Non-exhaustive, Overlapping k-means

Joyce Jiyoung Whang* Inderjit S. Dhillon* David F. Gleich†

Abstract

Traditional clustering algorithms, such as k-means, output a clustering that is disjoint and exhaustive, that is, every single data point is assigned to exactly one cluster. However, in real datasets, clusters can overlap and there are often outliers that do not belong to any cluster. This is a well recognized problem that has received much attention in the past, and several algorithms, such as fuzzy k-means have been proposed for overlapping clustering. However, most existing algorithms address either overlap or outlier detection and do not tackle the problem in a unified way. In this paper, we propose a simple and intuitive objective function that captures the issues of overlap and non-exhaustiveness in a unified manner. Our objective function can be viewed as a reformulation of the traditional k-means objective, with easy-to-understand parameters that capture the degrees of overlap and non-exhaustiveness. By studying the objective, we are able to obtain a simple iterative algorithm which we call NEO-K-Means (Non-Exhaustive Overlapping K-Means). Furthermore, by considering an extension to weighted kernel k-means, we can tackle the case of non-exhaustive and overlapping graph clustering. This extension allows us to apply our NEO-K-Means algorithm to the community detection problem, which is an important task in network analysis. Our experimental results show that the new objective and algorithm are effective in finding ground-truth clusterings that have varied overlap and non-exhaustiveness; for the case of graphs, we show that our algorithm outperforms state-of-the-art overlapping community detection methods.

1 Introduction

The traditional clustering problem involves exhaustively assigning each data point to a single group (or cluster) such that nearby points are also assigned to the same group. When separations between groups are clear and the data does not contain any outliers, then classical methods such as the k-means algorithm may succeed in correctly assigning points to groups in many realistic data models. In this paper, we revisit the clustering problem from the perspective of real-world data where groups still exist but may lack clean separations and the data contain outliers. In this setting, a more reasonable goal is a non-exhaustive, overlapping clustering where a data point may be outside of any cluster, and clusters are allowed to overlap with each other.

There is substantial prior research that has examined both of these problems individually – as would be expected for an area as well studied as clustering. For example, non-exhaustive clustering is highly related to outlier detection in a dataset, which itself has an extensive literature. Regarding overlap, both soft-clustering [1], which only makes probabilistic assignments, and overlapping clustering models are common [2]. Furthermore, many variations of the k-means algorithm have been proposed over the years [3] including recent work that considers overlapping clustering [4], [5], [6]. We discuss the related work in more detail in Section 4 in the context of the limitations of existing methods, and our new contribution. A key difference between our approach and existing ideas is that we treat the issues of non-exhaustiveness and overlap in a unified framework.

The result of our investigations is a novel improvement to the k-means clustering objective that enables a parametric trade-off between a clustering quality measure, overlap among the clusters, and non-exhaustiveness (the number of outliers not assigned to any group). To optimize the new objective, we present a simple iterative algorithm called non-exhaustive, overlapping k-means, or NEO-K-Means in short. Furthermore, by considering an extension to weighted kernel k-means, we can also tackle the problem of non-exhaustive, overlapping graph clustering. In the context of graph clustering, we extend a traditional normalized cut-based graph clustering objective to the non-exhaustive, overlapping setting, and show that this extended graph clustering objective is mathematically equivalent to the weighted kernel NEO-K-Means objective with a specific weight and kernel. This equivalence enables us to apply our NEO-K-Means algorithm to the overlapping community detection problem which is an important task in network analysis. Experimental results show that our new objective and algorithm are effective in finding ground-truth clusters; for the case of graphs, we show that our algorithm outperforms state-of-the-art overlapping community detection methods.

2 Non-exhaustive, Overlapping k-means

Developing a general purpose clustering algorithm is a challenging task. Even though many different clustering methods have been developed, k-means [7] is still one of the most popular clustering techniques due to its simplicity and empirical success [3]. We begin our

*Department of Computer Science, The University of Texas at Austin. Email: {joyce, inderjit}@cs.utexas.edu

†Department of Computer Science, Purdue University. Email: dgleich@purdue.edu

discussion by briefly reviewing k-means, and attempting our first and obvious extension of the k-means objective function. However, this obvious extension has serious limitations; after recognizing this, we propose our final objective function and a simple iterative algorithm for non-exhaustive, overlapping clustering.

2.1 k-means. Let us review the standard k-means setting. 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 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 pick the clusters to minimize the distance from the cluster centroid, or the mean of cluster, to each of its assigned data points. The k-means objective may 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 an NP-hard problem even for just two clusters. However, there is an efficient heuristic k-means algorithm [7], also known as Lloyd's algorithm, that proceeds by repeatedly assigning data points to their closest clusters and recomputing cluster centroids. This algorithm monotonically decreases the objective function.

2.2 An Intuitive, but Problematic, Extension. To extend the k-means objective function to a non-exhaustive, overlapping clustering setting, we first introduce an assignment matrix $U = [u_{ij}]_{n \times k}$ 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 U should be always equal to n because each data point should be assigned to exactly one cluster.

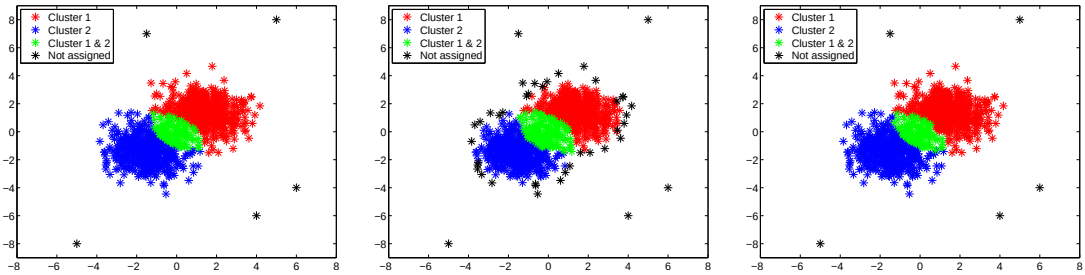
On the other hand, in a non-exhaustive, overlapping clustering, there are no restrictions on the assignment matrix U; there can be multiple ones in a row, meaning that a data point can belong to multiple clusters. Also, there can be rows of all zeros, meaning that some data points can have no membership in any cluster. So, now, we need to decide how many assignments we will make in U, i.e., we need to control the number of ones in U. One way to do this is to consider $U^T U$ which is a $k \times k$ matrix whose diagonal entries are equal to cluster sizes. The trace of $U^T U$ is equal to the sum of cluster sizes which is also equal to the total number of assignments in the assignment matrix U. To control how many additional assignments we will make in U, we add a constraint that the number of total assignments in U should be equal to $n + \alpha n$. We define our "first extension of k-means" as follows:

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

We require $0 \leq \alpha \leq (k-1)$ and note that $\alpha \ll (k-1)$ to avoid assigning each data point to every cluster. Similar to k-means, the above objective function is designed to minimize the sum of squared distances between every data point to its cluster centroid, but now the assignment is not necessarily restricted to be disjoint and exhaustive.

However, the seemingly reasonable objective function (2.2) has a limitation. To illustrate this, we test this objective function on synthetic data. As shown in the leftmost plot in Figure 1, we generate two ground-truth clusters which contain both overlap and outliers (details about this dataset are described in Section 5). Red data points are only assigned to cluster 1, blue data points are only assigned to cluster 2, green data points are assigned to both of the clusters, and black data points are not assigned to any cluster. Using a k-means like algorithm, we can optimize (2.2), and Figure 1 (b) shows the clustering result. There are 1,000 data points, and we set α=0.1 (which is the ground-truth value of α). So, 1,100 assignments are made in U. We can see that (2.2) fails to correctly recover the ground-truth clusters. Some data points that are relatively far from the cluster centers are not assigned to any cluster even though they are not outliers. This result indicates that just controlling the total number of assignments in U is not enough, and we need another constraint to correctly control the non-exhaustiveness. Based on these investigations, we now propose our final objective function in the following subsection.

2.3 The NEO-K-Means Objective. Recall that in our first extension of k-means objective function (2.2), we just added a constraint on the total number of assignments in the assignment matrix U, and it resulted in more false positive outliers than expected. To fix this problem, we introduce another important constraint which controls the degree of non-exhaustiveness. To state our new optimization problem, let us define the indicator function $\mathbb{I}\{exp\}$ to be $\mathbb{I}\{exp\} = 1$ if exp is true; 0 otherwise, and we let $\mathbf{1}$ denote a $k \times 1$ column vector having all the elements equal to one. Then, the vector $U\mathbf{1}$ denotes the number of clusters to which each data point belongs. Thus, $(U\mathbf{1})_i = 0$ means that $\mathbf{x}_i$ does not belong to any cluster. Now, by adding a non-exhaustiveness constraint to (2.2), we define our NEO-K-Means objective function as follows:



(a) Ground-truth clusters (b) First extension of k-means (c) NEO-K-Means

Figure 1: (a) Two ground-truth clusters are generated (n=1,000, α=0.1, β=0.005). Green points indicate overlap between the clusters, and black points indicate outliers. See Section 5 for details. (b) Our first extension of k-means objective function defined in (2.2) makes too many outlier assignments and fails to recover the ground-truth. (c) The NEO-K-Means objective defined in (2.3) adds an explicit term for non-exhaustiveness that enables it to correctly detect the outliers and find natural overlapping clustering structure which is very similar to the ground-truth clusters (α and β are automatically estimated by the heuristics discussed in Section 2.5).

(2.3)

$$\min_{U} \sum_{j=1}^{k}\sum_{i=1}^{n} u_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|^2, \text{ where } \mathbf{m}_j = \frac{\sum_{i=1}^{n} u_{ij}\mathbf{x}_i}{\sum_{i=1}^{n} u_{ij}}$$

$$\text{s.t. trace}(U^TU) = (1+\alpha)n,\ \textstyle\sum_{i=1}^{n} \mathbb{I}\{(U\mathbf{1})_i = 0\} \le \beta n.$$

We allow at most βn data points to be unassigned to any cluster, i.e., at most βn data points can be considered as outliers. We require $0 \le \beta n$ and note that $\beta n \ll n$ to cause most data points to be assigned to clusters. Specifically, by the definition of "outliers", βn should be a very small number compared to n. The parameters α and β offer an intuitive way to capture the degree of overlap and non-exhaustiveness; by "turning the knob" on these parameters, the user can explore the landscape of overlapping, non-exhaustive clusterings. If α=0 and β=0, the NEO-K-Means objective function is equivalent to the standard k-means objective presented in (2.1). To see this, note that setting the parameter β=0 requires every data point to belong to at least one cluster, while setting α=0 makes n assignments. Putting these together, the resulting clustering will be disjoint and exhaustive. Note that by setting α=0, objective (2.2) does not have this property.

To see if the objective function (2.3) yields a reasonable clustering, we test it on the same dataset we used in the previous subsection. Figure 1 (c) shows the result (α and β are automatically estimated by the heuristics discussed in Section 2.5). We see that NEO-K-Means correctly finds all the outliers, and produces very similar overlapping structure to the ground-truth clusters.

2.4 The NEO-K-Means Algorithm. We now propose a simple iterative algorithm which monotonically decreases the NEO-K-Means objective until it converges to a local minimum. Having the hard constraints in (2.3), we will make $n+\alpha n$ assignments such that at most βn data points can have no membership in any cluster. Note that the second constraint can be interpreted as follows: among n data points, at least $n - \beta n$ data points should have membership to some cluster. When our algorithm makes assignments of points to clusters, it uses two phases to satisfy these two constraints. Thus, each cluster $\mathcal{C}_j$ decomposes into two sets $\bar{\mathcal{C}}_j$ and $\hat{\mathcal{C}}_j$ that record the assignments made in each phase.

Algorithm 1 describes the NEO-K-Means algorithm. We first initialize cluster centroids. Any initialization strategies that are used in k-means may also be applied to our algorithm. Given cluster centroids, we compute all the distances $[d_{ij}]_{n\times k}$ between every data point and clusters, and for every data point, record its closest cluster and that distance. Then, the data points are sorted in an ascending order by the distance to its closest cluster. To ensure at least $n - \beta n$ data points are assigned to some cluster (i.e., to satisfy the second constraint), we assign the first $n - \beta n$ data points to their closest clusters. Let $\bar{\mathcal{C}}_j$ denote the assignments made by this step. Thus, $\sum_{j=1}^{k} |\bar{\mathcal{C}}_j| = n - \beta n$. Then, we make $\beta n + \alpha n$ more assignments by taking $\beta n + \alpha n$ minimum distances among $[d_{ij}]_{n\times k}$ such that $\mathbf{x}_i \notin \bar{\mathcal{C}}_j$. Let $\hat{\mathcal{C}}_j$ denote the assignments made by this step. Thus, $\sum_{j=1}^{k} |\hat{\mathcal{C}}_j| = \beta n + \alpha n$. Finally, $\sum_{j=1}^{k}(|\bar{\mathcal{C}}_j| + |\hat{\mathcal{C}}_j|) = n + \alpha n$. Once all the assignments are made, we update cluster centroids by recomputing the mean of each cluster. We repeat this procedure until the change in objective function is sufficiently small or the maximum number of iterations is reached. Note that the algorithm does not forcibly choose βn points as outliers; indeed, the number of outliers is *less* than βn and depends on the the distances between data points and their "secondary" clusters. We note that, if $\alpha = 0$ and $\beta = 0$, then the NEO-K-Means algorithm is identical to the standard k-means algorithm. Our algorithm

Algorithm 1 NEO-K-Means

Input: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n\}$, the number of clusters k, the maximum number of iterations t_{max}, α, β
Output: $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_k$

1: Initialize cluster means $\{\mathbf{m}_j\}_{j=1}^{k}$, $t = 0$.
2: **while** not converged and $t < t_{max}$ **do**
3: Compute cluster means, and then compute distances between every data point and clusters $[d_{ij}]_{n\times k}$.
4: Initialize $\mathcal{T} = \emptyset$, $\mathcal{S} = \emptyset$, $p = 0$, and $\bar{\mathcal{C}}_j = \emptyset$, $\hat{\mathcal{C}}_j = \emptyset$ $\forall j$.
5: **while** $p < (n + \alpha n)$ **do**
6: **if** $p < (n - \beta n)$ **then**
7: Assign $\mathbf{x}_{i^*}$ to $\bar{\mathcal{C}}_{j^*}$ such that $(i^*, j^*) = \underset{i,j}{\operatorname{argmin}}\, d_{ij}$ where $\{(i, j)\} \notin \mathcal{T}, i \notin \mathcal{S}$.
8: $\mathcal{S} = \mathcal{S} \cup \{i^*\}$.
9: **else**
10: Assign $\mathbf{x}_{i^*}$ to $\hat{\mathcal{C}}_{j^*}$ such that $(i^*, j^*) = \underset{i,j}{\operatorname{argmin}}\, d_{ij}$ where $\{(i, j)\} \notin \mathcal{T}$.
11: **end if**
12: $\mathcal{T} = \mathcal{T} \cup \{(i^*, j^*)\}$.
13: $p = p + 1$.
14: **end while**
15: $\forall j$, update clusters $\mathcal{C}_j = \bar{\mathcal{C}}_j \cup \hat{\mathcal{C}}_j$.
16: $t = t + 1$.
17: **end while**

guarantees monotonic decrease in the objective function as the following result shows:

THEOREM 1. *Algorithm 1 monotonically reduces the NEO-K-Means objective in (2.3) while satisfying the constraints specified by α and β.*

Proof. Let $J^{(t)}$ denote the objective at the t-th iteration. Then,

$$
\begin{aligned}
J^{(t)} &= \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j^{(t)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t)}\|^2 \\
&\geq \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \bar{\mathcal{C}}_j^{(t+1)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t)}\|^2 + \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \hat{\mathcal{C}}_j^{(t+1)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t)}\|^2 \\
&= \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j^{(t+1)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t)}\|^2 \text{ since } \mathcal{C}_j^{(t+1)} = \bar{\mathcal{C}}_j^{(t+1)} \cup \hat{\mathcal{C}}_j^{(t+1)} \\
&\geq \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j^{(t+1)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t+1)}\|^2 \text{ (property of centroids)} \\
&= J^{(t+1)}
\end{aligned}
$$

The first inequality follows from the update scheme used to form $\bar{\mathcal{C}}_j$ and $\hat{\mathcal{C}}_j$ by our algorithm (see steps 7 and 10 in Algorithm 1). Clearly the algorithm always maintains feasibility, i.e., the constraints specified by the parameters α and β are always satisfied.

2.5 Parameter Selection. We now discuss how to choose the parameters α and β. Since parameter selection is usually considered as a challenging task, many existing clustering algorithms leave this as an open problem. For example, in k-means-- algorithm [8], the number of outliers is a required input. Many other clustering methods (e.g., [1], [5], [6]) also have their own model parameters that should be set by a user. While some model parameters of other clustering methods tend to be non-intuitive to set or it can be hard to predict the effect of a particular parameter setting, the NEO-K-Means parameters α and β are intuitive parameters that allow users to specify how much overlap/non-exhaustiveness they want. So, users might be able to estimate these parameters from the domain knowledge. If any overlap and outlier statistics are unknown, we can estimate α and β values by using the heuristics discussed in the following subsections. The high level idea of our parameter estimation is to first run a disjoint, exhaustive clustering and perform cheap distance-based computation.

2.5.1 Choosing β. We first run a traditional k-means. Let d_i denote the distance between data point $\mathbf{x}_i$ and its closest cluster. We compute the mean (denoted by μ) and the standard deviation (denoted by σ) of d_i (i=1,$\cdots$,n). If a distance d_i is greater than $\mu + \delta\sigma$, then we consider the data point $\mathbf{x}_i$ as an outlier, where δ is a constant which controls how far from is it from the mean. We empirically observe that usually $\delta = 6$ leads to a reasonable estimate for β.

2.5.2 Choosing α. We use two different strategies for choosing α. We empirically observe that the first strategy is better when the overlap is small and the second strategy is better when the overlap is large.

The first strategy considers the distribution of distances in each cluster. For each $\mathcal{C}_j$, we consider the distances between the center of $\mathcal{C}_j$ and the data points which are assigned to $\mathcal{C}_j$, and compute the mean (denoted by μ_j) and the standard deviation (denoted by σ_j) of these distances. Then, for a data point $\mathbf{x}_l \notin \mathcal{C}_j$, we compute the distance between $\mathbf{x}_l$ and $\mathcal{C}_j$, denoted by d_{lj}. If d_{lj} is less than $\mu_j + \delta\sigma_j$ (usually, $-1 \leq \delta \leq 3.5$ gives a good estimate), we consider the data point $\mathbf{x}_l$ should be on the overlapped region. In this way, we can count the number of points which should be considered in the overlapped region, so, we can estimate α.

The second strategy considers normalized distances. Given a data point $\mathbf{x}_i$, let d_{ij} denote the distance between $\mathbf{x}_i$ and $\mathcal{C}_j$. We compute the normalized distance which is defined by $\bar{d}_{ij} = d_{ij} / \sum_{l=1}^{k} d_{il}$ (note that $\sum_{l=1}^{k} \bar{d}_{il} = 1$). Then, we count the number of $\bar{d}_{ij}$ whose value is less than $1/(k+1)$. Notice that if a data point is equidistant from every cluster, then the normalized distance is equal to $1/k$. To get a stronger bound, we set the threshold to be $1/(k+1)$. If the normalized distance is less than this threshold, we consider that the

data point should be in the overlapped region. In this way, we can estimate the amount of overlap.

2.6 Weighted Kernel NEO-K-Means. Now, let us discuss weighted kernel k-means. In kernel k-means, each data point is first mapped into a higher dimensional feature space, and then clustered using k-means in the feature space. A weighted version of kernel k-means [9] also has been introduced to differentiate each data point's contribution to the objective function by assigning a weight to each data point. Let ϕ denote a nonlinear mapping, and w_i denote a nonnegative weight for data point $\mathbf{x}_i$. Then, the weighted kernel k-means objective [9] is defined as follows:

$$\min_{\{\mathcal{C}_j\}_{j=1}^k} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j} w_i \|\phi(\mathbf{x}_i) - \mathbf{m}_j\|^2, \quad \text{where } \mathbf{m}_j = \frac{\sum_{\mathbf{x}_i \in \mathcal{C}_j} w_i \phi(\mathbf{x}_i)}{\sum_{\mathbf{x}_i \in \mathcal{C}_j} w_i}. \tag{2.4}$$

Most algorithms to optimize this objective exploit the well-known kernel trick to avoid forming the feature space explicitly and use the kernel matrix of inner products instead. Let us consider the weighted kernel NEO-K-Means objective function. Just like in (2.4), we introduce a nonlinear mapping ϕ and a weight w_i for each data point $\mathbf{x}_i$. Then the weighted kernel NEO-K-Means objective is defined as follows:

$$\min_{U} \quad \sum_{c=1}^{k} \sum_{i=1}^{n} u_{ic} w_i \|\phi(\mathbf{x}_i) - \mathbf{m}_c\|^2, \quad \text{where } \mathbf{m}_c = \frac{\sum_{i=1}^{n} u_{ic} w_i \phi(\mathbf{x}_i)}{\sum_{i=1}^{n} u_{ic} w_i}$$
$$\text{s.t.} \quad \text{trace}(U^T U) = (1+\alpha)n, \ \ \textstyle\sum_{i=1}^{n} \mathbb{I}\{(U\mathbf{1})_i = 0\} \leq \beta n. \tag{2.5}$$

The extension of NEO-K-Means to the weighted kernel case enables us to tackle the problem of non-exhaustive, overlapping graph clustering (also known as overlapping community detection), which we describe in the next section.

3 Graph Clustering using NEO-K-Means

In this section, we first review some of the traditional graph clustering objectives, and then present an extension of the traditional graph cut objectives to non-exhaustive, overlapping clustering setting. We show that this extended graph clustering objective is equivalent to the weighted kernel NEO-K-Means objective. Thus, we present a principled method to compute non-exhaustive, overlapping graph clustering by applying the NEO-K-Means algorithm.

3.1 Graph Clustering via Normalized Cut. Given a graph $G = (\mathcal{V}, \mathcal{E})$, the corresponding adjacency matrix is defined as $A = [a_{ij}]$ such that a_{ij} is equal to the edge weight between vertex i and j if there is an edge, and zero otherwise. We assume we are working with undirected graphs, where the matrix A is symmetric. We also assume that there is no self-loop in the graph, i.e., the diagonal elements of A are all zeros. The traditional graph partitioning problem seeks k pairwise disjoint clusters such that $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_k = \mathcal{V}$.

Normalized cut [10] is a popular measure to evaluate the quality of a graph partitioning or graph clustering. Let $\texttt{links}(\mathcal{C}_p, \mathcal{C}_q)$ denote the sum of edge weights between two sets $\mathcal{C}_p, \mathcal{C}_q$. Then, the normalized cut of a graph is defined as follows:

$$\texttt{NCut}(G) = \min_{\mathcal{C}_1, \mathcal{C}_2, \cdots \mathcal{C}_k} \sum_{j=1}^{k} \frac{\texttt{links}(\mathcal{C}_j, \mathcal{V} \backslash \mathcal{C}_j)}{\texttt{links}(\mathcal{C}_j, \mathcal{V})}. \tag{3.6}$$

Using a linear algebraic formulation, the normalized cut objective may be expressed as follows:

$$\begin{aligned} \texttt{NCut}(G) &= \min_{\mathbf{y}_1, \mathbf{y}_2, \cdots \mathbf{y}_k} \sum_{j=1}^{k} \frac{\mathbf{y}_j^T (D - A) \mathbf{y}_j}{\mathbf{y}_j^T D \mathbf{y}_j} \\ &= \max_{\mathbf{y}_1, \mathbf{y}_2, \cdots \mathbf{y}_k} \sum_{j=1}^{k} \frac{\mathbf{y}_j^T A \mathbf{y}_j}{\mathbf{y}_j^T D \mathbf{y}_j}, \end{aligned} \tag{3.7}$$

where D is the diagonal matrix of vertex degrees, and $\mathbf{y}_j$ denotes an indicator vector for cluster $\mathcal{C}_j$, i.e., $\mathbf{y}_j(i) = 1$ if a vertex v_i belongs to cluster $\mathcal{C}_j$, zero otherwise.

3.2 Extending Graph Cut Objectives to Non-exhaustive, Overlapping Clustering. Note that the traditional normalized cut objective (3.7) is for disjoint, exhaustive graph clustering. To consider non-exhaustive, overlapping graph clustering, we first introduce an assignment matrix $Y = [y_{ij}]_{n \times k}$ such that y_{ij}=1 if a vertex v_i belongs to cluster $\mathcal{C}_j$; y_{ij}=0 otherwise. Let $\mathbf{y}_j$ denote jth column of Y. Then, we can extend (3.7) to non-exhaustive, overlapping graph clustering by introducing the same constraints as in (2.3):

$$\max_{Y} \sum_{j=1}^{k} \frac{\mathbf{y}_j^T A \mathbf{y}_j}{\mathbf{y}_j^T D \mathbf{y}_j}$$
$$\text{s.t.} \, \text{trace}(Y^T Y) = (1+\alpha)n, \ \ \textstyle\sum_{i=1}^{n} \mathbb{I}\{(Y\mathbf{1})_i = 0\} \leq \beta n. \tag{3.8}$$

By adjusting α and β, we can control the degree of overlap and non-exhaustiveness. If α=0, and β=0, the above objective enforces disjoint and exhaustive clustering, thus is equivalent to the traditional normalized cut objective. We have focused on the normalized cut objective, but other graph clustering objectives (e.g., ratio association [10]) also can be extended to non-exhaustive, overlapping clustering using the same approach.

3.3 Equivalence of the Objectives. We now show that (2.5) is equivalent to (3.8) by defining an appropriate kernel and weights. Let $W = [w_{ii}]_{n \times n}$ denote a

diagonal weight matrix whose diagonal entries are equal to vertex weights, let K denote a kernel matrix such that $K_{ij} = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$, and let $\mathbf{u}_c$ denote the cth column of U. Then, (2.5) can be rewritten as follows:

$$\min_U \sum_{c=1}^{k} \sum_{i=1}^{n} u_{ic} w_i \|\phi(\mathbf{x}_i) - \mathbf{m}_c\|^2$$

$$= \min_U \sum_{c=1}^{k} \left(\sum_{i=1}^{n} u_{ic} w_i K_{ii} - \frac{\mathbf{u}_c{}^T W K W \mathbf{u}_c}{\mathbf{u}_c{}^T W \mathbf{u}_c} \right) \tag{3.9}$$

Let us define the kernel as $K \equiv \gamma W^{-1} + W^{-1} A W^{-1}$ where γ is a positive constant typically chosen to make K positive definite. Then (3.9) can be expressed as follows:

$$= \min_U \sum_{c=1}^{k} \left(\sum_{i=1}^{n} u_{ic} w_i \frac{\gamma}{w_i} - \frac{\mathbf{u}_c{}^T A \mathbf{u}_c}{\mathbf{u}_c{}^T W \mathbf{u}_c} \right)$$

$$= \min_U \left(\gamma(1+\alpha)n - \sum_{c=1}^{k} \frac{\mathbf{u}_c{}^T A \mathbf{u}_c}{\mathbf{u}_c{}^T W \mathbf{u}_c} \right)$$

$$= \max_U \sum_{c=1}^{k} \frac{\mathbf{u}_c{}^T A \mathbf{u}_c}{\mathbf{u}_c{}^T W \mathbf{u}_c} \tag{3.10}$$

Now, in (3.10), let us define the weight matrix as $W \equiv D$. Notice that $U = Y$ in (3.8). Putting these together, we can see that the weighted kernel NEO-K-Means objective (2.5) is equivalent to the extended normalized cut objective (3.8).

3.4 Algorithm. The equivalence between (3.8) and (2.5) implies that we can optimize the non-exhaustive, overlapping graph clustering objectives using the weighted kernel NEO-K-Means algorithm. The difference between standard k-means and weighted kernel k-means is how to compute the distance between a data point and clusters. Using our definitions of kernel and weights, the distance between a vertex v_i and cluster $\mathcal{C}_j$ can be quantified as follows:

(3.11)

$$\texttt{dist}(v_i, \mathcal{C}_j) = - \frac{2\,\texttt{links}(v_i, \mathcal{C}_j)}{\deg(v_i)\deg(\mathcal{C}_j)} + \frac{\texttt{links}(\mathcal{C}_j, \mathcal{C}_j)}{\deg(\mathcal{C}_j)^2} + \frac{\gamma}{\deg(v_i)} - \frac{\gamma}{\deg(\mathcal{C}_j)}$$

where $\deg(v_i)$ denotes the degree of vertex v_i, and $\deg(\mathcal{C}_j)$ denotes the sum of edge weights of vertices in $\mathcal{C}_j$. Then, Algorithm 1 can be applied to graph data by computing the distances using (3.11).

Popular software for graph partitioning, e.g., Graclus [9] and Metis [11], employs a multilevel approach. In this framework, an input graph is coarsened by merging nodes level by level. As a result, a series of smaller graphs are created. Once the input graph is coarsened into a small enough graph, an initial partitioning is performed. The clustering result of the coarsest level graph is first projected onto the graph at the level above it. Many different heuristics can be used at any of these coarsening or projection stages in order to improve the overall performance. The clustering is then refined through a refinement algorithm which plays the most important role in optimizing an objective function.

We also exploit the multilevel framework for non-exhaustive, overlapping graph clustering. While we use similar heuristics for coarsening and initial partitioning phases as in [9], we implement the weighted kernel NEO-K-Means algorithm for the refinement phase. Besides the multilevel refinement, any reasonable initialization can be directly given, and we can apply the weighted kernel NEO-K-Means algorithm to optimize the non-exhaustive, overlapping graph clustering objective.

4 Related Work

Both the aspects of overlap and non-exhaustiveness in clustering have been studied before, albeit rarely considered together in a unified manner as we do. We recognize that [12] also considers both overlap and non-exhaustiveness by modifying the traditional k-medoid algorithm, but their methodologies include complicated heuristics. A few recent papers study the clustering problem with outlier detection. In particular, [8] have proposed the k-means-- algorithm, which discovers clusters and outliers in a unified fashion; however it does not find overlapping clusters. We focus our discussion on overlapping clustering as that literature is the most closely related to our contribution. Soft clustering methods, such as fuzzy k-means [1], relax the binary assignment constraint and replace it by a probabilistic assignment. Thresholding these probabilities may result in both overlapping assignments to clusters and non-exhaustive partitions, although it is difficult to control these effects. There have been many attempts to extend k-means to overlapping clustering. For example, [4] defines OKM. However, it has been recognized that OKM tends to yield large overlap among the clusters, which the restricted OKM method [5] should address. Separately, [6] also has reformulated the OKM objective function by adding a sparsity constraint. On the other hand, from the Bayesian perspective, [2] proposed a generative model, called MOC, where each data point is assumed to be generated from an exponential family.

In the context of graph clustering, many different types of overlapping graph clustering, or overlapping community detection methods, have been presented. We recently proposed a seed expansion based community detection algorithm [13]. In this algorithm, good seeds are detected in a network, and the seeds are expanded using a personalized PageRank scheme. Compared to this seed-and-grow algorithm, our NEO-K-

Means algorithm adopts a more principled approach. Among the existing methods, the scalable alternatives include *demon* [14] and *bigclam* methods [15]; which we compare against. We also compare with *oslom* [16] which detects outliers and produces statistically significant overlapping communities. Many other methods are discussed in a recent survey [17], although the majority of successful approaches tend to suffer scalability issues on large networks like those we consider. Our derivation of the relationship between NEO-K-Means and overlapping community detection is inspired by [9] which showed the connection between k-means and graph partitioning.

5 Experimental Results

We show the experimental results of NEO-K-means on both vector data and graph data.

5.1 Evaluation Metrics. To evaluate the resulting set of clusters from each method, we focus primarily on the average F_1 score which measures how well each algorithm finds the ground-truth clusters. Formally, given a set of algorithmic clusters $\mathcal{C}$, and the ground-truth clusters $\mathcal{S}$, the average F_1 score is computed by averaging the F_1 score of the best match between each ground-truth cluster and algorithmic clusters [13]. The F_1 score of a single ground-truth cluster S_i is computed as the harmonic mean of precision($\mathcal{S}_i$) and recall($\mathcal{S}_i$):

$$\text{precision}(\mathcal{S}_i) = \frac{|\mathcal{C}_j \bigcap \mathcal{S}_i|}{|\mathcal{C}_j|}, \quad \text{recall}(\mathcal{S}_i) = \frac{|\mathcal{C}_j \bigcap \mathcal{S}_i|}{|\mathcal{S}_i|},$$

where $\mathcal{C}_j \in \mathcal{C}$, and $F_1(\mathcal{S}_i) = F_1(\mathcal{S}_i, \mathcal{C}_{j^*})$ such that $j^* = \underset{j}{\text{argmax}}\ F_1(\mathcal{S}_i, \mathcal{C}_j)$. In other words, we pick the algorithmic cluster $\mathcal{C}_j$ with the highest F_1 score. The average F_1, denoted by $\bar{F}_1$, is computed as follows:

$$\bar{F}_1 = \frac{1}{|\mathcal{S}|} \sum_{\mathcal{S}_i \in \mathcal{S}} F_1(\mathcal{S}_i).$$

5.2 Vector Data. We compare NEO-K-Means with fuzzy k-means [1] (denoted by *fuzzy*), MOC [2] (denoted by *moc*), OKM [4] (denoted by *okm*), restricted OKM [5] (denoted by *rokm*), and explicit/implicit sparsity constrained clustering [6] (denoted by *esp* and *isp*, respectively). For NEO-K-Means, we use our methodologies for estimating α and β (discussed in Section 2.5). We initialize all the methods using k-means with exactly the same centroids, run each of the algorithms 5 times, and pick the assignment matrix which leads to the best objective function value of each method. If an algorithm happens to return empty clusters or clusters that contain all the data points, we exclude these clusters when we compute F_1 score. Table 1 shows a summary of our vector datasets. 'dim.' denotes the dimension of the data points, and $|\bar{\mathcal{C}}|$ denotes the average cluster size.

5.2.1 Synthetic Data. Three synthetic datasets are generated from the Gaussian distribution. We first fix the cluster centroid of each cluster, and then draw $n-\beta n$ data points from the Gaussian distribution whose mean is the cluster centroid and covariance matrix is the identity. To create the ground-truth clusters, we first assign all the data points to its closest cluster, and then make additional assignments by taking minimum distances such that the total number of non-zeros of the ground-truth cluster matrix is equal to $n + \alpha n$. Finally, βn outliers are added to the data matrix, and these data points are not assigned to any cluster in the ground-truth cluster matrix.

Table 1: Vector datasets.

	n	dim.	$\lvert\bar{\mathcal{C}}\rvert$	outliers	k
synth1	5,000	2	2,750	0	2
synth2	1,000	2	550	5	2
synth3	6,000	2	3,600	6	2
yeast	2,417	103	731.5	0	14
music	593	72	184.7	0	6
scene	2,407	294	430.8	0	6

The first three rows of Table 2 shows the F_1 scores on the synthetic datasets. Because the datasets are simple, almost all the methods achieve high F_1 scores (above 0.9) except *moc*. On synth3, one of the clusters produced by *moc* contains all the data points in the cluster. As a result, *moc* gets a particularly low F_1 score on this dataset. We can see that NEO-K-Means achieves the highest F_1 score on all of the datasets. While synth1 does not contain outliers, synth2 and synth3 contain five and six outliers, respectively. We observe that NEO-K-Means finds the correct number of outliers, and perfectly finds all the outliers. On the other hand, all the other baseline methods do not have a functionality of non-exhaustive clustering, so they assign the outliers to some clusters.

5.2.2 Real-world Data. We use three real-world multi-label datasets from [18], which are presented in Table 1: 'yeast', 'music', and 'scene'. On these datasets, we treat each label as a ground-truth cluster. Details of each of these datasets are described in [18]. The F_1 scores are presented in the last three rows of Table 2. On 'yeast' dataset, among 14 clusters, *moc* returns 13 empty clusters, and one cluster that contains all the data points. So we cannot report the F_1 score of *moc*. We can see that NEO-K-Means always shows the best F_1 score while the algorithmic performance of the other methods varies. For instance, *rokm* is the worst for 'music', but is the second best for the 'scene' dataset.

5.3 Community Detection in Graph Data.

5.3.1 Karate Club Network. As an illustration of the method, we first apply our NEO-K-Means algorithm on Zachary's Karate Club network, which is a classical

Table 2: F_1 scores on vector datasets. NEO-K-Means (the last column) achieves the highest F_1 score across all the datasets while the performance of other existing algorithms is not consistent across all the datasets.

	moc	*fuzzy*	*esp*	*isp*	*okm*	*rokm*	NEO
synth1	0.833	0.959	0.977	0.985	0.989	0.969	**0.996**
synth2	0.836	0.957	0.952	0.973	0.967	0.975	**0.996**
synth3	0.547	0.919	0.968	0.952	0.970	0.928	**0.996**
yeast	-	0.308	0.289	0.203	0.311	0.203	**0.366**
music	0.534	0.533	0.527	0.508	0.527	0.454	**0.550**
scene	0.467	0.431	0.572	0.586	0.571	0.593	**0.626**

Table 3: Average normalized cut of each algorithm on large real-world networks. Lower normalized cut indicates better clustering. NEO-K-Means achieves the lowest normalized cut on all the datasets.

	demon	*oslom*	*bigclam*	*sse*	NEO
Amazon	0.555	0.221	0.392	0.116	**0.105**
DBLP	0.606	0.355	0.617	0.204	**0.188**
Flickr	-	-	0.596	0.515	**0.331**
LiveJournal	-	-	0.912	0.643	**0.373**

example for testing clustering algorithms. This network represents friendship relationships between 34 members in a karate club at a US university in 1970. In this network, node 1 and node 34 are known to be the instructor and the student founder of the club, respectively. These two nodes are central in the network forming two natural clusters around them. We run the NEO-K-Means with α=0.2, and β=0, so in this setting, the algorithm will make 41 assignments in total, i.e., 7 nodes can belong to both clusters (note that the number of common friends of node 1 and node 34 is four, so we are looking for something a little less than twice the obvious overlap). Figure 2 shows the clustering result of NEO-K-Means. We can see that the nodes that are assigned to both clusters have strong interactions with both of the underlying clusters.

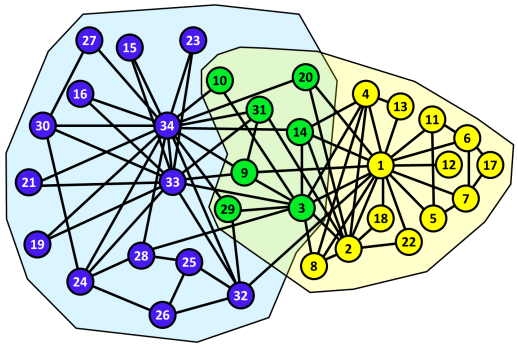


Figure 2: Clustering result of NEO-K-Means on Karate Club network. NEO-K-Means is able to reveal the natural underlying overlapping structure of the network.

5.3.2 Large Real-world Networks. For comparisons on large real-world networks, we use four real-world networks from [19], which are presented in Table 4. We compare the weighted kernel NEO-K-Means algorithm with state-of-the-art overlapping community detection methods: the seed set expansion method (denoted by *sse*) [13], *bigclam* [15], *demon* [14], and *oslom* [16]. For the comparison with *sse*, we use 'graclus centers' seeding method because it produces better average normalized cut values than 'spread hubs' seeding method. As in [13], we set k as 15,000 for Flickr and LiveJournal, and 25,000 for DBLP and Amazon. For NEO-K-Means, we set α=10, β=0 for Flickr and LiveJournal, and α=40, β=0.0001 for DBLP and Amazon. We choose small values of β and large values of α because (i) we expect that there are graph-based preprocessing techniques that remove obvious outliers (for example, connected components analysis) and (ii) real-world networks have vertices in many clusters.

Table 4: Graph datasets

	No. of vertices	No. of edges
Amazon	334,863	925,872
DBLP	317,080	1,049,866
Flickr	1,994,422	21,445,057
LiveJournal	1,757,326	42,183,338

We first compare the methods in terms of the average normalized cut. Recall that the normalized cut of a graph clustering is computed by (3.6). A lower normalized cut should indicate a better clustering. We compute the average normalized cut by dividing the total normalized cut by the returned number of clusters. Table 3 shows the average normalized cut of each algorithm. The *demon* and *oslom* programs fail on the Flickr and LiveJournal networks. We can see that NEO-K-Means achieves the lowest normalized cut value

Table 5: F_1 score of each algorithm on Amazon and DBLP. NEO-K-Means shows the highest F_1 score on Amazon, and comparable F_1 score with *sse* on DBLP.

	demon	*oslom*	*bigclam*	*sse*	NEO
Amazon	0.165	0.318	0.269	0.467	**0.490**
DBLP	0.137	0.132	0.151	**0.176**	0.174

Table 6: Average normalized cut and F_1 score of NEO-K-Means with different α and β on Amazon dataset.

	α=30, β=0	α=35, β=0	α=45, β=0	α=30, β=0.0001	α=35, β=0.0001	α=45, β=0.0001
ncut	0.107	0.104	0.104	0.106	0.104	0.104
F_1	0.488	0.490	0.490	0.488	0.490	0.490

across all the networks. This indicates that the weighted kernel NEO-K-Means is effective in optimizing the normalized cut objective. In Amazon and DBLP, the ground-truth communities are known. Table 5 shows F_1 scores on these networks. The NEO-K-Means method shows the best F_1 score on Amazon, but is slightly outperformed by *sse* on DBLP. However, we note the *sse* is a highly tuned, complicated heuristic, whereas the NEO-K-means algorithm is a more principled method. Table 6 shows the results of NEO-K-Means with different α and β on Amazon dataset. We can see that the results are not too sensitive to the particular α and β picked.

6 Conclusions & Future Work

We present a novel extension of the k-means formulation that simultaneously considers non-exhaustive and overlapping clustering called NEO-K-Means. The underlying objective and algorithm seamlessly generalize the classic k-means approach and enable a new class of applications. When we evaluate this new method on synthetic and real-world data, it shows the best performance in terms of finding the ground-truth clusters among a large class of state-of-the-art methods. Furthermore, we show that a weighted kernel k-means variation of the NEO-K-Means provides a principled way to find a set of overlapping communities in large scale networks. We conclude that NEO-K-Means is a useful algorithm to analyze much of the complex data emerging in current data-centric applications. We plan to extend the method to clustering with other Bregman divergences, and study simultaneous discovery of overlap and outlier detection for the co-clustering problem.

Acknowledgments This research was supported by NSF grants CCF-1117055 and CCF-1320746 to ID, and by NSF CAREER award CCF-1149756 to DG.

References

[1] J. C. Bezdek, R. Ehrlich, and W. Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 1984.

[2] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney. Model-based overlapping clustering. In *KDD*, 2005.

[3] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 2010.

[4] G. Cleuziou. An extended version of the k-means method for overlapping clustering. *ICPR*, 2008

[5] C.-E. ben N'Cir, G. Cleuziou, and N. Essoussi. Identification of non-disjoint clusters with small and parameterizable overlaps. In *ICCAT*, 2013.

[6] H. Lu, Y. Hong, W. N. Street, F. Wang, and H. Tong. Overlapping clustering with sparseness constraints. In *ICDM Workshops*, 2012.

[7] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 1982.

[8] S. Chawla and A. Gionis. k-means--: a unified approach to clustering and outlier detection. In *SDM*, 2013.

[9] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: a multilevel approach. *IEEE TPAMI*, 2007.

[10] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 1997.

[11] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *JPDC*, 1998.

[12] Y.-L. Chen and H.-L. Hu. An overlapping cluster algorithm to provide non-exhaustive clustering. *Eur. J. Oper. Res.*, 2006.

[13] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using seed set expansion. In *CIKM*, 2013.

[14] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi. Demon: a local-first discovery method for overlapping communities. In *KDD*, 2012.

[15] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, 2013.

[16] A. Lancichinetti, F. Radicchi, J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PLOS ONE*, 2011.

[17] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: the state of the art and comparative study. *ACM Comput. Surv.*, 2013.

[18] `http://mulan.sourceforge.net/datasets.html`

[19] `http://snap.stanford.edu/`