SIMULTANEOUS MODEL FOR CLUSTERING AND INTRA-GROUP FEATURE SELECTION

YUAN YANCHENG

(B.Sc., USTC, China)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MATHEMATICS

NATIONAL UNIVERSITY OF SINGAPORE

2019

Supervisor:

Professor Kim Chuan TOH

Examiners:

Professor Gong Yun, ZHAO

A/P Hui, JI

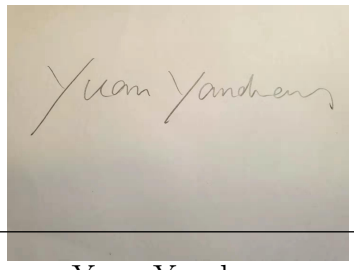
Professor Sunyoung, KIM, Ewha W. University

To my parents

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.



Yuan Yancheng

January 28, 2020

Acknowledgements

I would like to express my deepest gratitude to my supervisor Professor Toh Kim Chuan for his professional guidance and supervision during my Ph.D study. I am so lucky to have a supervisor like Professor Toh who are open-minded to support me to explore the topics I am interested in. In the summer of my first year Ph.D study, I worked with Professor Toh on the development of user-friendly interface for the highly efficient SDP solver SDPNAL+ and we co-authored a paper in the end, this is my first research paper in the area of optimization. After that, I shift my interests to machine learning. At the very beginning, I was a little bit nervous since I was not sure whether I could convince Professor Toh for the change of interests or not. Surprisingly, he fully supported and suggested me to work on the topic of convex clustering. Eventually, we come up with three papers which forms the main parts of this thesis. There is no doubt that this thesis will not be possible without the supports and guidance from Professor Toh.

My sincere thanks also goes to Professor Sun Defeng, who was my main supervisor before he moved to The Hong Kong Polytechnic University. He introduced me to the world of optimization and trained me for the theoretical foundation of optimization in a systematical way. He keeps supporting me even after he moves to Hong Kong. The supports I get from Professor Sun are much more beyond the

purely academic research.

I also would like to say thanks to the members in our optimization group: Dr. Li Xudong, Dr. Cui Ying, Dr. Chen Bo, Dr. Lam Xin Yee, Dr. Zhang Yangjing, Dr. Yang Lei, Miss Lin Meixia and Mr. Liang Ling. I learned a lot from all of them during the past a few years. In particular, I want to thank my co-author Lin Meixia for all the help she provided during the collaborations.

Last but not least, I want to express my deepest gratitude to my father, Yuan Shijian, for his constant and unconditional understanding and support throughout my whole life.

Contents

Summary

In this thesis, we focus on developing an algorithmic framework to perform clustering and intra-group feature selection simultaneously. In order to achieve this goal, we first study the convex clustering model and the exclusive lasso model in Chapter 3 and Chapter 4, respectively. Then, we study the new sparse convex clustering model in Chapter 5, which can achieve the goal of performing clustering and data point wise feature selection simultaneously.

In Chapter 3, we first analyze the recovery property of the general convex clustering model. More specifically, we propose a new mild sufficient condition which can guarantee the perfect recovery of the weighted convex clustering model. Our new theoretical results also include and improve the existing results for the convex clustering model. Then, we propose a highly efficient and robust semismooth smooth Newton based augmented Lagrangian method (SSNAL) to solve the weighted convex clustering model, which has been demonstrated to out-perform some existing state-of-art algorithms numerically, like the alternating direction method of multipliers (ADMM) and the alternating minimization algorithm (AMA).

In Chapter 4, we study the exclusive lasso regularizer which could enforce the intra-group level structured sparsity. We provide a rigorous proof for the closed-form solution to the proximal mapping $\text{Prox}_{\rho\|\cdot\|_1^2}(\cdot)$ and we derive an explicit form of the

corresponding Han-Sun (HS) generalized Jacobian $\partial_{\rm HS}{\rm Prox}_{\rho\|\cdot\|_1^2}(\cdot)$. Based on these results, we propose a dual Newton based preconditioned proximal point algorithm (PPDNA) to solve machine learning models with the exclusive lasso regularizer. The new proposed algorithm is more efficient and robust comparing to some popular first order methods, like ADMM, accelerated proximal gradient method (APG) and iterative least-square algorithm (ILSA).

Lastly, we focus on the sparse convex clustering model in Chapter 5. We demonstrate numerically that the new sparse convex clustering model is able to do the clustering and feature selection simultaneously on the high dimensional datasets. In order to solve the sparse convex clustering model efficiently, we propose the SSNAL algorithm to solve the 3-block convex composite programming, which can include the sparse convex clustering model as a special case.

In summary, this thesis contributes to the topic of clustering and intra-group level feature selection from both the model analysis and numerical optimization algorithm perspectives.

Chapter 1

Introduction

This thesis focuses on developing a systematic model and efficient numerical algorithms to perform clustering and intra-group level feature selections simultaneously. In order to achieve this goal, we first investigate the convex clustering model in Chapter 3. We develop a highly efficient and scalable numerical algorithm, called a semismooth Newton CG based augmented Lagrangian method (SSNAL), to solve the general weighted convex clustering model. Furthermore, we study the theoretical recovery guarantee of the weighted convex clustering model and propose a mild sufficient condition to guarantee the perfect recovery property of the model for a given collection of a finite number of data points. We study the exclusive lasso regularizer, which could enforce intra-group sparsity in Chapter 4. We revisit the closed-form solution to the proximal mapping of the exclusive lasso regularizer and provide a rigorous proof, we also derive the corresponding HS Jacobian of the proximal mapping. Based on these theoretical analysis, we develop a dual Newton based preconditioned proximal point algorithm (PPDNA) to solve the machine learning model with the exclusive lasso regularization. Lastly, in Chapter 5, we design an efficient SSNAL for three-block convex composite programing problems, then we apply it to solve the sparse convex clustering model which could perform clustering and intra-group feature selection simultaneously.

1.1 Literature review

Clustering is a fundamental topic in unsupervised learning. Given a collection of n data points and an integer k, clustering is to assign these n data points to k clusters based on some kinds of metrics. One of the most popular metric that is based on the Euclidean distance is called minimal sum-of-squares (MSSC). Specifically, for n given data points in d-dimensional Euclidean space

$$S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n | \mathbf{s}_i \in \Re^d\},$$

the idea of MSSC is trying to find k centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ and k corresponding clusters $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k)$ based on the sum-of-squares:

$$\min_{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k} \sum_{i=1}^{n} \min\{\|\mathbf{s}_i - \mathbf{c}_1\|_2^2, \|\mathbf{s}_i - \mathbf{c}_2\|_2^2, \ldots, \|\mathbf{s}_i - \mathbf{c}_k\|_2^2\}. \tag{1.1}$$

Note that finding the optimal solution of the above bi-level programming problem is NP-hard.

On the other hand, if the points in each cluster $\mathcal{S}_i$ are fixed, then the minimal of the function

$$f(S, \mathcal{S}) := \sum_{i=1}^{k} (\frac{1}{|\mathcal{S}_i|} \sum_{j \in \mathcal{S}_i} \|\mathbf{s}_j - \mathbf{c}_i\|_2^2)$$

is achieved by

$$\mathbf{c}_i := \frac{1}{|\mathcal{S}_i|} \sum_{j \in \mathcal{S}_i} \mathbf{s}_j.$$

Based on the discussions, we can now introduce the popular and efficient algorithm in clustering: K-means clustering algorithm [4, 43] in Algorithm 1. Roughly speaking, the idea of K-means algorithm is to first randomly generate k cluster centers, then updating the membership of the data points and the center of each clusters alternatively, until the cluster memberships are stable.

1.1.1 Convex clustering

Although traditional clustering models, such as K-means clustering, hierarchical clustering are quite popular and scalable, they may suffer from poor performance

Algorithm 1: K-means Algorithm

For a given collection of data points $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ in $\Re^n$ and an integer k for the number of clusters. The K-means algorithm performs the following steps:

Initialization . Choose k cluster centers $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$ randomly in a domain containing all the points.

Repeat :

Step 1 Assign each point to the closest cluster center, i.e., for each data point $\mathbf{s}_i$, assign $\mathbf{s}_i$ to the cluster $\mathcal{S}_j$ if

$$\|\mathbf{s}_i - \mathbf{c}_j\|^2 \leq \|\mathbf{s}_i - \mathbf{c}_l\|, \quad \forall l \in \{1, 2, \ldots, k\}.$$

Step 2 Update the cluster centers based on current cluster memberships,

$$\mathbf{c}_i := \frac{1}{|\mathcal{S}_i|} \sum_{j \in \mathcal{S}_i} \mathbf{s}_j.$$

Until convergence criterion is satisfied.

because of the non-convexity of the models and the difficulties in finding global optimal solutions for such models. The clustering results are generally highly dependent on the initializations and the results could differ significantly with different initializations. Moreover, these clustering models require the prior knowledge about the number of clusters which is not available in many real applications. Therefore, in practice, K-means is typically tried with different cluster numbers and the user will then decide a suitable value based on his judgment on which computed result agrees best with his domain knowledge. Obviously, such a process could make the clustering results subjective.

To address these difficulties, a new convex clustering model was proposed recently in [31, 41, 55], which has been demonstrated to be more robust compared to those

traditional ones.

Let $A \in \mathbb{R}^{d\times n} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n]$ be a given data matrix with n observations and d features. The convex clustering model for these n observations solves the following convex optimization problem:

$$\min_{X\in\Re^{d\times n}} \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|^2 + \gamma \sum_{i<j} \|\mathbf{x}_i - \mathbf{x}_j\|_p, \tag{1.2}$$

where $\gamma > 0$ is a tuning parameter, and $\|\cdot\|_p$ denotes the p-norm. We denote $\|\cdot\|$ as the vector 2-norm or the Frobenius norm of a matrix. The p-norm above with $p \geq 1$ ensures the convexity of the model. Typically p is chosen to be $1, 2$, or ∞. After solving (1.2) and obtaining the optimal solution $X^* = [\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*]$, we assign $\mathbf{a}_i$ and $\mathbf{a}_j$ to the same cluster if and only if $\mathbf{x}_i^* = \mathbf{x}_j^*$. In other words, $\mathbf{x}_i^*$ is the centroid for observation $\mathbf{a}_i$. (Here we used the word "centroid" to mean the approximate one associated with $\mathbf{a}_i$ but not the final centroid of the cluster to which $\mathbf{a}_i$ belongs to.) The idea behind this model is that if two observations $\mathbf{a}_i$ and $\mathbf{a}_j$ belong to the same cluster, then their corresponding centroids $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ should be the same. The first term in (1.2) is the fidelity term while the second term is the regularization term to penalize the differences between different centroids so as to enforce the property that centroids for observations in the same cluster should be identical.

The advantages of convex clustering lie mainly in two aspects. First, since the clustering model (1.2) is strongly convex, the optimal solution for a given positive γ is unique and is more easily obtainable than traditional clustering algorithms like K-means. Second, instead of requiring the prior knowledge of the cluster number, we can generate a clustering path via solving (1.2) for a sequence of positive values of γ.

To handle cluster recovery for large-scale data sets, various researchers, e.g., [31,41,54,55,67,89] have suggested the following weighted clustering model modified from (1.2):

$$\min_{X\in\Re^{d\times n}} \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|^2 + \gamma \sum_{i<j} w_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_p, \tag{1.3}$$

where $w_{ij} = w_{ji} \geq 0$ are given weights that are generally chosen based on the given input data A. One can regard the original convex clustering model (1.2) as a special case if we take $w_{ij} = 1$ for all $i < j$. To make the computational cost cheaper when evaluating the regularization term, one would generally put a non-zero weight only for a pair of points which are nearby each other, and a typical choice of the weights is

$$w_{ij} = \begin{cases} \exp(-\phi\|\mathbf{a}_i - \mathbf{a}_j\|^2) & \text{if } (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathcal{E} = \cup_{i=1}^{n}\{(i,j) \mid j \text{ is among } i\text{'s } k\text{-nearest neighbors}, i < j \leq n\}$, and ϕ is a given positive constant. Although we need to compute the pair-wise distance among all the data points and sort them by distance (whose computational complexity is $O(n^2)$), but we only need to do it once.

The advantages just mentioned and the success of the convex model (1.2) in recovering clusters in many examples with well selected values of γ have motivated researchers to provide theoretical guarantees on the cluster recovery property of (1.2). The first theoretical result on cluster recovery established in [89] is valid for only two clusters. It showed that the model (1.2) can recover the two clusters perfectly if the data points are drawn from two cubes that are well separated. The paper [67] further analyzed the statistical properties of (1.2). Recently, [54] provided theoretical recovery results in the general case of k clusters under relatively mild sufficient conditions for the fully uniformly weighted convex model (1.2).

In the practical aspect, various researchers have observed that better empirical performance can be achieved by (1.3) with well chosen weights when comparing to the original model (1.2) [16, 31, 41]. However, to the best of our knowledge, no theoretical recovery guarantee has been established for the general weighted convex clustering model (1.3). As a step forward in this direction, in Chapter 3 of this thesis, we propose mild sufficient conditions for (1.3) to attain perfect recovery guarantee, which also include and improve the theoretical results in [54, 89] as special cases. Our theoretical results thus definitively strengthened the theoretical foundation of

convex clustering model.

As expected, the conditions provided in the theoretical analysis are usually not checkable before one finds the right clusters and thus the range of parameter values for γ to achieve perfect recovery is an unknown priori. In practice, this difficulty is mitigated by choosing a sequence of values of γ to generate a clustering path.

The challenges for the convex model to obtain meaningful cluster recovery is then to solve the model efficiently for a range of values of γ. In [41], the authors used the off-the-shelf solver, CVX, to generate the solution path. However, Hocking et. al. [31] realized that CVX is competitive only for small-scale problems and it does not scale well when the number of data points increases. Thus the paper [31] introduced three algorithms based on the subgradient methods for different regularizers corresponding to $p = 1, 2, \infty$. Recently, some new algorithms have been proposed to solve this problem. In particular, Eric Chi et al. [16] adapted the ADMM and AMA to solve (1.2). However, as we will see in Chapter 3, both algorithms may still encounter scalability issues, albeit less severely than CVX. Furthermore, the efficiency of these two algorithms are sensitive to the parameter value γ. This is not a favorable phenomenon since we need to solve (1.2) with γ in a relative large range to generate the clustering path. In [54], the authors proposed a stochastic splitting algorithm for (1.2) in an attempt to resolve the aforementioned scalability issues. Although this stochastic approach scales well with the problem scale (n in (1.2)), the convergence rate shown in [54] is rather weak in that it requires at least $l \geq n^4/\varepsilon$ iterations to generate a solution X^l such that $\|X^l - X^*\|^2 \leq \varepsilon$ is satisfied with high probability. Moreover, because the error estimate is given in the sense of high probability, it is difficult to design an appropriate stopping condition for the algorithm in practice.

As the readers may observe, all the existing algorithms are purely first-order

methods that do not use any second-order information underlying the convex clustering model. In contrast, in Chapter 3, we design and analyse a deterministic second-order algorithm, the semismooth Newton based augmented Lagrangian method (SSNAL), to solve the convex clustering model. Our algorithm is motivated by the recent work [39] in which the authors have proposed a semismooth Newton augmented Lagrangian method (ALM) to solve Lasso and fused Lasso problems, and the algorithm has been demonstrated to be highly efficient for solving large, or even huge scale problems accurately. We are thus inspired to adapt this ALM framework for solving the convex clustering model (1.3) in Chapter 3.

1.1.2 Regularizers for structure sparsity

Structured sparsity is very important in feature learning, not only for avoiding over-fitting, but also in making the model more interpretable. Many regularizers and their combinations have been proposed to enforce sparsity for parameterized machine learning models [68, 69, 78, 79]. The most popular regularizers among them are probably the standard lasso [68] and the group lasso [78, 79] regularizers. Lasso, group lasso and their variants have been intensively studied in terms of both their statistical properties [12, 14, 68, 79, 85, 90] and efficient numerical computations [8, 24, 39, 52, 74, 78, 84]. The classical lasso model has been important in enforcing sparsity on variables while performing feature selection. However, there is no structure enforced in the sparsity pattern. Instead, the group lasso is known to enforce the sparsity at an inter-group level, where variables from different groups compete to be selected. The idea behind the group lasso is that if a few features in one group are important, then most of the features in the same group should also be important.

However, in some real applications, instead of the unstructured sparsity (e.g. lasso) or the inter-group level structured sparsity (e.g. group lasso), we also need the intra-group level sparsity. That is, not only features from different groups, but also features in a seemingly cohesive group are competing to survive. A realistic example

comes from building an index exchange-traded fund (index ETF) to track a specific index in the stock market. To diversify the risk across different sectors, we need to do portfolio selection both across and within sectors, which indeed means that we also need the intra-group level sparsity. To achieve this, a new regularizer called the exclusive lasso has been proposed in [34,87] (also named as elitist lasso [35,36]). Given a positive weight vector $w \in \Re^n_{++}$, and a partition of variable index groups $\mathcal{G} := \{g | g \subseteq \{1, 2, \ldots, n\}\}$ such that $\bigcup_{g\in\mathcal{G}} g = \{1, 2, \ldots, n\}$ and $g_i \bigcap g_j = \emptyset$ for any $g_i, g_j \in \mathcal{G}$. For $x \in \Re^n$, the weighted exclusive lasso regularizer is defined as

$$\Omega^{\mathcal{G},w}(x) := \textstyle\sum_{g\in\mathcal{G}} \|w_g \circ x_g\|_1^2, \tag{1.4}$$

where "$\circ$" denotes the Hadamard product, and x_g denotes the sub-vector of x with those elements not in g removed from x. As indicated in the above expression, an ℓ_2 norm square is applied to combine different groups, and a weighted ℓ_1 norm is used to enforce sparsity within each group. Naturally, when solving machine learning problems which involves a loss function and the exclusive lasso regularizer, we can expect that each x_g is nonzero. Under some strict assumptions, this statistical property is carefully studied in [11].

The exclusive lasso regularizer was first proposed for multi-task learning [87], and has been widely applied in applications such as image processing [15,82], sparse feature clustering [76] and NMR spectroscopy [11]. Some numerical optimization algorithms have been proposed for solving models involving the exclusive lasso regularizer, such as the smooth minimization via APG [15, 82], the iterative least squares algorithm (ILSA) [34,76], and the coordinate descent method [11]. However, some popular algorithmic frameworks like the accelerated proximal gradient (APG) [52,80], FISTA [8] and alternating direction method of multipliers (ADMM) [23,28] have not been used to solve these kind of problems. The main reason could be due to the fact that the closed-form solution to $\text{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$ is not well known in the community. In order to adopt a proximal gradient method to solve the exclusive lasso model, Campbell et.al [11] used an iterative subroutine to compute $\text{Prox}_{\rho\|\cdot\|_1^2}(\cdot)$ without the weights.

In this thesis, we recap the closed-form solution to the proximal mapping of the exclusive lasso regularizer reported in [35], Then we provide a rigorous proof in Chapter 4 based on a quadratic programming reformulation and the corresponding Karush-Kuhn-Tucker (KKT) conditions. As mentioned above, such a closed-form solution will be directly used in some popular algorithmic framework such as the APG and ADMM for solving the exclusive lasso models. However, based on our numerical experiments, these first-order algorithms are still not efficient enough in solving the large scale exclusive lasso problems.

To overcome the challenges in solving large scale cases, we design a highly efficient second-order type algorithm, the dual Newton based preconditioned proximal point algorithm (PPDNA), to solve the exclusive lasso model. As a key ingredient for PPDNA, we derive the Han-Sun (HS) Jacobian of $\text{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$. The numerical experiments shown in later section will demonstrate the superior performance of PPDNA for solving popular machine learning models with the exclusive lasso regularizer, comparing to other state-of-art algorithms mentioned previously.

1.1.3 Sparse convex clustering

Although the performance of the convex clustering model (1.3) is attractive, it's easy to realize that the convex clustering model could perform well only under the scenarios when the features of the input data are meaningful for clustering. The performance of convex clustering model could be severely deteriorated when clustering high-dimensional data, especially for the scenario where a number of features contain no information about the clustering structure.

To overcome the difficulties coming from uninformative features, we hope to have a new model that could do clustering and feature selection at the same time. Binhuan Wang et. al. [73] proposed a sparse convex clustering model by adding a group lasso regularization term for the features. For a given input data $A \in \Re^{p\times n}$ with n data points and p features, the sparse convex clustering model proposed

in [73] has the following form:

$$\min_{X\in\Re^{p\times n}} \frac{1}{2}\sum_{i=1}^{n}\|X_{\cdot,i}-A_{\cdot,i}\|_2^2+\Gamma(X;\gamma_1,\gamma_2), \tag{1.5}$$

where

$$\Gamma(X;\gamma_1,\gamma_2)=\gamma_1\sum_{(i,j)\in\mathcal{E}} w_{ij}\|X_{\cdot,i}-X_{\cdot,j}\|_2+\gamma_2\sum_{j=1}^{p}u_j\|X_{j,\cdot}\|_2.$$

Here $A_{\cdot,i}$ and $X_{\cdot,i}$ are the i-th column of A and X, respectively, and $X_{j,\cdot}$ is the j-th row of X.

The feature-wise group regularization term (the second term in $\Gamma(X;\gamma_1,\gamma_2)$) will help to enforce the sparsity of the features, which will be helpful in feature selection.

The sparse convex clustering model (1.5) has been proven to be useful for clustering on high dimensional data [73]. However, it's not very effective in selecting local feature sets, especially in the scenario when there are overlapping features in different clusters.

To capture better local hidden features, Yamada et al. [76] proposed a new model including a sample-wise regularization term for the features, which is

$$\min_{X\in\Re^{p\times n}} \frac{1}{2}\sum_{i=1}^{n}\|X_{\cdot,i}-A_{\cdot,i}\|_2^2+\Lambda(X;\gamma_1,\gamma_2), \tag{1.6}$$

where

$$\Lambda(X;\gamma_1,\gamma_2)=\gamma_1\sum_{(i,j)\in\mathcal{E}} w_{ij}\|X_{\cdot,i}-X_{\cdot,j}\|_2+\gamma_2\sum_{i=1}^{n}\|X_{\cdot,i}\|_1^2.$$

By imposing the sample-wise $\ell_{1,2}$ norm in the regularization terms, the model can select a small number of elements within each $X_{\cdot,i}$ [34, 76, 87]. As mentioned in [76], taking the square of ℓ_1 norm will enforce $X_{\cdot,i}$ to be sparse but still remain nonzero[1], which will make the results more interpretable.

The authors in [76] proposed an iterative least square algorithm to solve (1.6) that involves the computation of the inverse of a large matrix. The computational cost is expensive for large datasets. As mentioned in [76], we can adopt the ADMM

[1]For this claim, no rigorous mathematical proof could be found to the best of our knowledge.

to solve (1.6), where each iteration of the algorithm requires the efficient computation of proximal mapping of $\|\cdot\|_1^2$.

Inspired by [81], we will design a semismooth Newton based augmented Lagrangian method (SSNAL) to solve a more general case of (1.6), given as

$$\min_{X\in\Re^{p\times n}} \frac{1}{2}\sum_{i=1}^{n}\|X_{\cdot,i}-A_{\cdot,i}\|_2^2+\Lambda_H(X;\gamma_1,\gamma_2), \tag{1.7}$$

where

$$\Lambda_H(X;\gamma_1,\gamma_2)=\gamma_1\sum_{(i,j)\in\mathcal{E}} w_{ij}\|X_{\cdot,i}-X_{\cdot,j}\|_2+\gamma_2\sum_{i=1}^{n}\|H_{\cdot,i}\circ X_{\cdot,i}\|_1^2,$$

and $H\in\Re^{p\times n}$ is a given weight matrix.

With the theoretical results of the proximal mapping and corresponding Jacobian derived in chapter 3 and chapter 4, we will derive a semismooth Newton CG based augmented Lagrangian method of multipliers (SSNAL) to solve the 3-block convex composite programming reformulation of (1.7) in chapter 5, which will include the model (1.7) as a special case.

1.2 Thesis organization

The rest of the thesis is organized as follows. In Chapter 2, we present some preliminaries that will be repeatedly used in the subsequent discussions. In Chapter 3, we first show the theoretical recovery guarantee of the general weighted convex clustering model, then we introduce the highly efficient and robust algorithm SSNAL for solving the convex clustering model. At the end, we present numerical results to demonstrate that SSNAL achieves the state-of-art results comparing to other popular numerical algorithms. In Chapter 4, we study the exclusive lasso regularizer for intra-group feature selections. We proposed the dual Newton based preconditioned proximal point algorithm (PPDNA) for solving a two-block convex composite programing problem, which includes the exclusive lasso model and the logistic regression model with the exclusive lasso regularizer as special cases. In order to perform the

algorithm PPNDA practically, we study the closed form solution to the proximal mapping of the exclusive lasso regularizer and derive the corresponding HS Jacobian. The numerical experiments on randomly generated data demonstrate the high efficiency and scalability of PPDNA in solving the models with the exclusive lasso regularizer. The numerical results on a realistic example also demonstrate the power of the exclusive lasso regularizer in intra-group feature selections. Lastly, we study the sparse convex clustering model in Chapter 5, which could perform clustering and intra-group feature selection simultaneously. We propose the SSNAL algorithm for 3-block convex composite programming and applies it in solving the sparse convex clustering model. We summarize the thesis and discuss about the future research plans in Chapter 6.

Chapter 2

Preliminaries

In this chapter, we provide some important preliminary knowledge which will be frequently used in the remaining parts of this thesis.

2.1 Moreau-Yosida regularization

In this section, we will discuss Moreau-Yosida regularization, a very important tool that will be frequently used in this thesis. Moreau-Yosida regularization is commonly used in nonsmooth optimization algorithm since it could smooth a nonsmooth function.

Let $\mathcal{X}$ be a real finite dimensional Euclidean space equipped with an inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\| \cdot \|$. Let $f : \mathcal{X} \to \Re$ be a proper closed convex function. The Moreau-Yosida regularization of f at point $x \in \mathcal{X}$ is defined as

$$\phi_f(x) := \min_{y \in \mathcal{X}} \{ f(y) + \frac{1}{2} \|y - x\|^2 \}, \quad \forall x \in \mathcal{X}. \tag{2.1}$$

The function ϕ_f is also known as the Moreau envelope of f. The following proposition shows that (2.1) is well defined [49].

Proposition 1. *For any $x \in \mathcal{X}$, the problem (2.1) has a unique optimal solution.*

Definition 1 (Proximal mapping)**.** *The unique optimal solution of (2.1), denotes by* $\mathrm{Prox_f(x)}$*, is called the proximal mapping of x associated with f.*

Many popular numerical optimization algorithms, especially first-order algorithms, highly depend on the efficient computation of the proximal mapping of some given functionals. Here, we list some closed form formulas of the proximal mappings for some popular functions in Table 2.1.

Table 2.1: Proximal maps for selected functions

$p(\cdot)$	$\text{Prox}_{\text{tp}}(\mathbf{x})$	Comment
$\|\cdot\|_1$	$\left[1-\frac{t}{\|\mathbf{x}_l\|}\right]_+\mathbf{x}_l$	Elementwise soft-thresholding
$\|\cdot\|_2$	$\left[1-\frac{t}{\|\mathbf{x}\|_2}\right]_+\mathbf{x}$	Blockwise soft-thresholding
$\|\cdot\|_\infty$	$\mathbf{x}-\Pi_{t\mathcal{S}}(\mathbf{x})$	$\mathcal{S}$ is the unit ℓ_1-ball

Proposition 2. *Let $f:\mathcal{X}\to\Re$ be a closed proper convex function, ϕ_f be the Moreau-Yosida regularization of f, and Prox_f be the associated proximal mapping. Then the following properties hold.*

(i) $\arg\min_{x\in\mathcal{X}} f(x) = \arg\min_{x\in\mathcal{X}} \phi_f(x)$.

(ii) Both Prox_f and $I-\text{Prox}_f$ are firmly non-expansive, i.e., for any $x,y\in\mathcal{X}$,

$$\|\text{Prox}_f(x)-\text{Prox}_f(y)\|^2 \le \langle \text{Prox}_f(x)-\text{Prox}_f(y), x-y\rangle,$$

$$\|(x-\text{Prox}_f(x))-(y-\text{Prox}_f(y))\|^2 \le \langle (x-\text{Prox}_f(x))-(y-\text{Prox}_f(y)), x-y\rangle.$$

(iii) The Moreau envelope ϕ_f is continuously differentiable, and its gradient can be computed by

$$\nabla\phi_f(x) = x-\text{Prox}_f(x), \quad x\in\mathcal{X}.$$

Next, we introduce an important ingradient in algorithm design, Moreau decomposition.

Proposition 3 (Moreau decomposition)**.** *Let $f:\mathcal{X}\to(-\infty,+\infty]$ be a closed proper convex function and σ be a positive scalar. Then the following Moreau identity holds:*

$$\text{Prox}_{\sigma f}(x)+\sigma\text{Prox}_{\sigma^{-1}f^*}(\sigma^{-1}x) = x, \quad \forall x\in\mathcal{X},$$

where f^* is the conjugate function of f defined as

$$f^*(y) := \sup_{x\in\mathcal{X}} \{\langle x, y\rangle - f(x)\}.$$

2.2 Proximal ADMM for convex composite programming

In this section, we recap the proximal alternating direction method of multipliers (PADMM), which will be used as one of the benchmark algorithms in our numerical experiments.

Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be finite dimensional real Euclidian spaces. Let $f : \mathcal{Y} \to (-\infty, +\infty]$ and $g : \mathcal{Z} \to (-\infty, +\infty]$ be closed proper convex functions, $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ and $\mathcal{B} : \mathcal{X} \to \mathcal{Z}$ be linear maps. Let ∂f and ∂g be the subdifferential mappings of f and g, respectively. Since both ∂f and ∂g are maximally monotone [64, Theorem12.17]. By [26], there exists two self-adjoint and positive semidefinite operators Σ_f and Σ_g such that for all $y, \tilde{y} \in dom(f)$, $\xi \in \partial f(y)$ and $\tilde{\xi} \in \partial f(\tilde{y})$,

$$\langle y - \tilde{y}, \xi - \tilde{\xi}\rangle \geq \|y - \tilde{y}\|^2_{\Sigma_f}, \tag{2.2}$$

and for all $z, \tilde{z} \in dom(g)$, $\eta \in \partial g(z)$ and $\tilde{\eta} \in \partial g(\tilde{z})$,

$$\langle z - \tilde{z}, \eta - \tilde{\eta}\rangle \geq \|z - \tilde{z}\|^2_{\Sigma_g}. \tag{2.3}$$

Now, we review the semi-proximal ADMM algorithm proposed in [26] for solving a generic two blocks convex composite programming problem. Consider the convex optimization problem with the following 2-block separable structure

$$\begin{aligned} \min \quad & f(y) + g(z) \\ s.t. \quad & \mathcal{A}^*y + \mathcal{B}^*z = c. \end{aligned} \tag{2.4}$$

The dual problem of (2.4) is given by

$$\begin{aligned} \min \quad & \langle c, x\rangle + f^*u + g^*(v) \\ s.t. \quad & \mathcal{A}x + u = 0, \\ & \mathcal{B}x + v = 0. \end{aligned} \tag{2.5}$$

Let $\sigma > 0$ be a given scalar, the augmented Lagrangian function associated with (2.4) is given as follows:

$$\mathcal{L}_\sigma(y, z; x) := f(y) + g(z) + \langle x, \mathcal{A}^* y + \mathcal{B}^* z - c\rangle + \frac{\sigma}{2}\|\mathcal{A}^* y + \mathcal{B}^* z - c\|^2.$$

The semi-proximal ADMM proposed in [26], when applied to (2.4), has the template given in Algorithm 2. Since the proximal terms added here are allowed to be positive, the corresponding method is referred to as semi-proximal ADMM instead of proximal ADMM as in [26].

Algorithm 2: sPADMM: A generic 2-block semi-proximal ADMM for solving (2.4).

Let $\sigma > 0$ and $\tau \in (0, \infty)$ be given parameters. Let $\mathcal{T}_f$ and $\mathcal{T}_g$ be given self-adjoint positive semidefinite, not necessarily positive definite, linear operators defined on $\mathcal{Y}$ and $\mathcal{Z}$, respectively. Choose $(y^0, z^0; x^0) \in dom(f) \times dom(g) \times \mathcal{X}$. For $k = 0, 1, 2, \ldots,$ perform the kth iteration as follows:

Step 1 . Compute

$$y^{k+1} = \arg\min_y \mathcal{L}_\sigma(y, z^k; x^k) + \frac{\sigma}{2}\|y - y^k\|^2_{\mathcal{T}_f}. \tag{2.6}$$

Step 2 . Compute

$$z^{k+1} = \arg\min_z \mathcal{L}_\sigma(y^{k+1}, z; x^k) + \frac{\sigma}{2}\|z - z^k\|^2_{\mathcal{T}_g}. \tag{2.7}$$

Step 3 .

$$x^{k+1} = x^k + \tau\sigma(\mathcal{A}^* y^{k+1} + \mathcal{B}^* z^{k+1} - c). \tag{2.8}$$

In Algorithm 2, the presence of $\mathcal{T}_f$ and $\mathcal{T}_g$ is to guarantee the existence of solutions for the subproblems (2.6) and (2.7). However, the choice of $\mathcal{T}_f$ and $\mathcal{T}_g$ are problem dependent. When we choose $\mathcal{T}_f = 0$ and $\mathcal{T}_g = 0$, the Algorithm 2 becomes the classical ADMM for a two-block convex composite programming problem.

For the convergence of the 2-block semi-proximal ADMM, we need the following assumption.

Assumption 2.1. *There exists* $(\hat{y}, \hat{z}) \in ri(domf \times domg)$ *such that* $\mathcal{A}^*\hat{y} + \mathcal{B}^*\hat{z} = c$.

Theorem 2.1. *[26, Theorem B.1] Let* Σ_f *and* Σ_g *be the self-adjoint and positive semidefinite operators defined by (2.2) and (2.2), respectively. Suppose that the solution set of problem (2.4) is nonempty and Assumption 2.1 holds. Assume that* $\mathcal{T}_f$ *and* $\mathcal{T}_g$ *are chosen such that the sequence* $\{(y^k, z^k, x^k)\}$ *generated by Algorithm sPADMM is well defined. Then, under the condition either (a)* $\tau \in (0, (1+\sqrt{5})/2)$ *or (b)* $\tau \geq (1+\sqrt{5})/2$ *but* $\sum_{k=0}^{\infty}(\|\mathcal{B}^*(z^{k+1}-z^k)\|^2+\tau^{-1}\|\mathcal{A}^*y^{k+1}+\mathcal{B}^*z^{k+1}-c\|^2) < \infty$, *the following results hold:*

(i) If $(y^\infty, z^\infty, x^\infty)$ *is an accumulation point of* $\{(y^k, z^k, x^k)\}$, *then* (y^∞, z^∞) *solves problem (2.4) and* x^∞ *solves (2.5), respectively.*

(ii) If both $\sigma^{-1}\Sigma_f + \mathcal{T}_f + \mathcal{A}\mathcal{A}^*$ *and* $\sigma^{-1}\Sigma_g + \mathcal{T}_g + \mathcal{B}\mathcal{B}^*$ *are positive definite, then the sequence* $\{(y^k, z^k, x^k)\}$, *which is automatically well defined, converges to a unique limit, say,* $(y^\infty, z^\infty, x^\infty)$ *with* (y^∞, z^∞) *solving problem (2.4) and* x^∞ *solving (2.5), respectively.*

(iii) When the y-part disappears, the corresponding results in parts (i) and (ii) hold for (2.4) under the condition either $\tau \in (0, 2)$ *or* $\tau \geq 2$ *but* $\sum_{k=0}^{\infty}\|\mathcal{B}^*z^{k+1} - c\|^2 < \infty$.

2.3 Accelerated proximal gradient algorithm (APG)

In this section, we will discuss another popular first order algorithm, accelerated proximal gradient algorithm for solving the general unconstrained nonsmooth convex minimization problem which will include the convex clustering model and the exclusive lasso model as special cases. Consider

$$\min F(x) := f(x) + p(x), \tag{2.9}$$

where $p : \Re^n \to \Re$ is a proper, convex, lower semicontinuous (lsc) [61] function and f is convex smooth (i.e., continuously differentiable) on an open subset of $\Re^n$ containing $dom\ p = \{x \in \Re^n \mid p(x) < \infty\}$. We assume that $dom\ p$ is closed and ∇f is Lipschitz continuous on $dom\ p$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\|^2 \leq L_f \|x - y\|^2, \quad \forall x, y \in dom f, \tag{2.10}$$

for some positive scalar L_f.

Now, we introduce the proximal gradients method and some accelerated version to solve (2.9).

For any $y \in dom\ p$ and a given positive scalar L, consider the following quadratic approximation of $F(\cdot)$ at y as

$$Q_L(x, y) := f(y) + \frac{L}{2}\|x - u\|^2 + p(x) - \frac{1}{2L}\|\nabla f(y)\|^2, \tag{2.11}$$

where $u = y - \frac{1}{L}\nabla f(y)$. Since $Q_L(x, y)$ is strongly convex, if admits a unique minimizer. We denote the unique minimizer by

$$S_L(u) : \arg\min\{Q_L(x, y) \mid x \in dom\ p\}. \tag{2.12}$$

We introduce a general proximal gradient algorithmic framework shown in [70] for solving (2.9) in Algorithm 3.

Note that Algorithm 3 is a more general algorithmic framework, Fukushima and Mine [27] studied a proximal gradient descent method with $t^k = 1$ for all k and step size α^k chosen by an Armijo-type rule. We first establish the convergence result for the proximal gradient descent method in Theorem 2.2.

Theorem 2.2. *[70, Theorem 2.1] Assume the optimal solution of problem (2.9) Ω^* is nonempty. Let $\{x^k\}$ be the sequence generated by the Algorithm 3 with $L^k = L_f$, $t^k = 1$, and $\alpha^k = 1$ for all k. Then, for any $k \geq 1$, we have*

$$F(x^k) - F(x^*) \leq \frac{L_f \|x^0 - x^*\|^2}{2k}, \quad \forall x^* \in \Omega^*.$$

Algorithm 3: A general proximal gradient algorithmic framework for solving (2.9).

Choose $x^0 = x^{-1} \in dom(p)$, $t^0 = t^{-1} \in [1, \infty)$. For $k = 0, 1, 2, \ldots$, generate x^{k+1} from x^k according to the following iteration:

Step 1 . Set

$$y^k = x^k + \frac{t^{k-1} - 1}{t^k}(x^k - x^{k-1}).$$

Step 2 . Set

$$u^k = y^k - (L^k)^{-1}\nabla f(y^k),$$

where $L^k > 0$ and compute

$$S_{L^k}(u^k).$$

Step 3 . Choose a step size $\alpha^k > 0$ and set

$$x^{k+1} = x^k + \alpha^k(S_{L^k}(u^k) - x^k).$$

Step 4 . Choose $t^{k+1} \in [1, \infty)$ satisfying

$$(t^{k+1})^2 - t^{k+1} \leq (t^k)^2. \tag{2.13}$$

As a direct corollary of Theorem 2.2, we have

$$F(x^k) - \inf_{x \in \Re^n} F(x) \leq O(L_f/k) \quad \forall k.$$

So for any $\epsilon > 0$, the algorithm terminates in $O(L_f/\epsilon)$ iterations with an $\epsilon -$ optimal solution. So the sequence $\{x^k\}$ converges relatively slowly.

In the smooth setting, Nesterov [51] proposed an algorithm using only interpolation strategy to achieve $O(1/k^2)$ iteration complexity. Later, Beck and Teboulle [8] extend the results in [51] to solve the nonsmooth problem (2.9).

The condition (2.13) allows t_k to increase, but cannot increase too rapidly. Since larger t^k will improve the convergence rate of the algorithm, we alternatively solve

(2.13) with equality instead of inequality, which yields

$$t^{k+1} = \frac{1 + \sqrt{1 + 4(t^k)^2}}{2}.$$

The accelerated version of the proximal gradient based on the interpolation is shown in Algorithm 4.

Algorithm 4: APG for solving (2.9).

Choose $x^0 = x^{-1} \in dom(p)$, $t^0 = t^{-1} = 1$. For $k = 0, 1, 2, \ldots$, generate x^{k+1} from x^k according to the following iterations:

Step 1 . Set

$$y^k = x^k + \frac{t^{k-1} - 1}{t^k}(x^k - x^{k-1}).$$

Step 2 . Set

$$u^k = y^k - (L^k)^{-1}\nabla f(y^k),$$

where $L^k = L_f$ and compute

$$S_{L^k}(u^k).$$

Step 3 . Set

$$x^{k+1} = S_{L^k}(u^k).$$

Step 4 . Compute

$$t^{k+1} = \frac{1 + \sqrt{1 + 4(t^k)^2}}{2}.$$

We end this section by showing the convergence result of the APG algorithm in Theorem 2.3, detailed proof could be found in [8].

Theorem 2.3. *[8, Theorem 4.4] Let $\{x^k\}$ be generated by Algorithm 4. Then for any $k \geq 1$*

$$F(x^k) - F(x^*) \leq \frac{2L_f\|x^0 - x^*\|^2}{(k+1)^2}, \quad \forall x^* \in \Omega^*. \tag{2.14}$$

2.4 Semismooth Newton method

In this section, we will discuss a semismooth Newton method, which will be an important sub-routine in our proposed algorithmic framework. Before that, we first give some preliminaries.

Definition 2 (Directional differentiability)**.** *[10, Definition 2.44] We say that $F : \Re^n \to \Re^m$ is directionally differentiable at a point $x \in \Re^n$ in the direction $h \in \Re^n$ if the limit*

$$F'(x,h) := \lim_{t \downarrow 0} \frac{F(x+th) - F(x)}{t}$$

exists. If F is differentiable at x in every direction $h \in \Re^n$, we say that F is directionally differentiable at x.

Next, we define the differentiability in the sense of Fréchet.

Definition 3 (Fréchet Differentiability)**.** *[10, Definition 2.48] We say that $F : \Re^n \to \Re^m$ is directionally differentiable at a point $x \in \Re^n$ in the direction $h \in \Re^n$ in the* Fréchet *sense if F is directionally differentiable at x and*

$$F(x+h) = F(x) + F'(x,h) + o(\|h\|), \quad h \in \Re^n.$$

If, in addition, $F'(x,\cdot)$ is linear and continuous, it is said that F is Fréchet *differentiable at x.*

We now introduce the important Rademacher's theorem which will lead to the definition of the generalized Jacobian in Clark's sense.

Theorem 2.4 (Rademacher's theorem)**.** *Suppose that $F : \Re^n \to \Re^m$ is locally Lipschitz continuous on an open set $\mathcal{O} \subseteq \Re^n$. Then F is almost everywhere (*Fréchet*) differentiable in $\mathcal{O}$.*

Let $F : \Re^n \to \Re^m$ be a locally Lipschitz continuous function, then it's almost everywhere (Fré*chet*) differentiable. Denote $\mathcal{D}_F$ to be the set of points in $\Re^n$ where F is differentiable and $F'(x)$ be the Jacobian of F at $x \in \mathcal{D}_F$. We define the Bouligand subdifferential and the Clark generalized Jacobian of F at any $x \in \Re^n$ below.

Definition 4. *For a locally Lipschitz continuous function F and the corresponding differentiable set $\mathcal{D}_F$ defined above, the Boulingand subdifferential (B-subdifferential) of F at any $x \in \Re^n$ is defined as*

$$\partial_B F(x) = \{ \lim_{x^k \to x} F'(x^k) \mid x^k \in \mathcal{D}_F \},$$

and the Clark generalized Jacobian of F at $x \in \Re^n$ is defined as the convex hull of $\partial_B F(x)$, i.e.,

$$\partial F(x) = conv\{\partial_B F(x)\}.$$

The following proposition about the B-subdifferential and the Clark generalized Jacobian is from [18].

Proposition 4. *Let $\mathcal{O} \subseteq \Re^n$ be an open set and $F : \mathcal{O} \to \Re^m$ be a locally Lipschitz continuous function. Then the following properties hold:*

(i) $\partial_B F(x)$ is a nonempty compact subset of $\Re^{m \times n}$ for any $x \in \mathcal{O}$.

(ii) $\partial_B F(x)$ is upper semicontinuous at $x \in \mathcal{O}$, i.e., for any $\epsilon > 0$, there exists $\delta > 0$ such that

$$\partial_B F(y) \subseteq \partial_B F(x) + \epsilon \mathcal{B}, \quad \forall y \text{ satisfying } \|y - x\| < \delta,$$

where $\mathcal{B} \subseteq \Re^{m \times n}$ is the open unit ball centered at the origin.

The properties above are also true for $\partial F(\cdot)$.

With all the preparations above, we now introduce the definitions of semismoothness, which are mainly adopted from [37, 46, 57].

Definition 5. *Let $\mathcal{O} \subseteq \Re^n$ be an open set and $F : \mathcal{O} \to \Re^m$ be a locally Lipschitz continuous function. F is said to be G-semismooth at $x \in \mathcal{O}$ if for any $V \in \partial F(x + \Delta x)$ with $\Delta x \to 0$,*

$$F(x + \Delta x) - F(x) - V \Delta x = o(\|\Delta x\|).$$

F is said to be strongly G-semismooth at $x \in \mathcal{O}$ if for any $V \in \partial F(x + \Delta x)$ with $\Delta x \to 0$,

$$F(x + \Delta x) - F(x) - V\Delta x = O(\|\Delta x\|^2).$$

If, in addition, F is directionally differentiable at x, then it's said that F is semismooth and strongly semismooth at x, respectively.

Definition 6. *Let $\mathcal{O} \subseteq \Re^n$ be an open set, $\mathcal{K} : \mathcal{O} \rightrightarrows \Re^{m\times n}$ be a nonempty compact valued, upper semicontinuous multifunction, and $F : \mathcal{O} \to \Re^m$ be a locally Lipschitz continuous function. F is said to be semismooth at $x \in \mathcal{O}$ with respect to the multifunction $\mathcal{K}$ if F is directionally differentiable at x and for any $V \in \mathcal{K}(x + \Delta x)$ with $\Delta x \to 0$,*

$$F(x + \Delta x) - F(x) - V\Delta x = o(\|\Delta x\|).$$

*Let α be a positive constant, F is said to be $\alpha-$*order *(strongly if $\alpha = 1$) semismooth at $x \in \mathcal{O}$ with respect to $\mathcal{K}$ if F is directionally differentiable at x and for any $V \in \mathcal{K}(x + \Delta x)$ with $\Delta x \to 0$,*

$$F(x + \Delta x) - F(x) - V\Delta x = O(\|\Delta x\|^{1+\alpha}).$$

F is said to be a semismooth (respectively, α-order semismooth, strongly semismooth) function on $\mathcal{O}$ with respect to $\mathcal{K}$ if it is semismooth (respectively, α-order semismooth, strongly semismooth) everywhere in $\mathcal{O}$ with respect to $\mathcal{K}$.

We usually regard Definition 5 as the classic and standard definition of semismoothness, whereas Definition 6 is more general as it involves a multifunction which could be but not limited to the Clark generalized Jacobian.

Before we introduce the semismooth Newton method, we note that, the class of semismooth functions includes many nonsmooth functions that we are interested in. In particular, the convex functions are examples of semismooth functions [46].

Now, we introduce the semismooth Newton (SSN) method [57] to solve unconstrained convex optimization problems with SC^1 objective functions, which are

essentially the subproblem in our algorithmic framework. Consider the SSN method for solving the following optimization problem

$$\min_{x\in\Re^n} f(x), \tag{2.15}$$

where $f : \Re^n \to \Re$ is a convex LC^1 function. A function is said to be SC^1 if it is a continuously differentiable function with locally Lipschitz continuous gradient, and the gradient is semismooth.

Since the objective function f of the unconstrained convex optimization problem (2.15) is differentiable, solving (2.15) is equivalent to solving the following nonsmooth equation

$$\nabla f(x) = 0. \tag{2.16}$$

Since f is SC^1, then, $\nabla f(x)$ is semismooth. We can adopt the semismooth Newton method to solve the nonsmooth equation (2.16) with the appealing superlinear convergence (even quadratic convergence) rate. The SSN method for solving (2.16) is shown in Algorithm 5.

Note that the Jacobian used in Algorithm 5 is not limited to Clark generalized Jacobian, whether we use the Clark generalized Jacobian or not is problem dependent. In some cases, the Clark generalized Jacobian is not easy to compute and we need to work with other computationally available generalized Jacobians. This will be seen in details when we apply the SSN[1] method in solving some specific models in the following chapters.

To close this section, we present the convergence result for the SSN method in Theorem 2.5, the proof could be found in [86].

Theorem 2.5. *Suppose that the equation (2.16) admits a unique solution $\bar{x}$, $\mathcal{K}$ is a nonempty compact valued, upper semicontinuous multifunction, with respect to which ∇f is semismooth, and every $V \in \mathcal{K}(\bar{x})$ is nonsingular. Let $\{x^j\}$ be the infinite sequence generated by Algorithm 5. Then $\{x^j\}$ converges to the unique solution $\bar{x}$ of*

[1] If the conjugate gradient method is used to solve (5), we denote the algorithm by SSNCG.

Algorithm 5: A semismooth Newton method for solving (2.16).

Given $\mu \in (0, 1/2)$, $\bar{\eta} \in (0, 1)$, $\tau \in (0, 1]$ and $\delta \in (0, 1)$. Let $\mathcal{K}$ be a nonempty compact valued, upper semicontinuous multifunction, with respect to which ∇f is semismooth. Choose $x^0 \in \Re^n$. Do the following steps for $j = 0, 1, \ldots$

Step 1 . (Newton direction) Choose $V_j \in \mathcal{K}(x^j)$. Solve the following linear system

$$V_j d = -\nabla f(x^j) \tag{2.17}$$

by a direct method or by the conjugate gradient (CG) algorithm to find d^j such that $\|V_j d^j + \nabla f(x^j)\| \leq \min(\bar{\eta}, \|\nabla f(x^j)\|^{1+\tau})$.

Step 2 . (Line search) Set $\alpha_j = \delta^{m_j}$, where m_j is the smallest nonnegative integer m for which

$$f(y^j + \delta^m d^j) \leq f(y^j) + \mu \delta^m \langle \nabla f(y^j), d^j \rangle. \tag{2.18}$$

Step 3 . Set $x^{j+1} = x^j + \alpha_j d^j$.

equation (2.16). *Moreover, the convergence rate is at least superlinear:*

$$\|x^{j+1} - \bar{x}\| = O(\|x^j - \bar{x}\|^{1+\tau}),$$

where $\tau \in (0, 1]$ *is the parameter given in Algorithm 5.*

2.5 Error bounds

To close this chapter, we introduce an important concept, the so called the error bound, which is critical for establishing the convergence rate results of an algorithm. Here, we just recap some important results, for more details about the error bounds, readers can refer to [20, 44, 71, 88] and the references therein.

Consider the following problem

$$\min_{x \in \mathcal{E}} F(x) := h(\mathcal{A}x) + \langle c, x \rangle + p(x), \tag{2.19}$$

where $\mathcal{E}$ and $\mathcal{O}$ are finite-dimensional Euclidean spaces, $\mathcal{A}:\mathcal{E}\to\mathcal{O}$ is a linear map. In this section, we also assume that the following assumptions hold for problem (2.19).

Assumption 2.2. *(a) $p:\mathcal{E}\to(-\infty,\infty]$ is a proper, closed and convex function.*

(b) $h:\mathcal{O}\to(-\infty,\infty]$ is convex smooth (i.e., continuously differentiable) function on $int(dom(h))$. In addition, we also assume that h is strongly convex and its gradient ∇h is Lipschitz continuous on any conpact convex set $V\subseteq dom(h)$.

(c) The optimal solution set $\mathcal{X}$ of problem (2.19) is nonempty and compact.

2.5.1 Some preliminaries from set-valued analysis

Before we discuss the error bounds for problem (2.19), we first introduce some necessary concepts and results in set-valued analysis [20, 64, 88].

Definition 7. *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be finite-dimensional Euclidean spaces, we say a mapping $\Gamma:\mathcal{E}_1\rightrightarrows\mathcal{E}_2$ is a multi-function (or set-valued mapping) if it assigns a subset $\Gamma(u)$ of $\mathcal{E}_2$ to each vector $u\in\mathcal{E}_1$.*
For a multi-function $\Gamma:\mathcal{E}_1\rightrightarrows\mathcal{E}_2$, we define its graph and domain by

$$gph(\Gamma):=\{(u,v)\in\mathcal{E}_1\times\mathcal{E}_2\mid v\in\Gamma(u)\},$$
$$dom(\Gamma):=\{u\in\mathcal{E}_1\mid\Gamma(u)\neq\emptyset\},$$

respectively. Furthermore, we define the inverse mapping of Γ, denote by Γ^{-1}, as a multi-function from $\mathcal{E}_2$ to $\mathcal{E}_1$ defined by

$$\Gamma^{-1}(v):=\{u\in\mathcal{E}_1\mid v\in\Gamma(u)\}.$$

Next, we introduce the definition of calmness and metric sub-regularity of the multi-function, which are high related to the error bounds.

Definition 8. *(i) A multi-function $\Gamma:\mathcal{E}_1\rightrightarrows\mathcal{E}_2$ is said to be calm at $\bar{u}\in\mathcal{E}_1$ for $\bar{v}\in\mathcal{E}_2$ if $(\bar{u},\bar{v})\in gph(\Gamma)$ and there exist constants $\kappa,\epsilon>0$ such that*

$$\Gamma(u)\cap\mathbb{B}_{\mathcal{E}_2}(\bar{v},\epsilon)\subseteq\Gamma(\bar{u})+\kappa\|u-\bar{u}\|\mathbb{B}_{\mathcal{E}_2}\quad\forall u\in\mathcal{E}_1. \tag{2.20}$$

Furthermore, we say Γ *is isolated calm at* $\bar{u} \in \mathcal{E}_1$ *for* $\bar{v} \in \mathcal{E}_2$ *if* $(\bar{u}, \bar{v}) \in gph(\Gamma)$ *and there exist constants* $\kappa_1, \epsilon_1 > 0$ *such that*

$$\Gamma(u) \cap \mathbb{B}_{\mathcal{E}_2}(\bar{v}, \epsilon_1) \subseteq \{\bar{v}\} + \kappa_1 \|u - \bar{u}\| \mathbb{B}_{\mathcal{E}_2} \quad \forall u \in \mathcal{E}_1. \tag{2.21}$$

(ii) A multi-function $\Gamma : \mathcal{E}_1 \rightrightarrows \mathcal{E}_2$ *is said to be metrically sub-regular at* $\bar{u} \in \mathcal{E}_1$ *for* $\bar{v} \in \mathcal{E}_2$ *if* $(\bar{u}, \bar{v}) \in gph(\Gamma)$ *and there exist constants* $\kappa, \epsilon > 0$ *such that*

$$dist(u, \Gamma^{-1}(\bar{v})) \leq \kappa dist(\bar{v}, \Gamma(u)) \quad \forall u \in \mathbb{B}_{\mathcal{E}_1}(\bar{u}, \epsilon). \tag{2.22}$$

The calmness and the metric subregularity of a given multi-function are not easy to check directly from the definitions. Fortunately, we have the following result [3] for a special class of multi-valued mappings, i.e., the sub-differential of convex functions.

Theorem 2.6. *[3, Theorem 3.3] Let* $\mathcal{H}$ *be a real Hilbert space endowed with the inner product* $\langle \cdot, \cdot \rangle$ *and* $f : \mathcal{H} \to (-\infty, +\infty]$ *be a proper lower semicontinuous convex function. Let* $\bar{v}, \bar{x} \in \mathcal{H}$ *satisfy* $\bar{v} \in \partial f(\bar{x})$*. Then* ∂f *is metric subregular at* $\bar{x}$ *for* $\bar{v}$ *if and only if there exists a neighborhood* $\mathcal{N}(\bar{x})$ *of* $\bar{x}$ *and a positive constant* c *such that*

$$f(x) \geq f(\bar{x}) + \langle \bar{v}, x - \bar{x} \rangle + c(dist(x, (\partial f)^{-1}(\bar{v})))^2, \quad \forall x \in \mathcal{N}(\bar{x}). \tag{2.23}$$

The next proposition shows the equivalence between the calmness of a multi-function and the metric subregularity of its inverse.

Proposition 5. *[22, Theorem 3H.3] For a multi-function* $\Gamma : \mathcal{E}_1 \rightrightarrows \mathcal{E}_2$*, let* $(\bar{u}, \bar{v}) \in gph(\Gamma)$*. Then* Γ *is calm at* $\bar{u}$ *for* $\bar{v}$ *if and only if its inverse* Γ^{-1} *is metrically subregular at* $\bar{v}$ *for* $\bar{u}$*.*

2.5.2 Error bounds

We now discuss the error bound conditions for (2.19) based on the preliminaries above. In general, the error bound condition gives us a handle of the structure on

the objective function near the optimal solution to deduce some useful quantitative controls.

Let $\mathcal{X}$ be the nonempty optimal solution set of problem (2.19). Let $T \subseteq \mathcal{E}$ be a set satisfying $\mathcal{X} \subseteq T$ and $r : \mathcal{E} \to \Re_+$ be a function satisfying $r(x) = 0$ if and only if $x \in \mathcal{X}$. We say that problem (2.19) satisfies the error bound condition for $\mathcal{X}$ with test set T and residual function r if there exists a constant $\kappa > 0$ such that

$$dist(x, \mathcal{X}) \leq \kappa \cdot r(x), \quad \forall x \in T, \tag{2.24}$$

where $dist(x, \mathcal{X}) = \inf_{z \in \mathcal{X}} \|z - x\|_2$ denotes the Euclidean distance from a vector $x \in \mathcal{E}$ to the set $\mathcal{X}$.

One popular and practical choice of the residual function $r(x)$ is $r_{\text{prox}}(x) := \|R(x)\|_2$ where $R : \mathcal{E} \to \mathcal{E}$ is the residual map defined by

$$R(x) := Prox_p(x - (\mathcal{A}^* \nabla h(\mathcal{A}x) + c)).$$

This popular choice of the residual function induces the following specific error bound property with the proximal map based residual function.

Definition 9 (EBP). *For any $\alpha \geq v^* := \min_{x \in \mathcal{E}} F(x)$, there exist constants $\kappa > 0$ and $\epsilon > 0$ such that*

$$dist(x, \mathcal{X}) \leq \kappa \|R(x)\|_2, \quad \forall x \in \mathcal{E} \text{ with } F(x) \leq \alpha, \|R(x)\|_2 \leq \epsilon.$$

The error bound property is important for the analysis of the convergence rate of first order algorithms. If the error bound condition holds for (2.19), then popular first order algorithms like proximal gradient (PG), proximal points algorithm (PPA) can be shown to converge linearly [63, 71].

The known results for the error bound property to hold relied on the polyhedral property of multi-valued mappings. Recently, Zhou Zirui et al. [88] and Cui Ying et al. [19] established the error bound property for an important non-polyhedral function, the nuclear norm regularizer. Here, we recap some important results.

Theorem 2.7. *[72, Theorem 4] Consider the general two-block convex optimization problem*

$$\min_{x\in\Re^n} \quad f(x)+p(x),$$

where f is strongly convex and differentiable with Lipschitz continuous gradient ∇f, and p is a closed, proper convex function, then the EBP holds.

Theorem 2.8. *([72, Lemma 7], [71, Theorem 2]) Consider the following problem*

$$\min_{x\in\Re^n} F(x) := h(\mathcal{A}x)+\langle c,x\rangle + p(x),$$

where $\mathcal{A}:\Re^n\to\Re^m$ is a linear map, $h:\Re^m\to(-\infty,\infty]$ is convex smooth (i.e., continuously differentiable) function on $int(dom(h))$. In addition, h is strongly convex and its gradient ∇h is Lipschitz continuous on any compact convex set $V\subseteq dom(h)$.

(i) If $p:\Re^m\to(-\infty,\infty]$ is a proper, convex, closed function with a polyhedral epigraph. Then the EBP holds.

(ii) If p is the group lasso regularizer, i.e., $p(x)=\sum J\in\mathcal{J} w_J\|x_J\|_2$, where $\mathcal{J}$ is a partition of the index set $\{1,2,\ldots,n\}$, $x_J\in\Re^{|J|}$ is the vector obtained by restricting $x\in\Re^n$ to the entries in $J\in\mathcal{J}$, and $w_J\geq 0$ is a given parameter. Then the EBP holds.

Next, we show the relationships between the error bound property and the calmness property of the solution mapping for the convex composite programming problem (2.19).

The following proposition shows that we can replace the test set in (EBP) by the neighborhood of the solution set $\mathcal{X}$.

Proposition 6. *[88, Proposition 3] Consider the optimization problem (2.19), under the Assumption 2.2, the error bound (EBP) holds if there exists constants $\kappa,\rho>0$ such that*

$$(EBN)\quad dist(x,\mathcal{X})\leq\kappa\|R(x)\|_2,\quad \forall x\in\mathcal{O}\text{ with }dist(x,\mathcal{X})\leq\rho. \tag{2.25}$$

The following results from [88] build the bridge between the calmness of the solution mapping and the error bound properties.

Theorem 2.9. *[88, Proposition 1, Proposition 4, Theorem 1] Consider the optimization problem (2.19), under the Assumption 2.2, there exist a $\bar{y} = \mathcal{A}x$ for all $x \in \mathcal{X}$ such that*

$$\mathcal{X} = \{x \in \mathcal{E} \mid \mathcal{A}x = \bar{y}, -\mathcal{A}^*\nabla h(\bar{y}) - c \in \partial p(x)\}. \tag{2.26}$$

Then the error bound condition (EBN) holds if and only if the solution map $\Gamma : \mathcal{T} \times \mathcal{E} \rightrightarrows \mathcal{E}$ is calm at $(\bar{y}, \mathcal{A}^*\nabla h(\bar{y}) + c)$ for any $\bar{x} \in \Gamma(\bar{y}, \mathcal{A}^*\nabla h(\bar{y}) + c)$.

Chapter 3

Convex Clustering

In this chapter, we will focus on the general weighted convex clustering model. Specifically, for a given data matrix $A \in \Re^{d\times n} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$, we consider the following general weighted convex clustering model

$$\min_{X\in\Re^{d\times n}} \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|^2 + \gamma \sum_{i<j} w_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_p, \tag{3.1}$$

where $p \geq 1$ and $w_{ij} = w_{ji} \geq 0$ are given weights that are generally chosen based on the given input data A.

As discussed in Chapter 1, the general weighted convex clustering model includes the following uniform weighted convex clustering model as a special case:

$$\min_{X\in\Re^{d\times n}} \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|^2 + \gamma \sum_{i<j} \|\mathbf{x}_i - \mathbf{x}_j\|_p. \tag{3.2}$$

This chapter will be organized as follows. We first discuss some related work in section 3.1, some preliminaries could be found in section 3.2. The theoretical recovery guarantee will be shown in section 3.3. In section 3.4, we will present the semismooth Newton-CG based augmented Lagrangian method (SSNAL) for solving (3.1). Convergence analysis of the algorithm SSNAL will also be included. Then, we present all the numerical experiments in section 3.5.

3.1 Related work

In this section, we discuss some additional related work to the convex clustering model. In addition to the papers [17,31,41,54,55,67,89] on the convex models (3.2) and (3.1), other convex models have been proposed to deal with the non-convexity of the K-means clustering model. One such model is the convex relaxation of the K-means model via semidefinite programming (SDP) [5,47,56].

As discussed in Chapter 1, for a given data matrix $A \in \mathbb{R}^{d\times n} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$, the classical K-means model solves the following non-convex optimization problem

$$\begin{array}{ll} \min & \sum_{t=1}^{k} \sum_{i\in I_t} \|\mathbf{a}_i - \frac{1}{|I_t|}\sum_{j\in I_t} \mathbf{a}_j\|^2 \\ \text{s.t.} & I_1, \ldots, I_k \text{ is a partition of } \{1, 2, \ldots, n\}. \end{array} \tag{3.3}$$

Now, if we define the $n \times n$ matrix D by $D_{ij} = \|\mathbf{a}_i - \mathbf{a}_j\|^2$, then by taking

$$X := \sum_{t=1}^{k} \frac{1}{|I_t|} \mathbf{1}_{I_t} \mathbf{1}_{I_t}^T,$$

where $\mathbf{1}_{I_t} \in \mathbb{R}^n$ is the indicator vector of the index set I_t. We can express the objective function in (3.3) as $\frac{1}{2}\mathrm{Tr}(DX)$. Based on this, [56] proposed the following SDP relaxation of the K-means model

$$\min \Big\{ \mathrm{Tr}(DX) \;\mid\; \mathrm{Tr}(X) = k,\; X\mathbf{e} = \mathbf{e},\; X \geq 0,\; X \in \mathbb{S}_n^+ \Big\}, \tag{3.4}$$

where $X \geq 0$ means that all the elements in X are nonnegative, $\mathbb{S}_+^n$ is the cone of $n \times n$ symmetric and positive semidefinite matrices, and $\mathbf{e} \in \mathbb{R}^n$ is the column vector of all ones.

Recently, [47] proved that the K-means SDP relaxation approach can achieve perfect cluster recovery with high probability when the data A is sampled from the stochastic unit-ball model in $\mathbb{R}^d$, provided that the cluster centriods $\{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(k)}\}$ satisfy the condition that $\min\{\|\mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)}\| \mid 1 \leq \alpha < \beta \leq k\} > 2\sqrt{2}(1 + 1/\sqrt{d})$. However, the computational efficiency of SDP based relaxations highly depends on the efficiency of the available SDP solvers. While recent progress [65,77,86] in solving large-scale SDPs allows one to solve the SDP relaxation problem for clustering 2–3

thousand points, it is however prohibitively expensive to solve the problem when n goes beyond 3000.

The work in [16] has implicitly demonstrated that it is generally much cheaper to solve the model (3.1) instead of the SDP relaxation model. However, based on our numerical experiments, the algorithms ADMM and AMA proposed in [16] for solving (3.1) only work efficiently when the number of data points is not too large (several thousands depending on the feature dimension of the data). Also, it is not easy for the proposed algorithms in [16] to achieve relatively high accuracy. This also explains why we need to design a new algorithm in this paper to overcome the aforementioned difficulties.

3.2 Preliminaries and notation

In this section, we first introduce some preliminaries and notation which will be used later in this chapter. For theoretical analysis, we adopt some definitions and notation from [54, 89].

Definition 10. *For a given finite set $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\} \subset \mathbb{R}^d$ and its partitioning $\mathcal{V} = \{V_1, V_2, \ldots, V_K\}$, where each V_i is a subset of A.*

(a) We say that a map ψ on A perfectly recovers $\mathcal{V}$ when $\psi(\mathbf{a}_i) = \psi(\mathbf{a}_j)$ is equivalent to $\mathbf{a}_i$ and $\mathbf{a}_j$ belonging to the same cluster. In other words, there exist distinct vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K$ such that $\psi(\mathbf{a}_i) = \mathbf{v}_\alpha$ holds whenever $\mathbf{a}_i \in V_\alpha$.

(b) We call a partitioning $\mathcal{W} = \{W_1, W_2, \ldots, W_L\}$ of A a coarsening of $\mathcal{V}$ if each partition W_l is obtained by taking the union of a number of partitions in $\mathcal{V}$. Furthermore, $\mathcal{W}$ is called the trivial coarsening of $\mathcal{V}$ if $\mathcal{W} = \{A\}$. Otherwise, it is called a non-trivial coarsening.

Definition 11. *For any finite set $S \subset \Re^d$, its diameter with respect to the q-norm for $q \geq 1$ is defined as*

$$D_q(S) := \max\{\|\mathbf{x} - \mathbf{y}\|_q \mid \mathbf{x}, \mathbf{y} \in S\}.$$

Moreover, we define its separation and centroid, respectively, as

$$d_q(S) := \min\{\|\mathbf{x}-\mathbf{y}\|_q \mid \mathbf{x},\mathbf{y}\in S, \mathbf{x}\neq\mathbf{y}\}, \qquad c(S) = \frac{\sum_{\mathbf{x}\in S}\mathbf{x}}{|S|}.$$

For convenience, for any family of mutually disjoint finite sets $\mathcal{F} = \{F_i \subset \mathbb{R}^d\}$, we define $\mathcal{C}(\mathcal{F}) = \{c(F_i)\}$.

Later in this chapter, we will establish the theoretical recovery guarantee based on the above definitions. Next, we introduce some preliminaries and notations for the design and analysis of the numerical optimization algorithms.

For a given simple undirected graph $\mathcal{G} = (\{1,\ldots,n\},\mathcal{E})$ with n vertices and edges defined in $\mathcal{E}$, we define the symmetric adjacency matrix $G \in \Re^{n\times n}$ with entries

$$G_{ji} = G_{ij} = \begin{cases} 1 & if\ (i,j)\in\mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

Based on an enumeration of the index pairs in $\mathcal{E}$ (say in the lexicographic order), which we denote by $l(i,j)$ for the pair (i,j), we define the node-arc incidence matrix $\mathcal{J} \in \Re^{n\times|\mathcal{E}|}$ as

$$\mathcal{J}_k^{l(i,j)} = \begin{cases} 1 & \text{if } k=i, \\ -1 & \text{if } k=j, \\ 0 & \text{otherwise,} \end{cases} \tag{3.5}$$

where $\mathcal{J}_k^{l(i,j)}$ is the k-th entry of the $l(i,j)$-th column of $\mathcal{J}_k$.

Proposition 7. *With matrices G, $\mathcal{J}$ defined above, we have the following results*

$$\mathcal{J}\mathcal{J}^T = \mathrm{diag}(G\mathbf{e}) - G =: L_G, \tag{3.6}$$

where $\mathbf{e}\in\Re^n$ is the column vector of all ones, and L_G is the Laplacian matrix associated with the adjacency matrix G.

Now, for given variables $X\in\Re^{d\times n}$, $Z\in\Re^{d\times|\mathcal{E}|}$ and the graph G, we define the linear map $\mathcal{B}:\Re^{d\times n}\to\Re^{d\times|\mathcal{E}|}$ and its adjoint $\mathcal{B}^*:\Re^{d\times|\mathcal{E}|}\to\Re^{d\times n}$, respectively, by

$$\mathcal{B}(X) = [(\mathbf{x}_i-\mathbf{x}_j)]_{(i,j)\in\mathcal{E}} = X\mathcal{J}, \tag{3.7}$$

$$\mathcal{B}^*(Z) = Z\mathcal{J}^T. \tag{3.8}$$

Thus, by Proposition 7, we have

$$\mathcal{B}^*(\mathcal{B}(X)) = X\mathcal{J}\mathcal{J}^T = XL_G. \tag{3.9}$$

For a given proper and closed convex function $p : \mathcal{X} \to (-\infty, +\infty]$, its proximal mapping $\mathrm{Prox_{tp}(x)}$ for p at any $x \in \mathcal{X}$ with $t > 0$ is defined by

$$\mathrm{Prox_{tp}(x)} = \arg\min_{\mathrm{u}\in\mathcal{X}}\{\mathrm{tp(u)} + \frac{1}{2}\|\mathrm{u} - \mathrm{x}\|^2\}. \tag{3.10}$$

In this chapter, we will often make use of the following Moreau identity (See [7, Theorem 14.3(ii)])

$$\mathrm{Prox_{tp}(x)} + \mathrm{tProx_{p^*/t}(x/t)} = \mathrm{x},$$

where $t > 0$ and p^* is the conjugate function of p. In particular, if $p(x) = \rho\|x\|_2$, it's not difficult to show that, $p^*(y)$ is the indicator function defined as follows:

$$p^*(y) = \begin{cases} 0 & \text{if } \|y\|_2 \le \rho, \\ +\infty & \text{if } \|y\|_2 > \rho. \end{cases}$$

It is well known that proximal mappings are important for designing optimization algorithms and they have been well studied. The proximal mappings for many commonly used functions have closed form formulas. Here, we summarize those that are related to this chapter in Table 2.1. In the table, $\delta_C(\cdot)$ denotes the indicator function of a given closed convex set C, which is

$$\delta_C(x) = \begin{cases} 0 & \text{if } x \in C, \\ +\infty & \text{if } x \notin C. \end{cases}$$

Π_C denotes the projection onto C.

3.3 Theoretical recovery guarantee of convex clustering models

The empirical success of the convex clustering model (3.2) has strongly motivated researchers to investigate its theoretical clustering recovery guarantee. The perfect

Table 3.1: Proximal maps for selected functions

$p(\cdot)$	$\text{Prox}_{tp}(\mathbf{x})$	Comment
$\|\cdot\|_1$	$\left[1-\frac{t}{\|\mathbf{x}_l\|}\right]_+ \mathbf{x}_l$	Elementwise soft-thresholding
$\|\cdot\|_2$	$\left[1-\frac{t}{\|\mathbf{x}\|_2}\right]_+ \mathbf{x}$	Blockwise soft-thresholding
$\|\cdot\|_\infty$	$\mathbf{x}-\Pi_{t\mathcal{S}}(\mathbf{x})$	$\mathcal{S}$ is the unit ℓ_1-ball
$\delta_C(\cdot)$	$\Pi_C(\mathbf{x})$	Projection

recovery results for convex clustering model (3.2), where all pairwise differences are considered with equal weights, have been proved by [89] for the 2-clusters case and later by [54] for the k-clusters case. [67] analyzed the statistical properties of model (3.2) and [58] analyzed the statistical properties of model (3.2) with the ℓ_1-regularization term. In practice, many researchers (e.g. [16, 67]) have suggested the use of the model (3.1), which is not only computationally more attractive but also lead to more robust clustering results. However, so far no theoretical guarantee has been provided for the convex clustering model with general weights. In this section, we first review the nice theoretical results proved by [89] and [54] for (3.2), and then we will present our new theoretical guarantee for the more challenging case of the general weighted convex clustering model (3.1).

3.3.1 Theoretical recovery guarantee of convex clustering model (3.2)

The first theoretical result by [89] guarantees the perfect recovery of (3.2) for the two-clusters case when the data in each cluster are contained in a cube and the two cubes are sufficiently well separated.

More recently, much stronger theoretical results have been established by [54] wherein the authors proved the theoretical recovery guarantee of the fully uniformly weighted model (3.2) for the general case of k-clusters.

Theorem 3.1 ([54]). *Consider a finite set* $A = \{\mathbf{a}_i \in \mathbb{R}^d \mid i = 1, 2, \ldots, n\}$ *of vectors and its partitioning* $\mathcal{V} = \{V_1, V_2, \ldots, V_K\}$. *For the SON model in (3.2), denote its optimal solution by* $\{\bar{\mathbf{x}}_i\}$ *and define the map* $\phi(\mathbf{a}_i) = \bar{\mathbf{x}}_i$, $i = 1, \ldots, n$.

(i) If γ *is chosen such that*

$$\max_{V \in \mathcal{V}} \frac{D_2(V)}{|V|} \leq \gamma \leq \frac{d_2(\mathcal{C}(\mathcal{V}))}{2n\sqrt{K}},$$

then the map ϕ *perfectly recovers* $\mathcal{V}$.

(ii) If γ *satisfies the following inequalities,*

$$\max_{V \in \mathcal{V}} \frac{D_2(V)}{|V|} \leq \gamma \leq \max_{V \in \mathcal{V}} \frac{\|c(A) - c(V)\|_2}{|A| - |V|},$$

then the map ϕ *perfectly recovers a non-trivial coarsening of* $\mathcal{V}$.

It was shown in [54] that one can treat the theoretical results in [89] as a special case of Theorem 3.1.

We shall see in the next subsection that we can improve the upper bound in part (i) of Theorem 3.1 to $\gamma \leq \frac{d_2(\mathcal{C}(\mathcal{V}))}{2n}$, as a special case of our new theoretical results.

3.3.2 Theoretical recovery guarantee of the weighted convex clustering model (3.1)

Although the convex clustering model (3.2) with the fully uniformly weighted regularization has the nice theoretical recovery guarantee, it is usually computationally too expensive to solve since the number of terms in the regularization grows quadratically with the number of data points n. In order to reduce the computational burden, in practice many researchers have proposed to use the partially weighted convex clustering model (3.1) described in the Introduction. Moreover, they have observed better empirical performance of (3.1) with well chosen weights, comparing to the original model (3.2) [16, 31, 41]. However, to the best of our knowledge, so far no theoretical recovery results have been established for the general weighted convex clustering model (3.1). Here we will prove that under rather

mild conditions, perfect recovery can be guarantee for the weighted model (3.1). In additional, our theoretical results subsume the known results for the fully uniformly weighted model (3.2) as special cases.

Next, we will establish the main theoretical results for (3.1). Our results and part of the proof have been inspired by the ideas used in [54]. For convenience, we define the index sets

$$I_\alpha := \{i \mid \mathbf{a}_i \in V_\alpha\}, \quad \text{for } \alpha = 1, 2, \ldots, K.$$

Let $n_\alpha = |I_\alpha|$,

$$\mathbf{a}^{(\alpha)} = \frac{1}{n_\alpha} \sum_{i \in I_\alpha} \mathbf{a}_i, \quad w^{(\alpha,\beta)} = \sum_{i \in I_\alpha} \sum_{j \in I_\beta} w_{ij}, \quad \forall\, \alpha, \beta = 1, \ldots, K$$

$$w_i^{(\beta)} = \sum_{j \in I_\beta} w_{ij}, \quad \forall\, i = 1, \ldots, n,\ \beta = 1, \ldots, K.$$

Here we will interpret $w_i^{(\beta)}$ as the coupling between point $\mathbf{a}_i$ and the β-th cluster, and $w^{(\alpha,\beta)}$ as the coupling between the α-th and β-th clusters. We also define for $p \geq 1$,

$$h(\mathbf{v}) := \|\mathbf{v}\|_p = \Big(\sum_{i=1}^{d} |v_i|^p \Big)^{\frac{1}{p}}, \quad \mathbf{v} = (v_1, v_2, \ldots, v_d) \in \mathbb{R}^d,$$

and note that the subdifferential of $h(\mathbf{v})$ is given by

$$\partial h(\mathbf{v}) = \begin{cases} \{\mathbf{y} \in \Re^d \mid \|\mathbf{y}\|_q \leq 1, \langle \mathbf{y}, \mathbf{v} \rangle = \|\mathbf{v}\|_p\} & \text{if } \mathbf{v} \neq 0, \\ \{\mathbf{y} \in \Re^d \mid \|\mathbf{y}\|_q \leq 1\} & \text{if } \mathbf{v} = 0, \end{cases}$$

where $q \geq 1$ is the conjugate index of p such that $\frac{1}{p} + \frac{1}{q} = 1$. Observe that for any $\mathbf{y} \in \partial h(\mathbf{v})$, we have $\|\mathbf{y}\|_q \leq 1$.

Theorem 3.2. *Consider an input data $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n] \in \Re^{d \times n}$ and its partitioning $\mathcal{V} = \{V_1, V_2, \ldots, V_K\}$. Assume that all the centroids $\{\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(K)}\}$ are distinct. Let $q \geq 1$ be the conjugate index of p such that $\frac{1}{p} + \frac{1}{q} = 1$. Denote the optimal solution of (3.1) by $\{\mathbf{x}_i^*\}$ and define the map $\phi(\mathbf{a}_i) = \mathbf{x}_i^*$ for $i = 1, \ldots, n$.*

1. *Let*

$$\mu_{ij}^{(\alpha)} := \sum_{\beta=1,\beta\neq\alpha}^{K} \left| w_i^{(\beta)} - w_j^{(\beta)} \right|, \quad i,j \in I_\alpha,\ \alpha = 1,2,\ldots,K.$$

Assume that $w_{ij} > 0$ *and* $n_\alpha w_{ij} > \mu_{ij}^{(\alpha)}$ *for all* $i,j \in I_\alpha$, $\alpha = 1,\ldots,K$. *Let*

$$\begin{aligned} \gamma_{\min} &:= \max_{1\leq\alpha\leq K} \max_{i,j\in I_\alpha} \left\{ \frac{\|\mathbf{a}_i - \mathbf{a}_j\|_q}{n_\alpha w_{ij} - \mu_{ij}^{(\alpha)}} \right\}, \\ \gamma_{\max} &:= \min_{1\leq\alpha<\beta\leq K} \left\{ \frac{\|\mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)}\|_q}{\frac{1}{n_\alpha}\sum_{1\leq l\leq K, l\neq\alpha} w^{(\alpha,l)} + \frac{1}{n_\beta}\sum_{1\leq l\leq K, l\neq\beta} w^{(\beta,l)}} \right\}. \end{aligned} \tag{3.11}$$

If $\gamma_{\min} < \gamma_{\max}$ *and* γ *is chosen such that* $\gamma \in [\gamma_{\min}, \gamma_{\max})$, *then the map* ϕ *perfectly recovers* $\mathcal{V}$.

2. *If* γ *is chosen such that*

$$\gamma_{\min} \ \leq\ \gamma \ <\ \max_{1\leq\alpha\leq K} \frac{n_\alpha \|\mathbf{c} - \mathbf{a}^{(\alpha)}\|_q}{\sum_{1\leq\beta\leq K,\beta\neq\alpha} w^{(\alpha,\beta)}},$$

where $\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{a}_i$, *then the map* ϕ *perfectly recovers a non-trivial coarsening of* $\mathcal{V}$.

Proof. First we introduce the following centroid optimization problem corresponding to (3.1):

$$\min \left\{ \frac{1}{2}\sum_{\alpha=1}^{K} n_\alpha \|\mathbf{x}^{(\alpha)} - \mathbf{a}^{(\alpha)}\|^2 + \gamma \sum_{\alpha=1}^{K}\sum_{\beta=\alpha+1}^{K} w^{(\alpha,\beta)} \|\mathbf{x}^{(\alpha)} - \mathbf{x}^{(\beta)}\|_p \mid \mathbf{x}^{(1)},\ldots,\mathbf{x}^{(K)} \in \Re^d \right\}. \tag{3.12}$$

Denote the optimal solution of (3.12) by $\{\bar{\mathbf{x}}^{(\alpha)} \mid \alpha = 1,2,\ldots,K\}$. The proof will rely on the relationships between (3.1) and (3.12).

(1a) First we show that, if $\gamma < \gamma_{\max}$, then $\bar{\mathbf{x}}^{(\alpha)} \neq \bar{\mathbf{x}}^{(\beta)}$ for all $\alpha \neq \beta$. From the optimality condition of (3.12), we have that

$$n_\alpha(\bar{\mathbf{x}}^{(\alpha)} - \mathbf{a}^{(\alpha)}) + \gamma \sum_{\beta=1,\beta\neq\alpha}^{K} w^{(\alpha,\beta)} \bar{\mathbf{z}}^{(\alpha,\beta)} \ = \ 0, \quad \forall\ \alpha = 1,\ldots,K, \tag{3.13}$$

where $\bar{\mathbf{z}}^{(\alpha,\beta)} \in \partial h(\bar{\mathbf{x}}^{(\alpha)} - \bar{\mathbf{x}}^{(\beta)})$, $\alpha \neq \beta$. Now from (3.13), we get for $\alpha \neq \beta$,

$$\begin{aligned}
\bar{\mathbf{x}}^{(\alpha)} - \bar{\mathbf{x}}^{(\beta)} &= \mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)} - \tfrac{\gamma}{n_\alpha}\textstyle\sum_{l=1,l\neq\alpha}^{K} w^{(\alpha,l)}\bar{\mathbf{z}}^{(\alpha,l)} + \tfrac{\gamma}{n_\beta}\sum_{l=1,l\neq\beta}^{K} w^{(\beta,l)}\bar{\mathbf{z}}^{(\beta,l)} \\
\Rightarrow \quad \|\bar{\mathbf{x}}^{(\alpha)} - \bar{\mathbf{x}}^{(\beta)}\|_q &\geq \|\mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)}\|_q - \tfrac{\gamma}{n_\alpha}\textstyle\sum_{l=1,l\neq\alpha}^{K} w^{(\alpha,l)}\|\bar{\mathbf{z}}^{(\alpha,l)}\|_q - \tfrac{\gamma}{n_\beta}\sum_{l=1,l\neq\beta}^{K} w^{(\beta,l)}\|\bar{\mathbf{z}}^{(\beta,l)}\|_q \\
&\geq \|\mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)}\|_q - \gamma\left(\tfrac{1}{n_\alpha}\textstyle\sum_{l=1,l\neq\alpha}^{K} w^{(\alpha,l)} + \tfrac{1}{n_\beta}\sum_{l=1,l\neq\beta}^{K} w^{(\beta,l)}\right) \\
&\geq \|\mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)}\|_q \left(1 - \tfrac{\gamma}{\gamma_{\max}}\right) > 0.
\end{aligned}$$

Thus $\bar{\mathbf{x}}^{(\alpha)} \neq \bar{\mathbf{x}}^{(\beta)}$ for all $\alpha \neq \beta$.

(1b) Suppose that $\gamma < \gamma_{\max}$. Then from (a), $\bar{\mathbf{x}}^{(\alpha)} \neq \bar{\mathbf{x}}^{(\beta)}$ for all $\alpha \neq \beta$. Next we prove that, if $\gamma \geq \gamma_{\min}$, then

$$\mathbf{x}_i^* = \bar{\mathbf{x}}^{(\alpha)}, \quad \forall\, i \in I_\alpha, \quad \alpha = 1, \ldots, K$$

is the unique optimal solution of (3.1).

To do so, we start with the optimality condition for (3.1), which is given as follows:

$$\mathbf{x}_i - \mathbf{a}_i + \gamma \sum_{j=1,j\neq i}^{n} w_{ij}\mathbf{z}_{ij} = 0, \quad i = 1, 2, \ldots, n, \tag{3.14}$$

where $\mathbf{z}_{ij} \in \partial h(\mathbf{x}_i - \mathbf{x}_j)$. Consider

$$\mathbf{z}_{ij}^* = \begin{cases} \bar{\mathbf{z}}^{(\alpha,\beta)} & \text{if } i \in I_\alpha,\ j \in I_\beta,\ 1 \leq \alpha, \beta \leq K,\ \alpha \neq \beta, \\ \frac{1}{n_\alpha w_{ij}}\left[\frac{1}{\gamma}(\mathbf{a}_i - \mathbf{a}_j) - (\mathbf{p}_i^{(\alpha)} - \mathbf{p}_j^{(\alpha)})\right] & \text{if } i, j \in I_\alpha,\ i \neq j,\ \alpha = 1, \ldots, K, \end{cases}$$

where

$$\mathbf{p}_i^{(\alpha)} = \sum_{\beta=1,\beta\neq\alpha}^{K} \left[w_i^{(\beta)} - \frac{1}{n_\alpha} w^{(\alpha,\beta)}\right] \bar{\mathbf{z}}^{(\alpha,\beta)}.$$

We can readily prove that

$$\|\mathbf{p}_i^{(\alpha)} - \mathbf{p}_j^{(\alpha)}\|_q \leq \mu_{ij}^{(\alpha)}$$

and

$$\begin{aligned}
\sum_{j\in I_\alpha} \mathbf{p}_j^{(\alpha)} &= \sum_{j\in I_\alpha}\left(\sum_{\beta=1,\beta\neq\alpha}^{K}\left[w_j^{(\beta)} - \frac{1}{n_\alpha} w^{(\alpha,\beta)}\right]\bar{\mathbf{z}}^{(\alpha,\beta)}\right) \\
&= \sum_{\beta=1,\beta\neq\alpha}^{K}\left(\sum_{j\in I_\alpha}\left[w_j^{(\beta)} - \frac{1}{n_\alpha} w^{(\alpha,\beta)}\right]\right)\bar{\mathbf{z}}^{(\alpha,\beta)} = \mathbf{0}.
\end{aligned}$$

For convenience, we set $\mathbf{z}_{ii}^* = 0$ for $i = 1, 2, \ldots, n$. Now, we show that $\mathbf{z}_{ij}^* \in \partial h(\mathbf{x}_i^* - \mathbf{x}_j^*)$.

If $i \in I_\alpha$ and $j \in I_\beta$ for $\alpha \neq \beta$, then we have that

$$\mathbf{z}_{ij}^* = \bar{\mathbf{z}}^{(\alpha,\beta)} \in \partial h(\bar{\mathbf{x}}^{(\alpha)} - \bar{\mathbf{x}}^{(\beta)}) = \partial h(\mathbf{x}_i^* - \mathbf{x}_j^*).$$

It remains to show that $\|\mathbf{z}_{ij}^*\|_q \leq 1$ for all $i, j \in I_\alpha, \alpha = 1, 2, \ldots, K$. By direct calculations, we have that for $\gamma \geq \gamma_{\min}$,

$$\begin{aligned}
\|\mathbf{z}_{ij}^*\|_q &= \frac{1}{n_\alpha w_{ij}} \Big\| \frac{1}{\gamma}(\mathbf{a}_i - \mathbf{a}_j) - (\mathbf{p}_i^{(\alpha)} - \mathbf{p}_j^{(\alpha)}) \Big\|_q \leq \frac{1}{\gamma n_\alpha w_{ij}} \|\mathbf{a}_i - \mathbf{a}_j\|_q + \frac{1}{n_\alpha w_{ij}} \mu_{ij}^{(\alpha)} \\
&\leq \frac{1}{n_\alpha w_{ij}} (n_\alpha w_{ij} - \mu_{ij}^{(\alpha)}) + \frac{1}{n_\alpha w_{ij}} \mu_{ij}^{(\alpha)} = 1,
\end{aligned}$$

which implies that $\mathbf{z}_{ij}^* \in \partial h(\mathbf{x}_i^* - \mathbf{x}_j^*) = \partial h(\mathbf{0})$ for all $i, j \in I_\alpha$.

Finally, we show that the optimality condition (3.14) holds for $(\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*)$. We have that for $i \in I_\alpha$,

$$\begin{aligned}
\mathbf{x}_i^* - \mathbf{a}_i + \gamma \sum_{j=1, j\neq i}^{n} w_{ij} \mathbf{z}_{ij}^* &= \bar{\mathbf{x}}^{(\alpha)} - \mathbf{a}_i + \gamma \sum_{\beta=1}^{K} \sum_{j \in I_\beta} w_{ij} \mathbf{z}_{ij}^* \\
&= \bar{\mathbf{x}}^{(\alpha)} - \mathbf{a}^{(\alpha)} + \gamma \sum_{\beta=1, \beta\neq\alpha}^{K} \Big(\sum_{j \in I_\beta} w_{ij} \Big) \bar{\mathbf{z}}^{(\alpha,\beta)} + \mathbf{a}^{(\alpha)} - \mathbf{a}_i + \gamma \sum_{j \in I_\alpha} w_{ij} \mathbf{z}_{ij}^* \\
&= \gamma \sum_{\beta=1, \beta\neq\alpha}^{K} \Big[w_i^{(\beta)} - \frac{1}{n_\alpha} w^{(\alpha,\beta)} \Big] \bar{\mathbf{z}}^{(\alpha,\beta)} + \mathbf{a}^{(\alpha)} - \mathbf{a}_i + \gamma \sum_{j \in I_\alpha} w_{ij} \mathbf{z}_{ij}^* \\
&= \gamma \mathbf{p}_i^{(\alpha)} + \mathbf{a}^{(\alpha)} - \mathbf{a}_i + \frac{\gamma}{n_\alpha} \sum_{j \in I_\alpha} \Big[\frac{1}{\gamma}(\mathbf{a}_i - \mathbf{a}_j) - (\mathbf{p}_i^{(\alpha)} - \mathbf{p}_j^{(\alpha)}) \Big] \\
&= 0.
\end{aligned}$$

Thus $(\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*)$ is the optimal solution of (3.1). Since $\phi(\mathbf{a}_i) = \mathbf{x}_i^* = \bar{\mathbf{x}}^{(\alpha)}$ for all $i \in I_\alpha$, $\alpha = 1, \ldots, K$, we see that the mapping ϕ perfectly recovers the clusters in $\mathcal{V}$.

(2) Suppose on the contrary that $\bar{\mathbf{x}}^{(1)} = \bar{\mathbf{x}}^{(2)} = \cdots = \bar{\mathbf{x}}^{(K)}$. In this case, the second term of (3.1) disappears, so $\bar{\mathbf{x}}$ is the solution of $\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x} - \mathbf{a}_i\|^2$. Which means the optimal solution for (3.12) degenerates to

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{a}_i = \mathbf{c}.$$

Thus, the optimality condition (3.13) gives

$$n_\alpha \|\mathbf{c} - \mathbf{a}^\alpha\|_q \leq \gamma \sum_{\beta=1,\beta\neq\alpha}^{K} w^{(\alpha,\beta)}, \quad \forall\, \alpha \in \{1, 2, \ldots, K\}.$$

This implies that

$$\gamma \geq \max_{1\leq\alpha\leq K} \frac{n_\alpha \|\mathbf{c} - \mathbf{a}^{(\alpha)}\|_q}{\sum_{\beta=1,\beta\neq\alpha}^{K} w^{(\alpha,\beta)}},$$

which is a contradiction. Thus $\{\bar{\mathbf{x}}^{(1)}, \ldots, \bar{\mathbf{x}}^{(K)}\}$ must have a distinct pair. □

The above theorem has established the theoretical recovery guarantee for the general weighted convex clustering model (3.1). Later, we will demonstrate that the sufficient conditions that γ must satisfy is practically meaningful in the numerical experiments section. Now, we explain the derived sufficient conditions intuitively.

For unsupervised learning, intuitively, we can get meaningful clustering results when the given dataset has the properties that the elements within the same cluster are "tight" (in other words, the diameter should be small) and the centroids for different clusters are well separated. Indeed, the conditions we have established are consistent with the intuition just discussed. First, the left-hand side in (3.11) characterizes the maximum weighted distance between the elements in the same cluster. On the other hand, the right-hand side in (3.11) characterizes the minimum weighted distance between different centroids. Thus based on our discussion, we can expect perfect recovery to be practically possible for the weighted convex clustering model if the right-hand side is larger than the left-hand side in (3.11).

Remark 3.1. *(a) Note that the assumption that $w_{ij} > 0$ is only needed for all the pairs (i, j) belonging to the same cluster I_α for all $1 \leq \alpha \leq K$. Thus the weights w_{ij} can be chosen to be zero if i and j belong to different clusters. As a result, the number of pairwise differences in the regularization term can be much fewer than the total of $n(n-1)/2$ terms. This implies that we can gain substantial computational efficiency when dealing with the sparse weighted regularization term.*

(b) The quantity $\mu_{ij}^{(\alpha)} = \sum_{\beta=1,\beta\neq\alpha}^{K} |w_i^{(\beta)} - w_j^{(\beta)}|$, for $i, j \in I_\alpha$, measures the total difference in the couplings between $\mathbf{a}_i$ and $\mathbf{a}_j$ with the β-th cluster for all $\beta \neq \alpha$.

Next, we show that the results in Theorem 3.1 are special cases of our results. Therefore, we also include the result in [89] as a special case.

Corollary 3.1. *In (3.1), if we take $w_{ij} = 1$ for all $1 \le i < j \le n$, then the results in Theorem 3.2 reduce to the following.*

(i) If

$$\max_{1\le\alpha\le K} \frac{D_q(V_\alpha)}{|V_\alpha|} \;\le\; \gamma \;<\; \min_{1\le\alpha,\beta\le K,\alpha\neq\beta} \Big\{ \frac{\|\mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)}\|_q}{2n - n_\alpha - n_\beta} \Big\},$$

then the map ϕ perfectly recovers $\mathcal{V}$.

(ii) If

$$\max_{1\le\alpha\le K} \frac{D_q(V_\alpha)}{|V_\alpha|} \;\le\; \gamma \;\le\; \max_{V\in\mathcal{V}} \frac{\|c(A) - c(V)\|_q}{|A| - |V|},$$

then the map ϕ perfectly recovers a non-trivial coarsening of $\mathcal{V}$.

Proof. The results for this corollary follow directly from Theorem 3.2 by noting that $D_q(V_\alpha) = \max_{i,j\in I_\alpha} \|\mathbf{a}_i - \mathbf{a}_j\|_q / n_\alpha$, and using the following facts for the special case:

(1) $\mu_{ij}^{(\alpha)} = \sum_{\beta=1,\beta\neq\alpha}^{K} |w_i^{(\beta)} - w_j^{(\beta)}| = \sum_{\beta=1,\beta\neq\alpha}^{K} |n_\beta - n_\beta| = 0$, for all $i, j \in I_\alpha$, $1 \le \alpha \le K$.

(2) $\frac{1}{n_\alpha} \sum_{\beta=1,\beta\neq\alpha}^{K} w^{(\alpha,\beta)} = \frac{1}{n_\alpha} \sum_{\beta=1,\beta\neq\alpha}^{K} n_\alpha n_\beta = n - n_\alpha$, for all $1 \le \alpha \le K$.

We omit the details here. □

If we compare the upper bound we obtained for γ in part (i) of Corollary 3.1 to that obtained in Theorem 3.1 of [54] for the case $p = 2$ (and hence $q = 2$), we can see that our upper bound is more relax in the sense that

$$\min_{1\le\alpha,\beta\le K,\alpha\neq\beta} \Big\{ \frac{\|\mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)}\|_2}{2n - n_\alpha - n_\beta} \Big\} \;>\; \min_{1\le\alpha,\beta\le K,\alpha\neq\beta} \Big\{ \frac{\|\mathbf{a}^{(\alpha)} - \mathbf{a}^{(\beta)}\|_2}{2n} \Big\} = \frac{d_2(\mathcal{C}(\mathcal{V}))}{2n} \;\ge\; \frac{d_2(\mathcal{C}(\mathcal{V}))}{2n\sqrt{K}}.$$

Remark 3.2. *(i) More recently, Xu et al. [75] investigated the theoretical guarantee for perfect recovery for the following weighted convex clustering model based on ℓ_1 norm and guassian kernel weights:*

$$\min_{X\in\Re^{d\times n}} \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|^2 + \gamma \sum_{i<j} e^{-\phi\|\mathbf{a}_i - \mathbf{a}_j\|} \|\mathbf{x}_i - \mathbf{x}_j\|_1, \tag{3.15}$$

where $\phi > 0$ is a given constant. It's not difficult to realize that, model (3.15) is component-wise separable. This makes the model difficult to do the clustering task well in some scenario since each component may result in different membership. Of course, on the other hand, since the model is component-wise seperable, solving model (3.15) is relatively easier.

(ii) Recently, Jiang et al. [33] analyzed the recovery property of (3.2) for the mixture gaussian model. They provide the theoretical guarantee for convex clustering model on the mixture gaussian model when the number of data points increases to infinity.

3.4 A semismooth Newton-CG augmented Lagrangian method

In this section, we introduce a fast convergent ALM for solving the weighted convex clustering model (3.1). For simplicity, we will first focus on designing a highly efficient algorithm to solve (3.1) with $p = 2$. The other cases can be done in a similar way. In particular, the same algorithmic design and implementation can be applied to the case $p = 1$ or $p = \infty$ without much difficulty.

3.4.1 Duality and optimality conditions

In this chapter, we will focus on the following weighted convex clustering model with the 2-norm:

$$\min_{X\in\mathbb{R}^{d\times n}} \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|^2 + \gamma \sum_{i<j} w_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2.$$

By ignoring the terms with $w_{ij} = 0$, we consider the following problem:

$$\min_{X\in\mathbb{R}^{d\times n}} \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|^2 + \gamma \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2, \tag{3.16}$$

where $\mathcal{E} := \{(i,j) \mid w_{ij} > 0\}$.

Now, we present the dual problem of (3.16) and its Karush-Kuhn-Tucker (KKT) conditions. First, we write (3.16) equivalently in the following compact form

$$(P) \quad \min_{X,U} \Big\{ \frac{1}{2}\|X - A\|^2 + p(U) \mid \mathcal{B}(X) - U = 0 \Big\},$$

where $p(U) = \gamma \sum_{(i,j)\in\mathcal{E}} w_{ij}\|U^{l(i,j)}\|$ and $\mathcal{B}$ is the linear map defined in (3.7). Here $U^{l(i,j)}$ denotes the $l(i,j)$-th column of $U \in \mathbb{R}^{d\times|\mathcal{E}|}$. The dual problem for (P) is given by

$$(D) \quad \max_{V,Z} \Big\{ \langle A, V\rangle - \frac{1}{2}\|V\|^2 \;\mid\; \mathcal{B}^*(Z) - V = 0, Z \in \Omega \Big\},$$

where $\Omega = \{Z \in \mathbb{R}^{d\times|\mathcal{E}|} \mid \|Z^{l(i,j)}\| \le \gamma w_{ij}, (i,j) \in \mathcal{E}\}$. The KKT conditions for (P) and (D) are given by

$$(KKT) \quad \begin{cases} V + X - A &= 0, \\ U - \mathrm{Prox}_p(U + Z) &= 0, \\ \mathcal{B}(X) - U &= 0, \\ \mathcal{B}^*(Z) - V &= 0. \end{cases}$$

3.4.2 A semismooth Newton-CG augmented Lagrangian method for Solving (P)

In this section, we will design an inexact ALM for solving the primal problem (P) but it will also solve (D) as a byproduct.

We begin by defining the following Lagrangian function for (P):

$$l(X,U;Z) \;=\; \frac{1}{2}\|X - A\|^2 + p(U) + \langle Z, \mathcal{B}(X) - U\rangle. \tag{3.17}$$

For a given parameter $\sigma > 0$, the augmented Lagrangian function associated with (P) is given by

$$\mathcal{L}_\sigma(X,U;Z) \;=\; l(X,U;Z) + \frac{\sigma}{2}\|\mathcal{B}(X) - U\|^2.$$

The algorithm for solving (P) is described in **Algorithm** 6. To ensure the convergence of the inexact ALM in **Algorithm** 6, we need the following stopping criterion

for solving the subproblem (3.19) in each iteration:

$$(A) \quad \mathrm{dist}(0, \partial\Phi_k(X^{k+1}, U^{k+1})) \le \epsilon_k / \max\{1, \sqrt{\sigma_k}\}, \tag{3.18}$$

where $\{\epsilon_k\}$ is a given summable sequence of nonnegative numbers.

Algorithm 6: SSNAL for (P)

Input: Choose $(X^0, U^0) \in \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times|\mathcal{E}|}$, $Z^0 \in \mathbb{R}^{d\times|\mathcal{E}|}$, $\sigma_0 > 0$ and a summable nonnegative sequence $\{\epsilon_k\}$.

Iterate until convergence,

Step 1 . Compute

$$(X^{k+1}, U^{k+1}) \approx \arg\min\{\Phi_k(X, U) = \mathcal{L}_{\sigma_k}(X, U; Z^k) \mid X \in \Re^{d\times n},\ U \in \Re^{d\times|\mathcal{E}|}\} \tag{3.19}$$

to satisfy the condition (A) with the tolerance ϵ_k.

Step 2 . Compute

$$Z^{k+1} = Z^k + \sigma_k(\mathcal{B}(X^{k+1}) - U^{k+1}).$$

Step 3 . Update $\sigma_{k+1} \uparrow \sigma_\infty \le \infty$.

Since a semismooth Newton-CG method will be used to solve the subproblems involved in the above ALM method, we call our algorithm a semismooth Newton-CG augmented Lagrangian method (SSNAL in short).

3.4.3 Solving the subproblem (3.19)

The inexact ALM is a well studied algorithmic framework for solving convex composite optimization problems. The key challenge in making the ALM efficient numerically is in solving the subproblem (3.19) in each iteration efficiently to the required accuracy. Next, we will design a semismooth Newton-CG method to solve (3.19). We will establish its quadratic convergence and develop sophisticated numerical techniques to solve the associated semismooth Newton equations very efficiently

by exploiting the underlying second-order structured sparsity in the subproblems. For a given σ and $\tilde{Z}$, the subproblem (3.19) in each iteration has the following form:

$$\min_{X\in\Re^{d\times n},U\in\Re^{d\times|\mathcal{E}|}} \Phi(X,U) := \mathcal{L}_\sigma(X,U;\tilde{Z}). \tag{3.20}$$

Since $\Phi(\cdot,\cdot)$ is a strongly convex function, the level set $\{(X,U)|\Phi(X,U)\le\alpha\}$ is a closed and bounded convex set for any $\alpha\in\Re$ and problem (3.20) admits a unique optimal solution which we denote as $(\bar{X},\bar{U})$. Now, for any X, denote

$$\begin{aligned}\phi(X) &:= \inf_U \Phi(X,U) = \tfrac{1}{2}\|X-A\|^2 + \inf_U\Big\{p(U)+\tfrac{\sigma}{2}\|U-\mathcal{B}(X)-\sigma^{-1}\tilde{Z}\|^2\Big\} - \tfrac{1}{2\sigma}\|\tilde{Z}\|^2\\ &= \tfrac{1}{2}\|X-A\|^2 + p(\mathrm{Prox}_{p/\sigma}(\mathcal{B}(X)+\sigma^{-1}\tilde{Z})) + \tfrac{1}{2\sigma}\|\mathrm{Prox}_{\sigma p^*}(\sigma\mathcal{B}(X)+\tilde{Z})\|^2 - \tfrac{1}{2\sigma}\|\tilde{Z}\|^2.\end{aligned}$$

Therefore, we can compute $(\bar{X},\bar{U}) = \arg\min\Phi(X,U)$ by first computing

$$\bar{X} = \arg\min_X \phi(X),$$

and then compute $\bar{U} = \mathrm{Prox}_{p/\sigma}(\mathcal{B}(\bar{X})+\sigma^{-1}\tilde{Z})$. Since $\phi(\cdot)$ is strongly convex and continuously differentiable on $\Re^{d\times n}$ with

$$\nabla\phi(X) = X - A + \mathcal{B}^*(Prox_{\sigma p^*}(\sigma\mathcal{B}(X)+\tilde{Z})), \tag{3.21}$$

we know that $\bar{X}$ can be obtained by solving the following nonsmooth equation

$$\nabla\phi(X) = 0. \tag{3.22}$$

It is well known that for solving smooth nonlinear equations, the quadratically convergent Newton's method is usually the first choice if it can be implemented efficiently. However, the usually required smoothness condition on $\nabla\phi(\cdot)$ is not satisfied in our problem. This motivates us to develop a semismooth Newton method to solve the nonsmooth equation (3.22). Before we present our semismooth Newton method, we introduce the following definition of semismoothness, adopted from [37, 46, 57], which will be useful for analysis.

Definition 12. *(Semismoothness). For a given open set $\mathcal{O}\subseteq\mathbb{R}^n$, let $F:\mathcal{O}\to\mathbb{R}^m$ be a locally Lipschitz continuous function and $\mathcal{G}:\mathcal{O}\rightrightarrows\mathbb{R}^{m\times n}$ be a nonempty compact*

valued upper-semicontinuous multifunction. F is said to be semismooth at $x \in \mathcal{O}$ with respect to the multifunction $\mathcal{G}$ if F is directionally differentiable at x and for any $V \in \mathcal{G}(x + \Delta x)$ with $\Delta x \to 0$,

$$F(x + \Delta x) - F(x) - V\Delta x = o(\|\Delta x\|).$$

F is said to be strongly semismooth at $x \in \mathcal{O}$ with respect to $\mathcal{G}$ if it is semismooth at x with respect to $\mathcal{G}$ and

$$F(x + \Delta x) - F(x) - V\Delta x = O(\|\Delta x\|^2).$$

F is said to be a semismooth (respectively, strongly semismooth) function on $\mathcal{O}$ with respect to $\mathcal{G}$ if it is semismooth (respectively, strongly semismooth) everywhere in $\mathcal{O}$ with respect to $\mathcal{G}$.

The following lemma shows that the proximal mapping of the 2-norm is strongly semismooth with respect to its Clarke generalized Jacobian (See [18] [Definition 2.6.1] for the definition of the Clarke generalized Jacobian).

Lemma 3.1 ([83], Lemma 2.1)**.** *For any $t > 0$, the proximal mapping $\mathrm{Prox}_{t\|\cdot\|_2}$ is strongly semismooth with respect to the Clarke generalized Jacobian $\partial \mathrm{Prox}_{t\|\cdot\|_2}(\cdot)$.*

Next we derive the generalized Jacobian of the locally Lipschitz continuous function $\nabla\phi(\cdot)$. For any given $X \in \Re^{d\times n}$, the following set-valued map is well defined:

$$\begin{aligned} \hat{\partial}^2\phi(X) &:= \{\mathcal{I} + \sigma\mathcal{B}^*\mathcal{V}\mathcal{B} \mid \mathcal{V} \in \partial Prox_{\sigma p^*}(\tilde{Z} + \sigma\mathcal{B}X)\} \\ &= \{\mathcal{I} + \sigma\mathcal{B}^*(\mathcal{I} - \mathcal{P})\mathcal{B} \mid \mathcal{P} \in \partial Prox_{p/\sigma}(\tfrac{1}{\sigma}\tilde{Z} + \mathcal{B}X)\}, \end{aligned} \tag{3.23}$$

where $\partial Prox_{\sigma p^*}(\tilde{Z} + \sigma\mathcal{B}X)$ and $\partial Prox_{p/\sigma}(\frac{1}{\sigma}\tilde{Z} + \mathcal{B}(X))$ are the Clarke generalized Jacobians of the Lipschitz continuous mappings $Prox_{\sigma p^*}(\cdot)$ and $Prox_{p/\sigma}(\cdot)$ at $\tilde{Z} + \sigma\mathcal{B}X$ and $\frac{1}{\sigma}\tilde{Z} + \mathcal{B}X$, respectively. Note that from [18] [p.75] and [30] [Example 2.5], we have that

$$\partial^2\phi(X)(d) = \hat{\partial}^2\phi(X)(d), \quad \forall d \in \Re^{d\times n},$$

where $\partial^2\phi(X)$ is the generalized Hessian of ϕ at X. Thus, we may use $\hat{\partial}^2\phi(X)$ as the surrogate for $\partial^2\phi(X)$. Since $\mathcal{I}-\mathcal{P}=\mathcal{V}\in\partial Prox_{\sigma p^*}(\cdot)$ is symmetric and positive semidefinite, the elements in $\hat{\partial}^2\phi(X)$ are positive definite, which guarantees that (3.24) in **Algorithm** 7 is well defined. Now, we can present our semismooth Newton-CG (SSNCG) method for solving (3.22) and we could expect to get a fast superlinear or even quadratic convergence.

Algorithm 7: SSNCG for (3.22)

Given $X^0\in\Re^{d\times n}$, $\mu\in(0,1/2)$, $\tau\in(0,1]$, and $\bar{\eta},\delta\in(0,1)$. For $j=0,1,\ldots,$ repeats until convergence,

Step 1 . Pick an element $\mathcal{H}_j$ in $\hat{\partial}^2\phi(X^j)$ that is defined in (3.23). Apply the conjugate gradient (CG) method to find an approximate solution $d^j\in\mathbb{R}^{d\times n}$ to

$$\mathcal{H}_j(d)\approx-\nabla\phi(X^j) \tag{3.24}$$

such that $\|\mathcal{H}_j(d^j)+\nabla\phi(X^j)\|\le\min(\bar{\eta},\|\nabla\phi(X^j)\|^{1+\tau})$.

Step 2 . (Line Search) Set $\alpha_j=\delta^{m_j}$, where m_j is the first nonnegative integer m for which

$$\phi(X^j+\delta^m d^j)\le\phi(X^j)+\mu\delta^m\langle\nabla\phi(X^j),d^j\rangle.$$

Step 3 . Set $X^{j+1}=X^j+\alpha_j d^j$.

To close this section, we discuss an implementable stopping criteria for (3.22) in the algorithm SSNAL. Note that if the algorithm SSNCG is applied to solve the optimization problem given by

$$X^{k+1}=\arg\min_X\phi_k(X)\quad\text{and}\quad U^{k+1}=\mathrm{Prox}_{p/\sigma}(\mathcal{B}(X^{k+1}+\sigma^{-1}Z^k)),$$

where $\phi_k(\cdot):=\inf_U\Phi_k(X,U)$, we have

$$(\nabla\phi_k(X^{k+1}),0)\in\partial\Phi_k(X^{k+1},U^{k+1}).$$

Therefore, the stopping criteria (A) in (3.18) could be achieved by the following implementable stopping criteria

$$\text{(A')} \quad \|\nabla\phi_k(X^{k+1})\| \le \epsilon_k/\max(1,\sqrt{\sigma_k}), \quad \sum_{k=0}^{\infty} < \infty. \tag{3.25}$$

3.4.4 Using the conjugate gradient method to solve (3.24)

In this section, we will discuss how to solve the very large (of dimension $dn \times dn$) symmetric positive definite linear system (3.24) to compute the Newton direction efficiently. As the matrix representation of the coefficient linear operator $\mathcal{H}_j$ in (3.24) is expensive to compute and factorize, we will adopt the conjugate gradient (CG) method to solve it. It is well known that the convergence rate of the CG method depends critically on the condition number of the coefficient matrix. Fortunately for our linear system (3.24), the coefficient linear operator typically has a moderate condition number since it satisfies the following condition:

$$I \preceq \mathcal{V}_j \preceq I + \sigma\mathcal{B}^*\mathcal{B} \preceq (1+\sigma\lambda_{\max}(L_G))I,$$

where $\lambda_{\max}(L_G)$ denotes the maximum eigenvalue of the Laplacian matrix L_G of the graph $\mathcal{G}$, and the notation "$A \preceq B$" means that $B - A$ is symmetric positive semidefinite. It is known from [2] that $\lambda_{\max}(G)$ is at most 2 times the maximum degree of the graph. In the numerical experiments, the maximum degree of the graph is roughly equal to the number of k nearest neighbors. In those cases, the condition number of $\mathcal{V}_j$ is bounded independent of dn, and provided that σ is not too large, we can expect the CG method to converge rapidly even when n and/or d are large.

The computational cost for each CG step is highly dependent on the cost for computing the matrix-vector product $\mathcal{H}_j(\tilde{d})$ for any given $\tilde{d} \in \mathbb{R}^{d\times n}$. Thus we will need to analyze how this product can be computed efficiently. Let $D := \mathcal{B}X^j + \sigma^{-1}\tilde{Z}$. For $(i,j) \in \mathcal{E}$, define

$$\alpha_{ij} = \begin{cases} \frac{\sigma^{-1}\gamma w_{ij}}{\|D^{l(i,j)}\|} & \text{if } \|D^{l(i,j)}\| > 0, \\ \infty & \text{otherwise.} \end{cases}$$

Note that for the given $D \in \mathbb{R}^{d\times|\mathcal{E}|}$, the cost for computing α is $O(d|\mathcal{E}|)$ arithmetic operations. For later convenience, denote

$$\widehat{\mathcal{E}} = \{(i,j) \in \mathcal{E} \mid \alpha_{ij} < 1\}.$$

Now we choose $\mathcal{P} \in \partial \mathrm{Prox}_{p/\sigma}(D)$ explicitly. We can take $\mathcal{P} : \mathbb{R}^{d\times|\mathcal{E}|} \to \mathbb{R}^{d\times|\mathcal{E}|}$ that is defined by

$$(\mathcal{P}(U))^{l(i,j)} = \begin{cases} \alpha_{ij}\frac{\langle D^{l(i,j)}, U^{l(i,j)}\rangle}{\|D^{l(i,j)}\|^2} D^{l(i,j)} + (1-\alpha_{ij})U^{l(i,j)} & \text{if } (i,j) \in \widehat{\mathcal{E}}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus to compute $\mathcal{H}_j(X) = X + \sigma\mathcal{B}^*\mathcal{B}(X) - \sigma\mathcal{B}^*\mathcal{P}\mathcal{B}(X) = X(I_n + \sigma L_G) - \sigma\mathcal{B}^*\mathcal{P}\mathcal{B}(X)$ efficiently for a given $X \in \mathbb{R}^{d\times n}$, we need the efficient computation of $\mathcal{B}^*\mathcal{P}\mathcal{B}(X)$ by using the following proposition.

Proposition 8. *Let $X \in \mathbb{R}^{d\times n}$ be given.*
(a) Consider the symmetric matrix $M \in \mathbb{R}^{n\times n}$ defined by $M_{ij} = 1 - \alpha_{ij}$ if $(i,j) \in \widehat{\mathcal{E}}$ and $M_{ij} = 0$ otherwise. Let $Y = [M_{ij}(\mathbf{x}_i - \mathbf{x}_j)]_{(i,j)\in\mathcal{E}} = X\mathcal{M}$, where $\mathcal{M}$ is defined similarly as in (3.5) for the matrix M. Then we have

$$\mathcal{B}^*(Y) = XL_M,$$

where L_M is the Laplacian matrix associated with M. The cost of computing the result $\mathcal{B}^*(Y)$ is $O(d|\widehat{\mathcal{E}}|)$ arithmetic operations.

(b) Define $\rho \in \mathbb{R}^{|\mathcal{E}|}$ by

$$\rho_{l(i,j)} := \begin{cases} \frac{\alpha_{ij}}{\|D^{l(i,j)}\|^2}\langle D^{l(i,j)}, \mathbf{x}_i - \mathbf{x}_j\rangle, & \text{if } (i,j) \in \widehat{\mathcal{E}}, \\ 0, & \text{otherwise.} \end{cases}$$

For the given $D \in \mathbb{R}^{d\times|\mathcal{E}|}$, the cost for computing ρ is $O(d|\widehat{\mathcal{E}}|)$ arithmetic operations. Let $W^{l(i,j)} = \rho_{l(i,j)}D^{l(i,j)}$. Then,

$$\mathcal{B}^*(W) = W\mathcal{J}^T = D\mathrm{diag}(\rho)\mathcal{J}^T.$$

(c) The computing cost for $\mathcal{B}^*\mathcal{P}\mathcal{B}(X) = \mathcal{B}^*(Y) + \mathcal{B}^*(W)$ in total is $O(d|\widehat{\mathcal{E}}|)$.

With the above proposition, we can readily see that $\mathcal{V}_j(X)$ can be computed in $O(d|\mathcal{E}|)+O(d|\widehat{\mathcal{E}}|)$ operations, where the first term comes from computing $X(I+\sigma L_G)$ and the second term comes from computing $\sigma\mathcal{B}\mathcal{P}\mathcal{B}^*(X)$ based on Proposition 8.

Besides the algorithmic aspect, the next remark shows that the second-order information gathered in the semismooth Newton method can capture data points which are near to the boundary of a cluster if we wisely choose the weights w_{ij}. We believe this is a very useful result since boundary points detection is a challenging problem in practice, especially in the high dimensional setting where locating boundary points is challenging even if we know the labels of all the data points.

Remark 3.3. *If we choose the weights based on the k-nearest neighbors, for example, set*

$$w_{ij} = \begin{cases} \exp(-\phi\|\mathbf{a}_i - \mathbf{a}_j\|^2) & if\ (i,j) \in \mathcal{E}, \\ 0 & otherwise, \end{cases}$$

where $\mathcal{E} = \cup_{i=1}^n\{(i,j) \mid j$ *is among* i*'s* k*-nearest neighbors,* $i < j \leq n\}$*. Then,* $\alpha_{ij} < 1$ *means that* j *is among* i*'s* k*-nearest neighbors but do not belong to the same cluster as* i*. Naturally we expect there will only be a small number of such occurrences if* γ *is properly chosen. Hence,* $|\widehat{\mathcal{E}}|$ *is expected to be much smaller than* $|\mathcal{E}|$*. On the other hand, for* $\alpha_{ij} \geq 1$*, it means that points* i *and* j *are in the same cluster. This result implies that after we have solved the optimization problem (3.1) with a properly selected* γ*,* $\alpha_{ij} < 1$ *indicates that point* i *is near to the boundary of its cluster. Also, we can expect most of the columns of the matrix* $\mathcal{P}(\mathcal{B}(X))$ *to be zero since its number of non-zero columns is at most* $|\widehat{\mathcal{E}}|$*. We call such a property inherited from the generalized Hessian of* $\phi(\cdot)$ *at* X *as the* **second-order sparsity**. *This also explains why we are able to compute* $\mathcal{B}^*\mathcal{P}\mathcal{B}(X)$ *at a very low cost.*

3.4.5 Convergence results

In this section, we will establish the convergence results for both SSNAL and SSNCG under mild assumptions. First, we present the following global convergence result of our proposed Algorithm SSNAL.

Theorem 3.3. *Let $\{(X^k, U^k, Z^k)\}$ be the sequence generated by Algorithm 6 with stopping criterion (A). Then the sequence $\{X^k\}$ converges to the unique optimal solution of (P), and $\|\mathcal{B}(X^k) - U^k\|$ converges to 0. In addition, $\{Z^k\}$ is converges to an optimal solution $Z^* \in \Omega$ of (D).*

The above convergence theorem can be obtained from [62, 63] without much difficulties. Next, we state the convergence property for the semismooth Newton algorithm SSNCG used to solve the subproblems in Algorithm 6.

Theorem 3.4. *Let the sequence $\{X^j\}$ be generated by Algorithm* SSNCG*. Then $\{X^j\}$ converges to the unique solution $\bar{X}$ of the problem in (3.22), and for j sufficiently large,*

$$\|X^{j+1} - \bar{X}\| = O(\|X^j - \bar{X}\|^{1+\tau}),$$

where $\tau \in (0, 1]$ is a given constant in the algorithm, which is typically chosen to be 0.5.

Proof. From Lemma 3.1, we know that $\mathrm{Prox}_{t\|\cdot\|_2}$ is strongly semismooth for any $t > 0$, together with the Moreau identity $\mathrm{Prox}_{tp}(x) + t\mathrm{Prox}_{p^*/t}(x/t) = x$, we know that

$$\nabla\phi(X) = X - A + \mathcal{B}^*(\mathrm{Prox}_{\sigma p^*}(\sigma\mathcal{B}(X) + \tilde{Z})),$$

is strongly semismooth. By [86] [Proposition 3.3], we know that d^j obtained in SSNCG is a descent direction, which guarantees that the Algorithm SSNCG is well defined. From [86] [Theorem 3.4, 3.5], we can get the desired convergence results. □

3.4.6 Generating an initial point

In our implementation, we use the inexact alternating direction method of multipliers in Algorithm 8, which is developed in [13], to generate an initial point to warm-start SSNAL. Note that with the global convergence result stated in Theorem 3.3, the performance of SSNAL does not sensitively depend on the initial points, but it is still helpful if we can choose a good one.

Algorithm 8: iADMM for (P)

Choose $\sigma > 0$, $(X^0, U^0, Z^0) \in \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times|\mathcal{E}|} \times \mathbb{R}^{d\times|\mathcal{E}|}$, and a summable nonnegative sequence $\{\epsilon_k\}$. For $k = 0, 1, \ldots,$ repeat until converges:

Step 1 . Let $R^k = A + \sigma\mathcal{B}^*(U^k - \sigma^{-1}Z^k)$. Compute

$$\begin{aligned} X^{k+1} &\approx \arg\min_X\{\mathcal{L}_\sigma(X, U^k; Z^k)\}, \\ U^{k+1} &= \arg\min_U\{\mathcal{L}_\sigma(X^{k+1}, U; Z^k)\}, \end{aligned}$$

where X^{k+1} is an inexact solution satisfying the accuracy requirement that $\|(I_n + \sigma\mathcal{B}^*\mathcal{B})X^{k+1} - R^k\| \le \epsilon_k$.

Step 2 . Compute

$$Z^{k+1} = Z^k + \tau\sigma(\mathcal{B}(X^{k+1}) - U^{k+1}),$$

where $\tau \in (0, \frac{1+\sqrt{5}}{2})$ is typically chosen to be 1.618.

Observe that in Step 1, X^{k+1} is a computed solution for the following large linear system of equations:

$$(I_n + \sigma\mathcal{B}^*\mathcal{B})X = R^k \quad \Longleftrightarrow \quad (I_n + \sigma L_G)X^T = (R^k)^T.$$

To compute X^{k+1}, we can adopt a direct approach if the sparse Cholesky factorization of $I_n + \sigma L_G$ (which only needs to be done once) can be computed at a moderate cost; otherwise we can adopt an iterative approach by applying the conjugate gradient method to solve the above fairly well-conditioned linear system.

3.5 Numerical experiments

In this section, we will first demonstrate that the sufficient conditions we derived for perfect recovery in Theorem 3.2 is practical via a simulated example. Then, we will show the superior performance of our proposed algorithm Ssnal on both simulated and real datasets, comparing to the popular algorithms such as ADMM

and AMA which are proposed in [16]. In particular, we will focus on the efficiency, scalability, and robustness of our algorithm for different values of γ. Also, we will show the performance of our algorithm on large datasets and unbalanced data. Previous numerical demonstration on the scalability and performance of (3.1) on large datasets is limited. The problem sizes of the instances tested in [16] and other related papers are at most several hundreds ($n \leq 500$ in [16], $n \leq 600$ in [54]), which are not large enough to conclusively demonstrate the scalability of the algorithms. In this paper, we will present numerical results for n up to $\mathbf{200,000}$. We will also analyze the sensitivity of the computational efficiency of SSNAL and AMA, with respect to different choices of the parameters in (3.1), such as k (the number of nearest neighbors) and γ.

We focus on solving (3.1) with $p = 2$ since the rotational invariance of the 2-norm makes it a robust choice in practice. Also, this case is more challenging than $p = 1$ or $p = \infty$.[1] As the results reported in [16] have been regarded as the benchmark for the convex clustering model (3.1), we will compare our algorithm with the open source software CVXCLUSTR[2] in [16], which is an R package with key functions written in C. We write our code in MATLAB without any dedicated C functions. All our computational results are obtained from a desktop having 16 cores with 32 Intel Xeon E5-2650 processors at 2.6 GHz and 64 GB memory.

In our implementation, we stop our algorithm based on the following relative KKT residual:

$$\max\{\eta_P, \eta_D, \eta\} \leq \epsilon,$$

where

$$\eta_P = \frac{\|\mathcal{B}X - U\|}{1 + \|U\|}, \ \eta_D = \frac{\sum_{(i,j)\in\mathcal{E}} \max\{0, \|Z^{l(i,j)}\|_2 - \gamma w_{ij}\}}{1 + \|A\|},$$
$$\eta = \frac{\|\mathcal{B}^*(Z) + X - A\| + \|U - \mathrm{Prox}_p(U + Z)\|}{1 + \|A\| + \|U\|},$$

[1] Our algorithm can be generalized to solve (3.1) with $p = 1$ and $p = \infty$ without much difficulty.

[2] https://cran.r-project.org/web/packages/cvxclustr/index.html

and $\epsilon > 0$ is a given tolerance. In our experiments, we set $\epsilon = 10^{-6}$ unless specified otherwise. Since the numerical results reported in [16] have demonstrated the superior performance of AMA over ADMM, we will mainly compare our proposed algorithm with AMA. We note that CVXCLUSTR does not use the relative KKT residual as its stopping criterion but used the duality gap in AMA and $\max\{\eta_P, \eta_D\} \leq \epsilon$ in ADMM. To make a fair comparison, we first solve (3.1) using SSNAL with a given tolerance ϵ, and denote the primal objective value obtained as P_{Ssnal}. Then, we run AMA in CVXCLUSTR and stop it as soon as the computed primal objective function value (P_{AMA}) is close enough to P_{Ssnal}, i.e.,

$$P_{\text{AMA}} - P_{\text{Ssnal}} \leq 10^{-6} P_{\text{Ssnal}}. \tag{3.26}$$

We note that since (3.1) is an unconstrained problem, the quality of the computed solutions can directly be compared based on the objective function values. We also stop AMA if the maximum of 10^5 iterations is reached.

When we generate the clustering path for the first parameter value of γ, we first run the IADMM introduced in Algorithm 3 for 100 iterations to generate an initial point, then we use SSNAL to solve (3.1). After that, we use the previously computed optimal solution for the lastest γ as the initial point to warm-start SSNAL for solving the problem corresponding to the next γ. The same strategy is used in CVXCLUSTR.

3.5.1 Numerical verification of Theorem 3.2

In this section, we demonstrate that the theoretical results we obtained in Theorem 3.2 are practically meaningful by conducting numerical experiments on a simulated dataset with five clusters. We generate the five clusters randomly via a 2D Gaussian kernel. Each of the cluster has 100 data points, as shown in Figure 3.1.

Since we know the cluster assignment for each data point, we can construct the corresponding centroid problem given in (3.12). Then, we can solve the weighted convex clustering model (3.1) and the corresponding centroid problem (3.12) separately to compare the results. In our experiments, we choose the weight w_{ij} as

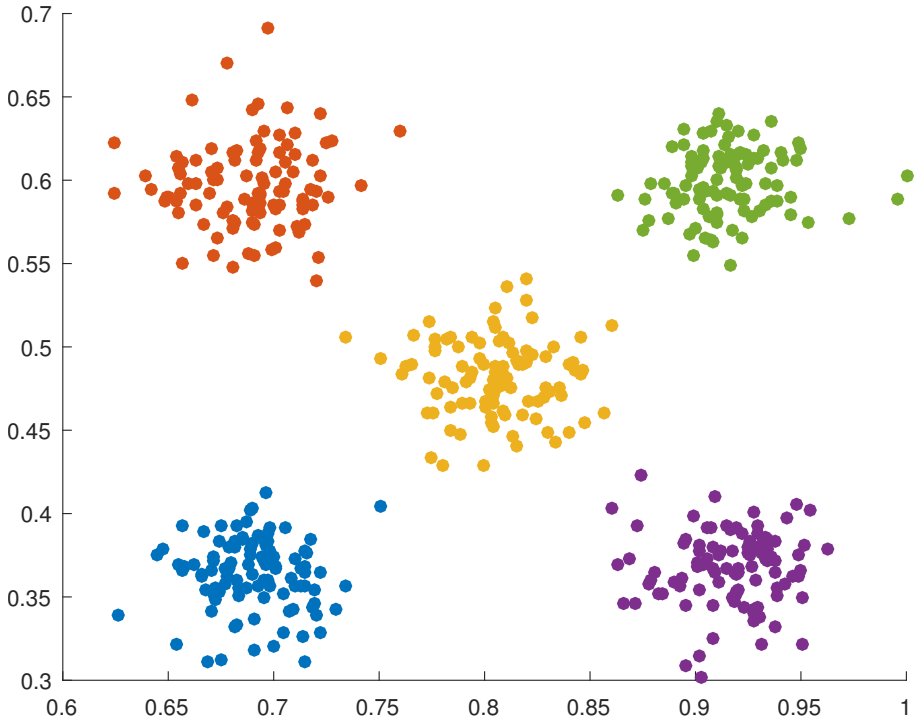


Figure 3.1: Visualization of the generated data.

follows

$$w_{ij} = \begin{cases} \exp(-0.5\|\mathbf{a}_i - \mathbf{a}_j\|^2) & \text{if } (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathcal{E} = \cup_{i=1}^{n}\{(i,j) \mid j \text{ is among } i\text{'s 30-nearest neighbors}, i < j \leq n\} \cup_{\alpha=1}^{5} \{(i,j) \mid i,j \in I_\alpha, i < j\}$.

First, we solve (3.1) and (3.12) separately to find their optimal solutions, denoted as $X^* = [\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_n^*]$ and $\bar{X} = [\bar{\mathbf{x}}^{(1)}, \bar{\mathbf{x}}^{(2)}, \ldots, \bar{\mathbf{x}}^{(K)}]$, respectively. Then, we can construct the new solution $\hat{X}$ for (3.1) based on $\bar{X}$ as

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}}^{(\alpha)} \ \ \forall\, i \in I_\alpha, \quad \alpha = 1, \ldots, 5.$$

We also compute the theoretical lower bound $\gamma_{\min}$ and upper bound $\gamma_{\max}$ based on the formula given in Theorem 3.2, and they are given by

$$\gamma_{\min} = 1.56 \times 10^{-3}, \quad \gamma_{\max} = 0.485$$

in this example.

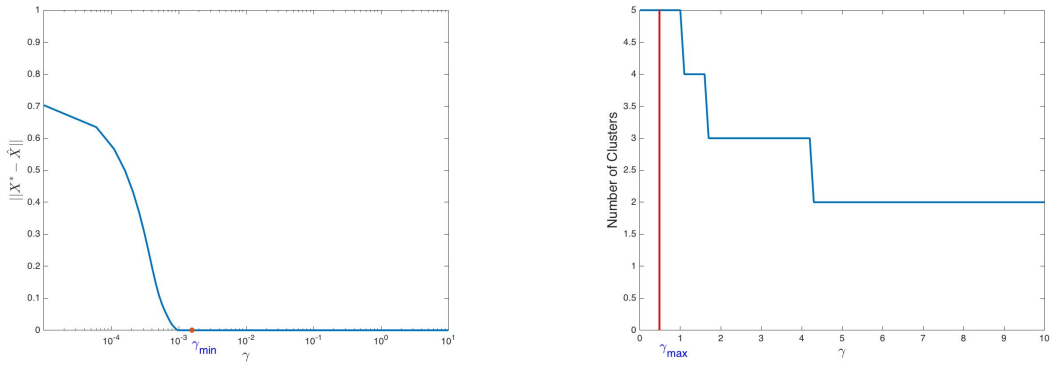


Figure 3.2: Left panel: $\|X^* - \hat{X}\|$ vs γ; Right panel: number of clusters vs γ.

Based on the computed results shown in the left panel of Figure 3.2, we can observe the phenomenon that for very small γ, X^* and $\hat{X}$ are different. However, when γ becomes larger, X^* and $\hat{X}$ coincide with each other in that $\|X^* - \hat{X}\|$ is almost 0 (up to the accuracy level we solve the problems (3.1) and (3.12)). In fact, we see that for γ larger than the theoretical lower bound $\gamma_{\min}$ but less than $\gamma_{\max}$, we have perfect recovery of the clusters by solving (3.1), and when γ is slightly smaller than $\gamma_{\min}$, we lose the perfect recovery property.

Furthermore, from our results in Theorem 3.2, we know that when γ is smaller than $\gamma_{\max}$ but larger than $\gamma_{\min}$, we should recover the correct number of clusters. This is indeed observed in the result shown in the right panel of Figure 3.2 where we track the number of clusters for different values of γ. Moreover, when γ is about two times larger than $\gamma_{\max}$, we get a coarsening of the clusters. The results shown above demonstrate that the theoretical results we have established in Theorem 3.2 are meaningful in practice.

Remark 3.4. *Actually, if we consider the convex clustering model with equal weights for the above example, then, based on the results in part (i) of Corollary 3.1, we can get the following lower and upper bound for γ:*

$$\hat{\gamma}_{min} = 1.53 \times 10^{-3}, \quad \hat{\gamma}_{\max} = 2.00 \times 10^{-4},$$

which is actually not feasible. This example also demonstrates the importance of the weighted convex clustering model and the new theoretical bounds obtained in Theorem

3.2.

Next, we show the numerical performance of our proposed optimization algorithm for solving (3.1) via (3.16).

3.5.2 Simulated data

In this section, we show the performance of our algorithm SSNAL on three simulated datasets: Two Half-Moon, Unbalanced Gaussian [59] and semi-spherical shells data. We compare our SSNAL with the AMA in [16] on different problem scales. The numerical results in Table 3.2 show the superior performance of SSNAL. We also visualize some selected recovery results for Two Half-moon and Unbalanced Gaussian in Figure 3.3.

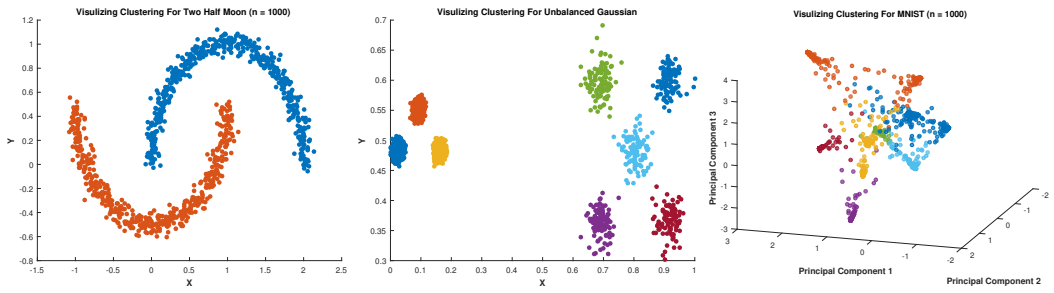


Figure 3.3: Selected recovery results by model (3.1) with 2-norm. Left: Two Half-Moon data with $n = 1000$, $k = 20$, $\gamma = 5$. Middle: Unbalanced Gaussian data with $n = 6500$, $k = 10$, $\gamma = 1$. Right: a subset of MNIST with $n = 1000$, $\gamma = 1$.

Two half-moon data

The simulated data of two interlocking half-moons in $\mathbb{R}^2$ is one of the most popular test examples in clustering. Here we compare the computational time between our proposed SSNAL and AMA on this dataset with different problem scales. We note that AMA could not satisfy the stopping criteria (3.26) within 100000 iterations when n is large. In the experiments, we choose $k = 10$, $\phi = 0.5$ (for the weights

w_{ij}) and $\gamma \in [0.2 : 0.2 : 10]$ (in MATLAB notation) to generate the clustering path. After generating the clustering path with SSNAL, we repeat the experiments using the same pre-stored primal objective values and stop the AMA using the criterion (3.26). We report the average time for solving each problem (50 in total) in Table 3.2. Observe that our SSNAL can be more than 50 times faster than AMA.

Table 3.2: Computation time (in seconds) comparison on the Two Half-Moon data. (— means that the maximum number of 100,000 iterations is reached)

n	200	500	1000	2000	5000	10000
AMA	0.41	4.43	28.27	78.36	—	—
SSNAL	**0.11**	**0.19**	**0.49**	**0.91**	**3.82**	**9.15**

We also compare the recovery performance between the convex clustering model (3.1) and K-means (3.3). We choose the Rand Index [32] as the metric to evaluate the performance of these two clustering algorithms. In Figure 3.4, we can see that comparing to the K-means model, the convex clustering model is able to achieve a much better Rand Index, even when the number of clusters is not correctly identified.

Unbalanced Gaussian and semi-spherical shells data

Next, we show the performance of SSNAL and AMA on the Unbalanced Gaussian data points in $\mathbb{R}^2$ [59]. In this experiment, we solve (3.1) with $k = 10$, $\phi = 0.5$ and $\gamma \in [0.2 : 0.2 : 2]$. For this dataset, we have scaled it so that each entry is in the interval $[0, 1]$. We can see from Figure 3.3 that the convex clustering model (3.1) can recover the cluster assignments perfectly with well chosen parameters.

In the experiments, we find that AMA has difficulties in reaching the stopping criterion (3.26). We summarize some selected results in Table 3.3, wherein we report the computation times and iteration counts for both AMA and SSNCG. Note that we report the number of SSNCG iterations because each of these iterations constitute the main cost for SSNAL. In Figure 3.5, we show the recovery performance between the convex clustering model and K-means on this dataset.

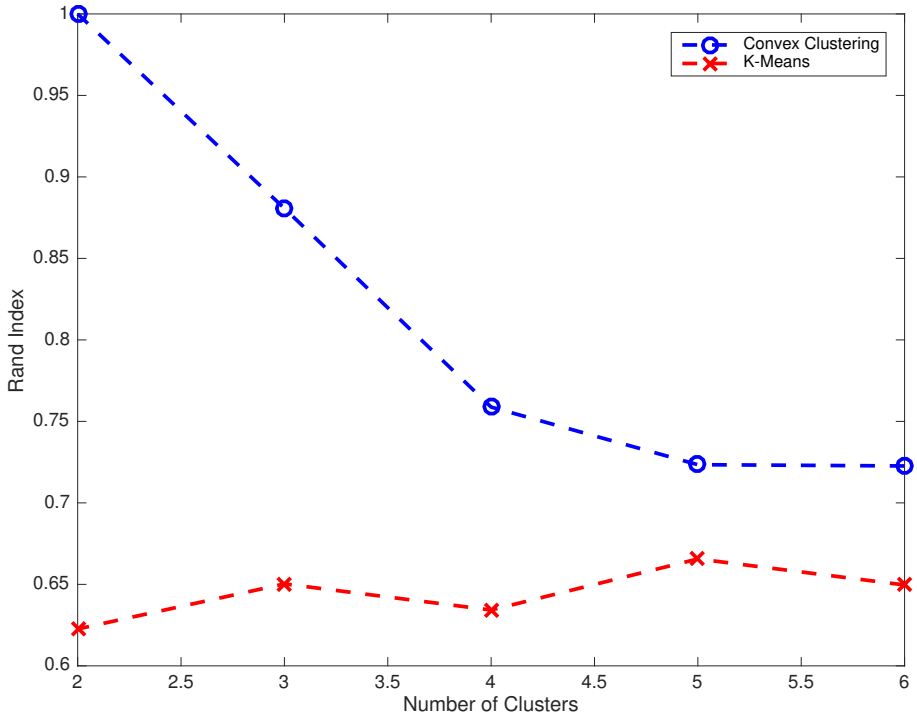


Figure 3.4: Clustering performance (in terms of the Rand Index) of the convex clustering and K-means models on the Two Half Moon dataset.

Table 3.3: Numerical results on Unbalanced Gaussian data.

γ	0.2	0.4	0.6	0.8	1.0
t_{AMA}	264.54	256.21	260.06	262.16	263.27
t_{SSNAL}	**1.15**	**0.57**	**0.65**	**0.64**	**0.83**
Iter_{AMA}	100000	97560	97333	100000	100000
$\text{Iter}_{\text{SSNCG}}$	**23**	**21**	**24**	**24**	**27**

In order to test the performance of our SSNAL on large data set, we also generate a data set with $\mathbf{200,000}$ points in $\mathbb{R}^3$ such that 50% of the points are uniformly distributed in a semi-spherical shell whose inner and outer surfaces have radii equal to 1.0 and 1.4, respectively. The other 50% of the points are uniformly distributed in a concentric semi-spherical shell whose inner and outer surfaces have radii equal to 1.6 and 2.0, respectively. Figure 3.6 depicts the recovery result when we use only

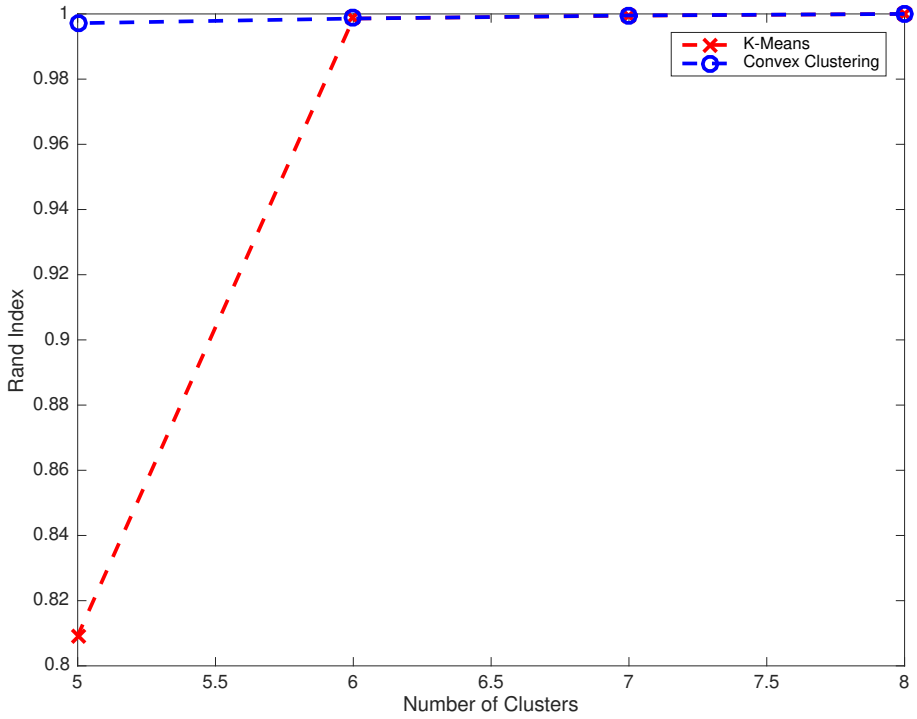


Figure 3.5: Clustering performance (in terms of the Rand Index) of the convex clustering and K-means models on the Unbalanced Gaussian dataset.

6,000 points. For the data set with $\mathbf{n = 200,000}$, our algorithm takes only **374 seconds** to solve the model (3.1) when we choose $\gamma = 50$, $\phi = 0.5$ and $k = 10$. In solving the problem, our algorithm used 32 SSNCG iterations and the average number of CG steps needed to solve the large linear system (3.24) is 79.3 only. Thus, we can see that our algorithm can be very efficient in solving the convex clustering model (3.1) even when the data set is large. Note that we did not run AMA as it will take too much time to solve the problem.

3.5.3 Real data

In this section, we compare the performance of our proposed SSNAL with AMA on some real datasets, namely, MNIST, Fisher Iris, WINE, Yale Face B(10Train subset). For real datasets, a preprocessing step is sometimes necessary to transform the data to one whose features are meaningful for clustering. Thus, for a subset of

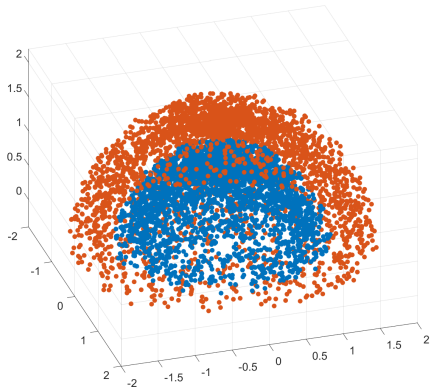


Figure 3.6: Recovery result by model (3.1) for a semi-spherical shells data set with $6,000$ points.

MNIST (we selected a subset because AMA cannot handle the whole dataset), we first apply the preprocessing method described in [48]. Then we apply the model (3.1) on the preprocessed data. The comparison results between SSNAL and AMA on the real datasets are presented in Table 3.4. One can observe that SSNAL can be much more efficient than AMA.

Table 3.4: Computation time comparison on real data. (*) means that the maximum of 100000 iterations is reached for all instances.

Dataset	d	n	AMA(s)	SSNAL(s)
MNIST	10	1,000	79.48	**1.47**
MNIST	10	10,000	1753.8*	**69.3**
Fisher Iris	4	150	0.58	**0.16**
WINE	13	178	2.62	**0.19**
Yale Face B	1024	760	211.36	**35.13**

3.5.4 Sensitivity with different γ

In order to generate a clustering path for a given dataset, we need to solve (3.1) for a sequence of $\gamma > 0$. So the stability of the performance of the optimization

algorithm with different γ is very important. In our experiments, we have found that the performance of AMA is rather sensitive to the value of γ in that the time taken to solve problems with different values of γ can vary widely. However, SSNAL is much more stable. In Figure 3.7, we show the comparison between SSNAL and AMA on both the Two Half-Moon and MNIST datasets with $\gamma \in [0.2 : 0.2 : 10]$.

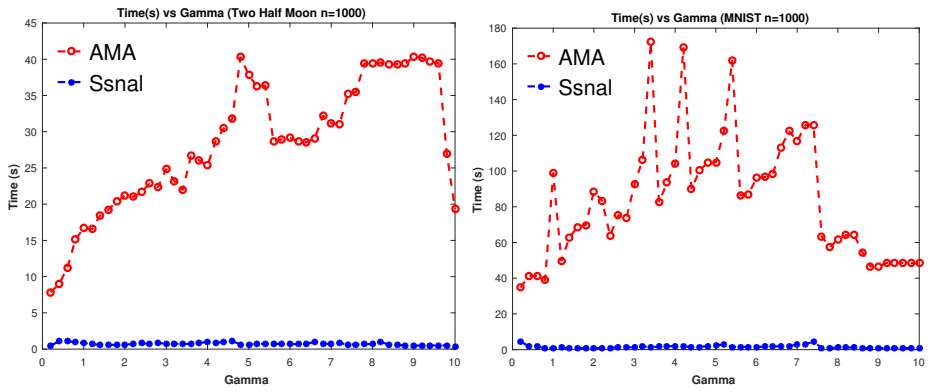


Figure 3.7: Time comparison between SSNAL and AMA on both Two Half-Moon and MNIST data with $\gamma \in [0.2 : 0.2 : 10]$.

3.5.5 Scalability of our proposed algorithm

In this section, we demonstrate the scalability of our algorithm SSNAL. Before we show the numerical results, we give some insights as to why our algorithm could be scalable. Recall that the most computationally expensive step in our framework is in using the semismooth Newton-CG method to solve (3.22). However, if we look inside the algorithm, we can see that the key step is to use the CG method to solve (3.24) efficiently to get the Newton direction. According to our complexity analysis in Section 3.4.4, the computational cost for one step of the CG method is $O(d|\mathcal{E}| + d|\widehat{\mathcal{E}}|)$. By the specific choice of $\mathcal{E}$, $|\mathcal{E}|$ and $|\widehat{\mathcal{E}}|$ should only grow slowly with n. The low computational cost for the matrix-vector product in our CG method, the rapid convergence of the CG method, and the fast convergence of the SSNCG are the key reasons behind why our algorithm can be scalable and efficient.

In our experiments, we set $\phi = 0.5$, $k = 10$ (the number of nearest neighbors). Then we solve (3.1) with $\gamma \in [0.4 : 0.4 : 20]$. After generating the clustering path, we compute the average time for solving a single instance of (3.1) for each problem scale. Another factor related to the scalability is the number of neighbors k used to generate $\mathcal{E}$ in (3.1). So, we also show the performance of SSNAL with different values of k. For each $k \in [5 : 5 : 50]$, we generate the clustering path for the Two Half-Moon data with $n = 2000$. Then we report the average time for solving a single instance of (3.1) for each k. We summarize our numerical results in Figure 3.8. We can observe that the computation time grows almost linearly with n and k.

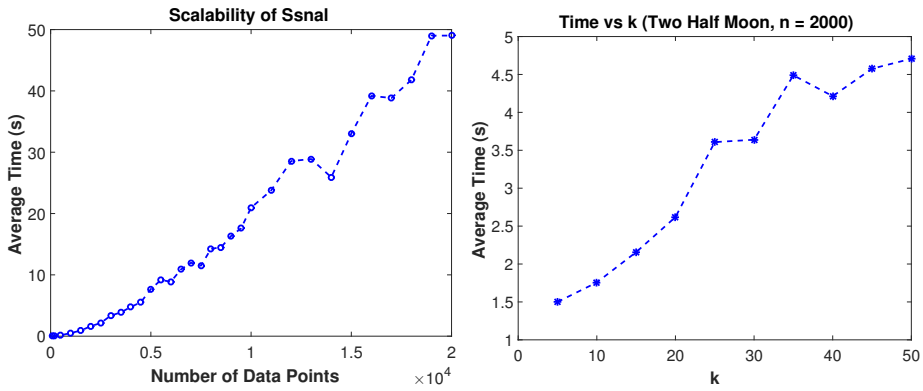


Figure 3.8: Numerical results to demonstrate the scalability of our proposed algorithm SSNAL with respect to n and k.

Comparing to the numerical results reported in [16] and [54] with $n \leq 500$ and $n \leq 600$, respectively, in our experiments, we apply our algorithm on the Half-Moon data with n ranging from 100 to 20000. Together with the semi-spherical shells with $\mathbf{200,000}$ data points, our results have convincingly demonstrated the scalability of SSNAL.

3.6 Summary of the chapter

In this chapter, we mainly focus on the convex clustering model. We first establish the theoretical perfect recovery guarantee for the general weighted convex

clustering model (3.1) under some mild sufficient conditions. Then, we propose a semismooth Newton CG based augmented Lagrangian method (SSNAL) to solve the convex clustering model efficiently. We numerically validate the feasibility of our proposed sufficient condition for the recovery properties of the weighted convex clustering model via a randomly generated example from a mixture of Gaussian distributions. Also, we demonstrate the efficiency and robustness of the algorithm SSNAL for solving the general convex clustering model.

Chapter 4

Intra-group Level Feature Selection

In this chapter, we focus on the topic of the intra-group level feature selection. As we discussed in section 1.1.2, the exclusive lasso regularizer can enforce this kind of desired structured sparsity. Consider a given positive weight vector $w \in \Re^n_{++}$, and a partition of variable index groups $\mathcal{G} := \{g | g \subseteq \{1, 2, \ldots, n\}\}$ such that $\bigcup_{g \in \mathcal{G}} g = \{1, 2, \ldots, n\}$ and $g_i \bigcap g_j = \emptyset$ for any $g_i, g_j \in \mathcal{G}$. For $x \in \Re^n$, the weighted exclusive lasso regularizer is defined as

$$\Omega^{\mathcal{G},w}(x) := \sum_{g \in \mathcal{G}} \|w_g \circ x_g\|_1^2, \tag{4.1}$$

where "$\circ$" denotes the Hadamard product, and x_g denotes the sub-vector of x with those elements not in g removed from x.

In order to demonstrate the power of the exclusive lasso regularizer in intra-group level feature selection, in this chapter, we consider the machine learning model with the exclusive lasso regularizer.

For given data $A \in \Re^{m \times n}$, $y \in \Re^m$, a partition of variable index groups $\mathcal{G} := \{g | g \subseteq \{1, 2, \ldots, n\}\}$ and weights $\{w_g | g \in \mathcal{G}\}$, we consider the following machine learning model:

$$\min_x l(A, y; x) + \lambda \sum_{g \in \mathcal{G}} \|w_g \circ x_g\|_1^2, \tag{4.2}$$

where $l(A, y; x)$ is the loss function.

In this chapter, we will mainly focus on the following points.

(1) Inspired by [38], we propose a dual Newton based preconditioned proximal point algorithm (PPDNA) to solve the machine learning models with the exclusive lasso regularizer.

(2) We provide a rigorous proof for the closed-form solution to the proximal mapping of $\|w \circ \cdot\|_1^2$ and derive the corresponding generalized Jacobian. These results are critical for the computational efficiency of various algorithmic frameworks for solving (4.2).

(3) We demonstrate numerically that PPDNA is highly efficient and robust comparing to ILSA, APG and ADMM, even with the availability of the closed-form proximal mapping of the exclusive lasso regularizer. Furthermore, we apply the exclusive lasso model in index ETF and achieve better out-of-sample results, comparing to the lasso and group lasso model.

The remaining parts of the paper are organized as follows. In section 4.1, we propose the preconditioned proximal point algorithm (preconditioned PPA) for solving a general 2-block convex composite programming problem. And the dual Newton algorithm (DNA) for solving the PPA subproblem is introduced in section 4.2. In section 4.3, we provide a rigorous proof for the closed-form solution to $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$, followed by the derivation of the corresponding HS-Jacobian. In section 4.4, we present our numerical results when solving regression problems and classification problems, on both synthetic data and real applications. In the end, we conclude the chapter and discuss some possible future work.

Notations and preliminaries: For any $z \in \Re$, sign(z) is defined to be 1 if $z \geq 0$, -1 otherwise. For $x \in \Re^n$, denote $x^+ := \max\{x, 0\}$ and $x^- := -\min\{x, 0\}$. Denote I_n as the identity matrix in $\Re^{n\times n}$. We use "diag(X)" to denote the vector consisting of the diagonal entries of the matrix X and "Diag(x)" to denote the diagonal matrix whose diagonal is given by the vector x. For given matrix C, we also use $C^\dagger$ to represent its Moore-Penrose inverse. For any self-adjoint semidefinite linear operator $\mathcal{M} : \Re^n \to \Re^n$, we define $\langle x, x'\rangle_{\mathcal{M}} := \langle x, \mathcal{M}x'\rangle$, and $\|x\|_{\mathcal{M}} := \sqrt{\langle x, x\rangle_{\mathcal{M}}}$ for all

$x, x' \in \Re^n$. Let $\mathcal{C}$ be a given subset of $\Re^n$, denote the weighted distance of $x \in \Re^n$ to $\mathcal{C}$ as $\mathrm{dist}_{\mathcal{M}}(x, \mathcal{C}) := \inf_{x' \in \mathcal{C}} \|x - x'\|_{\mathcal{M}}$. The largest eigenvalue of $\mathcal{M}$ is denoted as $\lambda_{\max}(\mathcal{M})$.

For any closed proper convex funtion $p : \Re^n \to (-\infty, \infty]$, the conjugate function is defined as $p^*(z) := \sup_{x \in \Re^n}\{\langle x, z\rangle - p(x)\}$. The Moreau envelope of p at x is defined by

$$\mathrm{E}_p(x) := \min_{y \in \Re^n}\{p(y) + \frac{1}{2}\|y - x\|^2\},$$

and the associated proximal mapping is

$$\mathrm{Prox}_p(x) := \arg\min_{y \in \Re^n}\{p(y) + \frac{1}{2}\|y - x\|^2\}.$$

It is known that $\nabla \mathrm{E}_p(x) = x - \mathrm{Prox}_p(x)$ and $\mathrm{Prox}_p(x)$ is Lipschitz continuous with modulus 1 [49, 53, 63].

4.1 A preconditioned proximal point algorithm for solving the exclusive lasso problem

The exclusive lasso problem is a special case of the following general 2-block convex composite programming problem, which is given as

$$\min_x\{f(x) := h(\mathcal{A}x) - \langle c, x\rangle + p(x)\}, \tag{4.3}$$

where $\mathcal{A} : \Re^n \to \Re^m$ is a linear mapping, $c \in \Re^n$, $h : \Re^m \to \Re$ is a smooth convex function, and $p : \Re^n \to (-\infty, +\infty]$ is a closed, proper, convex function. In particular, if $p(\cdot) = \lambda\Omega^{\mathcal{G},w}(\cdot)$, where $\Omega^{\mathcal{G},w}(\cdot)$ is the exclusive lasso regularizer defined in (4.1) and $\lambda > 0$ is a parameter, then (4.3) is reduced to the so-called exclusive lasso model.

Define the proximal residual function $R : \Re^n \to \Re^n$ by

$$R(x) = x - \mathrm{Prox}_p(x - \mathcal{A}^*\nabla h(\mathcal{A}x) + c), \quad \forall x \in \Re^n, \tag{4.4}$$

and the set-valued map $\mathcal{T}_f(x) := \partial f(x)$. Assume that the solution set Ω to (4.3) is nonempty. The first order optimality condition of (4.3) implies that $\bar{x} \in \Omega$ if and only if $R(\bar{x}) = 0$.

The proximal point algorithm (PPA) [62, 63] is a well established algorithmic framework for solving the minimization problems. Under mild conditions, the PPA is proved to have an asymptotic superlinear convergence rate. Recently, Li et.al extend the classical PPA to the preconditioned proximal point algorithm (preconditioned PPA) in [38]. In this section, we apply the preconditioned PPA to solve the general 2-block convex composite programming problem.

4.1.1 Preconditioned PPA for 2-block convex composite programming problem

For any starting point $x^0 \in \Re^n$, the preconditioned PPA generates a sequence $\{x^k\} \subseteq \Re^n$ by the following approximate rule:

$$\begin{aligned} x^{k+1} \approx \mathcal{P}_k(x^k) &:= \arg\min_{x\in\Re^n}\{f(x) + \frac{1}{2\sigma_k}\|x - x^k\|^2_{\mathcal{M}_k}\} \\ &= \arg\min_{x\in\Re^n}\{h(\mathcal{A}x) - \langle c, x\rangle + p(x) + \frac{1}{2\sigma_k}\|x - x^k\|^2_{\mathcal{M}_k}\}, \end{aligned} \tag{4.5}$$

where $\{\sigma_k\}$ is a sequence of nondecreasing positive real numbers ($\sigma_k \uparrow \sigma_\infty \leq \infty$) and $\{\mathcal{M}_k\}$ is a sequence of self-adjoint positive definite linear operators satisfying the following conditions:

$$\mathcal{M}_k \succeq \mathcal{M}_{k+1}, \quad \mathcal{M}_k \succeq \lambda_{\min} I_n, \quad \forall k \geq 0,$$

with some constant $\lambda_{\min} > 0$. When $\mathcal{M}_k \equiv I_n$ for all $k \geq 0$, the preconditioned PPA reduces to the classical PPA.

To ensure the convergence of the preconditioned PPA, we need the following

stopping criteria as proposed in [38]:

$$\|x^{k+1} - \mathcal{P}_k(x^k)\|_{\mathcal{M}_k} \leq \epsilon_k, \quad \epsilon_k \geq 0, \quad \sum_{k=0}^{\infty} \epsilon_k < \infty, \tag{A}$$

$$\|x^{k+1} - \mathcal{P}_k(x^k)\|_{\mathcal{M}_k} \leq \delta_k \|x^{k+1} - x^k\|_{\mathcal{M}_k}, \quad 0 \leq \delta_k < 1, \quad \sum_{k=0}^{\infty} \delta_k < \infty. \tag{B}$$

4.1.2 Convergence of the preconditioned PPA

We adopt the convergence results of the preconditioned PPA here for the convenience of the readers, which can be found in [38].

Theorem 4.1. *(1) Let $\{x^k\}$ be the sequence generated by the preconditioned PPA (4.5) with the stopping criterion (A). Then $\{x^k\}$ is bounded and*

$$\mathrm{dist}_{\mathcal{M}_{k+1}}(x^{k+1}, \Omega) \leq \mathrm{dist}_{\mathcal{M}_k}(x^k, \Omega) + \epsilon_k, \quad \forall k \geq 0.$$

In addition, $\{x^k\}$ converges to some $x^* \in \Omega$.

(2) Let $r := \sum_{i=0}^{\infty} \epsilon_k + \mathrm{dist}_{\mathcal{M}_0}(x^0, \Omega)$. Assume that for this $r > 0$, there exists a $\kappa > 0$ such that the monotone multifunction $\mathcal{T}_f(x)$ associated with (4.3) satisfies the following error bound assumption

$$\mathrm{dist}(x, \Omega) \leq \kappa \mathrm{dist}(0, \mathcal{T}_f(x)), \quad \forall x \in \Re^n \text{ satisfying } \mathrm{dist}(x, \Omega) \leq r. \tag{4.6}$$

Suppose that $\{x^k\}$ is generated by the preconditioned PPA with the stopping criteria (A) and (B). Then it holds for all $k \geq 0$ that

$$\mathrm{dist}_{\mathcal{M}_{k+1}}(x^{k+1}, \Omega) \leq \mu_k \mathrm{dist}_{\mathcal{M}_k}(x^k, \Omega), \tag{4.7}$$

where

$$\mu_k = \frac{1}{1-\delta_k} \frac{\delta_k + (1+\delta_k)\kappa\lambda_{\max}(\mathcal{M}_k)}{\sqrt{\sigma_k^2 + \kappa^2 \lambda_{\max}^2(\mathcal{M}_k)}} \to \mu_\infty = \frac{\kappa\lambda_\infty}{\sqrt{\kappa^2\lambda_\infty^2 + \sigma_\infty^2}} < 1, \quad k \to \infty,$$

where $\lambda_\infty = \lim_{k\to\infty} \lambda_{\max}(\mathcal{M}_k)$. In addition, it holds that for all $k \geq 0$,

$$\mathrm{dist}(x^{k+1}, \Omega) \leq \frac{\mu_k}{\sqrt{\lambda_{\min}(\mathcal{M}_{k+1})}} \mathrm{dist}_{\mathcal{M}_k}(x^k, \Omega).$$

The following proposition, which is an application of [88, Theorem 2], can be used to establish error bound conditions for many commonly used loss function plus piecewise linear-quadratic regularizer.

Proposition 9. *Assume that Ω is non-empty and compact. Suppose that (1) h is continuously differentiable on $\Re^m$ and strongly convex on any compact convex set in $\Re^m$; (2) $p(\cdot)$ is a piecewise linear-quadratic convex function. Then for any $\xi \geq \inf_{x\in\Re^n} f(x)$, there exist constants $\kappa, \varepsilon > 0$ such that*

$$\operatorname{dist}(x, \Omega) \leq \kappa \|R(x)\| \quad \text{for all } x \in \Re^n \text{ with } f(x) \leq \xi,\ \|R(x)\| \leq \varepsilon.$$

Proof. From [88, Proposition 1], we know that that there exists a $\bar{y} \in \Re^m$ such that

$$\mathcal{A}x = \bar{y}, \quad \mathcal{A}^*\nabla h(\mathcal{A}x) - c = \bar{g}, \quad \forall x \in \Omega,$$

where $\bar{g} = \mathcal{A}^*\nabla h(\bar{y}) - c$. Consider the collection $\mathcal{C} := \{\Gamma_h(\bar{y}), \Gamma_p(\bar{g})\}$, where

$$\Gamma_h(y) := \{x \in \Re^n \mid \mathcal{A}x = y\}, \quad \Gamma_p(g) := \{x \in \Re^n \mid -g \in \partial p(x)\}.$$

Since $\Gamma_h(\bar{y})$ is the set of solutions to a linear system, it is a polyhedral closed convex set. According to [61, Corollary 23.5.1],

$$\Gamma_p(\bar{g}) := \{x \in \Re^n \mid -\bar{g} \in \partial p(x)\} = \{x \in \Re^n \mid x \in \partial p^*(-\bar{g})\} = \partial p^*(-\bar{g}).$$

Since p is piecewise linear-quadratic, p^* is also piecewise linear-quadratic by [64, Theorem 11.14(b)]. Then ∂p and ∂p^* are both polyhedral due to [64, Proposition 10.21]. Therefore, $\Gamma_h(\bar{y})$ and $\Gamma_p(\bar{g})$ are closed convex polyhedral sets. Due to [6, Corollary 3], we can see that $\mathcal{C}$ is boundedly linearly regular. Since ∂p^* is a polyhedral multifunction, we can see from [22, Proposition 3H.1] that ∂p^* is calm at $-\bar{g}$ for any $\bar{x} \in \Omega$, thus $\partial p = (\partial p^*)^{-1}$ is metrically subregular at $\bar{x}$ for $-\bar{g}$ [22, Theorem 3H.3]. Therefore, by [88, Theorem 2], the solution map $\Gamma(y, g) := \{x \in \Re^n \mid \mathcal{A}x = y, -g \in \partial p(x)\}$ is calm at $(\bar{y}, \bar{g})$ for any $\bar{x} \in \Omega$. Then the desired conclusion holds by [88, Corollary 1]. □

For the exclusive lasso regularized models, we can see from the next proposition that the error bound assumption (4.6) holds for the linear regression problem and the logistic regression problem, which means the preconditioned PPA can be expected to have fast linear convergence.

Proposition 10. *Assume that in* (4.3), $p(\cdot) = \lambda\Omega^{\mathcal{G},w}(\cdot)$. *Then the error bound assumption* (4.6) *holds in the following two cases:*

(1) $h(y) = \sum_{i=1}^{m}(y_i - b_i)^2/2$, *for some given vector* $b \in \Re^m$;

(2) $h(y) = \sum_{i=1}^{m}\log(1+\exp(-b_i y_i))$, *for some given vector* $b \in \Re^m$ *and* $b_i = 1$ *or* -1.

Proof. (1) When $h(y) = \sum_{i=1}^{m}(y_i - b_i)^2/2$, $f(\cdot)$ is a piecewise linear-quadratic convex function, from [66], $\mathcal{T}_f(\cdot)$ is piecewise polyhedral, thus it satisfies the error bound assumption (4.6) [38, 60].

(2) When $h(y) = \sum_{i=1}^{m}\log(1+\exp(-b_i y_i))$, since $f(\cdot)$ is nonnegative, and $f(x) \to +\infty$ as $\|x\| \to +\infty$, Ω is non-empty and compact. Given $r > 0$, define $\Omega_r := \{x \in \Re^n \mid \mathrm{dist}(x,\Omega) \le r\}$. Due to the fact that Ω is compact, Ω_r is compact and thus $\xi := \max_{x\in\Omega_r} f(x)$ is finite. From Proposition 9, we know that for this ξ, there exist constants $\kappa, \varepsilon > 0$ such that

$$\mathrm{dist}(x,\Omega) \le \kappa\|R(x)\| \quad \text{for all } x \in \Re^n \text{ with } f(x) \le \xi,\ \|R(x)\| \le \varepsilon, \tag{4.8}$$

where $R(x)$ is defined as in (4.4). We consider two cases:
Case 1: $x \in \Omega_r$ and $\|R(x)\| \le \varepsilon$. From (4.8), we have $\mathrm{dist}(x,\Omega) \le \kappa\|R(x)\|$;
Case 2: $x \in \Omega_r$ and $\|R(x)\| > \varepsilon$. Then $\mathrm{dist}(x,\Omega) \le (r/\varepsilon)\varepsilon \le (r/\varepsilon)\|R(x)\|$.

Therefore, it holds that

$$\mathrm{dist}(x,\Omega) \le \max\{\kappa, (r/\varepsilon)\}\|R(x)\|, \quad \forall x \in \Omega_r = \{x \in \Re^n \mid \mathrm{dist}(x,\Omega) \le r\}.$$

We follow the ideas in [21, Theorem 3.1] and [20, Proposition 2.4]. Let $y \in \mathcal{T}_f(x)$, which means $y \in \mathcal{A}^*\nabla h(\mathcal{A}x) - c + \partial p(x)$, we have that $x = \mathrm{Prox}_p(x + y - \mathcal{A}^*\nabla h(\mathcal{A}x) +$

c). Thus

$$\|R(x)\| = \|\mathrm{Prox}_p(x - \mathcal{A}^*\nabla h(\mathcal{A}x) + c) - \mathrm{Prox}_p(x + y - \mathcal{A}^*\nabla h(\mathcal{A}x) + c)\| \le \|y\|.$$

As a result,

$$\mathrm{dist}(x, \Omega) \le \max\{\kappa, (r/\varepsilon)\}\mathrm{dist}(0, \mathcal{T}_f(x)), \quad \forall x \in \Omega_r = \{x \in \Re^n \mid \mathrm{dist}(x, \Omega) \le r\}.$$

□

Note that the key challenge in executing the preconditioned PPA is whether the nonsmooth problem (4.5) can be solved efficiently. We consider two special cases: one is $\mathcal{M}_k \equiv I_n$ for all $k \ge 0$, the other is $\mathcal{M}_k \equiv I_n + \tau\mathcal{A}^*\mathcal{A}$, where $\tau > 0$ is a given positive number. In order to get a superlinear (or even quadratic) convergence rate in the subroutines, we design a dual Newton algorithm (DNA) to solve (4.5) for these two cases.

4.2 A dual Newton algorithm for solving the subproblem

For all $k \ge 0$, we aim to solve the preconditioned PPA subproblem

$$\min_{x \in \Re^n}\{f_k(x) := h(\mathcal{A}x) - \langle c, x\rangle + p(x) + \frac{1}{2\sigma_k}\|x - x^k\|^2_{\mathcal{M}_k}\}. \tag{4.9}$$

Obviously, $f_k(\cdot)$ is a strongly convex function, albeit nonsmooth or non-Lipschitian. Thus the above minimization problem admits a unique solution $\bar{x}^{k+1}$. The main point is how one can solve (4.9) in a fast and robust way. Our choice is the dual Newton algorithm (DNA).

4.2.1 The case when $\mathcal{M}_k \equiv I_n$

In this case, one can write (4.9) equivalently as

$$\min_{x,z \in \Re^n, y \in \Re^m}\{h(y) - \langle c, x\rangle + p(x) + \frac{1}{2\sigma_k}\|x - x^k\|^2 \mid \mathcal{A}x - y = 0\}. \tag{4.10}$$

The dual of the above problem, after ignoring the constant term, is

$$\max_{u\in\Re^n}\{\phi_k(u) := -h^*(u) - \frac{1}{2\sigma_k}\|x^k + \sigma_k(c - \mathcal{A}^*u)\|^2 + \frac{1}{\sigma_k}\mathrm{E}_{\sigma_k p}(x^k + \sigma_k(c - \mathcal{A}^*u))\}. \tag{4.11}$$

Suppose that the following assumption holds for h^*.

Assumption 4.1. *$h^*(\cdot)$ is twice continuously differentiable and strongly convex with modulus α_h in* $\mathrm{int}(\mathrm{dom}(h^*))$.

It needs to be mentioned that when we consider the least squares loss function $h(y) = \sum_{i=1}^m (y_i - b_i)^2/2$, Assumption 4.1 holds with $\alpha_h = 1$. Under Assumption 4.1, we can see that $\phi_k(\cdot)$ is strongly concave, thus (4.11) has a unique optimal solution $\bar{u}^{k+1}$, and $\bar{x}^{k+1}$ can be obtained by

$$\bar{x}^{k+1} = \mathrm{Prox}_{\sigma_k p}(x^k + \sigma_k(c - \mathcal{A}^*\bar{u}^{k+1})).$$

Since $\phi_k(\cdot)$ is continuously differentiable, $\bar{u}^{k+1}$ can be obtained by solving the following nonlinear and nonsmooth equation

$$\nabla\phi_k(u) = -\nabla h^*(u) + \mathcal{A}\mathrm{Prox}_{\sigma_k p}(x^k + \sigma_k(c - \mathcal{A}^*u)) = 0.$$

Now we can give the full expression of the preconditioned PPA with subproblems solved by the DNA, which is given in Algorithm 9.

As one can see in the algorithm, we need the implementations of the stopping criteria (A) and (B) associated with u^{k+1} and x^{k+1}. By the discussions in [39,45,62], the stopping criteria (A) and (B) can be achieved by the following implementable criteria:

$$\|\nabla\phi_k(u^{k+1})\| \le \sqrt{\alpha_h/\sigma_k}\epsilon_k, \quad \epsilon_k \ge 0, \quad \sum_{k=0}^{\infty}\epsilon_k < \infty, \tag{A'}$$

$$\|\nabla\phi_k(u^{k+1})\| \le \sqrt{\alpha_h/\sigma_k}\delta_k\|x^{k+1} - x^k\|, \quad 0 \le \delta_k < 1, \quad \sum_{k=0}^{\infty}\delta_k < \infty. \tag{B'}$$

Then we need to discuss how to solve (4.12) in Algorithm 9. For a fixed $\sigma > 0$, $\tilde{x} \in \Re^n$, we aim to solve

$$\min_{u\in\Re^n}\{\phi(u) := -h^*(u) - \frac{1}{2\sigma}\|\tilde{x} + \sigma(c - \mathcal{A}^*u)\|^2 + \frac{1}{\sigma}\mathrm{E}_{\sigma p}(\tilde{x} + \sigma(c - \mathcal{A}^*u))\},$$

Algorithm 9: PPDNA for (4.3)

Initialization: Choose $x^0 \in \Re^n$, $\sigma_0 > 0$. For $k = 0, 1, 2, \ldots$. Repeat

Step 1 . Compute

$$u^{k+1} \approx \arg\max\{\phi_k(u)\}, \tag{4.12}$$

$$x^{k+1} = \mathrm{Prox}_{\sigma_k p}(x^k + \sigma_k(c - \mathcal{A}^* u^{k+1})), \tag{4.13}$$

where $\phi_k(\cdot)$ is defined as in (4.11), to satisfy the stopping conditions (A) and (B).

Step 2 . Update $\sigma_{k+1} \uparrow \sigma_\infty \leq \infty$.

Until stopping criterion is satisfied.

which is equivalently to solve the nonsmooth equation

$$\nabla\phi(u) = -\nabla h^*(u) + \mathcal{A}\mathrm{Prox}_{\sigma p}(\tilde{x} + \sigma c - \sigma\mathcal{A}^* u). \tag{4.14}$$

Note that $\nabla\phi(\cdot)$ is Lipschitz continuous, but not differentiable. Due to the quadratic convergence of Newton's method, it is usually the first choice for solving a nonlinear equation if it can be efficiently implemented. However, the direct application of Newton's method to (4.14) is infeasible since the function $\nabla\phi(\cdot)$ is nonsmooth. Fortunately, the semismooth version of the Newton's method has been established in [37, 57]. This allows us to solve (4.14) by a semismooth Newton method (SSN), which has at least the superlinear convergence property.

We now derive the generalized Jacobian of the Lipschitz continuous function $\nabla\phi(\cdot)$. For given u, the following set-valued map is well defined:

$$\hat{\partial}^2\phi(u) := -\nabla^2 h^*(u) - \sigma\mathcal{A}\partial\mathrm{Prox}_{\sigma p}(\tilde{x} + \sigma c - \sigma\mathcal{A}^* u)\mathcal{A}^*,$$

where $\partial\mathrm{Prox}_{\sigma p}(\tilde{x}+\sigma c-\sigma\mathcal{A}^* u)$ is the generalized Jacobian of the Lipschitz continuous mapping $\mathrm{Prox}_{\sigma p}(\cdot)$ at $\tilde{x} + \sigma c - \sigma\mathcal{A}^* u$. Then we can treat $\hat{\partial}^2\phi(u)$ as the surrogate generalized Jacobian of $\nabla\phi(\cdot)$ at point u.

Next we present our semismooth Newton (SSN) method in Algorithm 10 for solving (4.14), which can be expected to get a fast superlinear (or even quadratic) convergence rate.

Algorithm 10: SSN for (4.14)

Initialization: Given $u^0 \in \text{int}(\text{dom}(h^*))$, $\mu \in (0, 1/2)$, $\tau \in (0, 1]$, and $\bar{\gamma}, \delta \in (0, 1)$. For $j = 0, 1, 2, \ldots$. Repeat

Step 1 . Select an element $\mathcal{H}_j \in \hat{\partial}^2 \phi(u^j)$. Apply the conjugate gradient (CG) method to find an approximate solution $d^j \in \Re^m$ to

$$\mathcal{H}_j(d^j) \approx -\nabla\phi(u^j) \tag{4.15}$$

such that $\|\mathcal{H}_j(d^j) + \nabla\phi(u^j)\| \leq \min(\bar{\eta}, \|\nabla\phi(u^j)\|^{1+\tau})$.

Step 2 . Set $\alpha_j = \delta^{m_j}$, where m_j is the first nonnegative integer m for which

$$\phi(u^j + \delta^m d^j) \leq \phi(u^j) + \mu\delta^m \langle \nabla\phi(u^j), d^j \rangle.$$

Step 3 . Set $u^{j+1} = u^j + \alpha_j d^j$.

Until stopping criteria (A') and (B') based on u^{j+1} are satisfied.

The convergence result of the SSN can be found in Theorem 4.2.

Theorem 4.2. *Assume that for any $\sigma > 0$, $\text{Prox}_{\sigma p}(\cdot)$ is strongly semismooth with respect to $\partial\text{Prox}_{\sigma p}(\cdot)$. Let the sequence $\{u^j\}$ be generated by Algorithm 10. Then $\{u^j\}$ converges to the unique solution $\bar{u}$ of the problem in (4.14), and for j sufficiently large,*

$$\|u^{j+1} - \bar{u}\| = O(\|u^j - \bar{u}\|^{1+\tau}),$$

where $\tau \in (0, 1]$ is a given constant in the algorithm, which is typically chosen to be 0.5 in our experiments.

Proof. By [86, Proposition 3.3 and Theorem 3.4], we can see that $\{u^j\}$ converges to the unique solution $\bar{u}$. Then by mimicking the proof of [40, Theorem 3], we can get the convergence rate of $\{u^j\}$. □

Remark 4.1. *Note that all finite-valued convex functions are semismooth [46]. In particular, we consider the case for $p(\cdot) = \lambda\Omega^{\mathcal{G},w}(\cdot)$. It can be proved in the next*

section that $\mathrm{Prox}_{\sigma p}(\cdot)$ *is strongly semismooth with respect to* $\partial_{\mathrm{HS}}\mathrm{Prox}_{\sigma p}(\cdot)$*, where* $\partial_{\mathrm{HS}}\mathrm{Prox}_{\sigma p}(\cdot)$ *is the HS Jacobian of* $\mathrm{Prox}_{\sigma p}(\cdot)$.

We should emphasize that the efficiency in computing the Newton direction in (4.15) depends critically on exploiting the sparse structure of the generalized Hessian of the underlying nonsmooth function, which in turns relies on the structure of the generalized Jacobian of $\mathrm{Prox}_{\sigma p}(\cdot)$. The case of $p(\cdot) = \lambda\Omega^{\mathcal{G},w}(\cdot)$ will be discussed in the next section, where an important property called the second-order sparsity is carefully treated in the implementation.

4.2.2 The case when $\mathcal{M}_k \equiv I_n + \tau\mathcal{A}^*\mathcal{A}$

In some cases, Assumption 4.1 may not hold, e.g. when $h(y) = \|y - b\|_2$ for a given vector $b \in \Re^m$. Then we can choose $\mathcal{M}_k \equiv I_n + \tau\mathcal{A}^*\mathcal{A}$, where τ is a given positive number. The reason why we add the $\mathcal{A}^*\mathcal{A}$ term is to deal with the function h^*.

To be specific, (4.9) can be equivalently written as

$$\min_x\{h(y) - \langle c, x\rangle + p(x) + \frac{1}{2\sigma_k}\|x - x^k\|^2 + \frac{\tau}{2\sigma_k}\|y - \mathcal{A}x^k\|^2 \mid \mathcal{A}x - y = 0\}. \quad (4.16)$$

As discussed before, we can solve (4.16) by the dual Newton algorithm. The dual of (4.16) is given as

$$\begin{aligned}\max_u\{\psi_k(u) := &-\frac{\tau}{2\sigma_k}\|\mathcal{A}x^k + \frac{\sigma_k}{\tau}u\|^2 + \frac{\tau}{\sigma_k}\mathrm{E}_{\sigma_k h/\tau}(\mathcal{A}x^k + \frac{\sigma_k}{\tau}u) + \frac{\tau}{2\sigma_k}\|\mathcal{A}x^k\|^2 \\ &- \frac{1}{2\sigma_k}\|x^k + \sigma_k(c - \mathcal{A}^*u)\|^2 + \frac{1}{\sigma_k}\mathrm{E}_{\sigma_k p}(x^k + \sigma_k(c - \mathcal{A}^*u)) + \frac{1}{2\sigma_k}\|x^k\|^2\}.\end{aligned} \quad (4.17)$$

As long as we can obtain $\bar{u}^{k+1} \in \arg\max \psi_k(u)$, the update of x in the preconditioned PPA will be obtained by

$$\bar{x}^{k+1} = \mathrm{Prox}_{\sigma_k p}(x^k + \sigma_k c - \sigma_k\mathcal{A}^*\bar{u}^{k+1}).$$

Therefore, one can still apply the general algorithmic framework Algorithm 9 to solve (4.3) in this case by replacing $\phi_k(\cdot)$ in (4.12) by $\psi_k(\cdot)$ in (4.17). The following

proposition shows that the stopping criteria (A) and (B) can be achieved by using u^{k+1} and x^{k+1}. The result is adapted from [42].

Proposition 11. *Suppose we use the preconditioned PPA to solve (4.3) with $\mathcal{M}_k \equiv I_n + \tau\mathcal{A}^*\mathcal{A}$, where τ is a given positive number. The stopping criteria (A) and (B) can be achieved by the following two implementable ones:*

$$f_k(x^{k+1}) - \psi_k(u^{k+1}) \le \frac{\epsilon_k^2}{2\sigma_k}, \quad \epsilon_k \ge 0, \quad \sum_{k=0}^{\infty} \epsilon_k < \infty, \tag{A"}$$

$$f_k(x^{k+1}) - \psi_k(u^{k+1}) \le \frac{\delta_k^2}{2\sigma_k}\|x^{k+1} - x^k\|_{\mathcal{M}_k}^2, \quad 0 \le \delta_k < 1, \quad \sum_{k=0}^{\infty} \delta_k < \infty, \tag{B"}$$

where $f_k(\cdot)$ is defined in (4.9) and $\psi_k(\cdot)$ is defined in (4.17).

Proof. By noting that

$$f_k(x) = f(x) + \frac{1}{2\sigma_k}\|x - x^k\|_{\mathcal{M}_k}^2,$$

we know from [64, Exercise 8.8] that

$$\partial f_k(x) = \partial f(x) + \frac{1}{\sigma_k}\mathcal{M}_k(x - x^k).$$

Since $\mathcal{P}_k(x^k) = \arg\min f_k(x)$, we have that $0 \in \partial f_k(\mathcal{P}_k(x^k))$, which means there exists $v \in \partial f(\mathcal{P}_k(x^k))$ such that

$$0 = v + \frac{1}{\sigma_k}\mathcal{M}_k(\mathcal{P}_k(x^k) - x^k).$$

Since $f_k(\mathcal{P}_k(x^k)) = \inf f_k$, it holds that

$$\begin{aligned} f_k(x^{k+1}) - \inf f_k &\ge f(x^{k+1}) - \inf f_k + \frac{1}{2\sigma_k}\|x^{k+1} - x^k\|_{\mathcal{M}_k}^2 - \frac{1}{2\sigma_k}\|\mathcal{P}_k(x^k) - x^k\|_{\mathcal{M}_k}^2 \\ &= f(x^{k+1}) - \inf f_k + \frac{1}{2\sigma_k}\langle x^{k+1} + \mathcal{P}_k(x^k) - 2x_k, x^{k+1} - \mathcal{P}_k(x^k)\rangle_{\mathcal{M}_k} \\ &\ge \langle v, x^{k+1} - \mathcal{P}_k(x^k)\rangle + \frac{1}{2\sigma_k}\langle x^{k+1} + \mathcal{P}_k(x^k) - 2x_k, x^{k+1} - \mathcal{P}_k(x^k)\rangle_{\mathcal{M}_k} \\ &= \frac{1}{2\sigma_k}\|x^{k+1} - \mathcal{P}_k(x^k)\|_{\mathcal{M}_k}^2. \end{aligned}$$

By the strongly duality, we know that $\inf f_k = \sup \psi_k$, thus

$$\frac{1}{2\sigma_k}\|x^{k+1} - \mathcal{P}_k(x^k)\|_{\mathcal{M}_k}^2 \le f_k(x^{k+1}) - \inf f_k = f_k(x^{k+1}) - \sup \psi_k \le f_k(x^{k+1}) - \psi_k(u^{k+1}).$$

Therefore, the stopping criteria (A) and (B) can be achieved by:

$$f_k(x^{k+1}) - \psi_k(u^{k+1}) \le \frac{\epsilon_k^2}{2\sigma_k}, \quad \epsilon_k \ge 0, \quad \sum_{k=0}^{\infty} \epsilon_k < \infty, \tag{A"}$$

$$f_k(x^{k+1}) - \psi_k(u^{k+1}) \le \frac{\delta_k^2}{2\sigma_k}, \quad 0 \le \delta_k < 1, \quad \sum_{k=0}^{\infty} \delta_k < \infty. \tag{B"}$$

□

Now we discuss how to solve (4.17). As one can see, ψ_k is continuously differentiable with

$$\nabla\psi_k(u) = -\mathrm{Prox}_{\sigma_k h/\tau}(\mathcal{A}x^k + \frac{\sigma_k}{\tau}u) + \mathcal{A}\mathrm{Prox}_{\sigma_k p}(x^k + \sigma_k(c - \mathcal{A}^*u)).$$

The surrogate generalized Jacobian of the Lipschitz continuous function $\nabla\psi_k(\cdot)$ at the point u can be defined as

$$\hat{\partial}^2\psi_k(u) := -\frac{\sigma_k}{\tau}\partial\mathrm{Prox}_{\sigma_k h/\tau}(\mathcal{A}x^k + \frac{\sigma_k}{\tau}u) - \sigma_k\mathcal{A}\partial\mathrm{Prox}_{\sigma_k p}(\tilde{x} + \sigma_k c - \sigma_k\mathcal{A}^*u)\mathcal{A}^*.$$

Under some conditions on h, e.g. the elements in $\partial\mathrm{Prox}_{\nu h}(\cdot)$ are positive definite for any $\nu > 0$, one can still apply the SSN algorithm to solve (4.17) just as in the previous subsection.

Remark 4.2. *Consider the logistic regression problem, i.e.* $h(y) = \sum_{i=1}^m \log(1 + \exp(-b_i y_i))$, *for some given vector* $b \in \{-1, 1\}^m$. *Since it can be proved that* $h^*(\cdot)$ *satisfies Assumption 4.1, it is natural for us to apply the classical PPA (preconditioned PPA with* $\mathcal{M}_k \equiv I_n$*). Besides, we can also apply the preconditioned PPA with* $\mathcal{M}_k \equiv I_n + \tau\mathcal{A}^*\mathcal{A}$, *and the motivation is that the condition number of the linear system in the SSN will be better. For the proximal mapping of* h, *it can be computed coordinate-wise via Newton's method efficiently.*

4.3 Closed-form solution to the proximal mapping of $\rho\|w \circ \cdot\|_1^2$ and its generalized Jacobian

Due to the discussion in the previous section, when we want to solve the exclusive lasso model with the PPDNA algorithm, we need the proximal mapping $\mathrm{Prox}_p(\cdot)$

and its generalized Jacobian. In this section, for a given weight vector $w \in \Re^n_{++}$ and $\rho > 0$, we derive the closed-form solution to $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$ and its generalized Jacobian. From there, the results for $\mathrm{Prox}_p(\cdot)$ can be obtained readily.

4.3.1 Closed-form solution to $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$

We first consider the case when $a \geq 0$, then it is easy to see that $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a)$ could be equivalently computed via

$$\begin{aligned} x(a) &:= \arg\min_{x\in\Re^n}\{\frac{1}{2}\|x-a\|^2 + \rho\|w\circ x\|_1^2 \;\mid\; x\geq 0\} \\ &= \arg\min_{x\in\Re^n}\{\frac{1}{2}\|x-a\|^2 + \rho x^T(ww^T)x \;\mid\; x\geq 0\}. \end{aligned} \tag{4.18}$$

Note that since the objective function is strongly convex, the above minimization problem obtains a unique solution, which can be computed as in the following proposition.

Proposition 12. *Given $\rho > 0$ and $a \in \Re^n_+$. Let $a_w \in \Re^n$ be defined as $(a_w)_i := a_i/w_i$, for $i = 1, \cdots, n$ (here, we assume $w > 0$). There exists a permutation matrix Π_{a_w} such that $\Pi_{a_w} a_w$ is sorted in a non-increasing order. Denote $\tilde{a} = \Pi_{a_w} a$, $\tilde{w} = \Pi_{a_w} w$, and*

$$s_i = \sum_{j=1}^{i} \tilde{w}_j\tilde{a}_j, \quad L_i = \sum_{j=1}^{i}\tilde{w}_j^2, \quad \alpha_i = \frac{s_i}{1+2\rho L_i}, \quad i = 1,2,\ldots,n.$$

Let $\bar{\alpha} = \max_{1\leq i\leq n}\alpha_i$. Then, $x(a)$ defined in (4.18) can be computed as $x(a) = (a - 2\rho\bar{\alpha}w)^+$.

Proof. The Karush-Kuhn-Tucker (KKT) conditions for (4.18) are given by

$$x - a + 2\rho ww^T x + \mu = 0,\ \mu\circ x = 0,\ \mu \leq 0,\ x \geq 0, \tag{4.19}$$

where $\mu \in \Re^n$ is the dual multiplier. If (x^*, μ^*) satisfies the KKT conditions (4.19), by denoting $\beta = w^T x^*$, we can see that

$$x^* + \mu^* = a - 2\rho\beta w,\ \mu^*\circ x^* = 0,\ \mu^* \leq 0,\ x^* \geq 0.$$

Therefore, (x^*, μ^*) have the representations:

$$x^* = (a - 2\rho\beta w)^+, \quad \mu^* = -(a - 2\rho\beta w)^-.$$

Then our aim is to find the value of β. By the definition of β, we can see that

$$\beta = \sum_{i=1}^{n} w_i x_i^* = \sum_{i=1}^{n} w_i (a_i - 2\rho\beta w_i)^+ = \sum_{i=1}^{n} w_i^2 ((a_w)_i - 2\rho\beta)^+ = \sum_{i=1}^{n} \tilde{w}_i^2 ((\Pi_{a_w} a_w)_i - 2\rho\beta)^+.$$

Note that there must exist j such that $(\Pi_{a_w} a_w)_j > 2\rho\beta$, otherwise, we have $\beta = 0$, which contradicts the above formula. Since $\Pi_{a_w} a_w$ is sorted in a non-increasing order, there exists k such that $(\Pi_{a_w} a_w)_i = \tilde{a}_i / \tilde{w}_i \geq 2\rho\beta$, for all $i = 1, \ldots, k$, and $(\Pi_{a_w} a_w)_i = \tilde{a}_i / \tilde{w}_i < 2\rho\beta$, for all $i = k+1, \ldots, n$. Therefore,

$$\beta = \sum_{i=1}^{k} \tilde{w}_i^2 ((\Pi_{a_w} a_w)_i - 2\rho\beta) = \sum_{i=1}^{k} \tilde{w}_i \tilde{a}_i - 2\rho\beta \sum_{i=1}^{k} \tilde{w}_i^2 = s_k - 2\rho\beta L_k,$$

which means

$$\beta = \frac{s_k}{1 + 2\rho L_k}.$$

Claim that $\beta = \bar{\alpha}$, where $\bar{\alpha}$ is defined as in the statement of the proposition. Since

$$\begin{aligned} 0 \leq \alpha_k - \alpha_{k-1} = \frac{s_k}{1 + 2\rho L_k} - \frac{s_{k-1}}{1 + 2\rho L_{k-1}} &= \frac{(1 + 2\rho L_{k-1}) s_k - (1 + 2\rho L_k) s_{k-1}}{(1 + 2\rho L_k)(1 + 2\rho L_{k-1})} \\ &= \frac{\tilde{w}_k^2 (1 + 2\rho L_k)(\tilde{a}_k / \tilde{w}_k - 2\rho\alpha_k)}{(1 + 2\rho L_k)(1 + 2\rho L_{k-1})}, \end{aligned}$$

then, $\tilde{a}_k / \tilde{w}_k \geq 2\rho\alpha_k$. Furthermore,

$$\begin{aligned} 0 < \alpha_k - \alpha_{k+1} = \frac{s_k}{1 + 2\rho L_k} - \frac{s_{k+1}}{1 + 2\rho L_{k+1}} &= \frac{(1 + 2\rho L_{k+1}) s_k - (1 + 2\rho L_k) s_{k+1}}{(1 + 2\rho L_k)(1 + 2\rho L_{k+1})} \\ &= \frac{\tilde{w}_{k+1}^2 (1 + 2\rho L_k)(2\rho\alpha_k - \tilde{a}_{k+1} / \tilde{w}_{k+1})}{(1 + 2\rho L_k)(1 + 2\rho L_{k-1})}, \end{aligned}$$

which implies that $2\rho\alpha_k > \tilde{a}_{k+1} / \tilde{w}_{k+1}$. As a result, $\tilde{a}_1 / \tilde{w}_1 \geq \cdots \geq \tilde{a}_k / \tilde{w}_k \geq 2\rho\alpha_k > \tilde{a}_{k+1} / \tilde{w}_{k+1} \geq \cdots \geq \tilde{a}_n / \tilde{w}_n$. By the definition of κ, we can see that $\beta = \bar{\alpha}$.

Therefore, since the solution to (4.18) is unique, we have

$$x(a) = x^* = (a - 2\rho\beta w)^+ = (a - 2\rho\bar{\alpha} w)^+.$$

□

With the results above, we now give the closed-form solution to $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a)$ for any $a \in \Re^n$.

Proposition 13. *For given $\rho > 0$ and $a \in \Re^n$, we have*

$$\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a) = \mathrm{sign}(a) \circ \mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(|a|) = \mathrm{sign}(a) \circ x(|a|),$$

where $x(|a|)$ is defined in (4.18) and can be computed by Proposition 12. Consequently, $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a)$ can be computed in $O(n \log n)$ operations.

Proof. Since $\|w \circ x\|_1^2$ is invariant to the changes of signs, the conclusions of this proposition hold. □

Remark 4.3. *The closed form solution to the proximal mapping of $\rho\|w \circ \cdot\|_1^2$ is consistent with the result in [35, Proposition 4]. However, in section 4.1 of [35], after a change of variables, the author try to find the optimal solution of a constrained optimization problem by directly setting the gradient to zero (equations (23) and (24) in [35]). Although the obtained closed-form solution formula is correct, but this heuristic proof is not mathematically rigorous.*

4.3.2 The generalized Jacobian of $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$

In order to design the SSN method to solve the nonsmooth equations involving $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$, it is critical for us to derive an explicit formula for some form of the generalized Jacobian of $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$. Here, we derive a specific element in the set of the so-called HS-Jacobian of $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$ based on the quadratic programming (QP) reformulation of $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$.

By Proposition 13, we know that in order to get the generalized Jacobian of $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$, we need to study the generalized Jacobian of $x(a)$ first. For any $a \in \Re^n_+$, if we denote

$$Q = I_n + 2\rho w w^T \in \Re^{n\times n},$$

(4.18) can be equivalently written as

$$x(a) = \arg\min_{x\in\Re^n} \Big\{ \frac{1}{2}\langle x, Qx\rangle - \langle x, a\rangle \mid x \geq 0 \Big\}. \tag{4.20}$$

Based on the above strongly convex QP, we can derive the HS-Jacobian of $x(a)$ by applying the general results established in [29, 40], which will be described in the following paragraphs.

As one can see from (4.19) and the fact that $x(a)$ admits a unique solution, the corresponding dual multiplier μ also has a unique solution, which can be denoted as $\mu(a)$. The optimality of $x(a)$ given in (4.19) can be equivalently given as

$$Qx(a) - a + \mu(a) = 0,\ \mu(a)^T x(a) = 0,\ \mu(a) \leq 0,\ x(a) \geq 0. \tag{4.21}$$

Denote the active set

$$I(a) := \{i \in \{1, \ldots, n\} \mid x(a) = 0\}. \tag{4.22}$$

Now, we define a collection of index sets:

$$\mathcal{K}(a) := \{\ K \subseteq \{1, \ldots, n\} \mid K \subseteq I(a), \mathrm{supp}(\mu(a)) \subseteq K, I_K \text{ is of full row rank}\},$$

where $\mathrm{supp}(\mu(a))$ denotes the set of indices i such that $\mu(a)_i \neq 0$ and I_K is the matrix obtained from the rows of I_n, indexed by K. Note that the set $\mathcal{K}(a)$ is non-empty [29]. Since the B-subdifferential $\partial_B x(a)$ is usually very difficult to compute, in [40], generalizing the concept in [29], we define the following multi-valued mapping $\partial_{\mathrm{HS}} x(a)$: $\Re^n \rightrightarrows \Re^{n\times n}$:

$$\partial_{\mathrm{HS}} x(a) := \left\{ P \in \Re^{n\times n} \mid P = Q^{-1} - Q^{-1} I_K^T \left(I_K Q^{-1} I_K^T\right)^{-1} I_K Q^{-1}, K \in \mathcal{K}(a) \right\} \tag{4.23}$$

as a computational replacement for $\partial_B x(a)$. The set $\partial_{\mathrm{HS}} x(a)$ is known as the HS-Jacobian of $x(\cdot)$ at a. The following proposition from [40] provides a computationally efficient way to get an element in $\partial_{\mathrm{HS}} x(a)$ without the need to compute $\mu(a)$.

Proposition 14. *([40, Proposition 2]) For $a \in \Re^n_+$, there exists a neighborhood U of a such that for any $a' \in U$, it holds that $\mathcal{K}(a') \subseteq \mathcal{K}(a)$, $\partial_{\mathrm{HS}} x(a') \subseteq \partial_{\mathrm{HS}} x(a)$. If $\mathcal{K}(a') \subseteq \mathcal{K}(a)$, then*

$$x(a') = x(a) + P(a' - a), \quad \forall P \in \partial_{\mathrm{HS}} x(a').$$

Furthermore, let $I(a)$ *be given in* (4.22)*, then*

$$\Omega := Q^{-1} - Q^{-1}I_{I(a)}^T(I_{I(a)}Q^{-1}I_{I(a)}^T)^\dagger I_{I(a)}Q^{-1} \in \partial_{\mathrm{HS}}x(a).$$

Based on the results in Proposition 13 and Proposition 14, we can now compute a specific element in $\partial_{\mathrm{HS}}\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$ at any $a \in \Re^n$ as follows.

Theorem 4.3. *Given* $a \in \Re^n$*,* $\rho > 0$*, the HS-Jacobian* $\partial_{\mathrm{HS}}\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a)$ *can be given as*

$$\partial_{\mathrm{HS}}\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a) = \left\{ \ \Theta P\Theta \mid P \in \partial_{\mathrm{HS}}x(|a|) \ \right\},$$

where $\partial_{\mathrm{HS}}x(\cdot)$ *is defined as in* (4.23) *and* $\Theta = \mathrm{Diag}(\mathrm{sign}(a))$*. Moreover,* $M_0 \in \Re^{n\times n}$ *given by*

$$M_0 := \Theta P_0 \Theta, \ \textit{with}\ P_0 = Q^{-1} - Q^{-1}I_{I(|a|)}^T\left(I_{I(|a|)}Q^{-1}I_{I(|a|)}^T\right)^\dagger I_{I(|a|)}Q^{-1}. \tag{4.24}$$

is an element in the HS-Jacobian $\partial_{\mathrm{HS}}\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a)$*.*

For efficient implementation, we can use the result in the following proposition to compute M_0 in (4.24).

Proposition 15. *Define* $\xi \in \Re^n$ *with* $\xi_i = 1$ *if* $i \in I(|a|)$*, and* $\xi_i = 0$ *otherwise, and* $\Xi = \mathrm{Diag}(\xi)$*. Then denoting* $\tilde{w} = \Theta(I_n - \Xi)w$*,* M_0 *defined in* (4.24) *can be computed as*

$$M_0 = I_n - \Xi - \frac{2\rho}{1 + 2\rho(\tilde{w}^T\tilde{w})}\tilde{w}\tilde{w}^T.$$

Proof. It can be proved that

$$I_{I(|a|)}^T\left(I_{I(|a|)}Q^{-1}I_{I(|a|)}^T\right)^\dagger I_{I(|a|)} = (\Xi Q^{-1}\Xi)^\dagger = \Xi(\Xi Q^{-1}\Xi)^\dagger\Xi,$$

where the last inequality follows from the fact that Ξ is a 0-1 diagonal matrix. Then by [40, Proposition 3], we can see that

$$\begin{aligned} P_0 &= Q^{-1} - Q^{-1}I_{I(|a|)}^T\left(I_{I(|a|)}Q^{-1}I_{I(|a|)}^T\right)^\dagger I_{I(|a|)}Q^{-1} \\ &= Q^{-1} - Q^{-1}\Xi(\Xi Q^{-1}\Xi)^\dagger\Xi Q^{-1} \\ &= (\Sigma Q\Sigma)^\dagger, \end{aligned}$$

where $\Sigma = I_n - \Xi$, is also a 0-1 diagonal matrix. Since $Q = I_n + 2\rho ww^T \in \Re^{n\times n}$, denoting $\hat{w} = \Sigma w$, we have that

$$P_0 = (\Sigma Q \Sigma)^\dagger = (\Sigma + 2\rho \hat{w}\hat{w}^T)^\dagger = \Sigma - \frac{2\rho}{1 + 2\rho(\hat{w}^T\hat{w})}\hat{w}\hat{w}^T.$$

Note that $\Theta = \mathrm{Diag}(\mathrm{sign}(a))$ is a diagonal matrix with its diagonal elements being 1 or -1. Thus

$$M_0 = \Theta\Omega\Theta = \Theta(\Sigma - \frac{2\rho}{1 + 2\rho(\hat{w}^T\hat{w})}\hat{w}\hat{w}^T)\Theta = \Sigma - \frac{2\rho}{1 + 2\rho(\hat{w}^T\hat{w})}\tilde{w}\tilde{w}^T = \Sigma - \frac{2\rho}{1 + 2\rho(\tilde{w}^T\tilde{w})}\tilde{w}\tilde{w}^T,$$

where $\tilde{w} = \Theta\hat{w} = \Theta(I_n - \Xi)w$. □

As one can see, $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$ is piecewise linear and Lipschitz continuous, thus it is directionally differentiable [25]. And from Proposition 14, we can obtain that there exists a neighborhood $\mathcal{U}$ of a such that for all $a' \in \mathcal{U}$,

$$\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a') - \mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a) - M(a' - a) = 0, \quad \forall M \in \partial_{\mathrm{HS}}\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a').$$

Therefore, $\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$ is strongly semismooth with respect to $\partial_{\mathrm{HS}}\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$.

4.4 Numerical experiments

In this section, we perform some numerical experiments to test our proposed algorithm PPDNA to solve the commonly used exclusive lasso model. For simplicity, we take the weight vector w to be all ones. The exclusive lasso model can be described as

$$\min_{x\in\mathbb{R}^n}\Big\{h(Ax) + \lambda\sum_{g\in\mathcal{G}}\|x_g\|_1^2\Big\}, \tag{4.25}$$

where $\mathcal{G} = \{g \mid g \subseteq \{1, 2, \ldots, n\}\}$ is a disjoint partition of $\{1, 2, \ldots, n\}$, $A \in \mathbb{R}^{m\times n}$ and $\lambda > 0$. By taking $c = 0$, and $p(x) = \lambda\sum_{g\in\mathcal{G}}\|x_g\|_1^2$, we can reformulate (4.25) in the form of (4.3). Thus, all the analyses in previous sections are applicable for the above model. All our computational results are obtained by running MATLAB on a windows workstation (12-core, Intel Xeon E5-2680 @ 2.50GHz, 128G RAM).

In the numerical experiments, we mainly focus on two aspects. (1) We compare our proposed PPDNA for solving (4.25) to three popular state-of-the-art first-order methods, ILSA [34], ADMM with step length $\kappa = 1.618$ [26] and APG with restart under the setting described in [9]. To demonstrate the efficiency and scalability of PPDNA, we perform the time comparison on synthetic datasets over a range of problem dimensions. (2) We apply the exclusive lasso model (4.25) with least squares loss function to ETF index tracking in finance. The out-of-sample results show the superior performance of the exclusive lasso model in index tracking, comparing to the lasso model and the group lasso model.

We stop all the four algorithms by the following criterion based on the relative KKT residual:

$$\eta_{\text{KKT}} := \frac{\|x - \text{Prox}_p(x - A^T\nabla h(Ax)\|}{1 + \|x\| + \|A^T\nabla h(Ax)\|} \leq \epsilon,$$

where $\epsilon > 0$ is a given tolerance, which is set to **1e-6** in our experiments. We also terminate PPDNA when it reaches the maximum iteration count of **1e2** and terminate ILSA, ADMM and APG when they reach the maximum iteration count of **5e5**. In addition, we set the maximum computation time as 1 hour.

4.4.1 The regularized linear regression problem with synthetic data

In this subsection, we test the efficiency of PPDNA for solving (4.25) with $h(y) := \sum_{i=1}^m (y_i - b_i)^2/2$, and compare it against ILSA, ADMM and APG on simulated datasets. In this case, we apply the classical PPDNA, which means $\mathcal{M}_k \equiv I_n$. Here we focus on the time comparison among the algorithms. For the comparison of prediction errors among the exclusive lasso, lasso and other linear regression models, we refer the readers to [11] for more details.

In the experiments, we adopt the design of synthetic datasets described in [11]. We generate the synthetic data using the model $b = Ax^* + \epsilon$, where x^* is the predefined true solution and $\epsilon \sim \mathcal{N}(0, I_m)$ is a random noise vector. Given the

number of observations m, the number of groups s and the number of features p in each group, for the matrix $A \in \mathbb{R}^{m\times sp}$, we generate each row of A by sampling a vector from a multivariate normal distribution $\mathcal{N}(0,\Sigma)$, where Σ is a Toeplitz covariance matrix with entries $\Sigma_{ij} = 0.9^{|i-j|}$ for features in the same group, and $\Sigma_{ij} = 0.3^{|i-j|}$ for features in different groups. For the ground-truth x^*, we randomly generate 10 nonzero elements in each group with values drawn i.i.d from the uniform distribution over the interval $[0, 10]$.

Here we mainly focus on feature selection by the exclusive lasso model in the high-dimensional settings. Hence, we fix m to be 200 and s to be 20, but vary the number of features p in each group from 50 to 1000. That is, we vary the total number of features $n = sp$ from 1000 to 20000. To compare the robustness of different algorithms with respect to the hyperparameter λ, we test all the algorithms under two widely different values of λ. The results are shown in Figure 4.1, which demonstrate the superior performance of PPDNA, especially for large-scale instances, comparing to ILSA, ADMM and APG.

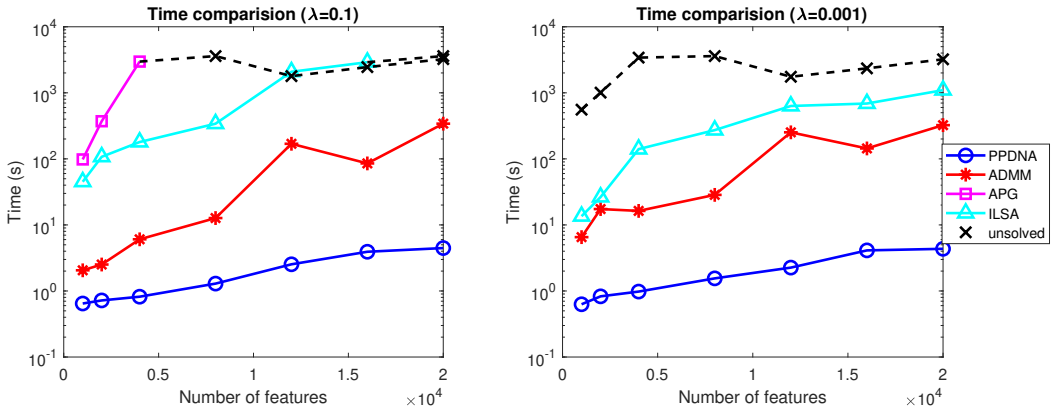


Figure 4.1: Time comparison among PPDNA, ILSA, ADMM and APG for linear regression on synthetic datasets. The black dash line with '×' represents that the algorithm fails to solve the instance.

More results on higher dimensional cases are shown in Table 4.1. As one can see from Figure 4.1, APG and ILSA are not efficient enough to solve large-scale

instances, thus we only compare PPDNA with ADMM in these cases.

Table 4.1: Time comparison between PPDNA and ADMM for linear regression on synthetic datasets. Time is in the format of (hours:minutes:seconds). Bold fonts mean that the algorithm fails to solve the instance.

		iter	η_{KKT}	time
Data (m, s, p)	λ	PPDNA | ADMM	PPDNA | ADMM	PPDNA | ADMM
$(500, 20, 2000)$	1e-1	23 | 23332	8.5e-7 | 1.0e-6	00:00:37 | 00:08:08
	1e-3	30 | 167472	6.3e-7 | **1.5e-6**	00:00:33 | 01:00:00
$(500, 20, 3000)$	1e-1	23 | 46226	3.9e-7 | **2.1e-6**	00:00:48 | 01:00:00
	1e-3	29 | 50402	7.9e-7 | **9.0e-6**	00:00:49 | 01:00:01
$(1000, 20, 2000)$	1e-1	21 | 16208	5.0e-7 | 1.0e-6	00:01:27 | 00:09:03
	1e-3	28 | 89242	7.8e-7 | 1.0e-6	00:01:48 | 00:50:41
$(1000, 20, 4000)$	1e-1	22 | 15644	7.1e-7 | **1.2e-5**	00:02:03 | 01:00:00
	1e-3	29 | 15680	9.7e-7 | **3.6e-3**	00:02:28 | 01:00:01

4.4.2 The regularized logistic regression problem with synthetic data

In this subsection, we show the performance of PPDNA for solving the logistic regression model with the exclusive lasso regularizer. The logistic regression model could be formulated by taking

$$h(y) = \sum_{i=1}^{m} \log(1 + \exp(-b_i y_i))$$

in (4.25), where $b \in \{-1, 1\}^m$ is given. For robustness, we apply the preconditioned PPA with $\mathcal{M}_k \equiv I_n + \tau A^* A$ to solve this exclusive lasso model with $\tau = 1/\lambda_{\max}(AA^T)$.

We use the same synthetic datasets described in the previous subsection, except for letting $b_i = 1$ if $Ax^* + \epsilon \geq 0$, and $b_i = -1$ if otherwise. As one can see in the previous subsection, APG and ILSA are very time-consuming when solving

large-scale exclusive lasso problems compared to PPDNA and ADMM. Thus for logistic regression problems, we only compare PPDNA with ADMM (we describe the ADMM for solving the regularized logistic regression problem in Remark 4.4). The numerical results are shown in Table 4.2.

Table 4.2: Time comparison between PPDNA and ADMM for logistic regression on synthetic datasets. Time is in the format of (hours:minutes:seconds). Bold fonts mean that the algorithm fails to solve the instance.

		iter	η_{KKT}	time
Data (m, s, p)	λ	PPDNA | ADMM	PPDNA | ADMM	PPDNA | ADMM
$(500, 20, 3000)$	1e-1	13 | 1689	5.2e-7 | 1.0e-6	00:00:19 | 00:02:27
	1e-3	48 | 5850	9.4e-7 | 1.0e-6	00:00:27 | 00:06:38
	1e-5	73 | 17208	9.3e-7 | 1.0e-6	00:00:37 | 00:16:52
$(500, 20, 5000)$	1e-1	12 | 2167	2.6e-7 | 1.0e-6	00:00:32 | 00:04:45
	1e-3	37 | 6187	2.1e-7 | 1.0e-6	00:00:38 | 00:10:17
	1e-5	67 | 21584	8.9e-7 | 1.0e-6	00:00:52 | 00:33:54
$(1000, 20, 5000)$	1e-1	13 | 1186	2.9e-7 | 1.0e-6	00:01:09 | 00:06:12
	1e-3	47 | 5593	6.6e-7 | 1.0e-6	00:01:35 | 00:22:45
	1e-5	66 | 17829	9.9e-7 | **2.2e-6**	00:01:45 | 01:00:00
$(1000, 20, 8000)$	1e-1	13 | 1947	9.1e-8 | 1.0e-6	00:01:58 | 00:14:02
	1e-3	57 | 6991	9.2e-7 | 1.0e-6	00:02:41 | 00:39:01
	1e-5	89 | 10519	9.7e-7 | **1.4e-5**	00:03:40 | 01:00:00
$(2000, 20, 10000)$	1e-1	11 | 1625	7.3e-7 | 1.0e-6	00:04:32 | 00:33:02
	1e-3	62 | 3522	6.0e-7 | **5.8e-5**	00:07:16 | 01:00:05
	1e-5	79 | 4415	9.9e-7 | **2.5e-4**	00:08:32 | 01:00:05

Remark 4.4. *The minimization form of the dual of* (4.3) *is given as*

$$\min_{w,u\in\Re^m,v\in\Re^n} \{h^*(w) + p^*(v) \mid \mathcal{A}^*u + v - c = 0, w - u = 0\}. \tag{4.26}$$

The augmented Lagrangian function associated with (4.26) *is*

$$\begin{aligned}\mathcal{L}_\sigma(w,u,v;x,y) =& h^*(w)+p^*(v)-\langle x,\mathcal{A}^*u+v-c\rangle-\langle y,w-u\rangle\\ &+\frac{\sigma}{2}\|\mathcal{A}^*u+v-c\|^2+\frac{\sigma}{2}\|w-u\|^2.\end{aligned}$$

The alternating direction method of multipliers (ADMM) for solving (4.3) *and* (4.26) *could be described as follows:*

$$u^{k+1}=\arg\min_u \mathcal{L}_\sigma(w^k,u,v^k;x^k,y^k), \tag{4.27a}$$

$$(w^{k+1},v^{k+1})=\arg\min_{w,v} \mathcal{L}_\sigma(w,u^{k+1},v;x^k,y^k), \tag{4.27b}$$

$$x^{k+1}=x^k-\kappa\sigma(\mathcal{A}^*u^{k+1}+v^{k+1}-c),\quad y^{k+1}=y^k-\kappa\sigma(w^{k+1}-u^{k+1}), \tag{4.27c}$$

where the step length $\kappa=1.618$ *and* $\sigma>0$ *is a given parameter. For the subproblem* (4.27b)*, we have that*

$$\begin{aligned}w^{k+1}&=\mathrm{Prox}_{h^*/\sigma}(u^{k+1}+y^k/\sigma)\\ &=(u^{k+1}+y^k/\sigma)-\frac{1}{\sigma}\mathrm{Prox}_{\sigma h}(\sigma u^{k+1}+y^k),\\ v^{k+1}&=\mathrm{Prox}_{p^*/\sigma}(-\mathcal{A}^*u^{k+1}+c+x^k/\sigma)\\ &=(-\mathcal{A}^*u^{k+1}+c+x^k/\sigma)-\frac{1}{\sigma}\mathrm{Prox}_{\sigma p}(-\sigma\mathcal{A}^*u^{k+1}+\sigma c+x^k),\end{aligned}$$

where the Moreau identity $\mathrm{Prox}_{tp}(x)+t\mathrm{Prox}_{f^*/t}(x/t)=x$ *is used. For the subproblem* (4.27a)*, the optimality condition is*

$$(I_m+\mathcal{A}\mathcal{A}^*)u=\mathcal{A}(c+x^k/\sigma-v^k)+(w^k-y^k/\sigma).$$

One can solve this linear system by a direct solver or use an iterative solver such as the preconditioned conjugate gradient method.

4.4.3 Application: index exchange-traded fund (index ETF)

In this subsection, we apply the exclusive lasso model in a realistic application in finance. Consider the portfolio selection problem where a fund manager wants to select a small subset of stocks (to minimize transaction costs and business analyses)

to track a target time series such as the S&P 500 index. Furthermore, in order to diversify the risks, the portfolio is required to span across all sectors. Such a problem naturally leads us to consider the exclusive lasso model for this application.

In our experiments, we download all stock price data in the US market between 2018-01-01 and 2018-12-31 (251 trading days) from Yahoo finance [1]. We drop the stock if more than 10% of its price data is missing. After that, we get 3074 stocks in our stock universe. For the remaining stocks, we handle the missing data via the common practice of forward interpolation. We then compute the daily return and get the historical return matrix $R \in \mathbb{R}^{250\times 3074}$. We try to build a portfolio to track the popular S&P 500 index. Let $y \in \mathbb{R}^{250}$ be the daily return of the S&P 500 index in 2018. Since there are 12 sectors in the US market (e.g., finance, healthcare, technology, etc.), we have a natural group partition for our stock universe as $\mathcal{G}_{\text{US}} = \{g_1, g_2, \ldots, g_{12}\}$, where g_i is the index set for stocks in the i-th sector.

To test the performance of the exclusive lasso model in index tracking, we use the rolling window method to test the in-sample and out-of-sample performance of the model. We use the historical data in the last 90 trading days to estimate a portfolio vector via the model for the future 10 days. More specifically, at day T, we solve the following optimization problem (we explain why we could drop the constraints: $x \geq 0$ and $\sum_i x_i = 1$ in Remark 4.5.):

$$x_T^* = \arg\min_x \frac{1}{2}\|R_T x - y_T\|_2^2 + \lambda_T \textstyle\sum_{g\in\mathcal{G}_{\text{US}}} \|x_g\|_1^2,$$

where R_T, y_T are the daily return matrix of all stocks in our stock universe and S&P 500 index in the last 90 trading days prior to day T, respectively. We select the hyperparameter λ_T using 9-folds cross validation. After we get the estimated portfolio vector x_T^*, we invest in the market based on it for the next 10 trading days. The in-sample and out-of-sample performance of the exclusive lasso model, the lasso model and the group lasso model are shown in Figure 4.2.

Remark 4.5. *Here we explain why we can drop the simplex constraint $x \geq 0$, $\sum_i x_i = 1$ in the index ETF application. We assume that we can short stocks in the*

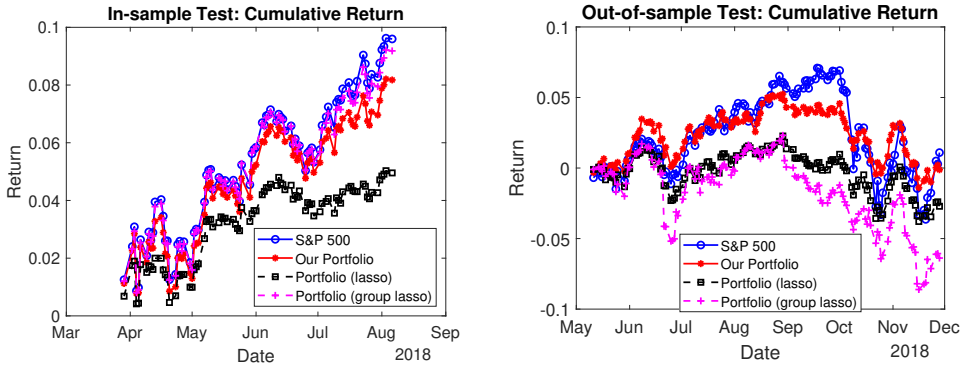


Figure 4.2: In-sample and out-of-sample performance of the exclusive lasso, the group lasso and the lasso model for index tracking of S&P 500.

market, which means we can drop the nonnegative constraint $x \geq 0$. *Furthermore, we assume the interest rate is* r_C. *Then, for a given return vector* $r \in \Re^n$ *of* n *stocks and a weight vector* x^*, *the return of the whole investment is given by*

$$r^T x^* + (1 - \sum_{i=1}^{n} x_i^*) r_C = \sum_{i=1}^{n} (r_i - r_C) x_i^* + r_C.$$

Then, if we assume $r_C = 0$, *or we set*

$$r = r - r_C, \quad y = y - r_C.$$

We could drop the constraint $\sum_{i=1}^{n} x_i = 1$ *in the index ETF model.*

For in-sample performance, we just show the selected results of one rolling window. The performance demonstrates the viability of using the exclusive lasso model in index tracking by selecting a small subset of about 80 stocks from the large universe of 3074 stocks. For comparison, the lasso model selects about 70 stocks and the group lasso model selects about 2000 since the group lasso could only enforce sparsity on the sector level.

We plot the percentage of stocks from each sector in the portfolio obtained from the three previously mentioned models in Figure 4.3. The result shows that our exclusive lasso model can select stocks from all the 12 sectors, but the lasso model

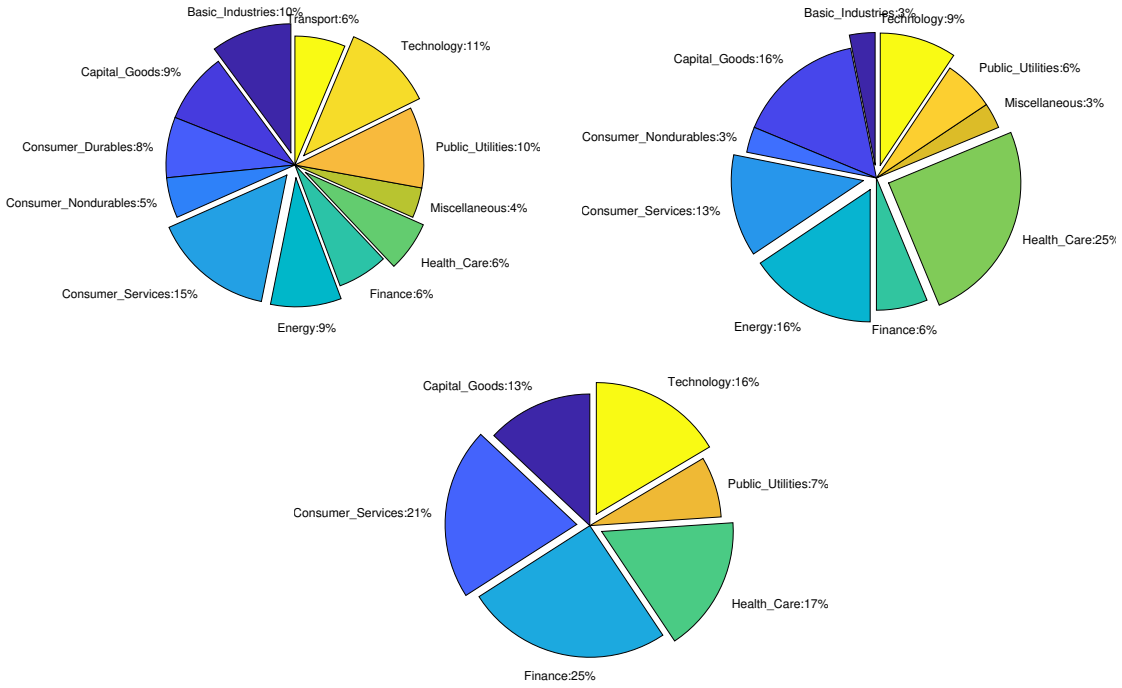


Figure 4.3: Percentage of selected stocks by sectors. Top left: exclusive lasso model. Top right: lasso model. Bottom: group lasso model.

selects stocks only from 10 sectors and the group lasso model selects stocks only from 6 sectors in the universe.

4.5 Summary of the chapter

In this chapter, we provide a rigorous proof for the closed-form solution to $\text{Prox}_{\rho\|w\circ\cdot\|_1^2}(\cdot)$ and derive its corresponding HS-Jacobian. Based on these theoretical results, we design an efficient and scalable second-order based algorithm PPDNA to solve the exclusive lasso model. We also apply the exclusive lasso model in index ETF, which achieves better out-of-sample performance comparing to the lasso model and group lasso model.

Chapter 5

Simultaneous Clustering and Feature Selection

In this section we focus on the sparse convex clustering model which can do clustering and intra-group level feature selection simultaneously for high dimensional data. We consider the following sparse convex clustering model:

$$\min_{X\in\Re^{p\times n}} \frac{1}{2}\|X-A\|_F^2 + \gamma_1 \sum_{i<j} w_{ij}\|X_{\cdot,i}-X_{\cdot,j}\|_2 + \gamma_2 \sum_{i=1}^{n} \|X_{\cdot,i}\|_1^2, \tag{5.1}$$

where γ_1, γ_2 are two positive hyper-parameters, $w_{ij} \geq 0$ are given weights, $A = [a_1, a_2, \ldots, a_n] \in \Re^{p\times n}$ and $X = [x_1, x_2, \ldots, x_n] \in \Re^{p\times n}$ are the concatenations of $\{a_1, a_2, \ldots, a_n\}$ and $\{x_1, x_2, \ldots, x_n\}$, respectively. Here $X_{\cdot,i} = x_i \in \Re^p$ is the i-th column of X.

By ignoring the terms with $w_{ij} = 0$, we consider the following problem

$$\min_{X\in\Re^{p\times n}} \frac{1}{2}\|X-A\|_F^2 + \gamma_1 \sum_{(i,j)\in\mathcal{E}} w_{ij}\|X_{\cdot,i}-X_{\cdot,j}\|_2 + \gamma_2 \sum_{i=1}^{n} \|X_{\cdot,i}\|_1^2, \tag{5.2}$$

where $\mathcal{E} := \{(i,j)|w_{ij} > 0\}$.

As discussed in section 1.1.3, the exclusive lasso regularization term $\sum_{i=1}^{n} \|x_i\|_1^2$ in the model (5.2) can help the convex clustering model also to perform the data point-wise feature selections simultaneously.

The remaining parts of this chapter will be organized as follows: we will introduce a semismooth Newton based augmented Lagrangian method (SSNAL) for solving three-blocks convex composite programming problem in section 5.1. In section 5.2, we will introduce a semismooth Newton method to solve the subproblem involved in the SSNAL framework. As necessary ingredients for applying SSNAL in solving the sparse convex clustering model (5.2), we will recap some results derived in Chapter 3 and Chapter 4 in section 5.3. In section 5.4, we will present some numerical results.

5.1 A semismooth Newton based augmented Lagrangian method

In this section, we will introduce a semismooth Newton based augmented Lagrangian method (SSNAL) for solving a class of three-blocks convex composite programming problem, which will include the sparse convex clustering model as a special case.

5.1.1 SSNAL for three-blocks convex composite programming problem

Consider the following three-blocks convex composite programming problem

$$\min_{X\in\mathcal{X}} \quad f(X) + g(\mathcal{A}X) + h(X), \tag{5.3}$$

where $f : \mathcal{X} \to \Re$ is a strongly convex and twice continuously differentiable function, $g : \mathcal{Y} \to \Re$, $h : \mathcal{X} \to \Re$ are convex, closed and proper functions, which could be nonsmooth. $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ is a linear mapping and $\mathcal{X}, \mathcal{Y}$ are finite dimensional Euclidean spaces. Furthermore, we assume that the proximal mappings $\mathrm{Prox}_{\rho g}(\cdot)$ and $\mathrm{Prox}_{\rho h}(\cdot)$ could be computed efficiently for any $\rho > 0$.

Note that (5.2) is a special case of (5.3) by taking

$$\mathcal{X} = \Re^{p\times n}, \quad \mathcal{Y} = \Re^{p\times|\mathcal{E}|},$$
$$f(X) = \frac{1}{2}\sum_{i=1}^{n}\|X_{\cdot,i} - A_{\cdot,i}\|_2^2, \quad g(Y) = \gamma_1 \sum_{(i,j)\in\mathcal{E}} w_{ij}\|Y_{ij}\|_2, \quad h(X) = \gamma_2 \sum_{i=1}^{n}\|X_{\cdot,i}\|_1^2,$$

where

$$\mathcal{A}X = [(X_{\cdot,i} - X_{\cdot,j})]_{(i,j)\in\mathcal{E}} \in \Re^{p\times|\mathcal{E}|}.$$

Next, we derive the dual problem of (5.3). First, we write (5.3) equivalently as

$$\min_{X,Y,Z}\{f(X) + g(Y) + h(Z) \mid \mathcal{A}X - Y = 0, X - Z = 0\}, \tag{P}$$

Denote the Lagrangian function of (P) as

$$l(X, Y, Z; V, W) = f(X) + g(Y) + h(Z) + \langle V, \mathcal{A}X - Y\rangle + \langle W, X - Z\rangle.$$

The minimization form of the dual problem for (P) is given by

$$\min_{V,W}\{f^*(-\mathcal{A}^*V - W) + g^*(V) + h^*(W)\}, \tag{D}$$

where $f^*(\cdot)$, $g^*(\cdot)$ and $h^*(\cdot)$ are the conjugate functions of $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$, respectively. The KKT condition for (P) and (D) is as follows:

$$(KKT) \quad \begin{cases} \nabla f(X) + \mathcal{A}^*V + W = 0, \\ Y - \mathrm{Prox}_g(V + Y) = 0, \\ Z - \mathrm{Prox}_h(W + Z) = 0, \\ \mathcal{A}X - Y = 0, \\ X - Z = 0. \end{cases}$$

Now, we introduce the augmented Lagrangian method (ALM) for solving (P), which will also solve the dual problem (D) as a byproduct. For a given parameter $\sigma > 0$, the augmented Lagrangian function associated with (P) is given by

$$\mathcal{L}_\sigma(X, Y, Z; V, W) = l(X, Y, Z; V, W) + \frac{\sigma}{2}(\|\mathcal{A}X - Y\|_2^2 + \|X - Z\|_2^2).$$

We describe the SSNAL algorithm for solving (P) in Algorithm 11.

Algorithm 11: SSNAL for (P)

Initialization: Choose $(X^0, Y^0, Z^0) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{X}$ and $(V^0, W^0) \in \mathcal{Y} \times \mathcal{X}$, $\sigma_0 > 0$. For $k = 0, 1, \ldots,$ repeat

Step 1 . Compute

$$(X^{k+1}, Y^{k+1}, Z^{k+1}) \approx \arg\min\{\Phi_k(X, Y, Z) = \mathcal{L}_{\sigma_k}(X, Y, Z; V^k, W^k)\} \quad (5.4)$$

to satisfy the conditions (A), (B1) and (B2).

Step 2 . Compute

$$V^{k+1} = V^k + \sigma_k(\mathcal{A}X^{k+1} - Y^{k+1}),$$

$$W^{k+1} = W^k + \sigma_k(X^{k+1} - Z^{k+1}).$$

Step 3 . Update $\sigma_{k+1} \uparrow \sigma_\infty \leq \infty$.

Until stopping criteria is satisfied.

To ensure the convergence of the algorithm, we need the following stopping criteria

$$\Phi_k(X^{k+1}, Y^{k+1}, Z^{k+1}) - \inf \Phi_k \leq \epsilon_k^2/2\sigma_k, \quad \sum_{k=0}^{\infty} \epsilon_k < +\infty, \quad \text{(A)}$$

$$\Phi_k(X^{k+1}, Y^{k+1}, Z^{k+1}) - \inf \Phi_k \leq (\delta_k^2/2\sigma_k)\|(V^{k+1}, W^{k+1}) - (V^k, W^k)\|^2, \quad \sum_{k=0}^{\infty} \delta_k < +\infty, \quad \text{(B1)}$$

$$\text{dist}(0, \partial\Phi_k(X^{k+1}, Y^{k+1}, Z^{k+1})) \leq (\delta_k'/\sigma_k)\|(V^{k+1}, W^{k+1}) - (V^k, W^k)\|, \quad 0 \leq \delta_k' \to 0, \quad \text{(B2)}$$

where $\{\epsilon_k\}$, $\{\delta_k\}$ and $\{\delta_k'\}$ are given sequences of nonnegative numbers.

The inexact augmented Lagrangian method is a well established algorithmic framework for solving convex programming problems [62, 63], which is proved to have an asymptotically superlinear convergence rate. The challenges for the inexact ALM is to solve (5.4) efficiently. In the next section, we will introduce the semismooth Newton method (SSN) to solve (5.4), which could achieve superlinear (even

quadratic) convergence rate.

5.2 Semismooth Newton-CG method for the subproblem (5.4)

In this section, we will design a semismooth Newton-CG algorithm to solve (5.4). For a given $\sigma > 0$, and $(\tilde{V}, \tilde{W}) \in \mathcal{Y} \times \mathcal{X}$, the subproblem in each iteration of Algorithm 11 is given by

$$\min_{X,Y,Z} \quad \Phi(X, Y, Z) = \mathcal{L}_\sigma(X, Y, Z; \tilde{V}, \tilde{W}). \tag{5.5}$$

Since $\Phi(X, Y, Z)$ is strongly convex, the optimization problem (5.5) admits a unique optimal solution, which we denote as $(\bar{X}, \bar{Y}, \bar{Z})$. Now, for any $X \in \mathcal{X}$, denote

$$\begin{aligned}
&\phi(X) := \inf_{Y,Z} \Phi(X, Y, Z) \\
&= f(X) + g(\mathrm{Prox}_{g/\sigma}(\mathcal{A}X + \tilde{V}/\sigma)) + \frac{\sigma}{2}\|(\mathcal{A}X + \tilde{V}/\sigma) - \mathrm{Prox}_{g/\sigma}(\mathcal{A}X + \tilde{V}/\sigma)\|^2 \\
&\quad + h(\mathrm{Prox}_{h/\sigma}(X + \tilde{W}/\sigma)) + \frac{\sigma}{2}\|(X + \tilde{W}/\sigma) - \mathrm{Prox}_{h/\sigma}(X + \tilde{W}/\sigma)\|^2 - \frac{1}{2\sigma}(\|\tilde{V}\|_2^2 + \|\tilde{W}\|_2^2).
\end{aligned} \tag{5.6}$$

Then, we can solve for $(\bar{X}, \bar{Y}, \bar{Z})$ as follows:

$$\bar{X} = \arg\min_X \phi(X), \quad \bar{Y} = \mathrm{Prox}_{g/\sigma}(\mathcal{A}\bar{X} + \tilde{V}/\sigma), \quad \bar{Z} = \mathrm{Prox}_{h/\sigma}(\bar{X} + \tilde{W}/\sigma).$$

It's not difficult to see that $\phi(X)$ is strongly convex and continuously differentiable. The gradient $\nabla\phi(X)$ of $\phi(X)$ is given by

$$\begin{aligned}
\nabla\phi(X) = \nabla f(X) + \sigma\mathcal{A}^*(\mathcal{A}X + \tilde{V}/\sigma) - \sigma\mathcal{A}^*\mathrm{Prox}_{g/\sigma}(\mathcal{A}X + \tilde{V}/\sigma) \\
+ \sigma(X + \tilde{W}/\sigma) - \sigma\mathrm{Prox}_{h/\sigma}(X + \tilde{W}/\sigma).
\end{aligned}$$

We know that $\bar{X}$ can be obtained by solving the following nonlinear equation

$$\nabla\phi(X) = 0. \tag{5.7}$$

As one can see, since $\mathrm{Prox}_{g/\sigma}(\cdot)$ and $\mathrm{Prox}_{h/\sigma}(\cdot)$ are Lipschitz continuous, $\nabla\phi(\cdot)$ is locally Lipschitz continuous. However, $\nabla\phi(X)$ is not differentiable.

As discussed in Chapter 3 and Chapter 4, $\mathrm{Prox}_{\rho\|\cdot\|_2}(\cdot)$ and $\mathrm{Prox}_{\rho\|\cdot\|_1^2}(\cdot)$ are strongly semismooth. Hence $\nabla\phi(\cdot)$ is strongly semismooth. This allows us to solve (5.7) by the semismooth Newton method, which enjoys the quadratic convergence rate [46].

In order to apply the semismooth Newton method, we now derive the generalized Jacobian of the locally Lipschitz function $\nabla\phi(\cdot)$. For a given $X \in \mathcal{X}$, the following set-valued map is well defined:

$$\hat{\partial}^2\phi(X) = \left\{ \nabla^2 f(X) + \sigma\mathcal{A}^*(\mathcal{I}-\mathcal{P})\mathcal{A} + \sigma(\mathcal{I}-\mathcal{Q}) \;\middle|\; \begin{array}{l} \mathcal{P} \in \partial\mathrm{Prox}_{g/\sigma}(\mathcal{A}X + \tilde{V}/\sigma) \\ \mathcal{Q} \in \partial\mathrm{Prox}_{h/\sigma}(X + \tilde{W}/\sigma) \end{array} \right\}, \tag{5.8}$$

where $\partial\mathrm{Prox}_{g/\sigma}(\mathcal{A}X + \tilde{V}/\sigma)$ and $\partial\mathrm{Prox}_{h/\sigma}(X + \tilde{W}/\sigma)$ are the generalized Jacobians of the Lipschitz continuous mappings $\mathrm{Prox}_{g/\sigma}(\cdot)$ and $\mathrm{Prox}_{h/\sigma}(\cdot)$ at $\mathcal{A}X + \tilde{V}/\sigma$ and $X + \tilde{W}/\sigma$, respectively.

Now, we present our semismooth Newton-CG (Ssncg) method for solving (5.4), which can be expected to have a fast superlinear or even quadratic convergence.

Next, we give the implementable stopping criteria of the stopping criteria (A), (B1) and (B2) for Algorithm Ssncg to solve the subproblem of Algorithm Ssnal.

When we apply the Ssncg algorithm to solve (5.4), we have that

$$\begin{aligned} &X^{k+1} = \mathrm{SSNCG}(V^k, W^k, \sigma_k), \\ &Y^{k+1} = \mathrm{Prox}_{g/\sigma_k}(\mathcal{A}X^{k+1} + V^k/\sigma_k), \quad Z^{k+1} = \mathrm{Prox}_{h/\sigma_k}(X^{k+1} + W^k/\sigma_k). \end{aligned}$$

Denote $\phi_k(X) = \inf_{Y,Z}\Phi_k(X,Y,Z)$, then

$$\Phi_k(X^{k+1}, Y^{k+1}, Z^{k+1}) - \inf\Phi_k = \phi_k(X^{k+1}) - \inf\phi_k.$$

Suppose that f is strongly convex with modulus α_f, then we have that

$$\Phi_k(X^{k+1}, Y^{k+1}, Z^{k+1}) - \inf\Phi_k = \phi_k(X^{k+1}) - \inf\phi_k \le (1/(2\alpha_f))\|\nabla\phi_k(X^{k+1})\|^2$$

and

$$(\nabla\phi_k(X^{k+1}), 0, 0) \in \partial\Phi_k(X^{k+1}, Y^{k+1}, Z^{k+1}).$$

Algorithm 12: SSNCG$(\tilde{V}, \tilde{W}, \sigma)$ for (5.5)

Initialization: Given $X^0 \in \mathcal{X}$, $\mu \in (0, 1/2)$, $\tau \in (0, 1]$, and $\bar{\gamma}, \delta \in (0, 1)$.

For $j = 0, 1, 2, \ldots,$ repeat

Step 1 . Select an element $\mathcal{H}_j \in \hat{\partial}^2 \phi(X^j)$ that is defined in (5.8). Apply the conjugate gradient (CG) method to find an approximate solution $d^j \in \mathcal{X}$ to

$$\mathcal{H}_j(d) \approx -\nabla \phi(X^j) \tag{5.9}$$

such that

$$\|\mathcal{H}_j(d^j) + \nabla \phi(X^j)\| \leq \min(\bar{\eta}, \|\nabla \phi(X^j)\|^{1+\tau}).$$

Step 2 . (Line search) Set $\alpha_j = \delta^{m_j}$, where m_j is the first nonnegative integer m for which

$$\phi(X^j + \delta^m d^j) \leq \phi(X^j) + \mu \delta^m \langle \nabla \phi(X^j), d^j \rangle.$$

Step 3 . Set $X^{j+1} = X^j + \alpha_j d^j$.

Until stopping criterion based on $\|\nabla \phi(X^{j+1})\|$ is satisfied.

In this way, we can achieve the stopping criteria (A), (B1) and (B2) with the following implementable stopping criteria

$$\|\nabla \phi_k(X^{k+1})\| \leq \sqrt{\alpha_f / \sigma_k} \epsilon_k, \quad \sum_{k=0}^{\infty} \epsilon_k < +\infty, \tag{A'}$$

$$\|\nabla \phi_k(X^{k+1})\| \leq \sqrt{\alpha_f / \sigma_k} \delta_k \|(V^{k+1}, W^{k+1}) - (V^k, W^k)\|^2, \quad \sum_{k=0}^{\infty} \delta_k < +\infty, \tag{B1'}$$

$$\|\nabla \phi_k(X^{k+1})\| \leq (\delta'_k / \sigma_k) \|(V^{k+1}, W^{k+1}) - (V^k, W^k)\|. \quad 0 \leq \delta'_k \to 0, \tag{B2'}$$

Therefore, as long as $\nabla \phi_k(X^{k+1})$ is sufficiently small, the stopping criteria (A), (B1) and (B2) will be satisfied.

5.3 Proximal mappings and generalized Jacobian

In this section, we will recap some necessary ingredients for applying the SSNAL to solve the problem (5.2), which have been derived in Chapter 3 and Chapter 4.

As one can see, in order to update Y^k and Z^k in Algorithm 11, we need the computation of the proximal mappings of the function $g(\cdot)$ and $h(\cdot)$. In particular, the function $g(\cdot)$ and $h(\cdot)$ in the sparse convex clustering model (5.2) have the following form:

$$g(Y) = \gamma_1 \sum_{(i,j)\in\mathcal{E}} w_{ij}\|Y_{ij}\|_2, \quad h(X) = \gamma_2 \sum_{i=1}^{n} \|X_{\cdot,i}\|_1^2.$$

Since the function $g(\cdot)$ and $h(\cdot)$ are separable, we only need to know how to compute the proximal mapping of $\|\cdot\|_2$ and $\|\cdot\|_1^2$, respectively.

For a given function $p(\cdot) = \|\cdot\|_2$, it's well known that the proximal mapping $\mathrm{Prox}_{\mathrm{tp}}(\mathrm{x})$ has the following form:

$$\mathrm{Prox}_{\mathrm{tp}}(\mathrm{x}) = \left[1 - \frac{\mathrm{t}}{\|\mathrm{x}\|_2}\right]_+ \mathrm{x}.$$

The proximal mapping of the function $\|\cdot\|_1^2$ is given by the following proposition from Chapter 4.

Proposition 16 (Proposition 12, Proposition 13). *(a) Given $\rho > 0$, $w \in \Re^n_{++}$ and $a \in \Re^n_+$. Let $a_w \in \Re^n$ be defined as $(a_w)_i := a_i/w_i$, for $i = 1, \cdots, n$. There exists a permutation matrix Π_{a_w} such that $\Pi_{a_w} a_w$ is sorted in a non-increasing order. Denote $\tilde{a} = \Pi_{a_w} a$, $\tilde{w} = \Pi_{a_w} w$, and*

$$s_i = \sum_{j=1}^{i} \tilde{w}_j \tilde{a}_j, \quad L_i = \sum_{j=1}^{i} \tilde{w}_j^2, \quad \alpha_i = \frac{s_i}{1 + 2\rho L_i}, \quad i = 1, 2, \ldots, n.$$

Let $\bar{\alpha} = \max_{1\le i\le n} \alpha_i$. Then, $\mathrm{Prox}_{\rho\|\mathrm{w}\circ\cdot\|}(\mathrm{a})$ can be computed as: $\mathrm{Prox}_{\rho\|\mathrm{w}\circ\cdot\|}(\mathrm{a}) = (\mathrm{a} - 2\rho\bar{\alpha}\mathrm{w})^+$.

(b) For given $\rho > 0$ and $a \in \Re^n$, we have

$$\mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(a) = \mathrm{sign}(a) \circ \mathrm{Prox}_{\rho\|w\circ\cdot\|_1^2}(|a|),$$

where $\text{Prox}_{\rho\|\text{wo}\cdot\|}(|\text{a}|)$ *can be computed by the result in part (a), and* $\text{sign}(\cdot)$ *is computed coordinate-wise.*

Next, we discuss how to solve the linear system (5.9) for the sparse convex clustering model (5.2). In the Algorithm 12, we apply the conjugate gradient method to solve the linear system (5.9). To do so, we need to know how to compute the matrix vector product $\mathcal{H}_j(d)$ efficiently, where $\mathcal{H}_j \in \hat{\partial}^2\phi(X^j)$ is defined as follows:

$$\hat{\partial}^2\phi(X^j) = \{(1+\sigma)\mathcal{I} + \sigma\mathcal{A}^*\mathcal{A} - \sigma\mathcal{A}^*\mathcal{P}\mathcal{A} - \sigma\mathcal{Q}\},$$

where

$$\mathcal{P} \in \partial\text{Prox}_{g/\sigma}(\mathcal{A}X^j + V^j/\sigma), \quad \mathcal{Q} \in \partial\text{Prox}_{h/\sigma}(X^j + W^j/\sigma).$$

Thus, the key point is to compute the following two matrix vector product efficiently for a given $d \in \mathcal{X} := \Re^{p\times n}$.

$$\mathcal{A}^*\mathcal{P}\mathcal{A}(d) \quad \text{and} \quad \mathcal{Q}(d).$$

We know that $\mathcal{P}$ has a block diagonal structure, and from the analysis in Chapter 3, we know that we can compute $\mathcal{A}^*\mathcal{P}\mathcal{A}(d)$ efficiently. Now, we recap the important results.

Let $D := \mathcal{A}X^j + \sigma^{-1}V^j$. For $(i,j) \in \mathcal{E}$, define

$$\alpha_{ij} = \begin{cases} \frac{\sigma^{-1}\gamma_1 w_{ij}}{\|D^{l(i,j)}\|} & \text{if } \|D^{l(i,j)}\| > 0, \\ \infty & \text{otherwise.} \end{cases}$$

Where $l(i,j)$ is the index pair in $\mathcal{E}$ (in the lexicographic order) for the pair (i,j).

Note that for the given $D \in \Re^{p\times|\mathcal{E}|}$, the cost for computing α is $O(p|\mathcal{E}|)$ arithmetic operations. For later convenience, denote

$$\widehat{\mathcal{E}} = \{(i,j) \in \mathcal{E} \mid \alpha_{ij} < 1\}.$$

Now we choose $\mathcal{P} \in \partial\text{Prox}_{g/\sigma}(D)$ explicitly. We can take $\mathcal{P} : \Re^{p\times|\mathcal{E}|} \to \Re^{p\times|\mathcal{E}|}$ that is defined by

$$(\mathcal{P}(U))^{l(i,j)} = \begin{cases} \alpha_{ij}\frac{\langle D^{l(i,j)}, U^{l(i,j)}\rangle}{\|D^{l(i,j)}\|^2} D^{l(i,j)} + (1-\alpha_{ij})U^{l(i,j)} & \text{if } (i,j) \in \widehat{\mathcal{E}}, \\ 0 & \text{otherwise,} \end{cases}$$

for any $U \in \Re^{p\times|\mathcal{E}|}$.

Based on those notations, we can compute $\mathcal{A}^*\mathcal{P}\mathcal{A}(d)$ efficiently by using the following proposition.

Proposition 17 (Proposition 8). *Let $d \in \Re^{p\times n}$ be given.*
(a) Consider the symmetric matrix $M \in \Re^{n\times n}$ defined by $M_{ij} = 1 - \alpha_{ij}$ if $(i,j) \in \widehat{\mathcal{E}}$ and $M_{ij} = 0$ otherwise. Let $Y = [M_{ij}(d_{\cdot,i} - d_{\cdot,j})]_{(i,j)\in\mathcal{E}} = X\mathcal{M}$, where $\mathcal{M}$ is defined similarly as in (3.5) for the matrix M and d_i is the i-th column of matrix d. Then we have

$$\mathcal{A}^*(Y) = dL_M,$$

where L_M is the Laplacian matrix associated with M. The cost of computing the result $\mathcal{A}^*(Y)$ is $O(d|\widehat{\mathcal{E}}|)$ arithmetic operations.

(b) Define $\rho \in \Re^{|\mathcal{E}|}$ by

$$\rho_{l(i,j)} := \begin{cases} \frac{\alpha_{ij}}{\|D^{l(i,j)}\|^2}\langle D^{l(i,j)}, d_{\cdot,i} - d_{\cdot,j}\rangle, & \text{if } (i,j) \in \widehat{\mathcal{E}}, \\ 0, & \text{otherwise.} \end{cases}$$

For the given $D \in \Re^{p\times|\mathcal{E}|}$, the cost for computing ρ is $O(p|\widehat{\mathcal{E}}|)$ arithmetic operations. Let $W^{l(i,j)} = \rho_{l(i,j)}D^{l(i,j)}$. Then,

$$\mathcal{A}^*(W) = W\mathcal{J}^T = D\mathrm{diag}(\rho)\mathcal{J}^T.$$

(c) The computing cost for $\mathcal{A}^*\mathcal{P}\mathcal{A}(d) = \mathcal{A}^*(Y) + \mathcal{A}^*(W)$ in total is $O(p|\widehat{\mathcal{E}}|)$.

Next, we show how to compute $\mathcal{Q}(d)$ efficiently. Let $K = X^j + W^j/\sigma$, we know

$$(\mathcal{Q}(d))_{\cdot,i} = \mathcal{Q}_i d_{\cdot,i},$$

where $\mathcal{Q}_i \in \partial\mathrm{Prox}_{\frac{\gamma_2}{\sigma}\|\cdot\|_1^2}(K_{\cdot,i})$. This motivates us to focus on the efficient computation of $\mathcal{Q}_i(d_{\cdot,i})$. From the results shown in section 4.3.2 of Chapter 4, we know that we can compute $\mathcal{Q}_i(d_{\cdot,i})$ very efficiently.

Based on the discussions above, we know that we can compute the matrix vector product $\mathcal{H}_j(d)$ efficiently, which means the computational cost for each step of the

conjugate gradient is not expensive. This is the main reason why the proposed algorithm SSNAL is efficient for solving problem (5.2), especially when the condition number of $\mathcal{H}_j$ is not very bad.

5.4 Numerical experiments

In this section, we will demonstrate that the sparse convex clustering model (5.2) can do the clustering and feature selection simultaneously for high dimensional data. Also, we will show the efficiency and scalability of SSNAL for solving (5.2). We compare the performance of SSNAL with two popular algorithms, iterative least-squares algorithm (ILSA) [1] [76] and alternating direction method of multipliers (ADMM).

In order to show the scalability of SSNAL for both the number of data points n and the dimension of data p, we show two sets of numerical results on simulated datasets. First, we fix the data dimension p to be 500, and scale the number of data points n from 1000 to 15000. Then, we fix the number of data points n to be 1000 and scale p from 1000 to 10000. The results show that our proposed algorithm SSNAL scales linearly on both n and p, which is thus a practically scalable algorithm for solving (5.2). Although ILSA is popular for solving models involving the $\|\cdot\|_1^2$ regularizer [34,76], based on our experiments, it's very challenging for ILSA to solve (5.2) to moderate accuracy for large scale problems.

We write our code in MATLAB. All our experiment results shown in this chapter are obtained from a desktop having 16 cores with 32 Intel Xeon E5-2650 processors at 2.6 GHz and 64 GB memory.

In our implementation, we stop our algorithm and ADMM based on the following criteria:

$$\eta = \max\{\eta_P, \eta_D, \eta_{gap}\} < 10^{-6},$$

[1] We adopt the code provided by the authors at: http://www.makotoyamada-ml.com/localizedlasso.html

where

$$\eta_P = \frac{\|\mathcal{A}X - Y\| + \|X - Z\|}{1 + \|X\| + \|Y\| + \|Z\|}, \eta_D = \frac{\|X - A + \mathcal{A}^* V + W\|}{\|A\|}, \eta_{gap} = \frac{|primobj - dualobj|}{1 + |primobj| + |dualobj|}.$$

Here, *primobj* and *dualobj* are the objective function values of the primal problem (P) and the dual problem (D), respectively.

In our experiments, we set $\epsilon = 10^{-6}$ unless specified otherwise. We realize that the stopping criteria of ILSA is different from our mentioned criteria. Moreover, the dual feasibility is not available since it's not a primal-dual algorithm. To make a fair comparison, we stop ILSA based on the objective function value. Since (5.2) is a unconstrained optimization problem, this criteria is reasonable. More specifically, we first solve (5.2) via SSNAL with tolerance $\epsilon = 10^{-6}$, and denote the primal objective function value by P_{Ssnal}. Then, we run ILSA and stop it as soon as the computed objective function value P_{ILSA} is close enough to P_{Ssnal}, i.e.,

$$P_{\text{ILSA}} - P_{\text{Ssnal}} \leq 10^{-6} P_{\text{Ssnal}}. \tag{5.10}$$

This is a reasonably fair criteria for ILSA since the quality of solutions to an unconstrained problem could be compared directly based on the objective function values. Furthermore, we also stop ILSA if the maximum iteration number 1e4 is reached.

5.4.1 Synthetic datasets

In this section, we show the efficiency and scalability of SSNAL on synthetic data. In our experiments, we show the scalability of SSNAL for n and p separately. First, we fix $p = 500$ and test n from 1000 to 15000. Then, we fix $n = 1000$ and test p from 1000 to 10000. For each pair of (p, n), we test SSNAL, ADMM and ILSA for hyperparameters (γ_1, γ_2) in (5.2) with values $\gamma_1 \in [1 : 1 : 10]$ and $\gamma_2 \in [0.2 : 0.2 : 1]$. Furthermore, we pick the weight w_{ij} as follows

$$w_{ij} = \begin{cases} \exp(-0.5\|A_{\cdot,i} - A_{\cdot,j}\|^2), & \text{if } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise}, \end{cases}$$

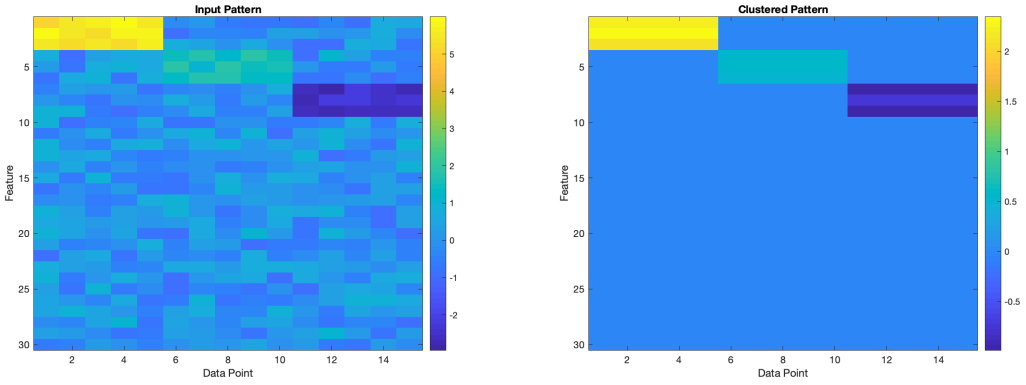


Figure 5.1: performance of (5.2) on a synthetic high dimensional dataset with uninformative features.

where $\mathcal{E} = \bigcup\{(i,j) \mid j \text{ is in } i\text{'s 10 nearest neighbors}\}$. For each pair of (p,n), we compute the average running time of each algorithm over 50 runs. We compare the three algorithms on the average running time.

Similar to [76], we generate the random data $A \in \Re^{p\times n}$ as follows:

$$A_{ij} = \begin{cases} Unif(7,8) & \text{if } (i,j) \in [1:p/10] \times [1:n/4], \\ Unif(3,4) & \text{if } (i,j) \in [1+p/10:2p/10] \times [1+n/4:n/2], \\ Unif(-1,0) & \text{if } (i,j) \in [1+2*p/10:3p/10] \times [1+n/2:3n/4], \\ Unif(-5,-4) & \text{if } (i,j) \in [1+3p/10:4p/10] \times [1+3n/4:n], \\ Unif(-1,1) & \text{Otherwise.} \end{cases}$$

Before we show the efficiency and robustness of SSNAL for solving (5.2), we show a simple example in Figure 5.1 to demonstrate the power of model (5.2) in high dimensional clustering with uninformative features.

In the example, we generate the synthetic data $A \in \Re^{30\times 15}$ with the following settings:

$$A_{ij} = \begin{cases} Unif(5,6) & \text{if } (i,j) \in [1:3] \times [1:5], \\ Unif(1,2) & \text{if } (i,j) \in [4:6] \times [6:10], \\ Unif(-2,-3) & \text{if } (i,j) \in [7:9] \times [11:15], \\ Unif(-1,1) & \text{Otherwise.} \end{cases}$$

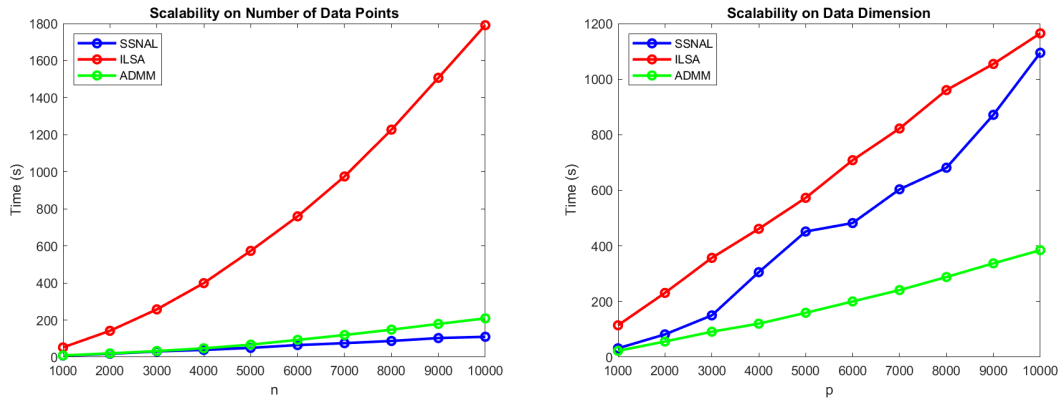


Figure 5.2: Time comparison among three algorithms on the synthetic data. Left: run time for the three algorithms with $p = 500$ and n varies from 1000 to 15000. Right: run time for the three algorithms with $n = 1000$ and p varies from 1000 to 10000.

We can see that model (5.2) could do clustering and feature selection at the same time, which is important and meaningful in clustering high dimensional data with uninformative features.

Now, we demonstrate the scalability of our proposed algorithm SSNAL for solving (5.2) in Figure 5.2. From the graph, we can see that our algorithm scales very well on the number of data points n. However, for the SSNAL, it's a little bit challenging for the algorithm to scale well on the dimension of the data p. One reason why the algorithm does not scale very well on p is because of the difficulties come from the regularizer $\sum_{(i,j)\in\mathcal{E}} w_{i,j}\|X_{\cdot,i} - X_{\cdot,j}\|_2$ in the model (5.2). As a future research direction, we will investigate on how to design an algorithm that can scale well both on p and n. In addition, we will conduct in deep analysis on how to implement SSNAL more efficient to make it scale well with respect to p.

5.4.2 COIL 20 image dataset

In this section, we will apply the sparse convex clustering model on the Columbia object image library (COIL 20) [50]. COIL 20 dataset contains images for 20 objects from 72 different angels. The size for each gray image is 32×32. Some example images in the dataset could be found in Figure 5.3.

Figure 5.3: Sample images in the COIL 20 dataset.

We will apply the sparse convex clustering model (5.2) to do the clustering on this dataset. In the experiment, we reshape each image as a vector in $\Re^{1024}$ and concatenate all the 1440 images together to form the input matrix $A \in \Re^{1024\times 1440}$ in the model (5.2). After we solve the optimization problem and obtain the optimal solution $\bar{X}$, we apply the agglomerative hierarchical clustering algorithm on $\bar{X}$ to obtain the clusters.

We compare the performance of the sparse convex clustering model with the convex clustering model (by taking $\gamma_2 = 0$ in (5.2)) and K-means algorithm based on the adjusted rand index introduced in [32].

In the experiments of the sparse convex clustering model (5.2) and convex clustering model (by taking $\gamma_2 = 0$ in (5.2)) we pick the weight w_{ij} as follows

$$w_{ij} = \begin{cases} \exp(-0.5\|A_{\cdot,i} - A_{\cdot,j}\|^2), & \text{if } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathcal{E} = \bigcup\{(i,j) \mid j \text{ is in } i\text{'s 10 nearest neighbors}\}$. We run the sparse convex clustering model (5.2) with $\gamma_1 \in \{1, 2, \ldots, 15\}$, $\gamma_2 \in \{0.1, 0.5, 1\}$ and the convex clustering model with $\gamma_1 \in \{1, 2, \ldots, 15\}$, $\gamma_2 = 0$. The best ARI for both models are shown in Table 5.1.

Table 5.1: Adjusted Rand Index (ARI) results on the COIL20 dataset. Larger ARI means better performance. p is the dimension of the dataset, n is the number of data points and K is the number of clusters.

p	n	K	Sparse Convex Clustering ($\gamma_1 = 3, \gamma_2 = 0.5$)	Convex Clustering ($\gamma_1 = 6, \gamma_2 = 0$)	K-means
1024	1440	20	**0.7352**	0.7014	0.4974

From the ARI results in Table 5.1, we can see that the convex clustering model performs better on the high dimensional dataset than the convex clustering model.

5.4.3 LIBRAS movement dataset

In this section, we apply the sparse convex clustering model on the LIBRAS movement dataset[2] from the UCI Machine Learning Repository. The dataset contains 15 classes (each class represents a kind of hand movement) and each class contains 24 observations. Each observation is a vector with 90 features.

As discussed in [73], some of the original 15 clusters indicate similar hand movements, such as curved/vertical swing and horizontal/vertical straight-line. In this section, similar to the setting in [73], we evaluate the performance on 6 out of 15

[2]https://archive.ics.uci.edu/ml/datasets/Libras+Movement

classes including vertical swing (labeled as 3), anti-clockwise arc (labeled as 4), clockwise arc (labeled as 5), horizontal straight-line (labeled as 7), horizontal wavy (labeled as 11), and vertical wavy (labeled as 12) in the original dataset. The data is visualized in Figure 5.4.

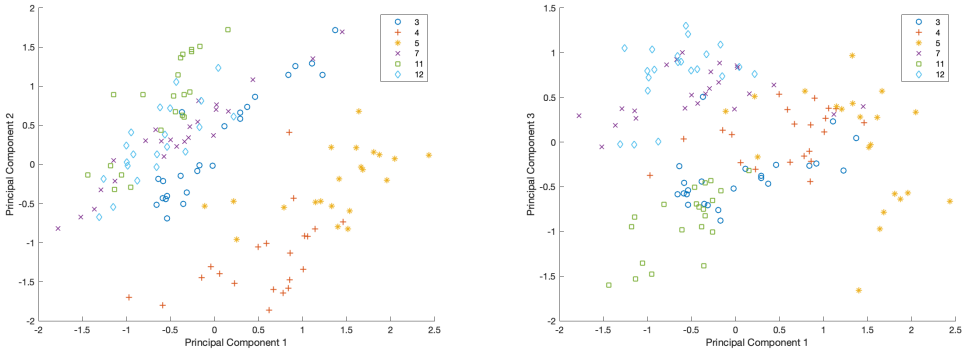


Figure 5.4: Visualization of selected data point from the LIBRAS movement dataset.

We compare the performance on this real dataset among three models: the sparse convex clustering model (5.2), the convex clustering model (by taking $\gamma_2 = 0$ in (5.2)) and the clustering model introduced in [73]. In the experiments, for the model introduced in [73] we follow the settings of the paper. However, for the sparse convex clustering model and the convex clustering model, we pick the weight w_{ij} as follows

$$w_{ij} = \begin{cases} \exp(-0.5\|A_{\cdot,i} - A_{\cdot,j}\|^2), & \text{if } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathcal{E} = \bigcup\{(i,j) \mid j \text{ is in } i\text{'s 10 nearest neighbors}\}$. We run the experiments with $\gamma_1 \in \{1, 2, \ldots, 10\}$ and $\gamma_2 \in \{0, 0.05, 0.1, 0.5, 1\}$. The comparison among the three models based on the adjusted Rand Index is shown in Table 5.2.

We visualize the clustering result of the sparse convex clustering model (5.2) in Figure 5.5.

From the results in Table 5.2, we can see that the performance of the sparse convex clustering model is comparable with the model in [73]. However, we should mention that instead of do the feature selection on the whole dataset, the sparse

Table 5.2: Adjusted Rand Index (ARI) results on the LIBRAS movement dataset. Larger ARI means better performance. p is the dimension of the dataset, n is the number of data points and K is the number of clusters.

p	n	K	Sparse Convex Clustering ($\gamma_1 = 2, \gamma_2 = 0.1$)	Convex Clustering ($\gamma_1 = 3, \gamma_2 = 0$)	Model in [73]
90	144	3	**0.455**	0.309	0.445

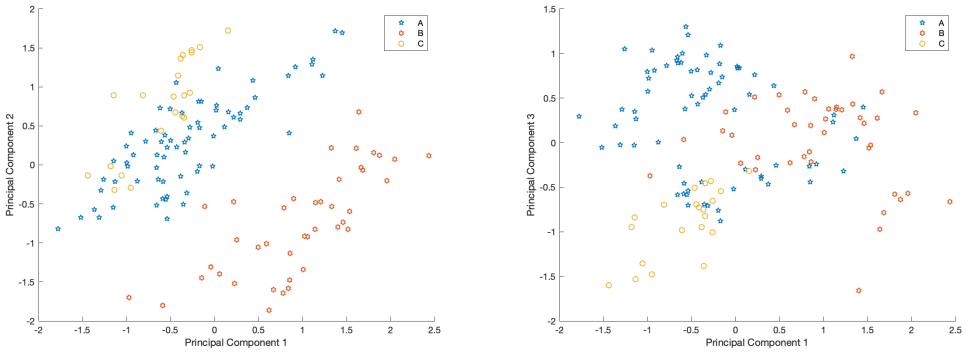


Figure 5.5: Visualization of the clustering results on the LIBRAS movement dataset by the sparse convex clustering model.

convex clustering model can achieve data point wise feature selection. This is very important since we can then explain the clustering results better by capturing the common features shared by the data points in the same cluster. We show the number of selected features by the sparse convex clustering model for each cluster in Table 5.3.

Table 5.3: Number of selected features for each cluster by different models.

Cluster	Sparse Convex Clustering	Convex Clustering	Model in [73]
1	39	90	13
2	40	90	13
3	24	90	13

5.5 Summary of the chapter

In this chapter, we design an efficient semismooth Newton-CG based augmented Lagrangian method to solve the three blocks convex composite programming which includes the sparse convex clustering model (5.2) as a special case. Also, in the numerical experiment section, we demonstrate that the sparse convex clustering model can perform the clustering and feature selection at the same time.

Chapter 6

Conclusion

In this thesis, we first analyze the theoretical guarantee for the perfect recovery of the clustering with general weighted convex clustering model. Our results improve the existing results shown in [54, 89]. We also propose a highly efficient semismooth Newton based augmented Lagrangian method (SSNAL) to solve the large scale convex clustering problem, which achieves the state-of-art performance. The numerical results shown in Chapter 3 demonstrate that the new theoretical guarantee provided in this thesis is more practically meaningful than those shown in [54, 89]. Furthermore, we also show the scalability and efficiency of our proposed numerical algorithm SSNAL for solving the convex clustering model.

In order to obtain the intra-group level sparsity for machine learning model, we focus on the exclusive lasso regularizer in Chapter 4. We propose a dual Newton based preconditioned proximal point algorithm (PPDNA) to solve the two-blocks machine learning models which include the exclusive lasso regularization term. As important ingredients, we derive the closed-form solution to the proximal mapping $\mathrm{Prox}_{\rho\|\mathrm{wo}\cdot\|_1^2}(\cdot)$ and the explicit formula for the corresponding HS Jacobian $\partial_{\mathrm{HS}}\mathrm{Prox}_{\rho\|\mathrm{wo}\cdot\|_1^2}(\cdot)$. The numerical experiment results in Chapter 4 show the superior performance of the PPDNA for solving the linear regression and logistic regression models with the exclusive lasso regularizer, comparing to some popular algorithmic frameworks, like alternating direction method of multipliers (ADMM)

and accelerated proximal gradient (APG).

In Chapter 5, we focus on the sparse convex clustering model, which could simultaneously perform clustering and feature selections. We design a SSNAL for 3-block convex composite programming problem which includes the sparse convex clustering model as a special case. In the numerical experiment section, we demonstrate the scalability and efficiency of the SSNAL for solving the convex clustering model on high dimensional datasets. More importantly, we demonstrate the power of the sparse convex clustering model on high dimensional datasets which containing a lot of uninformative features.

Appendix A

Solving the SDP Relaxation of K-means with SDPNAL+

In this appendix, we show how to solve the SDP relaxation of K-means clustering using the state-of-art SDP solver SDPNAL+ [77, 86] with a newly developed user-friendly interface [65].

SDPNAL+ is a MATLAB software package that implements an augmented Lagrangian based method to solve large scale semidefinite programming problems with bound constraints. The implementation is based on an inexact symmetric Gauss-Seidel based semi-proximal ADMM/ALM (alternating direction method of multipliers/augmented Lagrangian method) framework for the purpose of deriving simpler stopping conditions and closing the gap between the practical implementation of the algorithm and the theoretical algorithm. The basic code is written in MATLAB, but some subroutines in C language are incorporated via Mex files.

A.1 SDP with Bounded Constraints

Let $\mathcal{S}^n$ be the space of $n \times n$ real symmetric matrices and $\mathcal{S}^n_+$ be the cone of positive semidefinite matrices in $\mathcal{S}^n$. For any $X \in \mathcal{S}^n$, we may sometimes write $X \succeq 0$ to indicate that $X \in \mathcal{S}^n_+$. Let $\mathcal{P} = \{X \in \mathcal{S}^n : L \leq X \leq U\}$, where L, U are

given $n \times n$ symmetric matrices whose elements are allowed to take the values $-\infty$ and $+\infty$, respectively. Consider the semidefinite programming (SDP) problem:

$$(\mathbf{SDP}) \quad \min \Big\{ \langle C, X \rangle \mid \mathcal{A}(X) = b,\ l \le \mathcal{B}(X) \le u,\ X \in \mathcal{S}^n_+,\ X \in \mathcal{P} \Big\},$$

where $b \in \Re^m$, and $C \in \mathcal{S}^n$ are given data, $\mathcal{A} : \mathcal{S}^n \to \Re^m$ and $\mathcal{B} : \mathcal{S}^n \to \Re^p$ are two given linear maps whose adjoints are denoted as $\mathcal{A}^*$ and $\mathcal{B}^*$, respectively. The vectors l, u are given p-dimensional vectors whose elements are allowed to take the values $-\infty$ and ∞, respectively. Note that $\mathcal{P} = \mathcal{S}^n$ is allowed, in which case there are no additional bound constraints imposed on X. We assume that the $m \times m$ symmetric matrix $\mathcal{A}\mathcal{A}^*$ is invertible, i.e., $\mathcal{A}$ is surjective.

Note that (**SDP**) is equivalent to

$$(\mathrm{P}) \quad \min \Big\{ \langle C,\ X \rangle \mid \mathcal{A}(X) = b,\ \mathcal{B}(X) - s = 0,\ X \in \mathcal{S}^n_+,\ X \in \mathcal{P},\ s \in \mathcal{Q} \Big\},$$

where $\mathcal{Q} = \{ s \in \Re^p : l \le s \le u \}$. The dual of (P), ignoring the minus sign in front of the minimization, is given by

$$(\mathrm{D}) \ \min \left\{ \delta^*_{\mathcal{P}}(-Z) + \delta^*_{\mathcal{Q}}(-v) + \langle -b,\ y \rangle \ \middle|\ \begin{array}{l} \mathcal{A}^*(y) + \mathcal{B}^*(\bar{y}) + S + Z = C,\ -\bar{y} + v = 0, \\ S \in \mathcal{S}^n_+,\ Z \in \mathcal{S}^n,\ y \in \Re^m,\ \bar{y} \in \Re^p,\ v \in \Re^p \end{array} \right\},$$

where for any $Z \in \mathcal{S}^n$, $\delta^*_{\mathcal{P}}(-Z)$ is defined by

$$\delta^*_{\mathcal{P}}(-Z) = \sup\{ \langle -Z,\ W \rangle \mid W \in \mathcal{P} \}$$

and $\delta^*_{\mathcal{Q}}(\cdot)$ is defined similarly. We note that our solver is designed based on the assumption that (P) and (D) are feasible.

Actually, the solver SDPNAL+ is capable of solving the following more general problem with N blocks of variables:

$$\begin{array}{ll} \min & \sum_{j=1}^N \langle C^{(j)},\ X^{(j)} \rangle \\ \text{s.t.} & \sum_{j=1}^N \mathcal{A}^{(j)}(X^{(j)}) = b, \quad l \le \sum_{j=1}^N \mathcal{B}^{(j)}(X^{(j)}) \le u, \\ & X^{(j)} \in \mathcal{K}^{(j)},\ X^{(j)} \in \mathcal{P}^{(j)},\ j = 1, \ldots, N, \end{array} \tag{A.1}$$

where $\mathcal{A}^{(j)} : \mathcal{X}^{(j)} \to \Re^m$, and $\mathcal{B}^{(j)} : \mathcal{X}^{(j)} \to \Re^p$ are given linear maps, $\mathcal{P}^{(j)} := \{X^{(j)} \in \mathcal{X}^{(j)} \mid L^{(j)} \leq X^{(j)} \leq U^{(j)}\}$ and $L^{(j)}, U^{(j)} \in \mathcal{X}^{(j)}$ are given symmetric matrices where the elements are allowed to take the values $-\infty$ and ∞, respectively. Here $\mathcal{X}^{(j)} = \mathcal{S}^{n_j}$ ($\Re^{n_j}$), and $\mathcal{K}^{(j)} = \mathcal{X}^{(j)}$ or $\mathcal{K}^{(j)} = \mathcal{S}^{n_j}_+$ ($\Re^{n_j}_+$). For later expositions, we should note that when $\mathcal{X}^{(j)} = \mathcal{S}^{n_j}$, the linear map $\mathcal{A}^{(j)} : \mathcal{S}^{n_j} \to \Re^m$ can be expressed in the form of

$$\mathcal{A}^{(j)}(X^{(j)}) = \left[\begin{array}{c} \langle A_1^{(j)}, X^{(j)} \rangle, \ldots, \langle A_m^{(j)}, X^{(j)} \rangle \end{array} \right]^T, \tag{A.2}$$

where $A_1^{(j)}, \ldots, A_m^{(j)} \in \mathcal{S}^{n_j}$ are given constraint matrices. The corresponding adjoint $(\mathcal{A}^{(j)})^* : \Re^m \to \mathcal{S}^{n_j}$ is then given by

$$(\mathcal{A}^{(j)})^* y = \textstyle\sum_{k=1}^m y_k A_k^{(j)}.$$

SDPNAL+ is designed for solving (**SDP**) or more generally (A.1), where the maximum matrix dimension is assumed to be moderate (say less than 5000) but the number of linear constraints $m + p$ can be large (say more than a million).

The purpose of this appendix is to show how to solve the SDP relaxation of the K-means problem using SDPNAL+, so we omit the details about the algorithm behind the solver. Readers can refer to [65, 77, 86] for details.

A.2 A User-friendly Interface for SDPNAL+

SDPNAL+ is one of the best solvers for solving general SDP problems, however, some users feel that the defined data structures of the solver is not very easy to understand. Thus, we developed a basic user-friendly interface to help users use the solver more conveniently [65].

Now, we describe the design of the interface. First, we show how to use it via a

small SDP example given as follows:

$$\begin{aligned} \min \quad & \mathrm{trace}(X^{(1)}) + \mathrm{trace}(X^{(2)}) + \mathrm{sum}(X^{(3)}) \\ \text{s.t.} \quad & -X^{(1)}_{12} + 2X^{(2)}_{33} + 2X^{(3)}_{2} = 4, \\ & 2X^{(1)}_{23} + X^{(2)}_{42} - X^{(3)}_{4} = 3, \\ & 2 \le -X^{(1)}_{12} - 2X^{(2)}_{33} + 2X^{(3)}_{2} \le 7, \\ & X^{(1)} \in \mathcal{S}^{6}_{+},\ X^{(2)} \in \Re^{5\times 5},\ X^{(3)} \in \Re^{7}_{+}, \\ & 0 \le X^{(1)} \le 10E_6,\ 0 \le X^{(2)} \le 8E_5, \end{aligned} \tag{A.3}$$

where E_n denotes the $n \times n$ matrix of all ones. In the notation of (A.1), the problem (A.3) has three blocks of variables $X^{(1)}$, $X^{(2)}$, $X^{(3)}$. The first linear map $\mathcal{A}^{(1)}$ contains two constraint matrices $A^{(1)}_1, A^{(1)}_2 \in \mathcal{S}^6$ whose nonzero elements are given by

$$(A^{(1)}_1)_{12} = (A^{(1)}_1)_{21} = -0.5, \quad (A^{(1)}_2)_{23} = (A^{(1)}_2)_{32} = 1.$$

With the above constraint matrices, we get $\langle A^{(1)}_1, X^{(1)} \rangle = -X^{(1)}_{12}$ and $\langle A^{(1)}_2, X^{(1)} \rangle = 2X^{(1)}_{23}$.

The second linear map $\mathcal{A}^{(2)}$ contains two constraint matrices $A^{(2)}_1, A^{(2)}_2 \in \Re^{5\times 5}$ whose nonzero elements are given by

$$(A^{(2)}_1)_{33} = 2, \quad (A^{(2)}_2)_{42} = 1.$$

Since the third variable $X^{(3)}$ is a vector, the third linear map $\mathcal{A}^{(3)}$ is a constraint matrix $A^{(3)} \in \Re^{2\times 7}$ whose nonzero elements are given by

$$(A^{(3)})_{12} = 2, \quad (A^{(3)})_{24} = -1.$$

In a similar fashion, one can identify the matrices for the linear maps $\mathcal{B}^{(1)}, \mathcal{B}^{(2)}$, and $\mathcal{B}^{(3)}$.

The example (A.3) can be coded using our interface as follows:

```
n1 = 6; n2 = 5; n3 = 7;
model = ccp_model('Example_simple');
X1 = var_sdp(n1,n1);
```

```
X2 = var_nn(n2,n2);
X3 = var_nn(n3);
model.add_variable(X1,X2,X3);
model.minimize(trace(X1)+trace(X2)+sum(X3));
model.add_affine_constraint(-X1(1,2)+2*X2(3,3)+2*X3(2)==4);
model.add_affine_constraint(2*X1(2,3)+X2(4,2)-X3(4)==3);
model.add_affine_constraint(2<=-X1(1,2)-2*X2(3,3)+2*X3(2)<=7);
model.add_affine_constraint(0 <= X1 <= 10);
model.add_affine_constraint(X2 <= 8);
model.solve;
```

Note that although the commands

```
model.add_affine_constraint(-X1(1,2)+2*X2(3,3)+2*X3(2)==4);
model.add_affine_constraint(2*X1(2,3)+X2(4,2)-X3(4)==3);
```

are convenient to use for a small example, it may become tedious if there are many such constraints. In general, it is more economical to encode numerous such constraints by using the constraint matrices of the linear maps $\mathcal{A}^{(1)}$, $\mathcal{A}^{(2)}$, $\mathcal{A}^{(3)}$, which we illustrate below:

Listing A.1: Example (A.3) with constraints specified via linear maps as cell arrays.

```
A1 = {sparse(n1,n1); sparse(n1,n1)};
A2 = {sparse(n2,n2); sparse(n2,n2)};
A3 = sparse(2,n3);
A1{1}(1,2) = -1; A2{1}(3,3) = 2; A3(1,2) = 2;
A1{2}(2,3) = 2;  A2{2}(4,2) = 1; A3(2,4) = -1;
b = [4;3];
mymodel.add_affine_constraint(A1*X1 + A2*X2 + A3*X3 == b);
```

As the reader may have noticed, in constructing the matrix `A1{1}` corresponding to the constraint matrix $A_1^{(1)}$, we set `A1{1}(1,2) = -1` instead of `A1{1}(1,2) = -0.5; A1{1}(2,1) = -0.5`. Both ways of inputing `A1{1}` are acceptable as internally, we will symmetrize the matrix `A1{1}`.

In following subsections, we will discuss the details of the interface.

A.2.1 Creating a ccp model

Before declaring `variables`, `constraints` and setting `parameters`, we need to create a `ccp_model` class first. This is done via the command:

$$\texttt{mymodel} = \textbf{ccp\_model}(\texttt{model\_name});$$

The string `model_name` is the name of the created `ccp_model`. If no model name is specified, the default name is `` `Default' ``.

After solving the created `mymodel`, we save all the relevant information in the file `` `model_name.mat' ``. It contains two structure arrays, `input_data` and `solution`, which store all the input data and solution information, respectively.

A.2.2 Delcaring variables

Variables in SDPNAL+ can be real vectors or matrices. Currently, our interface supports four types of variables: free variables, variables in SDP cones, nonnegative variables and variables which are symmetric matrices. Next, we introduce them in details.

1. **Free variables.** One can declare a free variable $\texttt{X} \in \Re^{m\times n}$ via the command:

$$\texttt{X} = \textbf{var\_free}(\texttt{m},\texttt{n});$$

   where the parameters `m` and `n` specify the dimensions of `X`. One can also declare a column vector variable $\texttt{Y} \in \Re^{n}$ simply via the command:

$$\texttt{Y} = \textbf{var\_free}(\texttt{n});$$

2. **Variables in SDP cones.** A variable $\texttt{X} \in \mathcal{S}^n_+$ can be declared via the command:

$$\texttt{X} = \textbf{var\_sdp}(\texttt{n},\texttt{n});$$

   In this case, the variable must be a square matrix, so $\texttt{X} = \textbf{var\_sdp}(\texttt{m},\texttt{n})$ with $\texttt{m} \neq \texttt{n}$ is invalid.

3. **Variables in nonnegative orthants.** To declare a nonnegative variable `X` $\in \Re^{m\times n}_+$, one can use the command:

$$\texttt{X} = \textbf{var\_\_nn}(\texttt{m},\texttt{n});$$

We can also use `Y` = **var__nn**(`n`) to declare a vector variable `Y` $\in \Re^n_+$.

4. **Variables which are symmetric matrices.** To declare a symmetric matrix variable `X` $\in \mathcal{S}^n$, one can use the command:

$$\texttt{X} = \textbf{var\_\_symm}(\texttt{n},\texttt{n});$$

In this case, the variable must be a square matrix.

5. **Adding declared variables into a model.** Before one can start to specify the objective function and constraints in a model, the variables, say `X` and `Y`, that we have declared must be added to the `ccp_model` class `mymodel` that we have created before. This step is simply done via the command:

$$\texttt{mymodel}.\textbf{add\_\_variable}(\texttt{X},\texttt{Y});$$

Here `mymodel` is a class object and **add__variable** is a method in the class.

A.2.3 Declaring the objective function

After creating the model `mymodel`, declaring variables (say `X` and `Y`) and adding them into `mymodel`, we can proceed to specify the objective function. Declaring an objective function requires the use of the functions (methods) **minimize** or **maximize**. There must be one and only one objective function in a model specification. In general, the objective function is specified through the sum or difference of the **inprod** function (inner product of two vectors or two matrices) which must have two input arguments in the form: **inprod**(`C`,`X`) where `X` must be a declared variable, and `C` must be a constant vector or matrix which is already available in the workspace

and having the same dimension as `X`. The input `C` can also be a constant vector or matrix generated by some MATLAB built-in functions such as `speye(n,n)`.

Although we encourage users to specify an optimization problem in the standard form given in (A.1), as a user-friendly interface, we also provide some extra functions to help users to specify the objective function in a more natural way. We summarize these functions and their usages in Table A.1.

Function	Description
inprod(`C`, `X`)	The inner product of a constant vector or matrix `C` and variable `X` of the same dimension.
trace(`X`)	The trace of a square matrix variable `X`.
sum(`X`)	The sum of all elements of a vector or matrix variable `X`.
l1_norm(`X`)	The ℓ_1 norm of a variable `X`.
l1_norm($\mathcal{A}$`*X +b`)	The ℓ_1 norm of an affine expression. For the exact meaning of the expression "$\mathcal{A}$`*X`", the reader can refer to (A.5).

Table A.1: Supported functions for specifying the objective function in a model.

For the class `mymodel` created above, we can see that the objective function of (A.3) is specified via the command:

```
mymodel.minimize(trace(X1) + trace(X2) + sum(X3));
```

A.2.4 Adding affine constraints into the model

Affine constraints can be specified and added into `mymodel` after the relevant variables have been declared. This is done via the function (method) `add_affine_constraint`. The following constraint types are supported in the interface:

- Equality constraints `==`
- Less-or-equal inequality constraints `<=`
- Greater-or-equal inequality constraints `>=`

where the expressions on both the left and right-hand sides of the operands must be affine expressions. Strict inequalities `<` and `>` are **not** accepted. Inequality and equality constraints are applied in an elementwise fashion, matching the behavior of MATLAB itself. For instance, if `U` and `X` are $m \times n$ matrices, then `X <= U` is interpreted as mn (scalar) inequalities `X(i,j) <= U(i,j)` for all $i = 1, \ldots, m$, $j = 1, \ldots, n$. When one side is a scalar and the other side is a variable, that value is replicated; for instance, `X >= 0` is interpreted as `X(i,j) >= 0` for all $i = 1, \ldots, m$, $j = 1, \ldots, n$.

In general, affine constraints have the following form

$$\mathcal{A}_1 * \mathtt{X_1} + \mathcal{A}_2 * \mathtt{X_2} + \cdots + \mathcal{A}_k * \mathtt{X_k} \texttt{ <= (>= or ==) } b, \tag{A.4}$$

where $\mathtt{X_1}, \mathtt{X_2}, \ldots, \mathtt{X_k}$ are declared variables, b is a constant matrix or vector, and $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ are linear maps whose descriptions will be given shortly.

Next, we illustrate how to add affine constraints into the model object `mymodel` in detail.

General affine constraints

In this section, we show users how to initialize the linear maps $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_k$ in (A.4).

- If $\mathcal{A}_i = a_i$, is a scalar, then $a_i * \mathtt{X_i}$ has the same dimension as the variable $\mathtt{X_i}$.
- If $\mathtt{X_i}$ is an n-dimensional vector, then $\mathcal{A}_i$ must be a $p \times n$ constant matrix, and $\mathcal{A}_i * \mathtt{X_i}$ is in $\Re^p$.
- If $\mathtt{X_i}$ is an $m \times n$ ($n > 1$) matrix, then $\mathcal{A}_i * \mathtt{X_i}$ is interpreted as a linear map such that

$$\mathcal{A}_i * \mathtt{X_i} = \begin{bmatrix} \langle A_1^{(i)}, \mathtt{X_i} \rangle \\ \vdots \\ \langle A_p^{(i)}, \mathtt{X_i} \rangle \end{bmatrix} \in \Re^p, \tag{A.5}$$

where $A_1^{(i)}, \ldots, A_p^{(i)}$ are given $m \times n$ constant matrices. In this case, $\mathcal{A}_i$ is a $p \times 1$ constant cell array such that

$$\mathcal{A}_i\{j\} = A_j^{(i)}, \quad j = 1, \ldots, p.$$

Coordinate-wise affine constraints

Although users can model coordinate-wise affine constraints in the general form given in (A.4), we allow users to declare them in a more direct way as follows:

$$a_1 * \mathtt{X_1}(i_1, j_1) + a_2 * \mathtt{X_2}(i_2, j_2) + \cdots + a_k * \mathtt{X_k}(i_k, j_k) \ \texttt{<=} \ (\texttt{>=} \ \text{or} \ \texttt{==}) \ b, \tag{A.6}$$

where $a_1, a_2, \ldots, a_k, b$ are scalars and $\mathtt{X_1}, \mathtt{X_2}, \ldots, \mathtt{X_k}$ are declared variables. The index pairs (i_1, j_1), (i_2, j_2), $\ldots$, (i_k, j_k) extract the corresponding elements in the variables. From Listing **??**, we can see how a constraint of the form (A.6) is added, i.e.,

$$\mathtt{mymodel}.\text{add\_affine\_constraint}(2 * \mathtt{X_1}(2,3) + \mathtt{X_2}(4,2) - \mathtt{X_3}(4) \ \texttt{==} \ 3)$$

Our interface also allows users to handle multiple index pairs. For example, if we have a declared variable $\mathtt{X} \in \Re^{m \times n}$ and two index arrays

$$I = [i_1, i_2, \ldots, i_k], \quad J = [j_1, j_2, \ldots, j_k],$$

where $\max\{i_1, i_2, \ldots, i_k\} \leq m$ and $\max\{j_1, j_2, \ldots, j_k\} \leq n$, then $\mathtt{X}(I, J)$ is interpreted as

$$\mathtt{X}(I, J) = \begin{bmatrix} \mathtt{X}(i_1, j_1) \\ \mathtt{X}(i_2, j_2) \\ \vdots \\ \mathtt{X}(i_k, j_k) \end{bmatrix} \in \Re^k.$$

An example of such a usage can be found in Listing **??**.

Element-wise multiplication

In our interface, we also support element-wise multiplication (.∗) between a declared variable `X` and a constant matrix A with the same dimension. Suppose

$$\mathtt{X} = \begin{bmatrix} \mathtt{X_{11}} & \cdots & \mathtt{X_{1n}} \\ \vdots & \ddots & \vdots \\ \mathtt{X_{m1}} & \cdots & \mathtt{X_{mn}} \end{bmatrix}, \quad A = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix}.$$

Then $A.*\mathtt{X}$ is interpreted as

$$A.*\mathtt{X} = \begin{bmatrix} A_{11}*\mathtt{X_{11}} & \cdots & A_{1n}*\mathtt{X_{1n}} \\ \vdots & \ddots & \vdots \\ A_{11}*\mathtt{X_{m1}} & \cdots & A_{mn}*\mathtt{X_{mn}} \end{bmatrix}.$$

Specifying affine constraints using predefined maps

For convenience, we also provide some predefined maps to help users to specify constraints in a more direct way. We summarize these maps and their usages in Table A.2.

Chained constraints

In our interface, one can add chained inequalities into the created `ccp_model mymodel`. In general, chained affine constraints have the form

$$\mathtt{L} <= \mathcal{A}_1 * \mathtt{X}_1 + \mathcal{A}_2 * \mathtt{X}_2 + \cdots + \mathcal{A}_k * \mathtt{X_k} <= \mathtt{U},$$

where `L` and `U` are scalars or constant matrices with having the same dimensions as the affine expression in the middle. As an example, one can add bound constraints for a declared variable `X` via the command:

```
mymodel.add_affine_constraint(L <= X <= U);
```

It is important to note that in chained inequality constraints, the affine expression in the middle should only contain declared variables but not constants.

Function	Description	Dimension
inprod(C, X)	The inner product of a constant vector or matrix C and a variable X of the same dimension.	1×1
trace(X)	The trace of a square matrix variable X.	1×1
sum(X)	The sum of all elements of a vector or matrix variable X.	1×1
l1_norm(X)	The ℓ_1 norm of a variable X.	1×1
l1_norm($\mathcal{A}$*X + b)	The ℓ_1 norm of an affine expression.	1×1
map_diag(X)	Extract the main diagonal of an $n \times n$ matrix variable X.	$n \times 1$
map_svec(X)	For an $n \times n$ symmetric variable X, it returns the corresponding symmetric vectorization of X, as defined in (**??**).	$\frac{n(n+1)}{2} \times 1$
map_vec(X)	For a $m \times n$ matrix variable X, it returns the vectorization of X.	$mn \times 1$

Table A.2: Supported predefined maps.

A.2.5 Adding positive semidefinite constraints into the model

Positive semidefinite constraints can be added into a previously created object `mymodel` using the function (method) `add_psd_constraint`. Such a constraint is valid only for a declared symmetric variable or positive semidefinite variable. In general, a positive semidefinite constraint has the form

$$a_1 * \mathtt{X_1} + a_2 * \mathtt{X_2} + \cdots + a_k * \mathtt{X_k} \ \succeq \ \mathtt{G}, \tag{A.7}$$

where $a_1, a_2, \ldots, a_k$ are scalars, and $\mathtt{X_1}$, $\mathtt{X_2}$, $\ldots$, $\mathtt{X_k}$ are declared variables in symmetric matrix spaces or PSD cones, and G is a constant symmetric matrix. Note that one can also have the version "$\preceq$" in (A.7). We can add (A.7) into `mymodel` as follows:

`mymodel.add_psd_constraint`($a_1 * \mathtt{X_1} + \cdots + a_k * \mathtt{X_k}$ `>=` G)

Specially,

- For a variable `X` $\in \mathcal{S}^n$, one can use `mymodel.add_psd_constraint(X>=0)` to specify the constraint `X` $\succeq 0$ or `X` $\in \mathcal{S}^n_+$.
- For a variable `X` $\in \mathcal{S}^n$ and a constant matrix `G` $\in \mathcal{S}^n$. One can use
  `mymodel.add_psd_constraint(X >= G)`
  and
  `mymodel.add_psd_constraint(X <= G)`
  to specify the constraint `X` $\succeq$ `G` and `X` $\preceq$ `G`, respectively.

Similar to affine constraints, one can also use chained positive semidefinite constraints together. For example, for a variable `X` $\in \mathcal{S}^n$ and two constant matrices `G1`, `G2` $\in \mathcal{S}^n$ (`G1` $\preceq$ `G2`), one can specify `G1` $\preceq$ `X` $\preceq$ `G2` as

```
mymodel.add_psd_constraint(G1 <= X <= G2);
```

A.2.6 Setting parameters for SDPNAL+

As described previously, there are mainly nine parameters in the parameter structure array `OPTIONS`. To allow users to set these parameters freely, we provide the function (method) `setparameter` for such a purpose. Now, we describe the usage of `setparameter` in details.

Assume that we have created a `ccp_model` class called `mymodel`. Since `setparameter` is a method in the `ccp_model` class, so the usage of `setparameter` is simply

```
mymodel.setparameter(`para_name',value)
```

In Table A.3, we summarize the parameters which can be set in `setparameter`. Note that users can set more than one parameters at a time. For example, one can use

```
mymodel.setparameter(`tol', 1e-4, `maxiter', 2000);
```

to set the parameters `tol = 1e-4` and `maxiter = 2000`.

Parameter Name	Usage	Default Value
tol	mymodel.setparameter(`tol', value)	1e-6
maxiter	mymodel.setparameter(`maxiter', value)	20000
maxtime	mymodel.setparameter(`maxtime', value)	10000
tolADM	mymodel.setparameter(`tolADM', value)	1e-4
maxiterADM	mymodel.setparameter(`maxiterADM', value)	200
printlevel	mymodel.setparameter(`printlevel', value)	1
stopoption	mymodel.setparameter(`stopoption', value)	1
AATsolve.method	mymodel.setparameter(`AATsolve.method', value)	'direct'
BBTsolve.method	mymodel.setparameter(`BBTsolve.method', value)	'iterative'

Table A.3: Usage of `setparameter`.

A.2.7 Solving a model and extracting solutions

After creating and initializing the class `mymodel`, one can call the method **solve** to solve the model as follow:

```
mymodel.solve
```

After solving the SDP problem, one can extract the optimal solutions using the function **get_value**. For example, if `X1` is a declared variable, then one can extract the optimal value of `X1` by setting

```
get_value(X1)
```

Note that the input of the function `get_value` should be a declared variable.

A.2.8 Further remarks on the interface

Here we give some remarks to help users to input an SDP problem into our interface more efficiently.

- If a variable must satisfy a conic constraint, it would be more efficient to specify the conic constraint when declaring the variable rather than declaring the variable and imposing the constraint separately. For example, it is better to use `X = var_nn(m,n)` to indicate that the variable `X` $\in \Re^{m\times n}$ must be in the cone $\Re^{m\times n}_+$ rather than separately declaring `X = var_free(m,n)` followed by setting

  ```
  mymodel.add_affine_constraint(X >= 0);
  ```

  Similarly, if a square matrix variable `Y` $\in \mathcal{S}^n$ must satisfy the conic constraint that `Y` $\in \mathcal{S}^n_+$, then it is better to declare it as `Y = var_sdp(n,n)` rather than separately declaring `Y = var_free(n,n)` followed by setting

  ```
  mymodel.add_psd_constraint(Y >= 0);
  ```

  The latter option is not preferred because we have to introduce extra constraints.

- When there is a large number of affine constraints, specifying them using a loop in MATLAB is generally time consuming. To make the task more efficient, if possible, always try to model the problem using our predefined functions

A.3 Solving the SDP Relaxation of K-means

In this section, we will solve the SDP Relaxation of K-means with the introduced interface of the solver SDPNAL+. For a given collection of n data points $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\}$, we define $D \in \Re^{n\times n}$ by $D_{ij} = \mathbf{a}_i^T\mathbf{a}_j$, then the SDP relaxation of the K-means clustering model for a given integer $k > 0$ has the following form:

$$\min\Big\{\mathrm{Tr}(DX) \mid \mathrm{Tr}(X) = k,\ X\mathbf{e} = \mathbf{e},\ X \geq 0,\ X \in \mathcal{S}^+_n\Big\}, \tag{A.8}$$

where $X \geq 0$ means that all the elements in X are nonnegative, $\mathcal{S}^n_+$ is the cone of $n\times n$ symmetric and positive semidefinite matrices, and $\mathbf{e} \in \Re^n$ is the column vector of all ones.

Now, we show how to solve (A.8) using the software SDPNAL+ with the interface introduced in section A.2. We take the famous two half moon dataset for example.

Listing A.2: Sample Code: SDP Kmeans.

```
clear all;
load('Half_Moon_Balance_100.mat');
D = X'*X;
K = 2;
n = length(W);
e = ones(n,1);
model = ccp_model('Clustering_HM_100');
X = var_sdp(n,n);
model.add_variable(X);
model.maximize(inprod(D,X));
model.add_affine_constraint(trace(X) == K);
%% add the constraints X*e = e
for k=1:n
ek = zeros(n,1); ek(k)=1;
Ak = e*ek';
model.add_affine_constraint(inprod(Ak,X) == 1);
end
model.add_affine_constraint(X >= 0);
model.solve;
Xsol = get_value(X);
```

In the experiments, we generate the two half moons data X with 100 or 1000 data points. We set $k = 2$ since there are two clusters. We visualize the heat maps of the input matrix D and the solution of (A.8) in Figure A.1.

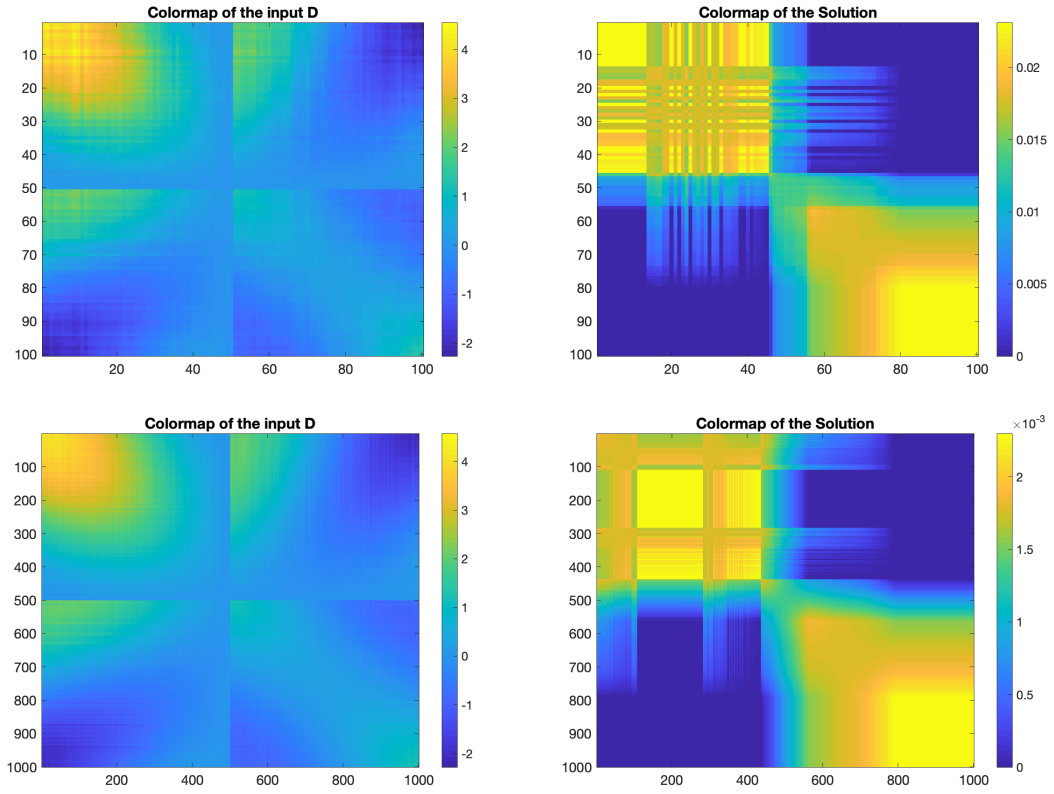


Figure A.1: Visualization of the Kmeans SDP model on the two half moon data set with $n = 100$ and $n = 1000$. First row: the results for $n = 100$. Second row: the results for $n = 1000$.

Bibliography

[1] *Yahoo Finance: https://finance.yahoo.com.*

[2] W. N. Anderson and T. D. Morley, *Eigenvalues of the Laplacian of a graph*, Linear and Multilinear Algebra, 18 (1985), pp. 141–145.

[3] F. J. Aragón Artacho and M. H. Geoffroy, *Characterization of metric regularity of subdifferentials*, Journal of Convex Analysis, 15 (2008), pp. 365–380.

[4] D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[5] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward, *Relax, no need to round: Integrality of clustering formulations*, in Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ACM, 2015, pp. 191–200.

[6] H. H. Bauschke, J. M. Borwein, and W. Li, *Strong conical hull intersection property, bounded linear regularity, Jameson's property (G), and*

error bounds in convex optimization, Mathematical Programming, 86 (1999), pp. 135–160.

[7] H. H. Bauschke, P. L. Combettes, et al., *Convex analysis and monotone operator theory in Hilbert spaces*, vol. 408, Springer, 2011.

[8] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 183–202.

[9] S. R. Becker, E. J. Candès, and M. C. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Mathematical programming computation, 3 (2011), p. 165.

[10] J. F. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*, Springer, 2000.

[11] F. Campbell and G. I. Allen, *Within group variable selection through the exclusive lasso*, Electronic J. of Statistics, 11 (2017), pp. 4220–4257.

[12] E. Candes and T. Tao, *The dantzig selector: Statistical estimation when p is much larger than n*, The Annals of Statistics, 35 (2007), pp. 2313–2351.

[13] L. Chen, D. Sun, and K.-C. Toh, *An efficient inexact symmetric Gauss–Seidel based majorized admm for high-dimensional convex composite conic programming*, Mathematical Programming, 161 (2017), pp. 237–270.

[14] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM Review, 43 (2001), pp. 129–159.

[15] X. Chen, X.-T. Yuan, Q. Chen, S. Yan, and T.-S. Chua, *Multi-label visual classification with label exclusive context*, in 2011 International Conference on Computer Vision, IEEE, 2011, pp. 834–841.

[16] E. Chi and K. Lange, *Splitting methods for convex clustering*, J. Computational and Graphical Statistics, 24 (2015), pp. 994–1013.

[17] E. C. Chi, B. R. Gaines, W. W. Sun, H. Zhou, and J. Yang, *Provable convex co-clustering of tensors*, arXiv preprint arXiv:1803.06518, (2018).

[18] F. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley and Sons, New York, 1983.

[19] Y. Cui and D. Sun, *A complete characterization of the robust isolated calmness of nuclear norm regularized convex optimization problems*, Journal of Computational Mathematics, 36 (2018), pp. 441–458.

[20] Y. Cui, D. Sun, and K.-C. Toh, *On the asymptotic superlinear convergence of the augmented Lagrangian method for semidefinite programming with multiple solutions*, arXiv preprint arXiv:1610.00875, (2016).

[21] Y. Dong, *An extension of luque's growth condition*, Applied Mathematics Letters, 22 (2009), pp. 1390–1393.

[22] A. L. Dontchev and R. T. Rockafellar, *Implicit functions and solution mappings*, Springer Monographs in Mathematics, (2009).

[23] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Mathematical Programming, 55 (1992), pp. 293–318.

[24] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least angle regression*, The Annals of Statistics, 32 (2004), pp. 407–499.

[25] F. Facchinei and J.-S. Pang, *Finite-dimensional variational inequalities and complementarity problems*, Springer Science & Business Media, 2007.

[26] M. Fazel, T. K. Pong, D. Sun, and P. Tseng, *Hankel matrix rank minimization with applications to system identification and realization*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 946–977.

[27] M. Fukushima and H. Mine, *A generalized proximal point algorithm for certain non-convex minimization problems*, International Journal of Systems Science, 12 (1981), pp. 989–1000.

[28] R. Glowinski and A. Marroco, *Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires*, Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique, 9 (1975), pp. 41–76.

[29] J. Han and D. F. Sun, *Newton and quasi-Newton methods for normal maps with polyhedral sets*, J. Optimization Theory and Applications, 94 (1997), pp. 659–676.

[30] J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. Nguyen, *Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data*, Appl. Math. Optim., 11 (1984), pp. 43–56.

[31] T. D. Hocking, A. Joulin, F. Bach, and J.-P. Vert, *Clusterpath: an algorithm for clustering using convex fusion penalties*, in 28th International Conference on Machine Learning, 2011.

[32] L. Hubert and P. Arabie, *Comparing partitions*, Journal of classification, 2 (1985), pp. 193–218.

[33] T. Jiang, S. Vavasis, and C. W. Zhai, *Recovery of a mixture of gaussians by sum-of-norms clustering*, arXiv preprint arXiv:1902.07137, (2019).

[34] D. Kong, R. Fujimaki, J. Liu, F. Nie, and C. Ding, *Exclusive feature learning on arbitrary structures via $l_{1,2}$-norm*, in Advances in Neural Information Processing Systems, 2014, pp. 1655–1663.

[35] M. Kowalski, *Sparse regression using mixed norms*, Applied and Computational Harmonic Analysis, 27 (2009), pp. 303–324.

[36] M. Kowalski and B. Torrésani, *Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients*, Signal, image and video processing, 3 (2009), pp. 251–264.

[37] B. Kummer, *Newton's method for non-differentiable functions*, Advances in Mathematical Optimization, 45 (1988), pp. 114–125.

[38] X. Li, D. Sun, and K.-C. Toh, *An asymptotically superlinearly convergent semismooth Newton augmented Lagrangian method for linear programming*, arXiv preprint arXiv:1903.09546, (2019).

[39] X. Li, D. F. Sun, and K.-C. Toh, *A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems*, SIAM J. Optimization, 28 (2018), pp. 433–458.

[40] ——, *On efficiently solving the subproblems of a level-set method for fused lasso problems*, SIAM J. Optimization, 28 (2018), pp. 1842–1866.

[41] F. Lindsten, H. Ohlsson, and L. Ljung, *Clustering using-of-norms regularization: With application to particle filter output computation*, in Statistical Signal Processing Workshop (SSP), IEEE, 2011, pp. 201–204.

[42] Y.-J. Liu, D. Sun, and K.-C. Toh, *An implementable proximal point algorithmic framework for nuclear norm minimization*, Mathematical programming, 133 (2012), pp. 399–436.

[43] S. Lloyd, *Least squares quantization in pcm*, IEEE transactions on information theory, 28 (1982), pp. 129–137.

[44] Z.-Q. Luo and P. Tseng, *On the linear convergence of descent methods for convex essentially smooth minimization*, SIAM Journal on Control and Optimization, 30 (1992), pp. 408–425.

[45] F. J. Luque, *Asymptotic convergence analysis of the proximal point algorithm*, SIAM Journal on Control and Optimization, 22 (1984), pp. 277–293.

[46] R. Mifflin, *Semismooth and semiconvex functions in constrained optimization*, SIAM Journal on Control and Optimization, 15 (1977), pp. 959–972.

[47] D. G. Mixon, S. Villar, and R. Ward, *Clustering subgaussian mixtures by semidefinite programming*, arXiv preprint arXiv:1602.06612, (2016).

[48] ——, *Clustering subgaussian mixtures by semidefinite programming*, Information and Inference: A Journal of the IMA, 6 (2017), pp. 389–415.

[49] J.-J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bulletin de la Société mathématique de France, 93 (1965), pp. 273–299.

[50] S. A. Nene, S. K. Nayar, H. Murase, et al., *Columbia object image library (coil-20)*.

[51] Y. NESTEROV, *A method for solving the convex programming problem with convergence rate* $O(1/k^2)$, in Dokl. Akad. Nauk SSSR, vol. 269, 1983, pp. 543–547.

[52] Y. Nesterov, *Gradient methods for minimizing composite functions*, Mathematical Programming, 140 (2013), pp. 125–161.

[53] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[54] A. Panahi, D. Dubhashi, F. Johansson, and C. Bhattacharyya, *Clustering by sum of norms: Stochastic incremental algorithm, convergence and cluster recovery*, in 34th International Conference on Machine Learning, vol. 70, PMLR, 2017, pp. 2769–2777.

[55] K. Pelckmans, J. De Brabanter, J. Suykens, and B. De Moor, *Convex clustering shrinkage*, in PASCAL Workshop on Statistics and Optimization of Clustering Workshop, 2005.

[56] J. Peng and Y. Wei, *Approximating k-means-type clustering via semidefinite programming*, SIAM journal on optimization, 18 (2007), pp. 186–205.

[57] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Mathematical Programming, 58 (1993), pp. 353–367.

[58] P. Radchenko and G. Mukherjee, *Convex clustering via l_1 fusion penalization*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79 (2017), pp. 1527–1546.

[59] M. Rezaei and P. Fränti, *Set-matching methods for external cluster validity*, IEEE Trans. on Knowledge and Data Engineering, 28 (2016), pp. 2173–2186.

[60] S. M. Robinson, *Some continuity properties of polyhedral multifunctions*, in Mathematical Programming at Oberwolfach, Springer, 1981, pp. 206–214.

[61] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[62] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Mathematics of Operations Research, 1 (1976), pp. 97–116.

[63] ——, *Monotone operators and the proximal point algorithm*, SIAM journal on control and optimization, 14 (1976), pp. 877–898.

[64] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*, vol. 317, Springer Science & Business Media, 2009.

[65] D. Sun, K.-C. Toh, Y. Yuan, and X.-Y. Zhao, *SDPNAL+: A matlab software for semidefinite programming with bound constraints (version 1.0)*, Optimization Methods and Software, (2019), pp. 1–29.

[66] J. Sun, *On monotropic piecewise quadratic programming*, PhD thesis, University of Washington, 1986.

[67] K. M. Tan and D. Witten, *Statistical properties of convex clustering*, Electronic J. Statistics, 9 (2015), p. 2324.

[68] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. of the Royal Statistical Society: Series B, (1996), pp. 267–288.

[69] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, *Sparsity and smoothness via the fused lasso*, J. of the Royal Statistical Society: Series B, 67 (2005), pp. 91–108.

[70] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, Pacific Journal of Optimization, (2010).

[71] P. Tseng, *Approximation accuracy, gradient methods, and error bound for structured convex optimization*, Mathematical Programming, 125 (2010), pp. 263–295.

[72] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Mathematical Programming, 117 (2009), pp. 387–423.

[73] B. Wang, Y. Zhang, W. W. Sun, and Y. Fang, *Sparse convex clustering*, Journal of Computational and Graphical Statistics, 27 (2018), pp. 393–403.

[74] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM J. Imaging Sciences, 1 (2008), pp. 248–272.

[75] G. Xu, Y. Xia, and H. Ji, *Weighted total variation based convex clustering*, arXiv preprint arXiv:1808.09144, (2018).

[76] M. Yamada, T. Koh, T. Iwata, J. Shawe-Taylor, and S. Kaski, *Localized lasso for High-Dimensional Regression*, in Artificial Intelligence and Statistics, 2017, pp. 325–333.

[77] L. Yang, D. Sun, and K.-C. Toh, *SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints*, Mathematical Programming Computation, 7 (2015), pp. 331–366.

[78] L. Yuan, J. Liu, and J. Ye, *Efficient methods for overlapping group lasso*, in Advances in Neural Information Processing Systems, 2011, pp. 352–360.

[79] M. Yuan and Y. Lin, *Model selection and estimation in regression with grouped variables*, J. of the Royal Statistical Society: Series B, 68 (2006), pp. 49–67.

[80] X.-T. Yuan and S. Yan, *A finite Newton algorithm for non-degenerate piecewise linear systems*, in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 841–854.

[81] Y. Yuan, D. F. Sun, and K.-C. Toh, *An efficient semismooth Newton based algorithm for convex clustering*, in Proceedings of the 35th International Conference on Machine Learning, 2018, pp. 5718–5726.

[82] T. Zhang, B. Ghanem, S. Liu, C. Xu, and N. Ahuja, *Robust visual tracking via exclusive context modeling*, IEEE Transactions on Cybernetics, 46 (2016), pp. 51–63.

[83] Y. Zhang, N. Zhang, D. Sun, and K.-C. Toh, *An efficient Hessian based algorithm for solving large-scale sparse group lasso problems*, Mathematical Programming, (2019).

[84] Y. Zhang, N. Zhang, D. F. Sun, and K.-C. Toh, *An efficient Hessian based algorithm for solving large-scale sparse group lasso problems*, Mathematical Programming, (2018).

[85] P. Zhao and B. Yu, *On model selection consistency of lasso*, J. of Machine Learning Research, 7 (2006), pp. 2541–2563.

[86] X.-Y. Zhao, D. Sun, and K.-C. Toh, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM Journal on Optimization, 20 (2010), pp. 1737–1765.

[87] Y. Zhou, R. Jin, and S. C.-H. Hoi, *Exclusive lasso for multi-task feature selection*, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 988–995.

[88] Z. Zhou and A. M.-C. So, *A unified approach to error bounds for structured convex optimization problems*, Mathematical Programming, 165 (2017), pp. 689–728.

[89] C. Zhu, H. Xu, C. Leng, and S. Yan, *Convex optimization procedure for clustering: Theoretical revisit*, in Advances in Neural Information Processing Systems 27, 2014, pp. 1619–1627.

[90] H. Zou, *The adaptive lasso and its oracle properties*, J. of the American Statistical Association, 101 (2006), pp. 1418–1429.

SIMULTANEOUS MODEL FOR CLUSTERING AND INTRA-GROUP FEATURE SELECTION

YUAN YANCHENG

NATIONAL UNIVERSITY OF SINGAPORE

2019

Simultaneous Model for Clustering and Intra-group Feature Selection **Yuan Yancheng** **2019**