K-means: A Semidefinite Programming's Perspective

Hui Wei
hw1666

May 14, 2018

Abstract

K-means is one of the most important unsupervised leaning algorithms, which assigns N points into k clusters on the basis of minimal sum-of-squared distances. However, this problem is NP-hard. In this report, I show a SDP relaxation method for K-means and conclude that this relaxation is tight under the stochastic ball model. Then examine this method also works under a more generalized model, subgaussian mixture model. Finally, I introduce several algorithms for solving this relaxation under two models and quickly giving a certifier for the results, according to [2] and [3].

1 Introduction

Optimization problems always come from data science and machine learning. In machine learning, what we usually do is to transfer a model to a object function and apply optimization algorithms to solve them. Unfortunately, for those problems, they are usually NP-hard, which is a worst case statement for computer scientists.

K-means is one of the well-known clustering algorithms, which requests partitions of $\{x_i\}_{i=1}^N$ to minimize the dissimilarity objective function. A popular option for the objective function of K-means is as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^{k} \sum_{i \in A_t} \left\| x_i - \frac{1}{|A_t|} \sum_{j \in A_t} x_j \right\|_2^2 \\ \text{subject to} \quad & A_1 \cup A_2 \ldots \cup A_k = \{1, \ldots, N\} \end{aligned} \tag{1}$$

In (1), we define

$$c_t = \frac{1}{|A_t|} \sum_{j \in A_t} x_j \tag{2}$$

as the centroid for every cluster A_t.

There is a very popular heuristic algorithm called Lloyd's algorithm, also called k-means algorithm, for solving this problem. It changes centroid of every cluster with the change of each cluster. It is described as follows [1].

Lloyd's Algorithm
(1) For the domain which contains all the data points, randomly generate k cluster centroids.
(2) For each data points, compute the distance between it and every centroid.
(3) Assign every data points to the closest centroid.
(4) Recompute the centroid according equation (2) using the current cluster memberships.
(5) If the objective function satisfies the criterion, then the algorithm stops; otherwise, repeat above steps.

Despite its simplicity and popularity, according to [1] and [2], there are two shortcomings about Lloyd's algorithm: 1.like other non-convex optimization problems, this algorithm is very sensitive to the initial options for the starting points of k centroids, therefore, it is easy to converge to a local minimum rather than the global minimum point as we expected. 2.from what mentioned above, the origin Lloyd's algorithm does not show that how close the result approaches the global optimal. In addition, the original problem (1) is NP-hard. It can be shown that the exact solution of it will take $O(n^{kd+1})$. Therefore, Lloyd's problem can only be used in a relatively small dataset.

Due to the drawbacks of this popular algorithm, many scientists are working on how to relax the constraint of the original problem in order to give an algorithm which can produce the result which is close to the global minimum. In particular, Peng and Wei [1] used 0-1 SDP to relax this problem and invented an algorithm called spectral clustering based on the SDP relaxation. Iguchi [2] proved that Peng and Wei's SDP relaxation is actually tight for K-means problem under the special model called stochastic ball model. In the same paper [2], they also introduced a fast algorithm to test the optimality of the k-means solution giving a certifier in quasilinear time. Furthermore, Mixon and Villar [3] introduced a relax-and-round algorithm which is able to be applied to a more general model known as subgaussian mixture model. In addition, they also explained that the new algorithm is the denoising process for the method of Lloyd's algorithm or the spectral clustering in [1].

My work. I studied all the theoretical knowledge in the referenced papers and summarized their methods. In addition, I implemented algorithms in the papers and did some experiments to test the results mentioned by the authors. I added a lot of my understandings about how to get the SDP relaxed k-means problem in this report, which is not obvious in the original papers.

Organization of this report. In Section 2, I summarized the theoretical background for algorithms appeared in the papers. Moreover, I will emphasize what interests me, including how to get the SDP relaxation from the original problem, and how to explain the well-rounded result for subgaussian mixture models. I skipped all complex proofs in papers about lemmas and theorems for simplicity. In Section 3, I will show my experiment results about algorithms mentioned in the papers.

2 Theory

2.1 SDP and Dual Program of K-means

In this section, I will introduce how to transfer the original K-means problem, which is NP-hard, to a solvable one using SDP relaxation. At the same time, for the completeness, this report will give the Dual Program of the SDP without proof. In the series of reference papers, the symbols used in Peng and Wei [1] are different from those used in other papers [2,3,4]. Since in the later sections, it is more convenient to utilize the latter ones, I will show how to change the symbols in [1] to get the equivalent form in other papers.

In [1], Peng and Wei introduced a programming model named 0-1 semidefinite programming. Then they derived the SDP form of K-means, and claimed it is a 0-1 semidefinite programming model. As in [1], I will introduce 0-1 semidefinite programming, and then show how to relax K-means by this model.

2.1.1 0-1 semidefinite programming model

In the class, we learnt about the general form of SDP problem, that is

$$\begin{array}{ll} \text{minimization} & Tr(WZ) \\ \text{subject to} & Tr(B_i Z) = b_i \quad \text{for i = 1, ..., m.} \\ & Z \succeq 0. \end{array} \tag{3}$$

where elements in the matrix Z can be any real number which makes Z positive semidefinite.

In 0-1 semidefinite programming, the original $Z \succeq 0$ is substituted with two constraints, $Z^2 = Z$ and $Z = Z^T$, which guarantee the elements in Z to be 0 or 1.

$$\begin{array}{ll} \text{minimization} & Tr(WZ) \\ \text{subject to} & Tr(B_i Z) = b_i \qquad \text{for i = 1, ..., m.} \\ & Z^2 = Z, Z = Z^T. \end{array} \tag{4}$$

2.1.2 0-1 SDP relaxation for K-means problem

In this subsection, I will show how to transfer the K-means problem to 0-1 SDP model, and then change the symbols in [1] into those in [2,3,4].

Consider the following problem: given a dataset $S = \{s_i = (s_{i1}, \ldots, s_{id})^T \in \mathbf{R}^d, i = 1, \ldots, n\}$, which is in the Euclidean space, partition the whole dataset into k disjoint clusters $S = \{S_1, \ldots, S_k\}$, whose centroid set is $C = \{c_1, \ldots, c_k\}$. Our goal is to minimize the following objective function which measures the total distance between all the data points and the centroids.

$$f(S, S) = \sum_{j=1}^{k} \sum_{i=1}^{|S_j|} \left\| s_i^{(j)} - c_j \right\|_2^2. \tag{5}$$

It is easy to prove that if we fix one of the clusters S_j, the distance objective function for that cluster is minimized by

$$c_j = \frac{1}{|S_j|} \sum_{i=1}^{|S_j|} s_i^j \tag{6}$$

From the previous definition, it is the centroid for the cluster S_j.

According to [1], there is another way to describe the centroid. Let $X = [x_{ij}] \in \mathbf{R}^{n \times k}$ be the matrix, where its elements are defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } s_i \in S_j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Therefore, we can see that in the matrix X, a row stands for which cluster a point belongs to, while a column stands for which points are assigned to this cluster. According to the properties of K-means, we can get two constraints for the matrix X. For every row, there is only one 1, and others are all 0's. This is because for every point, we can only assign it to the closest centroid. For every column, there is at least one 1, since if all the elements in that column are 0's, this centroid does not need to exist anymore. Because we expect the algorithm to cluster all the points into k clusters, all columns have to have at least one 1. In addition, note that every column is orthogonal to each other, since for every point, we can assign it to only one cluster.

Using the x_{ij}, we can define the centroid of the cluster by

$$c_j = \frac{\sum_{l=1}^{n} x_{lj} s_l}{\sum_{l=1}^{n} x_{lj}} \tag{8}$$

As a result, we can use this form to represent (5) by

$$\begin{aligned} \text{minimization} \quad & \sum_{j=1}^{k} \sum_{i=1}^{n} x_{ij} \left\| s_i^{(j)} - \frac{\sum_{l=1}^{n} x_{lj} s_l}{\sum_{l=1}^{n} x_{lj}} \right\|_2^2. \\ \text{subject to} \quad & \sum_{j=1}^{k} x_{ij} = 1 \text{ for i} = 1, \ldots, \text{n}, \\ & \sum_{i=1}^{n} x_{ij} \geq 1 \text{ for j} = 1, \ldots, \text{k}, \\ & x_{ij} \in \{0, 1\} \text{ for i} = 1, \ldots, \text{n; j}=1, \ldots, \text{k}. \end{aligned} \tag{9}$$

Until now, we have known the necessary theoretical background of 0-1 SDP and K-means problem. Next, we can transfer the new problem (9) to a 0-1 SDP model. Although Peng and Wei [1] gave a very detailed process, for me this is very interesting and essential for the following contexts. Therefore, I still want to give the complete derivation.

We can get the following objective function by rearranging (9),

$$\begin{aligned} f(S, S) &= \sum_{i=1}^{n} \|s_i\|^2 \left(\sum_{j=1}^{k} x_{ij} \right) - \sum_{j=1}^{k} \frac{\| \sum_{i=1}^{n} x_{ij} s_i \|^2}{\sum_{i=1}^{n} x_{ij}} \\ &= Tr(W_S W_S^T) - \sum_{j=1}^{k} \frac{\| \sum_{i=1}^{n} x_{ij} s_i \|^2}{\sum_{i=1}^{n} x_{ij}} \end{aligned} \tag{10}$$

where $W_S \in \mathbf{R}^{n\times d}$ is the matrix in which s_i^T is its i-th row. Moreover, note that for every row $\sum_{j=1}^{k} x_{ij}$ is 1. From the previous definition, X is the assignment matrix and its elements are 0 or 1, so we can have

$$X^T X = diag\Big(\sum_{i=1}^{n} x_{i1}^2, \ldots, \sum_{i=1}^{n} x_{ik}^2\Big) = diag\Big(\sum_{i=1}^{n} x_{i1}, \ldots, \sum_{i=1}^{n} x_{ik}\Big) \tag{11}$$

Let

$$Z = X(X^T X)^{-1} X^T \tag{12}$$

Using this definition, we can rewrite (10) as $Tr(W_S W_S^T (I - Z))$. We can see easily from the definition of Z (12) that $Z = Z^2$ and $Z = Z^T$.In addition, for some integer m, e^m is the all-one vector in $\mathbf{R}^m$ as we learnt in the class. Then, since the sum of every row of X is 1, we can get the constraint

$$Xe^k = e^n \tag{13}$$

As an immediate consequence,

$$Ze^n = ZXe^k = Xe^k = e^n \tag{14}$$

Due to some linear algebra, we can find that $\left(\frac{\sum_{i=1}^{n} x_{ij}^2}{\sum_{i=1}^{n} x_{ij}}\right)$ is on the diagonal of Z. Therefore,

$$\begin{aligned} Tr(Z) &= \sum_{j=1}^{k} \Big(\frac{\sum_{i=1}^{n} x_{ij}^2}{\sum_{i=1}^{n} x_{ij}}\Big) \\ &= \sum_{j=1}^{k} 1 \\ &= k. \end{aligned} \tag{15}$$

Here, note that although we compute the trace of Z, itself is not a diagonal matrix!

Hence, we can have the following 0-1 SDP model for the original problem of K-means

$$\begin{array}{ll} \text{minimization} & Tr(W_S W_S^T (I - Z)) \\ \text{subject to} & Ze^n = e^n, Tr(Z) = k \\ & Z \geq 0, Z^2 = Z, Z = Z^T. \end{array} \tag{16}$$

In other reference papers, there is another equivalent model as mentioned before. However, due to the different symbols, sometimes it is hard to understand the connection between those two models at the first glance. So the following part of this subsection will show the other form and the connection between them.

According to [2], we have the same objective function as (1) for K-means problem. Then we can have the following semidefinite relaxation

$$\begin{array}{ll} \text{minimization} & Tr(DX) \\ \text{subject to} & Tr(X) = k, X\mathbf{1} = \mathbf{1}, X \geq 0, X \succeq 0 \end{array} \tag{17}$$

where D is the $N \times N$ matrix defined by $D_{ij} = \|x_i - x_j\|_2^2$ and matrix X is defined by

$$X = \sum_{t=1}^{k} \frac{1}{|A_t|} 1_{A_t} 1_{A_t}^T \tag{18}$$

where X here is Z in paper [1]. Also,1_{A_t} is the column of X in Peng and Wei's paper, which denotes the indicator function about which data points belong to the cluster A_t. Therefore, the objective function (1) can be expressed as $\frac{1}{2}Tr(DX)$ (note D is a symmetric matrix).

2.1.3 Notes about (17) and (18)

There are some details about X which are not obvious in [2]. When I investigated this problem, they were obstacles for me to understand the SDP, so I write them down here as the supplement notes for the original paper.

1.there is an interesting property of X: it has a similar form of X in [1], but there is a little difference in the definition: when point i and point j belong to the same cluster, $x_{ij} = \frac{1}{|A_t|}$ instead of 1.

2.there is an essential key point about getting the objective function $\frac{1}{2}Tr(DX)$: the sum of the distances between points in the cluster and the centroid is equal to the sum of the mutual distance between any points in that cluster. This is because the centroid $c_t = \frac{1}{|A_t|}\sum_{j \in A_t} x_j$ is chosen as the average of the cluster points.

3.to get the correct answer, we need to shuffle the elements in columns of X in [1] to get together those points which belong to the same cluster. For example, there are four points in the datasets, and their assignment matrix is

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Then we need to shuffle the columns to get $1_{A_1} = (1,1,1,0,0)^T$ and $1_{A_2} = (0,0,1,1,1)^T$. The corresponding X is

$$X = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

where we can see that the corresponding blocks which containing all X_{ij} for the same cluster $A_j \in \{A_1, A_2, \ldots, A_k\}$ are on the diagonal according to the order of clusters. In addition, we need to shuffle the corresponding elements in D as well.

2.1.4 Reasons to relax by SDP

Why we use SDP to relax the original problem stated in (9), or why we need to relax it? From (9), we can see there are two difficulties for us to solve it directly.

First, because $x_{ij} \in \{0, 1\}$, we can only get the discrete constraint, which is much like Max Cut Problem in the textbook [5]. Second, the objective function is not convex, therefore, we cannot use the current tools provided such as CVX to solve it. Using SDP to relax this problem can overcome those two main obstacles, which we can easily see from (16) or (17). Moreover, from Theorem 2.1 of [1], we know that solving 0-1 problem (16) or (17), is equivalent to finding a global solution of the integer programming problem (9). Also, according to Iguchi [1], most non-convex optimization methods cannot produce a certificate of global optimality as we expect. However, if a non-convex problem has a corresponding convex relaxation, then by solving the dual problem of this relaxation, we can produce a certificate of a (approximate) optimality. So in the next section, I will show the idea provided in [2] to give the certificate of the global optimality for the K-means problem.

2.1.5 SDP dual problem for K-means

As I stated in the last section and the beginning of section 2.1, I will show the SDP dual problem for K-means problem (17) without proof. The detailed proof is in the paper [4]. For the completeness, I show the dual problem here, though it is used for the next section about getting the certificate. As mentioned before, we have k clusters in $\mathbf{R}^m$, each contains n points. Hence, the total number of data points is $N = k \times n$. In a unconventional notation, we index a data point with (a, i), where a stands for the cluster it belongs to, and i is the index of the point in that cluster. With this kind of notations, the distance between two points is $d_{(a,i),(b,j)}$. So the $N \times N$ matrix D consists of blocks $D_{ij}^{(a,b)} = d_{(a,i),(b,j)}^2$. Hence, we can describe our SDP dual problem for (17) as follows:

$$\begin{aligned} \underset{z \in \mathbf{R}, \alpha}{\text{minimization}} \quad & kz + \sum_{a=1}^{k} \sum_{i=1}^{n} \alpha_{a,i} \\ \text{subject to} \quad & Q = zI_{N \times N} + \sum_{a=1}^{k} \sum_{i=1}^{n} \alpha_{a,i} A_{a,i} \\ & \sum_{a,b=1}^{k} \sum_{i,j=1}^{n} \beta_{i,j}^{(a,b)} E_{(a,i),(b,j)} + D \\ & \beta_{i,j} \geq 0 \\ & Q \succeq 0 \end{aligned} \tag{19}$$

where $A_{a,i} = \frac{1}{2}(\mathbf{1}e_{a,i}^T + e_{a,i}\mathbf{1}^T)$, $E_{(a,j),(b,j)} = \frac{1}{2}(e_{b,j}e_{a,i}^T + e_{a,i}e_{b,j}^T)$, $\mathbf{1} \in \mathbf{R}^{N \times 1}$ and $e_{a,i} \in \mathbf{R}^{N \times 1}$ is the indicator function for index (a, i).

2.2 Certifiers for K-means clustering

As mentioned above, Lloyd's algorithm will usually not converge to the global optimum, and it will not give any information about how close the result is to the global optimal. Based on the those two disadvantages, Iguchi [2] introduced a fast probably certifiably correct algorithm to overcome these. In this section, I will introduce some essential theoretical background for the certifier from the paper [2] and the textbook [5]. Then introduce an important model, stochastic

ball model, and show that the 0-1 SDP relaxation (17) from [1] is tight under this model. Finally, this report will give two algorithms introduced in [2], one for testing K-means optimality and the other for solving K-means problem with two clusters.

2.2.1 Preliminaries

In this subsection, I will introduce some essential backgrounds for dual certificate, which are not covered in the paper [2].

Definition 1 *for a set C, the **dual cone** is such a set that satisfies*

$$C^* = \{x : \langle x, y\rangle \geq 0 \ \forall y \in C\}$$

With the definition of dual cone, we can represent the primal program

$$\begin{aligned} \min_{x} \quad & \langle c, x\rangle \\ \text{s.t.} \quad & Ax - b \in L \\ & x \in K \end{aligned}$$

and the dual program can be described as follows

$$\begin{aligned} \max_{y} \quad & \langle b, y\rangle \\ \text{s.t.} \quad & c - A^T y \in K^* \\ & y \in L^* \end{aligned}$$

There is a little difference between this form and that we learnt from the class. That is, now we use the dual cone to represent the primal and the dual program, which is more generalized.

Definition 2 *if x is feasible for primal program and y is feasible for dual program, then we say a optimization problem satisfies **strong duality** if*

$$\langle c, x_{opt}\rangle = \langle b, y_{opt}\rangle$$

we can use the definition of strong duality to find the certificate of optimality for a convex optimization problem, which is exactly the method Iguchi [2] used for finding the certificate for K-means problem using SDP relaxation.

In general, the **certificate of optimality** is the task that given x_{opt}, to quickly find y_{opt}. The common method is as follows:

1. Check x_{opt} is primal feasible
2. Find y such that $(x_{opt}, y) \in S = \{(x, y) : \langle x, c\rangle = \langle b, y\rangle\}$
3. Check y is dual feasible. If it is, then we can say that x_{opt} is optimal for the K-means clusters.

2.2.2 Stochastic ball model and tightness of SDP relaxation

It is shown that the 0-1 SDP relaxation is actually tight under a specific model, stochastic ball model, with a high probability [2]. In this subsection, I will give the definition of stochastic ball model, and give a theorem from [2] to show that the SDP relaxation is tight with some probability. Because the proof is very complicated, and it is not important for this report, if readers are interested in this, please take a look at Section 2 of [2].

Definition 3 ((𝒟,γ,n)-stochastic ball model) *Let* $\{\gamma_a\}_{a=1}^k$ *be ball centers in* $\mathbf{R}^m$. *For each a, draw i.i.d. vectors* $\{r_{a,i}\}_{i=1}^n$ *from some rotation-invariant distribution* $\mathcal{D}$ *supported on the unit ball. The points from cluster a are then taken to be* $x_{a,i} = r_{a,i} + \gamma_a$. *We denote* $\triangle = min_{a \neq b} \|\gamma_a - \gamma_b\|_2$.

Note here γ is the center of the unit ball and r is the distance between the data point and the center for a cluster.

From the next theorem, we can see that SDP relaxation (17) is typically tight for the K-means problem.

Theorem 4 *The k-means semidefinite relaxation (17) recovers the planted clusters in the (𝒟,γ,n)-stochastic ball model with probability* $1 - e^{-Omega_{\mathcal{D},r}(n)}$ *provided* $\triangle > 2 + k^2/m$.

where m is the dimension of the Euclidean space where the balls are located and the planted clusters in $\mathbf{R}^2$ are defined as follows:

Definition 5 (planted cluster) *Take two unit disks in* $\mathbf{R}^2$ *with centers on the x-axis at distance* $\triangle$. *Let* x_0 *denote the smallest possible x-coordinate in the disk on the right. Given* θ, *clusters the points according to whether the x-coordinate is smaller than* $x_0 + \theta$. *When* $\theta = 0$, *this clustering gives* ***planted clusters***.

2.2.3 A fast test algorithm for K-means optimality

As we have already known, Lloyd's algorithm does not give the certificate of optimality for K-means. However, if a non-convex problem (9), has a convex relaxation (17), then we can solve the dual program of it (19) to get the certificate for optimality. In [2], the authors relied on the technique introduced by Bandeira to devise a fast algorithm for K-means optimality. Therefore, in this section, I will introduce the technique and transfer it into a specific problem according to [2]. Then to solve this problem, they gave a subroutine called Power Iteration Detector of the certificate algorithm.

According to [2], there is a general technique to certify global optimality for optimization problem. There are three essential components considered in this technique:

(1) A fast non-convex solver which generates the optimal solution with high probability (under some reasonable probability distribution of problem instances).

(2) A convex relaxation that is tight with high probability (under the same distribution).

(3) A fast method of computing a certificate of global optimality for the output of the non-convex solver in (1) by exploiting convex duality with the relaxation in (2).

Now, let's analyze this technique for K-means particularly. For (1), we have already have a non-convex solver for K-means, which is Lloyd's algorithm. For (2), in the last section, we have shown that 0-1 SDP relaxation given in [1] is actually typically tight with high probability under the stochastic ball model. Therefore, why we need to invent the algorithm is because we need to tackle (3). After giving the algorithm, we can use the above-mentioned technique to check the optimality for K-means.

For tackling (3), we can transfer it equivalently to the following problem:

Problem 6 *Given a symmetric matrix $A \in \mathbf{R}^{n\times n}$ and an eigenvector v of A, determine whether the span of v is the unique leading eigenspace, that is , the corresponding eigenvalue λ has multiplicity 1 and satisfies $|\lambda| > |\lambda'|$ for every other eigenvalue λ' of A.*

where $A = \frac{z}{N}\mathbf{1}\mathbf{1}^T + P_{\Lambda^\perp} Z P_{\Lambda^\perp}$. All the symbols in it are from the following theorem:

Theorem 7 *Take X of the form (18), and let $P_{\Lambda^\perp}$ denote the orthogonal projection onto the orthogonal complement of the span $\{1_{A_t}\}_{t=1}^k$. Then there exists an explicit matrix $Z = Z(D, X)$ and scalar $z = z(D, X)$ such that X is a solution to the semidefinite relaxation (17) if*

$$P_{\Lambda^\perp} Z P_{\Lambda^\perp} \preceq z P_{\Lambda^\perp}.$$

To solve the Problem 6, we need to apply the power method: Let $A \in \mathbf{R}^{n\times n}$ be a symmetric matrix with eigenvalues $\{\lambda_i\}_{i=1}^n$ (counting multiplicities) satisfying $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$, and with corresponding orthonormal eigenvectors $\{v_i\}_{i=1}^n$. Pick a unit-norm vector $q_0 \in \mathbf{R}^n$ and consider the power iteration $q_{j+1} = Aq_j/\|Aq_j\|_2$. If q_0 is not orthogonal to $v1$, then $(v_1^T q_j)^2 \geq 1 - \left((v_1^T q_0)^{-2} - 1\right)\left(\frac{\lambda_2}{\lambda_1}\right)^{2j}$.

The power method is typically used to find a leading eigenvector, but for K-means problem, we have got the eigenvector v. What our task is to determine whether v is the unique leading eigenvector.

For describing the algorithm clearly, I will give the simple intuitive of how the algorithm comes from. Consider that when we run the power method from a random initialization, such as that of the centroid in K-means, and it converges to v coincidently, then this would have been a remarkable coincidence if v were not the unique leading eigenvector. Since we will only run finitely many iterations, how do we determine when we are sufficiently confident? Given a symmetric $A \in \mathbf{R}^{n\times n}$ in Problem 6, and a unit eigenvector v of A, consider the hypotheses

$$\begin{aligned} &H_0 : \text{span}(v) \text{ is not the unique leading eigenspace of } A, \\ &H_1 : \text{span}(v) \text{ is the unique leading eigenspace of } A. \end{aligned} \tag{20}$$

To test these hypotheses, pick a tolerance $\epsilon > 0$ and run the power iteration detector. This detector terminates either by accepting H_0 or by rejecting H_0 and accepting H_1. We say the detector **fails to reject** H_0 if it either accepts H_0 or fails to terminate. For the detailed analysis of the the detector, please see Section 3.1 in [2]. Next, I give the algorithm as follows.

Algorithm 1: Power iteration detector

Input: Symmetric matrix $A \in \mathbf{R}^{n\times n}$, unit eigenvector $v \in \mathbf{R}^n$, tolerance $\epsilon > 0$

Output: Decision of whether to accept H_0 or to reject H_0 and accept H_1 as given in (20)

$\lambda \leftarrow v^T A v$;

Draw q uniformly at random from the unit sphere in $\mathbf{R}^n$;

while *no decision has been made* **do**

  if $|q^T A q| > |\lambda|$ **then**

    Print "accept H_0"

  else if $(v^T q)^2 \geq 1 - \epsilon$ **then**

    Print "reject H_0 and accept H_1";

  $q \leftarrow Aq/\|Aq\|_2$;

For the result of this checking algorithm, please see the experiment section.

2.2.4 A fast K-means solver for two clusters

In the last section, I gave the algorithm to check whether the solution for K-means algorithm is global optimal under a special model, stochastic ball model. It turns out that Peng and Wei's SDP relaxation is tight in this kind of data model. Now according to [1] and [2], there is another way to relax the K-means problem (9). Based on this relaxation, [2] devised a new algorithm to solve the K-means problem with $k = 2$ under the stochastic ball model. In this section, I follow [2], to introduce the relaxation and then the algorithm.

Recall that the 0-1 relaxation of K-means problem is

$$\begin{array}{ll} \text{minimization} & Tr(DX) \\ \text{subject to} & Tr(X) = k, X\mathbf{1} = \mathbf{1}, X \geq 0, X \succeq 0 \end{array} \tag{21}$$

Then we can get the **spectral clustering** by discarding the constraint $X \geq 0$. To get our new relaxation, first we define a $m \times N$ matrix $\Phi = [x_1, x_2, \ldots, x_N]$ by the following equation,

$$D_{ij} = \|x_i - x_j\|_2^2 = \|x_i\|_2^2 - 2x_i^T x_j + \|x_j\| = (\nu 1^T - 2\Phi^T\Phi + 1\nu^T)_{ij} \tag{22}$$

where $\nu = \|x_j\|_2^2$ is the $N \times 1$ vector. It is easy to see that Φ is the matrix whose column is the data point. Without loss of generality, we assume that the data set is centered at the origin, so $\Phi 1 = 0$. With $D = \nu 1^T - 2\Phi^T\Phi + 1\nu^T$ and two constraints in (21),$X = X^T$ and $X1 = 1$, we can get the objective function of another format:

$$\begin{aligned} Tr(DX) &= Tr(\nu 1^T - 2\Phi^T\Phi + 1\nu^T) \\ &= Tr(\nu 1^T X^T) - 2Tr(\Phi^T\Phi X) + Tr(X 1\nu^T) \\ &= 2\nu^T 1 - 2Tr(\Phi^T\Phi X) \end{aligned} \tag{23}$$

So minimizing $Tr(DX)$ is equivalent to maximizing $Tr(\Phi^T\Phi X)$. In addition, the feasible X in the relaxation is precisely the rank-k $N \times N$ orthogonal projection matrix satisfying $X1 = 1$. Therefore, X can have the form $X = \frac{1}{N}11^T + Y$

where Y is a rank-(k-1) $N \times N$ orthogonal projection matrix satisfying $Y1 = 0$. Note that $\frac{1}{N}11^T$ is constant. In addition, we discard $Y1 = 0$ as well, so that we can get the next equivalent relaxation of (21):

$$\begin{array}{ll} \text{minimization} & Tr(\Phi^T \Phi Y) \\ \text{subject to} & Y^T = Y, Y^2 = Y, Tr(Y) = k-1 \end{array} \tag{24}$$

Note that after our initial goal is not to find Y! What problem we want to solve is to get the corresponding X. Therefore, after figuring out Y, we need to recover it back to $X = \frac{1}{N}11^T + Y$ and then round it to a closest member of the feasibility region in (21). As mentioned above, here we only care about the exact recovery under the stochastic ball model.

Now, we can use the (24) for getting our fast algorithm for $k = 2$ clusters. When $k = 2$, Y has the form $Y = yy^T$, where y is the leading unit eigenvector of $\Phi^T\Phi$. We need to find a matrix of form $\frac{1}{|A|}1_A 1_A^T + \frac{1}{|B|}1_B 1_B^T$ with $A \cup B = \{1, \ldots, N\}$ that is close to $\frac{1}{N}11^T + yy^T$. From this, since there are two clusters for the stochastic ball model, it is natural to consider the following:

$$A_\theta = \{i : y_i < \theta\},\ B_\theta = A_\theta^c$$

for some threshold θ. Here, we choose the θ which minimized the K-means objective of (A_θ, B_θ). Order the indices so that $y_1 \leq \cdots \leq y_N$. Then we need to minimize the following function:

$$f(i) = \frac{1}{i} \overbrace{\underbrace{\sum_{j=1}^{i}\sum_{j'=1}^{i} \|x_j - x_{j'}\|_2^2}_{v_i}}^{A_\theta} + \frac{1}{N-i} \overbrace{\underbrace{\sum_{j=i+1}^{N}\sum_{j'=i+1}^{N} \|x_j - x_{j'}\|_2^2}_{v_i^c}}^{B_\theta = A_\theta^c}.$$

We can expand the square and distribute sums and we can get

$$v_{i+1} = v_i + 2\sum_{j=1}^{i} \|x_j\|_2^2 - 4x_{i+1}^T \sum_{j=1}^{i} x_j + 2i\|x_{i+1}\|_2^2$$
$$v_{i+1}^c = v_i^c + 2\sum_{j=i+1}^{N} \|x_j\|_2^2 - 4x_{i+1}^T \sum_{j=i+1}^{N} x_j + 2i\|x_{i+1}\|_2^2$$

So we can iteratively compute the v_i's and v_i^c's before computing the $f(i)$'s and then minimizing as the following algorithm does

Algorithm 2: Spectral k-means clustering (for two clusters under stochastic ball model)

Input: $m \times N$ matrix $\Phi = \{x_1, \ldots, x_N\}$ of points to be clustered
Output: Clusters $A \cup B = 1, \ldots, N$
Subtract centroid $\frac{1}{N}\sum_{i=1}^{N} x_i$ from each column of Φ to produce Φ_0 ;
Compute the leading eigenvector y of $\Phi_0^T\Phi_0$;
Find θ that minimizes the k-means objective of $(\{i : y_i < \theta\}, \{i : y_i \geq \theta\})$;
$(A, B) \leftarrow (\{i : y_i < \theta\}, \{i : y_i \geq \theta\})$;

Now through the following analysis, we can get the conclusion that the overall operations for finding the optimal (A_θ, B_θ) is $\mathcal{O}((m + logN)N)$, which is much better than the Lloyd's algorithm.

Time complexity analysis

1. Sort the entries $y_1 \leq \cdots \leq y_N$ in $\mathcal{O}(NlogN)$ operations.

2. Iteratively compute

$$s_1(i) = \sum_{j=1}^{i} x_j,\ s_j^c(i) = \sum_{j=i+1}^{N} x_j,\ s_2(i) = \sum_{j=1}^{i} \|x_j\|_2^2,\ s_2^c(i) = \sum_{j=i+1}^{N} \|x_j\|_2^2$$

for every $i \in \{1, \ldots, N-1\}$ in $\mathcal{O}(mN)$ operations.

3. Compute $v_1 = 0$ and $v_{i+1} = v_i + 2s_2(i) - 4x_{i+1}^T s_1(i) + 2i\|x_{i+1}\|_2^2$ for every $i \in \{1, \ldots, N-2\}$ in $\mathcal{O}(mN)$ operations.

4. Compute $v_{N-1}^c = 0$ and $v_{i-1}^c = v_i^c + 2s_2^c(i) - 4x_{i+1}^T s_1^c(i) + 2(N-i)\|x_i\|_2^2$ for every $i \in \{N-1, \ldots, 2\}$ in $\mathcal{O}(mN)$ operations.

5. Compute $f(i) = v_i/i + v_i^c/(N-i)$ for every $i \in \{1, \ldots, N-1\}$ in $\mathcal{O}(N)$ operations.

6. Find i that minimizes $f(i)$ and output $\{1, \ldots, i\}$ and $\{i+1, \ldots, N\}$ in $\mathcal{O}(N)$ operations.

Therefore, we can see that the spectral algorithm requires only quasilinear computational complexity $\mathcal{O}((m + logN)N)$, which guarantees that under the stochastic ball model, it performs better than Lloyd's algorithm. For the experiment result, please see the experiment chapter.

2.3 Clustering subgaussian mixture data by SDP

In the last part, I followed [2] to give the result that under the stochastic ball model, Peng and Wei's 0-1 SDP relaxation is tight with high probability. Then I introduced an algorithm to quickly check whether the solution of K-means is close to the global optimum. In addition, a well-performance algorithm was given for clustering when $k = 2$ under the stochastic ball model with SDP relaxation (24).

We have known that the stochastic ball model allows the SDP relaxation (17) to be tight, and all algorithms mentioned above perform well under such a model. However, the presumption that the data conform to the stochastic ball model is strong and the real data may be not constrained by this model. Hence, can SDP relaxation still guarantee the tightness for the realistic data? How to find a more general algorithm to tackle the k-means problem for the real data? To answer those questions, I will follow the paper [3] to explore these questions under the subgaussian mixtures model, whose special cases include the stochastic ball model and Gaussian mixture model. In this section, I introduce a model-free relax-and-round algorithm for K-means cluster based on the SDP (17) and provide a generic method for proving the performance guarantees for this algorithm, and analyze this algorithm under the context of subgaussian mixture models. In all this parts, I think the explanation in [3] to interpret the SDP relaxation as the denoised version of the data is interesting, so I will emphasize this later.

2.3.1 Subgaussian mixture model and relax-and-round algorithm

First, I will give the definition of the subgaussian mixture model, and then use the same chain to introduce the relax-and-round algorithm for this model. I leave the analysis of the algorithm to the next section.

Definition 8 (subgaussian mixture model) *For each $t \in [k] = \{1, \ldots, k\}$, let $\mathcal{D}_t$ be an unknown subgaussian probability distribution over $\mathbf{R}^m$, with first moment $\gamma_t \in \mathbf{R}^m$ and second moment matrix with largest eigenvalue σ_t^2. For each t, an unknown number n_t of ransom points $\{x_{t,i}\}_{i \in [n_t]}$ is drawn independently form $\mathcal{D}_t$. Given the points $\{x_{t,i}\}_{i \in [n_t], t \in [k]}$ along with with the model order k, the goal is to approximate the centers $\{\gamma_t\}_{t \in [k]}$.*

It is easy to see from **Definition 3** and **Definition 8** that stochastic ball model is the subset of subgaussian mixture model.

Although we have shown without proof that SDP (17) is tight for the stochastic ball model, it is no longer the case in the subgaussian mixture model. However, the authors of [3] interpreted the result optimizer of SDP as the denoised version of the raw data, and can continue to cluster based on this.

Now, let's illuminate this. P denotes the $m \times N$ matrix whose columns are the points $\{x_{t,i}\}_{i \in [n_t], t \in [k]}$. This definition is the same as Φ in (24). It can be showed that when X has the form (18), which we call it integral, for each $t \in [k]$, PX has $|A_t|$ columns equal to the centroid of points assigned to A_t. That is,

$$PX = [\underbrace{\hat{\gamma_1} \cdots \hat{\gamma_1}}_{|A_1| \text{ copies}} \underbrace{\hat{\gamma_2} \cdots \hat{\gamma_2}}_{|A_2| \text{ copies}} \cdots \underbrace{\hat{\gamma_k} \cdots \hat{\gamma_k}}_{|A_k| \text{ copies}}]$$

Then, [3] noticed that every SDP-feasible matrix $X \geq 0$ satisfies $X^T 1 = X1 = 1$ as in (17), and so X_T is a stochastic matrix, meaning each column of PX is still a weighted average of columns from P. If the SDP relaxation (17) were close to being tight, then the SDP -optimal X would make the columns of PX close to the k-means-optimal centroids. Intuitively, we can interpret PX as a denoised version of the original data P, and we can use another method to get the good estimates for the original k-means-optimal centroids.

According to what says above, we can get the algorithm to first run SDP to get the denoised version PX of the raw data P, then cluster the PX to get a good estimates for k centroids.

Algorithm 3: Relax-and-round k-means clustering procedure

Input: $m \times N$ data matrix $P = [x_1, \ldots, x_N]$
Output: Clusters of the denoised data matrix PX
Compute distance squared matrix D defined by $D_{ij} = \|x_i - x_j\|$;
Solve k-means semidefinite program (17), resulting in optimizer X ;
Cluster the columns of the denoised data matrix PX

In the first step, we can use the columns in the raw data matrix P to get the distance matrix D, then we can use the above algorithm based on SDP relaxation to get the optimizer X. In the third step, we can use the Lloyd's algorithm to cluster the denoised data in order to get the good estimators of centroids. Here, we use Lloyd's algorithm is because after the process of denoising, the number

of the data decreases greatly; therefore, as stated before, we can use Lloyd's algorithm on a small dataset. It is proved that this algorithm works well as we expected.

2.3.2 Analysis of the relax-and-round algorithm

Followed paper [3], I will show succinctly in this section that the relax-and-round algorithm guarantees that it will recover the centroid of k-means problem under the subgaussian mixture model. There needs three steps to achieve this goal as stated in the paper.

Approximation Given the data points for the cluster t, $x_1 = \{x_{t,1}, \ldots, x_{t,n_t}\}$, drawn independently from $\mathcal{D}_t$, consider the squared-distance matrix D and the optimizer X_D of the SDP relaxation (17). We first construct a "reference" matrix R such that under the subgaussian model, when $D = R$, the optimizer X_R is tight. Take $\triangle_{ab} = \|\gamma_a - \gamma_b\|_2$, and let X_D denote the minimizer of (17), and X_R denote the minimizer of (17) when D is replaced by the matrix R. Now, we define the matrix R as follows:

$$(R_{ab})_{ij} = \xi + \triangle_{ab}^2/2 + max\{0, \triangle_{ab}^2/2 + 2\langle r_{a,i} - r_{b,j}, \gamma_a - \gamma_b\rangle\} \qquad (25)$$

where $r_{t,i} = x_{t,i} - \gamma t$ and $\xi > 0$. Now, according to the observation, the reference matrix R (25) enjoys the following property: Let $1_a \in \mathbf{R}^N$ denote the indicator function for the indices i corresponding to points x_i drawn from the a-th subgaussian. If $\gamma_a \neq b$, then $X_R = \sum_{t=1}^{k} \frac{1}{n_t} 1_t 1_t^T$. Please note that here, X_R has the same form of (18), not X_D. We can consider X_R as the ground truth and X_D as the estimator of optimizer X_R after the process of denoising.

The following theorem can ensure that the solution from solving SDP (17) is actually close to the ground truth.

Theorem 9 *For the fixed $\epsilon, \eta > 0$, there exist universal constants C, c_1, c_2, c_3 such that if*

$$\alpha = n_{max}/n_{min} \leq k \leq m \text{ and } N > max\{c_1 m, c_2 log(2/\eta, log(c_3/\eta))\},$$

then, $\|X_D - X_R\|_F^2 \leq \epsilon$ with probability $\geq 1 - 2\eta$ provided

$$\triangle_{min}^2 \geq \frac{C}{\epsilon} k^2 \alpha \sigma_{max}^2$$

where $\triangle_{min} = min_{a \neq b} \|\gamma_a - \gamma_b\|_2$ is the minimal cluster center separation.

Denoising Now, from **Theorem 9**, we can see that X_D is close to the ground truth. What we need to do next is convert X_D to an estimate for the centers $\{\gamma_t\}_{t \in [k]}$. Let P and X_R be the same definition as stated before. Then PX_R is an $m \times N$ matrix whose (a, i)th column is $\hat{\gamma_a}$, the centroid of the ath cluster, which converges to γ_a as $N \to \infty$, and PX_D is a denoised version of the points. By the next theorem, we can guarantee that the algorithm denoiding in the regime $K \ll \sqrt{m}$.

Theorem 10 *Let $c_{a,i}$ denote the (a,i)th column of PX_D. Assume the points $\{x_{a,i}\}_{i \in [n]}$ come from $\mathcal{N}(\gamma_a, \sigma^2 I_m)$ in $\mathbf{R}^m$ for each $a \in [k]$. If $k\sigma \leq \triangle_{min} \leq \triangle_{max} \leq K\sigma$,*

then

$$\frac{1}{N}\sum_{a=1}^{k}\sum_{i=1}^{n}\|c_{a,i}-\hat{\gamma_a}\|\leq K^2\sigma^2$$

with high probability as $n\to\infty$.

Rounding In the **Algorithm 3**, there is no clue how to "round" the data from what we got from SDP. Actually, there is a simple scheme provided to do this: for every cluster, do the following: (1) $v_i \leftarrow$ denoised point with most neighbors; (2) delete denoised point and neighbors. After this, we can see v_i as the "true" centroid for the cluster i.

Under the same hypothesis as **Theorem 10**, we have that there exists a permutation π on $\{1,\ldots,k\}$ such that

$$\frac{1}{k}\sum_{i=1}^{k}\|v_i-\hat{\gamma}_{\pi(i)}\|_2^2\leq k^2\sigma^2 \tag{26}$$

where v_i is what **Algorithm 3** chooses as the ith center estimation. Hence, we can estimate Gaussian centers with mean squared error $\mathbf{O}(k^2\sigma^2)$ provided the centers have pairwise distance $\mathbf{\Omega}(k\sigma)$.

3 Experiment results and interpretation

All experiments are implemented on Lenovo Thinkpad T470p, with 16GB memory, using Matlab R2017b.

3.1 Comparison of solvers for two clusters

This experiment corresponds to the **section 2.2.4**, where I showed a fast K-means solver for two clusters under the stochastic ball model. The results are gotten after implementing **Algorithm 2**.

First, to see the clustering results, please see Figure 1. There are four figures showing four experiments, during which the data points are generated independently.

Then in order to prove that the **Algorithm 2** performs well and show that SDP (17) is actually tight, I compare it with the Lloyd's Algorithm from both correctness and speed.

To compare the correctness of them, I used misclass rate, which is defined as follows:

$$\epsilon=\frac{\#\text{ of miscluster data}}{\#\text{ of total data}}$$

Since the Lloyd's algorithm has the high time complexity (which will be shown in the speed comparison), I only tested them on a relatively small dataset, about 2^6 data points, and did every method 10 times. **Table 1** shows the worst and best misclass rate for both methods.

We can see that in both cases, spectral clustering algorithm in paper [2] has the approximately same misclass rate as Lloyd's algorithm. Thus, we can conclude that this algorithm works well and also the SDP (17) is actually tight for K-means problem.

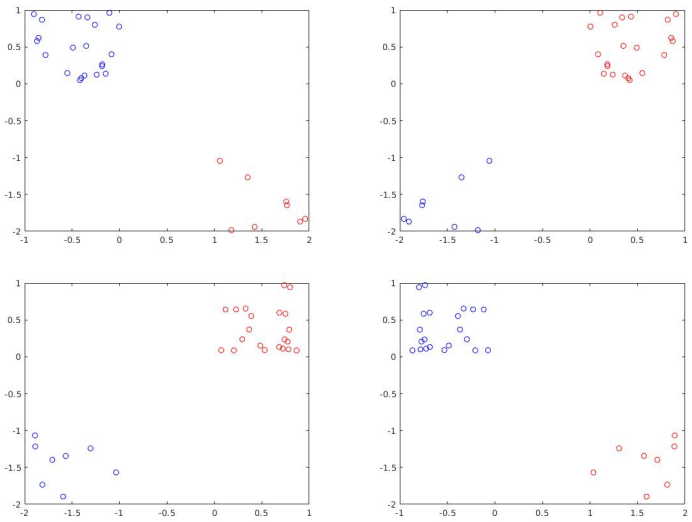

Figure 1: Results for clustering into two clusters. The data points in each figure are generated independently. Different colors represent different clusters.

Table 1: Misclass rate for algorithms

Algorithm	**worst rate**	**best rate**
Lloyd	0.2151	0.0963
Spectral	0.2052	0.0971

Next, let's see the result of the comparison of the speed, which was used the data of size $N = \{2^3, \ldots, 2^{16}\}$. In the theory part, we have known that the Lloyd's algorithm is NP-hard, which shares a very high time complexity. Therefore, I only tested this algorithm until 2^6 data points. In addition, after the analysis, it was shown in the **Section 2.2.3 and 2.2.4** that the time complexity of spectral clustering algorithm and power iteration detector is quasilinear. To show this, I add a linear line representing $\mathcal{O}(N)$ as a reference.

From what is shown in the Figure 2, it is easy to see the quasilinear time-complexity of spectral clustering algorithm and power detector algorithm. Combining the comparison of speed, we can see that spectral clustering algorithm is better than Lloyd's algorithm under the stochastic ball model.

3.2 Relax-and-round algorithm

In this section, I will show the results gotten from using the relax-and-round algorithm to cluster the more general data model. Because the figures about the algorithm are all in this section, the denoising process mentioned before is much clearer.

As stated in the **Section 2.1.3**, the matrix X has the form that all the same cluster are shuffled together into a block on the diagonal. Figure 3 is the result optimizer X, so the form of X is more intuitive.

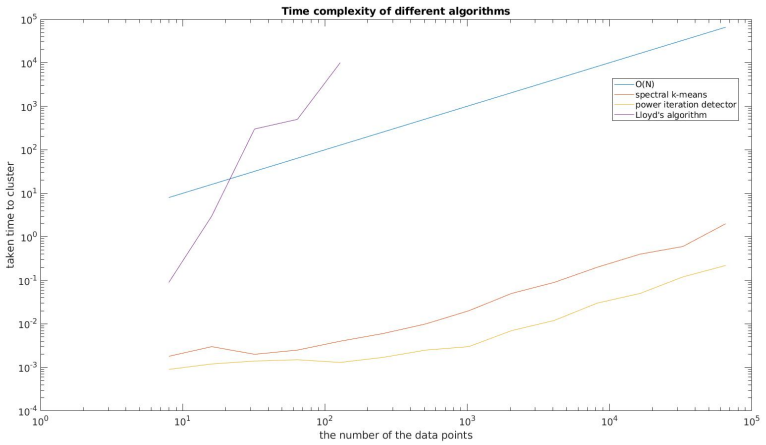

Figure 2: . Time complexity comparison. From the top to the bottom: Lloyd, $\mathcal{O}(N)$, spectral, power detector.

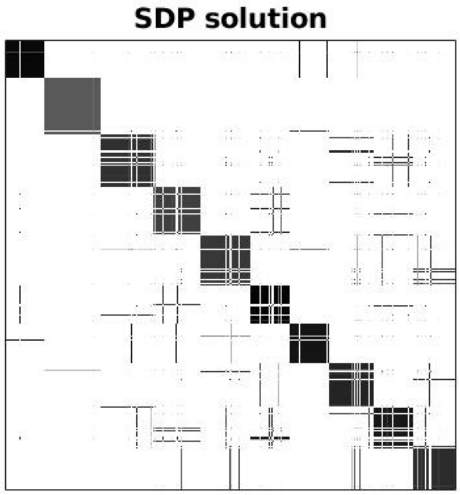

Figure 3: Solution matrix X from solving SDP relaxation, which has the form (18).

Next, recall the interpretation of the **Algorithm 3** given by [2]. With the next figures, it is easier to understand it.

In the first figure in the Figure 4, it is the raw data before solving SDP for subguassian mixture data model. The red points are the true centroids. In the second figure, there are "denoised" version of the original data shown in the first figure. It is clear now that after running SDP in **Algorithm 3**, what remains is only the true centroid and some close neighbors around them. Therefore, to find the true center, step 3 in the algorithm clusters the denoised version of data to find the true centroids, which are red points in the figure. Then depend on

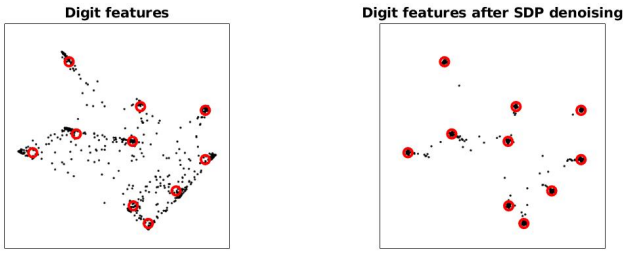

Figure 4: The process of "denoising".

these centers to cluster all the data.

4 Acknowledgements

The author of this report thanks Soledad Villar for giving an intuitive colloquium about her recent works of using SDP to solve K-means problem, which motivated the author to read papers about this interesting field and write this report. The author also thanks professor Michael L.Overton, who gave an entirely amazing and excellent course in this semester, through which the author supplemented a lot of essential mathematical knowledge and learnt a lot for convex and nonsmooth optimization methods, which are very significant for the author's researach interest, machine learning.

References

[1] J.Peng and Y.Wei. Approximating k-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*,18(1):186-205, 2007.

[2] T.Iguchi, D.G.Mixon, J.Peterson, and S.Villar. Probably certifiably correct k-means clustering. *Mathematical Programming, to appear*, 2015.

[3] D.G.Mixon, S.Villar, and R.Ward. Clustering subgaussian mixtures by semidefinite programming. *Information and Inference, to appear*, 2016.

[4] P.Awasthi, A.Bandeira, M.Charikar, K.Ravishankar, S.Villar, and R.Ward. Relax, no need to round: Integrality of clustering formulation. *http://arxiv.org/abs/1408.4045*, 2014.

[5] S.Boyd and L.Vandenberghe. Convex Optimization. 2004.