An Efficient Semismooth Newton Based Algorithm for Convex Clustering

Yancheng Yuan [1] **Defeng Sun** [2] **Kim-Chuan Toh** [3]

Abstract

Clustering is a fundamental problem in unsupervised learning. Popular methods like K-means, may suffer from instability as they are prone to get stuck in its local minima. Recently, the sum-of-norms (SON) model (also known as clustering path), which is a convex relaxation of hierarchical clustering model, has been proposed in (Lindsten et al., 2011) and (Hocking et al., 2011). Although numerical algorithms like alternating direction method of multipliers (ADMM) and alternating minimization algorithm (AMA) have been proposed to solve convex clustering model (Chi & Lange, 2015), it is known to be very challenging to solve large-scale problems. In this paper, we propose a semismooth Newton based augmented Lagrangian method for large-scale convex clustering problems. Extensive numerical experiments on both simulated and real data demonstrate that our algorithm is highly efficient and robust for solving large-scale problems. Moreover, the numerical results also show the superior performance and scalability of our algorithm comparing to existing first-order methods.

1. Introduction

Clustering is one of the most fundamental problems in unsupervised learning. Traditional clustering models such as K-means clustering, hierarchical clustering may suffer from instability because of the non-convexity of the model and the difficulties in finding a global optimal solution. The clustering results are generally highly dependent on the initialization 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 real applications, k-means is typically tried with different cluster numbers and the user will then decide on a suitable value based on his judgment on which computed result agrees best with his domain knowledge or experience. Obviously, such a process could make the clustering results subjective.

In order to overcome the above issues, a new clustering model has been proposed recently (Lindsten et al., 2011; Hocking et al., 2011) which is generally more robust comparing 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. Convex clustering model for these n observations solves the following convex optimization problem:

$$\min_{X\in\mathbb{R}^{d\times n}} \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_i\|^2 + \gamma \sum_{i<j} \|\mathbf{x}_i - \mathbf{x}_j\|_q, \qquad (1)$$

where $\|\cdot\|$ denotes the 2-norm, and $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. The q-norm $\|\cdot\|_q$ above with $q \geq 1$ ensures the convexity of the model. The popular choices of q are $q \in \{1, 2, \infty\}$. In this paper, we focus on $q = 2$. After solving (1) and obtaining the optimal solution X^*, $\mathbf{a}_i$ and $\mathbf{a}_j$ belong 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 input observations $\mathbf{a}_i$ and $\mathbf{a}_j$ belong to same cluster, then their corresponding centroids $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ should be the same. The first term in (1) is the fidelity term. The second term is the regularization term to penalize the difference between different centroids and 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) 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) for a sequence of positive values of γ.

To handle cluster recovery for large-scale data sets, various researchers, e.g., (Pelckmans et al., 2005; Lindsten et al., 2011; Hocking et al., 2011; Zhu et al., 2014; Tan & Wit-

[*]Equal contribution [1]Department of Mathematics, National University of Singapore [2]Department of Applied Mathematics, Hong Kong Polytechnic University [3]Department of Mathematics, National University of Singapore. Correspondence to: Yancheng Yuan <yuanyancheng@u.nus.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

ten, 2015; Panahi et al., 2017) have suggested the following weighted convex clustering model modified from (1)

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

where the edge set $\mathcal{E} = \cup_{i=1}^{n}\{(i,j) \mid j \text{ is } i\text{'s } k\text{-nearest neighbors}, i < j \le n\}$ and $w_{ij} = \exp(-\phi\|\mathbf{a}_i - \mathbf{a}_j\|^2)$ for $(i,j) \in \mathcal{E}$. We can regard the original convex clustering model (1) as a special case if we take $k = n$ and $\phi = 0$.

The advantages just mentioned and the success of the convex model (1) 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). However, the first theoretical result on cluster recovery established in (Zhu et al., 2014) is valid for only two clusters. It showed that the model (1) can recover the two clusters perfectly if the data points are drawn from two cubes and the distance between these two cubes are large enough. Tan & Witten (2015) analyzed the statistical properties of (1). Recently, Panahi et al. (2017) provided theoretical results for the general k clusters case under relatively mild sufficient conditions, thus laying the theoretical foundation for convex clustering. However, the conditions provided in the theoretical analysis are usually not checkable before we find the right clusters and the perfect value of γ is unknown. In practice, we need to try a sequence of values of γ to generate a clustering path.

The challenges for the convex model (1) to obtain meaningful cluster recovery is then to solve it efficiently for a range of values of γ. Lindsten et al. (2011) used the off-the-shelf solver, CVX, to generate the solution path. However, Hocking et al. (2011) realized that CVX is competitive only for small-size problems and it does not scale well when the number of data points increases. Thus the paper introduced three algorithms based on the subgradient methods for different regularizers. Recently, some new algorithms have been proposed to solve this problem. Chi & Lange (2015) adapted the ADMM and AMA to solve (1). However, based on our experiments, both algorithms may still encounter scalability issues, albeit less severe than CVX. Furthermore, the efficiency of these two algorithms are sensitive to the parameter value γ. This is not favorable since we need to solve (1) with γ in a relative large range to generate the clustering path. In (Panahi et al., 2017), the authors proposed a stochastic splitting algorithm for (1) in an attempt to resolve the aforementioned scalability issues. Although this stochastic approach scales well with the problem scale (n in (1)), the convergence rate shown in (Panahi et al., 2017) is rather weak in that it requires at least $l \ge n^4/\varepsilon$ iterations to generate a solution X^l such that $\|X^l - X^*\|^2 \le \varepsilon$ is satisfied with high probability. (Here and below, $\|\cdot\|$ is also used to denote the Frobenius norm of a matrix.) Moreover, because the error estimate is given in the sense of high probability, it is difficult to design appropriate stopping condition for the algorithm in practice.

As the reader 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 this paper, we design and analyze a deterministic second-order algorithm, the semismooth Newton based augmented Lagrangian method, to solve the convex clustering model. The algorithm is not only proven to be theoretically efficient but it is also demonstrated to be practically highly efficient and robust.

2. Related work

In addition to the papers (Pelckmans et al., 2005; Lindsten et al., 2011; Hocking et al., 2011; Zhu et al., 2014; Tan & Witten, 2015; Panahi et al., 2017) on the convex models (1) and (2), 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 via semidefinite programming (SDP) (Peng & Wei, 2007; Awasthi et al., 2015; Mixon et al., 2016). The computational efficiency of SDP based relaxations highly depends on the efficiency of the available SDP solvers. While recent progress (Zhao et al., 2010; Yang et al., 2015; Sun et al., 2017) 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 (Chi & Lange, 2015) has implicitly demonstrated that it is generally cheaper to solve the model (2) instead of the SDP relaxation model. Together with the motivation from the paper (Li et al., 2018) that proposed a highly efficient semismooth Newton augmented Lagrangian method (ALM) to solve Lasso and fused Lasso problems, we are thus inspired to adapt the ALM framework for solving the convex clustering model (2) in this paper.

3. A semismooth Newton-CG augmented Lagrangian method with fast linear convergence

In this section, we introduce a fast convergent ALM for solving the convex clustering model (2). Before that, we introduce some preliminaries and notations.

3.1. Preliminaries and notation

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

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

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 matrices $\mathcal{J}, \bar{\mathcal{J}} \in \mathbb{R}^{n\times|\mathcal{E}|}$ as

$$\mathcal{J}_k^{l(i,j)} = \begin{cases} 1 & if\ k=i, \\ 0 & otherwise; \end{cases} \quad \bar{\mathcal{J}}_k^{l(i,j)} = \begin{cases} 1 & if\ k=j, \\ 0 & otherwise, \end{cases} \tag{3}$$

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

Proposition 1. *With matrices $J, \mathcal{J}, \bar{\mathcal{J}}$ defined above, we have the following results*

$$\mathcal{J}\mathcal{J}^T + \bar{\mathcal{J}}\bar{\mathcal{J}}^T = \mathrm{diag}(J\mathbf{e}),\ \ \mathcal{J}\bar{\mathcal{J}}^T + \bar{\mathcal{J}}\mathcal{J}^T = J,$$

where $\mathbf{e} \in \mathbb{R}^n$ is the column vector of all ones.

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

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

and

$$\mathcal{B}^*(Z) = Z(\mathcal{J}^T - \bar{\mathcal{J}}^T).$$

Thus, by Proposition (1), we have

$$\mathcal{B}^*(\mathcal{B}(X)) = X(\mathcal{J}\mathcal{J}^T - \mathcal{J}\bar{\mathcal{J}}^T - \bar{\mathcal{J}}\mathcal{J}^T + \bar{\mathcal{J}}\bar{\mathcal{J}}^T) = XL_J, \tag{4}$$

where $L_J = diag(J\mathbf{e}) - J \in \mathbb{R}^{n\times n}$ is the unweighted Laplacian matrix associated with the graph G.

For a convex function $p : \mathbb{R}^d \to (-\infty, +\infty]$, which is proper and closed, the proximal mapping $\mathrm{Prox}_{tp}(x)$ for p at any $x \in \mathbb{R}^d$ with $t > 0$ is defined by

$$\mathrm{Prox}_{tp}(x) = \arg\min_{u\in\mathcal{X}}\{tp(u) + \frac{1}{2}\|u - x\|^2\}. \tag{5}$$

It is well known that proximal mappings are important for designing optimization algorithms and they have been well studied. The proximal mappings for many common used functions have closed form formulas. Here, we summarize those that are related to this paper in Table 1. Note that $\mathcal{P}_C$

Table 1. Proximal maps for selected functions

$p(\cdot)$	$\mathrm{Prox}_{tp}(\mathbf{x})$	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
$\|\cdot\|_\infty$	$\mathbf{x} - \mathcal{P}_{t\mathcal{S}}(\mathbf{x})$	$\mathcal{S}$ is the unit ℓ_1-ball

denotes the projection onto a given closed convex set C.

In this paper, we will often make use of the following Moreau identity

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

where $t > 0$ and p^* is the conjugate function of p.

3.2. Duality and optimality conditions

In this section, we will derive the dual problem of (2) and the KKT conditions. First, we write (2) 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 $p(U) = \gamma\sum_{(i,j)\in\mathcal{E}} w_{ij}\|U^{l(i,j)}\|$. Here $U^{l(i,j)}$ denotes the $l(i,j)$-th column of $U \in \mathbb{R}^{d\times|\mathcal{E}|}$.

The dual problem 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 $\Omega = \{Z \in \mathbb{R}^{d\times|\mathcal{E}|} \mid \|Z^{l(i,j)}\| \le \gamma w_{ij}, (i,j) \in \mathcal{E}\}$.

Now, denote by l the Lagrangian function for (P):

$$l(X,U;Z) = \frac{1}{2}\|X - A\|^2 + p(U) + \langle Z, \mathcal{B}(X) - U\rangle. \tag{6}$$

Furthermore, given $\sigma > 0$, 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 KKT conditions for (P) and (D) are given by

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

3.3. A semismooth Newton-CG augmented Lagrangian method for solving (2)

In this section, we will design an inexact ALM for solving the primal problem (P) but it will also solve (D) as a byproduct. Since a semismooth Newton-CG method will be used to solve the subproblems involved in the method, we call our algorithm a semismooth Newton-CG augmented Lagrangian method (SSNAL in short). The algorithm for solving (P) is shown in Algorithm 1. To ensure the convergence of the inexact ALM in **Algorithm** 1, we need the following stopping criterion for solving the subproblem (7) in each iteration:

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

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

3.4. Solving the subproblem (7)

The inexact ALM is a well studied and efficiently algorithmic framework for convex composite optimization problems.

Algorithm 1 SSNAL for (P)

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

repeat

Step 1. Compute

$$(X^{k+1}, U^{k+1}) \approx \arg\min\{\Phi_k(X, U) = \mathcal{L}_{\sigma_k}(X, U; Z^k)\} \quad (7)$$

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

until Stopping criterion is satisfied.

The key challenge in making the inexact ALM efficient numerically is solving the subproblem (7) in each iteration efficiently to the required accuracy. In this paper, we will design a semismooth Newton-CG method to solve (7). We will establish its quadratic convergence and derive sophisticated numerical techniques to solve the associated semismooth Newton equations very efficiently by exploiting the underlying second-order structured sparsity in the subproblem.
For a given σ and $\tilde{Z}$, the subproblem (7) in each iteration has the following form:

$$\min_{X\in\mathbb{R}^{d\times n}, U\in\mathbb{R}^{d\times|\mathcal{E}|}} \Phi(X, U) := \mathcal{L}_\sigma(X, U; \tilde{Z}). \quad (8)$$

Since $\Phi(X, U)$ is a strongly convex function, the level set $\{(X, U)|\Phi(X, U) \leq \alpha\}$ is a closed and bounded convex set for any $\alpha \in \mathbb{R}$. Hence (8) admits a unique optimal solution which we denote as $(\bar{X}, \bar{U})$. Now, for any X, denote

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

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

$$\bar{X} = \arg\min \phi(X), \quad \bar{U} = \text{Prox}_{p/\sigma}(\mathcal{B}(\bar{X}) + \sigma^{-1}\tilde{Z}).$$

Since $\phi(\cdot)$ is strongly convex and continuously differentiable on $\mathbb{R}^{d\times n}$ with

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

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

$$\nabla\phi(X) = 0. \quad (9)$$

It is well known that Newton's method is the best method to solve nonlinear equations if it can be implemented efficiently. However, Newton's method requires the smoothness of $\nabla\phi(X)$ which is not the case in our problem. This motivates us to develop a semismooth Newton method to solve the nonsmooth equation (9). Before we present our semismooth Newton method, we introduce the following definition of semismoothness.

Definition 1. *(Semismoothness). Let $F : \mathcal{O} \subseteq \mathcal{X} \to \mathcal{Y}$ be a locally Lipschitz continuous function on an open set $\mathcal{O}$. F is said to be semismooth at $x \in \mathcal{O}$ if F is directionally differentiable at x and 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 semismooth at x if F is semismooth at x 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 $\mathcal{O}$ if it is semismooth (respectively, strongly semismooth) everywhere in $\mathcal{O}$.

Lemma 1. *For any $t > 0$, the proximal mapping $Prox_{t\|\cdot\|_2}$ is strongly semismooth.*

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

$$\begin{aligned}\hat{\partial}^2\phi(X) &:= \{\mathcal{I} + \sigma\mathcal{B}^*\mathcal{V}\mathcal{B}|\mathcal{V} \in \partial\text{Prox}_{\sigma p^*}(\tilde{Z} + \sigma\mathcal{B}(X))\\ &= \{\mathcal{I} + \sigma\mathcal{B}^*(\mathcal{I} - \mathcal{P})\mathcal{B}|\mathcal{P} \in \partial\text{Prox}_{p/\sigma}(\tfrac{1}{\sigma}\tilde{Z} + \mathcal{B}(X))\}, \quad (10)\end{aligned}$$

where $\partial\text{Prox}_{\sigma p^*}(\tilde{Z} + \sigma\mathcal{B}(X))$ and $\partial\text{Prox}_{p/\sigma}(\frac{1}{\sigma}\tilde{Z} + \mathcal{B}(X))$ are the Clarke subdifferentials of the Lipschitz continuous mappings $\text{Prox}_{\sigma p^*}$ and $\text{Prox}_{p/\sigma}(\cdot)$ at $\tilde{Z} + \sigma\mathcal{B}(X)$ and $\frac{1}{\sigma}\tilde{Z} + \mathcal{B}(X)$, respectively. Note that from (Clarke, 1983) [p.75] and (Hiriart-Urruty et al., 1984) [Example 2.5], we have

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

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 $\mathcal{I} - \mathcal{P} = \mathcal{V} \in \partial\text{Prox}_{\sigma p^*}(\cdot)$ is symmetric and positive semdefinite, the elements in $\hat{\partial}^2\phi(X)$ are positive definite, and this guarantees that (11) in Algorithm 2 is well defined. Now, we can present our semismooth Newton-CG (SSNCG) method for solving (9) and we could expect to get a fast superlinear or even quadratic convergence.

3.5. Using the conjugate gradient method to solve (11)

In this section, we will discuss how to solve the very large symmetric positive definite linear system (11) to compute the Newton direction efficiently. As the matrix representation of the coefficient linear operator $\mathcal{V}_j$ in (11) is expensive to compute and factorize, we will adopt the conjugate gradient (CG) method to solve it. The computational cost for CG

Algorithm 2 SSNCG for (9)

Initialization: Given $X^0 \in \mathbb{R}^{d\times n}$, $\mu \in (0, 1/2)$, $\tau \in (0, 1]$, and $\bar{\eta}, \delta \in (0, 1)$. For $j = 0, 1, \ldots$

repeat

Step 1. Pick an element $\mathcal{V}_j$ in $\hat{\partial}^2\phi(X^j)$ that is defined in (10). Apply the conjugate gradient (CG) method to find an approximate solution $d^j \in \mathbb{R}^{d\times n}$ to

$$\mathcal{V}_j(d) \approx -\nabla\phi(X^j), \tag{11}$$

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

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

until Stopping criterion based on $\|\nabla\phi(X^{j+1})\|$ is satisfied.

is highly dependent on the cost for computing the matrix-vector product $\mathcal{V}_j(\tilde{d})$ 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

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

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

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

To compute $\mathcal{V}_j(X) = X + \sigma\mathcal{B}^*\mathcal{B}(X) - \sigma\mathcal{B}^*\mathcal{P}\mathcal{B}(X)$ 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 2. *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 $Y = [M_{ij}(\mathbf{x}_i - \mathbf{x}_j)]_{(i,j)\in\mathcal{E}} = X(\mathcal{M} - \bar{\mathcal{M}})$, where $\mathcal{M}$ and $\bar{\mathcal{M}}$ are defined similarly as in* (3) *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.

Now, define $\rho \in \mathbb{R}^{|\mathcal{E}|}$ by

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

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,

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

Thus, the computing cost for $\mathcal{B}^*\mathcal{P}\mathcal{B}(X) = \mathcal{B}^*(Y) + \mathcal{B}^*(W)$ in total is $O(d|\widehat{\mathcal{E}}|)$, where Y is given in Proposition 2.

Observe that $\alpha_{ij} < 1$ means that j is in i's nearest k neighbors but does not belong to the same cluster as i, so $|\widehat{\mathcal{E}}|$ should be much smaller than $|\mathcal{E}|$. On the other hand, for $\alpha_{ij} \ge 1$, it means that points i and j are in the same cluster. From this analysis, we can expect most of the columns of the matrix $\mathcal{P}(\mathcal{B}(X))$ to be zero. We call such a property inherited from the generalized Hessian of $\phi(\cdot)$ at X as the **second-order sparsity.** It is because of this important property that we are able to compute $\mathcal{B}^*\mathcal{P}\mathcal{B}(X)$ at a very low cost.

3.6. 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 1. *Let $\{(X^k, U^k, Z^k)\}$ be the sequence generated by Algorithm 1 with stopping criterion* (A)*. Then the sequence $\{X^k\}$ is bounded and converges to the unique optimal solution of* (P)*, and $\|\mathcal{B}(X^k) - U^k\|$ converges to* 0*. In addition, $\{Z^k\}$ is also bounded and converges to the optimal solution $Z^* \in \Omega$ of* (D)*.*

The above convergence theorem can be obtained from (Rockafellar, 1976a;b) without much difficulties. Next, we state the convergence property for the semismooth Newton algorithm SSNCG used to solve the subproblems in Algorithm 1.

Theorem 2. *Assume that $Prox_{tp}(\cdot)$ is strongly semismooth on $int(dom(p))$. Let the sequence $\{X^j\}$ be generated by Algorithm* SSNCG*. Then $\{X^j\}$ converges to the unique solution $\bar{X}$ of the problem in* (9) *and*

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

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

The proof of this theorem could be found in the supplementary material. Note that by Lemma 1, the strong semismoothness assumption holds true for our model (2).

3.7. Generating an initial point

In our implementation, we use the following inexact alternating direction method of multipliers (IADMM) developed in (Chen et al., 2017) to generate an initial point to warm-start SSNAL[1]. Note that in Step 1, X^{k+1} is a computed

Algorithm 3 IADMM for (P)

Initialization: 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

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

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

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

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

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

until the stopping criterion is satisfied.

solution for the following large linear system of equations:

$$(I_n + \sigma\mathcal{B}^*\mathcal{B})X = R^k \quad \underset{(4)}{\Longleftrightarrow} \quad (I_n + \sigma L_J)X^T = (R^k)^T.$$

To compute X^{k+1}, we apply the conjugate gradient method to solve the above linear system.

4. Numerical experiments

In this section, we 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 (Chi & Lange, 2015). 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 research on scalability and performance of (2) on unbalanced datasets is limited. The problem sizes of the instances tested in (Chi & Lange, 2015) and other related papers are only several hundreds ($n \le 500$ in (Chi & Lange, 2015), $n \le 600$ in (Panahi et al., 2017)), which are not large enough to clearly demonstrate the scalability of the algorithms. In this paper, we will show numerical results for n up to 20000. Also, we will analyze the sensitivity of the computational efficiency of SSNAL and AMA, with respect to different choices of the parameters in (2), such as k (number of nearest neighbors) and γ.

Our attention in this paper focuses on solving (2) with $q = 2$ since the rotational invariance of the ℓ_2 norm makes it a robust choice in practice. Also, this case is more challenging than $q = 1$ or $q = \infty$.[2] As the results reported in (Chi & Lange, 2015) have been regarded as the benchmark for the convex clustering model (2), we will compare our algorithm with the open source software CVXCLUSTR[3] in (Chi & Lange, 2015), 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:

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

where

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

and $\epsilon > 0$ is a given tolerance. In our experiments, we set $\epsilon = 10^{-6}$ unless specified otherwise. Since the numerical results reported in (Chi & Lange, 2015) 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 $\max\{\eta_P, \eta_D\} \le \epsilon$ in ADMM. To make a fair comparison, we first solve (2) 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.,

$$P_{AMA} - P_{Ssnal} \le 10^{-6}P_{Ssnal}. \tag{12}$$

We note that since (2) 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 (2). 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.

[1] With the global convergence result stated in Theorem 1, the performance of SSNAL does not sensitively depend on the initial points, but it is still helpful if we can choose a good one.

[2] Our algorithm can be generalized to solve (2) with $q = 1$ and $q = \infty$ easily.

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

4.1. Simulated data

In this section, we show the performance of our algorithm SSNAL on two simulated datasets: Two Half-Moon and Unbalanced Gaussian (Rezaei & Fränti, 2016). We compare our SSNAL with the AMA in (Chi & Lange, 2015) on different problem scales. The numerical results in Table 2 show the superior performance of SSNAL. We also visualize some selected recovery results for Two Half-moon and Unbalanced Gaussian in Figure 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 (12) within 100000 iterations when n is large. In the experiments, we choose $k = 10$, $\phi = 0.5$ (for the weights w_{ij}) and $\gamma \in [0.2 : 0.2 : 10]$ 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 (12). We report the average time for solving each problem (50 in total) in Table 2. Observe that our SSNAL can be more than 50 times faster than AMA.

Table 2. Computation time (in seconds) comparison on Two Half-Moon. (— means that the maximum iteration is reached)

n	200	500	1000	2000	5000	10000
AMA	0.41	4.43	28.27	78.36	—	—
SSNAL	0.11	0.15	0.51	1.63	7.69	20.96

UNBALANCED GAUSSIAN DATA

Next, we show the performance of SSNAL and AMA on the Unbalanced Gaussian data $\mathbb{R}^2$ (Rezaei & Fränti, 2016). We can see from Figure 1 that the convex clustering model (2) can recover the cluster assignments perfectly with well chosen parameters. In this experiment, we solve (2) 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 $[0, 1]$. In experiments, we find that AMA has difficulties in reaching the stopping criterion (12). We summarize some selected results in Table 3, wherein we report the computation times and iterations for both AMA and SSNAL. Since the main cost for SSNAL is in the inner iterations of SSNCG, so we show the iterations of SSNCG for comparison. The computed primal objective values and other results are reported in the supplementary material.

Table 3. Numerical results on Unbalanced Gaussian data. The parameters used are $k = 10$, $\phi = 0.5$.

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**	**0.67**	**0.66**	**0.86**
Iter_{AMA}	100000	97560	97333	100000	100000
$\text{Iter}_{\text{SSNCG}}$	**23**	**21**	**24**	**24**	**27**

4.2. 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).[4] 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 MNIST (we selected a subset because AMA cannot handle the whole dataset), we first apply the preprocessing method described in (Mixon et al., 2016). Then we apply the model (2) on the preprocessed data. We summarize the comparison results between SSNAL and AMA on real datasets in Table 4.

Table 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	1000	79.48	**1.54**
MNIST	10	10000	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	**52.71**

4.3. Sensitivity with different γ

In order to generate a clustering path for a given dataset, we need to solve (2) 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 2, we show the comparison between SSNAL and AMA on both the Two Half-Moon and MNIST datasets with $\gamma \in [0.2 : 0.2 : 10]$.

4.4. 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 (9). However, if we look inside the algorithm, we

[4] See the references for the data sources.

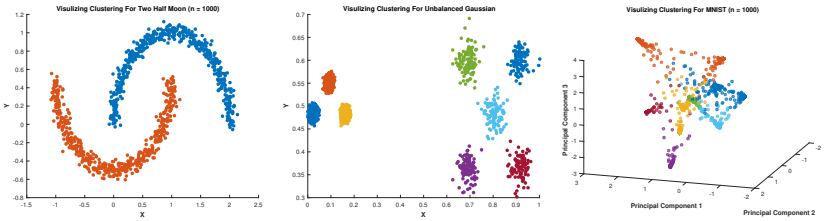

Figure 1. Selected recovery results by model (2) with ℓ_2 norm. Left: recovery result for a Two Half-Moon data with $n = 1000$, $k = 20$, $\gamma = 5$. Middle: recovery result for an Unbalanced Gaussian data with $n = 6500$, $k = 10$, $\gamma = 1$. Right: recovery result for a subset of MNIST with $n = 1000$, $\gamma = 1$.

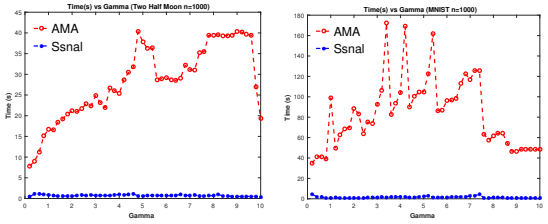

Figure 2. Time comparison between SSNAL and AMA on both Two Half-Moon and MNIST data with $\gamma \in [0.2 : 0.2 : 10]$.

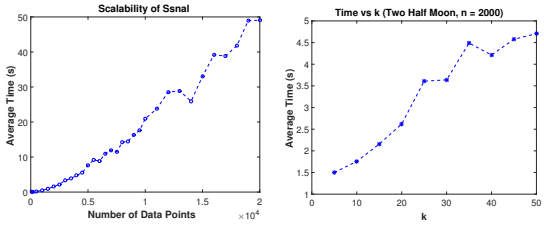

Figure 3. Numerical results to demonstrate the scalability of our proposed algorithm SSNAL with respect to n and k.

can see that the key step is to use the CG method to solve (11) efficiently to get the Newton direction. According to our complexity analysis in Section 3.5, the computational cost for one step of CG update is $O(d|\mathcal{E}| + d|\widehat{\mathcal{E}}|)$. By the specific choice of $\mathcal{E}$, $|\widehat{\mathcal{E}}|$ should only grow slowly with n. This low computational cost for the matrix-vector product in our Newton-CG method is the key point behind why our algorithm can be scalable and efficient. The numerical results shown in this section also strongly support this argument.

In our experiments, we apply our algorithm on Half-Moon data with n ranging from 100 to 20000. Comparing to the numerical results reported in (Chi & Lange, 2015) and (Panahi et al., 2017) with $n \leq 500$ and $n \leq 600$, respectively, our results have convincingly demonstrated the scalability of our SSNAL.

In our experiments, we set $\phi = 0.5$, $k = 10$ (the number of nearest neighbors). Then we solve (2) with $\gamma \in [0.4 : 0.4 : 20]$. After generating the clustering path, we compute the average time for solving a single instance of (2) for each problem scale. Another factor related to the scalability is the number of neighbors k used in $\mathcal{E}$ in (2). 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 (2) for each k. We summarize our numerical results in Figure 3. We can observe that the computation time grows almost linearly with n and k.

5. Conclusion

In this paper, we proposed a highly efficient and scalable semismooth Newton based augmented Lagrangian method to solve the convex clustering model (2). To the best of our knowledge, this is the first optimization algorithm for convex clustering model which uses the second-order generalized Hessian information. Extensive numerical results shown in the paper have demonstrated the scalability and superior performance of our proposed algorithm SSNAL comparing to the state-of-the-art software CVXCLUSTR. The convergence results for our algorithm are also provided. In our future work, we plan to design a distributed and parallel version of SSNAL with the aim to handle huge scale data sets. From the modeling perspective, we will also work on generalizing our algorithm to handle kernel based convex clustering models.

References

Awasthi, P.l, Bandeira, Afonso S., Charikar, M., Krishnaswamy, R., Villar, S., and Ward, R. Relax, no need to round: integrality of clustering formulations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pp. 191–200. ACM, 2015.

Chen, L., Sun, D.F., and Toh, K.C. An efficient inexact symmetric Gauss–Seidel based majorized ADMM for high-dimensional convex composite conic programming. *Mathematical Programming*, 161:237–270, 2017.

Chi, E.C. and Lange, K. Splitting methods for convex clustering. *J. Computational and Graphical Statistics*, 24 (4):994–1013, 2015.

Clarke, F. *Optimization and Nonsmooth Analysis*. John Wiley and Sons, New York, 1983.

Fisher. Fisher iris dataset, 1936. UCI Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/iris.

Georghiades, A.S., Belhumeur, P.N., and Kriegman, D.J. The yale face database b, 2001a. URL `http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html`.

Georghiades, Athinodoros S., Belhumeur, Peter N., and Kriegman, David J. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001b.

Hiriart-Urruty, J.-B., Strodiot, J.-J., and Nguyen, V.H. Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data. *Appl. Math. Optim.*, 11:43–56, 1984.

Hocking, T. D., Joulin, A., Bach, F., and Vert, J.-P. Clusterpath an algorithm for clustering using convex fusion penalties. In *28th International Conference on Machine Learning*, 2011.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Mnist dataset, 1998. http://yann.lecun.com/exdb/mnist/.

Li, X.D., Sun, D.F., and Toh, K.C. A highly efficient semismooth Newton augmented Lagrangian method for solving lasso problems. *SIAM J. Optimization*, 28:433–458, 2018.

Lichman, M. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Lindsten, F., Ohlsson, H., and Ljung, L. Clustering using sum-of-norms regularization: With application to particle filter output computation. In *Statistical Signal Processing Workshop (SSP)*, pp. 201–204. IEEE, 2011.

Mixon, D. G., Villar, S., and Ward, R. Clustering subgaussian mixtures by semidefinite programming. *arXiv preprint arXiv:1602.06612*, 2016.

Panahi, A., Dubhashi, D., Johansson, F.D, and Bhattacharyya, C. Clustering by sum of norms: Stochastic incremental algorithm, convergence and cluster recovery. In *34th International Conference on Machine Learning*, volume 70, pp. 2769–2777. PMLR, 2017.

Pelckmans, K., De Brabanter, J., Suykens, J., and De Moor, B. Convex clustering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*, 2005.

Peng, Jiming and Wei, Yu. Approximating k-means-type clustering via semidefinite programming. *SIAM journal on optimization*, 18(1):186–205, 2007.

Rezaei, M. and Fränti, P. Set-matching methods for external cluster validity. *IEEE Trans. on Knowledge and Data Engineering*, 28(8):2173–2186, 2016.

Rockafellar, R.T. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2):97–116, 1976a.

Rockafellar, R.T. Monotone operators and the proximal point algorithm. *SIAM J. Control and Optimization*, 14 (5):877–898, 1976b.

Sun, Defeng, Toh, Kim-Chuan, Yuan, Yancheng, and Zhao, Xin-Yuan. SDPNAL+: a Matlab software for semidefinite programming with bound constraints (version 1.0). *arXiv preprint arXiv:1710.10604*, 2017.

Tan, K. M. and Witten, D. Statistical properties of convex clustering. *Electronic J. Statistics*, 9(2):2324, 2015.

Yang, Liuqin, Sun, Defeng, and Toh, Kim-Chuan. SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

Zhao, Xin-Yuan, Sun, Defeng, and Toh, Kim-Chuan. A Newton-CG augmented Lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20(4): 1737–1765, 2010.

Zhu, C., Xu, H., Leng, C.L., and Yan, S.C. Convex optimization procedure for clustering: Theoretical revisit. In *Advances in Neural Information Processing Systems 27*, pp. 1619–1627, 2014.