Distributed Spectral Clustering based on Euclidean Distance Matrix Completion

Simone Scardapane, Rosa Altilio, Massimo Panella and Aurelio Uncini

***Abstract*—In this paper, we consider the problem of distributed spectral clustering, wherein the data to be clustered is (horizontally) partitioned over a set of interconnected agents with limited connectivity. In order to solve it, we consider the equivalent problem of reconstructing the Euclidean distance matrix of pairwise distances among the joint set of datapoints. This is obtained in a fully decentralized fashion, making use of an innovative distributed gradient-based procedure, where at every agent we interleave gradient steps on a low-rank factorization of the distance matrix, with local averaging steps considering all its neighbors' current estimates. The procedure can be applied to any spectral clustering algorithm, including normalized and unnormalized variations, for multiple choices of the underlying Laplacian matrix. Experimental evaluations demonstrate that the solution is competitive with a fully centralized solver, where data is collected beforehand on a (virtual) coordinating agent.**

I. Introduction

Performing inference on data which is partitioned over geographically distinct locations is now considered as a fundamental problem in many scientific endeavors [1], [2], including peer-to-peer networks [3], sensor networks [4], [5], power grids [6], and many others [7]. Separate domains of application may impose largely different constraints on the solution, including (but not limited to) low computational power at every location, limited underlying connectivity (e.g. no broadcasting capability), impossibility of having a single coordinating agent, privacy concerns on the exchange of sensible information [8], and so on.

Over the last years, different authors have proposed distributed variants for many of the standard supervised and unsupervised algorithms available in the machine learning community. Examples of the former include distributed algorithms for training support vector machines [3], [9], random-weights networks [10], [11], multilayer perceptrons [12], and others. Examples of the latter, instead, comprise distributed versions of the k-means procedure [13], principal component analysis [14], generative models for clustering [15], etc. The availability of distributed clustering procedures, in particular, has a wide range of possible applications, ranging from collaborative document clustering [16], sensors aggregation in *ad-hoc* networks [17] and grouping of medical patients across diverse clinical databases. A schematic example of a distributed clustering setting, comprising two separate clusters partitioned over three distinct *agents*, is shown in Fig. 1.

Authors are with the Department of Information Engineering, Electronics and Telecommunications (DIET), "Sapienza" University of Rome, Via Eudossiana 18, 00184 Rome, Italy. Emails: {simone.scardapane, rosa.altilio, massimo.panella, aurelio.uncini}@uniroma1.it.

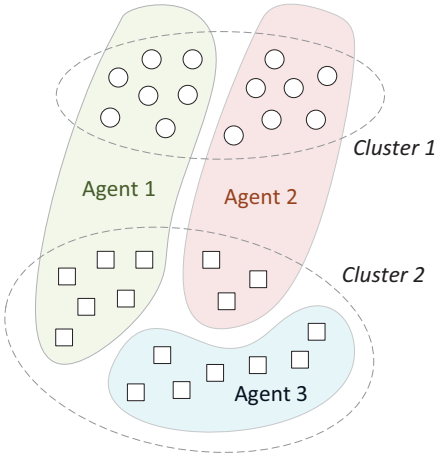


Fig. 1. Example of distributed clustering over a network, involving two clusters (denoted by circles and squares, respectively), and three agents. Every agent has a varying number of points, and it may not have representatives from each of the clusters.

To date, however, many fundamental clustering algorithms cannot be applied straightforwardly in a distributed context, due to a lack of available fully decentralized training protocols. Particularly, in this paper we are interested in performing spectral clustering (SC) [18], [19], whenever data is partitioned over a network as in Fig. 1. SC is a popular class of clustering algorithms, which has been shown to outperform alternative mechanisms in many real-world applications. Basically, it works by performing clustering in a suitably transformed space, whose mapping is constructed starting from a similarity graph corresponding to the data. Different ways of constructing the graph (and extracting its eigenstructure) give rise to alternative SC formulations, which have been applied (among others) to speech recognition [20], 3D segmentation [21], and so on. Since SC depends on the pairwise distances among *all* points, to the best of our knowledge no distributed protocol for its solution has been proposed, and most of the research has focused in parallelizing its computation with a single coordinating agent [22], [23].

In order to solve this problem, in this paper we consider the equivalent task of reconstructing (in a decentralized fashion) the matrix of Euclidean distances (EDM) among all points. Indeed, perfect knowledge of this matrix would allow each agent to solve independently the original SC problem, for a wide range of different choices of the underlying data graph [19]. Recasting the problem in this way, however, allows

978-1-5090-0620-5/16/$31.00

us to leverage over a large number of works on matrix completion [24], [25] and EDM completion [26], especially in the distributed setting [27], [28]. Particularly, we consider a distributed gradient-descent algorithm to this end, originally proposed for semi-supervised learning over networks [28]. This is an iterative procedure, where at each step every agent performs a single gradient descent step on its own estimate, followed by an averaging step with respect to its neighbors' estimates. This kind of techniques have a long history in the optimization field [29], and they have recently gained a wide popularity under the name of 'diffusion' strategies [2], [30]. Our proposed algorithm reduces the computational complexity by exploiting the specific structure of the EDM matrices, which allows it to operate on a suitable factorization of the original matrix in order to reduce the number of parameters to be estimated.

In the initial phase of the algorithm, we allow a small exchange of data patterns between agents. This is required since, in the beginning, each agent has only access to a limited block of the full EDM, making the problem of its estimation rather hard. However, our experimental results show that the process becomes feasible with only a limited exchange of patterns, which can be customized in a large variety of ways. If privacy is required, sensible information can also be protected by the inclusion of well-known privacy preservation strategies for distributed computation of Euclidean distances [31]. As we said, the proposed algorithm is the first truly decentralized procedure for performing SC, which respects the constraints put forward before, particularly: (i) it only considers local communication among neighbors; (ii) update steps can be performed easily even on low-power devices; and (iii) no coordinating entity is required, such that every node has the same importance in the overall network. Our experimental results show that the obtained solution is comparable with the optimal scenario in which data is collected beforehand on a centralized processor, to solve the global SC problem.

The rest of the paper is organized as follows. In Section II we provide a basic introduction to SC techniques. Next, in Section III we formulate the distributed SC problem, and detail our proposed algorithm for its solution. Section IV presents our experimental results, while Section V concludes the paper with some final remarks and possible future lines of research.

II. Spectral Clustering

In this section we describe the basic concepts pertaining to the SC framework. We refer to any introductory publication for a fuller treatment, e.g. [19], [32]. Suppose we are given a set of N points $S = \{\mathbf{x}_i\}_{i=1}^N$, where each $\mathbf{x}_i \in \mathbb{R}^d$. Our aim is to partition the set S into k disjoint clusters, where k is given beforehand, such that a predefined measure of quality is optimized (e.g. the Davies-Bouldin index [33]).

The first step of any SC algorithm is to construct a *data-adjacency* graph $\mathcal{G}^D = \left(\mathcal{V}^D, \mathcal{E}^D\right)$,[1] where each vertex in $\mathcal{V}^D$ corresponds to a point in S, and each edge in $\mathcal{E}^D$ weights the similarity among two points. We define an adjacency matrix $\mathbf{W}^D \in \mathbb{R}^{N \times N}$, where w_{ij}^D is the similarity among points $\mathbf{x}_i$ and $\mathbf{x}_j$. There are many ways of constructing $\mathbf{W}^D$, corresponding to different variants of the framework. As an example, a simple choice is a fully connected graph with a Gaussian weighting with bandwidth $\gamma > 0$:

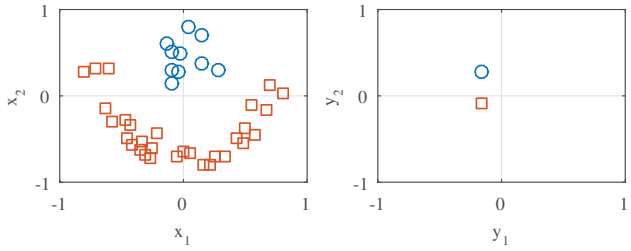
$$w_{ij}^D = \exp\left\{-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2\right\} . \tag{1}$$

Although there are multiple choices for $\mathbf{W}^D$, we stress here that practically every choice depends on the Euclidean distance between points, an aspect which is fundamental in our design of a distributed strategy in the next section. Once the weighting matrix $\mathbf{W}^D$ is specified, we can compute the so-called Laplacian adjacency matrix, which is defined as:

$$\mathbf{L}^D = \mathbf{D}^D - \mathbf{W}^D , \tag{2}$$

where the degree matrix $\mathbf{D}^D$ is a diagonal matrix with $\mathbf{D}_{ii}^D = \sum_{j=1}^N w_{ij}^D$. Similarly to what happens with $\mathbf{W}^D$, there are many variants for constructing $\mathbf{L}^D$, such as normalizing it by the degree matrix, or using an iterated version [19]. The third step of the SC procedure is to extract the leading k eigenvectors (where k is the number of desired clusters) from $\mathbf{L}^D$, an operation which can be implemented efficiently, particularly if the elements of $\mathbf{W}^D$ are symmetric [34].

Suppose that the eigenvectors are stacked column-wise in a matrix $\mathbf{U}$. The ith row of $\mathbf{U}$ can be interpreted as the transformation of $\mathbf{x}_i$ in the space induced by the eigenstructure of $\mathbf{L}^D$. Based on this consideration, the last step of SC algorithms is to cluster in k groups the rows of $\mathbf{L}^D$, typically by using the standard k-means procedure. The rationale for this is that clustering in this new space is generally simpler, since it exploits *a priori* the similarity information contained in the adjacency graph (similar ideas can be found in other areas of machine learning, such as semi-supervised learning with manifold regularization [35]). As an extreme case, consider the toy problem presented in Fig. 2. The original 2-dimensional points in Fig. 2a are organized in two optimal clusters, one of which is non-trivial to capture with a standard k-means, which assumes hypersphericity of the clusters. However, in the new



(a) Original data (b) After transformation

Fig. 2. An example of eigen-transformation on a 2D dataset. (a) 40 points from the original data. (b) First 2 eigenvectors of Laplacian matrix computed from a similarity graph with Euclidean distance on edges. All patterns belonging to a cluster are mapped to the same point.

[1] We use a superscript D to differentiate it from the agents' graph introduced in the next section.

domain induced by $\mathbf{U}$ (see Fig. 2b), all the patterns belonging to a cluster are mapped to a single point.

III. Distributed Spectral Clustering over Networks

A. Formulation of the problem

For the rest of this paper, we assume that the dataset S to be clustered is not available on a centralized location. Instead, it is partitioned over L agents, such that the kth agent has access to a dataset S_k and $\bigcup_{k=1}^{L} S_k = S$ (see Fig. 1). The connectivity among agents can be represented as a second graph, that we call the *agents*' graph, $\mathcal{G}^A = \left(\mathcal{V}^A, \mathcal{E}^A\right)$, such that each vertex $v \in \mathcal{V}^A$ is an agent, and two agents i and j can communicate (i.e. exchange information) between them only if the edge (i, j) is in the set $\mathcal{E}^A$. For simplicity, we assume that the agents' graph is time-invariant, connected (i.e., two agents are always reachable by traversing a finite amount of edges), and undirected, such that if $(i, j) \in \mathcal{E}^A$, then $(j, i) \in \mathcal{E}^A$. Additionally, we suppose that the agents have available a mechanism of synchronization for performing iterative computations. We note that this is a standard assumption in most of the literature on distributed learning [4], [11], [36].

From what we said in the previous section, it is clear that the distributed SC problem is equivalent to the distributed computation of the matrix $\mathbf{L}^D$, which in turn is equivalent to the computation of the EDM $\mathbf{E}$, where $E_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$. In the following section we propose an efficient distributed gradient algorithm to this end.

B. Distributed computation of the EDM

To begin with, we note that with a proper rearrangement of patterns, the global EDM $\mathbf{E}$ can always be expressed as:

$$\mathbf{E} = \begin{bmatrix} \mathbf{E}_1 & ? & ? \\ ? & \ddots & ? \\ ? & ? & \mathbf{E}_L \end{bmatrix}, \tag{3}$$

where $\mathbf{E}_k$ denotes the EDM computed only from the patterns in S_k. This particular structure implies that the sampling set is not random, and makes non-trivial the problem of completing $\mathbf{E}$ solely from the knowledge of the local matrices. At the opposite, the idea of exchanging the entire local datasets between nodes is unfeasible because of the amount of data which would need to be shared. Based on these considerations, we propose a framework for the distributed estimation of $\mathbf{E}_k$, which consists of four steps:

1) **Patterns exchange**: every agent exchanges a fraction p of the available S_k with its neighbors. This step is necessary so that the agents can increase the number of known entries in their local matrices. In order to maximize the diffusion of the data within the network, this step is iterated $n_{\max}^{(1)}$ times; at every iteration an increasing percentage of shared data is constituted by pattern received by the neighbors in previous iterations. A simple strategy to do this consists, at the iteration n, to choose $\frac{n_{\max}^{(1)}-n+1}{n_{\max}^{(1)}} p$ patterns from the local dataset, and $\frac{n-1}{n_{\max}^{(1)}} p$ patterns received in the previous $n-1$ iterations. In order to preserve privacy, this step can include one of the privacy-preserving strategies known in the literature [31].
2) **Local EDM computation**: each agent computes, using its original dataset and the data received from its neighbors, an incomplete approximation $\hat{\mathbf{E}}_k \in \mathbb{R}^{N \times N}$ of the real EDM matrix $\mathbf{E}$.
3) **Entries exchange**: the agents exchange a sample of their local EDMs $\hat{\mathbf{E}}_k$ with their neighbors. Again, this step is iterated $n_{\max}^{(2)}$ times using the rule of step 1. In this case, we denote as $n_{\max}^{(2)}$ the maximum number of iterations.
4) **Distributed EDM completion**: the agents complete the estimate $\tilde{\mathbf{E}}$ of the global EDM using the strategy detailed next.

To formalize this last step, define a local matrix $\mathbf{\Omega}_k$ as:

$$\mathbf{\Omega}_k = \begin{cases} 1 & \text{if } \hat{E}_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

We aim at finding a matrix $\tilde{\mathbf{E}}$ such that the following (joint) cost function is minimized:

$$\min_{\tilde{\mathbf{E}} \in \mathrm{EDM}(N)} \sum_{k=1}^{L} J_k(\tilde{\mathbf{E}}) = \sum_{k=1}^{L} \left\| \mathbf{\Omega}_k \circ \left(\hat{\mathbf{E}}_k - \tilde{\mathbf{E}} \right) \right\|_F^2, \tag{5}$$

where $\circ$ denotes the Hadamard product, and $\mathrm{EDM}(N)$ is the set of EDMs of size $N \times N$. To solve problem (5) in a fully decentralized fashion, we use the algorithm introduced in [28], which in turn derives from the framework of diffusion adaptation (DA) for optimization [36] and on previous works on EDM completion [26]. In particular, we approximate the objective function in Eq. (5) by:

$$J_k(\mathbf{V}) = \left\| \mathbf{\Omega}_k \circ \left[\hat{\mathbf{E}}_k - \kappa\left(\mathbf{V}\mathbf{V}^{\mathrm{T}}\right) \right] \right\|_{\mathrm{F}}^2 \; k = 1, \ldots, L, \tag{6}$$

where $\kappa(\cdot)$ is the Schoenberg mapping, which maps every positive semidefinite (PSD) matrix to an EDM, given by:

$$\kappa(\mathbf{E}) = \mathrm{diag}(\mathbf{E})\mathbf{1}^{\mathrm{T}} + \mathbf{1}\mathrm{diag}(\mathbf{E})^{\mathrm{T}} - 2\mathbf{E}, \tag{7}$$

such that $\mathrm{diag}(\mathbf{E})$ extracts the main diagonal of $\mathbf{E}$ as a column vector, and we also exploits the known fact that any PSD matrix $\mathbf{D}$ with rank r admits a factorization $\{\mathbf{D} = \mathbf{V}\mathbf{V}^{\mathrm{T}}\}$, where $\mathbf{V} \in \mathbb{R}_*^{N \times r} = \{\mathbf{V} \in \mathbb{R}^{N \times r} : \det(\mathbf{V}^{\mathrm{T}}\mathbf{V}) \neq 0\}$. This allows to strongly reduce the computational cost of our algorithm, as the objective function is now formulated only in terms of the low-rank factor $\mathbf{V}$.

The diffusion gradient descent for the distributed completion of the EDM is then defined by an alternation of updating and diffusion equations in the form of [28]:

1) **Initialization**: All the agents initialize the local matrices $\mathbf{V}_k$ as random $N \times r$ matrices.
2) Update of $\mathbf{V}$: At time n, the kth agent updates the local matrix $\mathbf{V}_k$ using a gradient descent step with respect to its local cost function:

$$\tilde{\mathbf{V}}_k[n+1] = \mathbf{V}_k[n] - \eta_k[n] \nabla_{\mathbf{V}_k} J_k(\mathbf{V}). \tag{8}$$

ALGORITHM I
PSEUDOCODE OF THE PROPOSED DISTRIBUTED SPECTRAL CLUSTERING ALGORITHM, AT THE kTH AGENT.

Input: Local dataset S_k, number of nodes L (global), maximum number of iterations T.
for $n = 1$ to $n_{\max}^{(1)}$ **do**
  Select a set of input patterns and share them with the neighbors $\mathcal{N}_k$.
  Receive patterns from the neighbors.
end for
Compute the incomplete EDM matrix $\hat{\mathbf{E}}_k$.
for $n = 1$ to $n_{\max}^{(2)}$ **do**
  Select a set of entries from $\hat{\mathbf{E}}_k$ and share them with the neighbors.
  Receive entries from the neighbors.
  Update $\hat{\mathbf{E}}_k$ with the entries received.
end for
Initialize $\mathbf{V}_k[0]$.
for $n = 1$ to T **do**
  Compute $\mathbf{V}_k[n]$ using Eq. (8).
  Diffuse local information using Eq. (10).
end for
Compute the Laplacian matrix $\tilde{\mathbf{L}}$ from $\tilde{\mathbf{E}}$.
Perform SC using $\tilde{\mathbf{L}}$.

where $\eta_k[n]$ is a positive step-size. It is straightforward to show that the gradient of the cost function is given by:

$$\nabla_{\mathbf{V}_k} J_k(\mathbf{V}) = \kappa^* \Big\{ \mathbf{\Omega}_k \circ \\ \circ \Big(\kappa \left(\mathbf{V}_k[n] \, \mathbf{V}_k^{\mathrm{T}}[n] \right) - \hat{\mathbf{E}}_k \Big) \Big\} \mathbf{V}_k[n] \,, \quad (9)$$

where $\kappa^*(\mathbf{A}) = 2\left[\mathrm{diag}\left(\mathbf{A}1\right) - \mathbf{A}\right]$ is the adjoint operator of $\kappa(\cdot)$.

3) **Diffusion**: In order to propagate information over the network, the updated matrices are combined according to some mixing weights $\mathbf{C} \in \mathbb{R}^{L \times L}$:

$$\mathbf{V}_k[n+1] = \sum_{i=1}^{L} C_{ki} \tilde{\mathbf{V}}_i[n+1] \,. \quad (10)$$

where $C_{ki} > 0$ if and only if agents k and i are connected, in order to send information only through neighbors. The mixing weights are generally chosen to provide a convex combination at every agent. A simple choice, which is used throughout our experimental results, are the so-called 'max-degree' weights:

$$C_{kj} = \begin{cases} \frac{1}{d+1} & \text{if } k \in \mathcal{N}_j \\ 1 - \frac{d_k}{d+1} & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} . \quad (11)$$

where $\mathcal{N}_k$ is the set of neighbors of node k, and d is the degree of the graph [37].

For a rationale of this approach, and an analysis of its convergence behavior in the case of convex cost functions, we refer to any introductory publication on DA [2], [36]. Convergence of a similar family of algorithms in the case of non-convex cost functions is instead derived in [38]. The proposed algorithm for distributed SC over networks is summarized in Algorithm I.

IV. EXPERIMENTAL EVALUATION

A. Experimental Setup

In this section, we evaluate the performance of the proposed algorithm when compared with a fully centralized approach, where local data is sent beforehand to a (virtual) centralized processor. We consider six different public datasets available on the UCI repository,[2] a schematic description of which is given in Table II. Except for the TwoMoons dataset, no graphical visualization of them is possible. In all cases, the optimal clustering is known beforehand for testing purposes, either in the case of classification datasets (where clusters correspond to classes), or because the dataset is artificially generated. In the following we present some additional information on each of them.

- *Australian credit approval*: It is composed by 690 instances and 14 attributes. It is a binary classification dataset, which concerns credit card applications. It is interesting because it contains a good mixture of attributes, both continuous and nominal.
- *Hill-Valley*: It is a classification dataset composed by 100 features on a two-dimensional graph. The task is to identify whenever the plotted points will create a Hill (a "bump" in the terrain) or a Valley (a "dip" in the terrain). There are 606 instances in the complete dataset.
- *Ionosphere*: The Ionosphere dataset consists of 16 high-frequency antennas and 17 pulse number for the Goose Bay system. Instances are described by 2 attributes per pulse number. 351 datapoints composed by 34 attributes each are used to classify radar returns in "good" returns or "bad" returns depending on the evidence of some type of structure in the ionosphere.
- *LSVT Voice Rehabilitation*: It is a classification dataset composed by 126 samples and 309 features. The aim is to assess whether voice rehabilitation treatment leads to phonations considered 'acceptable' or 'unacceptable'. Each feature corresponds to the application of a speech signal processing algorithm which has to characterize objectively the signal.
- *Twomoons*: It is a binary cluster dataset of 400 instances composed by two attributes. The dataset contains two clusters, whose shape is similar to a waxing and waning crescent moon.
- *Vehicle Silhouettes*: It contains four classes of 752 vehicles to classify with a set of 18 features extracted from their silhouette. The vehicles may be viewed from one of many different angles. Four "Corgie" model vehicles

[2] https://archive.ics.uci.edu/ml/datasets.html

TABLE II
DETAILED DESCRIPTION OF EACH DATASET. ADDITIONAL INFORMATION ON THEM IS PROVIDED IN SECTION IV-A.

Dataset	Features	Instances	Classes	Original Task
Australian credit approval	14	690	2	Classification
Hill-Valley	100	606	2	Classification
Ionosphere	34	351	2	Classification
LSVT Voice Rehabilitation	309	126	2	Classification
Twomoons	2	400	2	Clustering
Vehicle Silhouettes	18	752	4	Classification

must be recognized: a decker bus, Cheverolet van, Saab 9000 and an Opel Manta 400. This particular combination of vehicles is such that the bus, van and either one of the cars would be distinguishable, but it would be more difficult to distinguish between cars.

In all cases input features were normalized between -1 and 1 before the experiments. All experiments are then averaged over 5 different runs of simulations. In each run we consider a random topology of 7 nodes according to the so-called "Erdos-Rènyi model" [37] such that every pair of nodes is connected with a fixed probability $p = 0.5$. The only global requirement is that the overall topology is connected.

At each round the overall dataset is randomly partitioned among the agents (as in Fig. 1). We compare the following two algorithms:

- *Distributed Spectral Clustering*: this is the previously described approach where the Laplacian Matrix is evaluated in a distributed fashion using the distributed EDM completion algorithm presented in Section III-B. For the initial exchange phase, we set a small value of $n_{\max}^{(1)} = n_{\max}^{(2)} = 150$. The EDM estimation algorithm is run for $T = 1000$ iterations. For the step-size, we used a fixed step-size strategy. In particular, the optimal values of η is chosen searching it in the exponential interval 10^j with $j \in \{-10, -9, ..., 2, 3\}$.
- *Centralized approach*: this is equivalent to having a centralized agent, where the traditional spectral clustering algorithm is performed on the global dataset. It is used as an optimal benchmark to evaluate our approach.

We use a standard SC procedure, where the Laplacian matrix is constructed using Eq. (1) with $\gamma = 1$, which was found to work relatively well on all datasets. All experiments are carried out using MATLAB R2013b on a machine with Intel Core i5 processor with a CPU @ 3.00 GHz with 16 GB of RAM. The employed k-means is the one included in the Statistics and Machine Learning Toolbox of MATLAB.

B. Results

The goal of the experiment was to evaluate how the proposed distributed EDM algorithm can influence the performance of several quality indexes when compared with a centralized algorithm. We start our discussion of the results by analyzing the performance of four quality indexes that are computed for both the centralized and the distributed approach:

- Rand Index [39]
- Falks-Mallows [40]
- F-measure [40]
- K-Index [41]

Precisely all of the indexes range in $[0, 1]$, with 1 indicating a perfect correlation between the true label of the cluster and the output of the clustering algorithm, and 0 the perfect negative correlation. The results are summarized in Table III where they have been averaged over 10 k-means evaluations and over the different agents in the distributed case. For each dataset the mean and the standard deviation of each quality index is computed. The best result for each dataset and for each index is highlighted in bold.

As we can see, the results of the two approaches are reasonably aligned. Whereas the centralized approach can significantly boost performance for almost all the quality indexes in the Ionosphere dataset, in the other example results are more comparable. In particular in the Australian credit approval, Hill-Valley, LSVT Voice Rehabilitation and Twomoons datasets the values of the quality indexes are very similar, while in the Vehicle Silhouettes dataset our approach outperforms the centralized one with respect to the F-measure and the F-M Index. The most important result suggested from Table III is that the distributed EDM computation can indeed be an effective approach to be applied in a distributed scenario, since it is able to match very closely the performance of the centralized spectral clustering algorithm.

To further strengthen the results, a visual representation of the Rand-Index is given in Fig 3. As it has been noted the two approaches present a similar behavior in the majority of datasets. A slightly worse trend is reported for the Ionosphere dataset, but it is offset by the behavior of the Vehicle Silhouettes dataset that presents an increase of nearly 20% of the Rand Index.

To reinforce our conclusion, we also report the completion error during the iterations of the distributed EDM estimation algorithm. The error is evaluated at each iteration in the

TABLE III

EXPERIMENTAL RESULTS ON THE DIFFERENT DATASET. WE SHOW THE AVERAGE AND THE STANDARD DEVIATION OF THE F-INDEX, RAND INDEX, KAPPA INDEX, FM-INDEX FOR BOTH THE CENTRALIZED (C) AND THE DISTRIBUTED (D) APPROACHES. BEST RESULTS FOR EACH ALGORITHM ARE HIGHLIGHTED IN BOLD.

Dataset	Algorithm	F-Measure	Rand-Index	K-Index	F-M Index
Australian credit approval	C	0.445 ± 0.198	0.499 ± 0.004	0.028 ± 0.014	**0.564 ± 0.041**
	D	**0.473 ± 0.000**	**0.500 ± 0.000**	**0.032 ± 0.000**	0.504 ± 0.000
HIll-Valley	C	0.488 ± 0.178	**0.501 ± 0.001**	**0.042 ± 0.019**	**0.596 ±0.059**
	D	**0.506 ± 0.024**	0.500 ± 0.000	0.029 ± 0.000	0.499 ± 0.000
Ionosphere	C	**0.944 ± 0.000**	**0.811 ± 0.000**	0.046 ± 0.000	**0.655 ± 0.000**
	D	0.641 ± 0.035	0.504 ± 0.000	**0.119 ± 0.000**	0.638 ± 0.000
LSVT Voice Rehabilitation	C	0.418 ± 0.000	**0.509 ± 0.000**	0.091 ± 0.000	**0.542 ± 0.000**
	D	**0.434 ± 0.113**	0.499 ± 0.000	**0.139 ± 0.000**	0.532 ± 0.000
Twomoons	C	0.723 ± 0.361	**0.813 ± 0.000**	**0.792 ± 0.000**	**0.817 ± 0.000**
	D	**0.788 ± 0.254**	0.751 ± 0.000	0.711 ± 0.000	0.752 ± 0.000
Vehicle Silhouettes	C	**0.222 ± 0.139**	0.448 ± 0.036	0.026 ± 0.007	**0.386 ± 0.023**
	D	0.170 ± 0.080	**0.650 ± 0.000**	**0.187 ± 0.000**	0.312 ± 0.000

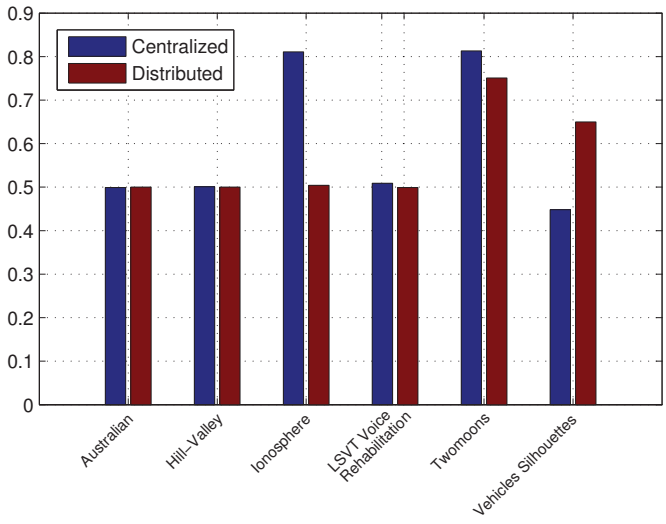


Fig. 3. The Rand Index for all of the described datasets for both the Centralized Spectral Clustering algorithm and the Distributed approach.

following way:

$$\text{Err} = \frac{1}{L}\sum_{k=1}^{L}\frac{\left\|\mathbf{E}-\tilde{\mathbf{E}}_k\right\|}{\|\mathbf{E}\|}. \tag{12}$$

The convergence of the proposed approach, averaged over the different runs, is summarized in Fig. 4, where on the x-axis is reported the iteration number and on the y-axis the EDM completion error. The reported trend proves that our approach is able to rapidly converge to the real EDM. We see that, with the solely exception of the Twomoons, the algorithm is able to obtain very similar results in all of the datasets, like the Hill-Valley dataset, the LSVT voice or Vehicle Silhouettes, where in a few number of iterations it obtains very low errors. The error is independent from the size of the dataset, although a low number of steps are necessary to reach convergence to a reasonable accuracy.

V. CONCLUSIONS

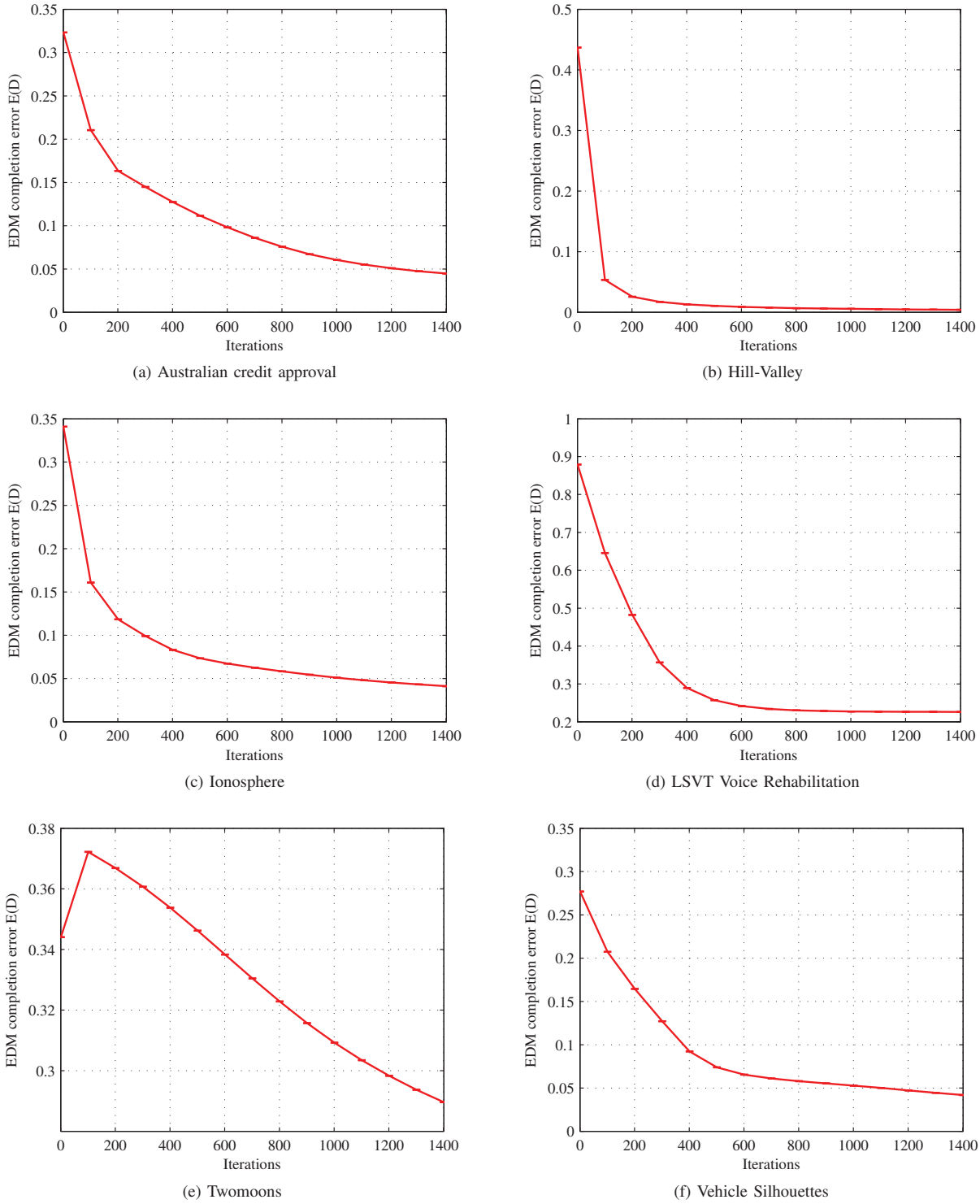
In this paper, we have introduced a fully distributed procedure for performing spectral clustering over networks of computing agents. In particular, we considered the equivalent problem of completing the matrix containing the pairwise distances among all datapoints. This is obtained via a distributed gradient procedure, interleaving gradient descent steps with point-to-point diffusion of information. Overall, our experimental results suggest that the procedure is able to efficiently match a fully centralized implementation, without however requiring the presence of a coordinating node, and using only in-network communication.

In the near future, we intend to test the resulting algorithm on multiple real-world distributed clustering applications, including distributed text mining and medical diagnosing. We also plan to consider more elaborate SC procedures based on the distributed computation of the Laplacian (e.g. using normalized versions of it), and to remove some of the assumptions made on this paper, particularly in relation to asynchronous message passing and time-varying connectivity of the agents' network.

REFERENCES

[1] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, 2014.

[2] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, 2014.



(a) Australian credit approval

(b) Hill-Valley

(c) Ionosphere

(d) LSVT Voice Rehabilitation

(e) Twomoons

(f) Vehicle Silhouettes

Fig. 4. Average EDM completion error of the distributed gradient procedure, at each iteration. The vertical bars represent the standard deviation from the error.

[3] H. H. Ang, V. Gopalkrishnan, S. C. Hoi, and W. K. Ng, "Classification in P2P networks with cascade support vector machines," *ACM Transactions on Knowledge Discovery from Data*, vol. 7, no. 4, p. 20, 2013.

[4] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, 2006.

[5] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Distributed detection and estimation in wireless sensor networks," in *E-Reference Signal Processing*, R. Chellapa and S. Theodoridis, Eds. Elsevier, 2013, pp. 329–408.

[6] S. Squartini, D. Liu, F. Piazza, D. Zhao, and H. He, "Computational Energy Management in Smart Grids," *Neurocomputing*, vol. 170, pp. 267 – 269, 2015.

[7] S. Scardapane, R. Fierimonte, D. Wang, M. Panella, and A. Uncini, "Distributed Music Classification using Random Vector Functional-Link Nets," in *2015 International Joint Conference on Neural Networks (IJCNN'15)*. IEEE, 2015, pp. 1–8.

[8] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM SiGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[9] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, 2010.

[10] S. Scardapane, D. Wang, and M. Panella, "A Decentralized Training Algorithm for Echo State Networks in Distributed Big Data Applications," *Neural Networks*, 2015, under press.

[11] S. Scardapane, D. Wang, M. Panella, and A. Uncini, "Distributed learning for Random Vector Functional-Link networks," *Information Sciences*, vol. 301, pp. 271–284, 2015.

[12] L. Georgopoulos and M. Hasler, "Distributed machine learning in networks by consensus," *Neurocomputing*, vol. 124, pp. 2–12, Jan. 2014.

[13] G. Jagannathan and R. N. Wright, "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 593–599.

[14] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 206–215.

[15] S. Merugu and J. Ghosh, "Privacy-preserving distributed clustering using generative models," in *Third IEEE International Conference on Data Mining (ICDM)*. IEEE, 2003, pp. 211–218.

[16] S. Greco, F. Gullo, G. Ponti, and A. Tagarelli, "Collaborative clustering of XML documents," in *International Conference on Parallel Processing Workshops (ICPPW)*. IEEE, 2009, pp. 579–586.

[17] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 4, pp. 366–379, 2004.

[18] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[19] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[20] F. R. Bach and M. I. Jordan, "Learning spectral clustering, with application to speech separation," *Journal of Machine Learning Research*, vol. 7, pp. 1963–2001, 2006.

[21] R. Liu and H. Zhang, "Segmentation of 3D meshes through spectral clustering," in *Proc. of the 12th Pacific Conference on Computer Graphics and Applications*. IEEE, 2004, pp. 298–305.

[22] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 568–586, 2011.

[23] W.-Y. Chen, Y. Song, H. Bai, C.-J. en Lin, and E. Y. Chang, "Large-Scale Spectral Clustering with MapReduce and MP1," *Scaling up Machine Learning: Parallel and Distributed Approaches*, 2011.

[24] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[25] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proc. IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[26] B. Mishra, G. Meyer, and R. Sepulchre, "Low-rank optimization for distance matrix completion," in *2011 50th IEEE conference on Decision and control and European control conference (CDC-ECC'11)*. IEEE, 2011, pp. 4455–4460.

[27] Q. Ling, Y. Xu, W. Yin, and Z. Wen, "Decentralized low-rank matrix completion," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'12)*. IEEE, 2012, pp. 2925–2928.

[28] R. Fierimonte, S. Scardapane, A. Uncini, and M. Panella, "Fully decentralized semi-supervised learning via privacy-preserving matrix completion," *IEEE Transactions on Neural Networks and Learning Systems*, 2016, accepted with minor revision.

[29] J. N. Tsitsiklis, D. P. Bertsekas, M. Athans *et al.*, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, 1986.

[30] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, 2012.

[31] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 92–106, 2006.

[32] C. C. Aggarwal and C. K. Reddy, *Data clustering: algorithms and applications*. CRC Press, 2013.

[33] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 12, pp. 1650–1654, 2002.

[34] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol. 1: Theory*. SIAM, 2002, vol. 41.

[35] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[36] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[37] M. Newman, *Networks: an introduction*. Oxford University Press, 2010.

[38] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE Trans. Autom. Control*, vol. 58, no. 2, pp. 391–405, 2013.

[39] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[40] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2, pp. 107–145, 2001.

[41] J. Cohen, "A coefficient of agreement for nominal scales, Educational and Psychological Measurement," *Educational and Psychological Measurement*, 1960.