12-2016

Low rank methods for optimizing clustering

Yangyang Hou
Purdue University

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Yangyang Hou

Entitled

Low Rank Methods for Optimizing Clustering

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

David F. Gleich
Chair

Xavier Tricoche

Alex Pothen

Ahmed Sameh

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): David F. Gleich

Approved by: Sunil Prabhakar / William J. Gorman 12/07/2016

Head of the Departmental Graduate Program Date

LOW RANK METHODS FOR OPTIMIZING CLUSTERING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Yangyang Hou

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2016

Purdue University

West Lafayette, Indiana

For my parents.

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. David Gleich for the continuous support in my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me a lot in all the time of research and writing of this thesis. Beyond that, David is like a friend to his students. It was a great memory to hike in Blue Mountains near Sydney before a conference in the summer of 2015, with David and his family, Laura Bofferding and little Isaac.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Alex Pothen, Prof. Ahmer Sameh, and Prof. Xavier Tricoche, for their insightful discussions and valuable comments.

I would like to thank the research group at Purdue University – Kyle Kloster, Nicole Eikmeier, Nate Veldt, Huda Nassar, Tao Wu, Yanfei Ren, Varun Vasudevan, and Bryan Rainey for their helpful feedback on much of the work that ended up in this thesis. I also want to thank Joyce Jiyoung Whang at Sungkyunkwan University for being a good collaborator. It was a great pleasure to work together, discuss ideas and write papers.

My appreciation also goes to all the staff in the Department of Computer Science for their assistance and patience to the students. I would like to especially thank Dr. William J. Gorman for his efforts on helping format this thesis before the deadline. I also thank my friends and roommates at Purdue for accompanying me during some hard periods and letting me have such a great time in West Lafayette.

Last but not the least, I would like to thank my parents for their love and warm support throughout my Ph.D. study. I dedicate this thesis to my parents.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Hou, Yangyang PhD, Purdue University, December 2016. Low Rank Methods for Optimizing Clustering. Major Professor: David F. Gleich.

Complex optimization models and problems in machine learning often have the majority of information in a low rank subspace. By careful exploitation of these low rank structures in clustering problems, we find new optimization approaches that reduce the memory and computational cost.

We discuss two cases where this arises. First, we consider the NEO-K-Means (Non-Exhaustive, Overlapping K-Means) objective as a way to address overlapping and outliers in an integrated fashion. Optimizing this discrete objective is NP-hard, and even though there is a convex relaxation of the objective, straightforward convex optimization approaches are too expensive for large datasets. We utilize low rank structures in the solution matrix of the convex formulation and use a low-rank factorization of the solution matrix directly as a practical alternative. The resulting optimization problem is non-convex, but has a smaller number of solution variables, and can be locally optimized using an augmented Lagrangian method. In addition, we consider two fast multiplier methods to accelerate the convergence of the augmented Lagrangian scheme: a proximal method of multipliers and an alternating direction method of multipliers. For the proximal augmented Lagrangian, we show a convergence result for the non-convex case with bound-constrained subproblems. When the clustering performance is evaluated on real-world datasets, we show this technique is effective in finding the ground-truth clusters and cohesive overlapping communities in real-world networks.

The second case is where the low-rank structure appears in the objective function. Inspired by low rank matrix completion techniques, we propose a low rank symmetric

matrix completion scheme to approximate a kernel matrix. For the kernel k-means problem, we show empirically that the clustering performance with the approximation is comparable to the full kernel k-means.

1. INTRODUCTION

Machine learning has become extremely important over the past few decades. It appears in many aspects of the industry and daily life, such as Google search engine, recommended items in Amazon, and credit card fraud detection to name a few. There are inseparable ties between machine learning and optimization. The field of optimization seeks methods to find the optimal solution that would minimize or maximize an objective function with constraints. Almost all tasks in machine learning involve deriving a mathematical model for a problem, which often can then be reduced to an optimization problem with complex objective functions in the end. A few examples include kernel k-means clustering, regularized regression problems, tensor optimization, factor analysis, etc. The optimization problems are mathematically well-defined and then can be solved using sophisticated optimization techniques.

Interestingly, these complex objective functions and problems in machine learning often involve a matrix that encodes information combined from both the mathematical model and the particular data. This could either be a solution matrix or a matrix in the objective function. This matrix often has a low rank subspace that captures the majority (or sometimes all) of the impact of the model and data. This is what we'd like to investigate. In this thesis, I present my work on how we can the exploit the low rank subspace in the solution matrix of an optimization problem and the low rank subspace in an objective function. The specific problems studied are clustering tasks. Consequently, we find that low rank structures can benefit complex objective functions and problems through our formulations and new optimization approaches. Overall, these reduce the amount of computational cost substantially.

Clustering is one general task in machine learning to find sets of cohesive objects. Non-exhaustive, overlapping k-means [Whang et al., 2015] (abbreviated NEO-K-Means) is a generalized model of the classical k-means clustering that can detect

overlapping regions between groups as well as outliers supposed not to belong to any groups at the same time. In addition, a weighted and kernelized version of the NEO-K-Means model can be equated to the problem of minimizing an extension of the normalized cut for finding overlapping communities in social and information networks. The NEO-K-Means model falls into an NP-Hard non-convex optimization problem with a complex objective function with constraints. To solve this optimization problem, a simple iterative algorithm that generalizes Lloyd's algorithm [Lloyd, 1982] is proposed in Whang et al. [2015]. It is *fast*, but it tends to get stuck into some regions given poor initialization of the cluster assignments. This motivated us to propose a convex semidefinite program (SDP) relaxation of the NEO-K-Means model in [Hou et al., 2015] for more accurate and reliable clustering results, since the convex problem in optimization field has attractive property – it can be globally optimized in principle. However, the convex formulation is not without problems in real-world applications. When the NEO-K-Means model is relaxed, the number of variables is *quadratic* in the number of data points. Off-the-shelf SDP solvers such as CVX can then only solve problems with fewer 100 data points.

My first work in this thesis focuses on exploring the low rank structures in the NEO-K-Means SDP solutions. We propose a practical method by optimizing a low-rank factorization of the SDP solution matrix directly [Hou et al., 2015]. The resulting optimization problem (called NEO-K-Means LRSDP) is a non-convex, quadratically constrained, bound-constrained problem. When we employ high-quality optimization techniques, e.g., an augmented Lagrangian approach, it frequently generates solutions that are *as good as the global optimal* from convex solvers. This algorithm takes much longer to run but considerably improve the quality and robustness of the clustering procedure.

The empirical success of the augmented Lagrangian framework motivates us to investigate developing faster solvers for the NEO-K-Means low-rank SDP problem. To improve the optimization procedures upon the augmented Lagrangian method more quickly, we consider two *multiplier methods* [Hou et al., 2016]. The first method adds

a proximal regularizer to the augmented Lagrangian method. This general strategy is called either the proximal augmented Lagrangian method (PALM) or the proximal method of multipliers. The second method is an alternating direction method of multipliers (ADMM) strategy by making use of the block structures of the variables in our objective function. Both strategies, when specialized on the NEO-LR problem, have the potential to accelerate the process of finding optimal solutions. We evaluate the two methods on real-world problems where the classical augmented Lagrangian approach takes over an hour of computation time. The proximal augmented Lagrangian strategy tends to run about $3 \sim 6$ times faster, and the ADMM strategy tends to run about $4 \sim 13$ times faster bringing the runtimes of these methods down into range of 5 to 10 minutes – without loss of the quality or performance. We also specialize a general convergence result about the proximal augmented Lagrangian to our algorithm from Pennanen [2002]. The resulting theorem is a general convergence result about the proximal augmented Lagrangian method for non-convex problems with bound-constrained subproblems. The proof involves adapting a few details from Pennanen [2002] to our case.

These augmented Lagrangian based methods for the non-convex NEO-K-Means LRSDP, when using the output of the simple iterative algorithm as the initialization, are able to make further progress on optimizing the objective function. As a result, they tends to achieve better F_1 performance on identifying ground-truth clusters and produce better overlapping communities in real-world networks than the simple iterative algorithm [Hou et al., 2015, 2016].

Despite the low rank structure in the solutions, the objectives often have low rank structures too. One important example is the kernel matrix that exists in many complex objective functions in machine learning, which my second work in thesis studies. The key property of kernel matrices that we hope to exploit is that they are often nearly low-rank for many of the common kernel functions studied. This can be formalized as follows: the singular values of the kernel matrix are localized –

this means that we can preserve the majority of the information in the kernel matrix while rounding small entries of singular values to zero.

Inspired by low rank matrix completion techniques, which predicts the unknown entries of target matrix $Y \in R^{n \times m}$ with the lowest rank given a random subset of observed entries, we hypothesize that a novel means of handling kernels through a low rank symmetric matrix completion framework will provide a means for these techniques to scale to datasets with large scale data points. In addition to computing low rank approximations of kernel matrix for reducing computation complexity, kernel matrix completion has other applications where much information of points is not available. For example, in biology, it is often the case that only a small subset of samples are available for other observed data. As a result, we can only form a partial kernel matrix derived from such data sets.

We apply the approximated kernel matrices on kernel k-means clustering method. The low rank approximation of kernel matrices reduce both the computational complexity and the memory requirements. We also show empirically that the clustering performance is comparable to the full kernel k-means.

All work presented in this thesis has been jointly pursued with my advisor David F. Gleich. The work in Chapter 3 & 4 are joint with Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon.

2. BACKGROUND

We begin with reviews of a few technical preliminaries from different areas. Most are standard techniques and are presented for the completeness of works across the chapters of this thesis.

2.1 Clustering and Non-Exhaustive, Overlapping K-Means

Clustering is one of the most fundamental tasks in machine learning and data mining. Given some data, the clustering problems consist of automatically grouping the data into subsets with a specific strategy. These techniques are not perfect, but they usually provide a starting point that suggest more refined and useful methods in the next stages. There are various strategies to determine the clusters, like connectivity-based clustering (hierarchical clustering), centroid-based clustering, distribution-based clustering, density-based clustering, etc. The clustering model one should use depends on the clustering tasks and there is no generally best algorithm for all.

Here, let's focus on k-means clustering, which lies into the centroid-based clustering. Given a set of data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n\}$, k-means seeks a partition of the data points into k clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ such that they cover all the data points (formally, $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_k = \mathcal{X}$), and the partitions are disjoint ($\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \; \forall i \neq j$). The goal of k-means is to find the clusters that minimize the sum of the distance from the cluster centroid, or the mean of cluster, to each of its assigned data points. The k-means objective can be written as:

$$\min_{\{\mathcal{C}_j\}_{j=1}^{k}} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j} \|\mathbf{x}_i - \mathbf{m}_j\|^2, \text{ where } \mathbf{m}_j = \frac{\sum_{\mathbf{x}_i \in \mathcal{C}_j} \mathbf{x}_i}{|\mathcal{C}_j|}. \tag{2.1}$$

It has been shown that minimizing the above objective function is very computationally diffcult to solve (NP-hard problem) even for just two clusters. However, different algorithms are known to work in different settings. For example, there is an efficient heuristic iterative k-means algorithm [Lloyd, 1982] – also known as Lloyd's algorithm – that repeatedly assign data points to their closest clusters and recompute cluster centroids. This algorithm can monotonically decreases the objective function and converges quickly to a local optimum.

The traditional clustering algorithms usually assign each data point to exactly one cluster. This assignment might be appropriate when clear groups exist in the data and the data do not contain any outliers. However, if the data do not have an obvious separation (this is often the case in the real-world applications) and contain both outliers and large regions of overlap between groups, the traditional disjoint and exhaustive algorithms might fail to get the reasonable patterns of the data. For example, in biological gene expression data, each cluster correspond to a group of genes which are likely to belong to the same functional class. Each gene can serve more than one functions, the clusters are overlapped with each other naturally.

To analyze complex real-world data emerging in many data-centric applications, recently, Whang et al. [2015] proposed a new formulation of this problem called non-exhaustive, overlapping k-means (abbreviated NEO-K-Means) as a generalization of the classical k-means clustering objective, to find overlapping clusters and also detect outliers simultaneously. To put it formally, the goal of non-exhaustive, overlapping clustering is to compute a set of cohesive clusters $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_k$ such that $\mathcal{C}_1 \cup \mathcal{C}_2 \cup ... \cup \mathcal{C}_k \subseteq \mathcal{X}$ and the clusters need not be disjoint (i.e., $\exists i \neq j \ \mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset$).

Before describing NEO-K-Means, we begin with notations. Let $\boldsymbol{U} = [u_{ij}]_{n \times k}$ be an assignment matrix such that $u_{ij} = 1$ if $\mathbf{x}_i$ belongs to cluster j; $u_{ij} = 0$ otherwise. Using this notation, if we seek a traditional disjoint and exhaustive clustering, the number of ones in the assignment matrix $\boldsymbol{U}$ should be always equal to n because each data point should be assigned to exactly one cluster. On the other hand, in a non-exhaustive, overlapping clustering, there are no restrictions on the assignment

matrix $\boldsymbol{U}$; there can be multiple ones in a row, meaning that a data point can belong to multiple clusters. Also, there can be rows of all zeros, meaning that some data points can have no membership in any cluster. Thus, we need to decide how many assignments we will make in $\boldsymbol{U}$. One way to do this is to consider $\boldsymbol{U}^T\boldsymbol{U}$ which is a $k \times k$ matrix whose diagonal entries are equal to cluster sizes. The trace of $\boldsymbol{U}^T\boldsymbol{U}$ is equal to the sum of cluster sizes which is also equal to the total number of assignments in the assignment matrix $\boldsymbol{U}$. To control how many additional assignments we will make in $\boldsymbol{U}$, a constraint is added such that the number of total assignments in $\boldsymbol{U}$ should be equal to $n+\alpha n$ where α controls the amount of overlap among the clusters. We require $0 \leq \alpha \leq (k-1)$ and note that $\alpha \ll (k-1)$ to avoid assigning each data point to every cluster.

By adding the constraint on the total number of assignments in $\boldsymbol{U}$, we are able to control the degree of overlap. Another constraint is added to avoid false positive outliers. Let us define an indicator function $\mathbb{I}\{exp\}$ to be $\mathbb{I}\{exp\} = 1$ if exp is true; 0 otherwise, and let $\mathbf{e}$ denote a $k \times 1$ column vector having all the elements equal to one. Then, the vector $\boldsymbol{U}\mathbf{e}$ denotes the number of clusters to which each data point belongs. Thus, $(\boldsymbol{U}\mathbf{e})_i = 0$ means that $\mathbf{x}_i$ does not belong to any cluster. By setting the upper bound of $\sum_{i=1}^{n} \mathbb{I}\{(\boldsymbol{U}\mathbf{e})_i = 0\}$ to be βn, we can control the non-exhaustiveness – at most βn data points can be considered as outliers. We require $0 \leq \beta n$ and note that $\beta n \ll n$ to make most data points to be assigned to clusters. Specifically, by the definition of "outliers", βn should be a very small number compared to n.

Putting all these together, we are now ready to define the NEO-K-Means objective function as follows:

$$\begin{aligned} &\min_{\boldsymbol{U}} \sum_{j=1}^{k}\sum_{i=1}^{n} u_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|^2, \text{ where } \mathbf{m}_j = \frac{\sum_{i=1}^{n} u_{ij}\mathbf{x}_i}{\sum_{i=1}^{n} u_{ij}} \\ &\text{s.t. } \operatorname{trace}(\boldsymbol{U}^T\boldsymbol{U}) = (1+\alpha)n,\ \textstyle\sum_{i=1}^{n} \mathbb{I}\{(\boldsymbol{U}\mathbf{e})_i = 0\} \leq \beta n. \end{aligned} \tag{2.2}$$

If α=0 and β=0, the NEO-K-Means objective function is equivalent to the standard K-means objective presented in (2.1).

The NEO-K-Means iterative algorithm.

Just like Lloyds method for k-means, there is a simple iterative scheme to monotonically reduce the NEO-K-Means objective. This simple algorithm outperform a variety of other types of overlapping clustering methods including various existing overlapping extension of k-means [Whang et al., 2015]. The NEO-K-Means iterative algorithm is a generalization of the k-means iterative procedure that first makes $(1-\beta)n$ assignments from data points to the nearest cluster centroids to satisfy the non-exhaustiveness condition. Then it makes $(1+\alpha)n - (1-\beta)n$ additional assignments that can overlap based on the smallest distances between data points and centroids. This procedure produces a non-increasing sequence of objective function values.

Graph clustering.

In another line of prior work, Dhillon et al. [2007] showed that optimizing the normalized cut objective is equivalent to a particular weighted, kernel k-means objective. Given a clustering of a graph, the normalized cut of a clustering is the sum of normalized cut scores of each cluster:

$$\text{ncut}(\mathcal{C}) = \sum_{j=1}^{k} \text{ncut}(\mathcal{C}_j) = \sum_{j=1}^{k} \frac{\text{cut}(\mathcal{C}_j)}{\text{links}(\mathcal{C}_j, \mathcal{V})},$$

where $\text{cut}(\mathcal{C}_j)$ is the number of edges leaving the cluster, and $\text{links}(\mathcal{C}_j, \mathcal{V})$ is the sum of degrees for all vertices in $\mathcal{C}_j$; see Dhillon et al. [2007] for more on this objective in the context of k-means. An extended version of this idea holds for the NEO-K-Means problem for normalized cut-based overlapping graph clustering [Whang et al., 2015]. Thus, it is possible to use the NEO-K-Means formulation to produce a set of overlapping clusters on a graph to minimize the sum of normalized cuts. The overlapping graph clustering problem is closely related to community detection, and the iterative NEO-K-Means algorithm achieves state-of-the-art performance at finding ground-truth communities in large networks.

2.2 Semidefinite programming

Semidefinite programs (SDP) are one of the most general classes of tractable convex optimization problems in mathematical programming. There are many fields where SDP is an important numerical tool for analysis, like systems and control theory, combinatorial optimization, etc. The canonical form of SDP is as follows:

$$\begin{aligned} \text{maximize} \quad & \text{trace}(\boldsymbol{C}\boldsymbol{X}) \\ \text{subject to} \quad & \boldsymbol{X} \succeq 0, \boldsymbol{X} = \boldsymbol{X}^T, \\ & \text{trace}(\boldsymbol{A}_i\boldsymbol{X}) = b_i, \quad i = 1, \ldots, m \end{aligned} \tag{2.3}$$

Notice that in an SDP the variable is the symmetric positive semi-definite matrix $\boldsymbol{X}$. The data for SDP consists of the symmetric matrix $\boldsymbol{C}$ for the objective function, and the m symmetric matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$ with the $m-$vector $\mathbf{b}$ which form the m linear equations. SDP can be solvable via interior-point methods. There are a variety of solvers for optimizing canonical SDP such as CVX [Grant and Boyd, 2008, 2014], SDPNAL+ [Yang et al., 2015].

In practice, there are many problems with SDP form relaxations where the goal is a low-rank solution, then we can directly solve for low-rank factorization of the semidefinite solution matrix. The low-rank variation of SDP is as follows:

$$\begin{aligned} \text{maximize} \quad & \text{trace}(\boldsymbol{C}\boldsymbol{Y}\boldsymbol{Y}^T) \\ \text{subject to} \quad & \boldsymbol{Y} : n \times k, \\ & \text{trace}(\boldsymbol{A}_i\boldsymbol{Y}\boldsymbol{Y}^T) = b_i, \quad i = 1, \ldots, m \end{aligned} \tag{2.4}$$

Notice the low-rank form drops the positive semidefinite ($\boldsymbol{X} \succeq 0$) and symmetry constraints ($\boldsymbol{X} = \boldsymbol{X}^T$) but replaces $\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{Y}^T$, which automatically satisfies these constraints. Low-rank SDP factorizations are non-linear and non-convex, and can be locally optimized via nonlinear programming techniques such as an augmented Lagrangian method [Burer and Monteiro, 2003].

2.3 Kernel functions and matrices

In machine learning, theory and algorithms have been very well studied for the linear relations. However, real world data analysis problems usually requires exploring the nonlinear relations between data. One possible way is to map the data from the original input space to a high-dimensional feature space, then do the well-developed linear analysis in that space. If the map is chosen suitably, complex nonlinear relations in the original input space can be simplified and easily detected in the feature space.

For input data $\mathbf{x}_1$ and $\mathbf{x}_2$, a kernel function can be expressed as an inner product in the feature space and is usually denoted as:

$$\mathcal{K}(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle$$

By using a positive definite kernel function, without knowing the coordinates of the data in the feature space, we can simply compute the inner products between all pairs of data in the feature space. Choosing $\mathcal{K}$ is equivalent to choosing the mapping function ϕ. That means, we never need to know the mapping functions explicitly. This is often called "kernel trick". Kernel functions must be continuous, symmetric and should have a positive semi-denite Gram matrix. The choices of kernel functions consist of polynomial kernels, Gaussian kernels, Laplacian kernels, etc. Choosing the most appropriate kernels generally depends on the problems.

Utilizing positive definite kernels are rather popular in machine learning over the last two decades, with applications in image recognition, text analysis, network analysis, and so on. Here, we can also extend NEO-K-Means (2.2) to a weighted kernel version by introducing a nonlinear mapping ϕ and a nonnegative weight w_i for each data point $\mathbf{x}_i$ as follows:

$$\begin{aligned} \min_{\boldsymbol{U}} \quad & \sum_{j=1}^{k} \sum_{i=1}^{n} u_{ij} w_i \|\phi(\mathbf{x}_i) - \mathbf{m}_j\|^2, \text{ where } \mathbf{m}_j = \frac{\sum_{i=1}^{n} u_{ij} w_i \phi(\mathbf{x}_i)}{\sum_{i=1}^{n} u_{ij} w_i} \\ \text{s.t.} \quad & \operatorname{trace}(\boldsymbol{U}^T \boldsymbol{U}) = (1+\alpha)n, \ \sum_{i=1}^{n} \mathbb{I}\{(\boldsymbol{U}\mathbf{e})_i = 0\} \leq \beta n. \end{aligned} \tag{2.5}$$

In (2.5), the nonlinear mapping enables us to cluster the data points in a higher dimensional feature space. We can avoid forming the feature space explicitly by using the well-known kernel trick. Let $\boldsymbol{K}$ denote a kernel matrix such that $\boldsymbol{K}_{ij} = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. On the other hand, the weight for each data point differentiates each data point's contribution to the objective function.

3. NEO K-MEANS SDP RELAXATION

The iterative procedure for NEO-K-Means is fast, but has the same classic problem as iterative algorithms for k-means. It gets trapped in the local areas given a poor guess initialization of the clusters. This issue is often addressed by using multiples runs with random initialization or using distance based initialization strategies [Arthur and Vassilvitskii, 2007] and then choosing the best cluster we get. However, we still don't know the quality of our solutions: How good is the cluster? Can we get a better cluster if we already have reasonable good clusters? To answer these questions, in this chapter, we continue the study of the non-exhaustive, overlapping cluster objective function by proposing a convex SDP relaxation to produce more accurate and reliable clustering.

SDP has wide applications in combinatorial optimization. A number of NP-hard combinatorial optimization problems have convex relaxations that are semidefinite programs. Examples of the use of SDP in combinatorial optimization are SDP relaxation of the Max-Cut problem, SDP relaxation of k-means problem [Lindsten et al., 2011], etc. One of the fascinating properties of the convex problem is that it can be globally optimized in time and memory which is polynomial in the input size.

Some materials presented in this chapter have been published in Hou et al. [2015].

3.1 SDP formulation

We begin by stating an exact SDP-like program for the weighted kernel NEO-K-Means objective defined in (2.5) and then describe how to relax it to an SDP. We use the same notation as the previous chapter and summarize our common notation

Table 3.1.: A summary of the notation used in the NEO-K-Means problem, the final assignment, and the SDP and low-rank approximations.

$\mathbf{x}_i$	the data points for k-means
k	the number of clusters
α	the overlap parameter (0 means no overlap)
β	the outlier parameter (0 means no outliers)
$\boldsymbol{U}$	the assignment matrix for a solution
$\boldsymbol{Z}$	the co-occurence matrix for the SDP relaxation
$\boldsymbol{K}$	the kernel matrix for NEO-K-Means
$\boldsymbol{W}$	a diagonal weight matrix for weighted problems
$\mathbf{d}$	a specialized weight vector for the SDP relaxation
$\mathbf{f}$	the cluster count variable for the SDP relaxation
$\mathbf{g}$	the outlier indicator for the SDP relaxation
$\boldsymbol{Y}$	the low-rank approximation of $\boldsymbol{Z}$ in NEO LRSDP

in Table 3.1. The essential idea with the SDP-like version is that we replace the assignment matrix $\boldsymbol{U}$ with a normalized cluster co-occurrence matrix $\boldsymbol{Z}$:

$$\boldsymbol{Z} = \sum_{j=1}^{k} \frac{\boldsymbol{W}\mathbf{u}_j(\boldsymbol{W}\mathbf{u}_j)^T}{s_j} \tag{3.1}$$

where $\boldsymbol{W}$ is a diagonal matrix with the data point weights w_i on the diagonal, $\mathbf{u}_j$ is the j-th column of matrix $\boldsymbol{U}$ and $s_j = \mathbf{u}_j^T \boldsymbol{W}\mathbf{u}_j$. When $\boldsymbol{Z}$ is defined from an assignment matrix $\boldsymbol{U}$, then values of Z_{ij} are non-zero when items co-occur in a cluster. With appropriate constraints on the matrix $\boldsymbol{Z}$, it serves as a direct replacement for the assignment matrix $\boldsymbol{U}$.

To state the problem, let $\boldsymbol{K}$ denote the kernel matrix of the data points, e.g., if $\boldsymbol{X}$ is the data matrix whose rows correspond to data vectors, then $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T$ is just the simple linear kernel matrix. Let $\mathbf{d}$ be a vector where $d_i = w_i K_{ii}$, i.e., a weighted diagonal from $\boldsymbol{K}$. We need two important new types of variables as well:

- Let $\mathbf{f}$ denote a vector of length n such that the ith entry indicates the number of clusters the data point i belongs to.

- Let $\mathbf{g}$ denote a vector of length n such that the ith entry is one if the data point i belongs to any clusters, and zero if the data point does not belong to any cluster.

Finally, we denote by $\mathbf{e}$ the vector of all 1s.

With new variables, we can rewrite the objective of (2.5) as follows:

$$\begin{aligned}
&\min_{\boldsymbol{U}} \quad \sum_{j=1}^{k}\sum_{i=1}^{n} u_{ij} w_i \|\phi(\mathbf{x}_i) - \mathbf{m}_j\|^2 \\
=&\min_{\boldsymbol{U}} \quad \sum_{j=1}^{k}(\sum_{i=1}^{n} u_{ij} w_i K_{ii} - \frac{\mathbf{u}_j^T \boldsymbol{W}\boldsymbol{K}\mathbf{u}_j}{\mathbf{u}_j^T \boldsymbol{W}\mathbf{u}}) \\
=&\min_{\boldsymbol{U}} \quad \sum_{j=1}^{k}\sum_{i=1}^{n} u_{ij} w_i K_{ii} - \sum_{j=1}^{k} \frac{\mathbf{u}_j^T \boldsymbol{W}\boldsymbol{K}\mathbf{u}_j}{\mathbf{u}_j^T \boldsymbol{W}\mathbf{u}} \\
=&\min_{\boldsymbol{Z},\mathbf{f}} \quad \mathbf{f}^T\mathbf{d} - \operatorname{trace}(\boldsymbol{K}\boldsymbol{Z})
\end{aligned}$$

Thus we get the following program which is equivalent to the weighted kernel NEO-K-Means objective (2.5) with a discrete assignment matrix:

$$\begin{aligned}
\underset{\boldsymbol{Z},\mathbf{f},\mathbf{g}}{\text{minimize}} \quad & \mathbf{f}^T\mathbf{d} - \operatorname{trace}(\boldsymbol{K}\boldsymbol{Z}) & \\
\text{subject to} \quad & \operatorname{trace}(\boldsymbol{W}^{-1}\boldsymbol{Z}) = k, & (a) \\
& Z_{ij} \geq 0, & (b) \\
& \boldsymbol{Z} \succeq 0, \boldsymbol{Z} = \boldsymbol{Z}^T & (c) \\
& \boldsymbol{Z}\mathbf{e} = \boldsymbol{W}\mathbf{f}, & (d) \\
& \mathbf{e}^T\mathbf{f} = (1+\alpha)n, & (e) \\
& \mathbf{e}^T\mathbf{g} \geq (1-\beta)n, & (f) \\
& \mathbf{f} \geq \mathbf{g}, & (g) \\
& \operatorname{rank}(\boldsymbol{Z}) = k, & (h) \\
& \mathbf{f} \in \mathcal{Z}_{\geq 0}^n, \mathbf{g} \in \{0,1\}^n. & (i)
\end{aligned} \tag{3.2}$$

Constraints $(a), (b), (c)$, and (h) encode the fact that $\boldsymbol{Z}$ must arise from an assignment matrix. Constraints $(d), (e), (f), (g)$ and (i) express the amount of overlap and

non-exhaustiveness in the solution. This is a mixed-integer, rank constrained SDP. As such, it is combinatorially hard to optimize just like the original NEO-K-Means objective.

The constraints that make this a combinatorial problem are (h) and (i). If we ignore the rank constraint (h) and relax integer constraint (i) to the continuous constraint:

$$\begin{aligned}
\underset{\boldsymbol{Z},\mathbf{f},\mathbf{g}}{\text{minimize}} \quad & \mathbf{f}^T\mathbf{d} - \text{trace}(\boldsymbol{K}\boldsymbol{Z}) \\
\text{subject to} \quad & \text{trace}(\boldsymbol{W}^{-1}\boldsymbol{Z}) = k, && (a) \\
& Z_{ij} \geq 0, && (b) \\
& \boldsymbol{Z} \succeq 0, \boldsymbol{Z} = \boldsymbol{Z}^T && (c) \\
& \boldsymbol{Z}\mathbf{e} = \boldsymbol{W}\mathbf{f}, && (d) \\
& \mathbf{e}^T\mathbf{f} = (1+\alpha)n, && (e) \\
& \mathbf{e}^T\mathbf{g} \geq (1-\beta)n, && (f) \\
& \mathbf{f} \geq \mathbf{g}, && (g) \\
& 0 \leq \mathbf{f} \leq k, 0 \leq \mathbf{g} \leq 1
\end{aligned} \tag{3.3}$$

then we arrive at a convex problem. Thus, any local optimal solution of (3.3) must be a global solution. As long as we can solve (3.3) successfully, we get the lower bound for the (2.5) and we can say there would be no clusters can achieve better objective values than the lower bound. The gap between the optimal value of (3.3) and the objective value of (2.5) for some cluster assignment is a measure (the smaller, the better) to test the quality of clustering results.

Solving (3.3) requires a black-box SDP solver such as CVX [Grant and Boyd, 2008, 2014]. As we convert the NEO-K-Means objective into a standard form of SDP, the number of variables becomes $\mathcal{O}(n^2)$ and the resulting complexity is worse than $\mathcal{O}(n^3)$ in most cases, and can be as bad as $\mathcal{O}(n^6)$. In the next Section 3.2, we examine the scalability of several well-known SDP solvers.

3.2 Scalability of SDP solvers

We test several well-known optimization solvers such as MOSEK, SDPT3, SEDUMI (these three solvers are included in CVX), and SDPNAL+ [Yang et al., 2015] to solve the problem (3.3). We generate synthetic datasets for experiments using the Gaussian distribution. We assume that there exists two clusters as well as overlap between the clusters. We then add five outliers to the data. We'd like to identify both overlapping regions and outliers simultaneously by NEO-K-Means objective. One simple example with a total number of 55 data points (including five outliers) is shown in Figure 3.1.

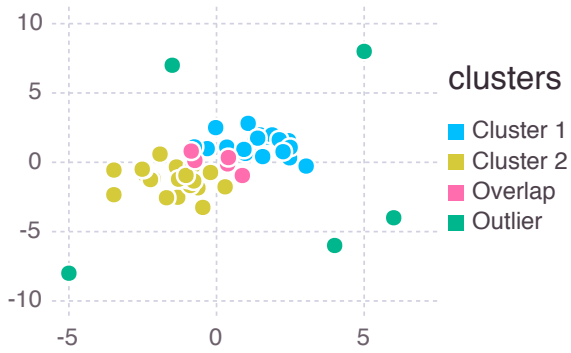


Fig. 3.1.: A simple example of synthetic data with 55 data points for clustering with overlapping regions and five outliers.

In the experiment, we measure the running time of four SDP solvers to solve (3.3) on synthetic datasets as we increase the numbers of data points (excluding five outliers) from 30 to 1000. The results is presented in Figure 3.2. We see that the solvers in CVX (including CVX-SEDUMI, CVX-SDPT3 and CVX-MOSEK) seem to be reasonable only for problems with fewer than 100 data points. Among the SDP solvers, SDPNAL+ is the fastest method, but it is still limited to solve less than five thousands data points. We conclude that although the SDP formulation enjoys attractive theoretical properties with respect to global optimization, it is currently unsuitable for large problem sizes.

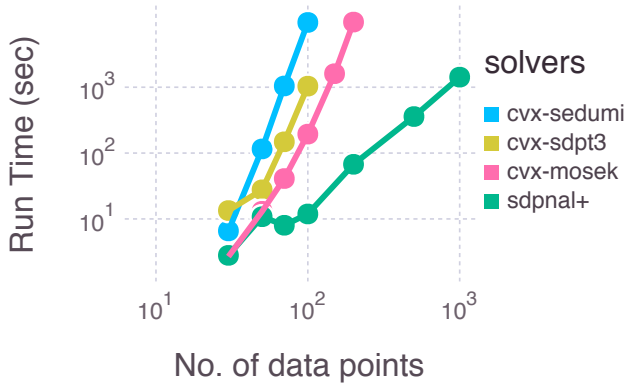


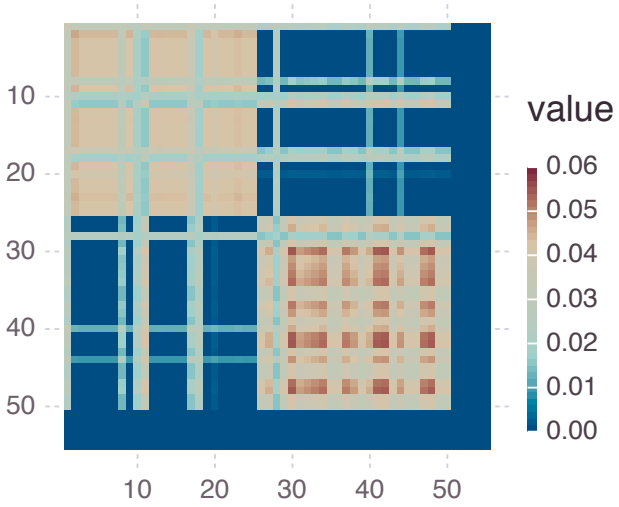
Fig. 3.2.: Run times of four SDP solvers on synthetic datasets when increasing the total numbers of data points.

3.3 Rank of SDP solutions

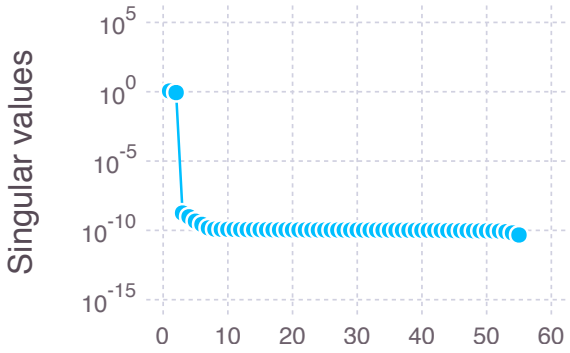
Now we would like to examine the rank of the solution matrix of (3.3). Here we present the co-occurrence solution matrix $\boldsymbol{Z}$ of the synthetic data in Figure 3.1 we get from NEO-K-Means convex semidefinite program relaxation in Figure 3.3. It clear that the solution matrix $\boldsymbol{Z}$ exhibits the low rank structure with two main diagonal blocks. Moreover, after computing its singular values, it is easy to see that $\boldsymbol{Z}$ has the rank of two, i.e., the number of clusters.

This low rank structure is not surprising. Since we formulate the co-occurrence matrix $\boldsymbol{Z}$ according to (3.1), we know that the rank of the matrix $\boldsymbol{Z}$ should be k. We then conduct more principle experiments to investigate the rank of the co-occurrence matrix on synthetic datasets.

As we change the distance between the cluster centroids and the variance level of each cluster, we compute the rank of the co-occurrence solution matrix. In Figure 3.4, we present the rank of the solution matrix in gray scale. Black indicates that the rank of $\boldsymbol{Z}$ is two, and the lighter the gray color is, the larger the rank is. We see that when the two clusters are reasonably separable from each other (i.e., larger distance between



(a) The co-occurrence matrix $\boldsymbol{Z}$



(b) Singular values of $\boldsymbol{Z}$

Fig. 3.3.: Consider the synthetic data in Figure 3.1 with overlapping regions and outliers for clustering. At top, we show the co-occurrence matrix $\boldsymbol{Z}$ from NEO-K-Means SDP objective. At bottom we show the singular values of $\boldsymbol{Z}$. It is easy to see that $\boldsymbol{Z}$ has low rank structure and its rank is two, i.e. the number of clusters.

the cluster centroids or smaller variance level), the rank of the solution matrix becomes close to two, which is the number of clusters.

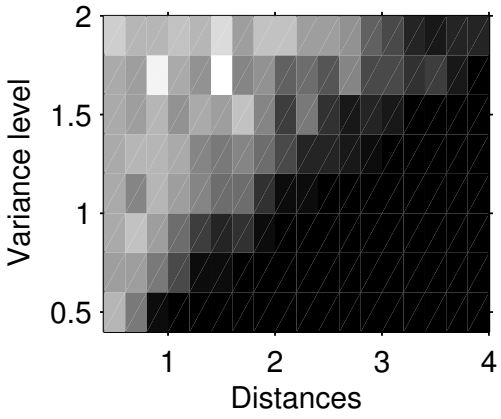


Fig. 3.4.: The rank of the co-occurrence matrix $\boldsymbol{Z}$ is represented in gray scale. We examine the rank of $\boldsymbol{Z}$ by varying the distance between two cluster centroids and the variance level of each cluster. If clusters are reasonably separable from each other, the rank of $\boldsymbol{Z}$ is equal to the number of clusters.

3.4 Rounding procedures

Once we get a real-valued co-occurrence matrix $\boldsymbol{Z}$, we need to convert it into a binary assignment matrix $\boldsymbol{U}$ through a rounding procedure. From the definition of $\boldsymbol{Z}$ and experimental result in the previous Section 3.3, we notice that the information of matrix $\boldsymbol{Z}$ lies in the subspace of rank k. Therefore, we can get a low-rank approximation of $\boldsymbol{Z}$ by factorizing $\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{Y}^T$ as we apply a symmetric nonnegative matrix factorization such as [Vandaele et al., 2015] where $\boldsymbol{Y}$ is an $n \times k$ non-negative matrix.

The resulting matrix $\boldsymbol{Y}$ can be regarded as the normalized assignment matrix

$$\boldsymbol{Y} = \boldsymbol{W}\hat{\boldsymbol{U}} \tag{3.4}$$

where $\hat{\boldsymbol{U}} = [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_k]$, and $\hat{\mathbf{u}}_j = \mathbf{u}_j / \sqrt{s_j}$ for any $j = 1, \ldots, k$.

After the $\boldsymbol{Z}$ matrix is represented by a low-rank matrix $\boldsymbol{Y}$, we can convert $\boldsymbol{Y}$ to a binary assignment matrix $\boldsymbol{U}$ by a rounding procedure. One way to get $\boldsymbol{U}$ from $\boldsymbol{Y}$ is to select the top $(1+\alpha)n$ entries in $\boldsymbol{W}^{-1}\boldsymbol{Y}$ as the clustering assignment. The other way is based on the observation that both the vectors $\mathbf{f}$ and $\mathbf{g}$ provide important information about the solution. Namely, $\mathbf{f}$ gives us a good approximation

to the number of clusters each data point is assigned to, and $\mathbf{g}$ indicates which data points are not assigned to any cluster. By utilizing these information, we propose Algorithm 1 to derive $\boldsymbol{U}$ from $\boldsymbol{Y}$. It uses the largest $n - \beta n$ entries of the vector $\mathbf{g}$ to determine the set of nodes to assign first. Each data point i is assigned to $\lfloor f_i \rfloor$ clusters based on the values in the ith row of $\boldsymbol{Y}$. The remaining assignments are all based on the largest residual elements in $\mathbf{f} - \lfloor \mathbf{f} \rfloor$.

Algorithm 1 Rounding $\boldsymbol{Y}$ to a binary matrix $\boldsymbol{U}$

Input: $\boldsymbol{Y}$, $\boldsymbol{W}$, $\mathbf{f}$, $\mathbf{g}$, α, β
Output: $\boldsymbol{U}$

Update $\boldsymbol{Y} = \boldsymbol{W}^{-1}\boldsymbol{Y}$.
Set $\mathcal{D}$ to be the largest $(n - \beta n)$ coordinates of $\mathbf{g}$.
for each entry i in $\mathcal{D}$ **do**
    Set $\mathcal{S}$ to be the top $\lfloor f_i \rfloor$ entries in $Y(i, :)$.
    Set $U(i, \mathcal{S}) = 1$.
end for
Set $\bar{\mathbf{f}} = \mathbf{f} - \lfloor \mathbf{f} \rfloor$.
Set $\mathcal{R}$ to be the largest entries in $\bar{\mathbf{f}}$.
for each entry i in $\mathcal{R}$ **do**
    Pick a cluster ℓ where $Y(i, \ell)$ is the maximun over all clusters where i is not currently assigned.
    Set $U(i, \ell) = 1$.
end for

4. LOW RANK STRUCTURE IN NEO K-MEANS SDP

The convex formulation of NEO-K-Means is not without problems. When the NEO-K-Means problem is relaxed to a convex semidefinite program (SDP), the number of variables is quadratic in the number of data points. Off-the-shelf SDP solvers such as CVX can then solve problems with fewer than 100 data points. Among the SDP solvers we investigate, SDPNAL+ is the fastest method, but it is still limited to solve at most five thousands data points. Even small modern dataset have a few thousand points, and they require a different approach.

The low rank structure of the solution matrix of convex formulation of NEO-K-Means enables us to consider a low-rank factorization of the SDP solution matrix directly, which allows us to deal with large-scale SDPs and implement a solution procedure using an augmented Lagrangian method. We discuss this low-rank SDP (LRSDP) approach for NEO-K-Means in this chapter. It is a standard technique to tackle large-scale SDPs [Burer and Monteiro, 2003]. The resulting optimization problem is a quadratically constrained problem with a quadratic objective that can no longer be globally optimized. An augmented Lagrangian procedure, for instance, will only converge to a local minimizer. Nevertheless, when this approach has been used in the past with high-quality optimization methods, it frequently generates solutions that are *as good as the global optimal* from convex solvers [Burer and Monteiro, 2003], a fact which has some theoretical justification [Burer and Monteiro, 2005]. Furthermore, similar ideas yielded stability improvements to convex relaxations of the classic k-means objective [Kulis et al., 2007].

Some materials presented in this chapter have been published in Hou et al. [2015, 2016].

4.1 LRSDP formulation

In the SDP formulation of the NEO-K-Means objective (3.3), the normalized co-occurrence solution matrix $\boldsymbol{Z}$ should only be rank k where k is the number of clusters we set as the input. Thus, we can represent $\boldsymbol{Z}$ as $\boldsymbol{Y}\boldsymbol{Y}^T$ via applying a low-rank factorization idea where $\boldsymbol{Y}$ is an $n \times k$ non-negative matrix. Then we arrive at the following low-rank SDP optimization problem for (3.3). We have chosen to write it in the standard form of a minimization problem with explicit slack variables $\mathbf{s}, r$ to convert the inequality constraints into equality and bound constraints.

$$
\begin{aligned}
\underset{\boldsymbol{Y},\mathbf{f},\mathbf{g},\mathbf{s},r}{\text{minimize}} \quad & \mathbf{f}^T\mathbf{d} - \text{trace}(\boldsymbol{Y}^T\boldsymbol{K}\boldsymbol{Y}) & \\
\text{subject to} \quad & k = \text{trace}(\boldsymbol{Y}^T\boldsymbol{W}^{-1}\boldsymbol{Y}) & (s) \\
& 0 = \boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}\mathbf{f} & (t) \\
& 0 = \mathbf{e}^T\mathbf{f} - (1+\alpha)n & (u) \\
& 0 = \mathbf{f} - \mathbf{g} - \mathbf{s} & (v) \\
& 0 = \mathbf{e}^T\mathbf{g} - (1-\beta)n - r & (w) \\
& Y_{ij} \geq 0, \mathbf{s} \geq 0, r \geq 0 & \\
& 0 \leq \mathbf{f} \leq k, 0 \leq \mathbf{g} \leq 1 &
\end{aligned}
\tag{4.1}
$$

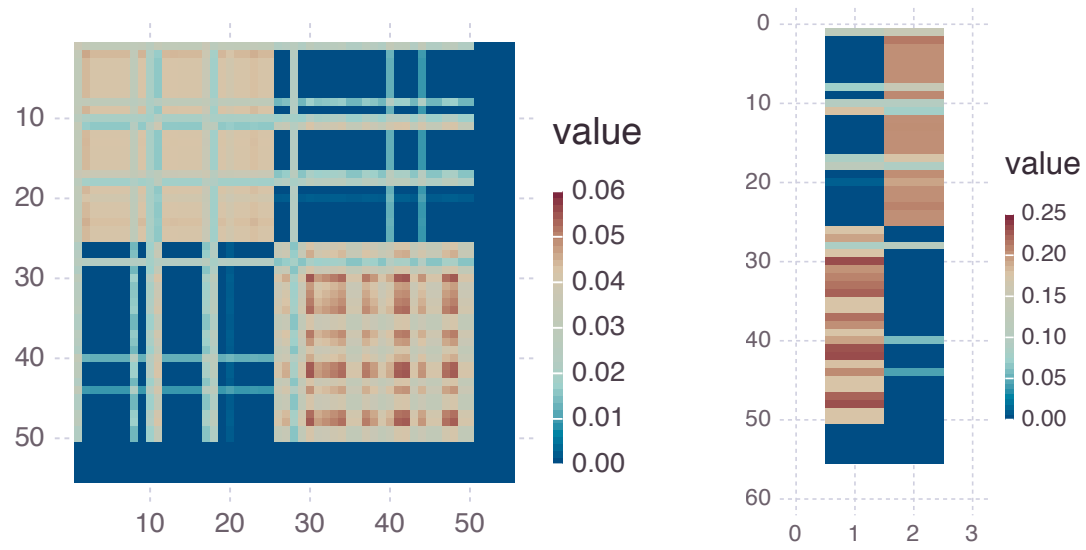
We replaced the constraint $\boldsymbol{Y}\boldsymbol{Y}^T \geq 0$ with the stronger constraint $\boldsymbol{Y} \geq 0$. This problem is a quadratic programming problem with quadratic constraints. Even though now we lose convexity by formulating the low rank SDP, this nonlinear programming problem only requires $\mathcal{O}(nk)$ memory and existing nonlinear programming techniques allow us to scale to large problems.

When we get a solution of (4.1) by some nonlinear optimization techniques, the solution $\boldsymbol{Y}$ can also be directly regarded as the normalized assignment matrix:

$$
\boldsymbol{Y} = \boldsymbol{W}\hat{\boldsymbol{U}} \tag{4.2}
$$

where $\hat{\boldsymbol{U}} = [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_k]$, and $\hat{\mathbf{u}}_j = \mathbf{u}_j / \sqrt{s_j}$ for any $j = 1, \ldots, k$.

Rounding procedures. Solutions from the the NEO K-Means LRSDP problem are real-valued. We need to convert $\boldsymbol{Y}$ into a binary assignment matrix $\boldsymbol{U}$ through a rounding procedure. We can still use the same rounding procedures discussed in Section 3.4 since we already have the low rank factorization $\boldsymbol{Y}$.



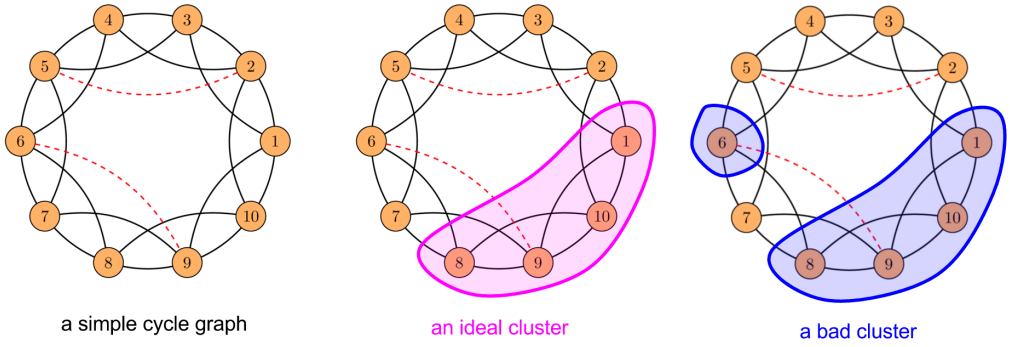
(a) The normalized co-occurrence matrix $\boldsymbol{Z}$

(b) The normalized assignment $\boldsymbol{Y}$

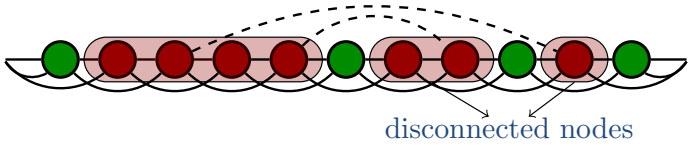
Fig. 4.1.: Consider the synthetic data in Figure 3.1 with overlapping regions and outliers for clustering. At left, we show the co-occurrence solution matrix $\boldsymbol{Z}$ from NEO-K-Means SDP objective. At right, we show the assignment solution matrix $\boldsymbol{Y}$ from NEO-K-Mean LRSDP objective.

To see the effectiveness of the new low rank SDP procedure, let's consider the synthetic data again in Figure 3.1. Now we can get directly the assignment matrix $\boldsymbol{Y}$ after solving (4.1). See Figure 4.1. It is easy to see $\boldsymbol{Y}$ exhibits two clusters with some overlapping assignments as well as no assignments for outliers. The difference between matrix $\boldsymbol{Z}$ and $\boldsymbol{Y}\boldsymbol{Y}^T$ is very small ($\|\boldsymbol{Z} - \boldsymbol{Y}\boldsymbol{Y}^T\|_F = 2.3290 \times 10^{-4}$), which means we can directly get the low rank factorization of $\boldsymbol{Z}$ with high accuracy.

Our goal with the new low rank SDP procedure is to produce more accurate and reliable clusterings than the previous iterative algorithm [Whang et al., 2015] in the



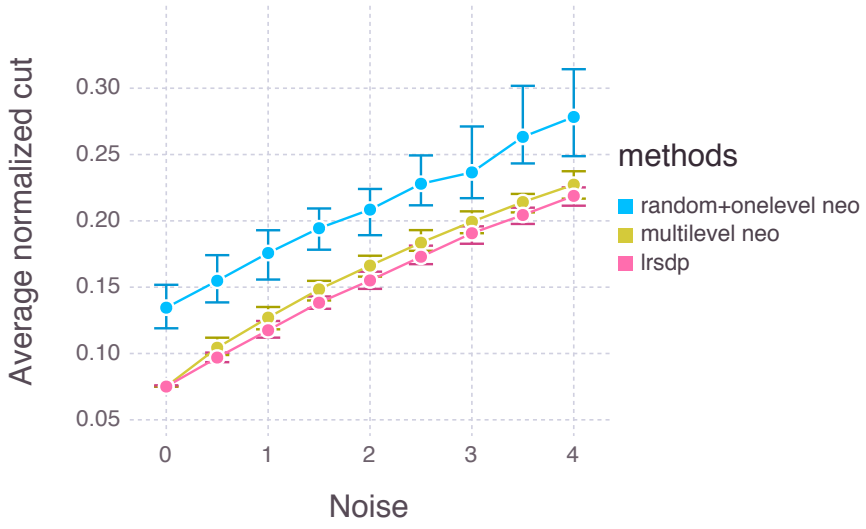
(a) Cycle graph with an ideal cluster and a bad cluster



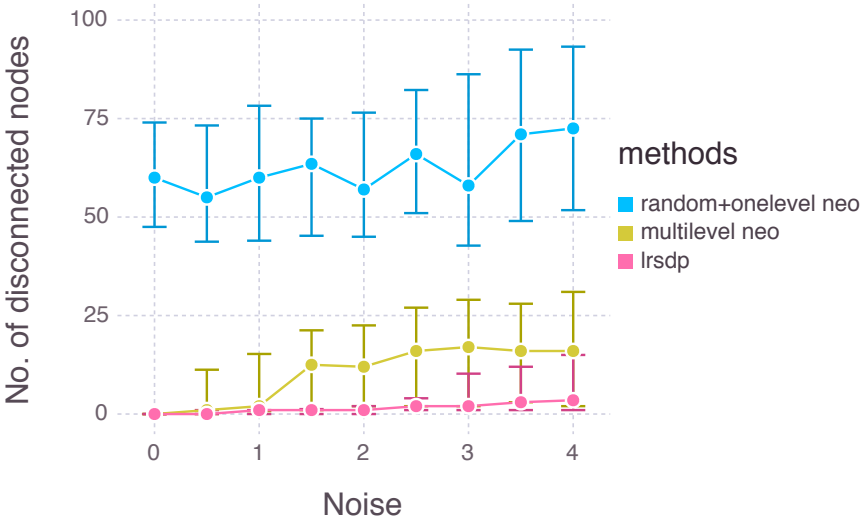
(b) The disconnected nodes error measure counts number of nodes that are disconnected from the largest connected component. These nodes are illustrated for the cluster in red. (Green nodes are not in that cluster.)

Fig. 4.2.: A synthetic study of overlapping community detection on a Watts-Strogatz cycle graph where each point should be assigned to two clusters: (a) A simple cycle graph with 10 nodes and examples showing what is an ideal cluster or bad cluster for the cycle; (b) an illustration of a portion of the cycle with dashed 'noise' edges and showing the disconnected points measure (which is 3).

regime of medium-scale problems. This regime is ideal because the new method is more computationally expensive than the iterative algorithm, which was an efficient procedure designed for problems with millions of data points. To see the difference between these methods, we study the behavior on a synthetic problem with community detection on a cycle graph. The graph is a Watts-Strogatz random graph where each node has edges to five neighbors on each side of the cycle. We also add random edges based on an Erdős-Rényi graph with expected degree d, which we consider as noise edges. See Figure 4.2(a) for a simple example cycle graph. When the noise is low, clusterings should respect the cycle structure and be continuous, connected regions. Hence, we compute an error measure for each cluster based on the number



(a) Avg. normalized cut



(b) No. of disconnected points

Fig. 4.3.: A synthetic study of overlapping community detection on a Watts-Strogatz cycle graph where each point should be assigned to two clusters: (a) & (b) the results of normalized cut and the number of disconnected points on graphs with 100 nodes returned by our new LRSDP procedure compared with two variations of the previous "neo" iterative algorithms.

of points disconnected from the largest connected component in the cycle; this measure is illustrated in Figure 4.2(b). We compare three methods: the straight-forward iterative NEO-K-Means method with random initialization, a multilevel variation on that method [Whang et al., 2015], and our LRSDP with random initialization. We run 100 trials and plot the the median, 25th and 75th percentiles of the normalized cut scores and the number of disconnected nodes by varying the noise level. Figure 4.3(a) & Figure 4.3(b) show the results. Our LRSDP method achieves the best performance in terms of both the normalized cut and the number of disconnected nodes. We observe that our LRSDP method often produces 0 disconnected points even as the noise increases whereas the faster iterative method starts to introduce many disconnected points with only a modest amount of error.

4.2 Solving NEO K-Means low-rank SDP

In this section, we will discuss how to solve the optimization problem (4.1). Because the low rank SDP of NEO K-Means (4.1) is a non-convex optimization problem, there is no guarantees that our methods would find the global solution, or even converge. Even if our methods do not find the global solution in theory, it usually finds a feasible point with reasonable and acceptable objective values. The experiments we try also find that in many cases, our methods can find global solutions. Our methods are all in the augmented Lagrangian framework, which aims for nonlinear constrained optimization problems.

Table 4.1.: Comparison of our solver ALM and SNOPT solver from NEOS Server.

	our solver ALM (obj/time)	SNOPT solver (obj/time)
music	79514.130/92s	79515.156/306s
scene	18534.030/3,798s	18534.021/8,910s
yeast	8902.253/4,331s	Not solved

To see the effectiveness of our methods, we compare them with commercial optimization solvers on NEOS Server (`https://neos-server.org/neos`) to solve the problem (4.1). The NEOS Server is a free internet-based service for solving numerical optimization problems. The details of datasets we use are available in Section 4.5.3. We run our solver with the classical augmented Lagrangian approach (denoted by "ALM", see details in Section 4.2.1) with single thread on a computer with 256GB of RAM and two 8-core 2.66Ghz Xeon. To use the solvers on NEOS Server, we submit the problems in AMPL input files. The hardware specs for the NEOS Server machine is 64GB of RAM and two 6-core 2.8Ghz Xeon. The result is reported in Table 4.1. We can see "ALM" is much faster than theirs (e.g., SNOPT which is suitable for large nonlinearly constrained problems with a modest number of degrees of freedom) while obtaining the similar solutions. And for "yeast" dataset, we didn't get the result from SNOPT solver since it reached the 8-hour time limits of online version.

We also show the scalability for our methods. First, in Figure 4.4, we compare the runtimes of the fastest SDP solver, SDPNAL+, and our proposed solvers (denoted by "lrsdp-ALM", "lrsdp-PALM"and "lrsdp-ADMM"; we will discuss in details in Section 4.2.1 and 4.2.2) for NEO-K-Means low rank SDP objective on synthetic datasets when increasing the number of data points and we set the number of clusters fixed to two. We see that the low rank SDP solvers are much faster than the SDP solvers. This observation allows us to tackle larger problems than SDP relaxation through the new procedure.

We then investigate the time complexity of our proposed solver (denoted by "ALM", see details in Section 4.2.1) when increasing the number of clusters on a synthetic problem with community detection on a big-block graph. The graph generated includes k blocks of the same size with the dense internal connections within blocks and random noise edges between blocks. We also add more connections from Block One to other blocks respectively to mimic the overlapping property. In Figure 4.5, we see the run times looks quadratic in the number of clusters. Since the

number of nodes is linear with the number of clusters for the big-block graph, we can say the time complexity of "ALM" is $O(nk)$.

In the next, we will describe our methods to solve NEO-K-Means LRSDP problem 4.1 in details.

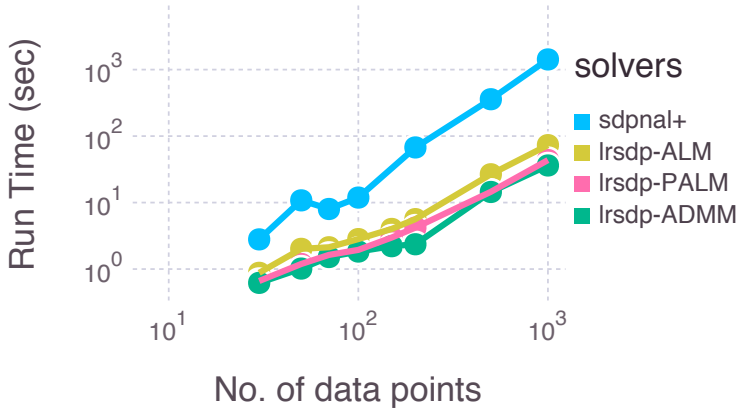


Fig. 4.4.: The fastest SDP solver SDPNAL+ and the LRSDP solvers on synthetic datasets when increasing the numbers of data points.

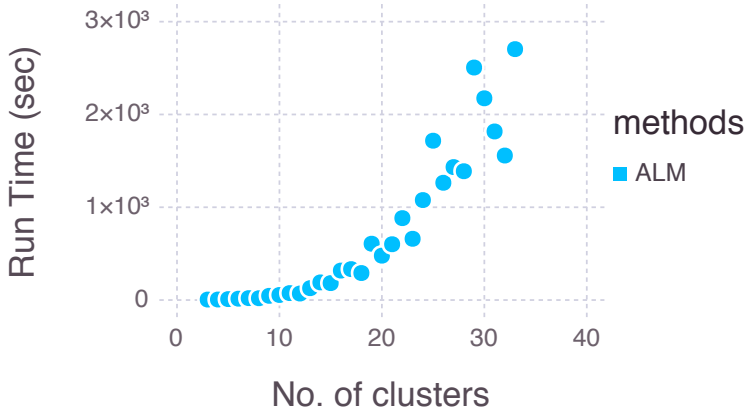


Fig. 4.5.: LRSDP solver on big block graph when increasing the numbers of clusters.

4.2.1 Classical augmented Lagrangian method

To solve (4.1), we first use a classical augmented Lagrangian framework. The augmented Lagrangian framework is a general strategy to solve nonlinear optimization problems with equality constraints. This is an iterative strategy where each step consists of minimizing an augmented Lagrangian of the problem that includes a current estimate of the Lagrange multipliers for the constraints as well as a penalty term that drives the solution towards the feasible set. Augmented Lagrangian techniques have been successful in previous studies of low-rank SDP approximations [Burer and Monteiro, 2003].

We briefly review a standard textbook derivation for completeness [Nocedal and Wright, 2006]. Consider a general optimization problem:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & c_i(\mathbf{x}) = 0,\ i = 1, \ldots, m \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \end{aligned} \tag{4.3}$$

the augmented Lagrangian for this problem involves a set of Lagrange multipliers λ_i to estimate the influence of each constraint on the objective as well as a quadratic penalty to satisfy the nonlinear constraints. It is defined as

$$\mathcal{L}_A(\mathbf{x}; \boldsymbol{\lambda}, \sigma) = f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i c_i(\mathbf{x}) + \frac{\sigma}{2} \sum_{i=1}^{m} c_i^2(\mathbf{x}).$$

An augmented Lagrangian algorithm iteratively proceeds from an arbitrary starting point to a local solution of (4.3). At each step, a bound-constrained solver minimizes

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathcal{L}_A(\mathbf{x}; \boldsymbol{\lambda}, \sigma) \\ \text{subject to} \quad & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \end{aligned} \tag{4.4}$$

Based on an approximate solution, it adjusts the Lagrange multipliers λ by following

$$\lambda_i \leftarrow \lambda_i - \sigma c_i(\mathbf{x})$$

and may update the penalty parameter σ. See Algorithm 17.4 in [Nocedal and Wright, 2006] for a standard strategy to adjust the multipliers, penalty, and tolerances for each subproblem. The major advantage of the method is that unlike the penalty method, it is not necessary to take $\sigma \to \infty$ in order to solve the original constrained problem. Instead, due to the presence of the Lagrangian multiplier estimate term, σ can stay much smaller [Nocedal and Wright, 2006].

Now we describe how to use classical augmented Lagrangian method to solve (4.1). Let $\lambda = [\lambda_1; \lambda_2; \lambda_3]$ be the Lagrange multipliers associated with the three scalar constraints $(s), (u), (w)$ in (4.1), and $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$ be the Lagrange multipliers associated with the vector constraints (t) and (v) in (4.1), respectively. Let $\sigma \geq 0$ be a penalty parameter. The augmented Lagrangian function for (4.1) is:

$$\begin{aligned}
\mathcal{L}_A(Y, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \boldsymbol{\mu}, \boldsymbol{\gamma}, \sigma) = & \underbrace{\mathbf{f}^T\mathbf{d} - \operatorname{trace}(\boldsymbol{Y}^T\boldsymbol{K}\boldsymbol{Y})}_{\text{the objective}} \\
& - \lambda_1(\operatorname{trace}(\boldsymbol{Y}^T\boldsymbol{W}^{-1}\boldsymbol{Y}) - k) \\
& \quad + \frac{\sigma}{2}(\operatorname{trace}(\boldsymbol{Y}^T\boldsymbol{W}^{-1}\boldsymbol{Y}) - k)^2 \\
& - \boldsymbol{\mu}^T(\boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}\mathbf{f}) \\
& \quad + \frac{\sigma}{2}(\boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}\mathbf{f})^T(\boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}\mathbf{f}) \\
& - \lambda_2(\mathbf{e}^T\mathbf{f} - (1+\alpha)n) + \frac{\sigma}{2}(\mathbf{e}^T\mathbf{f} - (1+\alpha)n)^2 \\
& - \boldsymbol{\gamma}^T(\mathbf{f} - \mathbf{g} - \mathbf{s}) + \frac{\sigma}{2}(\mathbf{f} - \mathbf{g} - \mathbf{s})^T(\mathbf{f} - \mathbf{g} - \mathbf{s}) \\
& - \lambda_3(\mathbf{e}^T\mathbf{g} - (1-\beta)n - r) \\
& \quad + \frac{\sigma}{2}(\mathbf{e}^T\mathbf{g} - (1-\beta)n - r)^2
\end{aligned} \tag{4.5}$$

At each step in the augmented Lagrangian solution framework, we solve the following subproblem:

$$\begin{aligned} \text{minimize} \quad & \mathcal{L}_A(\boldsymbol{Y}, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \boldsymbol{\mu}, \boldsymbol{\gamma}, \sigma) \\ \text{subject to} \quad & Y_{ij} \geq 0, \mathbf{s} \geq 0, r \geq 0, \\ & 0 \leq \mathbf{f} \leq k\mathbf{e}, 0 \leq \mathbf{g} \leq 1. \end{aligned} \tag{4.6}$$

We use a limited-memory BFGS with bound constraints algorithm [Byrd et al., 1995] to minimize the subproblem with respect to the variables $\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$ and r. This requires computation of the gradient of $\mathcal{L}_A$ with respect to the variables.

Gradients for NEO-K-Means LRSDP.

We now describe the analytic form of the gradients for the augmented Lagrangian of the NEO-K-Means low rank SDP objective and a brief validation that these are correct. Consider the augmented Lagrangian (4.5). The gradient has five components for the five sets of variables: $\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$ and r:

$$\begin{aligned} &\nabla_{\boldsymbol{Y}} \mathcal{L}_A(\boldsymbol{Y}, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \mu, \gamma, \sigma) = \\ &\quad - 2\boldsymbol{K}\boldsymbol{Y} - \mathbf{e}\mu^T\boldsymbol{Y} - \mu\mathbf{e}^T\boldsymbol{Y} \\ &\quad - 2(\lambda_1 - \sigma(\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{W}^{-1}\boldsymbol{Y}) - k))\boldsymbol{W}^{-1}\boldsymbol{Y} \\ &\quad + \sigma(\boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e}\mathbf{e}^T\boldsymbol{Y} + \mathbf{e}\mathbf{e}^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{Y}) - \sigma(\boldsymbol{W}\mathbf{f}\mathbf{e}^T\boldsymbol{Y} + \mathbf{e}\mathbf{f}^T\boldsymbol{W}\boldsymbol{Y}) \\ &\nabla_{\mathbf{f}} \mathcal{L}_A(\boldsymbol{Y}, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \mu, \gamma, \sigma) = \\ &\quad \mathbf{d} + \boldsymbol{W}\mu - \sigma(\boldsymbol{W}\boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}^2\mathbf{f}) \\ &\quad - \lambda_2\mathbf{e} + \sigma(\mathbf{e}^T\mathbf{f} - (1+\alpha)n)\mathbf{e} \\ &\quad - \gamma + \sigma(\mathbf{f} - \mathbf{g} - \mathbf{s}) \\ &\nabla_{\mathbf{g}} \mathcal{L}_A(\boldsymbol{Y}, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \mu, \gamma, \sigma) = \\ &\quad \gamma - \sigma(\mathbf{f} - \mathbf{g} - \mathbf{s}) - \lambda_3\mathbf{e} + \sigma(\mathbf{e}^T\mathbf{g} - (1-\beta)n - r)\mathbf{e} \\ &\nabla_{\mathbf{s}} \mathcal{L}_A(\boldsymbol{Y}, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \mu, \gamma, \sigma) = \gamma - \sigma(\mathbf{f} - \mathbf{g} - \mathbf{s}) \\ &\nabla_{r} \mathcal{L}_A(\boldsymbol{Y}, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \mu, \gamma, \sigma) = \lambda_3 - \sigma(\mathbf{e}^T\mathbf{g} - (1-\beta)n - r) \end{aligned}$$

Using analytic gradients in a black-box solver such as L-BFGS-B is problematic if the gradients are even slightly incorrectly computed. To guarantee the analytic gra-

dients we derive are correct, we use forward finite difference method to get numerical approximation of the gradients based on the objective function. We compare these with our analytic gradient and expect to see small relative differences on the order of 10^{-5} or 10^{-6}. This is exactly what Figure 4.6 shows.

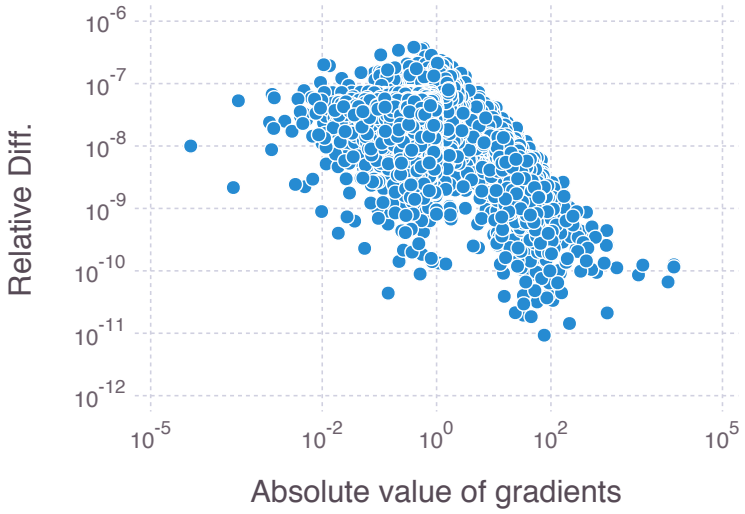


Fig. 4.6.: Finite difference comparison of gradients where $\epsilon = 10^{-6}$. This figure shows that the relative difference between the analytical gradient and the gradient computed via finite differences is small, indicating the gradient is correctly computed.

It has been shown that this augmented Lagrangian approach produces reasonable solutions for the NEO-K-Means objective. In particular, when the clustering performance is evaluated on real-world datasets, this technique has been shown to be effective in finding the ground-truth clusters. Furthermore, by optimizing the weighted kernel NEO-KMeans, this technique is also able to find cohesive overlapping communities in real-world networks. More details can be found in Section 4.5. The empirical success of the augmented Lagrangian framework (e.g., Table 4.3) motivates us to investigate developing faster solvers for the NEO-K-Means low-rank SDP problem, which will be discussed in the next.

4.2.2 Fast multiplier methods

There is a resurgence of interest in proximal point methods and alternating methods for convex and nearly convex objectives in machine learning due to their fast convergence rate. Here, we consider two fast multiplier methods to accelerate the convergence of the augmented Lagrangian scheme: the proximal augmented Lagrangian method (PALM) and an alternating direction method of multipliers (ADMM).

Proximal augmented Lagrangian

The proximal augmented Lagrangian method differs from the classical augmented Lagrangian method only in an additional proximal regularization term for primal updates. This can be considered as a type of simultaneous primal-dual proximal-point step that helps to regularize the subproblems solved at each step. This leads to the following iterates:

$$\mathbf{x}^{t+1} = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathcal{L}_{\mathcal{A}}(\mathbf{x}; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) + \frac{1}{2\tau}\|\mathbf{x} - \mathbf{x}^t\|^2$$

where $\mathbf{x}$ represents $[\mathbf{y}; \mathbf{f}; \mathbf{g}; \mathbf{s}; r]$ for simplicity with $\mathbf{y} = \boldsymbol{Y}(:)$ vectorized by column. Then we update the multipliers $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\gamma}$ as in the classical augmented Lagrangian. We may also need to update the penalty parameter σ and the proximal parameter τ respectively.

We use a limited-memory BFGS with bound constraints to solve the new subproblem with respect to the variable $\mathbf{x}$. If we let $\tau = \sigma$, this special case is called proximal method of multipliers, first introduced in Rockafellar [1976]. The proximal method of multipliers has better theoretical convergence guarantees for convex optimization problems compared with the classical augmented Lagrangian [Rockafellar, 1976]. In the non-convex setting like our NEO-K-Means problem, we believe it is likely to help to improve conditioning of the Hessian's in the subproblems and thus reduce the solution time for each subproblem. And this is indeed what we find.

Since we use the proximal augmented Lagrangian on the problem without any convexity, local convergence is the best we can achieve. By specializing a general convergence result about the proximal augmented Lagrangian in Pennanen [2002] to our problem (4.1), we can show the local convergence of our algorithm. The resulting theorem is a general convergence result about the proximal augmented Lagrangian method for non-convex problem with bound constrained subproblems.

From Pennanen [2002], we know that the proximal method of multipliers is locally convergent for a general class of problems with sufficient assumptions. We will show that our proximal method of multipliers algorithm applied to (4.1) can be handled by their approach and we extend their analysis to our case. Because we are imitating the analysis from Pennanen for a specific new problem, we decided to explicitly mimic the original language to highlight the changes in the derivation. Thus, there is a high degree of textual overlap between the following results and Pennanen [2002].

First, we state some notation and one useful fact. The indication function $\delta_{\mathcal{C}}$ of a set $\mathcal{C}$ in Hilbert Space $\mathcal{H}$ that has value 0 for $x \in \mathcal{C}$ and $+\infty$ otherwise. The subdifferential of $\delta_{\mathcal{C}}$ is the normal cone operator of $\mathcal{C}$: $N_{\mathcal{C}}(\mathbf{x}) = \{\mathbf{v} \in \mathcal{H} | \langle \mathbf{v}, \mathbf{y} - \mathbf{x} \rangle \leq 0, \forall \mathbf{y} \in \mathcal{C}\}$.

Proposition 1: Let $\bar{\mathbf{x}}$ be a solution to problem of minimizing $f(\mathbf{x})$ on $\mathcal{C}$ and suppose f is differentiable at $\bar{\mathbf{x}}$, then

$$\nabla f(\bar{\mathbf{x}}) + N_{\mathcal{C}}(\bar{\mathbf{x}}) \ni 0$$

Proof: We need to show that $\nabla f(\bar{\mathbf{x}}) + N_{\mathcal{C}}(\bar{\mathbf{x}}) \ni 0$ is equivalent to $\nabla f(\bar{\mathbf{x}})^T(\mathbf{y} - \bar{\mathbf{x}}) \geq 0$ for all $\mathbf{y} \in \mathcal{C}$, which is clear from the definition of the normal cone.

To simplify the convergence behavior analysis of the proximal method of multipliers on (4.1), we generalize the optimization problem in the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & c(\mathbf{x}) = \mathbf{0},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{4.7}$$

where $f(\mathbf{x})$ and $c(\mathbf{x})$ are continuous and differentiable. Let $\mathcal{C}$ be the closed convex sets corresponding to simple bound constrains $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$.

The Lagrangian and augmented Lagrangian function are defined respectively as follows:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T c(\mathbf{x})$$

$$\mathcal{L}_{\mathcal{A}}(\mathbf{x}, \boldsymbol{\lambda}, \sigma) = f(\mathbf{x}) + \boldsymbol{\lambda}^T c(\mathbf{x}) + \frac{\sigma}{2}\|c(\mathbf{x})\|^2$$

The multipliers $\boldsymbol{\lambda}$ can be added or subtracted. We choose adding the multipliers here in order to be consistent with the analysis in Pennanen [2002].

A point $(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}})$ is said to satisfy the strong second-order sufficient condition [Robinson, 1980] for problem (4.7) if there is a $\rho \in \mathcal{R}$ such that

$$\langle \boldsymbol{\omega}, \nabla^2_{xx}\mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}})\boldsymbol{\omega}\rangle + \rho \sum_i \langle \nabla c_i(\bar{\mathbf{x}}), \boldsymbol{\omega}\rangle^2 > 0 \quad \forall \boldsymbol{\omega} \in T_{\mathcal{C}}(\bar{\mathbf{x}})/\{0\} \tag{4.8}$$

where $T_{\mathcal{C}}(\mathbf{x})$ is the tangent cone of $\mathcal{C}$ at point $\mathbf{x}$.

We describe the proximal method of multipliers for the general form problem (4.7) below:

Algorithm 2 Proximal Method of Multipliers

1: Input: Choose $\mathbf{x}_0$, $\boldsymbol{\lambda}_0$, set $k = 0$.
2: Repeat
3: $\quad \mathbf{x}_{k+1} \leftarrow \underset{\mathbf{x}\in\mathcal{C}}{\operatorname{argmin}}\ \mathcal{L}_{\mathcal{A}}(\mathbf{x}, \boldsymbol{\lambda}_k, \sigma_k) + \dfrac{1}{2\sigma_k}\|\mathbf{x} - \mathbf{x}_k\|^2 \qquad (P^k)$
4: $\quad \boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k + \sigma_k c(\mathbf{x}_{k+1})$
5: $\quad k \leftarrow k + 1$
6: Until converged

Theorem 1: Let $(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}})$ be a KKT pair for problem (4.7) satisfying the strongly second order sufficient condition and assume the gradients $\nabla c(\bar{\mathbf{x}})$ are linearly independent. If the $\{\sigma_k\}$ are large enough with $\sigma_k \to \bar{\sigma} \leq \infty$ and if $\|(\mathbf{x}_0, \boldsymbol{\lambda}_0) - (\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}})\|$ is small enough, then there exists a sequence $\{(\mathbf{x}_k, \boldsymbol{\lambda}_k)\}$ conforming to Algorithm 1 along with open neighborhoods $\mathcal{C}_k$ such that for each k, $\mathbf{x}_{k+1}$ is the unique solu-

tion in $\mathcal{C}_k$ to (P^k). Then also, the sequence $\{(\mathbf{x}_k, \boldsymbol{\lambda}_k)\}$ converges linearly and Fejér monotonically to $\bar{\mathbf{x}}$, $\bar{\boldsymbol{\lambda}}$ with rate $r(\bar{\sigma}) < 1$ that is decreasing in $\bar{\sigma}$ and $r(\bar{\sigma}) \to 0$ as $\bar{\sigma} \to \infty$.

Proof: (Note that the theorem and proof are revisions and specializations of Theorem 19 from Pennanen 2002.) By Robinson [1980, Theorem 4.1], the strongly second-order sufficient condition and the linear independence condition imply that the KKT system for (4.7) is strongly regular at $(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}})$.

When we solve the subproblem (P^k) with explicit bound constraints, from proposition 1, we actually solve

$$\nabla f(\mathbf{x}) + \frac{1}{\sigma_k}(\mathbf{x} - \mathbf{x}_k) + N_{\mathcal{C}}(\mathbf{x}) + \nabla c(\mathbf{x})^*(\boldsymbol{\lambda}_k + \sigma_k c(\mathbf{x})) \ni 0$$

Thus, Algorithm 1 is equivalent to Algorithm 3 in Pennanen [2002] (their general algorithm), and by Theorem 17 of Pennanen [2002], we have the local convergence result stated in Theorem 1.

It remains to show that for large enough σ_k, the unique stationary point of is in fact a minimizer of (P^k). We apply the second-order sufficient condition in 13.26 from Rockafellar and Wets [2009] and the analogous derivation in the proof of Theorem 19 of Pennanen [2002], then a sufficient condition for $\mathbf{x}_{k+1}$ to be a local minimizer of (P^k) is that

$$\langle \boldsymbol{\omega}, \nabla^2_{xx}\mathcal{L}(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_{k+1})\boldsymbol{\omega}\rangle + \frac{1}{\sigma_k}\|\boldsymbol{\omega}\|^2 + \sigma_k \sum_i \langle \nabla c_i(\mathbf{x}_{k+1}), \boldsymbol{\omega}\rangle^2 > 0 \quad \forall \boldsymbol{\omega} \in T_{\mathcal{C}}(\mathbf{x}_{k+1})/\{0\}$$

This condition holds by the continuity of $\nabla^2_{xx}\mathcal{L}$ and ∇c_i, and by (4.8), provided σ_k is large enough.

A main assumption for the analysis above is that we can solve the subproblem (P^k) exactly. This was adjusted in Iusem et al. [2003], which showed local convergence for approximate solutions of (P^k).

Alternating direction method of multipliers

The alternating direction method of multipliers (ADMM) is an algorithm that solves convex optimization problems by coordinating the solutions to smaller and easier local subproblems to find a solution to a large global problem. It can be viewed as a way to combine the benefits of dual decomposition and augmented Lagrangian methods for constrained optimization [Boyd et al., 2011]. Even though ADMM was originally introduces as a tool for convex optimization problems, it turns out to be a powerful heuristic method even for NP-hard nonconvex problems.

We first briefly review the general pattern of ADMM method. Suppose we want to solve the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) + g(\mathbf{y}) \\ \text{subject to} \quad & \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{y} = \mathbf{b} \\ & \mathbf{x} \in \mathcal{X} \\ & \mathbf{y} \in \mathcal{Y} \end{aligned} \tag{4.9}$$

where $\mathbf{x}$ and $\mathbf{y}$ are variables in the sets $\mathcal{X}$ and $\mathcal{Y}$, respectively. For a given dual vector λ and positive scalar ρ, the augmented Lagrangian function is defined as:

$$\mathcal{L}_{\mathcal{A}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}, \sigma) = f(\mathbf{x}) + g(\mathbf{y}) - \boldsymbol{\lambda}^T(\boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{y} - \mathbf{b}) + \frac{\sigma}{2}\|\boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{y} - \mathbf{b}\|_2^2$$

Starting from an initial $\mathbf{y}^0$ and λ^0, primal and dual variables are updated at one iteration in ADMM as follows:

$$\begin{aligned} \mathbf{x}^{k+1} &= \min_{\mathbf{x}\in\mathcal{X}} \mathcal{L}_{\mathcal{A}}(\mathbf{x}, \mathbf{y}^k, \boldsymbol{\lambda}^k, \rho) \\ \mathbf{y}^{k+1} &= \min_{\mathbf{y}\in\mathcal{Y}} \mathcal{L}_{\mathcal{A}}(\mathbf{x}^{k+1}, \mathbf{y}, \boldsymbol{\lambda}^k, \rho) \\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k - \sigma(\boldsymbol{A}\mathbf{x}^{k+1} + \boldsymbol{B}\mathbf{y}^{k+1} - \mathbf{b}) \end{aligned} \tag{4.10}$$

It is natural to extend ADMM from two blocks of variables to multiple blocks of variables. It is easy to see that there are four sets of variables in our problem (4.1)

($\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$ and slack variables), we can use this structure to break the augmented Lagrangian function (4.5) into smaller subproblems for each set of variables. Some of these subproblems are then easier to solve optimally. For example, updating variable $\mathbf{f}$ alone is a simple convex problem, therefore it is very efficient to get a globally optimal solution. The alternating direction method of multipliers approach of updating block variables $\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$ and r respectively, utilizes this property, which leads to the following iterates:

$$
\begin{aligned}
\boldsymbol{Y}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{Y}} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}, \mathbf{f}^t, \mathbf{g}^t, \mathbf{s}^t, r^t; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) \\
\mathbf{f}^{t+1} &= \operatorname*{argmin}_{\mathbf{f}} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}^{t+1}, \mathbf{f}, \mathbf{g}^t, \mathbf{s}^t, r^t; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) \\
\mathbf{g}^{t+1} &= \operatorname*{argmin}_{\mathbf{g}} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}^{t+1}, \mathbf{f}^{t+1}, \mathbf{g}, \mathbf{s}^t, r^t; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) \\
\mathbf{s}^{t+1} &= \operatorname*{argmin}_{\mathbf{s}} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}^{t+1}, \mathbf{f}^{t+1}, \mathbf{g}^{t+1}, \mathbf{s}, r^t; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) \\
r^{t+1} &= \operatorname*{argmin}_{r} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}^{t+1}, \mathbf{f}^{t+1}, \mathbf{g}^{t+1}, \mathbf{s}^{t+1}, r; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma)
\end{aligned}
$$

Then the multipliers or dual variables $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{\gamma}$ and the penalty parameter σ are updated accordingly same as in classical augmented Lagrangian.

We expect that this strategy will aid convergence because it decouples the update of $\boldsymbol{Y}$ from the update of $\mathbf{f}$. In the problem with all variables, the interaction of these terms has the strongest non-convex interaction. We now detail how we solve each of the subproblems.

Update $\boldsymbol{Y}$.

We use a limited-memory BFGS with bound constraints [Byrd et al., 1995] to solve the subproblem with respect to the variables $\boldsymbol{Y}$ since it is non-convex.

Update $\mathbf{f}$ and $\mathbf{g}$.

The update for $\mathbf{f}$ and $\mathbf{g}$ respectively both have the following general form:

$$
\begin{aligned}
\underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) = \mathbf{x}^T\mathbf{a} + \frac{\sigma}{2}\mathbf{x}^T\boldsymbol{D}\mathbf{x} + \frac{\sigma}{2}(\mathbf{e}^T\mathbf{x})^2 \\
\text{subject to} \quad & 0 \leq \mathbf{x} \leq b\mathbf{e}
\end{aligned}
\tag{4.11}
$$

where $\boldsymbol{D}$ is a positive diagonal matrix, $\mathbf{a}$ is a constant vector, and b is a constant. To solve this, we use ideas similar to Parikh and Boyd [2014, S6.2.5]. Let $\tau = \mathbf{e}^T\mathbf{x}$, thus $0 \leq \tau \leq bn$. We solve this problem by finding roots of the following function $F(\tau)$:

$$F(\tau) = \tau - \mathbf{e}^T P[-\frac{1}{\sigma}\boldsymbol{D}^{-1}(\mathbf{a} + \sigma\tau\mathbf{e}); 0, b]$$

where the function $P[\mathbf{x}; 0, b]$ projects the point $\mathbf{x}$ onto the rectangular box $[0, b]$. To find these roots, bisection suffices because $F(0) \leq 0$ and $F(bn) \geq 0$. This is a globally optimal solution by the following lemma.

Lemma 1: $\mathbf{x}^* = P[-\frac{1}{\sigma}\boldsymbol{D}^{-1}(\mathbf{a}+\sigma\tau^*\mathbf{e}); 0, b]$, where τ^* is the root of $F(\tau)$, satisfies the first order KKT conditions:

$$\mathbf{x}^* - P[\mathbf{x}^* - \nabla f(\mathbf{x}^*); 0, b] = 0$$

(this form is given in equation 17.51 of Nocedal and Wright 2006).

Proof: We have three cases: $x_i^* = 0$; $x_i^* = b$; and $0 < x_i^* < b$ for any i.

For $x_i^* = 0$, which means $a_i + \sigma\tau \geq 0$, we have

$$x_i^* - P[x_i^* - (a_i + \sigma d_i x_i^* + \sigma\tau); 0, b] = -P[-a_i - \sigma\tau; 0, b] = 0$$

For $x_i^* = b$, which means $-(a_i + \sigma\tau)/(\sigma d_i) \geq b$, we have

$$x_i^* - P[x_i^* - (a_i + \sigma d_i x_i^* + \sigma\tau); 0, b] = b - P[b - (a_i + \sigma d_i b + \sigma\tau); 0, b] = b - b = 0$$

For $0 < x_i^* < b$, which means $x_i^* = -(a_i + \sigma\tau)/(\sigma d_i)$, we have

$$x_i^* - P[x_i^* - (a_i + \sigma d_i x_i^* + \sigma\tau); 0, b] = x_i^* - P[x_i^*; 0, b] = x_i^* - x_i^* = 0$$

Update s and r.

These updates just require solving one variable quadratic optimization with simple bound constraints; the result is a simple update procedure.

We observed empirical convergence of the ADMM method for problem (4.1), but currently lack any theoretical guarantees.

Simplified Alternating direction of multiplier method

One downside to both of the proposed methods is that they involve using the L-BFGS-B method to solve the bound-constrained non-convex objectives in the sub-steps. This is a complex routine with a high runtime itself. In this section, we are interested in seeing if there are simplified ADMM variants that might futher improve runtime by avoiding this non-convex solver. This corresponds to, for example, inexact ADMM (allowing inexact primal alternating minimization solutions, e.g., one proximal gradient step per block). Thus, we are interested in testing whether these variants can be even faster than the proposed method described in Section 4.2.2.

In the ADMM method from Section 4.2.2, we know that updating the block variables $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$ and r is simple and convex, so we can get globally optimal solutions. The only hard part is to update $\boldsymbol{Y}$, which is non-convex. However, there are few results about convergence for ADMM in the non-convex case as well as the case with multiple blocks, i.e., more than two blocks of variables (e.g., $\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$ and r) that would apply to this problem. For instance, in Chen et al. [2014], it has been shown that an ADMM method does not converge for a multi-block case even for a convex problem.

In fact, in our preliminary investigations, many common variations on the ADMM methods did not yield any performance increase or resulted in slower performance or did not converge at all. For example, we tried to avoid the L-BFGS-B in the update for $\boldsymbol{Y}$ by simply using a step of projected gradient descent instead. We found the resulting Simplified ADMM (SADMM) method converges much slower than our ADMM method with the non-convex solver. The same experiment with multiple steps of projected gradient descent only performed worse.

As evidentce of this claim, we illustrate the convergence behavior of each of the methods on an overlapping community detection task in a graph. We use the Zachary's karate club network [Zachary, 1977] which is a small social network among 34 members of a karate club.

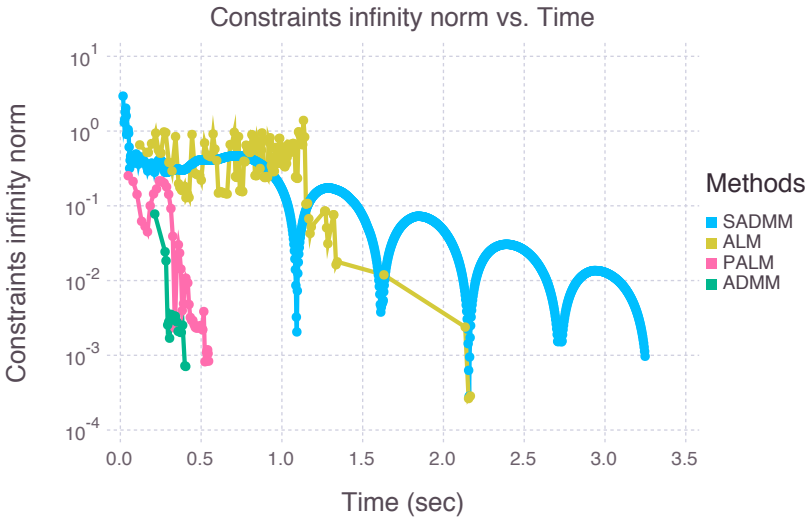
In Figure 4.7, (a) shows the infinity norm of the constraints vector and (b) shows the NEO-K-Means low-rank SDP objective function values defined in (4.1) as time progresses respectively. We set the infeasibility tolerance to be less than 10^{-3}. Both of our methods, PALM and ADMM, achieve faster convergence than ALM in terms of both the feasibility of the solution and the objective function value mainly because the subproblems for L-BFGS-B are faster to solve. To demonstrate that the common variants of ADMM do not accelerate the convergence in our problem, we also compare with the simplified alternating direction method of multipliers denoted by SADMM. Note that for SADMM, we do not need to use L-BFGS-B to solve the subproblems, instead, we use one single gradient-descent step to have the solution inexactly. It is clear to see that SADMM is much slower than ADMM, and even slower than ALM.

Therefore, common accelerated variants of ADMM proposed for convex problems with two-block case do not necessarily improve the performance of ADMM in our problem. We believe that the NEO-K-means low-rank SDP problem will be a useful test case for future research in this area.

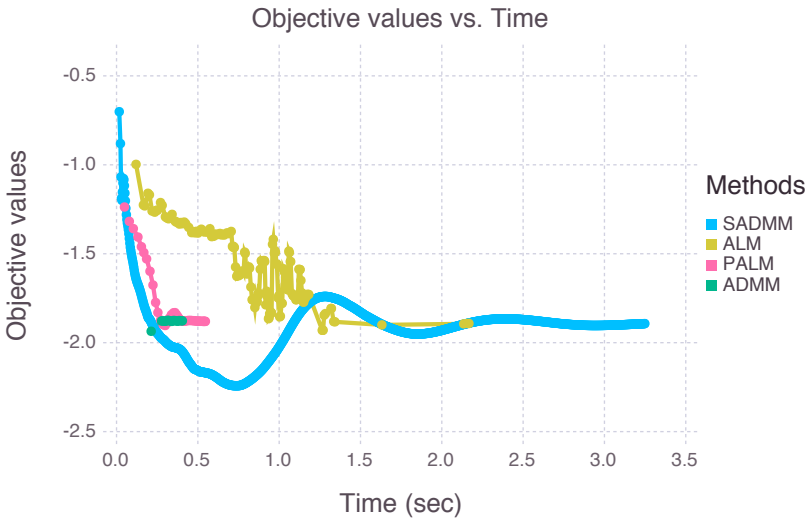
4.3 Complexity analysis

In this section, we would like to analyze the computational and memory complexity of the proposed methods to solve NEO-K-Means LRSDP problem (4.1). From the previous section, we have three methods, the classical augmented Lagrangian method (ALM), proximal augmented Lagrangian method (PALM), and alternating direction method of multipliers (ADMM).

Suppose n is the number of data points, k is the number of clusters, then the total number of variables (i.e., $\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$, r) in (4.1) is $N = kn + 3n + 1$.



(a) The infinity norm of the constraints vector vs. Time



(b) Objective function values vs. Time

Fig. 4.7.: The convergence behavior of ALM, PALM, ADMM and SADMM on a Karate Club network. PALM and ADMM converge faster than ALM while SADMM is much slower.

The most expensive step in these three methods is they involve using L-BFGS-B method [Byrd et al., 1995] to solve the bound-constrained non-convex objectives in the substeps. It is an optimization algorithm in the family of quasi-Newton methods and only stores a few m vectors of variables and their gradients that represent an estimation to the inverse Hessian matrix implicitly. This approximated Hessian matrix is used to make quasi-Newton step and we make line search along direction of this step. As a result, it requires only linear memory instead of storing a dense $N \times N$ approximation to the inverse Hessian and costs $O(mN)$ operations per iteration. We summarize the computational complexity and memory requirements for per-iteration work in L-BFGS-B routine in Table 4.2

Table 4.2.: Memory and computational cost for per-iteration work in L-BFGS-B routine for ALM, PALM, and ADMM.

	memory	computational complexity
ALM	O(kmn + 3mn + m)	O(kmn + 3mn + m)
PALM	O(kmn + 3mn + m)	O(kmn + 3mn + m)
ADMM	O(kmn)	O(kmn)

Therefore, for ALM and PALM, the total memory cost is $O(kmn + 3mn + m)$, and for ADMM, the total memory requirement is $O(kmn+3n+1)$. Also suppose p is the number of iterations in augmented Lagrangian framework, and t is the number of iterations in the routine of L-BFGS-B. Then for ALM and PALM, the computational complexity is $O(ptmN) = O(ptm(kn+3n+1))$. For ADMM method, since updating $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$ and r are just simple and require linear operations, the total computational complexity is $O(p(tmkn + 3n + 1))$.

4.4 Practical improvements

Finally, we describe a set of practical improvements for our methods. These are designed to accelerate the convergence of the augmented Lagrangian framework by moving it closer to a point that satisfies the constraints and is nearly optimal. They

are designed based on commonly used strategies in the relax and round approach to discrete optimization problems.

Final rounding.

At the conclusion of our rounding procedure, we have an assignment of points to clusters. We then use that as the initial cluster assignments for the iterative NEO-K-Means procedure from [Whang et al., 2015]. Since that procedure has monotone convergence behavior, this can only improve the solution.

Initialization.

We run the iterative NEO-K-Means algorithm multiple times and use the result with the best objective function value as the initialization to LRSDP. For problems over a few hundred data points, this procedure results in faster convergence and better final solutions.

Sampling.

For vector datasets without feature maps, we found that first using LRSDP on sampling 10% of the data points, then using this LRSDP solution as an initialization of the iterative algorithm produces similarly good results as using LRSDP on all the data points while taking significantly less time.

Hierarchical results.

For overlapping community detection on large graph data (e.g., the HepPh and AstroPh datasets we show later), we apply a two-level hierarchical clustering. In the first level, we use LRSDP with $k' = \sqrt{k}$, $\alpha' = \sqrt{1+\alpha} - 1$ and unchanged β, then in the second level, we run LRSDP with k', α' and $\beta' = 0$ for each cluster at level 1. These parameter settings produce a final assignment result with a total of $(1+\alpha)n$ assignments in k clusters.

4.5 Experimental Results

We begin by validating our implementation and comparing our solutions against the global optima from the CVX and SDPNAL+ program. We then show the effective-

ness of LRSDP as an initialization method of the iterative NEO-K-Means algorithm which is a simple greedy algorithm designed for optimizing the NEO-K-Means objective function. Finally, we show experimental results on vector and graph clustering problems by comparison with state-of-the-art clustering and community detection methods. All our algorithms are implemented in MATLAB and use the L-BFGS-B routine [Byrd et al., 1995] written in Fortran to solve the bound-constrained nonlinear subproblems.

4.5.1 Algorithmic validation

Now we compare the objective function values produced by proposed LRSDP solvers as well as the convex formulation of the problem solved by CVX and SDPNAL+ on real-world datasets. We consider two graph clustering problems using 'dolphins' [Lusseau et al., 2003] and 'les miserables' [Knuth, 1993] datasets. The 'dolphins' network represents frequent associations between 62 dolphins (there are 159 undirected edges in the network), and 'les miserables' network represents the co-appearance of characters in the novel Les Miserables (there are 77 nodes and 254 edges). We also consider a data clustering problem using MUSIC dataset which contains 593 music songs, each of which is represented by a 72 dimensional feature vector (details are available in Section 4.5.3).

Table 4.3 shows the results. We try a set of different configurations with k, α, and β for two graph clustering problem ('dolphins' and 'les miserables'). For MUSIC data set, we set k as the number of ground truth clusters and the parameters α and β values are automatically detected (see Whang et al. 2015 for details).

We compare the run time of CVX and SDPNAL+ solver and LRSDP methods and find that LRSDP methods are roughly an order of magnitude faster than CVX and much faster than SDPNAL+. Note that the two fast multiplier methods, PALM and ADMM, are faster than the classical augmented Lagrangian method.

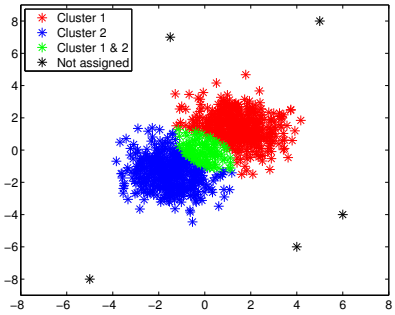
Table 4.3.: The objective values and the runtimes (in seconds) of the SDP and LRSDP solvers on real-world datasets.

			SDP		LRSDP		
			cvx-mosek	sdpnal+	ALM	PALM	ADMM
dolphins	k=2,α=0.2,β=0	obj	70.4311	70.0313	70.0317	70.0342	70.0323
		time	47.3581	13.4862	2.3986	1.9673	1.6496
	k=3,α=0.2,β=0.05	obj	68.5128	68.1170	68.1162	68.2019	68.2432
		time	80.1786	11.8781	5.6517	1.9854	1.3128
les mis	k=2,α=0.2,β=0	obj	88.4627	88.0635	88.0649	88.0801	88.1052
		time	89.7610	23.0577	5.0189	2.8710	1.4301
	k=3,α=0.2,β=0.05	obj	86.5542	86.1544	86.1809	86.2108	86.2277
		time	59.1738	23.0490	5.9618	3.7513	2.3374
music	k=6,α=1.587,β=0.002	obj	N/A	64823.513	64824.358	64823.315	64823.908
		time	N/A	437.44	107.68	43.53	29.14

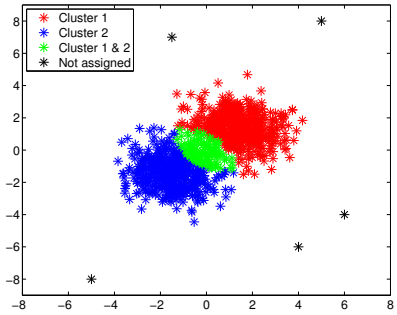
In Table 4.3, we also report the objective values before the relaxed solution is rounded to a discrete assignment solution to precisely measure how much our solution is different from the solution returned by CVX and SDPNAL+. We can see that the objective values returned from CVX, SDPNAL+, and returned from our LRSDP solvers are essentially identical—they are different in light of the solution tolerances given by the methods. *In these cases, then, we are successful in finding a globally optimal solution.*

4.5.2 Motivating example

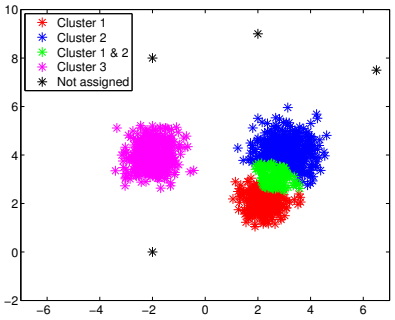
Now, we show how we can exploit the benefit of LRSDP by using it as an initialization of the simple iterative NEO-K-Means algorithm. We consider two synthetic datasets shown in Figure 4.8(a) & Figure 4.8(c). In these datasets, green data points indicate the overlapped region between clusters, and black data points indicate outliers which are not supposed to belong to any cluster. The first dataset was considered in Whang et al. [2015]. We run the iterative NEO-K-Means algorithm on these datasets with two different initialization methods: k-means and LRSDP. On a simpler dataset, Figure 4.8(a), we observe that the NEO-K-Means can always recover the underlying clustering structure regardless of the initialization methods. However, on



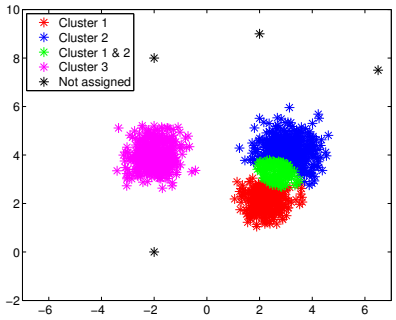
(a) Ground-truth clusters



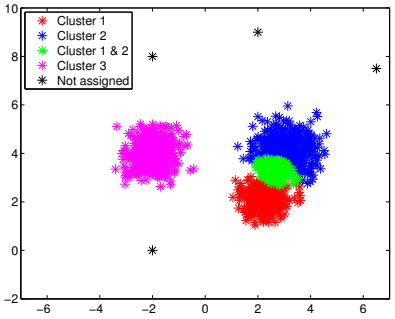
(b) k-means/LRSDP initialization



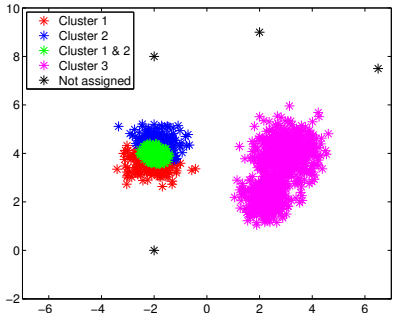
(c) Ground-truth clusters



(d) LRSDP initialization



(e) Success of k-means initialization



(f) Failure of k-means initialization

Fig. 4.8.: The output of NEO-K-Means algorithm with two different initialization methods on two synthetic datasets. (a) & (b) On a simple dataset, NEO-K-Means can easily recover the ground-truth clusters with k-means or LRSDP initialization. (c)–(f) LRSDP initialization allows the NEO-K-Means algorithm to consistently produce a reasonable clustering structure whereas k-means initialization sometimes (4 times out of 10 trials) leads to a failure in recovering the underlying clustering structure.

Figure 4.8(c), we observe the advantages of LRSDP over the k-means initialization. When we use the LRSDP initialization, the NEO-K-Means always yields a similar clustering structure as the ground-truth clusters as shown in Figure 4.8(d). On the other hand, when the k-means initialization is used, the NEO-K-Means fails to recover the underlying clustering structure 4 times out of 10 trials as shown in Figure 4.8(f). Thus, we see that on more complicated datasets, the dangers of bad initialization and being stuck in local minima become clearer, and LRSDP provides a more stable initialization, which enables the NEO-K-Means algorithm to consistently produce a reasonable clustering.

4.5.3 Data clustering

We show some experimental results on real-world vector datasets. We use three multi-label datasets which we get from [Tsoumakas et al., 2010]. Table 4.4 presents some basic statistics of these datasets ('dimension' denotes the dimensionality of the data vectors and 'avg. csize' denotes the average size of the ground-truth clusters). The MUSIC dataset [Trohidis et al., 2008] consists of a set of feature vectors extracted from 593 different music songs. In this dataset, each song is labelled by emotions presented in the song, e.g., happy, surprised, relaxing, etc. Since several different emotions can be expressed in a song, a song can have more than one label. The SCENE dataset [Boutell et al., 2004] is a set of scene image feature vectors. Each image can be labelled by their scenes, e.g., beach, sunset, mountain, and an image can contain more than one scene. The YEAST dataset [Elisseeff and Weston, 2001] is from a biology domain. This dataset is a set of feature vectors constructed based on micro-array expression data and phylogenetic profiles of genes. Each gene belongs to multiple functional classes, so each gene can have multiple labels. On these datasets, we treat each label as a ground-truth cluster. We set k as the number of ground-truth clusters; $k = 6$ for MUSIC and SCENE, and $k = 14$ for YEAST.

Table 4.4.: Real-world vector datasets.

	n	dimension	avg. csize	k
scene	2,407	294	430.8	6
yeast	2,417	103	731.5	14
music	593	72	184.7	6

To see the effectiveness of our LRSDP methods, we initialize ALM, PALM, and ADMM using the iterative NEO-K-Means algorithm and use a final iterative NEO-K-Means improvement step. The parameters α and β in the NEO-K-Means are automatically estimated by the strategies proposed in Whang et al. [2015]. This initialization renders the method sensitive to the local region selected by the iterative method, but this is usually a high-quality region. We use the procedure from Algorithm 1 to round the real-valued solutions to discrete assignments. Briefly, this uses the solution vectors $\mathbf{g}$ and $\mathbf{f}$ to determine which points to assign and roughly how many clusters each data point resides in. Assignments are then greedily made based on values of the solution matrix $\boldsymbol{Y}$. We run all the methods 25 times on the three datasets, and summarize the results in Figure 4.10 – 4.13. The results from these experiments illustrate the following points:

- (Figure 4.10 – objective values) The ALM, PALM, and ADMM methods are all indistinguishable as far as their ability to optimize the objective of the NEO-K-Means low-rank SDP problem (4.1).
- (Figure 4.11 – runtimes) Both the PALM and ADMM methods are significantly faster than ALM on the larger two datasets, SCENE and YEAST. In particular, ADMM is more than 13 times faster on the SCENE dataset. Since the MUSIC dataset is relatively small, the speedup is also relatively small, but the two new methods, PALM and ADMM are consistently faster than ALM. Note that we do not expect any of the optimization-based methods will be faster than the iterative NEO-K-Means method since it is a completely different type of algorithm (In particular, it optimizes the discretized objective).

Thus, we conclude that the new proposed optimization procedures (PALM and ADMM) are considerably faster than the ALM method while achieving similar objective function values.

The next investigation studies the discrete assignments produced by the methods. Here, we see that (Figure 4.12) there are essentially no differences among any of the LRSDP optimization methods (ALM, PALM, ADMM) in terms of their objective values after rounding to the discrete solution and evaluating the NEO-K-Means objective. A lower objective value indicates a better clustering. The optimization methods outperform the iterative method on the YEAST dataset by a considerable margin. There is also a significant difference in the objective value on MUSIC data for the worst case. We note that the benefit of LRSDP methods on SCENE dataset is not significant, but we also note that on this dataset, the average behavior of all methods is roughly the same. In this case, the overlaps among the ground-truth clusters are very small (the ground-truth α is 0.074) which implies that the traditional k-means should be a highly accurate initialization.

Finally, to see the clustering performance, we compute the F_1 score which measures the matching between the algorithmic solutions and the ground-truth clusters in Figure 4.13. Higher F_1 scores indicate better alignment with the ground-truth clusters. We also compare the results with other state-of-the-art overlapping clustering methods, such as model-based overlapping clustering [Banerjee et al., 2005] (denoted by MOC), explicit/implicit sparsity constrained clustering [Lu et al., 2012] (denoted by ESP and ISP), and overlapping k-means [Cleuziou, 2008] (denoted by OKM). On the YEAST dataset, MOC returns 13 empty clusters and one large cluster which contains all the data points. So, we do not report F_1 score of MOC on this dataset. We first observe that the NEO-K-Means based methods (Iterative, ALM, PALM, and ADMM) are able to achieve higher F_1 scores than other methods. When we compare the performance among the three NEO-K-Means LRSDP optimization methods (ALM, PALM, and ADMM), there is largely no difference among these methods except for the MUSIC dataset. On the MUSIC problem, the ADMM method has a

slightly lower F_1 score than PALM or ALM. This is because the objective values obtained by ADMM on this dataset seem to be minutely higher than the other two optimization strategies and this manifests as a noticeable change in the F_1 score. In this case, however, the scale of the variation is low and essentially, the results from all the NEO-K-Means based methods are equivalent. On the SCENE dataset, the iterative algorithm (NEO-iterative) can sometimes outperform the optimization methods in terms of F_1 although we note that the median performance of the optimization is much better and there is essentially no difference between ALM, PALM, and ADMM. On the YEAST dataset, the reduced objective function value corresponds with an improvement in the F_1 scores for notably better results than iterative NEO-K-Means algorithm.

4.5.4 Graph clustering: overlapping community detection

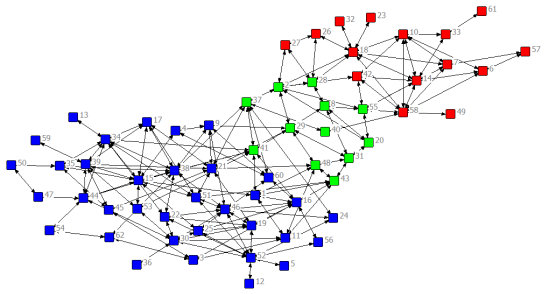
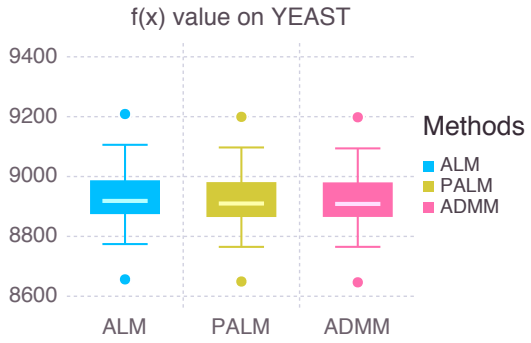
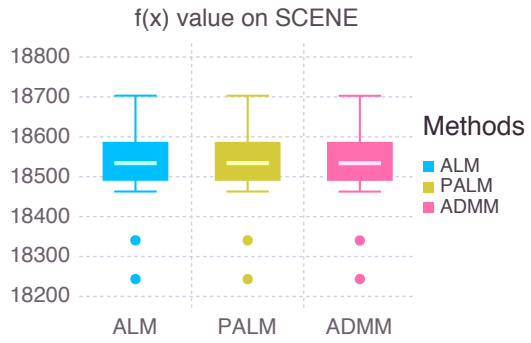


Fig. 4.9.: Visualization of the clustering result of LRSDP on 'dolphins' network. Blue nodes only belong to cluster 1, red nodes only belong to cluster 2, and green nodes belong to both of the clusters.

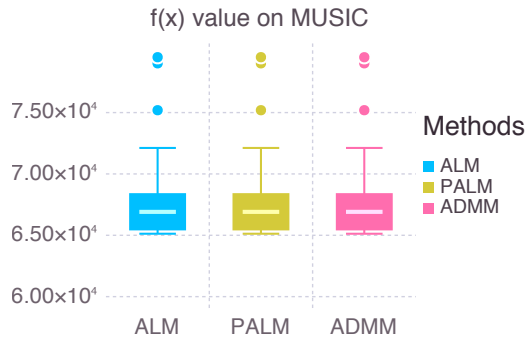
The iterative NEO-K-Means method and our new NEO K-Means LRSDP methods can both be used for overlapping community detection because optimizing the NEO-K-Means objective function corresponds to optimizing an extended version of normalized cut [Whang et al., 2015]. To see whether LRSDP produces a reasonable clustering structure on graphs, we visualize the clustering result of LRSDP (k=2, α=0.2, β=0) on the 'dolphins' network [Lusseau et al., 2003] in Figure 4.9. There



(a) Objective values in (4.1) on YEAST

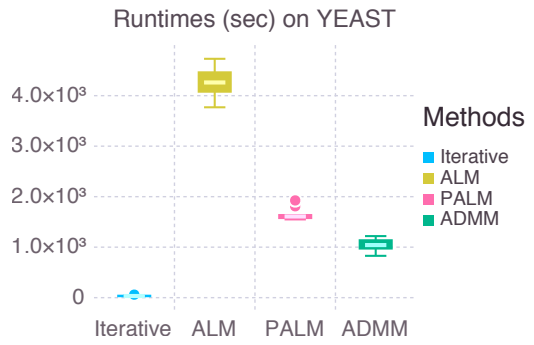


(b) Objective values in (4.1) on SCENE

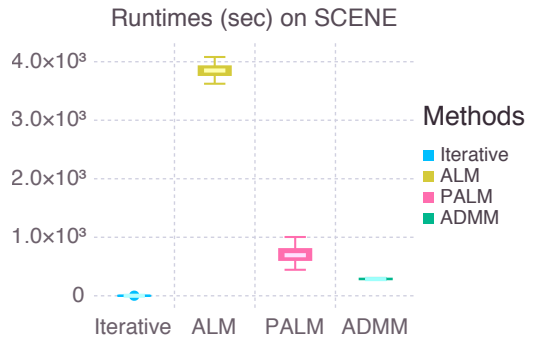


(c) Objective values in (4.1) on MUSIC

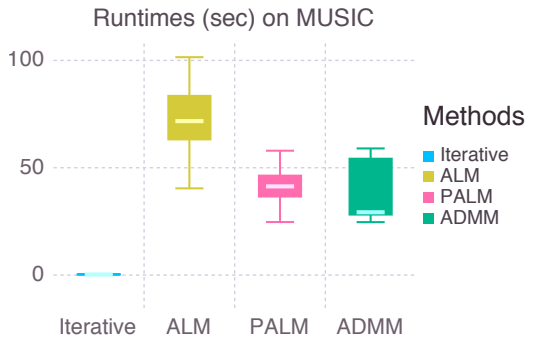
Fig. 4.10.: Box-plots comparing the results on 25-trials of the algorithms in terms of objective values in (4.1) on YEAST, SCENE, and MUSIC datasets. The median performance is indicated by the middle line and the box shows the 25% and 75% percentiles.



(a) Runtime on YEAST

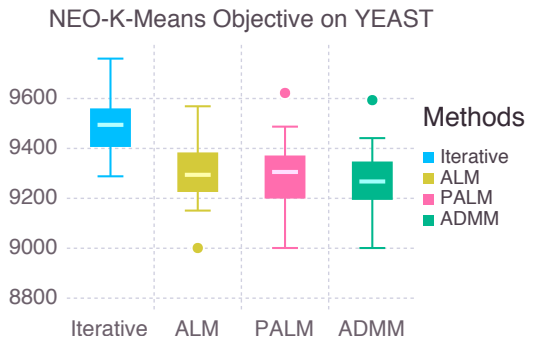


(b) Runtime on SCENE

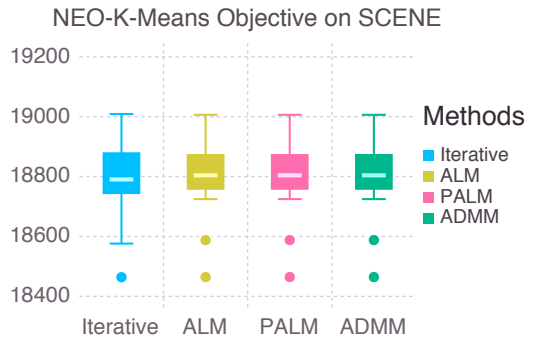


(c) Runtime on MUSIC

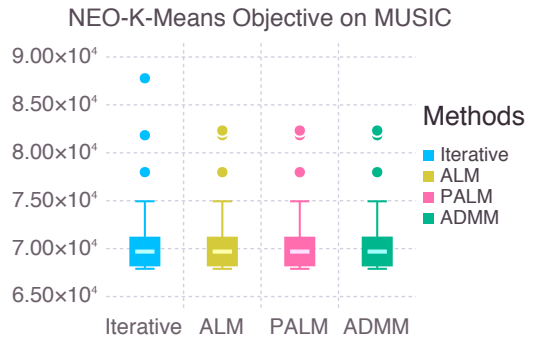
Fig. 4.11.: Box-plots comparing the results on 25-trials of the algorithms in terms of runtimes on YEAST, SCENE, and MUSIC datasets. The median performance is indicated by the middle line and the box shows the 25% and 75% percentiles.



(a) NEO-K-Means objective on YEAST



(b) NEO-K-Means objective on SCENE



(c) NEO-K-Means objective on MUSIC

Fig. 4.12.: Box-plots comparing the results on 25-trials of the algorithms in terms of NEO-K-Means objective values in (2.5) on YEAST, SCENE, and MUSIC datasets. The median performance is indicated by the middle line and the box shows the 25% and 75% percentiles.

are two clusters where green nodes indicate the overlapped region (blue and green nodes form one cluster, and red and green nodes form the other cluster). Notice that the green nodes have many interactions with both of the clusters, which shows that LRSDP produces a plausible solution aligned with an intuitive clustering structure.

Next, we consider real-world networks from [Leskovec, 2016]. We use three different networks which are summarized in Table 4.5. Facebook is a social network, and HepPh and AstroPh are collaboration networks. To run LRSDP on the two large networks, HepPh and AstroPh, we use a hierarchical clustering which we discussed in Section 4.4. Table 4.6 shows the comparison of the average normalized cut between the multilevel NEO-K-Means algorithm [Whang et al., 2015] and LRSDP. The multilevel NEO-K-Means (denoted by 'multilevel neo' or 'm-neo') is a variation of the iterative NEO-K-Means algorithm where the graph clustering problem is solved at multiple scales. We also use the multilevel NEO-K-Means as the final improvement step of LRSDP as we briefly discussed in Section 4.4. We see that LRSDP achieves the lower normalized cut than the multilevel NEO-K-Means, which indicates that LRSDP is beneficial to optimizing the objective function. Within these methods, we set k=32, α=3, β=0 on Facebook network. On large networks, we determine α and β values based on the statistics of the output of *nise* method [Whang et al., 2013].

Table 4.5.: Real-world graph datasets.

	No. of vertices	No. of edges
Facebook	348	2,866
HepPh	11,204	117,619
AstroPh	17,903	196,972

We compare the performance of the NEO-K-Means method with other state-of-the-art overlapping community detection methods: *demon* [Coscia et al., 2012], *oslom* [Lancichinetti et al., 2011], *bigclam* [Yang and Leskovec, 2013], *nise* [Whang et al., 2013]. For *nise*, we use the "Graclus centers" seeding strategy and the Fiedler PageRank clustering scheme.

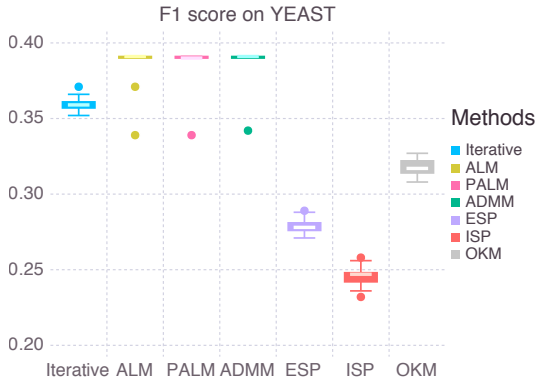
We use the conductance and the modularity metrics to gauge the quality of the communities returned by each of the algorithms. These two metrics are considered to be a standard way to measure the cohesiveness of the clusters. Since a good overlapping community detection method should cover a large portion of the graph with a set of cohesive clusters, we compute AUC (Area Under the Curve) of the metric-versus-coverage. See [Whang et al., 2013] for the details about how to compute the AUC score. In Table 4.7, a higher AUC score indicates a better clustering result. We see that the NEO-K-Means algorithm achieves the highest AUC on all the networks, which indicates that it produces the most cohesive groups of communities.

Table 4.6.: Average normalized cut of the iterative multilevel NEO-K-Means and NEO-K-Means Low Rank SDP

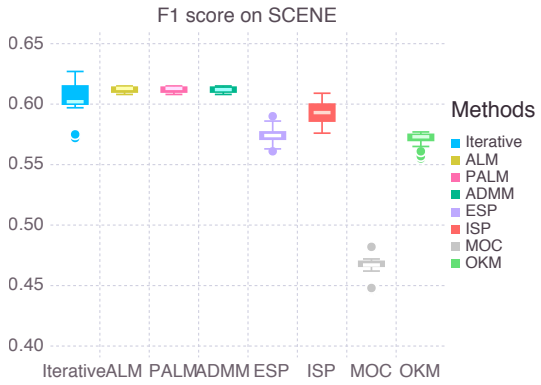
	multilevel neo	LRSDP
Facebook	0.371	**0.279**
HepPh	0.185	**0.169**
AstroPh	0.240	**0.201**

Table 4.7.: AUC scores on real-world graphs. A higher AUC score indicates a better clustering result.

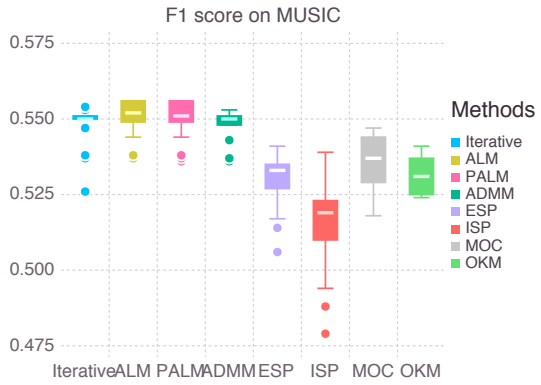
		demon	*oslom*	*bigclam*	*nise*	NEO
Facebook	conductance	0.5048	0.6805	0.1699	0.7027	**0.8090**
	modularity	0.0674	0.0932	0.0086	0.0938	**0.1586**
HepPh	conductance	0.4970	0.5349	0.3752	0.8981	**0.9166**
	modularity	0.0391	0.0066	0.0085	0.1886	**0.1947**
AstroPh	conductance	0.4304	0.4202	0.3545	0.8466	**0.8630**
	modularity	0.0224	0.0010	0.0051	0.1638	**0.1834**



(a) F_1 scores on YEAST



(b) F_1 scores on SCENE



(c) F_1 scores on MUSIC

Fig. 4.13.: Box-plots comparing the results on 25-trials of the algorithms in terms of F_1 scores on YEAST, SCENE, and MUSIC datasets. The median performance is indicated by the middle line and the box shows the 25% and 75% percentiles.

5. LOW RANK STRUCTURE IN KERNEL CLUSTERING

5.1 Introduction

It's become almost passe to say that data sets around us are growing in size in this era of big data. While it is now reasonably straightforward to collect repositories of many tens of terabytes, and no longer particularly difficult to reach the petabyte range, the owners of such repositories share a common question: what should we do to demonstrate the value of our data? Ideally, such questions should be addressed prior to the collection of such massive datasets. But careful experimental design is not yet common with these large collections of data — much of which was often collected for more reasons with clear business needs, such as billing or usage information, or technical needs, such as systems logging for performance anomalies. Thus, there is a critical need for methods to help organize large quantities of messy data in a variety of increasingly diverse applications from sensor data, image data, logging data, systems data, and biological data.

One of the principal means to address these problems is to utilize an unsupervised clustering method to automatically and efficiently group the data into sets where each set is composed of items *nearby* under an appropriate distance measure or similarity function. Again – these techniques are not perfect – but they often provide a starting point that suggests more refined methods in the future. For that reason, the results of methods like clustering are often dubbed exploratory data analyses.

Most large-scale clustering techniques reported in the literature focus on grouping based on the Euclidean distance by assuming that all the data points lie in a Euclidean distance. As we mentioned in Sec 2.3, kernel-based methods are a common tool in the arsenal of machine learning and data mining domains that are brought to bear on applications and data in image recognition, text analysis, network analysis,

and so on [Shi and Malik, 2000; Shawe-Taylor and Cristianini, 2004; Zhang et al., 2007]. Kernel-based clustering methods can overcome the limitation that only linear relations can be captured between data points, by embedding the data points into an arbitrary-dimensional non-linear feature space and defining their similarities using a nonlinear kernel distance function [Kim et al., 2005]. There are a number of kernel-based clustering methods proposed in recent years such as spectral clustering [Ng et al., 2001; Shi and Malik, 2000], non-negative matrix factorization (NMF) [Xu et al., 2003], kernel Self-Organizing Maps (SOM) [Inokuchi and Miyamoto, 2004].

In this thesis, we focus on kernel k-means clustering [Girolami, 2002] because of its simplicity and efficiency with simple iterative Lloyd's algorithm. For example, in spectral clustering method, top eigenvalues of the similarity matrix of data points are required. The time complexity of this computation is cubic in the number of data points. While kernel k-means does not require and the time complexity of simple iterative Lloyd-like algorithm is $O(n^2)$ [Dhillon et al., 2004]. Moreover, several studies [Dhillon et al., 2004; Zha et al., 2001; Ding et al., 2005] have shown the equivalence of kernel k-means and other kernel-based clustering methods. These imply that most kernel-based clustering methods perform similarly.

Table 5.1.: Examples of kernel functions.

Polynomial Kernel	$\mathcal{K}(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + c)^d$
Gaussian Kernel	$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2})$
Sigmoid Kernel	$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \tanh(a\mathbf{x}^T\mathbf{y} + b)$

Kernel k-means is a clustering method that uses a non-linear relationship between data points to capture application specific similarity criteria. It generalizes the celebrated k-means strategy for automatically organizing data into groups through the

use of a kernel matrix that encodes the set of all relationships between the data points. To be specific, it replaces the Euclidean distance function

$$d^2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$$

employed in the k-means algorithm with a non-linear kernel-based distance function defined as

$$d^2_{\mathcal{K}}(\mathbf{x}, \mathbf{y}) = \mathcal{K}(\mathbf{x}, \mathbf{x}) + \mathcal{K}(\mathbf{y}, \mathbf{y}) - 2\mathcal{K}(\mathbf{x}, \mathbf{y})$$

where $\mathbf{x} \in \mathcal{R}^d$ and $\mathbf{y} \in \mathcal{R}^d$ are two data points and $\mathcal{K}(.,.) : \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$ is the kernel function. It is actually the special case of the kernel version of NEO-K-Means (2.5) when $\alpha = 0$ and $\beta = 0$. The Euclidean distance in feature space from $\phi(\mathbf{x})$ to cluster $\mathcal{C}_j$'s mean $\mathbf{m}_j$ is given by

$$d^2_{\mathcal{K}}(\mathbf{x}, \mathbf{m}_j) = \mathcal{K}(\mathbf{x}, \mathbf{x}) - \frac{2}{|\mathcal{C}_j|} \sum_{\mathbf{y} \in \mathcal{C}_j} \mathcal{K}(\mathbf{x}, \mathbf{y}) + \frac{1}{|\mathcal{C}_j|^2} \sum_{\mathbf{y}, \mathbf{z} \in \mathcal{C}_j} \mathcal{K}(\mathbf{y}, \mathbf{z}) \tag{5.1}$$

Examples of popular kernel functions are summarized in Table 5.1. We summarize the Kernel k-means in Algorithm 3 here for completeness.

Algorithm 3 Iterative kernel k-means

1: Input: kernel matrix $\boldsymbol{K}$, number of clusters k
2: Initialize: k clusters $\mathcal{C}_1^{(0)}, \ldots, \mathcal{C}_k^{(0)}$, t=0
3: Repeat
4: For each point $\mathbf{x}$, find its new cluster index whose mean is the "nearest":

$$\text{Index}(\mathbf{x}) = \text{argmin}_j \, d^2_{\mathcal{K}}(\mathbf{x}, \mathbf{m}_j) \quad \text{using (5.1)}$$

5: Compute the updated clusters:

$$\mathcal{C}_j^{t+1} = \{\mathbf{x} : \text{Index}(\mathbf{x}) = j\}$$

6: t $\leftarrow$ t+1
7: Until converged

However, directly using kernel k-means clustering introduces a major scalability challenge. Consider, for example, a database of a million pictures represented as $32 \times$

32 pixel grids, where each pixel is a feature. This data is only a few gigabytes in size. However, common kernels to derive meaningful associations between images for the kernel k-means algorithm require forming a matrix that is one million-by-one million and dense. Storing this matrix using traditional techniques requires eight terabytes of memory. Put more formally, the space complexity of the kernel matrices necessary for kernel based clustering schemes is $O(n^2)$, while computing it takes $O(n^2 d)$ operations, where n is the number of data points and d is the dimension. This is infeasible for big data problems and clearly, we need a new approach.

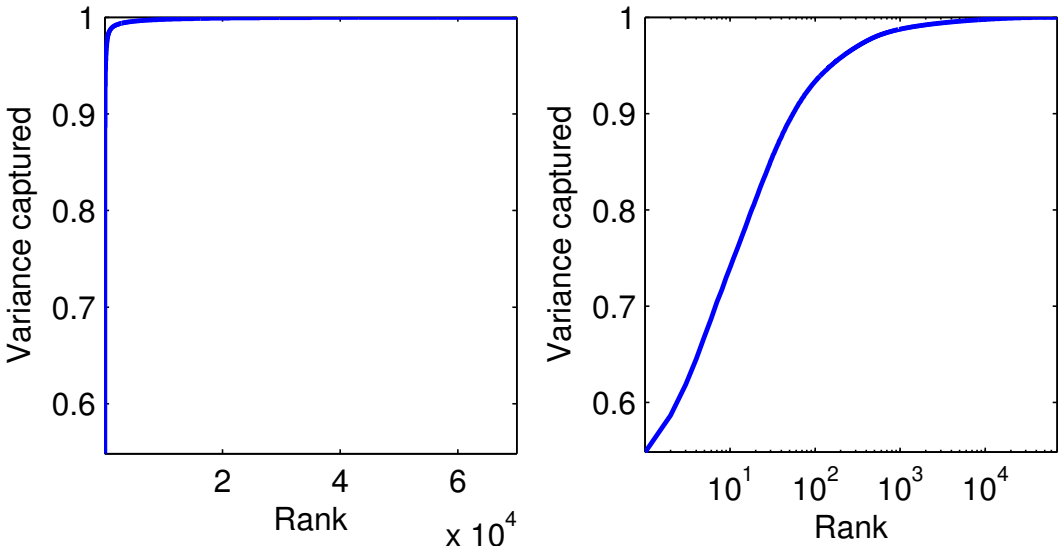


Fig. 5.1.: Consider the kernel matrix for the approximately 70,000 images of the MNIST dataset using the sigmoid kernel. At left, we show the amount of the total norm (or variance) captured by the best rank k approximation using a linear scale. At right, we show the same data using a log-scale. These figures suggest that this matrix is effectively rank 1000, despite having true rank 70,000. Most of the additional rank structure is effectively noise.

To remedy this issue, approximating the kernel matrix using fewer parameters or limited memory storage is a natural approach to reduce the complexity both in time and space. The key property of kernel matrices that we want to exploit is that they are often nearly low-rank [Gittens and Mahoney, 2016]. This can be formalized as follows: the singular values of kernel matrix are localized – this means that we can

preserve the majority of the norm of the singular values while rounding small entries to zero. Figure 5.1 makes this behavior clear. In that figure, we have a 70,000-by-70,000 kernel matrix formed using a sigmoid kernel on the MNIST digit dataset [Zhang and Rudnicky, 2002]. We find that it is effectively rank 1000. This means there exists a 70,000-by-1000 matrix $\boldsymbol{Y}$ such that $\boldsymbol{K} \approx \boldsymbol{Y}\boldsymbol{Y}^T$.

This observation implies that there is a nearby low-rank matrix that approximates the given kernel to a high degree of accuracy. This low rank structure serves as a low dimensional embedding of the kernel matrix that enables us to use traditional, scalable techniques in order to solve problems such as kernel k-means clustering.

Many methods have been proposed to find a low-rank matrix that can have a good approximation of kernel matrices. The underlying idea for most methods is to select a subset of columns of the kernel matrix in running-time complexity in $O(n)$ while the differences are the strategies of how to sample columns. Such methods appears through different formulations within numerical linear algebra or machine learing, including greedy basis selection techniques [Smola and Schökopf, 2000], incomplete Cholesky decomposition [Fine and Scheinberg, 2002; Bach and Jordan, 2005], Nyström methods [Williams and Seeger, 2001; Chitta et al., 2011], and CUR matrix decompositions [Mahoney and Drineas, 2009]. It has been thoroughly investigated in contexts where the goal is symmetric positive semi-definite (SPSD) matrix approximation [Gittens and Mahoney, 2016].

Inspired by low rank matrix completion techniques, which predicts the unknown entries of target matrix $\boldsymbol{Y} \in R^{n \times m}$ with the lowest rank given a random subset of observed entries, we propose a novel means of handling kernels in this thesis, through a matrix completion framework will provide a means for these techniques to scale to large data sets. The particular structure we exploit is the local *stable rank* or *localized rank* of the matrices that arise through many of the common kernel functions studied in these applications. In addition to computing low rank approximations of kernel matrix for reducing computation complexity, kernel matrix completion has other applications where much information of points is not available. For example,

in biology, it is often the case that only a small subset of samples are available for other observed data. As a result, we can only form a partial kernel matrix derived from such datasets. In addition, the missing data could result also from avoiding the calculation of all the kernel values in the case of computationally expensive kernel functions such as those on strings, graphs, or generally more structured, complex representations [Haussler, 1999].

5.2 Related works

In this section, we first review the related work on kernel approximations via columns sampling, and then review literatures on matrix completion techniques.

5.2.1 Kernel approximation via columns sampling

Nyström-based methods were originally used by [Williams and Seeger, 2001] to solve regression and classification problems involving Gaussian processes when the SPSD matrix is well-approximated by a low rank matrix. Given an SPSD matrix $\boldsymbol{K} \in \mathcal{R}^{n \times n}$ and randomly sample m columns of matrix $\boldsymbol{K}$, then we can compute a low rank approximation to the matrix $\boldsymbol{K}$ by the standard Nyström method as follows:

$$\tilde{\boldsymbol{K}} = \boldsymbol{K_B} \hat{\boldsymbol{K}}^{+} \boldsymbol{K_B^T}$$

where $\boldsymbol{K_B} \in \mathcal{R}^{n \times m}$ is a matrix formed by randomly sampling a small number m of columns ($m \ll n$) of $\boldsymbol{K}$, and $\hat{\boldsymbol{K}}$ is the matrix formed by the intersection between those m columns of $\boldsymbol{K}$ and the corresponding m rows of $\boldsymbol{K}$.

Many algorithms haven been proposed to improve the kernel approximation by improving the sampling strategy [Kumar et al., 2012; Gittens and Mahoney, 2016], or considering both low rank and clustering stucture of the kernel matrix [Si et al., 2014]. For kernel-based clustering problem, Chitta et al. [2011] proposes approximate kernel k-means clustering to address the issue in large scale kernel clustering and it

shows the equivalence to the standard Nyström method for low rank approximation of kernel matrix.

5.2.2 Matrix completion

Matrix completion methods were popularized by the Netflix challenge and address the problem: given a small set of entries randomly sampled from a low-rank, or nearly low-rank, matrix find the best low rank approximation that models those entries.

Let us now state the matrix completion problem formally. Suppose that a general matrix $\boldsymbol{M} \in \mathcal{R}^{m \times n}$ and let Ω be a subset of $[m] \times [n]$ of revealed entries of $\boldsymbol{M}$. Then we want to find a solution of the following problem

$$\begin{aligned} \underset{\boldsymbol{X}}{\text{minimize}} \quad & \text{rank}(\boldsymbol{X}) \\ \text{subject to} \quad & \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}), \\ & \boldsymbol{X} \in \mathcal{R}^{m \times n} \end{aligned} \tag{5.2}$$

Unfortunately, this problem is generally NP-hard [Vandenberghe and Boyd, 1996], but there is a convex relaxation for this problem called nuclear norm relaxation, which replace the rank function with the nuclear norm [Fazel, 2002; Candès and Recht, 2009]. For a matrix $\boldsymbol{A}$, the nuclear norm is defined as

$$\|\boldsymbol{A}_*\| = \sum_{i=1}^{\text{rank}(\boldsymbol{A})} \sigma_i(\boldsymbol{A})$$

where $\sigma_i(\boldsymbol{A})$ is the ith singular value of $\boldsymbol{A}$. Because the nuclear norm is convex,

$$\begin{aligned} \underset{\boldsymbol{X}}{\text{minimize}} \quad & \|\boldsymbol{X}\|_* \\ \text{subject to} \quad & \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}) \end{aligned} \tag{5.3}$$

is a tractable convex relaxation of (5.2).

Many algorithms have been proposed to solve matrix completion, such as Singular Value Projection [Jain et al., 2010], Singular Value Threshing [Cai et al., 2010] and so

on [Candès and Recht, 2009; Gross, 2011]. All of these algorithms require computing singular value decomposition (SVD) which becomes increasingly costly as the sizes and ranks of the underlying matrices increase. Therefore, [Wen et al., 2012] exploits an alternative approach avoiding SVD computation for solving large-scale problems. The method is based on the minimization of the following type:

$$\begin{aligned} \underset{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{Z}-\boldsymbol{X}\boldsymbol{Y}\|_F^2 \\ \text{subject to} \quad & \mathcal{P}_\Omega(\boldsymbol{Z}) = \mathcal{P}_\Omega(\boldsymbol{M}) \end{aligned} \tag{5.4}$$

where $\boldsymbol{X} \in \mathcal{R}^{m\times k}$, $\boldsymbol{Y} \in \mathcal{R}^{k\times n}$ and $k \ll m, n$.

Let's present the algorithm (denoted by LMaFit) in Wen et al. [2012] which suggests minimizing separately over $\boldsymbol{X}$, $\boldsymbol{Y}$. They use successive over-relaxation method applied to normal equations. For the matrix completion problem, they found their implementation outperformed many competitors. Thus, we first use the LMaFit in Wen et al. [2012] for symmetric kernel matrix completion problem. For completeness, we summarize their procedure in Algorithm 4 here.

Algorithm 4 A low-rank matrix completion algorithm (LMaFit) [Wen et al., 2012]

1: Input: index Ω, data $\mathcal{P}_\Omega(\boldsymbol{M})$, tolerance ϵ, step-size ω_t for t=0, 1, ...
2: Initialize: $\boldsymbol{X}^0 \in \mathcal{R}^{n\times k}$ is random matrix, $\boldsymbol{Y}^0 \in \mathcal{R}^{k\times n} = \boldsymbol{0}$, $\boldsymbol{Z}^0 = \mathcal{P}_\Omega(\boldsymbol{K})$, and t=0
3: Repeat
4: $\quad \boldsymbol{Z}_\omega^t \leftarrow \boldsymbol{X}^t\boldsymbol{Y}^t + \omega(\boldsymbol{Z}^t - \boldsymbol{X}^t\boldsymbol{Y}^t)$
5: $\quad \boldsymbol{X}^{t+1} \leftarrow \text{orth}((\boldsymbol{Z}_\omega^t)(\boldsymbol{Y}^t))$
6: $\quad \boldsymbol{Y}^{t+1} \leftarrow (\boldsymbol{X}^{t+1})^T\boldsymbol{Z}_\omega^t$
7: $\quad \boldsymbol{Z}^{t+1} \leftarrow \boldsymbol{X}^{t+1}\boldsymbol{Y}^{t+1} + P_\Omega(\boldsymbol{M} - \boldsymbol{X}^{t+1}\boldsymbol{Y}^{t+1})$
8: $\quad$ t $\leftarrow$ t+1
9: Until $\|\mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^t\boldsymbol{Y}^t)\| \le \epsilon$

The step 5 in Algorithm 4 is usually computed via QR-decomposition to get the orthogonal basis of some subspace, which is more stable than using normal equations. Parameter ω, which can be regarded as the step-size, is adjusted in every step using the procedure we will discuss later. This method may not converge to the global minimum but Wen et al. [2012] proved convergence to a stationary point.

There are also other algorithms designed for symmetric positive semi-definite (SPSD) matrices. As far as we know Graepel [2002] is the first to apply matrix completion by semidefinite programming to recover the unknown entries of the kernel matrix. Bishop and Yu [2014] also consider SPSD matrix completion but they drop the assumption of a random sampling of entries in favor of a deterministic sampling of principal submatrices of the matrix.

5.3 Our algorithms

5.3.1 Symmetric scheme

We first use LMaFit algortihm [Wen et al., 2012] for approximating kernel matrices from a subset of randomly sampled entries. However, this kind of matrix completion method and algorithm is not engineered to solve kernel approximation problems, and in particular, they often do not compute symmetric factorization from symmetric inputs – a key requirement for kernel matrices. We describe an example of this problem as follows. Suppose we consider a matrix is approximately rank 2 with the eigenvalue decomposition:

$$\begin{bmatrix} 14 & 0 & 2 \\ 0 & 14 & 2 \\ 2 & 2 & 7 \end{bmatrix} = \boldsymbol{Z}_0 = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^T, \text{where}$$

$$\boldsymbol{Q} = \begin{bmatrix} \frac{2}{3} & \frac{1}{\sqrt{2}} & -\frac{\sqrt{2}}{6} \\ \frac{2}{3} & -\frac{1}{\sqrt{2}} & -\frac{\sqrt{2}}{6} \\ \frac{1}{3} & 0 & \frac{2\sqrt{2}}{3} \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 15 & 0 & 0 \\ 0 & 14 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

If we provide the full set of non-zeros of $\boldsymbol{Z}_0$ as the subset of entries of the matrix, then after a single iteration of the algorithm by [Wen et al., 2012], the low-rank approximation is:

$$\boldsymbol{Z}_1 = \begin{bmatrix} 14 & 0 & 2.8824 \\ 0 & 14 & 2.8824 \\ 2 & 2 & 0.8235 \end{bmatrix}$$

This non-symmetric structure remains for almost all iterations.

Therefore, we propose a simple modification to hold the symmetry for symmetric inputs. Suppose that a symmetric matrix $\boldsymbol{K} \in \mathcal{R}^{n \times n}$ and let Ω be a subset of $[n] \times [n]$ of known entries of $\boldsymbol{K}$. The model with the symmetric constraint is as follows:

$$\begin{aligned} \underset{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{Z} - \boldsymbol{X}\boldsymbol{Y}\|_F^2 \\ \text{subject to} \quad & \mathcal{P}_\Omega(\boldsymbol{Z}) = \mathcal{P}_\Omega(\boldsymbol{K}), \\ & \boldsymbol{X}\boldsymbol{Y} = (\boldsymbol{X}\boldsymbol{Y})^T \end{aligned} \tag{5.5}$$

where $\boldsymbol{Z} \in \mathcal{R}^{n \times n}$, $\boldsymbol{X} \in \mathcal{R}^{n \times k}$, $\boldsymbol{Y} \in \mathcal{R}^{k \times n}$ and $k \ll n$.

In order to make sure $\boldsymbol{X}\boldsymbol{Y}$ is symmetric for all iterations, a simple modification is posed when updating $\boldsymbol{Y}$ for the above algorithm like the following:

$$\boldsymbol{Y}^{t+1} \leftarrow (\boldsymbol{X}^{t+1})^T \boldsymbol{Z}_\omega^t \boldsymbol{X}^{t+1} (\boldsymbol{X}^{t+1})^T \tag{5.6}$$

where $\boldsymbol{Y}^{t+1}$ is the solution of minimizing $\|\boldsymbol{Z}_\omega^t - \boldsymbol{X}^{t+1}\boldsymbol{Y}\|_F^2$ when $\boldsymbol{X}^{t+1}$ is fixed and orthogonal with the constraint that $\boldsymbol{X}^{t+1}\boldsymbol{Y}$ is symmetric. This modification is based on the following lemma:

Lemma: The solution for the following problem

$$\begin{aligned} \underset{\boldsymbol{Y}}{\text{minimize}} \quad & \|\boldsymbol{Z} - \boldsymbol{U}\boldsymbol{Y}\|_F^2 \\ \text{subject to} \quad & \boldsymbol{U}\boldsymbol{Y} = (\boldsymbol{U}\boldsymbol{Y})^T \end{aligned} \tag{5.7}$$

where $\boldsymbol{U} \in \mathcal{R}^{n \times k}$ is an orthogonal basis of some subspace, has the closed form

$$\boldsymbol{Y}^* = \frac{1}{2}\boldsymbol{U}^T(\boldsymbol{Z} + \boldsymbol{Z}^T)\boldsymbol{U}\boldsymbol{U}^T$$

Proof: From $\boldsymbol{U}\boldsymbol{Y} = (\boldsymbol{U}\boldsymbol{Y})^T$, we have $\boldsymbol{Y} = \boldsymbol{U}^T\boldsymbol{Y}^T\boldsymbol{U}^T, \boldsymbol{Y}^T = \boldsymbol{U}\boldsymbol{Y}\boldsymbol{U}$. Let $\boldsymbol{D} = \boldsymbol{U}^T\boldsymbol{Y}^T$, then $\boldsymbol{D} = \boldsymbol{U}^T\boldsymbol{Y}^T = \boldsymbol{U}^T\boldsymbol{U}\boldsymbol{Y}\boldsymbol{U} = \boldsymbol{Y}\boldsymbol{U} = \boldsymbol{D}^T$. Then the problem would be the following: choose D such that $\|\boldsymbol{Z} - \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T\|_F^2$ is minimal where $\boldsymbol{D} = \boldsymbol{D}^T$. Then

$$\underset{\boldsymbol{D}}{\text{minimize}} \quad \|\boldsymbol{Z} - \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T\|_F^2 \quad \Longleftrightarrow \quad \underset{\boldsymbol{D}}{\text{minimize}} \quad \|\boldsymbol{D} - \boldsymbol{A}\|_F^2$$

where $\boldsymbol{A} = \boldsymbol{U}^T\boldsymbol{Z}\boldsymbol{U}$. It is easy to get the solution

$$\boldsymbol{D}^* = \frac{1}{2}(\boldsymbol{A} + \boldsymbol{A}^T) = \frac{1}{2}\boldsymbol{U}^T(\boldsymbol{Z} + \boldsymbol{Z}^T)\boldsymbol{U}$$

which means, the solution for the original problem is

$$\boldsymbol{Y}^* = \frac{1}{2}\boldsymbol{U}^T(\boldsymbol{Z} + \boldsymbol{Z}^T)\boldsymbol{U}\boldsymbol{U}^T$$

When $\boldsymbol{Z}$ is symmetric, we get $\boldsymbol{Y} = \boldsymbol{U}^T\boldsymbol{Z}\boldsymbol{U}\boldsymbol{U}^T$.

However, this simple modification is not sufficient to make the algorithm work. Let the residual at the iteration t be $\boldsymbol{S}^t = \mathcal{P}_\Omega(\boldsymbol{K} - \boldsymbol{X}^t\boldsymbol{Y}^t)$. If we apply this modification only, we find that the algorithm may not guarantee the objective function is always decreasing at each iteration, that is $\boldsymbol{S}^t$ is not always decreasing.

To address this issue, we also observe that since $\boldsymbol{Y}^t \leftarrow (\boldsymbol{X}^t)^T\boldsymbol{Z}_\omega^{t-1}\boldsymbol{X}^t(\boldsymbol{X}^t)^T$, which means we can update $\boldsymbol{X}^{t+1} = \text{orth}(\boldsymbol{Z}_\omega^t(\boldsymbol{X}^t))$ instead of using $\boldsymbol{X}^{t+1} = \text{orth}(\boldsymbol{Z}_\omega^t(\boldsymbol{Y}^t)^T)$. Inspired by orthogonal iteration [Golub and Van Loan, 2012], we let

$$\boldsymbol{X}^{t+1} = \text{orth}((\boldsymbol{Z}_\omega^t)^p(\boldsymbol{X}^t))$$

where p is the smallest number of steps of an orthogonal iteration on $\boldsymbol{Z}_\omega^t$ to achieve a smaller objective function value, i.e. $\|\boldsymbol{S}^{t+1}\|_F^2 \leq \|\boldsymbol{S}^t\|_F^2$. We will always find a such

$\boldsymbol{X}^{t+1}$, since under mild conditions the orthogonal basis $\boldsymbol{X}^{t+1}$ converges to the invariant subspace spanned by the first k eigenvectors corresponding to the k dominant eigenvalues of $\boldsymbol{Z}_{\omega}^{t}$ where

$$|\lambda_1| \geq |\lambda_2| \geq \cdots |\lambda_k| \geq |\lambda_{k+1}| \geq \cdots \geq |\lambda_n|$$

and the convergence rate depends on $|\lambda_k|/|\lambda_{k+1}|$.

Algorithm 5 A low-rank symmetric matrix completion algorithm

1: Input: symmetric index Ω, data $\mathcal{P}_{\Omega}(\boldsymbol{K})$, tolerance ϵ, step-size ω_t for t=0,1,...
2: Initialize: $\boldsymbol{X}^0 \in \mathcal{R}^{n\times k}$ is random matrix, $\boldsymbol{Y}^0 \in \mathcal{R}^{k\times n} = \boldsymbol{0}$, $\boldsymbol{Z}^0 = \mathcal{P}_{\Omega}(\boldsymbol{K})$, and t=0
3: Repeat
4: $\quad \boldsymbol{Z}_{\omega}^{t} \leftarrow \boldsymbol{X}^t\boldsymbol{Y}^t + \omega(\boldsymbol{Z}^t - \boldsymbol{X}^t\boldsymbol{Y}^t)$
5: $\quad \boldsymbol{X}^{t+1} \leftarrow \text{orth}((\boldsymbol{Z}_{\omega}^{t})^p(\boldsymbol{X}^t))$
6: $\quad \boldsymbol{Y}^{t+1} \leftarrow (\boldsymbol{X}^{t+1})^T\boldsymbol{Z}_{\omega}^{t}\boldsymbol{X}^{t+1}(\boldsymbol{X}^{t+1})^T$
7: $\quad \boldsymbol{Z}^{t+1} \leftarrow \boldsymbol{X}^{t+1}\boldsymbol{Y}^{t+1} + P_{\Omega}(\boldsymbol{K} - \boldsymbol{X}^{t+1}\boldsymbol{Y}^{t+1})$
8: $\quad$ t $\leftarrow$ t+1
9: Until $\|\mathcal{P}_{\Omega}(\boldsymbol{K} - \boldsymbol{X}^t\boldsymbol{Y}^t)\| \leq \epsilon$

Summarizing all above discussions, we arrive at our Algorithm 5 for symmetric positive semi-definite matrices. The orthogonal basis is computed via QR. From our empirical studies, we find that in most iterations, p is typically 1 or 2. Parameter ω_t, which can be regarded as the step-size, is adjusted in every step using the following procedures: if the new residual is smaller than the previous one, accept the new triplet; if the ratio between current and previous residual is small enough, then keep the ω, otherwise increase the ω; if not, that is, the new residual is greater or equal than the previous one, compute $(\boldsymbol{X}^{k+1}, \boldsymbol{Y}^{k+1}, \boldsymbol{Z}^{k+1})$ once again from $(\boldsymbol{X}^{k}, \boldsymbol{Y}^{k}, \boldsymbol{Z}^{k})$ using $\omega = 1$; if after using $\omega = 1$, the new residual is still greater or equal than the previous one, increase p until the new residual decreases.

5.3.2 Convergence analysis

We now analyze the convergence for Algorithm 5. Since the model in (5.5) is non-convex, we are only able to establish convergence to a stationary point under a mild assumption. First, by introducing Lagrange multipliers $\boldsymbol{\Lambda} \in \mathcal{R}^{n\times n}$ so that $\boldsymbol{\Lambda} = \mathcal{P}_\Omega(\boldsymbol{\Lambda})$ and $\boldsymbol{\Gamma} \in \mathcal{R}^{n\times n}$, the Lagrangian function of our model is defined as

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{\Lambda}, \boldsymbol{\Gamma}) = \frac{1}{2}\|\boldsymbol{Z} - \boldsymbol{XY}\|_F^2 - \boldsymbol{\Lambda} \bullet \mathcal{P}_\Omega(\boldsymbol{Z} - \boldsymbol{K}) - \boldsymbol{\Gamma} \bullet (\boldsymbol{XY} - \boldsymbol{Y}^T\boldsymbol{X}^T)$$

where $\boldsymbol{A} \bullet \boldsymbol{B}$ is the inner product between two matrices $\boldsymbol{A} \in \mathcal{R}^{m\times n}$ and $\boldsymbol{B} \in \mathcal{R}^{m\times n}$ defined as $\boldsymbol{A} \bullet \boldsymbol{B} \coloneqq \sum_i \sum_j A_{ij}B_{ij} = \text{trace}(\boldsymbol{B}^T\boldsymbol{A})$. Differentiating the Lagrangian function $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{\Lambda}, \boldsymbol{\Gamma})$ with respect to $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ respectively, we have the first-order optimal condition:

$$\begin{aligned}
(\boldsymbol{XY} - \boldsymbol{Z})\boldsymbol{Y}^T - \boldsymbol{\Gamma}^T\boldsymbol{Y}^T + \boldsymbol{\Gamma}\boldsymbol{Y}^T &= 0 \\
\boldsymbol{X}^T(\boldsymbol{XY} - \boldsymbol{Z}) - \boldsymbol{X}^T\boldsymbol{\Gamma}^T + \boldsymbol{X}^T\boldsymbol{\Gamma} &= 0 \\
\mathcal{P}_{\Omega^c}(\boldsymbol{Z} - \boldsymbol{XY}) &= 0 \\
\mathcal{P}_\Omega(\boldsymbol{Z} - \boldsymbol{K}) &= 0 \\
\boldsymbol{XY} - \boldsymbol{Y}^T\boldsymbol{X}^T &= 0
\end{aligned} \tag{5.8}$$

plus the equation

$$\mathcal{P}_\Omega(\boldsymbol{Z} - \boldsymbol{XY}) = \boldsymbol{\Lambda} \tag{5.9}$$

We try to analyze the convergence from another perspective instead of looking at the objective function. It is obvious that in one update step, the algorithm finds an orthogonal basis of a new subspace, then gets the optimal symmetric solution on that subspace. That is similar with the orthogonal iteration, which is also called *subspace iteration* or *simultaneous iteration* [Golub and Van Loan, 2012].

Note that the objective function in (5.5) is bounded below by zero and is decreased at every iteration. There must hold that $\mathcal{P}_{\Omega^c}(\boldsymbol{X}^{t+1}\boldsymbol{Y}^{t+1} - \boldsymbol{X}^t\boldsymbol{Y}^t) \to 0$. It is entirely reasonable to assume that $\{\mathcal{P}_{\Omega^c}(\boldsymbol{X}^t\boldsymbol{Y}^t)\}$ remains bounded.

Working Conjecture: Let $\{(\boldsymbol{X}^t, \boldsymbol{Y}^t, \boldsymbol{Z}^t)\}$ be generated by Algorithm 5 and $\{\mathcal{P}_{\Omega^c}(\boldsymbol{X}^t\boldsymbol{Y}^t)\}$ be bounded. Then there exists at least a subsequence of $\{(\boldsymbol{X}^t, \boldsymbol{Y}^t, \boldsymbol{Z}^t)\}$ that satisfies the first-order optimality conditions in (5.8) in the limit.

Proof: It follows from the boundedness of $\{\mathcal{P}_{\Omega^c}(\boldsymbol{X}^t\boldsymbol{Y}^t)\}$ and the algorithm construction that both $\{\boldsymbol{Z}^t\}$ and $\{\boldsymbol{X}^t\boldsymbol{Y}^t\}$ are bounded sequences. It suffices to prove $(\boldsymbol{X}^t)^T(\boldsymbol{X}^t\boldsymbol{Y}^t - \boldsymbol{Z}^t) = 0$ and $(\boldsymbol{X}^t\boldsymbol{Y}^t - \boldsymbol{Z}^t)(\boldsymbol{Y}^t)^T = 0$ of the optimality conditions (5.8) since other conditions are satisfied by the construction of Algorithm 5. Note that

$$\begin{aligned} (\boldsymbol{X}^t)^T\boldsymbol{\Gamma}^T - (\boldsymbol{X}^t)^T\boldsymbol{\Gamma} = 0 \\ \boldsymbol{\Gamma}^T(\boldsymbol{Y}^t)^T - \boldsymbol{\Gamma}(\boldsymbol{Y}^t)^T = 0 \end{aligned} \tag{5.10}$$

if we assume that multiplier matrix $\boldsymbol{\Gamma}$ is symmetric from our algorithm construction.

From theoretical perspective, if we can not proceed the algorithm any more, i.e. we can not decrease the objective function any more, that is to say, ω is already set to 1, and $\boldsymbol{X}^t\boldsymbol{Y}^t$ is already the the best-k approximation of $\boldsymbol{Z}^t$, so $\boldsymbol{X}^t$ is the the first k eigenvectors corresponding to the k dominant eigenvalues of $\boldsymbol{Z}^t$. Then it is easy to get

$$(\boldsymbol{X}^t)^T(\boldsymbol{X}^t\boldsymbol{Y}^t - \boldsymbol{Z}^t) = 0$$

since $(\boldsymbol{Y}^t)^T$ can rewritten as $\boldsymbol{X}^t\boldsymbol{D}$ where $\boldsymbol{D}$ is a k-by-k matrix.

As long as we get $(\boldsymbol{X}^t)^T(\boldsymbol{X}^t\boldsymbol{Y}^t - \boldsymbol{Z}^t) = 0$, we will always get

$$(\boldsymbol{X}^t\boldsymbol{Y}^t - \boldsymbol{Z}^t)(\boldsymbol{Y}^t)^T = 0$$

in the similar way.

5.3.3 Kernel k-means approximation

After we get the low rank approximation of the kernel matrix $\boldsymbol{K} \approx \boldsymbol{X}\boldsymbol{Y}$ where $\boldsymbol{X} \in \mathcal{R}^{n\times r}$ and $\boldsymbol{Y} \in \mathcal{R}^{r\times n}$, we can run approximate kernel k-means clustering using

$\boldsymbol{X}$ and $\boldsymbol{Y}$. The space complexity is $O(nk)$ instead of $O(n^2)$ with the full kernel k-means clustering. We outline the proposed algorithm in Algorithm 6.

Algorithm 6 kernel k-means approximation

1: Input: kernel matrix approximation$\boldsymbol{X} \in \mathcal{R}^{n\times r}$, $\boldsymbol{Y} \in \mathcal{R}^{r\times n}$, number of clusters k
2: Randomly initialize the assignment matrix $\boldsymbol{U} \in \mathcal{R}^{k\times n}$, t=0
3: Repeat
4: Compute the normalized assignment matrix $\hat{\boldsymbol{U}} = [\text{diag}(\boldsymbol{U}\mathbf{e})]^{-1}\boldsymbol{U}$
5: Calculate $\boldsymbol{P} = \hat{\boldsymbol{U}}\boldsymbol{X}$, $\boldsymbol{Q} = \hat{\boldsymbol{U}}\boldsymbol{Y}^T$
6: For each point $\mathbf{x}_i$, find its new cluster index whose mean is the "nearest":

$$\text{Index}(\mathbf{x}_i) = \text{argmin}_j \, \mathbf{p}_j^T\mathbf{q}_j - 2\mathbf{p}_j^T\mathbf{y}_i$$

where $\mathbf{p}_j$ and $\mathbf{q}_j$ are the jth rows of matrix $\boldsymbol{P}$ and $\boldsymbol{Q}$, $\mathbf{y}_i$ is the ith column of matrix $\boldsymbol{Y}$.
7: Update the assignment matrix:

$$U_{ji} = 1 \quad \text{for j} = \text{Index}(\mathbf{x}_i)$$

and zero otherwise
8: t $\leftarrow$ t+1
9: Until the assignment matrix $\boldsymbol{U}$ does not change

5.4 Experimental results

We now provide experimental results to validate the usefullness of symmetric matrix completion scheme to approximate kernel matrices.

5.4.1 Kernel k-means on real-world data

In this section, we study the proposed method for large scale kernel matrices. Experiments are performed on kernel k-means clustering. We get a low-rank approximation of kernel matrices via our symmetric matrix completion scheme (Algorithm 5), and then approximate kernel k-means using the low rank approximation of kernel matrices (Algorithm 6).

Table 5.2.: Datasets for kernel k-means clustering

Data	# data points	dimensions	kernels used
CIFAR-10	50,000	3,072	Gaussian kernel
MNIST	70,000	784	Sigmoid kernel

We evaluate our algorithms on two popular image datasets: MNIST and CIFAR-10, summarized in Table 5.2. The MNIST dataset [LeCun et al., 1998] is a subset of the database of handwritten digits available from NIST. It constains a total of 70,000 binary images from 10 digit classes. The kernel function was used for this data set, as suggested in Zhang and Rudnicky [2002]: the neural or sigmoid kernel defined as:

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \tanh(a\mathbf{x}^T\mathbf{y} + b)$$

According to Zhang and Rudnicky [2002], we set the parameters a and b to 0.0045 and 0.11 respectively.

The CIFAR-10 dataset consists of 50,000 images for 10 classes (e.g. airplane, bird, cat, etc.) [Krizhevsky, 2009]. The kernel function we used for this dataset is the Gaussian kernel defined as

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|x - y\|_2^2}{2\sigma^2})$$

where we set $\sigma = 1$.

We compare our proposed method (denoted by "mc") with the full kernel k-means (denoted by "full kernel")and the algorithm from [Chitta et al., 2011](denoted by "nystrom"). For the low rank approximation of a target rank r, "nystrom" selects r columns from matrix K. For matrix completion method "mc", we vary the number of sampled entries by c*$\log_2$(n)*n where c is the sampling parameter we set from 10

to 50. We conduct multiple runs of experiments for each method and report average results here.

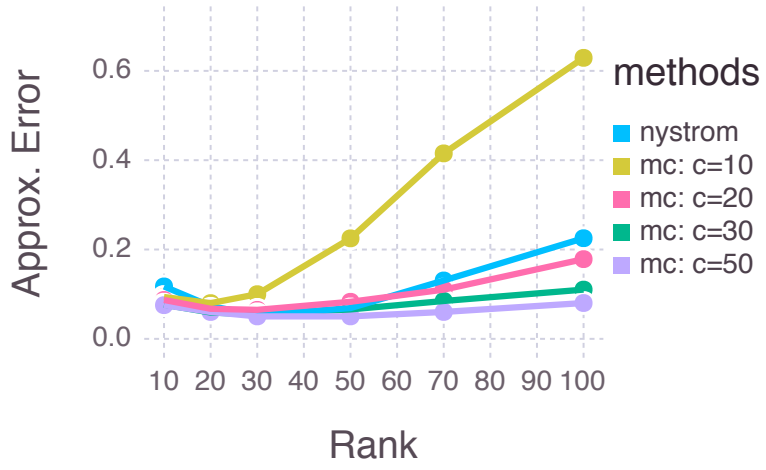
We first investigate the kernel approximation results. We use relative kernel approximation error

$$\frac{\|\boldsymbol{K} - \tilde{\boldsymbol{K}}\|_F}{\|\boldsymbol{K}\|_F}$$

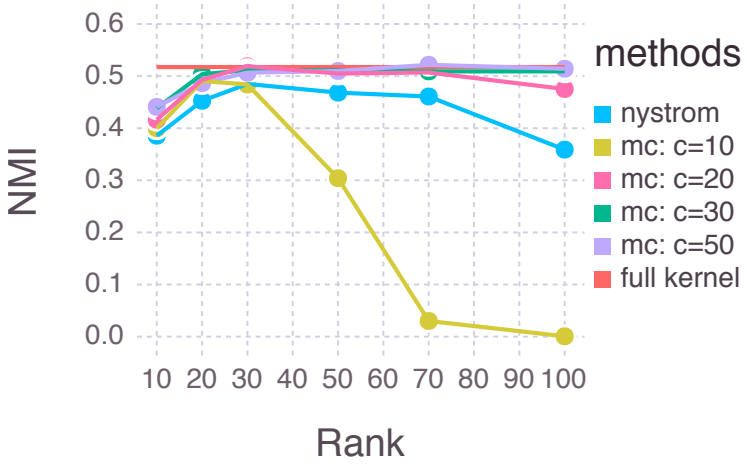
to measure the quality. Next we examine the prediction accuracy of clustering results by calculating the Normalized Mutual Information (NMI) [Kvalseth, 1987] with respect to the true class distribution. A value close to 1 indicates better matching with the true class distribution while 0 indicates perfect mismatch. We also report the clustering performance of "full kernel" for references.

As can be seen in Figure 5.2(a) & 5.4(a), the approximation errors for "mc" methods are increasing as the rank increases. It is not surprising because for matrix completion problem, more sample entries are needed to successfully recover for larger rank. For MNIST dataset in Figure 5.2(a), except the case $c = 10$, "mc" methods can get smaller approximation error compared with "nystrom". In Figure 5.3(a), we see that even "nystrom " method may achieve acceptable approximation error with smaller sample entries, it is not stable, while our method is more reliable when the amount the sample entries is sufficient. In Figure 5.4(a), our method didn't compete with "nystrom" in general for CIFAR-10 because of the relative large effective rank of full kernel marix, but for low ranks e.g., $r = 10$ or 20, "mc" can have smaller errors than "nystrom" method.

In Figure 5.2(b), Figure 5.3(b) & 5.4(b), we observe that, except the case $c = 10$, the proposed method empirically show the clustering performance is comparable to the full kernel k-means and better than "nystrom" method, especially for MNIST dataset. This means the proposed method is a promising alternative way other than column-sampling based methods to yield similar clustering results as the full kernel k-means algorithm. In particular, for the case we can only form a partial kernel matrix derived from datasets where column sampling method is not applicable, the proposed method can be used to complete kernel matrices with missing values.

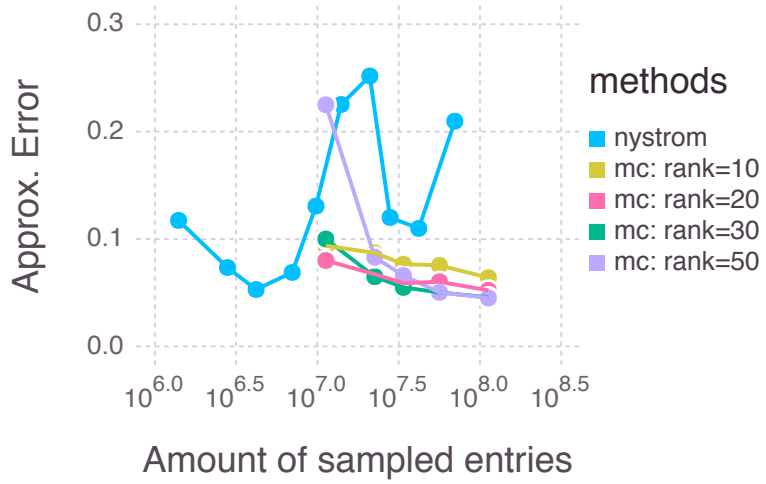


(a) Approximation Error vs. Rank

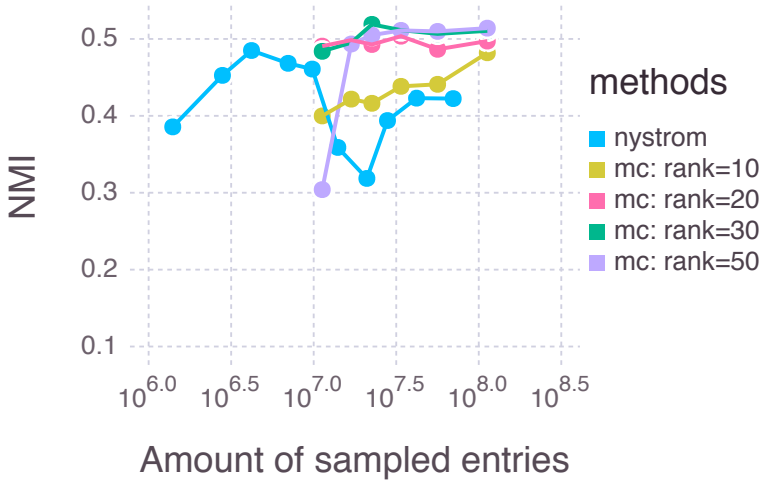


(b) NMI vs. Rank

Fig. 5.2.: Kernel approximation errors at top and clustering performance (Normalized Mutual Information with respect to the true class labels) at bottom for MNIST data set.

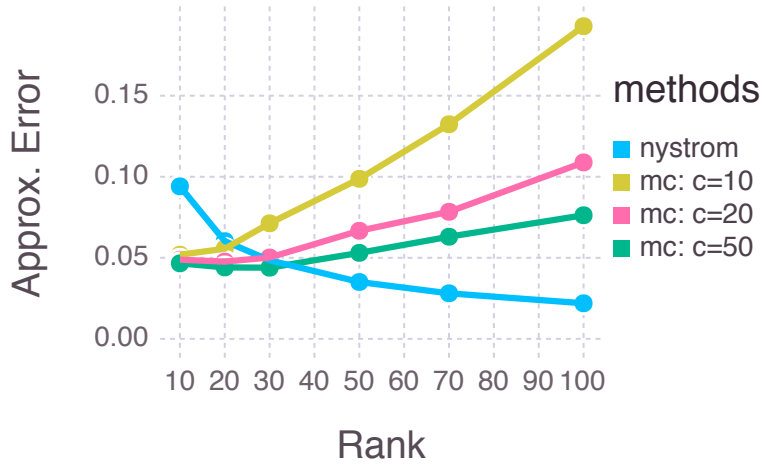


(a) Approximation Error vs. Amount of samples

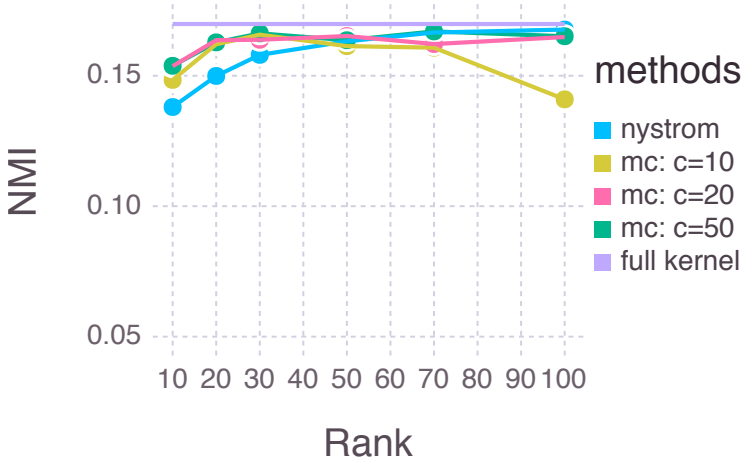


(b) NMI vs. Amount of samples

Fig. 5.3.: Kernel approximation errors at top and clustering performance (Normalized Mutual Information with respect to the true class labels) at bottom for MNIST data set with the amount of sample entries.



(a) Approximation Error vs. Rank



(b) NMI vs. Rank

Fig. 5.4.: Kernel approximation errors at top and clustering performance (Normalized Mutual Information with respect to the true class labels) at bottom for CIFAR-10 data set.

6. CONCLUSTIONS AND FUTURE WORK

In this thesis we consider the exploitation of low rank structures in the solutions and the objectives for clustering tasks in machine learning. We first study the low rank structures in the solution matrix of NEO-K-Means SDP problem. Second, we consider the low rank structures from the objective function of kernel-based clustering. We summarize our findings here, and discuss potential continuations of this research.

Regarding the low rank structures in the solution matrix (Chapter 3), we propose a practical method by optimizing a low-rank factorization of the NEO-K-Means SDP solution matrix directly in Chapter 4. The resulting optimization problem, i.e., NEO-K-Means LRSDP, is a non-convex, quadratically constrained, bound-constrained problem. When we employ high-quality optimization techniques, e.g., classical augmented Lagrangian approach (ALM) and two fast multiplier methods (PALM and ADMM), it frequently generates solutions that are *as good as the global optimal* from convex solvers (Chapter 4, Table 4.3). This approach takes much longer to run compared with simple Lloyd-like algorithm but considerably improve the quality and robustness of the clustering procedure for both vector clustering and community detection on graphs (Chapter 4, Figure 4.13 and Table 4.7).

The case of low rank structures in kernel k-means objective is inspired by matrix completion techniques. We propose a symmetric matrix completion approach to approximate the kernel matrix based on a subset of entries in Chapter 5. The low rank approximation of kernel matrices reduce both the computational complexity and the memory requirements for kernel k-means clustering even though we can only form a partial kernel matrix derived from datasets. We show empirically that the clustering performance is similar to the full kernel k-means (Chapter 5, Figure 5.2, 5.3 & 5.4).

Overall, the work in this thesis demonstrates that the low rank structures can benefit complex objective functions and problems by formulating the new optimization approaches that reduce the memory and computational cost.

In terms of future opportunities, we are attempting to identify a convergence guarantee for the ADMM method in the non-convex NEO-K-Means LRSDP case. This would put the fastest method we have for the optimization on firm theoretical ground. Currently, we use L-BFGS-B routine to solve the subproblems in the augmented Lagrangian framework. In future, we want to utilize the low rank Hessian methods to solve such subproblems. In terms of the clustering problem, we are exploring the integrality properties of the SDP relaxation itself. Our goal here is to show a result akin to that proved in Awasthi et al. [2015] about integrality in relaxations of the k-means objective. Finally, another goal we are pursuing involves understanding when our method can recover the partitions from an overlapping block-model with outliers. This should hopefully show that the optimization approaches have a wider recovery region than the simple iterative methods and provide a theoretical basis for empirically observed improvement.

REFERENCES

REFERENCES

David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007. ISBN 978-0-898716-24-5. http://dl.acm.org/citation.cfm?id=1283383.1283494.

Pranjal Awasthi, Afonso S. Bandeira, Moses Charikar, Ravishankar Krishnaswamy, Soledad Villar, and Rachel Ward. Relax, no need to round: Integrality of clustering formulations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 191–200, 2015. doi: 10.1145/2688073.2688116. http://doi.acm.org/10.1145/2688073.2688116.

Francis R. Bach and Michael I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 33–40, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102356. http://doi.acm.org/10.1145/1102351.1102356.

Arindam Banerjee, Chase Krumpelman, Joydeep Ghosh, Sugato Basu, and Raymond J. Mooney. Model-based overlapping clustering. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 532–537, 2005.

William E. Bishop and Byron M. Yu. Deterministic symmetric positive semidefinite matrix completion. In *Advances in Neural Information Processing Systems*, pages 2762–2770. 2014. http://papers.nips.cc/paper/5467-deterministic-symmetric-positive-semidefinite-matrix-completion.pdf.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37, 2004.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Samuel Burer and Renato D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95:329–357, 2003. ISSN 0025-5610. doi: 10.1007/s10107-002-0352-8.

Samuel Burer and Renato D.C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005. ISSN 0025-5610. doi: 10.1007/s10107-004-0564-1.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. doi: 10.1137/0916069.

Jian-Feng Cai, Emmanuel J. Cands, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010. doi: 10.1137/080738970. http://dx.doi.org/10.1137/080738970.

Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717, 2009. ISSN 1615-3383. doi: 10.1007/s10208-009-9045-5. http://dx.doi.org/10.1007/s10208-009-9045-5.

Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, pages 1–23, 2014. ISSN 0025-5610. doi: 10.1007/s10107-014-0826-5. http://dx.doi.org/10.1007/s10107-014-0826-5.

Radha Chitta, Rong Jin, Timothy C. Havens, and Anil K. Jain. Approximate kernel k-means: Solution to large scale kernel clustering. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 895–903, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020558.

Guillaume Cleuziou. An extended version of the k-means method for overlapping clustering. In *Proceedings of the International Conference on Pattern Recognition*, pages 1–4, 2008.

Michele Coscia, Giulio Rossetti, Fosca Giannotti, and Dino Pedreschi. Demon: A local-first discovery method for overlapping communities. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 615–623, 2012.

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: Spectral clustering and normalized cuts. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 551–556, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014118. http://doi.acm.org/10.1145/1014052.1014118.

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1115.

Chris Ding, Xiaofeng He, and Horst D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005.

André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems*, pages 681–687, 2001.

Maryam Fazel. Matrix rank minimization with applications. *PhD thesis, Stanford University*, 2002. http://faculty.washington.edu/mfazel/thesis-final.pdf.

Shai Fine and Katya Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, March 2002. ISSN 1532-4435. http://dl.acm.org/citation.cfm?id=944790.944812.

Mark Girolami. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, May 2002. ISSN 1045-9227. doi: 10.1109/TNN.2002.1000150. http://dx.doi.org/10.1109/TNN.2002.1000150.

Alex Gittens and Michael W. Mahoney. Revisiting the nyström method for improved large-scale machine learning. *Journal of Machine Learning Research*, 17(117):1–65, 2016. http://jmlr.org/papers/v17/gittens16a.html.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, volume 3. Johns Hopkins Press, 2012.

Thore Graepel. Kernel matrix completion by semidefinite programming. In *International Conference on Artificial Neural Networks*, pages 694–699. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-46084-8. doi: 10.1007/3-540-46084-5_113. http://dx.doi.org/10.1007/3-540-46084-5_113.

Michael C. Grant and Stephen P. Boyd. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, volume 371 of *Lecture Notes in Control and Information Sciences*, pages 95–110. 2008. ISBN 978-1-84800-154-1. doi: 10.1007/978-1-84800-155-8_7.

Michael C. Grant and Stephen P. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

David Gross. Recovering low-rank matrices from few coefficients in any basis. *IEEE Transactions on Information Theory*, 57(3):1548–1566, March 2011. ISSN 0018-9448. doi: 10.1109/TIT.2011.2104999. http://dx.doi.org/10.1109/TIT.2011.2104999.

David Haussler. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.

Yangyang Hou, Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon. Non-exhaustive, overlapping clustering via low-rank semidefinite programming. In *Proceedings of the 21th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 427–436, 2015.

Yangyang Hou, Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon. Fast multiplier methods to optimize non-exhaustive, overlapping clustering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 297–305, 2016.

Ryo Inokuchi and Sadaaki Miyamoto. LVQ clustering and SOM using a kernel function. In *IEEE International Conference on Fuzzy Systems*, volume 3, pages 1497–1500 vol.3, July 2004. doi: 10.1109/FUZZY.2004.1375395.

Alfredo N. Iusem, Teemu Pennanen, and Benar Fux Svaiter. Inexact variants of the proximal point algorithm without monotonicity. *SIAM Journal on Optimization*, 13 (4):1080–1097, 2003. doi: 10.1137/S1052623401399587. http://dx.doi.org/10.1137/S1052623401399587.

Prateek Jain, Raghu Meka, and Inderjit S. Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945. 2010. http://papers.nips.cc/paper/3904-guaranteed-rank-minimization-via-singular-value-projection.pdf.

Dae-Won Kim, Ki Young Lee, Doheon Lee, and Kwang H. Lee. Evaluation of the performance of clustering algorithms in kernel-induced feature space. *Pattern Recognition*, 38(4):607 – 611, 2005. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2004.09.006. http://www.sciencedirect.com/science/article/pii/S0031320304003656.

Donald E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, 1993.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Brian Kulis, Arun C. Surendran, and John C. Platt. Fast low-rank semidefinite programming for embedding and clustering. In *International Conference on Artifical Intelligence and Statistics*, pages 235–242, 2007.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.

Tarald O. Kvalseth. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):517–519, May 1987. ISSN 0018-9472. doi: 10.1109/TSMC.1987.4309069.

Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4), 2011.

Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist, 1998.

Jure Leskovec. Stanford network analysis project. http://snap.stanford.edu/, 2016.

Fredrik Lindsten, Henrik Ohlsson, and Lennart Ljung. Just relax and come clustering! A convexification of k-means clustering. Technical report, Linköpings universitet, 2011. http://liu.diva-portal.org/smash/get/diva2:650707/FULLTEXT01.pdf.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Haibing Lu, Yuan Hong, W. Nick Street, Fei Wang, and Hanghang Tong. Overlapping clustering with sparseness constraints. In *12th IEEE International Conference on Data Mining Workshops*, pages 486–494, 2012.

David Lusseau, Karsten Schneider, Oliver J. Boisseau, Patti Haase, Elisabeth Slooten, and Steve M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations: Can geographic isolation explain this unique trait? *Behavioral Ecology and Sociobiology*, 54(4):pp. 396–405, 2003. ISSN 03405443. http://www.jstor.org/stable/25063281.

Michael W. Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856. MIT Press, 2001.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014. ISSN 2167-3888. doi: 10.1561/2400000003.

Teemu Pennanen. Local convergence of the proximal point algorithm and multiplier methods without monotonicity. *Mathematics of Operations Research*, 27(1):170–191, 2002.

Stephen M. Robinson. Strongly regular generalized equations. *Mathematics of Operations Research*, 5(1):43–62, 1980. ISSN 0364765X. http://www.jstor.org/stable/3689393.

R. Tyrrell Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2): 97–116, 1976. ISSN 0364765X. http://www.jstor.org/stable/3689277.

R. Tyrrell Rockafellar and Roger J-B Wets. *Variational Analysis*. Springer, 2009.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge university press, 2004.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000. ISSN 0162-8828. doi: 10.1109/34.868688.

Si Si, Cho-Jui Hsieh, and Inderjit S. Dhillon. Memory efficient kernel approximation. In *Proceedings of the 31st International Conference on Machine Learning*, 2014. http://www.cs.utexas.edu/~ssi/meka/.

Alex J. Smola and Bernhard Schökopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, ICML '00, pages 911–918, San Francisco, CA, USA, 2000. ISBN 1-55860-707-2. http://dl.acm.org/citation.cfm?id=645529.657980.

Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P. Vlahavas. Multi-label classification of music into emotions. In *International Conference on Music Information Retrieval*, pages 325–330, 2008.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mulan: A Java library for multi-label learning. http://mulan.sourceforge.net/datasets.html, 2010.

Arnaud Vandaele, Nicolas Gillis, Qi Lei, Kai Zhong, and Inderjit S. Dhillon. Coordinate descent methods for symmetric nonnegative matrix factorization. *CoRR*, abs/1509.01404, 2015. http://arxiv.org/abs/1509.01404.

Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996. doi: 10.1137/1038003. http://dx.doi.org/10.1137/1038003.

Zaiwen Wen, Wotao Yin, and Yin Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361, 2012. ISSN 1867-2949. doi: 10.1007/s12532-012-0044-1.

Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon. Overlapping community detection using seed set expansion. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 2099–2108, 2013.

Joyce Jiyoung Whang, Inderjit S. Dhillon, and David F. Gleich. Non-exhaustive, overlapping k-means. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 936–944, 2015.

Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688. MIT Press, 2001.

Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 267–273, New York, NY, USA, 2003. ACM. ISBN 1-58113-646-3. doi: 10.1145/860435.860485. http://doi.acm.org/10.1145/860435.860485.

Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 587–596, 2013.

Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. Sdpnal+: A majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015. ISSN 1867-2957. doi: 10.1007/s12532-015-0082-6. http://dx.doi.org/10.1007/s12532-015-0082-6.

Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

Hongyuan Zha, Xiaofeng He, Chris Ding, Ming Gu, and Horst D. Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems*, pages 1057–1064, 2001.

Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007. ISSN 1573-1405. doi: 10.1007/s11263-006-9794-4. http://dx.doi.org/10.1007/s11263-006-9794-4.

Rong Zhang and Alexander I Rudnicky. A large scale clustering scheme for kernel k-means. In *Proceedings of the International Conference on Pattern Recognition*, volume 4, pages 289–292 vol.4, 2002. doi: 10.1109/ICPR.2002.1047453.

VITA

VITA

Yangyang Hou was born in Linyi, Shandong, China. She received a B.E. degree in computer science and technology from Shanghai Jiao Tong University in 2010 and an M.S. degree in computer science from Purdue University in 2012. Her research interests lie in matrix computation and optimization, especially with application to data mining, machine learning and network analysis.