# Large-scale clustering algorithms

Rocco Langone, Vilen Jumutc and Johan A. K. Suykens

**Abstract** Computational tools in modern data analysis must be scalable to satisfy business and research time constraints. In this regard, two alternatives are possible: i) adapt available algorithms or design new approaches such that they can run on a distributed computing environment ii) develop model-based learning techniques that can be trained efficiently on a small subset of the data and make reliable predictions. In this chapter two recent algorithms following these different directions are reviewed. In particular, in the first part a scalable in-memory spectral clustering algorithm is described. This technique relies on a kernel-based formulation of the spectral clustering problem also known as kernel spectral clustering. More precisely, a finite dimensional approximation of the feature map via the Nyström method is used to solve the primal optimization problem, which decreases the computational time from cubic to linear. In the second part, a distributed clustering approach with fixed computational budget is illustrated. This method extends the k-means algorithm by applying regularization at the level of prototype vectors. An optimal stochastic gradient descent scheme for learning with $l_1$ and $l_2$ norms is utilized, which makes the approach less sensitive to the influence of outliers while computing the prototype vectors.

**Key words:** Data Clustering · Big Data · Kernel Methods · Nyström Approximation · Stochastic Optimization · K-means · Map-reduce · Regularization · In-memory algorithms · Scalability

---

Rocco Langone (✉)
KU Leuven ESAT-STADIUS, Kasteelpark Arenberg 10 B-3001 Leuven (Belgium), e-mail: rocco.langone@esat.kuleuven.be

Vilen Jumutc
KU Leuven ESAT-STADIUS, Kasteelpark Arenberg 10 B-3001 Leuven (Belgium), e-mail: vilen.jumutc@esat.kuleuven.be

Johan A. K. Suykens
KU Leuven ESAT-STADIUS, Kasteelpark Arenberg 10 B-3001 Leuven (Belgium), e-mail: johan.suykens@esat.kuleuven.be

## 1 Introduction

Data clustering allows to partition a set of points into groups called clusters which are as similar as possible. It plays a key role in computational intelligence because of its diverse applications in various domains. Examples include collaborative filtering and market segmentation, where clustering is used to provide personalized recommendations to users, trend detection which allows to discover key trends events in streaming data, community detection in social networks, and many others [1].

With the advent of the big data era, a key challenge for data clustering lies in its scalability, that is, how to speed-up a clustering algorithms without affecting its performance. To this purpose, two main directions have been explored [1]: (i) sampling-based algorithms or techniques using random projections (ii) parallel and distributed methods. The first type of algorithms allows to tackle the computational complexity due either to the large amount of data instances or their high dimensionality. More precisely, sampling-based algorithms perform clustering on a sample of the datasets and then generalize it to whole dataset. As a consequence, execution time and memory space decrease. Examples of such algorithms are CLARANS [2], which tries to find the best medoids representing the clusters, BIRCH [3], where a new data structure called clustering feature is introduced in order to reduce the I/O cost in the in-memory computational time, CURE [4], which uses a set of well-scattered data points to represent a cluster in order to detect general shapes. Randomized techniques reduce the dimension of the input data matrix by transforming it into a lower dimensional space and then perform clustering on this reduced space. In this framework, [5] uses random projections to speed-up the k-means algorithm. In [6], a method called Colibri allows to cluster large static and dynamic graphs. In contrast to the typical single machine clustering, parallel algorithms use multiple machines or multiple cores in a single machine to speed up the computation and increase the scalability. Furthermore, they can be either memory-based if the data fit in the memory and each machine/core can load it, or disk-based algorithm which use Map-Reduce [7] to process huge amounts of disk-resident data in a massively parallel way. An example of memory-based algorithm is ParMETIS [8], which is a parallel graph-partitioning approach. Disk-based methods include parallel k-means [9], a k-means algorithm implemented on Map-Reduce and a distributed co-clustering algorithm named DisCO [10]. Finally, the interested reader may refer to [11] and [12] for some recent surveys on clustering algorithms for big data.

In this chapter two algorithms for large-scale data clustering are reviewed. The first one, named fixed-size kernel spectral clustering (FSKSC), is a sampling-based spectral clustering method. Spectral clustering (SC) [13, 14, 15, 16] has been shown to be among the most effective clustering algorithms. This is mainly due to its ability of detecting complex nonlinear structures thanks to the mapping of the original data into the space spanned by the eigenvectors of the Laplacian matrix. By formulating the spectral clustering problem within a least squares support vector machine setting [17], kernel spectral clustering [18, 19] (KSC) allows to tackle its main drawbacks represented by the lack of a rigorous model selection procedure and a systematic out-of-sample property. However, when the number of training data is large the

complexity of constructing the Laplacian matrix and computing its eigendecomposition can become intractable. In this respect, the FSKSC algorithm represents a solution to this issue which exploits the Nyström method [20] to avoid the construction of the kernel matrix and therefore reduce the time and space costs. The second algorithm that will be described is a distributed k-means approach which extends the k-means algorithm by applying $l_1$ and $l_2$ regularization to enforce the norm of the prototype vectors to be small. This allows to decrease the sensitivity of the algorithm to both the initialization and the presence of outliers. Furthermore, either stochastic gradient descent [21] or dual averaging [22] are used to learn the prototype vectors, which are computed in parallel on a multi-core machine [1].

The remainder of the Chapter is organized as follows. Section 3 summarizes the standard spectral clustering and k-means approaches. In Section 4 the fixed-size KSC method will be presented. Section 5 is devoted to summarize the regularized stochastic k-means algorithm. Afterwards, some experimental results will be illustrated in Section 6. Finally some conclusions are given.

## 2 Notation

| | |
|---|---|
| $\mathbf{x}^T$ | Transpose of the vector $\mathbf{x}$ |
| $\mathbf{A}^T$ | Transpose of the matrix $\mathbf{A}$ |
| $\mathbf{I}_N$ | $N \times N$ Identity matrix |
| $\mathbf{1}_N$ | $N \times 1$ Vector of ones |
| $\mathscr{D}_{\text{tr}} = \{\mathbf{x}_i\}_{i=1}^{N_{\text{tr}}}$ | Training sample of $N_{\text{tr}}$ data points |
| $\varphi(\cdot)$ | Feature map |
| $\mathscr{F}$ | Feature space of dimension $d_h$ |
| $\{\mathscr{C}_p\}_{p=1}^k$ | Partitioning composed of $k$ clusters |
| $\lvert\cdot\rvert$ | Cardinality of a set |
| $\lvert\lvert\cdot\rvert\rvert_p$ | $p$-norm of a vector |
| $\nabla f$ | Gradient of function $f$ |

## 3 Standard Clustering Approaches

### *3.1 Spectral Clustering*

Spectral clustering represents a solution to the graph partitioning problem. More precisely, it allows to divide a graph into weakly connected sub-graphs by making use of the spectral properties of the graph Laplacian matrix [14, 15, 13].

A graph (or network) $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ is a mathematical structure used to model pairwise relations between certain objects. It refers to a set of $N$ vertices or nodes

[1] The same schemes can be extended with little effort to a multiple machine framework.

$\mathscr{V} = \{v_i\}_{i=1}^N$ and a collection of edges $\mathscr{E}$ that connect pairs of vertices. If the edges are provided with weights the corresponding graph is weighted, otherwise it is referred as an unweighted graph. The topology of a graph is described by the similarity or affinity matrix, which is an $N \times N$ matrix $\boldsymbol{S}$, where $S_{ij}$ indicates the link between the vertices $i$ and $j$. Associated to the similarity matrix there is the degree matrix $\boldsymbol{D} = \text{diag}(\boldsymbol{d}) \in \mathbb{R}^{N \times N}$ , with $\boldsymbol{d} = [d_1, \ldots, d_N]^T = \boldsymbol{S}\boldsymbol{1}_N \in \mathbb{R}^{N \times 1}$ and $\boldsymbol{1}_N$ indicating the $N \times 1$ vector of ones. Basically the degree $d_i$ of node $i$ is the sum of all the edges (or weights) connecting node $i$ with the other vertices: $d_i = \sum_{j=1}^N S_{ij}$.

The most basic formulation of the graph partitioning problem seeks to split an unweighted graph into $k$ non-overlapping sets $\mathscr{C}_1, \ldots, \mathscr{C}_k$ with similar cardinality in order to minimize the cut size, which is the number of edges running between the groups. The related optimization problem is referred as the normalized cut (NC) objective defined as:

$$\begin{aligned} \min_{\boldsymbol{G}} \quad & k - \text{tr}(\boldsymbol{G}^T \boldsymbol{L}_n \boldsymbol{G}) \\ \text{subject to} \quad & \boldsymbol{G}^T \boldsymbol{G} = \boldsymbol{I} \end{aligned} \tag{1}$$

where:

- $\boldsymbol{L}_n = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{S} \boldsymbol{D}^{-\frac{1}{2}}$ is called the normalized Laplacian
- $\boldsymbol{G} = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k]$ is the matrix containing the normalized cluster indicator vectors $\boldsymbol{g}_l = \frac{\boldsymbol{D}^{\frac{1}{2}} \boldsymbol{f}_l}{||\boldsymbol{D}^{\frac{1}{2}} \boldsymbol{f}_l||_2}$
- $\boldsymbol{f}_l$, with $l = 1, \ldots, k$, is the cluster indicator vector for the $l$-th cluster. It has a 1 in the entries corresponding to the nodes in the $l$-th cluster and 0 otherwise. Moreover, the cluster indicator matrix can be defined as $\boldsymbol{F} = [\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k] \in \{0,1\}^{N \times k}$
- $\boldsymbol{I}$ denotes the identity matrix.

Unfortunately this is a NP-hard problem. However, approximate solutions in polynomial time can be obtained by relaxing the entries of $\boldsymbol{G}$ to take continuous values:

$$\begin{aligned} \min_{\hat{\boldsymbol{G}}} \quad & k - \text{tr}(\hat{\boldsymbol{G}}^T \boldsymbol{L}_n \hat{\boldsymbol{G}}) \\ \text{subject to} \quad & \hat{\boldsymbol{G}}^T \hat{\boldsymbol{G}} = \boldsymbol{I}. \end{aligned} \tag{2}$$

with $\hat{\boldsymbol{G}} \in \mathbb{R}^{N \times k}$ Solving problem (2) is equivalent to finding the solution to the following eigenvalue problem:

$$\boldsymbol{L}_n \boldsymbol{g} = \lambda \boldsymbol{g}. \tag{3}$$

Basically, the relaxed clustering information is contained in the eigenvectors corresponding to the $k$ smallest eigenvalues of the normalized Laplacian $\boldsymbol{L}_n$. In addition to the normalized Laplacian, other Laplacians can be defined, like the unnormalized Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S}$ and the random walk Laplacian $\boldsymbol{L}_{rw} = \boldsymbol{D}^{-1} \boldsymbol{S}$. The latter owes its name to the fact that it represents the transition matrix of a random walk associated to the graph, whose stationary distribution describes the situation in which the

random walker remains most of the time in the same cluster with rare jumps to the other clusters [23].

Spectral clustering suffers from a scalability problem in both memory usage and computational time when the number of data instances $N$ is large. In particular, time complexity is $O(N^3)$, which is needed to solve eigenvalue problem (3), and and space complexity is $O(N^2)$, which is required to store the Laplacian matrix. In Section 4 the fixed-size KSC method will be thoroughly discussed, and some related works representing different solutions to this scalability issue will be briefly reviewed in Section 4.0.1.

### *3.2 K-means*

Given a set of observations $\mathscr{D} = \{\mathbf{x}_i\}_{i=1}^N$, with $\mathbf{x}_i \in \mathbb{R}^d$, k-means clustering [24] aims to partition the data sets into $k$ subsets $\mathscr{S}_1, \ldots, \mathscr{S}_k$, so as to minimize the distortion function, that is the sum of distances of each point in every cluster to the corresponding center. This optimization problem can be expressed as follows:

$$\min_{\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(k)}} \sum_{l=1}^{k} [\frac{1}{2N_l} \sum_{\mathbf{x}\in\mathscr{S}_l} \|\boldsymbol{\mu}^{(l)} - \mathbf{x}\|_2^2], \tag{4}$$

where $\boldsymbol{\mu}^{(l)}$ is the mean of the points in $\mathscr{S}_l$. Since this problem is NP-hard, an alternate optimization procedure similar to the expectation-maximization algorithm is employed, which converges quickly to a local optimum. In practice, after randomly initializing the cluster centers, an assignment and an update step are repeated until the cluster memberships no longer change. In the assignment step each point is assigned to the closest center, i.e. the cluster whose mean yields the least within-cluster sum of squares. In the update step, the new cluster centroids are calculated.

The outcomes produced by the standard k-means algorithm are highly sensitive to the initialization of the cluster centers and the presence of outliers. In Section 5 we further discuss the regularized stochastic k-means approach which, similarly to other methods briefly reviewed in Section 5.1, allows to tackle these issues through stochastic optimization approaches.

## 4 Fixed-Size Kernel Spectral Clustering

In this Section we review an alternative approach to scale-up spectral clustering named fixed-size kernel spectral clustering, which was recently proposed in [25]. Compared to the existing techniques, the major advantages of this method are the possibility to extend the clustering model to new out-of-sample points and a precise model selection scheme.

#### 4.0.1 Related work

Several algorithms have been devised to speed-up spectral clustering. Examples include power iteration clustering [26], spectral grouping using the Nyström method [27], incremental algorithms where some initial clusters computed on an initial subset of the data are modified in different ways [28, 29, 30], parallel spectral clustering [31], methods based on the incomplete Cholesky decomposition [32, 33, 34], landmark-based spectral clustering [35], consensus spectral clustering [36], vector quantization based approximate spectral clustering [37], approximate pairwise clustering [38].

### *4.1 KSC overview*

The multiway kernel spectral clustering (KSC) formulation is stated as a combination of $k-1$ binary problems, where $k$ denotes the number of clusters [19]. More precisely, given a set of training data $\mathscr{D}_{\text{tr}} = \{\mathbf{x}_i\}_{i=1}^{N_{\text{tr}}}$, the primal problem is expressed by the following objective:

$$\begin{aligned} \min_{\mathbf{w}^{(l)},\mathbf{e}^{(l)},b_l} \quad & \frac{1}{2}\sum_{l=1}^{k-1}\mathbf{w}^{(l)^T}\mathbf{w}^{(l)} - \frac{1}{2}\sum_{l=1}^{k-1}\gamma_l\mathbf{e}^{(l)^T}V\mathbf{e}^{(l)} \\ \text{subject to} \quad & \mathbf{e}^{(l)} = \boldsymbol{\Phi}\mathbf{w}^{(l)} + b_l\mathbf{1}_{N_{\text{tr}}}, l = 1,\ldots,k-1. \end{aligned} \tag{5}$$

The $\mathbf{e}^{(l)} = [e_1^{(l)},\ldots,e_i^{(l)},\ldots,e_{N_{\text{tr}}}^{(l)}]^T$ denotes the projections of the training data mapped in the feature space along the direction $\mathbf{w}^{(l)}$. For a given point $\mathbf{x}_i$, the corresponding clustering score is given by:

$$e_i^{(l)} = \mathbf{w}^{(l)^T}\varphi(\mathbf{x}_i) + b_l. \tag{6}$$

In fact, as in a classification setting, the binary clustering model is expressed by an hyperplane passing through the origin, that is $e_i^{(l)} - \mathbf{w}^{(l)^T}\varphi(\mathbf{x}_i) - b_l = 0$. Problem (5) is nothing but a weighted kernel PCA in the feature space $\varphi : \mathbb{R}^d \to \mathbb{R}^{d_h}$, where the aim is to maximize the weighted variances of the scores, i.e. $\mathbf{e}^{(l)^T}V\mathbf{e}^{(l)}$ while keeping the squared norm of the vector $\mathbf{w}^{(l)}$ small. The constants $\gamma_l \in \mathbb{R}^+$ are regularization parameters, $\mathbf{V} \in \mathbb{R}^{N_{\text{tr}}\times N_{\text{tr}}}$ is the weighting matrix and $\boldsymbol{\Phi}$ is the $N_{\text{tr}} \times d_h$ feature matrix $\boldsymbol{\Phi} = [\varphi(\mathbf{x}_1)^T;\ldots;\varphi(\mathbf{x}_{N_{\text{tr}}})^T]$, $b_l$ are bias terms.

The dual problem associated to (5) is given by:

$$\mathbf{V}\mathbf{M}_V\boldsymbol{\Omega}\boldsymbol{\alpha}^{(l)} = \lambda_l\boldsymbol{\alpha}^{(l)} \tag{7}$$

where $\boldsymbol{\Omega}$ denotes the kernel matrix with $ij$-th entry $\boldsymbol{\Omega}_{ij} = K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$. $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ means the kernel function. $\mathbf{M}_V$ is a centering matrix defined as $\mathbf{M}_V = \mathbf{I}_{N_{\text{tr}}} - \frac{1}{\mathbf{1}_{N_{\text{tr}}}^T\mathbf{V}\mathbf{1}_{N_{\text{tr}}}}\mathbf{1}_{N_{\text{tr}}}\mathbf{1}_{N_{\text{tr}}}^T\mathbf{V}$, the $\boldsymbol{\alpha}^{(l)}$ are vectors of dual variables, $\boldsymbol{\lambda}_l = \frac{N_{\text{tr}}}{\gamma_l}$. By

setting[2] $\mathbf{V} = \mathbf{D}^{-1}$, being $\mathbf{D}$ the graph degree matrix which is diagonal with positive elements $D_{ii} = \sum_j \Omega_{ij}$, problem (7) is closely related to spectral clustering with random walk Laplacian [23, 23, 42, 43], and objective (5) is referred as the kernel spectral clustering problem.

The dual clustering model for the $i$-th training point can be expressed as follows:

$$e_i^{(l)} = \sum_{j=1}^{N_{\text{tr}}} \alpha_j^{(l)} K(\mathbf{x}_j, \mathbf{x}_i) + b_l, j = 1, \ldots, N_{\text{tr}}, l = 1, \ldots, k-1. \tag{8}$$

By binarizing the projections $e_i^{(l)}$ as $\text{sign}(e_i^{(l)})$ and selecting the most frequent binary indicators, a code-book $\mathscr{CB} = \{c_p\}_{p=1}^k$ with the $k$ cluster prototypes can be formed. Then, for any given point (either training or test), its cluster membership can be computed by taking the sign of the corresponding projection and assigning to the cluster represented by the closest prototype in terms of hamming distance. The KSC method is summarized in algorithm 1, and the related Matlab package is freely available on the Web[3]. Finally, the interested reader can refer to the recent review [18] for more details on the KSC approach and its applications.

**Algorithm 1:** KSC algorithm [19]

**Data**: Training set $\mathscr{D}_{\text{tr}} = \{\mathbf{x}_i\}_{i=1}^{N_{\text{tr}}}$, test set $\mathscr{D}_{\text{test}} = \{\mathbf{x}_r^{\text{test}}\}_{r=1}^{N_{\text{test}}}$ kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, kernel parameters (if any), number of clusters $k$.
**Result**: Clusters $\{\mathscr{C}_1, \ldots, \mathscr{C}_k\}$, codebook $\mathscr{CB} = \{c_p\}_{p=1}^k$ with $\{c_p\} \in \{-1,1\}^{k-1}$.

1 compute the training eigenvectors $\boldsymbol{\alpha}^{(l)}$, $l = 1, \ldots, k-1$, corresponding to the $k-1$ largest eigenvalues of problem (7)
2 let $\mathbf{A} \in \mathbb{R}^{N_{\text{tr}} \times (k-1)}$ be the matrix containing the vectors $\boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(k-1)}$ as columns
3 binarize $\mathbf{A}$ and let the code-book $\mathscr{CB} = \{c_p\}_{p=1}^k$ be composed by the $k$ encodings of $\mathbf{Q} = \text{sign}(A)$ with the most occurrences
4 $\forall i$, $i = 1, \ldots, N_{\text{tr}}$, assign $\mathbf{x}_i$ to $A_{p^*}$ where $p^* = \text{argmin}_p d_H(\text{sign}(\boldsymbol{\alpha}_i), c_p)$ and $d_H(\cdot,\cdot)$ is the Hamming distance
5 binarize the test data projections $\text{sign}(\mathbf{e}_r^{(l)})$, $r = 1, \ldots, N_{\text{test}}$, and let $\text{sign}(\mathbf{e}_r) \in \{-1,1\}^{k-1}$ be the encoding vector of $\mathbf{x}_r^{\text{test}}$
6 $\forall r$, assign $\mathbf{x}_r^{\text{test}}$ to $A_{p^*}$, where $p^* = \text{argmin}_p d_H(\text{sign}(\mathbf{e}_r), c_p)$.

## *4.2 Fixed-size KSC approach*

When the number of training datapoints $N_{\text{tr}}$ is large, problem (7) can become intractable both in terms of memory bottleneck and execution time. A solution to this issue is offered by the fixed-size kernel spectral clustering (FSKSC) method where

[2] By choosing $\mathbf{V} = \mathbf{I}$, problem (7) represents a kernel PCA objective [39, 40, 41].

[3] http://www.esat.kuleuven.be/stadius/ADB/alzate/softwareKSClab.php

the primal problem instead of the dual is solved, as proposed in [17] in case of classification and regression. In particular, as discussed in [25], the FSKSC approach is based on the following unconstrained re-formulation of the KSC primal objective (5), where $\mathbf{V} = \mathbf{D}^{-1}$:

$$\min_{\hat{\mathbf{w}}^{(l)},\hat{b}_l} \quad \frac{1}{2}\sum_{l=1}^{k-1} \hat{\mathbf{w}}^{(l)^T}\hat{\mathbf{w}}^{(l)} - \frac{1}{2}\sum_{l=1}^{k-1} \gamma_l (\hat{\boldsymbol{\Phi}}\hat{\mathbf{w}}^{(l)} + \hat{b}_l \mathbf{1}_{N_{\text{tr}}})^T \hat{\boldsymbol{D}}^{-1} (\hat{\boldsymbol{\Phi}}\hat{\mathbf{w}}^{(l)} + \hat{b}_l \mathbf{1}_{N_{\text{tr}}}) \tag{9}$$

where $\hat{\boldsymbol{\Phi}} = [\hat{\varphi}(\mathbf{x}_1)^T; \ldots; \hat{\varphi}(\mathbf{x}_{N_{\text{tr}}})^T] \in \mathbb{R}^{N_{\text{tr}} \times m}$ is the approximated feature matrix, $\hat{\boldsymbol{D}} \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{tr}}}$ is the corresponding degree matrix, and $\hat{\varphi} : \mathbb{R}^d \rightarrow \mathbb{R}^m$ indicates a finite dimensional approximation of the feature[4]. map $\varphi(\cdot)$ which can be obtained through the Nyström method [44]. The minimizer of (9) can be found by computing $\nabla J(\mathbf{w}^l, b_l) = 0$, that is:

$$\begin{aligned} \frac{\partial \mathscr{J}}{\partial \hat{\mathbf{w}}^{(l)}} = 0 \quad &\rightarrow \quad \hat{\mathbf{w}}^{(l)} = \gamma_l (\hat{\boldsymbol{\Phi}}^T \hat{\boldsymbol{D}}^{-1} \hat{\boldsymbol{\Phi}} \hat{\mathbf{w}}^{(l)} + \hat{\boldsymbol{\Phi}}^T \hat{\boldsymbol{D}}^{-1} \mathbf{1}_{N_{\text{tr}}} \hat{b}_l) \\ \frac{\partial \mathscr{J}}{\partial \hat{b}_l} = 0 \quad &\rightarrow \quad \mathbf{1}_{N_{\text{tr}}}^T \hat{\boldsymbol{D}}^{-1} \hat{\boldsymbol{\Phi}} \hat{\mathbf{w}}^{(l)} = -\mathbf{1}_{N_{\text{tr}}}^T \hat{\boldsymbol{D}}^{-1} \mathbf{1}_{N_{\text{tr}}} \hat{b}_l. \end{aligned}$$

These optimality conditions lead to the following eigenvalue problem to solve in order to find the model parameters:

$$\mathbf{R}\hat{\mathbf{w}}^{(l)} = \hat{\lambda}_l \hat{\mathbf{w}}^{(l)} \tag{10}$$

with $\hat{\lambda}_l = \frac{1}{\gamma_l}$, $\mathbf{R} = \hat{\boldsymbol{\Phi}}^T \hat{\mathbf{D}}^{-1} \hat{\boldsymbol{\Phi}} - \frac{(\mathbf{1}_{N_{\text{tr}}}^T \hat{\mathbf{D}}^{-1} \hat{\boldsymbol{\Phi}})^T (\mathbf{1}_{N_{\text{tr}}}^T \hat{\mathbf{D}}^{-1} \hat{\boldsymbol{\Phi}})}{\mathbf{1}_{N_{\text{tr}}}^T \hat{\mathbf{D}}^{-1} \mathbf{1}_{N_{\text{tr}}}}$ and $\hat{b}_l = -\frac{\mathbf{1}_{N_{\text{tr}}}^T \hat{\mathbf{D}}^{-1} \hat{\boldsymbol{\Phi}}}{\mathbf{1}_{N_{\text{tr}}}^T \hat{\mathbf{D}}^{-1} \mathbf{1}_{N_{\text{tr}}}} \hat{\mathbf{w}}^{(l)}$. Notice that we now have to solve an eigenvalue problem of size $m \times m$, which can be done very efficiently by choosing $m$ such that $m \ll N_{\text{tr}}$. Furthermore, the diagonal of matrix $\hat{\mathbf{D}}$ can be calculated as $\hat{\mathbf{d}} = \hat{\boldsymbol{\Phi}}(\hat{\boldsymbol{\Phi}}^T \mathbf{1}_m)$, i.e. without constructing the full matrix $\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\Phi}}^T$.

Once $\hat{\mathbf{w}}^{(l)}, \hat{b}_l$ have been computed, the cluster memberships can be obtained by applying the k-means algorithm on the projections $\hat{e}_i^{(l)} = \hat{\mathbf{w}}^{(l)^T} \hat{\varphi}(\mathbf{x}_i) + \hat{b}_l$ for training data and $\hat{e}_r^{(l),\text{test}} = \hat{\mathbf{w}}^{(l)^T} \hat{\varphi}(\mathbf{x}_i^{\text{test}}) + \hat{b}_l$ in case of test points, as for the classical spectral clustering technique. The entire algorithm is depicted in Figure 2, and a Matlab implementation is freely available for download[5]. Finally, Figure 1 illustrates examples of clustering obtained in case of the *Iris*, *Dermatology* and *S1* datasets available at the UCI machine learning repository.

[4] The $m$ points needed to estimate the components of $\hat{\varphi}$ are selected at random

[5] http://www.esat.kuleuven.be/stadius/ADB/langone/softwareKSCFSlab.php

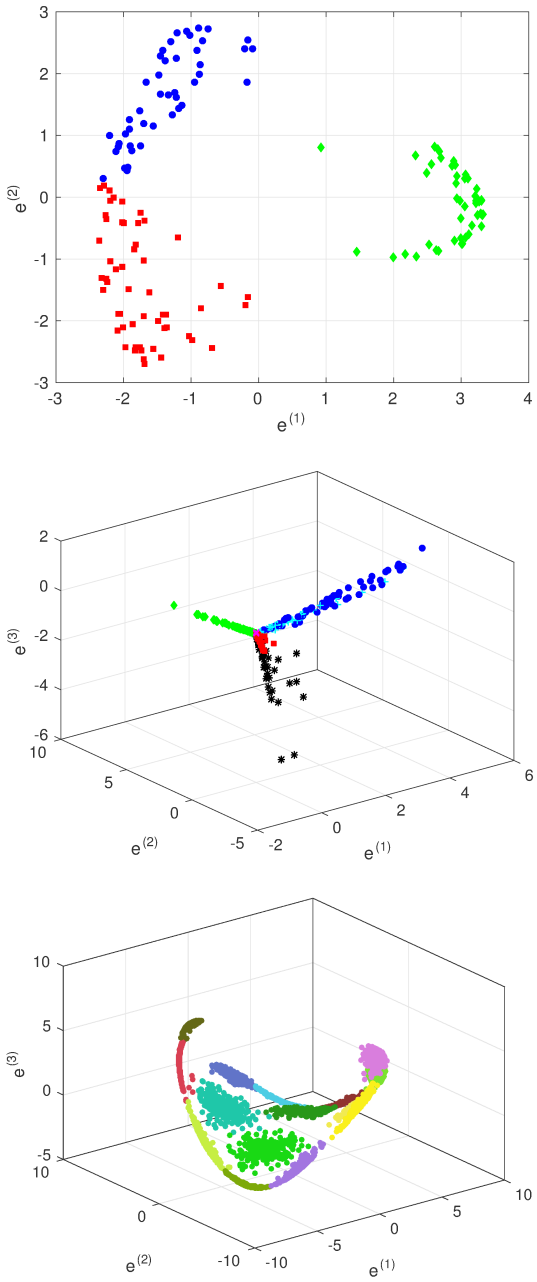



**Fig. 1 FSKSC embedding illustrative example**. Data points represented in the space of the projections in case of the Iris, Dermatology and S1 datasets. The different colors relate to the various clusters detected by the FSKSC algorithm.

**Algorithm 2:** Fixed-size KSC [25]

**Input** : training set $\mathscr{D} = \{\mathbf{x}_i\}_{i=1}^{N_{\text{tr}}}$, Test set $\mathscr{D}_{\text{test}} = \{\mathbf{x}_i\}_{r=1}^{N_{\text{test}}}$.
**Settings** : size Nyström subset $m$, kernel parameter $\sigma$, number of clusters $k$
**Output** : $\mathbf{q}$ and $\mathbf{q}_{\text{test}}$ vectors of predicted cluster memberships.

/* ***Approximate feature map:*** */
Compute $\boldsymbol{\Omega}_{m\times m}$
Compute [$\mathbf{U}$,$\boldsymbol{\Lambda}$] = SVD($\boldsymbol{\Omega}_{m\times m}$)
Compute $\hat{\boldsymbol{\Phi}}$ by means of the Nyström method

/* ***Training:*** */
Solve $\mathbf{R}\hat{\mathbf{w}}^{(l)} = \hat{\lambda}_l \hat{\mathbf{w}}^{(l)}$
Compute $\mathbf{E} = [\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{k-1}]$
[$\mathbf{q}$,$\mathbf{C}_{\text{tr}}$] = kmeans($\mathbf{E}$,k)

/* ***Test:*** */
Compute $\mathbf{E}_{\text{test}} = [\mathbf{e}_{\text{test}}^{(1)}, \ldots, \mathbf{e}_{\text{test}}^{k-1}]$
$\mathbf{q}_{\text{test}}$ = kmeans($\mathbf{E}_{\text{test}}$,k,'start',$\mathbf{C}_{\text{tr}}$)

### *4.3 Computational complexity*

The computational complexity of the fixed-size KSC algorithm depends mainly on the size $m$ of the Nyström subset used to construct the approximate feature map $\hat{\boldsymbol{\Phi}}$. In particular, the total time complexity (training + test) is approximately $O(m^3) + O(mN_{\text{tr}}) + O(mN_{\text{test}})$, which is the time needed to solve (10) and to compute the training and test clustering scores. Furthermore, the space complexity is $O(m^2) + O(mN_{\text{tr}}) + O(mN_{\text{test}})$, which is needed to construct matrix $\mathbf{R}$ and to build the training and test feature matrices $\hat{\boldsymbol{\Phi}}$ and $\hat{\boldsymbol{\Phi}}_{\text{test}}$. Since we can choose $m \ll N_{\text{tr}} < N_{\text{test}}$ [25], the complexity of the algorithm is approximately linear, as can be evinced also from Figure 6.

## 5 Regularized stochastic k-means

### *5.1 Related work*

The main drawbacks of the standard k-means algorithm are the instability caused by the randomness in the initialization and the presence of outliers, which can bias the computation of the cluster centroids and hence the final memberships. To stabilize the performance of the k-means algorithm [45] applies the stochastic learning paradigm relying on the probabilistic draw of some specific random variable dependent upon the distribution of per-sample distances to the centroids. In [21] one seeks to find a new cluster centroid by observing one or a small mini-batch sample at iterate $t$ and calculating the corresponding gradient descent step. Recent developments

[46, 47] indicate that the regularization with different norms might be useful when one deals with high-dimensional datasets and seeks for a sparse solution. In particular, [46] proposes to use an adaptive group Lasso penalty [48] and obtain a solution per prototype vector in a closed-form. In [49] the authors are studying the problem of overlapping clusters where there are possible outliers in data. They propose an objective function which can be viewed as a reformulation of the traditional k-means objective which captures also the degrees of overlap and non-exhaustiveness.

## 5.2 Generalities

Given a dataset $\mathscr{D} = \{\mathbf{x}_i\}_{i=1}^N$ with $N$ independent observations, the regularized k-means objective can be expressed as follows:

$$\min_{\boldsymbol{\mu}^{(1)},\ldots,\boldsymbol{\mu}^{(k)}} \sum_{l=1}^{k} [\frac{1}{2N_l} \sum_{\mathbf{x}\in\mathscr{S}_l} \|\boldsymbol{\mu}^{(l)} - \mathbf{x}\|_2^2 + C\psi(\boldsymbol{\mu}^{(l)})], \tag{11}$$

where $\psi(\boldsymbol{\mu}^{(l)})$ represents the regularizer, $C$ is the trade-off parameter, $N_l = |\mathscr{S}_l|$ is the cardinality of the corresponding set $\mathscr{S}_l$ corresponding to the $l$-th individual cluster. In a stochastic optimization paradigm objective eq.(11) can be optimized through gradient descent, meaning that one takes at any step $t$ some gradient $g_t \in \partial f(\boldsymbol{\mu}_t^{(l)})$ w.r.t. only one sample $\mathbf{x}_t$ from $\mathscr{S}_l$ and the current iterate $\boldsymbol{\mu}_t^{(l)}$ at hand. This online learning problem is usually terminated until some $\varepsilon$-tolerance criterion is met or the total number of iterations is exceeded. In the above setting one deals with a simple clustering model $c(\mathbf{x}) = \arg\min_l \|\boldsymbol{\mu}^{(l)} - \mathbf{x}\|_2$ and updates cluster memberships of the entire dataset $\hat{\mathscr{S}}$ after individual solutions $\boldsymbol{\mu}^{(l)}$, i.e. the centroids, are computed. From a practical point of view, we denote this update as an outer iteration or synchronization step and use it to fix $\mathscr{S}_l$ for learning each individual prototype vector $\boldsymbol{\mu}^{(l)}$ in parallel through a Map-Reduce scheme. This algorithmic procedure is depicted in Figure 2. As we can notice the Map-Reduce framework is needed to parallelize learning of individual prototype vectors using either the SGD-based approach or the adaptive dual averaging scheme. In each outer $p$-th iteration we `Reduce()` all learned centroids to the matrix $\mathbf{W}_p$ and re-partition the data again with `Map()`. After we reach $T_{out}$ iterations we stop and re-partition the data according to the final solution and proximity to the prototype vectors.

## 5.3 $l_2$-Regularization

In this section the Stochastic Gradient Descent (SGD) scheme for learning objective (11) with $\psi(\boldsymbol{\mu}^{(l)}) = \frac{1}{2}\|\boldsymbol{\mu}^{(l)}\|_2^2$ is presented. If we use the $l_2$ regularization, the optimization problem becomes:

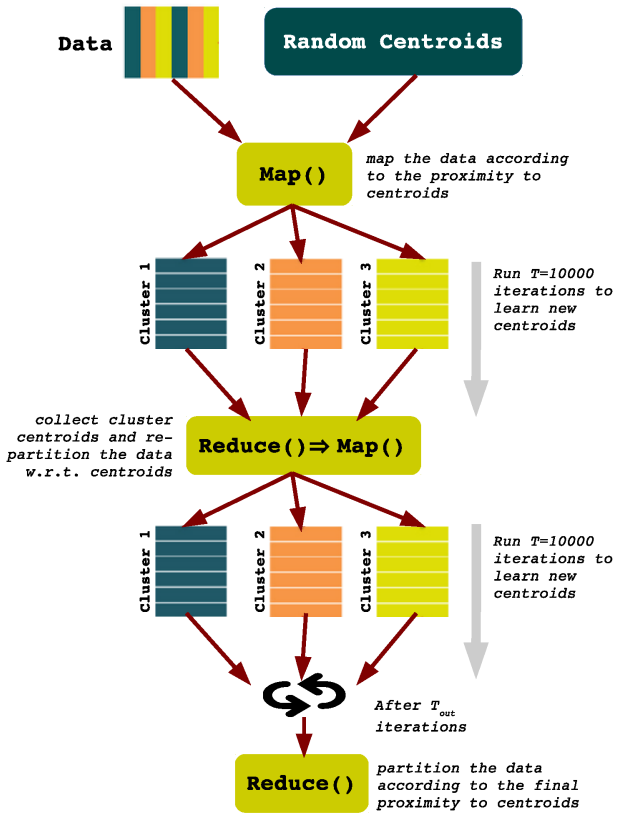



**Fig. 2** Schematic visualization of the Map-Reduce scheme.

$$\min_{\boldsymbol{\mu}^{(l)}} f(\boldsymbol{\mu}^{(l)}) \triangleq \frac{1}{2N}\sum_{j=1}^{N}\|\boldsymbol{\mu}^{(l)}-\mathbf{x}_j\|_2^2+\frac{C}{2}\|\boldsymbol{\mu}^{(l)}\|_2^2, \tag{12}$$

where function $f(\boldsymbol{\mu}^{(l)})$ is $\lambda$-strongly convex with Lipschitz continuous gradient and Lipschitz constant equal to $L$. It can be easily verified that $\lambda = L = C+1$ by observing basic inequalities which $f(\boldsymbol{\mu}^{(l)})$ should satisfy in this case [50, 51]:

$$\|\nabla f(\boldsymbol{\mu}^{(l)})-\nabla f(\boldsymbol{\mu}^{(l)})\|_2 \geq \lambda\|\boldsymbol{\mu}_1^{(l)}-\boldsymbol{\mu}_2^{(l)}\|_2 \Longrightarrow$$
$$\|(C+1)\boldsymbol{\mu}_1^{(l)}-(C+1)\boldsymbol{\mu}_2^{(l)}\|_2 \geq \lambda\|\boldsymbol{\mu}_1^{(l)}-\boldsymbol{\mu}_2^{(l)}\|_2$$

and

$$\|\nabla f(\boldsymbol{\mu}_1^{(l)}) - \nabla f(\boldsymbol{\mu}_2^{(l)})\|_2 \leq L\|\boldsymbol{\mu}_1^{(l)} - \boldsymbol{\mu}_2^{(l)}\|_2 \Longrightarrow$$
$$\|(C+1)\boldsymbol{\mu}_1^{(l)} - (C+1)\boldsymbol{\mu}_2^{(l)}\|_2 \leq L\|\boldsymbol{\mu}_1^{(l)} - \boldsymbol{\mu}_2^{(l)}\|_2$$

which can be satisfied if and only if $\lambda = L = C+1$. In this case a proper sequence of SGD step-sizes $\eta_t$ should be applied in order to achieve optimal convergence rate [52]. As a consequence, we set $\eta_t = \frac{1}{Ct}$ such that the convergence rate to the $\varepsilon$-optimal solution would be $\mathcal{O}(\frac{1}{T})$, being $T$ the total number of iterations, i.e. $1 \leq t \leq T$. This leads to a cheap, robust and stable to perturbation learning procedure with a fixed computational budget imposed on the total number of iterations and gradient re-computations needed to find a feasible solution.

The complete algorithm is illustrated in Algorithm 3. The first step is the initialization of a random matrix $\mathbf{M}_0$ of size $d \times k$, where $d$ is the input dimension and $k$ is the number of clusters. After initialization $T_{out}$ outer synchronization iterations are performed in which, based on previously learned individual prototype vectors $\boldsymbol{\mu}^{(l)}$, the cluster memberships and re-partition $\hat{\mathscr{S}}$ are calculated (line 4). Afterwards we run in parallel a basic SGD scheme for the $l_2$-regularized optimization objective (12) and concatenate the result with $\mathbf{M}_p$ by the `Append` function. When the total number of outer iterations $T_{out}$ is exceeded we exit with the final partitioning of $\hat{\mathscr{S}}$ by $c(x) = \arg\min_i \|\mathbf{M}_{T_{out}}^{(l)} - \mathbf{x}\|_2$ where $l$ denotes the $l$-th column of $\mathbf{M}_{T_{out}}$.

**Algorithm 3:** $l_2$-Regularized stochastic k-means

**Data**: $\hat{\mathscr{S}}, C > 0, T \geq 1, T_{out} \geq 1, k \geq 2, \varepsilon > 0$
1 Initialize $\mathbf{M}_0$ randomly for all clusters ($1 \leq l \leq k$)
2 **for** $p \leftarrow 1$ **to** $T_{out}$ **do**
3  Initialize empty matrix $\mathbf{M}_p$
4  Partition $\hat{\mathscr{S}}$ by $c(x) = \arg\min_l \|\mathbf{M}_{p-1}^{(l)} - \mathbf{x}\|_2$
5  **for** $\mathscr{S}_l \subset \hat{\mathscr{S}}$ ***in parallel*** **do**
6  Initialize $\boldsymbol{\mu}_0^{(l)}$ randomly
7  **for** $t \leftarrow 1$ **to** $T$ **do**
8  Draw a sample $\mathbf{x}_t \in \mathscr{S}_l$
9  Set $\eta_t = 1/(Ct)$
10  $\boldsymbol{\mu}_t^{(l)} = \boldsymbol{\mu}_{t-1}^{(l)} - \eta_t(C\boldsymbol{\mu}_{t-1}^{(l)} + \boldsymbol{\mu}_{t-1}^{(l)} - \mathbf{x}_t)$
11  **if** $\|\boldsymbol{\mu}_t^{(l)} - \boldsymbol{\mu}_{t-1}^{(l)}\|_2 \leq \varepsilon$ **then**
12  `Append` $(\boldsymbol{\mu}_t^{(l)}, \mathbf{M}_p)$
13  **return**
14  **end**
15  **end**
16  `Append` $(\boldsymbol{\mu}_T^{(l)}, \mathbf{M}_p)$
17  **end**
18 **end**
19 **return** $\hat{\mathscr{S}}$ *is partitioned by* $c(x) = \arg\min_l \|\mathbf{M}_{T_{out}}^{(l)} - \mathbf{x}\|_2$

### 5.4 $l_1$-Regularization

In this section we present a different learning scheme induced by $l_1$-norm regularization and corresponding regularized dual averaging methods [53] with adaptive primal-dual iterate updates [54]. The main optimization objective is given by [55]:

$$\min_{\boldsymbol{\mu}^{(l)}} f(\boldsymbol{\mu}^{(l)}) \triangleq \frac{1}{2N}\sum_{j=1}^{N}\|\boldsymbol{\mu}^{(l)}-\mathbf{x}_j\|_2^2 + C\|\boldsymbol{\mu}^{(l)}\|_1. \tag{13}$$

By using a *simple dual averaging* scheme [22] and adaptive strategy from [54] problem (13) can be solved effectively by the following sequence of iterates $\boldsymbol{\mu}_{t+1}^{(l)}$ :

$$\boldsymbol{\mu}_{t+1}^{(l)} = \arg\min_{\boldsymbol{\mu}^{(l)}}\{\frac{\eta}{t}\sum_{\tau=1}^{t}\langle g_\tau, \boldsymbol{\mu}^{(l)}\rangle + \eta C\|\boldsymbol{\mu}^{(l)}\|_1 + \frac{1}{t}h(\boldsymbol{\mu}^{(l)})\}, \tag{14}$$

where $h_t(\boldsymbol{\mu}^{(l)})$ is an adaptive strongly convex proximal term, $g_t$ represents a gradient of the $\|\boldsymbol{\mu}^{(l)}-\mathbf{x}_t\|^2$ term w.r.t. only one randomly drawn sample $\mathbf{x}_t \in \mathscr{S}_l$ and current iterate $\boldsymbol{\mu}_t^{(l)}$, while $\eta$ is a fixed step-size. In the regularized Adaptive Dual Averaging (ADA) scheme [54] one is interested in finding a corresponding step-size for each coordinate which is inversely proportional to the time-based norm of the coordinate in the sequence $\{g_t\}_{t\geq 1}$ of gradients. In case of our algorithm, the coordinate-wise update of the $\boldsymbol{\mu}_t^{(l)}$ iterate in the adaptive dual averaging scheme can be summarized as follows:

$$\boldsymbol{\mu}_{t+1,q}^{(l)} = \text{sign}(-\hat{g}_{t,q})\frac{\eta t}{H_{t,qq}}[|\hat{g}_{t,q}|-\lambda]_+, \tag{15}$$

where $\hat{g}_{t,q} = \frac{1}{t}\sum_{\tau=1}^{t} g_{\tau,q}$ is the coordinate-wise mean across $\{g_t\}_{t\geq 1}$ sequence, $H_{t,qq} = \rho + \|g_{1:t,q}\|_2$ is the time-based norm of the $q$-th coordinate across the same sequence and $[x]_+ = \max(0,x)$. In eq. (15) two important parameters are present: $C$ which controls the importance of the $l_1$-norm regularization and $\eta$ which is necessary for the proper convergence of the entire sequence of $\boldsymbol{\mu}_t^{(l)}$ iterates.

An outline of our distributed stochastic $l_1$-regularized k-means algorithm is depicted in Algorithm 4. Compared to the $l_2$ regularization, the iterate $\boldsymbol{\mu}_t^{(l)}$ now has a closed form solution and depends on the dual average (and the sequence of gradients $\{g_t\}_{t\geq 1}$). Another important difference is the presence of some additional parameters: the fixed step-size $\eta$ and the additive constant $\rho$ for making $H_{t,qq}$ term non-zero. These additional degrees of freedom might be beneficial from the generalization perspective. However, an increased computational cost has to be expected due to the cross-validation needed for their selection. Both versions of the regularized stochastic k-means method presented in sections 5.3 and 5.4 are available for download[6].

[6] http://www.esat.kuleuven.be/stadius/ADB/jumutc/softwareSALSA.php

**Algorithm 4:** $l_1$-Regularized stochastic k-means [55]

**Data**: $\hat{\mathscr{S}}, C > 0, \eta > 0, \rho > 0, T \geq 1, T_{out} \geq 1, k \geq 2, \varepsilon > 0$
Initialize $\mathbf{M}_0$ randomly for all clusters ($1 \leq l \leq k$)
**for** $p \leftarrow 1$ **to** $T_{out}$ **do**
  Initialize empty matrix $\mathbf{M}_p$
  Partition $\hat{\mathscr{S}}$ by $c(x) = \arg\min_l \|\mathbf{M}_{p-1}^{(l)} - \mathbf{x}\|_2$
  **for** $\mathscr{S}_l \subset \hat{\mathscr{S}}$ ***in parallel*** **do**
    Initialize $\boldsymbol{\mu}_1^{(l)}$ randomly, $\hat{g}_0 = 0$
    **for** $t \leftarrow 1$ **to** $T$ **do**
      Draw a sample $\mathbf{x}_t \in \mathscr{S}_l$
      Calculate gradient $g_t = \boldsymbol{\mu}_t^{(l)} - \mathbf{x}_t$
      Find the average $\hat{g}_t = \frac{t-1}{t}\hat{g}_{t-1} + \frac{1}{t}g_t$
      Calculate $H_{t,qq} = \rho + \|g_{1:t,q}\|_2$
      $\boldsymbol{\mu}_{t+1,q}^{(l)} = \text{sign}(-\hat{g}_{t,q})\frac{\eta_t}{H_{t,qq}}[|\hat{g}_{t,q}| - C]_+$
      **if** $\|\boldsymbol{\mu}_t^{(l)} - \boldsymbol{\mu}_{t+1}^{(l)}\|_2 \leq \varepsilon$ **then**
        `Append` $(\boldsymbol{\mu}_{t+1}^{(l)}, \mathbf{M}_p)$
        **return**
      **end**
    **end**
    `Append` $(\boldsymbol{\mu}_{T+1}^{(l)}, \mathbf{M}_p)$
  **end**
**end**
**return** $\hat{\mathscr{S}}$ *is partitioned by* $c(x) = \arg\min_l \|\mathbf{M}_{T_{out}}^{(l)} - \mathbf{x}\|_2$

## 5.5 Influence of Outliers

Thanks to the regularization terms that have been added to the K-means objective in Eqs. (13) and (12), the regularized stochastic k-means becomes less sensitive to the influence of the outliers. Furthermore, the stochastic optimization schemes allow to reduce also the sensitivity to the initialization. In order to illustrate this aspects, a synthetic dataset consisting of three Gaussian clouds corrupted by outliers is used as benchmark. As shown in Figure 3, while K-means can fail to recover the true cluster centroids and, as a consequence, produces a wrong partitioning, the regularized schemes are always able to correctly identify the three clouds of points.

## 5.6 Theoretical Guarantees

In this section a theoretical analysis of the algorithms described previously is discussed. In case of the $l_2$-norm, two results in expectation obtained by [52] for smooth and strongly convex functions are properly reformulated. Regarding the $l_1$-norm, our

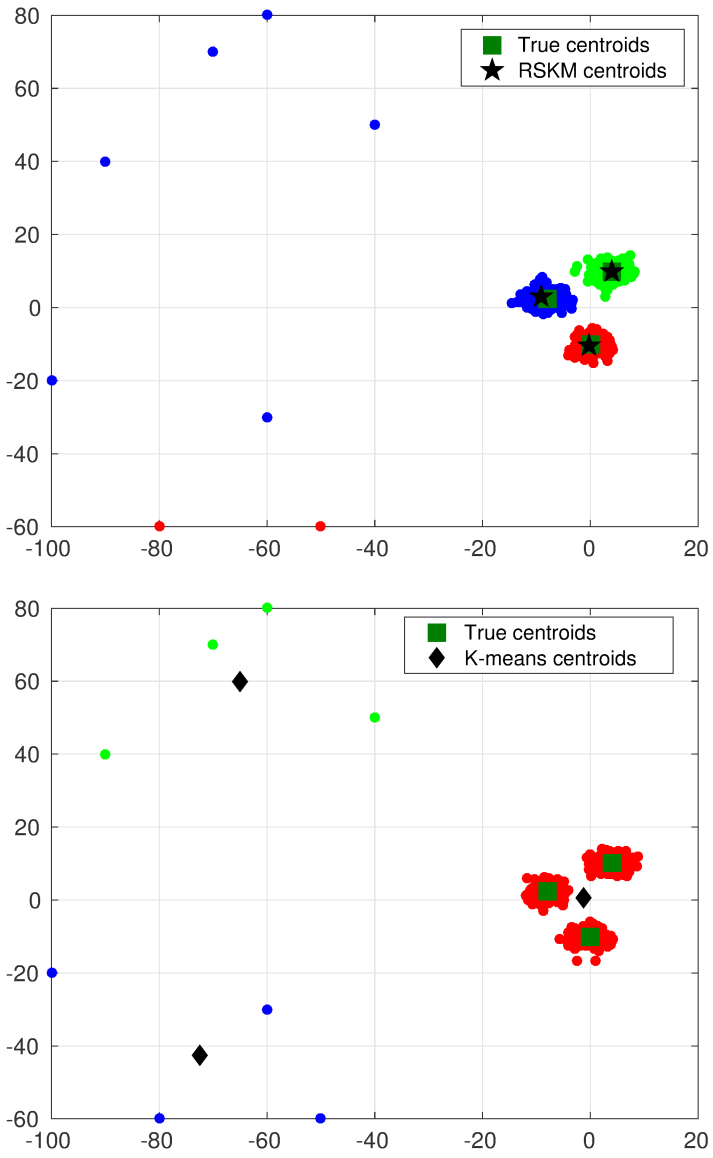



**Fig. 3 Influence of outliers**. **(Top)** K-means clustering of a synthetic dataset with three clusters corrupted by outliers. **(Bottom)** In this case RSKM is insensitive to the outliers and allows to perfectly detect the three Gaussians, while K-means only yields a reasonable result 4 times out of 10 runs.

theoretical results are stemmed directly from various lemmas and corollaries related to the adaptive subgradient method presented in [54].

### 5.6.1 $l_2$-norm

As it was shown in Section 5.3 the $l_2$-regularized k-means objective (12) is a smooth strongly convex function with Lipschitz continuous gradient. Based on this, an upper bound on $f(\boldsymbol{\mu}_T^{(l)}) - f(\boldsymbol{\mu}_*^{(l)})$ in expectation can be derived, where $\boldsymbol{\mu}_*^{(l)}$ denotes the optimal center for the $l$-th cluster, where $l = 1, \ldots, k$.

**Theorem 1.** *Consider strongly convex function $f(\boldsymbol{\mu}^{(l)})$ in eq.(12) which is $\nu$-smooth with respect to $\boldsymbol{\mu}_*^{(l)}$ over the convex set $\mathscr{W}$. Suppose that $\mathbb{E}\|\hat{g}_t\|^2 \leq G^2$. Then if we take any $C > 0$ and pick the step-size $\eta = \frac{1}{C}t$, it holds for any $T$ that:*

$$\mathbb{E}[f(\boldsymbol{\mu}_T^{(l)}) - f(\boldsymbol{\mu}_*^{(l)})] \leq \frac{2G^2}{(C+1)T}. \tag{16}$$

*Proof.* This result follows directly from Theorem 1 in [52] where the $\nu$-smoothness is defined as $f(\boldsymbol{\mu}^{(l)}) - f(\boldsymbol{\mu}_*^{(l)}) \leq \frac{\nu}{2}\|\boldsymbol{\mu}^{(l)} - \boldsymbol{\mu}_*^{(l)}\|$. From the theory of convex optimization we know that this inequality is a particular case of a more general inequality for functions with Lipschitz continuous gradients. From Section 5.3 we know that our Lipschitz constant is $L = C + 1$. Plugging the already known constants into the aforementioned Theorem 1 completes our proof.

Furthermore, an upper bound on $\|\boldsymbol{\mu}_T - \boldsymbol{\mu}_*\|^2$ in expectation can be obtained:

**Theorem 2.** *Consider strongly convex function $f(\boldsymbol{\mu})$ in eq.(12) over the convex set $\mathscr{W}$. Suppose that $\mathbb{E}\|\hat{g}_t\|^2 \leq G^2$. Then if we take any $C > 0$ and pick the step-size $\eta = \frac{1}{C}t$, it holds for any $T$ that:*

$$\mathbb{E}[\|\boldsymbol{\mu}_T - \boldsymbol{\mu}_*\|^2)] \leq \frac{4G^2}{(C+1)^2T}. \tag{17}$$

*Proof.* This result directly follows from Lemma 1 in [52] if we take into account that $f(\boldsymbol{\mu})$ is $(C+1)$-strongly convex.

### 5.6.2 $l_1$-norm

First consider the following implication of Lemma 4 in [54] over the running subgradient $g_t = \boldsymbol{\mu}_t^{(l)} - \boldsymbol{x}_t$ of the first term in the optimization objective defined in Eq.(13):

$$\sum_{t=1}^{T} \|f_t'(\boldsymbol{\mu}_t^{(l)})\|^2 \leq 2\sum_{l=1}^{d} \|g_{1:T,q}\|_2. \tag{18}$$

where $\|g_{1:T,q}\|_2$ is the time-based norm of the q-th coordinate. Here we can see a direct link to some of our previously presented results in Theorem 2 where we operate over the bounds of iterate specific subgradients.

**Theorem 3.** *By defining the following infinity norm* $D_\infty = \sup_{\boldsymbol{\mu}^{(l)} \in \boldsymbol{M}} \|\boldsymbol{\mu}^{(l)} - \boldsymbol{\mu}_*^{(l)}\|_\infty$ *w.r.t. the optimal solution* $\boldsymbol{\mu}_*^{(l)}$*, setting the learning rate* $\eta = D_\infty/\sqrt{2}$ *and applying update steps to* $\boldsymbol{\mu}_t^{(l)}$ *in Algorithm 4 we get:*

$$\mathbb{E}_{t\in\{1\ldots T\}}[f(\boldsymbol{\mu}_t^{(l)}) - f(\boldsymbol{\mu}_*^{(l)})] \leq \frac{\sqrt{2}D_\infty}{T}\sum_{l=1}^{d}\|g_{1:T,l}\|_2. \tag{19}$$

*Proof.* Our result directly follows from Corollary 6 in [54] and averaging the regret term $R_\phi(T)$ (defining an expectation over the running index $t$) *w.r.t.* the optimal solution $f(\boldsymbol{\mu}_*^{(l)})$.

Our bounds imply faster convergence rates than non-adaptive algorithms on sparse data, though this depends on the geometry of the underlying optimization space of $\boldsymbol{M}$.

## 6 Experiments

In this Section a number of large-scale clustering algorithms are compared in terms of accuracy and execution time. The methods that are analyzed are: fixed-size kernel spectral clustering (FSKSC), regularized stochastic k-means (RSKM), parallel plane clustering [56] (PPC), parallel k-means [9] (PKM). The datasets used in the experiments are listed in Table 1 and mainly comprise databases available at the UCI repository [57]. Although they relate to classification problems, in view of the cluster assumption [58] [7] they can also be used to evaluate the performance of clustering algorithms (in this case the labels play the role of the ground-truth).

The clustering quality is measured by means of two quality metrics, namely the Davies-Bouldin (DB) [59] criterion and the adjusted rand index (ARI [60]). The first quantifies the separation between each pair of clusters (in terms of between cluster scatter) and, within each cluster, how tightly grouped all the data are (in terms of within cluster scatter). The ARI index measures the agreement between two partitions and is used to assess the correlation between the outcome of a clustering algorithm and the available ground-truth.

All the simulations are performed on an eight cores desktop PC in *Julia* [8], which is a high-level dynamic programming language that provides a sophisticated compiler and an intuitive distributed parallel execution.

[7] The cluster assumption states that if points are in the same cluster they are likely to be of the same class.

[8] http://julialang.org/

**Table 1** Datasets

| Size | Dataset | N | d |
|---|---|---|---|
| Small | Iris | 150 | 4 |
| | Ecoli | 336 | 8 |
| | Libras | 360 | 91 |
| | Dermatology | 366 | 33 |
| | Vowel | 528 | 10 |
| | Spambase | 4601 | 57 |
| | S1 | 5000 | 2 |
| | S2 | 5000 | 2 |
| | S3 | 5000 | 2 |
| | S4 | 5000 | 2 |
| Medium | Opt Digits | 5620 | 64 |
| | Pen Digits | 10992 | 16 |
| | Magic | 19020 | 11 |
| | RCV1 | 20242 | 1960 |
| | Shuttle | 58000 | 9 |
| Large | Skin | 245057 | 3 |
| | Covertype | 581012 | 54 |
| | GalaxyZoo | 667944 | 9 |
| | Poker | 1025010 | 10 |
| | Susy | 5000000 | 18 |
| | Higgs | 11000000 | 28 |

The selection of the tuning parameters has been done as follows. For all the methods the number of clusters $k$ has been set equal to the number of classes and the tuning parameters are selected by means of the within cluster sum of squares or WCSS criterion [61]. WCSS quantifies the compactness of the clusters in terms of sum of squared distances of each point in a cluster to the cluster center, averaged over all the clusters: the lower the index, the better (i.e. the higher the compactness). Concerning the FSKSC algorithm, the Gaussian kernel defined as $k(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\frac{||\mathbf{x}_i-\mathbf{x}_j||^2}{\sigma^{l^2}})$ is used to induce the nonlinear mapping. In this case, WCSS allows to select an optimal bandwidth $\sigma$ as shown at the top side of Figure 4 for the S1 dataset. Furthermore, the Nyström subset size has been set to $m = 100$ in case of the small datasets and $m = 150$ for the medium and large databases. This setting has been empirically found to represent a good choice, as illustrated at the bottom of Figure 4 for the S1 dataset. Also in case of RSKM and PPC the regularization parameter $C$ is found as the value yielding the minimum WCSS. An example of such a tuning procedure is depicted in Figure 5 in case of a toy dataset consisting of a Gaussian mixture with three components surrounded by outliers.

Table 2 reports the results of the simulations, where the best performance over 20 runs is indicated. While the regularized stochastic k-means and the parallel k-means approaches perform better in terms of adjusted rand index, the fixed-size kernel spectral clustering achieves the best results as measured by the Davies-Bouldin cri-

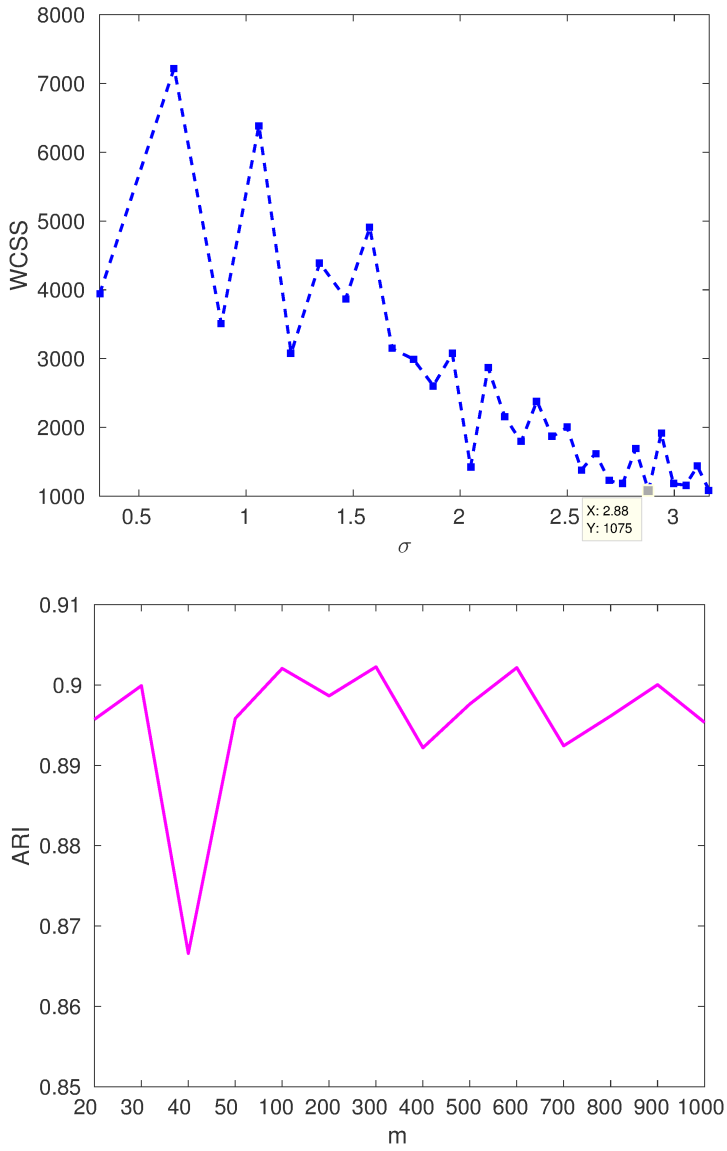



**Fig. 4 FSKSC parameters selection**. **(Top)** Tuning of the Gaussian kernel bandwidth $\sigma$ **(Bottom)** Change of the cluster performance (median ARI over 30 runs) with respect to the Nyström subset size $m$. The simulations refer to the S1 dataset.

terion. The computational efficiency of the methods is compared in Figure 6, from which it is evident that parallel k-means has the lowest runtime.

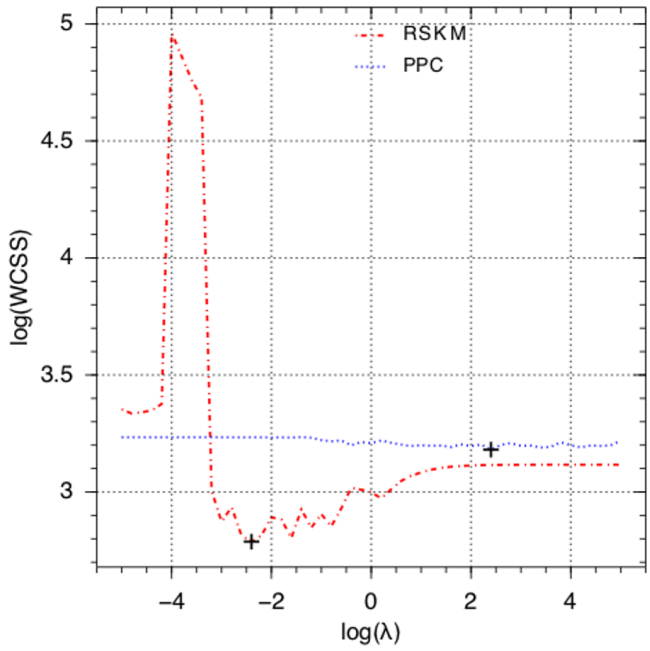



**Fig. 5 RSKM and PPC parameters selection**. Tuning of the regularization parameter for RSKM and PPC approaches by means of the WCSS criterion, concerning the toy dataset shown in Figure 3. In this case RSKM is insensitive to the outliers and allows to perfectly detect the three Gaussians (ARI = 0.99), while the best performance reached by the PPC method is ARI = 0.60.

## 7 Conclusions

In this chapter we have revised two large-scale clustering algorithms, namely regularized stochastic k-means (RSKM) and fixed-size kernel spectral clustering (FSKSC). The first learns in parallel the cluster prototypes by means of stochastic optimization schemes implemented through MapReduce, while the second relies on the Nyström method to speed-up a kernel-based formulation of spectral clustering known as kernel spectral clustering. These approaches are benchmarked on real-life datasets of different sizes. The experimental results show their competitiveness both in terms of runtime and cluster quality compared to other state-of-the-art clustering algorithms such as parallel k-means and parallel plane clustering.

**Acknowledgements** EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923). This chapter reflects only the authors' views, the Union is not liable for any use that may be made of the contained information. Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants. Flem-

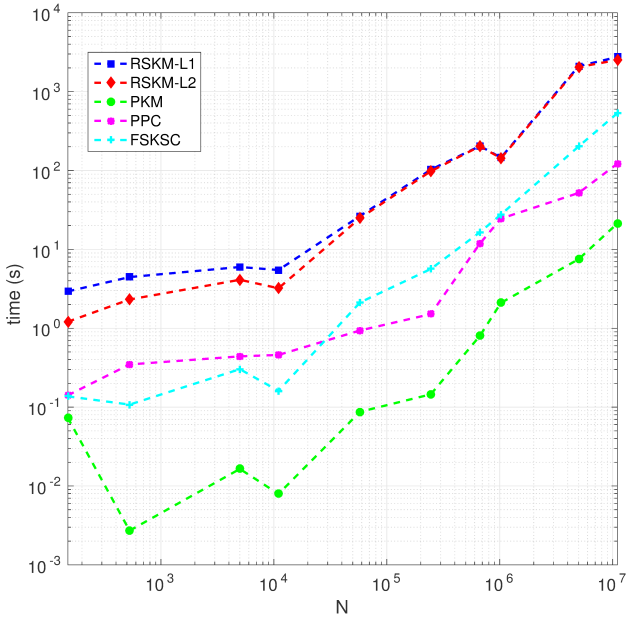



**Fig. 6 Efficiency evaluation**. Runtime of FSKSC (train + test), RSKM with l1 and l2 regularization, parallel k-means and PPC algorithms related to the following datasets: Iris, Vowel, S1, Pen Digits, Shuttle, Skin, Gzoo, Poker, Susy, Higgs. 2.

| **Dataset** | **RSKM-L1** | | **RSKM-L2** | | **PKM** | | **PPC** | | **FSKSC** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **DB** | **ARI** | **DB** | **ARI** | **DB** | **ARI** | **DB** | **ARI** | **DB** | **ARI** |
| Iris | 0.834 | 0.653 | 0.834 | 0.641 | 0.663 | 0.645 | 0.872 | **0.758** | **0.000** | 0.744 |
| Ecoli | **0.865** | 0.722 | 0.960 | **0.727** | 1.032 | 0.704 | 3.377 | 0.430 | 1.063 | 0.679 |
| Dermatology | 1.188 | 0.891 | 1.557 | **0.903** | 1.479 | 0.891 | 1.619 | 0.669 | **1.252** | 0.829 |
| Vowel | 1.553 | 0.121 | 1.512 | 0.127 | 1.544 | **0.133** | 4.846 | 0.060 | **0.000** | 0.109 |
| Libras | **1.231** | **0.357** | 1.234 | 0.338 | 1.288 | 0.327 | 4.900 | 0.109 | 1.352 | 0.298 |
| Pen digits | 1.381 | **0.638** | 1.262 | 0.624 | 1.352 | 0.634 | 4.024 | 0.257 | **0.000** | 0.546 |
| Opt Digits | 1.932 | 0.559 | 1.991 | 0.571 | 1.967 | **0.593** | 6.315 | 0.139 | **1.140** | 0.512 |
| Spambase | 3.527 | **0.549** | 0.650 | 0.532 | **0.613** | 0.517 | 1.244 | 0.288 | 3.871 | 0.382 |
| S1 | 0.367 | 0.981 | 0.509 | 0.903 | **0.365** | **0.986** | 1.581 | 0.551 | 0.366 | 0.984 |
| S2 | 0.478 | 0.905 | 0.571 | 0.867 | **0.465** | **0.938** | 2.051 | 0.420 | 0.518 | 0.876 |
| S3 | 0.686 | 0.681 | 0.689 | 0.679 | 0.644 | **0.725** | 3.360 | 0.395 | **0.000** | 0.648 |
| S4 | 0.663 | 0.602 | 0.665 | 0.598 | 0.657 | **0.632** | 4.158 | 0.305 | **0.000** | 0.607 |
| Magic | 1.430 | 0.005 | 1.435 | 0.008 | 1.439 | 0.006 | 1.424 | 0.002 | **0.313** | **0.089** |
| Shuttle | 0.775 | 0.395 | 0.733 | 0.396 | 0.730 | 0.395 | 1.295 | **0.468** | **0.000** | 0.440 |
| Skin | 0.695 | -0.020 | 0.696 | -0.028 | 0.699 | -0.030 | 0.811 | **-0.005** | **0.687** | -0.027 |
| Gzoo | 0.989 | **0.302** | 0.988 | 0.297 | **0.981** | 0.295 | 1.420 | 0.253 | 1.693 | 0.255 |
| Covertype | 2.399 | 0.107 | 2.450 | 0.099 | 1.825 | **0.120** | 5.600 | 0.059 | **0.000** | 0.095 |
| Poker | 2.193 | **0.001** | 2.190 | 0.000 | 2.204 | 0.000 | 17.605 | 0.000 | **0.000** | 0.000 |
| Susy | 2.08 | **0.12** | 2.10 | **0.12** | **2.07** | 0.11 | 3.46 | 0.09 | 2.17 | **0.12** |
| Higgs | **2.68** | 0.007 | 2.84 | 0.007 | **2.68** | 0.006 | 3.50 | 0.002 | 3.34 | **0.008** |

**Table 2 Clustering performance**. Comparison of the RSKM and FSKSC approaches against k-means (KM) and PPC algorithms in terms of ARI (the higher the better) and DB (the lower the better). The results indicate the best performance over 20 runs.

ish Government: FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants. IWT: projects: SBO POM (100031); PhD/Postdoc grants. iMinds Medical Information Technologies SBO 2014. Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017.)

## References

[1] H. Tong and U. Kang, "Big data clustering," in *Data Clustering: Algorithms and Applications*, 2013, pp. 259–276.

[2] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *IEEE Trans. on Knowl. and Data Eng.*, vol. 14, no. 5, pp. 1003–1016, 2002.

[3] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases," in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 1996, pp. 103–114.

[4] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," *SIGMOD Rec.*, vol. 27, no. 2, pp. 73–84, 1998.

[5] C. Boutsidis, A. Zouzias, and P. Drineas, "Random projections for k-means clustering," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 298–306.

[6] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos, "Colibri: Fast mining of large static and dynamic graphs," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 686–694.

[7] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[8] G. Karypis and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *J. Parallel Distrib. Comput.*, vol. 48, no. 1, pp. 96–129, 1998.

[9] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Proceedings of the 1st International Conference on Cloud Computing*, 2009, pp. 674–679.

[10] S. Papadimitriou and J. Sun, "Disco: Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 512–521.

[11] A. F. et al., "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE Transactions on Emerging Topics In Computing*, vol. 2, no. 3, pp. 267–279, 2014.

[12] A. M. et al., "Iterative big data clustering algorithms: a review," *Journal of Software: practice and experience*, vol. 46, no. 1, pp. 107–129, 2016.

[13] F. R. K. Chung, *Spectral Graph Theory*, 1997.

[14] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *NIPS*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., Cambridge, MA, 2002, pp. 849–856.

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

[16] H. Jia, S. Ding, X. Xu, and R. Nie, "The latest research progress on spectral clustering," *Neural Computing and Applications*, vol. 24, no. 7-8, pp. 1477–1486, 2014.

[17] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[18] R. Langone, R. Mall, C. Alzate, and J. A. K. Suykens, *Unsupervised Learning Algorithms*. Springer International Publishing, 2016, ch. Kernel Spectral Clustering and Applications, pp. 135–161.

[19] C. Alzate and J. A. K. Suykens, "Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 335–347, February 2010.

[20] C. Baker, *The numerical treatment of integral equations*. Clarendon Press, Oxford, 1977.

[21] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT 2010)*, Y. Lechevallier and G. Saporta, Eds. Paris, France: Springer, Aug. 2010, pp. 177–187.

[22] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical Programming*, vol. 120, no. 1, pp. 221–259, 2009.

[23] M. Meila and J. Shi, "A random walks view of spectral segmentation," in *Artificial Intelligence and Statistics AISTATS*, 2001.

[24] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1985, pp. 193–218.

[25] R. Langone, R. Mall, V. Jumutc, and J. A. K. Suykens, "Fast in-memory spectral clustering using a fixed-size approach," in *Proceedings of the European Symposium on Artitficial Neural Networks (ESANN)*, 2016, pp. 557–562.

[26] F. Lin and W. W. Cohen, "Power iteration clustering," in *International Conference on Machine Learning*, 2010, pp. 655–662.

[27] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 214–225, Feb. 2004.

[28] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. S. Huang, "Incremental spectral clustering with application to monitoring of evolving blog communities." in *SIAM International Conference on Data Mining*, 2007, pp. 261–272.

[29] A. M. Bagirov, B. Ordin, G. Ozturk, and A. E. Xavier, "An incremental clustering algorithm based on hyperbolic smoothing," *Computational Optimization and Applications*, vol. 61, no. 1, pp. 219–241, 2014.

[30] R. Langone, O. M. Agudelo, B. De Moor, and J. A. K. Suykens, "Incremental kernel spectral clustering for online learning of non-stationary data," *Neurocomputing*, vol. 139, no. 0, pp. 246–260, September 2014.

[31] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Chang, "Parallel spectral clustering in distributed systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 568–586, March 2011.

[32] C. Alzate and J. A. K. Suykens, "Sparse kernel models for spectral clustering using the incomplete Cholesky decomposition," in *Proc. of the 2008 International Joint Conference on Neural Networks (IJCNN 2008)*, 2008, pp. 3555–3562.

[33] K. Frederix and M. Van Barel, "Sparse spectral clustering method based on the incomplete cholesky decomposition," *J. Comput. Appl. Math.*, vol. 237, no. 1, pp. 145–161, Jan. 2013.

[34] M. Novak, C. Alzate, R. Langone, and J. A. K. Suykens, "Fast kernel spectral clustering based on incomplete Cholesky factorization for large scale data analysis," *Internal Report 14-119, ESAT-SISTA, KU Leuven (Leuven, Belgium)*, pp. 1–44, 2014.

[35] X. Chen and D. Cai, "Large scale spectral clustering with landmark-based representation," in *AAAI Conference on Artificial Intelligence*, 2011.

[36] D. Luo, C. Ding, H. Huang, and F. Nie, "Consensus spectral clustering in near-linear time," in *International Conference on Data Engineering*, 2011, pp. 1079–1090.

[37] K. Taşdemir, "Vector quantization based approximate spectral clustering of large datasets," *Pattern Recognition*, vol. 45, no. 8, pp. 3034–3044, 2012.

[38] L. Wang, C. Leckie, R. Kotagiri, and J. Bezdek, "Approximate pairwise clustering for large data sets via sampling plus extension," *Pattern Recognition*, vol. 44, no. 2, pp. 222–235, 2011.

[39] J. A. K. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor, "A support vector machine formulation to PCA analysis and its kernel version," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 447–450, Mar. 2003.

[40] B. Schölkopf, A. J. Smola, and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[41] S. Mika, B. Schölkopf, A. J. Smola, K. R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Advances in Neural Information Processing Systems 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds. MIT Press, 1999.

[42] M. Meila and J. Shi, "Learning segmentation by random walks," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001.

[43] J. C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proceedings of the National Academy of Sciences*, vol. 107, no. 29, pp. 12 755–12 760, Jul. 2010.

[44] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, 2001.

[45] B. Kvesi, J.-M. Boucher, and S. Saoudi, "Stochastic k-means algorithm for vector quantization." *Pattern Recognition Letters*, vol. 22, no. 6/7, pp. 603–610, 2001.

[46] W. Sun and J. Wang, "Regularized k-means clustering of high-dimensional data and its asymptotic consistency," *Electronic Journal of Statistics*, vol. 6, pp. 148–167, 2012.

[47] D. M. Witten and R. Tibshirani, "A framework for feature selection in clustering," *Journal of the American Statistical Association*, vol. 105, no. 490, pp. 713–726, Jun. 2010.

[48] F. Bach, R. Jenatton, and J. Mairal, *Optimization with Sparsity-Inducing Penalties (Foundations and Trends in Machine Learning)*. Hanover, MA, USA: Now Publishers Inc., 2011.

[49] J. Whang, I. S. Dhillon, and D. Gleich, "Non-exhaustive, overlapping k-means," in *SIAM International Conference on Data Mining (SDM)*, 2015, pp. 936–944.

[50] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[51] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization)*, 1st ed. Springer Netherlands.

[52] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization." in *ICML*. icml.cc / Omnipress, 2012.

[53] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *J. Mach. Learn. Res.*, vol. 11, pp. 2543–2596, Dec. 2010.

[54] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011.

[55] V. Jumutc, R. Langone, and J. A. K. Suykens, "Regularized and sparse stochastic k-means for distributed large-scale clustering," in *IEEE International Conference on Big Data*, 2015, pp. 2535–2540.

[56] Y.-H. Shao, L. Bai, Z. Wang, X.-Y. Hua, and N.-Y. Deng, "Proximal plane clustering via eigenvalues," ser. Procedia Computer Science, vol. 17, 2013, pp. 41–47.

[57] A. Frank and A. Asuncion, "UCI machine learning repository, http://archive.ics.uci.edu/ml,," 2010.

[58] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[59] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 224–227, April 1979.

[60] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, pp. 193–218, 1985.

[61] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, pp. 107–145, 2001.