Several efforts have been made in solving CLRs. The earlier works date back to Späth (1982), which interchanges data samples between clusters until the objective function cannot be improved. Recently, Bagirov et al. (2015) developed an incremental algorithm, where the hyperbolic smoothing technique is used to approximate the non-smooth objective function of CLR. Besides, some works attempted to design algorithms to be less sensitive to initial clustering solutions. For example, Zhang (2003) replaced the hard partition constraints by a soft membership function. Zhao et al. (2018) introduced the generalized mountain method to determine the number of clusters and initial clustering solutions. Although numerical effectiveness was displayed, most proposed algorithms failed to converge to any stationary point. In this paper, we propose a regularized alternating minimization (RAM) to solve CluSL and show that it is convergent to the stationary point within a finite number of iterations without making any additional assumption.

Clustering first, then classification. Several recent works focused on improving classification results using clustering techniques (Arumugam and Christy 2018, Gallego et al. 2018, Khabia 2014, Wang et al. 2019). Arumugam and Christy (2018) implemented clusterwise principle component analysis embedded with a self-organizing map algorithm to improve classification results. Gallego et al. (2018) first obtained feature clustering by the approximated similarity search, and then used deep neural networks for classification. In Li et al. (2004), Zhang et al. (2006), Zhang et al. (2008), Tran et al. (2018), clustering was used to reduce the number of instances of classification. By incorporating clustering into the classification, it helps discover the inherent data structures and thus can improve classification results in some cases. However, as far as the authors are concerned, all the existing works simply treat clustering as a pre-processing without involving in classification, while the inexact clustering results can cause misleading learning results. Different from these works, our CluSL framework is more flexible, and simultaneously explores the discriminative information within the datasets and also optimizes the learning results.

1.2. Summary of Contributions

This paper studies the modeling properties of the CluSL framework, develops effective algorithm, and studies their extensions to feature cluster-aware supervised learning (denoted as F-CluSL). In particular, the main contributions of this paper are summarized below:

(i) We derive the modeling properties of the CluSL framework and show that the SL-based clustering and clustering-first-then-SL can be viewed as special cases.

(ii) We propose a regularized alternating minimization (RAM) algorithm to solve the CluSL framework and prove that it is always convergent to a stationary point without making any additional assumption, providing the first known convergent algorithm in the cluster-aware learning literature.

(iii) For the high-dimensional data, we extend CluSL to the F-CluSL framework. To solve the F-CluSL, a modification of the RAM algorithm, F-RAM is proposed and is proven to be convergent to an approximate stationary point.

(iv) We numerically evaluate the proposed CluSL, F-CluSL frameworks, and their corresponding solution algorithms on hypothetical and real-world datasets. Our numerical results demonstrate that the proposed frameworks can be superior both in interpretability and prediction accuracy compared to existing methods.

The remainder of the paper is organized as follows. Section 2 introduces the setting of CluSL framework, different choices of loss and dissimilarity functions as well as the model properties. Section 3 presents the RAM algorithm and its convergence analysis. Section 4 postulates the extension framework of CluSL, termed F-CluSL and its solution method, F-RAM. Section 5 presents the numerical studies. Finally, Section 6 concludes the paper.

Notation: The following notation is used throughout the paper. We use bold-letters (e.g., $\boldsymbol{x}, \boldsymbol{A}$) to denote vectors or matrices, and use corresponding non-bold letters to denote their components. We let $\boldsymbol{e}$ be the all-ones vector, and let $\boldsymbol{e}_i$ be the ith standard basis vector.

Given an integer n, we let $[n] := \{1, 2, \dots, n\}$, and use $\mathbb{R}^n_+ := \{\boldsymbol{x} \in \mathbb{R}^n : x_i \geq 0, \forall i \in [n]\}$ and $\mathbb{R}^n_{++} := \{\boldsymbol{x} \in \mathbb{R}^n : x_i > 0, \forall i \in [n]\}$. Given a real number t, we let $(t)_+ := \max\{t, 0\}$, $\lceil t \rceil$ be its round-up and $\lfloor t \rfloor$ be its round-down. Given a finite set I, we let $|I|$ denote its cardinality. Given a matrix $\boldsymbol{A}$, we let $\boldsymbol{A}_i$ denote the vector of its ith row and $\boldsymbol{A}_{\cdot j}$ denote the vector of its jth column. Additional notation will be introduced as needed.

2. Model Formulation

2.1. Setting

This paper considers a dataset of n training samples $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$, where each sample $(\boldsymbol{x}_i, y_i)$ has response $y_i \in \mathbb{R}$ and data point $\boldsymbol{x}_i$ with p features, i.e. $\boldsymbol{x}_i \in \mathbb{R}^p$. Suppose that all samples belong to K clusters, and within each cluster $k \in [K] := \{1, \dots, K\}$, the responses can be described by the data points. Hence, we propose the following cluster-aware supervised learning (CluSL) framework:

$$v_\rho := \min_{\boldsymbol{\theta}, \boldsymbol{\delta}, \boldsymbol{m}} R_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m}) := \sum_{i \in [n]} \sum_{k \in [K]} \delta_{ik} \ell_k(y_i, \boldsymbol{x}_i, \boldsymbol{\theta}_k) + \rho \sum_{i \in [n]} \sum_{k \in [K]} \delta_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}_k\right), \tag{1a}$$

$$\text{s.t.} \quad \sum_{k \in [K]} \delta_{ik} = 1, \forall i \in [n], \tag{1b}$$

$$\sum_{i \in [n]} \delta_{ik} \geq 1, \forall k \in [K], \tag{1c}$$

$$\delta_{ik} \in \{0, 1\}, \forall k \in [K], \forall i \in [n]. \tag{1d}$$

Above, ρ is a non-negative tuning parameter, which controls the overlapping areas among different clusters. The binary variable δ_{ik} denotes the cluster assignment decision, i.e. $\delta_{ik} = 1$ implies that the data point $\boldsymbol{x}_i$ belongs to cluster k, otherwise, $\delta_{ik} = 0$, for all $i \in [n], k \in [K]$. In each cluster $k \in [K]$, we let $\ell_k : \mathbb{R} \times \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}_+$ denote the loss function with estimator $\boldsymbol{\theta}_k \in \mathbb{R}^d$, and $S_k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}_+$ denote the dissimilarity function, which measures the dissimilarity between a data point $\boldsymbol{x}$ and the centroid of the cluster $\boldsymbol{m}_k$ (i.e., $S_k(\boldsymbol{x}, \boldsymbol{m}_k)$ becomes smaller if $\boldsymbol{x}$ and $\boldsymbol{m}_k$ are more similar to each other). The goal of CluSL (1) is to simultaneously divide the training samples into K disjoint clusters and minimize the sum of loss functions within each cluster. Note that the CluSL framework (1) can also be viewed as the mixture of experts (Jacobs et al. 1991), where there are K experts and δ_{ik}=1 represents that the expert k is responsible for data point $\boldsymbol{x}_i$.

Note that (i) loss functions can be very general, for example, they can even be convolutional neural networks (CNNs); (ii) the centroids $\{\boldsymbol{m}_k\}_{k \in [K]}$ are essential for the testing and prediction purpose, i.e., given a testing sample, we first compute its similarities with all the centroids, then assign it to the cluster with the smallest dissimilarity value, finally predict its value using the learning results of the assigned cluster; (iii) when the loss functions are all zeros, CluSL becomes conventional k-means clustering problems. Thus, the developed solution algorithms in this paper also work for the k-means clustering problems; and (iv) this framework essentially integrates unsupervised learning (UL) and supervised one, delivering interpretable learning results and increasing the prediction accuracy. Finally, for the notational convenience, we let set Δ denote the feasible region of CluSL (1), i.e., $\Delta = \{\boldsymbol{\delta} : (1b) - (1d)\}$.

2.2. Loss and Dissimilarity Functions

In this subsection, we list different types of loss and dissimilarity functions, which can be applied to CluSL framework.

Loss functions. Below are several widely used loss functions.

(a) (Negative log-likelihood loss function for linear regression, see in Hasselblad 1966) Suppose that the response variable y is linear in explanatory variables $\boldsymbol{x} \in \mathbb{R}^p$ and subject to Gaussian noise, i.e., $y = \boldsymbol{\beta}^\top \boldsymbol{x} + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian random variable with mean zero and variance σ^2. Then the negative log-likelihood loss function for linear regression is formulated as below:

$$\ell^1\left(y, \boldsymbol{x}, (\boldsymbol{\beta}, \sigma)\right) := \log(\sqrt{2\pi}\sigma) + \frac{(y - \boldsymbol{\beta}^\top \boldsymbol{x})^2}{2\sigma^2}.$$

Here, the estimator $\boldsymbol{\theta}_k$ in CluSL (1) is $(\boldsymbol{\beta}, \sigma)$ within each cluster $k \in [K]$. In practice, the standard deviation often has a lower bound $\underline{\sigma}$ (Day 1969).

(b) (Logistic loss function, see in Czepiel 2019) Logistic regression is a widely-used binary classification method. Suppose that the binary response variable $y \in \{0, 1\}$ is linear in explanatory variables $\boldsymbol{x} \in \mathbb{R}^p$ and subject to logistic error, i.e., $y = \boldsymbol{\theta}^\top \boldsymbol{x} + \epsilon$, $\epsilon \sim \mathrm{Logistic}(0, 1)$ is the standard logisitic random variable. Then the negative log-likelihood loss function for logistic regression is formulated as below:

$$\ell^2\left(y, \boldsymbol{x}, \boldsymbol{\theta}\right) := y \log(1 + \exp^{-\boldsymbol{\theta}^\top \boldsymbol{x}}) - (1 - y) \log\left(1 - \frac{1}{1 + \exp^{-\boldsymbol{\theta}^\top \boldsymbol{x}}}\right).$$

(c) (Multi-class hinge loss function, see in Weston and Watkins 1999) Hinge loss is popular in support vector machines (SVMs). Given a sample $(\boldsymbol{x}, y)$, suppose that the response y is an integer, i.e., $y \in [J] := \{1, \ldots, J\}$ for some positive integer J, and linear classifiers $\{\boldsymbol{\theta}_j^\top \boldsymbol{x}\}_{j\in[J]}$. Then the hinge loss function introduced by Weston and Watkins (1999) has the following form:

$$\ell^3(y,\boldsymbol{x},\boldsymbol{\theta}) := \sum_{j\in[J],j\neq y} \max\left(0, \boldsymbol{\theta}_j^\top \boldsymbol{x} - \boldsymbol{\theta}_y^\top \boldsymbol{x} + 1\right).$$

Here, the estimator $\boldsymbol{\theta}_k$ in CluSL (1) is $(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_J)$ within each cluster $k\in[K]$.

Dissimilarity functions. In clustering analysis, there are many different choices of dissimilarity distance functions (Rokach and Maimon 2005). Below are some examples.

(a) Squared Euclidean distance-based- $S^1(\boldsymbol{x},\boldsymbol{m}) := \|\boldsymbol{x}-\boldsymbol{m}\|^2$, which has been used in image classification (Mary and Dharma 2017, Stokes and Deane 2009), outlier detection (Rosso 2014), climate classification (Aliaga et al. 2017), etc.

(b) Cosine distance-based- $S^2(\boldsymbol{x},\boldsymbol{m}) := 1 - \frac{\boldsymbol{x}^\top \boldsymbol{m}}{\|\boldsymbol{x}\|\|\boldsymbol{m}\|}$, which has been used in attribute weight assignment (Suo et al. 2018), image segmentation (Panigrahi et al. 2019), face recognition (Ren et al. 2018).

(c) Gaussian kernel distance- $S^3(\boldsymbol{x},\boldsymbol{m}) := 1 - \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{m}\|^2}{2\varsigma^2}\right)$, where ς^2 is the given variance. Gaussian kernel distance has been used in speech recognition (Kumar et al. 2015, Lei Lei 2016).

2.3. Model Properties

In this subsection, we show that the CluSL (1) can be quite flexible, and by choosing a proper tuning parameter, it can be (i) SL-based clustering, e.g., cluster-wise regression, and (ii) clustering-first-then-SL, e.g., clustering first, then classification.

To begin with, we make the following observation.

Observation 1 *If $\rho = 0$, then CluSL* (1) *reduces to SL-based clustering.*

This implies that if the clustering is the main learning task, then we can use SL to improve the clustering results by simply choosing $\rho = 0$ in the CluSL (1).

If ρ is large enough or more precisely, there exists a threshold $\bar{\rho}$ such that $\rho \geq \bar{\rho}$, then the CluSL (1) becomes clustering-first-then-SL, namely,

$$\min_{\boldsymbol{\theta}} \sum_{i\in[n]} \sum_{k\in[K]} \delta_{ik}^* \ell_k(y_i,\boldsymbol{x}_i,\boldsymbol{\theta}_k) \tag{2a}$$

$$\text{s.t.} \quad (\boldsymbol{\delta}^*,\boldsymbol{m}^*) \in \arg\min_{\boldsymbol{\delta}\in\Delta,\boldsymbol{m}} \sum_{i\in[n]} \sum_{k\in[K]} \delta_{ik} S_k\left(\boldsymbol{x}_i,\boldsymbol{m}_k\right). \tag{2b}$$

This result is summarized below.

Proposition 1 *There exists a positive $\bar{\rho} > 0$ such that for any $\rho \geq \bar{\rho}$, the CluSL* (1) *is equivalent to clustering-first-then-SL problem* (2). *In particular, $\bar{\rho}$ can be chosen as $\bar{\rho} = \frac{\widehat{v}}{\widehat{s}}$, where $\widehat{v} \geq 0$ and $\widehat{s} > 0$ are defined as below:*

$$\widehat{s} := \min_{\boldsymbol{\delta} \in \Delta} \left\{ \min_{\boldsymbol{m}} \sum_{i \in [n]} \sum_{k \in [K]} \delta_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}_k\right) - \sum_{i \in [n]} \sum_{k \in [K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right) > 0 \right\},$$

$$\widehat{v} := \max_{\boldsymbol{\delta} \in \Delta} \left\{ \min_{\boldsymbol{\theta}} \sum_{i \in [n]} \sum_{k \in [K]} \delta^*_{ik} \ell_k(y_i, \boldsymbol{x}_i, \boldsymbol{\theta}_k) - \min_{\boldsymbol{\theta}} \sum_{i \in [n]} \sum_{k \in [K]} \delta_{ik} \ell_k(y_i, \boldsymbol{x}_i, \boldsymbol{\theta}_k) \geq 0 \right\}.$$

where $(\boldsymbol{\delta}^*, \boldsymbol{m}^*)$ is defined in (2b).

Proof: For any $\rho > 0$, let $(\widehat{\boldsymbol{\delta}}_\rho, \widehat{\boldsymbol{\theta}}_\rho, \widehat{\boldsymbol{m}}_\rho)$ and $(\boldsymbol{\delta}^*, \boldsymbol{\theta}^*, \boldsymbol{m}^*)$ denote optimal solutions of CluSL (1) and clustering-first-then-SL problem (2), respectively; and let $\widehat{v}_\rho, v^*$ denote their corresponding optimal objective values. Then, to prove Proposition 1, it is equivalent to show that $\widehat{v}_\rho = v^* + \rho \sum_{i \in [n]} \sum_{k \in [K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right)$ for any $\rho \geq \bar{\rho}$, i.e., $(\boldsymbol{\delta}^*, \boldsymbol{\theta}^*, \boldsymbol{m}^*)$ is optimal to CluSL (1).

Since $(\boldsymbol{\delta}^*, \boldsymbol{\theta}^*, \boldsymbol{m}^*)$ is feasible to CluSL (1), thus we have

$$v^* + \rho \sum_{i \in [n]} \sum_{k \in [K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right) \geq \widehat{v}_\rho. \tag{3}$$

Thus, it remains to show that $v^* + \rho \sum_{i \in [n]} \sum_{k \in [K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right) \leq \widehat{v}_\rho$ when $\rho \geq \bar{\rho}, \bar{\rho} > 0$. We prove it by contradiction. Suppose

$$v^* + \rho \sum_{i \in [n]} \sum_{k \in [K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right) > \widehat{v}_\rho,$$

which implies that $(\boldsymbol{\delta}^*, \boldsymbol{\theta}^*, \boldsymbol{m}^*)$ is not optimal to CluSL (1), or equivalently, $(\widehat{\boldsymbol{\delta}}_\rho, \widehat{\boldsymbol{\theta}}_\rho, \widehat{\boldsymbol{m}}_\rho)$ is not optimal to clustering-first-then-SL problem (2). Hence, due to optimality condition of $(\boldsymbol{\delta}^*, \boldsymbol{\theta}^*, \boldsymbol{m}^*)$, we must have

$$\sum_{i \in [n]} \sum_{k \in [K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right) < \sum_{i \in [n]} \sum_{k \in [K]} \widehat{\boldsymbol{\delta}}_{ik\rho} S_k\left(\boldsymbol{x}_i, \widehat{\boldsymbol{m}}_{k\rho}\right). \tag{4}$$

Additionally,

$$\sum_{i \in [n]} \sum_{k \in [K]} \widehat{\boldsymbol{\delta}}_{ik} \ell_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{\theta}}_k) = \widehat{v}_\rho - \rho \sum_{i \in [n]} \sum_{k \in [K]} \widehat{\boldsymbol{\delta}}_{ik\rho} S_k\left(\boldsymbol{x}_i, \widehat{\boldsymbol{m}}_{k\rho}\right)$$

$$
\begin{aligned}
&< v^* + \rho \sum_{i\in[n]} \sum_{k\in[K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right) - \rho \sum_{i\in[n]} \sum_{k\in[K]} \widehat{\boldsymbol{\delta}}_{ik\rho} S_k\left(\boldsymbol{x}_i, \widehat{\boldsymbol{m}}_{k\rho}\right) \\
&< v^* := \sum_{i\in[n]} \sum_{k\in[K]} \delta^*_{ik} \ell_k(y_i, \boldsymbol{x}_i, \boldsymbol{\theta}^*_k), \qquad (5)
\end{aligned}
$$

where the first inequality is due to our assumption and the second one is because of (4).

Then according to (4) and (5) as well as the definition of $\widehat{s}$ and $\widehat{v}$, in this case, we must $\widehat{s}, \widehat{v} > 0$. Hence, we can further conclude that

$$
\sum_{i\in[n]} \sum_{k\in[K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right) + \widehat{s} \leq \sum_{i\in[n]} \sum_{k\in[K]} \widehat{\boldsymbol{\delta}}_{ik\rho} S_k\left(\boldsymbol{x}_i, \widehat{\boldsymbol{m}}_{k\rho}\right), \qquad (6)
$$

$$
\sum_{i\in[n]} \sum_{k\in[K]} \widehat{\boldsymbol{\delta}}_{ik} \ell_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{\theta}}_k) \geq \min_{\boldsymbol{\theta}} \sum_{i\in[n]} \sum_{k\in[K]} \widehat{\boldsymbol{\delta}}_{ik} \ell_k(y_i, \boldsymbol{x}_i, \boldsymbol{\theta}_k) \geq \sum_{i\in[n]} \sum_{k\in[K]} \delta^*_{ik} \ell_k(y_i, \boldsymbol{x}_i, \boldsymbol{\theta}^*_k) - \widehat{v}. \qquad (7)
$$

According to (6) and (7), we obtain

$$
\begin{aligned}
\widehat{v}_\rho &= \sum_{i\in[n]} \sum_{k\in[K]} \widehat{\boldsymbol{\delta}}_{ik} \ell_k(y_i, \boldsymbol{x}_i, \widehat{\boldsymbol{\theta}}_k) + \rho \sum_{i\in[n]} \sum_{k\in[K]} \widehat{\boldsymbol{\delta}}_{ik\rho} S_k\left(\boldsymbol{x}_i, \widehat{\boldsymbol{m}}_{k\rho}\right) \\
&\geq \sum_{i\in[n]} \sum_{k\in[K]} \delta^*_{ik} \ell_k(y_i, \boldsymbol{x}_i, \boldsymbol{\theta}^*_k) - \widehat{v} + \rho \left[\sum_{i\in[n]} \sum_{k\in[K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right) + \widehat{s} \right].
\end{aligned}
$$

Thus, for $\rho \geq \bar{\rho} := \frac{\widehat{v}}{\widehat{s}}$, we must have

$$
\widehat{v}_\rho \geq v^* + \rho \sum_{i\in[n]} \sum_{k\in[K]} \delta^*_{ik} S_k\left(\boldsymbol{x}_i, \boldsymbol{m}^*_k\right),
$$

a contradiction. Q.E.D.

Observation 1 and Proposition 1 imply that by allowing to change ρ, our CluSL (1) is indeed more general. It is a flexible integration of two kinds of learning tasks, i.e., clustering and SL. Finally, we remark that if within each cluster, there is an over-fitting issue, then we recommend including L_1 or L_0 penalty into the loss functions.

3. Regularized Alternating Minimization with Its Convergence Analysis

The CluSL framework (1) is, in general, a non-convex problem, and is NP-hard even if the loss functions are all zero (Mahajan et al. 2009). To solve it, in particular, with the large-scale datasets, we propose a efficient convergent algorithm based on the alternating minimization (AM) method. Due to the fast empirical convergence speed, AM has been widely applied for solving practical problems such as compressive sensing (Liao et al.

2014, Lai and Varghese 2017), matrix completion (Jain et al. 2012, Tanner and Wei 2016, Cui and Fan 2017), power system (Vu 2019), etc. The earlier works can data back to Csiszar and Tusnady (1984), Von Neumann (1950). Since then, its variants have been extensively explored in many convex problems (Brègman 1965, Luo and Tseng 1993, Boyd et al. 2003, Beck and Amir 2015, Lai and Varghese 2017). However, convergence results of AM in nonconvex problems are quite limited and often require certain assumptions. For example, the three-point and four-point properties were introduced in Csiszar and Tusnady (1984) to prove the convergence of AM to a stationary point. For some special nonconvex programs satisfying the transversality condition, Drusvyatskiy et al. (2015) proved the local linear convergence of AM algorithm. Recently, with an assumption of three-point and local contraction properties, the sequence generated by AM algorithm was proved by Zhu and Li (2018) to converge to a stationary point in the nonconvex problem. Other similar works can also be found in Attouch et al. (2008), Lewis et al. (2009), Attouch et al. (2010), Czepiel (2019). Unfortunately, all of these conditions are not applicable to the setting of this paper. Therefore, we develop a variant of AM algorithm and further prove that it always converges to stationary point without making any additional assumption.

The basic AM algorithm proceeds as follows: (i) separate the decision variables of CluSL (1) into two parts, i.e., $(\boldsymbol{\theta}, \boldsymbol{m})$ and $\boldsymbol{\delta}$, (ii) solve CluSL (1) with respect to $(\boldsymbol{\theta}, \boldsymbol{m})$ by fixing $\boldsymbol{\delta}$ and vice versa, and (iii) terminate whenever the clustering solutions from two consecutive iterations are close to each other. However, the clustering solutions generated by this basic AM algorithm might not converge, which is illustrated in Appendix A. Thus, to fix it, we propose a regularized alternating minimization (RAM) algorithm, that is, at each iteration of AM algorithm, we add an $L_{1,1}-$ penalty term into the objective function of CluSL (1) to ensure that the current clustering solution and the previous one are not too far away from each other. In the following subsections, we introduce the detailed implementation of the RAM algorithm for solving CluSL (1) and provide its convergence analysis.

3.1. Regularized Alternating Minimization (RAM) Algorithm

Recall that we let $R_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m})$ denote the objective function of CluSL (1). Given a reference clustering solution $\widehat{\boldsymbol{\delta}}$, let us penalize $R_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m})$ with respect to $\widehat{\boldsymbol{\delta}}$ using the $L_{1,1}$ norm as below:

$$H_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m}, \widehat{\boldsymbol{\delta}}) := R_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m}) + \epsilon \|\boldsymbol{\delta} - \widehat{\boldsymbol{\delta}}\|_{1,1} \tag{8}$$

where $\epsilon > 0$ is a small regularization parameter and $\|.\|_{1,1}$ denotes $L_{1,1}$ norm. Using the fact that both $\boldsymbol{\delta}$ and $\widehat{\boldsymbol{\delta}}$ are binary matrices, we observe that the function $H_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m}, \widehat{\boldsymbol{\delta}})$ has an equivalent form, which is affine in $\boldsymbol{\delta}$.

Observation 2 *Function $H_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m})$ is equivalent to*

$$\widehat{H}_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m}, \widehat{\boldsymbol{\delta}}) := R_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m}) + \epsilon \sum_{k\in[K]} \sum_{i\in[n]} \left[\widehat{\delta}_{ik} + \left(1 - 2\widehat{\delta}_{ik}\right) \delta_{ik}\right], \tag{9}$$

which is affine in $\boldsymbol{\delta}$.

The proposed RAM algorithm proceeds as follows. We start with an initial clustering solution $\boldsymbol{\delta}^0 \in \Delta$. At iteration t, we obtain the optimal $(\boldsymbol{\theta}^t, \boldsymbol{m}^t)$ by solving $\min_{\boldsymbol{\theta},\boldsymbol{m}} H_\rho(\boldsymbol{\delta}^{t-1}, \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\delta}^{t-1})$, or equivalently, solving $\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho(\boldsymbol{\delta}^{t-1}, \boldsymbol{\theta}, \boldsymbol{m})$ and then obtain the optimal $\boldsymbol{\delta}^t$ by solving $\min_{\boldsymbol{\delta}\in\Delta} H_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}^t, \boldsymbol{m}^t, \boldsymbol{\delta}^{t-1})$. We continue this procedure until the clustering solution $\boldsymbol{\delta}^t$ no longer changes. The detailed implementation can be found in Algorithm 1.

Algorithm 1 RAM algorithm for solving CluSL (1)

```
Input: $\{\boldsymbol{x}_i, y_i\}_{i\in[n]}$, $K$, $\rho > 0$, $\epsilon > 0$
1: $t = 0$, an initial clustering solution $\boldsymbol{\delta}^0$ and $(\boldsymbol{\theta}^0, \boldsymbol{m}^0) \in \arg\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho(\boldsymbol{\delta}^0, \boldsymbol{\theta}, \boldsymbol{m})$
2: do
3:     $(\boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}) \in \arg\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}, \boldsymbol{m})$
4:     $\boldsymbol{\delta}^{t+1} \in \arg\min_{\boldsymbol{\delta}\in\Delta} H_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}, \boldsymbol{\delta}^t)$
5:     $t = t + 1$
6: while $\boldsymbol{\delta}^t \neq \boldsymbol{\delta}^{t-1}$
Output: $(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t)$
```

We remark that (i) one can improve the results by running RAM Algorithm 1 multiple times by choosing the initial clustering solution randomly; and (ii) in the Step 4 of RAM Algorithm 1, the optimization over $\boldsymbol{\delta} \in \Delta$ can be effectively solved as a linear program, according to Observation 2 and the fact that the constraint system defining Δ is integral (Schrijver 1998). This result is summarized below.

Proposition 2 *Let $\widehat{\Delta}$ denote the continuous relaxation of set Δ, i.e., $\widehat{\Delta} = \{\boldsymbol{\delta} \in [0,1]^{n\times K} : (1b) - (1c)\}$. Then*

$$\min_{\boldsymbol{\delta}\in\Delta} H_\rho(\boldsymbol{\delta},\boldsymbol{\theta},\boldsymbol{m},\widehat{\boldsymbol{\delta}}) = \min_{\boldsymbol{\delta}\in\widehat{\Delta}} \widehat{H}_\rho(\boldsymbol{\delta},\boldsymbol{\theta},\boldsymbol{m},\widehat{\boldsymbol{\delta}})$$

for all $\widehat{\boldsymbol{\delta}} \in \Delta$.

Thus, to solve $\min_{\boldsymbol{\delta}\in\Delta} H_\rho(\boldsymbol{\delta},\boldsymbol{\theta},\boldsymbol{m},\widehat{\boldsymbol{\delta}})$, we only need to solve a linear program. Additionally, to solve $\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho(\widehat{\boldsymbol{\delta}},\boldsymbol{\theta},\boldsymbol{m})$ for a fixed $\widehat{\boldsymbol{\delta}}$ can be decomposed into $2K$ subproblems.

Observation 3 *For any given $\widehat{\boldsymbol{\delta}}$,*

$$\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho(\widehat{\boldsymbol{\delta}},\boldsymbol{\theta},\boldsymbol{m}) = \sum_{k\in[K]} \min_{\boldsymbol{\theta}_k} \sum_{i\in[n]} \widehat{\delta}_{ik}\ell_k(y_i,\boldsymbol{x}_i,\boldsymbol{\theta}_k) + \rho \sum_{k\in[K]} \min_{\boldsymbol{m}_k} \sum_{i\in[n]} \widehat{\delta}_{ik} S_k\left(\boldsymbol{x}_i,\boldsymbol{m}_k\right). \tag{10}$$

In (10), solving $\min_{\boldsymbol{m}_k} \sum_{i\in[n]} \widehat{\delta}_{ik} S_k\left(\boldsymbol{x}_i,\boldsymbol{m}_k\right)$ for each $k \in [K]$ is often relatively easy, for example, if the dissimilarity function is squared Euclidean or Gaussian, then the optimal $\widehat{\boldsymbol{m}}_k = \frac{\sum_{i\in[n]}\widehat{\delta}_{ik}\boldsymbol{x}_i}{\sum_{i\in[n]}\widehat{\delta}_{ik}}$, while if the dissimilarity function is cosine, then the optimal $\widehat{\boldsymbol{m}}_k = \frac{\sum_{i\in[n]}\widehat{\delta}_{ik}\boldsymbol{x}_i}{\|\sum_{i\in[n]}\widehat{\delta}_{ik}\boldsymbol{x}_i\|}$. On the other hand, solving $\min_{\boldsymbol{\theta}_k} \sum_{i\in[n]} \widehat{\delta}_{ik}\ell_k(y_i,\boldsymbol{x}_i,\boldsymbol{\theta}_k)$ depends on the loss functions, and there is rich literature on solving SL problems; see, e.g., Zheng and Hong (2018), Yang et al. (2018), Derisma et al. (2018), Dass et al. (2019), etc. Thus, for the simplicity of analysis, we assume in this paper that it can be solved to optimality effectively.

We remark that to choose the best tuning parameter ρ from a predetermined list using cross validation, one might need to run RAM Algorithm 1 multiple times. To accelerate this procedure, we can use a warm start, i.e., the best solution found in the previous CluSL with a different ρ can be set as an initial solution of the current CluSL. In practice, this warms start procedure can significantly improve the convergence of the RAM algorithm.

3.2. Convergence Analysis

In this subsection, we analyze the RAM algorithm, prove the convergence of sequence $\{\boldsymbol{\delta}^t\}_t$, and show that the RAM algorithm will always terminate in a finite number of iterations.

First, we observe that the sequence of the objective values $\{R_\rho(\boldsymbol{\delta}^t,\boldsymbol{\theta}^t,\boldsymbol{m}^t)\}_t$ of RAM Algorithm 1 is monotonically non-increasing.

Lemma 1 *The sequence of the objective values $\{R_\rho(\boldsymbol{\delta}^t,\boldsymbol{\theta}^t,\boldsymbol{m}^t)\}_t$ of RAM Algorithm 1 is monotonically non-increasing and bounded from below, and hence converges.*

Proof: In Algorithm 1, at $(t+1)$th iteration, given the previous clustering decision $\boldsymbol{\delta}^t$, the fact that $(\boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}) \in \arg\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho\left(\boldsymbol{\delta}^t, \boldsymbol{\theta}, \boldsymbol{m}\right)$ implies that

$$R_\rho\left(\boldsymbol{\delta}^t, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}\right) \le R_\rho\left(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t\right), \tag{11}$$

and $\boldsymbol{\delta}^{t+1} \in \arg\min_{\boldsymbol{\delta}\in\Delta} H_\rho\left(\boldsymbol{\delta}, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}, \boldsymbol{\delta}^t\right)$ implies that

$$H_\rho\left(\boldsymbol{\delta}^{t+1}, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}, \boldsymbol{\delta}^t\right) \le H_\rho\left(\boldsymbol{\delta}^t, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}, \boldsymbol{\delta}^t\right) := R_\rho\left(\boldsymbol{\delta}^t, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}\right), \tag{12}$$

which further implies that

$$R_\rho\left(\boldsymbol{\delta}^{t+1}, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}\right) \le R_\rho\left(\boldsymbol{\delta}^t, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}\right) \tag{13}$$

since $\|\boldsymbol{\delta}^{t+1} - \boldsymbol{\delta}^t\|_{1,1} \ge 0$.

Summing up (11) and (13) yields

$$R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t) \ge R_\rho(\boldsymbol{\delta}^{t+1}, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}),$$

which implies that the sequence of output objective values $\left\{R_\rho\left(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t\right)\right\}_t$ of RAM algorithm is monotone non-increasing. On the other hand, according to the definition, $R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t) \ge 0$ for all t. Hence, the monotone convergence theorem implies that the sequence $\left\{R_\rho\left(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t\right)\right\}_t$ is indeed convergent. Q.E.D.

In RAM Algorithm 1, penalizing the distance between the current clustering solution and the one from the previous iteration guarantees that the sequence of the clustering solutions $\{\boldsymbol{\delta}^t\}_t$ is convergent. Additionally, the fact that $\boldsymbol{\delta}^t$ is binary for all t, and there is only a limited number of binary matrices in set Δ, implies that RAM Algorithm 1 will terminate in a finite number of iterations.

Theorem 1 *There exists a positive integer $\mathcal{T} < \infty$, such that*

(i) $\lim_{t\to\infty} \boldsymbol{\delta}^t = \boldsymbol{\delta}^{\mathcal{T}}$,

(ii) $\mathcal{T} \le (2\epsilon)^{-1}(R_\rho(\boldsymbol{\delta}^0, \boldsymbol{\theta}^0, \boldsymbol{m}^0) - v_\rho)$, *and*

(iii) $\lim_{t\to\infty} R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t) = R_\rho(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}})$.

Proof: We will first prove part (i). From (11) and (12) in the proof of Lemma 1, we have

$$R_\rho(\boldsymbol{\delta}^{t+1}, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}) + \epsilon\|\boldsymbol{\delta}^{t+1} - \boldsymbol{\delta}^t\|_{1,1} \le R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}) \le R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t). \tag{14}$$

Then we prove it by contradiction. Suppose that for all t, we have $\boldsymbol{\delta}^{t+1} \neq \boldsymbol{\delta}^t$. Since $\boldsymbol{\delta}^t \in \{0,1\}^{n\times K}$ for all t, thus $\|\boldsymbol{\delta}^{t+1} - \boldsymbol{\delta}^t\|_{1,1} \geq 2$ for all t. Thus, (14) further implies that

$$R_\rho(\boldsymbol{\delta}^{t+1}, \boldsymbol{\theta}^{t+1}, \boldsymbol{m}^{t+1}) + 2\epsilon \leq R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t). \tag{15}$$

Summing it up over all t and using the fact that $\lim_{T\to\infty} R_\rho(\boldsymbol{\delta}^{T+1}, \boldsymbol{\theta}^{T+1}, \boldsymbol{m}^{T+1}) < \infty$, we have

$$\infty = \lim_{T\to\infty} (R_\rho(\boldsymbol{\delta}^{T+1}, \boldsymbol{\theta}^{T+1}, \boldsymbol{m}^{T+1}) + 2T\epsilon) \leq R_\rho(\boldsymbol{\delta}^1, \boldsymbol{\theta}^1, \boldsymbol{m}^1) < \infty$$

a contradiction.

Part (ii) directly follows that at each iteration if the algorithm does not stop, then the objective function will decrease by at least 2ϵ. To prove part (iii), we know that from part (i), there exists a $\mathcal{T} > 0$ such that $\lim_{t\to\infty} \boldsymbol{\delta}^t = \boldsymbol{\delta}^{\mathcal{T}}$, i.e., $\boldsymbol{\delta}^t = \boldsymbol{\delta}^{\mathcal{T}}$ for all $t \geq \mathcal{T}$. According to Algorithm 1, we have

$$R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t) = R_\rho(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}})$$

for all $t \geq \mathcal{T}$. Q.E.D.

Although Theorem 1 ensures that the sequence of clustering solutions converges finitely, the output of RAM Algorithm 1, denoted by $\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right)$, might not be a stationary point of CluSL (1). That is, we would like to make sure that

$$\left(\boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) \in \arg\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}, \boldsymbol{m}\right), \quad \boldsymbol{\delta}^{\mathcal{T}} \in \arg\min_{\boldsymbol{\delta}\in\Delta} R_\rho\left(\boldsymbol{\delta}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right). \tag{16}$$

In fact, these two conditions can be achieved when the regularization parameter ϵ is small enough.

Theorem 2 *Suppose that for each $\boldsymbol{\delta} \in \Delta$, the set $\Phi(\boldsymbol{\delta}) := \min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m})$ is finite. Then there exists a threshold $\widehat{\epsilon}$ such that for any $0 < \epsilon < \widehat{\epsilon}$, the output from RAM Algorithm 1, denoted by $\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right)$, will be a stationary point of CluSL* (1)*, i.e., $\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right)$ satisfied the conditions* (16)*. In particular, the threshold $\widehat{\epsilon} = \frac{\widehat{d}}{2n}$, where $\widehat{d}$ is defined as below:*

$$\widehat{d} := \min_{\boldsymbol{\delta}\in\Delta} \left\{ \min_{\widehat{\boldsymbol{\delta}}\in\Delta, (\bar{\boldsymbol{\theta}},\bar{\boldsymbol{m}})\in\Phi(\boldsymbol{\delta})} \left[R_\rho\left(\widehat{\boldsymbol{\delta}}, \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{m}}\right) - R_\rho\left(\boldsymbol{\delta}, \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{m}}\right)\right] > 0 \right\}. \tag{17}$$

Proof: From Theorem 1, we know that there exists a positive integer $\mathcal{T} < \infty$ such that $\lim_{t\to\infty} \boldsymbol{\delta}^t = \boldsymbol{\delta}^{\mathcal{T}}$ and $\lim_{t\to\infty} R_\rho(\boldsymbol{\delta}^t, \boldsymbol{\theta}^t, \boldsymbol{m}^t) = R_\rho(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}})$.

Since $\left(\boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) \in \arg\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}, \boldsymbol{m}\right)$, we only need to show that

$$\boldsymbol{\delta}^{\mathcal{T}} \in \arg\min_{\boldsymbol{\delta}\in\Delta} R_\rho\left(\boldsymbol{\delta}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right). \tag{18}$$

We prove it by contradiction. Suppose (18) does not hold, and let $\boldsymbol{\delta}^*$ be an optimal solution of (18). Then according to Algorithm 1, we know that

$$\boldsymbol{\delta}^{\mathcal{T}} \in \arg\min_{\boldsymbol{\delta}\in\Delta} R_\rho\left(\boldsymbol{\delta}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) + \epsilon\|\boldsymbol{\delta} - \boldsymbol{\delta}^{\mathcal{T}}\|_{1,1},$$

which implies that

$$R_\rho\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) \le R_\rho\left(\boldsymbol{\delta}^*, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) + \epsilon\|\boldsymbol{\delta}^* - \boldsymbol{\delta}^{\mathcal{T}}\|_{1,1}.$$

As $\boldsymbol{\delta}^*, \boldsymbol{\delta} \in \Delta$, hence we have $\|\boldsymbol{\delta}^*\|_{1,1} \le n$, $\|\boldsymbol{\delta}^{\mathcal{T}}\|_{1,1} \le n$. Hence, the above inequality is further reduced to

$$\begin{aligned} R_\rho\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) &\le R_\rho\left(\boldsymbol{\delta}^*, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) + \epsilon\|\boldsymbol{\delta}^* - \boldsymbol{\delta}^{\mathcal{T}}\|_{1,1} \le R_\rho\left(\boldsymbol{\delta}^*, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) + 2n\epsilon \\ &< R_\rho\left(\boldsymbol{\delta}^*, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) + \widehat{d}, \end{aligned}$$

where the second inequality is because triangle inequality and third one is due to $\epsilon < \frac{\widehat{d}}{2n}$. According to the definitions of $\widehat{d}$ and optimality of $\boldsymbol{\delta}^*$, we must have $R_\rho\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) = R_\rho\left(\boldsymbol{\delta}^*, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right)$, a contradiction. Q.E.D.

Note that (i) in Theorem 2, $\widehat{d}$ represents the smallest difference of arbitrarily two non-identical elements in the finite sequence $\{\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho(\boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{m})\}_{\boldsymbol{\delta}\in\Delta}$ and it is well-defined since $|\Delta| < \infty$; and (ii) the theoretical upper bound $\widehat{\epsilon}$ of regularization parameter ϵ might be too small and difficult to compute, and oftentimes, a larger ϵ can help accelerate the convergence. Therefore, in practice, we recommend users selecting an appropriate ϵ considering both the computational time and quality of solutions. Finally, we remark that if we choose ϵ to be small but not to satisfy the condition in Theorem 2, then $\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right)$ is an approximate stationary point.

Corollary 1 *For any $0 < \epsilon$, the output from RAM Algorithm 1, denoted by $\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right)$, is an approximate stationary point of CluSL* (1)*, i.e.,*

$$\left(\boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) \in \arg\min_{\boldsymbol{\theta},\boldsymbol{m}} R_\rho\left(\boldsymbol{\delta}^{\mathcal{T}}, \boldsymbol{\theta}, \boldsymbol{m}\right), \quad R_\rho\left(\boldsymbol{\delta}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) - \min_{\boldsymbol{\delta}\in\Delta} R_\rho\left(\boldsymbol{\delta}, \boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{m}^{\mathcal{T}}\right) \le 2n\epsilon. \tag{19}$$

4. Extension: F-CluSL Framework for the High-dimensional Datasets

In this section, we extend CluSL to the high-dimensional datasets and generate the feature cluster-aware supervised learning (F-CluSL) framework. In F-CluSL, instead of clustering data points, we consider clustering the features.

4.1. Setting

Formally, suppose that we are given a dataset of n training samples $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$ where data points $\boldsymbol{x}_i \in \mathbb{R}^p$. We consider clustering similar features of $\boldsymbol{x}$ (i.e., $\{\boldsymbol{x}_{.j}\}_{j\in[p]}$) into K groups. Specifically, we let binary variable $\delta^F_{kj} = 1$ denotes that the jth feature belongs to cluster $k \in [K]$; 0, otherwise. For each cluster $k \in [K]$, we generate a new feature $\boldsymbol{m}^F_{.k}$ to represent those within this cluster. By doing so, we can effectively reduce the number of features into K. Similar ideas can be found in group LASSO (Friedman et al. 2010, Simon et al. 2013, Ma et al. 2007, Lai and Mckenzie 2020). Thus, each sample $\boldsymbol{x}_i$ is transformed into a truncated one $\boldsymbol{m}^F_i$ with only K columns, i.e., $\boldsymbol{m}^F_i = \left(m^F_{i1}, m^F_{i2}, ..., m^F_{iK}\right)$ for each $i \in [n]$.

With the notation above, we formulate the following F-CluSL framework:

$$v^F_\rho := \min_{\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, \boldsymbol{m}^F} R_\rho(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, \boldsymbol{m}^F) := \sum_{i\in[n]} \ell_i(y_i, \boldsymbol{m}^F_i, \boldsymbol{\theta}^F) + \rho \sum_{k\in[K]} \sum_{j\in[p]} \delta^F_{kj} S_k\left(\boldsymbol{x}_{.j}, \boldsymbol{m}^F_{.k}\right), \tag{20a}$$

$$\text{s.t.} \quad \sum_{k\in[K]} \delta^F_{kj} = 1, \forall j \in [p], \tag{20b}$$

$$\sum_{j\in[p]} \delta^F_{kj} \geq 1, \forall k \in [K], \tag{20c}$$

$$\delta^F_{kj} \in \{0,1\}, \forall k \in [K], \forall j \in [p], \tag{20d}$$

where $\ell_i : \mathbb{R} \times \mathbb{R}^K \times \mathbb{R}^d \to \mathbb{R}_+$ denote ith loss function and $S_k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ represents kth dissimilarity function. The specific examples of loss function and dissimilarity function can be found in Section 2.2.

For the notational convenience, we let set Δ_F denote the feasible region of F-CluSL (20). Similarly, if ρ is large enough (larger than a threshold), the F-CluSL framework can be also reduced to feature-extraction-first-then-SL, which can be formulated as below:

$$\min_{\boldsymbol{\theta}} \sum_{i\in[n]} \ell_i(y_i, \boldsymbol{m}^F_i, \boldsymbol{\theta}^F) \tag{21a}$$

$$\text{s.t.} \quad (\boldsymbol{\delta}^{F*}, \boldsymbol{m}^{F*}) \in \arg\min_{\boldsymbol{\delta}^F \in \Delta_F, \boldsymbol{m}^F} \sum_{k\in[K]} \sum_{j\in[p]} \delta^F_{kj} S_k\left(\boldsymbol{x}_{.j}, \boldsymbol{m}^F_{.k}\right). \tag{21b}$$

In fact, feature-extraction-first-then-SL (21) is a generalization of the state-of-the-art in image classification (Chen et al. 2016, Yue et al. 2015, Zhao and Du 2016), which extracts the representative low-dimension features for classification. This result is summarized below and since the proof is similar to Proposition 1, thus is omitted here.

Proposition 3 *There exists a positive $\bar{\rho} > 0$ such that for any $\rho \geq \bar{\rho}$, the F-CluSL* (20) *is equivalent to feature clustering, extraction first-then-SL problem* (21). *The threshold $\bar{\rho}$ can be chosen as $\bar{\rho} = \frac{\widehat{v}^F}{\widehat{s}^F}$, where $\widehat{v}^F \geq 0$ and $\widehat{s}^F > 0$ are defined as below:*

$$\widehat{s}^F := \min_{\boldsymbol{\delta}^F \in \Delta_F} \left\{ \min_{\boldsymbol{m}^F} \sum_{k\in[K]} \sum_{j\in[p]} \delta^F_{kj} S_k\left(\boldsymbol{x}_{.j}, \boldsymbol{m}^F_{.k}\right) - \sum_{k\in[K]} \sum_{j\in[p]} \delta^{F*}_{kj} S_k\left(\boldsymbol{x}_{.j}, \boldsymbol{m}^{F*}_{.k}\right) > 0 \right\},$$

$$\widehat{v}^F := \max_{\boldsymbol{\delta}^F \in \Delta_F} \left\{ \min_{\boldsymbol{\theta}^F} \sum_{i\in[n]} \ell_i(y_i, \boldsymbol{m}^{F*}_{i.}, \boldsymbol{\theta}^F) - \min_{\boldsymbol{\theta}^F, \boldsymbol{m}^F} \sum_{i\in[n]} \ell_i(y_i, \boldsymbol{m}^F_{i.}, \boldsymbol{\theta}^F) \geq 0 \right\},$$

where $(\boldsymbol{\delta}^{F*}, \boldsymbol{m}^{F*})$ is defined in (21).

4.2. Regularized Alternating Minimization for F-CluSL and Its Convergence Analysis

We first observe that in F-CluSL (20), once the extracted features $\boldsymbol{m}^F$ are known, then it can be decomposed into two problems: (i) optimization over $\boldsymbol{\theta}^F$, which is assumed to be effectively solved to optimality; and (ii) optimization over $\boldsymbol{\delta}^F$, which is a polynomial-solvable assignment problem. This motivates our F-RAM algorithm for F-CluSL framework. Different from RAM Algorithm 1, we should choose to add a regularization term $\epsilon\|\boldsymbol{m}^F - \widehat{\boldsymbol{m}}^F\|_{1,1}$ given a previous extracted feature matrix $\widehat{\boldsymbol{m}}^F$ and the regularization parameter $\epsilon > 0$. That is, we define our regularized objective function as

$$H_\rho(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, \boldsymbol{m}^F, \widehat{\boldsymbol{m}}^F) := R_\rho(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, \boldsymbol{m}^F) + \epsilon\|\boldsymbol{m}^F - \widehat{\boldsymbol{m}}^F\|_1. \tag{22}$$

The proposed F-RAM Algorithm 2 proceeds as follows. We first randomly generate K extracted features, denoted as $(\boldsymbol{m}^F)^0$. At iteration t, given $(\boldsymbol{m}^F)^t$, we obtain the optimal feature clustering and estimator $((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1})$ by solving

$$((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}) \in \operatorname*{arg\,min}_{\boldsymbol{\theta}^F, \boldsymbol{\delta}^F \in \Delta_F} H_\rho(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, (\boldsymbol{m}^F)^t, (\boldsymbol{m}^F)^t) := R_\rho(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, (\boldsymbol{m}^F)^t),$$

which can be decomposed into two subproblems:

$$(\boldsymbol{\theta}^F)^{t+1} \in \operatorname*{arg\,min}_{\boldsymbol{\theta}^F} \sum_{i\in[n]} \ell_i(y_i, (\boldsymbol{m}^F)^t_i, \boldsymbol{\theta}^F), \quad (\boldsymbol{\delta}^F)^{t+1} \in \operatorname*{arg\,min}_{\boldsymbol{\delta}^F \in \Delta_F} \sum_{k\in[K]} \sum_{j\in[p]} \delta^F_{kj} S_k\left(\boldsymbol{x}_{.j}, (\boldsymbol{m}^F)^{t-1}_{.k}\right).$$

Next, given $((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1})$, we obtain the optimal $(\boldsymbol{m}^F)^t$ by solving

$$(\boldsymbol{m}^F)^{t+1} \in \arg\min_{\boldsymbol{m}^F} H_\rho((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, \boldsymbol{m}^F, (\boldsymbol{m}^F)^t).$$

We continue this procedure until the extracted feature matrix $(\boldsymbol{m}^F)^t$ no longer changes. The detailed implementation can be found in Algorithm 2.

Algorithm 2 F-RAM Algorithm 2 for solving F-CluSL (20)

```
Input: Given $\{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$, $K$, $\rho > 0$, $\epsilon > 0$
1: Let $t = 0$, randomly select $K$ features $(\boldsymbol{m}^F)^0$ and $\left((\boldsymbol{\theta}^F)^0, (\boldsymbol{\delta}^F)^0\right) \in \arg\min_{\boldsymbol{\theta}^F, \boldsymbol{\delta}^F \in \Delta_F} R_\rho\left(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, (\boldsymbol{m}^F)^0\right)$
2: do
3:     $\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}\right) \in \arg\min_{\boldsymbol{\theta}^F, \boldsymbol{\delta}^F \in \Delta_F} R_\rho\left(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, (\boldsymbol{m}^F)^t\right)$
4:     $(\boldsymbol{m}^F)^{t+1} \in \arg\min_{\boldsymbol{m}^F} H_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, \boldsymbol{m}^F, (\boldsymbol{m}^F)^t\right)$
5:     $t = t + 1$
6: while $(\boldsymbol{m}^F)^t \neq (\boldsymbol{m}^F)^{t-1}$
Output: $\left((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t\right)$
```

We also analyze the convergence of both the sequence of the objective function values $\{R_\rho((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t)\}_t$ and the sequence $\{(\boldsymbol{m}^F)^t\}_t$ output from F-RAM Algorithm 2. First, Lemma 2 shows the monotonically non-increasing property of objective function $\{R_\rho((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t)\}_t$ of F-RAM Algorithm 2. The proof of Lemma 2 is quite similar to that of Lemma 1 and can be found in Appendix B.1.

Lemma 2 *The sequence of the objective values $\{R_\rho((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t)\}_t$ of F-RAM Algorithm 2 is monotonically non-increasing and bounded from below, and hence converges.*

Proof: See Appendix B.1. □

In F-RAM Algorithm 2, the sequence of extracted feature matrices $\{(\boldsymbol{m}^F)^t\}_t$ is Cauchy due to the penalized distance between current solution and previous one, just as shown in Theorem 3.

Theorem 3 *Suppose that $\left\{\left((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t\right)\right\}_t$ denotes the iterated solution sequence of F-RAM Algorithm 2. Then the following two limits exist:*

(i) $\lim_{t\to\infty}(\boldsymbol{m}^F)^t = (\boldsymbol{m}^F)^\star$ *and*

(ii) $\lim_{t\to\infty} R_\rho((\boldsymbol{\theta}^F)^t,(\boldsymbol{\delta}^F)^t,(\boldsymbol{m}^F)^t) = R_\rho((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star,(\boldsymbol{m}^F)^\star)$, *where* $\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star\right) \in \arg\min_{\boldsymbol{\theta}^F,\boldsymbol{\delta}^F\in\Delta_F} R_\rho\left(\boldsymbol{\theta}^F,\boldsymbol{\delta}^F,(\boldsymbol{m}^F)^\star\right)$.

Proof: See Appendix B.2. □

The convergence results in Theorem 3 are not sufficient to ensure the solution quality of F-RAM Algorithm 2. Therefore, we would like to prove that its output $\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star,(\boldsymbol{m}^F)^\star\right)$ is a stationary point of F-CluSL (20) when $\epsilon\to 0$, i.e,

$$\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star\right) \in \mathop{\arg\min}_{\boldsymbol{\theta}^F,\boldsymbol{\delta}^F\in\Delta_F} R_\rho\left(\boldsymbol{\theta}^F,\boldsymbol{\delta}^F,(\boldsymbol{m}^F)^\star\right), \quad (\boldsymbol{m}^F)^\star \in \mathop{\arg\min}_{\boldsymbol{m}^F} R_\rho\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star,\boldsymbol{m}^F\right). \tag{23}$$

Theorem 4 *Suppose that* $R_\rho\left(\boldsymbol{\theta}^F,\boldsymbol{\delta}^F,\boldsymbol{m}^F\right)$ *is convex in* $\boldsymbol{m}^F$ *for any given* $(\boldsymbol{\theta}^F,\boldsymbol{\delta}^F)$*. Then following the same notation as Theorem 3, we have*

(i) For any $\epsilon>0$, $((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star,(\boldsymbol{m}^F)^\star)$ *is an* ϵ*-stationary point, i.e.,*

$$\begin{aligned}&\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star\right) \in \mathop{\arg\min}_{\boldsymbol{\theta}^F,\boldsymbol{\delta}^F\in\Delta_F} R_\rho\left(\boldsymbol{\theta}^F,\boldsymbol{\delta}^F,(\boldsymbol{m}^F)^\star\right),\\ &dist((\boldsymbol{m}^F)^\star,\partial R_\rho\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star,(\boldsymbol{m}^F)^\star\right)) \le \epsilon,\end{aligned} \tag{24}$$

(ii) If $\epsilon\to 0$*, then* $((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star,(\boldsymbol{m}^F)^\star)$ *becomes a stationary point satisfying* (23)*.*

Proof: To prove part (i), we only need to show that $dist((\boldsymbol{m}^F)^\star,\partial R_\rho\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star,(\boldsymbol{m}^F)^\star\right)) \le \epsilon$. Indeed, according to the convergent results in Theorem 3, we know that

$$(\boldsymbol{m}^F)^\star \in \mathop{\arg\min}_{\boldsymbol{m}^F} R_\rho\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{m}^F)^\star,\boldsymbol{m}^F\right) + \epsilon\|\boldsymbol{m}^F-(\boldsymbol{m}^F)^\star\|_{1,1} \tag{25}$$

Since $R_\rho\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{m}^F)^\star,\boldsymbol{m}^F\right) + \epsilon\|\boldsymbol{m}^F-(\boldsymbol{m}^F)^\star\|_{1,1}$ is convex in $(\boldsymbol{m})^F$, according to the optimality condition of convex program (Ben-Tal and Nemirovski 2001), (25) is equivalent to

$$\mathbf{0} \in \partial_{\boldsymbol{m}^F} R_\rho\left((\boldsymbol{\theta}^F)^\star,(\boldsymbol{\delta}^F)^\star,(\boldsymbol{m}^F)^\star\right) + \epsilon\boldsymbol{\zeta}_{(\boldsymbol{m}^F)^\star}, \tag{26}$$

with subgradient $\boldsymbol{\zeta}_{(\boldsymbol{m}^F)^\star} \in \partial_{\boldsymbol{m}^F}(\|\boldsymbol{m}^F-(\boldsymbol{m}^F)^\star\|_{1,1})|_{\boldsymbol{m}^F=(\boldsymbol{m}^F)^\star}$. Since $\|\boldsymbol{\zeta}_{(\boldsymbol{m}^F)^\star}\|_\infty \le 1$, by choosing the distance metric as L_∞, we arrive at (24).

Part (ii) follows directly from part (i) by letting $\epsilon\to 0$. Q.E.D.

5. Numerical Experiments

In this section, we conduct numerical studies on cluster-aware regression and classification, to (a) demonstrate the performance of RAM algorithm compared with existing non-convex solver, (b) illustrate the interpretable results based on the CluSL and F-CluSL framework, and (c) compare the prediction results of the proposed CluSL and F-CluSL framework with the state-of-the-art methods, i.e. CLR, random forest (RF), support vector regression (SVR), support vector classification (SVC), and CNNs. All of the algorithms were implemented in a Windows 64 bit Desktop—Intel(R) Core(TM) four-core i7-6700K CPU @ 4.00GHZ, 16GB of RAM. We used Gurobi version 8.1.0 via its Python interface to solve the linear programming subproblem $\min_{\boldsymbol{\delta}\in\widehat{\Delta}} H_\rho(\boldsymbol{\delta},\boldsymbol{\theta},\boldsymbol{m},\widehat{\boldsymbol{\delta}})$ and $\min_{\boldsymbol{\delta}^F\in\Delta^F} H_\rho(\boldsymbol{\theta}^F,\boldsymbol{\delta}^F,\boldsymbol{m}^F,\widehat{\boldsymbol{m}}^F)$. For all the numerical instances, we chose the regularization parameter $\epsilon = 0.001$. All the data and codes of eight experiments in this section can be found at `https://github.com/cst950928/Numerical_experiment/tree/master`.

5.1. RAM Algorithm Testing

Experiment 1: Note that the proposed RAM Algorithm 1 might only find a near-optimal solution. Hence, we first tested the performance of RAM Algorithm 1 and compared it with non-convex solver Knitro implemented in Matlab R2016a, which also yields the sub-optimal solution, with respect to the output objective values and computational time. Here, we considered the cluster-aware regression problem, where in CluSL (1), the loss function was selected as negative log-likelihood loss with squared Euclidean dissimilarity function.

In Experiment 1, we generated a series of data points $\{\boldsymbol{x}_i\}_{i\in[n]}$ using p-dimensional multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{\Sigma}$, i.e. $\boldsymbol{x}_i \sim N(0,\boldsymbol{\Sigma})$ for all $i\in[n]$. The entries of $\boldsymbol{\Sigma}$ were set to be $\sigma_{tj} = 0.5^{|t-j|}$ for each $t,j\in[p]$. The corresponding response was generated according to $y_i = \boldsymbol{\beta}^\top \boldsymbol{x}_i + \widetilde{\epsilon}_i$ for each $i\in[n]$, where each component of $\boldsymbol{\beta}\in\mathbb{R}^p$ was randomly generated from the i.i.d. uniform distribution between -5 and 5, and $\widetilde{\epsilon}_i$ is the Gaussian noise with $\widetilde{\epsilon}_i \sim N(0,\varsigma^2)$ for each $i\in[n]$. Specifically, we set $p = 1,2,3$, $\varsigma^2 = 1,2,3$ and sample size $n = 30$ respectively. Hence, total 9 datasets were generated.

Let K=1, 2, 3, 4, ρ=0.1, 0.2, 0.3, 0.4, 0.5 respectively, and we ran the RAM algorithm for 10 repetitions, and reported the minimum objective value and total computational time.

Correspondingly, for any given K and ρ, we solved the CluSL (1) using solver Knitro. The gap between the output objective values of RAM algorithm (denoted as $obj.rap$) and Knitro (denoted as $obj.kni$) is defined as $(obj.rap - obj.kni)/obj.kni$. For each k and ρ, the objective value gap and computational time among 9 datasets are displayed in Table 1 and Table 2 respectively.

Table 1 The output objective value gap between RAM algorithm and solver Knitro in Experiment 1

# of Clusters	$\rho = 0.1$	$\rho = 0.2$	$\rho = 0.3$	$\rho = 0.4$	$\rho = 0.5$
$K = 1$	0.00%	0.00%	0.00%	0.00%	0.00%
$K = 2$	-19.71%	-21.01%	-19.02%	-20.46%	-14.35%
$K = 3$	-46.09%	-29.04%	-21.79%	-22.20%	-24.51%
$K = 4$	-20.43%	-17.93%	-18.71%	-2.75%	-7.36%

Table 2 Computational time (in seconds) of RAM algorithm and solver Knitro in Experiment 1

Methods	Instances	$\rho = 0.1$	$\rho = 0.2$	$\rho = 0.3$	$\rho = 0.4$	$\rho = 0.5$
CluSL	$K = 1$	0.14	0.15	0.14	0.14	0.15
	$K = 2$	0.17	0.20	0.17	0.18	0.18
	$K = 3$	0.32	0.31	0.31	0.29	0.30
	$K = 4$	0.37	0.40	0.37	0.40	0.36
Knitro	$K = 1$	13.67	13.28	13.06	13.08	13.15
	$K = 2$	26.31	27.56	26.82	26.60	27.49
	$K = 3$	45.31	44.90	47.98	45.21	44.22
	$K = 4$	60.52	61.61	62.19	63.40	62.25

As can be seen, when $K \geq 2$, the proposed RAM algorithm yields smaller objective value compared with Knitro, which can be smaller than that of solver Knitro up to 46.09%. In terms of computational time, RAM is much faster than using the solver Knitro. This demonstrates the effectiveness of the RAM algorithm.

5.2. Cluster-aware Regression

In this subsection, we considered the cluster-aware regression problem, which is similar to the one studied in Experiment 1. We conducted three numerical experiments. Experiment 2 was based on hypothetical data, while Experiments 3 and 4 were based on UCI data[1] and California housing data[2], respectively. In particular, in Experiment 2, we also

[1] Available at `https://archive.ics.uci.edu/ml/index.php`.

[2] Available at `https://www.kaggle.com/camnugent/California-housing-prices`.

conducted a thorough comparison with the existing cluster-wise linear regression (CLR) method and showed that our proposed CluSL outperforms CLR with better interpretable results and smaller testing errors.

Experiment 2: CluSL with CLR were compared in this experiment. All hypothetical data were generated using the following piece-wise functions:

$$f_1(x) = \begin{cases} -0.1x - 0.5, & if -4 \le x \le -3 \\ x + 1, & if -2 \le x \le -1 \\ -2x + 2, & if \quad 0 \le x \le 1 \\ x - 2, & if \quad 3 \le x \le 4 \end{cases},$$

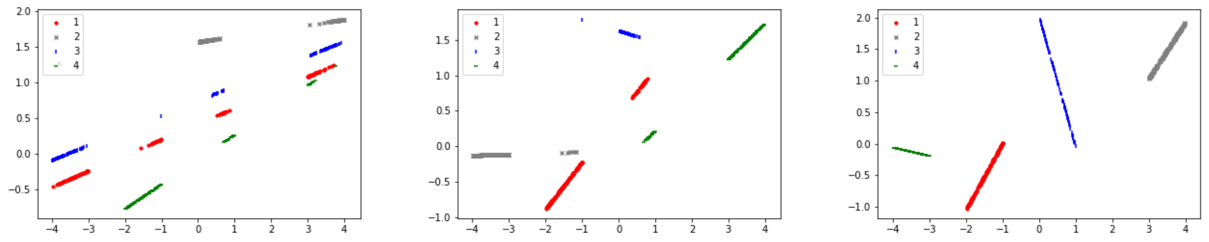
where each line represents one cluster. We also supposed that the data points are uniformly distributed at each interval of $[-4,-3], [-2,-1], [0,1], [3,4]$. Based on this assumption, we generated 125 training points for each interval. We also supposed that for each data point, the response is subject to i.i.d. Gaussian noise with mean zero and variance equal to 0.2, i.e., $y_i = f_1(x_i) + \widetilde{\epsilon}_i$ for each $i \in [500]$. For the testing, we generated three 500-sample testing groups $\boldsymbol{x}^{test_j}, j = 1,2,3$ in the similar way as the training dataset, while we assumed that within each testing group, its corresponding responses were subject to three different noises: $\tilde{\epsilon}_i \sim N(0,0)$, $\tilde{\epsilon}_i \sim N(0,0.25)$ and $\tilde{\epsilon}_i \sim N(0,0.5)$ for each $i \in [500]$. Therefore, in total, there are 9 testing datasets.

We set $K = 4$ for both CluSL and CLR. For CluSL, the tuning parameter ρ was set to be 0.05 and 0.1, denoted as CluSL_0.05 and CluSL_0.1, respectively. Since CLR is a special case of CluSL according to Observation 1, thus we ran RAM Algorithm 1 to solve all the instances. We ran the algorithm for 5 repetitions and chose the one with the minimum objective value. The Mean Squared Error (MSE) among training samples and computational time of CLR, CluSL_0.05, and CluSL_0.1 are displayed in Table 3. Further, Figure 1 illustrates the predicted function according to the training dataset.

Although the CLR yields a smaller training error, it requires a slightly more compu-

Table 3 Training MSE and computational time of CLR, CluSL_0.05 and CluSL_0.1 in Experiment 2

Methods	CLR	CluSL_0.05	CluSL_0.1
MSE	0.028	0.037	0.036
Time (s)	0.997	0.519	0.575



(a) CLR (b) CluSL_0.05 (c) CluSL_0.1

Figure 1 Illustration of predicted lines in Experiment 2

tational time than CluSL framework, which might be because CLR ignores the grouping structure of the training data and thus requires more iterations to converge. Figure 1 shows that CLR lacked interpretability and completely missed the shape of the 4-piecewise linear function. On the contrary, CluSL can learn the structure, making the predicted shape more consistent with the function f_1 with an increase of tuning parameter ρ. When ρ increases to 0.1, the CluSL framework can learn the discontinuity of function $f_1(x)$ exactly.

Figure 2 shows the prediction accuracy of the three methods. From the majority of the testing instances (except the 6th testing dataset), CluSL_0.05 and CluSL_0.1 both have smaller MSE than CLR, which implies the overestimation and poor prediction accuracy of CLR. In the 6th testing dataset, the CluSL_0.1 also outperforms CLR. Hence, Experiment 2 reveals that (i) although CLR might have the smallest MSE among training datasets, it can yield larger prediction errors. On the other hand, CluSL not only yields a similar training MSE but also can discover the hidden clustering structure within the data; and (ii) Compared with CLR, CluSL often has higher prediction accuracy by properly selecting the tuning parameter ρ.

Experiment 3: In this experiment, we conducted a comparison among CluSL, Random Forest (RF), and Support Vector Regression (SVR) methods using three UCI datasets. Both RF[3]and SVR[4] have open-sourced codes, respectively. The detailed information of UCI datasets is displayed in Table 4. All the datasets were shuffled first, then split into training samples (70%) and testing samples (30%). All the data were also normalized into mean zero and variance one. For each method, we fine-tuned the model parameters using

[3] We use the following package: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`

[4] We use the following package: `https://scikit-learn.org/stable/modules/svm.html#svm-regression`

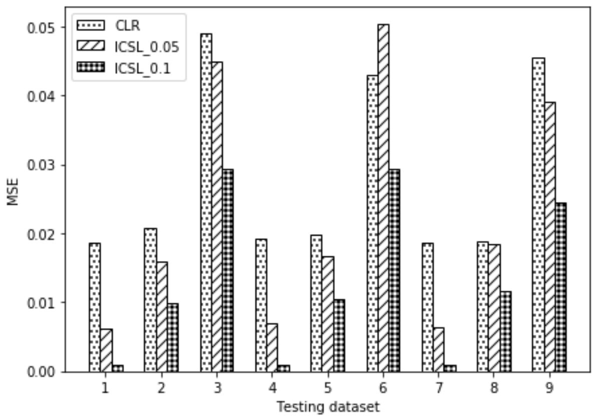


Figure 2 **MSE of CluSL and CLR among different testing datasets in Experiment 2**

Table 4 **Details of UCI datasets used in Experiment 3**

Dataset	Dimension	Training samples	Testing samples
Real estate valuation (REV)	6	290	124
Red wine quality (RQ)	11	1120	479
White wine quality (WQ)	11	3429	1469

Table 5 **The parameters for RF and SVR in Experiment 3**

Methods	Parameters
RF	Number of estimators (NE) =100, 110,..., 250 Number of splitting features (NF) =1, 2, ..., p
SVR	Kernel =RBF, linear, poly Tolerance (Tol) =1e-1, 1e-3, 1e-5 Penalty (Penal)= 1, 10, 20, 30, 40

the training samples and then calculated the MSE among testing samples. Specifically, in CluSL (1), we supposed that K and ρ can be chosen from 5,10,15,20 and 0.1,0.2,0.5, respectively. For RF and SVR methods, we also varied the parameters shown in Table 5.

The RAM algorithm was run for 10 repetitions. The summary of average testing MSE, best tuning parameter as well as the average training time of three methods are displayed in Table 6. More detail results are shown in Tables 18-20 of Appendix C.1. It is seen that the CluSL has the smallest MSE for all the instances, indicating its dominant prediction accuracy compared to other methods with slightly longer training time. When the number of clusters K is small, the computational time of CluSL is comparable to RF and SVR. Also, CluSL can both yield better prediction accuracy and improve interpretable results by exploring the clustering structure in the datasets with a proper tuning parameter ρ. When K becomes larger, both the computational time and the prediction accuracy of

Table 6 Summary of CluSL, RF and SVR testing and training results among three UCI datasets

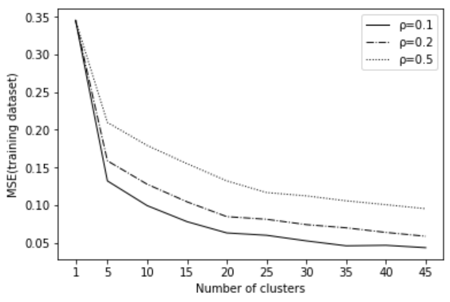
Dataset	Model	Average Testing MSE	Best Tuning Parameter	Average Training Time(s)
REV	CluSL	0.152	K=5, $\rho = 0.1$	0.480
	RF	0.222	NE=230, NF=2	0.332
	SVR	0.263	kernel=RBF, Tol=0.1, Penl=10	0.011
RQ	CluSL	0.339	K=5, $\rho = 0.1$	2.419
	RF	0.653	NE=180, NF=2	0.845
	SVR	0.686	kernal=linear, Tol=0.1, Penl=30	2.058
WQ	CluSL	0.288	K=5, $\rho = 0.1$	7.550
	RF	0.567	NE=240, NF=2	1.320
	SVR	0.578	kernel=RBF, Tol=0.001, Penl=1	0.679

CluSL increase. Overall, in the three UCI datasets above, CluSL framework outperforms two popular regression methods, i.e. RF and SVR in terms of MSE. In terms of computational time, we suggest that by selecting appropriate K and ρ, one can achieve the required MSE with a reasonable amount of time.

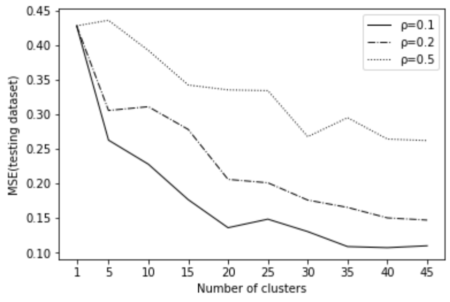
Experiment 4: Next, to demonstrate that CluSL can deliver interpretable results, we applied the proposed CluSL framework to a real-world case of California Housing Prices. The dataset contains 20,640 data points with 9 features, which are 1:longitude, 2:latitude, 3:housing median age, 4:total rooms, 5:total bedrooms, 6:population, 7:households, 8:median income, and 9:ocean proximity. The response variable is "median house value". Please note that we prepossessed the data by removing the dimension "ocean proximity" and 207 data points with NA value. Finally, the entire dataset was split as training dataset (14,303 points) and testing dataset (6,130 points). Each sample was normalized by mean zero and variance one.

How to select K: In order to find the most appropriate value of K, we ran the RAM algorithm for 5 times and chose the best result for each K=1, 5, 10,15, ..., 45, respectively. To discover the clustering pattern at different levels, we set ρ to be chosen from $\{0.1, 0.2, 0.5\}$. The MSEs among both training and testing datasets are plotted in Figure 3. Thus, according to the elbow method, we chose $K = 5$ and $\rho = 0.1$.

Next, to show the interpretability of CluSL framework, we selected two features "latitude" and "longitude" to illustrate the clustering results of CluSL_0.1 (i.e., $\rho = 0.1$) and CluSL_0 (i.e., $\rho = 0$) (see Figure 4). It can be seen that CluSL_0.1 yields more interpretable geographic clusters. For example, in Cluster 1 and Cluster 3 of CluSL_0.1, houses are mainly concentrated in large cities such as Los Angeles, San Diego (Cluster 1) and San



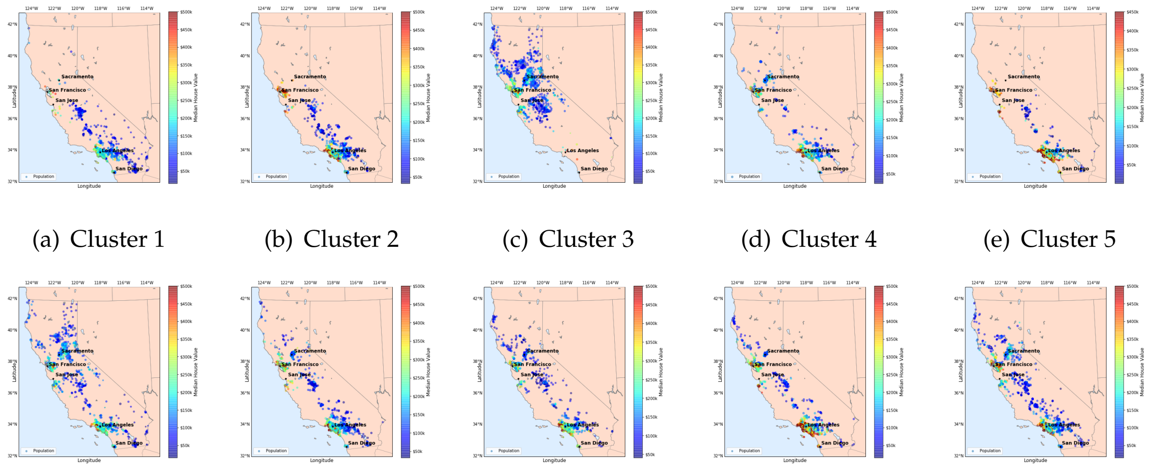
(a) MSE among training dataset



(b) MSE among testing datasets

Figure 3 Results of CluSL under different K and ρ in Experiment 4

Francisco, Sacramento, San Jose (Cluster 3). On the other hand, the clusters of CluSL_0 are much more disperse. Besides, we also illustrated the absolute values of regression



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5

(f) Cluster 1 (g) Cluster 2 (h) Cluster 3 (i) Cluster 4 (j) Cluster 5

Figure 4 Clustering results of CluSL in Experiment 4 (geographic locations). Figures (a)-(e): Clustering results with $\rho = 0.1$. Figure (f)-(j): clustering results with $\rho = 0$.

coefficients among each cluster in CluSL_0.1 in Figure 5, where the larger absolute value within each cluster means the more significant the feature is. It is seen that the dominating features, which largely influence the house value, vary significantly from clusters to clusters. For example, together with Figure 4(a)-(e), we see that in Clusters 2 and 5, the geographic location is the dominating feature, which implies that within these two clusters (mainly in San Francisco and Los Angeles), locations determine the housing prices. On the other hand, the key factors in Clusters 1, 3 and 4 are number of bedrooms, population, and median income, respectively. Therefore, this experiment shows that by splitting the dataset into different groups, one can improve the learning results and obtain valuable information.

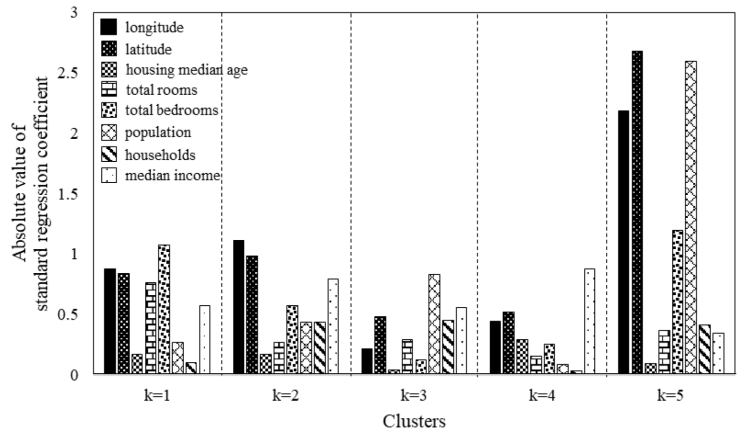


Figure 5 Absolute values of regression coefficients in Experiment 4

5.3. Cluster-aware Classification

In this subsection, we considered the cluster-aware classification problem, using both low-dimensional and high-dimensional datasets. Firstly, we conducted Experiments 5 by comparing the prediction accuracy, training accuracy, and computational time of CluSL with Support Vector Classification (SVC) based on low-dimensional datasets. The loss function in CluSL(1) was chosen to be hinge loss, which can be optimized effectively by existing python package sklearn.svm.LinearSVC (i.e., we ran sklearn.svm.LinearSVC at Step 3 of RAM Algorithm 1 when $\boldsymbol{\delta}^t$ is known). In Experiment 6 with high-dimensional data, we tested the performance of CluSL framework using several image datasets, where the loss function was chosen to be a neural network and CNNs were employed to solve Step 3 of RAM Algorithm 1 when $\boldsymbol{\delta}^t$ is known. The dissimilarity functions of both two experiments were squared Euclidean.

Experiment 5: We tested the performance of Cluster-aware classification framework using five UCI datasets. Detail information of datasets is shown in Table 7. In the dataset of white wine quality, the wine levels 3 and 9 contains very few data. Thus, we removed these two levels (i.e., classes) and focused on remaining 5 levels, where the resulting dataset is denoted as “White wine quality_5 classes (WW_5)”.

For each dataset, the number of clusters K and tuning parameter ρ in CluSL were chosen from 2,3,4,5 and 0,2,4,6,8, respectively. In each combination of K and ρ, three levels of “Tolerance for stopping criteria (Tol)”: (1e-1, 1e-3, 1e-5) and five levels of “Penalty

Table 7 Details of UCI datasets used in Experiment 5

Dataset	Dimension	Class	Training points	Testing points
Steel plates faults (SPF)	27	7	1358	583
Wireless indoor localization (WIL)	7	4	1400	600
Yeast	8	9	1035	443
Contraceptive Method Choice (CMC)	9	3	1031	441
White wine quality_5 classes (WW_5)	11	5	3412	1461

parameter of the error term (Penal)": (1, 10, 20, 30, 40) were considered respectively. The SVC model was also implemented using sklearn.svm.LinearSVC, where "hinge" and "squared_hinge" were considered for the loss function; three levels of "Tol", five levels of "Penal" were similarly set as that of CluSL. The RAM algorithm was run for 10 repetitions under each parameter setting. After calculation, the best prediction accuracy (BPA) among 10 repetitions, the corresponding the best tuning parameter, training accuracy (TA) as well as average computational time (Avg_time) of both CluSL and SVC are displayed in Table 8. More detail results of CluSL and SVC under different parameter settings are displayed in Table 21- Table 25 (CluSL) and Table 26- Table 30 (SVC) of Appendix C.2.

Table 8 Summary of CluSL and SVC results among five UCI datasets in Experiment 5

Dataset	Model	BPA	Best Tuning Parameter	TA	Avg_time(s)
SPF	CluSL	0.643	K=2, $\rho = 2$	0.743	8.442
	SVC	0.676	loss=squared_hinge, Tol=1e-1, Penal=1	0.465	0.716
WIL	CluSL	0.987	K=2, $\rho = 6$	0.981	2.844
	SVC	0.976	loss=squared_hinge, Tol=1e-5, Penal=40	0.974	0.086
Yeast	CluSL	0.609	K=3, $\rho = 6$	0.630	5.307
	SVC	0.578	loss=squared_hinge, Tol=1e-1, Penal=20	0.603	0.365
CMC	CluSL	0.560	K=3, $\rho = 2$	0.584	2.559
	SVC	0.526	loss=squared_hinge, Tol=1e-5, Penal=40	0.534	0.191
WW_5	CluSL	0.560	K=4, $\rho = 8$	0.538	17.807
	SVC	0.559	loss=squared_hinge, Tol=1e-5, Penal=40	0.511	1.352

We also see from Table 8 that in most of the datasets (except for SPF), the prediction accuracy CluSL is higher than those of the SVC results under the best parameter setting. This demonstrates that the integration clustering with SL is promising to enhance the prediction accuracy in the classification problems. The computational time of the RAM algorithm is longer since it takes several iterations to converge. As shown in the results of Appendix C.2, the computational time of the RAM algorithm increases as the number of clusters K increases. However, one might benefit from longer computational time to obtain higher prediction accuracy in some cases such as "Yeast" and "CMC" datasets. Hence, by varying parameter settings and a trade-off of the prediction accuracy and computational time, one can flexibly obtain desired learning results of CluSL according to specific requirements.

Experiment 6: In this experiment, we compare the prediction accuracy, training accuracy as well as training time of CluSL with CNNs using three image datasets: Magnetic-tile-defect-datasets (MT)[5], MNIST handwritten digits datasets (MNIST) [6] and CIFAR-10 datasets (CIFAR-10) [7]. Due to the inconsistency of image size in each class, we preprocessed the MT dataset by uniformly resizing all the images into 50×50 pixels. The dataset was then split into 70% training samples and 30% testing samples. For the CIFAR-10 dataset, we converted all the colored images into the gray ones, by scaling each pixel into the range [0, 1]. For both MNIST and CIFAR-10 datasets, due to out-of-memory issue, we selected 5000 images in their training sample pools as training samples and used the given testing samples. Detailed information of three datasets can be found in Table 9.

Table 9 Details of image datasets used in Experiment 6

Dataset	Features	Class	Training samples	Testing samples
MT	50×50	6	345	147
MNIST	28×28	10	5000	10000
CIFAR-10	32×32	10	5000	10000

In the CluSL framework, the number of clusters K and tuning parameter ρ were chosen from 2, 3, 4 and 0.1, 0.3, 0.5, 0.7, 0.9, respectively for three datasets. For CNNs, we set two 2D convolution layers, two MaxPooling layers, and two densely-connected NN layers. In addition, we allowed varying both of the epoch (EP) and kernel size (KS), ranging from 20, 30, 40 and (4, 4), (8, 8), (12, 12), respectively. All the CNNs experiments were implemented using the Keras Sequential model in Python deep learning library. Both RAM algorithm and CNNs were repeated for 5 times. For each combination of K and ρ in CluSL, the best prediction accuracy (BPA), the training accuracy (TA), and average training time (Avg_time) of the five repetitions are displayed in Table 10. The table also

[5] Available at `https://github.com/abin24/Magnetic-tile-defect-datasets`.

[6] Available at `http://yann.lecun.com/exdb/mnist/`.

[7] Available at `http://www.cs.toronto.edu/~kriz/cifar.html`.

shows the best prediction result among all the parameter settings of CNNs. More detailed experimental results of CluSL and CNNs can be found in Appendix C.3. Figure 6 illustrates the the average testing accuracy of both CluSL and CNNs with the different CluSL parameters k and ρ.

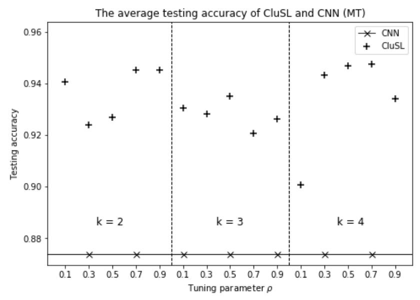
Table 10 Summary of CluSL and CNNs results among three image datasets in Experiment 6

Dataset: MT	CluSL			CNNs
	K = 2 ($\rho = 0.1$)	K = 3 ($\rho = 0.3$)	K = 4 ($\rho = 0.3$)	KS = (12, 12), EP = 20
BPA	0.973	0.973	0.966	0.979
TA	0.980	0.980	0.983	0.974
Avg_time(s)	58.882	109.127	206.764	44.052
Dataset: MNIST	CluSL			CNNs
	K = 2 ($\rho = 0.7$)	K = 3 ($\rho = 0.5$)	K = 4 ($\rho = 0.7$)	KS = (8, 8), EP = 20
BPA	0.975	0.972	0.968	0.981
TA	1.000	1.000	1.000	1.000
Avg_time(s)	277.002	351.053	508.856	133.326
Dataset: CIFAR-10	CluSL			CNNs
	K = 2 ($\rho = 0.5$)	K = 3 ($\rho = 0.7$)	K = 4 ($\rho = 0.9$)	KS = (4, 4), EP = 20
BPA	0.498	0.472	0.451	0.537
TA	0.991	0.999	1.000	0.957
Avg_time(s)	330.988	442.023	624.392	108.853

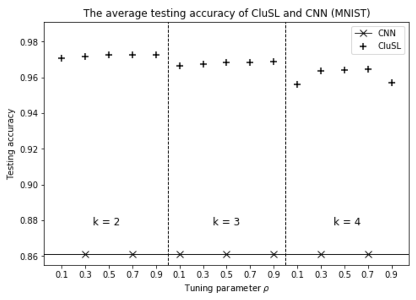
It is seen from Table 10 that for BPA, CNNs perform slightly better than CluSL, meaning that $K = 1$ might be the best choice for these instances. This phenomenon might be because the sample sizes of these three datasets are not very large and CluSL might have an overfitting issue. The higher training accuracy of CluSL framework further demonstrates this observation. On the other hand, in Figure 6, we see that for the average testing accuracy, the CluSL framework with $K \geq 2$ yields better performance comparing with CNNs. This might be because reducing heterogeneity within different clusters helps generate more stable predictions. Finally, we notice that for the training time, CNNs are more efficient since the iterated clustering procedure in RAM algorithm usually takes longer time.

5.4. Numerical Illustration of F-CluSL

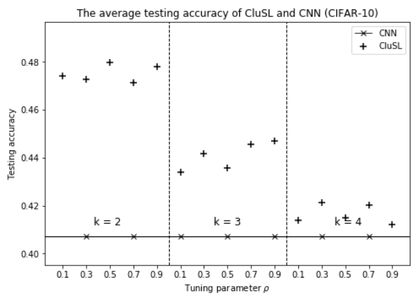
In this subsection, we tested the performance on image classification using the F-CluSL framework and F-RAM Algorithm 2 to extract critical features and learn classes simultaneously. There are two experiments. In the first Experiment 7, we used datasets with small-sized and simple images to illustrate the interpretability of F-CluSL as the increase



(a) MT



(b) MNIST



(c) CIFAR-10

Figure 6 The average testing accuracy of CluSL and CNNs in Experiment 6.

of K. Next, in Experiment 8, we tested the performance of F-CluSL using more complex and larger-sized images. Similarly, the minimization over loss function each iteration of F-RAM Algorithm 2 given the clustering result was solved by sklearn.svm.LinearSVC.

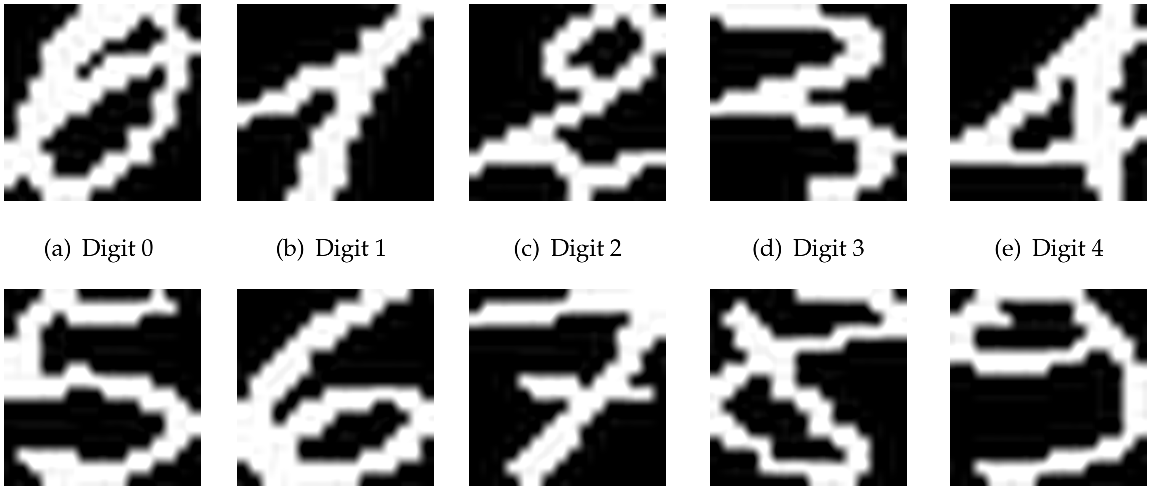
Experiment 7: In this experiment, we used small-sized images to show both the classification performance and interpretability of F-CluSL. For illustration purpose and the observation that the neighboring pixels are prone to belong to the same clusters, we set a large enough $\rho = 10^3$, and we did not tune ρ.

We compared our CluSL with the classical image classification method CNNs. Specifically, we focused on comparing the prediction accuracy, training accuracy as well as training time of F-CluSL with CNNs based on two UCI image datasets: Semeion Handwritten Digit (SHD) and Optical Recognition of Handwritten Digits (ORHD). For example, a set of handwritten digits 0-9 from SHD data set is shown in Figure 7. Similarly, all samples were split into 70% training samples and 30% testing samples. The detailed information of datasets is displayed in Table 11.

Table 11 Details of UCI image datasets used in Experiment 7

Dataset	Features	Class	Training samples	Testing samples
SHD	$16\times$ 16 gray scale values	10	1115	479
ORHD	Input matrix 8×8 range in 0..16	10	3823	1797

After inspection, for SHD and ORHD datasets, the number of clusters K in F-CluSL was chosen to be 8, 12, 16, 20, 24, and 8, 16, 24, 32, 40, respectively. For the CNNs, we used similar sequential model explained in Experiment 6. All of its optimizer (OPT), epoch (EP), and kernel size (KS) were allowed to vary, shown in Table 12.



(a) Digit 0 (b) Digit 1 (c) Digit 2 (d) Digit 3 (e) Digit 4

(f) Digit 5 (g) Digit 6 (h) Digit 7 (i) Digit 8 (j) Digit 9

Figure 7 Image samples of SHD dataset in Experiment 7

Table 12 The parameters for the CNNs in Experiment 7

Dataset	Parameter settings
SHD	OPT: rmsprop, adagrad EP: 1, 3, 5, 7, 9; KS: (3,3), (5,5)
ORHD	OPT: rmsprop, adagrad EP: 1, 3, 5, 7, 9; KS: (1,1), (2,2)

For each choice of K in F-CluSL, the best prediction accuracy (BPA), training accuracy (TA), average training time (Avg_time) as well as the test accuracy of each digit in two cases are displayed in Table 13 and Table 14 respectively. Similarly, the learning results of CNNs for two datasets are shown in Table 15 and 16, respectively. We can see that for CluSL as the increase of K, although the training time increases, both training accuracy and testing accuracy are improved, yielding the better prediction performance than that of CNNs. CNNs takes shorter training time than F-CluSL but has a lower prediction accuracy.

Further, we illustrate the result of F-CluSL with a small number of clusters: $K=2$ and $K=3$. Pixels belong to the same cluster are plotted using the same color; see for example, Figure 8. As can be seen, when $K=2$, the extracted features already fit the simple digit as "0" very well. With the increase of K, the extracted features can gradually fit more complex digit, which essentially helps increase the prediction accuracy. For example, in Figure 9, the red part almost overlaps with "0", '2", "5","6", "9", and the middle blue part largely contains the digit "1","2", "7". Thus, the extracted features using clustering can

Table 13 Results of SHD using F-CluSL in Experiment 7

Performance	Number of clusters				
	$K=8$	$K=12$	$K=16$	$K=20$	$K=24$
BPA	0.697	0.818	0.839	0.875	0.889
TA	0.744	0.843	0.890	0.909	0.917
Avg_time(s)	6.827	10.307	14.497	18.082	20.523
Digit 0	0.833	0.833	0.896	0.938	0.958
Digit 1	0.735	0.878	0.837	0.898	0.878
Digit 2	0.583	0.708	0.854	0.896	0.875
Digit 3	0.708	0.833	0.854	0.875	0.875
Digit 4	0.792	0.833	0.792	0.875	0.896
Digit 5	0.792	0.917	0.833	0.854	0.896
Digit 6	0.729	0.813	0.917	0.938	0.917
Digit 7	0.563	0.708	0.813	0.771	0.854
Digit 8	0.717	0.804	0.826	0.870	0.891
Digit 9	0.521	0.854	0.771	0.833	0.854

Table 14 Results of ORHD using F-CluSL in Experiment 7

Performance	Number of clusters				
	$K=8$	$K=16$	$K=24$	$K=32$	$K=40$
BPA	0.771	0.885	0.923	0.924	0.939
TA	0.807	0.913	0.958	0.959	0.968
Avg_time(s)	8.054	11.403	19.721	26.156	29.361
Digit 0	0.933	0.961	0.978	0.983	0.994
Digit 1	0.786	0.841	0.852	0.907	0.901
Digit 2	0.638	0.921	0.932	0.966	0.994
Digit 3	0.836	0.787	0.902	0.913	0.891
Digit 4	0.845	0.950	0.939	0.956	0.961
Digit 5	0.758	0.951	0.951	0.984	0.984
Digit 6	0.956	0.950	0.972	0.961	0.978
Digit 7	0.939	0.939	0.927	0.888	0.916
Digit 8	0.425	0.793	0.833	0.810	0.862
Digit 9	0.578	0.761	0.939	0.867	0.911

help distinguish these digits with simple structures when K is small. Figure 10 shows that as the increase of K from 2 to 3, the red part can now represent the middle turning parts in "2", "3", "4", "5", "7", "8", "9" more precisely. In this case, pixels in turning part are identified as important factors impacting the classification result, which further help increase the training accuracy. Thus, as the number of clusters increases, the extracted features can fit more complex digits better, which is also consistent with the results displayed in Table 9. Finally, we remark that one can select the best number of K as a trade-off and accuracy and training time.

Experiment 8: We tested the performance of F-CluSL using larger-sized images: MT, MNIST, and CIFAR-10, which have been used in Experiment 6 (see Table 9 for the data

Table 15 Results of SHD by CNNs in Experiment 7

# of CNNs	BPA (rmsprop)					BPA (adagrad)				
	EP=1	EP=3	EP=5	EP=7	EP=9	EP=1	EP=3	EP=5	EP=7	EP=9
KS=(3,3)	0.203	0.280	0.370	0.326	0.491	0.273	0.357	0.486	0.695	0.681
KS=(5,5)	0.253	0.328	0.355	0.392	0.470	0.238	0.461	0.541	0.628	0.737
# of CNNs	**TA (rmsprop)**					**TA (adagrad)**				
	EP=1	EP=3	EP=5	EP=7	EP=9	EP=1	EP=3	EP=5	EP=7	EP=9
KS=(3,3)	0.169	0.239	0.341	0.348	0.502	0.160	0.362	0.518	0.664	0.706
KS=(5,5)	0.178	0.333	0.361	0.495	0.465	0.168	0.439	0.558	0.644	0.742
# of CNNs	**Avg_time (rmsprop) (s)**					**Avg_time (adagrad) (s)**				
	EP=1	EP=3	EP=5	EP=7	EP=9	EP=1	EP=3	EP=5	EP=7	EP=9
KS=(3,3)	1.791	2.078	2.614	3.555	3.910	3.440	3.765	4.735	4.849	5.437
KS=(5,5)	2.605	2.769	3.055	3.589	3.870	4.074	4.599	5.087	5.626	5.933

Table 16 Results of ORHD by CNNs in Experiment 7

# of CNNs	BPA (rmsprop)					BPA (adagrad)				
	EP=1	EP=3	EP=5	EP=7	EP=9	EP=1	EP=3	EP=5	EP=7	EP=9
KS=(1,1)	0.203	0.244	0.393	0.425	0.516	0.274	0.495	0.511	0.573	0.536
KS=(2,2)	0.215	0.329	0.436	0.525	0.560	0.364	0.567	0.664	0.707	0.675
# of CNNs	**TA (rmsprop)**					**TA (adagrad)**				
	EP=1	EP=3	EP=5	EP=7	EP=9	EP=1	EP=3	EP=5	EP=7	EP=9
KS=(1,1)	0.157	0.245	0.377	0.439	0.543	0.195	0.518	0.528	0.575	0.574
KS=(2,2)	0.149	0.313	0.442	0.538	0.592	0.284	0.582	0.712	0.728	0.715
# of CNNs	**Avg_time (rmsprop) (s)**					**Avg_time (adagrad) (s)**				
	EP=1	EP=3	EP=5	EP=7	EP=9	EP=1	EP=3	EP=5	EP=7	EP=9
KS=(1,1)	1.294	1.684	2.043	2.460	2.927	2.503	2.911	3.395	3.893	4.448
KS=(2,2)	1.800	2.082	2.401	2.687	2.990	3.045	3.425	3.754	4.124	4.525

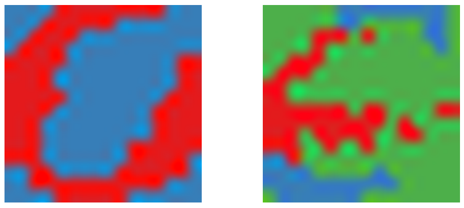

(a) $K=2$ (b) $K=3$

Figure 8 Examples of clustering results of SHD in Experiment 7

description). For three datasets, the number of clusters K in F-CluSL was chosen to be (800, 1000, 1200), (200, 400, 600) and (100, 200, 300), respectively. The tuning parameter ρ in three datasets was selected from 0.1, 0.3, 0.5, 0.7, 0.9. The parameter settings of CNNs were the same as those in Experiment 6. In F-RAM Algorithm 2, Step 3 was solved by Gurobi and sklearn.svm.LinearSVC when extracted feature matrix $\boldsymbol{m}$ was known. Given

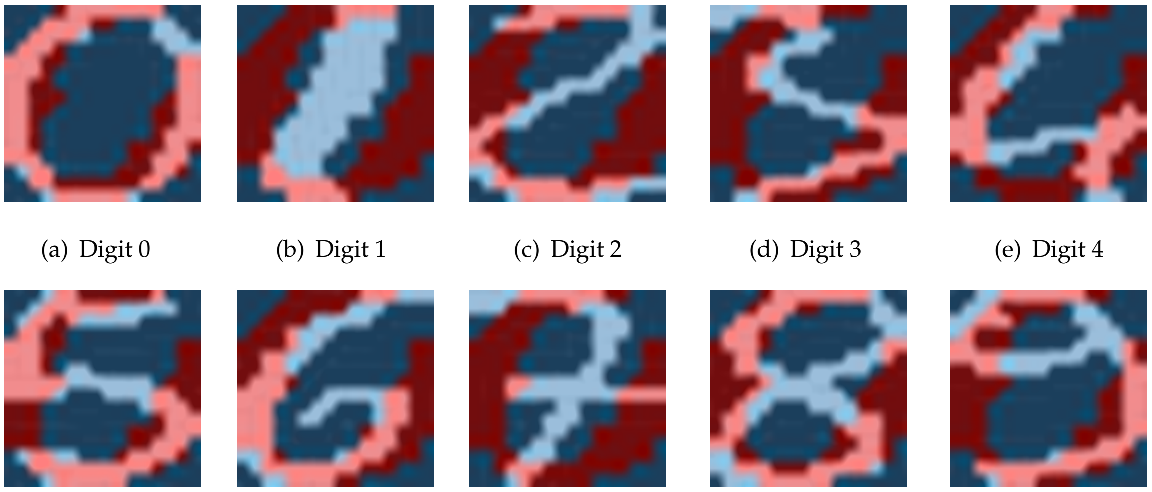

(a) Digit 0 (b) Digit 1 (c) Digit 2 (d) Digit 3 (e) Digit 4
(f) Digit 5 (g) Digit 6 (h) Digit 7 (i) Digit 8 (j) Digit 9

Figure 9 **Image fitting of SHD dataset in Experiment 7 ($K = 2$)**

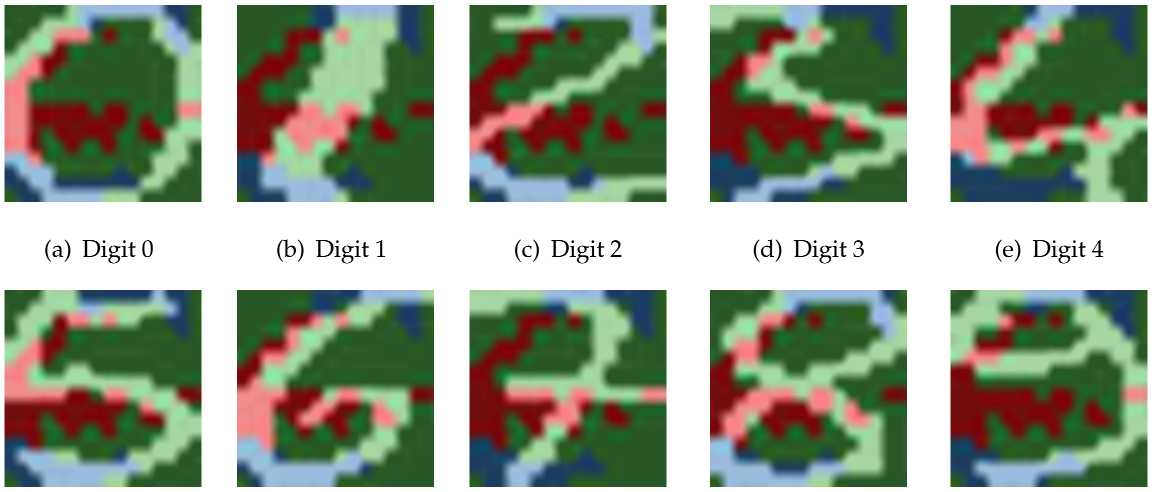

(a) Digit 0 (b) Digit 1 (c) Digit 2 (d) Digit 3 (e) Digit 4
(f) Digit 5 (g) Digit 6 (h) Digit 7 (i) Digit 8 (j) Digit 9

Figure 10 **Image fitting of SHD dataset in Experiment 7 ($K = 3$)**

the values of $\boldsymbol{\delta}$ and $\boldsymbol{\theta}$, Step 4 of F-RAM Algorithm 2 was solved by a simple gradient descent algorithm. Similarly, we repeated running F-RAM algorithm five times for each instance.

For each K, the best prediction accuracy (BPA), average prediction accuracy (Avg_PA), training accuracy (TA), and average training time (Avg_time) of three datasets are shown in Table 17 (ρ in bracket is the best tuned one for each K). More detailed results of F-CluSL can be found in Appendix C.4. Due to page limit, we put the results of the benchmark method (i.e.., CNNs) in Table 32 - Table 34 of Appendix C.3.

Table 17 Results of F-CluSL in Experiment 8

Dataset: MT	K = 800 (ρ = 0.1)	K = 1000 (ρ = 0.5)	K = 1200 (ρ = 0.1)
BPA	0.925	0.939	0.932
Avg_PA	0.868	0.890	0.860
TA	1.000	1.000	1.000
Avg_time(s)	232.847	257.834	309.765
Dataset: MNIST	K = 200 (ρ = 0.7)	K = 400 (ρ = 0.9)	K = 600 (ρ = 0.7)
BPA	0.889	0.883	0.878
Avg_PA	0.874	0.874	0.872
TA	1.000	1.000	1.000
Avg_time(s)	81.223	181.752	274.622
Dataset: CIFAR-10	K = 100 (ρ = 0.1)	K = 200 (ρ = 0.1)	K = 300 (ρ = 0.1)
BPA	0.213	0.232	0.224
Avg_PA	0.201	0.222	0.220
TA	1.000	1.000	1.000
Avg_time(s)	77.861	166.478	273.194

As can be seen from Table 17, Tables 32-34, although F-CluSL has higher TA, its best prediction performance is worse than with CNNs, especially for the dataset CIFAR-10. This might be due to the fact that dataset CIFAR-10 contains more complex and diverse images than other datasets. Thus, for datasets containing large-sized images, CNNs perform better at feature extraction with higher BPA. In addition, for the average prediction accuracy, we see that the performance of CNNs is more unstable than that of F-CLuSL. It is interesting to see that for more than half of parameter settings, their average prediction accuracies of CNNs are worse than those of F-CluSL. Therefore, although CNNs have higher BPA, F-CluSL can be more stable and have better average performances. As for the running time, since F-CluSL requires an iterated solution procedure, thus it often takes longer time than CNNs.

6. Conclusion

To exploit the inherent data structure as well as maintain the discriminative information in supervised learning, this paper provides a generic Cluster-aware Supervised Learning (CluSL) framework, which flexibly integrates the clustering of data points with the optimization of SL problems. By simultaneously grouping data and minimizing the training errors within each cluster, the proposed CluSL not only increases the prediction accuracy but also delivers interpretable results. Due to the nonconvexity of CluSL, a regularized alternating minimization (RAM) is proposed to solve the proposed formulation and is proven to always converge to the stationary point. We extend CluSL framework and RAM algorithm to the high-dimensional dataset, i.e., F-CluSL and F-RAM, where instead of clustering data points, we aim to cluster features within supervised learning. Our numerical results demonstrate that the proposed RAM and F-RAM algorithms can be effective and CluSL and F-CluSL can indeed be better than several benchmarks such as Random Forests (RF), Supper Vector Classification (SVC), Support Vector Regression (SVR), and CNNs in terms of prediction accuracy and interpretability. One possible future direction is to prove the approximation ratio of stationary solution obtained by the RAM or F-RAM algorithm, or derive the gap between the stationary solutions found by the proposed algorithms and the optimal one.

Acknowledgment

We appreciate the valuable comments from associated editor and two anonymous referees.

References

Aliaga, V., Ferrelli, F., and Piccolo, M. (2017). Regionalization of climate over the argentine pampas. *International Journal of Climatology*.

Arumugam, P. and Christy, V. (2018). Analysis of clustering and classification methods for actionable knowledge. *Materials Today: Proceedings*, 5(1, Part 1):1839 – 1845. International Conference on Processing of Materials, Minerals and Energy (July 29th 30th) 2016, Ongole, Andhra Pradesh, India.

Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. (2008). Alternating minimization and projection methods for nonconvex problems.

Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. (2010). Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-ojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457.

Bagirov, A. M., Ugon, J., and Mirzayeva, H. G. (2015). An algorithm for clusterwise linear regression based on smoothing techniques. *Optimization letters*, 9(2):375–390.

Beck and Amir (2015). On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *Siam Journal on Optimization*, 25(1):185–209.

Ben-Tal, A. and Nemirovski, A. (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, volume 2. Siam.

Boyd, S., Dattorro, J., et al. (2003). Alternating projections. *EE392o, Stanford University*.

Brègman, L. M. (1965). Finding the common point of convex sets by the method of successive projection. In *Doklady Akademii Nauk*, volume 162, pages 487–490. Russian Academy of Sciences.

Chen, Y., Jiang, H., Li, C., Jia, X., and Ghamisi, P. (2016). Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10):6232–6251.

Choi, E., Bahadori, M. T., Sun, J., Kulas, J., Schuetz, A., and Stewart, W. (2016). Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512.

Csiszar, I. and Tusnady, G. (1984). Information geometry and alternating minimization procedures. *Statistics & Decisions*, 1.

Cui, Z.-X. and Fan, Q. (2017). A nonconvex nonsmooth regularization method for compressed sensing and low rank matrix completion. *Digit. Signal Process.*, 62:101–111.

Czepiel, S. (2019). Maximum likelihood estimation of logistic regression models: Theory and implementation.

Dass, J., Sarin, V., and Mahapatra, R. N. (2019). Fast and communication-efficient algorithm for distributed support vector machine training. *IEEE Transactions on Parallel and Distributed Systems*, 30(5):1065–1076.

Day, N. E. (1969). Estimating the components of a mixture of normal distributions. *Biometrika*, 56(3):463–474.

Derisma, Silvana, M., and Imelda (2018). Optimization of neural network with genetic algorithm for breast cancer classification. In *2018 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 398–403.

Devijver, E. (2017). Model-based regression clustering for high-dimensional data: application to functional data. *Advances in Data Analysis and Classification*, 11(2):243–279.

Di Mari, R., Rocci, R., and Gattone, S. A. (2017). Clusterwise linear regression modeling with soft scale constraints. *International Journal of Approximate Reasoning*, 91:160–178.

Drusvyatskiy, D., Ioffe, A. D., and Lewis, A. S. (2015). Transversality and alternating projections for nonconvex sets. *Foundations of Computational Mathematics*, 15(6):1637–1651.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*.

Gallego, A.-J., Calvo-Zaragoza, J., Valero-Mas, J. J., and Rico-Juan, J. R. (2018). Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation. *Pattern Recognition*, 74:531–543.

Hasselblad, V. (1966). Estimation of parameters for a mixture of normal distributions. *Technometrics*, 8(3):431–444.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., Hinton, G. E., et al. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Jain, P., Netrapalli, P., and Sanghavi, S. (2012). Low-rank matrix completion using alternating minimization.

Khabia, Apeksha, M. B. C. (2014). A cluster based approach for classification of web results. volume 4, page 934.

Khadka, M., Paz, A., and Singh, A. (2018). Generalised clusterwise regression for simultaneous estimation of optimal pavement clusters and performance models. *International Journal of Pavement Engineering*, pages 1–13.

Kumar, C. S., George, K. K., Ramachandran, K., and Panda, A. (2015). Weighted cosine distance features for speaker verification. In *2015 Annual IEEE India Conference (INDICON)*, pages 1–5. IEEE.

Lai, M. J. and Mckenzie, D. (2020). Compressive sensing approach to cut improvement and local clustering. *SIAM J. Math. Data Science*.

Lai, M. J. and Varghese, A. (2017). On convergence of the alternating projection method for matrix completion and sparse recovery problems. *arXiv preprint arXiv:1711.02151*.

Lei Lei, S. K. (2016). Speaker recognition using wavelet cepstral coefficient, i-vector, and cosine distance scoring and its application for forensics. *Journal of Electrical and Computer Engineering*.

Lewis, A. S., Luke, D. R., and Malick, J. (2009). Local linear convergence for alternating and averaged nonconvex projections. *Foundations of Computational Mathematics*, 9(4):485–513.

Li, D., Deogun, J. S., Spaulding, W., and Shuart, B. (2004). Towards missing data imputation: A study of fuzzy k-means clustering method. In *Rough Sets and Current Trends in Computing*.

Liao, X., Li, H., and Carin, L. (2014). Generalized alternating projection for weighted-2,1 minimization with applications to model-based compressive sensing. *SIAM Journal on Imaging Sciences*, 7(2):797–823.

Liu, H., Han, J., Nie, F., and Li, X. (2017). Balanced clustering with least square regression. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Luo, Z. Q. and Tseng, P. (1993). Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46-47(1):157–178.

Ma, S., Song, X., and Huang, J. (2007). Supervised group lasso with applications to microarray data analysis. *BMC bioinformatics*, 8(1):60.

Mahajan, M., Nimbhorkar, P., and Varadarajan, K. (2009). The planar k-means problem is np-hard. In *International Workshop on Algorithms and Computation*, pages 274–285. Springer.

Mary, N. A. B. and Dharma, D. (2017). Coral reef image classification employing improved ldp for feature extraction. *Journal of Visual Communication and Image Representation*, 49:225–242.

Panigrahi, L., Verma, K., and Singh, B. K. (2019). Ultrasound image segmentation using a novel multi-scale gaussian kernel fuzzy clustering and multi-scale vector field convolution. *Expert Systems with Applications*, 115:486–498.

Park, Y. W., Jiang, Y., Klabjan, D., and Williams, L. (2017). Algorithms for generalized clusterwise linear regression. *INFORMS Journal on Computing*, 29(2):301–317.

Ren, D., Hui, M., Hu, N., and Zhan, T. (2018). A weighted sparse neighbor representation based on gaussian kernel function to face recognition. *Optik*, 167:7–14.

Rokach, L. and Maimon, O. (2005). Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer.

Rosso, G. (2014). Outliers emphasis on cluster analysis - the use of squared euclidean distance and fuzzy clustering to detect outliers in a dataset.

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons.

Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. (2013). A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245.

Späth, H. (1982). A fast algorithm for clusterwise linear regression. *Computing*, 29(2):175–181.

Stokes, M. D. and Deane, G. B. (2009). Automated processing of coral reef benthic images. *Limnology and Oceanography: Methods*, 7(2):157–168.

Suo, M., Zhu, B., Zhang, Y., An, R., and Li, S. (2018). Fuzzy bayes risk based on mahalanobis distance and gaussian kernel for weight assignment in labeled multiple attribute decision making. *Knowledge-Based Systems*, 152:26–39.

Tanner, J. and Wei, K. (2016). Low rank matrix completion by alternating steepest descent methods.

Tran, C. T., Zhang, M., Andreae, P., Xue, B., and Bui, L. T. (2018). Improving performance of classification on incomplete data using feature selection and clustering. *Applied Soft Computing*, 73:848 – 861.

Von Neumann, J. (1950). *Functional operators: Measures and integrals*, volume 1. Princeton University Press.

Vu, T. (2019). Plug-n-play alternating projection algorithm for large-scale security constraint optimal power flow. *arXiv preprint arXiv:1907.03173*.

Wang, B., Kong, Y., Zhang, Y., Liu, D., and Ning, L. (2019). Integration of unsupervised and supervised machine learning algorithms for credit risk assessment. *Expert Systems with Applications*.

Weston, J. and Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *ESANN*.

Yang, B., Wang, M., Xu, Z., and Zhang, T. (2018). Streaming algorithm for big data logistic regression. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2940–2950.

Yue, J., Zhao, W., Mao, S., and Liu, H. (2015). Spectral–spatial classification of hyperspectral images using deep convolutional neural networks. *Remote Sensing Letters*, 6(6):468–477.

Zhang, B. (2003). Regression clustering. In *Third IEEE International Conference on Data Mining*, pages 451–458.

Zhang, C., Qin, Y., Zhu, X., Zhang, J., and Zhang, S. (2006). Clustering-based missing value imputation for data preprocessing. In *2006 4th IEEE International Conference on Industrial Informatics*, pages 1081–1086.

Zhang, S., Zhang, J., Zhu, X., Qin, Y., and Zhang, C. (2008). Missing value imputation based on data clustering. *Trans. Computational Science*, 1:128–138.

Zhao, W. and Du, S. (2016). Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8):4544–4554.

Zhao, Y., Wang, P.-H., Li, Y.-G., and Li, M.-Y. (2018). Fuzzy weighted c-harmonic regressions clustering algorithm. *Soft Comput.*, 22(14):4595–4611.

Zheng, Y. and Hong, L. (2018). The model of wind power short-term prediction based on artificial fish swarm algorithm of support vector machine. In *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, pages 570–574.

Zhu, Z. and Li, X. (2018). Convergence analysis of alternating nonconvex projections. *arXiv preprint arXiv:1802.03889*.

Zhu, Z., Li, Y., and Kong, N. (2012). Clusterwise linear regression with the least sum of absolute deviations–an mip approach. *International Journal of Operations Research*, 9(3):162–172.

Appendix A. Non-convergent Example of Basic AM Algorithm

Example 1 Let us consider a cluster-aware linear regression problem with four samples: $\{(-2,0),(0,0),(0.5,0),(1.5,0)\}$. Let us choose the negative log-likelihood loss function ℓ^1 and Euclidean dissimilarity function S^1. Suppose that there are $K=2$ clusters of the data and the lower bound of standard deviation σ_k is $\underline{\boldsymbol{\sigma}}=1$ for each $k\in[2]$, and in (1), the tuning parameter $\rho=1$. Specifically, CluSL (1) now becomes:

$$\left\{\min_{\boldsymbol{\beta},\boldsymbol{\sigma},\boldsymbol{\delta}}\sum_{i=1}^{4}\sum_{k=1}^{2}\delta_{ik}\left[\ln\sigma_k+\frac{\delta_{ik}(y_i-\boldsymbol{\beta}_k^\top\boldsymbol{x}_i)^2}{2\sigma_k^2}+\|\boldsymbol{x}_i-\boldsymbol{m}_k\|^2\right]:\boldsymbol{\sigma}\geq 1,\boldsymbol{\delta}\in\Delta\right\} \tag{27}$$

The iterations of the basic AM algorithm steps are illustrated in Figure 11, and the results are displayed in Figure 12. We indeed see that although the sequence of the objective function values of basic AM algorithm converges, the sequence of its clustering solutions does not.

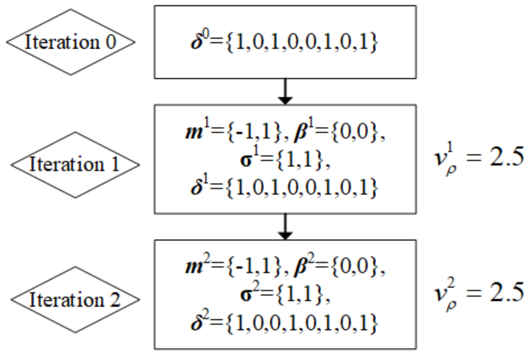
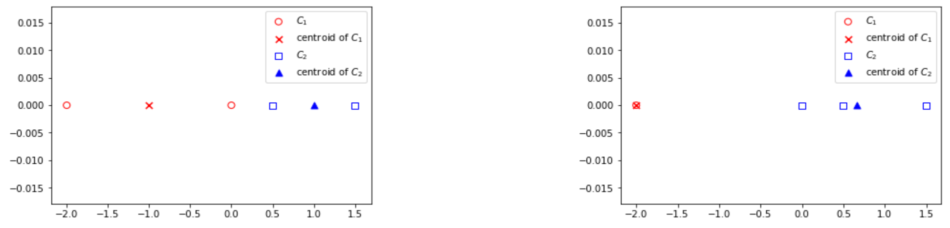

Iteration 0
$\boldsymbol{\delta}^0$={1,0,1,0,0,1,0,1}
Iteration 1
$\boldsymbol{m}^1$={-1,1}, $\boldsymbol{\beta}^1$={0,0}, $\boldsymbol{\sigma}^1$={1,1}, $\boldsymbol{\delta}^1$={1,0,1,0,0,1,0,1}
$v_\rho^1=2.5$
Iteration 2
$\boldsymbol{m}^2$={-1,1}, $\boldsymbol{\beta}^2$={0,0}, $\boldsymbol{\sigma}^2$={1,1}, $\boldsymbol{\delta}^2$={1,0,0,1,0,1,0,1}
$v_\rho^2=2.5$

Figure 11 **Iterations of basic AM in problem (27)**



(a) Result 1 (b) Result 2

Figure 12 **Non-convergent clustering results of problem (27)**

Appendix B. Proofs for F-CluSL Framework

B.1 Proof of Lemma 2

Lemma 2 *The sequence of the objective values* $\{R_\rho((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t)\}_t$ *of F-RAM Algorithm 2 is monotonically non-increasing and bounded from below, and hence converges.*

Proof: In Algorithm 2, at $(t+1)$th iteration, given the previous extracted features in all clusters $(\boldsymbol{m}^F)^t$, the fact that $\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}\right) \in \arg\min_{\boldsymbol{\theta}^F, \boldsymbol{\delta}^F \in \Delta_F} R_\rho\left(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, (\boldsymbol{m}^F)^t\right)$ implies that

$$R_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^t\right) \leq R_\rho\left((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t\right), \tag{28}$$

and $(\boldsymbol{m}^F)^{t+1} \in \arg\min_{\boldsymbol{m}^F} H_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, \boldsymbol{m}^F, (\boldsymbol{m}^F)^t\right)$ implies that

$$\begin{aligned} H_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^{t+1}, (\boldsymbol{m}^F)^t\right) &\leq H_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^t, (\boldsymbol{m}^F)^t\right) \\ &:= R_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^t\right), \end{aligned} \tag{29}$$

which further implies that

$$R_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^{t+1}\right) \leq R_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^t\right) \tag{30}$$

since $\|(\boldsymbol{m}^F)^{t+1} - (\boldsymbol{m}^F)^t\|_{1,1} \geq 0$.

Summing up (28) and (30) yields

$$R_\rho\left((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t\right) \geq R_\rho\left((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^{t+1}\right),$$

which implies that the sequence of output objective values $\left\{R_\rho\left((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t\right)\right\}_t$ of F-RAM Algorithm 2 is monotone non-increasing. On the other hand, according to the definition, $R_\rho\left((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t\right) \geq 0$ for all t. Hence, the monotone convergence theorem implies that the sequence $\left\{R_\rho\left((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t\right)\right\}_t$ is indeed convergent. Q.E.D.

B.2 Proof of Theorem 3

Theorem 3 *Suppose that* $\left\{\left((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t\right)\right\}_t$ *denotes the iterated solution sequence of F-RAM Algorithm 2. Then the following two limits exist:*

(i) $\lim_{t\to\infty}(\boldsymbol{m}^F)^t = (\boldsymbol{m}^F)^\star$ *and*

(ii) $\lim_{t\to\infty} R_\rho((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t) = R_\rho((\boldsymbol{\theta}^F)^\star, (\boldsymbol{\delta}^F)^\star, (\boldsymbol{m}^F)^\star)$, *where* $\left((\boldsymbol{\theta}^F)^\star, (\boldsymbol{\delta}^F)^\star\right) \in \arg\min_{\boldsymbol{\theta}^F, \boldsymbol{\delta}^F \in \Delta_F} R_\rho\left(\boldsymbol{\theta}^F, \boldsymbol{\delta}^F, (\boldsymbol{m}^F)^\star\right)$.

Proof: We first prove part (i). According to Lemma 2, the sequence of objective values $\{R_\rho((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t)\}_t$ of F-RAM Algorithm 2 is monotone non-increasing and convergent. Thus, for any $\sigma > 0$, there exists a positive integer T such that

$$R_\rho((\boldsymbol{\theta}^F)^{t_1}, (\boldsymbol{\delta}^F)^{t_1}, (\boldsymbol{m}^F)^{t_1}) - R_\rho((\boldsymbol{\theta}^F)^{t_2}, (\boldsymbol{\delta}^F)^{t_2}, (\boldsymbol{m}^F)^{t_2}) \le \sigma\epsilon$$

for any $T \le t_1 < t_2$.

By (28) and (29) in the proof of Lemma 2, for any t, we have

$$R_\rho((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^{t+1}) + \epsilon\|(\boldsymbol{m}^F)^{t+1} - (\boldsymbol{m}^F)^t\|_{1,1} \le R_\rho((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^t) \quad (31)$$

$$R_\rho((\boldsymbol{\theta}^F)^{t+1}, (\boldsymbol{\delta}^F)^{t+1}, (\boldsymbol{m}^F)^t) \le R_\rho((\boldsymbol{\theta}^F)^t, (\boldsymbol{\delta}^F)^t, (\boldsymbol{m}^F)^t). \quad (32)$$

Summing up (31) and (32) from $t = t_1$ to $t = t_2 - 1$ yields

$$\epsilon \sum_{t=t_1}^{t_2-1} \|(\boldsymbol{m}^F)^{t+1} - (\boldsymbol{m}^F)^t\|_{1,1} \le R_\rho((\boldsymbol{\theta}^F)^{t_1}, (\boldsymbol{\delta}^F)^{t_1}, (\boldsymbol{m}^F)^{t_1}) - R_\rho((\boldsymbol{\theta}^F)^{t_2}, (\boldsymbol{\delta}^F)^{t_2}, (\boldsymbol{m}^F)^{t_2}) \le \sigma\epsilon \quad (33)$$

Using triangle inequality, the above inequality is further reduced to

$$\|(\boldsymbol{m}^F)^{t_2} - (\boldsymbol{m}^F)^{t_1}\|_{1,1} \le \sigma$$

which implies that the sequence of extracted feature matrices $\{(\boldsymbol{m}^F)^t\}_t$ is Cauchy. Thus, there exists a $(\boldsymbol{m}^F)^\star$ such that $\lim_{t\to\infty}(\boldsymbol{m}^F)^t = (\boldsymbol{m}^F)^\star$.

Part (ii) follows by the fact that $\lim_{t\to\infty}(\boldsymbol{m}^F)^t = (\boldsymbol{m}^F)^\star$ and the observation that in F-RAM Algorithm 2, once the extracted features $(\boldsymbol{m}^F)^\star$ are determined, then the feature clustering $(\boldsymbol{\delta}^F)^\star$ and estimator $(\boldsymbol{\theta}^F)^\star$ will be also determined. Q.E.D.

Appendix C. Additional Results of Numerical Experiments

C.1 Additional Results of Experiment 3

The best testing MSE and its training time, average testing MSE, and average training time of three UCI datasets are displayed in Table 18-20, which are denoted as Testing MSE (best), Training Time (best), Testing MSE (average), and Training Time (average), respectively.

C.2 Results of Experiment 5

In each parameter setting, the best prediction accuracy (BPA), corresponding training accuracy (TA) and average time among 10 repetitions are shown in Table 21- Table 25 (CluSL) and Table 26- Table 30 (SVC).

Table 18 Results of Real estate valuation using CluSL in Experiment 3

# of Clusters	Testing MSE (best)			Testing MSE (average)		
	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$
$K=5$	0.114	0.165	0.261	0.152	0.249	0.323
$K=10$	0.103	0.159	0.178	0.130	0.198	0.264
$K=15$	0.096	0.139	0.258	0.123	0.181	0.283
$K=20$	0.081	0.142	0.174	0.120	0.173	0.272
# of Clusters	Training Time (best)			Training Time (average)		
	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$
$K=5$	0.474	0.491	0.479	0.480	0.496	0.513
$K=10$	0.964	1.007	1.069	0.929	0.997	0.990
$K=15$	1.517	1.559	1.576	1.434	1.504	1.495
$K=20$	2.157	2.205	2.203	1.992	2.080	2.054

Table 19 Results of red wine equality using CluSL in Experiment 3

# of Clusters	Testing MSE (best)			Testing MSE (average)		
	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$
$K=5$	0.267	0.433	0.618	0.339	0.494	0.660
$K=10$	0.230	0.357	0.498	0.264	0.395	0.570
$K=15$	0.198	0.294	0.537	0.227	0.360	0.583
$K=20$	0.182	0.260	0.480	0.204	0.340	0.558
# of Clusters	Training Time (best)			Training Time (average)		
	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$
$K=5$	2.327	2.479	2.440	2.419	2.591	2.545
$K=10$	4.961	5.047	5.023	5.056	5.161	5.129
$K=15$	8.130	8.193	8.180	8.339	8.308	8.330
$K=20$	11.791	11.855	11.992	11.990	12.141	12.141

Table 20 Results of white wine equality using CluSL in Experiment 3

# of Clusters	Testing MSE (best)			Testing MSE (average)		
	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$
$K=5$	0.264	0.361	0.532	0.288	0.397	0.561
$K=10$	0.178	0.331	0.506	0.212	0.361	0.556
$K=15$	0.182	0.295	0.473	0.199	0.332	0.520
$K=20$	0.160	0.275	0.478	0.176	0.304	0.510
# of Clusters	Training Time (best)			Training Time (average)		
	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$	$\rho=0.1$	$\rho=0.2$	$\rho=0.5$
$K=5$	7.339	7.270	7.222	7.550	7.484	7.466
$K=10$	16.296	16.126	16.231	16.536	16.483	16.506
$K=15$	25.380	25.702	26.055	26.189	26.271	26.500
$K=20$	37.115	37.201	37.204	37.859	37.874	37.714

C.3 Results of Experiment 6

In each parameter setting, the best prediction accuracy (BPA), corresponding training accuracy (TA) and average time (Avg_time) of CluSL among 5 repetitions are shown in

Table 21 Results of SPF using CluSL in Experiment 5

# of Clusters	BPA				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	0.696	0.676	0.688	0.696	0.671
$K = 2$	0.624	0.643	0.643	0.630	0.636
$K = 3$	0.597	0.628	0.624	0.624	0.623
$K = 4$	0.599	0.628	0.631	0.628	0.626
$K = 5$	0.642	0.638	0.642	0.640	0.631
# of Clusters	TA				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	0.701	0.690	0.684	0.710	0.717
$K = 2$	0.939	0.743	0.712	0.739	0.741
$K = 3$	0.982	0.760	0.756	0.739	0.753
$K = 4$	0.985	0.786	0.777	0.780	0.770
$K = 5$	0.996	0.800	0.786	0.784	0.798
# of Clusters	Avg_time(s)				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	2.205	1.861	1.874	1.887	1.899
$K = 2$	15.110	8.442	9.392	8.556	9.200
$K = 3$	19.864	16.609	15.581	16.885	15.756
$K = 4$	23.835	17.574	16.914	18.020	17.283
$K = 5$	37.189	21.180	20.767	20.116	21.891

Table 22 Results of WIL using CluSL in Experiment 5

# of Clusters	BPA				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	0.978	0.978	0.978	0.977	0.978
$K = 2$	0.973	0.985	0.985	0.987	0.987
$K = 3$	0.973	0.985	0.985	0.985	0.985
$K = 4$	0.980	0.985	0.985	0.985	0.985
$K = 5$	0.970	0.987	0.987	0.985	0.985
# of Clusters	TA				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	0.974	0.974	0.974	0.973	0.978
$K = 2$	0.994	0.984	0.983	0.981	0.981
$K = 3$	0.997	0.986	0.984	0.982	0.974
$K = 4$	0.999	0.987	0.985	0.985	0.984
$K = 5$	1.000	0.990	0.987	0.986	0.984
# of Clusters	Avg_time(s)				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	0.591	0.514	0.524	0.471	0.556
$K = 2$	2.105	2.042	2.470	2.844	2.915
$K = 3$	3.857	3.019	2.995	2.304	3.588
$K = 4$	2.297	5.805	5.547	5.832	5.646
$K = 5$	5.914	4.794	7.924	5.808	8.068

Table 23 **Results of Yeast using CluSL in Experiment 5**

# of Clusters	BPA				
	$\rho=0$	$\rho=2$	$\rho=4$	$\rho=6$	$\rho=8$
$K=1$	0.591	0.589	0.591	0.587	0.587
$K=2$	0.580	0.576	0.591	0.596	0.591
$K=3$	0.558	0.578	0.598	0.609	0.598
$K=4$	0.551	0.578	0.607	0.585	0.598
$K=5$	0.558	0.569	0.589	0.594	0.598
# of Clusters	**TA**				
	$\rho=0$	$\rho=2$	$\rho=4$	$\rho=6$	$\rho=8$
$K=1$	0.593	0.598	0.585	0.598	0.590
$K=2$	0.839	0.694	0.625	0.623	0.625
$K=3$	0.927	0.757	0.675	0.630	0.636
$K=4$	0.954	0.790	0.648	0.643	0.647
$K=5$	0.968	0.769	0.676	0.662	0.639
# of Clusters	**Avg_time(s)**				
	$\rho=0$	$\rho=2$	$\rho=4$	$\rho=6$	$\rho=8$
$K=1$	1.515	1.034	1.053	1.052	1.039
$K=2$	7.064	4.310	4.135	4.499	4.250
$K=3$	9.027	6.217	5.248	5.307	5.110
$K=4$	13.287	7.902	7.622	7.438	7.022
$K=5$	14.130	11.031	11.412	10.894	10.677

Table 24 **Results of CMC using CluSL in Experiment 5**

# of Clusters	BPA				
	$\rho=0$	$\rho=2$	$\rho=4$	$\rho=6$	$\rho=8$
$K=1$	0.531	0.537	0.522	0.531	0.544
$K=2$	0.531	0.549	0.551	0.542	0.549
$K=3$	0.497	0.560	0.558	0.558	0.558
$K=4$	0.492	0.556	0.556	0.560	0.558
$K=5$	0.476	0.551	0.551	0.558	0.560
# of Clusters	**TA**				
	$\rho=0$	$\rho=2$	$\rho=4$	$\rho=6$	$\rho=8$
$K=1$	0.532	0.535	0.526	0.540	0.546
$K=2$	0.843	0.613	0.566	0.566	0.580
$K=3$	0.984	0.584	0.589	0.596	0.580
$K=4$	0.996	0.616	0.593	0.597	0.581
$K=5$	0.997	0.678	0.612	0.602	0.606
# of Clusters	**Avg_time(s)**				
	$\rho=0$	$\rho=2$	$\rho=4$	$\rho=6$	$\rho=8$
$K=1$	0.463	0.469	0.475	0.521	0.487
$K=2$	3.162	1.493	1.917	1.947	1.968
$K=3$	6.894	2.559	2.545	3.114	2.568
$K=4$	5.232	3.199	3.058	3.235	3.257
$K=5$	4.238	6.226	5.634	5.499	6.319

Table 25 Results of WW_5 using CluSL in Experiment 5

# of Clusters	BPA				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	0.558	0.558	0.558	0.558	0.561
$K = 2$	0.534	0.548	0.543	0.544	0.546
$K = 3$	0.500	0.543	0.556	0.548	0.555
$K = 4$	0.500	0.552	0.540	0.543	0.557
$K = 5$	0.504	0.537	0.556	0.548	0.551
# of Clusters	TA				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	0.502	0.539	0.527	0.533	0.507
$K = 2$	0.797	0.545	0.536	0.530	0.531
$K = 3$	0.922	0.563	0.560	0.552	0.555
$K = 4$	0.968	0.599	0.562	0.557	0.538
$K = 5$	0.975	0.575	0.572	0.553	0.545
# of Clusters	Avg_time(s)				
	$\rho = 0$	$\rho = 2$	$\rho = 4$	$\rho = 6$	$\rho = 8$
$K = 1$	6.378	2.926	2.917	2.896	2.951
$K = 2$	19.139	16.248	12.952	8.854	13.517
$K = 3$	51.477	14.153	14.607	11.090	14.702
$K = 4$	40.065	21.809	21.691	17.602	17.807
$K = 5$	56.156	39.592	35.018	25.434	30.424

Table 26 Results of SPF using SVC model in Experiment 5

# of Clusters	BPA (hinge)			BPA (squared_hinge)		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.581	0.561	0.575	0.623	0.621	0.617
Penal=10	0.587	0.571	0.576	0.635	0.631	0.650
Penal=20	0.580	0.578	0.609	0.496	0.645	0.676
Penal=30	0.614	0.604	0.631	0.633	0.569	0.616
Penal=40	0.600	0.602	0.588	0.623	0.506	0.564
# of Clusters	TA (hinge)			TA (squared_hinge)		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.128	0.233	0.204	0.298	0.452	0.446
Penal=10	0.323	0.326	0.333	0.616	0.457	0.453
Penal=20	0.400	0.383	0.354	0.485	0.465	0.465
Penal=30	0.683	0.441	0.378	0.472	0.492	0.471
Penal=40	0.764	0.484	0.417	0.474	0.459	0.483
# of Clusters	Avg_time (hinge) (s)			Avg_time (squared_hinge) (s)		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.688	0.685	0.688	0.733	0.733	0.730
Penal=10	0.697	0.694	0.700	0.731	0.728	0.744
Penal=20	0.711	0.691	0.728	0.691	0.703	0.716
Penal=30	0.724	0.723	0.725	0.695	0.663	0.714
Penal=40	0.711	0.705	0.722	0.702	0.663	0.703

Table 27 Results of WIL using SVC model in Experiment 5

# of Clusters	BPA (hinge)			BPA (squared_hinge)		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.967	0.965	0.965	0.973	0.973	0.973
Penal=10	0.968	0.968	0.968	0.975	0.973	0.975
Penal=20	0.970	0.968	0.968	0.973	0.975	0.973
Penal=30	0.970	0.970	0.970	0.973	0.973	0.975
Penal=40	0.973	0.968	0.965	0.975	0.975	0.977
# of Clusters	**TA (hinge)**			**TA (squared_hinge)**		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.960	0.961	0.961	0.972	0.972	0.972
Penal=10	0.967	0.969	0.969	0.974	0.974	0.974
Penal=20	0.970	0.969	0.968	0.974	0.976	0.974
Penal=30	0.969	0.967	0.969	0.976	0.979	0.974
Penal=40	0.970	0.968	0.970	0.977	0.973	0.974
# of Clusters	**Avg_time (hinge) (s)**			**Avg_time (squared_hinge) (s)**		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.009	0.010	0.013	0.021	0.066	0.071
Penal=10	0.024	0.025	0.039	0.081	0.086	0.091
Penal=20	0.054	0.038	0.043	0.085	0.088	0.089
Penal=30	0.066	0.048	0.048	0.120	0.085	0.091
Penal=40	0.066	0.087	0.055	0.104	0.104	0.086

Table 28 Results of Yeast using SVC model in Experiment 5

# of Clusters	BPA (hinge)			BPA (squared_hinge)		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.460	0.515	0.519	0.567	0.569	0.569
Penal=10	0.470	0.467	0.499	0.569	0.567	0.571
Penal=20	0.540	0.467	0.508	0.578	0.576	0.571
Penal=30	0.512	0.506	0.533	0.578	0.569	0.578
Penal=40	0.503	0.483	0.517	0.555	0.567	0.573
# of Clusters	**TA (hinge)**			**TA (squared_hinge)**		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.516	0.543	0.542	0.608	0.609	0.607
Penal=10	0.526	0.523	0.531	0.607	0.608	0.611
Penal=20	0.544	0.489	0.534	0.603	0.604	0.606
Penal=30	0.543	0.515	0.533	0.601	0.598	0.609
Penal=40	0.519	0.500	0.519	0.586	0.597	0.591
# of Clusters	**Avg_time (hinge) (s)**			**Avg_time (squared_hinge) (s)**		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.026	0.029	0.034	0.210	0.304	0.343
Penal=10	0.181	0.151	0.186	0.367	0.379	0.353
Penal=20	0.261	0.210	0.217	0.365	0.365	0.382
Penal=30	0.231	0.244	0.258	0.388	0.376	0.384
Penal=40	0.277	0.251	0.257	0.366	0.383	0.395

Table 29 Results of CMC using SVC model in Experiment 5

# of Clusters	BPA (hinge)			BPA (squared_hinge)		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.460	0.456	0.433	0.501	0.501	0.501
Penal=10	0.476	0.444	0.467	0.503	0.501	0.497
Penal=20	0.458	0.420	0.465	0.503	0.492	0.506
Penal=30	0.458	0.438	0.444	0.485	0.472	0.488
Penal=40	0.442	0.460	0.447	0.506	0.447	0.526
# of Clusters	**TA (hinge)**			**TA (squared_hinge)**		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.496	0.487	0.483	0.523	0.526	0.525
Penal=10	0.497	0.486	0.484	0.530	0.525	0.530
Penal=20	0.500	0.480	0.496	0.525	0.521	0.518
Penal=30	0.484	0.468	0.498	0.509	0.507	0.540
Penal=40	0.475	0.500	0.491	0.498	0.526	0.534
# of Clusters	**Avg_time (hinge) (s)**			**Avg_time (squared_hinge) (s)**		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.011	0.010	0.009	0.049	0.158	0.210
Penal=10	0.051	0.050	0.049	0.189	0.185	0.185
Penal=20	0.085	0.084	0.088	0.238	0.189	0.241
Penal=30	0.130	0.117	0.110	0.180	0.188	0.273
Penal=40	0.151	0.157	0.196	0.194	0.193	0.190

Table 30 Results of WW_5 using SVC model in Experiment 5

# of Clusters	BPA (hinge)			BPA (squared_hinge)		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.493	0.459	0.499	0.545	0.543	0.543
Penal=10	0.504	0.449	0.491	0.547	0.539	0.549
Penal=20	0.536	0.451	0.511	0.502	0.542	0.519
Penal=30	0.503	0.537	0.463	0.559	0.525	0.532
Penal=40	0.494	0.499	0.485	0.491	0.533	0.451
# of Clusters	**TA (hinge)**			**TA (squared_hinge)**		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.460	0.445	0.472	0.536	0.536	0.536
Penal=10	0.482	0.443	0.467	0.533	0.536	0.541
Penal=20	0.495	0.441	0.467	0.526	0.511	0.522
Penal=30	0.489	0.480	0.459	0.511	0.507	0.529
Penal=40	0.492	0.493	0.482	0.498	0.496	0.504
# of Clusters	**Avg_time (hinge) (s)**			**Avg_time (squared_hinge) (s)**		
	Tol=1e-1	Tol=1e-3	Tol=1e-5	Tol=1e-1	Tol=1e-3	Tol=1e-5
Penal=1	0.140	0.111	0.198	0.872	1.116	1.084
Penal=10	0.592	0.598	0.581	1.292	1.946	1.158
Penal=20	0.852	0.833	0.820	1.243	1.441	1.202
Penal=30	0.981	0.966	0.972	1.352	1.239	1.231
Penal=40	1.053	1.214	1.061	1.229	1.255	1.372

Table 31. Similarly, the BPA, average prediction accuracy (Avg_PA), TA and Avg_time of CNNs are displayed in Table 32 - Table 34.

Table 31 Results of CluSL in Experiment 6

Dataset: MT	BPA			TA			Avg_time(s)		
	K = 2	K = 3	K = 4	K = 2	K = 3	K = 4	K = 2	K = 3	K = 4
$\rho = 0.1$	0.973	0.966	0.966	0.980	0.983	0.980	58.882	95.453	244.932
$\rho = 0.3$	0.973	0.973	0.966	0.983	0.980	0.983	62.143	109.127	206.764
$\rho = 0.5$	0.966	0.973	0.966	0.977	0.983	0.968	66.494	119.624	227.589
$\rho = 0.7$	0.966	0.973	0.966	0.980	0.983	0.983	70.039	129.729	288.588
$\rho = 0.9$	0.973	0.966	0.966	0.983	0.983	0.983	74.255	139.985	339.118
Dataset: MNIST	**BPA**			**TA**			**Avg_time(s)**		
	K = 2	K = 3	K = 4	K = 2	K = 3	K = 4	K = 2	K = 3	K = 4
$\rho = 0.1$	0.972	0.970	0.967	1.000	1.000	1.000	483.621	319.393	432.221
$\rho = 0.3$	0.974	0.970	0.967	1.000	1.000	1.000	509.271	327.050	457.168
$\rho = 0.5$	0.975	0.972	0.967	1.000	1.000	1.000	541.984	351.053	484.852
$\rho = 0.7$	0.975	0.972	0.968	1.000	1.000	1.000	277.002	364.993	508.856
$\rho = 0.9$	0.975	0.971	0.968	1.000	1.000	1.000	552.424	374.250	544.090
Dataset: CIFAR-10	**BPA**			**TA**			**Avg_time(s)**		
	K = 2	K = 3	K = 4	K = 2	K = 3	K = 4	K = 2	K = 3	K = 4
$\rho = 0.1$	0.490	0.455	0.432	1.000	0.999	1.000	610.472	401.564	515.440
$\rho = 0.3$	0.497	0.466	0.444	0.999	1.000	1.000	476.165	417.719	539.707
$\rho = 0.5$	0.498	0.465	0.447	0.991	0.998	1.000	330.988	430.251	566.032
$\rho = 0.7$	0.494	0.472	0.443	0.964	0.999	1.000	351.758	442.023	590.292
$\rho = 0.9$	0.496	0.468	0.451	1.000	0.987	1.000	700.368	457.304	624.392

Table 32 Results of MT using CNNs in Experiment 6

# of CNNs	BPA			Avg_PA		
	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)
EP = 20	0.966	0.973	0.979	0.905	0.800	0.818
EP = 30	0.966	0.973	0.973	0.812	0.495	0.664
EP = 40	0.959	0.959	0.966	0.905	0.958	0.661
# of CNNs	**TA**			**Avg_time(s)**		
	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)
EP = 20	0.983	0.977	0.974	16.619	28.853	44.052
EP = 30	0.983	0.983	0.980	25.087	42.868	64.588
EP = 40	0.980	0.983	0.983	35.255	56.911	82.783

C.4 Results of Experiment 8

In each parameter setting of F-CluSL, the best prediction accuracy (BPA), corresponding average prediction accuracy (Avg_PA), training accuracy (TA) and average time (Avg_time) among 5 repetitions are shown in Table 35 - Table 37.

Table 33 **Results of MNIST using CNNs in Experiment 6**

# of CNNs	BPA			Avg_PA		
	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)
EP = 20	0.979	0.981	0.976	0.803	0.979	0.449
EP = 30	0.979	0.979	0.976	0.978	0.627	0.800
EP = 40	0.980	0.980	0.977	0.802	0.628	0.804
# of CNNs	**TA**			**Avg_time(s)**		
	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)
EP = 20	1.000	1.000	1.000	86.666	133.326	150.881
EP = 30	1.000	1.000	1.000	126.394	199.909	226.393
EP = 40	1.000	1.000	1.000	168.058	266.923	304.604

Table 34 **Results of CIFAR-10 using CNNs in Experiment 6**

# of CNNs	BPA			Avg_PA		
	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)
EP = 20	0.537	0.457	0.100	0.434	0.304	0.100
EP = 30	0.521	0.457	0.404	0.349	0.312	0.219
EP = 40	0.532	0.478	0.405	0.437	0.248	0.218
# of CNNs	**TA**			**Avg_time(s)**		
	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)	KS = (4, 4)	KS = (8, 8)	KS = (12, 12)
EP = 20	0.957	0.923	0.097	108.853	169.687	216.932
EP = 30	0.993	0.966	0.969	165.320	255.002	326.413
EP = 40	1.000	0.999	0.985	218.412	340.654	437.635

Table 35 **Results of MT using F-CluSL in Experiment 8**

# of Clusters	BPA			Avg_PA		
	K = 800	K = 1000	K = 1200	K = 800	K = 1000	K = 1200
$\rho = 0.1$	0.925	0.932	0.932	0.868	0.808	0.860
$\rho = 0.3$	0.898	0.898	0.891	0.810	0.801	0.882
$\rho = 0.5$	0.925	0.939	0.891	0.818	0.890	0.842
$\rho = 0.7$	0.905	0.884	0.918	0.822	0.786	0.872
$\rho = 0.9$	0.918	0.912	0.912	0.864	0.899	0.856
# of Clusters	**TA**			**Avg_time(s)**		
	K = 800	K = 1000	K = 1200	K = 800	K = 1000	K = 1200
$\rho = 0.1$	1.000	1.000	1.000	233.151	258.750	309.765
$\rho = 0.3$	1.000	1.000	1.000	232.575	257.855	309.260
$\rho = 0.5$	1.000	1.000	1.000	232.847	257.834	308.483
$\rho = 0.7$	1.000	1.000	1.000	233.296	257.763	308.544
$\rho = 0.9$	1.000	1.000	1.000	231.879	258.423	308.984

C.5 Data and Codes of Numerical Experiments

All the data and codes of eight experiments are available at `https://github.com/cst950928/Numerical_experiment/tree/master`.

Table 36 Results of MNIST using F-CluSL in Experiment 8

# of Clusters	BPA K = 200	BPA K = 400	BPA K = 600	Avg_PA K = 200	Avg_PA K = 400	Avg_PA K = 600
$\rho = 0.1$	0.883	0.877	0.873	0.866	0.871	0.870
$\rho = 0.3$	0.882	0.879	0.877	0.869	0.871	0.871
$\rho = 0.5$	0.887	0.878	0.874	0.875	0.871	0.871
$\rho = 0.7$	0.889	0.880	0.878	0.874	0.873	0.872
$\rho = 0.9$	0.872	0.883	0.873	0.839	0.874	0.871
# of Clusters	**TA K = 200**	**TA K = 400**	**TA K = 600**	**Avg_time(s) K = 200**	**Avg_time(s) K = 400**	**Avg_time(s) K = 600**
$\rho = 0.1$	1.000	1.000	1.000	79.375	179.796	276.232
$\rho = 0.3$	1.000	1.000	1.000	79.928	183.465	272.969
$\rho = 0.5$	1.000	1.000	1.000	81.661	184.456	273.408
$\rho = 0.7$	1.000	1.000	1.000	81.223	184.380	274.622
$\rho = 0.9$	1.000	1.000	1.000	84.263	181.752	271.924

Table 37 Results of CIFAR-10 using F-CluSL in Experiment 8

# of Clusters	BPA K = 100	BPA K = 200	BPA K = 300	Avg_PA K = 100	Avg_PA K = 200	Avg_PA K = 300
$\rho = 0.1$	0.213	0.232	0.224	0.201	0.222	0.220
$\rho = 0.3$	0.201	0.228	0.221	0.197	0.220	0.216
$\rho = 0.5$	0.198	0.225	0.221	0.193	0.220	0.217
$\rho = 0.7$	0.189	0.225	0.219	0.188	0.219	0.216
$\rho = 0.9$	0.191	0.228	0.219	0.186	0.218	0.214
# of Clusters	**TA K = 100**	**TA K = 200**	**TA K = 300**	**Avg_time(s) K = 100**	**Avg_time(s) K = 200**	**Avg_time(s) K = 300**
$\rho = 0.1$	1.000	1.000	1.000	77.861	166.478	273.194
$\rho = 0.3$	1.000	1.000	1.000	79.599	164.871	275.416
$\rho = 0.5$	1.000	1.000	1.000	81.115	168.904	262.310
$\rho = 0.7$	1.000	1.000	1.000	79.519	170.266	272.500
$\rho = 0.9$	0.999	1.000	1.000	83.426	165.954	264.507