IoT Network Segmentation When Sensors Fail

Bo-Wei Chen

***Abstract*—This draft presents a fault-tolerant network segmentation system for the Internet of Things (IoT). When devices of the IoT malfunction or fail, recovery needs to be performed to maintain system functionalities. In modern ad hoc networks like mobile ad hoc networks (MANETs), devices usually form dynamical clusters to collaboratively handle highly diverse sensing environments. To recover cluster information when parts of the IoT are not functioning, this study develops a centroid-free network segmentation algorithm that diverts dependency on centroids into empirical-space kernel matrices. The original problem of handling nonvectorial centroids is deduced to kernel matrix estimation.**

***Index Terms*—Incomplete data analysis, missing-value analysis, information recovery, fault-tolerant network segmentation, kernel method, iterative pursuit, clustering, partial kernel matrix estimation, Internet of Things, smart city**

I. Introduction

The Internet of Things (IoT) is a key to enabling dynamic urban monitoring in a city [1, 2]. When sufficient IoT devices are deployed in a city, city dynamics can be delineated, ranging from crowd activities to environmental statistics [3, 4]. With the recent advancement of telecommunication technology, the IoT is capable of being integrated with ad hoc networks, like mobile ad hoc networks (MANETs), vehicle ad hoc networks (VANETs), and flying ad hoc networks (FANETs). The IoT becomes fluid and mobile. Therefore, the IoT is no longer fixed at the same place. An IoT device can move around the city. The topology and the number of the IoT may change from place to place in respond to diversified environments. In highly populated areas, more devices can join the IoT, and the density of the IoT becomes higher, so that the system can cope with complex city dynamics.

In MANETs [5], IoT devices collaborate with each other to share computational/communication burdens, e.g., visual IoT devices. Under such circumstances, the entire IoT needs to be dynamically segmented to groups, or clusters [6]. At present, a great deal of research on IoT clustering has been done. Typical approaches like hierarchical clustering, *K*-means, self-organizing maps, and support vector clustering are widely used in the IoT. For example, Wang *et al*. [7] utilized support vector clustering for time-series data. Different kernels were employed in the work to examine the effectiveness. The merit of support vector learning was that it generated a satisfactory result even when few data were present. Besides, support vector clustering was rapid for dealing with data streams. The system by Tsirmpas *et al*. [8] was designed for profiling living environments, where the IoT was deployed. The authors devised self-organizing maps to cluster IoT sensed data. In some other studies [9, 10], for instance, Hajjar *et al*. [10] examined machine learning techniques and proposed a hybrid clustering algorithm based on hierarchical clustering and *K*-means. Their application was aimed at collaborative resource planning for communication networks.

As discussed above, IoT clusters are important logical formation in networks as they help data processing and communications. This is because IoT clusters are conducive to balancing computational loads or data redundancy in a collaborative mode. Devices in the same cluster present homogeneous characteristics, either in device parameters or data collection. Such a homogeneous characteristic is important when a fault-tolerant network is established. In a network, especially for the IoT, faults are generated due to device failure or malfunctioned nodes. When such a situation occurs, harvested data contain missing values. If samples contain missing values, they become nonvectorial data. Subsequently, typical mathematical operations are inapplicable. This may cause a problem to the IoT. To resolve missing values, several approaches have been developed, such as deletion, replacement, regression [11], *K*-nearest neighbors [12], multiple imputation [13], and matrix completion [14-16]. Among these methods, matrix completion provides more flexible choices for data imputation. Data imputation can generate approximate values for those missing-value entries before IoT cluster recovery is performed. The system by Fekade *et al*. [17] adopted the same concept. They used typical *K*-means and matrix completion as the model. Incomplete data were firstly handled and imputed by matrix completion, and subsequently *K*-means was used for clustering. In general, matrix completion is usually the first step for cluster recovery. However, considering the objective is to recover cluster membership, matrix completion requires additional computational time.

Rather than performing matrix completion prior to cluster recovery, Chi *et al*. [18] devised an iterative procedure, which contained two phases. One was *K*-means, and the other involved generating imputed values based on centroids. Chi *et al*. [18] filled in missing-value entries with zeros at the initial stage, so that *K*-means could be done in the first iteration. Afterwards, missing-value entries were filled in with centroids generated by *K*-means. Two steps were iterated over many cycles until the predefined criterion stabilized. Another

B.-W. Chen is with the Department of Electrical Engineering, National Sun Yat-Sen University, Taiwan

interesting work that did not rely on matrix completion before clustering was the system by Wagstaff [19]. The intuition behind it was that [19] split every incomplete sample (i.e., a feature vector) into two parts. One was a complete subvector, and the other was the rest of the feature vector that contained partial data. All of the incomplete subvectors were used for constraint generation. Clustering was performed based on *K*-means and complete subvectors with previously discovered constraints. Inspired by [18, 19], this work is aimed at reducing complexity of the above-mention clustering algorithms while maintaining accuracy at the same time. As the above-mention methods relied on cluster centers during iterations, the dependency on cluster centers is diverted to kernel matrices. Updates in centroids become computation of difference in kernel matrices. Therefore, such a mechanism enhances speeds while avoiding computing centroids in each iteration.

The rest of this draft is organized as follows. Section II introduces the proposed model. Subsequently, the proposed fault-tolerant network segmentation based on the model is detailed in Section III.

II. Proposed Methodology

This section describes how malfunctioned IoT devices can be divided into clusters without using typical data imputation. Firstly, subsection II.A introduces efficient kernelized centroid-free clustering. This is because finding centroids among missing-value data is not practical. Subsection II.A diverts the dependency on centroids to centroid-free kernel matrices. Subsequently, subsection II.B details how kernel matrices are estimated under the condition of missing values. Two versions of kernel matrix estimation are discussed. One requires third-party information, and the other needs no imputation.

A. Efficient Kernelized Centroid-Free Clustering for the Large-Scale IoT

This step follows the algorithm by [20] with modifications to support missing-value data clustering in our scenario. Assume that the dataset contains no missing values. Also assume that the total number of clusters is *S*, so that the entire IoT can be divided into clusters $\mathbb{C}_s$, where *s* denotes the cluster index, and *s* = 1,2,…,*S*. The clustering algorithm is aimed at minimizing the distance between samples and their corresponding centroids, $E_{\text{Clustering}}$. That is,

$$\begin{aligned} E_{\text{Clustering}} &= \sum_{s=1}^{S} \sum_{\mathbf{x}_t \in \mathbb{C}_s} \left\| \phi(\mathbf{x}_t) - \boldsymbol{\mu}_s \right\|^2 \\ &= \sum_{s=1}^{S} \sum_{\mathbf{x}_t \in \mathbb{C}_s} \left(\left\| \phi(\mathbf{x}_t) \right\|^2 + \left\| \boldsymbol{\mu}_s \right\|^2 - 2\Re_{ts} \right) \end{aligned} \tag{1}$$

where $\|\cdot\|$ stands for the $\mathfrak{L}_2$ norm, $\boldsymbol{\mu}_s$ represents the centroid of cluster *s*, and ϕ is a kernel function that maps an input $\mathbf{x}_t$ onto the intrinsic space (see the appendix). Input $\mathbf{x}_t$ is regarded as a complete sample without missing values herein. Let *N* denote the total number of IoT devices and N_s signify the number of IoT devices in cluster $\mathbb{C}_s$. Therefore, $N_1 + N_2 + \ldots + N_S = N$. The centroid of cluster $\mathbb{C}_s$ is defined as

$$\boldsymbol{\mu}_s = \frac{1}{N_s} \sum_{\mathbf{x}_i \in \mathbb{C}_s} \phi(\mathbf{x}_i). \tag{2}$$

The objective of minimization of (1) is actually equivalent to minimization of

$$E'_{\text{Clustering}} = \sum_{s=1}^{S} \sum_{\mathbf{x}_t \in \mathbb{C}_s} \left(\left\| \boldsymbol{\mu}_s \right\|^2 - 2\Re_{ts} \right) \tag{3}$$

as $\|\phi(\mathbf{x}_t)\|^2$ does not affect the clustering result. Therefore, the update of cluster centroids relies merely on $\|\boldsymbol{\mu}_s\|^2$ and $\Re_{ts}$. The following description focuses on $\|\boldsymbol{\mu}_s\|^2$ and $\Re_{ts}$.

In (3), $\Re_{ts}$ is given as follows by definition, and it can be interpreted as the similarity between sample $\phi(\mathbf{x}_t)$ and centroid $\boldsymbol{\mu}_s$.

$$\Re_{ts} = \phi(\mathbf{x}_t)^{\top} \boldsymbol{\mu}_s \tag{4}$$

where $\top$ is the transpose operator.

Plugging (2) into (4) yields

$$\begin{aligned} \Re_{ts} &= \phi(\mathbf{x}_t)^{\top} \left(\frac{1}{N_s} \sum_{\mathbf{x}_i \in \mathbb{C}_s} \phi(\mathbf{x}_i) \right) = \frac{1}{N_s} \sum_{\mathbf{x}_i \in \mathbb{C}_s} \phi(\mathbf{x}_t)^{\top} \phi(\mathbf{x}_i) \\ &= \frac{1}{N_s} \sum_{\mathbf{x}_i \in \mathbb{C}_s} \mathbf{K}_{ti} \end{aligned} \tag{5}$$

where $\mathbf{K}$ refers to a kernel matrix formed by all the input samples, and K_{ti} is a scaler and an element of $\mathbf{K}$ corresponding to samples t and i. For the squared norm of centroid $\boldsymbol{\mu}_s$, i.e., $\|\boldsymbol{\mu}_s\|^2$, it can also be represented by using $\mathbf{K}$ as follows.

$$\begin{aligned} \left\| \boldsymbol{\mu}_s \right\|^2 = \boldsymbol{\mu}_s^{\top} \boldsymbol{\mu}_s &= \frac{1}{N_s} \sum_{\mathbf{x}_t \in \mathbb{C}_s} \phi(\mathbf{x}_t)^{\top} \cdot \frac{1}{N_s} \sum_{\mathbf{x}_i \in \mathbb{C}_s} \phi(\mathbf{x}_i) \\ &= \frac{1}{N_s^2} \sum_{\mathbf{x}_t \in \mathbb{C}_s} \sum_{\mathbf{x}_i \in \mathbb{C}_s} \phi(\mathbf{x}_t)^{\top} \phi(\mathbf{x}_i) \\ &= \frac{1}{N_s^2} \sum_{\mathbf{x}_t \in \mathbb{C}_s} \sum_{\mathbf{x}_i \in \mathbb{C}_s} K_{ti}. \end{aligned} \tag{6}$$

The squared norm of centroid $\boldsymbol{\mu}_s$ stands for a submatrix based on all the samples in cluster $\mathbb{C}_s$.

With (5) and (6), the connection with $\|\boldsymbol{\mu}_s\|^2$ and $\Re_{ts}$ in (3) hinges on kernel matrix $\mathbf{K}$. In clustering, the entire process can be decomposed into iterative updates on all the samples. For each iteration, a sample needs to select the closet centroid and disjoins its original cluster. In brief, two phases are involved for

a sample. One is to join the closest cluster, and the other is to disjoin its original cluster.

Let q and p respectively represent the index of the closest cluster and the original cluster for sample $\mathbf{x}_\iota$. Then, the new centroid $\boldsymbol{\mu}'_q$ of the closest cluster becomes

$$\left\|\boldsymbol{\mu}'_q\right\|^2 = \frac{N_q^2}{\left(N_q+1\right)^2}\left\|\boldsymbol{\mu}_q\right\|^2 + \frac{2N_q}{\left(N_q+1\right)^2}\Re_{\iota q} + \frac{1}{\left(N_q+1\right)^2}K_{\iota\iota}, \quad (7)$$

whereas the new centroid $\boldsymbol{\mu}'_p$ of the original cluster is

$$\left\|\boldsymbol{\mu}'_p\right\|^2 = \frac{N_p^2}{\left(N_p-1\right)^2}\left\|\boldsymbol{\mu}_p\right\|^2 + \frac{2N_p}{\left(N_p-1\right)^2}\Re_{\iota p} + \frac{1}{\left(N_p-1\right)^2}K_{\iota\iota}. \quad (8)$$

Notably, the above $\|\boldsymbol{\mu}_q\|^2$ and $\|\boldsymbol{\mu}_p\|^2$ can be replaced with computation of $\mathbf{K}$ in (6).

As the centroids to $\mathbb{C}_q$ and $\mathbb{C}_p$ change, the distance between the members and these two centroids, i.e., $\Re$, should be accordingly modified. Thus,

$$\Re'_{jq} = \frac{1}{N'_q}\sum_{i\in\mathbb{C}'_q} K_{ji} = \frac{1}{N_q+1}\sum_{i\in\mathbb{C}'_q} K_{ji}, \quad j=1,\ldots,N. \quad (9)$$

$$\Re'_{jp} = \frac{1}{N'_p}\sum_{i\in\mathbb{C}'_p} K_{ji} = \frac{1}{N_p-1}\sum_{i\in\mathbb{C}'_p} K_{ji}, \quad j=1,\ldots,N. \quad (10)$$

The complexity of the above algorithm is $\mathcal{O}(N^2/S)$; however, for a large scale of the IoT, the update is still inefficient. An improved and rapid clustering algorithm is introduced. Besides, its complexity is merely $\mathcal{O}(N)$. To remove the dependency on N_q and N_p in (7)–(10), (3) is rewritten as

$$s^* = \arg\min_{s=1,\ldots,S}\left\{\frac{\left\|\tilde{\boldsymbol{\mu}}_s\right\|^2}{N_s^2} - \frac{2\tilde{\Re}_{\iota s}}{N_s}\right\}. \quad (11)$$

Let $\left\|\tilde{\boldsymbol{\mu}}'_q\right\|^2 = \left(N_q+1\right)^2\left\|\boldsymbol{\mu}'_q\right\|^2$ and $\left\|\tilde{\boldsymbol{\mu}}'_p\right\|^2 = \left(N_p+1\right)^2\left\|\boldsymbol{\mu}'_p\right\|^2$. Then, the new centroid of the closest and the original cluster when the membership of sample $\mathbf{x}_\iota$ changes is respectively

$$\left\|\tilde{\boldsymbol{\mu}}'_q\right\|^2 = \left\|\tilde{\boldsymbol{\mu}}_q\right\|^2 + 2\tilde{\Re}_{\iota q} + K_{\iota\iota} \quad (12)$$

and

$$\left\|\tilde{\boldsymbol{\mu}}'_p\right\|^2 = \left\|\tilde{\boldsymbol{\mu}}_p\right\|^2 - 2\tilde{\Re}_{\iota p} + K_{\iota\iota}. \quad (13)$$

It becomes more efficient when one rewrites the above equations into the following form, where the amount of changes is introduced. Therefore,

$$\Delta\left\|\tilde{\boldsymbol{\mu}}_q\right\|^2 = \left\|\tilde{\boldsymbol{\mu}}'_q\right\|^2 - \left\|\tilde{\boldsymbol{\mu}}_q\right\|^2 = 2\tilde{\Re}_{\iota q} + K_{\iota\iota} \quad (14)$$

and

$$\Delta\left\|\tilde{\boldsymbol{\mu}}_p\right\|^2 = \left\|\tilde{\boldsymbol{\mu}}'_p\right\|^2 - \left\|\tilde{\boldsymbol{\mu}}_p\right\|^2 = -2\tilde{\Re}_{\iota p} + K_{\iota\iota}. \quad (15)$$

Likewise, let $\tilde{\Re}'_{jq} = \left(N_q+1\right)\Re'_{jq}$ and $\tilde{\Re}'_{jp} = \left(N_p-1\right)\Re'_{jp}$. Subsequently,

$$\Delta\tilde{\Re}_{jq} = \tilde{\Re}'_{jq} - \tilde{\Re}_{jq} = K_{j\iota}, \quad j=1,\ldots,N \quad (16)$$

and

$$\Delta\tilde{\Re}_{jp} = -K_{j\iota}, \quad j=1,\ldots,N. \quad (17)$$

B. Kernel Matrix Estimation

As mentioned earlier at the beginning of this section, the objective of subsection II.A is to shift the dependency on centroids to kernel matrices. This subsection subsequently details estimation of kernel matrices under a condition when IoT devices malfunction and incomplete data are generated. This work employs the concept by [21] with modifications in the Masked Partial Three-Side functions to support fault-tolerant network segmentation (see the next section).

Let $\mathbf{B}$ represent a mask that performs dimension selection. Then,

$$\mathbf{B}_\mathbf{x}(m) = \begin{cases} 1 & \text{if } x_m \text{ is given} \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

and

$$\tilde{\mathbf{x}} = \mathbf{x}\otimes\mathbf{B}_\mathbf{x} \quad (19)$$

where x_m denotes the m–th dimension of $\mathbf{x}$, and $\otimes$ is the element-wise operator, i.e., Hadamard operators.

Typically, a kernel matrix measures the similarity between two vectors, i.e., two samples. However, when the system, e.g., [22], calculates such a matrix, no third-party information is used. Take the cosine-similarity function for example. The kernel matrix is formed by calculating

$$K_{\text{Cosine}}\left(\mathbf{x}_i,\mathbf{x}_j\right) = \frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j}{\left\|\mathbf{x}_i\right\|\left\|\mathbf{x}_j\right\|} \quad (20)$$

where i and j respectively signify the indices of two instances. When these two vectors contain missing entries, it involves nonvectorial similarities and results in biased estimation.

Let

$$\begin{cases} \tilde{\mathbf{x}}_i = \mathbf{x}_i \otimes \mathbf{B}_{\mathbf{x}_i} \otimes \mathbf{B}_{\mathbf{x}_i} \\ \tilde{\mathbf{x}}_j = \mathbf{x}_j \otimes \mathbf{B}_{\mathbf{x}_i} \otimes \mathbf{B}_{\mathbf{x}_i} \end{cases}, \tag{21}$$

i.e., double masks. Thus, the Masked Partial-Cosine (MPC) function is used for nonvectorial similarities.

$$K_{\text{MPC}}\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right) = \frac{\tilde{\mathbf{x}}_i^\top \tilde{\mathbf{x}}_j}{\left\|\tilde{\mathbf{x}}_i\right\| \left\|\tilde{\mathbf{x}}_j\right\|} \tag{22}$$

It is worth noticing that no data approximation is performed in (19) and (21) during masking. Based on (22), the Masked Partial Three-Side (MPT) cosine function is derived as follows.

$$K_{\text{MPT}}\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right) = \frac{\left(\tilde{\mathbf{x}}_i \oplus \boldsymbol{\mu}_u\right)^\top \left(\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v\right)}{\left\|\tilde{\mathbf{x}}_i \oplus \boldsymbol{\mu}_u\right\| \left\|\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v\right\|} \tag{23}$$

where $\boldsymbol{\mu}_u$ and $\boldsymbol{\mu}_v$ are respectively centroids of clusters $\mathbb{C}_u$ and $\mathbb{C}_v$. Besides, $\tilde{\mathbf{x}}_i \in \mathbb{C}_u$ and $\tilde{\mathbf{x}}_j \in \mathbb{C}_v$. This creates an approximate value with a cluster-dependent average in the same attribute. The effect of "$\tilde{\mathbf{x}}_i \oplus \boldsymbol{\mu}_u$" and "$\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v$" indicates that the similarity between $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$ should also consider the similarity between $\tilde{\mathbf{x}}_i$ and the cluster centroid of $\tilde{\mathbf{x}}_j$ Second, the missing-value entries are filled with centroid information.

Equation (23) can be extended into MPT cosine polynomial kernels, MPT radial basis functions (RBFs), and MPT TRBFs, respectively, i.e.,

$$\begin{cases} K_{\text{MPC}}\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right) = \left(1 + \frac{1}{\sigma^2} \frac{\tilde{\mathbf{x}}_i^\top \tilde{\mathbf{x}}_j}{\left\|\tilde{\mathbf{x}}_i\right\| \left\|\tilde{\mathbf{x}}_j\right\|}\right)^D \\ K_{\text{MPT}}\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right) = \left(1 + \frac{1}{\sigma^2} \frac{\left(\tilde{\mathbf{x}}_i \oplus \boldsymbol{\mu}_u\right)^\top \left(\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v\right)}{\left\|\tilde{\mathbf{x}}_i \oplus \boldsymbol{\mu}_u\right\| \left\|\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v\right\|}\right)^D \end{cases}, \tag{24}$$

$$\begin{cases} K_{\text{MPC}}\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right) = \exp\left(-\frac{1}{2\sigma^2} \left\| \frac{\tilde{\mathbf{x}}_i}{\left\|\tilde{\mathbf{x}}_i\right\|} - \frac{\tilde{\mathbf{x}}_j}{\left\|\tilde{\mathbf{x}}_j\right\|} \right\|^2\right) \\ K_{\text{MPT}}\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right) = \exp\left(-\frac{1}{2\sigma^2} \left\| \frac{\tilde{\mathbf{x}}_i \oplus \boldsymbol{\mu}_u}{\left\|\tilde{\mathbf{x}}_i \oplus \boldsymbol{\mu}_u\right\|} - \frac{\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v}{\left\|\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v\right\|} \right\|^2\right) \end{cases}, \tag{25}$$

and

$$\begin{cases} K_{\text{MPC}}\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right) = \frac{-1}{\sigma^2} + \sum_{\tau=1}^{D} \frac{1}{(\tau!)\sigma^{2\tau}} \left(\frac{\tilde{\mathbf{x}}_i^\top}{\left\|\tilde{\mathbf{x}}_i\right\|} \frac{\tilde{\mathbf{x}}_j}{\left\|\tilde{\mathbf{x}}_j\right\|} \right)^\tau \\ K_{\text{MPT}}\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right) = \frac{-1}{\sigma^2} + \sum_{\tau=1}^{D} \frac{1}{(\tau!)\sigma^{2\tau}} \left(\frac{\tilde{\mathbf{x}}_i^\top \oplus \boldsymbol{\mu}_u^\top}{\left\|\tilde{\mathbf{x}}_i \oplus \boldsymbol{\mu}_u\right\|} \frac{\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v}{\left\|\tilde{\mathbf{x}}_j \oplus \boldsymbol{\mu}_v\right\|} \right)^\tau \end{cases} \tag{26}$$

where σ^2 is the variance, D signifies the kernel order, and τ denotes the index of components.

The upper parts of (24)–(26) rely on no centroids, whereas the lowers parts require centroids for imputation. These two versions affect the result of fault-tolerant network segmentation, detailed in the following section.

III. Proposed Fault-tolerant Network Segmentation

With kernel matrix estimation and kernelized centroid-free clustering in the previously mentioned sections, fault-tolerant network segmentation is applicable.

Two versions of fault-tolerant network segmentation are discussed below. Their difference is computation of kernel matrices. Kernelized centroid-free clustering is the same. The first one repeatedly fills in missing-value entries with centroids. The second one directly ignores missing-value entries.

A. Iterative Pursuit

The iterative pursuit algorithm consists of two steps. These steps are iterated until clusters are stabilized by checking (6). The first step is kernel matrix estimation with centroid-based imputation. The second step uses clustering to update centroids, so that changes can be reflected in kernel matrix estimation.

Algorithm I: MPT-Based Iterative Pursuit

Input: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ with missing values

Output: $\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_S$

```
Randomly initialize μ_s ∈ ℝ^M, where s = 1,2,...,S
Choose an MPT cosine kernel
for limited iterations
    Estimate the kernel matrix, K
        while x_n migrates between clusters (n= 1,2,...,N)
            Compute Δ of the squared norm based on (14) and (15)
            Compute ΔR based on (16) and (17)
        end
    Compute μ_s
end
```

Let $\mathbb{I}$ represent the number of iterations and $\mathcal{O}(M^2N^3)$ denote the complexity of kernel matrix estimation. Then, the complexity of this algorithm is $\mathcal{O}(M^2N^3\mathbb{I})$.

B. Rapid Computation

Unlike the above algorithm, this method does not require imputation. Besides, no iteration is performed for kernel matrix

estimation. This algorithm saves more computational time than the above algorithm.

Algorithm II: MPC-Based Rapid Computation

Input: $\mathbf{X} = [\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N]$ with missing values

Output: $\mathbb{C}_1,\mathbb{C}_2,\ldots,\mathbb{C}_S$

1. Randomly initialize $\boldsymbol{\mu}_s \in \mathbb{R}^M$, where $s = 1,2,...,S$
2. Choose an MPC kernel
3. Estimate the kernel matrix, $\mathbf{K}$
4. **while** $\mathbf{x}_n$ migrates between clusters ($n= 1,2,...,N$)
5. Compute Δ of the squared norm based on (14) and (15)
6. Compute $\Delta\mathfrak{R}$ based on (16) and (17)
7. **end**

As no iteration is used, this algorithm has complexity of $\mathcal{O}(M^2N^3)$.

APPENDIX

The appendix shows the drawback of centroid-based clustering and intrinsic kernel space. Consider the case of complete data $\mathbf{X}$. Given an M-by-N matrix $\mathbf{X}$ without missing values, where M specifies the number of dimensions, and N denotes the number of observed samples. In our case, observed samples refer to IoT devices. Besides, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_N]$. The objective of matrix completion is to minimize the difference between $\mathbf{X}$ and its approximate matrix formed by $\mathbf{UV}$. That is, $\mathbf{X} \approx \mathbf{UV}$. One of the implementation for matrix completion, i.e., ridge alternating least squares [14], is shown as follows

$$\min E_{\mathrm{rALS}}(\mathbf{U},\mathbf{V}) = \min\left\{\left\|\mathbf{X}-\mathbf{UV}\right\|_{\mathcal{F}}^2 + \rho_{\mathrm{U}}\left\|\mathbf{U}\right\|_{\mathcal{F}}^2 + \rho_{\mathrm{V}}\left\|\mathbf{V}\right\|_{\mathcal{F}}^2\right\} \tag{27}$$

where $\mathbf{U}$ and $\mathbf{V}$ are respectively M-by-H and H-by-N unknown matrices, H is the intermediate dimension, and $\|\cdot\|_{\mathcal{F}}$ represents the Frobenius norm. Ridge ALS uses ridge parameters, ρ_{U} and ρ_{V}, to regularize and prevent $\mathbf{U}$ and $\mathbf{V}$ from overfitting, respectively. Differentiating E_{rALS} and zeroing the equations yield

$$\mathbf{V} = \left(\mathbf{U}^\top\mathbf{U} + \rho_{\mathrm{V}}\mathbf{I}\right)^{-1}\mathbf{U}^\top\mathbf{X} \tag{28}$$

and

$$\mathbf{U}^\top = \left(\mathbf{V}\mathbf{V}^\top + \rho_{\mathrm{U}}\mathbf{I}\right)^{-1}\mathbf{V}\mathbf{X}^\top \tag{29}$$

where $\mathbf{I}$ is an identity matrix. The system can iteratively update $\mathbf{U}$ and $\mathbf{V}$ based on (28) and (29) to generate a solution.

For matrix $\mathbf{X}$ with missing values, the procedure is still the same, as shown in (30) and (31). The difference is that an element-wise mask G is imposed on $\mathbf{X}$. If an element of $\mathbf{X}$ is missing, then such an entry is temporally substituted with a zero first.

$$\mathbf{V} = \left(\mathbf{U}^\top\mathbf{U} + \rho_{\mathrm{V}}\mathbf{I}\right)^{-1}\mathbf{U}^\top \times \mathrm{G}(\mathbf{X}) \tag{30}$$

and

$$\mathbf{U}^\top = \left(\mathbf{V}\mathbf{V}^\top + \rho_{\mathrm{U}}\mathbf{I}\right)^{-1}\mathbf{V} \times \mathrm{G}(\mathbf{X})^\top. \tag{31}$$

Finally, the missing elements of $\mathbf{X}$ are replaced with the corresponding elements of the generated matrix, i.e., $\mathbf{UV}$. This completes the matrix approximation.

For centroid-based clustering, it requires mapping a sample from feature space into intrinsic space. Assume the size of the dimension in feature space is M, and then that of intrinsic space is

$$J = \binom{(M+1)+D-1}{D} = \frac{(M+D)!}{M!D!}. \tag{32}$$

Let d_m represent the power of x_m, where $m = 1,2,\ldots,M+1$. After kernel mapping, the new variable in each dimension follows the form

$$\sqrt{\binom{D!}{d_1!,\ldots,d_{M+1}!}}\left(x_l\right)^{d_1}\ldots\left(x_M\right)^{d_M}\left(x_{M+1}\right)^{d_{M+1}} \tag{33}$$

$$\text{s.t.}\begin{cases}0 \le d_1,\ldots,d_{M+1} \le D\\ d_1+\ldots+d_{M+1} = D\end{cases}$$

if polynomial kernels are used. When truncated RBFs (TRBFs) are employed,

$$\exp\left\{-\frac{\left\|\mathbf{x}\right\|^2}{2\sigma^2}\right\}\left(x_1\right)^{d_1}\ldots\left(x_M\right)^{d_M}\left(x_{M+1}\right)^{d_{M+1}} \tag{34}$$

$$\text{s.t.}\begin{cases}0 \le d_1,\ldots,d_{M+1} \le D\\ d_1+\ldots+d_{M+1} = D\end{cases}.$$

In intrinsic space, an M-by-1 vector is converted to a J-by-1 one. There are several drawbacks if centroid-based clustering and intrinsic space are used. Firstly, typical RBFs are inapplicable due to infinite dimensions in intrinsic space. This is inconvenient as centroids cannot be represented in intrinsic space when RBFs are used. Instead, TRBFs should be adopted. Second, kernel mapping requires the missing values in feature space to be imputed beforehand.

REFERENCES

[1] S. Dama, V. Sathya, K. Kuchi, and T. V. Pasca, "A feasible cellular Internet of Things: Enabling edge computing and the IoT in dense futuristic cellular networks," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 66–72, Dec. 2017.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[3] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, Mar. 2016.

[4] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[5] U. Aguilera and D. López-de-Ipiña, "An architecture for automatic service composition in MANET using a distributed service graph," *Future Generation Computer Systems*, vol. 34, pp. 176–189, May 2014.

[6] J. DeFranco, M. Kassab, and J. Voas, "How do you create an internet of things workforce?," *IEEE IT Professional*, vol. 20, no. 4, pp. 8–12, Jul.–Aug. 2018.

[7] C.-D. Wang, J.-H. Lai, D. Huang, and W.-S. Zheng, "SVStream: A support vector-based algorithm for clustering data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1410–1424, Jun. 2013.

[8] C. Tsirmpas, A. Anastasiou, P. Bountris, and D. Koutsouris, "A new method for profile generation in an Internet of Things environment: An application in ambient-assisted living," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 471–478, Dec. 2015.

[9] K. M. Thilina, K. W. Choi, N. Saquib, and E. Hossain, "Machine learning techniques for cooperative spectrum sensing in cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 11, pp. 2209–2221, Nov. 2013.

[10] M. Hajjar, G. Aldabbagh, N. Dimitriou, and M. Z. Win, "Hybrid clustering scheme for relaying in multi-cell LTE high user density networks," *IEEE Access*, vol. 5, pp. 4431–4438, Mar. 2017.

[11] Y. C. Yuan, "Multiple imputation for missing data: Concepts and new development," SAS Institute Incorporation, Rockville, MD, Technical, 2000.

[12] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[13] D. B. Rubin, *Multiple Imputation for Nonresponse in Surveys*. New York, NY: Wiley, 1987.

[14] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the Netflix prize," in *Proc. 4th International Conference on Algorithmic Applications in Management*, Shanghai, China, 2008, Jun. 23–25, pp. 337–348.

[15] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proc. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007)*, San Jose, California, United States, 2007, Aug. 12–15, pp. 39–42.

[16] M. Eirinaki, J. Gao, I. Varlamis, and K. Tserpes, "Recommender systems for large-scale social networks: A review of challenges and solutions," *Future Generation Computer Systems*, vol. 78, pp. 413–418, Jan. 2018.

[17] B. Fekade, T. Maksymyuk, M. Kyryk, and M. Jo, "Probabilistic recovery of incomplete sensed data in IoT," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2282–2292, Aug. 2018.

[18] J. T. Chi, E. C. Chi, and R. G. Baraniuk, "k-POD: A method for k-means clustering of missing data," *The American Statistician*, vol. 70, no. 1, pp. 91–99, Jan. 2016.

[19] K. Wagstaff, "Clustering with missing values: No imputation required," in *Proc. Meeting of the International Federation of Classification Societies*, Chicago, Illinois, United States, 2004, Jul. 15–18, pp. 649–658.

[20] S.-Y. Kung, *Kernel Methods and Machine Learning*. Cambridge, UK: Cambridge University Press, Jun. 2014.

[21] B.-W. Chen, S. Rho, L. T. Yang, and Y. Gu, "Privacy-preserved big data analysis based on asymmetric imputation kernels and multiside similarities," *Future Generation Computer Systems*, vol. 78, no. 2, pp. 859–866, Jan. 2018.

[22] S.-Y. Kung and P.-Y. Wu, "Kernel approach to incomplete data analysis (KAIDA)," in *Proc. 1st International Conference on Advances in Big Data Analytics*, Las Vegas, NV, 2015, Jul. 21–24, pp. 95–101.