Clustering distributed data streams in peer-to-peer environments

Sanghamitra Bandyopadhyay [1], Chris Giannella, Ujjwal Maulik [2], Hillol Kargupta [*,3], Kun Liu, Souptik Datta

Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, United States

Abstract

This paper describes a technique for clustering homogeneously distributed data in a peer-to-peer environment like sensor networks. The proposed technique is based on the principles of the *K*-Means algorithm. It works in a localized asynchronous manner by communicating with the neighboring nodes. The paper offers extensive theoretical analysis of the algorithm that bounds the error in the distributed clustering process compared to the centralized approach that requires downloading all the observed data to a single site. Experimental results show that, in contrast to the case when all the data is transmitted to a central location for application of the conventional clustering algorithm, the communication cost (an important consideration in sensor networks which are typically equipped with limited battery power) of the proposed approach is

significantly smaller. At the same time, the accuracy of the obtained centroids is high and the number of samples which are incorrectly labeled is also small.

Keywords: Data mining; Data streams; Cluster analysis; Peer-to-peer

1. Introduction

Clustering [1] is a well-known and widely used exploratory data analysis technique. Most of the clustering algorithms that are available in the literature deal with data available at a single location. However, there exists many applications where data sources are distributed over a network and collecting the data at a central location before clustering is not a viable option. Sensor networks connected over wireless networks offer one such environment where centralized data clustering is difficult and often not scalable because of various reasons such as limited communication bandwidth and limited battery power supply for running the sensor nodes. Sensor networks communicate in a peer-to-peer (P2P) fashion which allows only local communication among the neighboring sensor nodes. This requires that the data analysis algorithms also communicate in a *P2P* mode and work in an asynchronous way [2,3]. No such clustering algorithm has so far been reported in the literature.

This paper offers a *P2P* clustering algorithm that is designed for environments like sensor networks monitoring continuous data streams at various nodes. The clustering technique, referred to as the *P2P K-Means* clustering, is based on the principles of the *K*-Means algorithm, and utilizes certain statistical bounds for estimating the error in computing the centroids of the clusters in a distributed manner vis-a-vis the centralized approach. The algorithm deals with the general scenario where each node contains a subset of the overall data to be clustered and the nodes observe the same set of attributes. Although most sensor network applications deal with continuous data streams, the paper does not directly address that. The *P2P K-Means* algorithm can be easily extended following the work by [4] in order to handle stream data. The objective of the current work is to develop a *P2P* version of the *K*-Means algorithms so that it can be used by the stream version of the *K*-Means algorithm [4] for clustering stream data in a sensor network.

Section 2 describes some of the clustering applications for sensor networks that motivated this research. Section 3 provides some background discussion on clustering, research issues in sensor networks, and related work in the domain of distributed clustering. The proposed clustering technique is described in Section 4. A detailed theoretical analysis of the proposed algorithm is carried out in Section 5. Section 6 provides the experimental results. Finally, Section 7 concludes the article.

2. Motivation

Consider a scene segmentation and monitoring application using sensor networks where the sensor nodes are equipped with audio, vibration, temperature and reflectance probes. The sensors are monitoring a given geographical region. The sensors are battery powered. Therefore, in the normal mode they are designed not to be very active. However, as soon as a node detects a possible change in the scenario, sensors must wake up, observe, reason and collaborate with each other in order to track and identify the object of interest in the scene. The observations are usually time-series data sampled at a device specific rate. The objective here is for the active nodes to collaborate with each other in order to segment the scene, identify the object of interest (e.g., a vehicle), and thereafter to classify it (e.g., pick-up truck). One of the popular approaches of segmentation is clustering. In this scenario, the requirement would be to cluster the data stream collected by the active sensors. Note that the data collected at the different sensors could be distributed either homogeneously (i.e., each node observes a subset of the data points) or heterogeneously (i.e., each node observes a subset of the attributes/features). Collaboration within the active sensors usually requires sending a window of observations from one node to another node and performing the clustering. The traditional framework of centralized clustering algorithms does not really scale very well in such distributed applications. For example, the centralized approach will be to send the data vectors to the base station (usually connected through a wireless network) and then performing the clustering there. This does not scale up in large sensor networks since data transmission consumes a lot of battery power and heavy data transmission over limited bandwidth channel may produce poor response time.

Outlier detection is another typical data mining task with application in monitoring chemical spillage and intrusion detection, among others, using mote-based sensor networks. Cluster analysis is one of the common approaches for outlier detection. Again, the traditional centralized clustering approaches are not likely to scale well in such scenario. Therefore, development of efficient distributed clustering algorithms that require only limited communication, while being designed for the peer-to-peer type of environment of the sensor networks, is of crucial importance. Such an attempt is reported in the present article.

3. Background

As already mentioned, distributed clustering of data streams in an energy efficient manner is an emerging research problem with significant applications in sensor networks. This section briefly reviews some of the existing clustering

techniques, with particular emphasis on data stream clustering, distributed clustering, and information processing in sensor networks. Issues in sensor networks, which make the problem unique, are also discussed.

3.1. Clustering

In clustering, a set of patterns, usually vectors in a multi-dimensional space, are organized into coherent and contrasted groups, such that patterns in the same group are similar in some sense and patterns in different groups are dissimilar in the same sense. The aim of any clustering technique is to evolve a proper partitioning of the dataset X (consisting of, say, n patterns, $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^D$) into a number, say K, of clusters $(C_1, C_2, \ldots, C_K)$ with cluster centers $V = \{v_1, v_2, \ldots, v_K\}$ such that some measure of goodness of the clusters is maximized. In general, there are three fundamental issues that must be addressed while clustering: whether there is any clustering tendency in the data or not; if yes, then what is a good method to find the clusters; and in what way can one validate the obtained partitions.

Traditionally, clustering algorithms have been classified into two categories: hierarchical and partitional [1,5]. Commonly used algorithms in the hierarchical category are the single linkage and complete linkage algorithms. K-Means is a widely used algorithm in the partitional class. More recent attempts in clustering large datasets in the context of data mining are BIRCH, CURE, DBSCAN, etc. [6].

As already mentioned, clustering streaming data in peer-to-peer environments of sensor networks offers new challenges to data mining researchers. Such an attempt is made in this article, where the well-known K-Means algorithm is utilized for this purpose. Analysis of the proposed algorithm uses standard statistical techniques to estimate the confidence that a locally computed centroid is within a certain distance of the correct centroid at each iteration of the algorithm.[4] The nodes are assumed to be synchronized partially, in that each node waits for a certain time interval, t, before proceeding with its computation.

3.2. Clustering data streams

A study on clustering under the data stream model of computation is undertaken in [4]. Given a sequence of points, the objective in [4] is to maintain a consistently good clustering of the sequences observed so far, using a small amount of memory and time. Only a summary of the past data is stored, leaving enough

[4] Similar techniques were used earlier in developing centralized clustering algorithms that require only a small number of samples to be seen while still guaranteeing that the model produced does not differ significantly from the one that would be obtained with infinite data [7].

memory for the processing of future data. The *K*-Median algorithm is used as the underlying clustering methodology. The authors provide constant factor approximation algorithms for the *K*-Median problem, which make a single pass over the data and use small space. Some other work in the data stream scenario may be found in [8–11].

In [12], a technique for iterative incremental clustering of time series data is described. The algorithm utilizes the multi-resolution property of wavelets and proceeds with multiple levels of approximation for clustering the real-life time series dataset with cardinalities ranging from 1000 to 8000, and lengths (or, the dimensionality of the data) ranging from 512 to 1024. The cluster centers at each level are initialized using those returned at the coarser level of representation. Wavelets are used for their ability to find a representation at a lower dimensionality that preserves the original information and describes the original shape of the time-series data as closely as possible. The *K*-Means and Expectation Maximization algorithms are used as the basis of clustering. Keogh et al. [13] make an interesting claim about clustering subsequence time series (STS) (produced using sliding windows over a single time series). STS is different from the problem of clustering time series [12], where the former refers to subsequences within the same time series, while the latter refers to a set of time series data. Keogh et al. [13] claim that in STS clustering the output is independent of the input. In particular, clusters extracted from these time series are forced to obey certain constraints that are pathologically unlikely to be satisfied by any dataset, and therefore the clusters are essentially random [13]. They provide a number of references (Ref. [13, p. 116]) of STS clustering.

3.3. Distributed clustering algorithms

In this section, we present an overview of various distributed clustering solutions proposed to date. We classify distributed clustering algorithms into two categories. The first group consists of methods requiring multiple rounds of message passing. These methods require a significant amount synchronization. The second group consists of methods that build local clustering models and transmit them to a central site (asynchronously). The central site forms a combined global model. These methods require only a single round of message passing, hence, modest synchronization requirements.

3.3.1. Multiple communication round algorithms

Dhillon and Modha [14] develop a parallel implementation of the *K*-Means clustering algorithm on distributed memory multi-processors (homogeneously distributed data). The algorithm makes use of the inherent data parallelism in the *K*-Means algorithm. Given a dataset of size n, they divide it into P blocks (each of size roughly n/P). During each iteration of *K*-Means, each site computes an update of the current K centroids based on its own data. The sites

broadcast their centroids. Once a site has received all the centroids from other sites it can form the global centroids by averaging.

Forman and Zhang [15] take an approach similar to the one presented in [14], but extend it to *K*-harmonic means. Note that the methods of [14,15] both start by partitioning then distributing a centralized dataset over many sites. This is different than the setting we consider: the data is never centralized—it is inherently distributed.

Kargupta et al. [16] develop a collective principal components analysis (PCA)-based clustering technique for heterogeneously distributed data. Each local site performs PCA, projects the local data along the principal components, and applies a known clustering algorithm. Having obtained these local clusters, each site sends a small set of representative data points to a central site. This site carries out PCA on this collected data (computes global principal components). The global principal components are sent back to the local sites. Each site projects its data along the global principal components and applies its clustering algorithm. A description of locally constructed clusters is sent to the central site which combines the cluster descriptions using different techniques including but not limited to nearest neighbor methods.

Klusch et al. [17] consider kernel density based clustering over homogeneously distributed data. They adopt the definition of a density based cluster from [18]. Data points which can be connected by an uphill path to a local maxima, with respect to the kernel density function over the whole dataset, are deemed to be in the same cluster. Their algorithm does not find a clustering of the entire dataset. Instead each local site finds a clustering of its *local* data based on the kernel density function computed over *all* the data. An approximation to the global kernel density function is computed at each site using sampling theory from signal processing. The sites must first agree upon a cube and a grid (of the cube). Each corner point can be thought of as a sample from the space (not the dataset). Then each site computes the value of its local density function at each corner of the grid and transmits the corner points along with their local density values to a central site. The central site computes the sum of all samples at each grid point and transmits the combined sample grid back to each site. The local sites can now independently estimate the global density function over *all* points in the cube (not just the corner points) using techniques from sampling theory in signal processing. The local sites independently apply a gradient-ascent based density clustering algorithm to arrive at a clustering of their local data. In principle, the approach in [17] could be extended to produce a global clustering by transmitting the local clusterings to a central site and then combining them. However, carrying out this extension in a communication efficient manner is non-trivial task and is not discussed by Klusch et al.

Eisenhardt et al. [19] develop a distributed method for document clustering (hence operates on homogeneously distributed data). They extend *K*-Means

with a "probe and echo" mechanism for updating cluster centroids. Each synchronization round corresponds to a *K*-Means iteration. Each site carries out the following algorithm at each iteration. One site initiates the process by marking itself as engaged and sending a probe message to all its neighbors. The message also contains the cluster centroids currently maintained at the initiator site. The first time a node receives a probe (from a neighbor site p with centroids C_p), it marks itself as engaged, sends a probe message (along with C_p) to all its neighbors (except the origin of the probe), and updates the centroids in C_p using its local data as well as computing a weight for each centroid based on the number of data points associated with each cluster. If a site receives an echo from a neighbor p (with centroids C_p and weights W_p), it merges C_p and W_p with its current centroids and weights. Once a site has received either a probe or echo from all neighbors, it sends an echo along with its local centroids and weights to the neighbor from which it received its *first* probe. When the initiator has received echos from all its neighbors, it has the centroids and weights which take into account all datasets at all sites. The iteration terminates.

While all algorithms in this section require multiple rounds of message passing, [16,17] require only two rounds. The others require as many rounds as the algorithm iterates (potentially many more than two).

3.3.2. Centralized ensemble-based methods

Many of the distributed clustering algorithms work in an asynchronous manner by first generating the local clusters and then combining those at the central site. These approaches potentially offer two nice properties in addition to lower synchronization requirements. If the local models are much smaller than the local data, their transmission will result is excellent message complexity. Moreover, sharing only the local models may be a reasonable solution to privacy constraints in some situations; indeed, a trade-off between privacy and communication cost is discussed in [20].

We present the literature in chronological order. Some of the methods were not explicitly developed for distributed clustering, rather for combining clusterings in a centralized setting to produce a better overall clustering. In these cases we discuss how well they seem to be adaptable to a distributed setting.

Johnson and Kargupta [21] develop a distributed hierarchical clustering algorithm on heterogeneously distributed data. It first generates local cluster models and then combines these into a global model. At each local site, the chosen hierarchical clustering algorithm is applied to generate local dendograms which are then transmitted to a central site. Using statistical bounds, a global dendogram is generated.

Lazarevic et al. [22] consider the problem of combining spatial clusterings to produce a global regression-based classifier. They assume homogeneously distributed data and that the clustering produced at each site has the same number of clusters. Each local site computes the convex hull of each cluster and transmits

the hulls to a central site along with regression model for each cluster. The central site averages the regression models in overlapping regions of the hulls.

Samatova et al. [23] develop a method for merging hierarchical clusterings from homogeneously distributed, real-valued data. Each site produces a dendogram based on local data, then transmits it to a central site. To reduce communication costs, they do not send a complete description of each cluster in a dendogram. Instead an approximation of each cluster is sent consisting of various descriptive statistics e.g., number of points in the cluster, average square Euclidean distance from each point in the cluster to the centroid. The central site combines the dendogram descriptions into a global dendogram description.

Strehl and Ghosh [24] develop methods for combining cluster ensembles in a centralized setting. They argue that the best overall clustering maximizes the average normalized mutual information over all clusters in the ensemble. However, they report that finding a good approximation directly is very time-consuming. Instead they develop three more efficient algorithms (dealing with similarity based clustering and hyper-graph based techniques) which are not theoretically shown to maximize mutual information, but are empirically shown to do a decent job.

Fred and Jain [25] report a method for combining clusterings in a centralized setting. Given N clusterings of n data points, their method first constructs an $n \times n$ *co-association matrix*. Next a merge algorithm is applied to the matrix using a single link, threshold-based hierarchical clustering technique. For each pair (i,j) whose co-association entry is greater than a predefined threshold, merge the clusters containing these points.

Jouve and Nicoloyannis [26] also develop a technique for combining clusterings. They use a related but different approach than those described earlier. They reduce the problem of combining clusterings to that of clustering a centralized categorical data matrix built from the clusterings and apply a categorical clustering algorithm (KEROUAC) of their own. The categorical data matrix has dimensions $n \times N$ and is defined as follows. Assume clustering i, $1 \leqslant i \leqslant N$, has clusters labeled $1, 2, \ldots, k_i$. The (j,i) entry is the label of the cluster (in the ith clustering) containing data point j. The KEROUAC algorithm does not require the user to specify the number of clusters desired in the final clustering. Hence, Jouve and Nicoloyannis' method does not require the desired number of clusters in the combined clustering to be specified.

In principle, the ideas in [24–26] can be adapted to heterogeneously distributed data (they did not address the issue), though the problem of building an accurate centralized representation in a message efficient manner must be addressed.

Merugu and Ghosh [20] develop a method for combining generative models produced from homogeneously distributed data (a generative model is a weighted sum of multi-dimensional probability density functions i.e., components). Each site produces a generative model from its own local data. Their

goal is for a central site to find a global model from a predefined family (e.g., multi-variate, 10 component Gaussian mixtures) which minimizes the average Kullback–Leibler distance over all local models. They prove this to be equivalent to finding a model from the family which minimizes the KL distance from the mean model over all local models (point-wise average of all local models). They assume that this mean model is computed at some central site. Finally the central site computes an approximation to the optimal model using an EM-style algorithm along with Markov-chain Monte-Carlo sampling. They did not discuss how the centralized mean model was computed. But, since the local models are likely to be considerably smaller than the actual data, transmitting the models to a central site seems to be a reasonable approach.

Januzaj et al. [27] extend a density-based centralized clustering algorithm, DBSCAN, by one of the authors to a homogeneously distributed setting. Each site carries out the DBSCAN algorithm, a compact representation of each local clustering is transmitted to a central site, a global clustering representation is produced from local representations, and finally this global representation is sent back to each site. A clustering is represented by first choosing a sample of data points from each cluster. The points are chosen such that: (i) each point has enough neighbors in its neighborhood (determined by fixed thresholds) and (ii) no two points lie in the same neighborhood. Then *K*-Means clustering is applied to all points in the cluster, using each of the sample points as an initial centroid. The final centroids along with the distance to the furthest point in their *K*-Means cluster form the representation (a collection point, radius pairs). The DBSCAN algorithm is applied at the central site on the union of the local representative points to form the global clustering. This algorithm requires an ϵ parameter defining a neighborhood. The authors set this parameter to the maximum of all the representation radii.

Methods [27,20,23] are representatives of the centralized ensemble-based methods. These algorithms focus on transmitting compact representations of a local clustering to a central site which combines to form a global clustering representation. The key to this class of methods is in the local model (clustering) representation. A good one faithfully captures the local clusterings, requires few messages to transmit, and is easy to combine.

Both the ensemble approach and the multiple communication round-based clustering algorithms usually work a lot better than their centralized counterparts in a distributed environment. This is well documented in the literature. The following section organizes the distributed clustering algorithms based on the data distribution (homogeneous vs. heterogeneous) they can handle.

3.3.3. Homogeneous vs. heterogeneous clustering literature

A common classification of DDM algorithms in the literature is: those which apply to homogeneously distributed (horizontally partitioned) or heterogeneously distributed (vertically partitioned) data. To help the reader sort out

	Homogeneous	Heterogeneous
Centralized Ensemble	[27], [22], [28], [23]	[21], [25], [26], [24]
Multiple Rounds of Communication	[14], [19], [15], [17]	[16]

Fig. 1. Four-way clustering algorithms classification.

the clustering methods we have described, we present the four-way classification seen in Fig. 1.

None of the techniques discussed in the previous section were developed in an environment where the network is dynamic and nodes have limited communication range. In this setting links may be coming up and down and nodes only communicate with their immediate neighbors. In order to adapt the above techniques to this type of environment, there could be two approaches. The first one would be to elect a leader among the nodes, and transmit the locally built models to the leader. Since communication is of peer-to-peer type, this would involve multi-hop transmission, and hence a much increased communication cost. The other approach would be to keep on combining the models incrementally within local neighborhoods, that would ultimately percolate to the entire network if it is stable for sufficiently long. One of the pitfalls to such incremental model combination would be the accumulation of error over many stages.

Our approach fits into the incremental model combination category. We are motivated by the low communication cost of the approaches of Dhillon and Modha [14], Forman and Zhang [15]. Indeed these approaches require nodes to only communicate centroids and cluster counts at each iteration of K-Means. However, their approaches require each node to communicate with all other nodes before proceeding to the next iteration. In effect full synchronization must occur at each iteration. Our method proposed in the present article represents a first step toward weakening this synchronization requirement (in addition to further reducing the communication cost).

Other distributed clustering algorithms could be adapted to fit the incremental model computation category. This represents an interesting, yet untried area of future work. For example, the method in [20] requires that nodes combine models in a single round of communication. In this respect, extending their approach offers advantages to our approach of extending K-Means. However, quantifying analytically the error in our approach seems more feasible.

3.4. Information processing in sensor networks

Sensor networks are finding increasing number of applications in many domains, including battle fields, smart buildings, and even the human body. Most sensor networks consist of a collection of light-weight (possibly mobile)

sensors connected via wireless links to each other or to a more powerful gateway node that is in turn connected with an external network through either wired or wireless connections. Sensor nodes usually communicate in a peer-to-peer architecture over an asynchronous network. In many applications, sensors are deployed in hostile and difficult to access locations with constraints on weight, power supply, and cost. Moreover, sensors must process a continuous (possibly fast) stream of data.

A sensor network is generally designed to perform some high level information processing tasks like environmental monitoring, tracking and classification. Monitoring time critical activities like earthquake and chemical spills, vehicle tracking, habitat monitoring are some of the typical application areas for sensor networks [28,29]. Several attempts have been made in the recent past to develop efficient hardware devices, networking them and designing routing protocols in sensor networks [30–33]. A recent survey on key research issues in sensor networks is available in [34,35]. Some interesting links to publications on sensor networks may be found in [36,37].

Development of algorithms that take into consideration the characteristics of sensor networks, viz., energy and computation constraints, network dynamics, and faults, constitute an area of current research. It is well known that communicating messages over a sensor network consume far more energy than processing it. For example, for Berkeley motes [38], the ratio of energy consumption for communication and computation is of the range of 1000–10,000 [39]. It is therefore hypothesized in [29] that given the characteristics of sensor networks, designing localized collaborative algorithms may offer the advantages of robustness and scalability. In this scenario, the nodes interact with others only within a restricted neighborhood, though attempting to achieve a desired global objective. The authors also propose directed diffusion, a model for describing localized algorithms [29,40]. Self-organization and self-configuration are beneficial in the sensor network scenario since the environment is often dynamic. Some work in developing localized, distributed, self-configuration mechanisms in sensor networks may be found in [41,42]. In [39], a collaborative signal and information processing (CSIP) approach is used for target tracking, which is modeled as a constrained optimization problem. CSIP is used for carefully selecting the embedded sensor nodes that participate in the sensor collaboration, balancing the information contribution of each against its resource consumption or potential utility for other users.

In designing algorithms for sensor networks, it is imperative to keep in mind that power consumption has to be minimized. Even gathering the distributed sensor data in a single site could be expensive in terms of battery power consumed. LEACH, LEACH-C, LEACH-F [31,43], and PEGASIS [44] are some of the attempts towards making the data collection task energy efficient. The issue of energy-quality trade-off has been studied in [45] along with a discussion on energy-quality scalability of three categories of commonly used

signal processing algorithms viz., filtering, frequency domain transforms and classification. In [46], Radivojac et al., develop an algorithm for intrusion detection in a supervised framework, where there are far more negative instances than positive (intrusions). A neural network based classifier is trained at the base station using data where the smaller class is over-sampled and the larger class is under-sampled [47]. An unsupervised approach to the outlier detection problem in sensor networks is presented in [48], where kernel density estimators are used to estimate the distribution of the data generated by the sensors, and then the outliers are detected depending on a distance based criterion. Detecting regions of interesting environmental events (e.g., sensing which regions in the environment have a chemical concentration greater than a threshold) has been studied in [49] under the assumptions that faults can occur in the equipments though they would be uncorrelated, while environmental conditions are spatially correlated.

Clustering the nodes of the sensor networks is an important optimization problem. Nodes that are clustered together can easily communicate with each other. Ghiasi et al. [50] have studied the theoretical aspects of this problem with application to energy optimization. They illustrate an optimal algorithm for clustering the sensor nodes such that each cluster (that is characterized by a master) is balanced and the total distance between the sensor nodes and the master nodes is minimized. Some other approaches in this regard are available in [51,52].

Algorithms for clustering the data spread over a sensor network are likely to play an important role in many sensor-network-based applications. Segmentation of data observed by the sensor nodes for situation awareness, detection of outliers for event detection are only a few examples that may require clustering algorithms. The distributed and resource-constrained nature of the sensor-networks demands a fundamentally distributed algorithmic solution to the clustering problem. Therefore, distributed clustering algorithms may come handy [53] when it comes to analyzing sensor network data or data streams.

Clustering in sensor-networks offers many challenges, including,

1. limited communication bandwidth,
2. constraints on computing resources,
3. limited power supply,
4. need for fault-tolerance, and
5. asynchronous nature of the network.

Distributed clustering algorithms for such a domain must address these challenges. Little work has been done in collaborative clustering of the data streams obtained at the sensor nodes in a distributed, energy-efficient manner. Clustering algorithms for distributed data and data streams generally involve a significant amount of communication (and hence are not energy/power

efficient), and also assume the existence of a central site. Often the algorithms are fully synchronized (e.g., [19]) and assume a fault-free network. Such assumptions do not hold in the case of sensor networks. An attempt to bridge this gap is made in the present article, where a clustering algorithm, *P2P K-Means*, is developed that takes the aforementioned issues into consideration to a large extent.

4. Distributed peer-to-peer *K*-Means clustering

This section describes the proposed *P2P K-Means* clustering algorithm in a peer-to-peer homogeneously distributed setting. Since the *P2P K-Means* clustering technique utilizes the principles of the conventional *K*-Means algorithm, it is described first. Subsequently, the distributed clustering problem is formally stated, followed by a description of the proposed method. A theoretical analysis of *P2P K-Means* is provided in the next section.

4.1. K-Means clustering technique

A K-partition of $X=\{x_1,x_2,\ldots,x_n\}$ can be conveniently represented by a $K\times n$ matrix called the partition matrix $U=[u_{ik}]$, $i=1,2,\ldots,K$, $k=1,2,\ldots,n$, where u_{ik} is either 0 or 1, indicating that the pattern x_k does not belong or belongs respectively to cluster i.

K-Means [1,5,54] is a widely used technique for crisp partitional clustering. The minimizing criterion used to characterize good clusters for *K*-Means partitions is defined as

$$J(U,V)=\sum_{i=1}^{K}\sum_{k=1}^{n}(u_{ik})D_{ik}^2(v_i,x_k). \tag{1}$$

Here U is a partition matrix; $V=\{v_1,\ldots,v_K\}$ represents K cluster centers; $v_i\in\mathbb{R}^N$; and $D_{ik}(v_i,x_k)$ is the distance from x_k to the v_i.

In the *K*-Means algorithms, the K initial seeds are first chosen randomly to represent the K centroids. Thereafter, the data points are assigned to the cluster of the closest centroid. This provides a partition matrix $U=[u_{ik}]$. After the assignment phase is over, the centroids are recomputed as follows:

$$v_i=\frac{\sum_{k=1}^{n}(u_{ik})x_k}{\sum_{k=1}^{n}u_{ik}},\quad 1\leqslant i\leqslant K. \tag{2}$$

A common strategy for generating the approximate solutions of the minimization problem in Eq. (1) is by iteratively performing the reassignment of the points to the closest centroids, and updating the centroids of the cluster with the mean of the points assigned to the same cluster.

4.2. Distributed clustering problem

Let there be p nodes in the system, $N_1, N_2, \ldots, N_p$. Node N_i, $i = 1,2,\ldots,p$, has a set of neighbors to which it is directly connected. This set is denoted by $Neigh^{(i)}$. Let $X = X^{(1)} \cup X^{(2)} \cup \cdots \cup X^{(p)}$ be the full dataset where $X^{(i)} \subset X$, $i = 1,2,\ldots,p$, denotes the subset of the data at node N_i. Let $X^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots, x_{n_i}^{(i)}\}$ be the set of n_i points in node i. The aim is to partition each dataset $X^{(i)}$, $i = 1,2,\ldots,p$, into K clusters that is consistent with the global clustering of X using a clustering algorithm A. In other words, let $X_j^{(i)}, i = 1,2,\ldots,p, j = 1,2,\ldots,K$, be the subset of points of $X^{(i)}$ that belongs to cluster j using algorithm A. Similarly, let X_j, $j = 1,2,\ldots,K$, be the subset of X that belongs to cluster j after application of A centrally. Then we would like the following to ideally hold:

$$X_j = X_j^{(1)} \bigcup X_j^{(2)} \cdots \bigcup X_j^{(p)}, \quad j = 1,2,\ldots,K. \tag{3}$$

4.3. P2P K-Means clustering

The proposed algorithm is an adaption of the standard K-Means algorithm. We assume that each node will spend the same amount of time, t, executing each iteration and that all nodes start the algorithm at the same time. As such, we are assuming all nodes are executing the same iteration.[5] Later we discuss ways of adapting the algorithm to weaken this assumption.

We assume that each node has the same random number generator, thus, each node generates the same set of K initial centroid seeds, $v_{1,0}^{(i)}, v_{2,0}^{(i)}, \ldots, v_{K,0}^{(i)}$. Thereafter, node N_i assigns each point in $X^{(i)}$ to the nearest centroid. Once the assignment of data points is complete, the centroids are updated to produce $w_{j,k}^{(i)}$, the dimension-wise mean of the points labeled j during iteration k. If the centroids have changed significantly (based on a user-defined parameter γ), then a flag $Centroids\_Changed^{(i)}$ is set. N_i polls a collection of other nodes for their centroids (how this collection is determined is discussed later). Some nodes may not respond because they have terminated. All other nodes will respond; let $Comb^{(i)}(k)$ denote the set of nodes that did respond. Each of these will have sent their centroids, cluster counts, and their *Centroids_Changed* flag. Node N_i then compute the weighted mean of the centroids it receives with its local centroid to produce its final set of centroids for this iteration. Meanwhile, N_i processes polling requests from other nodes as they arrive. Once t time units have elapsed N_i evaluates the termination condition. If *Centroids_Changed* is not set for N_i and also not set for all other polled nodes that responded, N_i terminates.

[5] The amount of time spent at each iteration is not conceptually important in the algorithm.

Node N_i must decide at iteration k which other nodes to poll. We examine two ways to do so: (1) N_i selects a random sample (without replacement) of all other nodes, (2) N_i selects all of its immediate neighbors in the network ($Neigh^{(i)}$). The first method is more complicated because routing is required. However, it does not bias the centroids computed by N_i and, intuitively speaking, allows N_i to develop a global view of the data more quickly. Moreover, it is easier to analyze the error since statistical bounds based on random sampling can be applied (discussed later).

Algorithm 4.3.1 *P2P K-Means*

Node N_i:
  $k = 0$ /* iteration count */
  Set $Centroids\_Changed^{(i)}$ to TRUE.
  Initialize K centroids, $v_{1,k}^{(i)}, v_{2,k}^{(i)}, \ldots, v_{K,k}^{(i)}$
  /* Same seeds are generated in all the p sites */
  Repeat (once t time units have elapsed, the "Until" condition is evaluated)
    Perform assignment of the local points to the K centroids, $v_{1,k}^{(i)}, v_{2,k}^{(i)}, \ldots, v_{K,k}^{(i)}$
    Let $u_{m,k}^{(i)}$ indicate the label of the centroid closest to the mth point, $x_m^{(i)}$.
    /* $n_{j,k}^{(i)}$ = number of points assigned to centroid j. */
    Update the centroids producing $w_{1,k}^{(i)}, w_{2,k}^{(i)}, \ldots, w_{K,k}^{(i)}$ where
    $w_{j,k}^{(i)} = \frac{1}{n_{j,k}^{(i)}} \sum_{x_m^{(i)} | u_{m,k}^{(i)} = j} x_m^{(i)}, j = 1, 2, \ldots K$
    If $(\sum_{j=1}^{K} \| w_{j,k}^{(i)} - w_{j,k-1}^{(i)} \|^2 > \gamma)$, /* centroids changed significantly */
      Set $Centroids\_Changed^{(i)}$ to TRUE.
    Else,
      Set $Centroids\_Changed^{(i)}$ to FALSE.
    Do the following steps in an interleaved fashion
      Poll a collection of nodes for their centroids and cluster counts.
        Details are given elsewhere as to which nodes are polled.
        Let $Comb^{(i)}(k)$ denote those that responded (they also sent their centroids, cluster counts, and $Centroids\_Changed$ flag).
      Process polling requests received from other nodes, N_a.
      Send $\{(w_{j,k}^{(i)}, n_{j,k}^{(i)}) : j = 1, \ldots, K\}$ and $Centroids\_Changed^{(i)}$ to N_a.
    Produce new centroids $v_{1,k+1}^{(i)}, \ldots, v_{K,k+1}^{(i)}$ where

$$v_{j,k+1}^{(i)} = \frac{\sum_{l \in (Comb^{(i)}(k) \cup \{i\})} w_{j,k}^{(l)} n_{j,k}^{(l)}}{\sum_{l \in (Comb^{(i)}(k) \cup \{i\})} n_{j,k}^{(l)}}, \text{for } j = 1, \ldots, K.$$

  Until [$Centroids\_Changed^{(l)}$ is FALSE for all $l \in (Comb^{(i)}(k) \cup \{i\})$]

It may be noted that the proposed *P2P K-Means* is not designed, for the present, to deal directly with continuous data streams in sensor networks, though it incorporates the peer-to-peer communication protocol. The objective of the current work is to develop a *P2P* version of the *K*-Means algorithms so that it can be easily used by the stream version of the *K*-Means algorithm [4] for clustering stream data in a sensor network. The algorithm in [4] works by maintaining a history of the medians and clustering those. The proposed algorithm may similarly be extended to handle data streams by doing the same at every node.

4.4. Relaxing synchronization

The assumption that all nodes are simultaneously executing the same iteration can be relaxed at the expense of decreased accuracy. In this subsection we sketch an adaption of the above algorithm for doing so. However, we leave careful analysis of this adaption to future work.

The basic idea is that node N_i would send its current iteration number, say k, along with its polling request. A polled node, N_a, may not necessarily be on the same iteration. If N_a is on a later iteration, then it responds with its centroids, cluster counts, and *Centroids_Changed* flag from iteration k (N_a must keep a complete history). If N_a is on an earlier iteration, then it simply waits until it reaches iteration k before responding. However, to avoid slowing the network down to the slowest node, N_i does not wait for N_a to respond beyond a fixed amount of time (time out). As such, N_i will use whatever centroids it receives in the available time.

5. Analysis of the *P2P K*-Means algorithm

In this section we provide results which allow a node, at a given iteration, to compute an upper-bound on the centroid error based on current run-time information. Such a bound can be thought of as a "gage" by which the node can measure the degree to which accuracy has been sacrificed at the expense of lowered communication cost.

We analyze the variant of our algorithm which uses a random sampling of nodes to update centroids. Our analysis is an adaption of that provided in [7] where *K*-Means clustering was used with a small number of samples in order to learn a model that does not differ significantly from the one that would be obtained with infinite data. Our notation is similar to that of [7] and we repeat some of their analysis to remain self-contained.

In our analysis, we bound the error between each centroid at each node and its corresponding centralized centroid. More formally, recall $v_{j,k+1}^{(i)}$ denotes the jth centroid produced at node N_i at the end of iteration k. Let $v_{j,k+1}^{*}$ denote the

jth centroid produced at the end of iteration k if all the data were first centralized and a standard K-Means algorithm run. We upper-bound the Euclidean distance between $v_{j,k+1}^{(j)}$ and $v_{j,k+1}^{*}$. This is referred to as the jth *centroid error* at the end of iteration k. Let $\epsilon_{j,k}^{(j)}$ denote the upper-bound we obtain on this error.

There are two sources of error that may crop up in the distributed algorithm. One is the error due to taking into account the data local to a node and all the nodes used to update centroids. This error is referred to as *sampling error* (to remain consistent with [7]), and bounds can be estimated using standard statistical techniques. The other source of error is due to the wrong assignment of data points to the clusters. This type of error is referred to as *assignment error*.

5.1. Some statistical tools

In bounding the sampling error we will make use of some standard statistical tools, namely, ratio estimators. For a more detailed exposition see [55, Chapters 2 and 6]. Note that our notation below differs somewhat from [55].

Consider finite populations $\{y_1,\ldots,y_p\} \in \mathbb{R}$ and $\{x_1,\ldots,x_p\} \in \mathbb{Z}_{>0}$ (positive integers). Let $(X_1, Y_1),\ldots,(X_s, Y_s)$ denote random variables representing s simultaneous samples from both populations without replacement.[6] Let r denote the population mean ratio

$$\frac{\sum_{i=1}^{p} y_i}{\sum_{i=1}^{p} x_i}$$

and R denote the sample mean ratio

$$\frac{\sum_{i=1}^{s} Y_i}{\sum_{i=1}^{s} X_i}.$$

The sample mean ratio is a standard estimator of the population mean ratio. A confidence interval can be derived based on $\mathrm{Var}(R)$, the sampling variance of R. For a large enough sample size, the following approximation is good for any $z > 0$:

$$\Pr(|r - R| \leqslant z\sqrt{\mathrm{Var}(R)}) \approx \mathrm{conf}(z)$$

where $\mathrm{conf}(z)$ represents the probability of a standard normal random variable being within z of zero. For example, $\mathrm{conf}(2.575)$ equals 0.99 and $\mathrm{conf}(3.3)$ equals 0.999. Moreover, for a large enough sample size the following is a good approximation of the sampling variance

[6] More formally, let $Z_1,\ldots,Z_s$ represent s random samples from population $\{1,\ldots,n\}$ (without replacement) and (X_j, Y_j) denote (x_l, y_l), respectively, with $l = Z_j$, for $1 \leqslant j \leqslant s$.

$$\mathrm{Var}(R) \approx \left(\frac{p-s}{sp}\right)\left(\frac{v(Y)^2 + R^2 v(X)^2 - 2Rv(X,Y)}{\overline{X}^2}\right),$$

where

- $\overline{X}$ and $\overline{Y}$ denote $\frac{\sum_{i=1}^{s} X_i}{s}$ and $\frac{\sum_{i=1}^{s} Y_i}{s}$, respectively;
- $v(X)^2$ and $v(Y)^2$ denote $\frac{\sum_{i=1}^{s}(X_i-\overline{X})^2}{s-1}$ and $\frac{\sum_{i=1}^{s}(Y_i-\overline{Y})^2}{s-1}$, respectively;
- $v(X, Y)$ denotes $\frac{\sum_{i=1}^{s}(Y_i-\overline{Y})(X_i-\overline{X})}{s-1}$.

Therefore, we have the following approximation (for any $z > 0$)

$$\Pr\left(|r-R| \leqslant z\sqrt{\left(\frac{p-s}{sp}\right)\left(\frac{v(Y)^2 + R^2 v(X)^2 - 2Rv(X,Y)}{\overline{X}^2}\right)}\right) \approx \mathrm{conf}(z). \tag{4}$$

Some comments are in order. First of all, the "large enough sample" caveat is based on the result that as s and n tend to infinity, the approximation (4) becomes exact in the limit under mild restrictions regarding the populations. Quoting [55, p. 153]: "As working rule, the large-sample results may be used if the sample size exceeds 30 and is also large enough so that the coefficients of variation of $\overline{X}$ and $\overline{Y}$ are both less than 10%". In our analysis to follow, we will be using (4) with modest sample sizes. As such, assessing the accuracy of our probability approximation is not a trivial matter. However, in this paper, we assume the approximations are good and leave the assessment of this assumption to future work.

Secondly, we have chosen different statistical techniques for our sampling error bounds than those of [7]. They used Hoeffding bounds [56] to bound the sampling error bound probability. Our situation requires modest sample sizes while theirs need not be modest being that we are sampling nodes in a network rather than data points. Hoeffding bounds are not very good at modest sample sizes. However, they provide more straight-forward probability bounds (they do not carry a "large enough sample" caveat).

5.2. Bounding the error at iteration zero

There are no assignment errors during iteration zero, since we assume that the initial seeds are the same at all the sites and equal to the seeds that would be chosen in the centralized case. However, the centroid j at node N_i produced at the end of iteration zero, $v_{j,1}^{(i)}$, may not be the same as $v_{j,1}^{*}$. The reason being that $v_{j,1}^{(i)}$ is formed by combining centroids from a sample (without replacement) of all other nodes, $Comb^{(i)}(0)$ (if all of the nodes were used and responded, no error would be produced).

Assume $s_k^{(i)} - 1$ nodes responded to their polling request, hence, $s_k^{(i)}$ nodes are used in computing the new centroid i.e., $|(Comb^{(i)}(k) \cup \{i\})| = s_k^{(i)}$. Next we lower-bound the probability that, for some fixed $t_1, \ldots, t_D > 0$, $\|v_{j,1}^{(i)} - v_{j,1}^*\|$ (the standard 2-norm) is not greater than $\sqrt{\sum_{d=1}^{D} t_d^2}$. Letting $v_{j,d,1}^{(i)}$ and $v_{j,d,1}^*$ denote the dth dimension of $v_{j,1}^{(i)}$ and $v_{j,1}^*$, respectively, we have

$$\Pr\left(\|v_{j,1}^{(i)} - v_{j,1}^*\| \leqslant \sqrt{\sum_{d=1}^{D} t_d^2}\right) = \Pr\left(\sum_{d=1}^{D}\left(v_{j,d,1}^{(i)} - v_{j,d,1}^*\right)^2 \leqslant \sum_{d=1}^{D} t_d^2\right) \tag{5}$$

$$\geqslant \Pr\left(\bigwedge_{d=1}^{D}\left[|v_{j,d,1}^{(i)} - v_{j,d,1}^*| \leqslant t_d\right]\right) \tag{6}$$

$$= 1 - \Pr\left(\bigvee_{d=1}^{D}\left[|v_{j,d,1}^{(i)} - v_{j,d,1}^*| > t_d\right]\right) \tag{7}$$

$$\geqslant 1 - \sum_{d=1}^{D} \Pr\left(|v_{j,d,1}^{(i)} - v_{j,d,1}^*| > t_d|\right). \tag{8}$$

Now our results from Section 5.1 can be applied directly to approximate the probability that $|v_{j,d,1}^{(i)} - v_{j,d,1}^*|$ is greater than t_d. Let $\{n_{j,0}^{(a)}\}$ and $\{w_{j,d,0}^{(a)} n_{j,0}^{(a)}\}$ for $a = 1, \ldots, p$ denote the x and y populations in Section 5.1, respectively. Similarly, let $\{n_{j,0}^{(l)}\}$ and $\{w_{j,d,0}^{(l)} n_{j,0}^{(l)}\}$ for $l \in (Comb^{(i)}(0) \cup \{i\})$ denote the X and Y samples, respectively. Let $v_{j,d,0}(Y)$, $v_{j,0}(X)$, $R_{j,d,0} v_{j,d,0}(X, Y)$, and $\overline{X}_{j,0}$ denote the respective sample statistics. For some user-defined $z > 0$, by (4) it follows that

$$\Pr\left(|v_{j,d,1}^* - v_{j,d,1}^{(i)}| > z\sqrt{\left(\frac{p - s_k^{(i)}}{s_k^{(i)} p}\right)\left(\frac{v_{j,d,0}(Y)^2 + R_{j,d,0}^2 v_{j,0}(X)^2 - 2R_{j,d,0} v_{j,d,0}(X, Y)}{\overline{X}_{j,0}^2}\right)}\right)$$

is approximately $1 - \mathrm{conf}(z)$.

Letting t_d equal

$$z\sqrt{\left(\frac{p - s_k^{(i)}}{s_k^{(i)} p}\right)\left(\frac{v_{j,d,0}(Y)^2 + R_{j,d,0}^2 v_{j,0}(X)^2 - 2R_{j,d,0} v_{j,d,0}(X, Y)}{\overline{X}_{j,0}^2}\right)},$$

(8) implies that the sampling error (total error at iteration zero), $\|v_{j,1}^{(i)} - v_{j,1}^*\|$, is bounded above by

$$\epsilon_{j,0}^{(i)} = \sqrt{\sum_{d=1}^{D} z^2\left[\left(\frac{p - s_k^{(i)}}{s_k^{(i)} p}\right)\left(\frac{v_{j,d,0}(Y)^2 + R_{j,d,0}^2 v_{j,0}(X)^2 - 2R_{j,d,0} v_{j,d,0}(X, Y)}{\overline{X}_{j,0}^2}\right)\right]} \tag{9}$$

with probability approximately bounded below by $1 - D(1 - \mathrm{conf}(z))$.

5.3. Bounding the error at iteration $k > 0$

Here we must account for both assignment and sampling errors. A point x at node N_i is correctly assigned during iteration k if it were assigned to the same centroid as it would have been assigned if the centralized algorithm were run. Formally stated, x is correctly assigned if the following holds:

$$\operatorname{argmin}\left\{\|x - v_{j,k}^{(i)}\| : 1 \leqslant j \leqslant K\right\} = \operatorname{argmin}\{\|x - v_{j,k}^{*}\| : 1 \leqslant k \leqslant K\}. \quad (10)$$

Let $\overline{v}_{j,k+1}^{(i)}$ denote the jth centroid at node i produced if no assignment errors were made during iteration k, by any node. The total error equals

$$\|v_{j,k+1}^{(i)} - v_{j,k+1}^{*}\| \leqslant \|v_{j,k+1}^{(i)} - \overline{v}_{j,k+1}^{(i)}\| + \|\overline{v}_{j,k+1}^{(i)} - v_{j,k+1}^{*}\|. \quad (11)$$

The first term is the assignment error and the second the sampling error. Let $S_{j,k}^{(i)+}$ be the set of points x at node N_i which were incorrectly assigned to centroid j (during iteration k), i.e., points for which the left-hand side of Eq. (10) equals j but the right-hand side does not. Let $S_{j,k}^{(i)-}$ be the set of points at node N_i which were incorrectly assigned to a different centroid than j, i.e., left-hand side of (10) does not equal j but the right-hand side does. Let $n_{j,k}^{(i)+}$ denote $|S_{j,k}^{(i)+}|$ and $n_{j,k}^{(i)-}$ denote $|S_{j,k}^{(i)-}|$. Let $w_{j,k}^{(i)+}$ and $w_{j,k}^{(i)-}$ denote the dimension-wise average of $S_{j,k}^{(i)+}$ and $S_{j,k}^{(i)-}$, respectively.

Clearly the incorrect assignments made by node N_i on its own data must be included in the assignment error. In addition, the incorrect assignments made by all nodes which contribute their centroids and cluster counts ($Comb^{(i)}(k)$) must be included too. It can be seen that the dth dimension of the assignment error, $\|v_{j,k+1}^{(i)} - \overline{v}_{j,k+1}^{(i)}\|$, equals

$$\begin{aligned}
&\left|\frac{\sum_{l\in(Comb^{(i)}\cup\{i\})} w_{j,d,k}^{(l)} n_{j,k}^{(l)}}{\sum_{l\in(Comb^{(i)}\cup\{i\})} n_{j,k}^{(l)}} - \frac{\sum_{l\in(Comb^{(i)}\cup\{i\})}\left(w_{j,d,k}^{(l)} n_{j,k}^{(l)} - w_{j,d,k}^{(l)+} n_{j,k}^{(l)+} + w_{j,d,k}^{(l)-} n_{j,k}^{(l)-}\right)}{\sum_{l\in(Comb^{(i)}\cup\{i\})}\left(n_{j,k}^{(l)} - n_{j,k}^{(l)+} + n_{j,k}^{(l)-}\right)}\right| \\
&= \left|\frac{-v_{j,d,k+1}^{(i)}\left(\sum_l n_{j,k}^{(l)+}\right) + v_{j,d,k+1}^{(i)}\left(\sum_l n_{j,k}^{(l)-}\right) + \left(\sum_l w_{j,d,k}^{(l)+} n_{j,k}^{(l)+}\right) - \left(\sum_l w_{j,d,k}^{(l)-} n_{j,k}^{(l)-}\right)}{\sum_l (n_{j,k}^{(l)} - n_{j,k}^{(l)+} + n_{j,k}^{(l)-})}\right| \\
&= \left|\frac{\sum_l\left(\left[n_{j,k}^{(l)+}\left(w_{j,d,k}^{(l)+} - v_{j,d,k+1}^{(i)}\right)\right] - \left[n_{j,k}^{(l)-}\left(w_{j,d,k}^{(l)-} - v_{j,d,k+1}^{(i)}\right)\right]\right)}{\sum_l\left(n_{j,k}^{(l)} - n_{j,k}^{(l)+} + n_{j,k}^{(l)-}\right)}\right| \\
&= \left|\frac{\sum_l\left(\sum_{x\in S_{j,k}^{(l)+}}[x_d - v_{j,d,k+1}^{(i)}] - \sum_{x\in S_{j,k}^{(l)-}}[x_d - v_{j,d,k+1}^{(i)}]\right)}{\sum_l\left(n_{j,k}^{(l)} - n_{j,k}^{(l)+} + n_{j,k}^{(l)-}\right)}\right|.
\end{aligned}$$

For any point x in $(S_{j,k}^{(l)+} \cup S_{j,k}^{(l)-})$ let $\Delta x_{j,d,k}$ denote $(x - v_{j,d,k+1}^{(i)})$ if $x \in S_{j,k}^{(l)+}$ and denote $-(x - v_{j,d,k+1}^{(i)})$ if $x \in S_{j,k}^{(l)-}$. From the previous equation it follows that

$$|v_{j,d,k+1}^{(i)} - \overline{v}_{j,d,k+1}^{(i)}| = \frac{|\sum_l \sum_{x \in (S_{j,k}^{(l)+} \cup S_{j,k}^{(l)-})} \Delta x_{j,d,k}|}{|\sum_l (n_{j,k}^{(l)} - n_{j,k}^{(l)+} + n_{j,k}^{(l)-})|}. \tag{12}$$

This equation precisely quantifies the assignment error, but is not applicable for our purposes because $S_{j,k}^{(l)+}$ and $S_{j,k}^{(l)-}$ cannot be computed from run-time information. Computing these precisely requires knowledge of the true centroid, $v_{j,k}^*$. Next we develop bounds on the above equation involving only available run-time information. Consider a point x at node N_l which was incorrectly assigned to cluster j during iteration k when it should have been assigned to j'. Therefore,

$$\|x - v_{j',k}^{(l)}\| - \epsilon_{j',k}^{(l)} \leqslant \|x - v_{j,k}^{(l)}\| + \epsilon_{j,k}^{(l)}.$$

Note, this is a necessary but not sufficient condition. It may hold when x was correctly assigned. Let $\overline{S}_{j,k}^{(l)+}$ be the set of points at node N_l which satisfy the above condition for some j' (clearly, $S_{j,k}^{(l)+} \subseteq \overline{S}_{j,k}^{(l)+}$). Let $\overline{n}_{j,k}^{(l)+}$ denote $|\overline{S}_{j,k}^{(l)+}|$. Clearly, $n_{j,k}^{(l)+} \leqslant \overline{n}_{j,k}^{(l)+}$.

Similarly, consider a point x at node N_l which was incorrectly assigned to cluster j' when it should have been assigned to j. Therefore,

$$\|x - v_{j,k}^{(l)}\| - \epsilon_{j,k}^{(l)} \leqslant \|x - v_{j',k}^{(l)}\| + \epsilon_{j',k}^{(l)}.$$

Let $\overline{S}_{j,k}^{(l)-}$ be the set of points at node N_l which satisfy the above condition for some j' (clearly, $S_{j,k}^{(l)-} \subseteq \overline{S}_{j,k}^{(l)-}$).

Since $n_{j,k}^{(l)} \geqslant n_{j,k}^{(l)-} \geqslant 0$ and $n_{j,k}^{(l)+} \leqslant \overline{n}_{j,k}^{(l)+}$, then the denominator of (12) is lower-bounded by

$$\left| \sum_l (n_{j,k}^{(l)} - \overline{n}_{j,k}^{(l)+}) \right|. \tag{13}$$

To upper-bound the numerator, let $S_{j,k}(i)$ and $\overline{S}_{j,k}(i)$ denote the following expressions (respectively)

$$\bigcup_{l \in (Comb^{(i)} \cup \{i\})} (S_{j,k}^{(l)+} \cup S_{j,k}^{(l)-}),$$

$$\bigcup_{l \in (Comb^{(i)} \cup \{i\})} (\overline{S}_{j,k}^{(l)+} \cup \overline{S}_{j,k}^{(l)-}).$$

Moreover, let $PS_{j,d,k}(i)$ denote the set of points x in $S_{j,k}(i)$ for which $\Delta x_{j,k,d}$ is non-negative and let $NS_{j,d,k}(i)$ denote those for which $\Delta x_{j,k,d}$ is negative. Similarly, define $\overline{PS}_{j,d,k}(i)$ and $\overline{NS}_{j,d,k}(i)$ with respect to $\overline{S}_{j,k}(i)$. The numerator of (12) can be upper-bounded as follows:

$$\left|\sum_{l}\sum_{x\in(S_{j,k}^{(l)+}\cup S_{j,k}^{(l)-})}\Delta x_{j,d,k}\right| = \left|\sum_{x\in S_{j,k}(i)}\Delta x_{j,d,k}\right| \tag{14}$$

$$\leqslant \max\left\{\sum_{x\in PS_{j,d,k}(i)}\Delta x_{j,k,d}, \sum_{x\in NS_{j,d,k}(i)}|\Delta x_{j,k,d}|\right\} \tag{15}$$

$$\leqslant \max\left\{\sum_{x\in \overline{PS}_{j,d,k}(i)}\Delta x_{j,k,d}, \sum_{x\in \overline{NS}_{j,d,k}(i)}|\Delta x_{j,k,d}|\right\}. \tag{16}$$

Inequality (15) is due to the fact that the absolute value of a sum is upper-bounded by either the sum of the positive terms or the sum of the absolute value of the negative terms. Inequality (16) is due to the fact that $PS_{j,d,k}(i) \subseteq \overline{PS}_{j,d,k}(i)$ and $NS_{j,d,k}(i) \subseteq \overline{NS}_{j,d,k}(i)$. Putting together (12), (13), and (16), we get our desired bound on the dth dimension of the assignment error

$$|v_{j,d,k+1}^{(i)} - \overline{v}_{j,d,k+1}^{(i)}| \leqslant \frac{\max\left\{\sum_{x\in\overline{PS}_{j,d,k}(i)}\Delta x_{j,k,d}, \sum_{x\in\overline{NS}_{j,d,k}(i)}\Delta|x_{j,k,d}|\right\}}{\left|\sum_{l\in(Comb^{(i)}\cup\{i\})}(n_{j,k}^{(l)} - \overline{n}_{j,k}^{(l)+})\right|}. \tag{17}$$

We have the following bound on the assignment error:

$$\|v_{j,k+1}^{(i)} - \overline{v}_{j,k+1}^{(i)}\| \leqslant \sqrt{\sum_{d=1}^{D}\left[\frac{\max\left\{\sum_{x\in\overline{PS}_{j,d,k}(i)}\Delta x_{j,k,d}, \sum_{x\in\overline{NS}_{j,d,k}(i)}\Delta|x_{j,k,d}|\right\}}{|\sum_{l\in(Comb^{(i)}\cup\{i\})}(n_{j,k}^{(l)} - \overline{n}_{j,k}^{(l)+})|}\right]^2}. \tag{18}$$

Let $UAE_{j,k+1}^{(i)}$ denote the right-hand side of inequality (18).

The sampling error, $\|\overline{v}_{j,k+1}^{(i)} - v_{j,k+1}^{*}\|$, is bounded by applying a similar argument as in the iteration zero case. The only difference lies in the fact that the error approximate probability bound must take into account the error probabilities at all previous iterations. Since node sampling was done independently at each iteration, we must multiply the error bounds at all iterations. Hence, the sampling error on centroid j at node N_i at the end of iteration k is bounded above by

$$SAE_{j,k+1}^{(i)} = \sqrt{\sum_{d=1}^{D} z^2\left[\left(\frac{p - s_k^{(i)}}{s_k^{(i)} p}\right)\left(\frac{v_{j,d,k}(Y)^2 + R_{j,d,k}^2 v_{j,k}(X)^2 - 2R_{j,d,k}v_{j,d,k}(X,Y)}{\overline{X}_{j,k}^2}\right)\right]}$$

with probability approximately bounded below by $[1 - D(1 - \mathrm{conf}(z))]^{k+1}$. Therefore, from inequality (11) we have the following upper-bound on the total centroid error on centroid j at node N_i at the end of iteration k

Table 1
Bounding values for different numbers of iterations

k	
5	0.97
10	0.94
20	0.9
40	0.81
80	0.66
160	0.44

$$\epsilon_{j,k}^{(i)} = UAE_{j,k}^{(i)} + SAE_{j,k+1}^{(i)}$$

which holds with probability approximately bounded below by $[1 - D(1 - \mathrm{conf}(z))]^{k+1}$.

The value of z is assumed to be defined by the user at the start of iteration zero. To get an idea as to this approximate bound, consider $D = 5$ (five dimensional data) and $z = 3.3$. Table 1 shows the values for $[1 - D(1 - \mathrm{conf}(z))]^{k+1}$ as the number of iterations increases.

5.4. Computing the bound

Recall that the goal of the analysis is to provide node N_i with a "gage" by which the centroid error can be assessed. Specifically N_i can compute an upper-bound on the centroid error at the end of iteration k, $\epsilon_{j,k}^{(i)}$, which holds with probability approximately bounded below by $[1 - D(1 - \mathrm{conf}(z))]^{k+1}$. Both of these quantities can be computed using information local to N_i, information about sampled nodes sent to N_i, and the previous iteration error upper-bound, $\epsilon_{j,k-1}^{(i)}$.

Clearly, node N_i can compute all terms in $SAE_{j,k+1}^{(i)}$ using the centroids and cluster counts it received from other nodes assuming that the total number of nodes in the network p is known. If this assumption is not made, p can be replaced with any upper-bound $\hat{p}$. The result is an upper bound on $SAE_{j,k+1}^{(i)}$ as needed.

For node N_i to compute $UAE_{j,k+1}^{(i)}$ some additional information need be exchanged between nodes. First of all each node N_l must send to N_i, $\overline{n}_{j,k}^{(l)+}$ along with its jth centroid and cluster count. Second of all N_l must communicate (for each dimension d) its share of $\sum_{x \in \overline{PS}_{j,d,k}(i)} \Delta x_{j,d,k}$ and $\sum_{x \in \overline{NS}_{j,d,k}(i)} \Delta x_{j,d,k}$. To do so, N_i first sends $v_{j,k+1}^{(i)}$ to N_l, from which N_l can compute its shares and send them to N_i.

The total amount of additional communication required to compute the error bound is approximately a factor of two.

6. Experimental results

We carried out two sets of experiments. First we applied a random sampling-based approach, i.e., in each iteration, every node (site) updates its cluster centroids based on the cluster information from randomly selected nodes over the whole network. Then we experimented with the scenario where random sampling is replaced by the deterministic immediate neighbor-based approach, i.e., each node updates its cluster centroids by only considering the information from its immediate neighbors. The proposed algorithms are compared against centralized *K*-Means clustering algorithm in terms of accuracy and communication cost. Next, we describe the experimental environment, dataset used and performance measurement before reporting the experimental results.

We ran our experiments in a simulated environment where the number of computing nodes varies from 10 to 50. We adopted two kinds of network topologies. The first one is represented as a totally random graph without disconnected components, i.e., each node can find a path to all the other nodes in the network. The second one is again a random graph, however, we kept the number of immediate neighbors of each node constant, 5 for example. In each iteration of the proposed algorithm, each node runs local *K*-Means on its own data first. Then, based on the predefined topology, each node updates its cluster centroids by taking into consideration the cluster information from some other nodes (either randomly selected nodes or immediate neighbors) in the network. The process starts with the same set of *K* initial seeds, and it iterates till the termination condition is met.

We conducted all the experiments with synthetic datasets generated from multi-variate Gaussian distribution. For the purpose of visualization, we generated the data with only two attributes. A sketch of the data is shown in Fig. 2. The same dataset is non-uniformly (uniformly resp.) distributed over different nodes.

To measure the accuracy of our proposed algorithm, we compare the cluster membership of each data point from *P2P K-Means* with the membership of the same data point from centralized *K*-Means. Since both the centralized and the distributed clustering algorithms start from the same set of initial seeds, a particular data point is expected to be labeled by the same cluster index in the end. We report the total number of mislabeled data points as a percentage of the size of the dataset. Besides, we also report the average Relative Euclidean Distance between each centroid found by *P2P K-Means* and the corresponding one found by centralized *K*-Means. This index is computed via

$$\frac{1}{p}\sum_{i=1}^{p}\frac{\|W_j^{(i)} - W_j^{\text{Central}}\|}{\|W_j^{\text{Central}}\|} \times 100\% \quad (j = 1, \ldots, K), \tag{19}$$

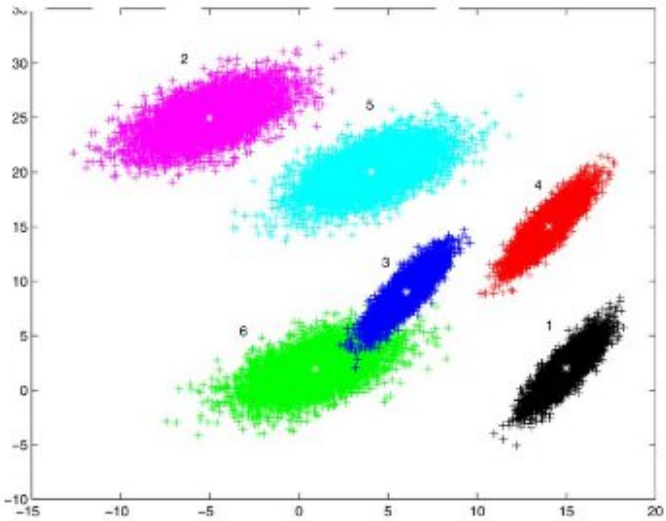

Fig. 2. Multi-variate Gaussian dataset with 37,800 data points and six clusters.

where $\|\cdot\|$ denotes the vector two norm, $W_j^{(i)}$ is the jth centroid in node i found by *P2P K-Means*, W_j^{Central} is the jth centroid found by centralized K-Means, and p is the total number of nodes in the network. For sake of simplicity, in the later part of this section, we use 'Relative Euclidean Distance' or 'RED' to denote this index.

To evaluate the communication complexity, we compare the total number of messages passing over the *P2P* network with the messages required to transfer all the data points into one single node which holds the biggest chunk of the data. We view the transmission of a single floating point number as one message. During each iteration, every node receives K centroids from each of its partners V_i (either randomly selected nodes or immediate neighbors), together with K cluster counts associated with these centroids. The total number of messages received by a single node U can thus be computed through $\sum_{j=1}^{k}\sum_{i=1}^{N}(K \times (D+1) \times ShortestPath(U, V_i))$, where k is the number of iterations, N is the number of partners that communicate with node U, K is the number of clusters, D is the dimensionality of the data point, and $ShortestPath(U, V_i)$ is the length of the shortest path from U to V_i. We assume the distance between two directly connected nodes is 1. Thus the communication complexity of *P2P K-Means* is roughly bounded by $O(pkNK(D+1)P)$ where p is the total number of nodes in the network, P is the maximum length of the all the shortest paths from the current node to its partners.

6.1. Random sampling-based approach

6.1.1. Non-uniformly distributed data

We varied the total number of nodes in the peer-to-peer network from 10 to 50. The same dataset is randomly, non-uniformly distributed over all the nodes. The topologies of all the systems are always random and there are no disconnected components. For *P2P K-Means*, in each iteration, each node U updates its cluster centroids based on the cluster information from a fixed number (5 in our experiment) of randomly selected nodes V_i in the whole network. The reason for keeping such fixed number of neighbors is to make the number of messages exchanged in different-size network comparable. The experiments were conducted several times, and Table 2 shows the results on average.

The experimental results show that *P2P K-Means* clusters most of the data points as correctly as the centralized *K*-Means does. The total number of mislabeled data points, when expressed as a percentage of the size of the original dataset, does not exceed 0.2%. The accuracy is roughly constant as the nodes in the *P2P* network changes. Compared with the communication cost required by centralizing the data into one single node, *P2P K-Means* only needs a very small portion of message passing, i.e., from 3.75% to 19.54%. Note that as the number of nodes in the network increases, more and more nodes join in the centroids update process, so the total number of messages also grows up, which verifies the theoretical analysis of the message complexity in the previous section.

Table 3 gives the average Relative Euclidean Distance (RED) (computed via Eq. (19)) between each local centroid and the centralized centroid. It can be seen that most of the REDs are only around 1%, which means the distributed clusters are very close to the corresponding centralized ones. Note that the REDs for centroids 3 and 6 are much higher than all the others. This is because these two clusters have lots of overlapping data points, so it is a little hard to separate them. Moreover, non-uniform sampling of the dataset over *P2P* network results in a very skewed distribution of the data in some nodes, which

Table 2
Accuracy and communication cost of *P2P K-Means* clustering with random sampling of nodes

	#Nodes								
	50	45	40	35	30	25	20	15	10
Max #points/node	1523	1506	1935	2336	1957	2688	3035	5266	5625
#Messages	14,175	11,778	10,665	9639	7674	6648	5148	3744	2412
Message rate (%)	19.54	16.23	14.87	13.59	10.71	9.47	7.40	5.75	3.75
#Mislabeled points	32	48	20	30	37	23	9	6	6
Error rate (%)	0.08	0.13	0.05	0.08	0.10	0.06	0.02	0.02	0.02

The original dataset (37,800 data points) is non-uniformly distributed over different-size P2P networks.

Table 3
Relative Euclidean Distance (RED) between each local centroid and the centralized centroid with random sampling of nodes

	#Nodes								
	50	45	40	35	30	25	20	15	10
RED for centroid 1	1.10	1.04	0.80	0.80	0.79	0.45	0.65	0.80	0.37
RED for centroid 2	0.86	0.71	0.78	0.80	0.65	0.55	0.53	0.62	0.35
RED for centroid 3	1.40	1.32	1.17	1.29	1.35	1.17	0.84	0.79	0.56
RED for centroid 4	0.73	0.74	0.72	0.65	0.66	0.65	0.62	0.47	0.32
RED for centroid 5	1.15	1.17	0.79	0.90	0.81	0.72	0.76	0.66	0.53
RED for centroid 6	10.99	9.88	7.71	7.78	15.41	13.81	7.24	7.35	3.84

The original dataset (37,800 data points) is non-uniformly distributed over different-size P2P networks.

Table 4
Accuracy and communication cost of *P2P K-Means* clustering with random sampling of nodes

	#Samples						
	3	5	7	9	11	13	15
#Messages	7668	12,960	17,820	24,282	28,152	33,552	38,808
Message rate (%)	10.57	17.86	24.56	33.47	38.80	46.24	53.49
#Mislabeled points	15	18	8	6	4	2	2
Error rate (%)	0.0397	0.0476	0.0212	0.0159	0.0106	0.0053	0.0053

The original dataset (37,800 data points) is non-uniformly distributed over 50 nodes. The sample size varies from 3 to 15.

hurts the local clustering results pretty much. To investigate the performance of the algorithm with regard to the sample size (the number of randomly selected partners), we fixed the initial seeds, the number of nodes in the network (50 nodes), as well as the topology, then we changed the number of sample nodes from 3 to 15. Table 4 reports the results. It shows that as the number of samples increases, the communication cost grows up, and the error rate drops down in the long run, though not strictly due to the randomness.

6.1.2. *Uniformly distributed data*

When data is uniformly distributed over different nodes, each node contains equal number of data points. The total number of points per node thus decreases as the number of nodes goes up. Since the distribution of data in each node is almost the same as the distribution of the original dataset, the number of iterations required to converge in *P2P K-Means* is supposed to be lower than that of the non-uniformly distributed scenario, which means less messages over the network; the quality of distributed clusters should also be a little better than the previous settings. Tables 5 and 6 validate the claims.

Table 5
Accuracy and communication cost of *P2P K-Means* clustering with random sampling of nodes

	#Nodes								
	50	45	40	35	30	25	20	15	10
#Points/node	756	840	945	1080	1260	1512	1890	2520	3780
#Messages	12,960	11,448	10,260	9000	7524	6480	5148	3744	2412
Message rate (%)	17.49	15.48	13.92	12.25	10.30	8.93	7.17	5.31	3.54
#Mislabeled points	44	31	14	8	12	14	17	4	5
Error rate (%)	0.12	0.08	0.04	0.02	0.03	0.04	0.04	0.01	0.01

The original dataset (37,800 data points) is uniformly distributed over different-size P2P networks.

Table 6
Relative Euclidean Distance between each local centroid and the centralized centroid with random sampling of nodes

	#Nodes								
	50	45	40	35	30	25	20	15	10
RED for centroid 1	0.96	1.00	0.75	0.75	0.63	0.60	0.67	0.45	0.39
RED for centroid 2	0.89	0.85	0.65	0.70	0.70	0.43	0.49	0.36	0.36
RED for centroid 3	1.55	1.39	1.06	1.25	0.78	0.89	1.14	0.49	0.67
RED for centroid 4	0.74	0.74	0.72	0.55	0.47	0.45	0.38	0.40	0.30
RED for centroid 5	0.99	0.89	0.89	0.76	0.90	0.62	0.69	0.34	0.52
RED for centroid 6	10.21	7.39	7.06	8.15	7.21	6.22	6.10	3.94	3.75

The original dataset (37,800 data points) is uniformly distributed over different-size P2P networks.

6.2. *Deterministic immediate neighbors-based approach*

In this set of experiments, we define the topology of the network in a manner such that each node contains a fixed number of immediate neighbors (5 in our experiment). When updating the cluster centroids, each node only communicates with its immediate neighbors. We again report the experimental results of two kinds of data dispatching strategies: non-uniformly distributed data and uniformly distributed data.

6.2.1. *Non-uniformly distributed data*

The same non-uniformly sampled datasets were used as before, and the initial seed of centroids were the same. Tables 7 and 8 give the results. We observed that the performance are fairly similar with the results from random sampling-based approach, i.e., low communication cost, and high accuracy. However, since each node only communicates with its immediate neighbors, and the length of path is always 1, the communication cost in this setting is less. On the other hand, random sampling of nodes enables each node in each iteration to communicate with different nodes in the network, thus every node can

Table 7
Accuracy and communication cost of *P2P K-Means* clustering with fixed immediate neighbors

	#Nodes								
	50	45	40	35	30	25	20	15	10
Max #points/node	1523	1506	1935	2336	1957	2688	3035	5266	5625
#Messages	9882	8190	7902	6912	5580	5364	3600	2700	1800
Message rate (%)	13.62	11.28	11.01	9.75	7.78	7.64	5.18	4.15	2.80
#Mislabeled points	53	35	32	33	25	43	12	16	17
Error rate (%)	0.14	0.09	0.08	0.09	0.07	0.12	0.03	0.04	0.05

The original dataset (37,800 data points) is non-uniformly distributed over different-size P2P networks.

Table 8
Relative Euclidean Distance between each local centroid and the centralized centroid with fixed immediate neighbors

	#Nodes								
	50	45	40	35	30	25	20	15	10
RED for centroid 1	1.10	1.01	0.81	0.78	0.80	0.45	0.65	0.80	0.37
RED for centroid 2	0.86	0.72	0.79	0.80	0.65	0.56	0.53	0.62	0.35
RED for centroid 3	1.61	1.43	1.24	1.48	1.39	1.28	0.83	0.80	0.59
RED for centroid 4	0.74	0.63	0.74	0.64	0.67	0.64	0.63	0.48	0.32
RED for centroid 5	1.14	1.16	0.80	0.90	0.81	0.72	0.76	0.66	0.53
RED for centroid 6	12.34	10.66	8.13	8.08	15.66	15.97	7.06	7.29	3.66

The original dataset (37,800 data points) is non-uniformly distributed over different-size P2P networks.

get much more information than what it can get in the deterministic immediate neighbors-based approach, so the error rate of random sampling-based approach is a little lower.

6.2.2. *Uniformly distributed data*

When the data is uniformly distributed over all the nodes, communicating with randomly selected nodes or with immediate neighbors does not make much difference in terms of accuracy. So the performance of the algorithm in this setting is similar with the performance in random sampling-based approach on uniformly distributed data. Tables 9 and 10 give the detailed experimental results.

To summarize, *P2P K-Means* delivers clusters that are very comparable to the clustering done by centralized *K*-Means. The average number of mislabeled data points compared with the centralized approach is really small, i.e., less than 1% in almost all cases, and usually the number of messages exchanged is less than 20% of the communication necessary to move the data points into a central node.

Table 9
Accuracy and communication cost of *P2P K-Means* clustering with fixed immediate neighbors

	#Nodes								
	50	45	40	35	30	25	20	15	10
#Points/node	756	840	945	1080	1260	1512	1890	2520	3780
#Messages	9000	8100	7200	6300	5400	4500	3600	2700	1800
Message rate (%)	12.15	10.96	9.77	8.58	7.39	6.21	5.01	3.83	2.65
#Mislabeled points	13	6	21	13	35	6	11	1	5
Error rate (%)	0.34	0.02	0.06	0.03	0.09	0.02	0.03	0.00	0.01

The original dataset (37,800 data points) is uniformly distributed over different-size P2P networks.

Table 10
Relative Euclidean Distance between each local centroid and the centralized centroid

	#Nodes								
	50	45	40	35	30	25	20	15	10
RED for centroid 1	0.96	1.00	0.75	0.75	0.63	0.60	0.67	0.45	0.39
RED for centroid 2	0.89	0.85	0.65	0.70	0.70	0.43	0.49	0.36	0.36
RED for centroid 3	1.58	1.34	1.08	1.25	0.76	0.87	1.11	0.49	0.67
RED for centroid 4	0.74	0.74	0.72	0.55	0.48	0.45	0.37	0.40	0.30
RED for centroid 5	0.99	0.89	0.89	0.76	0.90	0.62	0.69	0.34	0.52
RED for centroid 6	10.16	7.10	6.95	8.16	7.50	6.25	5.96	3.86	3.75

The original dataset (37,800 data points) is uniformly distributed over different-size P2P networks.

7. Discussion and conclusions

This article describes the *P2P K-Means* algorithm for distributed clustering of data streams in a peer-to-peer sensor network environment. Sensor networks are characterized by low communication and computational capabilities, limited battery power, asynchronous nature and existence of faults. In the *P2P K-Means* algorithm, computation is performed locally, and communication of the local data models (represented by the corresponding centroids and the cluster counts) is restricted only within a limited neighborhood. As opposed to the full synchronization required in certain algorithms (e.g., [19] where the next iteration of *K*-Means begins only after information regarding the global centroids percolates to all the nodes), synchronization in *P2P K-Means* is restricted only within a neighborhood. Moreover, even if some node and/or link fails, the algorithm can continue, though its performance will degrade gracefully with an increase in the number of failures. Although the present version of the *P2P K-Means* is not designed to deal directly with continuous data streams in sensor networks, it can be easily extended to this scenario following the work in [4].

Experimental results demonstrate the effectiveness of the *P2P K-Means* clustering algorithm for the cases when the full data is uniformly and non-uniformly distributed over the nodes. It is found that the accuracy of the result, as compared to the centralized *K*-Means, is reasonably good even with a relatively small amount of message passing. An extensive theoretical analysis of the proposed algorithm is provided that gives bounds on the error in computing the centroids in the distributed clustering process compared to the centralized approach.

As a scope for future work, the variation of the performance of the algorithm under faulty conditions needs to be studied. The energy-quality trade-off characteristic of *P2P K-Means* (or, the relationship between the accuracy of the solution and the amount of communication) needs to be established both theoretically as well as experimentally. When communicating with distant neighbors the shortest path routing protocol is considered for the present, since choice of a good routing protocol was not the focus of this article. However, in the future, effect of other, more real-life routing protocols, should be investigated. Finally, the present analysis of the bounds on the error in computing the centroids at each iteration of the *P2P K-Means* vis-a-vis the centralized case are somewhat conservative. In the future, tighter bounds on this error needs to be developed. The authors are currently working in this direction.

Acknowledgements

The authors acknowledge supports from the United States National Science Foundation (NSF) CAREER award IIS-0093353, NSF Grant IIS-0329143, and NASA (NRA) NAS2-37143.

References

[1] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[2] W. Kowalczyk, M. Jelasity, A.E. Eiben, Towards data mining in large and fully distributed peer-to-peer overlay networks, in: T. Heskes, P. Lucas, L. Vuurpijl, W. Wiegerinck (Eds.), Proceedings of the Fifteenth Belgium Netherland Conference on Artificial Intelligence, University of Nijmegen, 2003, pp. 203–210.

[3] R. Wolff, A. Schuster, Association rule mining in peer-to-peer systems, IEEE Trans. Syst. Man Cyberns. Part B 34 (6) (2004).

[4] S. Guha, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams, in: Proceedings of the Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, USA, 2000.

[5] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading, 1974.

[6] J. Han, M. Kamber, in: Jim Gray (Ed.), Data Mining: Concepts and Techniques, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, 2000.

[7] P. Domingos, G. Hulten, A general method for scaling up machine learning algorithms and its application to clustering, in: Proceedings of the Eighteenth International Conference on Machine Learning, Morgan Kaufmann, MA, 2001, pp. 106–113.
[8] C. Aggarwal, J. Han, J. Wang, P. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th VLDB, 2003, pp. 81–92.
[9] B. Babcock, M. Datar, R. Motwani, L. O'Callaghan, Maintaining variance and k-medians over data stream windows, in: ACM SIGMOD Principles of Database Systems (PODS), 2003. Available from: <http://www.db.ucsd.edu/SIGMODPODS03/PODSpapers.htm>l.
[10] P. Domingos, G. Hulten, Mining high speed data streams, in: Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining, ACM Press, Boston, MA, 2001, pp. 71–80.
[11] P. Domingos, L. Spencer, G. Hulten, Mining time-changing data streams, in: Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, ACM Press, San Francisco, 2001, pp. 97–106.
[12] J. Lin, M. Vlachos, E. Keogh, D. Gunopulos, Iterative incremental clustering of time series, in: Proceedings of the IX Conference on Extending Database Technology (EDBT 2004), IEEE Computer Society, Crete, Greece, 2004.
[13] E. Keogh, J. Lin, W. Truppel, Clustering of time series subsequences is meaningless: implications for past and future research, in: Proceedings of the 3rd IEEE International Conference on Data Mining, 2003, pp. 115–122.
[14] I. Dhillon, D. Modha, A data-clustering algorithm on distributed memory multiprocessors, in: Proceedings of the KDD'99 Workshop on High Performance Knowledge Discovery, 1999, pp. 245–260.
[15] G. Forman, B. Zhang, Distributed data clustering can be efficient and exact, SIGKDD Explorations 2 (2) (2000) 34–38.
[16] H. Kargupta, W. Huang, K. Sivakumar, E. Johnson, Distributed clustering using collective principal component analysis, Knowledge Inform. Syst. J. 3 (2001) 422–448.
[17] M. Klusch, S. Lodi, G. Moro, Distributed clustering based on sampling local density estimates, in: Proceedings of the Joint International Conference on AI (IJCAI 2003), 2003.
[18] A. Hinneburg, D. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Proceedings of the 1998 International Conference on Knowledge Discovery and Data Mining (KDD), 1998, pp. 58–65.
[19] M. Eisenhardt, W. Muller, A. Henrich, Classifying documents by distributed P2P clustering, in: Proceedings of Informatik 2003, GI Lecture Notes in Informatics, Frankfort, Germany, 2003.
[20] S. Merugu, J. Ghosh, Privacy-preserving distributed clustering using generative models, in: Proceedings of the IEEE Conference on Data Mining (ICDM), 2003.
[21] E. Johnson, H. Kargupta, Collective, hierarchical clustering from distributed, heterogeneous data, in: M. Zaki, C. Ho (Eds.), Large-Scale Parallel KDD Systems, Lecture Notes in Computer Science, vol. 1759, Springer-Verlag, 1999, pp. 221–244.
[22] A. Lazarevic, D. Pokrajac, Z. Obradovic, Distributed clustering and local regression for knowledge discovery in multiple spatial databases, in: Proceedings of the 8th European Symposium on Artificial Neural Networks, 2000, pp. 129–134.
[23] N. Samatova, G. Ostrouchov, A. Geist, A. Melechko, RACHET: An efficient cover-based merging of clustering hierarchies from distributed datasets, Distrib. Parallel Databases 11 (2) (2002) 157–180.
[24] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, J. Mach. Learning Res. 3 (2002) 583–617.
[25] A. Fred, A. Jain, Data clustering using evidence accumulation, in: Proceedings of the International Conference on Pattern Recognition 2002, 2002, pp. 276–280.

[26] P. Jouve, N. Nicoloyannis, A new method for combining partitions, applications for distributed clustering, in: Proceedings of Workshop on Parallel and Distributed Computing for Machine Learning as part of the 14th European Conference on Machine Learning, 2003.
[27] E. Januzaj, H.-P. Kriegel, M. Pfeifle, DBDC: density based distributed clustering, in: Proceedings of EDBT in Lecture Notes in Computer Science, vol. 2992, 2004, pp. 88–105.
[28] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, Wireless sensor networks for habitat monitoring, in: ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), Atlanta, GA, 2002.
[29] D. Estrin, R. Govindan, J. Heidemann, Scalable coordination in sensor networks, Tech. Rep. Technical Report 99-692, University of Southern California, 1999.
[30] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: scalable coordination in sensor networks, in: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking Seattle, WA USA, 1999, pp. 263–270.
[31] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of the Hawaii Conference on System Sciences, January 2000.
[32] G.J. Pottie, W.J. Kaiser, Wireless integrated network sensors, Commun. ACM 43 (5) (2000) 551–558.
[33] H. Gharavi, S.P. Kumar, Proc. IEEE, Special Issue on Sensor Networks and Applications 91 (8) (2003).
[34] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Commun. Mag. 40 (2002) 102–114.
[35] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (2002) 393–422.
[36] <http://www.cse.ohio-state.edu/jain/refs/sns_refs.htm>.
[37] <http://www.research.rutgers.edu/mini/sensornetworks.html>.
[38] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D.E. Culler, K.S.J. Pister, System architecture directions for networked sensors, in: International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2000. Available from: <http://csg.lcs.mit.edu/rudolph/start.html>.
[39] F. Zhao, J. Liu, J. Liu, L. Guibas, J. Reich, Collaborative signal and information processing: an information-directed approach, Proc. IEEE 91 (8) (2003) 1199–1209.
[40] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, Boston, MA USA, 2000, pp. 56–67.
[41] K. Sohrabi, J. Gao, V. Ailawadhi, G.J. Pottie, Protocols for self-organization of a wireless sensor network, IEEE Personal Commun. 7 (5) (2000) 16–27.
[42] A. Cerpa, D. Estrin, ASCENT: adaptive self-configuring sensor networks topologies, in: Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), New York, NY, USA, 2002.
[43] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Trans. Wireless Commun. 1 (4) (2002) 660–670.
[44] S. Lindsey, C. Raghavendra, K.M. Sivalingam, Data gathering algorithms in sensor networks using energy metrics, IEEE Trans. Parallel Distrib. Syst., Special Issue on Mobile Computing 13 (9) (2002) 924–935.
[45] A. Sinha, A. Wang, A. Chandrakasan, Algorithmic transforms for efficient energy scalable computation, in: Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), 2000.

[46] P. Radivojac, U. Korad, K.M. Sivalingam, Z. Obradovic, Learning from class-imbalanced data in wireless sensor networks, in: 58th IEEE Semiannual Conference on Vehicular Technology Conference (VTC), Orlando, FL, October 2003, vol. 5, pp. 3030–3034.
[47] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority oversampling technique, J. Artificial Intell. Res. 16 (2002) 321–357.
[48] T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos, Distributed deviation detection in sensor networks, SIGMOD Record 32 (4) (2003) 77–82.
[49] B. Krishnamachari, S. Iyengar, Distributed Bayesian algorithms for fault tolerant event region detection in wireless sensor networks, IEEE Trans. Comput. 53 (3) (2004) 241–250.
[50] S. Ghiasi, A. Srivastava, X. Yang, M. Sarrafzadeh, Optimal energy aware clustering in sensor networks, Sensors 2 (2002) 258–269.
[51] O. Younis, S. Fahmy, Heed: a hybrid energy-efficient distributed clustering approach for ad-hoc sensor networks, IEEE Trans. Mobile Comput. 3 (4) (2004).
[52] W. Chen, J.C. Hou, L. Sha, Dynamic clustering for acoustic target tracking in wireless sensor networks, IEEE Trans. Mobile Comput. 3 (3) (2004) 258–271.
[53] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, M. Klein, VEDAS: a mobile distributed data stream mining system for real-time vehicle monitoring, in: Proceedings of the 2004 SIAM International Conference on Data Mining, 2004.
[54] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, CA, 1967, pp. 281–297.
[55] W. Cochran, Sampling Techniques, third ed., John Wiley & Sons Inc., New York, 1977.
[56] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Statist. Assoc. 58 (1963) 13–30.